DEMAND-BASED WIRELESS NETWORK DESIGN BY TEST POINT REDUCTION

Ph. D. Dissertation Defense

by

Natthapol Pongthaipat

B.E. in Electrical Engineering, Chulalongkorn University, 1997

M.S. in Interdisciplinary Telecommunications, University of Colorado at Boulder, 2000

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2007

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Natthapol Pongthaipat

It was defended on

November 9, 2007

and approved by

Dissertation Director: Dr. Joseph Kabara, Assistant Professor, DIST

Dr. Richard Thompson, Director and Professor, DIST

Dr. Prashant Krishnamurthy, Associate Professor, DIST

Dr. Marwan Simaan, Professor, ECE

Dr. Michael McCloud, Assistant Professor, ECE

DEMAND-BASED WIRELESS NETWORK DESIGN BY TEST POINT REDUCTION

Natthapol Pongthaipat

University of Pittsburgh, 2007

The problem of locating the minimum number of Base Stations (BSs) to provide sufficient signal coverage and data rate capacity is often formulated in manner that results in a mixed-integer NP-Hard (Non-deterministic Polynomial-time Hard) problem. Solving a large size NP-Hard problem is time-prohibitive because the search space always increases exponentially, in this case as a function of the number of BSs. This research presents a method to generate a set of Test Points (TPs) for BS locations, which always includes optimal solution(s). A sweep and merge algorithm then reduces the number of TPs, while maintaining the optimal solution. The coverage solution is computed by applying the minimum branching algorithm, which is similar to the branch and bound search. Data Rate demand is assigned to BSs in such a way to maximize the total network capacity. An algorithm based on Tabu Search to place additional BSs is developed to place additional BSs, in cases when the coverage solution can not meet the capacity requirement. Results show that the design algorithm efficiently searches the space and converges to the optimal solution in a computationally efficient manner. Using the demand nodes to represent traffic, network design with the TP reduction algorithm supports both voice and data users.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to especially thank and express deep appreciation to, Dr. Joseph Kabara, who has always been my advisor, a mentor, and one-of-a-kind teacher for years that I have been with him. His patience with my stubbornness, his sincere willingness to push my research to be among the best of the best, have motivated me all these years to keep on persisting and not giving up on countless trials and errors to develop new algorithms for my research, to make the best of out of any possibilities given. Without his invaluable comments, suggestions, and continual supports, I would never have figured out and finished up my research, and would have had perhaps gone off track and run into a brick wall of hopelessness of unsolvable problems. I am also very grateful to all of my committees, which include Dr. Richard Thompson, Dr. Prashant Krishnamurthy, Dr. Marwan Simanna, and Dr. Michael McClound for their encouragement and numerous of helpful comments to better my research.

I am also greatly indebted to TOT Public Company Limited for supporting me financially during my graduate study at the University of Pittsburgh, giving me this wonderful opportunity to pursue my Ph.D. in telecommunications. I would like to further acknowledge and thank the TeleNet at Pitt program for financing me all the stipends and traveling expenses for every conference I have participated. Also, many thanks to Debdhanit, Maria, and Yuttasart of the SiNE group for always giving wonderful critiques, comments, and lot of thoughts.

Lastly and most importantly, I forever owe love to my lovely parents and all my family. With their unconditional loves and supports throughout my life — always empowering me to move on to the next step — I am now reaching my lifelong dream.

1.0 INTRODUCTION

Growth in wireless demand is occurring globally at steady pace. Increase in wireless usage has propelled researchers and standardized bodies to collaboratively develop new technologies to maximize network capacity with higher and higher data rates. Wireless technologies have evolved from analog to digital and from voice circuit-switched to data packet-switched. As a consequence, there exists in parallel a variety of emerging services such as HTTP web-browsing, file transfer with FTP, and emailing. However, actual network deployments have not kept pace with the improving technologies. Network designers and researchers still concentrate on either providing the coverage or maximizing the total network capacity, but none considers the data rate requirements or users. In this paper, wireless network designs refer to the Base Station (BS) placement that can provide sufficient signal coverage and satisfy user (i.e., with various data rate requirements). The BS placement problem is formulated as a combinatorial, mixed-integer problem, and categorized as NP-Hard. Convergence time to the optimal solution increases exponentially with the size of search space (section 5.2). The search space is typically a set of Test Points (TPs) at which the BSs are placed. The number of TPs can significantly increase when designing large networks, and therefore it may take hours if not days or weeks to find the optimal solution. For most network designs, TPs are either assigned randomly [1], or are assigned as grid points [2], [3]. Too few TPs may result in sub-optimal solutions, while too many may result in extra computations (chapter 7). This paper presents a sweep and merge algorithm to rapidly reduce a large number of TPs and determine a smaller set of TPs which always yields the optimal BS placement solution. The demand nodes (section 5.1) are also incorporated to represent user traffic in a quantum sum of voice calls or data bit rates to further minimize the computation.

1.1 WIRELESS EVOLUTION

Wireless networks have evolved rapidly to support new services and increasing demands for mobile wireless communications. Figure 1.1 shows the time line for evolving U.S. cellular technologies and the increasing data rate supported.

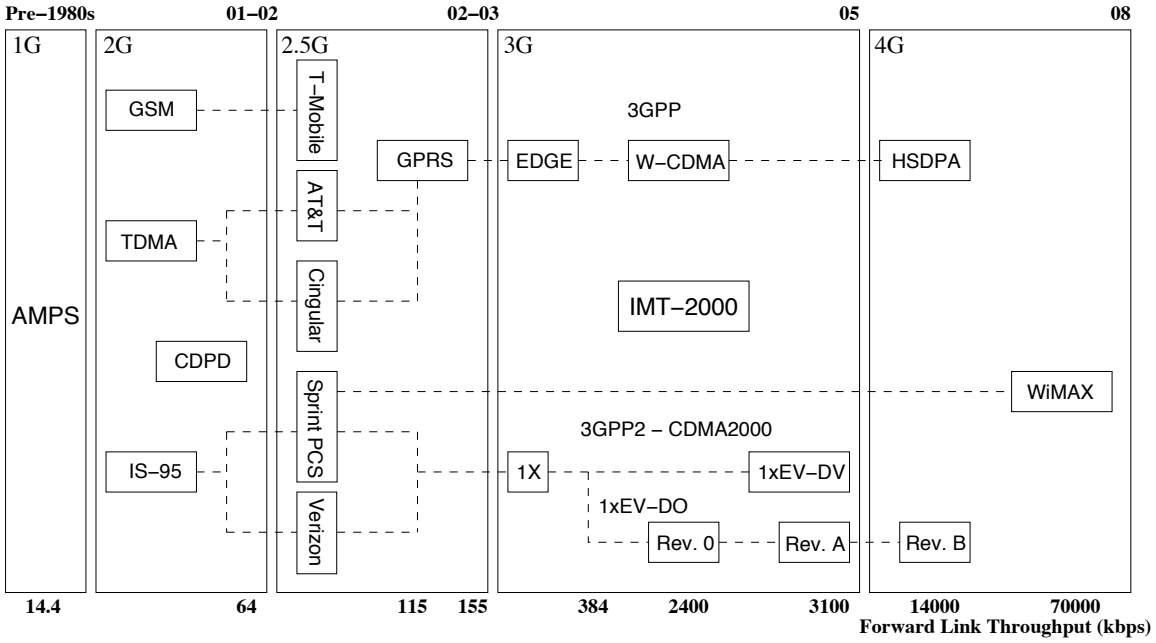

Figure 1.1: U.S. Cellular Technology Evolution Path

In early 1980s, the first generation (1G) mobile system was deployed using an analog transmission. 1G systems employed a circuit-switched technology to carry voice calls via Frequency Division Multiple Access (FDMA). Frequency reuse allowed 1G systems to achieve higher capacity given a limited frequency spectrum. Notable 1G systems are such as Advanced Mobile Phone Services (AMPS) system used in the United States, Narrowband AMPS (NAMPS), Total Access Cellular System (TACS), and Nordic Mobile Telephone System (NMT-900) [4].

Second-generation (2G) cellular systems employed digital technology to improve transmission quality, system capacity, and coverage in addition to 1G systems. Wireless access links in 2G systems were based on either Time Division Multiple Access (TDMA) or Code Division Multiple Access (CDMA) to achieve more efficient spectrum utilization. Several commercialized 2G digital systems includes the Global System for Mobile Communication (GSM), IS-95 (CDMAone), Pacific Digital Cellular (PDC) as well as United States Digital Cellular (USDC) standards IS-54 and IS-136 (D-AMPS) [4]. Transitions from 2G to 3G (2.5G) created new services such as Cellular Digital Packet Data (CDPD), General Packet Radio Service (GPRS), and Enhanced Data rates for Global Evolution (EDGE) to accommodate the increasing demand in data services [5]. CDPD was the AMPS successor, while GPRS and EDGE were adapted from the GSM.

1.2 THIRD GENERATION (3G) AND BEYOND

1.2.1 W-CDMA/HSDPA

Radio Access (UTRA), is a promising 3G technology that provides higher capacity to support higher data rates for voice, video, data, and image transmission. W-CDMA supports data rates up to 2 Mbps for local area access and 384 kbps for wide area access [6], [7]. W-CDMA is a spread spectrum technology, adopted as a standard by ITU under the name IMT-2000 spread spectrum. W-CDMA systems employ a Direct Sequence Code Division Multiple Access (DS-CDMA) as the main access scheme, spreading data signals over a wide 5 MHz bandwidth (projected for 10 or 20 MHz in the future). There are two different modes namely

- Frequency Division Duplex (FDD): two separated frequency bands are assigned to the uplink and downlink, and
- Time Division Duplex (TDD): two different time slots in the same synchronized frequency band are assigned for the uplink and downlink

To compete with CDMA2000 EV-DO, W-CDMA evolves to HSPDA (High-Speed Downlink Packet Access), aiming to support data rate up to 14.4 Mbps downlink. The improvement in

data rate derives from fast packet scheduling and adaptive modulation [8]. The future road map of W-CDMA track is to include Multiple-Input Multiple-Output (MIMO) antenna system employing OFDMA (Orthogonal Frequency Division Multiple Access) with the expected data rate up to 200 Mbps [9].

1.2.2 CDMA2000

CDMA2000 is the improvement version of IS-95. The CDMA2000 specification was developed by the Third Generation Partnership Project 2 (3GPP2), a partnership among five telecommunications standards: ARIB and TTC in Japan, CWTS in China, TTA in Korea and TIA in North America. CDMA2000 1X is the first phase of CDMA2000, offering a full backward compatibility with the existing IS-95 technology, but providing twice the voice capacity with an always on feature and supports higher data rates upto 144 kbps [10]. CDMA2000 1xEV is the enhancement to 1X systems with two competing technologies: 1xEV-DO (Qualcomm Inc.) and 1xEV-DV (Motorola and Ericsson). 1xEV-DO stands for Evolution Data Optimized, using a separate carrier to transmit data, while 1xEV-DV stands for Evolution Data and Voice which integrates voice and data on the same carrier. Adaptive Modulation and Coding techniques together with the TDM/CDM multiplexing enables both the 1xEV-DO and 1xEV-DV to achieve higher data rates than the predecessor CDMA2000 1X systems. 1xEV-DO revision 0 supports upto 153.6 kbps on the reverse link and the maximum of 2.4 Mbps on the forward link [11]. Last year, 3GPP2 released 1xEV-DO revision A to improve the maximum reverse and forward data rates to 1.8 and 3.1 Mbps respectively. 1xEV-DV also supports a peak data rate upto 3.1 Mbps in the forward link and allows multiple users to share spectrum and codes simultaneously [12], while 1xEV-DO dedicates the entire code space and power to one user at a time to optimize the average throughput on the forward link [13]. However, 1xEV-DV requires more complex resource allocation due to voice and data multiplexing with adaptive bandwidth and power management [14]. As reported by CDG, 147 out of 240 millions of CDMA subscribers worldwide are using CDMA2000 1X or 1xEV-DO networks [15], [16]. Currently, 112 operators from 6 continents are already servicing CDMA2000 1X and are upgrading to 1xEV-DO. Qualcomm claimed that CDMA2000

1xEV offers the lowest cost to deliver data traffic ($0.022 per MBytes) compared to other technologies such as GPRS ($0.415/MB) or even the W-CDMA ($0.069/MB) [17]. An upcoming 1xEV-DO line is the revision B standard, utilizing 5 MHz of bandwidth, and the data rate is expected to go up to 14.7 Mbps [18].

1.2.3 WiMAX

WiMAX stands for Worldwide Interoperability for Microwave Access, which was developed by WiMAX Forum as part of the IEEE 802.16 standard. WiMAX offers data rate up to 70 Mbps downlink through Scalable OFDMA with MIMO technology [9], [19]. Multiple frequency bands are available for WiMAX: 2.3 GHz, 2.5 GHz, 3.3 GHz and 3.5 GHz with wide range of spectrum bandwidth from 5 to 10 MHz [19]. WiMAX can be implemented in various ways as shown in figure 1.2 [20], [21].

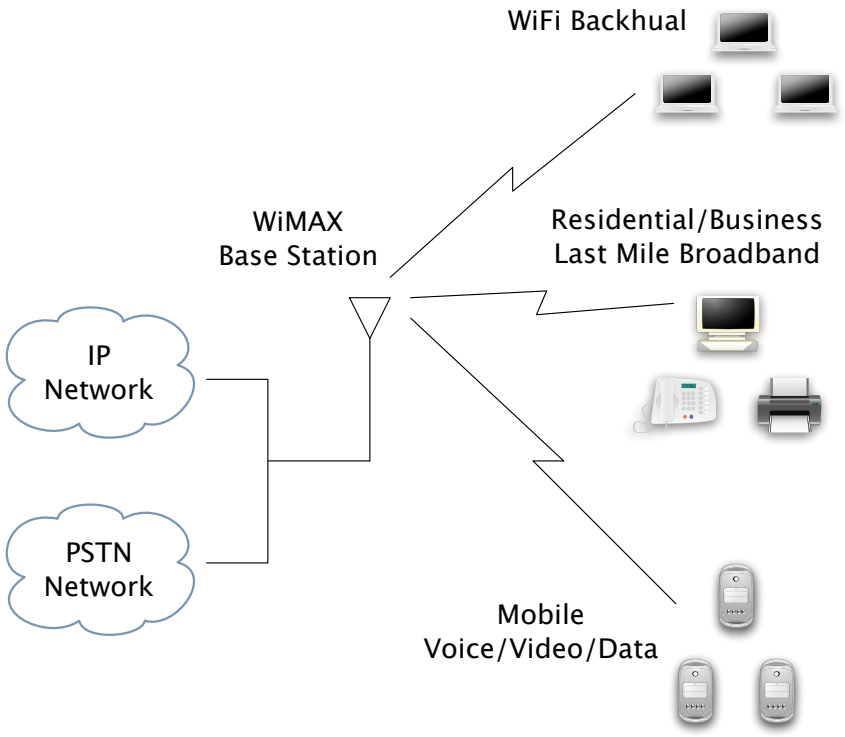

Figure 1.2: Possible WiMAX Implementations

Companies like Sprint Nextel and KDDI are considering WiMAX to enhance speed and to improve QoS for data-intensive mobile applications [22], [23]. WiMAX is also envisioned to be the "last mile" broadband access alternative to cable or DSL (Digital Subscriber Line) for remote areas where cost of wiring is prohibitive and not worth the investment [21].

1.3 NETWORK DESIGN TERMINOLOGY

The following terminologies are used throughout the paper. Table 1.1 explains the meaning of each terminology.

Table 1.1: Network DesignTerminology

Terminology	Definition
Base Station (BS)	A set of fixed circuitry equipments including antennas used to transmit and receive both voice-modulated and data-modulated signals over the air.
Demand Node	A virtual point representing the center of an area containing a quantum of user traffic.
Test Point (TP)	A set of candidate locations for placing BSs.
Reverse Link	A communication path or a wireless channel in the direction from a user to the BS.
Forward Link	A communication path or a wireless channel in the direction from the BS to a user.
Throughput	Raw bit rates successfully demodulated at the receiver terminal.

1.4 RESEARCH STATEMENT

The goal of this dissertation is to develop a method to reduce the computation requirement for computing the optimal BS placement that can provide sufficient signal coverage and satisfy user traffic — the optimal solution. To reduce the computation requirement for solving the optimal solution, this paper presents:

- A TP reduction algorithm to minimize the number of TPs for placing BSs, while the optimal solution set is still maintained.
- A minimum branching algorithm, which is an extension of the Depth First Search (DFS) algorithm, to compute the optimal coverage solution (replacing the exhaustive search).
- An algorithm based on Tabu Search (TS) to locate additional BSs to support excess capacity requirements (optimization of BS assignment).
- Demand node representation of user traffic (limitations are discussed in chapter 8)

The computation requirement is a function of the search space size, which increases exponentially as the number of TPs increases (section 5.2). The minimum number of TPs that always guarantee the optimal solution set are computed efficiently by the sweep and merge algorithm. Then, the coverage solution is first computed by applying the minimum branching algorithm. Similar to the DFS algorithm, the minimum branching algorithm always guarantees the optimal coverage solution but requires substantially few number of computations than the exhaustive search (section 6.1.2). In cases when the coverage solutions can not support the total demand (i.e., fail to meet the capacity requirement), the design incorporated the TS-based algorithm to solve for the new optimal assignment of BSs that can now satisfy both the coverage and the excess capacity requirements. Also, distributed user traffic is discretized as demand nodes, representing a quantum of either voice calls or data rate bit rates, to further reduce the number of coverage requirement tests. Using the TP reduction algorithm to reduce TPs, the minimum branching algorithm to compute for the coverage requirement, the TS-based algorithm to solve for the new optimal BS assignment, and the demand node representation of traffic, the design computation is minimized.

1.5 OUTLINE

Chapter 2 and 3 reviews past and present network design methods and path loss models for the BS placement problem. Chapter 4 provides a brief introduction to several well-known heuristics which are incorporated in most network design methods to solve the BS placement problem. Chapter 5 describes the sweep and merge algorithm to reduce TPs (the TP reduction algorithm), while chapter 6 explains the minimum branching algorithm used in computing the coverage solution. Chapter 6 also presents the TS-based algorithm developed specifically to optimize BS assignment when the coverage solution can not meet the capacity requirement. Chapter 7 provides results from various design examples ranging from small to large networks, including several design examples for CDMA2000 networks in Graz, Austria. The final chapter 8 concludes the dissertation, discusses its limitations, and suggests possible future research work.

2.0 WIRELESS NETWORK DESIGN REVIEW

Wireless network design is a process to determine the number of BSs and their placement to provide sufficient signal coverage and capacity to the designated area. Base station placement involves many constraints and restrictions. Traditional design approaches aim to create the coverage-based design; that is to ensure that an adequate signal strength is maintained in the intended Service Area (SA) as shown in figure 2.1.

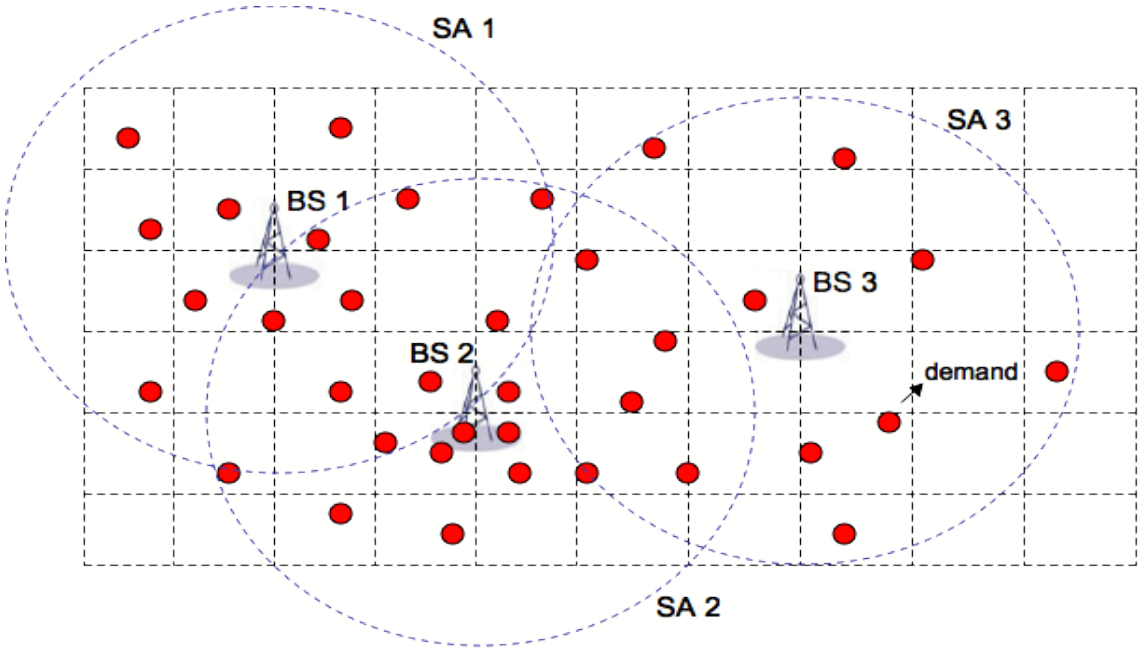

Figure 2.1: An example of BS placement showing grids, user demand, and SAs.

One simple approach to meet the coverage requirement is to employ a trial and error method, in which a set of candidate sites are randomly chosen, and the coverage is optimized by tuning antenna parameters such as azimuth, tilt, and power. However, the trial and error approach

consumes an enormous time [24] and usually ineffective since the process may overestimate the number of BSs or may result in a suboptimal signal strength pattern.

There are several missing components in the coverage-based design. First of all, traffic demand and user density are not considered. The coverage-based approaches may appear sufficient for networks where user density is low and traffic load is light. However, the problem may arise if the network has higher user concentration and applications demanding larger bandwidth. The coverage-based network may not have sufficient capacity, and as a result causing delays and interruptions. Capacity requirements and optimization are thus a primary concern for network designs such as the Tutschku method for TDMA networks and the Akl method for CDMA networks [25], [26].

Alternatively, one may concern minimizing the cost of deploying BSs. The total cost of one BS, including equipments, installation, and site acquisition may add up to about five hundred thousand to one million dollars per site [24]. Consequently, the main objective is to minimize the total number of BSs instead of either maximizing either the coverage or capacity. Example designs are such as the Amaldi method for power-controlled CDMA systems and the Hao method for TDMA systems [27], [1].

In some scenarios, excessive interference may occur due to poor BS layouts and frequency assigments. High interference either from frequency overlapping or frequency reuse usually results in substantially low data rate and low utilization of the network bandwidth. Minimizing interference is therefore an alternative approach to increase network utilization and to maximize the capacity. The Weicker method uses Genetic Algorithm to minimize both the interference and the deployment cost for TDMA systems [28], while the Hills design method minimizes interference in order to enhance the coverage in WLAN environments [29]. The Prommak method for WLAN designs improves on the Hills method by also minimizing the interference but at the same time satisfying both the coverage and capacity requirements by employing only a sufficient number of Access Points (APs) [3]. In summary, the network design or the BS placement problem may be classified into different categories according to

the problem objective: Coverage Objective, Capacity Objective, Cost Objective, and Interference Objective. The following literature review discusses in detail the advantages and disadvantages of each approach.

2.1 COVERAGE OBJECTIVE

The coverage-based designs are suitable for both indoor and outdoor environments requiring optimal signal reception. For indoor environments, Sherali and Unbehuan introduce the concept of Minisum and Minimax to minimize path loss to all receivers in the network [30], [31]. The Minisum function (f_1) minimizes the weighted sum of all the predicted path loss, while the Minimax function (f_2) minimizes the worst path loss receiver. The Sherali method compromises between the overall coverage and the worst-case coverage by weighting the Minisum and Minimax using the combined utility function:

$$f = \psi f_1 + (1 - \psi) f_2. \tag{2.1}$$

Sherali showed that $\psi = 0.5$ balances equally the average and the worst case coverage [30]. The Sherali method derives an initial solution for the multiple transmitter problem by partitioning the design space at the center of gravity point of its longest dimension. The process continues iteratively by dividing the resulting partition with the highest cumulative weight until there are n partitions (number of transmitters) [30]. The Unbehaun method obtains an initial solution by pruning out the transmitter at each grid point yielding the lowest f one at a time until there are only n transmitters left [31]. Given the initial solution with n transmitters, the neighborhood search in the Unbehaun method evaluates every adjacent grid point to the previous transmitter location and promotes the point producing the lower f as a new solution. The algorithm finishes when f stops improving. The Sherali method examines both the gradient-based and line-search techniques to locate the optimal transmitter placement [30]. The gradient-based approach moves the transmitter in the direction to lower the gradient of the objective function f. The line-search strategy is similar to the neighborhood search, however the search step is not limited to only the adjacent grid but varies at

each iteration. The resulting network design by the Sherali method requires approximately half the number of transmitters for achieving the same cumulative SIR compared to the Unbehaun method. The Minisum and Minimax approach can ensure ubiquitous coverage to all receivers, however since the capacity requirement is ignored, the approach is not suitable for networks with high traffic concentration — since some BSs may cover traffic higher than their maximum capacity.

For outdoor environments, the network characteristics are more diverse and the service areas are typically much larger than those of the indoor environments. Searching where to place BSs that maximizes the signal coverage may require indefinite amount of time. To expedite the design process, optimization heuristics for mixed-integer problems such Simulated Annealing (SA), Genetic Algorithm (GA), and Tabu Search (TS) are applied. In the Anderson and McGeehan SA implementation [2], the cost function C represents the coverage quality, which is measured as the square difference between the received signal level and the desired value. The SA process iteratively selects a new BS configuration at each iteration k; the updated BS configuration is accepted only if the new cost function C' is lower, however if C' is higher, it may still be accepted with probability.

$$P = \min\left\{1, e^{-A/B}\right\} \tag{2.2}$$

where $A = (C' - C)/C$, $B = (T_k/T_0)^2$, and T_0 is the starting temperature. The SA algorithm gradually improves the solution by reducing the magnitude of BS movement (probability of accepting inferior BS configurations) as the temperature T_k decreases, and the final temperature is reached when the cost reduces to near zero. Although the resulting solution from SA is comparable to the optimal solution from the exhaustive search of all grids (i.e., within a few meters [2]), solution convergence is not guaranteed since the algorithm is highly sensitive to parameters such as the initial temperature, the cooling schedule, etc. Another method employs GA to maximize the coverage [32]. The Leiska GA method partitions the service area into a small blocks of subareas instead of grid points, and the number of covered subareas are maximized accordingly [32]. For each solution (individual), the number of BSs can vary from 1 to n, where n is the number of candidate sites. The fitness value $f(I)$ is

used to indicate the solution quality of each individual such that, for an individual with k BSs

$$f(I) = \frac{\#\text{ of covered subareas}}{|k-n|+1} \tag{2.3}$$

At each iteration, each individual in the population (a pool of solutions) is mutated by randomly activating or deactivating BSs. The individual I with the best fitness value is then chosen to produce a new population for the next generation. The convergence property of GA depends on rules defining how mutation is conducted, and is analogous to the selection of SA parameters. Both the Anderson and Leiska design methods still do not consider the capacity requirement.

2.2 CAPACITY OBJECTIVE

Providing ubiquitous and sufficient signal coverage is the minimum requirement for all network designs. The coverage-based designs such the Sherali method for indoor environments [30] or the Anderson method for outdoor environments [2] work effectively on the condition that, traffic load is light and uniformly distributed. However, when traffic concentration is not uniform — i.e., clustered traffic in some spots especially in urban areas and sparse in suburban areas [33] — the coverage-based network may not be sufficient since high traffic in certain locations may exceed the maximum capacity of some BSs. Thus, some BSs may carry excess traffic while some may be responsible for traffic far below their threshold capacity, which results in suboptimal utilization of the network capacity. To account for the capacity requirement and to optimize the network capacity, a variety of different design approaches have been proposed. Alejandro and Hanly develop the analytical method to maximize the total network capacity for CDMA systems [34], [35]. Their methods are applied to different environment settings. The Alejandro design method focuses on the effect of irregular propagation in urban environments to the cell capacity. The Alejandro method aims to establish the optimal relationship between the break point distance R_b and the BS radius R_c for the full- and half-square cell configurations [34]. As pointed out by [36], radio propagation in urban environments usually exhibits two slope (n_1, n_2) path loss separated by the break point

distance R_b, which is a function of the BS antenna height h_b, the receiver antenna height h_m, and the signal wavelength λ

$$R_b = 8.41 h_b h_m / \lambda \tag{2.4}$$

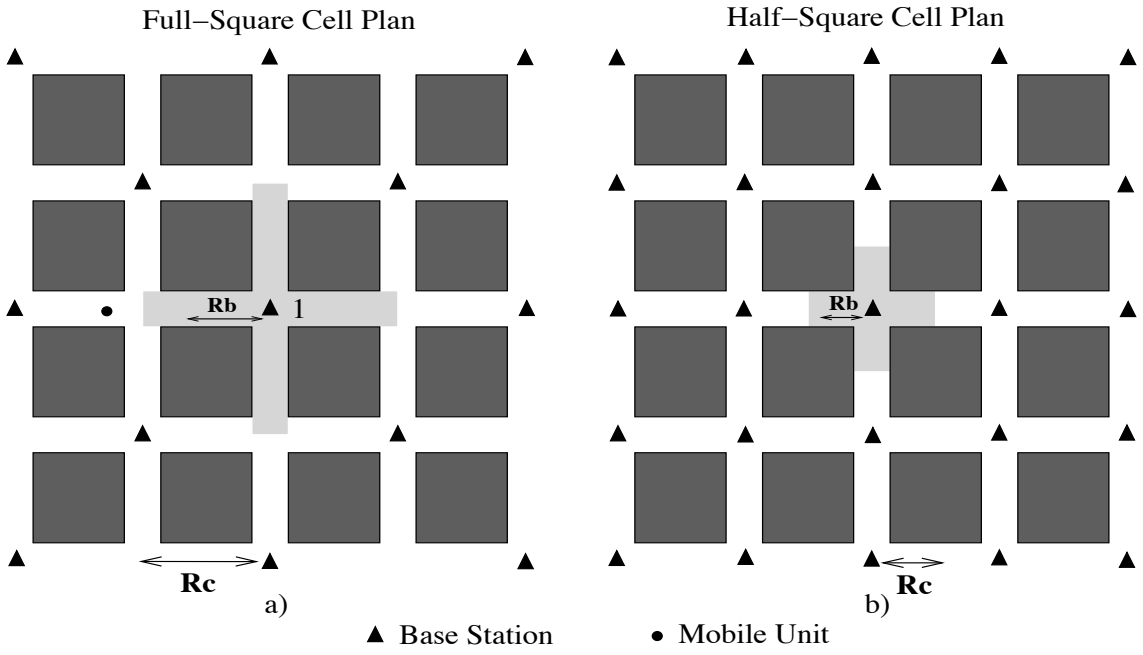

Figure 2.2: a) Full-square cell configuration, a BS at point 1 employs an omnidirectional antenna and the break point distance R_b is shown. The light gray shading is the street area covered by BS_1. The dark areas are buildings; the white areas are roadways. b) shows the half-square cell configuration. BS power is reduced to half of the full-square cell configuration BS.

If s is the distance between the transmitter and mobile, L_b is the path loss at the break point in dB, and ζ represents a log-normal shadowing loss component — Guassian random variable with zero mean and variance = σ^2 (dB). The path loss $L(s)$ is expressed as

$$L(s) = \zeta + \begin{cases} L_b + 10 n_1 \log(s/R_b), & s \leq R_b \\ L_b + 10 n_2 \log(s/R_b), & s > R_b \end{cases} \tag{2.5}$$

The full-square cell configuration allocates a BS with an omnidirectional antenna at every other intersection covering a block in all four directions. The half-square cell configuration increases network capacity by doubling the number of BSs where each BS covers half a block

in all four directions. Alejandro shows that for both full- and half-square cell topologies, the optimal ratio of the break point distance R_b to the BS radius R_c must be less than 0.7 in order to maximize the total network capacity, irrespective of traffic density [34].

Hanly alternatively suggests that, in less built-up environments the traffic density ρ (Poisson) and cross-correlation of the shadow fading ζ (jointly Guassian distributed) are now the factors that determine the optimal BS density [35]. In the Hanly design approach, the triangular grid of BSs configuration as shown in figure 2.3 is used.

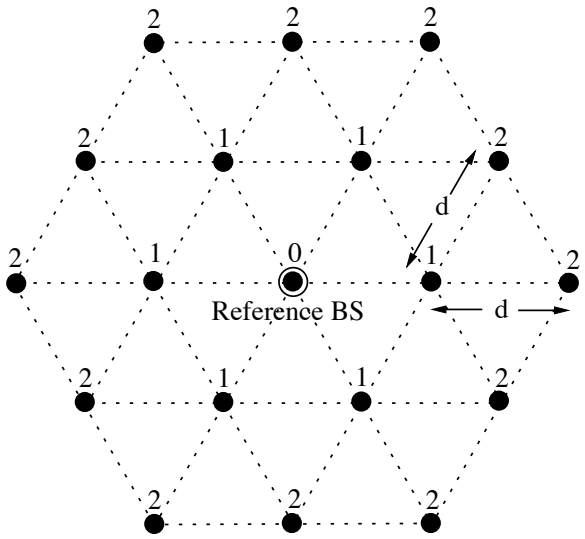

Figure 2.3: Triangular grid of BSs — with the reference BS_0 at the middle, the number above the BS represents a tier; the distance between BSs is d. Only the 1^{st} and 2^{nd} BSs are accounted for interference.

Hanly shows that when the distance between BSs is approximated $\approx 1/2/\sqrt{\rho}$, the network capacity is maximized. The Alejandro and Hanly design methods are derived from uniform traffic distribution, but as demonstrated in [33], the assumption may be too ideal for real world network traffic. To accommodate traffic non-uniformity and to aid the design procedure, a demand node concept is introduced [37]. The demand node is the center of an area containing a quantum of demand from a teletraffic viewpoint [25]. The demand node is generated by recursively bicsectioning the service area into two rectangles with the same amount of traffic until the traffic of every tessellation piece falls below a threshold. The

demand node location is the center of gravity of the traffic within the tessellation. Figure 2.4 illustrates the tesselation procedure.

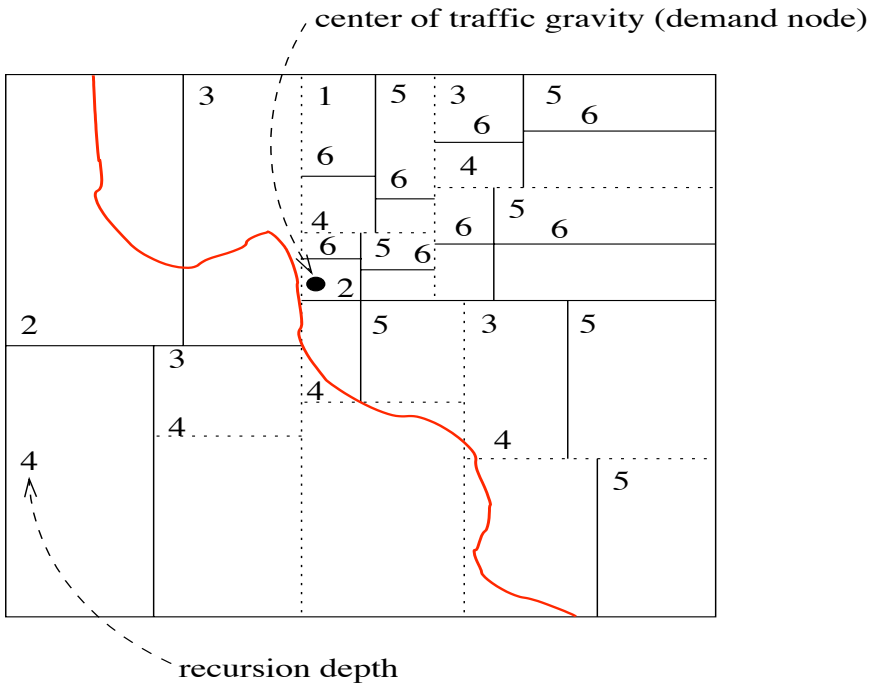

Figure 2.4: Service Area Tessellation and Demand Node (a curvy red-line represents a road)

The Tutschku design method starts by creating demand nodes, then seeks for the BS covering the maximum number of demand nodes. The same process is repeatedly applied to the remaining demand nodes until there is no BS left or all demand nodes are covered. The Tutschku method is fast compared to Simulated Annealing or Genetic Algorithm, however the total network capacity is not maximized because the BSs found in earlier iterations may cause network partitioning, and more BSs are needed to cover the remaining demand nodes. The Tutschku method is intended for TDMA systems where each BS has a fixed capacity and a portion of bandwidth is dedicated to each user separately. However in CDMA systems, the BS capacity is dynamic because users share the entire spectrum bandwidth — each contributes to the total amount of interference, and the higher the interference, the lower the BS capacity, and as a result the Tutschku method may fail. The Akl design instead maximizes the capacity by adjusting both the BS pilot power and the user transmit power

[26]. Users in smaller cells are assigned higher transmit power to compensate for excessive intercell interference from bigger cells. The Akl power compensation strategy helps balancing each cell capacity, and the net result is the improving in the total network capacity. Unfortunately, the strategy is based on the assumption that the required SIR is constant, therefore if the required SIR varies according to the transmit data rate (i.e., CDMA2000 EV-DO), the strategy may not yield the optimal result as expected.

All of the design methods discussed above and in the previous section succeed in either providing the maximum coverage or the maximum capacity for both indoor and outdoor environments. These approaches are suitable if the number of BSs are already determined and financial expense or cost is not a factor. However if the requirement is to provide sufficient coverage and capacity, the network may need additional number of BSs to satisfy, which may not be feasible due to cost restriction.

2.3 COST OBJECTIVE

The total cost of deploying BSs, including equipment, installation, and site acquisition is a key factor determining the feasibility of network designs. Minimizing cost is thus the primary objective of several network design methods discussed in this section. The Lee and Gang method minimizes the total cost to support network expansion in CDMA systems [38], while other design methods such as the Amaldi design and the Hao design not only consider the total BS cost but aims to maximize the amount of traffic coverage as well. The Amaldi design method incorporates the composite cost function to minimize the number of BSs while maximizing the traffic coverage in CDMA systems [1]. The Hao design method further breaks down the cost into: hardware and installation, antenna, transmitter and receiver, and the penalty for coverage holes [27]. Each of these methods is presented in more detail next.

The Lee and Gang design method considers only the BS cost, and the design objective is simply to minimize the total number of BSs needed. The design has two requirements: α

portion of traffic must be covered, and each BS must carry traffic less than its maximum capacity [38]. The method employs Tabu Search (TS) to solve the problem and assumes that all candidate locations of the new BSs are already determined and are assigned as the initial solution. Normalized Residual Capacity (NRC) is used as a benchmark to determine when to drop a BS. If C is the BS cost, M is the maximum BS capacity, then

$$NRC = \frac{\text{residual capacity}}{M} \times C \tag{2.6}$$

The TS process randomly adds and drops BSs until the cost is minimized and all traffic is covered. Table 2.1 compares the resulting network design for 400, 900, 2,500 demand points using TS vs. the linear optimization solver (CPLEX).

Table 2.1: CDMA Network Designs, showing the required number of BSs, and the computation time in (second). Total traffic demand is in Erlang.

# of Points	Total Traffic Demand	$\alpha = 0.90$		$\alpha = 0.95$		$\alpha = 0.99$	
		Tabu Search	CPLEX	Tabu Search	CPLEX	Tabu Search	CPLEX
400	1988.6	14.6 (0.48)	14.6 (709.97)	17.0 (0.90)	16.8 (>10000)	20.4 (1.40)	18.6 (>10000)
900	4470.3	29.2 (5.76)	28.6 (>10000)	35.6 (9.09)	35.6 (>10000)	46.0 (16.44)	40.2 (>10000)
2500	12490.0	81.0 (116.99)	80.0 (>10000)	101.4 (190.50)	98.8 (>10000)	129.2 (304.24)	112.8 (>10000)

As shown above, for $\alpha = 0.9$ and $\alpha = 0.95$, the Lee and Gang method requires approximately the same number of BSs as the CPLEX solver does, but requires about three orders of magnitude fewer computations. The method, however, must evaluate all the requirements at each iteration every time a BS is either added or dropped. The evaluation process consumes additional computational time and increases the algorithm complexity. To further reduce the complexity, the Amaldi design method bypass the evaluation by converting the requirement into one of the cost components in the objective function C_f [1]. If u_i represents the traffic demand at point i and c_i is the BS cost, then

$$C_f = \sum_i u_i - \frac{c_t}{m} \times \{\#\ \text{of activated BSs}\} \tag{2.7}$$

where m is the total number of random candidate sites. By maximizing C_f, it is equivalent to maximizing the coverage and at the same time minimizing the BS cost. Amaldi also focuses on ensuring that the network must meet either the target power level or the SIR level (at BSs). Solving for the solutions starts by forming an empty set and iteratively adding a BS producing the largest C_f until the addition of a new BS no longer increases C_f. Any demand points causing excessive interference are removed from the objective function C_f. Similar to the Lee and Gang design, the Amaldi design also employs TS to further maximize C_f, but in addition to the drop and add moves, a swap move is also defined (a combination of both the drop and add moves) [1]. The method deliberately assigns higher priority to sites relatively close to each other in order to reduce the possible number of swap moves. For small problem instances (number of demand points $n = 95$, number of candidate sites $m = 22$ in 400 x 400 m^2), the resulting solutions are comparable to the optimal solution from CPLEX in number of activated BSs. However, for large problem instances ($n = 750, m = 200$, in 1.5 x 1.5 km^2), the solution from the Amaldi method may not always satisfy the traffic demand since total BS cost may outweigh the coverage cost (the algorithm is prone to sacrifice the traffic portion in order to maximize the overall objective function C_f).

The Amaldi method assumes that each BS is equipped with antennas transmitting at constant gain and power [1]. However, allowing BSs to have flexible coverage may be advantageous when both geographical areas and land usages are diverse (e.g., rural, suburban, urban). In the Hao design method, the total cost of BS is assumed to be a linear function of the antenna gain, the transmit power, and the hardware and installation cost. The entire design area is divided into equal grids, and the seven-cell frequency-reuse pattern is used [27]. The first phase of the Hao design estimates the upper-bound number of BSs and their radiuses to provide full traffic coverage. The second phase employs SA to minimize the composite cost function C_f, which is now defined as

$$C_f = \#\ \text{of activated BS}_k \times C_k + \text{traffic violation} \tag{2.8}$$

where C_k is the total cost of activating BS_k. The last phase involves frequency assignment to reduce the interference. The SA process randomly reassigns traffic in each grid to different

BSs to minimize C_f at each temperature t_k. The new configuration is accepted if the updated cost C'_f is better than the current cost C_f or alternatively if

$$e^{(C'_f - C_f)/t_k} > \mathsf{random}[0, 1] \tag{2.9}$$

In the example network with a total number of 100 traffic grids; traffic density in each point ranges from 150 - 200 users/hour, the Hao design reduces the number of BSs in the first phase from 20 to 14. However, similar to the Amaldi design, the total BS cost may dominate the objective function, and only a small portion of traffic can be covered. Furthermore, the advantage of flexible BS coverage is offset by fixing the cell pattern, and more than that the method must loop back to the first phase if excessive interference still persists.

2.4 INTERFERENCE OBJECTIVE

Signal quality is a measure of the receiver's likelihood to extract the desired signal from both interfering signals and noise. Excessive interference by sharing either the same frequency channel or the entire spectrum may prevent the receiver to demodulate and recover the intended signals, thereby reducing the throughput and capacity [39]. Interference level is critical in a system where extensive frequency reuse is inevitable such as WLANs or picocell TDMAs. In such systems, the design feasibility is tied to the interference level it produces. Neither the coverage- nor capacity-based design considers interference, and therefore is not suitable. Minimizing cost is still mandatory for outdoor environments, but minimizing interference is also necessary because poor frequency assignment may still cause unacceptable interference level. Several design methods discussed earlier such as Tutschku's and Hao's delay the frequency assignment procedure until the main objective is reached [25], [27]. As a result, interference is not fully minimized, and feasibility is uncertain.

On the contrary, the Weicker design method weights equally both minimizing the BS cost and interference for TDMA systems [28]. The BS cost is assumed to be a linear function of the transmit power and capacity. The Weicker method treats interference as the number of

BSs encountering either the adjacent- or co-channel interference. Adjacent-channel interference is a result from partial bandwidth overlap among users in a cell. Whereas, co-channel interference occurs when users in the coverage overlap between two neighboring cells are assigned the same channel. Similar to the Lee design [38], a feasible solution (individual I) is a set of BSs located among grid points satisfying the traffic requirement. However, in the Weicker method, one feasible solution may require fewer BSs but produce higher interference than the others and vice versa. The solutions that dominate the others (dominating individuals) must have both the BS cost and interference less than the dominated individuals.

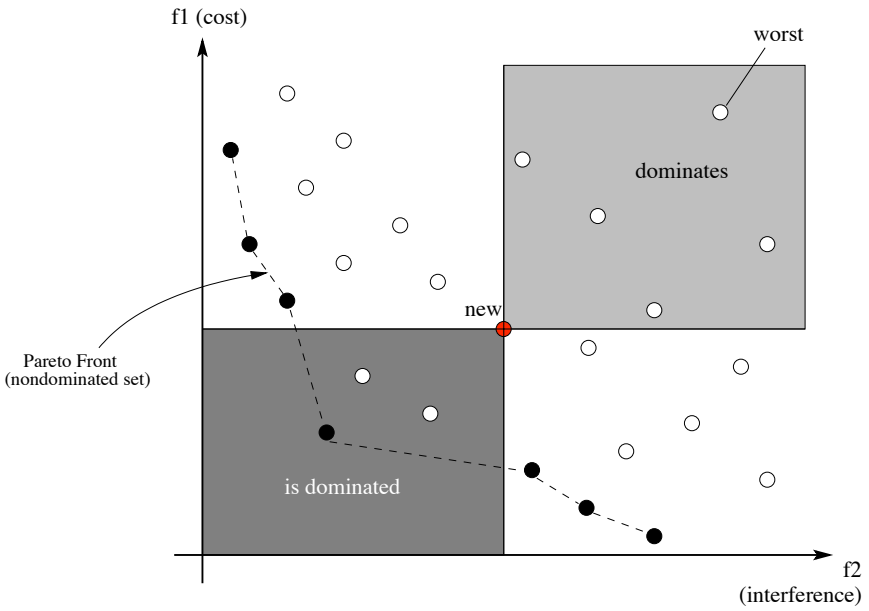

Figure 2.5: Cost and Interference of each individual, the darker shade illustrates a set of individuals that dominate the new individual. The lighter shade indicates a set of individuals dominated by the new individual. Dotted-line connects the best individuals known as — the Pareto front

As shown in figure 2.5, the Weicker design objective is to prune out a set of non-domimated individuals — the Pareto front. GA is used to improve the Pareto front at each iteration by creating new individuals from a current pool of existing individuals (population). Each individual is evaluated by the fitness value, which is defined as

$$f(I) = |\text{individuals dominating } I| \times Popsize + |\text{individuals dominated by } I| \qquad (2.10)$$

At each iteration, the population is randomly divided into groups of individuals. Each group selects the best individual as its parent to create new individuals by applying either the permutation or the recombination process [28]. Here, the permutation process adjusts cell capacity, transmit power, also adds and removes BSs. The recombination process decomposes the service area of one individual into two halves and recombines them to produce a new individual. The worst individual (highest fitness value) of a group is eliminated (as shown in figure 2.5). Weicker demonstrates that the best Pareto front occurs when the probability of permutation and recombination are 0.6 and 0.4 respectively [28].

The Weicker method emphasizes on reducing interference and minimizing the BS cost for outdoor environments (i.e., cellular TDMA systems). However, for indoor environments, the circumstances are different. Since the unit cost of Access Points (APs) can be neglected, minimizing the cost or equivalently the number of required APs may no longer be necessary. Unfortunately, the 802.11 WLAN has only three non-overlapping frequencies, and if the design approach is still based on the cost-objective, considerable interference may reduce the total network throughput, and the resulting design may not be feasible. Minimizing interference is therefore the first priority for several WLAN designs. As examples, the Hills design method focuses on minimizing interference while maximizing the coverage on a campus scale network (i.e., entire Carnegie Mellon University campus covering 65 buildings with 3 million ft^2 [29]). Hills proposes the linear array and rectangular array layouts for initial AP placement [29]. The linear array is employed when a floor span is smaller than the AP coverage radius, or otherwise the rectangular array is employed instead.

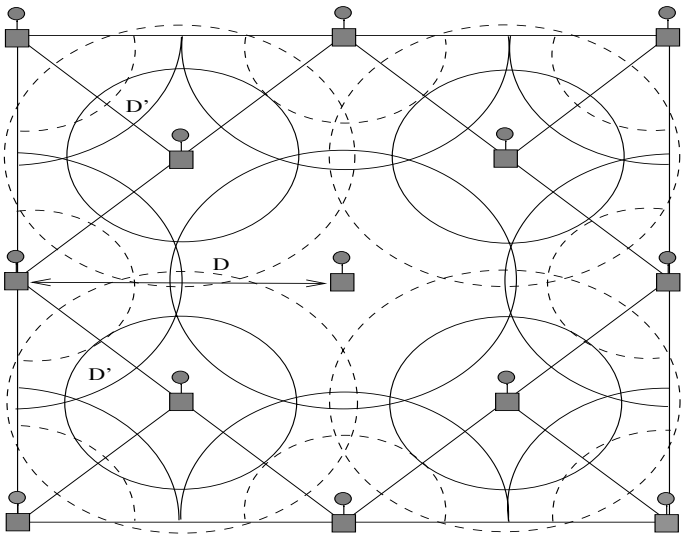

Figure 2.6: Rectangular array of APs in a multi-floor building, a solid circle represents the coverage area within the same floor, and a dotted circle is for the coverage area of the adjacent floor. The coverage diameter D and D' are shown respectively

After intialization, AP locations are further adjusted using feedback from the signal strength measurement to maximize the coverage. To minimize interference, higher density areas are given higher priority for frequency assignment than lower density areas, and the coverage map is used to avoid frequency overlapping. The Hills design is effective and fast for creating a start-up, coverage-based WLAN network, however the design does not consider the data rate requirement, and therefore may not have sufficient capacity to support a wide range data rate demand from various users. As observed by the Prommak design method, the correlation level between the number of users and traffic activities over the AP varies in accordance with the building usage space [3]. In the Prommak design approach, the traffic pattern is classified into two categories according to its data rate and activity level: Private sub-area and Public sub-area. Users in the private sub-areas such as graduate student offices and dorm rooms usually demand high data rates and maintain continuous activity, where in the public sub-areas such as classrooms, auditoriums, and cafeterias with irregularly scheduled usage, the data rate demand is lower and the activity level is sporadic. The Prommak method employs both SA and TS at different phases of the design. SA is used to produce frequency assignments that minimize interference, while TS is incorporated to reduce violations in

both the data rate demand and signal coverage by adjusting the AP locations and power levels. The method was tested on three different networks: the fourth floor of SIS building (SIS4), multi-floor SIS building, and the Hillman library (HL1), University of Pittsburgh. Compared with the exhaustive search of all possible grid locations (according to Prommak [3], the search space is as large as 98,611,128 combinations of AP location and power in the SIS4 case), TS is able to reduce the search time from 546 to 0.33 seconds for the SIS4 network and in the HL1 network from 10,239 to 84 seconds [3]. These results earn the Prommak method one the best candidates for WLAN designs. However, since the AP cost is not involved in the process, neither the Prommak or the Hills method is practical for the outdoor designs in which the BS cost must be minimized.

2.5 SUMMARY

Wireless network designs can be classified according to their objectives: Coverage, Capacity, Cost, and Interference. The coverage-based design is suitable in low user-density environments for which coverage is the minimum requirement. The capacity objective is effective when higher traffic concentration is present, but on the condition that cost is not a factor. The cost-objective is appropriate for outdoor designs where minimizing cost is necessary. The interference objective is more appealing for WLAN designs since minimizing interference is more important than maximizing either the coverage or capacity. Most design methods discussed here incorporate various optimization techniques such as SA, TS, and GA to reduce the computation time. However, none focuses on reducing the computational requirement directly by minimizing the number of candidate locations for placing BSs or Test Points (TPs). In determining TPs, all the existing design methods discussed in this paper settle on a fix number of grid points [25], [28], [3], [30], [2], [35], or occasionally on random points [1], but do not know if the optimal solution is included or not. Despite of its simplicity, working over the entire grid points may add extra complexity to the design, and the computational requirement may increase unnecessarily.

3.0 PATH LOSS MODELS

Wireless signals suffer attenuation caused by reflection, diffraction, and scattering from the surrounding environment. Signal attenuation between the transmitter and the receiver is referred to as a path loss. In the presence of noise (e.g, thermal), the transmitter must transmit at sufficiently high power to compensate for the path loss so that the receiver can receive and demodulate the intended signal correctly. Generally, path loss is a function of both the signal frequency and the distance between two communication entities. However, factors such as buildings, streets, and moving objects also largely affect the path loss characteristic as well. Effective network designs require accurate path loss models to predict the signal coverage without actual measurement (i.e. drive test). However, one given path loss model may be appropriate for one particular environment type, but may not be suitable for the others. Models such as the log normal, the two ray, and Okumura-Hata have been proved to accurately predict the path loss in cellular network environments, while the ray-tracing technique has gained a growing interest among researchers to provide higher precision and scalability, at the expense of increased computation. The next section explains these models in more details.

3.1 LOG NORMAL MODEL

For a relatively flat area, the path loss P_{ij} (in dB) between user$_i$ and BS$_j$ increases monotonically as a function of the separation distance r_{ij}:

$$P_{ij} = P_0 + 10\alpha \log_{10} r_{ij}, \tag{3.1}$$

where P_0 is the path loss at 1 m, and α is the path loss exponent. The α value typically ranges from 2 (for free space) to 6 (for heavily built-up areas due to extra attenuation from buildings). Path loss in (3.1) is an average value observed over a long period of time. However, the actual path loss fluctuates due to reflection, refraction, and scattering from moving objects, buildings, mountains etc. Generally, the fluctuation in path loss (shadow fading) is modeled as a log-normal random variable ζ with zero mean and standard deviation σ, then

$$P_{ij} = P_0 + 10\alpha \log_{10} r_{ij} + \zeta_j, \tag{3.2}$$

where ζ_j represents a shadowing loss of BS_j. For cellular networks, shadowing losses between two different BSs and a given user are correlated. To account for this correlation, Viterbi [40] assumes that ζ_j is composed of the near field component (ξ) common to all BSs and the component belonging to a given BS_j (ξ_j), in which

$$\zeta_j = a\xi + b\xi_j, \;\; a^2 + b^2 = 1, \tag{3.3}$$

$$\text{E}\left\{\xi\xi_j\right\} = \text{E}\left\{\xi_j\xi_{j'}\right\} = 0, \;\; \forall j \neq j' \tag{3.4}$$

where a and b are the propagation specific constants. $\xi, \xi_j, \xi_{j'}$ variables are also log-normal distributed with the same mean and standard deviation as ζ_j. Experiments show that the path loss exponent $\alpha \approx 4$, the shadowing loss standard deviation $\sigma \approx 8$, and the constant $a = b = 1/2$ are effective and accurate for most open area scenarios [40].

3.2 TWO RAY MODEL

Path loss (in dB) from the log-normal model above increases linearly as a function of distance. However, in microcell environments where the BS radius is usually less than 1 km., users located near the BS are likely to receive the signal directly without interference from the ground-reflected wave, and therefore path loss is less severe (i.e., smaller path loss exponent α). According to the two ray model, a break point is the distance where the ground reflected

wave becomes out of phase relative to the direct wave, resulting in the destructive sum and a considerably higher path loss exponent. Before the break point distance, the path loss exponent n_1 is approximately 2 as a result from free space propagation mechanism, after the break point greater path loss exponent n_2 between 2 and 7 is observed [36]. The break point distance d_f is a function of frequency $1/\lambda$, the receiver antenna height h_m, and transmitter antenna height h_b,

$$d_f = \frac{1}{\lambda}\sqrt{(\Sigma^2 + \Delta^2)^2 - 2(\Sigma^2 + \Delta^2)(\frac{1}{\lambda})^2 + (\frac{1}{\lambda})^4}, \tag{3.5}$$

where $\Sigma = h_b + h_m$ and $\Delta = h_b - h_m$. Figure 3.1 illustrates the two ray model path loss as a function of distances.

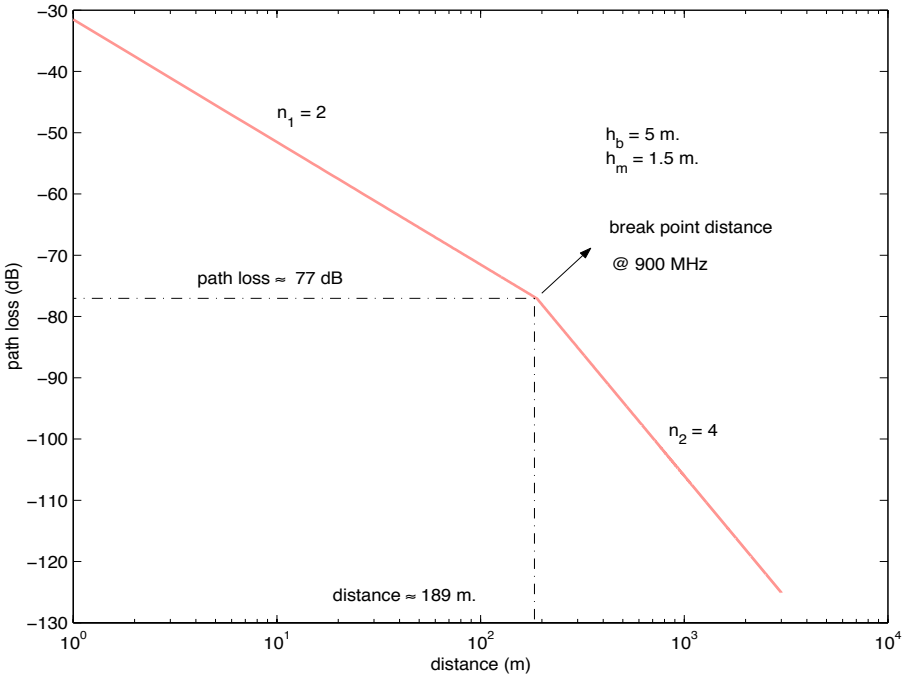

Figure 3.1: Two slope propagation path loss at 900 MHz. X axis represents the distance between the transmitter and the receiver. The transmitter height h_b and the receiver height h_m are = 5m and 1.5m respectively. The break point distance is ≈ 189 m.

The path loss is then a function of the break point distance which is expressed as:

$$P_{ij} = P_0 + \zeta_j + \begin{cases} 10n_1 \log_{10}(r_{ij}), & r_{ij} \leq d_f \\ 10n_2 \log_{10}(r_{ij}/d_f) + 10n_1 \log_{10} d_f, & r_{ij} > d_f \end{cases} \tag{3.6}$$

The two ray model can be relatively simple and accurate when applied to the microcell networks with LOS propagation — i.e, urban areas with BSs located along the street grid intersections. As an example, Alejandro [34] applied the two ray model to calculate the ratio between the maximum BS radius and the break point distance that maximizes the capacity.

3.3 OKUMURA-HATA MODEL

For macrocell networks, where the BS radius usually ranges between 1 to 20 km., man-made structures — such as cluster of buildings, street orientations, and highways — also greatly influence radio propagation and the path loss. Attenuation from city infrastructures is included as the initial loss irrespective of the distance, and the subsequent path loss follows a simple log-normal model formula. Okumura & Hata [41] conducted a series of experiments to determine the initial loss and the path loss exponent as a function of signal frequency for various city sizes based on the empirical field measurement data. In general, for a carrier frequency f_c (MHz), the path loss is expressed as:

$$P_{ij} = 69.55 + 26.16 \log_{10} f_c - 13.82 \log_{10} h_b - a(h_m) + (44.9 - 6.55 \log_{10} h_b) \log_{10} r_{ij} + \zeta_j, \quad (3.7)$$

where $a(h_m)$ represents the effective receiver height. For small or medium cities,

$$\begin{aligned} a(h_m) &= (1.1 \log_{10} f_c - 0.7) h_m - (1.56 \log_{10} f_c - 0.8), \\ & \quad 1 \le h_m \le 10 \text{ m}, \ 150 \le f_c \le 1500 \text{ MHz}. \end{aligned} \quad (3.8)$$

Alternatively, for large cities,

$$a(h_m) = \begin{cases} 8.29(\log_{10} 1.54 h_m)^2 - 1.1, & f_c \le 200\text{MHz} \\ 3.2(\log_{10} 11.75 h_m)^2 - 4.97, & f_c \ge 400\text{MHz} \end{cases} \quad (3.9)$$

The first two terms in (3.10) account for the average path loss at 1 km. The last term accounts for additional loss as a function of distances, and the path loss exponent ranges from 3 - 3.5, depending on the carrier frequency. The Okumura-Hata model is accurate

in frequency band ranging from 150 - 1500 MHz [41] . To account for additional losses at higher frequencies, the COST 231 model extended the Okumura-Hata model aiming for PCS applications (f_c = 1.5 - 2.0 GHz), and the resulting path loss formula changes to:

$$P_{ij} = 46.3+33.9\log_{10} f_c-13.82\log_{10} h_b-a(h_m)+(44.9-6.55\log_{10} h_b)\log_{10} r_{ij}+C+\zeta_j, \quad (3.10)$$

where C is the environment correction factor. $C = 0$ dB for medium cities and suburban areas, and $C = 3$ dB for metropolitan centers.

3.4 ATTENUATION FACTOR MODEL

Both the two-ray and Okumara-Hata models are designed for outdoor environments where path loss is mostly the result from large object reflectors and scatterers such as buildings, and streets. However, the path loss mechanism is different when signal propagates inside the building. There, the radio signal is likely to penetrate through several walls and floors before reaching at the receiver as shown in figure 3.2. The signal loses its energy each time it encounters the obstructions (i.e, partitions), resulting in larger path loss per distance traveled. To account for these losses, the attenuation factor model accounts for each obstruction with different attenuation values ranging from $1-5$ dB for soft partitions, and $5-20$ dB for hard partitions [42]. The path loss is calculated by summing up the attenuation factor from different types of floors and walls standing between a virtual straight line connecting the transmitter and the receiver. If $k_{fi'}$ and $k_{wi'}$ are the number of floor type i' and the number of wall type j', $L_{fi'}$ and $L_{wj'}$ are the attenuation factors respectively, then the path loss is

$$P_{ij} = P_0 + 10n\log r_{ij} + \sum_{i'} k_{fi'}L_{fi'} + \sum_{ij} k_{wj'}L_{wj'} \quad (3.11)$$

As shown in figure 3.2, there are three types of walls between the receiver i and transmitter j: a cubicle, a brick wall, and two wood walls, resulting in

$$P_{ij} = P_0 + 10n\log r_{ij} + L_{w1} + L_{w2} + 2L_{w3} \quad (3.12)$$

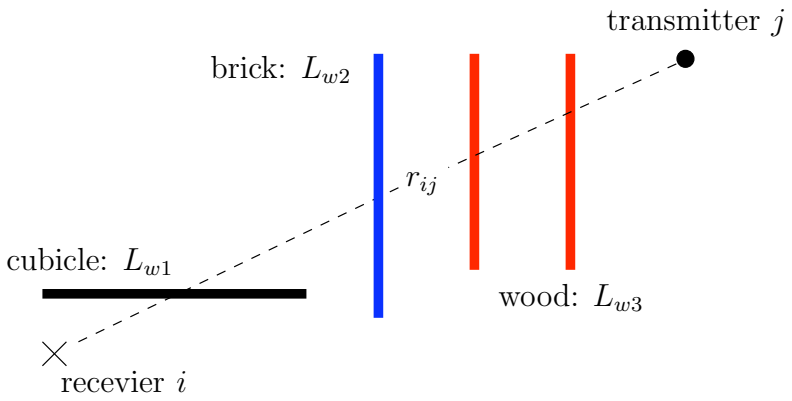

Figure 3.2: Attenuation Factors of Wall Types

Accuracy of the attenuation factor model relies on estimation of loss characteristic of various building materials. Some estimation data is difficult to obtain in places like elevator shaft, requiring further measurement and calibration. The attenuation factor model is used in the Prommak design method [3].

3.5 RAY TRACING TECHNIQUE

Path loss models discussed so far are based on approximations, ignoring specific details of surrounding environments (e.g, building facets, street corners, etc). Ray tracing is a technique adapted from the image rendering method in computer graphics to improve the path loss accuracy and to provide scalability for any site specific calculation. The ray tracing technique replaces electromagnetic field radiation by using infinitesimally small tubes called "rays". In ray tracing, rays always travel in straight lines, and only direct, reflected and refracted rays are considered. Building and terrain surfaces are represented by the polygonal plane facets (e.g, building walls with a four-sided facets, ground with a single facet) and are stored as vectors (e.g, Cartesian coordinate in a database [43]).

The ray tracing technique has two popular implementation methods: imaging method and ray-shooting method. The imaging method constructs a reflected ray from a virtual source point or an image behind the reflecting panel. The first order image then becomes a virtual source point for the second order image and its corresponding reflected ray, and so on until the ray bounces off with the maximum number of reflections. The resulting images are tied with the building layouts and the original source location, not the receiver position. Images are stored in a tree-liked database according to their order of being the virtual source point as shown in figure 3.3 [44]. The image is invisible if its reflecting point is not in the illuminating zone of the next lower order image (i.e, obstructed by other objects). Invisible images is prohibited to form the virtual source point and are deleted from the database. Checking the image visibility is difficult and requires complex computations [44]. Therefore, the imaging method is suitable in scenarios where only a few dominant reflectors are present (e.g., microcell networks). Figure 3.3 illustrates the imaging method.

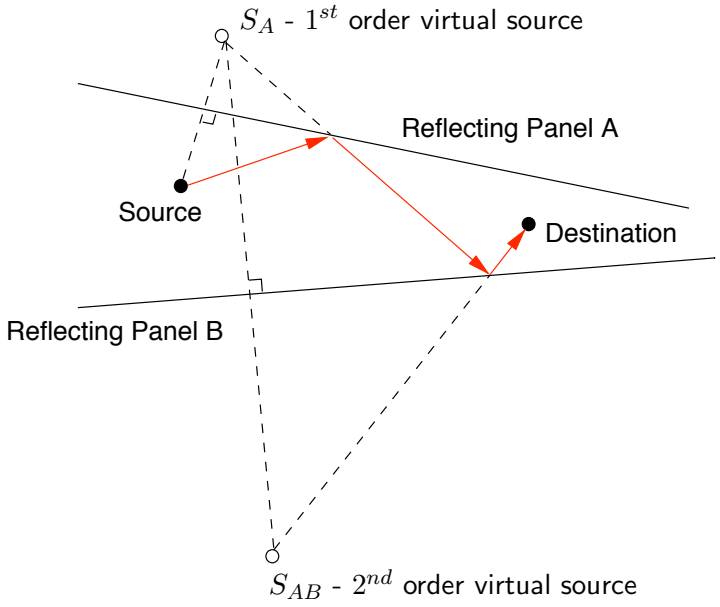

Figure 3.3: Ray Tracing - Imaging Method with the two reflecting panels (A, B) and their virtual sources S_A and S_{AB} respectively (maximum number of reflections = 2).

Alternatively, another approach to implement the ray tracing technique sends out a pin cushion of rays from the single point source, covering all directions in space — the ray

shooting method. Ray paths are traced until they intersect a sphere of radius R, which is centered at the receiver point. An oversized reception sphere may receive the same ray path twice, while an undersized reception sphere may not receive any ray at all [45]. To correctly receive rays, the size of the reception sphere must be

$$R = \alpha d/\sqrt{3}, \tag{3.13}$$

where α is the angular separation between two adjacent rays at the source, d is the total path length from the transmitter to the receiver [45]. Figure 3.4 shows the reception sphere of the ray shooting method.

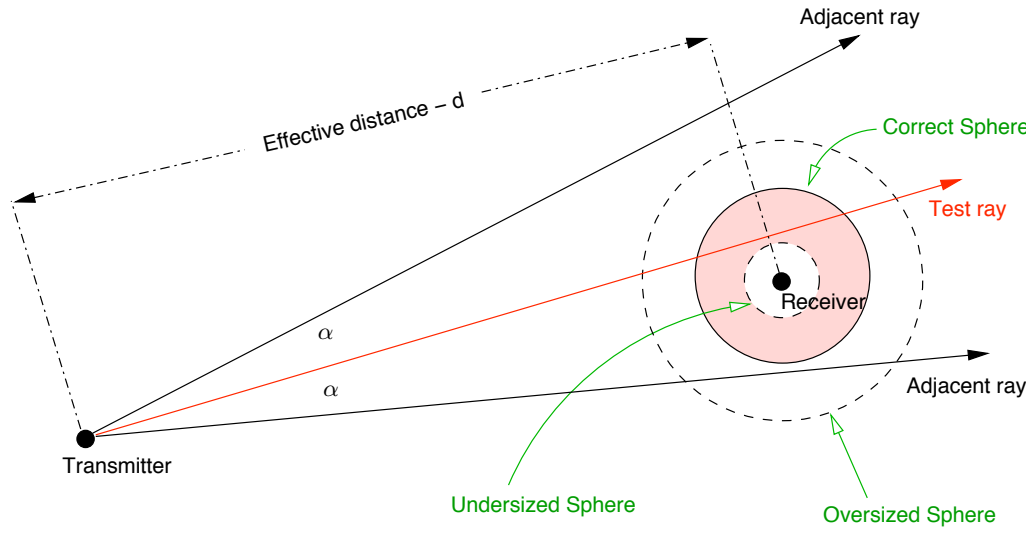

Figure 3.4: Ray Tracing - Ray Shooting Method showing a correct, oversized, and undersized reception spheres with ray launching angle = α.

Unlike the imaging method, the ray shooting method simply traces each ray path and determines whether it intersects the reception sphere or not. The number of reflectors are less likely to affect the computation requirement compared to the imaging method. Thus, the ray shooting method is more favorable when applying to areas with many irregular reflectors such as the indoor environment [46]. However, since a large number of ray paths must be traced to obtain the accurate result, the ray shooting method often requires longer execution time [44]. Nonetheless, the accuracy of both the imaging and ray shooting methods depends

heavily on details of the site specific layout and the choice of its electrical properties (e.g, reflection and refraction parameters).

Path loss models are important for wireless network designs. Accurate path loss calculation improves the design result and eliminates the need for further calibration. The log-normal model is simple but only accurate when applying to open areas. The two-ray model is the extension of the log-normal model and is suitable in the LOS microcell environments. Unlike the log-normal and the two ray models, both the Okumura-Hata and the COST 231 models are derived from the empirical data and are suitable for large cell size covering areas upto 20 km in radius. Lastly, to further improve the accuracy, one may apply the ray tracing technique, but must concern with the tradeoffs for increasing computational requirement and difficulty in preparing input data.

4.0 OPTIMIZATION HEURISTICS

The BS placement problem is known to be NP-hard (Non-deterministic Polynomial-time hard) [1]. The computational requirement for solving the problem grows exponentially as the problem size increases [3] (i.e, increasing number of BSs). As illustrated in the Prommak WLAN design, one example problem requires approximately $\approx 10^{19}$ searches to locate the optimal placement of 6 BSs [3]. Thus, employing the exhaustive search technique may not be effective and converge to the optimal solution may not be achieved in a reasonable time period when dealing with large size problems. To facilitate the search and to reduce the computational time, optimization heuristics such as Simulated Annealing (SA), Tabu Search (TS), and Genetic Algorithm (GA) were applied with different degrees of complexity and effectiveness — depending on the problem structure. As examples, The Anderson & McGeehan design used SA to determine the optimal BS locations to achieve the maximum coverage [2]. The Lee and Gang design employed TS to minimize the number of BSs in CDMA systems [38], while the Weicker design applied GA to minimize both the number of BSs and signal interference simultaneously [28]. Results from the Amaldi design [1], the Akl design [26], and the Lee and Gang design [38] are validated by comparing with the results from the IP branch and bound algorithm. The following sections further explains these heuristics.

4.1 SIMULATED ANNEALING

Simulated Annealing (SA) heuristic was proposed by Kirkpatrick [47] as a meta-algorithm. The problem with simple search heuristics such as the hill climbing method is that, the search always proceeds toward a better solution, consequently the search is bounded and likely to get caught in local optima. SA avoids being trapped in local optima by guiding the search to explore other regions of the search space more thoroughly. The logical idea of SA is similar to a metallic annealing process in which the heat allows atoms to escape from a defecting formation (local optima), and a proper cooling stage creates a perfect resulting crystal. SA imitates the same idea by accepting moves that produce inferior solutions with some probabilities, increasing the opportunity for the search to find better solutions in other regions. The choice of neighborhood solutions (candidate solutions) is flexible. As an example, the neighborhood solution in TSP (Traveling Saleman Problem) is created by swapping a pair of adjacent cities in the current tour, but can be to any pair of arbitrarily random cities as well. The transition probability is a function of the annealing temperature. If $C(s)$ represents the solution merit (cost), s and s' are the current solution and neighborhood solution respectively, the classic transition probability with respect to the temperature T_k at iteration k^{th} is

$$\mathsf{Pr}[\mathsf{accept}\ s'] = \mathsf{Pr}[e^{-(C(s')-C(s))/T_k} > \mathsf{random}(0,1)]. \tag{4.1}$$

Initially, the temperature is set to a high value (T_0), allowing high transition probability, but continues to decrease until reaching zero, then the algorithm stops. The annealing schedule is usually set as:

$$T_k = \alpha T_{k-1}, \tag{4.2}$$

where α is a constant. The pseudo-code to implement SA is described below:

Simple Simulated Annealing

```
s = initial solution;
k = 0;  i = 0;
while T_k > T_min do
      while i < i_max do
            s' = new neighbor solution of s;
            if C(s') < C(s) or e^{-(C(s')-C(s))/T_k} > random(0,1)
              s = s';  (accept a new neighborhood solution)
            i = i + 1;
      done;
      k = k + 1;
      T_k = αT_{k-1};
done;
```

$i_{\max}$ is the maximum number of iterations at each temperature T_k, where $T_{\min} \approx 0$ is the final temperature. The neighborhood solution structure, the transition probability, the annealing schedule are subject to different implementations. There are no specific general rule of thumps for any given problem. As examples, the Prommak method for WLAN designs employed SA to minimize interference caused by frequency assignments, however the Prommak's SA approach decreases $i_{\max}$ inversely proportional to T_k (normally $i_{\max}$ is constant) [3]. On the other hand (as discussed in the previous chapter), the annealing schedule in the Lee design method for TDMA networks is an exponential function of the previous temperature T_{k-1}, not a linear function. Choosing the right parameters for any given problem is thus critical to the algorithm effectiveness and its convergence property.

4.2 TABU SEARCH

Similar to SA, a Tabu Search (TS) heuristic is also classified as a meta-algorithm that can incorporate, modify, and guide any local search strategy to reaches other regions containing better solutions — not trapped in local optima. TS was introduced by Glover and Laguna [48] to solve a variety of combinatorial optimization problems ranging from job scheduling, telecommunications network design, to general mixed integer programming. Unlike SA, TS relies heavily on the use of memory to deviate the search to explore other regions, which may be more interesting, and to eventually escape from local optima (compared to the probabilistic-based approach in SA). There are two types of memories used in TS: short

term and longer term memories. The short term memory (often in attributive forms — e.g, move attributes) records a move made previously, and classifies this move as tabu for some number of iterations (can be either static or dynamic). This tabu status prevents solution recycling and helps propelling the search to escape from local optima. On the other hand, the longer term memory (often in explicit forms — e.g, some elite solutions) maintains parts of the solution properties (or completed solutions) previously visited and exploits information gathered to diversify the search to unexplored regions — diversification strategy, or to intensify the search to create a new solution by combining and incorporating good attributes of the recorded elite solutions — intensification strategy. The neighborhood structure of the current solution s — $\mathcal{N}(s)$, can vary in many forms. Normally, a candidate list strategy (e.g, specify a threshold for move quality) is used to reduce the number of neighborhood solution evaluations. A pseudo to implement TS is described as follows:

Simple Tabu Search

$s =$ initial solution; $i = 0$;
while $i < i_{\max}$ do
    if all moves $s \to s', \forall s' \in \mathcal{N}(s)$ are tabu
      $s = s' \in \mathcal{N}(s)$ with a move that is least tabu;
    else
      $s =$ optimum$[s' : \forall s' \in \mathcal{N}(s)$ with non-tabu moves$]$;
    end;
    a move made becomes tabu for *tabu tenure* iterations;
    $i = 1 + 1$;
done;

As shown above, if all moves to $s' \in \mathcal{N}(s)$ are tabu, TS invokes all tabu status of the least tabu move and selects the move accordingly — aspiration by default. In fact, TS requires more problem specific knowledge than SA in order to exploit the short term (e.g, tabu tenure, candidate list) and longer term memories (intensification and diversification) effectively. The neighborhood structure $\mathcal{N}(s)$, tabu classification, aspiration criteria, etc., can be adjusted and calibrated to fit the context of a particular problem. As examples when TS is applied to solve the BS placement problem, the Lee and Gang design method created $\mathcal{N}(s)$ by dropping a BS from or adding a new BS to the current solution s, while the Prommak design method formed $\mathcal{N}(s)$ by adjusting both the BS location and power. The Prommak method also includes an aspiration criteria that forgives any tabu moves yielding the best solution.

4.3 GENETIC ALGORITHM

A Genetic Algorithm (GA) heuristic also falls into a class of meta-algorithms similar to SA and TS. However, GA works differently by mimicking a biological evolution in nature to solve NP-Hard combinatorial problems. Similar to biological processes, GA employs evolutionary operators such as natural selection, mutation, and crossover (or recombination) to produce a new set of feasible solutions (population) in each iteration (generation). The natural selection process chooses a pool of individuals (solutions) either to represent as a group of parent individuals of the next generation based on the fitness value. Evaluating the fitness is problem specific and is critical to the algorithm convergence property (i.e, if handled poorly, it is likely that the algorithm will converge to local optima [49]). The mutation process is used to modify certain properties or attributes of parent individuals to create a new offspring, while the crossover process combines parts of both parent individuals to produce a new individual. GA invokes either the mutation or the crossover process randomly with some probabilities (e.g, the Weicker design method in the last chapter used 0.6 and 0.4 for mutation and crossover probabilities respectively). In summary, the pseudo algorithm to implement GA is described as follows:

Simple Genetic Algorithm

```
create a set of feasible solutions --- first generation population;
i = 0;
while i < i_max do
      - evaluate each individual fitness in the entire (or parts of) population;
      - select the best individuals as the candidate parents according to the fitness;
      - randomly crossover a pair of parent individuals with some probabilities;
      - randomly mutate parent individuals;
      i = i + 1;
done;
```

where $i_{\max}$ is the maximum number of generations created. GA implicitly inherits unstructured form memory similar to TS (partially structural) through its evolutionary processes (compared to memoryless SA). Similar to TS (e.g., tabu attributes), the randomized memory operations (mutation and crossover) help GA to escape from local optima, generations after generations (regions after regions), in which the surviving generation contains the best individuals (which is approximated to be the global optimal solution).

In conclusion, SA, TS, and GA are among the best candidate heuristics for solving NP-Hard problems. Their applications are very board, covering many areas of interests from operation research designs to telecommunications network plannings. The common objective of these heuristics is to locate the global optimal solution in a reasonable time period, using various forms of techniques to escape from local optima: probabilistic-based in SA, tabu attributes in TS, and evolutionary processes in GA. Successful implementation of these heuristics requires careful parameter calibration (e.g, annealing schedule in SA, tabu tenure value in TS, crossover and mutation probabilities in GA) to match specific contexts of the problem structure. Results (i.e., BS placement) from SA, TS, or GA are often verified and compared with the optimal result from the branch and bound algorithm, described next.

4.4 BRANCH AND BOUND ALGORITHM

The branch and bound algorithm is a general search method widely used to solve mixed-interger NP-Hard problems [50]. The algorithm searches the entire space by recursively dividing the space into smaller subspaces (branching) until the optimal solution is found. Each subspace can be considered a node of the branch and bound tree, where the terminating node is reached when a certain condition or a threshold is met. However, not all subspaces are generated or searched. The algorithm limits (bounding) the number of subspaces or nodes by discarding the subspace which is infeasible and the subspace that can not produce the optimal solution (compared to the current best solution). In general, the branch and bound algorithm consists of three steps: selection of the subspace to process, calculating the bound, and branching. Figure 4.1 shows the example tree of subspaces created by the branch and bound algorithm.

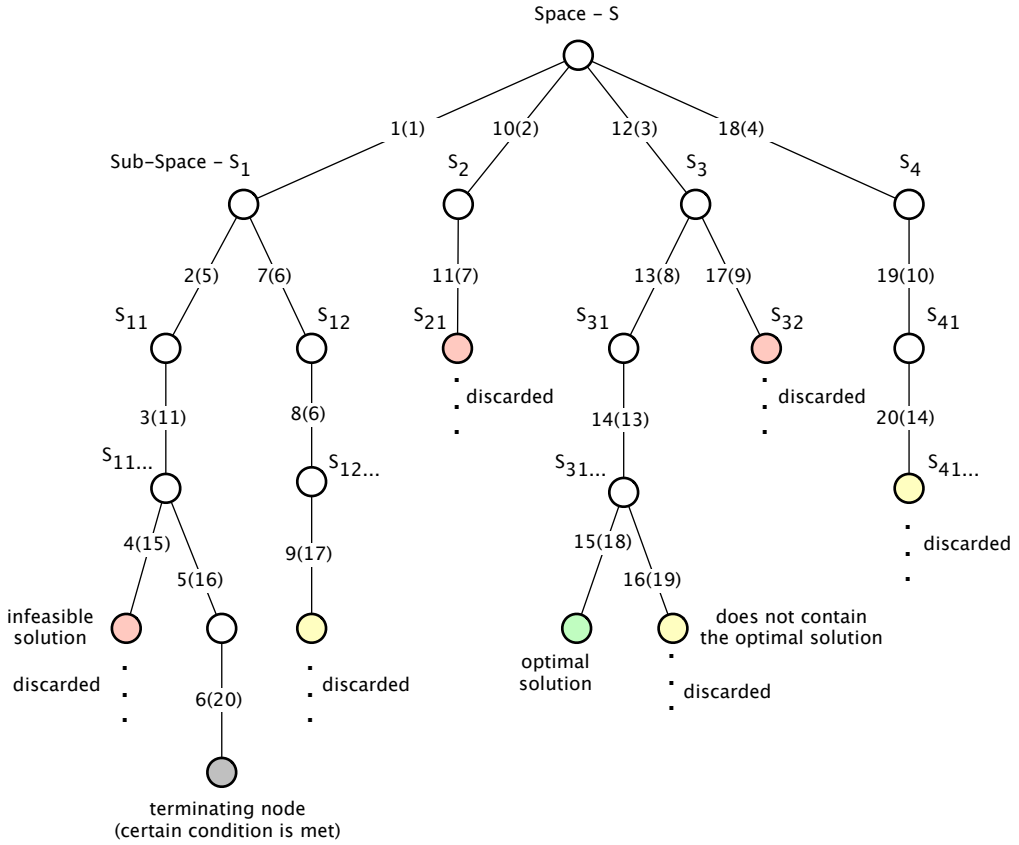

Figure 4.1: Tree of Subspaces. Each node except the root represents a subspace — red (infeasible subspace), yellow (subspace not containing the optimal solution), grey (a threshold or a condition is met), and green (optimal solution subspace).

Two strategies of subspace selection are also illustrated in figure 4.1 above: Depth First Search (DFS) and Breadth First Search (BFS). The sequence of DFS is marked by the first number in the middle of each branch, whereas the number in the parenthesis shows the sequence of BFS. As implied by the name, DFS traverses down the tree to the deepest level node as possible before backtracking. On the other hand, BFS keeps extending the search on the horizontal direction of the tree, in which all neighboring nodes at the same level are explored from left to right before traversing down to the node on the next level. DFS is flexible to implement through the use recursion and requires memory at most equal to the the maximum number of levels of the search tree multiplied by the maximum number of children of any node. The only disadvantage of DFS is that, if the optimal solution node is

located on the other end of the tree, DFS has to traverse all the nodes of all levels before reaching the optimal node. BFS, however, has only to traverse all the nodes to the level until the optimal node is found. Unfortunately, in terms of memory requirement or space complexity, BFS needs to keep memory of all the nodes traversed, therefore BFS works well only for narrow and deep trees, but not suitable and may be impractical for large trees. On the other hand, DFS works well for broad and shallow trees. Nevertheless, both BFS and DFS are not efficient when trees are broad and deep.

5.0 TEST POINT REDUCTION

The problem of locating the minimum number of Base Station (BSs) to provide sufficient signal coverage and to satisfy user demand, is often formulated in a manner that results in a mixed-integer NP-Hard (Non-Deterministic Polynomial-time Hard) problem. Solving a large size NP-hard problem (e.g., multiple BSs required) usually consumes time since the search space always increases exponentially with the problem size. Several well-known heuristics such as Simulated Annealing (SA), Tabu Search (TS), and Genetic Algorithm (GA) are incorporated to effectively solve the problem (as already discussed in chapter 2). However, implementing whether SA, TS or GA is not straightforward and is problem specific. Furthermore, convergence to the optimal solution, in most cases is sensitive to one or more parameters of the heuristics; a slight change in one parameter may produce unexpected results and optimal solutions are not guaranteed. Regardless of the heuristic used, convergence time always increases with the size of search space S. Search space, in the context of BS placement, is a collection of all solutions both feasible and infeasible; each solution may contain one or more BSs selected from a set of candidate locations or Test Points (TPs). The required number of TPs N ensuring optimal solutions, as later discussed, increases proportionally with the number of users and the size of service area. Since the size of search space S is an exponential function of N, S can grow very large when N increases, and in some cases so large that even with the fastest heuristic may never find the optimal solution in a reasonable time period. This chapter presents a method to reduce the number of TPs by employing the sweep and merge algorithm. The resulting number of TPs are substantially less than N, while optimal solutions are still maintained.

5.1 DEMAND NODE CONCEPT

Determining N and S requires information on both user distribution and design topology (service area). User distribution is often random in nature and hard to predict. Mobile users is assumed to be constantly moving and generating different types of traffic patterns based on applications they are running (i.e., voice call, web browsing, downloading, etc). As a consequence, it becomes virtually impossible to statically locate BSs and their fixed resources (i.e., bandwidth) to fully satisfy traffics from moving users, switching on and off different kinds of applications from time to time. However, it can be assumed that as one user departs an area, on average another user enters the area and demand remains constant. To relieve network designers from dealing with the complexity of traffic estimation, the demand node concept is introduced [25]. Demand node was previously used in network design methods such as the Prommak WLAN design [3], the Tutschku TDMA design [25], and the Almaldi CDMA design [1]. Employing the demand node not only accounts for user mobility, call behavior, and traffic distribution from both geographical and demographical variations, but also simplifies the design process substantially (section 7.3). The demand nodes are generated by discretizing the traffic load into points, where each point represents the center of an area containing a quantum of demand (e.g., a fixed number of call requests per time unit). To add granularity, the entire service area is replaced by a collection of $\Delta x \times \Delta y$ unit area elements. The traffic intensity $\lambda(x, y, t)$ (i.e., number of call attempts per unit time) in each unit area at time t is a function of user mobility or location probability $P_l(x, y, t)$, and the call arrival probability $P_a(t)$:

$$\lambda(x, y, t) = \int_{x}^{x+\Delta x} \int_{x}^{y+\Delta y} P_l(x', y', t) dx' dy' P_a(t) \quad \text{(calls)} \tag{5.1}$$

To estimate the location probability $P_l(x, y, t)$, network designers need to model user movement. User mobility models such as a Markovian model can be used. The model assumes a random walk for each individual movement (with a transition probability and actual human movement) to compute for $P_l(x, y, t)$. If $A(x, y, t)$ is the offered traffic at time t then[1],

$$A(x, y, t) = \lambda(x, y, t) \times \text{mean call duration} \quad \text{(Erlangs)} \tag{5.2}$$

[1]Network designers may omit the temporal variation t by considering only the busy hour traffic.

After determining the traffic load in each unit area, demand node are generated by recursively bisectioning the design area into two rectangles containing equal amounts of traffic load until the traffic of every tessellation piece falls below a threshold. The following pseudo algorithm summarizes the demand node generating function:

Demand Node Generating Function

```
void dnode(area){
      if traffic in area < threshold
        return;
      else
        turn partitioning line 90 degrees;
        dnode(left area);
        dnode(right area);
}
```

The resulting demand nodes are dense in areas with high traffic intensity and sparse in low traffic intensity areas. For a given design area of dimensions XY, if the demand node generating function produces M demand nodes, define

$$m_k = \text{demand node } k^{th}, \quad k \in \{1, \ldots, M\} \tag{5.3}$$

where m_k is located at X_k, Y_k, in which

$$0 \leq X_k \leq X \text{ and } 0 \leq Y_k \leq Y, \quad \forall k \in \{1, \ldots, M\} \tag{5.4}$$

where

$$X_k = i\Delta x, \text{ and } Y_k = j\Delta y, \quad i, j \in \{1, 2, \ldots\} \tag{5.5}$$

and A_k is the demand unit (e.g., in Erlangs, kbps, etc) of demand node m_k. Limitations of the demand nodes are discussed in chapter 8.

5.2 SEARCH SPACE SIZE

The size of search space influences both the effectiveness and outcomes of optimization algorithms. Generally, if N is the total number of TPs, and n^* is the minimum number of BSs required, it means that n^* different BSs can be placed at N different locations, it is equivalent to pick n^* different things out of N different things. Thus, the total possible number of candidate BS placement or the search space size S is

$$S = \binom{N}{n^*} = \frac{N(N-1)\dots(N-n^*-1)}{n^*!} \approx N^{n^*} \tag{5.6}$$

According to (5.6), S increases exponentially as N and n^* increase and could become very large. As an example, if $N = 100$ and $n^* = 5$, then the size of search space

$$S \approx 100^5 = 10^{10} \tag{5.7}$$

Depending on the distribution of demand nodes m_k, the minimum number of N that always guarantees the optimal solution may vary. The optimal solution (as later discussed) refers to any set of minimum number of BSs that can satisfy all the requirements of the network design (i.e., signal coverage, capacity, etc).

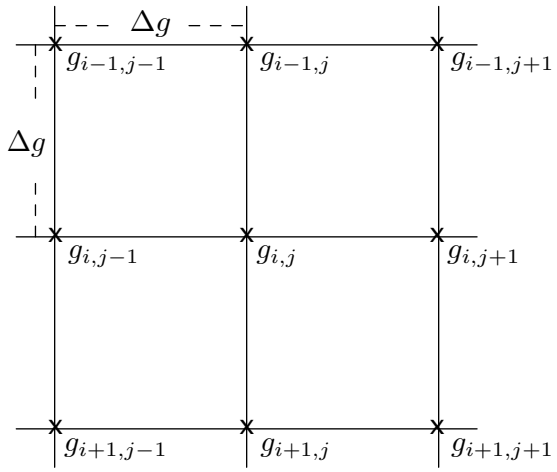

Figure 5.1: TPs as a matrix of grid points G

From (5.6), if TPs are continuous variables, $N \to \infty$ and so does the size of search space S. However, the continuous space may not be necessary for the optimal solution since the distribution of demand (according to section 5.1) is always discrete, and therefore may as well be replaced by a matrix of grid points G as shown in figure 5.1. The total number of grid points N is a function of the grid spacing Δg and the size of design area, and if the design area is assumed to be rectangular with width X and length Y, then

$$N = \left\lfloor \frac{X}{\Delta g} + 1 \right\rfloor \left\lfloor \frac{Y}{\Delta g} + 1 \right\rfloor = N_x N_y \tag{5.8}$$

where grid point $g_{i,j} \in G$ is located on coordinate $[x, y]$, where

$$g_{i,j} = [(i-1)\Delta g, (j-1)\Delta g] \tag{5.9}$$

$$G = \{g_{i,j}\}, \quad \forall i \in \{1, \ldots, N_x\}, \ \forall j \in \{1, \ldots, N_y\} \tag{5.10}$$

Selecting the proper TP grid spacing Δg is important. Too large Δg may produce suboptimal solutions, while too small Δg may result in too many unnecessary TPs (TPs which are not part of the optimal solutions). *To determine the maximum Δg (the minimum N) that always yield(s) the optimal solution(s), consider an arbitrary set of demand nodes in figure 5.2*

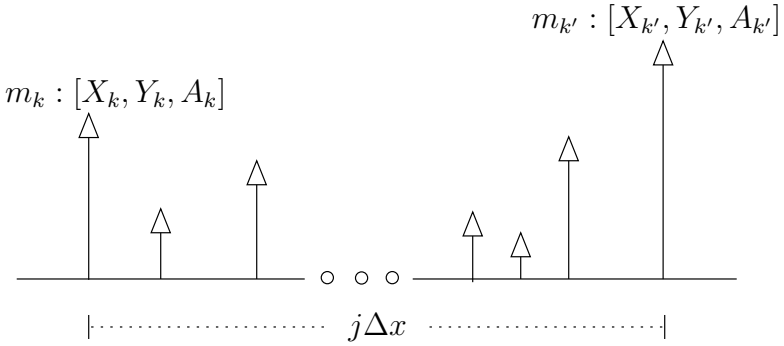

Figure 5.2: A given set of demand nodes. (all the demand nodes are on $y = Y_k = Y_{k'}$ plane)

Suppose that all demand nodes shown in figure 5.2 must be assigned to one BS. Also, assuming that the BS radius [2]

$$R = (|X_k - X_{k'}| + \delta)/2 \quad (\mathsf{m}) \tag{5.11}$$

[2] R represents the maximum distance allowed between a BS and any given demand node, presumably accounting the BS transmit power, receiver sensitivity, path loss, and fading.

where δ is any real number and must be ≥ 0 in order for the solution to exist. Thus, a BS must be placed at $[x, Y_k]$, such that

$$X_k + |X_k - X_{k'}|/2 - \delta/2 \leq x \leq X_k + |X_k - X_{k'}|/2 + \delta/2 \tag{5.12}$$

From (5.5), $X_k = i\Delta x$ thus

$$(i+j)\Delta x/2 - \delta/2 \leq x \leq (i+j)\Delta x/2 + \delta/2 \tag{5.13}$$

Equation (5.13) implies that, for any arbitrary set of demand nodes (i.e., as shown in figure 5.2), there can be many solutions depended on δ. The solution range decreases when δ decreases, and when δ reaches the lower bound ($\delta = 0$), only one solution exists

$$x = (i+j)\Delta x/2, \quad i, j \in \{1, 2, \ldots\} \tag{5.14}$$

Since x in (5.14) is the only possible solution, i and j are both integer numbers, therefore the resolution of x or equivalently, the possible TP grid spacing is

$$\begin{aligned} i\Delta g &= \Delta x/2 \\ \Delta g &= \Delta x/(2i), \quad i \in \{1, 2, \ldots\} \end{aligned} \tag{5.15}$$

According (5.15), the TP grid spacing Δg that always satisfies the optimal solutions must be no larger than half the demand node grid spacing Δx (for any i). Thus, the minimum number of grid points required to guarantee the optimal solution are

$$N = \left\lfloor \frac{2X}{\Delta x} + 1 \right\rfloor \left\lfloor \frac{2Y}{\Delta y} + 1 \right\rfloor \tag{5.16}$$

The resulting number of TPs from equation (5.16) can be quite large. As example, if $X = Y = 1000$ and $\Delta x = \Delta y = 10$, then

$$N = \left\lfloor \frac{1000}{10/2} + 1 \right\rfloor \left\lfloor \frac{1000}{10/2} + 1 \right\rfloor = 201^2 = 40,401 \tag{5.17}$$

However, the space is sparsely populated by solutions [3]. Thus, most TPs are redundant considering the optimal solution, and therefore can be ignored so that the computational requirement is reduced. The definition of optimality may differ when network environments change, and therefore must be discussed first.

5.3 DEFINITION OF OPTIMALITY

Optimality, for the BS placement, may refer to various design objectives. The coverage-maximized objective may be suitable when there are few enough users so that the network capacity is more than sufficient to support the demand. However, when the number of users and demand increases, several other requirements may also need to be satisfied. As a result, depending on the type of environment, the definition of optimality will vary and the importance of the various objectives will vary as well. *In this paper, the design criteria is to provide ubiquitous signal coverage and guarantee sufficient network capacity to all users (or demand nodes). The primary objective is then to minimize cost, which is in turn equivalent to minimizing the number of BSs, while satisfying the coverage, capacity, and interference requirements.* Given a set of N TPs, let

$$\text{initial set of TPs} = \{\text{TP}_1, \text{TP}_2, \ldots, \text{TP}_N\}$$

in which

$$\text{TP}_n = g_{i_n,j_n},\ n = \{1, \ldots, N\}$$

where g_{i_n,j_n} is a grid point at $((i_n - 1)\Delta g, (j_n - 1)\Delta g)$. Define

$$b_n = \begin{cases} 1 & : \text{ if a BS is installed at TP}_n \\ 0 & : \text{ otherwise} \end{cases} \tag{5.18}$$

Because wireless network standards are so diverse, in which both the technological aspects and implemation details can differ substantially (e.g., CDMA2000 vs WiFi), thus the design requirements may also differ accordingly. As examples, CDMA systems employ universal frequency reuse and exhibit a soft-capacity characteristic [51]. All users share the same bandwidth, and each contributes to the total noise seen by the others. The total noise then fluctuates with the number of users increase or decrease — so does the required SNR. Consequently, the cell size (the coverage radius) either shrinks or expands as a function of the number of users active at a given moment (cell-breathing effect) [51]. In comparison, static resource allocation schemes as implemented in FDMA/TDMA systems (AMPS, GSM, etc), dedicate part of the bandwidth to a given user for the entire communication session. The

maximum number of simultaneous connections between a given BS and its users are limited by the number of frequency channels or time slots allocated priori (hard capacity). The cell coverage is fix — bounded by path loss and the receiver ability to extract the intended signal. In general, whether the coverage is dynamic (CDMA) or static (FDMA/TDMA), still the SNR level at the receiver must be greater than the minimum target in order to guarantee error-free delivery and successful demodulation of information content [51].

$$\text{SNR} \geq \text{target level} \tag{5.19}$$

Generally,

$$\text{SNR} \;\propto\; \text{path loss} \;\propto\; d_{kn} = \text{distance between } m_k \text{ and } b_n$$

where an appropriate propagation model from chapter 3 may be used to compute the path loss for a particular type of environment, and R is the effective, maximum BS coverage radius (accounting for the path loss, fading, and the required target-SIR, then (5.19) can be rewritten as

$$d_{kn} = \sqrt{(X_k - (i_n - 1)\Delta g)^2 + (Y_k - (j_n - 1)\Delta g)^2} \leq R \tag{5.20}$$

If

$$m_{kn} = \begin{cases} 1 & : \text{ if demand node } m_k \text{ is assigned to } b_n \\ 0 & : \text{ otherwise} \end{cases} \tag{5.21}$$

Combining (5.20) and (5.21), the coverage requirement can be formulated as

$$m_{kn} \;\leq\; \max\,[0, \lceil R - d_{kn} \rceil] b_n, \tag{5.22}$$

$$\sum_{n=1}^{N} m_{kn} \;=\; 1, \;\; \forall k \in \{1, \ldots, M\} \tag{5.23}$$

Equation (5.22) ensures that each demand node m_k must be within the coverage radius R of b_n to which it is assigned, while (5.23) forces a unique assignment between $m_k \leftrightarrow b_n$. Similarly for the capacity requirement, if C represents the maximum capacity that the BS can support, the capacity requirement is also formulated as

$$\sum_{k=1}^{M} m_{kn} \leq C, \;\; \forall n \in \{1, \ldots, N\} \tag{5.24}$$

In summary, the optimal solution is achieved when

$$\begin{aligned} \min \quad & \sum_{n=1}^{N} b_n \\ \text{subject to} \quad & m_{kn} \leq \max\left[0, \lceil R - d_{kn} \rceil\right] b_n \\ & \sum_{n=1}^{N} m_{kn} = 1, \ \forall k \in \{1, \ldots, M\} \\ & \sum_{k=1}^{M} m_{kn} \leq C, \ \forall n \in \{1, \ldots, N\} \end{aligned} \tag{5.25}$$

(5.25) is in fact the design problem formulation. It is important to note that, in soft handoff individual user keeps a list of candidate BSs for handoff purposes, however data transmission only occurs between the user and one active BS, thus the soft handoff in CDMA systems does not impact the problem formulation in (5.25). In section 7.3, R and C are computed in such a way that they approximately reflect all the physical properties and requirements of any given wireless system. Using R in place of a complex link-budget calculation (i.e., path loss and SNR) enables flexible implementation of the TP reduction algorithm, while using a constant C is necessary for the design problem formulation. For TDMA or WLAN networks, C is pre-allocated (i.e., 11 Mbps for 802.11b) and fixed. However, the capacity in CDMA systems is soft and subject to current interference level in the cell. Thus, C must be approximated either by setting the interference threshold and assuming uniform loading distribution (where C may be overestimated [39]) or by averaging the most probable value from experimental results [52].

5.4 SET COVERAGE COMPARISON

As discussed in section 5.2, the required number of TPs (grid points) guaranteeing the optimal solution is a function of the demand node distribution and can be quite large. Section 5.2 also discussed why not all TPs are necessary. For each grid $g_{i,j}$, let define

$$z_{i,j} = \{m_k | \sqrt{(X_k - (i-1)\Delta g)^2 + (Y_k - (j-1)\Delta g)^2} \leq R\} \tag{5.26}$$

be a set of demand nodes that are within the coverage radius R from grid $g_{i,j}$, Consider the coverage requirement, if

$$|z_{i',j'}| < |z_{i,j}| \text{ and } z_{i',j'} \subset z_{i,j} \tag{5.27}$$

for example

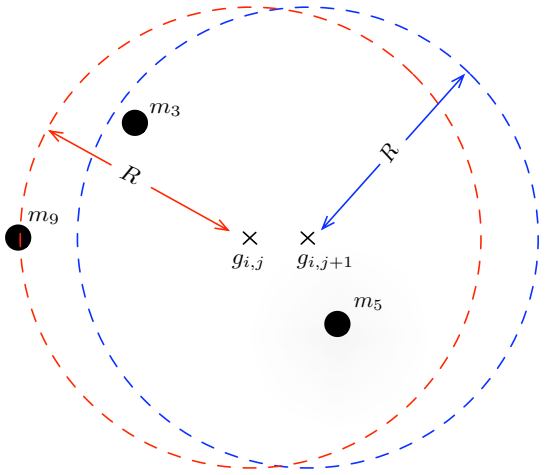

Figure 5.3: Set Coverage Example

The set coverage $z_{i,j} = \{m_3, m_5, m_9\}$, while $z_{i,j+1} = \{m_3, m_5\}$,

$$|z_{i,j+1}| = 2 < |z_{i,j}| = 3 \text{ and } \{m_3, m_5\} \subset \{m_3, m_5, m_9\}$$

Since using only grid $g_{i,j}$ is sufficient to cover the demand nodes $\{m_3, m_5, m_9\}$, therefore grid $g_{i,j+1}$ which covers a subset of demand nodes $\{m_3, m_5\}$ is not necessary for the coverage

requirement and can be removed. For N grid points, the total number of set comparisons could be as large as

$$\binom{N}{2} = \frac{N(N-1)}{2} \approx N^2$$

However, not all comparisons are necessary. There is no need to compare the set coverage of grid points separated by a distance greater than $2R$, since the two sets are always mutually exclusive. Alternatively, a raster sweep approach is used instead.

5.4.1 Raster Sweep

Raster sweep in a method to compare the set coverage of grid points in a more efficient manner than the exhaustive paring. The goal is to compare all adjacent grid points in all directions (i.e., horizontal, vertical, and diagonal), and to remove any grid point that contains a subset of demand nodes (compared to the adjacent one) — sweep. Four possible sweeping directions are used as shown in figure 5.4.

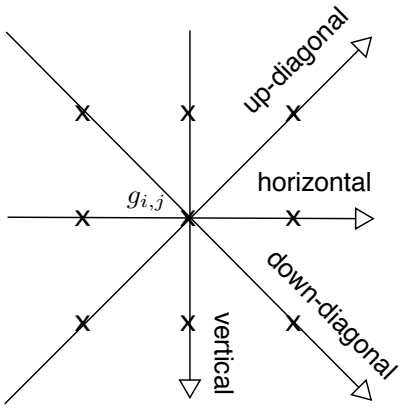

Figure 5.4: Four Possible Sweeping Directions

The horizontal sweep starts from the first grid $g_{1,1}$ at the top left corner, compares it to the adjacent grid $g_{1,2}$, moves on to the next grid on the right and so on until reaching the rightmost grid g_{1,N_x}, then goes down to the lower grid point, turns around to the opposite direction toward the leftmost grid on the second row and so on, where the raster patten

is then repeated until all adjacent grids on the horizontal direction are compared. The horizontal sweep pattern is shown in figure 5.5. The sweep process maps the matrix of grid

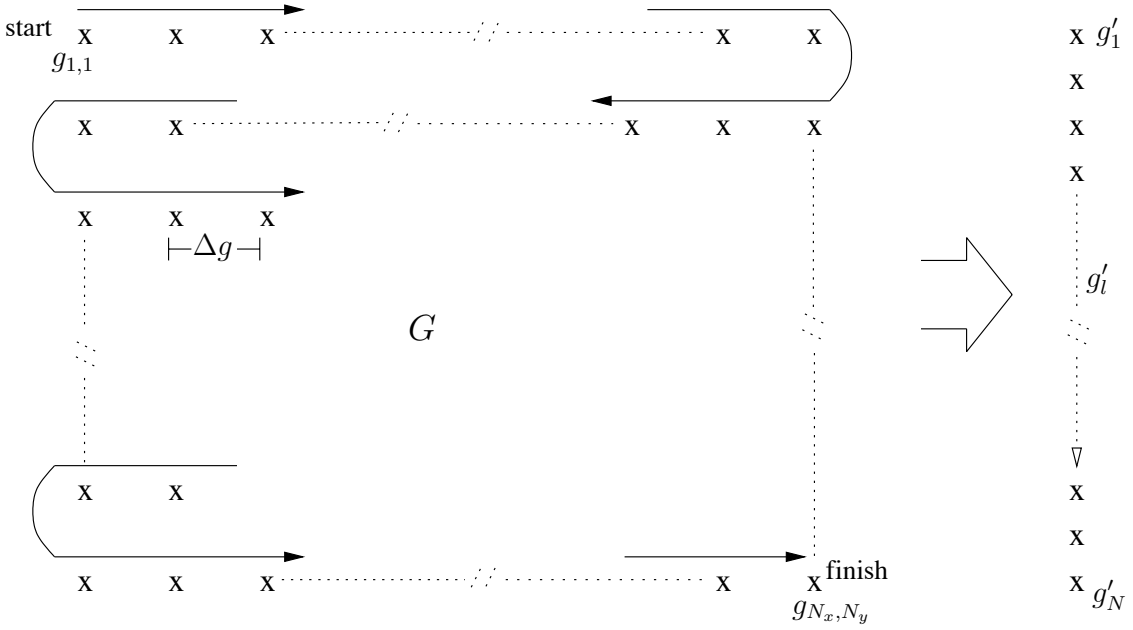

Figure 5.5: Horizontal Raster Sweep: from grid $g_{1,1}$ to g_{N_x,N_y}

points $g_{i,j}$ to a virtual straight line, where g'_l represents the l^{th} grid point on the line, and z'_l represents the corresponding set coverage. With a line-mapped series of grid points, the sweep process can be linearly executed as summarized in the pseudo code below

```
for l = 1 to N,
    • compare z'_l and z'_{l-1}
    if |z'_l| < |z'_{l-1}| and z'_l ⊂ z'_{l-1},
        delete g'_l;
    else if z'_l ≐ z'_{l-1} and g'_{l-1} was previously deleted,
           delete g'_l;

    • compare z'_l and z'_{l+1}
    if |z'_l| < |z'_{l+1}| and z'_l ⊂ z'_{l+1},
        delete g'_l and all of the preceding grids that are equivalent;
end;
```

The same pseudo code is applied to the other line-mapped of grid points from both the vertical and diagonal sweeps as shown respectively in figures 5.6 and 5.7.

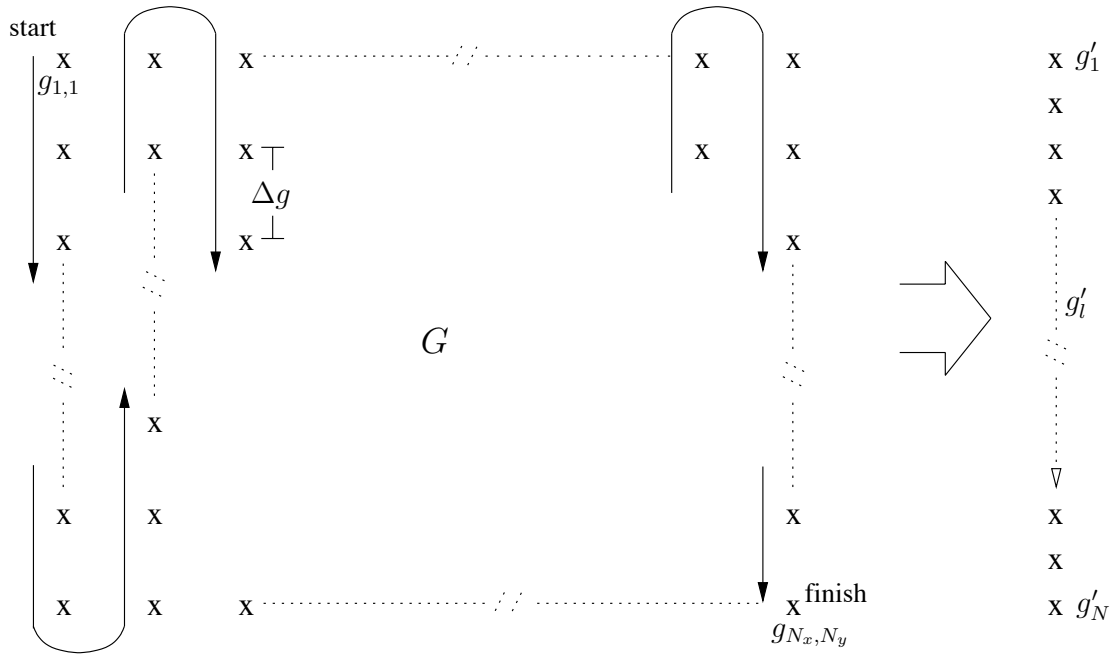

Figure 5.6: Vertical Raster Sweep: from grid $g_{1,1}$ to g_{N_x,N_y}

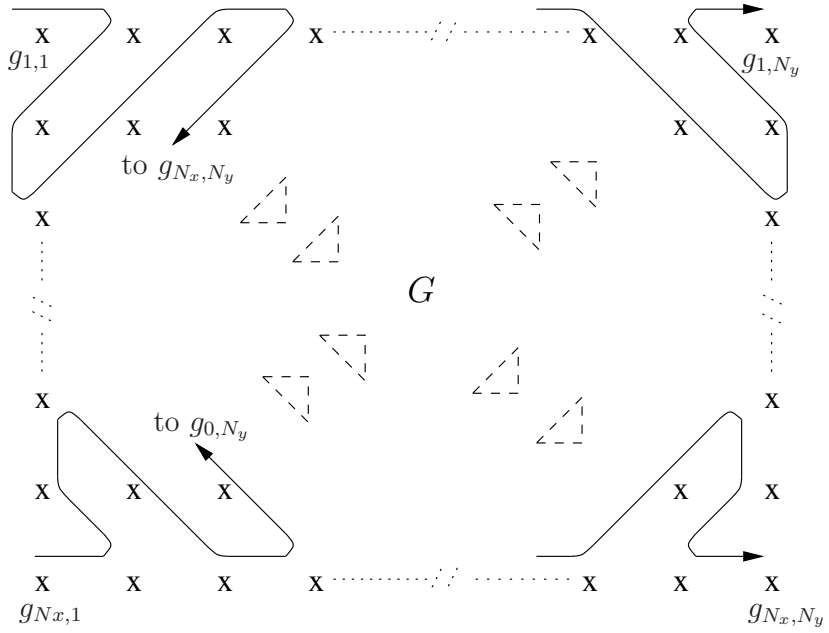

Figure 5.7: Diagonal Raster Sweeps: from grid $g_{1,1}$ to g_{N_x,N_y} and from grid $g_{N_x,1}$ to g_{1,N_y}

$$G \Rightarrow \text{horizontal sweep} \Rightarrow G^{\rightarrow}$$

$$G \Rightarrow \text{vertical sweeping} \Rightarrow G^{\downarrow}$$

$$G \Rightarrow \text{down-diagonal sweep} \Rightarrow G^{\searrow}$$

$$G \Rightarrow \text{up-diagonal sweep} \Rightarrow G^{\nearrow}$$

All deleted grids $g_{i,j}$ are assigned an empty set ($z_{i,j} = \{\emptyset\}$). Each sweep is a linear operation, tracing and comparing grid points one by one from start to finish (with occasional reversing for deletion), therefore requiring $O(N)$ computations. The intermediate result after the sweep is the intersection of

$$G^{\text{sweep}} = G^{\rightarrow} \times G^{\downarrow} \times G^{\searrow} \times G^{\nearrow} \tag{5.28}$$

where $\times$ denotes set intersection. However, there are still grid points $\in G^{\text{sweep}}$ that can be removed because they cover a subset of demand nodes. To ensure that these grid points are eliminated, the recursion process is used. G^{sweep} is subject to additional grid pairings, which work repeatedly in a recursive manner — each surrounding grid point is the start of another paring and comparison — and the recursion terminates when the surrounding grids either have already all been deleted, or are all non-subset TPs.

$$G^{\text{sweep}} \Rightarrow \text{recursive paring} \Rightarrow G^{\text{recursion}}$$

In the worst case, results from section 7.1 show that when the coverage radius R shrinks to a point ($R \approx 0$), the sweep process only eliminates grid points that cover an empty set, all other grid points at the locations of demand nodes always cover a different set (in this case the coverage set is the demand node $\{m_k\}$ itself), and therefore all the grid points at the demand node locations remain in G^{sweep}. Since all grid points surrounding the demand node are removed because they cover an empty set, thus the recursion is only one depth. Assume that the number of demand nodes M are maximum $= N/4$ (because $\Delta g = \Delta x/2 = \Delta y/2$), therefore the recursion process will require at most $O(N)$ computations.

5.4.2 Merging

Both the sweep and recursive pairing only eliminate grid points that are a coverage-subset, but do not distinguish or separate grid points that are redundant. Including redundant grid points increases the size of search space — from the optimality standpoint, all redundant grid points are indistinguishable because each covers the same set of demand node nonetheless. Therefore, for each group of redundant grid points remaining in $G^{\text{recursion}}$, only the grid point that is closest to all the demand nodes it covers (i.e., based on minimum average distance) is selected.

$$G^{\text{recursion}} \Rightarrow \text{merging} \Rightarrow G^{*}$$

G^{*} is the final matrix of TPs for the design BS placement.

5.4.3 Proof of Optimality of the Sweep and Merge algorithm

Suppose that the network has M demand nodes, where

$$m_k = \text{demand node } k^{th}, \quad k \in \{1, \ldots, M\} \tag{5.29}$$

Given a set of N TPs

$$\text{set of TPs} = \{\text{TP}_1, \text{TP}_2, \ldots, \text{TP}_N\}$$

in which

$$\text{TP}_n = g_{i_n, j_n}, \ n = \{1, \ldots, N\}$$

where g_{i_n,j_n} is a grid point at $((i_n - 1)\Delta g, (j_n - 1)\Delta g)$. Define

$$b_n = \begin{cases} 1 & : \text{ if a BS is installed at grid } \text{TP}_n \\ 0 & : \text{ otherwise} \end{cases} \tag{5.30}$$

$$m_{kn} = \begin{cases} 1 & : \text{ if demand node } m_k \text{ is assigned to } b_n \\ 0 & : \text{ otherwise} \end{cases} \tag{5.31}$$

and the distance d_{kn} between m_{kn} and b_n

$$d_{kn} = \sqrt{(X_k - (i_n - 1)\Delta g)^2 + (Y_k - (j_n - 1)\Delta g)^2} \tag{5.32}$$

for each grid g_{i_n,j_n}, let z_{i_n,j_n} be a set of demand nodes that are within the coverage radius R from grid g_{i_n,j_n}

$$z_{i_n,j_n} = \{m_k | d_{kn} \leq R\} \tag{5.33}$$

and let s be an optimal solution set of b_n, such that

$$s = \{b_n | b_n = 1\}$$

For any $b_n \in s$, there exists an assignment set z_{b_n} between demand node m_k and b_n in which

$$z_{b_n} = \{m_k | m_{kn} = 1\} \tag{5.34}$$

Given that $g_{i_{n'},j_{n'}} \in G^*$, therefore by definition

$$z_{i_n,j_n} \subset z_{i_{n'},j_{n'}} \tag{5.35}$$

thus replacing b_n by $b_{n'}$ in the optimal solution set s does not increase the size of the set and because $z_{b_n} \subset z_{i_n,j_n}$, there always exists an equivalent set of demand node assignment for $b_{n'}$, such that

$$z_{b_{n'}} \doteq z_{b_n} = \{m_k | m_{kn} = 1\} \tag{5.36}$$

and as a result, the demand node assignment of the optimal solution set s does not change either. In other words, the optimal solution remains unchanged. Therefore, $g_{i_{n'},j_{n'}} \in G^*$ resulted from the sweep and merge algorithm always contain the optimal solution.

5.4.4 Computational Reduction

Let N be the total number of grid points required, and $N^* = |G^*|$ be the total of TPs resulted from the sweep and merge algorithm. Each sweep requires $\approx N$ computations, the recursion process also requires $\approx N$ computations, while the merge process requires at most N computations. Thus, in the worst case the computational requirement (C') of the TP reduction algorithm is

$$C' \approx 4 * \text{sweeps} \; + \; \text{resursive paring} + \; \text{merge} = 4N + N + N = 6N \approx O(N) \tag{5.37}$$

If n^* is the minimum number of BSs needed, and because $N^* \leq N$ the search space size S is then reduced by a factor

$$\text{reduction factor} = \left(\frac{N}{N^*}\right)^{n^*} \tag{5.38}$$

in comparison to the exhaustive search case. There is no closed-form formula to determining N^*. Since G^* is derived from set coverage comparison of demand nodes, whose distribution is often unknown (i.e., random). In fact, N^* is a function of the total number of demand nodes M and the coverage radius R,

$$N^* \propto f(M^{-1}, R^{-1}) \tag{5.39}$$

When $R \to \infty$, all grid points are redundant (covering the same set of demand nodes), and the merge would result in one TP at the center of gravity of all demand nodes, such that

$$\min[N^*] = 1 \tag{5.40}$$

Discussed later in section 7.2.4, N^* approaches $N/2$ as M increases — given that the distribution of demand nodes is random. Thus, it can be concluded that

$$1 \leq N^* \leq N/2 \tag{5.41}$$

and S is at least reduced approximately by a factor 2^{n^*}.

6.0 DESIGN OPTIMIZATION

The design optimization presented in this paper consists of three parts: coverage optimization, capacity validation, and optimization of BS assignment by Tabu Search (figure 6.1). Given demand node locations, N initial grid points (TPs) that always guaranteeing the optimal solution must be determined first. The sweep and merge algorithm is then applied to remove grid points that are not part of the optimal solution maintain only the minimum number of N^* TPs. The coverage optimization phase determines the minimum number of BSs to satisfy the coverage requirement. One method to optimize the coverage is to search through all the possible combination of BS placement from N^* TPs. However, the exhaustive search requires extensive computation and as a consequence may consume indefinite time to complete the search [3]. Alternatively, a minimum branching algorithm is developed to solve for the coverage solution. The minimum branching algorithm, which is an extension of the DFS algorithm (section 4.4), builds a tree of solutions whose depth (the complexity of the algorithm) is bounded by the best solution (i.e., number of BSs needed to satisfy to coverage requirement) discovered earlier. Compared to the exhaustive search, the minimum branching algorithm requires fewer computations (section 6.1.2). As shown if figure 6.1, to validate whether a set of BSs in the coverage solution have sufficient capacity to support the demand or not, demand nodes are assigned to BSs in a fashion that the total network capacity is maximized. The optimal demand node assignment is solved by the branch and bound algorithm (using the CPLEX optimization software package). If the coverage solution with the optimal demand node assignment has sufficient capacity to support all the demand nodes, the resulting coverage solution is the optimal solution of the design. However if the capacity requirement is not meet, the design must reassign or add more BSs to support the excess capacity requirements.

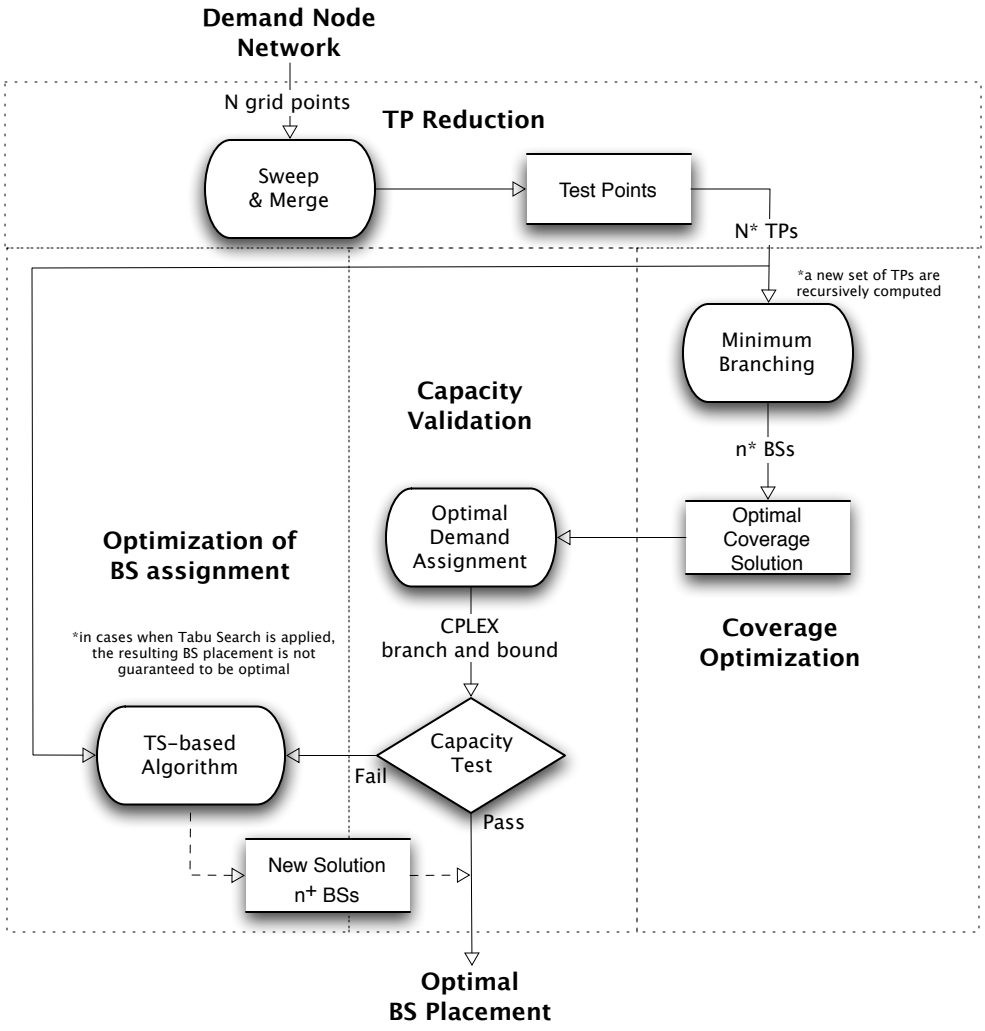

Figure 6.1: Design Optimization Flowchart

An algorithm based on Tabu Search (TS) is developed to solve the new BS assignment that can satisfy both the coverage and capacity requirements. TS is more favorable over other heuristics (chapter 4) because it offers a systematic approach to converge to the optimal solution. Using both short term and longer term memories, TS is more effective and less sensitive to changes in parameters than SA or GA, which relies upon random probability to escape from local optima. The chapter begins with the discussion of the minimum branching algorithm used to compute the coverage solution.

6.1 COVERAGE OPTIMIZATION

The minimum branching algorithm is employed to find the coverage solution. The algorithm, although more time consuming than TS or SA, always guarantees the optimal solution.

6.1.1 Minimum Branching Algorithm

Let n^* be the minimum number of BSs required and N^* be the candidate TPs computed from the sweep and merge algorithm. The exhaustive search algorithm tests all combinations of BSs placed at N^* candidate TPs, and the search stops whenever a combination of BSs satisfies the coverage requirement. If n^* is not known a priori, the whole process must begin with $n^* = 1$ and iteratively increases n^* until finding the optimal combination of BSs that satisfies the coverage requirement. Note that since n^* is the minimum number of BS, no solution will be better. Thus, the size of space S is

$$\begin{aligned} S &= \binom{N^*}{1} + \binom{N^*}{2} + \ldots + \binom{N^*}{n^*} \\ &= N^* + N^*(N^*-1)/2! + \ldots + N^*(N^*-1)\ldots(N^*-n^*-1)/n^*! \\ &\approx N^*(N^*-1)\ldots(N^*-n^*-1)/n^*! \approx N^{*n^*} \end{aligned} \tag{6.1}$$

Instead of having to place n^* BSs at N^* candidate TPs as in the exhaustive search case, the search process can be reversed by originating the search at any arbitrary demand node m_k. Suppose that the sweep and merge algorithm produces $N_1^* = N^*$ candidate TPs that always satisfies the coverage requirement, thus out of N_1^* TPs, there are always $N_{1,k}^*$ TPs within the coverage distance R from m_k. If there exists a coverage solution, the demand node m_k must be assigned with at least one BS. Therefore, one TP from $N_{1,k}^*$ candidates is certainly the TP where one of the BSs in the solution must be located at. To ensure the optimal solution, all of the $N_{1,k}^*$ candidate TPs must tested by selectively placing a BS at each individually. Suppose that, TP_{n_1} is selected from the set of $N_{1,k}^*$ TPs. Let $z_{i_{n_1},j_{n_1}}$ be the set of demand nodes covered when b_{n_1} is placed at TP_{n_1}

$$z_{i_{n_1},j_{n_1}} = \{m_k | m_{kn_1} = 1\} \tag{6.2}$$

Then,

$$\text{uncovered set of demand nodes} = \{m_k | m_k \notin z_{i_{n_1}, j_{n_1}}\}$$

A new set of N_2^* TPs are recomputed from $\{m_k | m_k \notin z_{i_{n_1}, j_{n_1}}\}$. Again, for any $m_k \notin z_{i_{n_1}, j_{n_1}}$ of the uncovered set, there are always $N_{2,k}^*$ TPs within the coverage distance R. These $N_{2,k}^*$ TPs are also the candidate TPs where the second BS in the solution set can be placed at. If TP_{n_2} is selected from the set of $N_{2,k}^*$ TPs. Let $z_{i_{n_2}, j_{n_2}}$ be the set of demand nodes covered when b_{n_2} is placed at TP_{n_2}

$$z_{i_{n_2}, j_{n_2}} = \{m_k | m_{kn_2} = 1\} \tag{6.3}$$

Now, because two BSs b_{n_1} and b_{n_2} are placed at TP_{n_1} and TP_{n_2} respectively

$$\text{uncovered set of demand nodes} = \{m_k | m_k \notin z_{i_{n_2}, j_{n_2}} \cup z_{i_{n_1}, j_{n_1}}\}$$

A new set of N_3^* TPs are recomputed from $\{m_k | m_k \notin z_{i_{n_2}, j_{n_2}} \cup z_{i_{n_1}, j_{n_1}}\}$ by the sweep and merging algorithm again, and the process of selecting arbitrary m_k and its associated TPs recurses until there is no demand nodes remain in the uncovered set.

$$\text{uncovered set of demand nodes} = \{\emptyset\}$$

For each arbitrary demand m_k selected at each level, there are always a number of candidate TPs ≥ 0 associated with. These TPs are among the choices for placing BSs. The chosen TP can be viewed as a node in a tree, branching a new set of TPs or nodes for the next level or recursion. The tree contains a particular set of feasible soltuions. However, the shape and the size of the tree created depends on the selection of arbitrary demand node m_k at each level. In other words, there are other trees of different shapes and sizes that can be created by selecting different demand node m_k so that they contain different feasible solution sets. Thus, the tree created is actually one subtree of the whole possible solution tree (contain all feasible solution sets). Figure 6.2 shows the example of a particular subtree of solutions

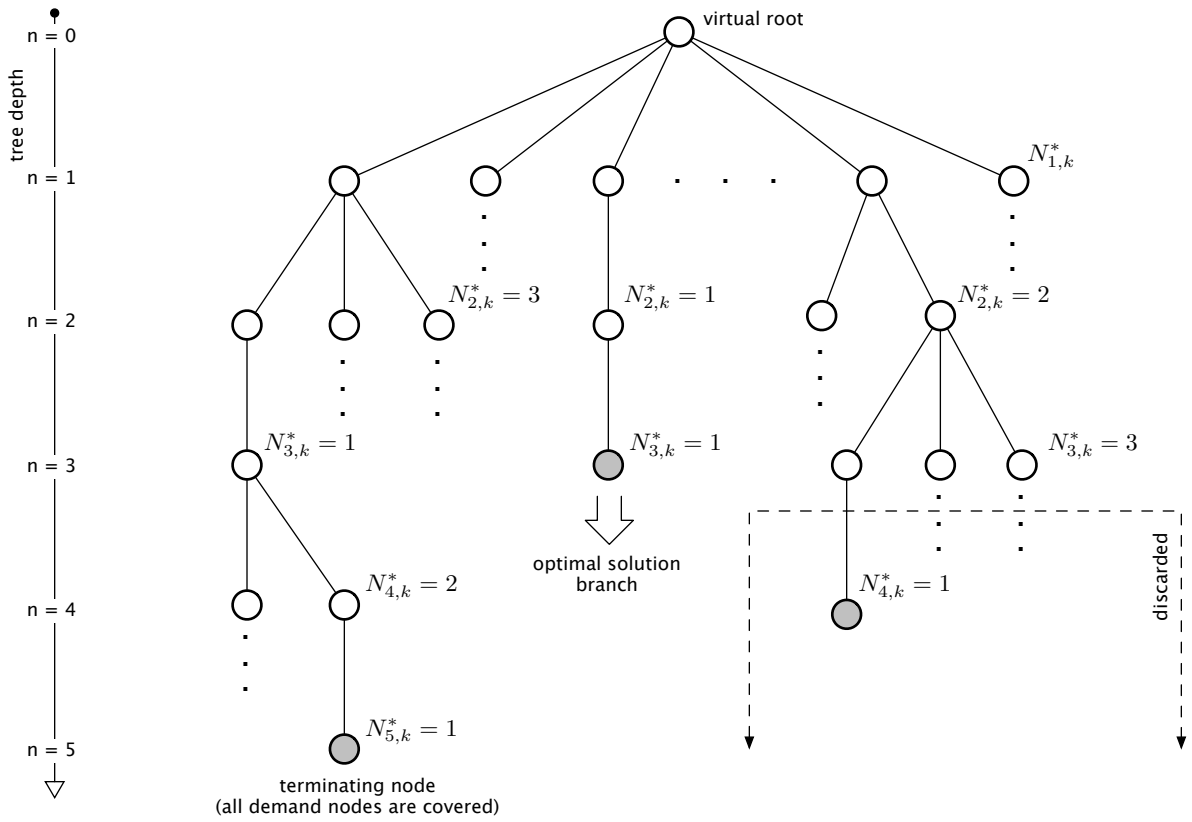

Figure 6.2: Subtree of Solutions. TP chosen at each recursion represents a node in the subtree.

The terminating node is always a TP (BS) that covers the remaining demand nodes. Each set of TPs along the branches to the terminating TP is a feasible solution. The optimal solution is a set of fewest number of TPs from the root down to the terminating node as shown in figure 6.2. Similar to the DFS algorithm (section 4.4), any node whose lower bound (i.e., order of tree depth) is greater than the upper bound seen previously is discarded. However, since the entire solution tree may contains so many subtrees (but different shapes and sizes) similar the one shown in figure 6.2. And in fact, if a particular subtree contains many number of nodes or branches, searching for the optimal solution may not be efficient. One approach to reduces the number of possible nodes in the subtree is to reduce the number of nodes in each level of the subtree by selecting the demand node m_k associated with the fewest number of TPs (i.e., $N^*_{1,k}, N^*_{2,k}, \ldots$). The outermost demand node, which is defined to be the

demand node associated with the fewest number of TPs, is selected in each recursion (after the previous BS is placed and a new set of TPs are computed). The outmost demand node at the tree level l^{th} can be defined as

$$\text{outermost demand node} = \min_{m_k}[N^*_{l,k}] \quad \forall m_k \notin \bigcup_{l-1} z_{i_{n_{l-1}}, j_{n_{l-1}}} \tag{6.4}$$

where the last term in (6.4) represents the union of all covered demand nodes for each BS placed previously before the recursion reaches the l^{th} level. If the outermost demand node is selected instead of randomly choosing m_k, each node of the subtree will have the fewest number of branches — minimum branching algorithm. However, the total number of nodes combined in the minimum branching tree may not be the fewest compared to other subtrees of the entire solution tree (the minimum branching algorithm is similar to the approach in which the greedy algorithm is used to solve the K-minimum spanning tree). Nevertheless, the minimum branching subtree will always contain the path (or branch) to the optimal solution and the number searches (total number of nodes) are reduced compared to most of the subtrees and the exhaustive search. The following pseudo code describes the `mbranch()` routine used to generate the minimum branching subtree of solutions.

```
• compute TPs by sweeping and merge
depth = 0;
mbranch() {
    depth = depth + 1;
    • find the outermost m_k
    for each of its candidated TPs,
        • update the remaining m_k
        if there is no m_k remaining,
            n* = depth;
            depth = depth - 1;
            return;
        endif;
        if depth ≥ n* − 1
            • continue to the next candidate TP
        endif;
        • compute new TPs by sweeping and merge on the remaining m_k
        • call mbranch() again
    done;
    depth = depth - 1;
}
```

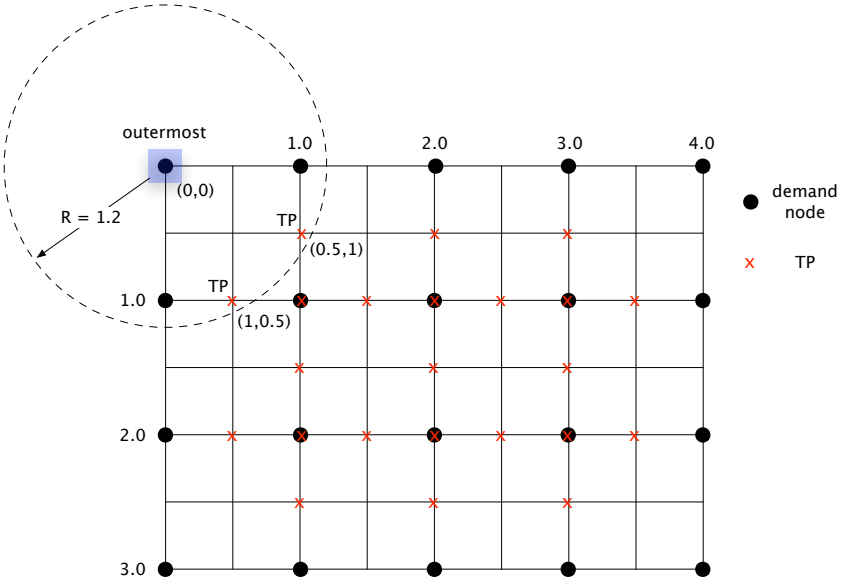

Figure 6.3: 1st recursion. The outermost m_k is at (0,0) with two candidate TPs at (1,0.5) and (0.5,1) respectively.

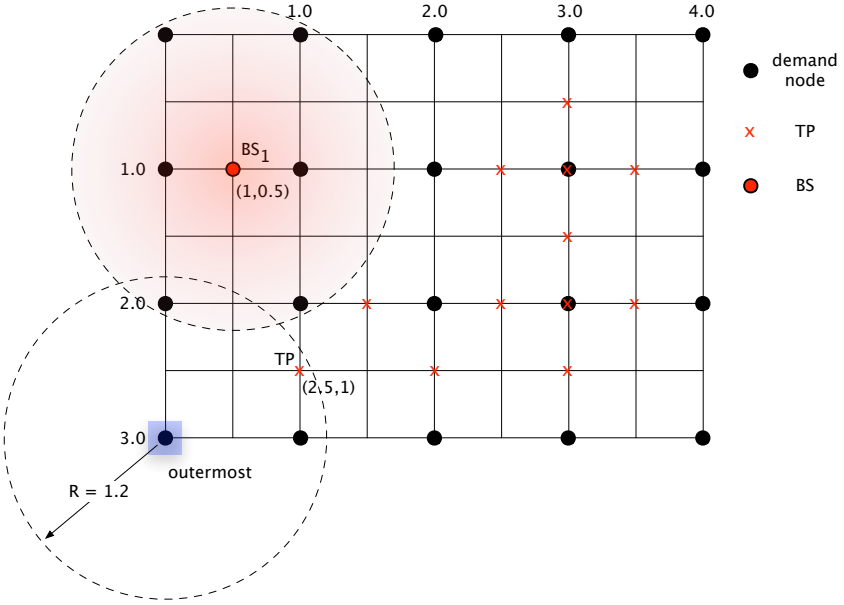

Figure 6.4: 2nd recursion. TP at (1,0.5) from the previous recursion is chosen as BS_1. The outermost m_k is at (3,0) with a single candidate TPs at (2.5,1).

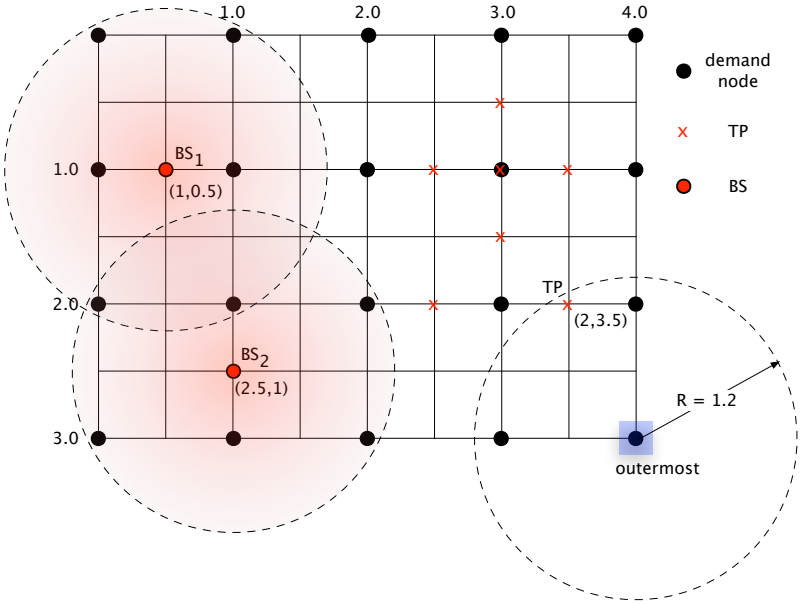

Figure 6.5: 3rd recursion. TP at (2.5,1) from the previous recursion is chosen as BS_2. The outermost m_k at (3,4) with a single candidate TPs at (2,3.5).

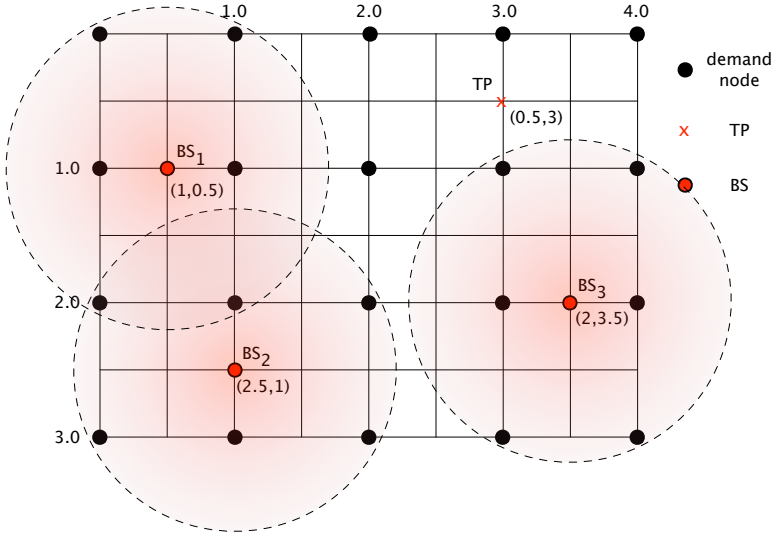

Figure 6.6: 4th recursion. TP at (2,3.5) from the previous recursion is chosen as BS_3. The sweep and merge only produces one TP at (0.5,3).

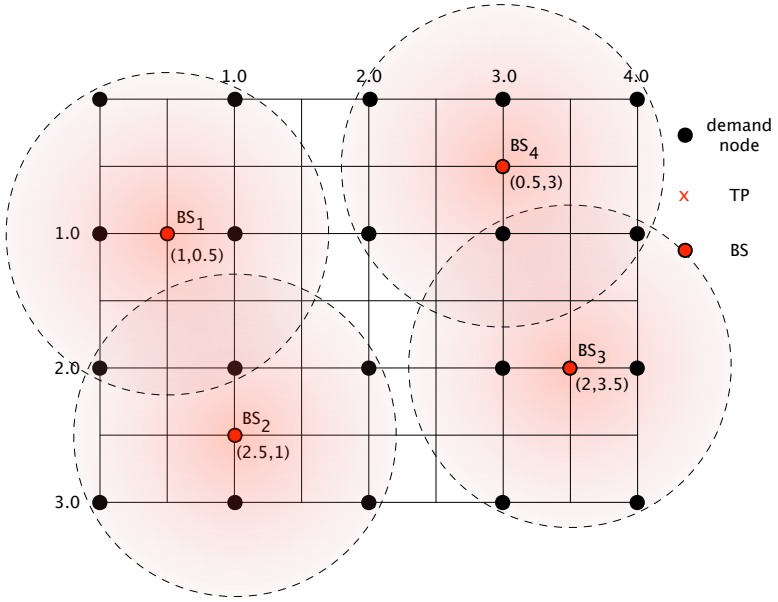

Figure 6.7: Coverage Solution. Only TP at (0.5,3) is chosen as BS_4. All demand nodes are covered, and the coverage requirement is satisfied.

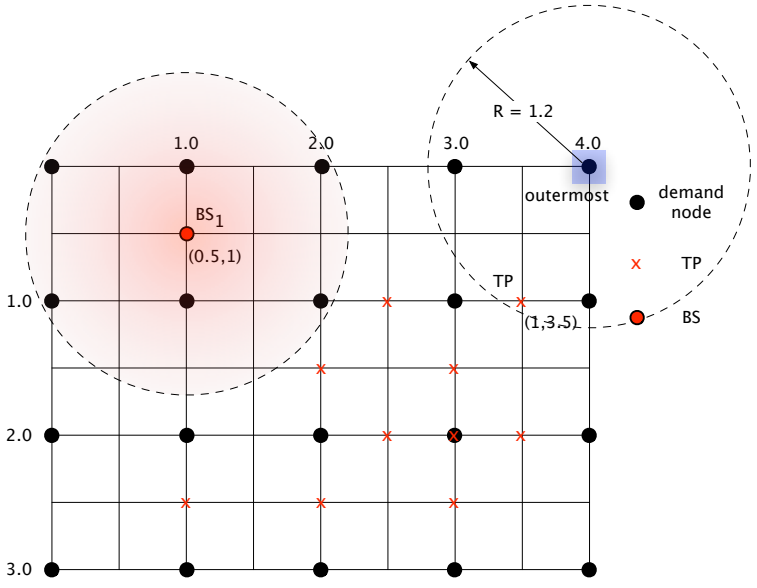

Figure 6.8: 5^{th} recursion. The other TP at (0.5, 1) from the 1^{st} recursion is now chosen as BS_1. The outermost m_k is at (0,4) with a single candidate TPs at (1,3.5).

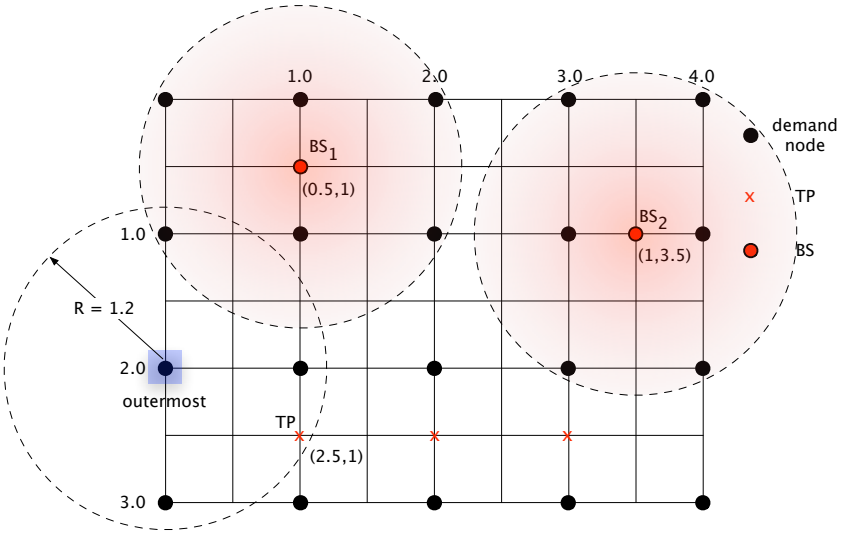

Figure 6.9: 6^{th} recursion. TP at (1,3.5) from the previous recursion is chosen as BS_3. The outermost m_k at (2,0) with a single candidate TPs at (2.5,1).

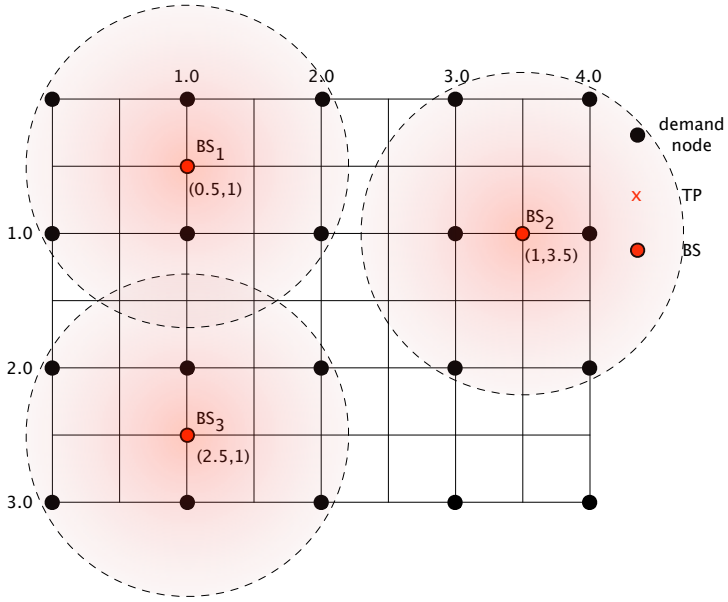

Figure 6.10: 7^{th} recursion. TP at (2.5,1) from the previous recursion is chosen as BS_3. Since any solution resulted after will never be better than the best solution found earlier (4 BSs are needed to satisfied to coverage requirement). The algorithm stops.

A set of figures from 6.3 - 6.10 illustrate the minimum branching algorithm in progress. For the first recursion shown in figure 6.3, there are a total of $N^* = N_1^* = 23$ TPs, the outermost demand node at grid (0,0) is associated with two candidate TPs at (1,0.5) and (0.5,1) — $N_{1,k}^* = 2$ (fewest among other demand nodes). The lower TP at (1,0.5) is selected as . Since some of the demand nodes are already covered by BS_1, the network is different now and only the uncovered demand nodes are considered. The sweep and merge algorithm is reapplied to compute a new set of TPs from demand nodes which are not covered by BS_1. For the second recursion, the outermost demand node at grid (3,0) is associated with only candidate TP at (2.5,1) — $N_{2,k}^* = 1$. Thus, the TP at (2.5,1) is chosen as BS_2. A new set of TPs are recomputed from the set of uncovered demand nodes (not covered by BS_1 and BS_2). After four recursions, all demand nodes are covered, four BSs are needed, and the first candidate solution is computed as shown in figure 6.7. For the fifth recursion, the algorithm recurses back to test and replace BS_1 at the other TP node at (0.5, 1) of the first recursion. After seven recursions, three BSs are placed, but there are still demand nodes uncovered. Since, the first candidate solution found in the four previous recursions requires four BSs to satisfy the coverage requirement, any solutions resulted after the seventh recursion will never be better than the first candidate solution (will require equal to or more than BSs to satisfy the coverage requirement), thus the algorithm truncates and stops. The optimal solution requires $n^* = 4$ BSs to cover all demand nodes (which is equal to the minimum tree depth). Compared to the exhaustive search which requires

$$S = \binom{N^*}{n^*} = \binom{23}{4} = 8,885$$

searches to compute the optimal solution, the minimum branching algorithm requires only 7 recursions (searches), which is much more computationally efficient. Figure 6.11 shows the resulting minimum branching subtree of solutions. Later on, the term 'subtree' will be replaced by just 'tree' for abbreviation and simplicity throughout the paper. The next section describes the computational requirement of the minimum branching algorithm.

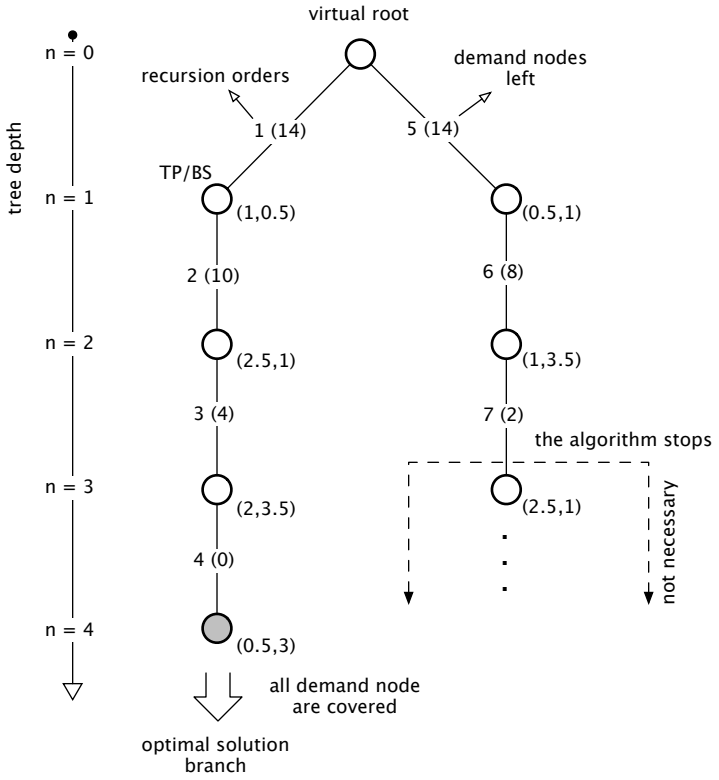

Figure 6.11: Minimum Branching Subtree.

6.1.2 Computational Requirement

In general, if S is the size of search space, M is the total number of demand nodes, n^* is the total number of BSs needed, then the computational requirement C' is

$$C' = n^* MS \tag{6.5}$$

For the minimum branching algorithm, the minimum tree depth is n^* (which is equal to minimum # of BSs needed), then

$$\begin{aligned} S &\approx N^*_{1,k} + \sum N^*_{2,k} + \ldots + \sum N^*_{n^*-1,k} + \delta \\ &\approx N^*_{1,k} + N^*_{1,k}N^*_{2,k} + \ldots + N^*_{1,k}N^*_{2,k}\ldots N^*_{n^*-1,k} + \delta \\ &\approx N^*_{1,k}(1 + N^*_{2,k}(1 + \ldots + (1 + N^*_{n^*-1,k})\ldots)) + \delta \end{aligned} \tag{6.6}$$

Note that, $N^*_{2,k}$ to $N^*_{n^*,k}$ are the average number of TPs chosen at each recursion depth. The residue δ is a function of how the minimum branching tree is constructed. The DFS algorithm (section 4.4) — as shown throughout all the examples — on average results in larger δ than the BFS implementation. The BFS algorithm (section 4.4), on average, requires $\delta \approx \sum N^*_{n^*,k}/2$, however at the expense of extra memories to store all intermediate solutions found in each tree depth. Nevertheless, for the sake of approximating S, the BFS algorithm is assumed. Thus,

$$\begin{aligned} S &\approx N^*_{1,k}(1 + N^*_{2,k}(1 + \ldots + (1 + N^*_{n^*-1,k})\ldots)) + \sum N^*_{n^*,k}/2 \\ &\approx N^*_{1,k}(1 + N^*_{2,k}(1 + \ldots + (1 + N^*_{n^*,k})\ldots)) \end{aligned} \tag{6.7}$$

Since

$$N = N^*_1 > N^*_2 > \ldots > N^*_{n^*} \tag{6.8}$$

because the network has fewer demand nodes when more BSs are placed. Thus, it is likely that

$$N^*_{1,k} > N^*_{2,k} > \ldots > N^*_{n^*,k} \tag{6.9}$$

For the worst case, if we assume that

$$N^*_{1,k} = N^*_{2,k} = \ldots = N^*_{n^*,k} \tag{6.10}$$

Thus, S in (6.7) can be approximated by

$$S \approx N^*_{1,k}(1 + N^*_{2,k}(1 + \ldots + (1 + N^*_{n^*,k})\ldots)) < {N^*_{1,k}}^{n^*} \tag{6.11}$$

With the minimum branching, a new set of TPs must be recomputed for every recursion. As discussed in section 5.4.4, the computational complexity of the TP reduction algorithm is $O(N)$, thus the computational requirement of the minimum branching algorithm can be approximated as

$$C' = (n^*M + N)S < (n^*M + N){N^*_{1,k}}^{n^*} \tag{6.12}$$

In the worst case, if the demand node density approaches the maximum ($M \to N/4$, because $\Delta g^2 = \Delta x^2/4$), thus

$$C' = (n^*N/4 + N){N^*_{1,k}}^{n^*} \approx n^*N{N^*_{1,k}}^{n^*} \tag{6.13}$$

Compared to the exhaustive search where,

$$S \approx N^{*n^*},\ C' \approx n^* N N^{*n^*} \tag{6.14}$$

Since, $N^*_{1,k} \ll N^*$, the minimum branching algorithm is much more computationally efficient than the exhaustive search. In conclusion, employing the TP reduction algorithm to reduce TPs from N to N^* (while the optimal solution is still maintained) and the minimum branching to reduce the number searches, the computation requirement for computing the optimal coverage solution can be reduced approximately by a factor

$$\approx (N/N^*_{1,k})^{n^*} \tag{6.15}$$

6.2 CAPACITY VALIDATION - OPTIMAL DEMAND NODE ASSIGNMENT

The optimal coverage solution resulted from the minimum branching algorithm may or may not satisfy the capacity requirement. The total capacity in which the network can support is a function of the assignment between the demand nodes and BSs. To validate whether the coverage solution has sufficient capacity to support all the demand nodes or not, the assignment must be done is such a way that the total network capacity of the coverage solution is maximized. Given a set of N^* TPs computed from the sweep and merge algorithm

$$\text{a set of TPs} = \{\text{TP}_1, \text{TP}_2, \ldots, \text{TP}_{N^*}\}$$

in which

$$\text{TP}_n = g_{i_n,j_n},\ n = \{1, \ldots, N^*\}, \quad \forall z_{i_n,j_n} \neq \{\emptyset\}$$

where g_{i_n,j_n} is a grid point at $((i_n - 1)\Delta g, (j_n - 1)\Delta g)$. Redefine

$$b_n = \begin{cases} 1 & : \text{ if a BS is installed at TP}_n \\ 0 & : \text{ otherwise} \end{cases} \tag{6.16}$$

$$m_{kn} = \begin{cases} 1 & : \text{ if demand node } m_k \text{ is assigned to } b_n \\ 0 & : \text{ otherwise} \end{cases} \tag{6.17}$$

and the Euclidean distance d_{kn} between demand node m_k at (X_k, Y_k) and b_n

$$d_{kn} = \sqrt{(X_k - (i_n - 1)\Delta g)^2 + (Y_k - (j_n - 1)\Delta g)^2} \tag{6.18}$$

Let s be the coverage solution resulted from the minimum branching algorithm,

$$s = \{b_n | b_n = 1\}, |s| = n^* \tag{6.19}$$

The optimal demand node assignment problem is then formulated as

$$\begin{aligned} \max \quad & \sum_{n=1}^{N^*} \sum_{k=1}^{M} m_{kn} \\ \text{subject to} \quad & b_n \in s^* \\ & m_{kn} \leq \max\,[0, \lceil R - d_{kn} \rceil] b_n \\ & \sum_{n=1}^{N^*} m_{kn} = 1,\ \forall k \in \{1, \ldots, M\} \\ & \sum_{k=1}^{M} m_{kn} \leq C,\ \forall n \in \{1, \ldots, N^*\} \end{aligned} \tag{6.20}$$

(all the variables TP_n, b_n, m_{kn}, and d_{kn} defined above will be used throughput the chapter). The optimal demand node assignment problem in (6.20) is solved by the CPLEX optimization software package (implementing the branch and bound algorithm). If the coverage solution with the optimal demand node assignment is unable to meet the capacity requirement, it means that one or more BSs may cover a larger demand than their maximum capacity C. One approach to fix the capacity problem is to add sufficient number of BSs to cover the rest of demand nodes (uncovered demand nodes from the coverage solution). However, this approach may result in too many BSs than necessary. An algorithm based on Tabu Search (TS) used to optimize BS assignment is described next.

6.3 OPTIMIZATION OF BS ASSIGNMENT BY TABU SEARCH

Optimization of BS assignment is required in cases when the coverage solution (resulted from the minimum branching algorithm with n^* BSs) has insufficient capacity to support the demand (despite of the optimal assignment). The goal of this phase is then to compute for a new solution that requires minimum BSs and yet able to provide the coverage and support all the user demand. Started by $n^+ = n^*$ BSs, the design algorithm keeps adding a BS one by one until the network has sufficient BSs to support the capacity requirement. An algorithm based on Tabu Search (TS) is developed and incorporated to maximize the total network capacity, given that $n^+ \geq n^*$ BSs are to be placed at N^* TPs. TS is the heuristic of

choice for several BS placement designs such as the Amaldi design method [1], the Prommak design method [3], the Akl design method [26], and the Lee and Gang design method [38] (as already discussed in chapter 2). Although the principle of TS is the same (section 4.2), actual implementation may differ from one problem to another due to several factors such as the objective function, solution neighborhood structure, and how the longer term and short term memories are incorporated in the diversification and intensification phase. Let revisit a simple TS procedure

Simple Tabu Search

$s =$ `initial solution;` $i = 0$;
while $i < i_{\max}$ do
    `if all moves` $s \rightarrow s', \forall s' \in \mathcal{N}(s)$ `are tabu`
      $s = s' \in \mathcal{N}(s)$ `with a move that is least tabu;`
    `else`
      $s =$ `optimum`$[s' : \forall s' \in \mathcal{N}(s)$ `with non-tabu moves];`
    `end;`
    `a move made becomes tabu for` *tabu tenure* `iterations;`
    $i = 1 + 1$;
`done;`

TS starts by forming a sufficient number of neighborhood solutions $\mathcal{N}(s)$. Each solution is evaluated, and the best non-tabu solution is selected. If all neighborhood solutions are tabu, TS invokes an aspiration criteria to accept the least-tabu solution and proceeds forward to the next iteration. A new set of neighborhood solutions are created, and the process of evaluating and selecting the new current solution repeat for a predetermined number of iterations. In this paper, the objective function is

$$\begin{aligned} \max \quad & \sum_{n=1}^{N^*} \sum_{k=1}^{M} m_{kn} \\ \text{subject to} \quad & \sum_{n}^{N^*} b_n = n^+ \\ & m_{kn} \leq \max\,[0, \lceil R - d_{kn} \rceil] b_n \\ & \sum_{n=1}^{N^*} m_{kn} = 1, \ \forall k \in \{1, \ldots, M\} \\ & \sum_{k=1}^{M} m_{kn} \leq C, \ \forall n \in \{1, \ldots, N^*\} \end{aligned} \tag{6.21}$$

An initial solution for TS is computed from a simple greedy algorithm

Simple Greedy Algorithm

```
while # of BSs ≤ n*
      • install a BS at TP that maximizes the total network capacity
      • increase the number of BSs by one
done;
```

Designed specifically for the problem in (6.21), the neighborhood structure $\mathcal{N}(s)$, the short term memory, the longer term memory with both intensification and diversification strategies are described next.

6.3.1 Neighborhood Structure

The neighborhood structure $\mathcal{N}(s)$ consists of a set of candidate solutions surrounding the current solution s, where

$$s = \{b_n | b_n = 1\}, |s| = n^+ \tag{6.22}$$

The candidate solution s' is created by dropping one $b_n \in s$ and adding a new $b_{n'}$ selected from the remaining TPs.

$$s' = s - \{b_n = 0\} + \{b_{n'} = 1\} \tag{6.23}$$

To limit the number of possible swapping/b_n, only the h closest TPs are considered, such that

$$\mathcal{N}(b_n) = \{s' | \forall b_{n'} \text{ activated from the closest } h \text{ TPs}\},\ b_{n'} \notin s \tag{6.24}$$

The neighborhood structure of the current solution s is thus a union of

$$\mathcal{N}(s) = \sum \mathcal{N}(b_n),\ \forall b_n \in s \tag{6.25}$$

It is possible that $\mathcal{N}(b_n)$ may intersect, but since each $b_n \in s$ is distinct, there are nevertheless a total of $n^+ \times h$ distinct neighbor solutions. Figure 6.12 show one example of the neighborhood structure $\mathcal{N}(s)$.

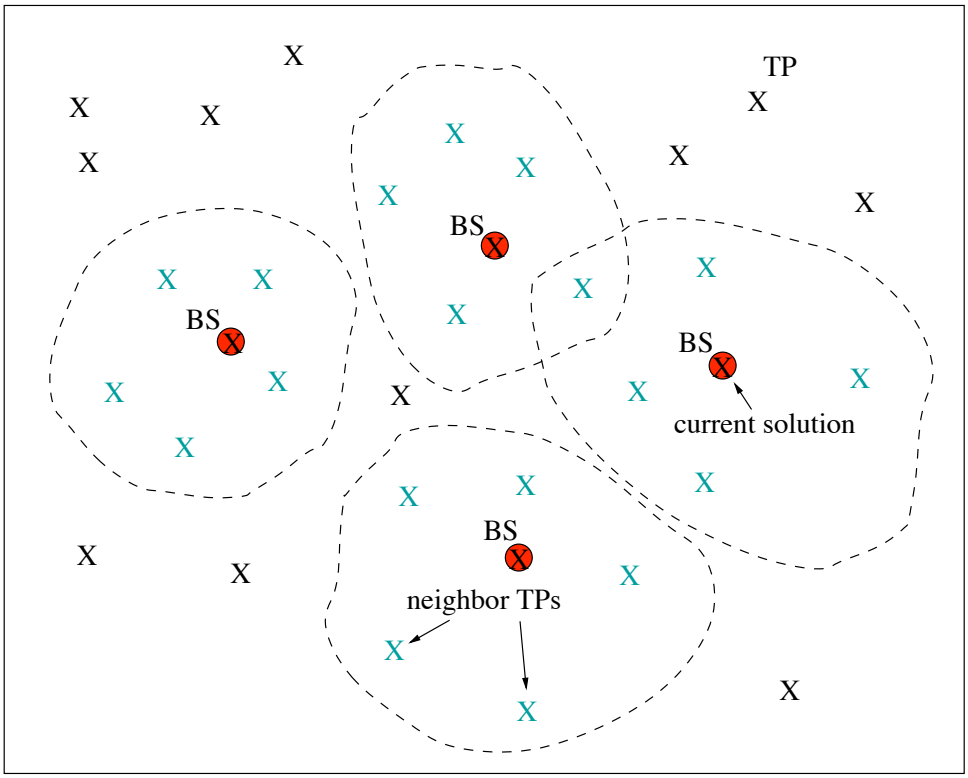

Figure 6.12: A neighborhood structure $\mathcal{N}(s)$ consisting of $n^+ = 4$ subsets of $h = 5$ closest neighbor TPs.

The neighborhood structure $\mathcal{N}(s)$ described above is one alternative to implement a candidate list strategy in TS [48]. One approach to construct a candidate list (neighbor solutions) is to include all solutions created from all the possible swaps. However, if the number of TPs in the pool is very large, the number of neighbor solutions to be evaluated will grow proportionally. Some of these neighbor solutions may not be a good candidate, and examining all of them could undermine the algorithm effectiveness. The proposed $\mathcal{N}(s)$ isolates a set of good candidate solutions and includes only the h closest solutions for each $b_n \in s$, which is similar to $\mathcal{N}(s)$ used in the Amaldi's TS implementation [1]. The neighborhood size h is problem-sensitive and must be carefully determined — not too large to ensure that the neighbor solutions in $\mathcal{N}(s)$ still retain most of the features belonging to the current solution s, but not too small to prevent the algorithm from exploring other regions of the search space. Forming $\mathcal{N}(s)$ is convenient compared to other candidate list strategies suggested

in [48], and applying the proposed $\mathcal{N}(s)$ for the BS placement problem proves to produce sufficiently high quality results (discussed later). As the algorithm proceeds, the tabu status may change, and some neighbor solutions in $\mathcal{N}(s)$ may be tabu (moves to these solutions are prohibited). Tabu classification is a rule for determining whether a move to a particular candidate solution is tabu or not. It is part of the short term memory implementation. Both the short term memory and tabu classification are discussed next.

6.3.2 Short Term Memory and Tabu Classification

TS employs short term memory to drive the search away from local optima, giving the algorithm opportunity to uncover regions that may contain the optimal solution. Solution recycling indicates the possibility that the search is confined in suboptimal regions. The cause may either inherit from the characteristic of the search space itself or from adopting unsuitable $\mathcal{N}(s)$. A simple remedy to prevent solution recycling is to memorize solutions previously visited, and for some iterations, exclude them from consideration until their tabu status are revoked [48]. Beside keeping a complete solution attribute (i.e., a whole set of TPs), a move attribute is used instead — which is more flexible and less complicated. Using $\mathcal{N}(s)$ defined in section 6.3.1, a move form one solution involves

$$\begin{aligned} &\textsf{drop (deactivate) } b_n = 0 \\ &\textsf{add (activate) } b_{n'} = 1, \textsf{selected from } \mathcal{N}(s) \end{aligned}$$

to form a new solution s'. For each added or dropped b_n, there associates a tabu status called "$TabuEnd(b_n)$", which signals the last iteration on which tabu activation is imposed. Tabu-active operations (either added or dropped) are suspended until the algorithm proceeds beyond iteration "$TabuEnd(b_n)$". Generally, the tabu status is simply expressed as

$$TabuEnd(b_n) = Iter + TabuTenure \tag{6.26}$$

where $Iter$ is the current iteration number that either an added or a dropped move is executed upon, and $TabuTenure$ is the number of tabu iterations applied to b_n. Determining the

effective $TabuTenure$ is problem-dependent: too small $TabuTenure$ may result in higher occurrences of solution recycling, whereas too large $TabuTenure$ may inadvertently preclude moves leading to good solutions, resulting in solution quality deterioration [48]. The expression in (6.26) assumes that each added or dropped move is equally likely in chances. However, since $n^+ \ll N^*$ (BSs $\ll$ TPs), therefore any $b_{n'}$ — dropped out earlier from the previous solution — is more likely to reenter the solution again compared to the possibility that $b_n \in s$ is dropped. As a consequence, $TabuAddTenure$ (# of tabu iterations for added move) should be higher than $TabuDropTenure$ (# of tabu iterations for dropped move). For dropped b_n,

$$TabuEnd(b_n) = Iter + TabuAddTenure \tag{6.27}$$

on the other hand, for added $b_{n'}$,

$$TabuEnd(b_{n'}) = Iter + TabuDropTenure \tag{6.28}$$

where $TabuAddTenure > TabuDropTenure$. In general, any move to the candidate solution s' by dropping $b_n \in s$ and adding $b_{n'} \in \mathcal{N}(s)$ is classified tabu if

$$\max[TabuEnd(b_n), TabuEnd(b_{n'})] \leq Iter \tag{6.29}$$

Following a series of added and dropped moves, it is possible that all TPs are now tabu, and all the corresponding next moves can not be made. For such case, the best move associated with the least tabu TPs is selected — aspiration by default. Evaluation is based on the total capacity $c_{s'}$ in which the candidate solution s' can support, where

$$\begin{aligned} & c_{s'} = \sum_{n=1}^{N^*} \sum_{k=1}^{M} m_{kn} \\ \text{subject to} \quad & b_n \in s' \\ & m_{kn} \leq \max\,[0, \lceil R - d_{kn} \rceil] b_n \\ & \sum_{n=1}^{N^*} m_{kn} = 1, \ \forall k \in \{1, \ldots, M\} \\ & \sum_{k=1}^{M} m_{kn} \leq C, \ \forall n \in \{1, \ldots, N^*\} \end{aligned} \tag{6.30}$$

(for each evaluation, $c_{s'}$ can be maximized by employing the optimal demand node assignment). Without tabu activation, the candidate solution s^* that yields the highest $c_{s'}$

$$c_{s^*} = \max_{s'}[c_{s'}], \quad \forall s' \in N(s) \tag{6.31}$$

is chosen to be a new current solution $s = s^*$ for the next iteration. The algorithm repeats to form a new set of neighbor solutions, selects the best move and so on for some number of predefined iterations, which must be large enough to allow the algorithm to roam through and cover as many regions of the search space. Unfortunately, there are unknown factors that may prevent the algorithm to penetrate more deeper into some uncharted regions that may contain the optimal solution. To enhance the search capability, TS also exploits longer term memory with intensification and diversification schemes. The best solution from the short term memory is used as a new starting solution for the longer term memory implementation.

6.3.3 Diversification and Intensification

Longer term memory is incorporated in TS as a mechanism to extend the scope of the search beyond capability of the short term memory. As pointed out in [48], diversification and intensification schemes are the two main components of the longer term memory implementation. Both schemes exploit information such as residence and transition frequencies collected from solutions encountered during iterations of the short term memory. Diversification is a process aiming to elude the search away from the regions frequently visited. Among a number of choices to implement diversification, restarting and modified choice rule are preferred [48]. Restarting is the most simple form of diversification strategies. Initially, moves are evaluated fairly (except their tabu classification). The new starting solution is simply the best solution found so far. Tabu status is refreshed, and only the residence frequency $f(b_n)$ is carried forward

$$f(b_n) = \frac{\#\ \text{of times}\ b_n \in s}{\text{total}\ \#\ \text{of iterations}} \tag{6.32}$$

b_n with high residence frequency is likely to be an anchor that holds back the search from escaping to other regions. High occurrence b_n must be avoided at all cost in order to diversify the search. One possibility is to modify the rule for evaluating candidate solutions. The new

rule weights in both $f(b_n)$ and the total capacity to assess the move. Any candidate solutions containing b_n with high residence frequency are penalized more heavily than the candidate solutions with lower residence frequency. The penalty $p_{s'}$ associated with the candidate solution s' is thus computed as

$$p_{s'} = w \sum f(b_{n'}), \quad b_{n'} \in s' \tag{6.33}$$

where w is a weight. With the penalty $p_{s'}$, the rule for selecting the best candidate solution in (6.31) is then modified to

$$c_{s^*} = \max_{s'}[c_{s'} - p_{s'}], \quad \forall s' \in \mathcal{N}(s) \tag{6.34}$$

such that s^* is the best solution in $\mathcal{N}(s)$. Diversification is the most effective when $p_{s'}$ and $c_{s'}$ are in the same order. Too large w may overweight the evaluation and may divert the problem objective, while too small w may subdue the diversification effect. As a compromise, one approach used in this paper sets the weight w as function of the total number of demand nodes M and the number of BSs n^+, such that

$$w \approx \frac{n^+ M}{2} \tag{6.35}$$

Diversification is not the only strategy to improve solution quality. Intensification is also needed when the focus is on specific regions in which high quality solutions may reside. Some particular regions are searched more thoroughly again in order to squeeze out a better solution. A variety of schemes such as intensification by decomposition or combining features from elite solutions are suggested by [48]. However, these schemes are complex and not suitable for the BS placement problem. One strategy is to increase the number of iterations, for which diversification is repeated until the solution quality stops improving. Other strategies to incorporate intensification and diversification include strategic oscillation, such that diversification is induced by allowing the search to cross a target level back and forth (i.e., crossing between the number of BSs or the maximum BS capacity), while intensification is initiated each time the oscillation arrives at the target level (by increasing the number of iterations at this level). In this paper, only diversification by modifying choice rule with intensification by repeated restarts is implemented as summarized in figure 6.13.

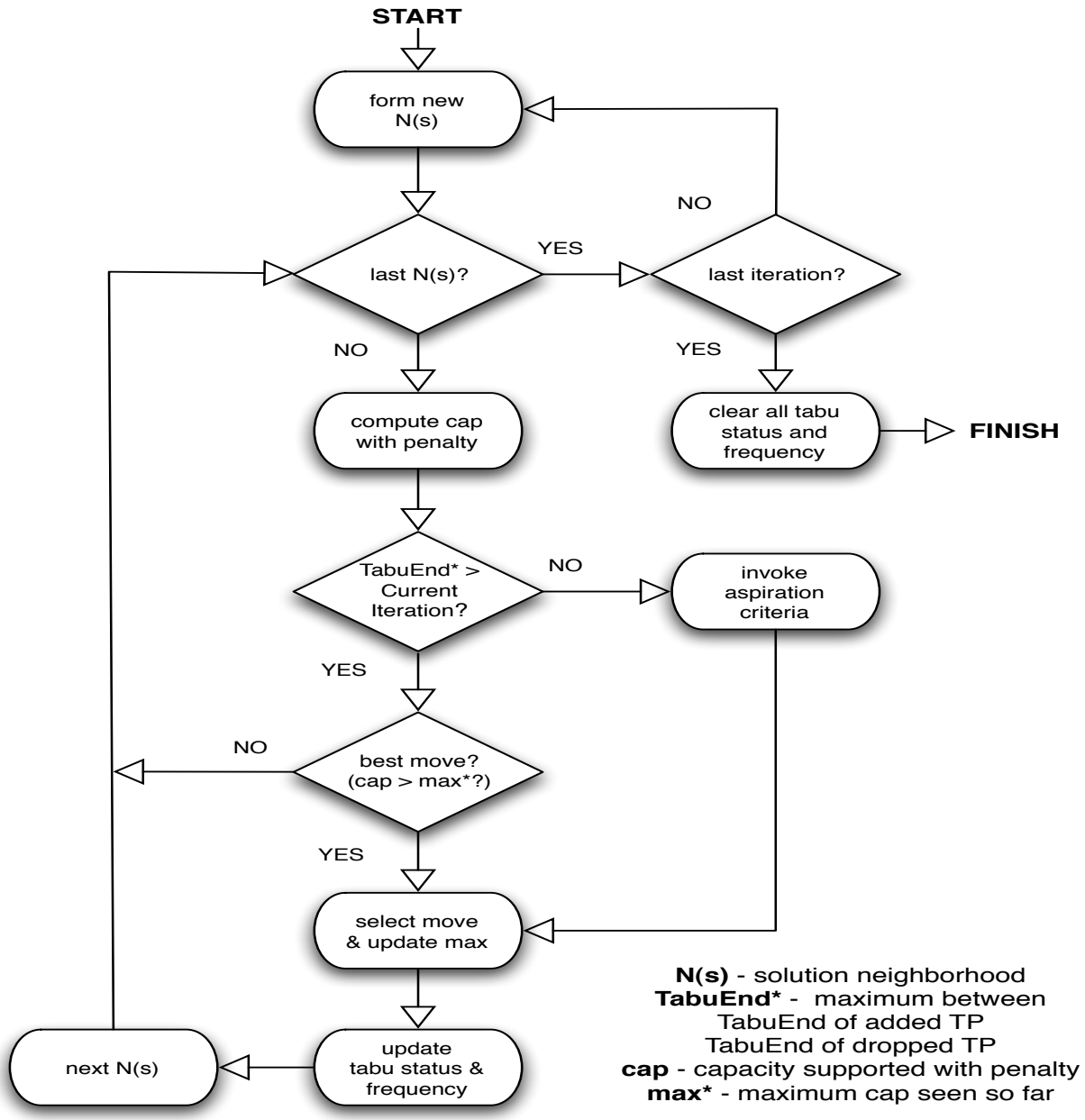

Figure 6.13: TS with Modified Choice Rule

6.3.4 Parameter Selection

Implementation of TS is problem-specific — the neighborhood structure, tabu classification, aspiration criteria, etc., must be adjusted to fit the context of a particular problem. In this paper, the problem is to search for the optimal placement of n^+ BSs that maximizes the total capacity as described in (6.21). To solve (6.21), TS with modified choice rule as shown in figure 6.13 is employed. Table 6.1 lists all relevant parameters.

Table 6.1: TS Parameters for Capacity Optimization Problem

Parameter	Description	Function
$\lvert\mathcal{N}(s)\rvert$	# of neighbor solutions	$\mathcal{N}(s)$ construction
$TabuAddTenure$	tabu tenure for added TP	tabu classification
$TabuDropTenure$	tabu tenure for dropped TP	tabu classification
w	penalty weight for TP	diversification

$TabuAddTenure$ and $TabuDropTenure$ must be chosen properly. Too large $TabuTenure$ may result in degradation of solution quality because most candidate moves are prohibited, whereas too small $TabuTenure$ may result in frequent solution recycling.

$$TabuAddTenure = 2,\ TabuDropTenure = 1$$

(as suggested by [48] to produce high solution quality for most problems). For each candidate solution s, the maximum number of neighbor solutions allowed are

$$\lvert\mathcal{N}(s)\rvert = 0.2n^+N^*$$

so that sufficient number of candidate solutions are evaluated but not too many to cause unnecessary computational burden. Lastly, as already discussed in section 6.3.3, the penalty weight w is set as a function of the total number of demand nodes M and the number of BSs n^+ in s

$$w \approx 0.5n^+M$$

so that the diversification works effectively (most effective when the penalty and the objective function are in the same order). If the total number of α iterations are executed (including restarts), the TS-based algorithm presented in this paper will have to evaluate

$$S = 0.2\alpha n^{+} N^{*} \tag{6.36}$$

possible candidate solutions (search space size). These parameter values are applied to all the examples in section 7.2.3 and section 7.3 of the next chapter.

7.0 DESIGN RESULTS

The design objective is to employ the minimum number of BSs to satisfy both the coverage and capacity requirements. Accomplishing this goal requires that the computational requirement must be reduced, so that the optimal solution could be determined in a useful period of time. In this paper, a variety of example networks ranging from small to large are presented. Small network examples are used to illustrate the TP reduction algorithm. Large size examples convince how large the search space can be, and the need to incorporate the TP reduction algorithm to reduce the computation requirement. The minimum branching algorithm is applied to compute the coverage solution, while demand nodes are assigned to BSs in such a fashion that the total network capacity is maximized (optimal demand node assignment). The example designs of CDMA2000 networks supporting various data requirements are also illustrated.

7.1 SMALL NETWORKS

The examples presented in this section consider various example network topologies. The focus is on the TP reduction mechanism (the sweep and merge algorithm). Demand nodes are computed from user traffic assumed to have equal unit amplitude (i.e.,1 Erlang). Two extreme cases where $R \approx \infty$ and $R \approx 0$ are demonstrated first. The next example shows that the TP reduction algorithm always produces the optimal solution. The last example of this section illustrates step by step how the sweep and merge algorithm works.

7.1.1 $R \approx \infty$ Case

As discussed earlier, the number of TPs N^* is a function of both the demand node distribution and the coverage radius R. If R is larger than the entire service area,

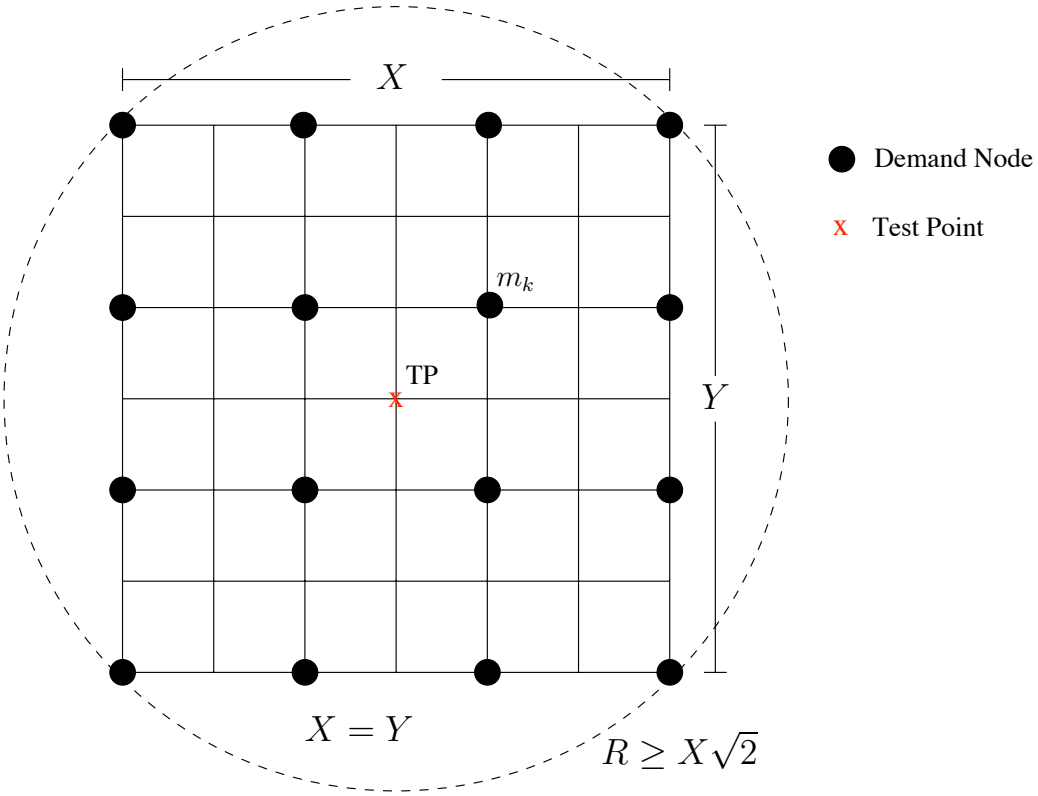

Figure 7.1: Infinite Coverage Radius, $R \approx \infty$

($R = \infty$) and only one TP at the CoG is always the result irrespective of the demand node distribution. In this case, the sweep process does not change or reduce the number of initial grid points, since every grid point shares the same set of demand node coverage. Thus, only one grid point closest to all demand nodes is selected by the merge process. The coverage requirement is satisfied.

7.1.2 $R \approx 0$ Case

The other extreme case occurs when the coverage radius is infinitesimally small, so small that the coverage area shrinks to a point.

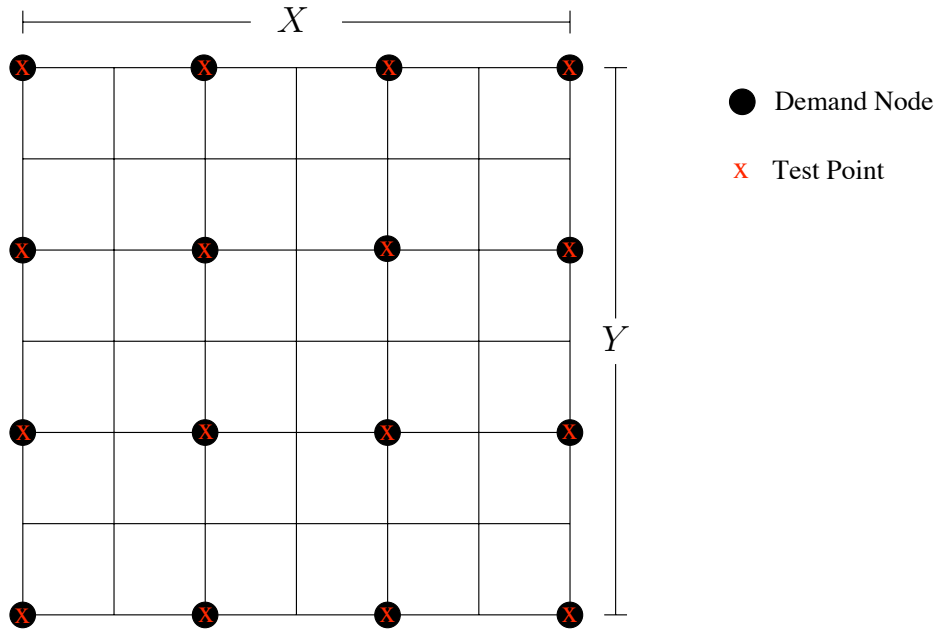

Figure 7.2: Zero Coverage Radius, $R \approx 0$

As a consequence, TPs only appear at the locations of the demand nodes as illustrated in figure 7.2. In this case, the sweep process is responsible for removing all the grid points. Both the coverage and capacity requirements are satisfied. BS must be located at each TP, and the minimum number of BSs n^* needed will be equal to the number of demand nodes M. When demand node density reaches the maximum and $R \approx 0$ (figure 7.2), the number of TPs approaches $N^* = N/4$. In most networks, demand nodes are either random or clustered in some spots, while the coverage radius R is finite (i.e., due to path loss). Thus, the resulting number of TPs $N^* \ll N$.

7.1.3 Greedy Flaw

Consider

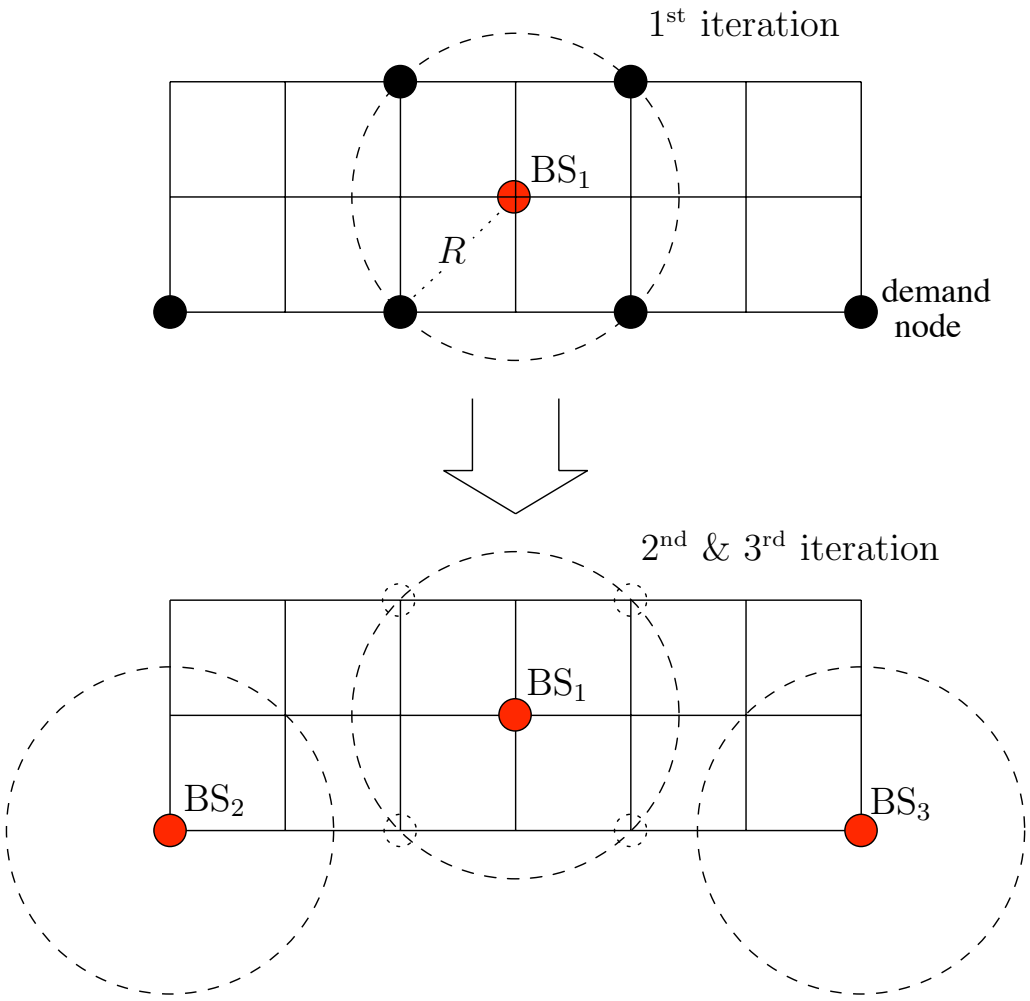

Figure 7.3: Greedy algorithm may occasionally produce sub-optimal solutions

the network of six demand nodes illustrated in figure 7.3. If a Greedy Algorithm is employed, then BSs are placed in such a fashion that the first BS always covers the maximum number of demand nodes and so on until all the demand nodes are covered. In this case, three BSs are needed — BS_1, first to be located, covers the maximum demand, while BS_2 and BS_3 cover the remaining left and right demand nodes respectively. The solution shown in the lower half of figure 7.3 is, however, not optimal. In contrast, the sweep and merge algorithm

yields,

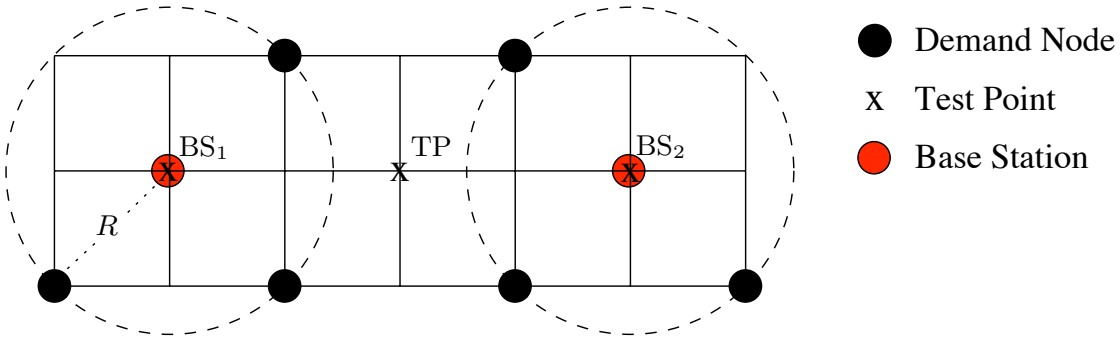

Figure 7.4: TP reduction always yields the optimal solution

three TPs as shown in figure 7.4. For this case, two BSs located on the left and on the right TPs are sufficient to satisfy the coverage requirement. This is the optimal solution because one BS is not sufficient to provide coverage to all demand nodes, and three BSs (as in the Greedy case) are more than necessary.

7.1.4 TP reduction illustrated

The example shown in figure 7.5 is used to illustrate step by step the progression TP reduction algorithm. The size of network (rectangular in this case) and the number of demand nodes are arbitrarily chosen. Since the TP reduction process is not trivial, thus only M = 13 demand nodes are used and randomly placed in a small rectangular area of size $X = 700, Y = 1000$ m. For simplicity, each demand node is assumed to have 1 unit of demand (i.e., 1 Erlang).

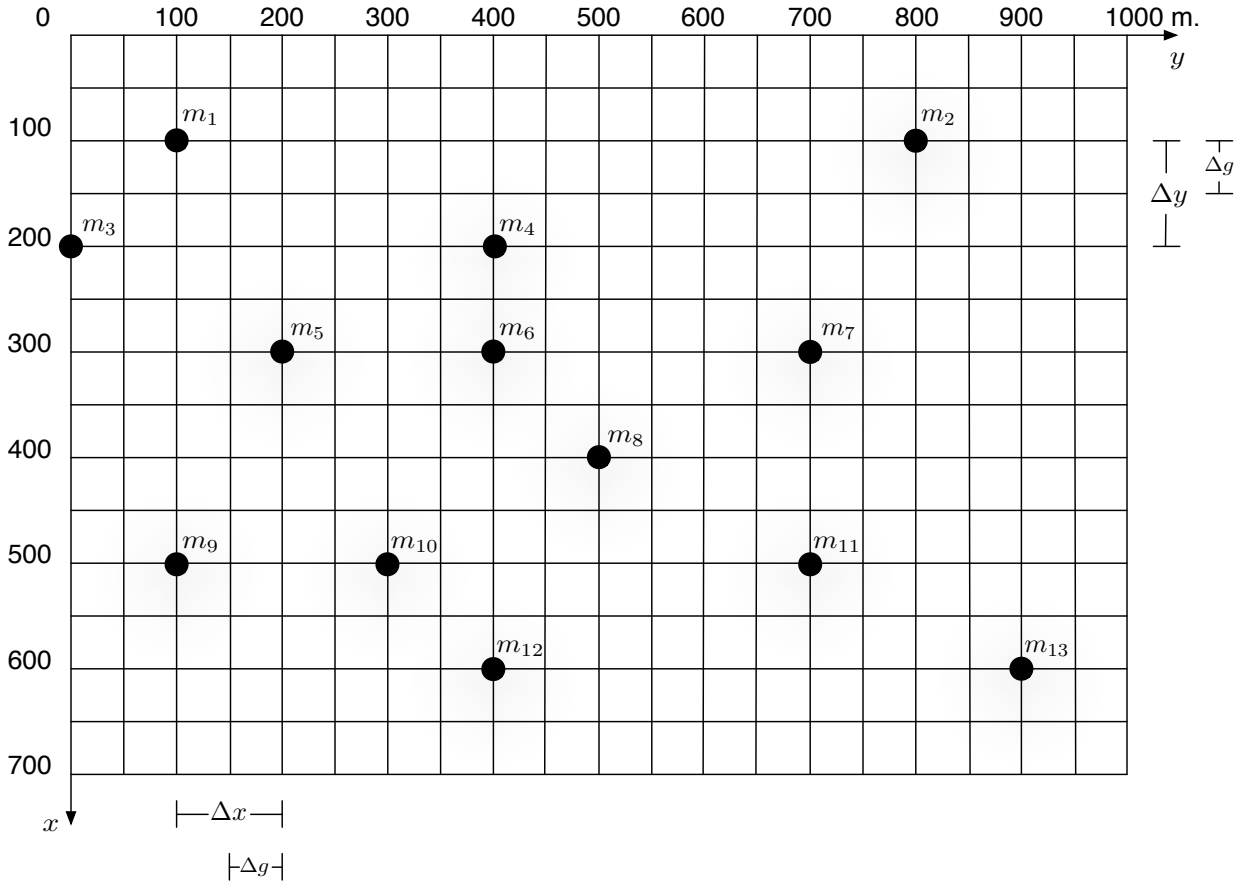

Figure 7.5: The example network used to illustrate the TP reduction process (adapted from [53])

The demand node grid spacing is set to $\Delta x = \Delta y = 100$ m to limit the number of TPs. To guarantee the optimal solution, the TP grid spacing must be $\Delta g = \Delta x/2 = 50$ m.

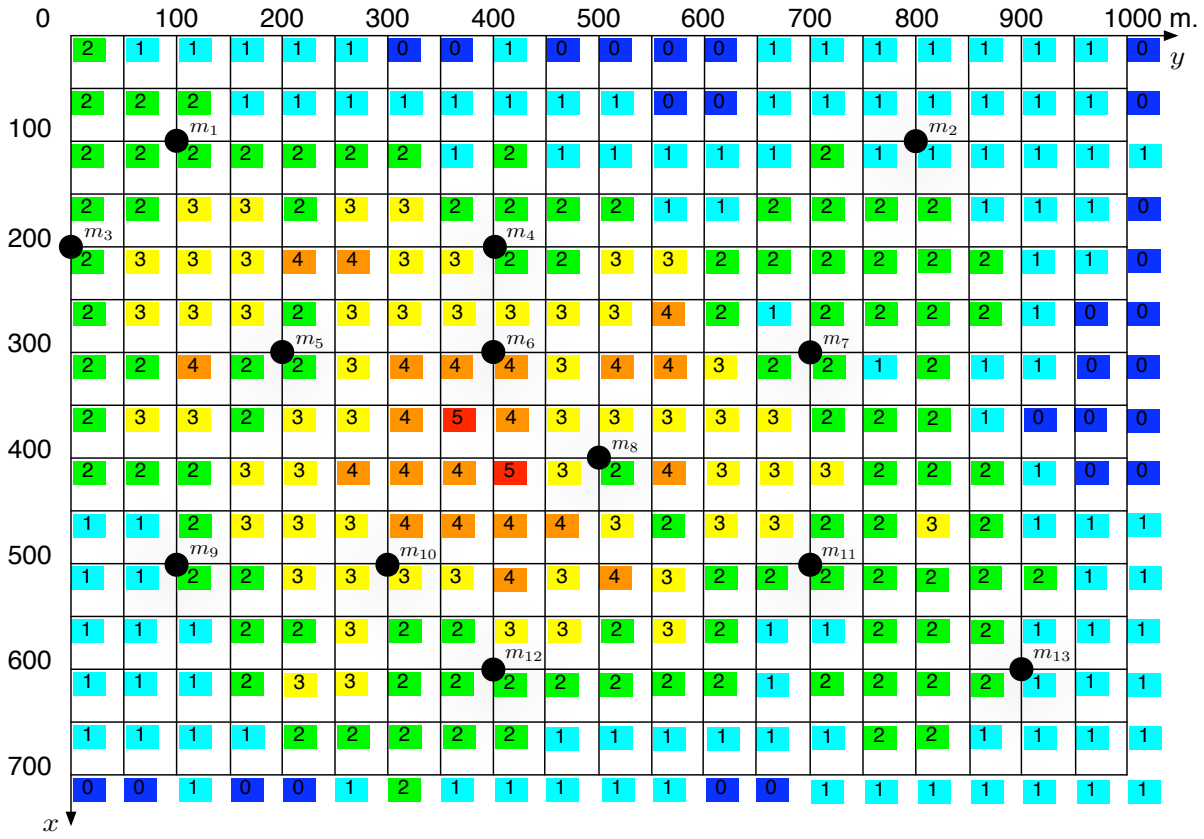

Figure 7.6: Grid Point Coverage. The figure shows the total number of demand nodes covered by each grid point, assuming that the coverage radius $R = 200$ m.

Figure 7.6 shows the total number of demand nodes covered by each grid point. The number highlighted in the upper left corner is the total number of demand nodes within the coverage radius R from that grid point. The dark blue color indicates zero coverage, while red indicates the maximum coverage, which is equal to 5 demand nodes in this case. To compute the TPs, the sweep process is applied first. As discussed earlier, four sweeping directions are used: horizontal, vertical, up-diagonal, and down-diagonal sweeps.

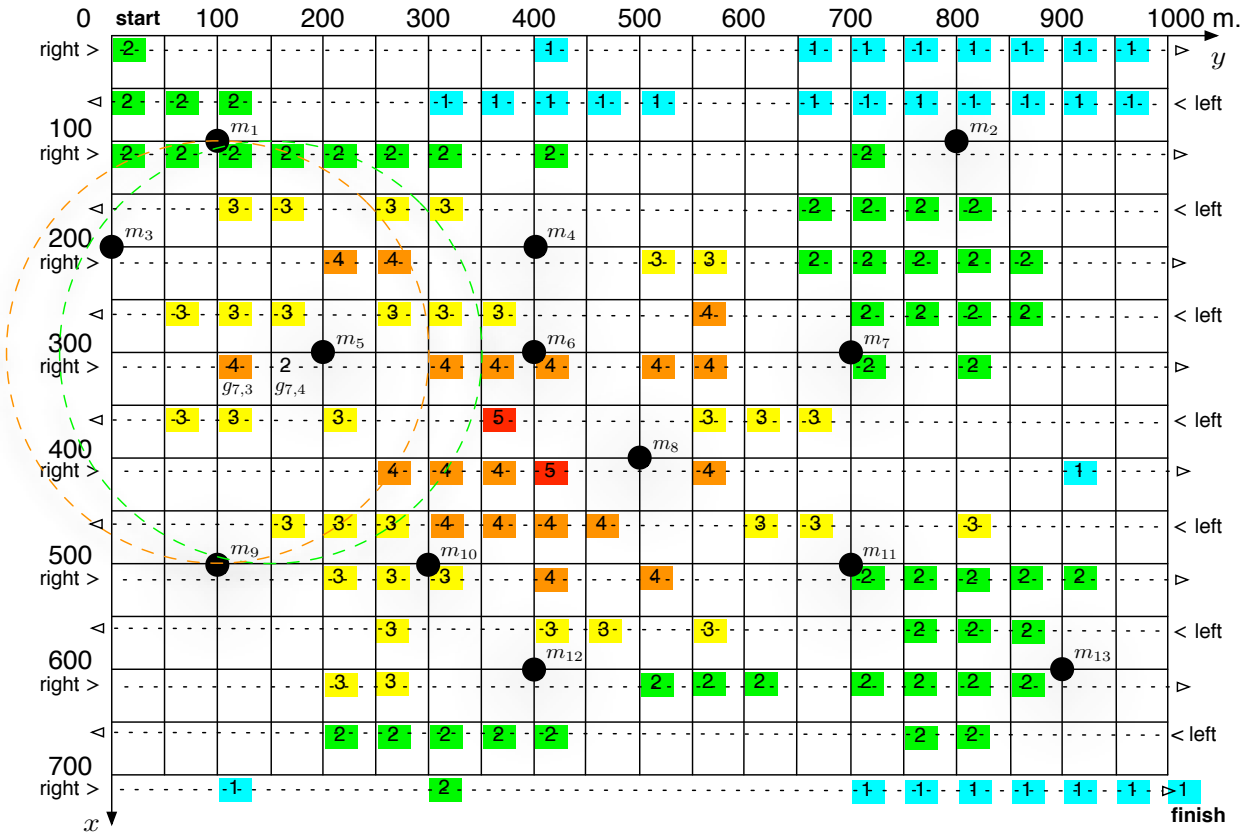

Figure 7.7: Horizontal Sweep. The sweep starts from the upper left corner grid at (0,0), moves to the right toward the grid at (0, 1000), turns around and moves to the left and so on. This alternate pattern repeats until the sweep reaches the final grid at the lower right corner. The resulting matrix of grid points $G^{\rightarrow}$ after the sweep are shown. Grid points not highlighted are removed by the sweep.

Figure 7.7 shows grid points remaining after applying the horizontal sweep. The sweep process compares the set coverage of all adjacent grid points. As an example, let consider the set coverage of grid $g_{7,3}$ and grid $g_{7,4}$ shown in the above figure. The set coverage $z_{i,j}$ at grid $g_{7,3}$ is (demand nodes within the coverage radius R form grid $g_{7,3}$)

$$z_{7,3} = \{m_1, m_3, m_5, m_9\}$$

whereas grid $g_{7,4}$ covers

$$z_{7,4} = \{m_3, m_5\}$$

Clearly, $z_{7,4} \subset z_{7,3}$, and therefore grid $g_{7,4}$ is not necessary and is eliminated.

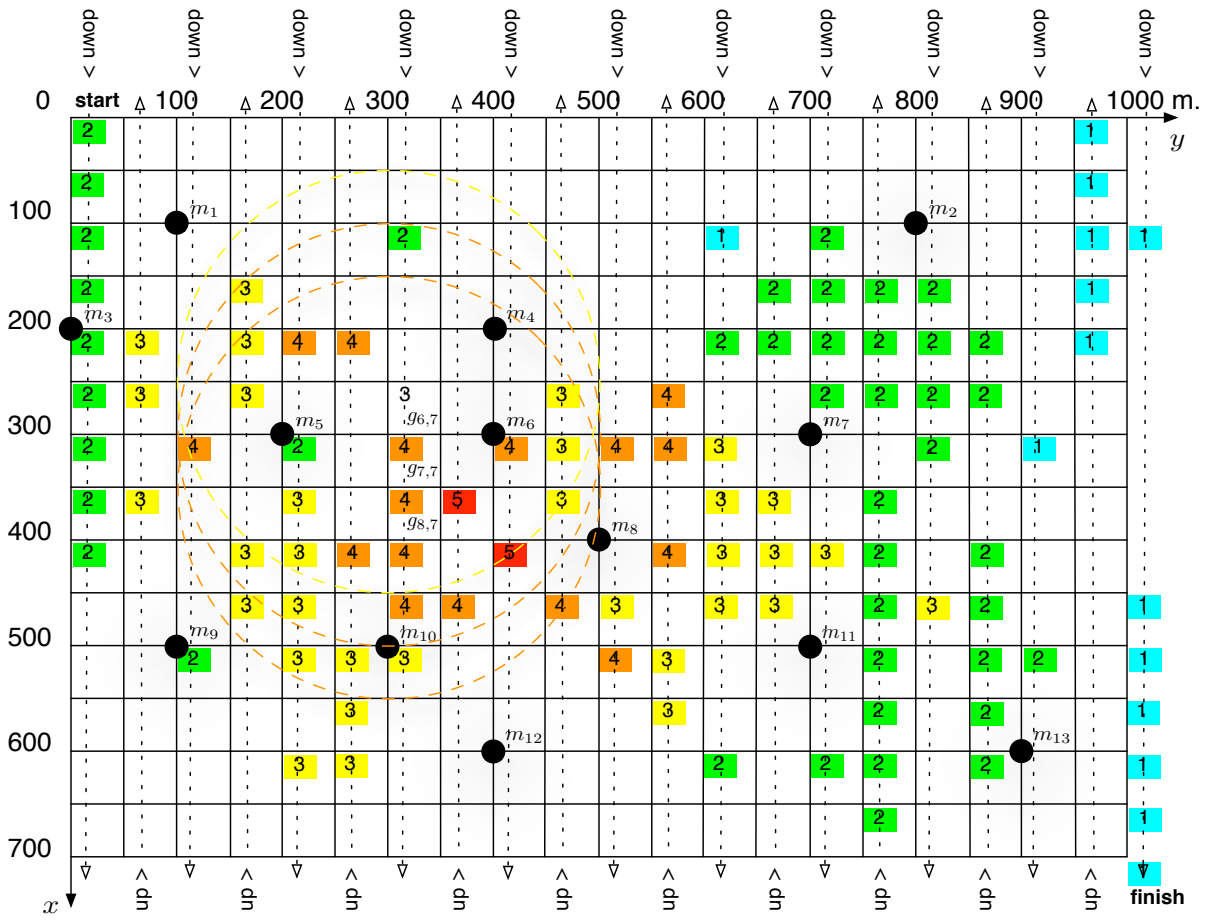

Figure 7.8: Vertical Sweep. The sweep starts down from the upper left corner grid at (0,0) toward the lower grid at (700, 0), turns around, moves up and so on. This alternate pattern repeats until the sweep reaches the final grid at the lower right corner. The resulting matrix of grid points $G^{\downarrow}$ after the sweep are shown. Grid points not highlighted are removed by the sweep

Figure 7.8 shows the result after applying the vertical sweep. Similar to the horizontal sweep, the set coverage of all adjacent grid points are compared but now on the vertical direction (either up or down). Let consider grid points $g_{6,7}$, $g_{7,7}$, and $g_{8,7}$. The corresponding set coverage of each grid point are

$$z_{6,7} = \{m_4, m_5, m_6\}$$

and

$$z_{7,7} = z_{8,7} = \{m_4, m_5, m_6, m_{10}\}$$

Since, $z_{6,7} \subset z_{7,7}$, thus the grid $g_{6,7}$ is not necessary and is removed. On the other hand,

$z_{7,7} = z_{8,7}$ (covering the same set of demand nodes), both grids $g_{7,7}$ and $g_{8,7}$ are therefore not eliminated by the sweep. The intersection result $G^{\searrow} \times G^{\nearrow}$ of the up- and down-diagonal sweeps are shown in the front pane of figure 7.9.

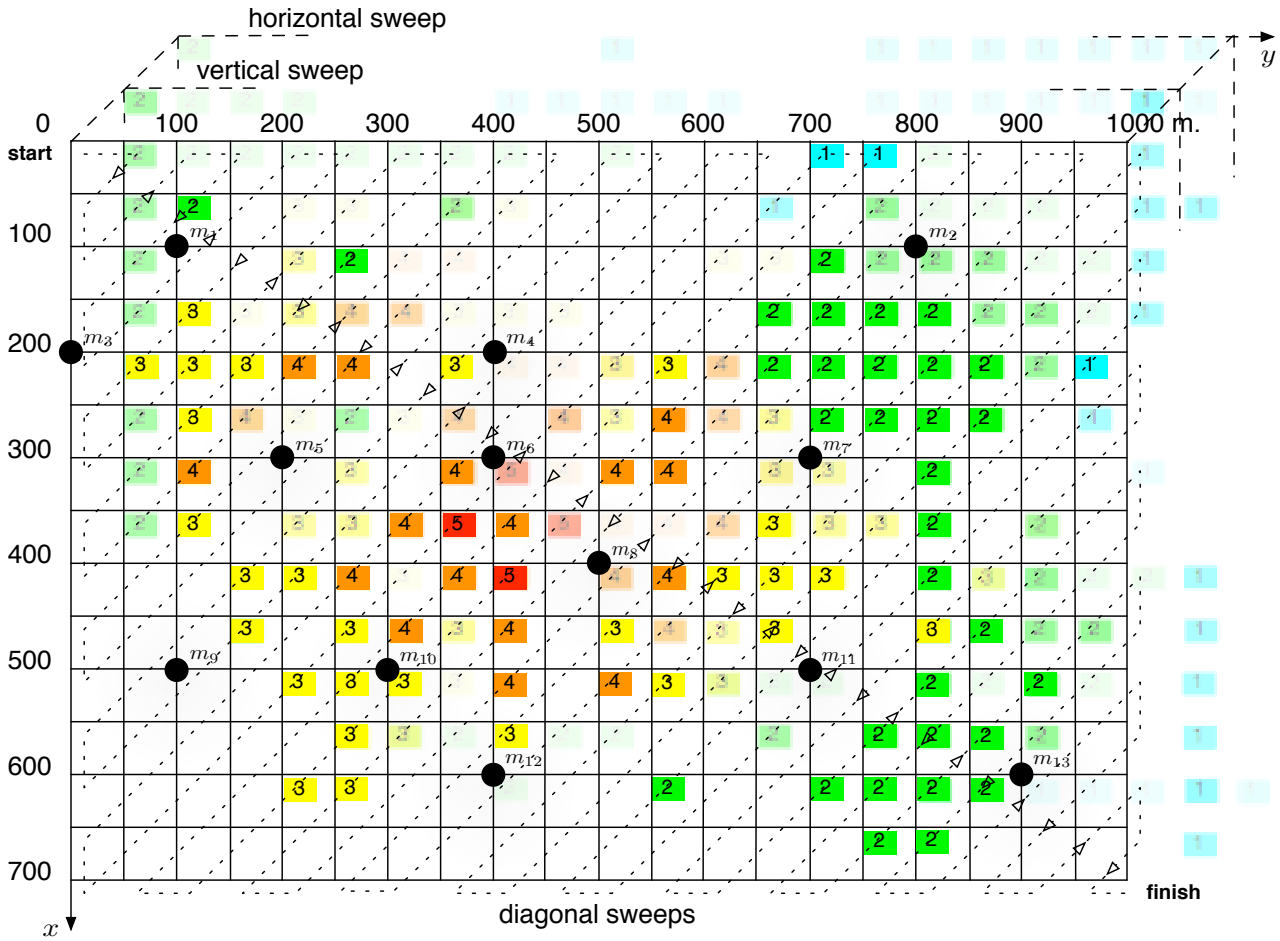

Figure 7.9: Diagonal Sweeps. The front pane shows the resulting matrix $G^{\searrow} \times G^{\nearrow}$ from both the up- and down diagonal sweeps. Only the down-diagonal sweeping pattern is illustrated here. Two of the background panes again show the results from the previous vertical and horizontal sweeps respectively.

The intermediate result is the intersection of all the sweeps, in which all the grid points removed from the horizontal, vertical, or diagonal sweep are also removed from the intermediate result G^{sweep} .

$$G^{\text{sweep}} = G^{\rightarrow} \times G^{\downarrow} \times G^{\searrow} \times G^{\nearrow}$$

where $\times$ denotes set intersection.

Recursion is also applied to G^{sweep} to ensure that, the grid points remaining cover demand nodes not a subset of demand nodes covered by other grid points. Figure 7.10 shows the resulting grid points after the recursion.

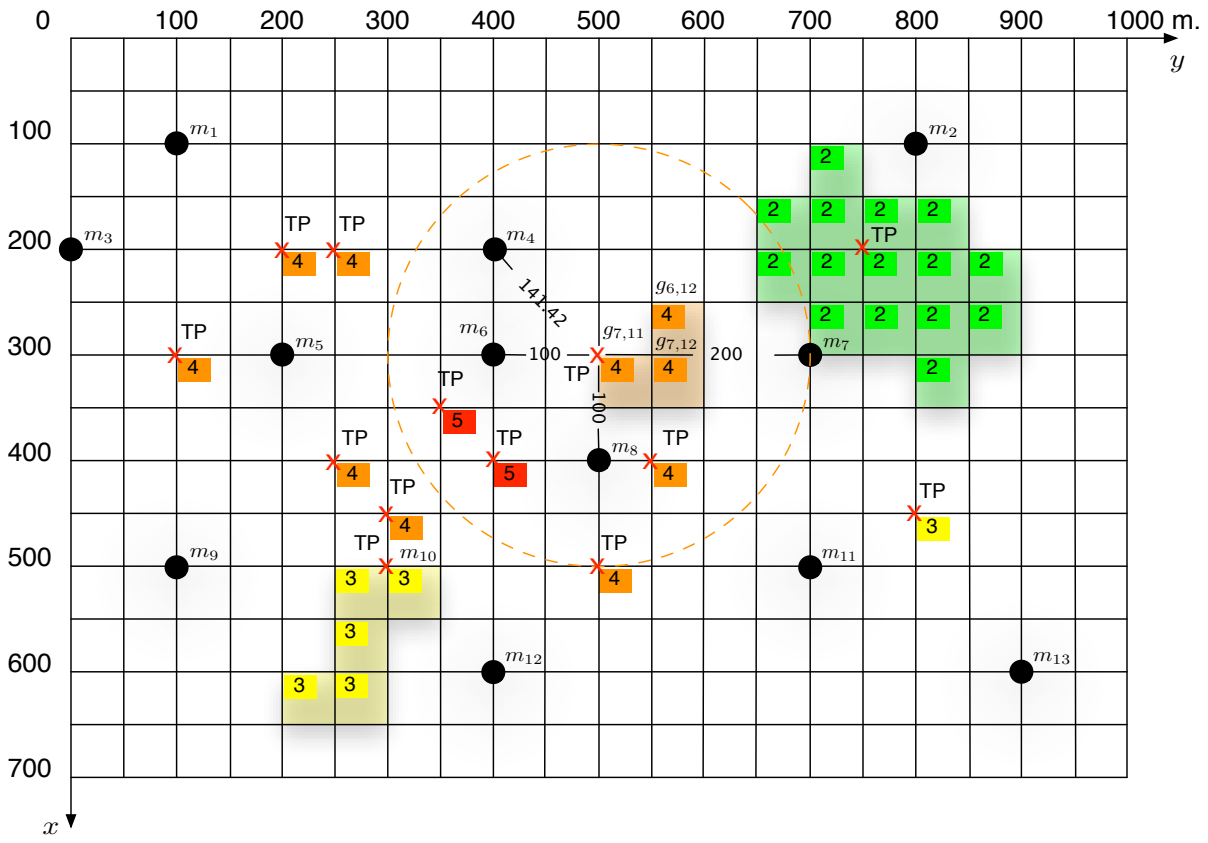

Figure 7.10: Merge Process. The process selects one grid point (closest to all demand nodes within the coverage radius) from a group covering the same set of demand nodes.

Since the grid points $\in G^{\text{sweep}}$ may still cover the same set of demand nodes (redundant grid points). As shown in figure 7.10, grid points $g_{6,12}$, $g_{7,11}$, and $g_{7,12}$, all cover demand nodes $\{m_4, m_6, m_7, m_8\}$. Thus, selecting either $g_{6,12}$, $g_{7,11}$, or $g_{7,12}$ does not effect the demand node coverage. By comparing the average distance from the demand nodes $\{m_4, m_6, m_7, m_8\}$ to these grid points:

$$\text{avg. distance to grid } g_{7,11} = (141.42 + 100 + 200 + 100)/4 \approx 135 \text{ m}$$

is the closest to all demand among the three (to grid $g_{7,12} \approx 148$ m, and to grid $g_{6,12} \approx 158$ m). Grid $g_{7,11}$ is then selected as a TP. The merge process produces $N^* = 13$ TPs. Figure 7.11 shows the resulting TPs and the coverage solution.

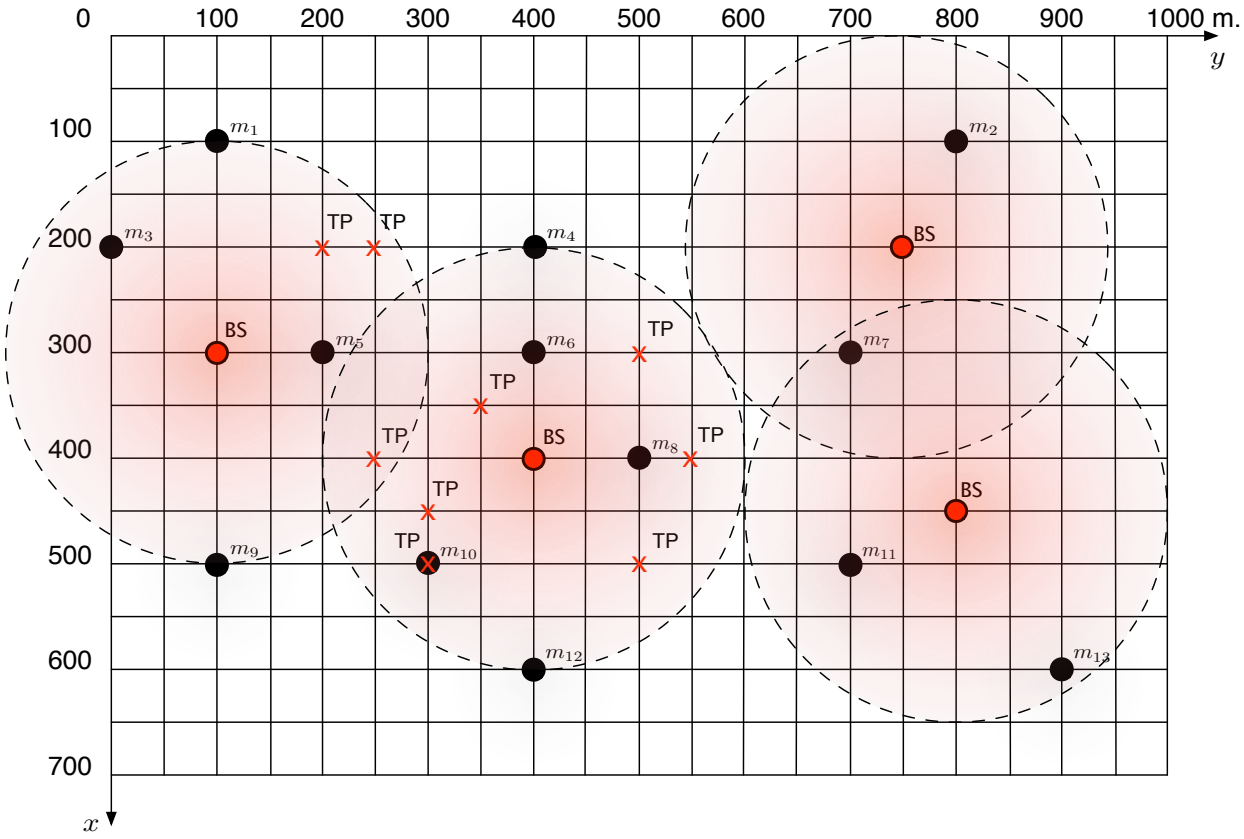

Figure 7.11: Coverage Solution

Using the exhaustive search, four BSs are needed to satisfy the coverage requirement, which in this case is the optimal solution ($n^* = 4$). The total number of grid points initially needed are

$$N = \left\lfloor \frac{X}{\Delta g} + 1 \right\rfloor \left\lfloor \frac{Y}{\Delta g} + 1 \right\rfloor = \left\lfloor \frac{700}{50} + 1 \right\rfloor \left\lfloor \frac{1000}{50} + 1 \right\rfloor = 15 * 21 = 315$$

Computing the optimal solution by employing the TP reduction thus reduces computations by a factor

$$\left(\frac{N}{N^*}\right)^{n^*} = \left(\frac{315}{13}\right)^4 \approx 3.5 \times 10^5$$

7.2 LARGE NETWORKS

Two types of networks are considered, one with random distribution of demand nodes, and the other with clustered distribution of demand nodes. The minimum branching algorithm is applied to both cases to compute for the coverage solution (instead of the exhaustive search).

7.2.1 Random Distribution

Figure 7.12 shows the example network with random demand node distribution. $M = 60$ demand nodes are chosen to represent moderate user density. For simplicity, let assume that

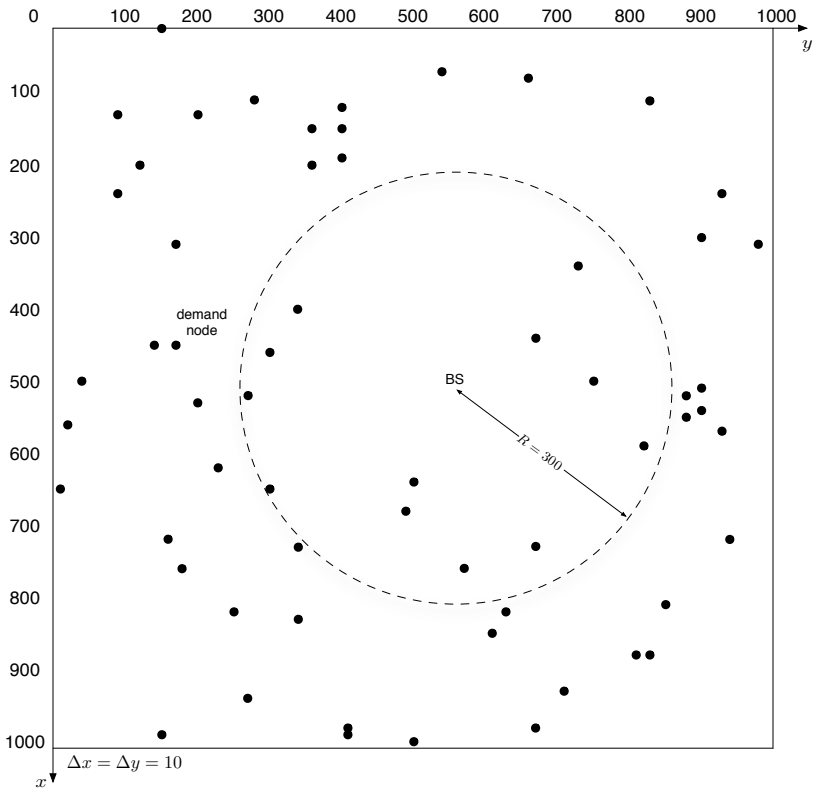

Figure 7.12: Random Demand Node Example

each demand node m_k has one unit of demand (i.e., 1 Erlang). Demand node grid spacing is set to $\Delta x = \Delta y = 10$ m to limit the number of computations needed, and therefore the

required TP grid spacing $\Delta g = \Delta x/2 = 5$ m in order to guarantee the optimal solution. The coverage radius $R = 300$ m is used to mimic the coverage of a picocell [54]. Applying the sweep and merge algorithm then produces the TPs shown in figure 7.13

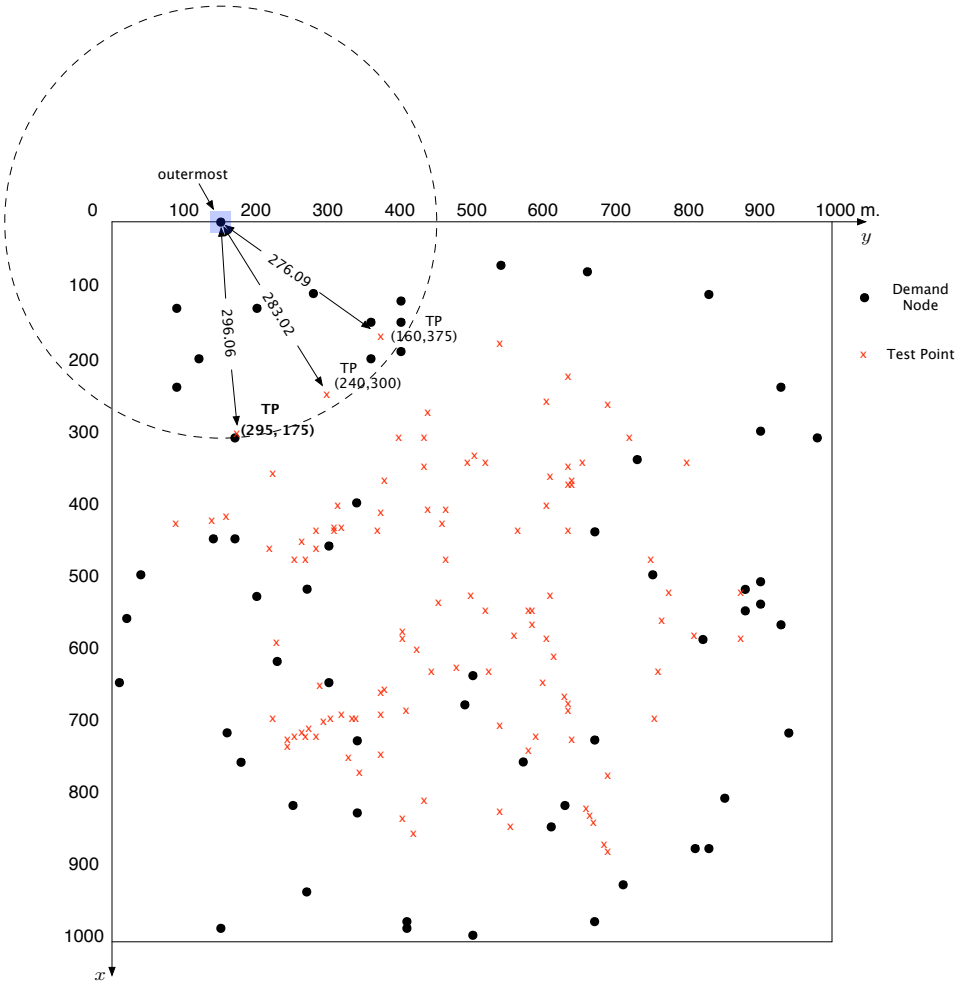

Figure 7.13: TPs computed from the sweep and merge processes

There are a total of 111 TPs throughout the space, compared to $N = 40,401$ initial grid points. The resulting set of 111 TPs is guaranteed to contain the optimal solution. Figure 7.13 also illustrates the minimum branching algorithm in progress. The minimum branching algorithm locates the demand node associated with the fewest number of TPs — the outermost demand node. In figure 7.13, the outermost demand node is the demand node at (0, 150), associated with only three TPs at (295, 175), (240, 300), (160, 375). The minimum

branching algorithm builds a tree of coverage solutions by first selecting the TP at (295, 175) as the first BS_1. Demand nodes covered by BS_1 are ignored. Reapplying the sweep and merge processes to the uncovered demand nodes then produces a new set of TPs shown in figure 7.14.

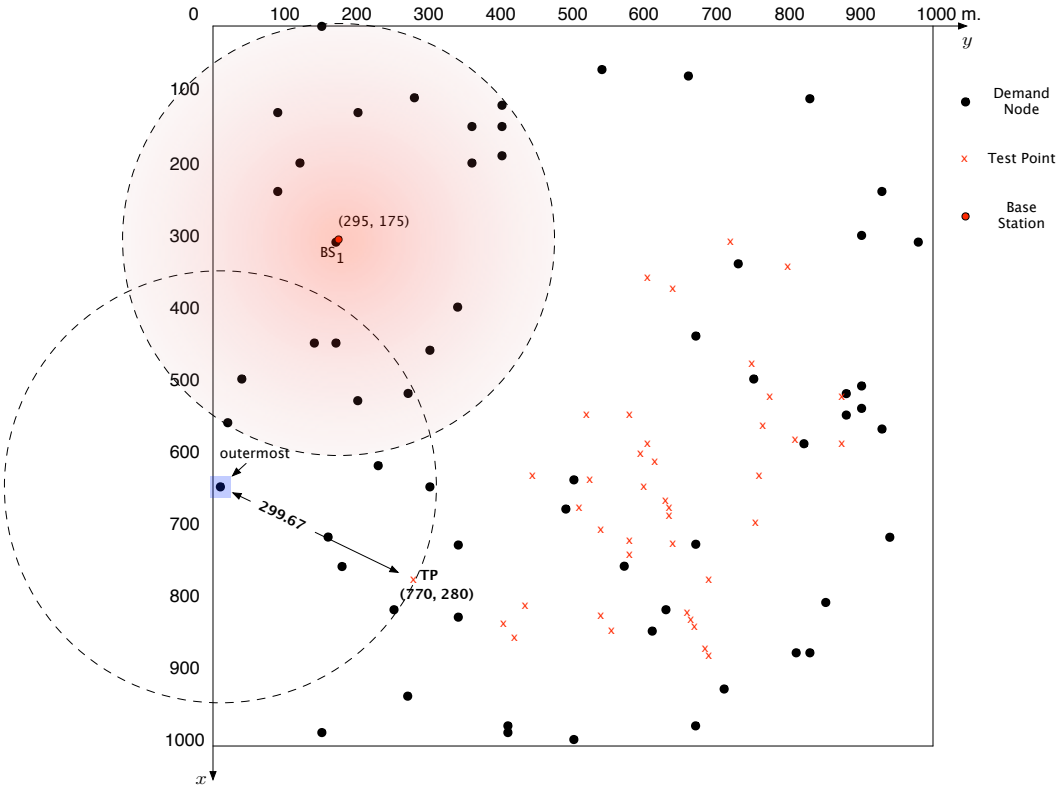

Figure 7.14: Minimum Branching in Progress (2^{nd} recursion).

The TP at (295, 175) selected in the first recursion is now BS_1 in the second recursion. Since there are fewer demand nodes (40), there are also fewer TPs (40) as a result. The demand node at (640, 10) in figure 7.14 is associated with one TP at (770, 280) (only TP within the coverage radius), and therefore is the outermost demand node. The TP at (770, 280) is then chosen as the next BS_2 in the solution tree. For the third recursion, demand nodes associated with BS_2 are again removed, the sweep and merge process is applied to compute a new set of TPs, the minimum branching algorithm then searches for the outermost demand node, the next BS, and so on until all demand nodes are removed. After five recursions, there are

no demand nodes left in the network, and the first feasible solution is found (assuming that the BS capacity, $C = \infty$). Figure 7.15 shows the resulting BS placement after the first five recursions.

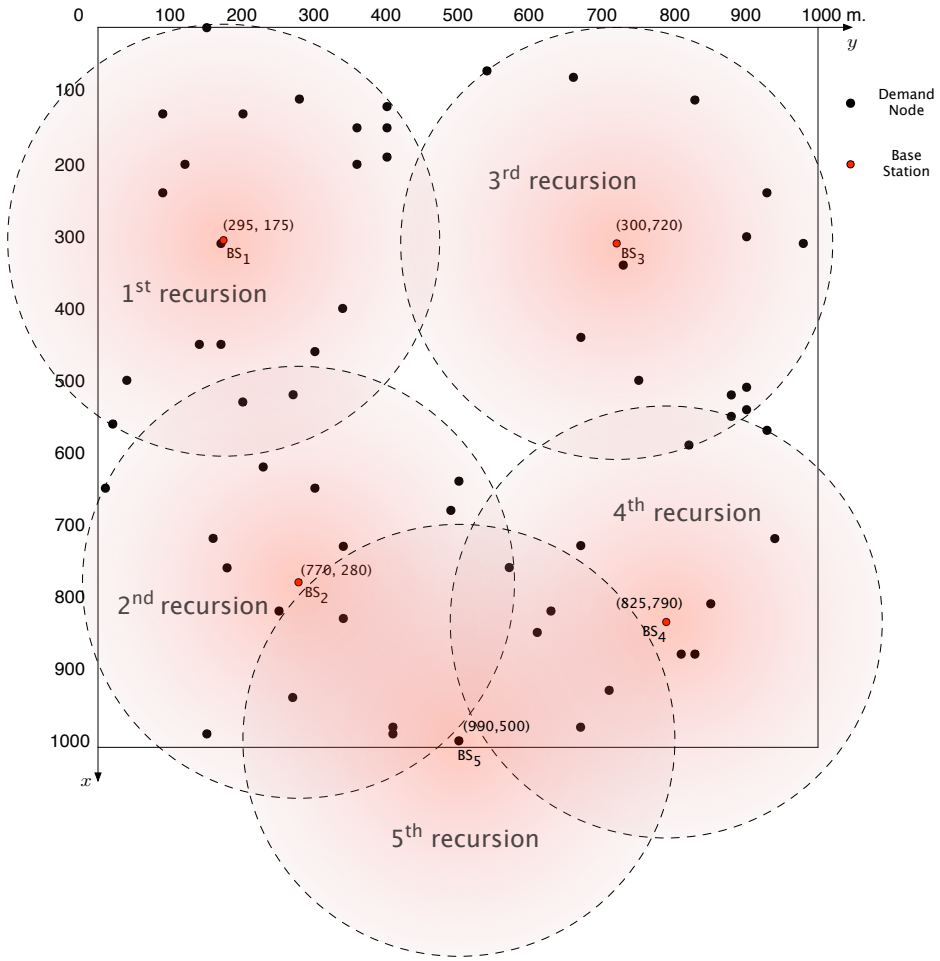

Figure 7.15: BS placement. 5 BSs are needed to satisfy the coverage requirement

The third and fourth recursion produce 8 and 2 TPs, and the results are BS_3 and BS_4 respectively. For the fifth recursion, since there is only one demand node left at (990, 500), this will be the location of the last TP and BS_5. The first five recursions are executed step by step down the tree depth.

The minimum branching requires a total of 19 recursions as shown in figure 7.16 below. Recursion #0 is always a root of the tree. The tree depth represents the total # of BSs

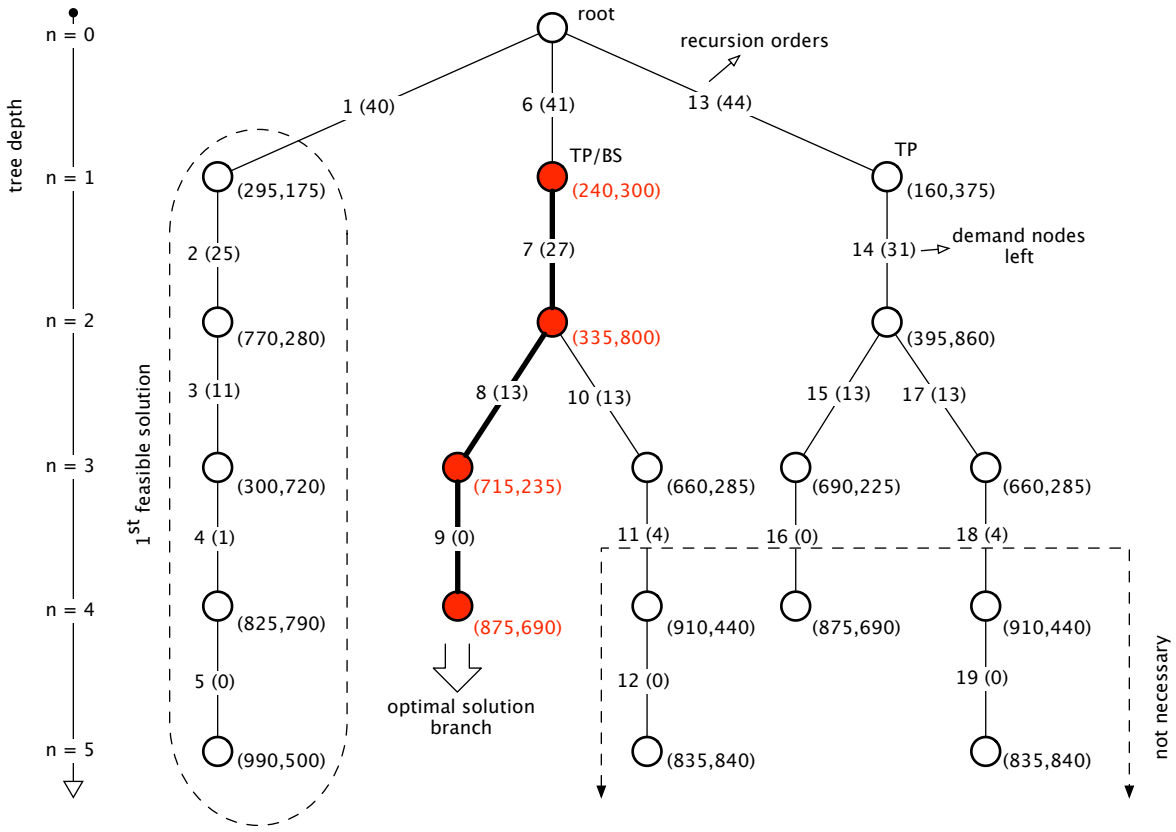

Figure 7.16: Minimum Branching Tree

needed. The tree produces five feasible solutions (# of branches from the root node to the terminating node), two of which are the optimal solution (require fewest # of BSs to satisfy the coverage requirement). The optimal solution branch and its TP nodes are shown in red, and only $n^* = 4$ BSs are needed to cover all the demand nodes. To reduce the number of recursions, the algorithm discards nodes that are at equal or higher than the tree depth $n^* = 4$, reducing the number of recursions to 14. The optimal solution is shown in figure 7.17

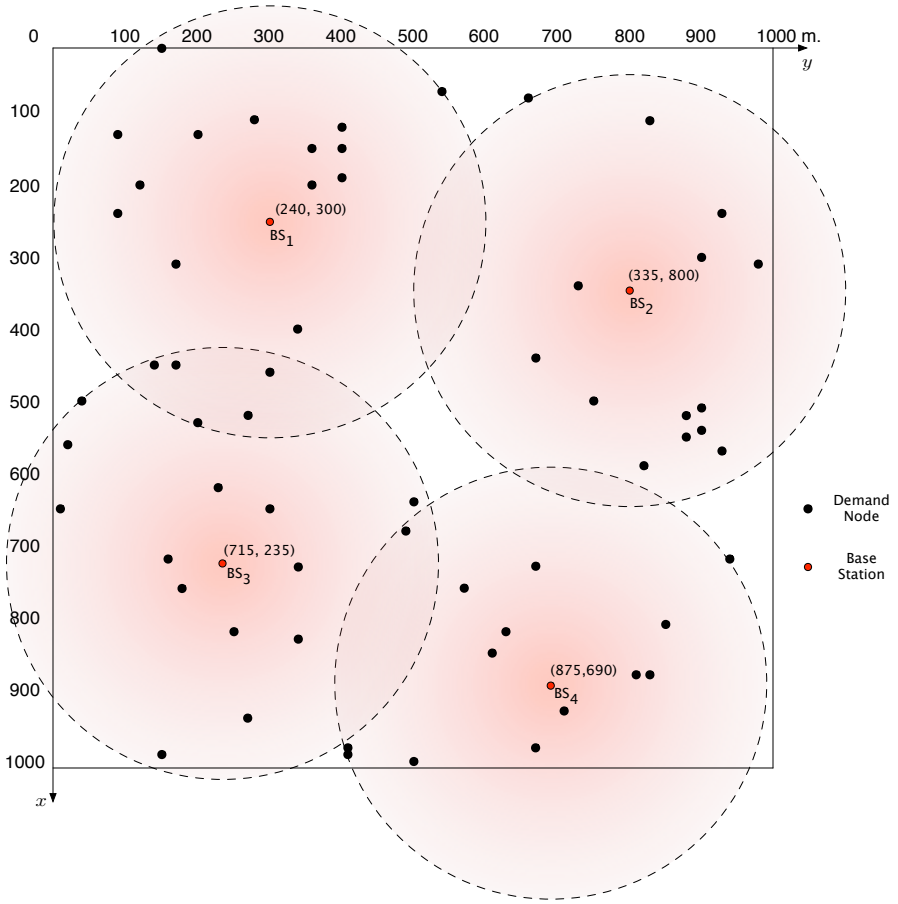

Figure 7.17: Optimal Coverage Solution. Only 4 BSs are needed to satisfy the coverage requirement

Employing the minimum branching algorithm, the search space size S is reduced from

$$\binom{N^*}{n^*} = \binom{111}{4} = 5,989,005 = \text{ to } 14 \text{ (total number of recursions)}$$

Together, the TP reduction and the minimum branching algorithms reduces the computation requirement by a factor (section 6.1.2)

$$\frac{C'(\text{exhaustive search})}{C'(\text{minimum branching})} \approx M\binom{N}{n^*}/14N = 60\binom{40,401}{4}/(14 * 40,401) \approx 1.18 \times 10^{13} \quad (7.1)$$

7.2.2 Fading Effect

To illustrate the effect of fading, the following example is considered. The same network with $M = 60$ demand nodes randomly located in 1000×1000 m^2 from the previous example is used. However, to compensate for the shadowing loss ζ_k, which is modeled as a log-normal random variable with zero mean and standard deviation σ dB [40], the fading margin γ must be added so that the coverage radius is reduced. As an example, using the two-ray model described in section 3.2 (which is suitable for the coverage radius R less than 1 km) to determine the path loss

$$P_{kn} = P_0 + \zeta_k + \begin{cases} 10n_1 \log_{10}(r_{kn}), & r \leq d_f \\ 10n_2 \log_{10}(r_{kn}/d_f) + 10n_1 \log_{10} d_f, & r > d_f \end{cases} \tag{7.2}$$

where n_1 and n_2 is the path loss exponent before and after break point distance d_f, and r_{kn} is the distance between BS$_n$ and demand node m_k. Without the fading, the previous example assumes that the coverage radius $= R$, which translates to the maximum path loss

$$P_{\max} = P_0 + 10n_2 \log_{10}(R/d_f) + 10n_1 \log_{10} d_f \tag{7.3}$$

Including the fading margin γ, then

$$P_{kn} + \gamma = P_{\max} \tag{7.4}$$

Thus, the new coverage radius R' can be computed as

$$\begin{aligned} P_0 + 10n_2 \log_{10}(R'/d_f) + 10n_1 \log_{10} d_f + \gamma &= P_{\max} \\ 10n_2 log(R'/d_f) &= 10n_2 log(R/d_f) - \gamma \\ R' &= R10^{\frac{\gamma}{10n_2}} \end{aligned} \tag{7.5}$$

As shown by [40] (when soft handoff is employed), to achieve 90% coverage, if $\sigma = 8$ dB, the fading margin must be at least $\gamma = 6.2$ dB. Therefore, for $R = 300$ m in the previous example, $n_2 = 4$ [36], using (7.5) the new coverage radius is

$$R' = 300 * 10^{(6.2/40)} \approx 210 \text{ m} \tag{7.6}$$

To account for the fading, the coverage radius shrinks from $R = 300$ to $R' = 210$ m. Since the same demand nodes from the previous example is used, the initial number of grid points needed to guarantee the optimal solution does not change ($N = 40,401$). However, with the new coverage radius $R' = 210$ m, a new set of TPs must be recomputed. Figure 7.18 shows the resulting new set of TPs from the sweep and merge algorithm when $R' = 210$ m.

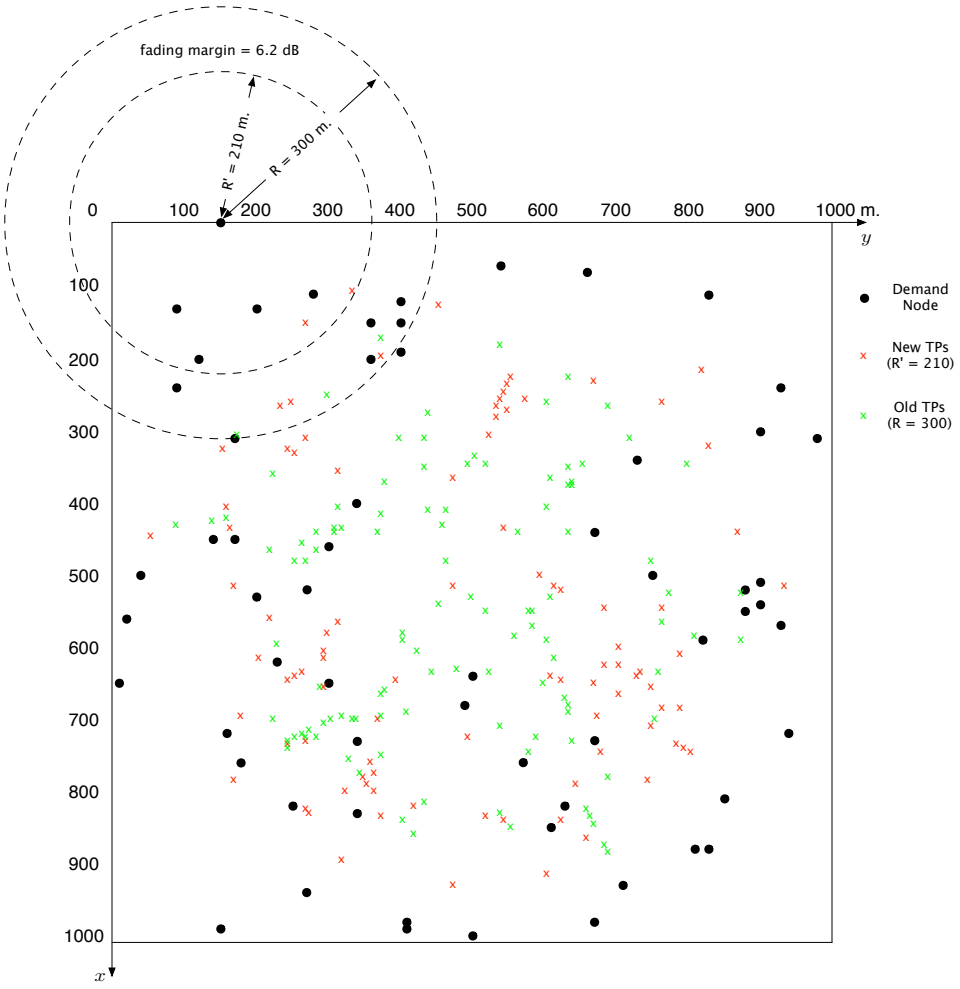

Figure 7.18: New Set of TPs (Fading Margin Included). A new set of $N^* = 93$ TPs (as shown in red x) when $R' = 210$ m with fading margin included, compared to 111 TPs (as shown in green x) when $R = 300$ m.

With the coverage radius reduced to $R' = 210$ to compensate for the fading margin $\gamma = 6.2$ dB, the sweep and merge algorithm results in a new set of $N^* = 93$ TPs, 18 TPs fewer than when $R = 300$ m. The new set of $N^* = 93$ TPs are also distributed throughout the space,

but are more spread out than the previous set of TPs — since the coverage radius $R' < R$. Figure 7.19 shows the corresponding new coverage solution.

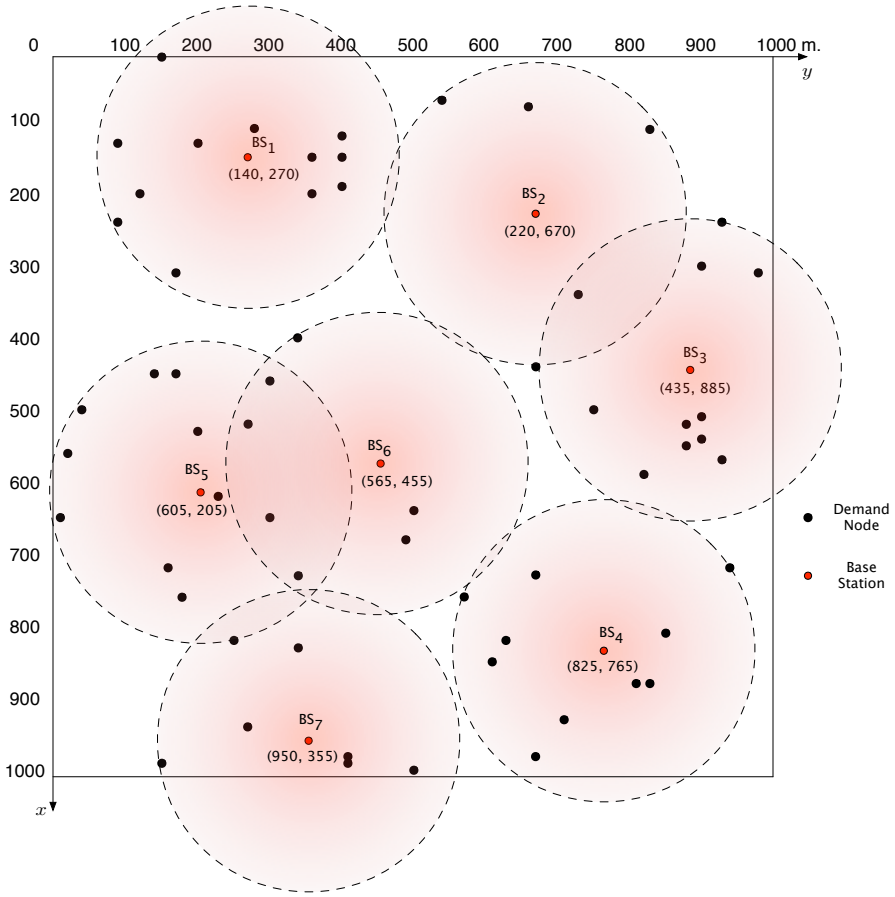

Figure 7.19: New Optimal Coverage Solution (Fading Margin Included) Now, 7 BSs are needed to satisfy the coverage requirement when $R' = 210$ m.

As expected, since the new coverage radius is smaller, more BSs are needed to provide the coverage. With $R' = 210$, the minimum branching algorithm (20 recursions in total) now produces 7 BSs, 3 more BSs when no fading margin is included (as in the case when $R = 300$ m). In summary, adding the fading margin in the path loss equation results in a smaller coverage radius, a new set of TPs must be recomputed, and additional BSs are needed to satisfy the coverage requirement.

7.2.3 Clustered Distribution

The previous examples considered the networks with random distribution of demand nodes. As a result, TP distribution is also random. Now consider the case where demand nodes are clustered as might be seen in urban environments. To be more general than the previous example, a larger network with $M = 200$ is used; three clustered of demand nodes are randomly generated and placed in a rectangular area with size $X = 1500$ and $Y = 2500$ m. One cluster is located at the middle, the other two on the top and lower right respectively. Since the network is larger, the demand node grid spacing is set to $\Delta x = \Delta y = 20$ m to limit the number of computations so that the solution can be computed in a reasonable time period. To guarantee the optimal solution, the TP grid spacing must therefore be $\Delta g = \Delta x/2 = 10$ m. Figure 7.20 shows the example network.

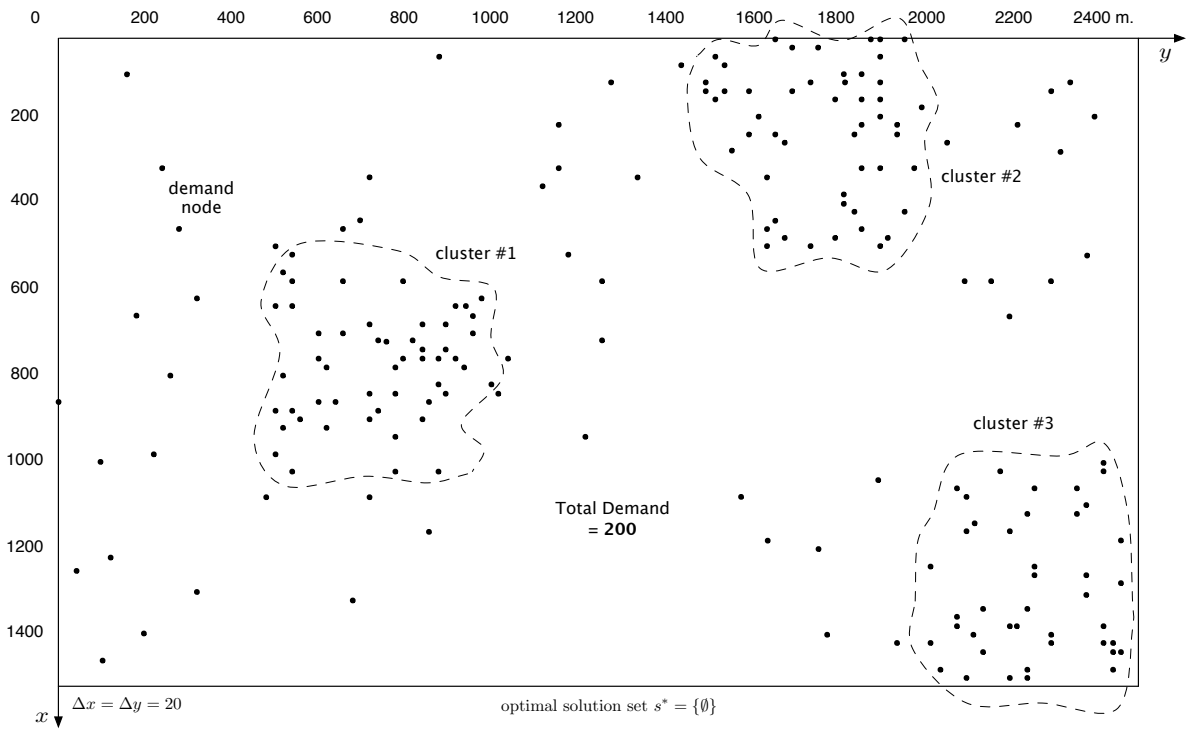

Figure 7.20: Clustered Demand Node Example. $M = 200$ demand nodes are randomly generated. However, three areas are weighted more than the others, resulting in three clusters of demand nodes.

In this example, the BS radius is set to $R = 500$ m to represent the coverage of a small microcell [54], which includes all the margins required to compensate for the shadow fading and path loss, while the BS capacity $C = 40$ is chosen to limit the total number of BSs needed. The total number of initial grid points $N = (1500/10+1)*(2500/10+1) = 37,901$. Applying the sweep and merge algorithm reduces the number of grid points from $N = 37,901$ to $N^* = 459$ TPs. The resulting TPs are shown in figure 7.21.

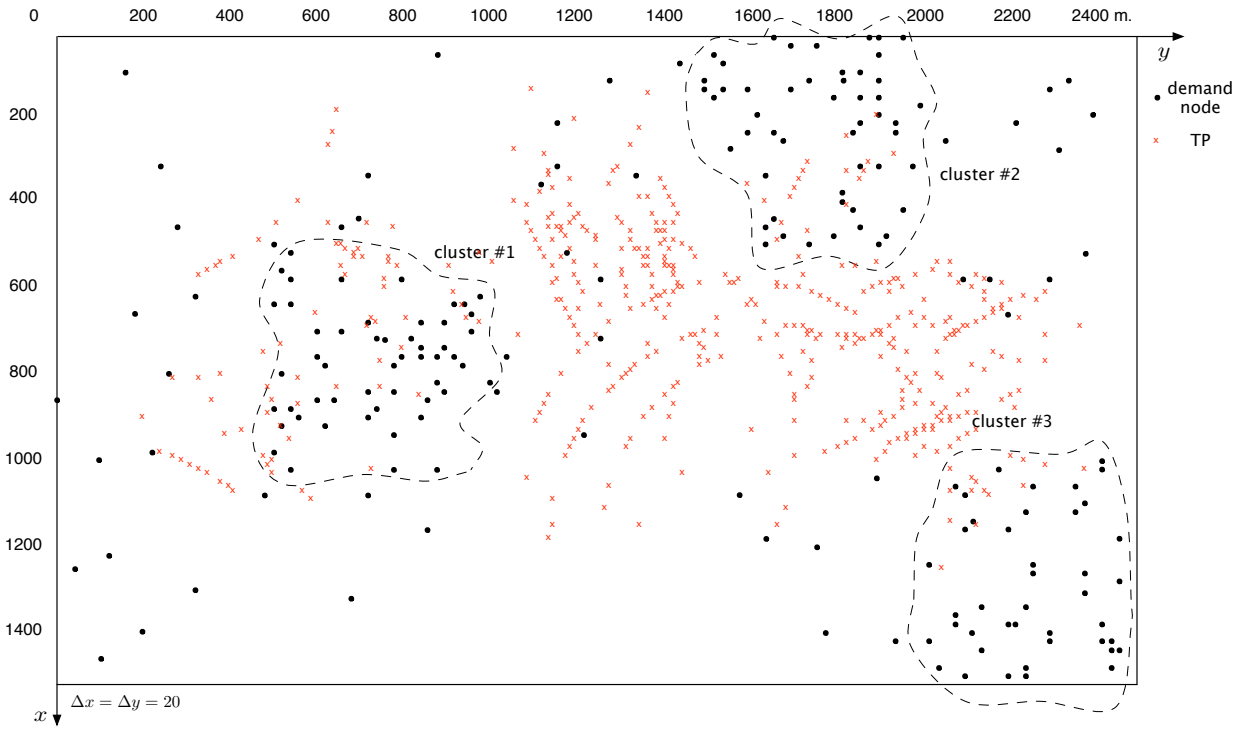

Figure 7.21: Test Points (Clustered Demand Node Example). Out of $N = 37,901$ grid points, the sweep and merge algorithm removes unnecessary grid points that are not required for the optimal solution, reducing the total number to only $N^* = 459$ TPs.

Since demand node are clustered, TPs are also clustered, and most of them are located in the middle among the three clusters of demand nodes. These TPs are guaranteed to contain the optimal solution. The design first employs the minimum branching algorithm to solve for the coverage solution. Figure 7.22 shows the corresponding coverage solution tree.

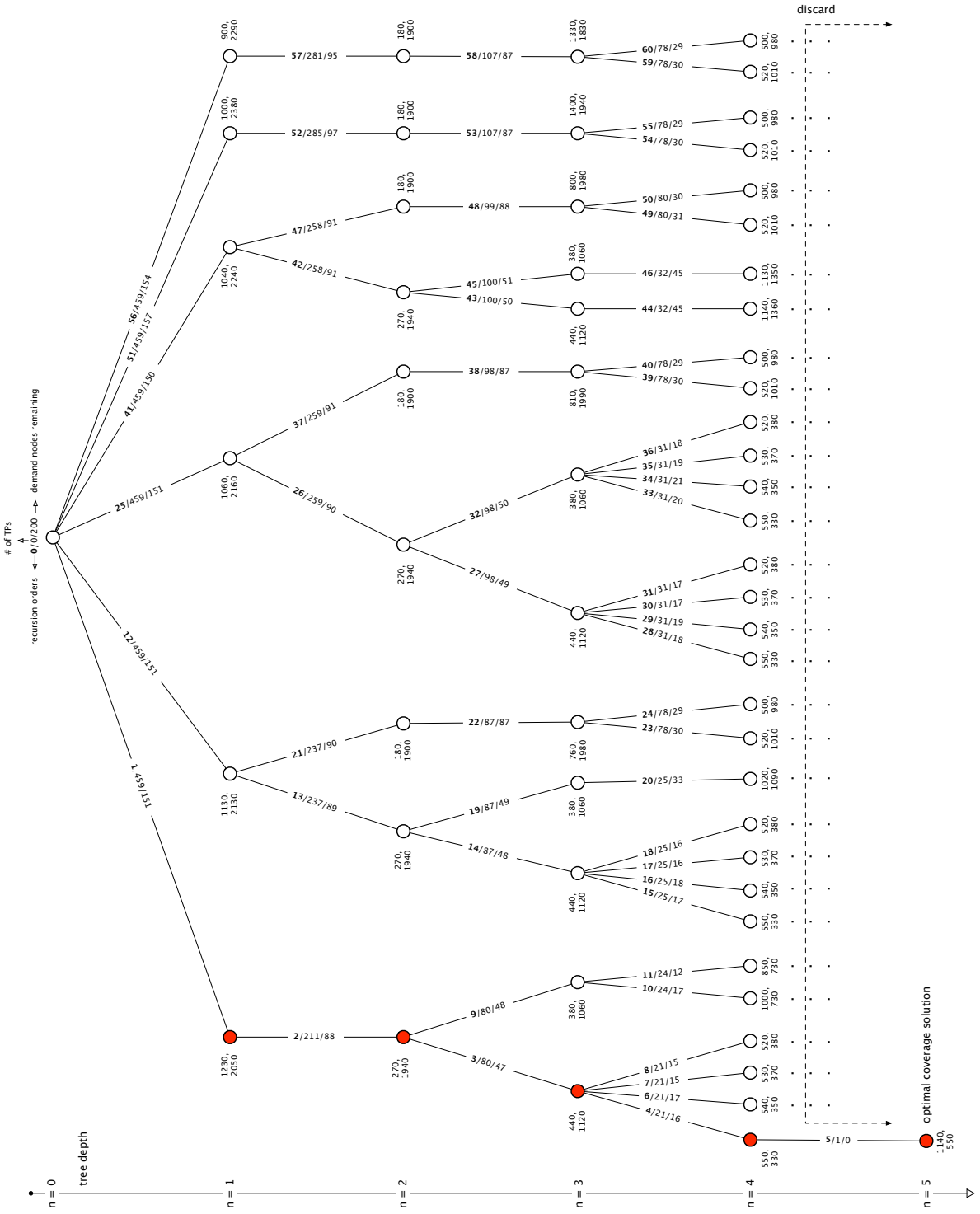

Figure 7.22: Coverage Solution Tree (Clustered Demand Node Example)

The minimum branching algorithm requires 60 recursions to complete. The first 5 recursions (as shown in figure 7.22) yield the optimal solution, therefore $n^* = 5$ BSs are needed to satisfy the coverage requirement. Figure 7.23 shows the resulting coverage solution.

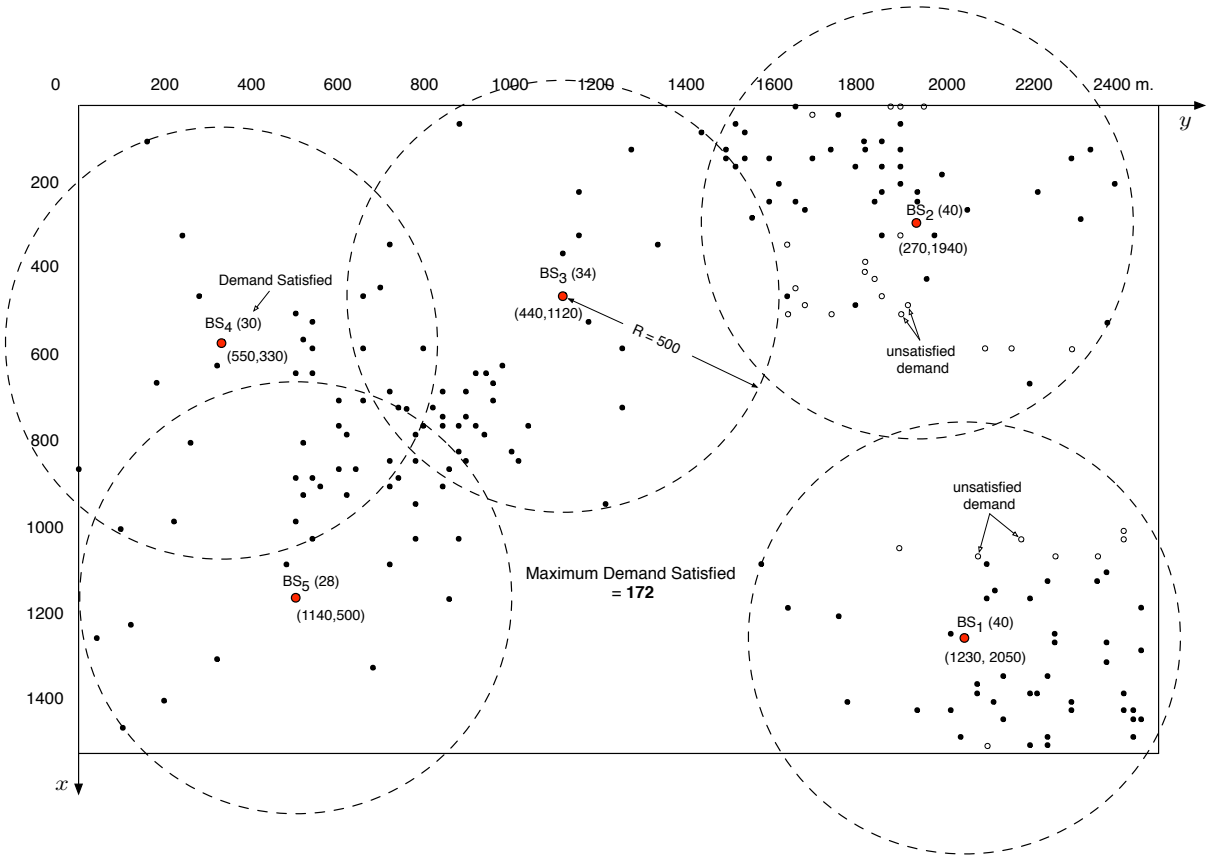

Figure 7.23: Coverage Solution (Clustered Demand Node Example) The minimum branching produces 5 BSs, sufficient satisfied the coverage requirement to all demand nodes. However, two BSs on the top and lower right side cover more demand $> C = 40$. Therefore, the capacity requirement is not met, and the solution is infeasible.

When demand nodes are randomly assigned to $n^* = 5$ BSs, 167 demand units are satisfied. After solving for the optimal assignment, the maximum demand units satisfied then improve to 172. However, it is still insufficient for the total of 200 required. Both BS_1 and BS_2 cover demand modes more than the maximum capacity $C = 40$. The coverage solution shown in figure 7.23 is thus unable to meet the capacity requirement. The TS-based algorithm described in section 6.3 is then employed to maximize the capacity and if it is necessary, search for additional BSs to satisfy the excess demand. In this paper, 100 TS iterations

(forming and selecting new candidate solutions) are executed, which is repeated 10 times (restart) to improve the solution quality through intensification and diversification processes. For $n^+ = n^* = 5$ BSs, with N^* 459 TPs computed earlier from the minimum branching algorithm, TS is able to relocate BSs and achieves the maximum capacity = 194, which is still insufficient. Therefore, one more BSs is needed ($n^+ = n^* + 1 = 6$). Now, with $n^+ = 6$ BSs, TS achieves the maximum capacity = 200, and thus the capacity requirement is satisfied. Figure 7.24 shows the resulting BS placement. The solution in figure 7.24 is

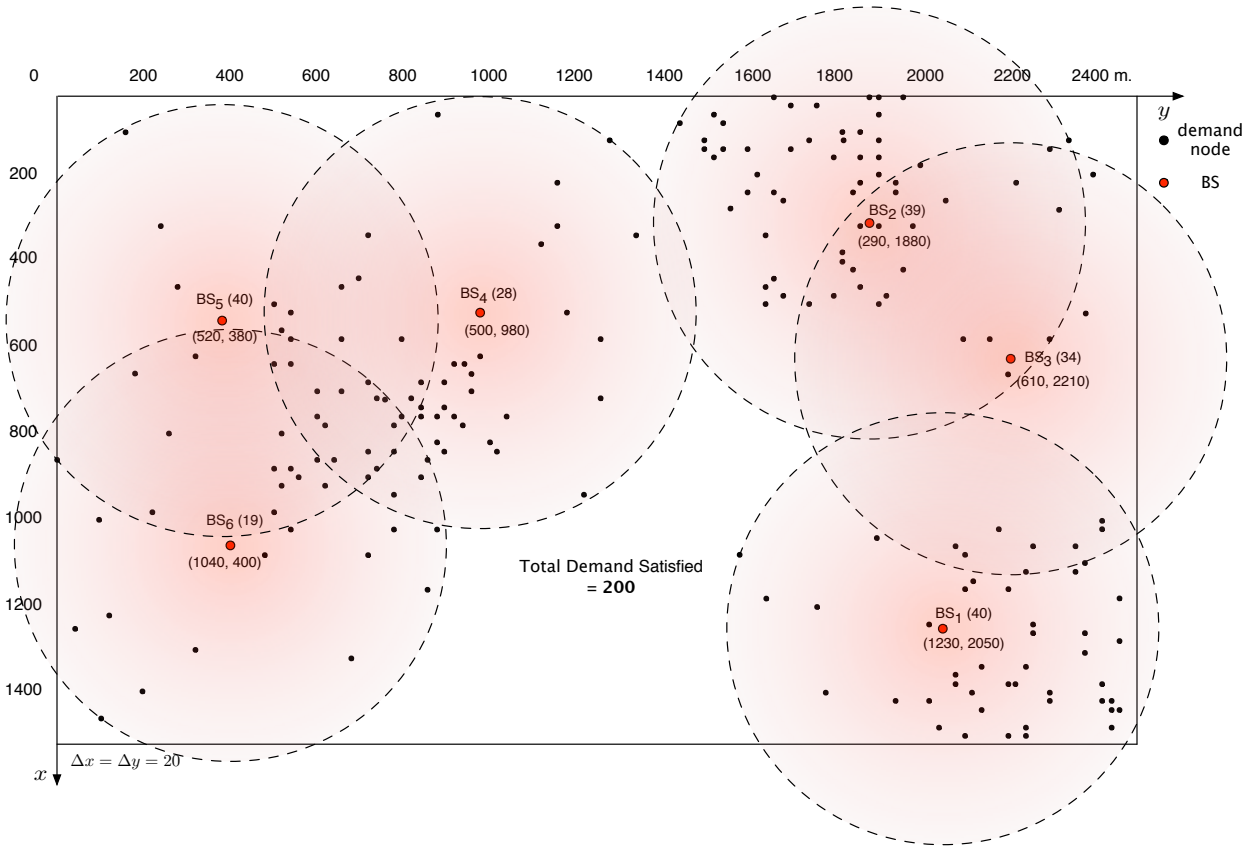

Figure 7.24: A Solution BS placement Solution by TS (Clustered Demand Node Example). A total of $n^+ = 6$ BSs are needed to satisfy the capacity requirement = 200.

computed from the minimum number TPs, otherwise it would be difficult to locate the solution scattered in the search space size of about $N^7 = 37,901^6 \approx 10^{27}$. With $N^* = 459$ TPs, the TS algorithm only searches the space of size $10^3 \times n^+ N^*/5 \approx 5.5 \times 10^5$ (section 6.3.4), the computation requirement is thus reduced by a factor of 10^{21}.

7.2.4 Statistical Analysis

The sweep and merge algorithm reduces the size of search space S by removing grid points that are not part of the optimal solution from N to N^* TPs (where $N^* \ll N$), thereby reducing the computation requirement accordingly. As discussed earlier, N^* can not be determined as a closed-form formula. N^* increases or decreases when the coverage radius shrinks or expands. Figure 7.25 plots N^* as a function of the coverage radius R.

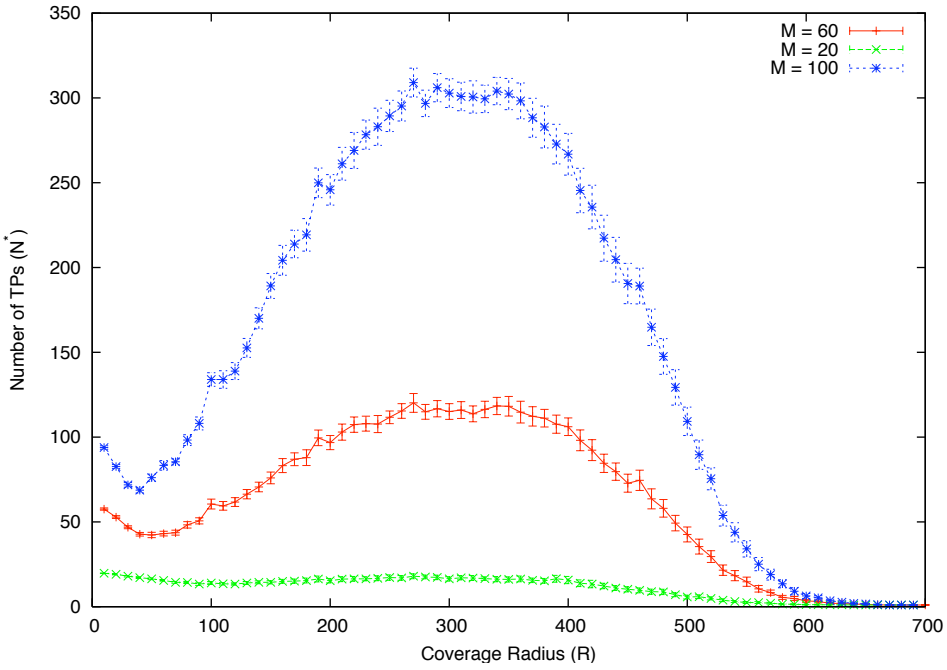

Figure 7.25: Number of TPs (N^*) vs. Coverage Radius (R)

Demand nodes are randomly generated in a square km^2 area. Each curve corresponds to the average number of TPs (with 95% c.i.) produced from the sweep and merge algorithm. When $R \approx 0$, N^* is exactly equal to M, as R increases, N^* gradually increases, reaches the peak point, and decreases — similar to a bell-shaped curve. N^* keeps increasing at small R because it is more likely that two larger sets (demand node set coverage) are distinct than two smaller ones are. However, when R is large, the set coverage is less sensitive to grid point displacement (likely to cover the same set of demand nodes), as a result there will be fewer TPs (redundant grid points are removed). Indeed, when more demand nodes are

involved, the average N^* also increases proportionally (i.e., $M = 60$ vs. $M = 100$). Figure 7.26 plots N^* as a function of demand node density.

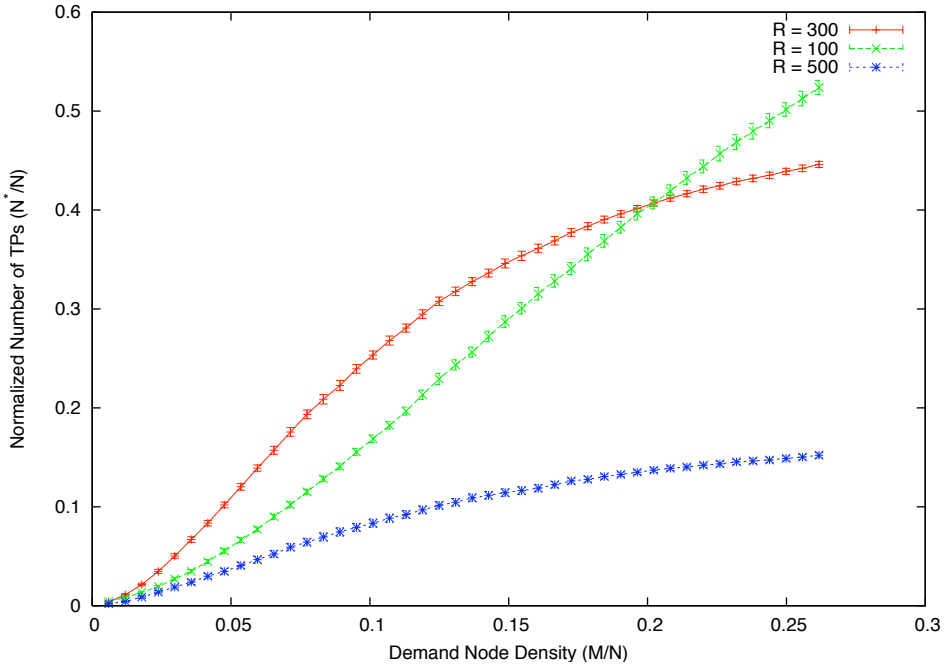

Figure 7.26: Normalized Number of TPs (N^*/N) vs. Demand Node Density (M/N)

As shown in figure 7.26, N^* increases monotonically as the demand node density increases. N^* is normalized by N (total number of grid points), and the density is measured as the ratio between M and N. The three plots ($R = 100, 300, 500$) in figure 7.26 shows that N^* increases more rapidly at low demand node density, but is stable when networks are denser. The statistics are collected from 30 independent tests per each density point. Each test generates a different set of random demand nodes based on the density given. Results from both figure 7.25 figure 7.26 lead to the conclusion that, N^* is a complex function of both R and M. As observed from figure 7.26 above, when M is large, N^* will approach $\approx 0.5N$.

As shown in section 6.1.2, the computational requirement of the minimum branching algorithm can be measured as the total number of recursions needed. Figure 7.27 shows the computational requirement as a function of the coverage radius R — where the total number of recursions decrease when the coverage area πR^2 is larger because fewer BSs n^* are needed.

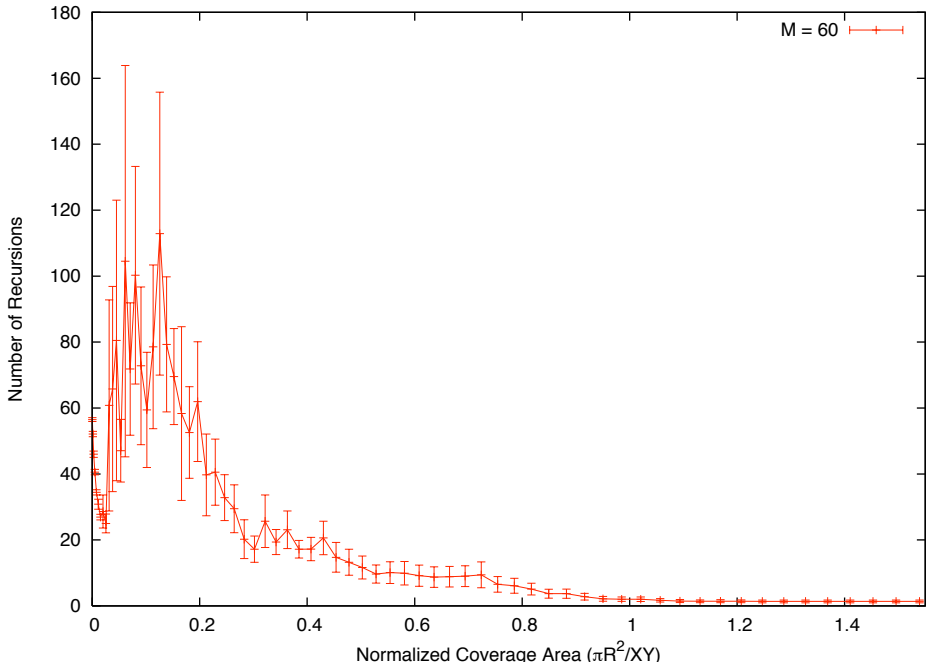

Figure 7.27: Number of Recursions vs. Normalized Coverage Radius ($\pi R^2/XY$)

When the demand node density increases, the variance is also large (large error bar) because more BSs are needed and there are more TPs in each recursion. Due to large error bars shown in figure 7.28, the confidence intervals of 19 out of 20 points are overlapped. The computational requirement, which is now measured as the total number of recursions normalized by S (search space size), is likely to be independent to the demand node density. Employing the minimum branching algorithm reduces the computational requirement by 6 to 14 orders of magnitude.

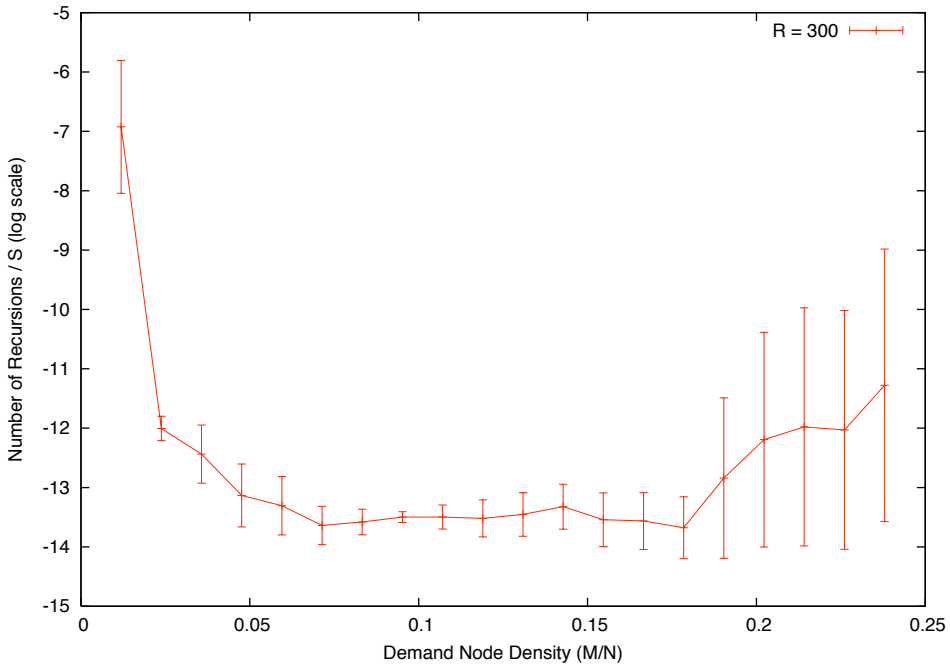

Figure 7.28: Number of Recursions (Normalized) vs. Demand Node Density (M/N)

7.3 DESIGN EXAMPLE: GRAZ, AUSTRIA

We now consider the actual network example to test the design algorithm. Graz, Austria is chosen because it represents a medium size city (similar to Pittsburgh), and more importantly the cell phone usage data is available through "Mobile Landscape | Graz in Real Time", SENSEable City Lab, MIT [55]. Graz is the second largest city in Austria. It has a population around 290,000 and a total area of about 128 km^2. Graz is also a student city, with 6 universities and over 40,000 students [56]. The size, population, and its being the academic center is very similar to the city of Pittsburgh (151 km^2 with 310,000 population [57]). Figure 7.29 shows a map of Graz from Google Maps.

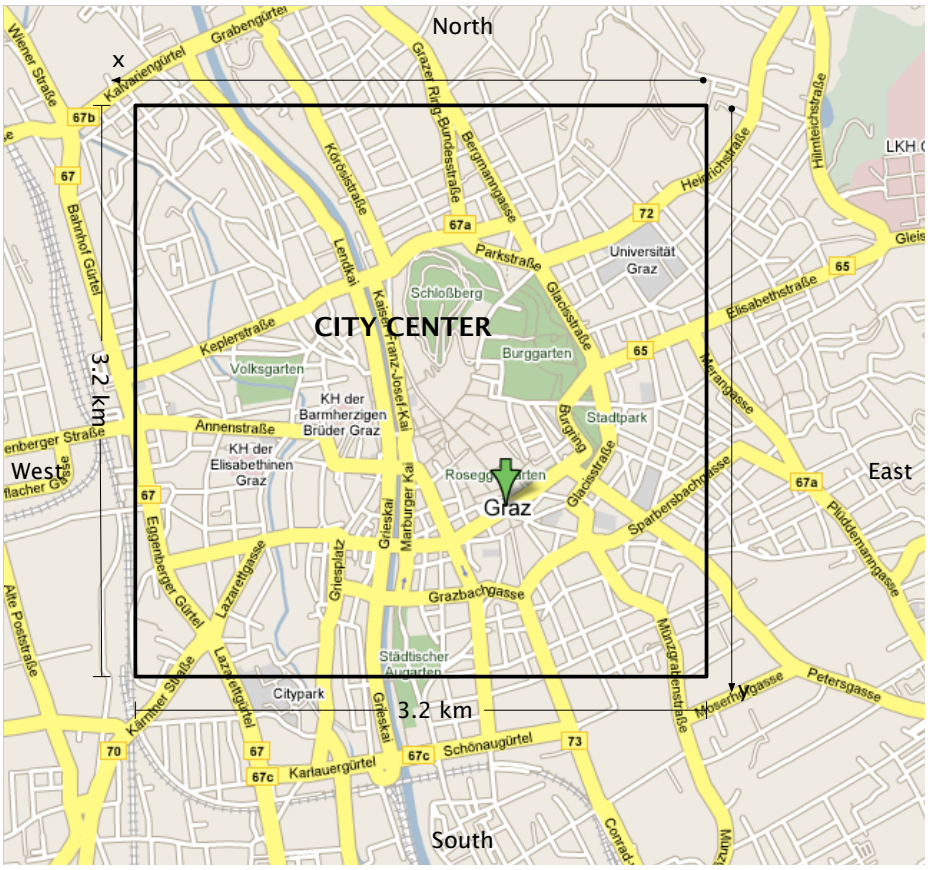

Figure 7.29: A map of Graz, Austria. A 3.2×3.2 km^2 square represents the city center.

7.3.1 Traffic Characterization

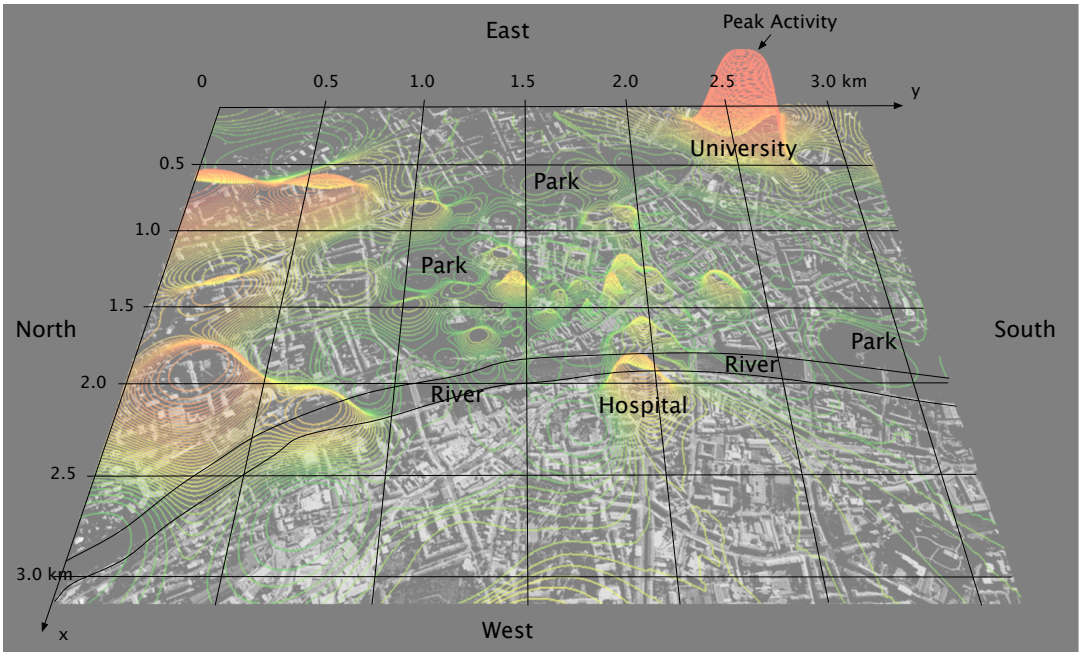

Figure 7.30: Cell Phone Activity in the City Center, Graz

Figure 7.30 illustrates an axonometric view of cell phone activity inside the city center. The activity varies from high to low, where red indicates high activity and green indicates low activity. High usage activity is concentrated at places such as universities and hospitals, while low usage activity occurs in parks, and none for most parts of the river. Location and usage statistics were collected from A1 Mobicom (Austria) by pinging one thousand cell-phones simultaneously as they move through the city [55]. The usage activity shown in figure 7.30 is, however, dimensionless: it differentiates between high and low usage, but no actual traffic volume or intensity is given. To compute the traffic intensity, we need to know or assume parameters such as population density and proportion of users active during the busy hour. The population density in Graz is approximately 2,000/km^2 [56]. If 20% of users are assumed to be active during the busy hour, thus the highest traffic intensity is

$$= 0.2 * 2,000 = 400 \text{ Erlangs/km}^2$$

Matchng the 400 Erlangs/km^2 to the peak activity in figure 7.30, the actual traffic intensity for the entire city center can be mapped proportionally. Figure 7.31 shows the approximated traffic intensity.

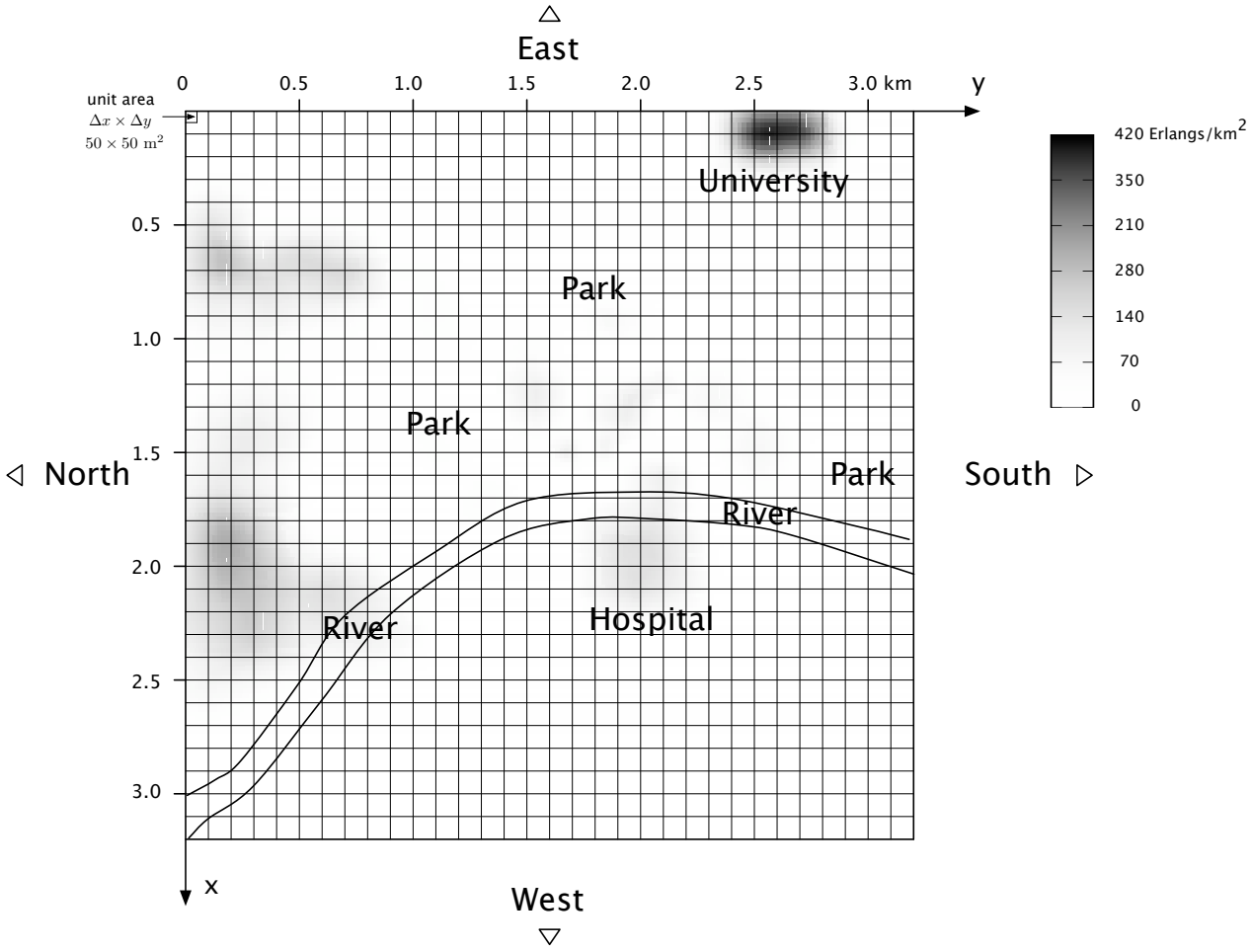

Figure 7.31: Approximated Traffic Intensity in Erlangs/km^2

The approximated traffic intensity is created by assuming Guassian distribution of user movement [53]. Areas with higher usage activity shown in figure 7.30 (i.e., university and hospital) are populated with more users to simulate higher traffic intensity. The total traffic is 273.38 Erlangs. The traffic load in each unit area ($\Delta x \times \Delta y = 50 \times 50$ m^2) can be computed, and the demand nodes are generated accordingly.

7.3.2 Demand Node Generation

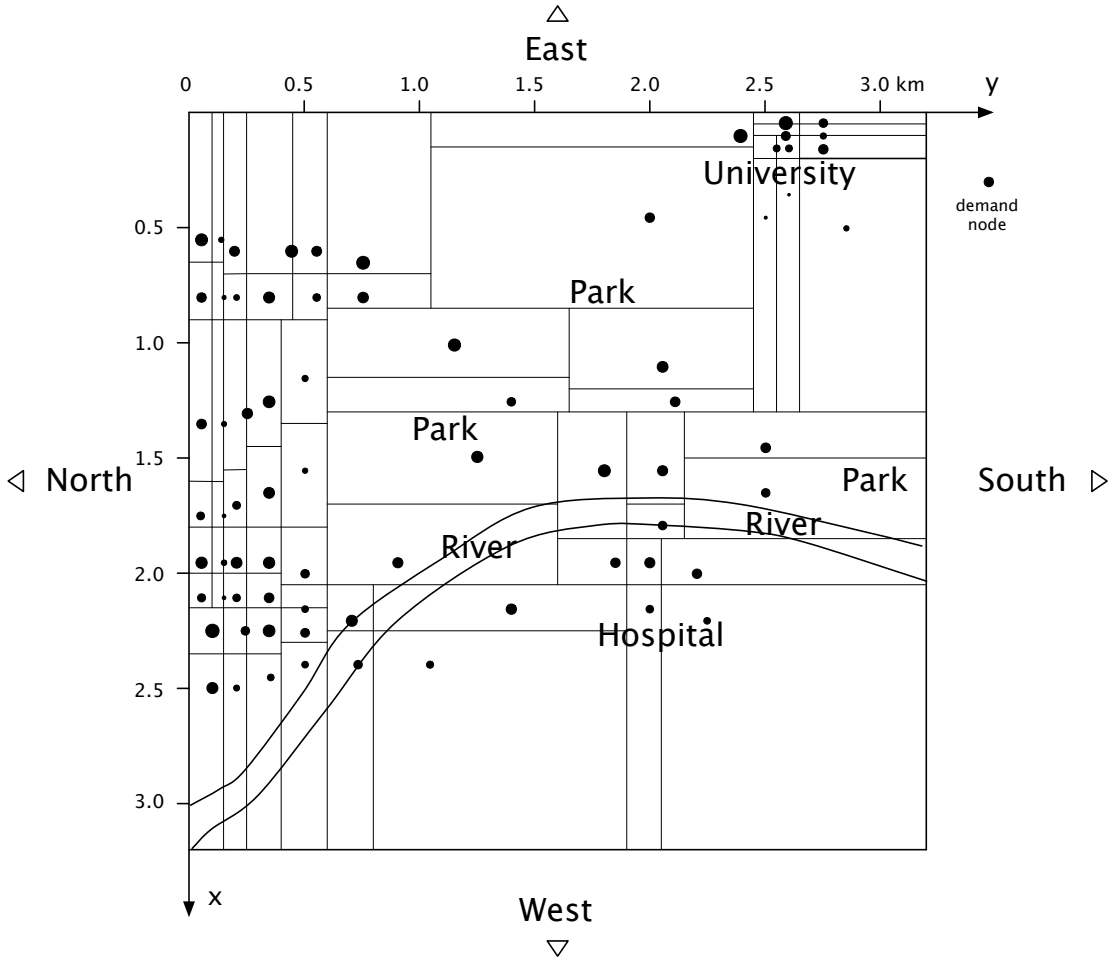

Figure 7.32: Demand node in each tessellation represents cell phone traffic in Graz, ranging from 1 to Δq = 6 Erlangs. The larger the demand node, the larger traffic it carries.

The demand nodes in shown figure 7.32 is recursively generated by bisectioning the design area into two halves (i.e, rectangles) with equal amount of traffic. The bisectioning turns 90° for each recursion until the traffic in every tessellation piece falls below a threshold $\Delta q = 6$, which is chosen so that the smallest demand node carries traffic more than one Erlang, or equivalently at least one user. There are a total of 72 demand nodes generated, with traffic ranging from 1 to 6 Erlangs (limitations are discussed in chapter 8). The sweep and merge algorithm is then applied to compute TPs, described next.

7.3.3 Coverage Radius (CDMA2000 EV-DO)

Before computing TPs, the coverage radius R must be determined. As discussed earlier, the coverage radius R is a function of the path loss and the required SNR at the receiver. The required SNR may vary, depending on which technology is implemented. In this case, the CDMA2000 EV-DO technology is chosen. CDMA2000 EV-DO is among the leading technologies for 3G networks, supporting burst data rate upto 153.6 kbps on the reverse link and 2,457.6 kbps on the forward link [11]. For the coverage requirement, the constraint is on the reverse link because the maximum mobile transmit power is limited to 23 dBm, compared to 43 dBm available to the BS [11]. As shown in the Appendix, the minimum SNR required to guarantee the maximum date rate 153.6 kbps at 0.7 loading factor must be greater than -98 dBm. If the BS antenna gain is 15 dB [58], the maximum path loss $P_L = 23 - (-98) + 15 = 136$ dB (symmetric in both directions). Using the COST 231 path loss formula described in chapter 3,

$$P_L = 46.3 + 33.9 \log f_c - 13.82 \log h_b + (44.9 - 6.55 \log h_b) \log R + 3$$

where the carrier frequency $f_c = 2,000$ MHz, and the BS antenna height $h_b = 32$ m [58], then

$$136 = 46.3 + 33.9 \log 2000 + -13.82 \log 32 + (44.9 - 6.55 \log 32) \log R + 3$$

and the coverage radius $R \approx 750$.

7.3.4 BS Capacity (CDMA2000 EV-DO)

The BS capacity C, is the maximum number of users (Erlangs) that can be supported simultaneously. Assuming that the upload traffic is negligible compared to the download traffic [59], therefore only the forward link capacity requirement is considered. For data services, C is a function of the average forward link throughput R_d, the target user-percieved data rate r, the distribution of packet size S_M, and the mean reading time D_{pc} (the average time users take to process data after receiving the packet). Several field experiments and simulation studies showed that, the EV-DO forward link throughput R_d ranges between 500 to 1225 kbps [52], [60], [61]. Dominant data applications for EV-DO networks are such as

HTTP (including WAP), FTP [58], [62]. The mean packet size S_M for HTTP is 55 kB vs 2 MB for FTP applications, and the mean reading time is 30 and 180s respectively [58]. The service time is also random because the forward link throughput fluctuates due to fading. Using the $m/m/1$ queuing model (only one server because the BS transmits to one user at a time), the BS capacity C (maximum number of users) is computed as [52]

$$C = \frac{R_d D_{pc}}{S_M} \left[\frac{S_M/r - 0.5 - S_M/R_d}{S_M/r - 0.5} \right] \tag{7.7}$$

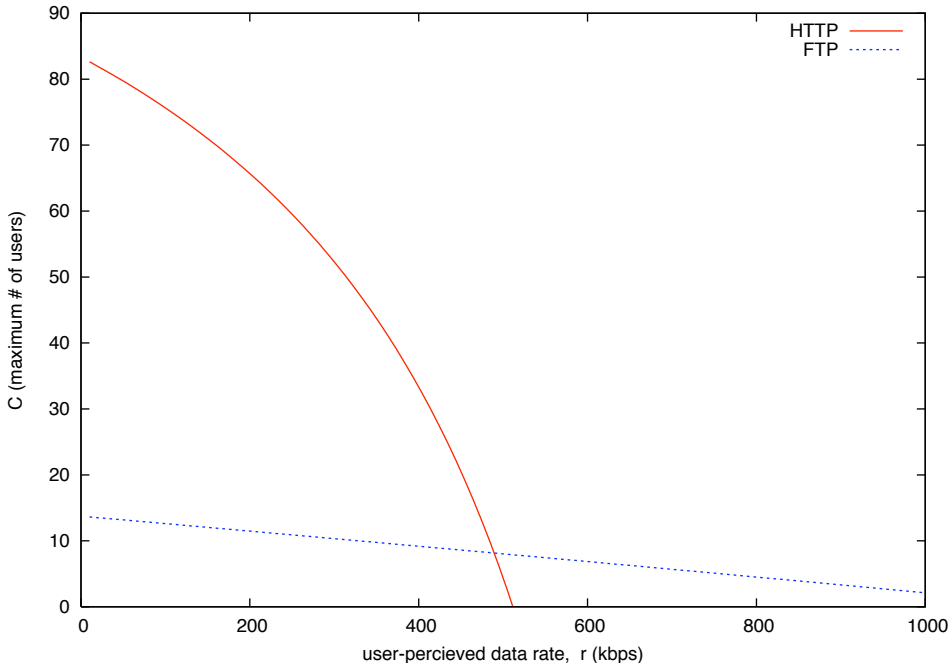

Figure 7.33: Maximum # of Users vs Data Rate for HTTP and FTP applications at 1225 kbps average forward link throughput. The mean reading time D_{pc} is 30 and 180s for HTTP and FTP respectively.

As shown figure 7.33, the EV-DO BS can support upto 80 HTTP users or 12 FTP users [52]. Using (7.7), if the required user-perceived data rate r is 64 kbps for HTTP applications [62],

$$C_{\text{http}} = \frac{1225 * 30}{55 * 8} \left[\frac{55 * 8/64 - 0.5 - 55 * 8/1225}{55 * 8/64 - 0.5} \right] \approx 79$$

about 79 HTTP users can be supported (at $R_d = 1225$ kbps), while if FTP applications require $r = 256$ kbps (equivalent to basic ADSL service),

$$C_{\text{ftp}} = \frac{1225 * 180}{2000 * 8} \left[\frac{2000 * 8/256 - 0.5 - 2000 * 8/1225}{2000 * 8/256 - 0.5} \right] \approx 11$$

about 11 FTP users can be supported. Suppose that approximately 70% of users are HTTP users (including WAP) and the remaining 30% are FTP users [58], [62], then the BS capacity is averaged to be

$$C = 0.7C_{\text{http}} + 0.3C_{\text{ftp}} = 58$$

The BS capacity C must be recomputed when either the traffic proportion or the required data rate changes. As an example, if the required data rate for HTTP users increases to 256 kbps (equal to FTP), then C_{http} reduces to 59 users. With the same traffic proportion (70% HTTP / 30% FTP), the average BS capacity C decreases from 58 to 44 users. However, for the Graz design case, HTTP data rate is fixed to 64 kbps, and $C = 58$ is used.

7.3.5 TP Reduction

Demand nodes shown in figure 7.32 are separated no closer than $\Delta x = \Delta y = 50$ m, equal to the dimension of unit area used in the traffic quantization process. According to (5.15), the required TP grid spacing Δg must be no larger than half the demand node grid spacing in order to guarantee the optimal solution.

$$\Delta g = \Delta x/2 = 25 \text{ m}$$

The design area is a rectangular with size X = 3200 and Y = 3200 m, therefore the total number of initial grid points required are

$$N = \left\lfloor \frac{X = 3200}{25} + 1 \right\rfloor \left\lfloor \frac{Y = 3200}{25} + 1 \right\rfloor = 129^2 = 16,441$$

The sweep and merge algorithm is applied to reduce N. There are 72 demand nodes with the total traffic of 273.38 Erlangs. The smallest demand node carries 1.09 Erlangs and the largest demand node has 5.81 Erlangs. Using the coverage radius $R = 750$ m derived in the previous section, the sweep and merge algorithm produces $N^* = 99$ TPs as shown in figure 7.34.

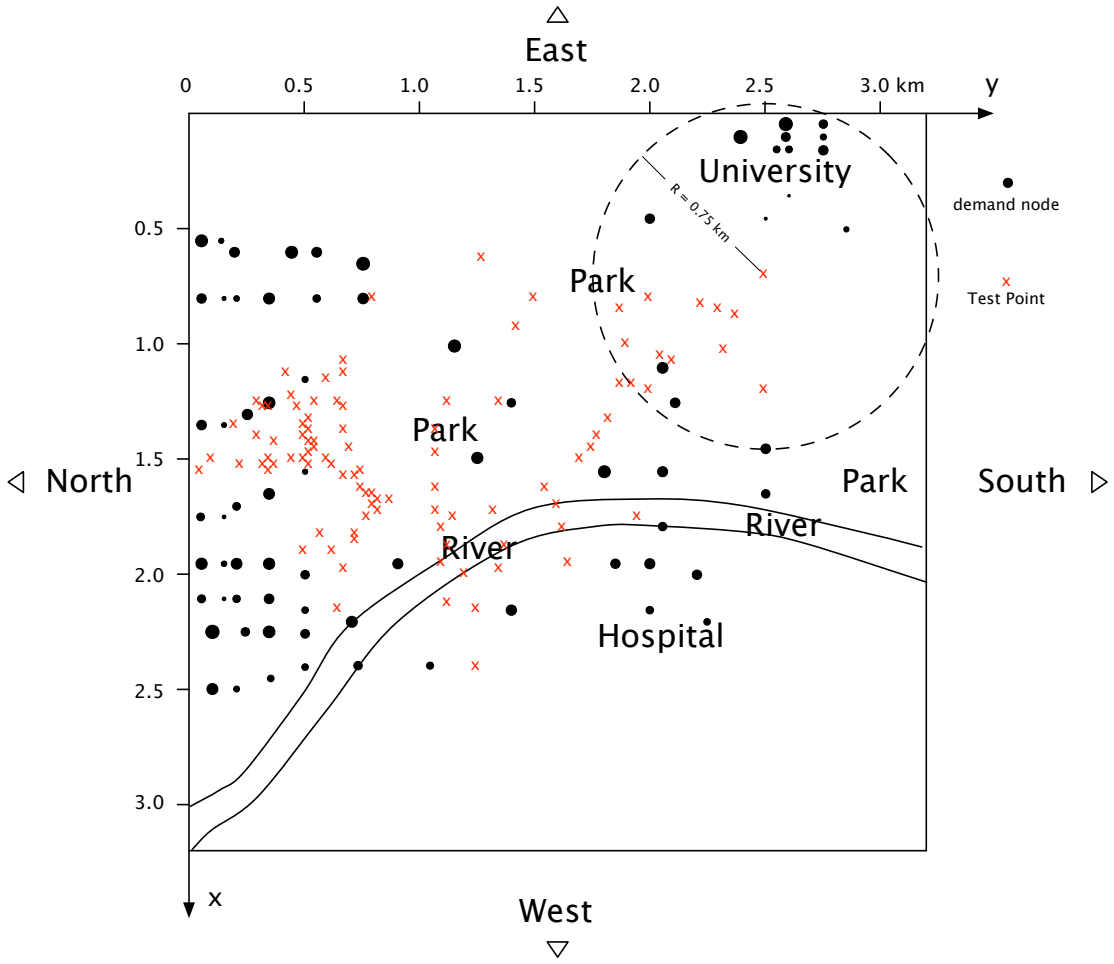

Figure 7.34: TPs resulted from the sweep and merge algorithm (Graz Example)

Figure 7.34 shows the resulting TPs. The sweep and merge algorithm reduces the total number of TPs to $N^* = 99$ TPs, compared to $N = 16,441$ initial grid points. Most TPs appear among demand nodes, while few TPs are present below the river and above the parks. These TPs are used to compute the coverage solution described next.

7.3.6 Minimum Branching

The first recursion of the minimum branching algorithm selects TP at (0.7, 2.5) km as the first BS in the solution tree as shown in figure 7.35.

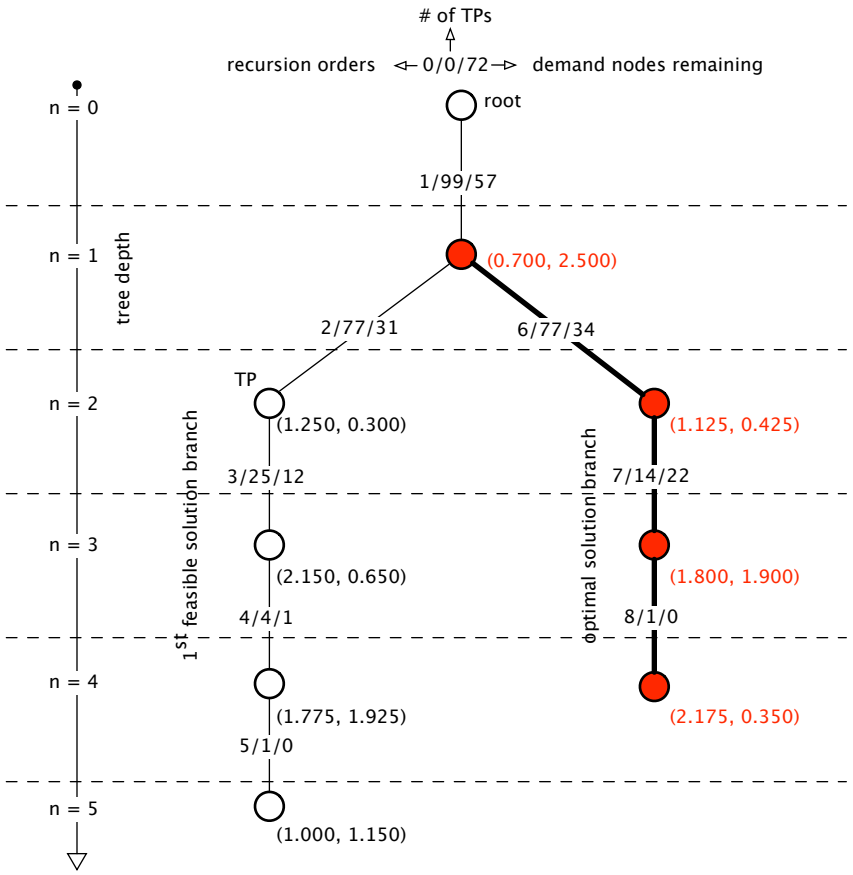

Figure 7.35: Minimum Branching Solution Tree. Two feasible solutions: the first one requires 5 BSs, while the second one, which is the optimal solution, needs $n^* = 4$ BSs to cover all demand nodes.

15 demand nodes within range of BS_1 are removed. Thus, the network now consists of $72 - 15 = 57$ uncovered demand nodes, and a new set of TPs must be recomputed. Applying the sweep and merge algorithm to the network of 57 uncovered demand nodes then produces a new set of 77 TPs. Out of 77 new TPs, 2 TPs at (1.250, 0.300) and (1.125, 0.425) km are selected to create a new branch of solution. The TP at (1.250, 0.300) leads to the first feasible solution requiring 5 BSs, whereas the TP at (1.125, 0.425) forms the optimal solution branch thereafter, requiring one fewer BS to satisfy the coverage requirement.

7.3.7 Capacity Requirement Validation

The coverage solution resulted from the minimum branching algorithm is shown in figure 7.36. Using CPLEX to obtain the optimal demand node assignment, the coverage solution

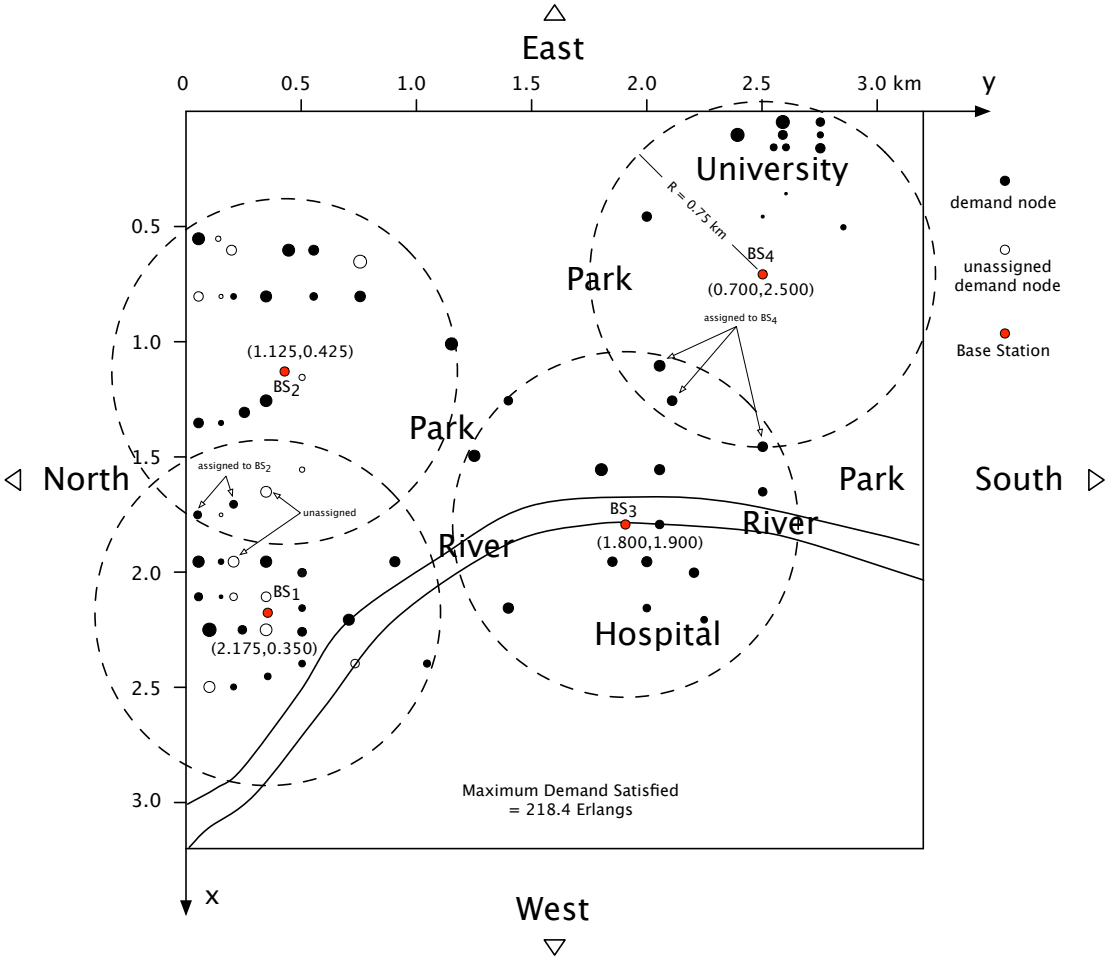

Figure 7.36: Coverage Solution: $n^* = 4$ BSs are needed to satisfy the coverage requirement. However, since BS_1 and BS_2 cover demand nodes with traffic more than the maximum capacity $C = 58$ Erlangs, therefore the excess demand nodes are not assigned.

with $n^* = 4$ BSs in figure 7.36 is able to support 218.4 Erlangs, still insufficient to support the total demand 273.38 Erlangs. Both BS_1 and BS_2 are among a large group of demand nodes. BS_1 alone covers demand nodes with the total traffic close to 99 Erlangs, more than its maximum capacity $C = 58$ Erlangs, thus some demand nodes are left unassigned and the capacity requirement is not met. In fact, $n^* = 4$ BSs will never be sufficient to satisfy

the capacity requirement, since $4 \times 58 = 232 \leq$ the total 273.38 Erlangs. The TS-based algorithm (described in section 6.3) is then employed to solve for a new solution with one additional BS, such that $n^+ = n^* + 1 = 5$ BSs are to be placed at $N^* = 99$ TPs computed earlier from the sweep and merge algorithm in figure 7.34. TS manages to locate $n^+ = 5$ BSs so that the network can support the total capacity requirement = 273.38 Erlangs, and all the BSs are assigned with demand nodes fewer than the maximum capacity $C = 58$ Erlangs. Figure 7.37 shows the resulting BS placement from the TS-based algorithm.

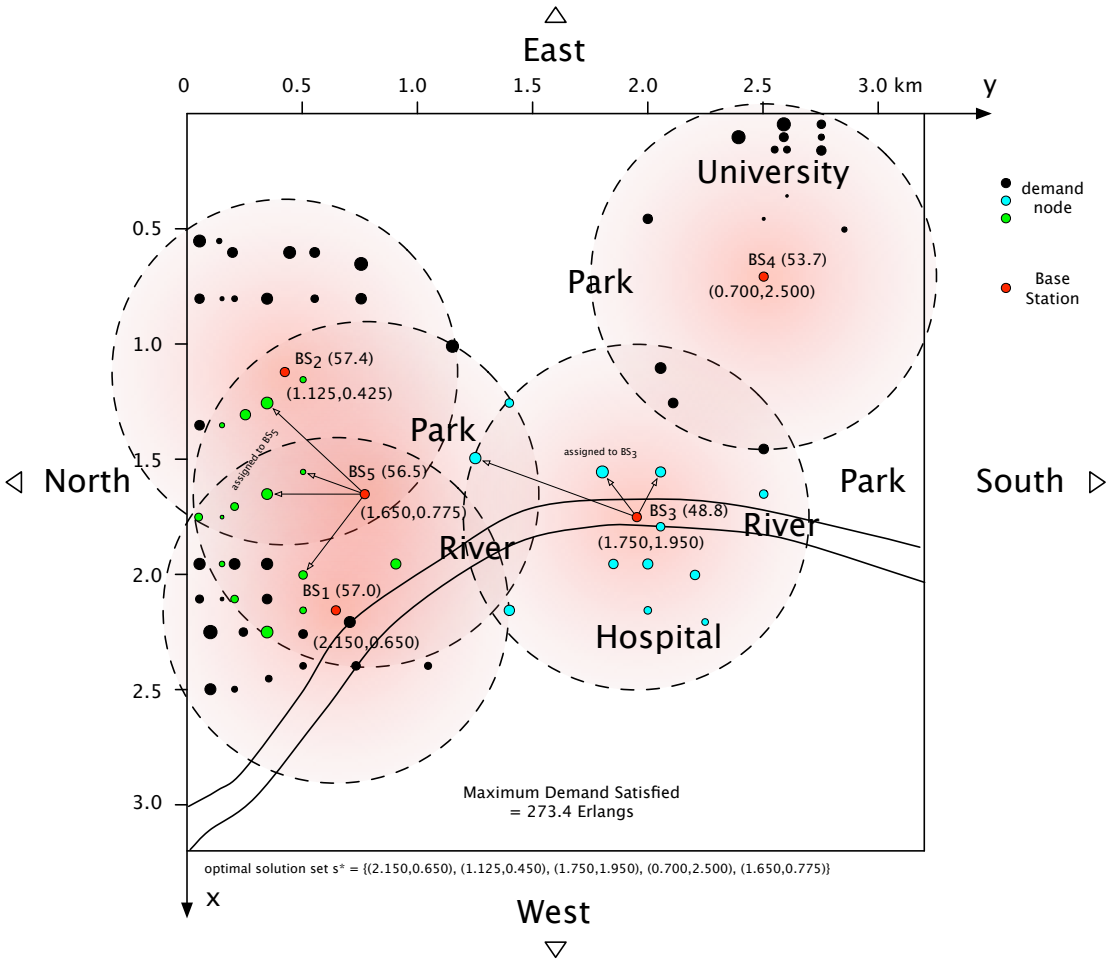

Figure 7.37: Optimal Solution: 5 BSs are needed to satisfy both the coverage and capacity requirements.

The optimal demand node assignment is applied here. Traffic is distributed more evenly among BSs with BS_3 cover the minimum traffic with 48.8 Erlangs. The new solution with $n^+ = 5$ BSs satisfies both the coverage and capacity requirements, and therefore is the

optimal solution. The TS algorithm needs to evaluate $S = 1,000 \times 0.2n^{+}N^{*} = 9.9 \times 10^{4}$ possible candidate BS placement (section 6.3.4), as opposed to $S = N^{n^{+}} = 16,441^{5} \approx 10^{21}$ as in the exhaustive search case. Therefore, the design algorithm with TP reduction reduces the computational requirement substantially by a factor 10^{16}. The resulting design supports 273.38 Erlangs of cell phone traffic, assuming that 70% are HTTP users requiring 64 kbps data rate and the remaining 30% FTP users requires 256 kbps data rate.

7.3.8 Expanded Network

Suppose that additional traffic is created (on top of the existing network) from the new university complex on the outskirts of the main university and from the extended hospital facilities just below the river as shown in figure 7.38

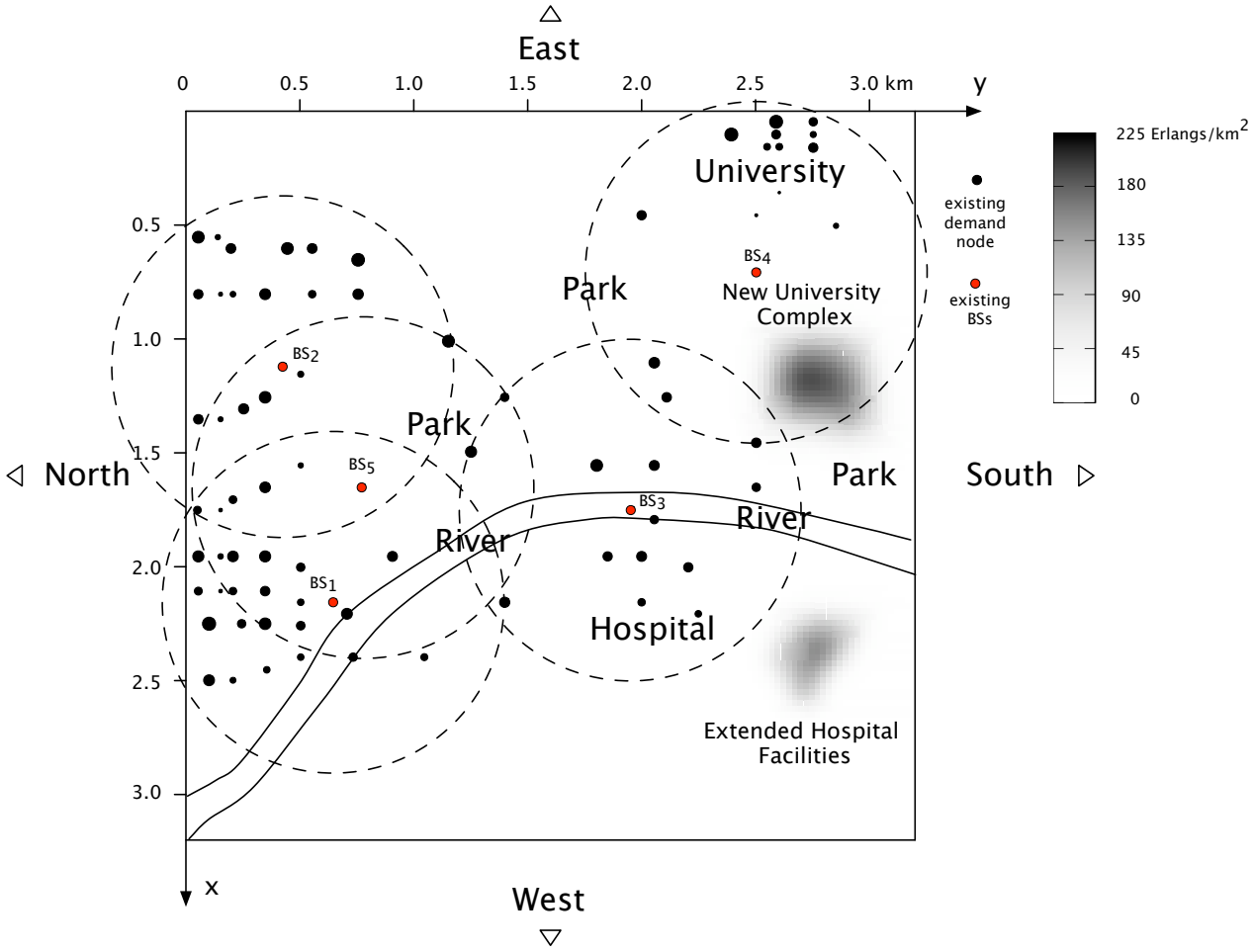

Figure 7.38: Additional Traffic (Graz Example - Expanded Network). There are two sources of additional traffic: one from the new university complex and the other from the extended hospital facilities.

The additional traffic from both the new university complex and the extended hospital facilities is 49.76 Erlangs. Combined with the traffic from existing demand nodes, the total network traffic now increases to 323.14 Erlangs. Clearly, with 5 existing BSs ($C = 58$ Erlangs), the maximum traffic that can be supported is $= 58 * 5 = 290 <$ the new 323.14

Erlangs required. Thus, the existing network with 5 BSs is not sufficient to support the increased traffic. There are two options to design a new network: one is to redesign the network from the scratch again without existing BSs (greenfield design), while the alternative is to locate fewest more BSs on top of the existing BSs to support the excess capacity requirements. The greenfield design is illustrated first.

7.3.8.1 Expanded Network - Greenfield Design Demand nodes of the additional traffic are computed. Figure 7.39 shows the resulting new demand nodes.

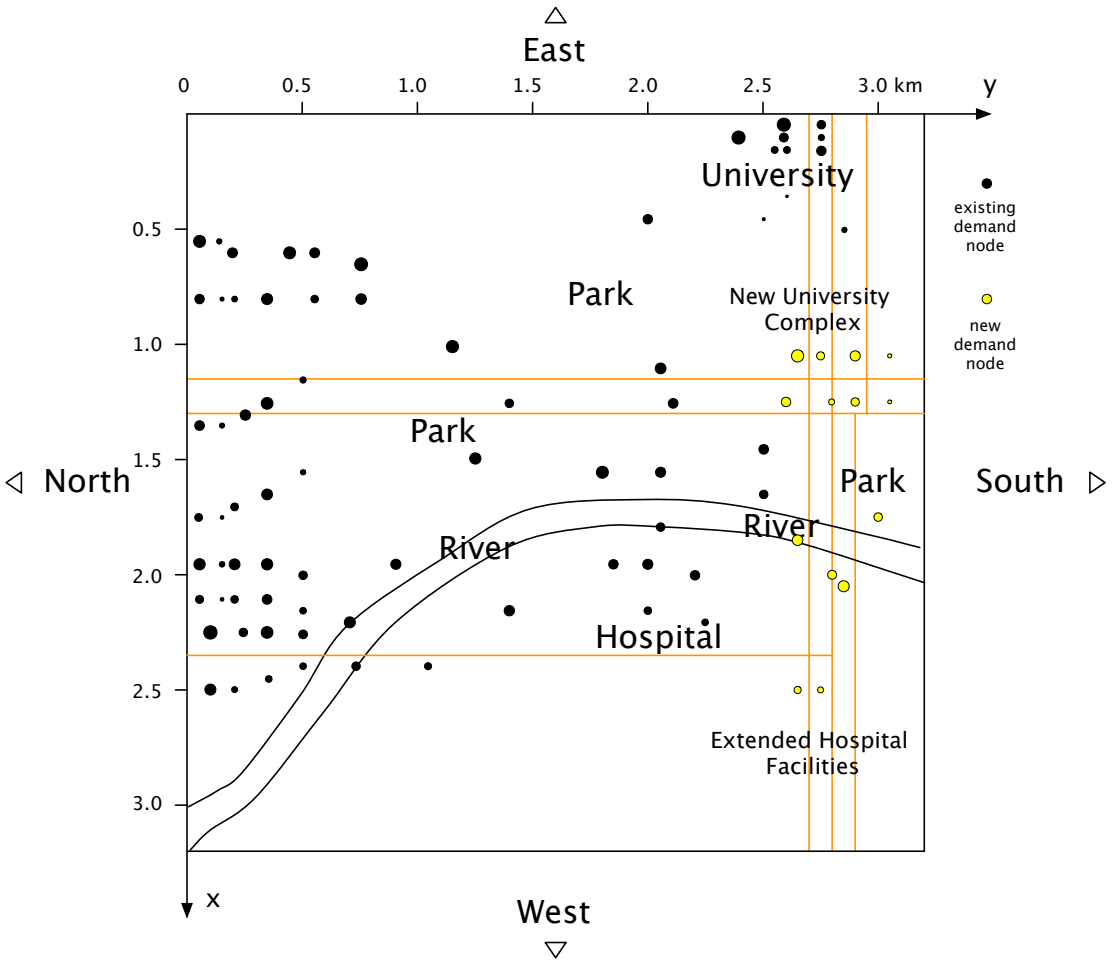

Figure 7.39: New Demand Nodes (Graz Example - Expanded Network). 14 new demand nodes (in addition to the existing demand nodes) is generated by the recursive bisectioning process (the bisectioning line is in orange). The new demand nodes are shown in yellow, while the existing demand nodes are in black.

The recursive bisectioning process (described in section 5.1) creates 14 new demand nodes. With the previous 72 demand nodes from the existing network, the total number of demand nodes in the expanded network are now 86. Since the 14 new demand nodes are placed at different locations, using the previous set of TPs (as shown in figure 7.34) of the existing network may not guarantee the optimal solution. To ensure the optimal solution for the expanded network, a new set of TPs must be recomputed. Figure 7.40 shows the new 122 TPs resulted from the sweep and merge algorithm. Out of the new 122 TPs, 96 TPs appear

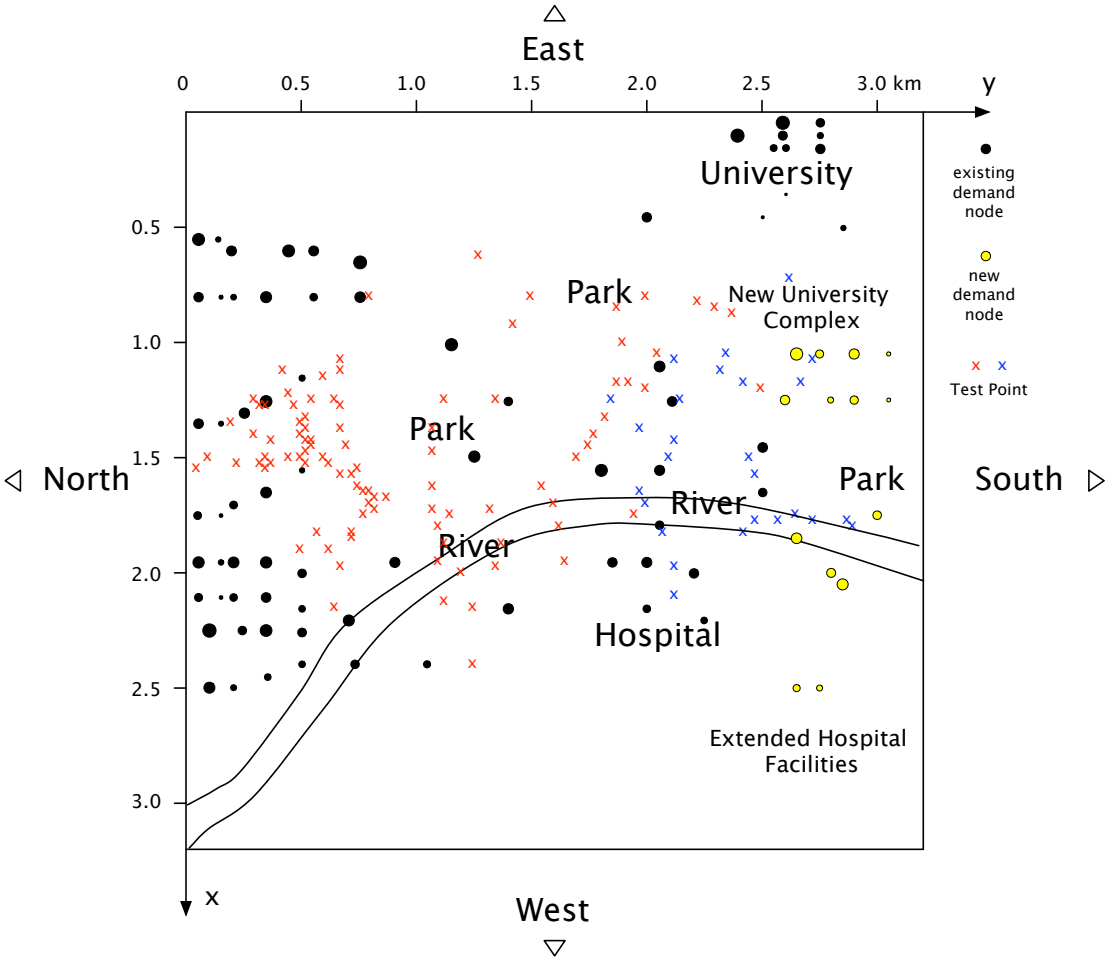

Figure 7.40: New Set of TPs (Graz Example - Expanded Network). 122 new TPs are recomputed from the sweep and merge algorithm.

at the previous locations as computed from the existing network (as shown in x), whereas the other 26 TPs are created at different locations (as shown in x) to support the new demand

nodes added. The next step is to compute for the coverage solution, in which the minimum branching is applied. The corresponding solution tree for the expanded network is shown in figure 7.41.

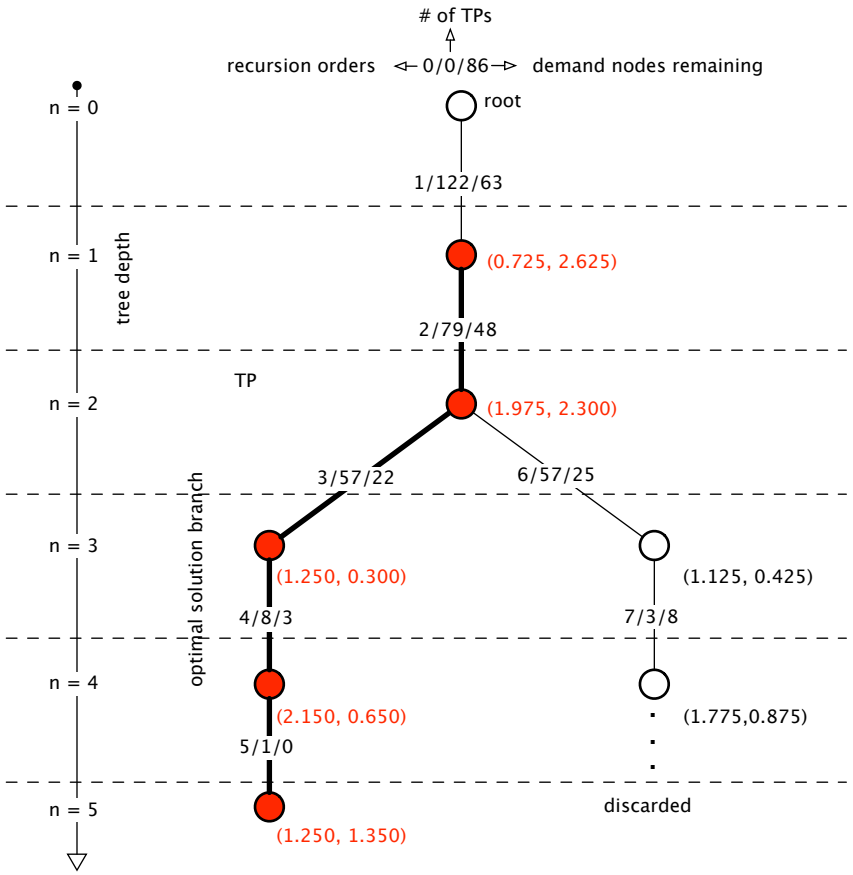

Figure 7.41: Coverage Solution Tree (Graz Example - Expanded Network). The first 5 recursions are the optimal solution branch, requiring 5 BSs (nodes) to satisfy the coverage requirement.

The minimum branching algorithm requires a total of 7 recursions to complete. The first 5 recursions already result in the optimal solution. At the seventh recursion (4 BSs are already employed, and there are still 8 demand nodes uncovered), therefore the coverage solution belong to this branch will not be better than the solution found from the first five recursions, thus the nodes that come after are discarded, and the algorithm stops. The expanded network requires at the minimum $n^* = 5$ BSs to provide coverage to all the demand nodes. Figure 7.42 shows the resulting BS placement of the coverage solution.

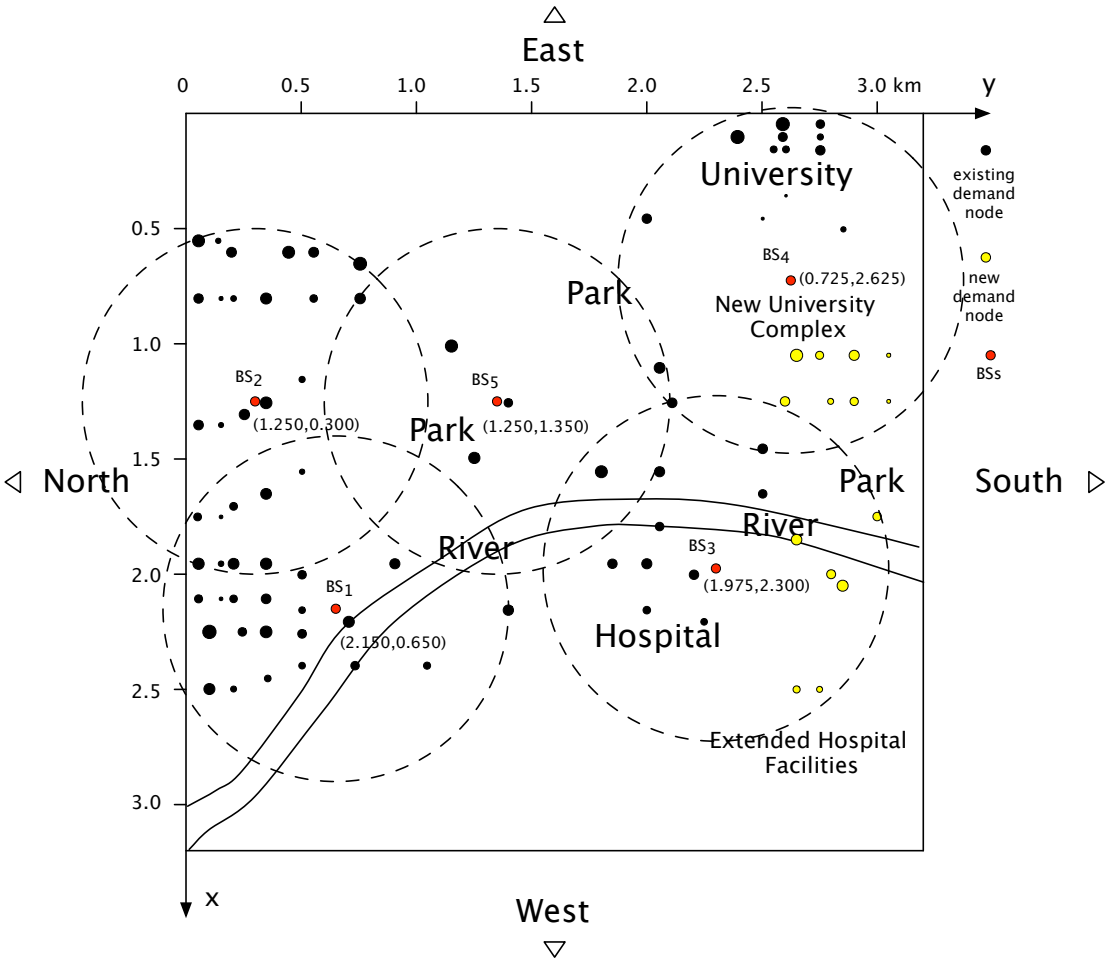

Figure 7.42: New Coverage Solution (Graz Example - Expanded Network) At minimum $n^* = 5$ BSs are needed to satisfy the coverage requirement.

Since the maximum BS capacity $C = 58$ — irrespective of the demand node assignment, $n^* = 5$ can only support upto $= 5 * 58 = 290$ Erlangs — which are not sufficient for the total demand required = 323.14 Erlangs. Thus, more BSs are needed ($n^+ > n^*$). The TS-based algorithm (described in section 6.3) is then employed to compute for the new optimal assignment of BSs. With the total 323.14 Erlangs, the minimum number of BSs needed are $= \lceil 323.14/58 \rceil = 6$. Given $n^+ = n^* + 1 = 6$ BSs and $N^* = 122$ candidate TPs computed from the sweep and merge algorithm (as shown in figure 7.40), the TS-based algorithm rearranges all the BS placement and (together with the optimal demand node assignment - section 6.2) searches for the set that will maximize the total network capacity. The resulting

BS placement form the TS-based algorithm is able to support the total demand = 323.14 Erlangs, meeting the capacity requirement of the expanded network. Therefore, the BS placement solution from TS is the optimal solution of the design (requires the minimum number of BSs $n^+ = 6$ to satisfy both coverage and capacity requirements). Figure 7.43 shows the optimal BS placement computed from TS.

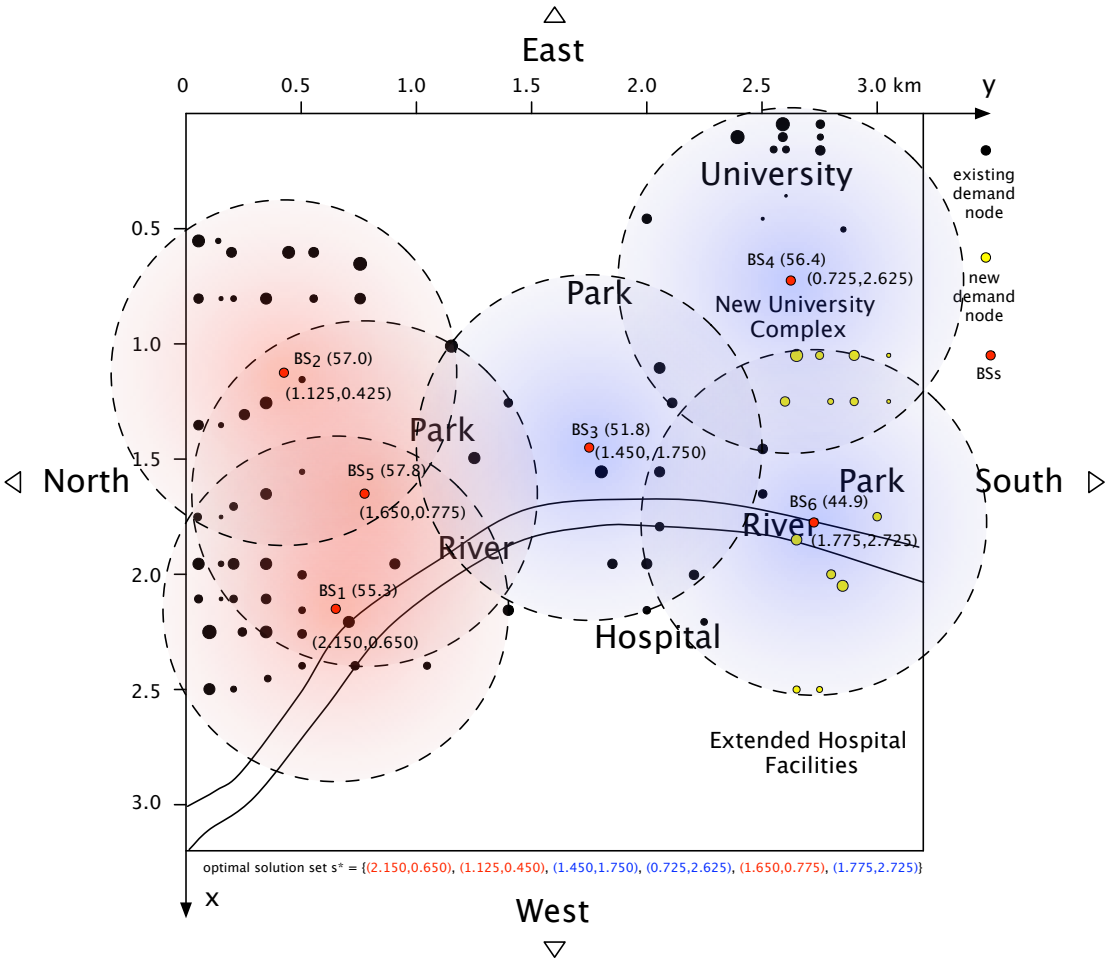

Figure 7.43: New Optimal Solution (Graz Example - Expanded Network) At least $n^+ = 6$ BSs are needed to supports the total demand of 323.14 Erlangs.

Compared with the previous design of the existing network (figure 7.37), three BSs which are BS_1, BS_2 and BS_5 remain at the same locations (the coverage is shown in red). However, the locations of BS_3 and BS_4 change, and the new BS_6 is added to accommodate the increased demand (the coverage is shown in blue). Note that, the BS_4 and BS_6 are placed at the new

TPs at (0.725, 2.625) of (1.775, 2.725) km (as shown in x in figure 7.40). In each iteration of the TS-based algorithm, $0.2n^+N^*$ candidate neighbor solutions are evaluated (section 6.3.4). In this case, 1,000 iterations are used, thus the total number of evaluations or the search space size $S = 1,000 \times 0.2n^+N^* \approx 1.46 \times 10^5$. Alternatively, if the initial number of grid points $N = 16,441$ are used instead (TP reduction not applied), together if the exhaustive search is applied, the search space S could be as large as $S \approx 16,441^6 \approx 2 \times 10^{25}$. Thus, both the TP reduction and the TS-based algorithms helps reduce the computation requirement by at least a factor $2.5 \times 10^{25}/1.46 \times 10^5 \approx 10^{20}$.

7.3.8.2 Expanded Network - Incremental Design As discussed earlier, the second approach to design the new network (alternative to the greenfield design described in the previous section) to meet the increased demand is to employ the fewest more BSs on top of the existing BSs (BS_1 to BS_5 in figures 7.37 and 7.38) — the incremental design. The first step of the incremental design approach, again is to locate the minimum number of candidate TPs. All 72 demand nodes of the existing network are ignored, and the TPs are only computed for the 14 newly added demand nodes (for all the yellow dots shown in figure 7.39) responsible for the additional traffic from both the new university complex and the extended hospital facilities. Then, together with the existing BSs (BS_1 to BS_5 in figures 7.37 and 7.38), additional BSs must be located from the candidate TPs (computed from the new 14 demand nodes) to satisfy both coverage and capacity requirements of the entire network. Note that, by adopting the assumption above since the solution is already fixed with 5 BSs which are BS_1 to BS_5 located respectively at {(2.150, 0.650), (1.125, 0.450), (1.750, 1.950), (0.700, 2.500), (1.650, 0.775)} km, the solution derived after adding fewest BSs to satisfy the requirements may or may not be the optimal solution, compared to when all the demand nodes are considered and there is no constraint to include all the existing BSs in the solution as in the greenfield design approach (the total additional number of BSs required plus the existing BSs might be equal to or greater than $n^+ = 6$ BSs of the greenfield design case). Nevertheless (following the incremental design approach), the candidate TPs accounting for the 14 new demand nodes must be computed first. Figure 7.44 shows the resulted new set of TPs from the sweep and merge algorithm.

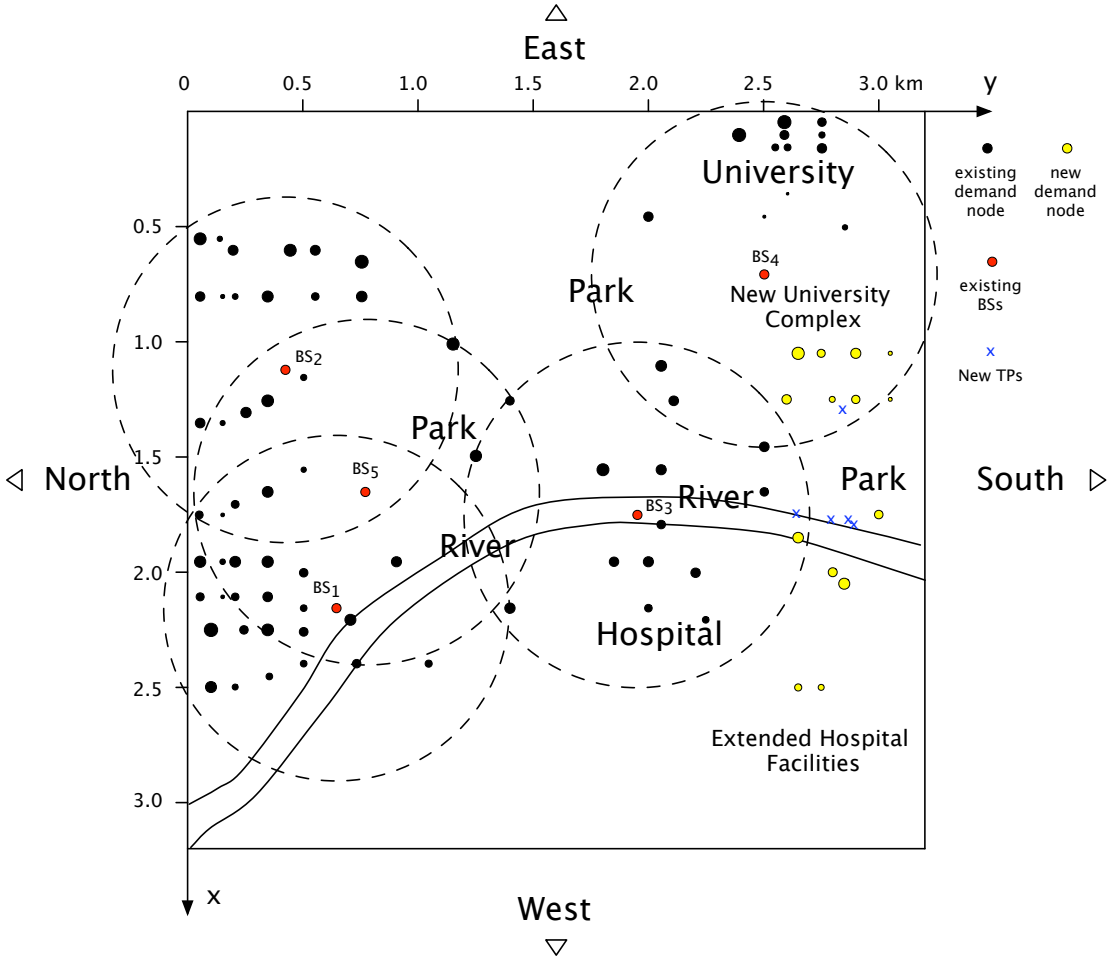

Figure 7.44: New Set of TPs for Added Demand Nodes (Graz Example - Expanded Network). Only 5 TPs are the result when only the 14 new demand nodes are considered.

Since there are only 14 new demand nodes, the sweep and merge algorithm only results in $N^* = 5$ TPs. For small N^*, the total number possible solutions (or the search space size S) are also small (i.e., $n^+ = 6, S = \binom{N^*}{n^+ - n^*} = \binom{5}{1} = 5$). Thus, an exhaustive search is reasonable for this case. When one more BS is added to the existing 5 BSs, the best solution is to place the new BS at TP (1.800, 2.900) km, as shown in figure 7.45.

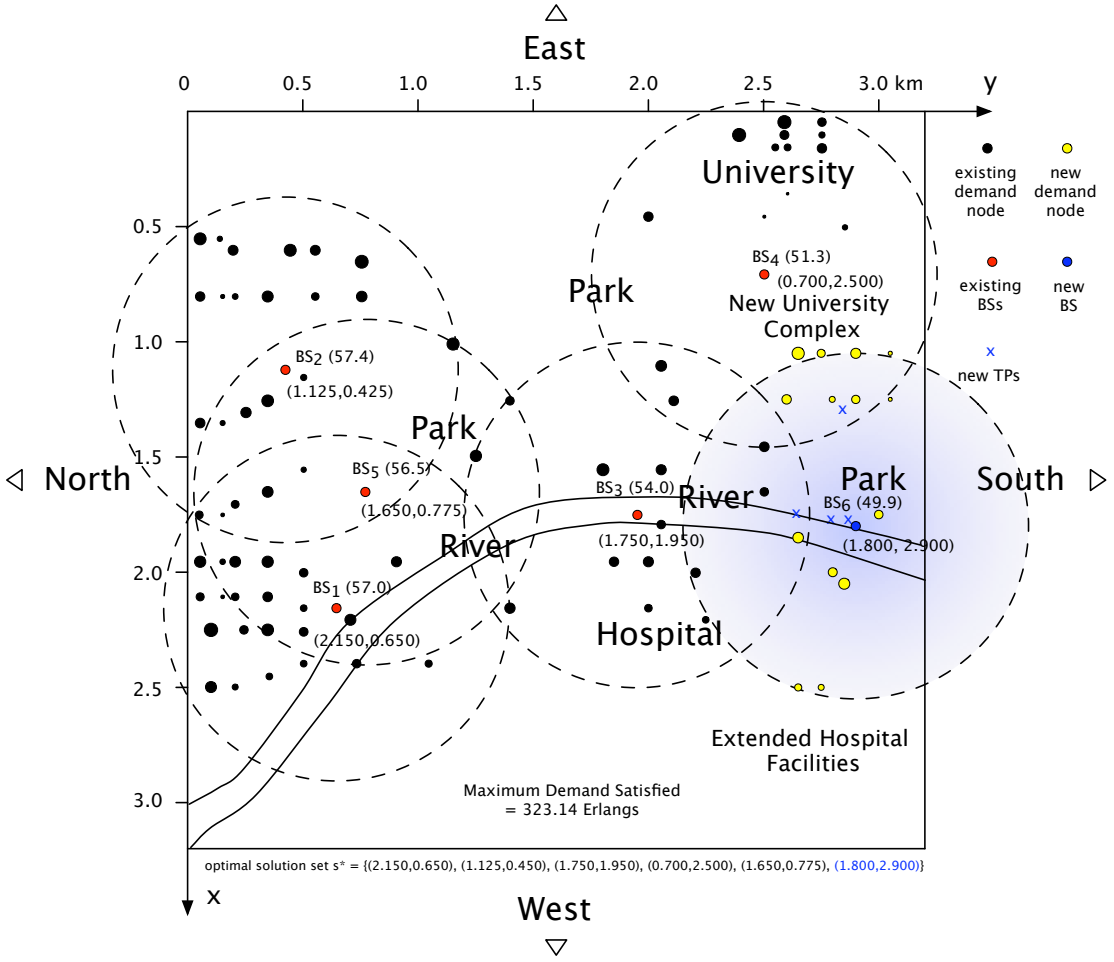

Figure 7.45: New Optimal Solution with Existing BSs (Graz Example - Expandedd Network). The new BS_6 is added and placed at TP (1.800, 2.900) km. The new network with $n^+ = 6$ BSs now can support the total demand = 323.14 Erlangs, meeting the capacity requirement.

Employing the optimal demand node assignment (described in section 6.2), the network with the newly installed BS_6 (including the existing BS_1 to BS_5) can support all the demand 323.14 Erlangs. Compared to previous optimal solution of the existing network with 273.38 Erlangs of traffic (figure 7.37), the demand assignment for BS_1, BS_2, and BS_5 remains unchanged, however the demand node assignment for the two existing BS_3 and BS_4 (also covering the new demand nodes) differ slightly — BS_3 are assigned with more demand than the previous from 48.8 to 54.0 Erlangs, while the demand assigned to BS_4 drops from 53.7 to 51.3 Erlangs. In this case, the solution in figure 7.45 is the optimal solution of the design (since the

minimum 6 BSs are needed to satisfy both the coverage and capacity requirements). Despite the fact that BS_3, BS_4, and BS_5 are placed differently from the greenfield design result (compare figure 7.45 and figure 7.43), the solution resulted from the incremental design approach described in this section matches the result from the greenfield design approach (employing equal number of BSs). Only 5 candidate solutions are needed to be evaluated (since $N^* = 5$ TPs, and only one BS is added), therefore the computational requirement for the incremental design approach is tremendously less than the greenfield design approach (by a factor $1.46 \times 10^5/5 \approx 10^4$). In general, the incremental design approach will produce an optimal solution when additional demand does not exceed a BS capacity and the existing BSs still have sufficient spare capacity to support the demand that can not be covered or exceed the capacity of the added BS(s). Otherwise, there is no solution to the incremental design unless more BSs are added.

8.0 CONCLUSIONS AND FUTURE WORK

Designing an optimal cellular network requires that a minimum number of Base Stations (BSs) must be employed, so that cost is minimized. To accomplish this objective, the concept of a demand node was incorporated to represent a quantum of user demand. An algorithm for computing a grid of Test Points (TPs) was developed so that the optimal solution is always guaranteed. The initial number of TPs (N) is usually very large since the required TP grid spacing (Δg) must be no larger than half the demand node grid spacing (Δx). The sweep and merge algorithm was then developed to reduce TPs from N initial grid points to N^* TPs, while maintaining the optimal solutions. In most cases, $N^* \ll N$, therefore the corresponding search space size $S \approx N^{n^*}$ (where n^* is the number of BSs needed) or equivalently the computation requirement, is reduced from N^{n^*} to N^{*n^*}. However, when user traffic is distributed over the entire space, N^* approaches $N/2$ (section 7.2.4). Demand nodes were employed to further reduce N^*. The demand node represents a point in the center of an area containing a quantum of user traffic, which can represent either voice calls or data bit rates. The minimum branching algorithm was developed to replace an exhaustive search to find the coverage solution, as the first step of the design optimization. Given demand node locations with $N^* \ll N$ TPs computed from the sweep and merge algorithm, the minimum branching algorithm always results in the optimal coverage solution in a more computationally efficient manner than an exhaustive search. In the worst case, the minimum branching algorithm reduces the computational requirement by a factor $\approx (N/N^*_{1,k})^{n^*}$ where $N^*_{1,k} \ll N^* \ll N$. Results from section 7.2.4 show a reduction $(N/N^*_{1,k})^{n^*}$ of 6 to 14 orders of magnitude.

Demand node assignment to a BS is also optimized, to maximize the total network capacity (section 6.2). However, even with an optimal demand node assignment, the coverage solution with n^* BSs computed from the minimum branching algorithm may or may not satisfy the capacity requirement. An algorithm based on Tabu Search (TS) was developed (section 6.3) to compute a new set of demand node to BS assignments (with $n^+ \geq n^*$ BSs) in cases when the coverage solution fails the capacity requirement test. Results from the Graz design examples in section 7.3 require the minimum number BSs to satisfy both the coverage and capacity requirements.

Results from various network topologies (i.e., random, clustered demand node distribution) and constraint requirements (i.e., voice call or data rate capacity, path loss and fading) in the previous chapters confirm that, the design algorithm efficiently searches the space and produces the optimal solution in a computationally efficient manner. Even with a moderate network size from the example in section 7.3, the computational requirement is reduced by a factor at least 10^{14}, while retaining the optimal solution.

The design algorithm was tested to measure its performance in producing an incremental design (section 7.3.8) when additional traffic arises and new demand nodes must be added to an existing network. The incremental design approach retains all existing BSs in the solution, then computes the fewest additional BSs needed to satisfy the increase in demand. Existing demand nodes are ignored when computing TPs, and only the new demand nodes are considered — resulting in fewer TPs than what is required by a greenfield design approach. Together with fewer BSs to place (because all existing BSs are assumed to be already in the solution), the incremental design approach requires an incremental number of computations. The result of the incremental design approach was compared against the greenfield design approach which employed the sweep and merge algorithm to compute a new set of N^* TPs considering both the existing and new demand nodes (where all BSs needed to satisfy both the coverage and capacity requirement of the new network are recomputed from the beginning). Since the incremental design must retain all existing BSs, which may or may not be

part of the optimal solution of the new network, therefore the solution from the incremental design approach is not guaranteed to be optimal. However, results show that the incremental design approach can produce the optimal solution as well as the greenfield design approach. In general, the incremental design will be optimal when the increased demand is less than the capacity of the added BS(s), and the existing BS(s) have sufficient spare capacity to support the excess demand.

8.1 LIMITATIONS

Incorporating demand nodes into the design algorithm introduces several tradeoffs. Demand nodes are calculated by quantizing the traffic load into points, where each point represents the center of an area containing a quantum of demand (i.e., Erlangs, kbps). Employing the demand node not only simplifies the design process, but it is a practical mean to account for user mobility, coverage granularity, and traffic classification. There are two parameters involved in calculating the demand nodes: traffic sampling resolution Δx, and quantization of demand Δq. The sampling resolution Δx determines the minimum grid spacing between demand nodes, while the quantization threshold Δq determines the maximum quantum of traffic that each demand node point can represent. When Δx is infinitesimally small ($\Delta x \approx 0$), the demand node point becomes a continuous space, and the number of demand nodes are infinite. Thus, choosing a small Δx provides finer granularity to both the aspects of signal coverage requirement and user movement, however demand nodes will no longer be stationary and the number of computations will increase because more evaluations are needed as the number of demand nodes increase. In the opposite case, choosing a larger Δx results in fewer demand nodes, decreases computations, and increases the likelihood that demand nodes remain stationary. However, the larger Δx is, the larger the area and the amount of traffic it represents, and therefore from the perspective of mobility, user movement and coverage granularity is reduced — some areas or points may not receive sufficient signal coverage. Similar to Δx, when the quantization of demand Δq approaches zero ($\Delta q \to 0$), it means that each of the demand nodes will account for the smallest unit of traffic from every

application employed by each individual user. Due to user movement and the bursty nature of data applications, choosing too small Δq may create more demand nodes than needed. However, when Δq is too large, fewer demand nodes are present, and the corresponding demand may be so large that it can no longer differentiate traffic from various users in different geographical topologies. In the design algorithm, a specific combination of Δx and Δq must be chosen in order to calculate the demand nodes, and as discussed earlier, both the coverage granularity and traffic classification must be compromised. However, the choice of Δx and Δq may differ when a demand space (section 8.2) is used instead of the demand node point.

In some situations, the coverage radius R may not be the same for all BSs (i.e., to reduce interference). However, when the problem size is large and more BSs are needed, the design process can be computationally excessive because the computation requirement (C') increases exponentially as a function of the number of BSs (n^*) multiplied by the choices of R (r), where $C' \propto N^{*(n^* r)}$.

8.2 FUTURE WORK

There are several possibilities to improve the design algorithm. One possibility is to better approximate the BS capacity by accurately mapping the data-bit-rate requirement to Erlangs through a complete of analysis of traffic queueing from various applications (i.e., HTTP, FTP, SMS, MMS, voice and video streaming, etc) with different usage distributions and data rate requirements. Using accurate approximation of the BS capacity, the minimum number of BSs needed can be pre-computed, and it is also possible to determine under what conditions the incremental design is optimal. To address the excessive interference problem, especially in CDMA systems, resulting from multiple coverage overlap among BSs, one may add the interference constraint in the problem formulation on top of the coverage and capacity requirements, when computing for the optimal BS placement. Additionally,

the demand node may be calculated so that it represents a demand space instead of a point, which will provide coverage granularity to the design results. The concept of demand node space is plausible by taking account the signal fading (fluctuation of signal strength over space and time) — such that with equal probability every point in a particular space in the design area may receive statistically the same average signal strength. Furthermore, since different grid points may be located on different geographical regions and environments, the sweep and merge algorithm may employ other path loss models to more accurately compute the demand node set coverage for each individual grid point. Lastly, to account for the fact that the candidate locations for placing BSs or TPs may not be just a simple grid points due to several factors such as natural obstructions, city regulations, spacing conflicts etc., the sweep and merge algorithm may be modified to work on any configuration of TP locations (not restricted only to the equally spaced grid points) and may also include a probability-based capacity requirement to further reduce TPs.

APPENDIX A

LIST OF NOTATIONS AND SYMBOLS

All important notations, symbols, and variables used throughout chapter 5 to chapter 7 of this paper are listed in the table below (in order of appearance).

Table A1: Notations, Symbols, and Variables

Symbol	Definition
Δx	(x-axis) demand node grid spacing (m)
Δy	(y-axis) demand node grid spacing (m)
X	(x-axis) design area dimension (m)
Y	(y-axis) design area dimension (m)
M	total number of demand nodes
k, k'	demand node index, $k, k' \in \{1, \ldots, M\}$
m_k	demand node k^{th}
X_k	m_k (x-axis) coordinate (m)
Y_k	m_k (y-axis) coordinate (m)
A_k	m_k demand unit (e.g., in Erlangs, kbps, etc)
i, i'	grid point (x-axis) index $i, i' \in \{1, 2, \ldots\}$
j, j'	grid point (y-axis) index $j, j' \in \{1, 2, \ldots\}$
Δg	TP grid spacing (m)
$g_{i,j}$	grid point located at $((i-1)\Delta g, (j-1)\Delta g)$

S	search space size
N	total number of grid points
n^*	minimum number of BSs needed to satisfied the coverage requirement
N_x	maximum (x-axis) grid index
N_y	maximum (y-axis) grid index
G	matrix of grid points $\{g_{i,j}\}, \quad \forall i \in \{1,\ldots,N_x\},\ \forall j \in \{1,\ldots,N_y\}$
δ	any real number (chapter 5)
R	effective BS coverage radius (m)
n, n'	TP index, $n, n' \in \{1, 2, \ldots\}$
i_n	grid point (x-axis) index corresponding TP_n
j_n	grid point (y-axis) index corresponding TP_n
g_{i_n,j_n}	grid point located at $((i_n-1)\Delta g, (j_n-1)\Delta g)$
b_n	$\begin{cases} 1 & : \text{ if a BS is installed at grid } \mathsf{TP}_n \\ 0 & : \text{ otherwise} \end{cases}$
d_{kn}	distance between m_k and b_n (m)
m_{kn}	$\begin{cases} 1 & : \text{ if demand node } m_k \text{ is assigned to } b_n \\ 0 & : \text{ otherwise} \end{cases}$
C	maximum BS capacity (e.g., in Erlangs, kbps, etc)
$z_{i,j}$	a set of demand nodes m_k within R from grid $g_{i,j}$
G^*	matrix of grid points resulted from the sweep and merging algorithm
s, s', s^*	a set of solution $\{b_n \vert b_n = 1\}$
z_{i_n,j_n}	a set of demand nodes m_k within R from grid g_{i_n,j_n}
z_{b_n}	a set of demand nodes assigned to b_n
N^*	size of matrix G^* (total number of TPs)
C'	computational requirement
$N^*_{1,k}$	1^{st} level average number of TPs (nodes) in the (minimum branching) tree
$N^*_{2,k}$	2^{nd} level average number of TPs (nodes) in the (minimum branching) tree
$N^*_{n^*,k}$	n^{*th} level average number of TPs (nodes) in the (minimum branching) tree
δ	residual number of nodes in the deepest level (chapter 6)
n^+	number of BSs

$\mathcal{N}(s)$	a set of neighborhood solutions belong to the current solution s
h	number of closest TPs to each b_n
$c_{s'}$	total network capacity in which the candidate solution s' can support
$p_{s'}$	penalty associated with the candidate solution s'
w	weight associated with $p_{s'}$

APPENDIX B

CDMA2000 EV-DO TECHNOLOGY

The 1xEV-DO system was proposed by 3GPP2 to improve data rates from the original CDMA2000. Through different modulations and codings, 1xEV-DO can support upto 153.6 and 2,457.6 kbps in the reverse and forward links respectively [11]. The reverse link exploits the pilot channel transmitted by a MS to coherently demodulate data signals. The forward link allocates all code spaces and transmits at a full power to each MS regardless of the data rates. The forward link employs a TDM technique to transmit data packets to different MSs, but does not specify the scheduling algorithm.

B.1 REVERSE LINK

The 1xEV-DO reverse link consists of the Access Channel and the Traffic channel. Figure B1 illustrates the 1xEV-DO reverse link channel diagram. A MS uses the Access channel to initiate communication with the BS. The traffic channel consists of the pilot, data, MAC (Medium ACess), and ACK (ACKnowledgement) channels. The physical layer packets of the reverse channel define a frame length of 16 slots, 1.67 ms per slot. A MS transmits on the pilot and data channels simultaneously. The pilot assists the BS in coherently demodulating the packets. Esteves measured values of the required pilot bit energy-to-total noise ratio ϵ to achieve a 1% PER (Packet Error Rate) for various reverse data rates to range from -24.8 to -22.8 dB [63]. In this paper, ϵ is modeled as a lognormal random variable with mean

22.75 dB and standard deviation 1.3 dB as suggested by Esteves [63]. The transmit power

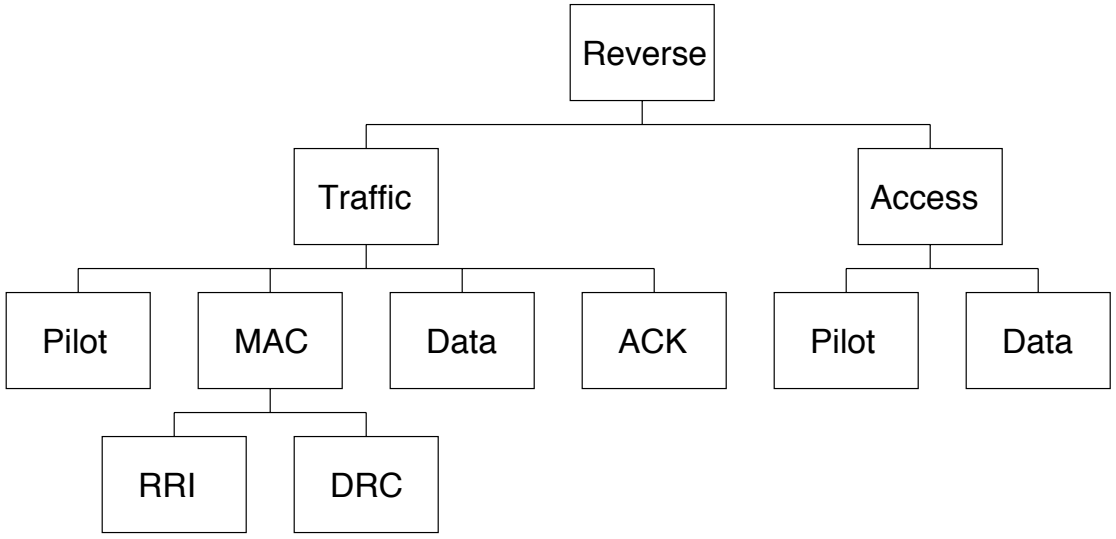

Figure B1: CDMA2000 1xEV-DO Reverse Channel Structure

on the data channel is proportional to the data rate and is measured as a gain over the pilot channel — data channel gain. Table B1 lists the required data channel gain for different transmit data rates [11].

Table B1: Default Data Channel Gain

Data Rate (kbps)	Data Channel Gain (dB)
Null	$-\infty$
9.6	3.75
19.2	6.75
38.4	9.75
76.8	13.25
153.6	18.50

B.1.1 Signal to Noise Ratio Requirement

The region serviced by the BS is determined by the received SNR (Signal to Noise Ratio). The SNR must be high enough to overcome signal fluctuations from channel fading while maintaining the minimum required bit energy-to-total noise density ratio (E_b/N_t) under the

loaded condition. For CDMA2000 1xEV-DO, the required SNR is a function of the transmit data rate and the cell loading factor ρ. Figure B2 shows SNR as a function of the loading factor ρ for various data rates. CDMA2000 system designers generally limit the cell loading

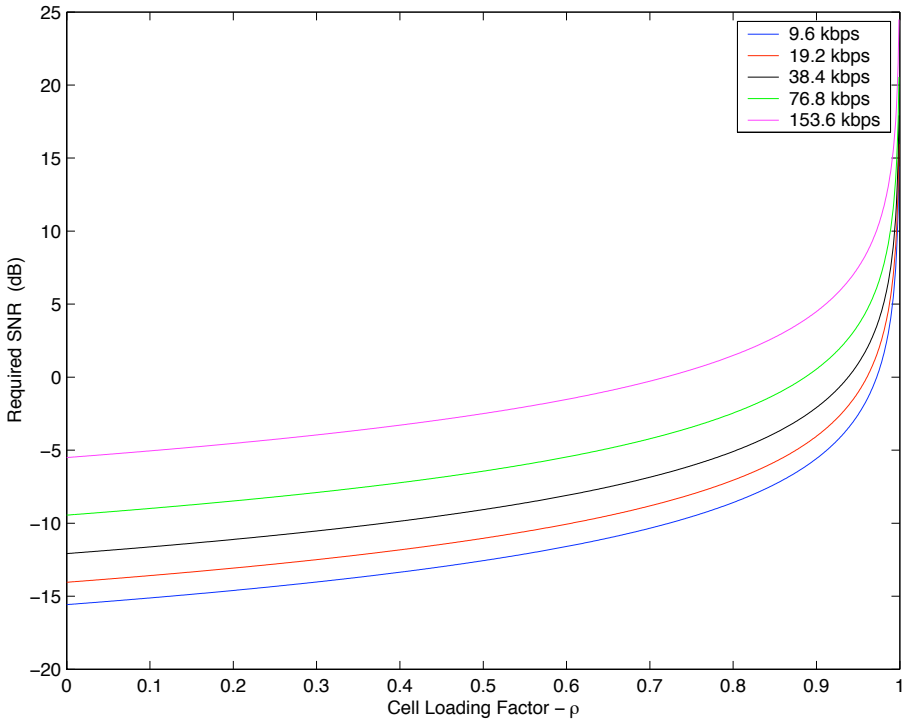

Figure B2: Signal-to-Noise Ratio (SNR) vs. Cell Loading (ρ) for various data rates ranging from 9.6 to 153.6 kbps by assuming that the required ϵ is fixed at the median value = -22.75 dB.

factor (ρ) between 0.5 to 0.7 to ensure a stable total noise level in a system [51], [60]. For the worst case when the transmit data rate = 153.6 kbps and the data channel gain = 18.5 dB, the corresponding SNR at (ρ = 0.7) $\approx$ -0.27 dB. In addition to the cell loading, the minimum SNR requirement must also include for the channel fading effect. Viterbi showed that a fading margin ς = 6.2 dB is sufficient to ensure that 90% of the time the SNR received from the cell edge meets the requirement [40]. Furthermore, to allow for increasing cell loading, additional margin = 4.07 dB (arbitrary) is also added. The minimum SNR requirement used in this paper is thus set to = -0.27 + 6.2 + 4.07 = 10 dB.

B.2 FORWARD LINK

The forward link channel supports data rates ranging from 38.4 kbps upto 2,457.6 kbps through different modulation and coding schemes [64]. Unlike the reverse link in which the data rate is regulated by the cell loading level, the data rate available for the forward channel depends on the channel condition. The higher the received SIR, the higher the data rate a BS can transmit. Table B2 shows a set of 1xEV-DO forward data rates, modulations, codings, and the required SIR [11], [13]. The CDMA2000 1xEV-DO employs both the TDM

Table B2: Modulation Schemes, Codings and SIRs at 1% PER for different data rates

Data Rate (kbps)	Modulation	Coding	# of slots	SIR (dB)
38.4	QPSK	1/5	16	-11.5
76.8	QPSK	1/5	8	-9.2
153.6	QPSK	1/5	4	-6.5
307.2	QPSK	1/5	2	-3.5
614.4	QPSK	1/3	1	-0.5
921.6	8PSK	1/3	2	2.2
1228.8	16QAM	1/3	2	4.0
1843.2	8PSK	1/3	1	8.0
2457.6	16QAM	1/3	1	10.3

and CDM (Time Division Multiplex and Code Division Multiplex) operating at a full power transmission per one user at a time. Wu and Esteves showed that allocating full power and the entire code space to one user per transmission maximizes the average BS throughput [13]. Figure B3 compares the IS-95 and CDMA2000 1xEV-DO forward channel power distribution.

The TDM operation on the forward channel defines a frame length of 26.67 ms, 16 slots in a frame, 2,048 chips/1.67 ms duration per slot [11]. A MS constantly monitors the BS pilot burst transmitted twice a time slot and calculates the pilot SIR accordingly. The MS then maps the calculated pilot SIR to the maximum data rate given in table B2 and issues a corresponding data rate request to the BS via the reverse DRC channel. Upon receiving the data rate request via the DRC channel, the BS invokes its scheduling algorithm to determine a time slot allocation for each MS. Transmission of the data packet occurs once every four

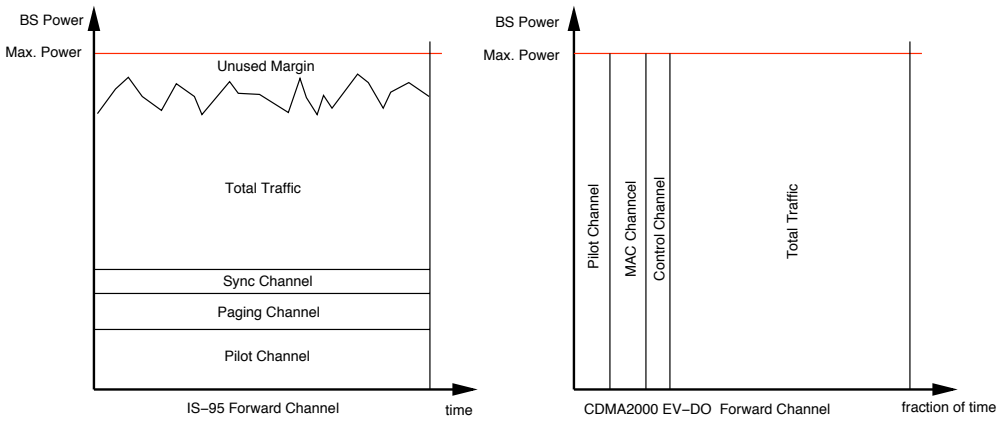

Figure B3: Power distribution among different channels of IS-95 vs. CDMA2000 1xEV-DO (adapted from [64]).

slots regardless of the rate. BS either transmits a multiple-slot packet in a successive manner or using a 4-slot interlacing technique. The interlacing technique enables the receiving MS to partially decode the packet — if encoded with enough redundancy — and to notify the BS via the ACK channel for early termination if the decoding is successful. Alternatively for a successive slot operation and a normal termination mode, a successful decoding requires a completed reception of multiple slots of a packet.

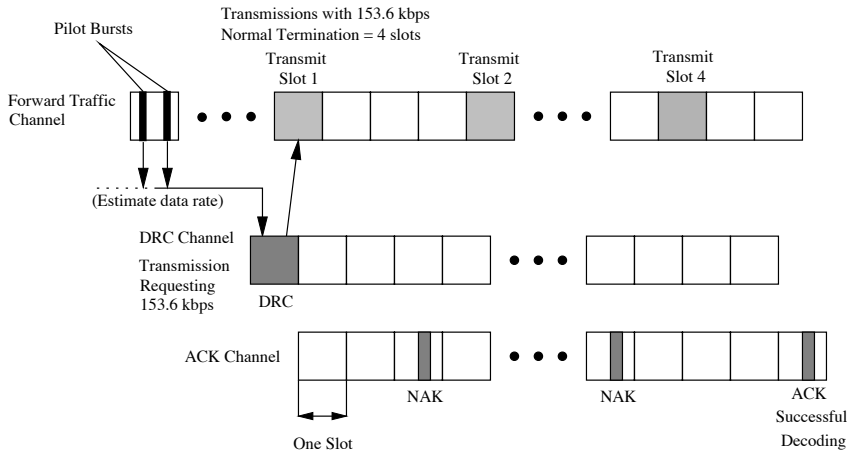

Figure B4: Multi-slot Physical Layer Packet with Normal Termination

Figure B4 depicts the forward link normal termination mode with the pilot burst and the DRC channel [11]. Unlike TDMA systems where different users sharing the same frequency channel but are assigned a fixed time slot allocation, the 1xEV-DO TDM operation does not mandate a specific time slot allocation or scheduling algorithm, thus allowing a flexible implementation. The next section briefly explains various scheduling algorithms proposed for 1xEV-DO systems.

B.2.1 Scheduling Algorithm

Various TDM scheduling algorithms were proposed to schedule data transmission and to allocate the number of slots for multiple users with various data rate requests. Qualcomm recommends a proportional fairness scheduling to maximize user's moving average throughput [64]. The algorithm delivers data transmission only when a user's DRC request is greater or equal to its running average throughput computed over the last — e.g., 1,024 time slots. Qualcomm claimed that moving users achieve a higher throughput than their average DRC requests when employing the proportional fairness scheduling. However, the algorithm requires long observation time to compute the running average and does not consider the delay problem.

Alternatively, the simple round robin algorithm schedules a transmission to different users in a first come first serve basis. Whenever there is a DRC request, the round robin scheduler just delivers a transmission at the data rate according to the request. If for every DRC request the scheduler allocates a constant number of slots regardless of the data rate, users with poor channel conditions will suffer longer delay than users with better channel conditions. *In the extreme case, only a small number of users may receive at or near the maximum throughput, while the rest may not receiver at all (due to considerable packet delay).* Nevertheless, the overall network throughput is maximized. To compromise between throughput and latency, Bender proposed a method that divides data rate request into two classes of latency — any data rate requests below the threshold are assigned the maximum latency, while the rest

above the threshold are assigned the minimum latency [65]. Bender's bimodal algorithm achieves the fairness by limiting the maximum latency ratio among all data rates. However, the latency distribution is not smooth — two adjacent data rates are assigned the minimum and the maximum latency respectively. The problem is undesirable in the situation for example, when these rates are the two most favorable modes of transmission. Kim extended Bender's work and fixed this problem by allowing a smooth transition in latency between two adjacent data rates. Instead of limiting to only the maximum and minimum delay assignments, Kim's proposed an algorithm that provides multiple delay assignments for different data rates — multimodal algorithm. The resulting average throughput reduces when compared with the Bender's scheme. However, the fairness in latency (e.g., waiting time) between two data rates improves.

In conclusion, determining a proper scheduling algorithm for the 1xEV-DO forward link involves several factors such as throughput, latency, and DRC probability distribution. The proportional fairness or the simple round robin with equal slot allocation may be effective for maximizing throughput. However, if the fairness in latency is concerned, either the Bender or the Kim algorithm should be more appropriate, depending on the DRC probability distribution.

BIBLIOGRAPHY

[1] E. Amaldi, A. Capone, and F. Malucelli, "Planning UMTS Base Station Location: Optimization Models with Power Control and Algorithms," *IEEE Trans. Wireless Commun.*, vol. 2, no. 5, September 2003.

[2] H. R. Anderson and J. P. McGeehan, "Optimizing Microcell Base Station Locations Using Simulated Annealing Techniques," in *Proc. 44th IEEE VTC Conference*, September 1994, pp. 858–862.

[3] C. Prommak, "Demand-Based Network Planning for WLANS," Ph.D. dissertation, School of Information Sciences, University of Pittsburgh, June 2004.

[4] D. J. Goodman, *Wireless Personal Communications Systems*. Reading, MA: Addison-Wesley, 1997.

[5] J. Schiller, *Mobile Communications*. London: Addison-Wesley, 2000.

[6] E. Dahlman *et al.*, "UMTS/IMT-2000 Based on Wideband CDMA," *IEEE Commun. Mag.*, pp. 70–80, September 1998.

[7] ——, "WCDMA — The Radio Interface for Future Mobile Multimedia Communications," *IEEE Trans. Veh. Technol.*, vol. 47, no. 4, pp. 1105–1118, November 1998.

[8] "HSDPA for Improved Downlink Data Transfer," Qualcomm Inc, Tech. Rep., October 2004.

[9] "How to Realise the Benefits of Mobile Broadband Today," GSM Association, February 2007.

[10] D. N. Knisely *et al.*, "Evolution of Wireless Data Services: IS-95 to cdma2000," *IEEE Commun. Mag.*, pp. 140–149, October 1998.

[11] *cdma2000 High Rate Packet Data Air Interface Specification*, 3rd Generation Partnership Project 2 (3GPP2) Std. C.S20 024 v2.0, October 2000.

[12] "Technical Overview of 1xEV-DV," Motorola, Inc., September 2002.

[13] Q. Wu and E. Esteves, "The CDMA2000 High Rate Packet Data System," Qualcomm Inc., Tech. Rep. 80-H0593-1, March 2002.

[14] V. P. Mhatre, "CDMA 2000: 1xEVDO and 1xEVDV, An Overview," April 2004.

[15] "CDMA2000: Market Facts," CDG, March 2005.

[16] "4Q 2004 CDMA Subscribers Statistics," CDG, December 2004.

[17] "The Economics of Mobile Wireless Data," Qualcomm Inc.

[18] "Delivering Voice and Data: Comparing CDMA2000 and GSM/GPRS/EDGE/UMTS," The CDMA Development Group, December 2005.

[19] "Mobile WiMAX Part I: A Technical Overview and Performance Evaluation," WiMAX Forum, Tech. Rep., August 2006.

[20] "Fixed, Nomadic, Portable and Mobile Applications for 802.16-2004 and 802.16e WiMAX Networks ," WiMAX Forum, Tech. Rep., November 2005.

[21] "Business Case Models for Fixed Broadband Wireless Access based on WiMAX Technology and the 802.16 Standard," WiMAX Forum, Tech. Rep., October 2004.

[22] Sprint Nextel Announces 4G Wireless Broadband Initiative with Intel, Motorola and Samsung. [Online]. Available: http://www2.sprint.com/mr/news_dtl.do?id=12960

[23] "KDDI and WiMAX Convergence in the Land of the Rising Sun," WiMAX Forum, August 2006.

[24] C. Smith, *Practical Cellular & PCS Design*. New York: McGraw-Hill, 1997.

[25] K. Tutschku and P. Tran-Gia, "Spatial Traffic Estimation and Characterization for Mobile Communication Network Design," *IEEE J. Select. Areas Commun.*, vol. 16, no. 5, June 1998.

[26] R. G. Akl *et al.*, "Multicell CDMA Network Design," *IEEE Trans. Veh. Technol.*, vol. 50, no. 3, May 2001.

[27] Q. Hao *et al.*, "A Low-Cost Cellular Mobile Communication System: A Hierarchical Optimization Network Resource Planning Approach," *IEEE J. Select. Areas Commun.*, vol. 15, no. 7, September 1997.

[28] N. Weicker, G. Szabo, K. Weicker, and P. Widmayer, "Evolutionary Multiobjective Optimization for Base Station Transmitter Placement with Frequency Assignment," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, April 2003.

[29] A. Hills, "Large-Scale Wireless LAN Design," *IEEE Commun. Mag.*, November 2001.

[30] H. D. Sherali, C. M. Pendyala, and T. S. Rappaport, "Optimal Location of Transmitters for Micro-Cellular Radio Communication System Design," *IEEE Trans. Veh. Technol.*, vol. 14, no. 4, May 1996.

[31] M. Unbehaun and M. Kamenetsky, "On the Deployment of Picocellular Wireless Infrastructure," *IEEE Wireless Communications*, December 2003.

[32] K. Lieska *et al.*, "Radio Coverage Optimization with Genetic Algorithms," in *Proc. IEEE*, September 1998, pp. 318–322.

[33] C. Ratti *et al.*, "Mobile Landscapes: Graz in Real Time," SENSEable City Lab, MIT, 2005.

[34] F. Alejandro *et al.*, "Full- and Half-Square Cell Plans in Urban CDMA Microcellular Networks," *IEEE Trans. Veh. Technol.*, vol. 52, no. 3, May 2003.

[35] S. Hanly and R. Mathar, "On the Optimal Base-Station Density for CDMA Cellular Network," *IEEE Trans. Commun.*, vol. 50, no. 8, pp. 1274–1281, August 2002.

[36] M. J. Feuerstein *et al.*, "Path Loss, Delay Spread, and Outage Models as Functions of Antenna Height for Microcellular System Design," *IEEE Trans. Veh. Technol.*, vol. 43, no. 3, pp. 487–498, August 1994.

[37] K. Tutschku, "Demand-based Radio network Planning of Cellular Mobile Communication Systems," in *Proc. IEEE INFOCOM (3)*, 1998, pp. 1054–1061.

[38] C. Y. Lee and H. G. Kang, "Cell Planning with Capacity Expansion in Mobile Communications: A Tabu Search Approach," *IEEE Trans. Veh. Technol.*, vol. 49, no. 5, September 2000.

[39] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communication*. Reading, MA: Addison Wesley, 1998.

[40] A. J. Viterbi *et al.*, "Soft Handoff Extends CDMA Cell Coverage and Increase Reverse Link Capacity," *IEEE J. Select. Areas Commun.*, vol. 12, no. 8, pp. 1281 – 1288, October 1994.

[41] M. Hata, "Empirical Formula for Propagation Loss in Land Mobile Radio Services," *IEEE Trans. Veh. Technol.*, vol. 29, pp. 317–235, August 1980.

[42] J. M. Keenan and A. J. Motley, "Radio Coverage in Buildings," *British Telecom Technology*, vol. 8, no. 1, pp. 19–24, January 1990.

[43] M. F. Catedra and J. Perez-Arriaga, *Cell Planning for Wireless Communications*. Artech House, 1999.

[44] J. W. McKown and R. L. Hamilton, "Ray Tracing as a Design Tool for Radio Networks," *IEEE Network Magazine*, pp. 27–33, November 1991.

[45] S. Y. Seidel and T. S. Rappaport, "Site-Specific Propagation Prediction for Wireless In-Building Personal Communication System Design," *IEEE Trans. Veh. Technol.*, vol. 43, no. 4, pp. 879–891, November 1994.

[46] K. Pahlavan and A. Levesque, *Wireless Information Networks*. Wiley, 1995.

[47] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[48] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA: Kluwer Academic, 1997.

[49] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Berlin, Germany: Springer, 2000.

[50] J. Clausen, "Branch and Bound Algorithms - Principles and Examples," March 1999.

[51] V. K. Garg, *Wireless Network Evolution: 2G to 3G*. Upper Saddle River, NJ: Prentice Hall, 2002.

[52] Q. Bi, "A Forward Link Performance Study of the 1xEV-DO Rev.0 System Using Field Measurements and Simulations," *Bell Labs Technical Journal*, vol. 10, no. 2, pp. 5 – 19, 2005.

[53] N. Pongthaipat and J. Kabara, "Designing Wireless Networks by Test Point Reduction," in *Proc. ISWPC '07*, February 2007, pp. 307–310.

[54] J. C. S. Cheung *et al.*, "Network Planning for Third-Generation Mobile Radio Systems," *IEEE Commun. Mag.*, pp. 54–59, November 1994.

[55] C. Ratti *et al.* Mobile Landscape | Graz in Real Time. [Online]. Available: http://senseable.mit.edu/projects/graz/graz.htm

[56] Graz. [Online]. Available: http://en.wikipedia.org/wiki/Graz

[57] Pittsburgh, Pennsylvania. [Online]. Available: http://en.wikipedia.org/wiki/Pittsburgh,_Pennsylvania

[58] *cdma2000 Evaluation Methodology — Revision 0*, 3rd Generation Partnership Project 2 (3GPP2) Std. 3GPP2 C.R1002-0, December 2004.

[59] "3G Offered Traffic - All Services," UMTS Forum, June 2003.

[60] Q. Bi *et al.*, "Performance of 1xEV-DO Third-Generation Wireless High-Speed Data Systems," *Bell Labs Technical Journal*, vol. 7, no. 3, pp. 97 – 107, 2003.

[61] "All-IP 1xEV-DO Wireless Data Networks," Airvana Inc., Tech. Rep., August 2004.

[62] K. L. D. Staehle and P. Tran-Gia, "Source Traffic Modeling of Wireless Applications," Lehrstuhl für Informatik III, Universität Würzburg, Tech. Rep., June 2000.

[63] E. Esteves, "On the Reverse Link Capacity of cdma2000 High Rate Packet Data Systems," in *Proc. IEEE ICC*, vol. 3, April 2002, pp. 1823 – 1828.

[64] "1xEV: 1x EVolution IS-856 TIA/EIA Standard," Airlink, Tech. Rep. Rev. 7.2, November 2001.

[65] P. Bender *et al.*, "CDMA/HDR: A Bandwidth Efficient High Speed Wireless Data Service for Nomadic Users," *IEEE Commun. Mag.*, vol. 38, no. 7, pp. 70–77, July 2000.