ACCELERATING THE SIMULATION OF WIRELESS CELLULAR SYSTEMS

A Dissertation

Presented to

The Faculty of the Department of Computer Science

The College of William & Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

by

Luiz Felipe de Lima Perrone

2000

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

Luiz Felipe de Lima Perrone

Approved, July 2000

David M. Nicol
Thesis Advisor

Phil Kearns

Stephen K. Park

Xiaodong Zhang

Larry M. Leemis
Department of Mathematics

To those who were, have been and will be... Friends.

Table of Contents

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to the many people who contributed in so many ways to my intellectual and emotional growth during the course of this program.

First and foremost, I must thank my advisor David Nicol. As it usually goes with most graduate students and their mentors, our academic relationship went through ups and downs, along the years. In spite of this, the fact that I could count on his friendship, guidance and diligent support was always a known constant. I've had a chance to learn much from him and I hope to have used it well.

My immediate and extended family was another safe harbor during all this time. My parents, Luiz and Ilka, and my brother André were there for me, even if *there* sometimes meant thousands of miles away, in another continent. The Ueki's, my "second" family, were responsible for motivating me to come to this country and continue my education, and also for easing my transition to a new lifestyle, virtually eliminating any threat of culture shock. Daniel and Thomas, especially, lent me a hand in many different ways and gave me words of cheer and encouragement at points when I needed them.

I made several good friends in these years, both at William & Mary and at Dartmouth, where I went as visiting student to be geographically close to my advisor. The camaraderie, consideration and kindness these people have shown me will never be forgotten. Anna Brunström and Xiaowen "Jason" Liu, in particular, have shown the disposition to engage me in productive discussions about research and several times fed me some good ideas to consider. I also express much thankfulness to my brave proofreaders, Jason and BJ Premore, who devoted much of their time to helping me improve this text. Thanks, also, to Andy Miner and to Tracey Beauchat for being there at the frantic stages that preceded my defense.

As I contemplate the possibility of an academic career, I recall two names who have contaminated me with enthusiasm for science and, by example, taught me much about *the art of lecturing.* Rahul Simha and Phil Kearns were more than professors, they were role models. If I ever get involved in teaching, these two take, at least, partial blame.

Even before I arrived at William & Mary, I already owned gratitude to a special person in the department. Vanessa Godwin kept me away from annoying bureaucracy as much as humanly possible. More than that, whenever she could offer any help, there she was. There are more times she made my life easier than I can tell. I have often thought of her as a guardian angel and, in a sense, she was one.

I must acknowledge the Dept. of Computer Science at Dartmouth College. As a visitor, I came to you seeing myself as an outsider, but you made me feel like one of your own. I was given the best possible work conditions and it was a true pleasure being in such an exciting and stimulating academic environment.

Finally, I would like to thank all the members of my committee, who helped me learn what it takes to put a dissertation together. I owe special gratitude to Larry Leemis, from the Math Department, who was extremely helpful in pointing out so many possibilities for improvement.

List of Figures

ABSTRACT

The simulation of comprehensive models for cellular wireless systems poses a computational burden of great proportions. When a sub-model for transmitter power level control is included in the simulation, a continuous process in discrete-time is introduced, requiring traditional execution to advance in small, regular time-steps. To accelerate these simulations, we propose the use of *interval jumping*, a novel technique which allows time to progress in adaptive, irregularly-sized jumps. The foundations for this mechanism are laid out in light of the simulation of a complex simulation model which includes teletraffic, radio propagation, channel allocation, transmitter power control, and user mobility. We demonstrate the performance of this method through the use of sequential and parallel simulation.

Approaching the problem of accelerating the simulation of wireless systems from a different angle, we also identify a second important performance bottleneck. Calculations for interference computation, which may be carried out hundreds of times for each second of simulated time, require the evaluation of $O(N^2)$ interactions, for a system with N transmitter/receiver pairs. In order to provide a computationally cheaper and more scalable alternative to these operations, we study the applicability of an N-body algorithm, which brings time complexity down to $O(N \log N)$.

ACCELERATING THE SIMULATION OF WIRELESS CELLULAR SYSTEMS

Chapter 1

Introduction and Motivation

Wireless cellular systems can be seen as the culmination of the evolutionary process of mobile communications. Today's digital systems, termed the *second generation* of wireless networks, provide a variety of services that encompass paging, voice and data transmission. The plans that have been laid out for a third generation (3G) will take the standards of speed and quality of wireless communication much higher, rivaling even the wired phone system.

The convenience and flexibility offered by wireless networks has created enough interest, up to now, to determine a level of demand that has grown exponentially over the years. Pagers, cellular phones and PCS (Personal Communication Service) units, which were initially very expensive gadgets, are now accessible to the common person. As the new standards become reality and subscription costs continue to drop, it is expected that the number of subscribers will grow even faster.

Cellular wireless networks have been created, from their very beginning, to meet this rapidly growing demand. Scalability in the face of a scarce and precious resource, the radio spectrum, has always been one of the foremost goals of these systems. Consequently, the design of a wireless network is no easy task. Not only is it hard to engineer a scalable and reliable system, but it is also difficult to validate and evaluate it, since the available tools for mathematical analysis provide very limited insight. The remaining alternatives are to test each specific design either by physical implementation in the field or through the use of computer simulation.

At early design stages, physical implementation is hardly ever an option. The costs of realizing the whole system before a design has been refined to a reasonable level of performance are simply prohibitive. At times, it may be possible to construct scaled-down versions of the system and use them for design evaluation. Clearly, great care must be taken in analyses with scaled-down implementations, so that the results obtained can be extrapolated to create valid predictions of behavior for the full-sized system.

What makes computer simulation the best option for the process of design refinement is the fact that it warrants reproducibility and control. Complex test scenarios can be created to exercise each specific system feature and expose potential design flaws. This process of problem identification and correction can be repeated until the system design reaches satisfactory levels of reliability and performance.

The total cost of a simulation study breaks down into three major factors: the human cost of model development, the computational cost of model execution and the combined costs of output analysis. Model development has become easier and cheaper due to the use of design patterns and component databases [27, 38]. This process has evolved to a point where the modeler is able to realize a project from base components previously developed, tested and validated by experts. Modeling has become more focused on design rather than on low-level programming and, thus, the associated costs have been much reduced.

Considering the dwindling costs of modeling and the fact that the costs of a simulation's output analysis are comparable to those of the analysis of the physical system, it is clear that the dominant cost in a simulation study will be that of model execution. The computational resources required for the simulation of large-scale, comprehensive and detailed models are, in general, equally massive.

It's a well-known fact that the most detailed models require the most computational power. Given the current rapid advances in computer technology, one would think that with a fast uniprocessor or perhaps a large multiprocessor, simulations of wireless systems could be executed in reasonable time. What *reasonable time* means, though, can be a point of contention. One unit of time in the real system can easily turn into hundreds or thousands of units of computational time depending on the model employed. Moreover, in order to produce statistically sound results, long and repeated runs with the same model are required. All these facts may seem to detract from the viability of simulation of wireless cellular systems, but as our research shows, if the right techniques are used, the simulations can be substantially accelerated.

The feature of that makes models of wireless cellular systems most intriguing is the fact that one can identify processes of diverse natures intertwined over the same time scale. First, interactions between user and system can be viewed as discrete events in continuous time. These events happen at unpredictable times, according to no specific pattern. For instance: a realistic model allows a user to place a call at any point in time, rather than at, say, every five minutes. Second, the wireless system performs other activities at regular intervals, suggesting that internal control processes operate on a discrete time scale. The events representing these "system activities", which involve complex calculations, are performed with such frequency that if one simulates the model by time-stepping, the computational burden becomes exceedingly heavy.

Here lies the principal challenge in the simulation of wireless systems and the main topic of this dissertation: how can we make simulation time advance at a faster rate and produce results as accurate as those of time-stepping? The technique of *interval jumping* we propose, allows the simulation to identify situations when it is possible to avoid processing a potentially large sequence of these system events making time advance in leaps and bounds instead of little steps.

Having developed *interval jumping* to its full potential when applied to accelerating sequential simulations, we go further and build a parallel simulation with it. We do not develop a new synchronization algorithm, but instead make use of simple conservative techniques to parallelize our model.

1.1 Related Work

Several articles on parallel simulation of wireless systems have been published recently. The level of abstraction in the models used in these works varies according with the depth and completeness expected from the simulation studies.

Lin and Fishwick [32] introduce issues of parallel discrete-event simulation (PDES) and, at the same time, describe a detailed model for the simulation of personal communication systems (PCS).

Carothers *et al* [9] use models for teletraffic and simplified mobility under a static channel allocation (SCA) scheme to simulate, in parallel, realistically sized PCS networks. Later articles by these researchers [7, 8] carry on with a more detailed mobility model in order to study the effect of different call workloads and population movement patterns on the system's performance. All these studies were conducted using Time Warp, an optimistic parallel simulation paradigm [26].

Greenberg *et al* [22] identify a class of models for dynamic channel assignment (DCA) and present a conservative technique for accelerating the simulation on a massively parallel SIMD machine. Using a large cellular network model, experiments are executed on a 16K processor MasPar MP–1 achieving speedups of up to 120 with teletraffic simulation.

Our work differs from those mentioned above at different levels. One of the motivations behind our research was to construct a more comprehensive simulation model including tightly coupled components of wireless systems such as power control and channel allocation. With this model, which brings simulations closer to real physical systems, we devised a technique offering good potential for performance improvements and started studying issues related to its parallelization.

The well-versed reader in this area will notice that some ideas explored here are closely related to those presented in [29]. We share the same rationale for development of a parallel simulation model and agree that the natural partitioning strategy for this problem is to break it up by channels and not by subdivision of the simulated space. Each channel or pool of channels can be simulated by a different processor fairly independently of others, especially when adjacent channel interference is minimal. In addition to the new technique for acceleration proposed, our work differs from [29] by including cross-channel interference in the model.

1.2 Overview of the Dissertation

Our research has focused on the identification of the major bottlenecks of the simulation of wireless cellular systems and on the development of computational methods to overcome them. When models include components for power control and channel allocation, the simulation goes through a large number of state updates, regularly spaced in time. Not only are these state updates numerous, but they are also computationally expensive. The contribution of this thesis concerns the acceleration of these models; we approached the problem from two distinct fronts.

First, we investigated methods to reduce the computational cost of each state update. Having determined that the computation of interference in the model is the dominant factor in this cost, we investigated how this computation can be accelerated through the application of the Barnes-Hut (BH) algorithm, originally created for the gravitational N-body problem. We compared the results obtained with this approach to another method from the literature and our results indicate the most appropriate choice for different circumstances.

Second, we developed *interval jumping*, a technique that leads to the reduction of the number of state updates. In the context of the development of this technique, we constructed a framework based on dynamic programming that allowed us to specify MULTI, a hierarchical search scheme that can be used to identify intervals of simulation time where simulation time can be accelerated. We compared this search scheme with other simple policies and showed that its importance is tantamount to the scalability of the simulation.

The results we observed in both fronts are quite encouraging. The application of the BH algorithm demonstrates that interference computations can be accelerated as much as

1000 times relatively to the brute-force method. We also observed that the relative errors produced with BH depend on the system load and can be significantly higher (one order of magnitude) than those produced by truncated interference, an alternative method for interference computation. In what regards interval jumping, the simulation of a model of 10×10 cells, under a load that produces blocking probability of about 10%, can be accelerated by a factor as low as 26 and as high as 82, relatively to time-stepping.

The remainder of this section describes the general structure of the dissertation. Chapter 2 presents a brief summary of concepts in wireless cellular systems. There we introduce basic terminology with which one needs to be acquainted in order to develop comprehensive simulation models of these systems.

In Chapter 3, we cover several of the main points in this dissertation. We start out defining the simulation model adopted throughout this study and then proceed to describe two techniques to accelerate simulations of wireless cellular systems. The first one of these is concerned with the application of N-body problem algorithms to reduce the cost of interference computations. The second, the principal focus of our research, is *interval jumping*, an alternative way to promote time advance in the simulation that drastically reduces its execution time. We conclude the chapter with a description of the initial experiments that demonstrated the potential of our technique.

Next, in Chapter 4, we analyze the costs and benefits of interval jumping. We show that there is more to interval jumping that meets the eye. Although the technique proves to be highly efficient through naïve application, when we attempt to extract better performance, an important problem arises. This chapter shows in detail how non-trivial it is to determine intervals of such length as to make interval jumping fulfill more of its promise. Using dynamic programming, we develop a framework that is general enough to allow the efficient use of interval jumping with a variety of simulation models. We close the chapter reporting a series of experiments in which we derive models that are used by a search scheme intended to expand the length of interval jumps.

Chapter 5 discusses research problems we have identified in the context of *interval jumping*. Furthermore, the chapter presents several open problems in the simulation of cellular wireless systems.

Finally, Chapter 6 presents our conclusions, briefly summarizing the research contributions made with this work.

Chapter 2

Concepts In Wireless System Technology

This chapter establishes the basic terminology used throughout this dissertation. We present here several issues pertinent to the design of a wireless system, points which are of central importance to understanding the technology and, consequently, to the development of a realistic and comprehensive simulation model that meets our goals.

2.1 Radio Waves, Cells and Mobile Users

According to the Federal Communication Commission (FCC), Title 47, Part 22 of the Code of Federal Regulation, a cellular system is defined as:

> A high capacity land mobile system in which assigned spectrum is divided into discrete channels which are assigned in groups to geographic cells covering a cellular geographic service area. The discrete channels are capable of being

reused in different cells within the service area.

From this definition we can deduce the basic characteristics of a cellular radio system. A *cell* is a well-defined geographic area that offers communication services to subscribers through a *base station*. Mobile users sign up for service upon entering a cell and communicate via a full-duplex radio link with its base station. When a user leaves a cell, the network tries to forward the ongoing connection to the next cell visited without interruption of service. The sensitivity of receivers at base stations together with the power of their transmitters, determine the coverage area of the cell.

Keeping transmission power as low as possible is key in wireless systems. On the mobile's side, this maximizes the efficiency of battery usage (autonomy) and on the base station side, it minimizes electricity consumption. What is more important than these two factors, however, is the fact that low power enables the system to accommodate more users and to provide better quality of service (QoS).

Transmission power establishes a geographic area covered by the signal: since radio waves decay exponentially fast (an effect known as *path loss*), only receivers within a certain distance can successfully pick up the signal. However, the decay function has a long tail and when the areas of coverage of different transmitters overlap, the signals "interfere" with each other and can't be received as clearly as they were sent out. Making areas of coverage small, that is, using low transmission power, tends to reduce this effect and allows more cells to be packed within the same geographic region.

The drawback of low power is that QoS, which is defined as the ratio of signal strength to interference strength (signal-to-interference ratio or SIR), can degrade quite easily. Wireless systems, however, employ clever mechanisms to circumvent this problem by continuously adapting transmission power, so that only the minimum power necessary is used to achieve a desired QoS is used. The most important consequence of this parsimony of power usage is the fact that as coverage areas are kept small, neighboring cells are able to use the same frequencies concomitantly.

The notion of *reuse* is a point of utmost importance in the design of wireless systems, which has direct impact on scalability. The radio spectrum is divided into bands allocated for different purposes and the portion reserved for commercial wireless networks is small relative to its service demand. Frequency reuse is, therefore, a big step in achieving high efficiency from a scarce resource, but not the only one.

Most commonly, a wireless system allows more than one active user within each cell. To provide multiple simultaneous connections to the same base station, different schemes or *multiple access* technologies have been devised and they vary widely in how they utilize the spectrum and in the quality of connection they provide.

2.2 Multiple Access

Perhaps the most straightforward way to grant multiple users access to the radio space is by *frequency division multiple access* (FDMA). The spectrum is split into a number of bands or slots, each one wide enough to carry one connection. A channel, in this setting, actually refers to two distinct frequencies: one to carry information from base to mobile, the *downlink*, and another, the *uplink*, for the reverse direction. While channel, uplink and downlink are concepts perhaps more directly related to FDMA, the same notions still exist in other system designs.

Another option is to create partitions in time rather than in the frequency domain: *time division multiple access* (TDMA) allows each connection to utilize the full band, at maximum transmission rate, for a limited time. Subscribers are then scheduled to use the medium one at a time, transmitting only when their turn is up.

An alternative to TDMA and FDMA, *code-division multiple access* (CDMA) has been the technology of choice in many of the more modern networks [46]. CDMA uses spread spectrum signals; that is, the bandwidth employed is much larger than the bandwidth of the messages transmitted. The basic idea requires an arbitrarily chosen spreading code, a pseudo-random sequence or *pseudo-noise* (PN), that is applied to modify the data. To decode the message, the receiver must know the PN used in the encoding procedure.

The two predominant flavors of CDMA are direct sequence (DS CDMA) and frequency hopped (FH CDMA). These techniques differ in how the PN is used to spread the signal on the spectrum. Like TDMA, CDMA uses all available frequencies at the same time, which characterizes a spread-spectrum technique. Effectively, all users produce interference for one another, but since a unique PN is associated with each link, each user can distinguish the real signal from interference [48].

DS CDMA takes a digital data stream as input and “stretches” each value out: a 1 bit is replaced by the PN and a 0 bit by the 1’s complement of the PN. This has the effect of multiplying the bit stream by the bit length of the PN; for instance: a PN of length 64 and an input stream with a rate of 16 Kbits/s result in an output stream with a rate of 1,024 Kbits/s. The encoded stream requires a bandwidth 64 times larger than the original, but is more resilient to errors and can be reconstructed with higher accuracy at the receiver.

FH CDMA, on the other hand, uses the PN to make the carrier frequency hop around the band. The transmitter uses a certain frequency for a fixed pre-determined amount of time and then moves on to another frequency. The receiver, having been informed of the PN value, knows the sequence of frequencies that will be visited by the transmitter and can be tuned in synchrony to reconstruct the data.

By spreading the signal in a broad band, CDMA achieves high efficiency of spectrum utilization at the cost of increased complexity [48]. It also tends to reduce deleterious effects such as multipath fading and interference (discussed in detail in section 2.3).

As of now, there is no single global standard for wireless systems and thus different designs use different access methods. One can easily get confused with the many acronyms in the jargon used to refer to different schemes, but these seemingly obscure arrangements of letters are all related by their common goals: to multiplex the medium and to enable the carrier to provide services with a well-established standard of quality. In this dynamic and competitive market, from the consumer's point of view, quality translates not only to a clean, noise free connection, but also to how easy it is establish one.

When a subscriber attempts to place a call, there is a chance the system cannot support it and the request must be refused. There can be two different reasons for this denial of service:

- Radio resources are unavailable; that is, no base station in the mobile's vicinity has a free transceiver to establish a new connection, or

- Although physical resources do exist, due to the system load and propagation conditions, it may not be possible to admit one more connection because its effects on the system would degrade the QoS in the network beyond what is tolerable.

The main issue here is that providers of wireless services engineer their system so that it is not only "good enough" from the start, but that it can stay "good enough" as the subscriber base grows larger. An increase in load places more stress on the system and may require that cells be made smaller so that capacity is increased and quality maintained.

While many different alternatives to scale up the system may be conceived, one cannot count on the possibility that the band of spectrum allocated for commercial wireless services can be enlarged. A relatively small range of frequencies is available and the systems have to make do with them. Frequency reuse is a big step in the right direction, though not the only one. Later on we present the problem on channel assignment and show how it can be coupled with power control to improve spectrum utilization. Next, we look at the main obstacles that lie on the road to achieving quality in radio connections.

2.3 Interference, Noise and Other Monsters

Basically, anything that rides on a radio wave and is not real information (signal) can be said to be some form of interference or noise. There exist different forms of spurious signals at play in a wireless network and they must be well understood in order to be accurately modeled in a simulation. We now discuss these in more detail.

Consider a radio transmitter in an FDMA system. When tuned to a central frequency f, depending on the type of modulation employed, the signal sent out occupies a certain bandwidth, that is, a range of frequencies defined around f. An ideal transmitter would have a transfer function with sharp rectangular bounds as shown in Figure 2.1. That would mean all the output signal is strictly concentrated in the interval defined by the bandwidth.

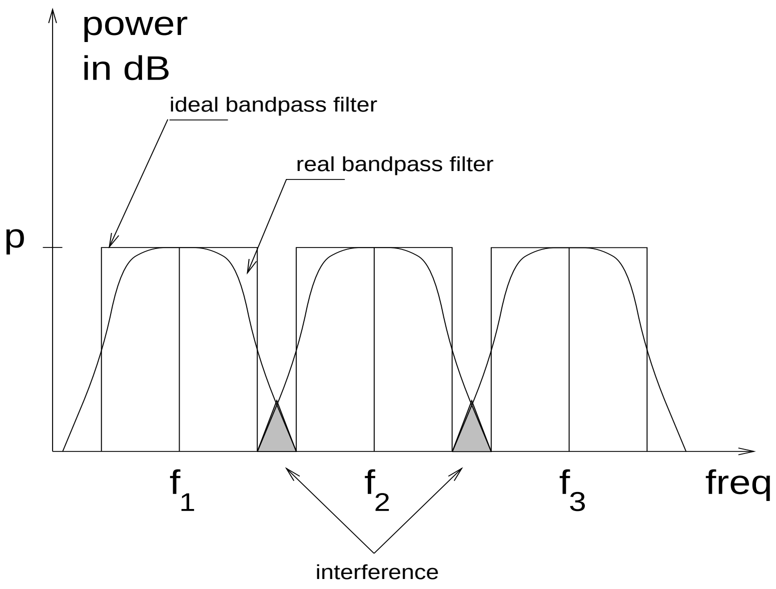


Figure 2.1: Filter transfer functions and adjacent channel interference in the frequency domain

---

However, such a transmitter is not realizable and what can be accomplished by bandpass filters is much less rectangular than the ideal.

As we try to pack the usable part of the spectrum with as many channels as possible, we may end up with adjacent channels which do not use disjoint intervals in the frequency domain. This characterizes a type of interference, depicted in Figure 2.1, called *adjacent channel interference*: part of the signal in one channel is garbled by the signal in a neighboring portion of the spectrum. The interfering signal is, of course, subject to the same decay with distance and thus may suffer severe attenuation until it reaches the receiver (the amount of disruption the spurious signal causes will greatly depend on how close the receiver/transmitter pairs are).

Another type of interference occurs when more than one transmitter uses the same channel and they are close enough for their coverage areas to overlap. What is signal for one transmitter is noise for the other: this is called *co-channel interference* and can be minimized or even completely avoided by careful channel assignment. It is possible to divide the available frequencies into disjoint sets: if set S_i is assigned to some cell c and we guarantee that no other cell $c^{'}$ within a certain distance uses the same set, there will be no significant co-channel interference.

A small example of a practical situation when interference imposes contraints of channel assignment is depicted in Figure 2.2. Consider two cells c_1 and c_2 and their immediate neighbors, the cells that form a ring around them (a macroring of radius one). Now, say the system can use a set of frequencies $S = \{f_0, f_1, f_2, \ldots, f_n\}$, where every f_i is adjacent to f_{i-1} and f_{i+1} in the spectrum. One can divide these frequencies into disjoint subsets (represented by different shadings in the figure) so that there is never a pair of adjacent frequencies in any of them. When each cell is assigned one of these subsets, we can ascertain that no adjacent channel interference will occur within those boundaries. More care has to be exercised when sets with neigboring frequencies are assigned to neighboring cells, however.

Now, to rule out the possility of co-channel interference as well, channel assignment has to be done even more carefully to avoid letting geographically close cells use the same frequencies. To a certain extent, this is a particular instance of the graph coloring problem, as illustrated by Figure 2.2. Low power and propagation decay constrain the coverage of each transmitter and so frequencies may be *reused* in some cell distant enough without creating interference.

While conservative partitioning and assignment of spectrum can perhaps totally elimi-

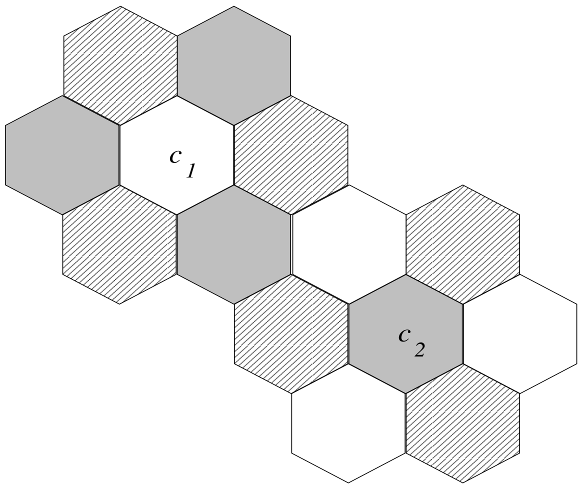


Figure 2.2: How interference affects channel assignment

nate the effect of interference, the risk of poorly utilizing the medium is potentially great. Many frequencies could be denied to an area with high load because they could possibly cause interference in a vicinity with little or no activity at all. Later on we discuss how an adaptive channel allocation algorithm may allow interference to exist and still achieve good QoS.

Interference is neither the only nor the worst effect that can degrade signal quality; others exist and can be mitigated, more or less easily and only to a certain extent, with the use of appropriate measures. A notable problem arises when a signal bounces off from different reflective surfaces arriving at the receiver after traversing different paths: this is called *multipath fading*. Random changes of phase in each different path can cause destructive addition at the receiver antenna. Also, because of different lengths for each transmission path, the signal arrives at the receiver in multiple replicas dispersed over time. When all these effects are added up, the information transmitted may reach the receiver distorted beyond all hope of being understood. Spread spectrum techniques, such as FH CDMA, are known to mitigate this type of problem.

Another effect that can attenuate radio signals, *shadowing* can also play an important part in wireless cellular systems [46]. As mobiles roam around, it is possible to have obstacles temporarily interposed between transmitter and receiver, creating random changes in the propagation conditions. The offending obstacles can range from natural terrain features, such as hills, to large buildings or even trucks or cars.

Last, but not least, *noise* is another effect that can substantially degrade the quality of communications in a radio system. We can divide sources of noise into two different categories: environmental and non-environmental [28]. The first of these encompasses atmospheric, human-made, galactic and solar noise. The second category consists of the *internal noise* electronically generated inside each radio device, and must also be considered. Semi-conducting devices are known to radiate a measure of thermal noise, which gets superimposed to the signal received at the antenna. The quality of the receiver determines not only its sensitivity, but also the level of internal noise generated. While this type of perturbation in the system is small, relatively to other factors, it must still be taken into account by a system dedicated to maintaining a minimum standard of quality.

2.4 Teletraffic

In a wireless system, users move about the service space placing and receiving calls, shifting the load from here to there as they go. To a certain extent, the system must keep track of users' positions so that incoming calls can be properly routed. Consider a user who usually operates in New York City, but who has traveled to Miami. Calls addressed to this user will first be routed to Miami using some regular long distance telephone and then to the user's receiver by the local cellular network. A similar problem exists when this user originates calls: the local cellular network is used but billing information has to be propagated to the accounting center in the area of subscription.

It is important to describe the interactions between mobile radio units and base stations. Base stations are usually located at the center of the cells and interact between themselves via wired network or, with the regular wired telephone system, with mobiles via radio and perhaps even with satellites. Mobiles choose a base station as their point of contact and communicate exclusively with it until either the call terminates or they move out of its coverage area. The choice of base station depends on a criterion of *closeness* that doesn't necessarily correpond to geographical distance; it is often more desirable to connect to the one base in the neighborhood that offers the cleanest signal path.

The first step for any user communication is, therefore, the selection of a base to operate with. This process begins with the mobile transmitting a pilot signal, which is received by the bases in its neighborhood. The bases communicate amongst themselves and appoint the one that received the pilot signal at greatest power to handle the mobile. What happens next is highly dependent on the multiple access technology and discussing those details is beyond our scope. Although FDMA is an outdated technology with well understood disadvantages, for the purposes of developing this study, it is fairly adequate. From this point on, we refer to the concepts of channels and frequencies as in they exist in the FDMA context. The relevance of our research lies in the fact that we have developed techniques at the simulator's infrastructure level, which can be modified to work with modern day wireless technologies, such as CDMA.

In the case that no base with a free channel can be found in the mobile's vicinity, the call is immediately *blocked*, that is, refused by the system. Just like in the wired phone network, a call request is never queued for service at a later time. The user is simply forced to retry establishing the connection by dialing again. When a request is sucessfully processed, the communication link between base and mobile is established and maintained until the user leaves the service area of that cell.

In the event that a call is in progress when the user crosses a cell boundary, to avoid interruption in service, the call must be *handed-off* to a newly chosen base, which may well be the one in the center of the cell just entered. If that base has free radio resources to carry the new connection, service goes on seamlessly, otherwise the call is *dropped*.

While this teletraffic model is fairly simple, the key point to observe is that the mobiles' interactions with the system occur at random points in time. Essentially, the system cannot predict when a call request will arrive nor how long a call will last. As we discuss in more detail when we present our simulation model in Section 3.2, this type of behavior is characteristic of event-driven systems. We show next that a few components of wireless systems require time to be viewed as continuous.

2.5 Power Control and Channel Assignment

SIR is the most basic measure of a system's quality of service and is commonly used as a synonym for QoS. It does not, however, fully describe how reliable and accessible a system is.

Call attempts may be blocked and ongoing calls may be dropped and, thus, it is important to quantify the likelihood of observing these events. Commonly, these two different parameters are referred to as *blocking probability* (P_b) and *dropping probability* (P_d) and, together with SIR, they give a much better idea of how good a wireless network is. Another meaningful performance measure, *grade of service* (GoS) employs these two probability values to define a value used to characterize the quality of the network in one number [46].

In order to make the most out of a wireless network while offering subscribers a good GoS, system designers use components that make dynamic adjustments based on service demand. Perhaps the two most important of these are power control and channel assignment mechanisms.

The importance of power control stems from the discussion introduced in Section 2.1, where we point out how crucial it is to transmit signals at the lowest power possible. A power control mechanism works hard to achieve this goal while maintaing SIR in all connections around a target value specified by the carrier company. Its basic function is to continuously sense noise levels and adjust power levels accordingly. When it is determined that signal quality in a connection cannot reach the desired standards regardless of the amount of power used, the power control mechanism may tag the call as a hopeless case.

However, when a connection is deemed too noisy in a certain channel, it can perhaps be moved to another one where it reaches the desired SIR. The system may choose to give up on it after having tried other free channels where it could possibly be carried to completion. In this sense, power control and channel assigment mechanisms can cooperate towards the single goal of keeping GoS high.

A channel assignment mechanism does exactly what its name implies: it deals with teletraffic choosing a free channel to accomodate a new call request, moves ongoing calls from noisy to potentially good channels, and hands off calls from one cell to another. All of this takes place in the context of satisfying constraints to minimize or totally avoid interference, as discussed in Section 2.3.

Some of these mechanisms are based on a central intelligence that collects data about the system's current state and takes action based on what it finds. In theory, these schemes could provide the best possible performance, but they are not practical. Distributed mechanisms, on the other hand, may be much easier to implement, but work with local information that may not lead to optimal solutions.

In the next chapter, as we present a formal simulation model, we discuss in detail a specific locally autonomous mechanism that implements both power control and channel assignment in one single system component.

2.6 Chapter Summary

This chapter introduces basic terminology and concepts on wireless systems. Some of the key terms to remember are:

Signal to interference ratio (SIR): ratio of signal strength to interference strength at the receiver site.

Quality of service (QoS): a measure of the goodness of connection a user experiences in a system; most commonly defined as SIR.

blocking probability (P_b): likelihood that a new call request is immediately rejected by a communication system.

dropping probability (P_d): likelihood that an ongoing call is terminated by the network rather than by the subscriber on account of poor SIR or unavailability of radio channel when an inter-cell hand-off occurs.

Grade of service (GoS): parameter that describes the accessibility of a system as a function of P_b and P_d.

receiver noise: thermal noise generated by semiconductor devices inside the receiver.

co-channel interference: signal generated by transmitters on the same channel other than the one to which the receiver is directly linked.

adjacent channel interference: spurious signal generated by transmitters using frequencies close to the one currently employed to connect a transmitter/receiver pair.

channel allocation mechanism: component of a wireless system in charge of assigning channels to establish connections betweem mobiles and base stations.

power control mechanism: component of a wireless system that manages transmitter power levels in a network striving to achieve a target SIR while, at the same time, keeping power as low as possible.

Chapter 3

Simulation the Faster Way

In this chapter we define a comprehensive simulation model for wireless systems and present, in detail, a specific algorithm for power control and channel assignment. Next, we introduce the technique of *interval jumping* and show how it can be applied to accelerate sequential simulations. Finally, we present the results of an experimental evaluation of the technique.

3.1 Foreword

Simulation has been at the core of works related to the engineering aspect of wireless systems. It's widely employed to first evaluate the correctness of protocols and network design and, ultimately, for performance prediction when analytical tools can't provide any answers.

With the explosive demand for wireless services, the systems of interest turn out to be very large and thus their simulation becomes problematic. The basic rule of thumb for model construction states that the more complex models are, the greater computational power they require. For this reason, much work in simulation of wireless technology has been focused on specific system components such as channel allocation algorithms, network capacity, etc., or on how the systems respond to different scenarios.

Pioneering work in the field using simulation has been carried out by the very people who were developing the technology. Part of the former AT&T Bell Laboratories, Lucent Technologies, was highly influential in this area. Leonard Cimini, Jr., Gerard Foschini and Zoran Miljanic have become important names in the growing community and have published key works. We highlight a few of them here:

- A study of network capacity for linear and planar arrays of cells [12] that working with a simplified environment model computes saturation measures for a network of microcells.

- An investigation of how user mobility affects network capacity [16], which estimates how many additional channels are required to maintain a certain blocking probability under different user mobility models and fixed channel allocation.

- The development of distributed autonomous algorithms for channel assignment and power control [15, 17], where a decentralized algorithm that exploits the natural coupling of the two system components is proposed and evaluated.

- Performance studies of dynamic channel allocation in microcells [11], which shows that impractical centrally administered fixed allocation algorithms do not perform much better than their dynamic counterparts.

Many other examples of what I dare call "circumstancial work" on simulation of wireless systems can be found in the literature. These simulations were simple engines constructed to prove a certain concept and no more than that. Researchers studying propagation models, protocols, and algorithms have had many times to write their own code to verify their propositions and so the simulation was not the object of the study, only a means to an end.

Another vein of work in this area began with the observation that testbeds for wireless system components were in high demand. Notable examples of this trend come from WIN-LAB, at Rutgers University. MADRAS [10] is a sequential simulator which targets mobility and dynamic channel allocation studies. WiPPET [39] extends the idea much further, providing an environment for parallel simulation of various types of wireless networks. This virtual testbed is implemented in the TeD/C++ [42] framework over the Georgia Tech Time Warp (GTW) [13] optimistic parallel simulator. Following the TeD philosophy of offering base objects that can be interconnected to form complex models, WiPPET enables the reuse of design patterns developed by experts in the area and also frees the simulationist of the burden of dealing with the low-level details of engine building.

More recently, the community has identified the need for investing effort in the study of the specific computational problems that arise from the simulation of wireless systems. In order to produce results representative of a real system's behavior, model complexity and system sizes lead to simulations which are extremely computation-intensive. Often enough, these simulations require the use of high performance multi-processor computers in conjunction with sophisticated problem-specific techniques in order to execute in reasonable time.

As products of this new line of research, while not immediately adequate for widespread use by wireless systems engineers, these simulators lay the groundwork for an improved generation of testbeds. In this category, we find efforts such as [22], which unleash the power of massive parallelism to study algorithms for dynamic channel assignment. Also, [40], a portion of this dissertation, shows that under a different treatment, comprehensive computational models of wireless networks may execute much faster.

The rest of this chapter introduces the reader to the complexity of this simulation problem and starts to present our solution to accelerate its execution.

3.2 A Simulation Model

Wireless systems subdivide the area of coverage into smaller regions, or cells: while, in space, wave fronts of radio signals propagate as concentric spheres, in a bi-dimensional simulation model, they are commonly represented by hexagons or squares.

In our model, we define the geographical space as a plane, where all base stations and mobiles are contained. All cells have hexagonal shape and the same dimensions. Furthermore, at the center of each, we assume there exists one base station transmitting with an omni-directional antenna. The space is arranged as a torus and so we can work with a simulation does not experience edge-effects [33].

To impose periodic boundaries on this hexagonal world, we assume that it must always have an odd number of rows. This allows us to merge top and bottom rows into a single one. We can do a similar trick with the columns: if there is an even number of them, we can fold the space along the y-axis and have the edges fit together like in a jigsaw puzzle. As illustrated in Figure 3.1, effectively, the simulation space then fits inside a rectangle.

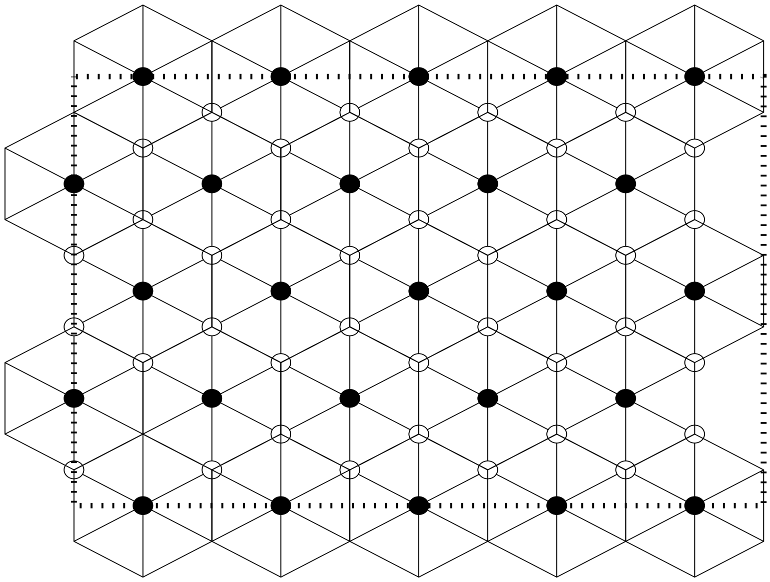


Figure 3.1: Street layout in a 5x10 simulated world

3.2.1 Radio Propagation

It is well known that, in a vaccum, electromagnetic waves are subject to an exponential decay with distance. That is, if p_t is the transmitter power and if the transmitter and the receiver are separated by d units of linear distance, received power is given by

$$p_r = p_t \frac{1}{d^2}. \tag{3.1}$$

In our model, we must consider that propagation decay is much faster than in a vaccum due to atmospheric conditions and, also, to possible obstacles in the propagation path. We can generalize this power law to be inversely proportional to d^α and then use $\alpha \geq 4$, as is most common in radio studies.

For each propagation path of interest, we can simplify the way we describe received power by writing $p_r = a\, p_t$, where a is the *attenuation coefficient*. One can go as far as defining different values of attenuation according the instantaneous positions of receiver and transmitter to reflect detailed terrain features. Although our model does not go down to this level of refinement, we consider that a can be made a function of time to account for the possibility of transient effects in the propagation path. These time varying effects, known as *fading*, can assume several forms, but for our purposes we consider only shadow fading, which is directly related to the amount of clutter in the propagation path and is commonly represented by a lognormal random variable [17, 29, 43]. Considering all these points, we write $a(t)$ as

$$a(t) \;=\; \frac{\mathrm{lognormal}(x, y, t)}{d(t)^{\alpha}},$$

where distance $d(t)$ is a function of time to model mobility and $\mathrm{lognormal}(x, y, t)$ follows the conventional definition of this kind of distribution, that is, it is equal to $\exp(x + yZ)$ with Z being a $Normal(0, 1)$ random variable, sampled at simulation time t.

It is important to point out that in our model, we need to address attenuation coefficients from two different points of view. First, let i denote a transmitter/receiver pair, where one is a mobile and the other a base station. Next, let $a^c_{ii}(t)$ describe the conditions on the line-of-sight path between the two elements of pair i on channel c. Conversely, $a^c_{ij}(t)$ represents the attenuation on the path beween a transmitter and a receiver on different pairs i and j, both tuned to channel c: this coefficient reflects the attenuation in the path of cochannel interference.

Through careful manipulation of these coefficients, it is possible to define the propagation conditions for the entire simulated world. In certain cases, it may be interesting to predefine fixed values to reflect a basic geography and on top of it add dynamically changing factors to model random effects such as shadowing or multipath fading.

As a final word on modeling the propagation of radio signals, we must say that although the inclusion of an up-to-date multiple access technology in the model could have added relevance, in this study, we restrict ourselves to considering the older FDMA exclusively. Motivated by the physical model in FDMA, throughout this dissertation, we work with the conceptual notion of *channel*: the medium capable of implementing a full-duplex connection between a user and the server at the center of a cell. Each channel can be seen as composed of two radio links: one that takes signals originating in a mobile transmitter to a cell server, called the *uplink*, and another, called the *downlink*, to carry signals from a cell server to a mobile receiver. Channels may or may not be orthogonal (contain cross-channel interference).

It will become obvious later on that the challenges of efficiently simulating wireless systems are multiple. Our goal, with this work, is to lay down a solid foundation upon which more complex structures can be built. Given the intricacies of timing in current technologies such as CDMA, we have chosen to work with FDMA, instead, because it allows us to concentrate on the major bottlenecks of the simulation rather than in particular details of the multiplexing mechanism. Although the technique we have developed in this research applies directly to simulations of analog cellular systems, it can be adapted for use with simulations of CDMA digital cellular systems, which rely on even more frequent power control.

3.2.2 User Mobility

Among the many possible models of user mobility, we have chosen to impose a "street" layout on the simulation space. Figure 3.1 shows the grid to which mobiles are restrained. Black dots mark the center of each cell and white dots mark points of intersection on vertices of the hexagons: at all these points a mobile can choose a new direction of movement with equal likelihood. The speed of travel is constant on a straight line segment between two dots, but a new one may be chosen at every intersection point.

In this context, we define the *trajectory* of each mobile as an ordered collection of tuples of the format (t, p_s, p_f), where

- $p_s = (x_s, y_s)$ and $p_f = (x_f, y_f)$ indicate the segment's start and endpoints, respectively, and
- t is the time the mobile takes to travel from p_s to p_f.

Base stations will often lie in the trajectory of a mobile causing a peculiar situation to arise: no user should be allowed closer than one unit of linear measure to the center of the cell. Since attenuation coefficients have a factor inversely proportional to the distance d between transmitter and receiver, a value of $d < 1$ would cause attenuation to become gain instead, creating a physical inconsistency in the propagation of radio signals.

We avoid this problem by introducing a physical inconsistency at another level, one to which the simulation should not be as sensitive. In our mobility model, a user experiences a spatial discontinuity when approaching the center of a cell. A mobile heading straight for the base station is never allowed to get "too close": it is simply moved over and across it, breaching two units of distance instantaneously.

Choosing the appropriate units, the effect of this artifice can be substantially reduced. Say that, in our model, distances are represented in centimeters and that, typically, mobiles move at 1m/s or faster: the time taken to traverse the critical 2cm across a base station would be 0.02s, which may be small enough to be treated as negligible. Changing the units to millimeters would further diminish the problem by an order of magnitude.

3.2.3 An Algorithm for Power Control and Channel Assignment

Assume that for every channel c in a wireless system, we can label each pair of connected base station and mobile using an integer number i. For some time t, let $p_i^c(t)$ represent the transmitter power levels at the mobile in pair i (the uplink) and let $\overline{p}_i^c(t)$ represent the transmitter power level at the base (the downlink). We say that sets of values $\{p_i^c, \forall i\}$ and $\{\overline{p}_i^c, \forall i\}$ are *power assignments* or *power vectors* for channel c.

Considering that all channels in the system are orthogonal—that is, there is no adjacent channel interference—we can view each channel as a self-contained, independent system. Simulating a system with multiple channels is then equivalent to having multiple simulations of one-channel systems which interact only when a connection is switched from one channel to another. When and how this happens is explained later in this section.

Now, for some channel c and some instant t, the best power assignment is one that minimizes total transmission power at the same time as achieving ρ, the target SIR, in the largest possible number of channels. This is clearly an optimization problem that can be defined more formally as:

- <u>Decision variables:</u> power assignments $p_i^c(t)$, $\overline{p}_i^c(t)$ for all mobiles i in the simulated universe.

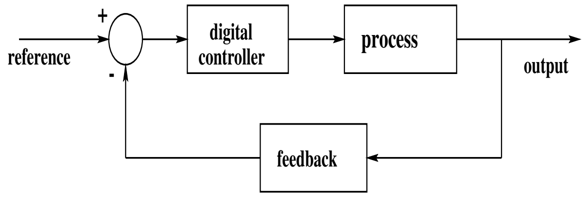


Figure 3.2: Block diagram of a digital feedback control system

- Goal: minimize power allocation while achieving a target SIR ρ on all connections.

- Constraints: transmitter power levels cannot exceed a maximum value M.

The distributed nature of the problem indicates that, ideally, one should seek to solve it using an algorithm based only on locally available information. Algorithms based on global information are extremely costly and impractical. Control switches used for centralized processing are not only a bottleneck for performance, but also a hindrance for the scalability of a cellular network. Therefore, research efforts have sought to produce methods of power control and channel access [17, 15, 24] that work exclusively with local measurements of SIR. In this fashion, level adjustment decisions can be made at each and every transmitter independently of any central intelligence.

As microprocessors got less and less expensive, digital control systems became almost ubiquitous and, nowadays, much of what used to be built out of analog circuits has been replaced. Digital controllers, however, do not work on a continuous time scale; the variables of interest are sampled at regular discrete instants, every Δ time units, and fed back to the control module (see Figure 3.2). The controller's output changes *only* when a new sample is acquired and processed. Looking at how power control is implemented in wireless networks, we see that it exactly follows the general principles of a digital control system.

Every Δ time units, the state of the system is inspected and transmitter power levels adjusted to meet the desired SIR. The power assignment changes and a new state is produced in accordance to the specification of the optimization problem. The choice of Δ has a direct impact on performance. Smaller values make the controller actuate more frequently, thus closely tracking the solution to the problem. Larger values reduce the frequency of computation required to produce new outputs, however Δ is bounded from above by Nyquist's sampling theorem [25]. Intuitively, the system cannot be kept under control if the process' state is not inspected often enough so that the controller can adapt and take the appropriate actions.

The actual value of Δ depends on complex mathematical analysis and engineering decisions, which invariably result in numbers small enough to pose a challenge for the system's simulation. The difficulty lies in the fact that if one is to follow the most naïve and straightforward way to implement the simulator, time advances very slowly. Imitating reality, simulated time progresses in very small, regular *time-steps*, each requiring much computation. To understand the magnitude of the problem, we introduce now an algorithm for distributed power control and channel allocation due to Foschini and Miljanic [17, 15].

Let ρ be the desired SIR for all connections in the system and $v_i^c(t)$ be the instantaneous value of total noise and interference over a receiver i on channel c. Consider the transmitter linked to receiver i and let the SIR at the receiver's location at time t be defined as

$$\rho_i(t) \;=\; \frac{a_{ii}(t)\; p_i^c(t)}{v_i^c(t)}$$

Foschini's method consists of looking at the system's state at a discrete point in time and then increasing or decreasing power levels according to the difference between $\rho_i(t)$, the current SIR, and the target SIR, ρ. This is formally expressed in the equation below, where β is a positive proportionality factor:

$$\dot{\rho}_i(t) \;=\; -\beta\,[\rho_i(t) - \rho] \tag{3.2}$$

The method is based on the premise that if user i has no control over $v_i^c(t)$ or any idea of how it evolves with time, its power level should be updated *as if* $v_i^c(t)$ *were not going to change*. The assumption that the power emanating from sources of noise remains constant while the power of each transmitter is adjusted allows for locally autonomous decisions. With this in mind, we can then write a "surrogate" derivative of $\rho_i(t)$ as:

$$\dot{\rho}_i(t) \;=\; \frac{a_{ii}(t)\,\dot{p}_i^c(t)}{v_i^c(t)}$$

Now, substituting this into (3.2), solving for $\dot{p}_i^c(t)$ and simplifying, we have

$$\dot{p}_i^c(t) \;=\; -\beta\,\left[p_i^c(t) - \rho\frac{v_i^c(t)}{a_{ii}(t)}\right] \tag{3.3}$$

which can be computed based solely on local measurements of $v_i^c(t)$. If there exists a vector $p^*(t)$ for which the desired SIR is attained, this algorithm will drive the power level $p_i^c(t)$ to match $p^*(t)$ no matter what its initial value is. Starting with an initial power assignment and applying (3.3) iteratively, $p_i^c(t)$ eventually converges to p^* with speed dictated by the value of β. Convergence is exponentially fast and it has been shown that $\beta = 1$ provides the fastest convergence [15].

Since the computational implementation of this algorithm works on discrete time, equation (3.3) must be rewritten as a finite difference equation:

$$p_i^c(k+1) \;-\; p_i^c(k) \;=\; -\beta\,\left[p_i^c(k) - \rho\frac{v_i^c(k)}{a_{ii}(k)}\right] \tag{3.4}$$

This means that at every point of discrete time k, in order to update power levels to satisfy the required constraints, it is necessary to have a measure of $v_i^c(k)$. In the physical implementation of the system, this is readily available from instruments. In a simulation, however, after every time increment of size Δ, the quantity must be re-computed. Since these calculations are inherently slow and done very frequently, this becomes a major bottleneck.

This power control algorithm guarantees that if a solution that satisfies all SIR requirements exits, it is eventually found. Note, however, that there will be instances when not all users will be able to achieve the desired ρ in both uplink and downlink frequencies. When this happens, the system must take actions to switch the connection to another channel or, perhaps, even to stop carrying it.

The situation is analogous to what could happen at a cocktail party:

> Suppose there is one big room where a number of people are paired up and engaged in lively conversation. Consider one of these pairs: these people are interested exclusively in what each one has to say and the conversation of all others is noise to them.
>
> If there is too much noise, they must speak louder in order to keep their conversation going. This increases the total amount of noise in the room and

causes other pairs to speak louder, too. The net effect is that soon everyone would be screaming at the top of their lungs! When two people are screaming as loud as possible and still cannot convey their ideas to each other, then it is certainly best that they give up talking. The overall noise in the room drops and some people may then be able to carry an enjoyable dialogue.

Cellular radio offers a choice of different *channels* for these communications. Referring back to the party analogy, we can imagine that the guests can move around to different rooms. When the environment gets too noisy, they can seek a quieter place to talk. The same can be achieved when power control and channel assignment are coupled in one algorithm: if noise on a given channel becomes too high, users can be moved to other channels with lower noise levels. The decision of moving a call to another channel or simply dropping it is based on the fact that if the transmitter power level is at the maximum and the SIR is still below ρ then there is no point in going on. Foschini's Local Autonomous Dynamic Channel Allocation (LADCA) algorithm, presented in Figure 3.3, embeds power control in the same picture.

Conceptually, the algorithm is very simple. At each instant of discrete time or *time-step*, transmitter power levels are recomputed throughout the system. After transmitters are adjusted in reaction to noise levels, the algorithm checks for channel allocation decisions that need to be made.

Each channel in the system can be in one of three states: *inactive*, *probe* or *service*. Once a new call request is received, a previously *inactive* channel is selected and marked as being in the *probe* state: for a brief period of time the system evaluates whether an acceptable QoS can be obtained. If an arbitrary number (GOOD) of consecutive time-steps

```
algorithm LADCA
begin
  for (all channels c) do
    if ((c.state == probing) &&
    ((c.uplink.power == max_power) | | (c.downlink.power == max_power)))
      if (no unvisited free channels)
        call request is blocked
      else
        switch probe to an unvisited randomly chosen free channel
    if ((c.state == probing) &&
    ((c.uplink.power < max_power) && (c.downlink.power < max_power)))
      if (SIR satisfied on c.uplink and c.downlink)
        if (++c.num_good == GOOD)
          c.state = service
      else
        c.num_good = 0
    if ((c.state == service) &&
    ((c.uplink.power == max_power) | | (c.downlink.power == max_power)))
      if (++c.num_bad == BAD)
        if (no free unvisited channels)
          drop call
        else
          move call to an unvisited randomly chosen free channel
    if ((c.state == service) &&
    ((c.uplink.power < max_power) | | (c.downlink.power < max_power)))
      c.num_bad = 0
end LADCA
```

Figure 3.3: Foschini and Miljanic's LADCA algorithm

go by and the target SIR can be met on uplink *and* downlink, the call enters *service* state. During this time, if a transmitter power level reaches its maximum and the SIR cannot be achieved, the call is switched to a different inactive channel, where the probing process is resumed. When a call in probing has visited all free channels without success, it is *blocked*.

From the user's point of view, this interaction happens in the interval of time that one would wait for a dial tone. A blocked call request means that the user is not offered a chance to dial and must hang up and try again. This reflects a common aphorism among the carriers: "it's better to offer no service than to offer bad service".

Once a call enters *service*, the algorithm becomes less aggressive. If an arbitrary number (BAD) of consecutive time-steps happen and the SIR couldn't be achieved even with transmitters at maximum power, then the call is switched (in the same state) to an inactive channel. If all available channels have been visited, the call is *dropped*. When a time-step with good SIR is observed, the algorithm restarts counting the bad ones. This fact illustrates another principle advocated by carriers which says that although offering bad service is not desirable, a call should only be interrupted as a last resort.

The LADCA algorithm exemplifies a tight coupling of power control and channel allocation. Both in the simulation and in reality, these two system components exhibit a stimulus/response interaction with teletraffic. A new call request (NCR) comes in and adds extra load to the network: from that moment on, after every Δ, the noise on the receivers is evaluated and transmission power corrected accordingly. Later on, if the channel is found to be too noisy, a channel switch (CS) is attempted and if it fails the call may be blocked (CBLK) or dropped (CDRP).

Figure 3.4 depicts a possible sequence of events for a call. Events denoted by HAND and COMP, respectively, correspond to inter-cell hand-off and call completion. Note that between two consecutive events originating within the teletraffic sub-model (in this case, NCR, HAND and COMP), there can be a large number of time-step events.

As we show in the next section, time-step events are handled by costly computations and situations will arise when it is not necessary to process each and every one of them. When these circumstances are detected, the simulation can progress faster by replacing

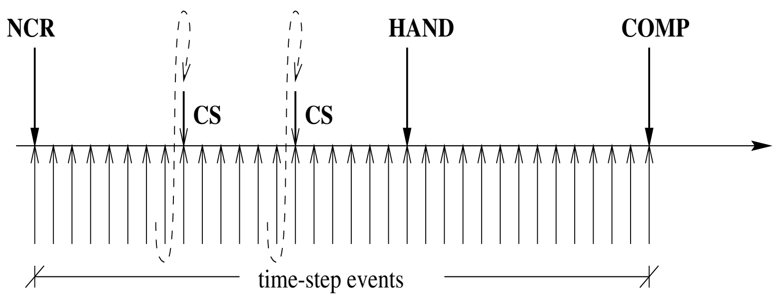


Figure 3.4: Simulation events associated with a typical call

many consecutive state updates by a single one. The key point to observe is that time-step events may cause other events, such as CS, CBLK and CDRP. These events resulting from a state update affect channel allocation and the teletraffic sub-model, and, ultimately, have the power to change the very statistical quantities the simulation is in charge of producing. For this reason, this faster simulation scheme has to be applied very deliberately, so that no events triggered by state updates are ever ignored.

Next we analyze the major component in the cost of time-stepping and present ideas to alleviate this burden.

3.3 Reducing the Cost of Interference Computation

The time-stepped simulation of our model can be outlined by the algorithm in Figure 3.5, where Δ is a time increment small enough to guarantee power control convergence. Note that the first two operations in the loop iterate over all elements in the set of channels defined for the system. Whenever time is advanced in the simulation model, the computational

```
algorithm TimeSteppingSimulation
begin
   while (! simulation terminated) do
      compute noise
      update power levels
      apply LADCA algorithm
      update user positions
      advance time by Δ
   end while
end TimeSteppingSimulation
```

Figure 3.5: Algorithm for time-stepping simulation

---

costs incurred are large because the asymptotic complexity of each step is a function of the number of mobiles.

Say we have a system with C channels and N mobiles. Given an even distribution of transmitters to channels, the per-mobile cost of interference computation is $O(N/C)$. Consider a mobile-base pair j, as in Figure 3.6. If ν is the receiver's thermal noise, the total interference over a receiver in j is given by the summation of signals produced by transmitters $i \neq j$ on the same channel c:

$$v_j^c(t) = \sum_{i \neq j} a_{ij}(t) p_i^c(t) + \nu$$

Since this computation is carried out for all receivers in the system, the total cost is $O(N^2/C)$.

We are interested in simulating large systems with many cells, many channels and many users, thus the computation of interference noise becomes a bottleneck. The approach described above can be characterized as *brute-force* and we can look for efficient methods to approximate those same results with little error.

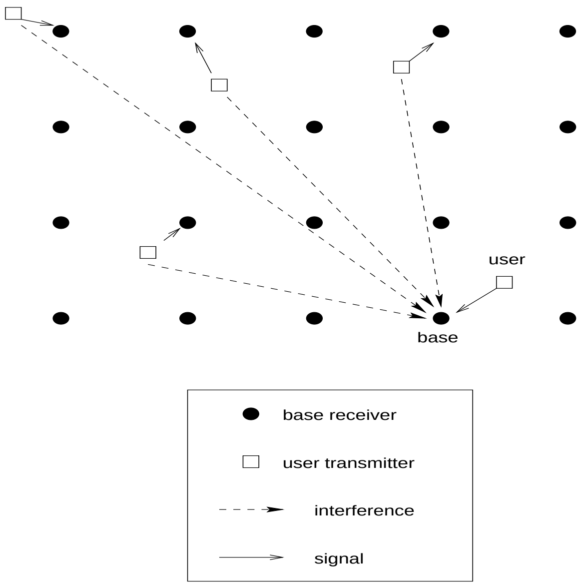


Figure 3.6: Interference noise in uplinks

3.3.1 Using N-body Algorithms

Originally, the N-body problem was defined as the simulation of a system with a large number of self-gravitating bodies. Each element would be a planet, galaxy or some object with a known mass subject to the laws of gravitation. Given the mass, initial position and velocity of each body, the goal is to determine the trajectory of each one as time progresses.

The first step in the process is to compute the total gravitational force over each body, which is a summation of pairwise interactions with every other element in the system. Once force has been calculated, the acceleration vector can be obtained by Newton's second law and the trajectory of each body computed by numerical integration over time.

The force between two particles with masses m_1 and m_2 is proportional to the product $m_1 m_2$ divided by the square of the distance between them. To compute the evolution of trajectories in time, one must look at "snapshots" of the system at each time-step and calculate the total force on each body assuming that *no interactions change during that* Δ. Note that this Δ has to be small enough so that no big changes in acceleration occur in the interval, which would deteriorate the quality of the trajectory approximations.

The similarities between interference noise computation for power control and particle interactions in the N-body problem are evident:

- one must compute interactions between a large number of elements
- interactions are defined by a power law with the distance between elements
- the problems are described by differential equations in time which can be solved numerically, resulting in approximations with quality dependent on the size of the integration step.

From this analogy we see that methods to solve the N-body problem can be easily adapted to compute interference noise in the simulation of cellular systems. The many algorithms for N-body simulation [2, 4, 5, 23] in the literature have one point in common: the use of a clever data structure that allows a drastic reduction in run-time. The time complexity can be reduced from $O(N^2)$ to $O(N \log N)$ or even $O(N)$, which represents a major improvement over the naïve algorithm for large N.

The data structures, usually quad-trees for the two-dimensional case and oct-trees for the three-dimensional case, implement a simple concept. Suppose we are computing the total interaction over a certain particle and want to consider the effect produced by a distant cluster of particles. If the distance d that separates the cluster and the particle is sufficiently large, we can come up with a good approximation considering that the whole cluster can be substituted by one single particle located at its center of mass with the total mass of the cluster. Instead of computing the interaction of each individual element in the cluster with the particle of interest, we compute only one interaction with the imaginary particle. This is called the *monopole approximation* and represents the essence of the Barnes–Hut (BH) algorithm [4] that we discuss from here on.

The monopole approximation, although good for its numerical simplicity, has the drawback of very restricted accuracy. In Newtonian gravitation, it corresponds to the assumption that the particles are all symmetrically arranged around the cluster's center of mass. The less symmetry we observe, the greater the error of the monopole approximation is. When high accuracy is a goal and particle distribution in a cluster is not symmetrical, a more complex expression to approximate total interaction must be used.

The summation of interactions at one point in space can be expanded into an infinite Taylor series: the first term is the monopole, the second is the dipole, the third is the quadrupole and so on. The dipole vanishes about the center of mass and thus the second term in the series, actually, corresponds to the quadrupole moment. In this series, each moment higher than the monopole corrects the approximation for the "shape" of the cluster. For practical computational purposes, when evaluating the summation of interactions, the number of terms in the series must be limited. Clearly, the number of terms considered is directly related to the error in the approximation.

The BH algorithm partitions the simulation space into *cells*, which are squares for 2-D simulations and cubes for 3-D simulations. Each cell can contain at most one particle: when more than one is placed in a cell, it is recursively subdivided into smaller cells of same dimension until each particle occupies its own cell. This defines a hierarchical organization of space; once a cell is subdivided it becomes the parent of the smaller cells it contains.

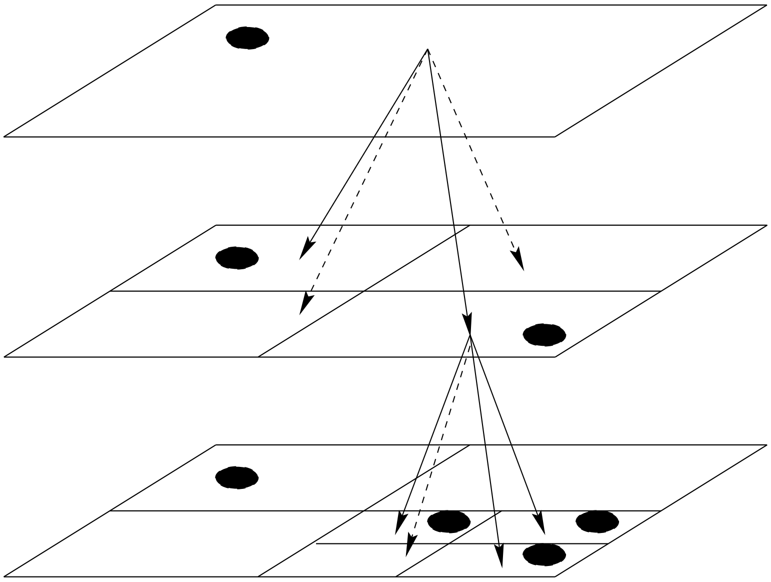


Figure 3.7: Organization of quad-tree used in BH algorithm

Each cell is mapped to a node in a tree (quad-tree for 2-D and oct-tree for 3-D) and each node contains a number of pointers, a mass value and a point coordinate.

The tree is constructed by a recursive procedure that starts with the definition of a cell big enough to contain the entire simulation space initially empty. Particles are loaded into the tree one at a time and with each addition a local subdivision of the space may occur. If any two particles fall into the same cell, it is divided into four square or eight cubic subcells. This subdivision is applied recursively until each particle is put into a different cell. Consider the situation depicted in Figure 3.7 corresponding to two-dimensional simulation space. The first cell created, covering the whole space, becomes the root of the tree.

When we load the first particle into the tree, it is added to the root cell. Then, as we add the second particle, we find that the root cell is not empty. As a cell cannot contain more than one particle, we create four children for the root. It so happens that, in this case, when we load the old and the new particles to the newly created children, their positions allow them to fall in different cells; so this step is considered terminated and we can go on loading more particles into the tree. Proceeding similarly until all particles have been loaded, we obtain the tree outlined in Figure 3.7. Particles must always occupy leaf nodes. Note that in the final structure, pointers to empty leaf cells are not maintained: we only need to keep the non-empty cells.

After the tree is constructed, each node must be tagged with a multipole value and a center of mass spatial coordinate. The same BH algorithm can be used with a monopole or higher order multipole approximation. Starting at the leaves, we make each node's mass equal to the mass of the particle therein, and the center of mass equal to the position of the particle. Then we propagate these values one level up the tree: for each node we evaluate a truncated multipole series considering the ensemble formed by its children and compute its center of mass. This procedure is repeated for each level in the tree until the root node is reached.

For simulations with moving particles, the trees must be updated to reflect the new arrangement of particles in space *each time the positions are updated.* Usually, at the beginning of each iteration, the tree is constructed from scratch and at the end it is destroyed once interactions have been computed.

An important fact to point out about this tree construction process is that the resulting form of the data structure is dependent on the particle distribution, as demonstrated by Aluru [1]. Consider the situation in which we have two particles very close to each other. Applying the cell subdivision algorithm recursively, until each particle falls in a different

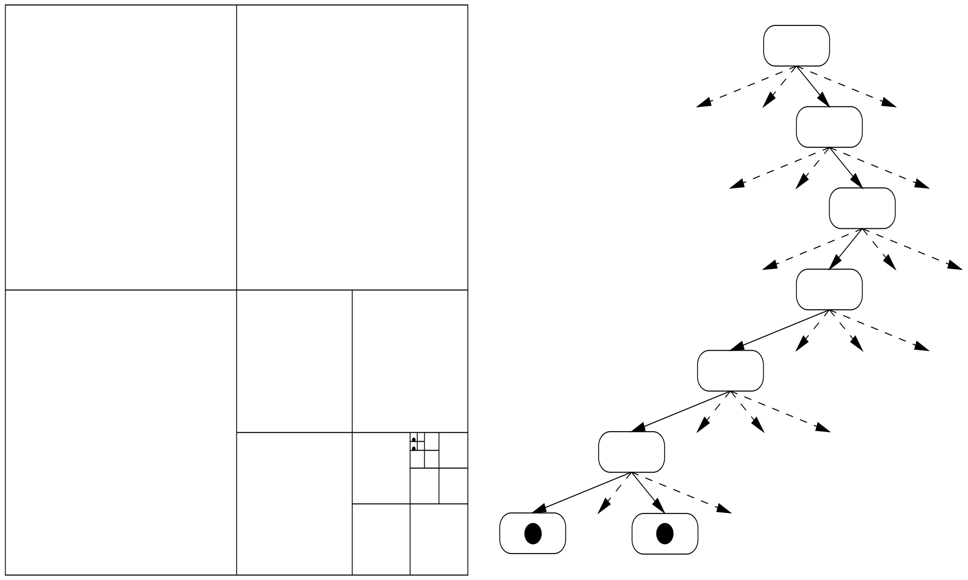


Figure 3.8: The BH tree can be sensitive to particle distribution

cell, may cause the resulting tree to be very tall, as shown in Figure 3.8. In many cases, including simulations in astrophysics, this is the cause of serious performance degradation, since bodies can come infinitely close to each other. The degradation comes not only from the tree construction process, but also from the added time taken in traversing the data structure during the computation of interactions.

Once the tree has been constructed, the computation of total interaction over a given particle p using the Barnes–Hut method follows the algorithm in Figure 3.9. The method determines whether it is necessary to compute individual interactions between a particle and an ensemble of particles using a *multipole acceptance criterion* (MAC) such as those defined in [47]. The original Barnes–Hut MAC is based on the ratio of the size l of the cell currently being inspected to the distance D between the cell's center of mass and p, what is called the *opening angle*. When l/D is less than an arbitrary accuracy parameter θ, the interaction can be approximated by the interaction of p with an imaginary particle of mass *ensemble.mass* at position *ensemble.centroid*. In this case the ensemble is far enough from p to be considered as a single particle with small approximation error. If on the other hand $l/D > \theta$, then the algorithm is applied recursively to compute the interactions between p and the subdivisions of the ensemble.

```
procedure BH_INTERACTION(p, ensemble)
begin
  if (singleton(ensemble))
    total_interaction = acceleration between the two particles
  else
    if  ((ensemble.diameter / distance(p, ensemble.centroid)) <  θ)
      total_interaction = acceleration between particle and ensemble
    else
      total_interaction =
        BH_Interaction(p, ensemble.child[0]) +
        BH_Interaction(p, ensemble.child[1]) +
        BH_Interaction(p, ensemble.child[2]) +
        BH_Interaction(p, ensemble.child[3])
  return total_interaction
end BH_INTERACTION
```

Figure 3.9: Barnes–Hut (BH) algorithm

We applied the BH algorithm with monopole approximations to the computation of interference noise in wireless simulations in a simple performance evaluation study. In our simulations, for each channel, we must compute noise in two different frequencies: the uplink and the downlink.

When we compute interference noise for uplinks, the particles in the algorithm are mobile transmitters. Since there can be more than one channel in the system, and we assume there is no interference between channels, we must have one BH tree for each distinct channel. Each tree will collect all the users transmitting on a channel. We are interested in computing the total noise incident on *base receivers* and thus we use the information in the BH trees to compute the total interaction over those points in space. Since in any one cell we cannot have more than one user transmitting on a given channel, we might have assumed that the users on one channel would be separated by a minimum distance large enough to prevent the degeneration of the BH tree. However, there exists the possibility that users in two neighboring cells are extremely close to the shared border and thus infinitely close to each other while using the same uplink frequency. In this case it would be advisable to employ a tree construction method like those proposed in [1].

For the downlink case, the situation is quite different. The bases, which are the particles in the downlink BH trees, are well separated; no two transmitters will be infinitely close to each other. So, in this case, the standard BH tree construction method can be used, allowing for a memory usage optimization. As bases are stationary objects, the trees can be constructed prior to the start of the simulation and would never change in structure. We can collapse the trees for all channels into a single tree for use with downlinks. the nodes then contain an array of elements, one for each channel the base can use. The elements in the array consist of mass and position of center of mass of the ensemble of transmitters using the corresponding channel.

3.3.2 Truncation of Interference Radius

A simpler alternative to brute-force computation and to N-body algorithms is to truncate the *sphere of influence* of each transmitter. For each cell in the simulated world, we can specify the signal's radius of coverage: outside this area no cochannel interference is observed.

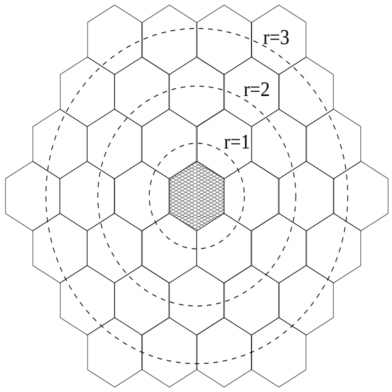


Figure 3.10: Macrorings in interference model

The rings of cells around a point are commonly referred to as *macrorings*. As shown in Figure 3.10, a ring of radius $r = 1$ includes all the closest neighbors of a cell. A ring of $r = 2$ is formed by the cells that surround a ring of radius $r = 1$; this same idea applies to define rings of larger radii. The number of cells in a ring of radius r is given by the expression $n(r) = 6 + 6(r - 1)$. We say that a *sphere* of radius r includes all cells in macrorings of radius i such that $1 \leq i \leq r$, a total of

$$s(r) \;=\; \sum_{i=1}^{i\leq r} 6 + 6(i-1) \;=\; 3i^2 + 3i.$$

To compute the total interference for each receiver in the system, one only needs to add up the contributions from transmitters that have the point of interest in their sphere of influence. Although the number of cells in a transmitter's coverage area increases quadratically with its radius, it's independent of the number of mobiles in the system. It's then easy to see that with this method, the asymptotic cost of interference computation drops down to $O(N)$.

The drawback of this method is that its side-effects are not all well-understood. Liljenstam and Ayani [31] have shown that simulation results can be severely affected when the interaction radius is too small, say $r = 1$. However, they have also observed that as the radius is increased, the errors diminish very rapidly: the accuracy obtained with $r = 3$ is not significantly better than that produced by $r = 2$. The good news is that small values of r produce substantial improvement in the simulation execution time.

What has not been totally explained is how different channel allocation and power control algorithms respond to this technique. The authors have experimented with two different scenarios and both of them were substantially different from the algorithms we employ in this work. As pointed out in their conclusions, *the actual error introduced in the simulation output depends largely on the resource allocation algorithm being used.*

Their time-stepped simulations were relatively short (300s), with a value of $\Delta = 2s$ that reflects a fairly relaxed power control. We are interested in attacking long simulations of systems with much stricter adaptive power control and thus with Δ smaller by one or two orders of magnitude. Putting together this observation and the authors' conjecture that the feedback between power control and channel allocation may increase errors in the simulation output, we conclude this is still an open problem. How LADCA and the truncated interference model interact is a point that bears further investigation and which, currently, lies outside the scope of this work.

Working on the reduction of time spent in interference computations addresses the problem of accelerating the simulation on one front, but perhaps not the most important one. The next point to investigate is how to reduce the number of state updates throughout the simulation.

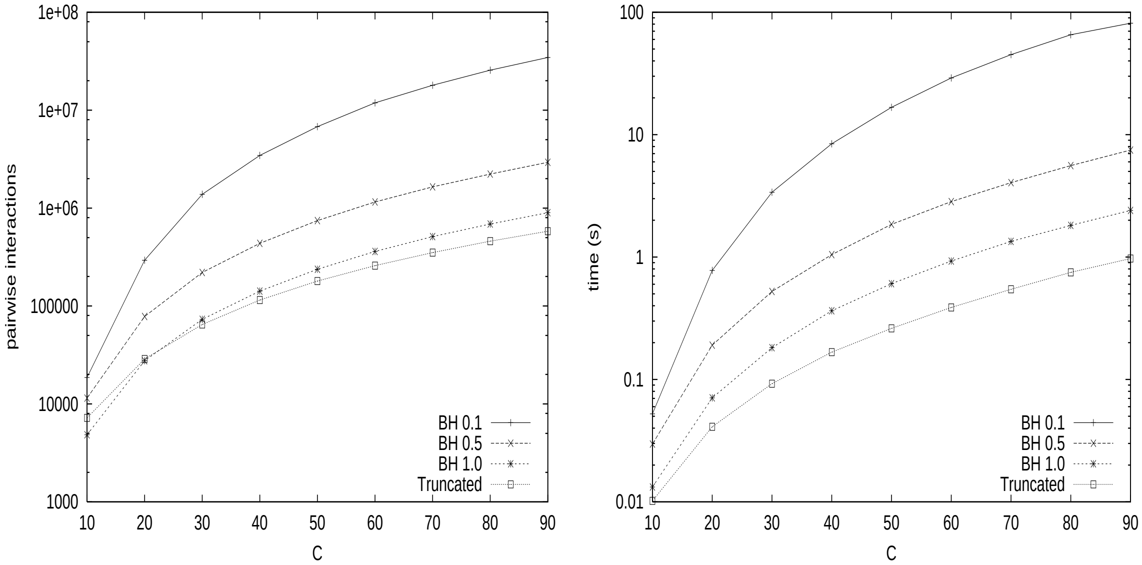
The algorithms we use require a value Δ with an order of magnitude of 10^{-2} seconds:

the weight this bears on the simulation is very heavy. Most of the simulator's execution time could be spent in interference and power computations. Next, we present a technique that allows the simulation to advance in irregular time increments adapted in accordance with the state of the system. As we show later, this substantially reduces execution time while *not affecting the accuracy of the simulation output.*

3.3.3 Comparing the Different Methods

We have experimented with three different methods to compute interference in a wireless cellular model: brute-force, BH (with monopole approximation) and truncated interference (with a sphere of radius $r = 3$) [41]. In this analysis, we limited our experiments to the case of a single channel model. The networks we use are composed of $C \times C$ hexagonal cells with side lengths equal to 1000m and the receivers are subject to internal noise $\nu = -120$dB. We assume each cell has an active call, each mobile is placed randomly within its cell, uniformly at random on one of the three straight lines that bisect opposing sides of the hexagonal cell. This provides a "worst-case" scenario for BH, insofar as its computational efficiencies are achieved when the mobiles are sparsely separated. If significant performance gains are observed in this case, we will see better gains on networks with the same number of mobiles, distributed more sparsely over larger grids.

Varying the size of the network for $C \in \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$ and, thus, also the number of mobiles in the system, we evaluated the total number of pairwise interactions (uplink and downlink) computed in each of the three methods. Figure 3.11(a) shows the results obtained (using a logarithmic scale on the vertical axis) for three different values of opening angle θ in the BH algorithm. It is clear that the use of the BH algorithm gives the

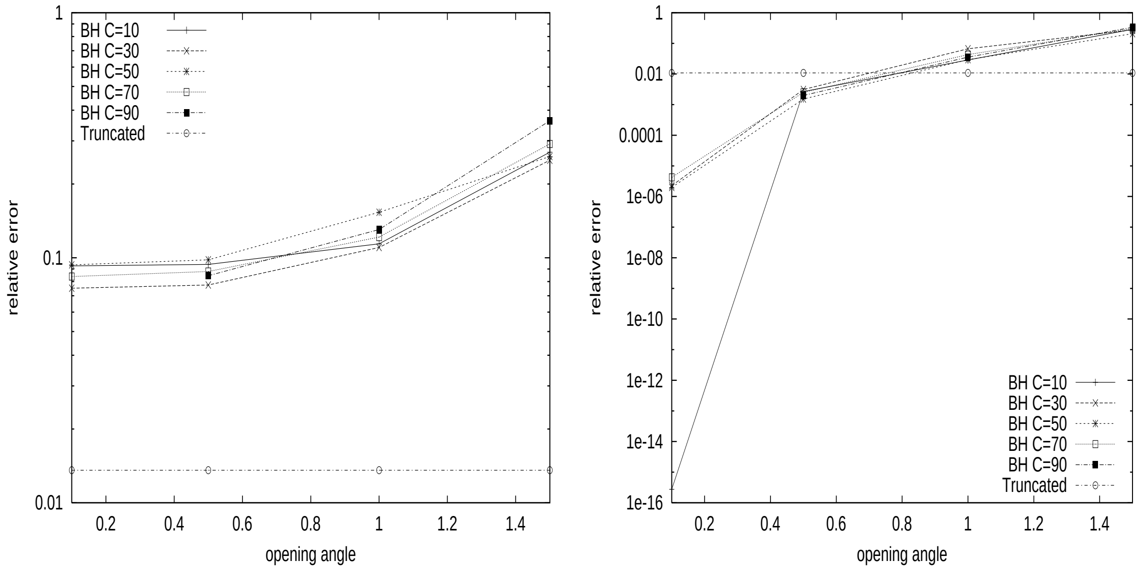


(a) Interaction count (b) Run-time

Figure 3.11: Performance comparison of BH, brute-force and limited interference under heavy load

modeler the power to choose different levels of accuracy according to how much runtime can be afforded. Smaller values of θ bring the number of pairwise interactions computed up toward the same as in brute-force. In fact, for a sufficiently small θ, the BH algorithm will compute exactly the same values obtained with brute-force. It is for larger values of θ that BH becomes an attractive option.

For the same scenario, we have measured the execution time of a complete round of interference computations in the model on an SGI Origin 2000 with 180MHz clock, shown in Figure 3.11(b). This data helps quantify the cost of interference calculations. We see that for large mobile counts the execution improvement (about two orders of magnitude) corresponds with the reduction in numbers of interactions (about two orders of magnitude). This shows that indeed the added cost of building the quad-tree is amortized away on large models.

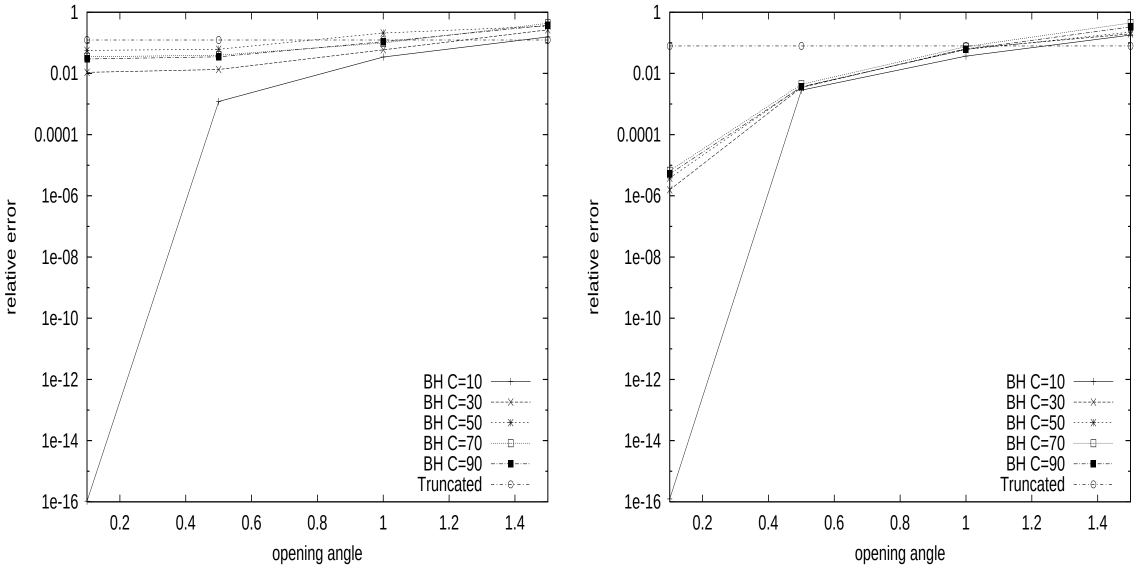


(a) Uplinks (b) Downlinks

Figure 3.12: Mean relative error in interference computation under heavy load

At $\theta = 1$, an entire cluster is treated as a single particle if the point where the interactions are measured is at least as far away from its center of mass as the BH cell containing it is large. The economy in the computation is so significant that interference can then be computed with a number of interactions in the same order of magnitude as with the limited interference method. The associated errors, however, are not negligible for a system under such heavy load. Figure 3.12 shows that, when $\theta = 1$, the mean relative error for uplinks and downlinks is one order of magnitude higher than that of the truncated method, which does not consider the contribution of transmitters farther away than three times a cell-width.

The case of a system under lighter load (blocking probability of at most 10%) is quite different. For both uplinks and downlinks, as θ increases, the mean relative error approaches that of the truncated method from below. At $\theta = 1$, for varying system sizes, the magnitude of the error is comparable to that of the truncated method (see Figure 3.13): this fact is



(a) Uplinks (b) Downlinks

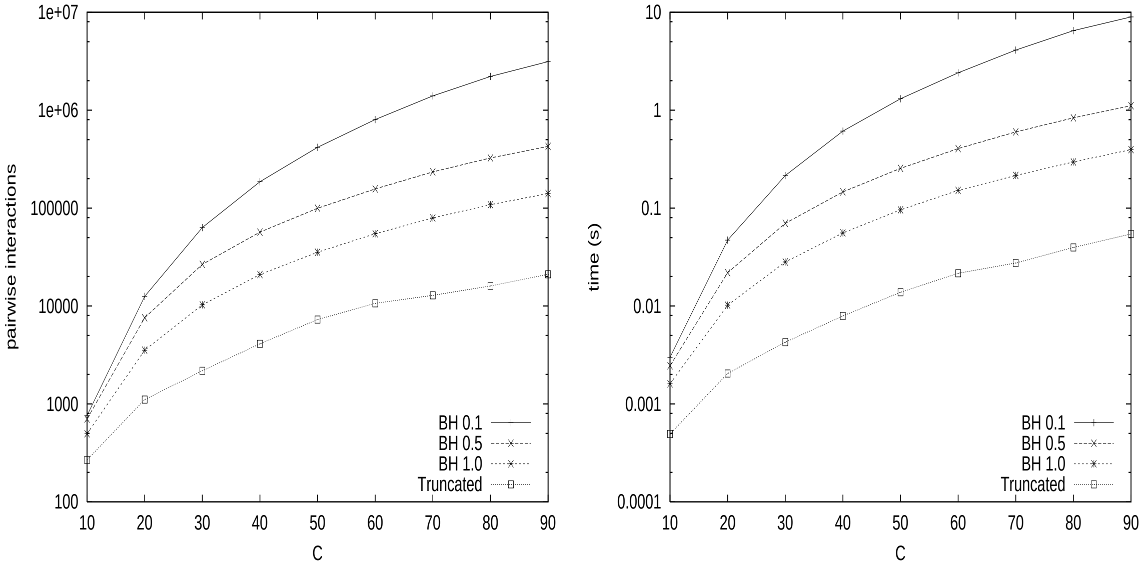
Figure 3.13: Mean relative error in interference computation under light load

---

encouraging in that it indicates that BH is a good alternative as far as accuracy is concerned.

On the other hand, comparing run-times for this same situation, Figure 3.14 shows that BH is considerably slower (close to one order of magnitude). We see that, as the system scales up, the run-times observed reflect the number of pairwise interactions. Again, the graphs indicate that our implementation performs just as theory predicts and, therefore the longer run-times observed with BH are due to factors inherent to the algorithm. A lightly loaded system offer less possibility of clustering and, thus, the strong point of the BH algorithm is not exercised.

An important observation to make at this point is that FDMA models will tend to “weed out” situations of high clustering, since, for these systems, channel reuse within a small radius is not favored. This will tend to drive the system to mobile arrangements in space similar to that of the lightly load case we observed. Although the use of the BH



(a) Interaction count (b) Run-time

Figure 3.14: Performance comparison of BH, brute-force and limited interference under lighter load

---

algorithm for the simulation of FDMA models will yield good accuracy, it will exhibit longer run-time than the truncated method.

CDMA models allow, comparatively, many more opportunities for clustering with the BH algorithm. In this case, there is no restriction on channel reuse: system allows the same frequency to be simultaneously used by several cells within the same neighborhood. In a case like this, the power control and channel allocation let configurations with more dense arrangement of mobiles persist and then BH *may* be a good alternative for interference computation. The run-time of BH in this case can come close to that of the truncated method, however, its accuracy will leave much to be desired. When low relative errors are imperative, one should consider using higher order multipoles in the interference calculations with BH. The implication this choice may have on run-time, however, is not known at this time and its investigation lies outside the scope of this work.

We conclude this comparison of methods for interference computation by stating the fact that, with all factors considered, the truncated approach is superior to the BH algorithm. We have shown that for sphere of influence of radius $r = 3$, truncation led to better run-time and accuracy than our implementation of BH.

A final message to keep in mind, however, is that, in spite of our results, N-body methods may still be interesting for interference computation. While truncating sphere of influences provides only one handle on accuracy (the radius of the sphere), an algorithm such as BH can provide at least two: the opening angle θ and the number of multipoles used in the approximation. Our study indicates the *possibility* that in special circumstances and with further enhancements, BH could be a good choice. How to make a winner out of this method is still an open problem.

3.4 Avoiding Time-Stepping through *Interval Jumping*

Observing the numerical algorithm embodied in Equation 3.4, we see that at its core, it is equivalent to a modest integration by a trapezoidal rule with equally spaced abscissas. Numerical integration has evolved much past that point and we can accelerate this simulation by taking advantage of this fact.

3.4.1 Foundations

The basic idea is to observe the evolution of states in our model and identify periods of simulation time that can be "fast-forwarded" without affecting the correctness of the results obtained. To this end, we have developed *interval jumping*, a technique named after the effect of advancing time in leaps and bounds rather than in steps. While time-stepping progresses at the cost of n state updates when traversing an interval of length $n\Delta$, interval jumping reaches the same final state with one single state update.

In order to be able to apply this idea, the model must conform to a couple of underlying assumptions that define conditions for the mechanism to work:

> **Assumption 1:** *A power control law is a fixed-point computation which converges to a power assignment that satisfies a well-defined goal, independently of the initial state of the system.*

Power control laws are inherently *fixed-point computations*: starting from some initial power allocation, this iterative process is repeated over and over until convergence is achieved. Among the innumerable examples that can be found in the literature, many of these algorithms reach the same solution independently of initial state, therefore this assumption is quite reasonable.

> **Assumption 2:** *A power control law strives to maintain $\rho_i(k)$, the instantaneous SIR for a receiver i, close to a pre-defined target ρ. To do so, it computes the power levels at time k based on the state of the system (values of interference and attenuation coefficients) at time $k-1$.*

For traditional simulation of power control, the notion of evolution of time gets naturally intertwined with the notion of successive iterations of the power control algorithm. Every Δ seconds (or one unit of discrete time), transmitter power allocation is updated. If the number of transmitters in the system is T, we can describe the power allocation at the k^{th}

time-step as P_k, a vector whose j^{th} component gives the power of transmitter j. Let A_k be a $T \times T$ matrix that collects $a_{ij}(k)$, the attenuation coefficient on the propagation path between each transmitter i and receiver j at time k. If we define the power law as a function $F_A : \mathbb{R}^T \to \mathbb{R}^T$, such that $P_{k+1} = F_A(P_k)$, we can express the time-stepped simulation by:

$$P_1 = F_{A_0}(P_0),\ P_2 = F_{A_1}(P_1),\ \ldots,\ P_{k+1} = F_{A_k}(P_k),\ \ldots$$

At each successive time advance, a new iteration of the algorithm brings the power allocation closer to the optimal solution. The parameter k in the equations not only represents a discrete instant of time, but also identifies one single iteration of power control. After a finite number of iterations, the power allocation converges to a fixed-point, which may be the optimal solution vector. For models with user mobility or changing propagation conditions, the same notion applies, but it must be understood that the optimal solution may not be achieved. Although several power control algorithms converge very fast, one iteration is hardly enough and thus the optimal solution may be "tracked" rather than reached.

To clarify why interval jumping is viable, we must invite the reader to dissociate discrete time from iterations of the power control algorithm. Imagine that at every instant k, time could be frozen and the power control law iterated to convergence. Note that this is quite a stretch, since, as we have mentioned, the variables change faster than an optimal solution can be found. In the simulation, however, time *can* be frozen and we use this fact to break the causal dependency between successive model states.

If a jump takes the simulation from time 0 to some instant k, we simply update mobiles' positions, compute the attenuation matrix at time $k-1$ and iterate the power control law to convergence. As long as Assumption 2 holds, the simulation progresses correctly from that point. Only the state at k is relevant to the rest of the simulation; all those before the jump do not matter. Formally, we describe interval jump by:

$$P_1' = F_{A_{k-1}}(P_0),\ P_2' = F_{A_{k-1}}(P_1'),\ \ldots,\ P_{k+1}' = F_{A_{k-1}}(P_k'),\ \ldots\ \text{until } P_{k+1}' \approxeq P_k'$$

Assuming that both methods of simulating the same interval produce nearly the same results, the latter one will only be more efficient if it requires fewer than k iterations to achieve convergence. Since, in practice, convergence is achieved within a few iterations (2 to 10 in the case of LADCA), interval jump should produce a significant performance improvement.

3.4.2 To Jump or Not to Jump

Even when Assumptions 1 and 2 hold, *interval jumping* cannot be indiscriminately applied to just any interval. Consider a jump over an arbitrary time interval: if any event due to maximum power usage should occur between the endpoints, the simulation would not be able to register it at the correct time. Therefore, to preserve the validity of the simulation results, before any jump is attempted we must make sure that, during that period, nothing "interesting" happens.

The gist of the idea is to determine intervals where two types of events do not occur: those that could produce a change in channel allocation that would increase power levels and those that would be triggered by an increase of power levels as the interval is traversed. We henceforth refer to all these events as *critical events* (CEs) and define a "jumpable" interval as one that does not contain any of these.

We start to construct an interval by choosing an arbitrary instant a where no event happens, then we inspect the random streams that determine *call arrivals* and *hand-offs* to compute the instant b when the earliest following event occurs. Knowing the current position of mobiles and their projected trajectories, we can figure out when cell boundaries will be crossed and compute the time when hand-offs happen.

The importance these two types of events have in determining the upper edge of an interval stems from the fact that, directly and indirectly, they lead to an increase in the overall transmitted power in a channel. When a new transmitter starts to operate, it adds to the interference observed by ongoing connections in the same frequency. As a consequence, power levels may rise to keep the SIR on target, possibly leading to the interruption of one or more connections.

Next, we look at the likelihood that events such as *channel switches*, *call drops* and *call blocks* will be seen in $[a, b]$. These events are scheduled when power levels reach maximum value and an attempt to predict their ocurrence involves an assessment of the rate of power increase. We cannot perform this analysis quickly and precisely for all transmitters because it would involve computing the power allocation at every instant of discrete time, exactly what we are trying to avoid.

We can, however, observe the evolution of attenuation coefficients in the interval, since they depend only on user mobility and on fading effects. Inspecting trajectories and pre-sampling the random stream that defines random variables used in the fading models, we can construct bounds for these coefficients and use them to construct a worst case scenario. Together, the bounds for coefficients and positions of transmitters and receivers describe a hypothetical situation that may never be observed in the course of the simulation. These fictitious conditions, however, are of great importance to interval jumping. If we compute the power allocation for this configuration and find that no transmitter reaches maximum power, then certainly no transmitter reaches maximum power in the real system. We elaborate on this point in the next section.

3.4.3 Formal Description of *Interval Jumping*

Having presented this procedure intuitively, we now proceed to give it a more precise treatment. Let $a_{ij}(k)$ be the attenuation coefficient between the transmitter in pair j and the receiver in pair i at time k as:

$$a_{ij}(k) = \frac{1}{d_{ij}^{\alpha}(k)},$$

where $d_{ij}(k)$ is the distance from transmitter j to receiver i. Also, if $c^{'}$ and $c^{''}$ are the two channels adjacent to channel c where a receiver i operates, we model the adjacent channel interference over i at time k as:

$$x_i^c(k) \;=\; \sum_{j\neq i} f\, p_j^{c'}(k)\, a_{ij}(k) \;+\; \sum_{j\neq i} f\, p_j^{c''}(k)\, a_{ij}(k),$$

where $0 < f < 1$ represents the fraction of transmitted power which extends into the portion of spectrum used by an adjacent channel.

We can now write a generic power law (in discrete time) for channel c:

$$\rho_i^c(k) \;=\; \frac{p_i^c(k-1)\, a_{ii}(k-1)}{v_i^c(k-1)}, \tag{3.5}$$

$$\text{where} \quad v_i^c(k) = \nu + x_i^c(k) + \sum_{j\neq i} p_j(k)\, a_{ij}(k)$$

and ν is the receiver's thermal noise.

Equation 3.5 can be rewritten as:

$$p_i^c(k+1) \;=\; d_{ii}^{\alpha}(k)\,\rho_i(k)\,v_i^c(k) \tag{3.6}$$

Next, consider an artificially constructed system S^*, such that the attenuation coefficients in the computation of power for every transmitter is the worst case observable in interval $[a,b]$. This is equivalent to saying that in S^*, $d_{ii}(k)\mid_a^b = \overline{d_{ii}}$, the largest distance between a transmitter-receiver pair i, and $d_{ij}(k)\mid_a^b = \underline{d_{ij}}$, the shortest distance between transmitter j and receiver i during interval $[a,b]$.

We can then state the theorem that lies at the crux of our adaptive time-step technique, which is valid for *any* power control algorithm that satisfies Assumptions 1 and 2.

Theorem 1: *If the power level of some transmitter rises to the maximum in the full time-stepped simulation of interval $[a,b]$, then the same will happen when we compute power levels for S^* constructed as above.*

Proof of Theorem 1: (by contradiction) Suppose there is a time $s \in [a,b]$ in the time-stepped simulation when the power level of a base transmitter i is maximal and this is not so in S^*. The positioning of mobiles at time s is such that all mobiles are at least as far away from their base stations as they are in S^*; we observe that $v_i^c(k)$ is a *decreasing* function of each $d_{ij}(k)$, and an *increasing* function of each $x_i^c(k)$. S^* is constructed to minimize each $d_{ij}(\cdot)$ and to maximize each $d_{ii}(\cdot)$, so the noise value computed in S^* is larger than in the real system. Since the noise value in S^* is larger, the power to which the fixed point algorithm converges must be larger than that in the real system, establishing

```
algorithm IntervalJumpingSimulation
begin
  while (! simulation terminated) do
    compute interval [a, b]
    if (b − a > mΔ and probe(a, b))
      advance time to b
      update mobiles positions to time b
      while (! power levels converged) do
        compute noise
        update power levels
      end while
      apply LADCA algorithm
      run time-stepped simulation for Γ steps
    else
      run time-stepped simulation to b
  end while
end IntervalJumpingSimulation
```

Figure 3.15: Algorithm for *Interval Jumping* simulation

the contradiction. (Note we present here only the downlink case understanding that, by symmetry, the same reasoning can be applied to uplinks.) □

From Theorem 1, we define predicate **probe**(a, b) which evaluates to TRUE if no transmitter reaches maximum power in $[a, b]$. If some transmitter violates this condition in S^*, the predicate evaluates to FALSE, indicating that in the real system the same *may* happen.

In the implementation of probe, we construct S^* and compute the fixed-point for this configuration, taking the power allocation at a an initial state. If at any time some power level rises to the limit, we say the probe *fails* and the iterations are stopped immediately. Otherwise, if convergence is achieved with all transmitters at lower power values, we say the probe *succeeds*. Using this predicate, we define *interval jumping* as shown in Figure 3.15—assume the **if** clause in the algorithm is evaluated in short-circuit mode.

The cost of applying **probe** on an interval is not negligible. Experience has shown that it lies between the cost of one and three time-steps, depending on the system load. To evaluate the predicate, many bounds have to be estimated and the power assignment for S^* has to be computed. Moreover, following the jump, there exists the added computational cost of iterating the power control law to convergence, which should be proportional to the length of the jump. Consequently, only intervals with a minimum length should be considered for jumping. One must set m appropriately to make sure all costs are being offset.

Assumption 1 gives us important leverage: all we need in order to compute the power assignment at b is the spatial arrangement of mobiles. However, to ensure the simulation resumes from the correct state after a jump, we iterate the power control law until a fixed point is reached. This produces no advance in simulation time; it only allows that a valid state for this instant of time is created. After the new power assignment is computed, we must apply the actions of the LADCA algorithm; that is, we must check whether any call blocks or drops need to be scheduled. Also, we need to update the algorithm counters `c.num_good`, which reflect for how many Δ the ongoing connections have experienced a satisfactory SIR.

Following the jump, it is important to time-step forward by a minimum of $\Gamma\Delta$ after the event at time b has been processed. The recent event may cause an increase in power levels and, as a consequence, trigger a CE. Since the time taken for power levels to adjust to the new conditions is related to the speed of convergence of the power control law, Γ should be set to the expected number of iterations to convergence or higher. At the end of this time-stepping period, another jump can be attempted.

To assess the potential of this technique, we conducted short simulation runs of 4,000s and recorded the size of jumpable intervals in terms of the number of time-steps contained in each. The model parameters used for this test were: 20 users with constant speed of 10m/s, per user call inter-arrivals exponentially distributed at a rate of 3.0calls/h, call lengths exponentially distributed with mean 360s, a network of 10 by 10 hexagonal cells with 1000m sides and time-step size of 0.01s. These conditions allow us to observe the system under a load compatible with the most common simulation scenarios, placing blocking probability in the range of 1% to 10%. Under very high loads, it is reasonable to expect that interval jumping should not perform well since jumpable intervals would tend to be short and scarce.

In this scenario, the aggregate distribution of call interarrival times is exponential. When this fact is put together with the procedure for interval construction explained in the previous section, one sees that the distribution of jumpable interval lengths is also exponential. Since interval length is, essentially, a minimum of exponentially distributed random variates (new call arrivals and inter-cell hand-off traffic), its distribution is also exponential. We confirmed this fact through exploratory data analysis of the results of our preliminary run and observed a mean interval length of 75Δ. This number indicates a considerable potential for accelerating simulations of this model both sequentially and in parallel.

3.5 The First Experiments

In this section, we present our first experiments with a sequential simulator built to provide insight into interval jumping's performance and, also, to serve as a basis for speedup com-

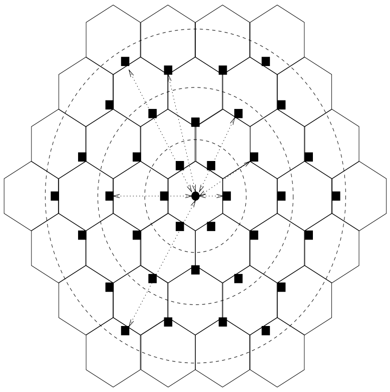


Figure 3.16: Computation of *"worst case"* bounds

putation. Before going into experimental data, we discuss a few implementation details, which are also relevant for the parallel simulator.

3.5.1 Computation of Bounds for probe

We compute noise over one particular receiver summing only interactions produced in its *sphere of influence*. The radius of this region is determined *a priori*, based on the worst possible arrangement of transmitters in space and an arbitrary, maximum acceptable error for interference computation.

Here we followed a simplistic approach, which has very low computational costs. The values of interest when we evaluate the **probe** predicate over an interval of time are $\underline{d_{ii}}$ and $\overline{d_{ij}}$. Let us now consider only the case of uplinks, understanding that an analogous situation applies to downlinks.

The questions to ask are: *"What is the greatest possible distance between a mobile and its base in this interval of time?"* and *"How close to the mobile's base station do other interfering mobiles get in this interval of time?"* The answer to the first is quite straightforward. In our model, base stations are chosen based on geographical proximity to the mobiles and, furthermore, handoffs occur when cell boundaries are crossed. The upper bound for $d_{ii}(t)$ in any circumstance, will then be given by the maximum possible distance between base and mobile in a cell—the side of the hexagon, in other words.

As for the second question, for every cell in the vicinity of a base, we can compute the closest that an interfering mobile in another cell can come to its location. As illustrated in Figure 3.16, once the interference radius has been defined, we can easily compute the worst case $\underline{d_{ij}}$ for mobiles in each particular cell in the sphere of influence of a base station.

Consider a receiver i positioned inside an arbitrary cell $\mathcal{C}$: the number of transmitters j that can create interference on this receiver is equal to the number of cells in $\mathcal{C}$'s sphere of influence. Note that the distance between each transmitter j outside $\mathcal{C}$ and receiver i inside $\mathcal{C}$ is bounded from below by cell geometries. By the same token, the maximum distance between the transmitter/receiver pair in the cell is bounded from above.

We explore these facts to pre-compute all values $\overline{a_{ii}}$ and $\underline{a_{ij}}$ for one generic cluster of cells, which can be remapped to the real space when we compute bounds for a specific receiver in **probe**.

The quality of bounds used to implement **probe** has an impact on the performance of interval jumping. The tighter the bounds, the more accurately the predicate will be able to indicate the threat of a discrete-event. If worst-case bounds are too loose, **probe** may indicate that the interval should not be jumped when it should be safe to do so. Note that an efficient method to compute bounds on the fly for each particular interval makes possible a refinement of our technique. If an entire interval cannot be jumped at once, the simulator might be able to identify safe subintervals to jump over.

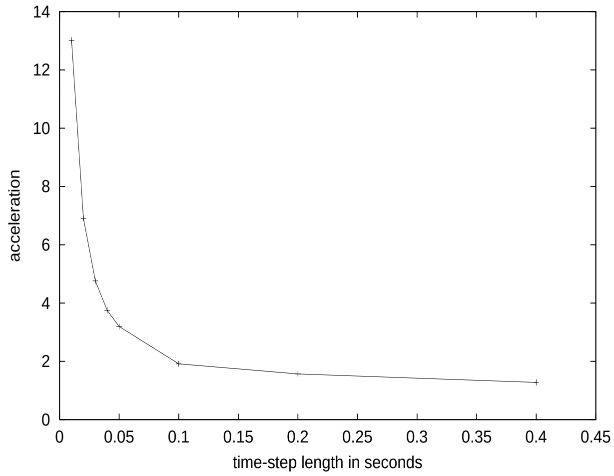


Figure 3.17: Acceleration vs. step size using interval jumping

3.5.2 Experimental Results with Sequential Simulation

The experiments we present next are simulations in which an interval is either entirely jumped over or fully time-stepped, a consequence of using worst-case bounds in **probe**. In this case, what determines whether the predicate returns true or false is solely the concentration of callers, in the same channel, around each cell.

Figures 3.17 and 3.18 illustrate the performance of the jumping technique as a function of step size. As could be expected, we see a fast degradation with increasing step sizes: the coarser resolution reduces the number of time-steps to the next discrete event.

In these experiments, we set $m = 2$ in the interval jumping algorithm, a value that keeps the cost of interval jumping balanced against time-stepping. As figure 3.17 shows, interval jumping's performance does not become worse than time-stepping, even for larger Δ.

Since we estimated that the expected interval length observable with these model parameters is of 75Δ, the best-case expected acceleration one could attain with interval jumping would be of 75. Our experiments, however, have shown an average acceleration of only 13.7.

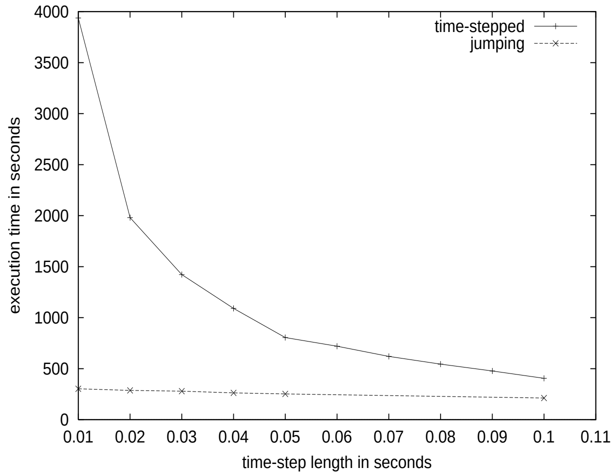


Figure 3.18: Execution time vs. step-size for time-stepped and jumping simulations

At this stage, an interval was only jumped over if probe was successful at its endpoint b. This "all or nothing" method cannot exploit the full potential of interval jumping. Intervals that don't pass the probe are time-stepped through, and so the number of actual jumps in the simulation turns out to be much smaller than the number of candidate intervals.

At this stage we also hypothesized that hidden costs in our implementation (due to memory utilization and threading mechanisms) contributed to slowing down the execution of jumping simulations. Later, we chose to retire this first implementation of the simulator (written in C++ and CSIM [44]), and re-start from square one. The project continued as a fully "home-grown" simulator, now exclusively written in C++ and with a splay tree [45] at the heart of its engine. As we show later, these improvements have made our simulations more reflective of the potential of interval jumping.

3.5.3 Parallel Discrete-Event Simulation

There are two principal classes of algorithms for parallel discrete-event simulation (PDES), *conservative* and *optimistic*. A complete exposition of the characteristics of these two types of PDES algorithms is beyond the scope of this work and we refer the reader to the excellent tutorials that can be found in the literature [19, 20, 21, 35]. We focus our work on very basic conservative algorithms with the intention of merely illustrating the issues related to further accelerating interval jumping through execution on parallel computing platforms.

Simulation models, both for serial and parallel execution, are described in terms of cooperating entities called *processes*. These entities transfer control, or data, from one to another by exchanging *messages*. Conservative methods rely on the definition of a static network of unidirectional, order-preserving *channels*. The transmission and reception of messages correspond to events in the simulation; events map to points in time when processes interact, potentially effecting changes on the state of one another.

Each channel is associated with two values: a channel *delay* and a channel *clock*. The channel delay models the time a message takes to traverse the channel from one endpoint to the other. The channel clock, which is monotonically increasing, reflects the time of the last message deposited into the channel. A channel connects two processes, acting as an *output channel* for the process that deposits messages in it and an *input channel* for the process that receives messages from it.

Processes are constructed with the knowledge of their input and output channels, and causality is enforced locally by inspecting input channels before processing each event. If all input channels contain messages, the process can advance to the time of the earliest one and consume it, perhaps sending other messages through its output channels as a result. If any one of the process' input channels does not contain a message, the process must block until this condition is reversed.

The key notion that allows conservative algorithms to achieve good performance even in the face of potentially low concurrency due to blocking is that of *lookahead* [18]. If a process knows the delays and clock values of its input channels, it can figure out the latest time it can advance to without risking a causality error. The better a conservative simulation can exploit lookahead, the higher is the performance one can obtain from it.

The main difficulty in parallelizing the simulation of wireless cellular systems lies in the fact that the nature of the problem does not offer much lookahead to exploit. This fact is explained in more detail further ahead, in 3.5.5, where we describe our experiments with two very simple conservative algorithms. Next, we present a brief review of the literature in PDES of wireless systems.

3.5.4 Simulation of Wireless Systems in Parallel

As wireless cellular systems become increasingly popular, more research effort is focused on their design and the need for ever faster and more accurate simulators becomes stringent. In the last 10 years, the simulation community has witnessed a trend toward the development of larger and more comprehensive models for wireless networks, which has inevitably led to research in PDES.

Several research groups throughout the world have been hard at work developing simulators that focus not only on one or another component of a wireless network, but on the problem as a whole. The WINLAB group at Rutgers University [27, 39, 37] has developed

WiPPET, a general parallel simulation testbed that allows users to easily put together complex models using building blocks designed by experts in each different sub-area. Drawing from a database of components, the simulationist can then concentrate on high-level model design rather than dealing with the intricacies of programming the simulator's core. Another group at UCLA has been involved in a project of similar scope called GloMoSim [3, 49]. While WiPPET relies on Georgia Tech Time Warp (GTW) [13], an optimistic synchronization kernel for parallel simulation, GloMoSim is situated at the other end of the spectrum and employs conservative algorithms.

The significance of these works has been paramount to making the construction of reliable simulation models of wireless systems more accessible. However, there are still several performance issues at the simulator core level that bear the need for deep investigation. Until these problems have been properly addressed and solved, the simulation of wireless models of realistic scale will remain a computational burden of almost intractable proportions. Our research with interval jumping has begun to blaze a trail in this direction.

Also on this same vein, the contributions by the group at the Royal Institute of Technology, in Sweden, have been key. First, they have shown the efficiency of partitioning wireless models by channel rather than by geography for parallel execution [29, 30]. Then, later on, they launched an investigation on interference computation, one of the main bottlenecks of a comprehensive wireless simulation. In their paper, Liljenstam and Ayani [31], working with a model that includes adaptive power control, simplify interference computations by deliberately truncating of interaction radius. The idea stems from the fact that radio signals decay exponentially fast with increasing distance between transmitters and receivers. They have shown that the careful application of this concept can produce small errors, while dramatically reducing the asymptotic complexity of computations. They also report that analysis of the errors introduced by truncation must be done on a case-by-case basis.

3.5.5 Experimental Results with Parallel Simulation

Using interval jumping, we have developed a sequential simulator that has served as the testbed for our ideas. Although the experiments we have described, so far, deal exclusively with the simulation of single channel systems, our simulator has been designed to handle models with multiple channels.

A simulation model for a wireless system with multiple channels is little more than a number of instances of single channel sub-models. Every time a call is switched from one channel to another, the models communicate. This partitioning strategy, first identified by Liljenstam and Ayani [29], naturally leads to some type of concurrent simulation (multithreaded, distributed or parallel).

When different sub-models are assigned to different threads of execution, each with its own simulated clock, care must be taken so that event causality is preserved. Assuming that clocks are loosely synchronized, when a call is migrated from one channel to another, the clock value of the process simulating the receiving channel must match that of the process simulating the sending channel. A factor that makes the parallel simulation of wireless systems with channel partitioning a harder problem is this absence of lookahead. Channel switches happen unpredictably, and when they do happen, the time difference between sending and receiving the event is either zero or virtually negligible.

Another model characteristic that can substantially increase the difficulty of developing a parallel wireless simulator is the possibility of adjacent channel interference. The existence or absence of this feature has serious implications on how time can be advanced.

In the serial implementation of interval-jumping, the interval endpoint b is the earliest time *in the system* when an event, such as a call arrival or hand-off, happens. When a new call arrives, the system first determines which channel takes it; to even out the load, this channel assignment is done uniformly, at random. For the duration of each call, the system keeps track of all the channels that have been visited—when a hand-off occurs, the channel chosen in the new cell is one that has not been used so far. Suppose now, that we associate channel c to this new event.

Knowing that an event happens at time b, we start the attempt to advance the time of channel c by probing at the time of the event. If this test passes for all channels in the system, they are all simultaneously brought up to time b and interval jumping proceeds exactly as in the sequential simulation. The question now is whether, in a parallel simulation, these probes and updates can be conducted asynchronously.

Suppose that channel c can induce interference in adjacent channels c' and c''. If the probe applied at time b in channel c passes, then the aggregate power used in channel c rises, thereby raising the noise level in c' and c''. This means the operating state in those channels may change—power levels may have to increase to meet the target SIR and it is possible that the increased noise can be enough to cause this to be impossible, triggering a channel switch, call block or call drop. Even if this does not occur, mathematically the power levels in c' and c'' will rise, thereby inducing greater noise in *other* channels adjacent to them, where the same logic applies.

Extending this line of reasoning out transitively, all channels must have their power levels recomputed at time b. If we are to synchronize conservatively in a parallel context,

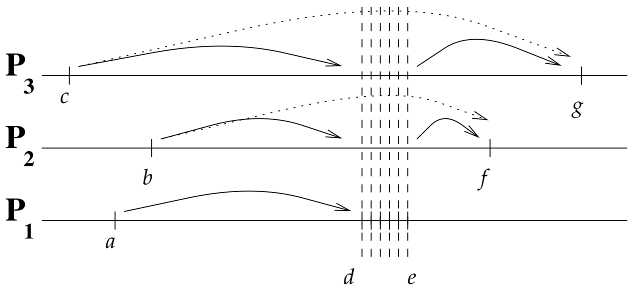


Figure 3.19: Synchronizing parallel jumps

we must either use more sophisticated bounds to avoid all these channel updates, use some other trick, or simply do all the updates. Given that for these experiments we compute bounds on attenuation coefficients based solely on *numbers* of users in cells and worst-case assumptions about their spatial coordinates, we are forced to keep all processors in tight synchrony, if we are to synchronize conservatively. Our parallel version does what the serial version does, with the concurrency coming solely from the obvious parallelization of the power-vector update step. As illustrated by Figure 3.19, the processors cooperate to compute the time of least next arrival ($\min\{g, f, d\}$) where the probe is applied. If the probe is successful, all processors engage in the power vector update at that time (time-steps from d to e). Otherwise, all processes time-step synchronously to the end of the interval.

There is not a great deal of lookahead to exploit in this model: we cannot predict the evolution of power levels, which can trigger changes in channel allocation. With the knowledge of model parameters and the LADCA algorithm, we can extract *some* lookahead out of the model:

- A call in probing phase is switched to a different channel as soon as power levels reach maximum. We cannot determine this moment, but knowing the order in which channels will be visited, we can predict when a call would migrate to each different processor, if it at all.

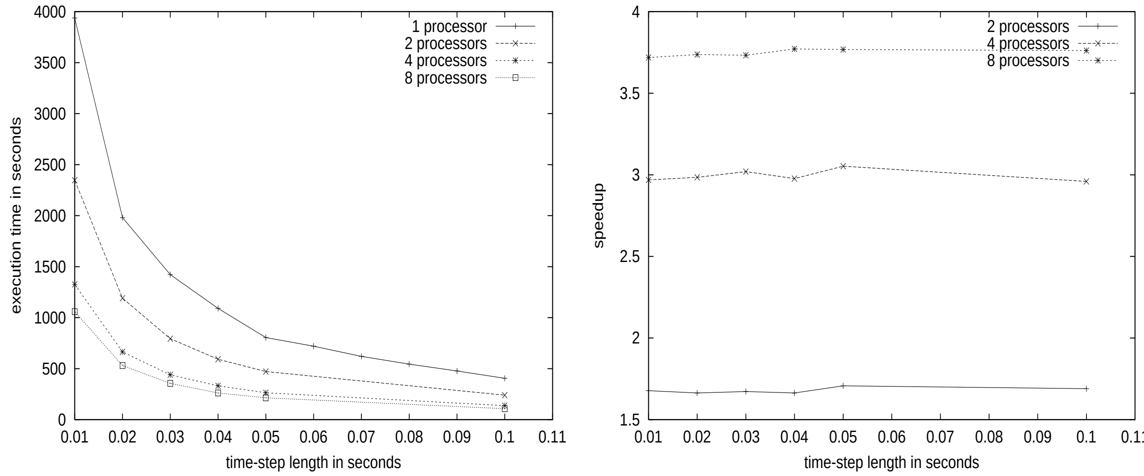
- A call in service phase is switched to another channel after at least $\Phi\Delta$. Again, knowing the order in which channels are visited we could exploit this knowledge just as above.

Another line of thinking to pursue is to refine attenuation coefficient bounds dynamically by exchanging bounds on future call arrivals by analyzing future event lists. The current bounds, computed based on the state at time a, could include the effects of future arrivals. This idea was explored in [22], and would allow a processor not to block while another engaged in a probe.

The experiments we present here are derived directly from our serial method which, being an adaptive time-stepping scheme, is parallelizable in a natural way [40]. All empirical data presented henceforth was obtained with an SGI Origin2000, a shared-memory parallel machine. The model parameters used are the same as those of the experiment presented in Section 3.5. This time, however, we simulate a larger total of 32 channels, which we divided into pools of the same size across the processors, for a simulation horizon of one hour.

As the algorithm presented in Section 3.2 will, at times, force the parallel simulation to proceed in lockstep, we decided to run entire simulations in this fashion to have a performance reference. Each processor was allowed to compute the events in one time-step and then forced to pause at a barrier until all others were ready to proceed. Figure 3.20 gives the execution times as a function of step-size and the corresponding speedups.

Next, we implemented the parallel version of interval jumping, which makes the simulation alternate between time-stepping and jumping over intervals. Figure 3.21 presents the



(a) (b)

Figure 3.20: Execution time and speed up of lockstep parallel interval jumping

performance results obtained with this method. The speedups obtained so far (relative to sequential interval jumping) are consistent with those of the simple time-stepping method, and like in that case, are not as large as we'd like. For the interval jumping case, one of the probable causes is that probes are serial bottlenecks—parallelism is exploited only at power vector updates. However, this does not explain the disappointing relative speedup of straight time-stepping. These experiments, similarly to those in Section 3.5, were conducted with an older version of the simulator, which had serious performance bottlenecks. We are currently finalizing the implementation of a new parallel simulator, which we describe further ahead, that will should mitigate, if not eliminate, some of the performance problems of the first.

More often than not the success of a probe at b and admittance of a new call will *not* trigger handoffs in other channels, so our conservatism may be unwarranted. One could approach synchronization optimistically, allowing each processor to asynchronously attempt

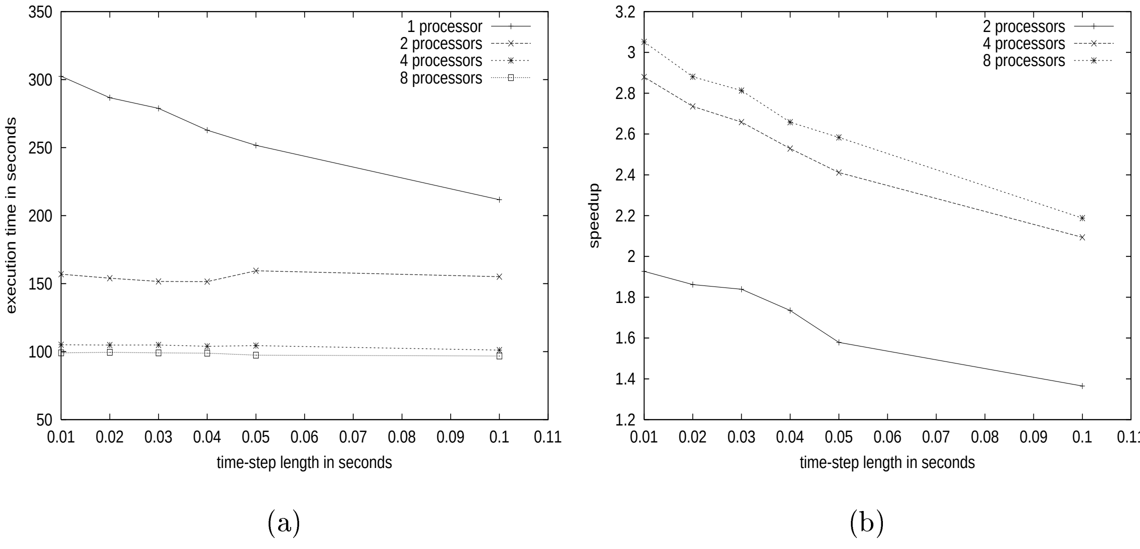


Figure 3.21: Execution time and speed up of next event parallel interval jumping

to interval jump based solely on its assigned channels. Let us consider the implications. A probe calculation at time b may need power levels from adjacent channels, possibly assigned to different processors, which may not yet have simulated up to that time. Since we know that the precise power state at b depends on those levels, to process optimistically is to make some assumptions which we hope will be correct and rollback only if they are not. One approach would be to have a processor responsible for channel c report all changes in usage of c to processors responsible for adjacent channels c' and c''. When a processor needs state information about a channel managed elsewhere, it uses the last known update and assumes it will not have changed. If the assumption turns out to be incorrect, a time-stamped update will arrive in its past to correct the assumption. Such an approach may work if one channel's interference does not depend on very many other channels, but this misses a key opportunity to exploit parallelism further. *Most state changes on one channel do not induce events on other channels.* What is needed is a mechanism so that a processor can apply a probe to channel c at time b using out-of-date information on channels c', and c'', but still come to the correct conclusion. This could be accomplished by having c's processor send messages to those of c' and c'' indicating worst-case assumptions being made about their state in c's probe, with the understanding that the recipient will respond with a NACK and more complete description of state, *only* if its state at that time is "worse" than assumed. Thus a rollback and corrective action occurs not if the noise values used in the probe were incorrect, but only if they were too low.

3.6 Chapter Summary

We started this chapter with the careful definition of our simulation model. Having identified radio propagation, user mobility, power control and channel allocation as major model components, we went about to describe each one of them in detail.

The analysis of the traditional simulation of this model revealed two principal bottlenecks. The first one, the costly interference computation required to simulate power control mechanisms, poses a problem for which we investigated two different solutions. The use of N-body problem algorithms was shown to produce substantial performance improvements in these computations. The relative errors produced in the process depend on how heavy the load (number of active transmitters) in the model is. We have shown that in situations when blocking probability is at most 10%, the Barnes-Hut algorithm produces relative errors comparable to the limited interference (truncated) method, but still requires more processing time. When the systems is loaded up to maximum capacity, the relative errors produced by BH can rise up to one order of magnitude higher than the truncated method.

We approached the second bottleneck, the regular and slow evolution of time-stepping, with the creation a new simulation technique we call *interval jumping.* Concluding the chapter, we stated the theoretical foundations of this method and shown encouraging initial results of its application in sequential and parallel simulation.

Chapter 4

Making Good Use of *Interval Jumping*

In the previous chapter we introduced interval jumping and discussed the foundations of the mechanism. A simulation based on the principles of the experiment we described in section 3.5 does not explore the full potential of the mechanism: once an interval is identified, it's either jumped in its entirety or not at all. In that first experiment, once we predict the possibility of a CE in an interval, the simulation gives up on the jump and time-steps through it. In this chapter we show that, when a probe fails, we can use this same predicate to perform a search for a viable jump point within the original interval. Given an interval of simulation time $[a, b]$, we discretize it into a set of points $\{x_i\}$. The problem we consider is how to devise an effective strategy for scheduling probes so that we can find the earliest point x_k for which a probe fails [36]. Once this point is found, we can then do an interval jump from a to x_{k-1}.

4.1 The Price of Efficiency

The probe predicate determines whether there exists the possibility of a CE occurring in an interval. We can use this as a tool not only to assess if an entire interval is viable. Alternatively, we can use probes to discover a viable subinterval within a range over which a jump would otherwise be unsafe. More than simply finding *just any* jumpable sub-interval, we want to find the largest one possible, since longer jumps translate into better performance. There may be several points x_k between $[a, b]$ for which $\mathbf{probe}(a, x_k) = \text{TRUE}$, but we are interested in finding one that allows for the longest jump possible without paying too high a cost.

Figure 4.1 illustrates the basic process: if $\mathbf{probe}(a, b)$ fails, we can probe different sub-intervals of $[a, b]$ until we have found the smallest k for which a probe at x_k also fails. We model this k as a random variable S: if the probe doesn't fail at b, we define $S = \infty$. The probability distribution of S is allowed to be general.

To complete the framework for a cost/benefit analysis of searching for $S = k$ in each interval [36], we must also define the cost of each probe. If $n = \frac{(b-a)}{\Delta} \in \mathbb{N}$ is the number of time-steps from a to b, we model the probing cost as a function $\phi(r, t)$: $\mathbb{Z}^2 \to \mathbb{R}$, for $r, t \in \mathbb{N}$ such that $0 \leq r \leq t \leq n$. Let x_r be the largest point for which we know that the probe succeeds. The state at x_r is used as the initial condition for the fixed-point computation in the probe; $\phi(\cdot)$ is defined to depend on this difference to support the idea that the temporal separation between x_r and x_t may require more iterations until convergence.

The question we investigate now is what we call the probe scheduling problem:

> Given an ordered set $\{x_i\}$ with $n + 1$ points, what is the sequence of probes that allows us to determine x_k while incurring in the smallest possible cost?

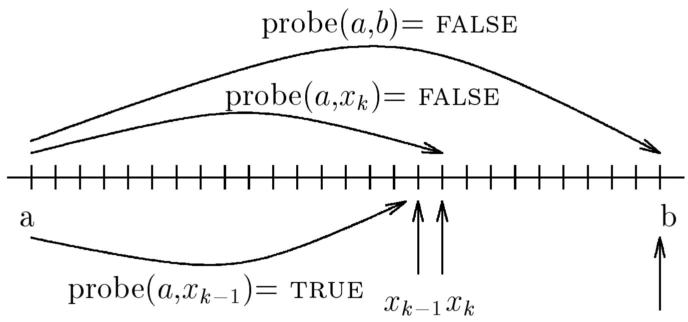


Figure 4.1: Looking for a good jump point

4.2 Controlling the Costs

The *state* of the search for x_k, in a given interval, can be described by an ordered pair (i, j) such that x_i is the largest time-step known to have passed the probe and x_j is the smallest time-step beyond which we will not consider probing. By definition, we say that a probe at x_0 is successful: remember that at the corresponding time a, no transmitter is at maximum power.

The search starts at $(0, n)$ and as we probe points in $\{x_i\}$, we gain knowledge of where x_k may lie, narrowing down the set of candidates to probe at future states. Associated with each state (i, j) is a *decision set* $\mathcal{D}_{ij} = \{x_{i+1}, x_{i+2}, \ldots, x_j\}$, and for each element in this set we can compute a probability that the probe at that point succeeds or fails.

This probability is derived from the distribution of S and reflects the knowledge that has been acquired with previous probes in this interval. If the search state is (i, j), for $0 < i < n$ and $i < j \leq n$, we have learned that the probe has succeeded at x_i and failed at x_j. Let $f(\cdot)$ and $F(\cdot)$ be the probability mass function and cumulative distribution function of S, respectively. Using the knowledge implicit in the state definition and the distribution of S, we can compute the probability of failing a probe at some point x_k as

$$q_{ij}^k \;=\; 1 - Pr\{S = k \mid S > i\} \;=\; 1 - \frac{F(k) - F(i)}{F(j) - F(i)}\;.$$

If $k = j$,

$$q_{ij}^k \;=\; 1 - Pr\{S = k \mid S > i\} \;=\; 1 - \frac{F(k) - F(i)}{1 - F(i)}\;.$$

As we will see ahead, the success or failure of the probe determines the state transition for the search process.

Described in these terms, the probe scheduling problem is, clearly, a sequential decision process and we can express the cost associated with each decision. The execution cost of the probe at x_k from state (i, j), as we have stated before, is given by $\phi(i, j)$. Let $C(i, j)$ denote the minimum expected cost (according to an optimal schedule) associated with the decision of probing at k from state (i, j). For some state (i, j), we look at $C^k(i, j)$, the minimum cost associated with the decision to probe at k, and write

$$C_{\mathrm{OPT}}(i, j) = \min_k \{C^k(i, j)\}\;. \tag{4.1}$$

That is, in order to progress to an optimal schedule, we always choose a decision that minimizes the cost from that state on. The structure of $C^k(i, j)$ is given by

$$C^k(i,j) \;=\; \begin{cases} 0, & \text{for } i = j \\ \phi(i,k), & \text{for } k = j \text{ and } \\ & j - i = 1 \\ \phi(i,k) \;+\; q_{ij}^k\, C_{\mathrm{OPT}}(i, k-1), & \text{for } k = j \text{ and } \\ & j - i > 1 \\ \phi(i,k) \;+\; q_{ij}^k\, C_{\mathrm{OPT}}(i,k) \;+\; (1 - q_{ij}^k)\, C(k,j), & \text{for } k < j \text{ and } \\ & j - i > 1\;. \end{cases} \tag{4.2}$$

The first case says that when $i = j$, we are at a terminal state (i, i) and need not probe any further. The second case says that from state $(i, i+1)$, the only possible decision is to probe at x_{i+1}, which leads to a terminal state. The third case says that from state (i, k) if we probe at k and succeed, we enter terminal state (k, k), otherwise if we fail the search continues from state $(i, k-1)$. Finally, the last case covers the possibilities which arise when the point probed lies inside an interval that has more than just two points.

The peculiarities of different models for wireless cellular systems determine that the properties of $\phi(\cdot)$ and q_{ij}^k. We cannot rely on any special features of these functions because we do not know what to expect when different models are simulated, or when varying parameters are used for the same model. Instead of solving this equation by brute force, we observe the problem is amenable to a solution by dynamic programming (DP) [6, 14]. Equation 4.2 exhibits the two characteristics that make this possible:

1. *optimal substructure*: the optimal solution for an interval $[x_l, x_m] \supset [x_i, x_j]$ is derived using the optimal solution to subproblem $[x_i, x_j]$, and

2. *overlapping subproblems*: in the search of an optimal solution for interval $[x_0, x_n]$, several different subproblems may use the solution to some specific subproblem $[x_i, x_j]$.

The obvious solution to this optimization problem is achieved through *recursive fixing*, a standard dynamic programming technique. The solution yields an optimal policy, henceforth referred to as OPT: from every state in the decision process, we will know which point to probe so that we reach the end of the search with the smallest possible cost.

Note, however, that the time-scale for power control is several orders of magnitude finer than the time-scale for *call arrivals* and *hand-offs*. Since time advances are done much more frequently than teletraffic events occur, if one is to derive an optimal search policy on the fly for each interval jumped, the computational costs would make the simulation impractical.

Another difficulty would be the estimation of a probability distribution for S. For now we carry on with a theoretical study considering a variety of forms that this distribution may take. In the next section, we return to the more practical problem and discuss what shape this distribution assumes in real simulations.

4.3 A Quick Decision is a Good Decision

Computing the OPT policy is not cheap. Consider that we work with an interval of $N = n+1$ points and we organize subproblems as a matrix, as is common practice in dynamic programming. To compute the solution, we need an $N \times N$ matrix. Each element (i,j) in the matrix corresponds directly to the equivalent state in the search process and contains the cost of computing OPT for sub-interval $[x_i, x_j]$.

Say we number the diagonals in the upper part of the matrix according to the size of the subproblems it collects, that is, the size of the associated decision sets. Discarding the trivial subproblems of size 0, diagonal g corresponds to subproblems (i,j) such that $j-i=g$. There will be $N-1$ such diagonals. Each one will contain $N-g+1$ subproblems and each subproblem will contain g probing points. The total number of points to test in the computation of OPT is

$$\sum_{g=1}^{N-1}(N-g-1)\,g = \frac{N^3-3N^2+2N}{6} = O(N^3)\ .$$

Halving N reduces the computational work of solving the DP equation by a factor of eight. This observation leads one to intuit that *binary search* is a policy that could work well for this problem since, at each step of the search, the workload would be halved. This claim may prove to be true under certain conditions. In order to lay the groundwork for this verification, and also for further steps in this study, we must state a few more assumptions.

For all $i < j$, we consider three different models for the probe cost function $\phi(i, j)$:

i. $\phi(i, j) = \tau$, where τ is some real valued constant,

ii. $\phi(i, j) = \tau \ \log(j - i + 1)$, and

iii. $\phi(i, j) = \tau \ \log(\log(j - i + 1) + 1)$.

The first case corresponds to scenarios when the probe converges so fast that its cost does not change with the length of the interval. In the second case, the logarithmic model covers situations when the gain matrix for the fictitious system is significantly different from the one used to compute the fixed-point at x_i. Finally, the third case applies to situations similar to those of (ii), but where the cost of the probe rises much more gently with the length of the interval.

Let x_S be the earliest point in $\{x_i\}$ where a probe fails. We consider two families of stochastic models for placement of this point. The first one is the conditional uniform: with probability p we have $i < S \leq j$, and conditioned on this event the distribution over $[i, j]$ is uniform. The second family is defined in terms of a linearly increasing hazard rate function $h(k)$ for $0 < k \leq n$. We make the slope of the hazard rate function the parameter that distinguishes a member in the family: we can define it to be steep enough to ensure that a probe fails within $[a, b]$ or make it gentle enough to create the opposite effect (see Figure 4.2).

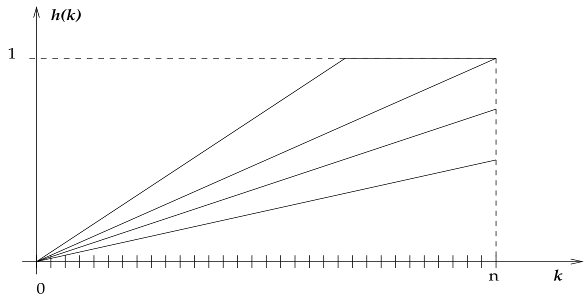


Figure 4.2: A family of linear hazard rate functions

Hazard rate functions, however, are usually defined for stochastic processes in continuous time and in this case we are concerned with a process in *discrete time.* Translating the idea to this different domain, we write $h(k)$ as

$$h(k) = \Pr\{S = k \mid S \geq k\} = \frac{\Pr\{S = k\}}{1 - \Pr\{S < k\}} = \frac{f(k)}{1 - F(k-1)} \; .$$

Then, from this definition, we compute the probability distribution of S over $[0, n]$ as

$$f(k) \;=\; h(k)\;(1 - F(k-1)), \;\; \text{where } h(0) = f(0) = F(0) = 0 \; .$$

For the case when $\phi(i,j) = \tau$ and the distribution of S is uniform, binary search is an optimal policy. Writing the functional equation to compute the cost of binary search, we have

$$C_{\mathrm{BS}}(i,j) \;=\; \begin{cases} 0, & \text{for } \; i = j \\ \phi(i,j), & \text{for } \; j - i = 1 \\ \phi(i,k) \;+\; (1 - q_{ij}^k)\, C_{\mathrm{BS}}(i,k) \;+\; q_{ij}^k\, C_{\mathrm{BS}}(k,j), & \text{for } \; j - i > 1, \\ & \text{where } \; k = \frac{i+j}{2} \; . \end{cases} \tag{4.3}$$

Equation 4.3 says that when the search reaches a terminal state (i,i), the process terminates. If it reaches a state (i,j) such that x_i passes the probe and $x_j = x_i + \Delta$, then we probe at x_j and enter a terminal state. Now, if the subinterval is at least 2Δ long, we probe its midpoint and continue searching on the left if the probe fails, or on the right if it succeeds.

Solving equations 4.1 and 4.3, for constant probe cost model and uniform distribution of S, the resulting cost is the same in both cases. The conclusion that, under these circumstances, binary search is an optimal policy is important due to the fact that it allows a jumping simulation to simply *use* the same search policy for every interval. There would be no need to compute the solution to the DP equation every time a jump was attempted. This would allow for a substantial reduction of execution time when an interval jumping simulation is used.

It is unrealistic to expect, however, that every model of a wireless system would conform to these special characteristics. Since different $\phi(\cdot)$ and $F(\cdot)$ are determined by the very nature of the simulation model, we must look for strategies that can adapt to changes in these parameters while maintaining good performance for interval jumping simulations.

Our motivation is either to use well established search policies or, lacking a better alternative, to *derive* good policies on demand during the simulation. Given that the time to compute OPT for every interval can be prohibitive, we investigate strategies which cost less to derive and can bring the search to a "good enough" jump point in just a few probes. Next, we discuss the several possibilities we have investigated.

4.4 Towards the Goal

The scheduling strategy we now propose, called MULTI, breaks up one large DP problem into smaller, more manageable ones. By solving these reduced problems, we can find the

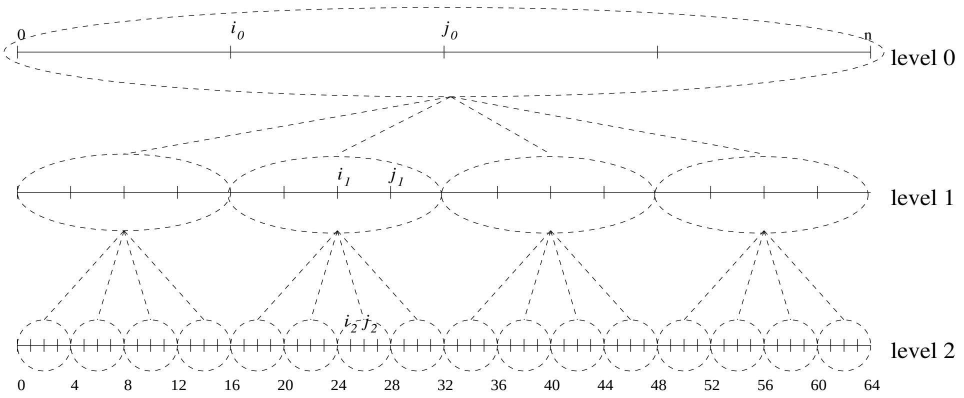


Figure 4.3: Point grids and search tree in MULTI

same x_S that the optimal scheduling would, paying a slightly higher probe execution cost, but spending much less time computing the schedule itself. This drastic reduction in the total costs makes dynamic schedule computation (for every interval) a viable goal.

For the sake of simplicity, we work with numbers of points n that are powers of two, that is, $n = 2^L$. Assume that the original problem has $N = 2^L + 1$ points and that d is any one of the integer divisors of L. Now, rather than discretize $[a, b]$ into $2^L + 1$ points at a distance of Δ from each other, let us discretize it more coarsely into $B = 2^d + 1$ points, $2^{L-d}\Delta$ apart. Call this grid *level* 0. We can define progressively finer grids for levels $l = 1, 2, \ldots, (L/d) - 1$, such that, in each level, we find a total of $2^{(l+1)d} + 1$ points separated from each other by $2^{L-(l+1)d}\Delta$. Also, for each level l, we construct 2^{ld} sets of B points each.

Figure 4.3 illustrates this concept of *multiple levels*, explaining the name we chose for this scheme. The figure shows an interval with $L = 6$, that is, 65 points, for which we chose $d = 2$. At level 0 we see only five points, all in the same set. At level 1 we see four sets, each with five points. At level 2, we see 16 sets, also with five points each. Note that points at the extremes of subintervals may belong to more than one set.

Assume that for each level and each set of points, we compute, through DP, an optimal policy to identify the earliest point on that grid that fails the probe. Starting at level 0, where there is only one set of points, we take a $B \times B$ matrix and set out to compute the optimal policy for this grid. Once this policy is determined, we use it to discover j_0, the earliest point in the grid that fails the probe. If i_0 is the point that immediately precedes j_0 on this grid, we will know that the critical point x_S lies within $(i_0, j_0]$. Next, we move down to the set at level 1 defined by this sub-interval, reuse the same $B \times B$ matrix employed at level 0, and repeat the process. This new step yields a narrower interval $(i_1, j_1]$ that contains x_S. As illustrated in Figure 4.4, we can continue moving down to further levels until some criterion is satisfied. The stopping condition may be reaching the deepest level $l = (L/d) - 1$, where we find x_S in the finest grid. Alternatively, we may stop the search any time we consider that an arbitrary maximum probing cost has been paid, or yet, when we conclude that a sufficiently long jump has been determined. This search scheme allows us to have tight control on the ratio of cost of searching to the benefit of making a longer jump.

This search scheme requires that we solve L/d DP problems, one for each level visited. The good news is that these problems are much smaller than the original, requiring only time $O(B^3)$. That is, we leave behind one big, costly DP problem and instead solve smaller versions of it incurring a total cost $O(B^3)$, that is orders of magnitude smaller than the original $O(N^3)$. As an added bonus, in the end, we are guaranteed to discover the same point x_S that OPT would have found.

A noteworthy aspect of this method is that it describes not just one solution strategy, but encompasses a family of them, depending on the choice of d. At one extreme, when

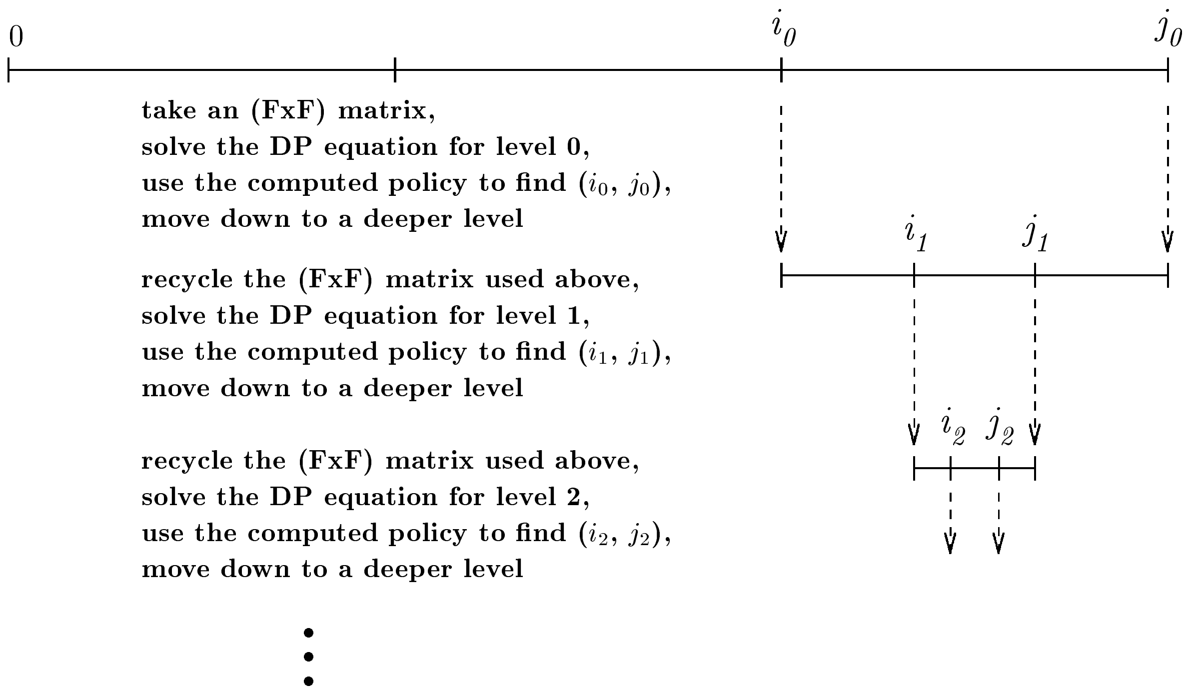


Figure 4.4: Algorithm for MULTI search

$d = L$, there is a single level with a fine grid and the computation of the probing schedule is identical to that of OPT. At the other extreme, when $d = 1$, each level has only three points—two endpoints plus one in the center— and this leads to a policy very similar (but not identical) to binary search. Between these extremes we have a spectrum of policies; their computation costs increase with d and so does the mean probing cost required to find x_S.

4.5 Performance Under Varying Scenarios

The question of how different probe cost models and probability distributions of S affect the cost of computing probe scheduling policies motivated the series of experiments we now

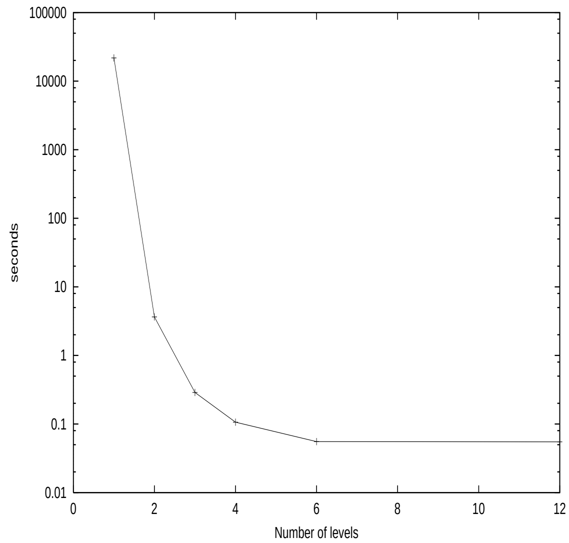


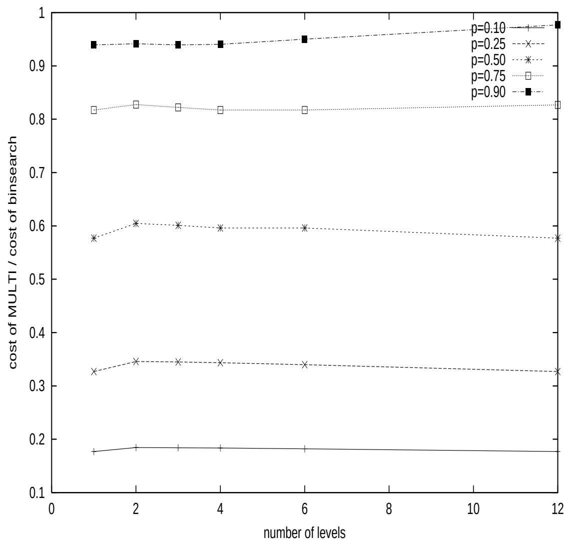
Figure 4.5: Execution cost of computing MULTI

describe. All of these experiments were run on an SGI Origin2000 and consider a problem with $N = 2^{12}+1 = 4097$ points on the finest grid. The costs we report in these experiments reflect those of a priori calculation of the whole hierarchical policy, that is, policies for all the point sets at every level. Note that these are even higher than what we'd have in a dynamic version that would calculate the policy for each successive grid only when the search process must be carried out.

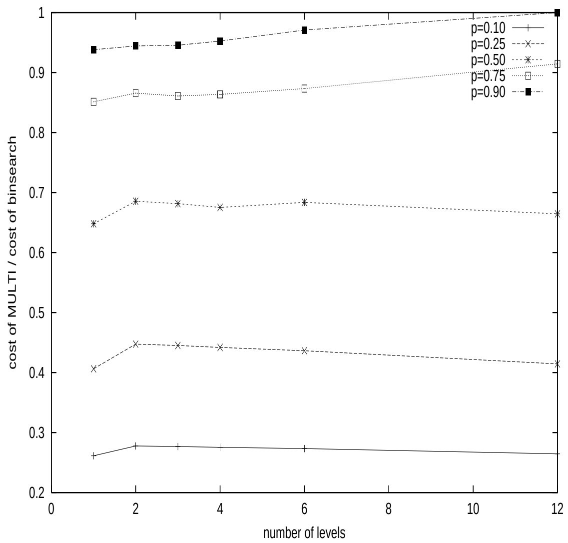
Using a conditional uniform with $p = 0.5$ to model S and probing cost $\phi(i, j) = \tau$, we first investigated the execution of policy computation for MULTI. Figure 4.5 plots the time for policy computation as a function of the number of levels. Recalling that computing MULTI with a single level corresponds to computing OPT, we see what a dramatic difference our scheme can make. While it takes six hours to compute OPT, it takes only 50 milliseconds to compute the hierarchical policy that discretizes each interval into three points. The computational savings are enormous. The question to ask, however, regards the quality of the quickly computed policy: "How does it compare *relatively* to OPT and to binary search?"

In order to investigate this question of relative performance, we have experimented with different probabilities p of finding a critical event in the interval. We have also studied how relative costs vary as we change models for probe cost and probability distributions of S.

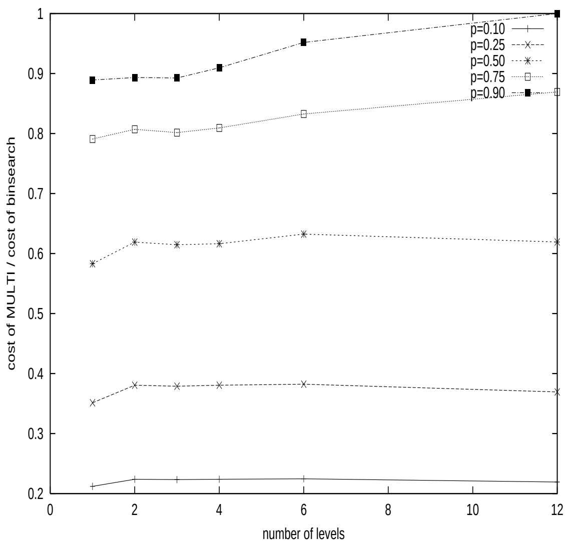
Figure 4.6 shows the performance of the hierarchical search relative to that of binary search for various values of p. These plots demonstrate our results for the three different probe cost models when a conditional uniform is used to model the critical event distribution. Figure 4.7 shows results obtained when we repeated the same study using a linear hazard rate to determine the critical event distribution. The overall picture is that all results are qualitatively similar. Performance relative to binary search is insensitive to the number of levels employed and also to the probability distribution of CEs. However, we see that it depends very much on the likelihood of finding a CE in the interval.



(a) Constant probe cost

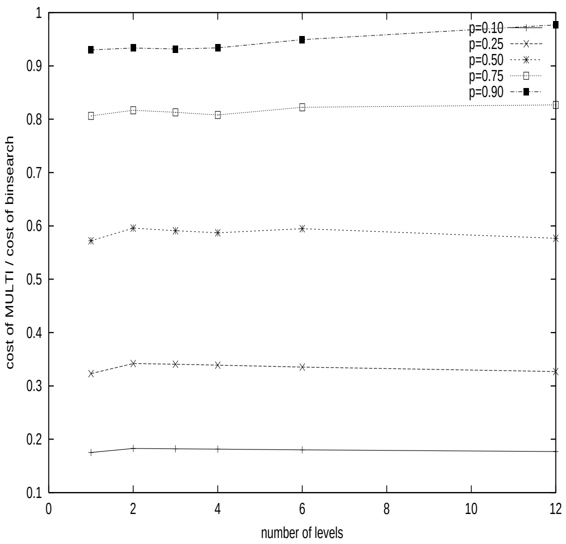


(b) Logarithmic probe cost

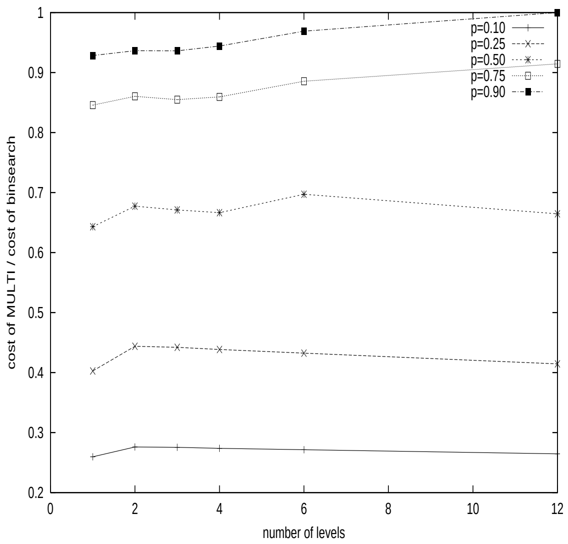


(c) Log-logarithmic probe cost

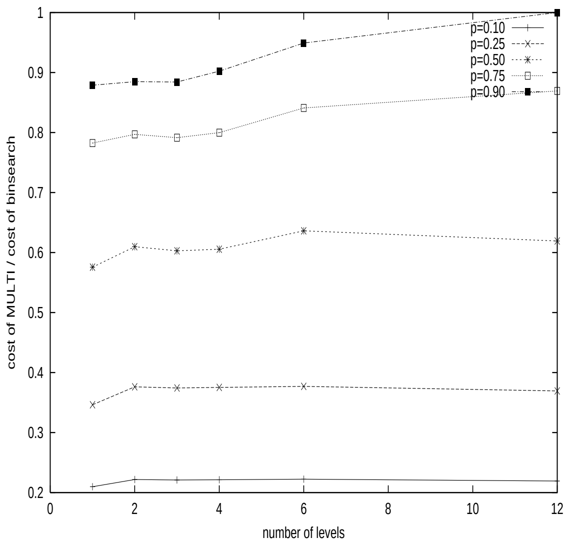
Figure 4.6: Performance of MULTI relative to binary search under a conditional uniform CE distribution



(a) Constant probe cost



(b) Logarithmic probe cost



(c) Log-logarithmic probe cost

Figure 4.7: Performance of MULTI relative to binary search under the CE distribution determined by a linear hazard rate function

The main difference between binary search and MULTI lies in the fact that when p is low, the latter is smart enough to exploit this knowledge. While binary search always probes at the interval's midpoint, MULTI encourages probes at the right end of the interval. If the chance of observing a CE is small, the better search policy will attempt to determine right away whether a particular interval has a CE or not. A successful probe near the end of the interval leads to a terminal state in the search in fewer steps than required by binary search. If a CE does not happen in $[a, b]$, however, MULTI behaves very much like binary search and thus the costs turn out to be similar.

Since wireless cellular systems are engineered to minimize unscheduled channel reassignments, most simulation models of interest will exhibit low p. This fact works to our advantage in that the use of MULTI proves to be beneficial exactly under those circumstances. The added effort of managing the hierarchical policy offers a substantial reduction in probe costs over binary search.

Binary search loses its allure compared to policies that exploit the probabilistic knowledge of critical event placement. This policy is oblivious to this information and can't help but blindly narrow down the search interval. It turns out that a very simple modification can render the performance of binary search substantially better. The new policy we describe now is then called *modified binary search* (MBS) for obvious reasons.

If we know that p is high, probing the very last point in the interval right at the start of the search is a better idea than probing the mid point. In the worst case, this probe will fail and we will have to apply binary search to the interval. Compared to binary search, the total cost increases by only one extra probe. If this first probe succeeds, however, the search immediately terminates and the total cost of the search is *only* the cost of this probe.

For an interval $[i, j]$, we write the expected cost of MBS as

$$C_{\mathrm{MBS}}(i,j) \;=\; (1-p)\,\phi(i,j) \;+\; p\,\left[C_{\mathrm{BS}}(i,j) + \phi(i,j)\right].$$

For the case of $\phi(i,j) = \tau$, it is easy to show the conditions under which MBS is a winner compared to binary search. We are interested in the case when the ratio of the cost of MBS relatively to binary search is less than one, that is,

$$\frac{C_{\mathrm{MBS}}}{C_{\mathrm{BS}}} \;=\; \frac{(1-p)\,\tau \;+\; p\,(C_{\mathrm{BS}} + \tau)}{C_{\mathrm{BS}}} \;<\; 1.$$

Thus, MBS is a good alternative to binary search as long as

$$\tau \;+\; \frac{p\tau}{1-p} \;<\; C_{\mathrm{BS}}.$$

Now, if $C_{BS} \;=\; \tau\, \log_2 N$, we can rearrange the inequality above and conclude that MBS performs better than traditional binary search for values of p such that

$$p \;<\; \frac{(\log_2 N) - 1}{\log_2 N}$$

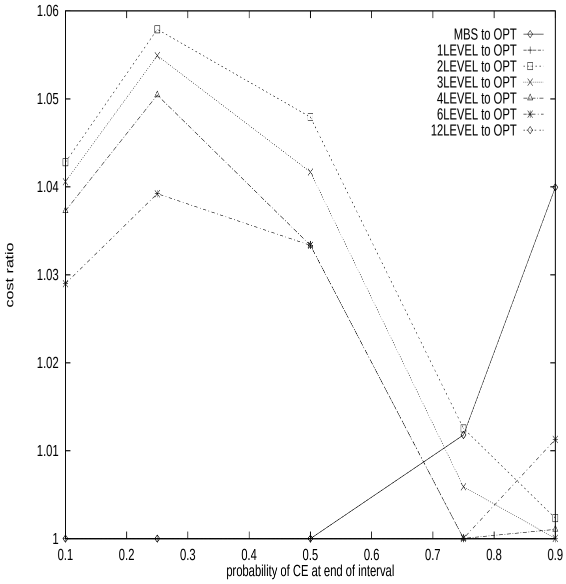
In large models of wireless cellular systems, the increase in the number of cells and the number of mobiles cause the cost of each probe to be substantial. When this is the case, each additional probe in the search significantly increases the cost of the whole process. Compared to MULTI, we can expect MBS to yield good performance when the cost of each probe is small relative to the effort of solving the dynamic programming problem for each level. Since $\phi(i,j)$ can be substantially affected when the model scales up, it is clear that the method that probes the least will be the most advantageous and, thus, in that case, MULTI should be a winner.

As evidenced in Figures 4.8 and 4.9, the cost of MBS matches the cost of OPT for a variety of different scenarios. The greatest asset MBS has to offer is that its performance comes almost for free: we don't need to know details about CE placement in the interval, only the probability it occurs.

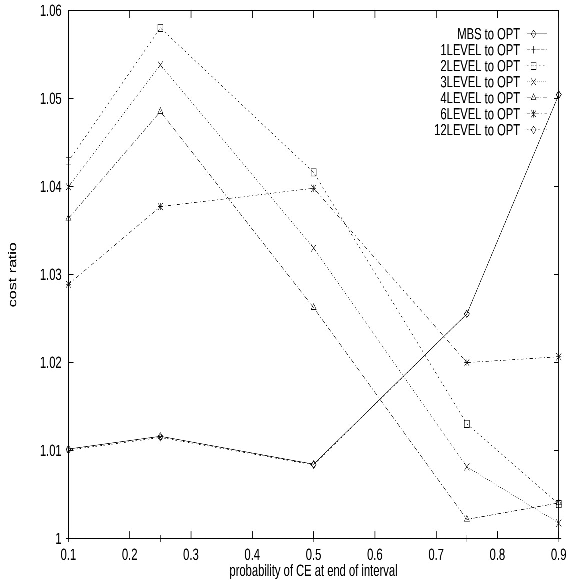
Next, we study the performance of MULTI relative to that of OPT. The graphs in Figures 4.8 and 4.9 explore the six combinations of probe cost and CE placement that we considered. Each point series corresponds to the ratio of mean cost of MULTI to mean cost of OPT as we vary the probability that the interval contains a critical event. In addition to plotting one curve for the number of levels determined by each of the integer divisors of $L = 12$, we also plot the performance of MBS relative to OPT. The most obvious conclusion to be reached from these plots is that, in our experiments, the performance degradation due to using MULTI instead of OPT is, most commonly, less than 10% and in several cases, *much* less. In light of our previous observation that MULTI performs significantly better than binary search, this added fact demonstrates the utility of our approach.

With regard to the shape of these curves, we again see that the probability of having a CE in the interval is a strong determining factor. When this value is low, using MULTI with more levels is more advantageous than using fewer levels (with the exception of 1-level MULTI which is indistinguishable from OPT). Having a coarser grid, with fewer decisions available in policy calculations, creates a stronger drive to probe the points towards the right end of the interval: a successful probe early in the search process leads to a fast arrival at a terminal state. This way, a final decision is reached without need to descend to lower and lower levels. When this probability grows large, the situation gets inverted and the policies with more points per sub-interval are more efficient at honing in on the critical event. Intermediate values of this probability cause the curves to intersect at various points.

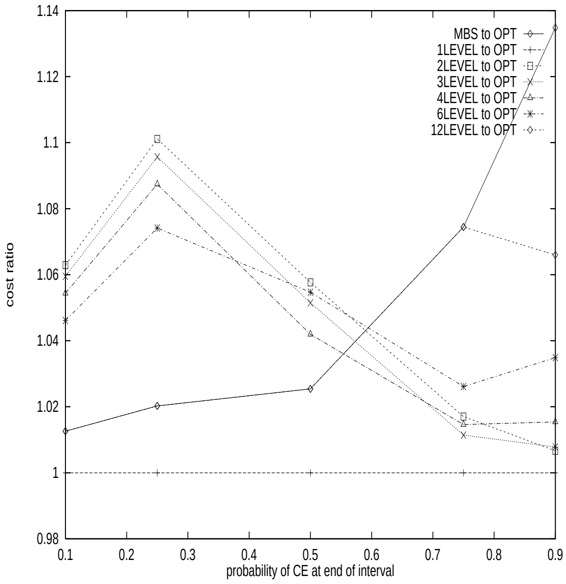
The final message in this data is the fact that all the variations of probe cost model and placement of CE in the interval have only a second-order effect on the relative performance of our method. Since MULTI has been shown to be fairly insensitive to changes in these parameters, we can expect that it will behave similarly under different simulation models. Obviously, this expectation is warranted only if the models don't deviate too much from our assumptions.



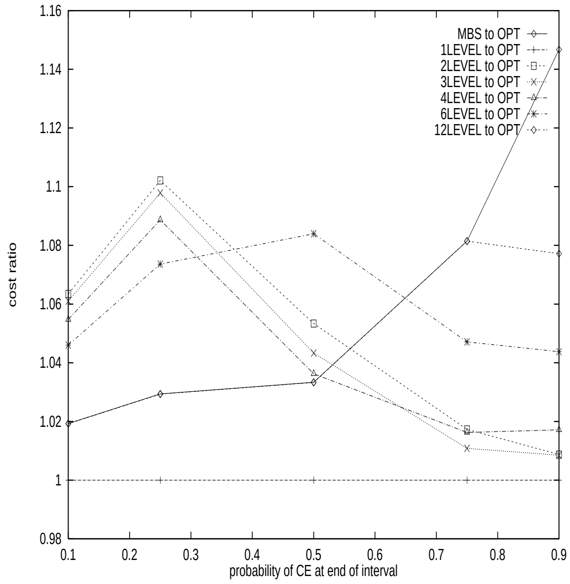
(a) Constant probe cost, conditional uniform CE distribution



(b) Constant probe cost, linear hazard rate function

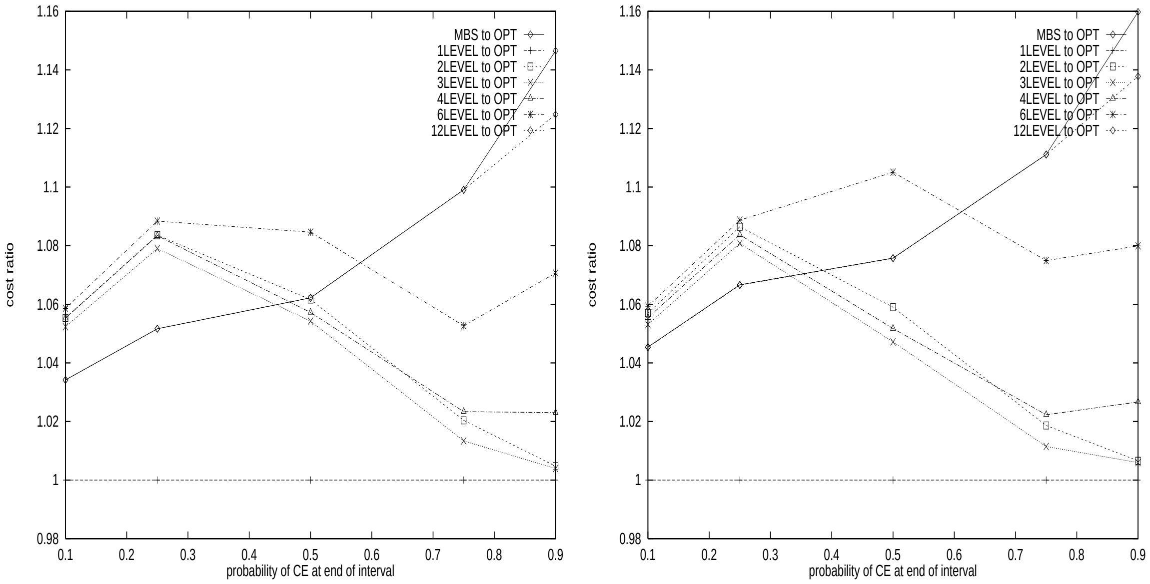


(c) Logarithmic probe cost, conditional uniform CE distribution



(d) Logarithmic probe cost, linear hazard rate function

Figure 4.8: Performance of MULTI relative to OPT



(e) Log-logarithmic probe cost, conditional uniform CE distribution

(f) Log-logarithmic probe cost, linear hazard hazard rate function

Figure 4.9: Performance of MULTI relative to OPT (continued)

4.6 Modeling Probe Cost and Probability Distribution

The theoretical study we have just presented gave us guidelines on how to use efficiently a hierarchical search policy developed to aid interval jumping. We have shown that MULTI is relatively inexpensive to compute and that it leads to search costs that are fairly insensitive to different probe cost models and probability distributions of critical event placement. This section reports our efforts to put the theory to practice and to evaluate the impact of the previously developed results in the context of a real simulation.

The most typical simulation of a wireless system uses model parameters that keep the probability of a call block on the order of 1%. At the system design stage, it is common for engineers to experiment with different settings (such as number of channels, cell size, etc.) until the performance can be fine-tuned to the desired level. It is, therefore, reasonable to expect that simulations of these models will have to deal with a varying range of parameter values.

The acceleration obtained with interval jumping simulations is highly dependent on the average length of time that can be skipped over. Higher probabilities of call blocks and drops indicate that critical events happen more frequently and, as a consequence, intervals that can be jumped over become shorter and scarcer. In order to provide a fair experimental scenario to evaluate interval jumping, we have chosen to tune model parameters to induce blocking probabilities of around 10%, one order of magnitude higher than the most typical. This should allows us to verify whether our techniques produce good results throughout a reasonable range of parameter variations.

Differently from the experiments described in Section 3.5, the bounds for attenuation coefficients $a_{ij}^c(k)$ used here are tighter estimates of the worst circumstances observed in the interval. Although we have not attempted to quantify the effect of these bounds empirically, it should be clear that the increased accuracy qualitatively improves the performance of the interval jumping simulation.

All the experiments we report from this point on refer to the same scenario unless otherwise noted: a single channel network of 10 by 10 cells, 20 mobile stations placing new calls at a rate of 3 calls per hour (according to a Poisson stream) with call duration times exponentially distributed with a mean of 3 minutes. All mobiles have a constant speed of 10 m/s and maximum transmitter power of 88dB. The thermal noise experienced by receivers was set to -120dB, system SIR was set to 12dB, and signal strength decay was set to the fourth power of the distance from the transmitter. The time-step length was set to 0.01s. As with all other experiments we conducted for this study, our simulator was run on an SGI Origin2000 with 4G bytes of memory and 180MHz processor speed.

These settings place the blocking probability of the system in our desired range. Under these conditions, we conducted a series of experiments and generated data to fit approximate models for $\phi(\cdot)$, the probe cost, and $F(\cdot)$, the probability distribution of critical event in an interval. While it seems counterproductive to have to make several slower runs in order to collect statistical information to accelerate program execution in the future, this approach has its attractions. Significant changes of load on the wireless system should cause correspondingly drastic changes in $\phi(\cdot)$ and $F(\cdot)$. When load variations are constrained to a small interval, however, the models derived from preliminary runs should remain valid and can be employed in many production runs of interval jumping simulations. Our first step in this analysis was, therefore, to produce data from which we could derive approximations

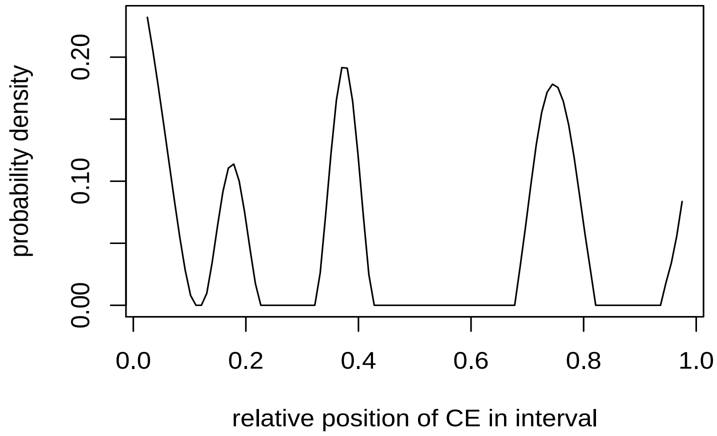


Figure 4.10: Estimation of the probability distribution of critical event within a normalized interval

for $\phi(\cdot)$ and $F(\cdot)$.

Making repeated runs of 10^5 seconds of simulated time, we recorded the length of each interval identified, together with the relative location of the CE, whenever one was found. For each interval, we noted the number of active MSs and tallied the data into three different histograms, each one for a different number of ongoing calls (4, 5 and 6). The number of samples obtained for each case allowed us to observe standard deviations of less than 1% for each histogram bin. Each of these histograms was constructed with 30 bins; taking the midpoint of these and fitting a smooth curve through them using the standard spline method, we obtained a set of estimates of probability density. Rescaling each curve so that the area underneath adds up to unity, we obtained three almost identical distributions. The curve in Figure 4.10 shows the probability distribution of the relative position of CE. Since all three curves overlap almost perfectly, we display only the one for five simultaneously active MSs.

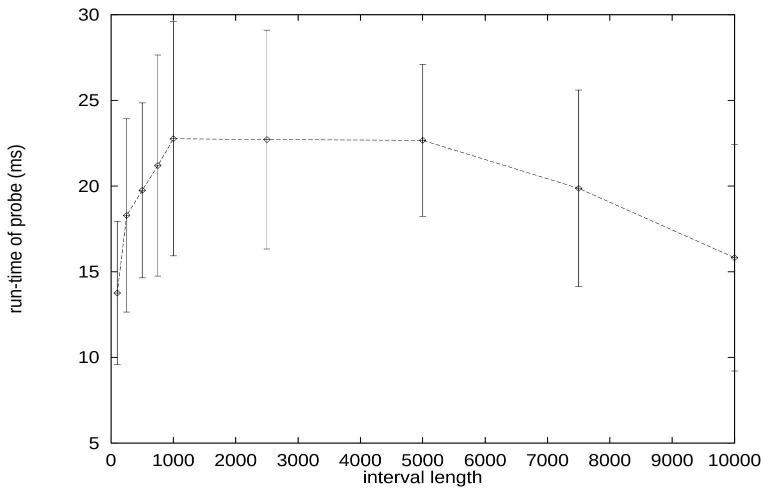


Figure 4.11: How the run-time of **probe** varies with the length of the interval

Arbitrarily choosing this same curve, we sampled it at 100 regularly spaced points and used this set of values to describe the probability distribution in the simulation. These values are read into an array that defines $f(\cdot)$, the probability mass function of S, before the simulation run begins. From this array we construct another of equal length for $F(\cdot)$, the cumulative density function of S. Whenever we need the density for a point that lies between two consecutive samples in $F(\cdot)$, we compute an estimate for the value using linear interpolation.

To seek out a model to describe the cost of probing an interval $[a, b]$, we recorded the execution time of the function that evaluates the predicate. Once more, in the name of consistency, we were careful to enough to record the number of ongoing calls at each observation. Keeping only the samples for five simultaneous calls, we computed the average probe cost for each interval length. The results are reported in Figure 4.11. Since the shape of the curve resembles a logarithmic function, we adopt this model for the cost of probing when we use DP to derive search policies during the simulation.

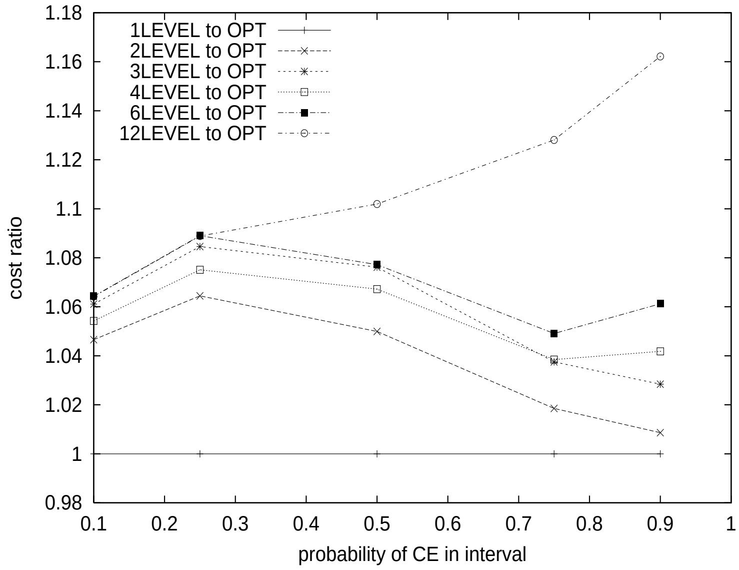


Figure 4.12: Performance of MULTI for empirical probe cost function and probability distribution

Using the same framework from Section 4.5, we computed the costs of performing MULTI search with our estimated probability distribution and probe cost model. Plotting the costs observed with varying number of levels relatively to the cost of the optimal policy OPT, we see that Figure 4.12 shows a behavior similar to the theoretical predictions in Section 4.5. The performance of MULTI with 2, 3, 4 and 6 levels doesn't vary more than 4% for changing values of the probability of finding a CE in the interval. The relative cost of 12-level MULTI, which is equivalent to MBS, on the other hand, starts to deviate from the other curves more significantly at $p = 0.5$.

4.7 Experimental Results

When the search for a jump point is incorporated into the picture, the interval jumping algorithm is slightly modified. Figure 3.15 shows that, after an interval is identified, we

```
algorithm IntervalJumpingSimulation
begin
  while (! simulation terminated) do
    compute interval [a, b]
    if (b − a > mΔ)
      use search to select jump point c
      if (c − a > mΔ)
        advance time to c
        update mobiles' positions to time c
        while (! power levels converged) do
          compute noise
          update power levels
        end while
        apply LADCA algorithm
        run time-stepped simulation for Γ steps
      else
        run time-stepped simulation to c
    else
      run time-stepped simulation to b
  end while
end IntervalJumpingSimulation
```

Figure 4.13: Modified algorithm for *Interval Jumping* simulation

must decide whether it is long enough to offset the costs of jumping. In case it is, we use some kind of search policy to determine if there exists a point $c \in [a, b]$ where a critical event may happen. Upon finding this point, we check if a jump from a to c is worthwhile. If that is true, the algorithm proceeds exactly like the original interval jumping. Otherwise, the simulation progresses in time-steps from a to c and only after that is a new jump attempted.

Our set-up for the hierarchical search was one that incorporated lessons learned from the previous analytical study. We observed that for the same probability distribution of S and probe cost model, a higher number of levels leads to reduced search costs. Given an interval $[a, b]$, we start by discretizing it into equidistant points separated by Δ units of time. This yields a set of N^* points, where each element corresponds to the instant when a state update would be made in the time-stepping simulation. We then compute 2^L as the next integer power of two greater than N^* and set $N = 2^L + 1$.

Recalling that the number of levels in a hierarchical search is defined by $l = L/d$ where d is an integer divisor of L, we have to make choices. Taking $d = 1$, the costs relative to the optimal policy would match those of binary search. Figures 4.8 and 4.9 indicate that for higher probabilities of CE in the interval, $d = 1$ should lead to correspondingly higher costs. From our estimated $F(\cdot)$, we learned that x_S tends to get concentrated towards the end of the interval and so we chose to avoid $d = 1$ in the face of other options (that is, whenever L is not a prime number). Our implementation picks $d = 1$ if L is prime, otherwise, it picks the smallest integer divisor greater than one and smaller than L.

We have experimented with two different search policies, MBS and MULTI, measuring the total run-time for each case varying only the simulation horizon. For interval jumping with each different search policy, we plotted the evolution of run-time with increasing simulation horizon, as shown in Figure 4.14. The data shows that, in this scenario, the overall costs of MULTI render it less efficient than MBS.

A deeper look into this problem set-up tells us why MULTI loses to MBS. If one works with a simulation model where the cost of probing is small relatively to the cost of constructing a probing schedule, the overall cost of using a straightforward policy such as MBS will always be small. When there is no schedule to compute, the total cost incurred is only that of the *application* of the policy. When probes don't cost very much, the simple policy may perform better than MULTI, even if the average number of probes determined is higher. On the other hand, models for which the computational cost of each probe is high benefit

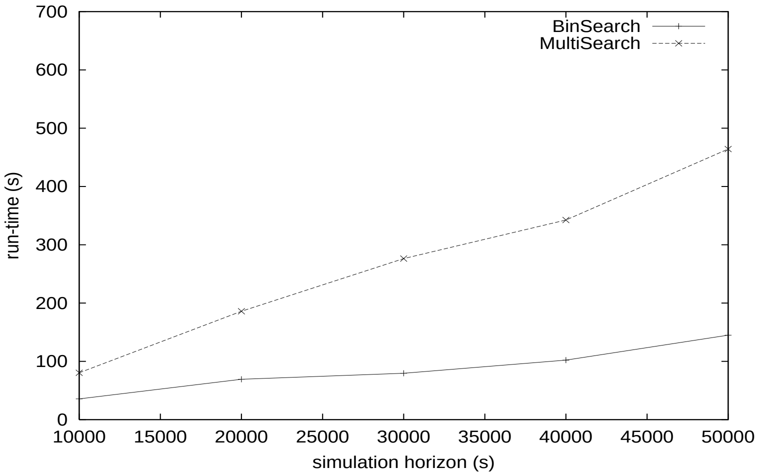


Figure 4.14: Run-time of interval jumping simulations of varying lengths using different search policies

much from a policy that reduces the average number of probes, because the added effort in generating efficient schedules is offset by the reduction in the cost of probing.

We can demonstrate, with a simple performance model, the importance of using MULTI in the simulation of models with high cost of probing. Consider that for an interval with N points, the average cost of doing one probe is equal to $\overline{\phi}(m)$, where m is the number of active mobiles in the system. We make this average cost of probing a function of m to reflect the fact that the computation, as described in Section 3.4.2, is heavily dependent on this parameter. When one scales the model up, increasing m, the cost of doing a single probe becomes more and more significant.

Let the effort of computing each probing schedule in a MULTI search with l levels be given by $X(l)$, the time taken to test all points in the minimization process corresponding to the computation with dynamic programming. Note that once l is fixed, the search computes l schedules for grids of the same size, one schedule for each level. Say we express costs using the average cost of doing a probe as our unit of measuring. The ratio $X(l)/\overline{\phi}(m)$

will then tell us how many probes one would be able to do for this model during the schedule generation. Now, if we let $\pi_i(l)$ be the average number of probes that the schedule determines for level i in an l-level MULTI search, we can approximate the average cost of applying MULTI by

$$C^*_{\text{MULTI}}(m,l) \;=\; \sum_{i=1}^{l}\left[\frac{X(l)}{\overline{\phi}(m)} \;+\; \pi_i(l)\right].$$

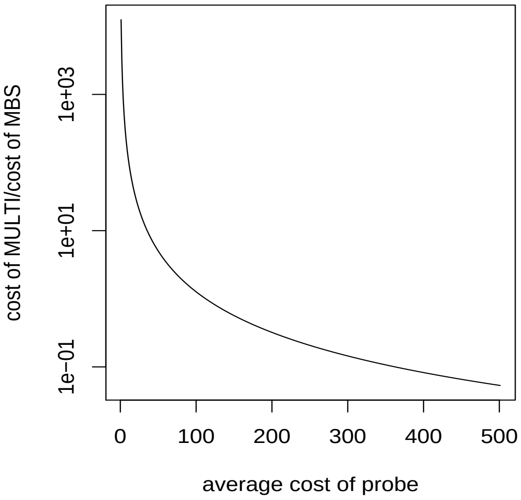
Rewriting this expression we have

$$C^*_{\text{MULTI}}(m,l) \;=\; l\,\frac{X(l)}{\overline{\phi}(m)} \;+\; \sum_{i=1}^{l}\pi_i(l).$$

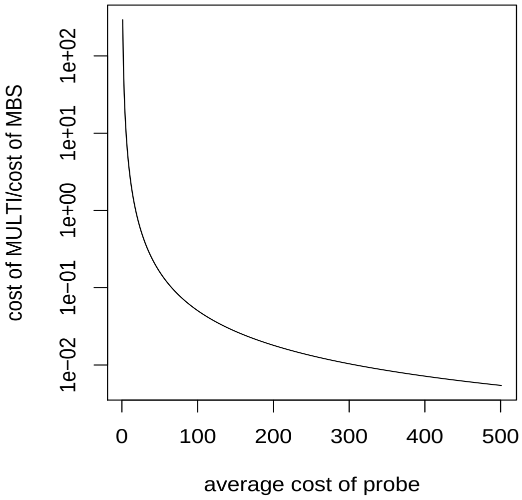
From the same framework used in the analysis that produced Figure 4.12, for a fixed value of $p = 0.5$, we obtain values of $\pi_i(l)$ that we can use to compute $C^*_{\text{MULTI}}(m,l)$. We can then show how this cost varies under arbitrary values of $1/\overline{\phi}(m)$ and this allows us to predict how MULTI performs as the model size is scaled up. Approximating the cost of MBS by

$$C^*_{\text{MBS}}(m) \;=\; \overline{\phi}(m)(1 + p\log_2 N) \;=\; \overline{\phi}(m)(1 + (1/2)12)$$

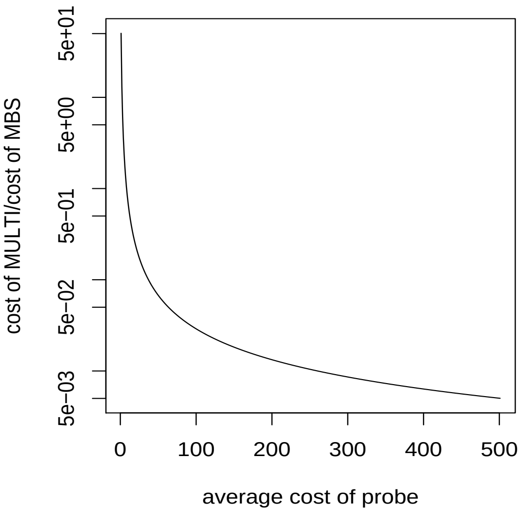
Figure 4.15 shows that as $\overline{\phi}(m)$ increases, the relative cost of MULTI to MBS decreases rapidly, indicating that the performance of our search scheme becomes significantly better with increasingly larger model sizes.



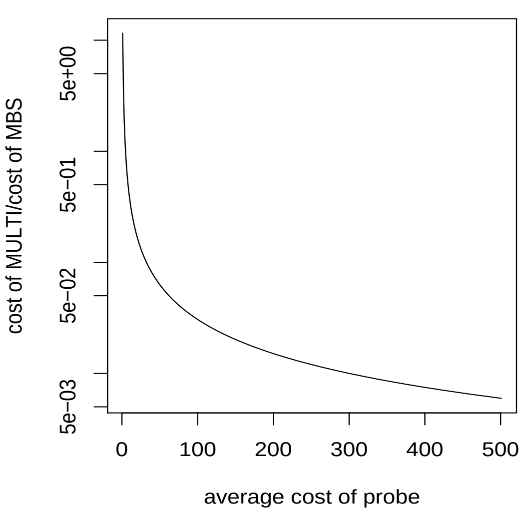
(a) 2-level MULTI



(b) 3-level MULTI



(c) 4-level MULTI



(d) 6-level MULTI

Figure 4.15: Performance of MULTI relatively to MBS as the cost of probing is scaled up

On a different note, an interesting factor that affects the overall performance of interval jumping is the ratio of time taken in a probe to the time taken in a time step. Measurements for these quantities are affected by the number of ongoing calls in the system, that is, the time taken for each individual time step or probe evaluation varies in direct proportion to the system load. For this reason, we measured the average time per probe and the average time per time-step over several simulation runs with varying horizons. We observed that the ratio $\overline{pt}/\overline{st}$, average probe time to average step time, stays in the range $[1.6, 1.7]$. This allows us to conclude that the m parameter in the interval jumping algorithm (see Figure 4.13) should be set to a value of 2 or greater. When the interval length is so small that the cost of determining whether it can be jumped is close to the cost of time-stepping through it, it's more reasonable to avoid the jump altogether.

The computational cost per jump encompasses more than just probing; the position of MSs has to be updated, a new power allocation computed, and the channel allocation verified. A justifiable estimate for m would be a value that takes into account the ratio $\overline{pt}/\overline{st}$ together with the average number of probes per interval search $\overline{\varpi}$ and the average probe cost $\overline{\phi}$. Although we chose $m = 10$ as a compromise solution, we suggest that its value satisfy the inequality $m > (\overline{pt}/\overline{st}) + (\overline{\phi}\,\overline{\varpi})$.

The end goal of this empirical analysis is the comparison of run-times of time-stepped and interval jumping simulations. As shown in Figure 4.16, for the problem size we used, interval jumping with MBS can accelerate the simulation by a factor of over 26 when the simulation horizon is approximately 14 hours. Although shorter simulations exhibit higher accelerations, these are more susceptible to the effect of ramp-up and ramp-down periods: as the wireless model starts and ends with no calls, the length of jumpable intervals is

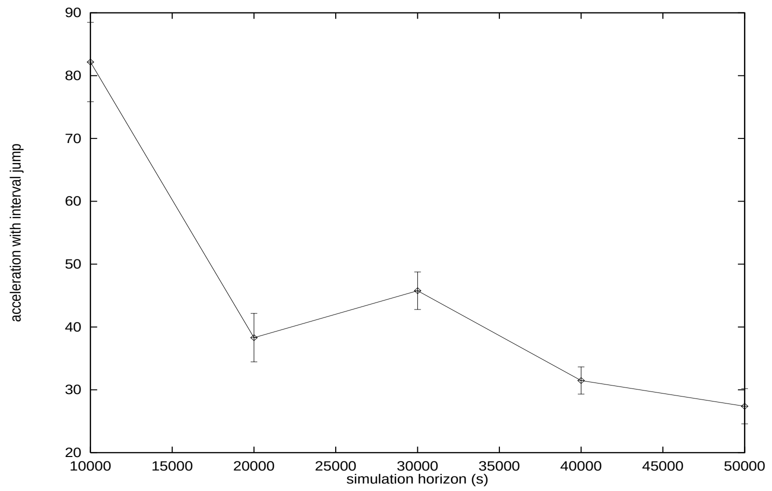


Figure 4.16: Acceleration of interval jumping with MBS search relative to time-stepping simulation

artificially increased at the beginning and end of the simulation. As simulations of longer length are run, the relative effect of these periods is amortized.

Using identical random number streams for runs of the same model using the two different simulation modes, we have observed that the power allocations at the endpoints of each jump are identical to values at corresponding instants with time-stepping. This corresponds to our expectations in that the state of power allocation after the jump should match exactly the corresponding state in the same instant of a time-stepping simulation.

4.8 Chapter Summary

In this chapter we have shown that our initial idea of interval jumping, although originally effective, left ample room for improvements. Intervals that would otherwise be summarily rejected for containing a critical event, can be trimmed down to exclude the event and thus made jumpable.

This observation led to the identification of the probe scheduling problem: "How can one efficiently apply the probe predicate to discover where the CE lies?" Upon formally describing the costs involved in this process, we have shown that an optimal solution to this problem can be devised with dynamic programming. We have also shown, however, that the computational costs of computing probing schedules that lead to the discovery of the CE are prohibitive for our purposes.

As our goal is to derive these probing schedules on-the-fly, we were compelled to seek alternatives that would produce comparably good schedules in very short time. We then introduced MULTI, a hierarchical search scheme that operates with grids of varying granularities and solves small, manageable DP problems only as it becomes necessary.

We presented an theoretical analysis of this new search scheme. Our experiments have evidenced that our scheme is not only fast enough that we can employ it within wireless system simulations, but also that its performance is quite close to that of optimal schedules. Moreover, we have confirmed its promise by discovering how fairly insensitive MULTI is to various models of probe cost and placement of CE in the interval.

Next, we described how we gathered data from a time-stepping simulation and derived models for probe cost and for the probability of finding a critical event in an interval. Using the same framework employed in our theoretical analysis, we studied the performance MULTI under the models determined empirically.

We compared the performance of real simulations using two different search methods: MBS, a modified version of binary search, and our hierarchical search MULTI. Our results have shown that, for the small problem sizes studied, MULTI does not perform as well as MBS. Through an approximated scalability analysis, we concluded that the poor performance of MULTI we observed is explained by the small costs of probing for our model sizes. We showed that as model size grows and interval length remains constant, the higher cost of probing becomes more severe and MULTI should be the best option: it becomes worthwhile to invest some time computing good probing schedules to minimize the number of probes applied.

Chapter 5

Directions for Future Work

In this chapter we indicate several interesting open problems that arise in the context of our research. The problems directly related to our contributions, that is, interval jumping and search policies, are described in detail at the beginning. Towards the end of the chapter, we present other open problems in the wider arena of simulation of wireless cellular systems.

5.1 Heuristics for Critical Event Search

In Section 4.7, we have indicated the attractiveness of using simple search policies to determine the possible location of a critical event (CE) in an interval. We have shown that binary search, or a slight variation of it, can perform well with interval jumping for small models. These policies do not require extensive information of the likelihood of where a CE may be, but still produce relatively small costs. The idea of reducing the search space by half at each step in the search allows for fast progress towards the solution.

Having taken inspiration on this same rationale to devise our hierarchical search scheme based on MULTI, we suggest the investigation of other simple policies that make better use of the stochastic clues to pinpoint the location of a CE.

Let S be a random variable that indicates where a CE happens. Assuming that S has a known but arbitrary probability distribution for every interval, statistics such as the median and mean of the earliest point x_S to fail the probe can always be computed. Now, suppose that these statistics are used to define the pivoting point in the search, just as the midpoint of the interval was used in binary search. *Median search* and *mean search* are simple heuristics based on this very idea; they both follow the same basic algorithm, changing only the nature of the pivoting point. Splitting the interval at the median attempts to reach a state where the probabilities of observing the CE in each half of the interval are balanced. On the other hand, pivoting at the mean tends to maximize the chances of probing at the right spot. Starting with the original interval $[a, b]$, one can compute x_k, the location of the pivot, and then probe this point. If the probe fails, the search continues on the left subinterval, otherwise, on the right. Each time the search interval is reduced, the probability distribution is truncated to reflect the gained knowledge of the possible location of the CE, and a new pivot computed. The search continues until the space is reduced down to an interval of length Δ.

These two policies have in common the fact that, at each subdivision of the search space, the computation required is minimal. Since samples of $f(\cdot)$ and $F(\cdot)$ of our random variable S are available, computing the statistics that drive these search policies is trivial. Finding the median requires only a linear search on $F(\cdot)$. Computing the mean requires a little more effort in terms of floating point operations, but it's still an easy task. These two policies may be substantially helpful to accelerate the execution of small models of wireless systems.

5.2 A Different Scenario for Parallel Jumps

Our first experiments with parallel interval jumping have considered a propagation model in which there exists adjacent channel interference. We have shown that under these circumstances the inherent parallelism in the model is very small. Since the models of most modern wireless systems deal with orthogonal channels, it would be interesting to make an effort to parallelize interval jumping to further accelerate these simulations. Each individual channel can execute in loose synchrony with others in the model, jumping or time-stepping according to its own *local* conditions, until an event such as an arrival or hand-off forces a global synchronization. Here is how this is possible.

5.2.1 Synchronization with Noncommittal Barrier

As we have pointed out in Section 3.2, when a base station processes an incoming call, a channel is chosen at random to serve it. In a parallel simulator, where each distinct processor has its own traffic generator, calls originated at a processor P_i may migrate to another processor P_j before even starting. The traffic generator in each processor associates with each call a vector containing a permutation of the system channels—each channel is tried in succession until one that can carry the call is found. The first element in these vectors is always a channel local to the processor that generates it. However, when the event associated with the call is first processed, it is possible that the local channel is not available at the desired cell and the call has to be migrated to another processor. This fact forces a global synchronization at the occurrence of events that deal with channel allocation, for example, arrivals and hand-offs.

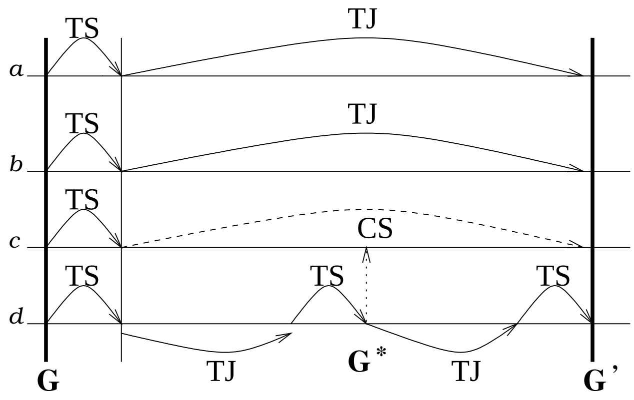


Figure 5.1: Synchronizing parallel jumps for orthogonal channels

When the operation of mapping calls to channels terminates at some time G, the next synchronization point $G^{'}$ can be calculated, and all channels simulate one time-step and enter a barrier. At point $G + \Delta$, if no channel switch was triggered in the simulation of the time-step, each processor is free to move forward in time, on its own, up to the next synchronization time $G^{'}$, where a *noncommittal barrier* is entered. This kind of primitive, attributed to Nicol [34], allows a processor to leave the synchronization point in case a message is received while it awaits the others. The noncommittal barrier allows us to define what should be a substantial improvement in parallelism for interval jumping.

Figure 5.1 illustrates, in detail, the idea we suggest. Let the horizontal lines in the figure correspond to the time scales of channels a, b, c and d, each allocated to a different processor. Assuming that the simulation starts from a global synchronization point G, all processors cooperate to compute the next time they synchronize, at point $G^{'}$. This computation is carried out in the same fashion that a jump interval is determined for serial simulation, that is, by inspecting lists of future events. The minimum from individual lists is combined by a min reduction that produces $G^{'}$. Next, each processor simulates one time-step for its channel and then enters a regular barrier. At $G + \Delta$, if no channel switch was triggered, each processor proceeds to search its interval for the next jump point. When a processor determines that it can make a jump all the way to $G^{'}$, it stops before actually processing the jump and enters the noncommittal barrier. Otherwise, if the next time advancement for the processor leads to an earlier time, it proceeds independently of the other. These processors may generate a channel switch, in which case the event is sent to the appropriate destination, possibly forcing the processor to leave the barrier. Whenever a channel switch is detected, its time G^* becomes the new global synchronization point and the appointment to meet $G^{'}$ is canceled. Either way, following a global synchronization, this process is repeated from scratch.

5.2.2 Extending Jump Intervals Further

Another idea related to interval jumping that deserves more investigation regards the construction of longer jumps. In a parallel simulation with channel partitioning, events are distributed across different processors, and perhaps even different address spaces. When a new jump interval is needed, one can use a parallel iterative scheme to resolve the mapping of events to channels and generate lookahead information.

Let all processors synchronize at a point G in time and then make each one collect a number Ω of its earliest events. If we have the processors work in parallel to map each event to the channel that will ultimately receive it, we may be able to construct a longer interval at a reduced time.

Suppose we have a total of NP processors at our disposal. Each processor, in parallel, checks if its earliest event can be admitted to one of the channels it simulates. After this quick operation, it synchronizes with the others and the same procedure is repeated for each one of the $NP\Omega$ events. At each iteration, NP events are mapped simultaneously. At the end of $NP\Omega$ iterations of this procedure, this phase terminates and the events are sent to the processors that simulates the corresponding channel. We will have figured out then if any call had to be immediately blocked or dropped due to channel allocation. Moreover, each processor would have a list of events that lie in its future. Barring the unexpected arrival of channel switches, each processor will have more information about what events will be sent from another processor into its own event list. The knowledge of whether and when a new event may arrive can be explored to construct longer intervals for jumping. This would be equivalent to doing a quick simulation within a larger framework. The costs and benefits of such a new scheme would be interesting points to explore.

5.3 Open Problems

It's been said that the simulation of wireless systems is a veritable gold mine of research problems. Here we enumerate a few that come to mind which could pose a number of interesting questions and lead to exciting new projects.

We have developed and studied techniques for the acceleration of FDMA wireless cellular models. Analyzing what is required for the simulation of these almost antiquated models, we have contributed knowledge that will be useful as a foundation for the acceleration of more modern, digital cellular systems. Networks built around GSM, TDMA and CDMA are substantially different from FDMA, and justify the development of their own particular simulation methods. Nevertheless, the aspects they have in common with the models we used should make the adaptation of our techniques a viable and exciting goal.

Power control is performed at different time granularities in GSM, CDMA and FDMA. Although it is reasonable to expect that interval jumping can be made to work under these different scenarios, the fact that there are major differences in the timing of activities in these systems opens the door to new challenges. GSM requires power updates much less frequently than FDMA and thus its simulation may not benefit from interval jumping. The opposite is true for CDMA, where updates happen one order of magnitude faster than in FDMA. The models for these systems, however, are much more intricate and require substantially different power control laws. How one can use adaptive time-steps as in interval jumping to accelerate these simulations should be a question worth pursuing.

The radio propagation models we have used were very simple. One might even say there were a bit coarse. For the purposes of producing first-order estimates of system statistics, however, they are more than adequate. Simulations with the most refined propagation models including fast and slow fading, multipath fading and detailed terrain descriptions pose a much greater computational load and, thus, are new research in and of themselves. Coupled with the added load of other model components, such as channel allocation, mobility and power control, the accurate simulation of a detailed model is an almost Herculean task, especially when the added feature of scalability is expected.

When one poses the problem of simulation of a system this complex, methods that allow one to look at qualitative aspects of the model at different levels of refinement become highly desirable. The application of hierarchical and multiresolution modeling to wireless cellular systems would be of great benefit to the communications engineering community. Currently, for each level of granularity, an entire simulator has to be developed. It all starts at the radio link level with the simulation of every bit sent riding on an electromagnetic wave, at a level of detail fine enough to allow the simulation of only one connection at a time. Then models evolve to single cell, small linear and planar arrays, entire metropolitan areas, and so forth, up to perhaps something so big as to encompass the entire planet. How something like this may be achieved with a single hierarchical model is an open problem.

Finally, it is still currently unknown how the results of simulations of large and/or complex models can be validated. Since models such as those for wireless systems can't be studied just with analytical tools, there is no easy way to provide a standard to assess the correctness of the simulations. Any meaningful step towards the development of guarantees of accuracy for the simulation results of wireless systems would be a godsend.

5.4 Chapter Summary

In this chapter we suggested several ideas related to the simulation of wireless cellular systems. The first of these is related to search policies used conjunction with interval jumping and aim at determining the location of a critical event. Next, we suggested two problems related to the parallelization of interval jumping for models with no adjacent channel interference. Finally, we concluded with an overview of more general open problems in the area.

Chapter 6

Conclusion

Our work has approached the problem of reducing the run-time of simulations of wireless cellular systems from two different points of view. First, we identified the fact that a comprehensive model for these systems comprises a component which is essentially a continuous process in discrete time. This sub-model, which embodies the power control algorithm, interacts closely with other system processes, and ordinarily requires long and computation-intensive time-stepping simulation.

To mitigate the major bottleneck in these simulations, we have developed *interval jumping*, a technique that allows the simulation to "fast-forward" through periods of time when no interesting event happens. We have shown that the nature of these models makes them especially susceptible to acceleration when the simulation proceeds in this fashion. The experimental results we have observed initially with the application of this technique were encouraging, albeit not reflective of its analytically predicted potential.

Going back to the core of this technique, we developed an analytical study that allowed us to increase the length of jumped intervals and, therefore, better exploit the benefits of our method. We have accomplished this goal by creating the means to inspect intervals and identify intermediate points known as *critical events*, where side-effects of power evolution could potentially trigger changes in channel allocation. Using dynamic programming, we created a hierarchical search scheme which is theoretically proven to outperform binary search. Our experiments, however, have shown that models with small number mobiles do not experience the benefits of our search scheme. We have observed that these reduced models run faster with a simple variation on binary search. A simple scalability analysis indicates, however, that as model size grows, the hierarchical search outperforms the binary search variant, whose costs increase linearly with the number of mobiles.

Having improved our technique with search heuristics that allow us to determine more intervals to jump, we built a faster simulator that has shown substantial performance increase compared to our first version. This confirmed the hypothesis made at the outset that the performance of interval jumping could be much improved by working out the inefficiencies of its implementation.

On a different angle, we have also worked to reduce the cost of interference computations, another major factor in determining the speed of the simulation of wireless cellular models. Directly applying a classic N-body problem algorithm due to Barnes and Hut [4] to the interference computation in our models, we have shown that it produces satisfactory results relatively to the brute-force method. When compared to the limited interference method, another approach to reduce the costs of interference computations, the Barnes-Hut algorithm performs slightly slower, but with higher relative errors. We took this as indication that perhaps higher order multipoles should be considered in the implementation, so that the errors can be reduced.

Although the techniques resulting from this research have been developed in the context of a model for FDMA, a technology that is currently being phased out, they apply equally well to *any* kind of system where frequent power control is a reality. CDMA, one of the leading edge systems of today, performs power control a frequency one order of magnitude higher than the FDMA models we considered. The main acceleration methods we presented, interval jumping, remains equally (if not more) relevant in this new context.

Bibliography

[1] Srinivas Aluru, G.M. Prabhu, and John Gustafson. Truly distribution-independent algorithms for the N-body problem. In *Proceedings of Supercomputing '94*, pages 420–428, 1994.

[2] Andrew M. Appel. An efficient program for many-body simulation. *SIAM Journal of Scientific and Statistical Computing*, 6(1):85–103, January 1985.

[3] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GloMoSim: A scalable network simulation environment. Technical Report 990027, UCLA, Computer Science Department, 1999.

[4] Josh Barnes and Piet Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(4):446–449, December 1986.

[5] Joshua E. Barnes. A modified tree code: Don't laugh; it runs. *Journal of Computational Physics*, 87:161–170, 1990.

[6] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA, 1995.

[7] C. D. Carothers, R. M. Fujimoto, and Yi-Bing Lin. A case study in simulating PCS networks using time warp. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS '95)*, pages 87–94, 1995.

[8] C. D. Carothers, R. M. Fujimoto, and Yi-Bing Lin. Simulating population dependent PCS network models using time warp. In *Proceedings of the 1995 Winter Simulation Conference*, pages 555–562, December 1995.

[9] C. D. Carothers, R. M. Fujimoto, Yi-Bing Lin, and P. England. Distributed simulation of large-scale PCS networks. In *2nd International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '94)*, pages 2–6, February 1994.

[10] K. G. Chen. Integrated dynamic radio resource management of wireless communication systems. Master's thesis, Rutgers, The State University of New Jersey, New Brunswick, NJ, May 1996.

[11] Leonard J. Cimini, Jr., Gerard J. Foschini, Chih-Lin I, and Zoran Miljanic. Call blocking performance of distributed algorithms for dynamic channel allocation in microcells. *IEEE Transactions on Communications*, 42(8):2600–2607, August 1994.

[12] Leonard J. Cimini, Jr., Gerard J. Foschini, and Larry A. Shepp. Single-channel user-capacity calculations for self-organizing cellular systems. *IEEE Transactions on Communications*, 42(12):3137–3143, December 1994.

[13] S. Das, R. M. Fujimoto, K. Panesar, D. Allison, and M. Hybinette. GTW: A time warp system for shared memory multiprocessors. In *Proceedings of the 1994 Winter Simulation Conference*, pages 1332–1339, December 1994.

[14] Eric V. Denardo. *Dynamic Programming.* Prentice–Hall, Englewood Cliffs, NJ, 1982.

[15] Gerard J. Foschini and Zoran Miljanic. A simple distributed autonomous power control algorithms and its convergence. *IEEE Transactions on Vehicular Technology*, 40(4):641–646, November 1993.

[16] Gerard J. Foschini and Zoran Miljanic. Channel cost of mobility. *IEEE Transactions on Vehicular Technology*, 42(4):414–424, November 1994.

[17] Gerard J. Foschini and Zoran Miljanic. Distributed autonomous wireless channel assignment algorithm with power control. *IEEE Transactions on Vehicular Technology*, 44(3):420–429, August 1995.

[18] Richard M. Fujimoto. Lookahead in parallel discrete event simulation. In *Proceedings of the 1988 International Conference on Parallel Processing*, August 1988.

[19] Richard M. Fujimoto. Parallel discrete event simulation. In *Proceedings of the 1989 Winter Simulation Conference*, 1989.

[20] Richard M. Fujimoto. Parallel discrete event simulation. *Communications of the ACM*, 33(10):30–53, October 1990.

[21] Richard M. Fujimoto. Parallel and distributed discrete event simulation algorithms application. In *Proceedings of the 1993 Winter Simulation Conference*, 1993.

[22] Albert Greenberg, Boris Lubachevsky, David Nicol, and Paul Wright. Efficient massively parallel simulation of dynamic channel assignment schemes for wireless cellular communications. In *Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS '94)*, 1994.

[23] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[24] S. V. Hanly. An algorithm for combined cell-site selection and power control to maximize cellular spread spectrum capacity. Internal AT&T Bell Laboratories Document, March 1994.

[25] Raymond G. Jacquot. *Modern Digital Control Systems.* Marcel Dekker, New York, 1995.

[26] D. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, July 1985.

[27] O. Kelly, J. Lai, N. Mandayam, A. T. Ogielski, J. Panchal, and R. Yates. Scalable parallel simulations of wireless networks with WiPPET: modeling of radio propagation, mobility and protocols. *IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'98)*, 1998.

[28] William C. Y. Lee. *Mobile Communications Engineering*. McGraw–Hill, New York, 1997.

[29] Michael Liljenstam and Rassul Ayani. A model for parallel simulation of mobile telecommunication systems. In *4th International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '96)*, pages 168–173, 1996.

[30] Michael Liljenstam and Rassul Ayani. Partitioning PCS for parallel simulation. *Proceedings Fifth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 38–43, 1997.

[31] Michael Liljenstam and Rassul Ayani. Interference radius in PCS radio resource management simulations. In *Proceedings of the 1998 Winter Simulation Conference*, pages 1629–1637, 1998.

[32] Yi-Bing Lin and Paul A. Fishwick. Asynchronous parallel discrete event simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 26(4):397–412, 1996.

[33] Yi-Bing Lin and Victor W. Mak. Eliminating the boundary effect of a large-scale personal communication service network simulation. *ACM Transactions on Modeling and Computer Simulation*, 4(2):165–190, April 1994.

[34] David M. Nicol. Noncommittal barrier synchronization. *Parallel Computing*, 21(4):529–549, 1995.

[35] David M. Nicol and Richard M. Fujimoto. Parallel simulation today. *Annals of Operations Research*, 53:249–285, 1994.

[36] David M. Nicol and Luiz Felipe Perrone. Cost/benefit analysis of interval jumping in wireless power-control simulation. In *Proceedings of the 2000 Winter Simulation Conference*, 2000.

[37] J. Panchal, O. Kelly, J. Lai, N. Mandayam, A. T. Ogielski, and R. Yates. Parallel simulations of wireless networks with TeD: radio progation, mobility and protocols. *Performance Evaluation Review*, 25(4):30–39, 1998.

[38] J. Panchal, O. Kelly, J. Lai, N. Mandayam, A. T. Ogielski, and R. Yates. WIPPET, a virtual testbed for parallel simulations of wireless networks. *Proceedings of PADS 98: Parallel and Distributed Simulation '98*, pages 162–9, 1998.

[39] J. Panchal, O. Kelly, J. Lai, N. Mandayam, A. T. Ogielski, and R. Yates. Wippet: A virtual testbed for parallel simulations of wireless networks. In *Proceedings*

of the 12th Workshop on Parallel and Distributed Simulation (PADS '98), pages 162–169, 1998.

[40] LUIZ FELIPE PERRONE AND DAVID M. NICOL. Rapid simulations of wireless networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS '98)*, pages 170–177, 1998.

[41] LUIZ FELIPE PERRONE AND DAVID M. NICOL. Using N-body algorithms for interference computation in wireless cellular simulations. In *4th International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2000)*, 2000.

[42] K. PERUMALLA, A. OGIELSKI, AND R. FUJIMOTO. A C++ instance of TeD. Technical Report TR-GIT-CC-96-33, College of Computing, Georgia Institute of Technology, 1996.

[43] THEODORE S. RAPPAPORT. *Wireless Communications Principles & Practice.* Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[44] H. D. SCHWETMAN. CSIM18 – The simulation engine. In *Proceedings of the 1996 Winter Simulation Conference*, December 1996.

[45] DANIEL DOMINIC SLEATOR AND ROBERT ENDRE TARJAN. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–686, July 1985.

[46] GORDON L. STÜBER. *Principles of Mobile Communication.* Kluwer Academic Publishers, Norwell, Massachussets, 1996.

[47] MICHAEL S. WARREN AND JOHN K. SALMON. Astrophysical N-body simulations using hierarchical tree data structures. In *Supercomputing '92*, pages 570–576, Los Alamitos, 1992. IEEE Comp. Soc.

[48] WILLIAM WEBB. *Understanding Cellular Radio.* Artech House, Inc., Norwood, Massachussets, 1998.

[49] X. ZENG, R. BAGRODIA, AND M. GERLA. GloMoSim: a library for parallel simulation of large-scale wireless networks. *Proceedings of PADS 98: Parallel and Distributed Simulation '98*, pages 154–61, 1998.

VITA

Luiz Felipe de Lima Perrone was born in Rio de Janeiro, RJ, Brazil, in 1965. From 1973 to 1983, he attended Colégio Santo Inácio, in his hometown, where he supposedly learned almost all one needs to know about life, the universe and everything. In 1984, he was admitted to the Federal University of Rio de Janeiro (UFRJ), where he worked towards a degree in Electrical Engineering, awarded in 1989. Upon realizing that engineers don't usually make good programmers, he joined the Pontifícia Universidade Católica (PUC-RJ) extension program, becoming a certified Computer Programmer in 1987. During the last three years of his college education, he interned with Centro de Pesquisas de Energia Elétrica (CEPEL), working with the division that researched automation of generation and distribution of electrical power. The year of 1989 marked the culmination of an academic life riddled by acronyms, when he joined the master's program in the Department of Systems Engineering and Computer Science at the Coordenação de Programas de Pós-Graduação em Engenharia (PESC-COPPE/UFRJ). His studies were funded both by a fellowship from CAPES, an agency of the Brazilian Ministry of Education, and by a fellowship from CEPEL. Immediately after receiving his M.Sc. degree in 1992, he joined the PhD program at the Department of Computer Science at The College of William & Mary in Virginia. In 1997, following in his advisor's footsteps, he relocated to New Hampshire, joining the Department of Computer Science at Dartmouth College as a visiting graduate student.