ABSTRACT

Title of dissertation: ARCOLOGY OPTIMIZATION AND SIMULATION FRAMEWORK

Rowin Andruscavage
Master of Science, 2007

Directed by: Associate Professor Mark Austin
Department of Civil/Environmental Engineering
and Institute for Systems Research

Arcology design combines urban planning and architecture with the mechanics of ecology, presenting a tangled mixture of functions, ideals, and goals well suited for systems engineering analysis. The physical design of an arcology would encompass the creation of a "hyperstructure" that delivers utility and transportation infrastructure in a highly integrated compact package that parcels out plots for residential, commercial, industrial, and municipal uses. This thesis defines and describes a prototype simulation framework that some day might be used to execute and evaluate intelligent demand-responsive multimodal mass transit schemes that would serve as an urban circulatory system, contributing to the effectiveness of an urban complex. Given a set of connected nodes serviced by different fleets of vehicles, a global optimizer attempts to generate a coordinated fleet schedule that meets various demand patterns. Factorial design of experiments and parametric analysis on the resulting simulated performance data of several simplified 1D and 2D scenarios help identify significant system

design variables, including the number and size of the vehicle fleet, station configuration, transit network topology, as well as the initial distribution of travel demand between station nodes. The open-ended formulation of this framework can allow analysis of different optimal modes of operation depending upon the properties of the scenario. This tool explores the effectiveness of transit-oriented design paradigms supporting arcologies and other urban forms.

ARCOLOGY OPTIMIZATION AND SIMULATION FRAMEWORK

by

Rowin Andruscavage

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2007

Advisory Committee:
Associate Professor Mark Austin, Chair/Advisor
Professor John Baras
Professor Hani Mahmassani

Table of Contents

List of Figures

List of Abbreviations

CPU	Central Processing Unit
GPS	Global Positioning System
graphML	Graph Markup Language
HCPPT	High Capacity Point-to-Point Transit
LP	Linear Program
MIP	Mixed Integer Program
MPS	Mathematical Programming System
OMG	Object Management Group
PRT	Personal Rapid Transit
RTA	Required Time of Arrival
RTD	Required Time of Departure
TPS	Traveling Salesman Problem
UML	Unified Modeling Language
WGS-84	World Geodetic Summit of 1984

Chapter 1

Introduction

1.1 Purpose

Arcology design combines urban planning and architecture with the mechanics of ecology, essentially forming a dense but ideally independently sustainable human habitation system. The complexity posed by these systems is well-suited for systems engineering analysis.

From a historical perspective, arcologies are mostly the subject of science fiction. In a typical scenario, the principles of arcology development are applied to the design of self-contained habitats, where the major challenge lies in finding ways for large groups of people to tolerate living in close proximity to one another. In addition to maximizing quality of life for its residents, the arcology needs to limit consumption of land, energy, time, and human resources. Achieving an appropriate balance in these (often competing) criteria requires that designers look beyond maximization of personal productivity and/or economic performance, and explicitly consider relationships among all factors affecting system functionality, performance, and resource consumption. The common characteristics of "good design solutions" include:

(1) Retrofit of present-day urban sprawls with large three-dimensional integrated urban

forms (*i.e.*, municipal hyperstructures),

(2) Extensive use of multi-functional spaces, and

(3) Use of sophisticated transportation networks and systems to transfer of people and resources (*e.g.*, power and water) between spaces.

Present-day trends in population growth and urbanization suggest that as time marches forward, spearheading ideas associated with arcology development will only become more important. To put this observation on a quantitative footing, we first note that as of July 2006, the World's population is 6.5 billion (and growing annually at 1.14%). Since 1900 there has been a significant movement of the World's population to urban areas, growing from 14% in 1900 to approximately 50% in 2000. The United Nations reports that by 2030, not only will 60% of the worlds population be urban, but nearly all of the anticipated growth will be urban growth.[?]

Rural-to-urban area migration is driven by a number of factors including: (1) the declining importance of agriculturally-based economies, (2) improved opportunity for access to services, and (3) improved opportunities that urban areas provide for specialization (when people congregate in urban areas they can specialize to a much greater extent than in rural areas[46]). Rural-to-urban area migration is enabled by access to housing and the ability of communication and transportation networks to form in response to supply and demand mechanisms. The latter is especially important because in order for the operation of high-population urban areas to be sustainable, efficient modes of transportation are needed to transport goods and people throughout the arcology.

From the earliest days of transportation infrastructure development, new transportation networks – initially railway, then road – were ultimately built to increase the value of the land surrounding the network.[?] Road and rail access stimulates urban growth, which in turn drives the need for more network development. Clearly, this cycle cannot continue forever. It is reasonable to expect that since land is a limited resource, its value will only increase at faster and faster rates as the worlds population increases. Since fewer and fewer people will have the economic means to migrate to sprawling (low-density) urban areas, high-density urban area will become more common. Thus, it may only be a matter of decades until modeling techniques tailored toward the needs of arcologies and/or other forms of well-integrated compact cities become common place. Early indicators of this outcome can be found in experimental prototypes for sustainable living (e.g., BioSphere 2^3) and long-term plans for dense sustainable communities and major city structures in East Asia.[12, 23, 24, 44, 45]

Project Objectives

This thesis defines and describes a simulation framework for the execution of optimized demand-responsive multimodal mass transit schemes, such as those found in complex urban environments. Attention is focused on resident and employer needs (i.e., shuttling passengers from their source stations to their destination stations.) By measuring the performance of various solutions, we aim to determine effective strategies for efficiently transferring people to their destinations in relation to input parameters such as demand, transit network topology, and the relative size(s) of the vehicles used in the fleet. This framework also al-

lows us to experiment with different urban planning layouts and to investigate relationships between services provided by data networks and transportation networks. For example, if a city decides to spend money upgrading their data infrastructure so more people might be able to telecommute to work, then this may have a critical impact on the load on their mass transit system.

While the simulation and optimization models are certainly generic enough to apply to most ordinary forms of mass transit, for two reasons this project chooses to frame models in the in the context of an arcology. First, the word "arcology" still remains rather unique in the global namespace of the engineering field, and connotes a flair for futurism. More importantly, the design focus of arcologies as an autonomous structure encourages us to analyze it in terms of control volumes, defining the flows of input and output products in ways much more conducive to identifying resource consumption and environmental impact. While the concept of analysis via the definition of control volumes may come naturally to engineers trained in thermodynamics, it is refreshing to see efforts emerging to track our "carbon footprint" as part of a global carbon dioxide emissions budget. Hopefully this step will preclude more complete tracking and accounting (and eventually optimization) of human environmental resource use and waste reclamation.

1.2 Inspiration for Arcology Modeling

It all goes back to the meaning of life, doesn't it? Humans spend inordinate amounts of time looking for love or money or happiness, always trying to get the most out of life - in

Category	Current Interfaces	Potential Future Interfaces
Physical	Driveway, Parking, Mailbox	Driveway, Automated Package Transport
Utilities	Electricity, Gas, Water, Sewage	Electricity, HVAC, Fuel (Gas, Hydrogen), Water, Sewage
Wired Communication	Copper / Fiber medium for Telephone, Cable TV, Internet	Junction Box, redundant trunks
Wireless Communication	Broadcast Radio/TV, Cellular phone / data networks, Satellite, WiFi access points	distributed cellular mobile ad-hoc networks with repeaters for inside reception

Table 1.1: Municipal home interfaces

essence optimizing our existence in some fashion. The optimization part is where simulation can be a useful tool, as we often disagree on what infrastructure improvements we could make in order to make us happier or richer or work not so far from our loved ones. For all the aspirations we've had over the decades of reaching for the stars and developing permanent space colonies, I'm surprised by the relatively little success we've had in improving the efficiency of our lifestyles in our dwellings right here on Earth. As summarized in Table 1.1, the ideal American domicile still typically consists of the single family home, an almost completely isolated pocket of land connected to the rest of the community only by a few wires, pipes, and a stretch of pavement. What goes across these interfaces? And how might they be improved and rearranged by municipal facilities to make the city as a whole more sustainable, flexible, and efficient?

As a first step toward understanding these complexities, let us note that living systems seem to have a natural tendency to miniaturize complexity, both in space and time. A mathematician might draw the analogy that we live on the interesting boundary region of a fractal, often surrounded by vast regions of fairly uniform space. While the sun and

stars and most of the universe are beautiful and magnificent only when observed on a scale spanning thousands to trillions of kilometers, I'd surmise that they are not as interesting when studied on a micrometer scale used to observe, say, the inner workings of a paramecium. That's one of the main reasons that, as astronomers, we might search the heavens for deeper understanding of celestial mechanics, but hope to discover other forms of complex alien life. For we could only have hope to interact with other living systems of sufficient complexity that exist on a similar space and time scale as we do. Meanwhile, the progression of life on Earth as a whole apparently strives to fit more and more complexity into the spaces it is able to fill.

If we drew a control volume around an ecosystem, we'd find that it functions as an engine that harnesses existing energy gradients in order to further decrease the entropy of its local area.[25] Through continuing that progress, we've begun to expand the boundaries between which objects of vastly different scales can interact. Lately we've been peering into the inner workings of relatively tiny, fast computing devices, which will soon be governed increasingly by subatomic interactions between quantum particles, which in turn affect what we do with our lives and our global economy. That's amazing. Someday soon, we also expect that the tiny electrical processes that occur in our microchips may go on to help us alter the courses of celestial bodies, perhaps to allow us to produce some kind of pronounced impact (or avoid an impact) in the cosmic ballet of planets. But for now, one of our primary (although not yet fully utilized) uses for our microprocessing technology often is the guidance of the course of our vehicles and information delivery systems.

Influential Literature

The specific concept of the arcology was first introduced in the 1950s by architect Paolo Soleri as the ultimate urban planning solution to the inherent problems of metropolitan growth.[42] Continuing trends in the expansion of metropolitan areas have contributed to explosive growth of low density suburban sprawl, the decay of inner city urban areas, and finally the indiscriminate destruction of natural environments to make room for a human habitat system which is increasingly less efficient, less convenient, less and aesthetically pleasing. The concept of the arcology attempts to reverse those trends by providing a compact city infrastructure that works well and manages to reprocess most of its waste before returning material back to the environment for further reclamation.

What exactly *is* an arcology by definition? Featured in several science fiction works as futuristic cities, an arcology is more than simply just a colossal structure or superbuilding. The arcology integrates living spaces and working spaces with transportation systems that connect it all together. One of the fundamental differences between arcologies and conventional cities is the emphasis on the effective use of the vertical dimension in city planning. An arcology design would strive to make use of several independent but functionally intertwined layers or horizontal planes, whereas current urban planning focuses more on flat zoning of commercial / residential / industrial areas through processes that result in a more ad hoc placement based on the situational needs and political landscape at the time. Another distinguishing characteristic is the arcology's roots in urban agriculture, meaning deliberate collection and reprocessing of waste byproducts. The arcology might simply be described

as what a city would look like if it was designed from the start by competent systems engineers, incorporating modular growth packages, standardized by upgradable interconnects, and fault tolerant, serviceable components. Of course, this feat is easier said than done, but the basic architectural core building blocks and modular techniques have been around. Meanwhile, the fact that the construction of many large city buildings has gradually been consolidated into one or two large developers means that a single party can finally expect to see gains through the extra effort of standardization.

In 1978 George Dantzig and Thomas Saaty (fathers of Linear Programming and the Analytic Hierarchy Process, respectively) got together to write *Compact City*,[17] providing a compelling vision on how this human habitat could work from a technical standpoint. This fascinating book contemplates the feasibility of constructing a livable city of between $\frac{1}{4}$ million to 2 million residents within a 2-4 square mile, 4-8 level cylindrical superstructure. Their proposal addresses many social and financial factors as well as provides major engineering design elements and outlines the major systems and physical characteristics of their ideal proposed layout.

The Modern Metropolis consists of a series of Hans Blumenfeld's essays and articles on urban growth versus urban planning.[10] These treatises generalize how cities have developed and evolved over the decades and centuries, and suggests some design principles for sustaining growth over time. These insights into how to cope with the forces that incrementally shape cities and inevitably stress them beyond their initially planned limits reinforce some of the ideas for flexibility provided by Dantzig and Saaty's design.

Arcologies in the Media

A good experiment in closed-system sustainability is Biosphere 2.[3] Unfortunately, its primary experiment was widely regarded by the public as a failure.[11,40] The facility has since come under the management of Columbia University as a research lab.

The closest present-day developments resembling arcologies are scattered around the world in various stages of completion. The truest in spirit of arcology projects in existence would include Arcosanti and Cosanti, the experimental communities arranged by architect and founding father of the "Arcology" concept Paolo Soleri himself.[2] These reduced scale experiments in the Arizona desert are currently reported to be hovering around 5% complete after 30 years of development. Like the Biosphere 2, these developments have shifted their focus into acting as urban laboratories.[30]

While this apparent lack of enthusiasm and resounding success paints a somewhat bleak outlook, the influence of these spearheading projects is definitely spreading. Large scale proposals have been cropping up more frequently, especially in population-dense Asia. Predictably, the Chinese have a keen interest in the arcology concept, both for expanding high-density urban areas,[24] and also in the form of constructing sustainable communities that would address their growing problem with semi-rural slums.[45] Several Chinese and Japanese design firms have been promoting various skyscraper approaches, such as the Ultima Tower,[44] Tokyo's Sky City,[12] and the on-hold Tokyo Millennium Tower[23] (the latter two are covered in Discovery Channel documentaries[18,19]). Arcology.com has a collection of other notable works and proposals.[1]

While excitement about radically redesigning urban forms hasn't quite taken off in practice, other environmentally-friendly initiatives have taken its place. Several publications focus more on modifying the design goals of current city planners to incorporate more alternative forms of transportation. The book and accompanying website *Carfree Cities* presents several concepts and examples that make urban areas more pedestrian, biker, and transit friendly.[15,16] Most contemporary urban revitalization works take this track of advocating increased use of multimodal transportation in current city design to cope with the strains of present-day metropolitan area growth. Many formerly suburban towns have already been pursuing more pragmatic policies encouraging higher-density mixed-use development. These philosophies go under the monikers of "New Urbanism", "Smart Growth", and "Transit Oriented Design/Development". For example, following successes in implementing this pattern in the Washington DC metropolitan areas of in Rosslyn and Silver Spring,[20] plans are underway to build higher density mixed-use population centers off of existing transit stations in Vienna[36,38] and to extend transit to existing office and residential spaces in Tysons Corner.[22,34] We'll likely see more of this type of development in the near future, especially seeing as how the Supreme Court has recently ruled to allow private homes to be seized for mixed use and other commercial development.[5] However, in his article "The Compact City Fallacy", Neuman defines how and cautions that higher density and other Smart Growth policies alone will not guarantee that we will meet the goals of sustainable development or even achieve progress relative to previous development patterns.[32]

1.3 Modeling and Design of Urban Environments

What makes a city special compared to a cluster of businesses and residences? Hans Blumenfeld[10] argues that a metropolitan area attracts corporations and residents with highly specialized skill sets. Also, as the population grows, a wider variety of niche businesses can sprout up and sustain themselves while catering to a relatively small segment of the market. So by this consideration, a good metropolitan area draws businesses and populations to it by maximizing the diversity and variety of specialized skills and jobs. See Figure 1.1 for a visual summary of these trends. A larger, more developed metropolitan area (represented by the green shaded area extending out from the smaller blue shaded area) would have more positions requiring advanced degrees, as well as offer more variety in terms of ethnic restaurants, specialized services, *etc.*

Geographically, as cities grow in population, they often grow "outwards" in area before they growing "upwards" in density. As noted in Figure 1.2, this typically follows a pattern of "fingers of development" that grow outwards from the urban core along established transportation corridors such as highways or waterways. The result is that most metropolitan areas eventually become victims of their own success. Drawing a more diverse and skilled population eventually increases their geographical size towards a point where a resident of the city can no longer access all of the resources the urban area has to offer due to congestion.

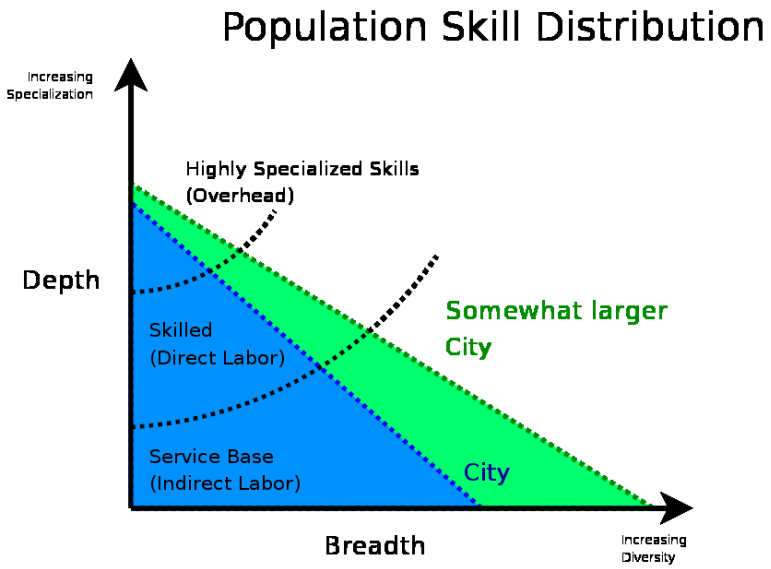


Figure 1.1: Population Skill Distribution

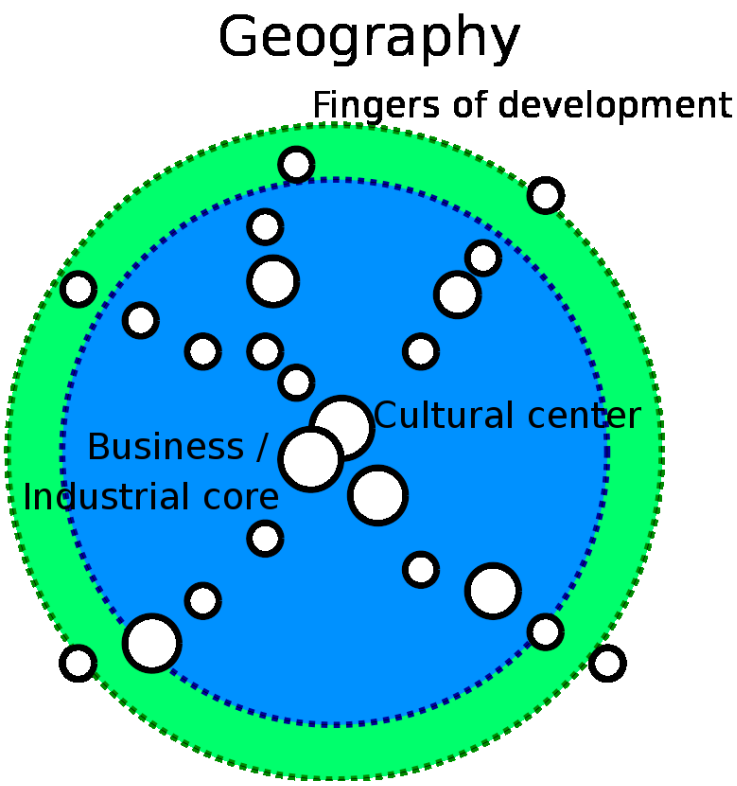


Figure 1.2: Geographical Distribution

The Role of Transportation Networks in Urban Environments

Most of our interactions with the urban environment that we live in, such as going to work, catching a bite to eat, buying groceries, or ironically even going out for a hike, involve transportation and delivery networks. These systems take many forms, ranging from various ground, air, and subterranean transit networks to power, water, and even information distribution pipelines that feed directly into each of our homes. Much of this infrastructure is put in place with funding or regulation from government agencies at national, state, and local levels. During times of rapid modernization, traditional governments can be a bit slow in figuring out what infrastructure to invest in.

Urban Design from a Systems Perspective

An advantage to designing cities from the complete-systems perspective of an arcology is that it forces you to take all scale levels – national, metropolitan, urban, neighborhood, personal – into account in the design. This would allow the arcology to transition better as new technologies evolve and are put into place. The physical aspect of an arcology is predicated on a municipal "hyperstructure" which could be sectioned off for residential, commercial, industrial, and civic use. The sectional lots would have tightly integrated people and package transportation in addition to the standard complement of water, utilities, and a more minimal road network.

On the **national level**, arcologies would be constructed to connect well to other cities, with effective transportation and distribution systems and quick transit times to most

major destinations. Current cities tend to have suboptimal transportation facilities. Many cities originally sprouted up around ports by major waterways, where maritime shipping accounts for over 90% of the tonnage of U.S. international imports and exports.[26] However, domestically we move freight predominantly by truck.[4] The United States has invested heavily in the interstate highway system. Around many cities these get tied up in rush hour congestion, resulting in delays and waste throughout. High speed rail is an option that works well in most of the rest of the industrialized world, but has languished in America. Airports are usually built too far from the city to connect easily to mass transit systems, and eventually get enveloped (and subsequently throttled) by suburban growth after which they become a noise nuisance to residents.

On the **metropolitan level**, rush hour congestion itself is an abomination that any commuter would readily identify with. We must look terribly silly to outsiders, repeatedly stressing our transit infrastructure past the capacity limit where it ceases to be effective. We tend to want to commute simultaneously simply to be in sync with everyone else - even those whom we don't even need to deal with during the workday. Dantzig and Saaty have dubbed this phenomenon as "cicadian rhythms" and have noted that it also applies to water and energy utility usage. They have outlined the very simple remedy of staggering a population's daily schedules, which is attainable once a community reaches a sufficient size to support commercial staffing arrangements into several shifts.

The U.S. metropolitan growth paradigm of roughly the last half-century has been characterized by suburbanization. Affordable housing seems to be in such short supply and

fuel prices had been so low that many chose to commute into job centers from suburban or exurban towns 30, 60, 90 miles away. Financial policies strongly encourage citizens to purchase homes and enter into mortgage agreements. This provides economic stability in the workforce, helping to affix them down in a geographic area and ensure they stay gainfully employed to keep up with mortgage payments. However, in today's increasingly unstable job market, this policy can have adverse effects as a workforce with impaired mobility will not have as much flexibility to take on employment that maximizes use of their skills.

So as more massive superhighways are built to relieve the strain on the original interstate connectors, more suburbanites continue to sprawl out along these new corridors. After a certain point, the ratio of space allocated between highways and developable, livable area becomes saturated to the point where we get diminishing returns from building more roadways. High capacity highways take up a lot of space relative to streets, and when we start to pack those highways close together, we end up spreading out actual useful land into isolated pockets nestled between interchanges. Looking down on our cities from above, we'd find that most have more land area allocated to paved roadways for cars to drive across than space for humans to go about their affairs.[32]

To their credit, automobiles are certainly the most flexible mode of transportation. All you need is a slab of pavement or even gravel connected to the nearest street, and you now have an interface to the "intercontinental road transportation network". Compared to the equipment you'd need to interface with the municipal power grid or water/sewer lines, this slip of asphalt is likely one of the simplest yet most capable ways of moving people

and products to and from your home. However, when be build cities almost exclusively around automotive transport, we end up losing a lot of what makes dense cities good for people and sustainable for the environment. Cars act as a multiplier to the amount of space each person takes up. Not only do you need a driveway space to park each person's car at their home, but also a space reserved at their work, as well as some shared spaces at all of the shops and venues at which they'd possibly spend time. Add to this the ganglia of roads connecting those parking spaces together, shoulders for emergencies, extra lanes for additional peak capacity, and of course spacious service stations, and we find that our cities have vastly outgrown the human scale. A transit-oriented city would provide more land use for people by introducing transit alternatives that allow them to travel between home and work from door-to-door. Park-and-Ride initiatives connecting to mass transit accomplishes little in regards to improving land utilization, since they do not eliminate parking spaces, only relocate them further away from the workplace. In an urban complex with sufficient transit, people should only need to use (or borrow) cars to leave the city, and rely on transit to move people and goods within the city.

As the urban area grows, we attempt to preserve an ideal population density while also preserving the practical reach of the transit system to prevent fragmenting the city. For civic planning authorities, this traditionally involves zoning and building out roads and utilities. At some point along the city's growth, they might consider the efficiencies of building infrastructure based on a futuristic arcology hyperstructure in order to meet their urban development goals in a compact physical package.

On the **personal level** most home infrastructure for living does not have much flexibility for change. We are still using much of the same basic physical interfaces developed over a century ago for power and voice communications. Additional systems have sprouted on top of and alongside these networks, such as DSL over existing telephone wiring, cable television, and various wireless and satellite networks. Add to that various combinations of buried water mains, sewage systems, natural gas pipelines, and perhaps we might begin to appreciate the need for developing more flexible and maintainable utility distribution and interconnect standards. The new standard interconnects would provide room for expansion and serviceability, supporting the adoption of emerging new infrastructure networks and allow easier retrofit of older homes and living spaces. Such standards help reduce the barriers to market entry, allowing economical deployments of upgrades such as fiber-to-the-premises or even some things for which markets haven't really been created for yet, such as the fully-automated package delivery systems or centralized HVAC services referred to by Dantzig and Saaty.[17]

1.4 Simulation and Optimization of Urban Environments

Simulation of Operational Concepts

Simulation is one tool that can help quantify the benefits of different operational concepts, which in turn can help answer questions about design options. A common engineering practice is to first document and construct a baseline validated simulation of the system you have in place, then extend the simulation with new proposals for changes to equipment

or operation. After analysts have evaluated the performance benefits projected by different options, engineers could make a decision, implement the change, and then re-validate the simulation to make sure their model matches the performance of the modified actual system. Unfortunately, few municipalities maintain validated simulated representations of their jurisdictions, much less use them as decision making tools, deferring more towards the use of surveys and standalone analytical teams. Building such a tool would not only give them better access to information about the physical arrangement and performance of their existing town, but could also be used as a "vision communication tool" to the populace in order to cut down on some of the arguments and political delays.

Simulation as a Decision Support Tool

Several initiatives are currently underway to rethink the way metropolitan areas are designed. This simulation modeling and analysis framework can provide a design planning and evaluation tool to assess several integrated mass transit paradigms such as busing, rail, and PRT-type (personalized rapid transport) networks to help identify and accelerate acceptance of the worthwhile investments.

The use of simulation as a decision-support tool could provide a measure of accountability that would help avoid or at least temper some of the larger controversies over the past century of rapid technological change. The history of our infrastructure has been peppered with some epic and ultimately costly battles over different modes of transfer, such as the turn of the century Edison - Tesla battle to establish AC or DC as the power de-

livery standard[27] or the politicized finger pointing over whether GM was duly responsible for taking control of streetcar operations in the 20s in order to dismantle them in favor of GM-manufactured buses.[14,41] Having detailed records of the simulations used to provide hard data on which broad policy decisions are based could help justify your decision later. With more exotic options pushed by several technology firms, we ought to determine the selection of major communications upgrades or transit systems based on available technical data, and not on which company has the best connections to the civil servants responsible for municipal decision making.

Ultimately, if this were to evolve into a fully-featured urban simulation tool, it could be used as a rapid prototyping environment for proposals to system changes big and small. When this functionality matures, a municipality might require a simulation-based analysis to accompany any new infrastructure proposal as part of a gateway approval process. As standard patterns are built up, the simulation framework may morph into a design tool, replete with a library of openly available blueprints, guidelines, and standards (as well as freely customizable sections) to that can be assembled to achieve development goals. Furthermore, as the process becomes automated, it might incorporate more direct civil input, turning review and evaluation of problem areas and proposals into something of an experiment with direct digital democracy governance, in which the citizens can interact as something like a hive mind. Or so goes the vision.

Urban Simulation in the Media

Previous well-known works that tackle the task of urban simulation includes two series of open-ended games from Maxis (now part of Electronic Arts) that approach the problem from different scales: SimCity and The Sims. Certain versions of SimCity (2000 and 3000) even had actual arcology units in them (although since they were entirely self-contained, they really added little to the game play other than to provide an easy way to boost your population tax base). To some extent, these games could be used to experiment with different urban or residence layouts, but they primarily pattern themselves after common current day paradigms and lack the flexibility needed to really turn its simulated environment into useful data. Hopefully these games will serve to influence the next generations of urban planners and administrators, who might come to expect and demand some of the streamlined user interfaces to command, control, and instantaneous reporting of city condition and resources. Beyond that, there is not much published in the way of complete city and/or lifestyle simulation. This might be the case partly because most analysis can simply be done on spreadsheets using historical data tracked by government agencies, and partly because most simulation programmers are still busy developing their craft while simulating more interesting things such as data[31] and transportation networks.[33]

Optimization of Mass Transit Operations

The purpose of the optimization tool embedded within the simulation is to provide some measure of intelligence that could demonstrate an advanced, demand-responsive mod-

eling scheme. We'd use this flexibility to investigate the potential effectiveness of various mass transit paradigms, especially with regards to network topologies and their ability to model:

1. The distribution of various loads generated by work nodes and residential nodes.
2. The size and connectivity constraints of various shared vehicle networks shuttling people and goods between nodes.
3. The ability for the passengers and cargo to make transfers between different vehicles as well as modes of transit.

Applying a schedule optimizer ensures that we evaluate different transit paradigms on a level playing field. Different transit schemes utilizing rail, bus, and PRT styles of vehicle sizes and routing will get a fair shake at providing the maximum theoretical performance possible given the same physical construction constraints. Each mode of transit will have some measure of routing intelligence that should reflect the optimization computing power that should become more pervasive in the near future. They would all operate with the benefit of an intelligent central dispatch that characteristic of advanced transportation systems. System operators will have the freedom to direct their fleet about the network and pick up, transfer, and drop off groups of passengers as necessary to meet passenger demand as quickly and efficiently as possible with their existing resources. Mass transit vehicle fleets will only be subject to the physical constraints of vehicle passenger capacity, station berth / terminal / gate capacity, and the existence of connective links between stations.

1.5 Scope and Contributions of this Work

This research and development project serves to realize an urban multi-modal transit simulation designed during the course of the systems engineering master's program. The work takes a systems approach to modeling human habitats and the transportation networks that keep them running. The hope is that such a simulation framework will create a baseline model of current day capacity, against which future models may be evaluated with respect to performance and investment decisions. The hypothesis of this work is that *these tools will be instrumental in making a case for the development and construction of highly efficient arcologies or other forms of well-integrated compact cities.* But nominally, and for the meantime, urban multi-modal transportation frameworks can be applied to the evaluation and tracking of present-day transit oriented growth philosophies.

Chapter 2 describes the formulation of a generic arcology system model – the result is a series of conceptual templates represented as classes and relationships among classes. Chapter 3 covers many of the practical details one needs to consider in creating discrete event simulation environments. Chapter 4 is all about the specific commuting transit system model analyzed in this work. Chapter 5 contains simulation scenarios, factorial design of experiments, and parametric analyses for various mass-transit topologies. The project conclusions and opportunities for future work are covered in Chapter 6.

Contributions. The contributions of this work are as follows:

1. A hierarchical level-of-detail organization that allows data from both top-down paramet-

ric models to interact with data generated from clusters of detailed simulation objects. This allows us to seed detailed objects in a subsystem using available aggregate data from the live system, and compare the live results to data generated by tallying up the individual contributions from individual simulation objects. The hierarchical organization also makes the simulation easier to partition across distributed compute nodes.

2. Definition of a data interchange schema between elements of a multi-modal transit infrastructure. The communication provides just enough information about each piece of passenger, cargo, vehicle , and connectivity graphs and defines minimal interfaces to allow them to report to and receive suggestions from a global transit optimization engine.

3. An inherent focus on meeting the needs and goals of the inhabitants. Many transportation simulations focus on maximizing throughput or minimizing delays or fuel expenditure. However, these metrics may not serve to help evaluate the layout of the urban area itself. This simulation infrastructure would ideally be used to measure the effectiveness of optimizing the layout of an urban area to reduce the need to load the transit infrastructure with commuters, people running petty errands, and other frequent but necessary tasks.

An ideal city would have a higher "efficiency" ratio, tracked by an admittedly somewhat elusive "productivity" metric divided by the amount of energy directly needed to produce it and energy overhead required to nominally sustain production.

$$\eta = \frac{GDP}{E_{direct} + E_{sustinence}}$$

A simplified multimodal mass transit optimization solver coupled to the simulation attempts to create a demand-responsive fleet schedule for several types of defined vehicle types that service transit networks within the simulation. This tool aims to provide a quasi-optimal means to transport people and goods around within city clusters to help reduce the overhead of the transit system.

Chapter 2

Generic Arcology System Model

One of the key characteristics of systems that grow by evolution rather than by design (such as cities), is a distinct lack fundamental policies and "systems wide" thinking to drive their design. Too often city components and services are created in a reactionary manner – for example, fire protection services are provided after too many buildings have burnt down, airports are built to serve cities after they have already grown too dense to accommodate one in a central location, tap water distribution systems are gutted out and replaced only after the old ones were too heavily loaded to be sanitary. A second problem is that too often new systems are developed without giving proper regard to their impact on the inhabitants and current infrastructure (e.g., congestion, environment). The central benefit of "systems wide" thinking for city/urban development is that it leads to methodologies that aim to identify and deal with cause-and-effect relationships among all system characteristics, early in the development life cycle where potential problems are easiest to mitigate.

With this backdrop in mind, this chapter describes a methodology for the specification of urban areas, highlighted by transportation network overlays that serve as a circulatory system. The result is a "generic system model" that separates development concerns. (e.g., system behavior is separated from system structure; system performance is assessed through measures of effectiveness).

2.1 Model of Systems Engineering Development

A good system design provides: (1) A desirable balance of functionality, performance, and economy, (2) A pathway to convenient and reliable operation in a wide range of environments, and (3) Ease of accommodation for future expansion and technical improvements. Assessment procedures need to consider not only metrics on the system functionality, performance, and economy, but the potential impact develops will have on the existing infrastructure and environment.

To maximize the likelihood of development efforts staying on track, most complex engineering systems are developed within the framework of an agreed upon (or established) process. Figures 2.1 through 2.3 illustrate the step-by-step procedure for front-end systems development that will guide this project. Points to note are as follows:[6]

1. At the front-end of development, systems engineers are concerned primarily with system functionality and identification of the key environmental conditions within which this functionality must occur. Therefore, models of system functionality need to describe what the system will do under both normal and abnormal operating conditions. Answers to these basic concerns are commonly expressed as functional requirements. Performance requirements describe how well a system should perform these functions. And economic requirements place constraints on the resources that will be made available to develop and operate the system.

2. Top-down development of system-level models begins with use cases, and proceeds to

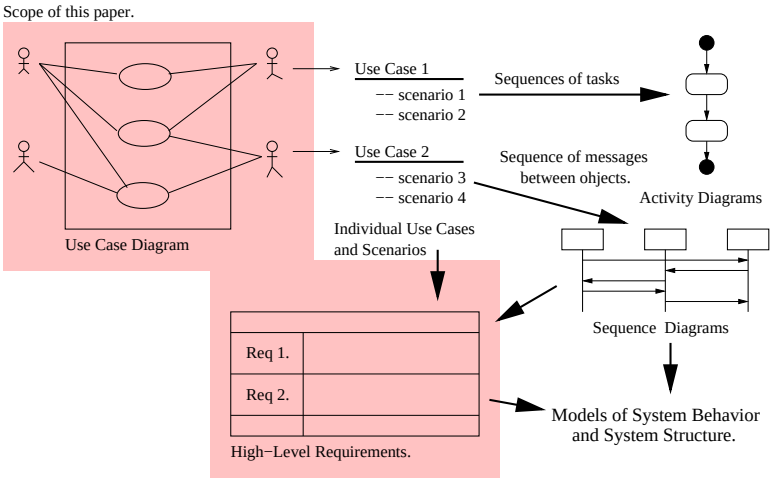


Figure 2.1: Pathway from Operations Concept to Models of Behavior/Structure to Requirements

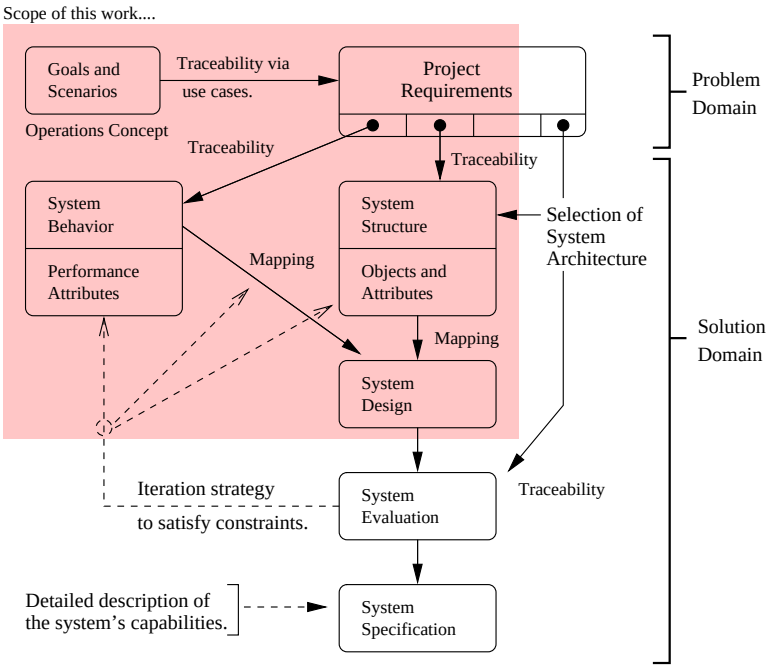


Figure 2.2: Traceability Mappings for the Development Pathway, Goals/Scenarios through System Evaluation (Source: Austin/Baras[6])

fragments of system behavior, expressed as activity and sequence diagrams. Requirements are organized according to the role they will play in the system-level design. For example, some requirements will be directed towards the system behavior because they place constraints on minimum/maximum levels of acceptable functional performance.

3. Models of behavior specify *what the system will actually do.* Usually, behavior can be represented as networks and hierarchies of tasks, functions and processes. Behavior is evaluated via attributes of performance. Models of structure specify *how* the system will accomplish its purpose. The system structure corresponds to collections of interconnected objects and subsystems, constrained by the environment within which the system must exist. The nature of each object/subsystem will be captured by its attributes.

4. We create the system-level design by mapping fragments of system behavior onto specific subsystems/objects in the system structure. Thus, the behavior-to-structure mapping defines the functional responsibility of each subsystem/component. System-level designs are typically viewed as collections of large, arbitrarily complex functional units, forming the major components of a system. Connections among units may be arbitrarily complex, carrying unspecified data and information.

5. In the system evaluation, performance and characteristics of the system-level design are evaluated against the test requirements. Several iterations of development may be needed to modify the system behavior, system structure, perhaps even the original

operations concept, and achieve a design that satisfies all of the system-level requirements.

6. The *system-level specification* is a detailed description of the system's capabilities (or required capabilities).

The activities in Figure 2.2 are repeated for each level of system development (i.e., system level; sub-system level; component level).

Specialization for Arcology Development. Figures 2.1 and 2.2 define a general-purpose process for the development of systems, independent of the participating disciplines. When discipline-specific knowledge is added to the design of appropriate development processes, some problems become more difficult, others easier.

For arcology development, models of system structure are simplified through the structured decomposition of the human habitat into groups of subsystems. Performance metrics capture the essential details of resource flows, which in turn, allow for the comparison of different types of arcologies to actual living conditions. As illustrated in Figure 2.3, a distinguishing feature of arcology evaluation is the significant role pre-existing environment and transportation infrastructure conditions play in system assessment. By itself, a new system may have adequate functionality, performance and cost. But if its impact on the pre-existing infrastructure is unacceptable, then it is unlikely the new development will be approved. A proper evaluation requires models for both the new system and the environment in which it will be placed.

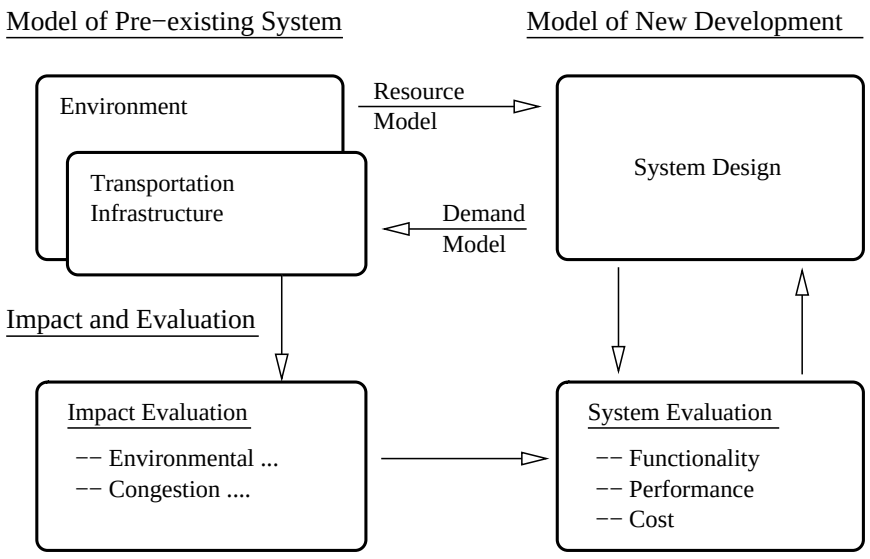


Figure 2.3: Model and Evaluation of New Development and Impact on Pre-Existing Infrastructure

2.2 Concept Development

Concept development for arcologies is formalized through objectives that capture the needs of the inhabitants.

2.2.1 Goals of SimCity

At first glance, the goal of a city (at least as envisioned in Maxis's *SimCity*^{TM}) ought to be to grow and prosper. Unfortunately, this viewpoint overlooks the city's primary responsibility to fulfill the needs and look after the well being of its inhabitants. These concerns can be captured by looking at the problem from an individual level, on par with the scale of Maxis's *The Sims*^{TM}: *The Sims*^{TM} offers 8 needs for each of their simulated characters: "Hunger", "Energy", "Comfort", "Fun", "Hygiene", "Social", "Bladder", and "Room", indicated by the green bars at the bottom of the user interface screen shot shown

Figure 2.4: *The Sims*^{TM} Entity Requirements Model

in Figure 2.4.

The model proposed in this chapter takes an even simpler approach:

Shelter: Where people live and sleep (accounts for "Energy", "Comfort", and "Room" from *The Sims*^{TM} model)

Food/Air/Water: The raw materials people need to consume to live, or at least not starve to death (accounts for "Hunger") and adds a ventilation requirement necessary in building design.

Health: Maintenance factors, such as cleanliness, waste, (accounts for Hygiene and Bladder)

Work: Most people need something productive to do when they aren't attending to their other needs. This could take the form of working for money, or being educated to increase their knowledge bank of information.

Entertainment: If people aren't doing something productive, they're probably doing something fun to while away their time (accounts for "Fun" and "Social")

In order to fulfill these needs for all of the city's inhabitants efficiently, what they are really looking at is developing infrastructure to move resources around. Accordingly, the high-level model developed in this chapter takes an abstract view of these resources and the transportation networks that move them around. This allows us to quantitatively measure the ability of the system to fulfill these individual goals.

2.2.2 Objectives

An organization responsible for running an arcology might track multiple composite performance variables and strive to pursue multiple goals. These objectives might include:

- Continuous improvement of the quality of life of the inhabitants
- Acceleration of research, development, and contributions to the educational body of knowledge.

- Optimization of resource consumption to achieve balance with interchanges with the outside environment.
- Maximization of productivity and economic performance.
- Achieving fairness by avoiding optimization of the whole at the expense of the few. Add structure to the system by providing opportunities and alternatives, not imposing restrictions on who gets to utilize available resources and transit capacity.

To be useful, each of these objectives need to be represented as a measurable quantity. The simulation model should be capable of collecting and assembling composite metrics representing these objectives, and computing their values (or valid estimates) based on simulation inputs. For example, a "quality of life" metric might be a composite of several measurable outputs, including the length of required commutes, the number and duration of times they are hit with a hunger event that can't immediately be serviced by the resource delivery system, amount of leisure time afforded after the optimal quota of daily work is done, and so forth.

2.2.3 Use Case Diagrams

Use cases are high-level representations of system functionality that do not reveal the details of implementation. Use case diagrams are a convenient way in which a real world actor (i.e., entities that are external to the system) will interact with the system, the use cases with which they are involved, and the boundary of the application.

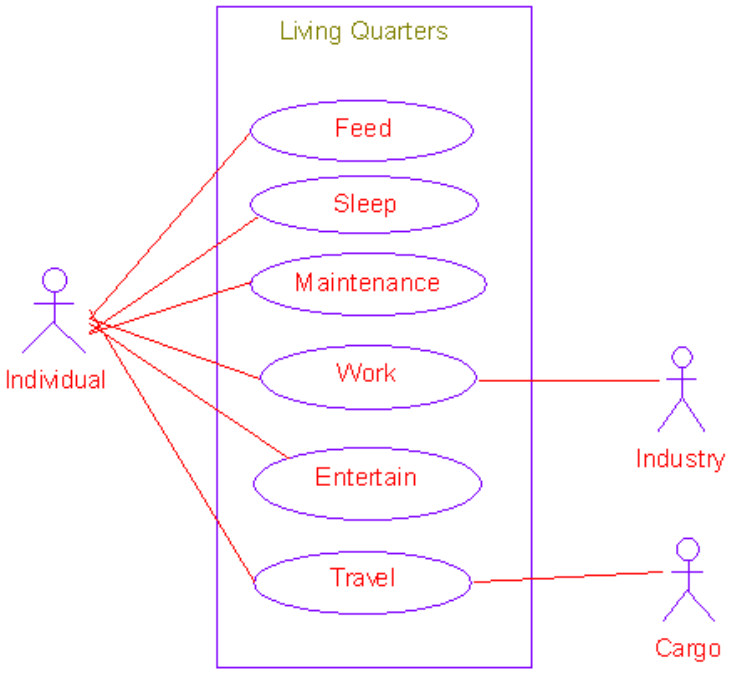


Figure 2.5: "Live" use case diagram.

The arcology use cases simply represent a few major activities engaged by individual residents. For details, see Figure 2.5 and Tables 2.1 and 2.2.

The system boundary is provided by the living quarters, which, contrary to its name, extend beyond an individual's residence and encompass all of the locations where they go about their business. The arcology simulation model will need to be flexible enough to model these types of activities in order to be used for design. The one new activity introduced by Figure 2.5 is the "Travel" interaction. Not all of these use cases occur in one location, so the Travel case takes care of moving the individual from one location to another. This interaction is performed through one of the Transportation Infrastructure classes, which will be detailed in the System Structure.

Individual : An inhabitant of the system.
Industry : Entity by which the individual is employed.
Cargo : Transportation Infrastructure responsible for moving people around (as well as resources).

Table 2.1: Actors in "Live" Use Case Diagram

Sleep : Everyone needs a place to rest for a significant portion of the daily cycle.
Feed : Consumption of food and water resources.
Maintenance : Miscellaneous cleaning tasks, such as bathing, brushing teeth, doing laundry, dishes, *etc.* would be represented here.
Work : Work is a transaction between and individual and an industry to exchange money for productivity. In this case, productivity fuels the reactions that the industry performs.
Entertain : Entertainment can take on several forms, from merely socializing with other individuals, engaging in solitary entertainment interactions (TV, games), to mass entertainment (theatre, *etc.*)
Travel : An individual is able to travel through the transportation infrastructure to commute to work or to travel to places to fulfill their other needs, such as for food or social interaction with friends.

Table 2.2: Individual Use Cases in "Live" Use Case Diagram

2.3 System Structure

Models of structure specify how a system will accomplish its purpose. Typically, the system structure corresponds to collections of interconnected objects and subsystems, constrained by the environment within which the system must exist. The nature of each object and/or subsystem may be described by its attributes.

Our basic model consists of an overall package named `GeneralHabitat`, which contains base classes and three more packages to organize resources, reactions between entities, and a separate transportation infrastructure overlay on which this project will direct its focus.

2.3.1 GeneralHabitat Package

The GeneralHabitat package contains a generalized resource queuing and transportation model of living support systems. A scenario is required to build up a model of a system by creating a hierarchy of cells that connect to each other via transportation network infrastructures. These cells would then begin to perform resource transactions between each other and resource reactions within themselves to simulate the daily operations of the system and observe it from different levels of detail, scaling from the individual to the city to the world. The transaction approach is well suited for implementation in a discrete event simulation.

Much of the model is static, such as monetary costs for resources or the structure of cells. This model is not intended to perform dynamic economic simulations or find ecological balances such as the equilibrium of birth and death rates of townspeople; those functions have

been well studied. That said, the nature of the event-driven simulation framework makes it easy to patch in such functionality by manipulating variables or cleverly reorganizing the scenario outside of the simulation.

Instead, this model is merely intended to construct a glorified spreadsheet used to perform preliminary design and calculate rough benefits analysis on making way-of-life changes, quantifying answers to such questions as: "how much energy might a city save if everyone installed more efficient light bulbs?" or "how much time and energy could we save if we staggered a city's work schedule to relieve rush hour congestion?"

2.3.2 GeneralClasses

The `GeneralClasses` object model diagram (Rhapsody's internal name for a UML class diagram) depicts the base simulation classes and generally provides a template for completing the design of any simulation based on this framework. See Figure 2.6.

All object model diagrams following this pattern of classes constitute scenario-specific use cases that highlight the use of the base simulation classes. The specific purposes of each class are as follows:

Cell: A Cell is the fundamental unit of structure. Each cell represents an identifiable entity, which contains its own collection of resources. These resources can be traded with other cells, or undergo reactions within the cell to transform groups of resources into other types of resources. Cells are containers for other cells, creating a hierarchy that can easily be traversed with recursive functions. Cells might contain any number

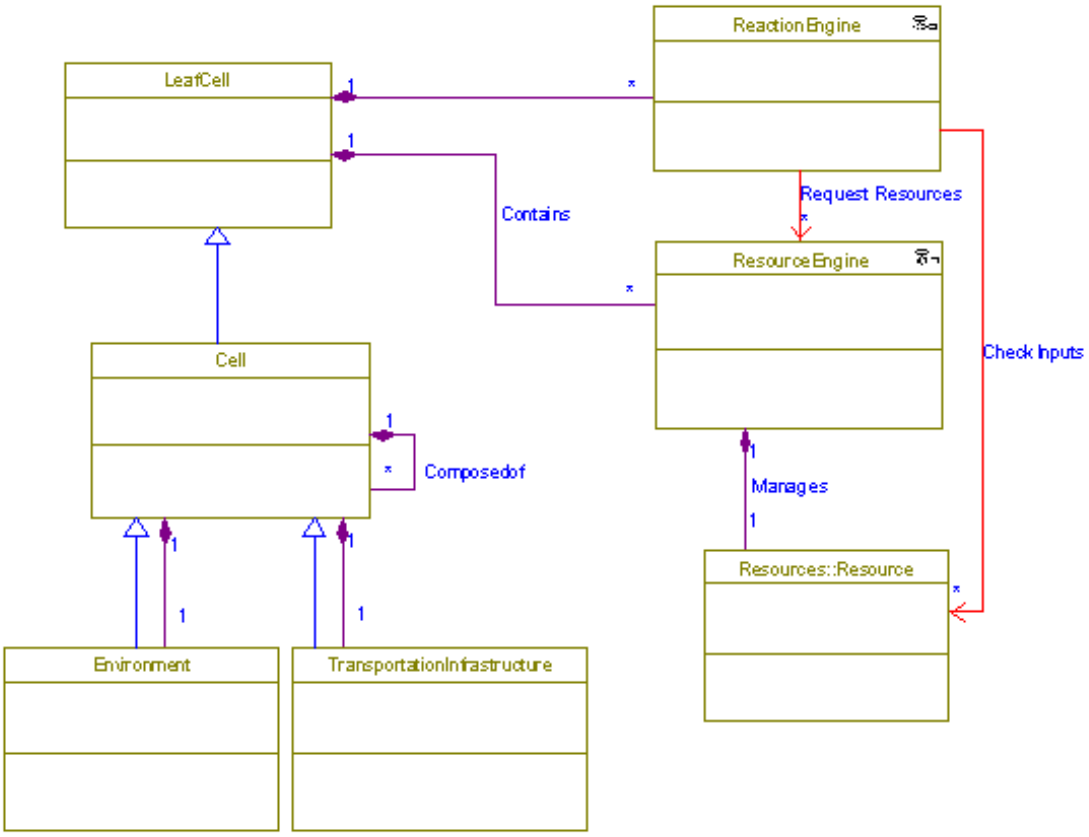


Figure 2.6: GeneralClasses Object Model Diagram

of subcells at the next level of detail down on the hierarchy, thus the Cell template is also a self-referential composition of itself. This allows us to view and gather metrics from the system on several levels of detail, from global down to the individual. To do this, we introduce the constraint that measurement of a cell's resources must always return the sum of the resources of all of its child subcells, plus any quantity it owns independently of its children.

LeafCell: Leaf cells are a special type of cell reserved for individuals and industries. These cannot be subdivided further into subcells, and thus also lack an environment or a transportation infrastructure to support any child subcells.

The handling of resources and reactions are covered by the classes `ReactionEngine`, `Resource`, and `ResourceEngine`, respectively. The `ReactionEngine` defines reactions that can occur within cells to transform one set of resources into another set of resources. The definition of the reaction governs changes to the quantities of inputs and outputs, and balances them the same way a chemical reaction would be balanced. Each resource engine keeps track of the flow of one resource within a cell. This includes the input of resource from the environment, trade of resources with other cells, internal reactions that transform resources to and from other resources, and waste resource output back to the environment. The resource engines are initialized to fire push / pull transaction events at regular intervals. Pull transactions would offer to exchange monetary resources for goods and services such as food or electricity. Push transactions relate to the expulsion of waste, and would end up in the immediate environment unless picked up by a transportation system to take to, say, a waste processing

plant (represented by an industry) first.

Resources and reactions are not yet utilized in the current programming portion of this thesis, but the hooks have been left in place for performing studies with detailed resources accounting in the future.

2.3.3 CellHierarchy

We model the area of interest by breaking it down into a hierarchy of cells and subcells that work at a different level of detail. There are essentially two types of units, parent nodes and leaf nodes, with the only distinction being that leaf nodes do not have any subcells. Levels of this hierarchy might correspond to jurisdictions of a society, as presented in the example `CellTypes` class diagram shown in Figure 2.7. It's important that all of the subcells add up exactly to form the parent cell, so in some cases, it would be necessary to define subcells that represent everything that might be left over after allocation into existing subcells. For example, the rural areas not part of a city would be lumped into a special residual "City" subcell to be included as part of a "Nation". Similarly, homeless people and vagrants would be lumped together into a special "Household" or "Community" subcell to be included as part of "City" data. This should be an acceptable practice, since these otherwise leftover units may tend to have similar characteristics.

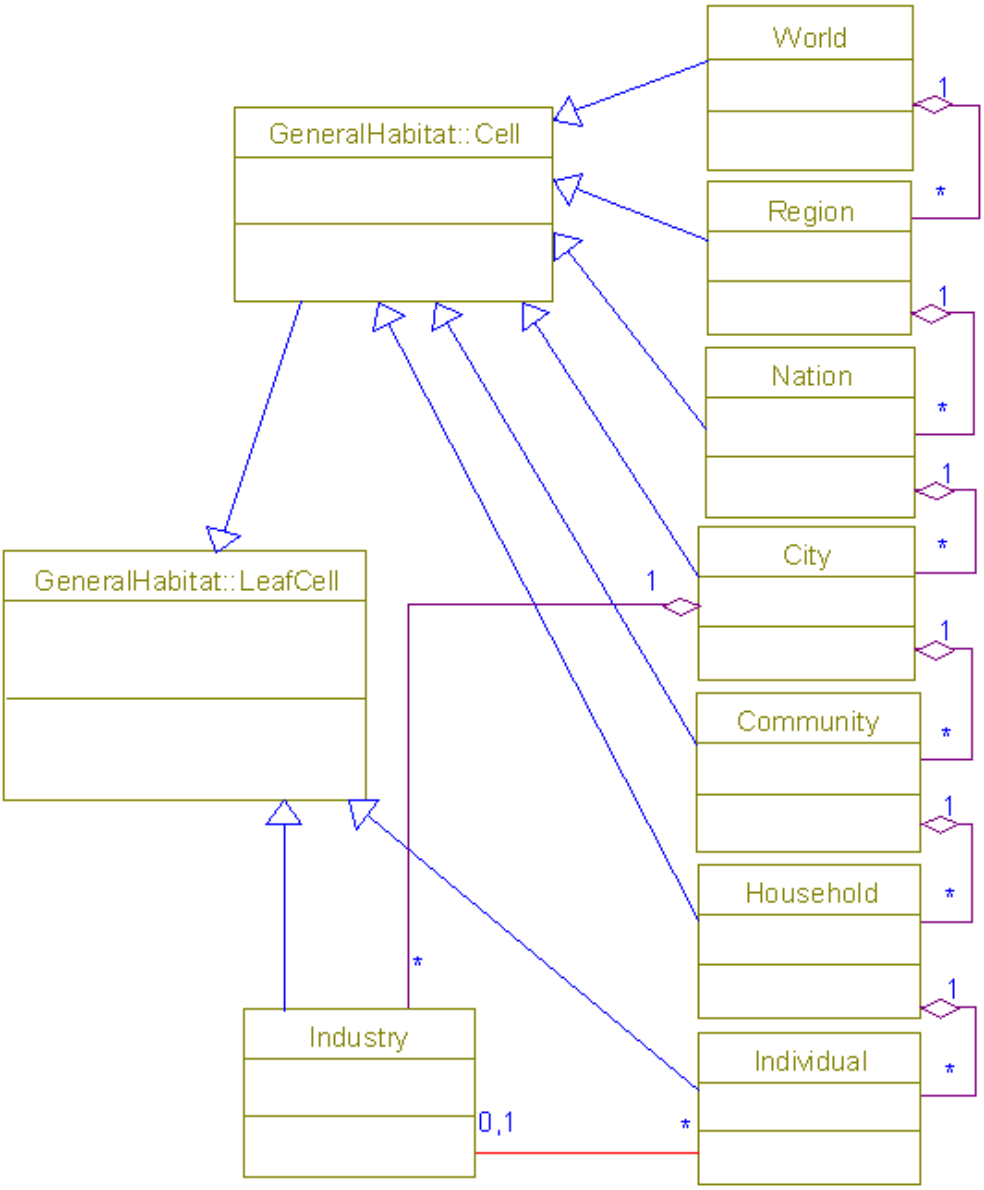


Figure 2.7: Example CellTypes Class Diagram

CellType Classes:

The purpose of the example class hierarchy lined up along the right-hand side of Figure 2.7 might possibly be described as follows:

World: This class represents the largest area of interest that modelers would most likely be interested in studying. At this high level of detail the branches that have any significant interaction with the particular unit are most important. Generally, there is no need to model the components of the cell in great detail.

Region: Geographic region tend to be composed of several nations with a common situation. Of course, large nations may exist over several regions. For our purposes, however, we will assume that all nations are smaller than the regions within which they are contained.

Nation: A nation sets the policy for controlling and tracking international trade and commerce. Data is often available on the national level for input into the top-down models.

City: A city is the highest level of organization represented by an individual arcology. Generally, an arcology corresponds to a network of connected cities. One "city" cell unit can be put aside to account for the production and consumption of all rural areas not included in other cities.

Community: Families tend to cluster into communities, which in turn form cities.

Household: A household consists of a family of several individuals living together in one residence. A family doesn't necessarily include extended family, or preclude the existence of other living arrangements such as roommates.

Individual: Because individuals cannot be partitioned into sub-components (we can only hope), they are modeled as a leaf node in the hierarchy. Most individuals will work for an industry. Individuals are free to move from place to place as part of their daily lives. This allows them to commute to work or to visit friends in another household and transfer their resource consumption to stress the infrastructure at other locations. When individuals travel, it puts a strain on the transportation infrastructure.

Industry: Cities have a special type of leaf node called Industry, which essentially employ several Individual units to perform certain specialized reactions on particular resources in bulk. Generally, they consume energy resources to refine material resources.

Environment: A special passive cell that will always yield any resources that it has and accept any waste that is ejected into it. Instead of interacting with other cells on the same level, it only interacts with subcells. So, for example, a nation's resources can be split amongst its cities, and city level waste gets deposited in the nation's environment.

TransportationInfrastructure: A special cell that interacts with subcells. It represents the connective tissue that allows resources to transit between subcells, and it takes both money and fuel in the process. Several types of transportation infrastructures can be defined with different characteristics in terms of resource burn rates.

Attributes: **Maintenance** Monetary maintenance cost incurred to keep this system up and running per unit cycle.

TransitCost Monetary cost required to move a unit of resource through this transportation infrastructure per unit distance.

Value Infrastructure build value, or how much money needs to be invested to put this transportation infrastructure in place so it can be used.

This thesis will focus on analyzing the people-mover component of a potential mass transit network.

Implementation Note. From an implementation standpoint, the "system model" can be viewed as a series of general-purpose templates, each represented as (conceptual) classes and relationships among classes. The latter translate to architectures in a software implementation.

UML diagrams of the Arcology model were created with I-Logix Rhapsody in C++ Development Edition. Work proceeded under the expectation that the code generation facilities for tools such as Rhapsody might someday be used to embed source code in the framework, and then compile and run a working executable. One of the side effects of using Rhapsody included some subtle differences in naming conventions, presumably used to simplify the merging of the standard OMG UML 1.1 specification with the practical realities of software engineering frameworks. Notably, Rhapsody uses "Object Model Diagrams" in place of both "Class Diagrams" and "Instance Diagrams", and prefixes class names with

"package" names that get translated into programming language namespaces. Since this project deals with abstract models, we will almost always be referring to class diagrams rather than specific object instances.

2.4 Transportation Infrastructure Overlay

The transportation infrastructure overlay consists of a demand model, a route graph, vehicle model, and environmental factors.

2.4.1 Demand Model

As an exercise, let us consider some of the data elements we would want a schema to include that would lend themselves to a good schedule optimizer. Each value of interest might need to be expressed and measured in different forms, to indicate whether its value has been projected from previous data, predicted based on current known conditions, or is actually measured after the fact. Uncertainties need to be attached to projections and predictions so that data elements can be used for contingency planning.

A good starting point for the demand model formulation is to list out the information a passenger or piece of cargo wishing to traverse the system would want to convey to us. The simplest schema would consist of a source location, a destination, and a desired time of arrival or departure. But much other information could also be of use:

1. Unique identifier: every database needs to refer to its elements by some unique ID at some point. Many privacy rights activists cringe every time a system forces them

to assume one that is traceable back to them. It's beyond the scope of this paper to address the requirements of what can or cannot be gleaned or pieced together by data mining this information. But suffice it to say that privacy and security concerns could be met by currently existing encryption, digital signature, and authentication technology. As an example, suppose that after payment, a unique system identifier was associated with an encrypted, one-time signature generated by the passenger's private key. Only that passenger would be able to decrypt the digital fingerprint that associated their personal identity information with the unique ID stored in the passenger roster. They would be able to prove that it was they who generated that unique signature ID at a later time, say, if they needed an alibi. However, government or private entities that somehow got a hold of the passenger roster wouldn't be able to run searches, such as "give me a list of all the people who traveled to this shopping mall" or "list all the places John has traveled to lately." For more restrictive governments or law enforcement / monitoring agencies, all or part of this data could be exposed through a key escrow system. The point is that all of this framework exists and should be set up from the inception of the system, since the security and authentication model will likely be deeply ingrained into how the rest of the software systems operate. The main problem that most privacy advocates see is that the minimum basic anonymity and data privacy safeguards are simply not being deployed into the systems of today.

2. Schedule constraints / flexibility : optimization thrives on having some slack or flexibility in its constraints. We could achieve more optimal schedules if only passengers had a

way to more adequately express constraints like:

- What range of times could they be expected to arrive at their destination? *e.g.* Not later than 9:00?
- How much extra would passengers be willing to pay to reduce their time in transit, say be giving them preferential treatment in the schedule optimization algorithm? In the same vein, would any of them be interested in paying less to reduce their "pull" on the scheduling algorithm, such that their scheduling might form by economically "hitchhiking" around on the empty seats left over in schedules generated to serve passengers paying for higher priority routing?
- What kind of safety factors or time buffers are passengers comfortable with? Would they be willing to run through an airport to make a tighter connection?

3. Accessibility needs : handicapped passengers could make special requests to suit their situation. This could help budget transfer time and resources better. For example, instead of equipping all of the vehicles in a fleet with minimal accessibility features at great expense, a bus system could have 5% of their fleet be fully equipped and serve handicapped passengers as their first priority.

Cargo would have much of the same properties as passengers, perhaps a few more to encode other special handling instructions, hazmat designations, and so forth. As cargo might spend significantly longer stretches of time in the system between warehouses and transfer stations, they might have more stringent tracking and tagging requirements, as

well as more flexibility in routing preferences, especially between low priority bulk and high priority overnight shipments.

Security is an important concern in a system that can be misused for malicious purposes. While we could easily set up detection stations at central transfer nodes to screen for explosive and hazardous materials and other contraband, we'd want to take another step to ensure that the sender can be traced and held accountable for the contents of a package. The system should require some form of digital signature and authentication from the sender in order to enter a package into the system.

Having all this passenger and cargo data pretty much takes care of knowing the transportation system demand inputs.

2.4.2 Route Graph

The next set of standardized data should describe how the transit network itself is set up to handle the demands placed on it. Every transit system could be expressed as a network, so we will liberally apply terms from the networking field to describe some of these concepts. The first assumption we'll have to make is that any transit system could be expressed and modeled as a collection of nodes and connector links. They might vary significantly in complexity and level of detail between transit systems, but they all need to be able to "plug in" to each other for inter-modal optimization to work properly.

A simple light rail or tram network might consist of a few dozen stations connected by a single track. On the other end of the spectrum, a metropolitan road network modeled in

detail would have thousands of connective paths, links to probably all of the other nodes of transit, relatively few fixed source and destination nodes, and likely not enough user planning data will ever be made available to predict traffic congestion resulting from construction, weather, accident, or just plain rush hour delays.

Minimal Transport Network Representation. The minimal elements needed to represent this transportation network would include:

- A unique node identifier
- A geographic node location, represented in a standard reference frame such as the WGS-84 latitude, longitude, and altitude used by GPS.
- A connectivity matrix, minimally of transit times between node pairs. A special value would indicate that certain node pairs (probably most of them) are not connected at all. This might even be digested from much more complicated routing algorithms, such as street navigation systems. The connectivity matrix will need adjustments over time, to schedule in planned closures for maintenance, or new routes opening up at particular times.
- Buffer and storage nodes, such as maintenance bays or taxiway queues. These might have special properties with regards to what activities or states can and cannot take place.

2.4.3 Vehicle Model

In order to actually traverse this network, a transit system ultimately needs vehicles. Each vehicle would have associated with it:

- A physical location within the network, whether a geographic location in transit, or a position in a queue waiting for arrival at a station node, or occupying a storage or a maintenance bay.
- A passenger or cargo capacity
- A set of rules governing how fast it can navigate across its network, how long it takes to load and unload, *etc.*
- Various maintenance details, such as fuel supply, crew refresh schedules, and at least some indicator of the probability that it will reach its destination without breaking down along the way or running late for some other reason.

The system would need a way to introduce its own arbitrarily fixed schedule or other constraints. This could be required merely as a way to allow legacy timetable-based systems to nominally interact with an optimized scheduling system. While we could squeeze a more optimal solution by imposing fewer constraints, for various reasons (suppose a communications equipment failure prevented us from giving a last-minute reroute to a vehicle), we need some way of communicating and enforcing pre-existing or immutable schedule constraints. This mechanism probably would be similar to one we'd use for introducing planned or unscheduled maintenance stops.

2.4.4 Environmental Factors

The last major category that would affect the performance of the system might include "environmental" factors. These factors could either be predicted in advance with some degree of certainty, or suddenly evolving events such as accidents or breakdowns that require a reformulation of the optimization problem to mitigate.

Weather conditions can have a predictable effect on a system. Updates on rain or snowstorms should be able to make their way into the system so it can plan on having some degree of constrained capacity in advance. Airports can plan to shut down for a few hours while "convective weather cells" (thunderstorms) pass by overhead. As better forecast data has become available, air traffic control centers have actually been able to institute ground delay programs for aircraft all the way at their points of departure, so they don't end up circling in holding patterns near the destination airport, waiting for the inclement weather to abate. Such techniques for contingency planning based on externally available data could make their way into streamlining other forms of transportation such as road and rail, albeit less dramatically.

These types of entries could manifest themselves as time-dependent changes to the network connectivity matrices and node constraints. Each cell would have a probable new value for transit time on that link, accompanied by probable start and end times of that effect.

2.5 System Behavior

The arcology simulation model is based on a discrete event simulation framework. This means that state changes in the system structure are triggered by the firing of events which occur along a global time line event queue. The model executes by populating the global time queue with scheduled events and firing those events in order. Every time an event is activated, the system global time is advanced to that new time. Any state transitions in the model that were blocking on this event are executed so they can perform their activities, which often result in the scheduling of more events in the future. Thus the simulation perpetuates events and continues in time until there are no more events left on the simulation timeline event queue.

2.5.1 Modeling the Behavior of an Individual

The individual transitions from state to state in their daily activities triggered by these events. A fairly simple schedule could be arranged as follows to implement a state chart representing a typical person's day. The state chart depends on having the right combination of events defined and triggered to advance the individual through the full daily cycle.

A massive quantity of these individuals independently going through their daily state cycles as indicated by Figure 2.8 would create a load on the system, either by requesting resources that must be delivered to them, as well as moving the individuals themselves about the transit network as required. At the same time, we can poll the history of each individual

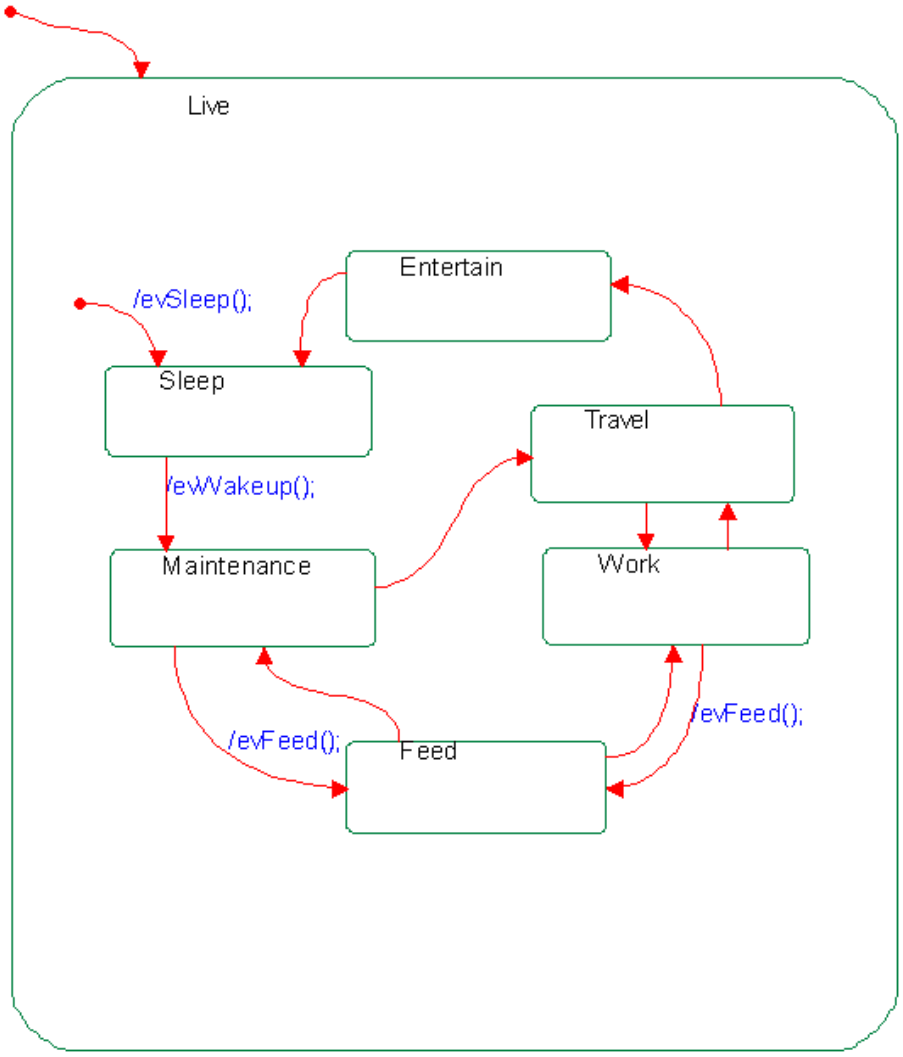


Figure 2.8: State Chart Model for Behavior of an Individual

to collect performance metrics on how well the system can respond to their demands.

2.6 System Performance/Measures of Effectiveness

The complete city system can only improve properly if we choose the right performance metrics to judge it by. An optimization function that optimizes the wrong metric will certainly cut you short of fulfilling your goals. For a city, the metrics we would want to track include:

1. Resource production/consumption ratio per cell. An effective system would need to be efficient at doing a lot with the resources it has available to consume. We should emphasize efficiency over minimizing total resource use, since the latter would often result in some form of stagnation.

2. Transportation overhead. We should establish metrics to track the ratio of resources spent on the connective infrastructure compared to the nodes and activities it actually supports. Of course, this also needs to be balanced with the need for growth and interconnectivity, so allocating resources towards maintaining and extending transportation systems should be a secondary goal to increasing productivity.

3. Sustainability. The environment is usually the first to give up resources or absorb waste when they can not be serviced elsewhere. However, we often do not know the environment's total capacity for restoring waste byproducts back into useful resources. The burden should be placed on the industries who exercise the environment the most

to prove what that capacity might be in order to achieve a suitable equilibrium.

4. Quality of life. We can measure this by tracking the ability of the system to respond to the population units' resource requests to maintain their desired standard of living. It will be difficult to establish a calibrated baseline, however, since this performance measure will be relative to historical measurements. We can only attempt to keep the quality from dropping below former levels.

Chapter 3

Framework for Arcology Simulation and Analysis

This chapter reports on the development of a simulation framework for arcology simulation and analysis. Emphasis is placed on the specification of an urban model that will generate demand on a transit network, a system simulation engine, and a schedule optimization algorithm capable of directing the action. Development activities follow the systems engineering framework described in Figure 2.1; that is, use cases lead to scenarios, simplified models of required functionality, and the identification of (functional, performance, economic) requirements. Then, models of simulation system behavior and support for real-time optimization of behaviors (i.e., transit schedules) are specified.

A secondary purpose of this chapter is to introduce technical considerations and design goals of mass transit systems based that might be based upon this framework. Of course, the ensuing chapters will fill in specific details of the mass transit system model, types of transit scenarios, and details of the actual software implementation.

3.1 Simulation Tool Use Cases

As detailed in Chapter 2, use cases are a mechanism for eliciting and structuring models of system functionality. Basic questions include: What will the simulation framework do? What are minimal levels of support for specification of scenarios, simulation, and post-

1. Set up a modeling scenario using input data.
2. Execution of simulation model to produce output data. The ability to perform multiple scenario simulations in parallel would greatly increase our data analysis throughput.
3. Postprocessing and analysis of output data into performance metrics.
4. Design of experiments methodology for exploration, optimization, and parametric analysis of the solution space.

Table 3.1: Potential use cases for the simulation model

simulation analysis? What types of decision making support will be provided? Table 3.1 contains a first-cut list of potential use cases for the simulation model.

Support for Scenarios. Part of the difficulty in framework development comes from a lack of specific constraints and, in this case, the need to simulate behavior in urban areas, the transportation overlay, and any potential interactions.

A well-designed framework will support assembly of simulation scenarios from multiple perspectives. In particular, we need to be able to build up scenarios in two ways: (1) through composition of collections of entities in a bottom-up manner, and (2) by breaking down aggregate data sources into a distribution of entities in a top-down manner. The latter would allow us to compare scenarios assembled from data gathered from a sampling of individual detailed sources, and from scenarios initialized using totals gathered at a higher level of detail. For example, we might know the number and type of cars registered in a city, their fuel efficiency, and the estimated distance they travel daily. From this we could assemble a simulation that gives us their average fuel consumption. Conversely, we might take a survey of all of the gas stations and know exactly how much fuel was pumped into

a number of vehicles in the city, and from that deduce the distance each of them drive. By comparing similar scenarios constructed from different data sources, we might check our models for consistency as well as calibrate it under a variety of existing sources of data.

1. **Bottom-up Scenario:** Since the arcology is designed from the ground-up, starting at the individual level, the structure of our model would allows us to calculate the aggregate performance at higher levels of organization, such as a the city and national level.

    - Define number of simulation units, connectivity between units, schedule of transaction events, schedule of reaction events, initial conditions.
    - Output aggregate performance for groups of units.

2. **Top-down Scenario:** The present day scenario is built in a top-down fashion from various data sources. Statistics are only tracked from relatively high levels on the organizational hierarchy, so we must extrapolate some data to flow down to fill the detailed subcells of the structure.

    - Define high-level consumption rates for groups (using publicly tracked and available data), provide distribution histograms for each type of resource and transaction rates for each subunit.
    - Output unit-level quality of life, performance.

Support for Simulation/Analysis. One of the main types of problems this transportation

network simulation needs to address is passenger demand generation. Different types of transportation infrastructures could be evaluated against each other to determine how well they meet that demand. Many existing transportation problems tackle ways to increase throughput or capacity. But the task of urban planning should also focus on arranging its physical layout to economize demand loads in addition to building out transit infrastructure for maximum capacity. For example, instituting staggered work hours or telecommuting programs can relieve peak rush hour traffic congestion without spending a fortune widening highways or building additional transit lines just to increase throughput for a few hours of peak usage a week. Local governments should know how much incentive they ought to offer to businesses to encourage them to implement flexible work hours. Similarly, they'd want to know how much to invest in telecommuting infrastructure (such as municipal broadband) in order to provide productivity benefits similar to simply adding highway lanes or additional thoroughfares.

By simulating demand, we can also create a transportation system that is more sensitive to the needs of individual travelers rather than the aggregate flow of passengers. This would allow us to create schedules around the traveler's itinerary rather than forcing the traveler to always plan around fixed train, bus, ferry, and aircraft timetables. By only tracking passenger flow through fixed schedules, we throw away some valuable data on when the passenger really wants to depart or arrive, which is a significant factor when comparing mass transit to personal vehicle use. For instance, if everyone starts work exactly at 8:30, but buses only run hourly on the hour to that particular stop, then the extra half hour everyone

spends waiting per day essentially counts as extra commuting time in their books. The system operators might think they're doing quite well by only measuring the time passenger spend sitting on the bus and making connections, but the passengers would perceive much higher inconvenience and time costs.

A more effective public transportation system should succeed in making the world "smaller" by making each district of a municipal area more readily accessible, allowing people to travel between places where they live, work, run errands, and seek entertainment. Under the trunk and feeder paradigm often used to organize mass transit in metropolitan areas today, travel through the system can take considerable time unless your source and destinations happen to be near major hubs or just down the street from each other on an established route. The worst case scenario for many trips off of a main trunk line would consist of catching a local feeder bus route to the nearest trunkline light rail station, making a transfer at a major hub or two, and finally catching another feeder route to your ultimate destination. Each transfer would typically consist of at least 5-20 minutes of waiting for the next connection. Compared to driving your independently owned vehicle, public transit would often take two to four times longer, even with traffic. Commuters would travel twice per day, so the time savings of taking a personally-owned vehicle could add up to an additional 1-2 hours of personal or family time at home each day. Public transportation systems could use much improvement to make mass transit preferable to driving, but often it becomes an alternative to escape congestion on the roadways rather than the primary mode of travel. Drivers typically support investment in public transportation only insofar

as it gets other cars off the road.

An advanced busing system (such as the HCPPT system proposed by Cortés[13]) would dynamically generate routes and schedules based on individual source and destination requests from each passenger, and thereby achieve efficiencies and meet customer requirements far better than current fixed schedule transit fleets. This could make public transportation much more attractive to people who drive their own vehicles everywhere in order to maintain that degree of flexibility. During peak commuting hours, intelligent scheduling has the potential to reduce individual commute times, as most buses could be scheduled more like express routes, filling up at one location and proceeding directly to stops at a common destination with minimal stops or transfers or jaunts down back roads along the way. During off-peak hours, buses would not run nearly empty along the exact same routes at a drastically reduced frequency, but would run only on demand, cutting down unnecessary wait times and making them more convenient for midday or late night errands. All we'd need to implement an intelligent, dynamically reroutable transit system is a robust communications network with a simplified interface to provide route updates to vehicles and transfer instructions to passengers.

3.1.1 Performance Metrics

What defines a good inter-modal transit system? The conflicting goals might be characterized as: speed, latency, coverage, and efficiency.

- **"Speed"** refers to how fast the transit system can get a passenger or cargo item from point A to point B. Unfortunately, this does not depend entirely on the cruise speed of the vehicle alone, but also time spent making transfers and additional preparations (such as passenger check-in and luggage screening at airports)

- **"Latency"** refers to additional waiting time between when a passenger wants to go somewhere and when the transit system can actually take them. It is affected primarily by the frequency of service, particularly how well it matches demand. Extra time that people have to wait at their source or destination should be counted against the system — though this is often overlooked in transit performance metrics today. The data just isn't available, or people have relegated themselves to adjust their schedules around the system's timetables. This "latency" metric will usually be at odds with efficiency due to economies of scale, since making passengers wait longer times between pickups can cluster them into larger groups.

- **"Coverage"** refers to how well the transit system covers the service area, which should include how far people have to walk from their doorstop to enter the system. Broad coverage is more difficult to achieve for a mass transit system, especially as population density decreases and residences and businesses become more spread apart.

- **"Efficiency"** might refer to two terms: monetary frugality on fixed infrastructure and operating costs, as well as in terms of conservation of fuel and resource utilization. Efficiency pretty much always counterbalances against each of the three other goals,

so we often must express how much extra money or fuel we are willing to expend to achieve gains in speed, latency, or coverage.

3.1.2 Transit Schedule Optimization

To achieve reasonable improvements in these multiple competing performance goals, we must employ some manner of vehicle fleet optimization in order to investigate the full potential performance of a transit platform. An intelligent transit system would take advantage of existing and emerging ubiquitous computing and communications networks to make itself more responsive to customer requests and to deliver passengers in a more coordinated manner. While fixed route schedules might work well for normal demand based on projections, the true performance of a transit system should take into account all available emerging information. This could include advance knowledge of special events that create spikes in demand to certain stations, or even unforeseen events such as breakdowns that close transit lines and trigger a failure mode of operation with an altered schedule.

An optimization problem formulation should dynamically take in data about the arrangement of stations, passenger requests for transit between available stations, and properties describing the vehicle fleet and constraints pertaining to their movements between stations, and provide an optimized plan for delivering passengers to their destinations. This would allow us to identify and focus on the performance impacts of physical design parameters independently of the often arbitrary fixed scheduling methods. The optimized schedule could be calculated in near real time, incorporating updates from the evolving system state

in a rolling-horizon fashion. It would incrementally compute the next series of interchanges and transfers in the future, at first working on projected demand data augmented with more solid data collected from vehicle and station sensors.

Once we identify transit scheduling paradigms that perform well, we might simplify the network somewhat by establishing some fixed strategies, patterns, or even routes. However, we see no reason why a fully dynamically-reoptimized transit system couldn't guide its passengers through a constantly updating and changing network or vehicles through the use of mobile phone text messages or a dedicated digital guide.

3.1.3 Coordination of Dynamic Optimal Schedules

The main way we'll be able to improve the efficiency of mass transit (aside from simply improving fuel economy) would be to use existing resources more effectively through extensive use of optimization. With enough planning and foresight, optimal scheduling is straightforward to perform. However, things never quite go as planned, due to a variety of unpredictable factors such as weather and accidents and just plain last-minute changes in schedules. In order for the optimal plan to be of much use, we ought to continually collect enough data in real-time to monitor and reevaluate schedules as able. This requires that we have a communications system in place that allows us to poll the status of our cargo, passengers, and transportation vehicles. Equipage for this type of system would have been cost prohibitive in the not-too-distant past, but now that geolocation devices, mobile computing, wireless networking, and cellular data network backbones have become nearly

ubiquitous, we'd be silly not to put all this capability to good use.

So instead of having fixed timetables locked down and fixed weeks, months, or even years in advance, based only on projections from previous observation of seasonal flows in the past, we could perform schedule optimization on actual data. This data would factor in individual requests from each customer, including their destination and schedule constraints (or better yet, their schedule *flexibility*). Vehicles could report their current location and status, meaning they'll always be right on time - especially since they could report their arrival time themselves. Monitoring and reporting of deteriorating road or weather conditions could automatically update the schedules of every vehicle in the network to account for and mitigate the effects of new delays.

3.1.4 Comparison Framework for Multiple Urban System Models

Now that we have specified a model for urban-enabled demand on a transit network, a system simulation engine, and a schedule optimization algorithm to direct/control the action, we can set up an iterative optimization of system design that will analyze the design parameters for each particular simulation scenario. This framework will help us evaluate urban design and infrastructure in ways that should help drive progress towards efficient and sustainable societies that serve the people who live in them.

To see the usefulness of this framework, consider the following scenario. We could propose a new construction or infrastructure project, show its benefits in this kind of a simulated model, and later validate those benefits using data collected from the real system.

Competing developers might even submit simulations of their designs to provide benchmarks using this analysis platform.

The ability to compare several optimization components, several system structures, different modeling methodologies, all using the same data interchange format to facilitate direct comparisons between both real and simulated evolution of the scenarios, allows us to take a systematic, objective approach to tackling urban improvement projects. Adapting such a simulated and real system performance comparison framework will allow us to have more complete impact assessments by making sure every study or proposal is analyzed consistently, using the same inputs, and doesn't sweep away or ignore unwanted side effects and consequences. Urban planners could use these studies to provide ammunition for driving changes toward the way they envision their communities. An intensified focus on operational efficiency and continuous improvement driven by pervasive measurement and analysis will lead towards a leaner, sustainable society where we could direct a higher ratio of resources towards forward progress instead of mere subsistence.

3.1.5 Transit Simulation Requirements

With that, we proceed to develop the outline for what our simulation must accomplish. See Table 3.2.

1. Read scenarios as inputs
    1.1. Hierarchical layout of transit stations and connectivity
    1.2. Initialize simulation entities involved (people, vehicles, residences, and workspaces, *etc.*)
    1.3. Load demand schedules for passenger movement
2. Simulation execution to evolve state of system entities
3. Output metrics defined and calculated
    3.1. Passenger measures of performance
    3.2. Vehicle measures of performance

Table 3.2: Simulation Requirements

3.2 Simulation System Structure

We used the Umbrello UML tool to create diagrams in this and subsequent chapters.[21] While its code generation capabilities are not as comprehensive as that of the commercial tools such as I-Logix Rhapsody, it can generate class stubs for several languages including Python objects (which is used for the simulation), and XML Schema (which might be used eventually to help validate data interchange).

3.2.1 Discrete Event Simulation Framework

The prototype framework depicted in Figure 3.1 consists of three major parts: scenario generation, simulation, and schedule optimization. These components are wrapped around a scenario generation and post-processing framework that creates a directory tree of scenarios, each with slightly different initial conditions for factorial analysis.

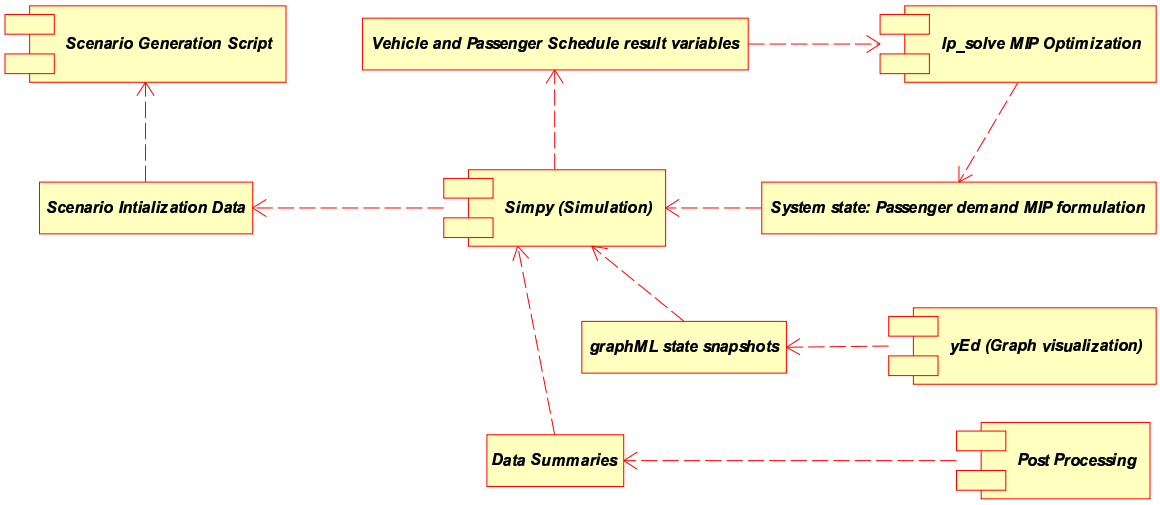


Figure 3.1: Data Dependencies Among Components in Simulation Framework

Implementation Approach. The simulation code written in Python makes heavy use of the SimPy module,[43] while the solution of the schedule optimization problem occurs using the LP-solve mixed integer linear program solver.[8] Python is a high-level programming language supporting object oriented programming. Its clear syntax and various interface modules make it ideal for rapid prototyping.

The schedule optimizer formulation was first prototyped in Perl before being ported to Python. While Python typically has a much better reputation than Perl in terms of code readability, Perl's in-line variable output syntax made its code terse and much easier to read and debug than the equivalent Python code.

Being the "brains" of this framework, the more complicated optimization problem imposes a few major constraints that help simplify the way we'd model our transit network. The transit system must consist of a network of "station" nodes representing the entry, exit, and transfer points for passengers and cargo. Passengers and cargo can only move between

nodes on vehicles, which can transfer them between any two connected stations or waypoints at regular, synchronized intervals. The simulation can handle several vehicle types, which can differ in passenger capacity per vehicle, cost per transit segment, connectivity graph between nodes, the maximum number of vehicles allowed to visit a station at the same time, and a host of other measures and constraints.

Modeling Difficulties. The most crippling part of the model deals with timing. Time is dealt with in terms of synchronized discrete timesteps, during which the state of the entire system can be represented at one point in time by a complete set of variables. At each time step, the state of the system must be such that every vehicle is located at a station or waypoint node. By the next time step, all passenger transfers must have been made and all vehicles must have completed their transit toward the next station node (or else stayed in place at their current station). When the simulation translates these actions to events in continuous time, this means that all stations synchronously act in unison, where every vehicle practically departs simultaneously, travel all at the same time, and offload passengers at their destinations, and wait for passengers to transfer to make connections. They all must act in rhythm to the beat of a central drummer. While this obviously constrains the flexibility of the model in a big way, this arrangement allows the schedule optimizer the flexibility it needs to balance hub-and-spoke transfers with more direct routes, depending on the capacity and economics of the vehicles made available.

Therefore, the model used in this analysis is that of equal-time segments separating transit stations and waypoints. Each and every vehicle must wait for passenger transfers to

complete before they continue on to their next destination. The result is that, in reality, some amount of time is bound to be wasted under this model as all vehicles must stop and wait for the next beat of the synchronization drum before proceeding. A vehicle that is running a little bit late might have to skip a beat completely.

3.3 Simulation System Behavior

The Scenario Generator runs first and generates a directory hierarchy of scenario data and a makefile that allows us to run simulations in parallel and only on scenario data that has not yet been computed. This allows us to add data points to large data sets without having to spend a lot of time recomputing existing results.

Snapshots of the simulation state occur regularly and trigger the creation of new optimization formulation files, which might be solved by Python's LP-Solve module several times throughout the evolution of the simulation. The optimization returns an object with a list of schedule results referenced by entities in the simulation to determine their actions.

After the runs are complete, the yEd graph viewer can neatly organize and display state snapshots of the transit entities in its pseudo-UML object view.[47] These visualizations of the simulation's graphML state dumps greatly assisted development and debugging. An example is depicted by Figure 5.2.

Finally, a post processing script assembles the raw output of SimPy data monitor taps into histograms for collective display on a summary web page for comparison. It also condenses a spreadsheet of input parameters and output metrics for each scenario to allow

further data reduction as described in Section5.6.1.

The program uses the Psyco and SciPy modules to greatly increase the execution speed of most operations. Simply including the Psyco optimizer replaces certain commonly-used interpreted Python routines with optimized native code that gives the overall model a 1-2 order of magnitude increase in computation speed.[39] The SciPy library additionally gives us a high performance numerical library for performing calculations and efficiently manipulating large matrices.[35]

3.3.1 Statistical Scenario Comparison

The simulation uses some pseudo-random distributions to initialize demand curves. In order for our simulation runs to maintain repeatability, each of the scenarios include an initial random seed. A simulation run with the same seed will always generate the same random variables. Conversely, we can also vary the initial random seed across several runs of the same scenario in order to do Monte Carlo type simulations that gives us a proper distribution of output metrics as well.

The use of randomized initial distributions has another useful feature, in that it prevents the optimization problem from becoming too symmetric. Too much symmetry would result in multiple equal-cost branches to search exhaustively. So adding a touch of entropy to the our system allows our MIP solver to converge on a better solution slightly faster.

3.3.2 Computational Considerations

Large scale global optimization falls under the class of NP-hard problems that scale exponentially with the number of transit nodes we add to the transportation system. Accordingly, the simulation is set up to run in parallel across several CPUs, scalable to a massive clustered system with a shared filesystem. Some linear and mixed integer solvers also have the capability to partition and solve individual optimization problems in parallel. Unfortunately LP-Solve is not one of them, but it does have the ability to export its model into other optimization problem formulation language formats such as CPLEX or the industry standard MPS, so we need not worry about supporting parallelization.

We still would need to resort to a host of other tricks to reduce the computational complexity enough to approach problems of any appreciable size. Most of them involve introducing some sort of constraint to reduce the number of branch and bound paths search in the solution space. But primarily we resort to LP-Solve options that allow us to accept suboptimal but feasible solutions.

- The easiest way to reduce the computational complexity is to partition the problem into smaller parts. Since these types of "traveling salesman" problems scale exponentially with respect to the number of nodes, the number of branches to search would be drastically reduced by replacing an exponential term with a linear one.

- Adding link constraints is also another way of reducing the search space. Not every node needs to be linked to every other node. So often we will resort to building a connectivity matrix to define which source nodes can get to which destination nodes.

With road and rail, only adjacent nodes are directly connected. Distant nodes would require transit through other city or station "nodes"

- With aircraft, of course, most vehicles can travel directly from any node to just about any other node in the network. In this case, it may be helpful to add "max passenger transfer" constraints, to keep the system for searching through impractically long schedules. An itinerary that made a passenger jump between more than two or three connecting airports would likely be rejected by that person. On the other hand, low priority bulk cargo may find some cost advantage through waiting for these multiple connections, filling in otherwise "empty" space leftover on flights that could get closer to its destination. But at some point all of the extra handling and transfer overhead ought to outweigh the cost saved.

- Any constraint that would help "lock down" otherwise free variables would help reduce the search space. Feeding in initial conditions - like the current locations of the vehicles in the fleet, or stops that must be made by a certain time (for example, to ensure buses take all passengers to a stadium well before a game starts) may help speed the optimization solution search along.

- Sometimes it may be necessary to simply add other heuristic or even arbitrary constraints to help the system converge on a solution. By design, many of these constraints might not even affect the final solution, but constrain the search space enough to deliver an answer more quickly.

All else failing, many mixed-integer programming solvers, including LP-solve, can return "good enough" solutions without completing an exhaustive search of the solution space. Modern MIP solvers can also be pretty clever about searching the "most promising" paths close to the relaxed linear solution first. Completing the exhaustive search often yields little incremental improvement over the best previously discovered feasible objective. Of course, this technique applies only if a feasible integer solution can be found in reasonable time at all.

LP-Solve has a host of options that can allow it to return suboptimal solutions before completing an exhaustive branch-and-bound search of the solution space. It can simply provide the first MIP solution it finds after locating the relaxed linear solution using its simplex solver. We can also converge upon solutions faster by increasing its MIP gap tolerance, which is the relative improvement between suboptimal solutions it finds along its way. This effectively reduces the "depth" of its branch-and-bound search. Finally, it also offers a wall clock timeout that stops an optimization from running virtually forever and returns the current best suboptimal solution.

Finally, a sophisticated optimization could involve pre-computing most of the possible schedules in advance, and then have the ability to account for the effects of small changes with only minimal recalculation of the final optimal solution. This type of "warm start" optimization could help recover the schedule quickly after small, unexpected breakdowns. Suppose a vehicle suddenly announces that it will be arriving 30 minutes late to a hub node. If recomputing the entire optimal solution taking this new information into account would

take a few hours of number crunching, we obviously don't want everything to grind to a halt while waiting for the scheduler to tell us what to do next. A warm start optimization would minimize recomputation, perhaps by determining a subset of decision variables which would likely be affected and formulating a highly-constrained problem that only searches through variables linked to unexpected changes in one or two input values. We'd need to develop a heuristic to determine exactly how far out this limited set of affected variables should reach.

Another warm start scheme might involve jumping back into a computational snapshot of the state of the large optimization and only recalculating internal values that have changed with the modified inputs. Perhaps some solvers have this ability.

3.4 Model Validation Against an Actual System

Ultimately we would want to calibrate our simulation against an actual transit system modeled by it. Due to the discrete timestep nature of our schedule optimization model, the simulation would only be capable of providing an approximation of a live system's performance. However, if the live system uses the same schedule optimization algorithm used in our simulation, we wouldn't expect simulated versus live performance to differ appreciably unless vehicles run late and passengers miss connections. The simulation currently does not model these types of unexpected events, but adding such probabilistic failures to the simulation shouldn't pose much of a challenge. The challenge lies in calibrating those probabilities against those that might occur in the live system due to factors discussed in section **??**.

Chapter 4

Multimodal Mass Transit Simulation

This chapter describes specific details of the schedule optimization problem formulation and modeling components used in the simulation of commuters traveling from their residences to their places of employment. Commuting makes a good starting scenario, since it accounts for a large proportion of the trips handled by present-day mass transit systems. Furthermore, the commute simulation may easily be adapted into other types of scenarios for analysis such as shopping errands or stadium events.

The formulation of the schedule optimizer is most similar to the mass transport vehicle routing problem (MTVRP) introduced by Pagès.[37] It creates schedules for moving a fleet of vehicles between a network of stations in such a way to deliver as many passengers from origin to destination stations as possible.

4.1 Framework Capabilities

The primary features that this optimization framework sought to achieve include:

- Demand-responsive routing rather than operation on a fixed schedule. This is necessary for us to worry less about generating transit designs around peak demand levels that do not function as efficiently with nominal demand levels. We also hope that

the system would utilize command and control networks that take advantage of available communications infrastructure to book requests and guide passengers through the system.

- Allow optimal transfer strategies to emerge. At different loading levels, the system vehicles may organize themselves like "hub & spoke" / "feeder & trunk" networks for efficiency, or begin to resemble more direct point-to-point routing during lighter loading or when existing hubs become constrained.

- Multi-objective goal functions, including terms for maximizing service quality such as:

    1. high throughput
    2. low average latency from sources to destinations
    3. efficiency terms that would minimize general operating costs associated with the number of vehicles operating in the fleet and the number of segments they would have to travel

4.2 Concept Requirements

This section describes the pathway from high-level goals to specific use-cases for the transit system, the mass transit optimization goals, fleet schedule optimization objectives, and transit use cases.

4.2.1 Mass Transit Optimization Goals

This simulation constructs a simple transit network with passengers traveling from source nodes to destination nodes. The scheduler attempts to provide an optimal or quasi-optimal schedule of transit fleet vehicles with various capacities, operating costs, and nodes serviced that will transfer the passengers to their final destination. Through parametric analysis of different demand loading and network topologies, we hope to define some characteristics of urban areas that enable the system to meet the opposing passenger demand and vehicle utilization objectives efficiently.

4.2.2 Fleet Schedule Optimization Objectives

The objective function of the transit vehicle schedule optimization is a weighted composite of the number of passengers served, the time they are delivered, and a flat cost incurred per vehicle leg traveled. The weight on each objective typically puts them on different orders of magnitude, such that a secondary objective should not be considered a factor until the primary objective reaches an optimal configuration.

The relative weighting of objectives is arranged as follows:

$$Objective\ 1 \gg Objective\ 2 \gg Objective\ 3 \gg Objective\ 4$$

Objective 1: Maximize the number of passengers delivered to their final destinations. All passengers are currently weighted equally, which means during instances where the system is operating beyond capacity, the optimizer will favor passengers who are close to their destinations. There is currently no zone tracking to ensure that passengers

traveling long distances can "pay more" to compensate for the higher transit cost. The objective function also provides no reward for moving passengers partway, so a feasible solution will either move a passenger all the way to their destination node or not at all.

Objective 2: Minimize the amount of time the passengers spend in the transit system. This is accomplished by adding a linear bonus term to the objective function that rewards the system for delivering passengers to their destinations at earlier times. These terms push the schedule "left" towards earlier arrival times. Otherwise the system would allow people to wait unnecessarily during the time window under consideration.

Objective 3: An optional objective to minimize deviation from a desired fleet size can be activated. We could simply minimize the number of vehicles in use, but we'd have to find some way to balance this with the passenger service objectives. Plus, most service operators have a fixed number of vehicles and drivers to employ. The optimizer could take advantage of extra vehicles to improve passenger service quality, as well as make recommendations at to when the operator might want to rent additional vehicles and drivers temporarily to meet demand.

Objective 4: Minimize the operating cost of moving vehicles. This is currently expressed by a simple flat cost incurred by each segment a vehicle travels. Each size vehicle could have a different cost per segment traversed, such that a vehicle with a higher capacity would presumably have a greater cost per time unit. Currently the system deducts no

cost for vehicles idling at stations, but we could insert this term easily enough if called for.

The objective function is subject to the following constraints:

Conservation of passengers and vehicles moving between nodes. Passengers and vehicles should be neither created or destroyed during the course of the schedule. The MIP schedule optimization follows a classic inventory management problem formulation.

Passenger movement between nodes is constrained by the capacity provided by vehicle movements between nodes. Passengers can only move around in the network when carried by vehicles. The optimization problem currently allows passengers to wait and transfer freely between vehicles at station nodes.

Vehicle movement about the transit network is constrained by several factors:

- Connectivity matrices allow certain types of vehicles to travel only between connected nodes. This allows us to connect nodes together with one or more modes of transit. For example, a certain subset of nodes could be served by a rail system, while the rest of the nodes would only be accessible via bus service. The connectivity matrix provides enough flexibility to model a transit system as a collection of directed graphs, so stations could be connected by one way or bi-directional links.
- Station and waypoint capacity constraints could prevent too many vehicles from visiting the same station or route simultaneously.

- A hard maximum fleet size might prevent some unrealistic solutions.

We could add some arbitrary constraints somewhat easily. For instance, we could limit the maximum number of vehicles on a group of segments or waypoints that have been grouped together to block each other on a constrained resource, such a bottle-necked intersection. This concept is discussed in more detail in Section **??**.

This optimization make no attempt to handle meeting passenger required times of arrival (RTA). It only optimizes based on the reported time passengers first become available to depart. Oftentimes people would want to arrive at their destination just before the fixed start of their work day, or at an airport in time to catch a flight. Because this optimizer uses an inventory management approach, adding this information would result in an exponential increase in decision variables. This added complexity would make the schedule take much longer to generate. Combined with the fact that several of the passengers would not even make use of this functionality (such as the ones who are *leaving* work and just want to get to their destination as soon as possible), the schedule optimizer declines to consider this constraint.

To add an RTA handling feature, an algorithm external to this schedule optimizer would need to provide a rough estimate of the required time of departure (RTD) necessary to meet a passenger's RTA, and submit a transit request with that RTD into the optimization. If the itinerary provided back to the passenger falls behind (or too far ahead) of their desired arrival time, the algorithm could redact the transit request and try again with a slightly adjusted RTD. A few cycles of this incremental optimization on a much simpler schedule

optimizer for the subset of passengers who actually need it should provide an acceptable solution more quickly than if the scheduling problem was reformulated to include variables and objectives to take everyone's RTAs into account.

4.2.3 Transit Use Case Diagram

A passenger begins by submitting a transit request for sometime in the future to the global scheduler. The scheduler collects requests and generates an optimized vehicle schedule that separates passengers into several pools based on their current and final destination node. When the time comes to act upon the schedule, a station master at each station loads waiting passengers into the proper vehicle, and then sends the vehicles on their way towards their next destination. When the passenger reaches their final destination, they exit the transit system.

For simplicity, this simulation unboards all passengers at each station so the station master can re-sort them into their next/final destination pools. Another logistics layer could be implemented to provide the convenience of maximizing the number of passengers that could stay aboard their vehicles during stops at transfer stations.

4.3 Mass Transit System Structure

The model is arranged in a hierarchy allowing the partitioning and relocation of units at different levels of the structure as described in section **??**. This allows us to use flexible recursive algorithms to facilitate a lot of searching and reporting tasks. For example, we

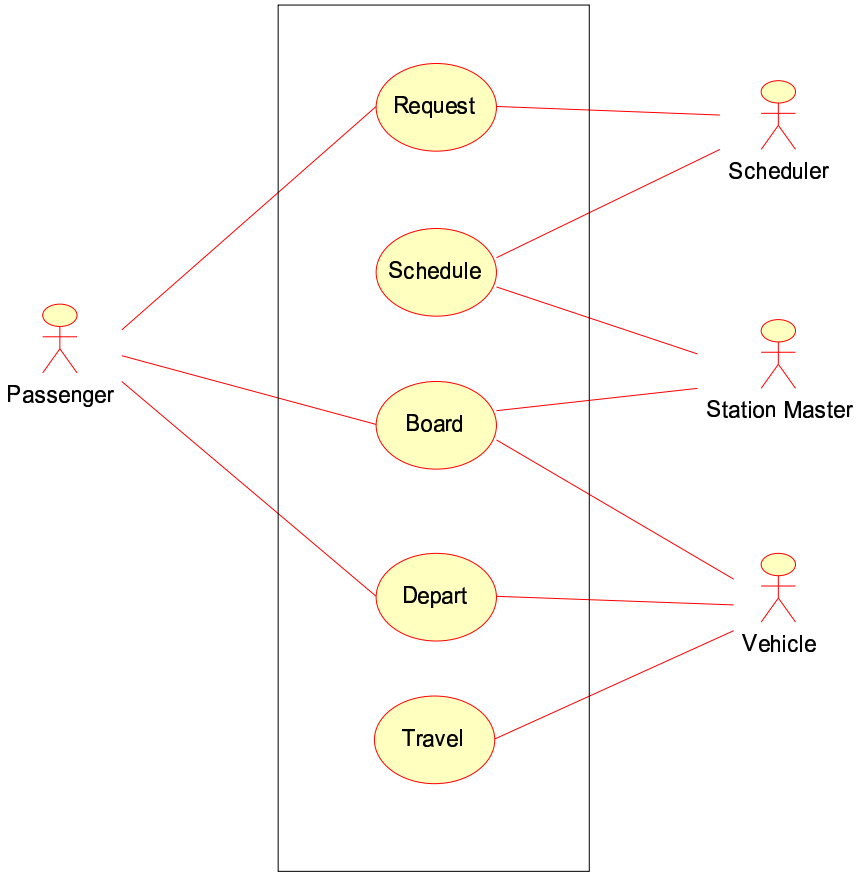


Figure 4.1: Transit use case diagram

recursively walk the tree to produce the incremental state snapshots of the system hierarchy in graphML format for viewing in yFiles's yEd application.

4.3.1 Generic Cell Class

All simulation entities inherit from the Cell class, which provides a sub cell container for any children classes. The cell class stores a handle to its own parent cell as well, so algorithms may traverse the tree in either direction. Subroutines allow child cells to move about the tree, updating associations so cells never have more than one parent. Each cell also has a className to distinguish between different types of children. They also inherit filtering functions that can search for and return a list of child cells meeting certain criteria.

4.3.2 Neighborhood Nodes

All of the elements in the model are comprised of various incarnations of the generic cell class. A master city cell forms the root of the tree hierarchy and contains several neighborhood node cells representing clusters of employers and residences that share a transit station. This hierarchy represents a realization of the general class template described in Figure 2.6. Each neighborhood can contain any number of employers or residences that share the same transit station in a pattern reminiscent of J.H Crawford's reference districts.[16]

An employer would have a number of job vacancies associated with a particular job code (representing the particular skill required of a worker) and additionally a work schedule that would dictate the employee's commute schedule. Assuming that each vacancy could

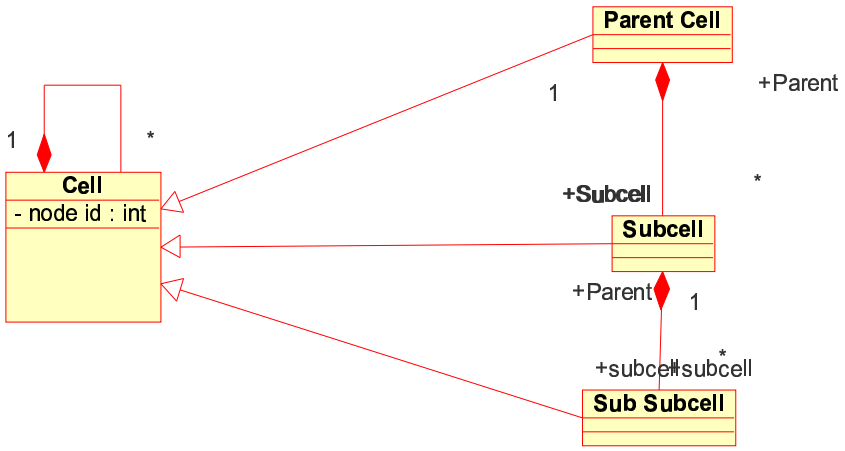


Figure 4.2: Cell Class Diagram

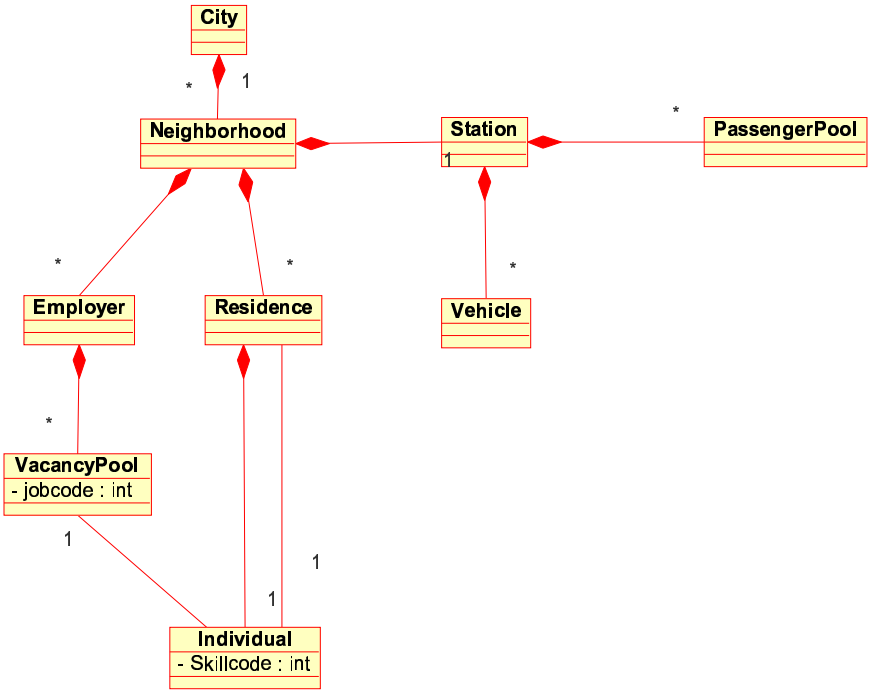


Figure 4.3: Neighborhood Class Structure

draw a qualified employee into the metropolitan area, an individual would be instantiated to fill that job vacancy and proceed to look for a residence somewhere in the city.

Since we're not interested in modeling real estate trends, the individual simply creates a new residence cell in any neighborhood of the city. Currently we use a simple uniform random distribution to allocate residences, but we could use different distributions to study other urban design factors, as suggested in Section **??**.

This job code and skill code accounting allows us to model the impact of specialized job centers and mixed populations in the urban area caused by development initiatives and zoning policies. The work schedules allow us to control and adjust the demand on the transit network in order to create loads and investigate peak congestion effects.

4.3.3 Transit Network

The transit network operates within the same cell hierarchy as the rest of the model. However, it behaves somewhat independently, serving to pick up passenger cells at **station** nodes and transport them to other station nodes through one or more layers of **vehicle** containers.

4.3.3.1 Stations

Each neighborhood contains one station cell that corresponds to a node in the transit network. As illustrated in Figure 4.4, all passengers transferring through a station are sorted into PassengerPool containers, one for each other station node in the network. While

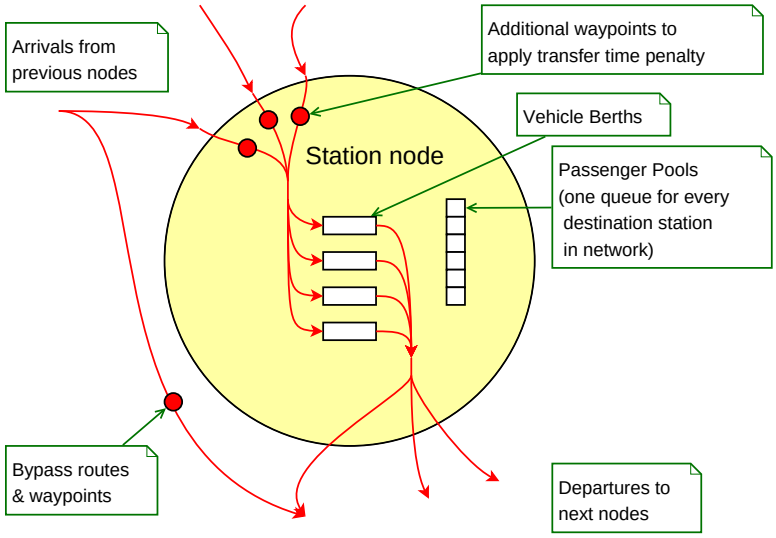


Figure 4.4: Station Conceptual Model

every passenger in a PassengerPool has the same final destination, they might take separate vehicles or even entirely different paths to get there. Additionally stations have a fixed number of vehicle berths that serve to constrain the maximum number of vehicles that can dock simultaneously. Figure 4.4 depicts a conceptual model of what a station node might look like for the purposes of our scenarios.

Station pairs are connected to each other via arcs defined in a connectivity matrix. Each type of vehicle has its own connectivity matrix, so different modes of transit can serve subsets of stations.

4.3.3.2 Waypoints

Typically, waypoints are sets of coordinates that identify a point in physical space. To make the modeling language a little more realistic, we added "non-station" waypoints on arcs between stations. Waypoints allow for the modeling of systems where stations separated by

varying distances (albeit in terms of integer units of time). Waypoints also allow passengers and vehicles to have a defined state while en route between stations while only moderately increasing the number of variables added to the optimization problem.

In the context of this project, waypoints act as one-way nodes in the transit network that allow the system to preserve the state of vehicles and passengers while traveling in-between stations. There are no constraints that prevent passengers from transferring between vehicles at the same waypoint, so to prevent passengers from train-hopping or plane-hopping en route, we apply an additional constraint that all the passengers and vehicles that enter a waypoint at one timestep must leave it the next timestep.

For some models, we might desire this kind of behavior. We could eliminate these waypoint constraints and allow vehicles to effectively enter a holding pattern at a waypoint just outside of a station. And there have been studies and patents filed to allow vehicles to transfer passengers at speed. In the future we may want to ease those constraints somewhat to allow these other types of behaviors.

Waypoints don't really belong to any parent cell in any meaningful way, since they are an artifact of the separate transportation infrastructure overlay as depicted in Figure 2.6. Other entities in the simulation have little reason to interact with them. We typically attach all waypoints directly to the master city cell since they would typically exist between neighborhoods.

We use waypoints to serve two purposes. While they are primarily used to represent the time and space traversed by vehicles between stations, they can also represent only the

time spent waiting at a station. In this mode, they would account for time spent slowing down and docking at a station berth, loading and unloading passengers, and – to some extent – the walking time of passengers between platforms or gates while transferring within the station.

All of the scenarios in this work employ at least one waypoint between stations in order to provide a time savings advantage to a vehicle bypassing a station. We use this measure to study the possible performance benefits of having offline stations, which is a feature necessary for rail service with express routing or almost any PRT-like transit proposal.

4.3.3.3 Vehicles

The vehicles in the various transit fleets traverse the network picking up passengers from stations and dropping them off at the next station. A completely separate transit layer represents each type of vehicle, with their own connectivity matrix that defines the segments and waypoints that each set of vehicles can traverse. The schedule optimizer only cares about two properties for each vehicle type: the maximum passenger capacity and the cost per segment traversed.

4.3.3.4 TransitTokens

The station master issues TransitTokens to identify passengers and cargo within the transit system, using them to store a customer's final destination. TransitTokens provide a handle used to sort passengers at each transfer station. Additionally, they log the path

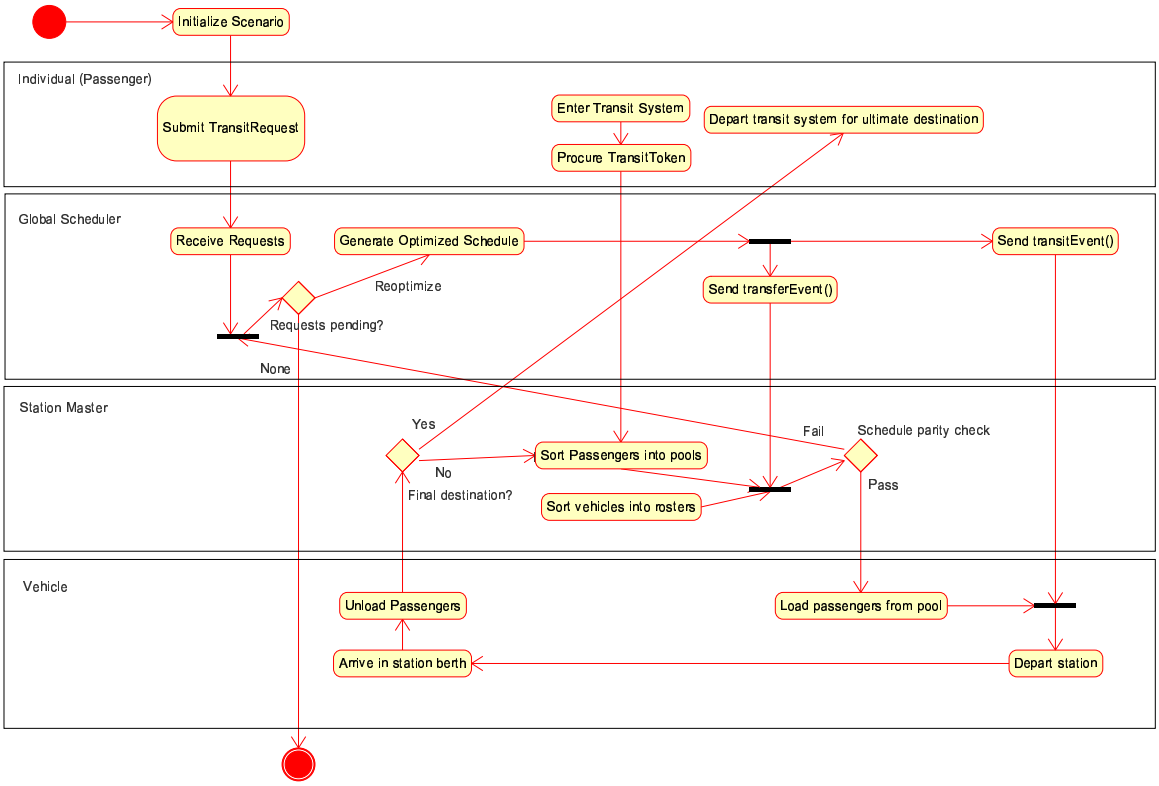


Figure 4.5: Transit Simulation Activity Diagram

taken and timestamps for each passenger, so they come in handy for collecting data on transit times and wait times during post processing analysis.

4.4 Mass Transit System Behavior

Behavior of individuals travelling on the mass transit system is defined by the sequence of activities shown in Figure 4.5. The four parties (or actors) that participate in transit behavior are: passenger, station manager, vehicle, and scheduler. Swim lanes show their roles and responsibilities in enabling behavior.

The simulation is based on a discrete event simulation engine. This means that state

changes in the system structure are triggered by the firing of events which occur along the global time line queue. The simulation engine runs a model by populating the global time queue with scheduled events and executing instructions triggered by events in order. The system global time advances to the time of the last event, and any state transitions triggered by that event are executed so they can perform their operations, sometimes scheduling additional events in the future event queue. Thus, the simulation perpetuates events and continues in time until the program reaches a time limit or there are no more events left on the simulation queue.

This transit simulation consists of a conglomeration of relatively simple entities working together. We'll introduce them roughly in order of increasing complexity.

4.4.1 Individual

The simulated people entities exist purely to create demand on the transit system. In the current simple commuting scenario, they simply live in a residence at one node and work at an employer at a possibly different node. They will enter the transit system based on their work schedule. Some configurable time in advance of their travel, they will submit a TransitRequest to the global transit scheduler system. By having advance knowledge of when the passenger needs to travel, the fleet schedule optimizer can ostensibly do a better job reducing passenger waiting time.

They enter the transit system by traveling to their local Station and procuring a TransitToken programmed with their final destination. From there on, they are shuffled

around by the other entities of the transit system until they reach their destination station. Once they arrive at their final stop, they are placed into the appropriate employer cell in that neighborhood.

4.4.2 Vehicle

Vehicles of the same type are considered completely interchangeable, so the only state information of any importance for them is their capacity and their next immediate destination node (either a station or a waypoint). Vehicles simply wait to receive a transitEvent and then pick up as many people as they can from the station's PassengerPools before leaving for their next destination.

When vehicles arrive at a station, they will dock in an available berth and immediately empty out all of their passengers into the station for sorting back into PassengerPools.

4.4.3 StationMaster

Each station has a StationMaster process that reads the global fleet schedule distributed with each transferEvent and organizes all passengers and vehicles. It first sorts all passengers into PassengerPool queues and all vehicles into rosters grouped by their final and next destinations, respectively. After a brief period of time allowing passengers to make their connections onto the next vehicle, the firing of the transitEvent signals that all transfers have completed and the vehicles disembark to their next destination.

4.4.4 GlobalScheduler

The global scheduler receives incoming passenger requests, occasionally triggering the generation of a new optimized schedule. Then it gradually advances the global clock until the time comes to serve the first passengers arriving at the station. The global scheduler begins firing a succession of transferEvents and transitEvents at regular intervals to synchronously push the Vehicles and StationMasters through their state actions.

4.4.5 Optimization Variables

The schedule generation MIP is formulated using the following sets of variables to track the inventory and movement of passengers and vehicles:

pp_{tik} : pool of passengers at node i at time t whose final destination is node k

ppi_{tik} : pool of new passengers initialized at node i at time t with destination node k

p_{tijk} : passengers from node i going to node j at time t with final destination k

ap_{tsi} : pool of vehicles of type s at node i at time t

$f_{tsij} \in$ integers : vehicles of type s from node i going to node j at time t

All variables are integers, but only the vehicle movement variables must explicitly be declared as integer.

Chapter 5

Analysis of Sample Transportation Scenarios

In this chapter we exercise the framework and mass transit system model by computing vehicle fleet performance for a variety of network configurations, combinations of loads, and vehicle fleet sizes.

5.1 Verification of Simulation Engine

Before the simulation environment can become fully operational, we need to verify that the underlying behaviors and conservation principles are strictly adhered to. For example, if a passenger declares the need to travel from locations A to B, we should verify that no matter what, the passenger will eventually arrive. Similarly, the number of passengers waiting at a station at any time needs to be consistent with flows of passengers arriving/leaving the station.

Several arbitrary transit networks such as the one shown in Figure 5.1 provides a variety of test cases for different combinations of connections between stations and waypoints used in system verification and validation. The graph demonstrates the functionality of both bidirectional station–station links, and different combinations of unidirectional station links connected via one or more waypoints. Additionally it provides multiple equal length routes linking several stations (such as n1–n2 and n0–n3) to encourage use of alternate pathways

during congestion. Several simulation runs with different demands and initial conditions provided test feedback during development.

The simulation leaves periodic dumps of the system state in graphML format that we can load into the yEd graph layout tool to visually inspect the object tree. These snapshots show us the locations of individual vehicles and passengers, along with records of their transit history. The entities are represented as nested UML objects, color coded by station. A portion of one of these zoomable diagrams is shown in Figure 5.2.

We could also attempt to model a regular 2D grid network. By applying several simplifications to allow our prototype fleet scheduler to scale up to a 25-node 2D triangular grid, we could model a PRT-like network. We simplified the model by restricting it to single passenger (or small party) vehicles, which eliminates transfers. The simulation will still unload and reload passengers as they traverse station nodes, but we can safely assume the passengers simply stay in the same vehicle they arrived on.

5.2 Simulation Requirements Verification

Throughout the test cases, we checked the following criteria to look for inconsistencies:

1. Passengers get sent to their destinations. Passengers that cannot receive service due to capacity or other constraints should remain at their point of origin (and not stranded in the middle of the transit network)

2. Connectivity constraints are not violated. Vehicles and passengers should only move between nodes that are connected on the connectivity matrix.

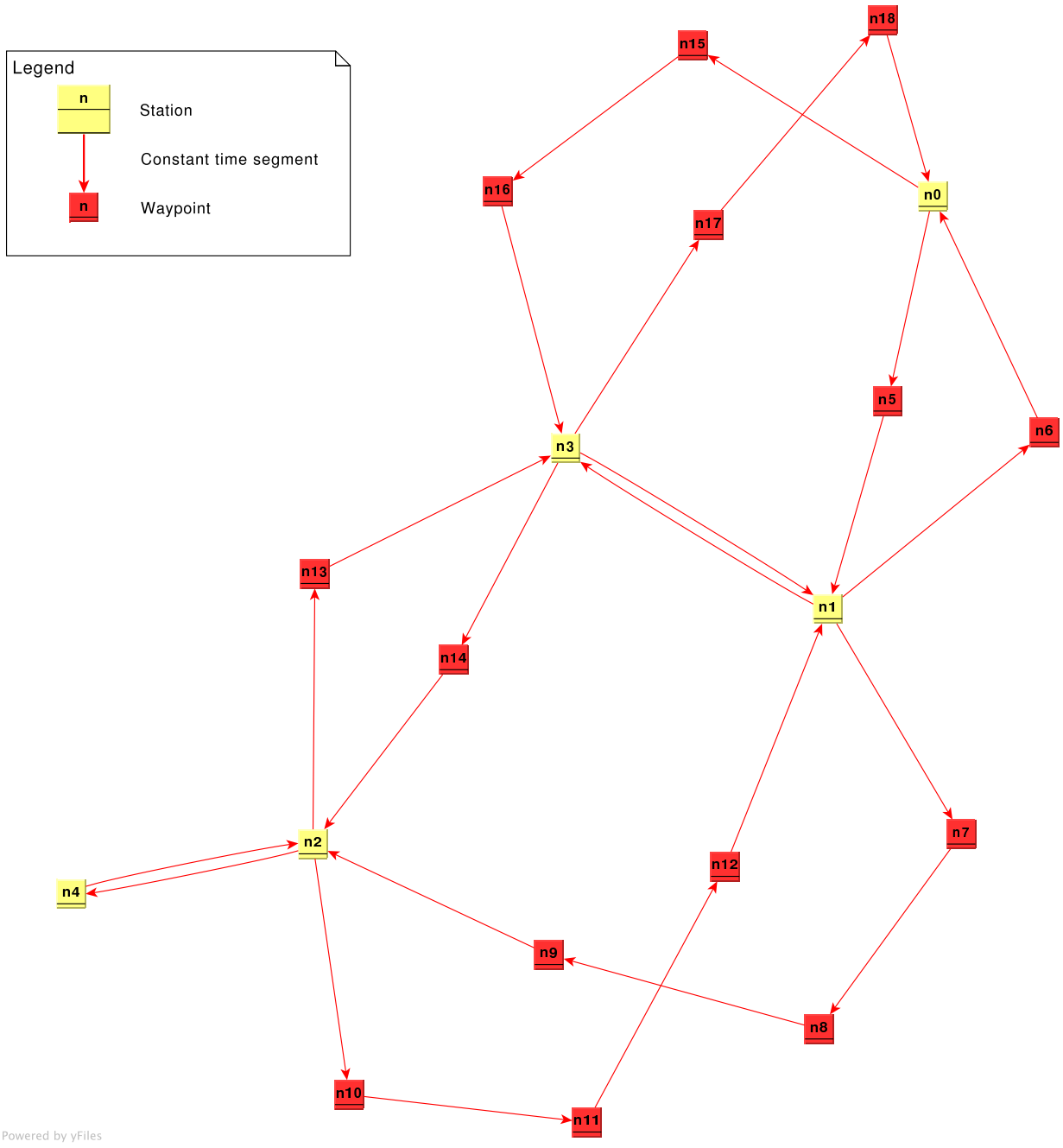


Figure 5.1: Arbitrary transit graph used for Verification and Validation. Yellow nodes indicate stations, red nodes indicate waypoints.

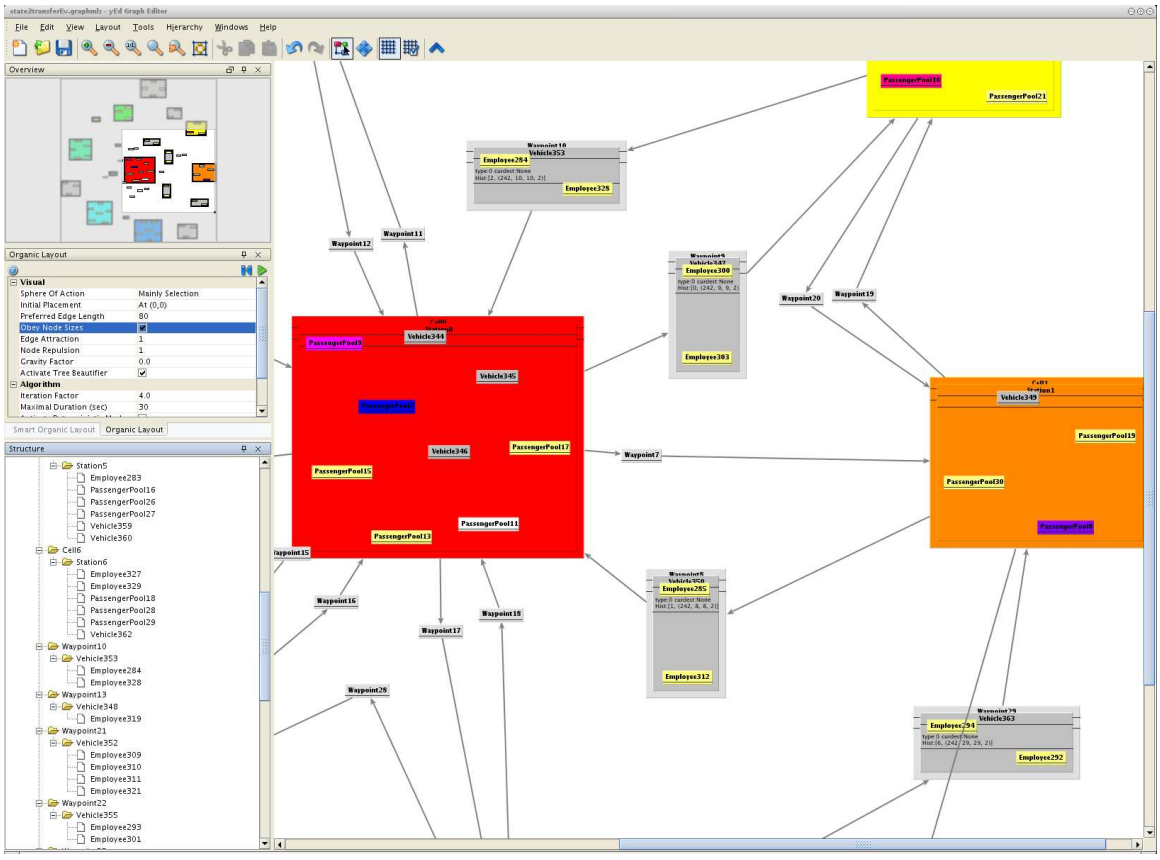
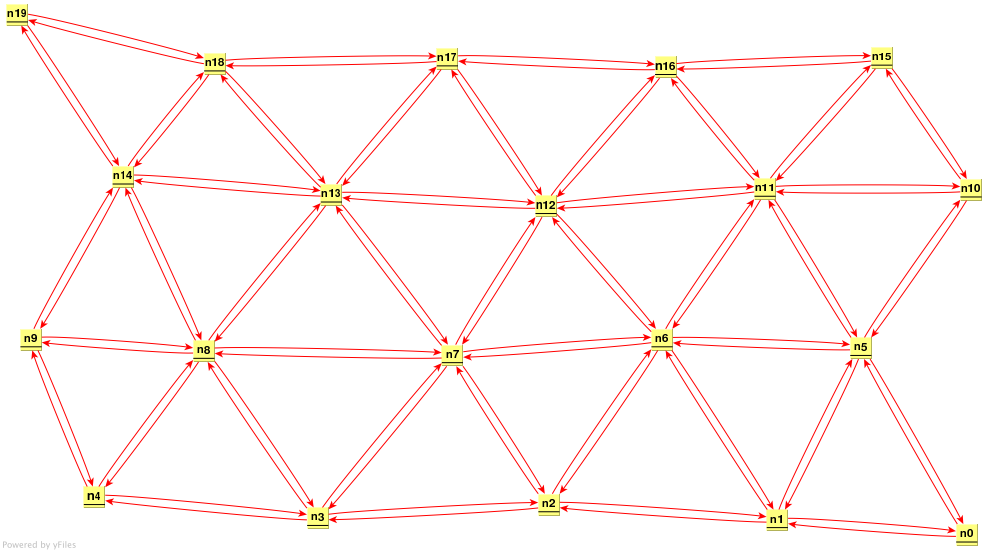


Figure 5.2: System Snapshot in yEd GraphML Viewer

3. Vehicle capacity constraints are not violated. Passengers should not be able to move independently between nodes without enough vehicles to carry them.

The simulation itself served as an invaluable verification tool for the schedule optimizer, as it was quick to point out inconsistencies in the number of passengers and vehicles available at stations while trying to follow the schedule. Tracing errors and crashes in the simulation helped debug the schedule optimizer's performance.



- width ymax = 4 station nodes
- length xmax = 5 station nodes

Figure 5.3: 20-node PRT Triangular Mesh Network

5.3 Validation of Analysis Data

During validation testing, desired behaviors and unwanted problems were found by taking some of the following measures:

1. Tracing the paths of individual passengers and vehicles to ensure they make sense.

- Vehicles shouldn't move on their own when they're empty, unless they do so to make way for vehicles carrying passengers.
- Passengers should travel on a reasonably direct path towards their final destination.

2. Checking for optimality by attempting to find improvements to or deficiencies in the schedule. This task should be hard, or even impossible if the solver had found an optimal solution.

- Scenarios with multiple vehicle sizes and a high enough load should show a "preference" for using larger, more economical vehicles, supplemented by a few small vehicles running around feeding the larger ones to full capacity.

5.4 Sample Scenarios

The simulation generates histograms plotting the transit system's response to input demand "pulses". The demand pulse is currently a one-time uniform random distribution across all source and destination nodes. The simulation generates demand by creating one employer cell in each neighborhood that employs a fixed number of commuters. The employees originate from any station node in the network with a uniform random probability. This arrangement for defining transit demand was chosen to reflect the fact that municipalities often have more control of where and how many jobs are created than where people choose to set up residences.

Scripts produce sets of results for two types of transit topologies meant to represent extremes that would combine to form complete transit network graphs: (1) a simple 1D linear light-rail system, and (2) a 2D hexagonal cluster unit on a triangular grid. We simplify our analysis by limiting the cluster size we consider to only 7 nodes, near the practical limit for computing quasi-optimal solutions to TSP-type problems. We arrange the stations in

a simple line, or clustered into a fully-connected star topology with a central hub and six points.

5.4.1 1D Light Rail Transit Network

The light rail system has two types of operation: a sequentially-routed "linear" rail network where trains must stop at every station, and an "express" rail network where stations are off the main line and trains can save time by bypassing stops. Note that the additional routes made available indicate temporal paths and not separate physical paths between stations.

5.4.2 2D Hexagonal Network

In order to scale up our schedule optimization to tackle larger, more practical transit networks, we'll have to start by concentrating our efforts on a simple triangular grid organized into hexagonal units. Building upon this basic unit, we could eventually decompose larger problems hierarchically into clusters of hexagonal units and solve them in parallel, as described in Section **??**.

As in the 1D rail scenario, we again distinguish between a sequentially-routed network (depicted by Figure 5.6) where every vehicle must stop for transfers at every station node it passes along its way, and a network with express routes (depicted by Figure 5.6) in which vehicles could bypass stations to save time and berthing space in the stations passed.

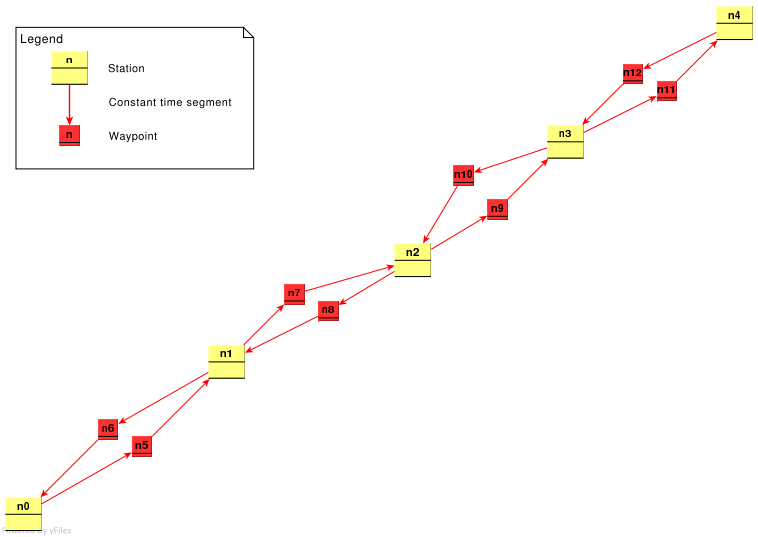


Figure 5.4: 1D Light-Rail System Connecting Four Neighborhoods‘

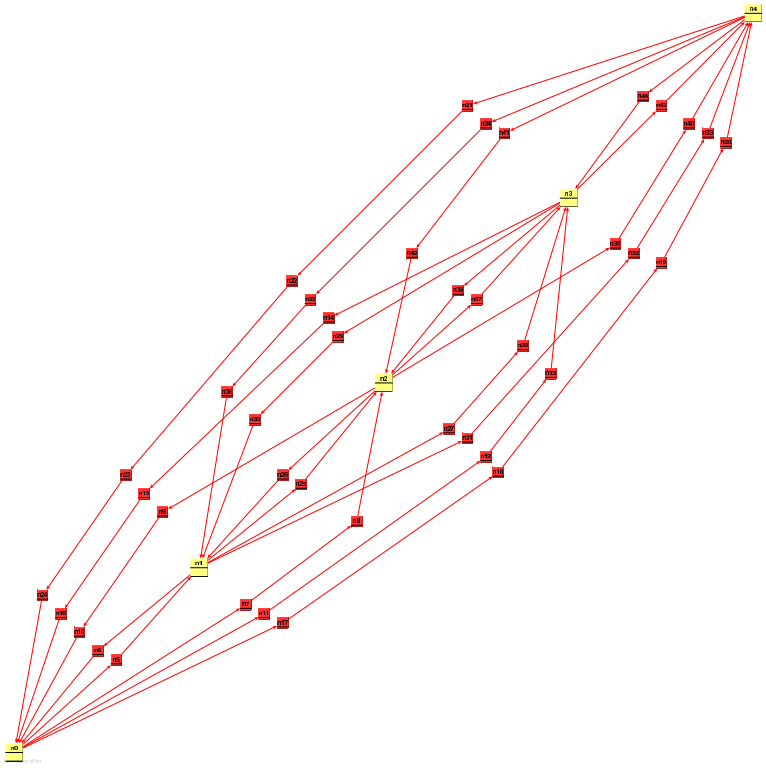


Figure 5.5: 1D Light-Rail System Connecting Four Neighborhoods, but with Local and Express Bypass Routes

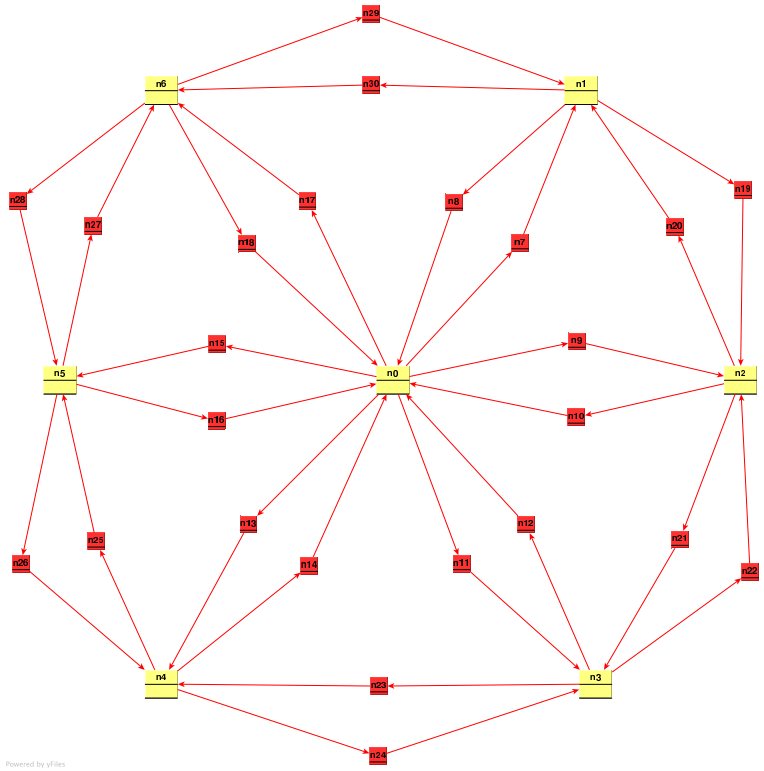


Figure 5.6: 2D Hexagonal Cluster

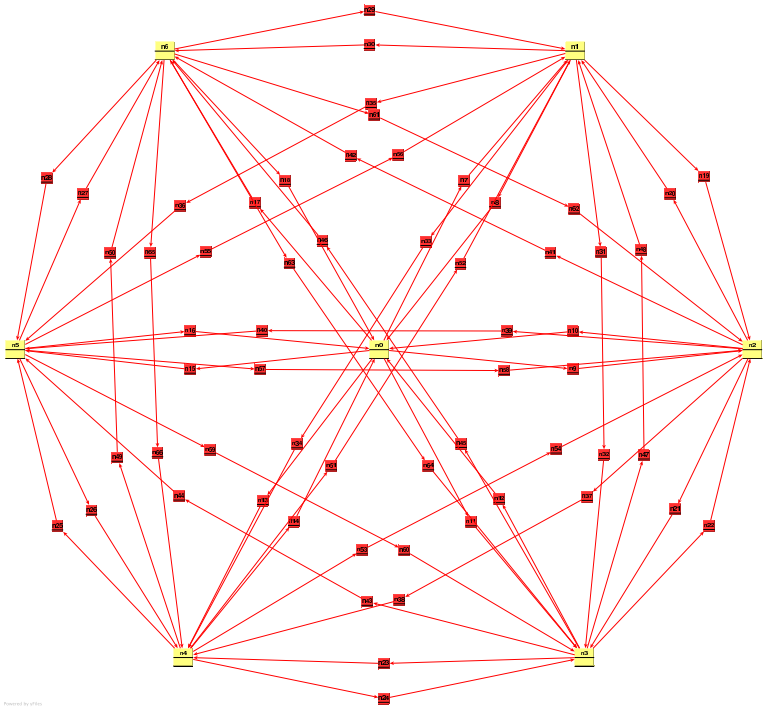


Figure 5.7: 3D Hexagonal Cluster with Express Bypass Routes

5.4.3 Assumptions and Limitations

The simulation currently does not specify all of the initial conditions to the schedule optimizer. In particular, the beginning locations of vehicles are currently left free. This gives the transit systems used in our scenarios an unfair advantage, as vehicles can initially be "magically" created anywhere in the network. The total number of vehicles in the fleet is also left unbounded. A more realistic study would consider setting the maximum number of vehicles in the fleet as a design parameter to explore.

To take away some of this advantage, in our scenarios we currently set the final vehicle state to equal the initial vehicle state. This means that each schedule that meets a certain demand pattern is at least repeatable for another run of the exact same demand pattern at the end of its active time interval. It also forces the vehicles to spend some energy returning to their initial state. Since the scheduler considers all vehicles of the same type as interchangeable, this repeatability constraint does not force all vehicles to make complete round-trips. They might operate in rotation with other vehicles of the same type.

As noted in the scenario description at the beginning of Section 5.4, each scenario has a uniform distribution of employers and residents spread out among its nodes, making the problem highly (and uncannily) symmetric. Future studies could easily vary the distribution of demand to model a network with business and residential centers, as noted in Sections **??** and **??**.

Each segment between waypoints and stations represents 2 simulation time units. Typically, the last segment entering a station represents the time penalty for stopping at

that station. As seen in the network models, bypassing a station results in a 1 time segment gain. This means that we're assuming that the amount of time a vehicle spends slowing down, stopping, waiting for passenger transfers, and getting back up to speed again is roughly the same amount of time taken to travel between stations at speed. This ratio could be adjusted by adding additional segments to increase the visit penalty at stations, or to increase the travel time between stations. For these scenarios we sill stick to a ratio of $1:1$, which seems reasonable for several forms of transit.

We impose a maximum number of time units allowed by a schedule. Having more time units translates to a higher possible capacity over that time. On the 1D linear rail networks, we limit schedules to 18 time segments, which works out to 1.5 times the diameter of the sequential network ($6 * 2$ segments between the 7 stations). On the 2D hexagonal networks, we chose a limit of 12 time segments, which works out to 3 times the diameter of the sequential network ($2 * 2$ segments between an outer station to any other outer station).

Due to the high computing time required to exhaustively search the entire schedule optimization solution space, we impose a time limit on the wall clock time that the computer is allowed to spend finding a solution. When this time limit is reached, the MIP solver will simply return the current best sub-optimal solution it has found. This sub-optimal solution will typically fall close to the final optimal solution, so we could argue that the diminishing improvements to the schedule would not be worth the additional calculation time. In a real transit network this timeout would prevent vehicles from idling while waiting for further instructions from the global scheduler. While this tradeoff may be acceptable

for practical purposes, this optimizer timeout defeats the repeatability of our simulation for academic purposes. The CPU would stop computation at a nondeterministic point in the MIP optimization and could return different schedules depending on the speed of the CPU or even on how much time was allocated to other processes in the background. This is a limitation of the LP_Solve software's wall-clock-based timeout feature and could be fixed in the future by adding a deterministic timeout function that returns a sub-optimal result after a fixed number of solver iterations.

5.5 Scenario Performance Data and Histograms

SimPy probes monitor and collect performance metrics experienced by individual entities in the simulation. By plotting several of these metrics as histograms, we can gain some deeper insight into what the system is doing over time.

We present a set of example histograms in Figures 5.8 through 5.12. This data was taken from one of the 1D rail networks with linear sequential routing serving a moderate load of passengers.

5.5.1 Passenger Performance Metrics

Improving system performance perceived by passengers of the system often comes at the expense of vehicle fleet operating costs. Improving passenger service goals typically increases the number of vehicles operating and the distance they must travel during the course of a schedule. Since our optimization goal favors the fulfillment of passenger goals

first, we expect to see some fairly straightforward trends.

Histograms in red pertain to the transit system performance from the point of view of the passengers.

Transit Time: In Figure 5.8, we see the total number of passengers served by the system, and time they were delivered to their final station. Thus, we can view this graph as the system response to the demand impulse at time $t = 0$. Each time segment takes 2 units of time. Therefore the width of each histogram bin is 2 time units long, and we don't see our first arrivals until $t = 4$, since passengers must travel through 2 segments via a waypoint before arriving. Large groups of passengers continue to get dropped off at their destinations after roughly every other time segment, since all stations are separated by one waypoint. The quantity of passengers delivered diminishes with time, as fewer combinations of station pairs are that number of segments apart. We do notice that some deliveries are also made on the "odd" time steps in between the "even" peaks, where the scheduler has shuffled some of the overflow passengers to take advantage of unused station capacity.

Departure Latency: Figure 5.9 shows us the amount of time most passengers spend waiting at their station of departure. Most passengers can board vehicles right away, whereas a small number of passengers must wait for the next time step. This helps shed light on the "even" peaks and "odd" valleys we saw in the passenger transit time histogram of Figure 5.8.

Number of Stops / Transfers: Finally, Figure 5.10 shows us how many stops passengers

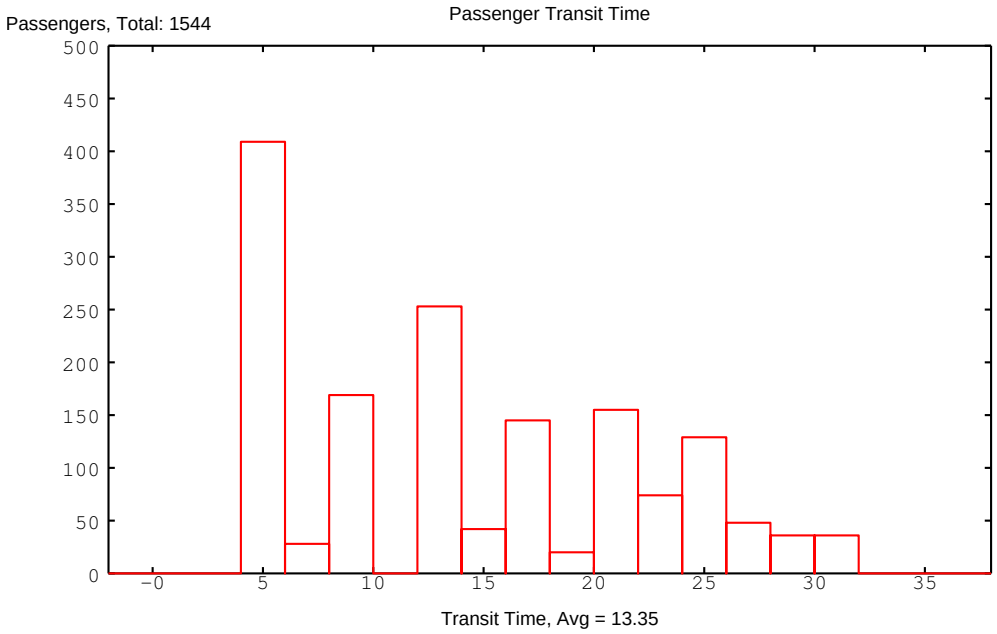


Figure 5.8: Passenger Transit Time

had to endure to get to their destinations. Since we only had 7 stations in our linear network that passengers must visit in sequence, we get a very predictable ramp down to 6 stops for passengers that traversed the entire line from one end to the other.

5.5.2 Vehicle Fleet Performance Metrics

Histograms in blue relate to vehicle fleet activity.

Segments Traveled: Figure 5.11 shows the system response of vehicles to the demand impulse. Again, we see a gradual ramp down, as fewer passengers remain in the system as time passes. In our scenarios, the second bin tends to have the most traffic. The first bin likely is lower because the initial location of the vehicles is free, so vehicles can essentially be created anywhere. Therefore, the difference in the first bin and the

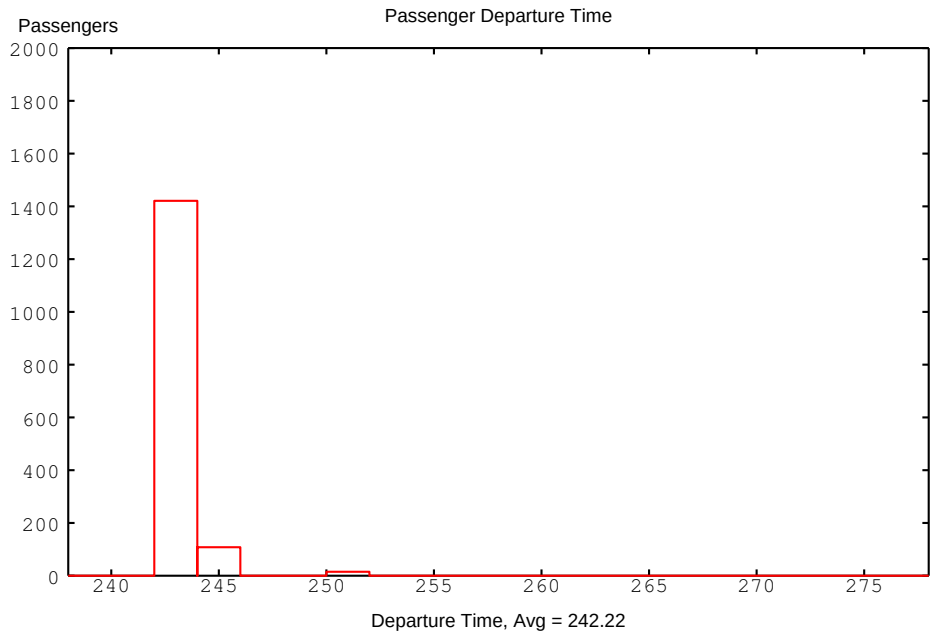


Figure 5.9: Passenger Departure Time

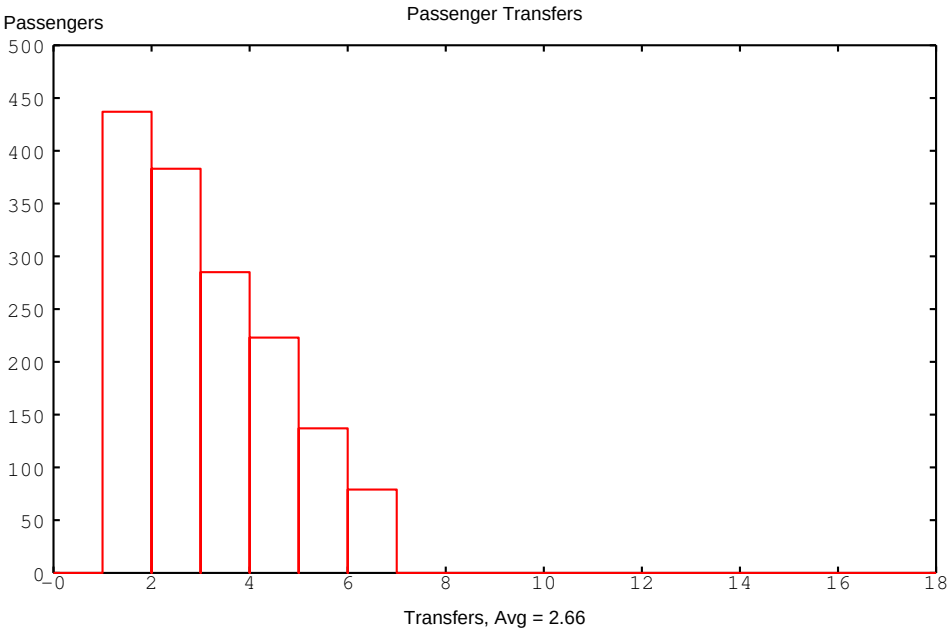


Figure 5.10: Passenger Stops / Transfers

second bin might be attributed to the advantage vehicles receive from the ability to magically appear exactly where they are needed for these scenarios.

A more difficult feature to explain is the valley right after the first peak of activity (and to some extent the less pronounced valleys following subsequent peaks). The system appears to operate in fits and starts, and it actually appears to rest between surges of activity. This is observed in all of the linear sequential routing scenarios. The optimization function attempts to deliver passengers as early as possible, so we would expect the system to push deliveries to the left and not waste time "resting". Perhaps the system is maxing out its capacity in a highly coordinated fashion during a delivery peak, and then uses the ebb of the rest period to do some "housekeeping", moving a small number of vehicles out of the way for the next coordinated surge. In this fashion, the system could deliver large numbers of people faster in two pulses with a gap between them as opposed to spreading the deliveries out evenly, especially if these coordinated pulses were necessary to allow passengers to make transfers.

Utilization per Time Step: Finally, we look at how full the vehicles were as they traveled their routes in Figure 5.12. Since minimizing vehicle travel is the lowest priority goal, the vast majority of vehicles operate nearly empty. In this scenario, the empty vehicles are likely moving out of their way of the ones with passengers. Also, some empty vehicles may move to restore the fleet's initial state at the end of the scenario, which offsets some of the advantage gained by their "magical" initial positioning discussed in Section 5.4.3.

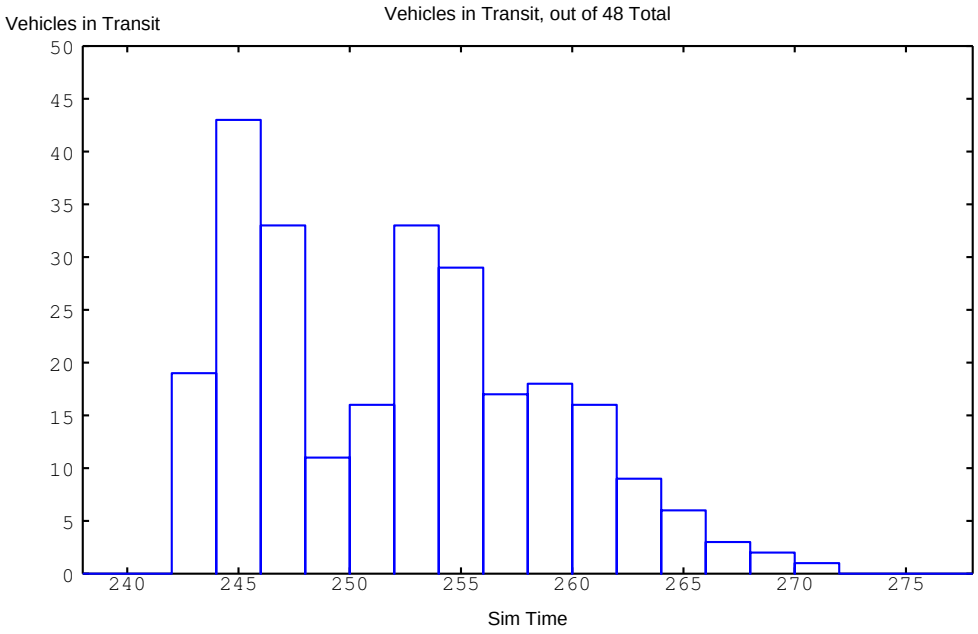


Figure 5.11: Vehicles In Transit

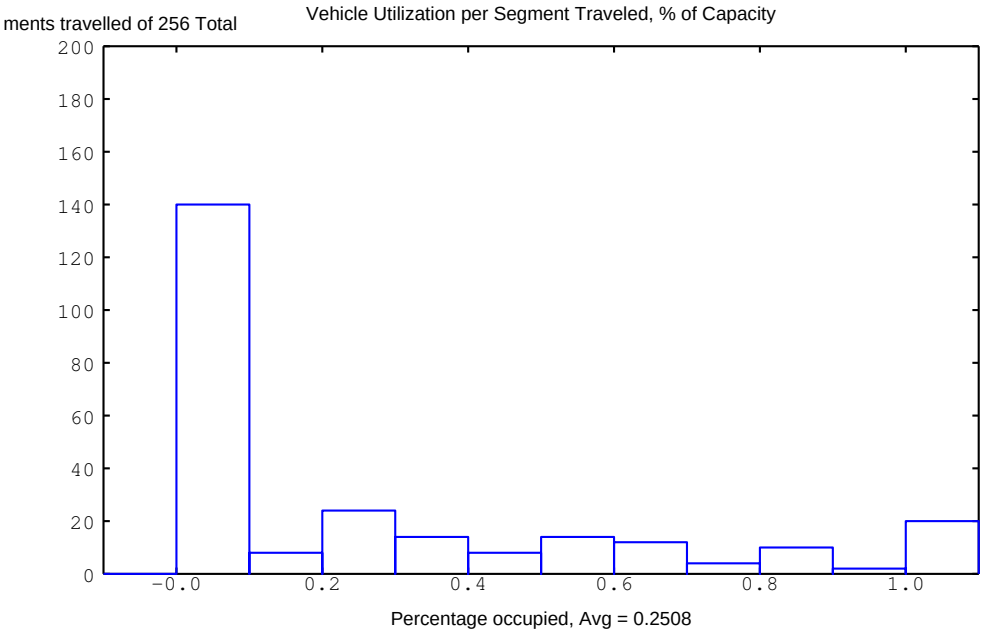


Figure 5.12: Vehicle Fleet Utilization

Gains in vehicle utilization and thus efficiency could be achieved by cutting back on the number of vehicles in the fleet and performing a new optimization. This would allow us to still emphasize passenger performance over vehicle efficiency. Otherwise, the scheduler would likely present passengers with unreasonable wait times and transfers in order to fill its vehicles up to capacity.

5.6 Parametric Analysis

To evaluate different transit paradigms, we'd want to reduce the relationship between system design variables and performance metrics into a series of parametric equations. We could then compare the projected levels of investment necessary for infrastructure and operations under competing transit paradigms to meet similar performance goals.

5.6.1 Design of Experiments

We set up a factorial experiment to determine the impact of several design factors. By individually varying each independent variable and observing its effects on our performance metrics, we can establish correlations between our design inputs and system outputs across a variety of conditions.

The scenario generation script populates a full factorial directory tree of 144 scenarios using the following input values for each parameter of interest:

Topology type: [1D rail, 1D rail w/ express service, 2D hexagon, 2D hexagon w/ express service] All topologies consist of 7 station nodes. The "express" routes indicate that

vehicles may bypass offline stations to gain a 1 time segment advantage over stopping.

Load [4, 64, 128, 256] Passenger demand per station. The total number of passengers on the network would work out to $\frac{6}{7}Load$, since $\frac{1}{7}$ of the employees would live in the same neighborhood where they worked.

Vehicle berths per station [2,4,8] Maximum vehicles that can visit and transfer passengers in a station simultaneously per time step. 2 could represent a typical rail station with cars running on 1-way tracks. 4 might represent a rail station with two tracks where cars at the platform could depart in either direction. Finally, 8 could represent a station with many individual, independent berths, such as a bus transfer station.

Vehicle capacity [16, 64, 128] Maximum number of passengers that can ride in each vehicle

A post processing script collects the following output metrics (condensed forms of the histogram data in Section 5.5) and summarizes them into rows on a spreadsheet:

- Total number of passengers handled
- Mean passenger transit time
- Mean passenger departure delay from time available to depart
- Mean passenger number of stops / transfers experienced
- Total number of each vehicle type utilized

- Total number of vehicle segments traveled, to give an estimate of operating costs
- Mean vehicle capacity utilization percentage, to summarize how 'full' the vehicles are running during each segment they travel. This metric gives some indication as to the system's operating efficiency)

Each row of input and output data on the summary spreadsheet represents a different demand level and transit network configuration. We can then use this spreadsheet to generate several families of plots that characterize the performance impact of a range of design input variables. Each of these plots include data regression curves useful for highlighting correlations and for creating an approximate parametric model of transit system design. By applying data filters to the spreadsheet, we can quickly create several plots that compare the different major design paradigms.

5.6.2 Comparison of Linear and Express Routing

We can use our factorial analysis to perform a quick comparison of design factors. In this case, we'd like to highlight some of the performance differences between a transit network with sequential routing and one built to accommodate express routes that can bypass intermediate stations.

Figures 5.13 and 5.14 plot both the linear sequential routing scenarios and the express routing scenarios on the same set of factorial sensitivity plots for two other design factors (vehicle capacity and station capacity, respectively). A regression line through the data indicates the correlation between the design factor and the performance metric of in-

terest averaged over all of the scenario conditions. A positive or negative slope indicates a correlation, a relatively horizontal regression line suggests that other factors dominate.

The top four charts in each set concern the impact of the design factors to passenger metrics, the bottom three charts affect the vehicle fleet performance.

The charts in Figure 5.13 show that vehicle capacity has little impact on passenger service for our optimized schedules. However, the type of routing used does have a profound impact on the average transit time and number of stops experienced by passengers. Allowing express routing to bypass stations can cut average transit time almost in half, while practically eliminating most stops and transfers.

The number of vehicles necessary to run a schedule and the amount of segments they run is also reduced. The reduction in segments comes about in large part because stops and transfers count as segments. While this may not be an accurate portrayal in terms of mileage saved, this could still represent savings of fuel and vehicle wear-and-tear, as braking and accelerating at a station can be every bit as taxing as cruising at a constant speed for many types of vehicles.

Figure 5.14 tells a similar story, though this time showing that increasing the number of vehicles that can dock at a station simultaneously can get passengers to their destinations more rapidly by easing congestion.

An additional curiosity shows that higher station berthing capacity can result in much a higher fleet size and operating costs in the linear sequential routing case which does not affect the express routing network much. This highlights some of the inefficiency of forcing

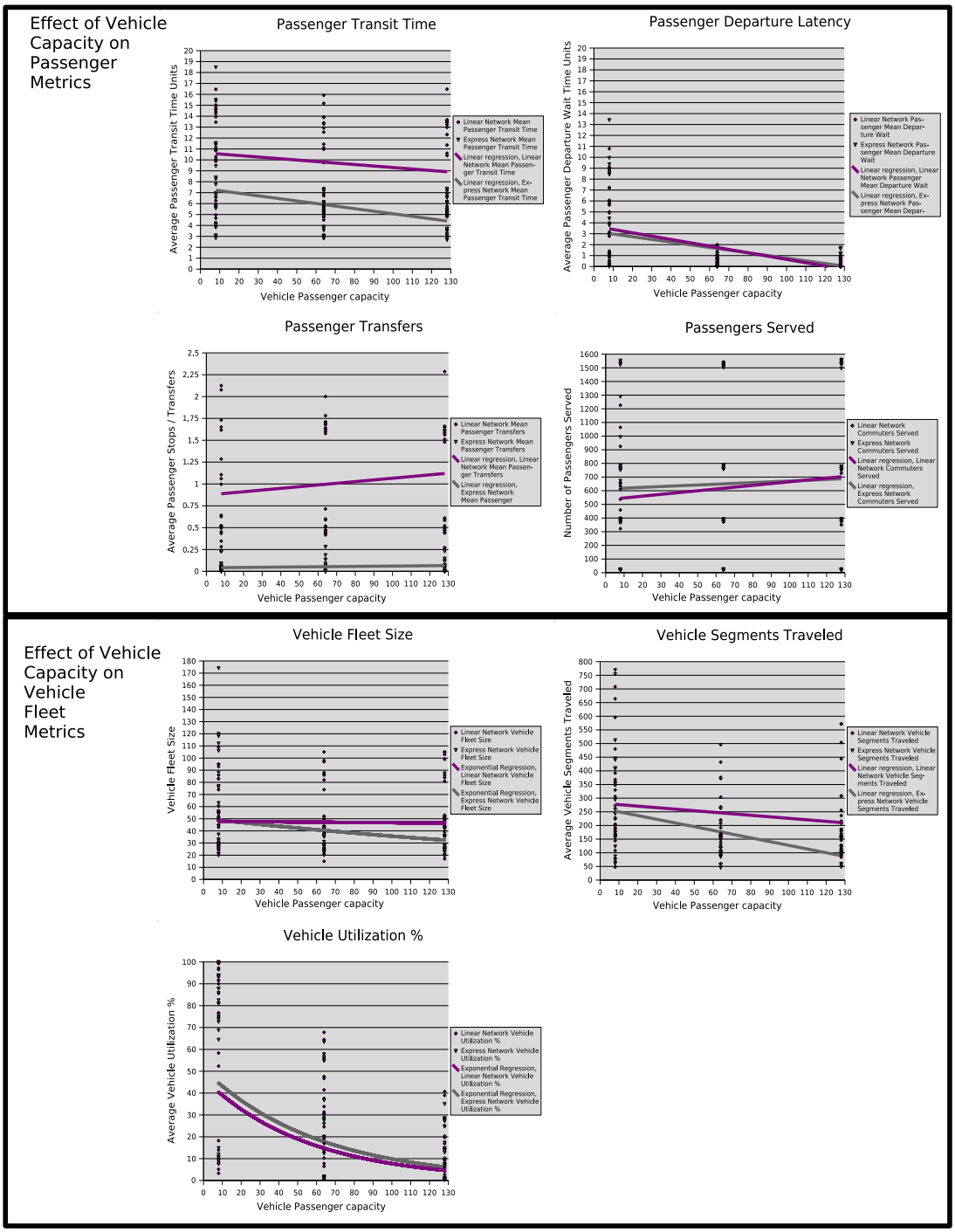


Figure 5.13: Effect of vehicle size on passenger and fleet performance metrics: a comparison of linear vs. express routing

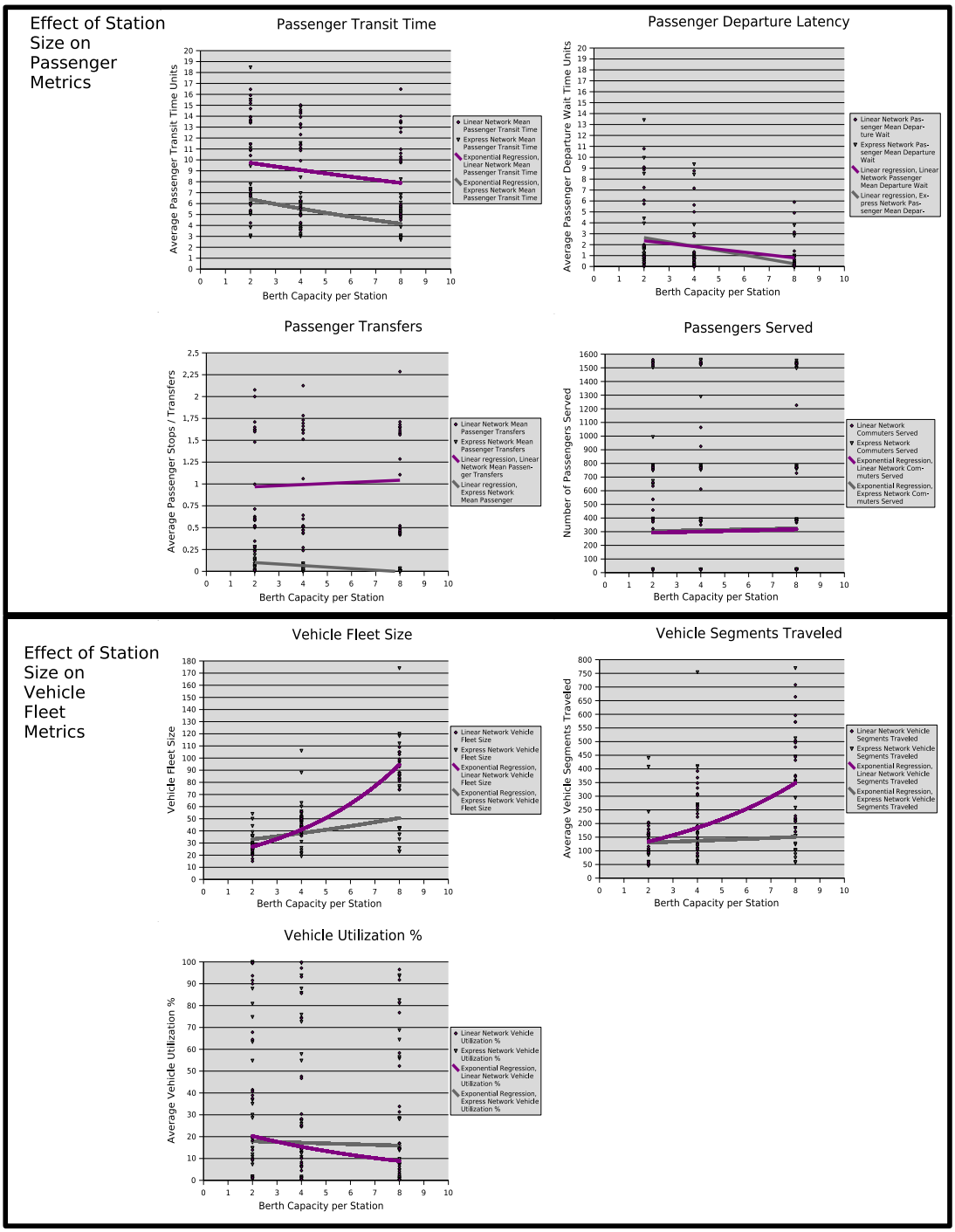


Figure 5.14: Effect of station size on passenger and fleet performance metrics: a comparison of linear vs. express routing

vehicles to stop at every station along a route, especially when there's extra capacity to handle these stopovers.

This design factor analysis supports notion that, all inputs being equal and subject to our assumptions, an intelligently scheduled mass transit system designed around a larger quantity of smaller vehicles running direct point-to-point routes between offline stations could provide much faster passenger service while meeting and exceeding the fleet operational efficiency of more conventional systems.

Chapter 6

Conclusions and Future Work

The first purpose of this chapter is to summarize the key findings of this work. Then, we make suggestions for directions for future work.

6.1 Conclusion

This thesis has identified the purpose and goals of an urban area, and attempted to define the characteristics of a mass transit system that could help meet the goals of diversity and specialization in a dense package. It has also applied intelligent routing in such a way as to allow comparisons of feeder and trunk transit networks with more futuristic PRT systems. The latter is accomplished through the use of a unified scheduling and simulation framework. Therefore, this work makes a step toward filling a gap between existing analyses of traditional transit featuring large, high-capacity vehicles, and smaller personal or group routed vehicles.

We hope that this work will ultimately inspire urban planners to invest intelligently in infrastructure to tackle 21^{st} century challenges, from reducing effects of peak oil and global climate change to creating equal access housing and environmental sustainability for a population. Often we try to look for solutions to the transportation problem by developing better vehicles, when we could address the problem more effectively with better urban design and development.

6.2 Future Work

Chapters 4 and 5 present work that is simply a first step in the development of a comprehensive platform for arcology simulation and optimization. Short-term opportunities for future work could focus on constructing additional studies for factorial analysis, and completing some missing usability features. Long-term development pivots on a major refactor of the schedule generation and interfaces to the simulation that would add detailed modeling features and increase the scalability of the transit network to useful dimensions. In more detail, specific opportunities for future work are as follows:

1. **Additional Design Factors.** We could easily take the opportunity to design new groups of experiments for factorial analysis on additional design parameters.

    1.1. **Spatial Load Distribution.** Rather than use a uniform random distribution of residences commuting to equal size employment centers scattered throughout the network, we could introduce ramp distributions that would give us separate employment and residence core sectors. We'd expect this asymmetry to reflect current city layouts, and reduce our vehicle utilization as empty vehicles must backfill those departing from residential stations.

    Furthermore, we could also make use of job codes and skill codes to model different distributions of employee or employer types throughout a population. For example, if the skill codes were to correlate to residents' socioeconomic status, we could set up a model to study the impact of efforts to integrate rich and poor

neighborhoods.

1.2. Temporal Load Distribution Passengers seldom appear at all source stations at exactly the same time. We could vary the shape of the demand input pulse to gradually inject passengers into stations over time.

1.3. Timeliness of Input Data. The amount of time in advance that a passenger submits a transit request to the global optimizer could affect how well the system can have a vehicle ready to pick that passenger up at their requested departure time. In the worst case scenario, the system does not find out about the passenger until the last moment, when they enter the station and buy a TransitToken to a specific destination. The more advance notice the global scheduler has, the better it can budget vehicles to reduce passenger wait times.

1.4. Multiple Vehicle Sizes. The simulation and optimization formulation are capable of handling multiple vehicle types, so it can provide solutions using a mixed fleet of small and large vehicles. The vehicles might even have different connectivity matrices. We'd need to adjust the factorial design generation and simulation monitor taps to allow data collection on multiple vehicle types so we can investigate what pairings of relative vehicles sizes work well.

2. Usability Enhancements. These additional user interface features would aid debugging and assist in further development.

2.1. Simulation Visualization Tool. We need a way to animate schedule results to help verify the simulation visually. Currently the only way to piece together passenger and vehicle movements is to compare schedule spreadsheets at each time step and to trace individual travel histories at the end of the simulation.

2.2. Scenario Editor GUI. In concert with developing an output visualization tool, we'd also want a visual tool for manipulating input data. We currently define scenarios programmatically.

2.3. LiveCD Packaging. The build and execution environment for this project is moderately involved with respect to software dependencies. To this end, we could create a bootable LiveCD based on KNOPPIX that could turn any compatible PC into a full development and compute node.[29]

3. Feature Completion. The core simulation engine and modeling language lacks some important features that prevent this project from achieving nominal functionality and modeling realism. Implementing the following features would go a long way towards applying this simulation to practical problems.

3.1. Schedule Optimization State Initialization. The simulation currently has limited support for initializing the state of a schedule optimization. As discussed in Section 5.4.3, we currently do not take the trouble to initialize the current positions of vehicles or existing passengers in the transit network. Adding this internal transfer of initial data constraints would allow us to update and re-optimize

a continuously evolving scenario, allowing us to achieve a rolling time horizon type of schedule optimization.

Without this enhancement, we can only run one schedule optimization per scenario with the assumption that the vehicles magically appear where they need to start, and no new information can affect the execution of the schedule once it begins to run.

3.2. Monte Carlo Analysis. We currently use a random seed to provide some entropy in our simulations and create randomized distributions for some input parameters. This means that to properly conduct an analysis, we'd want to run a scenario at least 20 times each with a varying random seed, and perform statistical data reduction on the observed output metrics.

3.3. Constraint Pooling. The system lacks a way to apply capacity constraints to groups of nodes or links, indicating that they share and block on the same physical resource. This would allow multiple routes to share common bottlenecks.

Without the ability to apply a single constraint to a group of nodes, each route linking pairs of nodes would either unintentionally add more capacity between station nodes, or would not allow one route to make full use of the unused capacity of other available routes sharing passage through an intended bottleneck.

3.4. Proper Handling of Long Distance Passengers. The objective function is weighted such that the number of passengers served (objective 1) takes priority over minimizing the number of segments traveled by all vehicles (objective 4).

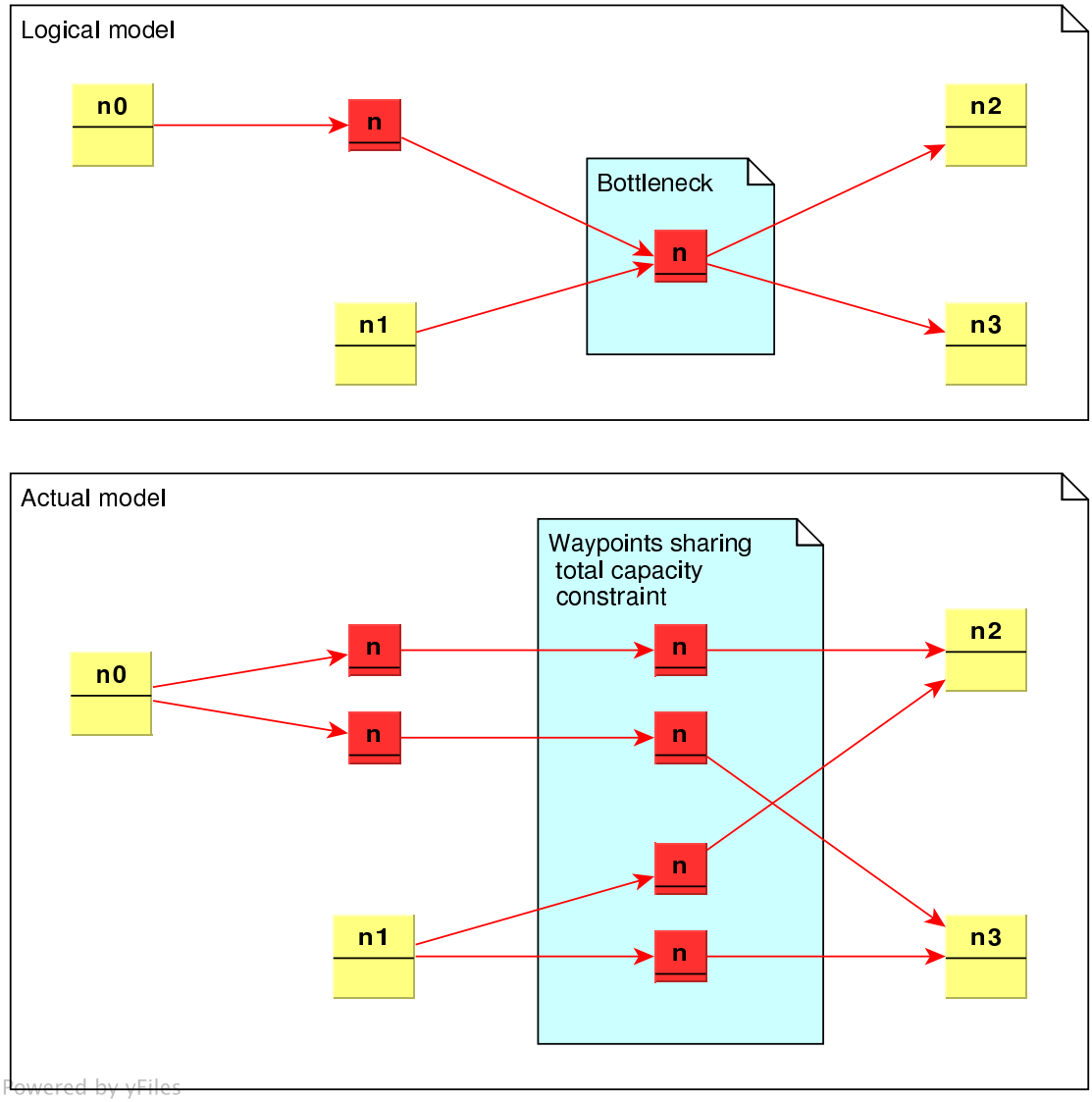


Figure 6.1: Constraint Pooling

In turn, objective 2 results in "compression" of the schedule to the left, in order to complete the schedule as early as possible. The compression is achieved by rewarding the system for sending passengers to their destinations at earlier times during the interval under consideration.

If optimization objectives 1 and 4 were of the same magnitude, we could better balance the conflicting goals between serving passengers at low-volume stations and keeping vehicles filled with paying passengers. The fleet optimizer could possibly even refuse to provide service to low-volume stations to increase total operating efficiency. However, this may also cause high volume routes to become unprofitable for cases where the cost of a long route exceeds the fixed reward for sending a passenger to their destination.

To remedy this, we'd need a more sophisticated reward system that would increase the fare value for long distance travelers. This would involve establishing another dimension to the set of passenger variables that would help track their starting point in addition to their final destination. However, this could easily increase the O^{3n} complexity of the optimization problem to O^{4n}. We could limit the impact of this additional term by grouping starting nodes together, so we'd end up with a zone-based pricing system. This would reduce the number of decision variables introduced into the MIP while still preserving the effect of variable fares.

Since this project's optimization formulation does not implement a zone-based fare system, the reward for objective 1 must always exceed the possible cost minimized

in objective 4, such that no passenger will get ignored for being unprofitable. To ensure this condition holds true, the passenger reward constant must always be greater than the cost of running a vehicle the diameter of the network by a comfortable margin. The next section addresses a means by which we could provide zone-based service.

4. **Scalability Enhancements.** Once we've fully defined the scope of the schedule optimizer, we can begin looking for ways to improve its performance. The following measures will allow us to harness the power of parallel processing to tackle larger problems in less time.

   4.1. Hierarchical Optimization. The current scheduler has problems finding solutions for more than 8 to 10 station nodes. One way to increase the scalability of our schedule optimizer would involve breaking up the problem hierarchically into smaller sets which we can solve in parallel. The structure of our Arcology model already supports this kind of organization, so we would only have to find a way to map the stations and segments of the transit network into this hierarchy. This scheme would work by grouping all of the stations into well interconnected clusters of no more than a handful of stations each. We'd need to choose an optimal cluster size that we could solve relatively quickly. E. Christian's HCPPT system employs a similar arrangement that optimizes across hexagonal clusters of 7 regions.[13] In our scheme, each cluster of stations will again be grouped into hexagonal super clusters. Splitting the problem space into this self-similar,

recursive hierarchy of cell clusters ensures that we apply TSP style optimizations to no more than a handful of nodes at a time to keep computational requirements within reason.

Optimization will start from the highest level first, and will determine how many vehicles would need to be exchanged between regional clusters. These regional fleet schedule numbers will feed down as boundary conditions to the next lower level in the hierarchy. Now we can solve each sub cluster in parallel, each of them reading the regional schedule in as constraints for their own local schedule optimization. In turn, these sub clusters would recursively split into smaller sub-sub clusters until the schedules they produce finally route vehicles directly between individual stations. This optimization scheme allows us to reduce the computation time for a network of n nodes from O^{3n} to $\frac{n}{m} * O^{3m}$, where the constant m represents the cluster size and $\frac{n}{m}$ is approximately the number of levels in the hierarchy. The cells at each level could be solved in parallel provided enough CPU resources were made available.

The catch is that the higher level regional scheduling takes precedence over local scheduling. On the bright side, this also nicely takes care of the zoned transit problem discussed in the previous Section **??**. However, we must use caution while rationing out station and vehicle capacity constraints so the higher level schedules leave some capacity to serve local passengers.

One possible self-similar recursive structure we could use to partition large prob-

lems is a modified heptree. Originally introduced by Bell in 1989,[7] it has several useful properties for our purposes. First of all, the cluster size $m = 7$ is near the maximum limit of conveniently computable TSPs. The hexagonal form has also been chosen for other mass transit optimization works, such as for local pick-up and delivery in Cristián Cortés' proposed HCPPT network.[13] Here, each vertex serves as a node. Each level of the hierarchy has a slightly different radial orientation relative to its immediate children, but this does not present a major problem for our purposes (compared to others who have considered using this structure).[28] We would overlay the entire structure top of a triangular grid created with waypoints, the segments of the grid representing possible transit network links. The distinguishing feature of our structure from conventional heptrees is the additional padding space that we can place between clusters, as depicted by the white space in Figures 6.2-b and 6.2-c.

This structure has some interesting properties that gives it some flexibility. First of all, not all nodes or sub clusters in the hierarchy need be fully populated or connected, so we can remove elements until we have a reasonable approximation of local conditions due to geographical or other constraints imposed on development. Furthermore, the padding between distance between clusters on the same level of the hierarchy is configurable. We have our clusters separated by 1 segment, but this could be uniformly be increased to provide more open space separating the clusters. The padded area might make good undeveloped area to turn into parks,

Counter-clockwise from top left:
(a) Cluster of 7 station nodes
(b) 2 level hierarchy of 49 station nodes
(c) 3 level hierarchy of 343 station nodes

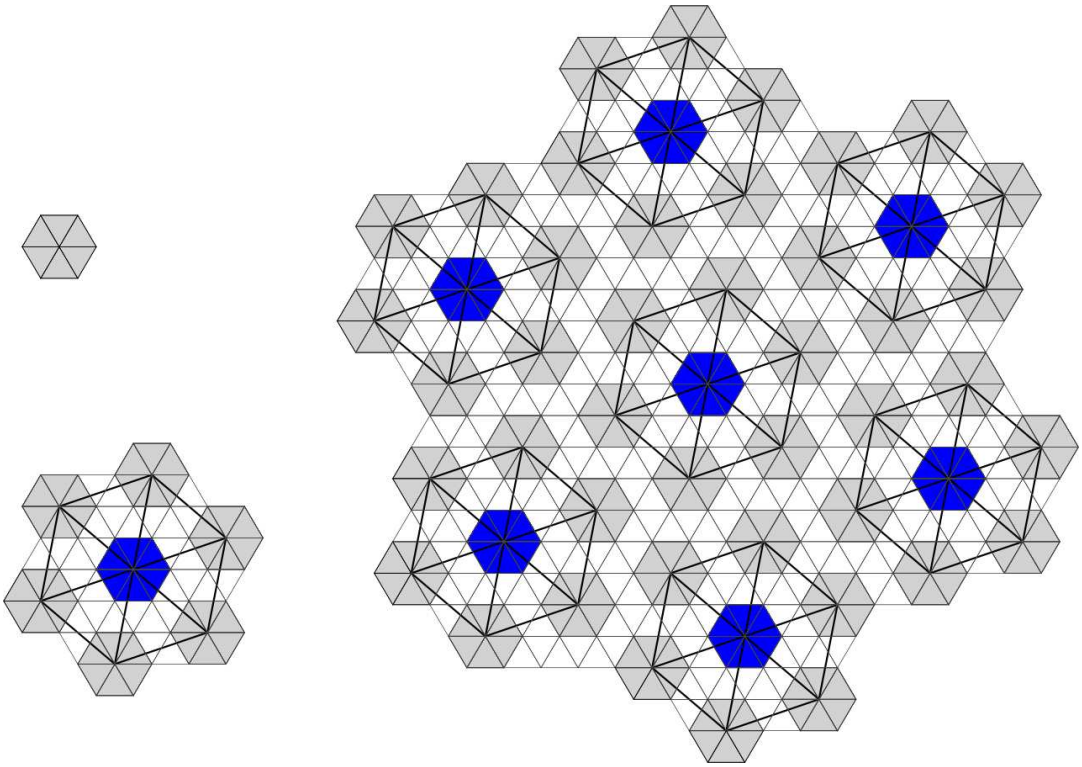


Figure 6.2: Recursive self-similar 2D space-filling Heptree Hierarchy

easements, and greenways. If all transit links crossing these padding areas were constrained to bridges or tunnels, pedestrians and migratory wildlife would have free passage across the land via virtually unbroken natural woodlands.

Clusters might be connected to each other in some combination of two different paradigms. We might enforce hub-and-spoke connectivity, where clusters are only joined by high capacity arterial lines (represented by the bold lines in Figure 6.2) linking their central hub nodes together. Alternatively, clusters might also be joined by the fine individual segments of the triangular mesh linking their

adjacent sub-clusters together. Traffic could filter through that lattice in a highly distributed fashion.

Whether the arborized or reticulated mode of connectivity dominates likely depends on the scale. Clusters at a neighborhood level would likely follow the hub-and-spoke model, channeling all traffic to a central mass transit station or a highway on-ramp, limiting through traffic through more restricted access. At a regional level, we'd want a more fully connected lattice so we could travel directly toward our destination without going out of our way to transfer at downtown hub stations. However, at the intraurban level, we'd again expect to see more arborization again as we'd funnel our access through consolidated airports or central high speed rail stations.

4.2. Optimization Heuristics. Finally, after we've ensured that we've achieved all of our desired functionality, we can look into other ways of optimizing computational time. The fleet schedule optimizer's work grows exponentially with the number of nodes, so LP_solve reached a scalability limit with merely 8-10 fully-connected stations. Beyond that, it takes more than 30 minutes for a 1.87Ghz AMD K7 PC to find any feasible suboptimal solution. Trying CPLEX instead of LP_solve might help here, especially if CPLEX can do better preprocessing to eliminate variables.

Breaking the problem down into smaller, more manageable chunks would give us our largest performance boost, but there are still other ways we could trim

decision variables out of the equation. By strategically pruning MIP branches that provide little performance benefits in comparison to the original algorithms, we could reduce the number of suboptimal dead ends that the optimization engine wastes time exploring. We would want to continue to optimize the algorithms until the scheduler is responsive enough to complete a recovery plan before the next time step. This would allow the system to react to service disruptions as soon as practicable.

Additionally, many of the decision variables in the formulation are zeros. If we can identify linked pools of variables that are consistently zero, we can prune them from the formulation completely. While a MIP solver's presolver should already do a good job of eliminating many of these extra rows from the problem, we could possibly cut down a significant fraction of its work and memory footprint.

5. New Modeling Features.

5.1. Holding Pattern Waypoints. Waypoints currently exist to give vehicles and passengers a state of existence while in transit in between stations. In order to prevent passengers and vehicles from mixing while they are grouped in the same waypoint bins, however, we must apply some additional constraints to effectively prevent mid air passenger transfers between vehicles. The constraints stipulate that all vehicles and passengers that enter a waypoint during one time step must continue on to the next waypoint in lockstep. All waypoints are constructed as part of one-way routes, so there is no possibility of capturing passengers en

passant.

To be fair, there have been proposals for improving transit efficiency by docking moving vehicles together and transferring passengers while in motion.[9] So it's comforting to know that we could model some of those scenarios by relaxing constraints.

However, sometimes we do want to allow vehicles to wait or enter "holding patterns" at waypoints, so they can create a buffer into another constrained resource, such as an airport runway or gate. This provides additional storage holding capacity outside of the station which can be put to use to increase network capacity. Mostly these buffers are used to help deal with uncertainty. Since our scenario schedules are deterministic, unperturbed by mechanical failures or passengers and vehicles turning up later than they're expected, we would gain little by allowing vehicles to hold at waypoints. Each holding waypoint would also double the number of decision variables needed to hold the new possible states as vehicles can decide to hold or proceed with their passenger load. Due to these factors, waypoint holds have been skipped at this time, but could add a useful modeling element later.

5.2. Continuous Time Model Definition. Currently a model must be expressed in synchronous integer time steps in order to work with the optimization formulation. We'd find it quite useful to define a modeling language that allows us to construct the model with constraints and distances expressed in terms of

continuous time.

We could then use algorithms to convert the continuous model into an integer time step model. This would likely introduce a lot of rounding and aliasing artifacts, the effects of which must be quantified and tracked. However, we'd gain the ability to run the same model at varying temporal granularity. This would allow us to study and set an optimal time step length that keeps these errors in check, instead of wasting time manually shoehorning physical models into approximate models using integer time quanta. This flexibility would greatly help address some of the issues introduced by the synchronous integer time step paradigm by allowing us to analyze the same scenario with different timing parameters, observe, and minimize the aliasing artifacts.

A continuous time modeling language would define capacity constraints in fractional units, such as vehicles per unit time. The ability to adjust the resolution of time steps allows us to detailed models with fine-grained time steps over an interval of interest, and solve coarse models much faster. The constraint values must scale with the time step length, such that double the capacity could pass through a constrained node in double the time.

Best of all, this continuous time model might allow us to more easily link together schedules of transit systems that have different paces of operation over different time intervals. For example, high frequency bus and rail transit could effectively serve low frequency but higher capacity airplanes or ships at port. This concept

provides a nice segue into the next section, discussing the challenge of coordinating schedules across modes of transit commanded by different fleet operators. Instead of a single organization asserting full sovereignty over all vehicles in the system, we'd want to provide a way to tie together disparate fleets by allowing operators to cooperate on forming a mutually optimal schedule.

5.3. Multiple Operator Collaboration. A global optimizer that could include multiple independent operator goals and scheduling constraints certainly isn't out of the realm of possibility. A central authority would maintain the global schedule formulation, and might allow participating members to add their own sets of constraint statements, or even weighted objective functions. This would allow individual fleet operators to synchronize schedules or even share coverage and balance loads with other fleets. Additionally, they can insert crew and maintenance schedules into the global scheduler to allow them to pick up and drop off drivers, pilots, and other staff at certain locations, or make sure that their vehicles end up in a certain maintenance bays every so often for refueling and service.

These inputs into the optimization problem can take the form of additional constraints. Constraints tend to help reduce the number of branch and bound paths that the optimizer needs to search to converge on a solution. As long as the solution remains feasible, expressing these constraints would be the job of the separate transit fleet organizations. The challenge comes in defining the abstract protocols needed to express these constraints, as they not only would require extensive

knowledge of how the global optimization problem is formulated and solved, but should also not be too explicit as to preclude changes and updates to the underlying formulation. It is undesirable to have this information format coupled too closely to the formulation, such as down to the level of variable naming conventions or specific MIP techniques for enforcing certain condition. This lock-in would make it more difficult to change and upgrade the optimization engine in the future. We wouldn't want to force every operator to radically change their code at the simultaneously every time system needs an incremental upgrade. We also wouldn't want a systems upgrade project to fail because of one or two late development efforts. We want enough abstraction built in so that participants might make changes at their own pace to take advantage of newly introduced scheduling and optimization features. The protocol specifying their constraints needs the ability to "compile" itself from an abstract to an explicit form so it can translate into both older and new versions of the optimization formulation.

Unfortunately, the design and specification of an abstraction language lies beyond the scope of this work, but we'll outline some of its major requirements. Such a language might allow the businesses to express what additional criteria a generic schedule optimizer must meet, without "cheating" and taking advantage of direct and specific knowledge of the optimization formulation and of its variables. A sophisticated abstraction language processor would have to take the expression and transform them into equations that relate particular variables to each other

and existing variables in the global problem formulation. This processor would be nontrivial to implement without a lot of prior experience, and even still could give way to unexpected errors and behaviors.

A simple, minimalist collaborative scheduler would allow input of operator-specific crew and vehicle maintenance schedule statements in the form of: "these vehicles must go off line at this particular station at a particular time step." This would imply that the operators determine their maintenance schedules separate from the globally optimized schedule, and then insert service stops as fixed constraints. These constraint inputs would be similar to the scheduling interface used to import legacy systems operating on fixed timetables. The end result of would not be as optimal compared to a system in which the global scheduler could directly search for solutions that also fulfilled operator schedule requirements, but at least they would start closer to an optimal solution. The iterations could proceed as follows:

1. The transit network operator provides the number and current locations of available vehicles at the beginning of the day
2. The global optimizer takes the customer demands and those initial conditions, and furnishes the schedule desired of that transit system.
3. The operators manually (or semi-heuristically) tweak the schedule to ensure that particular vehicles end up in nearby maintenance bays when they're due. These get fed back into the global optimization as constraints.
4. The global optimizer finds another solution taking these new constraints into

account, filling in any gaps created in the schedule and hopefully not straying too far from the original optimal objective function result.

This would let us converge on a schedule somewhat near the optimal one that takes maintenance factors into account without tying down the program formulation to a particular implementation of the optimizer. A more sophisticated collaborative optimization scheme with a more expressive abstraction language would be able to tell the global scheduler where its maintenance bays were located and how often each vehicle needs to visit them, allowing the global optimizer to include fleet maintenance requirements after a single pass of schedule generation.

We'd still need to exercise some discipline to keep the system stable. In a transit system run by a single operator, the entire problem formulated by one party and all the input data provided by passenger requests and vehicle locations add constraints in a consistent manner – the worst possible outcome caused by injecting faulty data might be infeasibility. However, by allowing third parties deeper control of objective functions and constraint statements, we're opening up the system to a host of potential pitfalls and vulnerabilities:

- Malformed or even malicious statements could make the problem intractable. There may be ways to identify some offending statements and automatically detect and flag them to somehow alert or even filter them out of the calculations. However, the latter approach could just as well create unpredictable results by orphaning variables (and thus leaving them unbounded).

- We'd need ownership and permissions on variables to separate the MIP components provided by different parties. This would ensure that operators don't introduce constraints that could penalize their competitors. This could be accomplished by prefixing variable names with namespaces and restricting access to allowed or "safe" variables.
- Operators could attempt to maliciously game the system by reserving excess capacity for themselves to lock out their competitors. This would be more difficult to do, since the global optimizer would balance the passenger load across all fleets.
- Many companies pride themselves on their own optimization capabilities. We may need a mechanism to protect proprietary information about their business strategies revealed in operator's contributed code. We might allow them to submit "black box" modules that manage to interact properly with the rest of the global optimization. An alternative method may be to partition the problem such that they're entirely responsible for optimizing their particular segment of the global calculation, interacting with the rest of the system through a layer of input and output variables.

Hopefully these considerations have helped articulate why the current incarnation of this thesis does not address these issues. At the same time, this highlights some of the exciting possibilities and new market opportunities made available by cooperatively linking disparate fleets into a cohesive end-to-end transit system.

5.4. Dealing with Unforeseen Events. Uncertainty is the antithesis to careful planning. Unfortunately, we live in an uncertain world, and need to make contingency plans for when things go wrong. How would the system deal with unexpected events that crop up during the execution of schedules?

While this thesis project makes no allowance for the possibility of its virtual vehicles getting held up by hail or high water, a practical system should factor in probabilities and fault trees into the schedule formulation. The optimizer might try to minimize is the impact of unfavorable events as an objective. Analysis of historical records can generate probability metrics associated with vehicle failures, route closures, inclement weather prediction, and so forth. A useful way of representing on-time vehicle performance probabilistically is to reconstruct the data from the cumulative distribution function (CDF) associated with the prediction as shown in Figure 6.3. This provides a much more useful picture than a simple mean and standard deviation, since most transit data is skewed towards the right. It's much easier to arrive late rather than early. We could quantize the CDF to reduce MIP formulation complexity, at the cost of adding longer, more conservative wait time buffers between connections. While complete arrival data should be monitored and collected, only vehicles arriving after the late arrival cutoff tail (indicated by the red dashed line) would impact the scheduler and trigger replan. The trick is to try to minimize the buffer time necessary to keep the schedule stable. Larger buffers would increase the time wasted waiting. We're primarily

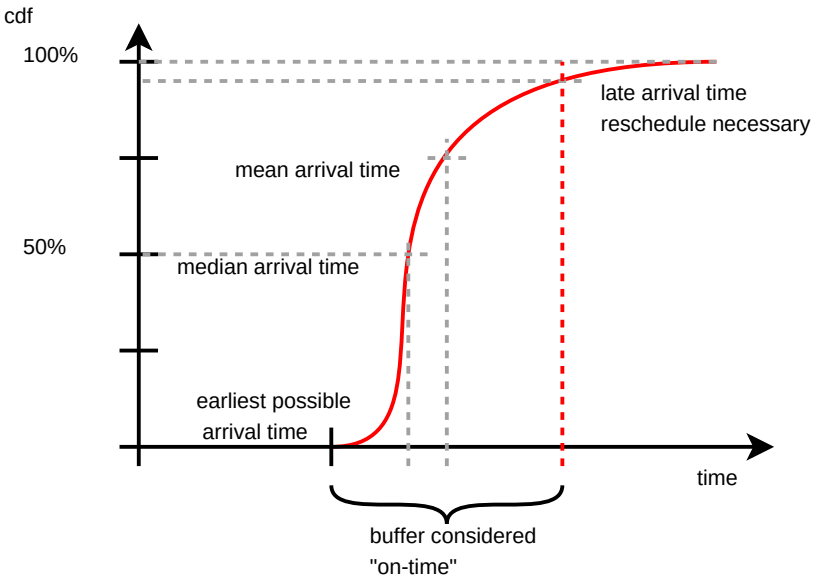


Figure 6.3: Notional CDF of vehicle arrivals vs. transit time, based on historical records

interested in what time the vast majority of the vehicles will arrive, as well as what hopefully small percentage of late vehicles cause schedule-impacting delays. We have no fixed "magic percentile" that would determine how much extra buffer time to schedule to ensure that mostly everyone makes their connections. This tolerance will likely get set arbitrarily at the beginning, as all of these factors contribute to an overall "schedule volatility" metric. With the optimizer system, we can recompute new schedules whenever an unexpected event occurs – such as when a vehicle is delayed enough to fall onto the tail end of the CDF and its passengers wind up missing their connections. The optimizer can take that emerging information into account and simply create a new schedule based on these new conditions – in this case it would likely divert other vehicles to pick up the stragglers stranded by the late vehicle. So the risk analysis that determines

how aggressively to schedule extra buffers into the system would depend on how much impact a schedule recovery plan would have. Planning in large buffers to reduce the likelihood of major delays means extra wait time for passengers and more idle time for vehicles in order to ensure that the schedule stays stable. The ability to drastically reduce these buffers means the whole system could run at a faster pace. If the cost of recovering from missed connections is low – suppose its a fixed train route that runs every 5 minutes – then the scheduler can comfortably deal with smaller buffers and higher schedule volatility. In the case of an airplane network where flights run between cities maybe once or twice a day, a missed connection would mean putting people up in hotels or chartering additional make-up flights. In this case, increased schedule awareness and dynamic optimization can help by figuring out the total system impact, and balance the cost of holding flights to allow latecomers to make their connections.

In light of this, we would want to introduce some practical optimization goals to the list of transit system performance goals discussed in section 3.1.1. This performance metric would characterize the system's stability in the face of unexpected breakdowns and delays, meaning it would help the scheduler intelligently create and maintain buffers to deal with uncertainty. How to actually formulate and compute this enhancement is beyond the scope of this thesis. The optimizer would likely need to do risk-impact assessments on every combination of missed connection to identify vehicles with critical path routes. But we do wish to en-

sure that the necessary on-time performance data is collected now, so that future generations of engineers could tackle this issue.

Bibliography

[1] *Arcology.com: Architecture + Ecology*. http://www.arcology.com/.

[2] Arcosanti: A prototype arcology. http://www.arcosanti.org/.

[3] Biosphere 2 biospherics. http://www.biospheres.com/.

[4] Commercial freight activity in the united states by mode of transportation. Technical report, Bureau of Transportation Statistics, Washington DC, 2002. http://www.bts.gov/publications/freight_in_america/html/table_01.html.

[5] Homes may be 'taken' for private projects. *The Associated Press*, June 23 2005. http://www.msnbc.msn.com/id/8331097/.

[6] Austin M.A. (with help from Baras J.). *An Introduction to Information-Centric Systems Engineering*. Tutorial F06, INCOSE, Toulouse, France, June 2004.

[7] S. B. M. Bell, B. M. Diaz, and F.C. Holroyd. The hor quadree: An optimal structure based on a non-square 4-shape. Technical report, Oxford, 1989.

[8] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. Lp-solve: Open source (mixed-integer) linear programming system. http://lpsolve.sourceforge.net/5.5/, 2007.

[9] Joseph Bittar, Frederick H. Barker, Anthony Cooney, David I. Perl, Richard E. Peruggi, and Michael D. Silverberg. Transferring freight or passenger cabs between moving bogies. Technical Report Patent number: 6038980, Otis Elevator, Feburary 1998.

[10] Hans Blumenfeld. *The Modern Metropolis: Its Origins, Growth, Characteristics, and Planning*. The M.I.T. Press, 1967.

[11] Daniel Carasso and David Carasso. Biosphere: Recreating the world. *Aish HaTorah*, September 17 2002. http://www.aish.com/societyWork/sciencenature/Biosphere_Recreating_the_World.asp.

[12] Takenaka Corporation. Takenaka's engineering: Sky city concept. http://www.takenaka.co.jp/takenaka_e/engi_e/c02/c02_1.html, 2000.

[13] Cristian Eduardo Cortes. *High Coverage Point to Point Transit (HCPPT): A New Design Concept and Simulation-Evaluation of Operational Schemes*. PhD thesis, University of California, Irvine, 2003.

[14] Christine Cosgrove. Roger rabbit unframed: Revisiting the gm conspiracy theory. *Institute of Transportation Studies*, 3(1), 2005. http://www.its.berkeley.edu/publications/ITSReviewonline/winter20042005/gm.html.

[15] J. H. Crawford. Carfree cities. http://www.carfree.com/, 1996-2005.

[16] J.H. Crawford. *Carfree Cities*. International Books, July 2000.

[17] George B. Dantzig and Thomas L. Saaty. *Compact City: A Plan for a Liveable Urban Environment*. W.H. Freeman and Company, 1973.

[18] Discovery Channel. *Millennium Tower*. http://dsc.discovery.com/convergence/eti/projects/towermain

[19] Discovery Channel. *Tokyo Sky City*. http://dsc.discovery.com/convergence/engineering/skycity/intera

[20] EPA. *Smart Growth*, 2007. http://www.epa.gov/smartgrowth/.

[21] Paul Hensgen et al. Umbrello uml modeller. http://uml.sourceforge.net/, 2007.

[22] Richard A. Etlin. The future of tysons corner: A fifteen-point blueprint for the new "downtown" of northern virgiina. University of Maryland School of Architecture, Planning, and Preservation, October 21 2004. http://www.smartgrowth.umd.edu/research/pdf/Tysonspdf.

[23] Foster and Partners. Millennium tower. http://www.fosterandpartners.com/internetsite/html/Project 1989-.

[24] Francis Frick. A seaside arcology for southern china. Master's thesis, University of Hong Kong, 2000. http://www.cityfarmer.org/frick.html.

[25] Georgi Gladyshev. *Thermodynamic Theory of the Evolution of Living Beings*. Nova Science Publishers, Commack, 1997.

[26] James Jeffords and Robert Smith. Surface and maritime transportation: Developing strategies for enhancing mobility: A national challenge. Technical Report GAO-02-775, US General Accounting Office, August 2002. http://www.gao.gov/cgi-bin/getrpt?GAO-02-775.

[27] Jill Jonnes. *Empires of Light : Edison, Tesla, Westinghouse, and the Race to Electrify the World*. Random House, August 19 2003.

[28] Julian Kardos. *Visualising Attribute and Spatial Uncertainty in Choropleth Maps using Hierarchical Spatial Data Models*. PhD thesis, University of Otago, Dunedin, NZ, December 2005.

[29] knoppix.net. Knoppix remastering howto. http://www.knoppix.net/wiki/Knoppix_Remastering_Howt 2007.

[30] Susan Metros. Creativity and leadership: A heart to heart on leadership: How to use your life experiences to become a better leader. *Association of College & Research Libraries News*, 66(6), June 2005. http://www.ala.org/ala/acrl/acrlpubs/crlnews/backissues2005/June05/hrttohrt.htm.

[31] SSF Research Network. Scalable simulation framework research network. http://www.ssfnet.org/, 2002.

[32] Michael Neuman. The compact city fallacy. *Journal of Planning Education and Research*, 25(1):11–26, 2005.

[33] Jarkko Niittymaki, Ari Karppinen, Jaakko Kukkonen, Pekko Ilvessalo, and Erkki Bjork. Citysim - validated assessment tool for simulating urban traffic and environmental impacts. In LJ Sucharov, editor, *Urban Transport V: Urban Transport and the Environment for the 21st Century*, pages 394–403. WIT Press, 2000.

[34] Fairfax County Department of Planning and Zoning. Tysons corner transportation and urban design study. http://www.co.fairfax.va.us/dpz/tysonscorner/, August 2005.

[35] Travis Oliphant. Scientific tools for python. http://scipy.org/, 2007.

[36] Laura Olsen, Cheryl Cort, and Roger Diedrich. Smart growth groups support vienna metro development. Technical report, Coalition for Smarter Growth, October 13 2004. http://www.smartergrowth.net/pressroom/PressReleases/2004.10.13.Viennametro.html.

[37] Laia Pages, R. Jayakrishnan, and Cristian E. Cortes. Real-time mass passenger transport network optimization problems. *Transportation Research Record*, 1964 / 2006(0361-1981):229–237, 2006. http://www.uctc.net/scripts/countdown.pl?748.pdf.

[38] Lisa Rein. Metrowest battle turns partisan. *The Washington Post*, page A10, July 28 2005. http://www.washingtonpost.com/wp-dyn/content/article/2005/07/27/AR2005072702295.html.

[39] Armin Rigo. Psyco. http://psyco.sourceforge.net/, 2007.

[40] Jeffery A. Schneider. Environmental investigations: Was the original "biosphere 2" project a failure? Technical report, SUNY Oswego, 2003. http://www.oswego.edu/ schneidr/CHE300/envinv/EnvInv01.html.

[41] Cliff Slater. General motors and the demise of streetcars. *Transportation Quarterly*, 51(3):45–66, 1997. http://www.hawaiireporter.com/storyPrint.aspx?7ece2dbe-c0ec-4590-ba1c-3d548bb8a758.

[42] Paulo Soleri. *Arcology: The City in the Image of Man*. Bridgewood Press, 1999.

[43] SimPy Developer Team. Simpy homepage. http://simpy.sourceforge.net/, 2007.

[44] Eugene Tsui. The "ultima" tower, two-mile high sky city. Technical report, Tsui Design and Research, Inc., 2005. http://www.tdrinc.com/ultima.html.

[45] Jo Twist. Eco-designs on future cities. *BBC News*, July 14 2005. http://news.bbc.co.uk/1/hi/sci/tech/4682011.stm.

[46] Vargha J. *Civilization*. In GAIA: An Atlas of Planet Management, Editors: Myers N., Anchor Books, DoubleDay, New York, 2005.

[47] yWorks. yed - java graph editor. http://www.yworks.com/en/products_yed_about.htm, 2007.