Distributed Simulation of Transport Networks

Rowin Andruscavage
University of Maryland, College Park
Systems Engineering Master's Thesis
ENSE 799 Fall 2005
Dr. Mark Austin

Table of Contents

Purpose

This project serves to realize a distributed simulation designed during the course of the systems engineering master's program. The program will take a systems approach to modeling human habitats and the multi-modal transport networks that keep them running. This simulation framework would be used to create a baseline model of current day capacity, and then used to model the effects and quantify the benefits of investments in future infrastructure.

The distinguishing characteristics of this simulation framework includes:

- a hierarchical level-of-detail organization that allows available data from both top-down parametric 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 (e.g. Using data on the total gallons of fuel consumed by an airport per month and distributing that consumption across the aircraft that use that airport) and compare it to data generated by tallying up the individual fuel consumption of those aircraft. This would help calibrate the model by quantifying the effects unknown fuel flows, such as waste or other fuel sources. The hierarchical organization also makes the simulation easier to partition across distributed compute nodes.
- Rigorous balance and accounting for matter, energy, resource, and waste products across control volume boundaries. This is to keep us honest and avoid sweeping data under the rug. All input and output resources going into and out of any subsystem will show up in the final

accounting, and conversions and reactions within the subsystem's components will always be balanced according to conservation of matter and energy laws, ensuring that an unwanted waste byproduct is never “dropped” out of the simulation, or energy required to power a conversion or transaction comes out of nowhere. (Of course, it's still possible to simply underestimate the amount of fuel necessary to move cargo from point A to point B, but that's what the calibration against actual data is for).

Inspiration

Well, it all goes back to the meaning of life, doesn't it? We're all hanging around, looking for love or money or happiness, always trying to get the most out of life, and optimize our existence in some fashion. The optimization part is where simulation can be a useful tool, as we often disagree on what systems infrastructure improvements we could make in order to make us happier or richer or work not so far from our loved ones.

Most of our interactions with the urban environment in which we live involve transportation and delivery systems. These take many forms, ranging from various ground, air, and subterranean transit networks to power, water, and even information distribution infrastructure that feeds directly to our homes. Much of this infrastructure is put in place with funding or regulation from government agencies at national, state, and local levels.

During these times of rapid modernization, traditional governments can be a bit slow figuring out what infrastructure to invest in. Simulation is one tool that can come in handy to help quantify the benefits of different operational concepts. This analysis can be used to answer questions about design options. For example, to see whether the resources saved by restructuring, say, alternate water or package delivery systems justifies the deployment and maintenance expenses.

Need for arcologies – self-sufficiency

What main characteristic might distinguish something called an “Arcology” from any other urban construction? An Arcology would take a more systems-oriented view towards design and operation. No matter what other goal the development was targeting, whether that might be high population density by stacking up living spaces vertically, or ecological conservation by reducing and recycling as many of the waste products as possible, or simply just reducing pollution through architectural planning and landscaping that encourages walking over driving between locations, I would contend that the main feature that they all hold in common is a heightened sensitivity to the use and interchange of mass and energy between their outside environment and within themselves. In order to determine whether the development has reached any of these goals, we

would need to place more emphasis on determining what passes across the imaginary system boundaries and between components. And with better awareness of what we take and what we dump back into the surrounding area, we might be able to take measures that would make us more self-sufficient. Once we have economical technology to give us self-sufficiency, we might even have a chance at tackling some of the world problems stemming from contention for limited environmental resources and other factors that threaten our existence. These would include the standard run-of-the-mill end-of-the-world stuff, like war, famine, solar heat death, or our inability to eventually establish a working colony in outer space. But in more near-term practicality, we'd see expect to see our per-capita energy bills go down too.

Arcologies in history, media

- Current works

  Chinese, Tokyo arcologies, mixed-use developments in DC area

The closest present-day developments resembling arcologies are smattered around the world in various stages of completion. The truest to spirit arcology project in existence would be Arcosanti and Cosanti, the experimental communities arranged by architect and founding father of the “Arcology” concept Paulo Soleri himself. These small scale experiments in the Arizona desert are currently reported to be hovering around 5% complete.

The largest scale proposals have been cropping up in population-dense areas in Eastern Asia. A construction company has been promoting Tokyo's Sky City, covered in a Discovery Channel Documentary. Several proposals also exist for Chinese cities like Shanghai

- Preliminary requirements -
- Groundwork behind “wouldn't it be cool if...”
- Open-source blueprints/guidelines/standards as well as defining areas to take artistic license with
- Governance, hive-mind

Implementation Plan

The simulation will be built upon the open-source Dartmouth Scalable Simulation Framework (DaSSF). This provides a solid but flexible core discrete event simulation environment that should also make the simulation portable to other DES engines that adhere to the SSF standard. The SSF standard is primarily used to simulate data networks at the moment, but there's no reason the framework couldn't be used to create simulations of (simpler) physical systems.

I'll shy away from using the Ilogix Rhapsody UML-to-code IDE for developing the simulation, since I no longer have a licensed version of the software, and it was really more trouble that it was worth at the time anyway. Plus it would have involved reverse-engineering hooks into the DaSSF code to do properly. The object-oriented code used to build components will attempt to remain as true to the original UML diagrams as possible, though.

The simulation basically boils down to an accounting of conversion and transaction events that move resources between themselves and their environment. Therefore, most of the coding involves making and managing container objects. Building from the ground up, here's the implementation plan:

1. Resource containers are the most elementary class. They merely have to choose an identity, and store a number representing how much of this resource the owning object has pooled together. It needs getter and setter functions, and a master dictionary for looking up other useful properties associated with that type of resource (such as density, , market value, etc.) that might be used for various other calculations. Money and information are considered resources as well for tracking purposes, but they basically constitute an “activation energy” for a reaction or transaction to proceed, so are treated somewhat differently.
2. Reactors come in various forms and are intended to provide balanced conversions from one set of resources to others (often waste).
3. Cell objects are the hierarchical units. These will be the most complex but most useful class used in the simulation. They each can contain some combination of:
    1. resources
    2. reactors for converting internal resources from one to another
    3. buffers and constraints on the amount of resources they can hold before having to push them elsewhere
    4. parent, child, and peer cells with which to interact, such as by scheduling transactions and reporting metrics up and down their chain of command.
    5. internal agendas used to schedule reaction and transaction events.
4. Connective meshes define which cells can actually interact with each other, representing the function and capacity constraints of various transport networks that move resources between the cells in the system. They can exact a cost (in terms of money transactions and resources consumed.
5. The instantiation framework is what reads the scenario file and begins

to create cell objects and set up the simulation. This is where a modeling language would come in.

6. A reporting engine collects data from the simulation at desired intervals and needs to be programmed to extract useful data and analyses from the simulation.

Operational Concept

What potential uses could this transportation network simulation have? The types of problems I hope it will be useful for is demand generation. Different types of transportation infrastructures could be evaluated against each other to determine how well they meet that demand. Many existing transportation optimization problems tackle ways to increase throughput or capacity. But the task of urban planning should focus more on minimizing demand in addition to maximizing capacity. For example, instituting staggered work hours or telecommuting programs can relieve peak rush hour traffic congestion without spending a fortune widening highways and building additional infrastructure just to handle a few hours of peak usage a week. It would be nice to know how much incentives to provide to encourage employers to implement flexible work hours, or 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.

Also, by simulating demand, we can create a transportation system that is more sensitive to individual needs rather than the aggregate flow of travelers. 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. 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, even though the bus operators might only measure the time the passenger spends sitting on the bus and perhaps waiting for known connections.

An advanced busing system that dynamically generates routes and schedules based on individual source and destination requests from each passenger could achieve efficiencies and meet customer requirements far better than what we have today, and could make public transportation more attractive to people who drive their own vehicles in order to maintain that degree of flexibility. During peak commuting hours, this has the potential to reduce individual commute times, as buses could be scheduled more like express routes and fill up at one location and proceed 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 same routes with very low frequency, but would run on demand, cutting down wait times and making them a more convenient option for midday or late night errands. An effective public

transportation system should make a metropolitan area “smaller”, where each of its districts are easily accessible for connecting places where people live, work, and go for necessary errands and entertainment. Under the current hub and spoke paradigm, unless your source and destinations are near hubs or just down the street, travel on the system through two hubs can take up a significant portion of time. This time would typically consist of at least 5-10 minutes of waiting for each connection and perhaps 10-20 minutes riding each segment; the result being that driving independently in one’s own car would take between half or even a quarter of the time that the trip would take on public transit, even with traffic. For commuters, this time savings doubles, so it is of little surprise that most commuters prefer to spend the extra gas, auto maintenance, and toil to gain 1-2 hours of family time at home a day. Public transportation systems could still use a lot of improvement to make mass transit desirable over driving, rather than just an alternative to driving that merely relieves congestion on the roadways so that other drivers end up with a better traffic experience.

What defines a good inter-modal transit system? The conflicting goals might be characterized as: speed, response, 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)

- “Response” refers to the frequency of service, particularly how well it matches and meets demand. Extra time that people have to wait at their source or destination should be counted against the system… though this is almost always 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 “response” 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 are more spread apart.

- “Efficiency” might refer to two terms: that in terms of frugal monetary spending on operating costs and fixed infrastructure investments, 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 for whatever modest gains in speed, response, or coverage.

The main way we'll be able to improve efficiency (aside from simply improving fuel efficiency) would be to use existing resources smarter – 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 to not put all this capability to good use.

So instead of having fixed timetables locked down and set weeks, months, or even years in advanced, based only on projections from previous observations of seasonal, aggregate flows of the past, and barely ever followed to the minute, 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.

In order to operate in an inter-modal fashion, however, different segments of bus, rail, and even taxi and aircraft platforms must be able to exchange data with each other in order to feed the formulation of the global optimization problem. This also needs to interoperate between multiple jurisdictions and carriers, who will still want control over their own vehicle resources.

What kind of features would such a schedule collaboration system need to make a diverse set of platforms interoperate? First of all, we need to define a common language used to publish and exchange schedule and status data. Next, we would want to define schemas representing the types of data that are actually required, desired, or merely expressed as comments for general informational purposes. Some of the properties desired by this scheme could certainly be handled by an data representation framework like that provided by XML (extensible markup language):

- It should have a standard set of tools for processing and manipulating the data, a la XML's parsers and stylesheet transformations.

- The data representation format should be extensible, allowing newer versions of software to introduce new data types and tags without breaking older software that doesn't expect or understand the additional data. In a similar vein, older software in the system should

still preserve these newer data structures in messages that it passes along between other, perhaps newer or more capable software components that understand and can make use of it.

- The schemas should be centrally version controlled and available for verifying data types, etc.

This language feature set would allow different organizations to continue to share and integrate their logistics information, even as the set and functionality of the data schemas grow, change, and evolve over time. Incremental additions can be introduced, such as adding field for, say, the error or uncertainty surrounding a predicted arrival time – information that we might not be able to make good use of now, but could give us tangible benefits once we learn to process it better. Major version changes that alter the meaning of data fields in ways that are fundamentally incompatible with earlier versions could be introduced and managed by a central standards body, while a set of standard transformation filters could be provided to convert as much data between major revisions as possible.

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 of these values of interest might need to be expressed and measured in different forms, to indicate whether their values have been projected from previous data, predicted based on current known conditions, or are the actual measured values after the fact. Additionally, projections and predictions would want uncertainties attached to them in order to be of use for contingency planning.

First off, we will 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 be collected that would be of use:

- 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 them 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 runs 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 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 safeguards are simply not being deployed into the systems of today.

- Schedule constraints / flexibility : optimization thrives on having some slack or flexibility in its constraints. We could achieve more optimal schedules if only passengers could more adequately express things like:
    - What range of times could they be expected to arrive at their destination? Not later than 9:00?
    - How much extra would they 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, so their scheduling might flow around "hitchhiking" economically around the empty seats left over in schedules generated to server passengers paying for higher priority routing?
    - What kind of safety factor or time buffer are they comfortable with? Would they be willing to run through an airport to make a tighter connection?
- 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.

FIXME: Cargo security via digital signatures, accountability.

Having all this passenger and cargo data pretty much takes care of knowing the transportation system inputs. 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 intermodal optimization to work properly.

FIXME: network diameter, node degree

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 upon 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.

In any case, 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 the GPS system.
- 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 can and cannot take place.

In order to finally traverse this network, though, a transit system ultimately needs some set of vehicles (though many parts of a transit network might be represented as walkways on foot, which we might as well model too in order to help design capacity for escalators, moving walkways, ticketing and security checkpoints… perhaps even to make sure hallways and doorways are wide

enough to meet capacity and fire codes). Each vehicle would have associated with it:

- A geographic location within the network, whether it was a geographic location in transit, at or waiting for arrival at a station node, or even occupying a storage or a maintenance bay.
- A passenger or cargo capacity
- A set of rules governing how fast it can navigate 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 the optimized system. While we could squeeze a more optimal solution by imposing fewer constraints, for various reasons (such as lack of equipage to perform last-minute reroutes), we need some way of communicating and enforcing pre-existing schedule constraints. In the end, this probably isn't any different than the mechanism we'd use for introducing scheduled maintenance stops.

The last major category might include "environmental" factors that would affect the performance of the system. 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 predicable 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 contingency planning based on externally available data could make their way into streamlining other forms of transportation, albeit less dramatically.

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

We live in an uncertain world. How will the system deal with uncertainty and unexpected events in schedules? Probability should be built in to the optimization problem formulation, and one of the goals of the optimizer might be to minimize the impact of unfavorable (but probable) events. Analysis of historical records can generate performance metric associated with each vehicle, route, weather prediction, *etc*. A useful way of representing on-time performance probabilistically is to reconstruct the data from the cumulative distribution function (cdf) associated with the prediction. This would work much better than simply providing means and standard deviations, since most transit data is so skewed towards being late than being early. It's much easier to break down and be several hours late, than to speed across a transportation link in record time. The cdf can be quantized to reduce computational complexity, at the cost of adding extra conservative wait time buffers between connections.

FIXME: figure 1

While this type of data will be monitored and collected, only certain parts of the tail will interest the schedule optimizer.

FIXME: figure 2

We're primarily interested in what time the vast majority of the vehicles will arrive, as well as what hopefully small percentage are beset by schedule-impacting delays. There's no fixed "magic percentile" that would determine how much extra buffer time to schedule to make sure everyone makes their connections. This will likely be set arbitrarily at the beginning, as all of these factors contribute to an overall "confidence in planned schedule volatility" metric (maybe more easily expressed as an opposing "schedule stability" metric). With the optimizer system, we can recompute new schedules whenever an unexpected event comes up – such as when a vehicle is delayed enough to fall on the tail end of the cdf and it misses its connection. The optimizer can take that new information into account and simply create a new schedule based on these existing conditions – which will likely result in diverting other vehicles over to take care of the late straggling passengers. 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 risk likelihood 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 – say to catch a subway train that runs every 5 minutes – then the scheduler can comfortably deal with smaller buffers and higher schedule volatility risk. 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 can also help by figuring out the total impact on whether it's even worth holding flights for latecomers to make their connections.

FIXME: figure 3

So in addition to the overall transportation system performance optimization goals we discussed in [FIXME: Ref], we also want to introduce some practical optimization goals that will help the scheduler intelligently create and maintain buffers to deal with uncertainty. Now, how to formulate and computer this enhancement is beyond me, since it would likely require the optimizer to do risk-impact assessments on every combination of missed connection. But that's no reason to shirk away from providing the necessary information about on-time performance in the data protocol now, so that future generations of engineers could tackle it.

The final category of optimization constraints would come from the operators of the various transit networks. This would allow them to add crew and maintenance schedules, such that they can pick up and drop off drivers, pilots, and other staff at certain locations, or make sure that a vehicle ends up in a certain maintenance bay every so often for refueling and service.

These constraints are typically easy to add without a lot of heartburn, since they tend to help reduce the number of branch and bound paths that a mixed integer programming optimizer needs to search through to converge on a solution – at least as long as the solution remains feasible. The challenge comes in that expressing these constraints should be the job of the separate transit network organizations, and the abstract protocols needed to express these constraints would likely require extensive knowledge of how the global optimization problem is formulated and solved. It is undesirable to have this information format coupled too closely to the formulation, since it will make it more difficult to change and upgrade the optimization engine in the future. We don't want to force everyone to have to radically change their code at the same time throughout the system every time we want to introduce an incremental upgrade. We also don't want the entire systems upgrade to fail because of one or two late development efforts. We want enough abstraction built in so that they might make changes at their own pace to take advantage of new scheduling and optimization features and capabilities. Their abstract representation of their constraints needs the ability to compile itself so it can be applied to both the old and the new versions of the optimization formulation.

Unfortunately, I'm not able to come up with a language abstract enough that would allow the businesses to express what maintenance needs a generic optimizer must meet, without cheating and taking advantage of intimate knowledge of the formulation and the meaning of its various variables. A sophisticated abstraction language processor would have to take the expression and transform them into equations that relate particular variables to each other or to newly introduced variables. This processor would likely be nontrivial to implement and be prone to unexpected behaviors and errors. So a more practical way to handle crew and vehicle maintenance schedules would have the operators compute maintenance schedules separately from the main globally optimized schedule, and insert them as fixed constraints using the legacy

scheduling interface. The end result of performing iterations of this would not be as optimal as if the global optimizer took maintenance into account. But at least it starts close to an optimal solution, and provides our necessary layer of abstraction. The iterations would proceed something like:

1. Transit network operator would provide 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 find a new solution taking these new constraints into account, filling in new gaps in the schedule and hopefully not straying too far from the original optimal objective function result.

This would let us converge on a solution set somewhat near the optimal one that takes maintenance factors into account without tying down the programming to a particular implementation of the optimizer.

A global optimizer that did include operator goals and scheduling constraints isn't out of the realm of possibility, however. Additional complexity could be added by allowing these third parties to add their own set of constraint statements, even weighted objective functions. Some discipline would still be needed to keep the system stable. In the original form, the problem is formulated in advance, and the data provided by passengers and schedules add constraints in a consistent manner – the worst thing we should need to worry about are infeasible solutions. However, by allowing third parties deeper control of objective functions and constraint statements, we're exposing the system to a host of potential problems and vulnerabilities:

- Malformed or even malicious statements can 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 – but the latter approach could likely create unpredictable results.

- We'd need ownership and permissions on variables to separate the components provided by different parties. This would ensure that operators don't introduce constraints that could penalize their competitors.

- FIXME:REDUNDANT It would be nice to have that abstraction language and processor, so we wouldn't have to force everyone to recode all of their logic statements at once every time we want to

upgrade the problem formulation.

- Many companies pride themselves on their own optimization capabilities. We may need a mechanism to protect proprietary information about their mode of operation revealed in their contributed code statements. We could 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 segment of the global calculation, interacting with the rest of the system through the input and output protocols.

Hopefully these reasons (and probably others) have helped to articulate why I haven't addressed these issues in the current incarnation of this thesis. But this might be the beginning of an outline to tackle these considerations in the future.

Computational Considerations

Large scale global optimization can require a lot of computing power. It falls under the class of NP hard problems that scale exponentially with the number of transit nodes we add to the transportation system. Let's look at some of the ways in which problems of this size can be tackled.

The solver should be set up to run in parallel across several CPUs, scaleable to a massive clustering system. Many linear and mixed integer solvers have the capability to run on this type of platform, so it's not something we have to worry about directly.

We still would need to resort to a host of other tricks to reduce the computational complexity enough to approach any 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.

- 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.
- 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 requirer 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 connections" 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. Of course, low priority bulk cargo may find some advantage through waiting for these multiple connections, filling in otherwise "empty" space leftover on any flight where the opportunity arose to get it slightly closer to its destination. But at some point all of the extra handling and transfer overhead ought to outweigh whatever small price break.

- Just about any schedule constraint that would help "lock down" otherwise free-floating variables would help reduce the search space. Feeding in initial conditions - like the current location of 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) would help speed the optimization along.
- Sometimes it may be necessary to simply add other heuristic or even arbitrary constraints to help the system converge on a solution. Many of these constraints probably won't even affect the solution, but constrain the search space enough to allow a much quicker answer.

All else failing, many mixed-integer programming solvers also allow "good enough" solutions to be given without a complete exhaustive search of the solution space. Modern MIP solvers can be pretty clever about searching the "most promising" paths first, so completing the entire exhaustive search would yield little improvement on the objective function. Of course, this technique only applies if a feasible solution is found at all.

Finally, a sophisticated optimization would involve precomputing 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 incremental adjustment may be necessary to recover from small, unexpected schedule 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. An "incremental update" to the solution performed with minimal recalculation might be achieved by determining which vast majority of system variables shouldn't be affected, and formulate a highly-constrained optimization problem that only searches through a small set of variables affected by the unexpected change in one or two schedule input values. We'd need to develop a heuristic to determine exactly how far out this limited set of "affected variables" should reach.

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

So what can we do once we have a coupled system of transit networks, a simulation of that system, and an optimization framework that can set up schedules for the simulation (or the actual system) to evaluate? We can set up yet another iterative optimization – this time of the actual system configurations

and not just one schedule. This 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. We can propose a new construction or infrastructure project, show its benefits in a simulated model, and later validate those benefits using data collected from the real system. Competing models for improvements might even have the chance to provide benchmarks using the same methodology.

FIXME: Insert diagram

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. And the focus on operational efficiency and continuous improvement driven by pervasive measurement and analysis will lead towards a leaner, sustainable society where more resources could be directed towards forward progress instead of consumption.

Several initiatives are currently underway to rethink the way metropolitan areas are designed. This simulation modeling & analysis framework can help accelerate the worthwhile changes.

The urban development paradigm of roughly the last century has been characterized by suburbanization. Massive superhighways were built between cities while suburbanite families sprawled out along these corridors. As the space between these corridors filled out and the main thoroughfares became congested, more highways were built to relieve congestions. However, after a certain point, the ratio of highway space to developable, livable area becomes saturated, and you get diminishing returns from building more roadway infrastructure. Highways take up a lot of space, and when we start to pack those highways close together, we end up spreading out actual useful land into isolated pockets nestled between interchanges. The more complicated the interchange between multiple highways, the larger and more funded the construction project gets. What's more, having multiple highways down busy rush-hour corridors don't really make the world any smaller. A sparse network of good, uncongested highways should make it take just as long to get from point A to point B without having to build and maintain several alternate routes that exist just to relieve rush hour congestion (I'm thinking of the Baltimore-Washington corridor in particular, with I-95 flanked by 295 and Rt.29).

References

http://www.ssfnet.org/ Scalable Simulation Framework Research Network

http://www.cs.dartmouth.edu/research/DaSSF/ Dartmouth Scalable Simulation Framework

http://www.crhc.uiuc.edu/~jasonliu/projects/ssfnet/

http://www.renesys.com/research.html Renesys Corporation

https://gradus.renesys.com/exe/Raceway Renesys Raceway

http://dsc.discovery.com/convergence/engineering/skycity/interactive/interactive.html "Tokyo Sky City", Discovery Channel

http://www.biospheres.com/ Biosphere 2 Biosphics

http://www.nd.edu/~ndmag/arcosw98.htm John Monczunski "Arcosanti, a Habitat for Humanity" *Notre Dame Magazine*, Winter 1998-99