CONGESTION-AWARE DYNAMIC ROUTING IN AUTOMATED MATERIAL HANDLING SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Kelly K. Bartlett

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology
December 2014

CONGESTION-AWARE DYNAMIC ROUTING IN AUTOMATED MATERIAL HANDLING SYSTEMS

To my son, Garrett.

ACKNOWLEDGEMENTS

First, I would like to thank Samsung and the U.S. Army Accessions Command for supplying both the financial support and the motivation for my work. I hope that our insights and solutions can be implemented in practice to make someone's job easier and/or more effective.

Second, thank you to my advisors George Nemhauser, Shabbir Ahmed, and Joel Sokol. You have supported me both professionally and personally through this process and provided motivation, mentorship, and example that will always inspire me. I cannot thank you enough.

Third, thank you to my fellow students, particularly Junho Lee who has put up with my never-ending questions and hectic schedule. I also appreciate your willingness to play tour guide in Seoul. Thank you to Tim Sprock who has always been available to listen and to Jackie Griffin and Kate Lindsey who provided proof that it is indeed possible to complete a PhD.

Lastly, thank you to my family. Thank you beyond words to my husband Pat and my son Garrett. Garrett, no matter how much I have learned over the past several years, you have learned infinitely more. You inspire me every day with your curiosity. Pat, I truly appreciate your support throughout this process. Thank you for believing in me. Lastly, thank you to my parents for their unending support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Semiconductors are a central component of all electronic devices including mobile phones, computers, and televisions. Demand for these items is only expected to increase as electronics become more and more integrated into our society. In 2011, over 7 billion semiconductor wafers were produced worldwide totaling $300 billion in sales [27]. By 2014, demand was expected to exceed $336 billion, a 12% increase in three years [69]. In order to keep up with demand and improve responsiveness, the International Technology Roadmap for Semiconductors (ITRS) has set an objective of a 15% reduction in cycle time for 25 wafer lots by 2020 [24]. An important component in cycle time reduction is an efficient material handling system. Efficient material handling results in low tool-to-tool delivery times and keeps tool utilization high [24]. To improve efficiency in material handling, this thesis proposes an adaptive dynamic routing approach for unified AMHS systems that allows the system to self-regulate, reducing steady-state travel times and avoiding excessive congestion and deadlock. Additionally, our approach allows the AMHS layout to be improved such that vehicle travel times are further reduced while still avoiding heavy congestion. To demonstrate the effectiveness of the proposed approach, we developed a high-fidelity simulation of vehicle movement and an associated automated layout generation tool. This allows broad analysis of system design and operational control.

In the first part of this thesis, we detail the complexity of AMHS operation and motivate the need for analysis through simulation. We then present a high-fidelity discrete-event simulation developed in AutoMod to simulate the movement of vehicles in a unified AMHS. It simulates the movement of vehicles move through a facility on a top-mounted track system. Vehicle movement is highly complex and difficult to model analytically due to the high number of interacting vehicles, acceleration and deceleration when exiting or entering curved edges, and the inability to pass slow-moving or stopped vehicles. Our simulation accounts for these factors, implements a complex policy for the control of idle vehicles, and mimics realistic traffic patterns through the use of prototype production sequence data.

Secondly, we propose a dynamic routing approach that allows vehicles to be rerouted while in progress in response to changes in the locations and severity of congestion. If the origin, destination, and timing of all movements were known in advance, a comprehensive routing plan to reduce delay could be generated a priori via large-scale optimization. The uncertainty and production flexibility inherent in this type of system, however, make this an online or real-time problem. The scale of the system (hundreds of vehicles, processing locations, and storage locations) and the frequency with which transfer requests occur (often less than one second between requests) require an approach without excessive computation time or highly sophisticated methods.

We propose that predicted edge traversal times be stored for each edge and updated via exponential smoothing each time a vehicle completes traversal of the edge. A vehicle's route is controlled by routing tables stored at each diverging node. Tables are periodically updated to reflect current predicted edge traversal times. A vehicle selects the route that minimizes its own predicted travel time. Because predicted edge traversal times are based on recent history and do not depend on the routes selected by other vehicles, the solution can be decomposed by vehicle and the system-optimal solution is equivalent to the user-optimal solution at a specific point in time.

We demonstrate that the consideration of congestion in routing decisions results in a significant reduction in the frequency of heavy congestion and a 4%-6% improvement in steady-state performance in a simulated prototype facility. Further, in the case of exceptions such as vehicle breakdowns, dynamic routing allows the system to recover to steady-state 80% faster than static routing even if static routing considers congestion.

Lastly, we consider how the use of dynamic routing changes the effectiveness of layout design. We present an Excel-based user interface that automatically generates simulation files with the click of a button. The user selects characteristics from a modular template and Visual Basic for Applications (VBA) generates files formatted for use with AutoMod as well as all required input text files. This tool is flexible enough to allow analysis of thousands of layouts for a bay-based rail structure. Together with the simulation, these tools allow extensive analysis of the potential impact layout changes made possible by dynamic routing.

Using this tool, we investigate the dual impact of dynamic routing and center and outer loop redesign. Shortcuts on the center and outer loop allow both shorter point-to-point distance and more alternate routes and detours. Under dynamic routing, the system is able to use these alternate routes effectively to avoid heavy congestion while taking the shortest possible route given current congestion. In our simulation, we demonstrate that the preferred layout for dynamic routing differs from that for static routing. Further, dynamic routing allows an eight-fold increase in the number of shortcuts on the center and outer loops which reduces delivery time by an additional 24%.

CHAPTER I

INTRODUCTION

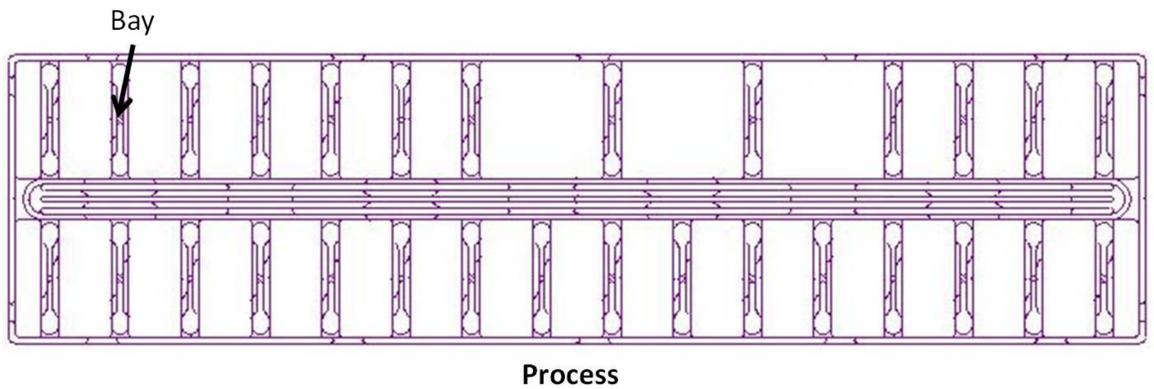
Semiconductors are a central component of all electronic devices including mobile phones, computers, and televisions. Demand for these items is only expected to increase as electronics become more and more integrated into our society. In 2011, over 7 billion semiconductor wafers were produced worldwide totaling \$300 billion in sales [27]. By 2014, demand was expected to exceed \$336 billion, a 12% increase in three years [69]. In order to keep up with demand and improve responsiveness, the International Technology Roadmap for Semiconductors (ITRS) has set an objective of a 15% reduction in cycle time for 25 wafer lots by 2020 [24]. An important component in cycle time reduction is an efficient material handling system. Efficient material handling results in low tool-to-tool delivery times and keeps tool utilization high [24]. In this thesis, we focus on a particular type of automated material handling systems used in semiconductor manufacturing. Specifically, we propose an adaptive dynamic routing approach for unified automated material handling systems (AMHSs) that allows the system to self-regulate, reducing steady-state travel times and avoiding excessive congestion and deadlock. We use simulation to demonstrate the effectiveness of this approach. Lastly, we demonstrate that dynamic routing makes particular layout changes possible that further improve steady-state travel times.

1.1 Motivation

1.1.1 System Description

Semiconductor manufacturing is a highly complex, re-entrant process in which wafers, packaged in cartridges, are transported through hundreds of processing steps in a clean-room facility. Because the production process is not linear, like an assembly line, but re-entrant where parts return to the same type of machine several times, the flow of materials through the facility is highly complex. A cartridge will typically progress through 400-500 processing steps over 6-8 weeks and often be placed in storage locations between processing steps. There are typically fewer than ten types of processes, so a cartridge returns to the same process many times.

Semiconductor manufacturing facilities, also know as wafer fabrication facilities or wafers fabs, are typically organized into bays. Each bay contains machines that perform one of several processes and many adjacent bays may contain machines of the same type. This is done for equipment maintenance reasons and/or to facilitate the storage of chemicals used by the process. Figure 1.1.1 shows the bay-based rail structure and process assignment on a simulated prototype layout. The processes that we consider are chemical vapor deposition (CVD), cleaning (CLN), photolithography (PHO/PHOTO), etching (ETC/ETCH), chemical-mechanical planarization (CMP), implantation (IMP), diffusion (DIFF), and metalization (MET/METAL). Three bay locations have been removed to represent processing equipment that is large, taking up more space than that typically allotted for a bay.



Process

CVD	CVD	CVD	CVD	CVD	CLN	CLN		PHO		PHO		ETC	ETC	ETC	ETC
CMP	CLN	IMP	IMP	IMP	DIFF	DIFF	DIFF	DIFF	MET	MET	MET	MET	CLN	ETC	ETC

Figure 1: Prototype Facility Structure and Bay-to-Process Assignment

In previous technology generations, cartridges were transported by humans or a combination of humans and automated systems. As technology has progressed, however, the size, weight, and value of a cartridge has increased such that now this requires automated material handling systems (AMHSs). In 300mm wafer fabrication facilities, the current technology generation, various types of automated material handling systems (AMHSs) are used, including track and conveyor-based systems. Track-based systems employ Overhead Hoist Transport (OHT) vehicles that move along tracks suspended from the ceiling. An example from www.spectrum.ieee.org is shown in Figure 1.1.1.

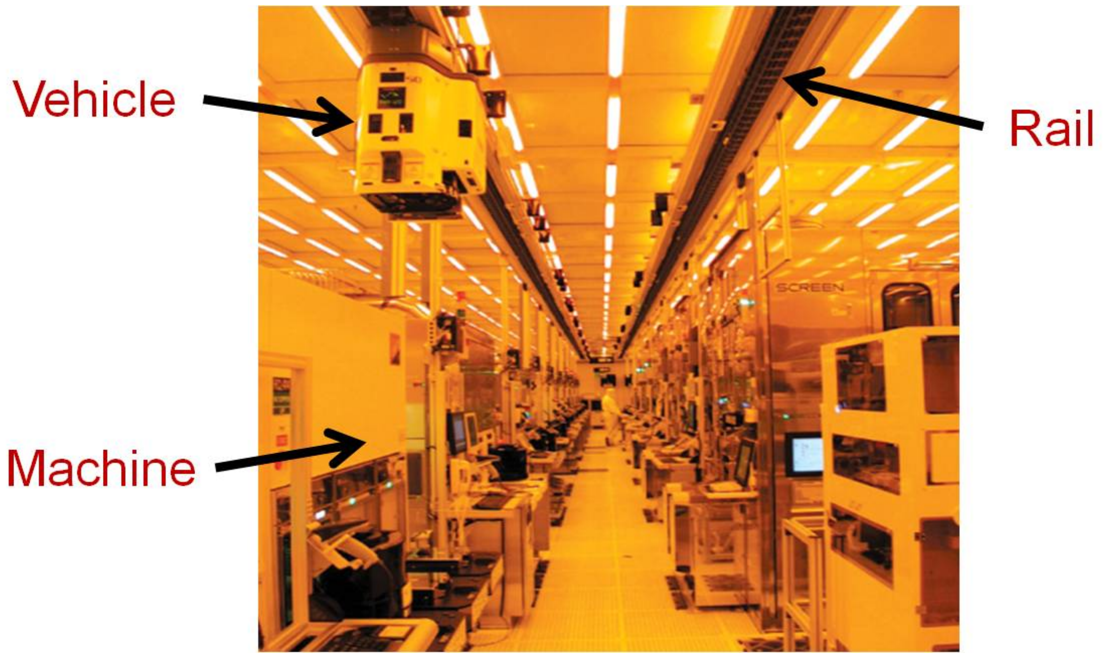


Figure 2: Overhead Hoist Transport (OHT) System

OHT systems may either be segregated, where vehicles are confined to one area of the facility and transfered from area to another via storage containers known as stockers, or unified, where vehicles may move throughout the facility. In a unified OHT system, bays are connected via a center loop and an outer loop and each bay has two entrances and two exits. The center loop typically has four to eight travel lanes with shortcuts to move among them. Figure 1.1.1 shows representations of these two configurations. Segregated systems were most common in early 300mm facilities, but unified systems have become more popular as the inefficiencies of segregated systems are realized [6, 30].

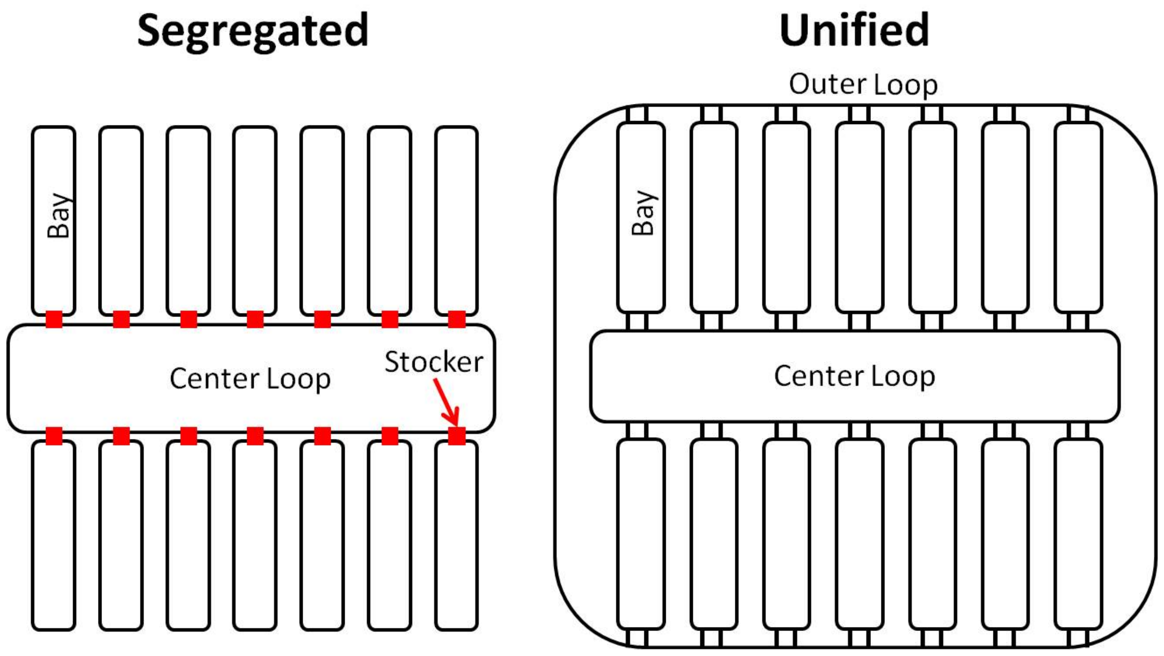


Figure 3: Segregated and Unified AMHS Configurations

Conveyor systems have also been proposed where cartridges move along conveyors and are, thus, not associated with vehicles. This continuous transport method is considered a low-cost alternative in low-volume and small-batch facilities [51]. It is not frequently used in 300mm facilities but is being considered as the industry transitions to the next generation, 450mm chip facilities [46]. In this thesis, we focus on unified systems, motivated by current practice.

The production scheduling system, which determines when and to where each cartridge will move, is designed to minimize cycle time and maximize throughput. Due to processing time uncertainty, the system does not pre-assign cartridges to machines, but makes this assignment in real-time. Although the next process step that a cartridge will visit is fixed by its production sequence, the exact machine or storage location it will visit is not determined until it has complete processing at its current step. At this time, a transfer request is generated specifying an origin and destination and the material handling system responds as required to transport the cartridge from one location to another.

In an OHT system, when a transfer request is generated, a centralized control system assigns a vehicle to transport it. If no vehicle is available, the system adds the request to a queue for requests awaiting assignment. Figure 1.1.1 represents vehicle movement graphically. Upon assignment, the vehicle travels to the current location of the cartridge, unloads the cartridge from the machine into the vehicle, travels to either the next machine in the processing sequence or to a storage location, and loads the cartridge into the next machine or storage location. The vehicle then becomes idle until it is either assigned to a new request or asked to move so another vehicle may pass. Routes must be selected from the vehicle's starting location to the request origin and from the request origin to its destination.

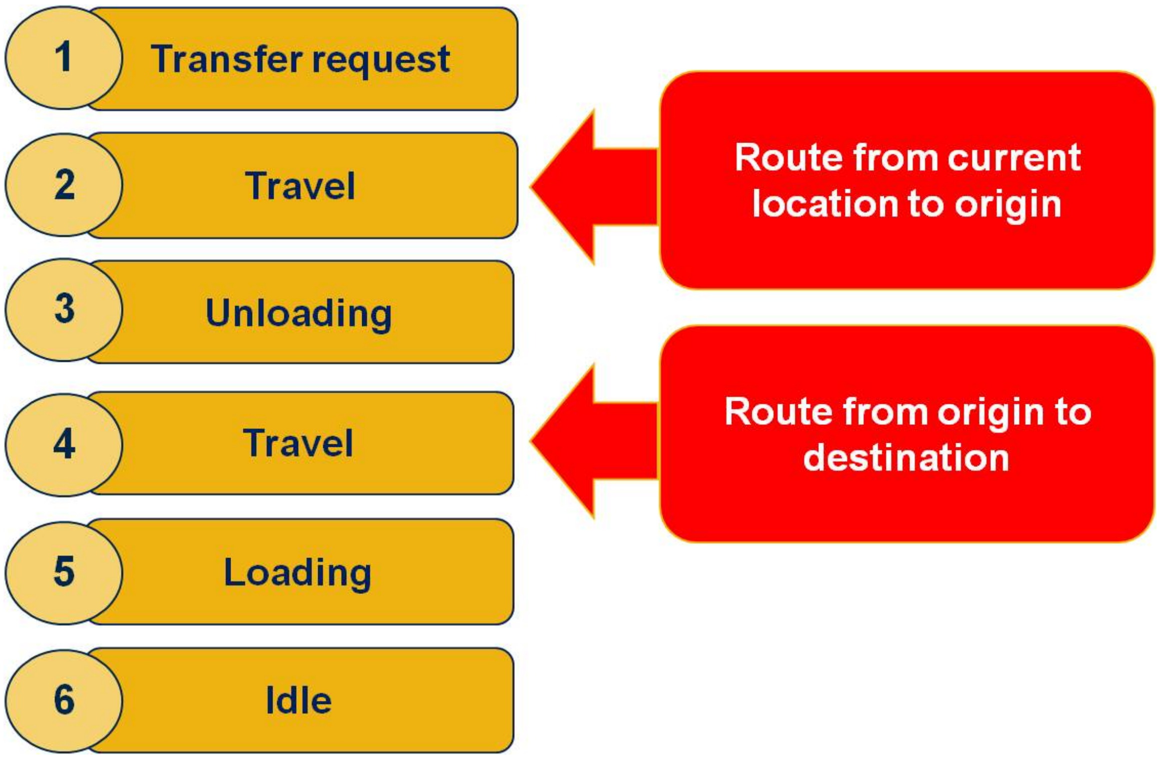

1 Transfer request
2 Travel
3 Unloading
4 Travel
5 Loading
6 Idle
Route from current location to origin
Route from origin to destination

Figure 4: Logic of Vehicle Movement

1.1.2 Challenges

There are several design and operational decisions involved in making this complex system run smoothly and efficiently. We categorize these decisions into layout design, production scheduling, and material handling decisions. Design decisions include locating equipment on the production floor, selecting a material handling system, and designing the material handling system layout. In unified systems, the material handling system layout design involves both high-level design (i.e. a bay-based structure) and low-level design (placement of shortcuts). Production scheduling decisions determine when and to where each part will move. If a machine is available or will be available soon, the production scheduling system determines which part will be processed next. Overall, it aims to reduce cycle time and increase throughput while considering complexities such as high-priority production lots and engineering test lots. Additionally, it controls the use of storage locations throughout the facility. Material handling decisions determine the assignment of vehicles to requests, the routing of vehicles to serve requests, and the behavior of idle vehicles. The routing of vehicles and control of idle vehicles becomes much more complex in unified systems.

Material handling decisions are typically slave to production scheduling decisions in the sense that the material handling system has little advance knowledge of production scheduling decisions and has no influence over them. Because the production scheduling and material handling systems do not exchange information to aid in decision making, the material handling system cannot make anticipatory decisions. Vehicles cannot avoid areas where new requests are likely to occur and production scheduling does not avoid assigning a cartridge to a machine in a congested area.

In this thesis, we focus on vehicle routing and the impact of vehicle routing policy on layout design. Specifically, we seek to improve the system's ability to transport material quickly and predictably by improving vehicle routing policies and analyzing rail layout design. Achieving this goal involves reducing congestion and the resulting average travel times as well as reducing the need for human intervention to resolve case of severe congestion. Because layout design, production scheduling, and material handling decisions are intrinsically linked, studying one of the three requires consideration and modeling of of all three. Operationally, we focus on the high-fidelity modeling of vehicle movement because this is our primary focus. To do this effectively, however, we must also model production scheduling in sufficient detail to capture the impact of production scheduling on vehicle routing. Even with a simplified version of production scheduling, this system is difficult to model analytically due to the inherent uncertainty due to processing time variability and the complexity of vehicle interaction. The size of the network and the frequency with which decisions must be made make computational efficiency a priority.

1.1.3 System Scale

Facilities often have 30 or more bays and thousands of stopping locations. Stopping locations include both machine ports and storage locations. Suppose the facility is modeled as a graph where nodes represent both stopping locations and rails intersection points and edges represent rails connecting nodes, the number of nodes may be several thousand. Although the number of edges is limited because stopping location nodes will have only one incoming and one outgoing edge and intersection point nodes will have degree three (the sum of the number of incoming and outgoing edges will equal three) because of the physical facility structure, the number of edges will be at least equal to the number of nodes but will not exceed 1.5 times the number of nodes.

This network size is not, in itself, excessively large. However, when considered in combination with the frequency with which decisions must be made, it becomes intractable for computationally intense methods. Requests typically occur at a rate of more than one per second and are not known in advance. Each time a request arrives, a decision must be made on which vehicle to assign to a request and which route a vehicle should take from its current location to the origin and from origin to destination. Additionally, because vehicles may not pass one another, idle vehicles must yield to active vehicles, moving out of the way so the active vehicles may continue on their paths. At the time of a yield request, the vehicle either moves out of the way of the active vehicle as quickly as possible or is redistributed to another location in the network so that not too many idle vehicles become clumped in one location. Each yield request, thus, requires an additional routing decision. Even if we assume the vehicles are routed statically, that is their routes may not change once they have been assigned, this still amounts to more than one decision each second, on average.

1.1.4 Uncertainty

If all requests were known in advance, the routing problem could theoretically be modeled as a large-scale integer program (although computational tractability may still be an issue). All vehicle routes could be assigned a priori and system performance would be predetermined. In our system, processing times are uncertain and the production scheduling system is flexible in that cartridge-to-machine assignments are not made ahead of time. For the material handling system, this means that transfer requests are revealed over time and are not known in advance. This is know as an online or real-time problem. Most online problems are impossible to model deterministically. Stochastic methods, such as queuing have been used to approximate system performance in similar systems, but they typically do not adequately model vehicle interaction. They have been used extensively in the modeling of individual loops in segregated AMHSs because the scale of these systems is smaller and vehicle interaction less complex.

1.1.5 Vehicle Interaction

The complexity of vehicle interaction results from the track-based nature of the system, the size of vehicles relative to edge size, and the non-negligible effect of acceleration and deceleration. When a vehicle stops to load or unload, it blocks other vehicles from continuing along their intended route. Loading/unloading time is not negligible, approximately 10 seconds. Loading and unloading occurs mostly in bays but also at storage locations along the center loop. Thus, as a vehicle travels from one location to another, it will necessarily pass many potential stopping locations. Also, in the case of very heavy demand in one area of the network (frequent loading/unloading), traffic may spill out from a bay into the center or outer loop.

Vehicle interaction is further complicated by acceleration and deceleration. Vehicles require time to stop and start prior to and after loading/unloading and when entering or exiting a curved edge because maximum velocity differs on straight and curved edges. Not only will a decelerating vehicle delay itself while decelerating (this, of course, is known in advance), but it will delay vehicles that are following in close proximity. These effects are difficult to model exactly.

1.1.6 Production Scheduling

Even with a primary focus on modeling vehicle movement, it is important to consider how changes in production scheduling input or decisions may impact these decisions. In particular, we care about the impact of product mix on demand pattern. Each product has a unique production sequence, visiting production processes in a different order, and product mixes change frequently. Thus, the flow of traffic through the facility, or demand pattern, may differ. Ideally, any vehicle movement policy or layout design decision is robust to changes in product mix. The challenge in modeling this in our problem is to effectively incorporate demand pattern while not requiring detailed input data or excessive computation time.

1.2 Contributions

In this thesis, we address the questions:

1. How do we model a unified semiconductor OHT system to evaluate operational policies and layout design decisions?
2. How do we improve operational vehicle routing policies to reduce steady-state travel times and avoid heavy congestion and deadlock?
3. How do we evaluate potential layout design changes, considering the impact of operational policies on performance?

1.2.1 High-Fidelity AMHS Simulation

As described in Section 1.1, unified OHT systems present several modeling challenges that are difficult to deal with analytically. Most significantly, demand uncertainty and vehicle interaction. By demand uncertainty, we mean that transfer requests are not known in advance and, thus, cannot be modeled deterministically. The high level of variability in the material handling system, due to on-track loading and unloading, and the size of the system and complexity of vehicle interaction make it difficult to model via queueing methods. Thus, simulation is required to effectively evaluate operational policies.

In Chapter 2, we present a high-fidelity discrete-event simulation developed in AutoMod to simulate the movement of vehicles in a unified AMHS. It simulates track-based vehicle movement and incorporates complexities of vehicles interaction, such as acceleration and deceleration, on-track loading and unloading, and an idle vehicle policy that requires that idle vehicles move out of the way of active vehicles. To model demand patterns, it uses probability distributions generated from prototype production sequences. We use our simulation to evaluate a new dynamic routing policy and, further, to evaluate potential layout changes for use with dynamic routing.

1.2.2 Dynamic Routing

In practice, the route that a vehicle travels is typically selected based on static factors such as distance, maximum travel velocity on an edge, and the location of loading/unloading points. All requests for transport between a specific origin and destination will use the same route regardless of current congestion location and intensity and follow this route regardless of how the system changes. If congestion develops on the intended path, the vehicle will still travel directly into the congestion, both delaying itself and making the congestion worse.

Due to the structure of semiconductor manufacturing facilities, static routing often causes significant congestion including heavy congestion that requires manual intervention to resolve. Heavy congestion is defined as a system state where vehicles are unable to progress to their destinations at a reasonable pace. For our purposes, we define heavy congestion as a speed index falling below 0.25 where speed index is the average ratio across all vehicles of current velocity to maximum velocity. The average speed index does not consider vehicles that are stopped to load or unload or idle and intentionally stationary and maximum velocity is determined by a whether a vehicle is on a curved or straight edge. Figure 1.2.2 shows a simulated occurrence of heavy congestion. Blue and black vehicles should be moving and red vehicles are stopped to load or unload. Though this situation is not a deadlock because vehicles are still able to move, too many vehicles are trying to use the same edges, including one where frequent stopping occurs. This heavy congestion results in severe delay and may result in deadlock where vehicles are unable to progress. In practice, the system may not reach as severe a state as in the simulation because engineers will intervene before that happens. The need for intervention, however, disrupts system efficiency and requires close monitoring. In order avoid such situations, engineers use ad hoc congestion penalties. The method is not systematic, does not easily accommodate changes in production sequence and product mix, and does not respond quickly to changing congestion characteristics.

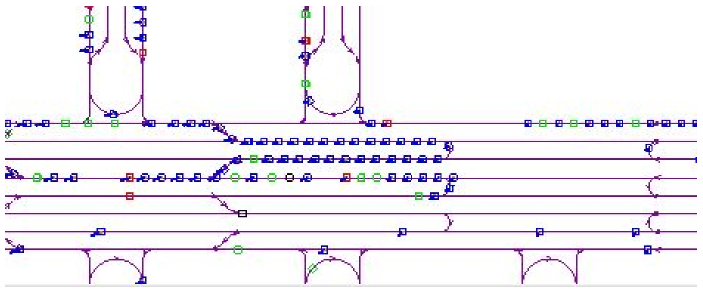


Figure 5: Example of Heavy Congestion

In Chapter 3, we propose a *dynamic routing* approach that allows vehicles to be rerouted while in progress in response to changes in the locations and severity of congestion. We consider that rerouting vehicles while in progress will increase the number of decisions significantly, often by more than a factor of ten. Thus, the proposed method must be computationally efficient enough to handle this but sensitive enough to effectively respond to congestion.

We propose a lookup table-based method where tables are periodically updated to reflect the current system state. Each node is associated with a lookup table that contains the next node in the path to each potential destination. When a vehicle must make a routing decision, it looks up its next node in the appropriate routing table and no new calculations are necessary. This is a dynamic routing method because routing tables always send a vehicle over the path with the earliest estimated arrival time regardless of whether that path has changed since the vehicle's last decision point.

We propose the use of an all-to-all shortest path method to compute the routing table where edge weights are continually updated based on recent history. Although other table updates methods are possible, this method requires the fewest computations. The challenge is then how to estimate travel time on each path in order to calculate shortest paths. We use an exponential smoothing method where the edge weight for a particular edge is updated each time a vehicle traverses the edge. Given these edge traversal time estimates, at a particular point in time, the all-to-all shortest path solution is a system optimal solution. Due to the uncertainty in the system, however, there is no guarantee that the solution will be optimal over time.

We evaluate this approach relative to static routing using our simulation. We consider the frequency of heavy congestion, steady state travel times and congestion, and the efficiency of recovery from vehicle breakdown. We define a replication of the simulation as resulting in heavy congestion if the speed index, as defined previously, falls below 0.25. While static routing frequently results in heavy congestion, dynamic routing does not experience any replications with heavy congestion. For replications that do not result in heavy congestion, we consider steady-state travel times, both origin-to-destination delivery time and total system time, and speed index. In steady-state, speed index is a measure of how smoothly traffic is moving through the system. In our simulation, we observe approximately a 5% improvement in steady-state metrics. Lastly, we consider response to and recovery from vehicle breakdown. We observe that dynamic routing immediately begins to route vehicles away from this area of the network while static routing does not. This results in improved system recovery, both in the frequency of recovery (in most cases, static routing is not able to recover at all) and in the time it takes to recover.

1.2.3 Layout Analysis for Dynamic Routing

Having demonstrated that dynamic routing improves performance on a fixed layout, we consider the question of how the layout can be adapted to make better use of dynamic routing. It has been observed, both in practice and in simulation, that the addition of a edge can lead to increased congestion and decreased routing performance. With dynamic routing, however, vehicles will avoid the new edge and the surrounding area if it becomes congested. Thus, we would like to determine what the optimal layout is for use with dynamic routing. However, most facility layout problems have been shown in be NP-Hard [13]. In particular, even in simplified flow-based networks, such as the traffic assignment problem, the problem of determining an optimal layout is NP-Hard even for a single commodity [54]. Again, we turn to simulation to address this question.

In the layout design process, ideally we are able to compare several (or more) layouts in an iterative design process. To create simulation layouts manually is both time-consuming and error-prone. In Chapter 4 we present an Excel-based user interface to automatically generate simulation files with the click of a button. The user selects characteristics from a modular template and Visual Basic for Applications (VBA) generates files formatted for use with AutoMod as well as all required input text files. This tool is flexible enough to allow analysis of thousands of layouts for a bay-based rail structure. The user may change the number and travel direction of the center loop, the connection type between the center and outer loops, and the number and placement of center and outer loop shortcuts. Shortcuts allow movement among long parallel travel lanes. The number of bays can be changed as can parameters that allow sensitivity analysis with respect to production scheduling, such as bay-to-process assignment and the probability distributions associated with demand pattern.

This tool may be used as part of an iterative design process where a system layout or set of candidate layouts is selected through engineering experience or an optimization model. The tool is used to translate those layouts into simulation files, and the simulation provides performance data resulting from running the simulation on these layout(s). These results may be used to refine an optimization model or provide insight to engineers. This framework is shown in Figure 1.2.3.

We demonstrate, through the use of this tool, that dynamic routing allows an eight-fold increase in the number of shortcuts on the center and outer loops without resulting in heavy congestion. Whereas in static routing, adding a shortcut may

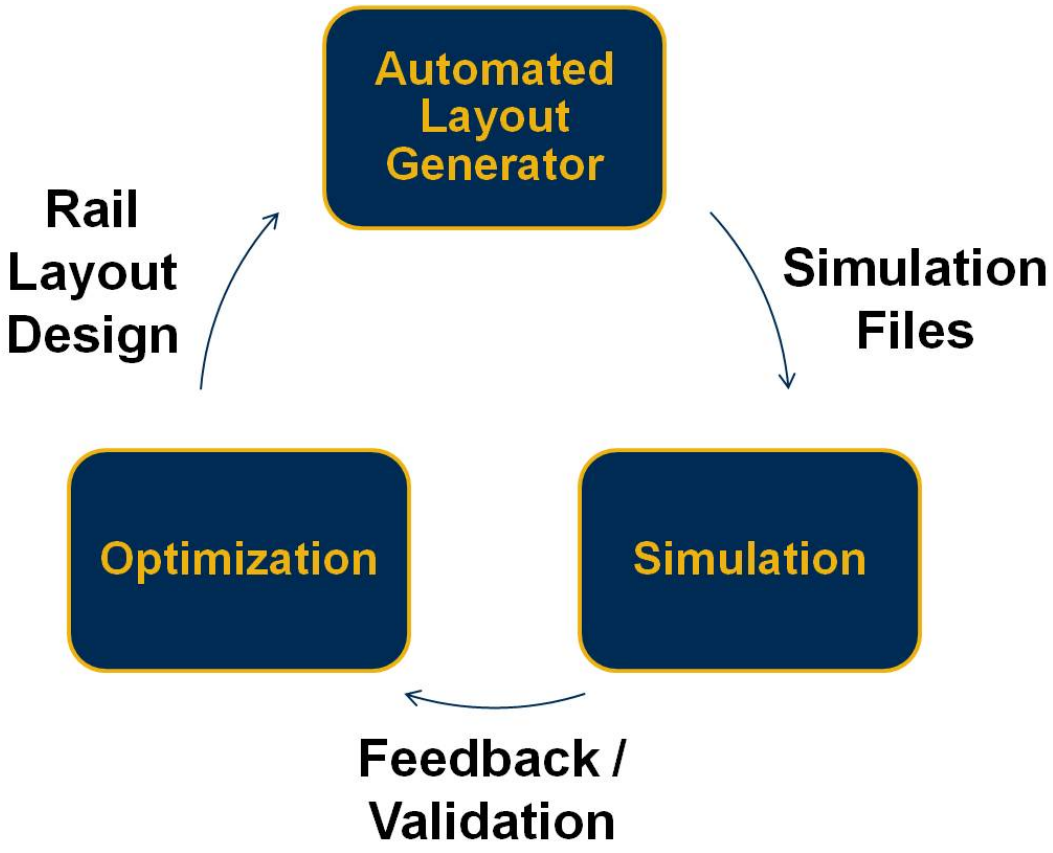


Figure 6: Layout Design Process

cause additional congestion, with dynamic routing shorter point-to-point distances result in shorter travel times without heavy congestion. We consider four candidate center loop layouts that differ by the number and placement of shortcuts. With static routing, the system is particularly sensitive to the placement of shortcuts and frequently encounters heavy congestion in the same locations. With dynamic routing, heavy congestion is avoided until bays become overloaded with their own demand. Because the addition of shortcuts also reduces point-to-point distances, travel times decrease even though speed index increases slightly in some cases. This analysis can be viewed as the first step in an iterative process.

1.3 Literature

We review relevant literature in AMHS and AGV systems. While the body of literature is much broader for AGV systems, most systems studied are significantly smaller than our system, operating fewer than twenty vehicles. Our system typically operates several hundred vehicles. AGV system design incorporates design considerations such as equipment placement, rail configuration or flow path layout, and stopping point placement as well as operational policy management such as vehicle dispatching, vehicle routing, power system management, and idle vehicle control [65]. Production scheduling decisions such as work-in-process (WIP) management and lot prioritization also play an important role. We focus on decisions affecting vehicle routing and layout design decisions. In this section, we first discuss several relevant surveys and the focus on literature in modeling techniques, vehicle routing, and layout design.

Several surveys from the last fifteen years provide high-level overviews on decisions related to AGV and AMHS system design. Vis [65] covers the whole spectrum of design decisions including flow path layout, collision avoidance in systems with bi-directional travel, stopping location placement, selecting an optimal number of vehicles, vehicle dispatching, vehicle routing, and idle vehicle control. It points out, in particular, that these decisions and systems are interrelated but often considered separately.

Agrawal and Herugu [1] surveys fab design and analysis in semiconductor manufacturing. Design issues focus mostly on the difference between segregated, conveyor-based, and unified AMHSs. Recall that segregated systems comprise independent loops for each bay and the center loop and transfer cartridges among loops via stockers. In unified systems, vehicles travel throughout the facility. Lin et al [39], Lin et al [40], and Bahri et al [6] also compare several layouts, including segregated and unified layouts, and conclude that unified systems perform better in terms of delivery time, throughput, and vehicle utilization. Agrawal and Herugu[1] also cite several simulation-based analyses on design decisions and operational policies, emphasizing that simulation is the most commonly used analysis tool because of the complexity of the system. Most cited analyses focus on segregated systems. Agrawal and Herugu [1] do not address flow path design or vehicle routing for unified systems.

Montoya-Torres [45] also surveys AMHS design and operation, including layout design, vehicle dispatching, and idle vehicle control. It includes segregated, unified, and conveyor-based system comparisons from literature, pointing out that no one type performs best under different operating conditions. Most analyses are simulation-based and the scope of the simulations is limited. Operationally, Montoya-Torres [45] focuses on vehicle dispatching and refers to Qiu et al [52] for a survey on general automated guided vehicle (AGV) routing. It states that much of the operational policy management is proprietary.

Qiu et al [52] surveys vehicle dispatching (termed scheduling) and routing in general AGV systems. For general path topologies, it defines static, time-windows based, and dynamic methods. It defines static methods as the case where a vehicle's entire route must remain unoccupied until it completes travel, time-windows based as the case where portions of the route are reserved incrementally as the vehicle progresses, and dynamic as the case where the route is determined incrementally while the vehicle progresses. Static methods are only feasible in networks with few vehicles. Time-windows based methods are feasible for larger numbers of vehicles but the computational complexity and time becomes large as the number of vehicles increases. The dynamic routing methods presented are also computationally intense when the number of vehicles or number of tasks is large. Routing on specific topologies such as linear, loop, and mesh is discussed, but none are similar to unified AMHS layouts. Qiu et al [52] also discusses flow path layout optimization, but all included methods ignore the impact of congestion and vehicle interaction.

Le-Anh and De Koster[37] also surveys general AGV systems, including flow path design, optimizing the number of vehicles, vehicle dispatching, idle vehicle control, and vehicle routing. Most flow path design methods do not consider vehicle interaction and congestion. Lim et al[38] use a Q-learning method, which is a machine learning technique, to estimate congestion. The discussion of routing is limited and references Qiu et al [52].

1.3.1 Modeling AGV Systems

In Chapter 2, we describe the discrete-event simulation we developed to analyze routing and layout in a unified AMHS. Simulation is the most commonly used analysis tool for modeling these systems due to the complexity of the system and its uncertainty[1]. Kim et al [29, 31, 30] simulate an 18 bay unified AMHS in AutoMod to analyze dispatching and idle vehicle policies. They conclude that performance improvements can be realized by allowing vehicles to be reassigned before they have reached either origin and by implementing a continuous circulation idle vehicle control policy. Kim et al [28] presents a tool to translate CAD files into AutoMod layout files and a generic simulation framework to more efficiently generate simulation instances. Lin et al [39] propose the unified AMHS system and demonstrate its effectiveness in an eight bay simulation in *e*-M Plant. Han et al [19] compare single and double level tracks using a 22 bay simulation and conclude that the double level track provides a significant improvement in the number of moves per hour. Fukunari et al and Bahri and Gaskins [15, 5] use AutoMod to compare node-based methods for managing congestion on several layouts. Fukunari et al conclude that incorporating historical information in setting node penalties improves performance. Bahri and Gaskins conclude that the use of node-balancing reduces delivery times by up to 30%. Bahri et al [6] compare unified and segregated AMHS layouts using a simulation developed in AutoMod and conclude that unified systems provides a 32% improvement in delivery times for normal lots. Sha and Yang [57] simulate a 10 bay unified AMHS to compare transportation strategies with different objectives an conclude that improvements can be made by using more sophisticated vehicle dispatching techniques. Christopher et al [11] use a 20 bay simulation to compare dispatching policies, considering both vehicle dispatching and machine scheduling. They conclude that the significance and interaction of these factors depend upon the AMHS utilization. Simulation is also commonly used to evaluate most AGV systems and other types of AMHS layouts due to the high levels of complexity and uncertainty (e.g. [21, 42, 44, 41, 66, 36]). In the relevant literature, the vehicle routing policy is either shortest distance or not specified.

1.3.2 Routing in AGV Systems

In Chapter 3, we discuss routing methods and propose a congestion-aware dynamic routing method. Limited work has been done in AMHS vehicle routing because the transition to unified systems from segregated systems is relatively recent. In unified systems, a vehicle moves throughout the entire network as opposed to being confined to one specific area as it is in the older segregated design. Selection of routes is more complex in unified systems. In terms of congestion-aware dynamic routing, Yang et al [70] reroutes vehicles at each diverging intersection by selecting a route from a candidate set using the number of vehicles currently traveling along a route as a measure of congestion. It demonstrates a moderate improvement in steady-state metrics in a small simulation but does not discuss the occurrence of heavy congestion, deadlock, or vehicle breakdown. Patents [17] and [22] also propose route selection using pre-determined candidate route sets. Both cover cases where vehicles are and are not rerouted while in progress and do not present results. Patent [17] uses the number of vehicles currently traveling along a route as a measure of congestion and Huang et al [22] recommends the use of dynamic traffic information to compare the first, second, and third fastest routes but does not define a traffic metric. Because these are patents, they do not provide results.

In general AGV systems, Lau and Woo [36] consider edge utilization, path distance, congestion-based waiting time, and processing locations to select a route but the specific metric is not provided. Their proposed approach demonstrates improvement in cycle time, load balancing, and network robustness in a 5x5 conveyor system. Tanchoco and Taghaboni-Dutta [63] uses uncongested edge time plus estimated waiting time to cross each node. Waiting time is estimated via a queueing model but always assumes that vehicles take the shortest path to their destination. They conclude that this method is effective for small networks where traffic conditions are relatively easy to predict. Other proposed AGV routing methods include time-windows [32, 33, 23, 59, 10], incremental route planning [63], hierarchical simulation [56], petri-nets [49, 50], zone-control [20], and agent-based [36, 47] methods. Most systems are small with fewer than twenty vehicles and, thus, are difficult to implement is large systems with several hundred vehicles.

1.3.3 Layout Design in AGV Systems

In Chapter 4, we address the issue of rail layout or flow path design in AMHSs. Although recent research addresses the integration of flow path design with other factors such as equipment layout [55, 53, 18, 2] and vehicle dispatching [4], to our knowledge layout or flow path design for dynamic routing has not been considered. Most unidirectional flow path design methods do not consider vehicle interaction and congestion and most seek to minimize shortest path distance between a fixed set of origin/destination pairs [37]. Basic models that do not consider congestion include the integer programming model developed by Gaskins and Tanchoco [16] and improved by Kaspi and Tanchoco, Kim and Tanchoco, and Kaspi et al [26, 33, 25] and others. Lim et al [38] proposes a Q-learning method, a machine learning method, to estimate congestion.

Transportation planning faces similar network design questions. Integer programming multi-commodity flow models are the most common method with congestion modeled through non-linear objective functions or edge capacities. Magnanti and Wong [43] provide a survey. Most models assume that demand is fixed and does not very over time. Recent work focuses on the incorporation of uncertainty, with respect to both travel lane availability and demand. Chen et al [9] provides a recent survey on the incorporation of uncertainty. Most approaches use some form of stochastic programming. After summarizing several approaches, Chen et al [9] describes in detail a dependent change multi-objective programming model in a bi-level programming framework.

1.3.4 State-of-the-Art

While most of the operational details of AMHSs in practice are proprietary, it is clear from the literature that several areas provide opportunity for improvement. The need for dynamic routing that considers congestion in unified AMHSs has been expressed, including the existence of patents [17, 22], but limited literature addresses specific methods or demonstrates improvement via high-fidelity simulation. Fukunari et al and Bahri and Gaskins demonstrate that considering and/or balancing congestion in routing decisions reduces delivery times [15, 5]. With respect to layout or flow path design, we are not aware of any literature that considers layout effectiveness under dynamic routing in AMHS or AGV systems, although [65] stresses the importance of considering the interaction of operational and design decisions. In this thesis, we propose a specific dynamic routing method that considers congestion and is efficient enough to be used in large systems with frequent decisions. We then consider how the use of this routing method impacts the effectiveness of particular flow path layouts. In order to demonstrate the effectiveness of our routing method and illustrate the impact on layout, we use a high-fidelity simulation and associated automated layout generation tool.

CHAPTER II

SIMULATION

Semiconductor manufacturing is a highly complex and dynamic environment. Our system, as described in Chapter 1, is challenging to model because of its large scale, uncertainty, and complex vehicle movement. For this reason, we employ discrete-event simulation. Discrete-event simulation incorporates randomness and is able to mimic complex vehicle interaction. Because material handling systems are often difficult to model analytically, commercial simulation software has been developed specifically to model material handling systems.

Facilities often contain hundreds of machines and thousands of storage locations and several decisions are made each second by the production scheduling and material handling systems. Production scheduling decisions, such as when and to/from where to move parts, are made in real-time instead of determined in advance, so it is not possible to deterministically model the future course of the system given the current available information. Though stochastic methods, such as queueing models, incorporate uncertainty, they are unable to model detailed vehicle movement. In semiconductor manufacturing, vehicles are large relative to edge size and stopping locations are closely spaced making the use of a queueing model challenging.

Vehicle movement and interaction in our system are complicated by physical system constraints. Vehicles must accelerate and decelerate when exiting and entering curved edges, behind slow moving or stopped vehicles, and to stop to load or unload. Loading and unloading occur on track so vehicles often delay one another for the duration of loading/unloading time. Because there are no parking locations, unassigned or idle vehicles must be moved to allow active vehicles to pass.

In this chapter, we present a detailed view of the high-fidelity discrete-event simulation we use to evaluate vehicle routing policies. Updates to the simulation for use when considering layout are discussed in Chapter 4. Because we are interested primarily in vehicle routing policies, we simulate vehicle movement in detail. We incorporate production scheduling decisions through a production-based request generation method that mimics realistic demand patterns. First, we provide an overview of the simulation, followed by detailed discussions of the prototype facility layout, operational policies that govern vehicle movement, the request generation method, simulation of vehicle breakdown, metrics that we record, and how we determine run control parameters.

2.1 Overview

The simulation was developed in AutoMod simulation software and simulates the movement of vehicles in a prototype facility. Vehicles are assigned in real-time to transfer requests consisting of an origin and a destination. The user selects the number of vehicles and parameters associated with request generation, vehicle movement, vehicle breakdown, metric tracking, and run control. Figure 2.1 shows a snapshot of the simulation while running. Green vehicles are moving to pick up a request, blue vehicles are moving from origin to destination, red vehicles are stopped to load or unload, and black vehicles are unassigned.

2.2 Layout

We consider a prototype facility based on guidance from a large electronics manufacturer. Though they have significant variability between facilities, our prototype represents a typical facility. In Chapter 4, we present a layout generation tool that allows the evaluation of thousands of different layouts by automating the generation of simulation files. We also discuss updates to the simulation for that phase of analysis. In this chapter, we focus on the prototype facility and simulation as originally

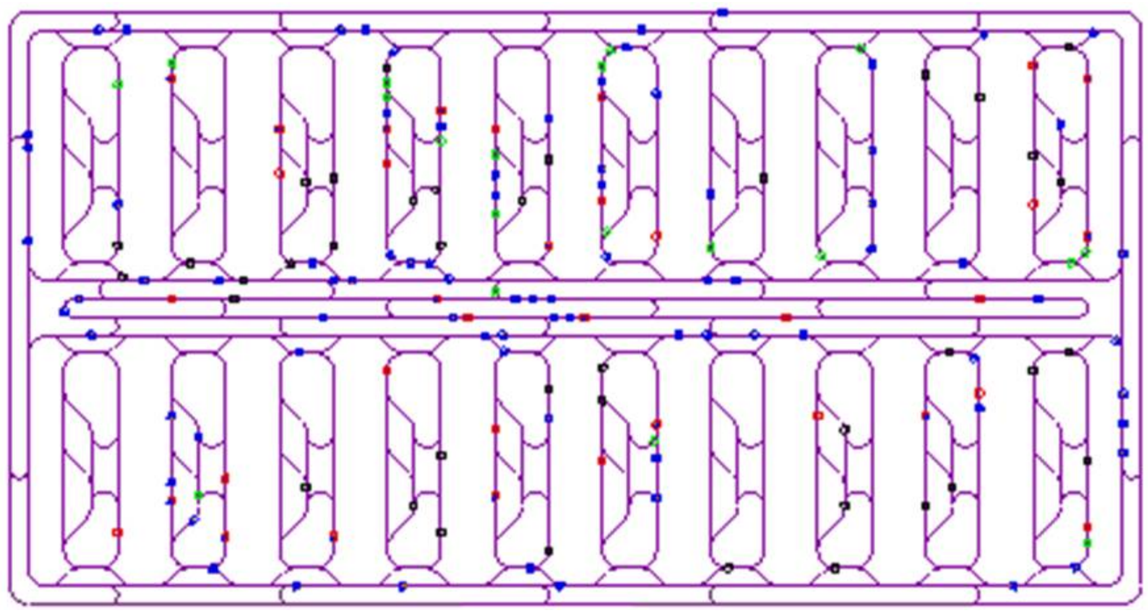


Figure 7: Simulation in Action

developed.

2.2.1 Rail Configuration

We simulate a unified AMHS with the layout shown in Figure 2.2.1. It comprises 20 bays aligned in two parallel rows, a center loop, and an outer loop. All rails are unidirectional. Each bay has two entrances and two exits and vehicles travel throughout the facility. The center loop has four lanes with alternating travel directions. In the figure, blue Xs represent stopping locations and red Xs represent intersections between rails. We made several updates in the second phase of our work to make the rail layout more realistic. See Chapter 4 for discussion.

2.2.2 Stopping Locations

In the prototype layout, each bay has 60 stopping locations and an additional 240 are spread throughout the middle two lanes of the center loop. Inside bays, stopping locations represent ports and storage locations called side-track buffers, we do not distinguish between the two. Along the center loop, they represent storage locations

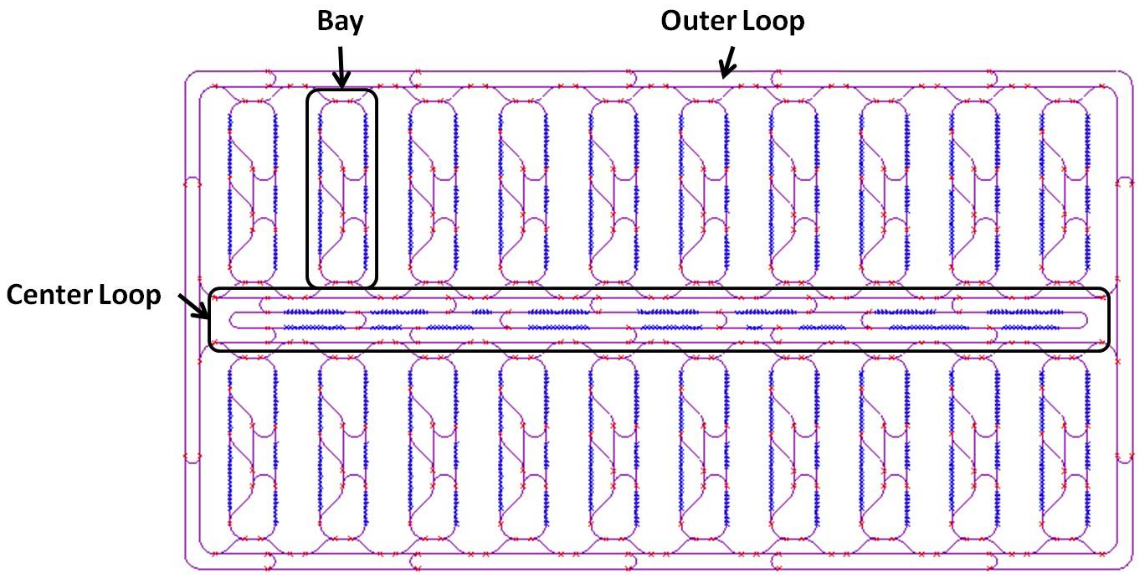


Figure 8: Prototype Facility Layout

called stockers. In the second phase, the number of stocker locations on the center loop is reduced and stopping locations associated with bays are introduced near bay entrances and exits on the center loop.

2.3 Transfer Request Generation

Demand in the simulation (and the real system) is generated through transfer requests. In the real system, transfer requests are generated by the production scheduling system which is, in itself, highly complex. It considers factors such as machine utilization, work-in-process (WIP) balancing, and lot prioritization. Because our focus is vehicle routing not scheduling, we model transfer requests in a simplified manner with a focus on incorporating factors that are likely to impact routing decisions and effectiveness. Primarily, we want the demand patterns or pattern of origin/destination pairs to mimic what is seen in the real system and for congestion to occur similarly to what is experienced in practice.

We model the impact of production scheduling through a request generation method based on production-based probability distributions. This method does not require the simulation of a wafer through its entire production sequence or detailed knowledge of the production scheduling process. While our method is an approximation, we have built in the flexibility to adjust several parameters in order to allow extensive sensitivity analysis. In the case that production information is not available, we also provide the option to use uniform distributions instead.

A transfer request consists of an origin location, a destination location, and a time. In the simulation, we use an infinite loop that randomly selects a value for the time until the next request and then waits that amount of time before repeating. After waiting the specified amount of time, an entity is created representing a request and an origin and destination are assigned. We provide three options for controlling the timing of requests and origin selection and two options for destination selection. We first discuss destination selection. In production-based methods, we consider two product types, A and B, and assume a probability of 0.5 is assigned to each. In the rest of this description, we discuss only one product type but both are handled in the same manner.

2.3.1 Destination Selection

We offer two options for destination selection. In one case, the destination does not depend upon the origin and we select it uniformly across all stopping locations. In the second method, we first select the destination process type, then the bay, then the port. The destination will depend on the origin selection. We consider eight processes and assign each bay to a particular process as shown in Figure 2.3.1. All center loop locations are assigned to a dummy process type STOCKER.

Process type selection is based on a probability distribution we call the transition matrix T. It specifies, for a particular origin type, the probability that the request destination will have a particular type. Rows represent the origin process and columns the destination process. Figure 2.3.1 shows an example of a transition matrix. Values

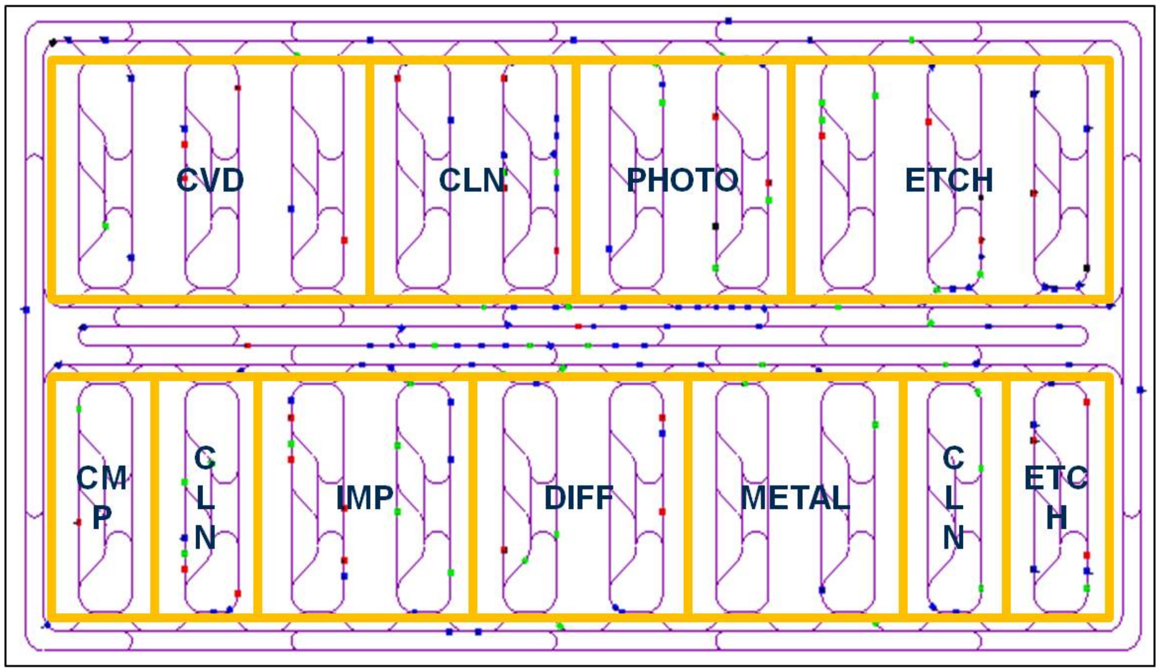
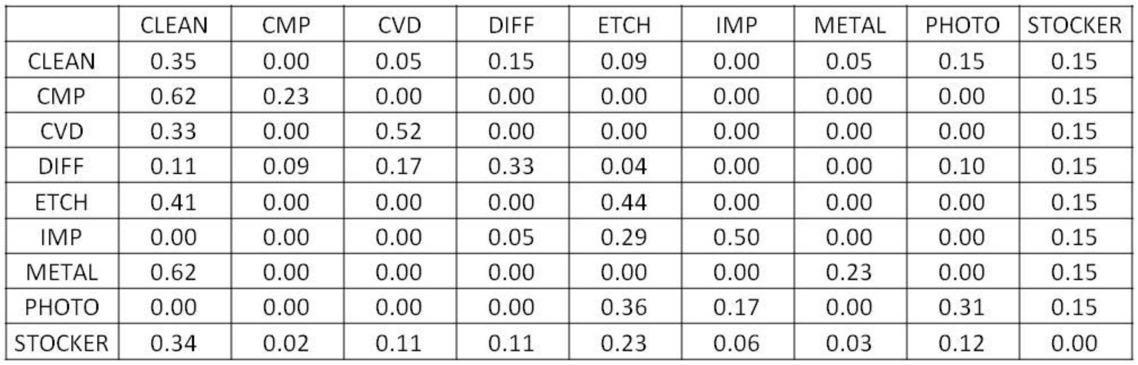


Figure 9: Bay-to-Process Assignment

are only shown to the second decimal point. In this matrix, if the origin is of type CLEAN, the destination has a 35% chance of being of type CLEAN, a 5% chance of being of type CVD, a 15% chance of being of type DIFF, a 9% chance of being of type ETCH, a 5% chance of being of type METAL, a 15% change of being of type PHOTO, and a 15% chance of being a STOCKER.



	CLEAN	CMP	CVD	DIFF	ETCH	IMP	METAL	PHOTO	STOCKER
CLEAN	0.35	0.00	0.05	0.15	0.09	0.00	0.05	0.15	0.15
CMP	0.62	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.15
CVD	0.33	0.00	0.52	0.00	0.00	0.00	0.00	0.00	0.15
DIFF	0.11	0.09	0.17	0.33	0.04	0.00	0.00	0.10	0.15
ETCH	0.41	0.00	0.00	0.00	0.44	0.00	0.00	0.00	0.15
IMP	0.00	0.00	0.00	0.05	0.29	0.50	0.00	0.00	0.15
METAL	0.62	0.00	0.00	0.00	0.00	0.00	0.23	0.00	0.15
PHOTO	0.00	0.00	0.00	0.00	0.36	0.17	0.00	0.31	0.15
STOCKER	0.34	0.02	0.11	0.11	0.23	0.06	0.03	0.12	0.00

Figure 10: Transition Matrix Example

The transition matrix is created using a prototype production sequence and parameters that specify the probability that a cartridge visits a side-track-buffer p_{STB}

or stocker p_{STK} between process steps. We use two 400-step prototype sequences and assume a 20% chance of visiting a stocker and at 30% change of visiting a side-track-buffer. To calculate the transition matrix, we first create a process-only transition matrix T_P with the eight process types from the production sequence. Cell (i, j) in the process-only transition matrix is calculated by counting the number of times that process type i is followed by process type j in the production sequence and dividing by the total number of times process type i occurs. The last step in the process sequence is excluded since there is no next step. We also calculate the overall process frequency distribution P by counting the total number of times each process occurs and dividing by the total number of steps.

To calculate cell (i, j) in T, we use:

$$T(i,j) = \begin{cases} \dfrac{(1 - p_{STK})T_P(i,j)}{(1 + p_{STB})}, & i \neq j, \quad i,j \neq STOCKER \\ \dfrac{(1 - p_{STK})T_P(i,j) + p_{STB}}{(1 + p_{STB})}, & i = j, \quad i,j \neq STOCKER \\ \dfrac{p_{STK}}{(1 + p_{STB})} & i \neq STOCKER, \quad j = STOCKER \\ P(j) & j \neq STOCKER, \quad i = STOCKER \\ 0 & i,j = STOCKER. \end{cases}$$

The $(1 - p_{STK})T_P(i, j)$ terms in the numerator when i and j do not represent stockers represent cartridges that go from a machine of one type to a machine or side-track-buffer of another type. The p_{STB} term in the numerator represent cartridges that go from a side-track-buffer of one type to a machine of the same type. When j represents a stocker, p_{STK} represents cartridges that go from machines or side-track-buffers to stockers. The denominator $(1 + p_{STB})$ represents all cartridges that go from machine or side-track-buffers of one type to anywhere else. When i represents a stocker, we use the overall process frequency distribution P except when j also represents a stocker. We do not allow stocker to stocker requests so $T(STOCKER, STOCKER) = 0$.

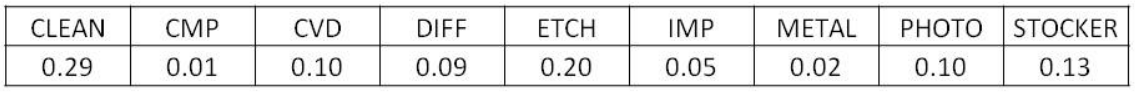
Once the destination process type has been selected, a specific bay is selected randomly from bays of that type. Stocker bays in the center loop are divided into four dummy bays for this purpose. The stopping location is then selected randomly from among locations in that bay. Both bay and stopping location selection use uniform distributions.

2.3.2 Origin Selection

For origin selection, we include three methods. In the first method, the origin is selected from a uniform distribution across all network locations. In the second method, the origin process type is selected first, followed by the bay and stopping location. In the second method, the origin process is selected from a stationary distribution S derived from the transition matrix T described previously. Because the transition matrix is of the form used in Markov Chains, the stationary distribution can be calculated using $\pi = \pi T$ where π is the stationary distribution and T is the transition matrix. By nature, our matrix is irreducible and positive recurrent, so a stationary distribution exists. This stationary distribution represents the long-run average number of transitions that begin (and end) in each process. The stationary distribution S can also be calculated directly using P as:

$$S(i) = \begin{cases} \dfrac{(1 + p_{STB})P(i)}{(1 + p_{STB} + p_{STK})}, & i \neq STOCKER \\ \dfrac{p_{STK}}{(1 + p_{STB} + p_{STK})}, & i = STOCKER. \end{cases}$$

The stationary distribution associated with T, $p_{STB} = .3$, and $p_{STK} = .2$ is shown in Figure 2.3.2.



CLEAN	CMP	CVD	DIFF	ETCH	IMP	METAL	PHOTO	STOCKER
0.29	0.01	0.10	0.09	0.20	0.05	0.02	0.10	0.13

Figure 11: Stationary Distribution Example

In the third method, we more closely enforce flow balance in each bay. Over time, inflow and outflow should be equal or balanced. To do this, we define *net deliveries* d as the number of items in a bay that are available to be transported (not currently involved in active requests). It is calculated as the total number of deliveries minus the total number requests assigned with origin in that bay. After an initialization period, the origin bay for a request is randomly selected from the distribution of net deliveries across all bays. Once the origin bay has been selected, a stopping location is assigned randomly across all locations in that bay using a uniform distribution. In the simulation animation, we track net deliveries on screen as shown in Figure 2.3.2. Net deliveries for each bay are shown above the layout. Note that the highest number correspond to bays (see Figure 2.3.1) associated with the highest demand in the stationary distribution in Figure 2.3.2. Numbers inside each bay represent the number of origins and destinations currently assigned to that bay and can be used for analysis.

2.3.3 Request Timing

The time between requests is selected from an exponential distribution with either a constant mean or a mean based on the current number of net deliveries as well as an input parameter r. With the first or second origin selection method, the mean is constant. With the third origin selection method, request timing may also depend upon net deliveries. In this way, if vehicles are prevented from delivering a request due to congestion, fewer new requests will be generated with an origin in that bay. This more closely mimics behavior in the real system. In this case, when time until the next request is generated, we calculate the mean μ for the exponential distribution as $\mu = \frac{d}{r}$. If the system is in steady-state, the value of d varies but not excessively. r is an input parameter that must be set considering the number of vehicles and the demand pattern. Often we target an 80%-90% vehicle utilization and determine an

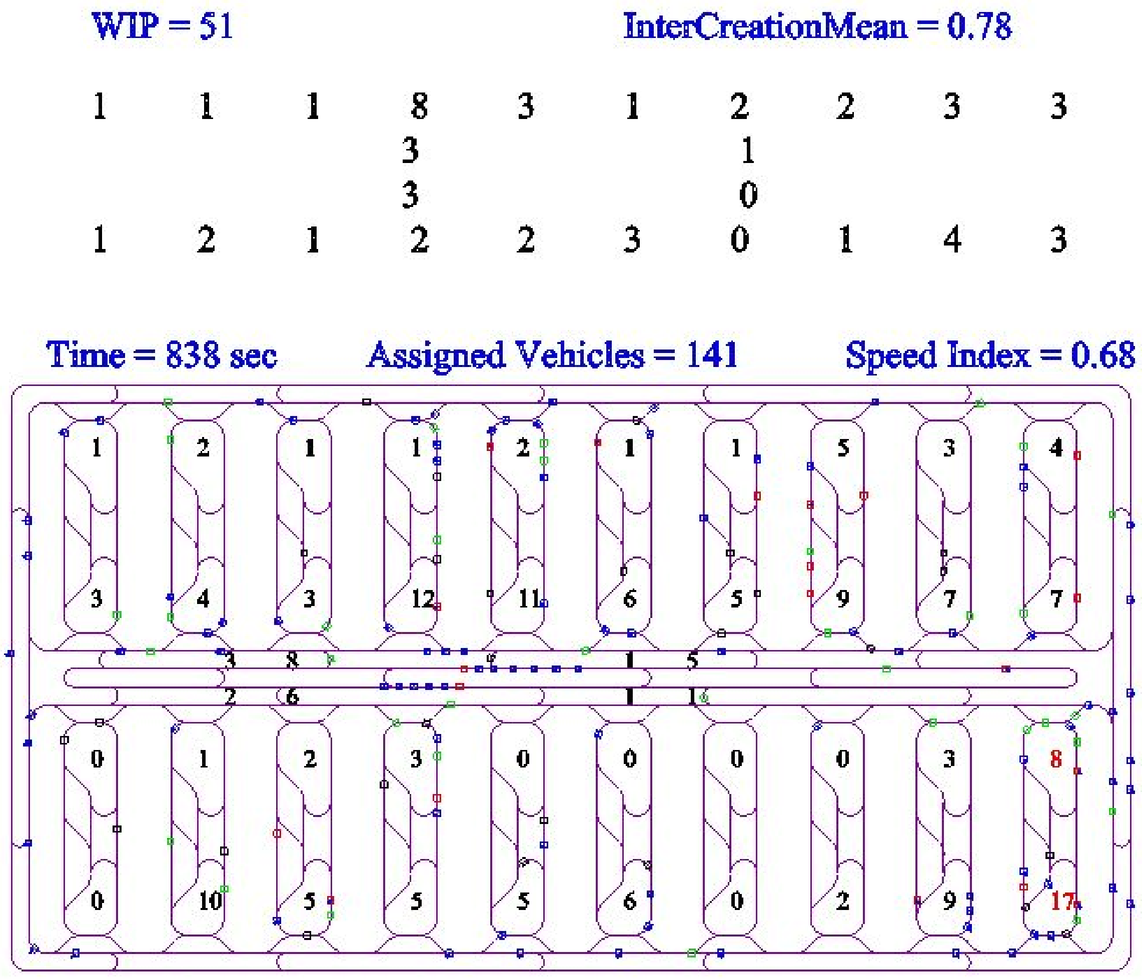


Figure 12: Net Deliveries Tracking Example

appropriate value of r by iterative trial-and-error. If d and r are not set appropriately, the system may become overloaded and never to able to recover or may be excessively under-utilized.

To achieve an appropriate steady-state value of d, we run an initialization period distinct from (and shorter than) the statistical warm-up period. During this time, the mean used for request generation is constant and net deliveries build up in the system because we do not allow the number of net deliveries in any bay to drop below zero. After the initialization period, we switch to request timing based on net deliveries and the system enter steady-state, if possible, after an adjustment period. We sometimes refer to the parameters that control request timing as workload parameters because they determine how heavily the system is used.

2.4 Vehicle Control

In discrete-event simulation, the system progresses over time by moving from one scheduled event to the next. When one event occurs, it may trigger the scheduling of another event. For example, when a vehicle arrives at its origin an event will be scheduled that indicates that the vehicle completes pickup. The scheduling of events associated with vehicles is controlled by operational logic and associated parameters. Operational logic includes vehicle assignment, vehicle routing, vehicle movement, loading/unloading, and idle vehicle control.

2.4.1 Vehicle Assignment

When a transfer request is generated, the system identifies the available vehicle with the earliest predicted arrival time by running a one-to-all shortest path calculation using Dijkstra's algorithm on the reverse network (i.e. arcs directions are reversed). Available vehicles include those not currently assigned to a request and those currently delivering at their destinations. If a vehicle is currently delivering, the time remaining until it complete delivery is included in the calculated arrival time. If no vehicle is available, the system adds the request to a queue for requests awaiting assignment.

2.4.2 Vehicle Routing

We include static and dynamic routing with either distance-based edge weights or edge weights that are updated via exponential smoothing each time a vehicle traverses an edge. Both dynamic routing and our edge weight updated procedure are described in detail in Chapter 3. With static routing, the system calculates a vehicle's routes from current location to origin and origin to destination at the time of the request based on current edge weights. The vehicle follows this path regardless of what happens in the system while en route. With dynamic routing, the system maintains node-based lookup tables that are periodically updated based on current information. When a vehicle approaches a diverging node, it looks up its next node in the lookup table based on its destination. For updating edge weights, the user specifies a weight λ to be used in exponential smoothing. For dynamic routing, the user specifies the table update interval.

2.4.3 Vehicle Movement

Much of the complexity of vehicle movement resulting from acceleration and deceleration and intersection control. A vehicle's maximum velocity is lower on curved edges than on straight edges so a vehicle must decelerate to enter a curved edge. Any following vehicles must also decelerate. AutoMod controls this logic by user specified maximum velocities and acceleration/deceleration rates. We use a straight edge maximum velocity of 3 m/s, a curved edge maximum velocity of 1 m/s, and rate of acceleration of 2 m/s, and a rate of deceleration of 3 m/s. These values may be changed for sensitivity analysis. Vehicles also must maintain a 1m separation between them. Intersections are first-in-first-out (FIFO) and a vehicle may continue through an intersection at maximum velocity if it is not entering a curved edge or slowing for other reasons.

2.4.4 Loading/Unloading

We use a deterministic loading and unloading time of 10 seconds. All loading and unloading occurs on-track and other vehicles are not allowed to pass a stopped vehicle. A vehicle located on an edge behind a stopped vehicle may only be rerouted if there is a diverging node between this vehicle and the stopped vehicle.

2.4.5 Unassigned Vehicle Policy

A vehicle that is not assigned to a request remains stationary until an active vehicle requests it to yield. Active vehicles look ahead in order to request stationary vehicles to yield before decelerating behind the stationary vehicle. When requested to yield, the unassigned vehicle moves forward to the next diverging intersection and follows the edge opposite that which the calling vehicle will take. If the number of unassigned vehicles in a bay exceeds a threshold, a vehicle requested to yield will be redistributed to a bay with fewer unassigned vehicles.

2.5 Vehicle Breakdown

The system's reaction to and ability to recover from exceptions such as vehicle breakdown are also important. For this reason, we provide the ability to simulate vehicle breakdown. The location of a breakdown can either be selected from anywhere in the network or can be restricted to the center loop. The time between breakdowns is selected from an exponential distribution. We use 2700s as the mean. The breakdown is either short or long, determined randomly based on an input parameter that specifies the probability that it is a long breakdown. Typically, we use 25%. If it is a short breakdown, the actual length of time is selected randomly from a uniform distribution with specified range. We use a range of 90-180s. If the breakdown is long, it lasts a deterministic 600s.

2.6 Metric Tracking

Several different types of metrics are of interest and the simulation collects a variety of metrics as well as detailed tracking information that can be used for post-analysis. Detailed tracking can be turned on or off because it significantly increases run time. We categorize metrics into request-based metrics, recorded for each request, and time-based metrics, recorded at a fixed time interval. Time-based metrics reflect the system state at each point in time. For ease of analysis, metrics frequently used in analysis are averaged (either over all requests or all time points) and output in a separate file. For these summary metrics, requests generated during the statistical warm-up period and time points in the statistical warm-up period are excluded.

2.6.1 Request-Based Metrics

We record request-based metrics at the time a request completes delivery. Several types of metrics are important in evaluating system performance. Of primary interest is vehicle travel time, including total time as well as the individual retrieval and delivery components. For our purposes, retrieval time is defined as the time from when a vehicle is assigned to the request to the time pickup at the origin is complete and delivery time is defined as the time from when a vehicle completes pickup to the time it completes delivery at the destination. Retrieval time depends upon vehicle utilization because if fewer vehicles are unassigned, the assigned vehicle is likely to be further away from the origin at the time it is assigned. For this reason, we often focus on delivery time as a measure of the effectiveness of a vehicle routing method. In the case where vehicle utilization reaches one, we also record the time a request waits to be assigned a vehicle.

We are also interested in travel delay, or the difference between travel time and a lower bound on travel time. For the prototype facility, we calculate lower bound travel times using edge length, maximum velocity, and acceleration/deceleration required when entering and exiting curved edges. We record lower bound for retrieval, delivery, and total travel times. We also attempt to break down delay into in-bay delay caused by loading and unloading vehicles, center loop delay caused by loading and unloading vehicles, delay caused by a vehicle breakdown, and other delay. For these delay metrics, only delay when a vehicle is fully stopped is considered.

In summary form, we record travel times, the difference between travel time and lower bound time, the ratio between travel time and lower bound time. Having these metrics for each load, we can also do more detailed analysis of the variability in travel times and delay including calculating standard deviations and generating histograms for the associated distributions.

2.6.2 Time-Based Metrics

In addition to measuring in impact of policy changes on individual requests, we want to understand the impact of system state. For this, we record various system-wide metrics at a predefined interval, typically 5s. Of importance are speed index and vehicle utilization. Speed index for an individual vehicle is defined as the ratio of the current velocity to the maximum velocity given location in the network. At a particular point in time, we record the average speed index across all vehicles that should be moving, i.e. are not loading, unloading, or idle and stationary. Speed index measures how smoothly traffic is flowing in the network, with a higher speed index indicating less congestion. Vehicle utilization is the ratio of number of vehicles assigned to a request divided by the total number of vehicles. All else equal, lower travel times result in a lower vehicle utilization. From another perspective, if the vehicle utilization goes down due to lower travel times, the workload and thus transportation throughput are increased.

We also collect several metrics for use in detailed analysis including the current number of net deliveries, the current mean time between requests, the number of slow moving and stopped vehicles, and the average current vehicle velocity of those vehicles that should be moving. Viewed graphically over time, these metrics help in understanding the progression of the simulation over time. For example, since speed index is used to define heavy congestion, viewing the speed index one can immediately see when the system starts to experience a phase change into heavy congestion.

2.6.3 Detailed Tracking

Detailed tracking allows extensive analysis of the progression of the simulation over time. We provide the option to track detailed vehicle movements and detailed edge usage. If either of these options is turned on, we collect information in a form that allows flexible post-analysis. For vehicle tracking, each time a vehicle reaches a node, we records the time and request that the vehicles is carrying (if applicable). For edge tracking, we report the average edge traversal time for each edge in the network. This information can also be computed from detailed vehicle tracking, but we provide the option to output it directly for ease of analysis. Both options extend simulation time significantly.

2.7 Run Control

2.7.1 Warm-Up Period and Simulation Time

We establish a warm-up period based on speed index and vehicle utilization in an initial set of replications of distance-based static routing. Speed index is defined as the average over all vehicles that should be moving (i.e. not stopped for loading/unloading and not idle and stationary) of the current velocity divided by the maximum velocity given a vehicle's location in the network. We graphically determine when the system reaches steady-state. We use a total simulation run time of eleven times the warm-up period (ten times plus the warm-up). In our analyses, we use a warm-up time of 1200s and a total simulation time of 13200s.

2.7.2 Number of Replications

To determine a sufficient number of replications for steady-state analysis, we run twenty initial replications. We remove those replications that reach heavy congestion and calculate standard deviations on key metrics in the remaining replications. If they are sufficiently low, we use these replications for steady-state analysis. In our simulation, few replications often result in low standard deviations. Even though fewer than twenty replications may be sufficient, we always run at least twenty replications in order to be able to compare the frequency of heavy congestion. We discuss the impact of the number of replications on the statistical significance of specific results in Chapter 3.

CHAPTER III

ROUTING

In this chapter, we address the problem of vehicle routing in the unified automated material handling system (AMHS) described in Chapter 1. In this system, requests occur online or in real-time and specify that a cartridge is ready to be transported from an origin to a destination. We assume that the vehicle with the earliest predicted arrival time is assigned to the request and consider the question of how to determine and control the route a vehicle takes from its current location to the request origin and from the request origin to the destination.

This problem is challenging due to the large scale of the network, the frequency with which decisions are made, the uncertainty associated with online system, and the complexity of vehicle interaction. In particular, computational efficiency is of utmost importance. In this chapter, we first discuss the general category of routing problems followed by potential approaches to our problem and a selected approach. We follow this with a simulation-based analysis of our proposed approach that demonstrates significant improvement over existing methods.

3.1 Routing Problems

Routing problems are widespread in industrial engineering and operations research with applications in transportation, logistics, supply chain, manufacturing, telecommunications, and network optimization. In a routing problem, a set of cars, trucks, trains, automated guided vehicles, data, or electronic signals is routed through a network such that a set of requests is satisfied in a way that minimizes some objective function (i.e. cost, time, distance). The meaning of the term request varies by context but may include visiting delivery, pickup, or customer service locations, driving from an origin to a destination, transporting a person or load from an origin to a destination, or routing an electronic data packet from an origin to a destination.

3.1.1 Problem Definition

The term routing problem covers a wide variety of contexts and assumptions. In some cases, a route refers to a sequence of customers that a vehicle will visit such as in a delivery or service route. These problems are sometimes called tour selection problems and are usually modeled via the standard Vehicle Routing Problem (VRP) and its variants. We focus on routing problems sometimes called path selection problems where a route is a series of edges connecting an origin and a destination. In our discussion, we use the terms path and route interchangeably.

A routing problem can be defined on a graph $G = (N, E)$ where N is a set of nodes representing locations and E is a set of edges representing travel lanes connecting nodes. N includes both stopping locations N_S and travel lane intersections N_I. We use the term stopping location to refer to delivery, pickup, or customer service locations or transport origins and destinations. Intersections represent physical locations where travel lanes cross, merge, or diverge. Edges in E may be directed (i.e. one-way travel) or undirected (i.e. two-way travel). For each edge $e \in E$, there is a weight w_e that represents travel distance, time, or cost. In some instances, w_e may be a function of other parameters and variables (e.g. the amount of traffic on the edge). The objective of the routing problem is usually to minimize the total time, distance, or cost of a solution.

A set of requests R is defined as a set of origin/destination pairs $R = \{(o, d)\}$ where $o \in N_s, d \in N_s$ such that a load must travel or be transported from origin to destination. In some cases, additional information such as customer demand or release times are also associated with each request. Requests may be known in advance or may be revealed over time. If known in advance, the problem is called offline. If revealed over time, it is called online. Online problems present unique challenges because the system must either be re-optimized each time a new request arrives or the new request should be routed assuming existing routes are fixed. Also, because future requests will affect the realized travel time of current requests, it is not possible to guarantee optimality.

3.1.1.1 Shortest Path Problems

The simplest path selection model is the shortest path problem (SPP) of finding the shortest route from a node s to another node t, which is defined on the graph G as:

$$\min_{P \in \mathcal{P}_{st}} \sum_{(i.j) \in P} w_{ij} \tag{1}$$

where $\mathcal{P}_{st}$ is the set of elementary paths from s to t. When all w_{ij} values are independent of s and t, we can solve the path selection problem between each pair of nodes independently. Dynamic programming algorithms such as Dijkstra's [12] or Bellman-Ford [8] are typically use to solve this problem. On networks, such as ours, with non-negative costs and no negative cycles, Dijkstra's algorithm can be implemented with complexity $O(|V| \log |V| + |E|)$ [60] and Bellman-Ford with complexity $O(|V||E|)$ [7]. The one-to-all shortest path problem where one origin has many destinations is also efficiently solved by these algorithms. All-to-all (or some-to-some) shortest path problems are typically either solved by solving one one-to-all problem for each origin or by specialized algorithms such as Floyd-Warshall [14]. [67] proposes an algorithm particularly applicable for the some-to-some problem.

3.1.1.2 Mutli-Commodity Flow Problems

The one-to-one, some-to-some, and all-to-all path selection problems can be formulated as mathematical programs. They are a type of uncapacitated multicommodity flow problem (MCFP) defined on the graph $G = (N, E)$ and a set of origin/destination pairs $R = \{(o, d)\}$ where $o \in N_s$ and $d \in N_s$.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(o,d)\in R} \sum_{(i,j)\in E} w_{ij} x_{ij}^{od} & & \\
\text{subject to} \quad & \sum_{j|(i,j)\in E} x_{ij}^{od} - \sum_{j|(j,i)\in E} x_{ij}^{od} = 1 & \forall i \in N, (o,) \in R | o = i & \qquad (2) \\
& \sum_{j|(i,j)\in E} x_{ij}^{od} - \sum_{j|(j,i)\in E} x_{ij}^{od} = -1 & \forall i \in N, (o, d) \in R | d = i & \qquad (3) \\
& \sum_{j|(i,j)\in E} x_{ij}^{od} - \sum_{j|(j,i)\in E} x_{ij}^{od} = 0 & \forall i \in N, (o, d) \in R | o \neq i, d \neq i & \qquad (4) \\
& x_{i,j}^{od} \in \{0, 1\} & \forall (i, j) \in E, (o, d) \in R &
\end{aligned}
$$

Constraints (2), (3), (4) ensure preservation of flow and demand satisfaction. Variables are binary with the variable x_{ij} taking the value 1 if a request (o, d) uses edge (i, j). If $|R| = 1$, this is equivalent to the $s - t$ shortest path problem. If $|R| > 1$, the problem can be decomposed into $|R|$ independent s-t shortest path problems.

3.1.1.3 Modeling Vehicle Interaction and Congestion

In practice, applications involving path selection problems are challenging to model due to the complexity of vehicle interaction. While simple shortest path and multi-commodity flow models consider only origin to destination routes, physical routing problems typically require the consideration of empty vehicle movement, both the route an assigned vehicle takes to the request origin and the vehicle's behavior when not assigned to a request. Physical systems also require the consideration of vehicle interaction. Because two vehicle cannot occupy the same location at the same time, they necessarily impact one another. This interaction is typically modeled either through collision-avoidance or congestion-impacted travel times. Collision-avoidance methods model the movement of vehicles precisely in a time-dependent network. Time-dependent networks are described in the next subsection. Congestion-impacted travel time methods model the impact of vehicle interaction or congestion on edge traversal times. Congestion may also be limited through the use of edge capacities.

The classes of models described previously models the movement of vehicles as a constant flow. In the multi-commodity flow model, capacities may be added to edges to constrain the amount of flow allowed on each edge. If an edge reaches its capacity, some of the flow will be forced to use an alternate path. Edge weights may also be used to model vehicle interaction, for example, making w_{ij} a function $w_{ij}(x_{ij})$ of flow on the edge. In these cases, the path selection problem becomes nonlinear and cannot be decomposed into several independent shortest path problems. In addition, it is often difficult to determine an appropriate edge weight congestion model, particularly in system where vehicles are large relative to edge size.

3.1.1.4 Time-Dependent Routing

Time-dependent networks may be time-expanded or incorporate time-windows on edges or nodes. In a time-expanded network, the graph G is copied such that each node is repeated for each time interval and an edge e connects two nodes i_{t_1} and j_{t_2} if it takes $t_2 - t_1$ time units to travel from i to j. An edge from i_{t_1} to i_{t_1+1} represents a holding action at i. Edge capacities are used to limit the number of vehicles on an edge at one time. Sometimes the graph is transformed such that each edge may be traversed in one time interval so edges only connect nodes where $t_2 - t_1 = 1$. Because G is copied for each time interval, the size of the time-expanded graph depends upon the granularity of the time intervals and the length of the time horizon. To model a system with high-fidelity requires relatively small granularity and, thus, a large network.

Time-windows based methods incorporate reserved time-windows on each node or edge in G and impose a limit on the number of vehicles on the edge/node at one time. For discussion, we assume time windows are on edges. Typically, the graph is discretized such that only one vehicle may occupy an edge at a time. This approach is most common when new routes are determined with respect to existing time windows as opposed to optimizing the whole system at once. When a new route is determined, if the next edge in a potential route is scheduled to be occupied at the time the vehicle would like to enter, the vehicle must wait and enter once the edge becomes free. A modified shortest path algorithm can be used to solve this problem, but it is more computationally intense than the basic form. If waiting is not allowed at nodes, it becomes NP-Hard [58].

Time-dependent methods are computationally intense, making them difficult to implement in systems where system status changes frequently. They are also difficult to use in systems where vehicles are not allowed to preemptively wait at intersections because these methods typically assume that a vehicle can wait until its scheduled time to enter an edge. In reality, if the edge is unoccupied upon arrival, the vehicle will enter immediately. To keep the model and the real system in sync over time, scheduled routes must be continually updated which can be computationally and data intensive. Though time-dependent methods have been used in the literature to model AGV systems [32, 33, 23, 62, 48, 61, 64, 59, 10], in most cases the physical network is small and the number of vehicles is fewer than twenty.

3.2 Dynamic Routing Methods

To address the specific problem of vehicle routing in a unified AMHS, we need a method that is computationally efficient enough to handle frequent requests in a large network, allows routes to change while in progress, and considers the impact of vehicle interaction in selecting routes. We care both about changes in congestion due to vehicle interaction and about predicted vehicle interaction. We consider seven possible approaches and select the one that meets our requirements. We assume that each request is routed immediately when it arrives unless no vehicles are available. It would also be possible to route new requests in batches of a fixed size or at fixed intervals. We determine that time-dependent networks are too computationally intense for our purposes and non-dynamic methods are not responsive enough.

3.2.1 Static Shortest Path

In a static shortest path approach, a vehicle's route is selected via a shortest path algorithm at the time of the request. The vehicle follows this route regardless of changes in network status as it progresses. Edge weights used in route calculation may be based on edge length and maximum velocity, based on historical average edge traversal time, updated with recent edge traversal times, or predicted based on future routes. We call these edge weights *Distance-Based*, *Long-Run Average*, *Updating*, and *Predictive*, respectively. In this thesis, we consider the first three. We use Distance-Based edge weights as a baseline, representing current practice. Long-Run Average edge weights incorporate regular delay such as that due to acceleration and deceleration at curved edges and places with frequent congestion but they do no adjust to changes in system status. Updating edge weights capture both long-run delay and changes in system status. The sensitivity to new information is determined by an input parameter. More detail on this parameter can be found in Section 3.3. We do not consider Predictive edge weights at this time, due to the difficultly of making accurate predictions and the computational requirements of these methods. Predictive edge weights may consider the future routes of existing requests and/or the probability of new requests occurring in various locations.

3.2.2 Dynamic Shortest Path

In our dynamic shortest path approach, a vehicle evaluates its route at every diverging intersection using routing tables periodically updated using current edge weights. With Updating edge weights, this means that routes may change at each routing table update. For computational tractability, rather than the vehicle re-computing a shortest path algorithm, we use a node-based lookup table method where the vehicle looks up its next node in a table associated with the diverging node based on its destination. More detail on this approach can be found in Section 3.3. If Distance-Based or Long-Run Average edge weights are used with this method, it is equivalent to static shortest path because edge weights do not change over time.

3.2.3 Static K-Shortest Path

A set of candidate paths may be used instead of recomputing a single shortest path each time a request comes in. The candidate set is generated a priori using Distance-Based or Long-Run Average edge weights. At the time of a request, each route in the candidate set is evaluated based on current edge weights. This is often computationally more efficient than static shortest path, but does not allow routes to change while in progress. In terms of performance, with Distance-Based or Long-Run-Average edge weights, this method is equivalent to static shortest path. Dynamic k-shortest path may also be considered where a vehicle re-evaluates its route at every diverging node, but this is computationally inefficient since even the static version is too time consuming for our application.

3.2.4 Randomized K-Shortest Path

Instead of using edge weights that change over time to select a path from a candidate set, we may randomly select a path from that set. In this case, the system is not able to respond to congestion, but traffic may be better distributed throughout the network. In implementing this method, one must be careful to select only candidate routes that are not excessively long. The number of paths that are of reasonable length may vary significantly by origin/destination pair.

3.2.5 Static Time-Expanded Shortest Path

Another approach is to model the movement of vehicles exactly through the use of a time-expanded network. The network may either be discretized over time such that there is a node for each location at each time step or each node or edge may have time windows associated with it. The network must be discretized in a way that each edge (or node) may only be occupied by one vehicle at a time. At the time of a request, a vehicle selects a route using either a shortest path algorithm or a candidate route set, ensuring that it does not occupy a location at the same time as anther vehicle. These methods are both computationally expensive and difficult to implement in practice due to the complexity of vehicle movement. Particularly in FIFO (first in, first out) networks, where it is not possible to stop a vehicle at an intersection preemptively, vehicles will not progress along their routes as modeled. Thus, for the method to be effective, the location of a vehicle and the timing of its future routes must be periodically synchronized with the real system. Dynamic versions of theses methods are also possible where the entire system is frequently re-optimized, but this is computationally inefficient since even the static version is too time consuming.

3.2.6 Pre-Computed Multi-Commodity Flow

A static multi-commodity flow approach allows the consideration of predicted vehicle interaction even with pre-computed routes. Using Distance-Based or Long-Run Average edge weights, a multi-commodity flow model with binary flow variables selects a route for each origin/destination pair based on expected average demand. Vehicle interaction is considered through edge capacities or the objective function. Example of objective functions that consider vehicle interaction are minimizing the maximum edge flow or a piecewise linear function of edge flow. Computational efficiency of this method will vary depending on the objective function and constraints, but since the computation is only done once it is not a problem. This method does not respond to the changing congestion status, however, and does not allow routes to change while in progress.

3.2.7 Dynamic Multi-Commodity Flow

Dynamic versions of multi-commodity flow models may be considered where a multi-commodity flow model is run either each time a decision must be made or at pre-defined intervals based on existing requests. Depending on the objective and constraints, this may become computationally intractable. Without edge capacities and with an objective of minimizing travel time, this method when computed at pre-defined intervals is equivalent to dynamic shortest path.

3.2.8 Comparison

In Table 1, we compare the possible methods in terms of the requirements. An X indicates that the method meets the requirement and a P indicates that the method may meet the requirement. Static shortest path and k-shortest path consider congestion if they use updating edge weights. Note that no single method meets all four requirements. Dynamic shortest path and dynamic multi-commodity flow both meet three of the four but dynamic multi-commodity flow is either equivalent to dynamic shortest path or is too computationally intense for use in our application. Thus, our proposed method is a dynamic shortest path method using updating edge weights. Although it does not directly consider future vehicle interaction, updating edge weights provide an approximation based on recent history.

3.3 Proposed Dynamic Routing Method

Our proposed method updates edge weights via exponential smoothing and an all-to-all shortest path problem is run at a defined time interval based on current edge weights. Vehicles may be rerouted each time they approach a diverging node (where

Table 1: Comparison of Possible Routing Methods

	Computationally Efficient	Allows Route Change	Considers Congestion	Considers Future Vehicle Interaction
Static Shortest Path	X		P	
Dynamic Shortest Path	X	X	X	
Static K-Shortest Path	X		P	
Randomized K-Shortest Path	X			
Static Time-Expanded Shortest Path			X	X
Pre-Computed Multi-Commodity Flow	X			X
Dynamic Multi-Commodity Flow		X	X	X

they have a route choice) based on the most resent route calculation. In this way, routes always reflect the most recently available edge traversal time estimates and the system, at any point in time, reflects the system-optimal solution given this information. In this section, we present the details of our proposed approach.

3.3.1 Graph

We model our system as a directed graph where *nodes* correspond to loading/unloading ports and rail intersection points and *edges* to rail segments connecting nodes. Each edge is assigned a *weight* that represents its estimated traversal time. For some purposes, we consider the *intersection point network* where nodes represent only rail intersection points, not ports. We use the term *diverging node* to indicate a node that has more than one outgoing edge. Note that in typical OHT networks, the total number of incoming plus outgoing edges associated with a node is no more than three.

3.3.2 Congestion-Aware Edge Weights

For each edge, we store an edge weight that represents the estimated traversal time. When a vehicle passes a node, the system records the time that it took for that vehicle to traverse that edge and updates the estimated edge traversal time using exponential smoothing,

$$t_{ij}^{n} = (1-\lambda)t_{ij}^{n-1} + \lambda l_{ij}^{n},$$

where t_{ij}^{n} is the edge weight for edge (i, j) after the n^{th} traversal, l_{ij}^{n} is the n^{th} traversal time, and λ is a parameter indicating the sensitivity of the estimated edge traversal time to new information.

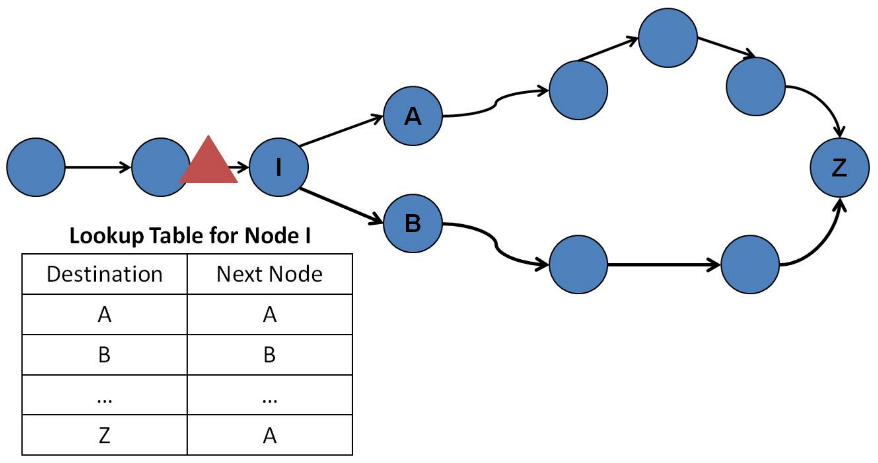
Initially, we set the edge weight equal to the length of the edge divided by the maximum travel velocity on that edge. With a small value of λ, estimated edge traversal times approach the long-run average. Good values of λ may vary with layout and operational characteristics affecting the likelihood of heavy congestion or deadlock. We use one value of λ across the whole system but could use edge-specific values of λ. For example, it may be preferable to use larger values of λ for areas of the network with more frequent congestion and smaller for areas that are sparsely used.

Previously studied measures of congestion, such as the number of vehicles currently moving on a route, represent an instantaneous view of system congestion. By incorporating recent information in addition to current information, a more robust representation can be achieved. Our method also incorporates expected delay due to layout characteristics such as likely deceleration behind vehicles entering curved edges.

3.3.3 Route Computation and Selection

In our proposed method, vehicles are routed and rerouted using node-based lookup tables. Traditionally, a vehicle's route is stored as a vehicle attribute. At each diverging node the vehicle selects its next node from the stored route. To route dynamically, a vehicle recalculates its route at each diverging node to ensure that its preferred route has not changed. If we have R requests during a T second simulation time and each route has an average of D diverging nodes, the total number of route calculations will be $R * (D + 1)$, which may not be evenly distributed over time.

Our proposed method stores a lookup table at each diverging node containing the next node in the route to each destination. Tables are updated periodically over time at a fixed time interval. As a vehicle approaches the diverging node, it looks up its next node in the table. For example, in Figure 3.3.3 a vehicle (red triangle) is approaching node I and has a destination of node Z. The vehicle looks in the table associated with node I in the row for destination Z. The table indicates that the vehicle should next go to node A based on the current congestion status. Note that each vehicle with a particular destination approaching a given node between two subsequent table updates will follow the same route.



Lookup Table for Node I

Destination	Next Node
A	A
B	B
…	…
Z	A

Figure 13: Lookup Table-Based Routing

To update a table, we run a one-to-all shortest path calculation on the intersection point network using Dijkstra's algorithm. Table updates may be easily parallelized since the one-to-all calculation for each diverging node is independent. With an I second update interval over a T second simulation time, we update tables T/I times. Thus, the number of calculations does not increase with a higher number of requests or more diverging nodes, allowing efficient use in large networks. Because tables are updated at fixed intervals, computational requirements are uniform over time. In our implementation, the complexity of each table update is $O(|N|^3)$ where $|N|$ is the number of intersection point nodes.

3.4 Computational Results

To test proposed our approach, we use the simulation described in Chapter 2. We compare our dynamic shortest path routing method with updating edge weights against the static shortest path routing method using Distance-Based, Long-Run Average, and Updating edge weights. Our method provides a moderate improvement in steady-state performance, a significant reduction in the frequency of heavy congestion, and an improved response to vehicle breakdown. Even though transfer requests are generated at the rate of 1-2 per second, the dynamic routing algorithm is fast enough so that no delays occur in its simulated implementation.

3.4.1 Routing Parameters

We compare static and dynamic routing using three parameter settings for each λ and the table update interval. In the tables and graphs, *Routing Method* indicates either static (S) or dynamic (D) routing, *Table Updates Interval* indicates the time in seconds between table updates for dynamic routing, *Edge Type* indicates either distance-based (DB), long-run average (LRA), or updating (U) edge weights, and *Edge Weight Parameter* indicates the value of λ for updating edge weights.

3.4.2 Metrics

For steady-state replications, we report total time in the system, deliver time, vehicle utilization, speed index, and number of requests completed. Total time in the system is defined as the time from when a request is generated until delivery is complete. The deliver time is the time from when pickup is complete until the time delivery is complete. Vehicle utilization at a particular point in time is the percent of vehicles assigned to a request. Speed index is the average ratio of current velocity to maximum velocity. At a particular point in time, the average is taken across all vehicles that should be moving (i.e. not stopped to load/unload or intentionally idle). Maximum velocity will depend on whether the vehicle is on a curved or straight edge.

Time in system and deliver time are reported as the average over all completed requests generated after the warm-up period. Vehicle utilization and speed index are a system-wide value recorded every 5 seconds of simulation time. A lower total time in system, deliver time, and vehicle utilization and higher speed index and number of requests completed are desirable. Due to acceleration and deceleration, a speed index of 1 is not attainable but with no congestion the speed index can reach 0.9.

3.4.3 Base-Case Results

The base case scenario has 175 vehicles and an average steady-state vehicle utilization of 88%. Because heavy congestion occurs in six of twenty replications using distance-based static routing, our steady-state metrics comprise the average across the remaining fourteen replications. We consider the same replications for other parameter settings. These replications are also steady-state because they do not reach heavy congestion. A heavy congestion replication is defined as one that has a speed index below 0.2 at the time of completion.

Six of the initial twenty replications reach heavy congestion under static routing. The remaining fourteen replications result in standard deviations that are sufficiently low so we use these fourteen replications for steady-state analysis. Table 2 presents these steady-state results including 95% confidence intervals. The routing method is identified in the form *routing method - table update interval - edge weight type - edge weight parameter* where routing method is static (S) or dynamic (D), table update interval is in seconds, edge weight type is distance-based (DB), long-run average (LRA), or updating (U), and edge weight parameter is a value between 0 and 1. We present edge weight parameters of 0.1 and 0.9. We tested additional values and found only a slight difference in steady-state performance. The value of 0.1 tended to perform best with performance decreasing slightly as the edge weight parameter increased toward 0.9. This is likely because as the value increases, it diverts traffic to longer routes more easily in response to congestion. At some workload levels, this may be required to avoid heavy congestion.

Tables 3 and 4 show the t-statistic for paired t-tests for deliver time and number of requests delivered, respectively. With 13 degrees of freedom in a two-sided t-test at a 95% confidence level, the difference between two parameter settings is significant if the absolute value of the t-statistic is greater than 2.16. In Tables 3 and 4, we show, in bold, values where the column parameter setting shows a significant improvement

Table 2: Steady-State Mean and 95% Confidence Interval

Routing Method	Total Time in System	Deliver Time	Vehicle Utilization	Speed Index	Requests Completed
S-NA-DB-NA	113.0	77.6	0.88	0.66	16357
	(112.99,119.45)	(77.6,77.97)	(0.88,0.92)	(0.66,0.67)	(16356.86,16679.49)
S-NA-LRA-NA	106.7	74.3	0.86	0.70	16945
	(106.67,110.83)	(74.35,74.9)	(0.86,0.9)	(0.7,0.71)	(16945,17244.93)
S-NA-U-0.1	107.5	74.8	0.87	0.70	16975
	(107.48,111.49)	(74.83,75.34)	(0.87,0.91)	(0.7,0.7)	(16975.29,17281.18)
S-NA-U-0.9	109.3	75.8	0.87	0.69	16784
	(109.26,113.57)	(75.83,76.41)	(0.87,0.91)	(0.69,0.7)	(16783.71,17103.95)
D-1-U-0.1	106.9	74.7	0.86	0.70	16919
	(106.88,110.81)	(74.68,75.29)	(0.86,0.9)	(0.7,0.71)	(16919.29,17258.47)
D-10-U-0.1	107.1	74.9	0.86	0.70	16923
	(107.13,110.69)	(74.88,75.39)	(0.86,0.9)	(0.7,0.71)	(16922.93,17285.86)
D-60-U-0.1	107.5	74.8	0.86	0.70	16918
	(107.49,112.29)	(74.81,75.41)	(0.86,0.91)	(0.7,0.71)	(16918.21,17224.94)
D-1-U-0.9	108.1	75.2	0.86	0.70	16830
	(108.06,111.87)	(75.18,75.77)	(0.86,0.91)	(0.7,0.71)	(16829.64,17191.1)
D-10-U-0.9	108.5	75.5	0.86	0.70	16802
	(108.47,112.41)	(75.53,76.01)	(0.86,0.91)	(0.7,0.7)	(16801.57,17093.02)
D-60-U-0.9	108.9	75.8	0.87	0.69	16797
	(108.94,112.24)	(75.79,76.26)	(0.87,0.91)	(0.69,0.7)	(16796.93,17125.11)

over the row parameter setting. Note that deliver time improvement has a positive t-statistic and number of requests delivered improvement has a negative t-statistic because a lower deliver time and a higher number of requests completed are preferred. The tables are symmetric so all values in the first column are significant in the negative direction. Only deliver time and number of requests delivered are shown but other metrics show similar patterns.

Table 3: Deliver Time Paired T-Test Test Statistic

Routing Method	S-NA-DB-NA	S-NA-LRA-NA	S-NA-U-0.1	S-NA-U-0.9	D-1-U-0.1	D-10-U-0.1	D-60-U-0.1	D-1-U-0.9	D-10-U-0.9	D-60-U-0.9
S-NA-DB-NA		**22.73**	**20.44**	**10.47**	**16.03**	**19.61**	**14.71**	**13.33**	**10.84**	**11.33**
S-NA-LRA-NA	-22.73		-3.83	-18.59	-2.07	-4.74	-3.48	-8.00	-11.08	-10.64
S-NA-U-0.1	-20.44	**3.83**		-8.53	1.17	-0.68	0.12	-2.85	-5.74	-8.16
S-NA-U-0.9	-10.47	**18.59**	**8.53**		**8.87**	**7.59**	**10.41**	**6.89**	**2.95**	0.29
D-1-U-0.1	-16.03	2.07	-1.17	-8.87		-1.67	-1.39	-3.67	-7.02	-6.39
D-10-U-0.1	-19.61	**4.74**	0.68	-7.59	1.67		0.47	-3.04	-6.86	-8.07
D-60-U-0.1	-14.71	**3.48**	-0.12	-10.41	1.39	-0.47		-3.08	-5.62	-6.08
D-1-U-0.9	-13.33	**8.00**	**2.85**	-6.89	**3.67**	**3.04**	**3.08**		-3.99	-5.95
D-10-U-0.9	-10.84	**11.08**	**5.74**	-2.95	**7.02**	**6.86**	**5.62**	**3.99**		-1.98
D-60-U-0.9	-11.33	**10.64**	**8.16**	-0.29	**6.39**	**8.07**	**6.08**	**5.95**	1.98	

All alternative parameter settings show a statistically significant improvement over distance-based static routing in most metrics. The long-run average edge type and

Table 4: Number of Requests Delivered Paired T-Test Test Statistic

Routing Method	S-NA-DB-NA	S-NA-LRA-NA	S-NA-U-0.1	S-NA-U-0.9	D-1-U-0.1	D-10-U-0.1	D-60-U-0.1	D-1-U-0.9	D-10-U-0.9	D-60-U-0.9
S-NA-DB-NA		**-7.07**	**-9.06**	**-5.49**	**-7.35**	**-6.82**	**-8.75**	**-6.51**	**-5.78**	**-6.04**
S-NA-LRA-NA	7.07		-0.94	2.92	0.65	0.57	0.55	2.40	3.55	3.27
S-NA-U-0.1	9.06	0.94		3.64	1.78	1.69	2.03	3.56	5.59	5.47
S-NA-U-0.9	5.49	**-2.92**	**-3.64**		**-3.18**	**-2.19**	**-2.82**	-1.21	-0.38	-0.23
D-1-U-0.1	7.35	-0.65	-1.78	3.18		-0.11	0.04	3.99	4.81	3.42
D-10-U-0.1	6.82	-0.57	-1.69	2.19	0.11		0.11	2.33	2.77	2.72
D-60-U-0.1	8.75	-0.55	-2.03	2.82	-0.04	-0.11		2.37	4.10	2.92
D-1-U-0.9	6.51	**-2.40**	**-3.56**	1.21	**-3.99**	**-2.33**	**-2.37**		0.76	0.80
D-10-U-0.9	5.78	**-3.55**	**-5.59**	0.38	**-4.81**	**-2.77**	**-4.10**	-0.76		0.12
D-60-U-0.9	6.04	**-3.27**	**-5.47**	0.23	**-3.42**	**-2.72**	**-2.92**	-0.80	-0.12	

dynamic routing with a table update interval of one second and an edge weight parameter of 0.1 show the best performance. All settings with an edge weight parameter of 0.1 outperform those with an edge weight parameter of 0.9. There is less sensitivity to the table update interval parameter. We conclude that static routing with long run average edge weights and dynamic routing with a low sensitivity to new information perform well under low workload scenarios that do not reach heavy congestion. The benefit of dynamic routing over static routing is seen in the high-workload and vehicle breakdown scenarios.

3.4.4 High-Workload Results

In the high-workload scenario, with 250 vehicles and more frequent requests, distance-based routing results in heavy congestion in all twenty replications. Figure 3.4.4 shows the smoothed progression of the speed index over time for one replication of this scenario. Only one instance of dynamic routing is shown but the performance is similar across parameter settings. The speed index for distance-based edge weights experiences a quick transition from steady-state to near zero. Each of the other static routing cases ends in heavy congestion but some effort is made periodically to recover. The dynamic routing case does not enter heavy congestion.

Table 5 shows the frequency of heavy congestion in the base case and high-workload scenarios. The frequency of heavy congestion decreases from 100% to 80%

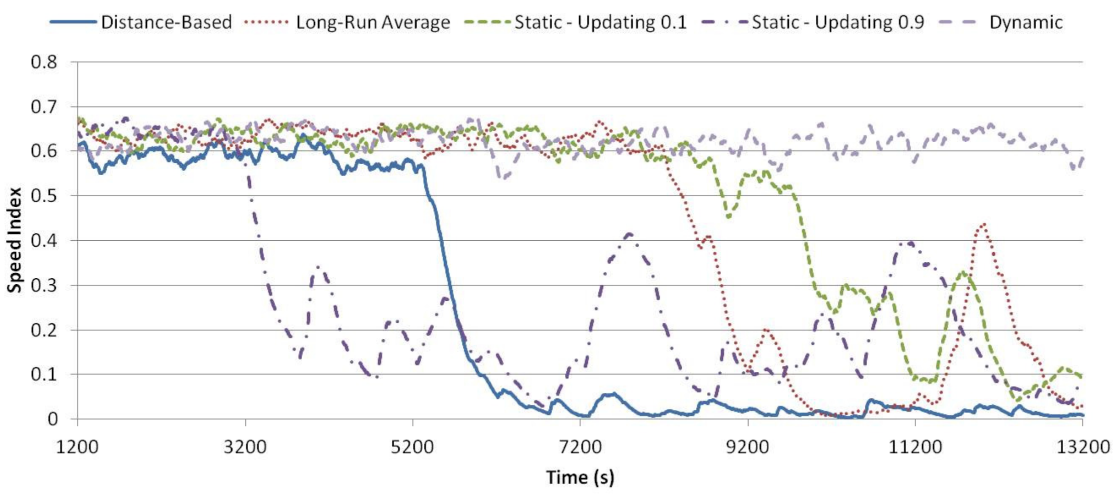


Figure 14: Higher-Workload Speed Index Example

with long-run average edge weights with static routing. Updating edge weights with static routing further reduces the frequency to 35%-45%. With dynamic routing, the frequency is further reduced. A one second update interval with an edge weight parameter of 0.1 does not result in heavy congestion in any of the twenty replications.

Table 5: Frequency of Heavy Congestion

Routing Method	Base Case	Higher-Workload Scenario
S-NA-DB-NA	30%	100%
S-NA-LRA-NA	0%	80%
S-NA-U-0.1	0%	35%
S-NA-U-0.9	0%	45%
D-1-U-0.1	0%	0%
D-10-U-0.1	0%	30%
D-60-U-0.1	0%	15%
D-1-U-0.9	0%	15%
D-10-U-0.9	0%	10%
D-60-U-0.9	0%	5%

We use the binomial test to determine statistical significance. We find that long-run average edge weights do not provide a statistically significant improvement over distance-based static routing at the 95% confidence level but all settings with adjusting edge weights do provide improvement over both distance-based and long-run average edge weights. For settings with an edge weight parameter of 0.9, dynamic routing shows a statistically significant improvement over static routing. We cannot draw any conclusions with respect to the edge weight parameter because the difference is not statistically significant. The determination of an appropriate edge weight parameter is likely to workload-specific.

3.4.5 Vehicle Breakdown Results

In the vehicle breakdown scenario, the number of vehicles and the request frequency are the same as in the base case but vehicle breakdowns occur on the center loop with an exponential inter-breakdown time with mean 2700 seconds. When a breakdown occurs, each center loop edge is selected with equal probability and the next vehicle to enter that edge must come to a stop. A breakdown is short (90-180 seconds) with a probability of 0.75 and long (600 seconds) with a probability of 0.25. We focus on the center loop in order to observe deadlock and recovery behavior.

For analysis, we focus on the response to and recovery from vehicle breakdown. Figure 3.4.5 shows the smoothed progression of the speed index over time for a replication with a long breakdown at approximately 6300 seconds. All parameter settings react to the breakdown with a significant decrease in speed index. Dynamic routing allows vehicles to be immediately routed away from the congested area while static routing only allows new requests to be routed away from this area. Dynamic routing quickly returns to a steady-state speed index. Static routing settings with updating edge weights recover after more than 1500 seconds and static routing with distance-based and long run average edge weight settings never recover. In this example, a short breakdown also occurs at approximately 12,800 seconds. Dynamic routing does not experience a significant drop in the speed index whereas static routing does.

Across all replications, we observe results consistent with those presented in the above example. Static routing with distance-based and long run average edge weights are unlikely to recover, static routing with updating edge weights recover but after a substantial length of time, and dynamic routing recovers more quickly. As before we

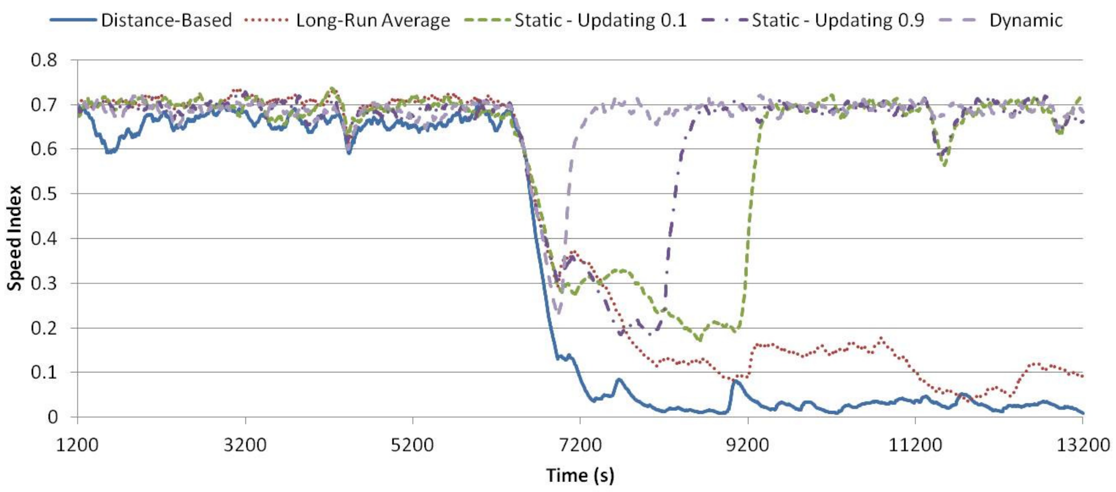


Figure 15: Vehicle Breakdown Speed Index Example

define heavy congestion as a speed index falling below 0.2. We define recovery as a speed index reaching above 0.5 after falling below 0.2.

Figures 3.4.5 and 3.4.5 summarize these results. Static routing with distance-based edge weights reaches heavy congestion in seventeen of twenty replications and only recovers in one replication. Static routing with long run average edge weights reaches heavy congestion in thirteen of twenty replications and recovers in two of them. The remaining settings reach heavy congestion in ten to fourteen replications but recover in either all cases or all except one. The difference between static and dynamic routing in scenarios with updating edge weights is seen in the time to recovery as shown in Figure 3.4.5. Both cases of static routing have average recovery times of over 1500 seconds. Dynamic routing average recovery times are below 350 seconds. The time to recovery in dynamic routing is not sensitive to differences in parameter settings.

3.5 Conclusion

In this chapter, we proposed a congestion-aware dynamic routing method that uses edge weights continually updates based on edge traversal times. For computational efficiency, our method routes and re-routes vehicles using node-based lookup tables

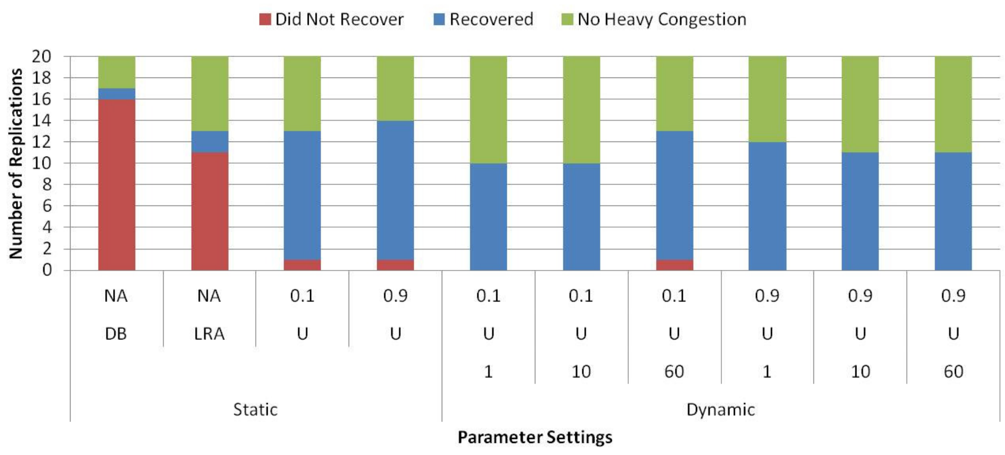


Figure 16: Frequency of Heavy Congestion and Recovery

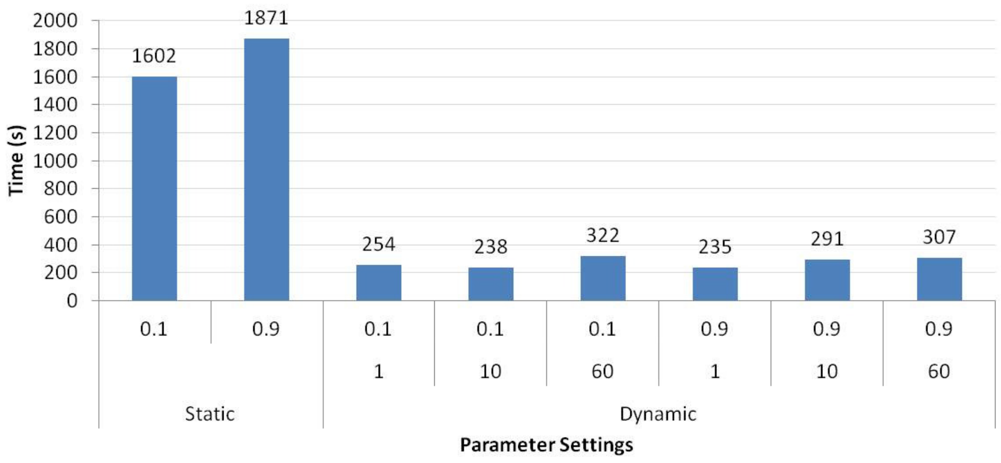


Figure 17: Average Time to Recovery

where, at each diverging node, a vehicle looks up its next node in a table based on its destination. Routing tables are periodically updated using current edge weights. By considering congestion and allowing vehicles to be rerouted, we demonstrate via simulation a moderate improvement in steady-state performance, a significant reduction in the frequency of heavy congestion, and an improved ability to react to and recover from exceptions such as vehicle breakdowns.

CHAPTER IV

LAYOUT DESIGN

Having demonstrated in Chapter 3 that dynamic routing provides significant improvement over static routing for a fixed layout, in this chapter we address the question of how layout can be improved to make better use of dynamic routing. In particular, we focus on how the number and placement of shortcuts on the center and outer loops affects system performance differently with static versus dynamic routing. For our purposes, a shortcut is a short edge connecting two longer edges. We assume that the long parallel travel lanes that make up the center and outer loops are fixed and consider four configurations for center and outer loop shortcuts.

This focus on shortcuts in the center and outer loops is motivated by what has been experienced in practice, suboptimal placement of a shortcut with the use of static routing sometimes causes prohibitive congestion. Two possible reasons for this are that 1) the new shortcut is included in the shortest path for a large number of origin/destination pairs and that 2) the shortcut shifts shortest paths such that increased traffic travels through an already busy area of the network. We observe the second case in our simulation when using static routing. With dynamic routing, if these areas become congested vehicles will avoid them because with congested travel time estimates they no longer lie along the shortest path. With dynamic routing, the system is less sensitive to shortcuts placement and continues to function efficiently with a significant increase in the number of shortcuts. With the increased number of shortcuts, point-to-point distances are reduced enough to reduce travel time by 25%.

Traditional network design problems assume static routing and, if congestion is considered, the congestion patterns created by static routing. Because dynamic routing adapts to changes in congestion by changing traffic patterns, the ideal layout for use with dynamic routing may differ significantly from that for static routing. Finding an optimal layout for dynamic routing is a challenging problem.

Even for static routing, with a realistic set of constraints, network design is challenging. Network design problems are typically modeled as integer programs [68] and, with a realistic set of constraints, are difficult to solve. Although the basic network design problems, without the consideration of congestion or any other additional constraints, is a series of shortest path problems [68] and thus polynomially solvable, with the addition of a fixed charge in the objective function associated with the inclusion of an edge, the problem becomes NP-Hard [35] even with fractional flow. Also, the consideration of congestion either with edge capacities or in edge weight functions causes the problem to be NP-Hard [34].

Incorporating dynamic routing into a network design problem would also require an effective method of modeling it analytically. The complexities of our problem, particularly the nature of vehicle interaction and congestion and the uncertainty associated with demand, make this difficult. Most network design models that consider congestion use a flow-based representation where congestion is either limited by arc capacities or modeled as a function of flow [43]. For example, in the user-optimal or selfish equilibrium models used in traffic and internet routing, it is common for travel time to be represented as a non-decreasing and convex function of flow [3]. Linear and polynomial functions are common [54]. In our system, the interaction of individual vehicles is significant as vehicles are non-infinitesimal in size. Also, because loading/unloading occurs on track, a stopped vehicle prevents other vehicles from continuing on their paths creating a highly variable congestion pattern that cannot necessarily be modeled by a simple function.

To aid in developing such an analytical model, we develop a simulation support tool that allows efficient generation of thousands of potential layouts. We intend to use this tool in an iterative framework where an optimization or other model produces a layout that can be translated into a simulation layout and tested via simulation. The results from the simulation will then be used to provide feedback to the optimization model and insight to the analyst. In this chapter, we present an Excel-based automated layout generation tool that allows efficient generation of thousands of potential layouts for a bay-based unified AMHS. In addition, we update the request generation method in our simulation to provide additional flexibility and the bay structure to be more realistic.

In this chapter, we also present results that highlight the potential impact of making layout changes for use with dynamic routing. We use our layout generation and simulation tools to demonstrate that the impact of layout changes may impact performance significantly differently depending on whether static or dynamic routing is used. We observe that with static routing, suboptimal shortcut placement creates delay and results in heavy congestion when shortest paths travel through high demand bays. Dynamic routing is able to avoid high demand bays as they become congested, making the system more robust to variation over time. With dynamic routing, we are able to increase the number of shortcuts eight-fold which reduces point-to-point distance and allows a 25% decrease in travel time.

4.1 Automated Layout Generation Tool

Building a simulation layout and adjusting parameters requires at least several hours for a knowledgeable user. To facilitate the generation of simulation layouts, we developed an Excel-based automated layout generation tool that generates all required simulation file with the click of a button. The user selects from a modular menu of facility and operational characteristics in an Excel template. Visual Basic for Applications (VBA) generates text files formatted for AutoMod and associated input text files. It allows efficient generation of thousands of different layouts defined by the modular selection of characteristics used in the design of a typical unified AMHS system. The tool was developed in Excel in order to be portable across users without requiring unique software or programming knowledge. The user selects characteristics of the bays, center loop, outer loop, demand pattern, and operational control. In addition to generating simulation files, the tool records network information that can be used in post-analysis such as node coordinates and edge lengths. In combination with simulation output, this allows efficient and flexible graphical data analysis.

The layouts produced by this tool are unified AMHS layouts with one line of bays along the top and one line along the bottom. Bays can be selectively removed as shown in Figure 4.1 to represent machine that require more floor space. The bay structure, either 3-lane or 4-lane, shown in Figure 4.1 differs from that used in the initial simulation, reflecting a more realistic structure.

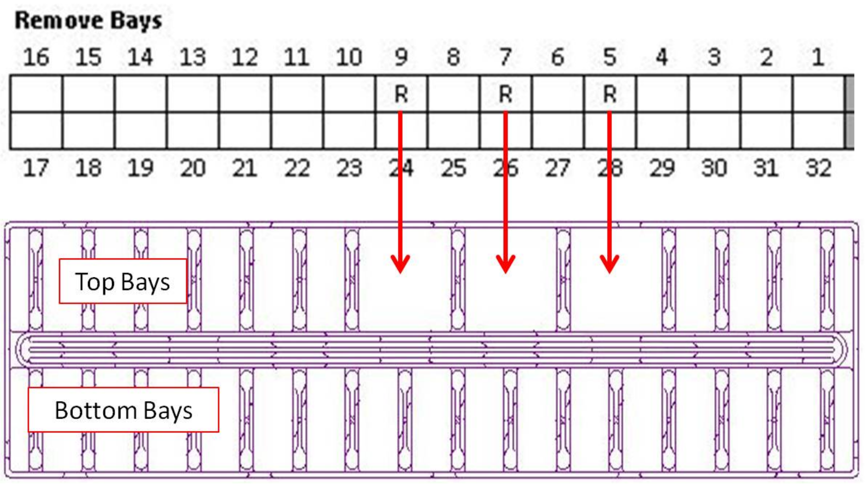


Figure 18: Bay Placement and Optional Bay Removal

The center loop area may have four or eight lanes with several travel directions options as shown in Figures 4.1 and 4.1. Several types of connections between the center and outer loops are possible, examples of which are shown in Figure 4.1. Together, there are fourteen options for the center loop configuration, not considering

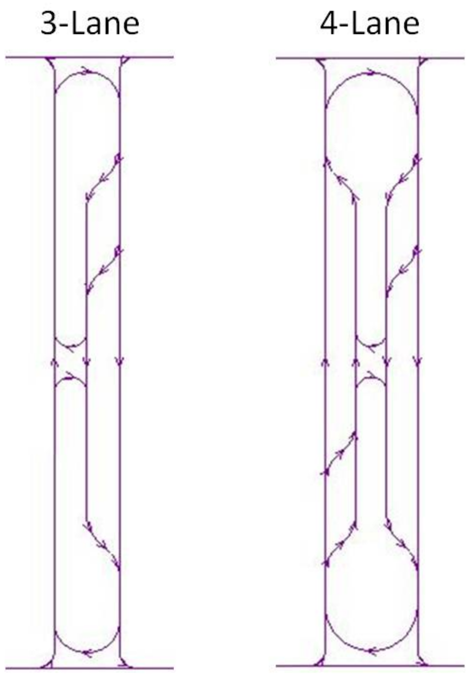


Figure 19: Bay Structure

the placement of shortcuts.

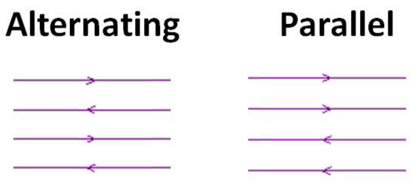


Figure 20: Center Loop Travel Direction: Four Lanes

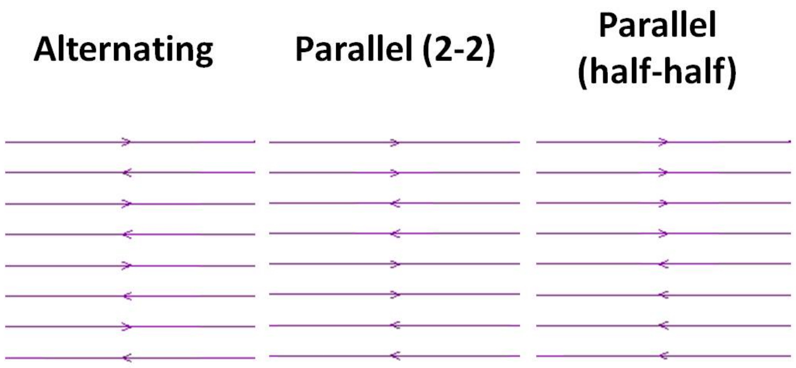


Figure 21: Center Loop Travel Direction: Eight Lanes

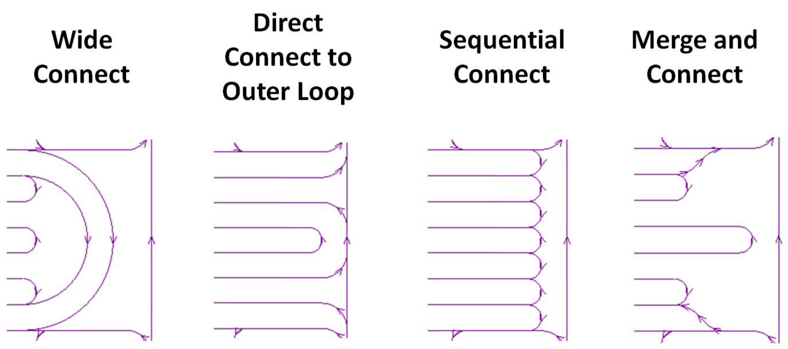
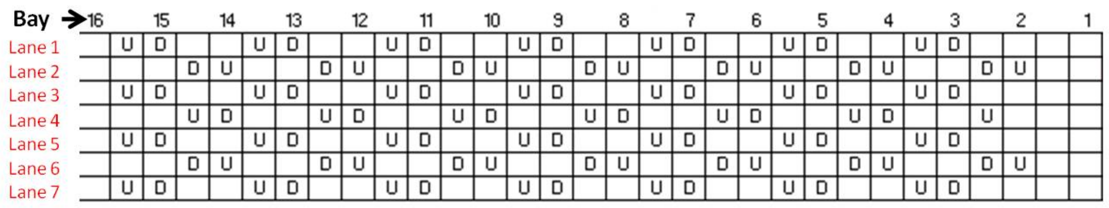


Figure 22: Examples of Center to Outer Loop Connection Types

Center and outer loop shortcuts may be aligned either with the center of a bay or directly between two bays. Shortcuts are placed via a table, as shown in Figure 4.1, where the user indicates whether the shortcuts should direction should be up or down. We allow the placement of up to $n - 1$ shortcuts between each set of travel lanes on the center loop, where n is the number of bays, with some limitations on the placement of shortcuts at the right and left sides. The shape of a shortcut differs based on the direction of travel of the connecting lanes as shown in Figure 4.1.



Bay →	16		15		14		13		12		11		10		9		8		7		6		5		4		3		2		1
Lane 1		U	D			U	D			U	D			U	D			U	D			U	D			U	D				
Lane 2				D	U			D	U			D	U			D	U			D	U			D	U			D	U		
Lane 3		U	D			U	D			U	D			U	D			U	D			U	D			U	D				
Lane 4				U	D			U	D			U	D			U	D			U	D			U	D			U			
Lane 5		U	D			U	D			U	D			U	D			U	D			U	D			U	D				
Lane 6				D	U			D	U			D	U			D	U			D	U			D	U			D	U		
Lane 7		U	D			U	D			U	D			U	D			U	D			U	D			U	D				

Figure 23: Placement of Center Loop Shortcuts

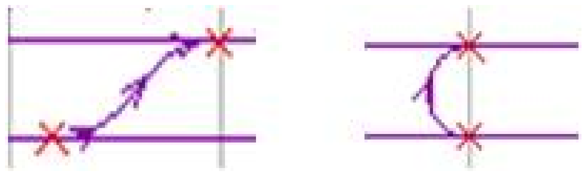
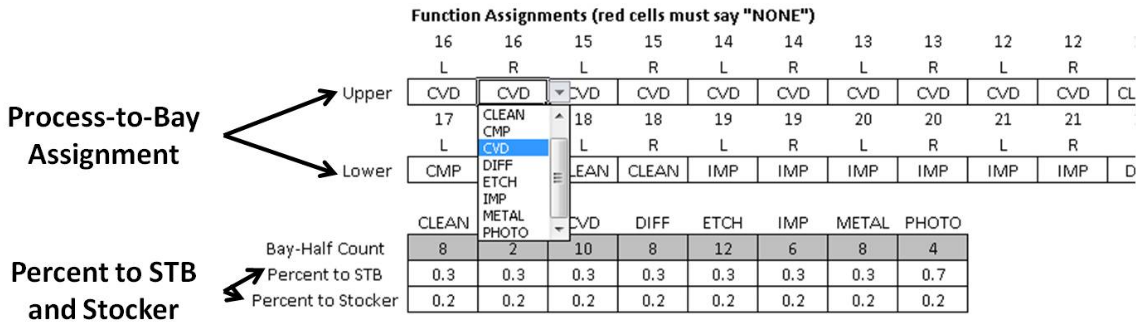


Figure 24: Shortcut Shape

All parameters associated with vehicle control, request generation, and scenario definition are also entered in the Excel file so that only limited manipulation is required within the simulation itself. Vehicle characteristics include the number of vehicles, the vehicle velocity on straight edges, the vehicle velocity on curved edges, the rate of acceleration, and the rate of deceleration. In addition being used directly in the simulation, these values are used to calculate initial edge weight values and lower bound travel times. Request generation parameters include bay-to-process assignment and storage parameters (see Figure 4.1) and transition matrices.



Function Assignments (red cells must say "NONE")

	16	16	15	15	14	14	13	13	12	12	
	L	R	L	R	L	R	L	R	L	R	
Upper	CVD	CVD	VD	CVD	CVD	CVD	CVD	CVD	CVD	CVD	CL
	17		18	18	19	19	20	20	21	21	
	L		L	R	L	R	L	R	L	R	
Lower	CMP		LEAN	CLEAN	IMP	IMP	IMP	IMP	IMP	IMP	D

	CLEAN		VD	DIFF	ETCH	IMP	METAL	PHOTO
Bay-Half Count	8	2	10	8	12	6	8	4
Percent to STB	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.7
Percent to Stocker	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2

Figure 25: Process-to-Bay Assignment and Storage Parameters

4.2 Simulation Updates

In addition to developing the layout generation tool, in this phase of our work we made several updates to the production-based request generation method to allow more flexibility in analysis and to better represent the real system. We allow a more flexible use of storage and modify the initialization procedure. In addition, we introduce a batch process, which is explained below.

In our initial simulation, we use one parameter to specify each the probability of a cartridge visiting a stocker between processing steps and the probability of a cartridge visiting a side-track-buffer between processing steps. In this phase, we allow different parameters for each processing type. For example, the probability of visiting a stocker when going to a CLEAN machine may differ from the probability of visiting a stocker when going to a PHOTO machine. This mimics the real system where storage may be used more often to keep bottleneck machines highly utilized. In addition, stocker locations are consolidated along the center loop and assigned one to each bay as shown in Figure 4.2.

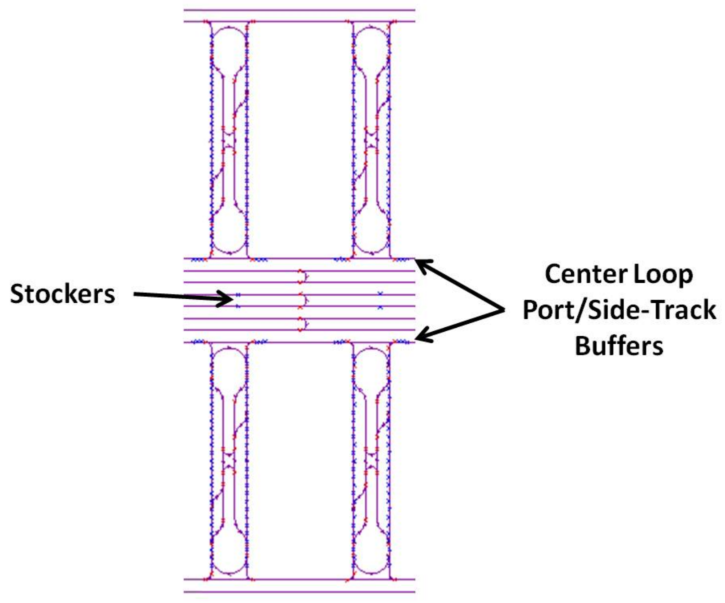
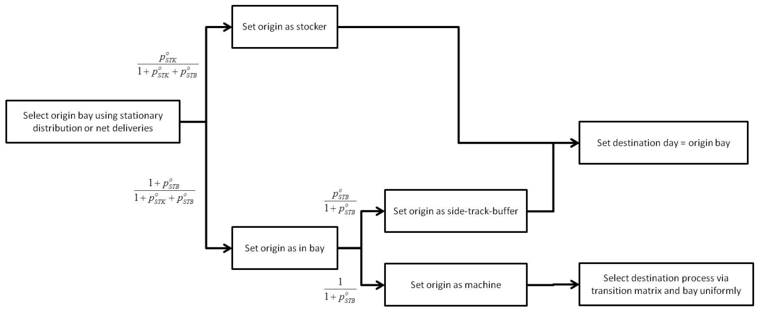


Figure 26: Center Loop Port/Side-Track Buffer and Stocker Placement

Instead of using a transition matrix that incorporate the process-only transition matrix and the storage parameters, we use the process-only transition matrix and storage parameters directly in the simulation. The overall process is shown in Figure 4.2. As in the initial simulation, the request origin may be selected from a stationary distribution or from the distribution of net deliveries. We use a stationary distribution generated in the same was as in the initial simulation but using the new storage parameters. Once the bay has been selected, it is assigned as a stocker with probability $\frac{p_{STK}}{1+p_{STK}+p_{STB}}$ where p_{STK} and p_{STB} are the probabilities that a cartridge visits a stocker and side-track-buffer, respectively, between process steps. p_{STK} and p_{STB} are those associated with the origin process. This is important in determining the destination process. When tracking net deliveries, stockers are considered separately from the bays themselves. Once the origin bay has been selected, if it is a bay as opposed to a stocker, it is assigned as a side-track-buffer with probability $\frac{p_{STB}}{1+p_{STB}}$. As before, for machines and side-track-buffers the specific location is selected uniformly from all stopping locations in the bay.



Set origin as stocker
$\frac{p^o_{STK}}{1+p^o_{STK}+p^o_{STB}}$
Select origin bay using stationary distribution or net deliveries
Set destination day = origin bay
$\frac{1+p^o_{STB}}{1+p^o_{STK}+p^o_{STB}}$
$\frac{p^o_{STB}}{1+p^o_{STB}}$
Set origin as side-track-buffer
Set origin as in bay
$\frac{1}{1+p^o_{STB}}$
Set origin as machine
Select destination process via transition matrix and bay uniformly

Figure 27: Origin/Destination Selection Process

Once the origin has been selected, the destination function is selected. If the origin is a stocker or side-track-buffer, the destination process is the same as the origin process. If the origin is a side-track-buffer, the bay is the same, as well. If is a stocker, the destination bay is the bay associated with the stocker. If the origin is a machine, the destination process is selected via the transition matrix. In this case, the destination is assigned as a stocker with probability p_{STK} where p_{STK} is the parameter associated with the use of a stocker for the destination process.

Instead of using a request generation warm-up period as in the initial simulation, we initialize the system via a stationary distribution. An input parameter specifies the initial number of net deliveries and they are distributed across processes via the stationary distribution. Distribution among bays uses a uniform distribution.

In the real system, some processes are batched meaning that several items are processed at one time. We model this using the DIFF process with batch size B as an input parameter. Every B^{th} time a request with an origin of type DIFF is selected, B requests are created with potentially differing destinations. Those requests with DIFF origin that are selected in between are discarded.

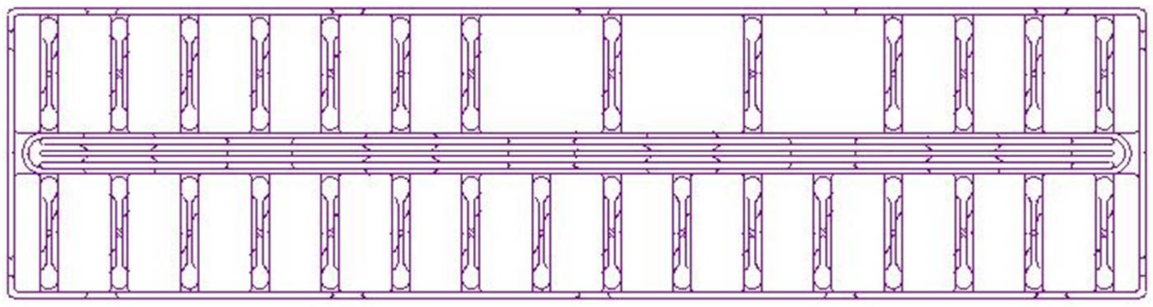
4.3 Computational Results

To demonstrate the potential of our simulation analysis tools and gain insight into the improvements possible with layout changes for dynamic routing, we compare four configurations for the center and outer loops, differing only by the number and placement of shortcuts. We observe that static routing results in heavy congestion caused by suboptimal shortcut placement. Layouts that increase flow too much in high-demand areas of the network become overloaded. Dynamic routing eliminates this heavy congestion in all layouts considered. Further, adding shortcuts always has a positive impact on travel times. By maximizing the number of shortcuts on the center and outer loops, delivery times and total system times decrease by 24%-26% with constant request frequency parameters. By also increasing request frequency, the transportation throughput, or number of requests delivered, increases by more than 26% without the occurrence of heavy congestion. Thus, we conclude that substantial improvement is possible with the implementation of layout changes for use with dynamic routing.

4.3.1 Configurations

4.3.1.1 Rail Layout

We consider four center and outer loop configurations for the prototype layout shown in Figure 4.3.1.1 with 32 four-lane bay locations, 16 along the top and 16 along the bottom. Three bays are removed from the top row to represent the additional space required for photolithography equipment. The center loop has eight lanes. Lanes 1 and 2 go from left to right, 3 and 4 go from right to left, 5 and 6 go from left to right, and 7 and 8 go from right to left. Shortcut placement for the four center loop configurations is shown in Figure 4.3.1.1. We refer to these configurations as *1*, *2*, *3*, and *4*. From layout *1* which has the fewest shortcuts, we increase the number of shortcuts by up to a factor of eight as in layout *4*. The two lanes in the outer loop have opposite travel directions and outer loop shortcut density follows the same pattern as the center loop.



Process

CVD	CVD	CVD	CVD	CVD	CLN	CLN		PHO		PHO		ETC	ETC	ETC	ETC
CMP	CLN	IMP	IMP	IMP	DIFF	DIFF	DIFF	DIFF	MET	MET	MET	MET	CLN	ETC	ETC

Figure 28: Baseline Layout

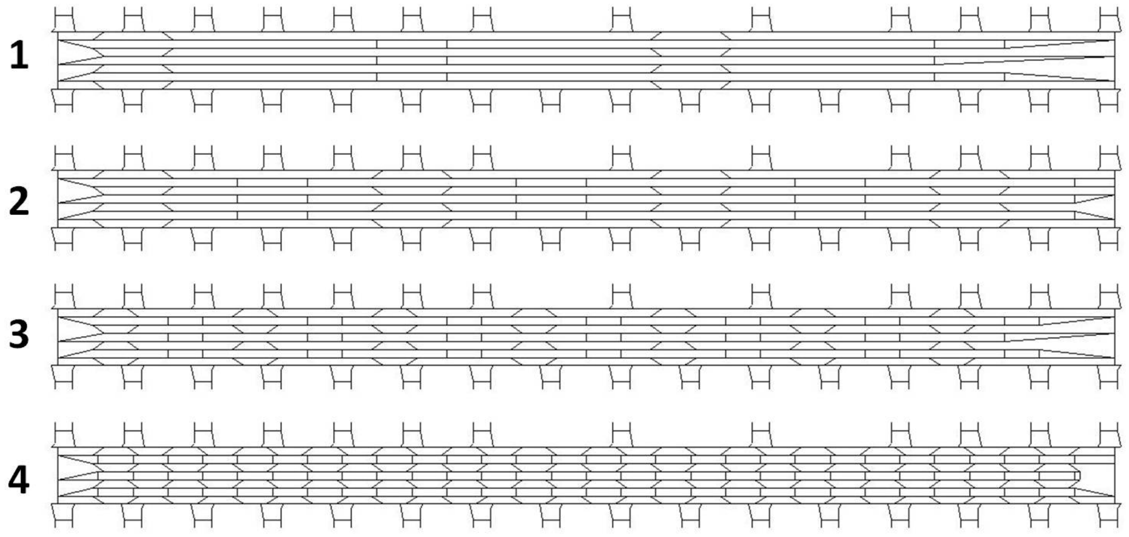
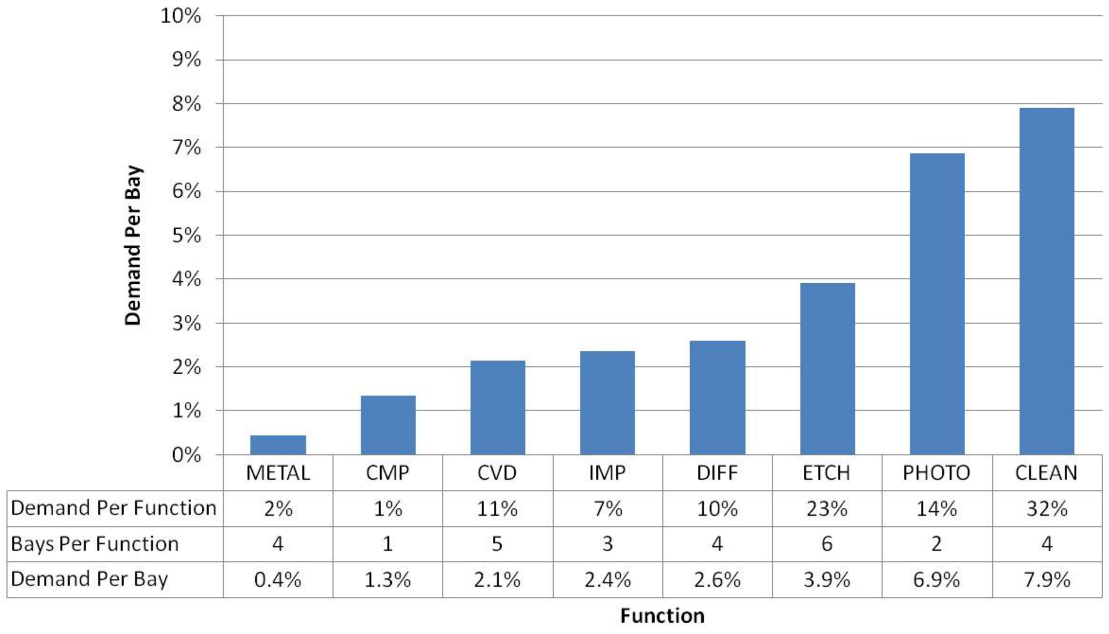


Figure 29: Center Loop Configurations

4.3.1.2 Request Generation Parameters

Bay-to-process assignment is shown in the table at the bottom of Figure 4.3.1.1. Process-to-process transition matrices are determined based upon prototype production sequences. All processes except PHOTO have the same side-track buffer and stocker usage parameters: cartridges visit a side-track-buffer between process steps 30% of the time and a stocker 20% of the time. For the PHOTO process, 70% go to side-track buffers and 20% to stockers. The bay-to-process assignment, transition matrices, and storage parameters result in the demand per bay shown in Figure 4.3.1.2.



	METAL	CMP	CVD	IMP	DIFF	ETCH	PHOTO	CLEAN
Demand Per Function	2%	1%	11%	7%	10%	23%	14%	32%
Bays Per Function	4	1	5	3	4	6	2	4
Demand Per Bay	0.4%	1.3%	2.1%	2.4%	2.6%	3.9%	6.9%	7.9%

Figure 30: Demand per Bay

4.3.1.3 Workload Settings

To test for sensitivity to the number of vehicles in the system, we consider scenarios with 400, 500, and 550 vehicles. As the number of vehicles increases, vehicle interaction causes more delay but in the absence of excessive congestion the system is able to deliver more requests. Request frequency parameters are set such that vehicle utilization for layout *1* with 400 vehicles is between 80% and 90% for both steady-state static and dynamic routing. We use these parameters for each scenario. We also present results for the 400 vehicle scenarios where request frequency is increased as average point-to-point distance decreases due to the addition of shortcuts.

4.3.1.4 Run Control Parameters

We establish a warm-up period of 1200s based on analysis of speed index and vehicle utilization in an initial set of replications of distance-based static routing. Speed index is defined as the average over all vehicles that should be moving (i.e. not stopped for loading/unloading and not idle and stationary) of the current velocity divided by the maximum velocity given a vehicle's location in the network. We graphically determine when the system reaches steady-state.

The number of replications is initially set at 20 to evaluate frequency of heavy congestion. With dynamic routing no heavy congestion occurs, so we consider all replications for steady-state analysis. Standard deviations are very low indicating that this number of replications is sufficient.

4.3.2 Results

For static routing, we present results that confirm what has been experienced in practice, that suboptimal layout design may increase the frequency of prohibitive congestion and delay. This significantly limits the subset of feasible layout options and prevents the inclusion of many short point-to-point routes. We then demonstrate that the use of dynamic routing eliminates heavy congestion even with the addition of a large number of shortcuts. The addition of these shortcuts reduces delivery and total time by 24%-26%. If we increase request frequency but not to the point of overload, 26% more requests can be accommodated in the simulation time for a fixed number of vehicles.

4.3.2.1 Static Routing Results

With static routing, if an edge is used on the shortest distance path for many origin/destination pairs the edge and adjacent areas will likely become overloaded. This is particularly true if frequent loading and unloading also occurs on that edge. We observe this to be the case in the simulation as shown in the simulation snapshot in

Figure 4.3.2.1.

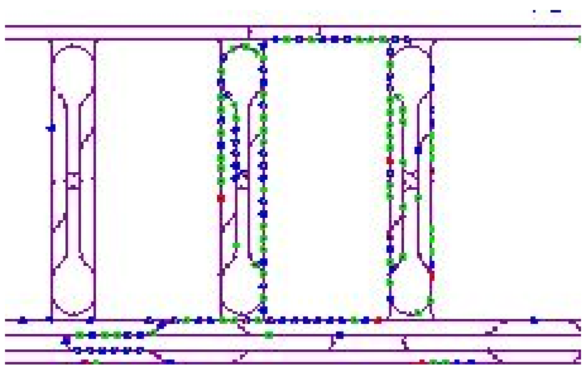
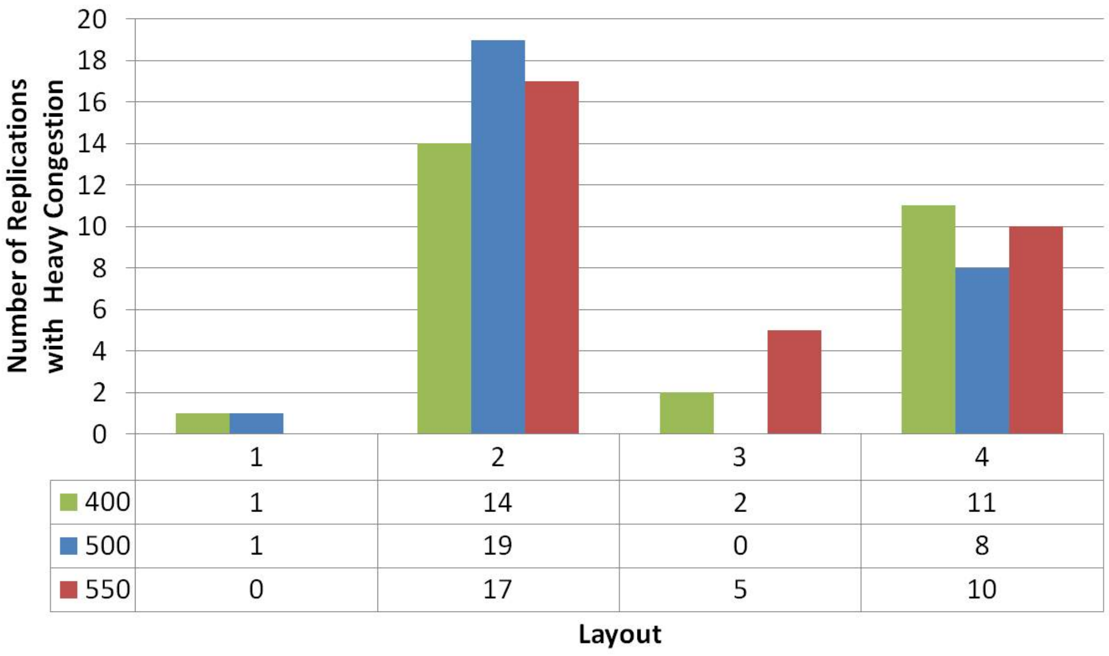


Figure 31: Snapshot of Heavy Congestion

Figure 4.3.2.1 shows graphically the frequency of heavy congestion for each scenario. A replication with heavy congestion is defined as having a speed index below 0.25 at the end of a replication. Note that with static routing, once a replication reaches heavy congestion it does not recover. Layout *2* experiences significantly more heavy congestion than the other layouts followed by layout *4*. To demonstrate why, Figure 4.3.2.1 shows shortest path flows weighted by the percent of requests using that path. Dark edges are used by highest percent of demand. In all layouts, high demand bays such as CLEAN and PHOTO have higher flow. In layout *2*, a higher percentage of requests cut through bays in the top row, specifically the leftmost of the high-demand bays. In layouts *1*, *3*, and *4* approximately 5% of the total demand uses the right lane in this bay, but in layout *2* 6.5% uses this lane. This is a 25% to 30% increase in traffic on this edge. Because this lane is in a bay, vehicles frequently stop here to load and unload. This combination of frequent stopping and additional cut-through traffic appears to overload this area of the network.



	1	2	3	4
🟩 400	1	14	2	11
🟦 500	1	19	0	8
🟥 550	0	17	5	10

Figure 32: Frequency of Heavy Congestion with Static Routing

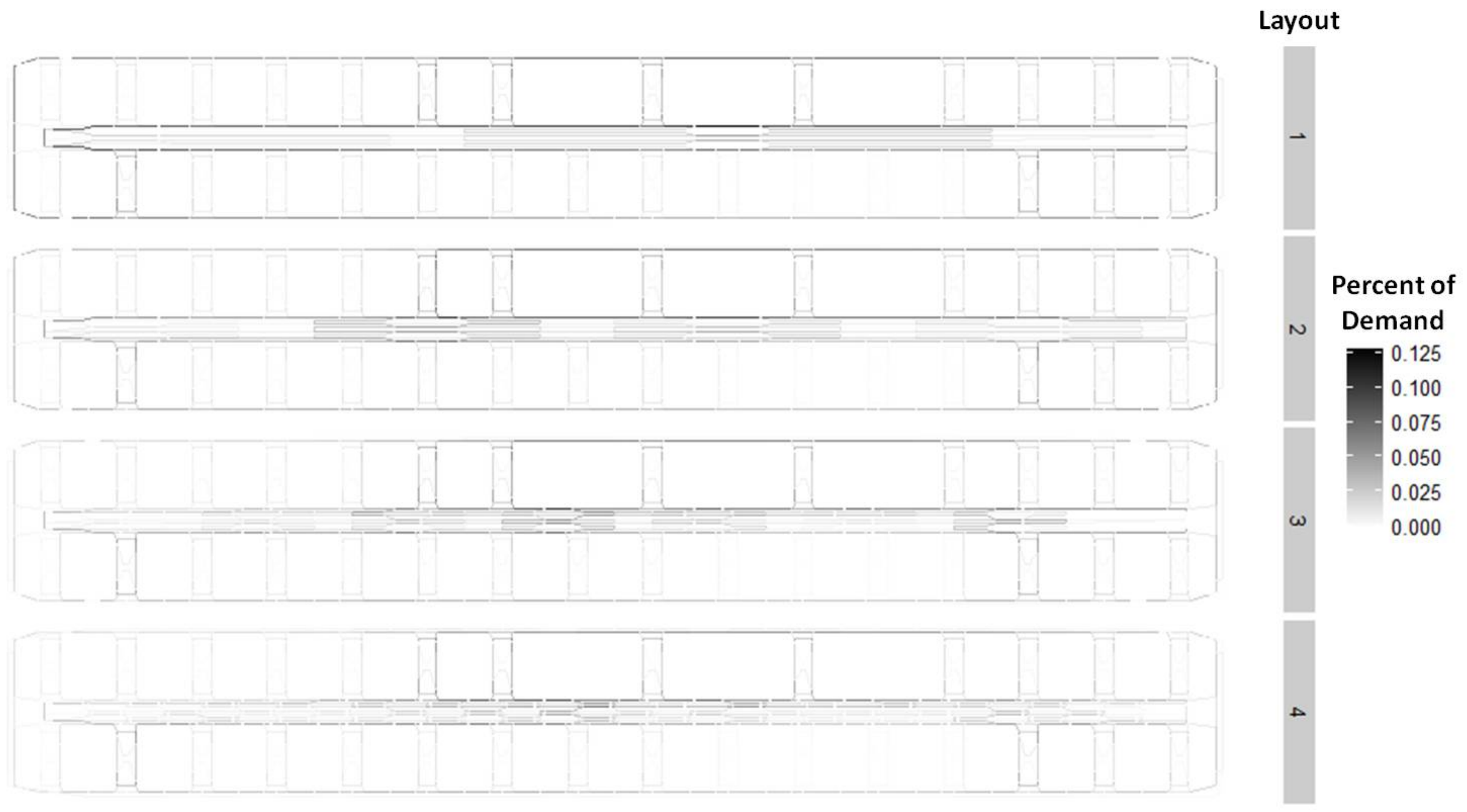


Figure 33: Weighted Shortest Path Flow

Figure 4.3.2.1 shows the locations where heavy congestion occurs which confirms the correlation between shortest path flow and the location of heavy congestion. Each cell contains a network map where black edges indicate that at the end of the simulation time vehicles were stopped on this edge. Every instance of heavy congestion in layout *2* involves the high-demand bays in the top row. Across layouts, only 6 of 88 replications that reach heavy congestion do not involve this part of the network, indicating that the combination of high demand bays in close proximity and vehicles cutting through these bays to reach the center loop often results in deadlock. These results highlight the idea that the placement of shortcuts and their impact on shortest path routing is significant with static routing.

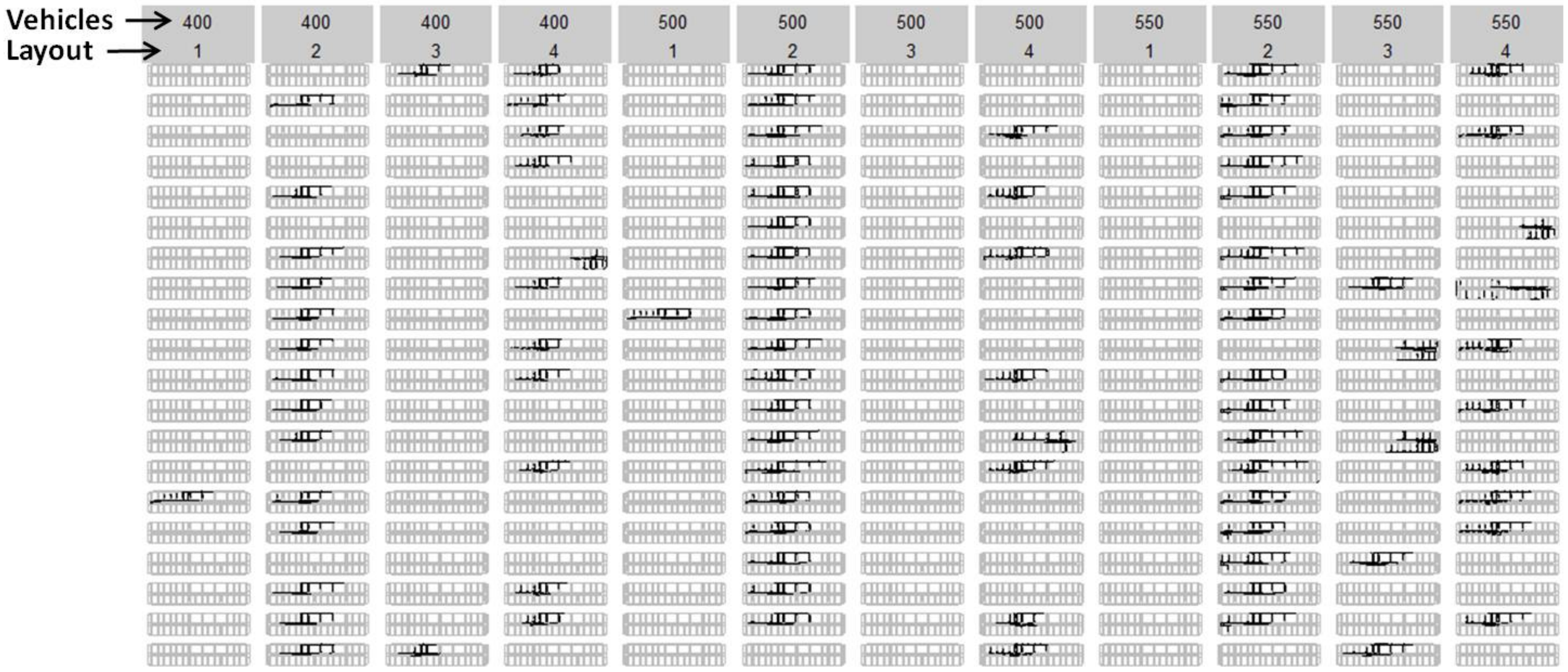


Figure 34: Location of Heavy Congestion

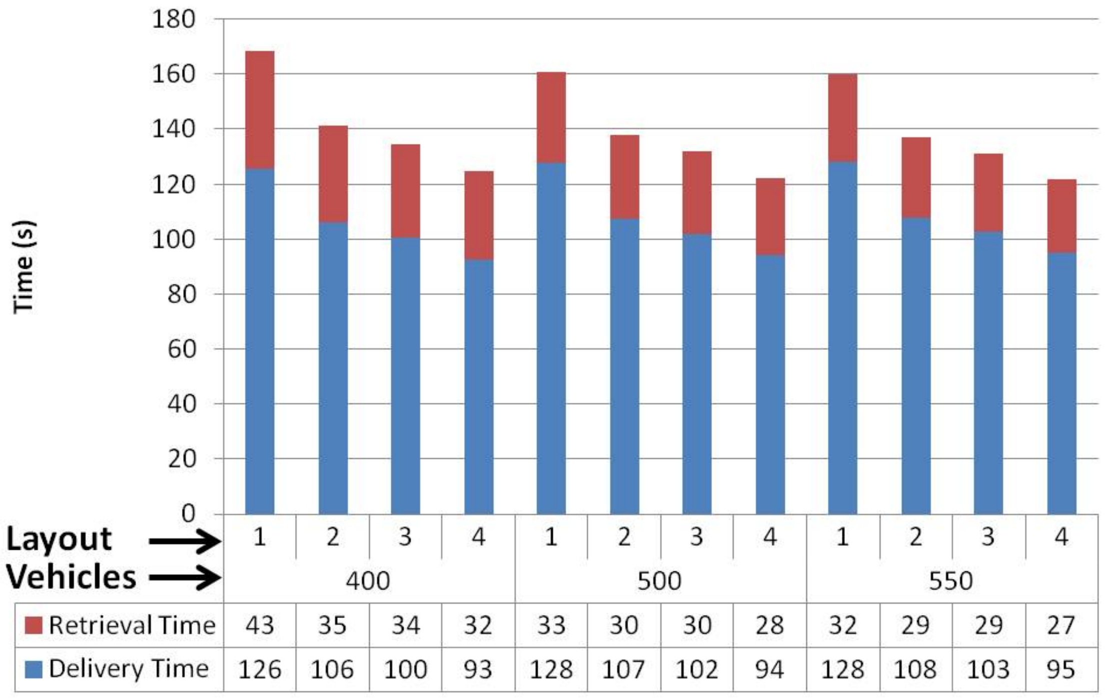
4.3.2.2 Dynamic Routing Results

Dynamic routing avoids the heavy congestion experienced with static routing. No heavy congestion occurs in 20 replications of each of the scenarios presented above. With dynamic routing, vehicles use the in-bay center lanes to avoid lanes with frequent loading and unloading and routes vehicles away from congested bays that they have no need to enter.

Since no heavy congestion occurs, the addition of shortcuts to reduce point-to-point distances becomes feasible with dynamic routing. Figure 4.3.2.2 shows the retrieval and delivery times for each scenario. As the number of shortcuts increase, from layout *1* to layout *4*, average delivery time decreases by 24%-26% and retrieval time by 15% - 24%. Note that retrieval times depend on vehicle utilization so they decrease between the 400 and 500 vehicle scenarios. Because vehicle utilization becomes quite low as more vehicles are added and request frequency parameters are held constant, there is diminishing return and little difference is seen between 500 and 550 vehicles. Note that confidence intervals are very tight on these values.

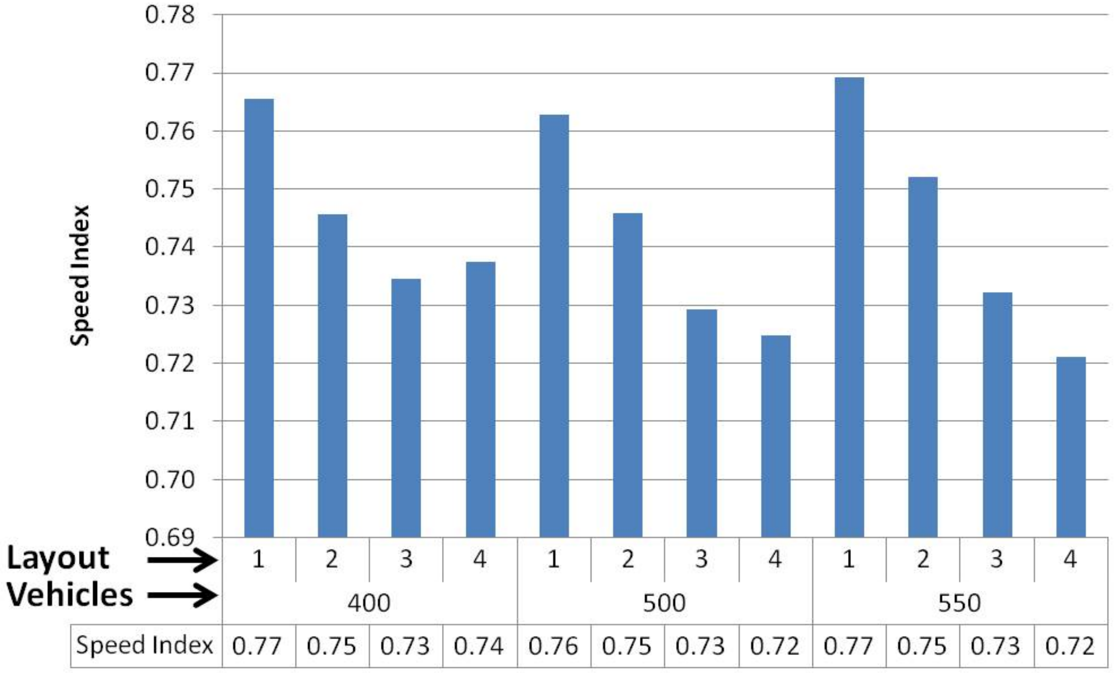
Delivery times may not always decrease with the addition of shortcuts as one would intuitively expect. If congestion increases too severely, delivery times may instead increase. We do see congestion increase as shortcuts are added. Figure 4.3.2.2 shows the speed index across scenarios. A higher speed index indicates that traffic is moving more smoothly. We see that the speed index decreases as the number of shortcuts increases except for 400 vehicles between layouts *3* and *4*. Interestingly, we do not see a consistent change in speed index across layouts as the number of vehicles increases.

With request frequency held constant, vehicle utilization drops as low as 52%. In practice, engineers typically target a vehicle utilization of 80% or higher. For this reason, we consider scenarios where request frequency increases as a function of point-to-point distance. This does not provide a consistent workload across scenarios,



Layout	1	2	3	4	1	2	3	4	1	2	3	4
Vehicles	400				500				550			
🟥 Retrieval Time	43	35	34	32	33	30	30	28	32	29	29	27
🟦 Delivery Time	126	106	100	93	128	107	102	94	128	108	103	95

Figure 35: Retrieval and Delivery Times with Dynamic Routing

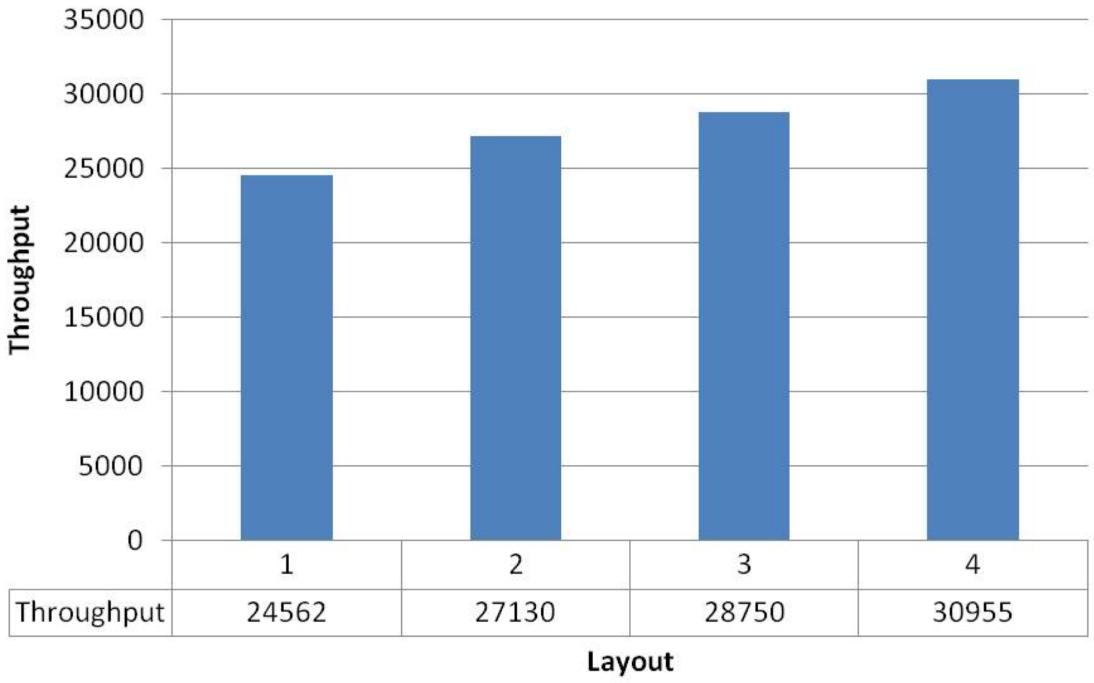


Layout	1	2	3	4	1	2	3	4	1	2	3	4
Vehicles	400				500				550			
Speed Index	0.77	0.75	0.73	0.74	0.76	0.75	0.73	0.72	0.77	0.75	0.73	0.72

Figure 36: Speed Index with Dynamic Routing

but targets a vehicle utilization between 80% and 90%. Ideally, the transportation throughput or number of requests delivered will increase as the request frequency increases but delivery times will not change. Because vehicle utilization is higher, retrieval time will increase.

Figure 4.3.2.2 shows the transportation throughput for each layout with 400 vehicles and increased request frequency. No heavy congestion occurs so these results represent the average across 20 replications and have very tight confidence intervals. Layouts *1* and *4* both have vehicle utilizations of 86%-87% but layout *4* is able to deliver 26% more requests due to shorter deliver times. Thus, the system capacity is much higher.



	1	2	3	4
Throughput	24562	27130	28750	30955

Figure 37: Transportation Throughput with Dynamic Routing and Increased Request Frequency

4.4 Conclusion

In this chapter, we investigated how rail layout performance differs between static and dynamic routing. With static routing, adding shortcuts may lead to prohibitive congestion and delay. With dynamic routing, vehicles are able to avoid overloading particular areas of the network while taking advantage of shorter point-to-point distances. Using an automated layout generation tool, we simulate four center and outer loop configurations to demonstrate that, with dynamic routing, the system continues to function in steady-state even with an eight-fold increase in the number of shortcuts. This improves routing performance by 25%.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

This thesis considers the issues of vehicle routing and layout design in large-scale unified automated material handling systems (AMHSs). We develop a high-fidelity simulation of vehicle movement that incorporates complexities of vehicle control and uncertainty in request generation. To allow flexible analysis, we develop a layout generation tool that automatically generates simulation layout files when the user selects a set of modular characteristics.

We propose a dynamic routing method that allows vehicles to be rerouted in progress in response to changes in the severity and location of congestion. Congestion is modeled using recent historical information and updated via exponential smoothing. We demonstrate, via simulation, that our dynamic routing method results in a 4%-6% improvement in steady-state routing performance, a significant reduction in the frequency of heavy congestion, and an improved response to and recovery from vehicle breakdown.

Having demonstrated the benefit of dynamic routing on a fixed layout, we consider the question of how to improve the layout to take advantage of dynamic routing. We demonstrate that dynamic routing allows an eight-fold increase in the number of shortcuts along the center and outer loop which results in a 35% improvement in steady-state routing performance and the elimination of the occurrence of prohibitive congestion.

Three areas of future research follow naturally from this thesis:

- dynamic routing in general AGV systems,
- integration of material handling decisions with other production decisions, and

- AMHS layout optimization and analytical modeling of dynamic routing.

We have demonstrated the benefits of using dynamic routing in the unified AMHSs used in semiconductor manufacturing. Automated guided vehicles systems, however, are varied in their characteristics and purposes. In particular, bidirectional travel lanes present a host of additional challenges. In our system, is it assumed that intersection control if first in, first out and that the this logic is controlled independently from the routing system via sensors. In bidirectional systems, routing and intersection control are often considered at the same time. For example, the routing system may preemptively request that a vehicle stop before entering an intersection to allow another vehicle to pass. In our system, the routing decision controls only which edge a vehicle selects when it has an option. In bidirectional systems and some other types of systems, many more decisions are controlled by the routing system. It would be an interesting and challenging questions to extend our approach to other specific system types as well as more general networks.

As we have discussed, in semiconductor AMHSs, production scheduling decisions are nearly independent of material handling decisions although they mutually depend on one another. It is likely that considering them together rather than in isolation will improve the performance of both systems. For example, if the material handling systems knows that in the near future a particular bay will be generating several new requests, a vehicle may preemptively avoid that bay on its route. Likewise, if the production scheduling systems knows that a particular bay is heavily congested when selecting a destination for a particular request, it may choose to select a destination in a different bay. Because the production scheduling system is itself so complex, to simulate it in detail would likely be computationally prohibitive. Thus, one must determine which production scheduling characteristics and decisions are most important and relevant when modeling the systems together.

In addition to production scheduling, better integrating routing with other aspects of AGV system design are an ongoing research questions. Equipment location, flow path or layout design, optimizing the number of vehicles, vehicle dispatching, inventory, and idle vehicle control are some of these characteristics. In addition, facilities continue to evolve as technology changes. Many facilities are multi-level now, using lifts to transport cartridges among floors, and some may be several buildings large. Each of these systems presents unique challenges for design and control. Overall, the end goal is throughput with consideration of constraints such as lot priority.

Lastly, having shown that layout can be improved for dynamic routing, a next question is what is and how do we find an optimal layout for use with dynamic routing. As we have discussed, static routing and dynamic routing behave different and result in different preferences for rail layout. Network design is a difficult problem with any realistic set of constraints, even with static routing. Because the problem is well-studied, we would like to be able to take advantage of existing methods, however, doing so requires the incorporation of dynamic routing into an integer programming framework. This is an open and challenging problem. Because of the operational complexity of the system, any optimization model will need to be validated through simulation. Because of the complexity, the most effective design frameworks may incorporate the iterative use of an optimization model with simulation.

REFERENCES

[1] Agrawal, G. and Heragu, S., "A Survey of Automated Material Handling Systems in 300-mm Semiconductor Fabs," *IEEE Transactions on Semiconductor Manufacturing*, vol. 19, pp. 112–120, Feb. 2006.

[2] Aiello, G., Enea, M., and Galante, G., "An integrated approach to the facilities and material handling system design," *International Journal of Production Research*, vol. 40, pp. 4007–4017, Jan. 2002.

[3] Anshelevich, E. and Ukkusuri, S., "Equilibria in Dynamic Selfish Routing," *Lecture Notes in Computer Science: Algorithmic Game Theory*, vol. 5814, pp. 171–182, 2009.

[4] Asef-Vaziri, A., "Evaluation of operational policies in the design phase of material handling networks," in *2007 Winter Simulation Conference*, pp. 1651–1654, IEEE, Dec. 2007.

[5] Bahri, N. and Gaskins, R., "Automated material handling system traffic control by means of node balancing," in *2000 Winter Simulation Conference Proceedings*, vol. 2, pp. 1344–1346, IEEE, 2000.

[6] Bahri, N., Reiss, J., and Doherty, B., "A comparison of unified vs. segregated automated material handling systems for 300 mm fabs," in *2001 IEEE International Symposium on Semiconductor Manufacturing Conference Proceedings*, pp. 3–6, IEEE, 2001.

[7] Bannister, M. and Eppstein, D., "Randomized Speedup of the Bellman–Ford Algorithm," in *Proceedings of the Ninth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, 2012.

[8] Bellman, R., "On a Routing Problem," *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, Dec. 1958.

[9] Chen, A., Zhou, Z., Chootinan, P., Ryu, S., Yang, C., and Wong, S. C., "Transport Network Design Problem under Uncertainty: A Review and New Developments," *Transport Reviews*, vol. 31, pp. 743–768, Nov. 2011.

[10] Chen, T. J., Sun, Y., Dai, W., Tao, W., and Liu, S., "On the Shortest and Conflict-Free Path Planning of Multi-AGV System Based on Dijkstra Algorithm and the Dynamic Time-Window Method," *Advanced Materials Research*, vol. 645, pp. 267–271, Mar. 2013.

[11] Christopher, J., Kuhl, M., and Hirschman, K., "Simulation analysis of dispatching rules for automated material handling systems and processing tools in semiconductor fabs," in *ISSM 2005, IEEE International Symposium on Semiconductor Manufacturing, 2005.*, pp. 84–87, IEEE, 2005.

[12] Dijkstra, E. W., "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, Dec. 1959.

[13] Drira, A., Pierreval, H., and Hajri-Gabouj, S., "Facility layout problems: A survey," *Annual Reviews in Control*, vol. 31, pp. 255–267, Jan. 2007.

[14] Floyd, R. W., "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5, p. 345, June 1962.

[15] Fukunari, M., Rajanna, S., Gaskins, R., and Sparrow, M., "Data-based node penalties in a path-finding algorithm in an automated material handling system," in *Proceedings of the Winter Simulation Conference*, vol. 2, pp. 1383–1386, IEEE, 2002.

[16] Gaskins, R. J. and Tanchoco, J. M. A., "Flow path design for automated guided vehicle systems," *International Journal of Production Research*, vol. 25, pp. 667–676, May 1987.

[17] Gaskins, R., Mariano, T., and Sparrow, M., "Patent US6285951 - Dynamic traffic based routing algorithm," 2001.

[18] Guan, X., Baijing, Q., and Yang, H., "Integrated RMS Layout and Flow Path Design: Modelling and a Heuristic method," in *Lecture Notes in Electrical Engineering: Computer, Informatics, Cybernetics and Applications* (He, X., Hua, E., Lin, Y., and Liu, X., eds.), vol. 107 of *Lecture Notes in Electrical Engineering*, pp. 403–411, Dordrecht: Springer Netherlands, 2012.

[19] Han, C., Pare, K., Tokumoto, M., and Aoki, A., "High Throughput AMHS Design with Dual Unified OHT System," in *2006 IEEE International Symposium on Semiconductor Manufacturing*, pp. 185–188, IEEE, Sept. 2006.

[20] Ho, Y.-C. and Liao, T.-W., "Zone design and control for vehicle collision prevention and load balancing in a zone control AGV system," *Computers & Industrial Engineering*, vol. 56, pp. 417–432, Feb. 2009.

[21] Huang, C.-J., Chang, K.-H., and Lin, J. T., "Optimal vehicle allocation for an Automated Materials Handling System using simulation optimisation," *International Journal of Production Research*, vol. 50, pp. 5734–5746, Oct. 2012.

[22] Huang, C.-W., Chen, H.-Y., Yu, R.-C., and Yu, C.-Y., "Patent US7356378 - Method and system for smart vehicle route selection," 2008.

[23] Huang, J., Palekar, U., and Kapoor, S., "A labeling algorithm for the navigation of automated guided vehicles," *Journal of Engineering for Industry*, vol. 115, pp. 315–321, Aug. 1993.

[24] ITRS, "International Technology Roadmap for Semiconductors," Retrieved from ITRS website at www.itrs.net/reports.html. 2013.

[25] Kaspi, M., Kesselman, U., and Tanchoco, J. M. A., "Optimal solution for the flow path design problem of a balanced unidirectional AGV system," *International Journal of Production Research*, vol. 40, pp. 389–401, Jan. 2002.

[26] Kaspi, M. and Tanchoco, J., "Optimal flow path design of unidirectional AGV systems," *International Journal of Production Research*, vol. 28, no. 5, pp. 1023–1030, 1990.

[27] Kazmierski, C., "Semiconductor Industry Posts Record-Breaking Revenues Despite 2011 Challenges," Retrieved from Semiconductor Industry Association website www.semiconductors.org. 2012.

[28] Kim, B.-I., Jeong, S., Shin, J., Koo, J., Chae, J., and Lee, S., "A Layout- and Data-Driven Generic Simulation Model for Semiconductor Fabs," *IEEE Transactions on Semiconductor Manufacturing*, vol. 22, pp. 225–231, May 2009.

[29] Kim, B.-I., Oh, S., Shin, J., Jung, M., Chae, J., and Lee, S., "Effectiveness of vehicle reassignment in a large-scale overhead hoist transport system," *International Journal of Production Research*, vol. 45, pp. 789–802, Feb. 2007.

[30] Kim, B.-I. and Park, J., "Idle vehicle circulation policies in a semiconductor FAB," *Journal of Intelligent Manufacturing*, vol. 20, pp. 709–717, Aug. 2008.

[31] Kim, B.-I., Shin, J., and Chae, J., "Simple blocking prevention for bay type path-based automated material handling systems," *The International Journal of Advanced Manufacturing Technology*, vol. 44, pp. 809–816, Dec. 2008.

[32] Kim, C. and Tanchoco, J., "Conflict-free shortest-time bidirectional AGV routeing," *International Journal of Production Research*, vol. 29, no. 12, pp. 2377–2391, 1991.

[33] Kim, C. W. and Tanchoco, J., "Operational control of a bidirectional automated guided vehicle system," *International Journal of Production Research*, vol. 31, no. 9, pp. 2123–2138, 1993.

[34] Kleeman, M. P., Lamont, G. B., Hopkinson, K. M., and Graham, S. R., "Solving Multicommodity Capacitated Network Design Problems using a Multiobjective Evolutionary Algorithm," in *2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications*, pp. 33–41, IEEE, Apr. 2007.

[35] Lamar, B. W., Sheffi, Y., and Powell, W. B., "A lower bound for uncapacitated, multicommodity fixed charge network design problems," tech. rep., University of California - Irvine, Irvine, CA, 1987.

[36] Lau, H. and Woo, S., "An agent-based dynamic routing strategy for automated material handling systems," *International Journal of Computer Integrated Manufacturing*, vol. 21, no. 3, pp. 269–288, 2008.

[37] Le-Anh, T. and De Koster, M., "A review of design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 171, pp. 1–23, May 2006.

[38] Lim, J. K., Lim, J. M., Yoshimoto, K., Kim, K. H., and Takahashi, T., "A construction algorithm for designing guide paths of automated guided vehicle systems," *International Journal of Production Research*, vol. 40, pp. 3981–3994, Jan. 2002.

[39] Lin, J. T., Wang, F. K., and Wu, C. K., "Connecting transport AMHS in a wafer fab," *International Journal of Production Research*, vol. 41, pp. 529–544, Jan. 2003.

[40] Lin, J. T., Wang, F.-K. W. F.-K., and Wu, C.-K. W. C.-K., "Simulation analysis of the connecting transport AMHS in a wafer fab," *IEEE Transactions on Semiconductor Manufacturing*, vol. 16, pp. 555–564, 2003.

[41] Lin, J. T., "Dynamic vehicle allocation in Automated Material Handling System," in *2010 IEEE 17Th International Conference on Industrial Engineering and Engineering Management*, pp. 1523–1527, IEEE, Oct. 2010.

[42] Mackulak, G. and Savory, P., "A simulation-based experiment for comparing AMHS performance in a semiconductor fabrication facility," *IEEE Transactions on Semiconductor Manufacturing*, vol. 14, no. 3, pp. 273–280, 2001.

[43] Magnanti, T. L. and Wong, R. T., "Network Design and Transportation Planning: Models and Algorithms," *Transportation Science*, vol. 18, pp. 1–55, Feb. 1984.

[44] Miller, L., Bradley, A., Tish, A., Jin, T., Jimenez, J. A., and Wright, R., "Simulating conveyor-based amhs layout configurations in small wafer lot manufacturing environments," in *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pp. 1939–1947, IEEE, Dec. 2011.

[45] Montoya-Torres, J. R., "Internal transport in automated semiconductor manufacturing systems," *4OR*, vol. 5, pp. 93–97, July 2006.

[46] Nazzal, D., Jimenez, J. A., Carlo, H. J., Johnson, A. L., and Lasrado, V., "An Analytical Model for Conveyor-Based Material Handling System With Crossovers in Semiconductor Wafer Fabs," *IEEE Transactions on Semiconductor Manufacturing*, vol. 23, pp. 468–476, Aug. 2010.

[47] Ng, A. K., Efstathiou, J., and Lau, J. Y., "A Load Scattering Algorithm for Dynamic Routing of Automated Material Handling Systems," in *Computational Intelligence and Security* (Wang, Y., Cheung, Y.-m., and Liu, H., eds.), vol. 4456 of *Lecture Notes in Computer Science*, pp. 704–713, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[48] Nishi, T., Ando, M., and Konishi, M., "Experimental studies on a local rescheduling procedure for dynamic routing of autonomous decentralized AGV systems," *Robotics and Computer-Integrated Manufacturing*, vol. 22, pp. 154–165, Apr. 2006.

[49] Nishi, T. and Maeno, R., "Petri Net Decomposition Approach to Optimization of Route Planning Problems for AGV Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 7, pp. 523–537, July 2010.

[50] Nishi, T. and Tanaka, Y., "Petri Net Decomposition Approach for Dispatching and Conflict-Free Routing of Bidirectional Automated Guided Vehicle Systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, pp. 1230–1243, Sept. 2012.

[51] Paprotny, I. and Mackulak, G., "Simulation based comparison of semiconductor AMHS alternatives: continuous flow vs. overhead monorail," in *2000 Winter Simulation Conference Proceedings*, vol. 2, pp. 1333–1338, IEEE, 2000.

[52] Qiu, L., Hsu, W.-J., Huang, S.-Y., and Wang, H., "Scheduling and routing algorithms for AGVs: A survey," *International Journal of Production Research*, vol. 40, pp. 745–760, Jan. 2002.

[53] Rezapour, S., Zanjirani-Farahani, R., and Miandoabchi, E., "A machine-to-loop assignment and layout design methodology for tandem AGV systems with single-load vehicles," *International Journal of Production Research*, vol. 49, pp. 3605–3633, June 2011.

[54] Roughgarden, T., "On the severity of Braess's Paradox: Designing networks for selfish users is hard," *Journal of Computer and System Sciences*, vol. 72, pp. 922–953, Aug. 2006.

[55] Sedehi, M. S. and Farahani, R. Z., "An integrated approach to determine the block layout, AGV flow path and the location of pick-up/delivery points in single-loop systems," *International Journal of Production Research*, vol. 47, pp. 3041–3061, June 2009.

[56] Seifert, R., Kay, M., and Wilson, J., "Evaluation of AGV routeing strategies using hierarchical simulation," *International Journal of Production Research*, vol. 36, no. 7, pp. 1961–1976, 1998.

[57] Sha, D. Y. and Yang, C. J., "The transport strategies for fully automated manufacturing in 300 mm wafer fab," *International Journal of Computer Integrated Manufacturing*, vol. 22, pp. 962–975, Oct. 2009.

[58] Sherali, H. D., Ozbay, K., and Subramanian, S., "The time-dependent shortest pair of disjoint paths problem: Complexity, models, and algorithms," *Networks*, vol. 31, pp. 259–272, July 1998.

[59] Smolic-Rocak, N., Bogdan, S., Kovacic, Z., and Petrovic, T., "Time Windows Based Dynamic Routing in Multi-AGV Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 7, pp. 151–155, Jan. 2010.

[60] Sneyers, J., Schijvers, T., and Demoen, B., "Dijkstra's Algorithm with Fibonacci Heaps: An Executable Description in CHR," *Workshop on Logic Programming*, vol. 6, pp. 182–191, 2006.

[61] Stenzel, B., *Online Disjoint Vehicle Routing with Application to AGV Routing.* PhD thesis, Technischen Universitat Berlin, 2008.

[62] Taghaboni, F. and Tanchoco, J. M. A., "A LISP-based controller for free-ranging automated guided vehicle systems," *International Journal of Production Research*, vol. 26, pp. 173–188, Feb. 1988.

[63] Tanchoco, M. A. and Taghaboni-Dutta, F., "Comparison of dynamic routeing techniques for automated guided vehicle system," *International Journal of Production Research*, vol. 3, no. 10, pp. 2653–2669, 1995.

[64] ter Mors, A. W., Witteveen, C., Zutt, J., and Kuipers, F. A., "Context-Aware Route Planning," in *Lecture Notes in Computer Science* (Dix, J. and Witteveen, C., eds.), vol. 6251 of *Lecture Notes in Computer Science*, pp. 138–149, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[65] Vis, I. F., "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, pp. 677–709, May 2006.

[66] Wang, F. and Lin, J., "Performance evaluation of an automated material handling system for a wafer fab," *Robotics and Computer-Integrated Manufacturing*, vol. 20, pp. 91–100, Apr. 2004.

[67] Wang, I.-L., Johnson, E. L., and Sokol, J. S., "A Multiple Pairs Shortest Path Algorithm," Nov. 2005.

[68] Wong, R. T., "A Survey of Network Design Problems," *Operations Research Center Working Paper*, vol. OR 080-78, Aug. 1978.

[69] World Semiconductor Trade Statistics, "WSTS Semiconductor Market Forecast Spring 2012 ," Retrieved from World Semiconductor Trade Statistics website at www.wsts.org. 2012.

[70] Yang, J.-W., Cheng, H.-C., Chiang, T.-C., and Fu, L.-C., "Multiobjective lot scheduling and dynamic OHT routing in a 300-mm wafer fab," in *2008 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1608–1613, IEEE, Oct. 2008.