POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations

Joshua Auld, Michael Hope, Hubert Ley, Vadim Sokolov, Bo Xu, Kuilin Zhang

Abstract

This paper discusses the development of an agent-based modeling software development kit, and the implementation and validation of a model using it that integrates dynamic simulation of travel demand, network supply and network operations. A description is given of the core utilities in the kit: a parallel discrete event engine, interprocess exchange engine, and memory allocator, as well as a number of ancillary utilities: visualization library, database IO library, and scenario manager. The overall framework emphasizes the design goals of: generality, code agility, and high performance. This framework allows the modeling of several aspects of transportation system that are typically done with separate stand-alone software applications, in a high-performance and extensible manner. The issue of integrating such models as dynamic traffic assignment and disaggregate demand models has been a long standing issue for transportation modelers. The integrated approach shows a possible way to resolve this difficulty. The simulation model built from the POLARIS framework is a single, shared-memory process for handling all aspects of the integrated urban simulation. The resulting gains in computational efficiency and performance allow planning models to be extended to include previously separate aspects of the urban system, enhancing the utility of such models from the planning perspective. Initial tests with case studies involving traffic management center impacts on various network events such as accidents show the potential of the system.

1. Introduction

There are several confluent factors in the transportation community which have influenced the need for the development of a new kind of tool to answer questions about the transportation system. Historically, existing transportation-related models have looked at different aspects of the transportation system (travel demand, traffic flows, emissions, etc.) independently from each other. When the realization developed with the transportation community that these phenomena needed to be

Corresponding author. Tel.: +1 630 252 5460.
E-mail addresses: jauld@anl.gov (J. Auld), bo.5.xu@here.com (B. Xu), klzhang@mtu.edu (K. Zhang).

Senior Researcher. Office: +1 (872) 208 9048, fax: +1 (312) 894 3965.

Assistant Professor. Tel.: +906 487 1828.

http://dx.doi.org/10.1016/j.trc.2015.07.017
0968-090X/ © 2013 Elsevier Ltd. All rights reserved.
modeled in an integrated manner, attempts were made to link these unrelated models into a unified system. The integrated solutions produced have frequently been either inflexible, non-modular, or low performance. There was a need for the different models to inter-operate with one another to answer these questions about a transportation system, but transportation models generally lacked a common framework to do so in a straightforward manner. In addition, as the complexity of transportation models increases and more and more agent behaviors and system operator responses are being simulated, there is an increased need for higher performance travel modeling tools. The research introduced in this paper grew out of the need to address these topics of interest in the transportation field: modeling integrated systems and addressing interoperability among current and future transportation algorithms.

Transportation system modeling covers a wide array of emerging and existing methodologies. The range of solutions employed in one region may look quite different from those in another; in addition, new technologies are being continuously developed and evaluated. This variability introduces a certain difficulty in providing a “black box” solution to the question of system operations and planning. It was decided that a modular and extensible software system which could be re-developed to add or modify such technologies provided the most tractable overall solution. Our goal is not only to develop a simulation framework that covers all major aspects of a transportation system but also to implement it in a computationally efficient way so that operational-level characteristics: vehicle-to-vehicle technologies, high resolution road sensors, advanced traveler information systems (ATIS), and others can be simulated faster than real-time.

The POLARIS project, then, was undertaken to create a model development system which addresses the following design goals:

- **Generality**: Provide agent based modeling tools which are optimized for transportation modeling.
- **Code Agility**: Use advanced software engineering techniques and well-defined structural guidelines to allows models to be rapidly adapted to new emerging applications.
- **High Performance**: Choice of a low level a programming language, low level optimization, and intense focus on multithreading.
- **Develop an integrated demand-network model that utilized the agent-based approach and is high performance.**

In order to address the needs identified above, the research team created an agent-based modeling software called POLARIS. A part of the POLARIS is the set of low level software libraries that provide a usable application programming interface (API), common run time environment that facilitates development of transportation modeling software. Then we build on the low level libraries to implement the network and demand components of the transportation system. The use of an agent-based modeling approach to develop an integrated transportation system model allows us to overcome some of the limitations of traditional aggregated transportation models, particularly with respect to sensitivity to behavioral aspects of the travelers. The benefit of this approach was demonstrated through the implementation of an integrated travel-demand and network operations agent-based microsimulation model. The simulation model includes demand estimation, network simulation and ITS operation components. A key aspect of these various model components is that they do not exist as stand-alone models, but are implemented as agent capabilities in the general agent-based framework.

The main contribution of the newly developed system is the capability to model large scale transportation networks using an approach that integrates demand and network modeling aspects into a single system that is implemented in a computationally efficient manner using persistent agents. The computational efficiency is achieved via using multithreading with a shared memory implementation and utilizes modern computer hardware architecture. Integrated transportation model requires large quantities of data being changed among multiple model component frequently. Implementations that rely on slower memory like hard drives are very restrictive and lead to long run times as hard drive access can be orders of magnitude slower than memory access. We alleviate the issue by relying on fast shared memory approach, which not only improves performance, but allows the full range of agent characteristics and context to be considered at every step of the model – including during traffic simulation.

The paper is organized as follows. In Section 2, we give a general background on agent-based modeling frameworks and integration efforts in transportation modeling is discussed. Section 3 introduces the underlying POLARIS agent-based model development toolkit. Section 4, provides results of numerical experiments that demonstrate the computational efficiency of the model. Section 5 discusses future directions for both the development of the framework and the refinement of the implementation for Chicago.

### 2. Background

The agent-based modeling literature contains some excellent examples of general purpose frameworks including REPAST (Collier, 2003; Collier et al., 2003) and MASON (Luke et al., 2004). A multi-agent programming language and modeling environment NetLogo (Tisue and Wilensky, 2004), allows quick prototyping and a low learning curve, thus it is widely applicable for small scale systems, especially for education purposes. The lack of scalability of NetLogo prevents it from using for even medium-size transportation systems. Another ABM system, Swarm (Minar et al., 2012), used to be at the frontier among other complex modeling tools, however, the system is not being developed since 2002 and thus is not applicable to any real-life applications. REPAST was developed in a middle ground between two other agent-based modeling packages:
Swarm and NetLogo, and is used for general agent-based model development. It is more scalable than NetLogo and is actively developed. More recently, a high-performance version REPast HPC was released. However, general structures of the REPast HPC model and the fact that it relies on Message Passing Interface (MPI) assumes a steep learning curve by a researcher. Our goal is to build on the work done by REPast HPC and NetLogo teams to achieve wide accessibility of agent-based modeling system within the transportation community. The Java-based discrete-event multivalent simulation library MASON is a good example of middle ground between general and scalable REPast and easy to use NetLogo. However, the MASON package is not being actively developed, and it is not clear whether it will be maintained in the future. For an overview and comparison of some of the more common general agent-based modeling packages see Railsback et al. (2006).

In the transportation field, several projects stand out as having similar goals involving the development of general modeling frameworks, among them are: OPUS (Waddell et al., 2005), MATSIM (Balmer et al., 2009), and TRANSIMS (Nagel et al., 1999; TRANSIMS, 2012). MATSim (multiagent micro-simulation) is an open-source transportation modeling framework which establishes a common platform for: demand modeling, traffic flow simulation, replanning, and a controller for iterative simulation runs. MATSim considers each modeled person as an agent and simulates both demand and vehicular traffic flow for this agent. In Ziliaskopoulos and Waller (2000), the authors describe a framework that facilitates collaborative work on a set of transportation models via an internet-based GIS application and emphasizes data management and representation. Dia (2002) and Wahle et al. (2002) showed how the agent-based approach can be applied to model route choice behavior and impact of real-time information on it. Arentze et al. (2010) develop agent-based methodology to model activity generation and location choices, taking dynamic nature of the behavior into account. Chen et al. (2009) developed Mobile-C system, which is an agent-based real-time traffic detection and management system (ABRTTDM), it allows to simulate distributed traffic detection and management systems. A multi-agent transportation management system and route guidance system developed by Adler and Blue (2002) allows to efficiently redistribute the traffic flows so that roads are used more efficiently. Beyond traffic and travel behavior models, agent-based approach was also applied in transportation community to model crowd motion. Duives et al. (2013) provided an overview of agent-based approaches to simulate dynamics of crowds. In particular, they focused on the research done in the last decade. Another popular application of the approach is to study transportation systems from a logistical perspective, Davidsson et al. (2005) provide a survey of research that focuses on freight transportation.

The models utilized in the field of travel demand modeling have grown increasingly complex as the usage of such models has shifted away from analyzing capacity and toward a focus on transportation demand and system management. This new policy environment has led to advances in the field such as activity-based modeling and dynamic traffic assignment, as well as simulation tools for evaluating network operations. The various tools for evaluating activity-based travel demand, dynamic network simulations and operations have proven useful in policy analysis, however, the full range of responses to operations have not previously been investigated as a tool which can look at both demand and travel responses in a holistic manner that has not previously been available.

Since the initial development and deployment of Intelligent Transportation Systems there has been a need to evaluate the impacts of ITS investment on the transportation system, as for any other system investment, which has often been done using simplified sketch planning tools that rely on existing macro model demand, such as IDAS (McHale, 2000), or a Cube-based ITS tool developed for Florida DOT (Hadi et al., 2008). Also, USDOT developed the integrated corridor management analysis, modeling, and simulation (ICM AMS) methodologies that assume combining macroscopic travel demand model with meso/macroscopic traffic flow simulation to test impact of different ITS technologies (Alexiadis, 2008). Another approach used is to combine existing general purpose modeling tools to evaluate limited aspects of the system (Cragg and Demetsky, 1995; Barcelo et al., 2004; Tettamanti et al., 2008; Hu et al., 2003).

Dynamic traffic assignment is often the core methodology for both off-line evaluation of intelligent traffic operational planning and on-line traffic control and management of the Intelligent Transportation Systems. Peeta and Ziliakopoulos (2001) provide an excellent survey of the DTA models in terms of analytical and simulation-based approaches, and point out that the simulation-based DTA models have gained greater acceptability in the context of real-world deployment. There are two software tools that use DTA-based approach for the purpose of evaluating the impacts of traffic management systems design, namely DynaMIT (Ben-Akiva et al., 2002) and Dynasmart (Mahmassani et al., 1993). Both DynaMIT and Dynasmart come in two variations: on-line and off-line and were developed in late 90s. Both DynaMIT and Dynasmart share similar functionality: (i) estimation of traffic conditions, (ii) adjustment of OD tables as a result of management strategy, (iii) prediction of network conditions as a result of changed demand and network controls. There are several management strategies supported in DynaMIT and Dynasmart, such as information dissemination, lane control, and intersection control. There are many nuances that distinguish those models, for example, the way traveler’s response to information is modeled, the algorithms for predicting change in OD tables, support for congestion pricing in Dynasmart, for more details see (FHWA, 2007; Ben-Akiva et al., 2002). Another similar tool MITSIMLab (Yang et al., 2000) allows a simulation based evaluation of Advanced Traffic Management Systems (ATMS) and Advanced Traveler Information Systems (ATIS). Unlike DynaMIT and Dynasmart it is not based on a mesoscopic traffic flow simulation and uses microscopic simulator MITSIM combined with traffic management simulator. However, there is no demand prediction available in MITSIMLab.

In parallel with the development of advanced dynamic traffic assignment models capable of evaluating ITS impacts, there has been much advancement in understanding individual activity travel behavior, through developments in activity-based modeling. Activity-based models based on the DaySIM (Bowman and Ben-Akiva, 2001) or CT-RAMP (Vovsha et al., 2011) models have been implemented by various regional planning agencies, and many models, such as FAMOS (Pendyala
et al., 2005), TASHA (Miller and Roorda, 2003), CEMDAP (Bhat et al., 2004), and ALBATROSS (Arentze and Timmermans, 2000) have been developed by the academic community. The models are intended to simulate the impact policies will have on individual travel behavior and with their focus on the underlying processes of activity-travel demand, the models could extend the understanding and modeling of demand level responses to ITS operations. As the models are generally developed separately from DTA, it is often the case that a complex activity-based model will be used to estimate demand, and the demand will be simply aggregated to OD tables and fed to a static traffic assignment routing. For this reason, the integration of travel demand models and traffic assignment models has received much attention recently. Several activity-based models have recently been integrated with a dynamic traffic assignment routine. Castiglione et al. (2010) integrated DaySim and TRANSIMS router. Lin et al. (2008) integrated CEMDAP with Visual Interactive System for Transport Algorithms (VISTA). Hao et al. (2010) integrated TASHA with MATSim. Pendyala et al. (2012) developed the SimTRAVEL model which integrates land use, travel demand and DTA models. Finally, the MATSim project represents both an integration between the demand and network simulation aspects as well as an agent-based modeling framework (Balmer et al., 2009).

It should be noted that the reported integration efforts, outside of MATSim, are often handled by establishing connections between disparate model components which pass information between themselves at intervals through file-sharing, inter-process communications, etc. The POLARIS model, then, differs from other existing model integration efforts in that is a completely integrated agent-based model, with individual agents persisting throughout the simulation, where the traditional activity-based model components and network routing and simulation components are implemented as agent-behaviors of persistent agents, with all of the agents operating in a shared memory space. Next, the design of the Polaris agent-based modeling software development toolkit, which builds on ideas from many past research project such as OPUS, TRANSIMS, MATSIM, and Repast, and is optimized for implementing integrated simulation models, is discussed.

3. Polaris agent-based model software development kit

The integrated activity based model discussed in this paper, was created using the concurrently developed POLARIS agent-based model development framework. The framework consists of two major conceptual components. First, a high level "agent-based" modeling framework targeted specifically at transportation which can be utilized to succinctly describe elements of a transportation system. Second, it has an SDK (software development kit) which facilitates the development, execution, and review of a model written in such a language. This framework allows the user to develop an integrated simulation of a transportation system in a standardized, extensible manner. The framework creates an open source model development environment which helps to develop and enforce transportation modeling standards and protocols. The goal of the framework is to connect sub-communities with a common modeling platform within this model development environment.

In order to collect disparate transportation models within this common framework effectively, it is important to properly organize model parts into tiers based on their level of generality so that the portions of the model which are used most frequently will be changed the least often and in the most predictable manner (see Fig. 1). At the lowest level are the POLARIS

![Fig. 1. The POLARIS model development framework.](image-url)
core libraries including the discrete event engine, memory allocator, inter-process engine, and coding conventions. These are tightly controlled and in the hands of just a few developers. Just above these are the fundamental extensions which are a set of generally useful utilities, for visualization, data I/O, etc. Both of these layers are general enough to be used across any model (even outside the realm of transportation) which can be written in terms of discrete events. The following sections will discuss these more general libraries and tools which comprise the framework.

Next is what is termed the Repository – this would include model fragments which are well tested and have a proven level of generality – for example, well-established routing algorithms, comprehensive interfaces for links, intersections, or vehicles. It is a goal that the Repository be versioned and extended systematically so that external developers can rely on its content. Above this level is the POLARIS user space – here developers can experiment with new model extensions and test niche functionalities. Finally, using elements from all layers, developers can weave together an application, of which the POLARIS ABM discussed herein is an example.

3.1. Simulation-oriented memory allocator

Memory allocation is an important topic in high performance software design, and provides substantial opportunities for low-level optimization of the system. Excessive dynamic memory allocation can have a detrimental effect on program performance for several reasons. First, unlike many other common operations, memory allocation needs to invoke the underlying operating system. Second, while CPUs and GPUs have advanced rapidly in performance, memory has lagged behind (Drepper, 2007). Additionally, orchestrating memory de-allocation so that it does not cause a conflict within a multi-threaded simulation is a non-trivial task.

The POLARIS memory allocator utilizes management techniques which are targeted at improving performance for agent-based simulations running in a discrete event paradigm. Agent-based code can differ from other types of applications as there is a very high demand for allocations/deallocations of homogenous objects which have the same type (for instance, traveler agents for each member of a population). Additionally, in a multi-threaded execution paradigm, objects used by agents may be allocated/deallocated sporadically and frequently necessitating the management of memory with minimal thread contention if high performance is desired.

The base allocator chosen was TCMalloc (Thread-CachingMalloc), this allocator is notably utilized by Google (Chewawat, 2007). TCMalloc is distinguished from traditional allocation as being specially designed for multi-threaded applications; it was selected over other alternative memory allocators due to its performance and cross-platform availability.

The allocation/deallocation routines are strongly linked with the scheduling of individual agents. Different strategies are used to allocate or free memory depending on whether the simulation is running, whether a given type has (will) execute at this point in the iteration, and which thread is performing the operation. Blocks are spinlock controlled individually with respect to deallocation and can only be allocated from by a controlling thread – this minimizes thread contention to the point where it is nearly fully parallel. Threads will return excess memory back to the global allocator at regular points during the simulation to minimize memory bloat. The user is given three options when freeing memory, corresponding to different simulation needs – immediate, agent-scheduled, or as-needed. All of these memory management techniques have the effect of significantly improving performance in a discrete-event simulation context.

3.2. Parallel discrete event engine

Central to the POLARIS framework is the POLARIS discrete event engine (DEVE) which has been optimized for multi-threaded code development in conjunction with the memory allocator. This provides an API to: create an agent of any kind (be it a person, a vehicle, or even a weather system), describe when it wants to act, what it does when it does act, and then set it on its way to perform independently.

The agents created in the DEVE organize themselves first among iterations and then within iterations called sub-iterations. Iteration in this case provides a useful analogue for a clock ticking at any rate while sub-iteration can be used to describe what happens at this particular slice in time. Similar to Repast (Collier, 2003), the iteration only serves as a way of organizing the execution schedule – iterations are not visited if no work is performed within it. Further, iterations may be repeated, allowing reverse or itinerant execution.

Agents can talk about what they want to do using “events”. These callback functions serve the purpose of allowing the agent to say when they want to be visited next, whether they want to do anything at this particular time, and what action the agent would like to take. It may be swapped in and out over the course of the simulation creating an agent which can react in a multi-faceted ways. External entities can affect the scheduled behavior of an agent through calling a reschedule function.

The DEVE organizes all of the agents so that they execute in the order requested by the developer, then marshals multiple threads to do the execution work without conflict, strides over all data, and must do so as optimally as possible for a wide variety of execution patterns. By collecting all of these tasks into one place, optimizations which are made here will ripple through all simulations which make use of the DEVE. The DEVE has the additional benefit that the threading of object execution is transparent to the user, in effect creating automated threading.
All systems in the DEVE have been profiled for multiple simulation cases using the Intel VTune software to identify and help eliminate performance bottlenecks and shortcomings, resulting in a highly optimized system for implementing agent-based models.

3.3. Additional software development framework capabilities

Along with the core Polaris framework components of memory allocation and discrete event simulation discussed previously, are a number of ancillary capabilities which are useful when developing agent-based transportation simulation models. These include a graphics library for adding visualization and interactive elements to a simulation, database input-output capabilities, and scenario management for running multiple case studies.

A critical feature of any simulation is the capability to show the results and track the progress of a running case. The Antares library is built to provide developers with the tools necessary to visualize in either 2D or 3D. The philosophy is similar to, and heavily integrated with, the discrete event engine – i.e. describe in atomic terms (agent by agent) what you would like to display and allow Antares to collect all such requests and render them. The first step to utilize Antares’ capabilities is to create a layer which represents a certain type of object in the simulation, e.g. cars, links, plotted network measures of effectiveness (MOEs), etc. Once allocated, the layer is configured to communicate the type (triangles, quads, points, lines, polygons) and format of the geometry which will be used as well as how the user would like the geometry data to be managed over time (when will it be deleted, when will it get updated). Antares also allows the developer to designate a texture (image) which can be placed on the drawn objects (for instance decaling an image of a car on top of a point). From that point forward as the simulation runs, the user may push visual elements to the layer which will then be drawn on either the 3D canvas or corresponding 2D plot.

There is a degree of interactivity which may be programmed as well. The developer may attach object pointers to visual elements and specify callbacks which will be invoked when the object is selected, double clicked, or depending on window events. In addition, the simulation writer may display attributes in a table or define a custom dialogue for the simulation user to interact with during the course of execution. Further, the user may react to the selection of an object by pushing new geometry (perhaps to highlight the object or to display something characteristic of it such as a vehicle’s path).

Internally, Antares utilizes high performance techniques and data structures to ensure the user’s requests are translated quickly and reliably into 3D or 2D. The underlying graphics library utilized is OpenGL, which is well known as an industry standard; the underlying interfacing library is WxWidgets, which also has a well established reputation; finally, the underlying plotting library is a WxWidgets binding of the PLplot library. In particular, OpenGL delivers extremely high performance rendering by making efficient use of the PC’s graphics hardware and maintaining a low level interface with the software. In addition, Antares is fully multi-threaded and has been optimized utilizing the Intel VTune profiler to ensure bottlenecks have been minimized.

An additional design future used to achieve the goals of reusability and standardization is the utilization of a standardized data interchange mechanism, although custom data sources can also be used. SQLite was selected as the default database engine as it is a lightweight database engine that requires no configuration and does not require installing and operating server-like engines on the deployment machine. C++ and Python libraries available as part of the POLARIS project that allow developers and users to read all of the data from the SQLite database into memory data structures. The IO library uses the Object-Relational-Mapping library ODB. ODB maps database tables to C++ objects and allows seamless navigation from table to table. The navigation property of each of the C++ objects are defined by referential integrity constraints defined in a database (foreign keys). Another capability provided by ODB is the query language implemented in C++. Thus a user can query the database using native C++ functions and objects without using SQL.

Finally, the Scenario Manager is a general set of tools used to develop and run various model scenarios allowing the code to be able to exploit high performance computing resources to simulate multiple scenarios concurrently and thus enables faster and more sophisticated evaluation. The major components of the scenario manager are a Scenario-generator which generates multiple scenarios based on a JavaScript Object Notation format scenario description. The scenario description defines scenarios using parameters such as weather conditions and incident types as well as the occurrence probability associated with each scenario. After generation the Scenario-executor creates simulation jobs from scenarios and submits jobs to a cluster computing system. During job execution, the Scenario-monitor monitors the status of each submitted job, including the progress percentage, estimated remaining running time, summary of network statistics, etc. Finally, the Scenario-analyzer collects the performance measures reported for each scenario, weights them by their occurrence probabilities, and sums across scenarios to provide overall measures of effectiveness. The scenario analyzer also produces plots of performance measures from the outputs of scenario execution for comparative analysis.

The scenario-executor is a Python program that executes scenarios as computing jobs in a computer cluster. The cluster automatically schedules and assigns the jobs so that they are executed on different nodes concurrently. The scenario manager supports the PBS (Portable Batch System) job management system and works with Linux cluster environments. The status of the submitted jobs is then monitored by the scenario monitor which is a Shell program that relays information from the simulation and the PBS system.

4. Polaris integrated activity-based model

A simulation model which serves as an example of a fully-integrated, agent-based simulation of both person travel and ITS operations has been developed using the POLARIS framework described above. The model consists of a series of agent
classes which implement events corresponding to typical components found in travel demand, network simulation and operations models. At the center of the model is a person-agent which represents the travelers in the system and their activity and travel planning behavior. The agent and their behaviors are implemented using a specific design pattern referred to as a computational process model. The computational process-type model is a natural fit for the interoperation of activity demand with the network simulation, as the focus is on the ‘bottom-up’ process of developing and implementing an activity-travel pattern from low-level decision models and behavior processes. In fact, in this instance, the network simulation actions become such low-level decision process, i.e. route choice becomes another decision of the agent constrained by the agent’s context, and traffic simulation is implemented as a series of agent movements driven by another behavioral model, the bounded rationality route switching model. The person agents operate in an environment represented by the transportation network agents to handle movements through the system. A set of ITS components and an automated Traffic Management Center (TMC) agent controls the ITS system and monitors the network agents. The model allows to analyze the benefits of different network operational improvements. The various components are discussed at length in the following section, followed by a section which describes the actual case studies to be carried out using the model described. Throughout these sections, it is important to note that both the simulation model itself and the Chicago-area network event case studies are currently only intended as demonstration prototypes. The following sections document each of the primary components of the simulation model and discuss a case study.

4.1. Activity-based travel demand model

The first primary set of components in the POLARIS Integrated ABM is the activity-based demand model which is implemented as a series of actions and behaviors that traveler agents engage in during the simulation process. Unique features of the demand model include dynamic activity generation, within simulation activity attribute planning and re-planning, and a detailed activity scheduling model which resolve schedule conflicts and maintains a consistent daily schedule for the agent. The demand components are also responsive to network and traffic management events, which can result in agent re-planning. The demand components implemented in the POLARIS demonstration model derive from previous work in modeling activity-planning and scheduling behaviors found in the development of the ADAPTS (Agent-based Dynamic Activity Planning and Travel Scheduling) model (Auld and Mohammadian, 2009), which is unique amongst activity-based models in that it is formulated as a dynamic model of how the activity planning and scheduling process is implemented for an individual over time. The ADAPTS model components have been reorganized to more closely fit the agent-based paradigm and have been implemented using the POLARIS framework.

The ADAPTS model is an activity-based computational process model, which simulates activity planning, scheduling and execution for agents and was originally developed to simulate the underlying activity and travel planning and scheduling processes which lead to observed activity-travel patterns (Auld and Mohammadian, 2009). The model was explicitly designed to continuously integrate with traffic simulation forming a truly dynamic model, with planning and scheduling occurring in a time-dependent manner and impacted by the results of the time-dependent traffic network, unlike TRANSIMS (Nagel et al., 1999) and other similar model systems where demand generation and network assignment are separate processes. The model is intended to be able to represent a wide range of travel demand management policies, especially policies which are expected to impact the planning process of individuals. In contrast with previous activity scheduling models, by considering activity planning and scheduling steps as discrete events within the overall activity-travel simulation, and furthermore considering each attribute decision as a separate event, a more complete picture of the dynamics of activity planning and scheduling will be developed. Therefore the ADAPTS model lends itself naturally to make use of the POLARIS discrete event engine, and with its focus on understanding agent planning and re-planning processes and its integrated design pattern it is a good fit for the case study simulation. For a comparison of the POLARIS ABM (ADAPTS) with other activity-based model systems see Henson et al. (2009) or Auld et al. (2009).

The various ADAPTS model components have been implemented as a set of agents and sub-agents which perform the planning and scheduling events. There are two primary executable agent types in the demand model which are:

- Person agent – basic unit of analysis, the traveler in the transportation system.
- Activity Planning agent – the person’s cognition of the activity planning process.

The person agent has several sub-agents which represent different cognitive and physical capabilities of the traveler including, which are performed as discrete events. The primary sub-agents are:

- Perception agent – gather and process information from world.
- Planner agent – activity generation, planning and re-planning triggers.
- Scheduler agent – creates and maintains consistent daily activity-travel schedule.
- Movement handler agent – initiates simulation of physical movements in the system.
- Routing agent – choose routes in network (discussed in network model section).

The demand model interacts with the network model components through the person agent, by way of the movement handler, which pushes the person agent onto the transportation network for simulation, and through the routing faculty,
which allows the person to choose routes for various movements during the simulation. The interactions between the various demand model agents and between the demand model agents and the network model agents are shown in Fig. 2. The figure shows the primary connections between the agents, the basic simulation process which each agent follows, and the time resolution at which the various discrete events are scheduled. The agent check periodically to see whether any new activities need to be generated or any activity-travel plans need to be executed. The remaining agent events occur at continuous time resolution. The activity planning events and the scheduling of the activities, occur throughout the simulation, as does the routing and traveling. Agent actions can generally be classified into four categories. These are Activity Generation, Planning, Scheduling, and Traveling.

4.1.1. Activity generation modeling

Activity generation in ADAPTS model is implemented as a set of joint hazard-duration models for the various activity types represented in the model. Further details regarding the development and estimation of the activity generation model can be found in Auld et al. (2011). The use of hazard modeling, with the time since the last occurrence of representing the time component in the hazard model and the new occurrence of an activity representing the failure condition, allows activity generation to be represented as a dynamic process. Additionally, the use of a joint competing hazard model allows for the representation of competition for time between different activity types as well as the natural grouping of activity types in activity chain. In other words, the generation of certain activities makes other activities either more or less likely to occur. The activity generation model implemented in the case study model is a simplified version of the original ADAPTS generation model as the case study focuses on single day simulations while ADAPTS simulated one month of planning.

4.1.2. Activity planning and re-planning

The activity planning process follows any activity generation event a person agent performs. The activity generation event, as discussed above, generates a new activity planner agent which represents both the activity to be engaged in as well as the cognitive process relating to the planning of that activity. Immediately following the generation event, the new activity planner agent schedules a planning order evaluation event. The planning order model sets what are considered in the ADAPTS framework to be intrinsic attributes of the activity such as the activity planning horizon, and the attribute plan horizons, and the attribute flexibilities. The model consists of a series of linked multivariate ordered probit regression models which use the individual and schedule characteristics to determine likely activity planning characteristics. Factors which have a tendency to increase scheduling difficulty, such as the individual being employed, having less discretionary time, and scheduling longer duration activities on average, tend to increase planning time and reduce planning flexibility. The models also have a learning component, in that the expected frequency and duration based on past experience factors into the model. More detail regarding the model can be found in Auld and Mohammadian (2012).

![Fig. 2. Demand model agent simulation framework.](image-url)
The planning order model, then, schedules the remaining activity planning events for each generated activity, including mode choice, location choice, etc. Each choice model is considered a separate discrete event, scheduled to occur at a time consistent with the planning order model results. Currently, the attribute choice models are implemented in a simplified fashion, except for the location choice and mode choice models. The destination choice model is a fully implemented planning-constrained choice model, meaning that the choice set is specified based on the state of the activity and the planned activity schedule at the time the choice is made (Auld and Mohammadian, 2011). Different constraints are applied depending on what aspects of the activity have already been planned at the time of the choice. For example, if the start time is known, the available time in the schedule surrounding the start time is used to determine the available set, while if it is not known the largest open block of time in the schedule is used. The destination choice model uses a competing destinations formulation of a standard multinomial logit model with stratified sampling, based on distance and attractor variables, from the constrained choice set. The mode choice model is a simple logit model which operates in a similar fashion, with varying constraints based on the state of the activity and schedule. For example, if the person is already out of home without an automobile the automobile mode becomes unavailable. If the location of the activity is known, the skimmed or routed travel times by mode are used in the choice utility, while if the location is not known the travel times are substituted for a weighted accessibility measure. The remaining attribute models are implemented as draws from constrained observed probability distributions.

4.1.3. Activity scheduler

Activity scheduling in the ADAPTS model implementation consists of a set of heuristic scheduling rules and a conflict resolution model (Auld et al., 2009), which has been estimated from scheduling behavioral process data collected during the CHASE survey (Doherty et al., 2004). The scheduling rules are an extension of those found in the ILUTE/TASHA model system (Roorda et al., 2005), which use the estimated conflict resolution model to inform a set of heuristic scheduling rules. The rules determine whether the conflicting or original activity involved in a conflict is moved forward or backward in time, shortened, split or deleted, depending on the details of the individual activities and surrounding schedule. The amount that the activities can be shifted or shortened is governed by the modeled start time and duration flexibilities determined in the activity planning process model. The scheduler operates whenever a new activity is added to the person agent’s activity schedule, whenever an existing activity is re-planned, and whenever deviations from the original schedule are observed during simulation of an episode. The core function of the scheduler is to ensure schedule consistency, in terms of the space and time constraints.

4.1.4. Simulation of activity and travel

The final primary demand model component is the interface with the network model components which implement the traffic simulation. This interface occurs in the demand model side primarily through the person agent’s movement handler. This event serves to place the person in a vehicle for simulation, and push the vehicle onto its departure link as specified in the movement plan developed by the person router. Once the person is loaded to the network, the network model components carry out the simulation of the actual trip specified by the movement plan. It is important to note that while the person is being moved by the network model components, all other aspects of the person agent can continue to operate, including the generation, planning, replanning and scheduling of new activity-travel plans. The only restriction here is that while the persona agent’s movement handler is operating no new movements can be started. Next, the travel simulation is discussed in further detail.

4.2. Network simulation model components

The second primary set of components in the POLARIS simulation model is implemented as a series of three network model components: (1) an individual traveler’s route choice model in response to traffic information, (2) a route generation model using simulated travel costs, and (3) a mesoscopic traffic simulation model. The network model components are implemented in the agent-based framework provided by the POLARIS core event scheduler. The network model is seamlessly integrated with the demand model by providing pre-trip and/or en-route path information for travelers taking demand side actions, and the ITS model by publishing sensor information to TMC agent for real-time traffic information provision. In turn, the demand side actions and TMC operations also impact traveler’s route choice and traffic flow pattern. In POLARIS, the network model agent includes three sub-agents: traveler agent, routing agent (which is a sub-component of the traveler agent), and traffic simulation agent. In the route choice model traveler agents make route choice decisions with respect to their own user characteristics and in response to pre-trip and/or en-route traffic information and events from Internet, VMS, GPS, and Radio. The routing event calculates the least time routes for the individual traveler using simulated travel costs. The traffic simulation agent simulates the movement of each individual traveler agent based on the Kinematic Wave theory of traffic flow. In addition, intersection operations are also simulated for signal controls such as pre-timed and actuated signals, as well as stop and yield signs. The traffic simulation model agent also captures dynamic capacity reductions due to special events such as weather and accidents.

The interactions among these network model agents as well as the demand model agent and the ITS model agent is depicted in Fig. 3. Given the inputs of an activity plan determined by the demand components, each traveler makes route choice decision using routes generated by the routing agent based on his/her route choice behavior with response to
information provision from the ITS model. These travelers with route choice decisions are then simulated by the traffic simulation agent. The output of the traffic simulation model is network performance, which will be inputs to the route choice model, demand model, and ITS model in the integrated modeling framework.

4.2.1. Route choice model

The route choice model describes traveler agents’ dynamic route choice decisions with response to pre-trip and en-route traffic information. It models both travelers with pre-trip traffic information only, and equipped travelers with both pre-trip and en-route information. Note here that the traveler agent refers to the same agent as the Person Agent discussed in Section 4.1, but that uses a separate POLARIS interface which implements the demand-network interactions in terms relatable to the network model. Both traveler types are assumed to be able to access prevailing traffic information at their origins. Only equipped travelers can access real-time traffic information during their trip through their equipped devices with navigation services using real-time traffic information. The unequipped travelers can access real-time traffic information disseminated from a TMC through ITS dissemination infrastructures such as VMS and Radio to respond to both recurrent and non-recurrent traffic congestions. In light of these traveler information assumptions, the core function of the route choice model is to realistically address the en-route switching behavior of traveler agents. A bounded rationality en-route switching model (Mahmassani and Stephan, 1988; Mahmassani and Jayakrishnan, 1991; Jayakrishnan et al., 1994) is used. This route choice modeling framework can also be extended to incorporate traveler agents’ dynamic route choice decisions with response to experienced time-dependent traffic information through solving the user equilibrium dynamic traffic assignment problem with multiple user classes (Peeta and Mahmassani, 1995).

4.2.2. Route generation model

The route generation model calculates the least time routes for travelers from their origin link to their destination link, and is implemented through a routing agent owned by the traveler. Each routing agent has its own copy of the network topology and costs, which enables the use of heterogeneous sources of traffic information such as historical, prevailing, experienced and predictive travel costs for different user classes and for both demand and network models. The routing agent provides a set of shortest path algorithms such as one-to-one shortest path algorithm and one-to-all shortest path tree algorithms for different purposes. In the route choice model for each traveler agent, different to the shortest path tree calculation approach in the route generation models in the present dynamic traffic assignment tools, we implement an A-Star based one-to-one shortest path algorithm. This implementation allows the parallelization of route calculation by each vehicle. It is also consistent with the integration with activity-based demand model to calculate routes between two activity locations instead of two traffic analysis zones, where the number of activity locations is much larger than number of zones in a regional network. In addition to calculate routes for the network model, the routing agent also calculates routes for the activity-based demand model for the choices of activity locations and schedules.
4.2.3. Traffic simulation model

Currently, the traffic simulator uses a variant of the Lighthill–Whitham–Richards (LWR) traffic flow model, which is a combination of a conservation law defined via a partial differential equation and a flow-density relation, which is called the fundamental diagram. The non-linear first-order partial differential equation describes the aggregate behavior of drivers. The model explicitly represents the dynamics of the primary variable of interest – traffic density which is a macroscopic characteristic of traffic flow and the control variable of interest in transportation system management strategies. Traffic density is defined as a number of vehicles per unit of length. The model is very well studied and was used in many transportation applications. Another benefit is that there exist an efficient numerical solution to the partial differential equation underlying the model.

To numerically solve the partial differential equation we use the Newell’s Simplified Kinematic Waves Traffic Flow model discretization scheme (Newell, 1993), which is a link-based solution method and has been recently recognized as an efficient and effective method for large-scale networks (Lu et al., 2013) and dynamic traffic assignment formulations (Zhang et al., 2013). A notable implementation of this model is in an open source dynamic traffic assignment tool – DTALite (Zhou and Taylor, 2014; DTALite, 2012). The model is used as the traffic simulation model agent in the POLARIS framework. The traffic simulation model includes a set of traffic simulation agents for intersections, links, and traffic controls as shown in Fig. 2. Given as input a set of travelers with route decisions and the traffic operation and control strategies in the network, the network simulation model agent simulates traffic operations and controls to provide capacities and driving rules on links and turn movements at intersections. With these capacity and driving rule constraints, link and intersection agents simulate the traffic flows using cumulative departures and arrivals as decision variables based on the Newell’s Simplified model, which then determines the network performance for the route choice model, the demand model, and as well as the ITS model in the integrated framework. The traffic simulation model agents also produce a set of measures of effectiveness (MOE) such as average speed, density, and flow rate, as well as individual vehicle trajectories.

4.3. Traffic management/its components

The final primary set of components in the prototype POLARIS ABM relates to the Intelligent Transportation System/Network Operations simulation. There are three components in the POLARIS ABM that were developed to work with the demand and network components including for simulating network operations. These include Network Events, ITS Infrastructure Agents and Traffic Manager Agent. Fig. 4 shows the relation of these components to the demand and network models.

![Fig. 4. Traffic management and operations model components.](image-url)
The Event Manager provides information about network events, such as accidents, weather conditions, and special events.

The Sensor Model imitates sensor readings by adding noise to the ground truth speed data calculated in the Traffic Simulation Model. The locations and sensor types are specified in the input data that is stored in ITS Infrastructure object. The job of the TMC Agent is to infer new network events (congestion, delays, etc.) and make decisions based on the inferred events, current network state as well as events provided by the Event Manager (model inputs). This aspect of the model is intended to allow planning agencies to analyze the benefits of different network operational improvements.

The TMC agent currently is implemented as an interactive agent which can be manipulated through the run-time Graphical User Interface/model visualizer. This allows a user to watch a running simulation and change the state of the various ITS components as the simulation progresses based on the observed network performance. This mode then allows for the evaluation of human-in-the-loop TMC operational strategies. It is important to note here that this interacts directly with the running simulation and can change the state of the network and the individual agents, which is a particular strength of this modeling approach. As part of the framework an automated Traffic Management Center Agent is being developed to model the TMC response to the observed network events. The goal of the automated TMC agent is to monitor the status of the transportation network (speed, travel times, etc.) as well as network-related events (weather, incidents, etc.) and decide on a response that would allow to mitigate unusual congestion level on the network. The automated responses are currently limited to pushing incident messages to VMS and HAR, but more detailed operational strategies can be implemented.

5. Case study and initial results

To evaluate the effectiveness and efficiency of the POLARIS ABM, a case study was conducted for the Chicago metropolitan area. In this case study our model was used to analyze the benefit of a simple ITS infrastructure in terms of the improved network performance. In this section we present the setup of the case study and the results that have been obtained. The case study is extracted from a larger regional model developed for the Chicago area, which includes 10 million individuals making 27 million trips. The component models of the POLARIS ABM have been estimated using Chicago travel survey data, and initial calibration has been performed for each model component against observed data. Details of this validation of individual activity-based model components can be found in Auld and Mohammadian (2013). In order to validate the performance of the system as a whole, a comparison against the network loading characteristics observed for the Chicago region over time was performed as shown in Fig. 5. This comparison is important as the network loading (or density of vehicles in the network over time) is sensitive to model misspecification, as inconsistencies in the trip lengths, activity generation, mode shares, etc. will all cause substantial differences in the network loading distributions, and the model performs well in this regard.

5.1. Case study environment

The area surrounding the Chicago CBD has been extracted from the full model in order to perform a Variable Message Sign impact assessment. The area has around 5000 links, 3000 intersections, 6500 activity locations, and 400,000 travelers. The simulated ITS infrastructure includes 20 VMS (Variable Message Sign) located along the expressways throughout the area. Three scenarios were studied, including a normal day scenario, a scenario with incidents but with the ITS infrastructure disabled, and a scenario with incidents and the ITS infrastructure enabled. The incidents in the case study include nine accident events which have been extracted from a database of historical incidents in the area from a representative day. The VMS sign locations, traffic incident location and times and the road network can be seen in Fig. 6. The impact of the accident events to the network traffic was modeled by the reduction of link capacity and free-flow speed per the rules introduced in the FHWA guidebook (FHWA, 2013). The ITS responses include notifications about accident events displayed on VMS located along the expressways that drivers observe when traversing links along which the VMS signs are located. When the drivers receive a

![Fig. 5. Number of in network vehicles as simulated by POLARIS Chicago model compared to CMAP survey.](image-url)
VMS message, the en-route switching behavioral model is triggered, which enables them to route around accident events if necessary. If no VMS messages are displayed, drivers only switch routes when directly observing the congestion from the accident on the link ahead, which limits the opportunities for choosing suitable alternative routes.

5.2. ITS evaluation and benefit estimation

Here we show the average delay on the expressways as a function of time over 5-min intervals and how it compares amongst the three scenarios. Note that the analysis is limited to expressways as the VMS signs are only located along such links so the impacts are highly localized. The baseline delay distribution is plotted along with the accidents-only and accidents-plus-VMS scenarios. The results in Fig. 7 show the impact that the network events have on the network performance, with substantially more delay experienced on the expressway links following the accidents than on an average day. The case with VMS responses shows the impact these have on the system performance, with the delay generally decreasing when vehicles are informed of events occurring. The provision of the accident information allows the vehicle to reroute prior to experiencing the congestion, thereby reducing the experience delay on the expressways, although all
vehicles are not able to reroute around the incidents. Overall, the use of ITS saves 3462 h of delay which is a reduction of 18% of the excess delay caused by the network events. It is important to note here that the current model and case study are intended solely to demonstrate the capabilities of the modeling system. Much future work is needed before the tool is capable of performing actual policy analysis.

5.3. Computational efficiency/runtime performance

The simulations were performed on standard desktop computers, with Intel i7-980X processor, which has six processor cores, and 12 GB of RAM. The run-time for the CBD case study averaged 10.5 min and used a maximum of 4 GB of RAM. The model has also been run for the full Chicago regional network, which has 30,000 links, 20,000 nodes and 10 million travelers. The full model takes approximately 1.2 h to run on a desktop machine with two eight-core processors and 64 GB of RAM.

6. Discussion

The modeling approach proposed resolves the integration issues postulated at the beginning of the paper. It relies on agent based methodology and implements all of the aspects of transportation modeling in a single stack of libraries implemented in C++. The behavior of agents in the transportation system have been implemented as an agent-based computational process model, where all aspects of travel are represented as individual agent behaviors. The specifics of the ADAPTS activity-based model and the Newell’s traffic flow model have been adapted to represent such behaviors, but the design should apply as well to a range of different model implementation strategies such as traffic flow microsimulation or alternative activity demand generation models which employ a bottom-up modeling strategy. The paradigm may be less well suited to models which employ top-down estimation of activity-travel patterns as such a modeling strategy allows less interaction with the network simulation. We demonstrated the advantages of the approach from a computational standpoint and a methodological point of view.

We would like to note that formulating the transportation system modeling problem as an agent-based simulation presents challenges to integrate with existing modeling tools. This is due largely to the frequency with which agents in POLARIS alter their behaviors – this makes it difficult to serve efficiently by existing tools, such as an external router or DTA. In addition, many tools are being developed in higher level languages such as Python or Java which present more complex integration paths with a C++ application. There are many good open source software packages developed by the community and we are in the process of adjusting our approaches so that external applications can be integrated with our existing simulation models.

Here, we have demonstrated one possible implementation of integration of demand and network models. We are still exploring possible extensions of the methodology and possibilities to accommodate different model structures. For example, instead of doing a one-shot simulation would it be possible (and beneficial) to formulate a dynamic traffic assignment the proposed framework.

7. Conclusion and future work

This paper has detailed the development and test implementation of the POLARIS framework. It covered the methodologies utilized by POLARIS which represent innovations in computation, such as optimizing memory management for the transportation simulation structure, implementing efficient discrete event scheduling, and integrated model visualization. The framework includes such aspects as covered the parallel discrete event engine, interprocess exchange engine,
The application of a simulation-oriented memory allocator, database IO library, and scenario manager, which utilize these techniques for developing an optimized environment for agent-based modeling of complex agents such as travelers.

This paper also detailed the development of the first integrated, agent-based microsimulation models based on the new POLARIS extensible agent-based framework. This framework was specifically designed to address a lack of interoperable models and integration in the transportation simulation field. The framework was used to implement the prototype integrated demand, network supply and operational characteristics model in order to demonstrate how such models can be developed and to show the utility of the POLARIS core modeling language. The prototype demonstration model is a fully functional (if somewhat preliminary) agent-based simulation model including travel demand, traffic simulation and rudimentary ITS operations, which could be extended to a tool appropriate for planning purposes with continued model development along with substantial calibration and validation effort. The use of a high performance modeling framework, optimized for detailed agent-based simulations allow planning models to incorporate network operational characteristics, enhancing the utility of such models from the planning perspective. Initial tests with case studies involving traffic management centers to various network events such as accidents and weather events show the potential of the system. The overall POLARIS framework is designed in such a way that it is straightforward for researchers to both extend and incorporate the work implemented for this prototype, improving the prototype model to potentially be useful for real-world policy analysis.

Acknowledgements

The authors would like to acknowledge the contributions of Brian Gardner and Jeremy Raw to this research. This research was supported by the Federal Highway Administration TRANSIMS Research and Deployment program under DOT Interagency Agreement No. DTFH61-11-X-30029.

References

Arentze, T., Timmermans, H., 2000. ALBATROSS – A Learning Based Transportation Oriented Simulation System. European Institute of Retailing and Services Studies (EIRASS), Technical University of Eindhoven.


