Evolving Traffic Assignments for Urban Evacuations

A Thesis

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Science

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Keith J. Drew

Major Professor: Robert B. Heckendorn, Ph.D

Committee Members: Daniel Conte de Leon, Ph.D; Ahmed Abdel-Rahim, Ph.D

Department Chair: Rick Sheldon, Ph.D

May 2017

Authorization to Submit Thesis

This thesis of Keith J. Drew, submitted for the degree of Master of Science with a Major in Computer Science and titled "Evolving Traffic Assignments for Urban Evacuations," has been reviewed in final form. Permission, as indicated by the signatures and dates below is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor:	Robert B. Heckendorn, Ph.D	Date
Committee Members:	Daniel Conte de Leon, Ph.D	Date
	Ahmed Abdel-Rahim, Ph.D	Date
Department Chair:	Rick Sheldon, Ph.D	Date

Abstract

Evacuation planning is a fundamental part of emergency management. An effective evacuation plan can mitigate significant loss of life. During a disaster many factors can complicate evacuation, such as traffic congestion, real-time damage to the transportation network, changes in the safety of areas in the city, and noncompliance with established evacuation plans. Previously, traffic in an evacuation has been managed as a routing problem from origins to destinations. This work describes an evolution-based approach to the problem of evacuation planning using a Markov model approach. It is designed to better adapt to unpredictable complications of disaster evacuation.

In our approach an Evolution Strategies algorithm is applied to sets of probabilities describing assignment of traffic to streets in networks representing urban areas. A mesoscopic traffic simulation is used to evaluate the fitness of each set of probabilities, establishing a traffic assignment distribution. Fitness is evaluated by measuring the safety of all vehicles at the end of the simulation. This method allows abstraction of individual vehicles and origin-destination pairs, allowing a more generalized solution. Sources of danger creating the need for evacuation are also abstracted, allowing the application of this model to arbitrary disaster events.

Acknowledgments

I would like to acknowledge the assistance and guidance of Dr. Robert B. Heckendorn. He has served as a mentor and shining example of a computer scientist throughout my work with him. He has been an exemplar for rigor and questioning. I am grateful for my experiences working with him.

I would also like to acknowledge the Scholarship for Service program managed by the Center for Secure and Dependable Systems at the University of Idaho. This scholarship has provided me with guidance in how to transition from the academic realm to the professional realm, and provided many opportunities for career development.

I wish to extend special thanks to Antonius Stalick for his contributions as well. Anton helped contribute to the code-base of the work presented within, and was invaluable in his willingness to discuss difficult hurdles in this work.

Finally, I would like to thank the remaining faculty and staff of the University of Idaho's Computer Science department. They have been a great resource and have provided more insight and experience than they may suspect.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Problem

Natural and man-made disasters pose serious threats to urban areas and human lives. Possible disasters range from hurricanes to terror attacks. In preparation for and response to these events it may be in the best interest of the population to evacuate the area. Accordingly, evacuation planning can help mitigate loss of life. However, traffic networks are not generally designed to handle the atypical traffic patterns that may occur during an evacuation. Roads in the network may become congested, resulting in less-than-optimal roadway throughput. This could lead to delayed evacuations with potentially negative consequences.

Evacuation faces a number of potential problems. These problems include roadway congestion, changes in traffic network topology, and moving areas of safety and danger. Real-world examples of these problems include traffic jams, roadway blockage from debris, vehicle accidents, and disasters that move over time, such as hurricanes, tsunamis, or potentially terrorist attacks. All of these factors heavily impact the availability of evacuation routes.

Creating an evacuation planning model that can address these problems is important for the safety of people living in urban areas that may need to evacuate. Current approaches to evacuation planning involve high-level origin-destination (OD) based plans that are designed for specific events [13]. While planning for disasters is important, reacting to unforeseen events during an evacuation is also critical.

The problems we wish to address in this work are related to evacuation planning and constraints. Specifically, we wish to provide a model for managing evacuations in advance of and response to arbitrary disaster events, subject to real-world constraints like roadway capacity, traffic network topology, and changing areas of safety and danger.

Question

In this work we investigate the effectiveness of evolution in providing traffic assignments for vehicles in urban areas subject to evacuation. Traffic assignments are directions given to traffic moving through the evacuation area. Our initial question is this: *can evolution effectively optimize traffic assignment as a function of safety, under significant constraints, in real-time?* Other questions arise as we describe our approach.

The constraints we wish to address in this work include changes in safety, traffic network topology, and vehicle distribution. Therefore, the question becomes: *can evolution effectively optimize traffic assignment as a function of safety during an evacuation, while responding to changes in safety, topology, and vehicle distribution, in real-time?.*

Planning for evacuations ahead of time is important for both emergency management personnel preparation and public knowledge. The need to plan in advance of an evacuation modifies our research question, producing our final query:

Can evolution effectively optimize traffic assignment as a function of safety in advance of and during an evacuation, while responding to changes in safety, topology, and vehicle distribution?

We hypothesize that evolution can optimize traffic assignments for safety. Further, we hypothesize that evolution can provide robust solutions which are capable of adapting to changes in an evacuation environment. Finally, we hypothesize that if an evolutionary approach can route traffic to safe areas, evacuation plans can be formulated from *safety functions*. A safety function in this context describes a mapping of safe or dangerous areas across the area being evacuated. Such safety functions should be able to capitalize on the characteristics of specific disasters. For example, elevation may be used as a measure of safety when creating a plan for tsunami or flood responses.

Approach

We propose an evolutionary approach to the Dynamic Traffic Assignment (DTA) problem. We argue that evolution provides the desirable characteristics of being adaptable, robust, and the ability to carry forward important information from one population of solutions to the next. Our proposed approach is independent of any specific traffic system or disaster event, thereby allowing adaptation to any evacuation scenario.

The Evolutionary Algorithm (EA) we chose is an Evolution Strategies (ES) algorithm. We evolve a cloud of probabilities that map to possible routing choices at each intersection in the evacuation zone. To evaluate the effectiveness of each set of probabilities, we simulate traffic moving through the evacuation zone for a time, t, after which we measure the overall safety of the vehicle population. Traffic in the simulation is routed based on the cloud of probabilities. This method of evaluation is our fitness function.

To validate our model we use a number of small problems designed to test if certain parameters affect evolution as predicted. These problems include simple routing tasks, tests to check if the model splits groups based on capacity of safe areas, and a number of tests designed to evaluate the model's adaptability. These tests show that our model is effective for the problems described.

Long-term Goals

The work described here is preliminary and aims to show that traffic assignments for large evacuation scenarios in urban areas can be evolved. Another aim is to show that the model can adapt in response to a dynamic environment. The model described here simply produces traffic assignments; it does not describe how the traffic assignments should be communicated to drivers, nor how vehicle location information should be communicated to the model.

The long-term goal of this work is to provide a model for evacuation optimization,

based on vehicle safety, for arbitrary evacuation events. The implementation should provide a clean, usable interface, mechanisms for communicating traffic assignment information to vehicles, and integration of mechanisms that allow tracking disaster events, all in real-time.

Overview of Thesis

This thesis demonstrates the evolution of clouds of probabilities used for traffic assignment in urban evacuation events. It is also demonstrated that this approach can quickly adapt to changes in traffic networks and disaster characteristics.

The contributions described in this thesis form an optimization algorithm that finds optimal traffic assignments for evacuation areas, in response to arbitrary threats. The contributions consist of: 1) a new evacuation traffic assignment model, 2) a traffic simulation, and 3) an ES representation that leverages the simulation to optimize traffic assignment. These contributions are tied together in the following chapters.

Chapter 2 discusses relevant literature and problem specific concepts. Chapter 3 discusses our approach, design, and implementation. Chapter 4 presents a paper on our model accepted to The 2017 Genetic and Evolutionary Computation Conference (GECCO) in Berlin, Germany. Chapter 5 summarizes additional work and experiments that test our model and explore new ideas. Finally, Chapter 6 concludes the thesis and discusses areas for future work.

Chapters 3, 4, and 5 describe specific contributions. These consist of: 1) a new approach to evacuation planning that focuses on evolving sets of probabilities used to assign traffic at intersections in an evacuation zone, 2) an ES algorithm representation that allows those probabilities to be optimized, and 3) a priority-queue-based traffic simulation used to evaluate sets of static probabilities during optimization.

Chapter 2: Background

Problem Characteristics

During an evacuation event transportation networks can become congested. This congestion results in portions of the evacuating population being unable to evacuate or reach safety. Consequently, it should be the goal of any evacuation plan to mitigate traffic congestion. Other issues may also arise during an evacuation, such as changes in traffic network topology, safe zones, and non-compliance with traffic assignments by evacuees. Yuan and Han [22] suggest that multi-objective optimization is necessary for evacuation planning due to similar factors. Their work indicates that minimization of evacuation time alone is insufficient because the fastest routes may not be the safest. In their work they optimize network clearance time and space-based risk, seeking to mitigate routing through dangerous areas for the sake of network clearance time reduction.

Previous work has focused on optimizing traffic assignment and routing for origin-destination (OD) pairs [1], [4], [12], seeking to get people from one area to another by assigning routes to each evacuee. This research focuses instead on routing the population of evacuees to safe areas, defined by a safety function. The safety function represents both the safe areas and the threat(s) driving the evacuation. This implies the model is independent from the threat, provided a reasonable safety function can be applied. For example, in the event of a tsunami, it may be reasonable to define safety by elevation, but during a hurricane or terrorist attack, safety might be defined by distance from the dangerous area. This flexibility extends to the solution space of any specific problem as well. Evolution has the potential to find trade-offs and assignments that might be missed by experts who may tend to focus on specific destinations or routes.

The approach taken in this research models evacuation as a dynamic problem requiring continuous re-optimization in response to changes in the evacuation environment. Because

of this need to respond to an ever-changing environment, an evolutionary approach seems fitting for evolution's ability to adapt. Evolution can carry forward important information from one population to the next. Some information carried forward doesn't need to change in response to changes in the environment, allowing the algorithm to focus on optimizing areas where the environment has changed.

Evacuation Problems

There are a number of problems that arise during an evacuation. Issues we are concerned with in this work include congestion, topology changes, roadway capacity, safety, and vehicle distributions. Managing problems such as these is referred to as Dynamic Traffic Assignment (DTA) [17]

Our model must handle several significant problems that arise during evacuations: congestion, changes in topology, capacity constraints, moving safety/danger, and changing vehicle distributions. These issues are described here.

Congestion is perhaps the most significant problem our work seeks to address and solve. In an evacuation commonly used roadways and those with high capacities, such as freeways and main thoroughfares may become congested enough to reduce throughput to unsafe levels. Ideally, our model should compensate for congestion by assigning traffic in a fashion that maximizes throughput of the traffic system undergoing evacuation.

Changes in topology during an evacuation are another significant issue that our work seeks to address. Changes in topology imply that some portion of the traffic network becomes unusable, which is modeled as the removal of a portion of the network. Such changes can occur as a result of many different real-world events. An extreme case might be the collapse of a bridge. Another case may be roadway blockage due to debris, or perhaps flooding renders some number of roadways unusable.

Capacity is closely related to congestion and indicates the maximum number of vehicles a portion in the traffic network can handle per unit time. For example, a road segment

may have a maximum throughput of 2000 vehicles per hour. Capacity is a significant constraint in this work and can change as a result of a number of real-world factors. For example, a traffic accident can cause a reduction in roadway capacity, as the incapacitated vehicles may block a lane or multiple lanes. The roadway is still usable, but accommodates less vehicles per unit time.

Of course safety is a very important part of our model, considering traffic assignment is optimized based on a safety function. Changing safe areas over time allows modeling of moving disasters, such as hurricanes, tsunamis, floods, poisonous chemical clouds or spills, and more. Our model must be able to change traffic assignments over time, accurately and quickly, in order to manage evacuation in the case of such moving disasters.

Changes in vehicle distribution represent a very real issue that our model has to manage. We must assume that individuals will not follow directions in some or many cases. It is reasonable to assume that someone familiar with the area under evacuation may rely on their experience over the assignments being provided at any given time. In such a case our model cannot depend on predictions of vehicle distributions based on our optimizations, but must be updated regularly with information about the current vehicle distribution.

These constraints, combined with our evolutionary approach, indicate that the model must be continuously re-optimized. The need to re-optimize further suggests an evolutionary approach is fitting, provided our model is represented so that significant useful information can be carried from one optimization to the next. Provided optimizations can be carried out quickly in response to environment changes, this also suggests a real-time model, where re-optimization begins when significant changes in the network are indicated by real-world observations.

Traffic Simulation

There exist a significant number of traffic simulation packages. A non-exhaustive list of some more commonly used simulators can be found in [10]. Most existing traffic simulation software seeks to present a view of the traffic as time progresses, providing a visual understanding of how traffic flows through certain network topologies. However, our model needs to simulate traffic internally, without a visual representation, for the sake of wall clock time speedups.

Passos, Rossetti, and Kokkinogenis describe a more complete picture of traffic simulation software and taxonomy in [16]. In their paper, the authors break simulation models into four groups: microscopic, macroscopic, mesoscopic, and nanoscopic. Macroscopic simulations model traffic at a high level based on network flow models, and do not treat vehicles individually, losing what may be key information, depending on the goals of the model. On the opposite end of the spectrum are nanoscopic simulations, which include a high level of detail about each individual vehicle, including sensors and vision parts.

Of particular concern for our work are microscopic and mesoscopic simulations. We compare our work to some work done using VISSIM [7], a microscopic simulation tool. Our simulation, however, falls into the mesoscopic category. Mesoscopic models fall into a granularity between macroscopic and microscopic. They model individual vehicles, but also provide vehicle grouping, which our model uses for speedups. Femke van Wageningen-Kessels et al. in [21], state: "Mesoscopic models describe vehicle behavior in aggregate terms such as in probability distributions. However, behavioral rules are defined for individual vehicles." In our model we use probability distributions to control traffic routing, as opposed to describing traffic that is routed by other means.

Timing is a key consideration in traffic simulation. Some models, such as VISSIM and SUMO [7], [11], use time-steps where each time step requires calculation of the next simulation state. Managing these time steps leads to another taxonomy, fine vs. coarse-

grained models. Our simulation does not fit neatly into either side of this taxonomy, as time is managed per event. Our simulation is not concerned with traffic behaviors which require discrete time step evaluation. However, it is required that our simulation addresses events which happen across the whole range of simulation time, with high precision. Therefore, our model is sometimes finely grained, when there is a lot of activity in the traffic network, yet coarse-grained when events are spread over the total simulation time.

Evolution Strategies

Evolutionary computation refers to a group of algorithms which draw on classical Darwinian evolution for inspiration. In general, a population of individuals representing possible solutions (genotypes) are maintained and evaluated using a fitness function. While a number of Evolutionary Algorithms (EA) exist, an Evolution Strategies (ES) approach was chosen for its compatibility with our representation.

In general EAs require a representation, variation operators, parent selection operators, and a survivor selection operator. The representation refers to how individual solutions are represented. Variation operators alter individuals in a unary (mutation) or n-ary (recombination) manner. The parent selection operator is used to select which individuals should produce offspring. Finally, survivor selection is the method by which the whole set of children and parents compete for survival [6]. Generally, survivor selection is a function of each individual's fitness. The fitness of an individual is determined by a *fitness function*, typically a way of measuring the effectiveness of a solution. The fitness function for any EA or ES is problem specific.

Evolution strategies are a group of evolutionary computation algorithms that represent genotypes as a vector of real numbers [6]. Mutation is performed by adding a random number pulled from a Gaussian distribution, with a mean of zero, to alleles in the genotype. Recombination is performed using *discrete recombination*, where a child's alleles are

chosen from the parents at random. There are many parent selection operators, but in the work here we use uniform random selection for a tournament in which genotypes compete against each other.

The variable parameters for evolution strategies represent algorithm characteristics, as opposed to representation details. The significant variables include mutation step size, mutation probability, recombination probability, parent and child population sizes, and maximum number of generations.

The mutation step size is generally indicated by σ, and refers to the standard deviation from the mean given to random values pulled from the Gaussian distribution with a mean value of zero. In ES algorithms this value affects how much real number values are modified in each mutation operation.

Mutation is performed on each individual selected to produce an offspring. The mutation probability for an ES is used to select which allele in the genotype is mutated. The mutation occurs when a random value from a Gaussian distribution, with mean 0 and standard deviation σ, is added to the selected allele. The recombination probability is used to determine if recombination will be used to generate a child from two parents.

The parent and child population sizes are significant for an ES algorithm and indicated by μ and λ respectively. It is suggested in [6] that a parent-child population ratio of $1:7$ or $1:4$ be used, but is not strictly required.

The maximum number of evaluations is used to end the ES algorithm. This is not a strictly required value by any EA, and there exist a number of other methods for terminating an EA. Alternatives include schemes that recognize convergence, reaching a fitness threshold, population diversity reaching a lower bound, and more.

None of the parameters used for ES implementations have a clearly known way of determining the most ideal value. The most effective value depends on the problem and implementation, requiring experimentation to discover what works best.

SLang - The Simulation Language

A mechanism was needed for specifying network topologies and evolution parameters. SLang was used for this purpose. Created by Dr. Robert B. Heckendorn and Damien Ball, SLang is a grammar that includes commands which define a traffic network topology. SLang also provides a standard that can be used to share experiments for the sake of generating meaningful comparisons between models.

The original SLang specification was modified for this work. The previous version was designed for use with Ant-Colony Optimizations techniques, not ES approaches. Necessary adjustments were made to accommodate the new optimization algorithm. The significant commands used for this work specify the desired ES parameters, network topology, vehicle starting locations, and the safety function.

Chapter 3: Approach

Our approach for optimizing the safety of vehicles in a traffic network is to evolve sets of static probabilities that can be replaced and re-optimized quickly. Our goal is to optimize the probabilities representing the chance a driver is directed to take a certain route leaving an intersection in the traffic network, for all intersections in the network. These probabilities should be optimized in such a way that traffic is directed towards safety. For large amounts of traffic in the system, these probabilities begin to represent traffic distributions at each intersection. Distributing vehicles at each intersection this way allows our model to interchange vehicles, which can be powerful and adaptive. Route-based approaches are often constrained by the routes they assign and work under an assumption that all vehicles follow the specified route. Our model can re-assign any given vehicle to make up for non-compliance with assignments.

An Evolution Strategies (ES) algorithm was chosen to evolve probabilities. Our initial work uses a simple ES approach, not containing measures for ensuring diversity, or multi-objective optimization. Our goal was to first verify an ES would be effective. We want to establish a basis for future research and refinement of this work.

Encoding the Problem

Effectively encoding the problem is fundamental to any evolution-based model. Without an appropriate design, operators can be ineffective at producing individuals that search the problem's solution space. This section seeks to describe the encoding we used for our model, and justify it.

We assign traffic using probabilities throughout a graph that represents the network being evacuated. Nodes correspond to intersections, while edges correspond to streets. See Figure 3.1. A vehicle makes a choice at each intersection based on the probabilities for each possible exit from the intersection. We evaluate sets of probabilities by running

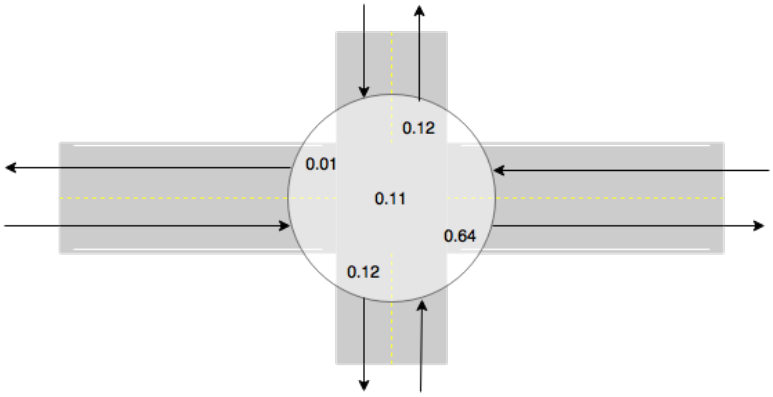


Figure 3.1: A figure showing how our model maps to the real world.

a traffic simulation where vehicles are routed based on the probabilities. In this manner we suggest that traffic distributions should correspond to the evolved probabilities. Evacuating vehicles can then be routed to safety based on the optimized distributions. For these reasons we represent the genome as a set of probabilities. Each node has a subset of these probabilities that sum to 1.0. Loading the genome into a graph produces probability-based choices for vehicles moving through the simulated network, distributing traffic accordingly.

Mutation is relatively simple. Upon selection from the parent population a genome is mutated by iterating over each node's probability subset and choosing whether or not to mutate the given node using a mutation probability. When selected, a node is mutated by adding a normalized, random value, with standard deviation of σ, to one of the edge's probability values. The values are then re-normalized to sum to 1.0.

While typically not used in evolution strategies, we tested crossover to see if it reduced optimization time. Crossover was carried out by iterating over the set of nodes in each of two parents and randomly selecting which parent would supply that node for the child (see Figure 3.2). Nodes, in this sense, indicate the subset of probabilities that assign traffic at the respective intersection in the network. Crossover was performed probabilistically, using a value defined in the set of ES parameters, $P_{crossover}$, the probability of crossover.

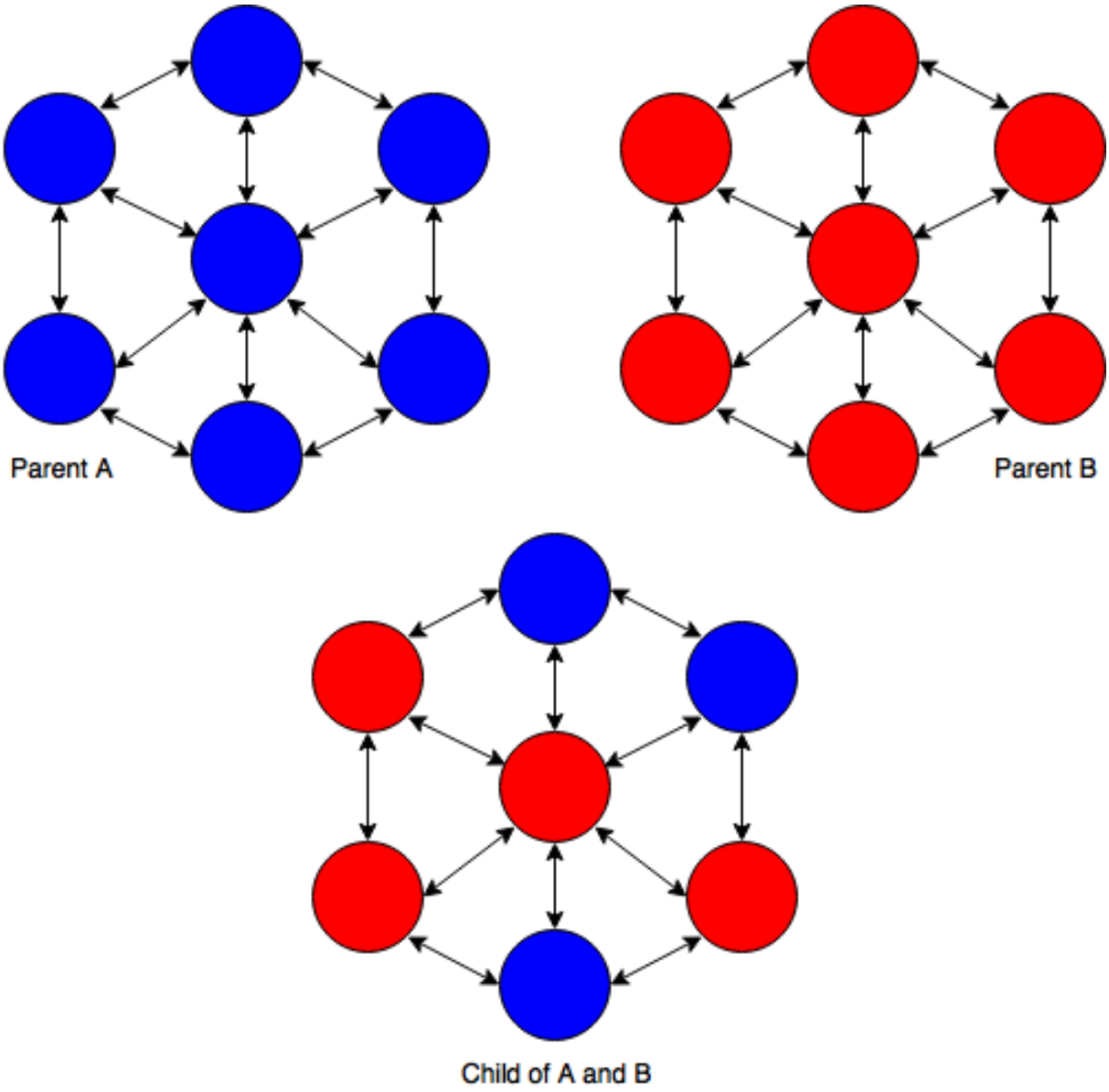


Figure 3.2: An example of how crossover works in our model.

Selecting an individual from the parent population was performed using uniform random tournament selection. Selecting individuals tournament style helps prevent premature convergence on local optima by promoting diversity. It also helps prevent "good" portions of the genome from being eliminated due to "bad" portions. Tournament selection works

by selecting n individuals from the population randomly and having them compete against each other. The most fit individual in the tournament is selected to become a parent. In this way, the worst $(n-1)$ individuals will never be selected (assuming no replacement). Also, this allows a chance for the best individuals to not be selected as well, helping to mitigate early dominance of the population by one individual (local optima).

The fitness function we use is relatively simple. The function is a sum of individual safety for all vehicles in the network, at time t. Inputs to the function include the set of all evacuating vehicles, E, and the time t. S provides the safety of a location, L. L is a function of an agent and a time parameter.

$$fitness(E,t) = \frac{1}{||E||} \sum_{a \in E} S(L(a,t)) \tag{3.1}$$

Equation 3.1 is simple, but sufficient for proof-of-concept. As stated in [22], evacuation should be treated as a multi-objective optimization problem.

Model Design

We opted to use an object-oriented design for our model. This corresponds nicely to a number of real-world objects we needed to represent in our model, such as vehicles, intersections, and roads. Also, OO design lends itself nicely to our evolution strategy, as genotypes and the population of genotypes should be self-contained with a number of operators for each data structure.

When specifying a problem in SLang, a keyword, *city* is used to create a city object. The city is defined as a graph with nodes and edges that represent intersections and roads between them, respectively. The city also contains a set of agents, which represent vehicles that move through the traffic network during a simulation. The city is designed to represent the traffic network and to run simulations which operate as a function of the sets of probabilities evolved. Figure 3.3 shows this relationship.

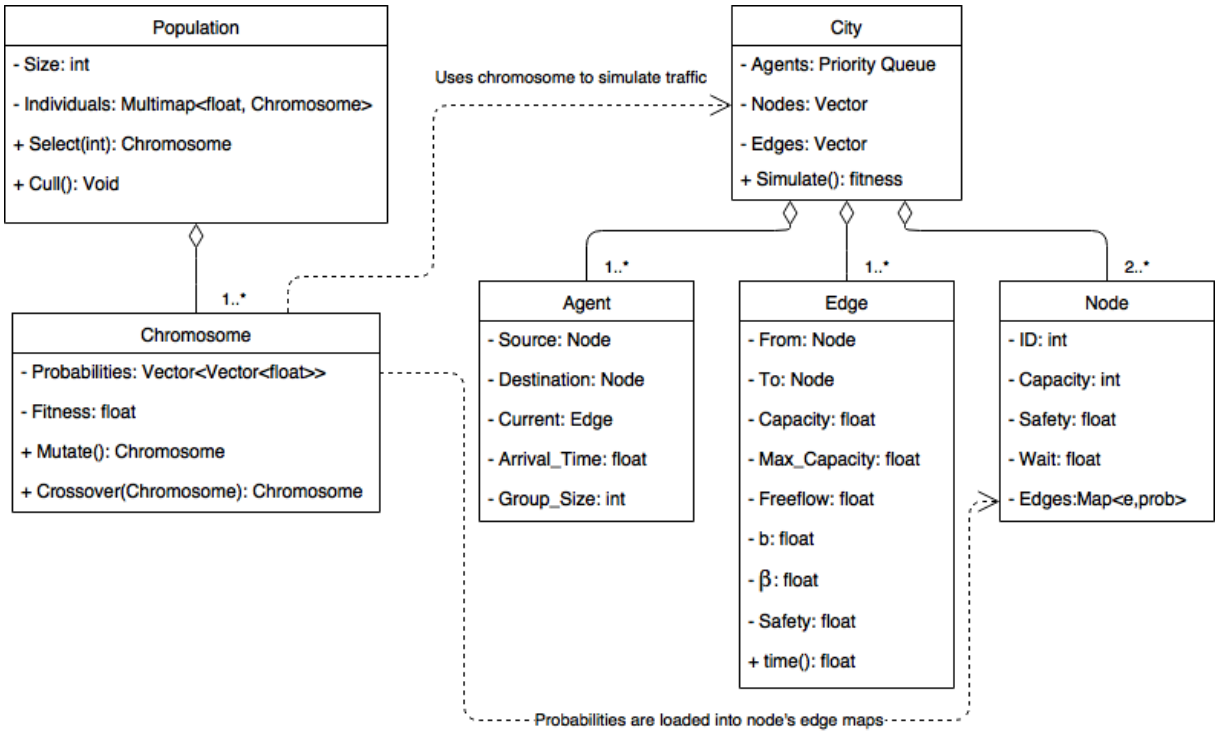


Figure 3.3: A UML diagram highlighting important relationships.

The two dominant algorithms at work in our model are the Evolution Strategies (ES) algorithm and the traffic simulation. Traffic behavior during a simulation is dependent on a single member of the ES population, a chromosome. The simulation loads the probabilities from a chromosome object into the nodes of the graph, giving probabilistic traffic assignments for agents moving through the network.

While the population of chromosomes and the city's simulation runs are heavily associated, it is important to make clear that the ES and simulation algorithms are separate. Our ES algorithm operates on the chromosomes within the population of individuals. The ES algorithm uses the city object to evaluate the fitness of each chromosome. Therefore, if the ES child population size parameter is set to 100, every generation of the ES algorithm will use the city to simulate traffic 100 times.

Model Implementation

Our implementation of the specified design was carried out using Flex and Bison to parse SLang specifications. The ES and simulation portions were coded in C++11, chosen for familiarity, speed, and access to the Standard Template Library (STL). One fundamental data structure for our model is provided by the STL: the *priority queue*. A priority queue operates like a normal queue, except all elements in the queue are sorted.

Nodes, edges, and agents are all specified as C++ classes, the significant members and methods of each class are shown in Figure 3.3. More methods and fields were required for full implementation (accessors, mutators, etc.), but are omitted here for the sake of brevity. Nodes, edges, and agents all come together to build a city object. The city object maps a single chromosome's probabilities to its set of nodes, and runs a simulation. Running the simulation produces a fitness value using Equation 3.1. The population object maintains a set of chromosomes operated on by the ES.

Nodes

Nodes represent intersections or areas where route decisions can be made by vehicles traversing the network. Of particular importance are the edge-probability map, node ID, node capacity, node safety, and node wait time.

A node object contains a map of edge pointers and corresponding probabilities. When an agent is evaluating which way to leave a node, this map object is consulted and an edge is selected based on the corresponding probabilities. The number of edge pointer-probability pairs in the map object is n, where n is the number of edges leading out of the node, or, $n + 1$, where the node contains a self-referential edge, allowing agents to stay in a given node. Self-referential edges are used for ease of implementation, and to allow all agents to remain in the priority queue during simulation, instead of moving them around and having to manage bookkeeping for a more data structures than necessary.

Generally, node IDs are used for specifying which nodes are connected by an edge. The node capacity is a hard limit on the number of agents that are allowed to be in the node at any one time. The safety of a node is dictated by the safety function being optimized for the traffic network. A node's wait time is the length of time added to an agent's arrival time when the node's self-edge is selected by an agent. This wait time is necessary for the simulation, but hard to model based on reality. Typically, a time equivalent to one minute has been used. If the wait time is zero an infinite loop can be created, and if it is too high, agents run the risk of never being able to reach their destinations.

Edges

Edges represent roads or links between intersections and carry important information that dictates how long an agent must spend traversing an edge. The particularly important features of an edge are the *from* and *to* nodes, the parameters for calculating edge traversal time, and the method that performs the calculation.

Edges are uni-directional in our model. Therefore, they more closely correspond to lanes, or sets of lanes in the same direction. This choice allows us to represent one-way roads and disparities in directional throughputs due to inconsistencies in the number of lanes for opposite directions on roads. Each edge holds a pointer to a node object that the edge's traffic comes from, and another for the node traffic is going to.

Parameters for the traversal time method are mostly static characteristics of an edge, except the *capacity* value. Capacity represents the number of vehicles currently on the edge, and directly affects the total traversal time for any new vehicles or agents moving into the edge. Each edge also has a hard limit for the number of vehicles that it can hold: maximum capacity.

Freeflow time, f, for an edge is the amount of time that it takes to traverse the edge in ideal conditions. Ideal conditions usually implies the road is empty, in accordance with Equation 3.2. This is a more generalizable parameter than some alternatives, as it captures

distance and speed limit implicitly, by way of the classic formula $time = \frac{distance}{speed}$. c and c_{max} represent the current and maximum capacity of the edge, respectively. The parameters b and β are tuning variables used to fit modeled data to recorded data. Observed travel times for actual roadways can be matched to this formula by adjusting those values. However, default values are used for cases where such data is not readily available. Those values are 4 and 0.15, respectively.

The *time()* function computes the traversal time of the edge, using the values described above and the Bureau of Public Roads formula [9]:

$$t = f + b\left(\frac{c}{c_{max}}\right)^{\beta} \tag{3.2}$$

For a node's self-edge Equation 3.2 is not used. The wait time value specified in the node object is used instead.

Agents

Agent objects are used to represent vehicles moving through the network. They hold information about where they are coming from (source), going to (destination), and current location (edge). They also hold the time at which they are set to arrive at their next destination and a value called group size, indicating how many vehicles are moving together.

Source and Destination are pointers to the respective nodes. Similarly, the current edge is a pointer to the edge the agent is currently on. These pointers provide each agent with a clear picture of where it is at, at any given time. Agents never exist on a node, except semantically. Agents always remain on edges, and nodes have a self-edge in our representation, to represent being at a node.

The *arrival time* value each agent holds represents the time it will reach the node is it traveling towards. Semantically, this value represents the next time each agent will make

a decision related to routing. The arrival time of each agent is used to sort the city's priority queue.

Finally, the group size of an agent represents how many vehicles are in a group. For sufficiently large groups of vehicles, they are grouped together to improve run time. It is important to not use too large of a group size, as it runs the risk of losing accuracy.

City

The city object maintains an agent priority queue used during a simulation. The city also maintains a set of nodes and a set of edges defined in the SLang input file, which represent a traffic network topology. Finally, the city holds a Simulate function for evaluating the fitness of chromosomes.

The sets of nodes and edges are generally static, but can be updated with changes in topology, between simulations.

The agent priority queue is the data structure used to manage simulation of traffic. For some period of time specified in SLang, the simulation method simulates traffic moving throughout the city's topology. Agents representing that traffic are stored in the priority queue and sorted by their arrival times.

In general, the simulation algorithm works by dequeuing the agent with the lowest arrival time (which is at the front of the priority queue, by definition). That agent is understood to be arriving at its destination node, from where a computation is performed to select the next edge the agent will travel along. The selection is a function of the portion of the chromosome that corresponds to the node our agent has just arrived at. Next, the agent's new arrival time is computed with a call to the *time()* method of the edge selected and adding the value returned to the agent's previous arrival time. Finally, the agent's source, destination, and current edge pointers are updated, and the agent is placed back into the priority queue. By definition, the priority queue places the agent in a location corresponding to the new arrival time.

It is important to understand how the simulation is terminated and that variations occur in general flow of the simulation. First, a total simulation time value is supplied in the SLang specification. When an agent is dequeued a check is performed to see if the arrival time of that agent exceeds the total simulation time. If so, the simulation is ended and fitness is calculated using Equation 3.1. Additionally, there are special cases where agents will select an edge that is at maximum capacity. In such a case, the agent has the corresponding node's wait time added to its arrival time, and is placed back into the priority queue. This effectively forces the agent to wait in place, as if at a traffic light.

Chromosome

The chromosome objects contain the probabilities that are being optimized, the fitness value of their set of probabilities, and mutation and crossover functions. The chromosome objects are accessed by the city object and the evolution strategies algorithm.

The chromosome objects contains its set of probabilities in a vector-of-vectors data structure. Each sub-vector contains a set of floats that are ordered and map directly to the edges stored in a node's edge-probability map. Each sub-vector's float values are also normalized to sum to 1.0.

The fitness value stored in the chromosome is returned from the city when a simulation is performed using the chromosome's probability set. It is also used by the population object to sort and select chromosomes, as part of the ES algorithm.

The mutate function is of crucial importance to the ES algorithm. Mutation is performed on every child chromosome, in every generation, and is the method by which the solution space of the evacuation traffic assignment problem is searched. The mutation operator is designed to do uniform random selection across each "node" or sub-vector of probabilities. For each selection, the function adds a random value, pulled from a Gaussian distribution with a standard deviation of σ, and a mean of 0, to one of the probability values. The sub-vector's values are then re-normalized to one.

The crossover function is less important than the mutation function, but helps the population more effectively search the solution space. The crossover function is performed by swapping sub-vectors at the same index from two different chromsomes, using random uniform selection of those sub-vectors. Semantically, this means that at a given intersection in the traffic network, one chromosome suggests the majority of traffic should leave the intersection in one direction, while the other chromosome might suggest a different direction. Because the routing is not independent of other nodes, this can help explore new possible solutions, where mutation might not be able to explore as quickly. However, there is also a chance of creating very poor solutions.

Population

The population object maintains the population of chromosomes, for use by the ES algorithm. The object maintains its size, a multimap object of individuals, keyed by their fitness values, a selection operator, and a cull operator.

The size value maintained by the population object is determined by the ES parameters, and corresponds to λ. The multimap object is used to allow easy sorting function operations on the population. The map is sorted using C++ STL functions, by fitness values. Sorting is only done prior to a cull operation. This allows fast culling, as opposed to searching for the individuals that need to be removed.

The selection operator works by tournament selection. The operator takes an integer indicating the size of the tournament, and performs the selection. The tournament selection is performed by selecting n random individuals from the population and choosing the individual from that group that has the highest fitness.

The cull operator is straightforward - it removes the least fit individuals from the population. During the ES algorithm, as children are created, they are stored in a temporary data structure, and finally added to the population. Upon insertion into the population, they are sorted. Cull works by removing the least fit individual until the number of indi-

viduals in the multimap is equal to λ.

Summary of Approach

We approach the evacuation optimization problem with an ES algorithm. Our ES algorithm is designed to operate on sets of real numbers corresponding to probabilistic traffic assignments in a traffic network. Each set of probabilities is evaluated using a mesoscopic traffic simulation designed for this problem.

Each generation of the ES algorithm selects sets of probabilities from the population and loads them into the city object, where they will serve as traffic assignments for each intersection in the city. Then, using a set of predefined vehicles in the city, a simulation runs, routing traffic according to those probabilities. At the end of the simulation, the safety of each vehicle in the city is evaluated, and a fitness score is assigned using 3.1. This sequence is performed for each set of probabilities that need to be evaluated, which are the children, or new individuals mutated from the previous generation. Figure 3.4 and 3.5 illustrate the ES and simulation algorithms, respectively. The "FITNESS EVALUATION" step in Figure 3.4 uses the simulation to evaluate fitness. It is important to note that the algorithm depicted in Figure 3.4 is the classic ES algorithm, with the highlighted portion indicating a contribution.

Each simulation continues until the time traveled by any vehicle in the network exceeds that of the total simulation time specified in SLang. The ES algorithm continues until either a maximum fitness value is achieved (meaning a "good" solution has been found) or until the maximum number of generations allowed is reached.

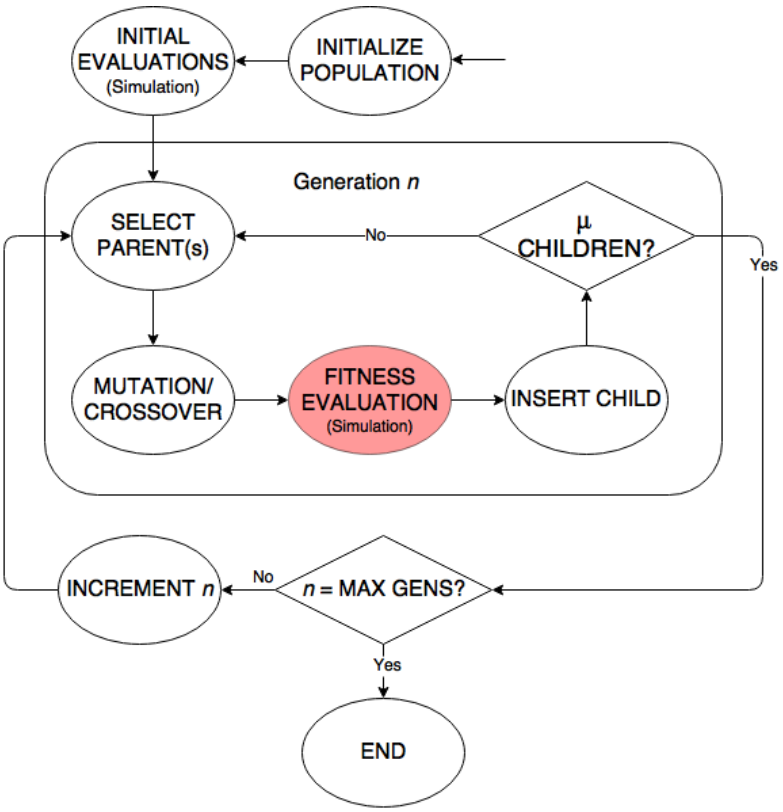


Figure 3.4: A UML diagram describing the Evolution Strategy used.

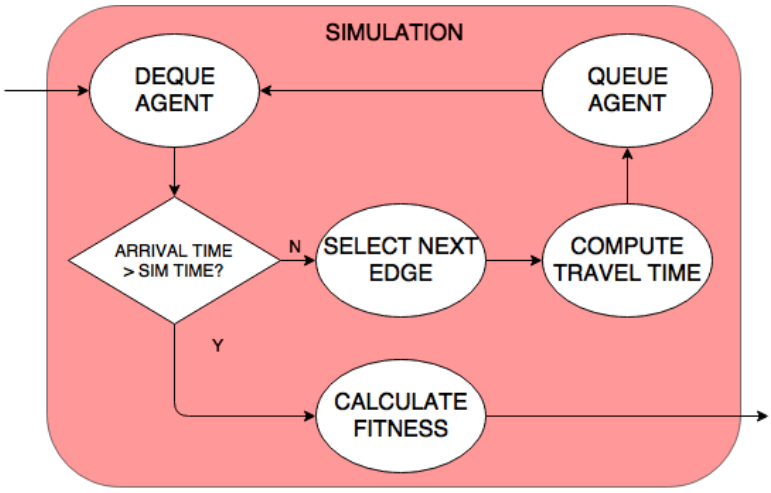


Figure 3.5: A UML diagram describing the traffic simulation used.

Chapter 4: Evolving a Real-time Evacuation for Urban Disaster Management

The paper presented in this chapter summarizes the approach described in the previous chapters. It continues to describe a number of experiments run using the model presented here, which are not described anywhere else.

The paper demonstrates preliminary work in this body of research. Our model needs to be improved upon, and the work in the paper serves to demonstrate that the model works for difficult and large problems. The research described in the paper demonstrates that our model is a method for adapting to changes in dynamic evacuation environments quickly.

This paper was accepted to The Genetic and Evolutionary Computation Conference (GECCO) 2017, under the Real-world Applications track, and nominated for Best Paper. The paper is presented, in its entirety, below.

Forthcoming in *The Genetic and Evolutionary Computation Conference, 2017*

Keith J. Drew
University of Idaho
895 Perimeter Drive
Moscow, Idaho 83844
keithd@vandals.uidaho.edu

Robert B. Heckendorn
University of Idaho
895 Perimeter Drive
Moscow, Idaho 83844
heckendo@uidaho.edu

Ahmed Abdel-Rahim
University of Idaho
895 Perimeter Drive
Moscow, Idaho 83844
ahmed@uidaho.edu

Homaja Pydi Kumar Marisetty
University of Idaho
895 Perimeter Drive
Moscow, Idaho 83844
mari7627@vandals.uidaho.edu

Anton Stalick
University of Idaho
895 Perimeter Drive
Moscow, Idaho 83844
stal6565@vandals.uidaho.edu

ABSTRACT

In an urban disaster it is important to efficiently evacuate people to safety. We use evolution strategies and a probability model to route the population by optimizing their safety. The algorithm is designed to use the strengths of evolutionary computing to repeatedly optimize an evacuation under the dynamics of a disaster such as accidents blocking critical roadways, bridge collapses, debris closures, changes in safety, and people not following evacuation directions. Our model is unconcerned with specific evacuation routes but rather evolves a robust cloud of probabilities to represent best directions of escape. We show that maintaining a population of diverse solutions may allow for rapid adaptation as a disaster unfolds. The core optimization algorithm is tested using challenging test cases as well as real-world data.

Keywords

Evacuation Planning; Evolution Strategies; Traffic Congestion, Safety Optimization, Traffic Assignment

Introduction

Natural and man-made disasters can rapidly render portions of large urban areas unsafe for people. To save lives, emergency managers may order time sensitive evacuation from unsafe areas. This can lead to overloading of transportation networks resulting in the failure to evacuate all of the population from the unsafe areas and large loss of life. Even if an evacuation is planned for long before it is needed, numerous unforeseen events may occur during the evacuation such as accidents blocking critical roadways, bridge collapses, debris closures, and flooding can cause evacuation failure and loss of life. Changes in safe zones can occur as well such as failure to predict the level of storm surge in a hurricane or wind changes in a poison gas leak. Finally, the human factor of people simply not following evacuation instructions may also lead to the need to repeatedly re-optimize to avoid congestion resulting in loss of life.

Unlike some previous work, we see the evacuation problem not as getting individuals to their predefined safe destinations, nor as distributing the people across a small set of safe destinations but rather as getting people to regions of safety indicated by the level of safety of the nodes in the transportation network. For example, the problem may be to get people to the half of the city that is higher ground or safety may even be equated with elevation of each intersection above sea level. Often city emergency planners have safety maps for various scenarios. There is also a capacity constraint on nodes so that solution must conform to physical capacities of nodes and edges. We use both real and test city networks. Traffic congestion is modeled using classical traffic formulas. This problem specification is very general in that we optimize routing based on a arbitrary safety function which can be established for each type of event and so is not necessarily limited to disasters, although this paper focuses on evacuation planning and response.

We approach the problem of evacuation as a dynamic problem that needs to be repeatedly solved as a disaster unfolds to handle the unforeseen changes in the problem

parameters. This we feel is an added strength of an evolutionary solution. A diverse population in nature or *in silico* maintains a set of contingent genes that help it adapt to changing environments. We will attempt to exploit this.

Our solution is to treat the evacuation as a problem of optimizing the sum of the safety of all the people at a time t. While more complex and interesting optimizations can be done, we believe that this building block optimization is the "proof of concept" we will base future work upon. We use an evolutionary approach in which we maintain a population of potential solutions and evolve an evacuation plan (see details below). We hypothesize first: that with an effective evacuation representation, we can efficiently solve a wide variety of evacuation scenarios using an Evolution Algorithm. Second, if during the evacuation unexpected changes in the transportation network, the safety of regions, or the evacuee's position occurs, the population of evacuation plans in the algorithm can be used to more quickly compute a response to the change than if the response was evolved from a restart of the optimization. That is, as one would expect, the population acts as a reservoir of innovation for the adapting of an evacuation to changing conditions. Experiments are performed on a variety of problems to test and support both hypotheses.

We proceed as follows: Section 4.2 discusses previous work, Section 4.3 discusses our algorithm, Section 4.4 describes in detail a suite of different problems to challenge our algorithm, and Section 4.5 presents a summary of conclusions.

Background

Evacuation planning is not a new topic. Previous work has addressed problems varying from room or building evacuation [20], [8], to city or region evacuation [1], [2], [12]. Optimizing traffic while considering the dynamics of traffic interaction (e.g. congestion [3]) is known as Dynamic Traffic Assignment (DTA) [17]. Our approach models evacuation as a DTA problem with the single objective of optimizing the safety of the evacuees. Some authors consider evacuation as a multi-objective optimization problem [18], [22].

Saadatseresht et al. optimize evacuation with respect to capacity and nearness of safety zones, while Yuan and Han suggest that evacuation is an inherently multi-objective optimization problem, and recommend optimizing space-based risk and travel time.

Stepanov and Smith also approach evacuation planning as a multi-objective optimization problem in [19], where they use Integer Programming instead of evolution. A route-based approach is used. This has the limitation of having to establish routes for each individual, creating a specific solution that may be ignored.

Attempts have been made at handling roadway congestion using genetic algorithms, as well. In Dezani et al. [4], Petri net analysis is used to evaluate fitness in a GA designed to optimize routes for urban traffic, strictly to reduce roadway congestion in real-time. Their approach is not multi-objective, and optimizes routes for shortest time. However, their approach also focuses on route-based solutions for individuals, and optimizes for minimization of travel time alone.

Our solution focuses on safety optimization, without determining routes for each traveler. This provides a generalization for DTA, removing the requirement that each vehicle follow a specific path.

Approach

We use an Evolution Strategy (ES) [5] to evolve sets of probabilities that determine the probability p_{ne} of traffic at node n exiting on edge e with the constraint that

$$\sum_{e \in E(n)} p_{ne} \leq 1 \quad \forall \quad n \tag{4.1}$$

where $E(n)$ is the set of edges leaving node n. The amount of probability left between the sum above and 1 is the probability of remaining at that node before trying again to leave the node. The fixed cycle-time is specified for each node when the node is defined. This approach does not build explicit routes but rather relies on probabilistic routing at each

intersection to move traffic to safety. An advantage of this is that local information about the best route to safety is stored at each intersection. The disadvantage is that in this first implementation, the probabilities are static and only change with re-optimization.

The genome for the ES is the vector of the probabilities given by the list of $p_{ne}\ \forall\ n$. Fitness at time t is computed:

$$fitness(E,t) = \frac{1}{||E||}\sum_{a\in E} S(L(a,t)) \tag{4.2}$$

where E is the set of all evacuees, t is time, $S(\ell)$ is the safety of location ℓ, and $L(a,t)$ is the location of agent a at time t which can be a node or edge. Because this is an average of safeties, the fitness may make trade-offs based on level of safety across the whole population. This is a practical consideration for the user of the application.

We chose to use an $ES(\mu+\lambda)$ algorithm. For many of our tests we used, $\mu = 100, \lambda = 100$, and $\sigma = 0.1$ (see specific test results). Using the + operator, the child population competes against the parent population for survival. This is performed using a simple sort operation, followed by removal of individuals with the lowest λ fitness scores. Mutation probability $P_{mutation}$ for a selected genome is 0.5 for each real number in the genome. Mutation happens by mutating the value by a normal distribution with σ. Unlike many ES there is also crossover with probability $P_{crossover} = 0.2$. Crossover is uniform crossover respecting that all probabilities for a given node change together.

It is important to note that *time* appears in two contexts in this paper. First, *simulation time* refers to the amount of time simulated during a fitness evaluation (the t in 4.2). Second, the amount of real-world time it takes to run the model for a given problem, referred to as *wall-clock time*.

The SLang Language

We developed a language called SLang (Simulation Language) that can be used to specify cities and optimizations so that tests can be easily generated, run, saved and shared. SLang is used to specify such things as city topology, the level of safety of intersections, and agent distribution. SLang also specifies experiment parameters for the evolution strategy algorithm and simulation of traffic. SLang is also used to control I/O features of the code.

SLang commands are used to specify all necessary information needed by a model. Therefore, in our model we use SLang to indicate ES parameters such as population size, σ and the maximum number of generations. SLang is also used to specify the traffic network data, indicating freeflow time, simulation time, and more.

Simulation

During simulation agents move through the provided topology, according to the current set of probabilities being evaluated. Agents are stored in a priority queue, sorted by time of next arrival at an intersection. Time progresses in steps governed solely by the next arrival at a intersection.

The simulation operates by dequeuing an agent from the queue, checking the arrival time of that agent compared to the current time in the simulation, selecting the next edge that agent will take based on the probabilities for the node the agent is currently at, calculating the traversal time of the selected edge, then re-queuing the agent. This cycle repeats until the next agent in the priority queue has an arrival time exceeding the total simulation time (see Figure 4.1).

We defined the travel time t between nodes using the classic Bureau of Public Roads (BPR) formula [9]:

$$t = f + b\left(\frac{c}{c_{max}}\right)^{\beta} \tag{4.3}$$

where f is **freeflow time** of the edge, which is the time required to travel the edge in ideal

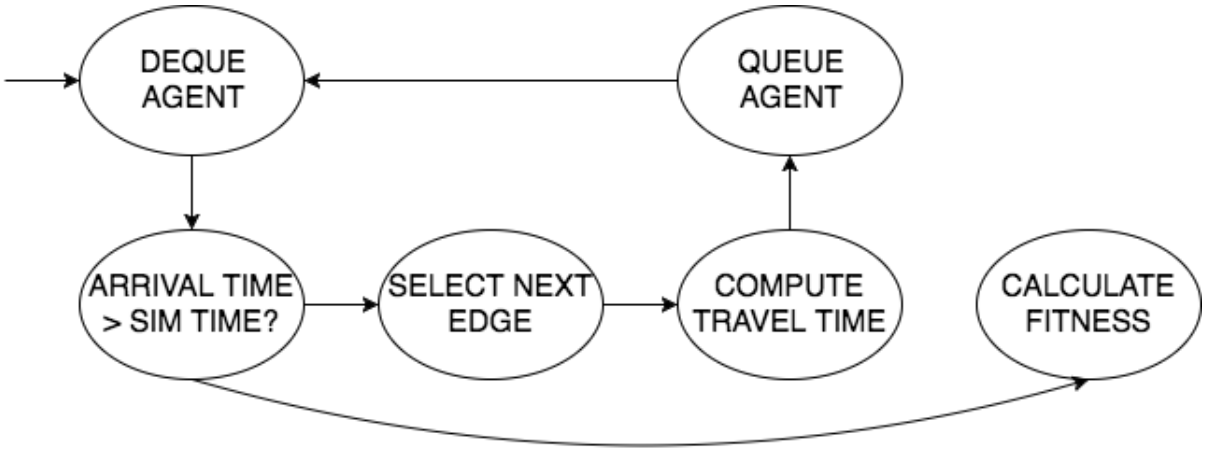


Figure 4.1: A diagram of the how time progresses using a priority queue, during simulation.

conditions, typically meaning no other traffic. For our simulations **all time is measured in hours**. c is the current number of evacuees on the edge, while c_{max} is the maximum number of evacuees that the edge can handle per unit time (hour). This is a combination of speed limit, number of lanes, road condition, and length of road. b and β are tuning variables used to fit real data measured from actual roadways. For our purposes we used the default values of $b = 4.00$, $\beta = 0.15$, and $c_{max} = 1000$ but is dependent on road size, speed limit and length. Note that the formula does not include explicit distance or speed limit values, as those values are implied by the freeflow value, f, as a result of $t = distance/speed$.

Historically, in normal traffic conditions with working traffic lights, c_{max} must account for total throughput per unit time. For instance, the value is reduced from 1000 by a proportion equivalent to the proportion of "green time" of the traffic light at the intersection.

Once the simulation is complete, the set of probabilities used in the simulation are evaluated for their fitness. Fitness is a sum of the safety of all the agents (see Equation 4.2). The maximum safety for any location is 1.0. Because fitness is currently evaluated only at the end of the simulation, fitness is a measure of safety at a specified time t.

This fitness function was chosen to allow adaptability to any disaster since safety can be designated by any factor. For example, a flood or tsunami event might measure safety as both distance from the event, as well as elevation. An earthquake may quantify safety as distance from tall structures or other hazards. Some events will use distance alone, but other events may have more complex safety functions.

In summary, a fitness evaluation is computed as follows:

1. ***Initialization***

   The set of probabilities to be evaluated is loaded into the city topology; the city has been defined in SLang prior to any fitness evaluations and the set of probabilities must match that topology.

   Agents are initialized in their starting locations and times; their initialization values are specified in SLang before any fitness evaluations are made. Typically, agents are initialized with an *arrival time* of zero seconds.

   Finally, the *current time* is set to zero seconds. This value is replaced with each next agent evaluated, and is replaced with the next agent's *arrival time*.

2. ***Simulation***

   The main activity during simulation consists of popping agents from the priority queue one at a time, evaluating their next move and arrival time at the next node, and replacing them in the priority queue at the appropriate location. In the case where an agent chooses to stay in a node, the agent's next time of arrival is specified by the node's *wait time* field, and is generally fixed.

3. ***Fitness***

   The fitness of the set of probabilities is calculated when the simulation time has expired, using Equation 4.2.

Experiments

A series of experiments were performed to test our software and hypotheses. The tests included **validation tests** to show that the code works in simple transparent problems, **dynamic tests** to show that the code can use populations to adapt to changing problem parameters such as capacity, **comparison tests** to compare our work to the work of others, and **scaling tests** to see if the code will work for topologies and traffic found in a real city. While we ran many tests, due to publication size, only a subset of tests is given in this paper.

Validation Tests

To validate our model we performed a variety of simple transparent tests. First, we created a test with a 5×5 grid, with all edge and node variables equal. Agents were initialized to begin in the top left corner of the grid, while safety was only in the two nodes in the bottom right of the grid (see Figure 4.2). The program quickly routed all evacuees to safety.

To validate our model we tested that capacity and freeflow constraints effected our results as expected. We recorded the path to safety indicated by highest probabilities calling it the selected path. We then repeat the experiment by creating a single path from danger to safety by reducing the freeflow time of the edges along the selected path, while increasing freeflow along every other edge in the graph the path is easily found, as it should be. Similarly, if we increase maximum capacity along the selected path, while decreasing maximum capacity along all other edges, the path is again easily found.

We then constructed a maze as seen in Figure 4.3. The path indicated in the figure was marked in one test by high capacity and marked by low freeflow time in the other test. Notably the safety was zero except for the safe zone at the end. Both tests found the easier routes highlighted in the figure. As the number of evacuees increases past the

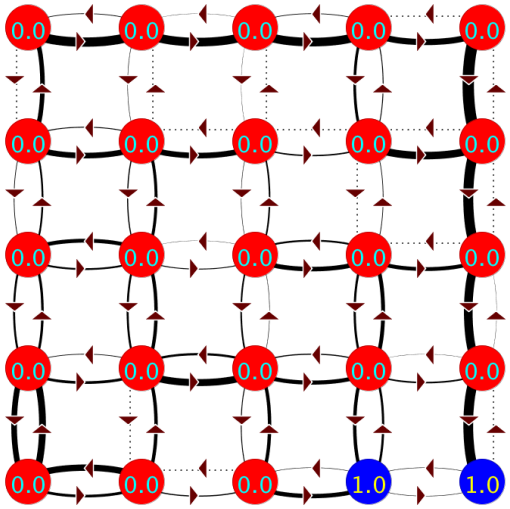


Figure 4.2: Simple 5×5 grid validation test. The heavier weight edges in the figure show higher probabilities. The thinnest lines are generally very near zero but are just drawn to show the topology. The numbers in the nodes are level of safety. Unless otherwise stated, best case of several is shown.

limit of the easy path side streets become used to share the load.

Dynamic Tests

The following experiments test if our model accommodates dynamic events during an evacuation. We hypothesize that if probability distributions are optimized in advance, they can be used to initialize an evolution strategy (ES) to reduce optimization time, both in real time, and number of generations without loss of accuracy. The results of these tests demonstrate that pre-optimization leads to faster adaptability in response to real-time events. This is a **very important feature** for a real-world evacuation management system.

For these experiments we make several assumptions. First, we assume changes are not

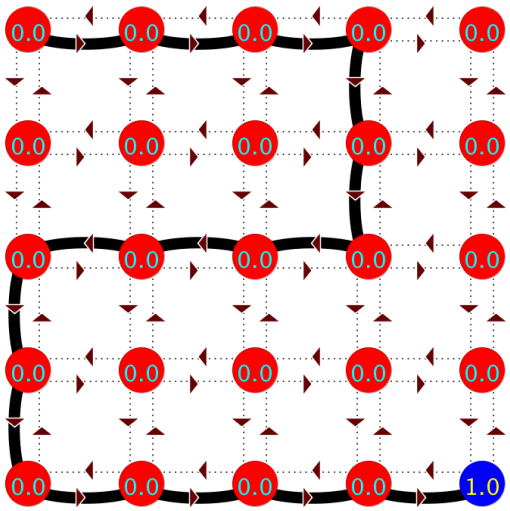


Figure 4.3: A cartoon of 5×5 maze test showing the route of least resistance. In this case, these are not a probabilities but rather just an indication of where we tried to make the roads faster or higher capacity.

significantly large. We assume an algorithm's population will have genetic contingency information usable for similar problems. For example, we assume that from one optimization to the next during an evacuation, we won't suddenly switch from West side of the city is safe to East side is safe. In fact, in such a case where the graph characteristics have been changed significantly, re-optimization using a previous solution may take significantly longer than using random initialization. The events we are concerned with are changes in topology, safety zones, capacity, and agent distribution.

To test our hypothesis, we perform two initial optimizations (case 1) we optimize a population of solutions that are initialized with random values for the initial graph configuration and (case 2) we optimize a population of solutions that are initialized with random values for the graph configuration after some change in topology, safety, capacity, or vehicle distribution has been modified. Finally, (case 3) we use the final population

from the first optimization to initialize the starting population used to optimize the second graph configuration. Results will be labeled 1, 2, or 3 corresponding to the three different cases. For each test, the ES parameters were kept the same for the sake of comparison. The parameters are $ES(100 + 100), P_{crossover} = 0.2, P_{mutation} = 0.5, \sigma = 0.1$

Adapting to Capacity Changes

During an evacuation event a roadway may see a change in capacity as a result of becoming partially blocked by debris, accidents in one or more lanes on a multi-lane road, or as a result of other unpredictable events. In such a case our model must be able to quickly adapt through re-optimization. We predict that using pre-optimized traffic probability distributions will allow for quick adaptation. To test this prediction we run three tests. First, we create a topology with a maximum capacity of 1000 agents per unit time for all edges, G_1. Next, we create a topology that is identical, except that maximum capacity is lowered by 100 for each edge, G_2.

We then optimize both of these graphs using our models and record the average number of generations until maximum fitness is achieved. We also store the population of solutions found from G_1, as they will be the starting population for optimization in the next test. For the third part of the test we re-optimize G_2, starting with the solutions from G_1. We record and compare the number of generations needed to reach maximum fitness below. The probabilities are graphical represented in Figures 4.4 and 4.5. Since the landscape is flat and load is light, many evacuations plans are equivalent and so variation in the probabilities occur. Solution time is shorter in case 3. Results are shown Table 4.1.

Adapting to Topology Changes

Another event that might occur during an evacuation is a change in topology, meaning some route or partial route becomes unusable, such as a bridge collapse or a road becoming flooded or completely blocked by debris. It is important that our model handles

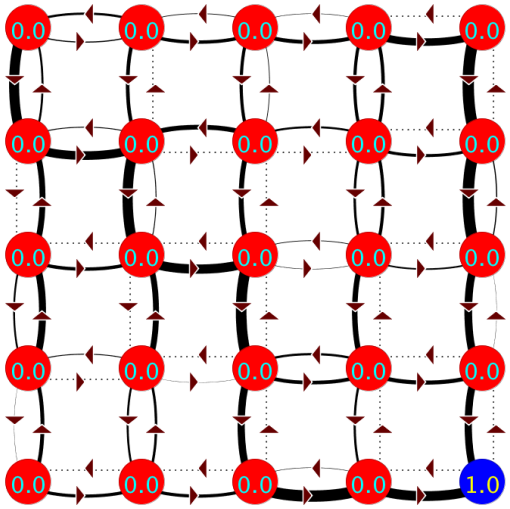


Figure 4.4: 5×5 capacity test results for case 1.

such changes quickly. We make a similar prediction to the capacity change test as well. Note that a change in topology is a more extreme version of a change in capacity where maximum capacity is reduced to zero for a roadway. We again run three tests but using the bridge topology. The first test is on G_1, which represents three bridges connecting two areas, with safety being on the side opposite where vehicles are started for these tests (See Figure 4.6). G_2 for this test is identical to G_1, except with an edge on the center "bridge" removed representing a collapse. The first two graphs are optimized from random traffic assignment probabilities. We then use the solutions from the optimization of G_1 to optimize G_2, and record the number of generations for comparison (See 4.7). Results are shown Table 4.1. Solution quality is similar but time to solve is substantially shortened.

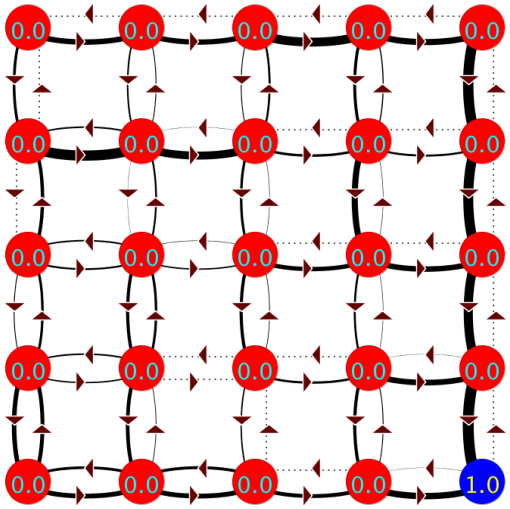


Figure 4.5: 5×5 capacity test results for case 2.

Adapting to Agent Distribution Changes

In an evacuation, it is expected that individual drivers will ignore evacuation instructions and instead upon arriving at and intersection rely on their experience driving in the area. This is expected because people are under stress and optimal traffic assignments may produce non-intuitive results in an attempt to balance traffic load on many streets at once. In such cases, we must adapt and adjust based on where vehicles actually are not where we expect them to be. This is another reason why it is crucial to continually re-optimize to adapt to the unforeseen. We make a similar prediction here - our model will find an optimal solution faster if initialized with a pre-computed solution for a similar traffic distribution. Again, we use three tests. All tests use an identical topology, but G_1 has traffic distributed differently than G_2. The first two runs optimize for different traffic distributions, starting with randomized traffic assignment probabilities. The third run uses the optimized traffic assignment probabilities from the first test, but with the traffic

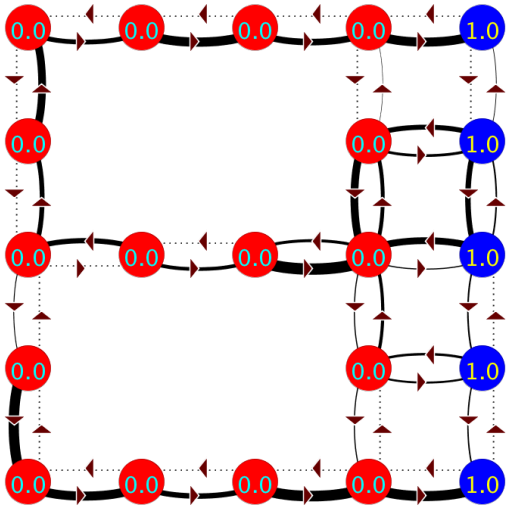


Figure 4.6: The three bridge test. Probabilities for bridge intact and random initialization.

distribution of the second run. The traffic probability distributions are qualitatively not different but the time to solve is much lower. Results are shown in Table 4.1.

Adapting to Safety Changes

During many evacuation events, the threat creating the need for evacuation is not static. Examples include hurricanes, floods, tsunamis, poison gas. As a result of the dynamic nature of the threat, it is imperative that our model handles changes in safe areas quickly. We predict that our model handles changes in safety quickly, again provided the change is not too great. Three runs are used to perform this experiment. The first and second are again identical in topology (see Figure 4.8). In the case 3 test the safe nodes move over one node (see Figure 4.9). The optimized traffic assignment probabilities from run one are used to initialize the third run, which has safety in the same location as run two. Results are shown Table 4.1. Safety was easily solve quickly. Perhaps a more difficult safety test

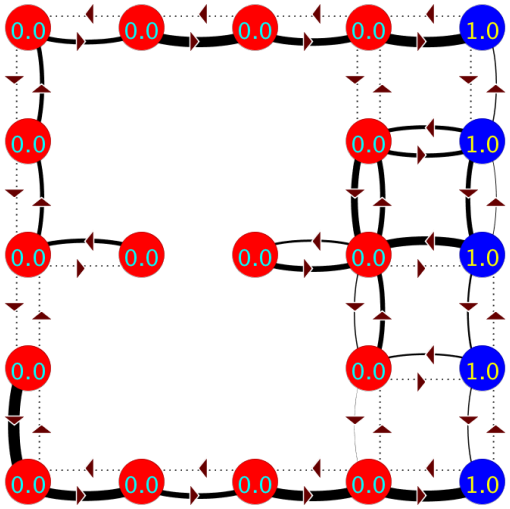


Figure 4.7: The three bridge test. Probabilities for bridge destroyed and population initialized with population from previous solution.

will tease out the difference between case 1 and 3 for this problem.

Summary

In the dynamic tests we adjusted one of four attributes to represent events that can change a transportation network in an evacuation. Graphs shown in Figure 4.11 show that priming the population with a population that produced a successful solution for the pre-event problem produces a solution more quickly than just starting the optimization over. Figure 4.10 shows the quality of the answers is substantially the same. We believe that as long as the change is modest the population contains "contingency genes" that will help speed the adaption to the new problem. These results are encouraging, but we are following with more focused tests in our future work.

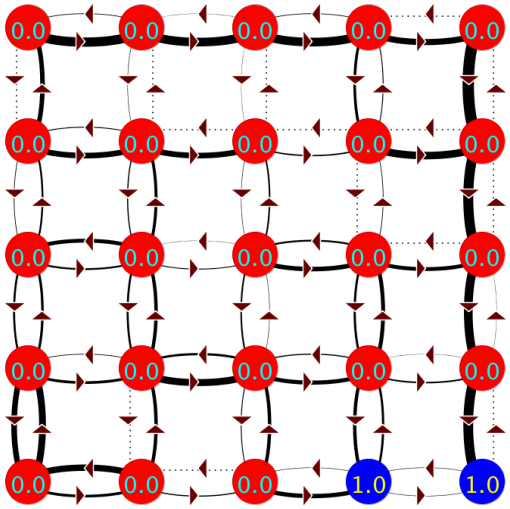


Figure 4.8: 5×5 safety test for case 1.

Comparison Test

Here we compare our model to the approach used by Bazzan et al. in [1]. In their paper they seek to minimize average travel time by evolving traffic assignments through the network, given a set of vehicles, origin/destination pairs for those vehicles, and a set of k shortest paths. They use a classic graph from [15], (see Figure 4.12) with the freeflow times specified. The way we model capacity is different, but comparable. We seek to compare our results to provide some context of how our model holds up compared to other work.

In their experiment they run k origin/destination pairs, with k ranging from 2 to 16. We compare to the case where $k = 4$. We run our experiment with two vehicle origin nodes, corresponding to their origins, with two safe nodes, corresponding to their destinations. We let our model optimize for safety, and provide the results and comparison below. Their evacuation (network clearance) time is significantly lower. We believe this is probably due, in part, to differences vehicle scheduling and is an area for future analysis.

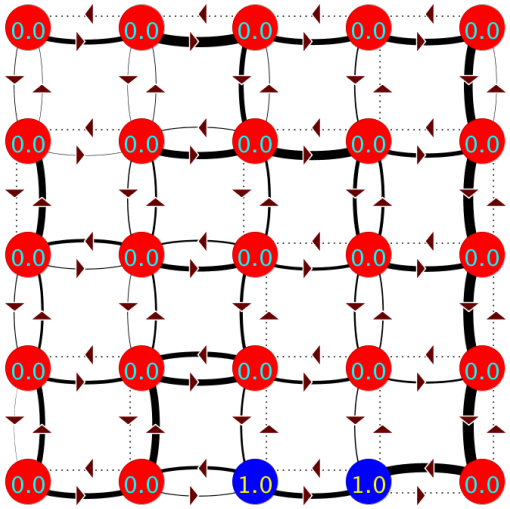


Figure 4.9: 5×5 safety test for case 3.

While our model does implicitly find routes to safety, the routes found are not as fast as those found by Bazzan et al. The average time to safety (TTS), with standard deviation is shown in Figure 4.13.

Scaling Tests

To test the ability of our model to handle real-world urban evacuation problems, we used traffic model data for Boise, Idaho, a city with approximately 650,000 people. Our goal was to discover the wall-clock time it takes to optimize such a city, using our model. For this experiment we made simple assumptions about graph properties where we didn't have accurate data (freeflow time, maximum capacity, etc). Our data includes 283 nodes with 457 edges. These do not represent every road and intersection in Boise, but rather many of the artery roads and feeders (see Figure 4.14). We believe the vast majority of the evacuee's time will be spent on these roads and so the test is a realistic estimation

Table 4.1: Generations to Complete Optimizations in Dynamic Tests: Capacity, Topology, Agents, Safety. In all cases the suffix numbers 1, 2, or 3 represent the three cases in each of the dynamic tests. 100 runs of each case were performed. "Gens" is generations and t_{hours} is the simulated time.

Test	Avg. Gens	σ_{Gens}	Avg. Fit	σ_{fit}	t_{hours}
Capacity1	47.75	5.208	1.0	0.0	1.0
Capacity2	55.1	6.64	1.0	0.0	1.0
Capacity3	4.35	2.19	1.0	0.0	1.0
Topology1	73.54	15.835	1.0	0.001	0.5
Topology2	69.92	13.9	0.998	0.009	0.5
Topology3	7.01	4.574	1.0	0.0	0.5
Agents1	77.99	9.605	0.999	0.003	0.7
Agents2	89.69	8.577	0.998	0.004	0.7
Agents3	30.5	8.325	1.0	0.0	0.7
Safety1	43.02	8.359	1.0	0	1.0
Safety2	40.855	8.2	1.0	0	1.0
Safety3	34.5	4.574	1.0	0	1.0

of evacuation time. We feel that Boise is a good representation of a real example, as it includes a mix of urban downtown areas and suburban neighborhoods.

We ran a set of 6 experiments 10 times each, and averaged the wall-clock time each experiment took. Each experiment used the same ES and city parameters, varying only in the way and number vehicles were distributed. Each experiment was initialized with a base-set of agents (representing 10 vehicles each) at each node. This base-set is used to ensure agents begin at every node, thereby applying evolutionary pressure across the whole genome. Not applying pressure to probabilities at each node could produce dangerous results in a real-world application, as nodes may end up with traffic assignments that are no better than random. Vehicle group starting locations are random in each test. Vehicles are grouped to improve run time. Each experiment simulated one hour of traffic. The number vehicles, number of groups, size of each group, and wall-clock run time are shown for each experiment in Table 4.2. The experiment shows that for a real world city of over a half-million people execution times on the order of minutes is a reasonable expectation.

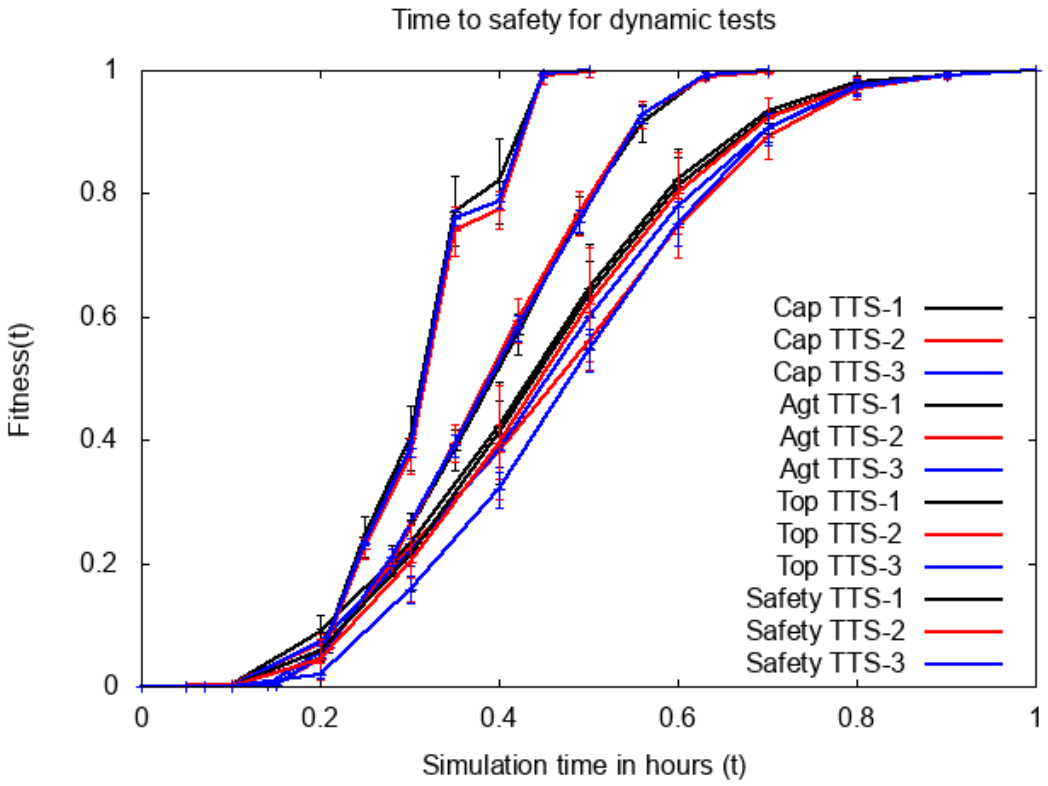
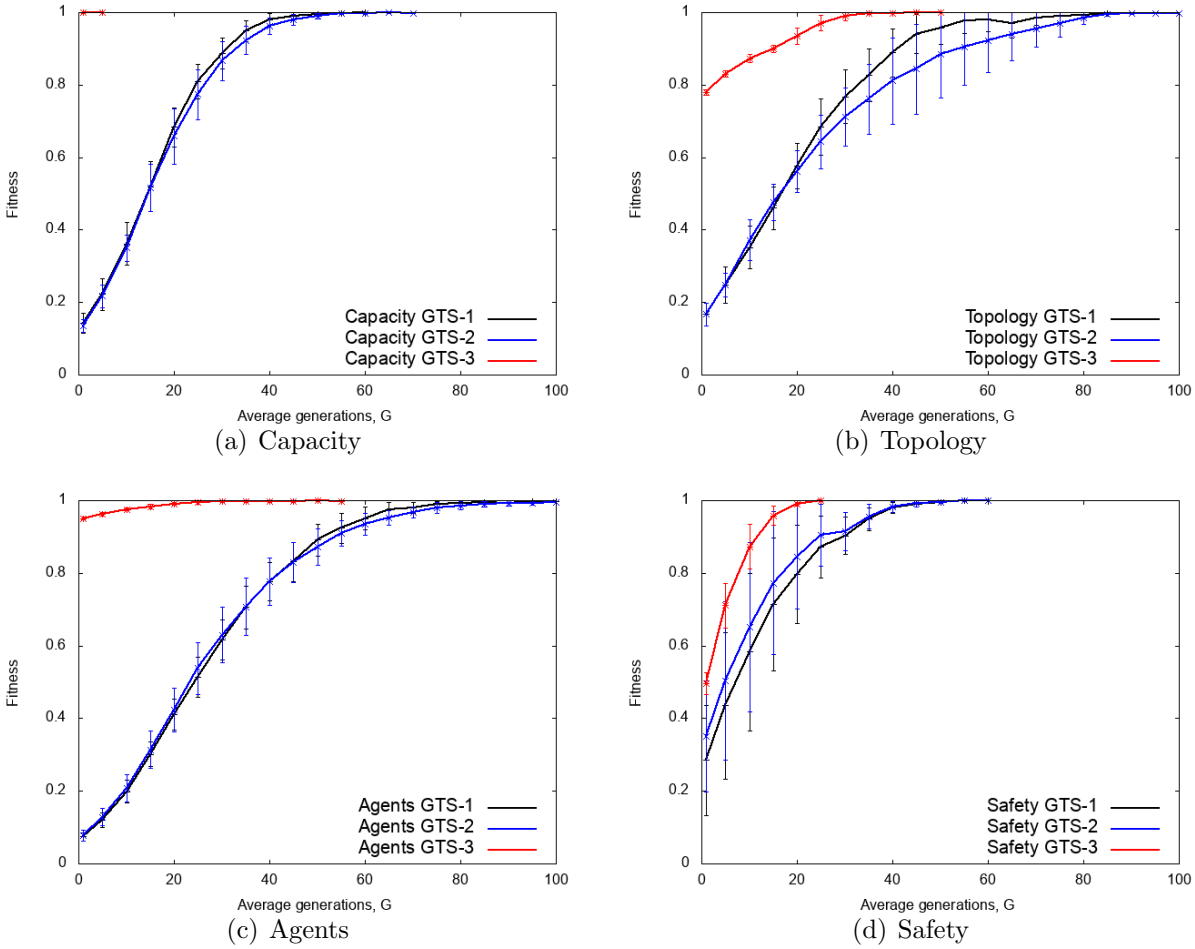


Figure 4.10: A comparison of the quality of answers for the four dynamic tests. This indicates the answers are of comparable quality.

Conclusions

In conclusion, we have evolved static probability distributions that are shown to provide routing to safe areas in reasonable time. Further, the solutions used are provided as initializations for evacuation parameter changes, which reduce the optimization times of the updated scenarios. This makes an evolution based real-time continuous optimization of an evacuation feasible. The main advantages of our proposed model includes a robust method for adapting to the dynamic conditions of a real disaster, relatively fast computation time, and robustness achieved through both evolution and a vehicle distribution model, instead of a more ridged point-to-point routing for individual agents.



(a) Capacity (b) Topology

(c) Agents (d) Safety

Figure 4.11: The speed of solution for the four dynamic tests and across cases 1, 2, and 3. Generations along X-axis and fitness along Y-axes. Values averaged over 100 runs.

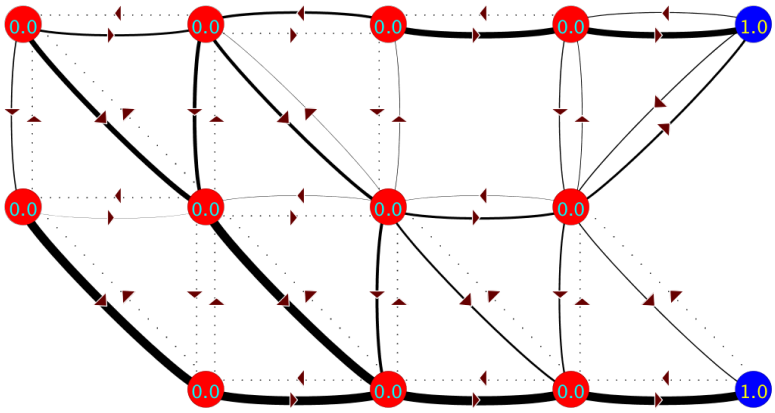


Figure 4.12: The network from Bazzan et al. 2014.

Future work includes the use of OpenStreetMap [14] for generating graphs representing urban areas. Also, self-adaptive parameters for the ES algorithm, diversity promotion, multi-objective optimization, and novel mutation operators.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454. We would like to thank the U.S. National Science Foundation (NSF), the State of Idaho, and the M.J. Murdock Charitable Trust, for making this research work possible. NSF under CyberCorps® awards 1027409 and 1565572 and BigSTEM award 1229766. Idaho under IGEM grants. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the State of Idaho. We thank Michigan State faculty member Dr. Kalyanmoy Deb for his collaboration

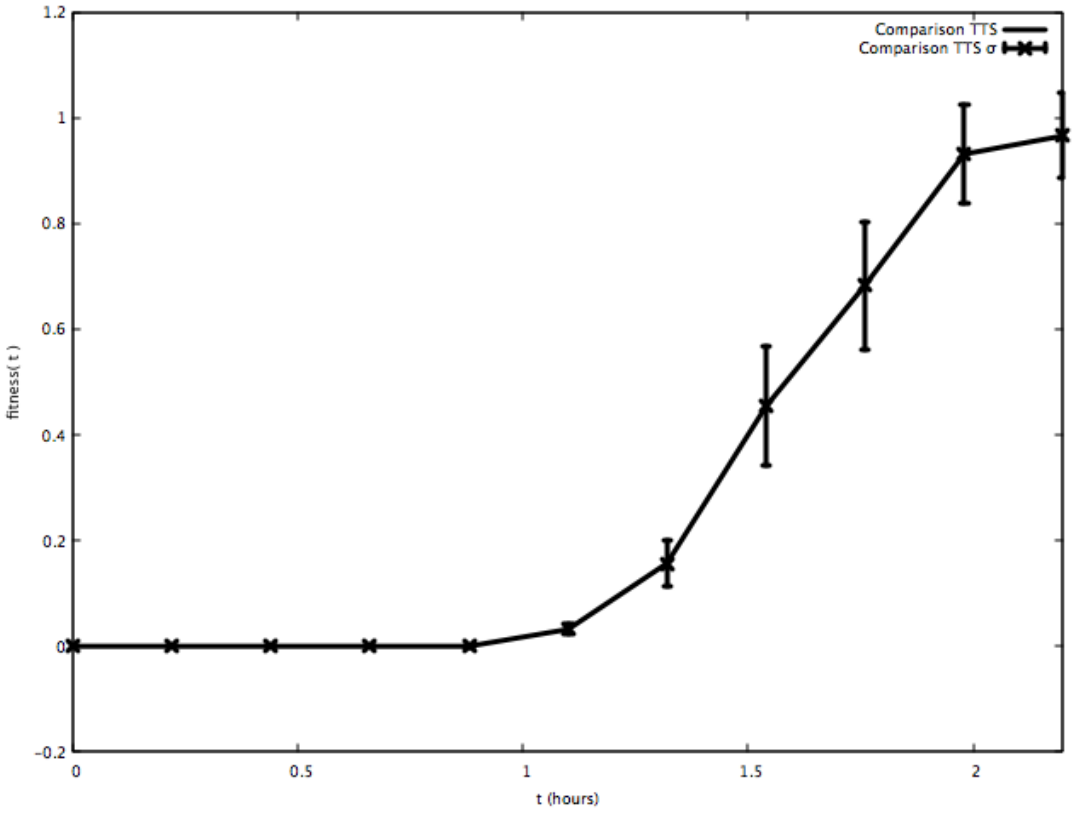


Figure 4.13: Performance of our algorithm on the Bazzan test averaged over 100 trials.

in this research.

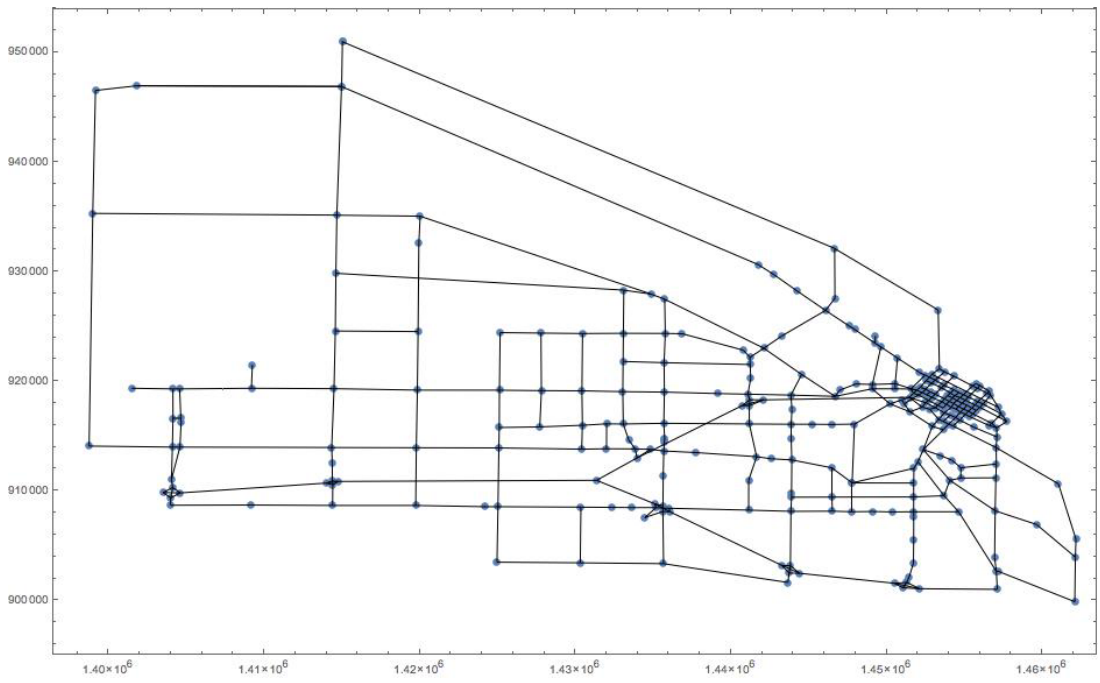


Figure 4.14: The Boise Transportation Network. Notice the dense packed streets in the downtown area. The map is of an area approximately ten miles wide.

Table 4.2: Time to solve the Boise problem. Vehicles is the number of vehicles being evacuated. Num Groups is the number of randomly placed groups of agents in the city above the groups that are evenly distributed across the 283 nodes of the city. Group size is the number of individuals in each group or agent. It defines the grouping of the agents. For timing the runs were performed using Centos Linux system running on 2.4 GHz Intel Xeon X56xx processors.

Vehicles	Num Groups	Group Size	Run Time (min)
2,830	0	10	3.5
2,930	10	10	3.28
3,830	10	100	3.5
12,830	100	100	4.4
102,830	100	1000	4.25
1,002,830	1000	1000	19.05

Chapter 5: Additional Experiments

In this chapter an experiment testing the effectiveness of different population sizes used by the Evolution Strategies (ES) algorithm is described. Next, an approach to reducing the number of generations needed to reach an ideal fitness value is discussed. An experiment designed to test this approach is described, as well.

Testing Population Size Parameters

In an ES algorithm, population size greatly effects running time of the algorithm. Search and Sort algorithms are evaluated in terms of the number of elements that must be searched or sorted. Evolutionary Algorithms (EA) use the *number of evaluations*, a function of the population size. For ES algorithms, it is a function of the child population size, specifically. To improve run time for the model, it is important to find the smallest population size that still performs effectively, under the assumption that minimizing runtime is important for a real-time solution. The experiment here is a simple one, and seeks to answer the question: *what is the smallest effective population size for the Boise traffic network?* The question is specified for Boise because the most effective population size will almost certainly differ from one network to another.

To answer this question, we designed an experiment which runs the same optimization on ten different population sizes, $\lambda = \mu$, ranging from ten to 100. The experiments each run for 1000 generations. Given 1000 generations, the number of evaluations for each experiment is equal to $1000\lambda+\lambda$. The additional λ is included to represent initialization of the population at the beginning of the ES. The wall clock running time of each experiment ran from one hour to one day. The experiments were run 10 times each, and the results presented here are averaged from those runs. The ES parameters used are as follows: $ES(\lambda + \lambda), \sigma = 0.1, P_{xover} = 0.2, P_{mutation} = 0.5, t\_size = 3$. The simulation time for each fitness evaluation was set to four hours. All areas of safety are placed on the Western side

of the Boise traffic network, while the majority of vehicles start on the Eastern side of the network. Approximately 60,000 vehicles are simulated, beginning in the Eastern portion of the network. Another 2,830 vehicles are included in each simulation, with 10 starting on each node in the network, to ensure selective pressure is applied to each gene in an individual's genome.

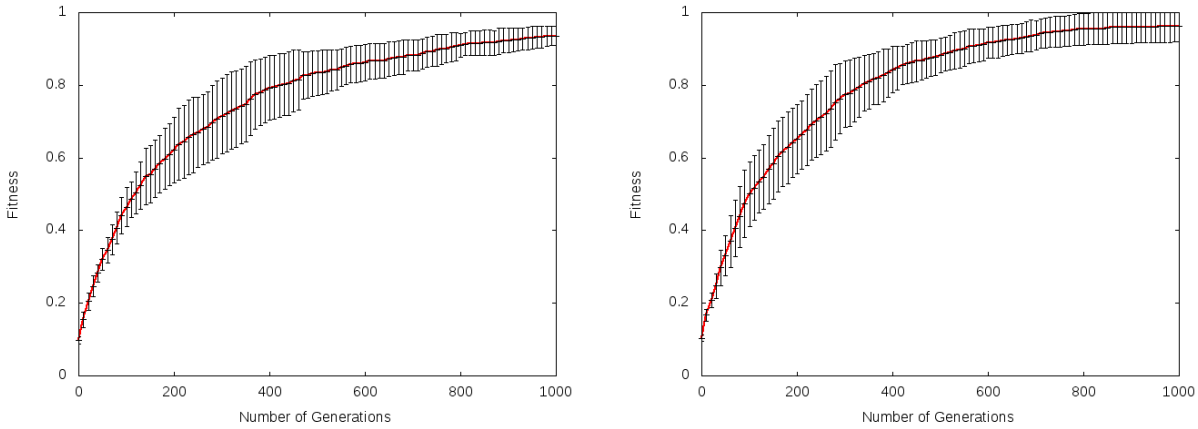
The fitness at each generation is shown in Figures 5.1 and 5.2. Each graph represents the fitness at each generation for each population size tested. It is clear that a higher number of individuals finds a high fitness in less generations.

In [22], Yuan and Han argue that 100% network clearance is not often reached, therefore a 95% clearance rate is more meaningful. Figure 5.1 shows that a population of size 10 is not sufficient, reaching an average of 93.6% evacuation after 1000 generations. However, a population of 20 and greater does yield network clearances that are on average greater than 95%. Figures 5.1 and 5.2 also indicates a trade-off between the number of generations to run the ES algorithm for versus the population size. As the number of individuals in the population increases, the number of generations needed to reach 95% network clearance seems to decrease. Ideally, the model would like to reach 95% fitness in 100 generations or less. That doesn't seem to be reasonable given the current model. However, if the population was initialized with probabilities that suggest where safety is, the number of generations needed to reach 95% network clearance might drop significantly. This is the topic of the next section.

Population Initialization

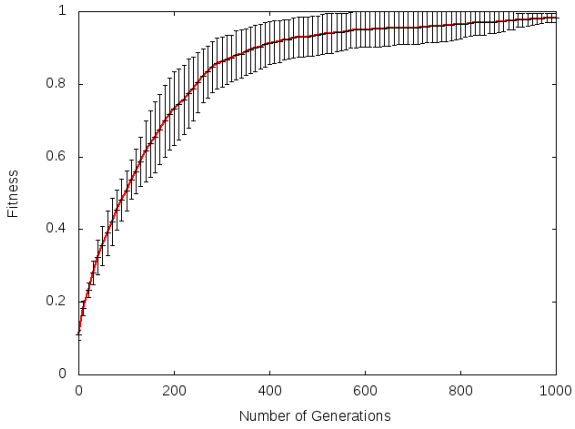
As mentioned in the previous section, we wish to reduce the amount of time it takes to optimize traffic assignments for large networks. We hypothesized that initializing a population of solutions in a way that provides them with an idea of where safety is to begin with might reduce computation time.

There is another problem in our model that we wish to address, regarding the way

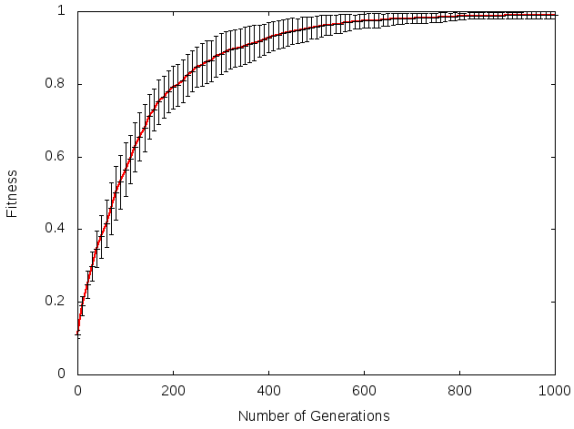


(a) Population 10

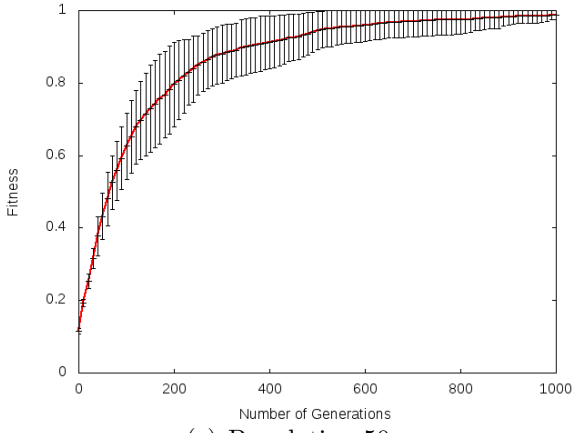
(b) Population 20



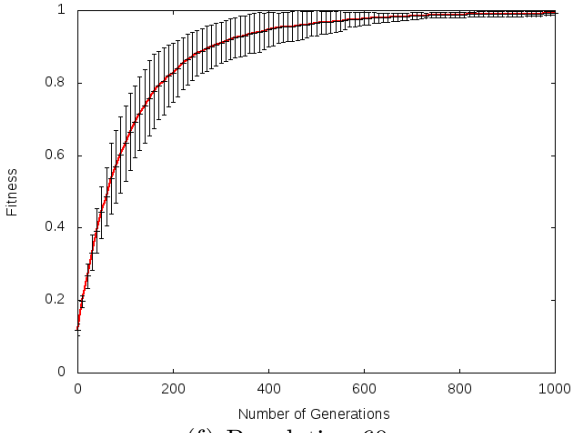
(c) Population 30



(d) Population 40

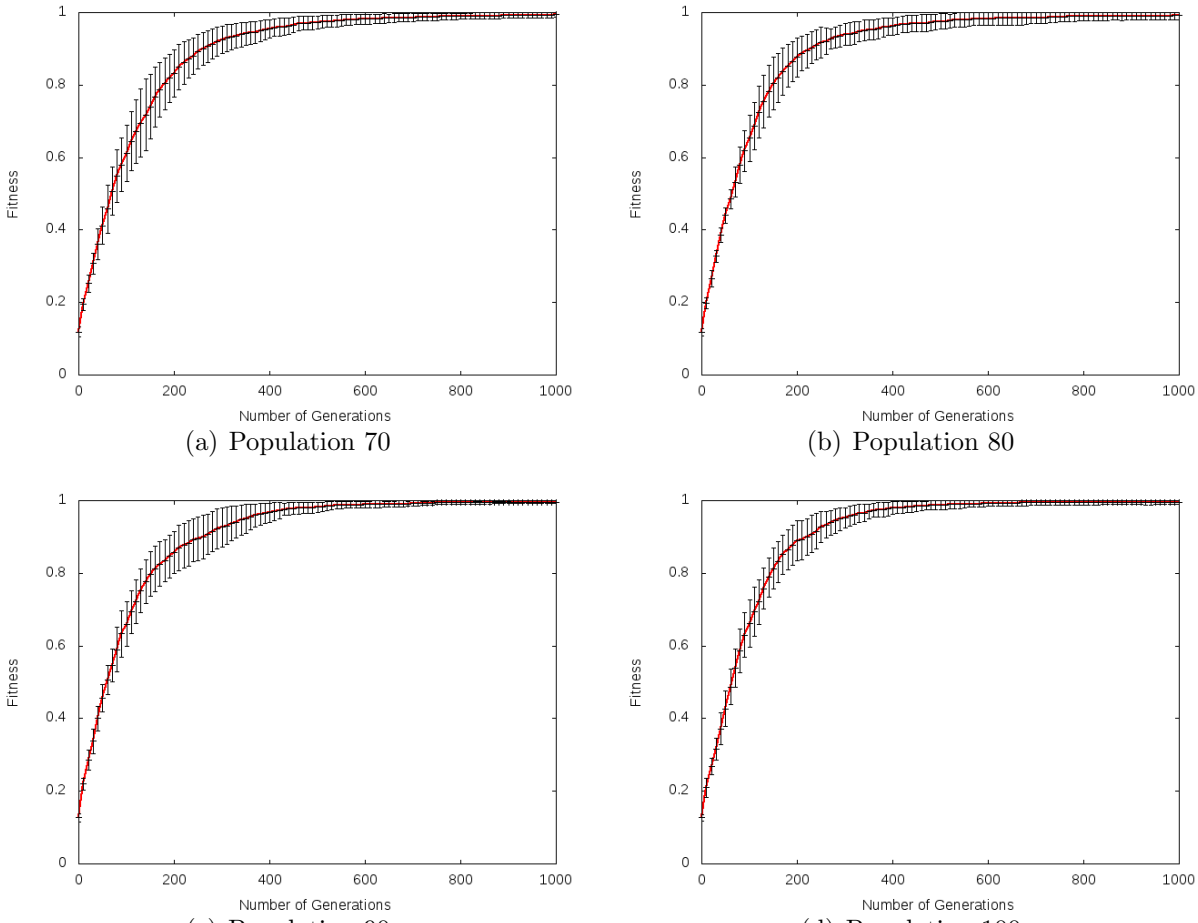


(e) Population 50



(f) Population 60

Figure 5.1: Generations x Fitness for population sizes 10-60



(a) Population 70

(b) Population 80

(c) Population 90

(d) Population 100

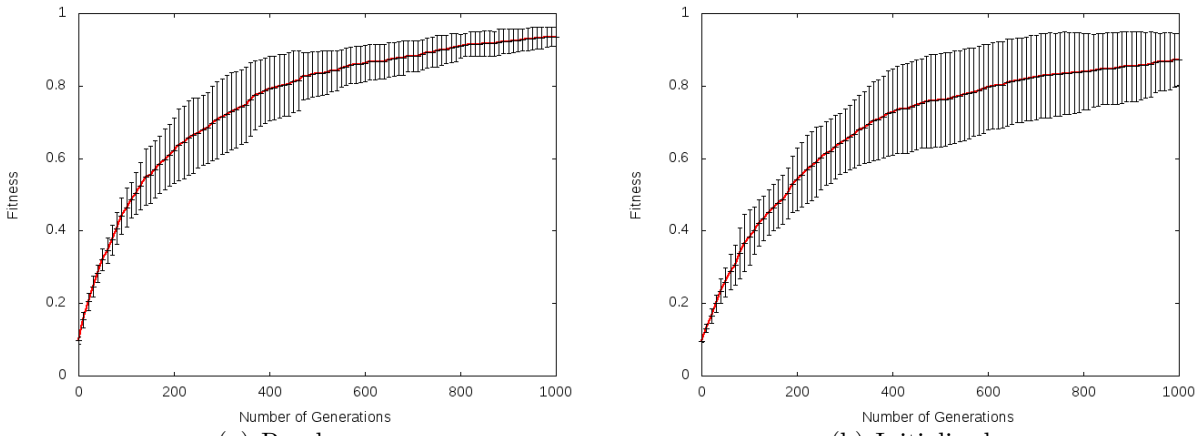
Figure 5.2: Generations x Fitness for population sizes 70-100

evolution works with our representation. In a sufficiently large traffic network, it is fair to assume that, for some nodes, there will be no traffic. In that case, no selective pressure is applied to the allele in question, and can result in probabilities no better than random. This is a problem for real-world situations. Should a vehicle find themselves in an area with random traffic assignments, that vehicle might be directed towards danger, or get stuck in a loop.

To mitigate this problem we decided an approach was needed to ensure that all nodes in any network contribute to the fitness of each genotype. Our initial idea was to simply place a vehicle at every node in each simulation, however this idea is more computationally expensive than desirable, and could also represent congestion inaccurately. This approach also does nothing to reduce the number of generations needed to reach an idea level of safety. The approach described was designed to apply pressure to all nodes in a genotype, as well as reduce computation time.

To initialize the genotypes, we place a vehicle population in sets of nodes that begin with the safest areas in the traffic network, and work out from them incrementally. These increments start vehicles in the network progressively further from safety, until vehicles have been initialized at all nodes. This applies selective pressure to each node. Initialization is performed as a special variant of our optimization algorithm, with the goal of providing a population that "knows" where safety is, in advance of any evacuation.

Our initialization algorithm is designed to find all maximum safety nodes in the traffic network, according to some safety map. Then optimization is performed with some number of agents in each of the nodes. The best performing individuals are saved and used as the initial population for the next optimization. The next optimization starts agents in the neighbors of the nodes used in the previous optimizations, moving them effectively one node away from maximum safety. This is repeated until optimization has been performed with agents beginning in every node. The result is a population of *seeded* individuals, for which every node has contributed to the genotype's fitness. This population can then be



(a) Random (b) Initialized

Figure 5.3: A comparison of initialized vs. random probabilities.

used to start optimization for an evacuation event that has a similar safety function to the one used for initialization.

We compare the results of this approach with the experiment of the previous section. A population size of 10 was used, run for 1000 generations, and averaged over 10 runs. The graph of fitness per generation is shown in Figure 5.3.

Unfortunately, the results of this approach were inconclusive. It is important to pressure each node in the genome, however this method did not improve our results. It is worth mentioning that while the approach does not seem to yield consistently better results, it proved to be faster in terms of wall-clock time. This method did decrease run time by about 20 minutes, but a larger sample size is needed to be sure of the results.

Despite inconclusive results, two facts are apparent: 1) optimization of the model is still needed. This includes idealizing the ES parameters, perhaps evolving σ, and increasing the parent-child population size ratio. The other apparent fact is 2) pressure needs to be applied to each node in a traffic network solution. The real-world complexities of evacuating a large area are significant; relying on a predictive model of human behavior in such a case can be dangerous.

Optimization of this model will require further research. Optimizing parameters for

the ES algorithm has the potential to significantly reduce wall-clock run times. Another area of investigation for speedups lies in ES operators. It may be the case that search the solution space can be done in a more effective manner.

This work has described two methods for applying pressure to each node in a traffic network. First, a brute-force method of starting a vehicle at every node in every simulation was discussed. Next, the algorithm presented in this section was discussed. The potential approaches to applying this pressure have not been exhausted, and this remains a fundamental area of interest for this work. Some other options include initializing the routes using known path-finding algorithms such as Bazzan et al. in [1]. While their work focused on using k-shortest paths, other options include Dijkstra's Algorithm and A*.

One other significant option is to let an expert initialize the probabilities by hand, or with computer assistance. This is a brute-force method, but seems intuitive. Experts may still want to designate evacuation zones, allowing this model to optimize traffic assignment for congestion mitigation and changes in the evacuation environment.

Chapter 6: Conclusion

The work described in this thesis explores the feasibility of a real-time evacuation management system based on evolution of static traffic assignment probabilities. The specific question this work sought to answer was *Can evolution effectively optimize traffic assignment as a function of safety, in advance of and during an evacuation, while responding to changes in safety, topology, and vehicle distribution?*

Evolution Strategies (ES) were found to be effective at routing traffic based on locations of safety. Experiments described in this work answer specific portions of the question posed, while some aspects will better be answered in future works. Specifically, the experiments within show that evolution strategies are well suited for the problem, given our encoding, and the model presented is effective for preemptive evacuation planning and responding to changes in a traffic network quickly.

The contributions presented in this work consist of 1) a new approach to the Dynamic Traffic Assignment (DTA) problem for evacuations (evolving static probabilities), 2) an ES algorithm and representation that allows optimization of those probabilities for vehicle safety, and 3) a priority-queue-based traffic simulation which evaluates how well the set of probabilities routes traffic to safety.

Future Work

Reviewers of the paper in Chapter 4 suggest applying several modifications to the approach described within, specifically adaptive parameters for the Evolution Strategies (ES) algorithm and a multi-objective optimization approach. ES algorithms often co-evolve the mutation step size, σ along with the population of individuals being optimized. Other ES parameters that might co-evolved include the child population size or the mutation probability, P_m, as in Simulated Annealing. Other objectives to optimize along with safety could be time and space. These objectives are fundamental parts of evacuation and traffic

assignment, as it is important to achieve high network clearance times quickly, without routing vehicles through dangerous areas, if possible.

Other future work areas include exploring ways to increase population diversity, seeding algorithms, and experimentation and comparison with real-world data. There is also room for exploring integration with other commonly used traffic simulation tools [7], [11].

Increasing population diversity would be helpful for effectively searching the solution space. There are a number of existing techniques that can be adapted to this model and tried. Seeding algorithms are similar to the population initialization experiment described in Chapter 5. Such algorithms may prove useful to emergency planners who have expertise with the network in need of evacuation, eliminating the need for this model to "learn" what the expert already knows. Commonly used traffic simulation tools, such as VISSIM and SUMO provide unique opportunities for this software. VISSIM can be used to evaluate the solutions provided by the model described here, while helping to make the model more accurate. SUMO, an open-source traffic simulation tool, could be adapted to use the traffic simulation described in this work. SUMO integration could provide microscopic views of traffic behavior during optimizations.

Should these areas of future exploration increase the efficacy of our model, additional real-world applications also become areas of future work. How can this model be effectively used in a real-world situation? Should the model be supported by smart-phones? Vehicle-to-vehicle ad hoc networks? The "cloud"?

References

[1] Ana LC Bazzan, Daniel Cagara, and Bjorn Scheuermann. An evolutionary approach to traffic assignment. In *Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2014 IEEE Symposium on*, pages 43–50. IEEE, 2014.

[2] Yi-Chang Chiu. Traffic scheduling simulation and assignment for area-wide evacuation. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 537–542. IEEE, 2004.

[3] Yi-Chang Chiu, Jon Bottom, Michael Mahut, Alex Paz, Ramachandran Balakrishna, Travis Waller, and Jim Hicks. Dynamic traffic assignment: A primer. *Transportation Research E-Circular*, (E-C153), 2011.

[4] Henrique Dezani, Regiane DS Bassi, Norian Marranghello, Luís Gomes, Furio Damiani, and Ivan Nunes Da Silva. Optimizing urban traffic flow using genetic algorithm with petri net analysis as fitness function. *Neurocomputing*, 124:162–167, 2014.

[5] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Spinger, second edition, 2015.

[6] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[7] Martin Fellendorf. Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority. In *64th Institute of Transportation Engineers Annual Meeting*, pages 1–9. Springer, 1994.

[8] S Gwynne, ER Galea, M Owen, Peter J Lawrence, L Filippidis, et al. A review of the methodologies used in evacuation modelling. *Fire and Materials*, 23(6):383–388, 1999.

[9] A. J. Horowitz. *Delay/Volume Relations for Travel Forecasting Based Upon the 1985 Highway Capacity Manual*. Federal Highway Administration, U.S. Department of Transportation, 1991.

[10] G Kotusevski and KA Hawick. A review of traffic simulation software. 2009.

[11] Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner. Sumo (simulation of urban mobility)-an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, pages 183–187, 2002.

[12] Qingsong Lu and Shashi Shekhar. Capacity constrained routing for evacuation planning. In *Intelligent Transportation Systems Safety and Security Conference (Miami, Florida), USDOT*, 2004.

[13] Ada City-County Emergency Management. City of boise evacuation plan. `https://adacounty.id.gov/Portals/Accem/Doc/PDF/boiseevacuationplan2010s.pdf`, mar 2010.

[14] OpenStreetMap contributors. OpenStreetMap. `https://www.openstreetmap.org`, 2017.

[15] J. Ortúzar and L. G. Willumsen. *Modelling Transport, 3rd ed.* John Wiley & Sons, 2001.

[16] Lucio Sanchez Passos, Rosaldo JF Rossetti, and Zafeiris Kokkinogenis. Towards the next-generation traffic simulation tools: a first appraisal. In *Information Systems and Technologies (CISTI), 2011 6th Iberian Conference on*, pages 1–6. IEEE, 2011.

[17] Srinivas Peeta and Athanasios K Ziliaskopoulos. Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, 1(3):233–265, 2001.

[18] Mohammad Saadatseresht, Ali Mansourian, and Mohammad Taleai. Evacuation planning using multiobjective evolutionary optimization approach. *European Journal of Operational Research*, 198(1):305–314, 2009.

[19] Alexander Stepanov and James MacGregor Smith. Multi-objective evacuation routing in transportation networks. *European Journal of Operational Research*, 198(2):435–446, 2009.

[20] Peter A Thompson and Eric W Marchant. A computer model for the evacuation of large building populations. *Fire safety journal*, 24(2):131–148, 1995.

[21] Femke van Wageningen-Kessels, Hans Van Lint, Kees Vuik, and Serge Hoogendoorn. Genealogy of traffic flow models. *EURO Journal on Transportation and Logistics*, 4(4):445–473, 2015.

[22] Fang Yuan and Lee D Han. A multi-objective optimization approach for evacuation planning. *Procedia Engineering*, 3:217–227, 2010.