

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**ADAPTIVE RULES IN EMERGENT LOGISTICS (ARIEL)
AN AGENT-BASED ANALYSIS ENVIRONMENT TO STUDY
ADAPTIVE ROUTE-FINDING IN CHANGING ROAD-NETWORKS**

by

Thomas Orichel

June 2003

Thesis Advisor:

Eugene Paulo

Co-Advisor:

John Hiles

**This thesis is done in cooperation with the MOVES Institute.
Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE June 2003	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE: Adaptive Rules In Emergent Logistics (ARIEL) – An Agent-based Analysis Environment To Study Adaptive Route-finding in Constantly Changing Road-Networks.			5. FUNDING NUMBERS	
6. AUTHOR(S) Thomas Orichel				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) JWAC (joint warfare analysis center)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release: distribution unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) The delivery of supply in combat operations is very important and often results in success or failure of a mission. This activity, as well as other transportation problems, has traditionally been modeled using global optimization techniques, such as linear programming. However, the goal of this thesis is to examine the feasibility of an agent-based solution to study the movement of material through a road network. The requirement is to build an agent-based system that finds the optimal route through a given road network and is capable of adapting to disruptions introduced to the network and then find alternative routes through the network. The agents act from a local perspective, and can represent more realistically the decisions being made throughout the delivery process. This thesis implements an analysis environment for road networks and develops an agent-based model to build truck-driver agents that are capable of delivering supplies through a changing road network.				
14. SUBJECT TERMS complex adaptive systems, agent-based modeling, multi-agent systems, optimization, network-routing			15. NUMBER OF PAGES 68	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release: distribution is unlimited

**ADAPTIVE RULES IN EMERGENT LOGISTICS (ARIEL)
AN AGENT-BASED ANALYSIS ENVIRONMENT TO STUDY
ADAPTIVE ROUTE-FINDING IN CHANGING ROAD-NETWORKS**

Thomas Orichel
Captain, German Army
Dipl.-Ing., University of Federal Armed Forces Munich, 1994

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS AND
SIMULATION**

and

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
June 2003**

Author: Thomas Orichel

Approved by: Eugene Paulo
Thesis Advisor

John Hiles
Co-Advisor

Rudy Darken
Chairman, MOVES Curriculum Committee

Peter Denning
Chairman, Computer Science Department

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The delivery of supply in combat operations is very important and often results in success or failure of a mission. This activity, as well as other transportation problems, has traditionally been modeled using global optimization techniques, such as linear programming. However, the goal of this thesis is to examine the feasibility of an agent-based solution to study the movement of material through a road network. The requirement is to build an agent-based system that finds the optimal route through a given road network and is capable of adapting to disruptions introduced to the network and then find alternative routes through the network. The agents act from a local perspective, and can represent more realistically the decisions being made throughout the delivery process. This thesis implements an analysis environment for road networks and develops an agent-based model to build truck-driver agents that are capable of delivering supplies through a changing road network.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	AREA OF RESEARCH	1
1.	Description.....	1
2.	Agent-based Approach	1
B.	PROJECT OVERVIEW	3
1.	Goal	3
2.	General Scenario.....	3
II.	AGENT-BASED MODELLING	5
A.	OVERVIEW.....	5
B.	COMPLEX ADAPTIVE SYSTEMS	6
C.	CURRENT AGENTS AND AGENT MODELS	7
1.	Wooldridge's Agents.....	7
2.	Hiles' Composite Agent Structure.....	10
3.	VanPutte's Maria Agent Model.....	12
4.	Ferber's Multiagent System Equation	14
III.	THE ARIEL MODEL	17
A.	GENERAL MODEL.....	17
B.	MODEL INPUT SETTINGS	19
C.	MODEL APPROACH.....	20
1.	Layered Design.....	20
2.	Information Flow and Global Knowledge	21
3.	Inner Environment and Map Reading.....	22
4.	Model Flow	24
D.	MODEL DETAILS.....	26
1.	Basic entities	26
2.	Destruction layer Entities.....	29
3.	Gradient Encoding.....	31
4.	Propensities.....	33
a.	<i>Skepticism</i>	33
b.	<i>Propensity for Routing Decision</i>	34
c.	<i>Learning</i>	36
5.	Performance Evaluation.....	37
6.	Combination of the elements.....	38
IV.	VALIDATION.....	41
1.	Experimental Setup	42
2.	Experimental Runs	43
V.	SUMMARY AND FUTURE WORK	47
	LIST OF REFERENCES.....	49
	APPENDIX.....	51
	INITIAL DISTRIBUTION LIST	53

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	Problem overview	3
Figure 2.	Wooldridge's agent structure	9
Figure 3.	Composite Agent Architecture	11
Figure 4.	Maria Agent Architecture	12
Figure 5.	Ferber's multi-agent systems equation	14
Figure 6.	Network Layer	20
Figure 7.	Layered Design	21
Figure 8.	Sensor radius of the truck-driver agents	22
Figure 9.	Gradient Encoding	23
Figure 10.	Model Flow	25
Figure 11.	Basic Entities	28
Figure 12.	Disruption layer entities	30
Figure 13.	Detailed Encoding Process	32
Figure 14.	Skepticism	34
Figure 15.	Points of decision-making	34
Figure 16.	Decision after finishing current road (arc)	35
Figure 17.	Decision after finishing the current node	35
Figure 18.	Learning Propensity	36
Figure 19.	Loss Factors	37
Figure 20.	Composite ARIEL agent	38
Figure 21.	Test Network for Evaluation	41
Figure 22.	Trace of Best Route	42
Figure 23.	Results for $R_i=2000$, uncertain network	44
Figure 24.	Results for $R_i=2000$, certain network	45
Figure 25.	ARIEL Class Diagram	51

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author would like to express his thanks to many people who were very helpful to the completion of this thesis. To LTC Eugene Paulo, for his support, guidance and motivation to do this work. To John Hiles, who stimulated my interests in the field of agent-based technology and helped me building and refining the agent-based model. To Prof. Peter Purdue for his helpful advice and support. To Mr. Frank Mahnke and Mr. Paul Schneider for the financial support and the opportunity to study multi-agent systems in a real problem domain. Finally, to my wife Karin, my daughter Jasmin and my son Jan for their support and huge patience. Without the security and support of a loving home, none of this would have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. AREA OF RESEARCH

1. Description

This thesis explores the capabilities of multi-agent systems in addressing logistical transportation problems. The project ARIEL (Adaptive Rules In Emergent Logistics), which was funded by the Joint Warfare Analysis Center (JWAC) served as a basis for this work.

The central goal of project ARIEL is to show the feasibility of multi-agent systems to move material through a road network. This problem has traditionally been solved by global optimization techniques.

2. Agent-based Approach

The main difference between an agent-based approach and traditional global optimization techniques is that agents act individually and make their decisions based on a local view of the world. Generally, this method allows a more human-like implementation of the decision-making process.

Agents, as opposed to other software objects, are situated in an environment. They are located at a specific position in their environment and are only able to oversee (access) the surrounding environment in a certain range. The agents can act upon their environment within a defined radius, which may differ from the viewing range. Essentially they do not have global knowledge of the world and cannot modify the entire environment, which gives agents a human-like perspective of their environment.

Agents make their decisions based on three aspects. The first aspect is their local view of the world. They sense all information from the local area, which is or might be interesting or important for them. The local view is defined by the filtered information from the environment and the agent's own interpretation of this information. The second aspect is the agent's history, which is build by recordings of the agent's actions and the

perceived responses from the environment. The agent evaluates this historical information for jumps, or slower, more steady developments. Patterns that the agent recognizes will be used to build up a specific knowledge about its environment. The third aspect is the agent's individual personality. The agent has certain preferences and propensities that will make his actions differ from other agents. Those propensities will be modified to fit the current environmental conditions. A certain diversity in the agent's behavior is beneficial for realistic simulations.

Another important property of agent based systems is the improvement over time. That is achieved by a step-by-step emergent optimization of the agent performance. Each decision that the agent makes results in a change of his movement or a change of the local environment. Each step contains the decision of an agent. The results of the agent's decisions are evaluated by an appropriate metric, which indicates the quality the agent's performance during the last decision step(s). According to this self-assessment each agent adapts his behavior (personality) to get better next time.

In contrast to the traditional optimization approach, agent based systems can learn and specialize themselves on a certain type of problem. They can include other variables which could not be handled previously (behavior, goals) and in that way add realism and enrichment to the model. Agent based systems usually operate on a near optimal level, yet have enough flexibility to solve dynamic problems.

The agent-based approach in this research utilizes all of these agent characteristics, including their responses to the environment, decision-making aspects, and improvement over time. The specifics of how this is accomplished is the focal point of Chapter 3.

B. PROJECT OVERVIEW

1. Goal

The project goal is to examine the feasibility of using an agent-based solution to study the movement of material through a road network. The system should determine the best possible routing through the road network in dynamic conditions. Due to disruptions, the road network is subject to changes. The system adapts to those changes and takes an alternative route. That way it is possible to explore the impact of those disruptions and observe the adaptability of the system.

2. General Scenario

Combat operations rely on timely logistical support. The delivery of supply is very important and often results in success or failure of a mission. Modeling a realistic scenario to represent a combat road network is critical to this research.

We begin with a typical road network, as seen in Figure 1 below. The depots, also called sources, are known and have fixed locations in the network. They have an inventory of supplies. The destination facilities, also called sinks, are also known. They have a certain demand for supplies. However, the current condition of the roads and the enemy situation are not known. Trucks have to deliver material from the sources to the sinks and there are multiple possible routes to take. They need to find an optimal route which provides a delivery at the lowest possible “cost”.

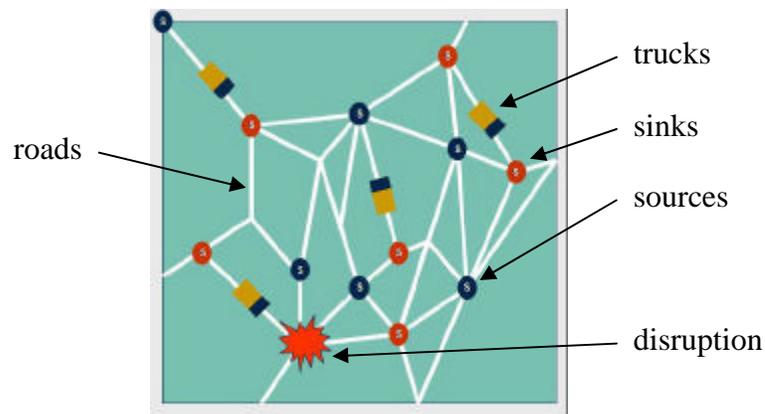


Figure 1. Problem overview

THIS PAGE INTENTIONALLY LEFT BLANK

II. AGENT-BASED MODELLING

A. OVERVIEW

A huge step in software engineering is the orientation to objects, which essentially combine data or properties with the methods or abilities together in one construct. Many associated concepts like inheritance became possible through object orientation and made software more reusable and as a whole more manageable.

The invention of software agents was as important, and is a similar step in the evolution of software engineering. Compared to ordinary objects, agents show the following three differences.

- Agents are autonomous. They have a stronger notion of autonomy, which means that they can decide themselves when and whether or not to perform an action.
- Agents are smart. They are capable of flexible and context dependent behavior. This is shown through interaction with other agents or with objects in the shared environment.
- Agents are active. They have at least one active thread of control, which made independent and autonomous actions possible [Wooldrige].

*Agents are concerned with their own welfare,
although acting on behalf of the same user/owner
[Wooldrige]*

The key problem in building agents is the creation of an autonomous acting software construct that allows them to cooperate and coordinate to satisfy their interests, even if their individual interests might conflict. Social science is a rich repository of concepts which have been used to design agents. At the same time agent based simulations have been used as an experimental tool for understanding societies and to investigate social processes. The BIOWAR project, for example, studies disease outbreaks using social networks [CARLEY]. In general it is a very good tool to gain insights in complex problem domains.

B. COMPLEX ADAPTIVE SYSTEMS

When we look at a city like San Francisco with its thousands of shops, restaurants and hotels, we could ask how this complex system is controlled. For example, when we walk into a bakery to buy 10 bagels, we expect that the clerk has bagels available. But how does the clerk know how many customers he may have and how many bagels he needs to bake for that day.

In this example we see two important features of complex adaptive systems. The first is decentralized control. There is no central planning agency, as instead each actor is autonomously trying to reach its own goal (to succeed). Over time the bakery owner figures out himself how many bagels are sufficient to satisfy his customers. Essentially history (his experiences in the past) is important for that decision. The second feature is the interaction among various actors with different intentions and goals. Assuming there are other bakeries in the neighborhood, our bakery owner needs to constantly adjust his prices and offered products to at least match what the others are doing, in order to be competitive and to keep his market share.

There are a variety of complex adaptive systems in the world. Other examples are: the central nervous system, where millions of neurons are interacting; the immune system, which needs to continuously adapt to fight new invaders; or the classic example of an ecosystem, where adaptation is crucial for surviving. In these complex adaptive systems many different actors (agents) interact with each other.

The important quality of those actors is their ability to adapt to their environment. In real life we can find several ways how this adaptation is achieved. Almost all adaptation procedures rely on a certain amount of history. If the bakery owner realizes that last week he sold less bagels, he will either bake less bagels the next week, or he improves the taste of his bagels. This is a shorter adaptation cycle in the order of weeks. If he has had his bakery for a long time, he might know the differences in bagel consumption of each month. This is learning and is usually involved in a longer cycle. In biology we see a very time consuming but effective mechanism which operates in the order of generations: the natural selection process. Key elements are random mutations

and environment guided selection. If an actor does not adapt to environmental changes he will die. Others who happen to better fit into their ecological niche will survive.

Those are very useful concepts. They have been used in a modified and sometimes time compressed way in simulations of complex adaptive systems.

C. CURRENT AGENTS AND AGENT MODELS

We traditionally think of agents to exist physically as software on computational devices. They are used in various applications, like electronic commerce, web crawler, and security systems. In simulations, agents are used to model reality, and mostly to model complex adaptive systems. It is important to keep in mind is that those models are not reality, like all models. The modeler uses simplifications, aggregations and his own perception or assumptions about the real world system to create an artificial image of the real world. Those models should be used only to gain insight or to explore the problem domain, but never to draw conclusions from the model or to generate predictions for the real world.

1. Wooldridge's Agents

Although agents are used in various applications there is no commonly used and accepted definition so far. Wooldridge summarizes the notion of agents the following way:

Agents are computer systems that are capable of autonomous action in some environment in order to meet their design objectives. An agent will typically sense its environment and will have available a repertoire of actions that can be executed to modify the environment, which may appear to respond non-deterministically to the execution of these actions [Wooldridge: p17].

Agents are situated which means they are capable of perceiving their environment through sensors. At the same time they have a repertoire of actions available to act upon their environment. Therefore they have the ability to at least partially modify their environment.

Agents are intelligent, so they pursue their goals, execute tasks to optimize some given measure of performance and are capable of flexible autonomous actions. Agents make their own decisions and they can take the initiative as opposed to simple reacting. This is made possible through an active thread of control for each agent. At all times an agent has full control of the execution of its methods, as no matter who requests an action to be performed by an agent, the agent decides whether or not he will perform that action.

Objects do it for free; Agents
do it because they want to.
[Wooldridge]

Wooldridge suggested the following capabilities may be expected of an intelligent agent:

Reactivity:

If an environment is guaranteed to be fixed, an agent does not need to worry about its own success or failure and can execute blindly. But most real environments are dynamic. So a successful agent must maintain an ongoing interaction with its environment. This includes the perception of the shared environment and the response in a timely and fair fashion to changes that occur in it, in order to reach its design objective. This is an ongoing process with no end or termination, unlike functional systems.

Proactiveness:

Reaction to an environment is easy to implement (as for example a simple thermostat). But we do not want agents that are driven solely by events. Generally we want more complex behavior, in that agents are capable of taking the initiative. We want an agent to commit to a set of goals and attempt to achieve those goals by choosing the appropriate actions.

Social abilities:

Usually environments include multiple agents and their interactions. So when an agent tries to achieve his goals, he has to take the other agents into account. Agents are affected not only by the environment they are situated in, but also by other agents with different intentions and goals. Some goals might be only achievable with the cooperation

of others. Some might be not achievable at all, because the goals are conflicting. Agents then must be able to reach agreements or, if necessary, drop a specific goal because it is unachievable. Within these interactions, agents are communicating their own beliefs and in return they are capable of sensing and recognizing the beliefs of others and whether those beliefs are conflicting. Therefore a successful agent must be able to interact with other agents (possibly humans) to satisfy their design objectives.

Wooldridge lists some additional properties which are sometimes discussed in the context of agents. Mobility is the ability of an agent to move around an electronic circuit. Veracity means the agent will not knowingly communicate false information. Benevolence means the agent will always do what is asked of it. Rationality is the property of an agent, such that it will always act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved – at least insofar as its beliefs permit. Learning and adaptation are mechanisms which allow agents to improve over time.

Wooldridge’s model of a basic agent looks as follows:

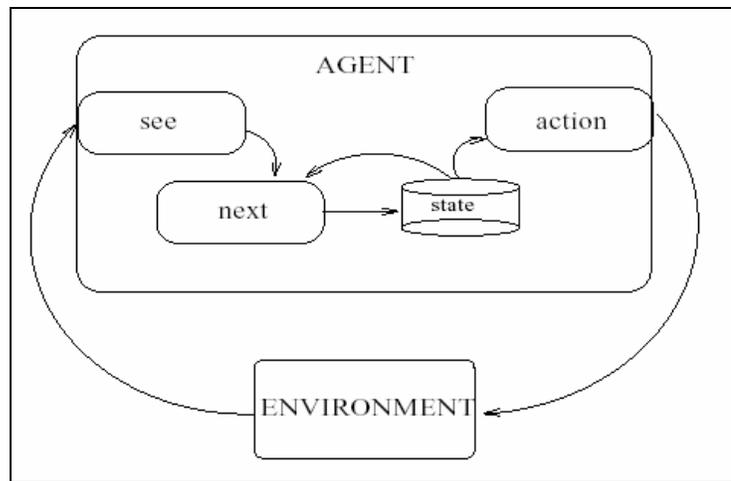


Figure 2. Wooldridge’s agent structure

As seen in Figure 2, Wooldridge’s agents have a “see” portion and an "action" portion. The "see" portion allows an agent to observe its environment. Essentially it maps environmental states to a set of perception constructs, which are referred to as percepts. In addition the agents show an internal data structure, which is typically used to record

information about the environment state and history, as well as the current internal state of the agent. The next function maps the combination of the current internal state and the available set of percepts to a new internal state. This allows an agent to change its internal state according to environmental changes (the context). The “action” portion finally maps current internal states to appropriate actions. These actions will have an effect on the environment. The agent perceives the changed environment and the process starts all over again. All those aforementioned intelligence and social abilities are packed into those three basic functions: “see”, “next” and “action”.

2. Hiles’ Composite Agent Structure

Multi-agent systems simulations typically consist of numerous high-level agents, which represent entities operating in a common shared environment. The environment, or more precisely the "outer environment", is what is sensed by the agents and where they interact with each other. Based on their perception and interpretation of the environment, the agents make their decisions as to what action to take.

Reactive agents typically are more action oriented. They map their perceptions of the world into actions. Cognitive agents sense and interpret their environment. They do not need much effectory capability at all, as their strength is the evaluation, understanding and translation of sensory input.

Hiles’ composite agent structure is an attempt to capture the strength of both cognitive and reactive agents and at the same time simplify the design of such a complex agent (see Figure 3). A composite agent contains a set of cognitive agents, which are called symbolic constructor agents (SCA). They represent the perception portion of the composite agent and sense the shared environment (E_{outer}) they are situated in. The perceived sensory stream is filtered by the symbolic constructor agent for specific aspects or impressions that are important for the composite agent. Therefore unnecessary information is filtered out right at the beginning, which is beneficial for the efficiency of the whole agent. The remaining sensory input, containing the important aspects and impressions is then interpreted and translated into a symbolic inner environment (E_{inner}). This symbolic representation of the perceived world is tailored and optimized for each

specific agent. The symbolic inner environment depends not only on what the agent senses but often on its personality and current inner state. Hiles gives an example for that in a predator-prey context. If the predator is hungry and senses an animal, it would show up in E_{inner} as food. On the other hand, if the predator has just eaten, then the animal would appear in E_{inner} as just another animal.

The symbolic inner environment is the agent's perception of the shared outer environment within which it operates. E_{inner} often has little resemblance to the actual outer environment. It is an encoding of E_{outer} that is optimized to suit the composite agent's specific functions. The role of an SCA is like the role of a navigation aid used by a pilot. The navigation aid senses radio signals from the outer environment and converts them into directional information that the pilot can use to navigate the aircraft. The inner environment used by the pilot for making his decisions has little resemblance to the view looking out the window, but is optimized for the pilot's needs [Hiles 2001].

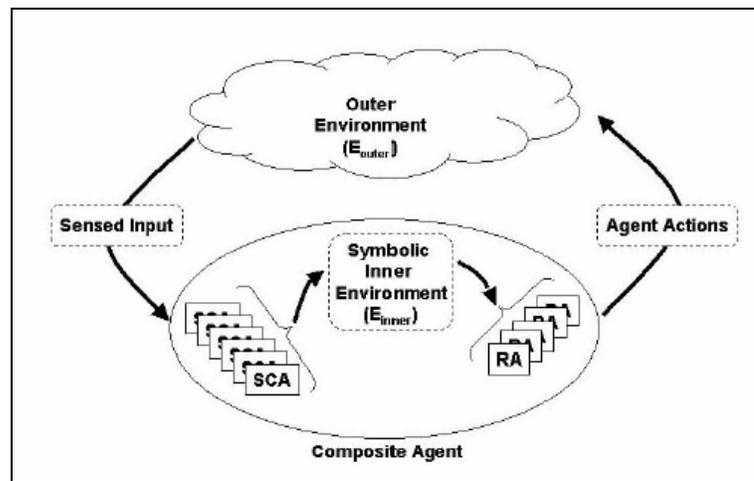


Figure 3. Composite Agent Architecture

A composite agent contains numerous reactive agents (RA), which represent the agent's effectory capability. The reactive agents make the decisions for their next action based on the state of the symbolic inner environment, although the actions they perform will affect the shared outer environment. The RA's are interacting either with other agents or with objects in the outer environment. Each reactive agent is responsible for promoting a specific behavior. So the set of all reactive agents taken as a group define the

composite agents' high-level behavior. Each RA has a set of one or more goals which are important for its specific behavior. The composite agent has the combined set of all goals competing for its attention. Just like humans have multiple goals (sometimes conflicting) an agent too has multiple goals it wishes to satisfy, and therefore the agent needs a mechanism to prioritize and de-conflict its goals. Reactive agents make use of a variable goal apparatus which attempts to mimic human goal management and decision making. Goals are prioritized by the agents inner state and personality and its perception of the symbolic inner environment (the current context). It is from this goal apparatus where contextually appropriate, intelligent behavior emerges. This behavior results in actions which try to satisfy the agent's highest-priority goals.

3. VanPutte's Maria Agent Model

The extended model of the composite agent architecture is called Maria [VANPUTTE]. In Maria there are two subclasses of composite agents: actors – which represent people – and infrastructures – which represent an organization's information resource and processing capabilities.

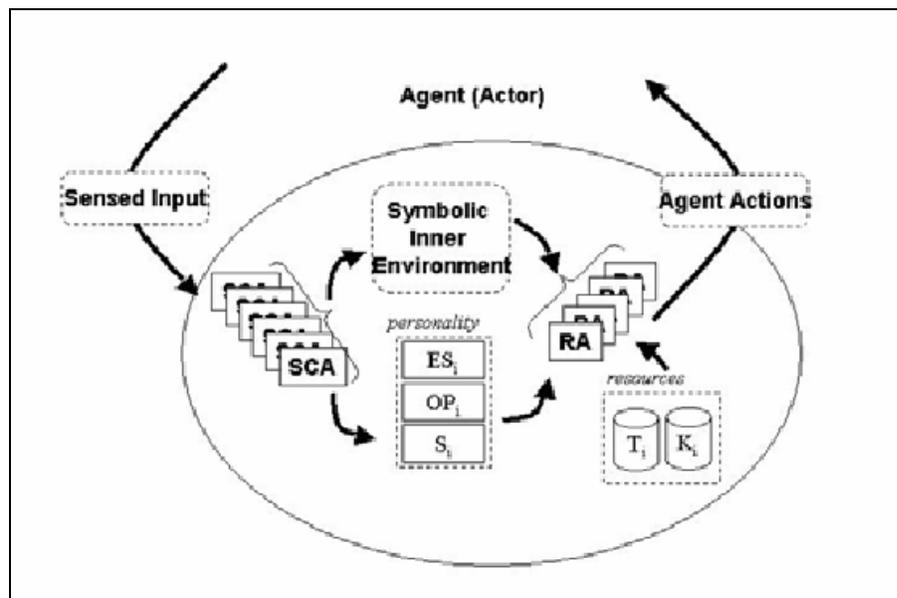


Figure 4. Maria Agent Architecture

As in the basic composite agent structure, there is a set of symbolic constructor agents (SCA), which sense and interpret the outer environment and create a symbolic inner environment. Additionally there is a set of reactive agents (RA), which use a goal apparatus to translate the symbolic inner environment into contextually appropriate actions to pursue their most relevant goals. The set of reactive agent is called the role set, because it is implemented such that each RA represents a specific social role. Therefore the role set is the set of all organizational roles to which an agent has committed.

In addition, a Maria agent contains two data structures: a resource set and a personality state. The resource set has two components. The first component is the knowledge set \mathbf{K}_i , which represents the actor's procedural problem solving capabilities. This knowledge base provides the actor with procedures to be used to achieve the various goals. The second resource component is the token container \mathbf{T}_i which holds the collection of all objects the actor has collected.

The personality state has three components: the emotional state, the observable personality and the skill state. As a whole these states determine the actor's commitment and dedication for each goal. The goal apparatus in the actor's set of reactive agents will use this information for prioritizing goals. This design creates outwardly observable differences in actor behavior. The first component, the emotional state \mathbf{ES}_i , contains a set of attributes that represent the actor's current internal condition or feelings. The emotional state can change rather quickly over time. It tracks feelings like loneliness, security and excitement. The second component, the observable personality \mathbf{OP}_i , defines the actor's long term behavior and represents a relatively static set. This set includes propensities for risk, loyalty to organizations, ethics, etc. The third personality component, the skill set \mathbf{S}_i , is an abstract set of the actor's abilities. Skills can be in different areas, like technical, social, informational, security or management skills [VANPUTTE].

4. Ferber's Multiagent System Equation

Ferber defines a multi-agent system (MAS) as a set of interacting elements. He describes it with the following equation:

$$\text{MAS} = \{ \text{E, O, A, R, Ops, Laws} \}$$

E:	Environment
O:	Objects situated in the environment
A:	Agents (intelligent objects)
R:	Relations linking to objects
Ops:	Operations
Laws:	Constraints governing the environment

Figure 5. Ferber's multi-agent systems equation

This equation splits the complexity of a multi-agent system into six components which can be handled separately. This approach is especially helpful in the analysis and design phase of building a multi-agent system.

The environment contains the system or the set of systems that will be investigated or modeled. In a multi-agent simulation the environment has the level of detail the modeler believes to be sufficient. All unnecessary detail has already been filtered out, to increase the efficiency of the model.

Wooldridge lists several properties to categorize environments. For example environments can be termed accessible, where "the agent can obtain complete, accurate, up-to-date information about the environment's state". Most complex environments like the physical world are inaccessible, based on the fact that it is impossible to obtain complete information regarding the physical world. Environments are said to be deterministic, when each action performed by an agents has always the desired effect guaranteed. However, most environments are non-deterministic, so the effect of an action is uncertain or probabilistic. Environments can be static, where only the agent's actions lead to a change, or dynamic, where the environment is subject to constant changes. Environments can be discrete, with a fixed and finite problem space, or continuous.

Objects are all passive entities in the environment. They can be moved, modified or destroyed by agents. They can be resources, obstacles or equipment. They are capable of predefined actions, but there is no goal or intent behind that action. They only do what they are told to do. Usually there are a lot of objects in a real environment, but we use only the really necessary objects in the multi-agent system. All background objects or scenery needs to be filtered out.

Agents are the actors in the multi-agent system. They have intent and can act autonomously.

Relationships can be between agents and other agents, agents and objects or agents and the environment. They can be cooperative or competitive, temporary or constant. Each relationship requires an actor to assume a role, which may be different for the various relationships an actor forms.

Operations are the elementary actions an agent can perform. They can be distributed throughout the whole system or have just a local effect. They can be constant or changing over time.

Laws are defined between the environment and agents, between objects and agents or inside an agent. Laws are the physical limitations of agents or objects, the boundary of the system or simply the defined rules for the whole system.

THIS PAGE INTENTIONALLY LEFT BLANK

III. THE ARIEL MODEL

A. GENERAL MODEL

The first step to approach the thesis was to do an analysis of the problem domain and its inherent mechanics, boundaries and specifications. The structure of the analysis is based on the multi-agent system equation by Ferber, as seen in Figure 5. The goal was to filter out all unnecessary information to focus on the important parts in the system as well as to identify the intelligent actors, which are able to act independently.

The environment (E_{outer}) is a complex and fully detailed road network, with highways, interstates, crossroads, small roads, bridges, tunnels, parking lots, towns and cities. An important requirement is to determine what level of detail is sufficient for the model. Certainly it is not necessary to have a representation of all road types and sizes and there is no need to model things like bridges, tunnels, since they are just special kinds of roads. Our focus is on road entities and their associated capacities. This capacity value takes care of the differences in road size and is related to the nominal speed that is possible to achieve on that road. The intersections between roads are represented by network nodes. Other entities in the environment are simply omitted, since they are just background or scenery (like cities).

There are several different objects that can be identified in the outer environment like depots, customers, trucks, supplies and truck-drivers. Customers request the delivery of supplies. It can be a huge facility or just a platoon out in the field. Each type of customer requests specific amounts and types of supply which satisfy their current demands. In network theory they are called "sinks". Depots or "sources" provide supplies according to the amount they have in stock. We can treat sources and sinks as special kinds of nodes, where one has supplies and the other demands.

Trucks are the only moving parts in the environment, as opposed to the identified entities so far, which are assumed to be static in their location. The most useful properties to be associated with a truck in this context are maximum speed, the amount of supply it can carry and its current location.

The intelligent entities that are situated in the environment are the truck-drivers. They have intent and can act autonomously. Therefore they are modeled using agents. Each driver-agent has an associated truck that it controls and which is moved by the agent through the road-network. Drivers do not leave their truck and do not need to switch trucks, so they are assumed to be connected at all times. Usually a truck-driver pair will start at their home source.

Elementary operations of the driver-agents are basically moving the truck through the road-network, like move to the next node or back off. However, other elementary operations need to be performed, like receiving a delivery order, loading the truck with ordered supplies, read the map and at the destination source unloading the truck.

Elementary decision are necessary during the whole delivery process, like figuring out the best route to the destination source (based on the map) deciding which road to take next, deciding whether to wait at a blockages or to back off and, when the destination is reached and the delivery is completed, deciding which source to drive to next. The goal of the driver-agents is to combine their effort and knowledge to reduce the delivery costs, so the main focus is to cooperate, rather than to compete.

There are only basic relationships and laws in the system. Trucks can only move along roads, and they are constrained by the road capacity and orientation. The nodes, sources and sinks are the locations where the decisions are made. Once the driver is moving along a road it will continue until it reaches the next decision point, which is a node, source or sink.

B. MODEL INPUT SETTINGS

To further define the problem the following assumptions and specifications have been made. The system will be optimized to minimize "cost" for each individual agent. The function that calculates the current cost will use three input values: time, distance and risk. Risk thereby is influenced by the size of the used roads and the amount of stops that become necessary at blocked nodes.

The interface of the system allows for the definition of the network with all nodes, sources, sinks and roads, as well as the delivery orders for the trucks. Disruptions can be introduced at runtime. The entities that can be affected by those disruptions and sustain damage are either nodes (sources, sinks) or the truck itself. The damage is expressed in terms of a repair time for the nodes and the current state for the trucks. The output of the system is a 2D simulation which will be displayed continuously.

For this approach there is only one type of truck which delivers one type of supply from one source to one sink. There is no communication available for now and that means no radio communication between drivers and no telephone connection between nodes, sources or sinks.

There are two entities which are affected by disruptions to the system: Nodes and Trucks. Nodes can have one of three possible states: free to enter (repair time: $R_i = 0$), blocked ($R_i > 0$) or destroyed ($R_i = \infty$). Trucks can be either on track, delayed, destroyed or have their current destination destroyed.

Roads are assumed to be straight. If there are meandering roads in the network we either approximate the road with a straight road or split it up into several straight nodes. All roads have a transit time T_i which is also assumed to be constant over the entire road. The time depends on the size of the road (nominal speed), the amount of damage to the road (condition), the number of trucks currently on the road (traffic). Those three values are combined into one value that represents the capacity of the road. The transit time would also depend on the maximum speed of the current truck, although the one type of truck which is defined in the system always reach the nominal speed of the road. Therefore the maximum speed of the truck becomes important once we introduce different type of trucks. Roads do not get blocked through disruptions in this model,

although it is possible to set the capacity of a road to zero which could represent a disrupted or destroyed road.

Decisions are not made continuously. There are only two situations where decisions are made. The first is after finishing the current arc before reaching the next node. Here the truck driver checks if the node to be entered is blocked or destroyed. He must then decide whether to back off or to wait. In the latter case he would then enter the queue for that node. The second situation is at the current node, before entering the next arc. Here the current route is questioned and, depending on the internal map and the attained local information including the status of all the possible arcs leaving the current node, a new routing decision has to be made.

C. MODEL APPROACH

1. Layered Design

The overall structure of the ARIEL model consists of multiple layers. The first layer is the network layer (see Figure 6 below).

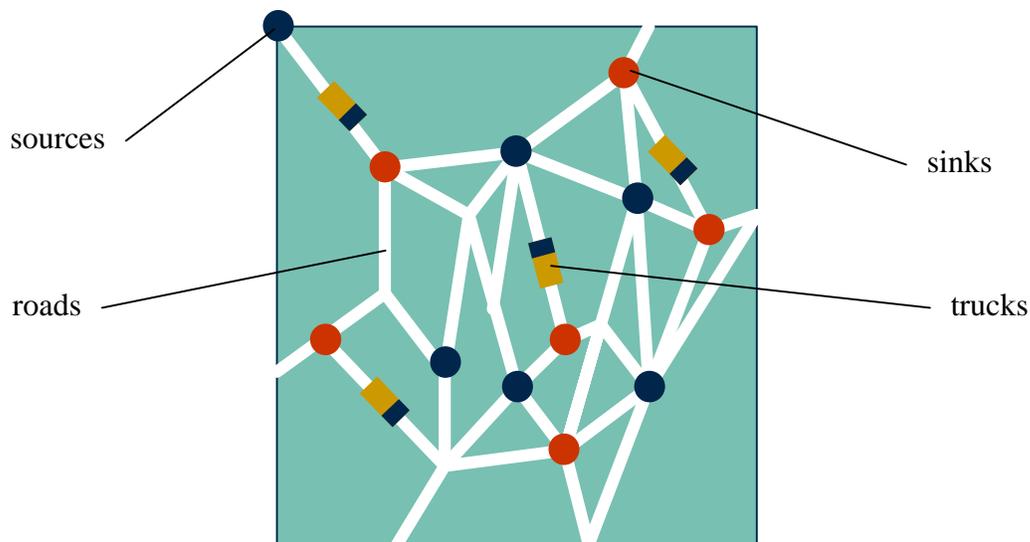


Figure 6. Network Layer

It contains all the entities in the environment, which are building blocks of the network and which location is rather static. These are the roads and nodes as well as the two higher level nodes: the sources and sinks. In addition, the network layer contains the trucks, which are the only moveable objects in the system. In essence they provide functionality and cannot act themselves. The layer on top is the delivery layer. It contains the autonomously acting truck-driver agents that are controlling the movements of the trucks in the system. The third layer is the disruption layer, which contains agents that are able to automatically introduce disruptions to the network, and force the truck-driver agents to adapt and choose alternative routes through the network. For a view of the entire layer structure of the simulation see Figure 7 below.

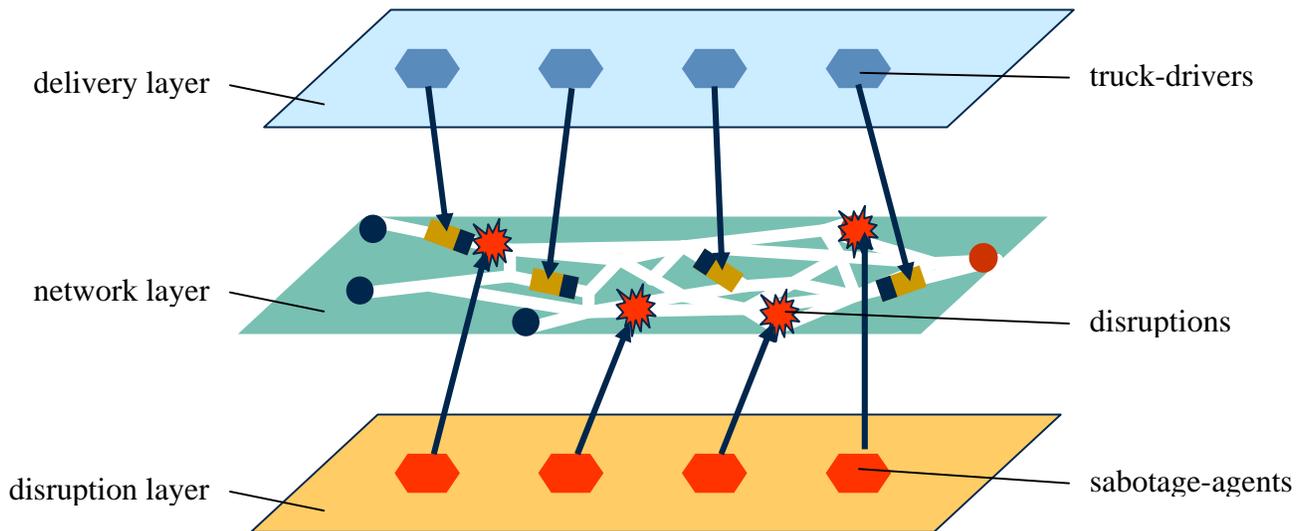


Figure 7. Layered Design

2. Information Flow and Global Knowledge

The truck-driver agents have no global knowledge about the current network state. They are equipped with a map of the network, but this map will not be up-to-date at all times. As the agents traverse the network they sense their local environment within a predefined sensing radius. Therefore they may encounter differences between the map and the actual environment. At each node they update their internal map and attach a

timestamp at each new piece of information. That way they are able to decide later which information is more recent.

For this research no communication was allowed between agents. In order to have some ability to spread the news about blockages, agents are able to deposit their attained knowledge about the network to nodes. Essentially, nodes keep their own internal map and agents are able to sync their internal map with the ones of the nodes. Therefore, agents have two ways of knowing about blockages. They are either direct encounters, or when they happen to drive through a node where this information was previously deposited by another agent.

The sensing radius of the truck-drivers was kept reasonably small, similar to real truck drivers. When a driver is in front of a node (see Figure 8, left side), he can assess the capacity of the current road and the repair time of the next node, but the capacities of the leaving roads on the other side is not visible at that point. If the driver is at a node (see Figure 8, right side), he can oversee all roads that leave this node and is able to assess their capacities. However, the repair times of nodes at the other end of these roads are not visible.

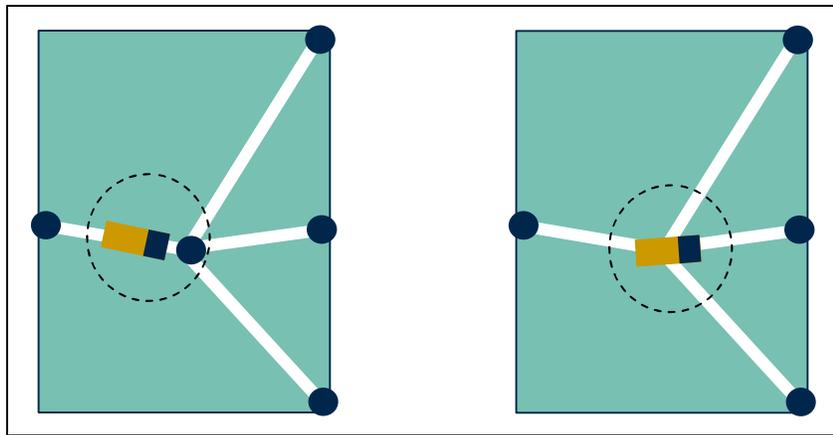


Figure 8. Sensor radius of the truck-driver agents

3. Inner Environment and Map Reading

Agents sense their (outer) environment and gain information about it. But the completeness of information that exists in a real environment needs to be filtered to the level of detail that is sufficient for the agent's decision making process. In addition they often need this information in a slightly different format, to be able to operate on it. So

they convert the sensory stream of the outer environment into a symbolic representation – the inner environment.

In the ARIEL system the truck-driver agents are situated in a road network. Before they move through the network they are given a road map, which displays the network's state at some point in the past. It shows what roads are connected to which nodes and contains all roads and their capacities, and all nodes and their repair times. The same map is given to each truck-driver as the initial information common to all truck-drivers. This map is encapsulated by each truck-driver and is constantly being updated, while the truck-driver traverses the network and gets to encounter the current state of the network. This internal map is then different for each agent, due to its direct encounter and through map syncing at sources or sinks.

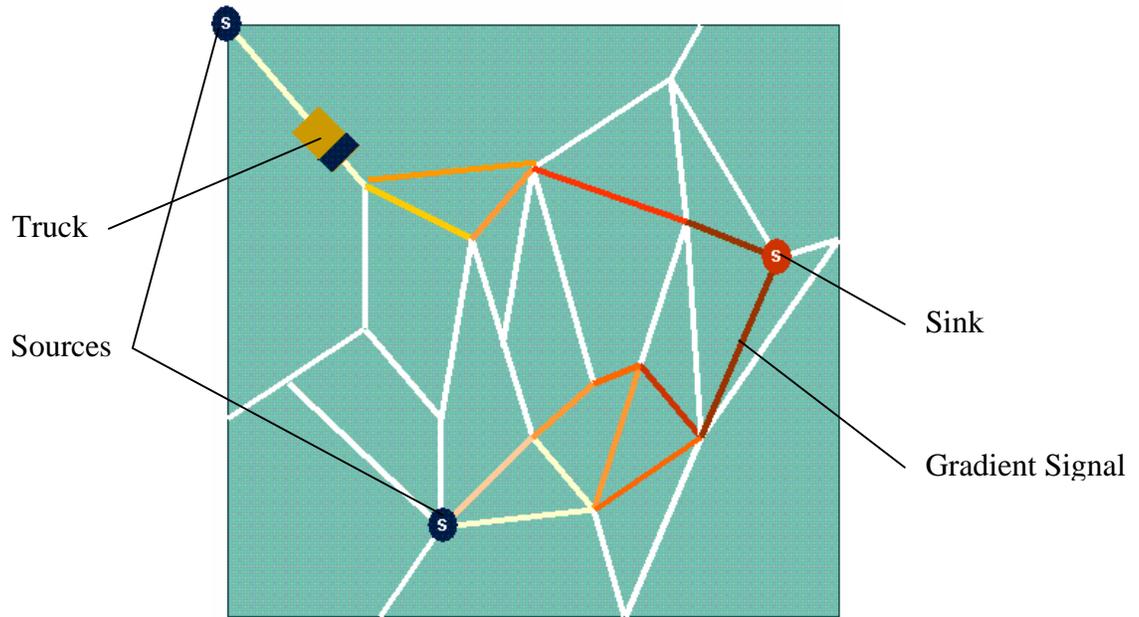


Figure 9. Gradient Encoding

However, agents need information in a more numeric format to be able to efficiently find a route through the given network. The chosen approach was introduced by Hiles [HILES2] and is based on an imaginary radio transmission sent by the agent's destination sink (see Figure 9). This signal traverses the network and is modified each time it passes a node. At the same time, it is being stored at each node, such that an agent that arrives at a node is able to sense the transmission signal. The information transmitted

with this signal has four dimensions. The first two are: time and distance, which the agent probably needs to get from the current node to its destination sink. Additionally, the transmission contains a risk factor and an estimate for the cost the agent will consume to reach its destination sink.

These values define a gradient distribution of the entire network, based on the current destination sink. Therefore, wherever the agent is located within the network, he can sense the gradient. From each node, the gradient values will tell him which would be the best road to take. This approach is the agent's way to read a map. It is similar to a real truck-driver, who looks at a map and figures out which route to take, although this method is more precise. The separate transmission of all four dimensions allows the agent to react differently according to the current context and his internal states. This means that if he is running out of time for the delivery, he can weight the time dimension more than the other dimensions, or if he is running out of fuel, he probably would have a greater propensity towards the distance dimension. Each agent could have different attitudes toward risk, like a brave driver or a more frightened driver.

Therefore the gradient approach allows much flexibility for the agent to express and make use of his unique personality. At the same time it is a simple but effective way for the agent to find its route through the road network.

4. Model Flow

Before the simulation starts, the base map (M_0) of the network is being published. Every node, source and sink gets the same copy, which is stored as their internal map. When the simulation is about to begin, the agent is usually located at one of the sources – his home source – ready to pick up supply and process a delivery order.

At that time he will execute his first syncing with the current source. That way he will get the base map (M_0) and gets the initial information about the network to proceed. As displayed in Figure 10, when a driver gets his first delivery order, he will encode his gradient distribution based on the stored map information and his propensities for time, distance and risk.

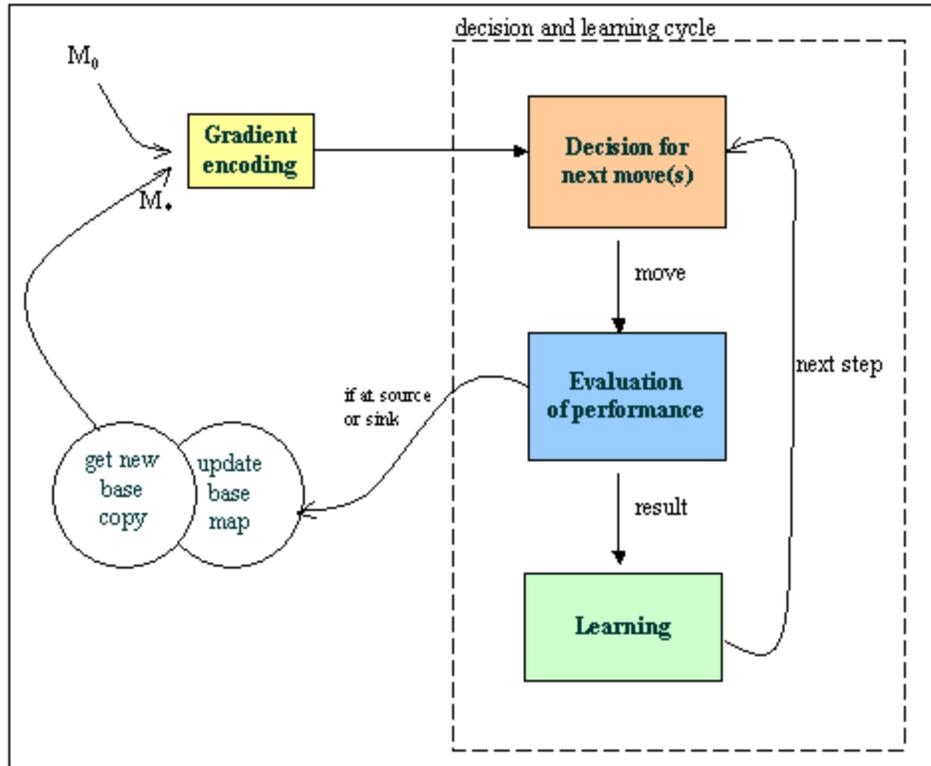


Figure 10. Model Flow

Now the truck-driver is equipped with all necessary information and can start his mission by entering his decision and learning cycle. This cycle will run continuously until he reaches his destination sink.

The first step in the decision and learning cycle is to decide which of the current node's leaving roads are considered to be optimal to reach the destination sink. This decision will be based on the agent's internal map, the local information of current repair times and capacities, which will be added to the internal map, as well as the individual preferences of the truck-driver agent. When the decision is made the agent moves its truck to the next node.

When all agents are done with their move their performance is evaluated. This can be set to occur either after each move or after a defined number of moves. The evaluation function is often called the objective function, as it separates good from bad performers. The calculation is based on the agent main metric, which is the cost.

After a defined number of cycles the agent can make a learning step. Here the agent compares his performance to the performance of the other agents and then gradually adapt some of his current preferences towards the preferences of the top performers. This adaptation is intended to result in a better performance in subsequent steps.

This cycle is repeated until the truck-driver reaches its destination sink. Whenever the truck-driver passes a source or sink on its way, he will deposit all his gained information about the network, and get some new information if available. Therefore if the agent needs to recalculate his gradient distribution, it would be based on the updated internal map of the agent.

D. MODEL DETAILS

1. Basic entities

The basic model entities are described in detail below, and are shown in Figure 11. The road network is built using two types of entities: roads and nodes. They form the network layer which is assumed to be static in its structure, although the internal variables like capacities or repair times might change during the simulation.

Nodes are basically a point with a given location. The important internal variable is the repair time, which is initially set to zero. When it is set during simulation it represents the approximate time the repair will take. To define the structure of the network, each node stores all leaving roads from that node (fan).

To enable the agents to deposit their information about the network, the node needs to provide the same structure to store its own internal map. This has been done by two variables. One stores the leaving roads for each of the nodes of the network (fanMap); the other stores the capacity for each road and the repair time for each node (map).

Roads are basically a line with a starting point and an end point. In this case starting and end points are nodes; therefore roads store a reference to the starting and

ending node in their internal data structure. The most important variable is the capacity, which refers to the surface quality of the current road. The capacity defines the maximum speed a vehicle can make on the road (nominal speed).

The second part of the network layer, as seen in Figure 11, consists of all objects which are moving or could be relocated, but do not need much intelligent behavior, like trucks, sources and sinks. In a usual logistics scenario sources and sinks might be considered as static, but in combat scenarios it is not surprising if both the provider and the receiver need to be highly mobile and adapt their location to where they are needed.

Sources and sinks are basically nodes. At the current state of the project they do not incorporate much more data structures or methods. But they have been defined to allow a quick extension by a source and/or sink control system.

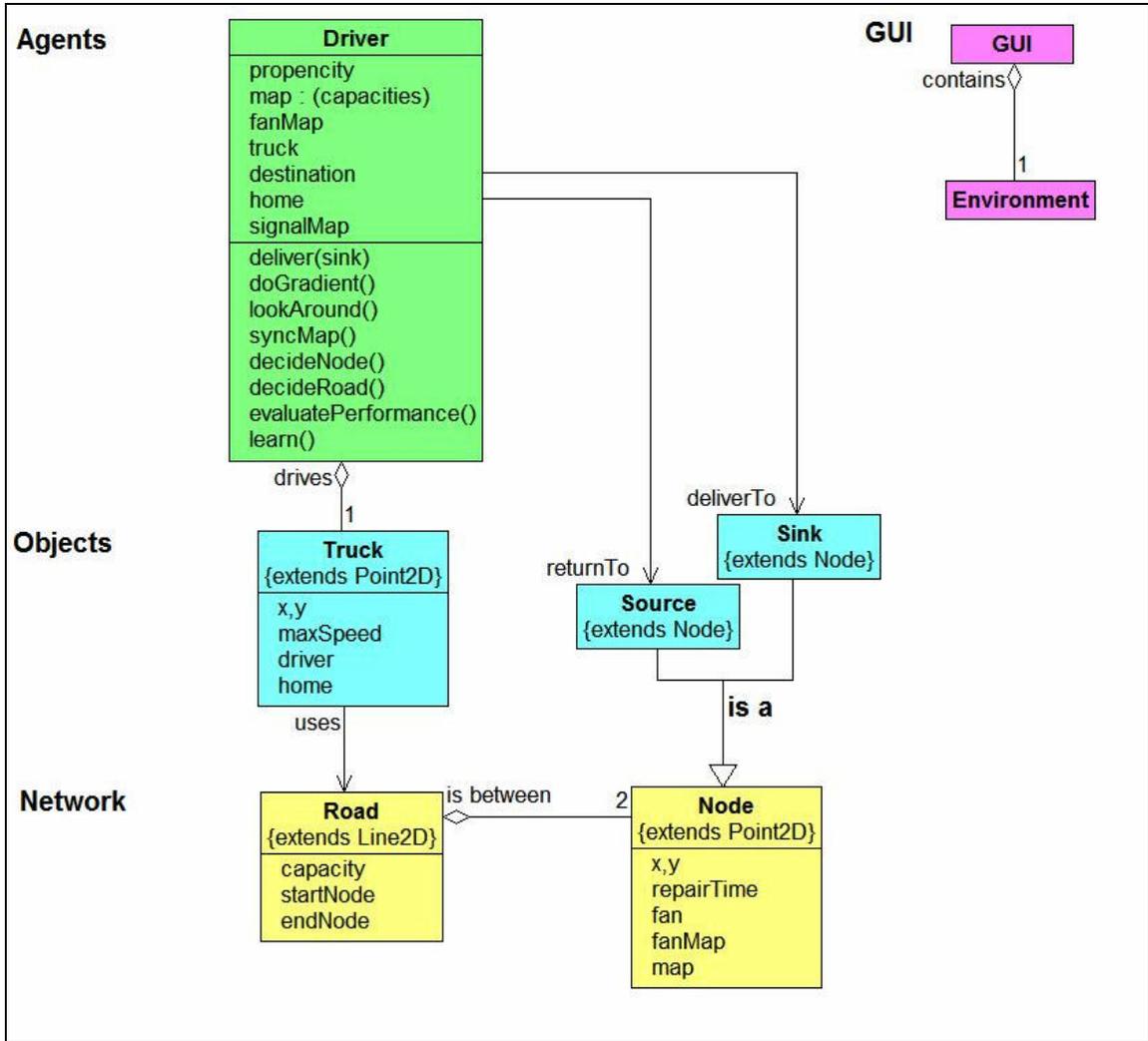


Figure 11. Basic Entities

Trucks are basically points with a current location. They have one defining variable which is the maximum speed (maxSpeed). The resulting speed that a truck can actually make can be computed, by taking the minimum of the road's nominal speed (capacity) and the truck's maximum speed. In addition the truck stores a reference to its driver agent and to its home source.

The agent layer on top contains the driver agents. This is the layer where the intelligent behavior is located, where decisions are made and evaluated and where the learning takes place.

Drivers have as their defining variable the set of propensities. Then they have an internal map of the road network. The internal map consists of two parts, the fan map which contains the leaving roads for each node in the network and the map which contains the capacity for each road and the repair time for each node. Both parts are constantly updated while traversing the road network. In addition the driver stores a reference to his home source, his destination and the assigned truck he is using. From the information in the internal map and his propensities the driver calculates the gradient distribution before he gets going. This gradient information is stored in the signal map.

The driver's major methods are indicated in the bottom part of the “Driver” box in Figure 11. Hereby the *deliver* function is the only method that is called from outside, while all the other methods are called within the agent. *DoGradient* generates the gradient distribution, *lookAround* takes the local view of the environment and updates the internal map with the actual values, *syncMap* performs the two-way update between a driver and a source or sink. *DecideNode* and *DecideRoad* generates the next action take by the driver and *evaluatePerformance* and *learn* let the agent rank itself among the drivers and learn from the best.

The GUI (Graphical user interface) portion of the model consists of two basic classes. The Environment class shows the real-time 2D simulation of the road network with the truck-driver agents moving through it. The GUI class puts a frame around the simulation portion. In addition, it allows for the design of the network by entering all necessary entities and their defining properties.

2. Destruction layer Entities

As seen in Figure 12, the author implemented three different methods to introduce disruptions on the road network. First, it is possible to change capacities for a specific road or repair times of a node through direct manipulation. That way the system user can set up a specific network situation for his study and can change the network at runtime. The next two methods manipulate the repair times of multiple nodes at once. However, road capacities remain unchanged in the current implementation.

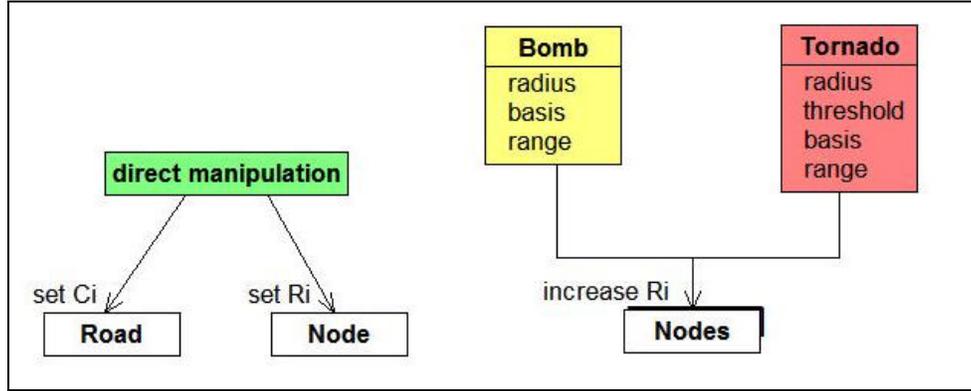


Figure 12. Disruption layer entities

The second method is the bomb tool. The bomb modifies the repair times of a node that lie within a certain range from the bomb's center of impact. Before starting the simulation, the user can specify the radius for the effective range, and two additional parameters: the basis and the range. The basis value defines the repair time portion R_{basis} , which is added to each node that lies within the specified radius. The range is used as the upper boundary R_{range} , to generate random repair time portions. The random values are uniformly distributed between 0 and 1.

The application of the bomb tool results in a recalculation of the repair times of each node which lies within the effective radius and is modified as follows:

$$R_{new} = R_{old} + \left(R_{basis} + (R_{range} * random) \right)$$

Equation 1. Bomb tool equation

The third method of destruction is the tornado tool. The name of a whirlwind was chosen in reference to its capability to move over land and introduce disruptions in an undeterministic way. Speed and heading are determined randomly within a certain predefined range. Each time the tornado reaches the border of the road-network it changes the speed and determines a new heading. The modification of the repair time of

each node within the radius is basically the same as for the bomb tool. But as we see in Figure 12, a new parameter has been added for the tornado: the threshold (T). This parameter allows to specify how likely it is for a node to be modified by the tornado. Usually all nodes within the radius are modified, but there is an additional random value (m_i) generated for each of those nodes and only those nodes with values below the threshold value are actually modified by the tornado.

if ($m_i < T$) **then**

$$R_{\text{new}} = R_{\text{old}} + \left(R_{\text{basis}} + (R_{\text{range}} * \text{random}) \right)$$

Equation 2. Tornado tool equation

The tornado is a sabotage agent (compare Figure 7) that is constantly acting in a non-deterministic way and makes random based changes to the network nodes.

3. Gradient Encoding

As seen in Figure 9, the driver agent starts his encoding process at the destination sink. Moving backwards through the entire network, he calculates the signal for each node, based on the current destination, and stores it into his internal map. The process stops as soon as each node has been processed. During the entire process the agent uses only his internal map for information retrieval and to store the gradient and other values. He cannot access the network directly, except the local area around his current location.

Figure 13 shows the three steps of the encoding process. During the preparation step, the agent gets each node from the collection of all current nodes and marks each of them as “not queued”. The initialization step, at first, creates a queue (nodeQueue). The nodeQueue contains nodes, where the gradient information is not yet completely calculated and stored. The destination is added to the nodeQueue and is marked “queued”.

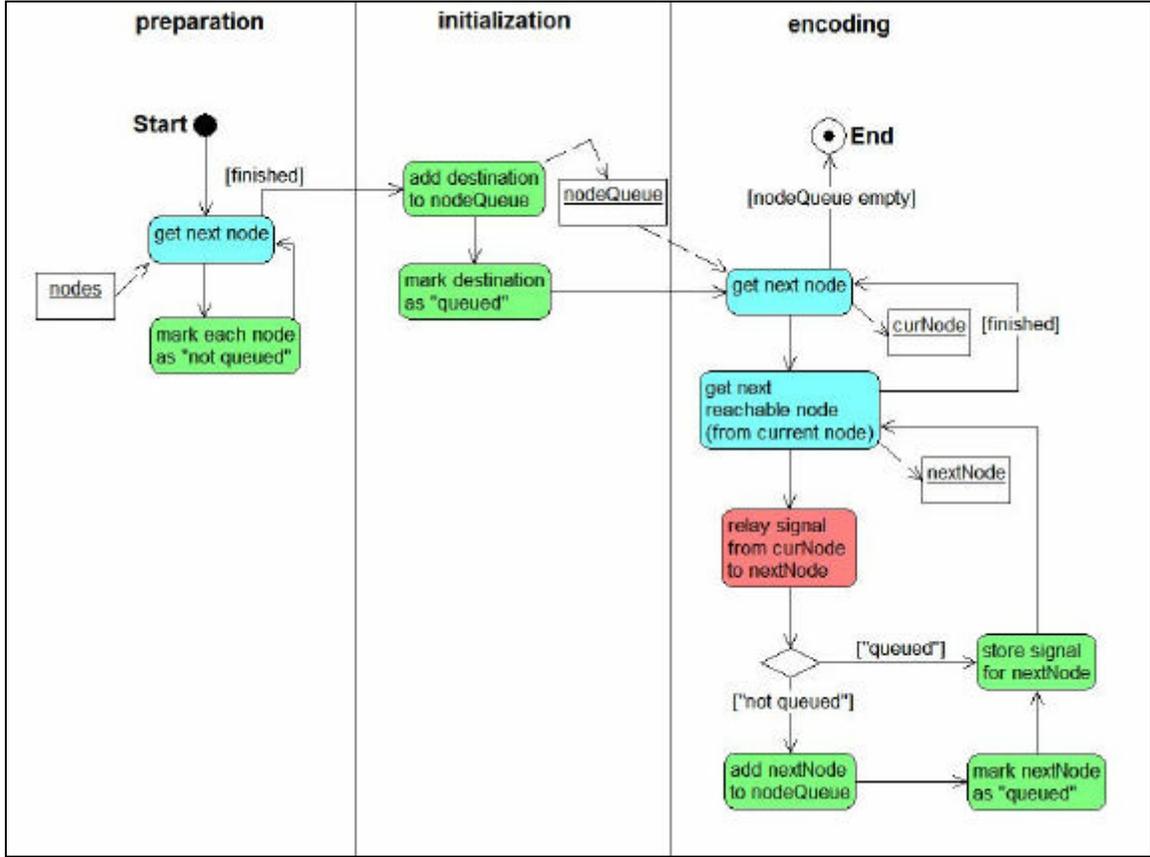


Figure 13. Detailed Encoding Process

Finally the encoding step calculates the gradient values for each node. The next Node is taken out of the nodeQueue and stored as curNode. Then the set of all leaving roads (fan), which the agent knows for each node, is used to get the collection of all reachable nodes from curNode.

Each reachable node is processed separately and stored temporarily as nextNode. The signal portion from curNode to nextNode is calculated based on the distance between the nodes, the capacity of the road, risk factors and the agent’s propensities. If nextNode has been queued before (marked as “queued”) the new signal information is stored into the gradient information for nextNode. If nextNode was not queued before, some additional steps are necessary. It will then be added to the nodeQueue and marked as “queued”. The container for the gradient information (signalMap) is reset for nextNode

and the calculated new signal information is added as the only entry. Then the next reachable node is processed. When there are no more nodes for the current node (“finished”), then the next node is taken out of the nodeQueue.

The process repeats itself until all nodes have been queued and processed. The resulting gradient distribution stores for each node of the network four dimensions: cost, distance, time and risk. These values are based on the current destination and display the distance, time and risk to get from the current node to the destination. The cost dimension is a composition of the other three dimensions, and it is based on the agent’s preferences for distance, time and risk. It is tailored that way to meet the specific needs for the current driver agent.

4. Propensities

a. Skepticism

The skepticism propensity represents the first of three ways the agent can express his personality and show his independence. When the agent reaches a blocked node, he requests information about the estimated repair time. The value he receives is not the value that he will use for his decision-making process, due to his specific level of skepticism. Figure 14 displays three personalities types and how the estimated repair times are affected in each of them.

The first personality adds a constant value A to a received repair time. The next believes that repair times estimates are almost correct, but the actual time varies a small amount from the received value. A random amount, based on the range given by B , is added to the repair time. The last personality relies on historical data. Instead of the repair time, the average of the last C repair times is taken as the new repair time.

The personality is defined through the set of values for A , B and C . A personality of type “A” for instance, has a nonzero value for A and zero for B and C . Personality combinations are also possible. In this case the average is taken among the repair time results of each personality type.

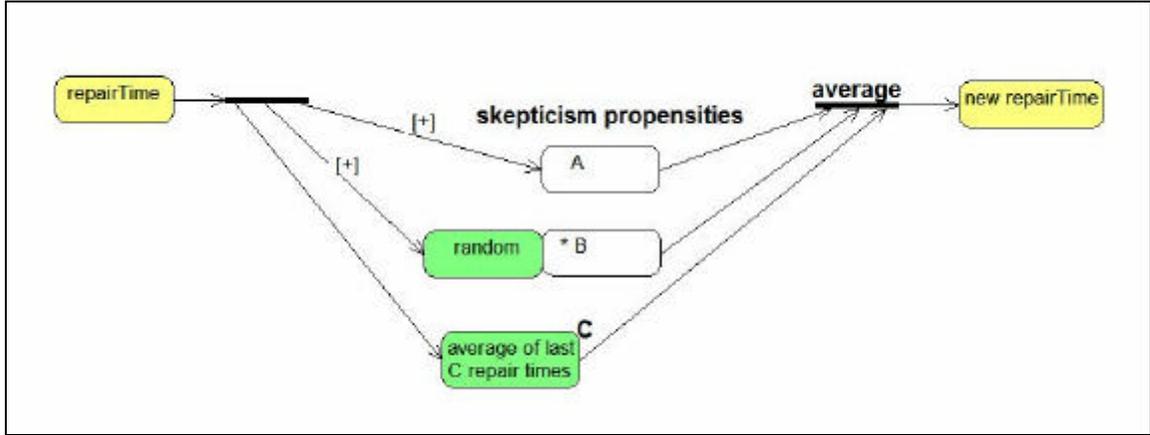


Figure 14. Skepticism

b. Propensity for Routing Decision

Decisions are not made continuously. Figure 15 shows the two situations in which the agent is triggered to make a decision for the next step.

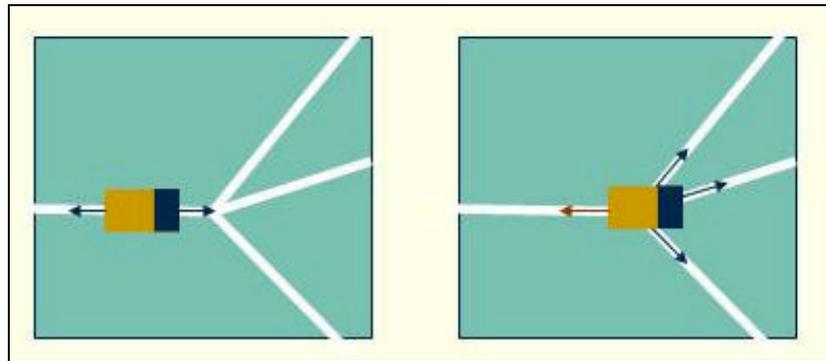


Figure 15. Points of decision-making

The first situation is after finishing the current arc, before the next node is reached (see Figure 16). Here the truck driver checks if the node to be entered is free, blocked or destroyed. When the node is free the agent proceeds immediately and enters the node. If the node is destroyed, waiting would not make any sense and the agent backs off right away. In the case of a blocked node the agent has two choices. One is to back off and try an alternative route; the other is to wait at the node until the blockage has been resolved. The decision is based on the expected repair time the agent has perceived (see

Skepticism), his propensity value for patience and the difference for cost between the current route(waiting) and all alternative routes(backing off).

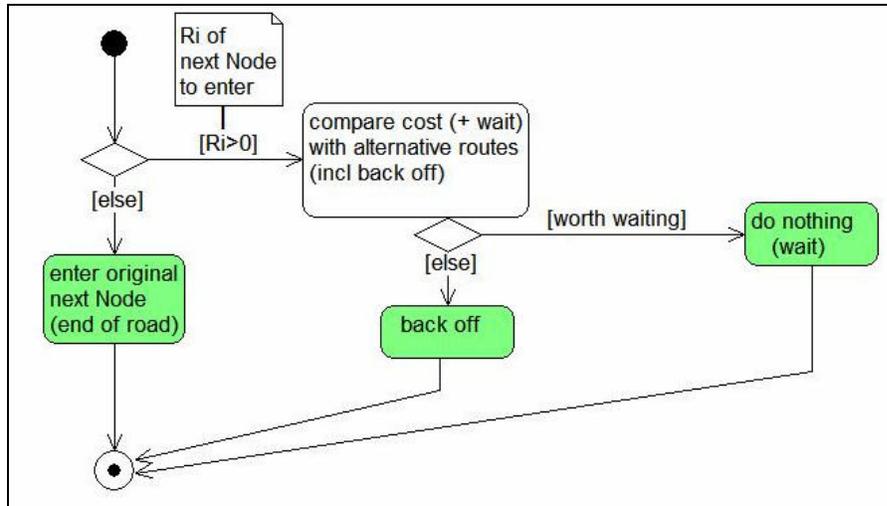


Figure 16. Decision after finishing current road (arc)

The other situation is after finishing the current node, before entering the next arc. Here the agent compares time, distance and risk of all leaving routes and takes the one that best suits his propensities for those dimensions (see Figure 17).

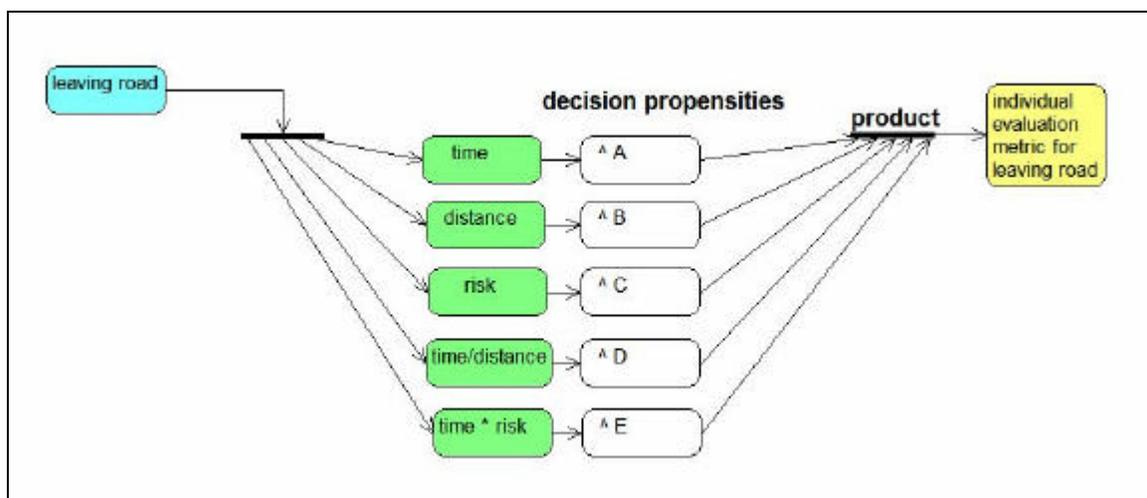


Figure 17. Decision after finishing the current node

The personalities are defined through the set of values A, B, C, D, E. For example and “B”-type agent takes only distance into account. The value for B would be nonzero and the other values for A, C, D, E would be zero. Usually the agents have personality which combines one or more of those types.

The resulting metric is based on his propensities for the different dimensions. Essentially the metric is the agent’s understanding of “cost”, which he wishes to optimize. He uses the comparison of this value to decide which of the leaving roads is best for him. The same calculation is made during the gradient encoding process and stored as one of the dimensions in the gradient information for each node.

c. Learning

The learning process is implemented in two ways in the model. First, there is knowledge based learning through the update of their internal map. This is done continuously by each driver agent and through the syncing process at sources/sinks when knowledge from other agents is taken over into his own internal map.

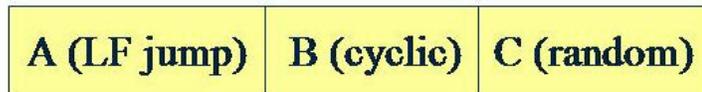


Figure 18. Learning Propensity

Secondly agents learn by adaptation. They get the deposited performance information, which was generated in the evaluation process, and compare their performance to those of other agents. If another agent has a better performance, the agent moves his propensities gradually toward the propensity of the better performer, in order to increase his own performance.

Essentially the agent decides himself whether and how often this learning step takes place. The personality is defined by the values A, B and C (see Figure 18). In this case there are no combinations between the types. The A-type agent learns as soon as he observes an unusual jump in his loss factors. Here the value of A defines a percentage

threshold for a jump of a loss factor. The B-type triggers the learning process at every Bth syncing procedure. Finally the C-type performs the learning process randomly. Here C defines a threshold, and learning takes place during the syncing, whenever a generated random value is lower than this threshold.

5. Performance Evaluation

The evaluation of the agent's performance is essential for his learning and adaptation process. The agent performs two steps: loss factor calculation and reporting.

The loss factors are the agent's measure of performance. They show the difference between his estimates for time, distance and risk to get to his destination and the actual values.

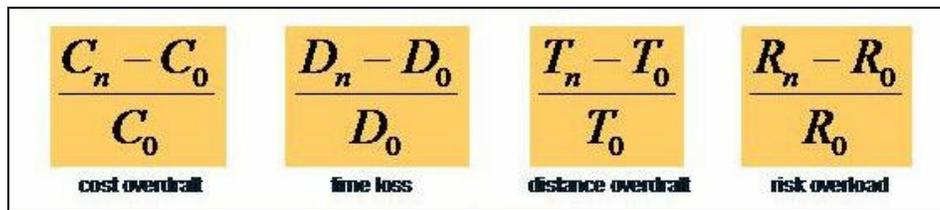


Figure 19. Loss Factors

The 0-subscript represents the estimate of the value, before the agent starts moving. While the agent moves through the network the n-subscript represents the actual value at time n. This allows a loss factor to be defined for each of the four dimensions (see Figure 19).

The second step in the evaluation process is the reporting. Here the agents attach their current loss factors to their propensities. They deposit this performance information at each source and sink during the usual sync process between driver agents and sources/sinks. That way they can compare the performance of other agents to their performance.

6. Combination of the elements

Figure 20 displays the combination of the described elements in a composite agent structure.

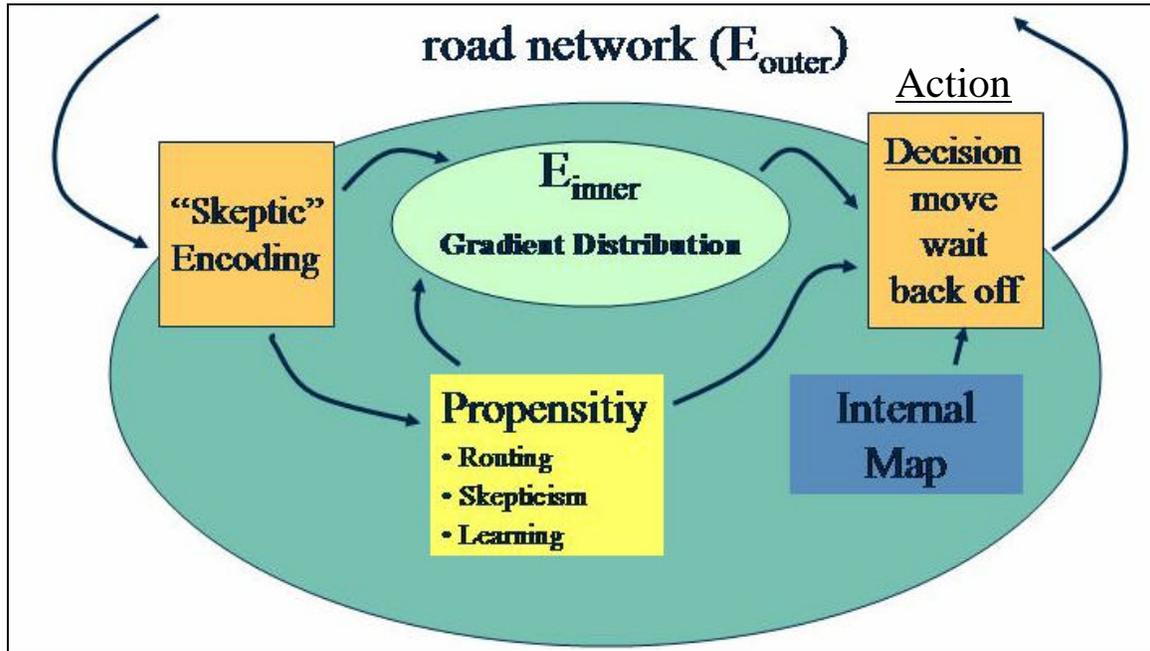


Figure 20. Composite ARIEL agent

The gradient encoding calculation represents the sensing portion of the agent. The encoding is called “skeptical”, since the received repair times are interpreted based on the agent’s skepticism propensities. The result of the encoding process is a gradient distribution. It represents the agent’s way to see and interpret the outer environment (road network) in which he is situated. The gradient is a symbolic representation (E_{inner}) of the outer environment. The encoding is specific for each agent and is based on his routing propensity. The action-portion of the agent uses gradient distribution, his routing propensity and his most current gained knowledge, to update his internal map in order to make his decision and to perform the appropriate actions.

As described in the previous chapters, the current model contains three propensity sets: decision, skepticism and learning. These sets can be viewed as genes of the agent. That way the model can easily be extended to genetic programming or evolutionary strategies.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. VALIDATION

The author used a fixed test network to evaluate the model. The road network was randomly generated and contains fifty nodes, six sources, six sinks, one hundred roads, six trucks and six driver-agents. At the beginning each driver-agent has been placed on one of the six available sources (see Figure 21, left side). Their task during each of the runs is to deliver the supply to one of the six sinks on the opposite side of the network (see Figure 21, right side). The topmost driver will deliver to the topmost sink and so on. Once they have reached their destination sink they will immediately return to their home source and then wait for the next order.

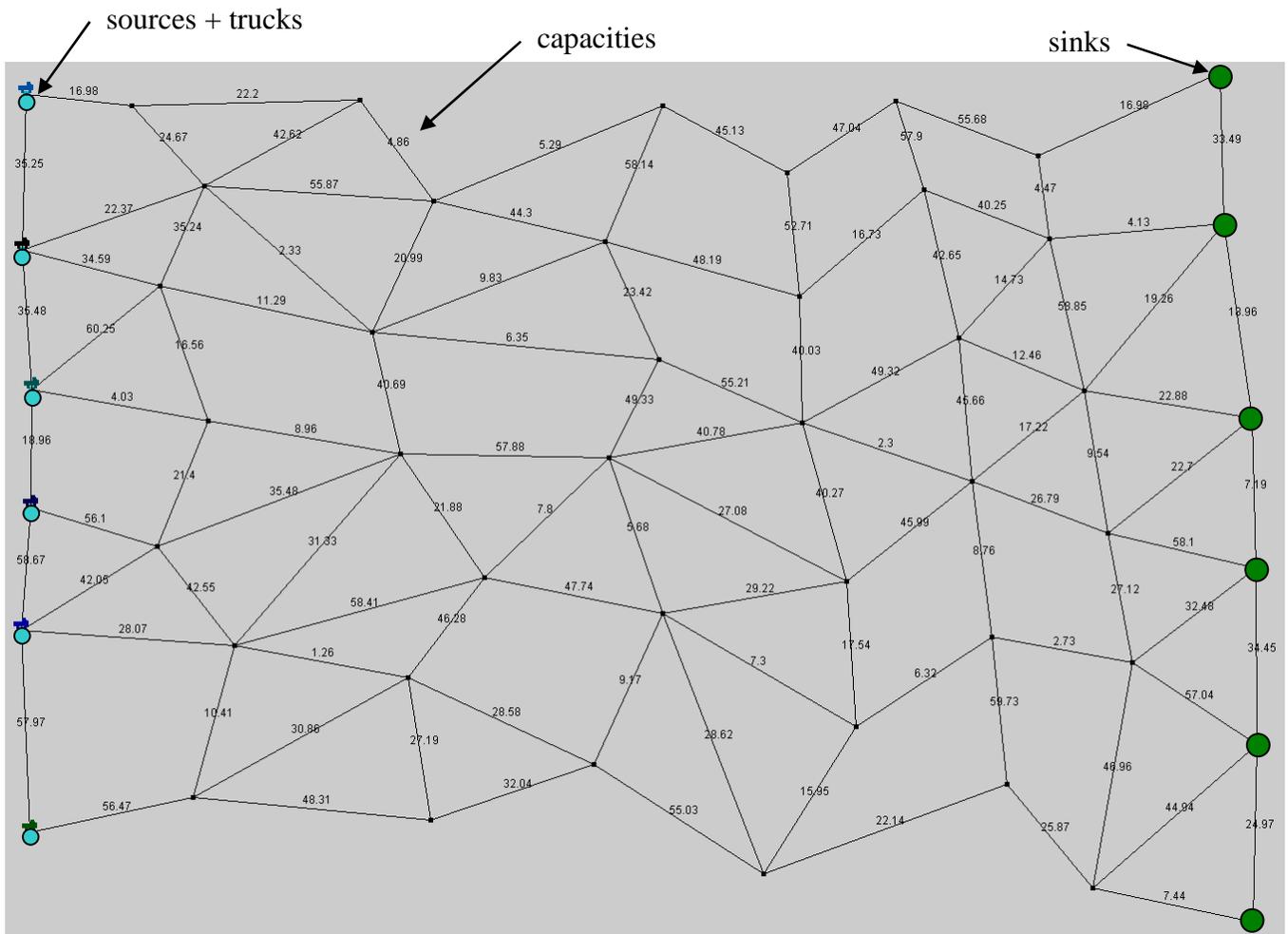


Figure 21. Test Network for Evaluation

The model will be validated through the method of face validation, as described by Law and Kelton [LAW] and in the Department of the Army pamphlet 5-11 [PAM]. This technical method is used as an initial step to show that the model seems reasonable to personnel who are knowledgeable about the subject area. The author chose two model parameters for that purpose: the agent's level of skepticism and the uncertainty of the network. Through variation of the input parameters, the relationship between those parameters was examined to determine if the model behaves as expected.

1. Experimental Setup

At the beginning the repair times of each node are zero. That means that no node is blocked, and therefore the driver-agents can enter and pass each of the network nodes. During the evaluation disruptions were introduced, which causes the repair times to be bigger than zero. In Figure 22 the trace for an optimal route is displayed. These optimal routes assume that no disruption occurred yet. As seen in Figure 22, the topmost driver-agent will pass node 15 (circled node) on its optimal route to its destination sink. During the experiments this node is being disrupted and as a result it will have a repair time of 2000.

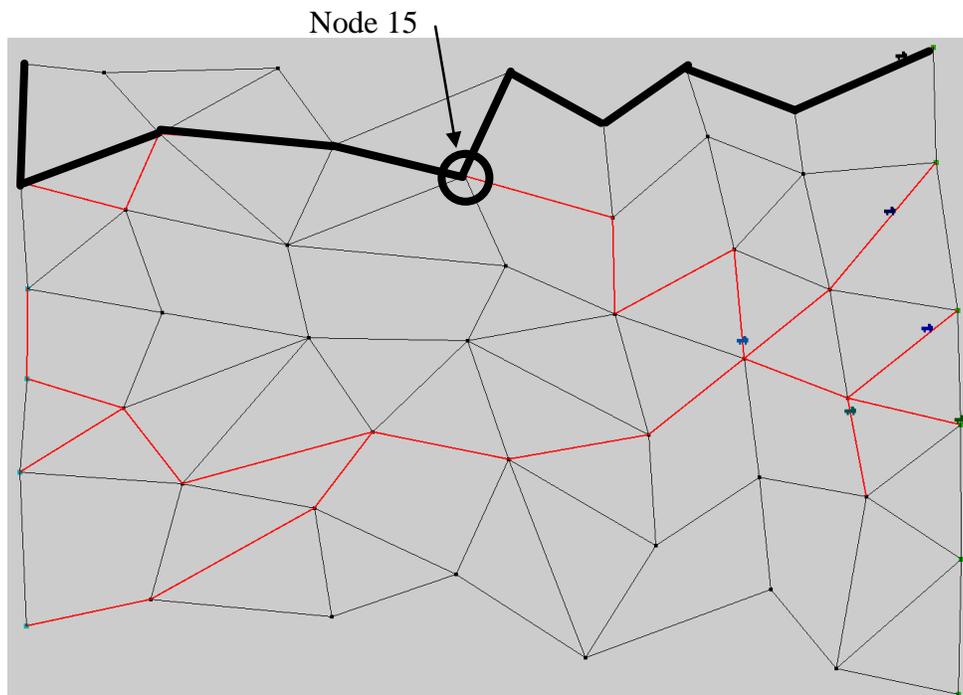


Figure 22. Trace of Best Route

The experiment investigated how well the driver-agents deal with this blocked node and how they improve their performance over time. The input parameters are: 1) the reliability/uncertainty of the repair times in the network, and 2) the skepticism of the driver-agents. The repair times are called certain if the repair time that the agent receives from the current node is the true repair time. The repair time is uncertain if the repair time received by the driver-agent is an estimate and will be randomly decreased over time. The skepticism propensity defines how a driver-agent interprets the given repair time. The author expects that there is a dependency between both parameters, which should result in a better performance for non-skeptic drivers on a network with certain repair times, while on the other hand there is a better performance for skeptic driver-agents on a network with uncertain repair times.

2. Experimental Runs

During the experiment the repair time of node 15 is set to 2000. The metric to evaluate the performance of the driver-agents is the overall cost. In this case the author chooses to have the overall cost to be the time the agents need for one round trip. Figure 23 shows the average outcomes of the first seven runs in an uncertain network. The seven runs represent seven round trips by driver from source to sink and portray his level of success depending on circumstances. The blocked node appears immediately in the simulation, prior to the first round trip.

The agent's behavior is based on the network state and their propensity. The usual pattern can be described the following way: In their first run the agents are surprised by the blockage and search for an alternative route. On the second run they know about the blockage and take the alternative route right away. The cost of 59146 shown in the second run is the optimal result for the alternative route. In the delivery phase of the third run they take the alternative route again, but in the return phase some agents tend to think that the blockage is already resolved. Therefore they run into the blockage again and store the new repair time. In the delivery-phase of the fourth run they take the alternative route again. The return phase is quite different and depends on the true repair time.

Usually the agents believe that the node is repaired and therefore would run into the blockage again. At the beginning of the fifth run the node is usually repaired.

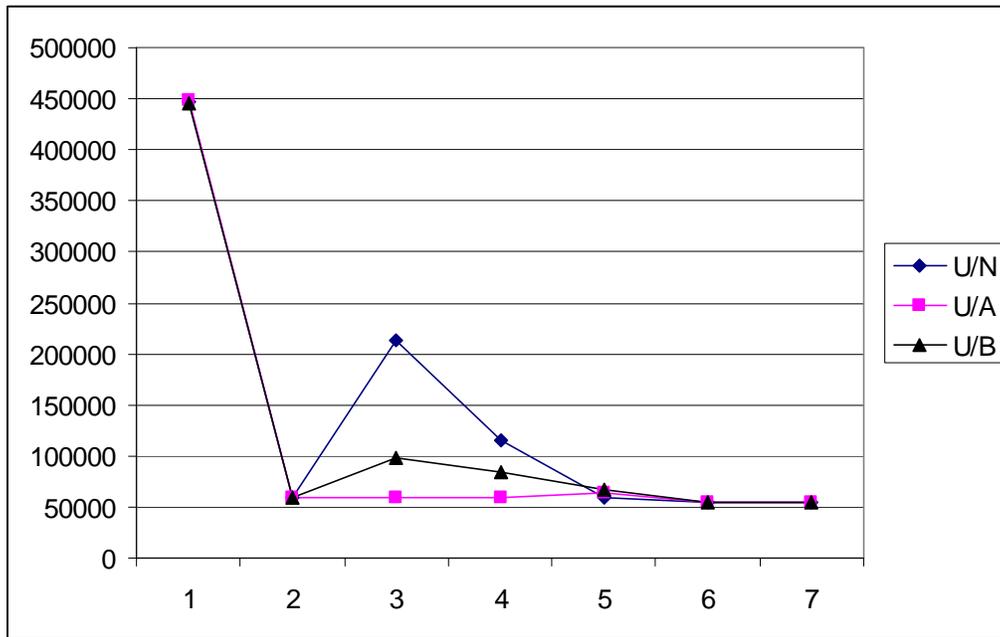


Figure 23. Results for $R_1=2000$, uncertain network

As seen in Figure 23, the performance of the first two runs are approximately the same but the difference is shown especially in the third and fourth run. Non-skeptical agents have a peak while skeptical agents have a lower cost. Obviously skeptical agents of type A are best suited for that environment, since their cost stays almost flat after the first run. This fits the assumption that skeptical driver-agents would perform better in a network with uncertain repair times.

The pattern for agents operating in a certain network environment is similar to the uncertain case in its basic structure. The main difference is the point in time when the agents think that the node is repaired compared to the true repair time of the node. The results of the average outcomes of the first six runs in a certain environment are shown in Figure 24.

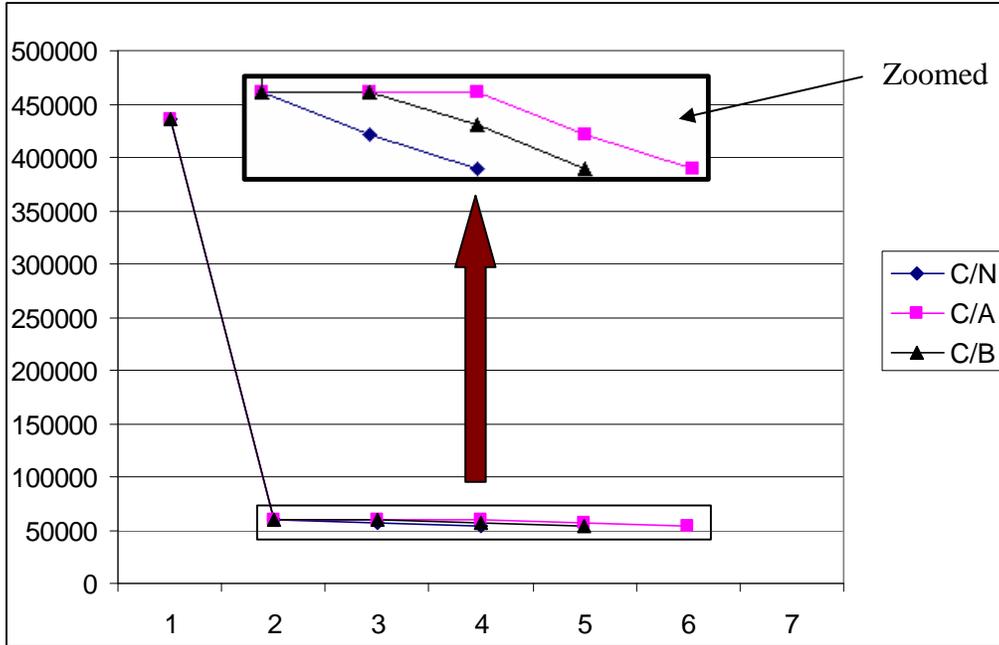


Figure 24. Results for $R_i=2000$, certain network

The outcomes in a network with certain repair times have much less variability than in the uncertain case. As seen in Figure 24, the cost of the first runs is almost the same, although we can see a small difference in the time it takes to get to the optimal cost. Here, the non-skeptical agents are down to the optimal cost at the fourth run (see zoomed area in Figure 24). The skeptical agents of type B need five runs and type A agents used up to six runs to get back to the optimal cost. The reason for that is that those agents take to sub-optimal route even though the blocked road is already repaired. Skepticism seems to be not beneficial in certain networks, which again matches the previous assumption in that non-skeptical drivers would perform better in “certain” networks.

The experiment showed that the assumption the author made at the beginning holds for the model. Out of a variety of input parameters the author chose: certainty and skepticism. There would be more parameters to analyze, but this result is sufficient for an initial validation effort. The author is confident enough that the model works properly.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SUMMARY AND FUTURE WORK

This thesis implemented an analysis environment for road networks. An agent-based model was developed in which agents were placed as truck-drivers in the road network environment. This has been done to study the impact of disruptions on the network on the behavior and the decision-making process of the agents. Each agent was equipped with an internal map that he kept updated. In addition each agent had its own personality, which was defined through three propensity genes. The first propensity gene determined the agent's level of skepticism. The second gene contained his learning preferences. The third gene showed its preferences for the three basic dimensions (time, distance and risk) and defined a basis for the agent's decision-making process to find the optimal route through the road network. In addition, the preferences for time, distance and risk were used to generate the agent's own interpretation on the road environment: the gradient distribution. This distribution is the agent's way to sense his environment. The decision making process was based on the gradient distribution together with its local view and its propensities.

This agent-based system can be used as a laboratory for various kinds of network analysis. In future related research, the impact of communication between agents can be investigated. Communication can range from no communication (current model) to direct radio communication between agents. Another interesting addition to the model would be an improved disruption layer that contained more sophisticated sabotage agents. A third step could be the implementation of management systems for sinks and sources to further complete the model. As a last refinement, a maintenance layer could be introduced to the model. This layer contains maintenance agents, which try to repair the blocked nodes in the most efficient way. These steps would further refine the model and make network interdiction analysis possible.

A goal of this thesis is to show that agent based models are a valuable tool to gain insights in complex problem domains. The diversity among agents, through their personality is beneficial for a rich and realistic behavior. Often users of agent-based models are tempted to immediately add all features to the model that they can observe in

the real world. The system build in this thesis has been focused on the basic requirements that are necessary to study the decision-making process of truck-drivers.

LIST OF REFERENCES

[**HOLLAND**] Holland J.H., "Hidden Order", Perseus Books, Cambridge Massachusetts, 1995

[**FERBER**] Ferber J., "Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence", Addison-Wesley, Harlow, England, 1999

[**HILES**] Hiles J. et al, "Innovations in Computer Generated Autonomy at the MoVES Institute", Technical Report MoVES Institute, Monterey, California, 2001

[**HILES2**] Hiles J., Class Notes to Introductory Agents Course, MV4015, MoVES Institute, Monterey, California, 2002

[**VANPUTTE**] VanPutte M. et al, "A Composite Agent Architecture for Multi-Agent Simulations," 11th Computer Generated Forces and Behavioral Representation Conference, Orlando, Florida, 7 - 9 May 2002

[**WOOLDRIGE**] Wooldridge M.J., "An Introduction to Multiagent Systems", Wiley, West Sussex, England, 2002

[**CARLEY**] Carley, K. "BIOWAR: Simulation of Disease Outbreaks using Social Networks", <http://www.casos.ece.cmu.edu/projects/BioWar/>, Carnegie Mellon University, Pittsburgh, PA

[**LAW**] Law A.M., Kelton W.D., "Simulation Modeling and Analysis" 3rd Edition, Mc Graw Hill, 2000

[**PAM**] Headquarters Department of the Army, Pamphlet 5-11 , "Verification, Validation and Accreditation of Army Models and Simulations", Washington, DC, 1999

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

The ARIEL software system was written completely in java, using the packages that are included in SUN's standard development kit (SDK Ver 1.4.1). No additional software package has been used.

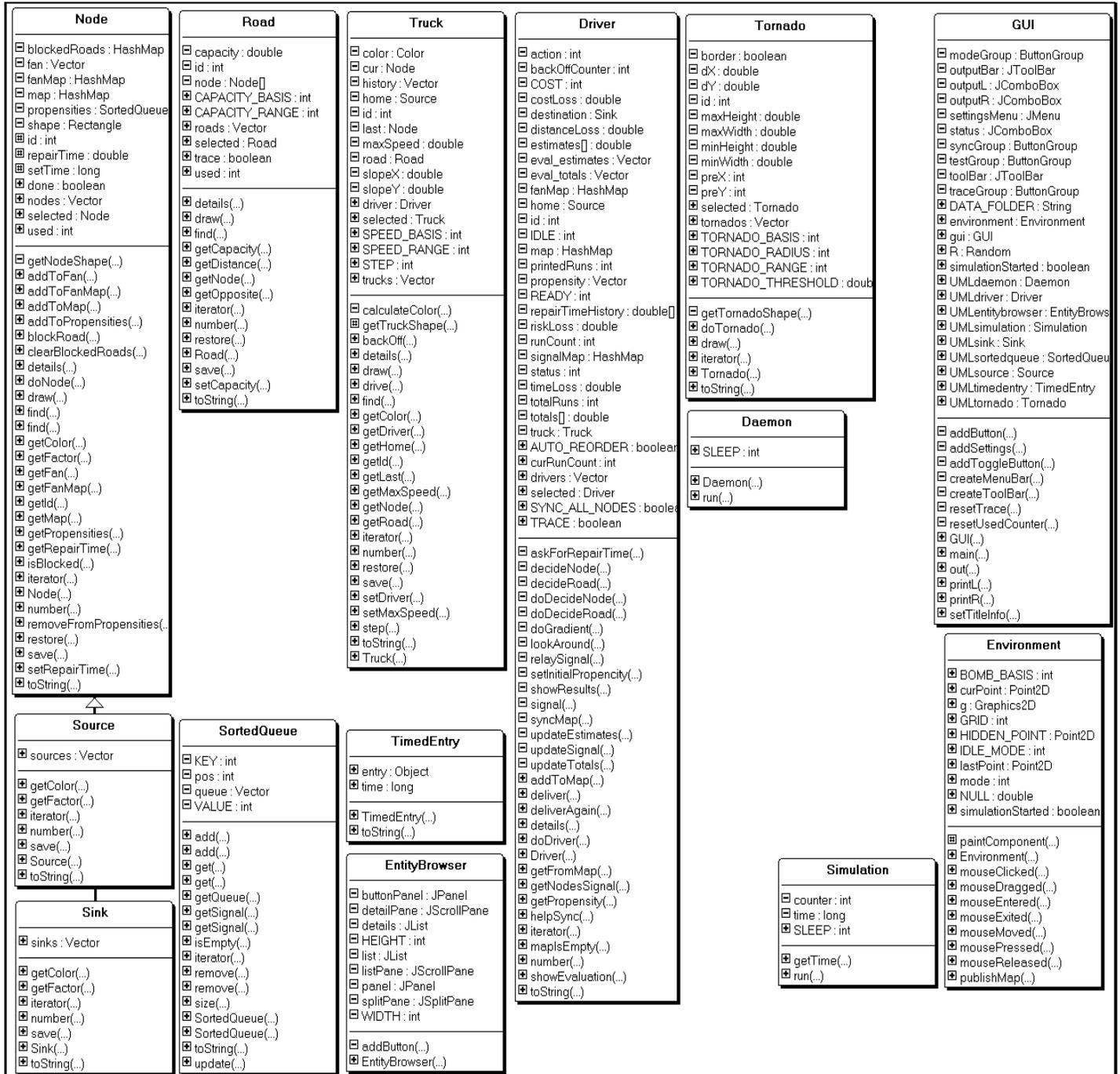


Figure 25. ARIEL Class Diagram

Figure 25 gives an overview about the classes used in the ARIEL system. The source code is downloadable at: <http://www.multiagent.info> or by contacting the author at thomas@orichel.de.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. Michael Zyda
Naval Postgraduate School
Monterey, California
4. Eugene Paulo
Naval Postgraduate School
Operations Research Department
Monterey, California
5. John Hiles
Naval Postgraduate School
MOVES Institute
Monterey, California
6. Peter Purdue
Naval Postgraduate School
Operations Research Department
Monterey, California
7. Frank Mahnke
Joint Warfare Analysis Center
Dahlgren, Virginia
8. Paul Schneider
Joint Warfare Analysis Center
Dahlgren, Virginia
9. Joerg D. Becker
University of Armed Forces Munich
Maisingerschluchtstr. 4a
82319 Starnberg
GERMANY