Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Studienarbeit

Till Peters

## Maschinelles Lernen in der „Robocup Rescue Simulation League“

*Fakultät Technik und Informatik*
*Studiendepartment Informatik*

*Faculty of Engineering and Computer Science*
*Department of Computer Science*

Till Peters

# Maschinelles Lernen in der "Robocup Rescue Simulation League"

Studienarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Softwaretechnik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck

Abgegeben am 18. April 2006

**Autor:**
Till Peters

**Thema der Studienarbeit:**
Maschinelles Lernen in der „Robocup Rescue Simulation League“

**Stichworte:**
RoboCup, Rettungssimulation, Agenten, künstliche Intelligenz, Muliti-Agenten-Systeme, rationale Agenten, Lernen, Maschinenlernen, (Sub)Symbolisches Lernen

**Kurzzusammenfassung:**
Diese Arbeit bereitet praktische und theoretische Ausarbeitungen auf, die Till Peters bei der Untersuchung mehrerer Teams der „RoboCup Rescue Simulation League“ anfertigen konnte.
Dies schließt Studienaufenthalte bei den „RoboCup Rescue Simulation League“ - Teilnehmern „Black Sheep“ (Universty of Auckland, Neuseeland) und vor allem „ResQ“ (Universität Freiburg, Deutschland) ein.
Die Arbeit beschreibt zunächst Einleitendes zu „RoboCup“ und der „Rescue Simulation League“. Es folgt eine Übersicht zu relevanten Ansätzen der künstlichen Intelligenz, die in den Hauptteil zum Maschinenlernen in der „RoboCup Rescue Simulation League“ überleitet. Abschließend sind die wichtigsten Ergebnisse der Arbeit, sowie ein Ausblick auf weitere Entwicklungen in einem Fazit zusammengefasst.

**Author:**
Till Peters

**Title of the paper:**
Mashine Learning  in the „Robocup Rescue Simulation League“

**Keywords:**
RoboCup, rescue simulation, agents, artificial intelligence, mulit agent systems, rational agents, learning, mashine learning, (sub)symbolic learning

**Abstract**
This thesis processes practical and theoretical approaches that Till Peters gained during his survey of different „RoboCup Rescue Simulation League“ – teams.
This includes the scrutinizing of the „RoboCup Rescue Simulation League“ – members „Black Sheep“ (Universty of Auckland, New Zealand) and especially  „ResQ“ (Universität Freiburg, Deutschland).
The following paper is introduced with general information on „RoboCup“ and its „Rescue Simulation League“, followed by an overview of relevant approaches concerning artificial intelligence. The main part consists of an elaboration about maschine learning in the „RoboCup Rescue Simulation League“ wich is conlcuded by a summary of the most important results and future prospects in this field.

# Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 10. April 2006

# Maschinelles Lernen in der „Robocup Rescue Simulation League“


**Eine Studienarbeit von Till Peters, Student an der Hochschule für Angewandte Wissenschaften Hamburg**

Universität Freiburg 2005,
University of Auckland 2005/2006,
University of Applied Sciences Hamburg 2006

# Inhalt

# 1 Einleitung

## 1.1 RoboCup

Die RoboCup - Initiative betätigt sich in unterschiedlichen Feldern der künstlichen Intelligenz und Robotik. Es ist eine internationale Organisation, die erfolgreich versucht diese Technologien möglichst publikumsnah zu realisieren. Dabei spielen Forschung und Lehre eine große Rolle – Die meisten Entwickler sind Studenten, Professoren und Mitarbeiter an diversen Universitäten auf dem ganzen Globus. Eine gute Möglichkeit in der Allgemeinheit für diese Arbeit Interesse zu wecken stellt Fußball dar, Roboterfußball. In diesem Gebiet liegt der Forschungsschwerpunkt und auch das große Ziel von RoboCup: Im Jahre 2050 eine Mannschaft von humanoiden Robotern zur Verfügung zu haben, die den derzeitigen menschlichen Fußballweltmeister schlagen. Zurzeit sind allerdings hauptsächlich andere Arten von Robotern im Einsatz. Die meisten fahren auf Rädern oder Rollen, es gibt aber auch eine Simulation in der virtuelle Softwarefußballer gegeneinander antreten.
Viele der beteiligten Hochschulen stellen ein Team, das sich in der Liga der entsprechenden Kategorie durchsetzen muss.

*Abbildung 1.1: Verschiedene Klassen von Fußballrobotern innerhalb des RoboCup*

### 1.1.1 RoboCup Rescue

Neben dem Fußball haben sich auch weitere Bereiche innerhalb der Organisation herausgebildet. Dazu gehören unter anderem das weniger populäre aber für die praktische Anwendung weitaus interessantere Gebiete der Lebensrettung in Katastrophensituationen:

„RoboCup Rescue“ und „RoboCup Rescue Simulation“. Diese beiden Projekte haben zum Ziel in einem Katastrophengebiet mit Hilfe von Robotern Menschen zu retten. Zum Beispiel durch Einsatz der Kräfte an Orten, die für menschliche Rettungseinheiten aufgrund von (nicht einzuschätzenden) Gefahren nicht zugänglich sind.
Seit 2001 gibt es bei den jährlichen RoboCup Weltmeisterschaften auch Wettbewerbe in diesen Ligen. Dabei geht es bei den realen Robotern eher um räumlich sehr beschränkte lokale Rettungsaktionen, während die virtuellen Kollegen in einer von Erdbeben und Feuern heimgesuchten Stadt ihr bestes geben, um Menschen und Gebäude vor dem Tod beziehungsweise der Zerstörung zu bewahren
Obwohl die beiden Projekte von einander getrennt arbeiten, ist natürlich langfristig eine Integration erwünscht und geplant.

*Abbildung 1.2: Unterschiedliche Rettungskräfte, entwickelt für Robocup Rescue*

### 1.1.2 RoboCup Rescue Simulation

Diese Arbeit befasst sich mit dem Softwareteil, also mit der „RoboCup Rescue Simulation“ und geht dabei auf die Schwerpunkte des Projektes ein. Dabei handelt es sich um Entscheidungsunterstützungen durch das Sammeln, Analysieren, Integrieren, Verarbeiten und

Verbreiten von Informationen. Für das Akkumulieren und die Distribution ist eine effiziente Art der Kommunikation nötig, während intelligentes Integrieren, Analysieren und Verarbeiten akkurate Vorhersagen und Planung ermöglichen.

#### 1.1.2.1 Umgebung der RoboCup Rescue Simluation League

Für alle Entwicklerteams gleich ist der Simulator, der aus verschiedenen Komponenten besteht:

Einem globalen Informationssystem, das sämtliche aktuelle Informationen über die Stadt, so wie Feuer, Gesundheitszustand der Zivilisten oder den Zerstörungsgrad der Gebäude speichert und an die Einsatzkräfte vermittelt. Einem Feuersimulator, der versucht möglichst realistisch die Ausbreitung des Feuers zu berechnen und diese an das globale Informationssystem weitergibt. Einem Massensimulator, der das Verhalten der Zivilisten darstellt, sowie einem Verkehrssimulator, der sich auf die Berechnung des Verkehrs konzentriert. Außerdem gibt es eine grafische Benutzerschnittstelle in 2D und in 3D.

*Abbildung 1.3: Die 2D- und 3D-Benutzerschnittstelle während der Simulation*

Die verschiedenen Teilnehmer der Liga arbeiten an Teams von Einsatzkräften, die sich aus Polizei, Feuerwehr und Ambulanz zusammensetzen. Jeder Typ dieser Einheiten hat mehrere bestimmte Aufgaben zu erledigen. So soll sich die Feuerwehr natürlich hauptsächlich mit dem Löschen der Feuerherde beschäftigen während die Polizei Barrikaden aus dem Weg räumen muss, um die Straßen passierbar zu machen, und die Ambulanz soll die verschütteten Zivilisten und auch Einsatzkräfte aus ihrer misslichen Lage befreien.

Neben diesen Primärzielen ist es aber zum Beispiel auch notwendig, das Terrain zu erkunden und neu entdeckte Zivilisten, Feuer und Barrikaden an die Stationen zu melden. Es gibt für jeden Typ der Einsatzkräfte eine entsprechende Station. Also Polizeirevier, Feuerwache und Krankenhaus. Diese sind dafür zuständig auf höherer Ebene gewisse Entscheidungen zu treffen, die Einheiten zu dirigieren und die Kommunikation zu koordinieren.

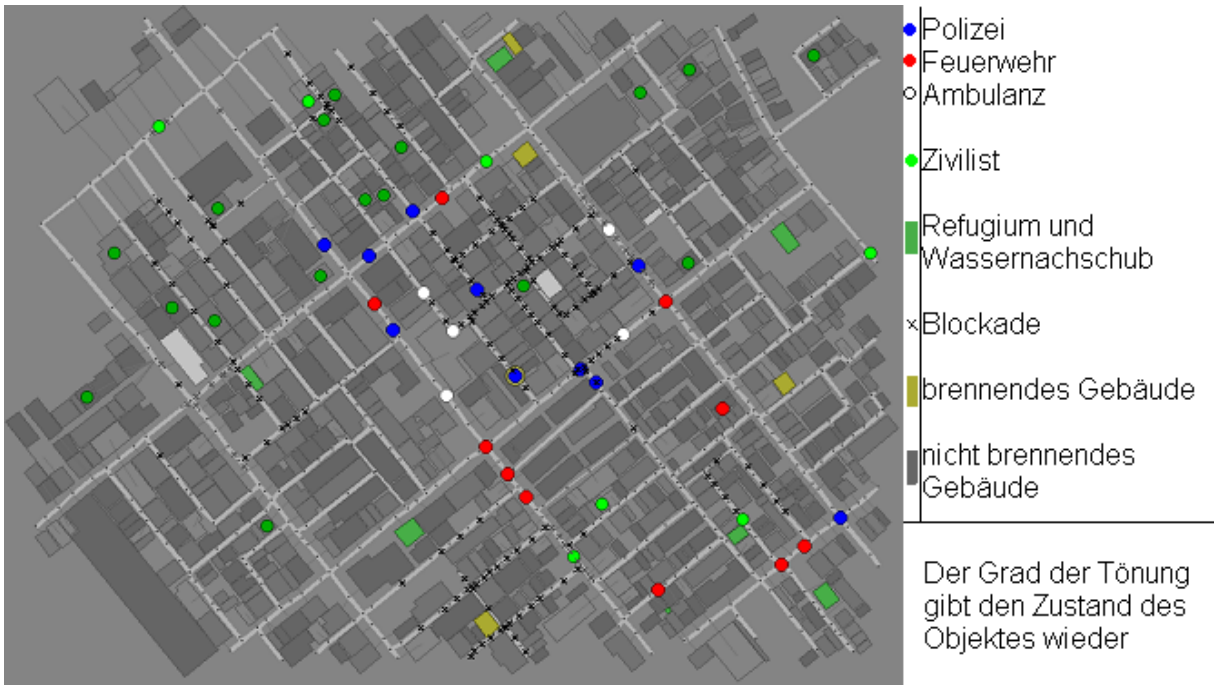


*Abbildung 1.4: Der übersichtlichere 2D-Viewer*

### 1.1.3 Von RoboCup zum maschinellen Lernen

Die Anforderungen an eine Rettungseinheit sind bei wenig komplexen Strukturen übersichtlich in vollständig ausprogrammierten Agenten unterzubringen. So waren die frühen Agenten innerhalb des RoboCup auch mit Hilfe traditioneller Informatik auf verteilten Systemen implementiert. Durch gestiegene Anforderungen und dem Interesse vieler Teams wurde es zunehmend schwerer, alle denkbaren Züge eines Agenten oder einer Gruppe von Agenten schon bei der Programmierung zu beleuchten: Der optimale Ansatzpunkt für ein lernendes System, dass durch Erfahrung die eigenen bevorstehenden und zurückliegenden Züge bewerten kann. Dies ist besonders Interessant einerseits durch die Autonomie der Agenten, und im Gegensatz dazu deren Teilnahme an verteilter künstlicher Intelligenz.

Mittlerweile sind viele Teams dabei, Aspekte maschinellen Lernens zu implementieren und es stellt sich die Frage inwiefern sich diese Strukturen von ihren traditionellen Vorgängern unterscheiden und welcher Vorteil sich aus der Nutzung von Strategien der künstlichen Intelligenz ziehen lässt.

# 2 Künstliche Intelligenz

## 2.1 Intelligenz, Wissen, Lernen

Bis heute ist es weder der Informatik noch Psychologie, Philosophie, Kognitions- oder Neurowissenschaften gelungen die Begriffe Intelligenz, Wissen und Lernen eindeutig zu Erfassen und Beschreiben. Es gibt diverse Ansätze, jedoch keinen Allgemein anerkannten. Unbestritten aber ist, dass diese Konzepte einander bedingen.
Als elementare Bausteine werden hier unter anderem die Symbolverarbeitung und -manipulation, die rationale Interaktion mit der Umwelt und das Vorhersagen von Veränderungen dieser gesehen.

## 2.2 Rationale Agenten

Es gibt ebenso keine eindeutige Definition eines Agenten im Sinne der künstlichen Intelligenz, jedoch existiert eine weitgehende Übereinkunft über minimale Merkmale eines solchen. Dies ist zum einen die Autonomität. Agenten handeln eigenständig um ein gegebenes Ziel zu erreichen. Sie erhalten weder genaue Anweisungen für jeden Aktionsschritt, noch wird die Auswahl der Aktionen vom Benutzer bestätigt oder zurückgewiesen. Des Weiteren sollte ein Agent zur Kommunikation in der Lage sein, zum Beispiel um Informationen mit Menschen oder anderen Agenten auszutauschen (siehe Multi Agenten Systeme). Ein Agent, ob Soft- oder Hardware, besteht üblicherweise aus Aktoren und Sensoren, die einen Kreislauf von Wahrnehmung und Reaktion ermöglichen. Dabei sollte er zielorientiert arbeiten ohne den genauen Weg zum Ziel im Vornhinein zu kennen – für diesen sorgt seine Eigenschaft lernend zu sein. Als rational wird er bezeichnet, da er aus einer Menge von Regeln eine vernünftige auswählen kann. Vernünftig im Sinne der Maximierung der Erfolgsrate auf Basis seines Wissens der Umgebung und seiner möglichen Aktionen.

Dabei kann es sich um einen sehr komplexen Agenten, mit diversen Aktoren und Sensoren, gesammeltem Wissen und Erfahrungen handeln, wie einem Rettungsroboter, der mit Hilfe einer umfangreichen Algorithmenbibliothek die Aktion räsoniert.

Allerdings ist per Definition sogar ein simpler Feuermelder ein Agent, ein so genannter reflexbasierter Agent, dessen Aktor Alarm auslöst, sobald die Sensorwerte durch den Rauch einen bestimmten Schwellwert überschreiten. Die Regeln dieses Agenten wären einfach: „Alarm aus, solange Schwellwert nicht überschritten“ und „Alarm an, sobald Schwellwert überschritten“.

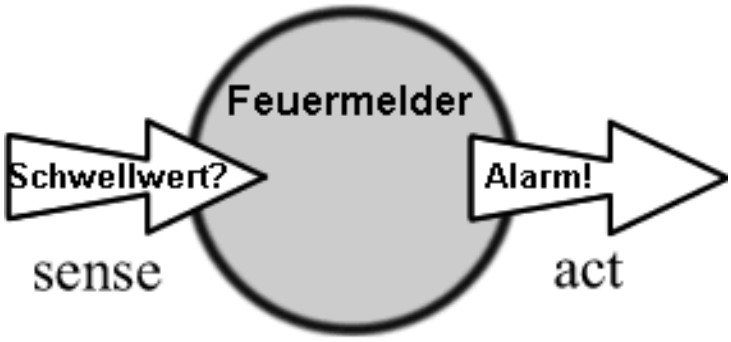


*Abbildung 2.1: Ein einfacher Agent*

Eine weitere Art eines Agenten ist die zustandsbasierte, bei der sich der Agent vorangegangene Beobachtungen merkt und die Auswahl der Regeln unter anderem von ihnen abhängig macht.
Zielbasierter Agent wird dieser genannt, sobald er Konsequenzen seines Handelns abwägt und versucht, den wahrscheinlichen zukünftigen Zustand der Umwelt mit einzubeziehen.
Interessanter Weise ist es möglich für jeden zielbasierten Agenten einen ausprogrammierten zustandsbasierten Agenten anzugeben, der äquivalent arbeitet. Dieser wäre allerdings bei komplexen Strukturen unverhältnismäßig aufwendig zu programmieren, da alle Eventualitäten bedacht werden müssen und eine weitaus größere Anzahl von Regeln implementiert werden müssten. Auch bei wechselnden Zielen ist der zustandsbasierte Agent wesentlich unflexibler.
Natürlich gibt es noch weitere Klassen von Agenten, unter anderem lernende oder eigenständig Ziele wählende Systeme.

## 2.3 Multi Agenten Systeme

Multi Agenten Systeme zeichnen sich dadurch aus, dass mehrere Agenten,
mit gegebener Weise unterschiedlichen Fähigkeiten, zur Lösung eines Problems kooperieren.
Eine solche verteilte künstliche Intelligenz besitzt keine zentrale Steuerung, das Resultat
ergibt sich aus dem Zusammenwirken unabhängiger Agenten. Um vernünftige Ergebnisse zu

erzielen bietet es sich an, dass die Agenten untereinander kommunizieren zum Beispiel um
Wissen und Vorhaben auszutauschen oder Aufgaben zu delegieren. Dabei treten immer
wieder Interessenkonflikte auf (zum Beispiel über Ressourcen und Aufgabenverteilungen),
die von den Agenten eigenständig zu lösen sind.
Kooperatives Problemlösen teilt sich also in das Erstellen und Verteilen von Teilaufgaben,
deren Bearbeitung und die Synthese dieser auf. Bei ungelösten Aufgaben wird mithilfe des
Wissens über Probleme über das Verfahren iteriert.
So können sich bei den Rettungsagenten dann die unterschiedlichen Einsatzkräfte
koordinieren, etwa das Anfordern der Räumung von Blockaden durch Polizisten in
unmittelbarer Nähe von geraden entdeckten Feuerherden oder die Absprache zum
gemeinsamen Löschen eines Gebäudes.

## 2.4 Maschinenlernen und Rationalität

Da das Lernen eine zentrale Rolle der natürlichen Intelligenz einnimmt, fällt dem
Maschinenlernen auch eine wichtige Bedeutung innerhalb des künstlichen Gegenstückes zu.
Maschinenlernen oder maschinelles Lernen wird definiert als *Computergestützte
Modellierung und Realisierung von Lernphänomenen*. Diese ungenaue Definition lässt
allerdings konkrete Methoden offen. Einige Herangehensweisen sollen hier diskutiert werden.
Wichtig ist aber immer das Rationalitätskriterium. Das Erlernte muss immer vor dem
Hintergrund der Rationalität gesehen werden, da die Maschine aufgrund des neuen Wissens
über richtige Lösungen eines Problems so wie ein rationaler Mensch entscheiden soll.

### 2.4.1 Überwachtes, Unüberwachtes und Verstärkungslernen

Überwachtes Lernen ist das effizienteste der drei Verfahren, wenn auch am wenigsten durch die Natur motiviert. Hierbei liegen sowohl Beispiel als auch Lösung vor, so dass nach dem Training der Erfolg direkt überprüft und das Gelernte gegebenenfalls angepasst werden kann. Zum Beispiel könnte ein Feuerlöschtrupp innerhalb der Simulation seine Vorgehensweise beim Löschen modifizieren, wenn ein bestimmter Brandherd mit einer anderen Anordnung der Löschfahrzeuge vorteilhafter zu löschen wäre. Es ist somit festzustellen, wie stark das Ergebnis von der eigentlichen Lösung abweicht.

Naturnäheres, allerdings weniger effizientes Lernen stellt das Verstärkungslernen dar. Es ist nicht möglich herauszufinden, wie groß die Abweichung vom Optimalwert ist, da die genaue Lösung nicht bekannt ist. Es ist jedoch zu erkennen, ob das Ergebnis eher richtig oder falsch zu bewerten ist. Obiger Löschtrupp kann durch die Zahl der abgebrannten und gelöschten Gebäude innerhalb einer Zeitspanne abschätzen, ob mehr Einsatzkräfte zum Löschen benötigt werden oder zum Entdecken verschütteter Zivilisten genutzt werden können.
Beim unüberwachten Lernen gibt es gar keine Rückmeldung, wie gut eine Aufgabe gelöst wurde. So muss selber gemerkt werden wie auf die Ergebnisse zu reagieren ist. Die Feuerwehr müsste selber herausfinden, dass nicht gelöschte Feuer sich ausbreiten und Zivilisten gefährden.

### 2.4.2 Symbolisches Lernen

#### 2.4.2.1 Versionenraum - Methode

Diese relativ einfache aber oft implementierte induktive Methode des Lernens von Regeln aus einer Menge von Beispielen stellt das Lernen als Suchproblem dar. Sie deckt allerdings nur Regeln ab, die sich mit dem logischen „und“ verketten lassen. Das logische „oder“ ist nicht erlaubt und daher fehlt eine große Menge möglicher Regeln schon in der Anfangsbetrachtung. Die spezifischste Regel, die sich aus den positiven Beispielen ergibt und die allgemeinsten, die sich aus den negativen ergeben, begrenzen den Bereich der möglichen Regeln. Falls ein solcher Algorithmus nur eine Regel aus den negativen Beispielen ableitet und dieser gleich der aus den positiven ist, so existiert nur diese eine Regel.

Diese Methode funktioniert allerdings nur bei einfachen Problemen. Sobald eine (einfachere) Regel auch Disjunktionen erhalten muss, versagt dieses Verfahren.

#### 2.4.2.2 Entscheidungsbäume

Ebenso übersichtlich und einfach zu erstellen sind Entscheidungsbäume. Sie bieten eine gute Möglichkeit um Disjunktionen, so wie Konjunktionen, darzustellen. Als Wurzel des Baumes bietet sich das Kriterium der Beispiele an, das der beste Indikator für das Ergebnis ist. Die Kanten sind die Elemente des Ergebnisraumes jedes Kriteriums. Weitere Knoten sind jeweils wieder die besten Indikatoren unter den restlichen Kriterien, bis hinunter zu den Blättern, an denen nur noch Gruppen von gleichem Ergebnis zu finden sind.
Diese Ketten können, als Graph dargestellt, übersichtlich abgelesen und als Regeln interpretiert werden.
Da der Baum schnell groß wird gibt es die Möglichkeit, ihn zur besseren Übersicht und zur Verminderung des Risikos der Überanpassung zu stutzen und damit einen Baum geringerer Tiefe und gröberer Unterteilung der Instanzen zu erhalten. Es resultieren jedoch eher Fehler aus stark gestutzten Bäumen.

#### 2.4.2.3 Instanzbasiertes Lernen

Eine bei großen Mengen von Beispielen aufwendige Methode stellt das instanzbasierte Lernen dar, bei dem ein neues Datum mit sämtlichen Beispielen verglichen und die Bewertung des Ähnlichsten kopiert wird.
Ein Vorteil ist allerdings, dass dieses Verfahren einfach sowohl mit diskreten als auch mit numerischen Werten arbeiten kann.
Es existieren zu diesem Verfahren diverse Varianten, die sich unter anderem mit Kompensierung des Nachteils teuren Durchsuchens sämtlicher Beispiele befassen.

### 2.4.3 Subsymbolisches Lernen

#### 2.4.3.1 Künstliche Neuronale Netze

Nach dem biologischen Vorbild bestehen Künstliche Neuronale Netze aus vernetzten, allerdings stark vereinfachten Neuronen. Ein solches Neuron besitzt mehrere Eingabewerte, die nach einer Gewichtung in einer Funktion (üblicherweise eine Summenfunktion) kombiniert werden. Unter Berücksichtigung eines Schwellwertes entscheidet die Aktivierungsfunktion, ob sich der Zustand (im allgemeinen 0 oder 1) des Neurons ändert. Dieser wird über den Ausgabeteil an angebundene Neuronen als deren Eingabewert verwendet. Das Lernen erfolgt nach Rückkopplung durch das Anpassen der Gewichte. Das Überprüfen des Netzes durch bisher nicht durchlaufene Trainingssätze.
Durch diese Art des Zusammenschaltens einfachster Einheiten können komplexe Aufgaben bewältigt werden. Besonders gut eignen sich Künstliche Neuronale Netze in Situationen, in denen sich keine effektive algorithmische oder regelbasierte Lösung anbietet.

#### 2.4.3.2 Genetische Algorithmen

Diese nicht induktive Art eines Suchverfahrens ermöglicht das überwachte Lernen durch Experimente und Empirie.
Hierbei gibt es nach Vorbild der Natur eine Population von möglichen Lösungen, die sich von Generation zu Generation durch Kreuzen zweier dieser Lösungen und Eliminierung der schlechtesten, verbleibenden Lösungen verbessert. Die Auswahl der zu eliminierenden und der zu kreuzenden Individuen vollzieht sich aufgrund einer Bewertungsfunktion und dem Zufall.
Ein weiterer Operator ist die Mutation, bei der eine Lösung willkürlich verändert wird. Variationen sind möglich durch Veränderungen in der Größe der Population und der Folgegeneration und der Häufigkeit von Mutationen.
Der Algorithmus kann nach Ablauf einer bestimmten Zeit, Erreichen einer gewissen Anzahl von Generationen, bei genügend guten oder konvergierenden Ergebnissen der Bewertungsfunktion abgebrochen werden.
Der Vorteil hierbei ist, dass der Algorithmus solange laufen kann, wie Zeit ist und auf jeden Fall ein Ergebnis liefert, während andere Algorithmen bis zum Ende laufen müssen, um ein Ergebnis zu liefern. Nachteilig wirkt sich allerdings die im Allgemeinen lange Laufzeit aus.

# 3 Maschinenlernen in der RoboCup Rescue Simluation League

## 3.1 Praktische Erfahrungen in Freiburg

Während meines Studienaufenthaltes an der Universität Freiburg war es mir möglich, eine umfangreiche Einsicht in die Konzeption eines Rescue-Agenten zu bekommen. Die API des Freiburger Teams bietet für den direkten Einstieg sämtliche Grundfunktionen, beispielsweise zur Kommunikation der Agenten untereinander und zur Pfadplanung und eignet sich daher hervorragend um eigene Ideen zu implementieren und zu testen.

In der Wettbewerbssituation läuft der Simulator auf dem Server während die Agenten des gerade antretenden Teams sich von einem anderen Rechner aus über TCP verbinden. Die Simulation ist Runden basiert und in 300 Zeitabschnitte eingeteilt. Innerhalb dieser zeitlichen Begrenzung können die Agenten ihre Berechnungen durchführen. Je nach Verlauf der Simulation verringert sich der anfängliche Punktestand von 100. Vor allem der Tod von Zivilisten und das Ausbreiten von Feuer wirken sich negativ aus. Für jedes Team wird zu Beginn der Simulation der Ursprungszustand wiederhergestellt um allen Agenten die gleichen Voraussetzungen zu geben.

Der modular aufgebaute Simulator, in der von mir benutzten Version 0.48, kontrolliert neben der Zeit die Ausbreitung des Feuers, das Verhalten der Zivilisten, sowie das Auftreten neuer Barrikaden.
Bestimmte Module sind zum Beispiel der Verkehrsimulator, der für die Zivilisten, für die Feuerausbreitung und so weiter. Deren jahrelange Entwicklung hat sie ziemlich nah an die Realität heran gebracht. So wird Wind bei der Feuerausbreitung berücksichtigt und der Menschenmengensimulator berechnet das Panikverhalten der Opfer.

Die Anzahl der Agenten ist nicht festgelegt, sie bewegt sich allerdings etwa bei um die zehn je Kategorie (Polizei, Feuerwehr, Ambulanz) und mindestens dreimal so vielen Zivilisten. Diese müssen Anforderungen für Bewegungen und Aktionen an den Simulator senden. Vom Simulator erhalten die Agenten ihre Position und eine Sicht auf die aktuelle Umgebung, sowie die zu empfangenen Nachrichten.

Die Rettungskräfte sind zweigeteilt in Kopf und Körper. Letzterer wird vom Simulator kontrolliert um sicherzustellen, dass die Agenten der unterschiedlichen Teams gleiche körperliche Vorraussetzungen haben. Es ist so zum Beispiel nicht möglich, eine Einheit schneller laufen zu lassen, als vom Simulator erlaubt. Auch die Größe des Wassertanks der Feuerwehrfahrzeuge ist nicht vom jeweiligen Team zu bestimmen, sondern allgemein für alle Teams festgelegt. Diese Strategie führt zu einem ausgeglichenen Wettbewerb, da kein Agent auf diese Art und Weise einen Vorteil erlangen kann.

Das Hauptaugenmerk bei der Entwicklung innerhalb der RoboCup Rescue Simulation League liegt also auf Kognition, Kommunikation und rationaler Entscheidungsfindung.

Auch für die Wahrnehmung gibt es bei den Agenten gewisse, vom Simulator festgelegte Grenzen. So können die Agenten in einem kleinen Bereich “sehen” und “sprechen” beziehungsweise “hören”. So bemerken die Agenten unter anderem verschüttete Zivilisten und Rettungskräfte, Blockaden oder Feuer. Mit dem Funk haben sie aber auch eine über das ganze Gebiet reichende Kommunikationsmöglichkeit.
Die gesammelten Informationen müssen bewertet und kategorisiert werden, damit eine vernünftige Entscheidung getroffen werden kann.

Für die Kommunikation gibt es allerdings verschiedene Beschränkungen. Zum einen sprechen die Agenten einer Kategorie, zum Beispiel die Polizisten, nur untereinander und mit der Polizeistation, niemals zu  Agenten anderer Art. Die Stationen kommunizieren untereinander und können so auch Nachrichten der Feuerwehr an die Ambulanz weiterleiten. Wenn sich zwei Agenten in unmittelbarer Nähe befinden können Sie sich auch zusätzlich über den “Hear/Say”-Kanal unterhalten über den sich aber vor allem Zivilisten melden. Oft lohnt es sich aber nicht, so Informationen auszutauschen, da die betroffenen Agenten meist dieselbe lokale Sicht auf die Welt haben. Auch ist nicht garantiert, dass alle gewünschten Einheiten die Nachricht bekommen. So kann es passieren, dass bei einer eventuellen Auswahl eines neues Anführers bei der Feuerwehr die am meisten geeignete Einheit nicht mit einbezogen und somit nicht die optimale Lösung gefunden wird.

Die Kommunikation mit den Stationen über Funk wird mit so genannten Token durchgeführt. Es gibt Token für bestimmte Nachrichten, wie zum Beispiel die Meldung eines verschütteten

Zivilisten oder den Auftrag diesen zu retten. Die Art und Struktur dieser Token ist durch die API festgelegt. Es ist jedoch möglich weitere Token für neue Nachrichtenarten nach diesem Prinzip zu erstellen. Wichtig hierbei ist der geringe Speicherbedarf eines Tokens, der es ermöglicht Nachrichten schnell zu versenden und zu interpretieren.

Die API der Freiburger vermindert die Anzahl der Nachrichten, die ein Agent je Zeiteinheit versenden kann von vier auf zwei. Dies geschieht, weil die Erfahrung gezeigt hat, dass trotz eines Prioritätensystems zu oft wichtige Nachrichten nicht verarbeitet werden, da einfach zu viele gesendet werden und die mobilen und stationären Agenten mit dem Verarbeiten nicht hinterher kommen.

Die Klassenhierarchie des Freiburger Teams stellt sich folgendermaßen dar:

```
Thread
    Agent
        CenterAgent
            AbulanceCenterAgent
            FireStationAgent
            PoliceOfficeAgent
        HumanoidAgent
            AmbulanceTeamAgent
            FireBrigadeAgent
            PoliceForceAgent
```

Die von Thread abgeleiteten Klassen unterteilen sich in stationäre (CenterAgent) und mobile (HumanoidAgent) Agenten. Diese werden entsprechend ihres Typs (Feuerwehr, Polizei, Ambulanz) erneut aufgespalten. So können die allgemein nutzbaren Fähigkeiten, wie die Kommunikation, in der Klasse Agent untergebracht werden. Bewegung ist natürlich nur den Instanzen der Subklassen von HumanoidAgent möglich. CenterAgent Derivate haben wiederum andere Kommunikationsmöglichkeiten als ihre Humanoiden Einsatzkräfte.

Das detaillierte Klassemodel ist natürlich viel umfangreicher und an dieser Stelle nicht sinnvoll zu erklären. Die genannten Klassen sind allerdings ausschlaggebend für neue

Entwickler. Dieses Framework erlaubt den Agenten noch kein rationales Verhalten, sondern bietet lediglich die Grundfunktionen um ein solches Verhalten zu implementieren. Die Klassen enthalten Funktionen um auf gewisse Ereignisse reagieren zu können. Diese müssen allerdings noch mit Code gefüllt werden. So kann zum Beispiel der Entwickler bei einer Anforderung zur Räumung einer oder mehrerer Blockaden eine eigene Herangehensweise implementieren.

Zunächst habe ich den Polizeikräften das Verhalten einprogrammiert, Blockaden in einer Reihe alle hintereinander aufzuräumen. Vorher sind die Polizisten blind durch die Stadt gelaufen und haben nach einer Räumung nicht gleich die nächste Blockade geräumt, sondern sind einfach wieder losgelaufen – meistens in eine unsinnige Richtung. Diese Veränderung brachte schon starke Verbesserungen mit sich - allerdings wäre es noch effektiver, wenn die Polizisten herausfänden, mit welcher Priorität sie sich in welchem Stadium der Simulation am besten dem Räumen von Blockaden beziehungsweise dem Erkunden des Terrains widmen.
Bei Meldung eines Feuers habe ich der Feuerstation erst einmal vorgegeben, den nächsten nicht beschäftigten Löschzug abzukommandieren. Je nach Größe des Feuerherdes werden natürlich unterschiedlich viele Einheiten benötigt. Dies im vornherein festzulegen stellte sich als relativ schwer heraus, da es bei der Abschätzung der Dringlichkeit viele entscheidende - teilweise schlecht zu beurteilende - Faktoren gibt. So zum Beispiel der Erfahrungswert, wie sich das Feuer ausbreitet, wie viele Zivilisten bedroht sind oder wie sich die Verfügbarkeit von Rettungskräften in naher Zukunft darstellt. Es zeigte sich also bereits bei den Vorüberlegungen, dass ein lernender Ansatz durchaus erfolgsversprechend ist. Ebenso da es sich wenig später herausstellte, dass sich die Agenten bei strenger Befolgung ihrer statischen Routinen gegenseitig stark stören: Es bilden sich temporäre Blockaden, die sich aufgrund von Verklemmung oft gar nicht mehr auflösen.
Das Verhalten der Agenten nimmt bei einer solchen umfangreichen Simulation stark an Komplexität zu. Somit fällt es immer schwieriger alle Möglichkeiten zu bedenken und statisch zu programmieren. Und schon das winzige Aktionsrepertoire meiner Agenten birgt ein großes Potential für unerwartete und unerwünschte Nebeneffekte, denen man mit dynamischen Lernmethoden gezielt entgegenwirken kann.

Der von den Freiburgern entwickelte Offline Viewer ermöglicht eine umfangreiches Debugging des Codes und eine tiefgehende Analyse des Verhaltens der verschiedenen Agenten, indem er die Logfiles lädt und verarbeitet.

Es ist möglich für jeden Agenten zu jedem Zeitpunkt beispielsweise die genaue Position, Gesundheit und den aktuellen Auftrag zu observieren. Auch die Eigenschaften eines jeden Gebäudes, wie Stärke des Feuers oder Anzahl der Zivilisten, sind übersichtlich dargestellt.

In einem weiteren Fenster ist in das Protokoll der Kommunikation aller Agent seinen zu sehen. Hieran lassen sich Aktionen der Einsatzkräfte hervorragend nachvollziehen.
Da der Offline Viewer modular erstellt wurde, sind beliebige Erweiterungen denkbar. So könnte man den kompletten zurückgelegt den Weg eines Polizisten farbig unterlegen um zu untersuchen an welchen Stellen der Agent unsinnige Aktionen durchführt.


*Abbildung 3.1: Der Offline Viewer Freiburg ResQ*

## 3.2 Lernziele

Die Rettungsagenten können, sollen sie komplexes Verhalten simulieren, bewerten wie erfolgreich eine bestimmte Aktion war oder (aufgrund von Erfahrung in ähnlichen Situationen) sein wird. Gelernt werden soll dabei, sich in einer mehr oder weniger übersichtlichen Umgebung zurechtzufinden und sich mit den anderen Agenten zu koordinieren um möglichst effizient Menschen zu retten, Gebäude zu löschen und Barrikaden zu räumen.

## 3.3 Untersuchungen beim Team “ResQ Freiburg”

Das Team „ResQ Freiburg“ der Universität Freiburg, Gewinner der RoboCup Rescue Simulation Weltmeisterschaft von 2004, hat mehrere neue Methoden zum maschinellen Lernen innerhalb der Liga implementiert. Dies betrifft vor allem die Planung von zeitkritischen Rettungsmissionen und das Auswerten von großen Datenbeständen der Logdateien. Neben der Voraussage der verbleibenden Lebenszeit von verschütteten Zivilisten und einer darauf basierenden Rettungsreihenfolge, spielt das Maschinenlernen auch bei der Löschreihenfolge und beim aktiven Erkunden eine Rolle.

Um die Zivilisten im Katastrophengebiet möglichst genau zu lokalisieren gibt es innerhalb der Wissensbasis der Agenten für jeden Zivilisten eine Menge seiner möglichen Aufenthaltsorte und für jeden dieser Orte eine Menge von Zivilisten, die sich an diesem Ort aufhalten könnten. Zu Beginn enthalten die Mengen der Orte sämtliche Orte und die Mengen der Zivilisten jedweden Zivilisten. Durch positive oder negative Berichte der Agenten wird diese Wissensbasis permanent auf den neuesten Stand gebracht. So ist es dann möglich die Aufenthaltswahrscheinlichkeit eines bestimmten Zivilisten an einem bestimmten Ort zu berechnen. Umgekehrt ebenso die wahrscheinliche Anzahl von Zivilisten an einem ausgewählten Ort.
Diese Ausbeutung der Wissensbasis gibt nach kurzer Zeit eine gute Übersicht, wo sich innerhalb der Stadt noch Zivilisten aufhalten und ermöglicht somit ein viel effizienteres Suchen nach den verbliebenen Verschütteten.

Auch beim weiteren Erkunden des Gebietes benutzen die Agenten das Wissen aus der Wissensbasis um die Stadt in Cluster aufzuteilen. Dies geschieht aufgrund der Kenntnis von

allen Orten und den Verbindungen zwischen ihnen. Da sich diese, zum Beispiel durch Räumung von Blockaden ständig ändern, muss die Aufteilung in die Cluster durchgehend neu berechnet werden. Des Weiteren wird noch kalkuliert ob die Nutzenwahrscheinlichkeit, also die Aussicht auf Entdeckung eines Zivilisten, mit den Kosten für den Weg in einem vernünftigen Verhältnis steht. Entsprechende Werte sind je nach Verlauf der Simulation unterschiedlich und werden daher experimentell herausgearbeitet und gelernt.

Um der mehrfachen Durchsuchung gleicher Abschnitte der Stadt durch verschiedene Agenten entgegen zu wirken, kommunizieren die Einsatzkräfte untereinander. Einmal indirekt über die Stationen und eine Wahrscheinlichkeitsverteilung – also im Auftrag. Andererseits, bei direktem Zusammentreffen der Einheiten, verhandeln sie die Aufgaben auch über die „hear-say“ Kanäle auf kürzesten Strecken. Sie lernen so, wo schon gesucht wird. Allerdings ist die Wegstrecke, die ein Agent je Zeitabschnitt gehen kann, ist mit Blick auf den Aufwand der Kommunikation unverhältnismäßig groß und so manchmal schneller doppelt abgesucht. Aufgrund der begrenzten Anzahl von Nachrichten, die ein Agent je Zeiteinheit verarbeiten kann, bringt die Nutzung des Funks auf großen Reichweiten keinen Vorteil. So lohnt es sich meistens nicht, die Nachrichtenkanäle mit Botschaften dieser Art zu überfüllen. Zumal die Kommunikation zwischen den Agenten auf weiten Strecken über die Stationen laufen müssten.

Als Grundlage für das Erstellen einer Rettungsreihenfolge nimmt „ResQ Freiburg“ die Vorhersage der Lebenserwartung der bekannten Zivilisten. Diese werden klassifiziert als Überlebende, zu Rettende und praktisch tote. Dies ergibt sich, da es von vornherein Verschüttete gibt, die am Ende der Simulation auch ohne Hilfe noch am Leben sind und wiederum andere, die trotz verstärkten Einsatzes nicht mehr zu retten sind. Alle anderen müssen je nach Dringlichkeit gerettet werden. Diese wird durch Klassifikationsalgorithmen unter anderem mithilfe des Verschüttungsgrades und der Lebensenergie des Opfers ermittelt und mit den übrigen Agenten in Zusammenhang gestellt.
Die Reihenfolge der Rettung ist ein typisches Handlungsreisenden Problem, dass mit einer Laufzeit von O(n!) und einem relativ großen n nicht direkt effizient berechnet werden kann.

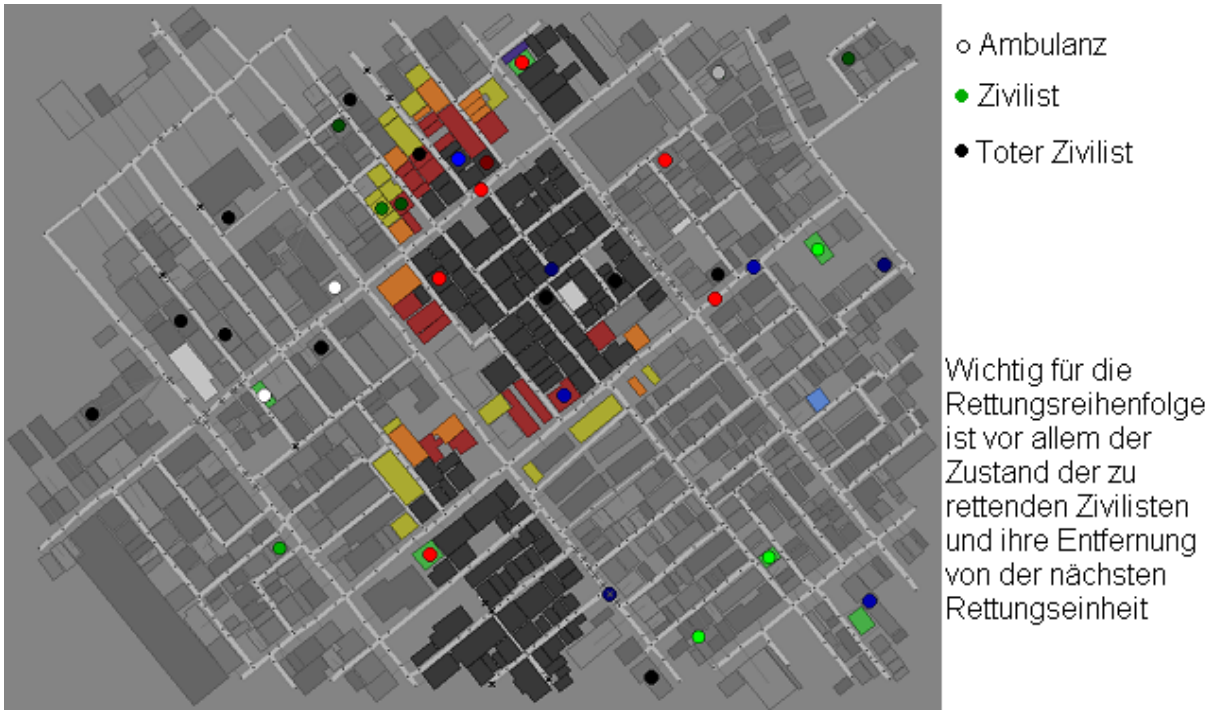


*Abbildung 3.2: Rettungsreihenfolge ermitteln*

Zusätzlich wäre noch die voraussichtliche Todeszeit des Zivilisten mit zu kalkulieren. An dieser Stelle setzt „ResQ Freiburg“ mit einer Reihenfolgeoptimierung auf der Basis von einem genetischen Algorithmus an. Initialisiert wird dieser mit gierigen heuristischen Lösungen, die zum Beispiel nahe zusammen liegende oder früh sterbende Opfer bevorzugen. Die Populationsgröße von zehn Lösungen ermöglicht die Berechnung von 500 Populationen durch die Ambulanz in jedem Zeitabschnitt. Die Fitnessfunktion bewertet nach geretteten Zivilisten. Die beste Lösung wird dann an die Rettungseinheiten weitergeleitet. Nebenher läuft noch ein reaktives System, das überprüft ob eine neue Information besonders wichtige Rettungsfälle beinhaltet und die Rettungssequenz neu erstellt werden muss. Da sich die Voraussetzungen permanent ändern, käme es zu ineffizienten Ergebnissen, wenn sich die Rettungsreihenfolge jedes Mal anpassen würde. Die Einsatzkräfte würden jeden Moment in eine andere Richtung geschickt werden, was bei einer so zeitkritischen Aufgabe sehr negativ auffällt.

Auch beim Löschen der Feuer ist eine gute Vorhersage der Ausbreitung besonders wichtig. Für jedes Gebäude werden also die umliegenden Bauten bestimmt und die Zeit überschlagen, die es braucht, damit der Brand übergreifen könnte. Diese „orangetime“ genannte Zeit lernen

die Agenten von „ResQ Freiburg“ durch Data Mining Techniken auf dem Datenbestand großer Mengen von aufgezeichneten Simulationsläufen. Diese Methode ermittelt so neben dem wahrscheinlichen Ausbreitungsmoment auch den Schwierigkeitsgrad des Gebäudes. Besonders ausschlaggebend für die Klassifikation ist natürlich auch die Anzahl von Zivilisten in den betrachteten Gebäuden und deren Nachbarbauten.

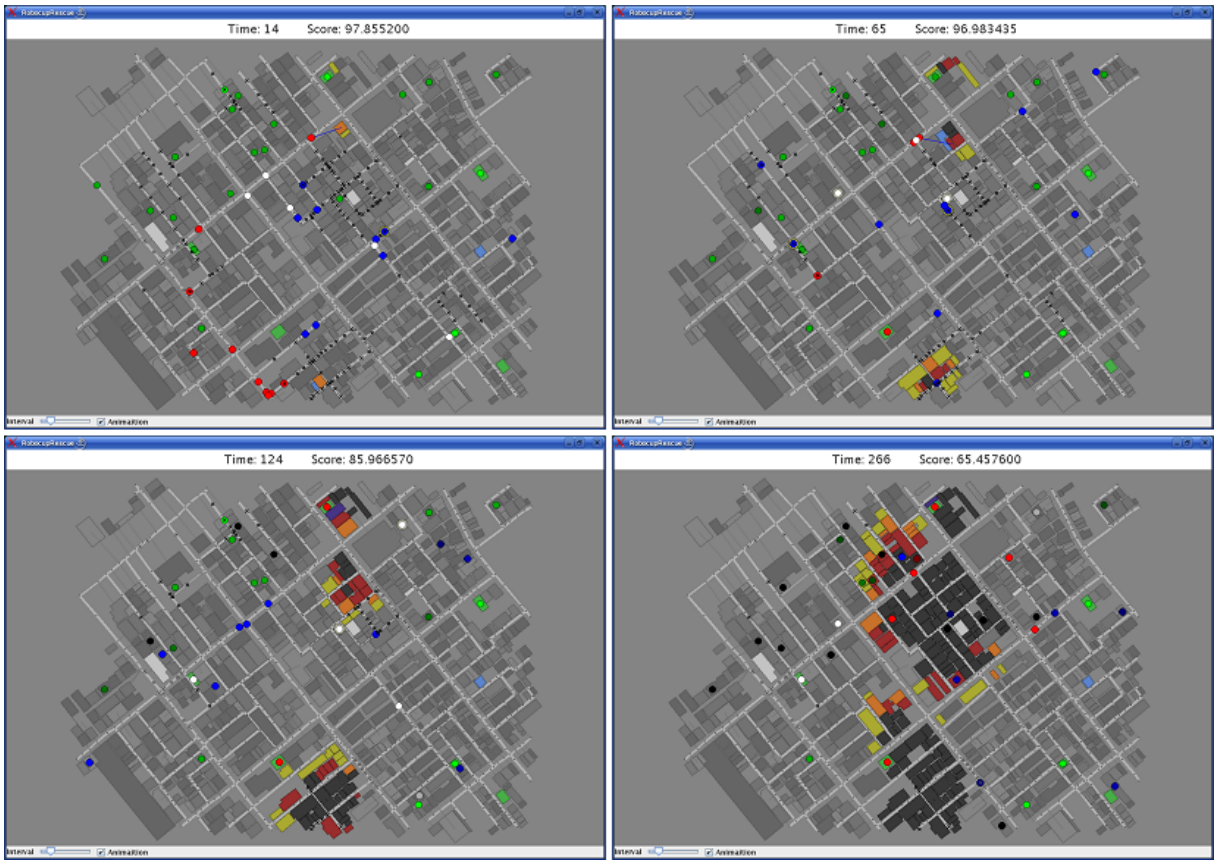


*Abbildung 3.3: Ausbreitung und Verschmelzung von Feuerherden*

Zur Ermittlung einer sinnvollen Reihenfolge werden auf dieser Basis dann Cluster von Feuern gebildet, die aktuell verbundene oder in weniger als n Zeiteinheiten verbundene Brandherde gruppieren. Zusammen mit den Wegkosten und der Anzahl der schon an einem Brandherd aktiven Einheiten ergibt sich daraus eine Abarbeitungssequenz.
Die Entscheidung, welches Gebäude innerhalb eines Brandes zunächst gelöscht werden soll fällt ein Agent ebenso aufgrund der angrenzenden brennenden Bauten und den verschütteten Zivilisten. Gebäude, die inmitten eines Feuers liegen oder mit den verfügbaren Einsatzkräften nicht zu löschen sind, werden dabei zuerst aus der Betrachtung entfernt. Zusätzlich geht noch

in die Bewertung mit ein, wie weit das jeweilige Gebäude vom Rand der Stadt und damit dem der Simulation entfernt ist.
Das Gewicht einzelner Kriterien ist allerdings festgelegt und es wird nicht erlernt, ob zum Beispiel die Anzahl der zu rettenden Zivilisten stärker zu berücksichtigen sein sollte als die Entfernung zum Stadtrand.

## 3.4 Vorgehensweisen vom Team “Damas”

Ähnlich dem Team „ResQ Freiburg“ hat das Team „Damas“ der Laval University in Kanada Methoden zum maschinellen Lernen untersucht und für ihre Agenten genutzt. Der Schwerpunkt liegt hier auf den Feuerwehrkräften. Diese sollen lernen zu beurteilen, wie effektiv ihre Löschmethoden sind um sie gegebenenfalls zu optimieren.

Um die eigenen Möglichkeiten und den Nutzen einer Aktion abschätzen zu können, bedient sich ein Agent ebenfalls der Wissensbasis. So können im Voraus Berechnungen zur Gruppierung der Feuer angestellt werden. Dies übernimmt beim Team „Damas“ die Feuerstation, da sie über ein besseres globales Wissen vom Zustand der Stadt verfügt als die einzelnen mobilen Agenten, die fortwährend neue lokale Informationen an die Station senden. Die gesammelten Daten ermöglichen unter Verwendung von Kriterien, ähnlich denen vom Team „ResQ Freiburg“, eine erfolgreiche Unterteilung in Feuerzonen.
Im zweiten Teilschritt wählen die Feuerwehrkräfte auf Basis ihrer lokalen Kenntnisse das wichtigste Gebäude aus. Dabei lernen sie, wie sich in bestimmten Fällen Aufwand und Nutzen zueinander verhalten.
Beide, mobile und stationäre Agenten, benutzen dabei die gleiche Herangehensweise. Anhand eines Entscheidungsbaumes kann jeder Agent seine Erfahrungen beim Löschen festhalten. Dies geschieht, indem Blätter des Baumes expandiert, also weiter unterteilt werden. Dabei sorgt er selber dafür, dass der Baum nicht zu groß wird: Er behält nur Verzweigungen, die einen hohen Nutzen versprechen.
Die Aktionen und den entsprechenden Nutzen speichert der einzelne Agent in mehreren Aktionsketten. Eine solche enthält zum Beispiel die Abfolge der Aktionen um ein bestimmtes Gebäude zu löschen. Ebenso den berechneten Nutzen.

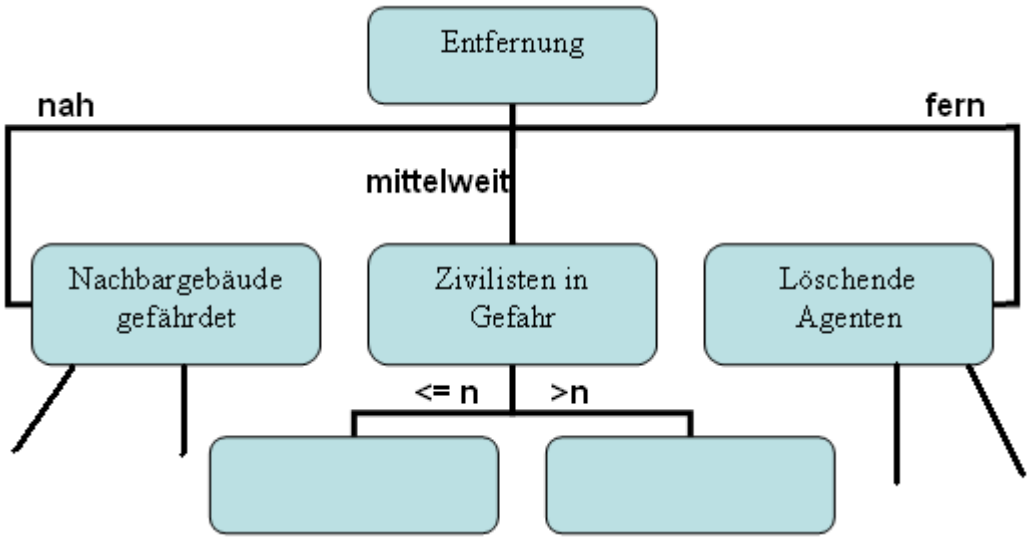


*Abbildung 3.4 Beispilehafter Entscheidungsbaum eines Rettungsagenten*

Bei nicht diskreten Attributen, wie den gefährdeten Zivilisten, wird ein Schwellwert eingeführt, der die Aufteilung in zwei Zustände ermöglicht. Bei der Entfernung wird mit unscharfen Werten gearbeitet.
Da das Lernen nicht online, während der Simulation läuft, sondern hinterher, haben die Agenten genügend Zeit ihre Aktionsketten auszutauschen und den neu gruppierten Baum an alle Agenten zu verteilen. Diese Maßnahme beschleunigt das Lernen ungemein.
Lernen heißt in diesem Fall eigentlich Klassifizieren der Erfahrungen und so werden Aktionsketten mit ähnlicher Wirkung zusammen gruppiert. Dies beschränkt wieder die Größe des Baumes. Der Wert des erwarteten Nutzens wird in den Blättern des Baumes gespeichert. Das Aktualisieren des Baumes und der Werte in den Blättern läuft in mehreren Schritten ab. Nachdem alle Aktionsketten zusammengetragen und in den Baum eingeordnet wurden,

aktualisiert der Algorithmus die Werte in den Blättern entsprechend den neuen Einträgen. In diese Aktualisierung gehen direkter und indirekter Nutzen einer Aktion sowie Abschätzungen über den Zusammenhang zwischen Einträgen innerhalb und zwischen Blättern ein. Nachdem

dies erledigt ist, wird überprüft, ob es sinnvoll ist, einige Blätter zu expandieren und die enthaltenen Aktionen aufzuteilen. Dafür werden für jedes Blatt alle Attribute, die eine Unterscheidung ermöglichen getestet. Das Attribut mit der größten Unterscheidung, sofern es einen bestimmten Schwellwert überschreitet, wird benutzt, um den Baum zu verfeinern. Im Anschluss müssen die Werte in den Blättern erneut berechnet werden.
Es kann nun jeder mobile Agent abschätzen, welches Gebäude am wichtigsten ist. Ebenso wie viele Einsatzkräfte für das Löschen der Gebäude in einer zugewiesenen Zone benötigt werden und wie viele bereits vor Ort oder auf dem Weg sind. Anhand dieser Informationen kann er dann entscheiden, zu welchem Feuer er sich begibt. Da sich die lokale Übersicht aller Agenten stark ähnelt, funktioniert die Koordination ganz ohne teure Kommunikation zwischen den Einheiten.
Um die Einsatzkräfte den verschiedenen Zonen zuzuweisen benutzt der stationäre Agent, die Feuerwehrwache, einen ähnlichen Baum. Dieser wird auf die gleiche Weise erstellt und erweitert und ermöglicht ebenso das Lernen. Allerdings auf einer höheren Wahrnehmungsebene. So kann die Station lernen, wie es die verfügbaren mobilen Agenten am besten verteilt, beziehungsweise welche Feuerzone am wichtigsten ist und wie viele Löschkräfte benötigt werden. Wichtig dabei ist natürlich die Entfernung der Agenten von der jeweiligen Zone.

# 4 Fazit

## 4.1 Nutzen des Maschinenlernens in der RoboCup Rescue Simluation League

Für komplexe Aufgaben ist es schwierig bis unmöglich vollständig ausimplementierte Problemlösungen zu entwerfen. Es mag noch mit Übersicht beurteilen und programmieren zu sein, auf welche Weise ein bestimmter Feuerherd am effizientesten zu löschen ist. Da es sich aber bei RoboCup um eine dynamische Umgebung handelt, ist eine solche Herangehensweise unpassend. Die Untersuchung von den Teams „ResQ Freiburg“ und „Damas“ haben gezeigt, dass sich ein lernender Ansatz, speziell bezüglich des Löschens, als die wesentlich flexiblere Alternative herausgestellt hat.

Die große Menge der anfallenden Logdateien zu nutzen, um wichtige Zusammenhänge von Situationen und erfolgreichen Aktionen herzustellen, erweist sich bei den Agenten des Teams „ResQ Freiburg“ als gewinnbringend. Bei der Vorhersage möglicher Feuerausbreitung liefern die durch Data Mining Techniken erlangten Daten sehr gute Ergebnisse. Ebenso für den Schwierigkeitgrad eines brennenden Gebäudes. Nach vielen Simulationsläufen haben die Agenten gelernt, wie schwierig es wahrscheinlich sein wird, ein bestimmtes Gebäude oder einen Feuerherd zu löschen und können entsprechend disponieren. So ergibt sich ein klarer Vorteil durch effizientes Ausnutzen der verfügbaren Einsatzkräfte.
Mit dieser Methode ist es den Agenten auch möglich auf geänderte Parameter sinnvoll zu reagieren und sich damit einem dynamischen Simulator unter ständiger Entwicklung anzupassen.

Der klare Vorteil von einem genetischen Algorithmus, dass er zu jeder Zeit ein Ergebnis liefert, ermöglicht dem Team „ResQ Freiburg“ eine schnelle Reihenfolgeoptimierung. Die Laufzeit von O(n) ist bei großer Anzahl von Zivilisten (n) nicht mehr akeptabel und verschwendet  wertvolle Rettungszeit.

Auch beim Erkunden des Gebietes lohnen sich Lernstrategie und Wissensausbeutung. Beim Freiburger Team  legen die Agenten eine Wissensbasis über Aufenthaltsorte von Zivilisten, Barrikaden und Straßen an und erlernen daraus den Nutzen verschiedener aktuell möglicher Aktionen.

Statische Vorgehensweisen haben durch ihren fest vorgegebenen Ablauf wenig Spielraum, da die Vielzahl von möglichen Aktionen und Situationen nicht vollständig abgedeckt werden kann. Ihre lernenden Kollegen können sich besser anpassen, allerdings benötigen Sie auch eine in diesem Fall akzeptable Zeitspanne als Lernphase. Vernünftige Voreinstellungen können hierbei sehr hilfreich sein.
Es hat sich aber auch herausgestellt, dass es sich nicht immer lohnt, aus allen möglichen Informationen zu lernen. So ist bei der Kommunikation der Agenten untereinander der große Aufwand in Hinsicht auf den Nutzen des Erlernten nicht zu rechtfertigen.

## 4.2 Aussicht auf weitere Entwicklungen

In der RoboCup Rescue Simulation League werden bereits Methoden des Maschinenlernens angewendet um die Agenten in verschiedenen Bereichen besser an die dynamische Umgebung anzupassen. Es wäre allerdings noch an einigen weiteren Stellen möglich, den Agenten bestimmte Verhaltensweisen anzutrainieren.

In der Simulation kommt es oft zu temporären Blockaden, wenn sich Einsatzkräfte verklemmen. Da die Blockade nur für eine gewisse unbekannte Zeit vorhanden ist, muss geschätzt werden, wann die Straße wieder frei ist. An dieser Stelle wäre denkbar die Agenten lernen zu lassen, welche Zeitspanne in welcher Situation angemessen ist. So kann der beste Wert zwischen unnötigem Warten auf das Lösen der Blockade und erneutem Auflaufen auf dieser herausgearbeitet werden.

Oft müssen die Einsatzkräfte entscheiden, ob sie den zu löschenden Feuerherd wechseln oder den aktuellen weiter bearbeiten. Bei einem Wechsel sind natürlich vor allem die Wegkosten und die Dringlichkeit des Löschens ausschlaggebend. Beim Durchlaufen diverser Simulationsläufen in einem Szenario wären die verschiedenen Entscheidungen beurteilbar und als Grundlage für das Erlernen einer Wechselstrategie gewinnbringend.

Eine große Menge unbekannter Blockaden bedeutet, dass die Agenten immer wieder in Sackgassen geraten und durch den verlängerten Weg viel Zeit verlieren. Allerdings nimmt auch mit intelligenten Verfahren das Erkunden des Gebietes einen nicht ungeringen Aufwand in Anspruch und sollte sich daher auf das Notwendigste konzentrieren. Hier gilt es, die beiden

Entgegen gesetzten  Ziele, möglichst wenig unbekannte Blockaden und möglichst wenig Erkundungsaufwand, zu optimieren.
Je nach Fortschritt der Simulation ändern sich die Ansprüche an die Erkundung, da zwar neue Barrikaden entstehen aber das Gebiet immer bekannter wird. Die Agenten könnten die aktuelle Situation durch Werte wie das Auftreten unbekannter Blockaden oder gefundenen Zivilisten beurteilen und daraus erlernen, wie das beste Verhältnis zwischen den Zielen zu erreichen ist.

# 5 Literatur

**Computational Intelligence** – A Logical Approach
Poole, Mackworth, Goebel
Oxford University Press
1998

**Expert Lexikon** – Künstliche Intelligenz
Hesse
Expert Verlag
1999

**Genetische Algorithmen und Evolutionsstrategien**
Schöneburg, Heinzmann, Feddersen
Addison-Wesley
1994

**Information Mining** – Methoden, Algorithmen und Anwendungen intelligenter Datenanalyse
Runkler
Vieweg
2000

**Mobile Agents** – Basic Concepts, Mobility Models & The Tracy Toolkit
Braun, Rossak
dpunkt.verlag
2005

**Wikipedia** – Die freie Enzyklopädie
http://www.wikipedia.de

**Techweb** – The Business Technology Network
http://www.techweb.com

# 6 Unterstützung

Mein Dank für die freundliche Unterstützung geht an:

**Cameron Skinner**
University of Auckland
Vorsitzender des technischen Komitees der „Robocup Rescue Simulation League“,
Hauptverantwortlicher des „Black Sheep“ Teams und Entwickler des Simulatorkerns

**Hans Guesgen**
University of Auckland
Mitglied des „Black Sheep“ Teams

**Moritz Goebelbecker**
Universität Freiburg
Mitglied des „Freiburg ResQ“ Teams und Entwickler von Simulatorkomponenten

**Michael Brenner**
Universität Freiburg
Mitglied des „Freiburg ResQ“ Teams und Entwickler von Simulatorkomponenten

**Alexander Kleiner**
Universität Freiburg
Mitglied des „Freiburg ResQ“ Teams und Entwickler von Simulatorkomponenten

**Prof. Dr. rer. nat. Kai von Luck**
Hochschule für Angewandte Wissenschaften, HAW
Abteilung Künstliche Intelligenz an der HAW und Betreuer der Studienarbeit

**Und viele andere „RoboCup Rescue Simulation League“ Teilnehmer aus diversen Teams, die mir einen Einblick in ihre Strategien ermöglicht haben und die über die Mailing Liste eine große Hilfe waren.**