

Evolving Dynamic Traffic Assignments

Andrés Arribas

Thesis submitted to

Obtain the degree of

Master of Science

In Artificial Intelligence

**Promoter:**

Prof. dr. Tom Holvoet.

**Advisors:**

Rutger. Claes

Rinde. Van Lon

**Academic Year 2012 - 2013**

© Copyright by K.U.Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promoter(en) als de auteur(s) is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot de K.U.Leuven, Faculteit Ingenieurswetenschappen - Kasteelpark Arenberg 1, B-3001 Heverlee (België). Telefoon +32-16-32 13 50 & Fax. +32-16-32 19 88.

Voorafgaande schriftelijke toestemming van de promoter(en) is eveneens vereist voor het aanwenden van de in dit afstudeerwerk beschreven (originele) methoden, producten, schakelingen en programma’s voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

© Copyright by K.U.Leuven

Without written permission of the supervisor(s) and the authors it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilize parts of this publication should be addressed to K.U.Leuven, Faculty of Engineering - Kasteelpark Arenberg 1, B-3001 Heverlee (Belgium). Telephone +32-16-32 13 50 & Fax. +32-16-32 19 88.

A written permission of the supervisor(s) is also required to use the methods, products, schematics and programs described in this work for industrial or commercial use, and for submitting this publication in scientific contests.

Foreword

May this thesis be the high note with which this challenging but rewarding year at KU Leuven comes to a close.

I would like to express my gratitude to my daily advisors Rutger Claes and Rinde van Lon, for their support and clarity of ideas. My appreciation too, to Willem Himpe who was generous with his time and explained to me the most important traffic concepts and the details of the original simulator.

Finally, I would like to dedicate this work to my family in Spain, always supportive despite the distance and to Judith, my partner in crime, the one with whom I ride the risks of life.

Andrés Arribas

Table of Contents (to update: right-click in list 🡪 ‘update field’)

Foreword iii

Table of Contents (to update: right-click in list 🡪 ‘update field’) v

Abstract vii

List of figures and tables viii

List of figures (to update: right-click in list 🡪 ‘update field’) viii

List of tables viii

List of abbreviations and symbols ix

Chapter 1: Introduction 1

Chapter 2: Traffic Theory 2

2.1 Traffic Systems Modelling 2

2.2 Demand Model 3

2.2 Network Flow Theory 3

2.2.1 Network Structure 3

2.2.1 Flows 4

2.2.1 Performance Variables and Transportation Costs 4

2.2 Traffic Flow Theory 5

2.2.1 Dynamic Network Loading 6

2.2.1 Dynamic Traffic Assignment 9

2.2.1 Flows 10

blabla 10

2.2.1 Performance Variables 10

2.3 Conclusion of this chapter 10

Chapter 3: A new chapter 12

3.1 First subject in this chapter 12

3.1.1 An item 12

3.2 Second subject in this chapter 12

3.3 Conclusion of this chapter 12

Chapter 4: The final chapter 14

4.1 First subject in this chapter 14

4.1.1 An item 14

4.2 Second subject in this chapter 14

4.3 Conclusion of this chapter 15

Chapter 5: Conclusion 16

Appendices 17

Appendix A: The first appendix 18

Appendix B: The final appendix 19

Bibliography 20

Master’s Thesis file 21

Abstract

Maximum 1 page

List of figures and tables

List of figures (to update: right-click in list 🡪 ‘update field’)

Figure 2 castle 1

Figure 4 airplane 13

Figure 6 www etc 14

List of tables

Table 1 This is the first table containing data 10

Table 2 A second table 11

List of abbreviations and symbols

π the number pi

∞ infinity …

Chapter 1: Introduction

Simulation of traffic propagation and its impact over a traffic network is a powerful and common tool to study default user behaviour and impact of events and changes to the network.

A simulation of dynamic traffic, i.e., in which a user departing at a given time may influence users departing at different times, is typically considered realistic if it has reached Dynamic User Equilibrium (DUE). DUE states that no vehicle may be able to reach its destination faster by choosing an alternative route. The DUE is based on the assumption that users are rational and take their decisions based on experience.

The Dynamic Traffic Assignment (DTA) is the process at the hearth of such simulations. More specifically, the DTA is an iterative process that incorporates a route assignment step and a step computing traffic propagation and network performance (network loading). In essence, the route assignment splits the units of travel demand (vehicles) according to their departure times, and assigns them to pre-computed routes connecting their origin and destination.

In complex traffic networks the DTA rarely attains equilibrium in the first iteration. In practice, given the lack of an exact formal solution, a heuristic, which is a computationally efficient approximation algorithm, that does, however, not guarantee optimality, is used to perform the route assignment towards equilibrium. The speed of convergence of the DTA towards DUE depends, first and foremost, on this heuristic.

Despite a number of attempts at designing such assignment heuristics, two of which are considered in this work, it has proven difficult to produce by hand efficient heuristics due to the complexity of the problem.

The objectives in this thesis are, first, to consider the adequacy of using a particular automatic computer-based method to generate such heuristics: genetic programming, and, secondly, by using genetic programming, attempt to ameliorate existing manual heuristics.

Genetic programming is a technique that mimics natural selection to evolve computer programs. Given that many problems can be reformulated as requiring the search for a computer program, genetic programming has been applied to many problems, often with success. Nevertheless, genetic programming has never been tried on this specific problem in traffic.

The rest of this thesis is organised as follows: chapter 2 covers the essentials of traffic theory including demand and supply modelling, Dynamic Network Loading, the DTA and route assignment; chapter 3 covers genetic programming by first introducing genetic algorithms, and then focusing on genetic programming specifics; chapter 4 is a detailed account of all performed experiments; chapter 5 focuses on the results of the experiments; and finally chapter 6 concludes and proposes further research steps.

BLA BLA

Example of an embedded image:

(once inserted 🡪 right-click ‘insert caption’: choose between figure or table 🡪 ‘Automatically update list of figures and tables on page viii: right-click: update field)



Figure 1 castle

Chapter 2: Traffic Theory[[1]](#footnote-1)

In this chapter basics of traffic theory are explained. The objective is to serve as an introduction to the most important concepts directly used in this thesis. This chapter is intended at readers with little or no background in Traffic theory. Out of scope remains to give a detailed picture of the field.

2.1 Traffic Systems Modelling

Transportation systems can be studied from a variety of angles e.g. economic, social, purely mathematical, etc., and at different scales e.g. regional, local or state wise.

In this thesis we are only concerned with a simplified view of what a transportation system is: we will focus on simulating simple small traffic networks and measuring their efficiency in terms of performance (e.g. travel times) given predefined fictional travel demands.

In order to simulate a traffic system, the first step is to model it.

A traffic model is a mathematical representation of the physical and organisational elements comprising a traffic network, as well as the travel demand and the emerging interactions.

Even the simplest traffic models have to take into consideration the inherent complexity of transportation systems, which arises from the multiplicity of non-linear interactions and feedback cycles.

In order to do so, a traffic model is most commonly defined by means of a demand model plus a combination of a traffic flow model and network model. The former is used to analyse and simulate the performances of the main supply elements, the latter to represent the topological and functional structure of a system.

The subset of traffic flow theory used in this work is a very specific one that requires careful consideration. But first, let us briefly summarize the demand representation to be used and the fundamentals of network/supply models.

2.2 Demand Model

The number of users consuming particular services offered by a transportation system in a given time period is the travel-demand flow. Travel demand flows, result from the aggregation of individual trips made in the area of study during the period of reference. A trip is defined as the act of moving from an origin to a destination.

The spatial characterization of trips is made by grouping them by place of origin and destination; accordingly demand flows can be arranged in tables commonly referred to as O-D matrices, whose rows correspond to the different origin and destination zones, respectively (SEE FIG AND INSERT FIG P 16).

A cell [ins dod] in the matrix gives the number of trips in the reference period from origin o to destination d.

The accumulated number of trips with origin o is therefore:

[FORMULAS IN P15 MAKE IT CONSISTENT WITH THE REST OF THE THESIS DOC]

And the accumulated number of trips attracted by destination d is:

[FORMULAS IN P15 MAKE IT CONSISTENT WITH THE REST OF THE THESIS DOC]

2.2 Supply Model

In this section we look at the foundations of congested network models.

2.2.1 Network Structure

A network structure is represented by a directed graph. A directed graph consists of a set of nodes, N, and a set of connections between pairs of nodes, L, called links, such that L SYMBOL NxN. In a directed graph links are oriented.

A link does not necessarily correspond to a construction in the physical world. Links rather represent phases and/or activities of possible trips between different traffic zones. The core idea behind a link is that its physical and functional characteristics can be assumed to be homogeneous for the whole link. In this sense, links can be seen as the partition of trips into segments, each of which has certain characteristics.

Nodes correspond to significant events delimiting trip phases. A node can represent the same event occurring at different time instants (between two trip phases). For example, the different entry or exit times in a road segment.

A trip is a sequence of several phases/links called a path. A path is defined from an initial node, the origin, to a final node, the destination. Each path is unambiguously associated with one, and only one O-D pair, whereas several paths can connect the same O-D pair.

[ref and pic p 47]

2.2.1 Flows

An instantaneous link flow is the instantaneous number of units using a link (i.e. units in that phase of the trip at that time instant).

PB with the previous and time [AAN]

[REV POSSIBLE FORMULAS]

Typically, when computing the link flow, the difference between types of units has to be taken into account, however, in this work, we consider all units to be of the same homogenous type.

A path flow is obtained as the sum of the link flows for all links in the path, considering in the flow calculations only those units matching the origin and destination of the path.

2.2.1 Performance Variables and Transportation Costs

Some variables perceived by the users can be associated to individual trip phases (hence links). Examples of such variables are travel times, monetary cost, and discomfort. These variables are referred as level-of-service or performance attributes. In general, performance variables correspond to disutilities or costs for the users.

Given our limited scope and simplified approach we focus on travel times as the only performance variable. Accordingly, from now onwards, “travel time”, “cost” and “performance variable/indicator” may be used indistinctly.

Remark that a cost will reflect not only the assigned physical characteristics of the link at hand (e.g. length, free speed), but also the interaction of the units in the link. In particular, the phenomenon of congestion, which we define precisely in the next section, will have a significant impact on the cost.

Similarly to the flows, a path cost can be computed as the sum of the cost of all links in the path.

2.2 Traffic Flow Model

Usually traffic flow models can be split in stationary models, where there is no variance in time of variables, and dynamic models, where there is variance in time.

Stationarity could only be observed in the real work if demand, path choices and supply system remained constant for a very long period of time. In less than a day of time, this is not the case and the behaviour of the traffic system does not only depend on its characteristics but also on its history.

Hence, we are interested in dynamic models, that, contrary to stationary models, allow the effective, realistic, simulation of important phenomena, especially the creation, propagation and dissipation of queues.

There are several approaches to dynamic traffic modelling (also called “within-day dynamics) depending on the scale at which units are considered e.g. continuous fluid analogy, packet based, or individual units. We are concerned with so called “Macroscopic” models.

Macroscopic models are based on the analogy with fluid dynamics. In such a models the individual vehicles are treated as a continuous[[2]](#footnote-2) (one-dimensional) fluid for which variables such as flow, density and velocity are defined at each point in space and time. Macroscopic models comprise equations for the conservation of mass (vehicles) and the relation between flow and density.

More specifically, we are interested in “Space-discrete” Macroscopic models for which it is assumed that basic variables affecting link performance, such as density or speed, do not vary along the link.

A fundamental aspect to take into account given dynamicity, is that the number of users within a link depends now on the travel times required to reach that link, which, in turn, depends on the number of users encountered on the network links in the previous instants. This circular dependency between flow and link performance, illustrated in fig [FIG p 435 Cascetta], does not allow the resolution of the two models (flow propagation an link performance) in a sequential fashion. This problem is known as Dynamic Network Loading.

2.2.1 Dynamic Network Loading

The DNL model is responsible for propagating path demands through the network. The traffic flows are calculated in time and space, congestion may form, delays may be encountered and the travel times on the network are determined.

The DNL we use is the Link Transmission Model (LTM)

Whereas a mastery of the LTM or even in depth knowledge of it are not a requirement for this Master Thesis it is desirable to look at a brief introduction to the LTM which will also serve as a means of covering some key aspects of traffic flow theory worth keeping in mind when assessing the following sections and the final results.

2.2.1 Link Transmission Model

The Link Transmission Model is a DNL model proposed by Yperman at KU Leuven in 2007 [BIB].

Amongst its most significant characteristics we may count a high realism of representation of congestion formation (whenever the traffic in a link surpasses its free flow capacity) and spillback (backwards propagation of the effect of congestion) as well as computational efficiency.

We will now discuss firstly, the concept of Cumulative Vehicle Numbers, which forms the basis of the calculations in the LTM, and secondly, the involved basics of traffic propagation.

Many details of traffic theory of no interest to this work have been omitted.

2.2.1 Cumulative Vehicle Numbers

The aggregation of vehicles having passed a location x by time t is the Cumulative Vehicle Number, or CVN, denoted by *N(x,t).*

In the LTM units are assumed to follow a predefined route between origin and destination creating different streams of traffic given different origins and destinations.

For a particular link, CVN in the LTM are calculated at all upstream and downstream link boundaries. Through time the CVN, given that the locations of measurement are fixed, will draw a shape through time. Additionally, CVN curves are also computed for complete streams at origin and destination.

Once all the curves have being obtained different variables can be derived:

* Link travel times can be derived as the horizontal distance between the curve at the upstream boundary N(x0,t), and the curve at the downstream boundary N(xL,t). Analogously, total travel time for a path is the horizontal difference between CVN at origin and CVN ay destination.
* The total number of vehicles in a link is the vertical distance between upstream and downstream CVN curves.
* The flow q (veh/h) is simply the slope of the CVN curve.
* The density k (veh/km) on a link can easily be calculated by dividing the number of vehicles N by the length L.

2.2.1 Traffic Propagation

In the LTM, the “fundamental triangular diagram” is considered an accurate representation of the propagation of traffic in unidirectional links. The diagram itself is just a plot of the flow in relation to the density. What is fundamental is the set of assumptions and definitions behind the diagram.

Below the critical density kc, which corresponds to the link’s capacity C, vehicles are assumed to travel with a fixed free speed vf (km/h) characteristic of the link.

Above the critical density kc , hence for congested traffic, the speed in the link v is given by q/k.

The maximum density is defined as the jam density, kjam, and corresponds to vehicles standing still.

Different traffic states are separated by shock waves that may propagate up- or downstream with a speed ws, that can be derived as the slope of the connecting line between the adjoining traffic states.

The maximum negative shockwave speed w is the fastest possible speed with which congestion may spill back in the upstream direction.

[intro in the text references –as in the figure-]

[INSERT FIG OF FUNDAMENTAL DIAGRAM]

In this thesis it suffices to know that all the above can be computed for a particular time step from CVN curves (including curves produced in previous time steps).

Finally, note that the above is a description of the propagation of traffic in a link.

The LTM may use different models to describe intersections i.e. propagation of incoming flows to outgoing links. These are out of scope; suffice to say, nevertheless, that once the flows are computed the CVN per link can be easily updated.

2.2.1 Dynamic Traffic Assignment

But, does DNL account for a realistic simulation of traffic? No, it does not on its own. DNL serves as a means of computing the model response for an a-priori assignment of traffic. That is, given that the path each vehicle will follow is known, we may use DNL to compute flow propagation and link performance. From the DNL point of view, it is not relevant whether the assignment reflects or does not reflect appropriate route choices.

It is the task of so called Dynamic Traffic Assignment (DTA) Models to, in addition, consider assignment accuracy and henceforth reproduce more adequately system dynamics within the period of simulation.

2.2.1 Introduction

Dynamic Traffic Assignments are used to study user route choice and propagation of traffic in time, including the reaction to changes and the impact on other users of congestion and driver’s decisions.

DTA models are mainly made of four sub-models, the demand model, the supply model, the DNL and the route choice model. We have already seen the first three; we will focus on the route choic.

Route choice, also called traffic assignment and traffic swapping, determines the division of traffic to alternative routes for every origin-destination pair, by entry time to the network. The final aim of the route choice is to attain user equilibrium, commonly called Dynamic User Equilibrium (DUE), stating that the cost of any route being used is not higher than the travel cost of any alternative route. In the dynamic case this condition must hold for all times. Analogously, no user can improve his situation (in terms of cost) by transferring to an alternate route, under the assumption that the users are rational and will chose the shortest route to their destination.

The general formulation of the DUE requires consideration of continuous time [INSERT T]. However, we are interested in practical applications, which require discretization of time into intervals of the form [INSERT FORM].

The cost or travel time is designated by [INSERT]. The instantaneous route demand (veh/h) at a time [insert t] at the entry of route [insert r] is denoted by [drt]. The total instantaneous demand assigned to any alternative route from origin [p] to destination [q] at time [t] is denoted by [Tpqt], hence:

[insert formula]

where [zpq] is the set of alternative routes between origin [p] and destination [q]. The general DUE condition in continuous time is:

[ins formula]

where [C\*] indicates the minimal travel cost from origin [p] to destination [q] a time t. That is to say, at every point in time the travel cost for each route in use [crt], is equal to the minimal cost of the origin-destination pair, and not higher than the travel cost of any alternative route.

In the discrete version we assume that the demands are constant within each interval, so … [copy phrase from p3 article].Therefore,

[insert updated formula]

Given that travel times are not really constant a key dilemma in the formulation of the DUE is which travel time should be associated with an interval of departure times. The most computationally stable option has been proven to associate with each interval the travel time at the end of the interval i.e [intro c= c. ].

In addition, route choice is often modelled as being the product of user experience. User experience is limited by the users perception of the traffic situation. In order to model the sensitivity of the user to traffic changes, the route choice algorithm may consider the users impervious to changes to the traffic cost for a period of time [def of t prime] This leads us to an alternative formulation of the DUE condition:

[insert my versoin of eq 4.]

The excess cost of a single route, denoted by λr(τ), is defined as the difference between the travel cost on route r and the minimal travel cost from origin p to destination q. So that,

ECr(τ)=λr(τ)=Cr(τ)−C∗pq(τ), ∀r∈Zpq,∀pq,∀τ.

(5) Total excess cost of traffic travelling on the route r, denoted by ρr(τ), is

TECr(τ) = ρr(τ) = dr(τ)·ECr(τ), ∀r ∈ Zpq,∀pq,∀τ. (6)

In perfect equilibrium the TEC of all routes should be zero. Convergence is mea- sured by the Global Excess Cost (GEC), which is the summation of the total excess travel costs for all departure times, in all the alternative routes and for all origin- destination pairs.

GEC=∑ ∑ ∑TECr(τ).

Route Choice Algorithms

The route choice algorithms we are concerned with follow an iterative process in which the assigned demand is updated between iterations according to a correction factor.

This can described as follows:

[insert func]

where F is a correction function suggesting a change to the demand flow matrix [d] containing the demands for all routes and all departure intervals. The suggested change is dependent on the demand flow matrix [dj-1] of the previous iteration. The solution [dj] for iteration j is the weighted average of the proposed solution [Fdj] in iteration j and of the solution [dj-1] attained in iteration [j-1] , where [aj] represents the step size in iteration j.

Analogously, one can directly consider the correction function that expresses the proposed change matrix [lj] in iteration j. This correction matches the difference between the proposed solution in iteration j and the actual solution in iteration j-1.

[insert func]

In traffic dynamics applying a change to a demand assignment for a route may affect, not only traffic departing at later times on the same route, but also already departed traffic on different routes. As a consequence, attaining the DUE requires the iterative process to compute a full DNL solution per iteration.

Notice that as a requisite to the usage of the DNL lays the need to specify the paths in a network. For small networks all possible paths may be enumerated a-priori, but with bigger networks and more realistic scenarios this becomes impractical very fast. Henceforth a pathfinder component is required.

Path finding is very costly computationally and as such is often applied incrementally. When computing inputs to the DNL for the first time, potential shortest routes are computed by the pathfinder based on costs a priori (assuming no congestion). Once a DNL solution has been obtained the new shortest routes according to the new costs are computed (and the GEC is calculated), if these differ from the existing ones they are added to the pool of potential paths.

The route choice algorithm uses the complete pool of paths in order to assign the demand to different routes with the DUE as an objective.

Given comparable DNL models and pathfinders, the route choice algorithm is the component that influences the most the speed of convergence of the DTA. Finding an efficient route choice algorithm is therefore a key challenge. We will know look at two of the existing simple methods.

[FIG TO BE ADDED]

Fig [] depicts the complete iterative process.

MSA

MSA, or method of successive averages, is a very popular route choice numerical method applied to the DUE problem.

MSA works by proposing an all or nothing assignment then…

DEC

Conclusion

Amongst the possible angles transportation systems can be studied from we are only concerned with the simulation of simple traffic networks. A simulation is considered to reflect a realistic traffic situation if the Dynamic User Equilibrium (DUE) has been attained. In practice we may attain the DUE through a Dynamic Traffic Assignment (DTA) process. A demand model, a supply model, a Dynamic Network Loading (DNL) model, and a route choice/assignment model are all components of the DTA. The DTA is an iterative process that successively run the DNL for a pre-defined demand and supply, then computes the distance to the DUE and adjusts the route assignment to, once again, run the DNL. Route assignment heuristics are central to this thesis. We have seen two: MSA and DEC. In the next chapter we will introduce genetic programming, which may prove useful into searching automatically for new heuristics.

Genetic Programming

//In this chapter…

Introduction

A central idea of genetic algorithms is that they are not based on the traditional correct, consistent, justifiable, certain, orderly, parsimonious, and decisive principles of the vast majority of techniques in machine learning. Rather than seeking an exact mathematical reasoning that leads us to the solution, genetic algorithms are based on entirely new principles, namely on applying the same kind of uncertain and non-deterministic evolutionary process observed in nature, where diversity is often wanted and therefore inconsistency and contradiction are considered of significant value.

Genetic programming (GP) is in fact a type of genetic algorithm (GA), which focuses on the evolution of computer programs. In the subsequent sections we will have a look at the basic of what a genetic algorithm is, then, we will re-interpret the genetic algorithm paradigm under the light of genetic programming, and finally we will consider some of the details and practicalities of genetic programming itself.

Genetic Algorithms

In nature, given a population of self-reproducing entities, diversity amongst them, and a connection between the entities diversity and their survival ability, we observe a tendency for the fittest individuals to reproduce more frequently, as well as a higher rate of survival of their genetic information. Through time this process reshapes the structure of the population towards a population dominated by fitter individuals. We say that the population has then evolved.

John Holland proposed in his book “Adaptation in Natural and Artificial Systems” to extend to artificial systems this sort of evolutionary process ruled by natural selection. The kind of algorithms that solve adaptation problems formulated in genetic terms is what we call a “genetic algorithm”.

A genetic algorithm is typically a highly parallel algorithm, which will transform a population of mathematical objects that we call individuals, each with an associated fitness value, into a new population using operations based on the principles of survival of the fitness and sexual reproduction.

Briefly put and in plain terms a typical genetic algorithm proceeds as follows:

It begins with an effort to learn something about the environment by testing a number of M randomly chosen points in the search space, equivalent to generating M individuals randomly. This is generation 0.

Per generation the fitness of each individual is assessed against the environment. Depending on the fitness of individuals and a probabilistic parameter fixed a-priori [pr], a pool of individuals is created by reproducing (copying) individuals based on a criterion that favours the fittest ones. Individuals in this pool are then recombined into new individuals (mixing their parts to produce 2 individuals), this operation, which also favours fittest individuals and uses a new parameter [pc], is called crossover and it allows exploring new points in the search space by replacing parents with their fitter offspring. An additional operation of mutation may uniformly computed per individual with parameter [pm].

The iterative repetition of the previous steps has the potential to improve the average fitness of the population. The best individual encountered, often labelled best-so-far is our solution. Once the optimal solution has been reached or a maximum number of generations has been reached we have attained the termination criterion and the evolutionary process can be stopped.

More formally:

1) The preparation of a genetic algorithm requires:

The selection of a representation scheme

A fitness measure

The parameters and variables controlling the algorithm

A way of designating the result

A termination criterion

The representation scheme is a mapping that allows expressing each possible point in the search space. A representation that facilitates the search for a solution by means of the genetic algorithm requires significant insight into the problem domain, for example, in the travelling salesman problem, or TSP, a possible solution or path may be represented by enumeration of all nodes: 1-3-4-6-5.

The fitness measure must ensure the evaluation of all possible individuals. It is often inherent to the problem, for example, in the TSP the fitness may be the length of the path.

The primary parameters for controlling a genetic algorithm are the population size (M) and the maximum number of generations to be run (G). Secondary parameters are [pr], [pc] and [pm]. Other control variables are often required depending on the specifics of the GA.

The most common way of designating the result is obviously to take the best individual so far.

A run may be terminated according to different criteria. The most frequent criteria are encountering the best possible result and reaching the maximum number of generations.

2) Once the preparation is complete the genetic algorithm can be run as follows:

Randomly crate an initial population of individuals.

Iteratively perform the following substeps on the population until the termination criterion has been satisfied

Evaluate the fitness of each individual in the population

Create a new population by applying at least the first two of the following operations, on individuals chose with a probability based on the fitness:

Copy existing individuals to the new population

Create two new individuals by genetically recombining two existing individuals

Create a new individual by randomly mutating an existing individual

The best-so-far individual is then designated as the result of the genetic algorithm for the run.

This process is illustrated in figure [ins num]

[INSERT FLOWCHART]

There are numerous variations on the basic genetic algorithm but this is the most frequent setting. It is important to understand that domain specific knowledge may help substantially into taking the best choices during the preparation step, which will in turn improve the performance of the genetic algorithm. However, the main point is that the genetic algorithm is a domain-independent approach to rapidly searching an unknown search space for good solutions.

For further explanations, particularly regarding why genetic algorithms work, the reader is referred to [insert Koza book and chapter 3.2].

Overview of Genetic Programming

Genetic programming arises from the observation that numerous problems in machine learning, artificial intelligence and symbolic processing can be interpreted as requiring the discovery of a computer program that provides a certain output given a certain input.

The discovery of such a computer program can be reinterpreted as the search for the fittest computer program in a space of computer programs. As we have seen such a search can be achieved through the principles of survival of the fittest, reproduction and crossover.

The leap between basic genetic algorithms and genetic programming lies at the representation level. The type of structure undergoing adaptation is now a hierarchy of functions and terminals of varying size and shape. In addition, such computer programs are now “active” in the sense that they are capable of being executed.

Genetic programming is therefore a domain-independent method that provides a single unified approach to the problem of finding a computer program to solve a problem.

The execution of a genetic algorithm under the genetic programming paradigm becomes:

Randomly crate an initial population of random compositions of the functions and terminals of the problem (computer programs)

Iteratively perform the following substeps on the population until the termination criterion has been satisfied

Execute each program in the population and assign it a fitness value according to how well it solves the problem.

Create a new population by applying at least the first two of the following operations, on computer programs in the population chosen with a probability based on their fitness:

Copy existing programs to the new population

Create new computer programs by genetically recombining two existing individuals

Create a new individual by randomly mutating an existing individual

The best-so-far computer program is then designated as the result of the genetic algorithm for the run.

Genetic Programming Details

As in Koza’s book we will describe in more detail genetic programming focusing on John Holland’s key features common to tall adaptive systems, one at a time.

These key features are:

Structures that undergo adaptation

Initial structures

Fitness measure

Operations that modify the structures

State of the system at each stage

Termination method

Method to designate the result

Parameters that control de process

The following explanations cover all that is needed in order to understand this thesis, not more.

Structures that undergo adaptation

As already introduced the structures that undergo adaptation are computer programs. More precisely hierarchical structures made of functions and terminals.

The functions may be:

Arithmetic operations, e.g. +,-,\*

Mathematical functions, e.g. sin, exp

Boolean operations, e.g. and, not

Conditional operators, e.g. if, then

Functions causing iteration, e.g. loop

Functions causing recursion and

Domain-specific functions that may be defined

The terminals are atomic entities, i.e., variables, constants or functions taking no arguments.

A common representation of these structures is the s-expression used in the LISP language, for example (OR (AND (NOT d0) (NOT d1) ) (AND d0 d1)) which corresponds to the tree diagram in figure [INS FIG REF].

[INSERT FIGURE P 81]

Initial structures

The initial structure in genetic programming consists of the individuals in the initial population of individual computer programs for the problem.

The generation of each initial computer program is done by randomly generating a tree structure made of functions and terminals.

In practice there are three dominant techniques to do this: full method, grow method and the ramped half-and-half.

The full method involves generating trees with a fixed length, in which all branches from root to the endpoint have same length. On the other hand, the grow method produces trees that are variably shaped, but not deeper than a maximum length.

Ramped half-and-half is a method in which the population is split in two parts, often in half but not necessarily. The full method is then applied to one of the parts, whereas the grow method is applied to the other part.

Fitness measure:

The most common approach is to define an explicit measure of fitness for each individual. This is accomplished via an explicit well-defined evaluation procedure producing a scalar fitness value as output.

We are interested in 3 interpretations of the fitness value:

Raw fitness: r, the fitness as normally measured in the problem domain.

Standardized fitness: raw fitness manipulated so as to ensure that a lower fitness coincides with a better individual. In particular if the raw fitness increases for worse individuals s = r.

Adjusted fitness: Defined as [INS EQUATION]. Adjusted fitness lies between 0 and 1. The adjusted fitness is bigger for better individuals in the population.

Operations that modify the structures:

We already know that the primary operations in genetic programming are reproduction and crossover.

Reproduction is the basic engine of survival of the fittest. It consists of two steps. First and individual is selected. Second, the individual is copied, without alteration, from the current population into the new population (i.e., the new generation).

A popular method of selection is tournament selection, in which a specified group of individuals are chosen at random from the current population and the one with the better fitness is then selected. Note that tournament selection is a selection strategy with replacement; therefore the same individual may be selected to become part of the next generation several times. Tournament selection has the particularity of adding additional control to the selection pressure by increasing or decreasing the size of the group of individuals chosen at random.

The crossover starts from two parental structures and produces an offspring structure. Typically both parents are chosen with the same selection method used in the reproduction step. Subsequently, a point in the structure of the both parents is selected with a uniform probability. The crossover fragment for a particular parent is the rooted subtree which has as its root the crossover point for that parent and which consists of the entire subtree lying below the crossover point.

The first offspring is then produced by deleting the crossover fragment of the second parent at the crossover point of the first parent. The second offspring is produced in a symmetric manner. See fig [ ins ref].

[Insert figures in this chapter as a single figure]

Mutation may be applied in addition to crossover and reproduction. However, it is considered a secondary operation because it does not substantially contribute to the evolutionary process. Its sole purpose is to potentially reintroduce diversity in a population that may be converging prematurely.

In mutation an individual is chosen randomly with the same or a similar selection method as in reproduction, then, a point in the structure of the individual is selected with random probability and the subtree rooted at that point is then replaced by a newly generated subtree. The depth of the inserted subtree is limited to a maximum parameter.

State of the system at each stage:

The current population gives the state of the system at any point in time. Control parameters (which may be dynamic in complex implementations) and best-so-far individual are commonly kept in memory too.

Termination Method:

As we have already discussed the most common termination criterion is to attain the maximum number of generations or to encounter a result that matches a given condition. Other criteria such as the loss of diversity (stagnation) may be part of the termination criteria.

Result:

Again, as already discussed the designated result is typically the best-so-far. An alternative is to focus on the best individual in the last generation. However, there is no guarantee that the best individual in the last generation is the overall best encountered result.

Parameters that control the process:

Values for the following control parameters must be defined in order to control the evolutionary process:

Size of the population M.

Maximum number of generations G

Probability of crossover pc

Probability of reproduction pr

Probability of mutation pm

Maximum depth for offspring after crossover

Maximum depth for individuals in the initial population

In addition, in qualitative terms:

A clear selection of the selection method must be provided,

the type of fitness used has to be explicitly set,

and the generative method for the initial population must also be clearly defined.

The optimal values for the control parameters are very hard to obtain. In fact, finding the optimal control parameters is in itself an optimisation problem. Hence, it is often empirically though experience that an approximation of the best values is inferred.

Conclusion

A genetic algorithm is a domain independent technique based on the principles of natural selection and sexual reproduction designed to find solutions to problems in artificial systems. Given the well-known execution flow and characteristics of a genetic algorithm, it is sufficient to give one conceptual step further to propose the search for computer programs as a powerful generic method to solve problems arriving to genetic programming. We have gone through the theoretical fundamentals of genetic programming; it is now the time to move from theory to practice and apply genetic programming to our problem of interest.

Experiments

In this chapter we will go through the experiments that are the hearth of the research undertaken in this thesis. We will, first, explain a number of common aspects, then, enumerate the details that are specific for each experiment.

Objectives and Overview

Our objectives are, to assess the adequacy of using genetic programming regarding route assignment heuristic generation, and to attempt to improve on existing manual heuristics.

The proposed method to achieve the target research objectives is to evolve a heuristic using genetic programming, then compare against MSA and DEC.

For this we devise two experiments: experiment one evolves a heuristic using a training set of traffic networks; the result of which is then tested against MSA and DEC using a test set of traffic networks under different assumptions. Experiment two is very similar. Again, a heuristic is evolved for a training set of traffic networks; this training set is in fact the original from experiment one, but modified taking into consideration the results of experiment one; the test set is also different but we measure once again against MSA and DEC under different assumptions.

In both cases, the main metric to consider when evaluating and comparing heuristics is the speed of convergence towards DUE.

Commonalities

Tools

The following tools are at our disposal:

A port to Java of a traffic simulator provided by Willem Himpe from KU Leuven. The object oriented re-interpretation of the original code in Matlab and its porting to Java was a substantial time consuming task in this thesis. The software development aspect however, is not the focus of this work; hence more information has been provided separately in [Appendix reference]. The simulator provides the capability to run a full DTA process, which embeds the LTM as DNL component, and Dynamic Dijkstra as pathfinder. The simulator has also de capability to receive and run a user defined route assignment.

ECJ by Sean Luc et al. from Goerge Mason University, and a wrapper provided by Rinde Van Lon from KU Leuven. ECJ is an evolutionary computation system. ECJ is a popular java framework for evolutionary computing and genetic programming in particular. Part of its popularity is due to its simplicity of use. ECJ, on the other hand, often requires the user to create prodigious amounts of separate files. The wrapper allows us to control directly with java code ECJ reducing the amount of extra files. In addition the wrapper allows the distribution of ECJ’s computations by integrating with the open-source java-based parallel computing framework JPPF.

Finally, a Java application integrating the previous tools into one; Developed specifically for this thesis, this application contains all the logic and control aspects behind the definition and execution of the experiments.

Control over the Evolved Heuristic

Genetic Programming is a domain-independent technique. However, in practice, if one wants to obtain valid programs in a realistic time frame, one has to, first, reduce the search space and, secondly, direct the search in such a way that mostly valid programs are visited.

In our case we direct the search by aiming at route assignment heuristics that are similar to DEC. More precisely, we aim for heuristics of the form:

[INSERT FORMULA] (take into account all routes)

Where delta [insert Symbol] in the case of routes that are not optimal/shortest for and OD pair, and only that delta, is provided by the program that we evolve. This implies that as in DEC, we allow delta to surpass the maximum and minimum demand for an OD pair on a route, but we explicitly limit the final change to the route to the maximum and minimum possible demand for that OD pair on the route.

This is a convenient setting because it allows the genetic programming to focus on changes to the non-optimal routes, while we ensure programmatically that the demand over all routes for an OD pair does not surpass the total demand for that OD pair.

In addition we reduce the search space by constraining the heuristics to those formed of the a reduced set functions and terminals.

Functions:

MUL, Multiplication: receives two scalars, returns their product.

DIV, Division: receives two scalars, returns the first one divided by the second one.

SUB, subtraction: receives two scalars, returns the first one minus the second one.

ADD, addition: receives two scalars, returns their addition.

IF4: receives 4 scalars, if the first value is greater than the second the third value is returned, otherwise the second.

Terminals:

Constants: 0 and 1.

Variables

Additional Control and Speed-Ups

First and foremost, we want to produce valid individuals as frequently as possible. In order to accomplish this, we add a double staged manipulation that increases the validity of the evolved programs; first, we compute all deltas proposed in the first iteration for all OD pairs, routes and time instants; Subsequently, we look at the prevailing sign for the deltas. The deltas are expected to be mostly positive. If not the case, all deltas are multiplied by -1 from the first iteration onwards. Secondly, if any of those initial deltas surpasses the maximum possible demand we divide all deltas by that value from the first iteration onwards.

Despite the previous, it is still very probable to generate invalid heuristics that we do not want to spend time evaluating. A set of additional control mechanisms has been devised to detect such undesired heuristics

Zero delta: If the deltas are all zero in the first iteration, the DTA is aborted.

Increasing GEC: if the GEC augments for more than 10 iterations the DTA is aborted.

GEC stalled: if the GEC does not change for more than 10 iterations the DTA is aborted.

Average gap stalled: if the GEP oscillates, but results in an steady GEC average over 5 iterations of the DTA, the DTA is aborted.

Components of the GP heuristic

The Eiffel Tower has three floors:

the first;

the second;

the third.

But do this:

The Eiffel Tower has three floors: the first, the second and the third.

2.2.1 Flows

blabla

2.2.1 Performance Variables

2.3 Conclusion of this chapter

If you have reached important findings or conclusions in this chapter, it is only logical that you should end the chapter by summarising them. This is not necessary for chapters such as the introduction or list of the cited literature.

|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| A | A1 | A2 |
| B | B1 | B2 |
| C | C1 | C2 |

Table 1 This is the first table containing data

Text

|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| 1 | 11 | 12 |

Table 2 A second table

Chapter 3: A new chapter

A chapter contains a cohesive[[3]](#footnote-3) whole that stands, more or less, on its own. It is therefore only logical that it should start with an introduction, i.e. that part of the text which you are now reading.

3.1 First subject in this chapter

Information introducing the subject.

3.1.1 An item

Text is never presented on its own. This means that references are bound to be needed. Reference can be made to online documents[2] or books[3].

3.2 Second subject in this chapter

A chapter will contain several subjects. Let us assume that this one is the last.

3.3 Conclusion of this chapter

If you have reached important findings or conclusions in this chapter, it is only logical that you should end the chapter by summarising them. This is not necessary for chapters such as the introduction or list of the cited literature.



Figure 2 Airplane

Chapter 4: The final chapter

A chapter contains a cohesive whole of information that stands, more or less, on its own. It is therefore only logical that it should start with an introduction, i.e. that part of the text which you are now reading.

4.1 First subject in this chapter

Information introducing the subject.

4.1.1 An item

The accompanying text. Remember to keep paragraphs long enough, but make sure the sentences are not too long.

A paragraph contains a train of thought and so will always contain a couple of sentences. Do not write a paragraph which consists of only one line.



Figure 3 www etc

4.2 Second subject in this chapter

A chapter will contain several subjects. Let us assume that this one is the last.

4.3 Conclusion of this chapter

If you have reached important findings or conclusions in this chapter, it is only logical that you should end the chapter by summarising them. This is not necessary for chapters such as the introduction or list of the cited literature.

Chapter 5: Conclusion

The master’s thesis is brought to a close with a chapter summarising all the conclusions once again. This is also the place to include suggestions on further use of the results, in both industrial applications and further research.

Appendices

Appendix A: The first appendix

The appendices contain information that is likely to be useful to the reader, but not essential to a sound understanding of the argument in the normal body of text. Examples include source files, configuration information, lengthy mathematical deductions, etc.

Needless to say, an appendix may be further divided into sections, or contain figures and references[1].

Appendix B: The final appendix

The appendices contain information that is likely to be useful to the reader, but not essential to a sound understanding of the argument in the normal body of text. Examples include source files, configuration information, lengthy mathematical deductions, etc.

Bibliography

[1] D. Adams, The Hitchhiker’s Guide to the Galaxy. Del Rey (reprint), 1995, no.

ISBN-13: 978-0345391803.

[2] Scriptie. Last reviewed: 31 March 2008. [Online].

Available at: <http://nl.wikipedia.org/wiki/Masterproef>

[3] T. Pratchett and N. Gaiman, Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch. HarperTorch (reprint), 2006, no. ISBN-13: 978-0060853983.

K.U.Leuven Faculty of Engineering 20xx-20xx

Master’s Thesis file

Students: First name Surname

Title: Title of master’s thesis

English title: English translation of the title

UDC:

Content in brief: (maximum 500 words)

(Title of the article: Title of the article)

(English translation: English translation of the title)

Thesis submitted to obtain the degree of Master in Engineering: ….....................................

Supervisor(s):

Assessors:

Mentor(s):

1. This chapter is essentially based on reformulated excerpts from Transportation Systems Analysis by Ennio Cascetta [↑](#footnote-ref-1)
2. Notice that despite the continuous fluid analogy the simulation and resolution of in Macroscopic models requires the discretization of time. [↑](#footnote-ref-2)
3. Insert footnote via References 🡪 insert footnote [↑](#footnote-ref-3)