Pursuit and Evasion

Bachelor Thesis

by

Felix Blumenberg
870605-0633
Fredrik Bberg
880312-0511
Mats Malmberg
860802-0338

under the guidance of

Doctoral Student Johan Thunberg
and
Professor, Ph.D. Xiaoming Hu

Department of mathematics
Division of Optimization & Systems Theory
Royal Institute of Technology KTH, Sweden
Feb 2011

Abstract

Abstract in ENGLISH

Sammanfattning

Abstract på SVENSKA

Contents

List of Figures

Chapter 1

Introduction

Giving a general overview of the report.

1.1 Organization Of the Report

- Chapter 2 explains the simulation environment we built.
- Chapter 3 contains a description of the algorithms we decided to use.
- Chapter 4 contains the results of our simulations.
- Chapter 5 contains a discussion of our results, comparison and conclusions.

1.2 Background

The objective of this section is to give a short theoretical background of optimization. Also a general description of the Pursuit and Evasion problem.

1.2.1 What is optimization

What is optimization?
What is an optimal solution?
Non-optimal solutions?
Good solutions and computational time?

1.2.2 (the need for) a near optimal solution

Giving a very short background for the following section.

1.2.3 P & NP problems

explain what P and NP-hard problems is, and what consequenses it yields.

1.2.4 Heuristic methods

Explanation of what heuristic methods is, and how one sacrifices optimality for gain in computational time.

Chapter 2

Problem formulation

The essentials of our problem formulation today is "To test implementation of greedy approach, tabu search and genetic programming and evaluation." The exact formulation is in progress...

2.1 Problem with the problem

the headline is self explanatory, the ide her is to discuss the P vs NP aspekt. maybe this will be incorporated with another part of the introduction

2.2 Approach

A description of our approach to the problem. Describing the overview of our approach. Mentioning the creation of an testing enviroment, choice and application of heuristics and target data to evaluate the effectiveness.

Chapter 3

Problem formulation

The essentials of our problem formulation today is "To test implementation of greedy approach, tabu search and genetic programming and evaluation." The exact formulation is in progress...

3.1 Problem with the problem

the headline is self explanatory, the ide her is to discuss the P vs NP aspekt. maybe this will be incorporated with another part of the introduction

3.2 Approach

A description of our approach to the problem. Describing the overview of our approach. Mentioning the creation of an testing enviroment, choice and application of heuristics and target data to evaluate the effectiveness.

Chapter 4

Simulation Environment

In order to attain the data needed for a comparison of our different algorithms it was nessecary to construct a good testing enviroment. We decided to create this enviroment by the use of two separate parts. One part is called the "Map generator". This part creates a map of the enviroment, tests the feasibility and prints feasible enviroments into an output file. The other part is called the "Network generator". This part reads in an enviroment from a file, creates a graph network to the corresponding map and gives each node in the network its relevant information.

4.1 Map generator

The Map Generator (MG) creates random feasible enviroments. A feasible enviroment is described more in detail in section 2, but in short one could say that an enviroment is feasible if it is simply connected and can be divided into a finite set of convex regions. Given the desired size and the density (percentage of obstacles per total area) as inputs, MG creates square shaped feasible enviroment with randomly placed obstacles and saves the map in an external file. For simplicity we have chosen to construct enviroments consisting only of square regions. We suggest that this does not result in a loss of generality since one can construct any feasible enviroment by a sufficiently fine meshing of squares.

pseudo-code, Map Generator:
input variables:
int size; //Specifies the width and height of the square matrix A.
int NumberOfEnv; // Specifies how many feasible enviroments to create.
int Obstacle; // Specifies the number of obstacles in percents, e.g. number of obstacles per total area of A.

while (created feasible enviroments is less than the variable NumberOfEnv)

Create a matrix A with all elements set to one;
in a random maner, turn elements in A to zero (corresponding to obstacles) until the desired amount of obstacles have been placed.
Test if A is connected
if A is connected write the matrix A to the file OK.out
if A is not connected write the matrix to a file NotOK.out

4.2 Network Generator

The Network Generator (NG) generates a node network from an environment matrix. Each node consists of its adjacent nodes, all the nodes visible from it and its current state. For input the NG uses an environment matrix that is either created by the MG or by hand.

Pseudo-code, Node Network Generator:

Input: Environment matrix A, from file "ok.txt"
Read environment matrix from file, call it A
Represent each element in A with a node in a matrix B
For each node in B:
Set name to its position in the matrix
Find and store all nodes that is adjacent to the current node
Set state of the node to 4
Find and store all visible nodes

To find all visible nodes, the following algorithm is used:

Add the current node to the list of visible nodes.
Find the maximum available nodes to the left and right from the current node.
Find the maximum number of nodes up and down from the current node.
Take a step up, add the current node to the list of visible nodes and repeat until either no more nodes are available or the maximum nodes upwards is reached.
Repeat for downwards.
Take a step to the left, repeat the three previous steps.
Repeat for right.

To find the visible nodes we start in the node that we wish to calculate visible nodes for. From there we step through each allowed node to the left, adding every node upwards that is allowed and at most the same number of moves upwards that the previous step allowed. By then repeating this for every node downwards, and finally repeating everything for every allowed node to the right, each node that the starting node can see

is discovered. Due to the choice of using square regions, limiting the number of allowed steps upwards and downwards for each step to the left and right guarantees that the set of visible nodes will be correct.

Chapter 5

Methods

A Description of the methods and algorithms we have used.

5.1 Greedy

Greedy

5.1.1 Description

A description of what Greedy is

5.1.2 Algorithm

A description of the algorithm

5.1.3 Implementation

A description of how the algorithm is implemented

5.1.4 Development process

How the development of the algorithm have proceeded.

5.2 Tabu

Tabu

5.2.1 Description

A description of what Tabu is

5.2.2 Algorithm

A description of the algorithm

5.2.3 Implementation

A description of how the algorithm is implemented

5.2.4 Development process

How the development of the algorithm have proceeded.

5.3 Genetic

5.3.1 Description

Genetic algorithms is based on the idea of evolution. Using a combination of reproduction, mutation and survival of the fittest a solution is generated.

5.3.2 Algorithm

A description of the algorithm

5.3.3 Implementation

A description of how the algorithm is implemented

5.3.4 Development process

How the development of the algorithm have proceeded.

Chapter 6

Results

Statistics, tables and a description of the tables. Also why we have chosen these tables etc.

Results for MILP evaluation could also be added here.

Chapter 7

Discusion

This chapter contains analysis of each algorithm, why it did or did not work, how it compares to the other algorithms and a conclusion. The main purpose is to present conclusions from the data presented in the previous chapter.

7.1 Analysis

An analysis of each algorithm, evaluation of why it did or did not work.

7.2 Comparsion and statistics

A comparsion between the algorithms, and perhaps also with MILP.

Chapter 8

Conclusion

A conclusion of our work, and future work.

Bibliography

[1] Abraham Silberschatz, Peter Baer Galvin, "Operating System Concept", Addison Wesley, Reading Massachusetts, USA, 1998

[2] John P. Hayes, "Computer Architecture and Organization", McGraw-Hill International Company, Singapore, 1988

[3] PVM 3 User Guide and Reference Manual, Edited by Al Gist, Oak Ridge National Laboratory, Engineering Physics and Mathematics Divison, Mathematical Science Section, Oak Ridge, Tennessee, USA, 1991

[4] PVM's HTTP Site, "http://www.epm.ornl.gov/pvm/"

[5] Brian W. Kernighan, Dennis M. Ritchie, "The C - Programming Language, (ANSI C Version)", Prentice-Hall of India Pvt. Ltd., New Delhi, 1998

[6] Thomas H. Corman, Charles E. Leiserson, Ronald L. Rivest, "Introduction to Algorithm", MIT Press, Cambridge, MA, USA, 1990

[7] Kenneth Hoffmann, Rey Kunze, " Linear Algebra", Prentice-Hall of India Pvt. Ltd., New Delhi, 1997

[8] G.H. Golub and C. F. Van Loan , " Matrix Computations", Third Edition. The Johns Hopkins University Press, Baltimore, 1996

[9] David A. Patterson, John L. Hennessy, "Computer Architecture, A Quantitative Approach", Morgan Kaufmann Publications Inc., San Mateo, California, USA, 1990

[10] Jack Dongarra, Iain Duff, Danny Sorensen, and Henk van der Vorst, Numerical Linear Algebra for High-Performance Computing",Society for Industrial and Applied Mathematics, Philadelphia, 1998