pgf **Institutionen fdatavetenskap**
Department of Computer and Information Science

Final thesis

Implementation and testing of a fpt algorithm for computing the $\widehat{}+$ heuristic

by

Lars Niclas Jonsson

DATEOFPUBLICATION

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Linkngs universitet
Institutionen fdatavetenskap

Final thesis

Implementation and testing of a fpt algorithm for computing the ⌢+ heuristic

by

Lars Niclas Jonsson

DATEOFPUBLICATION

Supervisor: Christer Bäckstrőm

Examiner: Christer Bäckstrőm

Abstract

§ This is the abstract. Yow! Are we in the perfect mood? Do you really think I would like not want to watch wrestling? Half a mind is a terrible thing to waste! Possibly your plans could have caused this. Now Im concentrating on a specific tank battle toward the end of World War II! I dont understand.

Ask me the DIFFERENCE between PHIL SILVERS and ALEXANDER HAIG!! Earlier you said nobody is buying your urine sample bottles? hy should I give to have this suntan to you?

Contents

1 Introduction **1**
1.1 Background and Motivation 1
1.1.1 Introduction 1
1.2 Problem definition 2
1.3 Limitations 4
1.4 Previous Work 4

2 Theory **5**
2.1 Definitons 5
2.1.1 SAS Planing instance 5
2.1.2 Transition Graphs 5
2.1.3 Causal Graphs 6
2.1.4 Tree decompostions 6
2.2 Heuristic planing 6
2.3 Fast Wordward 6

Bibliography **7**

Chapter 1

Introduction

This is a bachelor thesis written with the motivation to analyze an fpt algorithm for computing the *+ heuristic and compare its practical speed with ”name”.

Heuristic search functions are used in artificiell intelligens to estimate the distance between a state (node) and its closest solution in automated planning and scheduling.

1.1 Background and Motivation

1.1.1 Introduction

Heuristic function are used in automated planning and scheduling to estimate the distance from a state to a goal.

An example of a planning task could be to solve the famous problem Tower of Hanoi. In the game, there is n disks and 3 rods. All disks have a different size. In the start positions, all disks are stacked in non growing order on road 1(see **??**). To solve the problem, all disks has to be stacked on road number 3 in non increasing order. One is only allowed to lift one disk at the time and it is not allowed to stack a bigger disk on a smaller disk.

To transform this problem to a planing task, we can create a graph where the vertexes in the graph are all the different cases that can occur in the game. There is an edge (u,v) between the vertexes u and v if it iss possible to go from state u to state v in one move. In this problem, all edges are undirected since there is always possible to go back one step. This is not the general case in planing. An example of a graph(although this is not the graph that represent the cases in Tower of Hanoi) can be seen in figure 1.2.

A path from the initial state(the start position) to the state were all disks are at rod 3 is a solution to the problem. If there are n disks, the shortest path has the length 2n̂ - 1 which means that the planing problem require

exponential time to solve. This is the case for many planning problems and therefore, many planing problems are said to be untrackable.

Because of that, heuristic functions are used before an actual search algorithm to estimate how far a vertex/state is from a goal. A search algorithm that are ran after a heuristic algorithm can look on the vertexes that can be reach from the current state and go to the state that was estimated to be the closest one to the goal.

In figure 3, we see the graph with numbers in each vertex which are the estimated distance for each vertex. There are two vertex that has the value 0, which means that they are both a goal to the problem. If a graph has the value infinity, it means that the state is a dead end and no solution can be found from that state. Backtracking can be applied if a dead end has been reached.

In this thesis Christers algorithm [1] will calculate the delete relaxing heuristic (h+)[3] of a relaxed planing instance.

A relaxed problem is a problem where all negative effect for all action has been remove. An example of a delete negative effect could be a robot that moves from A to B. The negative effect here is that the robot will not be at A when it is at B. If delete relaxing would been applied here, we would say that the robot is both at A and B. With relaxed delete instances, tower of Hanoi can be solved in linear time[1].

A relaxed planing problem is NP-complete and therefore there is not so much hope to find a fast solution to the problem, therefor the values are usually just estimated. However, Christers algorithm is a FPT algorithm which means that the it can be calculated in polynomial speed with respect to the input size and expontial and in time seen to the number of paths in the graph that represent the planning problem.

1.2 Problem definition

Can Christer algortihm be used in practise?

Figure 1.1: Tower of Hanoi with three disks. Image source: http://www.nkonecny.com/blog/2011/11/25/towers-of-hanoi/

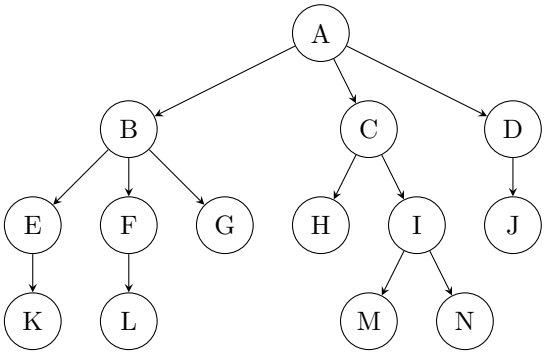


Figure 1.2: A graph were the intial state is A

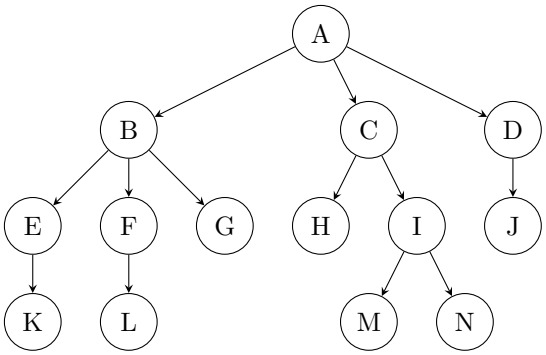


Figure 1.3: A graph were each vertex store a positive number which is the estimates distance from the goal.

1.3 Limitations

- We will only implement, execute and compare our algorithm on the planning system Fast Downward.
- Good in this case means two things: How fast the algortihm is in time and how many nodes it is visiting.
- Only binary varibels will be considered in this thesis.

1.4 Previous Work

I am not sure what to write here! Maybe other heuristic that use relax plaining?

Chapter 2

Theory

In this chapter will about the planning system Fast Downward(FD), Christers algorithm and some defintions like casual graph and tree decomponsition be explaind which is nessery for undersdanding the algorithm and how's the planner work

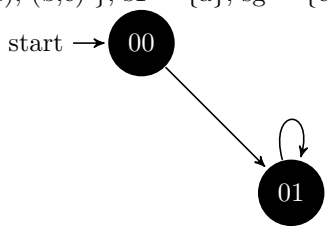
2.1 Definitons

2.1.1 SAS Planing instance

P = {V, D, A, s1, sg } is a SAS planning instance where V is the set of varibales, D(a) is the domain function for each v ∈ V, A is a pair of a precondition pre(a) and a efftect eff(a), s1 is the initial state and s2 is the goal. Since only binary variabels are used in this thesis, the domain for each v is {0,1} were v ∈ V.

2.1.2 Transition Graphs

A transition graph G = <S,E >is a directed graph were S is the **state space** for the planning instance P and e = (u, v) /*in* E if there is a action from u to v. The **state space** for V and D is S(V,D) = D(v1)*...*D(vn).
If P = {V, D, A, s1, sg } is a planning instance and V = {a,b}, A = {(a,b), (b,c), (b,c) }, s1 = {a}, sg = {c}, then its transition graph looks like:



2.1.3 Causal Graphs

-

2.1.4 Tree decompostions

A Tree decompostion of a Graph G = V,E is a pair N,T where N = { N1,N2,...,Nn} and Ni⊂V, and
NOT COMPLETE

2.2 Heuristic planing

The planning instance P in section 2.1 were very inpcomplex and therefore also its graph (figure 1) which makes it easy to solve. In reality, a planning instance can have 10000 varibales and actions which is not solvable today since planning is a NP-problem

2.3 Fast Wordward

Fast Wordward(FD) is a planning system that read PDDL-code as its input and calculate a solution to the planning problem the PDDL code descripe. When FD solves a problem, it goes thorugh 3 different phases(translation, knowledge compilation and search.

The first phase takes the PDDL-code as its input and return a multi-valued planning task. This phase takes the outpout from phase 1 as its input and generates four data structures: Domain transition graphs, Causal graph, the successor generator, and the axiom evualtor. In the last phase is the actual searching done. There are 3 search algortihm in FD, two of this are use a heuistic algorithm but the last one does not.

[**?**] [**?**] [**?**] [**?**]

Bibliography

[1] HL Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *Proceedings of the twenty-fifth annual ACM symposium ...*, 1993.

[2] M Helmert. The Fast Downward: Planning System. *J. Artif. Intell. Res.(JAIR)*, 2006.

Pvenska

Detta dokument hs tillgligt pnternet – eller dess framtida ersare – under en lre tid frpubliceringsdatum under ftsning att inga extra-ordin omstigheter uppst

Tillg till dokumentet innebtillst fvar och en att l, ladda ner, skriva ut enstaka kopior fenskilt bruk och att anva det ofndrat fickekommersiell forskning och fundervisning. rfng av upphovs-ren vid en senare tidpunkt kan inte upph detta tillst. All annan anvning av dokumentet krr upphovsmannens medgivande. Fatt garantera heten, srheten och tillgligheten finns det lingar av teknisk och administrativ art.

Upphovsmannens ideella r innefattar r att bli nd som upphovsman i den omfattning som god sed krr vid anvning av dokumentet pvan beskrivna s samt skydd mot att dokumentet ras eller presenteras i sn form eller i snt sammanhang som krande fupphovsmannens litter eller konstniga anseende eller egenart.

Fytterligare information om Linkng University Electronic Press se fagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet – or its possible replacement – for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linkng University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/