Marvin Abisrror Zarate

On Selecting of heuristics functions for Domain−Independent planning.

Brasil

2015, v-1.9.5

Marvin Abisrror Zarate

On Selecting of heuristics functions for Domain−Independent planning.

Paper presented to the Federal University of Viçosa, as part of the requirements of Graduate Computer Science program, for obtaining the title of Magister Scientiae.

Universidade de Viçosa – UFV

Centro de Ciencias Exactas e Tecnologicas (CCE)

Programa de Pós-Graduação

Supervisor: Levi Henrique Santana de Lelis

Co-supervisor: Santiago Franco

Brasil

2015, v-1.9.5

Marvin Abisrror Zarate
On Selecting of
heuristics functions for Domain−Independent planning./ Marvin Abisrror Zarate. –
Brasil, 2015, v-1.9.5-
67 p. : il. (algumas color.) ; 30 cm.

Supervisor: Levi Henrique Santana de Lelis

Tese (Mestrado) – Universidade de Viçosa – UFV
Centro de Ciencias Exactas e Tecnologicas (CCE)
Programa de Pós-Graduação, 2015, v-1.9.5.

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

Errata sheet

Elemento opcional da ABNT (2011, 4.2.1.2). Exemplo:

FERRIGNO, C. R. A. **Tratamento de neoplasias ósseas apendiculares com reimplantação de enxerto ósseo autólogo autoclavado associado ao plasma rico em plaquetas**: estudo crítico na cirurgia de preservação de membro em cães. 2011. 128 f. Tese (Livre-Docência) - Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo, São Paulo, 2011.

Folha	**Linha**	**Onde se lê**	**Leia-se**
1	10	auto-conclavo	autoconclavo

Marvin Abisrror Zarate

On Selecting of heuristics functions for Domain−Independent planning.

Paper presented to the Federal University of Viçosa, as part of the requirements of Graduate Computer Science program, for obtaining the title of Magister Scientiae.

Trabalho aprovado. Brasil, 24 de novembro de 2012:

Levi Henrique Santana de Lelis
Orientador

Professor
Convidado 1

Professor
Convidado 2

Brasil
2015, v-1.9.5

This thesis is dedicated to my Mother.

Acknowledgements

I would like to express my sincere gratitude to my advisor PhD. Levi Henrique Santana de Lelis, for the continuous support and guidance during the thesis process. His valuable advice, patience and encouragement have been of great importance for this work.

Besides my advisor, I would like to thank to my co−advisor: PhD. Santiago Franco for his insightful feedback, interest and tough questions.

To the professors of the DTI, particularly the Master Degree program with all its members, played an invaluable role in my graduate education.

Last, but not least, I would like to thank my Mother.

“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Abstract

In this dissertation we present a greedy method based on the theory of supermodular optimization for selecting a subset of heuristics functions from a large set of heuristics with the objective of reducing the running time of the search algorithms.

(HOLTE et al., 2006) showed that search can be faster if several smaller pattern databases are used instead of one large pattern database. We introduce a greedy method for selecting a subset of the most promising heuristicss from a large set of heuristics functions to guide the A* search algorithm. If the heuristics are consistent, our method selects a subset which is guaranteed to be near optimal with respect to the resulting A* search tree size. In addition to being consistent, if all heuristics have the same evaluation time, our subset is guaranteed to be near optimal with respect to the resulting A* running time. We implemented our method in Fast Downward and showed empirically that it produces heuristics which outperform the state of the art heuristics in the International Planning Competition benchmarks.

Key-words: Heuristics. selection.

List of Figures

List of Tables

Contents

Introduction

This thesis is concerned with cost−optimal state−space planning using the A* algorithm (HART P. E.; NILSSON; RAPHAEL, 1968). We assume that a pool, ζ, of hundreds or even thousands of heuristics is available, and that the final heuristic used to guide A*, h_{max}, will be defined as the maximum over a subset $\zeta^{'}$ of those heuristics ($h_{max}(s, \zeta^{'}) = max_{h \in \zeta^{'}} h(s)$). The choice of the subset $\zeta^{'}$ can hugely affect the efficiency of A*. For a given size N and planning task ∇, a subset containing N heuristics from ζ is optimal if no other subset containing N heuristics from ζ results in A* expanding fewer nodes when solving ∇.

Exists many problems of Artificial Intelligent (AI), such as: Finding the shortest path from one point to another in a game map, 8−tile−puzzle, Rubick's cube, etc. The level of difficulty to solve the problems mentioned are linked with the size of the search space generated.

State−space search algorithms have been used to solve the problems mentioned above. And in this dissertation we study the approach to solve problems in order to reduce the size of the search tree generated and the running time of the search algorithm using the best subset of heuristics selected from a large set of heuristics.

Part I

Preparation of the research

1 About the Problem

The purpose of this section if to motivate the problem.

1.1 Problem Statement and Motivation

Every problem of Artificial Intelligent can be cast as a state space problem. The state space is a set of states where each state represent a possible solution to the problem and each state is linked with other states if exists a function that goes from one state to another. In the search space there are many solutions that represent the same state, each of this solutions are called node. So, many nodes can be represented as one state. To find the solution of the problem is required the use of search algorithms such as: Depth First Search (`DFS`), which looks the solution of the problem traversing the search space exploring the nodes in each branch before backtracking up to find the solution. Another search algorithm is Breadth First Search (`BFS`), which looks for the solution exploring the neighbors nodes first, before moving to the next level of neighbors. The mentioned algorithms have the characteristic that when they do the search, they generate a larger search space. The search space that these algorithms generate are called Brute force search tree (`BFST`).

There are other types of algorithms called heuristics informed search, which are algorithms that requires the use of heuristics. The heuristic is the estimation of the distance for one node in the search tree to get to the near solution. The heuristic informed search generates smaller search tree in comparison to the `BFST`, because the heuristic guides the search exploring the nodes that are in the solution path and prunes the nodes which are not. Also, the use of heuristics reduce the running time of the search algorithm.

There are different approaches to create heuristics, such as: Pattern Databases (`PDBs`), Neural Network, and Genetic Algorithm. These systems that create heuristics receive the name of Heuristics Generators. And one of the approaches that have showed most successfull results in heuristic generation is the PDBs, which is memory-based heuristic functions obtained by abstracting away certain problem variables, so that the remaining problem ("pattern") is small enough to be solved optimally for every state by blind exhaustive search. The results stored in a table, represent a PDB for the original problem. The abstraction of the search space gives an admissible heuristic function, mapping states to lower bounds.

Exists many ways to take advantage of all the heuristics that can be created, for example: (HOLTE et al., 2006) showed that search can be faster if several smaller pattern databases are used instead of one large pattern database. In addition (DOMSHLAK; KARPAS; MARKOVITCH, 2010) and (TOLPIN et al., 2013) results showed that evaluating the heuristic lazily, only when they are essensial to a decision to be made in the search process is worthy in comparison to take the maximum of the set of heuristics. Then, using all the heuristics do not guarantees to solve the major number of problems in a limit time.

1.2 Aim and Objectives

1.2.1 Aim

The objective of this dissertation is to develop meta-reasoning approaches for selecting heuristics functions from a large set of heuristics with the goal of reducing the running time of the search algorithm employing these functions.

1.2.2 Objectives

- Demostrate that the problem of finding the optimal subset of ζ of size N for a given problem task is supermodular respect the size of the search tree.
- Develop an approaches to obtain the cardinality of the subsets of heuristics found.
- Develop an approach to find a subset of heuristics from a large pool of heuristics that optimize the number of nodes expanded in the process of search.
- Develop an approach for selecting a subset of heuristic functions based on the minimum evaluation cost of each heuristic.
- Develop an strategy to drop heuristics during the sampling that do not improve the objective function.
- Use Stratified Sampling (SS) algorithm for predicting the search tree size of Iterative-Deepening A* (IDA*). And use SS as our utility function.

1.3 Scope, Limitations, and Delimitations

We implemented our method in Fast Downward (HELMERT, 2006) and the problems we want to solve are the optimal domains benchmarks. Our meta$-$reasoning described in this thesis are going to try to solve the major number of problems using the most promising heuristics from a large set of heuristics. The exact way to create the large set of heuristics is beyond the scope of this thesis.

1.4 Justification

In the last few decades, Artificial Intelligence has made significant strides in domain−independent planning. The use of heuristics search approach have contribuited to problem solving, where the use of an appropriate heuristic often means substantial reduction in the time needed to solve hard problems.

That is why we propose a meta−reasoning that will try to solve the major number of problems without relying on domain knowledge, to guide the A* search algorithm.

1.5 Hypothesis

This thesis will intend to prove the hypotheses listed below:

- **H1:** Probe that our objective function of selection is related with two mathematical properties: Monotonicity and Submodularity.
- **H2:** Reducing the size of the search tree generated helps to solve more problems.

1.6 Contribution of the Thesis

The main contributions of this Thesis are:

- Provide a prediction method to estimate the size of the search tree generated.
- Provide a meta−reasoning approach based on the size of the search tree generated.
- Provide a meta−reasoning approach based on the evaluation cost of each heuristic.

1.7 Organization of the Thesis

The Thesis is organized as follows:

1. In Part 1, the background of the thesis is provided which also includes our motivation and define the scope.
2. In Part 2, we review the State of the Art.
3. In Part 3, we introduce our meta−reasoning approach.
4. In Part 4, we introduce.
5. In Part 5, we .

6. We conclude in Part 6 by discussing further improvements and future work.

In the next chapter, the domain 8−tile−puzzle is used to understand the concepts that will be helpful for the other Parts.

Part II

Literature Review

2 Background

The purpose of this section is to understand the problem.

2.1 Similar Selection Systems

An optimization procedure which is similar to ours is presented by (RAYNER; STURTEVANT; BOWLING, 2013), but their procedure maximizes the average heuristic value. By contrast, the meta−reasoning we are proposing minimizes the search tree size.

Our meta−reasoning requires a prediction of the number of nodes expanded by A* using any given subset. Although there are methods for accurately predicting the number of nodes expanded by Iterative Deepening−A* (KORF, 1985) (IDA^*). (`SS` system (LELIS; ZILLES; HOLTE, 2013)), these methods can't be easily adapted to A* because A*'s duplicate pruning makes it very difficult to predict how many nodes will occur at depth d of A^*'s search tree (the tree of nodes expanded by A^*). As a part of our proposal, we present `SS` for predicting the size of the search tree.

The system most similar to ours is **RIDA*** (**??**). **RIDA*** also selects a subset from a pool of heuristics to guide the A^* search. In **RIDA*** this is done by starting with an empty subset and trying all combination of size one before trying the combination of size two and so on. **RIDA*** stops after evaluating a fixed number of subsets. While **RIDA*** is able to evaluate a set of heuristics with tens of elements, our meta−reasoning is able to evaluate a set of heuristics with thousands of elements.

2.2 Problem definition

A SAS^+*planning task* (BÄCKSTRÖM; NEBEL, 1995) is a 4 tuple $\nabla = \{V, O, I, G\}$. V is a set of *state variables.* Each variable $v \in V$ is associated with a finite domain of possible D_v. A state is an assignment of a value to every $v \in V$. The set of possible states, denoted V, is therefore $D_{v_1} \times ... \times D_{v_2}$. O is a set of operators, where each operator $o \in O$ is triple $\{pre_o, post_o, cost_o\}$ specifying the preconditions, postconditions (effects), and non-negative cost of o. pre_o *and* $post_o$ are assignments of values to subsets os variables, V_{pre_o} *and* V_{post_o}, respectively. Operator o is applicable to state s if s and pre_o agree on the assignment of values to variables in V_{pre_o}. The effect of o, when applied to s, is to set the variables in V_{post_o} to the values specified in $post_o$ and to set all other variables to the value they have in s. G is the goal condition, an assignment of values to a subset of variables, V_G. A state is a goal state if it and G agree on the assignment of values to the variable in V_G. I is the initial state, and the planning task, ∇, is to find an optimal (least-cost) sequence of operators leading from I to a goal state. We denote the optimal solution cost of ∇ as C^*

The state space problem illustrated in the figure 1 is a game that consists of a frame of numbered square tiles in random order with one tile missing. The puzzle also exists in other sizes, particularly the smaller 8−puzzle. If the size is 3×3 tiles, the puzzle is called the 8−puzzle or 9−puzzle, and if 4×4 tiles, the puzzle is called the 15−puzzle or 16−puzzle named, respectively, for the number of tiles and the number of spaces. The object of the puzzle is to place the tiles in order by making sliding moves that use the empty space.

The legal operators are to slide any tile that is horizontally or vertically adjacent to the blank into the blank position. The problem is to rearrange the tiles from some random initial configuration into a particular desired goal configuration. The 8−puzzle contains 181,440 reachable states, the 15−puzzle contains about 10^{13} reachable states, and the 24−puzzle contains almost 10^{25} states.

Initial

4	1	2
8		3
5	7	6

Goal

1	2	3
4	5	6
7	8	

Figure 1: The left tile−puzzle is the initial distribution of tiles and the right tile−puzzle is the goal distribution of tiles. Each one represent a State.

Instead of using an algorithm of Brute force search that will analyze all the possible solutions. We can obtain heuristics from the problem of the slide tile puzzle that will help us to solve the problem.

2.3 Heuristics

State−space algorithms, such as A* (HART P. E.; NILSSON; RAPHAEL, 1968), are important in many `AI` applications. A* uses the $f(s) = g(s) + h(s)$ cost function to guide its search. Here, $g(s)$ is the cost of the path from the start state s, and $h(s)$ is the estimated cost−to−go from s to a gial; $h(.)$ is known as the heuristic function. The heuristic is the mathematical concept that represent to the estimate distance from the node s to the nearest goal state.

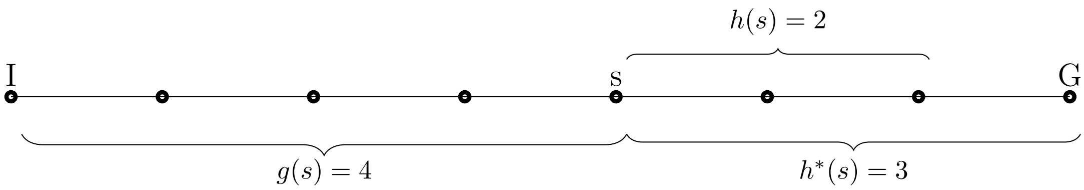


Figure 2: Heuristic Search: I: Initial State, s: Some Sate, G: Goal State

In the figure 2 the optimal distance from the Initial State I to the state s is 4 and represented by $g(s)$. The $h^*(s)$ represent the optimal distance from s to the Goal State G. And the $h(s)$ is the estimation distance from s to G.

A heuristic function $h(s)$ estimates the cost of a solution path from s to a goal state. A heuristic is admissible if $h(s) \leq h^*(s)$ for all $s \in V$, where $h^*(s)$ is the optimal cost of s. A heuristic is consisten iff $h(s) \leq c(s,t) + h(t)$ for all states s and t, where $c(s,t)$ is the cost of the cheapest path from s to t. For example, the heuristic function provided by a pattern database (`PDB`) heuristic (CULBERSON; SCHAEFFER, 1998) is admissible and consistent.

Given a set of admissible and consistent heuristics $\zeta = \{h_1, h_2, \ldots, h_M\}$, the heuristic $h_{max}(s,\zeta)$ =max$_{h \in \zeta} h(s)$ is also admissible and consistent. When describing our method we assume all heuristics to be consistent. We define $f_{max}(s,\zeta) = g(s) + h_{max}(s,\zeta)$, where $g(s)$ is the cost of the path expanded from I to s. $g(s)$ is minimal when A* using a consistent heuristic expands s. We call an A* search tree the tree defined by the states expanded by A* using a consistent heuristic while solving a problem ∇.

The heuristics can be obtained from each state of the problem. For example, for the problem of the 8−tile−puzzle figure 1 we can get two heuristics.

2.3.1 Out of place (O.P)

Counts the number of objects out of place.

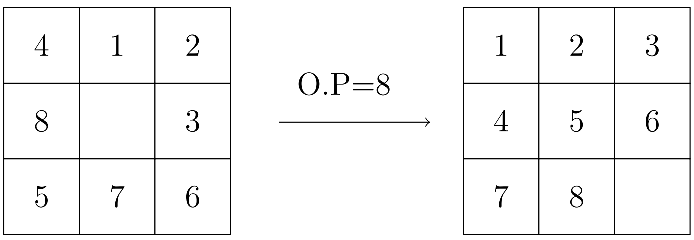


Figure 3: Out of place heuristic

The tiles numbered with 4, 1, 2, 3, 6, 7, 5, 8, and 4 are out of place then each object count as 1 and the sum would be 8.

2.3.2 Manhatham Distance (M.D)

Counts the minimum number of operations to get to the goal state.

4	1	2
8		3
5	7	6

$\xrightarrow{\text{M.D=1O}}$

1	2	3
4	5	6
7	8	

Figure 4: Manhatham distance heuristic

The tile 4 count 1 to get to the goal position. The tile 1 count 1 to get to the goal position. The tile 2 count 1 to get to the goal position. The tile 3 count 1 to get to the goal position. The tile 6 count 1 to get to the goal position. The tile 7 count 1 to get to the goal position. The tile 5 count 1 to get to the goal position. The tile 8 count 1 to get to the goal position. Then the sum would be 10.

In order to solve the problem, we get the heuristics, which are information from the problem to solve the problem. Exists systems that can create heuristics for each problem. Those systems are called Heuristic Generators.

2.4 Heuristic Generators

Heuristic Generators works by creating abstractions of the original problem space. The approach that has showed more successful results lately is `PDB`.

2.4.1 Pattern Database (PDB)

It's obtained by abstracting away certain problem variables, so that the remaining problem ("pattern") is small enough to be solved optimally for every state by blind exhaustive search. The results stored in a table, represent a PDB for the original problem. The abstraction of the search space gives an admissible heuristic function, mapping states to lower bounds.

2.5 Take advantage of Heuristics

The heuristics generators can create hundreds or even thousand of heuristics. In fact, exists different ways to take advantage of those heuristics. For example: If we want to use all the heuristics created by the heuristic generator. It would not be a good idea to use all of them because the main problem involved would be the time to evaluate each heuristic in the search tree, it could take too much time.

One way to take advantage of heuristics would be to take the maximum of the set of heuristics. For example, using three different heuristics $h1, h2$ and $\max(h1, h2)$. Heuristic $h1$ and $h2$ are based on domain abstractions and the $\max(h1, h2)$ is the maximum heuristic value of $h1$ and $h2$.

Exists different approaches to take advantage from a large set of heuristics. In this dissertation we use the meta−reasoning based on the minimum evaluation time.

2.6 Number of heuristics created

Let's suppose we have to run our meta−reasoning using M amount of memory available. The question would be: How many heuristics our system should handle in order to avoid out of memory errors? So. one of the objectives of this tesis is to find the number of heuristics that our subset ζ' should have.

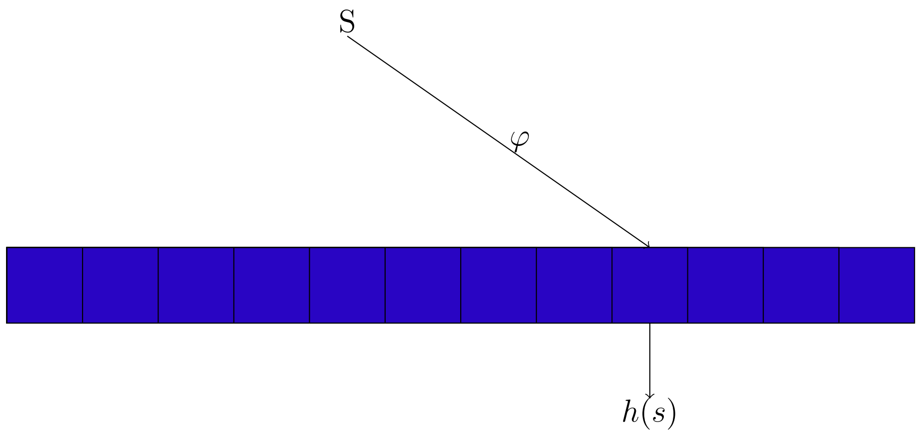


Figure 5: One heuristic of size M

In the Figures 6 and 7 we are taking advantange of the heuristics doing the maximization of all the heuristics created.

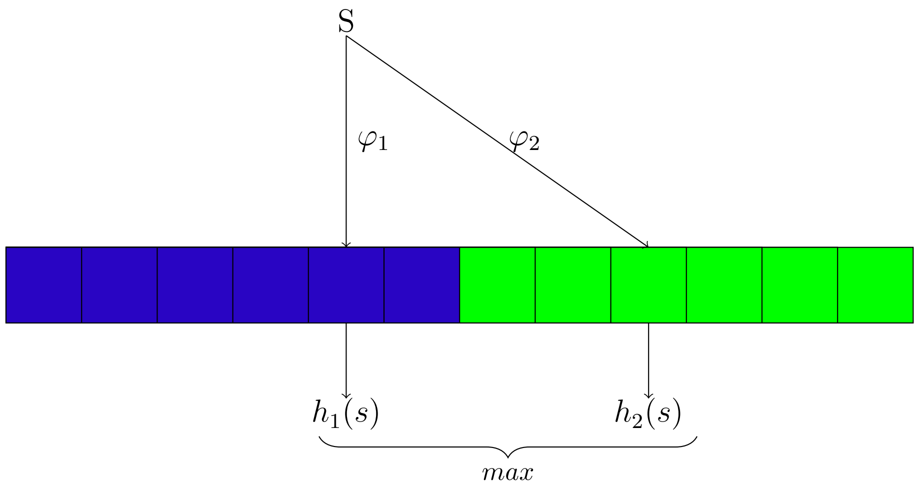


Figure 6: Two heuristics of size M/2

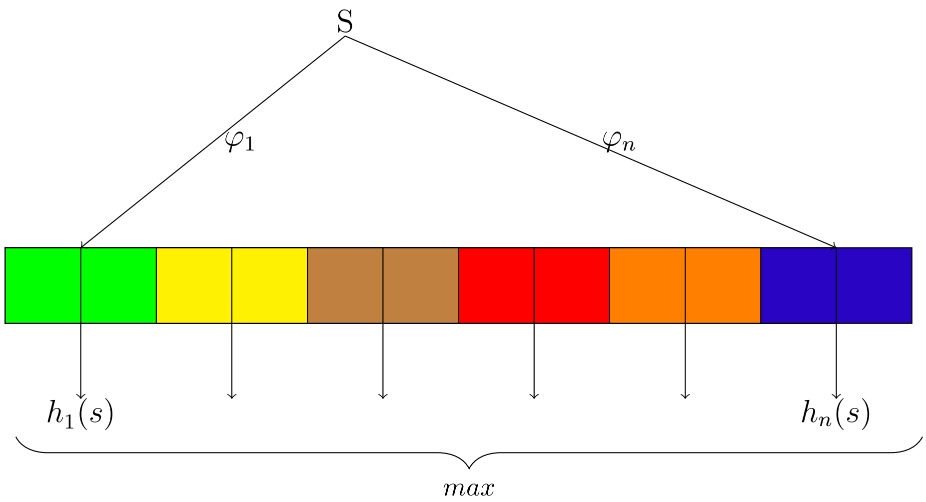


Figure 7: N heuristics of size M/N

2.7 Heuristic Subset

The heuristics generator systems can create a large number of heuristics. Let's suppose $|\zeta| = 1000$ heuristics were created considering the time and memory avaiable and we want to select the best N = 100 heuristics. This would be:

$$\binom{1000}{100} = 10^{138} possibilities$$

So, try to select heuristics from a large set of heuristics are going to be treated as an optimization problem. Then, in order to obtain a good selection of subset of heuristics, our objective function should guarantee two properties: Monotonicity and Submodularity, that would be explained in the next Part.

In the next Part, we will introduce the meta−reasoning proposed for selecting heuristics and will expand on the properties of our objective functions.

Part III

Approach Proposal

3 Greedy Heuristic Selection

The purpose of this section is to introduce the meta−reasoning proposed.

3.1 Problem Formulation

When solving $\triangledown$ using the consistent heuristic function $h_{max}(\zeta^{'})$ for $\zeta^{'} \subseteq \zeta$, A* expands in the worst case $J(\zeta^{'}, \triangledown)$ nodes, where

$$J(\zeta^{'}, \triangledown) = |\{s \in V | f_{max}(s, \zeta^{'} \leq C^{*})\}| \tag{3.1}$$

$$J(\zeta^{'}, \triangledown) = |\{s \in V | h_{max}(s, \zeta^{'} \leq C^{*}) - g(s)\}| \tag{3.2}$$

We present a greedy algorithm for approximately solving the following optimization problem,

$$\begin{aligned} \textbf{minimize}_{\zeta^{'} \in 2^{|\zeta|}} J(\zeta^{'}, \triangledown) \\ \textbf{subject to} |\zeta^{'} = N| \end{aligned} \tag{3.3}$$

Where N could be determined by a hard constraint such as the maximum number of `PDBs` one can store in memory.

3.2 GHS Algorithm

Algorithm 1 presents Greedy Heuristic Selection (`GHS`), an approximation algorithm for selecting a subset $\zeta^{'} \subseteq \zeta$.

The algorithm receives as input a planning problem $\triangledown$, a set of heuristics ζ, a cardinality size N, and it returns a subset $\zeta^{'} \subseteq \zeta$ of size N. In each iteration `GHS` greedily selects from ζ the heuristic h which will result in the largest reduction of the value of J (line 3). `GHS` returns $\zeta^{'}$ once it has the desired cardinality size N.

Input: Problem ∇, set of heuristics ζ, cardinality N

Ouput: heuristic subset $\zeta' \subseteq \zeta$ of size N

$\zeta' \leftarrow \emptyset$

while $|\zeta'| < N$ *do* **do**

  $h \leftarrow \texttt{argmin}_{h \in \zeta} J(\zeta' \cup \{h\}, \nabla)$

  $\zeta' \leftarrow \zeta' \cup \{h\}$

  return ζ'

end

Algoritmo 1: Greedy Heuristic Selection

3.2.1 GHS Approximation Analysis

In the following analysis all heuristic functions are assumed to be consistent. We also assume that A* expands all nodes n with $f(n) \leq C^*$ while solving ∇, as shown in Equation (3.1).

3.3 Stratified Sampling (SS)

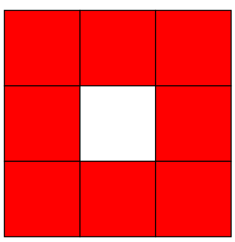
Stratified Sampling is a prediction algorithm that estimate the number of nodes expanded by some heuristic.

(**??**) created a method to estimate the size of the search tree such as IDA*. It works doing random walk from the root of the tree. Knuth's assumption is that all branches have the same structure. So, performing a random walk down one branch is enought to estimate the size of the search tree. However, the method does not work well for unbalanced search tree. (**??**) solved this problem with a stratification of the search tree through a *type system* to reduce the variance of the sampling process(LELIS; ZILLES; HOLTE, 2013)

In the figure 8 each node of the Search Space is mapped to the *Type System*

3.3.1 Type System

The *Type System* is calculated based of any property of each node in the search tree.



(2, M)

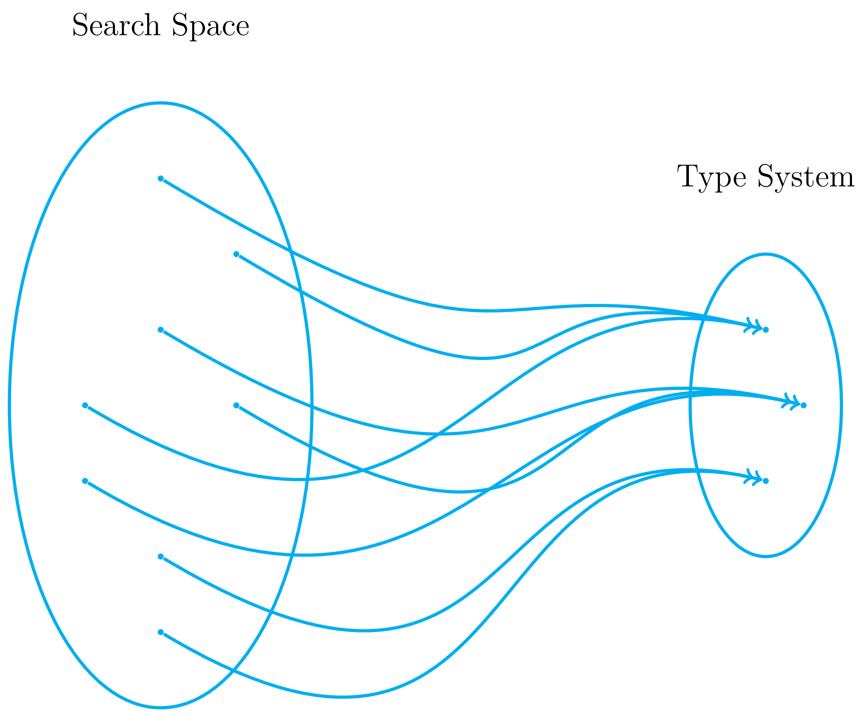
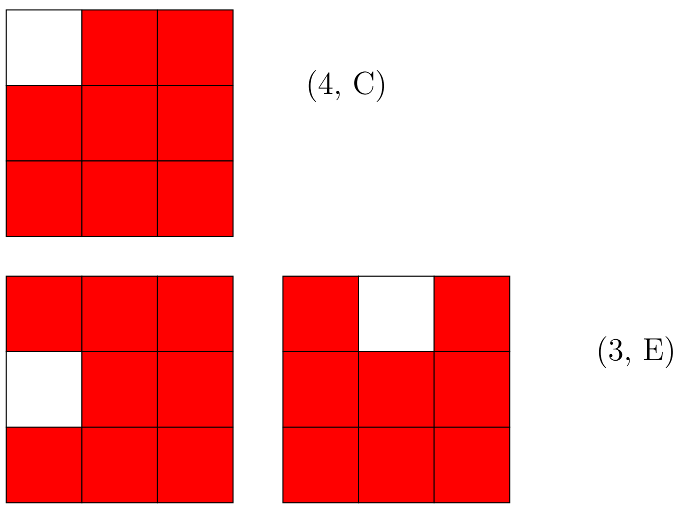
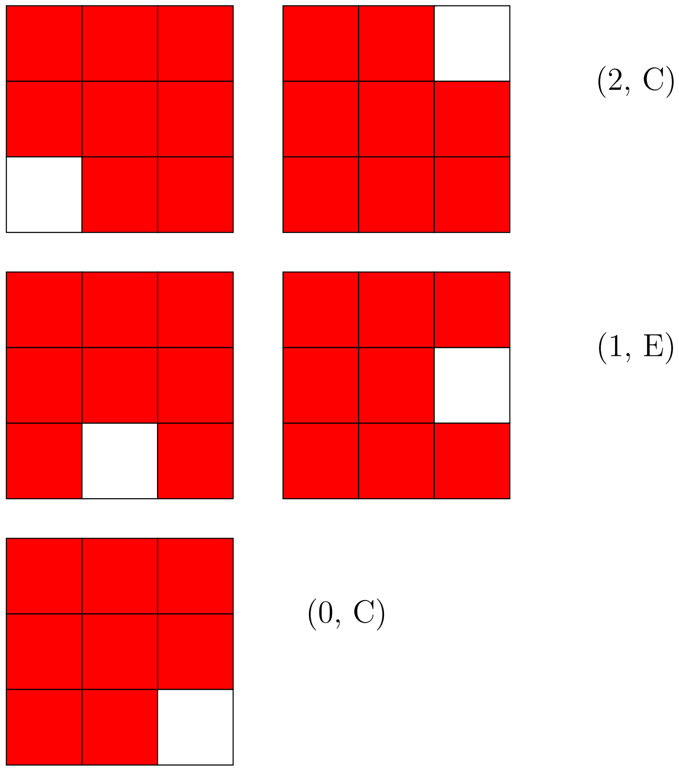


Figure 8: Type system is an abraction of the Search Space.

Figure 9: The heuristic value is the position of the empty space in a Specific state.





In the Figure 10, we can see how *Type System* works. In the Level 1, we have the root node, we add the property called weight or (W) initialized with one. Let's suppose that three nodes are generated by the root node in the Level 2. The nodes in the Level 2 have the following types: red, blue and red respectively, and each node recive the same W of the father. In the Level 2 we apply the concept of *Type System*, two nodes in the same level that have the same type (The same color) generate the same subtree. There are two nodes with type red in Level 2. So, we choose randomly one of them. Let's suppose we choose the right red node. Then, we have to update the number of nodes with the type red using the W, both red node types have $W = 1$, then we sum the W and the the new $W = 2$. So, in the Level 2 we will have two nodes of red type and one node with blue type.

When nodes in the Level 2 are expanded. The blue node expands one node of type blue and the red node expands two nodes of type red and blue. The question here is how many nodes would be generated in the Level 3? The answer is: $1 \times blue + 2 \times red + 2 \times blue$. So, in the Level 3 we will have 2 nodes of red type and 3 nodes of type blue.

In the Level 3 the W of the node blue would be the same W of the father. The father has the $W = 1$, then the child has the $W = 1$. The W of the red type and blue type would be 2. Once the W has been updated for each node in the Level 3 we apply the concept of *Type System* again. There are two nodes with type blue. So, we choose randomly one of them and update the W. Let's choose the right blue type and the updated W would be 3 because 1 from the left blue type plus the 2 from the right blue type.

When nodes in the Level 3 are expanded. The red node expands two nodes of types red and blue and the blue node expands one of the red. How many nodes would be generated at Level 4? The answer is: $2 \times red + 3 \times red + 2 \times blue$. So, in the Level 4 we will have five nodes of type red and two nodes of type blue.

The number of nodes expanded in the search tree is obtained summing all W plus one (The root node). So, the number of nodes expanded in the search tree would be $15 + 1 = 16$.

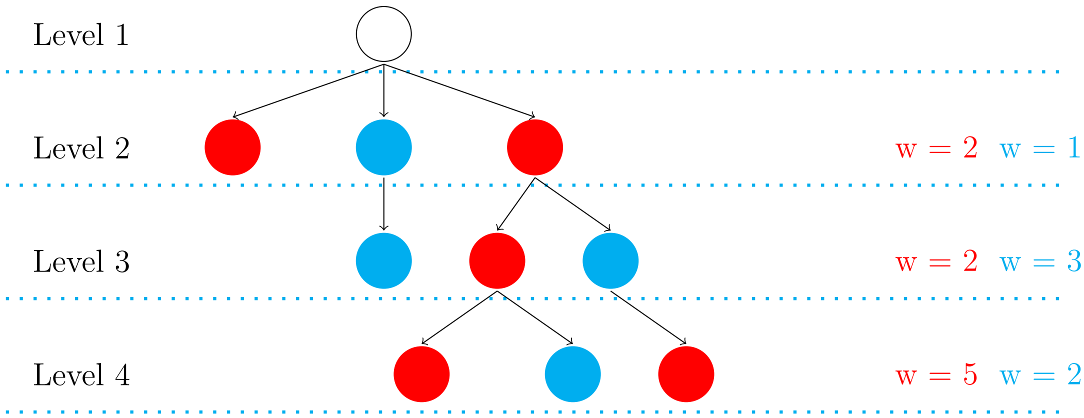


Figure 10: Search tree using Type System

3.4 Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

4 Nam sed tellus sit amet lectus urna ullamcorper tristique interdum elementum

4.1 Pellentesque sit amet pede ac sem eleifend consectetuer

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

5 Conclusão

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.

Sed eleifend, eros sit amet faucibus elementum, urna sapien consectetuer mauris, quis egestas leo justo non risus. Morbi non felis ac libero vulputate fringilla. Mauris libero eros, lacinia non, sodales quis, dapibus porttitor, pede. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi dapibus mauris condimentum nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam sit amet erat. Nulla varius. Etiam tincidunt dui vitae turpis. Donec leo. Morbi vulputate convallis est. Integer aliquet. Pellentesque aliquet sodales urna.

Bibliography

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 14724*: Informação e documentação — trabalhos acadêmicos — apresentação. Rio de Janeiro, 2005. 9 p. Citado na página 51.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 14724*: Informação e documentação — trabalhos acadêmicos — apresentação. Rio de Janeiro, 2011. 15 p. Substitui a Ref. ABNT (2005). Citado na página 3.

BÄCKSTRÖM, C.; NEBEL, B. Complexity results for sas+ planning. *Computational Intelligence*, Wiley Online Library, v. 11, n. 4, p. 625–655, 1995. Citado na página 31.

CULBERSON, J. C.; SCHAEFFER, J. Pattern databases. *Computational Intelligence*, Wiley Online Library, v. 14, n. 3, p. 318–334, 1998. Citado na página 33.

DOMSHLAK, C.; KARPAS, E.; MARKOVITCH, S. To max or not to max: Online learning for speeding up optimal planning. In: *AAAI*. [S.l.: s.n.], 2010. Citado na página 26.

HART P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC*, IEEE, v. 4, n. 2, p. 100–107, 1968. Citado 2 vezes nas páginas 21 and 32.

HELMERT, M. The fast downward planning system. *J. Artif. Intell. Res.(JAIR)*, v. 26, p. 191–246, 2006. Citado na página 26.

HOLTE, R. C. et al. Maximizing over multiple pattern databases speeds up heuristic search. *Artificial Intelligence*, Elsevier, v. 170, n. 16, p. 1123–1136, 2006. Citado 2 vezes nas páginas 13 and 26.

KORF, R. E. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, v. 27, p. 97–109, 1985. Citado na página 31.

LELIS, L. H.; ZILLES, S.; HOLTE, R. C. Predicting the size of ida* search tree. *Artificial Intelligence*, Elsevier, v. 196, p. 53–76, 2013. Citado 2 vezes nas páginas 31 and 42.

RAYNER, C.; STURTEVANT, N.; BOWLING, M. Subset selection of search heuristics. *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, p. 637–643, 2013. Citado na página 31.

TOLPIN, D. et al. Towards rational deployment of multiple heuristics in a*. In: AAAI PRESS. *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. [S.l.], 2013. p. 674–680. Citado na página 26.

Appendix

APPENDIX A – Quisque libero justo

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

APPENDIX B – Nullam elementum urna vel imperdiet sodales elit ipsum pharetra ligula ac pretium ante justo a nulla curabitur tristique arcu eu metus

Nunc velit. Nullam elit sapien, eleifend eu, commodo nec, semper sit amet, elit. Nulla lectus risus, condimentum ut, laoreet eget, viverra nec, odio. Proin lobortis. Curabitur dictum arcu vel wisi. Cras id nulla venenatis tortor congue ultrices. Pellentesque eget pede. Sed eleifend sagittis elit. Nam sed tellus sit amet lectus ullamcorper tristique. Mauris enim sem, tristique eu, accumsan at, scelerisque vulputate, neque. Quisque lacus. Donec et ipsum sit amet elit nonummy aliquet. Sed viverra nisl at sem. Nam diam. Mauris ut dolor. Curabitur ornare tortor cursus velit.

Morbi tincidunt posuere arcu. Cras venenatis est vitae dolor. Vivamus scelerisque semper mi. Donec ipsum arcu, consequat scelerisque, viverra id, dictum at, metus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut pede sem, tempus ut, porttitor bibendum, molestie eu, elit. Suspendisse potenti. Sed id lectus sit amet purus faucibus vehicula. Praesent sed sem non dui pharetra interdum. Nam viverra ultrices magna.

Aenean laoreet aliquam orci. Nunc interdum elementum urna. Quisque erat. Nullam tempor neque. Maecenas velit nibh, scelerisque a, consequat ut, viverra in, enim. Duis magna. Donec odio neque, tristique et, tincidunt eu, rhoncus ac, nunc. Mauris malesuada malesuada elit. Etiam lacus mauris, pretium vel, blandit in, ultricies id, libero. Phasellus bibendum erat ut diam. In congue imperdiet lectus.

Annex

ANNEX A – Morbi ultrices rutrum lorem.

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

ANNEX B – Cras non urna sed feugiat cum sociis natoque penatibus et magnis dis parturient montes nascetur ridiculus mus

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

ANNEX C – Fusce facilisis lacinia dui

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.

Index