



<http://researchspace.auckland.ac.nz>

ResearchSpace@Auckland

Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

To request permissions please use the Feedback form on our webpage.

<http://researchspace.auckland.ac.nz/feedback>

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the [Library Thesis Consent Form](#) and [Deposit Licence](#).

Note : Masters Theses

The digital copy of a masters thesis is as submitted for examination and contains no corrections. The print copy, usually available in the University Library, may contain corrections made by hand, which have been requested by the supervisor.

Automatic Heuristic Selection, on a Problem by Problem Basis, Using an Analytical Model and *In Situ* Sampling

Author: Santiago Franco

Supervisor: Dr. Mike Barley

The Department of Computer Science

The University of Auckland

New Zealand

March 2012



A THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

Abstract

One of the main drives in automated planning is to automate the translation from the knowledge level description of the problem to the symbol level implementation of a solver for that problem. The field is at the point where there are techniques that can automatically synthesize very large numbers of heuristics that can be used by heuristic search based problem solvers [Holte *et al.*, 2005], [Edelkamp, 2006], [Culberson and Schaeffer, 1994], [Haslum *et al.*, 2007], etc. However, currently there is no *a priori* method available to automatically tailor them to a specific problem instance. Nor are there any automated techniques to automatically determine which heuristics are best for a problem. There are some automated techniques to generate problem specific heuristics based on *in situ* gathered information but they are not competitive with state-of-the-art manual heuristic selection.

Currently, the state-of-the-art is for the user to determine this *a priori*. This is usually done by running each of the heuristics on a wide range of problems in the chosen domain. The heuristic that is best on average for the domain is then used for each problem in the domain. The current approach is empirical, this thesis explores an analytical approach; which uses a simplified run-time formula that represents the important components of the problem-solver. This formula predicts the impact of the selected heuristic subsets upon run-time. Some of the formula's parameters can be determined *a priori* (e.g., how long it takes to check for a goal state) while others can only be determined *a posteriori*. Unfortunately, finding out those values after the problem has been solved is not usually very useful.

This thesis explores the possibility of using the early parts of the problem-solving task (*in situ* sampling) to determine approximate values for those parameters. A system called RIDA* was created to do this. The danger is that the cost of determining these approximate values may outweigh the benefits those values bring. The ideal situation is to only do as much extra work as is needed to obtain the approximate values that allow us to make a decision that saves enough work to compensate for the time invested.

We found that RIDA* was competitive, and in some cases significantly faster, than manual heuristic selection. However, there are two caveats: RIDA* has a scalability issue for large heuristic sets due to the combinatorial explosion in the heuristics search space. Also RIDA*'s *in situ* sampling effort is currently determined *a priori* for the domain, it does not have an *in situ* mechanism to adjust its sampling effort. Both of these caveats are the subject of future directions for research. This thesis work (RIDA*) significantly advances the automation of *in situ* heuristic selection.

Finally, RIDA* uses new data structures which significantly reduce the costs associated with *in situ* sampling. Significantly reducing *in situ* sampling costs is critical for any technique using problem-specific heuristics to be competitive with state-of-the-art manual heuristic selection.

To my mother, Maria Reyes Aixela Barasona, wife, Susan Elizabeth
Quantrell and my two beautiful children, Rodrigo and Megan.

Acknowledgement

It is with great pleasure that I am able to take this opportunity to express my sincere appreciation to my supervisor, Dr. Mike Barley, to whom I will always be indebted for his guidance, wisdom, dedication to my work, and, most of all, endless support.

I must also express my gratitude to the Computer Science Department at the University of Auckland. Whenever I needed any help, all the administrative and IT support staff were helpful and friendly. I would also like to thank my colleagues in the postgraduate office, always ready for a good nature chat whenever the thesis proved to much, or a bit of LaTeX advice that saved me hours of online search (Thanks All!).

I would like to mention that I am very thankful to the University of Auckland for fully financing my doctoral studies through a Doctoral Scholarship.

Also thanks to my dear friend, Dr. Clara Palleja-Lopez, whose friendship, advice and support helped me finishing this thesis.

Finally, I must express my deepest gratitude to my wife, Susan Elizabeth Quantrell, whose love and support helped me to never give up and finish this thesis.

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Problem Description	3
1.2.1	Classic Problem	3
1.2.2	Modern Problem	3
1.3	Evaluation Basis	6
1.4	Comparison Basis to the Manual and Automatic Approaches	7
1.5	Brief Approach Description	10
1.6	Main Claims	13
1.7	Roadmap	16
2	Reconfigurable-IDA* (RIDA*) <i>In Situ</i> Selection Framework	17
2.1	Introduction	17
2.2	Possible Approaches to Heuristic Selection	20
2.2.1	<i>A Priori</i>	20
2.2.2	<i>A Posteriori</i>	22
2.3	Best Heuristic Subset on Average for the Domain	22
2.4	What is Automatic <i>In Situ</i> Heuristic Selection	27
2.4.1	RIDA* Efficiently Combines Problem Solving and <i>In Situ</i> Sampling	28
2.5	Time Vs Size as a Performance Metric	30

2.6	Brief Discussion on the Sampling Effort and Capping	32
2.7	Main Claims	33
2.8	RIDA* Summarized Description and Example	35
2.8.1	Sampling Module Example	36
2.8.2	Prediction Module Example	41
2.9	<i>In Situ</i> Search and the Literature	44
2.9.1	In Situ Planner	44
2.9.2	Time Modelling of Overall Search Cost	47
3	Compacting the In Situ Gathered Data	51
3.1	Why the Heuristic Union Search Tree	51
3.2	On the Fly Generation of the Heuristic Union Search Tree: the Mintree Rule	55
3.3	The HUST's Credit Assignment Problem	55
3.3.1	Culprit Counters	56
3.3.2	Culprit Counters Are Memory Intensive but Time Efficient	57
3.4	Solving the Credit Assignment Problem Using the Culprit Counter Lattice	58
3.4.1	Detailed Example	58
3.4.2	Parent Improvement	60
3.5	Managing Large Heuristic Powersets	61
3.5.1	Limiting the combination degree	61
3.5.2	Sparse Representation of the Culprit Counter Lattice	63
3.6	HUST's Safe Early Truncation	65
4	RIDA* Run-time Formula	69
4.1	Introduction	69
4.1.1	Time vs Size as a Performance Comparison Metric	70
4.1.2	Parametric Modelling for IDA*	72
4.2	A Generic Parametric Model for IDA*	75
4.2.1	Generic Algorithm for IDA*	75
4.2.2	Parametric Equations for Generic Algorithm for IDA*	75

4.3	Obtaining the Parametric Equation Variables Value for IDA*	85
4.3.1	Uniform Search Tree Method	86
4.3.2	HBF Method	88
4.3.3	Heuristic Distribution Method	91
4.3.4	Conclusions Regarding Node Generated Prediction Models	97
4.4	Conclusion	99
4.5	IDA* Modelling Relevant Literature	100
5	Extended Literature Review	105
5.1	Introduction	105
5.2	Hyper-heuristics	111
5.2.1	Introduction	111
5.2.2	Classification	113
5.3	ABSOLVER II	119
5.3.1	Composer	120
5.3.2	Dropper	121
6	Experiments	123
6.1	Introduction	123
6.1.1	Pattern Databases(PDBs)	127
6.1.2	Pattern Databases' Relevant Literature	129
6.2	Experimental Setup	132
6.2.1	Fifteen Puzzle Experiments Setup	132
6.2.2	Twenty-four Puzzle Experimental Setup	134
6.2.3	Avoiding Goal Placement Stochastic Performance Comparison Effects	136
6.2.4	Competitive Random Generation of Neighboring Patterns	137
6.2.5	Towers of Hanoi Setup	139
6.2.6	Pattern Database for Towers of Hanoi	140
6.3	Claim #1: . . . Automating Heuristic Selection Using a Run-time Formula. . .	142
6.3.1	Run-time Formula	142

6.3.2	This Runtime Formula Uses <i>In Situ</i> Parameters to Approximate <i>A Posteriori</i> Values.	147
6.3.3	RIDA* Is Faster than The Standard Automated Solution	150
6.3.4	RIDA* Is Competitive With The “Best Heuristic Subset on Average for the Domain” Approach	154
6.3.5	Another Domain: Towers of Hanoi	160
6.4	Claim#2: Compact Representation of HSTs Reduces Overall Sampling Costs	162
6.4.1	Fifteen Puzzle	162
6.4.2	Twenty-four Puzzle	164
6.4.3	Towers Of Hanoi	166
6.5	Claim #3: Solution to the Credit Assignment Problem	168
6.5.1	Towers Of Hanoi Credit Assignment Problem	169
6.6	Summary of Results for Claims 1-3	171
6.6.1	Claim#1	171
6.6.2	Claim #2	176
6.6.3	Claim #3	177
6.7	RandomizationExperiments	178
6.7.1	Introduction	178
6.7.2	Fifteen Puzzle Results	180
6.7.3	Twenty-four Puzzle Results	182
6.7.4	Randomization Approach for Towers of Hanoi	185
6.8	CrossoverExperiments	187
6.8.1	Introduction	187
6.8.2	Fifteen Puzzle	188
6.8.3	Twenty-Four Puzzle	189
6.8.4	Crossovers for Towers of Hanoi	194

7 Conclusions

197

7.1	Problem Description and Solution Approach	197
-----	---	-----

7.1.1	Problem Description	197
7.1.2	Solution Approach	198
7.2	Claims	200
7.2.1	First Claim:Using Run-time Formula to Automate Heuristic Selection	200
7.2.2	Second Claim: Compact Representation Reduces Sampling Costs .	204
7.2.3	Third Claim:A Credit Assignment Solution	206
7.2.4	Future Work	208
7.3	Usefulness of RIDA* to Different Problem Domains	211
7.3.1	Generic Heuristic Creation using Abstraction Technique(GHCAT) .	211
7.3.2	Memory Partitioning	211
7.3.3	ABSOLVER II	212
A	Theory and Definitions	215
A.1	Algorithm Related definitions	215
A.2	Graph Theory	217
A.3	Heuristic Search Trees definitions	220
B	Detailed Experimental Data	225
B.1	Fifteen Puzzle Data	225
B.2	Twenty-four Puzzle Data	227
B.2.1	Random Heuristic generation	227
B.2.2	Full Parametric Model Versus Average Time per Node Model . . .	231
B.2.3	Results	234
B.2.4	Results Vs Choosing the Best on Average Heuristic Combination .	236
B.2.5	Data Sampling Results	237
B.2.6	Separated <i>Sampling</i> and <i>In Situ</i> Run-times	239
B.2.7	Number of Generated Nodes	241
B.2.8	Randomization	244

List of Figures

1.1	Best On Average Vs Problem By Problem Possible Outcomes	8
1.2	Diagram Showing RIDA* Modules Basic Functionality.	11
2.1	Diagram Showing RIDA* Modules' Basic Functionality.	37
3.1	HUST example with Culprit Counter Lattice and Final Size Counters. . . .	52
3.2	Culprit Counters Lattice	59
4.1	Iterative HBF comparison of single 6_6_6 PDB Heuristic Vs Max of 8 PDBs(5_5_5_5_4) for First 6 Instances of 24 Puzzle in [Holte <i>et al.</i> , 2006]. Horizontal axis are F-values, Vertical axis are HBF values.	92
6.1	Our Five 7-1-7 Heuristics	133
6.2	Four-Disk four peg Towers of Hanoi HBF.. . . .	139
6.3	Iterative HBF for our Twenty-five heuristic set (section 6.2.4). The hori- zontal axis is the iteration number, the vertical axis are the HBF values.. Avg_HBF is the average HBF for the last HBF for each problem.	147
6.4	Optimality Vs Average Savings as a Function of the Capping Limit For the Fifteen puzzle Experiments. (Table B.2, p. 226). $Optimality = avg(100 * \frac{best\ time}{chosen\ time})$; $AvgSavings = avg(100 \times (1 - \frac{selected\ time + sampling\ time}{Max.of.5\ time}))$	147

6.5	In this graph we show the overall run-times for both the “maximizing the whole available heuristic set” approach and RIDA*. For each of the heuristic sets we used the maximum combination degree with the best results. We show overall times, for problem specific data see B.8	154
6.6	In this graph we show the RIDA*’s run-times compared to using the best subset on average for the problem suite. We show overall times. For problem specific data see B.10. These results are for the Twenty-four puzzle problem suite in [Holte <i>et al.</i> , 2006].	159
6.7	In this graph we show RIDA*’s run-time for the Fifty heuristic set. We also show the run-time when selecting the best subset on average for the problem suite. We also show the results when the best subset on average is selected from a different set of random problems of the same size. We show overall times. For problem specific data see B.10. These results are for the Twenty-four puzzle problem suite in [Holte <i>et al.</i> , 2006].	160
6.8	In this graph we show the Credit Assignment impact on the HUST time compression. The more subsets to assign, the more the original time compression is reduced. The actual number of subsets for each entry is in table 6.11	168
6.9	Overall Run Time with/without Randomization for Twenty-four suite of six random problems as in [Holte <i>et al.</i> , 2006]	183
6.10	Schema Displaying the Three Critical Distances in a Crossover.	190
B.2	PDB-based Heuristic used for testing parametric prediction models. Tiles picked randomly but ensuring that final patterns follow Korf and Felner rule of thumb[Korf and Felner, 2002]	231
B.2	Individual Run Time with/without Randomization for Twenty-four suite of six random problems as in [Holte <i>et al.</i> , 2006]	244

List of Tables

2.1	Sampling Phase example for Twenty-Four Puzzle problem #1 [Korf <i>et al.</i> , 2001]. Fifty Heuristic Set, Maximum Combination Degree 4. HUST Compression Factors without Including the Credit Assignment Problem.	38
2.2	Sampling Phase example for Twenty-Four Puzzle problem #1 [Korf <i>et al.</i> , 2001]. Fifty Heuristic Set, Maximum Combination Degree 4. HUST Compression Factors with/without including the Credit Assignment Problem.	40
2.3	Solving Phase example for Twenty-Four Puzzle problem #1 [Korf <i>et al.</i> , 2001]. Fifty Heuristic Set, Maximum Combination Degree 4. Inverse operator check and Early Stopping are being used. Selected four heuristic combina- tion: 5, 18, 31, 37 (Table B.3)	43
3.1	Use only one parent to avoid culprit duplication and reduce additions by almost one order of magnitude	61
3.2	Number of Additions when Limiting Maximum Combination Degree Example	63
3.3	Time to build the HUST vs time to solve its Credit Assignment Problem (C.A.P) as a function of the maximum combination degree. Always using a set with thirty heuristics and the same example problem. See algorithm 3.1. Time in seconds.	65
4.1	Grouping Tasks from Algorithm 4.1, p. 76 as Variables for Parametric Equation.	78

4.3	Advantages and Caveats of Domain Based Sampling vs <i>In Situ</i> Sampling	100
6.1	Time per Node as a Function of the Number of Heuristics, see Eq (6.2), (p. 143). Each additional heuristic adds 0.096μ seconds.	135
6.2	Stochastic Changes in Comparative Performance Generated by Stopping Once the Goal Is Found. Using Twenty-four Puzzle PDB-based Heuristics as in [Holte <i>et al.</i> , 2006]	136
6.3	Average Prediction Accuracy per F-bound. For detailed experimental data see table B.6 in appendix B	144
6.4	Number of Problems Solved while Still in RIDA* Sampling Phase as a Function of the <i>A Priori</i> Fixed Capping Limit. There were 1,000 problems in the Fifteen puzzle experiment suite. For detailed experimental data see table B.2 (p. 226) in appendix B	149
6.5	Overall Run-time for State-Of-The-Art Heuristic Sets in [Holte <i>et al.</i> , 2006] and [Korf and Felner, 2002] and Our Random Generated Heuristics. No heuristic selection, taking the maximal for all heuristics in the set.	152
6.6	Speed-up ratio when Using In Situ Selection For the Eight, Twenty-five, Fifty and one Hundred Set Overall Times. $Speed_up\ Ratio = \frac{Maximize\ Whole\ Set\ Runtime}{RIDA*\ Overall\ Runtime}$	152
6.7	Best Subset On Average Vs RIDA* In Situ Selection for Fifteen Puzzle Five (4-4-4-4) Heuristic Set.	156
6.8	Overall time results for Towers of Hanoi(4 Pegs-16 Disks)	161
6.9	RIDA* overall time for its three distinct phases (HUST creation or sam- pling, Credit Assignment or Prediction and Solving time	162
6.10	HUST Savings	163
6.11	Redundancy Reduction and Time Compression No Credit Assign- ment Impact Included. Only overall Results, for problem specific data see B.2.5.	165
6.12	Redundancy Reduction and Time Compression No Credit Assign- ment Impact Included.	168

6.13	Time Compression with and without Credit Assingment(C.A.) . . .	170
6.14	Credit Assignment Costs	171
	180	
6.16	Features of the Used Approaches for the 24-Puzzle. All the PDBs Used for the 24-Puzzle were Stored as a Sparse Array in RAM.	182
6.17	Overall Time Ratios With&Without Randomization. Overall time data in Figure 6.9(p.183). Specific time data in Figure B.2(p.244)	184
6.18	Overall Time results when solving the 16 Disk Tower of Hanoi with 4 pegs using the Randomization approach for the whole set(with A*) and for RIDA*'s selected subset with M.C.D=3 and a capping limit of 100,000 nodes. RIDA* used A* as well for the Solving Phase.	187
	189	
6.20	Crossover Speed-up Ratio for the Twenty-four Puzzle Suite . Suite of six random problems, using the 25,50 and 100 heuristic sets.	191
6.21	Crossover Speed-up Ratio for the Twenty-four Puzzle Suite of Six Random Problems. Using the 25,50 and 100 heuristic sets.	196
B.1	Time per node	225
B.2	Fifteen Puzzle Results for a Thousand Random Problems Divided in Ten Groups	226
	231	
B.5	Avg. Cost per Node for “Average Cost per Node” Prediction Model.	232
B.6	Prediction of Average Run Time for IDA* Iterations on Twenty-Four Puz- zle Random Instances, with a PDB-based Heuristic(figB.2) and Inverse Operator Check	233
B.7	Twenty-four Puzzle Time Results(seconds) when Maximizing across All Heuristic Sets.	234
B.8	Twenty-Four Puzzle Time Results(seconds) Comparing RIDA* <i>In Situ</i> Heuristic Selection vs Heuristics in [Korf and Felner, 2002, Holte <i>et al.</i> , 2006]	234

B.9 Problem Specific Speed-up Ratio using RIDA* vs the Maximization Approach (Definition A.3.11) ($Speedup\ Ratio = \frac{MaximizationTime}{RIDA*Time}$) for the Twenty-five(5-5-5-5-4), Fifty(5-5-5-5-4) and Hundred(5-5-5-5-4) Set Overall Run-times.	235
	236
B.11 HUST Compression Factors for Twenty-four Puzzle Suite Of Six Experiments[Holte <i>et al.</i> , 2006]. Twenty Five Heuristic Set, Maximum Combination Degree 4.	237
B.12 HUST Compression Factors for Twenty-four Puzzle Suite Of Six Experiments[Holte <i>et al.</i> , 2006]. Twenty Five Heuristic Set, Maximum Combination Degree 5.	237
B.13 HUST Compression Factors for Twenty-four Puzzle Suite Of Six Experiments[Holte <i>et al.</i> , 2006]. Fifty Heuristic Set, Maximum Combination Degree 4.	238
B.14 HUST Compression Factors for Twenty-four Puzzle Suite Of Six Experiments[Holte <i>et al.</i> , 2006]. Fifty Heuristic Set, Maximum Combination Degree 5.	238
B.15 HUST Compression Factors for Twenty-four Puzzle Suite Of Six Experiments[Holte <i>et al.</i> , 2006]. Hundred Heuristic Set, Maximum Combination Degree 3.	239
B.16 HUST Compression Factors for Twenty-four Puzzle Suite Of Six Experiments[Holte <i>et al.</i> , 2006]. Hundred Heuristic Set, Maximum Combination Degree 4.	239
B.17 Twenty-four Puzzle Problem Specific Sampling (HUST and C.A.P) vs RIDA* Run-time for Eight(5-5-5-5-5,M.C.D=8), Twenty-five(5-5-5-5-4,M.C.D=4),Fifty(5-5-5-5-4,M.C.D=4) and Hundred(5-5-5-5-4,M.C.D=3) sets.	240
B.18 Twenty-four Puzzle Problem Specific Sampling (HUST and C.A.P) Time vs RIDA*'s Run-Time for Eight(5-5-5-5-5,M.C.D=8), Twenty-five(5-5-5-5-4,M.C.D=5),Fifty(5-5-5-5-4,M.C.D=5) and Hundred(5-5-5-5-4,M.C.D=4) sets.	240
B.19 Twenty-four Puzzle Number of Generated Nodes. Note that Results include both HUST nodes and regular IDA* nodes. Use HUST tables if need to separate data.	242