CENTRO DE CIENCIAS EXATAS E TECNOLOGICAS - CCE

DEPARTAMENTO DE INFORMATICA

REPORT:

COMPARISON BETWEEN SS AND IDA*

Marvin Abisrror Zarate
MSc Student in Computer Science

Levi Henrique Santana de Lelis
(Advisor)

Santiago Franco
(Co-Advisor)

VIÇOSA - MINAS GERAIS
JULY - 2015

Contents

1 Introduction

The new approach for selecting a subset of heuristics functions for domain-independent planning has two main objectives: First, make a selection of heuristics from a large set of heuristics with the goal of reducing the running time of a search algorithm employing the subset functions. Second, find out if the prediction of Stratified Sampling (SS) might be helpful in selecting a subset of heuristics to guide the A* search.

Maybe writing [Krause and Golovin, 2012] should demostrated something. Cite by holte [Holte et al., 2006] and this.

In order to achieve the first objective we present The Greedy Algorithm, which provides a good approximation to the optimal solution of the NP-hard optimization problem [Krause and Golovin, 2012].

In order to achieve the second objective we use the *relative unsigned error* to probe the accuracy of the predictions of SS with respect to IDA*. We know that SS does not make even reasonable predictions for the number of nodes expanded by A*. Nevertheless, even though SS produces poor predictions for the number of nodes expanded by A*, we would like to verify whether these predictions can be helpful in selecting a subset of heuristics to guide the A* search.

This report have three sections, the first section is the introduction, the second is the experiment 1 which contains four main tables showing the results of the *relative unsigned error*. and the last section in the conclusion.

2 Comparison of SS with IDA*

Stratification Sampling is an algorithm that estimates the number of nodes expanded performed by heuristic search algorithm seeking solutions in state space. We apply SS to predict the number of nodes expanded by IDA* in a given f-layer when using a consistent heuristics.

We first ran IDA* for Fast-Downward benchmark for optimal domains. Our evaluation metric is coverage, *i.e.,* number of problems solved within 24 hours time limit. We note that in 24 hours non all the instances for a specific domain using a consistent heuristic can be solved. Afterwards, run SS using as a threshold the f-layer for each instance of each domain, this process is executed using different number of probes *i.e.,* 1, 10, 100, 1000, and 5000.

In our experiment 1, prediction accuracy is measured in terms of the *Relative Unsigned Error* (ss-err), which is calculated as:

$$\frac{\sum_{s \in PI} \frac{Pred(s,d) - R(s,d)}{R(s,d)}}{|PI|}$$

Where *PI* is the set of problem instances, *Pred(s, d)* and *R(s, d)* are the predicted and actual number of nodes expanded by IDA* for start state s and cost bound d. A perfect score according to this measure is 0.000.

The heuristics used for this experiment 1 were: hmax, ipdb, lmcut, and merge_and_shrink. There are 4 tables, each table shows the results running IDA* and SS using one consistent heuristic. The first column represent the optimal domains for Fast-Downward benchmark. The remaining 10 columns shows the 5 different probes *i.e.,* 1, 10, 100, 1000, and 5000. Each probe has two columns which represent the ss-err and the ss-time. The last two columns are the information for IDA* which represent the average value of the number of nodes expanded and the average time respectively. The text "—" means that IDA* could not solve the problems, consequently there are not results for SS.

In the Table **??** we can see that there are seven domains which IDA* could not solve any instance in 24 hours. The ss-err decrease for each domain according the number of probes increases. The domains that have the perfect score are: openstacks-opt11-strips, parcprinter-opt11-strips.

Table 1: Experiment 1 - Comparison using hmax heuristic

	hmax												
	1		10		100		1000		5000				
Domain	error	time	error	time	error	time	error	time	error	time	ida*	ida*-time	n
barman-opt11	0.60	0.06	0.45	0.32	0.20	3.21	0.07	32.57	0.04	214.59	8835990.00	6016.38	20
blocks	0.42	0.02	0.17	0.10	0.06	1.06	0.03	10.86	0.01	65.97	28510300.00	3030.97	35
elevators-opt08	0.67	1.61	0.48	11.13	0.21	110.38	0.13	1140.05	0.48	3012.95	923397.00	4795.09	30
elevators-opt11	0.84	1.40	0.42	9.85	0.23	96.37	0.13	994.33	0.14	4223.73	966309.00	4759.72	20
floortile-opt11	2.02	0.01	0.62	0.07	0.40	0.69	0.14	6.93	0.11	36.60	30522300.00	3919.72	2
nomystery-opt11	0.53	0.07	0.26	0.38	0.07	3.63	0.03	36.35	0.01	181.03	6565740.00	3256.86	20
openstacks-adl	—	—	—	—	—	—	—	—	—	—	—	—	—
openstacks-opt08	0.58	82.20	0.04	800.37	0.10	1260.88	0.10	12368.90	0.22	9594.93	73087.30	2669.55	30
openstacks-opt11	0.03	94.79	0.03	774.86	0.10	991.57	0.10	10148.40	0.24	8779.80	62942.40	3156.36	20
parcprinter-opt11	0.00	0.01	0.00	0.04	0.00	0.35	0.00	3.48	0.00	17.29	1.00	0.00	20
parking-opt11	0.17	1.79	0.04	11.36	0.01	114.28	0.00	1196.83	0.00	5835.03	374925.00	5607.50	20
pegsol-opt11	0.17	0.01	0.04	0.04	0.02	0.37	0.01	3.69	0.00	17.88	68763.70	5.00	20
scanalyzer-opt11	0.43	3.13	0.25	28.79	18.63	273.74	0.02	3033.06	0.04	10021.00	8257850.00	4808.75	20
sokoban-opt08	0.35	0.27	0.22	1.95	0.09	20.24	0.05	214.01	0.03	965.40	2657890.00	3385.95	30
sokoban-opt11	0.41	0.31	0.26	2.00	0.11	21.42	0.05	222.47	0.04	1056.61	3118530.00	3932.69	20
tidybot-opt11	300.86	4.40	1072.40	26.48	5.88	238.76	0.01	2747.10	0.04	11572.90	431336.00	5465.62	20
transport-opt08	0.55	0.33	0.36	2.57	0.27	23.11	0.13	236.72	0.10	1363.04	1462640.00	957.41	27
transport-opt11	0.63	0.09	0.54	0.61	0.24	5.89	0.15	59.37	0.11	290.31	2622880.00	2253.51	20
visitall-opt11	0.12	0.00	0.04	0.05	0.01	0.56	0.00	5.77	0.00	28.07	71032400.00	3704.78	20
woodworking-opt08	0.89	0.16	0.57	1.44	0.35	13.94	0.13	140.83	0.07	685.86	4170080.00	4055.03	30
woodworking-opt08	1.28	0.15	0.69	1.33	0.27	13.21	0.17	130.82	0.07	664.08	5139070.00	4944.76	20

References

Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3:19, 2012.

Robert C Holte, Ariel Felner, Jack Newton, Ram Meshulam, and David Furcy. Maximizing over multiple pattern databases speeds up heuristic search. *Artificial Intelligence*, 170(16):1123–1136, 2006.