An Empirical Comparison of the FF (RPG) and Context-Enhanced Additive Heuristics for the Fast Downward Planner

Shivam Gandhi[1], **Thomas Herring**[2], **Reece Roberts**[3], **Kui Wu**[4], **Yuhan Xu**[5]
MSc Advanced Computing[2], MSc Artificial Intelligence[1,3,4], MSc Robotics[5]
Department of Informatics, Kings College London
{forename.surname}@kcl.ac.uk

Abstract

The Fast Downward planner supports a number of different heuristics to guide search. In this paper, we compare the admissible FF (RPG) heuristic with the inadmissible Context-Enhanced Additive heuristic. Guided by our hypotheses and prior research, we present and critically analyse our results, with the overall aim to ascertain the situational effectiveness of both.

Introduction

The Fast Forward (FF) Relaxed Planning Graph (RPG) heuristic, introduced by Hoffmann and Nebel (2001) was the first to consider delete relaxation to simplify a planning task. As an admissible heuristic, RPG is able to never overestimate the goal, however, has the downside of not being particularly informative.

Unlike the RPG heuristic, the Context-Enhanced Additive heuristic (Helmert and Geffner 2008) does not rely on creating a relaxed plan, but rather determines the heuristic value by recasting the Causal Graph Heuristic, h^{CG} as a variation of the Additive heuristic, h^{add}.

In this report we compare a number of features for both of the heuristics: number of problems solved, number of states expanded, total solution time, and heuristic calculation time. In general, the results we present here show that h^{FF} is best at finding an **optimal** plan, whereas, h^{cea} is superior in finding a plan quickly and efficiently, when one exists.

Background

FF (RPG) Heuristic

The FF heuristic h^{FF} works by generating a so called "relaxed plan" which does not consider delete actions as a standard planning graph would. Modifying the plan in such a way reduces the problem into a much simpler form that can be used to generate a heuristic value. This such value is calculated as the summation of the number of action layers which appear in the solution to the relaxed plan, denoted:

$$h(s) = \sum |O_i| \tag{1}$$

Since FF only draws upon a single attribute, i.e. the number of action layers in the relaxed plan, it is not particularly informative. However, this gives the heuristic an advantage in calculation speed, whilst also retaining admissibility; an area where the context-enhanced additive heuristic falls down.

Context-Enhanced Additive Heuristic

The causal graph heuristic works on providing the estimated cost of reaching the goal on the basis of two structures: Causal Graphs (CG) and Domain Transition Graphs (DTG). The causal graph heuristic, as it suggests, works on causal effects between state variables, whereas DTGs are responsible for finding the shortest path to the goal. The cost associated with each state variable is calculated using a modified version of Dijkstra's algorithm (Cormen, Leiserson, and Rivest 1990; Bertsekas 2000), which takes into account the causal effects of its parent variables. This procedure of measuring cost is neither optimal nor complete. Moreover, the process works only with acyclic graphs, and needs to be simplified using appropriate relaxations:

$$1 + \sum_{(v_i = e_i) \in z} \text{cost}_{v_i}\left(e_i', e_i\right) \tag{2}$$

The additive heuristic h^{add} provides a simplified view of estimating the heuristic value of reaching goal states from all internal states. Here the heuristic value of a state s is the sum of heuristic values of reaching all goal states from state s:

$$h^{\text{add}}(s) \stackrel{\text{def}}{=} \sum_{x \in s_\star} h^{\text{add}}(x \mid s) \tag{3}$$

Recursively, the heuristic value of reaching a particular goal state x from a state s is calculated as:

$$h^{\text{add}}(x \mid s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x \in s \\ \min\limits_{o: z \to x} (1 + \sum\limits_{x_i \in z} h^{\text{add}}(x_i \mid s)) & \text{if } x \notin s \end{cases} \tag{4}$$

The cost function of additive heuristic is true and simple, however, does not include causal effects captured of equation 2, i.e. in equation 4, the heuristic values of reaching states in conditions z of a goal state x from all other states, except s, are not taken into account.

We can generalise both the causal graph and additive heuristics, whilst also taking into account the causal effects

of other states on a particular variable in calculating heuristic as additive (contexts). Doing so yields a new heuristic: context-enhanced additive h^{cea}, whose value for a particular state s with goal state $s_\star$ is defined as:

$$h^{\text{cea}}(s) \stackrel{\text{def}}{=} \sum_{x \in s_\star} h^{\text{cea}}(x \mid x_s) \tag{5}$$

where x is the value of one of the variables in goal state, and x_s is the value of same variable in state s. This heuristic over a particular variable can be further represented as:

$$h^{\text{cea}}(x \mid x') \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x = x' \\ \min\limits_{0:x'',z \to x} (1 + h^{\text{cea}}(x'' \mid x') \\ \qquad + \sum\limits_{x_i \in z} h^{\text{cea}}(x_i \mid x_i')) & \text{if } x \neq x' \end{cases} \tag{6}$$

Breaking down the equation, the first term 1, is presumed as a unit cost of applying an action. The second term $h^{\text{cea}}(x'' \mid x'$ is the heuristic value of achieving the pre-condition value x'' of the variable from its initial value x'. The final term is the sum of heuristics of the remaining pre-conditions in a partial state $s(x'' \mid x')$ achieved as:

$$s(x'' \mid x') \stackrel{\text{def}}{=} \begin{cases} s[x'] & \text{if } x'' = x' \\ s(x''' \mid x')[z'][x'', y_1, \ldots, y_n] & \text{if } x'' \neq x' \end{cases} \tag{7}$$

where x''' is the value of variable achieved at one step before result of shortest path from x' to x''. Therefore, the state $x''' \mid x'$ is calculated recursively. For the current state to be true, all its other conditions (z') must be included and followed by the effects of the operator on the variables of those conditions i.e. $x'', y_1, ..., y_n$, hence the resulting state $s(x''' \mid x')[z', x'', y_1, \ldots, y_n]$.

Experiment Design

The comparison between h^{FF} and h^{cea} is particularly interesting due the two being complete opposites in terms of admissibility. When designing experiments, we ensured that a wide range of domains were selected to determine the effectiveness of not just the context-enhanced additive heuristic (although it was the focus), but also all heuristics that diverge from the more familiar relaxed planning graph approach.

Domain and Problem Selection

As discussed above, a wide range of domains were selected, for a total of 7. Experiments were carried out using the first 20 problems for each respective domain, as listed in the IPC "PDDL Benchmark Instances" GitHub repository (Potassco 2017).

Selection of domains was on the basis complexity of problem structure, diversity of applications and simplicity of the possible plans that would cover both our expectations, as well as allowing for unforeseen results. Since variants of domains can cause significant variation in the nature of problems, we tended to select propositional and non-temporal domains wherever possible, but with appropriate constraints applied. The following domains were selected:

1. **BLOCKSWORLD** Involves the stacking of blocks in a particular from any initial order. With no constraints, there is always a solution for this problem, but paths may vary significantly, which is where different heuristics may prove useful.
2. **TRUCKS** A basic package delivery domain where trucks move from one area to another, loading and unloading packages within given time constraints.
3. **DEPOTS** A combination of BLOCKSWORLD and TRUCKS. As part of TRUCKS domain, trucks are responsible for transportation of blocks from their source to destination, while blocks need to be stacked in a particular order at the destination, therefore giving the truck a similar role as the table in BLOCKSWORLD.
4. **AIRPORT** A domain that involves planning to regulate traffic at the airport in the simulation runs created using a tool called Astras (by Wolfgang Hatzack)
5. **PIPESWORLD** Works on a pipeline network controlling the flow of oil derivatives with a number of constraints like product compatibility and tankage restrictions. Creating plans for this domain can become complex due to the uncertainty of output based on any input within the pipeline.
6. **ELEVATOR** A simple domain involving the transportation of passengers from their current floor to their desired floor. Finding the optimal plan is necessary to avoid energy consumption, similarly to other domains. Finding the fastest plan is also essential in maintaining better customer satisfaction prevent unnecessary wasting of time, as is the case of elevators for VIP guests.
7. **GRIPPER** Involves a number of balls that need to be transferred from one room to another using a robot with 2 hands (or "grippers").

Hypotheses

In our investigation into the internal workings of each heuristic, we arrived at a number of hypotheses relating to the performance of each:

1. h^{cea} **should perform poorly with respect to the FF heuristic with a greater number of independent goals:**
   (a) As the heuristic of each goal is calculated individually, it does not take into account that multiple goals can be satisfied in the same path.
   (b) Significantly higher solution length and time taken for large numbers of independent goals
   (c) Always equal or better in terms of solution length and time with least number of independent goals.
2. **Calculation of the context-enhanced additive heuristic will often take much longer than the FF heuristic:**
   (a) It is recursive.
   (b) It involves the summation of 3 different terms (which each are quite complex in and of themselves).
   (c) One of the terms involves calculating the heuristic in a partial state, which itself needs to be calculated.
3. **Admissibility of h^{FF}:**
   (a) The heuristic only draws upon a single factor - being the number of action layers in the relaxed plan. Unlike

h^{cea}, the lack of multiple terms causes the heuristic to never overestimate the length to the goal.

4. **Informativeness of the context-enhanced additive heuristic:**

   (a) The calculated heuristic value for a goal is closer to the actual path length, since it takes into account the causal effects of other variables in reaching current goal variable.

   (b) The number of expanded states will always be smaller.

Alternative Configurations

When selecting our initial set of domains, we opted for those which have proven popular in other heuristic comparisons: BLOCKSWORLD, DEPOTS, TRUCKS, and PIPESWORLD. We began our testing on these domains, whilst analysing the inner-workings of the h^{cea}, and drawing comparisons with h^{FF}.

After familiarization and detailed analysis of h^{cea} and our results, we observed a significant performance differential when working with domains with high variance in the amount of goal independence. Following our experiments, and based on analysis of results so far, we proceeded to select three more promising domains which shared similar, yet subtly distinct qualities to our initial domains to reaffirm our findings: AIRPORT, ELEVATOR and GRIPPER.

These three domains work on the basis of movement of people / packages from location to location. The dependence of the constraints on each domain varies greatly:

- **AIRPORT** is the most interdependent of all, having to regulate high volumes of traffic.
- **ELEVATOR** is slightly less dependent, since people can travel in parallel, as well as generally being less busy than AIRPORT.
- **GRIPPER**, however, contains tasks that are all independent of each other.

The variance of independent tasks in GRIPPER should help going forward, especially in analysing the performance each heuristic using the basis of their methods of calculation.

Time, Memory Limits, and Specifications

For all tests, a time-limit of 1800 seconds (30 minutes), and a memory limit of 8192MB was imposed. These were found to be a good compromise between allowing plans to run to completion, whilst still giving a representative amount of data for later analysis. For all experiments A* search was used.

All tests were carried out on the same machine, with a 3.8GHz AMD Ryzen 9 3900X (hyperthreading disabled), and a total of 32GB RAM. This ensured maximum consistency across the dataset.

Presentation of Results

The table which follows shows three different results obtained in our experiment. We have defined the comparison of the heuristics by using ratios of h^{FF} to h^{cea}.

Domain	Solved	Expanded	T^{total}	T^{hcalc}
BLOCKSWORLD (20)	1.00	32.98	14.77	0.92
TRUCKS (20)	0.84	147.57	61.19	1.04
AIRPORT (20)	1.00	11.42	9.92	1.35
PIPESWORLD (20)	0.89	16.42	4.36	0.97
ELEVATOR (20)	1.00	1.08	1.18	1.35
DEPOTS (20)	1.15	46.31	12.02	0.90
GRIPPER (20)	1.00	3.14	0.65	0.92

The first column - The ratio of number of problems solved by h^{FF} compared to h^{cea}, shows that each can counter the other based on time and memory constraints, as well as the search strategy used. This is mainly because FF has more nodes within same depths of the search tree, and also the fact that takes FF takes a breadth-first search based strategy. However, the nodes of h^{cea} are at many different depths, therefore favouring a depth-first search strategy, which can penetrate deeper into the tree much more quickly and efficiently. Each has its own advantages and can counter each other in variance of problems and constraints.

The second column - The ratio of number of nodes expanded by h^{FF} compared to h^{cea}, proves our hypothesis that h^{cea} is more informative compared to its counterpart. This fact helps in significantly reducing expansion of states. We can draw interesting comparisons between the heuristics when looking at the ELEVATOR and GRIPPER domains, in which the number of state expansions between FF and CEA is very similar. This is likely due to the greater number of independent goals associated with these domains. As we already know, h^{cea} is expected to perform worse with a greater number of independent goals, and furthermore, it may tend to find a different path for each goal; ignoring the fact that a number of those independent goals can in fact lie on the same path.

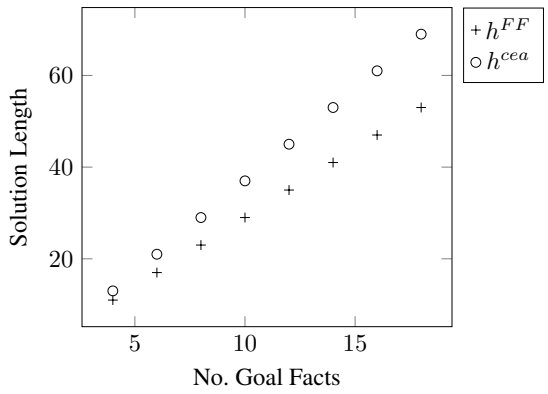
The third column - The ratio of time taken to solve problems by h^{FF} compared to h^{cea} somewhat continues our result from second column, with the fact that h^{cea} tends to find a different path for independent goals which may lie on same path. Hence, we can see with domains with more inter-related goals, it is much quicker with respect to its counterpart due to its more informative nature. Based on the results of AIRPORT, ELEVATOR and GRIPPER in which goal inter-dependency decreases gradually, from high inter-dependency in AIRPORT, to none in GRIPPER, we can clearly prove our second hypothesis, as the ratio decreases significantly from 9.92 to 1.18, even falling below 1 for GRIPPER.

The final column - The ratio of time taken to calculate the heuristic value by h^{FF} compared to h^{cea}, disproves our hypothesis as we see the ratio climbing above 1 for a few domains. Upon analysis, we have determined that heuristic calculation depends significantly on the number of variables involved. AIRPORT and ELEVATOR show significant variation in time, likely due to the fewer causal effects of from other variables leading to a non-recursive calculation.

Using T^{total}, we can prove another one of our hypotheses: that h^{cea} is never worse than h^{FF} in finding the optimal solution in the two cases where (i) there is only one goal, (ii) all the goals have to be satisfied sequentially one after

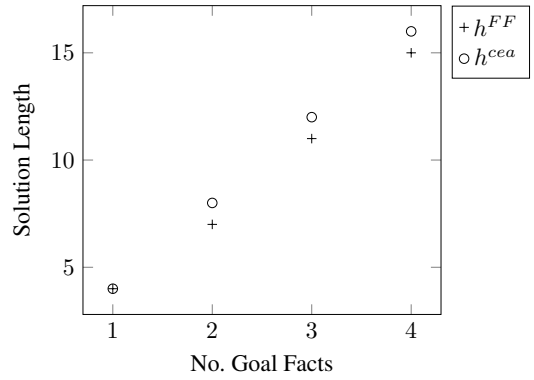
the other. To further analyse this, the following three graphs were produced, based on three domains which yielded interesting results relating to independent goals.

GRIPPER - Max Solution Length per No. Goal Facts



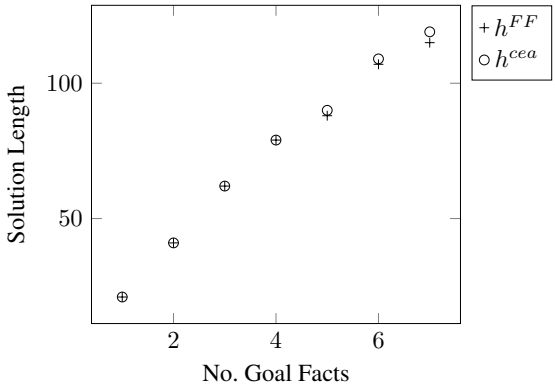
The GRIPPER domain, being the the only one with entirely independent goals, shows clear divergence of solution length with an increase in number of goal facts. h^{cea} tends to transfer each ball one-by-one to other room, without realizing two balls can be carried at once due to the difference in heuristic values. h^{FF} always opts to carry both at once since it has same heuristic value for achieving goals for both balls. h^{cea} unnecessarily adds the heuristic values of all goals which causes the node leading to first balls goal to be expanded much earlier than second balls goal.

ELEVATOR - Max Solution Length per No. Goal Facts



The ELEVATOR domain, having many scenarios relating transporting passengers efficiently to different floors, requires high amounts of cooperation. This such dependency still causes our graph to diverge, however, much less than GRIPPER. The node with a goal of making a passenger reach a floor in opposite direction is informed by the fact that all other passengers need to be dropped to their respective floors first in the current operating direction of the elevator.

AIRPORT - Max Solution Length per No. Goal Facts



The AIRPORT domain contains significant traffic control, where most tasks need to be executed in accordance with each other. This is a result of the fact that traffic is, the majority of the time, regulated sequentially.

When looking at the graphs, it is obvious that h^{cea} performs best when looking at GRIPPER and ELEVATOR due to the high amount of inter-dependent goal facts.

Conclusions

In our results and analysis, we were able to prove hypotheses 1, 3, and 4. Based on this, we can state that h^{cea} is much more efficient in terms of time and reducing the overhead of expanding the number of states; whilst h^{FF}, being admissible, can always give an optimal solution of the problem, providing a solution exists under the constraints. There is, however, an edge cases where h^{cea} can perform extremely poorly in terms of both time and plan length: when all, or most goal facts are independent of each other. In terms of finding the plan, both heuristics may outperform the other on the basis of specific time and memory constraints.

Our results did however highlight an anomaly in our hypotheses (number 2), where we stated that calculation of the context-enhanced additive heuristic will often take longer than FF. We suspect reason for this behaviour may be that domains have fewer causal effects between variables leading to a lack of recursion - the basis of our hypothesis.

In this paper we have been able to draw some particularly interesting comparisons between not just h^{FF} and h^{cea}, but also relaxed and non-relaxed approaches. From this, we have developed useful methods of comparison for heuristics with varying levels of admissibility.

References

Bertsekas, D. P. 2000. *Linear network optimization*. MIT Press.

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to algorithms*. MIT press.

Helmert, M., and Geffner, H. 2008. Unifying the causal graph and additive heuristics.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search.

Potassco. 2017. PDDL Benchmark Instances.