Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

An Empirical Study of Perfect Potential Heuristics

Augusto B. Corrêa, Florian Pommerening
University of Basel, Switzerland
{augusto.blaascorrea, florian.pommerening}@unibas.ch

Abstract

Potential heuristics are weighted functions over state features of a planning task. A recent study defines the complexity of a task as the minimum required feature complexity for a potential heuristic that makes a search backtrack-free. This gives an indication of how complex potential heuristics need to be to achieve good results in satisficing planning. However, these results do not directly transfer to optimal planning.

In this paper, we empirically study how complex potential heuristics must be to represent the perfect heuristic and how close to perfect heuristics can get with a limited number of features. We aim to identify the practical trade-offs between size, complexity and time for the quality of potential heuristics. Our results show that, even for simple planning tasks, finding perfect potential heuristics might be harder than expected.

Introduction

A simple potential heuristic (Pommerening et al. 2015) assigns a *weight* to every possible fact of the task. Its heuristic value for a state is simply the sum of all weights assigned to facts that are true in this state. More complex potential heuristics assign values to *state features* and sum the weights of all features that the state has. With sufficiently complex features, any finite function can be represented as a potential heuristic. It is thus interesting to investigate how much complexity is actually required for a specific function. For example, Seipp et al. (2016a) define the *correlation complexity* metric. It measures how large the features of a potential heuristic should be to guide a hill-climbing search to the goal without backtracking. They show that some popular domains in the planning literature only have correlation complexity two. However, the heuristics they describe are inadmissible, and the plans found with them can be suboptimal. Optimal planning requires admissible heuristics and little is known about their required complexity. Other related metrics like the *width* of a planning task (Geffner and Lipovetzky 2012) also consider only the difficulty in finding a plan without regarding its cost. If we are interested in the complexity necessary for optimal planning, these metrics do not help us.

Pommerening, Helmert, and Bonet (2017) show that computing admissible and consistent potential heuristics is tractable if their features consider at most two variables; but it becomes intractable if they consider three or more variables unless P equals NP. They also show that weights can be efficiently computed for larger features as long as the dependencies between features are limited. However, they did not study how much more informed the search would be when using larger features or how many features are required to achieve a good heuristic. Since considering only two variables excludes many more accurate potential functions, using a small number of larger features might pay off. But how many features are actually needed to achieve good heuristic values and how complex do they need to be?

In this paper, we empirically investigate *perfect potential heuristics*—potential heuristics that match the perfect heuristic. By studying how complex they need to be, we gain insight into the complexity of the task and also get an upper limit for the necessary complexity of any admissible potential heuristic. The results tell us whether tractable potential functions based on two-variable features are already informed enough or if it is worth to use more complex ones. We analyze the following trade-offs between quality and tractability: *How complex do features need to be in order to represent the perfect heuristic? How many complex features are necessary? Are the potential heuristics to solve a given planning task more complex than the ones to find out whether the task is solvable?*

We investigate those questions on planning tasks from the International Planning Competition (IPC) that are small enough to completely explore their full state space, compute their perfect heuristic values, and find perfect potential functions.

Planning Formalism

We consider planning tasks as factored transition systems representing the state space of the task. Any classical planning task can be represented as such a state space, independent of the language in which it is defined.

A *state space* is a transition system represented as a 6-

tuple $\mathcal{T} = \langle \mathcal{S}, L, T, c, s_I, S_* \rangle$, where $\mathcal{S}$ is a finite set of states, L is a set of transition labels, $T \subseteq \mathcal{S} \times L \times \mathcal{S}$ is a transition relation, $c : L \to \mathbb{R}^+$ is a cost function, $s_I \in \mathcal{S}$ is the initial state and $S_* \subseteq \mathcal{S}$ is a set of goal states. The states $\mathcal{S}$ are the complete assignments of a set of *state variables* $\mathcal{V}$ that map each variable $v \in \mathcal{V}$ to a value in its domain $dom(v)$. A *fact* is a pair $\langle v, d \rangle$ with $v \in \mathcal{V}$ and $d \in dom(v)$ and we can interpret a state s as a sets of facts $\langle v, s(v) \rangle$.

An *s-plan* from state s to s_n is a sequence of states $\langle s_0, \ldots, s_n \rangle$ where $s = s_0$, $s_n \in S_*$ and $(s_{i-1}, l, s_i) \in T$ for all $1 \leq i \leq n$ and some $l \in L$. The cost of an s-plan is the sum of $c(l)$ over all labels l used in the plan. If an *s_I-plan* exists in a state space, the planning task represented by it is called *solvable* and *unsolvable* otherwise.

A *heuristic* $h(s)$ maps a state $s \in \mathcal{S}$ to a number $n \in \mathbb{R} \cup \{\infty\}$, indicating the estimated cost of the cheapest s-plan. A heuristic value of ∞ indicates a state without any s-plan, also called an *unsolvable state*. The *perfect heuristic* h^* maps each state s to the cost of a cheapest s-plan.

A *feature* is a set of facts with pairwise different variables. We say a feature f is true in a state s if $f \subseteq s$. We also use the Iverson brackets (Knuth 1992) $[f \subseteq s]$, which have value 1 if $f \subseteq s$ and 0 otherwise. The *size* of the feature f is $|f|$ and we call features of size 1 and 2 *atomic* and *binary*.

Potential Heuristics

A *weight function* $w : \mathcal{F} \to \mathbb{R}$ associates a set of features $\mathcal{F}$ with weights. It induces a *potential heuristic* that maps each state s to the sum of weights for features of s:

$$h^{\text{pot}}_w(s) = \sum_{f \in \mathcal{F}} w(f)[f \subseteq s]$$

Since potential functions are necessarily finite, we represent heuristic functions with two potential functions where one captures all finite values and the other represents states with infinite values.

$$h^{\text{pot}}_{w_1,w_2}(s) = \begin{cases} \infty & \text{if } h^{\text{pot}}_{w_2}(s) > 0 \\ h^{\text{pot}}_{w_1}(s) & \text{otherwise.} \end{cases}$$

We say a potential heuristic h^{pot}_w *captures all unsolvable states* if $h^{\text{pot}}_w(s) > 0$ iff $h^*(s) = \infty$ and that it is *perfect on finite values* if $h^{\text{pot}}_w(s) = h^*(s)$ whenever $h^*(s) < \infty$. We say that $h^{\text{pot}}_{w_1,w_2}$ is a *perfect potential heuristic* if $h^{\text{pot}}_{w_1}$ is perfect on finite values and $h^{\text{pot}}_{w_2}$ captures all unsolvable states, i.e., if $h^{\text{pot}}_{w_1,w_2} = h^*$. In the rest of the paper, we look for feature sets and weight functions for such potential heuristics. Note that these are not unique as different weight functions and feature sets can lead to the same heuristic.

The *dimension* of a potential heuristic is the size of its largest feature. Intuitively, larger features are more flexible and allow us to express more fine-grained heuristics. With features of size $|\mathcal{V}|$ every finite heuristic can be expressed.

Analogously to Seipp et al. (2016a), we define the *optimal correlation complexity* of a planning task Π as the minimum dimension of a perfect heuristic for Π. Similarly, the optimal correlation complexity of a planning domain $\mathcal{D}$ is the maximal optimal correlation complexity of all tasks $\Pi \in \mathcal{D}$ or ∞ if such a maximum value does not exist.

Computing Perfect Potential Heuristics

To analyze the properties of perfect potential heuristics, we introduce methods that, given the perfect heuristic h^*, compute perfect potential heuristics with lowest dimension and number of features. Since these methods take h^* as an input, they cannot be used in practice to find good weights and we can only evaluate them on small tasks. We introduce them as tools to analyze the properties of such heuristics. We first focus on the solvable part of the state space and investigate potential heuristics that are perfect on finite values.

Minimal Dimension for Finite Values

Our main question is what feature size is required to represent a heuristic that is perfect on finite values. Given h^*, it is easy to find this out by iteratively testing if all finite values of h^* can be reproduced with features of size n, increasing n from 1 to $|\mathcal{V}|$. For each value of n, we generate the set $\mathcal{F}_n$ of all features of size $\leq n$, and check if there is a weight function $w_n : \mathcal{F}_n \to \mathbb{R}$ that satisfies $h^{\text{pot}}_{w_n}(s) = h^*(s)$ for all solvable states s.

Since $h^{\text{pot}}_{w_n}(s)$ is a linear expression over the weights, we can express the condition as a linear constraint:

$$\sum_{f \in \mathcal{F}_n} w(f)[f \subseteq s] = h^*(s) \qquad \text{for all solvable } s \in \mathcal{S}$$

We can use a linear program (LP) solver to find a solution to this system of equations where the weights $w(f)$ are the variables to optimize. If there is a feasible solution (independent of the objective function) we found a potential heuristic of dimension n that is perfect on finite values. Otherwise, we increase n and repeat the process. The method terminates at the latest with $n = |\mathcal{V}|$ since $\mathcal{S} \subseteq \mathcal{F}_{|\mathcal{V}|}$.

The first value n where we find a solution is the optimal correlation complexity of the task: if it would be possible to represent the finite values with smaller features we would have found a solution earlier.

As mentioned before, different potential functions with the same dimension can represent the finite values perfectly. Ideally, we would like to minimize the number of non-zero weights but this is hard to express in a linear program. Instead, we minimize the sum of the absolute weights $\sum_{f \in \mathcal{F}_n} |w(f)|$. We can do so by introducing a new variable β_f for each feature f and the two constraints

$$w(f) \leq \beta_f$$
$$-w(f) \leq \beta_f.$$

Minimizing $\sum_{f \in \mathcal{F}_n} \beta_f$ now minimizes the sum of absolute weights.

Minimal Error for Finite Values

It is possible that high-dimensional features are required to represent finite heuristic values perfectly but much smaller features are already sufficient to get reasonably close to perfect values. To explore this, we use another method to find weights: We start with an empty set of features and iteratively pick a single feature and a weight for it that minimizes the remaining error compared to h^*. In contrast to the

LP method, this greedy selection does not guarantee that the result has optimal dimension but empirically it terminates in more instances than the LP method (as shown later in the experimental results).

The *error* of a heuristic h in a state s is $E(h,s) = h^*(s) - h(s)$ and the total absolute error of h is $E(h) = \sum_{s\in\mathcal{S}} |E(h,s)|$. We want to add a feature f and a weight $w(f)$ to our heuristic that minimizes the total absolute error. Let $\mathcal{S}_f \subseteq \mathcal{S}$ be the set of states that have feature f and h^f be the potential heuristic h after adding feature f with weight $w(f)$. The total error of h^f is

$$\begin{aligned} E(h^f) &= \sum_{s\in\mathcal{S}} |E(h^f,s)| = \sum_{s\in\mathcal{S}} |h^*(s) - h^f(s)| \\ &= \sum_{s\in\mathcal{S}_f} |E(h,s) - w(f)| + \sum_{s\in\mathcal{S}\setminus\mathcal{S}_f} |E(h,s)|. \end{aligned}$$

The second part of this sum is the same for all values of $w(f)$ and the first part is minimal if $w(f)$ is a median of $\{E(h,s) \mid s \in \mathcal{S}_f\}$ (e.g., Schwertman, Gilks, and Cameron 1990). For a given feature, we can thus use the median error of all states with this feature as a weight.

We can find the best feature from a candidate set by computing the resulting error for each one and picking the one that minimizes the error. We initialize the set of candidate features to $\mathcal{F}_0 = \{\emptyset\}$ and continue to select the best feature from this set of candidates until no feature decreases the error anymore. We then expand the set of candidates with all features of the next higher size and repeat the process. As before, the process eventually terminates because the number of features to pick is limited by $|\mathcal{F}_{|\mathcal{V}|}|$.

Minimal Dimension for Infinite Values

The previous methods compute potential functions that are perfect on finite values. We now look at the remaining (unsolvable) part of the state space. The problem that we want to solve here is slightly different: Given the state space and a subset of states marked as unsolvable, find a compact function identifying this subset.

Similar to our first algorithm, we can specify the conditions a potential heuristic has to satisfy to detect all unsolvable states:

$$\begin{aligned} h(s) &\leq 0 \qquad \text{for all } s \in \mathcal{S} \text{ with } h^*(s) < \infty \\ h(s) &\geq 1 \qquad \text{for all } s \in \mathcal{S} \text{ with } h^*(s) = \infty \end{aligned}$$

Technically, we should require $h(s) > 0$ instead of $h(s) \geq 1$ but any weight function satisfying the first condition can be scaled by a sufficiently large constant to satisfy the second condition.

As before, $h(s)$ is a linear expression in the feature weights and we can thus check for a solution of these constraints with an LP solver. We again use an iterative procedure using all features of size n and increasing n when the equations have no solution.

Experiments

We implemented all the three methods described in the last section in the Fast Downward planning system (Helmert 2006). Since the methods assume that h^* is known, we selected only tasks where we could compute it using a time limit of 30 minutes and a memory limit of 3.5 GB. We first explore heuristics that are perfect on finite values. In total, there were 301 tasks over 38 different domains from the IPC optimal tracks from 1998 to 2018 where it is possible to compute h^* for the complete state space. We can consider a larger set of instances if we focus on the reachable part of the state space where we can compute h^* in 376 tasks over 47 domains. In those cases, we can find a potential heuristic that is perfect on the finite values in the reachable part of the state space but might be imperfect on unreachable states.

Methods to synthesize potential heuristics in the literature (e.g. Seipp, Pommerening, and Helmert 2015) so far have to consider global properties such as consistency and goal awareness in the complete state space and thus cannot focus only on reachable states. Yet, in practice a heuristic that is perfect on all reachable states would be sufficient and could be much easier to represent. It is thus interesting to analyze the properties of such heuristics as well. In most of the domains, our methods can only find perfect potential heuristics for the smallest instances available.

Domain	Lower Bound
`gripper`	7
`hiking-opt14`	6
`miconic`	7
`movie`	2
`nomystery-opt11`	5
`organic-synthesis-opt18`	6
`psr-small`	8
`rovers`	8
`satellite`	6
`scanalyzer-08`	5
`scanalyzer-opt11`	5
`storage`	5
`tpp`	5
`transport-opt08`	6
`vistall-opt11`	8
`zenotravel`	4
Avg. over all domains	**5.62**

Table 1: Lower bounds for the optimal correlation complexity. Showing only domains where the LP method finished for at least one instance.

Finite Values of the Complete State Space

For finite values, the LP-based method finds a heuristic with lower dimension than the greedy method for six instances. As mentioned earlier, both methods find potential heuristics that are perfect on finite values but only the former guarantees that the result has the optimal dimension. However, the LP-based method only terminated in 79 instances over 16 domains; while the greedy method terminated in 112 instances over 17 domains. (The only different domain was `logistics00`.) Most of the instances were solved within the first minutes. For instance, the greedy method terminated for 101 instances using 15 minutes or less.

Using the greedy method, the most complex feature found was in the domain `psr-small` and had a size of 11. Some

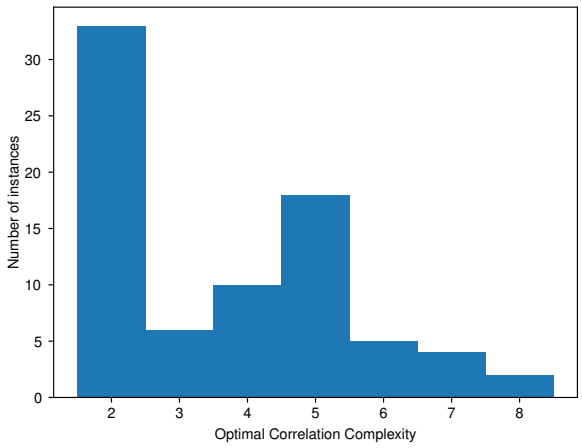

Figure 1: Distribution of optimal correlation complexity in instances where a perfect heuristic on finite values was computed.

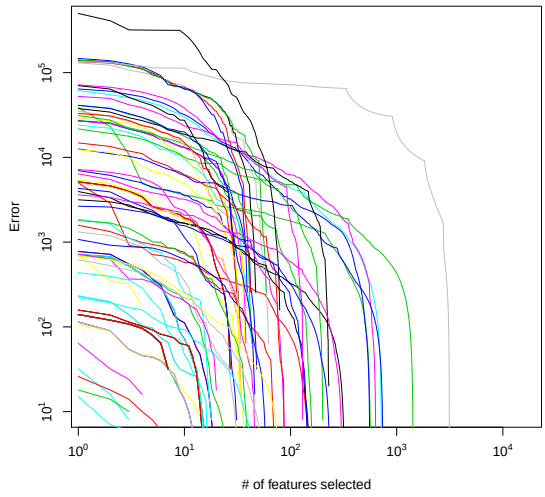

Figure 2: Decrease of total error rate E at each feature selection for the greedy method. (Plot in logarithmic scale.)

other domains also required very high-dimensional features: `rovers` and `visitall-opt11` had instances where the discovered heuristic has dimension 9, while instances in the domains `tpp`, `satellite`, `organic-synthesis`, `miconic`, and `storage` required functions with dimension 8.

The LP-based method establishes a lower bound for the optimal correlation complexity of the tested domains. Table 1 shows the lower bounds for each domain where at least one instance was completed. Almost all domains require complex features even for small instances. Interestingly, even domains considered easy, such as `gripper` and `visitall-opt11` have high optimal correlation complexity. Figure 1 shows the distribution of optimal correlation complexity lower bounds over all instances solved by the LP-based method. We also examined the correlation between the size of the state space and the optimal correlation complexity of the tasks. However, such correlation seems weak in our benchmark. For example, instance 03 of `psr-small` has a small state space (512 states) but needs features up to size 7; while `scanalyzer` and `storage` have instances with state spaces almost 10 times larger but only need features of size 5.

Our next experiment investigates how the dimension of a potential heuristic influences its overall quality. Figure 2 shows the error $E(h)$ computed by the greedy method after each iteration of the feature selection. They have a smooth decrease with sudden drops at the iterations where the set of candidate features increases. The first iterations for each feature size cause the largest reduction while the later iterations for a feature size seem to be mostly correcting noise. The most dramatic decrease in error often comes from a small number of large features. This is promising as it means that while we need large features to achieve high heuristic accuracy, a small number of large features often suffices.

We finally analyzed the number of required features in functions with minimal dimension using the LP-based method. Since this method focuses on minimizing the size of the features and not the number of features, the results serve only as upper bounds to the required number of features. Of the 79 instances, 56 need less than 100 features; in particular in domains such as `movie`, `organic-synthesis` and in both versions of `scanalyzer`. It is interesting to note that some instances use only a few but very complex features. For example, an instance in `psr-small` used less than 60 features but the largest one has size 7. The potential heuristic with the largest number of features was in `nomystery` and had 9802 features with non-zero weight.

An interesting case is the easiest instance of `gripper`: it is normally considered one of the easiest instances in the IPC but used 1856 features and dimension 7 to reproduce h^*. The reason for this is that the task contains many states which have a finite heuristic value but are not reachable. These states violate mutexes of the task and represent "nonsensical" information such as an agent being in two places at once. A heuristic that has to match h^* on such states as well is more restricted then one that does not and in the case of `gripper` many large features are necessary just to handle such states. We will now see if simpler functions can represent the finite values on only the reachable states.

Finite Values of the Reachable State Space

Perfect potential functions considering only the reachable state space of a task are indeed simpler than functions that are perfect on all finite values. As mentioned above, the first instance of `gripper` required a function with dimension 7 to be perfect on all finite values. When considering only reachable states, a dimension of 5 is sufficient, and the number of features used decreases from 1856 to 192. Almost all domains present such behavior. In particular, the largest difference occurs in the second instance of `rovers`: the dimension of the function encountered decreases from 8 to 5 and the number of features used from 900 to 26.

This decrease in complexity also means that we can find more potential functions that are perfect on all reachable

solvable states: the LP-based method finishes on 186 and the greedy method on 189 instances in this setting. The number of domains where we can generate results also increases in both cases: 16 to 23 with the LP-based method; and from 17 to 22 domains using the greedy method. The LP-based method still finds simpler functions than the greedy method, including in domains which were they previously ran out of resources. For example, considering only reachable states, our methods can find potential functions for the initial instances from `blocks`. While the greedy method finds a function with dimension 4 for the first three instances, the LP-based method has a solution with dimension 3.

Due to the way we compute h^* for the full search space, we have not considered domains with conditional effects and axioms so far, even though this is purely a limitation of the implementation. With the limitation to reachable states, we can compute h^* with an exhaustive forward search and can consider any ADL domain. We considered 19 ADL domains that were mostly designed for satisficing planning. As their instances are harder, we could only compute h^* for the reachable states in instances of four domains. Nonetheless, it is still possible to discuss some interesting results. For example, the ADL variant of `miconic` seems easier than its equivalent STRIPS variant. The maximum complexity is 4 for the 38 instances of the ADL variant that we could solve. In the STRIPS version, we could find h^* in only 20 instances but needed a complexity of 6. We speculate that the more compact representation in the ADL variant allows less complex potential heuristics.

Infinite Values

There are 29 domains with instances containing unsolvable states. In 57 instances over 14 domains, we can compute potential functions that capture all such states. These functions are generally simpler than those that are perfect on finite values. For example, `psr-small` has instances where we can capture all unsolvable states using atomic features; while we could not reproduce h^* for this domain using features with size smaller than 5. The instances with highest dimension for capturing unsolvable states occurred in `blocks`, `openstacks` and `tpp`, with dimension 3. One reason why this number is not lower is because h^* captures all unsolvable states in the state space, including ones that are unreachable and cannot be represented with smaller features. The low dimension required to detect unsolvable states could be an effect of considering only small instances but it is possible that most IPC domains have unsolvable states easily identified by small features. Further evidence for this is the good performance of Aidos (Seipp et al. 2016b) in the unsolvability planning competition. Among other components, Aidos computes many small pattern databases, extracts unsolvable states identified by them and declares a state unsolvable if it abstracts to one of the stored states. This can be viewed as defining a potential function that uses the extracted abstract states as features.

Conclusions

We analyzed the characteristics of perfect potential heuristics using instances of the International Planning Competition (IPC). In particular, we investigated the dimension needed for potential heuristics to match h^*. Our results show that even easy domains need potential functions with high dimension. While this sounds like bad news, we also showed that a small number of large features are often responsible for the largest reduction in heuristic error; while adding multiple features of same size does not cause the same increase of information.

In the future, it would be interesting to investigate which features add the most information. If we could automatically identify an interesting subset of high-dimensional features, finding good weights for them would be possible with the fixed-parameter tractable algorithm by Pommerening, Helmert, and Bonet (2017).

Acknowledgments

We have received funding for this work from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 817639).

References

Geffner, H., and Lipovetzky, N. 2012. Width and serialization of classical planning problems. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, 540–545. IOS Press.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Knuth, D. E. 1992. Two notes on notation. *The American Mathematical Monthly* 99(5):403–422.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From non-negative to general operator cost partitioning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3335–3341. AAAI Press.

Pommerening, F.; Helmert, M.; and Bonet, B. 2017. Higher-dimensional potential heuristics for optimal classical planning. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence (AAAI 2017)*. AAAI Press.

Schwertman, N. C.; Gilks, A. J.; and Cameron, J. 1990. A simple noncalculus proof that the median minimizes the sum of the absolute deviations. *The American Statistician* 44(1):38–39.

Seipp, J.; Pommerening, F.; Röger, G.; and Helmert, M. 2016a. Correlation complexity of classical planning domains. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3242–3250.

Seipp, J.; Pommerening, F.; Sievers, S.; Wehrle, M.; Fawcett, C.; and Alkhazraji, Y. 2016b. Fast Downward Aidos. In *Unsolvability International Planning Competition: planner abstracts*, 28–38.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New optimization functions for potential heuristics. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 193–201. AAAI Press.