On Using Admissible Bounds for Learning Forward Search Heuristics

Carlos Núñez-Molina[1] , **Masataro Asai**[2] , **Pablo Mesejo**[1] and **Juan Fernández-Olivares**[1]
[1]University of Granada, Spain
[2]MIT-IBM Watson AI Lab, USA
ccaarlos@ugr.es, masataro.asai@ibm.com, pmesejo@ugr.es, faro@decsai.ugr.es

Abstract

In recent years, there has been growing interest in utilizing modern machine learning techniques to learn heuristic functions for forward search algorithms. Despite this, there has been little theoretical understanding of *what* they should learn, *how* to train them, and *why* we do so. This lack of understanding has resulted in the adoption of diverse training targets (suboptimal vs optimal costs vs admissible heuristics) and loss functions (e.g., square vs absolute errors) in the literature. In this work, we focus on how to effectively utilize the information provided by admissible heuristics in heuristic learning. We argue that learning from poly-time admissible heuristics by minimizing mean square errors (MSE) is not the correct approach, since its result is merely a noisy, inadmissible copy of an efficiently computable heuristic. Instead, we propose to model the learned heuristic as a *truncated* gaussian, where admissible heuristics are used not as training targets but as lower bounds of this distribution. This results in a different loss function from the MSE commonly employed in the literature, which implicitly models the learned heuristic as a gaussian distribution. We conduct experiments where both MSE and our novel loss function are applied to learning a heuristic from optimal plan costs. Results show that our proposed method converges faster during training and yields better heuristics.

1 Introduction

Motivated by the success of Machine Learning (ML) approaches in various decision making tasks [Mnih *et al.*, 2015; Silver *et al.*, 2016], an increasing number of papers are tackling the problem of learning a heuristic function for forward state space search in recent years. Despite this interest, there has been little theoretical understanding of *what* these systems should learn, *how* to train them and *why* we do so. As a result, heuristic learning literature has adopted many different training targets (corresponding to either admissible heuristics [Shen *et al.*, 2020], suboptimal solution costs [Arfaee *et al.*, 2011; Ferber *et al.*, 2022; Marom and Rosman, 2020] or optimal solution costs [Ernandes and Gori, 2004; Shen *et al.*, 2020]) and training losses (e.g., square errors [Shen *et al.*, 2020], absolute errors [Ernandes and Gori, 2004] and piecewise absolute errors [Takahashi *et al.*, 2019]).

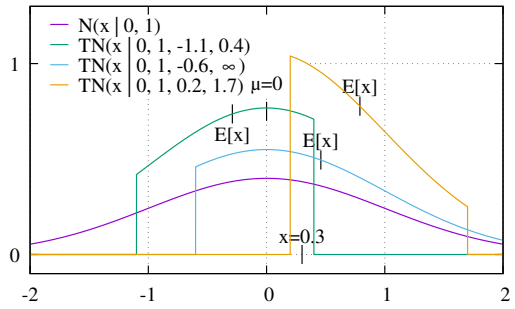

Figure 1: The probability density functions (PDFs) of Truncated Gaussian distributions $p(\mathrm{x}) = \mathcal{TN}(\mu = 0, \sigma = 1, l, u)$ with several lower/upper bounds (l, u). In the heuristic learning setting, x is the optimal solution cost h^* sampled from the dataset and $\mu = \mu_\theta(s)$ is the prediction associated with a state s. The $(l, u) = (0.2, 1.7)$ variant (yellow) shows that the mean $\mathbb{E}_{p(\mathrm{x})}[\mathrm{x}]$, which we use as the search heuristic, respects the bounds (l, u) even when the predicted $\mu = 0$ lies outside (l, u).

In this work, we try to answer these questions from a statistical lens, focusing on how to effectively utilize admissible heuristics in the context of heuristic learning. We argue that learning from poly-time admissible heuristics, such as h^{LMcut} [Helmert and Domshlak, 2009], by minimizing mean square errors (MSE) does not provide any practical benefits, since its result is merely a noisy, inadmissible copy of a heuristic that is already efficient to compute. Then, if admissible heuristics should not be used as training targets, how can we leverage them? In order to answer this question, we first analyze the statistical implications behind the commonly used loss function, the MSE, which implicitly models the learned heuristic as a Gaussian distribution. Nonetheless, we contend that a better modeling choice for heuristics is given by the *Truncated* Gaussian distribution (Fig. 1), due to the existence of *bounds* on the values a heuristic can take (e.g., heuristics never take on negative values).

The main contribution of this paper is a theoretically-motivated, statistical method for learning an inadmissible heuristic while exploiting an admissible heuristic. We pro-

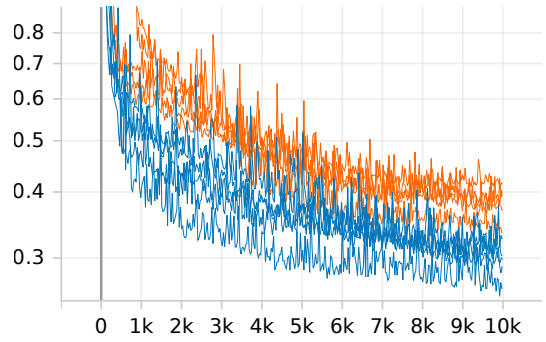

Figure 2: Comparison of the training curve (x-axis: training step) for the validation MSE loss (y-axis, logarithmic) between Gaussian (orange) and Truncated Gaussian (blue) models, independent runs recorded on 5 random seeds each. The losses converge faster for the latter due to the additional information provided by the admissible lower bound $l = h^{\mathrm{LMcut}}$.

pose to model the learned heuristic as a Truncated Gaussian, where an admissible heuristic provides the lower bound of this distribution, thus constraining heuristic predictions. This modeling choice results in a loss function to be minimized that is different from the standard MSE loss. We conduct extensive experimentation where both loss functions are applied to learning heuristics from optimal plan costs in several classical planning domains. Results show that those methods which model the learned heuristic as a Truncated Gaussian, i.e., which are trained with our novel loss function, learn faster and result in better heuristics than those which model it as an ordinary Gaussian, i.e., which are trained with the standard MSE loss. To the best of our knowledge, this is the first work that proposes the use of admissible heuristics to constrain heuristic predictions and improve learning.[1]

2 Backgrounds

2.1 Classical Planning and Heuristics

We define a propositional STRIPS Planning problem as a 4-tuple $\langle P, A, I, G\rangle$ where P is a set of propositional variables, A is a set of actions, $I \subseteq P$ is the initial state, and $G \subseteq P$ is a goal condition. Each action $a \in A$ is a 4-tuple $\langle \mathrm{PRE}(a), \mathrm{ADD}(a), \mathrm{DEL}(a), \mathrm{COST}(a)\rangle$ where $\mathrm{COST}(a) \in \mathbb{Z}^{0+}$ is a cost, $\mathrm{PRE}(a) \subseteq P$ is a precondition and $\mathrm{ADD}(a), \mathrm{DEL}(a) \subseteq P$ are the add-effects and delete-effects, respectively. A state $s \subseteq P$ is a set of true propositions (all of $P \setminus s$ are false), an action a is *applicable* when $s \supseteq \mathrm{PRE}(a)$ (read: s *satisfies* $\mathrm{PRE}(a)$), and applying action a to s yields a new successor state $a(s) = (s \setminus \mathrm{DEL}(a)) \cup \mathrm{ADD}(a)$.

The task of classical planning is to find a sequence of actions called a *plan* $(a_1, \cdots, a_n)$ where, for $1 \leq t \leq n$, $s_0 = I$, $s_t \supseteq \mathrm{PRE}(a_{t+1})$, $s_{t+1} = a_{t+1}(s_t)$, and $s_n \supseteq G$. A plan is *optimal* if there is no plan with lower *cost-to-go* $\sum_t \mathrm{COST}(a_t)$. A plan is otherwise called *satisficing*. In this paper, we assume unit-cost: $\forall a \in A; \mathrm{COST}(a) = 1$.

[1]Our full code and data can be found in github.com/pddl-heuristic-learning/pddlsl. We provide the Appendix in the Arxiv version of our paper: arxiv.org/abs/2308.11905.

A domain-independent heuristic function h in classical planning is a function of a state s and the problem $\langle P, A, I, G\rangle$. It returns an estimate of the shortest (optimal) path cost from s to one of the goal states (states that satisfy G), typically through a symbolic, non-statistical means such as *delete-relaxation*, a technique that ignores the delete-effects of actions in order to efficiently estimate the cost from s to G. The optimal cost-to-go, or a *perfect heuristic*, is denoted by h^*. A heuristic is called *admissible* if it never overestimates it, i.e., $\forall s; 0 \leq h(s) \leq h^*(s)$, and *inadmissible* otherwise. Notable admissible heuristics include h^{LMcut}, $h^{\max}$ and h^+ [Helmert and Domshlak, 2009; Bonet and Geffner, 2001; Betz and Helmert, 2009], whereas h^{FF}, h^{add} and h^{GC} [Hoffmann and Nebel, 2001; Bonet and Geffner, 2001; Fikes *et al.*, 1972] are prominent examples of inadmissible heuristics.

2.2 Task: Supervised Learning for Heuristics

Let $p^*(\mathrm{x})$ be the unknown ground-truth probability distribution of (an) observable random variable(s) x and let $p(\mathrm{x})$ be our current estimate of it. Given a dataset $\mathcal{X} = \{x^{(1)}, \ldots, x^{(N)}\}$ of N data points, we denote an empirical data distribution as $q(\mathrm{x})$, which draws samples from $\mathcal{X}$ uniformly. While often $q(\mathrm{x})$ may also be informally called a ground-truth distribution, $q(\mathrm{x})$ is entirely different from either $p(\mathrm{x})$ or $p^*(\mathrm{x})$ because it is a distribution over a finite set of points, i.e., a uniform mixture of dirac's delta δ distributions (Eq. 1). Our goal is to obtain an estimate $p(\mathrm{x})$ that resembles $p^*(\mathrm{x})$ as closely as possible. To do so, under the *Maximum Likelihood Estimation* (MLE) framework, we maximize the expectation of $p(\mathrm{x})$ over $q(\mathrm{x})$. In other words, MLE tries to maximize the expected probability $p(x)$ of observing each data point $x \sim q(\mathrm{x})$:

$$q(\mathrm{x}) = \sum_{\mathrm{i}} q(\mathrm{x}|\mathrm{i})q(\mathrm{i}) = \sum_{i=1}^{N} \delta(\mathrm{x} = x^{(i)}) \cdot \frac{1}{N} \tag{1}$$

$$\begin{aligned} p^*(\mathrm{x}) &= \arg\max_p \mathbb{E}_{q(\mathrm{x})} p(\mathrm{x}) \\ &= \arg\min_p \mathbb{E}_{q(\mathrm{x})} - \log p(\mathrm{x}) \end{aligned} \tag{2}$$

Typically, we assume $p^*(\mathrm{x})$ and $p(\mathrm{x})$ are of the same family of functions parameterized by θ, such as a set of neural network weights or the trees in random forests, i.e., $p^*(\mathrm{x}) = p_{\theta^*}(\mathrm{x})$, $p(\mathrm{x}) = p_\theta(\mathrm{x})$. This makes MLE a problem of finding the θ maximizing $\mathbb{E}_{q(\mathrm{x})} p_\theta(\mathrm{x})$. Also, we typically minimize a *loss* such as the *negative log likelihood* (NLL) $-\log p(\mathrm{x})$, since $\log$ is monotonic and preserves the optima θ^* (Eq. 2). Furthermore, $\mathbb{E}_{q(\mathrm{x})} \ldots$ is often estimated by Monte-Carlo sampling, e.g., $\mathbb{E}_{q(\mathrm{x})} - \log p(\mathrm{x}) \approx \frac{1}{N} \sum_{i=1}^{N} - \log p(x_i)$, where each x_i is sampled from $q(\mathrm{x})$.

We further assume $p(\mathrm{x})$ to follow a specific distribution such as a Gaussian distribution $\mathcal{N}(\mu, \sigma)$:

$$p(\mathrm{x}) = \mathcal{N}(\mathrm{x} \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mathrm{x}-\mu)^2}{2\sigma^2}}. \tag{3}$$

We emphasize that **the choice of the distribution determines the loss**. When the model designer assumes $p(\mathrm{x}) = \mathcal{N}(\mu, \sigma)$, then the NLL is a shifted and scaled squared error:

$$-\log p(\mathrm{x}) = \frac{(\mathrm{x}-\mu)^2}{2\sigma^2} + \log\sqrt{2\pi\sigma^2}. \tag{4}$$

Likewise, a Laplace distribution $L(\mathrm{x}|\mu, b) = \frac{1}{2b}e^{-\frac{|\mathrm{x}-\mu|}{b}}$ represents the absolute error because its NLL is $\frac{|\mathrm{x}-\mu|}{b} + \log 2b$.

The NLL loss is thus more fundamental and theoretically grounded than losses such as the Mean Squared Error, although it is "more complicated" due to the division $\frac{1}{2\sigma^2}$ and the second term. A reader unfamiliar with statistics may rightfully question why such complications are necessary or why σ is not commonly used by the existing literature. It is because many applications happen to require only a single prediction for a single input (*point estimate*): When we model the output distribution as a Gaussian $\mathcal{N}(\mu, \sigma)$, we often predict μ, which is simultaneously the mean and the mode of the distribution and does not depend on σ.

Moreover, the MSE is a special case of the NLL that can be derived from it. To derive the MSE, we first simplify the loss into the squared error $(x-\mu)^2$ by setting σ to an arbitrary constant, such as $\sigma = \frac{1}{\sqrt{2}}$, because the variance/spread of the prediction does not matter in a point estimation of μ. As a result, we can also ignore the second term which is now a constant. We then compute the expectation $\mathbb{E}_{q(\mathrm{x})}(\mathrm{x}-\mu)^2$ with a Monte-Carlo estimate that samples N data points $x_1, \dots x_N \sim q(\mathrm{x})$, predict $\mu = \mu_\theta(x_i)$ for each x_i using a machine learning model μ_θ, and compute the average: $\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_\theta(x_i))^2$. In other words, **the MSE loss is nothing more than the Monte-Carlo estimate of the NLL loss of a Gaussian with a fixed** $\sigma = \frac{1}{\sqrt{2}}$. In contrast, *distributional estimates* represent the entire $p(\mathrm{x})$; e.g., if $p(\mathrm{x}) = \mathcal{N}(\mu, \sigma)$, then the model predicts both μ and σ.

The MLE framework can be applied to the supervised heuristic learning setting as follows. Let $q(\mathrm{s}, \mathrm{x})$ be the empirical data distribution, where s is a random variable representing a state-goal pair (from now on, we will implicitly assume that states s also contain goal information) and x a random variable representing the cost-to-go (regardless of whether it corresponds to a heuristic estimate, optimal or suboptimal cost). Then, the goal is to learn $p^*(\mathrm{x} \mid \mathrm{s})$ where:

$$p^*(\mathrm{x} \mid \mathrm{s}) = \arg\max_\theta \mathbb{E}_{q(\mathrm{s},\mathrm{x})} p_\theta(\mathrm{x}|\mathrm{s}), \tag{5}$$

$$p_\theta(\mathrm{x} \mid \mathrm{s}) = \mathcal{N}(\mathrm{x} \mid \mu = \mu_\theta(\mathrm{s}), \sigma = \tfrac{1}{\sqrt{2}}), \tag{6}$$

and $\mu_\theta(\mathrm{s})$ is the main body of the learned model, such as a neural network parameterized by the weights θ. Supervised heuristic learning with distributional estimates is formalized similarly, where the only difference is that an additional model (e.g. a neural network) with parameters θ_2 predicts σ:

$$p_{(\theta_1,\theta_2)}(\mathrm{x} \mid \mathrm{s}) = \mathcal{N}(\mathrm{x} \mid \mu = \mu_{\theta_1}(\mathrm{s}), \sigma = \sigma_{\theta_2}(\mathrm{s})). \tag{7}$$

2.3 The Principle of Maximum Entropy

The discussion above models $p(\mathrm{x})$ as a Gaussian distribution. While the assumption of normality (i.e., following a Gaussian) is ubiquitous, one must be able to justify such an assumption. The *principle of maximum entropy* [Jaynes, 1957] states that $p(\mathrm{x})$ should be modeled as the maximum entropy (*max-ent*) distribution among all those that satisfy our constraints or assumptions, where the entropy is defined as $\mathbb{E}_{p(x)}\langle -\log p(x)\rangle$. A set of constraints defines its corresponding max-ent distribution which, being the *most random* among those that satisfy those constraints, minimizes assumptions other than those associated with the given constraints. Conversely, a non max-ent distribution implicitly encodes additional or different assumptions that can result in an accidental, potentially harmful bias. For example, if we believe that our random variable x has a finite mean, a finite variance and a support/domain/range equal to $\mathbb{R}$, then it *must* be modeled as a Gaussian distribution according to this principle because it is the max-ent distribution among all those that satisfy these three constraints.

In other words, a person designing a loss function of a machine learning model must devise a reasonable set of constraints on the target variable x to identify the max-ent distribution $p(\mathrm{x})$ of the constraints, which *automatically* determines the *correct* NLL loss for the model. This paper tries to follow this principle as faithfully as possible.

3 Utilizing Bounds for Learning

In the previous section, we provided some statistical background on heuristic learning. We now leverage this background to analyze many of the decisions taken in the existing literature, sometimes unknowingly, putting particular focus on how admissible heuristics are used during training. Based on this analysis, we argue that the proper way of utilizing the information provided by admissible heuristics is using them as the lower bound of a Truncated Gaussian distribution representing the learned heuristic.

We previously explained that the heuristic to be learned is modeled as a probability distribution (e.g., a Gaussian), instead of a single value: The ML model is unsure about the true heuristic value h^* associated with a state s. When it predicts μ, it believes not only that μ is the most likely value (the *mode*) for h^*, but also that other values are still possible. The uncertainty of this prediction is given by σ: The larger this parameter is, the more unsure the model is about its prediction. The commonly used MSE loss is derived from the ad-hoc assumption that σ is fixed, i.e., independent from s, which means that the model is equally certain (or uncertain) about h^* for every state s. This is unrealistic in most scenarios: it is generally more difficult to accurately predict h^* for states that are further from the goal, for which the uncertainty should be larger. Therefore, the model should predict σ in addition to μ, i.e., it should output a distributional estimate of h^* instead of a point estimate.

Another crucial decision involves selecting *what* to learn, i.e., the target / ground truth to use for the training. It is easy to see that training a model on a dataset containing a practical (i.e., computable in polynomial time) heuristic, admissible or otherwise, such as h^{LMcut} or h^{FF}, does not provide any practical benefits because, even if the training is successful, all we get is a noisy, lossy, slow copy of a heuristic that is already efficient to compute. Worse, trained models always lose admissibility if the target is admissible. To outperform existing poly-time heuristics, i.e., achieve a *super-symbolic benefit* from learning, it is imperative to train the model on data of a better quality, such as h^+ as proposed in [Shen *et al.*, 2020] or optimal solution costs h^*. Although obtaining these datasets may prove computationally expensive in practice, e.g., h^+ is NP-hard, we can aspire to learn a heuristic

that outperforms the poly-time heuristics by training on these targets.

If poly-time admissible heuristics are not ideal training targets, are they completely useless for learning a heuristic? Intuitively this should not be the case, given the huge success of heuristic search where they provide a strong search guidance toward the goal. Our main question is then *how* we should exploit the information they provide. To answer this question, we must revise the assumption we previously made by using squared errors: That $\mathrm{x} = h^*$ follows a Gaussian distribution $\mathcal{N}(\mu, \sigma)$. The issue with this assumption is that $\mathcal{N}(\mu, \sigma)$ assigns a non-zero probability $p(x)$ to every $x \in \mathbb{R}$, but we actually know that h^* cannot take some values: Given some admissible heuristic like h^{LMcut}, we know that $h^{\mathrm{LMcut}} \leq h^*$ holds for every state; therefore $p(x) = 0$ when $x < h^{\mathrm{LMcut}}$. Analogously, if for some state s we know the cost h^{sat} of a satisficing (non-optimal) plan from s to the goal, then h^{sat} acts as an *upper bound* of h^*.

According to the principle of maximum entropy, which serves as our *why*, if we have a lower l and upper u bound for h^*, then we should model h^* using the max-ent distribution with finite mean, finite variance, and a support equal to (l, u), which is the *Truncated Gaussian* distribution $\mathcal{TN}(\mathrm{x}|\mu, \sigma, l, u)$ as proven in [Dowson and Wragg, 1973], formalized as Eq. 8:

$$\mathcal{TN}(\mathrm{x}|\mu, \sigma, l, u) = \begin{cases} \frac{1}{\sigma} \frac{\phi(\frac{\mathrm{x}-\mu}{\sigma})}{\Phi(\frac{u-\mu}{\sigma}) - \Phi(\frac{l-\mu}{\sigma})} & l \leq \mathrm{x} \leq u \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where $\phi(\mathrm{x}) = \frac{1}{\sqrt{2\pi}} \exp \frac{\mathrm{x}^2}{2}$, $\Phi(\mathrm{x}) = \frac{1}{2}(1 + \mathrm{ERF}(\mathrm{x}))$,

l is the lower bound, u is the upper bound, μ is the pre-truncation mean, σ is the pre-truncation standard deviation, and ERF is the error function. $\mathcal{TN}$ has the following NLL loss:

$$-\log \mathcal{TN}(\mathrm{x}|\mu, \sigma, l, u) = \frac{(\mathrm{x}-\mu)^2}{2\sigma^2} + \log \sqrt{2\pi\sigma^2} + \log \Big(\Phi\big(\tfrac{u-\mu}{\sigma}\big) - \Phi\Big(\tfrac{l-\mu}{\sigma}\Big) \Big). \tag{9}$$

Modeling h^* as a $\mathcal{TN}$ instead of $\mathcal{N}$ presents several advantages. Firstly, $\mathcal{TN}$ constrains heuristic predictions to lie in the range (l, u) given by the bounds of the distribution. Secondly, $\mathcal{TN}$ generalizes $\mathcal{N}$ as $\mathcal{TN}(\mu, \sigma, -\infty, \infty) = \mathcal{N}(\mu, \sigma)$ when no bounds are provided. Finally, $\mathcal{TN}$ opens the possibility for a variety of training scenarios for heuristic learning, with a sensible interpretation of each type of data, including the satisficing solution costs.

In this work, we focus on the scenario where an admissible heuristic h is provided along with the optimal solution cost h^* for each state, leaving other settings for future work. In this case, h acts as the lower bound l of h^*, which is modeled as a $\mathcal{TN}(\mathrm{x} = h^*|\mu, \sigma, h, \infty)$, where μ and σ are predicted by an ML model. Note that we cannot use h^* as $\mathcal{TN}(h^*|\mu, \sigma, h^*, h^*)$ since, during evaluation/test time, we do not have access to the optimal cost h^*. Also, this modeling decision is feasible even when no admissible heuristic is available (e.g., when the PDDL description of the environment is not known, as in Atari games [Bellemare *et al.*, 2013]) since we can always resort to the blind heuristic $h^{\mathrm{blind}}(s)$ or simply do $l = 0$, which still results in a tighter bound than the one provided by an untruncated Gaussian $\mathcal{N}(\mu, \sigma) = \mathcal{TN}(\mu, \sigma, -\infty, \infty)$.

Finally, our setting is orthogonal and compatible with *residual learning* [Yoon *et al.*, 2008], where the ML model does not directly predict μ but rather a *residual* or offset $\Delta\mu$ over a heuristic h, where $\mu = h + \Delta\mu$. Residual learning can be seen as initializing the model output μ around h which, when h is a good *unbiased estimator* of h^*, facilitates learning. This technique can be used regardless of whether h^* is modeled as a $\mathcal{TN}$ or $\mathcal{N}$ because it merely corresponds to a particular implementation of $\mu = \mu_\theta(s)$, which is used by both distributions. Residual learning is analogous to the data normalization commonly applied in standard regression tasks, where features are rescaled and shifted to have mean 0 and variance 1. However, residual learning is superior in the heuristic learning setting because target data (e.g., h^*) is skewed above 0 and because the heuristic used as the basis for the residual can handle out-of-distribution data due to its symbolic nature.

3.1 Planning with a Truncated Gaussian

At planning time, we must obtain a point estimate of the output distribution, which will be used as a heuristic to determine the ordering between search nodes. As a point estimate, we can use any statistic of central tendency, thus we choose the mean. It is important to note that the μ parameter of $\mathcal{TN}(\mu, \sigma, l, u)$ is *not* the mean of this distribution since μ corresponds to the mean of $\mathcal{N}(\mu, \sigma)$ (i.e., the mean of the distribution *before truncation*) and does not necessarily lie in the interval (l, u). The mean of a Truncated Gaussian is obtained according to Eq. 10. Note that a naive implementation of this formula results in rounding errors (See the Appendix for a numerically stable implementation).

$$\mathbb{E}[\mathrm{x}] = \mu + \sigma \frac{\phi(\frac{l-\mu}{\sigma}) - \phi(\frac{u-\mu}{\sigma})}{\Phi(\frac{u-\mu}{\sigma}) - \Phi(\frac{l-\mu}{\sigma})} \tag{10}$$

Eq. 10 satisfies $l \leq \mathbb{E}[\mathrm{x}] \leq u$. This means that, when a lower bound l is provided (e.g., by an admissible heuristic), the heuristic prediction returned by the model will never be smaller than l. Analogously, when an upper bound u is also provided (e.g., by a satisficing solution cost), the model will never predict a heuristic value larger than u. With this, we hope that the use of a $\mathcal{TN}$ during planning helps the model make predictions that are closer to h^* than the bounds themselves, potentially helping it achieve a super-symbolic improvement over admissible heuristics.

In contrast, the mode $\arg\max_x p(x)$ of $\mathcal{TN}$ is uninteresting: While we could use it as another point estimate, it is the same as the untruncated mean μ when the predicted μ is within the bounds, and equal to one of the upper/lower bounds otherwise (see Fig. 1). However, this inspires a naive alternative that is applicable even to $\mathcal{N}$, which is to clip the heuristic prediction $\mathbb{E}[\mathrm{x}]$ (equal to μ for $\mathcal{N}$) to the interval $[l, u]$. We expect only a marginal gain from this trick because it only improves *really bad* predictions, i.e., those which would lie outside $[l, u]$ otherwise, and does not affect predictions that correctly lie inside $[l, u]$. In our experiments, we show that this approach is inferior to our first method.

We re-emphasize that despite the use of admissible heuristics during training the learned heuristic is inadmissible, just like any learning-based heuristics proposed so far. In case a distributional estimate is used, i.e., when the ML model also learns to predict σ, we could discuss *likely-admissibility* [Ernandes and Gori, 2004; Marom and Rosman, 2020]. However, this extension is left for future work.

4 Experimental Evaluation

We evaluate the effectiveness of our new loss function under the domain-specific generalization setting, where the learned heuristic function is required to generalize across different problems of a single domain. Due to space limitations, we focus on the high-level descriptions and describe the detailed configurations in the Appendix.

Data Generation. We trained our system on four classical planning domains: blocksworld-4ops, ferry, gripper, and visitall. Using PDDL domains as benchmarks for evaluating planning performance is a standard practice, as exemplified by the International Planning Competitions (IPCs) [Vallati *et al.*, 2015]. For each domain, we generated three sets of problem instances (train, validation, test) with parameterized generators used in the IPCs. We provided between 456 and 1536 instances for training (the variation is due to the difference in the number of generator parameters in each domain), between 132 and 384 instances for validation and testing (as separate sets), and 100 instances sampled from the test set for planning. The Appendix describes the domains and generator parameters. Notably, the test instances are generated with larger parameters in order to assess the generalization capability. To generate the dataset from these instances, we optimally solved each instance with A* [Hart *et al.*, 1968] and h^{LMcut} in Fast Downward [Helmert, 2006] under 5min runtime / 8GB memory (train,val) and 30min runtime / 8GB memory (test). Whenever it failed to solve a instance within the limits, we retried generation with a different random seed for a maximum of 20 times until success, thus ensuring a specified number of instances were generated. We also discarded trivial instances that satisfied the goal conditions at the initial state. For each state s in the optimal plan, we archived h^* and the values of several heuristics (e.g., h^{LMcut} and h^{FF}). Therefore, each instance was used to obtain several data points.

Model Configurations. We evaluated three different ML methods to show that our statistical model is implementation-agnostic. Neural Logic Machine (NLM) [Dong *et al.*, 2019] is an architecture designed for inductive learning and reasoning over symbolic data which has been successfully applied to classical planning domains for learning heuristic functions [Gehring *et al.*, 2022] with Reinforcement Learning (RL) [Sutton and Barto, 2018]. STRIPS-HGN [Shen *et al.*, 2020, HGN for short] is another architecture based on the notion of *hypergraphs*. Lastly, we used linear regression with the hand-crafted features proposed in [Gomoluch *et al.*, 2017], which comprise the goal-count [Fikes *et al.*, 1972] and FF [Hoffmann and Nebel, 2001] heuristics, along with the total and mean number of effects ignored by FF's relaxed plan.

We analyze our learning & planning system from several orthogonal axes. **Gaussian vs. Truncated:** Using $\mu(s)$ as the parameter of a Gaussian $\mathcal{N}(\mu(s), \sigma(s))$ or Truncated Gaussian $\mathcal{TN}(\mu(s), \sigma(s), l, \infty)$ distribution. **Learned vs. fixed sigma:** Predicting $\sigma(s)$ or using a constant value $\sigma(s) = \frac{1}{\sqrt{2}}$, as it is done for the MSE loss. **Lower bounds:** Computing the lower bound l with the h^{LMcut} heuristic. When we use a Gaussian distribution, l is used to clip the heuristic prediction $\mathbb{E}[\mathrm{x}] = \mu(s)$ to the interval $[l, \infty)$. Ablation studies with $l = h^{\max}(s)$ [Bonet and Geffner, 2001] and $l = h^{\mathrm{blind}}(s)$ are included in the Appendix. **Residual learning:** Either using the model to directly predict $\mu(s)$ or to predict an offset $\Delta\mu(s)$ over a heuristic $h(s)$, so that $\mu(s) = \Delta\mu(s) + h(s)$. We use $h = h^{\mathrm{FF}}$ as our unbiased estimator of h^*, as proposed in [Yoon *et al.*, 2008]. In the Appendix, we conduct experiments with h^{LMcut} as the basis of the residual.

Training. We trained each configuration with 5 different random seeds on a training dataset that consists of 400 problem instances subsampled from the entire training problem set (456-1536 instances, depending on the domain). Due to the nature of the dataset, these 400 problem instances can result in a different number of data points depending on the length of the optimal plan of each instance. We performed 4×10^4 weight updates (training steps) using *AdamW* [Loshchilov and Hutter, 2017] with batch size 256, weight decay 10^{-2} to avoid overfitting, gradient clip 0.1, learning rate of 10^{-2} for the linear regression and NLM, and 10^{-3} for HGN. All models use the NLL loss for training, motivated by the theory, but note that the NLL of $\mathcal{N}(\mu, \sigma = 1/\sqrt{2})$ matches the MSE up to a constant, as previously noted. For each model, we saved the weights that resulted in the best validation MSE metric during the training. On a single NVIDIA Tesla V100, each NLM training took ≈ 0.5 hrs except in visitall (≈ 2 hrs). HGN was much slower (≈ 3 hrs except ≈ 15 hrs in blocksworld). Linear models trained much faster (12-20 minutes).

Evaluation Scheme. We first report two different metrics on the test set: "MSE" and "MSE+clip". Here, MSE is the mean square error between $h^*(s_i)$ and $h(s_i) = \mathbb{E}[\mathrm{x}]$, i.e., $\frac{1}{N}\sum_{i=1}^{N}(h(s_i) - h^*(s_i))^2$, for i-th state s_i of N states in the test dataset. $\mathbb{E}[\mathrm{x}]$ of $\mathcal{TN}$ is given by Eq. 10 while $\mathbb{E}[\mathrm{x}]$ of $\mathcal{N}$ is simply μ. "+clip" variants are exclusive to $\mathcal{N}$ and they clip μ to l, i.e., use $\max(\mu, l)$ in place of μ to compute the MSE. We also obtained the MSE for $h = h^{\mathrm{FF}}$ and $h = h^{\mathrm{LMcut}}$.

We then evaluate the planning performance using the point estimate provided by each model as a heuristic function to guide a search algorithm. Since the learned heuristic is inadmissible, we evaluate our heuristics in an agile search setting, where Greedy Best-First Search [Bonet and Geffner, 1999, GBFS] is the standard algorithm. We do not use A* because it does not guarantee finding the optimal (shortest) plan [Russell and Norvig, 2010] with inadmissible heuristics and it is slower than GBFS in the agile search as it must explore all nodes below the current best $f = g + h$ value, which is unnecessary for finding a satisficing solution. In our experiments, we evaluate search performance as the combination of the number of solved instances and the number of heuristic evaluations required to solve each instance, with a limit of 10000 evaluations per problem. We do not use runtime as our metric so that results are independent of the hardware and software configuration. Additionally, we evaluated GBFS with the off-the-shelf

h^{FF} heuristic as a baseline. The planning component is based on Pyperplan [Alkhazraji *et al.*, 2020].

4.1 Heuristic Accuracy Evaluations

We focus on the results obtained by the NLM models, as our conclusions from the Linear and the HGN models (See Appendix) were not substantially different. Table 1 shows the MSE metric of the heuristics obtained by different configurations evaluated on the test instances (which are significantly larger than the training instances). Compared to the models trained with the NLL loss of $\mathcal{N}$, those trained with our proposed $\mathcal{TN}$ loss often result in significantly more accurate heuristics. For example, in ferry and gripper, some $\mathcal{N}$ models completely fail to learn a useful heuristic, as shown by the large heuristic errors (e.g., the base $\mathcal{N}$/fixed/none model on ferry obtains an MSE of 118.59). In these situations, the clipping trick often reduces errors significantly (e.g., the $\mathcal{N} + clip$/fixed/none model on the same domain obtains an MSE of 10.50). However, this simply indicates that the $\mathcal{N}$ models are falling back to the h^{LMcut} heuristic for those (many) predictions which are smaller than h^{LMcut}. This is why, even with clipping, $\mathcal{N}$ models fail to match the accuracy of $\mathcal{TN}$ models in many cases: For example, the MSE of $\mathcal{N}$+clip/learn/none on gripper is 7.7 points larger than the one of $\mathcal{TN}$/learn/none. This confirms our hypothesis that admissible heuristics such as h^{LMcut} should be used as the lower bound of $\mathcal{TN}$, instead of simply to perform post-hoc clipping of heuristic predictions.

Additional detailed observations follow. **First,** $\mathcal{TN}$ tends to converge faster during training, as shown in Fig. 2. **Second,** residual learning often improves accuracy considerably, thus proving to be an effective way of utilizing inadmissible heuristics. **Third,** we observed that the trained heuristics, including those that use residual learning from h^{FF}, tend to be more accurate than h^{FF}. This rejects the hypothesis that residual learning is simply copying h^{FF} values. **Fourth,** learning σ helps $\mathcal{TN}$ exclusively. For every $\mathcal{N}$ and $\mathcal{TN}$ model, Table 1 contains 2 comparisons related to σ (learn/none vs. fixed/none and learn/h^{FF} vs. fixed/h^{FF}) across 4 domains, resulting in a total of 8 comparisons. Out of 8, learning σ degrades the MSE of $\mathcal{N}$ in 5 cases, while it improves the MSE of $\mathcal{TN}$ in 7 cases. This happens because σ affects the expected value $\mathbb{E}[\mathrm{x}]$ of $\mathcal{TN}$ used as the heuristic prediction but it does not for $\mathcal{N}$. In other words, $\mathcal{TN}$ models requires both μ and σ in order to achieve good heuristic accuracy. This explains why $\mathcal{TN}$/fixed/h^{FF} is not as competitive as $\mathcal{N}$/fixed/h^{FF}: fixed/h^{FF} is an ill-defined configuration for $\mathcal{TN}$.

4.2 Search Performance Evaluations

We compared the search performance of GBFS using heuristic functions obtained by the different models as well as the State-of-the-Art off-the-shelf h^{FF} heuristic. We included our proposed *learn/*h^{FF} configuration and the baseline *fixed/none* configuration. Results for *learn/none* and *fixed/*h^{FF} can be found in the Appendix. Table 2 shows the average$\pm$stdev of the ratio of problem instances solved (i.e., coverage), where a value of 1 means all instances are solved, and the average number of node evaluations per problem over 5 seeds. The second metric is introduced to differentiate between methods that solve most or all of the instances.

We observed that, with our proposed *learn/*h^{FF} configuration, the learned heuristics significantly outperform the off-the-shelf h^{FF} heuristic. Additionally, $\mathcal{TN}$ outperforms $\mathcal{N}$ and $\mathcal{N}$+clip in every domain when both the ratio of solved instances and number of node evaluations are considered (the second metric is used to break ties in the first one).

Conversely, with the traditional but less ideal *fixed/none* configuration, several learned heuristics are surpassed by h^{FF} and, also, $\mathcal{TN}$ is outperformed by $\mathcal{N}$ or $\mathcal{N}$+clip in some cases. These results align with those shown in Table 1. Firstly, $\mathcal{N}$ models which do not use clipping sometimes learn dismal heuristics (e.g., in gripper, $\mathcal{N}$/fixed/none fails to solve any instance). Secondly, $\mathcal{TN}$ models need to predict σ (in addition to μ) in order to learn heuristics of good quality.

5 Related Work

Using admissible heuristics as lower bounds of a $\mathcal{TN}$ distribution may appear trivial in the hindsight. Existing work use $\mathcal{TN}$ for machine learning most often in the context of safety-aware planning, where the upper/lower bounds are *arbitrary* constraints imposed by the environment or by a domain expert. For example, [Murray *et al.*, 2023] uses $\mathcal{TN}$ to model a Simple Temporal Network with Uncertainty (STNU) which can model a distribution of time within a specific start time / deadline. [Eisen *et al.*, 2019] uses $\mathcal{TN}$ to optimize wireless device allocations, where the truncation encodes the range of signal power. In robotics, $\mathcal{TN}$ is often used to limit the measurement uncertainty [Kamran *et al.*, 2021].

In contrast, the admissible heuristics used as lower bounds in our work are *formal bounds automatically proved by* symbolic algorithms. For example, h^{LMcut} is computed by deriving a so-called landmark graph, and then reducing the costs on the edges that constitute a cut of the graph. To our knowledge, *our work is the first to show that such formally derived bounds for combinatorial tasks can be combined with* $\mathcal{TN}$.

For instance, in applications of machine learning to Operations Research problems (e.g., Vehicle Routing Problem, TSP), existing work often tries to learn to solve them without the help of heuristics [Nazari *et al.*, 2018]. Although [Xin *et al.*, 2021] uses the optimal solution obtained by a traditional optimal method (e.g. Concorde solver) for training and combines it with existing admissible heuristics (LKH heuristic) during testing, it does not use the heuristic for training.

In the context of heuristic learning in automated planning, off-the-shelf heuristics have only been used as a training target [Shen *et al.*, 2020], or as a residual basis [Yoon *et al.*, 2008].

In RL [Sutton and Barto, 2018], it is a common practice to accelerate the training through reward shaping, which is theoretically equivalent to residual learning [Ng *et al.*, 1999]. An extension of reward shaping [Cheng *et al.*, 2021] utilizes hand-crafted heuristics. [Gehring *et al.*, 2022] used h^{FF} to shape rewards for classical planning. However, to our knowledge, none has leveraged admissible heuristics as lower/upper bounds. [Cheng *et al.*, 2021] also discussed the *pessimistic* and *admissible* heuristics as desirable properties of RL and planning heuristics, but their method does not explicitly use

				learn/h^{FF}		learn/none		fixed/h^{FF}		fixed/none	
domain	metric	h^{FF}	h^{LMcut}	$\mathcal{N}$	$\mathcal{TN}$	$\mathcal{N}$	$\mathcal{TN}$	$\mathcal{N}$	$\mathcal{TN}$	$\mathcal{N}$	$\mathcal{TN}$
blocks	MSE	22.8	25.06	.76±.1	**.65±.1**	3.26±.6	**2.71±.4**	.83±.1	**.66±.1**	2.97±.9	**2.44±.3**
	+clip			.76±.2		2.91±.4		.83±.2		2.74±.6	
ferry	MSE	9.77	11.10	3.73±.7	**3.45±.8**	141.05±29.4	**8.63±2.7**	**2.98±1.4**	3.85±.9	118.59±10.4	**9.58±1.5**
	+clip			3.72±.6		10.44±28.4		**2.98±1.1**		10.50±9.6	
gripper	MSE	9.93	15.82	3.65±.9	3.70±.9	68.12±16.0	**5.65±1.3**	3.69±.9	3.72±.9	68.22±16.1	**11.97±2.2**
	+clip			3.65±.7		13.37±15.2		3.69±.8		13.38±14.5	
visitall	MSE	13.9	36.4	7.67±.4	**5.30±.6**	25.31±7.9	**9.70±1.6**	6.49±.6	6.62±.9	21.71±2.6	**14.11±1.0**
	+clip			7.60±.4		18.79±7.3		**6.35±.6**		16.38±2.3	

Table 1: Test metrics for NLM (smaller the better). Each number represents the mean±std of 5 random seeds. For each configuration, we performed 10^4 training steps, saving the checkpoints with the best validation MSE metric. We tested several orthogonal configurations: 1) Learning σ (*learn*) or fixing it to $\frac{1}{\sqrt{2}}$ (*fixed*) and 2) Using residual learning (h^{FF}) or not (*none*). For each configuration, we compare the test MSE metric of the Gaussian ($\mathcal{N}$) and Truncated Gaussian ($\mathcal{TN}$) models. Rows labeled as *+clip* denote a $\mathcal{N}$ model where μ is clipped above h^{LMcut}. For each configuration, the best average MSE among $\mathcal{N}$, $\mathcal{N}$+clip, and $\mathcal{TN}$ is highlighted in **bold**, if the value gap to the second-best is larger than 0.1. Results for linear regression and STRIPS-HGN models are provided in the Appendix.

		learn/h^{FF} (proposed)			fixed/none (baseline)		
domain	h^{FF}	$\mathcal{N}$	$\mathcal{N}$+clip	$\mathcal{TN}$	$\mathcal{N}$	$\mathcal{N}$+clip	$\mathcal{TN}$
Ratio of solved instances under 10^4 evaluations (higher the better)							
blocks	.13	.84±.19	.85±.19	**.88±.14**	**.79±.29**	.50±.35	.55±.33
ferry	.82	.91±.19	.91±.19	**.98±.05**	.01±.01	.57±.10	**.58±.13**
gripper	.96	**1**	**1**	**1**	0	.92±.12	**1**
visitall	.86	.97±.07	**.98±.06**	**.98±.05**	.82±.33	**1**	**1**
Average node evaluations (smaller the better)							
blocks	9309	2690±2128	2681±2121	**2060±1607**	**4118±2663**	6268±2675	5903±2685
ferry	5152	3216±1964	3117±1967	**2477±1093**	9933±92	6675±582	**6475±725**
gripper	3918	1642±139	1643±141	**1637±492**	10000±0	2941±1513	**1709±658**
visitall	3321	2156±1451	2148±1511	**1683±1290**	3384±3448	**591±216**	612±363

Table 2: Planning results on NLM weights saved according to the best validation MSE metric, comparing the average±stdev of the ratio of solved instances under 10^4 node evaluations and the average number of evaluated nodes across problems. The number of evaluated nodes is counted as 10^4 on instances the planner failed to solve. For each configuration (learn/h^{FF} or fixed/none), we highlight the best results in bold.

the upper/lower bound property for training.

6 Conclusion and Future Work

In this paper, we studied the problem of supervised heuristic learning under a statistical lens, focusing on how to effectively utilize the information provided by admissible heuristics. Firstly, we provided some statistical background on heuristic learning which was later leveraged to analyze the decisions made (sometimes unknowingly) in the literature. We explained how the commonly used MSE loss implicitly models the heuristic to be learned as a Gaussian distribution. Then, we argued that this heuristic should instead be modeled as a Truncated Gaussian, where admissible heuristics are used as the lower bound of the distribution. We conducted extensive experimentation, comparing the heuristics learned with our truncated-based statistical model versus those learned by minimizing squared errors. Results show that our proposed method improves convergence speed during training and yields more accurate heuristics that result in better planning performance, thus confirming that it is the correct approach for utilizing admissible bounds in heuristic learning.

Our findings serve to answer the three important questions we raised in the introduction: **What should the model learn?** To achieve super-symbolic benefits, we should use expensive metrics such as h^*, not poly-time heuristics or sub-optimal plan costs. **How should we train the model?** We maximize the likelihood of the observed h^* assuming a Truncated Gaussian distribution lower bounded by an admissible heuristic. **Why so?** The *principle of maximum entropy*: the Truncated Gaussian distribution encodes our prior knowledge without any extra assumptions that may cause harmful bias.

In future work, we will extend our proposed method to other learning settings. One interesting scenario is given by iterative search algorithms [Richter *et al.*, 2010; Richter *et al.*, 2011], where the cost of the best solution found so far could be used as the upper bound of a Truncated Gaussian. Another avenue for future work is to explore the RL setting where a value function is learned instead of a heuristic, extending the work on residual learning for RL [Gehring *et al.*, 2022].

Acknowledgements

This work has been partially funded by the Grant PID2022-142976OB-I00, funded by MICIU/AEI/ 10.13039/501100011033 and by "ERDF/EU", as well as the Andalusian Regional predoctoral grant no. 21-111-PREDOC-0039 and by "ESF Investing in your future".

References

[Alkhazraji *et al.*, 2020] Yusra Alkhazraji, Matthias Frorath, Markus Grützner, Malte Helmert, Thomas Liebetraut, Robert Mattmüller, Manuela Ortlieb, Jendrik Seipp, Tobias Springenberg, Philip Stahl, and Jan Wülfing. Pyperplan, 2020.

[Arfaee *et al.*, 2011] Shahab Jabbari Arfaee, Sandra Zilles, and Robert C Holte. Learning heuristic functions for large state spaces. *Artificial Intelligence*, 175(16-17):2075–2098, 2011.

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279, 2013.

[Betz and Helmert, 2009] Christoph Betz and Malte Helmert. Planning with h^+ in theory and practice. In *Advances in Artificial Intelligence*, volume 5803, pages 9–16. Springer, 2009.

[Bonet and Geffner, 1999] Blai Bonet and Hector Geffner. Planning as heuristic search: New results. In *Proc. of European Conference on Planning*, pages 360–372. Springer, 1999.

[Bonet and Geffner, 2001] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.

[Cheng *et al.*, 2021] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learning. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 34:13550–13563, 2021.

[Dong *et al.*, 2019] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural Logic Machines. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

[Dowson and Wragg, 1973] D Dowson and A Wragg. Maximum-entropy distributions having prescribed first and second moments (corresp.). *IEEE Transactions on Information Theory*, 19(5):689–693, 1973.

[Eisen *et al.*, 2019] Mark Eisen, Clark Zhang, Luiz FO Chamon, Daniel D Lee, and Alejandro Ribeiro. Learning Optimal Resource Allocations in Wireless Systems. *IEEE Transactions on Signal Processing*, 67(10):2775–2790, 2019.

[Ernandes and Gori, 2004] Marco Ernandes and Marco Gori. Likely-admissible and sub-symbolic heuristics. In *Proc. of European Conference on Artificial Intelligence*, volume 16, pages 613–617, 2004.

[Ferber *et al.*, 2022] Patrick Ferber, Florian Geißer, Felipe Trevizan, Malte Helmert, and Jörg Hoffmann. Neural network heuristic functions for classical planning: Bootstrapping and comparison to other methods. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 32, pages 583–587, 2022.

[Fikes *et al.*, 1972] Richard E Fikes, Peter E Hart, and Nils J Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Gehring *et al.*, 2022] Clement Gehring, Masataro Asai, Rohan Chitnis, Tom Silver, Leslie Kaelbling, Shirin Sohrabi, and Michael Katz. Reinforcement learning for classical planning: Viewing heuristics as dense reward generators. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 32, pages 588–596, 2022.

[Gomoluch *et al.*, 2017] Pawel Gomoluch, Dalal Alrajeh, Alessandra Russo, and Antonio Bucchiarone. Towards learning domain-independent planning heuristics. *arXiv preprint arXiv:1707.06895*, 2017.

[Hart *et al.*, 1968] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Helmert and Domshlak, 2009] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: what's the difference anyway? In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 19, pages 162–169, 2009.

[Helmert, 2006] Malte Helmert. The fast downward planning system. *J. Artif. Intell. Res.(JAIR)*, 26:191–246, 2006.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.(JAIR)*, 14:253–302, 2001.

[Jaynes, 1957] Edwin T Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106(4):620–630, 1957.

[Kamran *et al.*, 2021] Danial Kamran, Tizian Engelgeh, Marvin Busch, Johannes Fischer, and Christoph Stiller. Minimizing safety interference for safe and comfortable automated driving with distributional reinforcement learning. In *Proc. of IEEE International Workshop on Intelligent Robots and Systems (IROS)*, pages 1236–1243. IEEE, 2021.

[Loshchilov and Hutter, 2017] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:2010.13187*, 2017.

[Marom and Rosman, 2020] Ofir Marom and Benjamin Rosman. Utilising uncertainty for efficient learning of likely-admissible heuristics. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 30, pages 560–568, 2020.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis.

Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Murray *et al.*, 2023] Andrew Murray, Ashwin Arulselvan, Michael Cashmore, Marc Roper, and Jeremy Frank. A column generation approach to correlated simple temporal networks. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 33, pages 295–303, 2023.

[Nazari *et al.*, 2018] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 31:9861–9871, 2018.

[Ng *et al.*, 1999] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. of the International Conference on Machine Learning (ICML)*, volume 99, pages 278–287, 1999.

[Richter *et al.*, 2010] Silvia Richter, Jordan Thayer, and Wheeler Ruml. The joy of forgetting: Faster anytime search via restarting. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 20, pages 137–144, 2010.

[Richter *et al.*, 2011] Silvia Richter, Matthias Westphal, and Malte Helmert. Lama 2008 and 2011. In *International Planning Competition*, pages 117–124, 2011.

[Russell and Norvig, 2010] Stuart J Russell and Peter Norvig. *Artificial intelligence a modern approach*. London, 2010.

[Shen *et al.*, 2020] William Shen, Felipe Trevizan, and Sylvie Thiébaux. Learning domain-independent planning heuristics with hypergraph networks. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 30, pages 574–584, 2020.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Takahashi *et al.*, 2019] Takeshi Takahashi, He Sun, Dong Tian, and Yebin Wang. Learning heuristic functions for mobile robot path planning using deep neural networks. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 29, pages 764–772, 2019.

[Vallati *et al.*, 2015] Mauro Vallati, Lukás Chrpa, Marek Grzes, Thomas Leo McCluskey, Mark Roberts, and Scott Sanner. The 2014 international planning competition: Progress and trends. *AI Magazine*, 36(3):90–98, 2015.

[Xin *et al.*, 2021] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. NeuroLKH: Combining Deep Learning Model with Lin-Kernighan-Helsgaun Heuristic for Solving the Traveling Salesman Problem. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 34:7472–7483, 2021.

[Yoon *et al.*, 2008] Sungwook Yoon, Alan Fern, and Robert Givan. Learning control knowledge for forward search planning. *J. Mach. Learn. Res.(JMLR)*, 9(4):683–718, 2008.