Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

Potential Heuristics as Real-Valued Multilinear Polynomials

Augusto B. Corrêa[1], **Simon Dold**[2], **Malte Helmert**[2]

[1]University of Oxford, United Kingdom
[2]University of Basel, Switzerland
augusto.blaascorrea@chch.ox.ac.uk, {simon.dold,malte.helmert}@unibas.ch

Abstract

Potential functions are a compact and easy way to describe heuristics in classical planning. They are declarative, simple to understand, and clear to explain. Further, they are used as a tool to study the computational complexity of planning problems, and have connections to other concepts such as novelty measures and generalized planning. In this work, we cast potential functions as real-valued multilinear polynomials over the Booleans. While this might seem as a simple syntactic game at first glance, this automatically brings insights to the theory of potential heuristics. For example, we show that the multilinear polynomial representation can be computed using Fourier expansions. This immediately gives us a unique representation for each heuristic function, which allows us to study concepts such as correlation complexity more directly. Moreover, this also connects potential heuristics to a whole new field of computer science – analysis of Boolean functions – leading to consequences ranging from learning theory to performance prediction.

Introduction

Potential heuristics (Pommerening et al. 2015) are not only useful for estimating goal-distances in classical planning. They can also be used to study the state-space topology of planning tasks (e.g., Seipp et al. 2016; Corrêa and Pommerening 2019; Helmert et al. 2022; Dold and Helmert 2024a). Potential heuristics are real-valued functions that associate *weights* to *conjunctions of literals*. For example, in a planning task with propositional variables x and y, we could have the potential heuristic

$$h_1(s) = 3 - 2[x] - 1[y] \tag{1}$$

where $[e]$ is 1 if e is true in s and 0 otherwise (Knuth 1992). In a state $s = \{x \mapsto 1, y \mapsto 0\}$, we have $h(s) = 1$; in a state $s' = \{x \mapsto 0, y \mapsto 1\}$, we have $h(s') = 2$.

The descriptive nature of potential heuristics allows us to see *which interactions are relevant* for the task. For example, if a function h^{pot} synthesizing h^* for a task Π contains the feature $[x \wedge \overline{y}]$, we know that the interaction between variables x and y is relevant to encode h^*. By analyzing their representation, we can see how different variables interact (Pommerening, Helmert, and Bonet 2017; Steinmetz and Hoffmann 2018; Fišer and Steinmetz 2024), or how hard it is to synthesize specific heuristics with specific properties (Seipp et al. 2016; Corrêa and Pommerening 2019; Francès et al. 2019). For instance, Seipp et al. (2016) use the *correlation complexity* metric to estimate how complex a potential heuristic must be to guide a hill-climbing search directly to a goal state. This allows us to reason about factored state spaces using a format that is easy to describe and understand. Additionally, potential heuristics can be used to establish bounds of other metrics, such as novelty width (Dold and Helmert 2024b).

But the *representation of a heuristic function as a potential heuristic is not unique*. For instance, h_1 in (1) describes the same function as

$$h_2(s) = 2[\overline{x}] + 1[\overline{y}]$$

and also as

$$h_3(s) = 3[\overline{x} \wedge \overline{y}] + 2[\overline{x} \wedge y] + 1[x \wedge \overline{y}],$$

despite all three having very distinct representations. It is not immediate to recognize that h_1, h_2, and h_3 are all equivalent, and when we want to study larger tasks, comparing two potential representations to check their equivalence is even harder. Without having a unique, canonical representation for a function – such as h^* or h^+ – it becomes challenging to estimate their analytical properties (Seipp et al. 2016; Corrêa and Pommerening 2019).

It turns out that this issue can be easily addressed. This is a problem of the *representation* we use for potential heuristics, and it can be sorted out once we use a new representation: multilinear polynomials.

In this paper, we study how to cast potential heuristics as multilinear polynomials over the Booleans. The main advantage of our new representation is that every heuristic function has a *unique* representation as a multilinear polynomial. This unique representation, called the *Fourier expansion*, gives us several tools to analyze hardness of planning tasks and heuristic functions. For example, one can estimate the mean heuristic values of a state space just from the Fourier expansion of the function – which has been shown empirically to help estimate performance (Korf 1997; Korf, Reid, and Edelkamp 2001; Holte 2013); or estimate how many random samples we need to approximate a specific heuristic function only by looking at its Fourier coefficients.

Our main contribution is to *connect potential heuristics with a well-known area of computer science*. We can benefit from studying potential heuristics under the lens of analysis of Boolean functions as a multilinear polynomial. This change in representation allows us to extend the theory of potential heuristics, while still preserving important existing properties. In this paper, we present this connection and highlight the main results that can be translated into potential heuristics.

Planning Tasks

We consider propositional planning tasks, and we represent planning tasks as factored state spaces. A *factored state space* is a transition system $\mathcal{T} = \langle \mathcal{P}, \mathcal{S}, T, \text{cost}, s_I, S_* \rangle$, where $\mathcal{P}$ is a finite set of *propositional variables*, $\mathcal{S} = 2^{|\mathcal{P}|}$ is a finite set of *states*, $T \subseteq \mathcal{S} \times \mathcal{S}$ is a *transition relation*, $\text{cost} : T \to \mathbb{R}_{\geq 0}$ is a *cost function*, $s_I \in \mathcal{S}$ is the *initial state*, and $S_* \subseteq \mathcal{S}$ is a set of *goal states*.

A state $s \in \mathcal{S}$ is a complete Boolean assignment to the variables in $\mathcal{P}$. We represent the Boolean domain as $\{0, 1\}$.

An *s-plan* for $\mathcal{T}$ is a sequence of states $\pi = \langle s_0, \ldots, s_n \rangle$ where $s_0 = s$, $s_n \in S_*$, and $(s_{i-1}, s_i) \in T$ for all $1 \leq i \leq n$. If an s_I-plan for $\mathcal{T}$ exists, then the task represented by $\mathcal{T}$ is *solvable*. The cost of an s-plan is the sum of the costs of all transitions in the plan. A *heuristic* h maps a state $s \in \mathcal{S}$ to a number $h(s) \in \mathbb{R} \cup \{\infty\}$, indicating the estimated cost of an 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 conjunction of literals with pairwise distinct variables. We say a feature f is true in a state s if $s \models f$. We also use the Iverson brackets (Knuth 1992) $[f]$, which have value 1 if $s \models f$ and 0 otherwise. The *size* of the feature f is the number of literals in f.

Example 1. *Consider the planning task where a binary counter starting at* 00 *needs to be incremented to* 11*. We use this task as a running example throughout the paper. We can represent a state of this task with a variable x for the most significant bit and a variable y for the least significant one. There exists a total of four states $\mathcal{S} = \{\{x \mapsto 0, y \mapsto 0\}, \{x \mapsto 0, y \mapsto 1\}, \{x \mapsto 1, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 1\}\}$. We also define the initial state $s_I = \{x \mapsto 0, y \mapsto 0\}$ and the single goal state $s_* \in S_*$ as $s_* = \{x \mapsto 1, y \mapsto 1\}$. The transition relation T contains transitions (s, s') where state s' represents a simple binary increment to s.*

The function $h^*(s)$, in this case, defines the number of increments needed from s to state 11*. It is explicitly defined as $h^*(\{x \mapsto 0, y \mapsto 0\}) = 3$, $h^*(\{x \mapsto 0, y \mapsto 1\}) = 2$, $h^*(\{x \mapsto 1, y \mapsto 0\}) = 1$, and $h^*(\{x \mapsto 1, y \mapsto 1\}) = 0$.*

Potential Heuristics A *weight function* $w : \mathcal{F} \to \mathbb{R}$ associates a set $\mathcal{F}$ of features to real-valued weights. It represents a *potential heuristic* mapping a state s to the sum of weights for features of s:

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

Note that two distinct weight functions can represent the same heuristic. For now, we consider only potential heuristics that represent *real-valued Boolean functions*.[1]

The *dimension* of a potential heuristic is the size of its largest feature in its representation with non-zero weight.

Any heuristic function can be represented as a potential heuristic.[2] This requires knowing the heuristic in every state explicitly.

Example 2. *The following potential heuristics represent the perfect heuristic h^* of our running example:*

$$h_1(s) = 3 - 2[x] - 1[y],$$
$$h_2(s) = 2[\overline{x}] + 1[\overline{y}],$$
$$h_3(s) = 3[\overline{x} \wedge \overline{y}] + 2[\overline{x} \wedge y] + 1[x \wedge \overline{y}].$$

For a planning task with $|\mathcal{P}|$ state variables, an explicit representation (e.g., as a table) has size $O(2^{|\mathcal{P}|})$. However, even when the heuristic function is given explicitly as part of the input, the problem of synthesizing it as a potential function is challenging (Corrêa and Pommerening 2019).

We therefore refer to potential functions representing heuristics simply as the *potential representations* of heuristics. This will help us in the upcoming section, when we introduce new heuristic representations.

Heuristics as Multilinear Polynomials

A *multilinear polynomial* $f : \{0, 1\}^n \to \mathbb{R}$ is a *real-valued Boolean function* that maps a vector of n Boolean values to a real value. (*Multilinear* means that terms are multivariate and no variable x occurs in a power of 2 or higher.) We sometimes refer to f simply as a *polynomial*, as all polynomials studied in our paper are multilinear. Our notation and definitions are based on the work by O'Donnell (2014).

Multilinear polynomials are flexible with respect to the values for true or false. Each choice of values has benefits and drawbacks compared to the others. We usually use the classic 0 for false, and 1 for true. Other choices include $\{-1, +1\}$, $\{-1/2, +1/2\}$, and the field $\mathbb{F}_2$. It is possible to convert between all these different representations by changing the basis of the function, although the specific results and methods discussed next must also be adapted (O'Donnell 2014). We assume 1/0 for true/false, and when a different choice of true/false values is needed, we note it explicitly.

Let f be a real-valued Boolean function over variables $\{x_1, \ldots, x_n\}$. Let $S \subseteq \{x_1, \ldots, x_n\}$ be an arbitrary subset

[1] There are different ways to adapt potential heuristics to infinite values (e.g., to represent unsolvable states). For example, one can represent infinity as a number larger than the trivial upper bound for plan length or one can use nested potential heuristics – one to discriminate solvable states, and one to estimate the distance. We refer to the work by Corrêa and Pommerening (2019) and Helmert et al. (2022) for more information.

[2] Note that we do not refer to "procedural" functions – e.g., the algorithm to compute h^{add} (Bonet and Geffner 2001) over an arbitrary task – but to heuristics according to our definition, where a heuristic maps a specific set of states $\mathcal{S}$ to distance estimates.

of variables. We can obtain a multilinear polynomial representation of f, denoted here as F_f, via *interpolation*:

$$F_f(x_1, \ldots, x_n) = \sum_{S \subseteq \{x_1, \ldots, x_n\}} f(S) \prod_{x_i \in S} x_i \prod_{x_j \notin S} (1 - x_j),$$

where $f(S)$ is the valuation of f when $x_i = 1$ if $x_i \in S$, and $x_i = 0$ otherwise.

Example 3. *Using the interpolation method, we compute a multilinear polynomial for $f_{\max_2}$, the function computing the maximum between x and y, as follows:*

$$\begin{aligned} F_{f_{\max_2}}(x, y) &= 0 \cdot (1 - x)(1 - y) + 1 \cdot (1 - x)(y) \\ &\quad + 1 \cdot (x)(1 - y) + 1 \cdot (x)(y) \\ &= y - xy + x - xy + xy \\ &= x + y - xy. \end{aligned}$$

Given $S \subseteq \{x_1, \ldots, x_n\}$, we define the *monomial* corresponding to S as

$$x^S = \prod_{x_i \in S} x_i,$$

where $x^\emptyset = 1$ by convention. We define the *size of a monomial* x^S as $|S|$, the number of variables multiplied in x^S. It is easy to see that this also corresponds to the conjunction over the variables in S.

Every function $f : \{0, 1\}^n \to \mathbb{R}$ over $\{x_1, \ldots, x_n\}$ can be *uniquely expressed as a multilinear polynomial*

$$F_f(x_1, \ldots, x_n) = \sum_{S \subseteq \{x_1, \ldots, x_n\}} \hat{f}(S)\, x^S,$$

where $\hat{f}(S) \in \mathbb{R}$ is the *Fourier coefficient* of f on S, corresponding to the coefficient of the monomial x^S in the polynomial. The polynomial F_f is called the *Fourier expansion of* f and $\max_{S \subseteq \{x_1, \ldots, x_n\}, \hat{f}(S) \neq 0} |S|$ is its *degree*.

Since we focus on heuristics that are real-valued Boolean functions, we can use their Fourier expansion to obtain a unique representation for them. We use the Fourier coefficient on S as the weight for the conjunction over the variables in S (and 0 for all other features). To illustrate, we show how to obtain the Fourier expansion of the h^* function from Example 1 using the interpolation method.

Example 4. *Let F_{h^*} be the Fourier expansion of the h^* function from Example 1. We define it as follows:*

$$\begin{aligned} h^*(s) &= 3 \cdot (1 - x)(1 - y) + 2 \cdot (1 - x)(y) \\ &\quad + 1 \cdot (x)(1 - y) + 0 \cdot (x)(y) \\ &= 3 \cdot (1 - y - x + xy) + 2 \cdot (y - xy) + (x - xy) \\ &= 3 - 2x - y = F_{h^*}(x, y). \end{aligned}$$

This is the unique multilinear polynomial representing h^* from Example 1.

One can observe that F_{h^*} above is equivalent to the potential representation h_1 from Example 2, but it is not identical to h_2. This leads us to a new question: given a potential representation of a heuristic function h, can we compute the Fourier expansion of h? The answer is yes. This can be done by replacing, for any variable $x_i \in \{x_1, \ldots, x_n\}$, the features $[x_i]$ with the variable x_i, $[\overline{x_i}]$ with the complement $(1 - x_i)$, and replacing the conjunction of features with the product of the corresponding variables and complements.

Example 5. *Starting with h_1 or h_2 from our previous examples, we can compute F_{h^*} using the replacement rules just described above. From $h_1(s) = 3 - 2[x] - 1[y]$, we obtain F_{h^*} immediately:*

$$F_{h^*}(x, y) = 3 - 2x - y.$$

From $h_2(s) = 2[\overline{x}] + 1[\overline{y}]$, we require some arithmetic:

$$\begin{aligned} F_{h^*}(x, y) &= 2 \cdot (1 - x) + 1 \cdot (1 - y) \\ &= 2 - 2x + 1 - y \\ &= 3 - 2x - y. \end{aligned}$$

The case for h_3 is left as an exercise.

We can also do the *inverse* replacement: given a Fourier expansion, obtain a potential representation by replacing variables x with $[x]$, and a product of variables $x_1 x_2 \ldots x_n$ with $[x_1 \wedge \cdots \wedge x_n]$.

In our examples, we compute the Fourier expansion using interpolation. We can, however, use a slightly better method, based on *Fast Fourier Transform* (FFT; Cooley and Tukey 1965). The interpolation can be computed in $O(2^{2n})$ while the FFT method has $O(n2^n)$ runtime (O'Donnell 2014). We do not detail how the FFT works here, as it would involve adding another layer of definitions (e.g., complex numbers).

The main definitions of potential representation translate to the Fourier expansions. The dimension of a potential representation h is analogous to the degree of the Fourier expansion F_h. Moreover, the degree of F_h is bounded by the dimension of h: as the dimension of h is the size of its largest feature, this corresponds to the largest monomial when we apply the replacement rules described above. Similarly, the weight function used in h is equivalent to the Fourier coefficients $\hat{h}$. The definitions of correlation complexity and optimal correlation complexity can also be translated, but now referring to the degree of the Fourier expansion instead of the dimension of a potential representation. This shifts the view from syntactic properties to semantic properties.

This leads us to interesting results:

Proposition 1. *A potential representation h of dimension d can be converted into a Fourier expansion F_h of degree d or less.*

Proposition 2. *A Fourier expansion F_h of degree d can be converted into a potential representation of dimension d.*

Note that the dimension of h is a property of the potential representation h but *the degree of a Fourier expansion is a property of the function*. In other words, different potential representations $h_1, \ldots, h_n$ can synthesize the same heuristic h despite having different dimensions, but the degree of the Fourier expansion F_h is uniquely defined by h.

Given our replacement rules, these two propositions are not surprising. However, this brings a more subtle result about potential representations:

Proposition 3. *Given a potential representation h of dimension dim(h), there exists a potential representation h' with dimension dim$(h') \leq$ dim(h) where all features (with non-zero weights) use only positive literals.*

This follows directly from the replacement rules presented above. Given a potential representation h, we can "eliminate" features with negative literals as follows: convert to a Fourier expansion using the corresponding replacement rules, then convert back to a new potential representation using the replacement rule in the reverse direction. This gives a new potential representation without negative literals – since whenever we replace variables from the Fourier expansion, we only replace them with features containing positive literals – while preserving the dimension of the original function.

Due to symmetry, Proposition 3 also holds for using only negative literals (as in h_2), or any other parity choice per variable in which it is allowed to appear in the features. However, the number of non-zero weights could drastically change. In an extreme case, the function is constant 0 except for one input. If this input matches the parity choice, then a single feature is sufficient, otherwise $2^{|\mathcal{P}|}$ features are required. The size of the largest feature does not change. Any parity choice for the features results in the same dimension.

It is often desirable to express the planning problem with a finite domain representation, such as SAS$^+$ (Bäckström and Nebel 1995), instead of propositional variables. Here, a state is a full assignment for each variable v to their domains and a feature of size d is a partial assignment for d variables. Each domain can be represented as $\mathrm{dom}(v) = \{1, 2, \ldots, |\mathrm{dom}(v)|\}$. We say a feature f is true in a state s if f is a restriction of the full assignment s, for example $[x = 1, y = 3]$ has value 1 iff $s(x) = 1$ and $s(y) = 3$. For this view we get an analogous result to Proposition 3.

Proposition 4. *Given a potential representation h of dimension dim(h), there exists a potential representation h' with dimension dim$(h') \leq$ dim(h) and where no features (with non-zero weights) map any variable to domain value* 1.

We create such a representation by rewriting features in Iverson brackets to a product of size 1 features. For example, $[x = 1, y = 3]$ becomes $[x = 1] \cdot [y = 3]$. Then, we replace $[x = 1]$ with $(1 - [x = 2] - \cdots - [x = |\mathrm{dom}(x)|])$ to remove all features that map to 1. After this representation shift, we use arithmetic to get a sum of products of size 1 features which we rewrite to larger features. None of these products has more factors than the ones we started with. Thus, the dimension of the created representation is not larger.

Final Thoughts: Why Is This Relevant?

Fourier expansions allow us to have a unique representation for any heuristic function. But is this of any use? We survey some interesting results, properties, and improvements that come naturally once we think of potential functions as multilinear polynomials.

There are two different scenarios where the Fourier expansion can be useful. First, when we have the Fourier expansion at hand, we can *analytically* infer different properties of the heuristic. Second, when we have access only to some states and their heuristic values (e.g., in a learning setting, where we can obtain only some states), we can estimate with high confidence different properties of the Fourier expansion with a low number of queries.

The Basics. The simplest information we can obtain directly from a Fourier expansion heuristic F_h is its *mean value*, denoted by $\mathbb{E}[F_h]$. This corresponds to the average heuristic values over all states. For the next two propositions, we consider Fourier expansions $F_h : \{-1, +1\}^n \to \mathbb{R}$ where -1 and $+1$ represent false and true. As mentioned before, we can convert the traditional $\{0, 1\}$ representation to this new basis without problems (O'Donnell 2014).

Proposition 5. *Let F_h be a Fourier expansion heuristic, then $\mathbb{E}[F_h] = \hat{h}(\emptyset)$. In other words, the mean of F_h is the constant coefficient of the function.*

We can also infer the variance of the heuristic value distribution directly from the Fourier expansion:

Proposition 6. *Let F_h be a Fourier expansion heuristic. The* variance *of F_h is*

$$\textbf{\textit{Var}}[F_h] = \sum_{S \neq \emptyset} \hat{h}(S)^2.$$

Knowing the mean and the variance of a heuristic function can give us insight into how algorithms such as A* and IDA* should perform in this task. Holte et al. (2006) empirically show that if two heuristics have the same average value, then the one with heuristic values more concentrated around the mean is expected to perform better. Moreover, the distribution of heuristic values can be used to analyze the expected number of expansions in different search algorithms (Korf 1997; Korf, Reid, and Edelkamp 2001; Holte 2013).

With the Fourier expansion heuristics, we can do these inferences analytically, without having to know the entire state space. This becomes useful when we generate different heuristic functions (either potential heuristics or Fourier expansion heuristics) and want to estimate which one is best.

Learning Heuristics. A Fourier coefficient can be interpreted as a measure of how relevant the feature/term is. If this relevance is concentrated on a small collection $\mathcal{C}$ of terms, we can efficiently learn the function.

Let $\mathcal{C}$ be a collection of subsets of $\{x_1, \ldots, x_n\}$. We say a real-valued Boolean function f is ϵ-*concentrated* on $\mathcal{C}$ if

$$\sum_{S \notin \mathcal{C}} \hat{f}(S)^2 \leq \epsilon.$$

In words: the Fourier coefficients for terms *not in* $\mathcal{C}$ do not contribute much to the function. For a heuristic h that is ϵ-concentrated on $\mathcal{C}$, we can learn an approximation of h to an accuracy $O(\epsilon)$ with probability at least $1 - \delta$ in time $\mathrm{poly}(|\mathcal{C}|, 1/\epsilon)\mathrm{poly}(n)\log(1/\delta)$ with the use of *membership queries* (Kushilevitz and Mansour 1991). Additionally, if $\mathcal{C}$ only contains subsets with at most d variables (i.e., the most significant weights are in features of degree at most d) then *random queries* suffice (Linial, Mansour, and Nisan 1993).

Note that this result is consistent with previous work that showed that h^* can be approximated well using only a few features with significant weights (Corrêa and Pommerening 2019).

Acknowledgments

This research was supported by the Swiss National Science Foundation (SNSF) as part of the project "Unifying the Theory and Algorithms of Factored StateSpace Search" (UTA).

References

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS^+ Planning. *Computational Intelligence*, 11(4): 625–655.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1): 5–33.

Cooley, J. W.; and Tukey, J. W. 1965. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90): 297–301.

Corrêa, A. B.; and Pommerening, F. 2019. An Empirical Study of Perfect Potential Heuristics. In Lipovetzky, N.; Onaindia, E.; and Smith, D. E., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 114–118. AAAI Press.

Dold, S.; and Helmert, M. 2024a. Higher-Dimensional Potential Heuristics: Lower Bound Criterion and Connection to Correlation Complexity. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 151–161. AAAI Press.

Dold, S.; and Helmert, M. 2024b. Novelty vs. Potential Heuristics: A Comparison of Hardness Measures for Satisficing Planning. In Dy, J.; and Natarajan, S., eds., *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2024)*, 20692–20699. AAAI Press.

Fišer, D.; and Steinmetz, M. 2024. Towards Feasible Higher-Dimensional Potential Heuristics. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 210–220. AAAI Press.

Francès, G.; Corrêa, A. B.; Geissmann, C.; and Pommerening, F. 2019. Generalized Potential Heuristics for Classical Planning. In Kraus, S., ed., *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 5554–5561. IJCAI.

Helmert, M.; Sievers, S.; Rovner, A.; and Corrêa, A. B. 2022. On the Complexity of Heuristic Synthesis for Satisficing Classical Planning: Potential Heuristics and Beyond. In Thiébaux, S.; and Yeoh, W., eds., *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 124–133. AAAI Press.

Holte, R. C. 2013. Korf's Conjecture and the Future of Abstraction-Based Heuristics. In Frisch, A. M.; and Gregory, P., eds., *Proceedings of the Tenth Symposium on Abstraction, Reformulation, and Approximation (SARA 2013)*, 128–131. AAAI Press.

Holte, R. C.; Felner, A.; Newton, J.; Meshulam, R.; and Furcy, D. 2006. Maximizing over Multiple Pattern Databases Speeds up Heuristic Search. *Artificial Intelligence*, 170(16–17): 1123–1136.

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

Korf, R. E. 1997. Finding Optimal Solutions to Rubik's Cube Using Pattern Databases. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI 1997)*, 700–705. AAAI Press.

Korf, R. E.; Reid, M.; and Edelkamp, S. 2001. Time complexity of iterative-deepening A^*. *Artificial Intelligence*, 129: 199–218.

Kushilevitz, E.; and Mansour, Y. 1991. Learning decision trees using the Fourier spectrum. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing (STOC 1991)*, 455–464. ACM Press.

Linial, N.; Mansour, Y.; and Nisan, N. 1993. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40(3): 607–620.

O'Donnell, R. 2014. *Analysis of Boolean Functions*. Cambridge University Press.

Pommerening, F.; Helmert, M.; and Bonet, B. 2017. Higher-Dimensional Potential Heuristics for Optimal Classical Planning. In Singh, S.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3636–3643. AAAI Press.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In Bonet, B.; and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3335–3341. AAAI Press.

Seipp, J.; Pommerening, F.; Röger, G.; and Helmert, M. 2016. Correlation Complexity of Classical Planning Domains. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3242–3250. AAAI Press.

Steinmetz, M.; and Hoffmann, J. 2018. LP Heuristics over Conjunctions: Compilation, Convergence, Nogood Learning. In Lang, J., ed., *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 4837–4843. IJCAI.