Counting Plans with Heuristic State-Space Search

David Speck

University of Basel, Switzerland
davidjakob.speck@unibas.ch

Abstract

We present A$^\#$, a heuristic search algorithm for counting optimal and cost-bounded plans in classical planning. We prove that A$^\#$ terminates and correctly counts optimal plans, and describe how it extends to the cost-bounded case. A$^\#$ compactly represents multiple action sequences reaching the same state using counters while exploiting heuristic estimates. An ablation study evaluates the impact of the modifications to A* used to obtain A$^\#$. Our experiments show that A$^\#$ outperforms existing approaches, solving significantly more tasks overall and scaling to substantially higher plan counts (up to 10^{92}).

Introduction

After decades of progress in finding single plans, researchers have only recently intensified efforts to reason about multiple plans for given planning tasks. Such reasoning involves generating alternative plans (e.g., Boddy et al. 2005; Nguyen et al. 2012; Sohrabi et al. 2016; Katz et al. 2018; Speck, Mattmüller, and Nebel 2020) or explaining an agent's behavior by answering questions about its plans (e.g., Krarup et al. 2019; Eifler et al. 2020). However, existing solutions to these problems are often specific to a particular domain, which limits their broader utility, or they are based on naively enumerating solutions, which generally does not scale. An exception is recent work that reasons about plans by compiling planning tasks into SAT or ASP and representing the solution space using knowledge representation tools (Speck et al. 2025; Gnad et al. 2025). An important problem falling under the umbrella of reasoning over the plan space is counting the number of (optimal or cost-bounded) plans. This can help answer quantitative questions for important applications, such as measuring the robustness of network systems by determining the number of distinct exploit sequences (Boddy et al. 2005; Shaw and Meel 2024) (e.g., Fig. 1).

We introduce a heuristic search algorithm called A$^\#$, which uses the idea of dynamic programming (DP) to count the number of plans. It builds on the well-known approach of counting paths in a graph using Dijkstra's algorithm with DP and generalizes it by leveraging heuristic estimates. Furthermore, unlike SAT-based approaches for counting plans (e.g., Speck et al. 2025), it does not rely on a fixed cost bound (Kautz and Selman 1996; Rintanen 2012). And unlike top-k planners (e.g., Katz et al. 2018), it avoids explicit plan enumeration, scaling to higher plan counts.

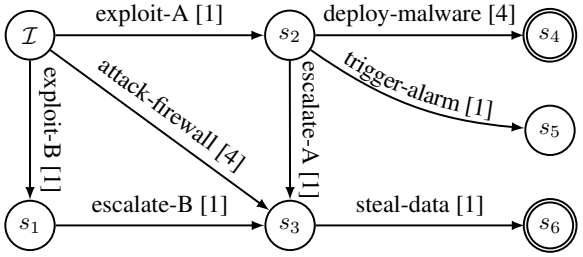

Figure 1: Transition system induced by the example cybersecurity task Π_{Exp}. Transitions are labeled with the action and its cost. Goal states are marked with double circles.

Preliminaries

We consider classical *planning tasks* $\Pi = \langle \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$ in the form of SAS$^+$ (Bäckström and Nebel 1995).$\mathcal{V}$ is a finite set of *variables* $v \in \mathcal{V}$, each associated with a finite *domain* $\mathit{dom}(v)$. An *atom* $\langle v, d \rangle$ is a variable–value pair where $v \in \mathcal{V}$ and $d \in \mathit{dom}(v)$. A consistent set of atoms s is called a *partial state*. If s contains an atom for each variable in $\mathcal{V}$, then s is called a *state*. We denote by S the set of all states. $\mathcal{A}$ is a finite set of *actions*, where each action $a = \langle \mathit{pre}(a), \mathit{eff}(a), \mathit{cost}(a) \rangle \in \mathcal{A}$ consists of partial states $\mathit{pre}(a)$ and $\mathit{eff}(a)$, representing the precondition and effect of a, and a positive *cost* $\mathit{cost}(a) \in \mathbb{N}^+$. $\mathcal{I}$ is the *initial state*. $\mathcal{G}$ is the partial state representing the *goal*. The set of *goal states* is defined as $S^* = \{s \in S \mid \mathcal{G} \subseteq s\}$.

An action $a \in \mathcal{A}$ is *applicable* in a state $s \in S$ if $\mathit{pre}(a) \subseteq s$. The result of applying action a to state s is the *successor state* $s' = s[\![a]\!]$, where for each variable $v \in \mathcal{V}$: $s'(v) = d$ if $\langle v, d \rangle \in \mathit{eff}(a)$, and $s'(v) = s(v)$ otherwise. For $A \subseteq \mathcal{A}$, we define $\mathit{successors}(s, A) = \{\langle s[\![a]\!], a \rangle \mid a \in A, \mathit{pre}(a) \subseteq s\}$ as the set of *successor-action pairs*.

A *plan* $\pi = \langle a_1, \ldots, a_n \rangle$ for planning task Π is a sequence of applicable actions that induces a sequence of states $s_0, \ldots, s_n$, where $s_0 = \mathcal{I}$, $\mathcal{G} \subseteq s_n$, and $s_i = s_{i-1}[\![a_i]\!]$ for all $i = 1, \ldots, n$. The cost of a plan $\pi = \langle a_1, \ldots, a_n \rangle$ is defined as the sum of the costs of its actions, i.e., $\sum_{i=1}^{n} \mathit{cost}(a_i)$. With $\mathcal{P}$, we refer to the (possibly infinite) *set of all plans* for a planning task Π. A plan $\pi \in \mathcal{P}$ is *optimal* if $\mathit{cost}(\pi) \leq \mathit{cost}(\pi')$ for all $\pi' \in \mathcal{P}$. We refer to the *set of optimal plans* as $\mathcal{P}^* \subseteq \mathcal{P}$, and we denote a *cost-*

bounded set of plans as $\mathcal{P}_{<c} = \{\pi \in \mathcal{P} \mid cost(\pi) < c\}$.

We are interested in counting the number of optimal or cost-bounded plans for a given planning task. More precisely, given a planning task Π, we seek to determine $|\mathcal{P}^*|$ or $|\mathcal{P}_{<c}|$ for a cost bound $c \in \mathbb{N}^+$. Since we consider strictly positive action costs, both $\mathcal{P}^*$ and $\mathcal{P}_{<c}$ are finite.

Example 1. Fig. 1 depicts the transition system for a planning task Π_{Exp}, which models network system vulnerabilities within a cybersecurity application. There are two optimal plans with cost 3: $\pi_1 = \langle$exploit-A, escalate-A, steal-data$\rangle$ and $\pi_2 = \langle$exploit-B, escalate-B, steal-data$\rangle$. Thus, the optimal plan count is $|\mathcal{P}^*| = 2$. For the cost-bounded problem, $|\mathcal{P}_{<c}| = 0$ for $c < 4$, while $|\mathcal{P}_{<4}| = 2$. If $c > 5$, the count increases to $|\mathcal{P}_{<c}| = 4$ due to two suboptimal plans with cost 5.

A *heuristic (function)* $h : S \to \mathbb{N} \cup \{\infty\}$ estimates the cost of reaching a goal state from a given state $s \in S$ (Pearl 1984). A heuristic h is *consistent* if it is goal-aware, i.e., $h(s) = 0$ for all $s \in S^*$, and satisfies $h(s) \leq h(s') + cost(a)$ for all $\langle s', a\rangle \in successors(s, \mathcal{A})$. Every consistent heuristic is *admissible*, meaning it never overestimates the cost of a cheapest action sequence from a state to any goal state.

It is well-known that *A* search* (Hart, Nilsson, and Raphael 1968) with a consistent heuristic finds optimal solutions without the need for reopening (i.e., expanding the same state more than once). To store information about states, A* uses nodes. A *node* n consists of a state s, an action a, the *g-value* (the cost of the action sequence leading to the node), and a *parent* node from which it was created via a. A* maintains an *open* and a *closed* list to track which states are promising for expansion and which have already been expanded. The algorithm iteratively selects from the open list a node n with state s that is not in the closed list, such that n has the minimal *f-value* in open, which is the sum of the *g-value* and the *h-value*, defined as $f(n) = g(n) + h(s)$. We sometimes write $g(s)$ and $f(s)$ instead of $g(n)$ and $f(n)$ when the node is clear from the context.

Counting Plans with A#

We present A# (Alg. 1), a novel heuristic search algorithm for counting optimal plans using a consistent heuristic. A# is a modification of A* (without reopening) with three key differences: 1. We order the open list by f-values and require tie-breaking in favor of smaller g-values, as opposed to the more common strategy of favoring smaller h-values. 2. We do not terminate upon expanding the first goal state, but instead continue until all goal states with the optimal cost have been expanded. 3. We track the number of action sequences to each state at its best-known cost, accumulating these counts when a sequence of equal cost is discovered or resetting them if a cheaper sequence is found.

Alg. 1 begins by initializing necessary data structures (lines 1–6): the cost bound *bound*, the plan counter *nplans*, the open and closed lists, and two maps, *best-g* and *nreach*. These maps store, for each encountered state, the best known cost (g-value) and the number of action sequences leading to that state with the best known cost. Then (lines 7–9), we set the reachability count and cost for the initial state $\mathcal{I}$ to 1 and 0, respectively, and add a search node for $\mathcal{I}$ to the open list using the *make-node* function (line 28).[1]

As in A*, each while loop iteration (line 10) considers the most promising state. However, in A#, we break ties between states with the same f-value in favor of those with smaller g-values (line 4) and terminate only when the f-value exceeds the cost bound. We then check if the state has already been expanded (lines 11–15). If so, we skip it; otherwise, we add it to the closed list. This is followed by a goal check (lines 16–18). If a goal state is expanded, we add the number of ways it can be reached to the plan count and set the cost bound accordingly to prune more expensive states.

In lines 19–26, the current state s is expanded, and its successors are generated. Importantly, when a cheaper action sequence to a successor *succ* is found, we reset its reach-

Algorithm 1: A#: Counting optimal solutions

```
   Input : Planning task Π = ⟨V, A, I, G⟩
           Consistent heuristic h
   Output: Number of optimal plans
1  bound ← ∞
2  nplans ← 0
3  closed ← HashSet
4  open ← MinHeap ordered by ⟨f, g⟩
5  nreach ← HashMap : S → ℕ (default 0)
6  best-g ← HashMap : S → ℕ ∪ {∞} (default ∞)
7  nreach[I] ← 1
8  best-g[I] ← 0
9  open.insert(make-node(I, 0, ∅, ∅))
10 while open.min-f() < bound do
11     n ← open.pop-min()
12     s ← n.state
13     if s ∈ closed then
14         continue
15     closed.insert(s)
16     if s ∈ S* then
17         nplans ← nplans + nreach[s]
18         bound ← n.g + 1
19     foreach ⟨succ, a⟩ ∈ successors(s, A) do
20         succ-cost ← n.g + cost(a)
21         if succ-cost < best-g[succ] then
22             best-g[succ] ← succ-cost
23             nreach[succ] ← nreach[s]
24             open.insert(make-node(succ, succ-cost, n, a))
25         else if succ-cost = best-g[succ] then
26             nreach[succ] ← nreach[succ] + nreach[s]
27 return nplans
28 function make-node(s, g, p, a)
29     return Node(state=s, g=g, parent=p, action=a)
```

[1] Storing the parent and the action leading to a state is not strictly necessary for counting plans. However, we view A# as a generalization of A*, and their inclusion allows to not only return the plan count but also to reconstruct a plan for each expanded goal state.

ability count and set $nreach[succ]$ to the number of ways $succ$ can be reached via s (lines 21–24). Otherwise, if $succ$ is reached via s at the same best cost previously known, we add the new reachability count to $nreach[succ]$ (lines 25–26). The case where $succ$ is reached with a higher cost is irrelevant, as we only seek optimal plans, i.e., suboptimal action sequences can be safely ignored.

Finally, once all reachable states with f-values less than or equal to the optimal plan cost are expanded, we have accumulated all action sequences to reach any goal state optimally and return the total number of optimal plans (line 27).

We now present an example, followed by a proof that A$^\#$ terminates and correctly counts the optimal plans.

Example 2. Consider A$^\#$ applied to the task Π_{Exp} (Fig. 1). Expanding $\mathcal{I}$ ($nreach = 1$) generates states s_1, s_2 ($g = 1$) and s_3 ($g = 4$), all with $nreach = 1$ (lines 19–24). The consistent heuristic and the tie-breaking ensures A$^\#$ prioritizes the states s_1 and s_2. Expanding s_1 updates s_3 to a lower cost of 2 and sets $nreach[s_3] = 1$ (lines 21–24). When s_2 is subsequently expanded, s_3 is reached again at the same optimal cost (2), incrementing $nreach[s_3]$ to 2 (lines 25–26). Additionally, s_4 and s_5 are added to the open list with costs 5 and 2. Since s_5 is a dead-end, a strong heuristic avoids ever expanding it. Note that even if an expensive path to a goal existed from s_5, A$^\#$ would still prune it if its f-value exceeded the optimal cost. Next, expanding s_3 generates the goal state s_6 with cost 3 and $nreach[s_6] = 2$. When expanding s_6, A$^\#$ updates the optimal plan count to 2, sets the cost bound to 4 (lines 16–17), and terminates because the open list contains no further nodes under the cost bound (line 10). Importantly, A$^\#$ has effectively pruned suboptimal action sequences (e.g., those involving the high-cost action "attack-firewall").

Theorem 1. *A$^\#$ (Alg. 1) with a consistent heuristic terminates and returns the number of optimal plans for a planning task with action costs in $\mathbb{N}^+$.*

Proof. Termination is guaranteed because the state space is finite, and each state is expanded at most once.

A$^\#$ is a variant of A*. With a consistent heuristic, A$^\#$ expands nodes in non-decreasing order of their f-value. Moreover, when a state s is expanded, the corresponding node has optimal cost $g(n) = g^*(s)$. In A$^\#$, we thus expand all goal states that are reachable with optimal cost, since the entire last f-layer is expanded (lines 10 & 18) and $h(s) = 0$ for goal states. A$^\#$ returns the sum of $nreach[s]$ over all such goal states with optimal cost. Thus, it suffices to show that $nreach[s]$ correctly counts the number of optimal action sequences reaching s when s is expanded. We prove this by induction over $g^*(s)$ of a state s. Base case ($g^*(s) = 0$): This holds only for the initial state, which can be reached by exactly one action sequence (line 7). Inductive step: Consider a state s with $g^*(s) > 0$. Every optimal action sequence reaching s ends in a transition from a state t to s with $g^*(t) < g^*(s)$. By the hypothesis, when t is expanded, $nreach[t]$ correctly counts all optimal action sequences reaching t. We show that all such predecessors t are expanded before s. By consistency, $f(t) = g^*(t) + h(t) \leq g^*(s) + h(s) = f(s)$. If $f(t) < f(s)$, then t is expanded before s. Otherwise $f(t) = f(s)$, and since action costs are strictly positive, $g^*(t) < g^*(s)$. With tie-breaking in favor of smaller g-values, t is again expanded before s. Consequently, all optimal contributions to $nreach[s]$ are accumulated before s is expanded (lines 21–26), and thus $nreach[s]$ equals the total number of optimal action sequences reaching s. □

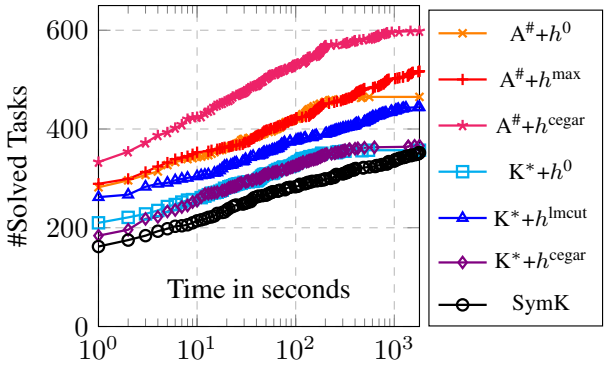

Figure 2: Coverage over time for counting optimal plans.

Cost-Bounded Plans

A$^\#$ can be modified to count not only optimal plans but all cost-bounded plans below a given bound. For this, we remove the initialization and update of the dynamic cost bound (lines 1 & 18) and augment the state representation with its g-value. This turns the search graph into a directed acyclic graph, and the same original state is considered different if it is reachable with different costs. Thus, the if-case of lines 21–24 will never trigger a reset of the reachability count, since the same original state with different g-values is considered distinct. Note that this can result in expanding the same state multiple times with different g-values. We refer to this version for counting cost-bounded plans as A$^\#_{\text{all}}$.

Experiments

We implemented A$^\#$ (and A$^\#_{\text{all}}$) in SymK (Speck, Geißer, and Mattmüller 2020), which is based on Fast Downward (Helmert 2006), and use the GMP library (Granlund 2023) for representing plan counts as arbitrary-precision integers. For A$^\#$, we considered three consistent heuristics: the constant zero heuristic h^0,[2] the delete-relaxation heuristic $h^{\max}$ (Bonet and Geffner 1999), and the CEGAR heuristic h^{cegar} (Seipp and Helmert 2018). We evaluated on 1467 instances from 50 STRIPS domains with positive action costs (IPCs 1998–2023), using a 30 min time limit and a 6 GiB memory limit. Code and data are available online (Speck 2026).

Counting Optimal Plans

We compare A$^\#$ against two state-of-the-art top-k planners capable of counting plans by enumeration: K* (Aljazzar and Leue 2011; Katz et al. 2018) using h^{blind}, h^{cegar}, and its default (inconsistent) heuristic h^{lmcut} (Helmert and Domshlak 2009); SymK with symbolic bidirectional search (Torralba et al. 2017; Speck, Seipp, and Torralba 2025).

[2] A$^\#$ with h^0 mimics Dijkstra's algorithm (1959) extended with dynamic programming for path/plan counting.

Config.	Coverage per Bound 1.0×	1.25×	1.5×	Log_{10}(#Plans) at 1.5× Median	Max
SymK	291	142	88	3.0	7
K^*+h^0	298	184	110	4.0	7
K^*+h^{cegar}	302	186	112	4.0	7
K^*+h^{lmcut}	359	200	113	4	7
Planalyst	344	234	175	7	19
$A^\#_{\text{all}}+h^0$	369	323	297	10	55
$A^\#_{\text{all}}+h^{\max}$	389	343	309	10	92
$A^\#_{\text{all}}+h^{\text{cegar}}$	**495**	**372**	**317**	10	55

Table 1: Solved tasks for cost-bounded plan counting where the bound is a multiplicative factor of known upper bounds (Left). Median and maximum plan counts at factor 1.5, shown as powers of ten (Right).

Fig. 2 shows the number of solved tasks over time for counting optimal plans. Among the solved tasks, 94% have more than one optimal plan, with up to 10^{71} distinct optimal plans. All $A^\#$ configurations solve more tasks than any other approach from start to end. $A^\#$ with a strong heuristic like h^{cegar} solves 133 more tasks than a Dijkstra search for plan counting (h^0). Furthermore, it solves 154 more instances than the strongest non-$A^\#$ approach. This shows that heuristic search for counting scales significantly better than blind search or enumeration-based methods.

$A^\#$ with h^{cegar} solves more tasks than K^* with h^{cegar} in 33 domains, whereas K^* is not superior in any. This gap narrows when comparing $A^\#$ with h^{cegar} against K^* with h^{lmcut}: $A^\#$ is superior in 28 and K^* in 11 domains. Thus, lifting the consistency requirement of $A^\#$ is a promising avenue for future work that may widen the performance gap even further.

Counting Cost-Bounded Plans

We compare $A^\#_{\text{all}}$ for counting cost-bounded plans with previous approaches, including Planalyst (Speck et al. 2025). Since Planalyst requires a predefined cost bound, it was only included in this experiment. The approach transforms the plan space into a SAT formula (Rintanen 2012) and represents it as a d-DNNF (Darwiche and Marquis 2002). As Planalyst only supports unit costs, we restrict the benchmarks to 1152 instances across 37 domains with unit costs.

Tab. 1 shows the coverage numbers and statistics for all approaches counting plans within a multiplicative factor $c \in \{1.0, 1.25, 1.5\}$ of known upper cost bounds (Muise 2016; Speck et al. 2025) for each instance. $A^\#_{\text{all}}$ achieves the highest coverage across the different bounds. In particular, $A^\#_{\text{all}}$ scales significantly better to larger cost bounds than the top-k approaches K^* and SymK, as it does not require the explicit enumeration of plans. It also outperforms Planalyst in this regard. This superior scalability is evidenced by the median and maximum number of plans counted: $A^\#_{\text{all}}$ can count up to 10^{92} plans, a task that is entirely hopeless for enumeration-based methods. Interestingly, $A^\#_{\text{all}}$ using the $h^{\max}$ heuristic achieved the highest plan counts. Its fast computation appears to pay off over the more informative h^{cegar} heuristic in vast state and plan spaces.

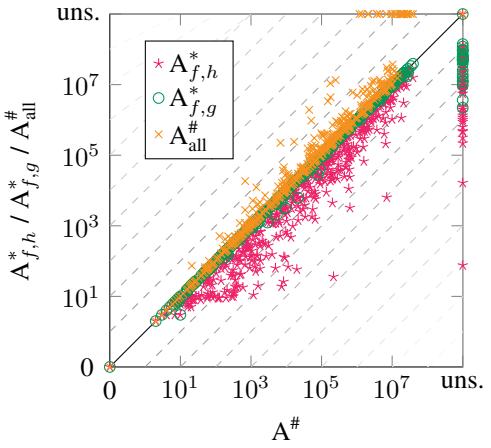

Figure 3: Node expansions of $A^\#$ using h^{cegar} compared to A^* (open list tie-breaking by smaller h or g values) and $A^\#_{\text{all}}$.

Ablation Study

A question that arises is how the changes to A^* to derive $A^\#$ or $A^\#_{\text{all}}$ impact performance. Fig. 3 shows the node expansions of $A^*_{f,h}$, $A^*_{f,g}$, and $A^\#_{\text{all}}$ in comparison to $A^\#$ using h^{cegar}. By $A^*_{f,h}$, we refer to tie-breaking by smaller h among nodes with equal f-values, and by $A^*_{f,g}$ to tie-breaking by smaller g (as done in $A^\#$). The change in tie-breaking has a greater effect, which is in line with the common approach of preferring $A^*_{f,h}$ over $A^*_{f,g}$ as it is more aggressive towards the goal. By construction, the expansions of $A^\#$ and $A^*_{f,g}$ are identical prior to the last f-layer. The difference results only from the need of $A^\#$ to expand all optimal goal nodes. Finally, in $A^\#_{\text{all}}$ for cost-bounded plan counting, the same state may be expanded multiple times with different costs. We can see that this again has a higher impact on performance.

Conclusions and Future Work

We introduced $A^\#$, a heuristic search algorithm for counting optimal and cost-bounded plans. The key idea of $A^\#$ is to collapse multiple action sequences to the same state by aggregating them into a counter while exploiting heuristic estimates. Empirically, $A^\#$ shows superior performance and scalability for plan counting compared to existing approaches based on plan enumeration (Katz et al. 2018; Speck, Mattmüller, and Nebel 2020) or compilations to SAT (Palacios et al. 2005; Speck et al. 2025).

A limitation of $A^\#$ is the reliance on consistent heuristics. Lifting this requirement may further improve performance. Furthermore, most approaches to counting and enumerating plans treat all actions as equally important. This may affect practical applications when modeling complex structures using auxiliary actions. Thus, we plan to investigate options for ignoring (i.e., projecting away) actions for plan counting, similar to projected model counting (e.g., Chakraborty, Meel, and Vardi 2013; Fichte, Hecher, and Hamiti 2021).

Acknowledgements

This work was funded by the Swiss National Science Foundation (SNSF) as part of the project "Unifying the Theory and Algorithms of Factored State-Space Search" (UTA).

References

Aljazzar, H.; and Leue, S. 2011. K*: A Heuristic Search Algorithm for Finding the K Shortest Paths. *AIJ*, 175(18): 2129–2154.

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

Boddy, M.; Gohde, J.; Haigh, T.; and Harp, S. 2005. Course of Action Generation for Cyber Security Using Classical Planning. In *Proc. ICAPS 2005*, 12–21.

Bonet, B.; and Geffner, H. 1999. Planning as Heuristic Search: New Results. In *Proc. ECP 1999*, 360–372.

Chakraborty, S.; Meel, K. S.; and Vardi, M. Y. 2013. A Scalable Approximate Model Counter. In *Proc. CP 2013*, 200–216.

Darwiche, A.; and Marquis, P. 2002. A Knowledge Compilation Map. *JAIR*, 17: 229–264.

Dijkstra, E. W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1: 269–271.

Eifler, R.; Cashmore, M.; Hoffmann, J.; Magazzeni, D.; and Steinmetz, M. 2020. A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning. In *Proc. AAAI 2020*, 9818–9826.

Fichte, J. K.; Hecher, M.; and Hamiti, F. 2021. The Model Counting Competition 2020. *ACM Journal of Experimental Algorithmics*, 26(13): 1–26.

Gnad, D.; Hecher, M.; Gaggl, S.; Rusovac, D.; Speck, D.; and Fichte, J. K. 2025. Interactive Exploration of Plan Spaces. In *Proc. KR 2025*, 599–609.

Granlund, T. 2023. *GNU MP: The GNU Multiple Precision Arithmetic Library 6.3.0*. Free Software Foundation.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Helmert, M. 2006. The Fast Downward Planning System. *JAIR*, 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proc. ICAPS 2009*, 162–169.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018. A Novel Iterative Approach to Top-k Planning. In *Proc. ICAPS 2018*, 132–140.

Kautz, H.; and Selman, B. 1996. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proc. AAAI 1996*, 1194–1201.

Krarup, B.; Cashmore, M.; Magazzeni, D.; and Miller, T. 2019. Model-Based Contrastive Explanations for Explainable Planning. In *ICAPS Workshop on Explainable AI Planning (XAIP)*.

Muise, C. 2016. Planning.Domains. In *ICAPS 2016 System Demonstrations and Exhibits*.

Nguyen, T. A.; Do, M. B.; Gerevini, A.; Serina, I.; Srivastava, B.; and Kambhampati, S. 2012. Generating diverse plans to handle unknown and partially known user preferences. *AIJ*, 190: 1–31.

Palacios, H.; Bonet, B.; Darwiche, A.; and Geffner, H. 2005. Pruning Conformant Plans by Counting Models on Compiled d-DNNF Representations. In *Proc. ICAPS 2005*, 141–150.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Rintanen, J. 2012. Planning as Satisfiability: Heuristics. *AIJ*, 193: 45–86.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *JAIR*, 62: 535–577.

Shaw, A.; and Meel, K. S. 2024. Model Counting in the Wild. In *Proc. KR 2024*, 755–764.

Sohrabi, S.; Riabov, A. V.; Udrea, O.; and Hassanzadeh, O. 2016. Finding Diverse High-Quality Plans for Hypothesis Generation. In *Proc. ECAI 2016*, 1581–1582.

Speck, D. 2026. Code, benchmarks and data for the paper "Counting Plans with Heuristic State-Space Search". https://doi.org/10.5281/zenodo.20596396.

Speck, D.; Geißer, F.; and Mattmüller, R. 2020. When Perfect Is Not Good Enough: On the Search Behaviour of Symbolic Heuristic Search. In *Proc. ICAPS 2020*, 263–271.

Speck, D.; Hecher, M.; Gnad, D.; Fichte, J. K.; and Corrêa, A. B. 2025. Counting and Reasoning with Plans. In *Proc. AAAI 2025*, 26688–26696.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic Top-k Planning. In *Proc. AAAI 2020*, 9967–9974.

Speck, D.; Seipp, J.; and Torralba, Á. 2025. Symbolic Search for Cost-Optimal Planning with Expressive Model Extensions. *JAIR*, 82: 1349–1405.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient Symbolic Search for Cost-optimal Planning. *AIJ*, 242: 52–79.