Make Planning Research Rigorous Again!

Michael Katz
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
michael.katz1@ibm.com

Harsha Kokel
IBM Almaden Research Center
San Jose, CA 95120
harsha.kokel@ibm.com

Christian Muise
Queen's University
Kingston, Canada
christian.muise@queensu.ca

Shirin Sohrabi
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
ssohrab@us.ibm.com

Sarath Sreedharan
Colorado State University
Fort Collins, CO 80523
sarath.sreedharan@colostate.edu

Abstract

In over sixty years since its inception, the field of planning has made significant contributions to both the theory and practice of building planning software that can solve a never-before-seen planning problem. This was done through established practices of rigorous design and evaluation of planning systems. It is our position that this rigor should be applied to the current trend of work on planning with large language models. One way to do so is by correctly incorporating the insights, tools, and data from the automated planning community into the design and evaluation of LLM-based planners. The experience and expertise of the planning community are not just important from a historical perspective; the lessons learned could play a crucial role in accelerating the development of LLM-based planners. This position is particularly important in light of the abundance of recent works that replicate and propagate the same pitfalls that the planning community has encountered and learned from. We believe that avoiding such known pitfalls will contribute greatly to the progress in building LLM-based planners and to planning in general.

1 Introduction

With the increased interest in language models came the increased interest in the problems where language models do not perform well. One such problem is planning, which could be informally described as sequential decision-making in the presence of information about the model, i.e., information about the task and the environment. While more efforts are being made to tackle the problems using tools built around large language models (LLMs), it is worth keeping in mind that planning is one of the oldest established subfields of artificial intelligence. The field has taken its roots in the 60's of the last century, with the development of best-first search algorithms [43] as part of the famous Shakey project, as well as of a formal language to represent planning problems [30]. Since then, the planning community has been developing both the representation languages and the generic solvers that handle problems represented in these languages. Instrumental in the development of these tools was the International Planning Competition (IPC), a series of roughly bi-annual events that put the state of the art to the test. The IPC had an immense effect on the development of planning systems, but also on the languages used to describe planning tasks, evaluation metrics used, the language features supported, and the development of other tools, such as parsers, grounders, and validators.

Since its inception, the field has made significant contributions to both the theory and practice of building domain-independent planners, i.e., planning software that can solve a never-before-seen planning problem. The main reason it could be achieved is the *rigor with which the research was conducted* on the topic, with methodologies developed over the years through trial and error. This brings us to our primary position: **"Make Planning Research Rigorous Again!"** We believe that this rigor is the cornerstone of thoughtful and reproducible research that can be built upon. Therefore, our belief is that **insights, methodologies, tools, and data from the automated planning community should be correctly incorporated into the design and evaluation of LLM-based planners**.

We strongly believe that the planning community's contributions are significant in ways that extend well beyond their historical context. The lessons learned by the community over the decades can help accelerate the development of LLM-based planners. This position is particularly important in light of the abundance of recent works that replicate and perpetuate errors previously made and subsequently addressed by the planning community. Helping the greater AI community to avoid known pitfalls will contribute greatly to the progress in building LLM-based planners and to planning in general.

A position paper, by its very nature, cannot meaningfully summarize all the lessons that have been identified by a field that has existed for more than half a century. Instead, our goal with this paper is to lay out some pointers and to establish the basic vocabulary that would allow the curious reader to identify the correct resources they can utilize for their own research. We also lay out some general guidelines for evaluating planners and list some widely used tools from the community that might be helpful for researchers who develop LLM-based planners. Further, we believe that this paper can serve as a guide for reviewers tasked with assessing such works.

2 Fundamental Planning Terminology

In this section, we provide a brief overview of different flavors, formalism, and representations of planning problems commonly used by the AI Planning community.

We start from the most basic formalism, commonly referred to as classical planning. A classical planning problem includes the initial state of the world, desired goals, and a set of possible actions. The objective of a planner is to synthesize a plan that is guaranteed to generate a state which contains the desired goals. More formally, the classical planning model is defined as a tuple $\mathcal{S} = \langle S, s_0, S_G, A, f, c \rangle$, where

- S is a finite and discrete state space
- $s_0 \in S$ is a known initial state
- $S_G \subseteq S$ of is a set of goal states
- A is a set of actions; $A(s) \subseteq A$ applicable in each $s \in S$
- f is a deterministic transition function, where $s' = f(a, s)$ for $a \in A(s)$
- c is a non-negative action costs, $c(a, s)$

The solution to a classical planning problem is a sequence of applicable actions that maps s_0 into S_G.

In the classical setting, the initial state is known, actions are deterministic, the system is fully observable, actions are instantaneous, goals are specified as final states, and the planning horizon is finite. Relaxing these assumptions introduces multiple variants of planning, including but not limited to:

- Conformant planning [102, 10] - where initial state is uncertain.
- Probabilistic planning [116, 23, 72, 94] - where the action dynamics is probabilistic and state spaces do not have to be discrete or finite.
- Non-deterministic planning [68], and fully observable non-deterministic planning (FOND) [80, 85, 86] - where the action dynamic is non-deterministic.
- Temporal Planning [32, 37, 75] -where actions have durations and temporal constraints.

- Numeric Planning [48, 90, 104] - where actions can affect numeric state variables.
- Hierarchical Task Network (HTN) planning [29, 87] - where the initial state and the goal are defined as a task network (a set of tasks and constraints), with recursive decomposition of high-level tasks into lower-level sub-tasks.
- Planning with soft or hard constraints (preferences [8, 103], temporally extended goals [4, 7]) - where the goal is to additionally optimize for the soft and hard constraints, as well as partial satisfaction planning, such as net-benefit [110, 66, 1], where the goal is to optimize for difference between the utility and the cost of achieving the goal, and the and oversubscription planning [101, 24, 63]) - where the goal is to find a plan that achieves the best possible utility given resource constraints.

Note that several of these flavors of planning can sometimes be compiled into classical planning for easier computation. Some examples include compiling conformant planning into classical planning [89], compiling a fragment of HTN planning into classical planning [2], and compiling away final state soft constraints/preferences [66].

While most of these models can be captured by the well-known (PO)MDP formalism, it is more efficient to capture these planning models in a compact way (e.g., with a factored representation), with a formal planning language. Multiple such formal languages for planning problems have been introduced. Starting with ground representations, the most popular are the simplest propositional language STRIPS [30], f-STRIPS [35], all the way to multi-valued variables based languages SAS/SAS+/FDR [5, 6, 52].

This variety, however, is a mixed blessing, as various existing solvers that were being developed supported one language or the other, making it more difficult to compare their performance. Consequently, with the birth of International Planning Competitions in 1998, a unified language for planning problems representation was born, named Planning Domain Definition Language (PDDL) [82]. This language has become the de facto standard and most commonly used representation language for planning. The planning knowledge is separated into two parts, the PDDL domain and the PDDL problem. PDDL domain contains knowledge of the constants, types, predicates, actions, their preconditions, and effects. PDDL problem consists of knowledge of the objects, the initial state, and the goal state. There are multiple extensions of PDDL, such as PDDL 2 [32], with the focus on durative actions, numeric fluents, and derived predicates, PDDL 3.0 [36], with the focus on capturing constraints. Linear Temporal Logic (LTL) and its variants are used to represent preferences, or constraints [100, 7, 19], and are partly captured in PDDL 3.0. Furthermore, RDDL [94] and PPDDL have been introduced to represent probabilistic planning tasks [117].

Representing a planning task compactly and accurately is challenging and often constitutes a bottleneck of using a planning system. As a result, learning planning models has become a growing area of interest over the past decade [113, 56, 54, 55]. Learning planning representation, particularly the PDDL representation, has gained special attention in the era of LLM. The existing work can be partitioned into learning the domain model [41, 88, 38, 107] and learning the problem instance representation, assuming the domain model exists [73, 119].

References to Some Tutorials: Readers can use the following tutorials as starting points to read more about many of the topics discussed above: on classical planning, on the Python library Unified Planning that supports modeling and programmatically invoking planners, on RDDL, on methods to generate multiple plans, and on learning planning models. For additional material, we refer the curious reader to the textbooks [39, 40, 81, 45].

Common Pitfall 1. While planning problems can have infinite state spaces, they are well-defined mathematically. Problems with fuzzy and ill-defined state and action spaces are generally not considered by the planning community. Instead, one may identify a well-defined variant of the problem, allowing also to define what constitutes a solution.

Common Pitfall 2. Just because a problem can be represented as a planning problem, it doesn't mean that planning tools are a good fit for it. For example, many reasoning problems, like the canonical NP-complete problem of boolean satisfiability (SAT) [16] and puzzles like Sudoku [99, 77], can be represented as planning problems. However, SAT and CSP solvers would be better suited to serve as baselines for these problems. A good indication for a planning problem is its sequential nature, if the order in which decisions are made matters.

3 Computational Problems, Algorithms, and Complexity

Even for the most restricted fragment of classical planning, there is a variety of decision and search problems one can think of. Two decision problems are the most popular, *plan existence* and *bounded plan existence*, a variant that asks whether the plan exists of quality under a given bound. There is a larger variety of search problems, including *cost-optimal planning*, asking to find a plan that minimizes summed action cost, or, in the case of unit-cost actions, minimizes plan length. Other search problems include *satisficing planning*, asking to find any plan, while cheaper plans are better; *agile planning*, where the only thing that matters is how quickly the plan is found; *top-k planning*, asking to find k plans such that no cheaper plans exist; *top-quality planning*, asking to find all plans up to a certain cost; and *diverse planning*, aiming at obtaining diverse set of plans, considering plan quality as well. A planner that is guaranteed to only return a valid plan is a *sound* planner, and a planner that is guaranteed to return a solution if one exists is a *complete* planner.

While all the problems described previously are PSPACE-hard [12] in general, when represented in a planning language such as PDDL, for some domains, the complexity of some of these search problems can significantly vary from the others. One prominent example is the BlocksWorld domain, where cost-optimal planning is NP-complete, while agile or satisficing planning are in P. To see that, one can think of a simple two-stage policy that can solve any BlocksWorld instance - simply unstack all blocks and put them on the table in the first stage and incrementally build the requested goal state in the second stage. Such policies are called *generalized policies* and are dealt with in the generalized planning subfield. Generalized policies solve planning problems without any search, and while it may be useful to be able to find these policies, possibly with the help of language models [98], one should be careful generalizing the evidence obtained on these problems to the entire field of planning.

When looking at individual domains or problems, one must ask oneself, how difficult is it to solve? In cases when the semantics of the domain are known, some computational complexity results exist for individual domains, e.g., [50, 18, 26, 47]. In other cases, a large body of research investigated the computational complexity of different fragments of planning, mostly based on the structure of the ground planning task, such as causal graph structure, variables domains size and structure, reversibility/invertability of actions, action preconditions size, and the mixture of those, e.g., [5, 6, 12, 59, 49, 61, 25]. The rationale behind the investigation, in addition to understanding where the boundary between easy and hard problems lies, was in using the poly-time fragments for automatically deriving poly-time computable distance to goal approximations, also called *heuristic functions* [53, 27, 62]. Other difficulty approximations include various notions of *width*, meant to capture how hard it is to solve a problem, with the goal of exploiting the property algorithmically [13, 71, 22, 79].

In order to solve these problems, a variety of methods were introduced, starting with the famous STRIPS algorithm [30], a total ordering planning backward from the goal, keeping a stack of subgoals, iteratively resolving a top subgoal. The algorithm is sound but incomplete for satisficing planning, but a powerful planner from the 70s nonetheless. The 80s gave rise to methods based on Partial Order Planning, a search in the space of partial plans (sequences of actions), a sound and complete approach to satisficing planning. The 90s were the Golden Age of classical planning, introducing four different approaches to planning: GraphPlan, SAT-plan, heuristic search planning, and model checking planning. GraphPlan [9] is based on constructing a graph containing all possible parallel plans up to certain length in a forward manner and then extracting a plan by searching the graph backward from the goal. This is a sound approach to satisficing planning, with the completeness obtained by restarting with an increased the length bound. The other three approaches were not just both sound and complete. Importantly, these approaches could produce optimal solutions. SAT-plan [65] exploited SAT solvers, transforming the planning task into a boolean formula. By iteratively increasing the plan length bound, it allowed to optimize for plan length. Model Checking Planning [15] constructed a layered graph, compactly representing multiple states in a layer with compact data structures for representing boolean functions. Then, a backward search is performed from the goal to find a plan. Finally, the most popular approach to date is heuristic search planning, a forward search in the problem state space, with automatically extracted from the problem heuristic function. Over the years, a large variety of search algorithms and heuristic functions were introduced, some guaranteeing the obtained solutions to be optimal. One such famous algorithm is A*, a best-first search algorithm that expands the search nodes in the order of their $f = g + h$ values, where g is the cost of getting to the node from the initial state and h is the heuristic function value for the node. When the heuristic function is *admissible* (guaranteed not to over-estimate the true cost of getting to the goal), the algorithm is guaranteed to produce cost-optimal plans. One additional property of the algorithm is that it is optimally efficient [20], meaning there cannot be any algorithm in its class that can do better than A*.

When going beyond classical planning problems, popular approaches include the aforementioned transformations to classical planning and use of classical planners, as well as the use of more complex heuristic search algorithm families, such as MCTS/UCT or And/Or best-first search.

Another important aspect is the complexity of solution validation. In classical planning, solutions (aka plans) are applicable to the initial state sequences of actions that result in a goal state. Validating whether a sequence of actions is a plan is poly-time, but in order to validate that a plan is cost-optimal, one needs to prove that there is no plan of smaller cost, which is PSPACE-complete. In HTN planning, any solution validation is NP-hard. For non-deterministic planning, solutions are typically represented as policies mapping states to actions or compactly as controllers, and validating their complexity is poly-time in the solution size.

Common Pitfall 1. Historically, proposed planning algorithms were at least sound, albeit sometimes incomplete, meaning when they produce a solution, this solution is guaranteed to be correct. Naturally, an implementation may include bugs, and therefore it is a good practice to validate the solutions produced. Unsound algorithms do not have the guarantee to output only correct solutions, and therefore must be followed by a validation.

Common Pitfall 2. It is a common practice to compare planners that target the same computational problems. In most cases, a comparison between planners built for different computational problems—and therefore providing different guarantees on the produced plans—is less meaningful.

Common Pitfall 3. It is important to know the properties of the algorithms used and of the resulting solutions. One common error is to use A* search with a heuristic function that has no admissibility guarantees as a planner. In such cases, there is no guarantee for a produced plan to be cost-optimal, and validating cost-optimality can be extremely resource consuming. Further, while A* is optimally efficient for cost-optimal planning, it is extremely inefficient for satisficing planning, when no optimality guarantees are needed. This is due to the fact that it spends most of its efforts attempting to prove that there are no better plans.

Common Pitfall 4. While regression from goal [92] is a valuable tool in planning, one must know that the process creates so-called *spurious* states, which can be invalid or simply unreachable from the initial state. In both cases, one should take great care with such states.

4 The Data

The majority of datasets used for evaluation of LLM-based planners fall into one of the following categories: (a) existing games repurposed for multi-step planning/search problems, (b) natural language planning problems generated specifically for LLM evaluations, (c) existing PDDL domains, and (d) natural language datasets generated from PDDL domains. Figure 1 provides an overview of various benchmarks. In this section, we offer insights into the PDDL benchmarks creation and highlight a few common pitfalls and misconceptions in benchmark selection for evaluations. We propose a few key considerations to guide a robust and meaningful assessments of planning systems.

Common Pitfall 1. As previously mentioned, International Planning Competitions (IPC) played a pivotal role in the development of planning systems. The intention behind IPC was to test the planners on unknown, previously unseen benchmarks. The participants therefore submit their planners, which are run by the IPC organizers on a collection of planning problems, using the same hardware, under the same time and memory restrictions. The competition organizers therefore needed to come up with a collection of new PDDL domains every time the competition was run. These domains were meant to capture some abstraction of a real life problem, but often also were crafted to double down

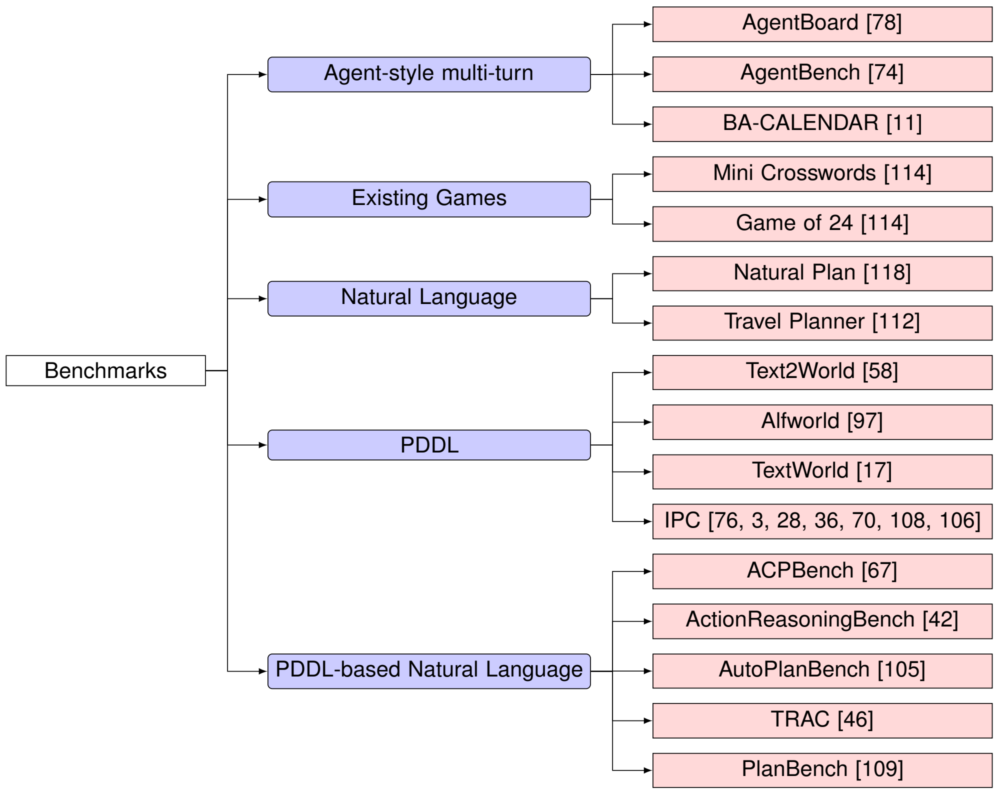

Figure 1: Benchmarks used in the literature for LLM-based Planning evaluations

on known limitations of popular approaches. One example is the *Gripper* domain, which introduces many *symmetries* into the state space, making it challenging for pure forward (heuristic) search based methods. This pushed the research towards investigating search pruning techniques [31, 91, 14]. Another aspect the competition organizers need to consider is the problem instances created for each domain. These problem instances should be challenging enough for the current state of the art, but also not too challenging, so a meaningful comparison would be possible. Consequently, a collection of problem instances of increasing difficulty was created. While initially domains varied significantly in the number of instances created for that competition, the later editions were more uniform. That way, a good performance on one domain would not swipe the scales when aggregating the performance across all instances. To automate the process, a tool to automatically scale the instances was developed, allowing us to create a uniform size benchmark set of instances of existing domains that are challenging for the current state of the art, for either cost-optimal, satisficing, or agile planning. Note that instances that are challenging for cost-optimal planning might be very easy for satisficing or agile planning, where no optimality is required.

Common Pitfall 2. A critical concern of evaluating models on publicly available planning domains and problem instances, such as BlocksWorld, Logistics, and Grid-based IPC benchmarks, is that the solutions are accessible through planning resources (c.f. Muise [83]). This issue is most aggravated when the benchmark is generated by scraping the data from the internet. For instance, the “Game of 24" and the “Mini crosswords" datasets [114] was generated by scraping `https://4nums.com` and `https://www.goobix.com/crosswords/0505/`, respectively. This raises the risk that answers may be included in the training data, leading to potential memorization and artificially inflated performance estimates. Indeed the contamination study by Hu et al. [58] shows that frontier models have memorized the PDDL. This underscores the need for careful attention to data provenance when selecting benchmarks. *Ideally, evaluation should be conducted on novel domains and problem instances that are guaranteed to be unseen during training.* Previously, similar concerns were addressed in the planning literature by introduction of “mystery domains", where predicates and object names were replaced with random words. The use of semantically uninformative or misleading names may introduce ambiguity for language models, potentially impairing their performance. Consequently, such domains may not be suitable for reliably evaluating the performance of language models. Hence, our practical proposal is to build generators that generate random instances for evaluations. Such generators are readily available for over 60 PDDL domains.

Common Pitfall 3. Another challenge in developing planning benchmarks is the lack of reliable ground truth and difficulty of verifying the correctness of the answers. In some domains, especially those involving open-ended reasoning, the correctness of a solution may not be binary. To overcome that challenge, certain datasets use LLMs for evaluation. For example, Butt et al. [11] and [42] use LLM as a judge for evaluating the generative tasks. While this has become acceptable approach for evaluation for some Question Answering tasks, it is not clear if such evaluation approach is reliable for planning problems. Another source of uncertainty in the ground truth also arises from the fact that some datasets are generated by scraping the internet. In such cases, individual examples and their ground truth are rarely verified. The unreliability of the data is most prolonged when they are generated using LLMs. For example: In an earlier version of AutoPlanBench [105], '(get-pop ?x)' action in the Movie domain is translated to "get population of an object" instead of getting a soda.

Common Pitfall 4. Greater care is required to split the data as equivalence and hardness is not immediately apparent from the questions. It is standard practice to *randomly* split generated data into a train-set and test-set for evaluating the performance of a model. While it may work in most cases, for data generated using PDDL-generators, a random split might not work. The PDDL-generator doesn't ensure that the generated problem instances are all unique. The generator might produce multiple problem instances that are structurally or functionally equivalent (i.e., isomorphic). Meaning that two instances might differ only in superficial aspects such as object naming or variable ordering, while preserving the same underlying structure and hence share the same solution strategy. Hence, selecting syntactically unique problem instances is not sufficient to evaluate the underlying reasoning ability of the model. To overcome this issue, some prior works partition the test and train split based on the number of objects or the plan-length. For example, train set might include problems with 3–7 objects where as test set might include problems with 7–20 objects. This approach ensures that the test instances are structurally different and systematically more complex then the train instances. However, caution is warranted when using such partitioning strategies, as they can give a misleading impression of "correctness". For instance, consider a BlocksWorld domain where test instances are said to include 7–20 blocks. If, in practice, the plan length is clipped at 10 steps, and assuming a 4-operator BlocksWorld domain, the planner can only move a maximum of 5 blocks. In such cases, despite the apparent increase in object count, the solutions for the test problems are structurally similar to those in the training set, undermining the intended evaluation [93].

5 Tools From The Planning Community

Recent years have seen a surge in development of tools for working with planning problems. From off-the-shelf planners to online services to debugging software, there is a whole host of software and libraries that researchers can take advantage of. Here, we detail some of what is now available.

For the manual specification of PDDL, an online editor is available for quick prototyping and solving of various planning formalisms, and an extended interface is available through the VSCode plugin for PDDL. Both of these services contain an integration with a suite of planners hosted in the cloud for free use (with limited resources). The service is open for programmatic access for anyone wishing to solve simple problems. The service is also available as an open source project for those who wish to host a version with different resources or planners [21].

For running planners and planning software directly, the planutils offers turn-key access to dozens of existing software, all pre-compiled [84]. This allows researchers to forgo the often fraught process of getting planners compiled and running. Not only does this include many state-of-the-art planning systems for various formalisms, but it also includes general planning software, such as model debugging [60] and plan validation [57, 44]. While planutils is a convenient way to access many tools, it is worth mentioning some of the more popular tools directly. First and foremost, is the most popular planning system Fast Downward [51], that incorporates implementations of many search algorithms, heuristics, pruning techniques, as well as methods for storing and accessing search queues. Many state-of-the-art planners are built on top of Fast Downward and find their way into the core code. As such, invoking Fast Downward with different parameters will result in planners for different formalisms, such as cost-optimal, satisficing, agile, etc. Another popular planning system is Pyperplan. Its popularity, however, stems from the ease of use rather than its power. It is important to know that Pyperplan was created for educational purposes and should not be used for experimental comparison. One category of particularly useful planning formalisms focuses on finding multiple solutions to planning problems. Planners for these formalisms are Forbid Iterative, K*, and SymK. All three can be conveniently invoked from Python. The former two are offered as PyPi packages. The latter is accessible via Unified Planning library.

Among the most useful tools for data generation are PDDL problems generators, offering customizable software to generate problem instances for over 50 unique planning benchmark domains [95].

Last, but not least, PDDL parsers and grounders allow one to parse, modify, and generate PDDL representations, as well as progress and regress through state spaces. Among the most commonly used are the pddl Python library , the *Tarski* parser [33], which can also be used via a lightweight wrapper, as well as CPDDL planning library, which implements a variety of tools for task manipulation. These libraries are absolutely invaluable when one must create PDDL content programmatically.

Common Pitfall 1. It is a good practice to mention which configuration is used, and it is absolutely necessary to mention what search problem is solved. For instance, saying that Fast Downward was used does not provide sufficient information about the planner, it could be a cost-optimal configuration, it could be a satisficing or agile one. It could also be a nonsensical configuration like A* search with inadmissible heuristic, which results in a highly inefficient way of finding non-optimal plans (refer to Common Pitfall 3. in Section 3).

Common Pitfall 2. Packages that allow to progress from one state to another via action application are based on the process of *grounding* a planning task. As the tools are oriented towards creating planning tasks for an efficient search, they often get rid of an information that is redundant for the search process. Such information includes so-called *static* information, that does not change from one state to another, like game maps or object types. While symbolic planners do not need the information once the task is grounded, LLM-based planners might find this information crucial. Being aware of how the tools work may help preserve the needed information.

6 On Evaluating Planners

In this section, we put everything together to describe what typically constitute a rigorous scientific evaluation of a planning method, or simply a *planner*. When proposing a new planner, it is important to clearly articulate the assumptions made in terms of the characteristics of the planning problems being addressed. As we discussed in Section 2, planning problems vary widely, each consider different assumptions. Clearly stating these assumptions allows the authors to accurately position their contribution within the relevant literature, select appropriate baselines for evaluation, and most importantly avoid claims that are too general and are not supported by the scope of the work. Our overview in Section 2 is by no means complete, but can serve as a useful starting point for researchers to better understand the extensive AI Planning literature and to properly situate their contributions.

Common Pitfall 1. It is critically important to have an understanding of the computational complexity of the problem at hand. For any newly proposed planner, it is standard practice to provide the complexity analysis of the planner, along with formal properties such as soundness, completeness, and where applicable, optimality [64]. These considerations are particularly important when selecting baselines. In general, unsound methods should be avoided, or, at the very least, not compared directly to sound approaches, as they do not provide the same guarantees. For newly proposed domains/datasets, if no validator exists, it is important to provide a sound validator. It is usually a good practice not to provide both the new planner and the new dataset for the planner to be tested on.

Common Pitfall 2. The most common evaluation metric is the overall aggregated coverage, where the coverage per instance is 1 if solved, otherwise 0. This is similar to the common success rate metric. While in principle this metric works for many computational problems, both optimal and non-optimal settings, the metric is more suitable for optimal settings, when all found plans should be of the same quality. For satisficing and agile settings a different metric was proposed, known as the *IPC score*. This metric scores the method relatively to other competitors performance per instance. For satisficing setting, each instance gets the value $\frac{c^*}{c}$ where c^* is the best among the plan costs obtained by all the competitors, and c is the cost of the plan obtained by the method. For agile planning, the time to get the solution is used instead of the plan cost. In cases when a search component is proposed, a measure of search effort, such as the number of expanded nodes for optimal search algorithms and the number of generated nodes for satisficing search. It is a good practice to evaluate an approach on a large variety of domains, to reduce the bias to particular domains. Further, it is common to compare methods under the same hardware and resource restrictions.

Common Pitfall 3. When proposing planners that do not have soundness guarantees, as it common in LLM-based planning literature [111, 115, 114, 96, 34], the planner must be paired with a *sound* validator, which can make the overall algorithm technically sound. It is important to separate the validation from the solution, reducing the possibility for error.

Common Pitfall 4. In the cost-optimal track of the IPC, when the planners must provide optimal solutions, if a participant returns non-optimal solutions it is disqualified, since the planner is supposed to provide guaranteed cost-optimal solutions. Recall that it is hard to validate that a solution is cost-optimal, and therefore we are meant to *trust* the cost-optimal planner. It does not matter how often the solution produced *happens to be* cost-optimal, if no such guarantee exists, and therefore it is meaningless to measure [69].

Common Pitfall 5. The search effort comparison is meaningful only when the same search algorithm is used, evaluating the relative power of the proposed heuristic function or pruning technique compared to existing ones. Further, this should be a reasonable algorithm for the problem at hand. For example, A* is not a reasonable choice of an algorithm for non-optimal planning, as it puts most of the effort in proving optimality rather than finding a plan. If, in addition, the proposed heuristic function is not guaranteed to not over-estimate the true goal cost, then this additional effort is essentially wasted, as optimality of the found solution cannot be guaranteed. Comparing the search efforts across algorithms might be tricky, even if they provide the same plan quality guarantees. If the plan quality guarantees are different, the comparison is meaningless, as they solve different problems, sometimes of different complexity (recall the discussion of BlocksWorld in Section 3). Thus, one might want to avoid claims of *"Better Planning"* based on such observations [69].

Common Pitfall 6. As discussed in Section 4, the choice of datasets for evaluation is equally important. First, the dataset must be relevant to the problem at hand and suitable for evaluating the proposed approach as well as the baselines. Moreover, it is important to evaluate performance across a variety of domains and to show how the proposed method scales with increasing problem instance difficulty. For example, many existing datasets, such as those in [114], consists of instances of similar size and complexity (e.g., all instances of the 24 game are of the same size), which limits the generality of the evaluation. Second, care must be taken to decouple the proposed method from the dataset to ensure unbiased evaluation. Unfortunately, it has become common practice to introduce both a new dataset and a new method in the same paper, potentially leading to evaluation bias. IPC benchmarks were established to help mitigate this issue by providing standardized evaluation domains. Please refer to 4 for further discussion on pitfalls related to dataset selection.

Common Pitfall 7. A key aspect of high-quality scientific paper is the transparency of the experimental setup. This includes clearly specifying the computation resources used, time limits, the number of LLM calls, and all the tools involved in both the proposed approach or baseline approaches. A frequent issue is the insufficient specification of the planner (see Common Pitfall 1. in Section 5), or failing to mention time or memory constraints placed on the planner. Such oversights make it hard to replicate the results from the current paper.

7 Conclusions

As we see more and more focus on solving planning problems using LLM-based tools, it is now more important than ever to approach this problem in a systematic and rigorous way. This would not only help us take an accurate stock of the state-of-the-art, but also prevent us from fooling ourselves into thinking we have made more progress than we actually have. One of our main proposals towards this goal would be to adopt insights, methodologies, tools, and data from the automated planning community and incorporate them in the design and evaluation of our AI systems. As discussed in the paper, existing work from the automated planning community provides us with a useful framework to categorize and analyze the various flavors of planning, including identifying their computational complexities. The field has not only developed an extensive set of benchmarks that could be directly adopted by researchers working on LLM-based planners, but also developed helpful tools and established useful experimental protocols. In addition to making this case, this paper also details basic terminology and pointers that could act as a starting point for researchers to learn more about existing work and the state of the art. In each section, we also laid out some advice for researchers less familiar with the field and some common pitfalls that should be avoided. Ultimately, we hope this paper can serve as a handbook of insight into how planning-based research can be rigorously conducted, both for researchers and reviewers alike.

References

[1] Meysam Aghighi and Peter Jonsson. Oversubscription planning: Complexity and compilability. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, pages 2221–2227. AAAI Press, 2014.

[2] Ron Alford, Ugur Kuter, and Dana Nau. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1629–1634. AAAI Press, 2009.

[3] Fahiem Bacchus. The AIPS'00 planning competition. *AI Magazine*, 22(3):47–56, 2001.

[4] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22(1,1):5–27, 1998.

[5] Christer Bäckström and Inger Klein. Planning in polynomial time: the SAS-PUBS class. *Computational Intelligence*, 7(3):181–197, 1991.

[6] Christer Bäckström and Bernhard Nebel. Complexity results for SAS$^+$ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[7] Jorge A. Baier and Sheila A. McIlraith. Planning with temporally extended goals using heuristic search. In Derek Long, Stephen F. Smith, Daniel Borrajo, and Lee McCluskey, editors, *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, pages 342–345. AAAI Press, 2006.

[8] Jorge A. Baier and Sheila A. McIlraith. Planning with preferences. *AI Magazine*, 29(4):25–36, 2008.

[9] Avrim Blum and Merrick L. Furst. Fast planning through planning graph analysis. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, pages 1636–1642. Morgan Kaufmann, 1995.

[10] Blai Bonet and Héctor Geffner. Planning with incomplete information as heuristic search in belief space. In Steve Chien, Subbarao Kambhampati, and Craig A. Knoblock, editors, *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, pages 52–61. AAAI Press, 2000.

[11] Natasha Butt, Varun Chandrasekaran, Neel Joshi, Besmira Nushi, and Vidhisha Balachandran. Benchagents: Automated benchmark creation with agent interaction. *arXiv preprint arXiv:2410.22584*, 2024.

[12] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

[13] Hubie Chen and Omer Giménez. Act local, think global: Width notions for tractable planning. In Mark Boddy, Maria Fox, and Sylvie Thiébaux, editors, *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, pages 73–80. AAAI Press, 2007.

[14] Yixin Chen and Guohui Yao. Completeness and optimality preserving reduction for planning. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1659–1664. AAAI Press, 2009.

[15] Alessandro Cimatti, Fausto Giunchiglia, Enrico Giunchiglia, and Paolo Traverso. Planning via model checking: A decision procedure for *AR*. In Sam Steel and Rachid Alami, editors, *Recent Advances in AI Planning. 4th European Conference on Planning (ECP 1997)*, volume 1348 of *Lecture Notes in Artificial Intelligence*, pages 130–142. Springer-Verlag, 1997.

[16] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (STOC 1971)*, pages 151–158. ACM, 1971.

[17] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018*, pages 41–75. Springer, 2019.

[18] Joseph C. Culberson. Sokoban is PSPACE-complete. Technical Report TR 97-02, Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada, 1997.

[19] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 854–860. AAAI Press, 2013.

[20] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A^*. *Journal of the ACM*, 32(3):505–536, 1985.

[21] Yi Ding, Cam Cunningham, Christian Muise, and Nir Lipovetzky. Planning as a service. In *ICAPS 2023 System Demonstrations and Exhibits*, 2023.

[22] Simon Dold and Malte Helmert. Novelty vs. potential heuristics: A comparison of hardness measures for satisficing planning. In Jennifer Dy and Sriraam Natarajan, editors, *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2024)*, pages 20692–20699. AAAI Press, 2024.

[23] Carmel Domshlak and Jörg Hoffmann. Probabilistic planning via heuristic forward search and weighted model counting. *Journal of Artificial Intelligence Research*, 30:565–620, 2007.

[24] Carmel Domshlak and Vitaly Mirkis. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *Journal of Artificial Intelligence Research*, 52: 97–169, 2015.

[25] Carmel Domshlak, Jörg Hoffmann, and Michael Katz. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221:73–114, 2015.

[26] Dorit Dor and Uri Zwick. SOKOBAN and other motion planning problems. *Computational Geometry*, 13:215–228, 1999.

[27] Stefan Edelkamp. Planning with pattern databases. In Amedeo Cesta and Daniel Borrajo, editors, *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, pages 84–90. AAAI Press, 2001.

[28] Stefan Edelkamp and Jörg Hoffmann. PDDL2.2: The language for the classical part of the 4th International Planning Competition. Technical Report 195, University of Freiburg, Department of Computer Science, 2004.

[29] Kutluhan Erol, James A. Hendler, and Dana S. Nau. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI 1994)*, pages 1123–1128. AAAI Press, 1994.

[30] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[31] Maria Fox and Derek Long. The detection and exploitation of symmetry in planning problems. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 1999)*, pages 956–961. Morgan Kaufmann, 1999.

[32] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[33] Guillem Francés, Miquel Ramirez, and Collaborators. Tarski: An AI planning modeling framework. `https://github.com/aig-upf/tarski`, 2018.

[34] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of Search (SoS): Learning to search in language. arXiv:2404.03683 [cs.LG], 2024.

[35] Héctor Geffner. Functional Strips: A more flexible language for planning and problem solving. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, volume 597 of *Kluwer International Series In Engineering And Computer Science*, chapter 9, pages 187–209. Kluwer, Dordrecht, 2000.

[36] Alfonso E. Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5–6):619–668, 2009.

[37] Alfonso E. Gerevini, Alessandro Saetti, and Ivan Serina. Temporal planning with problems requiring concurrency through action graphs and local search. In Ronen Brafman, Héctor Geffner, Jörg Hoffmann, and Henry Kautz, editors, *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS 2010)*, pages 226–229. AAAI Press, 2010.

[38] Elliot Gestrin, Marco Kuhlmann, and Jendrik Seipp. NL2Plan: Robust LLM-driven planning from minimal text descriptions. In *ICAPS 2024 Workshop on Human-Aware and Explainable Planning (HAXP)*, 2024.

[39] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice*. Elsevier, 2004. ISBN 978-1-55860-856-6.

[40] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016. ISBN 978-1-107-03727-4.

[41] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023.

[42] Divij Handa, Pavel Dolin, Shrinidhi Kumbhar, Chitta Baral, and Tran Cao Son. Actionreasoningbench: Reasoning about actions with and without ramification constraints. In *The Thirteenth International Conference on Learning Representations*, 2025.

[43] 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.

[44] Patrik Haslum. INVAL: the Independent PDDL plan Validator. `https://github.com/patrikhaslum/INVAL`, 2016. Accessed May 2, 2025.

[45] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. *An Introduction to the Planning Domain Definition Language*, volume 13 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool, 2019.

[46] Weinan He, Canming Huang, Zhanhao Xiao, and Yongmei Liu. Exploring the capacity of pretrained language models for reasoning about actions and change. In *ACL*. Association for Computational Linguistics, 2023.

[47] Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1–2):72–96, 2005.

[48] Malte Helmert. Decidability and undecidability results for planning with numerical state variables. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, pages 303–312. AAAI Press, 2002.

[49] Malte Helmert. Complexity results for standard benchmark domains in planning. *Artificial Intelligence*, 143(2):219–262, 2003.

[50] Malte Helmert. New complexity results for classical planning benchmarks. In Derek Long, Stephen F. Smith, Daniel Borrajo, and Lee McCluskey, editors, *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, pages 52–61. AAAI Press, 2006.

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

[52] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535, 2009.

[53] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[54] Chad Hogg, Héctor Muñoz-Avila, and Ugur Kuter. HTN-MAKER: Learning HTNs with minimal additional knowledge engineering required. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, pages 950–956. AAAI Press, 2008.

[55] Chad Hogg, Ugur Kuter, and Héctor Muñoz-Avila. Learning hierarchical task networks for nondeterministic planning domains. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1708–1714. AAAI Press, 2009.

[56] Chad Hogg, Ugur Kuter, and Héctor Muñoz-Avila. Learning methods to generate good plans: Integrating HTN learning and reinforcement learning. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, pages 1530–1535. AAAI Press, 2010.

[57] Richard Howey and Derek Long. VAL's progress: The automatic validation tool for PDDL2.1 used in the International Planning Competition. In Stefan Edelkamp and Jörg Hoffmann, editors, *Proceedings of the ICAPS 2003 Workshop on the Competition: Impact, Organisation, Evaluation, Benchmarks*, 2003.

[58] Mengkang Hu, Tianxing Chen, Yude Zou, Yuheng Lei, Qiguang Chen, Ming Li, Yao Mu, Hongyuan Zhang, Wenqi Shao, and Ping Luo. Text2world: Benchmarking large language models for symbolic world model generation, 2025. URL `https://arxiv.org/abs/2502.13092`.

[59] Peter Jonsson and Christer Bäckström. Tractable plan existence does not imply tractable plan generation. *Annals of Mathematics and Artificial Intelligence*, 22(3,4):281–296, 1998.

[60] Lucas Galery Käser, Clemens Büchner, Augusto B. Corrêa, Florian Pommerening, and Gabriele Röger. Machetli: Simplifying input files for debugging. In *ICAPS 2022 System Demonstrations and Exhibits*, 2022.

[61] Michael Katz and Carmel Domshlak. New islands of tractability of cost-optimal planning. *Journal of Artificial Intelligence Research*, 32:203–288, 2008.

[62] Michael Katz and Carmel Domshlak. Implicit abstraction heuristics. *Journal of Artificial Intelligence Research*, 39:51–126, 2010.

[63] Michael Katz and Vitaly Mirkis. In search of tractability for partial satisfaction planning. In Subbarao Kambhampati, editor, *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, pages 3154–3160. AAAI Press, 2016.

[64] Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi. Thought of search: Planning with language models through the lens of efficiency. In *Proceedings of the Thirty-Eighth Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*, pages 138491–138568, 2024.

[65] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI 1996)*, pages 1194–1201. AAAI Press, 1996.

[66] Emil Keyder and Héctor Geffner. Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, 36:547–556, 2009.

[67] Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. ACPBench: Reasoning about action, change, and planning. In Julie Shah and Zico Kolter, editors, *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2025)*. AAAI Press, 2025.

[68] Ugur Kuter, Dana S. Nau, Elnatan Reisner, and Robert P. Goldman. Using classical planners to solve nondeterministic planning problems. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric Hansen, editors, *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 190–197. AAAI Press, 2008.

[69] Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul McVay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping. In *Proceedings of the First Conference on Language Modeling (COLM 2024)*, 2024.

[70] Carlos Linares López, Sergio Jiménez Celorrio, and Angel García Olaya. The deterministic part of the seventh international planning competition. *Artificial Intelligence*, 223:82–119, 2015.

[71] Nir Lipovetzky and Hector Geffner. Width and serialization of classical planning problems. In Luc De Raedt, Christian Bessiere, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter Lucas, editors, *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 540–545. IOS Press, 2012.

[72] Michael L. Littman. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI 1997)*, pages 748–754. AAAI Press, 1997.

[73] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: empowering large language models with optimal planning proficiency. *CoRR*, abs/2304.11477, 2023.

[74] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. *CoRR*, abs/2308.03688, 2023.

[75] Derek Long and Maria Fox. Exploiting a graphplan framework in temporal planning. In Enrico Giunchiglia, Nicola Muscettola, and Dana Nau, editors, *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003)*, pages 51–62. AAAI Press, 2003.

[76] Derek Long, Henry Kautz, Bart Selman, Blai Bonet, Héctor Geffner, Jana Koehler, Michael Brenner, Jörg Hoffmann, Frank Rittinger, Corin R. Anderson, Daniel S. Weld, David E. Smith, and Maria Fox. The AIPS-98 planning competition. *AI Magazine*, 21(2):13–33, 2000.

[77] Inês Lynce and Joël Ouaknine. Sudoku as a sat problem. In *AI&M*. Citeseer, 2006.

[78] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn LLM agents. *CoRR*, abs/2401.13178, 2024.

[79] Jiayuan Mao, Tomas Lozano-Perez, Joshua B. Tenenbaum, and Leslie Pack Kaelbing. What planning problems can a relational neural network solve? In *Proceedings of the Thirty-Seventh Annual Conference on Neural Information Processing Systems (NeurIPS 2023)*, pages 59522–59542, 2023.

[80] Robert Mattmüller, Manuela Ortlieb, Malte Helmert, and Pascal Bercher. Pattern database heuristics for fully observable nondeterministic planning. In Ronen Brafman, Héctor Geffner, Jörg Hoffmann, and Henry Kautz, editors, *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS 2010)*, pages 105–112. AAAI Press, 2010.

[81] Mausam and Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2012.

[82] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL – The Planning Domain Definition Language – Version 1.2. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University, 1998.

[83] Christian Muise. Planning.Domains. In *ICAPS 2016 System Demonstrations and Exhibits*, 2016. https://api.planning.domains.

[84] Christian Muise, Florian Pommerening, Jendrik Seipp, and Michael Katz. Planutils: Bringing planning to the masses. In *ICAPS 2022 System Demonstrations and Exhibits*, 2022.

[85] Christian J. Muise, Sheila A. McIlraith, and J. Christopher Beck. Improved non-deterministic planning by exploiting state relevance. In Lee McCluskey, Brian Williams, José Reinaldo Silva, and Blai Bonet, editors, *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, pages 172–180. AAAI Press, 2012.

[86] Christian J. Muise, Sheila A. McIlraith, and Vaishak Belle. Non-deterministic planning with conditional effects. In Steve Chien, Alan Fern, Wheeler Ruml, and Minh Do, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, pages 370–374. AAAI Press, 2014.

[87] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[88] James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. Large language models as planning domain generators. In Sara Bernardini and Christian Muise, editors, *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, pages 423–431. AAAI Press, 2024.

[89] Hector Palacios and Hector Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, 35:623–675, 2009.

[90] Chiara Piacentini, Maria Fox, and Derek Long. Planning with numeric timed initial fluents. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 4196–4197. AAAI Press, 2015.

[91] Nir Pochter, Aviv Zohar, and Jeffrey S. Rosenschein. Exploiting problem symmetries in state-based planners. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, pages 1004–1009. AAAI Press, 2011.

[92] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[93] Swarnadeep Saha, Archiki Prasad, Justin Chen, Peter Hase, Elias Stengel-Eskin, and Mohit Bansal. System 1.x: Learning to balance fast and slow planning with language models. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025)*. OpenReview.net, 2025.

[94] Scott Sanner. Relational dynamic influence diagram language (RDDL): Language description, 2010.

[95] Jendrik Seipp, Álvaro Torralba, and Jörg Hoffmann. PDDL generators. `https://doi.org/10.5281/zenodo.6382173`, 2022.

[96] Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Lu Wang, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *CoRR*, abs/2308.10379, 2023.

[97] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *Proceedings of the Nineth International Conference on Learning Representations (ICLR 2021)*. OpenReview.net, 2021.

[98] Tom Silver, Soham Dan, Kavitha Srinivas, Josh Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. Generalized planning in PDDL domains with pretrained large language models. In Jennifer Dy and Sriraam Natarajan, editors, *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2024)*, pages 20256–20264. AAAI Press, 2024.

[99] Helmut Simonis. Sudoku as a constraint problem. In *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, volume 12, pages 13–27. Citeseer Sitges, Spain, 2005.

[100] Aravinda Prasad Sistla and Edmund Melson Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[101] David E. Smith. Choosing objectives in over-subscription planning. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 393–401. AAAI Press, 2004.

[102] David E. Smith and Daniel S. Weld. Conformant graphplan. In Charles Rich and Jack Mostow, editors, *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI 1998)*, pages 889–896. AAAI Press, 1998.

[103] Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. HTN planning with preferences. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1790–1797. AAAI Press, 2009.

[104] Siddharth Srivastava, Shlomo Zilberstein, Neil Immerman, and Hector Geffner. Qualitative numeric planning. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, pages 1010–1016. AAAI Press, 2011.

[105] Katharina Stein, Daniel Fišer, Jörg Hoffmann, and Alexander Koller. Automating the generation of prompts for llm-based action choice in pddl planning. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2025)*. AAAI Press, 2025.

[106] Ayal Taitler, Ron Alford, Joan Espasa, Gregor Behnke, Daniel Fišer, Michael Gimelfarb, Florian Pommerening, Scott Sanner, Enrico Scala, Dominik Schreiber, Javier Segovia-Aguas, and Jendrik Seipp. The 2023 International Planning Competition. *AI Magazine*, 45(2):280–296, 2024. doi: 10.1002/aaai.12169.

[107] Marcus Tantakoun, Xiaodan Zhu, and Christian Muise. LLMs as planning modelers: A survey for leveraging large language models to construct automated planning models. *CoRR*, abs/2503.18971, 2025.

[108] Mauro Vallati, Lukáš Chrpa, and Thomas L. McCluskey, editors. *The Eighth International Planning Competition: Description of Participant Planners of the Deterministic Track*, 2014.

[109] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Proceedings of the Thirty-Seventh Annual Conference on Neural Information Processing Systems (NeurIPS 2023)*, pages 38975–38987, 2023.

[110] Menkes van den Briel, Romeo Sanchez, Minh B. Do, and Subbarao Kambhampati. Effective approaches for partial satisfaction (over-subscription) planning. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, pages 562–569. AAAI Press, 2004.

[111] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the Thirty-Sixth Annual Conference on Neural Information Processing Systems (NeurIPS 2022)*, pages 24824–24837, 2022.

[112] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. In *ICML*. OpenReview.net, 2024.

[113] Qiang Yang, Kangheng Wu, and Yunfei Jiang. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence*, 171:107–143, 2007.

[114] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of the Thirty-Seventh Annual Conference on Neural Information Processing Systems (NeurIPS 2023)*, pages 11809–11822, 2023.

[115] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR 2023)*. OpenReview.net, 2023.

[116] Sungwook Yoon, Alan Fern, Robert Givan, and Subbarao Kambhampati. Probabilistic planning via determinization in hindsight. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, pages 1010–1016. AAAI Press, 2008.

[117] Håkan L. S. Younes, Michael L. Littman, David Weissman, and John Asmuth. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, 24:851–887, 2005.

[118] Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. NATURAL PLAN: benchmarking llms on natural language planning. *CoRR*, abs/2406.04520, 2024.

[119] Max Zuo, Francisco Piedrahita Velez, Xiaochen Li, Michael L. Littman, and Stephen H. Bach. Planetarium: A rigorous benchmark for translating text to structured planning languages. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2025)*, pages 11223–11240, 2025.