Decoupled Search for the Masses: A Novel Task Transformation for Classical Planning (Extended Abstract)*

David Speck[1,2] , **Daniel Gnad**[2,3]
[1]University of Basel, Switzerland
[2]Linköping University, Sweden
[3]Heidelberg University, Germany
davidjakob.speck@unibas.ch, daniel.gnad@liu.se

Abstract

Classical planning provides a framework for solving sequential decision-making problems, i. e., finding a sequence of actions that transforms the current state of the world into a state that satisfies a desired goal condition. Planning tasks are modeled in a logic that describes the environment and its dynamics. It is well known that the specific problem formulation can significantly affect the performance of planning systems solving problems like the Rubik's Cube or finding algorithms for matrix multiplication. In this work, we propose a domain-general problem reformulation that embodies decoupled search, a search-reduction technique from classical planning and model checking. Decoupled search decomposes a given problem to exploit its structure, achieving exponential reductions over other search techniques. We show that decoupled search can be captured exactly as a task reformulation and that, on many benchmark domains, it performs as good and sometimes even better than a native decoupled-search implementation.

1 Introduction

Classical planning is concerned with finding a sequence of operators (sometimes called actions) that transforms the initial state of a problem into a desired goal state. To solve planning tasks, a representation is required that allows to search for a solution within the induced state space. Both theory and practice show that the way planning problems are represented has a significant impact on the performance and success rate of many planning techniques.

In domain-specific settings, problem reformulations can be approached in a very targeted way. Common examples include solving puzzles such as the Rubik's Cube, where the search is not over atomic operators but over macro operators [Korf, 1997]. Similarly, in the design of algorithms for matrix multiplication, the search is often not in the space of arithmetic instructions, but encapsulated as a tensor decomposition [Fawzi *et al.*, 2022; Speck *et al.*, 2023]. In general, reformulating a problem can yield an alternative state space that may differ significantly in size and structure from the original one, while at the same time facilitating the search. Thus, problem reformulation is highly relevant in both domain-specific and domain-independent planning.

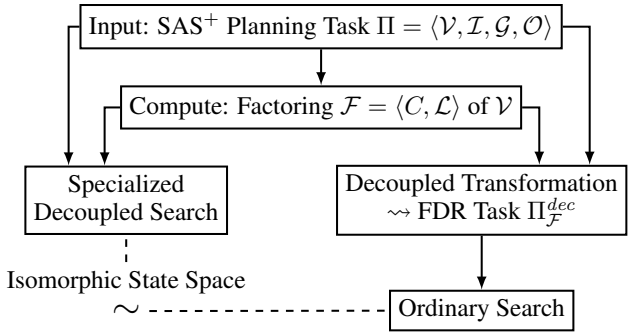

Figure 1: Overview of our approach, which embodies decoupled search, a decomposition technique for planning problems, through a task transformation. This eliminates the need for specialized algorithms and allows the full toolbox of planning techniques to be combined with decoupled search.

In this work, we study decoupled state-space search, which is based on a compact state representation, similar to symbolic search [McMillan, 1993] with binary decision diagrams [Bryant, 1986]. Decoupled search automatically decomposes a planning problem into conditionally independent leaf components with a synchronizing center factor that interacts with the leaves [Gnad and Hoffmann, 2018]. It can achieve an exponential reduction in search effort, which empirically leads to significant speed-ups and memory reductions when solving many planning problems.

The key contribution of our work is to show that it is possible to simulate decoupled search for non-optimal planning via a task transformation within the widely supported finite-domain representation formalism (FDR) [Helmert, 2009]. More precisely, we demonstrate that given a SAS$^+$ planning task (a subset of FDR) [Bäckström and Nebel, 1995], we can decompose the task as usual for decoupled search and create a FDR planning task for which the induced state space is isomorphic to that of decoupled search on the original task (Figure 1). Thus, a search algorithm on the transformed planning

*This is an abridged version of a paper that won the Best Paper Award at the International Conference on Automated Planning and Scheduling (ICAPS) 2024 [Speck and Gnad, 2024b].

task behaves in the same way as its native decoupled-search counterpart. This alleviates a major drawback of decoupled search — the need for specialized algorithms — and enables the full toolbox of past and future planning technology within the decoupled-search framework.

2 Background

We provide a brief background on classical planning and decoupled search to motivate and contextualize our approach of realizing decoupled search as a task reformulation. For a comprehensive formal introduction we refer to the full paper [Speck and Gnad, 2024b].

2.1 Classical Planning

Classical planning provides a declarative framework for sequential decision-making in complex environments. It deals with the problem of finding a sequence of operators that, starting from the current state of the world, leads to a state that satisfies the desired goal properties. Classical planning tasks are typically modeled in the SAS$^+$ formalism [Bäckström and Nebel, 1995], which uses propositional logic over a set of finite-domain variables $\mathcal{V}$ to describe the environment. Changes in the environment are described by deterministic operators $\mathcal{O}$, which are pairs of *precondition*, i. e., a set of variable assignments that have to be satisfied for the operator to be *applicable*, and *effect*, a set of assignments that is true after the operator is applied. A *state* is a full variable assignment. The solution of a planning task, a *plan*, is a sequence of operators that is applicable in the *initial state* $\mathcal{I}$ and ends in a state that satisfies the *goal condition* $\mathcal{G}$.

A planning task induces a transition system with states defined as complete variable assignments to $\mathcal{V}$ and transitions induced by the operators. Solving the task corresponds to finding a path from the initial state to a goal state.

The following example illustrates a simple logistics problem modeled as a SAS$^+$ planning task.

Example 1 (Running Example)**.** *Let us consider a simple logistics scenario with two connected locations, l_1 and l_2, along with two packages, p_1 and p_2, and one truck t. These are represented by the variables $\mathcal{V} = \{t, p_1, p_2\}$, with domains: $D_t = \{l_1, l_2\}$ and $D_{p_1} = D_{p_2} = \{l_1, l_2, t\}$.*

Initially, both the packages and the truck are at position l_1, modeled as $\mathcal{I}(v) = l_1$ for all $v \in \mathcal{V}$. The goal is to transport both packages to l_2, given as $\mathcal{G} = \{p_1 = l_2, p_2 = l_2\}$. Both the initial and goal states are illustrated in Figure 2.

There are three types of operators in this example: drive *operators that drive the truck between locations,* load *operators responsible for loading a package onto the truck, and* unload *operators for unloading a package from the truck. Formally, we have for any $i, j \in \{1, 2\}$:*

- $\mathrm{drive}(l_i, l_j) = \langle \{t = l_i\}, \{t = l_j\} \rangle$ *with* $i \neq j$
- $\mathrm{load}(p_i, l_j) = \langle \{t = l_j, p_i = l_j\}, \{p_i = t\} \rangle$
- $\mathrm{unload}(p_i, l_j) = \langle \{t = l_j, p_i = t\}, \{p_i = l_j\} \rangle$

A possible plan is to first load the two packages into the truck, then drive the truck to l_2, and unload both packages.

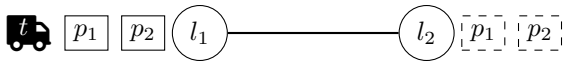

Figure 2: Illustration of the initial state (solid packages) and goal (dashed packages) of the running example.

The number of states in the example task grows exponential with the number of packages. This can pose a significant challenge to modern search algorithms.

2.2 Decoupled Search

Decoupled search is a search reduction technique that reformulates the state space of SAS$^+$ planning tasks and employs this alternative representation to find a plan [Gnad and Hoffmann, 2018]. It can efficiently solve problems like Example 1 by identifying and exploiting the causal structure of the task, i.e., the variable dependencies, via problem decomposition.

As a first step, decoupled search partitions the variables $\mathcal{V}$ of a planning task Π into a tuple $\mathcal{F} = \langle C, \mathcal{L} \rangle$ with $C \subseteq \mathcal{V}$ and $\mathcal{L} \subseteq 2^{\mathcal{V}}$. $\mathcal{F}$ is a *factoring* for a task Π if either $\{C\} \cup \mathcal{L}$ or $\mathcal{L}$ forms a partition of the set of variables $\mathcal{V}$. Then C represents the (possibly empty) *center* of $\mathcal{F}$, while $\mathcal{L}$ denotes its *leaves*. A complete assignment to the center C or to a leaf $L \in \mathcal{L}$ is called a *center state* or *leaf state*, respectively.

A factoring induces a separation of the set of operators $\mathcal{O}$ into *global operators* $\mathcal{O}^G$, which affect a center variable or touch multiple leaves, and the *leaf operators* $\mathcal{O}^{\mathcal{L}}$.[1] Important are the *leaf-only operators* for a leaf L, which have effects only on variables in L and preconditions on $C \cup L$.

Example 2. *A natural factoring $\mathcal{F}_t$ for the planning task outlined in Example 1 is $\mathcal{F}_t = \langle \{t\}, \{\{p_1\}, \{p_2\}\} \rangle$. Here, the truck forms the center $C = \{t\}$, while each package p_i forms a leaf $L_i = \{p_i\}$. The operators* load *and* unload *are leaf-only operators, with preconditions concerning the truck (center) and the respective package (leaf), and effects concerning the package (leaf) only. Conversely, the truck* drive *operators represent global operators, with preconditions and effects that only affect the truck (center).*

An alternative factoring, $\mathcal{F}_p = \langle \{p_1, p_2\}, \{\{t\}\} \rangle$, puts the package variables into the center, while assigning the truck to a leaf. Thus, in $\mathcal{F}_p$, the roles of the operators are swapped, i.e., the drive *operators become leaf-only operators, while the* load *and* unload *operators act as global operators.*

The search is then done over so-called *decoupled states* $s^{\mathcal{D}}$, which are pairs $\langle \mathsf{center}(s^{\mathcal{D}}), \mathsf{leaves}(s^{\mathcal{D}}) \rangle$, where $\mathsf{center}(s^{\mathcal{D}})$ is a center state and $\mathsf{leaves}(s^{\mathcal{D}})$ is a set of leaf states for each leaf factor $L \in \mathcal{L}$. In essence, a decoupled state $s^{\mathcal{D}}$ represents a set of explicit states from the original planning task Π, all sharing the same center state $\mathsf{center}(s^{\mathcal{D}})$, and where the leaf variables can take values from the cross-product of the leaf states $\mathsf{leaves}(s^{\mathcal{D}})$ across the leaves $\mathcal{L}$. Transitions between decoupled states are induced only by global operators, while the leaf-only operators serve to *saturate* the set of leaf states in a decoupled state. In particular, after the application of a global operator, the saturation computes the set of all leaf states that can be reached via leaf-only operators. The set of

[1] An operator can be both a global and a leaf operator.

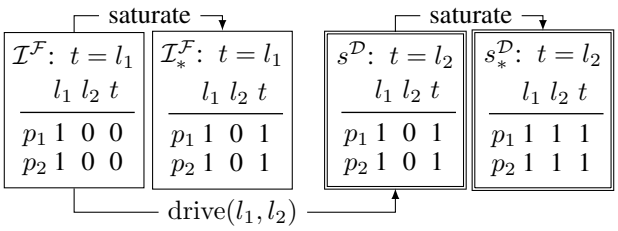

Figure 3: Illustration of the decoupled state space needed to determine a decoupled plan for the running example.

saturated leaf states is denoted by $\mathsf{leaves}^*(s^{\mathcal{D}})$. In the initial decoupled state $\mathcal{I}^{\mathcal{F}}$, only the leaf state resulting from the projection of the initial state $\mathcal{I}$ onto each leaf $L \in \mathcal{L}$ is reached, which is then saturated (denoted by $\mathcal{I}^{\mathcal{F}}_*$). After every application of a global operator, the set of reached leaf states is updated and saturated.

This decoupled state representation enables a compact description of a possibly large set of states, potentially leading to an exponential reduction in search effort compared to explicit state-space search in the regular state space.

Example 3. *Consider our running example with the factoring $\mathcal{F} = \mathcal{F}_t$. Part of the decoupled state space is illustrated in Figure 3. In the unsaturated initial state $\mathcal{I}^{\mathcal{F}}$ and its saturated counterpart $\mathcal{I}^{\mathcal{F}}_*$, the single center variable t has the value l_1. In $\mathcal{I}^{\mathcal{F}}$, each leaf has a single leaf state, $\{p_1 = l_1\}$ and $\{p_2 = l_1\}$, indicating the initial location l_1 of both packages. In the saturated initial state $\mathcal{I}^{\mathcal{F}}_*$, we have a total of four leaf states, namely $\mathsf{leaves}^*(\mathcal{I}^{\mathcal{F}}) = \{\{p_1 = l_1\}, \{p_1 = t\}, \{p_2 = l_1\}, \{p_2 = t\}\}$. This is due to the applicability of leaf-only operators $\mathrm{load}(l_1, p_i)$ for both packages.*

Since $\mathcal{I}^{\mathcal{F}}_*$ does not contain a goal state, we proceed by applying the only applicable global operator, $\mathrm{drive}(l_1, l_2)$, resulting in the unsaturated decoupled state $s^{\mathcal{D}}$. Here, the center variable is updated, while the reached leaf states remain unchanged, since they satisfy the preconditions of the operator and are not affected by it. The saturated decoupled state $s^{\mathcal{D}}_*$ matches the goal condition due to the unload *leaf-only operators, giving us a decoupled plan: $\langle \mathrm{drive}(l_1, l_2) \rangle$.*

We remark that a decoupled plan – a sequence of labels representing a path from an initial state to a goal state in the decoupled state-space – is not a plan in the original task because it considers only global operators and ignores the leaf-only operators. However, it is efficiently possible to construct a plan for the original task from the decoupled plan by scheduling leaf-only operators along the global ones [Gnad and Hoffmann, 2018].

3 Decoupled Search as a Task Transformation

Our task transformation approach is based on the FDR planning formalism [Helmert, 2009], which, in simple terms, extends the previously introduced SAS$^+$ formalism by a *background theory* in the form of a stratified logic program. Concretely, the FDR formalism has *two* sets of variables, the *primary variables* $\mathcal{V}$ as in SAS$^+$ and a set of *secondary* (or *derived*) variables $\mathcal{D}$. The secondary variables are binary, with

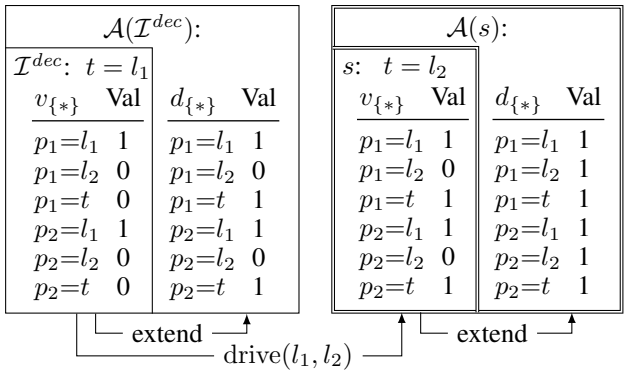

Figure 4: Illustration of the state space of the transformed task $\Pi^{dec}_{\mathcal{F}}$ (embodying decoupled search) needed to determine a decoupled plan for the running example.

a default value of 0, and in every state their value is exactly determined by the values of the primary variables. The values are obtained by performing a fixed-point computation over a set of *axiom* rules that derive whether a secondary variable becomes true (represented by the value 1). This computation is also called *extension* of the state. The conditions for applying an axiom can be over both primary and secondary variables, hence the axioms must be non-conflicting, or *stratifiable* [Apt *et al.*, 1988; Thiébaux *et al.*, 2005]. The FDR formalism is strictly more expressive than SAS$^+$ because it allows to represent planning problems and their solutions more compactly.

For our task transformation, we take an input SAS$^+$ task Π, compute a factoring $\mathcal{F} = \langle C, \mathcal{L} \rangle$, and transform it into a decoupled FDR task $\Pi^{dec}_{\mathcal{F}}$ in which the secondary variables and axioms encode the leaf semantics of decoupled search. An overview of the entire task transformation is shown in Figure 1. At a high level, this transformation ensures that each state in the transformed task corresponds exactly to a decoupled state in the original task, consisting of a single center state and a set of reached leaf states. Transitions between states are encoded by keeping and modifying the original global operators to maintain the decoupled state representation. Finally, the axioms describe a background theory accurately modeling the saturation of decoupled states after applying a global operator.

More concretely, we keep only the center variables C in $\mathcal{V}$ and add a new binary variable v_{s^L} for every leaf state s^L to indicate if that leaf state is reached. Similarly, there is a secondary variable $d_{s^L} \in \mathcal{D}$ for each leaf state, which is derived from the corresponding primary variable v_{s^L} or from an axiom that models a leaf-state transition. These axioms encode the leaf state spaces via the leaf-only operators, i. e., if a leaf state t^L can be reached from a leaf state s^L with a leaf-only operator o^L, then there exists an axiom for o^L that mimics this. The saturation of decoupled states is then achieved by the fixed-point computation over the axioms. The following example showcases the transformation on our example.[2]

Example 4. *Consider the running example with factoring*

[2]We omit some details of the exact modeling and refer to the full paper for more information.

$\mathcal{F} = \mathcal{F}_t$. *Figure 4 illustrates parts of the state space of* $\Pi^{dec}_{\mathcal{F}}$. *The primary variables include the center variable* t *and a variable* v_{s^L} *for each leaf state* s^L. *The secondary variables include a variable* d_{s^L} *for each leaf state. There are two global* drive *operators that move the truck between locations and also "copy" values from the secondary variables* d_{s^L} *to the primary variables* v_{s^L}, *preserving the reached leaf states. The transformed task further contains axioms which copy values from* v_{s^L} *to* d_{s^L} *variables, and the leaf-only operator axioms, representing* load *(*$d_{p_i=t} \leftarrow p_i = l_j \wedge t = l_j$*) and* unload *operators (*$d_{p_i=l_j} \leftarrow p_i = t \wedge t = l_j$*).*

Figure 4 shows the initial state $\mathcal{I}^{dec}$ *and its extension,* $\mathcal{A}(\mathcal{I}^{dec})$, *where the truck and both packages are at* l_1. *In* $\mathcal{A}(\mathcal{I}^{dec})$, *we can infer that the packages can be at* l_1 *or in the truck. After applying the only applicable operator, we find a goal state* s *that yields the plan* $\langle \text{drive}(l_1, l_2) \rangle$.

An important property of the transformation is that it *exactly* captures the behavior of decoupled search, i. e., any search algorithm X executed on the transformed task behaves exactly as the decoupled variant of X on the original task. Formally, the state space of the transformed task is isomorphic to the decoupled state space of the original task:

Theorem 1. *Let* $\Pi = \langle \mathcal{V}, \mathcal{I}, \mathcal{G}, \mathcal{O} \rangle$ *be a SAS*$^+$ *planning task and* $\mathcal{F}$ *be a factoring for* Π. *Then the FDR state space of* $\Pi^{dec}_{\mathcal{F}}$ *and the decoupled state space of* Π *are isomorphic.*

4 Experimental Evaluation

We implemented our decoupled task transformation in the Fast Downward 23.06 framework (FD) [Helmert, 2006]. Our experiments were conducted on all 2106 STRIPS instances from the satisficing sequential tracks of the International Planning Competitions 1998–2023. Our code and experimental data are available online [Speck and Gnad, 2024a].[3]

Setup. We evaluate our approach by performing search directly on the transformed task with two different configurations: lazy greedy best-first search (GBFS) with the h^{FF} heuristic [Hoffmann and Nebel, 2001] and a dual-queue open list with preferred operators (PO) [Richter and Helmert, 2009], and the first iteration of LAMA [Richter and Westphal, 2010]. We compare our decoupled task representation (dec) to the original SAS$^+$ encoding of FD (sas) and to the native decoupled-search implementation of Gnad and Hoffmann [2018] (gh). Furthermore, we include the Merge-And-Shrink task reformulation method proposed by Torralba and Sievers [2019] (ts), which to our knowledge is the only alternative technique that extensively restructures the state space. As factoring strategy for our transformation and native decoupled search, we pick the best configuration reported by Gnad *et al.* [2022] for satisficing planning.

Performance. Table 1 shows the total coverage results (number of instances solved) for all instances across multiple domains where the factoring process is successful. While our transformation-based approach is overall behind the native implementation (gh), it clearly outperforms the original encoding (sas), even if that uses LAMA. However, our approach beats gh in four domains. Compared to the Merge-And-Shrink reformulation (ts), either of the methods outperforms the other in some domains, but overall dec is ahead by 29 instances. We remark that both gh and ts would require a specialized adaptation of the landmark heuristic in LAMA, whereas our approach works out of the box. When using LAMA, our approach beats the baseline with SAS$^+$ encoding by 20 instances overall.

	GBFS(h^{FF}, PO)				LAMA	
#	sas	gh	ts	dec	sas	dec
1059	912	**980**	915	944	942	**962**

Table 1: Coverage (number of solved instances) of GBFS with h^{FF} and preferred operators, respectively LAMA, projected on the set of instances in which our factoring method is successful. Best coverage is highlighted in bold face.

5 Conclusion

We introduced a novel approach to performing a search reduction technique, in our case decoupled search, as a task transformation. By reformulating the input task we can exactly capture the behavior of decoupled search, which, like a native implementation, can achieve exponential savings in search effort. In practice, this dramatically simplifies the use of decoupled search by eliminating the need for specialized implementations. Instead, with our transformation, any planning technique can be used out-of-the-box in combination with decoupled search. We have demonstrated this advantage with the well-established LAMA planner, one of the most powerful classical planning systems. As future work, we plan to explore orthogonal planning techniques on our transformed tasks, such as symbolic search using binary decision diagrams, or planning as satisfiability, and want to investigate if other reduction techniques, such as symmetry breaking, can be implemented via task transformation.

Acknowledgements

David Speck 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). This work was partially supported by TAILOR, a project funded by the EU Horizon 2020 research and innovation programme under grant agreement no. 952215, and by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at the National Supercomputer Centre at Linköping University partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

[Apt *et al.*, 1988] Krzysztof R. Apt, Howard A. Blair, and Adrian Walker. Towards a theory of declarative knowl-

[3] https://github.com/speckdavid/decoupling-transformer

edge. In *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, 1988.

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

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[Fawzi *et al.*, 2022] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

[Gnad and Hoffmann, 2018] Daniel Gnad and Jörg Hoffmann. Star-topology decoupled state space search. *AIJ*, 257:24–60, 2018.

[Gnad *et al.*, 2022] Daniel Gnad, Álvaro Torralba, and Daniel Fišer. Beyond stars - generalized topologies for decoupled search. In *Proc. ICAPS 2022*, pages 110–118, 2022.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *JAIR*, 26:191–246, 2006.

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

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.

[Korf, 1997] Richard E. Korf. Finding optimal solutions to Rubik's Cube using pattern databases. In *Proc. AAAI 1997*, pages 700–705, 1997.

[McMillan, 1993] Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[Richter and Helmert, 2009] Silvia Richter and Malte Helmert. Preferred operators and deferred evaluation in satisficing planning. In *Proc. ICAPS 2009*, pages 273–280, 2009.

[Richter and Westphal, 2010] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR*, 39:127–177, 2010.

[Speck and Gnad, 2024a] David Speck and Daniel Gnad. Appendix, code, and experimental data of the ICAPS 2024 paper "Decoupled Search for the Masses: A Novel Task Transformation for Classical Planning". https://doi.org/10.5281/zenodo.10777533, 2024.

[Speck and Gnad, 2024b] David Speck and Daniel Gnad. Decoupled search for the masses: A novel task transformation for classical planning. In *Proc. ICAPS 2024*, pages 546–554, 2024.

[Speck *et al.*, 2023] David Speck, Paul Höft, Daniel Gnad, and Jendrik Seipp. Finding matrix multiplication algorithms with classical planning. In *Proc. ICAPS 2023*, pages 411–416, 2023.

[Thiébaux *et al.*, 2005] Sylvie Thiébaux, Jörg Hoffmann, and Bernhard Nebel. In defense of PDDL axioms. *AIJ*, 168(1–2):38–69, 2005.

[Torralba and Sievers, 2019] Álvaro Torralba and Silvan Sievers. Merge-and-shrink task reformulation for classical planning. In *Proc. IJCAI 2019*, pages 5644–5652, 2019.