ECAI 2025
I. Lynce et al. (Eds.)

doi:10.3233/FAIA251371

Learning of Lifted Macro-Events for Heuristic-Search Temporal Planning

Alessandro La Farciola[a,*], Alessandro Valentini[a] and Andrea Micheli[a]

[a]Fondazione Bruno Kessler, Trento, Italy

Abstract.
Learning domain knowledge from small training problems to improve planning performance on arbitrarily sized problems is a highly active research area. Many works explored the use of macro-actions to create "shortcuts" in the search space, at the cost of increasing the branching factor of the problem. In temporal planning, a recent technique proposes to equip a heuristic-search temporal planner with selected "macro-events": a "shortcut" mechanism similar to macro-actions but with state-dependent semantics.
In this paper, we generalize macro-events to a lifted representation, making them independent of specific problem objects. We devise a fully automated framework that, given a domain and a collection of small training problems, constructs and selects a suitable set of lifted macro-events. We define a learning pipeline that mixes the optimization of the statistical expectation on an abstraction of the problem with an empirical refinement of the selection on a validation set. We experimentally show that the proposed approach scales to complex problems, yielding substantial improvements over the baseline.

1 Introduction

Automated temporal planning is the problem of synthesizing a course of action to achieve a desired goal, given a formal description of the system to be controlled when time and temporal constraints are relevant [15]. Temporal planning has natural applications in domains such as robotics, logistics, and process automation. Scalability is still the key challenge for domain-independent temporal planners: real-world scenarios often remain beyond the reach of current systems.

One promising way to address this problem is to specialize (temporal) planners for a domain of interest [4]. Among the proposed methods to automatically achieve this specialization (which is mostly focused on the synthesis of search heuristics, e.g. [5]), learning small portions of plans that can be combined with basic actions to construct a complete solution is a simple yet effective approach, well studied in the context of classical planning. These "shortcuts" in the search space are usually called "macro-actions" and are sequences of actions that can be leveraged to reduce the distance to the goal. It is well-known that the selection of macro-actions is very critical and that their utility strongly depends on domains, problems, and planner algorithms [6, 13]. Adding macro-actions to a domain has a dual effect: it introduces shortcuts in the search space, possibly allowing the planner to find the goal with fewer steps, but it also increases the branching factor, potentially causing a blow-up in the search if guided in the wrong direction. Recently, La Farciola et al. proposed to learn and use "macro-events", which are ground structures similar to macro-actions for heuristic-search temporal planners [20]. Unlike macro-actions, macro-events cannot be encoded as new domain action schemata, but they are sequences of ground actions with a context-dependent semantic that can be exploited by a simple modification of any heuristic-search temporal planner. The major limiting factor of this approach is that it works at the grounded level, so it cannot generalize to problems with an arbitrary number of objects.

In this paper, we make three major contributions: (i) we generalize the (ground) macro-events technique of [20], defining a novel lifted formalization of the approach; (ii) we devise a fully automated, domain-independent selection procedure to select a suitable set of macro-events, given a domain and a set of training instances; and (iii) we experimentally demonstrate the effectiveness of our approach. We start by defining a lifted formalization of macro-events overcoming the inherent limitation of [20]: intuitively, lifted macro-events can be interpreted as generators of ground macro-events which the planner can then use. In this way, we extend the applicability of macro-events and provide a more compact representation that allows the learning of effective lifted macro-events from small training instances, capable of generalizing to larger problems. Then, we propose a learning pipeline for extracting and selecting a set of lifted macro-events given a temporal planning domain and a set of training instances. We start by sampling the training instances into two parts: the first part is used to construct a dataset of valid (non-necessarily optimal) plans, and the second (called "validation set") is used to empirically select a set of lifted macro-events for the domain. Our selection procedure is based on statistics and is divided into two main phases. After extracting a database of candidate lifted macro-events from the dataset of valid plans, we iterate over the validation set and select a subset of lifted macro-events for every problem using a local search technique to optimize the statistical expectation of the impact. The selection approach estimates —under certain simplifying assumptions— the number of states that would be expanded by a blind-search planner equipped with a set of lifted macro-events, and chooses the subset that minimizes this estimate. Some estimators adapt to the specific size of the problem, providing greater flexibility and accuracy in the selection process. Then, we solve each problem instance using a planner equipped with the selected set of lifted macro-events and collect the ones exploited by the planner to reach the goal. Finally, we conduct a selection of the most promising lifted macro-events based on their empirical impact on the validation set. Our experiments demonstrate that lifted macro-events generalize to complex problems yielding very good results in three different case studies.

* Corresponding Author. Email: alafarciola@fbk.eu

2 Related Work

In recent years, Generalization and Learning for Planning have seen increasing attention, especially due to the rapid advancement of Machine Learning (ML) techniques. Combining them with Automated Planning opens up promising challenges. The main objective is to learn domain knowledge from small training problems to scale up planning to problems of arbitrary size [4]. Several works focus on learning heuristics for search-based planners: for classical and numeric planning, some perform supervised learning [12], others explore Graph Neural Networks [5]. For temporal planning, Micheli and Valentini learn heuristics via Reinforcement Learning [22].

In this paper, we focus on learning a lifted representation of macro-events, a "shortcut" model similar to macro-actions. Macro-actions are well-known in classical planning: some authors focused on generating them during the planning process, such as in the case of Marvin [7], and YAHSP [27, 28]. However, most existing approaches aim at learning macro-actions off-line using training problems. Relevant examples are the MACRO-FF planner [1], which extracts macro-actions from either solution plans or static problem analysis, and the work by Newton et al., which employs a genetic algorithm to explore macro-actions [24]. A key challenge in using macro-actions is selecting the most useful ones without dramatically increasing the branching factor (sometimes referred to as the *utility problem*). In this direction, Dulac et al. extracts statistical information from valid plans based on a n-gram analysis to filter macro-actions [13]. Castellanos-Paez et al. use data mining techniques on frequent sequences to select macro-actions [3], and Chrpa and Vallati propose the method of Critical Section Macros to evaluate utility of macro-actions [6].

Very few works concern macro-actions for temporal planning. One critical difficulty in this context is the insufficiency of the usual sequential model for macro-actions for dealing with the temporal semantics of languages such as PDDL 2.1 [14] or ANML [25]: in fact, temporally expressive temporal planning [10] requires *concurrency*, hence a simple concatenation of durative actions is not expressive enough in many interesting domains. Some works try to encapsulate sequences of durative actions by constructing macro-operators [29, 17, 11]. In particular, De Bortoli et al. insist on guaranteeing, while using such temporal macro-operators, the sequential applicability of original actions avoiding unnecessary suppression of other concurrent actions [11]. Very recently, La Farciola et al. propose a different approach, introducing the concept of macro-events, which are sequences of actions with context-dependent semantics [20]. The basic idea is to represent a totally-ordered sequence of *events* by only indicating the ID of the action to progress. For example, a sequence $a; b; a; b$ could mean "start a, then start b, then end a, then end b" in a state where not a nor b are running, but the same macro-event could mean "end a, then start b, then start a new instance of a, then end b" in a state where a was started, but not ended, and b is not running. Unlike previous works, they do not construct macro-operators, but use such sequences on the fly during the search, discussing various approaches in which they can be used by a heuristic-search temporal planner. In every case, macro-events are exploited by the planner in addition to individual actions, preserving the original planner formal guarantees, such as completeness. The major limiting factor of the previous approach is the requirement of using a schematic naming of objects of the same type to make ground actions for one instance potentially applicable to others. What we present in this paper is a novel formalization that generalizes ground macro-events to a lifted representation. Our lifted macro-events are independent of the name chosen for objects and their groundings can be used by the planner on problems of arbitrary size. Furthermore, La Farciola et al. present a statistics-based procedure to approximate the number of search states expanded in the worst case by a blind-search planner equipped with a set of macro-events, under the assumption that all actions are applicable. They select the subset that minimizes this approximate value. In this paper, we extend this procedure into a complete and general framework for selecting lifted macro-events. Our selection method extracts candidates independently of the planner, and performs the selection by adapting to the problem size and the chosen planner.

3 Problem Formalization

In this section, we present the formalization of the problem we address in this paper. In particular, we adapt the ground formulation in [20] to a lifted representation.

Temporal Planning with ICE We start by formalizing the temporal planning problem that we tackle.

Definition 1. *A **term** t is either an object o or a variable v. An **atom** is an expression $p(t_1, \ldots, t_n)$ where p is a predicate with arity n and $t_1, \ldots, t_n$ are terms. In particular, we call it **ground atom** if all t_i are objects, **lifted atom** if at least one t_i is a variable.*

In our temporal language specification, conditions and effects can be declared to happen at any time within the duration of an action, and conditions can be durative, so they are associated with a time interval (this feature is called "Intermediate Conditions and Effects") [26].

Definition 2. *A **timing** τ is a syntactical expression that is either of the form* $\textsc{start} + k$ *or* $\textsc{end} - k$ *with* $k \in \mathbb{Q}_{\geq 0}$.

A timing is interpreted relatively to an action instance starting and ending times (by substituting START with the actual starting time and similarly END) and indicates a time instant during an action.

Definition 3. *An **effect** on atom $p(\vec{t})$ at relative time τ is a tuple $\langle \tau, p(\vec{t}) \rangle$ where τ is a timing. A **condition**[1] on atom $p(\vec{t})$ in the relative interval $[\tau_1, \tau_2]$ is a tuple $\langle [\tau_1, \tau_2], p(\vec{t}) \rangle$ where τ_i are timings. The atom $p(\vec{t})$ can be either a boolean or a numeric value.*

Definition 4. *A (durative) **action schema** $\mathtt{a}$ of arity n is a tuple $\langle d_{\mathtt{a}}^{min}, d_{\mathtt{a}}^{max}, C_{\mathtt{a}}, E_{\mathtt{a}}^{+}, E_{\mathtt{a}}^{-}, \vec{v}_{\mathtt{a}} \rangle$ where $d_{\mathtt{a}}^{min}$ and $d_{\mathtt{a}}^{max} \in \mathbb{Q}$ are minimal and maximal durations, $C_{\mathtt{a}}$ is the set of conditions, $E_{\mathtt{a}}^{+}$ and $E_{\mathtt{a}}^{-}$ are the sets of add and delete effects (with $E_{\mathtt{a}}^{+} \cap E_{\mathtt{a}}^{-} = \emptyset$), and $\vec{v}_{\mathtt{a}}$ is a vector of n distinct variables. Moreover, all the atoms included in the definition of $\mathtt{a}$ can only use the variables appearing in $\vec{v}_{\mathtt{a}}$.*

We define an instantiated action as an action $\mathtt{a}$ where all the parameters of each atom describing $\mathtt{a}$ are assigned to given terms.

Definition 5. *Given an action schema $\mathtt{a}$ of arity n and a vector of n terms $\vec{t}$, an (instantiated) **action** is a tuple $\langle d_{\mathtt{a}}^{min}, d_{\mathtt{a}}^{max}, C_{\mathtt{a}}, E_{\mathtt{a}}^{+}, E_{\mathtt{a}}^{-} \rangle$ where for any atom included in the definition of $\mathtt{a}$ each variable $\vec{v}_{\mathtt{a}}[i]$ is substituted with $\vec{t}[i]$. Moreover, we call it **ground action** if all t_i are objects (written $\mathtt{a}(\vec{o})$), **lifted action** if at least one t_i is a variable.*

In this paper, only lifted atoms where all terms are variables appear. Then, we can refer to a lifted action simply as $\mathtt{a}(\vec{v})$. We use different fonts $\mathtt{a}$ and a to emphasize lift vs ground level and, for simplicity, we omit the parameter in parenthesis $(\vec{o})$, $(\vec{v})$, unless necessary.

Finally, a planning domain is composed of a finite set of predicates and actions, while a planning problem is a finite set of objects, an initial state, and a goal to reach.

[1] We only formalize closed condition intervals; open and semi-open intervals are supported by our implementation.

Definition 6. *A **temporal planning domain** is a tuple $\langle P, A\rangle$ where P is a finite set of predicates, A is a finite set of (durative) action schemata. Moreover, every action schema in A contains only predicates included in P in its conditions and effects.*

Definition 7. *A **temporal planning problem** is a tuple $\langle O, I, G\rangle$ where O is a finite set of objects o_i; I and G are sets of ground atoms over predicates in P.*

Given a domain and a problem, we obtain all the ground actions instantiating every action schema with all the objects in O.

Definition 8. *A **temporal plan** π for a domain $\langle P, A\rangle$ and a problem $\langle O, I, G\rangle$ is a set of tuples $\{\langle a_1(\vec{o}_1), t_1, d_1\rangle, \ldots, \langle a_n(\vec{o}_n), t_n, d_n\rangle\}$, where each $a_i(\vec{o}_i)$ is a ground action, $t_i \in \mathbb{Q}_{\geq 0}$ is its start time, and $d_i \in \mathbb{Q}_{>0}$ is its duration.*

A temporal plan can be simulated by applying all the effects induced by its ground actions and their timings: each effect updates its fluent while others remain unchanged. The plan is valid if all action conditions hold during the simulation and the goal is achieved at the end.

Similarly to [20], we adopt the non-self-overlapping semantics, i.e. two instances of the same ground action are disallowed to overlap in time, making the problem PSPACE-complete [16] and allowing a much simpler representation of macro-events.

Macro Events for Heuristic-Search Temporal Planning. A common approach to solving temporal planning problems is heuristic search combined with a symbolic representation of time. First introduced in [8] and adopted by many planners (e.g., POPF [9], TAMER [26]), this approach decomposes ground actions into "events" and searches over their orderings, using symbolic timing constraints to ensure temporal feasibility.

Definition 9. *Given a ground action a, an **event** e is either:*
- *the starting (*$\textsc{Start}(a)$*) or ending (*$\textsc{End}(a)$*) of a;*
- *an effect x (*$\textsc{Eff}(x, a)$*) of a; or*
- *the start (*$\textsc{Start}(c, a)$*) or end (*$\textsc{End}(c, a)$*) of a condition c of a.*

We assume that we can refer to an event of a ground action (writing $action(e)$ for the ground action e belongs to) and that given a state s, we can check the applicability of the event in s, denoting with $e(s)$ the state resulting from applying e in s. We also assume that given a ground action a, its events are totally ordered[2] and, thanks to the non-self overlapping assumption, we can define the next event of a in s (written $next(a, s)$) as the start of a if a is not running in s, otherwise as the unique event in $events(a)$ after the last event of a in the path leading to s.

We now formalize what we mean by "macro-events", in particular by differentiating "lifted macro-events" and "ground macro-events". Furthermore, we describe how to exploit them in heuristic search temporal planning. We assume that a domain $\langle P, A\rangle$ and a problem $\langle O, I, G\rangle$ are given.

Definition 10. *A **lifted macro-event** (briefly **lifted macro**) of arity n is a tuple $\langle \mathtt{m}, \vec{v}\rangle$ (indicated with $\mathtt{m}(\vec{v})$), where $\mathtt{m}$ is a sequence of lifted actions $\mathtt{m} = \langle \mathtt{a}_1(\vec{v}_1), \ldots, \mathtt{a}_{|\mathtt{m}|}(\vec{v}_{|\mathtt{m}|})\rangle$, and $\vec{v}$ is a minimal vector of n distinct variables, in the sense that every variable in $\vec{v}_i$ is in $\vec{v}$ (for any i) and all variables in $\vec{v}$ are at least once in some $\vec{v}_i$. Moreover, we indicate with $|\mathtt{m}|$ the **length** of $\mathtt{m}$, and with $\mathrm{ar}(\mathtt{m})$ its **arity**.*

[2] If they are not, the planning problem can be compiled in an equivalent form where they are, at the cost of adding actions. Details can be found in [26].

Due to the previous definition, two different sets of variables define two different lifted macros on the same sequence of action schemata. Now, we formalize the concept of "ground macro-events".

Definition 11. *A **ground macro-event** (or more briefly **ground macro**) is a tuple $\langle m, \vec{o}\rangle$ (indicated by $m(\vec{o})$), where m is a sequence of ground actions $m = \langle a_1(\vec{o}_1), \ldots, a_{|m|}(\vec{o}_{|m|})\rangle$, and $\vec{o} \in O \times \cdots \times O$ is a minimal vector of objects, with the same meaning of the previous definition.*
*Given a ground macro-event m, we indicate with $m^{\leq i}$ the **ground prefix-macro-event**, i.e. the restriction of m to the first i elements of the sequence, with $m^{\geq i}$ the **ground suffix-macro-event**, i.e. the restriction of m to its tail starting from the i-th element, and with $m[i]$ the i-th element of m.*

The intuition behind ground macro-events is that, given a search state, one only needs to know which ground action has to be advanced. If an action is not started, advancing means starting it; if instead it is started, advancing means applying its next event. Formally, given a ground macro m and a state s, the successor state $m(s)$ is:

$$m(s) = \begin{cases} next(a, s)(s) & \text{if } m = \langle a\rangle \\ m^{\geq 2}(next(m[1], s)(s)) & \text{otherwise} \end{cases}$$

A lifted macro-event is basically a lift operator that can be interpreted as a generator of ground macro-events initialized on different objects of the problem. We define the following mapping relations.

Definition 12. *Given a lifted macro-event $\mathtt{m}(\vec{v})$ of arity n and a vector of n objects $\vec{o}$, we call **ground** the function such that $\mathtt{ground}(\mathtt{m}(\vec{v}), \vec{o})$ is the only ground macro-event obtained replacing every $\vec{v}[i]$ with $\vec{o}[i]$ (written $\mathtt{m}(\vec{o})$). Conversely, given a ground macro-event $m(\vec{o})$, we call **lift** the relation such that $\mathtt{lift}(m(\vec{o}))$ is its preimage under $\mathtt{ground}$: $\{\mathtt{m}(\vec{v}) \mid \mathtt{ground}(\mathtt{m}(\vec{v}), \vec{o}) = m(\vec{o})\}$.*

Definition 13. *Given a problem $\mathcal{P} = \langle O, I, G\rangle$ and a lifted macro-event $\mathtt{m}$, we define $\mathtt{G}(\mathtt{m}, \mathcal{P}) = \{\mathtt{ground}(\mathtt{m}, \vec{o}) \mid \vec{o} \in O^{\mathrm{ar}(\mathtt{m})}\}$ the set of all possible **groundings** of $\mathtt{m}$ in $\mathcal{P}$.*

We also define a *containment* relation between lifted macros with the same sequence of action schemata.

Definition 14. *Given a problem $\mathcal{P}$ and two lifted macro-events $\mathtt{m}_1$ and $\mathtt{m}_2$ with the same sequence of action schemata, we say that $\mathtt{m}_1$ **includes** $\mathtt{m}_2$ (written $\mathtt{m}_2 \subseteq \mathtt{m}_1$) if $\mathtt{G}(\mathtt{m}_2, \mathcal{P}) \subseteq \mathtt{G}(\mathtt{m}_1, \mathcal{P})$.*

Let us make an example: consider, for instance, two action schemata $move(x, y)$ and $pick(z)$ modeling an agent moving from a location x to a location y and picking a package at location z, respectively. It is possible to construct different lifted macros with the same sequence of action schemata $\langle move, pick\rangle$. One choice could be $\mathtt{m}_1(v_1, v_2, v_3) = \langle move(v_1, v_2), pick(v_3)\rangle$ with arity 3. In this case, a possible ground macro generated from m_1 is $\langle move(l_1, l_2), pick(l_3)\rangle$, where l_1, l_2, l_3 are distinct objects of the planning problem. A different possibility could be $\mathtt{m}_2(v_1, v_2) = \langle move(v_1, v_2), pick(v_2)\rangle$ with arity 2. In this case, the corresponding generated ground macro is $\langle move(l_1, l_2), pick(l_2)\rangle$ which restricts the action *pick* to occur in the same location where *move* ended. It is clear that $\mathtt{m}_1$ includes $\mathtt{m}_2$, but its generality increases the branching factor of the search space. The second has fewer groundings, but it is potentially more effective for the plan search.

We remark that what we call here "ground macro-events" is the same as what is defined in [20] simply as "macro-events". Thus, we

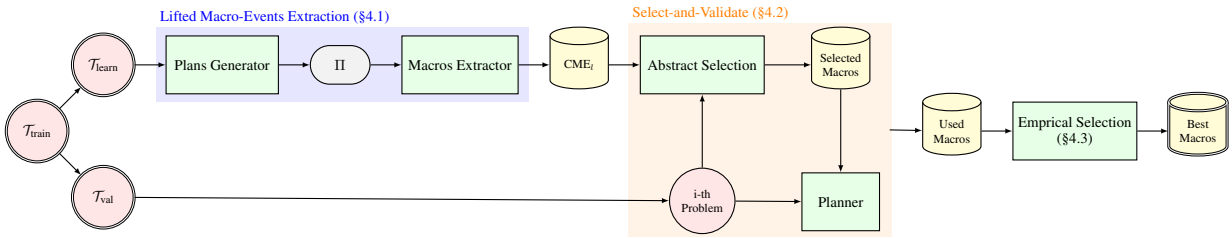

Figure 1. Overall Lifted Macro-Events selection framework indicating the three major phases discussed in Section 4.

borrow at the ground level all the definitions of the applicability and planning approaches described by La Farciola et al., as well as the formalization of the extraction of events from a given plan.

Definition 15. *A ground macro-event m is **fully-applicable (FA)** in s if $next(a_1, s)$ is applicable in s and $next(a_i, m^{\leq i-1}(s))$ is applicable in $m^{\leq i-1}(s)$ for all $i \in \{2, \ldots, |m|\}$. It is **partially-applicable (PA)** in s if $m^{\leq 2}$ is fully-applicable in s, and its order is the maximum length of its ground prefix-macro-events fully-applicable in s.*

Intuitively, a ground macro-event is fully-applicable if, starting from state s, it is completely executable, while it is partially-applicable if it is executable up to a certain point.

Definition 16. *We define a search with ground macro-events to be **with intermediate nodes** $(+)$ if when applying a ground macro-event it adds each intermediate state in the search queue. On the other hand, we define it to be **without intermediate nodes** $(-)$ if it only adds the last state.*

We assume a planner that performs a search at the ground level (so not a lifted planner); therefore, given a problem and a set of lifted macro-events, we construct all their groundings and execute the planner with these ground macro-events. The four variations of the search algorithm defined in [20] are supported ($\aleph \in \{FA, PA\} \times \{\pm\}$): in $+$ cases, the search enqueues all intermediate states expanded while evaluating a ground macro; otherwise, only the final state is added. Additionally, in PA, states are enqueued even if the ground macro is not completely applicable, up to the point where execution halts.

Finally, given a temporal plan π, we call its **timed set of events** (written $te(\pi)$) the set of all tuples of times and events appearing in the simulation of the plan. For example, if $\langle a, t, d\rangle \in \pi$, then $\langle t, \textsc{Start}(a)\rangle$ and $\langle t + d, \textsc{End}(a)\rangle \in te(\pi)$ (and also all the effects and intermediate conditions of a at the appropriate times are in $te(\pi)$). Moreover, $ev(\pi) = \langle e_i \mid e_i \in te(\pi)\rangle$ is defined as the sorted **sequence of events** originating from the plan, and the **(ground) plan-macro-event** is $\mu(\pi) = \langle action(e) \mid e \in ev(\pi)\rangle$.

4 Learning Lifted Macro-Events

In this section, we assume that a domain, a planning approach $\aleph$, and a set of training instances $\mathcal{T}_{\text{train}}$ are given and we focus on the problem of extracting and selecting a suitable set of lifted macro-events to be used for planning on any instance of the domain. Figure 1 illustrates the complete proposed selection framework. We start by separating the training problems into two subsets by random sampling: the learning set $\mathcal{T}_{\text{learn}}$ and the validation set $\mathcal{T}_{\text{val}}$. Solutions of instances in $\mathcal{T}_{\text{learn}}$ are used to collect a database of valid plan-macro-events Π, then we extract candidates lifted macros from sub-strings inside Π, counting their frequency. Furthermore, for every instance in $\mathcal{T}_{\text{val}}$, we perform a selection procedure optimizing the abstract impact of lifted macros and test the planner with the selected ones to collect empirical data. Using such data, we perform a final selection of the best promising lifted macro-events to use on the domain. In the following, we describe and formalize these three phases.

4.1 Lifted Macro-Events Extraction

Given the learning set of instances $\mathcal{T}_{\text{learn}}$, we generate the database of valid plan-macro-events Π. This set is composed of solutions to the problems in $\mathcal{T}_{\text{learn}}$. Similarly to [20], in our experiments we used a reinforcement learning approach and record the solutions found during training to avoid bias due to a specific planner or heuristic. We proceed by extracting all ground macro-events from sub-strings of plan-macro-events inside Π: we use the ℓ_{max} hyperparameter indicating the maximum length of ground macro-events. We call the **utility of** m **in** $\mu(\pi)$ (written u^{π}_{m}) the number of occurrences of a ground macro m in the plan $\mu(\pi)$. The **utility of** m **in** $\mu(\pi)$ **relative to** M (written $u^{\pi}_{m}(M)$) is the number of occurrences of m in $\mu(\pi)$ in which the ground macro is the longest in the set M that is applicable. Finally, the Candidates ground Macro-Events are the sequences with positive utility: $\text{CME}_g = \{m \mid |m| \leq \ell_{max} \wedge \sum_{\mu(\pi)\in\Pi} u^{\pi}_{m} > 0\}$.

From the database CME$_g$, we extract the database of Candidates lifted Macro-Events (CME$_l$), where the statistical information of ground macros is unified according to their corresponding lifted macros. In particular, from every ground macro m we compute the corresponding set $\texttt{lift}(m)$ and we extract the lifted macro that minimizes the arity, i.e. the minimal one according to the *containment* relation of Definition 14: $\text{CME}_l = \{\mathtt{m} \mid \mathtt{m} \in \texttt{lift}(m) \wedge m \in \text{CME}_g \wedge \forall \tilde{\mathtt{m}} \in \texttt{lift}(m).\ \mathtt{m} \subseteq \tilde{\mathtt{m}}\}$. Considering the same example of the previous section, if $m = \langle move(l_1, l_2), pick(l_2)\rangle \in \text{CME}_g$, then $\mathtt{m}_1(v_1, v_2, v_3)$ and $\mathtt{m}_2(v_1, v_2) \in \texttt{lift}(m)$, but only $\mathtt{m}_2 \in \text{CME}_l$.

4.2 Select-and-Validate for Lifted Macro-Events

After the set of candidate lifted macro-events CME$_l$ is generated, we proceed with the first round of selection. We perform a "select-and-validate" procedure for every instance in the validation set, that is, given an instance and the database CME$_l$, we first solve an optimization problem to identify the optimal subset of lifted macro-events in CME$_l$ to be used by the planner on that problem. Then, we execute the planner equipped with the identified lifted macros on the specific problem instance. For each selected lifted macro, we record in the "Used Macros" dataset (see Figure 1) the number of times it was used by the planner to construct a solution plan.

We first define the theoretical objective function (called "Abstract Impact"); then we discuss how to estimate it using statistical information on Π presenting the optimization problem to identify the optimal subset of lifted macros; finally, we describe the validation phase.

Abstract Impact. First, we want to characterize the number of states a heuristic-search algorithm would expand given a set of lifted macros $\mathcal{M}$ on a specific problem. We use this metric to select, across all $\mathcal{M} \subseteq \text{CME}_l$, the one that is expected to have the highest *impact*.

Definition 17. *Given a problem $\mathcal{P}$, a plan π, a (possibly empty) set of ground macro-events M, and a planning approach $\aleph$. We define:*

- *$ES(M, \aleph, \mathcal{P}, \pi)$ as the number of **expanded states** by the planning approach $\aleph$ to find $\mu(\pi)$;*
- *the **ground (abstract) impact** $I(M, \aleph, \mathcal{P}, \pi)$ of M as the gain in terms of expanded states $ES(\emptyset, \text{no macros}, \mathcal{P}, \pi) - ES(M, \aleph, \mathcal{P}, \pi)$;*
- *the **lifted (abstract) impact** of $\mathcal{M}$ as $\mathcal{I}(\mathcal{M}, \aleph, \mathcal{P}, \pi) := I(\mathtt{G}(\mathcal{M}, \mathcal{P}), \aleph, \mathcal{P}, \pi)$, where $\mathtt{G}(\mathcal{M}, \mathcal{P}) := \bigcup_{m \in \mathcal{M}} \mathtt{G}(m, \mathcal{P})$ is the set of all possible groundings of lifted macros in $\mathcal{M}$.*

Basically, we are saying that the impact of a set of lifted macros is the ground impact of the set of all their possible groundings. We adopt the same abstraction from [20]: every action is assumed to be applicable and a blind-search planner is assumed. Of course, this is not how a real heuristic search planner works, but it gives an effective metric for the sake of selection. Under these assumptions, the search space is a complete tree and the number of expanded states coincides with its total number of nodes. We use the results on ground impact introduced by La Farciola et al. to obtain the lifted impact when using a set of lifted macro-events $\mathcal{M}$ and a planning approach $\aleph$. Considering the search tree that uses the set of groundings of $\mathcal{M}$, we indicate by $A_{\mathcal{M}}$ its branching factor and by $D_{\mathcal{M}}$ its depth. Then, the total number of expanded states $\text{ES}(\mathtt{G}(\mathcal{M}, \mathcal{P}), \aleph, \mathcal{P}, \pi)$ is equal to $\sum_{j=0}^{D_{\mathcal{M}}} A_{\mathcal{M}}^j = \frac{(A_{\mathcal{M}})^{D_{\mathcal{M}}+1}-1}{A_{\mathcal{M}}-1}$.
The following table adapts the values for $A_{\mathcal{M}}$ and $D_{\mathcal{M}}$ computed in [20] for the four planning approaches considered, where $N(\mathcal{P})$ is the number of ground actions in problem $\mathcal{P}$ (briefly, *ground size* of $\mathcal{P}$), L^π is the length of $\mu(\pi)$, $\hat{\mathtt{G}}(\mathcal{M}, \mathcal{P}) := \bigcup_{m \in \mathtt{G}(\mathcal{M}, \mathcal{P})} \{m^{\leq j} \mid 2 \leq j \leq |m|\}$ is the union of all sub-sequences of ground macros in $\mathtt{G}(\mathcal{M}, \mathcal{P})$, and $\tilde{\mathtt{G}}(\mathcal{M}, \mathcal{P}) := \bigsqcup_{m \in \mathtt{G}(\mathcal{M}, \mathcal{P})} \{m^{\leq j} \mid 2 \leq j \leq |m|\}$ is its disjoint union (i.e., equal elements can appear multiple times).

	$A_{\mathcal{M}}$	$D_{\mathcal{M}}$
no macros	$N(P)$	L^π
FA-, PA-	$N(P) + \lvert\mathtt{G}(\mathcal{M}, \mathcal{P})\rvert$	$L^\pi - \sum_{m \in \mathtt{G}(\mathcal{M}, \mathcal{P})} u_m^\pi(\mathtt{G}(\mathcal{M}, \mathcal{P}))(\lvert m\rvert - 1)$
FA+, PA+	$N(P) + \lvert\tilde{\mathtt{G}}(\mathcal{M}, \mathcal{P})\rvert$	$L^\pi - \sum_{\hat{m} \in \hat{\mathtt{G}}(\mathcal{M}, \mathcal{P})} u_{\hat{m}}^\pi(\hat{\mathtt{G}}(\mathcal{M}, \mathcal{P}))(\lvert \hat{m}\rvert - 1)$

Estimation of the abstract impact. In practice, we cannot directly use the formulas in the previous table because they depend on a specific plan that is unknown when a new problem is given. Therefore, we now focus on how to empirically evaluate the formulas in the table to have an estimation of the abstract impact given a problem instance. Similarly to [20], we disregard the relative utility for every ground macro $m \in \text{CME}_g$, using the non-relative utility instead: $u_m^\pi(\mathtt{G}(\mathcal{M}, \mathcal{P})) \equiv u_m^\pi$. Moreover, we replace the utilities and the length of the plan with the empirical averages computed over the database of valid plans (Π):

- $\bar{u}_m = \frac{1}{|\Pi|} \sum_{\pi \in \Pi} u_m^\pi$, for any $m \in \text{CME}_g$.
- $\bar{L} = \frac{1}{|\Pi|} \sum_{\pi \in \Pi} L^\pi$;

As a result, the lifted impact becomes independent of any specific π and we refer to the estimated lifted impact as $\bar{\mathcal{I}}(\mathcal{M}, \aleph, \mathcal{P})$.

Algorithm 1 summarizes the entire *abstract selection* method. The EstimateImpact procedure computes the estimated lifted impact for a given set of lifted macros $\mathcal{M}$, a problem $\mathcal{P}$ and a planning approach $\aleph$. We start by computing the ground size of the given problem and the expected number of expanded states for the case with no macro-events ($ES(\emptyset)$). Then, we compute the set of groundings *GroundSet*, including all their prefixes in the + cases (lines 5-6). At this point, we can compute $A_{\mathcal{M}}$ and $D_{\mathcal{M}}$. In EstimateImpact, we assume that $\mathcal{M}$ does not contain multiple lifted macros with the same sequence of action schemata: this implies that the set of groundings of $\mathcal{M}$ is the disjoint union of the groundings of every lifted macro: $\mathtt{G}(\mathcal{M}, \mathcal{P}) = \bigsqcup_{m \in \mathcal{M}} \mathtt{G}(m, \mathcal{P})$. Hence, we can rewrite the previous formulas for branching factor and depth as follows.

Algorithm 1 Abstract Selection of Lifted Macro-Events

```
1:  procedure EstimateImpact(M, P, ℵ)
2:      N(P) ← GetGroundSize(P)
3:      ES(∅) ← ExpandedStates(N(P), L̄)
4:      GroundSet ← G(M, P)
5:      if ℵ ∈ {FA+, PA+} then
6:          GroundSet ← {m^{≤i} | m ∈ GroundSet, 2 ≤ i ≤ |m|}
7:      A_M ← BranchingFactor(GroundSet, N(P), ℵ)
8:      ℓ ← max_{m∈M}(|m|)
9:      D_M ← Max(Depth(GroundSet, L̄, ℵ), L̄/ℓ)
10:     return ES(∅) − ExpandedStates(A_M, D_M)

11: procedure AbstractSelection(CME_l, P, ℵ)
12:     SingleImpact ← ∅
13:     for all m ∈ CME_l do
14:         SingleImpact(m) ← EstimateImpact({m}, P, ℵ)
15:     FilterMacrosWithSameSchemata(CME_l, SingleImpact)
16:     return MetaheuristicSolve(M*(ℵ, P, CME_l))
```

	$A_{\mathcal{M}}$	$D_{\mathcal{M}}$
no macros	$N(P)$	$\bar{L}$
FA-, PA-	$N(P) + \sum_{m \in \mathcal{M}} \lvert\mathtt{G}(m, \mathcal{P})\rvert$	$\bar{L} - \sum_{m \in \mathcal{M}} \sum_{m \in \mathtt{G}(m, \mathcal{P})} \text{U}_m$
FA+, PA+	$N(P) + \sum_{m \in \mathcal{M}} \sum_{2 \leq i \leq \lvert m\rvert} \lvert\mathtt{G}(m^{\leq i}, \mathcal{P})\rvert$	$\bar{L} - \sum_{\hat{m} \in \hat{\mathtt{G}}(\mathcal{M}, \mathcal{P})} \text{U}_{\hat{m}}$

In the table, U_m estimates $u_m^\pi(\mathtt{G}(\mathcal{M}, \mathcal{P}))(|m| - 1)$ for a ground macro m, which can be carried out as proposed in [20]: $\text{U}_m = \bar{u}_m(|m| - 1)$ for the FA cases and $\text{U}_m = \sum_{i=2}^{|m|} \lambda_i \left(\bar{u}_{m^{\leq i}}(|m^{\leq i}| - 1)\right)$, where $\lambda_i := \frac{\bar{u}_{m^{\leq i}} - \bar{u}_{m^{\leq i+1}}}{\bar{u}_{m^{\leq 2}}}$ if $i \leq |m-1|$ and $\frac{\bar{u}_m}{\bar{u}_{m^{\leq 2}}}$ if $i = |m|$ for PA cases. Moreover, to avoid severe underestimation due to the assumption $u_m^\pi(\mathtt{G}(\mathcal{M}, \mathcal{P})) \equiv u_m^\pi$, the depth $D_{\mathcal{M}}$ is the maximum between the value obtained from the table and the minimum depth reachable with macros in $\mathcal{M}$ (line 9).

The AbstractSelection procedure builds on top of EstimateImpact and starts by computing a map of the impact of each lifted macro in CME_l in isolation (lines 12-14). We observe that CME_l might contain multiple lifted macros with the same action schemata: this is possible even if we chose only the minimal lifted macro (wrt *containment* relation) during the extraction phase. Continuing with the example above, if both $m_1 = \langle move(l_1, l_2), pick(l_3)\rangle$ and $m_2 = \langle move(l_1, l_2), pick(l_2)\rangle$ are in CME_g, both $\mathtt{m}_1(v_1, v_2, v_3)$ and $\mathtt{m}_2(v_1, v_2)$ are in CME_l. We filter CME_l selecting, for every subset of lifted macros with the same actions schemata, the one with the highest single impact (line 15). Finally, we formalize our objective as a maximization problem over the estimated lifted impact across all possible subsets of lifted macros.

Definition 18. *The **optimal set of lifted macro-events** $\mathcal{M}^*(\aleph, \mathcal{P}, \text{CME}_l)$ for a problem $\mathcal{P}$, and planning approach $\aleph$ is the subset of CME_l with the highest estimated impact:* $\arg\max_{\mathcal{M} \subseteq \text{CME}_l} \bar{\mathcal{I}}(\mathcal{M}, \aleph, \mathcal{P})$.

Solving this optimization problem involves dealing with a trade-off: a bigger set of lifted macros, with groundings that have great utilities, contributes to reducing the depth of the search tree and consequently the number of expanded states. However, especially for prob-

lems with many objects, bigger sets of lifted macros would generate a larger number of groundings, thus increasing the branching factor and the number of expanded states. Finally, to solve the optimization problem at line 16, we resort to a metaheuristic approach.

Validation Phase. So far, we have presented a general method for selecting a promising set of lifted macro-events given a problem, that can help the planner to find a solution. This procedure is not tied to any specific planner, making the extracted and selected sequences independent of the planning algorithm or any particular heuristic. As noted in the introduction, it is well-known that different planners may benefit from different macro-actions. For this reason, we introduce a planning validation phase. Some variables in the formulas derived above —such as the number of ground actions or the number of groundings of one or more lifted macros— are highly dependent on the size of the problem. Therefore, performing the selection on a specific instance allows the algorithm to identify the most suitable lifted macros based on the number of objects of each type available in that instance. The main objective of the validation phase is to determine which of the previously selected macro-events have a concrete impact on the planner's performance. In particular, when the planner reaches a goal, we can trace back the sequence of states up to the initial state, recording which of them were added to the queue as a result of applying a macro-event. We gather statistics on the frequency with which each lifted macro *contributes* to synthesize solutions.

Definition 19. *Given a problem $\mathcal{P}$ and a planning approach $\aleph$, we say that a ground macro-event m **contributes** to the plan synthesis if it generates at least one state in the valid trace for $\mathcal{P}$ found by $\aleph$.*

We then collect, for every planning approach $\aleph$, the set of *used macro-events* UME, defined as the lifted macros that have at least one grounding contributing to the plan synthesis of any $\mathcal{P} \in \mathcal{T}_{\text{val}}$.

4.3 Empirical selection

After the validation phase, the size of used macros can vary depending on the diversity and effectiveness of the abstract selection performed on each problem. When different instances yield different macros (each possibly used only once by the planner), the corresponding UME may be redundant, potentially including low-impact macros. Therefore, we introduce a second selection phase that refines the set of used macros, exploiting planner-specific empirical statistics. We define two metrics that can be used to further filter UME.

Definition 20. *Given the set of problems $\mathcal{T}_{val}$, a planning approach $\aleph$, and a lifted macro $m \in$ UME, we define the **frequency** $\text{freq}(m)$ as the number of occurrences of groundings of m contributing to the plan synthesis of problems in $\mathcal{T}_{val}$ with planning approach $\aleph$. We define the **support** $\text{supp}(m)$ as the number of problems $\mathcal{P} \in \mathcal{T}_{val}$ in which any grounding of m contributes to the plan synthesis.*

Definition 21. *Given the set of problems $\mathcal{T}_{val}$, a planning approach $\aleph$, and a set of lifted macro $\mathcal{M} \subseteq$ UME, we define **cumulative frequency percentage** and **cumulative support percentage** as:*
- $\text{cfp}(\mathcal{M}) := \frac{\sum_{m \in \mathcal{M}} \text{freq}(m)}{\sum_{m \in UME} \text{freq}(m)}$;
- $\text{csp}(\mathcal{M}) := \frac{\sum_{m \in \mathcal{M}} \text{supp}(m)}{|\mathcal{T}_{val}|}$.

For each approach, we choose a threshold $\vartheta \leq 1$ as a hyperparameter, and we consider subsets of the following two categories:
- $\mathcal{M}_1$ s.t. $\text{cfp}(\mathcal{M}_1) \geq \vartheta \ \wedge \ \forall \tilde{m} \in \mathcal{M}_1 . \text{cfp}(\mathcal{M}_1 \setminus \{\tilde{m}\}) < \vartheta$;
- $\mathcal{M}_2$ s.t. $\text{csp}(\mathcal{M}_2) \geq \vartheta \ \wedge \ \forall \tilde{m} \in \mathcal{M}_2 . \text{csp}(\mathcal{M}_2 \setminus \{\tilde{m}\}) < \vartheta$.

$\mathcal{M}_1$ and $\mathcal{M}_2$ are minimal subsets of UME that cover at least $\vartheta \times 100\%$ of the total frequencies and of the size of $\mathcal{T}_{\text{val}}$, respectively. For each category, we take the subset with the minimal cardinality and the highest value of cfp and csp, respectively, and we select the lifted macros that belong to at least one of them. If $\vartheta = 1$, it selects all used macros; if $\vartheta = 0$ no macros are selected. Pseudo-code summarizing the empirical selection can be found in the appendix [19].

In conclusion, in the PA+ case, the presence of sub-macros is redundant: if a macro such as $\langle a, b, c, d \rangle$ is included in the final set, all its sub-macros ($\langle a, b \rangle$, $\langle a, b, c \rangle$) can be safely removed.

5 Experimental Evaluation

For our experimental evaluation, we implemented a heuristic-search temporal planner supporting lifted macro-events in Python using the Unified Planning library [23] as modeling framework. Our planner relies on TAMER for the definition of the search space and optionally takes in input a set of lifted macro-events and the planning approach to follow. Then, the generated ground macro-events are used by the planner according to the algorithm presented in [20], with a technical improvement. In the search space, macros can lead to the expansion of multiple states with the same path (i.e., sequence of events from the initial state), even if they are conceptually identical. To prevent redundant expansions, we cache each visited state during the search and avoid exploring the same sub-tree multiple times.

We consider two benchmark domains used in [2, 20, 21, 22, 26], i.e. "MAJSP" and "Kitting", and the "Matchcellar" IPC domain. All of them are temporally expressive [10]. MAJSP consists of scheduling a fleet of agents to transport items between machines. Kitting requires robots to gather components from various locations to assemble a kit and deliver it in sync with a human operator. In Matchcellar, a set of fuses has to be mended while light is provided by a match to be lit. Our testing instances for Kitting and MAJSP are larger (in terms of number of objects) than the ones used in [20, 21, 22, 26].

To measure the effectiveness of our framework, we perform a 4-fold splitting of the training set: for each domain, we randomly partitioned such set into 4 equally sized subsamples. In turn, we use 3 subsamples (75% of training instances) as learning data to generate the dataset of plan-macro-events Π and to extract our database of candidate lifted macro-events. In particular, we employed the Reinforcement Learning (RL) approach described in [22] recording the two shortest valid traces for every learning instance to construct Π. Then, the remaining subsample (25% of training instances) is used in the validation phase. For every problem instance, we performed the *abstract selection* of lifted macro-events approximating the global optimum of the optimization problem defined in Section 4 through the simulated annealing technique [18].[3] The entire selection framework was implemented in Python. Then, for every planning approach, we performed the *empirical selection* with different choices of the threshold $\vartheta \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. We evaluated our planner equipped with the final selected lifted macros on the testing set against the baseline TAMER without macros for two different symbolic heuristics: h_{add} and h_{ff}. Similarly to [20], we chose as maximum length of candidates macro-events $\ell_{max} = 5$, and we set $w = 0.8$ for the A^* algorithm in the planning procedure. We run our experiments on a server with 4 AMD EPYC 7413 processors and 528GB of RAM, we allocated 4 cores for each planning run with a timeout of 600s and a memory limit of 30GB (the maximum memory used in the experiments was 8.2GB). Benchmarks, code, and all the plots are available in the additional material of this paper [19].

The average (and maximum) ground size (i.e. the number of ground actions) for problems in the training set is 21.5 (42) for MA-

[3] We used the *simanneal* library: https://github.com/perrygeo/simanneal

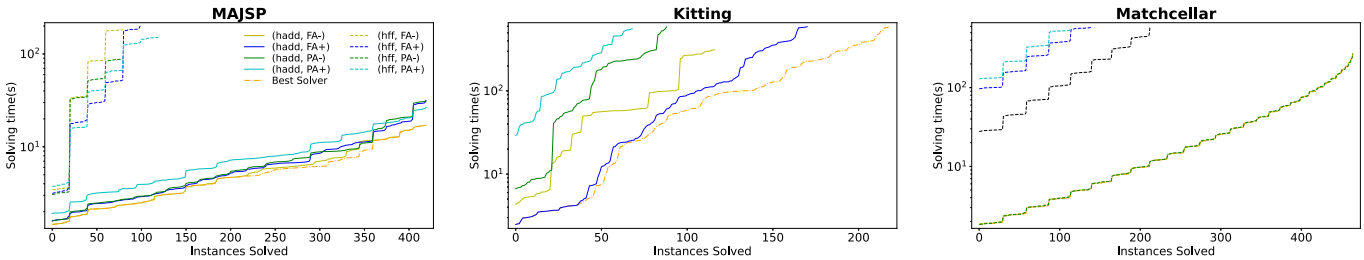

Figure 2. Cactus plots: for each domain, we report execution time as a function of the cumulative number of problems solved (sorted by execution time). The hypothetical best solver is also included, defined by taking the lowest execution time achieved for each instance across all techniques.

Domain	Testing Size	No Macros		Using Macro Events							
		h_{add}	h_{ff}	h_{add}				h_{ff}			
				FA-	FA+	PA-	PA+	FA-	FA+	PA-	PA+
MAJSP	420	0	0	**420**	**420**	**420**	**420**	80	100	80	120
Kitting	324	4	0	117	**171**	89	69	0	0	0	0
Matchcellar	465	0	214	0	0	0	0	**465**	140	**465**	114

Table 1. Coverage table: for each domain each heuristic and each planning approach, we report the result obtained with the best threshold.

JSP, 25.8 (35) for Kitting, and 10.3 (20) for Matchcellar; instead, for the testing set it is 144 (276) for MAJSP, 65 (92) for Kitting, and 41.3 (80) for Matchcellar. The average size pair of the extracted $(\text{CME}_g, \text{CME}_l)$ across the 4-fold split is $(5387, 1038)$ for MAJSP, $(3256, 544)$ for Kitting, and $(5611, 16)$ for Matchcellar. Table 1 shows the coverage results. It reports for each domain, heuristic, and planning approach the highest number of solved instances for the best-performing threshold ϑ. Approaches using macro-events significantly overcome the baseline without macros. What stands out most from these results is the difference in performance between the two heuristics. For MAJSP and Kitting, h_{add} outperforms h_{ff}, while for Matchcellar, the opposite holds. This contrast can be dramatic: in some cases, changes from solving no instances to saturating the test set. This highlights that when a heuristic recognizes the "shortcuts" potential provided by macro-events, the planner's performance improves significantly (including in terms of solving time, see Figure 2). On one hand, this confirms the importance of macros selection; on the other, it reveals a promising research opportunity. In line with the Learning for Planning direction discussed in the related work, jointly learning macro-events and a heuristic that has been exposed to them during training appears to be a highly promising idea.

Table 2 exhibits a sensitivity analysis of the parameter ϑ. For Matchcellar, a single lifted macro can capture highly informative domain knowledge on its own, and generates a large number of groundings because of symmetries in the domain. As a result, the + cases are the only ones that degrade performance compared to the baseline. This is because in those cases the mentioned phenomenon is amplified by two factors: the selected lifted macro is longer, and exploring intermediate states requires more computational effort. While this aspect represents a limitation for Matchcellar, it becomes an advantage in the other domains, where the best results are achieved by selecting a larger number of macros (see FA+ and PA- of Kitting-h_{add}) and by using intermediate states. Figure 2 shows the time performance of the different techniques. For MAJSP and Matchcellar, the techniques that saturate the test set perform similarly, with a modest advantage for FA- with h_{add} in MAJSP. In the Kitting domain, FA+ emerges as the most effective approach, although the performance of the best solver reveals a slight complementarity with other techniques.

MAJSP - h_{add} Ground Size : 144 (276)

ϑ	FA-		FA+		PA-		PA+	
0.5	330	2 (92)	0	5 (205)	0	3 (112)	**420**	4 (205)
0.6	330	3 (138)	0	8 (343)	420	4 (158)	420	5 (459)
0.7	330	5 (231)	0	10 (450)	420	6 (413)	420	5 (459)
0.8	330	6 (277)	**420**	12 (542)	420	10 (625)	420	5 (459)
0.9	**420**	10 (462)	420	20 (1305)	**420**	16 (1227)	420	7 (1051)
1	420	23 (1342)	420	32 (2073)	420	29 (3258)	420	15 (2156)

MAJSP - h_{ff} Ground Size : 144 (272)

ϑ	FA-		FA+		PA-		PA+	
0.5	20	4 (158)	0	4 (158)	0	4 (320)	0	4 (158)
0.6	20	5 (205)	0	6 (251)	20	6 (426)	0	4 (205)
0.7	20	7 (297)	0	8 (343)	40	8 (473)	0	5 (251)
0.8	20	9 (390)	80	11 (482)	80	11 (681)	100	7 (681)
0.9	**80**	12 (529)	80	17 (787)	**80**	16 (1074)	100	9 (1412)
1	80	24 (1353)	**100**	38 (3583)	80	38 (3828)	**120**	22 (3881)

Kitting - h_{add} Ground Size : 65 (92)

ϑ	FA-		FA+		PA-		PA+	
0.5	29	1 (2)	29	1 (2)	29	1 (2)	29	1 (2)
0.6	29	1 (2)	80	2 (9)	29	1 (2)	29	1 (2)
0.7	29	1 (2)	64	3 (23)	21	2 (4)	29	1 (2)
0.8	29	1 (2)	64	3 (23)	21	2 (4)	24	1 (2)
0.9	65	3 (23)	**171**	4 (30)	**89**	4 (39)	23	1 (2)
1	**117**	14 (235)	75	19 (403)	57	14 (328)	**69**	8 (184)

Matchcellar - h_{ff} Ground Size : 41.3 (80)

ϑ	FA-		FA+		PA-		PA+	
0.5	**465**	1 (478)	**140**	1 (12233)	**465**	1 (478)	**114**	1 (12233)
0.6	465	1 (478)	140	1 (12233)	465	1 (478)	114	1 (12233)
0.7	465	1 (478)	140	1 (12233)	465	1 (478)	114	1 (12233)
0.8	465	1 (478)	140	1 (12233)	345	2 (12711)	114	1 (12233)
0.9	360	2 (12711)	140	1 (12233)	345	3 (13190)	114	1 (12233)
1	330	3 (24944)	139	1 (12233)	181	5 (25902)	103	3 (13190)

Table 2. Sensitivity analysis of ϑ: for each planning approach, we report the number of solved instances (left), the number of lifted macro-events selected, and in parenthesis the average number (across testing set) of ground macro-events generated (right). In bold we highlight the best result, breaking ties by lowest average planning time (see appendix for times [19]).

6 Conclusion

In this paper, we tackled the problem of generalizing macro-events to a lifted representation. We defined a novel lifted formalization; we then discussed how to construct and select this kind of lifted macros from a collection of small training problems. We presented a fully automated learning pipeline that performs lifted macros selection in two different phases: first, through optimization of the statistical expectation on an abstraction of the problem; and second, refining the most promising lifted macro-events after a validation procedure involving the planner. Then, we experimentally showed the effectiveness of the proposed technique using an unsupervised approach for the generation of training plans.

For future work, as our experimental results suggested, we want to learn jointly the lifted macro-events and a heuristic that has been exposed to them during training. Furthermore, we want to study empirical estimators tailored to a specific planner, to perform a more accurate estimation of the number of states explored by the real planner instead of an abstract one.

Acknowledgments

This work has been supported by the STEP-RL project funded by the European Research Council under GA n. 101115870. This work has been carried out while Alessandro La Farciola was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome in collaboration with Fondazione Bruno Kessler.

References

[1] A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer. Macro-ff: Improving AI planning with automatically learned macro-operators. *J. Artif. Intell. Res.*, 24:581–621, 2005.

[2] M. Cardellini and E. Giunchiglia. Temporal numeric planning with patterns. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2025, 39(25)*, pages 26481–26489. AAAI Press, 2025.

[3] S. Castellanos-Paez, D. Pellier, H. Fiorino, and S. Pesty. Mining useful macro-actions in planning. In *2016 Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, pages 1–6, 2016.

[4] D. Z. Chen, M. Hao, S. Thiébaux, and F. W. Trevizan. Graph learning for planning: The story thus far and open challenges. *CoRR*, abs/2412.02136, 2024.

[5] D. Z. Chen, S. Thiébaux, and F. W. Trevizan. Learning domain-independent heuristics for grounded and lifted planning. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, 38(18)*, pages 20078–20086. AAAI Press, 2024.

[6] L. Chrpa and M. Vallati. Planning with critical section macros: Theory and practice. *J. Artif. Intell. Res.*, 74:691–732, 2022.

[7] A. Coles and A. Smith. Marvin: A heuristic search planner with online macro-action learning. *J. Artif. Intell. Res.*, 28:119–156, 2007.

[8] A. Coles, M. Fox, K. Halsey, D. Long, and A. Smith. Managing concurrency in temporal planning using planner-scheduler interaction. *Artificial Intelligence*, 173(1):1–44, 2009. ISSN 0004-3702.

[9] A. J. Coles, A. Coles, M. Fox, and D. Long. Forward-chaining partial-order planning. In R. I. Brafman, H. Geffner, J. Hoffmann, and H. A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, pages 42–49. AAAI, 2010.

[10] W. Cushing, S. Kambhampati, Mausam, and D. S. Weld. When is temporal planning really temporal? In M. M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI*, pages 1852–1859, 2007.

[11] M. De Bortoli, L. Chrpa, M. Gebser, and G. Steinbauer-Wagner. Enhancing temporal planning by sequential macro-actions. In S. A. Gaggl, M. V. Martinez, and M. Ortiz, editors, *Logics in Artificial Intelligence - 18th European Conference, JELIA, Proceedings*, volume 14281 of *Lecture Notes in Computer Science*, pages 595–604. Springer, 2023.

[12] T. de la Rosa, A. G. Olaya, and D. Borrajo. Using cases utility for heuristic planning improvement. In *Case-Based Reasoning Research and Development, 7th International Conference on Case-Based Reasoning, ICCBR 2007, Proceedings*, pages 137–148, 2007.

[13] A. Dulac, D. Pellier, H. Fiorino, and D. Janiszek. Learning useful macro-actions for planning with n-grams. In *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI*, pages 803–810. IEEE Computer Society, 2013.

[14] M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*, 2003.

[15] M. Ghallab, D. S. Nau, and P. Traverso. *Automated planning - theory and practice*. Elsevier, 2004.

[16] N. Gigante, A. Micheli, A. Montanari, and E. Scala. Decidability and complexity of action-based temporal planning over dense time. *Artif. Intell.*, 307:103686, 2022.

[17] E. Hansson. Temporal task and motion plans: Planning and plan repair: Repairing temporal task and motion plans using replanning with temporal macro operators, 2018.

[18] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[19] A. La Farciola, A. Valentini, and A. Micheli. Additional material for "Learning of Lifted Macro-Events for Heuristic-Search Temporal Planning". https://github.com/fbk-pso/step-rl, 2025.

[20] A. La Farciola, A. Valentini, and A. Micheli. Automatic selection of macro-events for heuristic-search temporal planning. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2025, 39(25)*, pages 26579–26586. AAAI Press, 2025.

[21] A. Micheli and E. Scala. Temporal planning with temporal metric trajectory constraints. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, 33(01)*, pages 7675–7682. AAAI Press, 2019.

[22] A. Micheli and A. Valentini. Synthesis of search heuristics for temporal planning via reinforcement learning. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, 35(13)*, pages 11895–11902. AAAI Press, 2021.

[23] A. Micheli, A. Bit-Monnot, G. Röger, E. Scala, A. Valentini, L. Framba, A. Rovetta, A. Trapasso, L. Bonassi, A. E. Gerevini, L. Iocchi, F. Ingrand, U. Köckemann, F. Patrizi, A. Saetti, I. Serina, and S. Stock. Unified planning: Modeling, manipulating and solving AI planning problems in python. *SoftwareX*, 29:102012, 2025.

[24] M. A. H. Newton, J. Levine, M. Fox, and D. Long. Learning macro-actions for arbitrary planners and domains. In M. S. Boddy, M. Fox, and S. Thiébaux, editors, *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, pages 256–263. AAAI, 2007.

[25] D. Smith, J. Frank, and W. Cushing. The anml language. In *KEPS 2008*, 2008.

[26] A. Valentini, A. Micheli, and A. Cimatti. Temporal planning with intermediate conditions and effects. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, 34(06)*, pages 9975–9982. AAAI Press, 2020.

[27] V. Vidal. The YAHSP planning system: Forward heuristic search with lookahead plans analysis. In *Proceedings of the Fourth International Planning Competition 2004*, 2004.

[28] V. Vidal. Yahsp2: Keep it simple, stupid. In *Proceedings of the Seventh International Planning Competition 2011*, page 83–90, 2011.

[29] P. Wullinger, U. Schmid, and U. Scholz. Spanning the middle ground between classical and temporal planning. In *Proceedings of the 22nd Workshop on Planen, Scheduling und Konfigurieren, Entwerfen (PuK 2008)*, 01 2008.