Automatic Selection of Macro-Events for Heuristic-Search Temporal Planning

Alessandro La Farciola, Alessandro Valentini, Andrea Micheli

Fondazione Bruno Kessler, Trento, Italy
alafarciola@fbk.eu, alvalentini@fbk.eu, amicheli@fbk.eu

Abstract

One of the major techniques to tackle temporal planning problems is heuristic search augmented with a symbolic representation of time in the states. Adding composite actions (macro-actions) to the problem is a simple and powerful approach to create "shortcuts" in the search space, at the cost of increasing the branching factor of the problem and thus the execution time of a heuristic search planner. Hence, it is of paramount importance to select the right macro-actions and minimize their number to optimize the planner performance.

In this paper, we introduce "macro-events": a simple, yet powerful, "shortcut" model similar to macro-actions for the case of temporal planning. Then, we present a novel ranking function to extract and select a suitable set of macro-events from a dataset of valid plans. In our ranking approach, we consider an estimation of the hypothetical search space for a blind search under four different exploitation schemata. Finally, we experimentally demonstrate that the proposed approach yields a substantial performance improvement for a state-of-the-art temporal planner.

Introduction

Automated temporal planning is the problem of synthesizing a course of actions to achieve a desired goal, given a formal model of the system to be controlled when time and temporal constraints are relevant. Despite a long history of research, scalability is still a major limiting factor for the applicability of domain-independent temporal planners: real-world scenarios are often out of reach for current systems.

One possibility to attack this problem is to specialize temporal planners for a domain of interest: if domain knowledge is provided to planners, the synthesis process can be accelerated. Among the proposed methods to automatically achieve this specialization (which nowadays is mostly focused on the synthesis of search heuristics using machine learning, e.g., (Chen, Thiébaux, and Trevizan 2024)), learning small portions of plans that the planner can use together with basic actions to form a complete plan is a simple and effective approach that is well-studied for the case of classical planning. These "shortcuts" in the search space are usually called "macro-actions" (or sometimes just "macro") and in classical and numeric planning (where actions are assumed to be instantaneous) they usually are sequences of actions that can be exploited to reduce the distance to the goal. In temporal planning, actions instances are intervals that can overlap in time, and sequencing durative actions is insufficient to capture many interesting temporal plans (Cushing et al. 2007).

In this paper, we tackle the problem of automatically learning structures similar to macro-actions for heuristic-search temporal planning. By assuming that the planner we specialize uses heuristic-search with time represented symbolically (like e.g. POPF (Coles et al. 2010), TAMER (Valentini, Micheli, and Cimatti 2020) and many others), we define a simple, yet effective, model of macros for temporal planning as sequences of actions with a semantics allowing for the representation of any concurrent temporal plan. The basic idea is to represent a totally-ordered sequence of *events* by only indicating the ID of the action to progress. We call these "macro-events" and in this paper we provide a formal, context-dependent semantics: 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-action 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. In general, macro-events cannot be encoded as normal durative actions, but they can be easily exploited by a heuristic search planner as additional domain knowledge provided in input. After formalizing this concept, we describe four different ways in which the same set of macro-events can be exploited by a temporal planner and then we discuss a novel statistics-based algorithm focused on extracting a suitable set of macro-events from a database of valid plans. Our selection approach estimates (under some simplifying assumptions) the search states expanded in the worst case by a blind-search planner equipped with a certain set of macro-events for each of the four exploitation schemata we propose. We then use this estimation to automatically select the most promising set of macro actions to be used at planning time. Finally, we experimentally show that our approach yields very good results on the two case-study problems used in (Micheli and Valentini 2021).

Background

Temporal Planning with ICE We start by formalizing the temporal planning problems we tackle by adapting the

ground model of (Valentini, Micheli, and Cimatti 2020).

Definition 1. *A **timing** τ is an expression that is either* START $+ k$ *or* END $- k$ *with* $k \in \mathbb{Q}_{\geq 0}$.

Intuitively, 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 an instant of time during an action.

Definition 2. *A **temporal planning problem** P is a tuple $\langle F, A, I, G \rangle$, where:*

- *F is a set of fluents, each $f \in F$ with a domain $Dom(f)$;*
- *A is a set of actions, each having a duration constraint; a set of timed effects of the form $[\tau] f := v$ with τ being a timing, $f \in F$ and $v \in Dom(f)$; and a set of conditions of the form $[\tau_1, \tau_2]\phi$ where τ_1 and τ_2 are timings and ϕ is an arbitrary boolean combination of atoms over F.*[1]
- *I is the initial total assignment of values to fluents; and*
- *G is the goal condition expressed as an arbitrary boolean combination of atoms over F.*

Basically, our planning model allows to specify a set of fluents (i.e., variables changing their value in time according to the selected actions) with arbitrary types. Moreover, it encompasses Intermediate Conditions and Effects (ICE), allowing the user to specify effects and conditions at arbitrary points during actions.

Definition 3. *A **temporal plan** π for a problem $\langle F, A, I, G \rangle$ is a set of tuples $\{\langle a_1, t_1, d_1 \rangle, \ldots, \langle a_n, t_n, d_n \rangle\}$, where each $a_i \in A$ is an action, $t_i \in \mathbb{Q}_{\geq 0}$ is its start time, and $d_i \in \mathbb{Q}_{>0}$ is its duration.*

For brevity, we do not formalize the semantics of this planning model (details can be found in (Valentini, Micheli, and Cimatti 2020) and in (Gigante et al. 2022)), but intuitively one can simulate a temporal plan by considering all the effects induced by the action instances in the plan and their timings: each effect changes the value of its fluent at that time and other fluents are left unchanged. The plan is valid if in the simulation, each condition of every plan action is satisfied and the goal is reached at the end of the trace.

As recognized by Gigante et al. (2022), one important semantical difference in temporal planning is self-overlapping, that is whether two instances of the same ground action are allowed or disallowed to overlap in time. The authors show that allowing self-overlapping makes the problem undecidable in general, while disallowing self-overlapping makes it PSPACE-complete. In this paper, we adopt the non-self-overlapping semantics, because it is almost never used in practice (Fox and Long 2007) and it will allow a much simpler representation of our macro-events. Formally, we disregard any plan π in which $\langle a, t_1, d_1 \rangle \in \pi$, $\langle a, t_2, d_2 \rangle \in \pi$ (with the same ground action a) and there exist a time t' s.t. $t_1 \leq t' \leq t_1 + d_1$ and $t_2 \leq t' \leq t_2 + d_2$.

Heuristic-Search for Temporal Planning One popular technique (but not the only one) to solve temporal planning problems is heuristic-search coupled with a symbolic representation of time. Introduced in Crikey (Coles et al. 2009) and adopted by many other planners (e.g., POPF, TAMER and others), the core idea of the approach is to decompose each action in its *events* to search in the space of ordering of events, using a symbolic representation of the timing constraints to check for temporal feasibility.

Definition 4. *Given an action $a \in A$, an **event** e is either:*

- *the start (*START(a)*) or ending (*END(a)*) of a;*
- *an effect x (*EFF(x, a)*) of a; or*
- *the starting (*START(c, a)*) or ending (*END(c, a)*) of a condition c of a.*

In the simpler setting of PDDL2.1 level 3 an event (also called "happening") is either the start or the end of a durative action; when ICE is allowed, events include effects and conditions starting and ending points during actions as well. Each concrete planning technique differs in the way it represents the search space and the temporal constraints and in the heuristic it employs; however, for the sake of this paper we can abstract these differences and consider high-level pseudo-code in Algorithm 1 (considering only the parts in black). Also, we do not need to further formalize events: we assume we can refer to an event of an action (writing $action(e)$ for the action e belongs to) and that given a state s, we can check the applicability of the event in s (line 9) and compute the state resulting from the application of an event e in s (line 10). Given an event e and a state s, we indicate with $e(s)$ the state resulting from applying e in s.

Without loss of generality, we assume that given an action $a \in A$, its events are totally ordered; i.e., we can define a list of events $events(a) = \langle e_1^a, \ldots, e_n^a \rangle$. As noted in (Valentini, Micheli, and Cimatti 2020), if this is not the case in the original problem, it suffices to add appropriate copies of an action splitting on the possible durations in order to avoid non-total orders. By exploiting our assumption of non-self-overlapping, given a state s and an action a we can define the next event of a in s (written $next(a, s)$) as either e_1^a (that is, 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 (GETNEXTEVENT at line 8).

One important note is that, unlike in classical or numeric planning, we do not check for already visited states, because in temporal planning states that have an identical assignment of the fluents might differ in terms of timings due to the path that has been followed to generate them, and checking equivalence of temporal states is a hard problem (Coles and Coles 2016). Therefore, this kind of planning algorithm usually explores a *tree* of states. Some temporal planners do check for state equivalence in some special cases, but this does not pose particular issues for our approach so we disregard this detail. Finally, the differences in terms of total/partial orderings in the symbolic time representation (e.g., POPF uses a partial ordering while TAMER uses a total ordering) are not relevant for what follows.

Planning with Macro-Events

We now formalize what we mean by "macro-events" and describe how to exploit them in heuristic search temporal planning. We assume a problem $P = \langle F, A, I, G \rangle$ is given.

[1] For this paper, the precise relations allowed in conditions are unimportant, our planner allows linear real arithmetic and equality.

Algorithm 1 Prototypical Heuristic-Search Algorithm for Temporal Planning (with Macro-Events)

```
1:  procedure SolveTemporalPlanning(P, M, ℵ)
2:    init ← InitialState(P)
3:    Q ← HeuristicPriorityQueue;  Push(Q, init)
4:    while NotEmpty(Q) do
5:      s ← Pop(Q)
6:      if IsGoal(s, P) then return GetPlan(s, P)
7:      for all a ∈ A do
8:        e ← GetNextEvent(a, s, P)
9:        if IsApplicable(e, s) then
10:         s' ← ApplyEvent(e, s)
11:         if CheckTemporalConstraints(s') then
12:           Push(Q, s')
13:     for all m ∈ M do
14:       ToAdd ← ∅;  b ← s;  steps ← 0
15:       for all a ∈ m do
16:         e ← GetNextEvent(a, s, P)
17:         if IsApplicable(e, s) then
18:           b' ← ApplyEvent(e, b)
19:           if CheckTemporalConstraints(b') then
20:             steps ← steps + 1
21:             if ℵ ∈ {FA+, PA+} ∧ steps ≥ 2 then
22:               ToAdd ← ToAdd ∪ {b'}
23:             b ← b'
24:           else break
25:         else break
26:       if ℵ ∈ {PA+, PA−} ∨ steps = |m| then
27:         if steps ≥ 2 then
28:           ToAdd ← ToAdd ∪ {b}
29:         PushAll(Q, ToAdd)
30:   return "No plan exists"
```

Definition 5. *A **macro-event** (or more briefly **macro**) is a sequence of actions $m = \langle a_1, \ldots, a_{|m|} \rangle$, where $a_i \in A$ and $|m|$ is the **length** of m.*
*Given a macro-event m, we indicate with $m^{\leq i}$ the **prefix-macro-event** $\langle a_1, \ldots, a_i \rangle$, with $m^{\geq i}$ the **suffix-macro-event** $\langle a_i, \ldots, a_{|m|} \rangle$ and with $m[i]$ the i-th element of m.*

The key intuition behind a macro-event is that, given a search state, one only needs to know which 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. More formally, given a macro m and a state s, the successor state $m(s)$ is the state obtained by iteratively applying the next event of each action in m:

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

Note that this is different from the classical notion of macro-action, where multiple (instantaneous) actions are summarized in a single, new action of the problem. A macro-event is more low-level: it describes a class of shortcuts in the search space of the problem when explored by a heuristic search planner. In general, a macro-event cannot be expressed as a macro-action at the planning level because it is context-dependent (its conditions and effect depend on the state it is applied into) and allows for arbitrary overlappings of different durative actions.

In order to exploit macro-events during planning, we need to define when a macro-event is considered applicable.

Definition 6. *A macro-event $m = \langle a_1, \ldots, a_{|m|} \rangle$ 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 $i \in \{2, \ldots, |m|\}$.*

Definition 7. *A macro-event m is **partially-applicable (PA)** in s if $m^{\leq 2}$ is fully-applicable in s. The **order** of m in s (written $o(m, s)$) is the maximum length of a prefix-macro-event of m fully-applicable in s.*

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

Another dimension that can be considered when exploiting macro-events is whether to search on states that are generated while evaluating prefixes of a macro or not.

Definition 8. *We define a search with macro-events to be **with intermediate nodes** $(+)$ if when applying a macro-event it adds each intermediate state in the search queue. Otherwise, we define it to be **without intermediate nodes** $(-)$ if it only adds the last state when applying a macro-event.*

The following table summarizes the four resulting approaches for exploiting macro-events during planning, while the red part of Algorithm 1 extends the basic heuristic-search pseudo-code to implement all these approaches (depending on the parameter ℵ) for a given set of macros M.

	without i.n.	**with i.n.**
fully-applicable	FA-	FA+
partially-applicable	PA-	PA+

The pseudo-code generates the intermediate states to evaluate each macro *in addition to the normal search performed when expanding a state*. We define a set of states *ToAdd* (line 14) that is initially empty and is filled by the intermediate states expanded while evaluating a macro in the FA+ and PA+ cases (line 22) and by the final state b in the other cases (line 28). Then, if all the events entailed by the macro are satisfied (*steps* $= |m|$, line 26) or we are in the PA cases, we enqueue all the generated states in *ToAdd* in the search queue (line 29), otherwise we discard these states as the macro is not deemed applicable. One minor note is that we disregard states generated after one step of a macro, because these are already enqueued by the basic search algorithm (lines 21 and 27). We highlight that the semantics usually adopted with macro-actions in the context of classical and numeric planning, and also in (De Bortoli et al. 2023) for temporal planning, coincides with the FA- case, as a macro-action is applied only if all its components are applicable in sequence. The other cases are novel and specific to macro-events as they impact the specific internal behavior of a planner. The key idea of exploring other approaches is to exploit a macro-event even in states where it would be inapplicable under FA-. For longer macro-events, PA can use meaningful prefixes even if later events become inapplicable, reducing the impact of imprecisions during learning or selection.

Learning Macro-Events

In this section, we discuss how to extract and select macro-events from a given database of valid plans. We do not assume optimality: we just need a set of *valid* plans on the

same domain for planning instances sharing the same object names[2]. This paper focuses on *ground* macro-events, that is we learn events on specific object instances without trying to generalize: we regard the synthesis of lifted macro-events as future work. We start by formalizing the extraction of events from a given plan by creating an ordered list of events it could be constructed from by using heuristic-search.

Definition 9. *The **timed set of events** (written $te(\pi)$) of a plan $\pi = \langle\langle a_1, t_1, d_1\rangle, \ldots, \langle a_n, t_n, d_n\rangle\rangle$ is such that:*
- *for each $\langle a_i, t_i, d_i\rangle$, $\langle t_i, \textsc{Start}(a_i)\rangle \in te(\pi)$ and $\langle t_i + d_i, \textsc{End}(a_i)\rangle \in te(\pi)$;*
- *for each $\langle a_i, t_i, d_i\rangle$ and for each effect $x = [\tau]f := v$ of a_i, $\langle t_\tau, \textsc{Eff}(x, a_i)\rangle \in te(\pi)$ where t_τ is the value of τ computed by setting $\textsc{Start} = t_i$ and $\textsc{End} = t_i + d_i$;*
- *for each $\langle a_i, t_i, d_i\rangle$ and each condition $c = [\tau_1, \tau_2]\phi$ of a_i, $\langle t_{\tau_1}, \textsc{Start}(c, a_i)\rangle \in te(\pi)$ and $\langle t_{\tau_2}, \textsc{End}(c, a_i)\rangle \in te(\pi)$.*

Definition 10. *Given a temporal plan π, its **sequence of events** is $ev(\pi) = \langle e_i \mid e_i \in te(\pi)\rangle$ sorted so that if e_i precedes e_j, then $\langle t_i, e_i\rangle, \langle t_j, e_j\rangle \in te(\pi)$ and $t_i \leq t_j$.*

To learn macro-events, we can forget about the specific events and simply transform the sequence of events of a plan in the sequence of actions each event belongs to.

Definition 11. *Given a temporal plan π, the **plan-macro-event** is a macro event $\mu(\pi) = \langle action(e) \mid e \in ev(\pi)\rangle$.*

From here on, we assume that a set of valid plan-macro-events Π is given, and we focus on the problem of extracting useful macro-events from this database. Concretely, we want to select a set of macro-events M that would increase the performance of a planner on a planning instance drawn from the same distribution as the ones represented in Π.

The first step consists in counting the frequencies (called utilities) of sub-strings inside Π: we assume a given maximum length ℓ_{max} for macro-events is chosen by the user and we count the frequencies of each sub-string of each plan-macro-event in Π. We also define the utility of a macro with respect to a set of macros M to only count the indices in which the macro is the longest one that is applicable in M.

Definition 12. *A macro m **appears in a plan-macro-event** $\mu(\pi)$ at position $j \in \mathbb{N}$ (written $at(m, \mu(\pi), j)$) if $\forall 1 \leq i \leq |m| \,.\, m[i] = \mu(\pi)[i + j]$.*
*The **utility of** m **in** $\mu(\pi)$ is $u^\pi_m = |\{j \mid at(m, \mu(\pi), j)\}|$.*
*The **utility of** m **in** $\mu(\pi)$ **relative to** M is $u^\pi_m(M) = |\{j \mid at(m, \mu(\pi), j) \wedge \nexists m' \in M.|m'| > |m| \wedge at(m', \mu(\pi), j)\}|$.*

We define the set of Candidate Macro-Events (CME) as the set of all macro-events that appear at least once in the dataset Π: $\text{CME} = \{m \mid |m| \leq \ell_{max} \text{ and } \sum_{\mu(\pi)\in\Pi} u^\pi_m > 0\}$.

At this point, we need a way to select a subset of CME to use during planning. The basic idea of our approach is to estimate, taking into consideration the planning approach $\aleph$, the number of states an heuristic-search algorithm would expand given a certain set of macros M. Using such a measure we could then select, among all the possible $M \subseteq \text{CME}$, the one that is expected to have the best *impact*.

[2]Planning instances might differ in the number of objects, but we assume a schematic naming of objects of the same type to make ground actions for one instance, potentially applicable to others.

Definition 13. *Given a problem P, a plan π, a (possibly empty) set of macro-events M, and a planning approach $\aleph$, we define $ES(M, \aleph, P, \pi)$ as the number of **expanded states** by the planning approach $\aleph$ to find $\mu(\pi)$. We define the **impact** $I(M, \aleph, P, \pi)$ of M as the gain in terms of expanded states: $ES(\emptyset, \text{no macros}, P, \pi) - ES(M, \aleph, P, \pi)$.*

Note that $I(M, \aleph, P, \pi) > 0$ iff M positively impacts the planning process, meaning that the total number of expanded states is reduced compared to planning without macros.

Definition 14. *The **optimal set of macro-events** $M^*(\aleph, P, \pi)$ for a problem P, plan π and planning approach $\aleph$ is the subset of CME with the highest impact:*

$$M^*(\aleph, P, \pi) = \underset{M\subseteq \textbf{CME}}{\arg\max}\, I(M, \aleph, P, \pi).$$

Clearly, the exact number of expanded states is heavily influenced by the planning algorithm characteristics and the chosen heuristic. However, we can present the following general result concerning the PA+ planning approach that holds irrespectively of the chosen algorithm and heuristic.

Proposition 1. *Let m_1 be a macro and m_2 a prefix of m_1, (i.e. $\exists k$ such that ${m_1}^{\leq k} = m_2$) then for all P and π, $I(\{m_1\}, PA+, P, \pi) \geq I(\{m_1, m_2\}, PA+, P, \pi)$.*

Proof. (Sketch) Note that thesis follows if and only if $\text{ES}(\{m_1\}, PA+, P, \pi) \leq \text{ES}(\{m_1, m_2\}, PA+, P, \pi)$. Let s be any state. The case in which both macros are not applicable in s is trivial. If m_1 is fully-applicable in s (and so also m_2), thanks to partial applicability definition, the planner would expand states $\{{m_1}^{\leq j}(s) : 2 \leq j \leq |m_1|\} \cup \{{m_2}^{\leq i}(s) : 2 \leq i \leq |m_2|\}$. Since one is a prefix of the other, the second element of the union is also equal to $\{{m_1}^{\leq i}(s) : 2 \leq i \leq k\}$. Therefore, possible paths to find plan-macro-event $\mu(\pi)$ are the same if we consider the union of two previous sets or just the first element. Finally, a similar argument follows also in the case in which m_1 is partially-applicable of order $o(m_1, s) < |m_1|$. □

We now switch to theoretical arguments to evaluate the impact of a single macro and the combined effects of a set of macros for a planner. We assume an abstraction of a real planner execution in which every action is applicable in every search state and the planner is blind (i.e. no heuristic is used). This is of course not how a real heuristic search planner works, but we use these assumptions to compare the theoretical number of states expanded by each approach in these abstract conditions. We will then use this estimation to select a set of macros from CME.

Definition 15. *Given an abstract problem P, a set of macros M and a plan π, we define the following notation.*
- *$N(P)$: number of ground actions in problem P.*
- *L^π: length of $\mu(\pi)$.*
- *$T(M, \aleph, P)$: search-space when using macros in M.*
- *$A(M, \aleph, P)$: branching factor (or arity) of $T(M, \aleph, P)$.*
- *$D(M, \aleph, P, \pi)$: min depth of $T(M, \aleph, P)$ to find $\mu(\pi)$.*

In our abstract setting, $\text{ES}(M, \aleph, P, \pi)$ coincides with the cardinality of $T(M, \aleph, P)$, i.e. the total number of nodes of

a complete tree of depth $D(M, \aleph, P, \pi)$ and branching factor $A(M, \aleph, P)$[3]. Hence, we have that:

$$\text{ES}(M) = \sum_{j=0}^{D_M} A_M^j = \frac{(A_M)^{D_M+1} - 1}{A_M - 1}. \quad (1)$$

Variables in the latter formula depend on the planning approach, but equation (1) still remains valid. The following table lists variables values in the four approaches we consider, and in the following paragraphs we give more details on how we obtained such results. We highlight that thanks to our assumptions (i.e., every action is always applicable), there are no differences in terms of applicability or depth between fully- and partially-applicable macros: we indicate with $A_M^{\pm}$ the branching factor and with $D_M^{\pm}$ the depth of the search tree in cases FA+, PA+ and FA-, PA-, respectively.

	$\boldsymbol{A_M}$	$\boldsymbol{D_M}$
no macros	N	L^π
FA-, PA-	$N + \lvert M\rvert$	$L^\pi - \sum_{m\in M} u_m^\pi(M)(\lvert m\rvert - 1)$
FA+, PA+	$N - \lvert M\rvert + \sum_{m\in M} \lvert m\rvert$	$L^\pi - \sum_{\hat{m}\in \hat{M}} u_{\hat{m}}^\pi(\hat{M})(\lvert \hat{m}\rvert - 1)$

No macros This is the case where $M = \emptyset$, and A_M coincides with the total number of ground actions of problem P. To find $\mu(\pi)$ we need to explore the tree until depth L^π.

FA-, PA- In this case, every macro expands exactly one state independently, i.e. different macros expand different states. Therefore, the branching factor of the search-space is augmented by the number of macro-events in M. Its depth depends on the length of every macro and on its utility in the plan-macro-event relative to M (i.e. $u_m^\pi(M)$).

FA+, PA+ During search, expanded states are always different; hence, to compute branching factor and depth in this case, we can use the following result.

Theorem 1. *Let P be an abstract problem, M a set of macros, and π a plan. Consider the union set $\hat{M} := \bigcup_{m\in M} \{m^{\le j} \mid 2 \le j \le |m|\}$, and the disjoint union set $\tilde{M} := \bigsqcup_{m\in M} \{m^{\le j} \mid 2 \le j \le |m|\}$. Then, (i) $A_M^+ = A_{\tilde{M}}^-$ and (ii) $D_M^+ = D_{\hat{M}}^-$ hold.*

Proof. (Sketch) Since each macro is assumed to be applicable and we consider all intermediate nodes, for every m we expand all the intermediate states related to $m^{\le j}$, with $2 \le j \le |m|$, counting them with repetitions. Then, $A_M^+ = A_{\tilde{M}}^-$ follows directly.

For the second identity, we observe that all prefixes can be used in order to pursue the plan-macro-event. However, in this computation, it is necessary to count each prefix as one. That's why the identity follows with $\hat{M}$ instead of $\tilde{M}$. □

An immediate consequence is that for planning approach *with intermediate nodes*, the following hold:

- $A_M = N + |\tilde{M}| = N - |M| + \sum_{m\in M} |m|$;
- $D_M = L^\pi - \sum_{\hat{m}\in\hat{M}} u_{\hat{m}}^\pi(\hat{M})(|\hat{m}| - 1)$.

[3]For simplicity, from now on we drop the full dependencies of each variable unless necessary; reducing $N(P)$ to N, $A(M, \aleph, P)$ to A_M, $D(M, \aleph, \pi, P)$ to D_M, and $\text{ES}(M, \aleph, P, \pi)$ to $\text{ES}(M)$.

Let us make a small example to clarify what has been presented so far. Suppose to have a problem with ground actions $\{a, b, c, x\}$, a plan-macro-event $\langle a, b, x, a, b, c\rangle$, and candidate macros $\text{CME} = \{\langle a, b\rangle, \langle a, b, c\rangle\}$. Then, $A_\emptyset = 4$, $D_\emptyset = 6$, and so $ES(\emptyset) = 5461$. Moreover, we obtain:

$\boldsymbol{M}$	$\boldsymbol{A_M^-}$	$\boldsymbol{D_M^-}$	$\text{ES}(\boldsymbol{M})^-$	$\boldsymbol{A_M^+}$	$\boldsymbol{D_M^+}$	$\text{ES}(\boldsymbol{M})^+$
$\{\langle a, b\rangle\}$	5	4	781	5	4	781
$\{\langle a, b, c\rangle\}$	5	4	781	6	3	259
$\{\langle a, b\rangle, \langle a, b, c\rangle\}$	6	3	259	7	3	400

The set that maximizes the impact is the one that minimizes column $\text{ES}(M)^-$ and $\text{ES}(M)^+$ respectively. Hence, $M^*(\aleph^-) = \{\langle a, b\rangle, \langle a, b, c\rangle\}$, and $M^*(\aleph^+) = \{\langle a, b, c\rangle\}$.

Estimation of the Abstract Impact

In the rest of the section, we aim at computing good estimators for the abstract impact defined above, in order to rank candidate macro-events and select the best set M^*.

Considering all utilities relative to every possible set of macros $M \subseteq \text{CME}$ would cost a huge computational effort. For this reason, for our estimations, we make the strong (but not restrictive) assumption to fix for any macro m its utility always equal to the base case: $u_m^\pi(M) \equiv u_m^\pi$.

Now, let us define a total order between candidate macros. In particular, for the approaches FA- and PA-, we say that $m_1 < m_2$ if the impact of $\{m_1\}$ is less than the impact of $\{m_2\}$. Then, a key theorem for macros selection holds.

Theorem 2. *Let $\aleph^- \in \{$FA-, PA-$\}$ and $M_1, M_2 \subseteq$ CME such that $|M_1| = |M_2|$ and $m_1 < m_2 \;\; \forall m_1 \in M_1, \forall m_2 \in M_2$. Then, $I(M_1, \aleph^-, P, \pi) < I(M_2, \aleph^-, P, \pi)$.*

Proof. Let $n = |M_1| = |M_2|$; the thesis follows from:

$$\frac{(N+n)^{D_{M_1}+1} - 1}{N + n - 1} > \frac{(N+n)^{D_{M_2}+1} - 1}{N + n - 1} \;. \quad (2)$$

Moreover, equation (2) is true iff $D_{M_1} > D_{M_2}$. But, $D_{M_i} = L^\pi - \sum_{m_i\in M_i} u_{m_i}^\pi(|m_i| - 1)$, for $i = 1, 2$. Then,

$$D_{M_1} > D_{M_2} \Leftrightarrow \sum_{m_1\in M_1} u_{m_1}^\pi(|m_1|-1) < \sum_{m_2\in M_2} u_{m_2}^\pi(|m_2|-1)$$

because $m_1 < m_2 \Leftrightarrow u_{m_1}^\pi(|m_1|-1) < u_{m_2}^\pi(|m_2|-1)$. □

An immediate consequence of previous theorem is that, in the approach *without intermediate nodes*, if we consider the set of candidates macros ordered according to previous order, i.e. $\text{CME} = \{m_1 > m_2 > \ldots > m_K\}$, then for any integer $2 \le n \le K$:

$$\underset{\{M\subseteq \text{CME} \,:\, |M|=n\}}{\arg\max} \; I(M, \aleph^-, P, \pi) \;=\; \{m_1, \ldots, m_n\} \,.$$

This means that fixed any cardinality of macros sets, the best one (i.e., the set with the highest impact value) is composed of macro-events with the highest single impact. That has a clear relevance for implementation, exponentially limiting the number of sets to take into account. We now consider the empirical means over the database of valid plans (Π) and the dataset of problems ($\mathbb{P}$) for the quantites defined above:

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

Equation (1) gives us a general formula to compute the abstract number of expanded states where we can use previous estimators for branching factor and depth of the space-search tree. In particular, the estimation changes depending on the macro-events schema we are interested in. To overcome the influence of the strong assumption made at the beginning of the paragraph, we estimate D_M as the maximum between value obtained from estimators and the minimum depth reachable with macros M, which corresponds to $\bar{L}/l_{max}$, where $l_{max} = \max\{|m| \ : \ m \in M\}$.

The abstract setting described above only discriminates the $\pm$ cases. What differentiates the FA vs PA cases is the estimation of the depth of the search-space tree. In the first case, for each macro m, we reduce the depth of the tree without macros of the quantity $\bar{u}_m(|m| - 1)$. In the second case, given the definition of partial applicability, we estimate such quantity as a convex combination of its prefixes, weighted on their utility: given a macro-event m, we use the expression $\sum_{i=2}^{|m|} \lambda_i \left(\bar{u}_{m^{\leq i}}(|m^{\leq i}| - 1)\right)$, 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 $i = |m|$.

Finally, we obtained complete procedures to find the optimal set M^* for any usage case. We estimate for every candidate macro the impact it provides alone (i.e. assuming only this one macro is passed to the planner), for the cases of *full applicability* using its utility, for the case of *partial applicability* using its weighted utility. Then, we sort our CME according to such computed value. Now, in the case *without intermediate nodes*, we construct sets of macros by adding candidates one by one according to the sorted list and comparing the impacts of such sets. With this procedure we are guaranteed to find the optimal set thanks to Theorem 2. Conversely, in the case *with intermediate nodes*, we must evaluate the impact of all possible combinations. Since this is an exponential effort, we provide an anytime algorithm that considers subsets according to the sorted list and we interrupt the search with a timeout using the best set reached. At the end, we check that the optimal set has a positive impact.

Related Work

Macro-actions are well-known in classical planning. Some authors focused on generating macro-actions during the planning process, like in the case of the Marvin planner (Coles and Smith 2007), which identifies plateaus in the search-space and learns how to escape from them thanks to suitable "shortcuts" based on heuristic values; then, it tries to replicate them in similar scenarios. Another on-line procedure is the YAHSP planner (Vidal 2004, 2011), which computes relaxed sequences of actions and tries to apply them in the search states (and to repair it if needed), adding the last state reachable in the expansion if the whole relaxed plan is inapplicable. Such *lookahead* strategy is similar in some aspects to our PA- approach, even if in our case we consider events and valid plans (instead of actions and relaxed plan). Most existing approaches aim at learning macro-actions off-line using training problems. One seminal work in this direction is the MACRO-FF planner (Botea et al. 2005), which extracts macros from either solution plans or from static problem analysis. Another off-line technique is (Newton et al. 2007), which employs a genetic algorithm in order to explore macros occurring in successful plans ranking them in terms of time gained solving more difficult problems. Other works extend or generalize previous approaches, like composing sequences of macro-actions (Botea, Müller, and Schaeffer 2007), or building a library of potentially useful macros for Marvin (Coles, Fox, and Smith 2007). More recently, (Dulac et al. 2013) introduced a domain-independent approach extracting statistical information from valid plans based on a n-gram analysis and using them to filter useful macros. Chrpa, Vallati, and McCluskey (2014) try to maximize macros utilities learning relations between planning operators and predicates (called entanglements), while Castellanos-Paez et al. (2018) use data mining techniques on frequent sequences in order to select macros.

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 (Fox and Long 2003) or ANML (Smith, Frank, and Cushing 2008). (Wullinger, Schmid, and Scholz 2008), (Hansson 2018) and the very recent work (De Bortoli et al. 2023) all try to encapsulate sequences of durative actions by constructing macro-operators. In particular, De Bortoli et al. insist on guaranteeing, while using such temporal macros, the sequential applicability of original actions avoiding unnecessary suppression of other concurrent actions.

What we present in this paper is considerably different. First, we do not construct macro-actions in the usual sense, with summarized conditions and effects; instead, we introduce the novel concept of *macro-event* as a sequence of actions that is dependent on the path context. Moreover, our discussion on the four exploitation approaches of macros in planning is completely novel, with the usual concept of applicability of a macro converging to the *fully-applicable* case. Finally, our selection schema based on expanded state estimation is also novel to the best of our knowledge.

Experimental Evaluation

We implemented a heuristic-search temporal planner supporting macro-events in Python using the Unified Planning library (Micheli et al. 2025) as modeling framework. Our planner relies on TAMER for the definition of the search space and optionally takes in input a set of macro-events to use and the planning approach to follow, implementing Algorithm 1. Moreover, we implemented the estimators for the different planning approaches defined in the previous section in Python, taking in input a list of plan-macro-events Π and the approach $\aleph$ and producing the estimated optimal set of macros. To construct the dataset Π, we employ the Reinforcement Learning approach described in (Micheli and Valentini 2021) recording all the solutions found during learning. In this way, we do not depend on a specific heuristic and the entire approach can be fully automated given a set of temporal planning instances. We considered the two benchmark domains in (Micheli and Valentini 2021) (enlarging the problem instances to make the problems more challenging). MAJSP consists of scheduling a fleet of agents to transport items between machines; Kitting requires a robot

MAJSP										
fold (testing size: 123)	No Macros		Using Macro Events							
			h_{add}				h_{ff}			
	h_{add}	h_{ff}	FA-	FA+	PA-	PA+	FA-	FA+	PA-	PA+
1	27	87	37 (26)	43 (25)	73 (66)	35 (20)	92 (69)	94 (63)	84 (51)	**96 (72)**
2	38	93	28 (8)	40 (13)	54 (41)	40 (15)	88 (49)	**98 (61)**	83 (48)	97 (63)
3	26	82	22 (15)	40 (28)	54 (48)	34 (18)	87 (69)	87 (44)	79 (55)	**91 (74)**
4	21	80	33 (26)	39 (22)	69 (62)	29 (13)	86 (66)	86 (54)	83 (52)	**87 (57)**
all	112	342	120 (75)	162 (88)	250 (217)	138 (66)	353 (253)	365 (222)	329 (206)	**371 (266)**

Kitting										
fold (testing size: 121)	No Macros		Using Macro-Events							
			h_{add}				h_{ff}			
	h_{add}	h_{ff}	FA-	FA+	PA-	PA+	FA-	FA+	PA-	PA+
1	48	57	57 (49)	39 (37)	54 (50)	54 (25)	**58 (49)**	**58 (49)**	53 (37)	58 (22)
2	54	66	64 (51)	44 (38)	58 (53)	43 (37)	64 (54)	**71 (59)**	71 (39)	71 (58)
3	51	60	57 (45)	37 (32)	57 (51)	44 (36)	61 (52)	**62 (46)**	62 (31)	61 (50)
4	54	54	52 (39)	32 (25)	60 (55)	**65 (37)**	57 (44)	59 (51)	51 (19)	53 (37)
all	207	237	230 (184)	152 (132)	229 (209)	206 (135)	240 (199)	**250 (205)**	237 (126)	243 (167)

Table 1: Coverage results: for each fold and each planning approach we report the number of solved instances and in parentheses the number of cases where the macro approach is faster than baseline using the same heuristic.

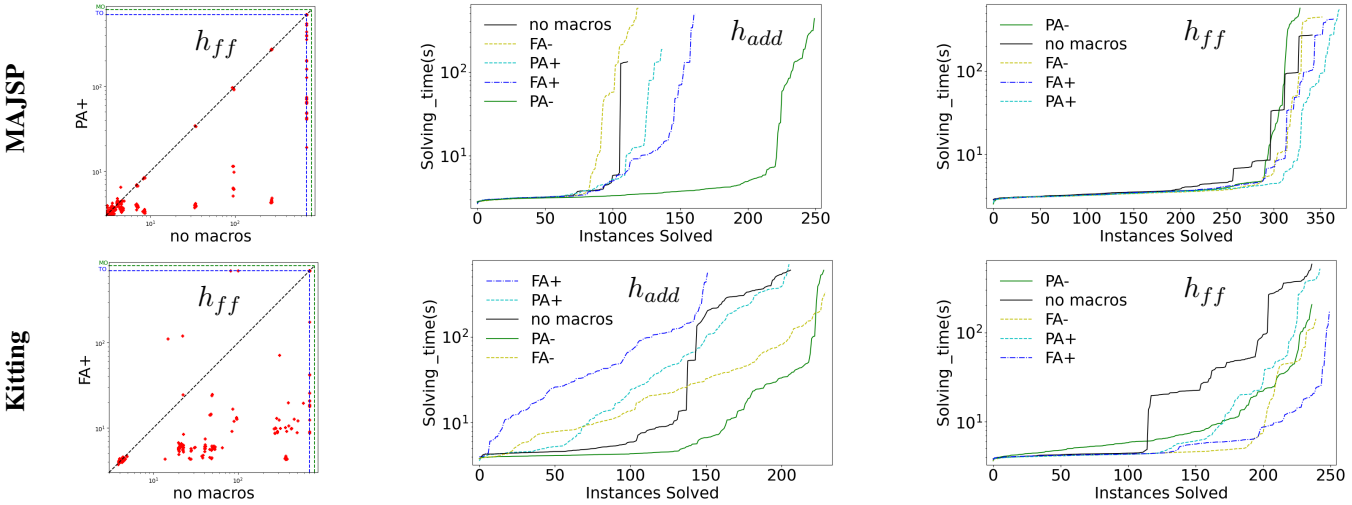

Figure 1: Plots of the results. The scatter plots consider the best performing solver against the baseline with the same heuristic.

to gather components from various warehouse locations to assemble a kit and deliver it in sync with a human operator. For each domain, we perform a 4-fold splitting of the instances using the training part to construct the plan-macro-events and the testing part to evaluate our planner equipped with the selected set of macros against the baseline TAMER planner without macros. We chose as maximum length of candidates macro-events $\ell_{max} = 5$. The size of the extracted CME is 23619 for Kitting and 61378 for MAJSP. The average (wrt the 4-folds) time of execution for macros selection is 5s (Kitting) and 13s (MAJSP) in case FA-, 54s (Kitting) and 355s (MAJSP) in case PA-; in cases *with intermediate nodes* we allocated 30 minutes for the anytime algorithm. 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 40GB (the maximum memory used in the experiments was 6.4GB). Benchmarks, code and all the scatter plots are available at https://github.com/fbk-pso/step-rl. Table 1 shows the coverage results using the h_{add} and h_{ff} heuristics provided by TAMER. In all cases, one of the approaches using macro-events is faster than the baseline without macros. The performance difference is also very significant, as highlighted by the cactus plots in Figure 1: macro-based approaches always dominate both for coverage and for instances where an approach is faster than the baseline.

Conclusions

In this paper, we present a novel representation for macros tailored to heuristic-search approaches for temporal planning. We discuss how to extract and select this kind of macros from a dataset of valid plans and experimentally show the effectiveness of the technique using a fully unsupervised approach for the generation of training plans.

For future work, we want to lift the learned macro-events (i.e., make the macro-events independent of the problem objects) and study empirical estimators (e.g., by using reinforcement learning) specialized to a specific planner, to estimate the number of states explored by the real planner.

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

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators. *Journal of Artificial Intelligence Research*, 24: 581–621.

Botea, A.; Müller, M.; and Schaeffer, J. 2007. Fast Planning with Iterative Macros. In Veloso, M. M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, 1828–1833.

Castellanos-Paez, S.; Pellier, D.; Fiorino, H.; and Pesty, S. 2018. Mining useful Macro-actions in Planning. *CoRR*, abs/1810.09145.

Chen, D. Z.; Thiébaux, S.; and Trevizan, F. W. 2024. Learning Domain-Independent Heuristics for Grounded and Lifted Planning. In Wooldridge, M. J.; Dy, J. G.; and Natarajan, S., eds., *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024*, 20078–20086. AAAI Press.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2014. MUM: A Technique for Maximising the Utility of Macro-operators by Constrained Generation and Use. In Chien, S. A.; Do, M. B.; Fern, A.; and Ruml, W., eds., *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014*. AAAI Press.

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

Coles, A.; Fox, M.; and Smith, A. 2007. Online Identification of Useful Macro-Actions for Planning. In Boddy, M. S.; Fox, M.; and Thiébaux, S., eds., *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007*, 97–104. AAAI Press.

Coles, A.; and Smith, A. 2007. Marvin: A Heuristic Search Planner with Online Macro-Action Learning. *Journal of Artificial Intelligence Research*, 28: 119–156.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In Brafman, R. I.; Geffner, H.; Hoffmann, J.; and Kautz, H. A., eds., *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010*, 42–49. AAAI Press.

Coles, A. J.; and Coles, A. I. 2016. Have I Been Here Before? State Memoization in Temporal Planning. In Coles, A. J.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016*, 97–105. AAAI Press.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. S. 2007. When is Temporal Planning Really Temporal? In Veloso, M. M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, 1852–1859.

De Bortoli, M.; Chrpa, L.; Gebser, M.; and Steinbauer-Wagner, G. 2023. Enhancing Temporal Planning by Sequential Macro-Actions. In Gaggl, S. A.; Martinez, M. V.; and Ortiz, M., eds., *Logics in Artificial Intelligence - 18th European Conference, JELIA 2023, Proceedings*, volume 14281 of *Lecture Notes in Computer Science*, 595–604. Springer.

Dulac, A.; Pellier, D.; Fiorino, H.; and Janiszek, D. 2013. Learning Useful Macro-actions for Planning with N-Grams. In *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2013*, 803–810.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20: 61–124.

Fox, M.; and Long, D. 2007. A note on concurrency and complexity in temporal planning. In *PlanSIG 2007*.

Gigante, N.; Micheli, A.; Montanari, A.; and Scala, E. 2022. Decidability and complexity of action-based temporal planning over dense time. *Artificial Intelligence*, 307: 103686.

Hansson, E. 2018. Temporal Task and Motion Plans: Planning and Plan Repair: Repairing Temporal Task and Motion Plans Using Replanning with Temporal Macro Operators.

Micheli, A.; Bit-Monnot, A.; Röger, G.; Scala, E.; Valentini, A.; Framba, L.; Rovetta, A.; Trapasso, A.; Bonassi, L.; Gerevini, A.; Iocchi, L.; Ingrand, F.; Köckemann, U.; Patrizi, F.; Saetti, A.; Serina, I.; and Stock, S. 2025. Unified Planning: Modeling, Manipulating and Solving AI Planning Problems in Python. *SoftwareX*. Forthcoming.

Micheli, A.; and Valentini, A. 2021. Synthesis of Search Heuristics for Temporal Planning via Reinforcement Learning. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, 11895–11902.

Newton, M. A. H.; Levine, J.; Fox, M.; and Long, D. 2007. Learning Macro-Actions for Arbitrary Planners and Domains. In Boddy, M. S.; Fox, M.; and Thiébaux, S., eds., *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007*, 256–263. AAAI Press.

Smith, D.; Frank, J.; and Cushing, W. 2008. The ANML language. In *KEPS 2008*.

Valentini, A.; Micheli, A.; and Cimatti, A. 2020. Temporal Planning with Intermediate Conditions and Effects. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 9975–9982. AAAI Press.

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

Vidal, V. 2011. YAHSP2: Keep It Simple, Stupid. In *Proceedings of the Seventh International Planning Competition 2011*, 83–90.

Wullinger, P.; Schmid, U.; and Scholz, U. 2008. Spanning the Middle Ground between Classical and Temporal Planning. In *Proceedings of the 22nd Workshop on Planen, Scheduling und Konfigurieren, Entwerfen (PuK 2008)*.