An Extensive Empirical Analysis of Macro-Actions for Numeric Planning

Diaeddin Alarnaouti, Francesco Percassi[0000−0001−7332−0636], and Mauro Vallati[0000−0002−8429−3570]

University of Huddersfield, Huddersfield, United Kingdom
diaeddin.alarnaouti@hud.ac.uk

Abstract. Automated Planning is a pivotal field of artificial intelligence, focusing on intelligent agents' ability to generate action sequences leading from an initial state to a desired goal condition. A well-known technique to improve planning performance is based on macro-actions, which can reduce search depth by merging multiple primitive actions together, generating "shortcuts" in the search space. Macros have been studied extensively in classical planning, but rarely in more expressive formalisms.
In this study, we investigate macro-actions in numeric planning, formalising the macro generation process and exploring a semi-automated methodology for selecting candidate primitive actions to be combined into macro-actions. Our extensive experimental analysis demonstrates the potential benefits of macros for numeric planning engines, providing useful insights into their effectiveness for efficient plan generation.

Keywords: Automated Planning · Macro Actions · Numeric Planning.

1 Introduction

Automated Planning is a prominent field of artificial intelligence, that focuses on the capability of intelligent agents to generate sequences of actions whose application, starting from a given initial state, would lead to a state where a given goal condition is satisfied [15].

In domain-independent planning, the separation between planning knowledge and reasoning supports the use of reformulation approaches [23]. These techniques involve automatically re-formulating, re-representing, or adjusting the planning knowledge [21, 25, 26, 30] to enhance the efficiency of planning engines or to allow the use of different classes of solving approaches. In literature, there is a substantial body of work on reformulation techniques, primarily focused on the classical planning paradigm [1]. Among other reformulation techniques, macro-actions are a well-studied approach, that aims to reduce the search depth by merging actions together, to provide shortcuts in the search space [2, 6, 8, 9, 22]. Scala [27] proposed the most relevant work on macro actions in numeric planning. Still, the work focuses on macros for repairing plans, rather than macros

that can boost the plan generation process. Given the positive performance impact that macro techniques allow to achieve in classical planning, a promising avenue for research is using macros in more expressive planning formalisms.

Given the recent renewed interest in numeric planning, both from a practical and theoretical point of view [5, 7, 16, 17, 19, 24, 28, 29], in this study, we investigate macro-actions in the context of numeric planning. We formalise the macro process generation in numeric planning, and introduce a semi-automated methodology for selecting candidate primitive actions to be combined into macro-actions. We introduce a categorisation of macros for numeric planning by accounting for syntactical aspects, which can help shed some light on performance and the expected impact of macros. Finally, we run an extensive empirical evaluation of macros in numeric planning to assess their impact on plan generation performance.

2 Background

Traditionally, numeric planning tasks are described by means of PDDL2.1 [11]. A *numeric planning task* is a pair $\Pi = \langle \mathcal{D}, \mathcal{P} \rangle$ where $\mathcal{D}$ is a *planning domain model* and $\mathcal{P}$ is a *planning problem*. $\mathcal{D}$ is a tuple $\langle F, X, A \rangle$ where F and X are sets of Boolean and numeric functions returning values in $\mathbb{B} = \{\bot, \top\}$ and $\mathbb{Q}$, respectively, and A is a set of lifted actions. Elements from $F \cup X$ are referred to as lifted functions. $\mathcal{P}$ is a tuple $\langle O, I, G \rangle$ where O is a set of objects, I is the initial state, and G is the goal description. A function $p \in F \cup X$ has arity k, and every occurrence of p in Π features k arguments $\{v_1, \ldots, v_k\}$, written as $p(v_1, \ldots, v_k)$. Numeric functions can be combined to obtain mathematical expressions defined as $\varphi := \varphi + \varphi \mid \varphi * \varphi \mid x \mid q$, where $x \in X$ and $q \in \mathbb{Q}$.

Let F^O (X^O) be the sets of all Boolean (numeric) variables formed from the functions F (X) by substituting objects O in the functions' arguments. Elements from $F^O \cup X^O$ are referred to as grounded variables. A state s is a complete assignment over the variables $F^O \cup X^O$, mapping elements from F^O to $\mathbb{B}$ and from X^O to $\mathbb{Q}$. The initial state I is then a complete assignment over $F^O \cup X^O$. The goal G is a partial assignment over $F^O \cup X^O$.

A lifted action a is a tuple $\langle \text{par}(a), \text{name}(a), \text{pre}(a), \text{eff}(a) \rangle$, where $\text{par}(a)$ is the set of arguments of a, $\text{name}(a)$ is the unique identifier of the action, $\text{pre}(a)$ (preconditions of a) is a set of conditions, $\text{eff}(a)$ (effects of a) is a set of effects. $\text{pre}(a)$ is partitioned in numeric and Boolean conditions, i.e., $\text{pre}(a) = \text{pre}_{\mathsf{num}}(a) \cup \text{pre}_{\mathsf{prop}}(a)$. A numeric condition in $\text{pre}_{\mathsf{num}}(a)$ has the form $\langle \varphi \bowtie 0 \rangle$, where φ is a mathematical expression defined over X and $\mathbb{Q}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. A Boolean condition in $\text{pre}_{\mathsf{prop}}(a)$ has the form p or $\neg p$ where $p \in F$. Similarly, the effects are partitioned into numeric and Boolean effects, i.e., $\text{eff}(a) = \text{eff}_{\mathsf{num}}(a) \cup \text{eff}_{\mathsf{prop}}(a)$. A numeric effect in $\text{eff}_{\mathsf{num}}(a)$ has the form $\langle x := \varphi \rangle$ where $x \in X$ and φ is a mathematical expression defined over X and $\mathbb{Q}$. A Boolean condition in $\text{pre}_{\mathsf{prop}}(a)$ has the form p or $\neg p$ where $p \in F$. To differentiate between positive and negative effects, we use $\text{eff}^+_{\mathsf{prop}}(a)$ and $\text{eff}^-_{\mathsf{prop}}(a)$, respectively.

Let A^O be the set of all grounded actions formed from the lifted actions A by substituting objects O for the parameter symbols in their preconditions and effects. This process produces *grounded actions* $\langle \text{pre}(a), \text{eff}(a) \rangle$ without arguments, where $\text{pre}(a)$ and $\text{eff}(a)$ are defined over variables $F^O \cup X^O$. All definitions of lifted conditions, effects, and mathematical expressions defined over $F \cup X$ can be syntactically redefined for grounded variables.

Let s be a state, v a variable in $F^O \cup X^O$ and φ a mathematical expression defined over X^O and $\mathbb{Q}$, we denote with $s[v]$ the value assumed by v in s, and with $s[\varphi]$ the evaluation of φ in s. A numeric condition $c = \langle \varphi \bowtie 0 \rangle$ is satisfied by a state s, denoted $s \models c$, if and only if $s[\varphi] \bowtie 0$ holds. Similarly, the Boolean condition v ($\neg v$) is satisfied if and only if $s[v] = \top$ ($s[v] = \bot$). A state s satisfies a set of conditions C, denoted $s \models C$ if and only if $\bigwedge_{c \in C} s \models c$. Applying a grounded action a in a state s yields a new state $s' = \gamma(s, a)$ where:

$$s[v] = \begin{cases} \top & \text{if } v \in \text{eff}(a), v \in F^O \\ \bot & \text{if } \neg v \in \text{eff}(a), v \in F^O \\ s[\varphi] & \text{if } \langle v := \varphi \rangle \in \text{eff}(a), v \in X^O \\ s[v] & \text{otherwise (frame axiom)} \end{cases}$$

An action is applicable in a state s if $s \models \text{pre}(a)$, there are no conflicting effects, i.e., more than one effect affecting the same variable differently, and the result of each numeric effect $s[\varphi]$ is well-defined. A plan π for Π is a sequence of grounded actions, i.e., $\langle a_1, \ldots, a_n \rangle$. A plan π is valid for Π if each action is iteratively applicable starting from I and the resulting state achieves G.

2.1 Example

To illustrate the concepts presented in the background, we will refer to a well-known domain, namely SETTLERS [20]. We will focus on two actions that will be used to demonstrate the generation of macro-actions.

Let $O = \{p_1, p_2\}$ be a set of objects representing two places. We consider a single Boolean function, parameterised by p, that indicates whether a coal stack is present at place p, i.e., $F = \{\textit{has-coal-stack}(p)\}$. Additionally, we use numeric functions to track the available timber or coal at each location, as well as the overall pollution emitted or labour consumed, i.e., $X = \{\textit{timb}(p), \textit{coal}(p), \textit{poll}, \textit{lab}\}$.

We consider two lifted actions a_1 and a_2 to build a coal stack or to burn the coal, respectively. Both actions are parametrised in p, i.e., $\textit{par}(a_1) = \textit{par}(a_2) = \{p\}$ and $\textit{name}(a_1) = \textit{build-coal-stack}$ and $\textit{name}(a_2) = \textit{burn-coal}$. The preconditions and effects of these actions are defined as:

$$\begin{aligned} \text{pre}(a_1) &= \{\langle \textit{timb}(p) \geq 1 \rangle\} \\ \text{eff}(a_1) &= \{\textit{has-coal-stack}(p), \langle \textit{lab} := \textit{lab} + 2 \rangle, \langle \textit{timb}(p) := \textit{timb}(p) - 1 \rangle\} \\ \text{pre}(a_2) &= \{\textit{has-coal-stack}(p), \langle \textit{timb}(p) \geq 1 \rangle\} \\ \text{eff}(a_2) &= \{\langle \textit{timb}(p) := \textit{timb}(p) - 1 \rangle, \langle \textit{coal}(p) := \textit{coal}(p) + 1 \rangle, \langle \textit{poll} := \textit{poll} + 1 \rangle\}. \end{aligned}$$

The grounded variables instantiated through the objects $O = \{p_1, p_2\}$ are $F^O = \{\textit{has-coal-stack}(p1), \textit{has-coal-stack}(p2)\}$ and $X^O = \{\textit{timb}(p1), \textit{timb}(p2), \textit{coal}(p1), \textit{coal}(p2), \textit{poll}, \textit{lab}\}$. To illustrate the grounding process for the lifted actions, we consider action a_1 grounded using place p_1. The resulting grounded action is $a_1^{p_1}$, where $\text{pre}(a_1^{p_1}) = \{\langle \textit{timb}(p_1) \geq 1\rangle\}$ and $\text{eff}(a_1^{p_1}) = \{\textit{has-coal-stack}(p_1), \langle \textit{lab} := \textit{lab} + 1\rangle, \langle \textit{timb}(p_1) := \textit{timb}(p_1) - 1\rangle\}$.

3 Generating Macro-Actions

This section describes how a pair of candidate lifted actions can be combined to obtain an equivalent macro-action. We refer to the input actions as candidates because two actions may not be combinable, for instance, if the first action consumes a resource violating a precondition of the second one.

The generation of macro-actions in the numeric context must consider the Boolean and numeric components. For the Boolean component, a standard approach from classical planning is used. On the other hand, the numeric component is handled by adopting a technique from the work of Scala [27].

Algorithm 1 outlines the overall methodology. This algorithm takes two candidate actions, namely a_i and a_j, along with an identifier id that ensures the generation of uniquely named macro-actions. The resulting macro-action, denoted as $a_{i,j}$ is obtained by separately combining the propositional and numeric preconditions, as well as effects.

The parameters of $a_{i,j}$ are obtained by merging the parameters of the candidate actions, while the name is obtained by simply chaining the names of a_i and a_j and the unique identifier id. The propositional precondition of $a_{i,j}$ is obtained by merging the preconditions of a_i and a_j, from which the additive effects of a_i are removed. Specifically, these are removed because they may not hold before applying a_i.

The merging of numeric preconditions is accomplished using the procedure MergeNumPrecs($\cdot$), the pseudocode of which is outlined in Algorithm 2. Specifically, the numeric preconditions of $a_{i,j}$ are constructed by directly including all the preconditions of a_i in their original form. In contrast, the preconditions of a_j must be manipulated to account for the effects of a_i. Specifically, the algorithm iterates over all the numeric conditions $\langle \varphi \bowtie 0\rangle \in \text{pre}(a_j)$, and adds the regressed condition $\langle \mathcal{S}(\varphi, \text{eff}(a_i)) \bowtie 0\rangle$ to $\text{pre}(a_{i,j})$. Here, $\mathcal{S}$ is a function that substitutes each numeric function x appearing in φ with the right-hand side expression of the numeric effect involving x in $\text{eff}(a_j)$. If there is no interaction between $\text{eff}(a_i)$ and φ the substitution function returns φ untouched.

Similarly, for the effects, it is necessary to assemble the Boolean and numeric effects of the two actions. For the former, the positive (negative) effects of $a_{i,j}$ are obtained by merging the positive (negative) effects of the two actions, from which the negative (positive) effects of a_j are subtracted. The numeric effects of $a_{i,j}$ are obtained using the procedure MergeNumEffs($\cdot$), the pseudocode of which is provided in Algorithm 3. Intuitively, the effects of $a_{i,j}$ are obtained (i) by adding all the effects of a_j and projecting forward the effects of a_i. (ii) All

Algorithm 1 Generation of a numeric macro-action.

Input: A pair of actions a_i, a_j and a unique identifier id.
Output: A macro-action $a_{i,j}$.

procedure GENERATEMACRO(a_i, a_j, id)
  $\text{par}(a_{i,j}) \leftarrow \text{par}(a_i) \cup \text{par}(a_j)$
  $\text{name}(a_{i,j}) \leftarrow \text{name}(a_i) + \text{“}+\text{”} + \text{name}(a_j) + \text{“}\_\text{”} + id$
  $\text{pre}_{\mathsf{prop}}(a_{i,j}) \leftarrow (\text{pre}_{\mathsf{prop}}(a_i) \cup \text{pre}_{\mathsf{prop}}(a_j)) \setminus \text{eff}^{+}_{\mathsf{prop}}(a_i)$
  $\text{pre}_{\mathsf{num}}(a_{i,j}) \leftarrow \text{MERGENUMPRECS}(\text{pre}_{\mathsf{num}}(a_i), \text{pre}_{\mathsf{num}}(a_j))$
  $\text{pre}(a_{i,j}) \leftarrow \text{pre}_{\mathsf{num}}(a_{i,j}) \cup \text{pre}_{\mathsf{prop}}(a_{i,j})$ ▷ $a_{i,j}$ preconditions
  $\text{eff}^{-}_{\mathsf{prop}}(a_{i,j}) \leftarrow (\text{eff}^{-}_{\mathsf{prop}}(a_i) \cup \text{eff}^{-}_{\mathsf{prop}}(a_j)) \setminus (\text{eff}^{+}_{\mathsf{prop}}(a_j))$
  $\text{eff}^{+}_{\mathsf{prop}}(a_{i,j}) \leftarrow (\text{eff}^{+}_{\mathsf{prop}}(a_i) \cup \text{eff}^{+}_{\mathsf{prop}}(a_j)) \setminus (\text{eff}^{-}_{\mathsf{prop}}(a_j))$
  $\text{eff}_{\mathsf{prop}}(a_{i,j}) \leftarrow \text{pre}^{+}_{\mathsf{prop}}(a_{i,j}) \cup \text{pre}^{-}_{\mathsf{prop}}(a_{i,j})$
  $\text{eff}_{\mathsf{num}}(a_{i,j}) \leftarrow \text{MERGNUMEFFS}(\text{eff}_{\mathsf{num}}(a_i), \text{eff}_{\mathsf{num}}(a_j))$
  $\text{eff}(a_{i,j}) \leftarrow \text{eff}_{\mathsf{num}}(a_{i,j}) \cup \text{eff}_{\mathsf{prop}}(a_{i,j})$ ▷ $a_{i,j}$ effects
  $a_{i,j} \leftarrow \langle \text{par}(a_{i,j}), \text{name}(a_{i,j}), \text{pre}(a_{i,j}), \text{eff}(a_{i,j}) \rangle$
  return $a_{i,j}$
end procedure

effects of a_i that do not interfere with the effects of a_j, and therefore have not been projected forward through a_j, must be added to the effects of $a_{i,j}$.

Point (i) is handled by the first *for* loop. Each effect $\langle x := \varphi \rangle \in \text{eff}_{\mathsf{num}}(a_j)$ is added to the effects of $a_{i,j}$ by projecting the effects of a_i using the substitution function $\mathcal{S}$. If a_i does not interfere, the function returns φ unchanged.

Algorithm 2 Merging of numeric preconditions.

Input: A pair of numeric preconditions $\text{pre}_{\mathsf{num}}(a_i)$ and $\text{pre}_{\mathsf{num}}(a_j)$.
Output: A numeric precondition $\text{pre}_{\mathsf{num}}(a_{i,j})$.

procedure MERGENUMPRECS($\text{pre}_{\mathsf{num}}(a_i), \text{pre}_{\mathsf{num}}(a_j)$)
  $\text{pre}_{\mathsf{num}}(a_{i,j}) = \text{pre}_{\mathsf{num}}(a_i)$
  for $\langle \varphi \bowtie 0 \rangle \in \text{pre}_{\mathsf{num}}(a_j)$ **do**
    $\text{pre}_{\mathsf{num}}(a_{i,j}) = \text{pre}_{\mathsf{num}}(a_{i,j}) \cup \{\langle \mathcal{S}(\varphi, \text{eff}(a_i)) \bowtie 0 \rangle\}$
  end for
  return $\text{pre}_{\mathsf{num}}(a_{i,j})$
end procedure

4 Selecting Candidates

This section outlines the methodology for selecting candidates for macro-actions in numeric planning, based on the algorithm described in the previous section.

The efficient generation of macro-actions is well-known to be extremely challenging, due to the potentially large number of possible combinations of actions.

Generating macro-actions efficiently is well-known to be extremely challenging, due to the potentially large number of possible combinations of actions to

Algorithm 3 A pair of set of numeric conditions $\text{eff}_{\mathsf{num}}(a_i)$ and $\text{eff}_{\mathsf{num}}(a_j)$.

Input: a_i and a_j actions.
Output: A macro-action $a_{i,j}$

procedure MergeNumEffs($\text{eff}_{\mathsf{num}}(a_i)$, $\text{eff}_{\mathsf{num}}(a_j)$)
  $\text{eff}_{\mathsf{num}}(a_{i,j}) \leftarrow \emptyset$
  for $\langle x := \varphi \rangle \in \text{eff}_{\mathsf{num}}(a_j)$ **do**
    $\text{eff}_{\mathsf{num}}(a_{i,j}) \leftarrow \text{eff}_{\mathsf{num}}(a_{i,j}) \cup \{\langle x := \mathcal{S}(\varphi, \text{eff}_{\mathsf{num}}(a_i))\rangle\}$
  end for
  for $\langle x := \varphi \rangle \in \text{eff}_{\mathsf{num}}(a_i)$ s.t. φ does not interfere with $\text{eff}_{\mathsf{num}}(a_j)$ **do**
    $\text{eff}_{\mathsf{num}}(a_{i,j}) \leftarrow \text{eff}_{\mathsf{num}}(a_{i,j}) \cup \{\langle x := \varphi \rangle\}$
  end for
  return $\text{eff}_{\mathsf{num}}(a_{i,j})$
end procedure

be considered. For instance, when considering only pairs of actions to combine, the number of candidates grows quadratically with respect to the number of actions. For this reason, in this work, we initially explored possible macros for each domain by partially adopting an approach used in classical planning [9, 10]. This approach generates macro-actions by analysing several plans and extracting pairs or groups of actions that are promising candidates for combination. Specifically, to select actions for creating effective macro-actions, we applied the following conditions:

- Actions were identified to often occur sequentially in generated plans across various problem instances, using different planning engines. This suggests that these actions are likely needed to be used in sequence to achieve common goals in the considered domain.
- There should be no conflicts between the preconditions and effects of candidate actions. In particular, the effects of the first action should not delete preconditions of the following.

After selecting the candidate actions for each domain, the input for the Algorithm 1 is ready.

The increased expressiveness of PDDL2.1 when compared to classical planning allows for a classification of generated macros based on a syntactic criterion. We did this to investigate whether different kinds of macro-actions might have varying experimental impacts. Specifically, we consider the following types:

- *same numeric fluent* (shortened as SNF): this category includes all macro-actions obtained by combining two primitive actions that share at least one numeric fluent in their preconditions or effects;
- *different numeric fluent* (shortened as DNF): this category includes all macro-actions in which the two primitives involve at least one numeric variable, and these variables do not overlap;
- *propositional* (shortened as PROP): this category includes all macro-actions in which at least one of the two primitives has no numeric component.

It is worth to remind that the most intuitive class to consider is SNF, as it represents macro-actions formed by combining actions that share at least one numeric variable and are therefore believed to be useful in a numeric setting.

For our investigation, we consider well-known numeric domains: DEPOTS, ROVERS, SETTLERS, TPP, and ZENOTRAVEL. These domains are taken from different editions of the International Planning Competition (IPC). Specifically, DEPOTS is from the second IPC [3]; ROVERS, SETTLERS, and ZENOTRAVEL are from the third IPC [20]; and TPP is from the fifth IPC [14]. In the following, we provide a brief description of each domain, followed by a description of the macro-actions that were derived from the original formulation, classifying them by type.

DEPOTS The domain focuses on actions involving loading and unloading trucks using hoists fixed at specific locations. The loads are crates, which can be stacked and unstacked onto pallets available at these locations. Trucks hold crates flexibly, allowing crates to be rearranged as needed. The trucks can be moved between locations, to deliver packages to specific depots. The numeric version includes weight attributes for crates and weight capacities for trucks, with fuel consumption needing to be minimised during travel and crate handling.

The actions in many plans often follow a similar pattern, with `lift` and `load` complementing each other, as well as `unload` and `drop`. Driving the truck to the required locations can occur before or after these actions. The macro-action `lift+load` is classified as DNF because the first action involves the `fuel-cost` variable, while the second action involves the `current_load` variable. The macro-action `unload+drop` is classified as PROP because `drop` is a pure propositional action. Additionally, we consider a macro-action obtained by sequencing three actions, i.e., `lift`, `load`, and `drive`. This macro-action is classified as SNF as `lift` and `drive` share the `fuel-cost` variable.

ROVERS Inspired by planetary rover problems, this domain involves navigating a planet's surface, collecting samples, and communicating findings to a lander. Some rovers can only traverse specific terrains, and data transmission requires direct visibility between waypoints and the lander. The numeric component of the domain introduces energy consumption for rover activities and allows recharging only in sunlight, emphasising efficient energy management.

According to our methodology, firstly, we combined the actions `calibrate` and `take_image`. We observed that whenever `calibrate` is used in any generated plan, the action `take_image` always follows for the same objects because the camera must be calibrated before use. This macro-action is classified as SNF because the `calibrate` action involves `energy` both in its preconditions and effects, and the same applies to the `take_image` action.

Secondly, we combined the actions `sample_soil` and `sample_rock` with `drop`. In both instances, the `sample` action should be followed by the `drop` action to ensure that the rover's storage is empty and available for other operations. These macros are classified as PROP since `drop` is a propositional action.

Settlers Due to the numerous actions available in this domain, it is particularly well-suited for exploring macro-actions. The analysis of the plans generated for settlers resulted in 16 macro-actions, covering the SNF and DNF classes, each with the same number of actions (see Table 1 for details).

TPP The Traveling Purchaser Problem (TPP) is an extension of the Traveling Salesman Problem. It involves selecting markets to buy a set of products, minimising both travel and purchase costs. Each market offers products at different prices and limited quantities. The numeric version features three operators, two of them, `buy-all` and `buy-allneeded`, handle purchasing actions. The `buy-all` operator buys all available goods of a specific type at a market, while the `buy-allneeded` operator buys only the remaining amount needed to meet the purchase goals. In this model, all markets are interconnected and linked to depots. There is one depot and one truck available for transportation.

In most generated plans, the action `drive` consistently precedes the actions `buy-all` and `buy-allneeded`. The two resulting macro-actions are thus classified as SNF since they involve distinct numeric variables. Specifically, both actions involve the `total-cost` variable, but even other variables.

ZenoTravel This domain revolves around transportation, utilising planes to carry passengers through two modes: fast movement, called `zoom`, and slow movement, called `fly`. In the numeric formulation, aircraft fuel consumption varies based on the mode of travel. Each plane is characterised by its unique fuel consumption rate and passenger capacity.

Analysing a multitude of plans, we have observed that the `board` action is typically followed by either `fly` or `zoom`. This makes sense since passengers are boarded to be transported. Similarly, the `fly` and `zoom` actions are always followed by `debark`. All of these macro-actions involve different numeric variables and are therefore classified as DNF.

Table 1: Summary of all the macro-actions generated in the domains.

Domain	Macro Class		
	SNF	DNF	PROP
Depots	`lift+load+drive`	`lift+load`	`unload+drop`
Rovers	`calibrate+take_image`	*na*	`sample_soil+drop`, `sample_rock+drop`
Settlers	`load+move-cart`, `move+cart-load`, `move+cart-unload`, `b-cabin+fell-timber`, `b-quarry+break-stone`, `b-coalstack+burn-coal`, `b-sawmill+saw-wood`, `b-docks+b-wharf`	`load+move-train`, `load+move-ship`, `move+train-load`, `move+ship-load`, `move+ship-unload`, `b-mine+mine-ore`, `b-ironworks+make-iron`, `b-docks+b-wharf`	*na*
TPP	`drive+buy-all`, `drive+buy-allneeded`	*na*	*na*
ZenoTravel	*na*	`board+fly`, `board+zoom` `fly+debark`, `zoom+debark`	*na*

4.1 Example

In this example, we analyse the generation of an SNF macro-action by referring to the SETTLERS example using Algorithm 1. We combine the actions a_1 (`build-coal-stack`) and a_2 (`burn-coal`).

The two involved actions share the same parameter p representing a place, then $par(a_{1,2}) = \{p\}$. Given a unique identifier $id =$ "1", we combine the names of a_1 and a_2 obtaining $name(a_{1,2}) = build\text{-}coal\text{-}stack{+}burn\text{-}coal\_1$.

As for the Boolean preconditions, note that the only Boolean condition of a_2 is $has\text{-}coal\text{-}stack(p) \in \text{pre}_{\mathsf{prop}}(a_2)$. However, $has\text{-}coal\text{-}stack(p) \in \text{eff}^{+}_{\mathsf{prop}}(a_1)$. Then, by Line 4 of Algorithm 1, we obtain $\text{pre}(a_{i,j}) = (\text{pre}_{\mathsf{prop}}(a_1) \cup \text{pre}_{\mathsf{prop}}(a_2)) \setminus \text{eff}^{+}_{\mathsf{prop}}(a_1) = (\emptyset \cup \{has\text{-}coal\text{-}stack(p)) \setminus \{has\text{-}coal\text{-}stack(p)\} = \emptyset$.

As for the numeric preconditions, note that there is an interaction between the effect $e = \langle timb(p) := timb(p) - 1\rangle \in \text{eff}_{\mathsf{num}}(a_1)$ and the condition $c = \langle timb(p) \geq 1\rangle \in \text{pre}_{\mathsf{num}}(a_2)$. Therefore the formula involved in the condition c, i.e., $timb(p) - 1$, must be regressed considering the effect e. By applying the substitution function as Line 4 from Algorithm 2, i.e., $\mathcal{S}(timb(p) - 1, \{timb(p) := timb(p) - 1, \ldots\}) = timb(p) - 2$, we obtain the numeric condition $\langle timb(p) \geq 2\rangle$ to be added to $\text{pre}_{\mathsf{num}}(a_{i,j})$. This condition reflects that a_2 requires one unit of fuel and, since a_1 consumes one unit, their sequential execution requires at least two units of fuel.

As for the Boolean effects, the only effect to be added to the effects of $a_{i,j}$ is $has\text{-}coal\text{-}stack(p) \in \text{eff}_{\mathsf{prop}}(a_1)$ as it is not negated by a_2. By applying Line 8 of Algorithm 2, we obtain $\text{eff}^{+}_{\mathsf{prop}}(a_{i,j}) = (\text{eff}^{+}_{\mathsf{prop}}(a_1) \cup \text{eff}^{+}_{\mathsf{prop}}(a_2)) \setminus \text{eff}^{-}_{\mathsf{prop}}(a_2) = (\{has\text{-}coal\text{-}stack(p)\} \cup \emptyset) \setminus \emptyset = \{has\text{-}coal\text{-}stack(p)\}$. Since there are no negated effects we have that $\text{eff}^{-}_{\mathsf{prop}}(a_{1,2}) = \emptyset$ and then $\text{eff}_{\mathsf{prop}}(a_{1,2}) = \{has\text{-}coal\text{-}stack(p)\}$.

As for the numeric effects, note that the only effects to be combined are those concerning the available timber. In contrast, all others can be merged without further manipulations as no interactions exist among them. Specifically, the right-hand-side effect $\langle timb(p) := timb(p) - 1\rangle \in \text{eff}_{\mathsf{num}}(a_2)$ must undergo the substitution function to take into account the effect of a_1 affecting $timb(p)$. Then, by applying Line 4 of Algorithm 3, i.e., $\mathcal{S}(timb(p) - 1, \text{eff}_{\mathsf{num}}(a_1)) = \mathcal{S}(timb(p) - 1, \{timb(p) := timb(p) - 1\}) = timb(p) - 2$, we obtain a new composite effect $\langle timb(p) := timb(p) - 2\rangle$ to be added to the effect of $a_{1,2}$. This effect reflects the fact that, since both actions consume one unit of fuel, their sequential execution results in the consumption of two units of fuel. Assembling all the elements obtained, the action $a_{1,2}$ has the following preconditions and effects:

$$\begin{aligned}\text{pre}(a_{1,2}) =& \{\langle timb(p) \geq 2\rangle\}\\ \text{eff}(a_{1,2}) =& \{has\text{-}coal\text{-}stack(p), \langle lab := lab + 2\rangle, \langle timb(p) := timb - 2\rangle,\\ & \langle coal(p) := coal(p) + 1\rangle, \langle poll := poll + 1\rangle\}\end{aligned}$$

5 Experimental Analysis

The experimental analysis aims to empirically assess the effectiveness of the macros in the context of numeric planning. We evaluated the impact of using

different types of reformulations—SNF, DNF, and PROP —compared to the original version of the models, across four well-established numeric planners. We also considered a fourth formulation, called ALL, in which we combined all macros of different types. Regarding the numeric planning systems, to explore a variety of methodologies, the following planners have been considered: ENHSP [27], LPG [12, 13], METRIC-FF [18], and OPTIC [4]. All experiments run on an Intel Core i9-10885H CPUs with 2.40GHz with a cutoff time of 900 seconds, and 8GB of RAM.

Table 2 provides an overview of the results in terms of coverage, i.e., the number of problems solved, and IPC-Score calculated for the runtime, denoted as IPC-Score(T). The IPC-Score for the runtime is calculated according to the metric commonly used in the International Planning Competitions [29]. Specifically, given a planning problem p in the test suite solved in t seconds, the IPC-Score(T) of p is assigned to 1 if $t \leq 1$, $1 - \log(t)/\log(900)$ otherwise.

As a first interesting result, it is easy to note that the effectiveness of macros in improving planning performance varies widely. This is influenced by several factors, including the type of macro employed, the characteristics of the domain, and the planning system. For example, in the case of DEPOTS, applying reformulation techniques consistently proves beneficial for ENHSP. When all macros are utilised together in this domain, there is a notable increase in coverage, with six additional instances being solved. Conversely, in the ZENOTRAVEL domain, the impact of macros is less favourable. Here, we observe a decrease in coverage for both ENHSP and LPG. This indicates that these macros in ZENOTRAVEL might introduce additional complexities to the problem, in the form of increased breadth, that outweigh their potential benefits. However, for other systems within this domain, the coverage is unaffected.

In all other examined benchmark domains, the influence of macros on coverage is minimal, with only minimal fluctuations observed. These oscillations are both positive and negative, indicating that the impact of macros is not uniformly beneficial or detrimental but varies depending on the specific circumstances, as observed in previous work on macros.

To clarify the mixed results, we measured the frequency with which macros provided faster solutions compared to the original formulation. Figure 1 illustrates, for each planner, the proportion of instances where a specific type of macro allows a planner to solve a problem instance faster. It is clear that ENHSP benefited the most from macros, especially with SNF, PROP, and ALL macros. Other planners saw more modest, but still significant, improvements, with up to 20% of instances being solved faster. Additionally, Figure 2 shows, for each planner, the proportion of instances where any macro enabled a faster solution than the original formulation, with results broken down by domain.

Overall, the performed analysis indicates that some planners are more prone than others to exploit the potential benefits of macros in numeric planning problems. However, for all considered planners macros have been beneficial for solving at least a few instances from the benchmarks. Turning our attention to the do-

Table 2: Results about the macro reformulation versus the original formulation (ORIGINAL) in terms of coverage and IPC-Score(T). *na* denotes the absence of macros of the considered type.

		Coverage					IPC-Score(T)				
		ORIGINAL	SNF	DNF	PROP	ALL	ORIGINAL	SNF	DNF	PROP	ALL
Depots	ENHSP	8	11	11	10	14	5.0	4.6	4.9	4.5	5.2
	LPG	20	20	20	20	20	18.8	15.5	18.2	17.3	12.8
	METRIC-FF	0	0	0	0	0	0	0	0	0	0
	OPTIC	0	0	0	0	0	0	0	0	0	0
	Σ	28	31	31	30	34	23.8	20.1	23.0	21.8	18.1
		ORIGINAL	SNF	DNF	PROP	ALL	ORIGINAL	SNF	DNF	PROP	ALL
Rovers	ENHSP	0	0	*na*	0	0	0	0	*na*	0	0
	LPG	19	19	*na*	19	20	17.0	17.7	*na*	17.5	17.7
	METRIC-FF	0	0	*na*	0	0	0	0	*na*	0	0
	OPTIC	0	0	*na*	0	0	0	0	*na*	0	0
	Σ	19	19	*na*	19	20	17.0	17.7	*na*	17.5	17.7
		ORIGINAL	SNF	DNF	PROP	ALL	ORIGINAL	SNF	DNF	PROP	ALL
Settlers	ENHSP	0	0	0	*na*	0	0	0	0	*na*	0
	LPG	0	0	0	*na*	0	0	0	0	*na*	0
	METRIC-FF	7	4	8	*na*	5	4.5	3.3	4.5	*na*	2.8
	OPTIC	4	3	1	*na*	1	1.9	2.2	0.5	*na*	0.5
	Σ	11	7	9	*na*	6	6.4	5.6	5.0	*na*	3.3
		ORIGINAL	SNF	DNF	PROP	ALL	ORIGINAL	SNF	DNF	PROP	ALL
TPP	ENHSP	9	9	*na*	*na*	9	7.2	7.2	*na*	*na*	7.2
	LPG	20	19	*na*	*na*	19	20.0	13.0	*na*	*na*	13.0
	METRIC-FF	0	0	*na*	*na*	0	0	0	*na*	*na*	0
	OPTIC	20	20	*na*	*na*	20	15.0	13.7	*na*	*na*	13.7
	Σ	49	48	*na*	*na*	48	42.2	33.9	*na*	*na*	33.9
		ORIGINAL	SNF	DNF	PROP	ALL	ORIGINAL	SNF	DNF	PROP	ALL
ZenoTravel	ENHSP	19	*na*	17	*na*	17	17.1	*na*	13.5	*na*	13.5
	LPG	20	*na*	15	*na*	15	19.7	*na*	14.5	*na*	14.5
	METRIC-FF	20	*na*	20	*na*	20	19.8	*na*	16.8	*na*	16.8
	OPTIC	17	*na*	17	*na*	17	16.0	*na*	14.3	*na*	14.3
	Σ	76	*na*	69	*na*	69	72.6	*na*	59.2	*na*	59.2

mains, ZENOTRAVEL is not suitable for using macros, while most of the others show significant benefits.

6 Conclusions

In this study, we conducted an extensive investigation into the use of macro-actions in numeric planning. We formalised the macro generation process and explored a semi-automated methodology for selecting candidate primitive actions to be combined into macro-actions. Our experimental analysis demonstrates both the potential benefits and drawbacks of using macros in numeric planning en-

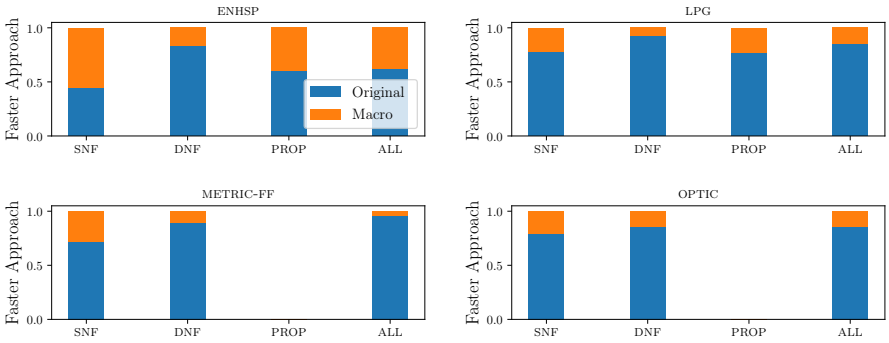

Fig. 1: Percentage of problems solved faster by a given planner when using the original domain model (blue) or the corresponding macro set (orange). Results are cumulative across all considered benchmarks.

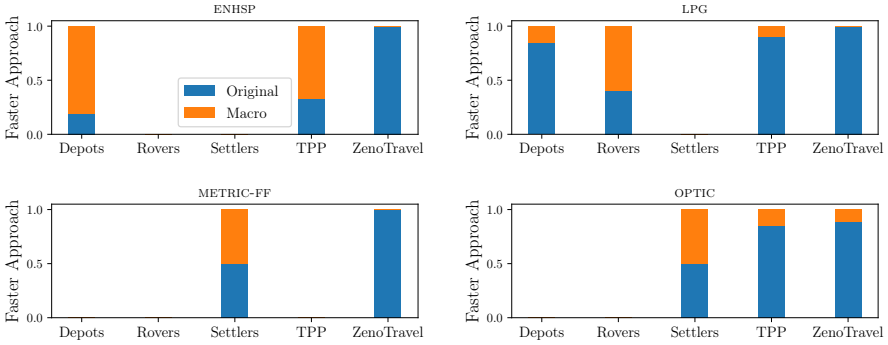

Fig. 2: Percentage of problems solved faster by a given planner when using the original domain model (blue) or any corresponding macro set (orange) on each considered benchmark domain.

gines. The results highlight that planners can respond very differently to domain models enhanced with macro-actions, and that some domain models may not be suitable for reformulation.

Future work will focus on identifying techniques for filtering and selecting macros that are tailored to the needs of the planning system, to characterise domain models suitable for macros reformulation, and explore the possibility to generate macros by combining more primitive actions together.

References

1. Alarnaouti, D., Baryannis, G., Vallati, M.: Reformulation techniques for automated planning: a systematic review. Knowl. Eng. Rev. **38**, e9 (2023)

2. Asai, M., Fukunaga, A.: Solving Large-Scale Planning Problems by Decomposition and Macro Generation. In: ICAPS. pp. 16–24 (2015)
3. Bacchus, F.: The AIPS '00 Planning Competition. AI Mag. **22**(3), 47–56 (2001)
4. Benton, J., Coles, A., Coles, A.: Temporal Planning with Preferences and Time-Dependent Continuous Costs. In: ICAPS. vol. 22, pp. 2–10 (2012)
5. Bonassi, L., Gerevini, A.E., Scala, E.: Dealing with Numeric and Metric Time Constraints in PDDL3 via Compilation to Numeric Planning. In: AAAI. pp. 20036–20043 (2024)
6. Botea, A., Enzenberger, M., Müller, M., Schaeffer, J.: Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators. J. Artif. Intell. Res. **24**, 581–621 (2005)
7. Cardellini, M., Giunchiglia, E., Maratea, M.: Symbolic Numeric Planning with Patterns. In: AAAI. vol. 38, pp. 20070–20077 (2024)
8. Castellanos-Paez, S., Pellier, D., Fiorino, H., Pesty, S.: Mining useful Macro-actions in Planning. In: AIPR. pp. 1–6. IEEE (2016)
9. Chrpa, L.: Generation of macro-operators via investigation of action dependencies in plans. Knowl. Eng. Rev. **25**(3), 281–297 (2010)
10. Chrpa, L., Barták, R.: Towards Getting Domain Knowledge: Plans Analysis through Investigation of Actions Dependencies. In: FLAIRS. pp. 531–536 (2008)
11. Fox, M., Long, D.: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. J. Artif. Intell. Res. **20**, 61–124 (2003)
12. Gerevini, A., Saetti, A., Serina, I.: Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. J. Artif. Intell. Res. **20**, 239–290 (2003)
13. Gerevini, A., Saetti, A., Serina, I.: An approach to efficient planning with numerical fluents and multi-criteria plan quality. Artif. Intell. **172**(8-9), 899–944 (2008)
14. Gerevini, A.E., Haslum, P., Long, D., Saetti, A., Dimopoulos, Y.: Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. Artificial Intelligence **173**(5-6), 619–668 (2009)
15. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: theory and practice. Elsevier (2004)
16. Gigante, N., Scala, E.: On the Compilability of Bounded Numeric Planning. In: IJCAI. pp. 5341–5349 (2023)
17. Helal, H., Lakemeyer, G.: An Analysis of the Decidability and Complexity of Numeric Additive Planning. In: ICAPS. pp. 267–275. AAAI Press (2024)
18. Hoffmann, J.: The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. J. Artif. Intell. Res. **20**, 291–341 (2003)
19. Kouaiti, A.E., Percassi, F., Saetti, A., McCluskey, T.L., Vallati, M.: PDDL+ Models for Deployable yet Effective Traffic Signal Optimisation. In: ICAPS. pp. 168–177. AAAI Press (2024)
20. Long, D., Fox, M.: The 3rd International Planning Competition: Results and Analysis. J. Artif. Intell. Res. **20**, 1–59 (2003)
21. Long, D., Fox, M., Hamdi, M.: Reformulation in Planning. In: Koenig, S., Holte, R.C. (eds.) SARA. vol. 2371, pp. 18–32 (2002)
22. McCluskey, T.L.: Combining Weak Learning Heuristics in General Problem Solvers. In: IJCAI. pp. 331–333 (1987)
23. McCluskey, T.L., Vaquero, T.S., Vallati, M.: Engineering knowledge for automated planning: Towards a notion of quality. In: Proceedings of the Knowledge Capture Conference, K-CAP. pp. 14:1–14:8 (2017)
24. Percassi, F., Scala, E., Vallati, M.: A Practical Approach to Discretised PDDL+ Problems by Translation to Numeric Planning. J. Artif. Intell. Res. **76**, 115–162 (2023)

25. Riddle, P.J., Barley, M., Franco, S.: Problem Reformulation as Meta-Level Search. In: Proc. of the Conference on Advances in Cognitive Systems. pp. 199–216 (2013)
26. Riddle, P.J., Holte, R.C., Barley, M.W.: Does Representation Matter in the Planning Competition? In: Proceedings of the Ninth Symposium on Abstraction, Reformulation, and Approximation, SARA (2011)
27. Scala, E.: Plan Repair for Resource Constrained Tasks via Numeric Macro Actions. In: ICAPS. vol. 24, pp. 280–288 (2014)
28. Shleyfman, A., Kuroiwa, R., Beck, J.C.: Symmetry Detection and Breaking in Linear Cost-Optimal Numeric Planning. In: ICAPS. pp. 393–401. AAAI Press (2023)
29. Taitler, A., Alford, R., Espasa, J., Behnke, G., Fišer, D., Gimelfarb, M., Pommerening, F., Sanner, S., Scala, E., Schreiber, D., Segovia-Aguas, J., Seipp, J.: The 2023 International Planning Competition. AI Mag. pp. 1–17 (2023)
30. Vallati, M., Chrpa, L., McCluskey, T.L., Hutter, F.: On the importance of domain model configuration for automated planning engines. J. Autom. Reason. **65**(6), 727–773 (2021)