

On the Relevance of Extracting Macro-operators with Non-adjacent Actions: Does It Matter?

Sandra Castellanos-Paez^a, Romain Rombourg and Philippe Lalanda
Univ. Grenoble Alpes, CNRS, Grenoble INP, France

Keywords: Automated Planning, Macro-operators, Learning, Data Mining.

Abstract: Understanding the role that plays the extraction phase on identifying potential macro candidates to augment a domain is critical. In this paper, we present a method to analyse the link between extracting macro-operators from non-adjacent actions and the correctness of (1) the frequency and (2) the number of occurrences per plan. We carried out experiments using our method on five benchmark domains and three different planners. We found that extracting macro-operators with only adjacent actions leads to important errors in macro-operator frequency and occurrences per plan.

1 INTRODUCTION


In AI planning, macros can model system routines. A *macro* consists of a sequence of actions that occurs frequently in solution plans. Once learned, macros can be re-injected directly into the planning domain. Thus, the domain benefits from the knowledge extracted from previous problem solving. The planning system applies a macro in the same way as a primitive action. However, macros allow jumping into the search space by building deep and promising states to reach a goal state. Learning macros from previously acquired knowledge has proven to be beneficial for improving a planner's performance (Botea et al., 2005b; Coles and Smith, 2007; Chrpa et al., 2014).

Macros have been widely studied to speed-up planning processes (Newton and Levine, 2010; Dulac et al., 2013). These approaches consist of two main phases: extraction and selection. Extraction consists in identifying, either by using previous knowledge or by *ad-hoc* construction, sequences of actions that could be potential candidates to augment the domain. The selection phase must find a trade-off between the benefit expected by adding macros and the additional cost induced by the branching factor increase.

Literature about macros presents various techniques to deal with the extraction phase, ranging from simple combination of primitive actions and the use of chunks of plans to the use of genetic learning algorithms or statistical analyses based on n-grams.

Understanding the impact of the extraction phase on the identification of potential macro candidates to augment a domain is critical. Indeed, through this phase, the infinite set of possible candidates to become macros passes through a first "filter". The aim is to make the set finite while keeping, among that set, the best possible macros to add to the domain in order for them to be identified and selected in the next phase. We are aware that the selection is the main concern when dealing with macro-operator research. However, lots of selection methods use information that can be acquired during the extraction phase (Botea et al., 2005a; Hofmann et al., 2017; Castellanos-Paez, 2019). For example, in (Botea et al., 2005a), they present the CA-ED method which ranks a given macro based on the number of plans where it occurs (hereafter called *support*) and the total number of times it occurs in each plan (hereafter called *number of occurrences*). It seems then crucial to assess the correctness of such information to obtain better results.

When extracting macro-operators from non-adjacent actions, the number of actions possibly ignored between two considered actions is called the gap. In this paper, we investigate if a finite gap could ensure in most cases that all macro occurrences are found. More precisely, we analyse the link between the use of a finite gap when extracting macro-operators and the correctness of (1) the support and (2) the number of occurrences per plan. By correctness we mean how close are the computed macro-operator characteristics (*i.e.* support and number of

^a  <https://orcid.org/0000-0002-6241-7974>

occurrences) to the ones computed when all macro-operator occurrences are found. Besides, we analyse the effect of the gap with plans obtained through three different planners by comparing extraction results with an infinite gap and a range of finite gaps.

The paper is structured as follows. First, we introduce the concepts of classical planning, macro-operators and some concepts borrowed from sequential pattern mining. Then, we present our method to evaluate the impact of a finite gap on the extraction of macro-operators. After, we present our results on five benchmark domains. Finally, we provide a discussion of the results and a conclusion.

2 BACKGROUND THEORY

We are interested in plan synthesis, a particular form of planning which takes a description of the world state, all its known actions and a goal (Ghallab et al., 2004). As a result, we get an organised set of actions whose execution makes it possible to solve the planning task. In this work, we address sequential planning in the STRIPS framework (Fikes and Nilsson, 1971).

A state is defined as a set of predicates. A planning task is composed of a *planning domain* and a *planning problem* and the purpose of this task is to find a *plan* to solve this problem. A planning domain describes the world through a set of predicates and a set of planning operators. A planning problem describes the initial state of the world and the goal state to be attained. A planning operator is a triple $o = (name(o), pre(o), effects(o))$ where its elements are defined as follows:

- $name(o)$ is in the form $name(x_1, \dots, x_n)$ where x_1, \dots, x_n are the object variable symbols that appear in o .
- $pre(o)$ is the set of predicates involving the variables of o and that must be satisfied when o is instantiated.
- $effects(o)$ is the set of literals (*i.e.* atoms and negation of atoms) involving the variables of o to be applied to a state when o is instantiated.

A grounded operator is an instance of a planning operator (*i.e.* a *lifted* operator for which variables are instantiated). A ground operator a is applicable in a state s if and only if all predicates in the preconditions of a belong to s . A state s' is reached from s if a grounded operator can be applied. Finally, a (solution) plan π is an ordered sequence of grounded operators to reach a goal state s_g from an initial state s_i .

Macros are based on the idea of composing a sequence of primitive operators and viewing the se-

quence as a single operator. For our purposes, we distinguish two related but different terms: grounded macros and lifted macros. A grounded macro is related to a lifted macro as a ground operator is related to a lifted operator. We use these terms according to literature common terminology.

In the following, we define a set of concepts (borrowed from Sequential Pattern Mining (Fournier-Viger et al., 2017; Han et al., 2011)) necessary for the understanding of this work.

A sequence database C is a set of pairs $\langle sid, s \rangle$, where sid is a sequence identifier and s is a sequence.

A sequence $S_A = X_1, X_2, \dots, X_k$, where X_1, \dots, X_k are ground operators, is a sub-sequence of another sequence $S_B = Y_1, Y_2, \dots, Y_m$, where Y_1, \dots, Y_m are ground operators, if and only if there exists integers $1 \leq e_1 < e_2 < \dots < e_k \leq m$ such that $X_1 = Y_{e_1}, X_2 = Y_{e_2}, \dots, X_k = Y_{e_k}$.

The absolute support of a sequence S is the number of sequences S_i , in the sequence database C , where S is a sub-sequence of S_i . The relative support of a sequence S is the absolute support of S divided by the total number of sequences in C . A frequent sequence is a sequence whose relative support satisfies a given relative support threshold. Thereafter, we will call this threshold *minsup*.

Besides, notice that a frequent sequence can occur in several other sequences but not necessarily in a contiguous fashion, as the definition of sub-sequence implies. So, we define a *gap* as the number of operators allowed between two consecutive operators of a sequence and it can take values in $[0, \infty]$. A sub-sequence $S = Y_{e_1}, Y_{e_2}, \dots, Y_{e_k}$ satisfies a gap constraint g if $\forall i \in [2, k], e_i - e_{i-1} \leq g + 1$.

3 EVALUATING THE IMPACT OF A FINITE GAP

In planning, each plan is composed of an ordered sequence of grounded operators which in turn are composed of parameters (objects). In Figure 1, we give an example of plan from the depots¹ domain, denoted π_1 .

Mining lifted macro-operators (in the following, macro-operators) from a set of plans requires an approach that ensures to find the frequent sequences of operators without a loss of information about their characteristics. The often expected characteristics for

¹Domain description: <https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume20/long03a-html/node38.html>
Domain PDDL definition: <https://www.cs.colostate.edu/meps/aips02data/depots/Strips/Depots.pddl>

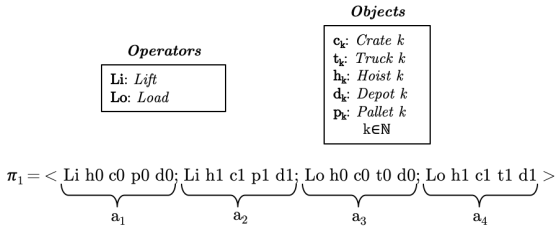


Figure 1: Example plan for depots domain.

each macro-operator m are : the support, *i.e.* the number of plans containing at least one occurrence of m , and the number of occurrences of m in each plan. The former represents a general view of the frequency of m . The latter represents a more detailed view of the use of m through each plan.

In (Castellanos-Paez et al., 2020), we highlighted the capital role of the gap in the extraction of macro-operators. In this paper, we analyse the loss of information when a finite gap is used to extract macro-operators. More precisely, we analyse the impact on the correctness of the characteristics computed for these macros (support, number of occurrences). Also, we investigate if there is a different impact on correctness when extracting macro-operators from optimal or sub-optimal plans.

By correctness we mean how close are the computed macro-operator characteristics (*i.e.* support and number of occurrences) to the ones computed when all macro-operator occurrences are found.

3.1 Overview of the Analysis Procedure

Our analysis focused on evaluating the impact of a finite gap on the correctness of the support and the number of occurrences. First, we extracted a reference set of frequent macro-operators, M_{ref} and their characteristics. This reference set is guaranteed to be correct since we used the ERA algorithm (Castellanos-Paez et al., 2020) with a *minsup* of 0.85, an infinite maximum length and an infinite gap. By correct, we mean that all possible occurrences were found, thus, the computed support and number of occurrences are exact. Indeed, the ERA algorithm produces exhaustive results with the aforementioned gap and length parameters.

Then, we extracted different sets of macro-operators by varying the gap parameter. Intuitively, using a finite gap implies that some occurrences may be missed by the algorithm. Therefore, a *minsup* of 0 was used to avoid losing information on the characteristics of macro-operators in the reference set.

Finally, we compared the results of the reference set (infinite gap) with the results of the sets obtained

by varying the gap parameter. The comparisons were made by using three different metrics.

Details on the extraction algorithm and the metrics definition are given in the following sections.

3.2 Extraction Algorithm: ERA

The ERA algorithm mines all macro-operators (regardless of their length or up to a maximum length) satisfying a frequency threshold and under a gap constraint from a set of solution plans. Additionally, for each macro-operator m , this algorithm yields the following characteristics:

- support [*integer*]: the number of plans containing at least one occurrence of m .
- sequence ids [*list*]: the plan identifiers where m appears.
- number of occurrences [*list*]: the number of occurrences of m in each plan.

The ERA algorithm checks, for each plan, every sub-sequence satisfying the gap constraint. If an infinite gap is used, the algorithm is then guaranteed to find all macro-operators, satisfying the frequency threshold, and all their occurrences. Also, macro-operator occurrences (even non-adjacent occurrences) registered by ERA are valid. Indeed, to be registered, an occurrence must be composed of grounded operators that can be moved contiguously in the analysed plan without impacting its final state or impeding its execution. Finally, if the macro-operator occurrences satisfy the frequency threshold, a macro-operator is built as a contiguous sequence of lifted operators.

To illustrate how ERA registers occurrences, let us consider the plan π_1 in Figure 1 and a gap parameter of 1. The set of all sub-sequences of π_1 satisfying the gap constraint of 1 is $\{(a_1, a_2), (a_2, a_3), (a_3, a_4), (a_1, a_3), (a_2, a_4)\}$. If we focus on the sub-sequence (a_1, a_3) , ERA will register it as a valid occurrence of the macro-operator *lift-load*². Indeed, the grounded operators composing that sub-sequence can be moved contiguously in the plan π_1 yielding the modified plan $\pi'_1 = \langle a_1, a_3, a_2, a_4 \rangle$, where π'_1 is executable and leads to the same final state as π_1 . Notice that gaps are only considered during the generation of sub-sequences to be analysed but not when building the macro-operator.

For more details about the mining and the building of macro-operators in the ERA algorithm, see (Castellanos-Paez et al., 2020).

²A hoist lifting a crate and loading it into a truck.

3.3 Metrics Definition

To compare the results of the reference set (infinite gap) with the results of the sets obtained by varying the gap parameter, we defined the following metrics:

The mean support error, Err_{sup} , defined in (1):

$$Err_{sup}(g) = \frac{1}{|M_{ref}|} \sum_{m \in M_{ref}} (s(m; \infty) - s(m; g)) \quad (1)$$

represents the mean difference across each macro-operator m in the reference set M_{ref} between the terms $s(m; \infty)$ and $s(m; g)$. The former represents the support obtained with an infinite gap for a macro m . The latter represents the supports obtained with a finite gap g for a macro m . The term $|M_{ref}|$ is the number of macros in the set M_{ref} .

The mean relative occurrence error, Err_{occ} , defined in (2):

$$Err_{occ}(g) = \frac{1}{|M_{ref}|} \sum_{m \in M_{ref}} \frac{o(m; \infty) - o(m; g)}{o(m; \infty)} \quad (2)$$

represents the mean relative difference across each macro-operator m in the reference set M_{ref} . In (2), $o(m; \infty)$ represents the total number of occurrences obtained with an infinite gap, and $o(m; g)$ represents the total number of occurrences obtained with a finite gap g .

Finally, the total relative occurrence error, Err_{tot} defined in (3):

$$Err_{tot}(g) = \frac{\sum_{m \in M_{ref}} (o(m; \infty) - o(m; g))}{\sum_{m \in M_{ref}} o(m; \infty)} \quad (3)$$

represents the fraction of occurrences missed when using a finite gap g .

4 RESULTS

In the following, we show the impact of a finite gap on the characteristics of the extracted macro-operators. For this purpose, we used five benchmark domains, two sub-optimal planners and one optimal planner. Keep in mind that this study does not address planner performance or macro quality. The planner choices were then solely based on the optimality or sub-optimality of the plans produced.

4.1 Experimental Setup

To the best of our knowledge, no open plan database is available. Thus, for each benchmark domain, we generated a set of 35 distinct problem instances using the

generators³ from the International Planning Competition. Then, we solved the problems by using each of the three planners (Helmert, 2006; Richter and Westphal, 2014; Helmert and Domshlak, 2011). As a result, we obtained a total of fifteen sets of solution plans (each with 35 plans) divided as follows: ten sets of sub-optimal plans (5 domains times 2 planners) and five sets of optimal plans (5 domains times one planner). Indeed, we also investigated the behaviour of the loss of information when extracting macro-operators from optimal or sub-optimal plans.

The analysis procedure was carried out on each one of the fifteen sets of solution plans and the impact of a finite gap was evaluated for gap values ranging from 0 to 15. The average plan length was about 30 for the sets of sub-optimal plans and was about 25 for the sets of optimal plans. Experiments were performed on dedicated machine with an Intel Core i7-4710MQ quad-core CPU clocked at 2.5GHz and with 8GB of RAM. Table 1 presents the different parameters tested.

Table 1: Parameters of the gap evaluation experiment.

Parameter	Values tested
Gap	0 - 15
Planner	FastDownward (A* and FF heuristic) LAMA LM-Cut
Domain	Barman, Blockworld, Depots, Rover, Satellite

4.2 Results of the Impact of a Finite Gap

In Figures 2, 3 and 4, for each domain and for each planner, we present respectively: Err_{sup} , Err_{occ} and Err_{tot} , as a function of the gap. As expected, we observed that these errors decreased consistently with the gap for all domains and planners.

We observed across almost all domains that the errors were less important when using plans obtained with the LM-Cut planner. However, between FD(A*+FF) and LAMA, there was not a planner that presented consistently higher errors. Compared to the other domains, blockworld showed significantly lower errors.

Additionally, we present in Table 2, for each domain and planner, the errors obtained with a gap value of 0. And, in Table 3 we show the minimal gap value required, for all domains and errors, to have an error value below 5% for all planners.

³<https://bitbucket.org/planning-tools/pddl-generators>

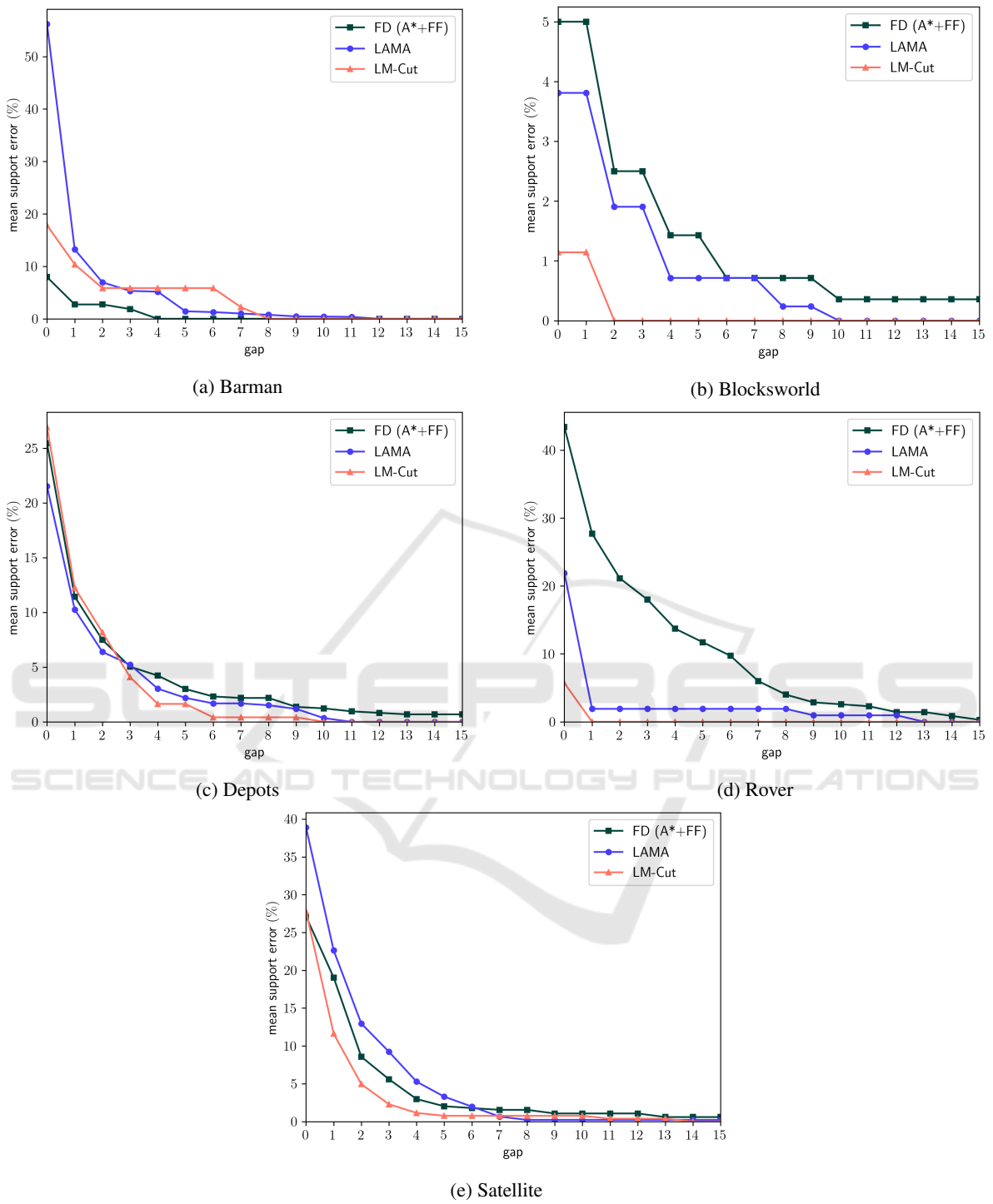
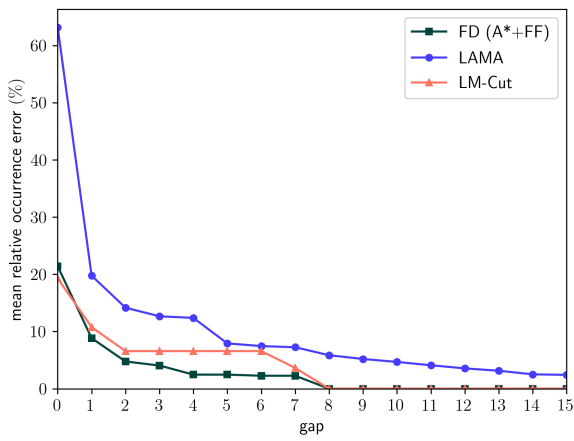
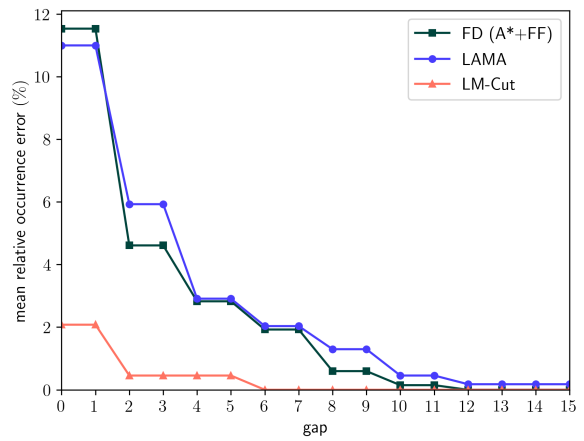


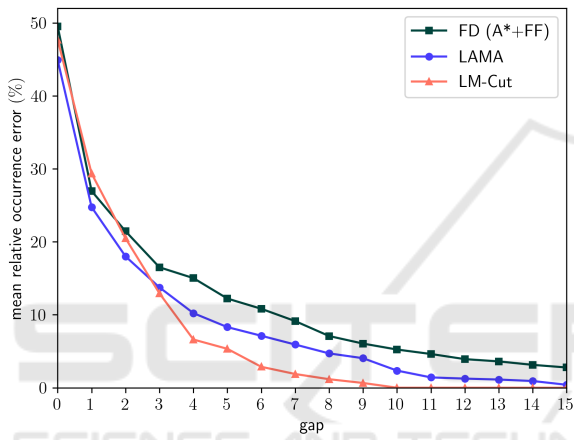
Figure 2: Mean support error. Errors are given in percentage of the plan set.



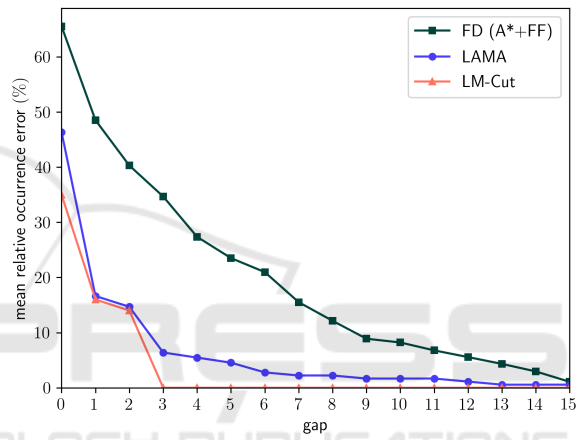
(a) Barman



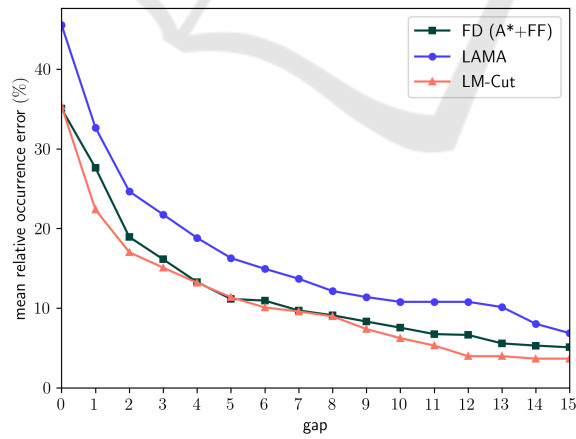
(b) Blocksworld



(c) Depots

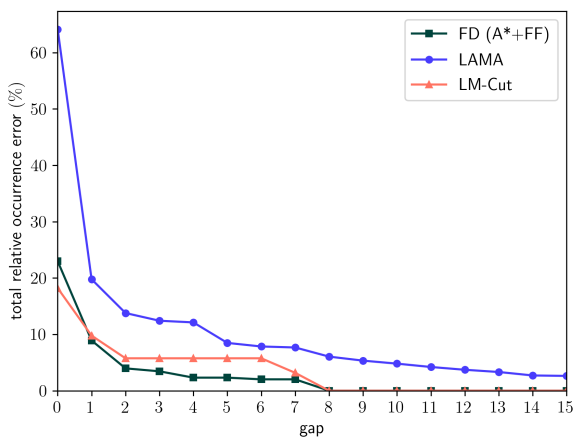


(d) Rover

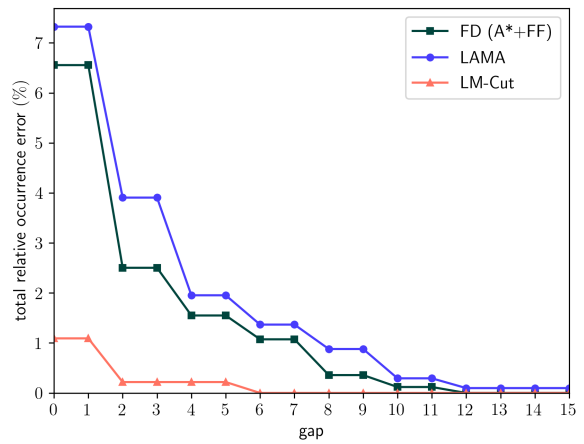


(e) Satellite

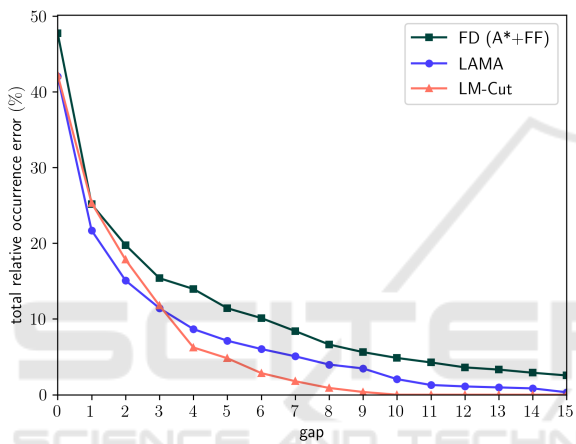
Figure 3: Mean relative occurrence error.



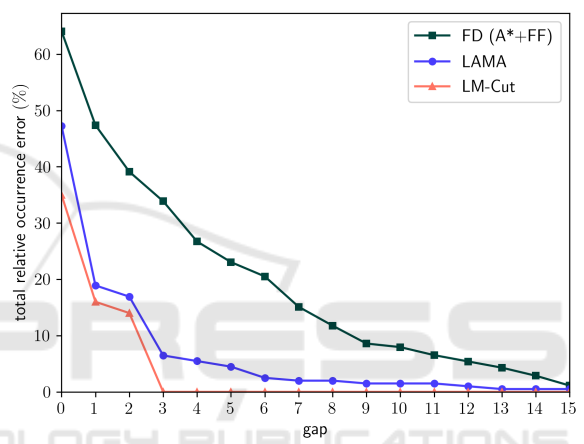
(a) Barman



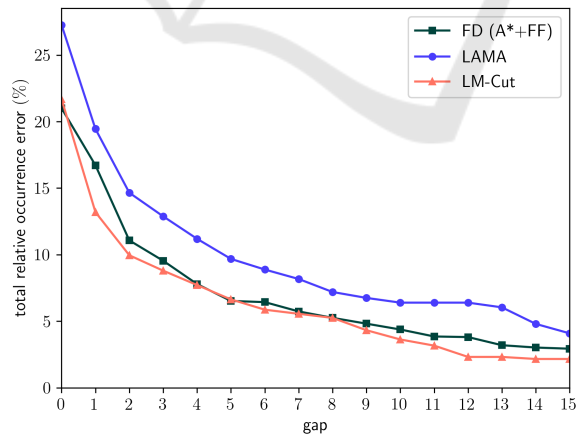
(b) Blocksworld



(c) Depots



(d) Rover



(e) Satellite

Figure 4: Total relative occurrence error.

Table 2: Errors for each domain and planner when using a gap value of 0 (extraction made only from contiguous actions).

	Err_{sup} (%)			Err_{occ} (%)			Err_{tot} (%)		
	FD	LAMA	LM-Cut	FD	LAMA	LM-Cut	FD	LAMA	LM-Cut
Barman	7.9	56.2	17.8	21.4	63.2	19.3	22.9	64.1	18.2
Blocksworld	5	3.8	1.1	11.5	11	2.1	6.5	7.3	1.1
Depots	25.4	21.5	26.9	49.5	44.9	47.7	47.7	42	42.2
Rover	43.4	21.9	5.7	65.5	46.4	35	64.1	47.3	35
Satellite	27.1	38.9	27.8	35	45.6	35.4	21	27.3	21.7

Table 3: Minimal gap value required to have an error value below 5% for all planners. '-' indicates that the error was still above 5% for a gap of 15.

	Err_{sup}			Err_{occ}			Err_{tot}		
	FD	LAMA	LM-Cut	FD	LAMA	LM-Cut	FD	LAMA	LM-Cut
Barman	1	5	7	2	10	7	2	10	7
Blocksworld	0	0	0	2	4	0	2	2	0
Depots	4	4	3	11	8	6	10	8	5
Rover	8	1	1	13	6	3	13	5	3
Satellite	4	5	11	-	-	12	9	14	9

5 DISCUSSION AND RELATED WORKS

Our results clearly show that extracting macro-operators by only considering adjacent groups of actions (zero gap) leads to high errors (often above 50%!) in terms of support and occurrences for every domain and planner. Besides, errors are similar for optimal and sub-optimal plans. Intuitively, one could think that high gap values would not impact significantly the extraction results since two distant operators should be less related than two close operators. Indeed, that would be true if the planning system was focused on accomplishing sequentially disconnected groups of sub-goals, e.g. preparing and serving a specific cocktail in barman. However, every plan (even optimal) can be reordered in hundreds or even thousands of different ways from which only a few correspond to a configuration where all disconnected goals are accomplished sequentially.

An infinite gap is required to guarantee correctness of the support and the number of occurrences. However, we showed that a finite and relatively small gap value (compared to the plan length) yields acceptable results. We also observed higher gap requirements for the `satellite` and `depots` domains. It could be explained by the fact that problems from these domains often consist in several disconnected sub-goals e.g. take a specific picture with any satellite or bring a given crate to a given truck.

One could point out that using non-adjacent actions to generate macro-operators leads to the extraction of more macros therefore possibly making the selection problem more difficult. However, should we

consider that with gap, more useless macros are extracted? Or, should we rather consider that without gap, the most useful macros could be missed? We believe in the second proposition, the extraction should never take the risk to rule out possible useful macros.

There are a number of works handling non-adjacent operators to build macros. Botea et al. (Botea et al., 2005b) extract macros from solution graphs of training problems. They enumerate and select sub-graphs from the solution graphs and build one macro for each selected sub-graph. They introduce a k parameter as the maximal number of operators that can be skipped between the first and the last element of the sub-graph (we can roughly see a sub-graph as a sub-sequence). Their handling of the gap does not allow an exhaustive search and can then miss many macro-operator occurrences. Also, they set a hard limit on the upper bound of the maximal length of the extracted macro-operators. Chrupa et al. (Chrupa et al., 2014) extract macros by iteratively combining operators (even non-adjacent) in plans that share some parameters while keeping the plan valid. However, their technique is not guaranteed to find all macro-operator occurrences. In Castellanos-Paez et al. (Castellanos-Paez et al., 2020), we present a pattern mining inspired algorithm to mine lifted macro-operators directly from a set of plans. Besides, this algorithm can detect the occurrence of a macro-operator even if the actions composing it are not adjacent.

To the best of our knowledge, our work is the only one evaluating the impact of extracting macro-operators from a set of plans by considering non-adjacent operators.

6 CONCLUSIONS

Macro-operators for automated planning are of great importance to improve performance by using past experiences. Understanding the role that plays the extraction phase on identifying potential macro candidates to augment a domain is critical. In this paper, we did not address planner performance or macro quality, however, we answered the question: "Does considering non-adjacent actions when extracting macro-operators matter?". The answer is yes, and a relatively small gap of 5 to 10 actions can go a long way in reducing the error in support and number of occurrences below five percent. The reader can clearly take away from this study that when dealing with macro-operators, we should not only look at adjacent operators. For future work, we believe that the consideration of these results could be advantageous when designing new macro-learning methods.

REFERENCES

- Botea, A., Enzenberger, M., Müller, M., and Schaeffer, J. (2005a). 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. (2005b). Learning partial-order macros from solutions. In *Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling*, pages 231–240. AAAI Press.
- Castellanos-Paez, S. (2019). *Learning routines for sequential decision-making*. PhD thesis, Université Grenoble Alpes.
- Castellanos-Paez, S., Rombourg, R., and Lalanda, P. (2020). ERA: extracting planning macro-operators from adjacent and non-adjacent sequences. In *Pacific Rim Knowledge Acquisition Workshop, PKAW 2020*. Springer.
- Chrapa, L., Vallati, M., and McCluskey, T. L. (2014). MUM: A technique for maximising the utility of macro-operators by constrained generation and use. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*.
- Coles, A. and Smith, A. (2007). Marvin: A heuristic search planner with online macro-action learning. *Journal of Artificial Intelligence Research*, 28:119–156.
- Dulac, A., Pellier, D., Fiorino, H., and Janiszek, D. (2013). Learning useful macro-actions for planning with n-grams. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 803–810.
- Fikes, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 3-4(2):189–208.
- Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., and Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77.
- Ghallab, M., Nau, D., and Traverso, P. (2004). Automated planning: theory and practice.
- Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.
- Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.
- Helmert, M. and Domshlak, C. (2011). Lm-cut: Optimal planning with the landmark-cut heuristic. *Seventh international planning competition (IPC 2011), deterministic part*, pages 103–105.
- Hofmann, T., Niemueller, T., and Lakemeyer, G. (2017). Initial results on generating macro actions from a plan database for planning on autonomous mobile robots. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*.
- Newton, M. A. H. and Levine, J. (2010). Implicit learning of macro-actions for planning. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*.
- Richter, S. and Westphal, M. (2014). The LAMA planner: Guiding cost-based anytime planning with landmarks. *CoRR*, abs/1401.3839.