Learning routines for sequential decision-making

Sandra Castellanos-Paez

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA
COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Sandra Milena CASTELLANOS-PAEZ

Thèse dirigée par **Sylvie PESTY,** Professeur, UGA
et **Damien PELLIER,** MCF, UGA
préparée au sein du **Laboratoire d'Informatique de Grenoble**
dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Apprentissage de routines pour la prise de décision séquentielle

Learning routines for sequential decision-making

Thèse soutenue publiquement le **24 octobre 2019**,
devant le jury composé de :

Madame Sylvie PESTY
PROFESSEUR, UNIVERSITE GRENOBLE ALPES, Directrice de thèse
Monsieur René MANDIAU
PROFESSEUR, UNIV. POLYTECHNIQUE DES HAUTS-DE-FRANCE, Rapporteur
Monsieur François CHARPILLET
DIRECTEUR DE RECHERCHE, INRIA CENTRE NANCY GRAND EST, Rapporteur
Monsieur Damien PELLIER
MAITRE DE CONFERENCES, UNIVERSITE GRENOBLE ALPES, Co-directeur de thèse
Monsieur Philippe MATHIEU
PROFESSEUR, UNIVERSITE DE LILLE, Président

Acknowledgements

I would like to thank the many people who have been involved in this adventure.

First of all, I would like to thank my supervisors for giving me the opportunity to do this thesis. More particularly, to Sylvie, for having given me back the motivation by reminding me that a thesis must not change the world but must above all contribute to advancing the understanding of a subject. To Damien, for introducing me to the tortuous world of automated planning and for sharing his knowledge that allowed me to find ways to do this work. I would also like to thank Humbert who shared his ideas at meetings.

I would also like to thank the members of the jury: René Mandiau and François Charpillet, for agreeing to be rapporteurs for this work by making relevant remarks that allowed me to better orient my oral presentation. Thanks also to Philippe Mathieu who chaired this jury with great attention.

I would like to warmly thank the past and present members of the MAGMA team, now HAWAI, who have listened to me and provided both human and scientific support during these years: Yves, Julie, Carole, Julius, Djalil, Ankuj, Ying, Manon, Luba, Omar, Galateia, Rémi and for the newcomers: Rouba and Thibauld, good luck, you'll see it's worth it!

It is with great nostalgia that I thank the members of the GETALP team, the first team that welcomed me during my master's studies. And in particular, I would like to thank Christian Boitet and Carlos Ramish who have passed on their passion for research and the academic world to me. I still have a pinch in my heart that I couldn't follow you, but I hope our paths will cross again.

Speaking of the academic world, I would like to thank the educational teams of Grenoble IAE and IM2AG who welcomed me for the realisation of my ATERs. I would also like to thank the administrative staff of the laboratory and the doctoral school.

I have met many people during these years of thesis that it is difficult to thank them all. However, I would like to thank Pier who has accompanied me well through the ups and downs and whose support has always been of great importance; Marc, Françoise, Gérard and Suzanne for having welcomed me with humour and love into your family; my colombian friends for your unfailing support: Sergio V. (amigazo!), Karen and Paola; my friends met at the Rabot and with whom I shared and still share many laughs and wild game Sundays: Eymeric, Pipo, Zoggy, Claire, Cassandra, Priss,...; my friends PPGA for the shared evenings that allowed me to decompress between two writing sessions of my thesis; thanks also to Raquel, Nicolas H., Emmanuel, Camilo M., Monsieur Caillat, Monique, Kenza and her family, Ali F. and to everyone who has marked this stage of my life.

I would like to express my love and gratitude to my parents who, even on another continent, have always encouraged and pushed me to move forward with tender messages every day and without fail and for whom my admiration has no limits. Finally, I would like to thank my brother for his unconditional support and for always having the right

words to say even if they are not the ones I want to hear (and which sometimes even sound like bullying).

I couldn't forget my faithful four-legged companion. Thank you Nask for waking me up in the morning and forcing me out of bed even when the motivation was not there. For all the moments of frustration that were overshadowed by your tender gaze and your multiple licks. To get me moving in the long days of writing by running behind you to get my slipper. Thanks also to Ada for accompanying me during the first years of my thesis and for finding a way to make me relax by asking for hugs.

My last and greatest thanks go to my love who I had the joy of meeting while I was still struggling with my endless experiments. Our scientific discussions have given me stars in the eyes, showing me that a passion for research can move mountains. Your enthusiasm has been key in motivating me to continue to pursue my dreams. Thank you for always being at my side during the nights of writing, the weekends of experimentation and even during the holidays never taken. Thank you for all the love you have for me and for always taking care of me when I need it most. For you, I'll keep spinning counterclockwise[1]...

[1] https://xkcd.com/162/

Contents

List of Figures

List of Tables

List of Algorithms

List of PDDL-Codes

List of Symbols and Acronyms

Hereafter is the list of symbols and notations introduced in the specified chapters.

Chapter 2

Symbol	Meaning
s_0	initial state
s_g	goal state
Σ	planning domain
$\mathscr{P}$	planning problem
s	state
S	subset of the set of all states
$\mathscr{L}$	representation language
a	action
A	set of all actions
γ	state-transition function
g	goal
o	operator
$name(o)$	name of operator o
$name(x_1,\ldots,x_n)$	operator *name* with object variables $x_1,\ldots,x_n$
$pre(o)$	set of preconditions of operator o
effects(o)	set of predicates of operator o to be applied to a state
$pre(a)$	set of preconditions of action a
$add(a)$	set of positive effects of action a
$del(a)$	set of negative effects of action a
π	sequences of actions or plan
$\langle a_1,\ldots,a_n\rangle$	sequences of actions a_1 through a_n
$A \cup B$	union of sets A and B
$A - B$	set of elements in A but not in B
$h(s)$	heuristic function
$h^*(s)$	true distance between s and the goal state

Chapter 3

Symbol	Meaning
$minsup$	minimum support threshold
P	set of patterns
sid	sequence identifier
σ	absolute support
$sup(x)$	relative support

Chapter 4

Symbol	Meaning
p_i	problem i
π_{id}	plan identifier
π_i	solution plan for a problem p_i
D	sequence database
m	macro-action
$name(m)$	name of macro-action m
$name(c_1,\ldots,c_n)$	macro-action *name* with object constants $c_1,\ldots,c_n$
$pre(m)$	set of preconditions of macro-action m
effects(m)	set of predicates of macro-action m to be applied to a state
G_T	Planning time metric
G_N	Space size metric
G_Q	Plan Quality metric

Chapter 5

Symbol	Meaning
m	macro-operator
S	sequence of actions
$add(m)$	set of positive effects of macro-operator m
$del(m)$	set of negative effects of macro-operator m
$p_1 \mid p_2$	predicate p_1 is incompatible with predicate p_2
d	planning domain
M	set of macro-operators
p	predicate
L_n	n^{th} layer of the incompatibility graph
A	set of instantiated operators or set of actions
$x \frown y$	node x is linked to node y
$L_n \backslash p$	equivalent to $L_n - \{p\}$
C	set of compatible predicates
I	set of potential incompatible predicates
$\|C\|$	number of elements in set C

Chapter 6

Symbol	Meaning
$< a, b >$	pair a, b
e_i	i^{th} element in set of solution plans
$minsup$	minimum support threshold
$maxLength$	maximal extraction length
M	set of macro-operators
C	set of solution plans
$\overline{G}(M;C)$	estimated mean node gain by adding M and with respect to C
M_{opt}	optimal macro-operator set with respect to $\overline{G}$

Hereafter the list of acronyms used in this thesis. They are listed using the order of their first apparition in the manuscript.

Acronym	Meaning
AI	Artificial Intelligence
NP	Nondeterministic Polynomial time
METEOR	Macro-operator Extraction, Trade-off Estimation and Optimisation from plan Recycling
AP	Automated Planning
PDDL	Planning Domain Definition Language
IPC	International Planning Competition
STRIPS	Stanford Research Institute Problem Solver
GPS	General Problem Solver
PSPACE	Polynomial space
SAT	Planning as satisfiability
CP	Constraint Programming
IP	Integer Programming
CSP	Constraint Satisfaction Problem
HSP	Heuristic Search Planner
FF planner	Fast Forward planner
NOAH	Nets of Actions Hierarchies
UCPOP	Universal quantification, Conditional effects Partial Order Planner
SHOP	Simple Hierarchical Ordered Planner
LM	Landmark
SAS	Simplified Action Structures
HTN	Hierarchical Task Network
ID	Identifier
FP	Frequent Pattern
CP	Closed frequent pattern
SDB	Sequence Database
FSP	Frequent Sequential patterns
FAST	Fast sequence mining Algorithm based on Sparse id-lisTs
GSP	Generalized Sequential Patterns
LAPIN	LAst Position INduction sequential pattern mining algorithm

Acronym	Meaning
SPADE	Sequential PAttern Discovery using Equivalence classes
SPAM	Sequential PAttern Mining algorithm
BIDE	BI-Directional Extension algorithm
CloFAST	Closed FAST sequence mining algorithm
ClaSP	Closed Sequential Patterns algorithm
VMSP	Vertical mining of Maximal Sequential Patterns
MaxSP	Maximal Sequential Pattern miner
ERA	Extraction of Rich patterns with Attribute structures

1

Introduction

Efforts and courage are not enough without purpose and direction.

John F. Kennedy

1.1 Context of research

This thesis presents results achieved during my PhD at Grenoble Alps University and Grenoble Informatics Laboratory. It is part of the Artificial Intelligence field.

The field of artificial intelligence (AI) intends to build intelligent agents. The intelligence of these agents is concerned with rational action. Given a situation, they can act

to achieve the best outcome. Thus, as stated by Russell and Norvig (2010), an intelligent agent is a system that can decide what to do and then do it.

An intelligent system perceives its environment, reasons about the best possible action corresponding to the current situation, and performs it. Therefore, the primary concern of AI has been the design of intelligent systems to perform complex tasks without any human intervention. As an example of a domain that benefits from applying intelligent systems, we have the automotive industry with the autonomous vehicles (Urmson et al., 2008). One of its potential benefits concerns safety. They are expected to significantly reduce accidents since the human error factor will be no longer exists (Fagnant and Kockelman, 2015).

From this application, one expects that intelligent systems make decisions as fast as possible in a reliable way. Thus, developing intelligent systems should include decision-making autonomy and capacity to learn from past experiences. Indeed, they can use their experiences to improve their performance. For example, faced with a situation already encountered, they will use the previously acquired knowledge to decide better and more effectively.

Because the applications tend to be complex, the decision-making component is essential, and it could not work without continuous advances in effective algorithms. They are still many techniques to explore in this way. We focus this thesis on routine learning for sequential decision making.

1.1.1 Automated Planning

Intelligent systems to be genuinely efficient need to organise their actions as fast as possible in a reliable way. Planning is then essential since it is a careful consideration process by which actions are chosen to achieve a specific goal. It is needed to understand the problem and to adapt the resources to attain an objective as best as possible.

Automated planning (AI planning) is a sub-field of AI that aims to study and design domain-independent general approaches to planning (Ghallab et al., 2004). In AI planning, a planning task consists of a planning domain and a planning problem. The former consist of a description of the world and a set of actions. The latter consist of an initial world situation and some objective to achieve. Thus, the automated planning community devises effective algorithms that produce action sequences (namely, a plan) to reach a planning task goal, from an initial state, for a potential execution by one or several agents.

Solving planning problems is a time-consuming and challenging process because algorithms must understand planning tasks without the use of domain-specific knowledge. To this must also be added the NP-hard complexity of planning. Namely, the time required to solve a planning problem increases very quickly as the size of the problem grows.

Therefore, it is essential to develop robust algorithms. These algorithms must efficiently explore the search space that grows exponentially with the plan length, which is unknown. Various approaches have been studied to enhance the efficiency of planning (Kautz et al., 1992; Van Den Briel and Kambhampati, 2005; Lopez and Bacchus, 2003; Blum and Furst, 1997; Geffner and Haslum, 2000). These are a result of the cross-fertilisation of ideas from different AI areas.

In planning, search algorithms usually explore a graph trying to find a sequence of actions from a given initial node n_0 to a goal node n_g (Bonet and Geffner, 2001; Nguyen and Kambhampati, 2001). To counter the exponential growth, search algorithms can use knowledge about the problem. This thesis focus on the study of learning macros, i.e. frequently encountered routines.

1.1.2 Learning Macros

Few planning approaches take advantage of previous problems solved to accelerate the search for a new problem. However, intuitively, a system capable of exploiting its experience, should be able to achieve better performance. Then, the main idea of learning macros is to capture specific knowledge from analysing learning examples.

A *macro* consist of a sequence of actions that occur frequently in solution plans. Take for example the `blocksworld` planning domain, it consists of a set of blocks settled on a table. The goal is to build one or more vertical stacks of blocks. We can move only one block at a time to perform one of the following actions (see Figure 1.1):

- Pick it up from the table.
- Put it down on the table.
- Unstack it from another block.
- Stack it on another block.

Intuitively, we can suggest that once we pick up a block from the table, the next most probable action will be to stack it on another block or to put it down. If we are trying to change the state of the world, putting down a block that we just picked up may be useless. After solving many times different `blocksworld` problems, we may observe more occurrences for the pick up action followed by the stack action than followed by the put down action. From this experience, we can deduce that pick up and stack actions build a potential macro. Thus, the next time we solve a problem, we can apply the *pick up and stack* macro directly, avoiding the analysis of what action comes after the pick up action.

From this example, we can observe that macros can model system routines. Once learned, they can be re-injected directly into the planning domain. Thus, the domain will benefit from the knowledge extracted from previous problems. The system applies

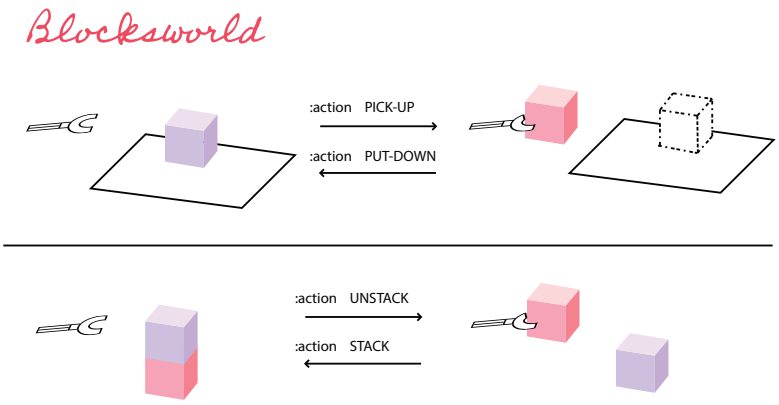


Figure 1.1: `blocksworld` actions

a macro in the same way that a primitive action. However, macros allow jumping into the search space by building deep and promising states to reach a goal state.

On the one hand, Figure 1.2 presents the search space in a `blocksworld` domain, for a set of actions, an initial state and a goal. On the other hand, Figure 1.3 presents the search space for an enhanced domain. Now, the set of actions also includes the macro *pick up and stack*. This example clearly shows that the shortcuts provided by the macro allow going deep quickly into the search space.

Macros has been widely studied among the different approaches to speed-up planning processes (Botea et al., 2005a; Coles and Smith, 2007; Newton and Levine, 2010; Dulac et al., 2013; Chrpa et al., 2014; Asai and Fukunaga, 2015). Macros literature presents various techniques to build them, ranging from a simple matter of combining primitive actions and the use of chunks of plans to the use of genetic learning algorithms or statistical analyses based on n-grams.

In these approaches, there are two main phases: extraction and selection. The *extraction* consists in identifying sequences of actions that could be potential candidates to enhance the domain. However, the main disadvantage of macros is to increase the branching factor of the search space. Indeed, by adding macros, the system must consider primitive actions as well as new macros (See Figure 1.3).

Therefore, the use of macros raises a utility issue. The *selection* phase must found a trade-off between the benefit expected from adding macros and the additional cost induced by the branching factor increase. The selection phase plays a vital role because carefully selected macros can significantly improve performance by reducing the depth of the search space.

For approaches extracting macros from past experiences, an assumption often used is that frequent sequences of actions are potentially good candidates to enhance the domain. Botea et al. (2004) use in their work this assumption when filtering the macros to

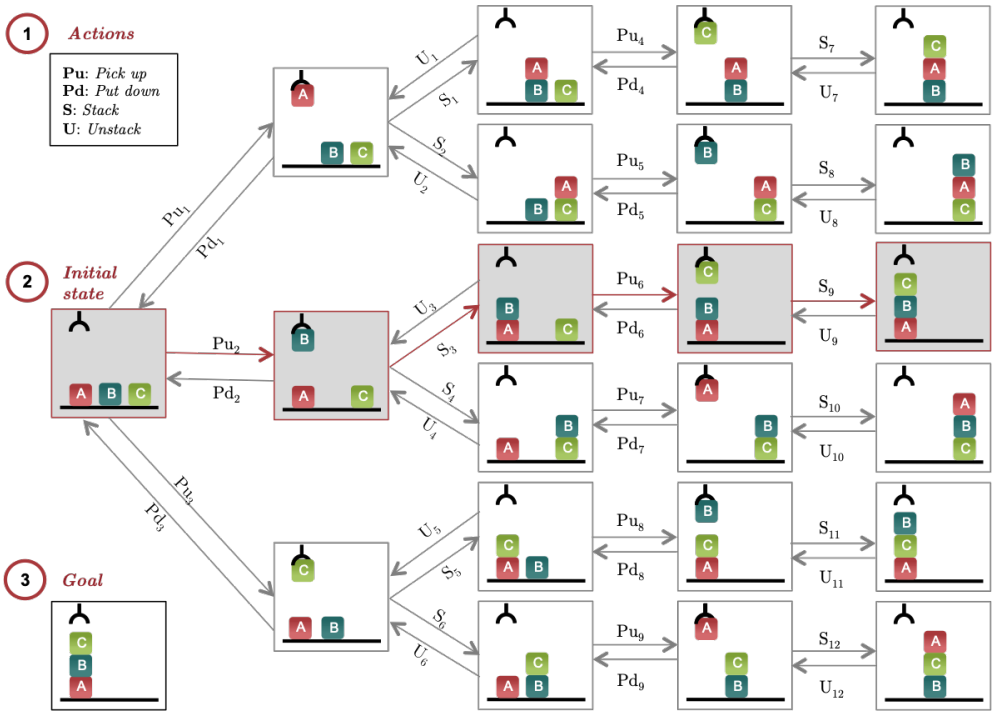


Figure 1.2: `blocksworld` domain with only primitive actions. Filled squares represent the sequence of actions to achieve the goal from the initial state.

be added to the domain. For them, the more often a macro is present in the solutions of past problems, the higher its weight will be. Only the two best macros will be part of the domain. Dulac et al. (2013) create a macros library from the most frequent action sequences derived from an n-gram analysis on successful plans previously computed by the planner.

We could, therefore, consider an approach that exploits this hypothesis — for example, the *pattern mining* technique from the field of data mining which aims to discover frequent patterns in data.

1.1.3 Pattern mining

The purpose of data mining is to look for patterns by searching automatically in data stored electronically. The descriptive task of data mining is called *Pattern mining*. Pattern mining intends to discover interesting and useful patterns in data.

Pattern mining has become popular because of its applications in multiple domains. Although the different pattern mining techniques are aimed at analysing data, techniques such as itemset mining and association rule mining do not take into account the sequen-

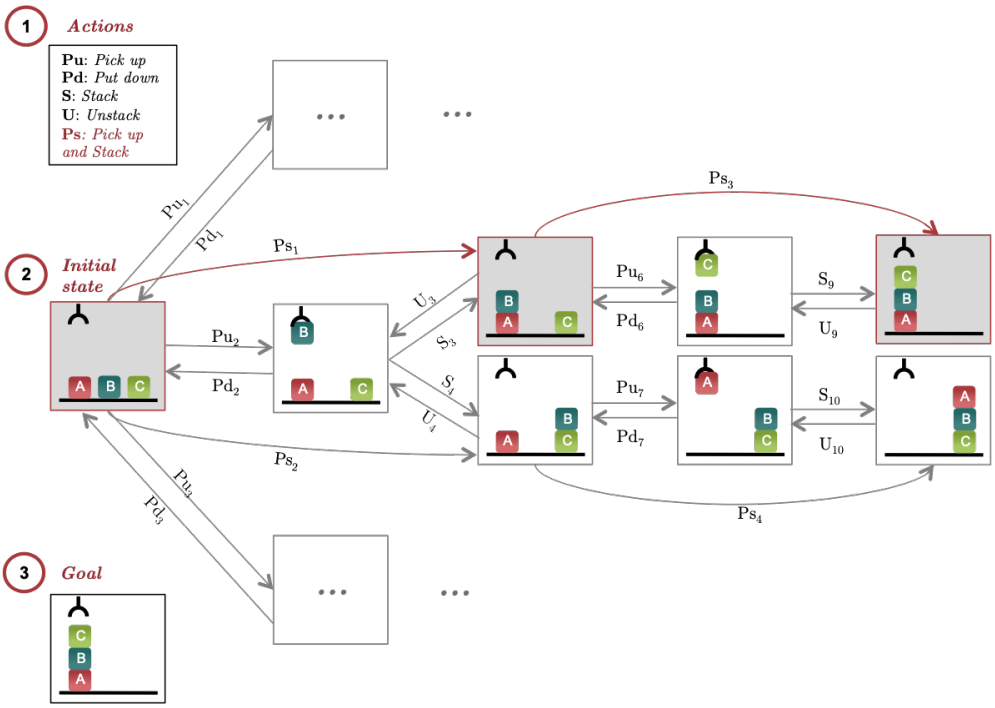


Figure 1.3: `blocksworld` enhanced domain with the macro *pick up and stack*. Filled squares represent the sequence of actions to achieve the goal from the initial state.

tial ordering of events. Therefore, there exists a technique for mining sequence data called *sequential pattern mining*.

Sequential pattern mining consists in analyse sequential data to discover frequent sequential patterns. We can distinguish two filter structures provided by sequential pattern mining. On the one hand, there is a parameter called *support*, which filters patterns based on the frequency of apparition. On the other hand, from following some restrictions, the resulting pattern set can be reduced.

1.2 Problem

Routines are present in real-life applications or closely related systems. These routines can be used to improve the performance of the solving process of new problems. One way to build on past experiences is to learn macros.

In automated planning, the challenge remains on developing powerful planning tech-

niques capable of effectively explore the search space that grows exponentially. Learning macros from previously acquired knowledge has proven to be beneficial for improving a planner's performance (Botea et al., 2004; Chrpa et al., 2014).

In this work, we address the two stages of learning routines:

1. The extraction via the exploration of pattern mining techniques. This approach should be domain-independent and particularly adapted to the extraction of recurrent patterns (i.e. the routines). Besides, we want to exploit the extraction of sequences of non-adjacent actions. Only a few works have explored this path (Botea et al., 2004; Chrpa et al., 2014). However, they have shown that this allows more routines to be extracted and therefore, would be more profitable for the system.

2. The selection via *a priori* macro utility. Among studied approaches, they select macros, either based on a frequency-based ranking (Botea et al., 2005b; Chrpa, 2010) or based on *a posteriori* evaluation (Botea et al., 2004; Newton and Levine, 2010; Hofmann et al., 2017). However, the benefits of a macro depend also on several other factors such as its length, i.e. the number of search nodes that a macro would save (Botea et al., 2005a); its impact on the branching factor; the other macros used (the purposes of two macros may overlap).

This thesis studies the following research questions:

Research question #1

Is there a monotonic relationship between the frequency of apparition of a macro and its utility? i.e. can the frequency alone be used as an estimator for macro ranking by utility?

Research question #2

Can we learn routines from past experiences that are not only frequent but above all useful?

1.3 Contributions

This thesis contributes mainly to the field of automated planning, and it is more specifically related to learning macros for classical planning. We focused on developing a domain-independent learning framework that identifies sequences of actions (even non-adjacent) from past solution plans and selects the most useful routines (i.e. macros), based on *a priori* evaluation, to enhance the planning domain.

Below, we highlight the advantages of our work:

- *Planner independent*, we do not need to modify the planner' structure.
- *Domain-independent*, the framework does not need a priori knowledge on the domain.
- *Non-adjacent actions*, we can identify routines from a sequence of adjacent and non-adjacent actions.
- *A priori evaluation*, there is no need to re-solve past problems to select the most useful routines.
- *An optimal set of routines*, we can identify not only useful routines but the optimal set to enhance a domain.

First, we studied the possibility of using sequential pattern mining for extracting frequent sequences of actions from past solution plans, and the link between the frequency of a macro and its utility. For that, we proposed a framework to extract macro-actions (i.e. sequences of actions with constant objects, see Figure 1.4) via sequential pattern mining algorithms and to select useful macro-actions based on their frequency (Castellanos-Paez et al., 2016). We found out that the frequency alone may not provide a consistent selection of useful macro-actions.

Additionally, we found some discrepancies in the results of the precedent study. To explore them, we transposed the study to macro-operators (i.e. sequences of actions with variable objects, see Figure 1.5) and we proposed a new approach to validate the generated macros. This approach proved to be successful in eliminating problematic macro-operators.

We discussed the problems of using classic pattern mining algorithms in planning. Despite the efforts, we find ourselves in a dead-end with the selection process because the pattern mining filtering structures are not adapted to planning. We concluded in the need for a novel approach allowing to extract macro-operators and assess in their utility.

Finally, we provided a novel approach called METEOR, which ensures to find the frequent sequences of operators from a set of plans without a loss of information about their characteristics. This framework was conceived for mining macro-operators from past solution plans, and for selecting the optimal set of macro-operators that maximises the node gain. It has proven to successfully mine macro-operators of different lengths for different domains and thanks to the selection phase, be able to deliver a positive impact on the search time without drastically decreasing the quality of the plans.

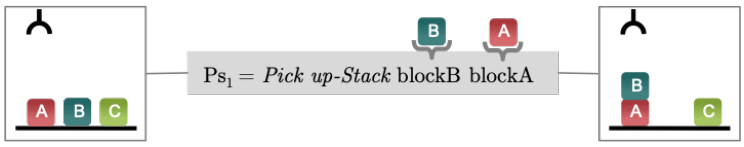


Figure 1.4: A macro-action *pick up and stack* for specific objects blockB and blockA.

Figure 1.5: A macro-operator *pick up and stack* for variable block objects.

1.3.1 Publications

Some of the work presented in this thesis has been subject to related publications:

Sandra Castellanos-Paez, Damien Pellier, Humbert Fiorino, Sylvie Pesty. "Learning Macro-actions for State-Space Planning". In: *Journées Francophones sur la Planification, la Décision et l'Apprentissage*, Jul 2016, Grenoble, France.

Sandra Castellanos-Paez, Damien Pellier, Humbert Fiorino, Sylvie Pesty. "Mining useful Macro-actions in Planning". In: *The third International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, IEEE, 2016. p. 1-6.

1.4 Outline of the dissertation

This thesis is structured in two parts (see Figure 1.6). The first part consists of theoretical background on topics covered in this work, followed by a context presentation of our main results, to finally, put them in perspective by reviewing the associated relevant literature. We organised this part into two chapters.

In Chapter 2, we introduce the background planning concepts used in the development of this work and the literature related to these concepts. First, we intend to establish the scope of work through the definition of classical planning concepts. Next, we present an overview of the approaches to speed-up planning processes and its corresponding literature. Finally, we review and analyse the relevant literature around macro learning methods in planning.

In Chapter 3, we provide some theoretical building blocks of pattern mining, and we put these concepts in the context of this work. First, we introduce the basic concepts of pattern mining. Next, we focus on the concepts of mining frequent patterns, and we describe the most popular algorithm for pattern mining. Finally, we focus on the sequential pattern mining approach since it is the most related to our data types.

The second part of this thesis presents our contribution, taking into account the research questions previously presented. We structured this part in three chapters.

In Chapter 4, we explore (1) the use of sequential pattern mining for learning useful macro-actions from past solution plans and (2) the link between the frequency of a macro and its utility (see the research question #1). With this goal, we propose a framework to learn useful sequences of actions (not necessarily adjacent) as macro-actions and use them to speed-up planning search. We based this learning framework on the filter structures provided by sequential pattern mining.

In Chapter 5, we detail two shortcomings found in the last chapter, namely the lack of (1) macro-actions generality and (2) verification of the validity of the generated macro-operators. For each shortcoming, we first discuss its implications and then, we detail a remedial measure to address it. At the end of the chapter, we discuss the difficulty of the selection process, even after using shortcoming remedial measures.

In Chapter 6, we discuss the encoding limitations of traditional pattern mining algorithms in the extraction of macro-operators. To answer the research question #2, we then present our METEOR framework, which (1) mine macro-operators from past solution plans and (2) select the optimal macro-operator set to enhance the planning domain. We validate the proposed framework in known planning benchmarks.

Finally, in Chapter 7, we review the contributions of the thesis, we give concluding remarks and we propose possible directions for future work.

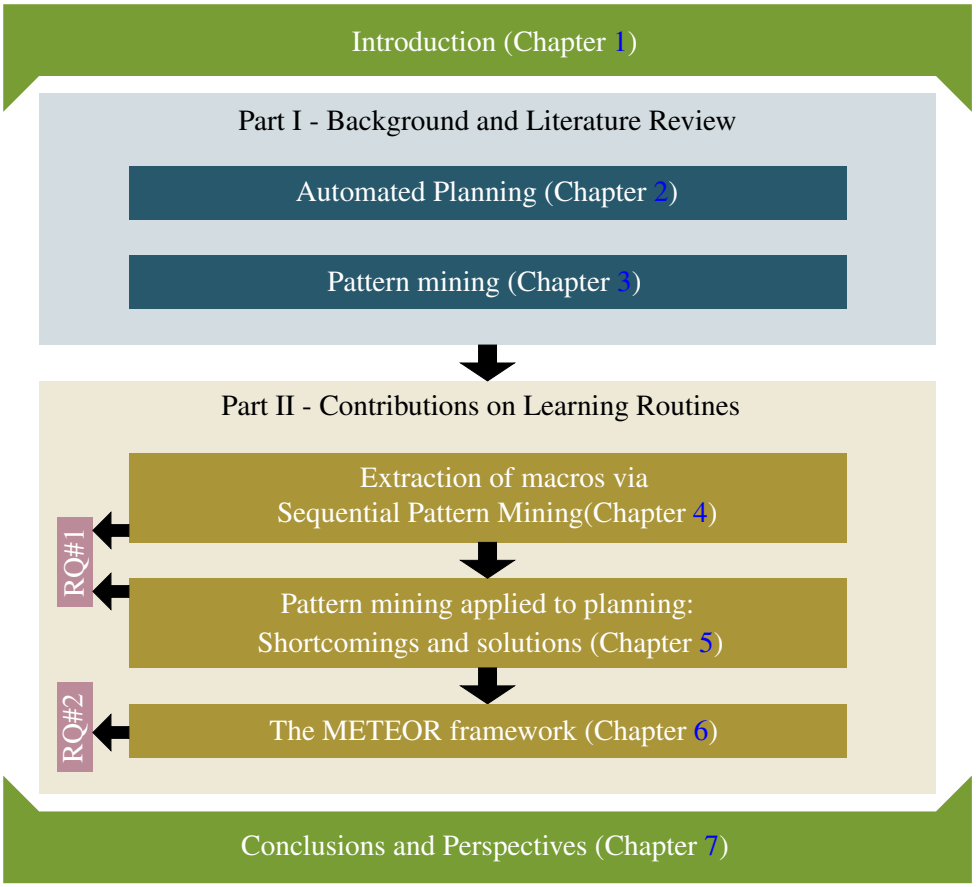


Figure 1.6: Thesis outline

Part I

Background and Literature Review

2

Automated Planning

A goal without a plan is just a wish.

Antoine de Saint-Exupéry

2.1 Introduction

In this work, we are interested in plan synthesis. It is a particular form of planning which takes a description of the world state, all its known actions and a goal. As a result, we get an organised set of actions whose execution makes it possible to solve the planning task.

Some systems are renown worldwide for their exceptional ability to solve very specific problems. Among many others, Deepblue (Campbell et al., 2002) and Alpha Go (Silver et al., 2016) show that a very efficient approach to deal with planning problems is to create predictive models adapted to the specific representations of a problem.

However, this is not suitable to build autonomous intelligent systems since their deliberative capabilities will be limited to the areas enclosed in their domain-specific planners. Thus, a better strategy consists in building a planning engine based on a domain-independent general approach.

Therefore, Automated Planning purpose is to develop a general approach that solves any problem described in a representation language, using a general algorithm. AP approaches rely on the model of state-transition systems since it describes a general model for a dynamic system.

2.2 Classical Planning

In an effort to develop well-founded approaches, a more practical model introducing several assumptions was defined. **Classical planning** stands for planning for restricted state-transition systems. Classical planning problems are well-formalised and well-characterised considering that their model obeys to the following assumptions :

- Finite: The state-transition system has a finite set of states.
- Fully observable: There exists a complete knowledge about the system.
- Deterministic: From a state, the application of an action brings to a single other state.
- Static: The system stays in the same state until an action is applied.
- Restricted Goals: The objective of the system is to find a sequence of states that

ends in a state satisfying all goals.

- Sequential plans: The solution plan consists in an ordered sequence of actions.
- Implicit time: Actions do not have any duration.

In order to provide a better understanding of the planning concepts through this chapter, we will use, as a basis for the examples, a well-known classical planning domain: the blocksworld domain.

Blocksworld domain

Blocksworld in Figure 2.1 is a toy problem that consists of a set of blocks settled on a table and a mechanical hand. The hand can move one block at a time to perform one of the following actions: place it on another block, place it on the table, pick it from the table or removes it from another block. The goal is to build one or more vertical stacks of blocks.

From this description, in Table 2.1 we define a series of non-independent predicates and four possible actions.

Predicates
• on(b,b'): Block b is on some block b'.
• ontable(b): Block b is on the table.
• clear(b): No block sits on top of block b.
• handempty: Hand is not holding a block.
• holding(b): Hand is holding block b.
Actions
• Pick-up a block b from the table.
• Put-down a block b on the table.
• Stack, to put a block b on top of a block b'.
• Unstack, to remove a block b from a block b'.

Table 2.1: Definition of predicates and actions for blocksworld domain

2.3 PDDL representation language

The representation language used in this work, and one of the languages used in automated planning, is called PDDL. PDDL stands for Planning Domain Definition Language (McDermott et al., 1998; McDermott, 2000). It was introduced in 1998 for the International Planning Competition with the aim of standardising the planning representation language. On top of that PDDL allowed a meaningful comparison of planners on different problems.

```
(define (problem blocksworld3)
    (:domain blocksworld)
    (:objects blockA blockB blockC -block)
    (:init (handempty)
           (on blockA blockB)
           (ontable blockB)
           (ontable blockC)
           (clear blockC)
           (clear blockA))
    (:goal (on blockB blockA)))
```

PDDL-Code 2.1: Definition of a blocksworld problem

As an example, we define the blocksworld domain in PDDL-Code 2.2 and the blocksworld problem of Figure 2.1 in PDDL-Code 2.1 by using PDDL. The former is composed of *predicates* which characterize the properties of the objects and a set of non-instantiated *actions* (later called operators) which establish the ways to move from one state to another. The latter is composed of *objects* which define the task relevant things in the world; an initial state s_0 which represents the starting configuration of the world; and a goal state s_g which describes the desired predicates that we want to be true.

(a) Initial state (b) Goal state

Figure 2.1: Representation of Code 2.1

```
(define (domain BLOCKS)
  (:requirements :strips :typing)
  (:types block)
  (:predicates (on ?x - block ?y - block)
         (ontable ?x - block)
         (clear ?x - block)
         (handempty)
         (holding ?x - block)
         )

  (:action pick-up
       :parameters (?x - block)
       :precondition (and (clear ?x) (ontable ?x) (
   handempty))
       :effect
       (and (not (ontable ?x))
       (not (clear ?x))
       (not (handempty))
       (holding ?x)))
  (:action put-down
       :parameters (?x - block)
       :precondition (holding ?x)
       :effect
       (and (not (holding ?x))
       (clear ?x)
       (handempty)
       (ontable ?x)))
  (:action stack
       :parameters (?x - block ?y - block)
       :precondition (and (holding ?x) (clear ?y))
       :effect
       (and (not (holding ?x))
       (not (clear ?y))
       (clear ?x)
       (handempty)
       (on ?x ?y)))
  (:action unstack
       :parameters (?x - block ?y - block)
       :precondition (and (on ?x ?y) (clear ?x) (
   handempty))
       :effect
       (and (holding ?x)
       (clear ?y)
       (not (clear ?x))
       (not (handempty))
       (not (on ?x ?y)))))
```


PDDL-Code 2.2: Definition of the `blocksworld` domain

2.4 Key concepts

In this section, we intend to present the formal definition of the planning key concepts used in this work.

Because the interest of planning lies in choosing actions to transform the system state, the transitions between states are represented with a state-transition system model. Additionally, we address sequential planning in the STRIPS framework (Fikes and Nilsson, 1971).

Definition 2.1.
A planning task consists of a planning domain Σ and a planning problem $\mathscr{P}$.

Example The planning task composed by the blocksworld domain in PDDL-Code 2.2 and the blocksworld problem of Figure 2.1 in PDDL-Code 2.1. ■

Definition 2.2.
A classical planning domain is a restricted state-transition system $\Sigma = (S, A, \gamma)$ such that:

- S is included in the set of all states that can be described with the representation language $\mathscr{L}$.
- A is the set of all actions a.
- $\gamma(s, a)$ is the state-transition function that defines the transition from a state s to an state s' using an action a.

Definition 2.3.
A classical planning problem $\mathscr{P} = (\Sigma, s_0, g)$ is composed of:

- s_0 an initial state where $s_0 \in S$.
- g a goal, namely a set of instantiated predicates. A goal is satisfied if the system attains a state s_g such that all predicates in g are in s_g.

Definition 2.4.
A state s is a set of logical propositions.

Example Let us consider the initial state from the blocksworld problem in PDDL-Code 2.3. Here, every predicate represents a proposition that can take a true or a false value. ■

```
(:init  (handempty)
        (on  blockA  blockB)
        (ontable  blockB)
        (ontable  blockC)
        (clear  blockC)
        (clear  blockA))
```

PDDL-Code 2.3: Initial blocksworld state represented in Figure 2.1(a)

Definition 2.5.
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 $name(x_1,...,x_n)$ where $x_1,...,x_n$ are the object variable symbols that appear in o.
- $pre(o)$ is the set of precondition formula that must be hold before exploiting the action.
- *effects(o)* is the set of predicates to be applied to a state.

Example In the *pick-up* operator in Code 2.4, the object variable symbols are defined in the `:parameters` clause and the preconditions (resp. effects) are defined in its `:precondition` (resp. `:effect`) clause. ■

```
(:action pick-up
        :parameters (?x - block)
        :precondition (and (clear ?x) (ontable ?x) (
   handempty))
        :effect
        (and
          //negative effects
        (not (ontable ?x))
        (not (clear ?x))
        (not (handempty))
        //positive effects
        (holding ?x)))
```

PDDL-Code 2.4: PDDL definition for *pick-up* blocksworld operator

Definition 2.6.
An action a is an instantiation of a planning operator. Thus, a is a triple $a = (pre(a), add(a), del(a))$. If an action can be applied, a new state is generated. First it deletes all instantiated pred-

icates given in the delete list $del(a)$, also known as the negative effects. Then, it adds all instantiated predicates given in the add-list $add(a)$, also known as the positive effects.

Example In Code 2.5, we have instantiated the pick-up operator with a blockC. ■

```
(:action pick-up
       :parameters (blockC)
       :precondition (and (clear blockC) (ontable
   blockC) (handempty))
       :effect
       (and
         //negative effects
       (not (ontable blockC))
       (not (clear blockC))
       (not (handempty))
       //positive effects
       (holding blockC)))
```

PDDL-Code 2.5: Instantiation of *pick-up* blocksworld operator

Definition 2.7.
A state s' is reached from s by applying an action a according to the transition function in formula 2.1.

$$s' = \gamma(s,a) = (s - del(a)) \cup add(a). \tag{2.1}$$

The application of a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ to a state s is recursively defined in Formula 2.2.

$$\gamma(s, \langle a_1, \ldots, a_n \rangle) = \gamma(\gamma(s,a_1), \langle a_2, \ldots, a_n \rangle). \tag{2.2}$$

Example In Figure 2.2, we represented the application of the pick-up action shown in Code2.5 to a state s in order to obtain a state s'. ■

Definition 2.8.
A plan is an ordered sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ such that $s_g = \gamma(s_i, \pi)$ satisfies the goal g and the latter is reachable if such a plan exists.

Example From the initial state in Figure 2.1(a), the plan satisfying the goal state in Figure 2.1(b) is $\pi = \langle$ `pick-up blockA, put-down blockA, pick-up blockB, stack blockB blockA` $\rangle$. ■

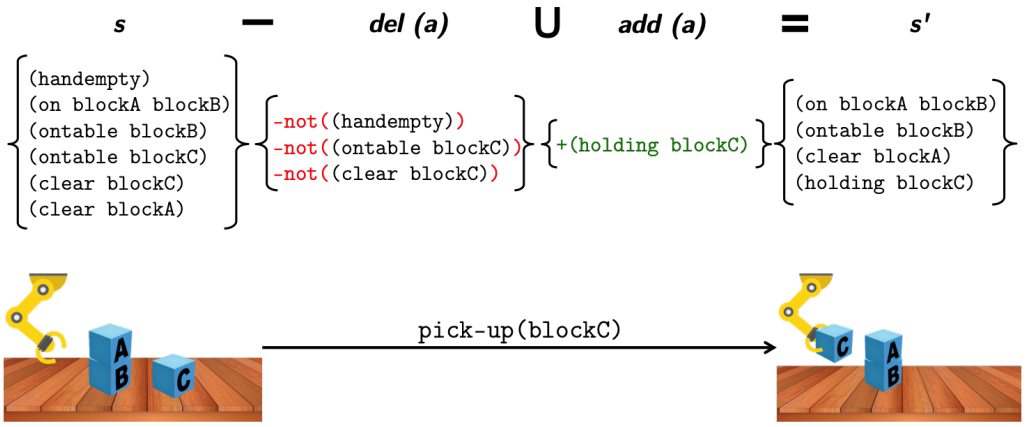


Figure 2.2: Pickup action applied to a state s

2.5 Development of Automated Planning

Under this section, we gather methods and concepts related to AP problems. We present their chronological apparition in Table 2.2. Although, some methods presented here are quite old, they are still not outdated. Also, some of these concepts may be partially or integrally combined with each other .

General Paradigms

Early works in AI for problem solving and search introduced the *weak methods*. Namely, basic search methods that have a high generality. These methods do not need neither a defined domain nor a precisely determined initial point. Paradoxical though it may seem, they can use highly specific knowledge of the domain but only for specific functions. Take the case of an evaluation function where that knowledge is used only to compare states and select the best.

Among these methods, we found the *means-ends analysis* (Newell et al., 1959) that uses knowledge about how operators reduce the difference between states and goals. The first AI system to implement this method to solve multiple problems was the General Problem Solver (Newell, 1963). GPS organises the information about the difference between objects into goals. If the goal is to transform an object into another object, the objects are compared and a subgoal is set up to reduce the difference (if any). If the goal is to reduce the difference between two objects, the program looks for an operator reducing this difference and a subgoal is set up to apply the operator to the object. Finally, if the goal is to apply an operator, the program validates if the conditions of the operator are satisfied and then it generates the new object. Otherwise a subgoal is created to reduce this difference.

Table 2.2: Timeline of classical planning approaches.

1963	General Problem Solver.
1971	Stanford Research Institute Problem Solver.
1972	MACROP generation UCPOP partial order planning algorithm.
1975	Nets Of Actions Hierarchies.
1977	Hierarchic Non-linear Planner.
1984	O-PLAN: Open planning architecture.
1985	PRODIGY.
1992	SATPLAN PRIAR.
1994	UMCP IxTeT.
1995	Graphplan.
1996	UNPOP.
1999	Heuristic search planner SHOP.
2000	GP-CSP CSP-PLAN.
2001	Fast-Forward Planner RePOP.
2005	MacroFF Optiplan.
2007	Marvin.
2014	MUM.
2017	BFWS.

We also found, the generate-and-test method. A generator is a process that takes information specifying a set and from it produces possible candidates for solution. Then, these elements pass one by one to the test process. It determines whether some condition holds for that candidate and decides its behaviour based on the needs of the rest of the processing.

A variation of this method is *hill-climbing*. Here, the generation of the next state depends on a feedback from the test process. It includes a heuristic function that provides a way to move only to states that are better than the current state, throwing out old states as it moves uphill.

Approaches to solve classical planning problems

Planning problems have been shown PSPACE-complete. To tackle this challenge, planning systems try to reduce the size of the search space they have to explore. Various approaches have been studied to enhance the efficiency of planning. These are a result of the cross-fertilisation of ideas from different AI areas.

2.5.1 Translation into another problems

A well-known approach is to translate the planning problem into another kind of combinatorial problem such as SAT, CP or IP. Then, solve this translated problem using an already existing efficient solver and finally, take the solution and translate it into a plan.

Planning problem as satisfiability problem

A SAT problem is a satisfiability problem, i.e., determining whether a Boolean formula can be true for some assignments of its variables. This formula is created from simple propositions or propositional variables which are associated using OR, NOT or AND connectors.

SAT planning approach defines the planning problem as a set of postulates where any model of the postulates corresponds to a valid plan. The initial state and the goal state are described as a set of propositions holding respectively at time 0 and at time N, where N is an horizon length. And, the possible transitions are represented one-to-one with the models of a propositional formula. The combination of these elements represents the formula to be satisfied. And any assignment of truth values achieving it, is a valid plan for the planning problem.

In SATPLAN (Kautz et al., 1992),the planning problem comes from a constructed planning-graph of some length N. SATPLAN encode all constraints of it into a formula

which is built as a conjunction of one or more clauses, where a clause is a disjunction of literals. It solves that formula using an efficient SAT solver. The planner increases N and starts again if no solution was found at that length. Considering that N increases in a step-wise manner, in terms of solution length, the output of this planner will necessarily be an optimal plan.

Planning problem as integer programming

An IP problem is defined as an optimisation problem, i.e., finding an assignment of values that maximises (or minimises) a cost function. In this kind of problem, some or all the variables are restricted to be integers.

IP planning approach consists in casting the planning problem as the minimisation or maximisation of a linear function. This function is created with integer-valued variables and is conditioned to linear equality and inequality constraints in the variables.

In Optiplan (Van Den Briel and Kambhampati, 2005), a planning graph is built and transformed into an IP problem. This planner only considers the actions and propositions instantiated in the planning graph. The IP problem is solved using a IP solver such as ILOG CPLEX (ILOG, 2002). If no plan is found, the process starts again by extending the planning graph by one step.

Planning problem as constraint programming

CP approach models the problem as a constraint satisfaction problem (CSP). A CSP consists of a set of variables $X = \{x_1, ..., x_n\}$, a set of domains $D = \{D_1, ..., D_n\}$ and a set of constraints $C = \{c_1, ..., c_k\}$. For each variable $x_i \in X$ there is a domain D_i and the scope of each constraint is a subset of X.

Constraint satisfaction searches for a compatible assignment of values to the variables that doesn't violate the constraints. This approach encodes the potential solution plans of length K as a CSP problem.

GP-CSP (Do and Kambhampati, 2000) and CSP-Plan (Lopez and Bacchus, 2003) were based on translation of the planning graph to a CSP. More recently, Barták proposed a novel view of contraint-based planning that used parallel plans and multi-valued state variables (Barták, 2011).

2.5.2 Search for planning

A planning problem can be solved by searching a solution in a search space. To accomplish that, the problem model must be translated into a search space and a search

algorithm must be chosen. In the last twenty years heuristic based approach to guide the search became popular.

Search algorithms explore a graph trying to find a sequence of actions from a given initial node n_0 to a goal node n_g. They are described by the search space that they explore, its search direction and if they have additional information to guide the search.

Search in the space of states

Each node corresponds to a state of the dynamic system, each arc corresponds to a state transition which is a result of an executed action on a state, and the plan correspond to the found path in the search space.

Based on this search space several planners were developed such as STRIPS (Fikes and Nilsson, 1971), HSP (Bonet and Geffner, 2001), FF Planner (Hoffmann and Nebel, 2001) and MacroFF (Botea et al., 2005a).

Search in the space of plans

In this search space, the state-transition system is not considered anymore. Nodes are partially specified plans and arcs are plan operations intended to complete a partial plan. The initial node corresponds to an empty plan and the goal node contains a solution plan that satisfies the required goals. A solution plan is a set of planning operators with ordering constraints and binding constraints.

Planners such as NOAH (Sacerdoti, 1975), UCPOP (Penberthy et al., 1992), IxTET (Laborie and Ghallab, 1995) and RePOP (Nguyen and Kambhampati, 2001) exploit this search space.

Search in the space of task networks

The input to the planning system consists in a set of operators and a set of methods. A method describes how to decompose some task into some set of subtasks. Searching in the space of task networks aims to perform some set of tasks instead of achieving a goal. To do that, the compounded tasks are decomposed recursively into smaller tasks, until primitive tasks are reached that can be performed using the operators. Then, the solution is an executable sequence of primitive tasks.

Planners in such search space provide a more available way to write problems for human domain experts. Among these planners, we have NONLIN (Tate, 1977), O-PLAN (Currie and Tate, 1991), UMCP (Erol et al., 1994), SHOP (Nau et al., 1999) and SHOP2 (Nau et al., 2003).

■ Planning graph search

The output is a sequence of set of actions being more general than a sequence of actions from the state-space planners but less general than a partial order from the plan-space planners. Planning graph provides a way to estimate the set of propositions reachable from the initial state s_0 and the actions leading to them.

It is a directed layered graph composed of two kinds of layers, action layers A_{i+1} and proposition layers P_i where $0 \leq i \leq n$. Layer at level 0 of the graph consists in the set P_0 of propositions describing the initial state s_0 of the planning problem.

- A_{i+1} is the set of actions whose preconditions are nodes in P_i
- P_{i+1} is the union of P_0 and the set of positive effects of actions in A_{i+1}

Graphplan planner (Blum and Furst, 1997) introduced this approach to reduce the amount of search needed to find the solution, improving considerably the performance over state-of-the-art planners of its time.

2.5.3 Techniques to improve planning search

Searching the path from a start node to a goal node is an important aspect of solving planning problems because this technique allows to find the most viable path to reach the goal and make the process efficient. Thus looking for ways to improve this technique is crucial to improve planner performance.

Basic search, also called blind search is an uninformed search. The search does not have additional information about states except from that provided in the definition of the problem. As a result, the total search space can potentially be explored and its exploration is exhaustive using brute-force algorithms. Based on this approach we found algorithms such as Breadth-First search and Depth-First search, they represent the state space in form of a tree where the initial state, the intermediate states and the goal states are nodes of the tree.

- Breadth-First search, the root node is expanded first, then all its successors are expanded and for each next step all successors of every node are expanded successively until a goal state is reached.
- Dept-First search, one branch of the tree is explored until the solution is found. The searches ends when a dead end is met or when the process becomes longer than the time limit. In that case, the process starts again with another branch of the tree to be explored.

A way to improve the search is adding additional information about the problem to guide the search in a specific direction.

2.5.3.1 Heuristic search

Using a heuristic function to estimate better choices during search has been shown to be a major progress in planning and the most common form to impart additional knowledge to the search algorithm.

Definition 2.9.
A heuristic function $h(s)$ is an evaluation expression that defines some criteria to rate the cost of an intermediate state s to a goal state.

Definition 2.10.
A heuristic $h(s)$ is admissible if it never overestimates the cost of reach the goal. Given that $h^*(s)$ is the true cost to reach the goal from a state s, we have $h(s) \leq h^*(s)$

Definition 2.11.
For every state s and every successor s' of s obtained by applying an action a, a heuristic is consistent if the estimated cost of reaching the goal from s is less than the sum of the cost of getting s' plus the cost of reaching the goal from s' $h(s) \leq c(s,a,s') + h(s')$

We will use the classification proposed by Torralba Arias de Reyna (2015) that groups heuristics into five families: Critical-paths, relaxation, abstraction, landmark and flow-based heuristics.

Critical paths

h^m heuristics introduced by Geffner and Haslum (2000) estimate the cost from a state to a goal by computing the maximal cost from that state to any sub-goals of length at most m. Thus the path from that state to the goal is at least as costly as the path leading to the most costly sub-goal. As m grows the computational complexity to calculate h^m grows exponentially because the number of sub-goals of length at most m increases as $\sum_{k \leq m} \binom{|G|}{k}$. Haslum et al. (2005) extended these heuristics to the additive h^m heuristics. Additive h^m computes partial $h^m_{A_i}$ where $\{A_i\}$ is a partition of A, the set of the actions in a planning problem P. $h^m_{A_i}$ is calculated in the same fashion as h^m except that the cost of every action not in A_i is relaxed. Finally, the $h^m_{A_i}$ are summed for all A_i to obtain the heuristic value.

Relaxation

h^+ heuristics (Hoffmann and Nebel, 2001) estimate the cost from a state to the goal as the cost of an explicit plan π calculated without considering the negative effects of actions. As the computation of h^+ is NP-hard, other approximations such as h^{add} and h^{max}

were introduced by Bonet and Geffner (2001), to calculate heuristics in polynomial-time. Respectively, one heuristic approximates h^+ by assuming that every fact in a conjunctive sub-goal must be achieved separately while the other heuristic assumes that achieving a single fact from a conjunctive sub-goal is sufficient. Unlike, h^+, h^{add} and h^{max} heuristics, h^{FF} (Hoffmann and Nebel, 2001) solves the relaxed problem by finding some relaxed plan whose is not necessarily optimal. $h^{FF}(s)$ is calculated as the length of that relaxed plan. Other approximations of h^+ take into account some negative effects (Helmert, 2006; Helmert and Geffner, 2008). Moreover, only h^+, max, landmark-cut (Helmert and Domshlak, 2009) and improved LM-Cut (Bonet and Helmert, 2010) heuristics are admissible.

Abstractions

h^α heuristics simplify the planning task by mapping to an abstract state space $\mathscr{S}^\alpha$ the original state space $\mathscr{S}$ by the means of an abstraction function α. This function α defines the states that should be characterised. Then $h^\alpha(s)$ for state s is the cost estimation of the cheapest path from the abstract state $\alpha(s)$ to the goal state in $\mathscr{S}^\alpha$.

There exists different abstractions classes to do different mappings of the search space. In pattern databases heuristic (Culberson and Schaeffer, 1998; Edelkamp, 2001; Haslum et al., 2007), the abstraction function α is a projection. It maps two states s_1 and s_2 to the same abstract state if and only if they agree on all variables in the pattern. A pattern P is the set of state variables. Merge-and-shrink heuristic (Helmert et al., 2007; Dräger et al., 2009) adds new variables to the abstraction using a merging strategy and then, it reduces the abstract space by fusing some abstract states following a shrinking strategy.

Landmarks

h^{LM} heuristics introduced by Porteous et al. (2014), are based on the definition of a landmark, namely a property (a fact or an action) that every plan must satisfy for the planning task. Obtaining landmarks is usually done prior to planning. Then $h^{LM}(s)$ is the minimal cost of the landmark actions for s meaning the set of actions that were found to happen in every plan. In other words, since all landmark actions are bound to happen at some point in the plan the heuristic ensures its admissibility by defining the distance to the goal as the minimal cost from those that can be applied for s and zero if no such action can be applied. The main difficulty is to find those landmarks. This technique was then improved by the introduction of the landmark cut heuristic (Helmert and Domshlak, 2009) and later the improved LM-cut (Bonet and Helmert, 2010).

Network Flows

h^{flow} heuristics (Van Den Briel et al., 2007) estimate the cost from a state s to the goal by using the flow constraint equation in a SAS+ formalism. This equation ensures the balance between the number of times that a given atom (or proposition) is produced versus the number of times that it is consumed to satisfy the goal state, and that for every atom. Linear programming is used to solve the flow equation for the number of times that each operator is used with the minimal cost. The idea is to produce a group of actions that, if they all could be applied to s would yield the goal state.

2.5.3.2 Learning strategy

Instead of trying to estimate the distance to the goal with a general function, one can use previously solved problems to increase the performance of planning systems. For instance, it is possible to learn commonly used actions or groups of actions or even adapt a heuristic function to a specific domain.

Practical planning systems require domain knowledge and control knowledge. The former describes the world and the available actions whereas the latter prescribes how the planner must to behave to attain its goals.

It cannot be denied that adding knowledge results in better planning system performance. Injecting knowledge can be time-consuming if done by human experts. A solution is that the planner automatically learns it.

Learning domain knowledge

- **Learning action preconditions and effects**: In classical planning, the basic idea (Wang, 1995) is to learn action preconditions by raising the propositions from a set of pre-states s_{i-1}. These pre-states are the states before action a_i is applied. On the other hand, learn action effects can be done by raising the difference between the propositions in the pre-states s_{i-1} and the post-states s_i, namely the state obtained after action a_i is applied. Also, each object is replaced with a variable. A more recent work (McCluskey et al., 2009) produces actions representations from training sequences without requiring large numbers of examples. These sequences are composed of an initial and a goal state, as well as the solution sequence written in terms of action names and affected objects. Moreover, the intermediate states are obtained without trainer intervention.

- **Learning hierarchical schema** : It is possible to learn preconditions of Hierarchical Task Network methods (see 2.5.2) by examining plan traces (Ilghami et al., 2002). These traces include a correct solution for the problem. Likewise, they include at each given point, a list of methods applicable to decompose the current task. This approach requires all methods' information to be given in advance. Instead, Ilghami et al. (2006), proposed an algorithm to learn domain description from traces in HTN planning with no prior information about the methods. The algorithm verifies for each decomposition point if the method implied exists. If not, the algorithm creates it as a new method and try to capture its preconditions by using the version space algorithm (Mitchell, 1981).

Learning control knowledge

The main idea is to capture specific knowledge, using one of the methods below, to guide the planning system when selecting operators and goals. This knowledge usually comes from analysing learning examples or, failing that, from analysing the relations between actions' preconditions and effects.

- **Control rules** (Borrajo and Veloso, 1997; Etzioni, 1993; Aler et al., 2002; Herrera et al., 1998): It consists of an `IF-THEN` rule for proposing node pruning or proposing node ordering during the search exploration. They introduce extra predicates enriching the planning model.

- **Cases** (Hammond, 1990; Carrick et al., 1999; De Mantaras et al., 2005; Craw et al., 2006): A case is defined as a trace of past solved planning problem. They are stored in a plan library and indexed for a later easy recovery. They are used when a new problem match a previously similar problem in the plan library. Thus they can be applied without changes or they can be modified to solve the new problem.

- **Heuristics** (Yoon et al., 2006; Xu et al., 2007): The purpose of learning heuristics (Definition 2.9) is to tackle domains where they are less accurate. The learned heuristic captures domain specific regularities through regression process that involves observations of the true distance to the goal from diverse states. A more recent work (Garrett et al., 2016) came up with a different approach, to consider learning heuristics as a "learning to rank" problem.

- **General Policies** (Khardon, 1999; Yoon et al., 2007; de la Rosa et al., 2008): A policy maps world states to preferred actions to execute. Accordingly, a general policy maps all combinations between initial and goal states to preferred actions to be executed. Recently, the learned policies are applied but combined with heuristic planning algorithms.

- **Macros** (Botea et al., 2005a; Newton and Levine, 2010): A macro is a sequence of actions that occurs frequently in solution plans. Learning macros is relevant since the use of macro-actions reduces the depth of the search tree. Their handling should be defined to ensure a good balance between performance improvement and search space enlargement.

2.6 Macro learning methods in Automated Planning

From the literature, we need to distinguish two related but different terms: macro-actions and macro-operators. A macro-action is related to a macro-operator as an action is related to an operator. They are based on the idea of composing a sequence of primitive operators and viewing the sequence as a single operator (Amarel, 1968).

Definition 2.12.
A macro-operator is a triple $m_o = (name(m_o), pre(m_o), effects(m_o))$ where its elements are defined as follows:

- $name(m_o)$ is in the form $name(x_1,...,x_n)$ where $x_1,...,x_n$ are the object variable

symbols that appear in m_o.

- $pre(m_o)$ is the set of precondition formula that must be hold before exploiting the macro-operator.
- *effects*(m_o) is the set of predicates to be applied to a state.

Example Let us consider the *pick-up_stack* macro-operator in Code 2.6 composed of the sequence of primitive operators *pick-up->stack*. Object variable symbols are defined in the `:parameters` clause and the preconditions (resp. effects) are defined in its `:precondition` (resp. `:effect`) clause. ■

```
(:action pick-up_stack
        :parameters (?x - block ?y - block)
        :precondition (and (clear ?x) (ontable ?x)
        (handempty) (clear ?y))
        :effect
        (and
          //negative effects
        (not (ontable ?x))
        (not (holding ?x))
        (not (clear ?y))
        //positive effects
        (clear ?x) (handempty) (on ?x ?y)))
```

PDDL-Code 2.6: PDDL definition for *pick-up_stack* blocksworld macro-operator

Definition 2.13.
A macro-action m_a is an instantiation of a macro-operator. Thus, m_a is a triple $m_a = (pre(m_a), add(m_a), del(m_a))$. If an action can be applied, a new state is generated. First it deletes all instantiated predicates given in the delete list $del(m_a)$, also known as the negative effects. Then, it adds all instantiated predicates given in the add-list $add(m_a)$, also known as the positive effects.

Example In Code 2.7, we have instantiated the *pick-up_stack* macro-operator with blockB and blockA. ■

```
(:action pick-up_stack
        :parameters (blockB blockA)
        :precondition (and (clear ?B) (ontable ?B)
        (handempty) (clear ?A))
        :effect
        (and
          //negative effects
        (not (ontable ?B))
        (not (holding ?B))
        (not (clear ?A))
        //positive effects
        (clear ?B) (handempty) (on ?B ?A)))
```

PDDL-Code 2.7: Instantiation of *pick-up_stack* blocksworld macro-operator

Although, there are some misuse of the terms macro-action and macro-operators in the literature (where one is substituted for the other), we use these terms according to the definitions presented above.

Learning macro-operators (aka *macros*) from previously acquired knowledge (plans) allows to go deep quickly into the search space by triggering them during the search (see example in Figure 1.2 and Figure 1.3).

In macro learning methods, we distinguish two main phases: generation and selection. The *generation* consists of identifying sequences of actions that could be potential candidates to enhance the domain. Macros literature presents various techniques to generate macros, ranging from a simple matter of combining primitive actions and the use of chunks of plans to the use of genetic learning algorithms or statistical analyses based on n-grams.

However, the main disadvantage of macros is to increase the branching factor of the search space. Indeed, by adding macros, the system must consider primitive operators as well as new macros.

Therefore, the use of macros raises a utility issue. The *selection* phase must found a trade-off between the benefit expected from adding macros and the additional cost induced by the branching factor increase. The selection phase plays a vital role because carefully selected macros can significantly improve performance by reducing the depth of the search space.

In the following, we present a chronological overview on macro learning literature in Table 2.3. Also, we present the most recent works.

Table 2.3: Timeline of macro learning methods.

1968	1st apparition of macro idea (Amarel, 1968).
1971	Use of MACROPs by STRIPS (Fikes et al., 1972).
1977	REFLECT system and its BIGOPS (Dawson and Siklossy, 1977).
1985	Macro problem solver (Korf, 1985).
1989	MCLEARN (Iba, 1989).
2005	CA-ED and SOL-EP methods (Botea et al., 2005a).
2007	MARVIN (Coles et al., 2007) WIZARD (Newton et al., 2007).
2013	Learning macros with n-grams (Dulac et al., 2013).
2014	MUM (Chrpa et al., 2014).
2015	Online generation of macros (Chrpa et al., 2015).
2017	Generation of macros from a plan database (Hofmann et al., 2017).

We group macros related work into two main categories: off-line and on-line techniques.

2.6.1 Off-line approaches

An off-line approach offers as an advantage an ease view over the macro-actions use, but also over the impact in the search time.

Macro-FF (Botea et al., 2005a) extracts macro-actions from solutions of training problems by identifying statically connected abstract components. Only the macro-actions showing effective performances in solving training problems are kept for future searches. Newton et al. (2007) proposed another offline method which uses a genetic algorithm as a learning technique and plans as the macro generation source. The algorithm generates the macros from plans of simple problems to seed the population and evaluates them through a ranking method based on the weighted average of time differences in solving more difficult problems with the original domain augmented with macros.

Dulac et al. (2013) introduced a domain-independent approach for learning macros from before computed solutions. It extracts statistical information from successful plans based on a n-gram analysis. Then it builds a macro library based on earlier information, a generalisation and a specialisation process. Finally, it adds selected macros into the planning domain after a filtering phase based on statistical information and heuristics. Later, Chrpa et al. (2014) proposed a technique to maximise the utility of macros. It first learns the causal relations between planning operators and initial or goal predicates (also known as outer entanglements) by using an approximation algorithm in several training plans. Then, exploiting this knowledge it generates macros and uses them to reformulate the original domain model.

In a more recent work, Hofmann et al. (2017) identifies operator sequences from a database of recorded plans by using the MapReduce database query paradigm. From these sequences, macros are generated with proper preconditions and effects. After adding one or multiple macros to a domain and solving problems with the augmented domain, the result is assessed with evaluation metrics. Finally, these metrics guide the selection of the best macro configuration.

2.6.2 On-line approaches

An on-line approach remove the need of extra training problems and off-line filtering.

Coles and Smith (2007) described Marvin planner. It identifies regions in the search space where the heuristic values of all successors is greater than or equal to the best seen so far. Then, it learns the escaping macro-actions to use them in similar regions during the search. This work was improved in (Coles et al., 2007) by keeping libraries of macro-actions for use on future problems. Chrpa et al. (2015) extended their early

technique by generating useful macros from outer entanglements in the search without an offline learning phase.

The presented works have some limitations. In the generation phase, for example, some works limit the length of the analysed sequences of operators (Botea et al., 2005a) or the number of generated macros (Chrpa et al., 2014). Also, in the selection phase, there exist limitations such as the maximum number of macros to add to the domain and usually, the performance of the augmented macro domain is tested before deciding on the utility of a macro, i.e. the evaluation on the utility of a macro is done experimentally.

Finally, for approaches extracting macros from past experiences, an assumption often used is that frequent sequences of actions are potentially good candidates to enhance the domain. We could, therefore, consider an approach that exploits this hypothesis — for example, the *pattern mining* technique from the field of data mining which aims to discover frequent patterns in data.

2.7 Conclusion

Planning is a careful consideration process by which actions are chosen to achieve a specific goal. It is needed to understand the problem and to adapt the resources to attain an objective as best as possible.

Automated planning aims to study and design effective algorithms that produce action sequences to reach a planning task goal, for a potential execution by one or several agents.

Solving planning problems is a difficult and time-consuming process because the planning task must be understood without the use of domain-specific knowledge. To this must also be added the NP-hard complexity of planning. Namely, the time required to solve a planning problem increases very quickly as the size of the problem grows.

Therefore, it is essential to develop powerful algorithms. They must efficiently explore the search space that grows exponentially. One way to do this is by exploiting knowledge about the structure of the planning tasks (Long and Fox, 2003) and thus, increase planner performance.

Given that, we decided to develop algorithms based on the study of macro learning methods which has been widely studied among the different approaches to speed-up planning processes.

3

Pattern Mining

You can have data without information, but you cannot have information without data.

Daniel Keys Moran

3.1 Introduction

Extracting useful information and most importantly, making the data intelligible from large volumes of data, is not possible with traditional data analysis tools and techniques. Therefore, an alternative method integrating traditional methods and new algorithms capable to process huge amounts of data is needed.

This extraction task is part of the process of knowledge discovery which also includes steps such as data preprocessing, pattern evaluation and knowledge presentation. The whole process is often called *data mining*. Data mining techniques are widely used in many fields including, among others, market, to identify customer profile, customer requirements, customer purchasing patterns; enterprise, to analyse and predict cash flow, to improve resource planning; security, to detect frauds by analysing the unexpected patterns; finances, to classify customers, to detect money laundering.

Hence, the purpose of data mining is look for patterns by searching automatically in data stored electronically. There are two major categories of data mining tasks : Predictive tasks and descriptive tasks. The aim of the former is to predict the value of an attribute (target) based on the values of other attributes while the aim of the latter is to explore patterns that compile general properties in data.

In this chapter, we introduce the descriptive task of data mining called *Pattern mining*. In addition, we focus on the concepts around mining a complex data type: *the sequence data*.

3.2 Pattern Mining: Basic concepts

Pattern mining intends to discover interesting and useful patterns in data. In this section, we present first the elementary forms of data for mining, followed by the patterns that can be mined.

3.2.1 Simple types of data

Basic data for mining include database data, warehouse data and transactional data.

Database data

These are part of a collection of interrelated data, known as a database. This one, in turn, is part of a database system which also includes software to manage and access the data. Most of the time, these data are modelled using an entity-relationship data model and they are stored as a collection of tables, each of which consists of a set of attributes and stores a large set of tuples.

Figure 3.1 shows a sample database representing entities and relationships of Employees (Crews and Maxia, 2015). It provides a large amount of data (4 millions records in total) distributed over six tables.

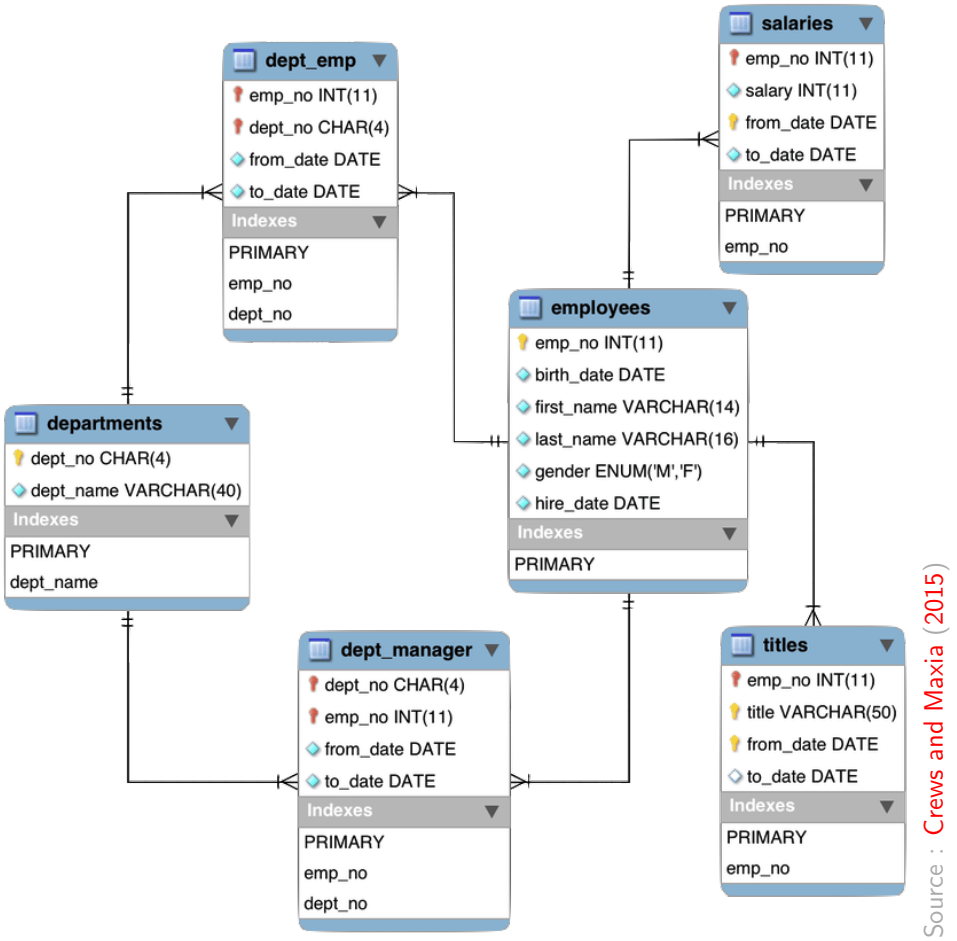


Figure 3.1: Employees database

Warehouse data

These are part of a repository of information collected from multiple sources. This repository is known as *data warehouse* and it usually exists in a single site. The data is stored to provide a historical overview and to summarize a set of attributes from the original data.

Example Let us consider an international company with several branches all over the world. Each branch has a employees database like the one in Figure 3.1. The data warehouse may store a summary of the employees evolution in time for each branch or for a region. ■

Transactional data

These data include a unique transaction identity number and a list of items belonging to this transaction. Each record is stored as a transaction in a *transactional database*, usually a flat file. In a transactional database, an item is not allowed to appear twice in the same transaction. Additionally, for each transaction, the items are assumed to be sorted by lexicographical order.

Example Let us consider the transactional database in Table 3.1 where each transaction corresponds to the sections consulted by an employee on the company's intranet during his session. However, the order in which the sections were consulted and the page consultation recurrence cannot be deduced. ■

Transaction_ID	items_IDs
T001	s1, s5, s11, s13, s16
T002	s8, s11, s12, s13
T003	s4, s5, s11, s13
T004	s8, s13, s16
T005	s1, s6, s8

Table 3.1: Transactional database for consulted sections on a company's intranet.

3.2.2 Types of patterns

There are many kind of patterns like frequent patterns, association rules and periodic patterns, among others.

Frequent patterns

These are patterns that appear frequently in data. In this group of patterns are included:

- The frequent itemsets: It consists of a set of items that appear together with a given frequency in transactional data. For example, the analysis of "which sections of a company's intranet are often consulted by its employees ?" give as a result a set of frequent itemsets.

- The frequent subsequences: It consists of a frequently occurring subsequence in complex data types. For example, the pattern that consists of an employee visiting the meeting room reservation page, followed by a visit of the meeting scheduling page.

In the following, we introduce some definitions to provide a better understanding of frequent itemsets concepts.

Definition 3.1.
An itemset $X = \{x_1, ..., x_k\}$ is a set of one or more items.

Definition 3.2.
The absolute support of an itemset X is the number of occurrences of an itemset X.

Example From Table 3.1, the absolute support of the itemset $\{s5, s11\}$ is two since it appears in T001 and in T003. ■

Definition 3.3.
The relative support of an itemset X is the fraction of transactions containing an itemset X.

Example From Table 3.1, the relative support of the itemset $\{s5, s11\}$ is $\frac{2}{5}$. ■

From these definitions, we can establish that an itemset is frequent if the support of X is greater or equal than a given threshold. This threshold, also known as *minsup*, is usually given by the user analyst.

We will present the concepts around the frequent subsequences in the next section since they are part of the focus of this work.

Association rules

They indicate if an item or a set of unordered items are likely to occur after another item or set of unordered items with a given probability.

Definition 3.4.
An association rule $X \mapsto Y(s,c)$ represents the association of an itemset X to an itemset Y having a support s and a confidence c.

Definition 3.5.
The support s of an association rule is the probability that a transaction contains both itemsets X and Y.

Example Let us assume the association rule $s11 \mapsto s13$. The support is $s(s11 \cup s13) = \frac{3}{5} = 60\%$. They are both contained in transactions with ids: T001, T002 and T003. ■

Definition 3.6.
The confidence c of an association rule is the conditional probability that a transaction containing X also contains Y.

Example Let us continue with the association rule $s11 \mapsto s13$. The confidence is $c(s11 \mapsto s13) = \frac{s(s11 \cup s13)}{s(s11)} = 100\%$. Every time that $s11$ appears, so does $s13$. ■

Indeed, the association rule mining consists of finding all the rules, in transactional data, having a support greater or equal than a minsup threshold and a confidence no less than a minconf threshold.

Periodic patterns

They consist of a set items that occur frequently in data within a given period.

Definition 3.7.
The period p of an itemset X is the number of transactions between two occurrences of X.

Example Let us consider the pattern $X = \{s8, s13\}$ in Table 3.1. This pattern has two periods, $p(X) = \{2,2\}$.

The first period has a length of two, from the first transaction until the first occurrence of the pattern (T002). The second period has a length of two, from the last occurrence (T002) until the next occurrence of the pattern (T004).

Thus, periodic pattern mining consists of finding the patterns, in transactional data, having a support no less than a minsup threshold and a maximum period no greater than a maxper threshold. A recent algorithm (Fournier-Viger et al., 2017a), makes this definition more flexible. The novelty includes the use of an average periodicity and a minimum periodicity.

3.3 Mining frequent patterns

In this section, we present the set of patterns that we can obtain when mining frequent patterns, followed by the basic algorithm for finding frequent itemsets.

3.3.1 Pattern sets

A long itemset contains a combinatorial number of frequent sub-itemsets.

Example Let us consider from Table 3.1, a frequent itemset $X = \{s5, s11, s13\}$. It contains seven frequent sub-itemsets:

- 1-itemsets: $\{s5\}, \{s11\}, \{s13\}$.
- 2-itemsets: $\{s5, s11\}, \{s5, s13\}, \{s11, s13\}$.
- 3-itemsets: $\{s5, s11, s13\}$. ■

Now, let us suppose a frequent itemset of length k. It will contain $\binom{k}{1} + \binom{k}{2} + \cdots + \binom{k}{k} = 2^k - 1$ itemsets. For high k values, this becomes a very large number of itemsets to compute.

All frequent patterns

Indeed, a vast number of itemsets satisfying the minimum support threshold can be generated when mining all frequent patterns, particularly, if this threshold is set low.

Definition 3.8.
A frequent pattern set FP contains all the patterns P such that the relative support of P is no less than the minsup parameter provided by an user, denoted $FP = \{P | \, s(P) \geq minsup\}$

Example The result of mining all frequent patterns from the transactional data in Table 3.1 is shown in Table 3.2. ■

Therefore, there exist other pattern sets that can reduce the number of generated frequent itemsets: the set of *closed frequent itemset* and the set of *maximal frequent itemset*.

Itemset	Support	Itemset	Support
{s16}	2	{s12,s13}	1
{s1}	2	{s13,s16}	2
{s4}	1	{s1,s5,s11}	1
{s5}	2	{s1,s5,s13}	1
{s6}	1	{s1,s5,s16}	1
{s8}	3	{s1,s6,s8}	1
{s11}	3	{s1,s11,s13}	1
{s12}	1	{s1,s11,s16}	1
{s13}	4	{s1,s13,s16}	1
{s1,s5}	1	{s4,s5,s11}	1
{s1,s6}	1	{s4,s5,s13}	1
{s1,s8}	1	{s4,s11,s13}	1
{s1,s11}	1	{s5,s11,s13}	2
{s1,s13}	1	{s5,s11,s16}	1
{s1,s16}	1	{s5,s13,s16}	1
{s4,s5}	1	{s8,s11,s12}	1
{s4,s11}	1	{s8,s11,s13}	1
{s4,s13}	1	{s8,s12,s13}	1
{s5,s11}	2	{s8,s13,s16}	1
{s5,s13}	2	{s11,s12,s13}	1
{s5,s16}	1	{s11,s13,s16}	1
{s6,s8}	1	{s1,s5,s11,s13}	1
{s8,s11}	1	{s1,s5,s11,s16}	1
{s8,s12}	1	{s1,s5,s13,s16}	1
{s8,s13}	2	{s1,s11,s13,s16}	1
{s8,s16}	1	{s4,s5,s11,s13}	1
{s11,s12}	1	{s5,s11,s13,s16}	1
{s11,s13}	3	{s8,s11,s12,s13}	1
{s11,s16}	1	{s1,s5,s11,s13,s16}	1

Table 3.2: All frequent itemsets from Table 3.1 (minsup = 0.01).

Closed frequent patterns

Because of the computing and the storage resources, sometimes it is useless to keep patterns included into another pattern having the same support. The closed frequent pattern set is a subset of the frequent pattern set.

Definition 3.9.
A closed frequent pattern is a pattern that is not included in another pattern having the same support.

Example The result of mining the closed frequent patterns from the transactional data in Table 3.1 is shown in Table 3.3. To summarize, if two itemsets have the same support, only the longest one will be kept. ■

Maximal frequent patterns

This pattern set is a subset of the closed sequential pattern set.

Definition 3.10.
A maximal frequent pattern is a pattern that it is not strictly included in another pattern.

Example To illustrate that, let us consider two closed patterns cp_1 and cp_2. They have a support of i and j, respectively. Also, cp_1 is included into cp_2 ($cp_1 \subset cp_2$) and the support of cp_2 is smaller than the support of cp_1 ($s(cp_2) < s(cp_1)$). Therefore, cp_2 will belong to the maximal pattern set, but not cp_1. The result of mining the maximal frequent patterns from the transactional data in Table 3.1 is shown in Table 3.4. ■

This set implies a loss of information because the kept itemsets no longer assure the notion of support but they are chosen based on the notion of inclusion into another itemset.

Itemset	Support
{s1}	2
{s8}	3
{s13}	4
{s8,s13}	2
{s11,s13}	3
{s13,s16}	2
{s1,s6,s8}	1
{s5,s11,s13}	2
{s8,s13,s16}	1
{s4,s5,s11,s13}	1
{s8,s11,s12,s13}	1
{s1,s5,s11,s13,s16}	1

Table 3.3: Closed frequent itemsets from Table 3.1 (minsup = 0.01).

Itemset	Support
{s8,s1,s6}	1
{s13,s8,s16}	1
{s13,s8,s11,s12}	1
{s13,s11,s5,s4}	1
{s13,s11,s1,s5,s16}	1

Table 3.4: Maximal frequent itemsets from Table 3.1 (minsup = 0.01).

To sum up, there exist different pattern sets that can be obtained from mining frequent itemsets. Each set is obtained from following some restrictions. These sets respect the property 3.1.

Property 3.1.
frequent pattern set $\supset$ *closed pattern set* $\supset$ *maximal pattern set*

3.3.2 Apriori algorithm

The Apriori algorithm is a basic algorithm for mining frequent itemsets introduced by Agrawal and Srikant (1994). It consists of iterations of a candidate generation step followed by a pruning step. In the former, the frequent subsets are extended one item at a time while in the latter, these extended subsets are tested against the data to keep only the frequent members. The main idea behind this algorithm is that *all nonempty subsets of a frequent itemset must also be frequent*. Based on this assumption, it concludes using prior knowledge that some combinations cannot have minimum support and it does not process them.

Example Let us consider a $minsup = 0.5$ and an itemset $X = \{s5\}$ having a support $s(X)$ of 0.4. Because the itemset X has a support less than the minimum support threshold $s(X) < minsup$ then this itemset is not frequent. Now, if an item I is added to this itemset, the obtained itemset cannot appear more times than X. Indeed, $s(X \cup I) < minsup$ ■

As stated before, Apriori algorithm follows an iterative approach. It starts by extracting from the transactional database the set of frequent 1-itemsets and its support. From these itemsets, it keeps only the ones satisfying the minsup threshold. Then, those are used to find the frequent 2-itemsets, and the transactional database is used to assign them their respective support. The kept itemsets are used to find the next k-itemsets and those are compared against the transactional database to find their respective support. The algorithm repeats this process until there are no more frequent k-itemsets.

Example Let us find, using the Apriori algorithm with a minimum threshold of 0.25, the frequent itemsets from Table 3.1.

Figure 3.2 shows the different steps carried out by the apriori algorithm. First, it scans the database to accumulate the count for each item. Then, it keeps the items having a relative support greater than 0.25. In our case, it means that the items must have an absolute support of 1.25.

From this resulting set, the algorithm generates a new set of candidates (2-itemsets) by joining it with itself. Then, it proceeds to scan the database to accumulate the count for each candidate. Afterwards, it compares the candidate support with the minimum support count and it removes candidates that do not satisfy the support condition. From here, it repeats the process to generate the 3-itemsets candidates and to prune them. Finally, the result of this process is presented in 3.3.

Candidate Generation		Pruning step	
1-Itemsets	**Supp.**	**1-Itemsets**	**Supp.**
{s1}	2	{s1}	2
{s5}	2	{s5}	2
{s11}	3	{s11}	3
{s13}	4	{s13}	4
{s16}	2	{s16}	2
{s8}	3	{s8}	3
{s12}	1	{s12}	1
{s4}	1	{s4}	1
{s6}	1	{s6}	1

(a) 1-itemsets

Candidate Generation		Pruning step			
2-Itemsets	**2-Itemsets**	**2-Itemsets**	**Supp.**	**2-Itemsets**	**Supp.**
{s1,s5}	{s5,s8}	{s1,s5}	1	{s5,s8}	0
{s1,s8}	{s11,s13}	{s1,s8}	1	{s8,s13}	2
{s1,s11}	{s11,s16}	{s1,s11}	1	{s8,s11}	1
{s1,s13}	{s11,s8}	{s1,s13}	1	{s8,s16}	1
{s1,s16}	{s13,s16}	{s1,s16}	1	{s11,s13}	3
{s5,s11}	{s13,s8}	{s5,s11}	2	{s11,s16}	2
{s5,s13}	{s16,s8}	{s5,s13}	2	{s13,s16}	2
{s5,s16}		{s5,s16}	1		

(b) 2-itemsets

Candidate Generation	Pruning step	
3-Itemsets	**3-Itemsets**	**Supp.**
{s5,s11,s13}	{s5,s11,s13}	2
{s8,s13,s16}	{s8,s13,s16}	1

(c) 3-itemsets

Figure 3.2: Apriori algorithm steps for mining frequent itemsets from Table 3.1 with a $minsup = 0.25$. "Supp." stands for absolute support.

Frequent Itemsets	Supp.
{s1}	2
{s5}	2
{s11}	3
{s13}	4
{s16}	2
{s8}	3
{s5,s11}	2

Frequent Itemsets	Supp.
{s5,s13}	2
{s8,s13}	2
{s11,s13}	3
{s11,s16}	2
{s13,s16}	2
{s5,s11,s13}	2

Figure 3.3: Frequent itemsets obtained by using the Apriori algorithm on Table 3.1 with a $minsup = 0.25$. "Supp." stands for absolute support.

3.4 Mining sequence data

Nowadays, a wide range of applications that use data mining continue to appear. This leads to new developments and research efforts in mining complex data types (Han et al., 2012). For the purposes of this work, we will focus in mining a complex type of data called *sequence data*.

Definition 3.11.
A sequence is an ordered list of itemsets, denoted $s =< I_1, I_2, \ldots, I_n >$

These data include a unique sequence identifier *sid* and a sequence *s*. Each record is stored as a sequence in a *sequence database*, usually a flat file. For each sequence, the order of the data matters, the entire sequence is known and there is no notion of future and past. Additionally, for each itemset, the items are assumed to be sorted by lexicographical order and they are not allowed to appear twice in the same itemset.

Definition 3.12.
A sequence database *SDB* is a set of pairs $< sid, s >$, where *sid* is a sequence identifier and *s* is a sequence.

Example Let us consider the sequence database in Table 3.5 where each sequence represents, for a given employee, the consulted sections on the company's intranet during different authentications over the curse of a day.

For example, the sequence *S*004 represents an employee who did two authentications on the same day: In his first authentication, he visited a section identified as *s*8 (e.g. the company's directory); and, in his second authentication, he visited two sections

Sequence_ID	Sequence
S001	{s1,s5,s11},{s4},{s8}
S002	{s8,s11,s12},{s5},{s1,s2}
S003	{s4},{s5,s11},{s13}
S004	{s8},{s13,s16}
S005	{s1,s2},{s8}

Table 3.5: Sequence database for consulted sections on a company's intranet

identified as $s13$ (e.g. the page for booking meeting rooms) and $s16$ (e.g. the page to schedule meetings). ■

3.4.1 Sequential pattern mining

It is a data mining task that consists of mining *sequential patterns* from a sequence database.

Definition 3.13.
A sequential pattern is a frequent *subsequence* existing in a sequence or a set of sequences.

Definition 3.14.
A sequence $S_A = X_1, X_2, \ldots X_k$, where $X_1, X_2 \ldots X_k$ are events, is a subsequence of another sequence $S_B = Y_1, Y_2, \ldots Y_m$, where $Y_1, Y_2 \ldots Y_m$ are events, if and only if there exists integers $1 \leq e_1 < e_2 \cdots < ek \leq m$ such that $X_1 \subseteq Y_{e_1}, X_2 \subseteq Y_{e_2}, \ldots X_k \subseteq Y_{e_k}$.

As shown in definitions 3.13 and 3.14, a sequential pattern is a sequence occurring in another sequence but not necessarily in a contiguous fashion.

Example From table 3.5, the sequence $S_A = \{s1\}, \{s8\}$ is contained in sequence $S001$ and $S005$. ■

This kind of analysis can be done by hand. However, this is time-consuming and it is not realistic to do it on a large dataset. Thus, the purpose of sequential pattern mining is tackle this problem by designing automatic techniques to analyse data and extract interesting patterns from data.

The problem of mining sequential patterns was introduced by Agrawal and Srikant (1995). It consists of finding all sequential patterns in a sequence database. To formally define this problem, we need the following definitions .

Definition 3.15.
The absolute support of a sequential pattern is the number of sequences where the pattern occurs, denoted $\sigma(x)$.

Definition 3.16.
The relative support of a sequential pattern is the number of sequences where the pattern occurs divided by the total number of sequences in the sequence database, denoted $sup(x)$.

Definition 3.17.
The parameter *minsup* is a user-specified threshold (a value in $[0,1]$ representing a percentage) allowing to discover a sequential pattern.

Therefore, discovering all sequential patterns comes to finding the set FSP of all sequences s_k such that $sup(s_k) \geq minsup$.

Example The result of mining all sequential patterns from the sequence database in Table 3.5 is shown in Table 3.6. ■

Sequential Pattern	σ	sup
{s1}	3	0.6
{s1 s2}	2	0.4
{s1}, {s8}	2	0.4
{s2}	2	0.4
{s4}	2	0.4
{s5}	3	0.6
{s5 s11 }	2	0.4
{s8}	4	0.8
{s11}	3	0.6
{s13}	2	0.4

Table 3.6: Sequential patterns from Table3.5 (minsup=0.25).

In the literature, we can find several algorithms performing frequent sequential pattern mining such as FAST (Salvemini et al., 2011), FreeSpan (Han et al., 2000), GSP(Srikant and Agrawal, 1996), LAPIN(Yang and Kitsuregawa, 2005), PrefixSpan(Pei et al., 2004), SPADE(Zaki, 2001; Fournier-Viger et al., 2014a), SPAM (Ayres et al., 2002; Fournier-Viger et al., 2014a), among others.

Analogous to itemset mining, in sequential pattern mining we can also define subsets with properties similar to closed pattern set and maximal pattern set.

Definition 3.18.
A closed sequential pattern is a pattern that is not included in another pattern having the same support.

Some of the closed sequential pattern algorithms found in the literature are: CloSpan (Yan et al., 2003), BIDE (Wang and Han, 2004), ClaSP (Gomariz et al., 2013; Fournier-Viger et al., 2014a) and CloFAST(Fumarola et al., 2016).

Definition 3.19.
A maximal sequential pattern is a pattern that it is not strictly included in another sequential pattern.

Maximal sequential pattern algorithms in the literature are: AprioriAdjust (Lu and Li, 2004), MFSPAN (Guan et al., 2005), MaxSP (Fournier-Viger et al., 2013), VMSP (Fournier-Viger et al., 2014c), among others.

For a given sequential pattern set (frequent, closed, maximal), the efficiency of the algorithms differs by their candidate generation strategy, the search strategy and their accompanying data structure (Mabroukeh and Ezeife, 2010). However, the resulting set is always the same for a given input (a sequence database *SDB* and a *minsup* threshold). The efficiency of sequential pattern algorithms is not part of the scope of this work. A detailed survey in sequential pattern algorithms is presented in Mabroukeh and Ezeife (2010); Fournier-Viger et al. (2017b).

3.5 Conclusion

The purpose of data mining is to look for patterns by searching automatically in data stored electronically. Pattern mining, the descriptive task of data mining, intends to discover interesting and useful patterns in data.

Pattern mining has become popular because of its applications in multiple domains. Although the different pattern mining techniques are aimed at analysing data, techniques such as itemset mining and association rule mining do not take into account the sequential ordering of events. Therefore, there exists a technique for mining sequence data called *sequential pattern mining*.

Sequential pattern mining consists in analyse sequential data to discover frequent sequential patterns. We can distinguish two filter structures provided by sequential pattern mining. On the one hand, there is a parameter called *support*, which filters patterns based on the frequency of apparition. On the other hand, from following some restrictions, the resulting pattern set can be reduced.

In this perspective, we aim to explore in the next section the use of sequential pattern mining for extracting useful sequences of actions (not necessarily adjacent) as macro-actions.

Part II

Contributions on learning routines for sequential decision-making

4

Extraction of macros via Sequential Pattern Mining

In short, no pattern is an isolated entity. Each pattern can exist in the world only to the extent that is supported by other patterns: the larger patterns in which it is embedded, the patterns of the same size that surround it, and the smaller patterns which are embedded in it.

Christopher Alexander

4.1 Introduction

Among the various approaches to scale up plan synthesis, macro learning methods have been widely explored (see Section 2.6). The whole idea behind this approach is to improve planning systems performance by exploiting the structure knowledge of planning tasks. This allows to properly define search control knowledge. In this chapter, we explore the use of sequential pattern mining for learning macro-actions from a set of plans. Indeed, sequential pattern mining is a sub-field of data mining that consist in analyse sequential data to detect sequential patterns. This kind of analysis can be done by hand. However, this is time-consuming and it is not realistic to do it on a large dataset. Thus, the purpose of sequential pattern mining is tackle this problem by designing automatic techniques to analyse data and extract interesting patterns from data.

We have several motivations behind the idea of using pattern mining algorithms to extract sequences candidates for becoming macro-actions. If a sequence of actions has a higher frequency of apparition on different plans, it will be mined and we can consider it as a useful macro-action for a given domain. Let us have an example, consider the `blocksworld` planning problem in Figure 4.1. The goal is to stack a set of blocks in a specific order. This domain has five operators: pick-up, picks a block x from the table; put-down, puts a block x on the table; stack, puts a block x on a block y; and unstack, removes a block x from a block y. It is logical to suggest that once we pick a block from the table the next most probably action will be stack it on another block or put it down. Moreover, if we observed from a representative data set, of previous acquired knowledge, a higher frequency of apparition for one of these sequences e.g. pick-up_stack_BA, we can use it as a candidate for the creation of a macro-action. Then, this macro-action can provide a way to go deep quickly into the search space in future `blocksworld` problems.

Nevertheless, the set of mined candidates can still be quite large for our purposes. In this context, we are interested in the filter structures provided by sequential pattern

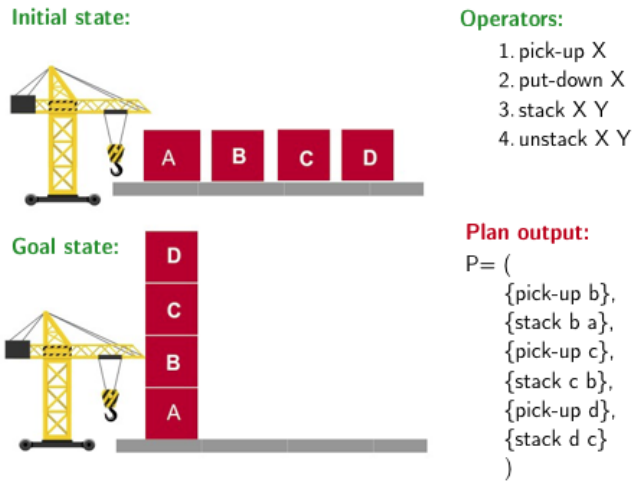


Figure 4.1: A `blocksworld` problem.

mining. On the one hand, the support parameter filters the possible candidates based on the frequency of apparition. On the other hand, depending on the used mining strategy (closed set, maximal set) the resulting pattern set can be reduced (see Section 3.3.1 in Chapter 3).

Our assumptions for carrying out this work are:

- Our data set of previous acquired knowledge is representative enough to focus on macro-actions;
- Highly recurrent sequences of actions (not necessarily adjacent) are more likely constrained to appear in that order to solve a given problem. As a consequence, these sequences are good candidates to build macro-actions;
- The frequency of apparition measure provided by sequential pattern mining algorithms is a good estimator to decide on the utility of macro-actions.

In this perspective, we aim to propose in the next section a framework to learn useful sequences of actions (not necessarily adjacent) as macro-actions and use them to speed-up planning search. This learning framework will be based on the filter structures provided by sequential pattern mining.

4.2 Plan encoding

Pattern mining has become popular because of its applications in multiple domains. Despite the different pattern mining techniques are aimed at analysing data, techniques such as itemset mining and association rule mining do not take into account the sequential ordering of events. Therefore, we will focus on working with sequential pattern mining.

To consider the extraction of macro-actions from sequential pattern mining algorithms, we present an intuitive formalism to represent a set of solution plans as a sequence database.

From the definitions in Chapter 3.4, we decide to represent each solution plan as an entry in the sequence database in the form $< \pi_{id}, \pi_i >$ where π_{id} is the plan identifier and π_i is a solution plan for a problem p_i. Then, a sequence s is equivalent to a plan solution π_i where the ordered list of events $< e_1 e_2 \dots e_n >$ are replaced by an ordered list of actions denoted $< a_1 a_2 \dots a_n >$.

We also give the definition 4.1 to represent the actions for a set of plans. We obtain a dictionary and by using it and the set of plans, we obtain a sequence database. In our proposed encoding, each plan is considered as a sequence, and each action, as an item

in this sequence.

Definition 4.1.
A dictionary of actions is a set of pairs $< k, a_k >$, where a_k is the k^{th} distinct encountered action in a set of plans and k is an integer referencing to a_k.

Example Let us consider the following set of plans where each π_i corresponds to a plan solution for a given problem p_i in the `blocksworld` domain:

$$\pi_1 = \textit{pick-up}(b); \textit{stack}(b,a); \textit{pick-up}(c); \textit{stack}(c,b); \textit{pick-up}(d); \textit{stack}(d,c)$$
$$\pi_2 = \textit{unstack}(b,c); \textit{put-down}(b); \textit{unstack}(c,a); \textit{put-down}(c); \textit{unstack}(a,d);$$
$$\textit{stack}(a,b); \textit{pick-up}(c); \textit{stack}(c,a); \textit{pick-up}(d); \textit{stack}(d,c)$$
$$\pi_3 = \textit{unstack}(c,b); \textit{stack}(c,d); \textit{pick-up}(b); \textit{stack}(b,c); \textit{pick-up}(a); \textit{stack}(a,b)$$
$$\pi_4 = \textit{unstack}(b,a); \textit{stack}(b,c); \textit{unstack}(a,d); \textit{stack}(a,e); \textit{unstack}(b,c);$$
$$\textit{stack}(b,a); \textit{pick-up}(c); \textit{stack}(c,b); \textit{pick-up}(d); \textit{stack}(d,c)$$

By following the definition 4.1, we obtain the dictionary shown in Table 4.1. Finally, we use this dictionary and the set of plans to obtain the sequence database in Table 4.2. Each sequence corresponds to a plan solution and each number to an action.

k	a_i
1	pick-up b
2	stack b a
3	pick-up c
4	stack c b
5	pick-up d
6	stack d c
7	unstack b c
8	put-down b
9	unstack c a
10	put-down c
11	unstack a d
12	stack a b
13	stack c a
14	unstack c b
15	stack c d
16	stack b c
17	pick-up a
18	unstack b a
19	stack a e

Table 4.1: Dictionary of actions from a set of plans in the current example.

π_{id}	π_i
π_1	1,2,3,4,5,6
π_2	7,8,9,10,11,12,3,13,5,6
π_3	14,15,1,16,17,12
π_4	18,16,11,19,7,2,3,4,5,6

Table 4.2: Sequence database from a set of plans in the current example. ■

4.3 Macro-actions learning framework

Let us recall the assumptions of our framework. First, our data set of previous acquired knowledge is representative enough to focus on macro-actions. Second, highly recurrent sequences of actions (not necessarily adjacent) are more likely constrained to appear in that order to solve a given problem. As a consequence, these sequences are good candidates to build macro-actions. Third, the frequency of apparition measure provided by sequential pattern mining algorithms is a good estimator to decide on the utility of macro-actions.

Thus, we aim to mine and filter frequent sequences of actions from a set of plans by using sequential pattern mining algorithms and its provided support measure. Useful extracted candidates will then build macro-actions which will be used to speed-up the planning search.

From this perspective, given a classic planning system, we propose the framework in Figure 4.2. It performs the following steps :

1. Mining candidates: Use of a sequential pattern mining algorithm on a sequence database (specific to a domain) to identify candidates.
2. Filtering of candidates: Perform an analysis on the identified candidates by using its frequency of apparition (provided by the pattern mining algorithm), in order to choose useful sequences candidates.
3. Macro-action construction: For each chosen candidate, concatenate all of its actions into a single, macro-action.
4. Enhancing planning domain: Add macro-actions to the original domain.

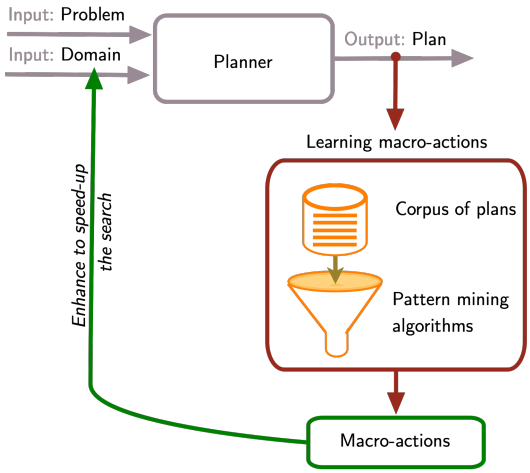


Figure 4.2: Macro-actions learning framework.

4.4 Mining and filtering candidates

For mining sequences of actions, we decided to use a strategy that reduces the number of resulting patterns, namely the extraction of closed sequential patterns (see Definition 3.18). Remember that sometimes it is useless to keep patterns included into another pattern having the same support. Indeed, we opted to avoid duplicate candidates. In other words, if a sequence s_1 of actions $\{a_1, a_2\}$ appeared twice and a sequence s_2 of actions $\{a_1, a_2, a_3\}$ appeared also twice, we kept the latter since every time that it is used the former is used.

The pseudo code is described in Algorithm 1. As input, it takes a sequence database D containing solution plans for a given domain and a *minsup* parameter specifying the minimum support that a candidate must satisfy. By using a closed sequential pattern mining algorithm, it extracts candidates (not necessarily adjacent) from D having a support greater or equal than *minsup* (line 2).

A first filtering is already done based on the *minsup* parameter because it keeps the set of patterns satisfying a minimum frequency threshold. For instance, on a sequence database using a minsup value of 0.2, it will keep closed sequential patterns appearing in at least 20% of the sequences; as a result, we obtain macro-actions candidates with a frequency of apparition of at least 20%.

Because sequential pattern mining algorithms consider sequential patterns of a single action but the choice of the minimum candidate length should be consistent with the macro-actions definition, a second filtering takes place. It is based on the candidate length (line 4). It keeps candidates with at least two actions and in order not to discard all long candidates, it limits the maximum length to 10. This choice allows to alleviate the processing of candidates when using low values for the minsup parameter. Moreover, we observed that the higher the value of the minsup parameter, the shorter the maximum length of the obtained candidates.

Algorithm 1 Mining and filtering candidates

Input A sequence database D of non-empty solution plans, an user-specified threshold *minsup*.

Output A list R of candidates (sequences of actions).

```
1: function PROCESSINGCANDIDATES(D, minsup)
2:     R ← closedSPM(D, minsup)
3:     for each sequence s in R do
4:         if length(s) = 1 or length(s)>10 then
5:             R ← R \ {s}
6:     return R
```

4.5 Macro-action construction

For each candidate obtained in the previous step, we built a new macro-action (see Definition 4.2). The pseudo code is described in Algorithm 2. It takes as input a sequence of at least two actions. It merges its two first actions into a single by using the *merge* procedure (see Algorithm 3 which will be described later) (line 2). After that, if the sequence of actions has more than two actions (line4), it turns again the *merge* procedure taking as parameters the last merged action and the next action in the input sequence (line 5). It repeats this process until all actions are merged. Finally, it removes all predicates appearing at the same time in the set of preconditions and in the set of positive effects of the obtained macro-action (line 8). Indeed, it is useless to have the same predicates appearing in both sets.

Definition 4.2.
A macro-action is a triple m = *(name(m), pre(m), effects(m))*. Its elements are defined as follows:

- $name(m)$ is in the form $name(c_1,...,c_n)$ where $c_1,...,c_n$ are the object constant symbols that appear in m.
- $pre(m)$ is the set of precondition formula that must be hold before exploiting the action.
- *effects(m)*= $\{add(m), del(m)\}$ is the set of positive and negative effects to be applied to a state.

The pseudo code of the *merge* procedure is described in Algorithm 3. It is based on the algorithm presented in Botea et al. (2004) whose formalism first appeared in Dawson and Siklossy (1977). It takes as input two actions and it validates the accuracy of predicates merge by using a series of conditions. First, predicates from the precondition set of the second action appearing in the delete effects of the first action, give as a result, an inability to continue with the merge procedure (line 7). Clearly, if the first action removes a predicate that is a precondition of the second action, the construction no longer makes sense. On the other hand, predicates from the precondition set of the second action are added to the precondition set of the first action, if they are no already included in this set or in the set of positive effects of the first action (line 8). Second, predicates from the delete effects sets are merged (line 11), and if needed, delete effects predicates of the second action are removed from the positive effects of the first action (line 10). Finally, predicates from the positive effects sets are merged(line 14), and if needed, positive effects predicates of the second action are removed from the delete effects of the first action (line 13). The output of the merge procedure is a single, merged action.

Algorithm 2 Macro-action construction

Input A sequence of actions $S = \{a_1, a_2, \ldots, a_n\}$ where $n \geq 2$
Output A ground macro-operator *gm*

```
1: function CONSTRUCTMACRO(S)
2:     gm = merge(S[0],S[1])
3:     i ← 2
4:     while i < length(S) and gm ≠ null do
5:         gm = merge(gm,S[i])
6:         i ← i+1
7:     if gm ≠ null then simplify(gm)
8:     return gm
```

Algorithm 3 The *merge* procedure

Input Two actions α and β
Output A merged action μ

```
1: function MERGE(α,β)
2:     P_γ, A_γ, D_γ where γ = {α, β}
3:     P_γ ← getPreconditions(γ)
4:     A_γ ← getPositiveEffects(γ)
5:     D_γ ← getNegativeEffects(γ)
6:     for each p in P_β do
7:         if p ∈ D_α then return null
8:         if p ∉ A_α and p ∉ P_α then add(p,P_α)
9:     for each d in D_β do
10:        if d ∈ A_α then remove(d,A_α)
11:        if d ∉ D_α then add(d,D_α)
12:    for each a in A_β do
13:        if a ∈ D_α then remove(a,D_α)
14:        if a ∉ A_α then add(a,A_α)
15:    μ = α
16:    return μ
```

4.5.1 Enhancing planning domain with macro-actions

Each obtained macro-action is added to the original domain. Thus, the original domain is enhanced since the goal of macro-actions is to provide a way to go deep quickly into

the search space while solving future problems. Also, the enhanced domain can be used for any planner using as input a domain defined in PDDL language. As an example of the actual step, we have the original Blocksworld domain in PDDL-Code 2.2 and a Blocksworld macro-action in PDDL-Code 4.1.

```
(:action unstack_put-down
    :parameters (?x - block ?y - block)
    :precondition
    (and (on ?x ?y)(clear ?x)(handempty))
    :effect
    (and (clear ?y)(clear ?x)(handempty)
    (ontable ?x)(not(on ?x ?y))(not(holding ?x))))
```

PDDL-Code 4.1: Macro-action for the `blocksworld` domain

4.6 Evaluation of the support parameter

In the following, an evaluation of the pattern mining support parameter is proposed, based on the hypothesis that this parameter is a good estimator for *a priori* macro-action utility (Castellanos-Paez et al., 2016). We present the methodology to conduct the evaluation, and we show and discuss some interesting results obtained by doing this evaluation.

4.6.1 Methodology

The evaluation was based on four benchmarks: `barman`, `blocksworld`, `depots` and `satellite`. They are described in more detail in Appendix A. These benchmarks problems were taken from past International Planning Competitions[1].

For each benchmark, a training set of problems of 1000 problems and a test set of 30 problems were generated. The problem generation stage uses the generators[2] from the International Planning Competition. In addition, it ensures that the generated problems can be different even using the same parameters. In Table 4.3, we show the parameters used for the generation of problems for each benchmark domain.

We used a heuristic search planner based on A* search strategy, from the PDDL4J library (Pellier and Fiorino, 2018), to obtain a set of solution plans from the training

[1] http://icaps-conference.org/index.php/Main/Competitions

[2] https://bitbucket.org/planning-tools/pddl-generators

Domain	Parameters	Range
Barman	cocktails	1-30
	ingredients	1-13
	shots	1-30
Blocksworld	blocks	5-30
Depots	depots	1-5
	distributors	1-3
	trucks	1-4
	pallets	1-8
	hoists	1-8
	crates	1-20
Satellite	satellites	1-6
	instruments	1-2
	modes	1-8
	targets	1-2
	observation	1-20

Table 4.3: Parameters for the generation of problems

set of problems. Then, we followed the *Macro-action learning framework* described in Section 4.3 for each benchmark. For the mining step, we first created a series of scripts, based on the presented plan encoding, to obtain a sequence database from a set of solution plans. After, we used the SPMF (Fournier-Viger et al., 2014b) data mining library, which implements CloFAST (Fumarola et al., 2016), a closed sequential pattern mining algorithm. It is important to note that for this algorithm each event of a pattern must appear in the same order in a sequence but it does not matter if it is in a consecutive way or not. During the mining step, we varied the *minsup* parameter in steps of 0.1 from 0.1 until no sequences were founded. By following the mentioned framework, at the end, we got an enhanced domain for each different *minsup* value used.

For the evaluation, we solved the set of test problems using the original domain, and then, using the enhanced domain. The relevant results, among others, such as the total run times of the different *minsup* values were assembled by using additional scripts.

Experimental setup

The macro-action learning steps and the evaluation of the support parameter were done on a notebook with an Intel Core i7-4980HQ quad-core CPU clocked at 2.8GHz and with 16GB of RAM, running OS X El Capitan v10.11.6. In the evaluation of the support parameter, to solve each problem from the test set, a maximum of 8GB of memory was

allocated and a time limit of 600 seconds was set in the planner. The experiments have been done in a non-graphical terminal session.

4.6.2 Evaluation criteria

Unlike other works, we did not base our evaluation only on the classical IPC score. IPC score[3] is intended, as the name implies, to give a score to rank different strategies. In other words, by using IPC score we can decide which strategy is better than another, but it does not quantify the gain.

Here, on top of the IPC ranking, we wanted to quantify the impact of enhancing planning domains by adding macro-actions constructed from different *minsup* values. Hence, different criteria were established in the evaluation of the support parameter as a good estimator to decide whether a macro is useful or not. In this perspective, we defined the planning time metric, the space size metric and the plan quality metric.

4.6.2.1 Planning Time metric

For a given problem p, let $T_o(p)$ be the time required by any planner to solve the problem p using the original domain. And, let $T_e(p)$ be the time required by any planner to solve the problem p using the enhanced domain. If the problem is not solved either using the original domain or the enhanced domain, we ignored it for evaluation. Also, if the problem was solved using the original (resp. enhanced) domain but not using the enhanced (resp.original) domain, T_o (resp. T_e) got the maximum time (600s). Thus, the planning time metric G_T, in Equation 4.1, is the sum of the quotient $\frac{To(p)}{Te(p)}$ for all problems p in the test set divided by P, where P is the number of non-ignored problems.

$$G_T = \frac{1}{P} \sum_{p \in \text{test problems}} \frac{To(p)}{Te(p)} \tag{4.1}$$

4.6.2.2 Space size metric

For a given problem p, let $N_o(p)$ be the total of nodes opened by the planner when solving the problem p using the original domain. And, let $N_e(p)$ be the total of nodes opened by the planner when solving the problem p using the enhanced domain. If the problem is not solved either by using the original domain or the enhanced domain, we ignored it for evaluation. We also ignored the problem, if it was only solved by using

[3] As defined in the Learning track of the 7th International Planning Competition (Jiménez Celorrio et al., 2011)

one of the domains. Thus, the space size metric G_N in Equation 4.2, is the sum of the quotient $\frac{No(p)}{Ne(p)}$ for all problems p in the test set divided by P, where P is the number of non-ignored problems.

$$G_N = \frac{1}{P} \sum_{p \in \text{test problems}} \frac{No(p)}{Ne(p)} \tag{4.2}$$

4.6.2.3 Plan Quality metric

For a given problem p, let $Q_o(p)$ be the plan length (number of actions) returned by the planner when solving the problem p using the original domain. And, let $Q_e(p)$ be the plan length (number of actions) returned by the planner when solving the problem p using the enhanced domain. If the problem is not solved either by using the original domain or the enhanced domain, we ignored it for evaluation. We also ignored the problem, if it was only solved by using one of the domains. Thus, the plan quality metric G_Q in Equation 4.3, is the sum of the quotient $\frac{Qo(p)}{Qe(p)}$ for all problems p in the test set divided by P, where P is the number of non-ignored problems.

$$G_Q = \frac{1}{P} \sum_{p \in \text{test problems}} \frac{Qo(p)}{Qe(p)} \tag{4.3}$$

4.7 Results

In this section, we present the results of the evaluation of the support parameter following the steps of the macro-action learning framework in Section 4.3.

The useful sequences candidates, obtained after the two first steps, are presented in Figure 4.3. The number of candidates decreases with increasing values of the *minsup* parameter. In domains such as `depots` and `satellite`, the number of candidates decreases and reaches zero much faster than for `barman` and `blocksworld` domains. Among the four domains used, only in the `barman` domain, there are still an important number of sequences candidates beyond a *minsup* value of 0.5. In other words, in this domain, there exist sequences of actions appearing in more than 50% of the plan solutions. Also, the order of magnitude of the number of candidates is variable for each value of the *minsup* parameter in each domain. For example, for a same *minsup* value of 0.1, the number of candidates covers all orders of magnitude ranging from units (10^0) for the `satellite` domain up to the tens of thousands (10^4) for the `barman` domain. Clearly, these results suggest that the number of mined candidates is related to the characteristics of the domain.

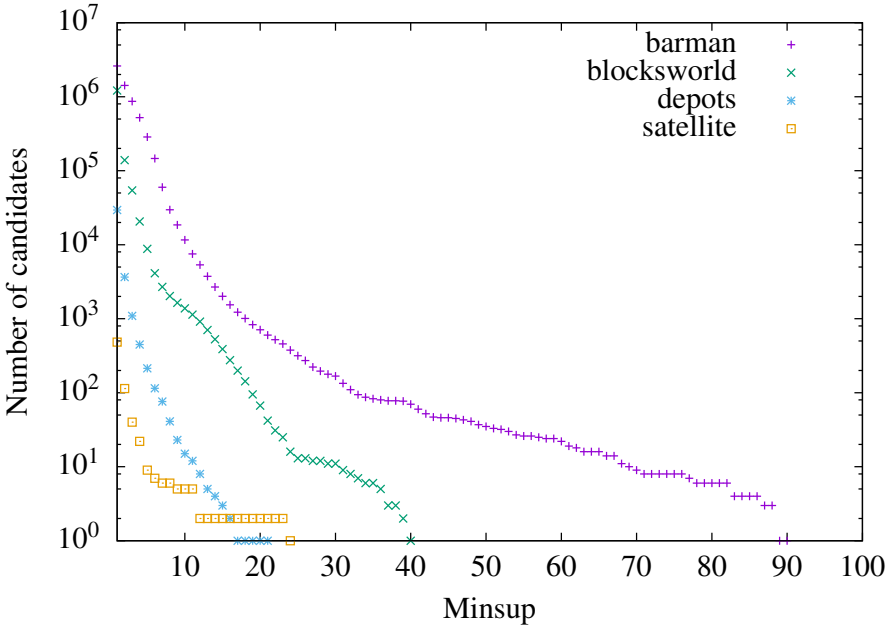


Figure 4.3: Sequences candidates. "Number of candidates" plotted in Log scale. "Minsup" plotted as percentages, e.g. a minsup of 0.1 corresponds to 10%

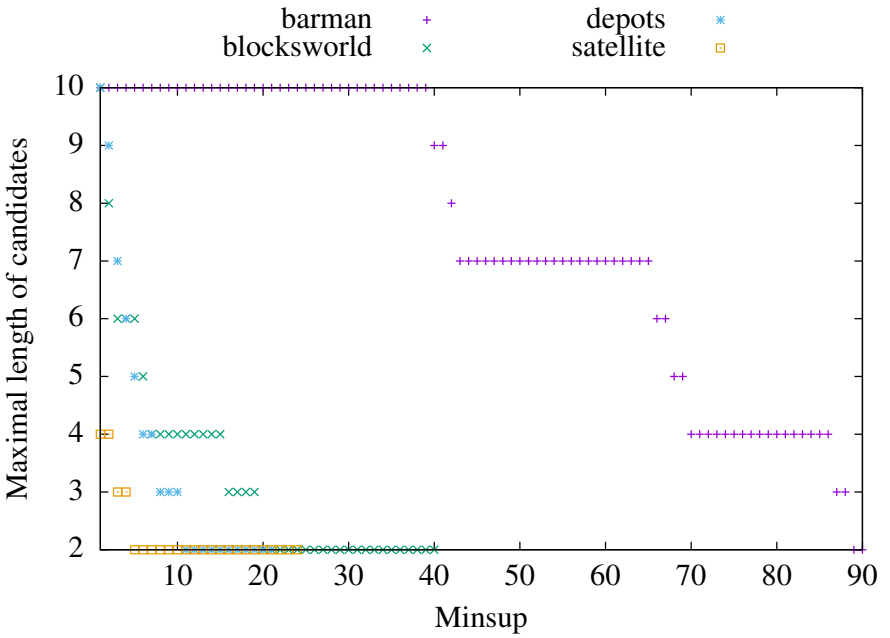


Figure 4.4: Maximal length candidates."Minsup" plotted as percentages, e.g. a minsup of 0.1 corresponds to 10%.

The maximal length of mined candidates is given in Figure 4.4. The maximal candidate length decreases with increasing values of the *minsup* parameter. It is not surprising to find longer candidates in lower values of the *minsup* parameter since it is more likely to have long sequences appearing a few times than to have them appearing frequently. Once again, it seems that the behaviour of sequences candidates is generally related to the domain.

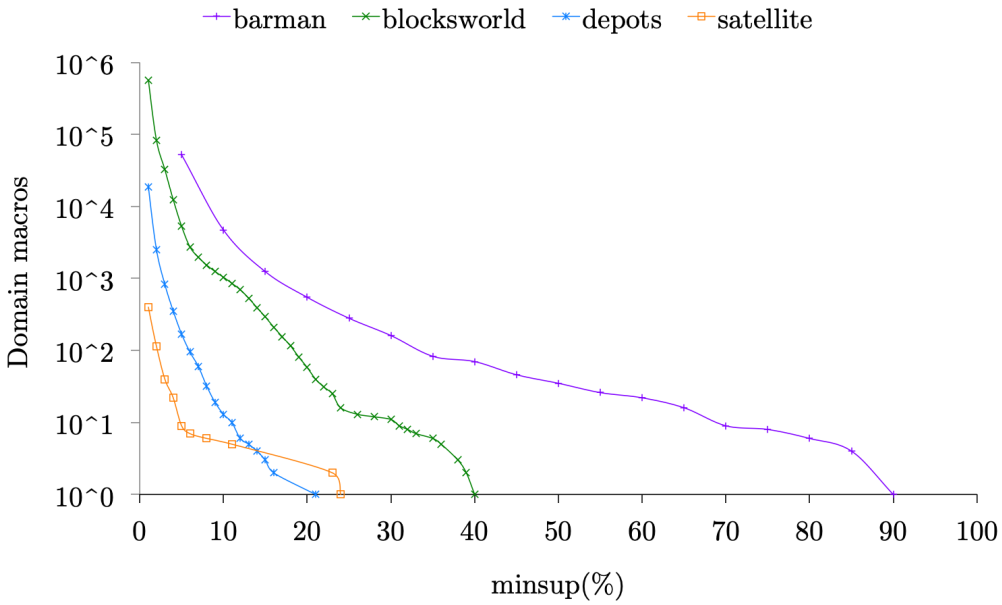


Figure 4.5: Number of macros added to the enhanced domain.

After the mining and filtering step, the macro-actions were constructed and added to the original domain as shown in Figure 4.5. For each domain, the x-axis is related to the enhanced domain containing the macro-actions that were constructed from the mined and filtered candidates by using the corresponding percentage of the *minsup*. The y-axis represents the number of macro-actions added to the original domain. For example, for the `depots` domain with a *minsup* of 0.1 (10%), its corresponding enhanced domain contains about ten macro-actions. These results do not always fit results from Figure 4.3. This is consistent with the macro-actions construction step, as not all mined and filtered candidates are valid macro-actions then not all candidates produce a macro-action.

The time performance for each domain and for each problem in the test set solved by using the original domain is shown in Figure 4.6. Here, problems are ordered in the x-axis with respect to their difficulty, i.e. the time required to solve it with the original domain, and the search time is showed in seconds in the y-axis using a log10 scale. If the problem was not solved either using the original domain or the enhanced domain, it was ignored. Also, in red, the problems not solved with the original domain but solved with at least one enhanced domain.

Now, we present the results obtained by solving the problem test set with the original

domain and with the enhanced domains. These results are represented in an easily understandable way by using the planning time metric, the space size metric and the plan quality metric. Each *minsup* value from the x-axis represents the enhanced domain obtained by using this value, and the corresponding point in the y-axis represents the gain obtained by using the enhanced domain.

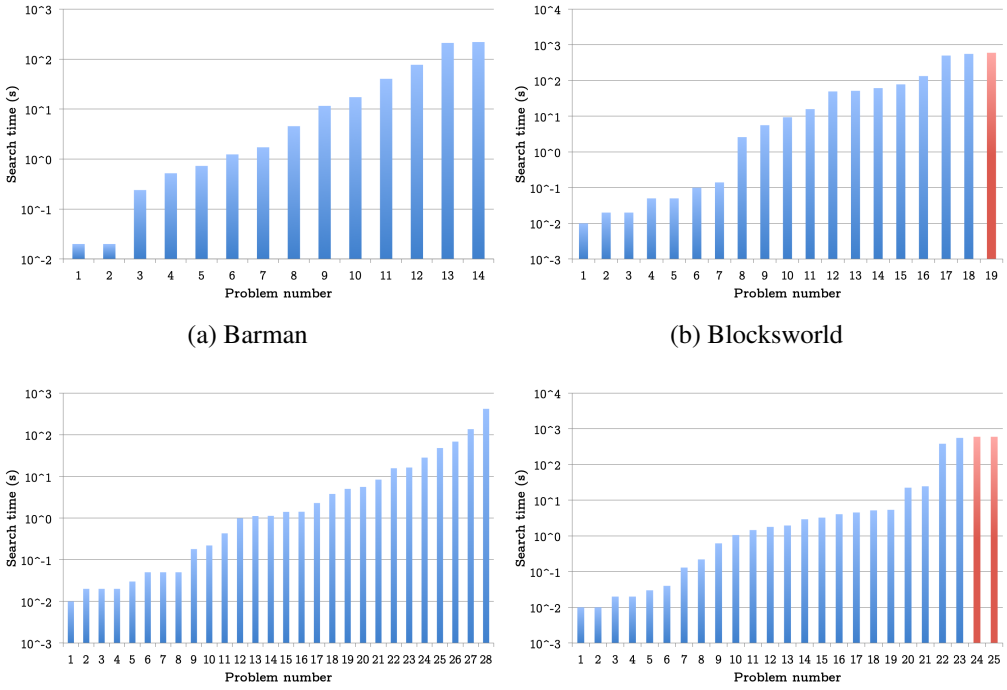
First, Figure 4.7 displays the planning time impact G_T of the enhanced domains compared to the original domain. To facilitate the visualisation, the results of `barman` enhanced domains above a *minsup* of 0.5 have not been displayed since they slightly differ from the results of the enhanced domain at 0.5. We observed that the `barman` domain exhibits a gain using the enhanced domains constructed from candidates mined with a *minsup* between 0.1 and 0.2. `blocksworld` domain also exhibits a gain but its behaviour is unpredictable. Finally, *depots* and *satellite* domains do not show a gain but a loss compared to the original domain. In general, the gain was the lowest when using the lowest *minsup*. This may be due to the large amount of macro-actions added to the domain when using low *minsup* values.

For readers used to analysing the IPC score, we also represented it in Figure 4.8. Each *minsup* value from the x-axis represents the enhanced domain obtained by using this value, and the zero value represents the original domain. The corresponding point in the y-axis represents the score obtained by using the enhanced domain. We displayed in red, the domain with the highest score.

Second, the average impact in the final space size G_N of each enhanced domain compared with the original domain is given in Figure 4.9. This impact is displayed using a log scale. The final space size was generally well impacted. As the use of macro-actions is supposed to provide a way to go deep quickly into the search space, this behaviour was expected. On the other hand, the negative impact in the `satellite` domain can be attributed to the lack of utility of the macro-actions added in that domain despite their support.

Third, Figure 4.10 shows G_Q, the average impact in the length of the found plans by using the enhanced domain, in comparison to the length of the found plans by using the original domain. Globally, plan length slightly increased when using the enhanced domains. The difference between the `barman` domain and the other domains is probably a consequence of the number of macro-actions used and their length.

To sum up, these results do not exhibit any clear link between the support and the gain, in this perspective, the support parameter may not be descriptive enough to decide on the utility of a macro-action.



(a) Barman

(b) Blocksworld

(c) Depots

(d) Satellite

Figure 4.6: Search time performance per domain. In red, the problems not solved with the original domain but solved with at least one enhanced domain.

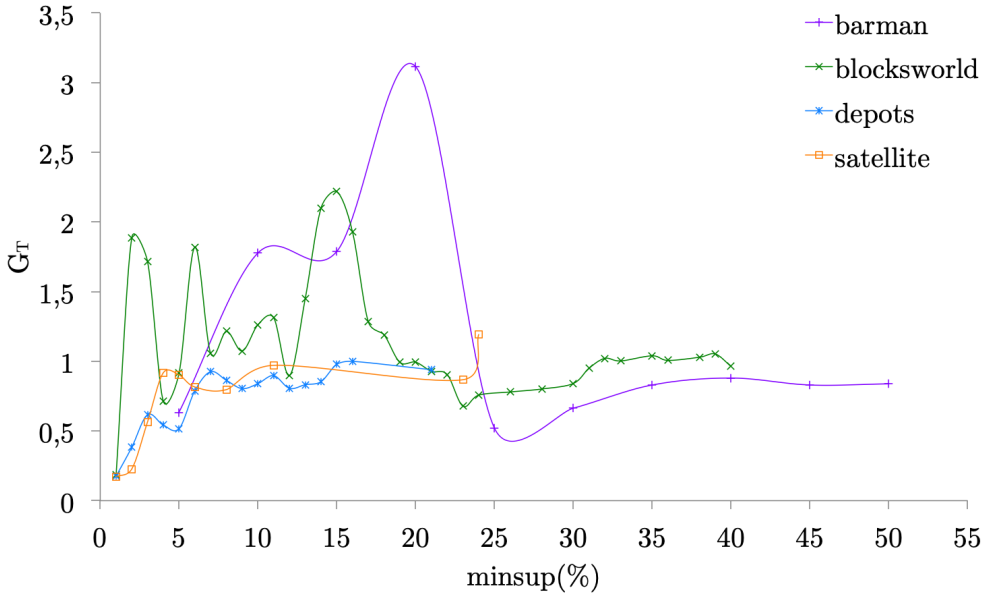
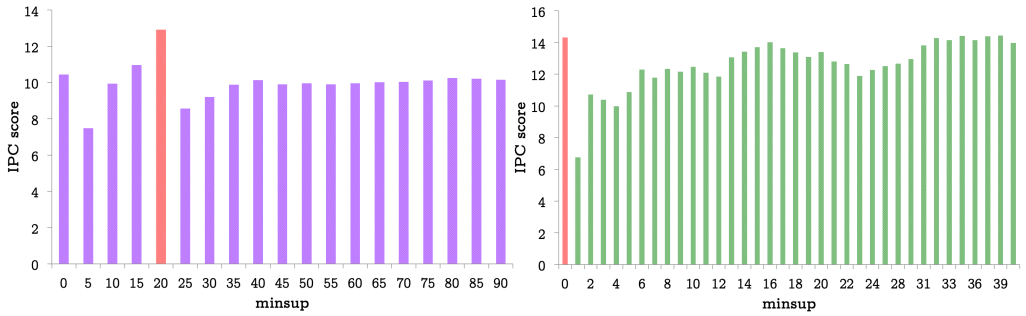
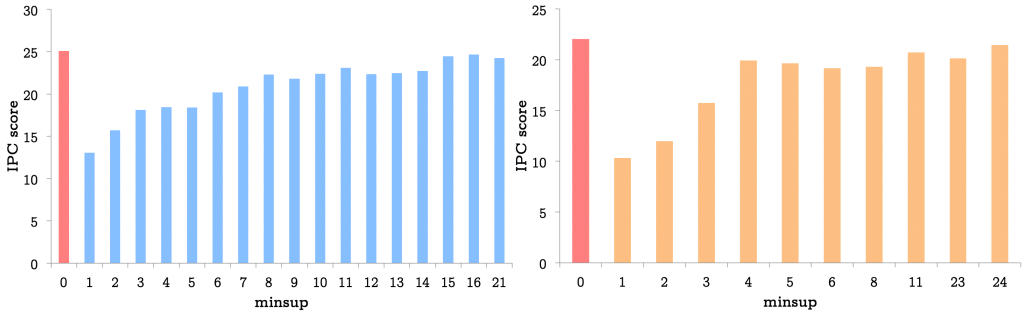


Figure 4.7: G_T per domain.



(a) Barman

(b) Blocksworld



(c) Depots

(d) Satellite

Figure 4.8: IPC score per domain. In red, the domain with the highest score.

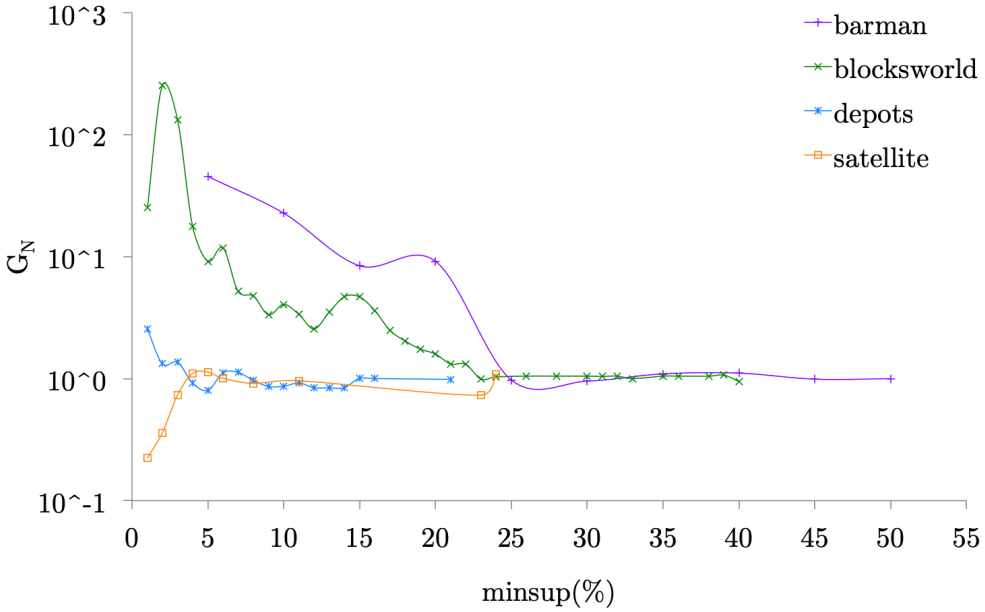


Figure 4.9: G_N per domain.

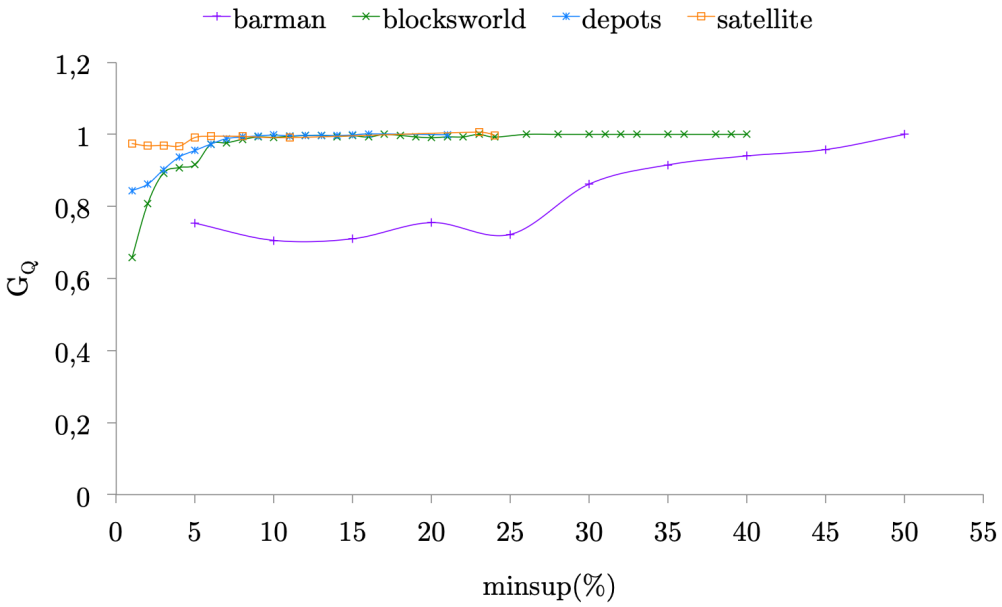


Figure 4.10: G_Q per domain.

4.8 Discussion

Our goal is the use of pattern mining techniques to mine in an easy way useful sequences of actions. Indeed, planning theory knowledge, common sense and previous works (Botea et al., 2004, 2005a; Newton and Levine, 2010) led us to infer that the most frequently encountered action sequences might be the most useful. Thus, we adopted the hypothesis of the support parameter as a good estimator to decide whether a sequence of actions is useful or not.

From this perspective, we sought to describe the behaviour of the support parameter in the mining of frequent action sequences. For that, we analysed the evolution, when increasing the support value, of elements such as the number of extracted candidates (Figure 4.3), the length of these candidates (Figure 4.4) and the order of magnitude of the macro-actions constructed from these candidates (Figure 4.5). The complexity (in terms of the length) and the number of macro-actions decrease when increasing values of *minsup*.

With that in mind, we originally expected that the gain in relation to the *minsup* parameter would have a shape close to a Gaussian bell curve. First of all, we presumed a negative gain when using low values of *minsup*. Indeed, a poor restriction regarding the frequency of the action sequences causes a large amount of useless candidates. Next, we looked for a positive gain as the value of *minsup* increases. That is because the number of candidates decreases but the action sequences appear more frequently (i.e. the candidates become more useful). Finally, we speculated about an interval guaranteeing useful candidates and a drop in the gain outside this interval.

Following this, we wanted to develop a method to learn the best support (i.e. where we could able to find useful candidates) by varying incrementally *minsup* values. However, even if the expected behaviour was observed for the `barman` domain (Figure 4.7 and Figure 4.9), for the other domains our results showed discrepancies. To conclude, the results showed that the support parameter may not provide a consistent selection of useful macro-actions.

4.9 Conclusion

We have shown that the use of pattern mining tools and concepts in the extraction of macro-actions is a promising lead. Indeed, the use of pattern mining algorithms has allowed us to extract, in an easy way, a huge number of sequences of actions (candidates).

We presented a macro-actions learning framework to learn useful sequences of actions as macro-actions to enhance a given domain. Also, in order to continue with our initial

hypothesis (See Section 4.1), we evaluated the support parameter concept as a good estimator for apriori macro-action utility. The results showed that the support parameter may not provide a consistent selection of useful macro-actions.

Additionally, some discrepancies were found in the results. This may be explained by the lack of: (1) macro-actions generality and (2) verification of the validity of the generated macros-actions. These leads will be explored in the next chapter.

5

Classical pattern mining applied to planning: shortcomings and solutions

Research is to see what everybody else has seen, and to think what nobody else has thought.

Albert Szent-Gyorgyi

5.1 Introduction

In the last chapter, we showed that the extraction of relevant macro-actions by using filter structures provided by pattern mining are not enough to provide satisfactory results by themselves. Indeed, the proposed macro-actions learning framework has various shortcomings restricting its practical use.

The purpose of this chapter is to explore these shortcomings, namely the lack of : (1) macro-actions generality and (2) verification of the validity of the generated macro-operators. We will also discuss the selection process. For each shortcoming, we first discuss its implications and then, we detail a remedial measure to address it. At the end of the chapter, we discuss what the support based selection process will become after using shortcoming remedial measures.

5.2 Macro-actions generality

To recapitulate, we encoded a set of plans as a sequence database to be able to use pattern mining algorithms to extract sequences actions candidates. This implies that every action in a plan represents an unique element in the sequence database. In other words, a same action in a plan, instantiated twice but with different objects, will represent two different elements in the sequence database.

This encoding results from an effort to mine frequent patterns in planning with existing pattern mining algorithms. In this respect, it should be noted that these algorithms are not planning-oriented. Thus, the resulting patterns correspond to sequences of actions.

We aimed to take advantage of the filter structures provided by pattern mining algorithms to extract frequent patterns. Therefore, after extraction, the only additional steps were to create macro-actions from extracted candidates and add them to the domain. This resulted in the creation of a large number of macro-actions (See Figure 4.5). As they represent only an instance and they have a limited applicability, these macro-actions do not increase significantly the branching factor.

However, the macro-actions added to each domain, still caused a negative impact in the planner performance since they were adding more processing time (to know if they are applicable or not) but they were not helping to solve the problem. To illustrate that, on the one hand, Figure 5.1 shows the average fraction of macro-actions added to the planner from the enhanced domain as instantiated operators for the problem. On the other hand, Figure 5.2 shows for each domain, the mean percentage of macros that have been used in solution plans. We can observe for the `barman` domain that almost (>90%) all the created macro-actions were added as operators for the problems, but a

low percentage (<2.8%) of them were used in the solution plan. The `blocksworld` and the `depots` domains showed a similar behaviour.

Although, the behaviour of the `satellite` domain seems completely different, it is important to note that very few macro-actions were extracted. On top of that, less than half of them were added as operators for the problems. Finally, very few macro-actions were used in the solution plan. All of this, led to no significant difference between the enhanced domain and the original domain for supports above 5%. And for very low supports ($\leq$ 5%), the behaviour is similar to other domains i.e. lots of macro-actions added but very few used.

All of these observations suggest that the created macro-actions may not be general enough to be used repeatedly through different problems. We will therefore present the other approach used: the generalisation of the extracted sequences of actions to macro-operators.

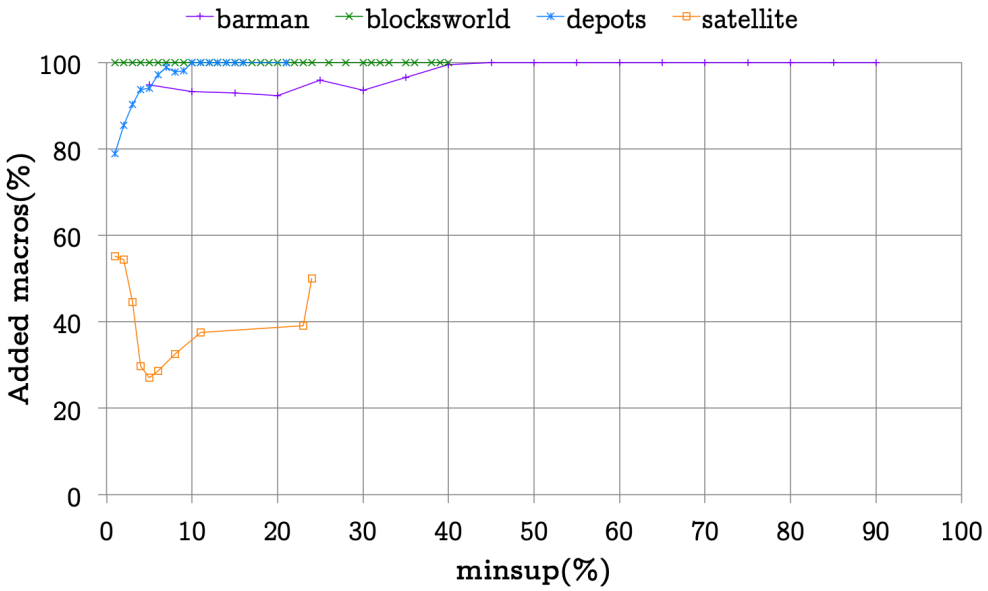
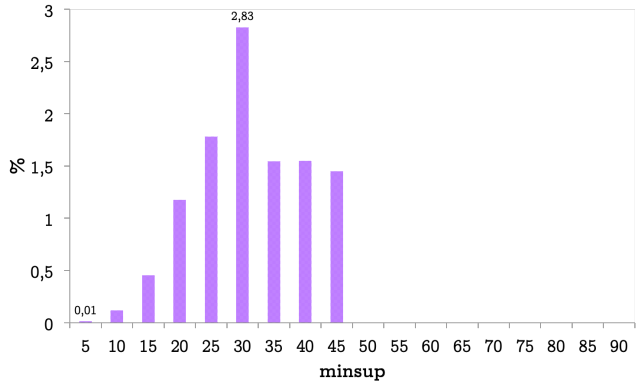
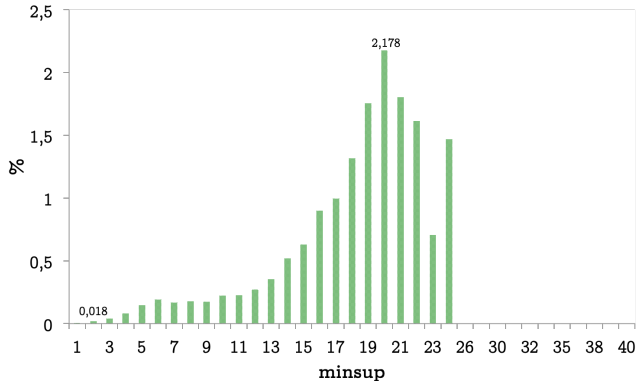


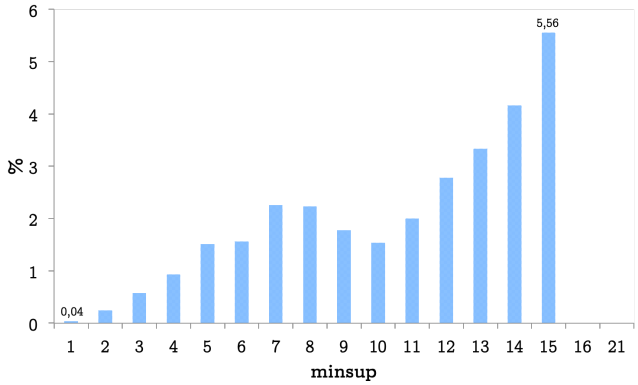
Figure 5.1: Average fraction of macros added from the enhanced domain as problem operators.



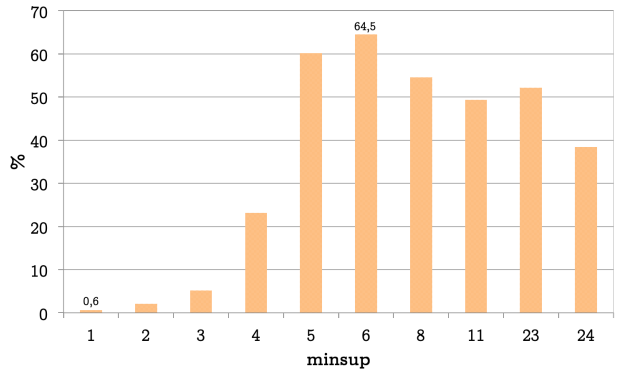
(a) Barman



(b) Blocksworld



(c) Depots



(d) Satellite

Figure 5.2: Mean percentage of macros used in solution plans for each domain. "Minsup" plotted as percentages, e.g. a minsup of 0.1 corresponds to 10%. The minimum and maximum percentage of macros are also displayed.

5.3 Macro-operators construction

From a set of sequence of actions, extracted by using pattern mining algorithms on a sequence database of plans (see Section 4.4), we built a set of macro-operators and we compute their respective support.

The pseudo code for the macro-operators construction is described in Algorithm 4. The algorithm is quite straightforward. It takes as input a set of sequences of actions for a given domain and the list of sequence identifiers, from the sequence database, where each sequence appears (see Table 5.1). For each sequence of actions $S = \{a_1, a_2, \ldots, a_n\}$, it attempts to construct a macro-operator m (line 4, procedure in Algorithm 5 described later). For each constructed macro-operator m, the algorithm adds the pair $< m, S >$ to the set of macro-operators only if it is a new macro-operator i.e. there is no macro-operator in the set of macros having the same parameters, preconditions and effects (line 6). Otherwise, it adds S as new sequence where the macro-operator m appears (line 7-10). Finally, for each macro-operator m in the set of macro-operators, the algorithm computes its support s by merging all the identifiers that correspond to the appearance of the sequences that created this macro-operator m (line 12-17). It gives as output a set of macro-operators with their respective supports (see Table 5.2).

Algorithm 4 Macro-operators construction

Input A set C of pairs $< S, L >$, S is a sequence of actions $S = \{a_1, a_2, \ldots, a_n\}$ where $n \geq 2$ and L is the list of the sequences ids where S appears.

Output A set M of pairs $< m, s >$, m is a macro-operator and s its respective support.

1: **function** CONSTRUCTMACROOPERATORS(C)
2: $D \leftarrow$ empty dictionary
3: **for each** sequence S in C **do**
4: $m \leftarrow createMacroOperator(S)$
5: **if** $m \neq$ null **then** ▷ if the creation of m succeeded
6: **if** $m \notin D$ **then** $D_{(key,value)} \leftarrow (m, \{S\})$ ▷ † explanation below
7: **else**
8: $newValue \leftarrow D(m)$ ▷ from D get value for key m
9: $newValue \leftarrow newValue \cup \{S\}$
10: $update(D_{(m,value)}, D_{(m,newValue)})$ ▷ upgrade value for key m
11: T $\leftarrow$ empty set
12: **for each** key-value pairs (k, v) in D **do**
13: **for each** sequence in v **do**
14: $nIds \leftarrow L$ from the pair $< S, L >$ in C such that S == sequence
15: T $\leftarrow$ T $\cup \bigcup_{id \in nIds} id$

16: $s \leftarrow$ number of elements in T
17: $M \leftarrow M \cup \{< k, s >\}$
18: **return** M

$\dagger : m \in D \iff \exists n \in D \mid pre(m) = pre(n), eff^{+}(m) = eff^{+}(n), eff^{-}(m) = eff^{-}(n)$

Sequence			Identifiers
unstack b3 b2	;	put-down b3	π_0 π_1 π_6 π_7 π_{15} π_{22} π_{27} π_{34} π_{35} π_{44} π_{56} π_{71} ...
unstack b3 b2	;	put-down b2	π_7 π_{35} π_{44} π_{116} π_{126} π_{141} π_{158} π_{162} π_{163} ...
unstack b13 b12	;	put-down b13	π_{386} π_{401} π_{404} π_{433} π_{454} π_{462} π_{486} π_{487} ...
unstack b1 b13	;	put-down b1	π_{388} π_{438} π_{446} π_{462} π_{480} π_{482} π_{493} π_{508} ...
pick-up b2	;	stack b2 b3	π_1 π_5 π_6 π_{11} π_{12} π_{27} π_{29} π_{30} π_{43} π_{48} π_{57} ...
pick-up b3	;	stack b3 b1	π_5 π_{10} π_{33} π_{46} π_{47} π_{48} π_{67} π_{79} π_{105} π_{113} ...
pick-up b3	;	stack b4 b3	π_{35} π_{44} π_{67} π_{95} π_{101} π_{129} π_{141} π_{167} π_{168} ...
put-down b4	;	stack b2 b4	π_{32} π_{46} π_{49} π_{67} π_{79} π_{87} π_{90} π_{91} π_{103} π_{116} ...

Table 5.1: Input sample for Algorithm 4, `blocksworld` domain.

Macro-operator	σ
macro-2-actions-3-1–1-1	174
macro-2-actions-3-1–2-1	55
macro-2-actions-0-2–3-1	132
macro-2-actions-1-2–1-1	52

Table 5.2: Output sample for Algorithm 4 from input Table 5.1, `blocksworld` domain. The macro-operator name is represented by macro-(number of actions)-actions-(id_a_1)-(...)-(id_a_n)-(id_commonObjects)-(id_commonParameters).

The pseudo code of the *createMacroOperator* procedure is described in Algorithm 5. It takes as input a sequence of two or more actions. It first verifies if there exist common objects between each pair of actions (line 2). After, it generalises each action to its respective operator (line 4-6). Then, it merges its two first operators into a single by using an oriented-operator version of the merge procedure presented in Chapter 4 in Algorithm 3. If S has more than two operators (line 9), it turns again the merge procedure taking as parameters the last merged operator and the next operator (line 10). It repeats this process until all operators are merged. Finally, it removes all predicates appearing at the same time in the set of preconditions and in the set of positive effects of the obtained macro-operator (line 12). It gives as output a macro-operator m.

The generalisation to macro-operators of the extracted sequences is intended to give the planner a way to have more opportunities to use shortcuts (aka macro-actions i.e. instantiated macro-operators) through different problems. In Figure 5.3, we observe the different size of each set obtained from the extracted candidates, either by creating macro-actions or by creating macro-operators.

Algorithm 5 *createMacroOperator* procedure

Input A sequence of actions $S = \{a_1, a_2, \ldots, a_n\}$ where $n \geq 2$.
Output A macro-operator m.

```
1:  function CREATEMACROOPERATOR(S)
2:      if not commonVariable(S) then return null
3:                  ▷ if there is no common objects between each pair of actions
4:      for each action a in S do
5:          O ← operator o from the domain such that a is an instantiation of o
6:          replace(a,O)
7:      m ← merge(o_1, o_2)
8:      i ← 2
9:      for each i < length(S) and m ≠ null do
10:         m ← merge(m, o_i)
11:         i ← i+1
12:     if m ≠ null then simplifyMacro(m)
13:     return m
```

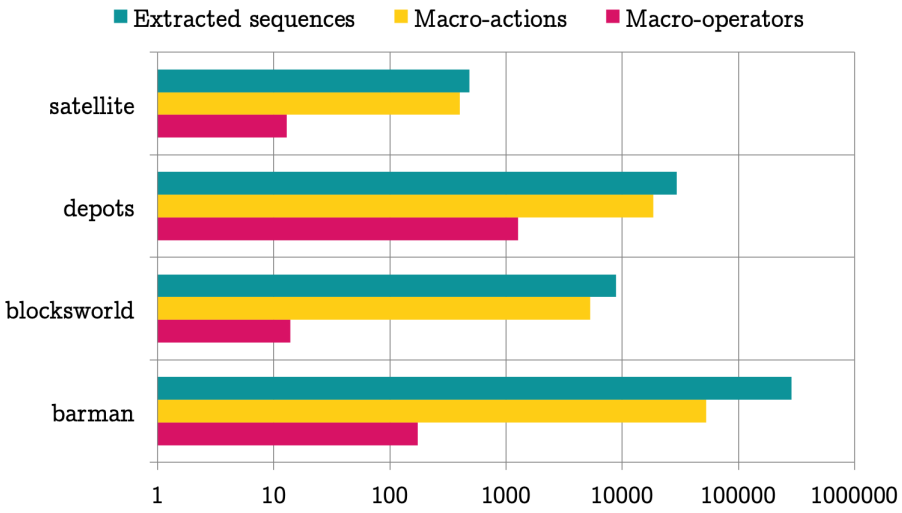


Figure 5.3: Size comparison: Extracted candidates vs macro-actions set vs macro-operators set. *minsup* = 5%

The size of the macro-operators set is considerably smaller. However, we should not forget that each macro-action adds only one instance when solving the problem while each macro-operator adds several instances. To sum up, we give more opportunities to the planner to use a shortcut when solving a problem by adding macro-operators to enhance the domains but we could also increase the branching factor. For this reason, a

selection process is mandatory to add only the most useful macro-operators.

Before addressing this problem, we will introduce a more important issue: the validity of the generated macro-operators.

5.4 Validity of the generated macro-operators

After analysing the obtained macro-operators, we identified that some created macro-operators were invalid. As a reminder, the Algorithm 4 allows the construction of macro operators, from the extracted candidates[1], only if some conditions are met, namely:

- The size of the sequence of actions (candidate) is greater than or equal to two i.e. $S = \{a_1, a_2, \ldots, a_n\}$ where $n \geq 2$.
- There exist common objects between each pair of actions of the candidate e.g. `pick-up B stack B C`
- The merge procedure is not null. In other terms, a candidate can be merged $\iff$ $\forall k \in [\![1, n-1]\!], negativeEffects(merge(\{a_1, \ldots, a_k\})) \cap preconditions(a_{k+1}) = \emptyset$ where $merge(\{a_1, \ldots, a_n\}) = merge(merge(\{a_1, \ldots, a_{n-1}\}), a_n)$.

Despite these conditions, we noticed that some of the created macro-operators were invalid because of the incompatibility of some predicates. Table 5.3 shows the results of the macro-operators analysis. For `blocksworld` and `satellite` domains, all created macro-operators were studied. For `barman` and `depots` domains, an estimation was done from a sample of the created macro-operators.

Domain	# Macro-operators	Invalid macros	± p
barman	173	25%	10%
blocksworld	14	21%	0
depots	1273	22%	10%
satellite	13	15%	0

Table 5.3: Results of the validity of created macro-operators per domain.

Additionally, Figure 5.4 shows an example of the identified problem. At the beginning, we have the extracted candidate `unstack b3 b2, put-down b2`. *How can we ensure,*

[1] These are obtained by using pattern mining algorithms and more specifically, closed frequent patterns algorithms with allowed gaps.

without the help of a human expert, that this sequence create an interesting macro-operator?.

First, the algorithm checks if the sequence has two or more actions, and in this case, it does. After, the algorithm validate that there exists a common object between each pair of actions i.e. `b2`. Then, it translates every action of the sequence with its respective operator from the domain and it merges both operators. However, the resulting macro-operator has two conflicting predicates, namely `handempty` and `holding ?X1`.

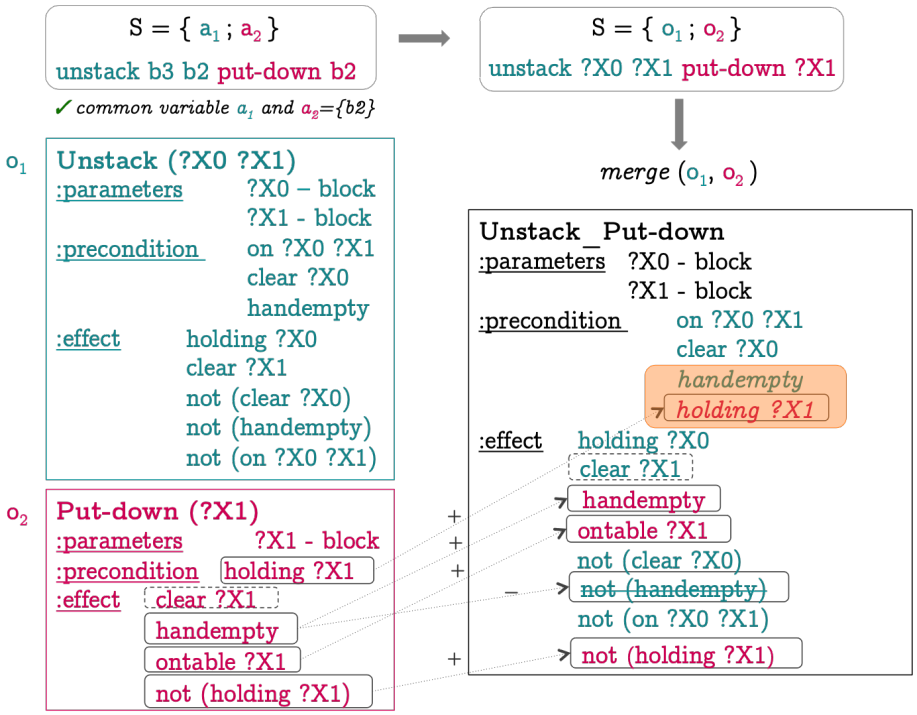


Figure 5.4: Macro-operator `Unstack_Put-down` with incompatible predicates `handempty` and `holding ?X1`

This kind of incompatibility is easily understandable to a human being but *how can we make the machine understand that these two operators are incompatible?*

5.5 Problematic macro-operators: Definition

In the following, we define the three types of problematic macro-operators: the incompatible predicates' macro-operators, the useless macro-operators and the redundant macro-operators.

Incompatible predicates' macro-operators

These are macro operators that cannot be applied during planning search. The main reason is the incompatibility of their predicates. From now on, we refer to them as *incompatible macro-operators*.

Definition 5.1.
Let $pre(m) = \{p_1, \ldots, p_n\}$ be the set of predicates belonging to the set of preconditions *pre* of the macro-operator m. A predicate p_1 is incompatible with predicate p_2, denoted as $p_1 \mid p_2$[2], if and only if they cannot coexist in the same state s.

From this point forward, for an operator (resp. macro-operator) we denote the set of its preconditions as $pre(o)$ (resp. $pre(m)$), the set of its positive effects as $add(o)$ (resp. $add(m)$) and the set of its negative effects as $del(o)$ (resp. $del(m)$).

Definition 5.2.
An incompatible macro-operator has an unattainable precondition, i.e. there is at least one pair of predicates that cannot coexist in the same state, $\exists(p_1, p_2) \in pre(m) \times pre(m), p1 \mid p2$.

Example Let us consider the macro-operator `unstack_put-down` in Figure 5.4. This macro-operator is an *incompatible macro-operator* since two of its predicates in the precondition set are incompatible, namely `handempty` and `holding ?X1`. ■

Useless macro-operators

These macro-operators can be applied during planning search but their application is equivalent either to do nothing or to use a primitive operator.

Definition 5.3.
A macro-operator m is useless in the two following cases:

1. If the application of a macro-operator m to a state s is equivalent to the same state s, i.e. $\gamma(s,m) = s$. In terms of predicates, it translates to the Equation (5.1)

$$(add(m) \subseteq pre(m)) \wedge (\forall p_1 \in del(m), \exists p_2 \in pre(m) : p_1 \mid p_2) \tag{5.1}$$

   meaning that all the positive effects of the macro-operator m were already present in the current state since they are equivalent to the set of preconditions. Additionally, for all the predicates p_1 in the set of negative effects there exists an incompatible predicate p_2 in the set of preconditions, and as a consequence, p_1 is useless because it could not have been true.

[2]The Sheffer's stroke symbol was chosen because its representation in propositional logic (Smullyan, 1995). Indeed, it represents that two propositions can not be true at the same time.

2. If the macro-operator m is equivalent to a primitive operator o. In terms of predicates, it translates to the Equation (5.2).

$$\begin{aligned}&\exists o,(add(m)-add(o)\subseteq pre(m))\wedge\\&\quad(pre(o)\subseteq pre(m))\wedge\\&\quad(\forall p_1\in del(m)-del(o),\exists p_2\in pre(m):p_1\mid p_2)\end{aligned}\tag{5.2}$$

meaning that the macro-operator m and the operator o are equivalent, if and only if, any additional predicate in the set of positive effects of m, compared to the set of positive effects of o, is useless because it already appears in $pre(m)$. Also, when m can be applied, o can as well. Finally, for any additional predicate p_1 in the set of negative effects of m, compared to the set of negative effects of o, there exists an incompatible predicate p_2 in the set of preconditions and p_1 is useless since it could not have been true.

Example Let us consider the macro-operators `pick-up_put-down` and `put-down_pick-up_stack` in Figure 5.5. The former is equivalent to do nothing in a state s, we have $add(m)\cap pre(m)=\emptyset$ and $del(m)=\emptyset$ since the highlighted negative effect `holding ?X0` is useless (it is incompatible with the `handempty` predicate in the set of preconditions). The later is equivalent to the primitive operator `stack`, and because the highlighted negative effect is useless (it is incompatible with the `holding ?X0` predicate in $pre(m)$), we have that $pre(m)\cap pre(stack)=\emptyset\wedge add(m)\cap add(stack)=\emptyset\wedge del(m)\cap del(stack)=\emptyset$. For all these reasons, the two macro-operators are *useless macro-operators*. ■

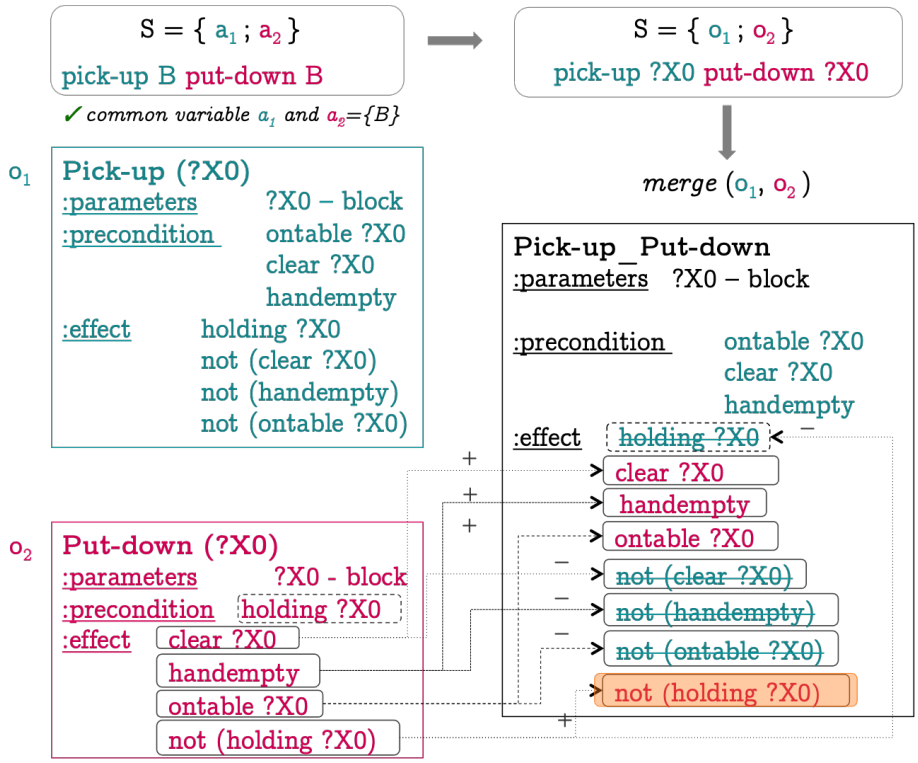
Redundant macro-operators

These macro-operators can be applied during planning search but their application is equivalent to use a simpler macro-operator i.e. a macro-operator with less actions.

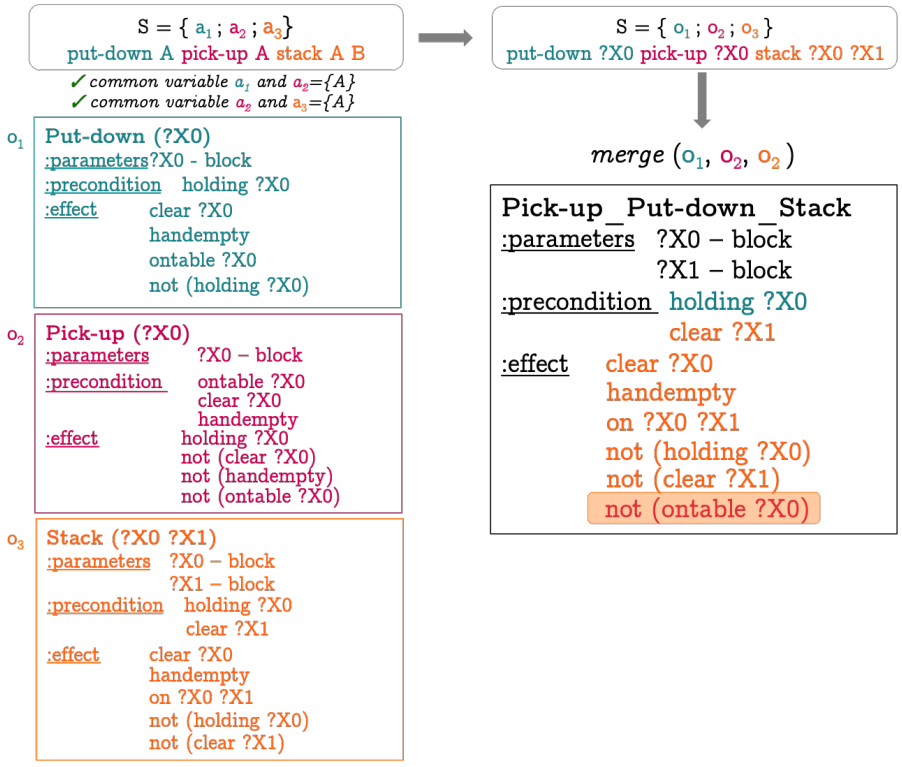
Definition 5.4.
A macro-operator m_1 is *redundant* if

$$\begin{aligned}&\exists m_2,(add(m_1)-add(m_2)\subseteq pre(m_1))\wedge\\&\quad(pre(m_2)\subseteq pre(m_1))\wedge\\&\quad(\forall p_1\in del(m_1)-del(m_2),\exists p_2:p_1\mid p_2)\wedge\\&\quad(nbActions(m_2)\leq nbActions(m_1))\end{aligned}\tag{5.3}$$

meaning that the macro-operator m_1 is redundant compared to the macro-operator m_2, if and only if, any additional predicate in the set of positive effects of m_1, compared to the set of positive effects of m_2, is useless because it already appears in $pre(m_1)$. Also, when m_1 can be applied, m_2 can as well. Likewise, for any additional predicate p_1 in the set of negative effects of m_1, compared to the set of negative effects of m_2, there exists an incompatible predicate p_2 in the set of preconditions and p_1 is useless since it could not have been true. Finally, the number of operators of m_1 is greater or equal than the number of operators of m_2.



(a) Macro-operator `pick-up_put-down` does not change the state *s*.



(b) Macro-operator `put-down_pick-up_stack` is equivalent to the primitive operator `stack`.

Figure 5.5: Examples of *useless macro-operators*. Highlighted predicates should not be taken into account.

Example Let us consider the macro-operator m_1 in Figure 5.6. This macro-operator is a *redundant macro-operator* since there exists a macro-operator m_2 which is simpler than $m1$. Indeed, both macro-operators have the same preconditions and effects but $m2$ is composed of less operators. The impact of both macro-operators during the search is the same but $m1$ would degrade the plan quality if it appears in the solution plan. ■

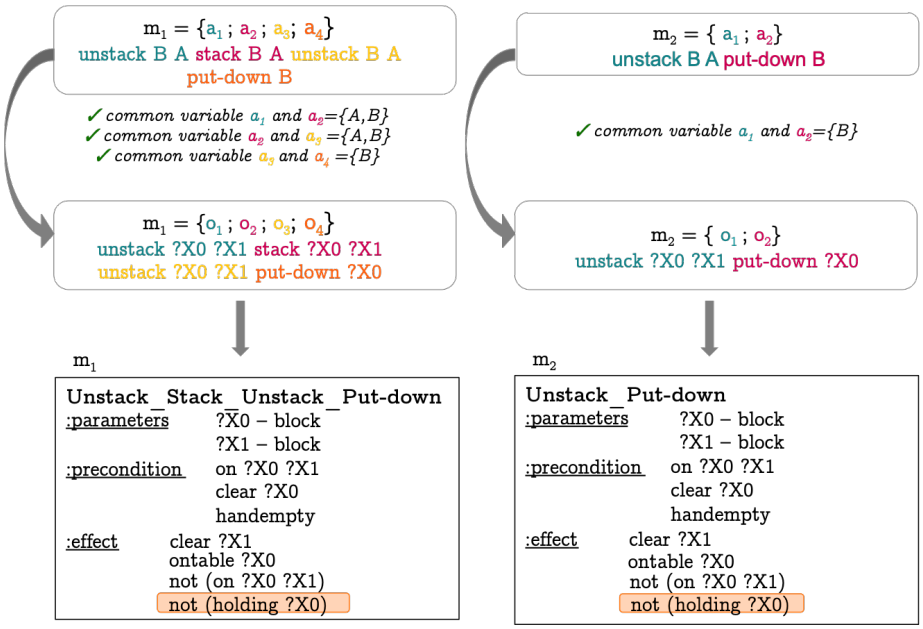


Figure 5.6: Redundant macro-operator `Unstack_Stack_Unstack_Put-down`. Highlighted predicates should not be taken into account.

5.6 Problematic macro-operators: Detection

To detect problematic macro-operators addressed in last section, we should be able to find the incompatible predicates. The idea behind finding incompatibilities for a predicate p_1 is:

- Simulate all the ways to obtain this predicate.
- Any predicate p_2 that was true at a given step, while p_1 was also true, is compatible with p_1.
- Any predicate p_2 that has been true at a given step, and that is not compatible with p_1 (i.e. there is no step where p_1 and p_2 were true at once), is incompatible with p_1.

In this regard, we present in Figure 5.7 a fully automatic method for detecting and eliminating problematic macro-operators. The pseudo code of this method is described in Algorithm 6. It takes as input a domain d and a set of macro-operators M and it passes them to the *extractIncompatibilities* procedure (pseudo code, in Algorithm 7, described later) to found incompatibilities based on a layered graph (line 2). After this extraction, it computes each problematic set (incompatible, useless, redundant) using the found incompatibilities, the domain and the set of macro-operators (line 3). Finally, it removes from the set of macro-operators, the detected problematic macro-operators (line 4).

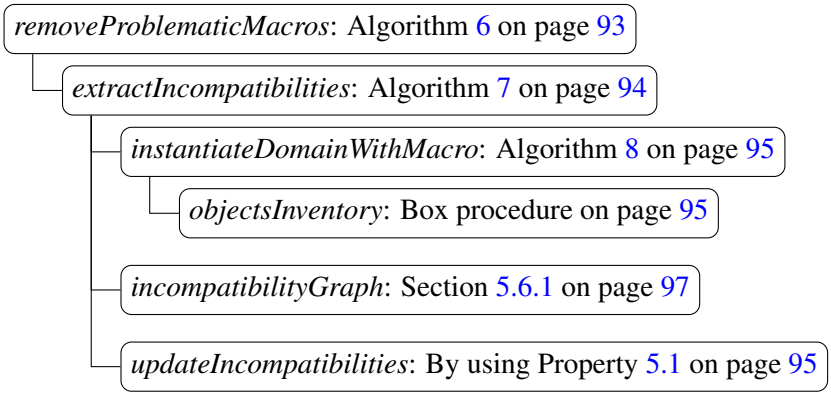


Figure 5.7: Explanation scheme for detecting and eliminating problematic macro-operators

Algorithm 6 Remove problematic macro-operators - Main algorithm

Input A domain d and a set M of macro-operators.

Output A set Mo of non problematic macro-operators.

1: **function** REMOVEPROBLEMATICMACROS(d, M)
2: $I \leftarrow extractIncompatibilities(d, M)$ ▷ see Algorithm 7
3: $Mo \leftarrow \bigcup_{f \in X} f(d, M, I)$ ▷ † explanation below
4: $Mo \leftarrow M \setminus Mo$ ▷ set difference
5: **return** Mo

† : $X = \{incompatibleMacro, uselessMacro, redundantMacro\}$

The pseudo code of the *extractIncompatibilities* procedure is described in Algorithm 7. It takes as input a domain and a set of macro-operators. For each macro-operator m, it instantiates the operators domain with respect to m and it instantiates m (line 3, procedure described later in Algorithm 8). Then, it creates a set of predicates from the preconditions of all instantiations of m (line 6). And, for each predicate p_1, it constructs a layered graph aimed to simulate all the ways to obtain p_1 by using the domain primitive

operators (especially their preconditions and their effects) (line 8). After the construction of this graph, it is used as a guideline to know which sequences of actions to test and on which states, and by analysing the output of this process, it extracts some incompatibilities. Then, it uses the transitivity property of incompatibilities (Property 5.1) to find more incompatible predicates (line 9). Finally, it returns to the main algorithm a list of predicates, and for each predicate, a list of its incompatible predicates.

Algorithm 7 Extraction of Incompatibilities

Input A domain d and a set M of macro-operators.

Output A dictionary I of pairs $< p, i >$, p is a predicate and i is a set of predicates i where $\forall k \in i, p \mid k$.

1: **function** EXTRACTINCOMPATIBILITIES(d, M)
2: **for each** macro-operator m in M **do**
3: $(A, Ma) \leftarrow instantiateDomainWithMacro(d, m)$
4: $predicatesToDo = \{\emptyset\}$
5: **for each** instance k in Ma **do**
6: $predicatesToDo \leftarrow predicatesToDo \cup \bigcup_{p \in pre(k)} p$
7: **for each** predicate p in $predicatesToDo$ **do**
8: update I with $incompatibilityGraph(p, A)$ ▷ see Section 5.6.1
9: $updateIncompatibilities(I)$ ▷ By using the Property 5.1
10: **return** I

The predicates analysed by the incompatibility graph (line 8 in Algorithm 7) come from a series of possible parameter configurations of the analysed macro-operator. This is especially useful when there are inclusion relationships between the object types in a domain. Our method requires, in order to obtain these predicates, to instantiate the primitive operators and the macro-operator taking into account the parameters of the latter. The pseudo code of the *instantiateDomainWithMacro* procedure is described in Algorithm 8.

It takes as input a domain d and a macro-operator m. It computes the objects inventory (see description in the box below) for each primitive operator in d. It also does this computation for m and it merges both results (line 3-6). After that, it instantiates each primitive operator (line 7-9) and the macro-operator (line 10) by using the computed objects inventory. It gives as result, a set of instantiated operators from d and a set of instances of the macro-operator m.

Algorithm 8 Domain instantiation relative to a macro-operator

Input A domain d and a macro-operator m.

Output A set A of instantiated operators from domain d and a set Ma of instances of the macro-operator m.

1: **function** INSTANTIATEDOMAINWITHMACRO(d, m)
2: $D \leftarrow$ empty dictionary
3: **for each** operator o in d **do**
4: update D with $objectsInventory(o)$
5: $D_m \leftarrow objectsInventory(m)$
6: update D with D_m ▷ † explanation below
7: **for each** operator o in d **do**
8: $listA \leftarrow$ list of actions from instantiating o by using D
9: add $listA$ to A
10: $Ma \leftarrow$ list of macro-actions from instantiating m by using D
11: **return** (A, Ma)

† : $\forall k \in (D \cap D_m), D(k) = max(D(k), D_m(k))$

Procedure (*objectsInventory*: Objects inventory computation)

This procedure aims to define for each type of object, how many of them are necessary to make the operators' instances sufficiently representative and especially to do not prevent some states from existing. First, it lists all object types and the inclusion relationships of the domain. It creates an objects inventory and it assigns zero to all. Second, it updates the number of objects per type with the maximal value from the list of objects for each operator. Third, each meta-type is set to zero in the inventory and its value is added to each one of its sub-types because the operators' instances will not contain meta-types. Finally, it adds a level of freedom to each object in the inventory whose value is greater than zero. As we cannot have an infinite number of objects, having free objects allows us not to restrict the instantiation.

Example Let us consider the `depots` domain, its object types in Figure 5.8 and its following primitive operators:

```
lift(?x-hoist ?y-crate ?z-surface ?p-place)
drive(?x-truck ?y-place ?z-place)
drop(?x-hoist ?y-crate ?z-surface ?p-place)
load(?x-hoist ?y-crate ?z-truck ?p-place)
unload(?x-hoist ?y-crate ?z-truck ?p-place)
```

The objects inventory process is shown in Figure 5.9. ■

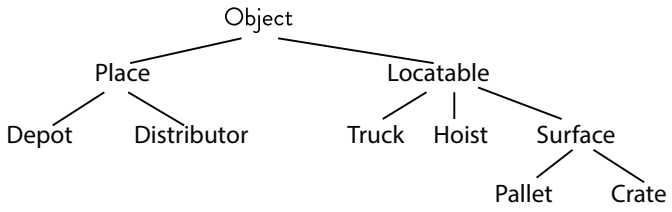


Figure 5.8: `depots` object types

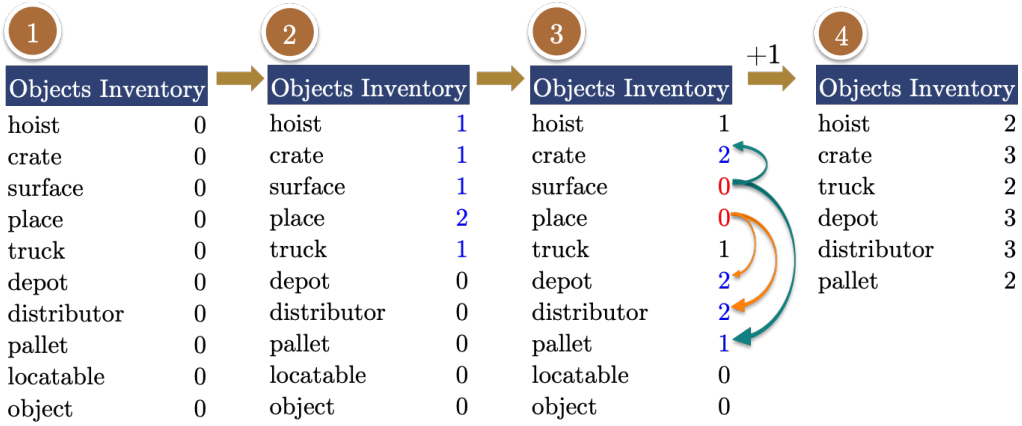


Figure 5.9: Objects inventory example

Property 5.1.
The transitivity property of incompatibilities defined in Equation (5.4) says that having a pair of predicates (p_1, p_2) and knowing its incompatibilities I, if p_1 (resp. p_2) is true, there is no action giving p_2 (resp. p_1) and if the pair (p_1, p_2) does not belong to the positive effects of any action, this implies that p_1 and p_2 are incompatible predicates.

$$\begin{aligned} \exists (p_1, p_2), \forall a \in A : p_1 \in add(a), \forall b \in A : p_2 \in add(b), \\ I(p_2) \cap pre(a) \neq \emptyset \wedge I(p_1) \cap pre(b) \neq \emptyset \wedge \\ \nexists c \in A : \{p_1, p_2\} \subseteq add(c) \implies p_1 \mid p_2 \end{aligned} \tag{5.4}$$

where A denotes the set of all possible actions and $I(p)$ denotes the set of all incompatibilities known for predicate p.

Example Let us consider the predicates p_1 et p_2 in Figure 5.10. They are incompatible predicates if knowing part of their respective incompatibilities, we are not able to find an action a (resp. b) that adds p_1 (resp.p_2) when p_2 (resp. p_1) is true. In this example, if a (resp. b) is the only action giving p_1 (resp. p_2), we can conclude that $p_1 \mid p_2$. Otherwise, we would have to check for all a (resp. b) giving p_1 (resp. p_2). ■

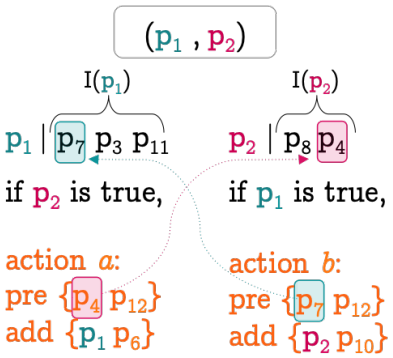


Figure 5.10: Example of the transitivity property of incompatibilities.

5.6.1 Incompatibility graph implementation

The implementation of the incompatibility graph aims to find for a predicate p, the predicates that are incompatible with p. The implementation follows two main steps: the graph construction and the graph exploitation.

Graph construction

It consists in the construction of a layered graph. Note that a relaxed behaviour is allowed. The root of this graph is the predicate for which we are looking for incompatibilities.

We start with the root and we alternate between the expansion and the link rules until the stop condition is reached. Expansion rules allow to deduce the nodes in the layer $n+1$ from the layer n. While, link rules dictate which nodes will be connected between layer n and layer $n+1$.

By following the expansion rules, we alternate the graph layers between predicate layers and action layers, beginning with a predicate layer (root). We therefore have the following rules:

- If we consider that the layer L_n is a predicate layer, the layer L_{n+1} is an action layer and A is a set of instantiated operators. We expand the graph from a predicate layer L_n to an action layer L_{n+1} by adding as nodes in the layer L_{n+1} all the actions a that create the predicates in layer L_n (see Equation (5.5)).

$$L_{n+1} = \{a \in A : \exists p \in L_n, p \in add(a)\} \tag{5.5}$$

- If we consider that the layer L_n is an action layer, the layer L_{n+1} is a predicate layer and A is a set of instantiated operators. We expand the graph from an

action layer L_n to a predicate layer L_{n+1} by adding as nodes in the layer L_{n+1} all the predicates p belonging to the preconditions of the actions in layer L_n (see Equation (5.6).

$$L_{n+1} = \{p : \exists a \in L_n, p \in pre(a)\} \tag{5.6}$$

Additionally, if the root predicate is found in a layer other than layer 0, this predicate no longer generates any node.

In order to link the layers with each other, the following rules are followed:

- If we consider that the layer L_n is a predicate layer and the layer L_{n+1} is an action layer. We linked every node in the predicate layer L_n to a node in the action layer L_{n+1}, if and only if, the predicate node belongs to the positive effects of the action node (see Equation (5.7). From now, we denoted an element x linked to an element y as $x \frown y$.
  In addition, if a predicate node does not belong to the positive effects of any action node, this node predicate is removed from L_n (see Equation (5.8)).

$$p \in L_n \frown a \in L_{n+1} \iff p \in add(a) \tag{5.7}$$

$$p \in L_n : \nexists a \in L_{n+1}, p \frown a \implies L_n \setminus p \tag{5.8}$$

- If we consider that the layer L_n is an action layer and the layer L_{n+1} is a predicate layer. We linked every node in the action layer L_n to a node in the predicate layer L_{n+1}, if and only if, the action node has in its negative effects the predicate node (see Equation (5.9).

$$a \in L_n \frown p \in L_{n+1} \iff p \in del(a) \tag{5.9}$$

The stop condition for the graph construction is reached when, after expanding the graph, we find a predicate layer such that the entire layer (with all its predicate nodes) has already been seen (see Equation (5.10)). When the stop condition is reached, we construct the next action layer L_{n+1} and we stop.

$$\text{if } L_n : \exists k < n, L_k = L_n \wedge mod(n,2) = 0 \implies stop \tag{5.10}$$

Example Let us consider the `depots` domain, its primitive operators and the predicate `lifting hoist0 crate0` for which the graph aims to find the incompatible predicates. An example of the incompatibility graph is shown in Figure 5.11. ■

Graph exploitation

Instead of considering taking all actions at random, concatenating them in a random order and testing everything until found any incompatibilities, the idea behind the graph exploitation is to have a reduced research space.

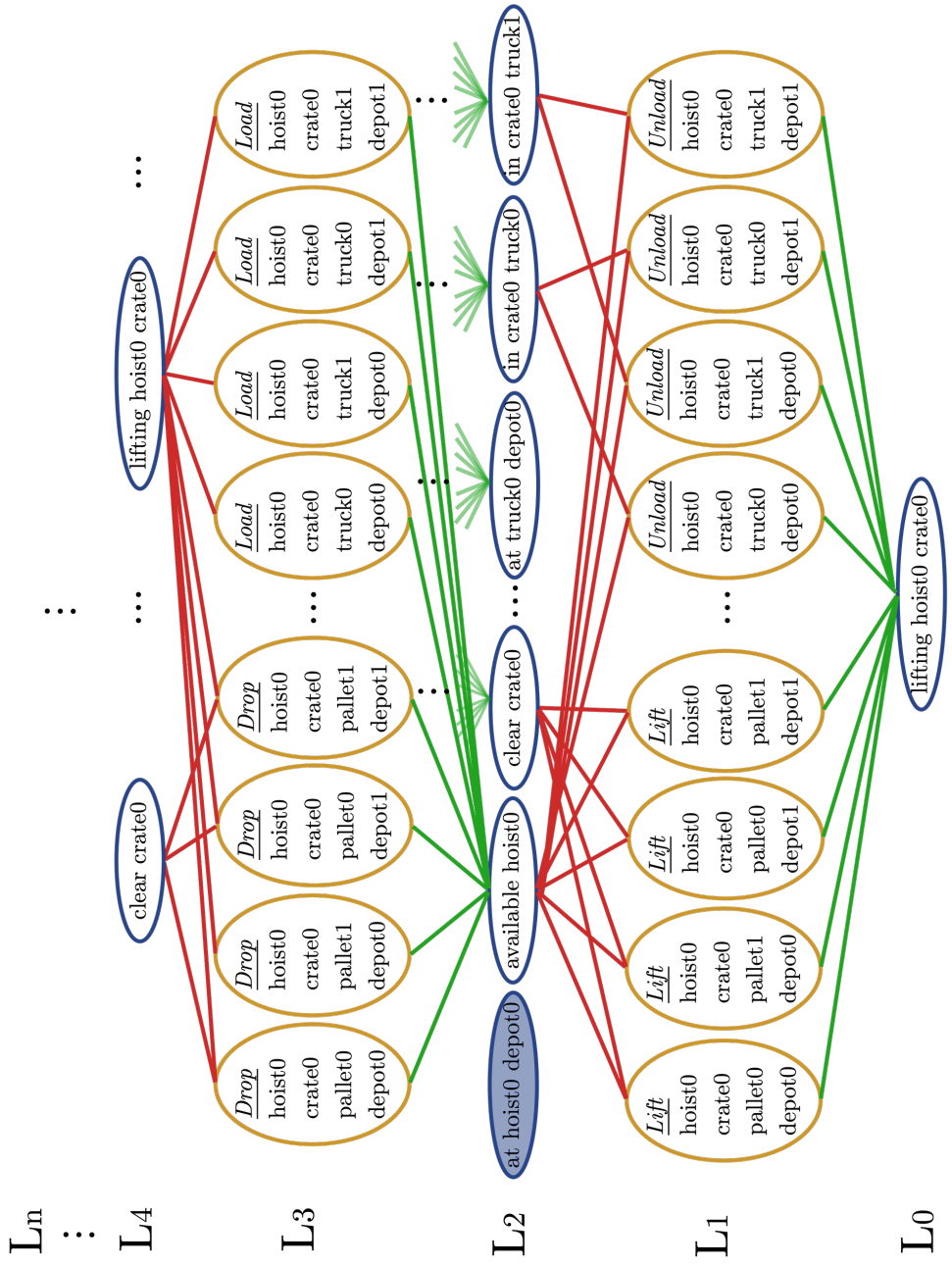


Figure 5.11: Incompatibility graph for predicate `lifting hoist0 crate0`. Green (resp. red) lines indicate the positive (resp. negative) effects. Yellow (resp. blue) circles are the actions (resp. predicates) and filled blue circles are predicates that will be removed from the layer.

Thus, after the construction of the layered graph, it is used as a guideline to know which sequences of actions to test and on which states. By analysing the output of this process, we extract some incompatibilities.

It is important to mention that in the construction step, when a new layer of predicates is created, if the root predicate p is found in the created layer, then this predicate is labelled as a *bud* node.

For each bud node, in a layer L_n, we define a set of initial states S_0. It consists of a set of predicates representing the minimum state that is required by each action, in the layer L_{n+1}, connected to the bud node, i.e. $pre(action) - del(action) \cup add(action)$. Then, we define a set of states S_1. Each state in S_1 will consist of a set of predicates that results from combining each state s of S_0 and each action of the next layer of actions, i.e. $s - del(action) \cup add(action)$ where $s \in S_0$. If the action cannot be applied, no new state is created. Also, if we found a state that already exists in the current set, it is not kept. We continue to define a set of states S_n by following the same idea, until we reach the last layer of actions. While doing this, we keep a record C of all the predicates found in the same state as the root predicate. We also keep a record I of all the predicates that were true in some state.

We repeat this process for each bud node by updating records C and I. In the end, we conclude on some incompatibilities by doing $I - C$. In other words, we keep the predicates that were true in some state but never coexisted with the root predicate.

Example Let us consider the layered graph for the `depots` domain, the predicate $p =$`lifting hoist0 crate0` and the *bud* node b (found in layer L_4) in Figure 5.12. We define S_0 in Equation 5.11 and Equation 5.12, where each state s_n in the set consist of the minimum state required by each action in layer L_5 connected to b. Notice that $operatorN$ denotes an operator instantiated with a configuration of objects identified by N (See PDDL-Code 5.1).

```
(:action Lift1
    :parameters (hoist0 crate0 pallet0 depot0)
    :precondition (and(at hoist0 depot0)
    (available hoist0)(at crate0 depot0)
    (on crate0 pallet0)(clear crate0))
    :effect (and(not(at crate0 depot0))
    (lifting hoist0 crate0)(not(clear crate0))
    (not(available hoist0))(clear pallet0)
    (not(on crate0 pallet0))))
```

PDDL-Code 5.1: Macro-action for the `blocksworld` domain

We have that $s_0 =$*(at hoist0 depot0)(lifting hoist0 crate0)(clear pallet0)*. Then, *(at hoist0 depot0)* and *(clear pallet0)* will be added to C record since they appear in the same state as *lifting hoist0 crate0*. They will be also added to I since they were true

in this state. Updating records C and I is fairly self-explanatory and will be left to the reader for exploration.

Then, we define S_1 in Equation 5.13 and Equation 5.14, where each state i_n consist of a set of predicates that results from combining each state s_n of S_0 and each action of the layer L_3. Finally, we define S_2 in Equation 5.15 and Equation 5.16, where each state j_n consist of a set of predicates that results from combining each state i_n of S_1 and each action of the layer L_1.

$$\begin{aligned} S_0 = \{ & \{pre(Lift1) - del(Lift1) \cup add(Lift1)\}, \\ & \{pre(Unload1) - del(Unload1) \cup add(Unload1)\} \} \end{aligned} \tag{5.11}$$

$$S_0 = \{ \{s_0\}, \{s_1\} \} \tag{5.12}$$

$$\begin{aligned} S_1 = \{ & \{s_0 - del(Drop1) \cup add(Drop1)\}, \\ & \{s_0 - del(Drop2) \cup add(Drop2)\}, \\ & \ldots, \\ & \{s_1 - del(Load4) \cup add(Load4)\} \} \end{aligned} \tag{5.13}$$

$$S_1 = \{ \{i_0\}, \{i_1\}, \ldots, \{i_n\} \} \tag{5.14}$$

$$\begin{aligned} S_2 = \{ & \{i_0 - del(Lift1) \cup add(Lift1)\}, \\ & \{i_0 - del(Lift2) \cup add(Lift2)\}, \\ & \ldots, \\ & \{i_n - del(Unload4) \cup add(Unload4)\} \} \end{aligned} \tag{5.15}$$

$$S_2 = \{ \{j_0\}, \{j_1\}, \ldots, \{j_n\} \} \tag{5.16}$$

■

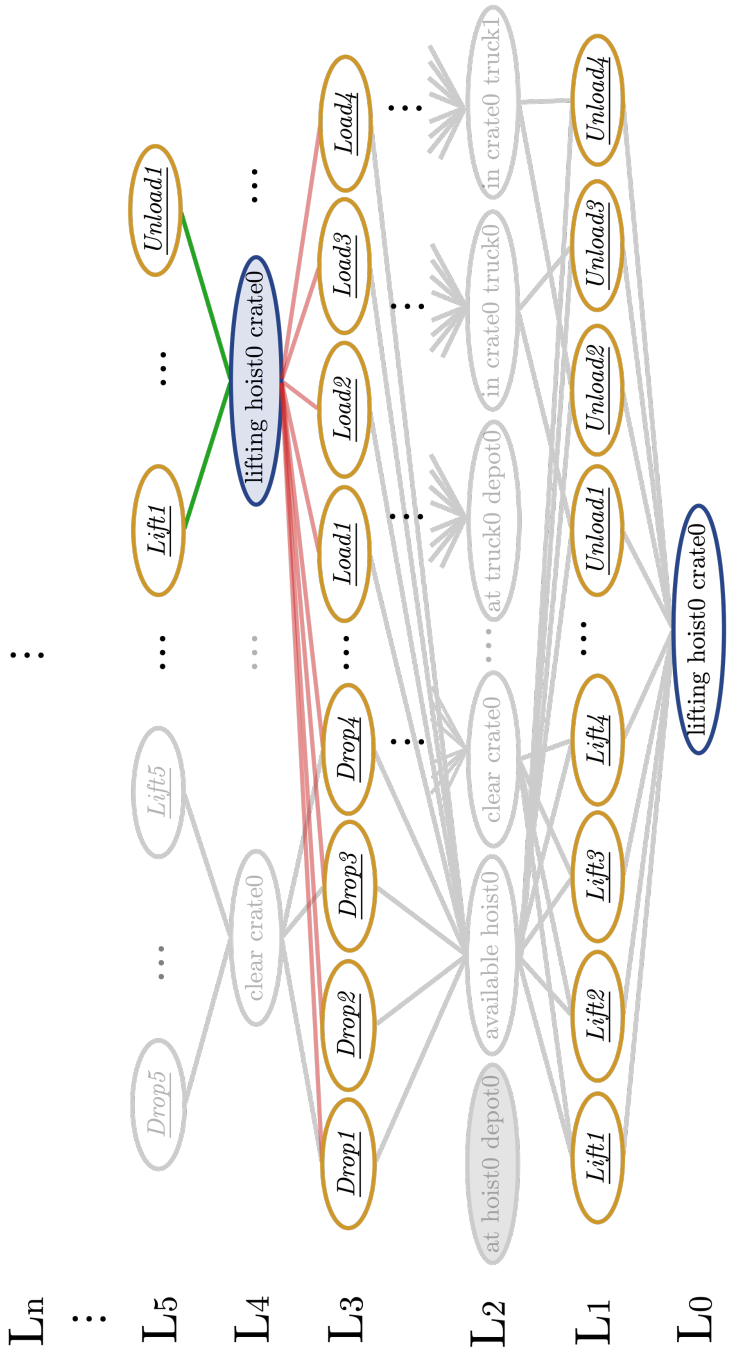


Figure 5.12: Exploiting the incompatibility graph for predicate p =`lifting hoist0 crate0`. Filled blue circle indicate the *bud* node. Yellow circles are the actions. Gray elements will not considered in the exploitation of the graph for predicate p.

5.6.2 Results for the removing method

We present in Table 5.4 a summary of the results for detecting problematic macro-operators from the graph-based approach. A more detailed report is presented in Appendix C.

Domain	M	I	U	R
Barman	152	24	7	18
Blocksworld	10	3	5	0
Depots	1175	565	157	308
Satellite	13	2	1	2

Table 5.4: Results from the graph-based approach to detect problematic macro-operators. "M" stands for number of macros, "I" number of incompatible macros found, "U" number of useless macros found and "R" number of redundant macros found.

Also, we show in Table 5.5 the results, in terms of percentages, for eliminating problematic macro-operators. Depending on the domain, we remove between 30% and 80% problematic macro-operators.

Domain	% Removed
Barman	32.2%
Blocksworld	80%
Depots	87.6%
Satellite	38.5%

Table 5.5: Percentage of removed problematic macro-operators per domain.

Our graph-based approach is proved to be successful in detecting and eliminating problematic macro-operators. Now, we need a measure allowing us to select from the remaining macro-operators those that will be added to the domain. In the next section, we will discuss about that selection process.

5.7 Selection process

After using the remedial measures such as the generalisation of the extracted candidates in macro-operators, and the graph-based approach to remove problematic operators, the

resulting macro-operators were numerous. Thus, we looked for a measure to decide which macro-operators will be added to the original domain.

First of all, we observed that the constructed set of macro-operators were neither representative nor significant enough in relation to the goals of each domain. For example, the goal of the `satellite` domain is to acquire desired images by diving the observation tasks between satellites. Although, we observe an instantiation of the operator `take-image` in every sequence (plan) of our sequence database (corpus of plans), not a single macro-operator contains this operator. Unless the satellite problems start with instruments already calibrated and turned towards the image to be taken, at least the `turn-to,take-image` candidate should be extracted by using pattern mining algorithms. However, it is not the case, for pattern mining algorithms there are not enough samples of this candidate having exactly the same objects to consider it as a frequent pattern.

Additionally, this observation raises another major flaw. If there are not enough samples of a sequence of operators instantiated with a set of objects A but there are enough samples of the same sequence of operators instantiated with a set of objects B, only the later will be extracted as a candidate. Then, when we will generalise candidates as macro-operators, the lack of all instances of the macro-operator will distort the computation of the support. Thus, the support obtained with this method is unreliable. Hence, in order to have an undistort computation of the support, one should use the pattern mining algorithms with a *minsup* equal to one appearance i.e. $minsup = \frac{1}{|C|}$ where $|C|$ is the size of the sequence database.

By following the last approach, we extracted all candidates with a *minsup* of one appearance but a new problem arised. Because we use closed sequential pattern mining algorithms, more interesting candidates e.g.`turn-to, take-image` have been consumed by longer candidates e.g. `switch-on, calibrate, turn-to, take-image`. Indeed, the later will be used less frequently than the former since the optimal strategy to solve satellite observation tasks consists of calibrating each satellite with a specific mode e.g. infrared and then, repeatedly, turn to each target to take its image. To avoid this problem, one should use pattern mining algorithms that allow us to extract all the frequent patterns.

Hence, to prevent the presented problems, one should use pattern mining algorithms with a *minsup* equivalent to *one appearance that extract all frequent patterns*. Though, that is equivalent to *do not use* pattern mining algorithms and to compute all combinations of actions for each plan in the sequence database. We can deduce that this approach is not suitable since it leads to a computational bottleneck. To conclude, we need an approach allowing to extract macro-operators from the sequence database and a measure deciding on their utility. Such an approach will be introduced in the next chapter.

5.8 Conclusion

In this chapter, we intended to explore the shortcomings of the macro-actions learning framework from the last chapter. We have shown that the generalisation in macro-operators of extracted candidates (obtained by using pattern mining algorithms on a sequence database) is essential. To do that, we presented a method that construct macro-operators from these candidates and compute its support value.

We have also presented a graph-based approach aiming to validate the created macro-operators. In this approach, we introduced the concept of *incompatible predicates* and we used it as a key to find problematic macro-operators (incompatible, useless and redundant). The graph-based approach proved to be successful in eliminating problematic macro-operators.

We discussed the problems of using classic pattern mining algorithms in planning. Despite the efforts, we find ourselves in a dead-end with the selection process because the pattern mining filtering structures are not adapted to planning. However, the presented approaches, namely the construction of macro-operators and the detection of incompatibilities, work well and could be used in other planning applications.

Finally, we concluded in the need for a novel approach allowing to extract macro-operators and assess in their utility. This approach will be presented in the next chapter.

6

Planning-oriented pattern mining: The METEOR framework

If you want something you have never had you must be willing to do something you have never done.

Thomas Jefferson

6.1 Introduction

Most existing work on macros are focused on macro-operators instead of macro-actions since they increase the probability of being used at planning time. In contrast, their number of parameters also increases the branching factor. To avoid this problem only useful macro-operators should be added to the domain.

In the last chapter, we explored an approach to generalise extracted candidates (obtained by using pattern mining algorithms on a sequence database of plans) in macro-operators. From a set of sequence of actions, extracted by using pattern mining algorithms on a sequence database of plans, we built a set of macro-operators and we compute their respective supports (i.e. the number of plans containing at least once the given macro-operator). Nonetheless, the information extracted from this approach did not allow for a reliable selection process. Indeed, as we have shown in Chapter 5 the computed support from this generalisation process was incorrect.

Thus, mining macro-operators from a set of plans requires an approach which ensures to find the frequent sequences of operators without a loss of information about their characteristics. Then, neither an operator can be dissociated from its objects nor a sequence of operators can disregard the relationship between operators' objects. Unfortunately, a central restriction in traditional pattern mining concerning its expressiveness is that each item is assumed to be a whole entity without any additional characteristics.

Also, a selection measure is essential to avoid undesirable side-effects of the use of macro-operators, namely the overload caused by increasing the branching factor in the search space when adding macro-operators.

In this chapter, we will first present the encoding limitations of traditional pattern mining algorithms in the extraction of macro-operators. After, we will present our METEOR framework to mine macro-operators from a set of plans and to select the optimal macro-operator set. Finally, the results obtained with this new approach will be shown and discussed.

6.2 Limitations of classical pattern mining algorithms in planning

Let us briefly summarise some key elements for this section. First, as the most popular algorithm for pattern mining[1], traditional pattern mining algorithms aim to answer to

[1] Apriori (Agrawal and Srikant, 1994).

the question: *how to identify the sets of items that occur more frequently in the analysed data?* Thus, these algorithms are designed to be applied on data containing entities that are part of a group e.g. transactions of items bought by costumers, transactions of user click-stream in a website, transactions of user logs in a set of communication services, etc. Moreover, if an order property exists between the elements of the analysed data, classical pattern mining provides algorithms for mining sequential data.

Second, in planning, each plan is composed of an ordered sequence of instantiated operators (aka *actions*) which in turn are composed of parameters (objects), e.g. $\pi = \langle$ `pick-up blockA, put-down blockA, pick-up blockB, stack blockB blockA`$\rangle$. We can improve the planning performance by learning sequences of operators that occur frequently in solution plans. Indeed, they reduce the depth of the search space. Thus, a question arises: *how to identify the sequences of operators that occur more frequently in plan solutions?*

In Chapter 4, we have proposed a framework which partially answers this last question. In other words, by using this framework, we were able to build macro-actions from the extracted sequences of *actions* that occur frequently in solution plans. To that end, we proposed an intuitive formalism to represent a set of solution plans as a sequence database (more details on the formalism in Chapter 4) and therefore, be able to apply sequential pattern mining algorithms. However, we observed that the created macro-actions were not general enough to be used repeatedly through different problems.

Most existing work on macros are focused on macro-operators instead of macro-actions since they increase the probability of being used at planning time. In contrast, their number of parameters also increases the branching factor, to avoid this problem only useful macro-operators should be added to the domain. With that in mind, we came up with two approaches to address the problem of generalisation.

We presented our first approach in Chapter 5.2. From a set of sequence of actions, extracted by using pattern mining algorithms on a sequence database of plans, we built a set of macro-operators and we compute their respective supports (i.e. the number of plans containing at least once the given macro-operator). Nonetheless, the information extracted from this approach did not allow for a reliable selection process. Indeed, as we have shown in Chapter 5 the computed support from this generalisation process is incorrect.

In this section, we aim to discuss the second approach[2], namely, to mine macro-operators directly from a set of solution plans. Because sequential pattern mining is applied on a set of sequences (i.e. a sequence database), we were looking for an encoding allowing to represent the set of plan solutions as a sequence database.

To show the complexity of the task, we will further discuss the encoding used on the first approach and we will present two others seemingly correct encodings.

[2]This approach is in line with our original question.

Let us consider as an example the following set of plan solutions:

$$\pi_1 = op_3(obj_1, obj_2); op_1(obj_1, obj_3); op_2(obj_1, obj_4)$$
$$\pi_2 = op_1(obj_5, obj_6); op_2(obj_5, obj_7); op_4(obj_7, obj_1)$$
$$\pi_3 = op_1(obj_1, obj_8); op_5(obj_{10}, obj_3); op_2(obj_1, obj_{11})$$
$$\pi_4 = op_6(obj_2, obj_9); op_1(obj_5, obj_3); op_2(obj_5, obj_7)$$

Here, each π_i corresponds to a plan solution for a given problem p_i and a planning domain D. It consists of an ordered set of operators $\{op_j\}$ which are instantiated with the objects $\{obj_k\}$.

Let us briefly recall the encoding used in the first approach: Each plan is considered as a sequence, and each instantiated operator, as an item in this sequence. It should be noted that the same identifier is assigned to two different items only if the operators and objects are exactly the same. By applying this encoding, we obtain the sequence database in Table 6.1.

Sequence_ID	Sequence
S001	{1},{2},{3}
S002	{4},{5},{6}
S003	{7},{8},{9}
S004	{10},{11},{5}

Table 6.1: Sequence database using the encoding from Chapter 4

We can observe that there is no sub-sequence appearing more than once. Although, we can easily detect four apparitions of the sub-sequence $op_1(\{obj_k\}); op_2(\{obj_k\})$, one for each plan in the original set of plan solutions. This discrepancy appears to be the result of the objects variability. In other words, there are not enough samples of this sub-sequence having *exactly the same objects* to consider it as a frequent pattern.

Hereafter, we will focus on the macro-operator $op_1(\{obj_k\}); op_2(\{obj_k\})$ having a support of 1. To get around the problem mentioned before, we can mine all frequent patterns with a minsup equal to one appearance i.e. $minsup = \frac{1}{4} = 0.25$. The result is presented in Table 6.2. By using these patterns to build macro-operators, we are able to find our macro-operator of interest and to compute the right value for its support. Even though it worked here, it is worth noting that:

- mining all frequent patterns with such a small minsup, is equivalent to compute all combinations of instantiated operators for each sequence.
- our sequence database is a small-scale example.
- the number of sub-sequences to analyse is a $O(2^{l_{max}})$ where l_{max} is the maximum sequence length in the analysed sequence database.

In a practical case we would expect a maximum plan length of 50 to 80 actions which would induce, considering that we can analyse one billion sub-sequences per second (based on an optimistic assumption), a computing time ranging from several weeks to several million years to analyse the longest plan only.

Sequential Pattern	σ	sup	Sequential Pattern	σ	sup
{1}	1	0.25	{7}	1	0.25
{1}, {2}	1	0.25	{7}, {8}	1	0.25
{1}, {2}, {3}	1	0.25	{7}, {8}, {9}	1	0.25
{1}, {3}	1	0.25	{7}, {9}	1	0.25
{2}	1	0.25	{8}	1	0.25
{2}, {3}	1	0.25	{8}, {9}	1	0.25
{3}	1	0.25	{9}	1	0.25
{4}	1	0.25	{10}	1	0.25
{4}, {5}	1	0.25	{10}, {5}	1	0.25
{4}, {5}, {6}	1	0.25	{10}, {11}	1	0.25
{4}, {6}	1	0.25	{10}, {11}, {5}	1	0.25
{5}	1	0.25	{11}	1	0.25
{5}, {6}	1	0.25	{11}, {5}	1	0.25
{6}	1	0.25			

Table 6.2: Result of mining all frequent patterns on the sequence database in Table 6.1 with a minsup of 0.25

Now, let us consider another encoding. Once again each plan is considered as a sequence, but from now, each word will become an item in this sequence. An item identifier is assigned to each different word. By applying this encoding, we obtain the sequence database in Table 6.3. We can easily discard this encoding because the sequential patterns obtained by mining this sequence database are mostly inconsistent. Indeed, this encoding does not take into account any delimitation between operators.

Sequence_ID	Sequence
S001	{1}, {2}, {3}, {4}, {2}, {5}, {6}, {2}, {7}
S002	{4}, {8}, {9}, {6}, {8}, {10}, {11}, {10}, {2}
S003	{4}, {2}, {12}, {13}, {14}, {5}, {6}, {2}, {15}
S004	{16}, {3}, {17}, {4}, {8}, {5}, {6}, {8}, {10}

Table 6.3: Sequence database using the encoding *an item by word*

Finally, one could imagine a translation (prior to the encoding) of the set of plans where

for each plan, each action is generalised to its respective operator and the relations between the actions are guaranteed. We obtain the following translated set of plans:

$$\pi_1 = op_3(?X0,?X1);op_1(?X0,?X2);op_2(?X0,?X3)$$
$$\pi_2 = op_1(?X0,?X1);op_2(?X0,?X2);op_4(?X2,?X3)$$
$$\pi_3 = op_1(?X0,?X1);op_5(?X2,?X3);op_2(?X0,?X4)$$
$$\pi_4 = op_6(?X0,?X1);op_1(?X2,?X3);op_2(?X2,?X4)$$

By using again the first encoding, we consider each plan as a sequence and each operator as an item. We obtain the sequence database in Table 6.4. We observe the same situation that has been discussed before, namely, a variability in the parameters leads to a loss in the observed support.

Sequence_ID	Sequence
S001	{1},{2},{3}
S002	{4},{5},{6}
S003	{4},{7},{8}
S004	{9},{10},{11}

Table 6.4: Sequence database using the encoding from Chapter 4 on a previously *translated* set of plans.

Mining macro-operators from a set of plans requires an approach which ensures to find the frequent sequences of operators without a loss of information about their characteristics. Then, neither an operator can be dissociated from its objects nor a sequence of operators can disregard the relationship between operators' objects. In other words, we are looking for the most frequent sequences of operators with a specific set of object relationships. Thus, it should be noted that for a same sequence of operators different set of object relationships lead to the construction of different macro-operators (see Figure 6.1). Unfortunately, a central restriction in traditional pattern mining concerning its expressiveness is that each item is assumed to be a whole entity without any additional characteristics.

Therefore, we are looking for mining macro-operators using an approach with an expressiveness that allows us to properly describe and process the mentioned planning requirements. This would allow us to learn as much as we can from past experiences.

Before concluding, let us now turn to another important issue of this work. If we have two macro-operators with two equal supports but one macro-operator has twice as many appearances as the other, then at equivalent impacts on the branching factor and at equal lengths, the first one will be more useful. We therefore also need a reliable measure to reflect these properties.

This measure should also be able to predict whether a macro-operator does not solve problems more quickly. It should also represent a return on investment linked to its

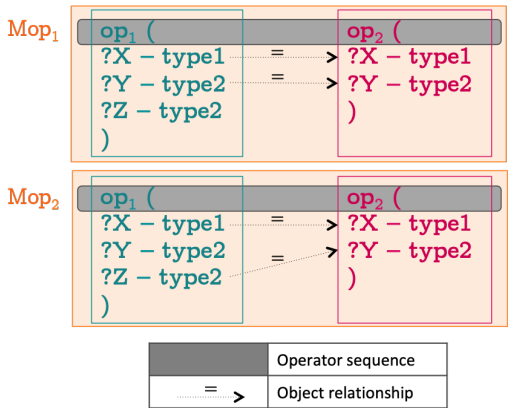


Figure 6.1: Macro-operators from different set of object relationships

length but also linked to the branching factor. Finally, we should be able to aggregate these measures to have a utility measure not only for a macro-operator but for a set of macro-operators.

In the next section, we will present our framework to mine macro-operators from a set of plans and the selection process in order to choose the optimal macro-operator set.

6.3 Description of the METEOR framework

This section presents METEOR, our framework for mining and selecting macro-operators from previous acquired knowledge. METEOR stands for **M**acro-operator **E**xtraction, **T**rade-off **E**stimation and **O**ptimisation from plan **R**ecycling.

Indeed, this framework includes two major steps (see Figure 6.2):

1. The extraction of macro-operators from plan recycling by using a pattern mining inspired algorithm which extracts rich patterns with attribute structures.
2. The estimation of the trade-off between the branching factor increase and the search depth reduction in order to choose the optimal macro-operator set.

Additionally, the METEOR framework can be used on past plans from the same domain regardless of the domain characteristics or the planner used to obtain them. The details of each step will be presented in the following sections.

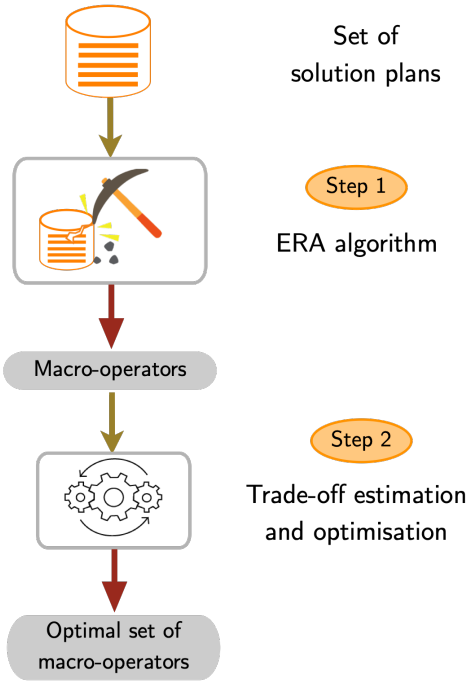


Figure 6.2: METEOR framework.

6.4 *ERA* Algorithm

In the following, we will present ERA, our pattern mining inspired algorithm for mining macro-operators and its characteristics from past plans. ERA stands for **E**xtraction of **R**ich patterns with **A**ttribute structures.

6.4.1 Encoding formalism

For processing purposes, we use the definition 6.1 to represent instantiated operators for the plans in Table 6.5. As a result, we obtain a dictionary as shown in Table 6.6. Finally, we use this dictionary and the original plans to obtain the sequence database in Table 6.7 where each sequence corresponds to a plan solution and each number to an instantiated operator.

Definition 6.1.
A dictionary of instantiated operators is a set of pairs $< k, a_k >$, where a_k is the k^{th} distinct encountered action (aka instantiated operator) in a corpus of plans and k is an integer referencing to a_k.

ID	Plan
pb1	{pick-up b}, {stack b a}, {pick-up c}, {stack c b}, {pick-up d}, {stack d c}
pb2	{unstack b c}, {put-down b}, {unstack c a},{put-down c}, {unstack a d}, {stack a b},{pick-up c}, {stack c a}, {pick-up d}, {stack d c}
pb3	{unstack c b}, {stack c d}, {pick-up b},{stack b c}, {pick-up a}, {stack a b}
pb4	{unstack b a}, {stack b c}, {unstack a d},{stack a e}, {unstack b c}, {stack b a},{pick-up c}, {stack c b}, {pick-up d}, {stack d c}

Table 6.5: Example of plans for `blocksworld` domain.

k	e_i
1	pick-up b
2	stack b a
3	pick-up c
4	stack c b
5	pick-up d
6	stack d c
7	unstack b c
8	put-down b
9	unstack c a
10	put-down c
11	unstack a d
12	stack a b
13	stack c a
14	unstack c b
15	stack c d
16	stack b c
17	pick-up a
18	unstack b a
19	stack a e

Table 6.6: Dictionary of instantiated operators for Table 6.5.

π_{id}	π_i
π_1	1,2,3,4,5,6
π_2	7,8,9,10,11,12,3,13,5,6
π_3	14,15,1,16,17,12
π_4	18,16,11,19,7,2,3,4,5,6

Table 6.7: Sequence database from dictionary in Table 6.6 and plans in Table 6.5.

Then, we use the definition 6.2 to represent the elements for the plans in Table 6.5. As a result, we obtain a dictionary as shown in Table 6.8. Finally, we use this dictionary and the original plans to obtain the sequence database in Table 6.9 where each sequence corresponds to an instantiated operator and each number to an element of this operator, i.e. operator name or parameter.

Definition 6.2.
A dictionary of elements is a set of pairs $< k, e_k >$, where e_k is the k^{th} distinct encoun-

tered element in a set of solution plans and k is an integer referencing to e_k.

k	e_i
1	pick-up
2	b
3	stack
4	a
5	c
6	d
7	unstack
8	put-down
9	e

Table 6.8: Dictionary of elements for Table 6.6.

o_{id}	o_i
o_1	1 2
o_2	3 2 4
o_3	1 5
o_4	3 5 2
o_5	1 6
o_6	3 6 5
o_7	7 2 5
o_8	8 2
o_9	7 5 4
o_{10}	8 5
o_{11}	7 4 6
o_{12}	3 4 2
o_{13}	3 5 4
o_{14}	7 5 2
o_{15}	3 5 6
o_{16}	3 2 5
o_{17}	1 4
o_{18}	7 2 4
o_{19}	3 4 9

Table 6.9: Sequence database from dictionary in Table 6.8 and plans in Table 6.5.

Thus, we use this encoding formalism to obtain from a corpus of past plans for a given domain: a sequence database of instantiated operators and a sequence database of action elements.

6.4.2 Description of the main algorithm

ERA pseudo code is described in Algorithm 9. The objective of this algorithm is to extract all macro-operators (regardless of their length or up to a maximal length *maxLength*) satisfying a frequency threshold *minsup* from a set of solution plans. ERA can detect the apparition of a macro-operator even if the actions composing it are not adjacent. For each macro-operator, this algorithm also yields the following characteristics:

- support [`integer`]: the number of plans containing at least one apparition.
- sequences ids [`list`]: the plan identifiers where the macro-operator appears.
- number of apparitions [`list`]: the number of apparitions of the macro-operator

in each plan.

It takes as an input a sequence database C of instantiated operators, a sequence database of action elements A, a *minsup* threshold that extracted sequences should ensure, and the maximal length of sequences to extract.

First, it searches the set of macro-operators of length two in C by using the procedure *MINE* (line 7, described in Algorithm 10). Second, for each macro-operator of this set, if it does not satisfies the *minsup* threshold we remove it, otherwise we keep the macro-operator and its characteristics (line 12,13,14). Finally, if no macro-operator has been found for the current length, it stops. Otherwise, it increases the length by one (line 15) and it continues to loop until it reaches the maximal length (line 6). It gives as a result a set of frequent macro-operators of different lengths with its respective characteristics.

6.4.3 The mining procedure

The objective of the mining procedure is to obtain the set of macro-operators of length l and its characteristics from the set of solution plans. To do so, it analyses for each plan all sub-sequences of length l and determines if the sub-sequence is a valid apparition[3] of a macro-operator. If the algorithm finds a valid macro-operator apparition, it records this apparition and updates the characteristics related to this macro-operator. To speed up its computation, it uses previous information obtained when mining length $l-1$.

The pseudo code of this procedure, called MINE, is described by algorithm 10. It takes as input both sequence databases C and A from the main algorithm, the length l to be evaluated and a dictionary M of all found macro-operators (of different lengths less than l) and their support. The first loop purpose (line 3) is to do all combinations of ordered sub-patterns for each sequence in C (line 7) and for each sub-pattern, determine if it is a valid macro-operator and if it is valid in a number of sequences greater than the *minsup* parameter. To accomplish this, the following steps are performed:

- It moves on to the next sub-pattern,
  - if the sub-pattern of length $l-1$ does not satisfy the *minsup*. For that, the current sub-pattern length should be greater than two in order to be able to build its identifier[4] of length $l-1$. Then, it checks if this identifier is found in the general dictionary of pairs <macro-operator,support> (line 10).
  - if there are not enough plans left to ensure that the sub-pattern is valid in a number of sequences greater than the *minsup* (line 12).

[3] The actions of the macro-operator can be moved contiguously in the plan without an impact on the final state or without impeding its execution

[4] See description in the box *computeId*

- Otherwise,

  - it removes from the current plan δ the individual actions of the sub-pattern sp, it builds a macro-action from sp (line 14) and puts it, each time, at a different position in the plan (line 19). It tries δ (line 20) from the calculated initial state S_i for the original plan ρ (line 6). If the result state ε is a superset (line 21) of the calculated final state S_g from the original plan ρ (line 6), then it stops trying positions for this sub-pattern. If it founds at least one valid position for the built macro-action, it stores the modified plan with the macro-operator identifier μ as the key access (line 24) and the macro-operator identifier μ in the list of the macro-operators found in the plan (line 25). To analyse new apparitions of an already found macro-action, the algorithm uses the corresponding modified plan (line 16).

Procedure (*computeId*: Identifier construction)

The identifier construction procedure takes as input a sub-pattern of instantiated operators sp and a length l. Only the first l elements of sp are kept in this procedure. A string identifier is built as follows. First, each element e is translated by using A. Next, the first sub-element of each e is used together with a character representing the actions. After, we use another character and a incremental number for each other sub-element of e because they represent the parameters. Notice that, the incremental number is reset to zero with each new identifier construction and a same parameter will have the same incremental number.
Example Let us consider the length $l = 2$ and the sub-pattern {1,2,3} from the sequence database of Table 6.7, this sub-pattern represents the actions {pick-up b, stack b a, pick-up c} and we can observe it, in the first plan from Table 6.5. We only kept the elements {1,2} and by using the Table 6.9, we translate them into {1 2, 3 2 4}. We choose the character "a" to represent actions and we have then {a1 2, a3 2 4}. Finally, we choose the character "p" to represent its parameters and we obtain the identifier {a1p0a3p0p1}. ■

Once it has analysed all combinations of sub-patterns of length l from ρ, it moves to the second loop (line 26). The purpose here is to compute and save or update, the characteristics of each found macro-operator. Thus, it updates the set of plans where the macro-operator with identifier μ appeared App_o by adding the index of the current plan *indexPlan* (line 27). Also, it computes and stores the number of apparitions in the plan for the analysed macro-operator (line 28,29). Finally, if the current macro-operator appears in the plan at least once, the support value is updated by one (line 30) or added with value of 1 if it did not appear before (line 32).

It gives as a result a set of frequent macro-operators of length l with its respective characteristics. They will be filtered, by using the *minsup* parameter in the main algorithm, before to be added to the final set of mined macro-operators.

Algorithm 9 ERA algorithm - Main algorithm

Input A sequence database C of instantiated operators, a sequence database A of elements of an action, a *minsup* parameter and a maximal length *maxLength*.

Output A dictionary M of pairs $< m, s >$, m is a macro-operator and s is its support; a dictionary *App* of pairs <*m,app*> *app* is the id of the sequence where m appears; a dictionary *nbApp* of pairs <*m,nbApp*>, *nbApp* is the number of occurrences of the macro-operator m for each sequence.

```
1:  function MINING MACROS(C, A, minsup, maxLength)
2:      Mo, App_o, nbApp_o ← empty dictionaries
3:      M, App, nbApp ← empty dictionaries
4:      stop ← False, l ← 2
5:      while (l ≤ maxLength) ∧ (stop is False) do
6:          Mo, App_o, nbApp_o ← MINE(C,A,M,l)
7:          stop ← True
8:          for each macro-operator m in Mo do
9:              if support(m) ≥ minsup then
10:                 stop ← False
11:                 add the key m with value s to M
12:                 add the pair <m,app> to App from App_o
13:                 add the pair <m,nbApp> to nbApp from nbApp_o
14:         increase l by one
15:     return M,App,nbApp
```

Algorithm 10 Mining macro-operators of length l

Input The sequence database C, the sequence database A, a dictionary M of pairs $< m, s >$, m is a macro-operator and s is its support and a length l.

Output A dictionary Mo of pairs $< m, s >$, m is a macro-operator of length l and s is its support; a dictionary *App_o* of pairs <*m,app*>, *app* is the id of the sequence where m appears; and a dictionary *nbApp_o* of key <*m,iP*>, *iP* is the index of the sequence where m appears, and value *nbApp*, the number of occurrences of the macro-operator m in each sequence.

1: **function** MINE(C, A, M, l)
2: *Mo, App_o, nbApp_o* $\leftarrow$ empty dictionaries
3: **for each** plan ρ in C **do**
4: $D, macroPlan \leftarrow$ empty dictionaries
5: $idsPlan \leftarrow \{\emptyset\}$
6: $S_i, S_g \leftarrow$ calculate initial and final state from ρ
7: $P \leftarrow$ all combinations of sub-patterns of length l from ρ ▷ †
8: **for each** ordered subpattern sp in P **do**
9: **if** $l > 2$ **then**
10: **if** $computeId(sp, l-1) \notin M$ **then** skip sp
11: **else** $\mu \leftarrow computeId(sp, l)$
12: **if** len(C)-$indexPlan < minsup - \text{supp}(\mu)$ **then** skip sp
13: **else**
14: add the key-value $< k, \{actions(sp)\} >$ to D ▷ $k \in \mathbf{Z}^*_-$
15: $i \leftarrow 0$
16: **if** $\mu \in macroPlan$ **then** $\delta \leftarrow macroPlan[\mu]$
17: **else** $\delta \leftarrow \rho$
18: **while** (not ok)$\wedge(i < \text{len}(\delta) - \text{len}(sp) + 1)$ **do**
19: $\delta \leftarrow$ remove sp from δ and insert k in δ in position i
20: $\varepsilon \leftarrow execute(S_i, \delta)$
21: **if** $S_g \subset \varepsilon$ **then** $ok \leftarrow True$
22: reset δ
23: **if** ok **then**
24: add the pair key-value $< \mu, \delta >$ to $macroPlan$
25: add μ to $idsPlan$

for each identifier μ in *idsPlans* **do**
  add the key μ with value *indexPlan* to *App_o*
  $nbA \leftarrow \frac{(\text{len}(\rho)-\text{len}(macroPlan[\mu]))}{(l-1)}$
  add the key $<\mu, indexPlan>$ with value *nbA* to *nbApp_o*
  if $(nbA > 0) \wedge (\mu$ in $Mo)$ **then** increase support of μ by one
  else
    if $nbA > 0$ **then** add the key μ with value *nbA* to *Mo*
return *Mo,App_o,nbApp_o*

† : the combinations keep the order of apparition in the original plan.

6.4.4 Complexity analysis

The main task of the ERA algorithm is the analysis of a sub-pattern. This task is repeated for each sub-pattern in a plan, for each plan in the set of solution plans and for each sub-pattern length. Let us first compute the number of sub-patterns n_{sp} of length k in a plan ρ of length $l(\rho)$.

n_{sp} is then, the number of ways to choose k elements in $l(\rho)$ elements:

$$n_{sp} = \binom{l(\rho)}{k} \tag{6.1}$$

Then, if n_{MINE} is the number of sub-patterns analysed in a execution of the MINE procedure at length k, we have:

$$n_{MINE} = \sum_{\rho_i \in C} \binom{l(\rho_i)}{k} \tag{6.2}$$

In the worst case, the ERA algorithm will mine up to the pattern length L where L is the maximal plan length in a set of solution plans C. Considering that all plan lengths are equal to the maximal plan length, we have N, the total number of sub-patterns analysed:

$$N = \sum_{\rho_i \in C} \sum_{k=2}^{l(\rho_i)} \binom{l(\rho_i)}{k} \tag{6.3}$$

$$N = \sum_{\rho_i \in C} 2^{l(\rho_i)} - l(\rho_i) - 1 \tag{6.4}$$

$$N = O\left(|C|2^L\right) \tag{6.5}$$

We show in Equation (6.5) that in the worst case, where the algorithm mines up to the maximal plan length L, the complexity is exponential in L and linear in the size of the solution plan set C.

In practice, we observed that with a high enough *minsup* parameter, the maximal sub-pattern length is much lower than the maximal plan length. For example, in our case we never mined macro-operators longer than eight (for a maximal plan length about 70). Let D be the maximal sub-pattern length (either forced by the *maxLength* parameter or induced by the *minsup* parameter). The total number of sub-patterns analysed N then becomes:

$$N = \sum_{\rho_i \in C} \sum_{k=2}^{D} \binom{l(\rho_i)}{k} \tag{6.6}$$

$$N = O\left(|C| \sum_{k=1}^{D} O\left(L^k\right) \right) \tag{6.7}$$

$$N = O\left(|C| L^D\right) \tag{6.8}$$

We show in Equation (6.8) that usually we can expect a polynomial complexity in L and linear in the size of the solution plan set C.

6.5 Selection of the optimal macro-operator set

In the following, we will present the second major step of the METEOR framework, namely the estimation of the trade-off between the branching factor increase and the search depth reduction in order to choose the optimal set of macro-operators.

Despite its intuitive correlation with the utility, the support fell short as a reliable *a priori* descriptor for macro utility. A good and reliable utility descriptor for planning performance should be able to answer the following question :

Given a macro set M, is the addition of M to the domain going to improve the performance of my planner ?

Under the assumption that most of the planning time is spent computing heuristic values, a good indicator of planning performance is the number of node opened during the search process. The gain of a macro m on a plan p (solution of a problem P), $G(m;p)$[5] can then be defined as :

$$G(m;p) = \frac{N(p)}{N(m;p)} \tag{6.9}$$

where $N(p)$ is the total number of open nodes when solving P with no macros and $N(m;p)$ the total number of opened nodes when solving P with m added to the domain.

[5] All notations refer to p since P is unknown.

Provided the number of opened nodes when solving the training plans ($N(p)$, hereafter noted N), we propose a new *a priori* utility estimation for macro-operators :

$$\overline{G}(M;C) = \frac{1}{|C|} \sum_{p \in C} \frac{N(p)}{\hat{N}(M;p)} \tag{6.10}$$

where M is the set of macro-operators to be evaluated, C is the set of solution plans and $\hat{N}(M;p)$ is the estimated number of opened nodes when solving P with a domain augmented with all macro-operators in M.

To see how we can derive that value, let us focus on computing the gain of a single macro m in a plan p (presented in Equation (6.9)).

First, we need to estimate the mean branching factor b of the problem, hereafter called the initial branching factor, where we consider that :

- there is no heuristic (every possible instance is opened)
- there is no macro added to the domain
- the objects present in the plan are the only objects present in the problem.

To do that, we recover the initial state of the plan (minimal set of predicates so that the plan can be executed) and compute each state of the plan from that initial state. We then count the total number of applicable operator instances $n_i(s)$ on each state s. The initial branching factor was estimated as :

$$b = \frac{1}{n_s} \sum_{s \in plan} n_i(s) \tag{6.11}$$

with n_s the number of states.

We then estimate the impact of the addition of a macro-operator m on the branching factor. Let b_m be the branching resulting from the addition of m and Δb be the additional branching brought by m so that :

$$b_m = b + \Delta b \tag{6.12}$$

Then, if $n_i(s;m)$ denotes the number of applicable instances of the macro m to a state s, Δb can be computed as :

$$\Delta b = \frac{1}{n_s} \sum_{s \in plan} n_i(s;m) \tag{6.13}$$

Let b_h be the observed branching factor when the problem P was solved to find p, so that :

$$b_h{}^l = N \tag{6.14}$$

where l is the plan length. Then b_h represents the mean branching factor with no macros when we consider the heuristic. From this we can derive the heuristic factor, i.e., the performance of the heuristic in the current plan. We chose to model it as a multiplicative factor to the initial branching factor :

$$b_h = f_h b \tag{6.15}$$

and thus :

$$f_h = \frac{N^{\frac{1}{l}}}{b} \tag{6.16}$$

By considering the heuristic efficacy constant when macros are added to the domain (i.e. macros are not considered in the heuristic computation), we have :

$$\hat{N}(m;p) = \left(f_h\,(b+\Delta b)\right)^{l-\Delta l} \tag{6.17}$$

With Δl the number of actions saved in p by adding m :

$$\Delta l = (|m| - 1) n_{app}(m; p) \tag{6.18}$$

And $n_{app}(m; p)$ the number of valid apparitions of m in p.

This estimation can be extended to sets of non overlapping macro-operators, i.e. a set where each operator never appears more than once (see Equation 6.19).

$$M \text{ is non overlapping} \iff \forall m \in M, o \in m \implies \forall n \in M, n \neq m, o \notin n \tag{6.19}$$

Example Let us consider macro-operators m_1 =`pick-up_stack ?X0 ?X1` and m_2 = `pick-up_put-down ?X0` from `blocksworld` domain. These macro-operators are overlapping since both have the `pick-up` operator. If we consider a third macro-operator m_3 =`unstack_put-down ?X0 ?X1`, we can observe that macro-operators m_1 and m_3 are not overlapping since they have no common operator. ■

For a non overlapping set of macro-operators $M = \{m_1, m_2, \ldots\}$, if $\Delta b(m_k; p)$ and $\Delta l(m_k; p)$ denote the respective values of Δb and Δl for the macro m_k in a plan p then :

$$\hat{N}(M; p) = \left(f_h(p) \left(b(p) + \sum_{m \in M} \Delta b(m; p) \right) \right)^{l(p) - \sum_{m \in M} \Delta l(m; p)} \tag{6.20}$$

Finally, the optimal macro set verifies, for the training corpus C :

$$M_{opt} = \underset{M}{\operatorname{argmax}} \left(\overline{G}(M; C) \right) \tag{6.21}$$

The pseudo code of the selection of the optimal macro-operator set procedure, called *computeOptimalMacroSet* is described in algorithm 11. It takes as input a set of macro-operators M and it tries to find the set of macro-operators that maximises the gain.

To accomplish that, it takes each macro-operator in M as a initial optimal set (line 4) and it tries to find the best gain by adding other macro-operators from M to this temporal optimal set (line 11). For that, if it founds a macro that increases the current gain, it adds this macro to the temporal set and continues to loop (always from the initial macro

in the first loop) until no more macros can be added (line 12-21). For each found set if its gain is greater than the maximal gain, then it becomes the new optimal set and the maximal gain is updated with its gain (line 22-24). Finally, it continues to loop until all macro-operators have been tried as a initial optimal set.

It gives as a result an optimal set of macro-operators according to the Equation 6.21.

Algorithm 11 Selection of the optimal macro-operator set

Input A set of macro-operators M.

Output An optimal set of macro-operators oS according to the Equation 6.21.

```
 1: function COMPUTEOPTIMALMACROSET(M)
 2:     maxGain ← 0
 3:     sortedMacros ← sort M by decreasing order of macro gain
 4:     for each macro in sortedMacros do
 5:         currentOptimalSet ← {∅}
 6:         add macro to currentOptimalSet
 7:         currentGain ← gain(macro)
 8:         done ← False
 9:         while not done do
10:             macroAdded ← False
11:             for each macro m in sortedMacros do
12:                 if m ∉ currentOptimalSet then
13:                     if m non-overlapping currentOptimalSet then
14:                         newGain ← gain(currentOptimalSet)          ▷ †
15:                         if newGain > currentGain then
16:                             macroToAdd ← m
17:                             macroAdded ← True
18:                             currentGain ← newGain
19:                 if macroAdded then
20:                     add macroToAdd to currentOptimalSet
21:                 else done ← True
22:             if currentGain > maxGain then
23:                 update oS with currentOptimalSet
24:                 maxGain ← currentGain
25:     return oS
```

† : Gain computed according to Equations 6.10 and 6.20.

This algorithm performs in polynomial time. In the worst case, it would perform in $O(|M|^3)$. As we will see in Section 6.7, $|M|^3$ is very low against the complexity of the ERA algorithm and it is negligible when considering the global execution time.

6.6 Evaluation of the METEOR framework

In the following, an evaluation of the METEOR framework is proposed. We present the methodology to conduct the evaluation and our evaluation criteria. In the next section, we show and discuss some interesting results obtained by doing this evaluation.

6.6.1 Methodology

The evaluationwas based on the same four benchmarks of the evaluation of Chapter 4: `barman`, `blocksworld`, `depots` and `satellite`. They are described in more detail in Appendix A. These benchmarks problems were taken from past International Planning Competitions[6].

For each benchmark, a training set of problems of 1000 problems and a test set of 30 problems were generated. The problem generation stage uses the generators[7] from the International Planning Competition. In addition, it ensures that the generated problems can be different even using the same parameters. In Table 6.10, we show the parameters used for the generation of problems for each benchmark domain.

We used a heuristic search planner based on A* search strategy, from the PDDL4J [8] library, to obtain a set of solution plans from the training set of problems. We followed, for each benchmark, the *METEOR framework* described before with a *minsup* of 0.85 and an infinite maximum length for the ERA algorithm. Then, we used each obtained optimal set of macro-operators to enhance each original domain. Finally, we evaluated METEOR as a good framework to learn macro-operators from past plans and decide on the utility of them. For that, we solved the set of test problems using the original domain, and then, using the enhanced domain. We compared the obtained results by using our evaluation criteria which will be described in the next section.

Experimental setup
The macro-operator learning steps (see Figure 6.2) were done on a notebook with an Intel Core i7-4710MQ quad-core CPU clocked at 2.5GHz and with 8GB of RAM, running MS Windows v8.1. The evaluation of the framework was conducted on a notebook

[6] http://icaps-conference.org/index.php/Main/Competitions
[7] https://bitbucket.org/planning-tools/pddl-generators
[8] https://github.com/pellierd/pddl4j

Domain	Parameters	Range
Barman	cocktails	1-30
	ingredients	1-13
	shots	1-30
Blocksworld	blocks	5-30
Depots	depots	1-5
	distributors	1-3
	trucks	1-4
	pallets	1-8
	hoists	1-8
	crates	1-20
Satellite	satellites	1-6
	instruments	1-2
	modes	1-8
	targets	1-2
	observation	1-20

Table 6.10: Parameters for the generation of problems

with an Intel Core i7-4980HQ quad-core CPU clocked at 2.8GHz and with 16GB of RAM, running OS X Mojave v10.14.1.

In the evaluation, to solve each problem from the test set, a maximum of 8GB of memory was allocated and a time limit of 600 seconds was set in the planner. The experiments have been done in a non-graphical terminal session.

6.6.2 Evaluation criteria

Unlike other works, we did not base our evaluation only on the classical IPC score. IPC score[9] is intended, as the name implies, to give a score to rank different strategies. In other words, by using IPC score we can decide which strategy is better than another, but it does not quantify the gain.

Here, on top of the IPC ranking, we wanted to quantify the impact of enhancing planning domains by adding the found optimal set of macro-operators obtained with the METEOR framework. In this perspective, we used for this evaluation the same different criteria established in Chapter 4: the planning time metric G_T, the space size metric

[9] As defined in the Learning track of the 7th International Planning Competition (Jiménez Celorrio et al., 2011)

G_N and the plan quality metric G_Q, according to Equations 4.1, 4.2 and 4.3. In addition, in order to prevent inaccuracies in the actual evaluation, we only kept the problems solved in a time greater than 0.1 seconds.

6.7 Results

In this section, we present the results of the evaluation following the steps of the METEOR framework.

For each domain, the number of mined macro-operators, the length of the longest macro-operator found and its optimal set of macro-operators are presented in Table 6.11. Among the four domains, barman domain present the most significant number of mined macro-operators but these results, clearly, suggest that the number of mined macro-operators is related to the characteristics of the domain. Also, notice that each optimal set is composed of two macro-operators, however, there was no restriction about the number of macro-operators in the optimal set.

Domain	# macro-operators	l	Optimal set†
Barman	52	8	macro-6-actions-0-10-11-8-9-0 macro-5-actions-6-5-2-7-3
Blocksworld	6	3	macro-2-actions-3-1 macro-2-actions-0-2
Depots	10	4	macro-4-actions-1-3-1-3 macro-2-actions-4-2
Satellite	13	5	macro-2-actions-1-3 macro-2-actions-0-4

Table 6.11: Number of mined macro-operators, the length of the longest macro-operator found and the selected optimal set for each domain.
† The translation of the operators and the full report given by the approach can be found in Appendix B.

After the extraction and optimisation step, the optimal set of macro-operators was added to the original domain. Now, we present the results obtained by solving the problem test set with the original domain and with the enhanced domain.

The time performance for each domain and for each problem is presented in Figure 6.3 and Figure 6.4. Here, problems are ordered in the x-axis with respect to their difficulty, i.e. the time required to solve it with the original domain, and the search time is showed

in seconds in the y-axis using a log10 scale. If the problem was not solved either using the original domain or the enhanced domain, it was ignored. The enhanced domain was found to solve more problems and faster than the original domain for all evaluated domains.

In order to quantify the behaviour observed in previous graphs, Figure 6.5 and Figure 6.6 display the planning time impact of the enhanced domains compared to the original domain. This impact is showed in terms of the log10 value of G_T in the y-axis for each problem in the x-axis. Remember that, for problems coloured in red, the gain is underestimated since the original domain did not solve them. Also, we only analysed problems solved in a time greater than 0.1s. Thus, we observed that the gain ranges are very significant. The domains present a similar behaviour, and even when the `barman` domain presents the lowest gains, they are still important.

Table 6.12 summarises the time performance results for each domain by presenting the well-known IPC score but also our planning time metric G_T. As stated before, the former allow us to decide which strategy is better than another, but the later quantifies the average time gain.

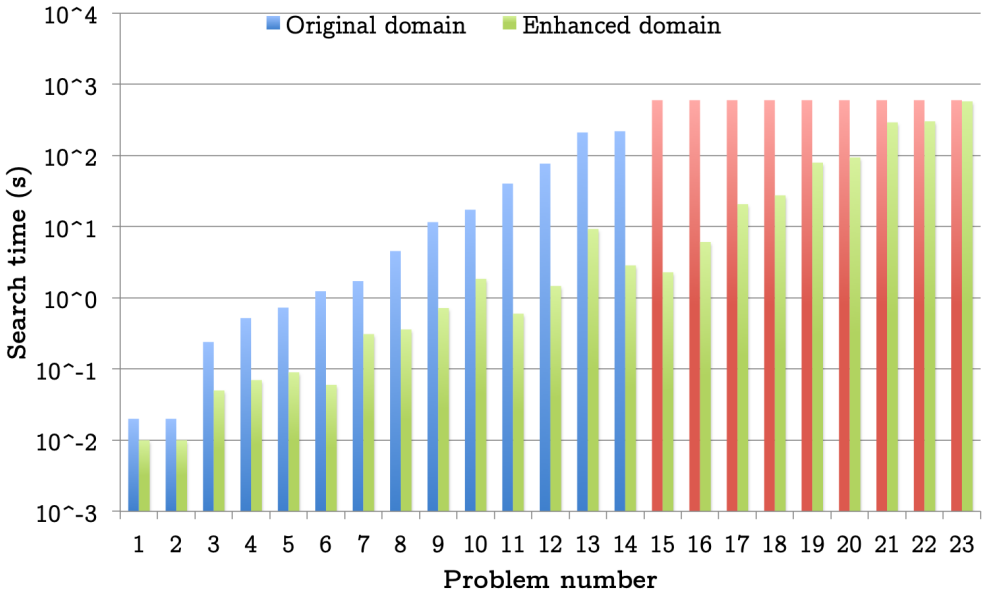
Domain	**IPC Score**		G_T
	Original domain	Enhanced domain	
Barman	7.6	23	34.95
Blocksworld	11.3	22	182.91
Depots	2.2	26	163.9
Satellite	15	30	201.16

Table 6.12: Results represented as IPC Score and average time gain for each domain.

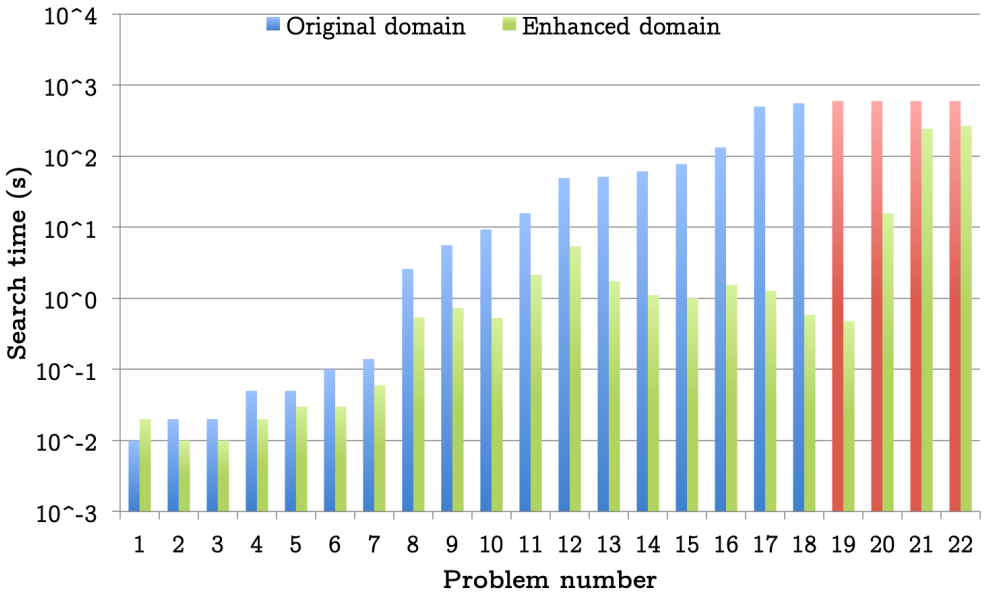
The impact in the size of the final search space is given in Figure 6.7 and Figure 6.8. Indeed, they present the number of nodes explored in the y-axis for each problem in the x-axis. This number is presented in terms of the log10 value of N. The final space size was best impacted when using the enhanced domain. As the use of macro-operators is supposed to provide a way to go deep quickly into the search space, this behaviour was expected.

The length of the found plans, for each domain, when using the original domain, in comparison to the length of the found plans when using the enhanced domain are shown in Figure 6.9 and Figure 6.10 show. For each problem in the x-axis, the plan length of the found plan is plotted in the y-axis. We observe a slightly difference between length of plans, often it is within 5.

Finally, Table 6.13 summarises these last observations by showing the average impact in the final space size G_N and the average impact in the length of the plans G_Q for each domain. Thus, we observed that the enhanced domain provides a positive impact in the

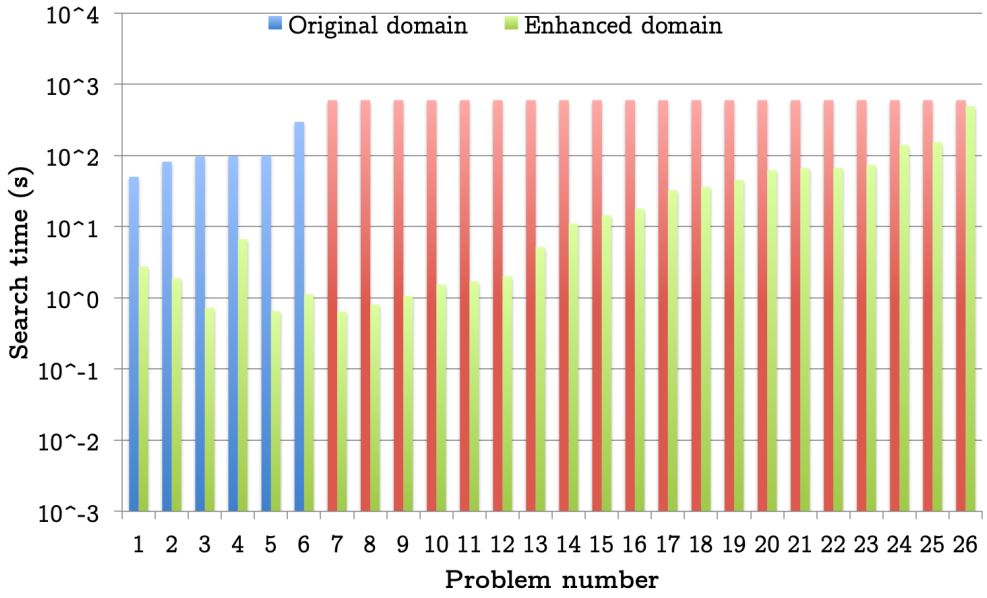


(a) Barman

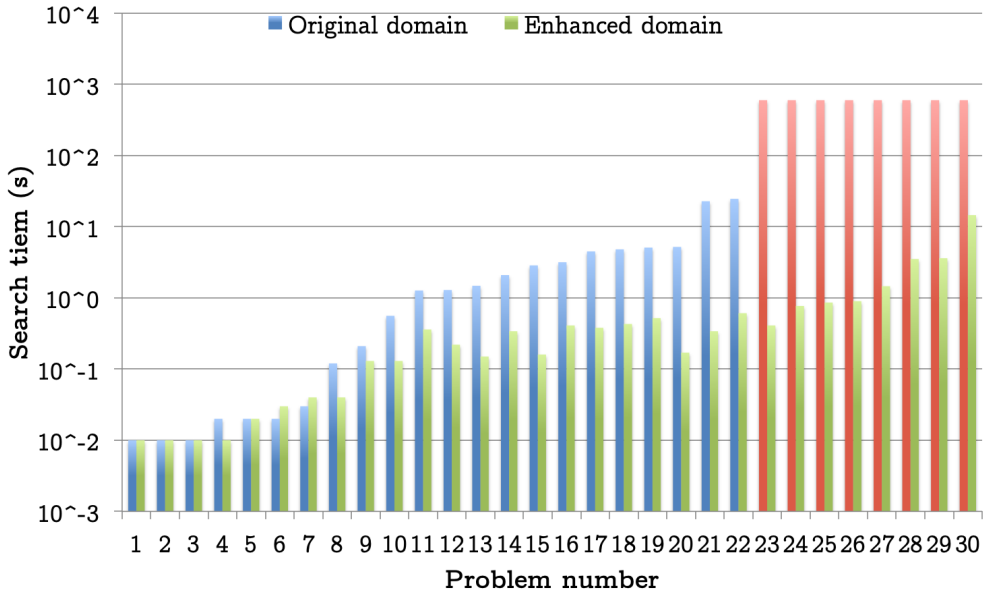


(b) Blocksworld

Figure 6.3: Search time performance for `barman` and `blocksworld` domains. In blue (resp. in green), the time performance for problems solved with the original domain (resp. enhanced domain). In red, the problems not solved with the original domain but solved with the enhanced domain.

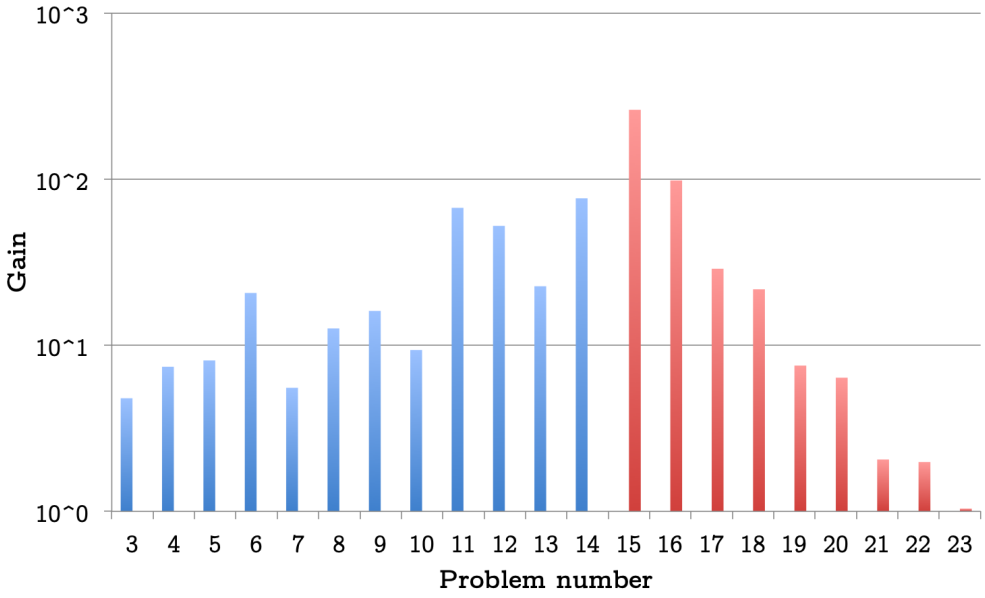


(a) Depots

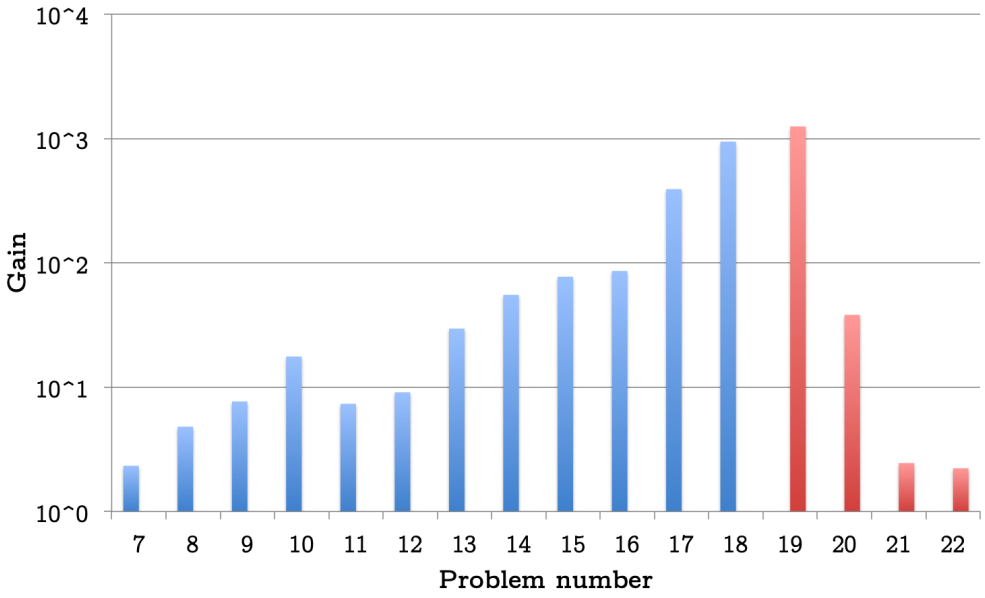


(b) Satellite

Figure 6.4: Search time performance for `depots` and `satellite` domains. In blue (resp. in green), the time performance for problems solved with the original domain (resp. enhanced domain). In red, the problems not solved with the original domain but solved with the enhanced domain.

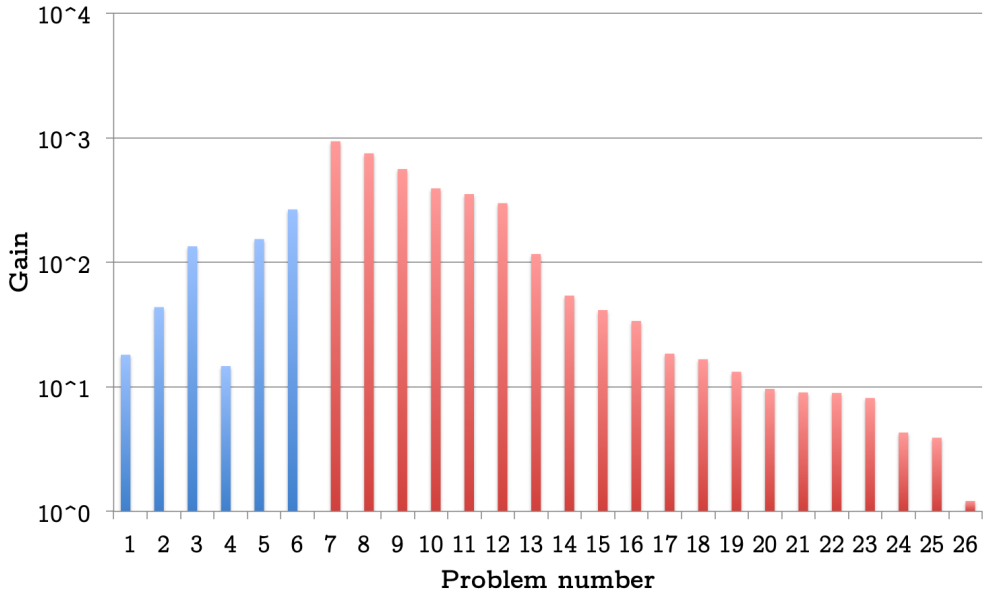


(a) Barman

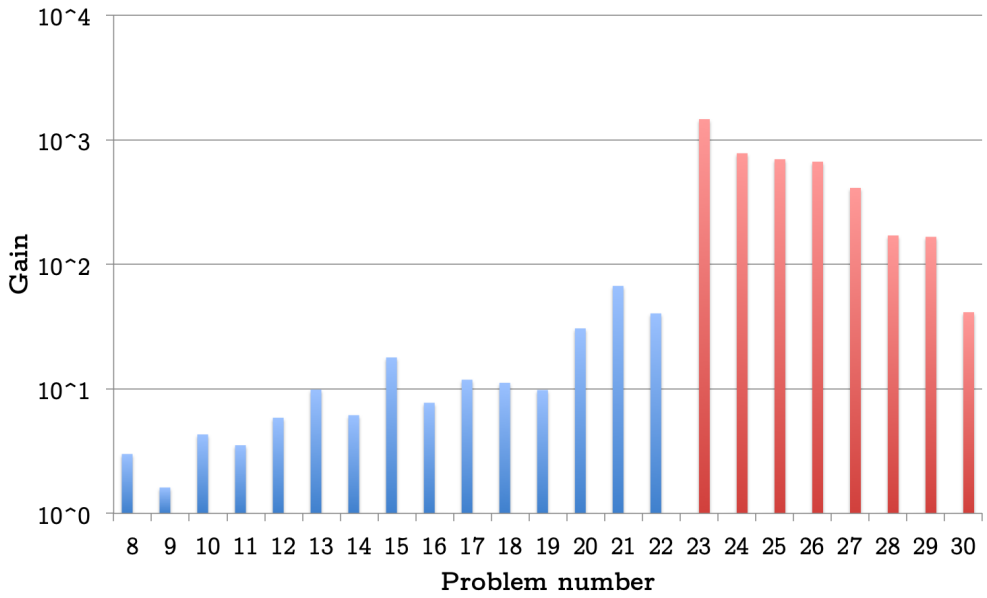


(b) Blocksworld

Figure 6.5: Time gain for `barman` and `blocksworld` domains. In blue, the gain for problems solved with the original domain and with the enhanced domain. In red, the gain for problems that were not solved with the original domain but solved with the enhanced domain. Thus, the gain is underestimated.



(a) Depots



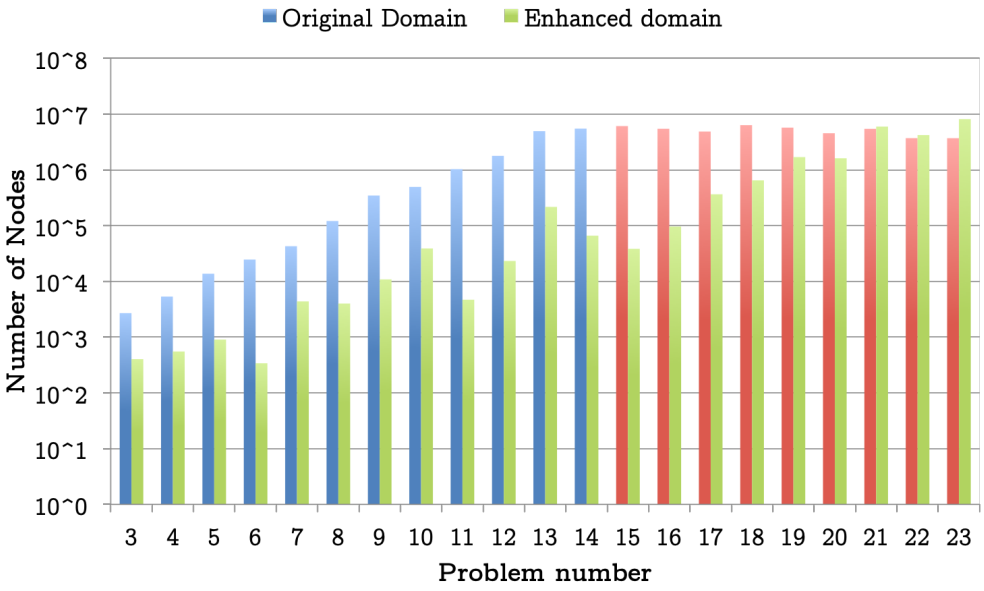
(b) Satellite

Figure 6.6: Time gain for `depots` and `satellite` domains. In blue, the gain for problems solved with the original domain and with the enhanced domain. In red, the gain for problems that were not solved with the original domain but solved with the enhanced domain. Thus, the gain is underestimated.

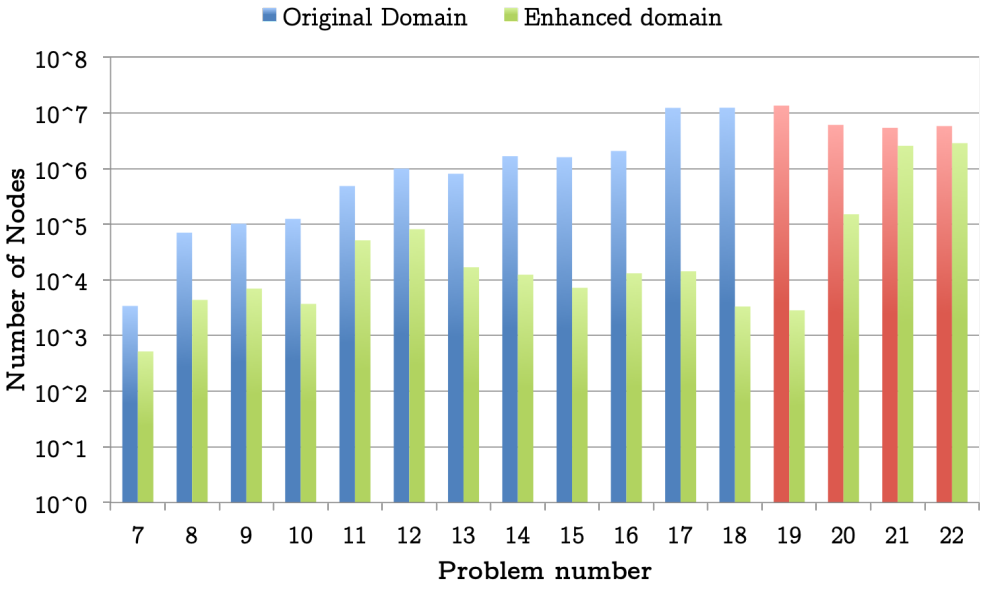
final search space without drastically decreasing the quality of the plans.

Domain	G_N	G_Q
Barman	49.30	0.89
Blocksworld	437.05	0.94
Depots	227.88	0.89
Satellite	54.62	0.98

Table 6.13: Results represented as the average impact in the final space size G_N and the average impact in the length of the plans G_Q for each domain.

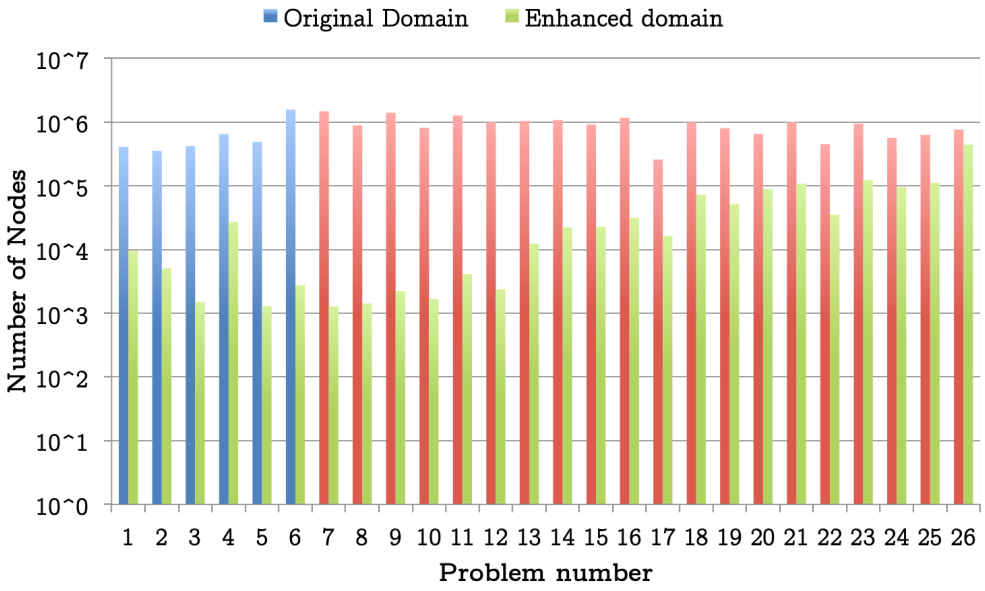


(a) Barman

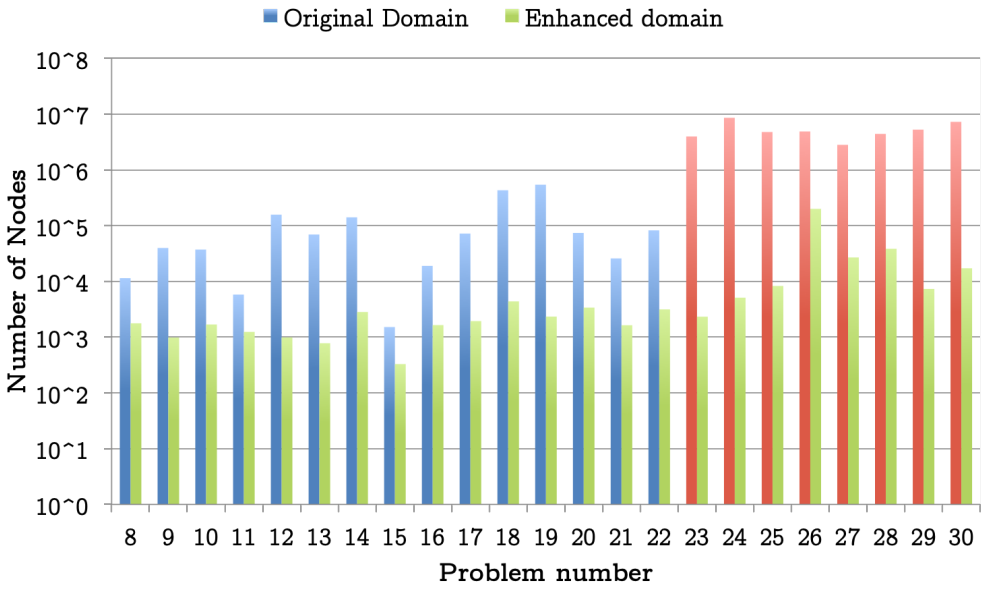


(b) Blocksworld

Figure 6.7: Number of explored nodes for `barman` and `blocksworld` domains. In blue (resp. in green), the nodes for problems solved with the original domain (resp. the enhanced domain). In red, the nodes for problems that were not solved with the original domain.

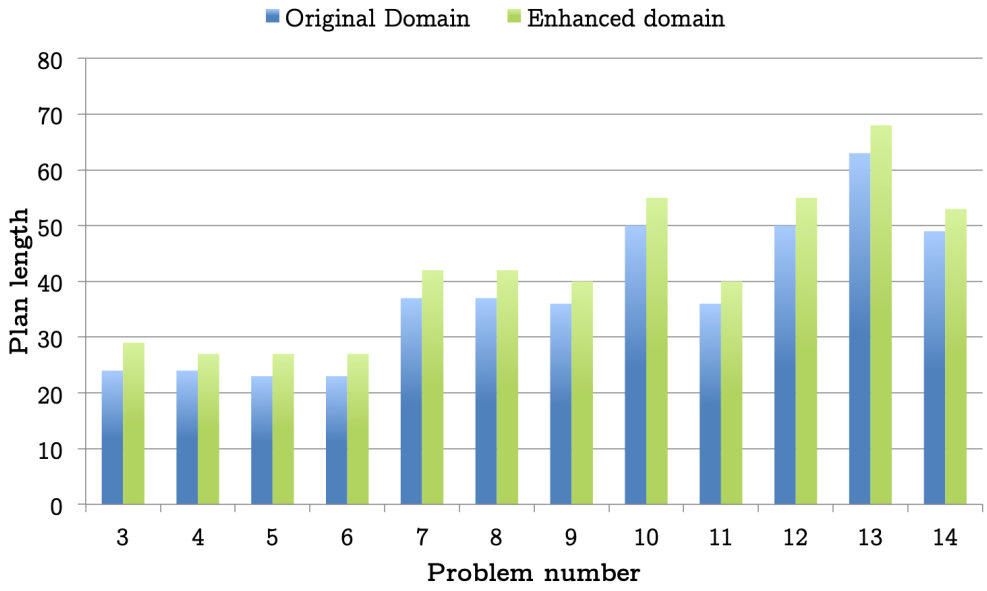


(a) Depots

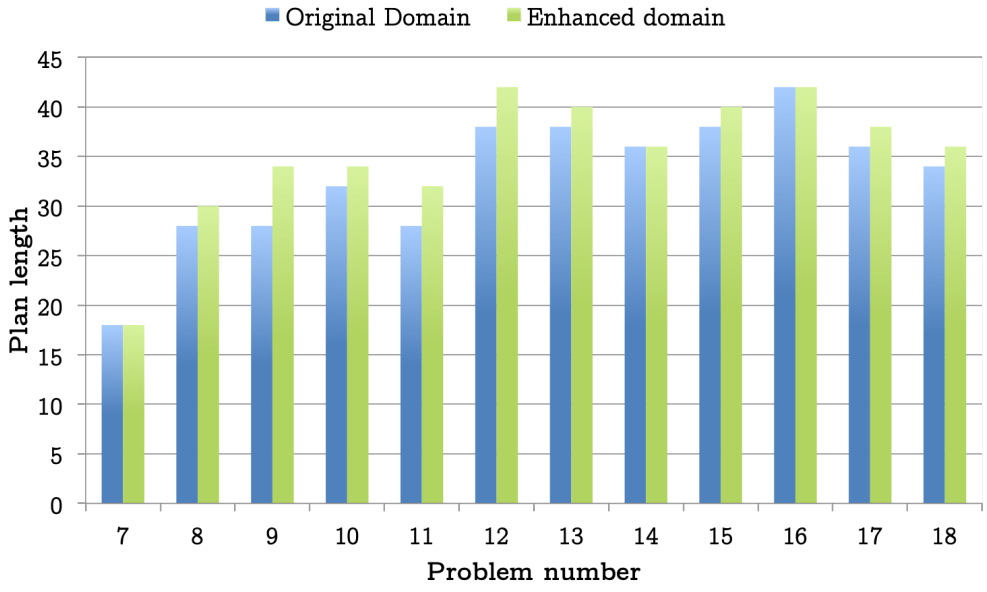


(b) Satellite

Figure 6.8: Number of explored nodes for `depots` and `satellite` domains. In blue (resp. in green), the nodes for problems solved with the original domain (resp. the enhanced domain). In red, the nodes for problems that were not solved with the original domain.

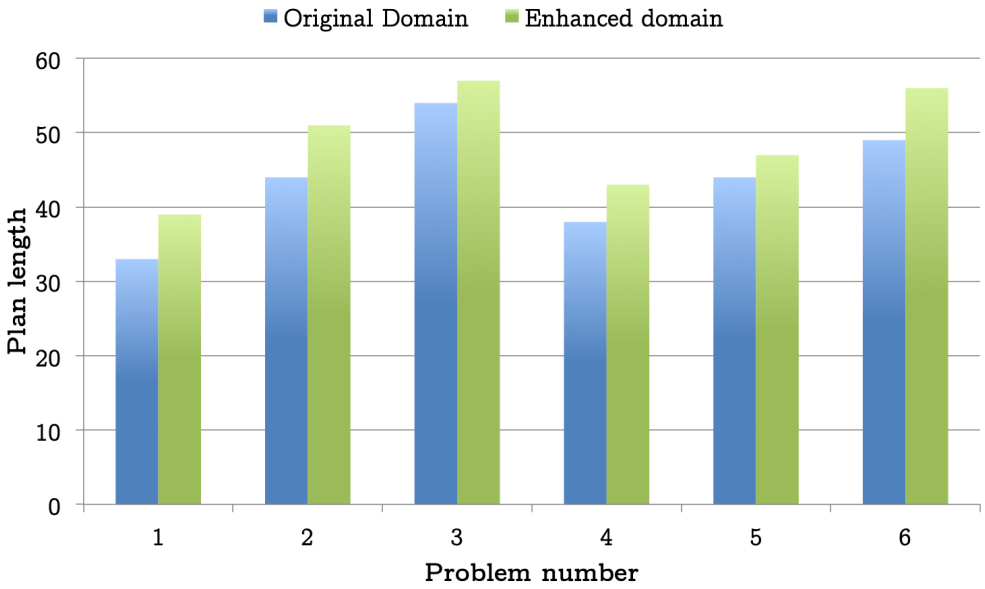


(a) Barman

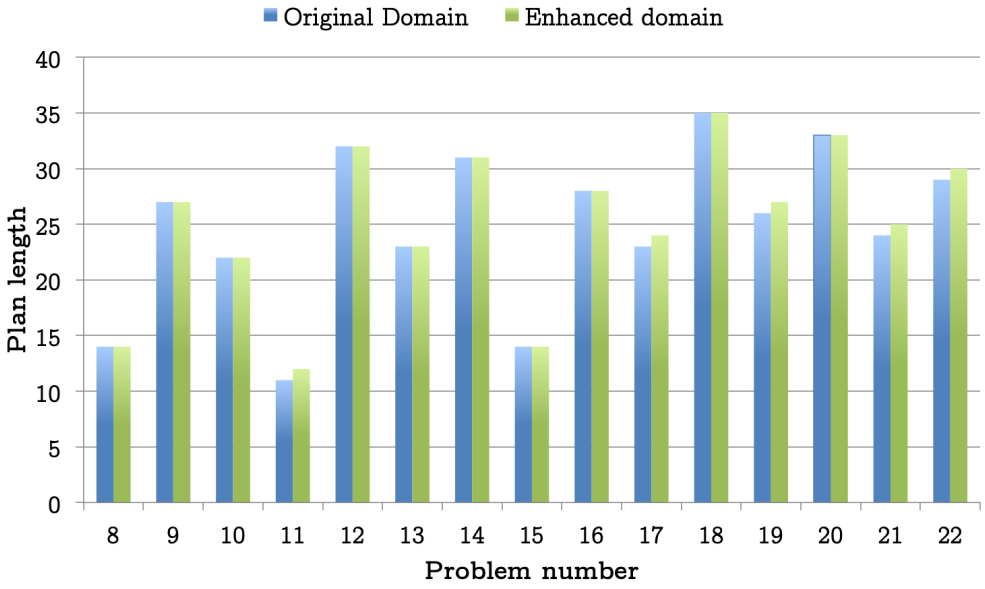


(b) Blocksworld

Figure 6.9: Plan length for each problem in `barman` and `blocksworld` domains. In blue (resp. in green), the plan length for problems solved with the original domain (resp. the enhanced domain).



(a) Depots



(b) Satellite

Figure 6.10: Plan length for each problem in `depots` and `satellite` domains. In blue (resp. in green), the plan length for problems solved with the original domain (resp. the enhanced domain).

6.8 Discussion

This chapter present and assess the METEOR framework. We assumed that the problem solving task would show better performances when using a domain enhanced with macro-operators learned through our framework from previous experiences. Indeed, our framework was conceived for mining macro-operators from previous acquired knowledge, and for selecting the optimal set of macro-operators that maximises the node gain. Our approach has the advantage that it can be used on past plans from a given domain regardless any domain characteristic or the planner used to obtain those plans.

The mining phase was done by using the presented ERA algorithm on past plans. The filter step in our algorithm was inspired from the Apriori (Agrawal and Srikant, 1994) pattern mining algorithm, i.e. for mining a pattern of length l, we consider if its sub-patterns of length $l-1$ satisfying the *minsup*. In addition, as an answer to the present work, our algorithm has the particular feature of being able to mine patterns with attribute structures, as is the case with macro-operators. Results in this phase were consistent (Table 6.11) since our algorithm successfully mined valid macro-operators of different lengths for different domains. It is possible that domains having longest task sequences (e.g. `barman` with twelve primitive operators) with a high support causes a big number of macro-operators because all its sub-sequences will also have a high support.

The selection phase was essential to avoid unwelcome side-effects of the use of macro-operators, namely the overload caused by increasing the branching factor in the search space when adding macro-operators. Thus, for a set of macro-operators, we have chosen to optimise this trade-off between the branching factor increase and the search depth reduction. It resulted in an optimal set of macro-operators for each domain (Table 6.11). Although there was no limitation about the number of macro-operators in the optimal set, each one was composed of two macro-operators. These findings led us to believe that the gain is often maximised with two macro-operators. These results are in agreement with those of Botea et al. (2005b). They allow only the two best macro-operators resulting from a training step and in their used domains (depots, satellite and rover), their experiments showed that using only one or two macros was helpful for reducing the search. Besides, resulting optimal sets show that our selection phase can successfully find as an optimal set, macro-operators of different lengths. However, the selected optimal set can only be composed of non-overlapping macro-operators.

We used the resulting optimal set of macro-operators to enhance the original domain and we evaluated our framework by analysing elements such as the improvement in the runtime, the number of explored nodes and the plan length. Problem solving was more efficient by using the enhanced domain compared to the original domain. The gain ranges were very significant and the domains presented a similar behaviour. Even when the `barman` domain presents the lowest gains, they are still important (34 times faster). It is possible that the non-overlapping restriction, when finding the optimal set, has an impact in this domain. Regarding the impact in the final size of the search space, results showed an impressive improvement in `blocksworld` domain when using the enhanced domain (Table 6.13). By looking more closely at its individual data, we found

that the enhanced domain has done exceptionally well for two complex problems. If we remove them from the computations, G_N is in the same order of magnitude as for the other domains. As the use of macro-operators is supposed to provide a way to go deep quickly into the search space, this behaviour was expected. Globally, plan length slightly increased when using the enhanced domain but the difference was not more than 5 actions.

In summary, our framework has proven to successfully mine macro-operators of different lengths for different domains and thanks to the selection phase, be able to deliver a positive impact in the search time without drastically decreasing the quality of the plans.

6.9 Conclusion

Our objectives in this chapter were twofold: (1) to provide a way out of the limitations of sequential pattern mining in planning; (2) to came up with a selection measure allowing us to avoid undesirable side-effects of the use of macro-operators. We then presented a novel approach called METEOR which ensures to find the frequent sequences of operators from a set of plans without a loss of information about their characteristics.

Our METEOR framework was conceived for mining macro-operators from previous acquired knowledge, and for selecting the optimal set of macro-operators that maximises the node gain. The mining phase was done by using the presented ERA algorithm on past plans. For the selection phase, we have chosen to optimise the trade-off between the branching factor increase and the search depth reduction. Our approach has the advantage that it can be used on past plans from a given domain regardless any domain characteristic or the planner used to obtain those plans.

Our framework has proven to successfully mine macro-operators of different lengths for different domains and thanks to the selection phase, be able to deliver a positive impact in the search time without drastically decreasing the quality of the plans.

7

Conclusion and Perspectives

A great accomplishment shouldn't be the end of the road, just the starting point for the next leap forward.

Harvey Mackay

This thesis contributes mainly to the field of automated planning, and it is more specifically related to learning macros for classical planning.

We studied the following research questions:

Research question #1

Is there a monotonic relationship between the frequency of apparition of a macro and its utility? i.e. can the frequency alone be used as an estimator for macro ranking by utility?

Research question #2

Can we learn routines from past experiences that are not only frequent but above all useful?

For approaches extracting macros from past experiences, an assumption often used is that frequent sequences of actions are potentially good candidates to enhance the domain. This assumption is then related to the first research question. We, therefore, chose the *sequential pattern mining* technique to exploit this hypothesis. This technique from the field of data mining aims to analyse sequential data to discover frequent sequential patterns.

To answer the first research question, we studied the possibility of extracting macro-actions (i.e. sequences of actions with known objects) via sequential pattern mining algorithms and selecting useful macro-actions based on their frequency (Chapter 4). As we found some discrepancies in the results of the precedent study, we wanted to explore them. We then transposed the study to macro-operators (i.e. sequences of actions with variable objects), and we proposed a new approach to validate the generated macros (Chapter 5). This approach proved to be successful in eliminating problematic macro-operators. However, we found out that the frequency alone may not provide a consistent selection of useful macro-actions.

We discussed the problems of using classic pattern mining algorithms in planning. Despite the efforts, we find ourselves in a dead-end with the selection process because the pattern mining filtering structures are not adapted to planning. Then, to answer the second research question, we proposed a novel approach which ensures to find the frequent sequences of operators from a set of plans without a loss of information about their characteristics (Chapter 6). It proved to successfully mine macro-operators of different lengths for four different domains and thanks to the selection phase, be able to deliver a positive impact on the search time (underestimated gain between 7 times and 200 times faster) with little influence on the quality of the plans (plan length difference was not more than 5 actions).

In the following, we first provide a reminder of our contributions. Then, we present the limitations of this work and finally, we introduce perspectives for future research directions.

7.1 Summary of contributions

7.1.1 Exploration of the link between macro-action frequency and macro-action utility

We proposed a framework to extract macro-actions (i.e. sequences of actions with known objects) via sequential pattern mining algorithms. We based the selection phase on filter structures provided by sequential pattern mining. On the one hand, the *support*, which filters patterns based on the frequency of apparition. On the other hand, the

closed frequent pattern set, which includes only patterns that are not included in another pattern having the same support.

Under this framework, i.e. the construction of macro-actions from extracted sequences of actions (not necessarily adjacent), we found out that regardless of the domain, the frequency alone may not provide a consistent selection of useful macro-actions. If there was a link between macro-actions frequency and macro-actions utility, from our study, we can conclude that the effect of this link is very weak compared to the negative effect caused by the lack of macro-actions generality and the use of non-adjacent actions.

7.1.2 Removing problematic macro-operators

In an attempt to diminish the influence of confounding factors (i.e. the lack of macro-actions generality and the use of non-adjacent actions) from the precedent study, we generalised macro-actions into macro-operators (i.e. sequences of actions with variable objects). We noticed that some of the created macro-operators were invalid (i.e. they cannot be applied to solve a problem) because of the extraction of sequences of not adjacent actions leading to the incompatibility of some predicates.

We proposed a formalism to identify three types of problematic macro-operators: the incompatible macro-operators, the useless macro-operators and the redundant macro-operators. The *incompatible* macro-operators cannot be applied during planning search because of the incompatibility of their predicates (e.g. `handempty` incompatible with `put-down blockA`). The *useless* macro-operators can be applied during planning search, but their application is equivalent either to do nothing (i.e. no changes in the state) or to use a primitive operator. The *redundant* macro-operators can be applied during planning search, but their application is equivalent to use a simpler macro-operator (i.e. a macro-operator with fewer actions).

The first type results from macro-operators created from sequences of actions whose actions do not appear contiguously in the solution plans. The second and the third type result from macro-operators created either from sequences of actions extracted from non-optimal solution plans or from sequences of actions whose actions do not appear contiguously in the solution plans.

From the definition of our formalism, we developed an algorithm to reduce problematic macro-operators by using a method which automatically detects predicates incompatibility. This approach proved to be successful in reducing problematic macro-operators. Because of its generality, it could be used in other planning applications.

However, we noticed that the use of sequential pattern mining led to a lack in the detection of all instances of a macro-operator. This conducted to an unreliable computation of the support. Despite the efforts, we find ourselves in a dead-end with the selection process because the pattern mining filtering structures are not adapted to planning.

7.1.3 METEOR framework

We provided a domain-independent learning framework that identifies sequences of actions (even non-adjacent) from past solution plans and selects the most useful macro-operators, based on *a priori* evaluation, to enhance the planning domain. It consist of two phases:

1. **Extraction phase**: We developed ERA, a pattern mining inspired algorithm which extracts rich patterns with attribute structures. It allows then to extract macro-operators from plan recycling and their characteristics (support, sequences ids where the macro appears and number of apparitions by sequence). We believe that this algorithm without the planning module can be a contribution to the pattern mining community.

2. **Selection phase**: We presented a formalism for the estimation of the trade-off between the branching factor increase and the search depth reduction not only for a single macro (as usually presented in the literature) but for a set of macros. By following this formalism, we developed an algorithm to choose an optimal macro-operator set that maximises our metric based on the *a priori* node gain.

Below, we highlight the advantages of our work:

- *Planner independent*, we do not need to modify the planner' structure.
- *Domain-independent*, the framework does not need *a priori* knowledge on the domain.
- *Non-adjacent actions*, we can identify routines from a sequence of adjacent and non-adjacent actions.
- *A priori evaluation*, there is no need to re-solve past problems to select the most useful routines.
- *An optimal set of routines*, we can identify not only useful routines but the optimal set to enhance a domain.

Our framework has proven to successfully mine macro-operators of different lengths for four different domains (barman, blocksworld, depots, satellite) and thanks to the selection phase, be able to deliver a positive impact on the search time (underestimated gain between 7 times and 200 times faster) with little influence on the quality of the plans (plan length difference was not more than 5 actions).

We present a comparative table between METEOR and other macro learning methods in Table 7.1 and Table 7.2. We observe that METEOR framework provides the most advantageous in the presented features. However, our framework presents some limitations that will be discussed in the next section.

	CA-ED	WIZARD	N-grams	MUM	DBMP/S	METEOR
Domain-independent	●	●	●	●	●	●
Planner independent	●	●	○	●	●	◐
Capable of handling non-adjacent actions	○	○	○	●	○	●
No hard limit on the # of generated macros	●	●	●	○	●	●
No hard limit on the length of generated macros	○	○	○	●	○	●
Does not need to solve problems with augmented domain to estimate macro utility	○	○	◐	●	○	●
Capable to evaluate the utility of a set of macros	○	●	○	○	●	●
Macro utility *a priori* evaluation based not only on macro frequency	-	-	●	●	-	●
No hard limit on the number of selected macros	○	○	○	○	○	●
Restriction on the selected macros	length=2	length and parameter count	equal length	none	single macro of length 2	non-overlapping

Table 7.1: Comparative table between METEOR and other macro learning methods. ●: has property ◐: partially has property ○: does not have property -: irrelevant.

	Used approach	
	Generation	Selection
CA-ED (Botea et al., 2005a)	Component abstraction	Performance on training problem solving
WIZARD (Newton et al., 2007)	Genetic algorithm	Performance on training problem solving
N-grams (Dulac et al., 2013)	N-grams analysis	Static using macro coverage and dynamic based on heuristics.
MUM (Chrpa et al., 2014)	Outer entanglements	Relational entanglement and component check
DBMP/S (Hofmann et al., 2017)	Map reduce query database paradigm	Most frequent macro of length 2
METEOR	ERA: PM based algorithm	A priori tradeoff estimation

Table 7.2: Overview of the approaches presented in comparative Table 7.1.

7.2 Limitations

7.2.1 Incompatibilities for *inertia* predicates

As defined by Koehler and Hoffmann (1999), *inertia* predicates represent propositions that are never produced or consumed by any operator. In the implementation of the incompatibility graph (see Chapter 5 Section 5.6), these predicates can be added to a predicate layer L_{n+1} during an expansion phase since they can be preconditions of some actions of the action layer L_n. During the link phase, however, inertia predicates can be linked neither to the top action layer because they do not appear as positive effects of any action, nor to the bottom action layer because they do not appear as negative effects of any action. Therefore, we cannot infer any incompatibilities for inertia predicates.

Besides, the transitivity property cannot infer any incompatibilities for inertia predicates either. In order to apply this property to two predicates, we must know part of their incompatibilities. We extract initial incompatibilities from the incompatibility graph, but as stated before this method cannot infer any incompatibilities for inertia predicates.

However, our main goal is to remove problematic macro operators. We eliminate them after detecting some incompatible predicates and the specific characteristics for each kind of problematic macro operator. In other words, to decide on whether or not a macro operator is problematic, we need to have at least two predicates mutually incompatible in its preconditions. Experimentally, we observed that most problematic macro-operators were captured.

7.2.2 Slight modification of the planner

The formalism proposed in Chapter 6 supposes that the heuristic has the same behaviour with or without macro-operators. Then, to have this assumption as accurate as possible, we ignore macro-operators during the heuristic computation. Besides that, we do not need to change any other structure of the planner or the way the planner behaves, i.e. apart from the heuristic computation, macro-operators and operators are indistinguishable. For a relaxation-based heuristic, we believe that not using macro-operators in the heuristic is an advantage rather than a limitation since they may cause more plateaus and thus, loose part of their guiding power for the search.

7.2.3 Set of non overlapping macro-operators

The selection phase of our METEOR framework is based on a formalism for the estimation of the trade-off between the branching factor increase and the search depth

reduction *not only for a single macro* (as usually presented in the literature) but for a set of macros. The algorithm of this selection phase chooses an optimal macro-operator set that maximises our metric based on the a priori node gain.

We, therefore, propose a new method to choose macro-operators that will enhance the planning domain. Although our method can only be applied to find a set composed of *non overlapping* macro-operators (see Chapter 6 Section 6.5), i.e. a set where each operator never appears more than once, it is the only limitation. Indeed, we have no limitations on the length of macro-operators (Dawson and Siklossy, 1977; Botea et al., 2005a; Dulac et al., 2013), or on the number of preconditions of the operators that compose a macro-operator (Jonsson, 2009) or on the number of macros to add to the planning domain (Dulac et al., 2013; Chrpa et al., 2014). We believe that this limitation is not too restrictive. Intuitively, macros sharing operators will probably have the same or a similar objective.

7.3 Perspectives

We have identified several research perspectives to extend this work.

Extend the study of the link between the frequency of a macro and its utility.

In this work, we explored the link between macro-action frequency and macro-action utility, for macro-actions built from sequences of actions (not necessarily adjacent) extracted using pattern mining algorithms on a set of solution plans.

More studies are required to investigate the link between frequency and utility under this framework, i.e. the construction of macro-actions from extracted sequences of actions (not necessarily adjacent), without the influence of confounding factors such that the lack of macro-actions generality and the presence of invalid macro-actions.

The study can be extended in different ways:

- the link between the utility of *valid* macro-actions built from sequences of actions not necessarily adjacent and their frequency.
- the link between the utility of *valid* macro-operators built from sequences of actions not necessarily adjacent and the reliable computation of their frequency. Notice that we do not recommend the use of pattern mining algorithms since they lead to an unreliable computation of the frequency via the macro-action generalisation method.

Study of different metrics for the total gain of a set of macro-operators.

In the selection phase of our METEOR framework, we first compute the macro set gain for each plan in the set of solution plans and then, we aggregate these gains to obtain the macro set gain over the set of solution plans. This lead us to a metric based on the arithmetic mean. However, outliers could influence this metric. For example, it could favour a macro with an exceptional utility on some solution plans but very limited utility on the rest of the set of solution plans.

Some metrics may be more appropriate to the great difference in the gains found in a set of solution plans. For example, the average log (geometric mean) of the gains could respond better to low gains. Another metric could be a weighted arithmetic (or geometric) mean which takes into account the complexity of each planning task. For example, a gain on a very difficult plan may be more valuable than a similar gain on a trivial plan. Finally, we could prefer a macro with more consistent performances than a macro with very erratic utility (i.e. sometimes very high but sometimes very low utility). This would lead to a metric based on the weighted difference between the average gain and the gain standard deviation.

Incremental learning

Intuitively, the macro-operators extracted and selected are dependent on our learning set of solution plans. Therefore, the effectiveness of macro-operators used on a planning domain is dependent on how well the problems in the training set represent the problems encountered during the application.

In addition, it is conceivable that in real learning, the objectives to be met could change periodically. Let us consider as example, a change in the products to be manufactured in a production line. A production line is in charge of three tasks: A, B and C. A change occurs: task A is removed and a new task D is added. The system must therefore forget the macro-operator that achieves task A and learn the macro-operator that achieves task D.

To overcome this problem, an incremental learning system could be implemented. First, the learning set of solution plans would consist of a sliding window of n past plans. Second, every m solved plans the system would forget its experiences and start a new learning process from scratch. Obviously, n and m would be very important parameters and they will be one of the main concerns of this study.

For small values of n, the computation time would be low as well as the reaction time, but learning would not be robust against small variations in the objectives. For large values of n, there would be some delay between an actual changing objective and the adaption of the system. Finally, we would like m as small as possible depending on the computation time constraint on n.

A

Benchmark domains

These benchmarks problems were taken from past International Planning Competitions[1].

For each benchmark domain, we present:

- A description.
- The generator parameters.
- The domain objects and their respective types.
- A graphical description of the operators domain.

A.1 Barman

A.1.1 Description

A robot barman manipulates drink dispensers, glasses and a shaker. The goal is to find a plan of the robot's actions that serves a desired set of drinks. In this domain deletes of actions encode relevant knowledge given that robot hands can only grasp one object at a time and given that glasses need to be empty and clean to be filled.

[1] http://icaps-conference.org/index.php/Main/Competitions

Parameters

- number of cocktails
- number of ingredients

Objects

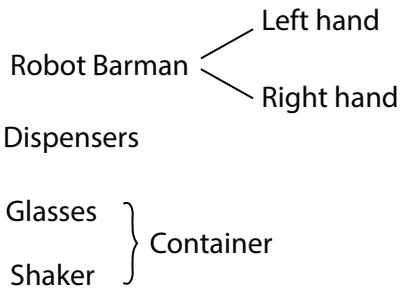


Figure A.1: Barman objects

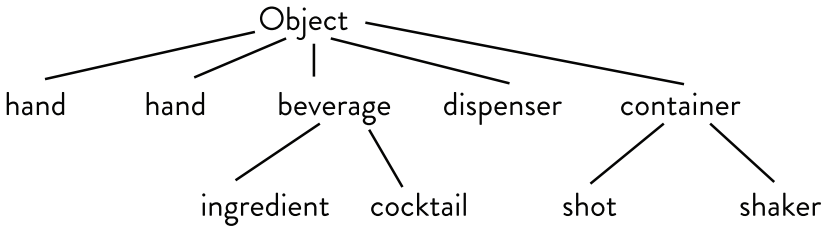


Figure A.2: Barman object types

Operators

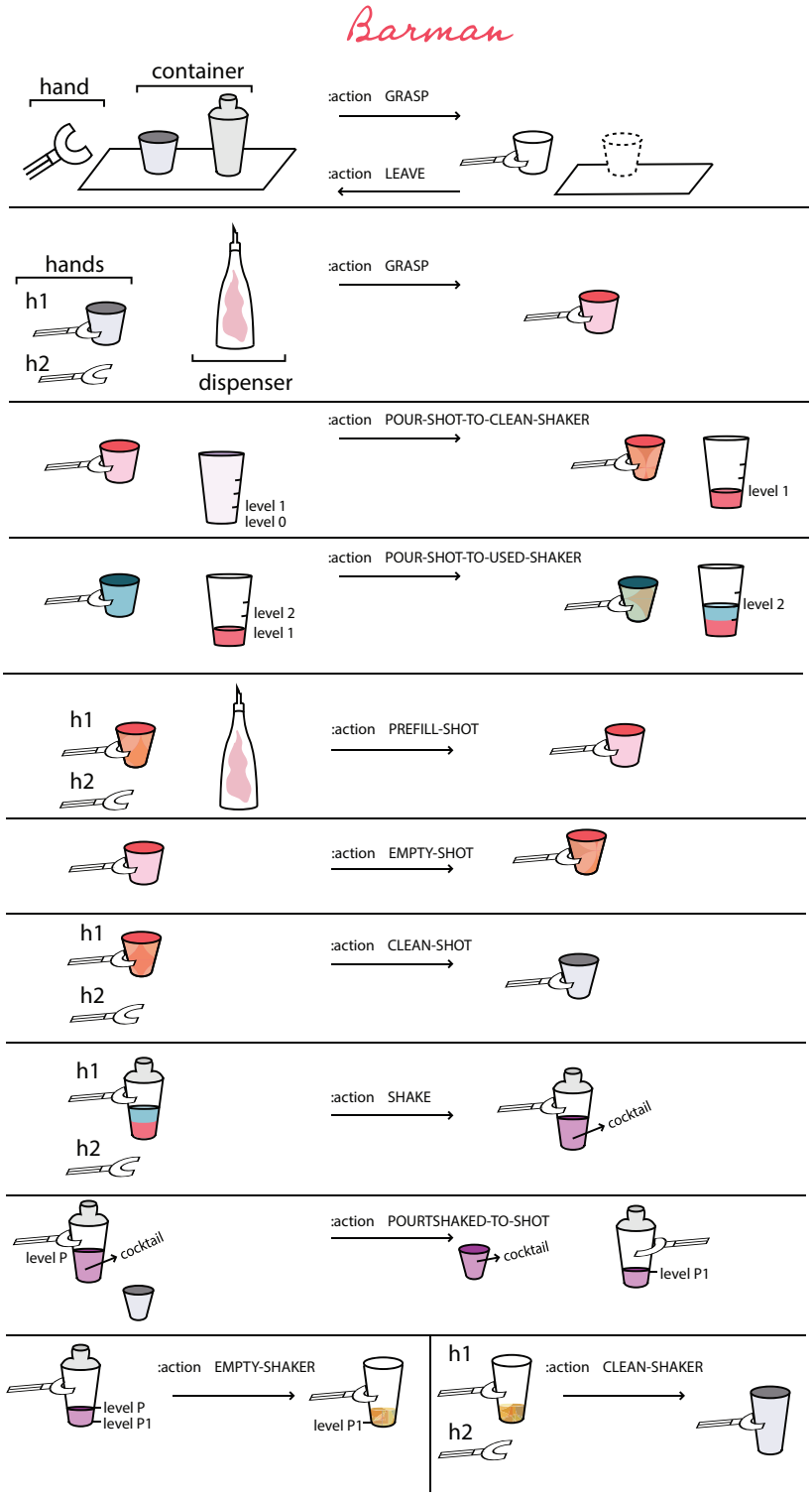


Figure A.3: Barman operators

A.2 Blocksworld

A.2.1 Description

It consists of a set of blocks settled on a table and a mechanical hand. The hand can move one block at a time to perform one of the following actions: place it on another block, place it on the table, pick it from the table or removes it from another block. The goal is to build one or more vertical stacks of blocks.

Parameters

- number of blocks

Objects

- Block

Operators

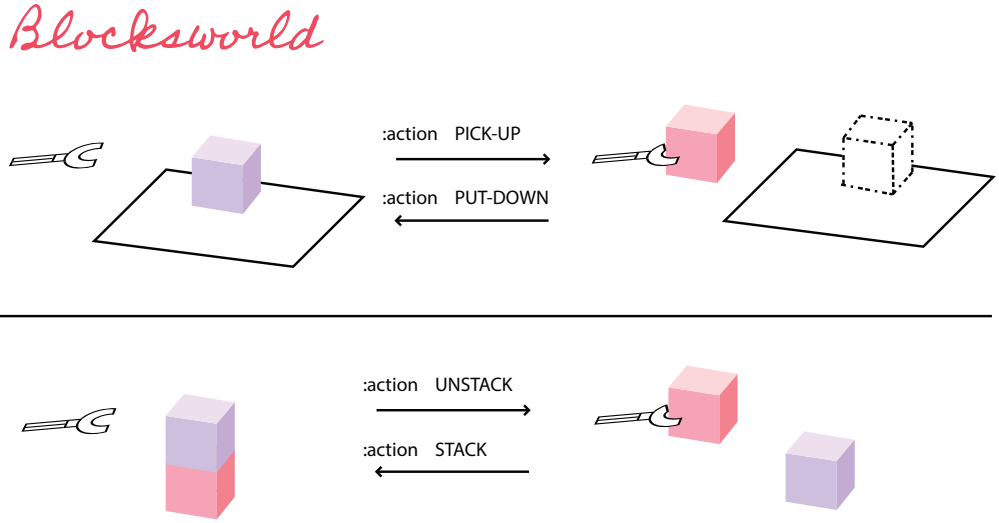


Figure A.4: Blocksworld operators

A.3 Depots

A.3.1 Description

This domain is a combination of a transportation domain and the Blocksworld domain. The transportation element of the task is to move crates from one depot to another using

trucks. The blocksworld element arises due to the need to stack and unstack crates, with the amount of space on the 'table' being limited by the number of pallets at each location. Hoists server the function of the robot arm, doubling as the mechanism by which crates are loaded/unloaded onto/from trucks. The goal is to find a plan where crates are stacked appropriately at their destinations.

Parameters

- number of depots
- number of distributors
- number of trucks
- number of pallets
- number of hoists
- number of crates
- number of ingredients

Objects

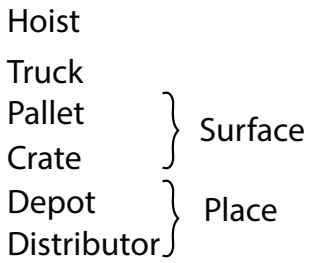


Figure A.5: Depots objects

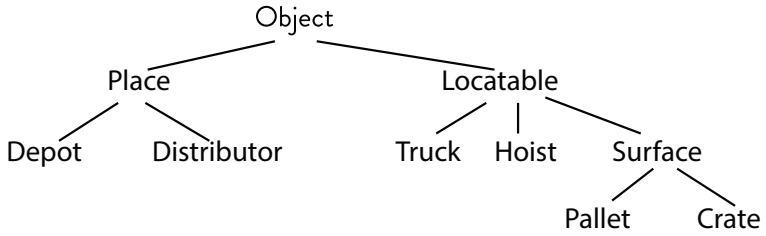


Figure A.6: Depots objects types

Operators

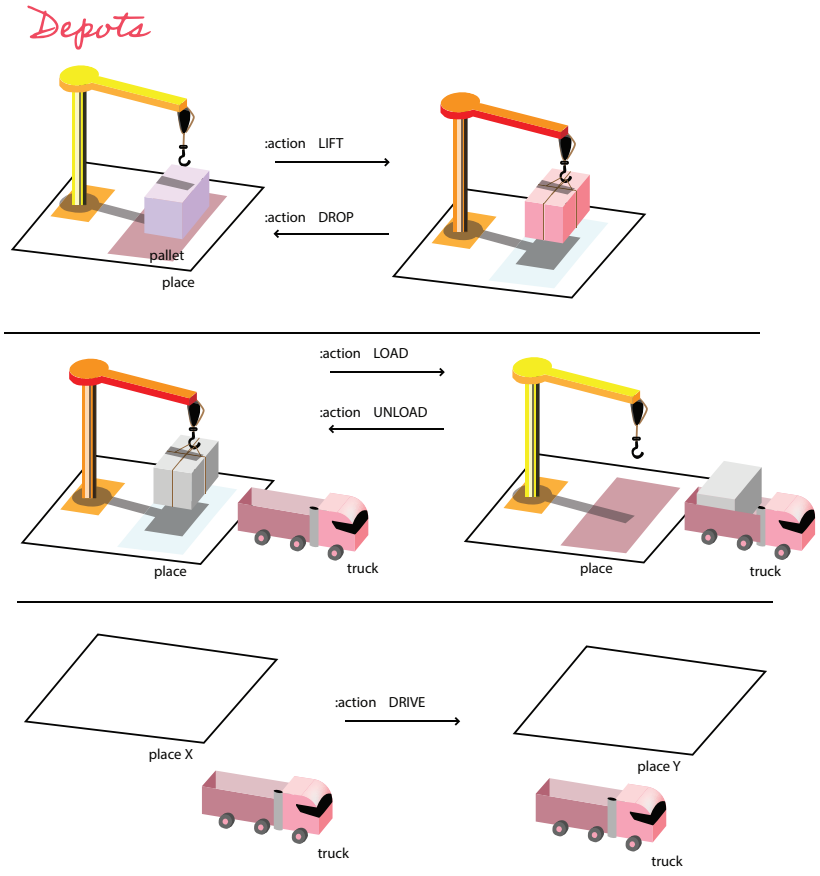


Figure A.7: Depots operators

A.4 Satellite

A.4.1 Description

In this domain there is a set of satellites equipped with different instruments, which can operate in different modes. The goal is to acquire desired images, dividing the observation tasks between the satellites, based on the capabilities of their instruments.

Parameters

- number of satellites
- number of instruments
- number of modes
- number of targets
- number of observations

Objects

- satellite
- instrument
- direction
- mode

Operators

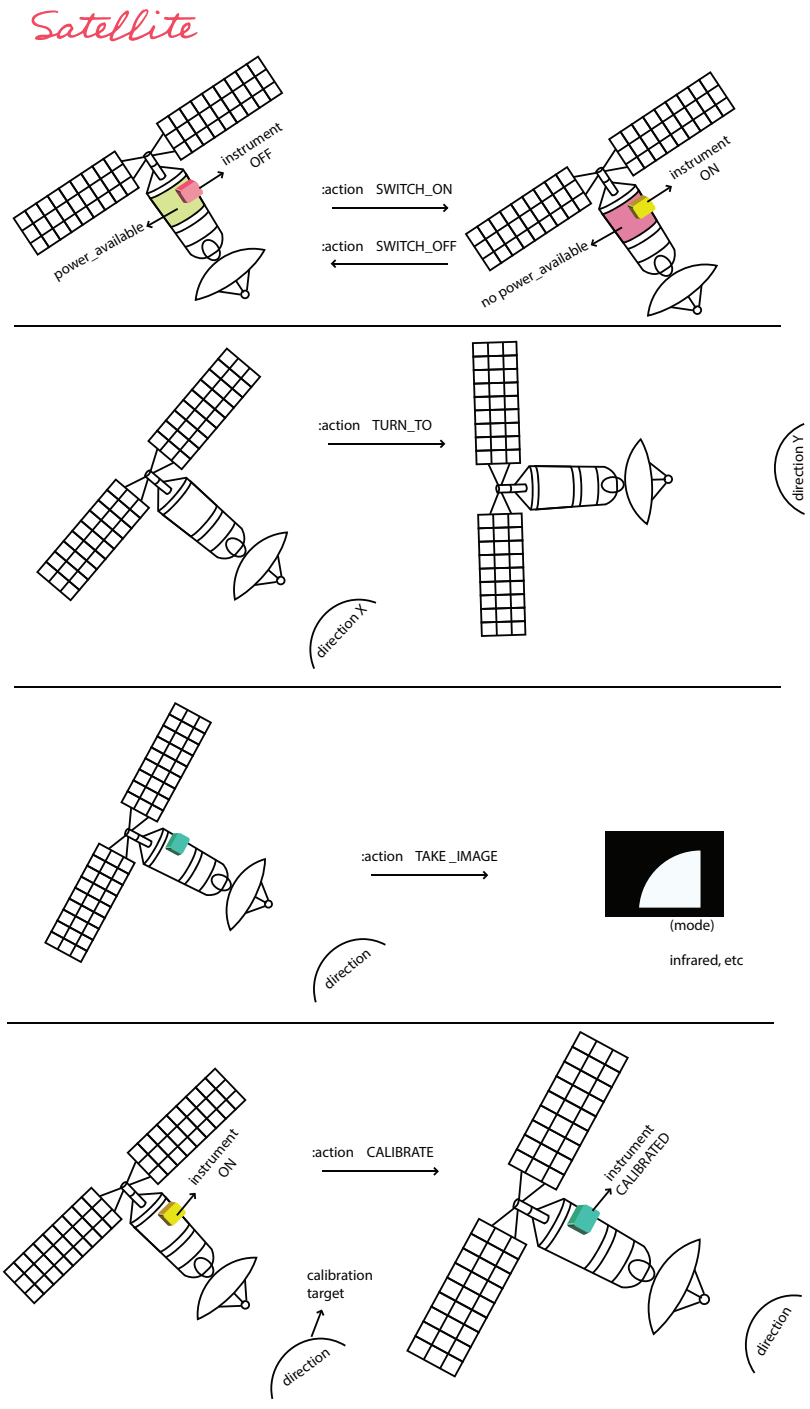


Figure A.8: Satellite operators

B

Understanding results: operators translation and full report

B.1 Barman

B.1.1 Operators translation

```
 0: grasp
 1: leave
 2: fill-shot
 3: refill-shot
 4: empty-shot
 5: clean-shot
 6: pour-shot-to-clean-shaker
 7: pour-shot-to-used-shaker
 8: empty-shaker
 9: clean-shaker
10: shake
11: pour-shaker-to-shot
```

B.1.2 Mining Log

```
52 macros extracted
Macro : macro-2-actions-3-1--1000 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-2-7--1001 with support 50 and 100 total apparitions (mean of 2.0/plan)
Macro : macro-2-actions-10-11--1002 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-0-2--1003 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-6-5--1004 with support 50 and 100 total apparitions (mean of 2.0/plan)
Macro : macro-2-actions-0-10--1005 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-9-1--1006 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-7-3--1007 with support 50 and 79 total apparitions (mean of 1.58/plan)
Macro : macro-2-actions-9-0--1008 with support 42 and 42 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-5-2--1009 with support 50 and 100 total apparitions (mean of 2.0/plan)
Macro : macro-2-actions-11-8--1010 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-0-6--1011 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-8-9--1012 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-2-actions-2-6--1013 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-2-7-3--1014 with support 50 and 79 total apparitions (mean of 1.58/plan)
Macro : macro-3-actions-6-5-2--1015 with support 50 and 100 total apparitions (mean of 2.0/plan)
Macro : macro-3-actions-0-6-5--1016 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-8-9-1--1017 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-8-9-0--1018 with support 42 and 42 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-0-10-11--1019 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-7-3-1--1020 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-2-6-5--1021 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-0-2-6--1022 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-5-2-7--1023 with support 50 and 100 total apparitions (mean of 2.0/plan)
Macro : macro-3-actions-10-11-8--1024 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-3-actions-11-8-9--1025 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-4-actions-0-6-5-2--1026 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-4-actions-2-6-5-2--1027 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-4-actions-5-2-7-3--1028 with support 50 and 79 total apparitions (mean of 1.58/plan)
```

```
Macro : macro-4-actions-10-11-8-9--1029 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-4-actions-6-5-2-7--1030 with support 50 and 100 total apparitions (mean of 2.0/plan)
Macro : macro-4-actions-0-10-11-8--1031 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-4-actions-0-2-6-5--1032 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-4-actions-2-7-3-1--1033 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-4-actions-11-8-9-0--1034 with support 42 and 42 total apparitions (mean of 1.0/plan)
Macro : macro-4-actions-11-8-9-1--1035 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-5-actions-0-10-11-8-9--1036 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-5-actions-6-5-2-7-3--1037 with support 50 and 79 total apparitions (mean of 1.58/plan)
Macro : macro-5-actions-0-6-5-2-7--1038 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-5-actions-10-11-8-9-1--1039 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-5-actions-0-2-6-5-2--1040 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-5-actions-2-6-5-2-7--1041 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-5-actions-10-11-8-9-0--1042 with support 42 and 42 total apparitions (mean of 1.0/plan)
Macro : macro-5-actions-5-2-7-3-1--1043 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-6-actions-0-2-6-5-2-7--1044 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-6-actions-0-10-11-8-9-1--1045 with support 50 and 50 total apparitions (mean of 1.0/plan)
Macro : macro-6-actions-2-6-5-2-7-3--1046 with support 48 and 48 total apparitions (mean of 1.0/plan)
Macro : macro-6-actions-6-5-2-7-3-1--1047 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-6-actions-0-10-11-8-9-0--1048 with support 42 and 42 total apparitions (mean of 1.0/plan)
Macro : macro-7-actions-0-2-6-5-2-7-3--1049 with support 48 and 48 total apparitions (mean of 1.0/plan)
Macro : macro-7-actions-2-6-5-2-7-3-1--1050 with support 45 and 45 total apparitions (mean of 1.0/plan)
Macro : macro-8-actions-0-2-6-5-2-7-3-1--1051 with support 45 and 45 total apparitions (mean of 1.0/plan)
```

B.1.3 Macro analyser log

```
macro-2-actions-3-1--1000 : total Gain = 0.09577121805998921
macro-2-actions-2-7--1001 : total Gain = 0.011679458833872667
macro-2-actions-10-11--1002 : total Gain = 0.8413896113524574
macro-2-actions-0-2--1003 : total Gain = 0.07146436302150379
macro-2-actions-6-5--1004 : total Gain = 1.306082546379102
macro-2-actions-0-10--1005 : total Gain = 0.9260495081948406
macro-2-actions-9-1--1006 : total Gain = 0.40335631667120797
macro-2-actions-7-3--1007 : total Gain = 0.06398043137259492
macro-2-actions-9-0--1008 : total Gain = 1.6838690121344482
macro-2-actions-5-2--1009 : total Gain = 3.311917328034674
macro-2-actions-11-8--1010 : total Gain = 0.8413896113524574
macro-2-actions-0-6--1011 : total Gain = 0.6765093034230286
macro-2-actions-8-9--1012 : total Gain = 0.40335631667120797
macro-2-actions-2-6--1013 : total Gain = 0.3289867762519236
macro-3-actions-2-7-3--1014 : total Gain = 0.06091256610049259
macro-3-actions-6-5-2--1015 : total Gain = 4.058107415362549
macro-3-actions-0-6-5--1016 : total Gain = 2.38894238784685
macro-3-actions-8-9-1--1017 : total Gain = 0.7985696613288876
macro-3-actions-8-9-0--1018 : total Gain = 2.9465929819029277
macro-3-actions-0-10-11--1019 : total Gain = 2.032888067461868
macro-3-actions-7-3-1--1020 : total Gain = 0.0028072000981503795
macro-3-actions-2-6-5--1021 : total Gain = 0.6562502368423868
macro-3-actions-0-2-6--1022 : total Gain = 0.15401426663115156
macro-3-actions-5-2-7--1023 : total Gain = 11.349024474245716
macro-3-actions-10-11-8--1024 : total Gain = 1.602105675220187
macro-3-actions-11-8-9--1025 : total Gain = 1.602105675220187
macro-4-actions-0-6-5-2--1026 : total Gain = 4.312695368579807
macro-4-actions-2-6-5-2--1027 : total Gain = 0.2127831090586036
macro-4-actions-5-2-7-3--1028 : total Gain = 30.757761367764267
macro-4-actions-10-11-8-9--1029 : total Gain = 3.076830260901096
macro-4-actions-6-5-2-7--1030 : total Gain = 16.077352064581554
macro-4-actions-0-10-11-8--1031 : total Gain = 3.8379972491745047
macro-4-actions-0-2-6-5--1032 : total Gain = 0.33483362387540466
macro-4-actions-2-7-3-1--1033 : total Gain = 0.028242245072074637
macro-4-actions-11-8-9-0--1034 : total Gain = 5.294994215979314
macro-4-actions-11-8-9-1--1035 : total Gain = 3.076830260901096
macro-5-actions-0-10-11-8-9--1036 : total Gain = 7.305970576066381
macro-5-actions-6-5-2-7-3--1037 : total Gain = 55.223603705058615
macro-5-actions-0-6-5-2-7--1038 : total Gain = 7.99108758229327
macro-5-actions-10-11-8-9-1--1039 : total Gain = 5.957312015759768
macro-5-actions-0-2-6-5-2--1040 : total Gain = 0.4340652545671022
macro-5-actions-2-6-5-2-7--1041 : total Gain = 0.47017127308198825
macro-5-actions-10-11-8-9-0--1042 : total Gain = 9.692475087497211
macro-5-actions-5-2-7-3-1--1043 : total Gain = 10.057982944920033
macro-6-actions-0-2-6-5-2-7--1044 : total Gain = 0.9672359175489974
macro-6-actions-0-10-11-8-9-1--1045 : total Gain = 14.016530032193836
macro-6-actions-2-6-5-2-7-3--1046 : total Gain = 1.0449423879929363
```

```
macro-6-actions-6-5-2-7-3-1--1047 : total Gain = 7.7249235709277855
macro-6-actions-0-10-11-8-9-0--1048 : total Gain = 17.980106909866578
macro-7-actions-0-2-6-5-2-7-3--1049 : total Gain = 2.167779240066138
macro-7-actions-2-6-5-2-7-3-1--1050 : total Gain = 2.3355802641005123
macro-8-actions-0-2-6-5-2-7-3-1--1051 : total Gain = 4.859857286152655
```

B.1.4 Recommended Optimal Macro Set

```
macro-6-actions-0-10-11-8-9-0--1048 with gain : 17.980106909866578
macro-5-actions-6-5-2-7-3--1037 with gain : 55.223603705058615
for total optimal gain : 96937.14826813425
```

B.2 Blocksworld

B.2.1 Operators translation

```
0: pick-up
1: put-down
2: stack
3: unstack
```

B.2.2 Mining Log

```
6 macros extracted
Macro : macro-2-actions-3-1--1000 with support 50 and 149 total apparitions
(mean of 2.98/plan)
Macro : macro-2-actions-0-2--1001 with support 47 and 114 total apparitions
(mean of 2.425531914893617/plan)
Macro : macro-2-actions-3-2--1002 with support 50 and 215 total apparitions
(mean of 4.3/plan)
Macro : macro-3-actions-0-2-3--1003 with support 42 and 60 total apparitions
(mean of 1.4285714285714286/plan)
Macro : macro-3-actions-3-2-3--1004 with support 43 and 74 total apparitions
(mean of 1.7209302325581395/plan)
Macro : macro-3-actions-3-1-3--1005 with support 43 and 76 total apparitions
(mean of 1.7674418604651163/plan)
```

B.2.3 Macro analyser log

```
macro-2-actions-3-1--1000 : total Gain = 0.15271442029409768
macro-2-actions-0-2--1001 : total Gain = 0.014905674155709975
macro-2-actions-3-2--1002 : total Gain = 0.006369325185660316
macro-3-actions-0-2-3--1003 : total Gain = 0.0004495624534990441
macro-3-actions-3-2-3--1004 : total Gain = 4.585863731632646e-05
macro-3-actions-3-1-3--1005 : total Gain = 0.0028053793719730632
```

B.2.4 Recommended Optimal Macro Set

```
macro-2-actions-3-1--1000 with gain : 0.15271442029409768
macro-2-actions-0-2--1001 with gain : 0.014905674155709975
for total optimal gain : 0.3813509128794204
```

B.3 Depots

B.3.1 Operators translation

```
    0: drive
    1: lift
    2: drop
    3: load
    4: unload
```

B.3.2 Mining Log

```
10 macros extracted
Macro : macro-2-actions-3-0--1000 with support 48 and 86 total apparitions
(mean of 1.7916666666666667/plan)
Macro : macro-2-actions-3-1--1001 with support 47 and 117 total apparitions
(mean of 2.4893617021276597/plan)
Macro : macro-2-actions-1-3--1002 with support 50 and 256 total apparitions
(mean of 5.12/plan)
Macro : macro-2-actions-4-2--1003 with support 50 and 116 total apparitions
(mean of 2.32/plan)
Macro : macro-2-actions-0-3--1004 with support 46 and 100 total apparitions
```

```
(mean of 2.1739130434782608/plan)
Macro : macro-3-actions-1-3-1--1005 with support 46 and 87 total apparitions
(mean of 1.891304347826087/plan)
Macro : macro-3-actions-3-1-3--1006 with support 44 and 77 total apparitions
(mean of 1.75/plan)
Macro : macro-3-actions-1-3-0--1007 with support 48 and 86 total apparitions
(mean of 1.7916666666666667/plan)
Macro : macro-3-actions-1-3-1--1008 with support 48 and 84 total apparitions
(mean of 1.75/plan)
Macro : macro-4-actions-1-3-1-3--1009 with support 43 and 76 total apparitions
(mean of 1.7674418604651163/plan)
```

B.3.3 Macro analyser log

```
macro-2-actions-3-0--1000 : total Gain = 0.17963054581634705
macro-2-actions-3-1--1001 : total Gain = 3.2698550109114723
macro-2-actions-1-3--1002 : total Gain = 14.107121929491381
macro-2-actions-4-2--1003 : total Gain = 1.606629581472632
macro-2-actions-0-3--1004 : total Gain = 1.059255162057205
macro-3-actions-1-3-1--1005 : total Gain = 6.51992081245396
macro-3-actions-3-1-3--1006 : total Gain = 5.633222610787586
macro-3-actions-1-3-0--1007 : total Gain = 0.8685936682235017
macro-3-actions-1-3-1--1008 : total Gain = 4.88311731220142
macro-4-actions-1-3-1-3--1009 : total Gain = 19.524723021667302
```

B.3.4 Recommended Optimal Macro Set

```
macro-4-actions-1-3-1-3--1009 with gain : 19.524723021667302
macro-2-actions-4-2--1003 with gain : 1.606629581472632
for total optimal gain : 1602.106171520573
```

B.4 Satellite

B.4.1 Operators translation

```
0: turn_to
1: switch_on
2: switch_off
3: calibrate
4: take_image
```

B.4.2 Mining Log

```
13 macros extracted
Macro : macro-2-actions-0-3--1000 with support 43 and 64 total apparitions
(mean of 1.4883720930232558/plan)
Macro : macro-2-actions-4-0--1001 with support 50 and 287 total apparitions
(mean of 5.74/plan)
Macro : macro-2-actions-1-3--1002 with support 50 and 80 total apparitions
(mean of 1.6/plan)
Macro : macro-2-actions-3-0--1003 with support 50 and 79 total apparitions
(mean of 1.58/plan)
Macro : macro-2-actions-0-4--1004 with support 50 and 331 total apparitions
(mean of 6.62/plan)
Macro : macro-3-actions-1-3-0--1005 with support 50 and 79 total apparitions
(mean of 1.58/plan)
Macro : macro-3-actions-3-0-4--1006 with support 50 and 76 total apparitions
(mean of 1.52/plan)
Macro : macro-3-actions-4-0-4--1007 with support 44 and 100 total apparitions
(mean of 2.272727272727273/plan)
Macro : macro-3-actions-0-4-0--1008 with support 50 and 155 total apparitions
(mean of 3.1/plan)
Macro : macro-4-actions-0-4-0-4--1009 with support 43 and 96 total apparitions
(mean of 2.2325581395348837/plan)
Macro : macro-4-actions-3-0-4-0--1010 with support 48 and 63 total apparitions
(mean of 1.3125/plan)
Macro : macro-4-actions-1-3-0-4--1011 with support 50 and 76 total apparitions
(mean of 1.52/plan)
Macro : macro-5-actions-1-3-0-4-0--1012 with support 48 and 63 total apparitions
(mean of 1.3125/plan)
```

B.4.3 Macro analyser log

```
macro-2-actions-0-3--1000 : total Gain = 0.9372774673267826
macro-2-actions-4-0--1001 : total Gain = 0.07481345300457362
macro-2-actions-1-3--1002 : total Gain = 2.735464323973243
macro-2-actions-3-0--1003 : total Gain = 0.6999653507639872
macro-2-actions-0-4--1004 : total Gain = 0.18351028000047603
macro-3-actions-1-3-0--1005 : total Gain = 7.836104007771476
macro-3-actions-3-0-4--1006 : total Gain = 1.302056574593954
macro-3-actions-4-0-4--1007 : total Gain = 0.13488815352205202
macro-3-actions-0-4-0--1008 : total Gain = 7.6151103258830905e-06
macro-4-actions-0-4-0-4--1009 : total Gain = 0.0027120438914648346
macro-4-actions-3-0-4-0--1010 : total Gain = 0.013685838564669378
macro-4-actions-1-3-0-4--1011 : total Gain = 23.358343546987943
macro-5-actions-1-3-0-4-0--1012 : total Gain = 29.82248891732685
```

B.4.4 Recommended Optimal Macro Set

```
macro-2-actions-1-3--1002 with gain : 2.735464323973243
macro-2-actions-0-4--1004 with gain : 0.18351028000047603
for total optimal gain : 34.00268187509237
```

C

Results from the graph-based approach to remove problematic macro-operators

C.1 Barman

C.1.1 Predicate incompatibilities

empty ?shot0 | contains ?shot0 ?ingredient0, contains ?shot0 ?ingredient3, contains ?shot0 ?ingredient1, contains ?shot0 ?ingredient2
dispenses ?dispenser1 ?ingredient2 |
dispenses ?dispenser1 ?ingredient1 |

empty ?shot1 | contains ?shot1 ?ingredient3, contains ?shot1 ?ingredient0, contains ?shot1 ?ingredient2, contains ?shot1 ?ingredient1
ontable ?shot0 | holding ?hand0 ?shot0, holding ?hand2 ?shot0, holding ?hand1 ?shot0
clean ?shot1 | used ?shot1 ?ingredient0, used ?shot1 ?ingredient2, used ?shot1 ?ingredient1, used ?shot1 ?ingredient3
next ?level1 ?level0 |
shaker-level ?shaker1 ?level1 | shaker-level ?shaker1 ?level3, shaker-level ?shaker1 ?level2, shaker-level ?shaker1 ?level0
dispenses ?dispenser0 ?ingredient0 |
shaker-level ?shaker0 ?level2 | shaker-level ?shaker0 ?level3, shaker-level ?shaker0 ?level1, shaker-level ?shaker0 ?level0

empty ?shaker1 |
next ?level2 ?level1 |
dispenses ?dispenser1 ?ingredient0 |
next ?level1 ?level2 |
shaker-level ?shaker0 ?level1 | shaker-level ?shaker0 ?level3, shaker-level ?shaker0 ?level2, shaker-level ?shaker0 ?level0
dispenses ?dispenser0 ?ingredient1 |
shaker-level ?shaker1 ?level0 | shaker-level ?shaker1 ?level3, shaker-level ?shaker1 ?level2, shaker-level ?shaker1 ?level1
next ?level0 ?level2 |
handempty ?hand0 | holding ?hand0 ?shaker1, holding ?hand0 ?shaker2, holding ?hand0 ?shaker0, holding ?hand0 ?shot0, holding ?hand0 ?shaker3, holding ?hand0 ?shot1, holding ?hand0 ?shot2
next ?level0 ?level1 |
clean ?shot0 | used ?shot0 ?ingredient0, used ?shot0 ?ingredient2, used ?shot0 ?ingredient1, used ?shot0 ?ingredient3
shaker-level ?shaker0 ?level0 | shaker-level ?shaker0 ?level3, shaker-level ?shaker0 ?level2, shaker-level ?shaker0 ?level1
handempty ?hand1 | holding ?hand1 ?shaker0, holding ?hand1 ?shaker3, holding ?hand1 ?shot1, holding ?hand1 ?shaker2, holding ?hand1 ?shot2, holding ?hand1 ?shaker1, holding ?hand1 ?shot0
clean ?shaker0 |
handempty ?hand2 | holding ?hand2 ?shaker3, holding ?hand2 ?shot2, holding ?hand2 ?shaker1, holding ?hand2 ?shot1, holding ?hand2 ?shaker2, holding ?hand2 ?shot0, holding ?hand2 ?shaker0
empty ?shaker0 |
dispenses ?dispenser0 ?ingredient2 |
ontable ?shot1 | holding ?hand2 ?shot1, holding ?hand0 ?shot1, holding ?hand1 ?shot1
next ?level2 ?level0 |
shaker-level ?shaker1 ?level2 | shaker-level ?shaker1 ?level3, shaker-level ?shaker1 ?level0, shaker-level ?shaker1 ?level1
clean ?shaker1 |
ontable ?shaker1 | holding ?hand2 ?shaker1, holding ?hand1 ?shaker1, holding ?hand0 ?shaker1
contains ?shot1 ?ingredient0 | empty ?shot1, contains ?shot1 ?ingredient2, contains ?shot1 ?ingredient1
contains ?shot1 ?ingredient1 | contains ?shot1 ?ingredient0, empty ?shot1, contains ?shot1 ?ingredient2
unshaked ?shaker0 |
contains ?shot1 ?ingredient2 | contains ?shot1 ?ingredient0, empty ?shot1, contains ?shot1 ?ingredient1
unshaked ?shaker1 |
contains ?shot0 ?ingredient2 | contains ?shot0 ?ingredient0, empty ?shot0, contains ?shot0 ?ingredient1
ontable ?shaker0 | holding ?hand1 ?shaker0, holding ?hand2 ?shaker0, holding ?hand0 ?shaker0
contains ?shot0 ?ingredient0 | empty ?shot0, contains ?shot0 ?ingredient1, contains ?shot0 ?ingredient2
contains ?shot0 ?ingredient1 | contains ?shot0 ?ingredient0, empty ?shot0, contains ?shot0 ?ingredient2
shaker-empty-level ?shaker0 ?level2 |

shaker-empty-level ?shaker0 ?level0 |
shaker-empty-level ?shaker0 ?level1 |
contains ?shaker1 ?cocktail1 |
contains ?shaker0 ?cocktail1 |
shaked ?shaker1 |
shaker-empty-level ?shaker1 ?level0 |
contains ?shaker1 ?cocktail0 |
shaker-empty-level ?shaker1 ?level2 |
shaked ?shaker0 |
contains ?shaker0 ?cocktail0 |
shaker-empty-level ?shaker1 ?level1 |
ontable ?shot2 | holding ?hand1 ?shot2, holding ?hand0 ?shot2, holding ?hand2 ?shot2
empty ?shot2 | contains ?shot2 ?ingredient1, contains ?shot2 ?ingredient0, contains ?shot2 ?ingredient2
clean ?shot2 | used ?shot2 ?ingredient0, used ?shot2 ?ingredient2, used ?shot2 ?ingredient1
contains ?shaker0 ?ingredient0 |
contains ?shaker1 ?ingredient1 |
contains ?shaker1 ?ingredient2 |
contains ?shaker0 ?ingredient1 |
contains ?shaker0 ?ingredient2 |
contains ?shaker1 ?ingredient0 |
contains ?shot2 ?ingredient0 | contains ?shot2 ?ingredient1, empty ?shot2, contains ?shot2 ?ingredient2
holding ?hand2 ?shot1 | holding ?hand1 ?shot1, handempty ?hand2, holding ?hand2 ?shot2, ontable ?shot1, holding ?hand2 ?shot0, holding ?hand0 ?shot1
contains ?shot2 ?ingredient1 | contains ?shot2 ?ingredient0, empty ?shot2, contains ?shot2 ?ingredient2
holding ?hand1 ?shot1 | handempty ?hand1, holding ?hand1 ?shot2, holding ?hand2 ?shot1, ontable ?shot1, holding ?hand0 ?shot1, holding ?hand1 ?shot0
holding ?hand1 ?shot2 | handempty ?hand1, ontable ?shot2, holding ?hand1 ?shot1, holding ?hand2 ?shot2, holding ?hand1 ?shot0, holding ?hand0 ?shot2
holding ?hand2 ?shot2 | ontable ?shot2, handempty ?hand2, holding ?hand1 ?shot2, holding ?hand2 ?shot1, holding ?hand2 ?shot0, holding ?hand0 ?shot2
holding ?hand0 ?shot1 | holding ?hand1 ?shot1, holding ?hand0 ?shot0, handempty ?hand0, ontable ?shot1, holding ?hand2 ?shot1, holding ?hand0 ?shot2
holding ?hand0 ?shot2 | ontable ?shot2, holding ?hand2 ?shot2, holding ?hand1 ?shot2, holding ?hand0 ?shot0, handempty ?hand0, holding ?hand0 ?shot1
contains ?shot2 ?ingredient2 | contains ?shot2 ?ingredient1, contains ?shot2 ?ingredient0, empty ?shot2
holding ?hand0 ?shot0 | handempty ?hand0, holding ?hand2 ?shot0, holding ?hand0 ?shot1, holding ?hand1 ?shot0, holding ?hand0 ?shot2, ontable ?shot0
holding ?hand2 ?shot0 | handempty ?hand2, holding ?hand2 ?shot2, holding ?hand0 ?shot0, holding ?hand2 ?shot1, holding ?hand1 ?shot0, ontable ?shot0
holding ?hand1 ?shot0 | handempty ?hand1, holding ?hand1 ?shot1, holding ?hand1 ?shot2, holding ?hand0 ?shot0, holding ?hand2 ?shot0, ontable ?shot0
holding ?hand1 ?shaker0 | handempty ?hand1, holding ?hand1 ?shaker3, holding ?hand1 ?shaker2, holding ?hand0 ?shaker0, holding ?hand1 ?shaker1, ontable ?shaker0, holding ?hand2 ?shaker0
holding ?hand0 ?shaker1 | ontable ?shaker1, holding ?hand0 ?shaker2, holding ?hand0 ?shaker0, holding ?hand2 ?shaker1, handempty ?hand0, holding ?hand1 ?shaker1, hold-

ing ?hand0 ?shaker3
cocktail-part2 ?cocktail0 ?ingredient1 |
cocktail-part1 ?cocktail0 ?ingredient1 |
holding ?hand1 ?shaker1 | ontable ?shaker1, handempty ?hand1, holding ?hand1 ?shaker0, holding ?hand0 ?shaker1, holding ?hand1 ?shaker2, holding ?hand2 ?shaker1, holding ?hand1 ?shaker3
cocktail-part2 ?cocktail1 ?ingredient2 |
cocktail-part1 ?cocktail0 ?ingredient0 |
cocktail-part1 ?cocktail0 ?ingredient2 |
cocktail-part1 ?cocktail1 ?ingredient0 |
cocktail-part2 ?cocktail0 ?ingredient0 |
cocktail-part2 ?cocktail1 ?ingredient0 |
holding ?hand2 ?shaker1 | ontable ?shaker1, holding ?hand0 ?shaker1, holding ?hand2 ?shaker3, handempty ?hand2, holding ?hand1 ?shaker1, holding ?hand2 ?shaker2, holding ?hand2 ?shaker0
cocktail-part1 ?cocktail1 ?ingredient1 |
holding ?hand2 ?shaker0 | holding ?hand1 ?shaker0, holding ?hand2 ?shaker3, holding ?hand0 ?shaker0, handempty ?hand2, holding ?hand2 ?shaker1, holding ?hand2 ?shaker2, ontable ?shaker0
cocktail-part2 ?cocktail0 ?ingredient2 |
cocktail-part2 ?cocktail1 ?ingredient1 |
holding ?hand0 ?shaker0 | holding ?hand1 ?shaker0, holding ?hand0 ?shaker1, holding ?hand0 ?shaker2, holding ?hand2 ?shaker0, handempty ?hand0, holding ?hand0 ?shaker3, ontable ?shaker0
cocktail-part1 ?cocktail1 ?ingredient2 |
used ?shot0 ?cocktail0 |
used ?shot1 ?ingredient1 |
used ?shot1 ?cocktail0 |
used ?shot0 ?ingredient2 |
used ?shot1 ?ingredient2 |
used ?shot0 ?ingredient0 |
used ?shot0 ?ingredient1 |
used ?shot1 ?ingredient0 |
used ?shot0 ?cocktail1 |
used ?shot1 ?cocktail1 |
used ?shot2 ?cocktail0 |
used ?shot2 ?ingredient1 |
used ?shot2 ?ingredient0 |
used ?shot2 ?cocktail1 |
used ?shot2 ?ingredient2 |
contains ?shaker2 ?ingredient2 |
holding ?hand1 ?shaker2 | holding ?hand1 ?shaker0, handempty ?hand1, holding ?hand0 ?shaker2, ontable ?shaker2, holding ?hand1 ?shaker1, holding ?hand2 ?shaker2, holding ?hand1 ?shaker3
shaker-level ?shaker2 ?level0 | shaker-level ?shaker2 ?level1, shaker-level ?shaker2 ?level2
ontable ?shaker2 | holding ?hand2 ?shaker2, holding ?hand0 ?shaker2, holding ?hand1 ?shaker2
unshaked ?shaker2 |
shaker-level ?shaker2 ?level1 | shaker-level ?shaker2 ?level0, shaker-level ?shaker2 ?level2

holding ?hand0 ?shaker2 | holding ?hand0 ?shaker1, ontable ?shaker2, holding ?hand1 ?shaker2, holding ?hand0 ?shaker0, handempty ?hand0, holding ?hand2 ?shaker2, holding ?hand0 ?shaker3
shaker-level ?shaker2 ?level2 | shaker-level ?shaker2 ?level1, shaker-level ?shaker2 ?level0
holding ?hand2 ?shaker2 | holding ?hand2 ?shaker3, holding ?hand0 ?shaker2, ontable ?shaker2, holding ?hand1 ?shaker2, handempty ?hand2, holding ?hand2 ?shaker1, holding ?hand2 ?shaker0
contains ?shaker2 ?ingredient0 |
contains ?shaker2 ?ingredient1 |
shaker-empty-level ?shaker2 ?level1 |
shaker-empty-level ?shaker2 ?level2 |
shaker-empty-level ?shaker2 ?level0 |
shaker-level ?shaker0 ?level3 | shaker-level ?shaker0 ?level2, shaker-level ?shaker0 ?level1, shaker-level ?shaker0 ?level0
next ?level3 ?level2 |
shaker-empty-level ?shaker0 ?level3 |
next ?level0 ?level3 |
next ?level1 ?level3 |
next ?level3 ?level1 |
shaker-level ?shaker1 ?level3 | shaker-level ?shaker1 ?level2, shaker-level ?shaker1 ?level0, shaker-level ?shaker1 ?level1
next ?level3 ?level0 |
shaker-empty-level ?shaker1 ?level3 |
next ?level2 ?level3 |
holding ?hand2 ?shaker3 | holding ?hand1 ?shaker3, handempty ?hand2, ontable ?shaker3, holding ?hand2 ?shaker1, holding ?hand2 ?shaker2, holding ?hand0 ?shaker3, holding ?hand2 ?shaker0
empty ?shaker3 |
holding ?hand0 ?shaker3 | holding ?hand0 ?shaker1, holding ?hand2 ?shaker3, holding ?hand0 ?shaker2, holding ?hand0 ?shaker0, ontable ?shaker3, handempty ?hand0, holding ?hand1 ?shaker3
empty ?shaker2 |
ontable ?shaker3 | holding ?hand1 ?shaker3, holding ?hand2 ?shaker3, holding ?hand0 ?shaker3
holding ?hand1 ?shaker3 | holding ?hand1 ?shaker0, handempty ?hand1, holding ?hand2 ?shaker3, holding ?hand1 ?shaker2, ontable ?shaker3, holding ?hand1 ?shaker1, holding ?hand0 ?shaker3
contains ?shaker2 ?cocktail1 |
contains ?shaker2 ?cocktail0 |
shaked ?shaker2 |
clean ?shaker2 |
contains ?shaker0 ?cocktail2 |
used ?shot1 ?cocktail2 |
used ?shot0 ?cocktail2 |
contains ?shaker1 ?cocktail2 |
used ?shot2 ?cocktail2 |
cocktail-part2 ?cocktail2 ?ingredient2 |
cocktail-part1 ?cocktail2 ?ingredient2 |
cocktail-part1 ?cocktail2 ?ingredient0 |
cocktail-part2 ?cocktail2 ?ingredient1 |

cocktail-part1 ?cocktail2 ?ingredient1 |
cocktail-part2 ?cocktail2 ?ingredient0 |
dispenses ?dispenser0 ?ingredient3 |
dispenses ?dispenser1 ?ingredient3 |
cocktail-part1 ?cocktail0 ?ingredient3 |
cocktail-part2 ?cocktail0 ?ingredient3 |
contains ?shaker0 ?ingredient3 |
cocktail-part2 ?cocktail1 ?ingredient3 |
contains ?shaker1 ?ingredient3 |
cocktail-part1 ?cocktail1 ?ingredient3 |
dispenses ?dispenser2 ?ingredient1 |
dispenses ?dispenser2 ?ingredient0 |
dispenses ?dispenser2 ?ingredient2 |

C.1.2 Understanding the found macro-operators

macro	I	U	R
action macro-4-actions-0-2-0-7–6-2	1	0	0
action macro-4-actions-0-2-0-7–6-6	1	0	0
action macro-4-actions-0-2-1-10–14-7	1	0	0
action macro-4-actions-0-2-1-9–16-8	1	0	0
action macro-4-actions-0-2-0-7–6-13	1	0	0
action macro-3-actions-0-1-10–28-15	0	1	0
action macro-4-actions-0-1-0-1–30-17	0	1	0
action macro-3-actions-0-1-11–35-20	0	1	0
action macro-2-actions-0-1–44-30	0	1	0
action macro-4-actions-0-5-1-10–51-34	1	0	0
action macro-4-actions-0-5-1-11–52-35	1	0	0
action macro-4-actions-0-10-1-5–59-42	1	0	0
action macro-4-actions-0-10-1-2–71-48	1	0	0
action macro-4-actions-0-9-1-2–85-58	1	0	0
action macro-4-actions-0-9-1-5–86-59	1	0	0
action macro-4-actions-0-9-1-6–87-57	1	0	0
action macro-4-actions-0-9-1-3–85-58	1	0	0
action macro-4-actions-0-9-1-7–87-57	1	0	0
action macro-4-actions-0-1-0-1–100-65	0	1	0
action macro-4-actions-0-1-0-7–121-57	1	0	0
action macro-3-actions-0-1-0–122-65	0	1	0
action macro-4-actions-0-6-0-11–132-77	1	0	0
action macro-4-actions-0-5-0-7–138-82	1	0	0
action macro-4-actions-0-2-1-10–14-84	1	0	0
action macro-4-actions-0-2-7-1–144-85	1	0	0
action macro-4-actions-0-2-7-3–145-86	1	0	0
action macro-3-actions-0-1-0–147-17	0	1	0
action macro-4-actions-0-3-0-7–6-2	1	0	0
action macro-4-actions-0-6-0-7–153-90	1	0	0
action macro-4-actions-0-5-0-7–138-92	1	0	0

Table C.1 continued from previous page

macro	I	U	R
action macro-4-actions-0-5-1-9–159-93	1	0	0
action macro-4-actions-0-2-6-5–1-1	0	0	0
action macro-4-actions-0-2-0-8–9-3	0	0	0
action macro-4-actions-0-2-0-1–11-4	0	0	1
action macro-4-actions-0-2-0-11–8-5	0	0	0
action macro-3-actions-0-2-0–12-4	0	0	0
action macro-4-actions-0-2-1-0–13-4	0	0	0
action macro-3-actions-0-2-1–17-9	0	0	0
action macro-4-actions-0-2-7-5–1-1	0	0	0
action macro-4-actions-0-2-7-1–18-1	0	0	0
action macro-4-actions-0-5-2-1–19-10	0	0	0
action macro-4-actions-0-5-1-0–20-11	0	0	0
action macro-3-actions-0-5-1–21-12	0	0	0
action macro-4-actions-0-2-0-7–6-1	0	0	1
action macro-4-actions-0-2-7-0–23-1	0	0	0
action macro-4-actions-0-2-7-3–24-1	0	0	0
action macro-4-actions-0-1-10-11–27-14	0	0	0
action macro-4-actions-0-1-0-2–29-16	0	0	1
action macro-4-actions-0-1-10-8–31-18	0	0	0
action macro-4-actions-0-1-10-1–32-15	0	0	0
action macro-4-actions-0-1-11-8–33-19	0	0	0
action macro-4-actions-0-1-11-1–34-20	0	0	0
action macro-4-actions-0-1-9-0–36-21	0	0	0
action macro-4-actions-0-1-9-1–37-22	0	0	0
action macro-4-actions-0-1-0-5–38-23	0	0	0
action macro-4-actions-0-1-0-10–39-24	0	0	1
action macro-4-actions-0-1-0-11–40-25	0	0	1
action macro-4-actions-0-1-0-8–41-26	0	0	0
action macro-4-actions-0-1-0-9–42-27	0	0	1
action macro-4-actions-0-1-0-11–40-28	0	0	1
action macro-4-actions-0-1-0-6–43-29	0	0	0
action macro-4-actions-0-1-0-7–43-29	0	0	0
action macro-4-actions-0-5-2-0–45-31	0	0	0
action macro-4-actions-0-5-2-7–46-32	0	0	0
action macro-4-actions-0-5-0-11–49-33	0	0	0
action macro-3-actions-0-5-0–50-11	0	0	0
action macro-4-actions-0-7-1-10–53-36	0	0	0
action macro-3-actions-0-7-1–54-37	0	0	0
action macro-4-actions-0-10-11-8–55-38	0	0	0
action macro-3-actions-0-10-11–56-39	0	0	0
action macro-4-actions-0-10-8-9–57-40	0	0	0
action macro-4-actions-0-10-1-0–58-41	0	0	1
action macro-3-actions-0-10-1–60-41	0	0	0
action macro-4-actions-0-10-11-1–61-39	0	0	0
action macro-2-actions-0-10–63-41	0	0	0
action macro-4-actions-0-11-8-9–64-43	0	0	0
action macro-4-actions-0-11-8-1–65-44	0	0	0
action macro-4-actions-0-11-1-2–66-45	0	0	0

Table C.1 continued from previous page

macro	I	U	R
action macro-2-actions-0-11–67-46	0	0	0
action macro-4-actions-0-10-0-1–69-47	0	0	1
action macro-4-actions-0-10-8-1–70-40	0	0	0
action macro-4-actions-0-11-1-5–72-49	0	0	0
action macro-4-actions-0-11-1-0–73-50	0	0	1
action macro-3-actions-0-11-1–74-46	0	0	0
action macro-4-actions-0-8-9-0–75-51	0	0	0
action macro-4-actions-0-8-9-1–76-52	0	0	0
action macro-4-actions-0-8-1-2–77-53	0	0	0
action macro-4-actions-0-8-1-5–78-54	0	0	0
action macro-4-actions-0-8-1-0–79-52	0	0	1
action macro-3-actions-0-8-1–80-55	0	0	0
action macro-4-actions-0-9-0-1–83-56	0	0	1
action macro-4-actions-0-9-0-6–84-57	0	0	0
action macro-4-actions-0-9-1-0–88-60	0	0	1
action macro-4-actions-0-9-1-0–89-56	0	0	0
action macro-3-actions-0-9-1–90-60	0	0	0
action macro-4-actions-0-9-0-7–84-57	0	0	0
action macro-4-actions-0-1-2-1–92-16	0	0	0
action macro-4-actions-0-1-2-7–93-61	0	0	0
action macro-4-actions-0-1-5-2–94-62	0	0	0
action macro-4-actions-0-1-5-1–95-23	0	0	0
action macro-4-actions-0-1-5-5–97-63	0	0	0
action macro-4-actions-0-1-0-8–98-64	0	0	0
action macro-4-actions-0-1-0-9–99-60	0	0	0
action macro-4-actions-0-1-0-11–101-66	0	0	1
action macro-4-actions-0-1-0-10–102-67	0	0	1
action macro-4-actions-0-1-2-6–93-61	0	0	0
action macro-4-actions-0-7-1-9–103-37	0	0	0
action macro-4-actions-0-7-1-0–104-37	0	0	0
action macro-4-actions-0-8-1-0–107-68	0	0	0
action macro-4-actions-0-7-1-0–108-69	0	0	0
action macro-4-actions-0-6-1-0–111-70	0	0	0
action macro-4-actions-0-6-1-0–111-71	0	0	0
action macro-3-actions-0-6-1–112-70	0	0	0
action macro-4-actions-0-2-6-1–18-1	0	0	0
action macro-4-actions-0-8-9-10–118-72	0	0	0
action macro-4-actions-0-9-10-11–119-73	0	0	1
action macro-4-actions-0-9-10-1–120-67	0	0	1
action macro-4-actions-0-1-0-7–43-74	0	0	0
action macro-4-actions-0-1-0-6–43-74	0	0	0
action macro-4-actions-0-1-0-3–29-16	0	0	0
action macro-4-actions-0-11-9-1–123-50	0	0	0
action macro-4-actions-0-11-9-0–124-75	0	0	0
action macro-4-actions-0-10-9-1–125-41	0	0	0
action macro-4-actions-0-10-9-0–126-47	0	0	0
action macro-4-actions-0-10-1-0–127-47	0	0	0
action macro-4-actions-0-10-11-9–128-39	0	0	0

Table C.1 continued from previous page

macro	I	U	R
action macro-4-actions-0-11-1-0–129-76	0	0	0
action macro-4-actions-0-2-6-0–23-1	0	0	0
action macro-4-actions-0-2-6-3–24-1	0	0	0
action macro-3-actions-0-6-0–133-78	0	0	0
action macro-4-actions-0-6-5-0–134-78	0	0	0
action macro-4-actions-0-6-5-1–135-78	0	0	0
action macro-4-actions-0-6-5-2–136-79	0	0	0
action macro-3-actions-0-3-0–12-4	0	0	0
action macro-3-actions-0-3-1–17-9	0	0	0
action macro-4-actions-0-1-10-11–27-80	0	0	0
action macro-4-actions-0-5-0-11–49-81	0	0	0
action macro-4-actions-0-7-3-0–139-83	0	0	0
action macro-4-actions-0-2-1-0–142-9	0	0	0
action macro-4-actions-0-7-1-0–149-78	0	0	0
action macro-4-actions-0-7-3-1–150-83	0	0	0
action macro-4-actions-0-7-5-1–135-78	0	0	0
action macro-4-actions-0-7-5-0–134-78	0	0	0
action macro-4-actions-0-2-6-5–151-87	0	0	1
action macro-4-actions-0-6-0-8–148-88	0	0	0
action macro-4-actions-0-6-1-10–154-89	0	0	0
action macro-4-actions-0-3-0-8–9-3	0	0	0
action macro-4-actions-0-3-7-0–23-1	0	0	0
action macro-4-actions-0-3-7-5–1-1	0	0	0
action macro-4-actions-0-5-0-8–157-91	0	0	0
action macro-4-actions-0-5-2-6–46-32	0	0	0
action macro-4-actions-0-5-2-6–160-1	0	0	0

Table C.1: Detail of the found macro-operators for `barman` domain.

C.2 Blocksworld

C.2.1 Predicate incompatibilities

clear ?block1 | on ?block0 ?block1, on ?block2 ?block1, holding ?block1
on ?block1 ?block2 | on ?block0 ?block2, clear ?block2, on ?block1 ?block0, holding ?block1, ontable ?block1
clear ?block0 | on ?block2 ?block0, on ?block1 ?block0, holding ?block0
on ?block0 ?block2 | on ?block1 ?block2, ontable ?block0, clear ?block2, holding ?block0, on ?block0 ?block1
on ?block2 ?block1 | holding ?block2, clear ?block1, on ?block2 ?block0, ontable ?block2, on ?block0 ?block1

macro	I	U	R
action macro-3-actions-3-1-0–1-1	0	1	0
action macro-2-actions-3-1–3-1	1	0	0
action macro-2-actions-1-2–4-1	1	0	0
action macro-2-actions-1-0–2-2	0	1	0
action macro-2-actions-2-3–5-1	0	1	0
action macro-3-actions-1-0-2–6-1	0	1	0
action macro-2-actions-0-1–2-2	0	1	0
action macro-3-actions-3-1-0–8-1	1	0	0
action macro-2-actions-3-1–2-1	0	0	0
action macro-2-actions-0-2–7-1	0	0	0

Table C.2: Detail of the found macro-operators for `blocksworld` domain.

clear ?block2 | holding ?block2, on ?block1 ?block2, on ?block0 ?block2
on ?block2 ?block0 | holding ?block2, clear ?block0, on ?block2 ?block1, ontable ?block2, on ?block1 ?block0
on ?block1 ?block0 | on ?block1 ?block2, clear ?block0, on ?block2 ?block0, holding ?block1, ontable ?block1
handempty | holding ?block2, holding ?block0, holding ?block1
on ?block0 ?block1 | clear ?block1, on ?block0 ?block2, on ?block2 ?block1, ontable ?block0, holding ?block0
holding ?block2 | on ?block2 ?block1, clear ?block2, on ?block2 ?block0, ontable ?block2, holding ?block0, holding ?block1, handempty
holding ?block0 | holding ?block2, clear ?block0, on ?block0 ?block2, ontable ?block0, holding ?block1, handempty, on ?block0 ?block1
holding ?block1 | clear ?block1, on ?block1 ?block2, holding ?block2, holding ?block0, on ?block1 ?block0, ontable ?block1, handempty
ontable ?block0 | holding ?block0
ontable ?block2 | holding ?block2
ontable ?block1 | holding ?block1

C.2.2 Understanding the found macro-operators

C.3 Depots

C.3.1 Predicate incompatibilities

at ?hoist2 ?distributor0 |
on ?crate0 ?crate2 | on ?crate2 ?pallet1, on ?crate0 ?pallet1, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, in ?crate0 ?truck0, on ?crate2 ?pallet0, on ?crate2 ?crate1, clear ?crate2, on ?crate1 ?crate2, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, on ?crate0 ?crate4, in ?crate2 ?truck0, lifting ?hoist1 ?crate2, lifting ?hoist1 ?crate0, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate1 ?crate0, on ?crate4 ?crate2, on ?crate0 ?pallet0, on ?crate0 ?pallet2, in ?crate0 ?truck1, in ?crate2 ?truck1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, on ?crate0 ?crate3
at ?truck0 ?distributor2 | at ?truck0 ?depot4, at ?truck0 ?distributor1, at ?truck0 ?depot2, at ?truck0 ?distributor0, at ?truck0 ?depot0, at ?truck0 ?depot1, at ?truck0 ?distributor4, at ?truck0 ?depot3, at ?truck0 ?distributor3
at ?crate2 ?distributor2 | lifting ?hoist1 ?crate2, at ?crate2 ?distributor0, at ?crate2 ?depot0, at ?crate2 ?depot1, at ?crate2 ?distributor1, lifting ?hoist2 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot3, at ?crate2 ?depot2, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0
at ?crate1 ?depot2 | at ?crate1 ?depot1, in ?crate1 ?truck1, at ?crate1 ?depot3, lifting ?hoist0 ?crate1, at ?crate1 ?distributor3, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, at ?crate1 ?depot0, at ?crate1 ?distributor2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, at ?crate1 ?distributor0, at ?crate1 ?distributor1
at ?crate2 ?distributor0 | lifting ?hoist1 ?crate2, at ?crate2 ?distributor2, at ?crate2 ?depot0, at ?crate2 ?depot1, at ?crate2 ?distributor1, lifting ?hoist2 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot3, at ?crate2 ?depot2, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0
on ?crate2 ?pallet1 | on ?crate0 ?crate2, on ?crate0 ?pallet1, clear ?pallet1, on ?crate2 ?crate0, on ?crate2 ?pallet0, on ?crate2 ?crate1, on ?crate1 ?crate2, on ?crate2 ?crate3, on ?crate3 ?crate2, on ?crate1 ?pallet1, lifting ?hoist0 ?crate2, on ?crate4 ?pallet1, in ?crate2 ?truck0, lifting ?hoist1 ?crate2, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate2 ?pallet2, on ?crate4 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, on ?crate3 ?pallet1
on ?crate0 ?pallet1 | on ?crate0 ?crate2, on ?crate2 ?pallet1, clear ?pallet1, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, in ?crate0 ?truck0, on ?crate1 ?pallet1, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist1 ?crate0, on ?crate1 ?crate0, on ?crate0 ?pallet0, on ?crate0 ?pallet2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate4 ?crate0, on ?crate0 ?crate3
available ?hoist1 | lifting ?hoist1 ?crate2, lifting ?hoist1 ?crate0, lifting ?hoist1 ?crate4, lifting ?hoist1 ?crate1, lifting ?hoist1 ?crate3
at ?hoist2 ?depot0 |

at ?crate2 ?depot1 | lifting ?hoist1 ?crate2, at ?crate2 ?distributor2, at ?crate2 ?distributor0, at ?crate2 ?depot0, at ?crate2 ?distributor1, lifting ?hoist2 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot3, at ?crate2 ?depot2, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0
clear ?crate0 | lifting ?hoist1 ?crate0, clear ?pallet1, clear ?crate3, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, clear ?crate4, in ?crate0 ?truck0, on ?crate1 ?crate0, clear ?crate2, in ?crate0 ?truck1, clear ?pallet0, clear ?crate1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, clear ?pallet2
on ?crate0 ?crate1 | on ?crate0 ?crate2, on ?crate0 ?pallet1, lifting ?hoist0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, in ?crate0 ?truck0, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, clear ?crate1, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, on ?crate0 ?crate4, on ?crate1 ?pallet0, in ?crate1 ?truck1, lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate1 ?crate3, on ?crate0 ?pallet0, on ?crate0 ?pallet2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, on ?crate0 ?crate3
available ?hoist0 | lifting ?hoist0 ?crate3, lifting ?hoist0 ?crate0, lifting ?hoist0 ?crate1, lifting ?hoist0 ?crate2, lifting ?hoist0 ?crate4
on ?crate2 ?crate0 | on ?crate0 ?crate2, on ?crate2 ?pallet1, on ?crate0 ?pallet1, clear ?crate0, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck0, on ?crate2 ?pallet0, on ?crate2 ?crate1, on ?crate1 ?crate2, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, on ?crate0 ?crate4, in ?crate2 ?truck0, lifting ?hoist1 ?crate2, lifting ?hoist1 ?crate0, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate2 ?pallet2, on ?crate1 ?crate0, on ?crate4 ?crate2, in ?crate2 ?truck2, on ?crate0 ?pallet0, in ?crate0 ?truck1, in ?crate2 ?truck1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, on ?crate0 ?crate3
at ?truck1 ?depot0 | at ?truck1 ?distributor0, at ?truck1 ?distributor3, at ?truck1 ?depot2, at ?truck1 ?depot3, at ?truck1 ?distributor1, at ?truck1 ?distributor4, at ?truck1 ?distributor2, at ?truck1 ?depot4, at ?truck1 ?depot1
at ?crate0 ?depot0 | lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0, at ?crate0 ?distributor0, at ?crate0 ?distributor3, at ?crate0 ?depot2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, at ?crate0 ?depot1, at ?crate0 ?depot3, at ?crate0 ?distributor2, at ?crate0 ?distributor1
at ?truck1 ?depot1 | at ?truck1 ?distributor0, at ?truck1 ?distributor3, at ?truck1 ?depot2, at ?truck1 ?depot3, at ?truck1 ?distributor1, at ?truck1 ?distributor4, at ?truck1 ?distributor2, at ?truck1 ?depot0, at ?truck1 ?depot4
on ?crate2 ?pallet0 | on ?crate0 ?crate2, on ?crate2 ?pallet1, on ?crate2 ?crate0, on ?crate3 ?pallet0, on ?crate2 ?crate1, on ?crate1 ?crate2, clear ?pallet0, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, in ?crate2 ?truck0, on ?crate1 ?pallet0, lifting ?hoist1 ?crate2, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate2 ?pallet2, on ?crate4 ?crate2, in ?crate2 ?truck2, on ?crate0 ?pallet0, in ?crate2 ?truck1, on ?crate4 ?pallet0
at ?hoist0 ?distributor1 |
on ?crate2 ?crate1 | on ?crate0 ?crate2, on ?crate2 ?pallet1, lifting ?hoist0 ?crate1, on ?crate0 ?crate1, on ?crate2 ?crate0, on ?crate1 ?crate4, on ?crate2 ?pallet0, on ?crate3 ?crate1, on ?crate1 ?crate2, clear ?crate1, on ?crate2 ?crate3, in ?crate1 ?truck0, on ?crate3 ?crate2, on ?crate1 ?pallet1, lifting ?hoist0 ?crate2, lifting ?hoist1 ?crate1, in ?crate2 ?truck0, on ?crate1 ?pallet0, lifting ?hoist1 ?crate2, in ?crate1 ?truck1, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, lifting ?hoist2 ?crate1, on ?crate2 ?pallet2, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate4 ?crate2, in ?crate2 ?truck2, on ?crate1 ?crate3, in ?crate2 ?truck1

at ?truck0 ?distributor1 | at ?truck0 ?depot4, at ?truck0 ?distributor2, at ?truck0 ?depot2, at ?truck0 ?distributor0, at ?truck0 ?depot0, at ?truck0 ?depot1, at ?truck0 ?distributor4, at ?truck0 ?depot3, at ?truck0 ?distributor3
at ?truck0 ?depot2 | at ?truck0 ?depot4, at ?truck0 ?distributor1, at ?truck0 ?distributor2, at ?truck0 ?depot0, at ?truck0 ?distributor0, at ?truck0 ?depot1, at ?truck0 ?distributor4, at ?truck0 ?depot3, at ?truck0 ?distributor3
at ?truck0 ?distributor0 | at ?truck0 ?depot4, at ?truck0 ?distributor1, at ?truck0 ?distributor2, at ?truck0 ?depot2, at ?truck0 ?depot0, at ?truck0 ?depot1, at ?truck0 ?distributor4, at ?truck0 ?depot3, at ?truck0 ?distributor3
at ?crate0 ?depot2 | lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0, at ?crate0 ?distributor0, at ?crate0 ?distributor3, at ?crate0 ?depot0, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, at ?crate0 ?depot1, at ?crate0 ?depot3, at ?crate0 ?distributor2, at ?crate0 ?distributor1
at ?crate1 ?distributor1 | at ?crate1 ?depot1, in ?crate1 ?truck1, at ?crate1 ?depot2, at ?crate1 ?depot3, lifting ?hoist0 ?crate1, at ?crate1 ?distributor3, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, at ?crate1 ?depot0, at ?crate1 ?distributor2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, at ?crate1 ?distributor0
at ?hoist0 ?distributor0 |
clear ?crate1 | in ?crate1 ?truck1, clear ?pallet1, clear ?crate0, on ?crate0 ?crate1, lifting ?hoist0 ?crate1, clear ?crate3, clear ?crate4, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, on ?crate4 ?crate1, on ?crate2 ?crate1, on ?crate3 ?crate1, clear ?crate2, clear ?pallet0, clear ?pallet2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1
on ?crate1 ?crate2 | on ?crate0 ?crate2, on ?crate2 ?pallet1, lifting ?hoist0 ?crate1, on ?crate0 ?crate1, on ?crate2 ?crate0, in ?crate1 ?truck2, on ?crate1 ?crate4, on ?crate2 ?pallet0, on ?crate2 ?crate1, on ?crate3 ?crate1, clear ?crate2, on ?crate1 ?pallet2, on ?crate2 ?crate3, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, lifting ?hoist0 ?crate2, on ?crate3 ?crate2, in ?crate2 ?truck0, on ?crate1 ?pallet0, lifting ?hoist1 ?crate2, in ?crate1 ?truck1, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate4 ?crate2, on ?crate1 ?crate3, in ?crate2 ?truck1
clear ?crate2 | lifting ?hoist1 ?crate2, on ?crate0 ?crate2, clear ?pallet1, clear ?crate0, clear ?crate3, lifting ?hoist2 ?crate2, clear ?crate4, on ?crate4 ?crate2, in ?crate2 ?truck2, on ?crate1 ?crate2, in ?crate2 ?truck1, clear ?pallet0, clear ?crate1, clear ?pallet2, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, in ?crate2 ?truck0
at ?crate0 ?depot1 | lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0, at ?crate0 ?distributor0, at ?crate0 ?distributor3, at ?crate0 ?depot0, at ?crate0 ?depot2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, at ?crate0 ?depot3, at ?crate0 ?distributor2, at ?crate0 ?distributor1
at ?truck1 ?distributor2 | at ?truck1 ?distributor0, at ?truck1 ?distributor3, at ?truck1 ?depot2, at ?truck1 ?depot3, at ?truck1 ?distributor1, at ?truck1 ?distributor4, at ?truck1 ?depot0, at ?truck1 ?depot4, at ?truck1 ?depot1
at ?hoist0 ?depot0 |
on ?crate1 ?pallet1 | on ?crate2 ?pallet1, on ?crate0 ?pallet1, clear ?pallet1, lifting ?hoist0 ?crate1, on ?crate0 ?crate1, in ?crate1 ?truck2, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, on ?crate1 ?pallet2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, on ?crate4 ?pallet1, on ?crate1 ?pallet0, in ?crate1 ?truck1, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate1 ?crate3, on ?crate3 ?pallet1
at ?hoist0 ?depot2 |
at ?hoist2 ?distributor2 |
at ?hoist1 ?depot0 |

at ?crate0 ?distributor1 | lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0, at ?crate0 ?distributor0, at ?crate0 ?distributor3, at ?crate0 ?depot0, at ?crate0 ?depot2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, at ?crate0 ?depot1, at ?crate0 ?depot3, at ?crate0 ?distributor2
on ?crate1 ?pallet0 | lifting ?hoist0 ?crate1, on ?crate0 ?crate1, in ?crate1 ?truck2, on ?crate3 ?pallet0, on ?crate1 ?crate4, on ?crate2 ?pallet0, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, clear ?pallet0, on ?crate1 ?pallet2, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, in ?crate1 ?truck1, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate1 ?crate3, on ?crate0 ?pallet0, on ?crate4 ?pallet0
at ?truck1 ?distributor0 | at ?truck1 ?distributor3, at ?truck1 ?depot2, at ?truck1 ?depot3, at ?truck1 ?distributor1, at ?truck1 ?distributor4, at ?truck1 ?distributor2, at ?truck1 ?depot0, at ?truck1 ?depot4, at ?truck1 ?depot1
at ?crate1 ?depot1 | in ?crate1 ?truck1, at ?crate1 ?depot2, at ?crate1 ?depot3, lifting ?hoist0 ?crate1, at ?crate1 ?distributor3, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, at ?crate1 ?depot0, at ?crate1 ?distributor2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, at ?crate1 ?distributor0, at ?crate1 ?distributor1
at ?truck0 ?depot0 | at ?truck0 ?depot4, at ?truck0 ?distributor1, at ?truck0 ?depot2, at ?truck0 ?distributor2, at ?truck0 ?distributor0, at ?truck0 ?depot1, at ?truck0 ?distributor4, at ?truck0 ?depot3, at ?truck0 ?distributor3
at ?hoist0 ?depot1 |
at ?crate2 ?depot0 | lifting ?hoist1 ?crate2, at ?crate2 ?distributor2, at ?crate2 ?distributor0, at ?crate2 ?depot1, at ?crate2 ?distributor1, lifting ?hoist2 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot3, at ?crate2 ?depot2, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0
at ?truck0 ?depot1 | at ?truck0 ?depot4, at ?truck0 ?distributor1, at ?truck0 ?depot2, at ?truck0 ?depot0, at ?truck0 ?distributor2, at ?truck0 ?distributor0, at ?truck0 ?distributor4, at ?truck0 ?depot3, at ?truck0 ?distributor3
at ?crate2 ?distributor1 | lifting ?hoist1 ?crate2, at ?crate2 ?distributor2, at ?crate2 ?distributor0, at ?crate2 ?depot0, at ?crate2 ?depot1, lifting ?hoist2 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot3, at ?crate2 ?depot2, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0
at ?hoist1 ?distributor0 |
at ?truck1 ?depot2 | at ?truck1 ?distributor0, at ?truck1 ?distributor3, at ?truck1 ?depot3, at ?truck1 ?distributor1, at ?truck1 ?distributor4, at ?truck1 ?distributor2, at ?truck1 ?depot0, at ?truck1 ?depot4, at ?truck1 ?depot1
available ?hoist2 | lifting ?hoist2 ?crate0, lifting ?hoist2 ?crate2, lifting ?hoist2 ?crate1, lifting ?hoist2 ?crate3, lifting ?hoist2 ?crate4
at ?hoist1 ?depot2 |
at ?crate0 ?distributor0 | lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0, at ?crate0 ?distributor3, at ?crate0 ?depot0, at ?crate0 ?depot2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, at ?crate0 ?depot1, at ?crate0 ?depot3, at ?crate0 ?distributor2, at ?crate0 ?distributor1
on ?crate1 ?crate0 | on ?crate0 ?crate2, on ?crate0 ?pallet1, lifting ?hoist0 ?crate1, clear ?crate0, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, on ?crate2 ?crate0, in ?crate1 ?truck2, in ?crate0 ?truck0, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, on ?crate1 ?pallet2, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, on ?crate0 ?crate4, on ?crate1 ?pallet0, in ?crate1 ?truck1, lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate1, on ?crate4 ?crate1, on ?crate1 ?crate3, on ?crate0 ?pallet0, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, on ?crate0 ?crate3

at ?hoist2 ?distributor1 |
on ?crate0 ?pallet0 | on ?crate0 ?crate2, on ?crate0 ?pallet1, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, in ?crate0 ?truck0, on ?crate3 ?pallet0, on ?crate2 ?pallet0, clear ?pallet0, on ?crate0 ?crate4, on ?crate1 ?pallet0, lifting ?hoist1 ?crate0, on ?crate1 ?crate0, on ?crate0 ?pallet2, in ?crate0 ?truck1, on ?crate4 ?pallet0, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, on ?crate0 ?crate3
at ?hoist2 ?depot1 |
at ?hoist1 ?distributor1 |
at ?hoist1 ?distributor2 |
at ?crate1 ?depot0 | at ?crate1 ?depot1, in ?crate1 ?truck1, at ?crate1 ?depot2, at ?crate1 ?depot3, lifting ?hoist0 ?crate1, at ?crate1 ?distributor3, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, at ?crate1 ?distributor2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, at ?crate1 ?distributor0, at ?crate1 ?distributor1
at ?crate1 ?distributor2 | at ?crate1 ?depot1, in ?crate1 ?truck1, at ?crate1 ?depot2, at ?crate1 ?depot3, lifting ?hoist0 ?crate1, at ?crate1 ?distributor3, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, at ?crate1 ?depot0, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, at ?crate1 ?distributor0, at ?crate1 ?distributor1
at ?truck1 ?distributor1 | at ?truck1 ?distributor0, at ?truck1 ?distributor3, at ?truck1 ?depot2, at ?truck1 ?depot3, at ?truck1 ?distributor4, at ?truck1 ?distributor2, at ?truck1 ?depot0, at ?truck1 ?depot4, at ?truck1 ?depot1
at ?crate2 ?depot2 | lifting ?hoist1 ?crate2, at ?crate2 ?distributor2, at ?crate2 ?distributor0, at ?crate2 ?depot0, at ?crate2 ?depot1, at ?crate2 ?distributor1, lifting ?hoist2 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot3, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0
at ?crate0 ?distributor2 | lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0, at ?crate0 ?distributor0, at ?crate0 ?distributor3, at ?crate0 ?depot0, at ?crate0 ?depot2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, at ?crate0 ?depot1, at ?crate0 ?depot3, at ?crate0 ?distributor1
at ?hoist0 ?distributor2 |
at ?hoist1 ?depot1 |
at ?crate1 ?distributor0 | at ?crate1 ?depot1, in ?crate1 ?truck1, at ?crate1 ?depot2, at ?crate1 ?depot3, lifting ?hoist0 ?crate1, at ?crate1 ?distributor3, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, at ?crate1 ?depot0, at ?crate1 ?distributor2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, at ?crate1 ?distributor1
at ?hoist2 ?depot2 |
at ?truck2 ?depot1 | at ?truck2 ?distributor4, at ?truck2 ?distributor3, at ?truck2 ?depot0, at ?truck2 ?distributor2, at ?truck2 ?depot4, at ?truck2 ?depot3, at ?truck2 ?distributor1, at ?truck2 ?distributor0, at ?truck2 ?depot2
at ?truck2 ?distributor0 | at ?truck2 ?distributor4, at ?truck2 ?depot1, at ?truck2 ?distributor3, at ?truck2 ?depot0, at ?truck2 ?distributor2, at ?truck2 ?depot4, at ?truck2 ?depot3, at ?truck2 ?distributor1, at ?truck2 ?depot2
at ?truck2 ?depot0 | at ?truck2 ?distributor4, at ?truck2 ?depot1, at ?truck2 ?distributor3, at ?truck2 ?distributor2, at ?truck2 ?depot4, at ?truck2 ?depot3, at ?truck2 ?distributor1, at ?truck2 ?distributor0, at ?truck2 ?depot2
at ?truck2 ?distributor1 | at ?truck2 ?distributor4, at ?truck2 ?depot1, at ?truck2 ?distributor3, at ?truck2 ?depot0, at ?truck2 ?distributor2, at ?truck2 ?depot4, at ?truck2 ?depot3, at ?truck2 ?distributor0, at ?truck2 ?depot2
at ?truck2 ?distributor2 | at ?truck2 ?distributor4, at ?truck2 ?depot1, at ?truck2 ?distributor3, at ?truck2 ?depot0, at ?truck2 ?depot4, at ?truck2 ?depot3, at ?truck2 ?distributor1, at ?truck2 ?distributor0, at ?truck2 ?depot2

at ?truck2 ?depot2 | at ?truck2 ?distributor4, at ?truck2 ?depot1, at ?truck2 ?distributor3, at ?truck2 ?depot0, at ?truck2 ?distributor2, at ?truck2 ?depot4, at ?truck2 ?depot3, at ?truck2 ?distributor1, at ?truck2 ?distributor0
in ?crate3 ?truck1 | lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, lifting ?hoist2 ?crate3, on ?crate3 ?crate1, at ?crate3 ?distributor0, at ?crate3 ?depot1, on ?crate2 ?crate3, on ?crate3 ?crate2, at ?crate3 ?distributor1, on ?crate4 ?crate3, in ?crate3 ?truck0, clear ?crate3, on ?crate3 ?crate4, at ?crate3 ?distributor2, at ?crate3 ?depot2, on ?crate1 ?crate3, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate3 ?crate0, at ?crate3 ?depot0, on ?crate0 ?crate3
at ?crate3 ?depot2 | at ?crate3 ?depot3, in ?crate3 ?truck0, in ?crate3 ?truck2, at ?crate3 ?distributor2, lifting ?hoist1 ?crate3, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, at ?crate3 ?distributor0, at ?crate3 ?distributor3, at ?crate3 ?depot1, lifting ?hoist0 ?crate3, at ?crate3 ?depot0, at ?crate3 ?distributor1
on ?crate2 ?crate3 | on ?crate0 ?crate2, on ?crate2 ?pallet1, on ?crate2 ?crate0, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate2 ?pallet0, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, in ?crate2 ?truck0, lifting ?hoist1 ?crate2, on ?crate4 ?crate3, in ?crate3 ?truck0, on ?crate2 ?crate4, clear ?crate3, lifting ?hoist2 ?crate2, on ?crate3 ?crate4, on ?crate2 ?pallet2, on ?crate4 ?crate2, on ?crate1 ?crate3, in ?crate2 ?truck2, in ?crate2 ?truck1, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate0 ?crate3
in ?crate1 ?truck0 | at ?crate1 ?depot2, lifting ?hoist0 ?crate1, on ?crate0 ?crate1, in ?crate1 ?truck2, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, clear ?crate1, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, at ?crate1 ?depot1, on ?crate1 ?pallet0, in ?crate1 ?truck1, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate1 ?crate3, in ?crate1 ?truck3, at ?crate1 ?depot0, at ?crate1 ?distributor2, at ?crate1 ?distributor0, at ?crate1 ?distributor1
on ?crate3 ?crate2 | on ?crate0 ?crate2, on ?crate2 ?pallet1, on ?crate2 ?crate0, on ?crate3 ?pallet2, in ?crate3 ?truck2, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate2 ?pallet0, on ?crate3 ?crate1, on ?crate2 ?crate1, clear ?crate2, on ?crate1 ?crate2, on ?crate2 ?crate3, lifting ?hoist0 ?crate2, in ?crate2 ?truck0, lifting ?hoist1 ?crate2, on ?crate4 ?crate3, in ?crate3 ?truck0, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate3 ?crate4, on ?crate4 ?crate2, on ?crate1 ?crate3, lifting ?hoist0 ?crate3, in ?crate2 ?truck1, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate0 ?crate3
at ?crate3 ?distributor1 | at ?crate3 ?depot3, in ?crate3 ?truck0, in ?crate3 ?truck2, at ?crate3 ?distributor2, lifting ?hoist1 ?crate3, at ?crate3 ?depot2, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, at ?crate3 ?distributor0, at ?crate3 ?distributor3, at ?crate3 ?depot1, lifting ?hoist0 ?crate3, at ?crate3 ?depot0
in ?crate1 ?truck1 | at ?crate1 ?depot2, lifting ?hoist0 ?crate1, on ?crate0 ?crate1, in ?crate1 ?truck2, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, clear ?crate1, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, at ?crate1 ?depot1, on ?crate1 ?pallet0, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate1 ?crate3, in ?crate1 ?truck3, at ?crate1 ?depot0, at ?crate1 ?distributor2, at ?crate1 ?distributor0, at ?crate1 ?distributor1
clear ?crate3 | on ?crate4 ?crate3, in ?crate3 ?truck0, clear ?pallet1, clear ?crate0, clear ?crate4, in ?crate3 ?truck2, lifting ?hoist1 ?crate3, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate1 ?crate3, clear ?crate2, lifting ?hoist0 ?crate3, clear ?pallet0, clear ?crate1, clear ?pallet2, on ?crate2 ?crate3, on ?crate0 ?crate3
on ?crate3 ?crate0 | on ?crate0 ?crate2, on ?crate0 ?pallet1, clear ?crate0, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, on ?crate2 ?crate0, in ?crate0 ?truck0, on ?crate3 ?pal-

let2, in ?crate3 ?truck2, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate3 ?crate1, on ?crate2 ?crate3, on ?crate3 ?crate2, on ?crate0 ?crate4, on ?crate4 ?crate3, in ?crate3 ?truck0, lifting ?hoist1 ?crate0, on ?crate3 ?crate4, on ?crate1 ?crate0, on ?crate1 ?crate3, on ?crate0 ?pallet0, in ?crate0 ?truck1, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, lifting ?hoist0 ?crate0, on ?crate4 ?crate0, on ?crate0 ?crate3
at ?crate3 ?depot0 | at ?crate3 ?depot3, in ?crate3 ?truck0, in ?crate3 ?truck2, at ?crate3 ?distributor2, lifting ?hoist1 ?crate3, at ?crate3 ?depot2, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, at ?crate3 ?distributor0, at ?crate3 ?distributor3, at ?crate3 ?depot1, lifting ?hoist0 ?crate3, at ?crate3 ?distributor1
on ?crate3 ?pallet0 | on ?crate3 ?pallet2, in ?crate3 ?truck2, lifting ?hoist1 ?crate3, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate2 ?pallet0, on ?crate3 ?crate1, clear ?pallet0, on ?crate2 ?crate3, on ?crate3 ?crate2, on ?crate1 ?pallet0, on ?crate4 ?crate3, in ?crate3 ?truck0, on ?crate3 ?crate4, on ?crate1 ?crate3, on ?crate0 ?pallet0, lifting ?hoist0 ?crate3, on ?crate4 ?pallet0, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate0 ?crate3
at ?crate3 ?distributor0 | at ?crate3 ?depot3, in ?crate3 ?truck0, in ?crate3 ?truck2, at ?crate3 ?distributor2, lifting ?hoist1 ?crate3, at ?crate3 ?depot2, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, at ?crate3 ?distributor3, at ?crate3 ?depot1, lifting ?hoist0 ?crate3, at ?crate3 ?depot0, at ?crate3 ?distributor1
in ?crate2 ?truck0 | at ?crate2 ?distributor2, on ?crate0 ?crate2, at ?crate2 ?distributor0, on ?crate2 ?pallet1, at ?crate2 ?depot1, on ?crate2 ?crate0, on ?crate2 ?pallet0, on ?crate2 ?crate1, clear ?crate2, on ?crate1 ?crate2, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, lifting ?hoist1 ?crate2, at ?crate2 ?depot0, at ?crate2 ?distributor1, in ?crate2 ?truck3, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate4 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot2
at ?crate3 ?distributor2 | at ?crate3 ?depot3, in ?crate3 ?truck0, in ?crate3 ?truck2, lifting ?hoist1 ?crate3, at ?crate3 ?depot2, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, at ?crate3 ?distributor0, at ?crate3 ?distributor3, at ?crate3 ?depot1, lifting ?hoist0 ?crate3, at ?crate3 ?depot0, at ?crate3 ?distributor1
in ?crate2 ?truck1 | at ?crate2 ?distributor2, on ?crate0 ?crate2, at ?crate2 ?distributor0, on ?crate2 ?pallet1, at ?crate2 ?depot1, on ?crate2 ?crate0, on ?crate2 ?pallet0, on ?crate2 ?crate1, clear ?crate2, on ?crate1 ?crate2, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, in ?crate2 ?truck0, lifting ?hoist1 ?crate2, at ?crate2 ?depot0, at ?crate2 ?distributor1, in ?crate2 ?truck3, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate4 ?crate2, in ?crate2 ?truck2, at ?crate2 ?depot2
in ?crate0 ?truck0 | on ?crate0 ?crate2, on ?crate0 ?pallet1, clear ?crate0, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, at ?crate0 ?depot0, at ?crate0 ?depot2, at ?crate0 ?depot1, on ?crate0 ?crate4, at ?crate0 ?distributor1, in ?crate0 ?truck3, lifting ?hoist1 ?crate0, at ?crate0 ?distributor0, on ?crate1 ?crate0, on ?crate0 ?pallet0, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, at ?crate0 ?distributor2, on ?crate0 ?crate3
on ?crate3 ?crate1 | lifting ?hoist0 ?crate1, on ?crate0 ?crate1, on ?crate3 ?pallet2, in ?crate3 ?truck2, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate1 ?crate2, clear ?crate1, on ?crate2 ?crate3, on ?crate3 ?crate2, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, on ?crate1 ?pallet0, in ?crate1 ?truck1, on ?crate4 ?crate3, in ?crate3 ?truck0, on ?crate3 ?crate4, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate1 ?crate3, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate0 ?crate3
at ?crate3 ?depot1 | at ?crate3 ?depot3, in ?crate3 ?truck0, in ?crate3 ?truck2, at ?crate3

?distributor2, lifting ?hoist1 ?crate3, at ?crate3 ?depot2, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, at ?crate3 ?distributor0, at ?crate3 ?distributor3, lifting ?hoist0 ?crate3, at ?crate3 ?depot0, at ?crate3 ?distributor1
in ?crate3 ?truck0 | lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate3 ?crate1, at ?crate3 ?distributor0, at ?crate3 ?depot1, on ?crate2 ?crate3, on ?crate3 ?crate2, at ?crate3 ?distributor1, on ?crate4 ?crate3, clear ?crate3, on ?crate3 ?crate4, at ?crate3 ?distributor2, at ?crate3 ?depot2, on ?crate1 ?crate3, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate3 ?crate0, at ?crate3 ?depot0, on ?crate0 ?crate3
on ?crate1 ?crate3 | lifting ?hoist0 ?crate1, on ?crate0 ?crate1, in ?crate1 ?truck2, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, on ?crate1 ?pallet2, on ?crate2 ?crate3, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, on ?crate3 ?crate2, on ?crate1 ?pallet0, in ?crate1 ?truck1, on ?crate4 ?crate3, in ?crate3 ?truck0, clear ?crate3, on ?crate3 ?crate4, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate0 ?crate3
in ?crate0 ?truck1 | on ?crate0 ?crate2, on ?crate0 ?pallet1, clear ?crate0, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, in ?crate0 ?truck0, at ?crate0 ?depot0, at ?crate0 ?depot2, at ?crate0 ?depot1, on ?crate0 ?crate4, at ?crate0 ?distributor1, in ?crate0 ?truck3, lifting ?hoist1 ?crate0, at ?crate0 ?distributor0, on ?crate1 ?crate0, on ?crate0 ?pallet0, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, at ?crate0 ?distributor2, on ?crate0 ?crate3
on ?crate3 ?pallet1 | on ?crate2 ?pallet1, on ?crate0 ?pallet1, clear ?pallet1, on ?crate3 ?pallet2, in ?crate3 ?truck2, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate3 ?crate1, on ?crate2 ?crate3, on ?crate3 ?crate2, on ?crate1 ?pallet1, on ?crate4 ?pallet1, on ?crate4 ?crate3, in ?crate3 ?truck0, on ?crate3 ?crate4, on ?crate1 ?crate3, lifting ?hoist0 ?crate3, on ?crate3 ?crate0, on ?crate0 ?crate3
on ?crate0 ?crate3 | on ?crate0 ?crate2, on ?crate0 ?pallet1, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, in ?crate0 ?truck0, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate3 ?crate1, on ?crate2 ?crate3, on ?crate3 ?crate2, on ?crate0 ?crate4, on ?crate4 ?crate3, in ?crate3 ?truck0, lifting ?hoist1 ?crate0, clear ?crate3, on ?crate3 ?crate4, on ?crate1 ?crate0, on ?crate1 ?crate3, on ?crate0 ?pallet0, on ?crate0 ?pallet2, in ?crate0 ?truck1, lifting ?hoist0 ?crate3, lifting ?hoist0 ?crate0, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate4 ?crate0
at ?hoist1 ?distributor3 |
at ?crate1 ?depot3 | at ?crate1 ?depot1, in ?crate1 ?truck1, at ?crate1 ?depot2, lifting ?hoist0 ?crate1, at ?crate1 ?distributor3, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, at ?crate1 ?depot0, at ?crate1 ?distributor2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, at ?crate1 ?distributor0, at ?crate1 ?distributor1
at ?crate0 ?distributor3 | lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0, at ?crate0 ?distributor0, at ?crate0 ?depot0, at ?crate0 ?depot2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, at ?crate0 ?depot1, at ?crate0 ?depot3, at ?crate0 ?distributor2, at ?crate0 ?distributor1
at ?hoist1 ?depot3 |
at ?crate2 ?depot3 | lifting ?hoist1 ?crate2, at ?crate2 ?distributor2, at ?crate2 ?distributor0, at ?crate2 ?depot0, at ?crate2 ?depot1, at ?crate2 ?distributor1, lifting ?hoist2 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot2, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0
at ?truck1 ?depot3 | at ?truck1 ?distributor0, at ?truck1 ?distributor3, at ?truck1 ?depot2,

at ?truck1 ?distributor1, at ?truck1 ?distributor4, at ?truck1 ?distributor2, at ?truck1 ?depot0, at ?truck1 ?depot4, at ?truck1 ?depot1
at ?crate0 ?depot3 | lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0, at ?crate0 ?distributor0, at ?crate0 ?distributor3, at ?crate0 ?depot0, at ?crate0 ?depot2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, at ?crate0 ?depot1, at ?crate0 ?distributor2, at ?crate0 ?distributor1
at ?crate2 ?distributor3 | lifting ?hoist1 ?crate2, at ?crate2 ?distributor2, at ?crate2 ?distributor0, at ?crate2 ?depot0, at ?crate2 ?depot1, at ?crate2 ?distributor1, lifting ?hoist2 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot3, at ?crate2 ?depot2, lifting ?hoist0 ?crate2, in ?crate2 ?truck0
at ?hoist0 ?distributor3 |
at ?truck1 ?distributor3 | at ?truck1 ?distributor0, at ?truck1 ?depot3, at ?truck1 ?depot2, at ?truck1 ?distributor1, at ?truck1 ?distributor4, at ?truck1 ?distributor2, at ?truck1 ?depot0, at ?truck1 ?depot4, at ?truck1 ?depot1
at ?hoist0 ?depot3 |
at ?crate1 ?distributor3 | at ?crate1 ?depot1, in ?crate1 ?truck1, at ?crate1 ?depot2, at ?crate1 ?depot3, lifting ?hoist0 ?crate1, in ?crate1 ?truck2, lifting ?hoist2 ?crate1, at ?crate1 ?depot0, at ?crate1 ?distributor2, in ?crate1 ?truck0, lifting ?hoist1 ?crate1, at ?crate1 ?distributor0, at ?crate1 ?distributor1
at ?truck0 ?depot3 | at ?truck0 ?depot4, at ?truck0 ?distributor1, at ?truck0 ?distributor2, at ?truck0 ?distributor0, at ?truck0 ?depot2, at ?truck0 ?depot0, at ?truck0 ?depot1, at ?truck0 ?distributor4, at ?truck0 ?distributor3
at ?truck0 ?distributor3 | at ?truck0 ?depot4, at ?truck0 ?distributor1, at ?truck0 ?distributor2, at ?truck0 ?depot0, at ?truck0 ?depot2, at ?truck0 ?distributor0, at ?truck0 ?depot1, at ?truck0 ?distributor4, at ?truck0 ?depot3
lifting ?hoist1 ?crate3 | available ?hoist1, on ?crate3 ?pallet2, in ?crate3 ?truck2, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate3 ?crate1, at ?crate3 ?distributor0, at ?crate3 ?depot1, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist1 ?crate1, at ?crate3 ?distributor1, lifting ?hoist1 ?crate2, at ?crate3 ?depot3, on ?crate4 ?crate3, in ?crate3 ?truck0, lifting ?hoist1 ?crate0, lifting ?hoist1 ?crate4, clear ?crate3, on ?crate3 ?crate4, at ?crate3 ?distributor2, at ?crate3 ?depot2, on ?crate1 ?crate3, at ?crate3 ?distributor3, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate3 ?crate0, at ?crate3 ?depot0, on ?crate0 ?crate3
at ?pallet0 ?depot0 |
lifting ?hoist1 ?crate1 | at ?crate1 ?depot2, at ?crate1 ?depot3, available ?hoist1, lifting ?hoist0 ?crate1, on ?crate0 ?crate1, in ?crate1 ?truck2, lifting ?hoist1 ?crate3, at ?crate1 ?depot4, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, clear ?crate1, on ?crate1 ?pallet2, in ?crate1 ?truck0, on ?crate1 ?pallet1, at ?crate1 ?depot1, on ?crate1 ?pallet0, lifting ?hoist1 ?crate2, in ?crate1 ?truck1, lifting ?hoist1 ?crate0, lifting ?hoist1 ?crate4, at ?crate1 ?distributor3, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate1 ?crate3, in ?crate1 ?truck3, at ?crate1 ?depot0, at ?crate1 ?distributor2, at ?crate1 ?distributor4, at ?crate1 ?distributor0, at ?crate1 ?distributor1
at ?pallet1 ?distributor2 |
at ?pallet1 ?depot1 |
at ?pallet0 ?distributor0 |
at ?pallet1 ?distributor1 |
at ?pallet0 ?depot2 |
at ?pallet0 ?distributor2 |
at ?pallet0 ?depot1 |
lifting ?hoist0 ?crate3 | lifting ?hoist0 ?crate1, available ?hoist0, on ?crate3 ?pallet2,

in ?crate3 ?truck2, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate3 ?crate1, at ?crate3 ?distributor0, at ?crate3 ?depot1, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, lifting ?hoist0 ?crate4, at ?crate3 ?distributor1, at ?crate3 ?depot3, on ?crate4 ?crate3, in ?crate3 ?truck0, clear ?crate3, on ?crate3 ?crate4, at ?crate3 ?distributor2, at ?crate3 ?depot2, on ?crate1 ?crate3, at ?crate3 ?distributor3, on ?crate3 ?pallet1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, at ?crate3 ?depot0, on ?crate0 ?crate3
at ?pallet1 ?depot0 |
lifting ?hoist0 ?crate1 | at ?crate1 ?depot2, at ?crate1 ?depot3, on ?crate0 ?crate1, available ?hoist0, in ?crate1 ?truck2, at ?crate1 ?depot4, on ?crate1 ?crate4, on ?crate3 ?crate1, on ?crate2 ?crate1, on ?crate1 ?crate2, clear ?crate1, on ?crate1 ?pallet2, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, lifting ?hoist0 ?crate2, lifting ?hoist0 ?crate4, at ?crate1 ?depot1, on ?crate1 ?pallet0, in ?crate1 ?truck1, at ?crate1 ?distributor3, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate1 ?crate3, in ?crate1 ?truck3, lifting ?hoist0 ?crate3, lifting ?hoist0 ?crate0, at ?crate1 ?depot0, at ?crate1 ?distributor2, at ?crate1 ?distributor4, at ?crate1 ?distributor0, at ?crate1 ?distributor1
lifting ?hoist0 ?crate2 | at ?crate2 ?distributor2, on ?crate0 ?crate2, on ?crate2 ?pallet1, at ?crate2 ?distributor0, at ?crate2 ?depot1, lifting ?hoist0 ?crate1, available ?hoist0, on ?crate2 ?crate0, at ?crate2 ?distributor4, on ?crate2 ?pallet0, on ?crate2 ?crate1, clear ?crate2, on ?crate1 ?crate2, at ?crate2 ?depot3, on ?crate2 ?crate3, on ?crate3 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0, lifting ?hoist0 ?crate4, lifting ?hoist1 ?crate2, at ?crate2 ?depot0, at ?crate2 ?distributor1, in ?crate2 ?truck3, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate2 ?pallet2, on ?crate4 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, lifting ?hoist0 ?crate3, lifting ?hoist0 ?crate0, at ?crate2 ?depot4, at ?crate2 ?depot2
at ?pallet0 ?distributor1 |
lifting ?hoist1 ?crate2 | at ?crate2 ?distributor2, on ?crate0 ?crate2, at ?crate2 ?distributor0, on ?crate2 ?pallet1, available ?hoist1, at ?crate2 ?depot1, on ?crate2 ?crate0, at ?crate2 ?distributor4, lifting ?hoist1 ?crate3, on ?crate2 ?pallet0, on ?crate2 ?crate1, clear ?crate2, on ?crate1 ?crate2, at ?crate2 ?depot3, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist1 ?crate1, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0, at ?crate2 ?depot0, lifting ?hoist1 ?crate0, lifting ?hoist1 ?crate4, at ?crate2 ?distributor1, in ?crate2 ?truck3, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate2 ?pallet2, on ?crate4 ?crate2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot4, at ?crate2 ?depot2
lifting ?hoist1 ?crate0 | on ?crate0 ?crate2, on ?crate0 ?pallet1, available ?hoist1, clear ?crate0, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, on ?crate2 ?crate0, in ?crate0 ?truck0, lifting ?hoist1 ?crate3, at ?crate0 ?distributor3, at ?crate0 ?depot0, at ?crate0 ?depot2, at ?crate0 ?depot1, at ?crate0 ?depot3, lifting ?hoist1 ?crate1, on ?crate0 ?crate4, at ?crate0 ?distributor1, lifting ?hoist1 ?crate2, in ?crate0 ?truck3, at ?crate0 ?distributor4, lifting ?hoist1 ?crate4, at ?crate0 ?distributor0, on ?crate1 ?crate0, on ?crate0 ?pallet0, on ?crate0 ?pallet2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, at ?crate0 ?depot4, at ?crate0 ?distributor2, on ?crate0 ?crate3
at ?pallet1 ?depot2 |
at ?pallet1 ?distributor0 |
lifting ?hoist0 ?crate0 | on ?crate0 ?crate2, on ?crate0 ?pallet1, clear ?crate0, on ?crate0 ?crate1, lifting ?hoist0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck2, available ?hoist0, on ?crate2 ?crate0, in ?crate0 ?truck0, at ?crate0 ?distributor3, at ?crate0 ?depot0, at ?crate0 ?depot2, at ?crate0 ?depot1, at ?crate0 ?depot3, lifting ?hoist0 ?crate2,

on ?crate0 ?crate4, lifting ?hoist0 ?crate4, at ?crate0 ?distributor1, in ?crate0 ?truck3, lifting ?hoist1 ?crate0, at ?crate0 ?distributor4, at ?crate0 ?distributor0, on ?crate1 ?crate0, on ?crate0 ?pallet0, on ?crate0 ?pallet2, in ?crate0 ?truck1, lifting ?hoist0 ?crate3, on ?crate3 ?crate0, on ?crate4 ?crate0, at ?crate0 ?depot4, at ?crate0 ?distributor2, on ?crate0 ?crate3
lifting ?hoist2 ?crate0 | on ?crate0 ?crate2, on ?crate0 ?pallet1, clear ?crate0, on ?crate0 ?crate1, in ?crate0 ?truck2, on ?crate2 ?crate0, in ?crate0 ?truck0, at ?crate0 ?distributor3, lifting ?hoist2 ?crate3, at ?crate0 ?depot0, at ?crate0 ?depot2, at ?crate0 ?depot1, at ?crate0 ?depot3, at ?crate0 ?distributor1, lifting ?hoist1 ?crate0, lifting ?hoist2 ?crate2, available ?hoist2, lifting ?hoist2 ?crate1, at ?crate0 ?distributor0, on ?crate1 ?crate0, on ?crate0 ?pallet0, on ?crate0 ?pallet2, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, at ?crate0 ?distributor2, on ?crate0 ?crate3
lifting ?hoist2 ?crate3 | lifting ?hoist2 ?crate0, on ?crate3 ?pallet2, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, on ?crate3 ?crate1, at ?crate3 ?distributor0, at ?crate3 ?depot1, on ?crate2 ?crate3, on ?crate3 ?crate2, at ?crate3 ?distributor1, at ?crate3 ?depot3, in ?crate3 ?truck0, clear ?crate3, lifting ?hoist2 ?crate2, available ?hoist2, lifting ?hoist2 ?crate1, at ?crate3 ?distributor2, at ?crate3 ?depot2, on ?crate1 ?crate3, at ?crate3 ?distributor3, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate3 ?crate0, at ?crate3 ?depot0, on ?crate0 ?crate3
lifting ?hoist2 ?crate2 | on ?crate0 ?crate2, at ?crate2 ?distributor2, at ?crate2 ?distributor0, on ?crate2 ?pallet1, at ?crate2 ?depot1, lifting ?hoist2 ?crate0, on ?crate2 ?crate0, lifting ?hoist2 ?crate3, on ?crate2 ?pallet0, on ?crate2 ?crate1, clear ?crate2, on ?crate1 ?crate2, at ?crate2 ?depot3, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, at ?crate2 ?distributor3, in ?crate2 ?truck0, lifting ?hoist1 ?crate2, at ?crate2 ?depot0, at ?crate2 ?distributor1, available ?hoist2, lifting ?hoist2 ?crate1, on ?crate2 ?pallet2, in ?crate2 ?truck2, in ?crate2 ?truck1, at ?crate2 ?depot2
lifting ?hoist2 ?crate1 | at ?crate1 ?depot2, at ?crate1 ?depot3, lifting ?hoist0 ?crate1, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate1 ?truck2, lifting ?hoist2 ?crate3, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, clear ?crate1, on ?crate1 ?pallet2, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, at ?crate1 ?depot1, on ?crate1 ?pallet0, in ?crate1 ?truck1, lifting ?hoist2 ?crate2, available ?hoist2, at ?crate1 ?distributor3, on ?crate1 ?crate0, on ?crate1 ?crate3, at ?crate1 ?depot0, at ?crate1 ?distributor2, at ?crate1 ?distributor0, at ?crate1 ?distributor1
at ?pallet2 ?depot1 |
at ?pallet2 ?distributor1 |
on ?crate3 ?pallet2 | in ?crate3 ?truck0, on ?crate3 ?crate4, on ?crate2 ?pallet2, lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate3 ?crate1, on ?crate0 ?pallet2, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate1 ?pallet2, clear ?pallet2, on ?crate3 ?crate2, on ?crate4 ?pallet2
clear ?pallet1 | on ?crate2 ?pallet1, on ?crate0 ?pallet1, clear ?crate0, clear ?crate3, clear ?crate4, clear ?crate2, clear ?crate1, on ?crate3 ?pallet1, clear ?pallet0, clear ?pallet2, on ?crate1 ?pallet1, on ?crate4 ?pallet1
on ?crate0 ?pallet2 | on ?crate0 ?crate2, lifting ?hoist1 ?crate0, on ?crate0 ?pallet1, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, in ?crate0 ?truck0, on ?crate3 ?pallet2, on ?crate2 ?pallet2, on ?crate0 ?pallet0, in ?crate0 ?truck1, lifting ?hoist0 ?crate0, on ?crate1 ?pallet2, clear ?pallet2, on ?crate0 ?crate4, on ?crate4 ?pallet2, on ?crate0 ?crate3
at ?pallet2 ?depot0 |
at ?pallet2 ?depot2 |
clear ?pallet0 | on ?crate1 ?pallet0, clear ?pallet1, clear ?crate0, clear ?crate3, clear

?crate4, on ?crate3 ?pallet0, on ?crate2 ?pallet0, on ?crate0 ?pallet0, clear ?crate2, on ?crate4 ?pallet0, clear ?crate1, clear ?pallet2
on ?crate1 ?pallet2 | on ?crate1 ?pallet0, in ?crate1 ?truck1, lifting ?hoist0 ?crate1, on ?crate3 ?pallet2, lifting ?hoist2 ?crate1, on ?crate2 ?pallet2, on ?crate1 ?crate0, on ?crate1 ?crate4, on ?crate1 ?crate3, on ?crate0 ?pallet2, on ?crate1 ?crate2, clear ?pallet2, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, on ?crate4 ?pallet2
at ?pallet2 ?distributor0 |
at ?pallet2 ?distributor2 |
on ?crate2 ?pallet2 | lifting ?hoist1 ?crate2, on ?crate2 ?pallet1, on ?crate2 ?crate4, lifting ?hoist2 ?crate2, on ?crate2 ?crate0, on ?crate3 ?pallet2, on ?crate2 ?pallet0, on ?crate2 ?crate1, on ?crate0 ?pallet2, in ?crate2 ?truck1, on ?crate1 ?pallet2, clear ?pallet2, on ?crate2 ?crate3, lifting ?hoist0 ?crate2, on ?crate4 ?pallet2, in ?crate2 ?truck0
clear ?pallet2 | clear ?pallet1, clear ?crate0, clear ?crate3, clear ?crate4, on ?crate3 ?pallet2, on ?crate2 ?pallet2, on ?crate0 ?pallet2, clear ?crate2, clear ?crate1, clear ?pallet0, on ?crate1 ?pallet2, on ?crate4 ?pallet2
in ?crate0 ?truck2 | in ?crate0 ?truck1, in ?crate0 ?truck3, lifting ?hoist1 ?crate0, lifting ?hoist0 ?crate0, in ?crate0 ?truck0
in ?crate1 ?truck2 | in ?crate1 ?truck1, in ?crate1 ?truck3, lifting ?hoist0 ?crate1, in ?crate1 ?truck0, lifting ?hoist1 ?crate1
in ?crate2 ?truck2 | lifting ?hoist1 ?crate2, in ?crate2 ?truck1, in ?crate2 ?truck3, lifting ?hoist0 ?crate2, in ?crate2 ?truck0
at ?truck3 ?distributor2 | at ?truck3 ?distributor0, at ?truck3 ?depot1, at ?truck3 ?distributor3, at ?truck3 ?depot0, at ?truck3 ?depot2, at ?truck3 ?depot3, at ?truck3 ?distributor1
at ?truck3 ?depot0 | at ?truck3 ?distributor0, at ?truck3 ?depot1, at ?truck3 ?distributor2, at ?truck3 ?distributor3, at ?truck3 ?depot2, at ?truck3 ?depot3, at ?truck3 ?distributor1
at ?truck3 ?distributor0 | at ?truck3 ?depot1, at ?truck3 ?distributor2, at ?truck3 ?distributor3, at ?truck3 ?depot0, at ?truck3 ?depot2, at ?truck3 ?depot3, at ?truck3 ?distributor1
at ?truck3 ?depot1 | at ?truck3 ?distributor0, at ?truck3 ?distributor2, at ?truck3 ?distributor3, at ?truck3 ?depot0, at ?truck3 ?depot2, at ?truck3 ?depot3, at ?truck3 ?distributor1
at ?truck3 ?depot2 | at ?truck3 ?distributor0, at ?truck3 ?depot1, at ?truck3 ?distributor2, at ?truck3 ?distributor3, at ?truck3 ?depot0, at ?truck3 ?depot3, at ?truck3 ?distributor1
at ?truck3 ?distributor1 | at ?truck3 ?distributor0, at ?truck3 ?depot1, at ?truck3 ?distributor2, at ?truck3 ?distributor3, at ?truck3 ?depot0, at ?truck3 ?depot2, at ?truck3 ?depot3
at ?truck2 ?distributor3 | at ?truck2 ?distributor4, at ?truck2 ?depot1, at ?truck2 ?depot0, at ?truck2 ?distributor2, at ?truck2 ?depot4, at ?truck2 ?depot3, at ?truck2 ?distributor1, at ?truck2 ?distributor0, at ?truck2 ?depot2
at ?truck2 ?depot3 | at ?truck2 ?distributor4, at ?truck2 ?depot1, at ?truck2 ?distributor3, at ?truck2 ?depot0, at ?truck2 ?distributor2, at ?truck2 ?depot4, at ?truck2 ?distributor1, at ?truck2 ?distributor0, at ?truck2 ?depot2
at ?hoist2 ?depot3 |
at ?hoist2 ?distributor3 |
at ?pallet0 ?distributor3 |
at ?pallet1 ?depot3 |
at ?pallet0 ?depot3 |
at ?pallet1 ?distributor3 |
at ?truck1 ?depot4 | at ?truck1 ?distributor0, at ?truck1 ?distributor3, at ?truck1 ?depot2, at ?truck1 ?depot0, at ?truck1 ?depot1, at ?truck1 ?depot3, at ?truck1 ?distributor1, at

?truck1 ?distributor4, at ?truck1 ?distributor2
at ?truck0 ?depot4 | at ?truck0 ?distributor2, at ?truck0 ?depot0, at ?truck0 ?depot1, at ?truck0 ?depot3, at ?truck0 ?distributor1, at ?truck0 ?depot2, at ?truck0 ?distributor0, at ?truck0 ?distributor4, at ?truck0 ?distributor3
at ?truck1 ?distributor4 | at ?truck1 ?distributor0, at ?truck1 ?distributor3, at ?truck1 ?depot2, at ?truck1 ?depot0, at ?truck1 ?depot4, at ?truck1 ?depot1, at ?truck1 ?depot3, at ?truck1 ?distributor1, at ?truck1 ?distributor2
at ?truck0 ?distributor4 | at ?truck0 ?depot4, at ?truck0 ?distributor2, at ?truck0 ?depot0, at ?truck0 ?depot1, at ?truck0 ?depot3, at ?truck0 ?distributor1, at ?truck0 ?depot2, at ?truck0 ?distributor0, at ?truck0 ?distributor3
at ?hoist0 ?depot4 |
at ?hoist0 ?distributor4 |
at ?hoist1 ?depot4 |
at ?hoist1 ?distributor4 |
at ?crate3 ?depot3 | in ?crate3 ?truck0, at ?crate3 ?distributor2, lifting ?hoist1 ?crate3, at ?crate3 ?depot2, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, at ?crate3 ?distributor3, at ?crate3 ?distributor0, at ?crate3 ?depot1, lifting ?hoist0 ?crate3, at ?crate3 ?depot0, at ?crate3 ?distributor1
at ?crate3 ?distributor3 | at ?crate3 ?depot3, in ?crate3 ?truck0, at ?crate3 ?distributor2, lifting ?hoist1 ?crate3, at ?crate3 ?depot2, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, at ?crate3 ?distributor0, at ?crate3 ?depot1, lifting ?hoist0 ?crate3, at ?crate3 ?depot0, at ?crate3 ?distributor1
at ?crate4 ?distributor2 | lifting ?hoist1 ?crate4, at ?crate4 ?depot2, in ?crate4 ?truck1, at ?crate4 ?depot1, at ?crate4 ?distributor1, at ?crate4 ?distributor0, in ?crate4 ?truck0, at ?crate4 ?depot0, lifting ?hoist0 ?crate4, lifting ?hoist2 ?crate4
at ?crate4 ?depot2 | lifting ?hoist1 ?crate4, in ?crate4 ?truck1, at ?crate4 ?distributor2, at ?crate4 ?depot1, at ?crate4 ?distributor1, at ?crate4 ?distributor0, in ?crate4 ?truck0, at ?crate4 ?depot0, lifting ?hoist0 ?crate4, lifting ?hoist2 ?crate4
at ?crate4 ?distributor0 | lifting ?hoist1 ?crate4, at ?crate4 ?depot2, in ?crate4 ?truck1, at ?crate4 ?distributor2, at ?crate4 ?depot1, at ?crate4 ?distributor1, in ?crate4 ?truck0, at ?crate4 ?depot0, lifting ?hoist0 ?crate4, lifting ?hoist2 ?crate4
on ?crate1 ?crate4 | lifting ?hoist0 ?crate1, on ?crate0 ?crate1, clear ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, on ?crate1 ?pallet2, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, on ?crate1 ?pallet0, in ?crate1 ?truck1, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, lifting ?hoist2 ?crate1, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate4 ?crate2, on ?crate1 ?crate3, in ?crate4 ?truck1, on ?crate4 ?pallet0, on ?crate4 ?crate0, in ?crate4 ?truck0
on ?crate4 ?pallet1 | on ?crate2 ?pallet1, on ?crate0 ?pallet1, clear ?pallet1, on ?crate1 ?crate4, on ?crate1 ?pallet1, on ?crate0 ?crate4, lifting ?hoist0 ?crate4, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, on ?crate4 ?crate1, on ?crate4 ?crate2, in ?crate4 ?truck1, on ?crate4 ?pallet0, on ?crate3 ?pallet1, on ?crate4 ?crate0, in ?crate4 ?truck0, on ?crate4 ?pallet2, lifting ?hoist2 ?crate4
on ?crate4 ?crate3 | lifting ?hoist1 ?crate3, on ?crate3 ?pallet0, in ?crate3 ?truck1, on ?crate1 ?crate4, on ?crate3 ?crate1, on ?crate2 ?crate3, on ?crate3 ?crate2, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, in ?crate3 ?truck0, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, clear ?crate3, on ?crate3 ?crate4, on ?crate4 ?crate1, on ?crate4 ?crate2, on ?crate1 ?crate3, in ?crate4 ?truck1, on ?crate4 ?pallet0, lifting ?hoist0 ?crate3, on ?crate3 ?pallet1, on ?crate4 ?crate0, on ?crate3 ?crate0, in ?crate4 ?truck0, on ?crate4 ?pallet2, on ?crate0 ?crate3, lifting ?hoist2 ?crate4
on ?crate3 ?crate4 | clear ?crate4, on ?crate3 ?pallet2, lifting ?hoist1 ?crate3, on ?crate3

?pallet0, in ?crate3 ?truck1, lifting ?hoist2 ?crate3, on ?crate1 ?crate4, on ?crate3 ?crate1, on ?crate2 ?crate3, on ?crate3 ?crate2, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, on ?crate4 ?crate3, in ?crate3 ?truck0, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate4 ?crate1, on ?crate4 ?crate2, on ?crate1 ?crate3, in ?crate4 ?truck1, lifting ?hoist0 ?crate3, on ?crate4 ?pallet0, on ?crate3 ?pallet1, on ?crate3 ?crate0, on ?crate4 ?crate0, in ?crate4 ?truck0, on ?crate0 ?crate3
on ?crate4 ?crate1 | lifting ?hoist0 ?crate1, on ?crate0 ?crate1, on ?crate1 ?crate4, on ?crate2 ?crate1, on ?crate3 ?crate1, on ?crate1 ?crate2, clear ?crate1, in ?crate1 ?truck0, on ?crate1 ?pallet1, lifting ?hoist1 ?crate1, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, on ?crate1 ?pallet0, in ?crate1 ?truck1, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, on ?crate1 ?crate0, on ?crate4 ?crate2, on ?crate1 ?crate3, in ?crate4 ?truck1, on ?crate4 ?pallet0, on ?crate4 ?crate0, in ?crate4 ?truck0, on ?crate4 ?pallet2, lifting ?hoist2 ?crate4
at ?crate4 ?depot1 | lifting ?hoist1 ?crate4, at ?crate4 ?depot2, in ?crate4 ?truck1, at ?crate4 ?distributor2, at ?crate4 ?distributor1, at ?crate4 ?distributor0, in ?crate4 ?truck0, at ?crate4 ?depot0, lifting ?hoist0 ?crate4, lifting ?hoist2 ?crate4
at ?crate4 ?distributor1 | lifting ?hoist1 ?crate4, at ?crate4 ?depot2, in ?crate4 ?truck1, at ?crate4 ?distributor2, at ?crate4 ?depot1, at ?crate4 ?distributor0, in ?crate4 ?truck0, at ?crate4 ?depot0, lifting ?hoist0 ?crate4, lifting ?hoist2 ?crate4
on ?crate0 ?crate4 | on ?crate0 ?crate2, on ?crate0 ?pallet1, on ?crate0 ?crate1, lifting ?hoist2 ?crate0, clear ?crate4, on ?crate2 ?crate0, in ?crate0 ?truck0, on ?crate1 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, on ?crate4 ?crate3, lifting ?hoist1 ?crate0, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate4 ?crate2, on ?crate0 ?pallet0, on ?crate0 ?pallet2, in ?crate4 ?truck1, in ?crate0 ?truck1, on ?crate4 ?pallet0, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, on ?crate4 ?crate0, in ?crate4 ?truck0, on ?crate0 ?crate3
on ?crate4 ?crate2 | on ?crate0 ?crate2, on ?crate2 ?pallet1, on ?crate2 ?crate0, on ?crate1 ?crate4, on ?crate2 ?pallet0, on ?crate2 ?crate1, clear ?crate2, on ?crate1 ?crate2, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, on ?crate0 ?crate4, on ?crate4 ?pallet1, in ?crate2 ?truck0, lifting ?hoist0 ?crate4, lifting ?hoist1 ?crate2, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, on ?crate4 ?crate1, in ?crate4 ?truck1, on ?crate4 ?pallet0, in ?crate2 ?truck1, on ?crate4 ?crate0, in ?crate4 ?truck0, on ?crate4 ?pallet2, lifting ?hoist2 ?crate4
at ?crate4 ?depot0 | lifting ?hoist1 ?crate4, at ?crate4 ?depot2, in ?crate4 ?truck1, at ?crate4 ?distributor2, at ?crate4 ?depot1, at ?crate4 ?distributor1, at ?crate4 ?distributor0, in ?crate4 ?truck0, lifting ?hoist0 ?crate4, lifting ?hoist2 ?crate4
on ?crate4 ?pallet2 | on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate3 ?pallet2, on ?crate2 ?pallet2, on ?crate4 ?crate1, on ?crate4 ?crate2, in ?crate4 ?truck1, on ?crate0 ?pallet2, on ?crate4 ?pallet0, on ?crate4 ?crate0, on ?crate1 ?pallet2, in ?crate4 ?truck0, clear ?pallet2, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, lifting ?hoist2 ?crate4
clear ?crate4 | lifting ?hoist1 ?crate4, clear ?pallet1, clear ?crate0, on ?crate2 ?crate4, clear ?crate3, on ?crate3 ?crate4, on ?crate1 ?crate4, in ?crate4 ?truck1, clear ?crate2, clear ?pallet0, clear ?crate1, clear ?pallet2, in ?crate4 ?truck0, on ?crate0 ?crate4, lifting ?hoist0 ?crate4, lifting ?hoist2 ?crate4
on ?crate2 ?crate4 | on ?crate0 ?crate2, on ?crate2 ?pallet1, clear ?crate4, on ?crate2 ?crate0, on ?crate1 ?crate4, on ?crate2 ?pallet0, on ?crate2 ?crate1, on ?crate1 ?crate2, on ?crate2 ?crate3, on ?crate3 ?crate2, lifting ?hoist0 ?crate2, on ?crate0 ?crate4, on ?crate4 ?pallet1, in ?crate2 ?truck0, lifting ?hoist0 ?crate4, lifting ?hoist1 ?crate2, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, lifting ?hoist2 ?crate2, on ?crate3 ?crate4, on ?crate2 ?pallet2, on ?crate4 ?crate1, on ?crate4 ?crate2, in ?crate4 ?truck1, in ?crate2 ?truck1, on ?crate4 ?pallet0, on ?crate4 ?crate0, in ?crate4 ?truck0

on ?crate4 ?pallet0 | on ?crate3 ?pallet0, on ?crate1 ?crate4, on ?crate2 ?pallet0, clear ?pallet0, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, on ?crate1 ?pallet0, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, on ?crate4 ?crate1, on ?crate4 ?crate2, on ?crate0 ?pallet0, in ?crate4 ?truck1, on ?crate4 ?crate0, in ?crate4 ?truck0, on ?crate4 ?pallet2, lifting ?hoist2 ?crate4
on ?crate4 ?crate0 | on ?crate0 ?crate2, on ?crate0 ?pallet1, clear ?crate0, on ?crate0 ?crate1, on ?crate2 ?crate0, in ?crate0 ?truck0, on ?crate1 ?crate4, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, lifting ?hoist1 ?crate0, on ?crate2 ?crate4, on ?crate3 ?crate4, on ?crate1 ?crate0, on ?crate4 ?crate1, on ?crate4 ?crate2, on ?crate0 ?pallet0, in ?crate4 ?truck1, in ?crate0 ?truck1, on ?crate4 ?pallet0, lifting ?hoist0 ?crate0, on ?crate3 ?crate0, in ?crate4 ?truck0, on ?crate4 ?pallet2, on ?crate0 ?crate3, lifting ?hoist2 ?crate4
in ?crate0 ?truck3 | in ?crate0 ?truck1, lifting ?hoist1 ?crate0, lifting ?hoist0 ?crate0, in ?crate0 ?truck2, in ?crate0 ?truck0
in ?crate2 ?truck3 | in ?crate2 ?truck2, lifting ?hoist1 ?crate2, in ?crate2 ?truck1, lifting ?hoist0 ?crate2, in ?crate2 ?truck0
in ?crate1 ?truck3 | in ?crate1 ?truck1, lifting ?hoist0 ?crate1, in ?crate1 ?truck0, in ?crate1 ?truck2, lifting ?hoist1 ?crate1
at ?truck3 ?depot3 | at ?truck3 ?distributor0, at ?truck3 ?depot1, at ?truck3 ?distributor2, at ?truck3 ?distributor3, at ?truck3 ?depot0, at ?truck3 ?depot2, at ?truck3 ?distributor1
at ?truck3 ?distributor3 | at ?truck3 ?distributor0, at ?truck3 ?depot1, at ?truck3 ?distributor2, at ?truck3 ?depot0, at ?truck3 ?depot2, at ?truck3 ?depot3, at ?truck3 ?distributor1
in ?crate4 ?truck1 | clear ?crate4, on ?crate1 ?crate4, at ?crate4 ?distributor2, at ?crate4 ?distributor1, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, at ?crate4 ?depot2, on ?crate4 ?crate1, on ?crate4 ?crate2, at ?crate4 ?depot1, on ?crate4 ?pallet0, on ?crate4 ?crate0, at ?crate4 ?distributor0, in ?crate4 ?truck0, at ?crate4 ?depot0
in ?crate4 ?truck0 | clear ?crate4, on ?crate1 ?crate4, at ?crate4 ?distributor2, at ?crate4 ?distributor1, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, at ?crate4 ?depot2, on ?crate4 ?crate1, on ?crate4 ?crate2, in ?crate4 ?truck1, at ?crate4 ?depot1, on ?crate4 ?pallet0, on ?crate4 ?crate0, at ?crate4 ?distributor0, at ?crate4 ?depot0
lifting ?hoist0 ?crate4 | lifting ?hoist0 ?crate1, clear ?crate4, available ?hoist0, on ?crate1 ?crate4, at ?crate4 ?distributor2, at ?crate4 ?distributor1, lifting ?hoist0 ?crate2, on ?crate0 ?crate4, on ?crate4 ?pallet1, on ?crate4 ?crate3, lifting ?hoist1 ?crate4, on ?crate2 ?crate4, on ?crate3 ?crate4, at ?crate4 ?depot2, on ?crate4 ?crate1, on ?crate4 ?crate2, in ?crate4 ?truck1, at ?crate4 ?depot1, on ?crate4 ?pallet0, lifting ?hoist0 ?crate3, lifting ?hoist0 ?crate0, on ?crate4 ?crate0, at ?crate4 ?distributor0, in ?crate4 ?truck0, at ?crate4 ?depot0, on ?crate4 ?pallet2
lifting ?hoist1 ?crate4 | available ?hoist1, clear ?crate4, lifting ?hoist1 ?crate3, on ?crate1 ?crate4, at ?crate4 ?distributor2, at ?crate4 ?distributor1, lifting ?hoist1 ?crate1, on ?crate0 ?crate4, on ?crate4 ?pallet1, lifting ?hoist0 ?crate4, lifting ?hoist1 ?crate2, on ?crate4 ?crate3, lifting ?hoist1 ?crate0, on ?crate2 ?crate4, on ?crate3 ?crate4, at ?crate4 ?depot2, on ?crate4 ?crate1, on ?crate4 ?crate2, in ?crate4 ?truck1, at ?crate4 ?depot1, on ?crate4 ?pallet0, on ?crate4 ?crate0, at ?crate4 ?distributor0, in ?crate4 ?truck0, at ?crate4 ?depot0, on ?crate4 ?pallet2
at ?truck2 ?distributor4 | at ?truck2 ?depot1, at ?truck2 ?distributor3, at ?truck2 ?depot0, at ?truck2 ?distributor2, at ?truck2 ?depot4, at ?truck2 ?depot3, at ?truck2 ?distributor1, at ?truck2 ?distributor0, at ?truck2 ?depot2
at ?truck2 ?depot4 | at ?truck2 ?distributor4, at ?truck2 ?depot1, at ?truck2 ?distributor3,

at ?truck2 ?depot0, at ?truck2 ?distributor2, at ?truck2 ?depot3, at ?truck2 ?distributor1, at ?truck2 ?distributor0, at ?truck2 ?depot2

C.3.2 Understanding the found macro-operators

macro	I	U	R
action macro-3-actions-1-3-4–4-3	1	0	0
action macro-3-actions-1-3-0–6-2	1	0	0
action macro-3-actions-1-3-4–7-5	0	1	0
action macro-3-actions-1-3-0–9-7	1	0	0
action macro-3-actions-1-3-2–12-6	1	0	0
action macro-3-actions-1-3-3–13-8	1	0	0
action macro-4-actions-1-3-4-2–14-9	1	0	0
action macro-3-actions-1-0-0–17-2	0	1	0
action macro-3-actions-1-4-2–23-12	1	0	0
action macro-2-actions-1-4–24-13	1	0	0
action macro-2-actions-1-2–24-14	1	0	0
action macro-3-actions-1-0-0–25-2	0	1	0
action macro-4-actions-1-0-0-4–29-1	1	0	0
action macro-2-actions-1-2–31-15	1	0	0
action macro-3-actions-3-0-0–36-19	0	1	0
action macro-2-actions-3-4–40-20	1	0	0
action macro-2-actions-3-0–41-19	1	0	0
action macro-2-actions-3-4–42-22	0	1	0
action macro-2-actions-3-2–30-24	1	0	0
action macro-2-actions-3-3–43-25	1	0	0
action macro-3-actions-0-1-4–49-26	1	0	0
action macro-3-actions-0-1-0–50-27	0	1	0
action macro-4-actions-0-4-3-0–56-29	0	1	0
action macro-3-actions-0-4-3–57-29	0	1	0
action macro-3-actions-0-4-3–58-31	1	0	0
action macro-4-actions-0-1-3-2–65-32	1	0	0
action macro-3-actions-0-1-0–70-27	0	1	0
action macro-3-actions-0-1-2–71-32	1	0	0
action macro-2-actions-0-3–73-30	1	0	0
action macro-4-actions-0-0-0-0–77-35	0	1	0
action macro-3-actions-0-0-0–78-35	0	1	0
action macro-3-actions-0-0-0–79-37	1	0	0
action macro-3-actions-0-0-0–80-37	1	0	0
action macro-4-actions-0-0-0-0–84-35	0	1	0
action macro-3-actions-0-0-0–87-35	0	1	0
action macro-3-actions-0-0-0–92-37	1	0	0
action macro-4-actions-0-3-4-0–56-29	0	1	0
action macro-3-actions-0-3-4–57-29	0	1	0
action macro-3-actions-0-3-2–99-39	1	0	0
action macro-3-actions-0-3-0–101-30	1	0	0
action macro-3-actions-0-3-4–100-41	1	0	0
action macro-3-actions-0-3-3–58-31	1	0	0
action macro-3-actions-0-4-0–101-30	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-0-4-0–107-30	1	0	0
action macro-3-actions-0-4-2–110-31	1	0	0
action macro-4-actions-0-4-3-0–95-31	1	0	0
action macro-3-actions-0-4-4–112-43	1	0	0
action macro-3-actions-0-0-1–115-27	0	1	0
action macro-3-actions-0-0-3–118-29	0	1	0
action macro-3-actions-0-0-1–122-27	0	1	0
action macro-4-actions-0-0-0-0–125-35	0	1	0
action macro-3-actions-0-0-0–128-35	0	1	0
action macro-3-actions-0-0-2–122-27	0	1	0
action macro-3-actions-0-0-4–118-29	0	1	0
action macro-4-actions-0-0-0-0–134-44	1	0	0
action macro-3-actions-0-0-0–135-44	1	0	0
action macro-4-actions-0-0-0-0–136-44	1	0	0
action macro-3-actions-0-0-0–137-44	1	0	0
action macro-4-actions-0-0-0-0–138-44	0	1	0
action macro-4-actions-0-0-0-0–139-44	0	1	0
action macro-3-actions-0-0-0–142-44	0	1	0
action macro-3-actions-0-0-4–143-30	1	0	0
action macro-3-actions-0-0-0–144-37	0	1	0
action macro-4-actions-0-0-3-0–145-30	1	0	0
action macro-4-actions-0-0-0-0–147-35	0	1	0
action macro-3-actions-0-0-0–152-35	0	1	0
action macro-3-actions-0-0-3–122-38	0	1	0
action macro-3-actions-0-0-0–154-37	0	1	0
action macro-3-actions-0-0-4–122-38	0	1	0
action macro-3-actions-0-0-2–115-27	0	1	0
action macro-4-actions-0-0-4-3–155-29	0	1	0
action macro-4-actions-0-0-3-3–117-31	1	0	0
action macro-4-actions-0-0-3-4–155-29	0	1	0
action macro-3-actions-0-0-0–157-37	0	1	0
action macro-3-actions-0-0-3–143-30	1	0	0
action macro-3-actions-0-0-0–158-37	0	1	0
action macro-2-actions-0-0–163-46	0	1	0
action macro-3-actions-0-2-2–164-32	1	0	0
action macro-3-actions-0-0-4–165-45	1	0	0
action macro-3-actions-0-0-0–166-47	1	0	0
action macro-3-actions-0-0-3–165-45	1	0	0
action macro-4-actions-0-0-4-0–167-45	1	0	0
action macro-3-actions-0-0-0–168-44	1	0	0
action macro-3-actions-0-0-2–169-48	1	0	0
action macro-4-actions-0-0-3-0–167-45	1	0	0
action macro-3-actions-0-0-1–179-48	1	0	0
action macro-4-actions-0-0-0-0–181-44	1	0	0
action macro-4-actions-0-0-0-1–183-48	1	0	0
action macro-4-actions-0-0-0-3–184-45	1	0	0
action macro-4-actions-0-0-0-4–184-45	1	0	0
action macro-3-actions-0-0-0–185-44	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-0-0-0–186-47	1	0	0
action macro-3-actions-0-0-0–187-47	1	0	0
action macro-3-actions-0-0-0–168-49	1	0	0
action macro-3-actions-0-0-0–188-47	1	0	0
action macro-3-actions-0-0-0–189-47	1	0	0
action macro-3-actions-0-0-2–179-48	1	0	0
action macro-2-actions-0-0–191-44	1	0	0
action macro-4-actions-0-0-0-0–195-44	1	0	0
action macro-4-actions-0-0-0-4–196-45	1	0	0
action macro-3-actions-0-0-0–197-47	1	0	0
action macro-3-actions-0-0-0–199-44	1	0	0
action macro-3-actions-0-0-4–202-45	1	0	0
action macro-3-actions-0-0-3–202-45	1	0	0
action macro-4-actions-0-0-0-0–203-44	1	0	0
action macro-4-actions-0-0-0-0–204-44	1	0	0
action macro-4-actions-0-0-0-3–206-45	1	0	0
action macro-4-actions-0-0-0-0–207-47	1	0	0
action macro-4-actions-0-0-0-0–208-47	1	0	0
action macro-4-actions-0-0-0-4–206-45	1	0	0
action macro-4-actions-0-0-0-4–209-45	1	0	0
action macro-4-actions-0-0-0-0–210-47	1	0	0
action macro-4-actions-0-0-0-0–211-47	1	0	0
action macro-3-actions-0-0-0–212-44	1	0	0
action macro-3-actions-0-0-0–214-47	1	0	0
action macro-3-actions-0-0-1–215-48	1	0	0
action macro-4-actions-0-0-0-0–216-44	1	0	0
action macro-4-actions-0-0-0-4–217-45	1	0	0
action macro-3-actions-0-0-0–218-44	1	0	0
action macro-3-actions-0-0-0–219-47	1	0	0
action macro-3-actions-0-0-0–220-47	1	0	0
action macro-3-actions-0-0-4–221-45	1	0	0
action macro-4-actions-0-0-4-0–222-45	1	0	0
action macro-4-actions-0-0-0-0–223-47	1	0	0
action macro-3-actions-0-0-0–224-47	1	0	0
action macro-4-actions-0-0-0-0–225-47	1	0	0
action macro-4-actions-0-0-0-0–226-47	1	0	0
action macro-3-actions-0-0-0–227-47	1	0	0
action macro-3-actions-0-0-2–215-48	1	0	0
action macro-2-actions-0-0–229-44	1	0	0
action macro-3-actions-0-4-2–259-50	1	0	0
action macro-2-actions-0-4–73-30	1	0	0
action macro-3-actions-0-0-0–260-37	0	1	0
action macro-4-actions-0-0-0-0–265-35	0	1	0
action macro-3-actions-0-0-0–268-35	0	1	0
action macro-4-actions-0-0-0-0–271-35	0	1	0
action macro-3-actions-0-0-0–274-35	0	1	0
action macro-3-actions-0-0-0–275-37	1	0	0
action macro-3-actions-0-0-0–276-37	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-0-0-0–284-37	1	0	0
action macro-4-actions-0-0-0-0–285-37	0	1	0
action macro-4-actions-0-1-3-0–288-31	1	0	0
action macro-3-actions-0-1-3–289-31	1	0	0
action macro-4-actions-0-1-3-1–65-39	1	0	0
action macro-4-actions-0-1-3-3–63-31	1	0	0
action macro-4-actions-0-3-3-0–95-31	1	0	0
action macro-4-actions-0-3-0-0–293-30	1	0	0
action macro-3-actions-0-3-0–107-30	1	0	0
action macro-3-actions-0-3-0–295-30	1	0	0
action macro-3-actions-0-3-0–296-30	1	0	0
action macro-3-actions-0-3-4–297-54	1	0	0
action macro-3-actions-0-0-0–298-49	1	0	0
action macro-2-actions-0-0–299-49	1	0	0
action macro-3-actions-0-0-4–300-55	1	0	0
action macro-3-actions-0-0-0–301-49	1	0	0
action macro-3-actions-0-0-0–302-49	1	0	0
action macro-3-actions-0-0-3–300-55	1	0	0
action macro-4-actions-0-0-0-0–312-44	0	1	0
action macro-4-actions-0-0-0-0–312-49	0	1	0
action macro-3-actions-0-0-0–319-44	0	1	0
action macro-4-actions-0-0-0-0–320-47	0	1	0
action macro-4-actions-0-0-0-4–323-45	0	1	0
action macro-4-actions-0-0-0-0–324-44	0	1	0
action macro-4-actions-0-0-0-3–323-45	0	1	0
action macro-4-actions-0-0-0-1–327-48	0	1	0
action macro-4-actions-0-0-0-0–328-47	0	1	0
action macro-4-actions-0-0-0-0–329-47	0	1	0
action macro-4-actions-0-0-0-0–330-47	0	1	0
action macro-4-actions-0-0-0-0–331-47	0	1	0
action macro-3-actions-0-0-0–332-44	0	1	0
action macro-4-actions-0-0-1-0–342-48	0	1	0
action macro-3-actions-0-0-0–338-49	1	0	0
action macro-4-actions-0-0-0-0–324-49	0	1	0
action macro-3-actions-0-0-0–332-49	0	1	0
action macro-3-actions-0-0-4–303-56	1	0	0
action macro-4-actions-0-0-0-0–348-47	0	1	0
action macro-3-actions-0-0-3–303-56	1	0	0
action macro-3-actions-0-0-0–352-49	1	0	0
action macro-4-actions-0-0-0-0–356-47	0	1	0
action macro-3-actions-0-0-0–358-49	1	0	0
action macro-4-actions-0-0-0-0–359-47	0	1	0
action macro-4-actions-0-0-3-4–364-45	0	1	0
action macro-4-actions-0-0-4-3–364-45	0	1	0
action macro-2-actions-0-0–366-44	0	1	0
action macro-3-actions-0-3-4–367-53	1	0	0
action macro-3-actions-0-3-0–368-30	1	0	0
action macro-4-actions-0-3-0-4–369-53	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-0-3-3–370-59	1	0	0
action macro-3-actions-0-0-0–372-37	0	1	0
action macro-3-actions-0-0-0–373-37	1	0	0
action macro-3-actions-0-0-0–378-37	0	1	0
action macro-3-actions-0-0-0–379-37	1	0	0
action macro-3-actions-0-0-0–380-37	1	0	0
action macro-3-actions-0-0-0–382-37	1	0	0
action macro-4-actions-0-0-0-0–385-37	0	1	0
action macro-3-actions-0-0-0–386-37	0	1	0
action macro-3-actions-0-1-3–69-32	1	0	0
action macro-4-actions-0-3-3-3–104-60	1	0	0
action macro-4-actions-0-3-3-4–104-31	1	0	0
action macro-3-actions-0-3-4–112-43	1	0	0
action macro-4-actions-0-3-3-4–392-31	1	0	0
action macro-3-actions-0-2-0–70-27	0	1	0
action macro-3-actions-0-4-0–295-30	1	0	0
action macro-3-actions-0-4-0–296-30	1	0	0
action macro-3-actions-0-4-3–398-30	1	0	0
action macro-3-actions-0-4-4–370-59	1	0	0
action macro-4-actions-0-1-3-4–66-61	1	0	0
action macro-4-actions-0-3-3-4–104-60	1	0	0
action macro-3-actions-0-4-3–112-43	1	0	0
action macro-3-actions-0-0-0–403-37	1	0	0
action macro-3-actions-0-0-0–407-37	0	1	0
action macro-3-actions-0-3-4–408-38	0	1	0
action macro-3-actions-0-0-0–411-37	1	0	0
action macro-3-actions-0-0-0–412-37	1	0	0
action macro-4-actions-0-0-0-0–414-37	0	1	0
action macro-3-actions-0-0-0–415-37	0	1	0
action macro-3-actions-0-2-0–50-27	0	1	0
action macro-3-actions-0-3-0–70-38	0	1	0
action macro-3-actions-0-4-0–417-38	1	0	0
action macro-4-actions-0-4-0-0–420-30	1	0	0
action macro-3-actions-0-3-4–370-59	1	0	0
action macro-4-actions-0-3-0-0–420-30	1	0	0
action macro-4-actions-0-3-0-0–421-30	1	0	0
action macro-4-actions-0-3-0-0–423-30	1	0	0
action macro-4-actions-0-3-0-0–424-30	1	0	0
action macro-3-actions-0-3-4–398-30	1	0	0
action macro-4-actions-0-3-0-0–425-30	1	0	0
action macro-3-actions-0-4-3–370-59	1	0	0
action macro-4-actions-0-4-0-0–423-30	1	0	0
action macro-4-actions-0-4-3-0–426-30	1	0	0
action macro-4-actions-0-3-0-0–427-30	1	0	0
action macro-3-actions-0-3-3–297-54	1	0	0
action macro-3-actions-0-3-1–430-62	1	0	0
action macro-3-actions-4-0-3–431-19	1	0	0
action macro-2-actions-4-0–41-19	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-4-3-4–432-23	0	1	0
action macro-3-actions-4-3-0–433-19	0	1	0
action macro-2-actions-4-3–42-22	0	1	0
action macro-4-actions-4-0-0-3–434-19	0	1	0
action macro-3-actions-4-0-0–36-19	0	1	0
action macro-2-actions-4-3–31-23	1	0	0
action macro-2-actions-4-0–435-66	1	0	0
action macro-2-actions-4-4–43-25	1	0	0
action macro-2-actions-4-2–436-23	1	0	0
action macro-3-actions-1-0-2–437-68	1	0	0
action macro-3-actions-1-0-0–438-7	1	0	0
action macro-4-actions-1-3-0-4–441-2	1	0	0
action macro-4-actions-1-3-0-4–441-68	1	0	0
action macro-4-actions-1-3-0-0–442-7	1	0	0
action macro-4-actions-1-3-4-0–444-69	1	0	0
action macro-4-actions-1-3-0-0–448-7	1	0	0
action macro-4-actions-1-3-0-0–449-7	1	0	0
action macro-4-actions-1-3-0-2–451-68	1	0	0
action macro-4-actions-1-3-0-0–452-7	1	0	0
action macro-4-actions-1-3-0-0–453-2	1	0	0
action macro-3-actions-1-3-2–454-70	1	0	0
action macro-4-actions-1-3-0-0–455-7	1	0	0
action macro-4-actions-1-3-0-0–456-7	1	0	0
action macro-4-actions-1-3-0-0–457-7	1	0	0
action macro-4-actions-1-3-0-0–459-7	1	0	0
action macro-4-actions-1-3-0-3–460-71	1	0	0
action macro-4-actions-1-3-4-0–461-72	1	0	0
action macro-3-actions-1-3-4–13-8	1	0	0
action macro-3-actions-1-3-3–4-3	1	0	0
action macro-3-actions-1-3-3–8-6	1	0	0
action macro-4-actions-1-3-3-0–463-73	1	0	0
action macro-3-actions-1-0-2–468-68	1	0	0
action macro-3-actions-1-0-0–469-7	1	0	0
action macro-3-actions-1-0-1–471-74	1	0	0
action macro-3-actions-1-0-2–471-75	1	0	0
action macro-2-actions-1-3–30-67	1	0	0
action macro-3-actions-1-0-4–19-1	1	0	0
action macro-3-actions-0-0-0–482-37	1	0	0
action macro-3-actions-0-4-0–491-38	1	0	0
action macro-3-actions-0-4-0–50-38	0	1	0
action macro-3-actions-0-4-3–493-38	0	1	0
action macro-3-actions-0-0-0–499-37	1	0	0
action macro-3-actions-0-0-0–501-37	0	1	0
action macro-3-actions-0-0-0–502-37	1	0	0
action macro-4-actions-0-0-0-0–503-37	0	1	0
action macro-4-actions-0-0-0-0–505-37	1	0	0
action macro-4-actions-0-0-0-0–506-37	1	0	0
action macro-4-actions-0-0-0-0–508-37	0	1	0

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-0-0-0–509-37	1	0	0
action macro-4-actions-0-0-0-0–510-37	1	0	0
action macro-3-actions-0-0-0–513-37	1	0	0
action macro-4-actions-0-0-0-0–516-37	0	1	0
action macro-3-actions-0-0-4–115-38	0	1	0
action macro-3-actions-0-0-3–115-38	0	1	0
action macro-4-actions-0-0-0-0–518-37	0	1	0
action macro-4-actions-0-0-0-0–519-37	0	1	0
action macro-4-actions-0-0-0-0–520-37	0	1	0
action macro-4-actions-0-0-3-0–521-38	1	0	0
action macro-3-actions-0-3-4–493-38	0	1	0
action macro-4-actions-0-0-0-0–524-37	0	1	0
action macro-4-actions-0-0-0-0–525-37	1	0	0
action macro-3-actions-0-0-0–528-37	1	0	0
action macro-3-actions-0-3-0–491-38	1	0	0
action macro-4-actions-0-0-0-0–529-47	1	0	0
action macro-4-actions-0-0-0-0–530-47	1	0	0
action macro-3-actions-0-4-0–70-38	0	1	0
action macro-3-actions-0-3-3–71-77	1	0	0
action macro-3-actions-0-3-0–417-38	1	0	0
action macro-3-actions-1-4-0–532-13	1	0	0
action macro-3-actions-3-3-0–534-84	1	0	0
action macro-2-actions-3-3–31-23	1	0	0
action macro-3-actions-3-4-0–540-20	1	0	0
action macro-3-actions-3-0-4–431-86	1	0	0
action macro-2-actions-3-4–43-25	1	0	0
action macro-3-actions-3-0-4–431-19	1	0	0
action macro-3-actions-3-4-2–541-65	0	1	0
action macro-4-actions-3-4-0-0–542-19	0	1	0
action macro-3-actions-3-4-0–433-19	0	1	0
action macro-3-actions-3-0-0–16-21	1	0	0
action macro-3-actions-3-0-0–17-21	0	1	0
action macro-3-actions-3-4-0–543-84	1	0	0
action macro-4-actions-3-0-0-4–434-19	0	1	0
action macro-3-actions-3-0-0–546-66	1	0	0
action macro-3-actions-3-0-2–548-87	1	0	0
action macro-3-actions-3-0-0–549-66	1	0	0
action macro-3-actions-3-4-2–550-88	1	0	0
action macro-2-actions-3-2–24-89	1	0	0
action macro-3-actions-3-0-0–551-66	1	0	0
action macro-4-actions-3-0-0-0–553-66	1	0	0
action macro-3-actions-3-0-0–554-66	1	0	0
action macro-2-actions-3-0–435-66	1	0	0
action macro-4-actions-3-0-0-0–556-66	0	1	0
action macro-3-actions-3-4-3–558-23	0	1	0
action macro-3-actions-3-3-4–559-25	1	0	0
action macro-3-actions-4-0-3–431-86	1	0	0
action macro-3-actions-4-0-0–560-19	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-4-0-3-0–561-19	1	0	0
action macro-3-actions-4-0-0–16-21	1	0	0
action macro-3-actions-4-0-0–17-21	0	1	0
action macro-4-actions-4-0-0-3–434-86	1	0	0
action macro-3-actions-4-2-2–563-63	1	0	0
action macro-3-actions-4-0-0–551-66	1	0	0
action macro-3-actions-4-0-0–554-66	1	0	0
action macro-3-actions-4-0-0–439-21	1	0	0
action macro-3-actions-4-0-0–25-21	0	1	0
action macro-3-actions-4-3-0–565-19	1	0	0
action macro-3-actions-4-3-3–432-23	1	0	0
action macro-2-actions-4-3–43-25	1	0	0
action macro-4-actions-0-0-1-4–568-26	1	0	0
action macro-4-actions-0-0-4-3–117-31	1	0	0
action macro-4-actions-0-0-0-3–571-45	1	0	0
action macro-4-actions-0-0-0-4–571-45	1	0	0
action macro-4-actions-0-0-0-0–574-47	1	0	0
action macro-4-actions-0-0-0-0–575-47	1	0	0
action macro-3-actions-0-0-0–579-49	1	0	0
action macro-4-actions-0-0-0-0–138-49	0	1	0
action macro-4-actions-0-0-3-0–584-30	1	0	0
action macro-4-actions-0-0-4-0–145-30	1	0	0
action macro-4-actions-0-0-4-0–584-30	1	0	0
action macro-4-actions-0-0-4-0–586-30	1	0	0
action macro-4-actions-0-0-0-0–587-37	0	1	0
action macro-4-actions-0-0-3-0–585-30	1	0	0
action macro-4-actions-0-0-4-0–585-30	1	0	0
action macro-4-actions-0-0-4-2–588-50	1	0	0
action macro-4-actions-0-0-3-4–589-54	1	0	0
action macro-3-actions-0-1-2–489-32	1	0	0
action macro-4-actions-0-0-0-0–595-37	1	0	0
action macro-3-actions-0-0-0–599-37	1	0	0
action macro-4-actions-0-0-0-0–602-44	1	0	0
action macro-4-actions-0-0-4-0–603-45	1	0	0
action macro-4-actions-0-0-0-0–607-44	1	0	0
action macro-4-actions-0-0-3-0–603-45	1	0	0
action macro-4-actions-0-0-0-0–608-44	1	0	0
action macro-4-actions-0-0-0-1–609-48	1	0	0
action macro-3-actions-0-0-2–610-48	1	0	0
action macro-3-actions-0-0-0–611-49	1	0	0
action macro-4-actions-0-0-4-0–612-45	1	0	0
action macro-3-actions-0-0-3–221-45	1	0	0
action macro-3-actions-0-0-0–616-47	1	0	0
action macro-4-actions-0-0-0-0–618-47	1	0	0
action macro-4-actions-0-0-0-0–619-47	1	0	0
action macro-4-actions-0-0-0-0–620-47	1	0	0
action macro-4-actions-0-0-0-0–621-47	1	0	0
action macro-4-actions-0-0-0-0–622-47	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-0-0-0-0–637-37	1	0	0
action macro-4-actions-0-0-0-0–638-37	1	0	0
action macro-4-actions-0-3-0-0–657-30	1	0	0
action macro-3-actions-0-0-0–659-37	0	1	0
action macro-3-actions-0-0-3–662-81	1	0	0
action macro-3-actions-0-4-0–368-30	1	0	0
action macro-4-actions-0-4-0-0–657-30	1	0	0
action macro-4-actions-0-0-0-0–670-37	1	0	0
action macro-4-actions-0-0-0-0–671-37	0	1	0
action macro-3-actions-0-0-0–672-37	1	0	0
action macro-4-actions-0-0-0-0–673-37	0	1	0
action macro-4-actions-0-3-0-0–675-30	1	0	0
action macro-2-actions-2-2–436-15	1	0	0
action macro-2-actions-2-1–42-93	0	1	0
action macro-4-actions-0-0-4-2–677-48	1	0	0
action macro-4-actions-0-0-0-3–196-45	1	0	0
action macro-4-actions-0-0-3-4–682-45	1	0	0
action macro-3-actions-0-0-1–169-48	1	0	0
action macro-4-actions-0-0-4-3–682-45	1	0	0
action macro-3-actions-0-1-0–684-33	1	0	0
action macro-4-actions-0-1-3-0–685-33	1	0	0
action macro-4-actions-0-1-3-0–686-33	1	0	0
action macro-4-actions-0-0-0-0–689-37	1	0	0
action macro-4-actions-0-0-0-0–690-37	1	0	0
action macro-3-actions-0-0-0–694-37	1	0	0
action macro-4-actions-0-4-0-0–425-30	1	0	0
action macro-4-actions-0-0-0-0–705-47	0	1	0
action macro-4-actions-0-0-0-0–707-47	0	1	0
action macro-4-actions-0-0-0-0–731-47	1	0	0
action macro-4-actions-0-0-1-0–732-48	1	0	0
action macro-4-actions-0-0-1-3–733-48	1	0	0
action macro-4-actions-0-0-1-0–734-48	1	0	0
action macro-3-actions-0-0-1–610-48	1	0	0
action macro-4-actions-0-0-0-4–737-45	1	0	0
action macro-4-actions-0-0-0-0–739-47	1	0	0
action macro-4-actions-0-0-0-0–740-47	1	0	0
action macro-4-actions-0-0-0-0–741-47	1	0	0
action macro-4-actions-0-0-0-0–742-47	1	0	0
action macro-4-actions-0-0-3-0–612-45	1	0	0
action macro-4-actions-0-0-3-4–743-58	1	0	0
action macro-4-actions-0-0-0-0–746-37	1	0	0
action macro-4-actions-0-3-3-0–95-94	1	0	0
action macro-4-actions-0-3-0-0–749-30	1	0	0
action macro-3-actions-0-1-0–50-33	1	0	0
action macro-4-actions-0-0-0-4–751-45	1	0	0
action macro-4-actions-0-0-0-3–751-45	1	0	0
action macro-4-actions-0-0-0-1–756-48	1	0	0
action macro-4-actions-0-0-3-0–586-30	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-0-0-0-0–761-37	0	1	0
action macro-4-actions-0-0-0-0–765-37	1	0	0
action macro-4-actions-0-0-0-0–766-37	0	1	0
action macro-4-actions-0-0-0-0–767-37	1	0	0
action macro-3-actions-0-0-0–771-37	1	0	0
action macro-4-actions-0-0-0-0–773-37	1	0	0
action macro-4-actions-0-0-0-0–774-37	1	0	0
action macro-4-actions-0-0-0-0–776-37	1	0	0
action macro-4-actions-0-0-0-0–777-37	1	0	0
action macro-4-actions-0-0-0-0–779-37	0	1	0
action macro-4-actions-0-0-0-0–781-37	1	0	0
action macro-4-actions-0-0-0-0–782-37	1	0	0
action macro-4-actions-0-3-0-2–391-39	1	0	0
action macro-4-actions-0-4-0-0–293-30	1	0	0
action macro-4-actions-0-4-0-0–749-30	1	0	0
action macro-4-actions-0-0-0-0–788-37	1	0	0
action macro-4-actions-0-0-0-0–791-37	1	0	0
action macro-4-actions-0-0-0-0–792-37	1	0	0
action macro-3-actions-0-0-0–796-37	1	0	0
action macro-4-actions-0-4-0-0–797-30	1	0	0
action macro-4-actions-0-4-0-3–798-30	1	0	0
action macro-4-actions-0-4-3-0–56-30	0	1	0
action macro-4-actions-0-0-0-0–803-37	1	0	0
action macro-4-actions-0-0-0-0–804-37	1	0	0
action macro-4-actions-0-0-0-0–807-37	1	0	0
action macro-4-actions-0-4-0-0–812-30	1	0	0
action macro-4-actions-0-4-0-4–813-43	1	0	0
action macro-3-actions-0-3-0–815-98	1	0	0
action macro-4-actions-1-3-1-3–562-99	1	0	0
action macro-4-actions-0-0-1-3–820-48	1	0	0
action macro-4-actions-0-0-0-0–823-47	0	1	0
action macro-4-actions-0-0-0-0–824-47	0	1	0
action macro-4-actions-0-0-0-1–825-48	0	1	0
action macro-4-actions-0-0-0-2–327-48	0	1	0
action macro-4-actions-0-0-0-0–826-47	0	1	0
action macro-4-actions-0-0-0-0–827-47	0	1	0
action macro-4-actions-0-0-0-0–828-47	0	1	0
action macro-4-actions-0-0-0-0–829-47	0	1	0
action macro-4-actions-0-0-0-0–832-37	0	1	0
action macro-4-actions-0-0-4-3–836-30	1	0	0
action macro-4-actions-0-0-0-0–837-44	1	0	0
action macro-4-actions-0-0-0-4–839-45	1	0	0
action macro-4-actions-0-0-0-3–839-45	1	0	0
action macro-4-actions-0-0-0-0–840-37	0	1	0
action macro-4-actions-0-0-1-0–841-33	1	0	0
action macro-4-actions-0-0-3-4–836-30	1	0	0
action macro-4-actions-0-0-0-0–846-37	1	0	0
action macro-4-actions-0-0-0-0–848-37	0	1	0

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-0-0-0-0–849-37	0	1	0
action macro-4-actions-0-0-0-0–850-37	0	1	0
action macro-4-actions-0-0-0-0–851-37	0	1	0
action macro-4-actions-0-0-0-3–737-45	1	0	0
action macro-4-actions-0-0-4-2–852-48	1	0	0
action macro-4-actions-0-0-3-4–853-45	1	0	0
action macro-4-actions-0-0-0-0–854-37	0	1	0
action macro-4-actions-0-0-0-0–855-37	1	0	0
action macro-4-actions-0-0-0-0–856-37	0	1	0
action macro-4-actions-0-0-0-0–857-37	1	0	0
action macro-4-actions-0-0-0-0–860-37	0	1	0
action macro-4-actions-0-0-0-0–861-37	1	0	0
action macro-4-actions-0-3-4-0–863-30	1	0	0
action macro-4-actions-0-4-0-2–865-50	1	0	0
action macro-4-actions-0-0-0-0–866-37	0	1	0
action macro-4-actions-0-0-0-0–867-37	0	1	0
action macro-4-actions-0-0-0-0–868-37	1	0	0
action macro-4-actions-0-3-0-4–798-30	1	0	0
action macro-4-actions-0-3-3-4–871-54	1	0	0
action macro-4-actions-0-0-0-0–874-37	0	1	0
action macro-4-actions-0-0-0-0–887-47	1	0	0
action macro-4-actions-0-0-0-0–894-37	1	0	0
action macro-4-actions-0-0-1-0–895-33	1	0	0
action macro-3-actions-0-0-4–169-57	1	0	0
action macro-3-actions-0-3-3–100-41	1	0	0
action macro-4-actions-0-3-4-0–56-30	0	1	0
action macro-4-actions-0-0-0-0–903-37	1	0	0
action macro-3-actions-0-0-4–662-81	1	0	0
action macro-4-actions-0-0-0-0–904-37	1	0	0
action macro-4-actions-0-0-0-0–906-37	1	0	0
action macro-4-actions-0-0-0-0–909-37	0	1	0
action macro-4-actions-0-3-0-0–797-30	1	0	0
action macro-4-actions-0-3-0-0–812-30	1	0	0
action macro-4-actions-0-4-3-0–911-30	1	0	0
action macro-3-actions-3-0-0–560-19	1	0	0
action macro-3-actions-3-0-0–914-66	1	0	0
action macro-4-actions-0-0-0-0–915-37	1	0	0
action macro-3-actions-0-4-3–408-38	0	1	0
action macro-4-actions-0-0-0-0–919-37	1	0	0
action macro-4-actions-0-0-0-0–922-37	0	1	0
action macro-4-actions-0-0-0-0–928-37	0	1	0
action macro-4-actions-0-0-0-0–929-37	1	0	0
action macro-4-actions-0-0-0-0–930-37	1	0	0
action macro-3-actions-0-3-3–489-77	1	0	0
action macro-3-actions-0-3-0–50-38	0	1	0
action macro-3-actions-0-3-4–932-76	1	0	0
action macro-2-actions-1-3–819-100	1	0	0
action macro-3-actions-1-3-0–939-101	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-1-3-1-0–940-102	1	0	0
action macro-4-actions-1-3-1-0–942-102	1	0	0
action macro-3-actions-1-3-1–12-99	1	0	0
action macro-4-actions-1-3-3-0–446-101	1	0	0
action macro-3-actions-1-3-3–8-100	1	0	0
action macro-3-actions-3-0-3–431-86	1	0	0
action macro-3-actions-3-1-0–943-83	1	0	0
action macro-3-actions-3-3-4–558-23	1	0	0
action macro-4-actions-3-3-0-0–947-84	1	0	0
action macro-4-actions-3-0-0-3–434-86	1	0	0
action macro-3-actions-3-0-1–948-104	1	0	0
action macro-3-actions-3-0-0–949-66	1	0	0
action macro-3-actions-3-0-1–950-104	1	0	0
action macro-3-actions-3-0-0–951-66	1	0	0
action macro-3-actions-3-0-0–952-66	1	0	0
action macro-3-actions-3-0-2–953-16	1	0	0
action macro-3-actions-3-4-0–540-107	1	0	0
action macro-3-actions-1-4-0–532-108	1	0	0
action macro-4-actions-1-0-0-0–954-7	0	1	0
action macro-3-actions-1-0-2–955-75	1	0	0
action macro-4-actions-0-0-4-3–853-45	1	0	0
action macro-4-actions-0-0-3-0–222-45	1	0	0
action macro-4-actions-0-0-3-4–956-45	1	0	0
action macro-4-actions-0-0-0-0–959-37	1	0	0
action macro-4-actions-0-0-3-2–156-39	1	0	0
action macro-4-actions-0-0-4-4–162-43	1	0	0
action macro-4-actions-0-0-0-2–825-48	0	1	0
action macro-4-actions-0-0-0-0–969-47	1	0	0
action macro-3-actions-0-0-4–170-56	1	0	0
action macro-4-actions-0-0-0-0–972-47	1	0	0
action macro-4-actions-0-0-0-0–974-47	1	0	0
action macro-4-actions-0-0-0-0–976-37	1	0	0
action macro-3-actions-0-1-4–567-61	1	0	0
action macro-4-actions-0-1-0-4–977-109	1	0	0
action macro-4-actions-0-0-0-0–983-37	1	0	0
action macro-3-actions-0-0-0–218-49	1	0	0
action macro-4-actions-0-0-0-0–985-37	0	1	0
action macro-4-actions-0-0-0-0–988-37	1	0	0
action macro-4-actions-0-0-0-0–989-37	0	1	0
action macro-3-actions-0-3-0–295-111	1	0	0
action macro-4-actions-0-4-0-4–910-59	1	0	0
action macro-4-actions-0-4-0-3–910-59	1	0	0
action macro-4-actions-0-4-3-4–991-41	1	0	0
action macro-4-actions-0-3-0-4–910-59	1	0	0
action macro-4-actions-0-3-4-0–990-41	1	0	0
action macro-4-actions-3-0-0-0–992-66	1	0	0
action macro-3-actions-3-0-3–993-112	1	0	0
action macro-3-actions-3-4-2–994-113	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-3-4-0–995-20	1	0	0
action macro-3-actions-3-4-3–996-20	1	0	0
action macro-3-actions-3-0-3–34-17	1	0	0
action macro-3-actions-3-4-0–997-114	1	0	0
action macro-4-actions-3-0-0-0–1001-66	1	0	0
action macro-4-actions-3-0-0-0–1002-66	1	0	0
action macro-3-actions-3-0-0–1005-66	1	0	0
action macro-2-actions-3-3–40-20	1	0	0
action macro-3-actions-3-1-4–1007-116	1	0	0
action macro-4-actions-3-0-4-0–561-19	1	0	0
action macro-3-actions-3-0-0–1008-117	1	0	0
action macro-3-actions-3-0-0–25-21	0	1	0
action macro-4-actions-0-0-0-0–1009-37	0	1	0
action macro-3-actions-1-3-0–465-118	1	0	0
action macro-4-actions-1-3-1-0–942-119	1	0	0
action macro-3-actions-1-3-1–12-120	1	0	0
action macro-4-actions-1-3-3-0–446-69	1	0	0
action macro-4-actions-1-3-3-3–445-121	1	0	0
action macro-4-actions-1-3-3-4–1010-122	1	0	0
action macro-4-actions-1-3-3-4–445-100	1	0	0
action macro-4-actions-1-3-1-3–1011-121	1	0	0
action macro-4-actions-1-3-1-0–1013-123	1	0	0
action macro-3-actions-1-3-1–936-124	1	0	0
action macro-3-actions-1-3-4–938-121	1	0	0
action macro-4-actions-1-3-0-0–1014-101	1	0	0
action macro-4-actions-1-3-0-4–1015-125	1	0	0
action macro-4-actions-1-3-3-0–1016-126	1	0	0
action macro-3-actions-1-3-3–938-121	1	0	0
action macro-3-actions-1-3-4–1017-122	1	0	0
action macro-2-actions-1-2–15-127	1	0	0
action macro-3-actions-3-0-0–1018-18	1	0	0
action macro-3-actions-3-3-4–544-23	1	0	0
action macro-4-actions-3-0-0-0–1021-66	1	0	0
action macro-3-actions-3-0-4–993-112	1	0	0
action macro-3-actions-3-3-3–544-128	1	0	0
action macro-3-actions-3-3-4–544-128	1	0	0
action macro-4-actions-3-3-1-3–1023-129	1	0	0
action macro-4-actions-3-3-1-0–1024-130	1	0	0
action macro-3-actions-3-3-1–1025-129	1	0	0
action macro-4-actions-3-3-0-4–1026-131	1	0	0
action macro-3-actions-3-3-4–1027-132	1	0	0
action macro-4-actions-3-3-4-0–1028-84	1	0	0
action macro-3-actions-3-1-3–1022-128	1	0	0
action macro-4-actions-3-1-3-3–944-128	1	0	0
action macro-3-actions-3-0-0–1031-18	1	0	0
action macro-4-actions-3-0-0-0–1032-66	1	0	0
action macro-4-actions-3-0-4-0–561-86	1	0	0
action macro-3-actions-3-0-0–439-21	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-3-1-3-1–1042-129	1	0	0
action macro-4-actions-3-3-3-0–1028-136	1	0	0
action macro-3-actions-3-4-0–565-19	1	0	0
action macro-3-actions-3-4-3–432-23	1	0	0
action macro-3-actions-1-0-4–19-125	1	0	0
action macro-3-actions-1-3-4–1044-100	1	0	0
action macro-4-actions-1-3-3-4–462-100	1	0	0
action macro-4-actions-1-3-3-4–445-121	1	0	0
action macro-2-actions-1-2–30-137	1	0	0
action macro-4-actions-4-0-0-0–556-66	0	1	0
action macro-3-actions-4-0-0–914-66	1	0	0
action macro-4-actions-4-3-0-0–542-19	0	1	0
action macro-3-actions-4-4-0–997-25	1	0	0
action macro-3-actions-4-0-0–1018-18	1	0	0
action macro-3-actions-4-0-0–1031-18	1	0	0
action macro-3-actions-4-0-0–546-66	1	0	0
action macro-3-actions-4-3-0–534-84	1	0	0
action macro-3-actions-4-3-3–544-23	1	0	0
action macro-3-actions-4-3-1–1046-24	0	1	0
action macro-3-actions-4-4-2–994-113	1	0	0
action macro-3-actions-4-2-0–5-64	1	0	0
action macro-3-actions-1-0-0–1047-7	1	0	0
action macro-3-actions-4-0-0–952-66	1	0	0
action macro-4-actions-3-0-0-4–1049-17	1	0	0
action macro-3-actions-3-1-2–475-103	1	0	0
action macro-4-actions-3-0-0-0–1050-21	1	0	0
action macro-4-actions-3-0-0-0–1051-21	1	0	0
action macro-4-actions-3-0-0-0–1053-21	1	0	0
action macro-4-actions-3-3-0-0–1058-84	1	0	0
action macro-3-actions-3-3-0–543-84	1	0	0
action macro-3-actions-3-3-0–1036-135	1	0	0
action macro-4-actions-3-3-4-0–1060-84	1	0	0
action macro-2-actions-3-3–30-139	1	0	0
action macro-3-actions-4-0-0–951-66	1	0	0
action macro-3-actions-4-0-0–949-66	1	0	0
action macro-4-actions-4-0-0-0–1021-66	1	0	0
action macro-3-actions-4-3-0–997-114	1	0	0
action macro-3-actions-2-0-0–17-2	0	1	0
action macro-4-actions-0-0-0-0–1067-37	1	0	0
action macro-3-actions-0-3-0–50-79	1	0	0
action macro-4-actions-0-0-0-0–1068-47	1	0	0
action macro-3-actions-0-0-0–1071-49	1	0	0
action macro-4-actions-0-0-0-4–1073-55	1	0	0
action macro-3-actions-4-3-4–544-128	1	0	0
action macro-4-actions-4-3-0-0–947-84	1	0	0
action macro-3-actions-4-3-4–558-23	1	0	0
action macro-4-actions-0-0-0-0–1076-37	0	1	0
action macro-4-actions-0-0-0-0–1079-37	0	1	0

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-1-3-3-0–444-101	1	0	0
action macro-3-actions-3-4-0–995-107	1	0	0
action macro-3-actions-3-4-0–1081-107	1	0	0
action macro-4-actions-3-4-0-0–1082-20	1	0	0
action macro-3-actions-3-0-4–1083-141	1	0	0
action macro-3-actions-3-0-1–1048-87	1	0	0
action macro-4-actions-3-0-0-0–1085-66	1	0	0
action macro-3-actions-3-3-0–997-25	1	0	0
action macro-4-actions-3-0-3-0–561-86	1	0	0
action macro-4-actions-3-3-0-4–1086-84	1	0	0
action macro-3-actions-3-4-0–1087-21	0	1	0
action macro-3-actions-3-4-0–1088-21	0	1	0
action macro-3-actions-2-0-0–25-2	0	1	0
action macro-4-actions-3-0-0-0–1090-18	1	0	0
action macro-4-actions-3-0-0-0–1091-18	1	0	0
action macro-4-actions-3-0-0-0–1092-18	1	0	0
action macro-4-actions-3-0-0-0–1093-18	1	0	0
action macro-4-actions-3-0-0-4–1049-90	1	0	0
action macro-4-actions-3-0-0-0–1094-18	1	0	0
action macro-4-actions-3-0-0-0–1095-18	1	0	0
action macro-4-actions-3-0-0-0–1097-21	1	0	0
action macro-4-actions-3-0-0-0–1098-21	1	0	0
action macro-4-actions-3-0-0-0–1099-21	1	0	0
action macro-4-actions-3-1-3-3–1103-142	1	0	0
action macro-4-actions-3-3-3-4–1104-128	1	0	0
action macro-3-actions-4-0-4–993-112	1	0	0
action macro-4-actions-4-0-3-0–1105-141	1	0	0
action macro-4-actions-4-3-0-0–1106-19	1	0	0
action macro-3-actions-4-3-4–1107-20	1	0	0
action macro-3-actions-1-0-3–28-68	1	0	0
action macro-4-actions-1-0-0-3–466-68	1	0	0
action macro-4-actions-1-3-0-3–441-68	1	0	0
action macro-4-actions-1-3-0-3–1108-143	1	0	0
action macro-3-actions-1-3-0–1109-101	1	0	0
action macro-3-actions-1-3-4–938-100	1	0	0
action macro-4-actions-4-0-0-0–992-66	1	0	0
action macro-3-actions-4-0-3–1110-66	1	0	0
action macro-3-actions-4-4-3–559-25	1	0	0
action macro-3-actions-4-3-0–997-25	1	0	0
action macro-3-actions-4-3-0–543-84	1	0	0
action macro-3-actions-4-0-0–1005-66	1	0	0
action macro-3-actions-4-0-0–549-66	1	0	0
action macro-4-actions-4-3-0-0–1111-66	1	0	0
action macro-4-actions-4-3-0-0–1112-66	1	0	0
action macro-3-actions-4-3-3–1113-25	1	0	0
action macro-4-actions-4-0-0-0–1002-66	1	0	0
action macro-4-actions-3-0-0-0–1114-66	1	0	0
action macro-4-actions-3-0-0-0–1115-66	1	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-3-0-4–1116-141	1	0	0
action macro-3-actions-3-4-0–997-25	1	0	0
action macro-4-actions-3-0-0-0–1117-66	1	0	0
action macro-4-actions-3-0-0-3–1049-17	1	0	0
action macro-3-actions-3-4-0–1119-25	1	0	0
action macro-3-actions-3-4-3–559-25	1	0	0
action macro-3-actions-3-3-0–1120-144	1	0	0
action macro-4-actions-1-3-0-4–1-1	0	0	0
action macro-4-actions-1-3-0-0–2-2	0	0	1
action macro-3-actions-1-3-0–3-2	0	0	0
action macro-3-actions-1-3-0–5-4	0	0	1
action macro-3-actions-1-3-4–8-6	0	0	0
action macro-3-actions-1-3-0–10-4	0	0	1
action macro-4-actions-1-3-0-0–11-7	0	0	1
action macro-2-actions-1-3–15-5	0	0	0
action macro-3-actions-1-0-0–16-10	0	0	1
action macro-3-actions-1-0-0–18-10	0	0	1
action macro-3-actions-1-0-3–19-11	0	0	0
action macro-3-actions-1-0-0–20-7	0	0	1
action macro-2-actions-1-0–22-2	0	0	1
action macro-2-actions-1-0–26-2	0	0	1
action macro-4-actions-1-0-3-0–27-2	0	0	1
action macro-3-actions-1-0-3–28-2	0	0	1
action macro-4-actions-3-0-4-2–32-16	0	0	0
action macro-4-actions-3-0-4-0–33-17	0	0	0
action macro-3-actions-3-0-4–34-17	0	0	0
action macro-3-actions-3-0-0–35-18	0	0	1
action macro-3-actions-3-0-0–37-18	0	0	1
action macro-2-actions-3-0–39-19	0	0	0
action macro-2-actions-3-0–26-21	0	0	1
action macro-2-actions-3-0–22-21	0	0	1
action macro-2-actions-3-4–31-23	0	0	0
action macro-3-actions-0-1-0–51-28	0	0	1
action macro-2-actions-0-1–52-27	0	0	0
action macro-3-actions-0-4-2–55-27	0	0	0
action macro-3-actions-0-4-0–59-29	0	0	0
action macro-3-actions-0-4-0–59-30	0	0	0
action macro-2-actions-0-4–60-29	0	0	0
action macro-2-actions-0-2–61-27	0	0	1
action macro-4-actions-0-1-3-4–62-27	0	0	1
action macro-4-actions-0-1-3-4–63-32	0	0	0
action macro-4-actions-0-1-3-0–64-27	0	0	0
action macro-4-actions-0-1-3-0–64-33	0	0	1
action macro-3-actions-0-1-3–67-27	0	0	0
action macro-3-actions-0-1-0–70-33	0	0	1
action macro-2-actions-0-1–61-27	0	0	0
action macro-3-actions-0-0-4–75-34	0	0	1
action macro-4-actions-0-0-0-4–76-34	0	0	1

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-0-0-0-0–77-36	0	0	1
action macro-4-actions-0-0-0-0–81-37	0	0	1
action macro-3-actions-0-0-0–82-37	0	0	1
action macro-4-actions-0-0-0-4–83-34	0	0	1
action macro-4-actions-0-0-0-0–85-36	0	0	1
action macro-4-actions-0-0-0-0–88-36	0	0	1
action macro-3-actions-0-0-0–78-36	0	0	1
action macro-3-actions-0-0-3–75-34	0	0	1
action macro-3-actions-0-0-0–89-37	0	0	1
action macro-4-actions-0-0-0-0–90-36	0	0	1
action macro-3-actions-0-0-0–91-36	0	0	1
action macro-2-actions-0-0–93-35	0	0	0
action macro-3-actions-0-3-0–94-38	0	0	1
action macro-4-actions-0-3-4-0–95-31	0	0	0
action macro-4-actions-0-3-4-2–96-39	0	0	0
action macro-3-actions-0-3-4–58-31	0	0	0
action macro-4-actions-0-3-0-4–98-40	0	0	0
action macro-3-actions-0-3-0–59-29	0	0	0
action macro-3-actions-0-3-0–102-38	0	0	1
action macro-4-actions-0-3-0-0–103-30	0	0	1
action macro-3-actions-0-3-0–59-30	0	0	0
action macro-2-actions-0-3–60-29	0	0	0
action macro-3-actions-0-4-0–94-38	0	0	1
action macro-4-actions-0-4-0-0–105-38	0	0	1
action macro-4-actions-0-4-0-4–98-42	0	0	0
action macro-4-actions-0-4-0-0–106-38	0	0	1
action macro-4-actions-0-4-0-0–108-38	0	0	1
action macro-4-actions-0-4-0-0–109-38	0	0	1
action macro-3-actions-0-4-0–102-38	0	0	1
action macro-4-actions-0-4-2-0–111-27	0	0	1
action macro-3-actions-0-4-3–55-38	0	0	0
action macro-4-actions-0-0-1-3–113-27	0	0	1
action macro-4-actions-0-0-3-4–117-31	0	0	1
action macro-4-actions-0-0-0-3–123-34	0	0	1
action macro-4-actions-0-0-0-0–126-36	0	0	1
action macro-4-actions-0-0-0-4–123-34	0	0	1
action macro-4-actions-0-0-0-0–127-36	0	0	1
action macro-4-actions-0-0-0-4–141-45	0	0	1
action macro-4-actions-0-0-0-3–141-45	0	0	1
action macro-4-actions-0-0-0-3–146-34	0	0	1
action macro-4-actions-0-0-0-0–149-36	0	0	1
action macro-4-actions-0-0-0-0–150-36	0	0	1
action macro-4-actions-0-0-0-3–151-34	0	0	1
action macro-4-actions-0-0-0-4–146-34	0	0	1
action macro-4-actions-0-0-3-1–156-39	0	0	1
action macro-4-actions-0-0-4-0–160-38	0	0	1
action macro-4-actions-0-0-4-2–161-27	0	0	1
action macro-3-actions-0-2-4–69-32	0	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-2-actions-0-2–52-27	0	0	1
action macro-2-actions-0-0–261-37	0	0	0
action macro-3-actions-0-0-1–262-51	0	0	0
action macro-3-actions-0-0-4–263-34	0	0	1
action macro-3-actions-0-0-1–263-51	0	0	1
action macro-4-actions-0-0-0-4–264-34	0	0	1
action macro-4-actions-0-0-0-0–265-36	0	0	1
action macro-4-actions-0-0-0-3–264-34	0	0	1
action macro-4-actions-0-0-0-0–267-36	0	0	1
action macro-3-actions-0-0-3–263-34	0	0	1
action macro-4-actions-0-0-4-0–269-34	0	0	1
action macro-4-actions-0-0-4-0–270-34	0	0	1
action macro-4-actions-0-0-0-4–272-34	0	0	1
action macro-4-actions-0-0-0-0–273-36	0	0	1
action macro-4-actions-0-0-0-0–277-36	0	0	1
action macro-3-actions-0-0-0–274-36	0	0	0
action macro-3-actions-0-0-0–278-37	0	0	1
action macro-3-actions-0-0-4–279-34	0	0	0
action macro-4-actions-0-0-3-0–269-34	0	0	1
action macro-3-actions-0-0-0–280-37	0	0	1
action macro-4-actions-0-0-3-0–270-34	0	0	1
action macro-4-actions-0-0-0-0–281-36	0	0	1
action macro-4-actions-0-0-0-0–282-36	0	0	1
action macro-3-actions-0-0-0–283-36	0	0	0
action macro-3-actions-0-0-2–263-51	0	0	0
action macro-3-actions-0-0-3–279-34	0	0	0
action macro-3-actions-0-0-3–263-52	0	0	0
action macro-2-actions-0-0–286-35	0	0	0
action macro-2-actions-0-3–61-38	0	0	0
action macro-4-actions-0-3-1-3–290-39	0	0	0
action macro-4-actions-0-3-1-0–291-39	0	0	1
action macro-3-actions-0-3-1–99-39	0	0	0
action macro-4-actions-0-3-4-2–292-27	0	0	1
action macro-4-actions-0-3-0-0–105-38	0	0	1
action macro-4-actions-0-3-0-4–98-42	0	0	1
action macro-4-actions-0-3-0-0–294-30	0	0	1
action macro-4-actions-0-3-0-4–98-53	0	0	0
action macro-3-actions-0-0-1–305-48	0	0	1
action macro-3-actions-0-0-0–306-47	0	0	1
action macro-3-actions-0-0-2–305-48	0	0	1
action macro-4-actions-0-0-0-4–311-45	0	0	1
action macro-4-actions-0-0-0-3–311-45	0	0	1
action macro-4-actions-0-0-0-1–316-48	0	0	1
action macro-4-actions-0-0-0-0–318-47	0	0	1
action macro-3-actions-0-0-4–339-45	0	0	1
action macro-3-actions-0-0-2–340-48	0	0	1
action macro-3-actions-0-0-0–341-47	0	0	1
action macro-4-actions-0-0-1-3–343-48	0	0	1

Table C.3 continued from previous page

macro	I	U	R
action macro-3-actions-0-0-1–340-48	0	0	1
action macro-4-actions-0-0-3-0–344-45	0	0	1
action macro-3-actions-0-0-3–339-45	0	0	1
action macro-3-actions-0-0-0–346-47	0	0	1
action macro-4-actions-0-0-0-0–347-47	0	0	1
action macro-3-actions-0-0-0–349-47	0	0	1
action macro-4-actions-0-0-4-0–344-45	0	0	1
action macro-4-actions-0-0-4-2–350-48	0	0	1
action macro-3-actions-0-0-0–357-47	0	0	1
action macro-4-actions-0-0-0-0–360-47	0	0	1
action macro-3-actions-0-0-0–361-47	0	0	1
action macro-3-actions-0-0-4–340-57	0	0	1
action macro-4-actions-0-0-3-4–363-58	0	0	1
action macro-3-actions-0-0-0–375-37	0	0	1
action macro-3-actions-0-0-0–376-37	0	0	1
action macro-3-actions-0-0-0–377-37	0	0	1
action macro-3-actions-0-0-0–383-37	0	0	1
action macro-2-actions-0-0–387-37	0	0	0
action macro-3-actions-0-4-0–388-38	0	0	1
action macro-2-actions-0-4–61-38	0	0	1
action macro-4-actions-0-3-0-1–391-39	0	0	0
action macro-4-actions-0-3-0-0–106-38	0	0	1
action macro-4-actions-0-3-0-0–108-38	0	0	1
action macro-3-actions-0-0-0–396-37	0	0	1
action macro-2-actions-0-0–397-37	0	0	1
action macro-3-actions-0-1-0–70-28	0	0	1
action macro-4-actions-0-4-0-0–400-38	0	0	1
action macro-4-actions-0-4-0-0–103-30	0	0	1
action macro-4-actions-0-4-0-0–294-30	0	0	1
action macro-4-actions-0-4-0-4–98-59	0	0	0
action macro-4-actions-0-4-2-4–401-32	0	0	0
action macro-4-actions-0-3-0-0–402-38	0	0	1
action macro-4-actions-0-3-0-0–400-38	0	0	1
action macro-4-actions-0-4-0-2–391-50	0	0	0
action macro-4-actions-0-0-0-0–271-36	0	0	1
action macro-3-actions-0-0-0–405-37	0	0	1
action macro-3-actions-0-0-0–406-37	0	0	1
action macro-2-actions-0-0–24-37	0	0	0
action macro-3-actions-0-3-0–388-38	0	0	1
action macro-4-actions-0-0-0-0–409-47	0	0	1
action macro-4-actions-0-4-3-4–394-31	0	0	1
action macro-3-actions-0-0-0–413-37	0	0	1
action macro-3-actions-4-2-4–12-63	0	0	0
action macro-3-actions-4-2-0–10-64	0	0	1
action macro-2-actions-4-2–15-65	0	0	0
action macro-2-actions-4-0–22-21	0	0	1
action macro-2-actions-4-0–39-19	0	0	0
action macro-2-actions-4-0–26-21	0	0	0

Table C.3 continued from previous page

macro	I	U	R
action macro-2-actions-2-4–30-67	0	0	0
action macro-2-actions-2-0–22-2	0	0	0
action macro-3-actions-1-0-0–439-10	0	0	1
action macro-3-actions-1-0-0–440-7	0	0	0
action macro-4-actions-1-3-4-0–443-2	0	0	1
action macro-4-actions-1-3-4-0–446-69	0	0	0
action macro-4-actions-1-3-4-2–447-6	0	0	0
action macro-4-actions-1-3-0-3–1-11	0	0	0
action macro-4-actions-1-3-4-3–462-6	0	0	1
action macro-4-actions-1-0-0-0–467-10	0	0	1
action macro-4-actions-1-0-0-0–470-7	0	0	1
action macro-4-actions-1-0-3-0–27-7	0	0	0
action macro-4-actions-1-0-0-0–472-7	0	0	1
action macro-3-actions-0-1-3–67-28	0	0	0
action macro-3-actions-0-0-4–477-34	0	0	0
action macro-3-actions-0-0-1–74-51	0	0	1
action macro-4-actions-0-0-3-0–478-34	0	0	1
action macro-3-actions-0-0-3–477-34	0	0	0
action macro-4-actions-0-0-4-0–478-34	0	0	0
action macro-4-actions-0-0-4-0–479-34	0	0	1
action macro-4-actions-0-0-0-3–480-34	0	0	1
action macro-4-actions-0-0-0-0–84-36	0	0	1
action macro-3-actions-0-0-2–74-51	0	0	1
action macro-4-actions-0-0-0-4–483-34	0	0	1
action macro-4-actions-0-0-0-3–483-34	0	0	1
action macro-4-actions-0-0-0-0–484-36	0	0	1
action macro-2-actions-0-4–52-38	0	0	0
action macro-3-actions-0-1-3–48-28	0	0	0
action macro-4-actions-0-3-0-4–486-76	0	0	1
action macro-4-actions-0-3-0-0–487-38	0	0	1
action macro-4-actions-0-3-0-0–488-38	0	0	1
action macro-3-actions-0-3-4–489-77	0	0	1
action macro-3-actions-0-3-0–490-38	0	0	1
action macro-2-actions-0-3–52-38	0	0	0
action macro-4-actions-0-4-0-0–492-38	0	0	1
action macro-3-actions-0-4-0–490-38	0	0	1
action macro-3-actions-0-4-2–48-78	0	0	0
action macro-4-actions-0-0-0-4–494-34	0	0	1
action macro-4-actions-0-0-0-3–494-34	0	0	1
action macro-4-actions-0-0-3-0–495-34	0	0	0
action macro-4-actions-0-0-0-4–497-34	0	0	1
action macro-3-actions-0-0-2–262-51	0	0	0
action macro-4-actions-0-0-0-0–504-37	0	0	1
action macro-4-actions-0-0-0-0–512-37	0	0	1
action macro-4-actions-0-0-0-3–515-34	0	0	1
action macro-4-actions-0-0-0-4–515-34	0	0	1
action macro-4-actions-0-0-4-0–517-38	0	0	1
action macro-4-actions-0-0-0-4–151-34	0	0	1

Table C.3 continued from previous page

macro	I	U	R
action macro-4-actions-0-0-3-0–517-38	0	0	1
action macro-4-actions-0-0-3-0–522-38	0	0	1
action macro-4-actions-0-0-4-0–523-38	0	0	1
action macro-3-actions-0-0-4–262-52	0	0	0
action macro-3-actions-0-3-0–490-79	0	0	0
action macro-3-actions-0-0-1–526-80	0	0	0
action macro-3-actions-0-0-4–527-81	0	0	0
action macro-3-actions-0-3-1–485-82	0	0	1
action macro-4-actions-0-3-0-0–492-38	0	0	1
action macro-3-actions-0-0-4–74-52	0	0	0
action macro-4-actions-0-4-0-0–487-38	0	0	1
action macro-4-actions-3-1-3-0–533-83	0	0	0
action macro-3-actions-3-1-3–464-24	0	0	0
action macro-4-actions-3-0-4-0–535-85	0	0	0
action macro-4-actions-3-0-4-0–536-85	0	0	0
action macro-4-actions-3-0-0-0–537-18	0	0	1
action macro-4-actions-3-0-0-0–538-18	0	0	1
action macro-4-actions-3-0-0-0–539-18	0	0	1
action macro-3-actions-3-4-0–534-84	0	0	0
action macro-3-actions-3-4-2–545-24	0	0	0
action macro-4-actions-3-0-0-4–434-86	0	0	1
action macro-3-actions-3-0-0–547-66	0	0	1
action macro-4-actions-3-0-0-0–557-66	0	0	1
action macro-3-actions-3-0-4–34-90	0	0	0
action macro-3-actions-3-0-3–34-90	0	0	0
action macro-3-actions-4-0-0–18-21	0	0	1
action macro-3-actions-4-0-2–548-91	0	0	0
action macro-3-actions-4-0-4–34-90	0	0	0
action macro-4-actions-4-2-4-2–562-63	0	0	0
action macro-3-actions-4-0-0–564-21	0	0	1
action macro-4-actions-4-3-0-4–566-17	0	0	1
action macro-4-actions-4-0-0-0–557-66	0	0	1
action macro-3-actions-4-0-0–547-66	0	0	1
action macro-3-actions-4-0-3–34-90	0	0	0
action macro-2-actions-4-3–15-92	0	0	0
action macro-4-actions-0-0-4-0–570-30	0	0	1
action macro-4-actions-0-0-0-1–124-51	0	0	1
action macro-4-actions-0-0-3-0–570-30	0	0	1
action macro-4-actions-0-0-0-1–582-48	0	0	1
action macro-4-actions-0-0-0-0–583-47	0	0	1
action macro-4-actions-0-0-0-2–582-48	0	0	1
action macro-4-actions-0-0-3-0–160-38	0	0	1
action macro-4-actions-0-1-0-3–590-27	0	0	1
action macro-4-actions-0-0-0-3–272-34	0	0	1
action macro-4-actions-0-0-0-0–596-37	0	0	1

Table C.3: Detail of the found macro-operators for `depots` domain.

C.4 Satellite

C.4.1 Predicate incompatibilities

pointing ?satellite1 ?direction0 | pointing ?satellite1 ?direction2, pointing ?satellite1 ?direction3, pointing ?satellite1 ?direction1
power_avail ?satellite0 |
pointing ?satellite0 ?direction1 | pointing ?satellite0 ?direction3, pointing ?satellite0 ?direction0, pointing ?satellite0 ?direction2
on_board ?instrument0 ?satellite1 |
pointing ?satellite1 ?direction1 | pointing ?satellite1 ?direction2, pointing ?satellite1 ?direction0, pointing ?satellite1 ?direction3
pointing ?satellite1 ?direction2 | pointing ?satellite1 ?direction0, pointing ?satellite1 ?direction3, pointing ?satellite1 ?direction1
on_board ?instrument1 ?satellite1 |
on_board ?instrument0 ?satellite0 |
power_avail ?satellite1 |
pointing ?satellite0 ?direction0 | pointing ?satellite0 ?direction1, pointing ?satellite0 ?direction3, pointing ?satellite0 ?direction2
on_board ?instrument1 ?satellite0 |
pointing ?satellite0 ?direction2 | pointing ?satellite0 ?direction1, pointing ?satellite0 ?direction3, pointing ?satellite0 ?direction0
calibration_target ?instrument1 ?direction0 |
calibration_target ?instrument1 ?direction1 |
calibration_target ?instrument0 ?direction3 |
pointing ?satellite0 ?direction3 | pointing ?satellite0 ?direction1, pointing ?satellite0 ?direction0, pointing ?satellite0 ?direction2
calibration_target ?instrument1 ?direction2 |
pointing ?satellite1 ?direction3 | pointing ?satellite1 ?direction2, pointing ?satellite1 ?direction1, pointing ?satellite1 ?direction0
calibration_target ?instrument0 ?direction0 |
calibration_target ?instrument1 ?direction3 |
calibration_target ?instrument0 ?direction2 |
calibration_target ?instrument0 ?direction1 |
supports ?instrument0 ?mode1 |
supports ?instrument0 ?mode0 |
supports ?instrument1 ?mode1 |
supports ?instrument1 ?mode0 |
power_on ?instrument1 |
calibration_target ?instrument2 ?direction0 |
on_board ?instrument1 ?satellite2 |
pointing ?satellite2 ?direction2 | pointing ?satellite2 ?direction1, pointing ?satellite2 ?direction0
on_board ?instrument0 ?satellite2 |
power_on ?instrument0 |
calibration_target ?instrument2 ?direction2 |
pointing ?satellite2 ?direction0 | pointing ?satellite2 ?direction1, pointing ?satellite2

?direction2
power_avail ?satellite2 |
power_on ?instrument2 |
calibration_target ?instrument2 ?direction1 |
on_board ?instrument2 ?satellite1 |
pointing ?satellite2 ?direction1 | pointing ?satellite2 ?direction0, pointing ?satellite2 ?direction2
on_board ?instrument2 ?satellite2 |
on_board ?instrument2 ?satellite0 |

C.4.2 Understanding the found macro-operators

macro	I	U	R
action macro-3-actions-1-3-0–4-2	1	0	0
action macro-3-actions-1-3-4–5-3	1	0	0
action macro-2-actions-0-0–13-8	0	1	0
action macro-2-actions-1-0–1-1	0	0	0
action macro-4-actions-1-3-0-0–2-2	0	0	1
action macro-3-actions-1-3-0–3-1	0	0	0
action macro-3-actions-1-3-3–6-4	0	0	0
action macro-2-actions-1-3–7-5	0	0	0
action macro-3-actions-1-0-0–9-2	0	0	1
action macro-4-actions-1-0-3-0–10-1	0	0	0
action macro-3-actions-1-2-1–11-6	0	0	0
action macro-2-actions-1-2–12-7	0	0	0
action macro-2-actions-0-3–13-9	0	0	0

Table C.4: Detail of the found macro-operators for `satellite` domain.

Bibliography

R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very large data bases.*, volume 1215, pages 487–499, 1994. (Cited on pages 49, 109, and 141.)

R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering.*, pages 3–14. IEEE, 1995. (Cited on page 52.)

R. Aler, D. Borrajo, and P. Isasi. Using genetic programming to learn and improve control knowledge. *Artificial Intelligence*, 141(1-2):29–56, 2002. (Cited on page 33.)

S. Amarel. On representations of problems of reasoning about actions. 1968. (Cited on pages 33 and 36.)

M. Asai and A. S. Fukunaga. Solving large-scale planning problems by decomposition and macro generation. In *Proceedings of the Twenty-Fifth International Conference on International Conference on Automated Planning and Scheduling*, 2015. (Cited on page 4.)

J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 429–435, 2002. URL http://doi.acm.org/10.1145/775047.775109. (Cited on page 53.)

R. Barták. A novel constraint model for parallel planning. In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*, pages 9–14, 2011. (Cited on page 27.)

A. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997. URL http://dx.doi.org/10.1016/S0004-3702(96)00047-1. (Cited on pages 3 and 29.)

B. Bonet and H. Geffner. Heuristic search planner 2.0. *AI Magazine*, 22(3):77, 2001. (Cited on pages 3, 28, and 31.)

B. Bonet and M. Helmert. Strengthening landmark heuristics via hitting sets. In *ECAI*,

volume 215, pages 329–334, 2010. (Cited on page 31.)

D. Borrajo and M. Veloso. Lazy incremental learning of control knowledge for efficiently obtaining quality plans. In *Lazy learning*, pages 371–405. Springer, 1997. (Cited on page 33.)

A. Botea, M. Müller, and J. Schaeffer. Using component abstraction for automatic generation of macro-actions. In *Proceedings of the Fourteenth International Conference on International Conference on Automated Planning and Scheduling.*, pages 181–190, 2004. (Cited on pages 4, 7, 64, and 76.)

A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer. Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators. *Journal of Artificial Intelligence Research*, 24:581–621, 2005a. (Cited on pages 4, 7, 28, 33, 36, 37, 38, 76, 148, and 150.)

A. Botea, M. Müller, and J. Schaeffer. 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, 2005b. (Cited on pages 7 and 141.)

M. Campbell, A. Hoane, and F. hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1): 57 – 83, 2002. URL http://www.sciencedirect.com/science/article/pii/S0004370201001291. (Cited on page 17.)

C. Carrick, Q. Yang, I. Abi-Zeid, and L. Lamontagne. Activating CBR systems through autonomous information gathering. In *International Conference on Case-Based Reasoning*, pages 74–88. Springer, 1999. (Cited on page 33.)

S. Castellanos-Paez, D. Pellier, H. Fiorino, and S. Pesty. Mining useful macro-actions in planning. In *Third International Conference on Artificial Intelligence and Pattern Recognition*, pages 1–6. IEEE, 2016. (Cited on pages 8 and 66.)

L. Chrpa. Generation of macro-operators via investigation of action dependencies in plans. *The Knowledge Engineering Review*, 25(3):281–297, 2010. (Cited on page 7.)

L. Chrpa, M. Vallati, and T. L. McCluskey. 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*, 2014. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/view/7905. (Cited on pages 4, 7, 36, 37, 38, 148, and 150.)

L. Chrpa, M. Vallati, and T. McCluskey. On the online generation of effective macro-operators. In Q. Yang and M. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1544–1550. AAAI Press, 2015. URL http://eprints.hud.ac.uk/24492/. (Cited on pages 36 and 37.)

A. Coles and A. Smith. Marvin: A heuristic search planner with online macro-action learning. *Journal of Artificial Intelligence Research*, 28:119–156, 2007. (Cited on pages 4 and 37.)

A. Coles, M. Fox, and A. Smith. Online identification of useful macro-actions for planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, 2007. (Cited on pages 36 and 37.)

S. Craw, N. Wiratunga, and R. C. Rowe. Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence*, 170(16-17):1175–1192, 2006. (Cited on page 33.)

P. Crews and G. Maxia. test_db. https://github.com/datacharmer/test_db, 2015. (Cited on page 42.)

J. C. Culberson and J. Schaeffer. Pattern databases. *Computational Intelligence*, 14(3): 318–334, 1998. (Cited on page 31.)

K. Currie and A. Tate. O-plan: the open planning architecture. *Artificial intelligence*, 52(1):49–86, 1991. (Cited on page 28.)

C. Dawson and L. Siklossy. The role of preprocessing in problem solving systems: An ounce of reflection is worth a pound of backtracking. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1*, pages 465–471. Morgan Kaufmann Publishers Inc., 1977. (Cited on pages 36, 64, and 150.)

T. de la Rosa, S. Jimenez, and D. Borrajo. Learning relational decision trees for guiding heuristic planning. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 60–67, 2008. (Cited on page 33.)

R. L. De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T COX, K. Forbus, et al. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(3):215–240, 2005. (Cited on page 33.)

M. B. Do and S. Kambhampati. Solving planning-graph by compiling it into csp. In *AIPS*, pages 82–91, 2000. (Cited on page 27.)

K. Dräger, B. Finkbeiner, and A. Podelski. Directed model checking with distance-preserving abstractions. *International Journal on Software Tools for Technology Transfer*, 11(1):27–37, 2009. (Cited on page 31.)

A. Dulac, D. Pellier, H. Fiorino, and D. Janiszek. Learning useful macro-actions for planning with n-grams. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 803–810, 11 2013. doi: 10.1109/ICTAI.2013.123. (Cited on pages 4, 5, 36, 37, 148, and 150.)

S. Edelkamp. Planning with pattern databases. In *ECP-01*, page 13, 2001. (Cited on page 31.)

K. Erol, J. A. Hendler, and D. S. Nau. Umcp: A sound and complete procedure for hierarchical task-network planning. In *AIPS*, volume 94, pages 249–254, 1994. (Cited on page 28.)

O. Etzioni. Acquiring search-control knowledge via static analysis. *Artificial Intelli-*

gence, 62(2):255–301, 1993. (Cited on page 33.)

D. J. Fagnant and K. Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167 – 181, 2015. ISSN 0965-8564. doi: https://doi.org/10.1016/j.tra.2015.04.003. URL http://www.sciencedirect.com/science/article/pii/S0965856415000804. (Cited on page 2.)

R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 3-4(2):189–208, 1971. (Cited on pages 21 and 28.)

R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial intelligence*, 3:251–288, 1972. (Cited on page 36.)

P. Fournier-Viger, C.-W. Wu, and V. S. Tseng. Mining maximal sequential patterns without candidate maintenance. In *International Conference on Advanced Data Mining and Applications*, pages 169–180. Springer, 2013. (Cited on page 54.)

P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas. Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer, 2014a. (Cited on pages 53 and 54.)

P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng. SPMF: A java open-source pattern mining library. *Journal of Machine Learning Research*, 15:3569–3573, 2014b. URL http://jmlr.org/papers/v15/fournierviger14a.html. (Cited on page 67.)

P. Fournier-Viger, C.-W. Wu, A. Gomariz, and V. S. Tseng. VMSP: Efficient vertical mining of maximal sequential patterns. In *Canadian Conference on Artificial Intelligence*, pages 83–94. Springer, 2014c. (Cited on page 54.)

P. Fournier-Viger, C.-W. Lin, Q.-H. Duong, T.-L. Dam, L. Ševčík, D. Uhrin, and M. Voznak. Pfpm: discovering periodic frequent patterns with novel periodicity measures. In *Proceedings of the 2nd Czech-China Scientific Conference 2016*. InTech, 2017a. (Cited on page 45.)

P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017b. (Cited on page 54.)

F. Fumarola, P. F. Lanotte, M. Ceci, and D. Malerba. Clofast: closed sequential pattern mining using sparse and vertical id-lists. *Knowledge and Information Systems*, 48(2): 429–463, 2016. (Cited on pages 54 and 67.)

C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning to rank for synthesizing planning heuristics. *arXiv preprint arXiv:1608.01302*, 2016. (Cited on page 33.)

P. H. H. Geffner and P. Haslum. Admissible heuristics for optimal planning. In *Proceedings of the 5th Internat. Conf. of AI Planning Systems (AIPS 2000)*, pages 140–149, 2000. (Cited on pages 3 and 30.)

M. Ghallab, D. Nau, and P. Traverso. Automated planning: theory and practice, 2004. (Cited on page 2.)

A. Gomariz, M. Campos, R. Marin, and B. Goethals. Clasp: An efficient algorithm for mining frequent closed sequences. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 50–61. Springer, 2013. (Cited on page 54.)

E.-Z. Guan, X.-Y. Chang, Z. Wang, and C.-G. Zhou. Mining maximal sequential patterns. In *Neural Networks and Brain, 2005. ICNN&B'05. International Conference on*, volume 1, pages 525–528. IEEE, 2005. (Cited on page 54.)

K. J. Hammond. Explaining and repairing plans that fail. *Artificial intelligence*, 45(1-2): 173–228, 1990. (Cited on page 33.)

J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359. ACM, 2000. (Cited on page 53.)

J. Han, M. Kamber, and J. Pei. 13 - data mining trends and research frontiers. In J. Han, M. Kamber, and J. Pei, editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 585 – 631. Morgan Kaufmann, Boston, third edition edition, 2012. ISBN 978-0-12-381479-1. doi: https://doi.org/10.1016/B978-0-12-381479-1.00013-7. URL http://www.sciencedirect.com/science/article/pii/B9780123814791000137. (Cited on page 51.)

P. Haslum, B. Bonet, H. Geffner, et al. New admissible heuristics for domain-independent planning. In *AAAI*, volume 5, pages 9–13, 2005. (Cited on page 30.)

P. Haslum, A. Botea, M. Helmert, B. Bonet, S. Koenig, et al. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI*, volume 7, pages 1007–1012, 2007. (Cited on page 31.)

M. Helmert. The fast downward planning system. *J. Artif. Intell. Res. (JAIR)*, 26:191–246, 2006. doi: 10.1613/jair.1705. URL http://dx.doi.org/10.1613/jair.1705. (Cited on page 31.)

M. Helmert and C. Domshlak. Landmarks, critical paths and abstractions: what's the difference anyway? In *ICAPS*, pages 162–169, 2009. (Cited on page 31.)

M. Helmert and H. Geffner. Unifying the causal graph and additive heuristics. In *ICAPS*, pages 140–147, 2008. (Cited on page 31.)

M. Helmert, P. Haslum, J. Hoffmann, et al. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, pages 176–183, 2007. (Cited on page 31.)

F. Herrera, M. Lozano, and J. L. Verdegay. A learning process for fuzzy control rules using genetic algorithms. *Fuzzy sets and systems*, 100(1-3):143–158, 1998. (Cited on page 33.)

J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)*, 14:253–302, 2001. doi: 10.1613/jair.855.

URL http://dx.doi.org/10.1613/jair.855. (Cited on pages 28, 30, and 31.)

T. Hofmann, T. Niemueller, and G. Lakemeyer. 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*, 2017. (Cited on pages 7, 36, 37, and 148.)

G. A. Iba. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3(4):285–317, 1989. (Cited on page 36.)

O. Ilghami, D. S. Nau, H. Munoz-Avila, and D. W. Aha. Camel: Learning method preconditions for htn planning. In *AIPS*, pages 131–142, 2002. (Cited on page 32.)

O. Ilghami, D. S. Nau, and H. Munoz-Avila. Learning to do htn planning. In *ICAPS*, pages 390–393, 2006. (Cited on page 32.)

I. C. ILOG. 8.0 user's manual. *ILOG SA, Gentilly, France*, 2002. (Cited on page 27.)

S. Jiménez Celorrio, A. Coles, and A. Coles. The seventh international planning competition - learning track, 2011. URL http://www.plg.inf.uc3m.es/ipc2011-learning/. (Cited on pages 68 and 129.)

A. Jonsson. The role of macros in tractable planning. *J. Artif. Int. Res.*, 36(1):471–511, Sept. 2009. ISSN 1076-9757. URL http://dl.acm.org/citation.cfm?id=1734953.1734964. (Cited on page 150.)

H. A. Kautz, B. Selman, et al. Planning as satisfiability. In *ECAI*, volume 92, pages 359–363. Citeseer, 1992. (Cited on pages 3 and 26.)

R. Khardon. Learning action strategies for planning domains. *Artificial Intelligence*, 113(1-2):125–148, 1999. (Cited on page 33.)

J. Koehler and J. Hoffmann. Handling of inertia in a planning system. 1999. (Cited on page 149.)

R. E. Korf. Macro-operators: A weak method for learning. *Artificial intelligence*, 26 (1):35–77, 1985. (Cited on page 36.)

P. Laborie and M. Ghallab. Ixtet: an integrated approach for plan generation and scheduling. In *Emerging Technologies and Factory Automation, 1995. ETFA'95, Proceedings., 1995 INRIA/IEEE Symposium on*, volume 1, pages 485–495. IEEE, 1995. (Cited on page 28.)

D. Long and M. Fox. The 3rd international planning competition: Results and analysis. *J. Artif. Int. Res.*, 20(1):1–59, Dec. 2003. ISSN 1076-9757. (Cited on page 38.)

A. Lopez and F. Bacchus. Generalizing graphplan by formulating planning as a csp. In *IJCAI*, volume 3, pages 954–960, 2003. (Cited on pages 3 and 27.)

S. Lu and C. Li. Aprioriadjust: An efficient algorithm for discovering the maximum sequential patterns. In *Proc. Intern. Workshop knowl. Grid and grid intell*, 2004. (Cited on page 54.)

N. R. Mabroukeh and C. I. Ezeife. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR)*, 43(1):3, 2010. (Cited on page 54.)

T. McCluskey, S. Cresswell, N. Richardson, and M. M. West. Automated acquisition of action knowledge. In *International Conference on Agents and Artificial Intelligence (ICAART)*, pages 93–100, 1 2009. URL http://eprints.hud.ac.uk/id/eprint/3292/. (Cited on page 32.)

D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. 1998. (Cited on page 19.)

D. M. McDermott. The 1998 ai planning systems competition. *AI magazine*, 21(2):35, 2000. (Cited on page 19.)

T. M. Mitchell. Generalization as search. In *Readings in artificial intelligence*, pages 517–542. Elsevier, 1981. (Cited on page 32.)

D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila. Shop: Simple hierarchical ordered planner. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 968–973. Morgan Kaufmann Publishers Inc., 1999. (Cited on page 28.)

D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. Shop2: An htn planning system. *Journal of artificial intelligence research*, 20:379–404, 2003. (Cited on page 28.)

A. Newell, J. C. Shaw, and H. A. Simon. *The processes of creative thinking*. Rand Corporation Santa Monica, CA, 1959. (Cited on page 24.)

A. S. Newell. H., 1963. gps, a program that simulates human thought. *Computers and thought*, pages 279–293, 1963. (Cited on page 24.)

M. A. H. Newton and J. Levine. Implicit learning of macro-actions for planning. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 8 2010. (Cited on pages 4, 7, 33, and 76.)

M. A. H. Newton, J. Levine, M. Fox, and D. Long. Learning macro-actions for arbitrary planners and domains. In *ICAPS*, volume 2007, pages 256–263, 2007. (Cited on pages 36, 37, and 148.)

X. Nguyen and S. Kambhampati. Reviving partial order planning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 459–466, 2001. (Cited on pages 3 and 28.)

J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge & Data Engineering*, (11):1424–1440, 2004. (Cited on page 53.)

D. Pellier and H. Fiorino. Pddl4j: a planning domain description library for java. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(1):143–176, 2018.

(Cited on page 66.)

J. S. Penberthy, D. S. Weld, et al. Ucpop: A sound, complete, partial order planner for adl. In *Proceedings of the third international conference KR'92*, 1992. (Cited on page 28.)

J. Porteous, L. Sebastia, and J. Hoffmann. On the extraction, ordering, and usage of landmarks in planning. In *Sixth European Conference on Planning*, 2014. (Cited on page 31.)

S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Pearson Education Limited., 2010. (Cited on page 2.)

E. D. Sacerdoti. A structure for plans and behavior. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1975. (Cited on page 28.)

E. Salvemini, F. Fumarola, D. Malerba, and J. Han. Fast sequence mining based on sparse id-lists. In *International Symposium on Methodologies for Intelligent Systems*, pages 316–325. Springer, 2011. (Cited on page 53.)

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. (Cited on page 17.)

R. Smullyan. *First-order Logic*. Dover books on advanced mathematics. Dover, 1995. ISBN 9780486683706. URL https://books.google.fr/books?id=kgvhQ-oSZiUC. (Cited on page 89.)

R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*, pages 1–17. Springer, 1996. (Cited on page 53.)

A. Tate. Generating project networks. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'77, pages 888–893, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1622943.1623021. (Cited on page 28.)

Á. Torralba Arias de Reyna. Symbolic search and abstraction heuristics for cost-optimal planning in automated planning. 2015. (Cited on page 30.)

C. Urmson et al. Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23(2):66–68, 2008. (Cited on page 2.)

M. Van Den Briel, J. Benton, S. Kambhampati, and T. Vossen. An lp-based heuristic for optimal planning. In *International Conference on Principles and Practice of Constraint Programming*, pages 651–665. Springer, 2007. (Cited on page 31.)

M. H. Van Den Briel and S. Kambhampati. Optiplan: Unifying ip-based and graph-based planning. *Journal of Artificial Intelligence Research*, 24:919–931, 2005. (Cited on pages 3 and 27.)

J. Wang and J. Han. BIDE: efficient mining of frequent closed sequences. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 79–90, 3 2004. doi: 10.1109/ICDE.2004.1319986. (Cited on page 54.)

X. Wang. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Machine Learning Proceedings 1995*, pages 549–557. Elsevier, 1995. (Cited on page 32.)

Y. Xu, A. Fern, and S. W. Yoon. Discriminative learning of beam-search heuristics for planning. In *IJCAI*, pages 2041–2046, 2007. (Cited on page 33.)

X. Yan, J. Han, and R. Afshar. Clospan: Mining: Closed sequential patterns in large datasets. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 166–177. SIAM, 2003. (Cited on page 54.)

Z. Yang and M. Kitsuregawa. Lapin-spam: An improved algorithm for mining sequential pattern. In *Data Engineering Workshops, 2005. 21st International Conference on*, pages 1222–1222. IEEE, 2005. (Cited on page 53.)

S. W. Yoon, A. Fern, and R. Givan. Learning heuristic functions from relaxed plans. In *ICAPS*, pages 162–171, 2006. (Cited on page 33.)

S. W. Yoon, A. Fern, and R. Givan. Using learned policies in heuristic-search planning. In *IJCAI*, volume 7, pages 2047–2052, 2007. (Cited on page 33.)

M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60, 2001. (Cited on page 53.)

Abstract

Intuitively, a system capable of exploiting its past experiences should be able to achieve better performance. One way to build on past experiences is to learn macros (i.e. routines). They can then be used to improve the performance of the solving process of new problems. In automated planning, the challenge remains on developing powerful planning techniques capable of effectively explore the search space that grows exponentially. Learning macros from previously acquired knowledge has proven to be beneficial for improving a planner's performance.

This thesis contributes mainly to the field of automated planning, and it is more specifically related to learning macros for classical planning. We focused on developing a domain-independent learning framework that identifies sequences of actions (even non-adjacent) from past solution plans and selects the most useful routines (i.e. macros), based on *a priori* evaluation, to enhance the planning domain.

First, we studied the possibility of using sequential pattern mining for extracting frequent sequences of actions from past solution plans, and the link between the frequency of a macro and its utility. We found out that the frequency alone may not provide a consistent selection of useful macro-actions (i.e. sequences of actions with constant objects).

Second, we discussed the problem of learning macro-operators (i.e. sequences of actions with variable objects) by using classic pattern mining algorithms in planning. Despite the efforts, we find ourselves in a dead-end with the selection process because the pattern mining filtering structures are not adapted to planning.

Finally, we provided a novel approach called METEOR, which ensures to find the frequent sequences of operators from a set of plans without a loss of information about their characteristics. This framework was conceived for mining macro-operators from past solution plans, and for selecting the optimal set of macro-operators that maximises the node gain. It has proven to successfully mine macro-operators of different lengths for four different benchmarks domains and thanks to the selection phase, be able to deliver a positive impact on the search time without drastically decreasing the quality of the plans.

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

Résumé

Intuitivement, un système capable d'exploiter son expérience devrait être capable d'atteindre de meilleures performances. Une façon de tirer parti des expériences passées est d'apprendre des macros (c.-à-d. des routines), elle peuvent être ensuite utilisés pour améliorer la performance du processus de résolution de nouveaux problèmes. Le défi de la planification automatique est de développer des techniques de planification capables d'explorer efficacement l'espace de recherche qui croît exponentiellement. L'apprentissage de macros à partir de connaissances précédemment acquises s'avère bénéfique pour l'amélioration de la performance d'un planificateur.

Cette thèse contribue principalement au domaine de la planification automatique, et plus spécifiquement à l'apprentissage de macros pour la planification classique. Nous nous sommes concentrés sur le développement d'un modèle d'apprentissage indépendant du domaine qui identifie des séquences d'actions (même non adjacentes) à partir de plans solutions connus. Ce dernier sélectionne les routines les plus utiles (c'est-à-dire les macros), grâce à une évaluation *a priori*, pour améliorer le domaine de planification.

Tout d'abord, nous avons étudié la possibilité d'utiliser la fouille de motifs séquentiels pour extraire des séquences fréquentes d'actions à partir de plans de solutions connus, et le lien entre la fréquence d'une macro et son utilité. Nous avons découvert que la fréquence seule peut ne pas fournir une sélection cohérente de macro-actions utiles (c.-à-d. des séquences d'actions avec des objets constants).

Ensuite, nous avons discuté du problème de l'apprentissage des macro-opérateurs (c'est-à-dire des séquences d'actions avec des objets variables) en utilisant des algorithmes classiques de fouille de motifs dans la planification. Malgré les efforts, nous nous sommes trouvés dans une impasse dans le processus de sélection car les structures de filtrage de la fouille de motifs ne sont pas adaptées à la planification.

Finalement, nous avons proposé une nouvelle approche appelée METEOR, qui permet de trouver les séquences fréquentes d'opérateurs d'un ensemble de plans sans perte d'information sur leurs caractéristiques. Cette approche a été conçue pour l'extraction des macro-opérateurs à partir de plans solutions connus, et pour la sélection d'un ensemble optimal de macro-opérateurs maximisant le gain en nœuds. Il s'est avéré efficace pour extraire avec succès des macro-opérateurs de différentes longueurs pour quatre domaines de référence différents. De plus, grâce à la phase de sélection l'approche a montré un impact positif sur le temps de recherche sans réduire drastiquement la qualité des plans.

Mots clés : Planification Automatique, Intelligence Artificielle, Macro-opérateurs, Macro-actions, Fouille de données, Apprentissage.