Journal of Artificial Intelligence Research 78 (2023) 527-577 Submitted 03/2023; published 11/2023

Maintenance of Plan Libraries for Case-Based Planning: Offline and Online Policies

Alfonso Emilio Gerevini ALFONSO.GEREVINI@UNIBS.IT
Alessandro Saetti ALESSANDRO.SAETTI@UNIBS.IT
Ivan Serina IVAN.SERINA@UNIBS.IT
Andrea Loreggia ANDREA.LOREGGIA@UNIBS.IT
Luca Putelli LUCA.PUTELLI@UNIBS.IT
Università degli Studi di Brescia
Brescia, Italy

Anna Roubíčková A.ROUBICKOVA@EPCC.ED.AC.UK
The University of Edinburgh
Edinburgh, The United Kingdom

Abstract

Case-based planning is an approach to planning where previous planning experience provides guidance to solving new problems. Such a guidance can be extremely useful, or even necessary, when the new problem is very hard to solve, or the stored previous experience is highly valuable, because, e.g., it was provided or validated by human experts, and the system should try to reuse it as much as possible. To do so, a case-based planning system stores in a library previous planning experience in the form of already encountered problems and their solutions.

The quality of such a plan library critically influences the performance of the planner, and therefore it needs to be carefully designed and created. For this reason, it is also important to update the library during the lifetime of the system, as the type of problems being addressed may evolve or differ from the ones the library was originally designed for. Moreover, like in general case-based reasoning, the library needs to be maintained at a manageable size, otherwise the computational cost of querying it grows excessively, making the entire approach ineffective.

In this paper, we formally define the problem of maintaining a library of cases, discuss which criteria should drive the maintenance, study the computational complexity of the maintenance problem, and propose offline techniques to reduce an oversized library that optimize different criteria. Moreover, we introduce a complementary online approach that attempts to limit the growth of the library, and we consider the combination of offline and online techniques to ensure the best performance of the case-based planner. Finally, we experimentally show the practical effectiveness of the offline and online methods for reducing the library.

1. Introduction

Automated Planning in AI deals with the task of generating a partially ordered set of actions, called plans, which allows a system to transition from an initial state to a state satisfying a goal specification (Ghallab, Nau, & Traverso, 2016). Finding a plan solving a planning problem specified by a simple planning language like STRIPS (Fikes & Nilsson, 1971) and even only deciding its existence are PSPACE-complete problems, unless very

severe restrictions are made (Bäckström & Nebel, 1995; Bylander, 1994). An extensive research has been devoted to developing methods and tools for efficiently deriving plans in spite of the worst-case intractability. Over time, different approaches and many techniques have been investigated, such as domain-independent heuristics for guiding the search of a plan (e.g., Domshlak, Hoffmann, & Katz, 2015; Gerevini & Serina, 2003; Gerevini, Saetti, & Serina, 2008; Helmert, 2006; Helmert & Domshlak, 2009; Hoffmann, 2001; Richter & Westphal, 2010), and the learning and use of domain-specific knowledge in various forms (Jiménez, de la Rosa, Fernández, Fernández, & Borrajo, 2012; Zimmerman & Kambhampati, 2003). In this paper, we investigate the approach to planning based on exploiting learning by case-based reasoning, which is also known as *planning by reuse* or *case-based planning* (Borrajo, Roubíčková, & Serina, 2014; Cox, Muñoz-Avila, & Bergmann, 2005; Hammond, 1989; Liberatore, 2005; Ontañón, Mishra, Sugandh, & Ram, 2010; Spalazzi, 2001).

Case-Based Planning (CBP) is a type of case-based reasoning (CBR) that uses the stored experiences to lower the difficulty of solving planning problems (Cox et al., 2005; Hammond, 1989, 1990). CBP relies on the observation that for many real domains the type of problems that are solved does not vary much, and tends to recur. Thus, one can expect that previous solutions to similar problems will be useful when solving new problems. For example, this can be true when a new problem involves goals and an initial state that are very similar to those of a previously solved one, due to a slight variation of goals during plan execution, execution time failures, or similar reasons. In those cases, it might be more efficient to change the existing plan rather than to re-plan from scratch (Hanks & Weld, 1995; Gerevini & Serina, 2010; Gerevini, Saetti, & Serina, 2012; Scala, Micalizio, & Torasso, 2015; Scala & Torasso, 2015). In CBP, a case is a pair consisting of a planning problem and some reuse information about it. Usually, the reuse information consists of a plan for solving the problem. The case base (CB) is a set of cases, and in the context of CBP, it is also called *plan library*. The planning domain formalization, specified by PDDL (Fox & Long, 2003) or other planning languages, is an additional part of the case base.

As the experience of the system grows, the competence of the case base is expected to increase since new solutions are found and stored. In the field of CBR, the notion of *competence* intends to define the utility of the case base in terms of "variety of problems the case base can help solving" (Smyth & McKenna, 2001) or to "a particular set of performance objectives" (Leake, Kinley, & Wilson, 1997). A suitable measure of this concept in planning is yet to be defined and will be investigated in this paper.

As in a typical CBR system, the high-level steps of a CBP system and their interactions with the case base can be organised in the cyclic structure depicted in Figure 1 (Aamodt & Plaza, 1994). The system queries the case base to retrieve case(s) considered useful for solving a new problem Π (RETRIEVE); the solution(s) from the retrieved case(s) are applied to Π in order to find a solution π (REUSE); the new solution may need to be revised to identify and fix possible flaws (REVISE); the possibly revised solution and problem Π are used to create a new case that may be introduced to the case base. In addition, the system may include techniques for analyzing and reorganizing the case base when reaching some limit conditions (RETAIN). When a CBR system faces a new problem, it performs these procedures in a sequence that starts by querying the case base and ends by (possibly) updating it.

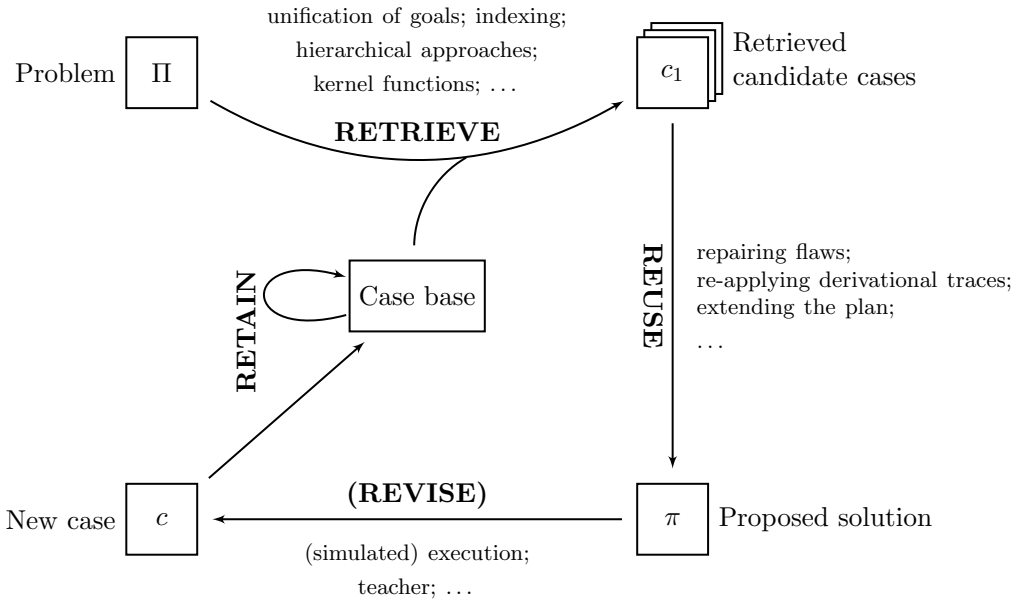

Figure 1: The typical cycle of a CBR system.

The larger the set of problems the plan library can help to address, the higher the quality of the plan library. The retention phase of the CBP methodology consists of policies to preserve and improve the quality of the case base. The existing work in CBP has focused mostly on the reuse and retrieval steps. The retention usually settles with one of the following extreme policies: either maintaining everything or using a pre-built case base that is maintained fixed during the lifetime of the system. In this paper, we study the problem of how to optimise the size and quality of the plan library in order to support efficient plan retrieval and reuse; we call such a problem *plan library maintenance problem*. The main contributions of our work are:

- A discussion on the assumptions underlying the case-based methodology in the context of planning and, in order to formalise these assumptions, the identification of conditions when an existing plan can be useful to solve a planning problem.
- A suitable measure of the notion of competence for case-based planning, as well as the identification of some criteria for evaluating the quality of the plan library.
- A formal definition of the problem of maintaining a plan library optimizing either its size or quality, and a computational complexity characterization of the plan library maintenance problem.
- A set of policies for maintaining the plan library. Some of them concern with reducing the size of the library when it becomes too big. They can incur a high computational cost and hence are suitable for offline usage. Other policies concern bounding the growth of the library by adding only the most beneficial cases, which can be done relatively fast, making such policies suitable for online usage.

- A thorough experimental analysis evaluating our CBP approach using different metrics for measuring case similarity and the effectiveness of the proposed offline and online policies addressing the plan library maintenance problem.

The remainder of the paper is structured as follows: Section 2 discusses the assumptions underlying the case-based methodology in the context of planning. Section 3 formalises the problem of maintaining the case base, introduces criteria for evaluating its competence, and proposes different policies for maintaining it. Section 4 studies the complexity of maintaining a plan library. Section 5 presents experimental results evaluating the proposed techniques. Section 6 summarises the related work. Finally, Section 7 draws conclusions and mentions future work.

2. Definitions and Assumptions

The *planning problem* consists of determining and ordering a set of actions (a plan) whose execution transforms a given initial state of the world into a new state satisfying some desired goals. This task is also called generative planning (Ghallab, Nau, & Traverso, 2004). More formally, a generative planning problem Π is a tuple $\langle \mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, where

- $\mathcal{F}$ is a set of positive literals called the *Facts* of Π;
- $\mathcal{A}$ is the set of *Actions* of Π; each action a is a triple $\langle \text{pre}(a), \text{eff}(a)^+, \text{eff}(a)^- \rangle$, where
  - $\text{pre}(a) \subseteq \mathcal{F}$ is a set of positive literals called the *preconditions* of a,
  - $\text{eff}(a)^+ \subseteq \mathcal{F}$ is a set of positive literals called the *positive effects* of a,
  - $\text{eff}(a)^- \subseteq \mathcal{F}$ is a set of negative literals called the *negative effects* of a;
- $\mathcal{I} \in 2^{\mathcal{F}}$ is the *initial world state* of Π;
- $\mathcal{G} \subseteq \mathcal{F}$ is a set of literals called the *goals* of Π.

For simplicity, in the rest of the paper we refer to this task as the "planning problem". A planning problem can be more concisely specified by a set of objects, a set of predicates, and a set of action schemas. An action schema consists of a precondition formula and an effect formula. These formulas are conjunctions of predicates, whose arguments are parameters of the action. The sets of predicates and action schemas form the so-called *planning domain*. The sets $\mathcal{F}$ and $\mathcal{A}$ defining the planning problem can be derived from the planning domain by instantiating parameters of domain predicates and action schemas with all the possible combinations of objects. Notably, the same sets of predicates and action schemas can be used to specify the sets $\mathcal{F}$ and $\mathcal{A}$ of different instances of planning problems; these instances are then associated with the same planning domain.

In our work, the language used for the encoding of a planning problem is the well-known PDDL (Ghallab, Howe, Knoblock, McDermott, Ram, Veloso, Weld, & Wilkins, 1998), and we focus on *propositional* planning problems (Fikes & Nilsson, 1971), i.e., planning problems where world states are sets of literals. Under the closed world assumption, if a literal is not in a world state then in that state it is false. An action a is executable in a world state s if the preconditions of a are satisfied in s, i.e., $\text{pre}(a) \subseteq s$. The state s' resulting from the

execution of a in s is obtained by adding the positive effects of a to s and removing the negative effects of a from s, i.e., $s' = (s \setminus \text{eff}(a)^-) \cup \text{eff}(a)^+$. For simplicity, we assume that plans are sequences of actions. A sequence of actions $\langle a_0, ..., a_{n-1} \rangle$ is said to be a *plan* if, when iteratively applied starting from $\mathcal{I}$, it is such that each action a_i is applicable in the state s_i resulting from the plan prefix before it (i.e., $s_i \models \text{pre}(a_i)$), and the last produced state s_n satisfies the goal, i.e., $s_n \models \mathcal{G}$.

The core idea of CBP is providing a complementary approach to traditional generative planning under two assumptions coming from the field of CBR: *the world is regular* and *similar problems recur* over time. The regularity of the world assumes that similar problems have similar solutions. Such an assumption obviously links together the similarities between problems and solutions, which also provides a guarantee that a retrieved case containing a problem similar to the new problem to be solved will provide a plan that can be adapted without many changes for its reuse.[1] The latter assumption that similar planning problems recur is meant to ensure that, together with the regular assumption, the reuse approach will eventually pay off. These assumptions of CBR and CBP have not been formally stated yet. In the following, we introduce some definitions that formalise them in the context of CBP and allow us to formally characterize the notion of case base maintenance policy.

To design a procedure for the case base maintenance, we start by deciding which parameters of the case base define its quality, and thus which criteria should guide the maintenance policy in determining which experiences to keep and which to discard. Obviously, an important criterion is the variety of problems that the case base can address, which in CBR is also referred to as the case base *competence* (Smyth, 1998), and its interplay with the size, or *cardinality*, of the case base. Unfortunately, this simple notion of competence cannot be directly adopted in the planning context. Differently from CBR, where a case usually either can or cannot be adapted to solve a new problem instance, the reuse procedure of a planning system in principle can *always* adapt the retrieved solution to solve a new problem (if solvable) by changing any unfit part, or by disregarding the whole stored solution and attempting to find a new solution from scratch. Hence, an effective CBP system needs to decide how much a new solution deviates from the one stored in the case base, how expensive the reuse would be, and, consequently, *how useful* the retrieved solution is. With the aim of obtaining a plan library that is useful to solve a wide range of problems, the notion of case quality has to consider the information redundancy within a case, which is captured by the case diversity or distance w.r.t. the other cases in the library.

We start by considering *how well* a case fits a new problem instance in terms of its usability to solve the new instance. Classical planning can be formulated as a search problem in a space of plans (e.g., Ghallab et al., 2004), where the search starts from an empty plan. The adaptation of a retrieved plan in CBP can be formulated similarly, with the difference that the initial plan, instead of the empty plan, is the retrieved one. We can define the *distance* between a stored solution and a solution of the new problem as the minimum number of actions that need to be added/removed in order to convert the stored plan to the new one.

Let $\wp$ denote the space of plans for a given planning domain. Ideally, a plan distance function $d_\pi : (\wp \times \wp) \to [0, 1]$ should measure the distance between two plans in terms of

1. During the retrieval phase, the system has no information about the plan it is looking for, and so it needs to decide solely based on the properties of the problems.

how hard it is to adapt one of the plans to obtain the other from a computational point of view. However, computing such a function is infeasible (Liberatore, 2005). Therefore, we use the following approximation that is easy to compute (Nguyen, Do, Gerevini, Serina, Srivastava, & Kambhampati, 2012).

Definition 1 (Normalized plan distance)**.** *Given two plans π_i and π_j, the* normalized plan distance *function $d_\pi(\pi_i, \pi_j)$ between two plans π_i and π_j is the number of actions that are in π_i and not in π_j plus the number of actions that are in π_j and not in π_i, normalized over the total number of actions in π_i and π_j, i.e.,*

$$d_\pi(\pi_i, \pi_j) = \frac{|\pi_i \ominus \pi_j| + |\pi_j \ominus \pi_i|}{|\pi_i| + |\pi_j|}. \tag{1}$$

Operator $\ominus$ works as follows. For each action a, let π_x and π_y contain l and m instances of a, respectively. If $l > m$, then $\pi_x \ominus \pi_y$ contains $l - m$ instances of a; it contains 0 instances of a otherwise. Clearly, if the case-based system needs to revert to an empty plan (because no part of the stored plan can be reused) and search from there, then the provided case should be considered useless. Hence, we say that a case can be *useful* to solve a problem if the distance between the corresponding plans is not larger than the distance from an empty plan.

Definition 2 (Function addresses)**.** *We say that a case $c_i = \langle \Pi_i, \pi_i \rangle$ can be useful to solve a problem Π, that is,* $\mathsf{addresses}(c_i, \Pi)$*, if there exists a plan π for Π that is close to π_i according to a threshold distance $\delta \in \mathbb{R}$, that is, $d_\pi(\pi_i, \pi) < \delta$ for some $\delta \in \mathbb{R}$.*

The definition of $\mathsf{addresses}(c_i, \Pi)$ relies on the distance between the solutions and completely disregards the relation of the relative problems. However, also the structural properties of the problems play a considerable role, as the case retrieval step is based on the planning problem descriptions. Therefore, we also use a distance function d_p that is intended to measure the similarity of problems. Let $\mathcal{P}$ denote the space of problems in a given planning domain, $\Pi \in \mathcal{P}$ be a new problem, and $\Pi' \in \mathcal{P}$ be a problem previously solved. It is worth noting that Π and Π' can involve different sets of objects, therefore, the reuse of a solution π' for Π' may require defining a matching between the objects of π' (or the corresponding solved problem Π') and those of Π. The following definition of problem distance function assumes that the matching between the objects of Π and Π' has been already performed.

Definition 3 (Problem distance)**.** *The problem distance function $d_p : (\mathcal{P} \times \mathcal{P}) \to [0, 1]$ between two problems Π and Π' in $\mathcal{P}$ is defined as*

$$d_p(\Pi, \Pi') = 1 - \frac{|\mathcal{I}' \cap \mathcal{I}| + |\mathcal{G}' \cap \mathcal{G}|}{|\mathcal{I}'| + |\mathcal{G}|} \tag{2}$$

where $\mathcal{I}$ and $\mathcal{I}'$ ($\mathcal{G}$ and $\mathcal{G}'$) are the initial states (sets of goals) of Π and Π', respectively (Serina, 2010).[2]

2. The denominator of d_p considers only the initial state of problem Π' and the goals of the new problem Π, because the goals in $G' - G$ are irrelevant for the new problem Π, and the facts in $I - I'$ are irrelevant for the executability of the plan solving Π' that is stored in the library together with Π'.

Let Π be a new problem and Π_1 and Π_2 two problems stored in the library. If $d_p(\Pi, \Pi_1) < d_p(\Pi, \Pi_2)$, we say that Π and Π_1 are *more similar* than Π and Π_2. Moreover, the lower $d_p(\Pi, \Pi_1)$ is w.r.t. $d_p(\Pi, \Pi_2)$, the more similar Π and Π_1 are w.r.t. Π and Π_2. Consequently, by the regular world assumption, the library problems with shortest distance w.r.t. Π are more likely to have similar solutions than other library problems, and so it is useful to retrieve from the case base the cases with problems that have the shortest distance to Π. We can say that distance d_p is helpful to *guide the retrieval phase*, while distance d_π can be used to *estimate the plan adaptation effort*.

Since a maintenance policy should consider both costs of retrieval and adaptation, the two functions are combined obtaining a combined distance function $d : ((\mathcal{P} \times \wp) \times (\mathcal{P} \times \wp)) \rightarrow [0, 1]$ measuring distance between cases. The combination of d_p and d_π allows us to assign different importance to the similarity of problems and their solutions, depending on the application requirements. Specifically, the distance d between two cases $c = \langle \Pi, \pi \rangle$ and $c' = \langle \Pi', \pi' \rangle$ is defined as follows:

$$d(c, c') = \alpha \cdot d_p(\Pi, \Pi') + (1 - \alpha) \cdot d_\pi(\pi, \pi')$$

where parameter $\alpha \in [0, 1]$ allows to assign greater importance to the problem descriptions (for the domains where the world is strongly regular) or to the stored plans (for the settings where the stability of the solution matters). The rest of the paper uses such a notion of case distance.

The assumption of regular world presented at the beginning of this section uses a notion of similarity between problems and solutions, neither providing details on how the similarity should be measured nor specifying *which* solutions are considered. This is an important concern when a problem may have several significantly different solutions. We formalise this assumption keeping the notion of similarity undetailed to preserve the generality of the definition, but establishing the quantification over the solutions as follows.

Definition 4 (Regular World Assumption)**.** *Let Π and Π' be two similar planning problems of a planning domain with problem space $\mathcal{P}$ and plan space $\wp$ such that $\Pi \neq \Pi'$. If the world is regular, then*

- *$\forall \pi \in \wp$ that is a solution of Π, $\exists \pi' \in \wp$ that is a solution of Π' such that π and π' are similar, and*
- *$\forall \pi' \in \wp$ that is a solution of Π', $\exists \pi \in \wp$ that is a solution of Π such that π' and π are similar.*

In our work, the problem and plan similarities are measured by the distance functions d_p and d_π, respectively. Case similarity is interpreted by means of the distance function d. The next definition formalises the notions of similarity for problems, plans, and cases.

Definition 5 (Similarity)**.** *Given a problem distance threshold $\delta_p \in \mathbb{R}$, two problems Π and Π' are similar iff $d_p(\Pi, \Pi') \leq \delta_p$. Given a (normalized) plan distance threshold $\delta_\pi \in \mathbb{R}$, two plans π and π' are similar iff $d_\pi(\pi, \pi') \leq \delta_\pi$. Finally, two cases c_i and c_j are similar, written* $\mathbf{similar}_\delta(c_i, c_j)$*, iff $d(c_i, c_j) \leq \delta$, where the value of $\delta \in \mathbb{R}$ depends on the specific definition choice for d as well as on the domain, similarly to δ_π and δ_p.*

The other assumption that we need to formalise in our context is the notion of problem recurrence:

Definition 6 (Recurring Problem Assumption). *For every problem* Π *that the CBP system encounters, it is likely that its case base contains a case* $c_i = \langle \Pi_i, \pi_i \rangle$ *such that* $\mathsf{addresses}(c_i, \Pi)$ *holds.*

Case-based planning relies on the two assumptions formalised in Definitions 4 and 6. Under these assumptions, the system is likely to produce solutions similar to the retrieved plans that are reused, as the problems are also assumed to be similar. Therefore, the new cases added to the library often solve new problem instances that are similar to other cases already solved. For this reason, it can be expected that the cases in the case base create groups of elements, that we call *case clusters*, similar to each other; such clusters could be reduced to smaller groups without significant loss of information.

Definition 7 (Case Cluster). *Given a plan library* $\mathcal{L}$ *and a case-distance threshold* δ*, a* **case cluster** *of* n *elements in* $\mathcal{L}$ *is a subset* C *of* $\mathcal{L}$ *formed by similar elements, i.e.,* $\forall c, c' \in C$, $\exists c_1, \ldots, c_k \in C$ *with* $k \leq n$ *such that* $\mathsf{similar}_\delta(c, c_1)$, $\mathsf{similar}_\delta(c_k, c')$, *and* $\mathsf{similar}_\delta(c_i, c_{i+1})$ *for* $1 \leq i < k$.

A case base can be represented by a graph where the cases c_i form vertices and there is an edge from c_i to c_j if and only if $\mathsf{similar}_\delta(c_i, c_j)$ holds according to the given case-distance threshold δ. A case cluster is a connected component in such a graph, i.e., between any two cases c and c' of the cluster there exists a path from c to c' in the graph.

3. Maintenance of the Plan Library

The plan library maintenance problem consists of deriving a case base $\mathcal{L}'$ that is smaller than the original case base $\mathcal{L}$, but that contains (very) similar experiences. Under such conditions, we say that $\mathcal{L}'$ *reduces* $\mathcal{L}$. The following notions of *case covering* and *case base coverage* are defined to help formalise this concept.

Definition 8 (Function covers). *Given a plan library* $\mathcal{L}$ *and a case-distance threshold* $\delta \in \mathbb{R}$*, a case* $c_i \in \mathcal{L}$ *covers a case* $c_j \in \mathcal{L}$ *according to* δ*, denoted* $\mathsf{covers}_\delta(c_i, c_j)$*, if* $d(c_i, c_j) \leq \delta$.

The relative amount of knowledge provided by the maintained case base $\mathcal{L}'$ compared to the original (full) case base $\mathcal{L}$ is captured by a measure that we call *coverage*.

Definition 9 (Function coverage). *Let* $\mathcal{L}$ *and* $\mathcal{L}'$ *be two case bases and let* $\mathcal{C}$ *be the set of all cases in* $\mathcal{L}$ *that are covered by the cases in* $\mathcal{L}'$ *according to a case-distance threshold* δ*, i.e.,* $\mathcal{C} = \{c_i \in \mathcal{L} \mid \exists c'_i \in \mathcal{L}', \mathsf{covers}_\delta(c'_i, c_i)\}$. *The coverage of* $\mathcal{L}'$ *over* $\mathcal{L}$ *according to* δ*, denoted* $\mathsf{coverage}_\delta(\mathcal{L}', \mathcal{L})$*, is defined as* $\frac{|\mathcal{C}|}{|\mathcal{L}|}$.

We say that a subset $\mathcal{L}'$ of cases of $\mathcal{L}$ contains a planning experience similar to $\mathcal{L}$ when all the cases of $\mathcal{L}$ are covered by $\mathcal{L}'$. In this sense, $\mathcal{L}'$ can be considered a reduction of $\mathcal{L}$.

Definition 10 (Function reduces). *Let* $\mathcal{L}$ *and* $\mathcal{L}'$ *be case bases.* $\mathcal{L}'$ *reduces* $\mathcal{L}$ *according to a case-distance threshold* δ*, denoted as* $\mathsf{reduces}_\delta(\mathcal{L}', \mathcal{L})$*, if and only if* $\mathcal{L}' \subset \mathcal{L}$ *and* $\mathsf{coverage}_\delta(\mathcal{L}', \mathcal{L}) = 1$.

There are many possible ways to reduce a case base, out of which some are more suitable than others. We introduce two criteria for reducing the case base that can significantly influence the performance of a case-based system:

- *minimising the size* of the reduced case base, which can have a significant impact on the efficiency of the retrieval phase;
- *maximising the quality* of the reduced case base, which can significantly influence the cost of adapting the plan in the retrieved case.

Considering the first criterion, the optimal result of the reduction takes into account the number of elements in the reduced case base:

Definition 11 (Cardinality Case Base Maintenance Problem)**.** *Given a case base $\mathcal{L}$ and a case-distance threshold $\delta \in [0,1]$, the cardinality case base maintenance problem is to identify the smallest set $\mathcal{L}' \subset \mathcal{L}$ which reduces $\mathcal{L}$ according to δ.*

In the decision version of the Cardinality Case Base Maintenance Problem, the input $\langle \mathcal{L}, \delta, k \rangle$ is the original case base $\mathcal{L}$, a case-distance threshold $\delta \in [0,1]$, and a natural number k. The question to answer is whether there exists a reduction of $\mathcal{L}$ according to δ that has a size less than or equal to k.

Concerning the second criterion, consider three cases c, c_1, and $c_2 \in \mathcal{L}$ so that $d(c,c_1) < d(c,c_2) < \delta$. By Definition 8, c covers both c_1 and c_2, however, the expected adaptation cost of c_1 is lower than the one of c_2, and therefore c covers c_1 better than c_2. Therefore, the (reduced) library obtained by removing c_1 is better than the (reduced) library obtained by removing c_2, because $d(c,c_1) < d(c,c_2)$ and the associated information loss is smaller.

The criterion of maximizing the quality of the reduced case base can be reformulated as minimizing the coverage information lost when we reduce the case base; such a loss can be measured as the average distance from the removed cases to the closest kept case, that we call *average coverage information loss*. The optimal result according to such a criterion is a case base $\mathcal{L}'$ reducing $\mathcal{L}$ with minimal average coverage information loss. Note, however, that if only such information loss were considered, then $\mathcal{L} = \mathcal{L}'$ would be a special case of optimal reduced case base. Therefore, the quality measure to optimise is more complex because it needs to also take the size of the reduced case base into account. In particular, we define the notion of *neighborhood of a case c* with respect to a similarity distance value δ, denoted $n_\delta(c)$. The similarity is interpreted as proximity in the sense of distance function d. So, the meaning of δ is the same as in Definition 5.

The idea of the case neighborhood is to group elements which contain redundant information and hence that can be reduced to a single case. Note that value δ, together with the structure and distribution of the cases in the case base, influences the size of the case neighborhoods and therefore it determines the amount by which $\mathcal{L}$ can be reduced.

Definition 12 (Neighborhood)**.** *Given a case base $\mathcal{L}$, a case $c \in \mathcal{L}$ and a case-distance threshold $\delta \in [0,1]$, the neighborhood of c, denoted $n_\delta(c)$, is the set of cases $\{c_i \in \mathcal{L} \mid c_i \neq c, d(c,c_i) \leq \delta\}$.*

Given a reduction $\mathcal{L}'$ of $\mathcal{L}$, we define the *lost neighborhood* $L_\delta(c)$ of an element $c \in \mathcal{L}$ as its neighbours in $\mathcal{L} \setminus \mathcal{L}'$, i.e.,

$$L_\delta(c) = \left\{ c_j \in \mathcal{L} \,|\, c_j \in n_\delta(c) \cap \mathcal{L} \setminus \mathcal{L}' \right\} \tag{3}$$

and the *cost* $v_\delta(c)$ *of a case* c as the real function

$$v_\delta(c) = 1 + \sum_{c_j \in L_\delta(c)} \frac{d(c, c_j)}{|L_\delta(c)|}. \tag{4}$$

The first term in (4), the unit cost, represents the cost of including c in $\mathcal{L}'$, while the second term corresponds to the average coverage loss of the neighbors of c, i.e., the elements of $\mathcal{L} \setminus \mathcal{L}'$ that may not to be included in the reduced case base because they are supported by c.

Definition 13 (Reduction cost)**.** *Given a reduction* $\mathcal{L}'$ *of* $\mathcal{L}$ *and a case-distance threshold* δ*, the reduction cost* $\mathcal{M}_\delta(\mathcal{L}')$ *of* $\mathcal{L}'$ *is the sum of costs of all the elements of the reduced set* $\mathcal{L}'$*, i.e.,* $\mathcal{M}_\delta(\mathcal{L}') = \Sigma_{c \in \mathcal{L}'} v_\delta(c)$*.*

Therefore, according to the criterion of maximizing the quality of the reduced case base, the case base maintenance problem can be stated as follows.

Definition 14 (Quality Case Base Maintenance Problem)**.** *Given a case-distance threshold value* $\delta \in [0, 1]$ *and a case base* $\mathcal{L}$*, the quality case base maintenance problem is to identify a reduction* $\mathcal{L}'$ *of* $\mathcal{L}$ *according to* δ *that minimises* $\mathcal{M}_\delta(\mathcal{L}')$*.*

The $\mathcal{M}_\delta$ cost function is used with the aim of minimizing both the total number of cases in the reduction $\mathcal{L}'$ and the distance w.r.t. the corresponding lost neighbors. Given two reductions derived using the same case-distance threshold δ, we prefer the one with the lowest $\mathcal{M}_\delta$, which takes the size of the reduction into account, and determines how closely related the cases in the lost neighborhood are; indeed, the lower total average coverage loss of the neighbors for the cases in the reduction, the better the cases in the reduction represent the original library.

In the decision version of the Quality Case Base Problem $\langle \mathcal{L}, \delta, q \rangle$, the question is whether there exists a reduction of $\mathcal{L}$ according with δ that has a cost less than or equal to a real number q. In Section 4 we will show that the decision versions of the considered case base maintenance problems are NP-complete, and hence also their optimization versions are intractable. Thus, it is worth investigating ways of computing approximate solutions to these problems.

3.1 Offline Maintenance

A high-quality approximation of the solution to the maintenance problems defined before can provide a reduction of the library without losing (too much) of the knowledge contained within. A policy addressing the maintenance problem needs to be highly informed in order to select the most suitable subset of the cases from the library, which can incur a high computational cost. Consequently, such a policy cannot be performed very often or in periods of high activity of the CBP system, and it is more suitable for *offline* usage.

This section introduces an uninformed random policy, which is the baseline in our experimental analysis, and two informed policies for offline use, called *Distance-Guided Policy* and

Coverage-Guided Policy. While Smyth (1998) considers maximising competence as a necessary property of a case base, our informed policies are guided by minimising the amount of knowledge that is lost in the maintenance process, where removing a case from the library implies losing the corresponding knowledge unless the same information is contained in some other case.

3.1.1 Random Policy

The random policy reduces the case base by randomly selecting elements to remove. The probability q of removing an element is the ratio between the required size and the current size. Such a policy is interesting for two reasons: it is easy to implement and very fast to compute. Moreover, it can provide a case base of reasonable quality (Smyth, 1998). However, it has the drawback that there are no guarantees about the coverage of the reduced case base $\mathcal{L}'$ over the original case base $\mathcal{L}$.

Notice also that the random policy heavily relies on the recurring problem assumption. The removal of a case is independent of the removal of other elements, therefore the probability that a whole cluster of cases with size m is removed is q^m. With $q < 1$, bigger case clusters are less likely to be (completely) removed. On the other hand, the probability of removing an "isolated" case (i.e., a single case cluster corresponding to a problem that has not recurred yet) amounts to q; consequently, an aggressive policy (with high q) is likely to significantly decrease the coverage of the case base.

3.1.2 Distance-Guided Policy

Due to the assumption of recurring problems, we expect that the plan library contains a problem that is similar to a new problem to solve and that the problems in the library can be grouped into sets of problems that are similar (close in the sense of d_p) to each other. Consequently, by the assumption of regular world, for a new problem Π' there exists a solution π' that is similar to the solution π of a stored case $c = \langle \Pi, \pi \rangle$, where Π is similar to Π'. The new case $c' = \langle \Pi', \pi' \rangle$ is then similar to c (close in the sense of d), and its inclusion in the case base introduces some redundancy because of its similarity with c.

We propose a distance-guided policy that attempts to remove the cases that are mostly redundant. These cases are those where the distance to other cases is too small. Let c_i^* denote the case with the minimum distance from a case c_i in $\mathcal{L}$, i.e., $d(c_i, c_i^*) \leq d(c_i, c_j),\ \forall c_j \in \mathcal{L} \setminus c_i^*$. The next definition formally states the policy.

Definition 15 (Distance-guided Policy)**.** *Given a case-distance threshold δ, the distance-guided policy iteratively selects a case $c_i \in \mathcal{L}$ defined as* $\operatorname{argmin}_{c_j \in \mathcal{L}}\ d(c_j, c_j^*)$ *and removes c_i from $\mathcal{L}$ if $d(c_i, c_i^*) < \delta$ in the current reduction of $\mathcal{L}$.*

Figure 2 gives an example of a reduced plan library obtained by using the distance-guided policy. In the figure, each circle denotes a case, the grey area is a two-dimensional hyper-space such that the distance between the case with a red "×" and the cases inside the dashed circle is lower than a case-distance threshold δ; white circles represent cases that the policy has already removed from the library, green circles are cases that will be kept in the library, black circles along with the green ones are cases that are currently part of the library under reduction. The example shows the removal of three cases from the library. On

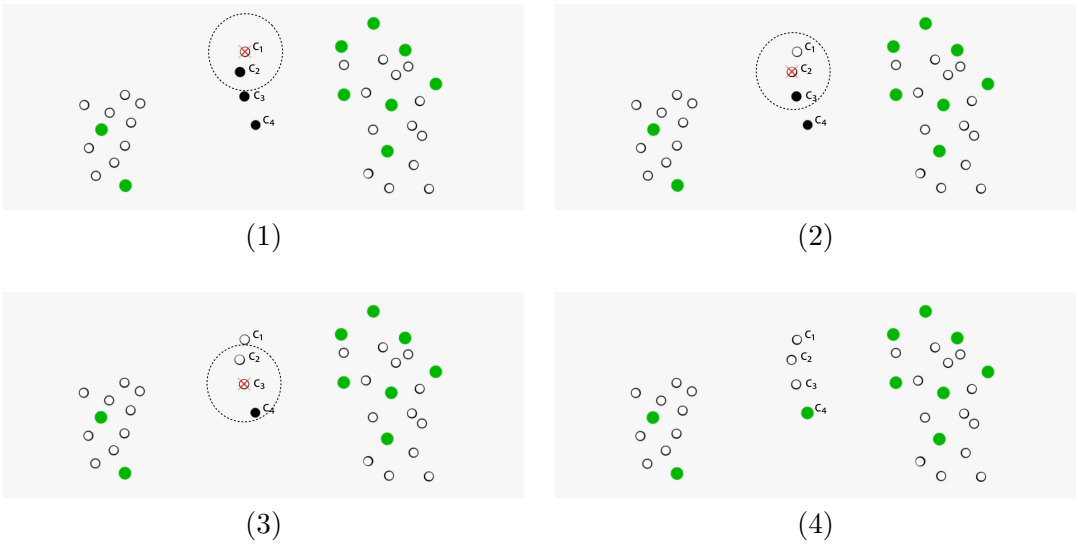

Figure 2: Example of the usage of the distance-guided policy; green circles represent the kept cases; white circles represent the removed cases; the policy is applied to the black cases.

the top-left part (1) of the figure, case c_1 is removed from the library because the distance with its closest case is the lowest in the current reduction of the library and such a distance is lower than a case-distance threshold δ (that is here indicated by the dotted line). For the same reason, the policy discards the cases c_2 and c_3 in the top-right part (2) and in the bottom-left part (3); while on the bottom-right part (4) of the figure case c_4 is kept in the library because it is quite far from the other cases kept in the reduction of the library (the green ones).

Of course, the distance-guided policy can preserve the knowledge in the case base better than the random maintenance. However, non-redundant cases can still be discarded by such a policy, as some information is missed when only the minimum distance cases are considered. For instance, considering the cases c_1 and c_2 in the bottom-right part (4) of the figure, they are not redundant in the final library reduction because they are quite far from the other cases in the reduced plan library (green circles), but they are still removed by the policy.

3.1.3 Coverage-Guided Policy

In order to overcome the limitations of the distance-guided policy, we generalise such a policy to consider *all* the cases that may contain redundant information at once. For that, we use the notion of case neighborhood n_δ and other notions introduced in the previous section. The resulting policy, which we call *Coverage-Guided Policy*, has two variants defined as follows.

Definition 16 (Cardinality-Coverage Policy)**.** *Given a case base $\mathcal{L}$ and a case-distance threshold δ, the cardinality-coverage policy finds a minimal set $\mathcal{L}'$ of cases such that the*

CoverageBasedPolicy($\mathcal{L}, \delta$)

Input: a plan library $\mathcal{L} = \{c_i \mid i \in 1 \leq i \leq n\}$, a threshold $\delta \in \mathbb{R}$.

Output: a plan library $\mathcal{L}'$ reducing $\mathcal{L}$.

1. $\mathcal{L}' \leftarrow \emptyset$;
2. $Uncovered \leftarrow \mathcal{L}$;
3. **repeat**
4. select $c_i \in Uncovered$ that satisfies $condition(c_i)$;
5. $Uncovered \leftarrow Uncovered \setminus n_\delta(c_i)$;
6. $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{c_i\}$;
7. **until** $Uncovered = \emptyset$;
8. **return** $\mathcal{L}'$.

Figure 3: Schema of a greedy algorithm computing an approximation of the Coverage-Based Policy.

union of all their neighborhoods covers all the elements of the given case base $\mathcal{L}$ according with δ, i.e., such that $\mathsf{reduces}_\delta(\mathcal{L}', \mathcal{L})$ *holds.*

Definition 17 (Quality-Coverage Policy)**.** *Given a case base $\mathcal{L}$ and a case-distance threshold δ, the quality-coverage policy finds a set $\mathcal{L}'$ of cases such that the union of all their neighborhoods covers all the elements of the given case base $\mathcal{L}$ according with δ and minimizes the reduction cost $\mathcal{M}_\delta(\mathcal{L}')$.*

Unfortunately, as we will show in Section 4, computing the exact reduced case base by the coverage-guided policy is intractable. Therefore, we propose to compute approximations of the reduced case bases that solve the cardinality and quality case base maintenance problems. For this scope, we use the simple greedy algorithm in Figure 3. The algorithm has two variants that depend on how line 4 is implemented and correspond to the policies of Definitions 16 and 17:

- For the Cardinality Coverage-Guided Policy, the condition test at line 4 of the algorithm is used to select the uncovered element c_i with the greatest $|L_\delta(c_i)|$, in order to maximise the number of uncovered elements in $n_\delta(c_i)$ that can be covered by inserting c_i into $\mathcal{L}'$.
- For the Quality Coverage-Guided Policy, in order to optimise the quality of the reduced case base $\mathcal{L}'$, the condition of line 4 is used to select the uncovered element c_i with the minimum value for $\frac{v_\delta(c_i)}{|L_\delta(c_i)|}$, i.e., the cost $v_\delta(c_i)$ of case c_i scaled down by $|L_\delta(c_i)|$ to favour the cases that cover a higher number of not yet covered elements.

Figure 4 shows an example of the plan library obtained using the cardinality-coverage policy. Let the grey area be a two-dimensional hyper-space such that the distance between the case at the center of the dashed circle and the other cases inside the circle is lower than a case-distance threshold δ. In this figure, the circles represent the cases in $\mathcal{L}$, the green circles represent the cases in the reduced case base $\mathcal{L}'$, the orange circles are the cases in set

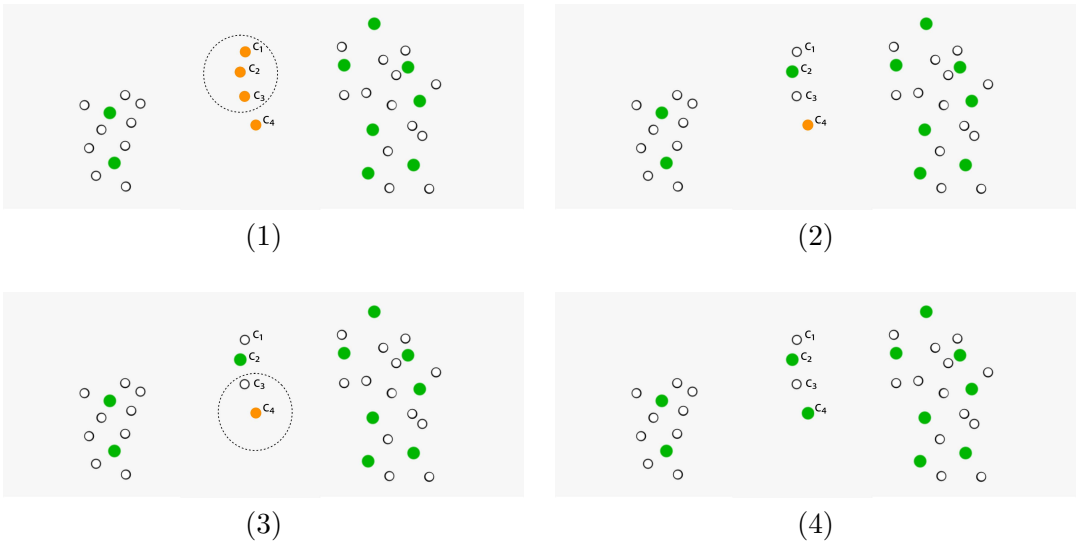

Figure 4: Example of the usage of the cardinality-coverage policy.

Uncovered of CoverageBasedPolicy. The top-left part (1) of the figure shows that case c_2 has the largest lost neighborhood since, differently from the other uncovered cases (in orange), it includes two other cases c_1 and c_3. On the top-right part (2) of the figure c_2 is added to $\mathcal{L}'$, while c_1 and c_3 are removed from *Uncovered*. The bottom-left part (3) shows that case c_4 is the only one left in *Uncovered*; finally, on the bottom-right part (4), c_4 is added to $\mathcal{L}'$. Considering the central cluster, both the distance-guided policy and the cardinality-coverage policy remove cases c_1 and c_3 but, differently from the distance-guided policy, the cardinality-coverage policy correctly keeps case c_2 in the reduced library, as without c_1 and c_3 it is not redundant.

3.2 Online Maintenance

In general, there are two major tasks related to the library maintenance problem. As we have already seen, one of them is concerned with how to reduce the size of the library when it becomes too big. The other task concerns bounding the growth of the library by *adding* only those cases that are beneficial to the overall planning process (plan retrieval and its adaptation for reuse).

Definition 18 (Online Case Base Maintenance Problem)**.** *Given a case base $\mathcal{L}$, and a new case $c = \langle \Pi, \pi \rangle$, $c \notin \mathcal{L}$, decide whether c should be added to $\mathcal{L}$.*

If all cases are added and kept in the library, the size of the library may grow very fast, significantly decreasing the performance of the whole system. Hence, an effective CBP system needs also to address the online maintenance problem.

The decision whether a new case should be added to the case base or not, and possibly also whether some other case should be removed, need to be taken relatively fast, as this is potentially done every time a new problem is presented to the system. The goal of the online policy is to maintain the case base in a way that supports efficient queries (case retrieval). To enable a fast retrieval, the case base needs to be small or well structured, and

BoundedOnlineMaintenance($\mathcal{L}, c, \delta, N$)

Input: a plan library $\mathcal{L}$, a case $c \notin \mathcal{L}$, a distance threshold δ, a positive integer N

Output: an updated plan library $\mathcal{L}'$

1. find $c^* \in \mathcal{L}$ s.t. $d(c, c^*) \leq d(c, c') \, \forall c' \in \mathcal{L}$;
2. **if** $d(c, c^*) > \delta$ **then** $\mathcal{L}' = \mathcal{L} \cup c$;
3. **if** $|\mathcal{L}'| > N$ **then**
4. $\quad c_i = \text{argmin}_{c_j \in \mathcal{L}'} d(c_j, c_j^*)$;
5. $\quad$ remove c_i;
5. $\quad$ update $d(c, c^*) \, \forall c \in \mathcal{L}$;
7. **return** $\mathcal{L}'$.

Figure 5: Algorithm for the bounded online maintenance policy.

hence the online system should add only new cases of significant "importance" (relatively to the existing cases) to the plan library.

The optimal criterion for deciding whether a new case is added or not should consider the deterioration of the retrieval step (the increased cost due to the enlarged case base), as well as the improvement of the reuse step (the benefit of possibly reusing the new case). However, such a measure depends not only on the case itself, but also on the rest of the case base, and hence it changes over time as the case base evolves. It is hard to predict the actual retrieval cost increase and reuse benefit before future problems are encountered by the system, without any assumption on the distribution of the planning problems. Therefore, the policy proposed here works by deciding the addition of the case solely based on the current state of the case base, allowing to remove this element later if the system identifies that it has become not useful.

As in the offline maintenance, the insertion of a new case is decided taking into account the diversity of the case with respect to the library. The case is included in the library only if it contains a sufficient amount of new information. More formally, let $c^* \in \mathcal{L}$ denote the element of the case base that is the most similar to a new case c according to distance function d. If $d(c, c^*) > \delta$ for some suitable distance threshold δ (that is, c is estimated to be sufficiently different from the whole case base), then c is inserted into $\mathcal{L}$.

However, such an approach can still lead to an uncontrollable growth of the plan library if all solved problems differ from each other more than δ. Therefore, we will address the *bounded* online maintenance problem defined as follows.

Definition 19 (Bounded Online Case Base Maintenance Problem)**.** *Given a case base $\mathcal{L}$, a bound $N \in \mathbb{N}$ on the maximum size of $\mathcal{L}$, and a case $c = \langle \Pi, \pi \rangle$ such that $c \notin \mathcal{L}$, decide whether c should be added to $\mathcal{L}$; if $|\mathcal{L}| = N$ and c should be added, find a suitable case in $\mathcal{L}$ to be removed from $\mathcal{L} \cup \{c\}$.*[3]

Figure 5 gives the pseudo-code of our algorithm for the bounded online maintenance of the plan library. The removal of a case is again guided by the case diversity in the case base.

3. Also the newly inserted case c is considered for removal because c might be the least valuable case in the augmented case base.

Informally, the policy attempts to identify a case less diverse than the others. Case diversity is defined by means of the distance function d. Let us recall that c_i^* denote the library case that is the most similar to a library case c_i. A case c_i is a candidate for removal if it satisfies $d(c_i, c_i^*) \leq d(c_j, c_j^*), \forall j \in \{1, \ldots, |\mathcal{L}|\}$. Algorithmically, such an element can be selected as follows: for every case $c_i \in \mathcal{L}$, we identify its most similar case c_i^*; we sort the case base elements by the growing distance with their most similar cases, and we remove the first such c_i. Such a removal can affect the definition of the minimum distance case for the cases that are still in the library; so, we also need to update the minimum case distance for the cases in the library such that their minimum case distance was c_i. In the worst case, updating these minimum distance cases requires quadratic time in the number of cases that are in the library. The algorithm in Figure 5 mimics a single iteration of the distance-guided policy, and requires only polynomial time if the distances to the most similar case have been stored during the case insertion phase.

3.3 Combined Maintenance

The offline and online approaches defined above have some complementary advantages and weaknesses. The offline policies have the potential drawback that between two consecutive maintenance operations the plan library could significantly and quickly grow. On the other hand, the online policies keep the library size limited to a given bound, but its quality is likely to degrade over time. In order to overcome these drawbacks and exploit the complementary advantages, we propose to combine the two approaches as described below.

The combined approach stores every new case in the library, but, in order to increase the performance during the retrieval, only a subset of the cases are considered; such cases are called *active cases*. The system behaves as if only the active elements were present in the case base, but it can dynamically redefine the set of active elements when a certain condition is met. The identification of the active elements is performed using the offline case base maintenance, which can be activated when the size of the case base exceeds a certain limit. All and only the elements in the reduction computed by the offline policy are considered active in the complete library.

In order to implement the combined maintenance policy, each case is marked by an *activation flag*. When the CBP system loads the library, it filters the cases by their activation. During the lifetime of the CBP system, for each newly encountered problem, the corresponding case is stored in the case base, and the bounded online maintenance is invoked to determine the activation flag of the new case and possibly modify the flag of another case. This is realised by an algorithm similar to the one given in Figure 5 for the bounded online maintenance, but with few changes:

- step 2 adds case c to the case base regardless of the distance from its closest case, and marks c active if and only if $d(c, c^\star) > \epsilon$, where ϵ is a distance threshold, and $c^\star$ is the *active* case which is closest to c in the library;
- step 4 considers only the active cases;
- step 5 marks case c_i as inactive, instead of removing c_i.

When a certain reduction condition is met, such as the system has encountered a maximum number of problems, or the retrieval time has increased above a specified limit, or the system

is in a period of low activity, etc., the planner marks every case as active and computes a suitable reduction of such a complete case base by means of the offline maintenance technique.

The fact that step 4 considers only the active cases implies that the system does not compute the distance values between all pairs of cases. Indeed, their computation during the reduction phase may significantly slow down the process: if the distance values between all pairs of cases (not only between the new case and the active ones) were computed during the insertion phase, overall the speed up when dealing with a new case would be less significant, as only the retrieval phase would be faster, while the insertion phase would take the same time as for the offline policies.

The main challenge of the combined maintenance is to identify a suitable condition at which the offline maintenance is invoked; this should not happen too often, due to its extensive computational costs, but also not too rarely to avoid an excessive deterioration of performance of the retrieval phase. Our experimental results reported in Section 5 suggest that even a very simple condition, such as a bound on the number of the encountered problems, leads to significant improvements.

4. Computational Complexity of Maintaining a Plan Library

In this section, first we study the complexity of the case base maintenance problem; then we analyze the worst-case time complexity of our policies.

4.1 Complexity of the Case Base Maintenance

We show that both the cardinality variant (Definition 11) and the quality variant of the plan-library maintenance problem (Definition 14) are NP-hard by proving that the corresponding decision problem versions are NP-complete. These NP-completeness proofs use reductions from the Dominating Set Problem (Garey & Johnson, 1990), where a dominating set for a graph $G = (V, E)$ is a subset V' of V such that every vertex not in V' is adjacent to at least one vertex of V'.

Definition 20 (Dominating Set Problem)**.** *Given an undirected graph $G = (V, E)$ and a natural number $\kappa \leq |V|$, the dominating set problem consists in testing whether there is a dominating set of size κ or less for G, that is, a subset $V' \subseteq V$, $|V'| \leq \kappa$, such that for all $u \in V \setminus V'$ there is a $v \in V'$ for which $(u, v) \in E$.*

The dominating set problem is NP-complete even under a number of various restrictions on the graph G (Garey & Johnson, 1990).

Lemma 1. *For every instance $\langle G, \kappa \rangle$ of the Dominating Set Problem there exists an instance of the Cardinality Case Base Maintenance Decision Problem $\langle \mathcal{L}, \delta, k \rangle$, for some distance threshold $\delta \in [0, 1]$ and $k = \kappa$, such that $\langle G, \kappa \rangle$ is a positive instance if and only if $\langle \mathcal{L}, \delta, k \rangle$ is a positive instance.*

Proof. We show how to create an instance $\langle \mathcal{L}, \delta, k \rangle$ of the Cardinality Case Base Maintenance Decision Problem from the undirected graph $G = (V, E)$ of the Dominating Set Problem, such that either both have a positive answer or they both have a negative answer. The

idea behind the reduction is to construct a plan library $\mathcal{L}$ where we have a case c for every vertex v of G, and encode in the initial state of c which other vertices are connected to v with an edge. The instance is then constructed as follows.

- For every graph vertex $v \in V$, we create a planning problem initial state $\mathcal{I}_v$ defined as the following set of propositions:

$$\{node(v)\} \cup \{edge(u,v) \,|\, u \in V,\, (u,v) \in E\} \cup \{edge(v,u) \,|\, u \in V,\, (u,v) \in E\},$$

  where proposition $edge(u,v)$ means that there is an edge between u and v in G. As G is an undirected graph, for every pair of vertices u and v connected by an edge in G both (u,v) and (v,u) are in $\mathcal{I}_v$ (and $\mathcal{I}_u$). Let n_v be the cardinality of $\mathcal{I}_v$ and $n = \max_{v \in V}(n_v)$. We extend every state $\mathcal{I}_v$ of cardinality less than n by additional "dummy" propositions (all different from each other) so that it contains exactly n propositions. Then the following holds: two initial states $\mathcal{I}_u$ and $\mathcal{I}_v$ share exactly two propositions, $edge(u,v)$ and $edge(v,u)$, if $(u,v) \in E$, otherwise their intersection is empty.

- For every graph vertex $v \in V$, we then construct a planning case c_v as follows: the initial state of c_v is $\mathcal{I}_v$, the action set is $\{act(v)\}$, the goal set is $\{goal(v)\}$, and the plan is $\langle act(v) \rangle$, where $act(v)$ is an action with set of preconditions $\{node(v)\}$ and set of effects $\{goal(v)\}$. The distance between two planning cases c_v and c_w constructed from a pair of vertices v and w such that $(v,w) \in E$ is:

$$d(c_v, c_w) = \alpha \cdot \left(1 - \frac{2+0}{n+1}\right) + (1-\alpha) \cdot \left(\frac{1+1}{1+1}\right) = \frac{n+1-2\cdot\alpha}{n+1},$$

  where α is the coefficient in the distance function definition. Otherwise, if the two planning case c_v and c_w correspond to a pair of vertices v and w that are not connected by an edge in G, their distance is:

$$d(c, c') = \alpha \cdot \left(1 - \frac{0+0}{n+1}\right) + (1-\alpha) \cdot \left(\frac{1+1}{1+1}\right) = 1.$$

- The distance threshold δ under which the maintenance policy removes cases is any value chosen in the interval $(\frac{n+1-2\cdot\alpha}{n+1}; 1)$, where the interval lower bound is the distance of two similar cases, whereas the interval upper bound is the distance between diverse cases. Note that the construction works for any value of $\alpha \in (0,1]$, because $\forall \alpha \in (0,1]$ there exists such a value of δ.

By construction of the case base and definition of δ, it is easy to see that, for every natural number $\kappa \leq |V|$ there exists a dominating set V' with $|V'| \leq \kappa$ if and only if the constructed case base can be reduced to a case base of size less than or equal to κ. Therefore, with $k = \kappa$, $\langle G, \kappa \rangle$ is a positive instance of the Dominating Set Problem if and only if $\langle \mathcal{L}, \delta, k \rangle$ is a positive instance of the Cardinality Maintenance Decision Problem. □

The main results of this section are formalised in the following theorems:

Theorem 1. *Given a case base $\mathcal{L}$, a case-distance threshold $\delta \in [0,1]$ and an integer $k \leq |\mathcal{L}|$, deciding whether there exists a reduced case base $\mathcal{L}' \subseteq \mathcal{L}$ according with δ such that $|\mathcal{L}'| \leq k$ is NP-complete .*

Proof. The NP-membership can be proven by the following simple guess-and-check algorithm:

- Guess a subset $\mathcal{L}'$ of $\mathcal{L}$ of size $|\mathcal{L}'| \leq k$.
- Then we check if $\mathcal{L}'$ reduces $\mathcal{L}$, i.e., $\mathsf{coverage}_\delta(\mathcal{L}, \mathcal{L}') = 1$, by building the set C_δ and verifying that $C_\delta = \mathcal{L}$. We can (naively) build C_δ as follows: $\forall c' \in \mathcal{L}'$ compute $d(c', c)$, $\forall c \in \mathcal{L}$, and if $d(c', c) \leq \delta$ then add c to C_δ. Constructing C_δ and verifying that $C_\delta = \mathcal{L}$ requires $\mathcal{O}(|\mathcal{L}'| \cdot |\mathcal{L}|)$, i.e., $\mathcal{O}(|\mathcal{L}|^2)$.

Hence, given a case base $\mathcal{L}'$, we can decide in polynomial time and space whether $\mathcal{L}'$ indeed reduces $\mathcal{L}$, which proves the membership in NP.

The NP-hardness follows from Lemma 1, the NP-completeness of the Dominating Set Problem, and the fact that the reduction used in Lemma 1 requires polynomial time and space. This is because in the reduction from an instance $\langle G, \kappa \rangle$ of the Dominating Set Problem, with $G = \langle V, E \rangle$, we have that: the total number of initial states, sets of goals, sets of actions and plans is $4 \cdot |V|$; every created initial state $\mathcal{I}_v$ contains at most $2 \cdot |E| + 1$ propositions; every created set $\mathcal{G}_v$ of goals contains one proposition; every created set of actions $\mathcal{A}_v$ and every created plan π_v contains one action. Moreover, this construction examines each state only twice — once to create it and once to make sure it contains exactly n propositions. □

Similarly, we show that finding the reduced case base that minimizes the reduction cost is intractable, i.e., the Quality Case Base Maintenance Problem is also NP-complete.

Theorem 2. *Given a case base $\mathcal{L}$, a case-distance threshold $\delta \in [0,1]$, and a real number k, it is NP-complete to identify whether there exists a reduced case base $\mathcal{L}' \subseteq \mathcal{L}$ according to δ such that the reduction cost of $\mathcal{L}'$ is k or less.*

Proof. The NP-membership is proven by a guess-and-check algorithm similar to the one described in the proof of Theorem 1. The difference is that this algorithm verifies that the reduction cost of $\mathcal{L}'$ is at most k, instead of $|\mathcal{L}'| \leq k$. Such a verification can be done in time that is quadratic in the input size.

The NP-hardness can be proven by using the same method described in the proof of Lemma 1 to reduce instances of Dominating Set Problem into instances of the Cardinality Case Base Maintenance Decision Problem. Let $\langle G, \kappa \rangle$ be an instance of the dominating set problem. Without loss of generality, we can assume that graph $G = \langle V, E \rangle$ does not contain isolated vertices. If it contained a set W of isolated vertices, the instance $\langle G, \kappa \rangle$ would be equivalent to the instance $\langle G', \kappa - |W| \rangle$ with $G' = \langle V - W, E \rangle$.

Let $\mathcal{L}$ be the case base derived by the reduction of $\langle G, \kappa \rangle$ described in the proof of Lemma 1. With the same case-distance threshold δ and definition of values n and α as in proof of Lemma 1, the distance between any case c of $\mathcal{L}$ and its neighbors is $\frac{n+1-2\cdot\alpha}{n+1}$. Therefore, it is easy to see that, according to Equations 3 and 4, the value of $v_\delta(c)$ for any case c of $\mathcal{L}$ is either 1 if $L_\delta(c) = \emptyset$, or $1 + \frac{n+1-2\cdot\alpha}{n+1}$ otherwise.

We show that $\langle G, \kappa \rangle$ is a positive instance of the Dominating Set Problem iff $\langle \mathcal{L}, \delta, k \rangle$ is a positive instance of the Quality Case Base Maintenance Decision Problem with $k = \kappa \cdot \left(1 + \frac{n+1-2\cdot\alpha}{n+1}\right)$. Assume that there exists a dominating set of G with at most κ vertices, and consider the reduction $\mathcal{L}'$ of $\mathcal{L}$ corresponding to such a dominating set. I.e., the cases of $\mathcal{L}'$ are those and only those corresponding to the vertices in the dominating set according with the construction of the case base described in the proof of Lemma 1. By definition, reduction cost $\mathcal{M}_\delta(\mathcal{L}') = \Sigma_{c \in \mathcal{L}'} v_\delta(c)$. Since, $|\mathcal{L}'| \leq \kappa$, and $v_\delta(c)$ is either 1 or $1 + \frac{n+1-2\cdot\alpha}{n+1}$, then $\mathcal{M}_\delta(\mathcal{L}') \leq \kappa \cdot (1 + \frac{n+1-2\cdot\alpha}{n+1})$ and hence, with $k = \kappa \cdot \left(1 + \frac{n+1-2\cdot\alpha}{n+1}\right)$, $\mathcal{L}$ is a positive instance of the Quality Case Base Maintenance Decision Problem.

Assume now that $\langle \mathcal{L}, \delta, k \rangle$ is a positive instance of the Quality Case Base Maintenance Decision Problem with $k = \kappa \cdot \left(1 + \frac{n+1-2\cdot\alpha}{n+1}\right)$. Then, there exists a reduced case base of $\mathcal{L}$ such that its reduction cost is $\kappa \cdot \left(1 + \frac{n+1-2\cdot\alpha}{n+1}\right)$ or less. Let $\mathcal{L}' \subseteq \mathcal{L}$ be the reduced case base of $\mathcal{L}$ with the minimum reduction cost, and κ' be the number of cases of $\mathcal{L}'$. The library $\mathcal{L}$ derived from a graph G without isolated vertices by the reduction described in the proof of Lemma 1 is such that $\mathcal{L}$ does not contain cases whose neighborhood is empty. Since $\mathcal{L}' \subseteq \mathcal{L}$, this property also holds for $\mathcal{L}'$. Suppose that there exists a case c of $\mathcal{L}'$ that covers no case in $\mathcal{L} \setminus \mathcal{L}'$. We distinguish three possible cases in which this can happen, and we show that each of them contradicts the hypothesis that $\mathcal{L}'$ is minimal (and thus that every case in $\mathcal{L}'$ covers at least one case in $\mathcal{L} \setminus \mathcal{L}'$).

(1) There exists exactly one case c of $\mathcal{L}'$ that covers no case in $\mathcal{L} \setminus \mathcal{L}'$. This implies that the non-empty set of cases that are neighbors of c in $\mathcal{L}$ are also in $\mathcal{L}'$, $L_\delta(c) = \emptyset$ and, according with Equation 4, $v_\delta(c) = 1$. So, there would exist a reduction $\mathcal{L}' \setminus \{c\}$ of $\mathcal{L}$ with reduction cost that is one unit lower than the minimum reduction cost, as the cost of the other cases in $\mathcal{L}' \setminus \{c\}$ remains the same. Indeed, by the definition of case cost given in formula (4) and construction of $\mathcal{L}'$, when a case is removed from $\mathcal{L}'$ the cost of the other cases in $\mathcal{L}'$ can only increase, the maximum cost of a case is $1 + \frac{n+1-2\cdot\alpha}{n+1}$, and all the cases in $\mathcal{L}' \setminus \{c\}$ already have such a maximum cost.

(2) There exists a set U of cases of $\mathcal{L}'$ each of which covers no case in $\mathcal{L} \setminus \mathcal{L}'$, $|U| > 1$, and $\mathcal{L}' \setminus U$ is a reduction of $\mathcal{L}$. Then, all the cases in U are covered by at least one case in $\mathcal{L}' \setminus U$, and hence there would exist a reduction $\mathcal{L}' \setminus U$ of $\mathcal{L}$ with reduction cost that is $|U|$ units lower than the reduction cost of $\mathcal{L}'$.

(3) This case is the same as case (2) with the difference that $\mathcal{L}' \setminus U$ is not a reduction of $\mathcal{L}$. Let U' be the minimal set $U' \subset U$ such that $\mathcal{L}' \setminus (U \setminus U')$ is still a reduction of $\mathcal{L}$. Note that $U' \neq U$ because, as for case c in (1), each case in U has its neighbor cases in $\mathcal{L}'$. Let $\mathcal{L}''$ be the case base derived by removing from $\mathcal{L}'$ the (non-empty) subset of cases that are covered by a case in either U' or $\mathcal{L}' \setminus U$, i.e., $\mathcal{L}'' = \mathcal{L}' \setminus (U \setminus U')$. $\mathcal{L}''$ is still a reduction of $\mathcal{L}$ and we show that $\mathcal{M}(\mathcal{L}'') < \mathcal{M}(\mathcal{L}')$, which contradicts the assumption that $\mathcal{L}'$ is a minimal reduction of $\mathcal{L}$.

The maximum possible number q of cases of U that are in $\mathcal{L}''$ is less than or equal to the number r of cases that are removed from $\mathcal{L}'$. In particular, we have q cases when each of them covers exactly one case c among those in U that are removed from $\mathcal{L}'$ to obtain $\mathcal{L}''$, and such a case c is not covered by another case in $\mathcal{L}' \setminus U$; under these

conditions, $q = r = \frac{|U|}{2}$. Thus, the number of cases in U that are in $\mathcal{L}''$ are less than or equal to q, i.e., $|U'| \leq q$. It follows that, if these q cases increase their costs in $\mathcal{L}''$ (because of the removal of cases in $U \setminus U'$ from $\mathcal{L}'$), their costs can globally increase by at most $q \cdot \frac{(n+1-2\alpha)}{(n+1)}$. Moreover, since every case in $\mathcal{L}' \setminus U$ cannot increase its cost in $\mathcal{L}''$ (because by construction of $\mathcal{L}'$ it must already be the maximum cost), we have that the cost of $\mathcal{L}''$ decreases by at least $q - r \cdot \frac{(n+1-2\alpha)}{(n+1)} > 0$ w.r.t. the cost of $\mathcal{L}'$. Hence, $\mathcal{M}(\mathcal{L}'') < \mathcal{M}(\mathcal{L}')$, which contradicts the hypothesis that the cost of $\mathcal{L}'$ is minimal.

Finally, we show that from $\mathcal{L}'$ we can build a solution to $\langle G, \kappa \rangle$. By construction of $\mathcal{L}$ and $\mathcal{L}'$, every case c_i in $\mathcal{L}'$ corresponds to a vertex v_i of G. Let $\{c_1, \ldots, c_{k'}\}$ be the set of cases forming $\mathcal{L}'$. We have that the set of vertices $S = \{v_1, \ldots, v_{k'}\}$, where each v_i corresponds to case c_i in $\mathcal{L}'$, is a dominating set solving $\langle G, \kappa \rangle$. This is because: (i) we have shown that $|S| = \kappa' \leq \kappa$; (ii) since $\mathcal{L}'$ is a reduction of $\mathcal{L}$, for each $c_j \in \mathcal{L} \setminus \mathcal{L}'$, the corresponding vertex v_j of G has an edge (v_j, v_t) such that $c_t \in \mathcal{L}'$ (otherwise $\mathcal{L}'$ would not be a reduction, since c_j would be uncovered in $\mathcal{L}'$ by Definition 8 and construction of $\mathcal{L}$). Thus, for each $v_j \notin S$ there exists an edge (v_j, v_t) such that $v_t \in S$, i.e., S is a dominating set solving the instance $\langle G, \kappa \rangle$. □

In the reminder of this section, we theoretically compare the computational complexity of the algorithms used to compute an approximate solution of the maintenance problem.

4.2 On the Complexity of the Policies

We distinguish two kinds of policies: informed policies, which define a relationship among the cases and use the information to select the cases to be kept; and uninformed policies, which select cases based on other criteria. Specifically, informed policies compute the distance between any pair of cases, and then reduce the case base on the basis of the computed distances. As there are two tasks to be performed and not both of them are necessarily performed, we split the complexity accordingly. Remember that the informedness of a policy comes at a cost of identifying the relationship among cases, which in our context translates to resolving $\mathcal{O}(|\mathcal{L}|^2)$ instances of the object matching problem, which is NP-hard in the size of the planning problem (we denote the required time by t_d). Table 1 summarises the complexity results related to the offline computation of the reduced case base. The size of the reduced case base is affected by the probability value q in the random policy and by the value of δ in the other policies.

The proposed policies vary significantly in the amount of information they take into account and the operations they apply to decide the set of cases to remove. The random policy needs to remove cases until a case base of the desired size is derived. Moreover, it does not care about the distances, and so the computation of the cases distribution can be avoided. Hence the time required is asymptotically linear in the size of the case base. The other policies are clearly better informed than the random policy, and they can recognise cases of high importance for the coverage of the case base (e.g., isolated elements that are dissimilar to any other case). The better information is however reflected by an increased computational complexity – the distance-guided policy needs to identify the closest retained case for every case, which requires the computation of all the distances and hence a work

Policy (parameter)	Retain c_i by	Informed	Complexity D.	Complexity M.
random (q)	$\text{rand}(0,1) \geq q$	no	– –	$\mathcal{O}(\lvert\mathcal{L}\rvert)$
distance (δ)	$d(c_i, c_i^*) \geq \delta$	yes	$\mathcal{O}(\lvert\mathcal{L}\rvert^2) \cdot t_d$	$\mathcal{O}(\lvert\mathcal{L}\rvert^2)$
cardinality coverage (δ)	$\max \lvert L_\delta(c_i)\rvert$	yes	$\mathcal{O}(\lvert\mathcal{L}\rvert^2) \cdot t_d$	$\mathcal{O}(\lvert\mathcal{L}\rvert^2)$
quality coverage (δ)	$\min \left\{ \frac{v_\delta(c_i)}{\lvert L_\delta(c_i)\rvert} \right\}$	yes	$\mathcal{O}(\lvert\mathcal{L}\rvert^2) \cdot t_d$	$\mathcal{O}(\lvert\mathcal{L}\rvert^2)$

Table 1: Offline maintenance policies and their complexity for computing the case distances (Complexity D.) and the case base reduction (Complexity M.).

Policy	CB size	Frequency	Procedure	Complexity
offline	finite	on request	coverage-guided	$\mathcal{O}(\lvert\mathcal{L}\rvert^2) \cdot t_d$
online	finite	every time	distance-guided	$\mathcal{O}(\lvert\mathcal{L}\rvert) \cdot t_d$
bounded online	bound	every time	distance-guided	$\mathcal{O}(\lvert\mathcal{L}\rvert^2) \cdot t_d$
combined	bound	every time on condition	distance-guided + coverage-guided	$\mathcal{O}(\lvert\mathcal{L}\rvert^2) \cdot t_d$

Table 2: Maintenance frequency, case base size, and complexity for the different considered strategies.

quadratic in the size of the case base. In addition, the selection of the case with the minimum distance from any other case in the library requires a work $\mathcal{O}(|\mathcal{L}| \cdot \log |\mathcal{L}|)$ time; to check whether an element has to be removed from $\mathcal{L}$ needs computing the distance between the element and its minimum distance case in the current reduction. This latter operation can be accomplished in linear time w.r.t. the number of cases in the library, and it has to be repeated a number of times that is asymptotically linear in the size of the library; hence the time complexity of the case base reduction done by the distance-guided policy is quadratic in the size of the case base. Similarly, in addition to the computation of all the distances, the coverage-guided policies requires running the greedy algorithm described in Figure 3, whose time complexity is also quadratic in the size of the case base.

We have already said that the offline, online and combined maintenance perform the same procedures guided by either the distance or the coverage, but differ in the condition invoking their application. Basically, the offline maintenance is performed upon a request from the user, the online maintenance is performed in every iteration of the case-based cycle, and the combined maintenance when a certain condition is met. Therefore, even though the performed procedures are algorithmically the same, the runs differ in the frequency with which they are performed, and also in the size of the case base they are reducing, as summarised in Table 2. The procedure with the best theoretical complexity is the online policy simply because it applies to only one case, i.e., the new case encountered by the CBP system. Note that even though the offline, bounded online and combined strategies have the same theoretical complexity, the size of the case bases may differ greatly, because the size of the case base in the bounded online and combined strategies is bound. Concerning the combined maintenance, the size is bounded by the switching condition which invokes the coverage-guided policy. On the other hand, the offline strategies may witness an

uncontrolled growth of the case base prior to the maintenance request. This is likely to result in significant difference in the run-time behavior, where the combined strategies are likely to reduce smaller case bases, and hence require less time. If the computation of the distances among the cases is fast, because, e.g., there is no need of matching names of problem objects, then, in practice, the coverage-guided maintenance may be computationally the most expensive step in the retain phase.

5. Experimental Results

We present here a thorough collection of experiments. Two different sets of benchmark problems were used with the aim of evaluating our approach when problems tend to recur with different chances. The main goals of our experiments are:

- evaluating the importance of measuring the distance between cases as a linear combination of the distance between case problems and between case plans (Section 5.2);
- evaluating the quality and effectiveness of maintaining the case base using different offline policies (Section 5.3);
- analyzing the usefulness of maintaining the case base using the bounded online policy and the combined policy (Section 5.4);
- evaluating the efficacy of adapting previously stored plans w.r.t. replanning from scratch (Section 5.5).

5.1 Experimental Settings

The techniques presented in the previous sections have been implemented in a new version of the CBP system OAKplan (Serina, 2010) that, to the best of our knowledge, is the state-of-the-art for CBP systems. In our experiments, the plan retrieved by OAKplan is adapted using planner LPG-td (Gerevini, Saetti, & Serina, 2003, 2006, 2011; Fox, Gerevini, Long, & Serina, 2006), which supports planning from an input (partial) plan. However, other plan adaptation systems, such as ADJ (Gerevini & Serina, 2010), can be used in OAKplan.

The retrieval of a case in OAKplan requires a linear scan of the cases in the library. With a linear scan, OAKplan makes sure to identify the most promising case in the library. This procedure is used by several other case-based reasoning systems, such as Rascal (Raymond, Gardiner, & Willett, 2002). Other CBP systems adopt a hierarchical retrieval. For example, Far-Off (Tonidandel & Rillo, 2002) and Prodigy/Analogy (Veloso, 1994) use the Footprint-based method for the retrieval procedure; i.e., they determine a set of footprint cases with the same competence of the case base. Then, their retrieval procedure consists of two phases: first, it finds the most promising footprint case among all footprint cases, and second, it looks for a promising case among the library cases that are similar to the chosen footprint case. The hierarchical retrieval might be faster than the linear scan of the library used by OAKplan, at the expense of the possibility of retrieving the most promising case to adapt. Specifically, the retrieval of a case in OAKplan consists of three main steps:

(i) for every case $\langle \Pi, \pi \rangle$ in the library, OAKplan changes the names of the objects in Π to names of objects in the new encountered problem Π'; as a consequence of the object renaming, the names of the actions of π are also changed accordingly;

(ii) for every object-renamed library case $\langle \Pi, \pi \rangle$, OAKplan measures the similarity between Π and Π', and derives the set of the library cases with problems mostly similar to Π';

(iii) for every object-renamed library case $\langle \Pi, \pi \rangle$ in the set of the mostly similar ones, OAKplan estimates the heuristic cost of adapting the library plan π to a solution plan of Π'.

The purpose of step (i) is mapping the object names of different problems into each other names so that the (structural) similarity of the problems increases. The purpose of (ii) is identifying the (sub)set of library problems that are mostly similar to the new encountered problem Π'. The purpose of (iii) is evaluating the library plans for the identified set of the most similar library problems in order to select the library plan that is most promising to adapt for deriving a solution of Π'. The problem of computing the object matching that maximizes the similarity between a pair of planning problems is NP-hard (Serina, 2010). OAKplan approximates the problem of finding the best matching between the objects of a pair of planning problems Π and Π' to the (polynomial) problem of computing the best association between the vertices of two special graphs, called *planning problem graphs* of Π and Π'. Planning problem graphs are particular graphs that represent the objects, predicates, initial state, and goals of a planning problem. Informally, the best association between the vertices of the planning problem graphs of Π and Π' is the association that maximizes the sum of the values of a similarity function between pairs of vertices in the two graphs (for more details the reader is referred to (Serina, 2010)). Unfortunately, computing this approximation for many (similar) cases in the library is computationally expensive and can become the bottleneck of the OAKplan's CBP process. Indeed, the time required for adapting a plan is usually relatively low w.r.t. the time required for the retrieval of the plan from a large library containing many similar cases. Therefore, maintaining a library with a limited size, and formed by few cases that are representative of a cluster of similar cases, can help to significantly reduce the time required for the retrieval.

The adaptation of an input plan in LPG-td consists of two main steps. First, LPG-td constructs a representation of the input plan, called *action graph*; then LPG-td uses a local search procedure that iteratively applies some graph modifications to transform the initial action graph into a solution. At each search step, LPG-td selects a flaw σ (unsupported precondition) in the current action graph. In order to resolve σ, LPG-td either adds an action node achieving σ into the current action graph or removes an action node having the selected flaw as a precondition. Given an action graph $\mathcal{A}$ and a flaw σ in $\mathcal{A}$, the local search neighborhood of $\mathcal{A}$ for σ is the set of action graphs obtained from $\mathcal{A}$ by applying a graph modification that resolves σ. At each step of the local search procedure of LPG-td, the elements of the neighborhood are evaluated according to a heuristic function estimating their quality, and an element with the best quality is then chosen as the next possible action graph (search state).

All the experimental tests were run on an Intel Xeon(tm) 2 GHz machine with 20 Gbytes of RAM. The CPU-time limit for each run of the experimented planners was 30 minutes,

after which termination was forced. Since LPG-td uses a randomized search algorithm, the results that we give are median values over five runs. When a problem OAKplan exceeds the CPU-time limit for more than two of the five runs, the problem is considered unsolved.

5.1.1 Plan Libraries and Benchmarks

The benchmark domains used in our experimental analysis are Driverlog, Logistics, Rovers, Satellite, Zenotravel and Elevators. The first five domains are from the 3rd international planning competition, and have been selected for a practical reason: the plan libraries used in the experimental analysis are constructed by solving many large problems, hand-coded domain-specific knowledge is available for these domains (Long & Fox, 2003), and this allows solving these (large) problems in a few seconds. Specifically, the plan libraries of these domains were generated using planner TLPlan (Bacchus & Kabanza, 2000), which exploits domain-specific search control knowledge to guide a forward chaining search. The sixth domain, Elevators, is from the 6th international planning competition (Helmert, Do, & Refanidis, 2008), and it was used to show that the conclusions derived from the plan libraries generated by TLPlan do not depend on the use of this specific planner, given that similar results can be obtained using a radically different planner. In particular, the plans in the library for Elevators were generated by FF (Hoffmann, 2003), which solves each of the large Elevators problems forming the library using from 1 to 30 CPU minutes without domain-specific control knowledge.

With the aim of evaluating our approach when problems tend to recur with different chances, for each considered domain we generated two sets of 6000 cases each, L1 and L2. The cases in L1 are grouped into a high number of small and middle-size clusters, while the cases in L2 are grouped into a much lower number of middle-size and large clusters. Therefore, in L2 problems tend to recur more often than the problems in L1. Specifically, first, we generated a number of problems randomly; then, for each of these problems, we randomly determined a cluster size (from 2 to 100 for L1, and from 200 to 500 for L2) and generated a problem cluster of such a size; finally, we iterated this process until the case base contains 6000 elements. As a result, L1 includes a number of cases ranging from 97 to 134 over different domains; each of these cases consists of a randomly generated problem and its plan, plus, for each case c among these, a random number of cases ranging from 2 to 100 obtained by changing the planning problem Π_c of c and solving the changed problem by either TLPlan or FF. Problem Π_c was modified either by randomly changing a number of literals (initial facts and/or goals) ranging from 1 to 20 or adding/deleting an object to/from the problem. For each domain, the number of cases of L2 with randomly generated problems ranges from 18 to 22, while the number of cases of L2 derived by changing the propositions or adding/deleting an object of the generated problems ranges from 200 to 500. Overall, the number of problem objects ranges from 25 to 320; the number of literals in the initial states ranges from 16 to 152; the number of literals in the sets of goals ranges from 8 to 257; the number of actions in the plans ranges from 29 to 664. In the rest of Section 5, we use L1 and L2 also to indicate the libraries obtained from sets L1 and L2; their specific meaning will be clear from the context.

Moreover, for each considered domain, we generated 25 test problems obtained from randomly selected problems in the libraries by (i) changing all object names, and (ii) per-

forming a number of modifications to the problem propositions (initial facts and/or goals) ranging from 1 to 10. The test problems were generated starting from the library problems because the aim of our experimental analysis is studying the effectiveness of the proposed techniques for domains with recurring problems. It is worth noting that the test problems generated for our experimental analysis are quite challenging for a domain-independent generative planning approach, since we observed that the state-of-the-art planners LAMA (Richter & Westphal, 2010) and BFWS (Lipovetzky & Geffner, 2017) solve only about 75% of these problems. Further details on the number of problems solved by these planners will be given in Section 5.5.

The problems and solutions used to construct the libraries, the libraries used for the experimental study, the test problems, as well as the executable of OAKplan, are made available from `https://lpg.unibs.it/OAKplan/`.

5.1.2 Performance Evaluation

Different maintenance policies can compute reduced case bases of different sizes. Comparing the usefulness of case bases of different sizes could be questionable since they contain a different amount of knowledge. The maintenance policies presented in Section 3 compute a reduction of a case base $\mathcal{L}$ according to a given distance threshold δ, i.e., a smaller case base $\mathcal{L}'$ such that $\mathsf{coverage}(\mathcal{L}', \mathcal{L}) = 1$ w.r.t. δ. So, we used the maintenance policies to compute reductions according to different values of δ with the goal of deriving reduced case bases of a predetermined size. Specifically, a given case base of 6000 cases is reduced to libraries of 3000, 2000, 1000, 500, 250, and 100 cases using values of δ estimated as the 50th, 67th, 83th, 92th, 96th, and 98th percentiles of the average minimum distance distribution, respectively. The process of reducing a case base with a value for δ and the corresponding number n of cases for the reduced case base is stopped when the number of elements in the reduced case base exceeds n. If the reduced case base has less than n cases, the reduction process is repeated using an increased value of δ. More precisely, the process is repeated with a case-distance threshold equal to the least minimum distance case value greater than δ among those of cases in the library.

The quality of a reduced case base $\mathcal{L}'$ is measured in terms of (a) its coverage over the original case base with 6000 cases, and (b) number of cases of the original case base that are not covered by the reduced case base. Both these measures depend on the value δ. For our analysis, we measured them using a case-distance threshold equal to the *average minimum distance* δ_μ of the cases in the original case base. I.e., δ_μ is defined as $\Sigma_{c_i \in \mathcal{L}} \frac{d(c_i, c_i^*)}{|\mathcal{L}|}$, where c_i^* denotes the minimum distance case.[4] Obviously, reduced case bases with higher coverage (or equivalently with fewer uncovered cases) correspond to better maintenance policies.

The effectiveness of a maintenance policy P is evaluated by using OAKplan to retrieve cases from the case bases reduced by P, and using LPG-td to adapt the retrieved plan cases to solve the test problems. It is measured in terms of average normalized plan distance and speed score, which are defined as follows. Given a case plan π' and a plan π computed by adapting π' for a new problem, the *plan distance* of π with respect to π' is $|\pi \ominus \pi'| + |\pi' \ominus \pi|$. The lower the plan distance, the better. Having a low value of plan distance can obviously be

4. The isolated cases are excluded in the computation of δ_μ. In this policy, a case c_i is considered isolated if distance $d(c_i, c_i^*) \geq 0.5$.

very important, because, e.g., low distance reduces the cognitive load on human observers of a planned activity by ensuring coherence and consistency of behaviours (Fox et al., 2006). Therefore, a good maintenance policy should allow the generation of reduced case bases from which a CBP system can produce solutions similar to case plans. Given a set of compared policies, for each policy in the set, the *average plan distance* is computed over the test problems that are solved by all the compared policies.

The speed score function used in this paper was introduced by the organisers of the 8th International Planning Competition (Vallati, Chrpa, Grzes, McCluskey, Roberts, & Sanner, 2015) for evaluating the relative performance of the competing planners, and it has become a standard method for comparing the performance of planning systems. The speed score for a maintenance policy P is defined as the sum of the speed scores assigned to OAKplan using P over all the test problems. The speed score for P with a planning problem Π is defined as: 0, if Π is unsolved using policy P; 1, if Π is solved within one second; and $1 - \frac{log(T(\mathsf{P},\Pi))}{\mathbf{T}}$, otherwise, where $T(\mathsf{P}, \Pi)$ denotes the CPU time required to solve problem Π using the case base reduced through policy P, and $\mathbf{T}$ is the CPU-time limit that we used for our experiments. Higher values of the speed score indicate better performance. Therefore, a good maintenance policy should allow the generation of reduced case bases from which a CBP system can quickly solve new encountered problems. Finally, for the comparison with generative planners, we evaluated the quality of the generated plans in terms of average plan length and total quality score. The quality score of a planner p for a problem Π is the ratio between the length of the shortest compared plan and the length of the plan computed by p for Π. As for the speed score, higher values of the total quality score indicate better performance.

5.2 Plan Distance versus Problem Distance

All the informed policies we study rely on the distance function $d : ((\mathcal{P} \times \wp) \times (\mathcal{P} \times \wp)) \rightarrow [0, 1]$, which measures the similarity between cases. In our work, distance function d is a linear combination of the problem distance function d_p and the normalized plan distance function d_π, i.e., $d = \alpha \cdot d_p + (1 - \alpha) \cdot d_\pi$. Such a combination allows us to assign different importance to the similarity of problems and their plans. The first experiment we conducted evaluates the effectiveness of maintaining the plan library by using function d with three different values of parameter α. Specifically, we tested:

- $\alpha = 1$, i.e., the distance between a pair of cases is measured only in terms of similarity of their case problems,
- $\alpha = 0$, i.e., the distance between a pair of cases is measured only in terms of similarity of their case plans;
- $\alpha = 0.5$, i.e., the importance assigned to the similarity between case problems and between case plans is the same.

Figure 6 gives the speed score and the average normalized plan distance obtained from the case bases in sets $\mathsf{L1}$ and $\mathsf{L2}$ derived using the quality coverage-guided policy with the three considered values of parameter α. These results show that, for libraries reduced from set $\mathsf{L1}$ with a number of cases lower than 2000, OAKplan is faster and computes plans closer

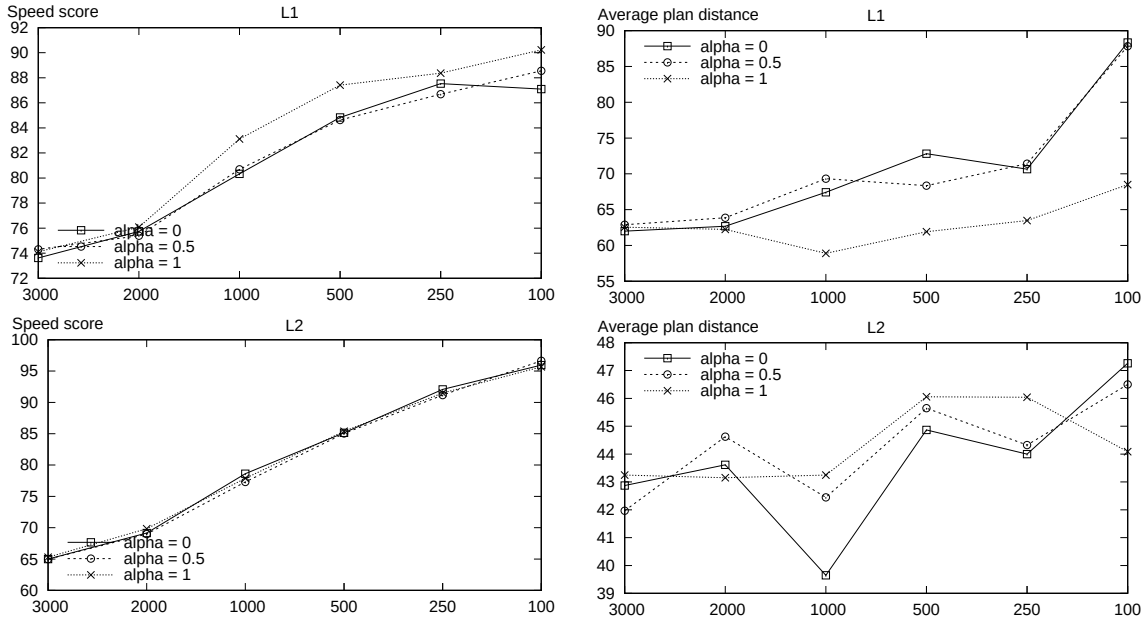

Figure 6: Total speed score and average plan distance of OAKplan with plan libraries reduced from the original libraries in sets L1 and L2 using three different measures of the distance between cases corresponding to $\alpha = 0$, $\alpha = 1$, and $\alpha = 0.5$. On the x-axis, there are the numbers of cases in the reduced plan libraries.

to the retrieved library plans when using $\alpha = 1$. For most of the reduced libraries obtained from set L2, OAKplan almost always computes plans closer to the retrieved library plans using $\alpha = 0$, while in terms of CPU time the results in Figure 6 show that the value of α does not impact on the performance of OAKplan significantly. Overall, we can derive the following conclusion.

Experimental result 1. *When the case clusters forming the library are small, the library cases are well characterized by the problem cases. When the case clusters are large, the libraries can contain many similar case problems, and hence the library cases tend to be better characterized by the plan cases than by the problem cases.*

In the rest of the paper, the case bases in L1 are reduced by measuring the case distance with $\alpha = 1$ (i.e., through the problem distance); on the contrary, for the case bases in L2 the case distance is measured with $\alpha = 0$ (i.e., through the normalized plan distance).

5.3 Offline Policies

The experimental analysis of this section compares different offline policies with the aim of evaluating the quality and the effectiveness of maintaining the case base using them. In the following, R, D, CC and QC abbreviate the random, distance-guided, cardinality coverage-guided and quality coverage-guided policy, respectively. Figure 7 shows the coverage and the number of uncovered cases of the plan libraries reduced from the original libraries in sets L1 and L2 through R, D, CC, and QC.

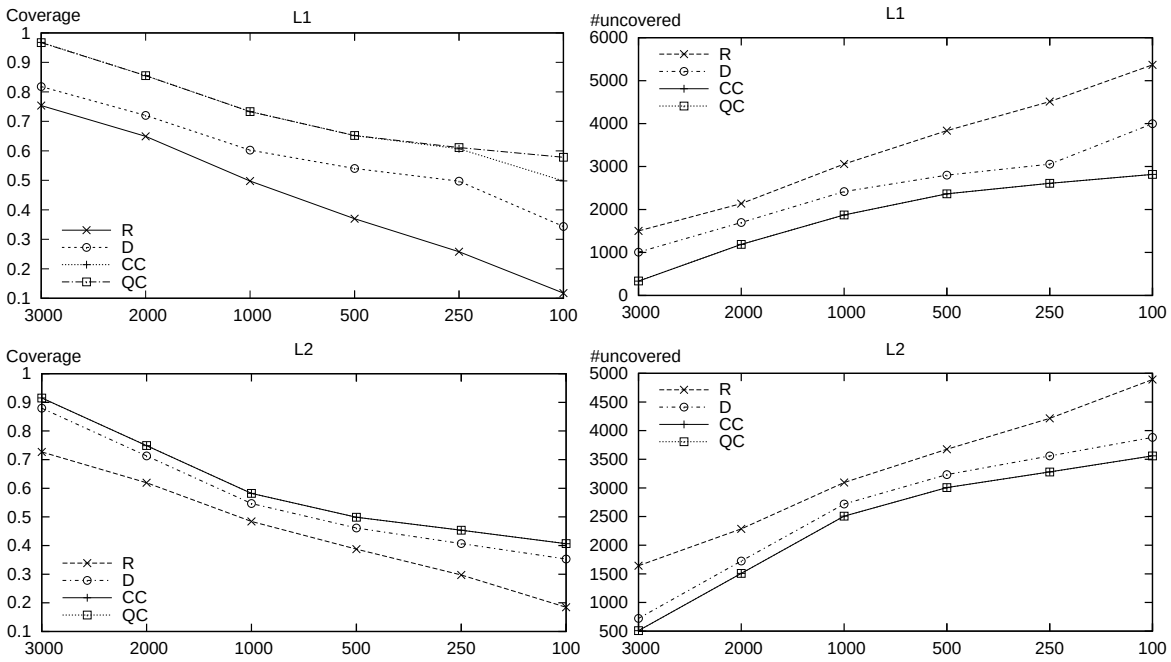

Figure 7: Coverage and number of uncovered cases of the plan libraries reduced from the original libraries in sets L1 and L2 through policies R (Random), D (Distance-guided), CC (Cardinality Coverage-guided), and QC (Quality Coverage-guided). On the x-axis, there are the numbers of cases in the reduced plan libraries.

Experimental result 2. *In terms of both coverage and number of uncovered cases, for almost every considered size of the reduced case bases, the cardinality coverage-guided policy performs as well as the quality coverage-guided policy; both perform better than the distance-guided and the random policies; finally, as expected, the distance-guided policy performs better than the random policy.*

Figure 8 shows the total speed score and the average plan distance of OAKplan with the analyzed reduced plan libraries.

Experimental result 3. *The speed score obtained using the libraries reduced through* CC *is consistently the best.*

As for the performance gap between CC and QC, we observe that their retrieval time is similar, while the adaptation time is usually better for CC. We conjecture that, for our test problems, the plan library computed by CC selects a set of cases that is more prominent than QC. For every reduced case base from L1 and L2 with a number of elements lower than or equal to 1000, we can also observe that the speed score derived using QC is better than using D and R; and, finally, OAKplan with D performs almost always better than or similar to with R.

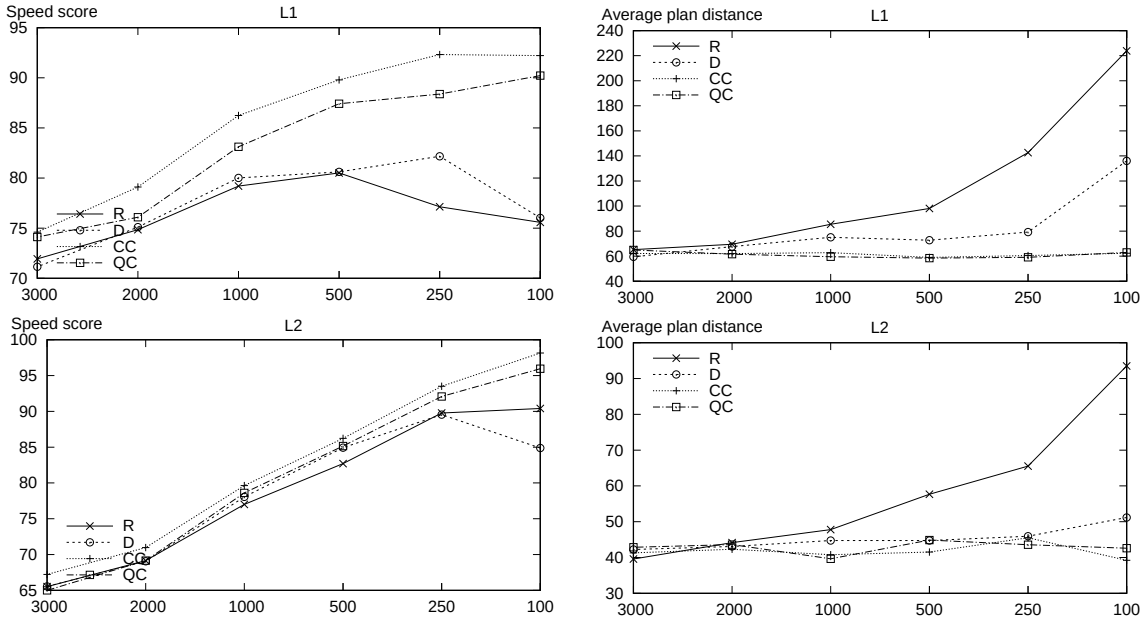

Figure 8: Total speed score and average plan distance of OAKplan using plan libraries reduced from the original libraries in sets L1 and L2 through policies R (Random), D (Distance-guided), CC (Cardinality Coverage-guided), and QC (Quality Coverage-guided). On the x-axis, there are the numbers of cases in the reduced plan libraries.

Experimental result 4. *The average distance between the adapted plan and the plan retrieved by OAKplan using the libraries reduced through* CC *is similar to* QC*, and better than using other policies.*

Specifically, for library set L1 and every analyzed case base with size lower than or equal to 2000, in terms of average plan distance OAKplan using the plan libraries reduced through either CC or QC performs better than using D and R; as expected, the average plan distance using D is better than using R. For library set L2, OAKplan performs similarly using either CC, QC, or D; for every analyzed case base size lower than or equal to 1000, OAKplan using R performs again much worse than using any other considered policy.

Remarkably, as shown in Figure 8, for the reduced case bases from set L1 with 3000 elements, OAKplan with the plan libraries reduced by R obtains speed score and average plan distance similar to those obtained using the informed policies (except for CC in terms of speed score). The observed performance gaps are similar for the reduced case bases of the L2 libraries with 3000 and 2000 elements. This happens because the original case base contains 6000 cases that are grouped into clusters; consequently, for the reduced case bases with 3000 and 2000 elements, on average, the policies remove half and two thirds of the elements from each cluster of the original case bases, respectively. Thereby, even using the random policy, it is likely that, for each cluster of the original case base, the reduced case base still contains at least one case in the original library that can be reused effectively.

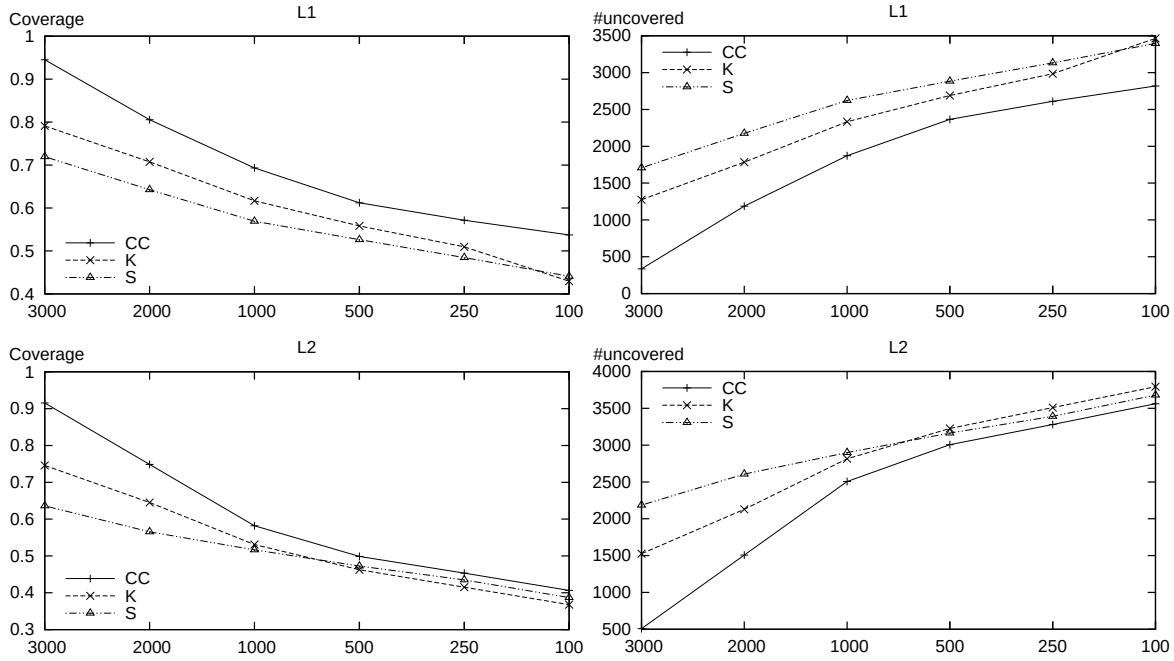

Figure 9: Coverage and number of uncovered cases of the plan libraries reduced from the original libraries in sets L1 and L2 through policies CC (Cardinality Coverage-guided), K (k-medoids clustering) and S (Spectral clustering). On the x-axis, there are the numbers of cases in the reduced plan libraries.

An approach to maintaining a case base that is alternative to the ones proposed in this paper is applying clustering analysis to a case base with the aim of building clusters of cases so that the case base obtained by selecting one prominent case from each cluster is smaller than the original case base and preserves the competence of the original case base (Smiti & Elouedi, 2013). In the rest of this section, we compare our maintenance policies with some known data clustering techniques applied to the plan library maintenance problem.

The intuitive goal of clustering is to divide a given set of data points into several groups such that points in the same group are similar to each other and points in different groups are dissimilar. In the context of CBP, data are planning cases. Cases can be represented in the form of a *similarity graph* defined as follows. Each vertex in this graph represents a case. Two cases are connected if the distance between the corresponding cases c_i and c_j is smaller than a certain threshold, and the edge is weighted by the value of the case distance between c_i and c_j. The problem of clustering can then be reformulated as the problem of finding a partition of the similarity graph such that the weight of edges between different groups is high, meaning that points in different clusters are dissimilar from each other, while the weight of edges within a group is low, meaning that points within the same cluster are similar to each other (von Luxburg, 2007).

The case base maintenance problem shares with standard clustering techniques the goal of selecting a representative set of elements. Such a goal is achieved by our approach

differently from standard clustering techniques. For instance, the k-medoids clustering technique selects the set of representative elements by minimizing the intra-cluster distances; such set is selected by the quality case base maintenance problem (without considering the reduction size) defining a reduction that minimizes the distances to the neighboring elements outside the reduction. Moreover, a number of standard clustering techniques, including k-medoids, require that the number of elements in the reduction is specified a priori, and each case belongs to only one cluster; while in our approach the number of elements in the reduction is not given, and a case of the original library can be a neighbor of more than one case of the reduction. This property allows our approach to retrieve more than one candidate case from the reduced case base when a case of the original plan library recurs or the CBP system encounters a similar case.

For our experimental analysis, we consider the well-known k-medoids and spectral algorithms for data clustering (von Luxburg, 2007; Sergios & Konstantinos, 2006), which in the following are denoted by K and S. With these techniques the number of clusters has to be specified a priori; in our experiments, such a number is equal to the size of the reduction. Indeed, as a side effect of the clustering, K and S return a prominent example for each defined cluster, and we can consider the set of these prominent examples as the reduction of the library. For this comparison, we considered policy CC, which exhibits the best performance among those proposed. Figure 9 compares the performance of CC w.r.t. clustering methods K and S.

Experimental result 5. *In terms of coverage and number of uncovered cases, for both library sets* L1 *and* L2 *and every size of the reduced libraries, overall the libraries maintained using policy* CC *are better than those derived using our implementations of clustering algorithms* K *and* S*.*

Appendix A gives the detailed results of this comparison showing, for every domain, the coverage and the number of uncovered cases of the maintained plan libraries. For the libraries in L1, when the size of the reduced plan library is equal to half of the original library, for almost all the considered domains, the coverage of the libraries maintained by either CC or QC is almost maximal (close to 1). The coverage of the reduced libraries for L2 is almost always lower than the coverage of the reduced libraries for L1, but it is still very high. For every domain and both library sets L1 and L2, when the size of the reduced plan library has the same order of magnitude as the size of the original plan library, the coverage obtained by maintaining the libraries using either CC or QC is usually *significantly* better than the coverage obtained by the other methods. The only method that sometimes obtains similar coverage is policy D. Finally, for almost all the considered domains, policies CC and QC again have the best coverage when the size of the reduced plan library is at least one order of magnitude lower than the size of the original plan library, although for these reductions the performance gap with our implementations of clustering methods K and S is lower.

In terms of the number of uncovered cases in the reduced plan libraries, the results in Appendix A show that the performance of the compared methods is similar to the performance we observed for the coverage: for every considered domain, the number of uncovered cases of the library maintained by policies CC and QC is almost always the lowest. It is worth noting that the policies we observed to perform best in terms of coverage

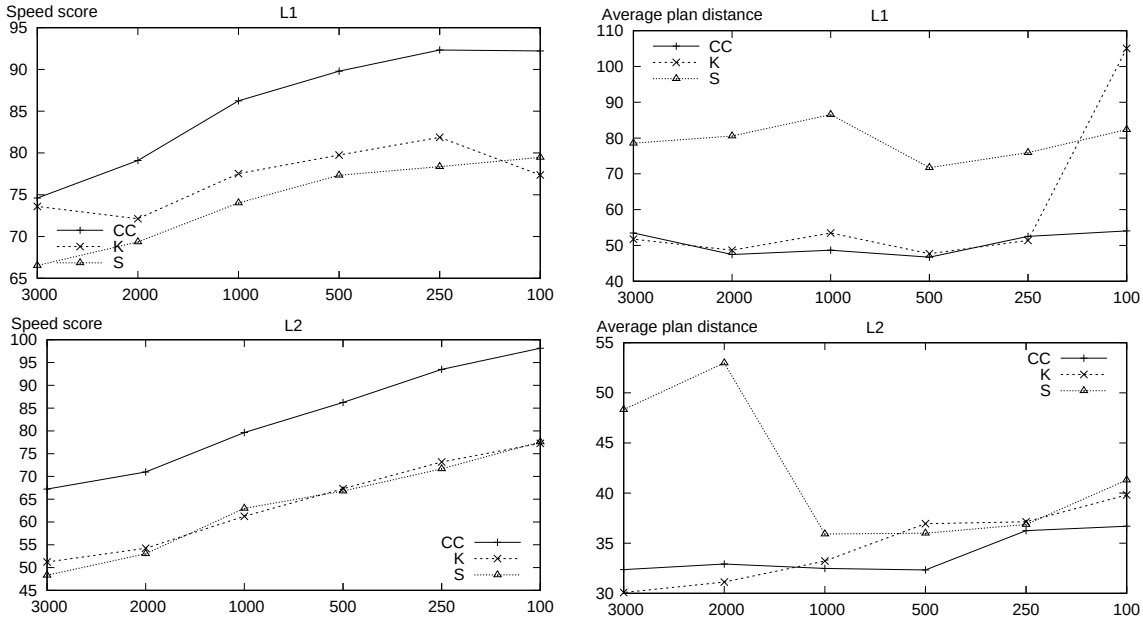

Figure 10: Total speed score and average plan distance of OAKplan using plan libraries reduced from the original libraries in sets L1 and L2 through policy CC (Cardinality Coverage-guided), clustering algorithms K (k-medoids) and S (Spectral). On the x-axis, there are the numbers of cases in the reduced plan libraries.

and number of uncovered cases are the same for the libraries with case plans generated using TLPlan (i.e., the libraries for all the considered domains except Elevators) and the libraries with case plans generated using FF.

Figure 10 gives the results of the comparison of policy CC and clustering algorithms K and S in terms of total speed score and average plan distance.

Experimental result 6. *In terms of total speed score, for both library sets* L1 *and* L2 *and every size of the reduced case bases,* OAKplan *using policy* CC *performs much better than using our implementations of clustering algorithms* K *and* S*. In terms of average plan distance,* OAKplan *using policy* CC *performs very often better.*

Specifically, in terms of average plan distance, for library set L1, OAKplan using policy CC performs almost always best. For the libraries in L2, OAKplan using CC performs always better than using S; moreover, it also performs much better than using K when the number of cases in the reduced library is 1000, 500, or 250, while it performs slightly worse when the number of cases in the reduced library is 3000 or 2000.

5.4 Online and Combined Policies

As remarked in Section 3, if every newly encountered planning case were added to the library, the size of the library could grow very quickly, significantly decreasing the performance of the whole system. On the other hand, the newly encountered planning cases that are substantially different from the cases stored in the library should not be discarded. Here,

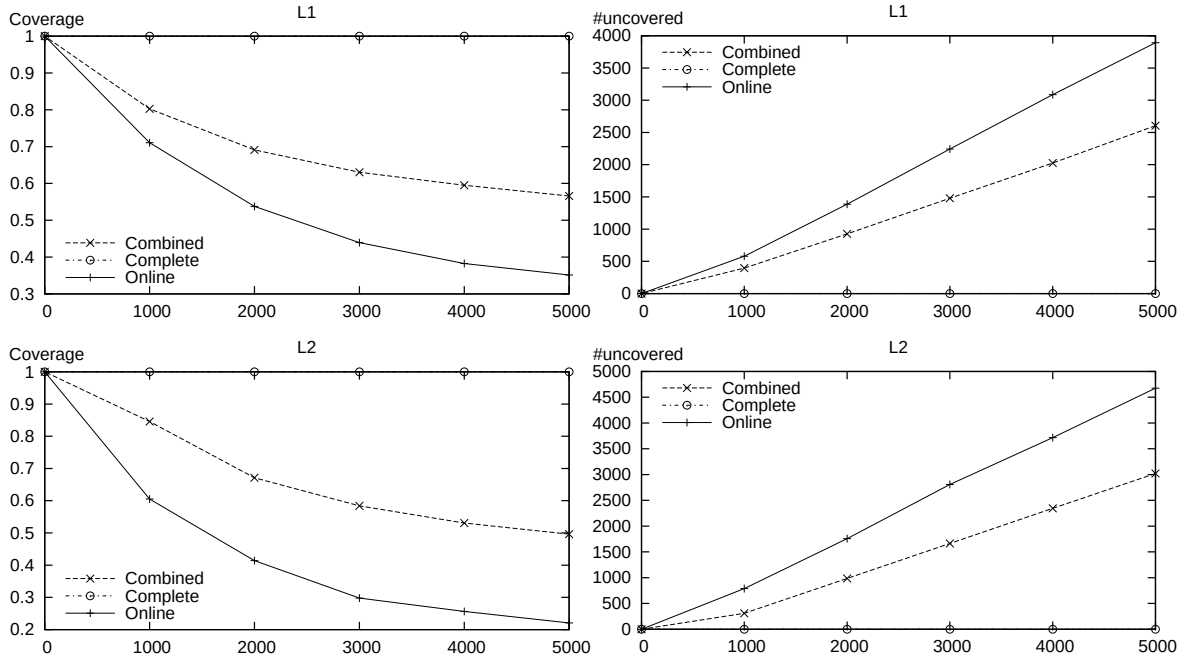

Figure 11: Coverage and number of uncovered cases of the plan libraries derived without maintenance, using online maintenance, and using combined maintenance. The initial library contains 1000 cases from library sets L1 and L2. The sets of newly encountered problems taken from L1 and L2 range from 0 to 5000. On the x-axis, there are the numbers of newly encountered problems.

we empirically evaluate the effectiveness of using our policies for deciding online whether the new case should be added to the case base or not, possibly removing some other case. These are the bounded online and the combined methods described in Sections 3.2 and 3.3. The limit condition of the combined policy is that 1000 new cases are encountered, i.e., every time 1000 new cases have been added to the case base, the combined policy invokes the offline policy. As a baseline of this evaluation, we consider the library obtained by storing each encountered case, which, in the following, is called the *complete library*.

Starting from a library containing 1000 cases, the libraries used in the evaluation are those derived by our policies when 1000, 2000, 3000, 4000, and 5000 new cases are encountered. These sets of cases are obtained by partitioning the sets L1 and L2 of 6000 elements into six sets, each of which consists of 1000 elements. It is worth noting that, while the complete library gradually grows with the newly encountered cases, the number of cases of the libraries reduced using the bounded online and the combined policies remains 1000.

Figure 11 shows the coverage and number of uncovered cases of the libraries maintained by the online policy and the combined policy w.r.t. the complete plan libraries. Obviously, the coverage of the complete libraries is always equal to 1, while the number of cases they do not cover is always 0. The coverage of the libraries obtained by the online and combined policies decreases logarithmically, while the number of uncovered cases grows linearly w.r.t.

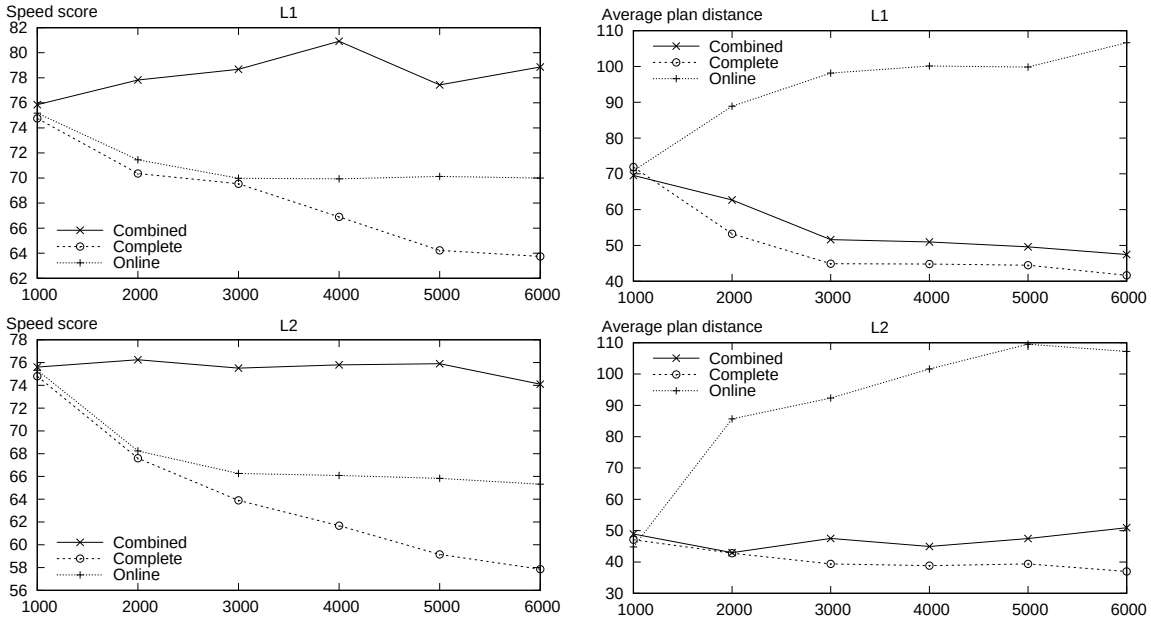

Figure 12: Total speed score and average plan distance of OAKplan with plan libraries derived without maintenance, using online maintenance, and using combined maintenance. The initial library contains 1000 cases from library sets L1 and L2. The sets of newly encountered problems taken from L1 and L2 range from 0 to 5000. On the x-axis, there are the number of newly encountered problems.

the number of newly encountered cases. Moreover, we can derive the following expected result.

Experimental result 7. *In terms of both coverage and number of uncovered cases, the libraries obtained by the combined policy are better than those generated by the online policy.*

The results in Figure 12 compare the effectiveness of exploiting the plan libraries used for the evaluation of Figure 11. While the coverage and the number of uncovered cases of the complete libraries are optimal, the usage of these libraries does not pay off. Indeed, we observed the following behavior.

Experimental result 8. *The speed score of OAKplan with the complete libraries is always worse than with the libraries obtained by the online policy, and it is always much worse than with the libraries obtained by the combined policy. Interestingly, OAKplan with the libraries obtained by the combined policy is also generally much faster than with the libraries obtained by the online policy.*

The reason why the online and combined policy give better performance than using the complete library is explained in Figure 13. The figure shows the average retrieval and search times of OAKplan with our policies and with the complete plan library. Since the complete library grows with the number of newly encountered cases, as expected, the time required to retrieve a case from this library also increases with the number of newly encountered

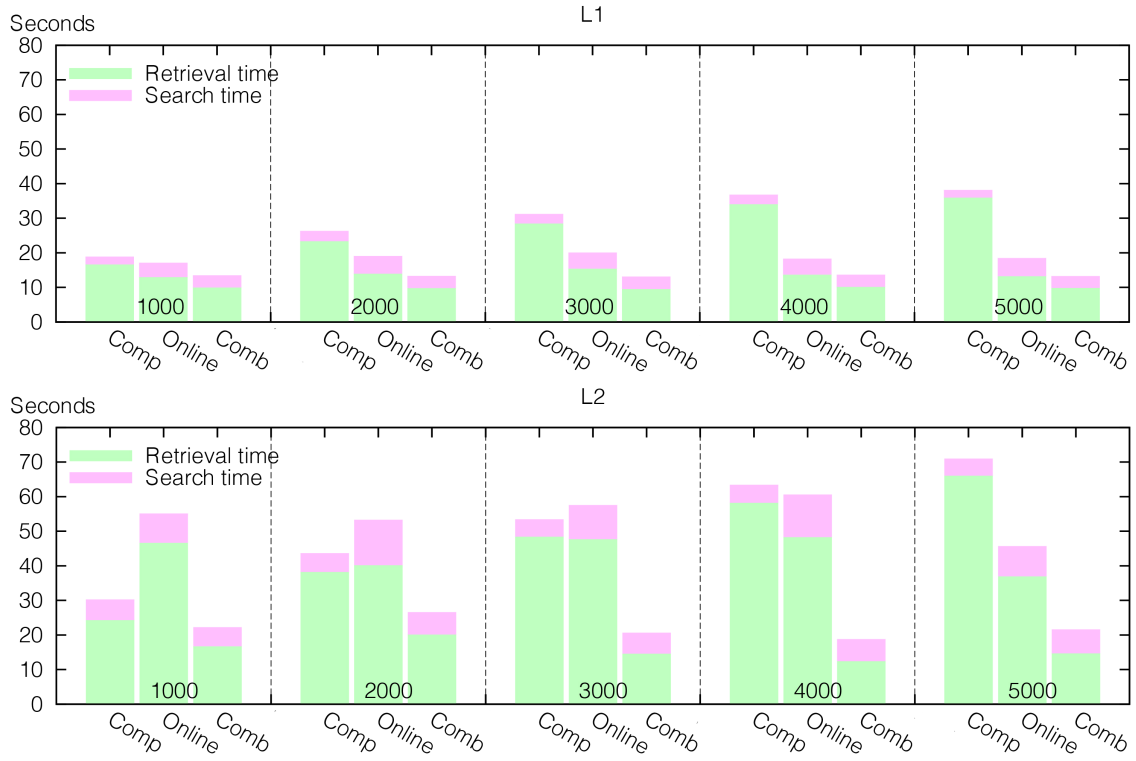

Figure 13: Average retrieval and search time of OAKplan with plan libraries derived without maintenance, using online maintenance, and using combined maintenance. The initial library contains 1000 cases from library sets L1 and L2. The sets of newly encountered problems taken from L1 and L2 range from 1000 to 5000. "Comp" and "Comb" abbreviate complete libraries (no maintenance) and combined maintenance, respectively.

cases. On the contrary, the time required to retrieve a case from the library reduced by the combined policy remains the same even though the number of newly encountered problems increases. This also happens with the libraries reduced by the online policy. The results in Figure 13 also show that to retrieve a case from the complete library is computationally more expensive than from the reduced libraries, and this makes the entire CBP system slower when the case base grows excessively. Moreover, as expected, the case retrieval from the library reduced using the combined policy is faster than the case retrieval from the library obtained using the online policy. This confirms that the library reduced using the combined policy makes CBP with OAKplan perform fastest.

As expected, in terms of average plan distance, the results in Figure 12 show that OAKplan using the complete libraries performs almost always the best, and in particular it performs much better than using the libraries obtained by the online maintenance. This result is somewhat expected since the coverage of the library obtained using the online policy significantly degrades with the number of new encountered problems. This implies that it is more frequent in the library maintained by the online policy to need to change

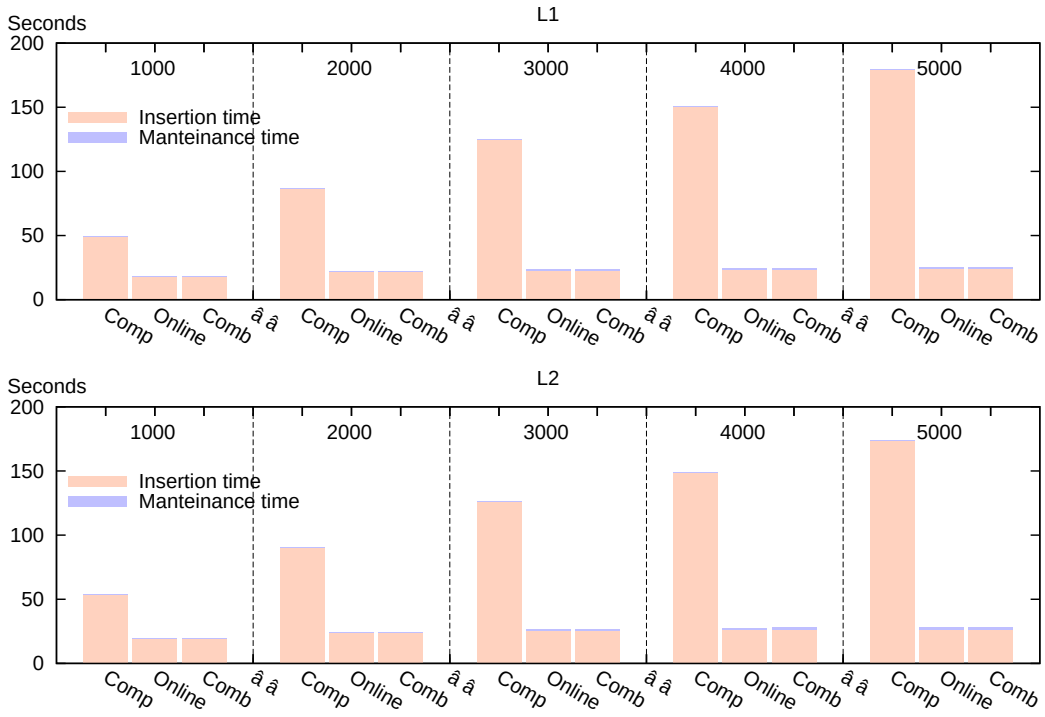

Figure 14: Average insertion and maintenance CPU times of OAKplan with plan libraries derived without maintenance, using online maintenance, and using combined maintenance. The initial library contains 1000 cases from library sets L1 and L2. The sets of newly encountered problems taken from L1 and L2 range from 1000 to 5000. "Comp" and "Comb" abbreviate complete libraries (no maintenance) and combined maintenance, respectively.

more the retrieved plans w.r.t. the plans retrieved from the complete library. However, it is interesting to observe that in terms of plan distances OAKplan using the libraries obtained by the combined maintenance performs only slightly worse than using the complete library. Overall, we can derive the following result.

Experimental result 9. *The combined maintenance shows the best trade-off between the computational cost of using the library and the average distance of the computed plans w.r.t. the retrieved cases.*

In our *online* setting, the tasks of deciding whether a case should be added to the library, deciding whether some other case should be removed, and adding the case need to be done quickly, as they are performed every time a new case is presented to the system. Therefore, we also evaluate the CPU time required to do these three related tasks. Figure 14 shows the average insertion time and maintenance time using the online and the combined policy w.r.t. using the complete library. The time required to insert a case into the library grows with the size of the library, because it includes the time required to compute the distance from the new case to the cases in the library. Therefore, the time required to insert a case into

the compete library grows with the number of newly encountered cases. On the contrary, as expected, for the library reduced by the online and combined policies such a time remains the same. Obviously, the complete library requires zero maintenance time. Interestingly, for both the online and combined policies, the maintenance time that is required every time a new case is presented to the system is very low; hence we derive the following result.

Experimental result 10. *The sum of the insertion and maintenance time for the online policy and the combined policy is much lower than the insertion time for the complete library.*

Finally, it is surprising to observe that the average maintenance time of the combined policy is pretty much the same as the maintenance time of the online policy although, differently from the online policy, the combined policy invokes the offline policy every time new 1000 cases are encountered. Given that, according to our experiments, the library reduced using the combined policy makes CBP with OAKplan more effective than using the online policy, and that the average insertion and maintenance time of the combined policy is substantially the same as the online policy, we can conclude as follows.

Experimental result 11. *The combined policy should be the preferred maintenance policy.*

5.5 Case-Based Planning vs Replanning

The existing literature has already shown that when the problems tend to recur and the planning domain is regular, adapting a previously generated plan is more effective than replanning from scratch (Serina, 2010). With the proposed policies, plan libraries keep only a (small) part of all the previously generated plans. In this section, we answer the question of whether adapting a previously generated plan that is also contained in the maintained library is still more effective than replanning from scratch.

The candidates we consider for this analysis as automated approaches that replan from scratch are planners LAMA (Richter & Westphal, 2010) and BFWS (Lipovetzky & Geffner, 2017). LAMA is very well-known and is often used as a baseline in the literature; BFWS is another planner in the state-of-the-art, as its variant (called LAPKT-DUAL-BFWS) was awarded at the 2018 international planning competition. For OAKplan, we consider policy CC because it is the best trade-off between speed and average plan distance.

For this analysis, we use OAKplan with a library containing 100, 500, and 1000 cases in the set of already encountered cases, which are the 6000 cases that we generated for our libraries. Table 3 shows the results, from which we can derive the following conclusion.

Experimental result 12. *Even with a library containing much fewer problems than the previously encountered ones, OAKplan almost always solves more problems than the generative approaches, and is often faster.*

The reason why LAMA and BFWS do not solve every problem in our benchmark is that sometimes they reach the time out (30 minutes).

We also observe that, in terms of plan quality, OAKplan and LAMA have comparable performances, and they perform better than BFWS (the total quality scores of OAKplan, LAMA and BFWS are 263.4, 274.2 and 202.2, respectively). In terms of average plan length, LAMA performs best (240.7 for LAMA against 286.2 and 285.4 for OAKplan and BFWS, respectively.) Most importantly, we observed that in our experiments the length of the

Domain	L	# solved problems					Speed score				
		BFWS	LAMA	OAKplan			BFWS	LAMA	OAKplan		
				100	500	1000			100	500	1000
DriverLog	L1	**25**	**25**	**25**	**25**	**25**	**24.1**	16.7	23.1	21.4	20.1
	L2	**25**	**25**	**25**	**25**	**25**	**25.0**	17.5	24.0	20.2	18.1
Elevators	L1	**25**	22	23	24	23	**11.2**	6.0	7.6	8.0	8.1
	L2	**25**	**25**	**25**	**25**	**25**	**23.0**	17.0	16.7	15.0	14.4
Logistics	L1	19	**25**	**25**	**25**	**25**	4.3	9.6	17.9	**18.0**	17.3
	L2	4	**25**	**25**	**25**	**25**	0.1	5.8	**13.4**	**13.4**	12.1
Rovers	L1	**25**	**25**	**25**	**25**	**25**	9.7	12.0	**15.3**	14.6	13.7
	L2	**25**	**25**	**25**	**25**	**25**	8.2	11.6	**15.5**	11.9	9.3
Satellite	L1	3	23	**25**	**25**	**25**	0.5	4.6	8.9	**9.4**	9.1
	L2	0	22	**25**	**25**	**25**	-	3.4	**8.9**	8.5	8.1
ZenoTravel	L1	**25**	**25**	**25**	**25**	**25**	15.1	10.1	17.3	**17.7**	17.0
	L2	**25**	**25**	**25**	**25**	**25**	**19.4**	10.1	17.8	17.8	17.7
Total	-	226	292	298	**299**	298	140.5	124.4	**186.4**	175.9	165.0

Table 3: Number of solved problems and speed score of BFWS, LAMA, and OAKplan using plan libraries reduced by policy CC from the original libraries in sets L1 and L2 to libraries with 100, 500, and 1000 cases.

plans computed by OAKPlan is quite similar to the length of the library plans (the median length difference is about 5%). This suggests that the quality of OAKPlan's plans strongly depends on the quality of the library plans. Therefore we conjecture that, if the elements of the plan library were all optimal plans, the plans computed by OAKPlan for the new encountered (similar) problems would be very often within a reasonable bound from the optimality.

Note that the analysis in Table 3 does not compare the approaches in terms of average plan distance because such a measure cannot be defined for the generative approaches. As pointed out before, computing plans that are not far away from previously encountered plans is important when humans have already validated the planning activities in these plans, and the effort required for such a validation may be considerable. Given the importance of this measure, we did an additional analysis using a special set of test problems derived from problems in the library without changing the object names, and comparing OAKplan with the specific generative planners that we used to derive the planning cases in the libraries (TLPlan and FF). With this setting, we can define the plan distance for TLPlan and FF as follows. It is defined as the distance between the plan π' computed for each test problem Π' and the library plan π for the library problem Π that we used to generate Π' from Π. It is worth noting that this comparison disfavors OAKplan, because plan π is likely the best plan to adapt for solving Π', and the correct identification of planning case $\langle \Pi, \pi \rangle$ is not an input for OAKplan, but it is rather the possible result of the retrieval process. Moreover, if the object names were renamed, the plan distance of TLPlan and FF would be the worst possible (i.e., the sum of the numbers of actions in π and π').

There is another issue to consider for this analysis. Indeed, differently from OAKplan, planner TLPlan is not fully automated: it is a domain-dependent planner that, in addition

Domain	L	Solved problems				Average plan distance			
		FF or TLPlan	OAKplan 100	OAKplan 500	OAKplan 1000	FF or TLPlan	OAKplan 100	OAKplan 500	OAKplan 1000
DriverLog	L1	24	**25**	**25**	**25**	53.2	42.5	39.2	**35.7**
	L2	23	**25**	**25**	**25**	32.8	**23.4**	28.4	24.1
Elevators	L1	**25**	23	24	23	329.3	**148.0**	166.0	160.9
	L2	**25**	**25**	**25**	**25**	115.5	**82.6**	92.6	92.6
Logistics	L1	**25**	**25**	**25**	**25**	**29.9**	59.7	60.5	60.5
	L2	**25**	**25**	**25**	**25**	**32.1**	42.0	41.9	60.5
Rovers	L1	**25**	**25**	**25**	**25**	51.2	35.6	**29.0**	30.1
	L2	**25**	**25**	**25**	**25**	33.7	31.4	**24.8**	25.1
Satellite	L1	**25**	**25**	**25**	**25**	44.2	45.9	**34.1**	34.4
	L2	**25**	**25**	**25**	**25**	16.9	26.8	24.4	**12.2**
ZenoTravel	L1	22	**25**	**25**	**25**	134.4	88.7	71.8	**70.4**
	L2	22	**25**	**25**	**25**	75.5	78.4	75.3	**63.0**
Total	-	293	298	299	298	948.8	705.0	688.0	**669.5**

Table 4: Number of solved problems and average plan distance of either TLPlan or FF, and OAKplan using plan libraries reduced by policy CC from the original libraries in sets L1 and L2 to libraries with 100, 500, and 1000 cases.

to the problem description, exploits further domain-specific knowledge. Such knowledge is extremely useful to speed up the search of the planner, but its definition was done by humans and required a considerable effort to formalize it.

Table 4 shows the comparison between OAKplan and planners TLPlan and FF. For all domains but Elevators, we used TLPlan for the generation of our libraries and hence for these domains we compared OAKplan with such a planner. For Elevators, we compared OAKplan with FF. Even if the setting of this experiment disfavors OAKplan, we can conclude as follows.

Experimental result 13. *OAKplan performs much better than the generative approaches in terms of plan distance.*

We also observed that, as expected, TLPlan is usually much faster than OAKplan. However, this is not because TLPlan replans from scratch, instead of from a library plan, but because it exploits the additional domain specific knowledge for controlling the search. Such extra knowledge, which considerably speeds up the search of TLPlan is available neither for fully automated generative planners nor for OAKplan.

6. Related Work

While the topic of case base maintenance has been of great interest in the CBR community for the last two decades, in CBP it has received little attention. Therefore, the related work mainly falls in the field of CBR. Moreover, in CBR most of the proposed systems for case base maintenance have been formulated as classification problems (where a case usually can

or cannot be adapted to solve a new problem), which makes applying these techniques in the context of CBP complicated.

Leake and Wilson (1998) defined the problem of case base maintenance as "an implementation of a policy to revise the case base to facilitate future reasoning for a particular set of performance objectives". Depending on the evaluation criteria, they distinguish two types of case base maintenance techniques — the *quantitative criteria* (e.g., time) used for performance-driven policies, and the *qualitative criteria* (e.g., coverage) used for competence-driven policies.

The quantitative criteria are usually easier to compute; among these performance-driven policies there are the very simple random deletion policy (Markovitch, Scott, & Porter, 1993) and a policy driven by a case utility metric (Minton, 1990), where the utility of a case is increased by its frequent reuse and decreased by costs associated with its maintenance and matching.

As for the qualitative criteria, in the context of CBR, Reinartz, Iglezakis, and Roth-Berghofer (2000) suggest to use quality measures reflecting user requirements, such as correctness, solution's stability, and diversity, and combine them into a measure that is easy to compute by a domain-independent computation. However, the most used qualitative criterion in CBR is based on the notion of competence introduced by Smyth (1998). Intuitively, elements are removed from the case base in reverse order with respect to their importance, where the importance of a case is determined by the case's "coverage" and "reachability". These two notions capture how many problems the case solves and how many cases solve the case problem. Note that, however, differently from our approach to CBP, Smyth considers systems without an underlying generic (generative) solver. With such a solver the CBP system can solve any problem independently of the quality of the case base, and the notion of competence needs to be revised.

The notion of competence was also used to define the "footprint deletion" and "footprint-utility deletion" policies (Smyth & Keane, 1998). Another extension is the RC-CNN algorithm (Smyth & McKenna, 1999), which compresses the case base using the compressed-nearest-neighbor algorithm (Hart, 1968) and the ordering derived by the relative coverages of the cases. Furthermore, Leake and Wilson (2000) suggested replacing the relative coverage with the relative performance of a case. Zhu and Yang (1998) however claim that the competence-driven policies of Smyth and his collaborators do not ensure competence preservation. They provide a case addition policy which mimics a greedy algorithm for set covering, always adding the case that has the biggest coverage (resp. high frequency of reuse and large neighborhood of cases) until the whole original case base is covered or the size limit is reached. The policies we propose here differ from the approach of Zhu and Yang (1998) mainly in the condition guiding the selection of the cases to keep in the case base — Zhu and Yang evaluate the utility of each case on the base of the frequency of its reuse in comparison with the frequency with which its neighbors are reused.[5] Moreover, their policy does not consider the quality with which the original case base is covered, whereas

5. Intuitively, Zhu and Yang (1998) define the neighborhood of a case c as a set of cases in the case base that are most similar to c, according to some similarity measure. They evaluate their policies through an experiment in CBP; in this experiment, the neighborhood of a case c contains only cases with plans that are plan-suffix extensions of the plan in c. The concept of neighborhood is interpreted very differently in our work.

our quality-coverage policy does. In a sense, we generalise the work of Zhu and Yang and adapt it to the context of planning.

Yang and Wu (2000) propose a competence-driven policy that keeps all the cases, but identifies similar problems and gathers them into smaller, more focused case bases. The retrieval is then performed in a two-level hierarchy and is guided by a decision tree forest. Shiu, Yeung, Sun, and Wang (2001) instead discard the majority of the case base after using the redundant cases to learn new adaptation rules based on the nature of their redundancy.

Muñoz-Avila (2001) studied the case retention problem in CBP with the aim of filtering redundant cases, which is closely related to the problem studied here. However, the policy proposed by Muñoz-Avila is guided by the case reuse effort, called *the benefit of the retrieval*, required by a specific "derivational" case-based planner to solve the problem. Intuitively, a case c is kept only if there is no other case in the case base that could be easily adapted by the planner to solve the problem represented by c. In our approach, the decision to keep a case is independent from the adaptation cost of the other cases. Moreover, the policy proposed by Muñoz-Avila can decide only about problems solved by the adopted derivational case-based planner; while the policies studied in this paper are independent from the planner used to generate the solutions of the cases.

In our own previous work, we investigated the problem of case base maintenance and gave an introduction of the policies presented in this paper (Gerevini, Roubíčková, Saetti, & Serina, 2013b, 2013a). In this paper, we have given a more comprehensive, detailed and formal treatment of the problem of case base maintenance. The work in this paper includes substantial new technical material and considerably extends our previous work. In particular, the new material includes:

- a formal definition and treatment of the problem of maintaining a plan library, optimizing two different criteria: minimizing the size, and maximizing the quality of the reduced plan library;
- theorems and proofs about the worst-case complexity (NP-hardness) of the investigated maintenance problem;
- a theoretical analysis and comparison of the worst-case complexity of our maintenance policies for computing approximations of the exact reduction solving the maintenance problem;
- and a large experimental study comparing our maintenance policies.

The experimental analysis is completely new because previous work compares reduced case bases that have *different* sizes. Evaluating the quality and usefulness of reduced case bases of different sizes is questionable since they contain different amounts of knowledge. In this paper, the (offline) maintenance policies are used with the aim of deriving reduced case bases of a predetermined size, so that the experimental comparisons regard reduced case bases that have the *same* size. Moreover, the experimental analysis in this paper is much larger than the analyses in our previous work since here we use many more domains and problems, the experimental analysis has many more goals, and it is more elaborated (e.g., it uses two sets of cases, called L1 and L2, with the purpose of evaluating our approach when problems tend to recur with different chances).

Finally, we observe that the issue of deciding a set of representative instances to store is in common with other research areas, such as instance-based learning (Aha, Kibler, & Albert, 1991). The goal of instance-based learning is selecting from the training set a number of exemplars with better generalization properties; these exemplars can be successively used as a new training set for supervised ML algorithms. One of the most straightforward instance-based learning algorithms is the nearest neighbor algorithm (Hart, 1968). Clustering techniques, including those used in our experimental analysis, can be used as a reduction technique for instance-based learning.

7. Conclusions

The retention of cases is an important phase of CBP. Retaining every encountered case in the library leads to uncontrolled growth of the library, which increases the retrieval effort till the point where the CBP approach may become ineffective. We have defined the problem of maintaining the cases forming a plan library using two different criteria: minimizing the size of the reduced case base, and maximizing its quality. Under the considered criteria, the maintenance problem is NP-hard, and hence computing an exact optimized reduction of the library can incur a high computational cost. Therefore, we have defined and investigated new policies for computing approximate solutions to the addressed problem.

One of the challenges when reducing a big library is the computational cost of finding a good-quality reduction, and some techniques are more suitable for offline use as they are not efficient enough for online library maintenance. The random policy, which is used in general CBR, is very fast to compute because it does not optimize the case base in any way. We have introduced two informed policies, the distance-guided policy and the coverage-guided policy, which attempt to generate reduced case bases by optimizing size and/or quality. Moreover, we developed an approach complementary to the offline policy, called online maintenance. This new approach attempts to contain the growth of the library over the lifetime of the planner by deciding, every time the CBP system encounters a new problem, whether to discard or to include the problem and its solution in the library.

The online maintenance is able to keep the library size limited to a given bound, but its quality can degrade over time. To overcome the problem with the quality of the reduced case base in the online approach, we have proposed a combined policy that has the benefits of both the online and offline approaches. It stores each new case in the library considering only certain cases as active; the case base grows slowly and the retrieval time remains low, with the quality of the case base addressed by a periodical re-evaluation of which cases are considered active.

An experimental analysis shows that the informed offline policies can be much more effective than the random policy, in terms of quality of the reduced case base and performance of a case base planner using it. We have also compared these policies with two known methods from clustering analysis that we applied to reduce the case base. The results of the experimental analysis show that our approach for reducing a plan library can perform much better than using existing clustering methods. Of course, this may largely depend on some design choices that we made for implementing the clustering techniques in our experimental analysis; we cannot rule out the possibility that with different choices the relative performance may be different. Finally, we have answered the question of whether adapting

a previously generated plan that is contained in the maintained library is more effective than replanning from scratch. Our experiments show that OAKplan using the maintained libraries is faster than two state-of-the-art generative planners, and computes plans closer to the plans retrieved from the library w.r.t. the two generative planners that we used for generating the library plans.

There are several research directions to extend our work. We intend to investigate alternative methods for efficiently computing good approximations of the optimally reduced case base (in terms of size and quality). Moreover, it would be useful to study other aspects related to online maintenance, such as different criteria that can guide the policy. In this work, the diversity of a case is only a proxy for the costs of reusing it. However, if the new problem is solved locally by the CBP system itself, we have access to additional information about the real effort that is needed to solve the problem. Hence, the policy could take the observed costs into account, and consider to include the cases that are hard to solve into the case base. Finally, the benefit of a case can be assessed more accurately if the system keeps track of which elements of the case base are being retrieved and lead to successful reuse since the frequency and time of the successful retrievals provide precise information about the real use of the cases in the system.

Acknowledgments

This work was supported by the EU H2020 project AIPlan4EU (GA n. 101016442), EU ICT-48 2020 project TAILOR (No. 952215), and MUR PRIN project RIPER (No. 20203FFYLK) The *IBM Power Systems Academic Initiative* substantially contributed to the experimental analysis.

Appendix A. Detailed Experimental Results

The following two tables show, for each domain and reduced library, the coverage (first table) and the number of uncovered cases (second table) computed using a case-distance threshold equal to the average minimum distance of the cases in the original case base. Bold numbers indicate the best performance.

CB	**Coverage of the reduced case bases**											
Size	Driverlog		Elevators		Logistics		Rovers		Satellite		Zenotravel	
Policy	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2
3000												
R	0.687	0.717	0.775	0.727	0.727	0.743	0.716	0.719	0.891	0.729	0.726	0.725
K	0.774	0.743	0.786	0.744	0.758	0.749	0.739	0.748	0.931	0.742	0.759	0.749
S	0.700	0.599	0.619	0.607	0.726	0.714	0.721	0.654	0.888	0.675	0.662	0.567
D	0.718	0.835	0.584	0.804	0.959	0.987	**0.916**	0.869	0.864	0.924	0.959	0.858
CC	**0.951**	0.899	**0.977**	**0.893**	**0.972**	**0.988**	0.912	**0.884**	**1.0**	**0.942**	**0.962**	**0.887**
QC	**0.951**	**0.900**	0.976	0.891	**0.972**	0.987	0.912	**0.884**	**1.0**	**0.942**	**0.962**	0.886
2000												
R	0.534	0.600	0.680	0.621	0.606	0.642	0.619	0.620	0.841	0.620	0.597	0.612
K	0.678	0.638	0.680	0.634	0.666	0.658	0.651	0.645	0.895	0.653	0.674	0.642
S	0.617	0.533	0.549	0.529	0.629	0.715	0.636	0.548	0.848	0.614	0.576	0.454
D	0.557	0.669	0.579	0.638	0.793	0.820	0.762	0.702	0.837	0.757	0.793	0.691
CC	**0.786**	0.733	**0.830**	**0.726**	**0.807**	**0.821**	**0.768**	0.717	**1.0**	0.775	**0.796**	**0.720**
QC	**0.786**	**0.734**	0.829	0.725	**0.807**	**0.821**	**0.768**	**0.718**	**1.0**	**0.776**	**0.796**	**0.720**
1000												
R	0.338	0.460	0.538	0.478	0.446	0.529	0.474	0.479	0.754	0.486	0.436	0.473
K	0.569	0.514	0.537	0.509	0.577	0.578	0.571	0.513	0.864	0.559	0.583	0.512
S	0.501	0.506	0.470	0.455	0.552	0.638	0.548	0.506	0.839	0.547	0.505	0.448
D	0.357	0.502	0.568	0.475	0.626	0.653	0.608	0.535	0.831	0.591	0.627	0.524
CC	**0.622**	0.566	**0.666**	**0.559**	**0.643**	**0.654**	**0.623**	**0.551**	**0.976**	0.609	**0.630**	**0.554**
QC	**0.622**	**0.567**	0.664	0.558	**0.643**	**0.654**	**0.623**	**0.551**	**0.976**	**0.610**	**0.630**	0.553
500												
R	0.221	0.364	0.441	0.378	0.293	0.418	0.352	0.389	0.604	0.415	0.307	0.361
K	0.512	0.431	0.405	0.428	0.528	0.536	0.528	0.446	0.842	0.504	0.534	0.429
S	0.445	0.454	0.393	0.433	0.507	0.551	0.500	0.464	0.817	0.516	0.496	0.418
D	0.238	0.406	0.559	0.389	0.543	0.570	0.531	0.451	0.825	0.507	0.544	0.445
CC	**0.540**	0.483	**0.582**	**0.476**	**0.561**	**0.571**	0.550	0.467	0.891	**0.526**	**0.547**	**0.470**
QC	**0.540**	**0.484**	0.581	0.475	**0.561**	**0.571**	**0.551**	**0.468**	**0.893**	**0.526**	**0.547**	0.469
250												
R	0.106	0.283	0.300	0.306	0.217	0.346	0.277	0.289	0.460	0.291	0.187	0.269
K	0.466	0.367	0.261	0.379	0.502	0.516	0.499	0.392	0.829	0.467	0.500	0.370
S	0.368	0.402	0.292	0.398	0.486	0.509	0.475	0.421	0.812	**0.488**	0.475	0.392
D	0.153	0.338	0.507	0.317	0.505	0.528	0.492	0.399	0.819	0.468	**0.508**	0.392
CC	**0.498**	0.426	**0.541**	**0.427**	**0.519**	**0.529**	0.515	**0.426**	0.839	0.484	0.505	**0.429**
QC	**0.498**	**0.427**	0.539	**0.427**	**0.519**	**0.529**	**0.516**	**0.426**	**0.851**	0.485	0.505	0.428
100												
R	0.052	0.200	0.150	0.168	0.067	0.245	0.110	0.186	0.234	0.203	0.092	0.106
K	0.369	0.306	0.148	0.304	0.451	0.500	0.467	0.340	0.718	0.444	0.425	0.312
S	0.275	0.335	0.174	0.337	0.485	0.494	0.486	0.374	**0.823**	0.443	0.404	0.338
D	0.074	0.291	0.352	0.241	0.360	0.503	0.282	0.317	0.562	0.442	0.430	0.324
CC	**0.457**	0.359	**0.504**	0.359	**0.486**	**0.504**	0.487	**0.381**	0.543	0.458	**0.467**	**0.377**
QC	**0.457**	**0.361**	0.503	**0.362**	**0.486**	**0.504**	**0.488**	**0.381**	**0.823**	**0.459**	**0.467**	0.375

CB	**Number of uncovered cases in the reduced case bases**											
Size	Driverlog		Elevators		Logistics		Rovers		Satellite		Zenotravel	
Policy	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2
3000												
R	1877	1700	1349	1636	1639	1545	1844	1686	656	1626	1648	1652
K	1354	1543	1281	1538	1451	1508	1694	1510	414	1550	1449	1504
S	1800	2408	2286	2356	1645	1716	1806	2077	670	1951	2032	2600
D	1692	988	2495	1173	243	79	**545**	789	817	456	244	854
CC	**295**	606	**140**	**644**	**165**	**74**	568	697	**1**	350	**226**	**678**
QC	**295**	**598**	145	652	**165**	76	568	**694**	**1**	**346**	**226**	683
2000												
R	2797	2403	1918	2273	2362	2145	2469	2280	955	2281	2424	2330
K	1934	2170	1922	2195	2003	2049	2260	2132	627	2080	1961	2145
S	2299	2799	2705	2826	2223	1710	2359	2711	915	2315	2550	3278
D	2660	1988	2525	2173	1243	1079	1543	1789	977	1456	1244	1854
CC	**1284**	1605	**1020**	**1642**	**1157**	**1074**	1507	1697	**3**	1349	**1226**	**1678**
QC	**1284**	**1598**	1026	1650	**1157**	1076	**1505**	**1694**	**3**	**1344**	**1226**	1683
1000												
R	3974	3241	2774	3129	3325	2823	3409	3124	1476	3082	3388	3160
K	2585	2917	2776	2944	2536	2534	2783	2922	815	2648	2509	2926
S	2993	2964	3178	3268	2688	2173	2928	2962	969	2719	2977	3311
D	3860	2988	2591	3147	2243	2079	2543	2789	1016	2456	2244	2854
CC	**2270**	2603	**2006**	**2643**	**2143**	**2073**	2447	2696	145	2347	2226	**2678**
QC	**2270**	**2597**	2014	2651	**2143**	2075	**2444**	**2693**	**143**	**2342**	**2225**	2683
500												
R	4677	3818	3353	3735	4238	3490	4201	3664	2376	3509	4169	3836
K	2926	3414	3569	3431	2829	2784	3059	3322	949	2977	2799	3426
S	3333	3274	3641	3403	2960	2696	3243	3218	1096	2903	3032	3494
D	4573	3564	2646	3668	2743	2579	3042	3297	1050	2956	2739	3331
CC	**2761**	3102	**2506**	**3143**	**2636**	**2573**	2917	3196	651	2846	2726	**3177**
QC	**2761**	**3096**	2514	3151	**2636**	2575	**2912**	**3193**	**643**	**2842**	**2725**	3183
250												
R	5366	4300	4198	4164	4698	3922	4688	4266	3243	4251	4886	4388
K	3206	3799	4432	3727	2989	2906	3247	3651	1029	3200	3006	3778
S	3790	3586	4245	3614	3080	2947	3401	3473	1128	**3069**	3155	3651
D	5079	3972	2958	4099	2970	2829	3294	3605	1087	3191	**2957**	3650
CC	**3009**	3446	**2756**	**3440**	**2885**	**2823**	3147	3446	969	3096	2976	**3427**
QC	**3009**	**3436**	2764	**3440**	**2885**	2825	**3140**	**3443**	**893**	3092	2975	3433
100												
R	5687	4801	5098	4995	5596	4529	5768	4881	4599	4781	5459	5362
K	3786	4161	5113	4173	3290	2999	3453	3958	1690	3336	3457	4130
S	4350	3992	4953	3981	3090	3034	3333	3757	**1064**	3341	3583	3970
D	5556	4256	3888	4551	3838	2979	4656	4096	2625	3346	3425	4059
CC	**3256**	3847	**2978**	3843	**3082**	**2973**	3325	3714	2743	3250	**3207**	**3740**
QC	**3256**	**3836**	2979	**3827**	**3082**	2975	**3317**	**3712**	**1064**	**3247**	**3207**	3752

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communication*, *7*(1), 39–59.

Aha, W., Kibler, D., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37–66.

Bacchus, F., & Kabanza, F. (2000). Using temporal logic to express search control knowledge for planning. *Artificial Intelligence*, *116(1-2)*, 123–191.

Bäckström, C., & Nebel, B. (1995). Complexity results for SAS+ planning. *Computational Intelligence*, *11*, 625–655.

Borrajo, D., Roubíčková, A., & Serina, I. (2014). Progress in case-based planning. *ACM Computing Surveys*, *47*(2), 35:1–35:39.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, *69*, 165–204.

Cox, M. T., Muñoz-Avila, H., & Bergmann, R. (2005). Case-based planning. *Knowledge Engineering Review*, *20*(3), 283–287.

Domshlak, C., Hoffmann, J., & Katz, M. (2015). Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, *221*, 73–114.

Fikes, R., & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*(3/4), 189–208.

Fox, M., Gerevini, A., Long, D., & Serina, I. (2006). Plan stability: Replanning versus plan repair. In *Proceedings of International Conference on AI Planning and Scheduling (ICAPS)*. AAAI Press.

Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, *20*, 61–124.

Garey, M. R., & Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gerevini, A., Saetti, A., & Serina, I. (2003). Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research*, *20*, pp. 239–290.

Gerevini, A., & Serina, I. (2010). Efficient plan adaptation through replanning windows and heuristic goals. *Fundamenta Informaticae*, *102*(3-4), 287–323.

Gerevini, A., Roubíčková, A., Saetti, A., & Serina, I. (2013a). Offline and online plan library maintenance in case-based planning. In *AI*IA 2013: Advances in Artificial Intelligence - 13th International Conference of the Italian Association for Artificial Intelligence*, Vol. 8249 of *Lecture Notes in Computer Science*, pp. 239–250.

Gerevini, A., Roubíčková, A., Saetti, A., & Serina, I. (2013b). On the plan-library maintenance problem in a case-based planner. In *Case-Based Reasoning Research and Development - 21st International Conference*, Vol. 7969 of *Lecture Notes in Computer Science*, pp. 119–133.

Gerevini, A., Saetti, A., & Serina, I. (2006). An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research*, *25*, 187–231.

Gerevini, A., Saetti, A., & Serina, I. (2008). An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence*, *172*(8-9), 899–944.

Gerevini, A., Saetti, A., & Serina, I. (2011). An empirical analysis of some heuristic features for planning through local search and action graphs. *Fundamenta Informaticae*, *107*(2-3), 167–197.

Gerevini, A., Saetti, A., & Serina, I. (2012). Case-based planning for problems with real-valued fluents: Kernel functions for effective plan retrieval.. Vol. 242 of *Frontiers in Artificial Intelligence and Applications*, pp. 348–353. IOS Press.

Gerevini, A., & Serina, I. (2003). Planning as propositional CSP: from walksat to local search techniques for action graphs. *Constraints An Int. J.*, *8*(4), 389–413.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). PDDL - the planning domain definition language. Technical report CVC TR98-003/DCS TR-1165, Yale Center for Computational Vision and Control, available at `http://cs-www.cs.yale.edu/homes/dvm/`.

Ghallab, M., Nau, D., & Traverso, P. (2016). *Automated Planning and Acting*. Cambridge University Press.

Ghallab, M., Nau, D. S., & Traverso, P. (2004). *Automated planning - theory and practice*. Elsevier.

Hammond, K. J. (1989). *Case-based planning: viewing planning as a memory task*. Academic Press.

Hammond, K. J. (1990). Case-based planning: A framework for planning from experience. *Cognitive Science*, *14*(3), 385–443.

Hanks, S., & Weld, D. (1995). A domain-independent algorithm for plan adaptation.. *Journal of Artificial Intelligence Research*, *2*, 319–360.

Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, *14*, 515–516.

Helmert, M. (2006). The Fast Downward planning system. *Journal of Artificial Intelligence Research*, *26*(1), 191–246.

Helmert, M., Do, M., & Refanidis, I. (2008). Sixth International Planning Competition IPC6: Deterministic part. In *http://ipc.informatik.uni-freiburg.de/*.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *Proceedings of the 19th International Conference on Planning & Scheduling (ICAPS)*. AAAI Press.

Hoffmann, J. (2001). FF: The Fast-Forward planning system. *AI magazine*, *22*(3), 57–62.

Hoffmann, J. (2003). The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, *20*, 291–341.

Jiménez, S., de la Rosa, T., Fernández, S., Fernández, F., & Borrajo, D. (2012). A review of machine learning for automated planning. *Knowledge Engineering Review*, *27*(4), 433–467.

Leake, D. B., Kinley, A., & Wilson, D. C. (1997). Case-based similarity assessment: Estimating adaptability from experience. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pp. 674–679. AAAI Press.

Leake, D. B., & Wilson, D. C. (1998). Categorizing case-base maintenance: Dimensions and directions. In *Proceedings of the 4th European Workshop on Case-Based Reasoning*. Springer.

Leake, D. B., & Wilson, D. C. (2000). Remembering why to remember: Performance-guided case-base maintenance. In *Proceedings of the 5th European Workshop on Case Base Reasoning*.

Liberatore, P. (2005). On the complexity of case-based planning. *Journal of Experimental & Theoretical Artificial Intelligence*, *17*(3), 283–295.

Lipovetzky, N., & Geffner, H. (2017). Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*. AAAI Press.

Long, D., & Fox, M. (2003). The 3rd International Planning Competition: Results and analysis. *Journal of Artificial Intelligence Research*, *20*, 1–59.

Markovitch, S., Scott, P. D., & Porter, B. (1993). Information filtering: Selection mechanisms in learning systems. In *Proceedings of the 10th International Conference on Machine Learning*. Morgan Kaufmann.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*(2-3), 363–391.

Muñoz-Avila, H. (2001). Case-base maintenance by integrating case-index revision and case-retention policies in a derivational replay framework. *Computational Intelligence*, *17*(2), 280–294.

Nguyen, T. A., Do, M. B., Gerevini, A., Serina, I., Srivastava, B., & Kambhampati, S. (2012). Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, *190*, 1–31.

Ontañón, S., Mishra, K., Sugandh, N., & Ram, A. (2010). On-line case-based planning. *Computational Intelligence*, *26*, 84–119.

Raymond, J. W., Gardiner, E. J., & Willett, P. (2002). Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, *45*(6), 631–644.

Reinartz, T., Iglezakis, I., & Roth-Berghofer, T. (2000). On quality measures for case base maintenance. In *Proceedings of the 5th European Workshop on Case-Based Reasoning*.

Richter, S., & Westphal, M. (2010). The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, *39*, 127–177.

Scala, E., Micalizio, R., & Torasso, P. (2015). Robust plan execution via reconfiguration and replanning. *AI Communication*, *28*(3), 479–509.

Scala, E., & Torasso, P. (2015). Deordering and numeric macro actions for plan repair. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*. AAAI Press.

Sergios, T., & Konstantinos, K. (2006). *Pattern Recognition, Third Edition*. Academic Press, Orlando, FL, USA.

Serina, I. (2010). Kernel functions for case-based planning. *Artificial Intelligence*, *174*(16-17), 1369–1406.

Shiu, S. C. K., Yeung, D. S., Sun, C. H., & Wang, X. (2001). Transferring case knowledge to adaptation knowledge: An approach for case-base maintenance. *Computational Intelligence*, *17*, 295–314.

Smiti, A., & Elouedi, Z. (2013). Modeling competence for case based reasoning systems using clustering. In *Proceedings of the 26th International Florida Artificial Intelligence Research Society Conference (FLAIRS-13)*.

Smyth, B. (1998). Case-base maintenance. In *Proceedings of the 11th International Conference on Industrial and Engineering Applications of AI and Expert Systems*. Springer.

Smyth, B., & Keane, M. T. (1998). Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, *102*(2), 249–293.

Smyth, B., & McKenna, E. (1999). Footprint-based retrieval. In *Proceedings of the 3rd International Conference on Case Base Reasoning (ICCBR-99)*, Vol. 1650 of *Lecture Notes in Computer Science*, pp. 343–357. Springer.

Smyth, B., & McKenna, E. (2001). Competence models and the maintenance problem.. *Computational Intelligence*, *17*(2), 235–249.

Spalazzi, L. (2001). A survey on case-based planning. *Artificial Intelligence Review*, *16*(1), 3–36.

Tonidandel, F., & Rillo, M. (2002). The FAR-OFF system: A heuristic search case-based planning. In Ghallab, M., Hertzberg, J., & Traverso, P. (Eds.), *AIPS*, pp. 302–311. AAAI.

Vallati, M., Chrpa, L., Grzes, M., McCluskey, T. L., Roberts, M., & Sanner, S. (2015). The 2014 international planning competition: Progress and trends. *AI Magazine*, *36*(3), 90–98.

Veloso, M. M. (1994). *Planning and Learning by Analogical Reasoning*, Vol. 886 of *Lecture Notes in Computer Science*. Springer.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395–416.

Yang, Q., & Wu, J. (2000). Keep it simple: A case-base maintenance policy based on clustering and information theory. In *Proceedings of the 13th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*. Springer.

Zhu, J., & Yang, Q. (1998). Remembering to add: Competence-preserving case-addition policies for case-base maintenance. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-98)*. Morgan Kaufmann.

Zimmerman, T., & Kambhampati, S. (2003). Learning-assisted automated planning: Looking back, taking stock, going forward. *AI Magazine*, *24*(2), 73–96.