CASE-BASE MAINTENANCE BEYOND CASE DELETION

Brian Schack

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Department of Computer Science,

Indiana University

November 28, 2023

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

David Leake, PhD

David Crandall, PhD

Ken Shan, PhD

Curt Bonk, PhD

November 28, 2023

Acknowledgments

I acknowledge the contributions of my advisor, Dr. David Leake; my minor advisor, Dr. Curt Bonk; and my other committee members, Dr. David Crandall and Dr. Ken Shan. I also acknowledge the contributions of my mentors, Dr. Michael Floyd and Dr. David Aha, at the Center for Applied Research in Artificial Intelligence at the Naval Research Laboratory. I also acknowledge the contributions of my colleagues: Vahid Jalali, Eriya Terada, Larry Gates, Caleb Kisby, Rin Metcalf, Xiaomeng Ye, Zach Wilkerson, and Kaitlynne Wilkerson. Finally, I acknowledge the support of my parents, Drs. Mark and Edna Schack.

This dissertation follows a LaTeX template from the Department of Mathematics.[1] "Chapter 3: Subdividing Cases for Deletion" (on page 26) is adapted from Leake and Schack (2015) and Schack and Summers (2017). "Chapter 4: Prioritizing Deletion by Recoverability via Adaptation Knowledge" (on page 44) is adapted from Leake and Schack (2016) and Schack (2016). "Chapter 5: Data Augmentation to Expand Choices for Deletion" (on page 61) is adapted from Leake and Schack (2018) and Schack (2019). "Chapter 6: Predictive Case Discovery for Problem-Distribution Drift" (on page 82) is adapted from Leake and Schack (2023) and Schack (2023).

The International Conference on Case-Based Reasoning provided travel funding to present this research. The Professional Development Grant from the Ursuline Center for Teaching and Learning, the Hart Travel Stipend from Brescia University Academic Affairs, and the Marshall Heuser and Mimi Gatus Mazzier-Heuser International Travel Grant from Brescia University Academic Affairs provided funding to co-present the research in Chapter 5. The Department of Defense (Contract W52P1J2093009) and the Office of Naval Research in the Department of the Navy (Award N00014-19-1-2655) provided funding for the research in Chapter 6.

[1]https://math.indiana.edu/student-portal/graduate/iu-thesis-class.html

Brian Schack

CASE-BASE MAINTENANCE BEYOND CASE DELETION

Case-based reasoning, a method of artificial intelligence, solves a problem by adapting the solution to a similar problem already solved. *Case-base maintenance* strategies support a compact, competent case base by deleting, modifying, or discovering cases. This dissertation presents four case-base maintenance strategies: flexible feature deletion, adaptation-guided feature deletion, expansion-contraction compression, and predictive case discovery. *Flexible feature deletion* deletes components of cases instead of whole cases. *Adaptation-guided feature deletion* prioritizes components for deletion according to their recoverability via adaptation knowledge. *Expansion-contraction compression*, in addition to deleting cases, also adds cases in unexplored regions of the problem space. And *predictive case discovery* anticipates and acquires cases expected to be useful for solving future problems. Evaluation of these case-base maintenance strategies compared to appropriate baselines on suitable data sets generally shows improvements in adaptation cost, competence, or solution quality.

Contents

Chapter 1

Introduction

Case-based reasoning (CBR) is a method of artificial intelligence which solves a problem by adapting the solution to a similar problem already solved (Aamodt & Plaza, 1994; de Mantaras et al., 2005; Kolodner, 1993; Leake, 1996a; Riesbeck & Schank, 2013). Case-based reasoning works similarly to how a judge decides a legal case by applying precedents (Ashley, 1986). Or how an unknown illness reminds a doctor of another patient with similar symptoms (Holt et al., 2005). Or, going back to the sixth century, how a priest decided a penance for a parishioner by referring to a penitential describing the sin under similar circumstances (Jonsen & Toulmin, 1988).

A large case base contains many cases that could potentially solve a wide range of future problems. On the other hand, a small case base takes less storage, less time to search through, less time to trasmit over a network, and less time to manually review. *Case-base maintenance* strategies navigate this trade-off by choosing the most valuable cases to retain and the least valuable cases to delete in order to yield a case base that is both competent and compact. This dissertation proposes and evaluates four case-base maintenance strategies: flexible feature deletion, adaptation-guided feature deletion, expansion contraction-compression, and predictive case discovery.

1.1 The Case-Based Reasoning Cycle and Knowledge Containers

Case-based reasoning is considered a methodology for problem solving because it describes a set of guiding principles rather than prescribing a specific technology (Watson, 1999). It traces its roots (Richter & Aamodt, 2005) back to both cognitive science (Kolodner, 1996; Leake, 1998; Ross, 1989) and casuistry (Searing, 2009) — specifically studying how people remember and solve problems. While case-based reasoning can use various technologies such as nearest neighbor, induction, fuzzy logic, or SQL, it is not limited to these specific technologies. Instead, case-based reasoning can be implemented using any technology that adheres to its guiding principles. Therefore, case-based reasoning is best viewed as a general methodology for building knowledge-based systems rather than an isolated technology for specific tasks.

Dynamic memory is an open-ended model of learning which uses remindings to explain failures of expectations instead of categories to classify knowledge (Schank, 1982, 1999). It formed the basis for the earliest case-based reasoning software like CYRUS and IPP. The *Computerized Yale Retrieval and Updating System (CYRUS)* modeled long-term memory and fact retrieval for events in the lives of important people (Kolodner, 1983), and the *Integrated Partial Parser (IPP)* read and generalized from news stories using memory-based parsing techniques (Lebowitz, 1983). Since then, case-based reasoning has been applied to a wide variety of domains, including medicine (Bichindaritz & Marling, 2006; Holt et al., 2005), law (Rissland et al., 2005), engineering (Shokouhi, 2012; Waheed & Adeli, 2005), and finance (Oh & Kim, 2007). Today, case-based reasoning continues to be an active area of research, with ongoing work on new algorithms, representations, and applications.

The *case-based reasoning cycle* (shown in Figure 1.1 on page 3) consists of four phases: retrieval, reuse, revision, and retention (Aamodt & Plaza, 1994; de Mantaras et al., 2005). Each phase works in turn to ultimately solve a given problem. First, the *retrieval phase* searches the case base for other problems similar to the given problem. Second, the *reuse phase* identifies the differences between the given problem and the retrieved case in order to adapt the solution to the retrieved case to account for those differences. Third, the *revision phase* attempts to apply the adapted solution in a simulation or the real world. Based on the performance of the solution, at times worse than expected, this phase modifies the solution to improve it. Finally, the *retention phase* packages the given problem and the revised solution as a new case. If the case base has sufficient space and the new case provides sufficient value, then this phase saves the case into the case base. Future iterations of the case-based reasoning cycle may use this new case to solve future problems.

Case-based reasoning has four *knowledge containers* (shown in Figure 1.2 on page 3) which each

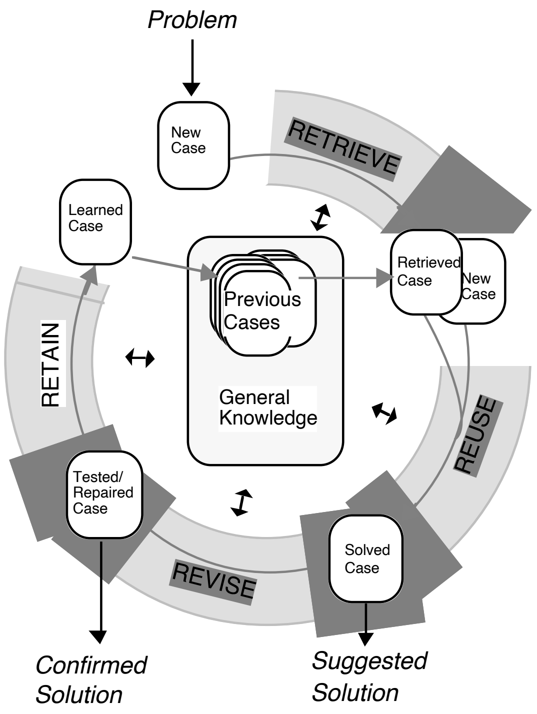


Figure 1.1: The *case-based reasoning cycle* consists of four phases: retrieval, reuse, revision, and retention (Aamodt & Plaza, 1994).

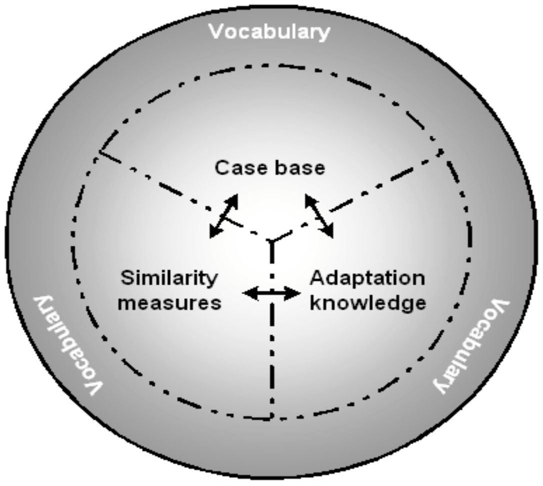


Figure 1.2: Case-based reasoning has four *knowledge containers* which each contain knowledge to support an aspect of the case-based reasoning cycle: vocabulary, case base, similarity measure, and solution transformations (Richter, 2003).

contain knowledge to support the case-based reasoning cycle: vocabulary, case base, similarity measure, and solution transformations (Richter, 2003). The *vocabulary container* stores the potential features of a problem or solution, the range of potential values, and the relationship between dependent features. The *case base* contains problems with known solutions either seeded by the training data or solved in the past. A *case* contains features and their values and has more or less structure depending on the domain. The *similarity measure* contains the function for calculating the resemblance between problems. It may compare the values of features with a specific weighting. The *solution transformation container* stores potential adaptations to a solution to account for differences between problems.

1.2 Comparison to Other Machine Learning Methods

Machine learning, a field of artificial intelligence, studies how to develop computer software that can progressively improve with experience and without explicit programming for a specific task (Mitchell, 1997). Case-based reasoning and various methods of machine learning like analogical reasoning, artificial neural networks, decision trees, explanation-based learning, k-nearest neighbors, and rule induction share similarities, but case-based reasoning also has key differences from alternative approaches (Leake, 1996b).

Analogical reasoning discovers and maps parallels between the relational structure of systems or reasoning steps usually from different domains (Burstein, 1989; Forbus et al., 1995). Analogical reasoning and case-based reasoning share similar processes, but they focus on different aspects. The standard schema for analogical reasoning, the *four-element comparison*, follows the syntactic form "W : X :: Y : Z" (Collins & Burstein, 1987). The *difference heuristic* approach in case-based reasoning applies this schema as "Saved Problem : Saved Solution :: Given Problem : Adapted Solution" in order to learn adaptation rules (Hanney & Keane, 2005; McSherry, 2006).

Analogical reasoning focuses primarily on the similarity of the compared structures, whereas case-based reasoning also adds a focus on retrieval and adaptation. Analogical reasoning can apply both across different domains and within the same domain (Carbonell, 1985), whereas the cases within a case base usually fall into the same domain, with the exception of case-based creativity (Schank & Leake, 1989) and cross-domain transfer learning (Sharma et al., 2007). In addition to specific cases, analogical reasoning can manipulate generalized concepts or domain principles. In contrast, while a case-based reasoning system may index its case base with generalized structures, the cases normally describe specific examples.

Inspired by animal brains, *artificial neural networks* loosely model interconnected neurons which process signals and then signal their neighbors (Haykin, 2004). Neural networks have garnered considerable interest from academia and industry for their effectiveness in computer vision (Krizhevsky et al., 2012), natural language processing (Collobert & Weston, 2008), and speech recognition (Hinton et al., 2012). Application of neural networks has outpaced theory, however, so researchers usually take a black box approach with empirical confirmation of methods and limited explanation for results (Samek et al., 2017). In contrast, *explainable artificial intelligence*, such as case-based reasoning, differs from neural networks because it follows a white box model which allows for human-interpretable justification of its methods and results (Core et al., 2006; Gunning & Aha, 2019; Leake & McSherry, 2005).

Decision tree learning constructs a tree from training data where each internal node tests an attribute and each leaf node draws a conclusion (Quinlan, 1986; Safavian & Landgrebe, 1991). Decision trees and case-based reasoning both share a common advantage of explainability. On the other hand, minor changes or additions to the training data can require reconstruction of a decision tree, whereas case-based reasoning can add and remove cases incrementally.

Explanation-based learning generalizes from instances by exploiting a strong domain theory to analyze why each specific instance exemplifies a broader concept (Minton et al., 1989). Explanation-based learning tends to generalize as much as possible, whereas case-based reasoning tends to keep its cases specific. Both explanation-based learning and case-based reasoning can suffer from the swamping utility problem (described in “Section 1.3: The Swamping Utility Problem and Case-Base Maintenance” on page 7) as either the proof macros or the cases, respectively, grow too numerous (Minton, 1990; Smyth & Cunningham, 2005).

The *k-nearest neighbors algorithm (kNN)* determines the label of an instance depending on the majority vote or average of the labels of its neighbors without maintaining any abstractions (Aha et al., 1991; Kramer, 2013). Both k-nearest neighbors and case-based reasoning search for training examples within a multi-dimensional feature space. In a way, the k-nearest neighbors algorithm specializes case-based reasoning by limiting the structure of cases to feature vectors and limiting the adaptation rules to voting and averaging. The k-nearest neighbors algorithm represents each potential neighbor as a feature vector with a class label, but case-based reasoning can represent cases with more or less structure depending on the domain. Also, the k-nearest neighbors algorithm chooses a solution by voting or averaging between the neighbors — often weighted according to their distance from the query — but case-based reasoning adapts the retrieved case to the problem via one or more solution transformation rules.

Rule induction, another area of machine learning, extracts formal rules which model patterns in training examples (Cohen, 1995). Both rule induction and case-based reasoning generalize from examples but in very different ways. Rule induction eagerly extracts rules from the training data, whereas case-based reasoning lazily postpones the generalization until given a problem to solve. In a rich domain with many possible generalizations, rule induction faces a difficult decision of which direction or directions to generalize in anticipation of an unknown future problem. Case-based reasoning defers this decision by always generalizing in the direction of a given problem.

1.3 The Swamping Utility Problem and Case-Base Maintenance

After the initial engineering of a knowledge-based system, how does the system sustain effective performance over its lifetime? *Knowledge maintenance* refers to the process of updating and refining a knowledge base to ensure that it remains accurate, relevant, and useful (Menzies, 1999). This process involves identifying and correcting errors or inconsistencies in the existing knowledge, as well as adding new information and insights as they become available. Knowledge maintenance is important for ensuring the long-term effectiveness of knowledge-based systems, as outdated or inaccurate information can reduce the system's usefulness and credibility. It typically involves a range of activities, such as data cleaning, knowledge validation, and knowledge updating, and may be automated or performed manually depending on the complexity of the knowledge base and the resources available for maintenance.

A case base can contain cases from training data, knowledge engineered by human experts, and the retention phase of the case-based reasoning cycle. On one hand, each case could potentially, through adaptation, solve future problems. If other cases could not solve these problems or could only solve them with greater adaptation cost or inferior solution quality, then that case contributes to overall problem-solving competence. On the other hand, each retained case makes the case base larger. A larger case base requires more storage, more time to search through, more bandwidth to transmit over a network, and more attention to manually review.

The *swamping utility problem* describes the trade-off between competence contribution and retrieval cost (Francis & Ram, 1993; Minton, 1990; Smyth & Cunningham, 2005). More broadly, this trade-off could also involve other factors such as solution quality, storage cost, and bandwidth requirements. Legacy systems, embedded systems, and unreliable networks worsen the problem by constraining resources. Big data and streaming data also worsen the problem by increasing resource usage.

Much research over the years has attempted to mitigate the utility problem (Juarez et al., 2018).

Case-base maintenance strategies judiciously choose the most valuable cases to retain and the least valuable cases to delete in order to maintain a compact, competent case base. The strategies differ in the order that they delete cases, and like machine learning algorithms in general, their effectiveness depends on their suitability to a particular dataset (Aha, 1992).

A popular maintenance strategy involves estimating the trade-off between coverage and reachability (Smyth & Keane, 1995). *Coverage* approximates the number of neighboring problems that a case can solve through adaptation. *Reachability* approximates the number of neighboring cases that can solve a problem through adaptation. More coverage tends to make a case more valuable to retain, and more reachability tends to make a case more expendable through deletion. The coverage and reachability sets of cases can intersect in interesting ways, so deleting a case while minimizing the resulting loss in overall problem-solving competence is like playing a game of n-dimensional Jenga where removing any brick could cause the tower to topple.

No maintenance strategy can calculate coverage and reachability exactly because the competence contribution of a case depends on its applicability to problems encountered in the future which the strategy cannot predict with certainty in the present. Therefore, case-base maintenance strategies rely on the *representativeness assumption* which states that future problems encountered will follow the same distribution within the domain as existing cases in the case base (Smyth, 2005). For domains where case-based reasoning works effectively, this assumption tends to hold reasonably well in practice.

1.4 A Broader View of Case-Base Maintenance

Research in case-based reasoning normally justifies case-base maintenance as solving two problems: reducing the size of the stored case base and reducing the retrieval time for solving problems (Francis & Ram, 1995; van Someren et al., 2005). But improvements to indexing and parallelization have substantially mitigated the swamping utility problem in all but the largest case bases (Houeland &

Aamodt, 2010). And, although it may not continue indefinitely, since the inception of Moore's Law, the number of transistors in an integrated circuit has doubled about every two years — leading to faster speeds and reduced costs (Schaller, 1997). In light of this technological progress, does case-base maintenance still matter? This dissertation asserts that, despite these improvements, the problem of maintaining a compact and competent case base remains.

Imagine, if you will, an ideal scenario in which storage space approaches infinity and processing time approaches zero. In what ways could the maintenance phase contribute to performance? This section proposes four potential directions for researching a broader view of maintenance: bandwidth, creativity, licensing cost, and diversity.

The usage and importance of networked systems (especially the Internet) has grown massively over the decades since Smyth and Keane (1995) — leading to distributed case-based reasoning systems (Plaza & McGinty, 2005). Furthermore, application requirements continue to outpace available bandwidth — especially for mobile devices. For a networked system, rather than memory or processing power, bandwidth and latency limit performance. These limitations necessitate judicious decisions of which cases the nodes in the network should share with which neighbors and at what level of detail.

Normally, the compactness of a case base requires a corresponding trade-off in competence. But a recent finding shows that a kind of maintenance called creative destruction could, in limited circumstances, actually improve both compactness and competence by subdividing cases to make their components accessible to adaptations of limited power (Leake & Schack, 2016; Schack, 2016). This finding suggests the potential to look for further opportunities for creative destruction when developing maintenance strategies.

As generative artificial intelligence tackles difficult design tasks, it must face the same intellectual property constraints as human designers. Different laws such as patents, copyrights, trademarks, trade secrets, personal privacy, and *sui generis* ("of its own kind") database rights may apply to data depending on its domain and jurisdiction. Reasoning about this requires knowledge of case provenance (Leake & Whitehead, 2007). Compositional adaptation strategies must check the compatibility of the licenses of their component parts. One could imagine a maintenance strategy that minimizes the licensing cost of a case base independently from its size. For example, Netflix manually maintains their catalog of films and television shows for a dual purpose: to support the competence of their recommender system and to stay within their licensing budget.

The performance of a case-based reasoning system depends not only on the problem-solving time and the solution quality, but also the diversity of its solutions (Smyth & McClave, 2001). For example, a travel agency employing a recommender system that perpetually proposes the same destination would lose customers, regardless of the merit of its itineraries or the speed of its response. Therefore a maintenance strategy could consider diversity as another factor when deleting or refactoring cases or case components.

Considering the swamping utility problem, the maintenance strategies developed to mitigate it, and this broader view of maintenance, several key questions are raised: How can a maintenance strategy minimize the loss in competence as it compresses a case base further and further? How do case bases compressed with different strategies to the same level of compression differ in their ability to solve problems? How do they differ in the quality of the solutions retrieved? How do they differ in the speed of retrieval? Which case bases call for the application of which maintenance strategies?

1.5 Thesis Statement

Main Claim: Case-based reasoning can benefit from maintenance strategies that involve more than merely deleting whole cases.

First Sub-Claim: The *flexible feature deletion* case-base maintenance strategy can compress suitable case bases with less loss in competence by deleting components of cases instead of whole cases (Leake & Schack, 2015; Schack & Summers, 2017).

Second Sub-Claim: The *adaptation-guided feature deletion* case-base maintenance strategy can compress suitable case bases with less loss in competence by prioritizing components of cases for deletion according to recoverability via adaptation knowledge (Leake & Schack, 2016; Schack, 2016).

Third Sub-Claim: The *expansion-contraction compression* case-base maintenance strategy can compress suitable, unrepresentative case bases with less loss in competence by, in addition to deleting cases, also selectively adding cases (Leake & Schack, 2018; Schack, 2019).

Fourth Sub-Claim: The *predictive case discovery* case-base maintenance strategy can reduce adaptation cost for suitable case bases by predicting and acquiring cases expected to be useful for solving future problems (Leake & Schack, 2023; Schack, 2023).

1.6 Dissertation Outline

The following outline gives an overview of the chapters in this dissertation, and Sections 1.7 - 1.10 on pages 12 - 17 provide a preview of Chapters 3 - 6.

1. **Introduction:** Introduce case-based reasoning, describe the case-based reasoning cycle and knowledge containers, place case-based reasoning in the context of other methods of machine learning, and explain the swamping utility problem which motivates case-base maintenance.

2. **Evolution of Case-Base Maintenance:** Offer an overview of the historical progression of case-base maintenance, analyze modern approaches, and link past methodologies to present ones.

3. **Subdividing Cases for Deletion:** Explain the assumptions of uniform storage cost and indivisible cases, propose the flexible feature deletion case-base maintenance strategy, and eval-

uate flexible feature deletion in comparison to per-case maintenance strategies in terms of retrieval accuracy and time.

4. **Prioritizing Deletion by Recoverability via Adaptation Knowledge:** Define the concept of recoverability, propose the adaptation-guided feature deletion case-base maintenance strategy which prioritizes features by recoverability, explain the creative destruction phenomenon, and evaluate adaptation-guided feature deletion in comparison to knowledge-light flexible feature deletion in terms of competence retention, solution quality retention, and processing time.

5. **Data Augmentation to Expand Choices for Deletion:** Explain the representativeness assumption, overfitting, and case discovery; propose expansion-contraction compression which applies data augmentation to case-base maintenance; evaluate expansion-contraction compression in comparison to condensed nearest neighbor in terms of retention in competence and solution quality.

6. **Predictive Case Discovery for Problem-Distribution Drift:** Explain competence groups and competence holes, propose a case-base maintenance strategy which targets data augmentation to fill competence holes between nearby competence groups, and evaluate the targeted strategy in comparison to untargeted expansion-contraction compression in terms of competence and solution quality.

7. **Conclusion:** Discuss the implications of the four proposed strategies, envision future work on case-base maintenance, and conclude by restating the key contributions of this dissertation.

1.7 Subdividing Cases for Deletion

Case-base maintenance (D. C. Wilson & Leake, 2002) is an active area of case-based reasoning research. Much of this work develops methods to compress the case base, such as competence-based case deletion (Smyth & Keane, 1995), deletion methods taking class boundaries into account by considering local complexity (Craw et al., 2007), optimizing the trade-off between size and accuracy (Lupiani et al., 2013), deletion aimed at preserving diversity (Lieber, 2005), strategies for case retention and forgetting (Muñoz-Avila, 1999; Ontañón & Plaza, 2003; Romdhane & Lamontagne, 2008; Salamó & López-Sánchez, 2011b), deletion aimed at preserving adaptation efficiency (Leake & Wilson, 2003), and deletion aimed at increasing accuracy (Marquer et al., 2023).

Case-base maintenance strategies, whether based on coverage and reachability or not, normally make two assumptions: (a) that all cases have a uniform storage cost and (b) that they must retain or delete whole cases. This dissertation proposes and evaluates *flexible feature deletion (FFD)*, a knowledge-light case-base maintenance strategy which, in contrast, subdivides variable-size cases for deletion of their components (Leake & Schack, 2015; Schack & Summers, 2017).

Cases can have varying storage cost when they contain varying amounts of information at varying levels of detail. The storage cost of both the problem and the solution can vary independently because a simple problem may have a complex solution and vice versa. This suggests balancing the competence contribution of a case against its storage cost.

A case-base maintenance strategy could delete an entire case, but it could also delete a single feature across all cases, or a single feature from a single case. Each of these alternatives presumably degrades problem-solving competence but not necessarily to the same degree. Compared to per-case strategies, flexible feature deletion can reduce the size of the case base with less reduction in the number of cases. It can also vary in the metric that it uses to order features for deletion. Each of the variations uses a knowledge-light metric like the size of a case, the rarity of a feature, or a hybrid of multiple metrics.

Research on maintenance of case contents has generally focused on quality improvement rather than case base compression (Racine & Yang, 2005; Salamó & López-Sánchez, 2011a). However, research on case abstraction, in aiming to compact the case base by removing concrete cases subsumed by abstractions (Bergmann & Wilke, 2005), can be seen as in the spirit of replacing cases with more compact versions.

Domains with large cases and multiple representations call for the application of flexible feature deletion. For example, cases based on medical imagery (D. C. Wilson & O'Sullivan, 2008) may have various resolutions and a large number of features of which only some are relevant to the diagnosis.

Flexible feature deletion can also apply to indexing features as well as cases. Maintenance of indexing features has been extensively studied in case-based reasoning, applying methods such as feature deletion, addition, and reweighting, but again with the goal of improving retrieval accuracy rather than decreasing the storage required for the indices themselves (Arshadi & Jurisica, 2005; Fox & Leake, 2005; Muñoz-Avila, 2002; Zhang & Yang, 2006). Feature set reduction has been combined with case selection to improve accuracy while compressing the case base (Li et al., 2006).

1.8 Prioritizing Deletion by Recoverability via Adaptation Knowledge

The adaptation-guided feature deletion case-base maintenance strategy builds on flexible feature deletion (described in "Section 1.7: Flexible Feature Deletion" on page 12 and "Chapter 3: Subdividing Cases for Deletion" on page 26). Whereas flexible feature deletion orders the features according to a knowledge-light metric, *adaptation-guided feature deletion (AGFD)* integrates additional knowledge from the solution transformation container about the recoverability of features (Leake & Schack, 2016; Schack, 2016). Similar to how reachability measures the ability of adaptation knowledge applied to other cases to restore the solution to a case considered for deletion, *recoverability* measures the ability of adaptation knowledge applied to other features to restore a feature considered for deletion.

A solution with recovered features may either match exactly the original uncompressed solution, or it may solve the same problem in a different way. Compression to smaller sizes can increase the time required for recovery and decrease the quality of the recovered solution until adaptation knowledge can no longer recover any solution at all. Therefore, in order to preserve problem-solving competence, adaptation-guided feature deletion deletes features in order from most recoverable to least.

In addition to deleting features, adaptation-guided feature deletion can also replace them with a smaller substitution or abstraction. Occasionally, this reorganization can make case contents more accessible to an adaptation rule of limited power. Even though case-base compression normally reduces competence, compression under these circumstances, termed *creative destruction*, can improve competence instead (Leake & Schack, 2016).

Adaptation-guided feature deletion relates to the many approaches in case-based reasoning which address the construction of compact competent case bases (several are surveyed in D. C. Wilson and Leake, 2002). These approaches include case retention and forgetting strategies (Muñoz-Avila, 1999; Ontañón & Plaza, 2003; Romdhane & Lamontagne, 2008; Salamó & López-Sánchez, 2011b), diversity-preserving deletion strategies (Lieber, 2005), making a trade-off between accuracy and case base size (Lupiani et al., 2013), taking into account local complexity in order to consider class boundaries (Craw et al., 2007), and competence-based deletion of cases (Smyth & Keane, 1995). Unlike adaptation-guided feature deletion, however, all of these methods assume that cases are indivisible.

When case-base maintenance research has considered internal contents of cases, its goal has generally been to improve the quality of the contents (Racine & Yang, 2005; Salamó & López-Sánchez, 2011a), whereas the goal of adaptation-guided feature deletion is to reduce case size. Some research on case-based abstraction has replaced concrete cases with abstractions (Bergmann & Wilke, 2005) which is similar to the more fine-grained substructure abstraction operation. Most similar to adaptation-guided feature deletion is work to consider the removal of parts of cases in the context of maintenance to control case-base size for preference-based case-based reasoning (Abdel-Aziz & Hüllermeier, 2015). The idea of connecting adaptation directly to retention can be seen in the same spirit as adaptation-guided retrieval, which connects adaptation to similarity assessment (Smyth & Keane, 1998).

1.9 Data Augmentation to Expand Choices for Deletion

As described in "Section 1.3: The Swamping Utility Problem and Case-Base Maintenance" (on page 7), by the representativeness assumption, maintenance strategies predict that future problems will follow the same distribution as the current case base, and this works reasonably well for mature case bases in stable domains. But the representativeness assumption may apply less accurately during early case base growth, to dynamically changing domains, or in cross-domain transfer learning. In these situations, case-base maintenance strategies optimizing for assumed representativeness may instead cause overfitting. *Overfitting* means that a statistical model or a machine learning algorithm makes predictions based on peculiarities in the training data not reflected in the testing data, thereby improving performance on the training data and sacrificing performance on the testing data (Dietterich, 1995).

The overfitting problem has received significant attention in the context of artificial neural networks (Lawrence et al., 1997). Among several potential mitigations, neural networks may employ *data augmentation* which perturbs training data in order to supplement it with additional instances (Wong et al., 2016). For example, cropping images without obscuring their subjects or other minor deformations which maintain overall cohesion.

Case-based reasoning does not normally apply data augmentation, but the solution transformation container provides a natural source for such perturbations. *Expansion-contraction compression (ECC)* explores unseen regions of the problem space using adaptation knowledge to generate *ghost cases* and then exploits the ghost cases to broaden the range of cases available for competence-based deletion (Leake & Schack, 2018; Schack, 2019).

1.10 Predictive Case Discovery for Problem-Distribution Drift

Case-based reasoning systems depend on *problem-distribution regularity*, a concept formalized by (Leake & Wilson, 1999), which says that "future problems will resemble past problems." This regularity is vital for the relevance and applicability of learned cases in addressing future problems, ensuring that the stored cases from past instances are informative and applicable to subsequent instances. The assumption that future problems are reflective of past ones is necessary to ensure that the case base maintains its utility and relevance over time.

However, the assurance of such regularity is not absolute. With the progression of time and the evolution of systems, this regularity can degrade when the distribution of problems changes. Drifts in problem distribution can significantly impact the coverage of a case base, necessitating maintenance to uphold effectiveness.

When there is a lack of problem-distribution regularity, it leads to what is referred to as *problem-distribution drift* (Leake & Schack, 2023; Schack, 2023). This form of drift is different from *concept drift*, a well-researched phenomenon in machine learning, where the relationships between problems and solutions change over time, rendering previous cases irrelevant or inaccurate (J. Lu et al., 2018; Widmer & Kubat, 1996). Concept drift concerns the changes in the underlying concepts or solutions, while problem-distribution drift involves changes in the occurrences and types of problems encountered by the system.

Problem-distribution drift can occur in diverse domains like disaster management or recommender systems. For example, climate change-induced alterations in weather patterns could invalidate the response plans of a disaster management system that are based on historical weather data. Likewise, shifts in consumer preferences could necessitate an overhaul in the range of recommendations from a travel agency.

Furthermore, *adversarial drift* occurs when an adversary presents characteristically different cases over time with the intention of degrading performance (Kantchelian et al., 2013). This form of drift, characterized by intentional alterations and disruptions, can exacerbate problem-distribution drift. And it can manifest in domains where strategic deceptions or obfuscations are advantageous, such as in imperfect information games, cybersecurity, or junk mail filtering (Delany et al., 2005).

Drift detection can be categorized into four general strategies: error rate-based, data distribution-based, multiple hypothesis-based, and competence-modeling strategies. For example, the ADWIN algorithm is error rate-based and adapts the window size based on changes in the error rate (Bifet & Gavaldà, 2007). The Kullback-Leibler divergence-based method, a data distribution-based strategy, determines drift by measuring discrepancies between two probability distributions (Dasu et al., 2006). Another example is Just-in-Time adaptive classifiers (JIT), a multiple hypothesis-based strategy, utilizing a sequence of hypothesis tests to detect changes in data distribution (Alippi & Roveri, 2008). Competence modeling is a strategy in case-based reasoning which provides descriptions and quantification of changes as well as statistical guarantees on reliability (N. Lu et al., 2014).

However, the effectiveness of drift detection can be impacted by the curse of dimensionality. The *curse of dimensionality* refers to the problem of high dimensionality in a problem space leading to sparsity and high computational costs (Köppen, 2000). The increase in dimensions makes cases more dispersed, and changes in their distribution become challenging to detect. One way to address this is to select relevant features to reduce dataset dimensionality before applying drift detection algorithms.

To counteract decreasing problem-distribution regularity, cases can be added to the case base. By calling upon a generative component or requesting cases from external sources, such as domain experts, gaps in case distribution can be filled, reducing potential future slow-downs or failures in those gaps. For example, solutions generated in advance by systems like Prodigy / Analogy do not increase competence but speed up learning by mitigating the need to create solutions from scratch during runtime (Veloso, 1994). The effectiveness of case discovery, given the high cost of case solicitation, relies on precise targeting.

The strategy proposed by McKenna and Smyth (2002) focuses on identifying competence holes to fill by discovering spanning cases. Methods like the SMOTE oversampling algorithm generate synthetic instances to address class imbalance (Fernández et al., 2018). Similar to data augmentation in neural networks, adaptation rules in case-based reasoning can create “ghost cases” that maintain case cohesion and improve efficiency (Leake & Schack, 2018; Schack, 2019).

This dissertation proposes a *predictive case discovery* strategy which involves dividing the problem space into parts, predicting the most active part, selecting a point in that part, and then discovering that case (Leake & Schack, 2023; Schack, 2023). This approach is applied through k-means discovery, where the problem space is divided into N regions. A random cluster is chosen, and a case at the centroid is altered to generate a variant. Clustering-based case discovery is beneficial where domain knowledge is scarce or costly, offering a representative variant of a cluster which is hypothesized to reflect “hot spots.” Alternative methods like spherical k-means or affinity propagation can replace k-means as needed, depending on the domain.

Each strategy for drift detection and case discovery can have varying effectiveness, and choosing suitable strategies and parameters is pivotal for accuracy and efficiency. A possible solution is developing a library of strategies and selecting them via a bandit meta-strategy. This allows for refining choices to favor successful strategies based on the problems addressed in the past. However, swift changes in problem distribution may render this knowledge obsolete, necessitating further research on this topic.

Chapter 2

Evolution of Case-Base Maintenance

Case-based reasoning learns from experience and adapts to changing environments by incrementally saving cases that it acquires into the case base. As case-base maintenance manages, deletes, and condenses those cases, this provides the benefit of the continuous availability of an up-to-date case base. Given the advent of big data, novel maintenance strategies have addressed the evolving needs of scenarios with varied and voluminous data. This chapter offers an overview of the historical progression of case-base maintenance, analyzes modern approaches, and links past methodologies to present ones. For further comparison and contextualization of approaches to case-base maintenance, please see Juarez et al. (2018) and D. C. Wilson and Leake (2002).

2.1 Historical Context and Modern Developments

The case base serves as a reservoir of cases, which are used to address and solve new problems (Craw, 2011). The design and maintenance of case bases is particularly important in the age of big data, ensuring the accuracy and effectiveness of case-based reasoning systems (Goel & Diaz-Agudo, 2017).

Conventional case-base maintenance algorithms have largely focused on reducing the number of cases by identifying redundant and noisy cases. Early efforts were directed towards studying nearest neighbor and instance-based learning methods, using algorithms such as Condensed Nearest Neighbor (CNN), considered one of the first algorithms for case-base maintenance (Hart, 1968), and other similar models like Reduced NN (RNN), Edited NN (ENN), and Selective NN (SNN) (Salamó & López-Sánchez, 2011a). However, these encountered challenges, particularly regarding sensitivity to noisy cases.

The idea of competence models diverged from viewing case-base maintenance as merely an instance selection problem (Smyth & Keane, 1995; Smyth & McKenna, 2002). It highlighted the significant role of each case in the cycle of case-based reasoning. The Smyth-Keane-McKenna Competence Model was influential, emphasizing the need to include cases based on their ability to solve problems in a case-based reasoning system context. Numerous algorithms were inspired by or directly based on the competence model, such as COV, RFD, RC, CTE, CRR, ICF, and CBE (Craw et al., 2007; Delany & Cunningham, 2004; Smiti & Elouedi, 2014).

2.2 Partitioning Approaches and Algorithm Adaptations

Several approaches and algorithms focus on the partitioning of the case base. These consider the case base as a whole and select cases to remove by dividing the case base into subsets. Each subset is then treated as an independent case-base (Smiti & Elouedi, 2014, 2018). An exemplary algorithm following this approach is WCOID-GM, which combines machine learning techniques to learn the weights of case attributes, apply DBSCAN-based clustering methods, and employ univariate outlier detection methods. Furthermore, the SCBM algorithm, an advanced version of WCOID-GM, integrates a competence model. It uses a fuzzy-based DBSCAN method for clustering the case-base and evaluates the competence of each cluster by analyzing different types of cases, including noisy, similar, and isolated cases.

2.3 Enhancements to Case-Base Maintenance Algorithms

Highlighting the importance of maintaining competent case bases, N. Lu et al. (2014) pointed out that the current competence group method is not optimal specifically critiquing its allowance for disjoint partitions within each group without ensuring complete splitting. To address these limitations, they introduced an extended model that consists of two new concepts: competence closure and related closure.

In this enhanced model, competence closure focuses on defining groups that exhibit a shared coverage path, emphasizing the exclusion of any shared coverage with elements outside the group. Related closure, on the other hand, broadens the related set concept. These extensions are used to propose two new competence measures, aiming to weigh the related sets within a related closure and to offer a competence-based empirical weight to assess case distribution within a competence cluster. This approach results in more defined and disjoint sets in comparison to previous methods, effectively improving the structure and clarity of competence clusters.

2.4 Restructuring Case Bases

Many case-base maintenance algorithms emphasize the uniformity of case structure. However, the integration of varied data sources may necessitate a redefinition of such structures. In response, several methods have been introduced. One notable method is compositional adaptation (CA), which focuses on representing the dependency between cases (Mathew & Chakraborti, 2017). This approach adopts an AND-graph representation, where nodes symbolize cases and edges represent solving capacities of single or combined cases. This approach thereby offers a refined comprehension of case dependencies and solution formation.

2.5 Preference Modeling and Temporal Maintenance

The preference model (Pref-CBM) introduced by Abdel-Aziz and Hüllermeier (2015) offers a nuanced version of case-based reasoning by integrating preference as a component of a case. It redefines the conventional problem-solution relationship by decomposing cases into smaller knowledge chunks and examining preference in solutions. The authors evaluated this method on the traveling salesman problem.

Further, Lupiani et al. (2014) focus on how case-based reasoning systems evolve over time and how this impacts case-base maintenance, emphasizing the necessity of temporal maintenance (T-CBM). This method extends the classic case structure to accommodate a sequence of heterogeneous events over time, requiring new distance measures and review of existing algorithms. T-CBM has proven effective at maintaining temporal case bases in risk scenario detection in commercial home-monitoring systems.

2.6 Concept Drift-Tolerant Maintenance (Drift-CBM)

The evolving nature of real-world data and the dynamic goals of intelligent systems necessitate the implementation of maintenance methodologies that can adapt to unforeseen changes, collectively termed the concept-drift problem. Such systems must mitigate accuracy degradation over time, a subject actively researched in machine learning but with minimal exploration in case-base maintenance literature (N. Lu et al., 2016).

Drift-CBM operates in two steps: Enhancement and Preservation. The Enhancement step ascertains whether a newly acquired case is noise, especially noise arising from concept drift, using the NEFCS algorithm. The subsequent Preservation step, contingent upon existing storage limits, employs the SRR algorithm to discard redundant cases. In this approach, competence-based drift detection uses related set density and competence-based weight. An essential feature of NEFCS is its ability to discern and retain novel cases in a detected drift-concept competence area, preventing their premature removal as noise. It employs the competence definition of a liability set (Delany & Cunningham, 2004).

2.7 Navigating Computational Complexity in Case-Base Maintenance

The computational complexity of case-base maintenance depends upon varying factors. Generally, case-based reasoning systems are employed in domains that lack comprehensive theoretical frameworks, rendering the efficacy of case-base maintenance algorithms dependent on either intricate heuristics or substantial computational costs (Smyth & McKenna, 2002). Even though absolute optimization is not always guaranteed, case-based reasoning algorithms typically yield reasonable results, often converging to satisfactory fitness values. The exploration of genetic algorithms (GAs) in this realm has shown prominence in solving complex problems where the knowledge base is relatively weak and analytical solutions are elusive.

2.8 Case-Base Near Insertion (CBNI)

Yamamoto et al. (2015) combines genetic algorithms with case-based reasoning for intelligent route optimization. This approach, referred to as Case-Based Human Oriented Genetic Algorithms (CB-HOGA), uses solutions to past problems as a basis for generating new solutions. The key contribution is a case-base maintenance strategy that attempts to maintain diversity and fitness within the case base. This is achieved by selecting and using previous genetic algorithm solutions to similar problems, which are then modified by human expert support.

The paper emphasizes reducing the role of chance ("accidence") in favor of an intentional and balanced search for fitness and diversity, particularly in cases related to human or cultural factors. The case base is combined with a Nearest Insertion (NI) method to incorporate and preserve human / cultural knowledge in the genetic algorithm. This knowledge, which is challenging to formally express, is maintained and inherited as the initial population for the genetic algorithm. Experiments are conducted to demonstrate the effectiveness of this approach in creating intentionally good solutions, rather than relying on accidental discoveries. The results show the benefits of this approach for route optimization problems, particularly those influenced by human and cultural factors.

2.9 Multi-Objective Evolutionary Case-Base Maintenance (MOE-CBM)

The performance of case-base maintenance algorithms significantly depends on the proportion of noisy and redundant cases within the case base. Lupiani et al. (2015) perceives general case-base maintenance as a multi-objective optimization problem with goals to minimize redundant cases and distance to non-redundant cases while maximizing the competence of the case-based reasoning system.

MOE-CBM is an adaptation of the multi-objective genetic algorithm, NSGA-II, and directs its focus towards exploring potential case bases in the problem space. It leverages noise and redundancy indicators to optimize its goals, recognizing that minimizing size and maximizing accuracy are conflicting objectives (Craw et al., 2007). Although MOE-CBM does not guarantee optimal solutions within finite time, practical implementations demonstrate its capability to approach acceptable solutions, albeit with extended runtime, constraining its use to offline processes.

Chapter 3

Subdividing Cases for Deletion

The performance of case-based reasoning systems depends on the coverage of their case bases and the quality of their cases. As the number of cases in the case base grows, increased retrieval costs (Francis & Ram, 1993; Smyth & Cunningham, 2005) or storage constraints may require controlling case base size. Extensive case-based reasoning research has aimed to address this problem through case-base maintenance (D. C. Wilson & Leake, 2002). A key focus of this work has been on strategies for selecting cases to retain in the case base to maximize the competence achieved for a given number of cases. Approaches include strategies for guiding the deletion of cases from an existing case base (Smyth & Keane, 1995), for determining when to retain a new case during problem solving (Muñoz-Avila, 1999), and for ordering the addition of cases from a candidate case set (Smyth & McKenna, 1999a; Zhu & Yang, 1999). All of these strategies treat cases as single units, adding or deleting entire cases. This dissertation calls such strategies "per-case" maintenance strategies.

Per-case strategies reflect two common implicit assumptions: (1) that all the cases in the case-based reasoning system will be of sufficiently uniform size so that the size effects of deletion or addition do not depend on the chosen case, and (2) that the size of the internal contents of cases cannot be reduced. In domains for which each case must contain uniform knowledge, and therefore removal of any case information would severely impair the ability to use the cases, per-case strategies are the only appropriate choice. However, in some case-based reasoning domains, case contents are more flexible.

This chapter questions the assumption of uniform case size in case-base maintenance. The assumption of uniform size means that, if cases are of different size, it is not possible, for example, to favor retention of smaller cases when those cases have comparable coverage. It also questions the assumption of maintenance only on a per-case basis, proposing that compression strategies can consider not only case deletion / addition but the deletion of components of particular cases. Rather than pre-determining a static set of features to be used throughout the life of the CBR system, the set of features to include in the case base could be adjusted based on requirements for storage, processing speed, and accuracy. There need be no requirement that all cases in the case base include the same set of features, just as there need not be uniform collections of components in the solution parts of cases, and the solutions need not be represented at the same level of granularity. This chapter proposes *flexible feature deletion (FFD)* in which selective compression can be done at the level of the contents of individual cases, by removing selected features from either indexing or solution information (Leake & Schack, 2015; Schack & Summers, 2017). This can be used to maintain both indexing features and features of a solution.

The motivation for adjusting case contents arises from domains in which cases are large and can be represented in multiple ways. For example, case-based reasoning has attracted interest for reasoning from imagery such as medical images (D. C. Wilson & O'Sullivan, 2008). From any image, different features may be extracted at different resolutions, and the amount of information required to represent different images might vary dramatically. In diagnostic domains, numerous features may carry information relevant to the diagnosis, with different pieces relevant to different degrees for different problems. When case-based reasoning is applied to design support, stored designs could selectively include different subsets of a full design or could include the design at different levels of detail. In a case-based planner generating highly complex plans, it is possible to retain the entire plan, or only key pieces, or to preserve full details for parts of the plans and high-level abstractions for others. Likewise, when case-based reasoning is applied to tasks such as aiding knowledge capture by supporting concept map construction (Leake et al., 2014), stored concept map cases could be retained at different levels of completeness. Exploiting this flexibility requires maintenance processes that can perform maintenance at a finer-grained level than simple retention or deletion of cases.

Feature compression is especially appropriate for complex domains in which cases are large, may contain extensive indexing or solution information, and in which partial information — for either indices or solutions — may still be useful. Feature compression prompts the question of when to delete an entire case versus when to achieve comparable space savings by abstracting, deleting, or otherwise compressing some of the features contained in one or more cases in the case base. There is no free lunch: either method may entail accuracy losses, case deletion, which removes what may be the most relevant solution to a problem; or flexible feature deletion, which reduces retrieval accuracy and solution quality. The interesting question is how these methods compare.

This chapter begins by discussing the range of applicability of flexible feature deletion and its relationship to standard case-base maintenance. It then defines a set of simple feature deletion strategies and evaluates their performance compared to per-case strategies for three domains, two with cases containing varying amounts of information and one with uniform size cases. This chapter analyzes competence as a function of compression, and this analysis supports flexible feature deletion for domains with variable-size cases.

3.1 When Feature Deletion Is Appropriate

Feature compression is appropriate for a particular class of domains: Those in which a particular case can be represented at varying levels of detail and still be useful. Feature deletion impacts the similarity metric, so it may reduce the retrieval accuracy. But even if retrieval accuracy is reduced, the retrieved cases will provide value if they are still adaptable to usable solutions with an acceptable level of adaptation effort. Even if some poor retrievals result, they may be acceptable given savings in space; just as per-case maintenance usually involves a trade-off of case base compactness against competence, feature deletion does as well.

The range of problems to which the case can be applied, and the reliability of its application, may vary with the specific information stored. Consequently, different feature deletion domains will exhibit different trade-offs between per-case maintenance strategies and feature-maintenance strategies, as well as different trade-offs between compression and quality. For some domains, such as a regression or numerical prediction task, feature deletion may only be possible for indexing information. In domains in which indexing information is based on many features, it may be possible to reduce case size by removing information about the values of some indices. Note that feature deletion of indices contrasts with the extensive work on selecting indexing vocabularies in the case-based reasoning literature, in that feature maintenance is aimed not at selecting an indexing vocabulary or maximizing retrieval accuracy, but instead at selectively compressing indexing information by deleting particular features, potentially from individual cases, with the recognition that some accuracy loss may result.

The deletion done by feature deletion is not necessarily limited to particular indexing dimensions (e.g., deleting the "age" attribute from all patient cases). Alternatively, it may delete specific attribute values (e.g., deleting the "age" attribute-value pair for specific patients, or only for a particular range of age values, such as those patients who fall into a default set for which age is not considered significant).

Feature deletion for indices could be especially relevant to situations in which extremely rich indexing information is available, as when a case-based agent responds in a real-time strategy game, or a prediction system for driver behavior, where the situation in which a plan was applied could be described with extremely rich detail — with fine-grained details which might be helpful to finding the perfect case, but not essential to finding a good case. Likewise, in a movie recommender domain, with movies characterized by their list of actors and the goal of recommending similar movies, a subset of the actors might be sufficient for good retrievals.

3.1.1 When to Apply Feature Deletion to Indices

Tasks are potential targets for feature deletion of indices if their cases have large indexing structures which can be reduced while retaining an acceptable level of indexing / similarity performance. Specifically, domains are appropriate if:

- *Indexing or similarity assessment depends on information about detail-rich situations from which many features could be generated.* If any low-level features of the current situation, or of a sequence of situations, might be available and potentially relevant to deciding a response, then, due to the potential for large amounts of indexing information, feature deletion could significantly impact the size of the case base.
- *Indexing or similarity assessment features are sufficiently closely related that acceptable accuracy is possible after removal of some features.* If features are closely related — even if they are not redundant — feature removal may have limited effects on system accuracy, helping to boost the amount of compression possible per unit of retrieval accuracy loss.

The case-based reasoning community has devoted substantial effort to methods for refining the indices used for cases, as well as on developing methods for assigning weights to features for similarity assessment. However, work in index / similarity refinement differs from feature deletion in a key way: The focus of index / similarity refinement is generally on increasing retrieval accuracy, rather than on compression of case data. Consequently, research on such methods does not address space / accuracy trade-offs. Feature deletion is a primary focus of research on dimensionality reduction for case-based reasoning. However, such deletion is done uniformly across all cases; this work does not attempt selective deletion of a feature from some cases but not others.

3.1.2 When to Apply Feature Deletion to Solutions

Feature deletion is useful for domains in which the solution to a single problem can capture varying levels of information and still be useful. In such domains, parts of a large or complex solution may be removed or abstracted while still retaining the usefulness of a case, even if the level of usefulness varies with the specific information retained.

For example, as previously mentioned, in case-based planning, certain parts of a plan could be elided or abstracted to reduce storage. When a new planning problem is precisely covered by the retained material, there is no solution quality or efficiency loss. When it is not, the maintenance may result in increased adaptation cost to reconstruct the plan, or if adaptation power is insufficient for perfect reconstruction, then some competence would be lost. However, partial deletion of case contents might still cause less competence loss than deletion of an entire case by per-case methods.

Case-based support for concept mapping (Leake et al., 2014) provides another example. *Concept maps* are informal two-dimensional visual representations of concepts and their relationships, representing the conceptualization of a domain from the perspective of a particular user (Novak et al., 1984). The goal of a support system is to aid humans using electronic tools to build concept maps. It accomplishes this by monitoring the concept map under construction, retrieving relevant past concept maps, and using those to suggest extensions. Concept map cases contain rich structures of interconnected concepts, from which deletion of some parts may reduce the range of problems for which suggestions can be provided, but for which the remaining parts can still be useful.

For supporting concept map extension, any part of a concept map case may be viewed as the index or the solution, depending on which features are available as the input problem and the context of the retrieval (Leake et al., 2003). Thus, in the concept mapping domain, the same feature deletion process can be seen as simultaneously maintaining indices and solutions.

3.2 Bundling Features for Deletion

Consider cases as composed of a set of primitive features which cannot be further decomposed. The following explanation, for simplicity, will consider these to be attribute-value pairs. However,

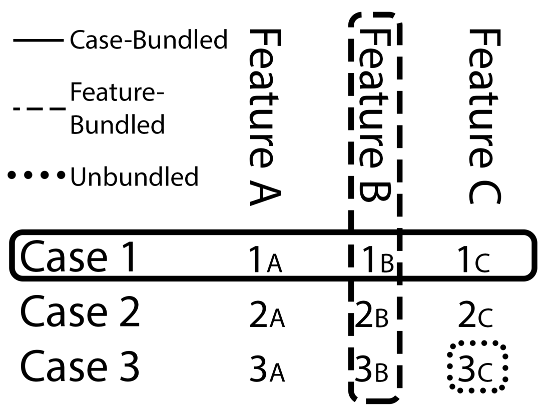


Figure 3.1: Feature selection with case-bundled, feature-bundled, and un-bundled strategies (Leake & Schack, 2015).

other representations are possible. Both indexing and solution information are defined by sets of features. For example, basic features could be combined to form complex structured cases, from which flexible feature deletion could remove multiple features corresponding to substructures.

Maintenance approaches for case-base compression can be seen as "bundling" different types of information together to treat as a unit. Conventional per-case maintenance for case-base compression bundles together all features associated with a particular case and deletes the entirety of features associated with a particular case. In contrast, feature-bundled maintenance does an orthogonal bundling, deleting a single feature in all cases for which it appears. Flexible feature deletion can also apply an "unbundled" approach, simply deleting specific features from selected individual cases. To distinguish un-bundled individual features from feature-based bundles, this chapter calls the individual features of a specific case "case-features." Figure 3.1 (on page 32) illustrates the case-bundled, feature-bundled, and un-bundled approaches.

Figure 3.2 (on page 33) summarizes eleven simple candidate strategies for selecting the next case or feature to delete, spanning case-bundled, feature-bundled, un-bundled, and hybrid strategies, which we describe in more detail below. Random deletion strategies are included as a baseline. The simplicity of these strategies enables comparing case-bundled and feature-bundled strategies

Strategy	Type of Bundling	Hybrid or Non-Hybrid
Random Case-Features	Unbundled	Non-Hybrid
Random Cases	Case-Bundled	Non-Hybrid
Large Cases	Case-Bundled	Non-Hybrid
Least Coverage	Case-Bundled	Non-Hybrid
Most Reachability	Case-Bundled	Non-Hybrid
Random Features	Feature-Bundled	Non-Hybrid
Rarest Features	Feature-Bundled	Non-Hybrid
Most Common Features	Feature-Bundled	Non-Hybrid
Largest Cases / Least Coverage	Case-Bundled	Hybrid
Rarest Features / Least Coverage	Unbundled	Hybrid
Rarest Features / Large Cases	Unbundled	Hybrid

Figure 3.2: Strategies for selecting the next case, feature, or case-feature to delete (Leake & Schack, 2015).

on an equal footing. "Section 3.4: Future Research Questions for Feature Deletion" (on page 41) discusses future paths for more sophisticated flexible feature deletion strategies.

1. **Case-Bundled Strategies**

   Case-bundled strategies follow the traditional CBR compression approach of removing entire cases, i.e., the bundle of features determined by the case. A key question for case deletion is the order in which to delete cases. A classic approach is to consider *coverage*, the set of target problems that a case can solve, and *reachability*, the set of cases that can solve a given target problem (Smyth & McKenna, 1999a). Cases with higher coverage are considered more valuable to preserve; cases with lower reachability are considered harder to replace. This chapter considers simple strategies favoring each criterion. Another simple criterion is to include removing the largest cases first (aiming to maximize size reduction).

2. **Feature-Bundled Strategies**

   Feature-bundled strategies ignore the boundaries of cases, replacing deletion of cases with deletion of common features across cases. For example, in a movie recommendation domain, one feature might be the presence of a particular (little-known) individual; if that was unimportant to recommendations, that feature could be deleted from all cases without impairing recommendation performance. This chapter considers the baseline strategy of random deletion, a strategy of removing the most common features (which might be expected to have the least information content), and an inverse strategy of removing the rarest features (which might be expected to be useful in fewer instances).

3. **Un-bundled Strategies**

   Un-bundled strategies ignore the boundaries of both cases and features. Deletion need not be done uniformly on a per-case or per-feature basis; individual features may be deleted from some cases and retained in others. For example, in the movie domain, the feature corresponding to a particular actor could be deleted only from selected cases (e.g., those in which the actor had a walk-on role). This chapter considers only one basic un-bundled strategy, removing random features of random cases.

4. **Hybrid Strategies**

   This chapter also considers three hybrid strategies, each combining two strategies with equal weight (weightings could also be tuned). The strategies are Large Cases / Least Coverage, Rare Features / Least Coverage, and Rare Features / Large Cases. Combining two case-bundled strategies, as in Large Cases / Least Coverage, yields a case-bundled strategy, and combining two feature-bundled strategies yields a feature-bundled strategy. However, combining two differently bundled strategies (e.g., Rare Features / Least Coverage) yields an un-bundled strategy in which the scores of the constituent parts are used to determine case-features to delete.

Strategies can have substantially different computational costs. Case size and feature rarity can be calculated rapidly because they do not require problem solving. However, coverage depends on the ability of a case to solve the problems associated with other cases, and so requires more costly testing involving other cases in the case base.

3.3 Evaluation

In an attempt to understand the relationship between per-case and flexible feature deletion strategies, the evaluation tested the compression / competence trade-off for the strategies in Figure 3.2 (on page 33), across three domains. The evaluation addressed the following two questions:

1. For a given level of compression, how does the retrieval accuracy of the strategies compare?
2. How does the retrieval time change as the number of case-feature pairs decreases, and does this depend on the retrieval strategy?

The author hypothesized that at higher levels of compression, accuracy would tend to decrease for all strategies, but that non-case-bundled maintenance strategies would outperform case-bundled strategies. The author also hypothesized that, as the total number of features decreases, retrieval time would decrease as well, with decreases roughly independent of the strategy used.

3.3.1 Test Data

Tests used three data sets, from movie, legal, and travel domains. Movie data was drawn from the Internet Movie Database (IMDb),[1] in which each case was a film or television show, and each feature was an actor in that film or show. The sample contained 100,000 case-feature pairs in 74,720 cases with 38,374 features.

Legal data was extracted from LegiScan[2] on the 113th session of the United States Congress. Each case was a bill, and each feature was a sponsor or co-sponsor of a bill. The sample contained 50,000 case-feature pairs in 7,785 cases with 552 features.

[1]http://www.imdb.com/interfaces

[2]https://legiscan.com/

Travel data was taken from the travel package case base on the Case-Based Reasoning Wiki.[3] Each case was a travel package, and the features were the types, prices, regions, etc. (represented as key-value pairs). This case base contained 14,700 key-value pairs in 1,470 cases, with 2,902 distinct key-value pairs.

All features for the IMDb and law domains were Boolean; features corresponded to the presence of a particular actor in a film or sponsor of a bill. The features for the travel domain were key-value pairs, which were treated as Boolean features based on whether a particular pair was present.

3.3.2 Indexing and Similarity Criteria

In the experiments, when features were deleted from case content, the corresponding indices were deleted as well, keeping indices and case content synchronized. Case similarity was calculated by Jaccard similarity (Niwattanakul et al., 2013) on sets of case-features. For calculating competence, problems were considered to be solved successfully if the system was able to retrieve a case for which the Jaccard similarity of case-features exceeded 50%. Additional tests were run for a scenario assuming minimal shared coverage, in which cases were considered to cover only the closest adjacent case in the original case base. Therefore, successful retrieval was defined as the system retrieving the same case retrieved during the initial leave-one-out testing. Results were similar under both conditions. For reasons of space, this chapter reports only the results for traditional similarity.

3.3.3 Hybrid Strategies

The hybrid strategies in the experiments rank case-feature pairs by summing normalized scores corresponding to each of their constituent strategies. The score assigned to a case for Large Cases is the size of that case divided by the size of the largest case in the case base. The coverage score assigned to a case for Least Coverage is the coverage of the case divided by the maximal case coverage. The score for Rare Features is based on the commonality of the feature, defined as the

[3] http://cbrwiki.fdi.ucm.es/mediawiki/index.php/Case_Bases

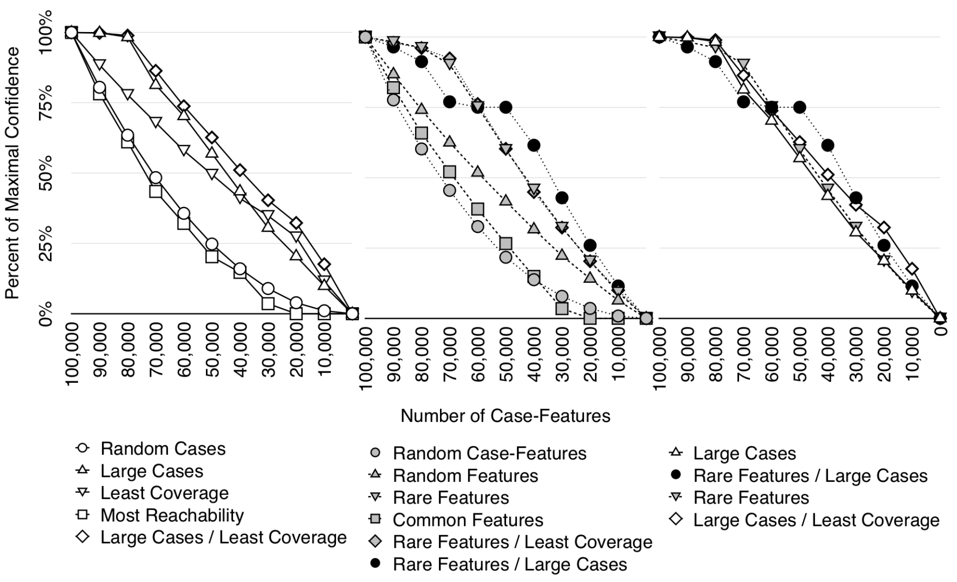


Figure 3.3: Competence retention for varying compression levels on the IMDb case base (Leake & Schack, 2015).

number of cases that contain that feature divided by the number of cases containing the maximally common feature in the case base; rarity of a feature f is $1 - \mathit{commonality}(f)$.

3.3.4 Evaluation Procedure

The evaluation first establishes baseline performance by leave-one-out testing for the entire case base. Next, performance is tested for compression to nine different case base sizes, ranging from 90% to 10% of the case base. For each test, the entire original case base is used as test problems, and a test problem is considered solved if there exists in the compressed case base a case (other than the test case) within the 50% similarity threshold.

When compressing the case bases, if the desired number of case-feature pairs does not fall exactly on a boundary between cases, then the single case in which this division falls is un-bundled to delete features within a case.

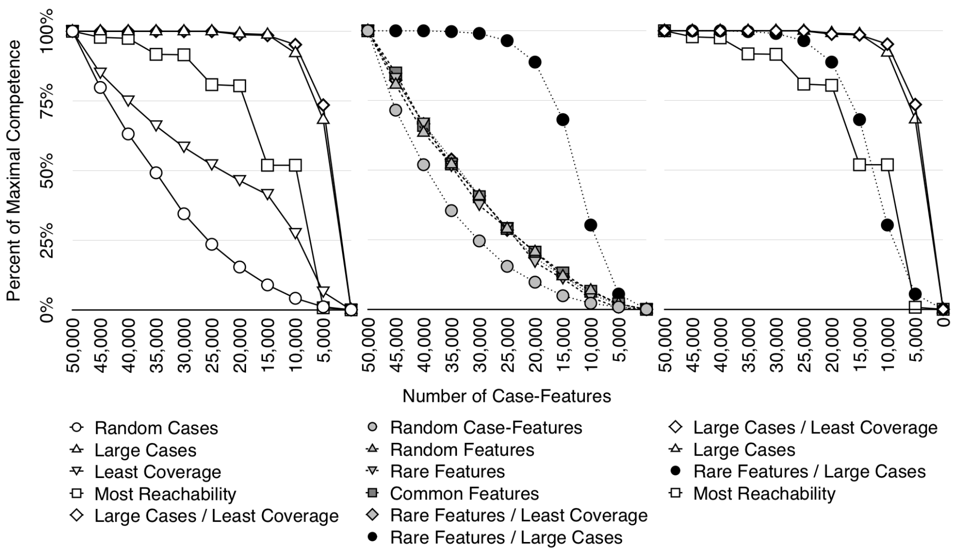


Figure 3.4: Competence retention for varying compression levels on the law case base (Leake & Schack, 2015).

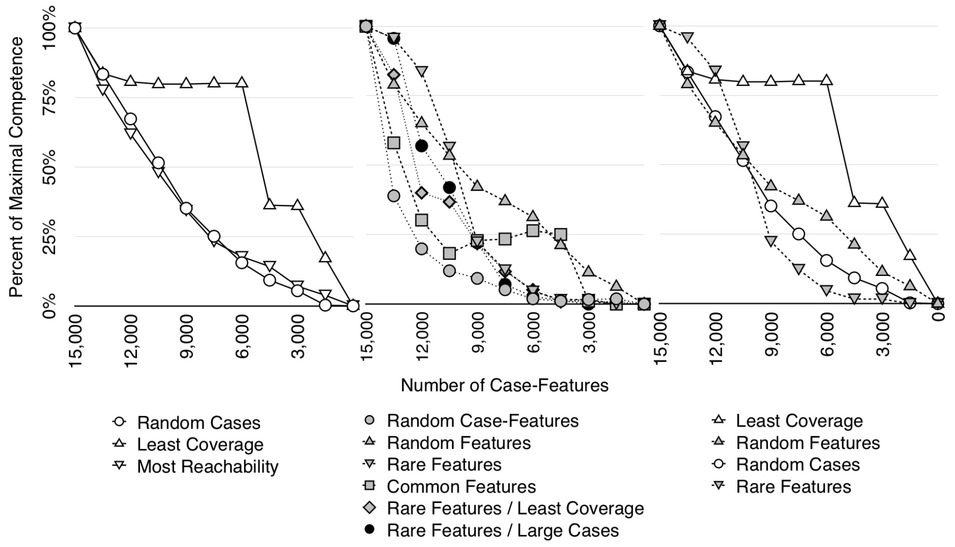


Figure 3.5: Competence retention for varying compression levels on the travel case base (Leake & Schack, 2015).

3.3.5 Experimental Results

Figures 3.3 - 3.5 (on pages 37 - 38) show accuracy after each round of maintenance. The graphs compare the eleven strategies across the movie, law, and travel domains. For readability, the graphs are divided into three parts with the same horizontal and vertical scales. The third graph compares the best four strategies from the other two graphs. Each type of bundling has a different type of connecting line. Solid lines indicate case-bundled strategies, dashed lines indicate feature-bundled strategies, and dotted lines indicate un-bundled strategies.

Figure 3.3 (on page 37) shows results for the IMDb data, for which the best four strategies were Large Cases, Rare Features / Large Cases, Rare Features, and Large Cases / Least Coverage. Three of the best strategies consider the size of the cases, which supports having maintenance consider not only the benefit of retaining a case (its solution coverage) but also its storage cost. Two of the best strategies are hybrid strategies, and two are non-case-bundled. The worst strategy was Most Reachability.

Given the established importance of coverage, that Least Coverage is outperformed by Large Cases on the IMDb and law data sets might seem surprising, but this is explained by the substantial case size variation in these domains. For example, the IMDb case base includes multi-episode soap operas such as *The Bill*, which span hundreds of actors but also include numerous relatively unknown actors who never appear widely.

Figure 3.4 (on page 38) shows results on the law data, for which the best four strategies were Large Cases / Least Coverage, Large Cases, Rare Features / Large Cases, and Most Reachability. These overlap with three of the best strategies on the IMDb case base, but in a different order. The worst strategy was Random Case-Features. The law data set has a much smaller number of features than the IMDb data set, therefore the author speculates that its features are more likely to have comparable importance, making random deletion more likely to remove significant content.

Figure 3.5 (on page 38) shows results for the travel data. Because all cases are initially the same

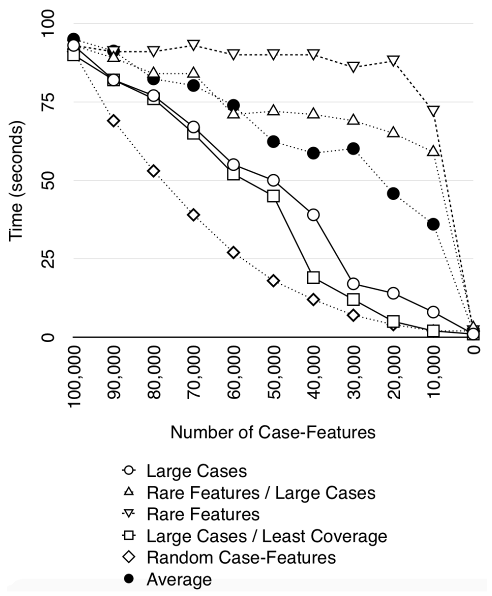


Figure 3.6: Comparison of the retrieval times after each round of maintenance between the four best strategies on the cinema data set (Leake & Schack, 2015).

size, the strategies Large Cases and Large Cases / Least Coverage do not apply and are omitted from the graphs. However, the hybrid strategy Rare Features / Large Cases is still applicable because, as the Rare Features strategy deletes features, only cases with those features will be compacted, resulting in different case sizes. The best strategies were Least Coverage, Random Features, Random Cases, and Rare Features. That deleting cases with least coverage is best is consistent with the key role coverage has been ascribed in case-base maintenance research. That Random Features is second is surprising but could be explained if many features in this domain have comparatively low information content. Although Rare Features is one of the top four strategies, its performance is quite poor, which could correspond to rare features tending to be important for distinguishing relevant cases. As with the other two data sets, two of the best strategies were non-case-bundled. However, in contrast, none of the best strategies were hybrid.

3.3.6 Retrieval Speed

Figure 3.6 (on page 40) compares the retrieval times after each round of maintenance for each of the four best strategies for the IMDb case base, for retrieval from a MySQL database. It also includes Random Case-Features as a baseline. The Average line shows the mean retrieval time of the five strategies in the graph. All tests were run on a MacBook Pro with a 2.5 GHz Intel Core i5 processor and 8 GB of RAM.

Random Case-Features, the baseline, gave the best retrieval times, and Rare Features, the only feature-bundled strategy, gave the worst. Both of the case-bundled strategies, Large Cases and Large Cases / Least Coverage yield similar retrieval times, but the two un-bundled strategies, Rare Features / Large Cases and Random Case-Features, yield very different retrieval times. Most of the strategies have a fairly linear decline, but Rare Features declines slowly until the 10,000 case-feature mark where it drops abruptly. Because the retrieval function uses Jaccard similarity, retrieval time depends on the number of case features in the intersection between cases. However, the rarest features would seldom fall into any intersections, which explains why removing them has the least effect on retrieval time.

3.4 Future Research Questions

The feature deletion approach raises a rich range of questions for fully exploiting its potential. A key question is how to develop knowledge-based feature deletion rules, especially for flexible feature deletion for complex structured cases. Other questions include how feature deletion strategies should interact with the indexing and adaptation knowledge containers, how feature deletion can preserve case integrity, and how feature deletion should be reflected in case provenance and explanation.

- *Coupling feature deletion with index maintenance:* As case contents are deleted, the relevance of case indices may change. Consequently, feature deletion may need to be accompanied by index maintenance to assure that as cases are compressed the system still retrieves the most similar cases. Feature weight information might be used to suggest features which could be deleted with limited harm.

- *Benefiting from the relationship of feature deletion to case adaptation:* Feature deletion can be seen as a form of "before the fact" adaptation of cases, in which the adaptation is driven not by a new problem to solve, but by a combination of (1) compression goals, and (2) performance goals. Richer feature deletion methods could draw on the adaptation knowledge of a case-based reasoning system to perform operations beyond simple deletion of case components, such as abstractions or substitutions of alternatives requiring less space. Enabling such methods requires reasoning about the competence effects of replacing a case with various candidate adapted versions, as well as performance effects (whether replacing a case with a given compressed version will decrease problem-solving speed), and the balance to strike between them.
- *Maintaining case integrity despite feature deletion:* Another question is the relationship of feature deletion to the cohesiveness of a case. From the early days, an argument for case-based reasoning has been that cases can implicitly capture interactions among case parts. Deleting portions of a case risks some of that cohesion, making it a concern to address in feature deletion strategies. That case adaptation faces the same risks but is effective supports optimism for some levels of compression, and research on hierarchical case-based reasoning has supported the usefulness of sometimes considering sub-parts of complete cases individually. However, how much compression can be done without excessive harm to case integrity, and how to manage the process to avoid such harm, are interesting questions.
- *Reflecting feature deletion in provenance and explanation:* Because feature deletion results in stored cases which differ from the cases originally captured, it (like case adaptation) may weaken the ability to justify proposed solutions by past experience. Likewise, changes from the

original cases may make it difficult to apply provenance-based methods for predicting solution characteristics such as solution accuracy and trust (Leake & Whitehead, 2007). Addressing these complications might require maintaining records of the case maintenance process as part of the provenance trace used for explanation, as well as reasoning about (and presenting to users) information about the parts of the case which have been affected by feature maintenance.

3.5 Summary

This chapter proposed a new case-base maintenance approach, flexible feature deletion, which questions the assumptions that cases are of uniform size and that maintenance must treat cases as unitary objects. Flexible feature deletion selectively deletes the contents of cases rather than restricting deletion to whole cases.

This chapter illustrated tasks for which flexible feature deletion may be desirable, such as domains in which reasoning can be done with different amounts of information and in which flexible feature deletion can selectively compress different parts of different cases. Experimental results show that case-base maintenance needs to consider when case contents are non-uniform; in such contexts, feature-based strategies may give better accuracy than per-case strategies. And experimental results also show that case base size and retrieval times may not always align, giving a space / time trade-off which may be exploited.

This chapter focused primarily on knowledge-light maintenance strategies. Interesting future directions are to refine the strategies tested with additional knowledge, for example, leveraging case adaptation knowledge, and to explore when other knowledge-light techniques for compression of cases and feature bundlings could yield useful maintenance strategies.

Chapter 4

Prioritizing Deletion by Recoverability via Adaptation Knowledge

As described in the previous chapter ("Chapter 3: Subdividing Cases for Deletion" on page 26), most research on case-base maintenance has focused on retention decisions at the case level, aimed at guiding case retention or deletion decisions based on the overall competence contributions of the cases. Per-case strategies are appropriate for the task domains to which they have been applied, which generally share two characteristics: (1) that cases are of fairly uniform size, and (2) that preserving the usefulness of a case depends on retaining its entire contents. However, these assumptions do not always hold and, in some circumstances, it may be useful to apply a finer-grained approach, focusing on compacting the contents of cases themselves by selectively deleting components. *Flexible feature deletion (FFD)* generalizes per-case maintenance by dropping the assumptions of uniform case size and indivisible cases (Leake & Schack, 2015; Schack & Summers, 2017).

Flexible feature deletion applies when information can be removed from a case while retaining some usefulness. For example, for cases capturing medical images, it may be possible to compact cases while retaining usefulness by adjusting resolution; for traces in trace-based reasoning (Cordier et al., 2013) or plans in case-based planning, it may be possible to compact while retaining usefulness by deleting routine portions of the steps in a case that are easily regenerated; for large and rich cases capturing recommendation information (e.g., movie recommendations), it may be possible to compact while retaining usefulness by selectively deleting features likely to hold less interest. In each of these instances, some information is lost — just as information is lost when deleting entire cases. However, the results showed that for suitable case bases, the flexible feature deletion approach provided better competence retention for a given case base size than conventional per-case deletion approaches (Leake & Schack, 2015; Schack & Summers, 2017).

The key question for flexible feature deletion is how to determine which features to remove. Initial tests of flexible feature deletion selected deletion targets by simple knowledge-light methods based on statistical feature properties. The tests demonstrated that for cases with varying sizes and for which not all information was essential to case usefulness, even such simple approaches can be sufficient to provide improved competence retention. However, a natural question is how to integrate richer knowledge into flexible feature deletion.

Because flexible feature deletion revises internal case contents, its operations can be viewed as performing a form of case adaptation, though with the goal of reducing case size rather than of solving a particular problem. The competence loss from flexible feature deletion can be mitigated if adaptation knowledge can recover the original case from the changed case. Consequently, this chapter proposes *adaptation-guided feature deletion (AGFD)* which uses adaptation knowledge to guide flexible feature deletion by focusing deletion on case components that can be recovered by adaptation (Leake & Schack, 2016; Schack, 2016). From the perspective of Richter's CBR "knowledge containers" (Richter, 2003), this approach aims to delete case knowledge overlapping with knowledge contained in the adaptation knowledge container. The perspective of storing and recovering partial cases can also be seen as related to reconstructive models such as dynamic memory theory (Schank, 1982) and constructive similarity assessment (Leake, 1992).

This chapter begins with a discussion of potential roles for adaptation in flexible feature deletion. It next describes a sample domain and case study of the use of adaptation knowledge to guide choices during flexible feature deletion and presents an evaluation of the approach. As expected, the evaluation shows that adding recoverability considerations can enable flexible feature deletion to improve competence retention for given levels of compression. It also shows, surprisingly, that in some situations, case-base compression by flexible feature deletion may actually improve case-base competence, a phenomenon this dissertation refers to as *creative destruction.*

4.1 Building on Flexible Feature Deletion

Flexible feature deletion removes components of cases. For flat feature representations, its function may be as simple as deleting particular features from a feature vector. However, flexible feature deletion for structured cases may include a wider space of possible operations, not restricted to deleting individual features, or even limited to deletion per se. For example, flexible feature deletion could include compressing cases through:

Substructure deletion: Removes components of any size, ranging from individual features to larger feature collections such as sub-plans of a plan.

Substructure substitution: Replaces components with more compact components.

Substructure abstraction: A knowledge-guided form of substructure substitution. Rather than deleting a substructure entirely, it replaces the substructure with a more compact abstraction. For example, in case-based image recognition, abstraction could be applied to decrease the resolution of some or all of the image, saving space. To reuse a case, it may sometimes be necessary to do an inverse adaptation to replace the abstraction with a more specific instantiation.

Flexible feature deletion may result in storing incomplete or un-elaborated cases. For example, if a case records a path between points A and B, and some internal segments of the path are deleted, then the case for the path would no longer be intact. However, such a deletion could still be allowed, with an annotation recording the gap. In that situation, the case would require adaptation before use to recover from the deletion.

Flexible feature deletion need not preserve the original usefulness of a case, or may transform its usefulness, making it less useful for the original problem but more useful for a different problem. For example, after substructure deletion, the case might require adaptation to solve the original problem but might no longer require the adaptation it once required to solve some different problem.

For example, for a case containing a route plan, flexible feature deletion might delete some initial segments. In that case, the plan would no longer be directly applicable to the same starting point, but it could be directly applied to generate a shorter path with the new starting point.

4.2 Applying Case Adaptation Knowledge

All of the flexible feature deletion operators correspond to common operations for case adaptation. Any such operations can be applied successively in an adaptation chain (Fuchs et al., 2014) to provide varying levels of adaptation-based compression. If procedures for these are already available in the adaptation component, then they can be applied directly for flexible feature deletion. If the adaptation knowledge includes specific guidance on applicability, or on circumstances when a particular adaptation is suitable, then the use of the adaptation knowledge for maintenance makes that guidance available to the maintenance phase.

Even when adaptation knowledge is not framed in terms of deletion, it may still be useful for compression. For example, consider a path planning system able to adapt plans to avoid roads that are closed. If the adaptation results in a route that can be described more compactly, then that adaptation contributes to compression of the case, and in principle could be applied as a flexible feature deletion operation.

Exploiting adaptation knowledge for flexible feature deletion raises the key question of when a particular adaptation should be applied. This depends crucially on four factors: compression benefit, case / feature recoverability, quality retention, and recovery cost.

Compression benefit: The reduction effected in the size of the case base. Because flexible feature deletion can change the sizes of individual cases, compression is measured not in terms of the number of cases in the case base, but in terms of finer-grained sub-units directly related to storage requirements. (For cases represented by feature vectors, a natural unit is a feature-value pair.) Compression benefit reflects the advantage to the case-based reasoning system of having a smaller case base. Often, this benefit is judged in terms of retrieval speed or overall processing cost (Smyth & Cunningham, 2005). However, this could also reflect factors such as hard limits on case base size (e.g., in a legacy or high-reliability system with limited storage) or transmission cost, if the case base will be provided to other agents.

Case / feature recoverability: The ability of the system to regenerate its knowledge state prior to compression from adaptation knowledge and the remaining case base. (Regenerating the knowledge may not require regenerating identical solutions, if multiple solutions are satisfactory, and the knowledge state might include knowledge not directly connected to competence, such as features used in indexing to increase retrieval speed.) This chapter refers to flexible feature deletion operations that are always recoverable as *lossless*; those that are not necessarily recoverable are *lossy*. Whether a particular strategy is lossless or lossy depends on the entire set of adaptations available to the system, on the length of adaptation chains allowed, and on all the cases in the case base.

Quality retention: The quality of the solutions the system is able to generate, beyond simply generating a correct solution. For example, in a path planning domain, a deletion from a path would be recoverable if the system were still able to generate *some* path between the same endpoints. Quality retention might be measured by the ratio of the costs of old and new paths.

Recovery cost: The resources required to generate a new solution to the problem. For example, in a case-based planner able to draw on a generative planner when necessary, all deletions are theoretically recoverable, but recovery by reasoning from scratch may be computationally expensive. In those instances, flexible feature deletion might be less appropriate. Likewise, in some domains, a complete case that is deleted may be unrecoverable, while internal deletions can be recovered. For example, consider a medical system whose cases are X-ray images.

If an image is deleted, then there may be no recourse other than re-taking the X-ray, at considerable expense. However, if portions are stored at lower resolution, then it may be possible to recover needed information by image processing algorithms at much lower cost.

Feature-centric recoverability is closely related to the notion of reachability — with an important difference: *Reachability* refers to the ability to adapt other cases in the case base to cover the problem addressed by a case (Smyth & Keane, 1995); *feature-centric recoverability* refers to the ability to adapt either other cases in the case base or the original case revised by flexible feature deletion to cover the competence contributions of that case (which may require adapting an internal subpart of the case) (Leake & Schack, 2016; Schack, 2016). This chapter will apply a restricted variant of recoverability, termed *local recoverability*, which is the ability of a case to solve the problem that it originally solved before applying flexible feature deletion but not necessarily all of the problems in the original competence contribution of that case (Leake & Schack, 2016; Schack, 2016). Local recoverability approximates recoverability but is more efficient to calculate.

4.3 A Case Study on Adaptation-Guided Feature Deletion

This evaluation studies recoverability-based flexible feature deletion in the context of a path planning task. The recoverability-based approach prioritizes the targets of flexible feature deletion according to local recoverability. This section describes the underlying domain and system. "Section 4.4: Evaluation" (on page 53) describes the experimental questions and results.

4.3.1 Testbed Domain

Path planning is a classic application of case-based reasoning (Anwar & Yoshida, 2001; Goel et al., 1994; Haigh & Veloso, 2005; Li et al., 2012). Experiments in mobile robot path planning observed that case-base growth is a serious problem, precluding retaining all cases (Kruusmaa & Willemson, 2003). The path planning task is carried out on a road network represented by a weighted graph

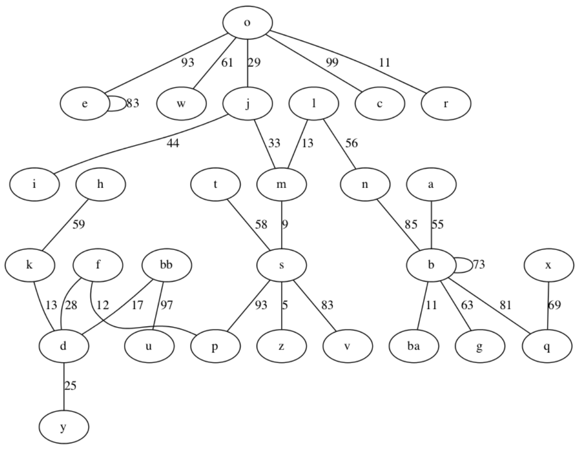


Figure 4.1: Sample graph for the path planning task (Leake & Schack, 2016).

with labeled vertices. The vertices represent neighborhoods, and they are collected into groups representing boroughs, with each group having an equal number of vertices. Each vertex in a group is intra-connected via an edge to another vertex in the same group.

The groups are also inter-connected with one or more vertices from each group having an edge to another vertex in a different group. This design represents characteristics such as streets intra-connecting neighborhoods in a borough and bridges inter-connecting boroughs in a city. To generate a road network, connections are randomly selected, with the constraint that at least one path must exist between any two vertices. Figure 4.1 (on page 50) illustrates a sample graph satisfying these constraints. Each edge has a random integer weight in [0, 100], representing the cost (e.g., distance or time) to travel across that edge. Problems to be solved by the system are described as lists of vertices which the solution path must include in order, starting with the source and ending with the destination.

4.3.2 Adaptation Strategies

The testbed system has five adaptation strategies, summarized in Figure 4.2 (on page 51). No attempt was made to optimize the adaptation process, which is done by the exhaustive application of adaptations. When adaptation must be done to generate solutions, all adaptations are tried;

Reuse Strategy	Description
Reverse	Reverse the given solution so the source swaps with the destination, the destination swaps with the source, and the intermediate points reverse.
Drop Vertices	Drop the vertices in the given solution before the source of the given problem.
Reverse Drop Vertices	Drop the vertices in the given solution after the destination of the given problem.
Cons Vertex	Append the source of the given problem to the front of the given solution.
Reverse Cons Vertex	Append the destination of the given problem to the back of the given solution.
Compose	Fill in a gap in the given solution with the solution from another case.

Figure 4.2: Testbed system adaptation strategies (Leake & Schack, 2016).

Strategy	Type of Bundling	Hybrid or Non-Hybrid
Random Case-Features	Unbundled	Non-Hybrid
Random Cases	Case-Bundled	Non-Hybrid
Large Cases	Case-Bundled	Non-Hybrid
Least Coverage	Case-Bundled	Non-Hybrid
Most Reachability	Case-Bundled	Non-Hybrid
Random Features	Feature-Bundled	Non-Hybrid
Rarest Features	Feature-Bundled	Non-Hybrid
Most Common Features	Feature-Bundled	Non-Hybrid
Rarest Cases / Least Coverage	Case-Bundled	Hybrid
Rarest Features / Least Coverage	Unbundled	Hybrid
Rarest Features / Large Cases	Unbundled	Hybrid

Figure 4.3: Strategies for selecting the next item to delete (Leake & Schack, 2015).

when the system assesses recoverability, the system attempts to adapt all cases until a solution is found. The system does not combine adaptations, except for the special case of the compose strategy, which can chain with one other strategy.

4.3.3 Flexible Feature Deletion Strategies

"Chapter 3: Subdividing Cases for Deletion" (on page 26) presented a set of knowledge-light flexible feature deletion strategies, categorized according to how they prioritize items for deletion and the type of item on which they operate (whether they delete cases, collections of features, or a mixture). These are illustrated in Figure 4.3 (on page 52). For the descriptions of the flexible feature deletion strategies, please see "Section 3.2: Bundling Features" (on page 31).

Given any flexible feature deletion strategy, it is possible to develop a recoverability-based version by guiding deletion decisions according to recoverability. Specifically in the recoverability-based flexible feature deletion approach, the system applies one of the original strategies to rank items for deletion. It then tests deletion candidates in order to determine whether the deletion is recoverable. This is done by attempting to solve the problems from the cases that would be modified, either (1) by adapting the modified cases or (2) by case-based reasoning applied to cases from the remainder of the case base. If these problems are solvable (and even if the solutions are different but satisfactory), then the modifications are accepted as recoverable. Otherwise, the process continues through the rest of the deletions according to the ordering specified by the given strategy. The process stops after the first successful recovery (in which case the modification is accepted), or after a maximum number of trials or when no untried cases remain, in which case the strategy fails and no further compression can be done.

The testbed system applies the five deletion strategies in Figure 4.4 (on page 54), selected to include high-performing flexible feature deletion strategies from "Chapter 3: Subdividing Cases for Deletion" (on page 26). These include two lossy strategies, Largest Case (which first deletes the largest cases, measured by the number of vertices in each solution) and Random Case-Feature (which deletes a randomly-selected vertex from the solution to a randomly-selected case and marks the gap for potential future recovery). Adding recoverability considerations leads to the strategies Reachability-Based Largest Case and Recoverability-Based Random Vertex. (Reachability-Based Largest Case deletes whole cases, and Recoverability-Based Random Vertex deletes components of cases.) The recoverability-based variants aim to mitigate the lossiness of the strategies by using recoverability to filter their deletion recommendations. The Shared Component strategy is lossless because it extracts a component shared by the solutions to multiple cases into a separate case before deleting the component from those solutions.

4.4 Evaluation

The evaluation focused on the following three questions:

1. **Competence retention:** How does the ability to solve problems for given levels of compression compare between adaptation-guided and non-adaptation-guided flexible feature deletion

Deletion Target	Lossiness	Description
Shared Component	Lossless	Extract components shared by the solutions of multiple cases into separate cases. Mark gaps for completion during recovery.
Reachability-Based Largest Case	Lossy	Delete cases in order from largest to smallest number of case-features, deleting only recoverable cases.
Largest Case	Lossy	Delete cases in order from largest to smallest number of case-features regardless of recoverability.
Recoverability-Based Random Vertex	Lossy	Delete randomly-chosen case-features from the solutions to cases, deleting only recoverable features.
Random Vertex	Lossy	Delete randomly-chosen case-features from the solutions to cases regardless of recoverability.

Figure 4.4: Sample deletion strategies, including recoverability-based strategies (Leake & Schack, 2016).

strategies?

2. **Solution quality retention:** For problems that can be solved for a given level of compression, how does solution quality compare between adaptation-guided and non-adaptation-guided flexible feature deletion strategies?

3. **Processing time:** How does the choice of flexible feature deletion strategy impact the processing time of the case-based reasoning cycle for different levels of compression?

4.4.1 Experimental Design

The case base was seeded with a set of training problems with randomly chosen vertices, with beginning and ending vertices chosen from different groups which are not neighbors. This ensures that solving a problem requires exiting the source group, traversing one or more other groups, and then entering the target group. The Bellman-Ford path finding algorithm (AbuSalim et al., 2020) was used to generate optimal solutions to the training problems, minimizing the sum of the weights of the edges along the path.

Experiments averaged results of 12 trials, each one using a different randomly generated graph of 28 vertices, initial case base of 33 randomly-generated seed cases and 17 test problems. Tests were run for each retention strategy in Figure 4.4 on page 54, with three-fold cross-validation, averaging the results by strategy and level of compression. Reported processing times reflect processing on a MacBook Pro with a 2.5 GHz Intel Core i5 processor and 8 GB of RAM.

For lossy strategies, compression was continued until only a single case remained. Sometimes a strategy can no longer compress a case base, either because the strategy is a lossless strategy and cannot compress beyond full competence, or because it exceeded a preset limit of 100 trials to find a recoverable deletion.

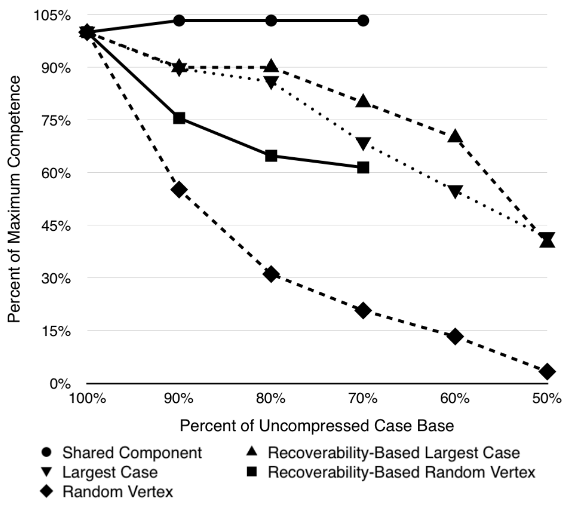


Figure 4.5: Competence retention (Leake & Schack, 2016).

4.4.2 Question 1: Competence Retention

Evaluation of competence retention measured how many problems the system could solve at increasing compression levels (decreasing numbers of case-feature pairs) for five deletion strategies: Shared Component, Recoverability-Based Largest Case, Largest Case, Recoverability-Based Random Vertex, and Random Vertex. Figure 4.5 (on page 56) shows the percent of competence retained from the uncompressed case base to the compressed case base, as a function of the percent of case-feature pairs retained from the uncompressed case base to the compressed case base, ranging from the full case base (100%) to 50% compression.

The best performing strategy for competence was Shared Component. Because it is lossless, its high performance is expected. However, its ability to compress the case base stops at 70% size when it cannot find any more shared components. To achieve more compression, one of the lossy strategies must be used.

Recoverability-based largest case does next best, and enables compression to 50%. Comparison to the non-recoverability-based version shows that the recoverability-based approach improves competence retention.

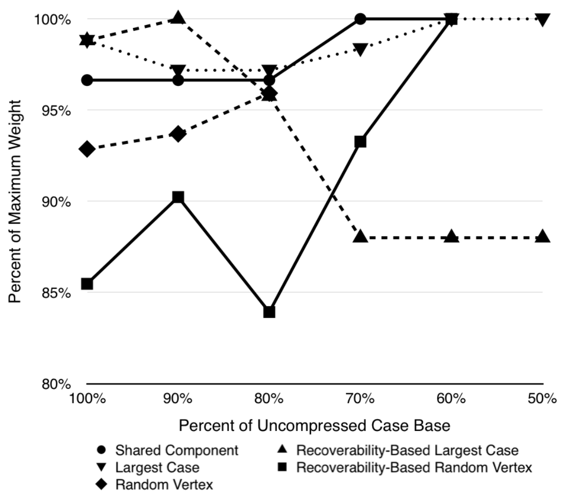


Figure 4.6: Relative average solution quality as a function of compression (Leake & Schack, 2016).

The worst-performing strategy was the simple unguided strategy Random Vertex. Considering recoverability, in the Recoverability-Based Random Vertex strategy, markedly improves competence over Random Vertex, but Recoverability-Based Random Vertex can only compress the case base to 70%, after which it can no longer find recoverable vertices.

The author expected that as compression increases, competence would remain stable or decrease. That was true for four of the five strategies. However, surprisingly, competence *increased* slightly (103% of original competence) for the Shared Component strategy at 70% of the original case base. For discussion of this phenomenon, please see "Section 4.4.5: Creative Destruction" (on page 58).

4.4.3 Question 2: Solution Quality Retention

The evaluation of solution quality retention measured the quality of the solutions by the sum of their edge weights such that lower aggregate weights were preferred. Figure 4.6 (on page 57) shows the relative average sum (percent of maximum) of the weights of the solutions generated at different levels of compression with the five retention strategies, as a function of the percent of case-feature pairs retained from the uncompressed case base. Here no strategies are clearly the best or worst. This suggests that more knowledge would be needed to reliably ensure high-quality solutions.

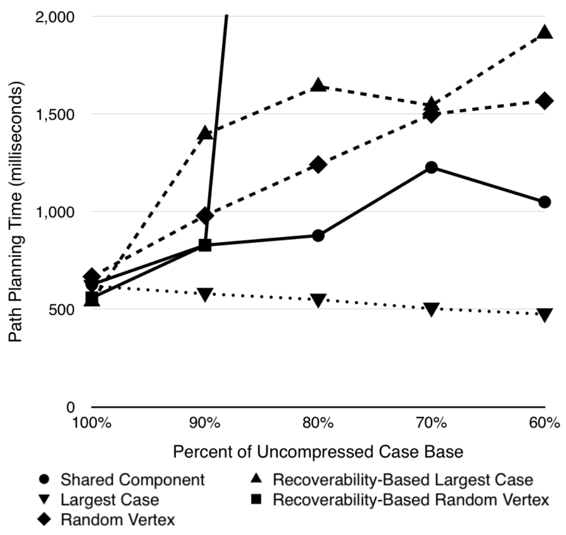


Figure 4.7: Average total case-based reasoning path planning time (Leake & Schack, 2016).

4.4.4 Question 3: Processing Time

Figure 4.7 (on page 58) shows the average total processing time for both adaptation-guided feature deletion and case-based problem-solving for the test problems at each stage of compression for each of the five retention strategies, as a function of the percent of case-feature pairs retained from the uncompressed case base. The Largest Case strategy was most efficient, and the Recoverability-Based Random Vertex strategy was least efficient, with very rapid growth, due to checking many alternatives before finding vertices to delete. The line for this strategy continues beyond the top edge of the plot; the graph is cropped in order to show the performance on the other strategies in detail.

4.4.5 Creative Destruction

In the competence retention experiment reported in "Section 4.4.2: Competence Retention" (on page 56), the author expected that competence would always decrease with increased compression, as is normally expected for any case-base compression method. Surprisingly, occasionally adaptation-guided compression applied to the case base could slightly improve the competence of

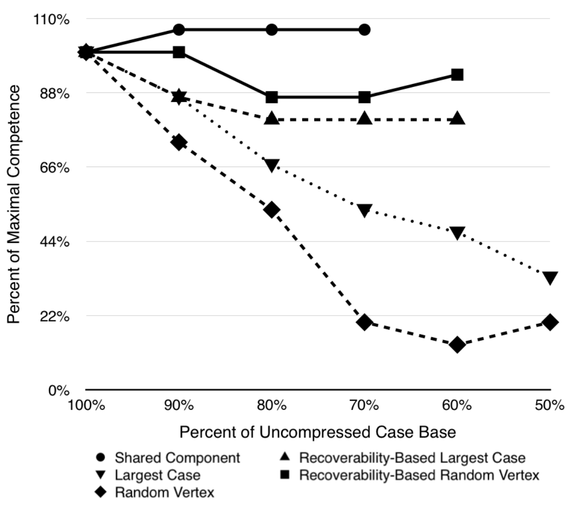


Figure 4.8: An example of the creative destruction phenomenon (Leake & Schack, 2016).

the system. The explanation is that compression strategies, by reorganizing the contents of cases, can sometimes make case contents accessible to adaptations of limited power.

For example, if the Shared Component compression strategy finds a component of a solution shared between several cases, it moves this shared component into a separate case, leaving a marker in each of the cases from which it was removed. Later, the case-based reasoning process can manipulate the shared component independently of the rest of the components of the case. Normally, the Drop Vertices reuse strategy can only remove vertices at the ends of a path, not within the path. However, after extraction, the middle of the case is “exposed” as its own case, and is therefore available to the Drop Vertices strategy.

Similarly, any retention strategy that removes a component creates a gap which could be filled by a different component which may be useful for further adaptation. Typically, the benefit of this effect is small, but as shown in Figure 4.8 (on page 59), taken from a single iteration of cross-validation, the effect can be large in some circumstances. This suggests that it could be worthwhile, when designing flexible feature deletion strategies, to consider creative destruction opportunities that they might provide. However, further study is needed to corroborate these results more generally and understand the characteristics and potential for creative destruction in different domains.

4.5 Summary

This chapter proposed a symmetry between compression and reuse for flexible feature deletion, such that the compression strategy draws on adaptation knowledge and is guided by the extent of the reversibility of deletions by adaptation. Evaluation of recoverability-based compression in a path-finding domain supported that recoverability-based methods can provide superior retention of competence compared to flexible feature deletion at the same level of compression. An interesting area for future research is exploring recoverability-based methods for richer adaptation knowledge.

A smaller case base can reduce retrieval cost, but a compressed case may require adaptation to recover an acceptable solution, so the reduced retrieval cost is balanced by a potentially increased adaptation cost. Using this trade-off to guide maintenance decisions could be an interesting topic for further study.

The experiments revealed an additional surprising result: that case-base compression (normally expected to decrease competence) can sometimes actually improve problem-solving competence. This creative destruction phenomenon suggests two interesting avenues for research. The first, in the context of case-base compression, is how to prioritize flexible feature deletion to maximize the chance of creative destruction occurring. The second arises because creative destruction, by improving competence, would be valuable even when compression is not needed. This suggests opportunities for maintenance aimed at improving competence by revising or restructuring cases to make them more amenable to adaptation, given characteristics of the adaptation knowledge of the system.

Chapter 5

Data Augmentation to Expand Choices for Deletion

Building on maintenance work by Smyth and Keane (1995) and many others (Angiulli, 2005; Brighton, 2001; D. R. Wilson & Martinez, 2000), the previous two chapters ("Chapter 3: Subdividing Cases for Deletion" on page 26 and "Chapter 4: Prioritizing Deletion by Recoverability via Adaptation Knowledge" on page 44) proposed and evaluated two approaches for controlling case base growth: flexible feature deletion and adaptation-guided feature deletion. Like their predecessors, both of these methods prioritize deletion with the goal of maximizing the competence retention of the case base.

Unfortunately, estimating the future competence contributions of a case is difficult because it depends on predicting the problems a case-based reasoning system will encounter. Smyth and McKenna (1999b) addressed this with the *representativeness assumption* which says that prior problems are representative of problems to be encountered in the future. Under this assumption, the future competence contribution of a case can be estimated as its competence contribution in the existing case base. Although the assumption may not always hold, Smyth and McKenna advanced an argument for its appropriateness in case-based reasoning systems: Because case-based reasoning is based on the assumption of problem-distribution regularity — that future problems will resemble past problems (Leake & Wilson, 1999) — domains for which the representativeness assumption fails would presumably be ill-suited for case-based reasoning.

Case-base compression relying on the representativeness assumption has been shown effective in many domains. However, in domains for which only a small part of the problem space has yet been encountered, or in which data drift shifts the problem distribution (Cunningham et al., 2003), representativeness may not hold, and in turn, competence may suffer (N. Lu et al., 2014). Likewise, if a case base generated for one task is applied to a new task for cross-domain problem-solving (Leake & Sooriamurthi, 2002), then there is no guarantee that the problems in the first space will be representative of problems in the second.

A well-known strength of case-based reasoning is that it can draw on multiple knowledge containers whose contributions overlap, in the sense that strengths in one can compensate for weaknesses in another (Richter, 2003). This chapter investigates how a case-based reasoning system can draw on adaptation knowledge to handle gaps in experience when building a case base. By adapting cases already in the case base, a case-based reasoning system can pre-populate sparsely populated regions of the case base — transferring some of the knowledge from its adaptation component into the case component to expand the set of cases. This in turn enables the system to generate the compressed case base from a set of candidates larger than its retained experience. This can be seen as shifting from maintenance that only exploits existing experiences, to maintenance that explores the space of future problems. Combining case base exploitation with problem space exploration is a novel step for case-base maintenance.

To combine exploitation and exploration for the purpose of case-base compression, this chapter proposes the expansion-contraction compression strategy. *Expansion-contraction compression (ECC)* adapts existing cases to generate additional candidate cases called *ghost cases*, providing a more diverse set of cases for compression to consider. Then the union of the case base and the ghost cases is provided to the condensed nearest neighbor algorithm in order of competence contribution. *Condensed nearest neighbor (CNN)* refers to a machine learning technique that reduces the size of a dataset while preserving its representativeness and classification accuracy by selecting a subset of the most informative and representative examples (Hart, 1968). This provides competence-based deletion with a wider set of cases from which to select that can include cases from unseen but solvable parts of the problem space. If case adaptation is considered sufficiently reliable, then the result of expansion-contraction compression can be used as is. Otherwise, the selected ghost cases can become targets for verification, e.g. by asking a human expert in an active learning process, or provenance information (Leake & Whitehead, 2007) about the origin of ghost cases could be used to predict confidence when they are used.

This chapter presents an evaluation of expansion-contraction compression for four standard data sets manipulated to enable controlled comparisons with condensed nearest neighbor for case bases with varying representativeness. The author hypothesized that expansion-contraction compression would provide better competence retention than condensed nearest neighbor as representativeness decreased, and the results supported this hypothesis. The author also hypothesized that expansion-contraction compression would not provide benefit for standard case bases and would even impose a competence penalty, due to ghost cases increasing the coverage density for non-representative problems at the expense of coverage density for representative problems. Surprisingly, expansion-contraction compression often improved competence retention even for standard case bases. This can be attributed to the addition of ghost cases providing condensed nearest neighbor with more extensive choices, enabling it to select a more effective mix of cases.

5.1 Background

5.1.1 Compressing the Case Base

A primary early motivation for case-base compression was the swamping utility problem for case-based reasoning (Francis & Ram, 1995; Smyth & Cunningham, 2005; van Someren et al., 2005). As the case base grows, case retrieval costs generally increase, while case adaptation costs tend to decrease due to the increased similarity of retrieved cases. The swamping utility problem occurs when the increased retrieval cost swamps the adaptation cost savings. Recent arguments propose that current computing resources and limited case base sizes for many tasks can make this less important in practice (Houeland & Aamodt, 2010). However, case-based reasoning for domains such as large-scale e-commerce and big data health care (Institute for Health Technology Transformation, 2013), for which customer data can measure in the hundreds of millions of cases, will continue to face challenges — necessitating either case-base compression or big data retrieval methods (Jalali & Leake, 2018). Likewise, provenance capture for e-science can result in extremely large provenance cases (Cheah et al., 2011) making case-base compression potentially important for sheer case size. In addition, controlling size can be important even before reaching extremes, if maintenance requires manual intervention, or if the case bases will be transmitted or replicated, as possible in distributed case-based reasoning.

5.1.2 Knowledge Container Transfer

The relationship of the four knowledge containers in case-based reasoning — vocabulary, similarity measure, case base, and adaptation knowledge — has given rise to much research on knowledge container transfer, such as improving adaptation knowledge by transfer from cases (Hanney & Keane, 2005; Jalali & Leake, 2013; Jalali et al., 2016; McDonnell, 2006; McSherry, 2006; Wilke et al., 1997), and from adaptation knowledge to similarity (Leake et al., 1996). Whenever a case-based problem-solver adapts a previous case and stores the result, it can be seen as transferring some of its adaptation knowledge into the case base, in a lazy manner, on demand. Expansion-contraction compression, which generates ghost cases by adapting existing cases and adding them to the case base, can be seen as performing eager knowledge transfer from the adaptation component to the case base.

5.1.3 Exploration vs. Exploitation

The notion of exploration versus exploitation concerns how an agent should allocate effort between exploiting existing resources versus exploring in search of others. The explore / exploit trade-off has proven a useful framework in many fields (Hills et al., 2014). In traditional case-base maintenance, all maintenance effort is focused on exploitation of existing cases; the expansion-contraction compression process of generating ghost cases using adaptation knowledge explores the space of potential cases. Expansion-contraction compression provides both existing cases and the fruits of exploration to condensed nearest neighbor to select those cases expected to be most valuable in the compressed case base.

Contrast between ghost case generation and adaptation on demand: Both expansion-contraction compression and normal case-based reasoning do adaptation, but the effect of the former is different from simply compressing the original case base and adapting the retained cases to solve new problems. For expansion-contraction compression, the system selects new problems to solve. This could potentially be guided by trend detection, to hypothesize areas in which additional case coverage is especially important. For example, the system could note shifts in the types of problems that the system is solving (D. C. Wilson & Leake, 2002) to focus ghost case generation there.

When a case-based reasoning system adds ghost cases to the case base, it must do so without benefit of the feedback that often enables systems to detect and repair flaws in solutions. Consequently, the solutions to ghost cases are not guaranteed to be correct. However, because the ghost case is generated in advance, there is an opportunity to seek confirmation of its solution to avoid future failure. (Even if the solution proposed by the ghost case remains unverified, and if later application of the ghost case results in an erroneous solution, the same failure would have occurred if the case had been adapted on the fly.)

Benefits of exploration: Augmenting the case base with ghost cases prior to compression has four potential benefits:

1. Potential to improve competence preservation: The author hypothesizes that for less representative case bases, expansion-contraction compression will enable increased compression for a given competence retention level. "Section 5.4: Experimental Results" (on page 72) shows the outcome of testing this hypothesis. (The author expects results to depend on the representativeness of the original case base and on the density of the original case base: If the original case base covers only a small region but all problems fall within that region, then

adding ghost cases for problems outside that region might exact a penalty as relevant cases are "crowded out" by the ghost cases.)

2. Potential to improve adaptation efficiency: If expansion-contraction compression applies a performance-based criterion for case retention that reflects not only competence but also adaptation cost (Leake & Wilson, 2003), then a case compressed via expansion-contraction compression could enable more efficient problem-solving, by adding ghost cases useful for decreasing expected adaptation cost. Where adaptation is not fully reliable but reliability can be estimated by similarity distance, such ghost cases can be chosen to reduce expected average similarity distances, and consequently, to increase expected solution reliability.
3. Potential to focus active learning / external pre-verification of adaptations: If case adaptation is unreliable, then the quality of ghost cases is not guaranteed. However, by selecting useful ghost cases, expansion-contraction compression identifies good candidates for external case acquisition / verification. For example, an expert could be asked to provide the solutions to these problems, or to review system-generated solutions (this may be easier than solution generation, e.g. verifying the routing of pipes in the design of a house is easier than finding a routing).
4. Potential to extend the reach of adaptation: Many case-based reasoning systems restrict the adaptation of any problem to a single adaptation step, limiting the problems they can solve (d'Aquin et al., 2006). When ghost cases are generated by applying an adaptation, future adaptations can start from that adapted state, effectively enabling two-step paths for adaptations going through that case. Generating ghost cases from longer adaptation paths further extends the range of problems and decreases adaptation costs.

5.2 The Expansion-Contraction Compression Algorithm

Figure 5.1 (on page 67) provides an intuitive visualization of ghost case generation. The pink rectangle represents the problem space, and the black circles represent training cases. The white

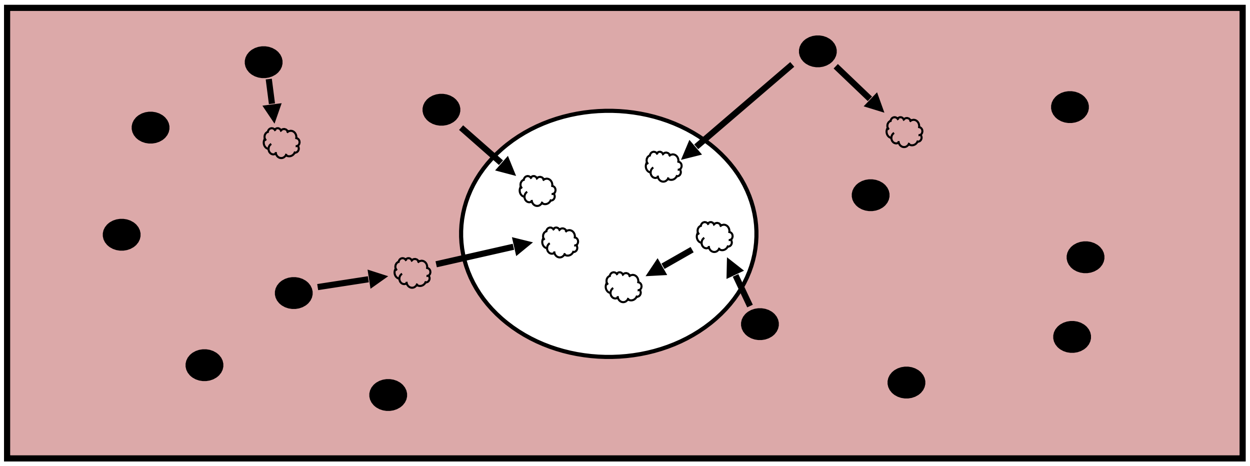


Figure 5.1: An intuitive visualization of ghost case generation.

circle shows a gap where the case base does not have training cases. These missing cases mean that the case base only partially represents the problem space. The arrows represent application of adaptation rules, and the clouds represents the ghost cases yielded. A case can adapt into zero, one, or more ghost cases. Ghost cases can occur both inside and outside of the gap in the case base. Ghost cases can further adapt (in a chain) into additional ghost cases.

Figure 5.2 (on page 68) outlines the expansion-contraction compression algorithm in pseudocode. Inputs to the algorithm include the maximum length of adaptation paths for generating ghost cases (*ghostSteps*) and the criterion for whether a case can be adapted to solve a given problem (*coverageCriterion*), which we implement as a similarity threshold. Expansion-contraction compression first expands the case base by adapting selected cases, according to an adaptation procedure which selects adaptations to perform.

Because expansion-contraction compression performs adaptations in the absence of a specific problem to solve, many strategies are possible for choosing the adaptation, e.g. selecting a random adaptation, selecting a high-confidence adaptation, selecting an adaptation expected to produce the greatest difference between old and new solutions, etc. Also, when generating a ghost case, expansion-contraction compression must adjust not only the solution, but also the problem descrip-

Input:

caseBase: the case base to compress

ghostSteps: the maximum number of adaptation steps for generating ghost cases from the case base

targetSize: the target number of cases in the compressed case base

coverageCriterion: the test for whether a case can be adapted to solve another case

Output: the compressed case base

```
$expandedCaseBase \leftarrow caseBase$
$ghostCases \leftarrow \emptyset$
for $ghostLevel = 0$ to $ghostSteps$ do
    for all $case$ in $expandedCaseBase$ do
        $ghostCases \leftarrow ghostCases \cup adapt(case)$
    end for
    $expandedCaseBase \leftarrow expandedCaseBase \cup ghostCases$
end for
$expandedCaseBase \leftarrow sort(expandedCaseBase, coverageCriterion, descending)$
$contractedCaseBase \leftarrow cnn(expandedCaseBase, targetSize, coverageCriterion)$
$additionalCases \leftarrow limit(expandedCaseBase - contractedCaseBase, targetSize -$
    $size(contractedCaseBase))$
$contractedCaseBase \leftarrow contractedCaseBase \cup additionalCases$
return $contractedCaseBase$
```

Figure 5.2: An outline of the expansion-contraction compression algorithm in pseudocode (Leake & Schack, 2018).

tion of the case, to keep the new problem and solution consistent. The adjustment is derived from the problem description part of the chosen adaptation rule — if the rule normally addresses a given difference D between an input problem and a retrieved case, then the problem part of the ghost case should be the problem of the current case, adjusted by D.

After generating ghost cases, expansion-contraction compression then compresses the expanded case base by condensed nearest neighbor, presenting cases in order of decreasing coverage (other ordering criteria, such as relative coverage (Smyth & McKenna, 1999b), could be used as well). If the resulting case base size is below the size limit, then expansion-contraction compression "fills out" the additional capacity by adding cases up to the size limit, prioritized by estimated competence contribution.

5.3 Evaluation

5.3.1 Experimental Questions

The evaluation of expansion-contraction compression considered five questions. The evaluation of the first four questions used standard domains without associated adaptation knowledge, and therefore modeled adaptation based on similarity. The experiments considered the effects of compression on both competence (measured by number of test problems solved) and solution quality (average similarity between problems and the cases retrieved for them):

1. How does expansion-contraction compression affect preservation of quality compared to conventional competence-based compression, for varying levels of representativeness?
2. How does expansion-contraction compression affect preservation of competence compared to conventional competence-based compression, for varying levels of representativeness?
3. How does the number of steps in the adaptation path to generate ghost cases affect competence retention for expansion-contraction compression, and how does competence compare to condensed nearest neighbor?

4. How does sparsity of the initial case base affect the relative preservation of competence for expansion-contraction compression and condensed nearest neighbor?

The evaluation then tested expansion-contraction compression in a standard domain augmented with automatically generated adaptation rules, to examine:

5. How does expansion-contraction compression affect preservation of competence and quality when applying generated adaptation rules?

5.3.2 Experimental Design

Data Sets: The evaluation used five data sets: Houses, with 781 cases and 8 features, from the Datasets Wiki of the California Polytechnic University Computer Science Department,[1] and four from the UCI Machine Learning Repository:[2] Iris, with 150 cases and 5 features, Wine, with 178 cases and 14 features, Car Evaluation, with 1,728 cases and 7 features, and Wine Quality, with 1,599 cases and 12 features.

Generating the Case Base and Problem Stream: Each trial partitioned each case base into three random subsets of equal size: (1) training cases — 33%, (2) testing cases — 33%, and (3) potential ghost cases — 33%. Because of the random selection of the subsets, initially the training cases have normal representativeness for the data (the effectiveness of standard competence-based compression suggests that these are reasonably representative). To test the effect of decreased representativeness, one of the experimental conditions modifies the case bases to place a gap in a region of the case base (as might exist, for example, if cases reflected seasonally varying outcomes and no problems had yet been encountered for a particular season). The gap generation process picks a random case as the starting point for the gap, and then removes all of the problems from the training case base within a given similarity threshold (the gap radius). The testing cases and potential ghost cases remain in their original distribution, without an added gap.

[1]https://wiki.csc.calpoly.edu/datasets/wiki/Houses

[2]http://archive.ics.uci.edu/ml

Modeling Adaptation and Generating Ghost Cases: None of the data sets include adaptation knowledge. The first four experiments simulate adaptation-driven generation of ghost cases as follows: The experiment repeatedly chooses a random case in the training data and filters the set of potential ghost cases for cases within a similarity threshold (the coverage criterion) required for adaptability from the selected training case. Those cases are then treated as the result of an adaptation and stored as ghost cases. This simulates deriving new cases by applying adaptation rules of limited power to the training case base. To test the effects of more powerful adaptation, additional experiments apply this process recursively, with selection of sequences of successive cases to simulate applying chains of one, two, or three adaptation steps.

Generating Adaptation Rules: The fifth experiment applied the case difference heuristic approach (Hanney & Keane, 2005) to generate a set of adaptation rules from the data. These rules were then used to generate ghost cases by adaptation, rather than drawing on potential ghost cases as in Experimental Questions 1 - 4.

The rule generation approach repeatedly selects two random cases, ascribes the difference in their solutions to the difference in their problems, and forms a rule to apply the same difference to a solution. The rule is applied when the input problem and the problem of the retrieved case have a difference similar to the one from which the rule was generated. Given a problem and retrieved case, the single rule for the most similar difference was applied. For similarity, categorical features in problem descriptions were only considered to match if identical. The rules adjusted the solution values by the proportional difference of the solution values of the cases from which they were generated.

Experimental Procedure: Compression by condensed nearest neighbor (Hart, 1968) was compared to compression by expansion-contraction compression. For each iteration of compression, both condensed nearest neighbor and expansion-contraction compression yield a case base match-

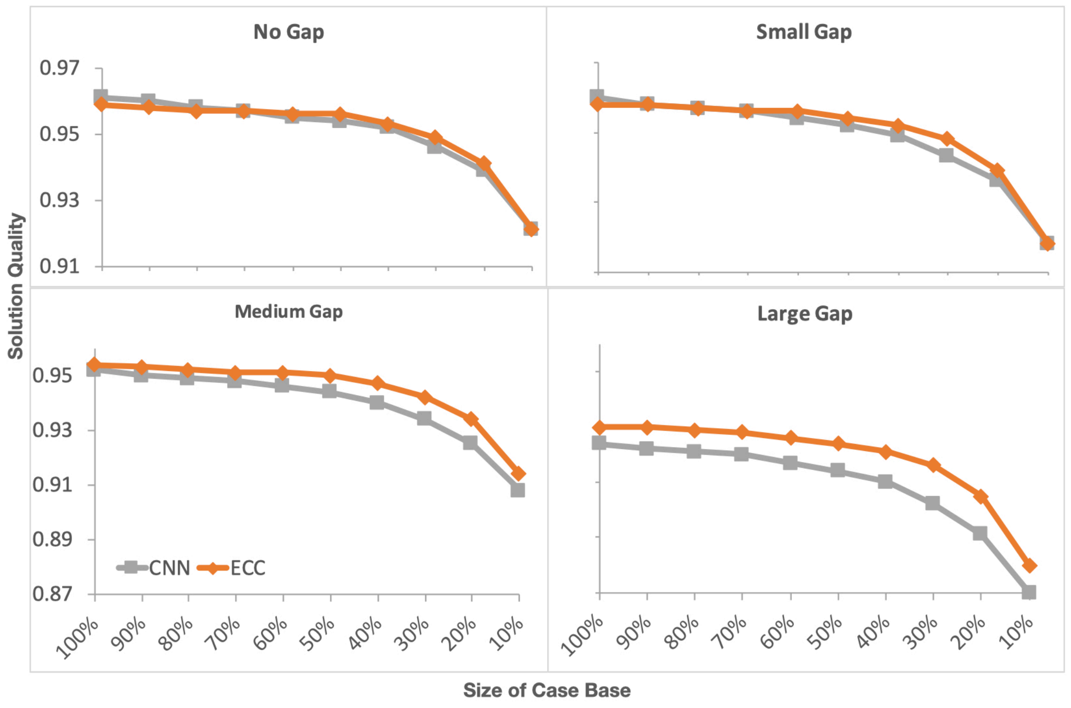


Figure 5.3: Absolute quality for condensed nearest neighbor (CNN) and expansion-contraction compression (ECC) on the Houses data set, for four different sizes of gaps in the training data (Leake & Schack, 2018).

ing the size limit, so that problem-solving has access to the same number of cases.

For both expansion-contraction compression and condensed nearest neighbor, cases were sorted in descending order of coverage. Compression was done in steps of 10% from 100% to 10% of the size of the uncompressed case base. Each level of compression starts from the uncompressed case base (not the result of the previous level of compression). Each experiment runs for ten trials with different randomly chosen partitions, with results averaged over those runs.

5.4 Experimental Results

5.4.1 Question 1: Relative Preservation of Quality

Figure 5.3 (on page 72) compares the absolute quality between condensed nearest neighbor and expansion-contraction compression strategies for the Houses case base. The coverage criterion for deriving ghost cases and solving testing problems is 5% — meaning that the difference (calculated according to the similarity metric) between pairs of cases must be less than 5% of the maximum difference. Each of the four graphs in Figure 5.3 uses a different value for the gap radius. When there is no gap, this value is 0%, a small gap is 5%, a medium gap is 10%, and a large gap is 20%. The size of the gap is measured not in the number of cases that the experiment can remove but in the similarity distance to the most different case that the experiment can remove. The horizontal axis shows the proportionate size of the compressed case base, ranging from 100% to 10% in steps of 10%.

In all four graphs of Figure 5.3, expansion-contraction compression uses adaptation paths with a maximum length of two steps. The vertical axis shows the quality of the compressed case base. Note that similarity of a retrieved case counts towards the average quality even when the coverage criterion is not met. Quality can fall anywhere in the range from 0 to 1, inclusive. The top and bottom graphs use different scales for the vertical axis (0.91 to 0.97 on the top, and 0.87 to 0.95 on the bottom) in order to "zoom in" on the difference between the strategies.

When the training case base has no gap (top left), condensed nearest neighbor outperforms expansion-contraction compression from 100% to 80% of the size of the original case base. However, from 70% to 10% size, expansion-contraction compression leads. Because neither the set of training cases nor the set of testing problems has a gap, the training case base is expected to be approximately representative of the testing problems. Therefore, in this graph, the author believes that the addition of ghost cases cannot be improving quality by improving representativeness, and must be doing so simply by providing a wider pool of cases from which condensed nearest neighbor can select alternatives. This was a small effect, but the benefit of the increased choice is an interesting result that contradicted the initial hypothesis. The results for Question 2 (in "Section 5.4.2: Relative Preservation of Competence" on page 75) show a similar, but more dramatic, effect on competence.

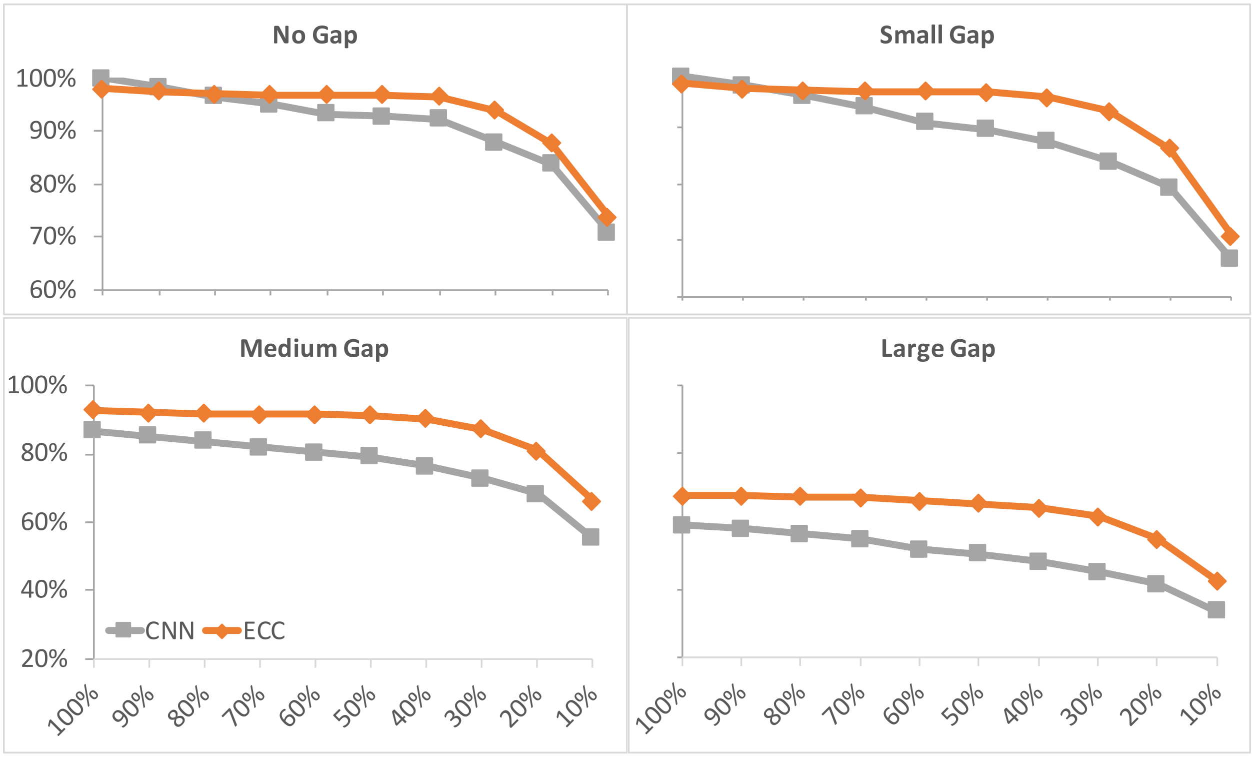


Figure 5.4: Relative competence for condensed nearest neighbor (CNN) and expansion-contraction compression (ECC) on the Houses data set with four different representativeness levels (Leake & Schack, 2018).

When the training case base has a small gap (top right), condensed nearest neighbor outperforms expansion-contraction compression only at 100% of the size of the original case base. Thereafter, from 90% to 10%, the expansion-contraction compression strategy leads. For the medium (bottom left) and large gap (bottom right), expansion-contraction compression dominates condensed nearest neighbor throughout. Overall, as the size of the gap increases, and the training case base less represents the testing problems, the quality difference increases between condensed nearest neighbor and expansion-contraction compression. This suggests that part of the benefit comes from the choice of additional cases for retention (as with the no-gap graph in the top left), and that part of the benefit comes from correction for unrepresentativeness.

5.4.2 Question 2: Relative Preservation of Competence

Figure 5.4 (on page 74) compares the relative competence between condensed nearest neighbor and expansion-contraction compression. The results are based on using the Houses case base and a coverage criterion of 5% both for deriving ghost cases and for solving testing problems. As for Figure 5.3 (on page 72), Figure 5.4 includes four graphs showing different sizes for the gap in the training case base, and the horizontal axis shows the size of the case base. However, in Figure 5.4, the vertical axis shows the relative competence, the ratio of the observed competence to the maximum competence observed with any strategy and any size of case base. (Intuitively, the maximum competence might be expected to be at the maximum size, but this need not always hold). The top and bottom graphs use different scales for the vertical axis (60% to 100% on the top, and 20% to 100% on the bottom) in order to “zoom in” on the difference between the strategies.

As shown in the no gap (top left) and small gap (top right) graphs, condensed nearest neighbor outperforms expansion-contraction compression at 100% and 90% of the size of the original case base. Thereafter, from 80% to 10%, expansion-contraction compression leads over condensed nearest neighbor. For the medium gap (bottom left) and large gap (bottom right) graphs, expansion-contraction compression dominates condensed nearest neighbor throughout all sizes of case bases. The amount of the difference between the strategies increases as the size of the gap increases.

5.4.3 Question 3: Effect of the Length of the Adaptation Path

Figure 5.5 (on page 76) presents the relative competence using condensed nearest neighbor and expansion-contraction compression for the Houses case base. The gap radius for this figure is 20% (a large gap). The coverage criterion for deriving ghost cases and solving testing problems is 5%, i.e., simulated adaptations can adapt solutions to address problem differences of up to 5%. Therefore, an adaptation path could require up to four steps to cross the gap. The horizontal axis shows the relative size of the case base decreasing from 100% to 10% in steps of 10%. Each of the lines shows

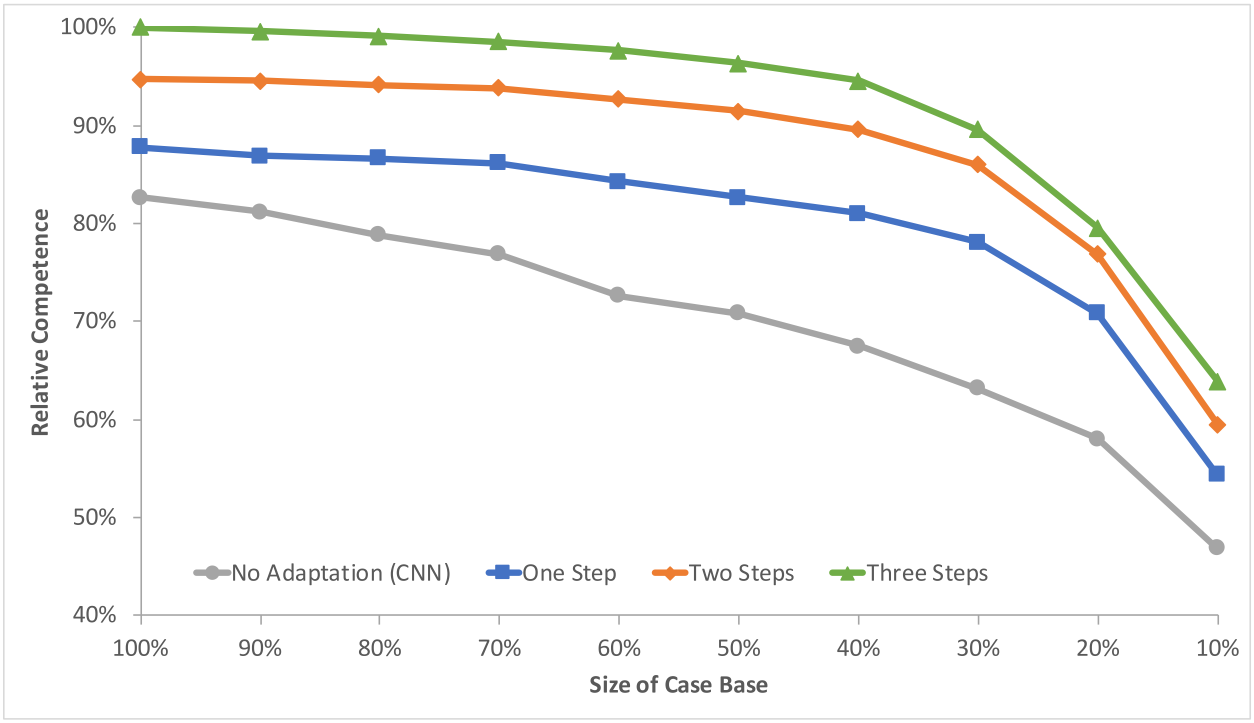


Figure 5.5: Relative competence for condensed nearest neighbor (CNN) and expansion-contraction compression (ECC) on the Houses data set with three different lengths of adaptation paths (Leake & Schack, 2018).

a different upper limit to the number of adaptations in the adaptation path to derive ghost cases. No adaptation, which is equivalent to condensed nearest neighbor, is the baseline strategy.

The vertical axis shows the relative competence of the compressed case base. For all adaptation path lengths and case base sizes, expansion-contraction compression dominates condensed nearest neighbor. The largest difference in relative competence between condensed nearest neighbor and expansion-contraction compression with one step of adaptation is 15% at 30% of the size of the original case base. The smallest difference between these two strategies is 5.1% at 100% size. The relative competence for each adaptation path length varies consistently, with longer paths associated with greater competence retention. The largest difference in relative competence between condensed nearest neighbor and expansion-contraction compression with three steps of adaptation is 27% at 40% of the size of the original case base. The smallest difference is 17% at 10% size.

5.4.4 Question 4: Effect of the Sparsity of the Case Base

Figure 5.6 (on page 78) compares the relative competence between condensed nearest neighbor and expansion-contraction compression with a sparse case base; the sparse case base models a case base in the early phases of case base growth. The basic presentation of results follows Figure 5.4 (on page 74), but the underlying experiment uses different proportions for the partitions for training, testing, and ghost cases. As described in "Section 5.3.2: Experimental Design" (on page 70), the partitions for Figures 5.3, 5.4, and 5.5 are equal thirds, but for Figure 5.6, the proportions are 10% training, 70% testing, and 20% potential ghost cases. The competence trend resembles Figure 5.4 but decreases more steeply because the case base begins with fewer cases.

Figure 5.7 (on page 78) shows the relative competence difference between expansion-contraction compression and condensed nearest neighbor on five different case bases, each modified by adding a medium gap in the training data, making it unrepresentative. For all five case bases, the gap radius is set to twice the the coverage criterion. (For the Houses data set, this corresponds to

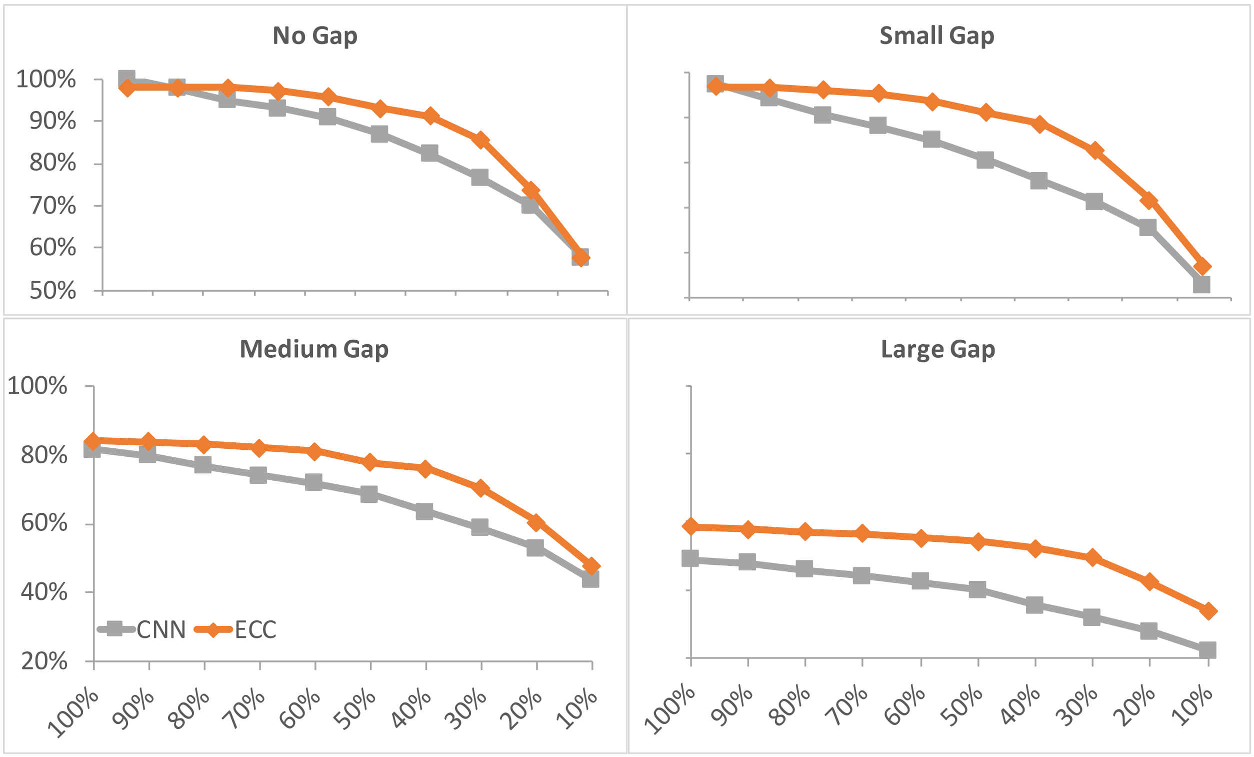


Figure 5.6: Relative competence for condensed nearest neighbor (CNN) and expansion-contraction compression (ECC) on the Houses data set with a sparse initial case base and four different levels of representativeness (Leake & Schack, 2018).

Case Base	No Compression	33% Compression	67% Compression
Houses	6.6%	11.2%	15.8%
Iris	1.2%	7.6%	7.0%
Wine	2.8%	9.6%	13.4%
Car Evaluation	-3.5%	0.0%	-8.0%
Wine Quality (Red)	9.4%	13.4%	14.9%

Figure 5.7: Difference in relative competence between expansion-contraction compression and condensed nearest neighbor on five different case bases (Leake & Schack, 2018).

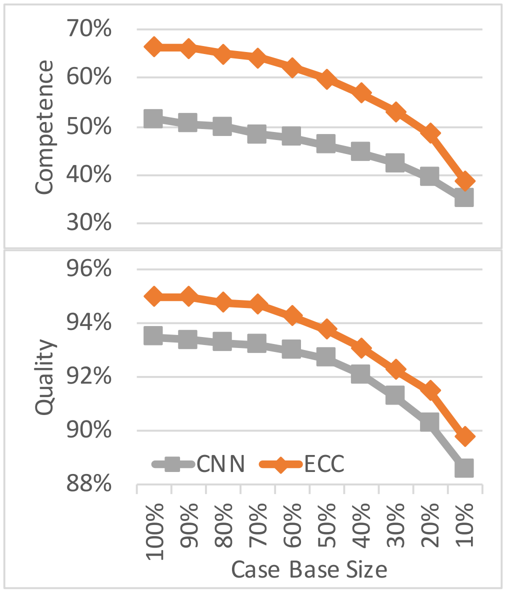


Figure 5.8: Competence and quality for expansion-contraction compression (ECC) and condensed nearest neighbor (CNN) on Houses data set with adaptation rules (Leake & Schack, 2018).

the lower left graph in Figure 5.4). Expansion-contraction compression outperformed condensed nearest neighbor on four of the five case bases (Houses, Iris, Wine, and Wine Quality), but not on Car Evaluation. This suggests that the expansion-contraction strategy might benefit compression across many domains and raises the question of which factors determine whether it will be beneficial.

5.4.5 Question 5: Applying Sample Adaptation Rule Sets

Experiments with adaptation rules used the Houses data set, with a gap radius of 20% (corresponding to a large gap in the previous experiments), a 5% coverage criterion, and 40 automatically-generated rules for each trial. Expansion-contraction compression used the adaptation rules both to generate ghost cases and to adapt solutions. Figure 5.8 (on page 79) shows the average preservation of competence and quality during expansion-contraction compression condensed nearest neighbor. Expansion-contraction compression outperformed condensed nearest neighbor in both dimensions at all compression levels. Thus expansion-contraction compression was beneficial both for modeled adaptation and adaptation with generated adaptation rules.

5.5 Future Work

In the experiments, ghost case generation is an unguided process, generating ghost cases from randomly-selected cases using randomly-generated adaptations. In addition, ghost cases are generated within neighborhoods of existing cases, making ghost cases most likely to be added near regions that are already densely populated. Ghost cases can be useful there, for example, as spanning cases (Smyth & Keane, 1995), bridging two competence regions. However, such placement may not help to populate a distant sparse region. Identifying and targeting sparse regions on which to focus ghost case generation might increase the benefit of expansion-contraction compression. On the other hand, populating such regions might be detrimental if representativeness generally holds. Thus determining guidance strategies for ghost case generation and assessing their effects is an interesting area for future study.

For example, areas to target for ghost generation could be selected by dividing the case base into regions and applying Monte Carlo methods to assess case base density (Leake & Wilson, 2011). Another interesting direction would be to develop maintenance strategies that tracked, reasoned about, and responded to expected future case distributions, to explicitly determine the expected utility of exploring unseen areas of the case space.

5.6 Summary

This chapter proposed and evaluated a novel approach to case-base maintenance, expansion-contraction compression, aimed at achieving better competence retention when the representativeness assumption may not hold. Expansion-contraction compression contrasts with other approaches, which focus on how best to exploit the cases in the case base, in going outside problems addressed by the case base to explore the larger problem space guided by case adaptation knowledge. Expansion-contraction compression can be seen as applying a knowledge container transfer strategy: It uses adaptation knowledge to generate case knowledge in the form of ghost cases which are then added to the pool of cases available to the compression algorithm. Chosen ghost cases can be retained as-is or can be used to guide verification or active learning of new cases. Experiments support the expected result of expansion-contraction compression improving competence retention for less-representative case bases. Surprisingly, expansion-contraction compression also often improved competence retention even for standard case bases. Thus expansion-contraction compression appears promising. Interesting questions remain for studying factors affecting performance and guiding ghost case generation to maximize the effectiveness of expansion-contraction compression.

Chapter 6

Predictive Case Discovery for Problem-Distribution Drift

"Chapter 3: Subdividing Cases for Deletion" (on page 26) questioned two assumptions of case-base maintenance: (1) that cases are of uniform size, and (2) that cases cannot be subdivided. Setting aside these assumptions gave rise to the flexible feature deletion case-base maintenance strategy (Schack, 2016; Schack & Summers, 2017). "Chapter 4: Prioritizing Deletion by Recoverability via Adaptation Knowledge" (on page 44) applied adaptation knowledge to flexible feature deletion — yielding adaptation-guided feature deletion (Leake & Schack, 2016; Schack, 2016). And "Chapter 5: Data Augmentation to Expand Choices for Deletion" (on page 61) questioned the representativeness assumption — yielding expansion-contraction compression (Leake & Schack, 2018; Schack, 2019). This chapter will question the assumption of regularity, and setting this aside will also lead to a novel case-base maintenance strategy.

It is well known that the effectiveness of any case-based reasoning system depends on two types of regularity, which have been formalized as problem-solution regularity and problem-distribution regularity (Leake & Wilson, 1999). *Problem-solution regularity* can be informally characterized as, "Similar problems have similar solutions," and is needed in order for the adaptation of the solutions to retrieved cases to be useful in similar situations. *Problem-distribution regularity* can be informally be characterized as, "Future problems will resemble past problems." This property is necessary so that learned cases will tend to be useful in the future. In addition to these types of regularity, another regularity is required by machine learning systems more generally: the regularity that learned concepts tend to remain valid over time.

The effectiveness of applications of case-based reasoning suggests that a combination of careful system design and suitable problem environment can provide these properties in practical situations. However, they are not guaranteed. As case-based reasoning is used in long-lived systems, developments over time may diminish any of these regularities, presenting difficulties for those systems. When concepts change over time, the result is concept drift (Widmer & Kubat, 1996), which makes prior cases no longer apply and requires case base updating (Cunningham et al., 2003). When similarity criteria no longer reflect similarity for system needs, for example due to changes in case adaptation knowledge, performance may degrade (Leake et al., 1997). When the distribution of problems changes over time, the quality of case base coverage may degrade, contravening problem-distribution regularity and requiring maintenance to ensure that the system has the cases that it needs going into the future.

This chapter first examines each of these types of regularity, and then focuses on problem-distribution regularity, which to the author's knowledge, has received little prior attention in case-based reasoning. To mitigate problem-distribution regularity failures, it proposes *predictive case discovery* to identify and anticipate problem-distribution drift to guide the selection of cases to request from an external source.

This chapter identifies general requirements and classes of detection methods that may be used to guide case discovery. It then illustrates with a clustering-based method aimed at identifying "hot spots" in the problem space on which to focus discovery. It presents an evaluation of this sample method in a path planning domain across four scenarios: no drift, non-cyclical drift, cyclical drift, and drift from obsolescence. The results support the general effectiveness of the strategy and also illustrate its limitations. The chapter concludes with future opportunities for predictive case discovery.

6.1 Regularities Underpinning Case-Based Reasoning

Problem-Solution Regularity: The effectiveness of the reuse of past cases depends on *problem-solution regularity* — the property that solutions to similar problems will provide a useful starting point for solving a new problem (Leake & Wilson, 1999). Often in the case-based reasoning literature, the assumption is that adapting the solution to a similar problem should reduce solution generation cost compared to reasoning from scratch, for generating an acceptable solution. Systems achieving this type of problem-solution regularity have been demonstrated in multiple scenarios (Smyth & Keane, 1998; Veloso, 1994).

Problem-Distribution Regularity: The benefit to the case-based reasoning process of storing past cases depends on *problem-distribution regularity* — the property that the distribution of future problems will tend to reflect that of past problems, such that accumulated stored cases from past episodes tend to provide useful information for future problems (Leake & Wilson, 1999).

Formalizing the Properties: Leake and Wilson provide a formalization of these properties (Leake & Wilson, 1999). They define problem-solution regularity as depending on four factors:

1. The retrieval function that the system uses to map problems to cases in the case base
2. A definition of the goals to be satisfied by retrieval — what would make a case a good starting point for solving a new problem
3. The initial set of cases available to the system
4. The problems that the system is called upon to solve

Retrieval goals are often defined in terms of a predefined similarity measure, such as the semantic similarity between a target problem and the problem part of stored cases. However, they can be based on other criteria, e.g. that the solution to the retrieved case should be inexpensive to adapt to generate a correct solution to the new problem (Smyth & Keane, 1998). Another possible criterion is that the retrieved case should be adaptable to generate a result within a certain accuracy.

Items (3) and (4) reflect that problem-solution regularity must be measured in terms of the problems the system has to solve. Items (1), (2), and (3) are under the control of a system designer. For example, to further problem-solution regularity, a retrieval function aimed at retrieving adaptable cases could be hand-designed, or it could be learned (Leake & Ye, 2021). However, new cases received by the system, as referenced in Item (4), may cause changes in problem-solution regularity if the retrieval function is ill-suited to judging similarity for those cases.

Problem-distribution regularity reflects the correlation between the distribution of cases in the case base and the distribution of future problems. Even a case base evenly distributed across the problem space may have low problem-distribution regularity if upcoming problems are not evenly distributed.

Problem-distribution regularity has been formalized in terms of the long-term behavior of a case-based reasoning system: The probability that, at a given point in processing a stream of problems, the system will be able to retrieve a case sufficiently close to an input problem (Leake & Wilson, 1999). The assumption of problem-distribution regularity relates to the influential *representativeness assumption* of case-base maintenance, "The case base is a representative sample of the target problem space," (Smyth & McKenna, 2002). This property is important for assessing case competence for the possible range of future problems. However, problem-distribution regularity only measures whether eventually the system will have a sufficient probability of containing the cases needed to cover new problems; it measures (after the fact) whether it was possible to cover most problems actually received by the system, rather than predicting whether the case base provides a good sample of possible future problems.

6.2 How Regularity May Degrade: Types of Drift

Even in suitable domains, regularities may not always hold. Existing cases may become obsolete (Leake & Wilson, 1999), space or time requirements may necessitate deletion (Smyth, 2005; Smyth & Keane, 1995) with corresponding competence loss (Smyth & McKenna, 2002), or the system may simply not have been provided with sufficient initial cases (or the experiences to acquire cases) to address current problems. The following subsections consider how drift may affect concept regularity and problem-distribution regularity.

6.2.1 Concept Drift

Concept drift, a widely explored phenomenon in machine learning, refers to the situation where the underlying concepts or relationships between the problem and solution change over time (J. Lu et al., 2018). For example, in a system for property valuation, inflation could cause prices to increase over time, providing a gradual transition, or a chemical spill could cause an abrupt decrease in valuations in a particular region. Or in a system for sentiment analysis, the evolution of slang could change the interpretation of the sentiment of online messages. By making cases inaccurate, concept drift can degrade the performance of a formerly accurate case-based reasoning system. When inaccuracy arises from the time passed since acquiring a case, then this is characterized as *case obsolescence*.

6.2.2 Problem-Distribution Drift

Another issue arises in domains for which the problem distribution changes over time. For a disaster management system, climate change can lead to shifts in weather patterns, reducing the usefulness of a case base containing response plans for historical weather patterns. For recommender system for a travel agency, certain areas may become "hot" destinations — resulting in a different range of necessary coverage. For a real estate appraisal system, developers may build newer properties with different characteristics from older properties and beyond the scope of reliable adaptation.

Problem-distribution drift refers to a change in the distribution of problems that the case-based reasoning system must solve. If a customer support system has cases for certain problems and the types of problems customers encounter change, then the existing case base may become less useful, and the case-based reasoning system may need to acquire new cases that are relevant to the new problem distribution. Because problem-distribution regularity is defined in terms of the limit of case base growth, even when a domain satisfies problem-distribution regularity in the long term, practical issues can still arise in the short term. Problem-distribution drift can be caused by various factors, including changes in the environment, changes in user preferences, changes in technology, or changes in the problem space itself:

Environmental changes The environment in which a case-based reasoning system operates can change over time. For example, in a medical diagnosis system, changes in the prevalence of certain diseases can lead to changes in the distribution of medical problems that the system needs to diagnose and treat.

User preferences User desires or the problems that they wish to address may change. For example, a recommender system for clothing or travel packages needs to change seasonally as the problem distribution changes with user preferences. In some scenarios, changes in fashion may also render prior cases obsolete.

Technological changes As technology advances, new problems can arise that were not present before. For example, in a software support system, upgrades in the software can incorporate new features with associated bugs for which the system needs to provide support.

Changes in the problem space Changes in the underlying physics or changes in the social or economic context of a domain may affect problem distributions. For example, in a financial prediction system, changes in the market conditions or regulations can lead to changes in the distribution of financial problems that the system needs to predict.

6.2.3 Adversarial Drift

Adversarial drift refers to data drift in which a reasoner responds to cases presented by an adversary which presents characteristically different cases over time with the intention of degrading performance (Kantchelian et al., 2013). Adversarial drift could involve concept drift, problem-solution regularity drift, or problem-distribution regularity drift. Adversarial drift can be observed in imperfect information games such as poker, in which players benefit from associating their opponents' bets and "tells" to their strategic position or the strength thereof. In such games, the opponent may bluff their bets or fake their tells, intentionally breaking regularity to gain an advantage. Delany et al. (2005) tracked and mitigated adversarial drift as spammers adapted their techniques to circumvent spam filters.

6.3 Addressing Problem-Distribution Drift with Guided Case Discovery

Problem-distribution drift may be addressed in a two-step process. First, the case-based reasoning system can detect and track drift. And second, it can extrapolate to anticipate the path of the drift and request cases in that path. For example, in a case-based reasoning system for recommending travel packages, if a new destination is published in a major magazine, then the number of requests for packages to that destination may increase. If the trend of having more requests in that area can be detected, then the system could request additional cases in that area to better prepare for future requests.

6.3.1 Prior Work on Drift Detection

Drift detection algorithms generally fall into four categories: error rate-based drift detection, data distribution-based drift detection, multiple hypothesis-based drift detection, and competence-modeling strategies.

Error rate-based strategies compare the error rate of the model before and after a certain time period. These algorithms are commonly used in classification tasks, where the error rate can be calculated by comparing the predicted class labels to the true class labels. Increased error rate over time can indicate data drift. One common error rate-based algorithm is the ADWIN algorithm which adapts the window size based on observed changes in the error rate (Bifet & Gavaldà, 2007).

Data distribution-based strategies compare the data distribution before and after a certain time frame. These algorithms can detect gradual changes in the data distribution which may not be reflected in the error rate. One popular algorithm is the Kullback-Leibler (KL) divergence-based method which measures the difference between two probability distributions. If the divergence exceeds a threshold, then this can indicate data drift (Dasu et al., 2006).

Multiple hypothesis-based strategies test multiple hypotheses simultaneously, making it possible to detect complex drift patterns. These algorithms are useful when the data drift is not well understood or cannot be modeled using a simple statistical model. One example of a multiple hypothesis-based algorithm is the Just-in-Time adaptive classifiers (JIT) algorithm which uses a sequence of hypotheses tests to detect changes in the data distribution (Alippi & Roveri, 2008).

Competence-modeling strategies detect drift based on changes in competence. For example, N. Lu et al. (2014) applies a competence-modeling strategy to detected data drift in case-based reasoning which provides descriptions and quantification of changes as well as statistical guarantees of reliability.

Depending on the strategy, the curse of dimensionality can also impact the detection of data drift. The *curse of dimensionality* refers to the problem of high dimensionality in a problem space leading to sparsity and high computational costs (Köppen, 2000). As the number of dimensions increases, cases can become widely dispersed in the high-dimensional space making changes in their distribution difficult to detect. One approach to addressing the curse of dimensionality is to reduce the dimensionality of the dataset by selecting relevant features before applying drift detection algorithms.

6.3.2 Case Discovery

When problem-distribution regularity decreases, a potential repair is to add cases to the case base. Case discovery can fill gaps in the distribution of the cases in the case base to cover upcoming problems that would otherwise fall into those gaps. In systems including a generative component capable of solving problems from scratch, discovery can be done by calling upon that component. In that situation, generating the solution in advance does not increase competence but provides speed-up learning by avoiding the need to generate a solution from scratch at run time (e.g. Prodigy / Analogy from Veloso (1994)). In other scenarios, discovery may be done by requesting cases beyond system competence from an external source such as a domain expert.

Which Cases to Discover As the cost of case solicitation may be high, effectiveness of discovery depends on targeting (Massie et al., 2005; McSherry, 2000, 2002; Owrang O., 1998). For example, McKenna and Smyth (2002) identify competence holes in a case base to fill by discovering spanning cases. Such approaches can be combined with problem prediction to further focus on the regions of the problem space likely to be relevant to incoming problems.

Additional methods could be brought to bear from outside case-based reasoning. For example, the SMOTE oversampling algorithm (Fernández et al., 2018) mitigates class imbalance by generating synthetic instances by interpolating between neighboring minority instances. Extensions to SMOTE can handle concept drift on time series data (Hoens et al., 2012). Case discovery and oversampling can also be seen as similar in spirit to data augmentation for neural networks (Iwana & Uchida, 2021). Applying a similar spirit to case-based reasoning, adaptation rules provide a knowledge-rich source of transformations that go beyond interpolation, yielding “ghost cases” that tend to preserve case cohesion (Leake & Schack, 2018; Schack, 2019). Ghost cases generated by adaptation can improve efficiency and compression but generally would not be expected to increase competence.

CB: the case base,

part: a partition function,

active-part: a function selecting one of the subsets of a partition,

rep: a function generating a case to discover given a set of cases

generate-target-case(CB, part, active-part, rep):

1. parts $\leftarrow$ part(CB)
2. chosen-part $\leftarrow$ active-part(parts)
3. target-case $\leftarrow$ rep(chosen-part)
4. Return target-case

Figure 6.1: General discovery procedure (Leake & Schack, 2023).

Let d be a distance metric, N the number of desired clusters, and CB the case base:

1. Divide the training cases into N clusters using the k-means algorithm for distance d.
2. Randomly choose a cluster CL of training cases.
3. Find cr = centroid case of CL.
4. Alter the value of a single feature of cr to yield a variation to request for discovery.

Figure 6.2: k-means discovery algorithm (Leake & Schack, 2023).

6.3.3 Clustering-Based Case Discovery

This chapter proposes a general case discovery strategy of dividing the problem space into parts, predicting the most active part, selecting a point in that part, and then requesting discovery of that case. This algorithm is illustrated in Figure 6.1 (on page 91). As an illustration and for empirical evaluation, this chapter applies this strategy in a simple clustering-based approach called *k-means discovery*.

The k-means discovery algorithm, shown in Figure 6.2 (on page 91), uses k-means clustering to divide the problem space into N regions for a predefined value N. It then selects a random cluster from the N regions, meaning that N determines the balance between exploration (for large N, resulting in small regions) and exploitation (for small N, resulting in large regions). Alternative methods could, for example, favor "up and coming" clusters with recent activity.

After a cluster is selected, k-means discovery finds the case at the centroid of that cluster. Then it generates a variation on the centroid case by altering that case. In this simple demonstration, it does so by altering the value of a single feature. For example, in the path planning domain, the variation could be a path where one endpoint is the same as in the centroid path, and the other endpoint is randomly chosen from the entire graph. Knowledge-rich methods could be used, such as adaptation strategies simultaneously altering several features of the problem description. (Leake and Schack (2018) discuss adaptation of both problem and solution parts of cases to generate ghost cases.) The variation on the centroid case becomes the case to discover.

This chapter uses clustering-based case discovery as an illustration because it is simple and requires minimal domain knowledge, making it suitable for domains where knowledge is scarce or costly to obtain. And because, by choosing the case at the centroid of the cluster, the strategy tends to discover a variant of a case representative of that cluster — which the author hypothesized would reflect "hot spots" because less representative cases would tend towards the edges. As needed, k-means could be replaced with other methods. For example, spherical k-means, using cosine similarity, could be used for textual cases, or affinity propagation could be used for clustering based on pre-computed distances and without predetermining a specific number of clusters.

6.4 Managing Multiple Strategies

Drift detection and case discovery strategies may have different strengths and weaknesses depending on the characteristics of the problem domain and the data drift (if any). Additionally, each strategy may have parameters (such as window size or number of clusters) that need to be tuned to achieve optimal performance. Choosing the "right" strategies and parameters can impact the accuracy and efficiency of drift detection and case discovery.

A potential approach to dealing with this issue is to develop a library of strategies and select which to apply via a bandit meta-strategy. If the strategies are initially selected at random, and the choice between strategies is weighted based on the number of problems for which the cases discovered by the strategy were successfully applied in the past, then choices could be refined to favor successful strategies. However, for a rapidly-changing distribution, this knowledge could quickly become obsolete. This is a subject for future study.

6.5 Evaluation

The experiments evaluate the proposed clustering-based case discovery strategy, examining the following question:

What effect does the k-means discovery strategy *have on the adaptation cost of retrieved cases compared to random case discovery across four scenarios: no drift, non-cyclical drift, cyclical drift, and obsolescence?*

6.5.1 Testing Scenario

The experiments are conducted in a simulated path planning domain. This is an established domain for evaluating case-based reasoning systems (e.g. PathFinder from Smyth and Cunningham (2005)), and it has practical applications such as for mobile robots (Hodál & Dvořák, 2008) and autonomous underwater vehicles (Vasudevan & Ganesan, 1996). The intuitive motivation for the scenario is that an agent needs to travel from place to place.

Cases are modeled with a problem part and a solution part. The problem part is the starting and ending points of the path to travel, and the solution part is a list of nodes along the path from the starting point to the ending point. Some routes and destinations will occur more frequently than others, and the scenario can change over time due to moving homes, changing jobs, closing roads, and so on, providing problem-distribution drift.

The experiments model the road and transit network as a graph in which each node is a potential destination. Variations on the scenario explore obsolescence of cases and cyclical and non-cyclical problem-distribution drift. Obsolescence is modeled by assigning an expiration age to cases; cases are no longer available after a certain number of problems have been solved. The following paragraphs in this section describe the details of the testing scenario, and code for replicating the evaluation is available in a public GitHub repository.[1]

Constructing the Graphs Each iteration was done on a different graph generated with 100 nodes. Each node was positioned randomly in two-dimensional space. The edges were constructed with k-edge augmentation where $k = 1$. The edge augmentation ensured that the graph could not be disconnected unless k or more edges were removed. Although the graphs were unweighted, the edge augmentation was weighted by the Euclidean distance between nodes. The number of edges varied from graph to graph.

The edge augmentation method served three purposes: First, some path should exist between

[1]https://github.com/schackbrian2012/ICCBR-2023

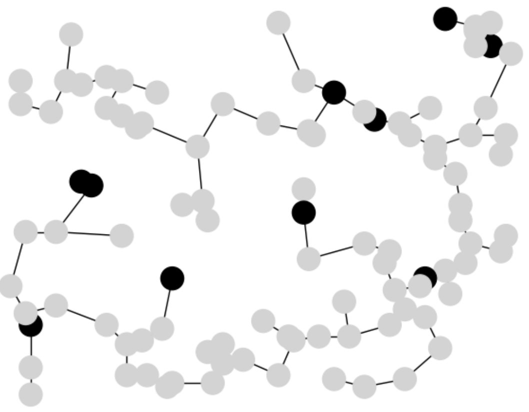


Figure 6.3: An example of a graph modeling a randomly constructed transit network. Frequently visited nodes are colored black, and infrequently visited nodes are colored light gray (Leake & Schack, 2023).

any pair of nodes so that path planning is possible. Second, most paths should be longer than a single edge so that path planning is non-trivial. And third, edges should be more common between nearby pairs of nodes than between distant pairs of nodes, for a correlation between Euclidean distance (for retrieval) and graph distance (for adaptation). Figure 6.3 (on page 95) illustrates a graph constructed by this method.

Populating the Case Base For each iteration, the case base was populated by first randomly choosing 10 distinct nodes from the graph, to serve as the frequently visited nodes. These were unequally weighted from most frequent (10) to least frequent (1). A node was randomly chosen from the frequently visited nodes, weighted by node weights. Another node was randomly chosen from the whole graph, giving each node an equal probability. Either a departing path (from a frequently visited node to a random node) or a returning path (from a random node to a frequently visited node) was constructed. The process was repeated to generate 1,000 paths.

This method of populating the case base served three purposes: First, because the nodes are randomly selected from the graph, the paths in the case base cover different parts of the graph.

This diversity is important for evaluating the ability of the case discovery strategy to adapt to changes in the problem distribution. Second, by giving higher weights to frequently visited nodes, the method accounts for the fact that some parts of the graph may be more important than others, a realistic assumption for the path planning domain. Third, 1,000 paths is large enough to capture the variability in the problem distribution and the ability of the case discovery strategy to adapt to it.

Discovery Methods The evaluation compared three discovery methods. The *No Discovery* method is a baseline that does not discover any cases. The *Random Discovery* and *Clustering-Based Discovery* methods discover one case at each time step. The Random Discovery method selects random values for each feature of the problem part of the case for discovery. In the path planning domain, the starting and ending points of the path for discovery are two nodes randomly chosen from the entire graph. The Clustering-Based Discovery method uses k-means discovery with eight clusters and a categorical distance metric. Exploratory analysis of different numbers of clusters showed that eight was effective for this task. The categorical distance metric treats each node as a category (instead of a two-dimensional coordinate) and compares nodes by exact match.

Distance Metric for Retrieval The retrieval process yields the most similar training or discovery case to the testing problem measured according to the Euclidean distance between two four-dimensional vectors — the x- and y-coordinates of the source and target nodes of each path — resolving ties arbitrarily.

Performance Assessment Methods The three discovery methods were compared using two performance assessment methods: Leave One Out and Time Series Cross-Validation. For both evaluation methods, at each time step, the reasoner is presented with the problem part of a different testing case — making the number of time steps equal to the number of cross-validation folds. The order of testing is the same as the order of generation described in the "Populating the Case Base" subsection (on page 95). For the Leave One Out method, the training cases include all cases other than the testing case. For the Time Series Cross-Validation method, the training cases include all cases encountered prior to the testing case. The experiment ran for 10 iterations with different graphs and case bases in each iteration.

The Leave One Out method tests generalizability by iteratively training on all but one data point, and then testing on the left-out data point, to prevent overfitting. The Time Series Cross-Validation method is suited for temporal data, as it evaluates the performance on future time points based on the training data available up to that point, simulating a real-world scenario where the reasoner has to predict future events based on past data.

Distance Metric for Evaluation The distance metric for evaluating the adaptation cost of a solution sums the number of edges in the shortest path to adapt the starting and ending points of a training case to the starting and ending points of the testing case. Unlike the distance metric for retrieval, the distance metric for evaluation ignores Euclidean distance. For an exact match, the distance metric returns zero.

6.5.2 Variations on the Testing Scenario

The experiments evaluated four variations on the testing scenario: No Drift, Non-Cyclical Drift, Cyclical Drift, and Obsolescence.

No Drift Scenario The frequently visited nodes remain the same throughout the time series. The No Drift Scenario serves as a baseline for comparison with the other scenarios. It tests the ability of the case discovery strategy to handle a stable problem distribution where the frequently visited nodes remain the same throughout the time series.

Non-Cyclical Drift Scenario Halfway through the time series, at time step 500, the frequently visited nodes are changed to a different set of frequently visited nodes constructed by the same random sampling of the nodes. This in turn abruptly changes the paths constructed from the frequently visited nodes. The Non-Cyclical Drift Scenario tests the ability of the case discovery strategy to handle abrupt changes in the problem distribution.

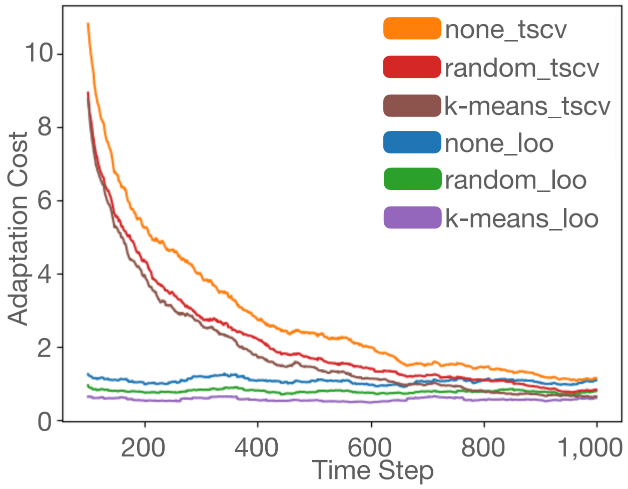
Cyclical Drift Scenario This scenario alternates between two sets of frequently visited nodes for two equal-length phases of each set. It tests the ability of the case discovery strategy to handle cyclic changes in the problem distribution, which can occur due to seasonal changes or recurring patterns in user behavior. Specifically, this scenario tests the ability of the case discovery strategy to adapt to two different sets of frequently visited nodes and construct paths that switch between the two sets.

Case Obsolescence Scenario This scenarios tests the adaptation cost of naively retrieving cases without regard to case obsolescence. It simulates case obsolescence by only reusing cases stored or discovered at less than 100 time steps before the testing problem that they solve (for the evaluation that allows use of all cases in the case stream, future cases are also only considered within 100 time steps). Evaluation penalizes the retrieval of an obsolete case by re-planning the entire path from the starting point to the ending point of the testing problem. The number of edges in the re-planned path is treated as the cost of adapting from the obsolete case to the testing problem. The distance metrics for retrieval and clustering do not consider obsolescence.

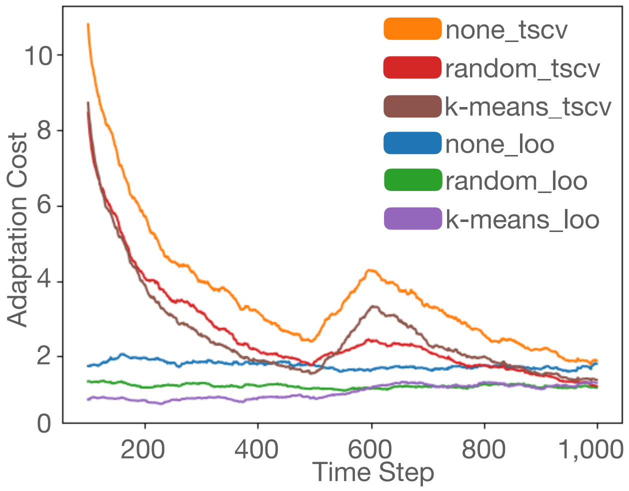
6.5.3 Experimental Results

Figure 6.4 (on page 99) shows the experimental results. The x-axis measures the time step of the testing case under evaluation. The y-axis measures the adaptation cost of the solution. Each graph presents the rolling average of adaptation cost over a window of 100 time steps. The order of the discovery strategies in the legend roughly matches the adaptation cost from highest to lowest.

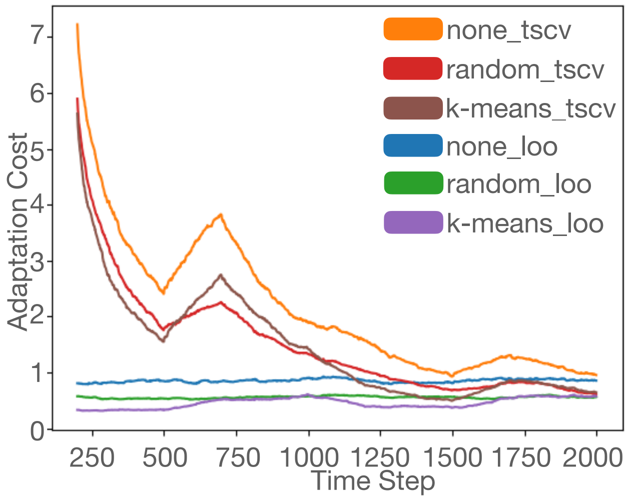
The none_loo and none_tscv series use the No Discovery strategy. The random_loo and ran-



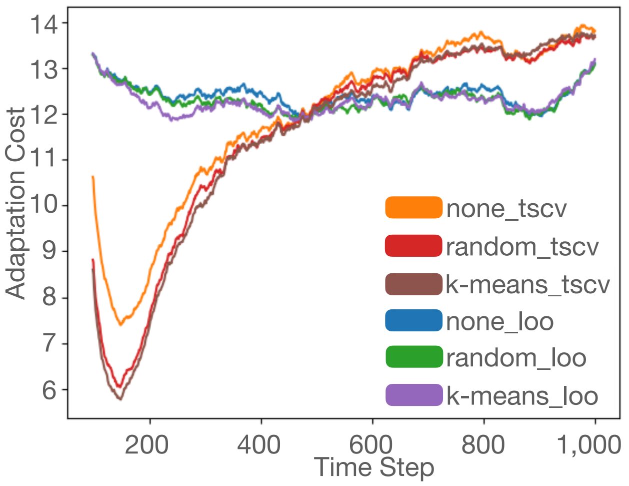
(a) No Drift



(b) Non-Cyclical Drift



(c) Cyclical Drift



(d) Case Obsolescence

Figure 6.4: Evaluation of clustering-based case discovery across four scenarios (Leake & Schack, 2023).

dom_tscv series use the Random Discovery strategy. And the k-means_loo and k-means_tscv series use the Clustering-Based Discovery strategy. The none_loo, random_loo, and k-means_loo series use the Leave One Out evaluation method; the none_tscv, random_tscv, and k-means_tscv series use the Time Series Cross-Validation evaluation method.

No Drift Scenario — Figure 6.4a The adaptation cost of Leave One Out evaluation remains steady over time because both past and future cases make up its training. The adaptation cost of Time Series Cross-Validation improves over time because the number of cases for training and discovery increase over time. The adaptation cost of each discovery method evaluated by Time Series Cross-Validation approaches the adaptation cost of the same discovery method evaluated by Leave One Out. Random Discovery outperforms No Discovery because discovery yields additional cases beyond the training cases. Clustering-Based Discovery outperforms Random Discovery because the cases discovered by the former tend to match the distribution of problems, and cases discovered by the latter tend towards an even distribution over the problem space.

Non-Cyclical Drift Scenario — Figure 6.4b The earlier phase goes from time steps 0–500, and the later phase goes from time steps 500–1,000. The plot does not show time steps 0–100 because of the rolling window of 100 time steps. The earlier phase resembles the No Drift Scenario in Figure 6.4a (on page 99) because no drift has yet occurred. Halfway through, at time step 500, adaptation cost increases for the Time Series Cross-Validation evaluation method because the problem distribution of training cases from the earlier phase does not match the problem distribution of testing cases from the later phase.

Adaptation cost increases more steeply for the none_tscv and k-means_tscv series than the random_tscv series because the latter discovers cases unbiased by the problem distribution of the earlier phase. The adaptation cost of k-means_loo also increases because it discovers the same cases as k-means_tscv — the majority of which follow the problem distribution of the earlier phase.

Around time step 600, adaptation cost for Time Series Cross-Validation begins to decrease again as training cases arrive from the later phase and the k-means clustering algorithm incorporates training cases from both phases. Like Figure 6.4a, approaching the end at time step 1,000, the adaptation cost of each discovery method evaluated by Time Series Cross-Validation approaches the adaptation cost of the same discovery method evaluated by Leave One Out.

Cyclical Drift Scenario — Figure 6.4c The first phase, from time steps 0–500, resembles the No Drift Scenario in Figure 6.4a (on page 99) because no drift has occurred yet. The first two phases, from time steps 0–1,000, resemble the Non-Cyclical Drift Scenario in Figure 6.4b (on page 99) because the drift has not yet repeated an earlier phase. The first drift (from the first phase to the second phase) and the third drift (from the third phase to the fourth phase) impact adaptation cost more than the second drift (from the second phase to the third phase) because the problem distribution of the training cases and the Guided Discovery cases in the first phase matches the third phase.

Obsolescence Scenario — Figure 6.4d Adaptation cost drops for the Time Series Cross-Validation evaluation method before time step 200 as training and discovery cases arrive to solve testing problems. Then the ratio of obsolete to contemporary training and discovery cases increases — causing an increase in the number of obsolete cases retrieved and an increase in the adaptation cost. Adaptation cost increases for the Leave One Out evaluation method at the start and end of the time series. The window of contemporary cases is 100 time steps before and after the test problem, but the start (resp. end) of the time series has fewer cases before (resp. after) the test problem.

6.6 Summary

This chapter discussed the regularities required by case-based reasoning and potential issues arising from various types of drift, including concept drift, problem-distribution drift, and adversarial drift. Then it discussed different strategies for detecting drift, such as error rate-based, data distribution-based, multiple hypothesis-based, and competence-modeling strategies. It illustrated a case discovery strategy, k-means discovery, guided by k-means clustering, and evaluated its effectiveness on synthetic time series data in a path planning domain across four different scenarios. The evaluation demonstrated that it outperforms baselines. However, because the effectiveness of discovery strategies depends on characteristics of the drift itself, there is no universal strategy. An important next step is to explore additional strategies for drift prediction and identifying cases to discover, including drawing on methods from outside case-based reasoning, and investigating ways to select the right strategy for the domain and task.

Chapter 7

Conclusion

This dissertation explored strategies for case-base maintenance that extend beyond the conventional paradigm of case deletion. The focal point has been the development and evaluation of four alternative maintenance strategies: flexible feature deletion, adaptation-guided feature deletion, expansion-contraction compression, and predictive case discovery.

7.1 Key Contributions

The first strategy, flexible feature deletion, diverges from the assumption that cases are indivisible, unitary objects and instead allows for the selective removal of case contents, thereby presenting a nuanced approach to compressing the case base when cases vary in size and features differ in importance. This approach is particularly beneficial in domains where different levels of detail can be used for reasoning.

Adaptation-guided feature deletion builds on the foundations laid by flexible feature deletion, applying adaptation knowledge to prioritize the deletion of features based on their recoverability. The evaluation of this method in a path-finding domain demonstrated its ability to preserve competence under compression, thereby supporting its applicability in domains with sufficient adaptation knowledge.

Expansion-contraction compression introduced the concept of ghost cases. These cases, generated through the adaptation-based exploration of unseen problem spaces, allow for the augmentation of the case base to improve competence-based deletion on unrepresentative case bases. This approach proved beneficial not just for less-representative case bases but also offered improvements in competence retention on representative case bases, making it a promising avenue for case-base maintenance.

Finally, predictive case discovery aims at addressing the effects of problem-distribution drift by anticipating and acquiring cases that are predicted to be beneficial in solving future problems, thus aiding in reducing adaptation costs in evolving problem-solving landscapes.

7.2 Implications

These strategies have shed light on potential enhancements that can be made in the field of case-based maintenance. By going beyond case deletion and exploring the granularity of feature deletion, leveraging adaptation knowledge for recoverability, and utilizing ghost cases for competence enhancement, there is a progression in the strategies available for maintaining the quality and competence of case bases.

These strategies mitigate the immediate challenges posed by big data and enhance the applicability of case-based reasoning to diverse domains, balancing competence and computational constraints. The approaches and applications examined in this dissertation support the necessity and viability of advancing maintenance strategies, ensuring that case-based reasoning can continue to adapt and evolve in response to ever-changing landscapes of problems and solutions.

7.3 Future Work

While this dissertation expanded the repertoire of maintenance strategies, it also opened several avenues for future exploration and development. The creation and placement of ghost cases, in the context of expansion-contraction compression, warrant further exploration to discern optimal strategies for ghost case generation in varying densities of case bases.

Further exploration of creative destruction, as presented in the studies on recoverability-based feature deletion, could yield maintenance strategies focused on improving competence by revising or restructuring cases to enhance adaptability. Additionally, the intersection of adaptability and recoverability in adaptation-guided feature deletion needs more extensive research to refine and optimize the trade-off between retrieval from the case base and recovery cost.

Moreover, the exploration of nuanced and knowledge-rich strategies in areas where flexible feature deletion is applicable can be pursued, particularly focusing on the development of knowledge-rich techniques for compression of cases and feature bundlings.

Lastly, the objective to track and predict different types of drifts, especially in the context of case discovery, needs further attention to develop mechanisms that can effectively respond to shifts in problem landscapes and ensure the sustained relevance and competence of case-based reasoning systems in dynamic environments.

In conclusion, this dissertation has strived to explore and contribute modestly to the domain of case-base maintenance by introducing and evaluating strategies that look beyond case deletion. The insights gained from these strategies are stepping stones towards further advancements in the field, aiming to fortify the adaptability and resilience of case-based reasoning in the face of evolving challenges.

Glossary

adaptation-guided feature deletion (AGFD) case-base maintenance strategy which subdivides cases and prioritizes their components for deletion according to their recoverability via adaptation knowledge (Leake & Schack, 2016; Schack, 2016). 14, 45

adversarial drift data drift in which a reasoner responds to cases presented by an adversary which presents characteristically different cases over time with the intention of degrading performance (Kantchelian et al., 2013). 87

analogical reasoning method of artificial intelligence which discovers and maps parallels between the relational structure of systems or reasoning steps usually from different domains (Burstein, 1989). 4

artificial neural network method of artificial intelligence inspired by animal brains which loosely models interconnected neurons which process signals and then signal their neighbors (Haykin, 2004). 5

case element of the case base of a case-based reasoning system containing features and their values and having more or less structure depending on the domain (Richter, 2003). 4

case base knowledge container of a case-based reasoning system which stores problems with known solutions either seeded by the training data or solved in the past (Richter, 2003). 4

case obsolescence inaccuracy of a solution to a case arising from the time passed since acquiring the case. 86

case-base maintenance mitigation for the swamping utility problem which chooses the most valuable cases to retain and the least valuable cases to delete in order to maintain a compact, competent case base (Juarez et al., 2018). 1, 8

case-based creativity area of case-based reasoning which aims to develop software capable of the production of novel and interesting products or assisting humans in the production thereof (Schank & Leake, 1989). 5

case-based reasoning (CBR) method of artificial intelligence which solves a problem by adapting the solution to a similar problem already solved (Aamodt & Plaza, 1994; de Mantaras et al., 2005; Riesbeck & Schank, 2013). 1

case-based reasoning cycle problem-solving process for case-based reasoning consisting of four phases: retrieval, reuse, revision, and retention (de Mantaras et al., 2005). 2

casuistry method of applied ethics and jurisprudence which solves a moral problem by extending or distinguishing rules extracted from other instances (Searing, 2009). 1

Computerized Yale Retrieval and Updating System (CYRUS) an early case-based reasoning system which modelled long-term memory and fact retrieval for events in the lives of important people (Kolodner, 1983). 2

concept drift refers to the situation in machine learning where the underlying concepts or relationships between the problem and solution change over time (J. Lu et al., 2018). 86

concept map informal two-dimensional visual representation of concepts and their relationships, representing a particular user's conceptualization of a domain (Novak et al., 1984). 31

condensed nearest neighbor (CNN) a machine learning technique that reduces the size of a dataset while preserving its representativeness and classification accuracy by selecting a subset of the most informative and representative examples which can be used to train classifiers with reduced computational costs (Hart, 1968). 62

coverage approximation calculated by case-base maintenance strategies of the number of neighboring problems that a case can solve through adaptation (Smyth & Keane, 1995). 8, 33

creative destruction replacement of features with a smaller substitution or abstraction by adaptation-guided feature deletion which makes case contents more accessible to an adaptation rule of limited power and thereby improves problem-solving competence (Leake & Schack, 2016). 15, 45

curse of dimensionality the problem of high dimensionality in a problem space leading to sparsity and high computational costs (Köppen, 2000). 18, 89

data augmentation mitigation for overfitting employed by artificial neural networks which perturbs training data in order to supplement it with additional instances (e.g. cropping or deforming images) (Wong et al., 2016). 16

decision tree learning method of machine learning which constructs a tree from training data where each internal node tests an attribute and each leaf node draws a conclusion (Quinlan, 1986; Safavian & Landgrebe, 1991). 5

difference heuristic method of learning an adaptation rule which computes the corresponding differences between the problem and solution parts of a pair of cases — effectively transferring knowledge from the case base to the solution transformation container (Hanney & Keane, 2005; McSherry, 2006). 4

dynamic memory an open-ended model of learning which uses reminders to explain failures of expectations instead of categories to classify knowledge (Schank, 1982, 1999). 2

expansion-contraction compression (ECC) case-base maintenance strategy which explores unseen regions of the problem space using adaptation knowledge to generate ghost cases and then exploits the ghost cases to broaden the range of cases available for competence-based deletion (Leake & Schack, 2018; Schack, 2019). 16, 62

explainable artificial intelligence a white box model of artificial intelligence which provides human-interpretable justification for its methods and results (Core et al., 2006; Gunning & Aha, 2019; Leake & McSherry, 2005). 5

explanation-based learning method of machine learning which generalizes from instances by exploiting a strong domain theory to analyze why each specific instance exemplifies a broader concept (Minton et al., 1989). 5

feature-centric recoverability the ability to adapt either other cases in the case base or the original case revised by flexible feature deletion to cover the competence contributions of that case (which may require adapting an internal subpart of the case) (Leake & Schack, 2016; Schack, 2016). 49

flexible feature deletion (FFD) knowledge-light case-base maintenance strategy which, in contrast to per-case strategies, subdivides variable-size cases for deletion of their components (Leake & Schack, 2015; Schack & Summers, 2017). 13, 27, 44

four-element comparison standard schema for analogical reasoning which follows the syntactic form "W : X :: Y: Z" (Collins & Burstein, 1987). 4

ghost case case generated by the expansion-contraction case-base maintenance strategy exploring unseen regions of the problem space using adaptation knowledge and which broadens the range of cases available for competence-based deletion to exploit (Leake & Schack, 2018). 16, 62

Integrated Partial Parser (IPP) an early case-based reasoning system which read and generalized from news stories using memory-based parsing techniques (Lebowitz, 1983). 2

k-nearest neighbors algorithm (kNN) instance-based method of classification and regression for determining the label of an instance depending on the majority vote or average of the labels of its neighbors without maintaining any abstractions (Aha et al., 1991; Kramer, 2013). 6

knowledge container storage module of a case-based reasoning system containing knowledge which supports an aspect of the case-based reasoning cycle such as the vocabulary, case base, similarity, and solution transformations (Richter, 2003). 2

knowledge maintenance the process of updating, revising, and managing knowledge representations in a knowledge-based system to ensure that the knowledge remains relevant, accurate, and effective over time, by adding new knowledge, removing outdated or irrelevant knowledge, and modifying existing knowledge to improve its quality and usability (Menzies, 1999). 7

local recoverability the ability of a case to solve the problem that it originally solved before applying flexible feature deletion but not necessarily all of the problems in the original competence contribution of that case (Leake & Schack, 2016; Schack, 2016). 49

machine learning field of artificial intelligence which studies how to develop computer software that can progressively improve with experience and without explicit programming for a specific task (Mitchell, 1997). 4

overfitting phenomenon where a statistical model or machine learning algorithm makes predictions based on peculiarities in its training data not reflected in its testing data thereby improving performance on the training data and sacrificing performance on the testing data (Dietterich, 1995). 16

predictive case discovery methods that identify and anticipate problem-distribution drift to guide the selection of cases to request from an external source (Leake & Schack, 2023; Schack, 2023). 19, 83

problem-distribution drift change in the distribution of problems to solve which violates problem-distribution regularity (Leake & Schack, 2023; Schack, 2023). 86

problem-distribution regularity the property that the distribution of future problems will tend to reflect that of past problems, such that accumulated stored cases from past episodes tend to provide useful information for future problems (Leake & Wilson, 1999). 84

problem-solution regularity the property that solutions to similar problems will provide a useful starting point for solving a new problem (Leake & Wilson, 1999). 83

reachability approximation calculated by case-base maintenance strategies of the number of neighboring cases that can solve a problem through adaptation (Smyth & Keane, 1995). 8, 33, 49

recoverability measure of the ability of adaptation knowledge applied to other features to restore a feature considered for deletion (Leake & Schack, 2016; Schack, 2016). 14

representativeness assumption proposition underlying most case-base maintenance strategies stating that future problems encountered will follow the same distribution within the domain as existing cases in the case base (Smyth, 2005). 8, 61, 85

retention phase final phase of the case-based reasoning cycle which packages the given problem and revised solution as a new case which it may or may not store for solving future problems (de Mantaras et al., 2005). 2

retrieval phase first phase of the case-based reasoning cycle which searches the case base for other problems similar to the given problem (de Mantaras et al., 2005). 2

reuse phase second phase of the case-based reasoning cycle which identifies the differences between the given problem and the retrieved case in order to adapt the solution to the retrieved case to account for those differences (de Mantaras et al., 2005). 2

revision phase third phase of the case-based reasoning cycle which attempts to apply the adapted solution in a simulation or the real world and modifies the solution based on its performance (de Mantaras et al., 2005). 2

rule induction area of machine learning which extracts formal rules which model patterns in training examples (Cohen, 1995). 6

similarity measure knowledge container of a case-based reasoning system which contains the function for calculating of the resemblance between problems perhaps with different features having different weights (Richter, 2003). 4

solution transformation container knowledge container of a case-based reasoning system which stores potential adaptations to a solution to account for differences between problems (Richter, 2003). 4

swamping utility problem the trade-off between the contribution to competence against the cost for retrieval of storing a case in a case base (Francis & Ram, 1993; Minton, 1990; Smyth & Cunningham, 2005). 7

transfer learning area of machine learning which applies knowledge gained from solving a source problem to solve a different target problem in a different domain (Sharma et al., 2007). 5

vocabulary container knowledge container of a case-based reasoning system which stores the potential features of a problem or solution, the range of potential values, and the relationship between dependent features (Richter, 2003). 4

Bibliography

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, *7*(1), 39–59. https://doi.org/10.3233/AIC-1994-7104

Abdel-Aziz, A., & Hüllermeier, E. (2015). Case base maintenance in preference-based CBR. *Case-Based Reasoning Research and Development*, *9343*, 1–14. https://doi.org/10.1007/978-3-319-24586-7_1

AbuSalim, S. W., Ibrahim, R., Saringat, M. Z., Jamel, S., & Wahab, J. A. (2020). Comparative analysis between Dijkstra and Bellman-Ford algorithms in shortest path optimization. *Materials Science and Engineering*, *917*, 1–11. https://doi.org/10.1088/1757-899X/917/1/012077

Aha, D. (1992). Generalizing from case studies: A case study. *Machine Learning Proceedings*, 1–10. https://doi.org/10.1016/B978-1-55860-247-2.50006-1

Aha, D., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37–66. https://doi.org/10.1007/BF00153759

Alippi, C., & Roveri, M. (2008). Just-in-time adaptive classifiers, part I: Detecting nonstationary changes. *IEEE Transactions on Neural Networks*, *19*(7), 1145–1153. https://doi.org/10.1109/TNN.2008.2000082

Angiulli, F. (2005). Fast condensed nearest neighbor rule. *Proceedings of the 22nd International Conference on Machine Learning*, 25–32. https://doi.org/10.1145/1102351.1102355

Anwar, M. A., & Yoshida, T. (2001). Integrating OO road network database, cases, and knowledge for route finding. *Proceedings of the 2001 ACM Symposium on Applied Computing*, 215–219. https://doi.org/10.1145/372202.372325

Arshadi, N., & Jurisica, I. (2005). Feature selection for improving case-based classifiers on high-dimensional data sets. *Proceedings of the FLAIRS Conference*, 99–104. Retrieved November

13, 2023, from https://www.researchgate.net/profile/Niloofar-Arshadi/publication/221438418_Feature_Selection_for_Improving_Case-Based_Classifiers_on_High-Dimensional_Data_Sets/links/5486f9bb0cf289302e2eaf46/Feature-Selection-for-Improving-Case-Based-Classifiers-on-High-Dimensional-Data-Sets.pdf

Ashley, K. D. (1986, June 30). *Modeling legal argument: Reasoning with cases and hypotheticals.* Defense Technical Information Center. Retrieved November 13, 2023, from https://apps.dtic.mil/sti/pdfs/ADA175925.pdf

Bergmann, R., & Wilke, W. (2005). On the role of abstraction in case-based reasoning. *Advances in Case-Based Reasoning*, *1168*, 28–43. https://doi.org/10.1007/BFb0020600

Bichindaritz, I., & Marling, C. (2006). Case-based reasoning in the health sciences: What's next? *Artificial Intelligence in Medicine*, *36*(2), 127–135. https://doi.org/10.1016/j.artmed.2005.10.008

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443–448. https://doi.org/10.1137/1.9781611972771

Brighton, H. (2001). Identifying competence-critical instances for instance-based learners. In *Instance selection and construction for data mining* (pp. 1–18, Vol. 608). Springer Link. Retrieved November 13, 2023, from https://eucaslab.github.io/downloads/2001.Brighton.Mellish.ISDM.pdf

Burstein, M. H. (1989). Analogy vs. CBR: The purpose of mapping. *Proceedings of the DARPA Case-Based Reasoning Workshop*, 133–136. Retrieved November 13, 2023, from https://www.bursteins.net/mark/docs/Burstein-CBR89.pdf

Carbonell, J. (1985, March 5). *Derivational analogy: A theory of reconstructive problem solving and expertise acquisition.* Carnegie-Mellon University Department of Computer Science.

Pittsburgh PA. Retrieved November 13, 2023, from https://apps.dtic.mil/sti/pdfs/ADA156817.pdf

Cheah, Y.-W., Plale, B., Kendall-Morwick, J., Leake, D., & Kamakrishnan, L. (2011). A noisy 10 GB provenance database. *Business Process Management Workshops*, *100*. https://doi.org/10.1007/978-3-642-28115-0_35

Cohen, W. W. (1995). Fast effective rule induction. *Proceedings of the Twelfth International Conference on Machine Learning*, 115–123. https://doi.org/10.1016/B978-1-55860-377-6.50023-2

Collins, A., & Burstein, M. (1987, December 1). *A framework for a theory of mapping.* BBN Labs Inc. Cambridge MA. Retrieved November 13, 2023, from https://apps.dtic.mil/sti/pdfs/ADA191071.pdf

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th International Conference on Machine Learning*, 160–167. https://doi.org/10.1145/1390156.1390177

Cordier, A., Lefevre, M., Champin, P.-A., Georgeon, O., & Mille, A. (2013). Trace-based reasoning: Modeling interaction traces for reasoning on experiences, 1–15. Retrieved November 13, 2023, from https://hal.science/hal-00830444/file/Liris-5264.pdf

Core, M. G., Lane, H. C., Lent, M. v., Gomboc, D., Solomon, S., & Rosenberg, M. (2006, January 1). *Building explainable artificial intelligence systems.* University of Southern California. Marina del Rey CA. Retrieved November 13, 2023, from https://cdn.aaai.org/AAAI/2006/AAAI06-293.pdf

Craw, S. (2011). Case-based reasoning. In *Encyclopedia of machine learning* (pp. 147–154). Springer Link. Retrieved November 20, 2023, from https://link.springer.com/content/pdf/10.1007/978-0-387-30164-8_97.pdf?pdf=inline%20link

Craw, S., Massie, S., & Wiratunga, N. (2007). Informed case base maintenance: A complexity profiling approach. *Proceedings of the 22nd National Conference on Artificial Intelligence*, *2*, 1618–1621. Retrieved November 16, 2023, from https://cdn.aaai.org/AAAI/2007/AAAI07-258.pdf

Cunningham, P., Nowlan, N., Delany, S. J., & Haahr, M. (2003). A case-based approach to spam filtering that can track concept drift. *Proceedings of the International Conference on Case-Based Reasoning*, *3*. Retrieved November 20, 2023, from https://publications.scss.tcd.ie/tech-reports/reports.03/TCD-CS-2003-16.pdf

d'Aquin, M., Lieber, J., & Napoli, A. (2006). Adaptation knowledge acquisition: A case study for case-based decision support in oncology. *Computational Intelligence*, *22*(3), 161–176. https://doi.org/10.1111/j.1467-8640.2006.00281.x

Dasu, T., Krishnan, S., & Venkatasubramanian, S. (2006). An information theoretic approach to detecting changes in multi-dimensional data streams. *Proceedings of the Symposium on the Interface of Statistics, Computing Science, and Applications*, 1–24. Retrieved November 20, 2023, from https://www.cse.ust.hk/~yike/datadiff/datadiff.pdf

Delany, S. J., & Cunningham, P. (2004). An analysis of case-base editing in a spam filtering system. *Advances in Case-Based Reasoning*, *3155*, 128–141. https://doi.org/10.1007/978-3-540-28631-8_11

Delany, S. J., Cunningham, P., Tsymbal, A., & Coyle, L. (2005). A case-based technique for tracking concept drift in spam filtering. *Applications and Innovations in Intelligent Systems XII*, 3–16. https://doi.org/10.1007/1-84628-103-2_1

de Mantaras, R. L., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M. L., Cox, M. T., Forbus, K., Keane, M., Aamodt, A., & Watson, I. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, *20*(3), 215–240. https://doi.org/10.1017/S0269888906000646

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, *27*(3), 326–327. Retrieved November 20, 2023, from https://dl.acm.org/doi/pdf/10.1145/212094.212114

Fernández, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, *61*, 863–905. https://doi.org/10.1613/jair.1.11192

Forbus, K. D., Gentner, D., & Law, K. (1995). MAC / FAC: A model of similarity-based retrieval. *Cognitive Science*, *19*(2), 141–205. https://doi.org/10.1207/s15516709cog1902_1

Fox, S., & Leake, D. B. (2005). Learning to refine indexing by introspective reasoning. *Case-Based Reasoning Research and Development*, *1010*, 431–440. https://doi.org/10.1007/3-540-60598-3_39

Francis, A. G., Jr., & Ram, A. (1993). Computational models of the utility problem and their application to a utility analysis of case-based reasoning. *Proceedings of the Workshop on Knowledge Compilation and Speedup Learning*, 1–8. Retrieved November 20, 2023, from https://citeseerx.ist.psu.edu/document?doi=c204a97e59adfd8d1a6d14dd803d8ddc0c59b75d

Francis, A. G., Jr., & Ram, A. (1995). Comparative utility analysis of case-based reasoning and control-rule learning systems. *Machine Learning: ECML-95*, *912*, 138–150. https://doi.org/10.1007/3-540-59286-5_54

Fuchs, B., Lieber, J., Mille, A., & Napoli, A. (2014). Differential adaptation: An operational approach to adaptation for solving numerical problems with CBR. *Knowledge-Based Systems*, *68*, 103–114. https://doi.org/10.1016/j.knosys.2014.03.009

Goel, A. K., Ali, K. S., & de Silva Garza, A. G. (1994). Computational trade-offs in experience-based reasoning. *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning*, 55–61. Retrieved November 20, 2023, from https://www-robotics.jpl.nasa.gov/media/documents/Computational_Trade-offs_in_Experience_Based_Reasoning.pdf

Goel, A. K., & Diaz-Agudo, B. (2017). What's hot in case-based reasoning. *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, *31*, 5067–5069. https://doi.org/10.1609/aaai.v31i1.10643

Gunning, D., & Aha, D. W. (2019). DARPA's explainable artificial intelligence program. *AI Magazine*, *40*(2), 44–58. https://doi.org/10.1609/aimag.v40i2.2850

Haigh, K. Z., & Veloso, M. (2005). Route planning by analogy. *Case-Based Reasoning Research and Development*, *1010*, 169–180. https://doi.org/10.1007/3-540-60598-3_16

Hanney, K., & Keane, M. T. (2005). Learning adaptation rules from a case base. *Advances in Case-Based Reasoning*, *1168*, 179–192. https://doi.org/10.1007/BFb0020610

Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, *14*(3), 515–516. Retrieved November 20, 2023, from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7c3771fd6829630cf450af853df728ecd8da4ab2

Haykin, S. (2004). *Neural networks: A comprehensive foundation* (2nd ed.). Prentice Hall. Retrieved November 20, 2023, from https://dl.acm.org/doi/abs/10.5555/521706

Hills, T. T., Todd, P. M., Lazer, D., Redish, A. D., Couzin, I. D., & Cognitive Search Research Group. (2014). Exploration versus exploitation in space, mind, and society. *Trends in Cognitive Sciences*, *19*(1), 46–54. https://doi.org/10.1016/j.tics.2014.10.004

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine*, *29*(6), 82–87. https://doi.org/10.1109/MSP.2012.2205597

Hodál, J., & Dvořák, J. (2008). Using case-based reasoning for mobile robot path planning. *Engineering Mechanics*, *15*(3), 181–191. Retrieved November 20, 2023, from http://www.engineeringmechanics.cz/pdf/15_3_181.pdf

Hoens, T. R., Polikar, R., & Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: An overview. *Progress in Artificial Intelligence*, *1*, 89–101. https://doi.org/10.1007/s13748-011-0008-0

Holt, A., Bichindaritz, I., Schmidt, R., & Perner, P. (2005). Medical applications in case-based reasoning. *The Knowledge Engineering Review*, *20*(3), 289–292. https://doi.org/10.1017/S0269888906000622

Houeland, T. G., & Aamodt, A. (2010). The utility problem for lazy learners: Towards a non-eager approach. *Case-Based Reasoning Research and Development*, *6176*, 141–155. https://doi.org/10.1007/978-3-642-14274-1_12

Institute for Health Technology Transformation. (2013). *Transforming health care through big data: Strategies for leveraging big data in the health care industry.* New York NY.

Iwana, B. K., & Uchida, S. (2021). An empirical survey of data augmentation for time series classification with neural networks. *PLoS One*, *16*(7), 1–32. https://doi.org/10.1371/journal.pone.0254841

Jalali, V., & Leake, D. (2013). Extending case adaptation with automatically generated ensembles of adaptation rules. *Case-Based Reasoning Research and Development*, *7969*, 188–202. https://doi.org/10.1007/978-3-642-39056-2_14

Jalali, V., & Leake, D. (2018). Harnessing hundreds of millions of cases: Case-based prediction at industrial scale. *Case-Based Reasoning Research and Development*, *11156*, 153–169. https://doi.org/10.1007/978-3-030-01081-2_11

Jalali, V., Leake, D., & Forouzandehmehr, N. (2016). Ensemble of adaptations for classification: Learning adaptation rules for categorical features. *Case-Based Reasoning Research and Development*, *9969*, 186–202. https://doi.org/10.1007/978-3-319-47096-2_13

Jonsen, A. R., & Toulmin, S. (1988). *The abuse of casuistry: A history of moral reasoning*. University of California Press. Retrieved November 20, 2023, from https://philpapers.org/rec/JONTAO-19

Juarez, J. M., Craw, S., Lopez-Delgado, J. R., & Campos, M. (2018). Maintenance of case bases: Current algorithms after fifty years. *Proceedings of the 27th International Joint Conference on Artificial Intelligence and 23rd European Conference on Artificial Intelligence*, 5457–5463. https://doi.org/10.24963/ijcai.2018/770

Kantchelian, A., Afroz, S., Huang, L., Islam, A. C., Miller, B., Tschantz, M. C., Greenstadt, R., Joseph, A. D., & Tygar, J. D. (2013). Approaches to adversarial drift. *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security*, 99–109. https://doi.org/10.1145/2517312.2517320

Kolodner, J. (1983). Reconstructive memory: A computer model. *Cognitive Science*, *7*(4), 281–328. https://doi.org/10.1016/S0364-0213(83)80002-0

Kolodner, J. (1993). *Case-based reasoning*. Morgan Kaufmann Publishers. Retrieved November 20, 2023, from https://folk.idi.ntnu.no/agnar/CBR%20papers/Kolodner-bok/Kolodner93_FirstPages.pdf

Kolodner, J. (1996). Making the implicit explicit: Clarifying the principles of case-based reasoning. In *Case-based reasoning: Experiences, lessons, and future directions* (pp. 349–370). MIT Press. Retrieved November 20, 2023, from https://homes.luddy.indiana.edu/leake/papers/a-96-book.html

Köppen, M. (2000). The curse of dimensionality. *Proceedings of the 5th Online World Conference on Soft Computing in Industrial Applications*, *1*, 4–8. Retrieved November 20, 2023, from https://www.class-specific.com/csf/papers/hidim.pdf

Kramer, O. (2013). K-nearest neighbors. In *Dimensionality reduction with unsupervised nearest neighbors* (pp. 13–23, Vol. 51). Springer Link. Retrieved November 20, 2023, from https://link.springer.com/content/pdf/10.1007/978-3-642-38652-7_2.pdf?pdf=inline%20link

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1–9. Retrieved November 20, 2023, from https://www.nvidia.it/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf

Kruusmaa, M., & Willemson, J. (2003). Covering the path space: A casebase analysis for mobile robot path planning. *Knowledge-Based Systems*, *16*(5), 235–242. https://doi.org/10.1016/S0950-7051(03)00024-8

Lawrence, S., Giles, C. L., & Tsoi, A. C. (1997). Lessons in neural network training: Overfitting may be harder than expected. *Proceedings of the National Conference on Artificial Intelligence*, 540–545. Retrieved November 20, 2023, from https://www.researchgate.net/profile/Steve-Lawrence-4/publication/221604796_Lessons_in_Neural_Network_Training_Overfitting_May_be_Harder_than_Expected/links/00b7d531e476d171be000000/Lessons-in-Neural-Network-Training-Overfitting-May-be-Harder-than-Expected.pdf

Leake, D. B. (1992). Constructive similarity assessment: Using stored cases to define new situations. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 313–318. Retrieved November 21, 2023, from https://citeseerx.ist.psu.edu/document?doi=5be198a38bcd175b27bd91a425424d23c922a3f0

Leake, D. B. (1996a). *Case-based reasoning: Experiences, lessons, and future directions*. Retrieved November 20, 2023, from https://dl.acm.org/doi/abs/10.5555/524680

Leake, D. B. (1996b). CBR in context: The present and future. In *Case-based reasoning: Experiences, lessons, and future directions* (pp. 3–30). AAAI Press / MIT Press. Retrieved November 21, 2023, from https://citeseerx.ist.psu.edu/document?doi=f65acdb27d92dfcf44b240632eeb5a913a48622

Leake, D. B. (1998). Cognition as case-based reasoning. In *A companion to cognitive science* (pp. 465–476). Blackwell.

Leake, D. B., Kinley, A., & Wilson, D. (1996). Linking adaptation and similarity learning. *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, 591–596. Retrieved November 21, 2023, from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=faec91c612f88b089e3adc27ae293af632ec54ed

Leake, D. B., Kinley, A., & Wilson, D. (1997). Learning to integrate multiple knowledge sources for case-based reasoning. *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, *1*, 246–251. Retrieved November 21, 2023, from https://citeseerx.ist.psu.edu/document?doi=b1a1ff47208f3fa86b60ddffc4864883e74b2af5

Leake, D. B., Maguitman, A., & Reichherzer, T. (2014). Experience-based support for human-centered knowledge modeling. *Knowledge-Based Systems*, *68*, 77–87. https://doi.org/10.1016/j.knosys.2014.01.013

Leake, D. B., Maguitman, A., Reichherzer, T., Cañas, A. J., Carvalho, M., Arguedas, M., Brenes, S., & Eskridge, T. (2003). Aiding knowledge capture by searching for extensions of knowledge models. *Proceedings of the 2nd International Conference on Knowledge Capture*, 44–53. https://doi.org/10.1145/945645.945655

Leake, D. B., & McSherry, D. (2005). Introduction to the special issue on explanation in case-based reasoning. *The Artificial Intelligence Review*, *24*(2), 103–108. https://doi.org/10.1007/s10462-005-4606-8

Leake, D. B., & Schack, B. (2015). Flexible feature deletion: Compacting case bases by selectively compressing case contents. *Case-Based Reasoning Research and Development*, *9343*, 212–227. https://doi.org/10.1007/978-3-319-24586-7_15

Leake, D. B., & Schack, B. (2016). Adaptation-guided feature deletion: Testing recoverability to guide case compression. *Case-Based Reasoning Research and Development*, *9969*, 234–248. https://doi.org/10.1007/978-3-319-47096-2_16

Leake, D. B., & Schack, B. (2018). Exploration vs. exploitation in case-base maintenance: Leveraging competence-based deletion with ghost cases. *Case-Based Reasoning Research and Development*, *11156*, 202–218. https://doi.org/10.1007/978-3-030-01081-2_14

Leake, D. B., & Schack, B. (2023). Towards addressing problem-distribution drift with case discovery. *Case-Based Reasoning Research and Development*, *14141*, 244–259. https://doi.org/10.1007/978-3-031-40177-0_16

Leake, D. B., & Sooriamurthi, R. (2002). Managing multiple case bases: Dimensions and issues. *Proceedings of the 15th International Florida Artificial Intelligence Research Society Conference*, 106–110. Retrieved November 21, 2023, from https://legacy.cs.indiana.edu/ftp/leake/p-02-03.pdf

Leake, D. B., & Whitehead, M. (2007). Case provenance: The value of remembering case sources. *Case-Based Reasoning Research and Development*, *4626*, 194–208. https://doi.org/10.1007/978-3-540-74141-1_14

Leake, D. B., & Wilson, D. C. (1999). When experience is wrong: Examining CBR for changing tasks and environments. *Case-Based Reasoning Research and Development*, *1650*, 218–232. https://doi.org/10.1007/3-540-48508-2_16

Leake, D. B., & Wilson, D. C. (2003). Remembering why to remember: Performance-guided case-base maintenance. *Advances in Case-Based Reasoning*, *1898*, 161–172. https://doi.org/10.1007/3-540-44527-7_15

Leake, D. B., & Wilson, M. (2011). How many cases do you need? Assessing and predicting case-base coverage. *Case-Based Reasoning Research and Development*, *6880*, 92–106. https://doi.org/10.1007/978-3-642-23291-6_9

Leake, D. B., & Ye, X. (2021). Harmonizing case retrieval and adaptation with alternating optimization. *Case-Based Reasoning Research and Development*, *12877*, 125–139. https://doi.org/10.1007/978-3-030-86957-1_9

Lebowitz, M. (1983). Memory-based parsing. *Artificial Intelligence*, *21*(4), 363–404. https://doi.org/10.1016/S0004-3702(83)80019-8

Li, Y., Shiu, S. C., & Pal, S. K. (2006). Combining feature reduction and case selection in building CBR classifiers. *IEEE Transactions on Knowledge and Data Engineering*, *18*(3), 415–429. https://doi.org/10.1109/TKDE.2006.40

Li, Y., Su, L.-M., & He, Q. (2012). Case-based multi-task pathfinding algorithm. *Proceedings of the 2012 International Conference on Machine Learning and Cybernetics*, 513–518. https://doi.org/10.1109/ICMLC.2012.6358976

Lieber, J. (2005). A criterion of comparison between two case bases. *Advances in Case-Based Reasoning*, *984*, 87–100. https://doi.org/10.1007/3-540-60364-6_29

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

Lu, N., Lu, J., Zhang, G., & de Mantaras, R. L. (2016). A concept drift-tolerant case-base editing technique. *Artificial Intelligence*, *230*, 108–133. Retrieved November 21, 2023, from https://www.iiia.csic.es/~mantaras/AIJ_Concept_Drift.pdf

Lu, N., Zhang, G., & Lu, J. (2014). Concept drift detection via competence models. *Artificial Intelligence*, *209*, 11–28. https://doi.org/10.1016/j.artint.2014.01.001

Lupiani, E., Craw, S., Massie, S., Juarez, J. M., & Palma, J. T. (2013). A multi-objective evolutionary algorithm fitness function for case-base maintenance. *Case-Based Reasoning Research and Development*, *7969*, 218–232. https://doi.org/10.1007/978-3-642-39056-2_16

Lupiani, E., Juarez, J. M., & Palma, J. (2014). A proposal of temporal case-base maintenance algorithms. *Case-Based Reasoning Research and Development*, *8765*, 260–273. https://doi.org/10.1007/978-3-319-11209-1_19

Lupiani, E., Massie, S., Craw, S., Juarez, J. M., & Palma, J. (2015). Case-base maintenance with multi-objective evolutionary algorithms. *Journal of Intelligent Information Systems*, *46*, 259–284. https://doi.org/10.1007/s10844-015-0378-z

Marquer, E., Badra, F., Lesot, M.-J., Couceiro, M., & Leake, D. (2023). Less is better: An energy-based approach to case-base competence. *Proceedings of the ICCBR ATA '23: Workshop on Analogies*, 27–42. Retrieved November 22, 2023, from https://inria.hal.science/hal-04184905/document

Massie, S., Craw, S., & Wiratunga, N. (2005). Complexity-guided case discovery for case-based reasoning. *Proceedings of the 20th National Conference on Artificial Intelligence*, *1*, 216–221. Retrieved November 22, 2023, from https://cdn.aaai.org/AAAI/2005/AAAI05-035.pdf

Mathew, D., & Chakraborti, S. (2017). Competence guided model for casebase maintenance. *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4904–4908. Retrieved November 22, 2023, from https://www.ijcai.org/proceedings/2017/0691.pdf

McDonnell, N. (2006, October). *A knowledge-light approach to regression using case-based reasoning* [Doctoral dissertation, University of Dublin, Trinity College]. Retrieved November 22, 2023, from http://www.tara.tcd.ie/bitstream/handle/2262/77638/McDonnell,%20Neil_TCD-SCSS-PHD-2006-04.pdf?sequence=1&isAllowed=y

McKenna, E., & Smyth, B. (2002). Competence-guided case discovery. *Research and Development in Intelligent Systems XVIII*, 97–108. https://doi.org/10.1007/978-1-4471-0119-2_8

McSherry, D. (2000). Automating case selection in the construction of a case library. *Research and Development in Intelligent Systems XVI*, 163–177. https://doi.org/10.1007/978-1-4471-0745-3_11

McSherry, D. (2002). Intelligent case-authoring support in CaseMaker-2. *Computational Intelligence*, *17*(2), 331–345. https://doi.org/10.1111/0824-7935.00148

McSherry, D. (2006). An adaptation heuristic for case-based estimation. *Advances in Case-Based Reasoning*, *1488*, 184–195. https://doi.org/10.1007/BFb0056332

Menzies, T. (1999). Knowledge maintenance: The state of the art. *The Knowledge Engineering Review*, *14*(1), 1–46. https://doi.org/10.1017/S0269888999134052

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*(2), 564–569. https://doi.org/10.1016/0004-3702(90)90059-9

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., & Gil, Y. (1989). Explanation-based learning: A problem-solving perspective. *Artificial Intelligence*, *40*(1), 63–118. https://doi.org/10.1016/0004-3702(89)90047-7

Mitchell, T. M. (1997, March 1). *Machine learning*. McGraw-Hill Science / Engineering / Math. Retrieved November 22, 2023, from https://ds.amu.edu.et/xmlui/bitstream/handle/123456789/14637/Machine_Learning%20-%20421%20pages.pdf?sequence=1&isAllowed=y

Muñoz-Avila, H. (1999). A case retention policy based on detrimental retrieval. *Case-Based Reasoning Research and Development*, *1650*, 276–287. https://doi.org/10.1007/3-540-48508-2_20

Muñoz-Avila, H. (2002). Case-base maintenance by integrating case-index revision and case-retention policies in a derivational replay framework. *Computational Intelligence*, *17*(2), 280–294. https://doi.org/10.1111/0824-7935.00145

Niwattanakul, S., Singthongchai, J., Naenudorn, E., & Wanapu, S. (2013). Using of Jaccard coefficient for keywords similarity. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, *1*, 380–384. Retrieved November 22, 2023, from https://www.iaeng.org/publication/IMECS2013/IMECS2013_pp380-384.pdf

Novak, J. D., Gowin, D. B., & Kahle, J. B. (1984). *Learning how to learn*. Cambridge University Press. Retrieved November 22, 2023, from https://cir.nii.ac.jp/crid/1363388846319416576

Oh, K. J., & Kim, T. Y. (2007). Financial market monitoring by case-based reasoning. *Expert Systems with Applications*, *32*(3), 789–800. https://doi.org/10.1016/j.eswa.2006.01.044

Ontañón, S., & Plaza, E. (2003). Collaborative case retention strategies for CBR agents. *Case-Based Reasoning Research and Development*, *2689*, 392–406. https://doi.org/10.1007/3-540-45006-8_31

Owrang O., M. M. (1998). Case discovery in case-based reasoning systems. *Information Systems Management*, *15*(1), 74–78. https://doi.org/10.1201/1078/43183.15.1.19980101/31107.12

Plaza, E., & McGinty, L. (2005). Distributed case-based reasoning. *The Knowledge Engineering Review*, *20*(3), 261–265. https://doi.org/10.1017/S0269888906000683

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106. https://doi.org/10.1007/BF00116251

Racine, K., & Yang, Q. (2005). Maintaining unstructured case bases. *Case-Based Reasoning Research and Development*, *1266*, 553–564. https://doi.org/10.1007/3-540-63233-6_524

Richter, M. M. (2003). Knowledge containers. *Readings in Case-Based Reasoning*. Retrieved November 22, 2023, from https://www.researchgate.net/profile/Michael-Richter-4/publication/225070310_Knowledge_Containers/links/00b7d539b22aa7bc95000000/Knowledge-Containers.pdf

Richter, M. M., & Aamodt, A. (2005). Case-based reasoning foundations. *The Knowledge Engineering Review*, *20*(3), 203–207. https://doi.org/10.1017/S0269888906000695

Riesbeck, C. K., & Schank, R. C. (2013, May 13). *Inside case-based reasoning*. Taylor & Francis. Retrieved November 22, 2023, from https://www.google.com/books/edition/Inside_Case_Based_Reasoning/tPxvuJDcPpEC

Rissland, E., Ashley, K. D., & Branting, L. K. (2005). Case-based reasoning and law. *The Knowledge Engineering Review*, *20*(3), 293–298. https://doi.org/10.1017/S0269888906000701

Romdhane, H., & Lamontagne, L. (2008). Forgetting reinforced cases. *Advances in Case-Based Reasoning*, *5239*, 474–486. https://doi.org/10.1007/978-3-540-85502-6_32

Ross, B. H. (1989). Some psychological results on case-based reasoning. *Proceedings of the Case-Based Reasoning Workshop*, 144–147. Retrieved November 22, 2023, from https://cir.nii.ac.jp/crid/1570854174757235968

Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, *21*(3), 660–674. https://doi.org/10.1109/21.97458

Salamó, M., & López-Sánchez, M. (2011a). Rough set based approaches to feature selection for case-based reasoning classifiers. *Pattern Recognition Letters*, *32*(2), 280–292. https://doi.org/10.1016/j.patrec.2010.08.013

Salamó, M., & López-Sánchez, M. (2011b). Adaptive case-based reasoning using retention and forgetting strategies. *Knowledge-Based Systems*, *24*(2), 230–247. https://doi.org/10.1016/j.knosys.2010.08.003

Samek, W., Wiegand, T., & Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. https://doi.org/10.48550/arXiv.1708.08296

Schack, B. (2016). Feature-centric approaches to case-base maintenance. *Proceedings of the ICCBR 2016 Workshops*, 287–291. Retrieved November 22, 2023, from https://ceur-ws.org/Vol-1815/paper32.pdf

Schack, B. (2019). Case-base maintenance beyond case deletion. *Proceedings of the ICCBR Workshops*, 191–195. Retrieved November 22, 2023, from https://ceur-ws.org/Vol-2567/paper20.pdf

Schack, B. (2023). Feature deletion and case discovery in case-base maintenance. *Proceedings of the Doctoral Consortium at ICCBR 2023*, 1–6. Retrieved November 22, 2023, from https://ceur-ws.org/Vol-3438/paper_20.pdf

Schack, B., & Summers, R. (2017). Flexible feature deletion: Companion video. *Proceedings of the ICCBR 2017 Workshops*, 293. Retrieved November 22, 2023, from https://ceur-ws.org/Vol-2028/paper36.pdf

Schaller, R. R. (1997). Moore's law: Past, present, and future. *IEEE Spectrum*, *34*(6), 52–59. https://doi.org/10.1109/6.591665

Schank, R. C. (1982). *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge University Press. Retrieved November 22, 2023, from https://cir.nii.ac.jp/crid/1130000796887155328

Schank, R. C. (1999). *Dynamic memory revisited*. Cambridge University Press. Retrieved November 22, 2023, from https://cir.nii.ac.jp/crid/1362825895335459712

Schank, R. C., & Leake, D. B. (1989). Creativity and learning in a case-based explainer. *Artificial Intelligence*, *40*(1), 353–385. https://doi.org/10.1016/0004-3702(89)90053-2

Searing, E. A. (2009). *Casuistry: The tape measure in the cognitive toolbox*. Retrieved November 25, 2023, from https://www.academia.edu/21855788/Casuistry_The_Tape_Measure_in_the_Cognitive_Toolbox

Sharma, M., Holmes, M., Santamaria, J., Irani, A., Isbell, C., & Ram, A. (2007). Transfer learning in real-time strategy games using hybrid CBR / RL. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1041–1046. Retrieved November 25, 2023, from https://www.ijcai.org/Proceedings/07/Papers/168.pdf

Shokouhi, S. V. (2012). An overview of case-based reasoning applications in drilling engineering. *Artificial Intelligence Review*, *41*, 317–329. https://doi.org/10.1007/s10462-011-9310-2

Smiti, A., & Elouedi, Z. (2014). WCOID-DG: An approach for case base maintenance based on weighting, clustering, outliers, internal detection and Dbsan-Gmeans. *Journal of Computer and System Sciences*, *80*(1), 27–38. https://doi.org/10.1016/j.jcss.2013.03.006

Smiti, A., & Elouedi, Z. (2018). SCBM: Soft case base maintenance method based on competence model. *Journal of Computational Science*, *25*, 221–227. https://doi.org/10.1016/j.jocs.2017.09.013

Smyth, B. (2005). Case-base maintenance. *Tasks and Methods in Applied Artificial Intelligence*, *1416*, 507–516. https://doi.org/10.1007/3-540-64574-8_436

Smyth, B., & Cunningham, P. (2005). The utility problem analysed: A case-based reasoning perspective. *Advances in Case-Based Reasoning*, *1168*, 392–399. https://doi.org/10.1007/BFb0020625

Smyth, B., & Keane, M. T. (1995). Remembering to forget. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, *1*, 377–382. Retrieved November 25, 2023, from https://folk.idi.ntnu.no/agnar/CBR%20papers/smyth-keane-remembering-95.pdf

Smyth, B., & Keane, M. T. (1998). Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, *102*(2), 249–293. https://doi.org/10.1016/S0004-3702(98)00059-9

Smyth, B., & McClave, P. (2001). Similarity vs. diversity. *Case-Based Reasoning Research and Development*, *2080*, 347–361. https://doi.org/10.1007/3-540-44593-5_25

Smyth, B., & McKenna, E. (1999a). Building compact competent case bases. *Case-Based Reasoning Research and Development*, *1650*, 329–342. https://doi.org/10.1007/3-540-48508-2_24

Smyth, B., & McKenna, E. (1999b). Footprint-based retrieval. *Case-Based Reasoning Research and Development*, *1650*, 343–357. https://doi.org/10.1007/3-540-48508-2_25

Smyth, B., & McKenna, E. (2002). Competence models and the maintenance problem. *Computational Intelligence*, *17*(2), 235–249. https://doi.org/10.1111/0824-7935.00142 van Someren, M., Surma, J., & Torasso, P. (2005). A utility-based approach to learning in a mixed case-based and model-based reasoning architecture. *Case-Based Reasoning Research and Development*, *1266*, 477–488. https://doi.org/10.1007/3-540-63233-6_517

Vasudevan, C., & Ganesan, K. (1996). Case-based path planning for autonomous underwater vehicles. *Autonomous Robots*, *3*, 79–89. https://doi.org/10.1007/BF00141149

Veloso, M. M. (1994, December 7). *Planning and learning by analogical reasoning* (1st ed., Vol. 886). Springer Link. Retrieved November 26, 2023, from https://link.springer.com/book/10.1007/3-540-58811-6

Waheed, A., & Adeli, H. (2005). Case-based reasoning in steel bridge engineering. *Knowledge-Based Systems*, *18*(1), 37–46. https://doi.org/10.1016/j.knosys.2004.06.001

Watson, I. (1999). Case-based reasoning is a methodology not a technology. *Knowledge-Based Systems*, *12*(5), 303–308. https://doi.org/10.1016/S0950-7051(99)00020-9

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*, 69–101. https://doi.org/10.1023/A:1018046501280

Wilke, W., Vollrath, I., & Bergmann, R. (1997). Using knowledge containers to model a framework for learning adaptation knowledge. *European Conference on Machine Learning Workshop Notes*, 68–75. Retrieved November 26, 2023, from https://www.wi2.uni-trier.de/shared/publications/1997_WilkeVollrathBergmannECML.pdf

Wilson, D. C., & O'Sullivan, D. (2008). Medical imagery in case-based reasoning. In *Case-based reasoning on images and signals* (pp. 389–418, Vol. 73). Springer Link. Retrieved November 26, 2023, from https://link.springer.com/chapter/10.1007/978-3-540-73180-1_13

Wilson, D. R., & Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, *38*, 257–286. https://doi.org/10.1023/A:1007626913721

Wilson, D. C., & Leake, D. B. (2002). Maintaining case-based reasoners: Dimensions and directions. *Computational Intelligence*, *17*(2), 196–213. https://doi.org/10.1111/0824-7935.00140

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). Understanding data augmentation for classification: When to warp? *Proceedings of the 2016 International Conference on Digital Image Computing: Techniques and Applications*. https://doi.org/10.1109/DICTA.2016.7797091

Yamamoto, Y., Kawabe, T., Kobayashi, Y., Tsuruta, S., Sakurai, Y., & Knauf, R. (2015). A refined case based genetic algorithm for intelligent route optimization. *Proceedings of the 11th International Conference on Signal-Image Technology & Internet-Based Systems*, 698–704. https://doi.org/10.1109/SITIS.2015.36

Zhang, Z., & Yang, Q. (2006). Towards lifetime maintenance of case base indexes for continual case based reasoning. *Artificial Intelligence: Methodology, Systems, and Applications*, *1480*, 489–500. https://doi.org/10.1007/BFb0057469

Zhu, J., & Yang, Q. (1999). Remembering to add: Competence-preserving case-addition policies for case-base maintenance. *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, *1*, 234–239. Retrieved November 26, 2023, from https://www.cse.ust.hk/~qyang/Docs/1999/ijcai99zhuyang.pdf