

# Feature Deletion and Case Discovery in Case-Base Maintenance

Brian Schack<sup>1,\*</sup>

<sup>1</sup>Indiana University Bloomington, 700 N Woodlawn Ave Ofc 3061T, Bloomington IN, 47408-3901

## Abstract

Case-based reasoning solves a problem by adapting the solution to a similar problem already solved. Motivated in part by the swamping utility problem, case-base maintenance strategies support a compact, competent case base by deleting, modifying, or discovering cases. This research summary briefly presents four case-base maintenance strategies developed by the author: 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.

## Keywords

case-based reasoning, artificial intelligence, swamping utility problem, case-base maintenance, flexible feature deletion, adaptation-guided feature deletion, expansion-contraction compression, predictive case discovery

## 1. Introduction

A *case base* can contain cases from training data, knowledge engineering by human experts, or the retention phase of the the case-based reasoning cycle. 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 expert attention to manually review.

The *swamping utility problem* describes this trade-off between the competence, quality, and speed contribution of a case versus its storage, retrieval, and bandwidth cost [1]. Technological progress has shifted the priority from storage cost to retrieval speed, but the swamping utility problem remains. Legacy systems, embedded systems, and unreliable networks worsen the

---

ICCBR DC'23: Doctoral Consortium at ICCBR2023, July 17 – 20, 2023, Aberdeen, Scotland

\*Corresponding author. The author gratefully acknowledges the contributions of his advisor Dr. David Leake at Indiana University Bloomington.


✉ schackb@iu.edu (B. Schack)

🌐 <http://bit.ly/schackbrian> (B. Schack)

🆔 0009-0000-1170-7632 (B. Schack)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

problem by constraining resources. Big data and streaming data worsen the problem by increasing resource usage. *Case-base maintenance* strategies attempt to mitigate the utility problem by judiciously choosing the most valuable cases to retain and the least valuable cases to delete in order to maintain a compact and competent case base [2]. Their effectiveness depends on their suitability to a particular dataset [3]. This research summary briefly presents four case-base maintenance strategies developed by the author that go beyond the deletion of cases by deleting features within cases or discovering new cases. (Due to space constraints, for more information about the strategies and for figures showing the experimental results, please see [4, 5, 6].)

## 2. Flexible Feature Deletion

Maintenance strategies normally make two assumptions: (1) that all cases have a uniform storage cost and (2) that they must retain or delete whole cases. For the first assumption, the storage costs of cases can vary when the cases contain varying amounts of data at varying levels of detail. Furthermore, the storage cost of both the problem and the solution can vary independently because a simple problem may require a complex solution and vice versa. For the second assumption, a 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 tends to degrade problem-solving competence -- but not necessarily to the same extent.

Domains that diverge from these two assumptions call for a different maintenance strategy: *Flexible feature deletion* subdivides variable-size cases for deletion of their components [4]. For example, cases based on medical imagery may have various resolutions and a large number of features of which only some are relevant to the diagnosis. For suitable data sets, compared to per-case strategies, flexible feature deletion can reduce the size of a case base with less reduction in the number of cases.

## 3. Adaptation-Guided Feature Deletion

Building on flexible feature deletion, instead of ordering features according to a knowledge-light metric, *adaptation-guided feature deletion* integrates additional knowledge from the solution transformation container about the recoverability of features [5]. 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. For example, if a cooking system has an adaptation rules for adding toppings to a pizza, then it can remove those toppings from the ingredients for the pizza recipe.

A solution with recovered features may exactly match 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. Evaluation in a path finding domain showed superior retention of competence compared to flexible feature deletion.

Alternatively, instead of deleting features, maintenance could also replace them with a smaller substitution or abstraction. Occasionally, this reorganization makes case contents more accessible to adaptation rules of limited power. For example, consider a maintenance strategy that extracts a component shared by multiple cases into a separate case leaving behind a marker in the cases from which the component was removed. This decreases the size of the case base by avoiding duplication. But it also makes the component available for reasoning as a standalone case independent of the cases from which it came. Even though case-base compression normally degrades competence, compression under these circumstances, termed *creative destruction*, can improve competence instead [5].

## 4. Expansion-Contraction Compression

By the *representativeness assumption*, maintenance strategies predict that future problems will follow a similar distribution to the current case base [7], and this works reasonably well for mature case bases in stable domains. But this assumption may apply less accurately during early case base growth, to dynamically changing domains, or in cross-domain transfer learning. For example, a recommender system for a travel agency needs to change with the seasons of the year.

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 [8]. The overfitting problem has received significant attention in the context of artificial neural networks [9]. Among several mitigations, neural networks may employ *data augmentation* which modifies training data in order to supplement it with additional instances [10]. For example, cropping images without obscuring their subjects.

Case-based reasoning does not normally apply data augmentation, but the solution transformation container provides a natural source for such adaptations. *Expansion-contraction compression* aims to improve competence preservation in case-base compression by combining exploration and exploitation [6]. The algorithm generates *ghost cases* by eagerly adapting existing cases and adds the ghost cases to the case base. For example, if a real estate appraisal system has a case for a house without a pool and a rule to adjust the price for putting in a pool, then it can construct a ghost case for a house with a pool. Then the condensed nearest neighbor algorithm selects cases for retention based on their competence contribution. Expansion-contraction compression has the potential to improve competence preservation by adding ghost cases that cover areas of the problem space not covered by the original case base. These ghost cases can provide diversity and increase the range of cases available for compression.

Experimental results show that expansion-contraction compression can outperform condensed nearest neighbor in terms of quality and competence preservation for some datasets -- especially when the training case base has gaps making it unrepresentative of the testing problems. The length of the adaptation path also influences competence retention, with longer paths associated with greater retention. The sparsity of the initial case base affects the competence trend, with a steeper decrease in competence for expansion-contraction compression

in the early phases of case base growth. The evaluation suggests that expansion-contraction compression has potential benefits for compression of unrepresentative case bases.

## 5. Predictive Case Discovery

Case-based reasoning depends on three types of regularity: problem-solution regularity, problem-distribution regularity, and concept regularity. *Problem-solution regularity* says that similar problems will have similar solutions, and this is necessary for the adaptation of a solution to a similar problem to be useful for solving the given problem. *Problem-distribution regularity* says that future problems will resemble past problems, and this is necessary for stored cases to remain useful over time. *Concept regularity* says that learned concepts remain valid, and this is necessary for machine learning generally. Leake and Wilson provide a formalization of problem-solution regularity and problem-distribution regularity [11].

The many effective applications of case-based reasoning prove that suitable task domains and thoughtful system design provide practical approximations of these regularities. But they are not guaranteed! Changes in the environment, user preferences, technology, or the problem space can degrade regularity and cause practical shortcomings -- especially for long-running systems. Lack of concept regularity, such as a working solution that becomes obsolete, leads to *concept drift*. Lack of problem-solution regularity, such as a divergence between similarity measures and adaptation rules, leads to *problem-solution drift*. And lack of problem-distribution regularity, such as heretofore unforeseen problems, leads to *problem-distribution drift*.

Much machine learning research has explored concept drift [12], but problem-distribution drift has received limited attention in case-based reasoning. In particular, problem-distribution drift impacts case-based maintenance because it breaks the *representativeness assumption* -- which says that the case base is a representative sample of the target problem space [7]. How to mitigate problem-distribution drift? One answer is to discover cases to add to the case base. Case discovery can be seen as similar in spirit to oversampling in SMOTE [13] and data augmentation in neural networks [14], and adaptation knowledge provides a natural source of cohesion-preserving transformations (à la ghost cases in expansion-contraction compression [6]).

*Predictive case discovery* attempts to mitigate problem-distribution drift by anticipating and acquiring cases expected to be useful for solving future problems. Building on prior work in drift detection [15], it identifies "hot spots" in the problem space to target for case discovery. When drift is detected (or at each time step, if the objective is to grow the case base), it divides the case base into clusters using the k-means algorithm [16] and a distance metric. Then predictive case discovery randomly chooses a cluster and finds the case at the center of that cluster. It discovers a variation on the centroid case by eagerly applying an adaptation rule or altering the value of a single feature. For example, when navigating an autonomous vehicle, a variation on a route could change one of the waypoints.

Evaluation on four scenarios (no drift, abrupt drift, cyclical drift, and drift from obsolescence) demonstrated that it outperformed baselines. But as the effectiveness of the case discovery strategy depends on characteristics of the drift itself, there is no universal strategy. An important next step is to investigate other strategies for drift detection and case discovery and how to

select the ideal strategy for a specific task domain.

## 6. Conclusion

The introduction to this research summary explained the swamping utility problem and how it motivates case-base maintenance. Middle sections briefly presented four case-base maintenance strategies that go beyond the deletion of cases by deleting features within cases or discovering new cases: flexible feature deletion, adaptation-guided feature deletion, expansion-contraction compression, and predictive case discovery. Evaluation of these case-base maintenance strategies, compared to appropriate baselines on suitable data sets, generally showed improvements in adaptation cost, competence, or solution quality. Future work will explore how to select the most appropriate strategy or combination of strategies for a particular scenario -- either using domain knowledge to guide maintenance policy or using cross-validation strategies that do not make assumptions about the case distribution.

## References

- [1] B. Smyth, P. Cunningham, The utility problem analysed: A case-based reasoning perspective, in: *Advances in Case-Based Reasoning*, Springer, 1996, pp. 392–399.
- [2] J. M. Juarez, S. Craw, J. R. Lopez-Delgado, M. Campos, Maintenance of case bases: Current algorithms after fifty years, in: *International Joint Conference on Artificial Intelligence*, volume 27, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 5457–5463.
- [3] D. W. Aha, Generalizing from case studies: A case study, in: *Machine Learning Proceedings*, Elsevier, 1992, pp. 1–10. doi:10.1016/B978-1-55860-247-2.50006-1.
- [4] D. Leake, B. Schack, Flexible feature deletion: compacting case bases by selectively compressing case contents, in: *Case-Based Reasoning Research and Development*, Springer, 2015, pp. 212–227. doi:10.1007/978-3-319-24586-7\_15.
- [5] D. Leake, B. Schack, Adaptation-guided feature deletion: Testing recoverability to guide case compression, in: *Case-Based Reasoning Research and Development*, volume 9969, Springer, 2016, pp. 234–248. doi:10.1007/978-3-319-47096-2\_16.
- [6] D. Leake, B. Schack, Exploration vs. exploitation in case-base maintenance: Leveraging competence-based deletion with ghost cases, in: *Case-Based Reasoning Research and Development*, volume 11156, Springer, 2018, pp. 202–218. doi:10.1007/978-3-030-01081-2\_14.
- [7] B. Smyth, E. McKenna, Competence models and the maintenance problem, *Computational Intelligence* 17 (2001) 235–249.
- [8] T. G. Dietterich, Overfitting and undercomputing in machine learning, *ACM Computing Surveys* 27 (1995) 326–327.
- [9] S. Lawrence, C. L. Giles, A. C. Tsoi, Lessons in neural network training: Overfitting may be harder than expected, in: *Proceedings of the 14th National Conference on Artificial Intelligence*, 1997, pp. 540–545. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.6468&rep=rep1&type=pdf>.

- [10] S. C. Wong, A. Gatt, V. Stamatescu, M. D. McDonnell, Understanding data augmentation for classification: when to warp?, in: 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), IEEE, 2016, pp. 1–6. URL: <https://arxiv.org/pdf/1609.08764.pdf>.
- [11] D. B. Leake, D. C. Wilson, When experience is wrong: Examining CBR for changing tasks and environments, in: ICCBR, volume 1650, Springer, 1999, pp. 218–232.
- [12] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, *IEEE transactions on knowledge and data engineering* 31 (2018) 2346–2363.
- [13] A. Fernández, S. Garcia, F. Herrera, N. V. Chawla, SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, *Journal of artificial intelligence research* 61 (2018) 863–905.
- [14] B. K. Iwana, S. Uchida, An empirical survey of data augmentation for time series classification with neural networks, *PLOS one* 16 (2021).
- [15] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: *Advances in Artificial Intelligence—SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence*, Sao Luis, Maranhao, Brazil, September 29–October 1, 2004. Proceedings 17, Springer, 2004, pp. 286–295.
- [16] M. Ahmed, R. Seraj, S. M. S. Islam, The k-means algorithm: A comprehensive survey and performance evaluation, *Electronics* 9 (2020) 1295.