Chapter 2
A Study of Two-Phase Retrieval for Process-Oriented Case-Based Reasoning

Joseph Kendall-Morwick and David Leake

Abstract Process-Oriented Case-Based Reasoning (PO-CBR) systems often use structured cases, which in turn require effective structure-based retrieval methods, especially when dealing with large processes and/or large case bases. Good retrieval performance can be facilitated by two-phased retrieval methods which first winnow candidate cases with a comparatively inexpensive retrieval phase, and then apply a more expensive strategy to rank the selected cases. Examples of such processes have been shown to provide good retrieval results in limited retrieval time. However, studies of such methods have focused primarily on overall performance, rather than on how the individual contributions of each phase interact to affect overall performance. This misses an opportunity to tune the component algorithms in light of the system task and case base characteristics, for specific task needs. This chapter examines two-phased retrieval as a means of addressing the complexity in many PO-CBR domains, and specifically examines the performance of each phase of two-phased retrieval individually, demonstrating characteristics of the phases' interaction and providing general lessons for how to design and deploy two-phased retrieval systems.

1 Introduction

Process-oriented tasks, such as business process management, workflow generation, and planning, require generating complex structured solutions. The generation of such solutions may be difficult due to imperfect domain knowledge and computational complexity. Case-based reasoning (CBR) (e.g., [1–3]) is appealing for

J. Kendall-Morwick (✉)
Computer Science Department, DePauw University, Greencastle, IN 46135, USA
e-mail: josephkendallmorwick@depauw.edu

D. Leake
School of Informatics and Computing, Indiana University, Bloomington, IN 47408, USA
e-mail: leake@cs.indiana.edu

addressing both problems. Because CBR generates solutions by revising the lessons of relevant prior solutions, it can increase efficiency by providing a head-start to problem-solving. Because its reasoning process has a strong analogical component, it enables solution generation to benefit from useful characteristics of retrieved solutions even if their relevance is not explicitly encoded. Consequently, process-oriented case-based reasoning (PO-CBR) has attracted considerable interest in the CBR community (e.g., [4]), for a wide range of tasks such as assisting in the development of new processes, supporting the enactment of existing processes, or the adaptation of processes during their execution.

A commonality across many PO-CBR tasks is the importance of representing structured information. Such information is generally represented in the form of labeled or attributed graphs, on which graph matching can play an important role in similarity judgments. However, graph matching algorithms have high computational cost, and their performance is constrained by theoretical limits: aspects of the graph matching problem are NP-complete. As a result, heuristic methods may be needed to speed up retrieval, even for small case bases, and especially for interactive applications requiring rapid response time. The need for interactive assistance places a special premium on retrieval efficiency, which may sometimes require balancing efficiency concerns against the quality of case selection when designing retrieval algorithms.

Greedy algorithms are often applied to structured case retrieval, but depending on the complexity of the data, greedy methods may not be optimal choices. Beyond the issue of how to compare a query case to a case in the case base, it may be necessary to focus the choice of which cases should be compared: An exhaustive case-by-case comparison wastes time comparing irrelevant cases to the query, but too strict restrictions on cases to compare could result in overlooking important cases.

This chapter provides a perspective on the retrieval of structured cases, illustrated with examples from the retrieval methods of Phala, a workflow authorship assistant. It focuses especially on two-phased techniques for retrieval, as used by Phala and also explored by others within the sub-field. Tools for supporting two-phase retrieval, developed for Phala, are implemented in a freely available system for structured case retrieval[1] which was used as the basis for the experiments in this chapter.

In particular, the chapter examines the design questions underlying development of two-phased retrieval systems and their ramifications on the performance of the two individual retrieval phases. The overall performance of two-phased retrieval systems has been studied in many task contexts (e.g., [5]). However, little study has been devoted to a key factor in how to design such systems: the individual contributions of the two phases. Analyzing such factors is important because it affects the characteristics to be sought in the design of each phase. This chapter presents experimental studies designed to illuminate the roles of each phase.

Results of the experiments suggest the value of tailoring overall parameter choices for two-phased retrieval choices to case base characteristics: in addition to any domain-specific choices for retrieval processes within each phase, the two-phased

[1] The Structure Access Interface (SAI): http://www.cs.indiana.edu/~jmorwick/SAI

retrieval strategy must itself be evaluated and the selectivity of each phase tuned. Our evaluation provides insight into what the important parameters and performance metrics for two-phased retrieval are, how changes in these parameters can affect performance, and how properties of the case base can complicate these relationships.

The chapter begins with background on PO-CBR and case retrieval in CBR. The following section expands by exploring representation and retrieval of structured cases. Section 4 goes into greater detail on two-phased retrieval, discussing various indexing strategies for two-phased retrieval and how to tune a two-phased retrieval configuration for an optimal combination of response time and accuracy, supported by an empirical study of a simple indexing strategy. The evaluation is performed using the Structure Access Interface system (SAI), a retrieval system developed for PO-CBR [6]. The chapter closes with observations and opportunities for future work.

2 Background

2.1 Retrieval in Case-Based Reasoning

The case-based reasoning process is often framed in terms of a cyclical process with four tasks which guide the access and application of stored cases, and learning through case storage: retrieve, reuse, revise, and retain [1]. Many retrieval approaches have been explored (see [3] for an overview). Often retrieval algorithms work in concert with cleverly designed domain-specific indexing schemes, to enable efficient selection of cases expected to be useful (e.g., [7, 8]); sometimes indices are designed to summarize structural characteristics, to enable retrieval to reflect structure without requiring full structure matching (e.g., [9]). However, domain-independent retrieval approaches cannot rely on such indexing obviating the need for structure matching. The work in this chapter focuses primarily on enabling rapid structural matching, which in turn can leverage semantic information when available.

2.2 Process-Oriented Case-Based Reasoning

PO-CBR is concerned with the application of case-based reasoning to process-oriented domains and with the resultant issues and opportunities [10]. PO-CBR research touches on many different domains, for example, business processes, e-Science, cooking, software development, health care, and game development, and PO-CBR systems use cases developed from data for varying types of processes such as workflows, plans, software models, and state machines. PO-CBR also studies the application of CBR to artifacts from processes which have executed, as for traces [11], logs, and provenance [12]. The study of workflows within PO-CBR has mainly fallen into two domains, business processes and e-Science.

Business Processes Business Process Management (BPM) concerns the management of business processes, the steps and procedures taken by an organization as whole to conduct some aspect of its business [13]. These steps are often modeled by workflows, which identify the order and conditions under which each step is taken, and what data or materials, potentially produced in a prior step, is necessary for each step.

BPM has been studied for several decades, and work to support BPM, particularly from CBR researchers, has gained traction in the last decade. Recently, Montani and Leonardi applied PO-CBR to cases of medical processes for the management of strokes, to determine their medical correctness [14]. Minor et al. have developed methods to support digital design by with a case-based means of adapting workflows [15]. Kapetanakis et al. have developed a system, CBR-WIMS, which provides a generic, case-based means of intelligent monitoring of business processes [16].

e-Science Information technology is playing an increasingly critical role in scientific experimentation, particularly in *e-Science*. e-Science consists of *in silico* experimentation (experiments that are run completely through execution of computer programs) as well as computational processes intended for data analysis and knowledge discovery [17]. Workflow technology is often employed in e-Science to manage these processes. Workflows developed for these purposes, called *Scientific Workflows*, consist of references to services (remote web-services, scripts, and local programs) and instructions for controlling the order of their execution and the flow of data between them. These services enact simulations, analysis, or transformations of data.

Some scientific workflows are small, but many contain a hundred steps or more, making them challenging for humans to compose. Also, the number of workflows that a scientist runs may be large. For example, ensemble simulation workflows run hundreds of highly similar workflows, differing slightly in structure or in parameters, to perform a parameter sweep study. In addition, the amount of data produced and consumed by services on the grid can be extremely large, and processing times long, making it important to generate the right workflow on the first attempt to avoid wasting computational resources. This provides the practical motivation for our work to apply PO-CBR to support scientific workflow generation.

The Phala Workflow Authorship Assistant Workflow authors are faced with many choices throughout the workflow generation process. Consequently, a number of efforts have made to assist scientists in workflow authorship, including workflow management systems such as Taverna [18], Kepler [19], Trident [20], and XBaya [21]. Additionally, resources such as myExperiment [22] and Biocatalogue [23] assist workflow authors in searching for past workflows or relevant services to include in their workflows.

Phala is a PO-CBR system that has been developed to assist authors of scientific workflows in the creative process of developing new workflows [24]. Like myExperiment, Phala relies on past works to provide assistance to workflow authors, however Phala differs in that users of the system are not burdened with locating relevant works and extracting the relevant information from those works; instead these are integrated into the Phala system, as part of its realization of the CBR cycle. Phala has been shown to produce on-point recommendations in a leave-one-out evaluation over a set of workflows from myExperiment.

Retrieval in Phala Phala is an interactive assistant, offering recommendations for edits to a partially completed workflow to the workflow author. Workflow authors use their discretion to determine which recommendations should be incorporated into the workflow.

In order to be an effective assistant, Phala must be able to react to user queries in seconds. However, rapid retrieval is made more difficult because Phala uses structured cases to represent dataflow through past execution traces of workflows (provenance)—which could slow retrieval to an unacceptable level for even a moderately sized case-base. The problem is further aggravated because Phala could potentially work with very large case-bases.

3 Structured Cases in CBR

Structured cases include information that is not global within the scope of the case; rather, it depends on relationships between individual components of the case. Graphs are a very common and powerful formalization frequently used to provide structure within data. Particularly useful are labeled graphs, which combine both data and structure. In labeled graphs, nodes are labeled with the information constrained to a particular component (an object or a concept). This information may be more complex than a simple classification. For example, ShapeCBR is a system that assists in the design of metal castings [25]. It represents casting components as values of which there are eight possible classifications. However, other details exist for individual instances of these components, such as scale. Lately, the cooking domain has been popular in CBR research, with recipes as cases [26]. In these cases, nodes may represent cooking tasks such as baking, which may be further parameterized by details such as temperature and duration.

Edges represent relationships between these entities. If only one type of relationship is modeled, edge labels are not necessary. However, this is often not the case. For instance, an unbounded number of relationship types must be modeled in the Concept Map CBR domain [27]. Likewise, in ShapeCBR represent physical linkage between casting components, and are not all equivalent; many components do not have an axis of symmetry between each joining location. Furthermore, different kinds of coupling objects are used to link larger components. This information must also be captured by the edge's labels.

Structured cases also have global attributes, similar to traditional feature vectors. In this sense they are a generalization of feature vectors, but more importantly, the global attributes provide a means to store unstructured data. For instance, in workflow domains, a number of global features exist, including semantic tags, contextual features such as author data, and data describing the structured components, such as the workflow language used.

3.1 Mapping and Comparing Structured Cases

Similarity metrics for structured cases can involve edit distance: the more edits required to transform one case to a state in which it is isomorphic to the other, the greater the difference between the two cases (e.g. [15]). Another, similar approach is to find the maximal common subgraph, which involves finding a correspondence between the nodes in each case and examining the number of resulting corresponding edges (e.g. [6]).

In either approach, determining the compatibility of nodes or edges to form a mapping between components of each case involves determining the compatibility of the features stored in the components. This can be a purely syntactic comparison, or can involve semantic constraints on the similarity of these features, such as those explored by Bergmann et al. [28, 29].

A key issue for retrieval and similarity assessment of structured data is that graph processing algorithms can be computationally expensive (for instance, subgraph isomorphism is NP-Complete [30]). A general retrieval tool must address the inherent complexity in these domains in order to handle similarity queries. Numerous methods have been developed to address these computational constraints, such as greedy algorithms, anytime algorithms, and heuristic search. For instance, Bergmann and Gil have adopted a knowledge intensive A* approach to mapping structured cases to queries in a workflow processing domain [28]. Reichherzer and Leake have used topological features of concept maps to weight the significance of local features [9]. Minor et al. have addressed the complexity in the workflow domain by converting graph structures to flat strings for comparison [31].

4 Two-Phased Case Retrieval

Achieving high retrieval performance requires both speeding up the case comparisons which are done and avoiding costly comparisons wherever possible. This approach has roots in cognitive science [32], as described below, and the idea of retrieving structures in two phases was adopted at an early stage by CBR researchers working with structured data [33, 34].

In a two-phased retrieval process, an initial "coarse-grained" and comparatively inexpensive retrieval process is used to winnow the cases to be considered. A more expensive "fine-grained" strategy is then applied to the pool of candidate cases selected by the coarse-grained strategy. PO-CBR researchers have noted the potential value of this strategy for scaling up case base sizes [35, 36], as have researchers in graph database systems [37–40]. For example, in one comparison, the Phala CBR system required approximately 15 minutes to apply its similarity metric to each case in a moderately sized case-base containing about 350 workflow cases, but required only a few seconds on the same hardware to perform the same search with two-phased retrieval and appropriate indexing for the first phase.

The use of indexing in two-phased retrieval makes it possible to add and remove cases with relatively little maintenance. Two-phase retrieval is flexible in that it can support multiple similarity measures simultaneously, to be applied as needed. It has some potential drawbacks—for example, the recall of the retrieval process will decrease if the coarse-grained strategy misses relevant cases—but for many tasks has proven useful compared to single-phase approaches.

In the remainder of this chapter, we use the following terminology for two-phased retrieval: *Phase 1* refers to the coarse-grained retrieval process aimed at efficiently selecting a subset of cases for more complete consideration, and *Phase 2* to the more computationally complex fine-grained ranking process. The maximum number of cases Phase 1 provides to Phase 2 is the Phase 1 Window, shortened to *Window 1*. The number of cases returned from the Phase 2 process for further processing by the CBR system is called the Phase 2 Window, shortened to *Window 2*.

MAC/FAC Gentner and Forbus's MAC/FAC ("Many are called but few are chosen") model, an early two-phased retrieval approach, was inspired by humans' capability to access structured knowledge quickly and soundly [32]. Gentner and Forbus argue that human retrieval speed implies that human retrieval must employ a fast, coarse-grained, massively parallel process which need not fully reflect structural constraints, and that the soundness implies the existence of a more thorough and precise process, which weighs structural constraints. The MAC/FAC algorithm models this behavior by performing retrieval in two sequential phases. The MAC phase involves comparing candidate cases (structures) through flat representations of features of their structural content. The FAC phase chooses a range of the top ranked cases from the MAC phase and re-orders them according to a more precise and computationally complex similarity metric. The smaller set of the top ranked cases from the FAC phase is returned.

The addition of the FAC phase can dramatically speed up retrieval by using a sublinear time lookup (in terms of the size of the case-base) for each case indexed by their features. Only the cases deemed most likely to be similar will then reach the FAC phase and have their structure examined and mapped in a more costly process (potentially requiring computation of the maximum common subgraph, or other hard graph comparison techniques).

Recent Applications of Two-Phased Retrieval Many systems have more recently applied two-phased retrieval approaches. The use of such approaches is a strong trend in the growing research area of graph databases [41]. Many graph database systems use a two-phased approach in which graphs are indexed by structural features used for an initial coarse-grained retrieval phase. During this phase, structural details of the graph are not loaded into memory, but are instead inferred through comparison of their association with flat indices representing structural summaries. Examples of graph database systems employing such a technique are Graphgrep [40], G-Hash [37], Periscope/GQ [38], and gIndex [39].

The SAI system, developed for Phala, joined this collection as a graph database system specifically tailored to the needs of PO-CBR researchers [42]. SAI provides a flexible means of implementing a retrieval algorithm. SAI was developed to assist

PO-CBR researchers and is intended for use in studying various retrieval algorithms of structured cases applied to various domains.

4.1 Indexing Strategies

Indexing is an effective means for efficient retrieval in large case-bases. Indices can be used to search for the most important and relevant cases without requiring a similarity assessment for each case in the case-base. Indexing is related to similarity assessment in that index selection should yield cases likely to be assessed as more similar. The representational power of a system's indices bears directly on how accurately index-based retrieval can estimate the outcome of a full similarity assessment.

Indices should capture important case properties in order to allow for content-based retrieval methods, such as conversational retrieval, retrieval by semantic tag, context-sensitive retrieval, and others. For example, Weber et al. retrieve related workflow instances through an interactive, conversational [43] process with the user [44], answering questions associated with cases in the case-base and retrieving cases related to the user's responses. Allampalli-Nagaraj et al. use semantic metadata tags to improve retrieval of medical imaging cases in their CBR application [45].

Many global case properties are easily attainable "surface features", stored in a literal form within the case. This is in contrast to deep features, which sometimes must be derived from the case data through potentially computationally expensive means [33]. Structural features are readily apparent in the case data, but they may correspond to deep features, because they are chosen from a very large feature-space (potentially constrained by the type of structural indexing used).

The following subsections detail three approaches to forming structural indices, with various degrees of distinguishing power and limits on the size of the index-space. An important consideration for these approaches is that each has its own advantages, and for a particular domain, any one of these methods may be optimal.

Path-1 Indexing SAI currently includes a simple indexing strategy that enumerates all paths of length 1 (a single edge), including select attributes from the edge and connected nodes. An example of this type of index can be seen in Fig. 1.

This indexing option was developed because the single-edge indices hold the advantage of $O(n)$ index generation while maintaining sufficient distinguishing power within the e-Science workflow domain to support Phala's retrieval needs. These short path indices are sufficient for the e-science domain but not for others, as later sections will demonstrate. This agrees with the result from the iGraph experiment, which determined that no single algorithm is best for every dataset in exact supergraph retrieval. This observation was the impetus behind SAI's modular approach, which allows for the integration of custom indexing strategies.

Path-K Indexing Beyond single edge indices such as those used by Path-1, longer paths could be used which increase the distinguishing power of each index. This also increases the size of the space of potential indices, in that graphs will, in the

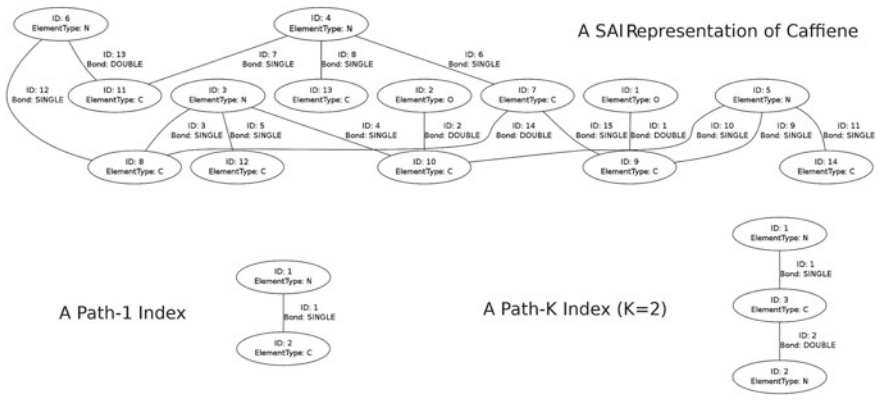

Fig. 1 A SAI representation of caffeine with path indices

worst case, have a $O(n^k)$ number of potential indices. With the growing size of the index-space, it may be advisable to only consider a subset of the space as viable indices. This also complicates the process of determining which viable indices are associated with a query at the time it is presented to the system. A system utilizing this type of indexing strategy is Graphgrep [40]. An example of this type of index can also be seen in Fig. 1.

Arbitrary Substructure Though Path-K indices can increase the distinguishing power of the indices, their topological limitations prevent the indexing strategy from providing optimal results as a phase-1 retrieval strategy for some domains. Yan et al. identify that Path-K indices do not sufficiently represent the structure of a graph in the domain of molecular structures, and present a system for mining common substructures for use in indexing [46]. This is a generalization of the Path-K indexing technique, in which the distinguishing power of indices is further increased, but the number of indices that could be associated with a case is $O(2^n)$ in the worst case scenario. This necessitates selectivity of which potential indices from the larger index-space will be used within a system, requiring a common substructure mining process, and additional complications during query-time to determine which indices are associated with the query. An example of this type of index can be seen in Fig. 2.

4.2 Evaluating Phase 1 Against Phase 2

This section analyzes how each phase affects various performance metrics. The first subsection outlines relevant specifics of the original MAC/FAC approach, the second subsection outlines how this approach can be adapted to address time constraints to accommodate interactive systems, and the third subsection outlines a method for comparing performance of each phase to determine which must be tuned when addressing performance problems.

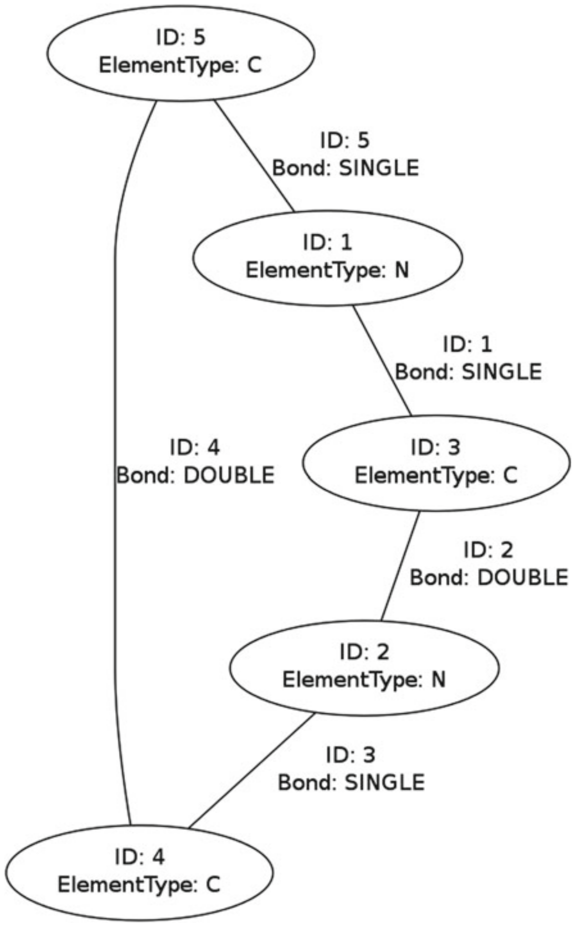

Fig. 2 An arbitrary substructure index

Effects of Limiting Phase 1 Retrieval by Similarity In Phase 1 of the original MAC/FAC model, the structure in memory sharing the most features with the query is selected, along with any others matching at least 90 % as many features. In this sense, the MAC phase works as a filter. Graph databases use a similar approach, in which features may be mined and used to approximately summarize structural details. These features are recorded as indices for the cases, enabling Phase 1 to retrieve a set of potentially relevant cases (cases which share many indices with the query). For such a Phase 1 process, the number of matches provided for consideration by Phase 2 depends on two main factors:

- The average size of clusters of cases with high similarity to each other
- The distinguishing power of the features selected for filtering

Applying Phase 2 processing to all cases within 10 % of the best match provides some assurance as to the quality of the result from Phase 2. For instance, if 99 %

of the time the top-ranked case according to the ranking of Phase 2 has a 90 % or greater match with the query based on Phase 1 features, then Phase 2 will output the top case in the case-base 99 % of the time. Unfortunately, providing this assurance requires that there be no constraint on the number of cases considered by Phase 1, and thus, there is also no lower-bound on the amount of time saved in Phase 2. We can avoid this problem by setting a limit on Window 1.

Using a Fixed Retrieval Window to Bound Retrieval Time If response time is important, as in an interactive system, we can set a bound on retrieval time by limiting the number of cases brought forth from the inexpensive Phase 1 to the significantly more expensive Phase 2. In the MAC/FAC model, cases are judged against each other relative to the degree to which they matched features within the query graph. We note rather than simply using this as a filter, it can be used to generate a ranking. For example, Phase 1 can create a nearest-neighbor ranking of the cases in the case-base according to the similarity of their features to those of the query, with Window 1 set to the maximum number of cases Phase 2 can process within the time allowed for Phase 2. Phase 1 will no longer be returning every case within a given similarity threshold of the query; as a result, the expected response time for a query can be bounded. Window 1 can be adjusted for the desired trade-off between response time and the similarity of the retrieved cases to the query.

Comparing Phase 1 and Phase 2 Rankings for Credit/Blame Assignment Evaluation of two-phased retrieval algorithms has traditionally been performed by examining performance of the system as a whole. The problem with this approach is that, if the desired case is not retrieved, it fails to identify which component of the system is most responsible for failure. Either phase may be responsible: Phase 1 may fail to rank the most relevant cases into the top cases in Window 1, making it impossible for the Phase 2 algorithm to find the most relevant cases. Similarly, Phase 2 may be provided with the most relevant case, but fail to rank it higher than other cases provided from Phase 1. To determine which bears more of the blame, we can compare the rankings from Phase 1 and Phase 2. Assuming that the finer-grained strategy of Phase 2 can be used as a "gold standard," increased difference between the rankings suggests blame for Phase 1. However, comparing similarity rankings, particularly in two-phased retrieval, is less straightforward than might be expected. The following paragraphs examine how such a comparison should be done for each retrieval phase.

Evaluating Phase 1 Retrieval In a study of case ranking for single-phase retrieval, Bogaerts and Leake [47] provide methods for comparing the results of a single-phase retrieval system to an ideal ranking of cases, to evaluate different retrieval algorithms. Adding a second phase to the retrieval process adds complications which may produce unexpected results in isolated cases. However, we have conducted an experiment which suggests that methods which are effective in comparing rankings for single-phased case retrieval systems may also be the most effective for comparing phase 1 and phase 2 rankings. This section first describes the complications which raise questions about assessing rankings for two-phased retrieval, and then sketches our experimental result.

For an example of how the problem of comparing phase 1 and phase 2 rankings can be complicated, suppose that Phase 1 happens to always return the top Window 1 cases of the ideal ranking, but in the opposite order. If Window 1 is large, this ranking scores very low when evaluated for single-phase retrieval. However, the end result of using this ranking for Phase 1 of the two-phased algorithm will be identical to the result of Phase 1 returning all cases in the ideal order: Phase 2 will re-rank the cases, so all that matters is that Phase 2 be provided with the correct set of Window 1 cases.

Another issue, addressed for single-phase systems by Bogaerts and Leake, concerns what they dub "the k-boundary" problem. This problem arises if a set of cases with the same similarity to the query straddle the retrieval window boundary, so that only some of the cases are provided to the next phase. In such situations, obvious methods of assessing retrieval quality may provide counter-intuitive results, making it difficult to determine which ordering is best. Depending on the similarity metrics used, this problem may arise frequently in both phase 1 and phase 2. For instance, we have used a simple matched feature count for phase 1 similarity assessment, which very frequently results in ties. For phase 2, we have used a covered edge count as a similarity metric, which often results in ties for small graphs.

Comparing ordering may also result in other anomalies for two-phased processes. For example, the Phase 1 ranking of the top Window 1 cases in opposite of ideal order will also score lower than some rankings which do not contain all of the top Window 1 cases. In such cases, the lower-scoring ranking will actually provide better performance than the higher scoring ranking! In this sense, traditional rank quality measures are not guaranteed to be a good measure for the effectiveness of Phase 1 ranking in every instance. Figure 3 illustrates such a case when Window 1 is 10 and Window 2 is 5. Ranking 1 will score lower than ranking 2 with a traditional weighted rank measure (our measure rates ranking 1 as 48.4 % similar to the ideal ranking and ranking 2 as 98.0 % similar to the ideal ranking). However, after phase 2 ranking, results from ranking 1 will yield the top 5 cases in order, whereas the results from

Fig. 3 Example rankings

Ideal	Ranking 1	Ranking 2
1	15	1
2	14	2
3	13	3
4	12	4
5	11	6
6	5	7
7	4	8
8	3	9
9	2	10
10	1	11
11	10	5
12	9	12
13	8	13
14	7	14
15	6	15

ranking 2 will miss case 5 and yield case 6 instead. Ranking 1 will yield better results than ranking 2, meaning that the traditional rank measure did not properly identify which ranking was superior.

However, such anomalies are generally unlikely to arise. In the prior example, sufficiently increasing or decreasing Window 1 or Window 2 eliminates the anomaly. In general, if the ranking generated by Phase 1 is a good approximation of the ranking generated by Phase 2, the combined algorithm will be more likely to locate the most similar cases.

To test whether considering rank order when comparing Phase 1 and Phase 2 rankings provides better information about the culpability of Phase 1 versus Phase 2 when the system fails to retrieve desired cases, we conducted experiments with a two-phased retrieval system in which the Phase 1 ranker was identical to the Phase 2 ranker, except for a predetermined random error introduced in Phase 1. We created 2 Phase 1 rankers, each with differing amounts of error. We used two rank measures to compare the rankings generated by each of these Phase 1 rankers: one measure which considered the ordering as part of the quality measure, and one which only considered the presence of the desired cases in the Window 2 set of cases (a recall measure). We ran 1,000 tests on a synthetic dataset, with Window 1 sizes of 20 and 25 and Window 2 sizes 5 and 10, with error differences from 0.01 to 0.30 and base error of 0.1. We then computed rank quality for both Phase 1 rankers with both measures, iteratively, until enough judgments were made to indicate which ranker had a higher error to a statistically significant degree. In all trials, the measure which considered order detected a statistically significant difference between the performance of the two phase 1 rankers faster than the recall measure, and only required an average of 5.5 % of the tests recall required to find a statistically significant difference with a Z test using 99 % confidence intervals. Thus considering the order in which phase 1 ranks cases provided better information on the source of retrieval problems.

4.3 Tuning Two-Phased Retrieval

In this section, we outline a process by which a system designer can answer basic questions about how to implement and deploy a two-phased retrieval subsystem. Specifically, we will examine the trade-offs involved in choosing Window 1 and Window 2 settings. We examined this question by using a simple two-phased retrieval algorithm within the SAI framework and applying it to two different datasets. The test retrieval algorithm indexes cases using the Path-1 indexing strategy which ships with SAI. The structural comparison component is a modified greedy algorithm.

Assessing the Dataset The first step in our process is to assess relevant properties of the case-base itself for prevalence of distinguishing features. Our first test dataset consists of scientific workflows downloaded from the myExperiment website [22]. We have used this dataset to evaluate the Phala CBR system for assisting authorship of scientific workflows [48]. In myExperiment the number of unique features is

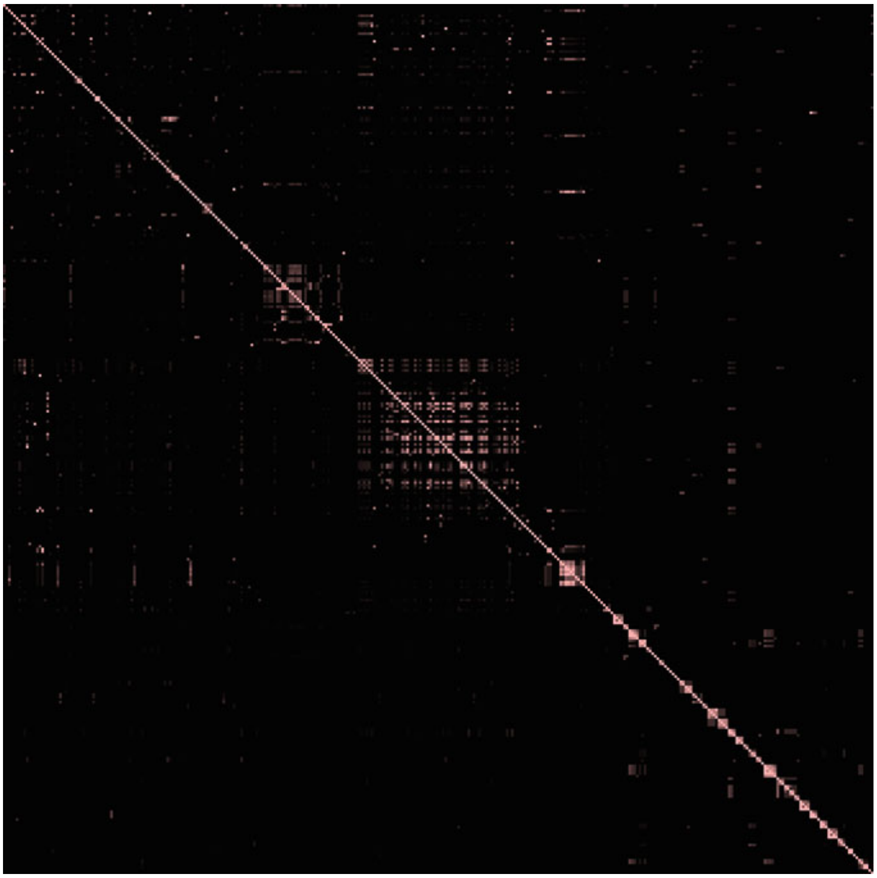

Fig. 4 Pairwise similarity of the myExperiment workflow dataset

relatively large in comparison to the number of nodes within cases in the case-base (specifically, there are 0.65 unique features per node). This results in high performance for our simple retrieval strategy, since the uniqueness of features leads to a larger index space and more distinguishing indices (for 341 graphs, there are 1627 unique indices). Figure 4 is a heat-map illustrating the similarity between each pair of cases from the myExperiment dataset. Note that most of this map is black, reflecting lack of similarity between most case pairs.

Determining Performance Needs The next step is to determine what is needed or desired in terms of performance. Performance of the system can be measured in many ways other than the problem-solving accuracy of the CBR system itself. Response time is important in interactive systems and constraints on the hardware used (e.g., memory limitations) are always relevant. To examine the impact the choice of indexing strategy has for a variety of window settings, and also to examine the

Table 1 Resources used per window 1 size

myExperiment			PubChem		
Window 1	Memory	Time	Window 1	Memory	Time
5	21573	1095	5	20142	5997
8	31768	1688	8	31146	10316
10	37581	1965	10	38811	13296
15	51053	2660	15	54403	19232
20	62741	3130	20	68887	25295
25	71527	3438	25	83618	30688
30	79996	3771	30	96774	35105

effectiveness of our indexing strategy, we generated rankings for both phases with the myExperiment dataset.

The results of this experiment are listed in the left-hand portions of Tables 1 and 2 (and graphed in Fig. 5). Table 1 lists the response time of the entire retrieval process and the memory consumed during retrieval. Response time is indicated in milliseconds and memory is indicated by an implementation-independent unit. Actual memory consumption is a function of this variable which also includes a constant factor for the size of the specific representation of structural elements in a specific implementation. Table 2 indicates how similar the ranking for the top Window 2 cases produced by Phase 1 is to the ranking produced by Phase 2, using a rank measure as described in the previous section.

Table 1 shows that, as Window 1 size increases, memory used and retrieval time also increase. This is expected, but what is more interesting is relating this data to the data in Table 2, which in turn shows that rank quality improves as either Window 2 decreases or Window 1 size increases. Specifically, we note the conflict between these qualities and the trade-off that results. This illustrates that values of Window 1 and Window 2 must be chosen carefully to achieve the optimal result for the system's specific goals.

Table 2 Average rank similarity

myExperiment				PubChem			
Window 1 / Window 2	3	5	10	Window 1 / Window 2	3	5	10
5	88.96	84.23		5	58.20	57.50	
8	92.27	89.01		8	59.83	59.62	
10	93.08	89.98	88.61	10	61.52	61.49	59.91
15	94.34	91.90	90.68	15	65.40	65.70	64.08
20	94.67	92.36	91.21	20	69.86	70.26	68.60
25	94.68	92.46	91.33	25	73.40	73.87	72.17
30	94.97	92.93	91.62	30	77.01	77.49	75.72

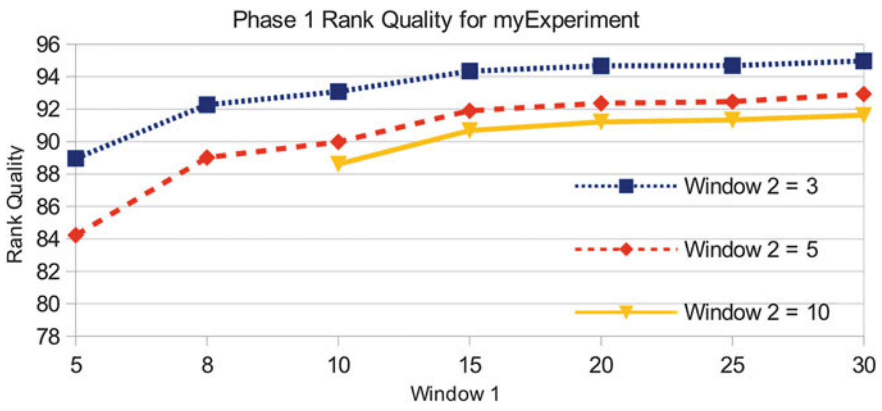

Fig. 5 Phase 1 rank quality with myExperiment dataset

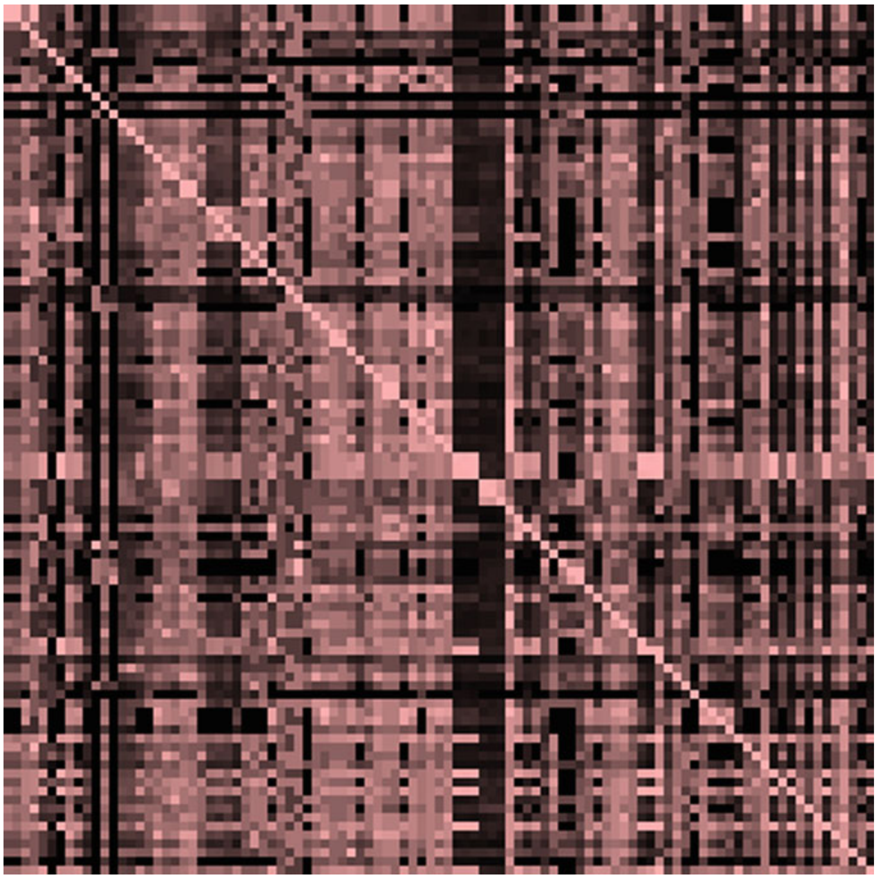

Fig. 6 Pairwise similarity of the PubChem molecular structure dataset

Examining the Impact of the Dataset We used a second dataset to examine how the properties of a dataset bear on decisions for implementing a retrieval system. The second dataset was 100 cases from the PubChem library of chemical structures [49]. Feature variety was much smaller, resulting in 0.0057 features per node and only 8 unique indices. Similarity between cases is much higher, as indicated by the uniformity of bright spots in Fig. 6, a heat-map indicating the similarity between each pair of cases in the PubChem dataset.

Tables 1 and 2 show poor Phase 2 performance compared to that for the first data set, which we ascribe to the small number of indices produced by Path-1. The tables also show the same trends noted for the myExperiment dataset.

5 Conclusions

This chapter presents a broad perspective on retrieval in PO-CBR, including a justification of the need to address complexity issues inherent in retrieval in order to scale up to larger case-base sizes, as well as concrete motivations from our research on case-based methods for supporting generation of scientific workflows. In particular, we have focused on two-phased retrieval to meet these needs.

Through our examination of two-phased retrieval, we have uncovered various factors affecting a trade-off between retrieval accuracy, in terms of the similarity of the cases retrieved, and system responsiveness, in terms of the time taken to respond to a user's query. We have directly examined two main categories of these factors, selection of an indexing strategy, and selection of a window size.

5.1 Selecting an Indexing Strategy

Our experiments illustrate that evaluation of each phase in a two-phased retrieval process may uncover which of the two components is directly contributing to a lack of system performance. In particular, the choice of phase-1 strategy may lead to varied results, depending on the domain. Given that Path-1 offers benefits by eliminating the need to limit the index space by mining frequent structures and also limits the complexity of determining which indices are associated with a query, it appears as a better choice for the myExperiment dataset, considering the rank quality is fairly high. However, this does not carry over to the PubChem dataset. In this case, rank quality is low enough to justify the use of a more complex indexing strategy.

5.2 Choosing Window Sizes

Our experiments show that choice of the Window 1 setting can have a dramatic impact on several competing performance benchmarks, so consideration of this parameter by a system designer is important to achieve the optimal balance between performance metrics. An analysis of the selection of the window size with a test dataset,such as was performed in this chapter, may be useful for determining optimal settings for a domain, thus liberating the user from the task of fine-tuning this parameter individually.

6 Future Work

We have outlined a set of considerations for implementing two-phased retrieval, taking its trade-offs into account and including key questions to examine. We believe that there are other important questions to ask during this process, and we intend to elaborate these in future work.

Beyond the feature to node ratio, there are many properties of case-bases which we expect to correlate with high or low performance for different indexing strategies (such as the size of the case base and its rate of growth, the average structural similarity of the cases, and others). We intend to study these characteristics by expanding the number and type of datasets we use in our experiments.

We also seek to study how these case-base properties predict performance of various indexing strategies. Specifically, we aim to implement the complete Graphgrep indexing strategy in order to study the effect of index size on the performance metrics for retrieval. We believe examination of choices affecting the size of indices, the range of index values, or the generality of indices should and will comprise an additional step in our approach to analyzing and implementing two-phased retrieval systems.

We seek to expand the performance metrics we consider in this process to include the time taken to store or index cases and the amount of long-term storage used, as well as to identify any other key questions not considered within this chapter in our future work.

We note that two-phased retrieval is not the only alternative for speeding up structured case retrieval; for example, other viable techniques include clustering [16], cover trees [50], and fish and shrink [51]. A more thorough comparative analysis of each of these techniques, which compares the drawbacks and benefits of using each technique and supports these distinctions with empirical evidence, is warranted. These techniques should be examined along side two-phased retrieval in an empirical study to determine strict criteria for judging when one technique is warranted over another. Additionally, as noted by Bergmann et al. [29], viable alternatives to indexing exist for performing phase 1 ranking within a two-phased retrieval approach, such as using surface features or other easily derived features, or techniques such as retrieval nets [52]. Similarly, a look at such alternatives would be an important aspect of an empirical study.

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. OCI-0721674 and by a grant from the Data to Insight Center of Indiana University. Portions of this chapter are adapted from a workshop paper [42] and a dissertation [24].

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. **7**(1), 39–52. http://www.iiia.csic.es/People/enric/AICom.pdf (1994)
2. Leake, D.: CBR in context: the present and future. In: Leake, D. (ed.) Case-Based Reasoning. Experiences, Lessons and Future Directions, pp. 3–30. AAAI Press, Menlo Park (1996)
3. Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision, and retention in CBR. Knowl. Eng. Rev. **20**(3), 255–260 (2005)
4. Minor, M., Montani, S.: Proceedings of the ICCBR-2012 workshop on provenance-aware case-based reasoning (2012)
5. Han, W.S., Pham, M.D., Lee, J., Kasperovics, R., Yu, J.X.: igraph in action: performance analysis of disk-based graph indexing techniques. In: Proceedings of the International Conference on Management of Data, SIGMOD '11, pp.1241–1242. ACM, New York (2011)
6. Kendall-Morwick, J., Leake, D.: Facilitating representation and retrieval of structured cases: principles and toolkit. Information Systems (2012, in press)
7. Leake, D.: An indexing vocabulary for case-based explanation. In: Proceedings of the 9th National Conference on Artificial Intelligence, pp. 10–15. AAAI Press, Menlo Park (1991)
8. Schank, R., Osgood, R., Brand, M., Burke, R., Domeshek, E., Edelson, D., Ferguson, W., Freed, M., Jona, M., Krulwich, B., Ohmayo, E., Pryor, L.: A content theory of memory indexing. Technical report 1, Institute for the Learning Sciences, Northwestern University (1990)
9. Reichherzer, T., Leake, D.: Towards automatic support for augmenting concept maps with documents. In: Proceedings of 2nd International Conference on Concept Mapping (2006)
10. Minor, M., Montani, S.: Preface. In: Proceedings of the ICCBR-12 Workshop on Process-Oriented Case-Based Reasoning (2012)
11. Floyd, M., Fuchs, B., Gonzlez-Calero, P., Leake, D., Ontañón, S., Plaza, E., Rubin, J.: Preface. In: Proceedings of the ICCBR-12 Workshop on TRUE: Traces for Reusing Users' Experiences—Cases, Episodes, and Stories (2012)
12. Leake, D., Roth-Berghofer, T., Smyth, B., Kendall-Morwick, J. (eds.): Proceedings of the ICCBR-2010 workshop on Provenance-Aware Case-Based Reasoning (2010)
13. Ko, R.K.L.: A computer scientist's introductory guide to business process management (bpm). Crossroads **15**(4), 4:11–4:18 (2009)
14. Montani, S., Leonardi, G.: Retrieval and clustering for business process monitoring: results and improvements. In: Díaz-Agudo, B., Watson, I. (eds.) ICCBR. Lecture Notes in Computer Science, vol 7466, pp. 269–283. Springer, New York (2012)
15. Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. In: Proceedings of the 7th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development, ICCBR '07, pp. 224–238. Springer, Berlin (2007)
16. Kapetanakis, S., Petridis, M., Ma, J., Knight, B., Bacon, L.: Enhancing similarity measures and context provision for the intelligent monitoring of business processes in cbr-wims. In: Proceedings of the ICCBR-11 Workshop on Process-Oriented Case-Based Reasoning (2011)

17. Oinn, T., Greenwood, M., Addis, M., Alpdemir, M.N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M.R., Senger, M., Stevens, R., Wipat, A., Wroe, C.: Taverna: lessons in creating a workflow environment for the life sciences: research articles. concurr. Comput. **18**(10), 1067–1100 (2006)
18. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics **20**(17), 3045–3054 (2004)
19. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S.: Kepler: An extensible system for design and execution of scientific workflows. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management, IEEE Computer Society, Washington (2004)
20. Barga, R., Jackson, J., Araujo, N., Guo, D., Gautam, N., Simmhan, Y.: The trident scientific workflow workbench. In: Proceedings of the 4th IEEE International Conference on eScience, pp. 317–318. IEEE Computer Society, Washington (2008)
21. Shirasuna, S.: A Dynamic Scientific Workflow System for the Web Services Architecture. PhD thesis, Indiana University (2007)
22. Goble, C.A., De Roure, D.C.: Myexperiment: social networking for workflow-using e-scientists. In: Proceedings of the 2nd Workshop on Workflows in Support of Large-Scale Science, WORKS ’07, pp. 1–2. ACM, New York (2007)
23. Bhagat, J., Tanoh, F., Nzuobontane, E., Laurent, T., Orlowski, J., Roos, M., Wolstencroft, K., Aleksejevs, S., Stevens, R., Pettifer, S., Lopez, R., Goble, C.A.: Biocatalogue: a universal catalogue of web services for the life sciences. Nucleic Acids Res. **38**(Web-Server-Issue), 689–694 (2010)
24. Kendall-Morwick, J.: Leveraging Structured Cases: Reasoning from Provenance Cases to Support Authorship of Workflows. PhD thesis, Indiana University (2012)
25. Mileman, T., Knight, B., Petridis, M., Preddy, K., Mejasson, P.: Maintenance of a case-base for the retrieval of rotationally symmetric shapes for the design of metal castings. In: Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning, EWCBR ’00, pp. 418–430. Springer, London (2000)
26. Stahl, A., Minor, M., Traphöner, R.: Preface: computer cooking contest. In: Schaaf, M. (ed.) ECCBR Workshops, pp. 197–198 (2008)
27. Cañas, A.J., Leake, D.B., Maguitman, A.G.: Combining concept mapping with CBR: towards experience-based support for knowledge modeling. In: Proceedings of the 14th International Florida Artificial Intelligence Research Society Conference, pp. 286–290. AAAI Press, Menlo Calif (2001)
28. Bergmann, R., Gil, Y.: Retrieval of semantic workfows with knowledge intensive similarity measures. In: Proceedings of 19th International Conference on Case-Based Reasoning, Springer, Berlin (2011, in Press)
29. Bergmann, R., Minor, M., Islam, M.S., Schumacher, P., Stromer, A.: Scaling similarity-based retrieval of semantic workflows. In: Proceedings of the ICCBR-12 Workshop on Process-Oriented Case-Based Reasoning (2012)
30. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, STOC ’71, pp. 151–158. ACM, New York (1971)
31. Minor, M., Bergmann, R., Grg, S., Walter, K.: Adaptation of cooking instructions following the workflow paradigm. In: Marling, C. (ed.) ICCBR 2010 Workshop Proceedings (2010)
32. Gentner, D., Forbus, K.: MAC/FAC: A model of similarity-based retrieval. In: Proceedings of the 13th Annual Conference of the Cognitive Science Society, pp. 504–509. Cognitive Science Society, Chicago (1991)
33. Mntaras, R.L.D., Bridge, D., Mcsherry, D.: Case-based reasoning: an overview. AI Commun. **10**, 21–29 (1997)
34. Börner, K.: Structural similarity as guidance in case-based design. In: Selected Papers from the 1st European Workshop on Topics in Case-Based Reasoning, EWCBR ’93, pp. 197–208. Springer, London (1994)

35. Kendall-Morwick, J., Leake, D.: A toolkit for representation and retrieval of structured cases. In: Proceedings of the ICCBR-11 Workshop on Process-Oriented Case-Based Reasoning (2011)
36. Bergmann, R., Gil, Y.: Retrieval of semantic workflows with knowledge intensive similarity measures. In: Proceedings of the 19th International Conference on Case-Based Reasoning Research and Development, ICCBR'11, pp. 17–31. Springer, Berlin (2011)
37. Wang, X., Smalter, A., Huan, J., Lushington, G.H.: G-hash: towards fast kernel-based similarity search in large graph databases. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '09, pp. 472–480. ACM, New York (2009)
38. Tian, Y., Patel, J.M., Nair, V., Martini, S., Kretzler, M.: Periscope/GQ: a graph querying toolkit. Proc. VLDB Endow. **1**, 1404–1407 (2008)
39. Yan, X., Yu, P.S., Han, J.: Graph indexing: a frequent structure-based approach. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD '04, pp. 335–346. ACM, New York (2004)
40. Giugno, R., Shasha, D.: Graphgrep: a fast and universal method for querying graphs. In: Proceedings of 16th International Conference on Pattern Recognition, vol. 2, pp. 112–115 (2002)
41. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Comput. Surv. **40**, 1:1–1:39 (2008)
42. Kendall-Morwick, J., Leake, D.: On tuning two-phase retrieval for structured cases. In: Proceedings of the ICCBR-12 Workshop on Process-Oriented Case-Based Reasoning (2012)
43. Aha, D., Breslow, L., Munoz-Avila, H.: Conversational case-based reasoning. Appl. Intell. **14**, 9–32 (2001)
44. Weber, B., Reichert, M., Wild, W.: Case-base maintenance for ccbr-based process evolution. In: Roth-Berghofer, T., Göker, M.H., Güvenir, H.A. (eds.) ECCBR. Lecture Notes in Computer Science, vol. 4106, pp. 106–120. Springer, Berlin (2006)
45. Allampalli-Nagaraj, G., Bichindaritz, I.: Automatic semantic indexing of medical images using a web ontology language for case-based image retrieval. Eng. Appl. Artif. Intell. **22**(1), 18–25 (2009)
46. Yan, X., Yu, P.S., Han, J.: Substructure similarity search in graph databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD '05, pp. 766–777. ACM, New York (2005)
47. Bogaerts, S., Leake, D.: Formal and experimental foundations of a new rank quality measure. In: Proceedings of the 9th European conference on Advances in Case-Based Reasoning, ECCBR '08, pp. 74–88. Springer, Berlin (2008)
48. Leake, D., Kendall-Morwick, J.: Towards case-based support for e-science workflow generation by mining provenance. In: Proceedings of the 9th European Conference on Advances in Case-Based Reasoning, ECCBR '08, pp. 269–283. Springer, Berlin (2008)
49. Bolton, E.E., Wang, Y., Thiessen, P.A., Bryant, S.H.: PubChem: integrated platform of small molecules and biological activities. Annu. Rep. Comput. Chem. **4**, 217–241 (2008)
50. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd International Conference on Machine Learning, ICML '06, pp. 97–104. ACM, New York (2006)
51. Schaaf, J.W.: Fish and shrink. a next step towards efficient case retrieval in large-scale case bases. In: Proceedings of the 3rd European Workshop on Advances in Case-Based Reasoning, EWCBR '96, pp. 362–376. Springer, London (1996)
52. Lenz, M., Burkhard, H.D.: Case retrieval nets: basic ideas and extensions. In: Grz, G., Hldobler, S. (eds.) KI-96: Advances in Artificial Intelligence. Lecture Notes in Computer Science, vol. 1137, pp. 227–239. Springer, Berlin (1996)