**QUEEN'S
UNIVERSITY
BELFAST**

Revisiting Fast and Slow Thinking in Case-Based Reasoning

Kaurav, S., Ganesan, D., P, D., & Chakraborti, S. (2021). Revisiting Fast and Slow Thinking in Case-Based Reasoning. In *29th International Conference on Case-based Reasoning* (Lecture Notes in Computer Science). Springer. https://doi.org/10.1007/978-3-030-86957-1_8

Published in:
29th International Conference on Case-based Reasoning

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
[Link to publication record in Queen's University Belfast Research Portal](#)

Publisher rights
© 2021 Springer.
This work is made available online in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Revisiting Fast and Slow Thinking in Case-Based Reasoning

Srashti Kaurav¹, Devi Ganesan¹, Deepak P^{1,2} and Sutanu Chakraborti¹

¹ Indian Institute of Technology, Madras, Chennai 600036, TN, India
{cs19s013,gdevi,sutanuc}@cse.iitm.ac.in

² Queen's University Belfast, UK deepaksp@acm.org

Abstract. A dichotomous Case-Based Reasoning (CBR) model is one in which two kinds of reasoning mechanisms are employed for realizing fast and slow problem-solving as demanded by the nature of the incoming query. This is inspired by Daniel Kahneman's seminal work on the two modes of thinking observed in humans. In this paper, we present the following three directions of refinement for a dichotomous CBR model: selection of attributes for a fast thinking model based on parsimonious CBR, switching from fast to slow thinking based on constraints derived from domain knowledge and arriving at a complexity measure for evaluating dichotomous models. For all the three improvements identified, we discuss the results on real-world data sets and empirically analyse the effectiveness of the same.

Keywords: Fast and Slow Thinking · Dichotomous CBR models · Cognitive CBR

1 Introduction

Case-Based Reasoning (CBR) is based on the idea of experiential problem solving, the idea that past problem solving experiences can be reused to solve new problems. CBR has found interesting applications in several real world tasks in domains such as diagnostics, planning, design and configuration [13]. However, it is indeed paradoxical that while CBR was inspired initially by models of human problem solving, many practical CBR realizations have made design choices that considerably compromised on CBR's cognitive appeal. In this paper, we attempt to explore avenues to realize CBR in ways that can mirror the dichotomy between *slow* and *fast* thinking, as elucidated in Daniel Kahneman's seminal work "Thinking, Fast and Slow" [10]. This paper reports follow-up work based on our earlier recent work [12], which introduced the possibility of realizing Kahneman's ideas within the CBR paradigm.

Kahneman's central thesis is that human cognition operates in two modes: (a) fast thinking, which is quick, intuitive and largely involuntary, and (b) slow thinking, which is deliberate, effortful and often involves complex computations. Kahneman identifies two systems corresponding to these two modalities, which

he calls System 1 and System 2, respectively. These two systems are merely conceptual abstractions and do not imply any physiological or biological separation within the brain. It is tempting to realize these two modalities in the CBR context since we often encounter applications where we would like a CBR system to effectively trade-off effectiveness for time efficiency or vice versa. This can be achieved by a CBR system that uses the fast thinking mode to solve a vast majority of queries (target problems), and switches over to slow thinking, only if it recognizes the target problem as hard. This raises two central questions: (a) On what dimensions are slow and fast thinking mechanisms different? (b) On what basis would we judge a target problem to be solvable by fast thinking, and what are the mechanisms of switching from fast to slow thinking? With respect to the first question, [12] presents two different schemes: the first in which fast thinking uses a subset of features used by slow thinking, and the second in which slow thinking makes use of time-consuming adaptation processes, which fast thinking does away with. With respect to the second question, a couple of switching strategies are presented in [12], details of which are outlined in the next section. One key idea is as follows: in a classification setting, cases are pre-tagged based on how complex their neighbourhoods are. If neighbours of a case have widely differing class labels, the neighbourhood is complex; else, it is simple. Given a target query, the system switches to slow thinking if the neighbours of the target case, as identified by fast thinking, are labelled as complex. Therefore the fast thinking will be more effective when the System 1 is highly aligned.

Despite making preliminary attempts to realize slow and fast thinking within CBR, several important questions remained unaddressed in [12]. This paper is aimed at identifying and addressing three of these gaps. Firstly, in a mechanism where fast thinking uses a subset of the attributes used in slow thinking, we need a principled approach to decide feature selection. This is clearly an optimization problem, and the objective function is decided by the nature of trade-offs required in the domain of interest. We formalize the problem and report empirical findings in a general setting where similarity estimation over different features have different time requirements, and relative importances to effectiveness and time efficiency can be flexibly tuned. Our second contribution is motivated by the observation that a student solving a high school problem in physics based on reusing his experience in solving similar problems in the past often realizes that he made a mistake when he observes that his solution fails to satisfy the constraints as demanded by the laws of physics. Thus domain constraints play an important role in effecting the switching from fast to slow thinking. We present novel CBR realizations that are inspired by this idea. The third contribution of the current work is to present a principled approach to quantify the tradeoff between effectiveness and efficiency in a CBR realization of the fast-slow dichotomy. Such a quantification also allows us to have a-priori insights on how the nature of the domain influences the design choices of a CBR system that trades off speed for accuracy or vice versa. The structure of the paper is as follows. We present a brief background and an overview of related works in this area in Section 2. Sections 3, 4 and 5 present our central contributions, as outlined above, along with

a critical analysis of empirical findings. Section 6 presents potential extensions of the work and summarizes its key contributions.

2 Background

Looking at the picture of an angry face, we instinctively and involuntarily seem to relate to the emotion expressed in the picture. In contrast, the problem of multiplying two three-digit numbers needs slow and deliberate effort. Kahneman's work discriminates between these two approaches of problem solving, calling them *fast* and *slow* thinking, respectively. It often happens that both modes of operation are at work in the same task. Given a chessboard setting, fast thinking may instinctively suggest a move, which in hindsight appears to be a wrong one, and hence slow thinking based on more elaborate cost-benefit analysis is called into action. Similarly, while solving a numerical physics problem, one may be tempted to reuse the steps in a set of problems solved previously, which on-the-surface appear to be similar, though more careful inspection reveals subtle differences that require one to revisit first principles and work from scratch. Thus, one central role of slow thinking is often to correct errors made in fast thinking. Kahneman demonstrates this tension between two aspects of our cognition: the availability heuristic, which is a cognitive bias that refers to our propensity to solve a complex problem by quickly reusing the solution of what appears to be a simple, more familiar problem, and metacognition, the ability to reason about our thinking processes and thus correct failures.

To the best of our knowledge, Crow and Aamodt [4] make the first effort to build a bridge between the dichotomous cognitive mechanisms in Kahneman's work and CBR. The central observation in [4] is that fast thinking can be realized in settings where the knowledge of similarity is straightforward and similar problems are likely to have similar solutions. In contrast, slow thinking may be necessitated when retrieved neighbours have conflicting solutions, and hence more complicated retrieval mechanisms or adaptation knowledge may be involved. Kannengiesser and Gero [11] have used the dual System theory proposed by Kahneman in case-based design task. Case-based design has the following phases: problem anticipation, search, match, retrieve, select, modify and repair, where each phase involves either System 1 or System 2 operations. Plaza and Aamodt [1] discussed about using Kahneman's dual system model in an integrated view of CBR encompassing both data-driven and knowledge-intensive processes. They have suggested that System 1 which is associative can involve data-driven approaches like deep learning.

We have seen in the earlier section that realizing the fast-slow dichotomy within CBR involves identifying the mechanisms that separate the two modalities, as also the criteria for switching from *fast* to *slow*, given a target problem. Our earlier work [12] presents three distinct models, referred to as Models 1, 2 and 3, which differ with respect to each other in terms of the mechanisms to realize fast and slow thinking. Two distinct approaches for switching from fast to slow thinking were also presented. We summarize the key ideas below.

Let us consider a classification problem. Model 1 uses a standard model-based Machine Learner like Support Vector Machine (SVM) to realize the fast thinking module. Classifiers like SVM output a score corresponding to the confidence of the classification. If the confidence is low, the system switches to slow thinking where a slower instance-based learner like a CBR system solves the target problem. Unlike Model 1, Model 2 uses CBR for both slow and fast thinking, except that in the fast thinking mode, the CBR system uses only a subset of the entire set of features used in the slow thinking mode, for similarity estimation purposes. Model 3 too uses CBR for realizing both fast and slow thinking, but here slow thinking makes use of adaptation, while fast thinking publishes its result without adaptation. Let us now turn our attention to the switching mechanisms presented in [12]. The first mechanism, called tag-based switching, involves examining each case in the case base and tagging it as simple/*FT* or complex/*ST*, based on the nature of its neighbourhood. If the class label (solution) of a case is largely in agreement with those of its neighbours, it is tagged simple; otherwise, it is labelled complex. Given a target case, a decision is made to switch from fast thinking to slow thinking if many of the neighbours of the target case are tagged as complex. This is justified, since in such cases, the solutions of the neighbours are likely to be poorly indicative of the true target case solution, and hence additional evidence is needed to arrive at a confident prediction. A second switching mechanism called oracle-based switching is also presented in [12], which is based on partial feedback from an expert on the solution generated by the fast thinking module. In particular, an expert can declare whether the solution is correct or not without giving the true solution; if the solution is incorrect, the system switches to slow thinking. In the current work, we confine our attention to realizations where both the fast and slow thinking modules are implemented using CBR. In particular, we choose Model 2 as the basis of our research. The current work is based on the observation that several questions needed to be answered in the earlier work [12], in order to make it more rigorous. For one, the mathematical premise for choosing a subset of features from the entire pool for fast thinking in Model 2 remained unclear. We show how this can be cast as an optimization problem that can be solved under diverse criteria pertaining to diverse similarity estimation time requirements across different features, and also the nature of effectiveness v/s time efficiency trade-offs required in the domain of interest. Secondly, we address the need of a principled study based on quantifying the gains achieved by splitting a CBR system between fast and slow thinking modalities. In some domains, the gains are expected to be higher than in some others. We generalize the basic idea of footprint-based competence models [16], and empirically demonstrate the advantages of this conceptual extension in facilitating a systematic study in this context. In addition to these contributions, we focus purely on the tag-based switching mechanism, while also presenting a novel mechanism called *constraint-based switching* inspired by Kahneman’s original work. The broad idea is to trigger a shift from fast to slow thinking when the system realizes that the solution generated by the fast thinking module fails to comply with certain underlying domain constraints.

Though the work presented in this paper has very little overlap with past work in terms of its specific goals and contributions, it may be worth highlighting a few other efforts to strike a trade-off between effectiveness of retrieval and time efficiency in CBR, and in the wider Machine Learning context. Dileep and Chakraborti [5] presented an approach where text classification, instead of treating each test document uniformly, resorted to fast classification in case the title or some noteworthy features were reckoned to be adequately indicative of the category label, and fell back to more elaborate slow thinking mechanisms only on demand. [7] is one of the earliest works in the CBR community, where retrieval failures trigger introspective reasoning processes, which in turn are used to refine indices in a CBR planner. Another interesting work is that on anytime algorithms [17], which are designed with the goal of giving increasingly better results when allowed more computation time.

3 Feature Selection

We now consider a feature selection problem which is targeted towards improving the fast response process, within the '*fast and slow thinking*' dichotomy.

3.1 Feature Selection for Parsimonious Search

Looking back at [12], it may be seen that the two systems (System 1 and System 2) in Model 2 differ based on the sufficiency and insufficiency of a small subset of attributes in solving the task. In other words, if a small subset of features are sufficient to solve a problem, it gets assigned the "*FT* tag", which indicates that the problem can be solved using fast thinking. This leads us to a natural question: *how do we choose a small subset of features to instantiate the parsimonious search in Model 2?* Towards addressing this question, we may observe two key aspects relating to the choice of features:

- Similarity computation over the chosen subset of features should be fast.
- The similarity space over the chosen subset of features should be meaningful for problem solving.

First, one may observe that different features differ in the computation that would be incurred for estimating similarities. As an example, numeric attributes lead to fast similarity computation since numeric comparisons could be realized using swift bit-wise operations. On the other hand, similarities between text attributes and set-valued attributes call for costly operations such as sub-sequence finding and set intersection computation. Thus, posed with two features that are equally relevant for problem solving, we may intuitively choose the feature whose similarity computation is relatively lighter computationally. *Second*, features may differ in the amount of utility towards solving a problem. Consider the scenario of estimating the chances of an individual's risk of fatal outcome if she were to contract COVID-19. Features relating to the past history of respiratory diseases would be considered far more relevant than, say, an attribute

such as gender. Thus, even when comparing respiratory disease histories may be more onerous computationally, we may choose to include it in preference to the gender attribute. In simple terms, people who are similar in their respiratory disease histories may have similar COVID-19 risks as compared to people with similar gender. *To summarize*, the set of features to be chosen for the parsimonious search depends on two criteria; that of computational load in estimating similarities, and problem solving utility.

3.2 Modelling Desirability of Feature Sets

In the feature selection approach we will describe, we will use *case base alignment* as a measure of problem solving utility. Given a set of chosen features, case base alignment measures the extent to which the maxim of *similar problems lead to similar solutions* holds within the similarity space defined by the chosen features. We use a popular case base alignment measure [14] in our model which considers, the correlation between the problem similarities, and the solution similarities of cases, as a measure of alignment. Other suitable alignment measures can be used.

Our desirability objective for a set of features F , is modelled as the following:

$$D(F) = \frac{(Alignment(X, F))^\alpha}{(k \times \sum_{f \in F} t(f))^{1-\alpha}} \quad (1)$$

$$F^* = \arg \max_{F' \subseteq \mathcal{F}} D(F') \quad (2)$$

where, $\mathcal{F}$ is the set containing all the features.

In other words, the task is to choose a subset of features, F^* from $\mathcal{F}$, balancing between the need for higher alignment and the need for fast computation. $Alignment(X, F)$ denotes the case base alignment of the case base X as measures over the feature set F . Each feature, f , also has a computation time, $t(f)$. The cumulative computation time appears in the denominator of Eq. 1 and is weighted by a weighting parameter k . The parameter α controls the relative importance of the alignment and computation time considerations. For higher values of α , the discovered subsets would be better in terms of alignment, and vice versa for smaller values of α . To put it another way, for scenarios where real-time responses are desired, α may be set to a very low value. F^* , as may be obvious by now, denotes the ideal choice of features based on our modelling of the objective.

3.3 Brute Force Approach

The simplest possibility of identifying the optimal desirability feature subset would be to inspect all possible non-empty feature subsets of $\mathcal{F}$, where $\mathcal{F}$ is the set containing all the features. This very simple but computationally expensive approach can be summarized in the procedural steps below:

- $F = \phi, V = 0$
- $\forall F' \in \text{PowerSet}(\mathcal{F})$
 - $\text{Val}(F') = D(F')$
 - $\text{if}(\text{Val}(F') > V)$
 - * $F = F'$
 - * $V = \text{Val}(F')$
- Return F as F^*

$\text{PowerSet}(\mathcal{F})$ contains all possible non-empty feature subsets of $\mathcal{F}$. The above procedure leads to an expensive computation since the number of elements in $\text{PowerSet}(\mathcal{F})$ would be in $\mathcal{O}(2^{|\mathcal{F}|})$. However, this process, by design, would be able to always identify the optimal desirability feature subset since all subsets are being evaluated.

3.4 Greedy Approach

The greedy approach reins in the computational load by channelizing the search towards certain candidates based on an estimated likelihood of them leading to desirable feature subsets. The procedural steps are outlined below:

- $F = \mathcal{F}, V = D(\mathcal{F})$
- *while* ($F \neq \phi$)
 - $F' = \arg \max_{f \in F} D(F - \{f\})$
 - $\text{if}(D(F') > V)$
 - * $F = F'$
 - * $V = D(F')$
- *return* F as F^*

This greedy approach starts with all the features as among the chosen features. This is followed by progressively dropping the feature that leads to the smallest drop in desirability, until all features are exhausted. This search excluding one feature at a time could miss the optimal F^* if it doesn't fall within that search path, but the heuristic search ensures that such misses are not very likely. In return for such approximation, the greedy search is able to achieve a quadratic response time, i.e., in $\mathcal{O}(|\mathcal{F}|^2)$, a massive improvement from the brute force method which was in $\mathcal{O}(2^{|\mathcal{F}|})$.

3.5 Experimental Analysis

Experimental Setup: Now we will study the tradeoff between the effectiveness and the efficiency of the Parsimonious CBR model, we will tune the desirability of the model using α as a hyperparameter. In order to evaluate the effectiveness of our modelling of desirability of features, we evaluated the greedy approach using two datasets, viz., *AutoMPG Dataset* (AMPG) [6] and *Pima Indians Diabetes Dataset* (PID) [15] with 399 and 768 instances respectively. Both datasets had 8 features each. We observed that the time taken for similarity computation

by each feature is almost the same for both datasets. Therefore, we have also experimented by externally modifying the time, $t(f)$, for each feature, to study the trends when different features take different amounts of time.

Experimental Results: Table 1 and Table 2 show the results for the AMPG and PID datasets respectively where the selected (and not selected) features are reported in the form of binary vectors. For example, the binary vector $[0, 0, 0, 1, 0, 0, 0]$ indicates that only the fourth feature was chosen, while all other features were discarded. The timing for similarity computation is also represented using a similar vector of length 8; for example, $[1, 1, 1, 10, 1, 1, 1]$ denotes that $t(f)$ for the fourth feature is 10 whereas for all other features, it is 1. This suggests that computing similarities for the fourth feature would be ten times as expensive as computing it over the other features.

Tables 1 and 2 show the results for varying values of α . For lower α (i.e., closer to 0 given that $\alpha \in [0, 1]$), we would expect that the computation time of feature are given primacy; on the other hand, for α closer to 1.0, even computationally costly features may be selected if they help improve case base alignment. In Table 1, moving from left to right, we are increasing the timing for features progressively; for example, the second column has $t(f_4) = 10$, whereas for the third and fourth columns, $t(f_2)$ and $t(f_1)$ are additionally set to 10 respectively. From the first column, where the favorite choice seems to be the choice of f_4 . When $t(f_4)$ is set to 10 for the second column, the choice is seen to shift to f_2 . For $\alpha = 0.93$, the choice with uniform timing vector is $\{f_4\}$; however, when f_4 is made a costly feature in the second column, observe that the method is forced to choose multiple features to heed to the high impetus on case alignment (since $\alpha = 0.93$) in order to compensate for the inability to choose the best feature. This shows a meaningful movement of choices based on how the computational expense can be traded off for problem solving utility as measured using case base alignment. Similar trends are observed in Table 2 for the PID dataset also, where the choice of desirable features shift away from computationally expensive features with higher settings of α . Fig 1 shows the tradeoff between the α and global alignment value [14] of System 1 with respect to the selected features. We can see that for lower α values, when $t(f)$ is high for important features, the alignment is low since the important attributes are neglected.

4 Constraint-Based Switching

In our earlier work [12] we have proposed various switching techniques to switch from System 1 to System 2. In the context of Model 2, having parsimonious CBR as fast thinking, the switching strategies are briefly described as follows: (a) Oracle-based switching, which involves a partial feedback given by an expert or an oracle. The partial feedback is such that the reasoner is informed whether the System 1 solution is *correct* or *incorrect*. (b) Tag-based switching is mainly proposed for the classification setting. It involves tagging all the cases in the case-base as *FT*, *ST* or *N*. Given a case base, each case C in the case base is solved using a leave one out simulation. If C can be successfully solved using

Table 1. Results for AutoMPG Dataset for various timing vectors

α value	Timing Vectors, t			
	[1,1,1,1,1,1,1,1]	[1,1,1,10,1,1,1,1]	[1,10,1,10,1,1,1,1]	[10,10,1,10,1,1,1,1]
0.1	[0,0,0,1,0,0,0,0]	[0,1,0,0,0,0,0,0]	[1,0,0,0,0,0,0,0]	[0,0,1,0,0,0,0,0]
0.5	[0,0,0,1,0,0,0,0]	[0,1,0,0,0,0,0,0]	[1,0,0,0,0,0,0,0]	[0,0,1,0,0,0,0,0]
0.8	[0,0,0,1,0,0,0,0]	[0,1,0,0,0,0,0,0]	[1,0,0,0,0,1,0,0]	[0,0,1,0,0,0,0,0]
0.9	[0,0,0,1,0,0,0,0]	[0,1,0,0,0,1,0,0]	[1,0,1,0,0,1,0,0]	[0,0,1,1,0,1,0,0]
0.93	[0,0,0,1,0,0,0,0]	[1,1,1,0,0,1,1,0]	[1,0,1,0,0,1,0,0]	[0,0,1,1,0,1,0,0]
0.94	[0,0,0,1,0,0,0,0]	[1,1,1,0,0,1,1,0]	[1,0,1,0,0,1,1,0]	[0,0,1,1,0,1,0,0]
0.95	[0,0,1,1,0,1,0,0]	[1,1,1,0,0,1,1,0]	[1,0,1,1,0,1,1,0]	[0,0,1,1,0,1,0,0]
0.99	[1,1,1,1,1,1,1,0]	[1,1,1,1,1,1,1,0]	[1,1,1,1,1,1,1,0]	[1,1,1,1,1,1,1,0]

Table 2. Results for Pima Diabetes Dataset for various timing vectors

α value	Timing Vectors, t			
	[1,1,1,1,1,1,1,1]	[1,100,1,1,1,1,1,1]	[100,100,1,1,1,1,1,1]	[100,100,1,1,1,1,1,100]
0.1	[0,1,0,0,0,0,0,0]	[1,0,0,0,0,0,0,0]	[0,0,0,0,0,0,0,1]	[0,0,0,0,1,0,0,0]
0.5	[0,1,0,0,0,0,0,0]	[1,0,0,0,0,0,0,0]	[0,0,0,0,0,0,0,1]	[0,0,0,0,1,0,0,0]
0.7	[0,1,0,0,0,0,0,0]	[1,0,0,0,0,0,0,0]	[0,0,0,0,0,0,0,1]	[0,0,0,0,1,1,0,0]
0.85	[0,1,0,0,0,0,0,0]	[1,0,0,0,1,1,1,1]	[0,1,0,0,0,1,0,0]	[0,1,0,0,0,1,0,0]
0.9	[1,1,0,0,0,0,0,0]	[1,1,0,0,0,1,1,0]	[0,1,0,0,0,1,0,0]	[0,1,0,0,0,1,0,0]
0.95	[1,1,0,0,0,0,0,0]	[1,1,0,0,0,1,1,0]	[1,1,0,0,0,1,1,0]	[1,1,0,0,0,1,1,0]
0.99	[1,1,0,0,0,1,1,0]	[1,1,0,0,0,1,1,0]	[1,1,0,0,0,1,1,0]	[1,1,0,0,0,1,1,0]

Fig. 1. Alignment represents the global alignment i.e Align Corr value of the case base with respect to the feature subset. The left and right graph shows Alignment v/s α values for AMPG and PID Dataset respectively.

fast thinking, it is tagged as FT . Otherwise, it is tagged as ST , indicating that it can be solved using slow thinking. A case C is tagged as N , i.e. none tag when, it can neither be solved using System 1 nor System 2. When a query q is encountered, if the majority of the k nearest neighbours to the query are tagged as FT , then the System 1 solution is predicted for the query, else it is solved using System 2. (c) Hybrid switching considers both tag-based and oracle-based switching. When majority of the nearest neighbours to the query are tagged as FT the System 1 solution is predicted. Otherwise, the feedback value is checked, and switching is done accordingly. In our previous work [12] the three switching strategies for Model 2, i.e Oracle-based, Tag-based and Hybrid Switching, lead to Model 2A, 2B and 2C respectively.

In this section, we propose a novel switching technique involving the use of domain constraints. In continuation with the previous work, we will call the model involving constraint based-switching as Model 2D. In his book, “Thinking, Fast and Slow” [9], Kahneman mentions an example where different subjects were asked, “How many murders occur in the state of Michigan in one year?” Most people responded with lower values than what they had answered when the same question was asked for Detroit, a high-crime city in the state of Michigan. Kahneman explains with this example that the fast, intuitive thinking failed to recognize that the number of crimes in Michigan will be inclusive of the number of crimes in Detroit. What if the subjects have verified their intuitive response against the domain constraints that Detroit is a high crime city in Michigan state, therefore Michigan will have at least as many crimes as Detroit? We believe they will be able to understand the fault in their intuitive answer. Similarly, when a person trying to solve a physics problem observes that his solution fails to satisfy the constraints of the laws of physics. He switches to a slower form of thinking to solve that particular problem. This motivated us to use the constraints based on domain knowledge to decide whether to invoke System 2 or not. The next subsection will describe constraint-based switching in detail.

4.1 Model 2D

Constraint: The constraint in the Detroit example mentioned above was that the no of crimes in any state would be inclusive of the number of crimes in any city of that state. Similarly, in the context of CBR, the most general constraint could be, *the similar problems will have similar solutions*. Therefore, we will apply the constraint that the cases retrieved by System 1 should be similar enough to the query with respect to all the attributes. System 1 with parsimonious CBR uses only a subset of the attributes used by System 2 to compute the similarity between any two cases. To explain the process of constraint checking, we will refer to the features that are used by System 1 as *parsimonious features* and the remaining features as *non-parsimonious features*. As a part of constraint checking, we will verify whether the most similar cases retrieved by System 1 are useful or not by applying the constraint on the similarity with respect to the *non-parsimonious features* which were neglected during retrieval. For a given query q , System 1 retrieves the k nearest neighbours. Then the similarity based

on the *non-parsimonious features* is computed between the query q and each of the k nearest neighbours. Note that this similarity is calculated based on the attributes which are not used by System 1. Suppose the similarity between the query and the nearest neighbours to the query is more than some threshold. In that case, we consider that the constraint is satisfied, and the System 1 solution is published as the reasoner’s response. Otherwise, the reasoner switches to System 2, where the nearest neighbours are retrieved using all the attributes.

In Model 2D, we have System 1 as Parsimonious CBR and System 2 as Full CBR. In this Model to decide whether to switch to System 2, along with the tag-based switching, we introduce an additional level of constraint checking explained above. In other words, it is a hybrid of tag-based switching and constraint-based switching. Given a query q , System 1 solution is published if either all the kNN to the query are tagged as *FT* or, the constraint is satisfied i.e. the similarity between the query and kNN to the query is more than some threshold. The similarity for constraint checking is based on the attributes that are not used in System 1.

4.2 Experimental Analysis

Experimental Setup: We have experimented the Model 2D i.e the hybrid model of Tag-based switching and Constraint-based switching with two classification datasets. *Red Wine Quality Dataset* (RWQ) [3] and *Fetal Health Classification Dataset* (FHC) [2] have 2 and 3 classes respectively. The RWQ has 1599 and FHC dataset 2126 data instances. This experiment aims to study the improvement in the performance i.e., accuracy and per query response time, of the proposed model by adding the constraint checking layer. We compare performance of Model 2D with basic CBR and the Model 2A, which was using only tag-based switching. The features for System 1 are computed using *Greedy Optimization Approach* mentioned in Section 3. The similarity threshold for constraint checking is computed empirically.

Experimental Results: The results for both the datasets i.e., RWQ and FHC are shown in Table 3. It can be observed that more cases are being solved using System 1 when the switching is done based on the constraints in addition to the kNN tags. Therefore, in Model 2D, the average per query response time is reduced as compared to Model 2A where switching was based only on the kNN tags. The accuracy of Model 2D is also improved as compared to 2A, therefore we can say that the Model 2D is performing better in terms of both effectiveness and efficiency.

5 Complexity Measure for Dichotomous Models

So far, we have analysed the proposed dichotomous models in terms of the time taken and prediction accuracy. In this section, we discuss an evaluation measure that can quantitatively assess the benefit of a dichotomous model over a singular model. The proposed measure is derived from the idea of using footprint size as a complexity measure as suggested in [8],[9].

Table 3. Results for Switching Strategies. "ST%" is % query cases passed on to System 2 and "Time" represents average per query response time

Model	Switching Strategy	RWQ			FHC		
		Accuracy	ST%	Time(ms)	Accuracy	ST%	Time(ms)
Model 2A	Tag-based	86.56	13.25	430.86	86.59	12.37	732.87
Model 2D	Tag or Constraint based	87.87	6.68	381.72	87.69	4.32	559.03
Full CBR		84.56		1054.94	85.36		2759.59

5.1 Footprint size as a Complexity Measure

The idea of footprint set was proposed by Smyth and McKenna [16] in the context of efficient retrieval. By definition, a footprint set is a minimal subset of the case base having the same problem-solving ability as that of the entire case base. To compute the footprint set of a given case base we need to first identify the *competence group* of every case. A brief overview of the competence model proposed by Smyth and McKenna [16] is discussed in the next paragraph.

Footprint Set: A *competence group* is a set of cases that makes an independent contribution to the competence of the whole case base and a footprint set contains the representative case(s) from every *competence group*. Construction of a competence group involves the use of the following sets of cases: *Coverage Set*, *Reachability Set* and *Related Set*. The set of all the target problems that can be solved by a case c is called the *coverage set* of c . *Reachability set* for a given target problem t is the set of the cases in the case base that can be used to solve the target problem t . A case c_1 is said to solve a case c_2 if and only if c_1 can be retrieved for c_2 and the solution of c_1 can be adapted to solve c_2 . The *Related set* of case c is the union of its *Coverage* and *Reachability* sets. Two cases c_1 and c_2 are said to exhibit a *shared coverage* if their related sets are overlapping. Any two cases having a non-empty *shared coverage* set belong to the same *competence group*. After calculating the *competence groups*, cases from each *competence group* are added to the footprint set, discarding the cases whose *coverage set* is incorporated by the other cases.

Footprint Size to Quantify Knowledge: In [8], the authors have proposed the use of footprint size as a measure of the knowledge contained in the case base. This is inline with the definition that a footprint set consists of the non-redundant cases in the case base. Further, the authors have made an observation that addition of any useful domain knowledge to the vocabulary, similarity or adaptation containers leads to a reduction in the number of non-redundant cases to be maintained in the case base. In other words, the reduction in footprint size can be used to quantify the domain knowledge added to the other containers. Footprint size can also be used to estimate the improvement in the reasoner's performance brought about by the addition of a given piece of domain knowl-

edge. In [9], the authors have used footprint size as a complexity measure to estimate the generalization ability of the reasoner. The smaller the size of the footprint, the lower the complexity of the case base and hence, the greater is the generalization ability. In the next section, we discuss how the definition of *solves* in footprint construction algorithm can be modified to accommodate the notion of time inherent to dichotomous CBR models.

5.2 Footprint with Time

An unique advantage of a dichotomous CBR model is that a reasoner could resort to fast thinking when there is a time constraint imposed on its performance. This motivated us to look for ways in which time can be factored into a complexity measure for dichotomous models. Since footprint size is a complexity measure that takes into account the interplay among the knowledge containers [8],[9], we propose a variant of the same for measuring the complexity of a dichotomous model. As we have seen earlier, a footprint set is a minimal subset of the case base that has the same problem solving ability as the entire case base. The original definition of *solves* in the footprint construction algorithm goes like this: A case c is said to solve some target problem t if and only if c can be retrieved and adapted to solve t . We propose the following modification to the definition of the *solves* relation: A case c is said to solve some target problem t if and only if c can be retrieved and adapted to solve t *within a prescribed time limit T* .

We use the modified *solves* relation in the following two contexts: firstly, evaluate the dichotomous models based on the reduction in footprint size; secondly, compare the response time per query when the modified *solves* relation is used for the selection of features. The time limit T in the modified *solves* definition is to be provided by the CBR end user. The time restrictions included in the *solves* relation will favour that dichotomous model in which System 1 is having computationally less expensive features.

5.3 Experimental Analysis

Experimental Setup: In this experiment we will evaluate the dichotomous models and Full CBR model using the proposed footprint based complexity measure. We will evaluate Models 2A, 2B and 2D, which are briefly explained in section 4. We have experimented with two different binary classification datasets i.e. *Pima Indians Diabetes Dataset* (PID) [15] and *Red Wine Quality Dataset* (RWQ) [3] with 768 and 1599 instances respectively. For models 2A, 2B and 2D, we have used the principled *Greedy Approach* mentioned in Section 3 to select the features for System 1. The footprint size is computed using the modified *solves* function where the T value can be provided as per the users requirements.

Experimental Results: Tables 4 and 5, report the results for RWQ and PID datasets respectively. We observe that the footprint set computed for model 2B is smaller than models 2A and 2D. Since model 2B has access to an external feedback value, it can efficiently switch to System 2. For a low T value, the Full CBR model has a large footprint size, indicating that such models may not

be a good choice when a low response time is needed. This evaluation measure evaluates based on the generalization ability of the reasoner considering the efficiency of System 1 and the switching strategy.

Table 4. Results for evaluation using RWQ having 1279 cases in the casebase

Model	Accuracy (%)	Average per query response time(ms)	Footprint size for $T = 1500$ ms	Footprint size for $T = 500$ ms
Full CBR	84.56	1054.94	480.2	1279
Model 2A	86.56	430.86	467	467
Model 2B	92.13	451.34	425.8	425.8
Model 2D	87.87	381.72	448	448

Table 5. Results for evaluation using PID Dataset having 614 cases in the casebase

Model	Accuracy (%)	Average per query response time(ms)	Footprint size for $T = 200$ ms	Footprint size for $T = 160$ ms
Full CBR	73.9	191.49	289.4	614
Model 2A	75.97	156.58	281.2	281.2
Model 2B	84.8	155.82	238.2	238.2
Model 2D	77.6	152.4	255	255

6 Conclusion and Future Work

We have presented three different directions that can contribute towards realizing the dichotomy of fast and slow thinking within CBR, and in particular, help in building CBR systems that flexibly trade-off time efficiency and effectiveness based on specific domain needs. *Firstly*, we have proposed an approach that can help in a principled selection of a subset of features to be used for fast thinking. We have formulated this in terms of an optimization problem with an objective function that can cater to domain specific requirements, such as prioritizing effectiveness over time efficiency or vice versa. Empirical results suggest that the approach successfully exploits large differences between similarity estimation times of diverse attributes. *Secondly*, we have proposed a novel switching mechanism that exploits domain-specific constraints to decide when to transfer control over from fast to slow thinking and demonstrated the effectiveness of this idea empirically. *Thirdly*, we have proposed a novel extension of the footprint approach that accounts for time efficiency while estimating the ability of a case to solve another; this leads to a fresh conceptual basis that allows us to quantify the gains that slow-fast dichotomy may bring about, in any given domain.

With respect to the first direction, future work can involve investigation into non-greedy approaches such as genetic algorithms or simulated annealing that incorporate randomness to explore the solution space more exhaustively. In the context of the second contribution, more work needs to go into large-scale real-world experiments that can demonstrate the effectiveness of incorporating domain constraints in the form of rules or models that encapsulate what experts already know from first principles. This can be tried out in diagnostic or planning tasks, where specific expert knowledge can be encapsulated as constraints. We envisage that the extended footprint approach presented in Section 5 may have wider applications in the context of CBR maintenance.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning and the upswing of ai (2017)
2. Ayres-de Campos, D., Bernardes, J., Garrido, A., Marques-de Sa, J., Pereira-Leite, L.: Sisporto 2.0: a program for automated analysis of cardiotocograms. *J Matern Fetal Neonatal Med* **9**(5), 311–318 (2000)
3. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.* **47**(4), 547–553 (2009)
4. Craw, S., Aamodt, A.: Case based reasoning as a model for cognitive artificial intelligence. In: ICCBR. pp. 62–77. Springer (2018)
5. Dileep, K., Chakraborti, S.: Eager to be lazy: Towards a complexity-guided textual case-based reasoning system. In: ICCBR. pp. 77–92. Springer (2016)
6. Dua, D., Graff, C.: UCI machine learning repository (2017), <http://archive.ics.uci.edu/ml>
7. Fox, S., Leake, D.: Using introspective reasoning to refine indexing. In: IJCAI. vol. 391397 (1995)
8. Ganesan, D., Chakraborti, S.: An empirical study of knowledge tradeoffs in case-based reasoning. In: IJCAI. pp. 1817–1823 (2018)
9. Ganesan, D., Chakraborti, S.: A reachability-based complexity measure for case-based reasoners. In: The Thirty-Second International Flairs Conference (2019)
10. Kahneman, D.: Thinking, fast and slow. Macmillan (2011)
11. Kannengiesser, U., Gero, J.S.: Design thinking, fast and slow: a framework for kahneman’s dual-system theory in design. *Design Science* **5** (2019)
12. Kaurav, S., Ganesan, D., Padmanabhan, D., Chakraborti, S.: Thinking fast and slow: A cbr perspective. In: 34th Florida Artificial Intelligence Research Society Conference (2021)
13. Kolodner, J.: Case-based reasoning. Morgan Kaufmann (2014)
14. Raghunandan, M., Chakraborti, S., Khemani, D.: Robust measures of complexity in tcbr. In: International Conference on Case-Based Reasoning. pp. 270–284. Springer (2009)
15. Smith, J.W., Everhart, J., Dickson, W., Knowler, W., Johannes, R.: Using the adap learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the ASCA in Medical Care. p. 261 (1988)
16. Smyt, B., McKenna, E.: Footprint-based retrieval. In: International Conference on Case-Based Reasoning. pp. 343–357. Springer (1999)
17. Zilberstein, S.: Using anytime algorithms in intelligent systems. *AI magazine* **17**(3), 73–73 (1996)