Improving Similarity-Based Retrieval Efficiency by Using Graphic Processing Units in Case-Based Reasoning

Lukas Malburg[1,2], Maximilian Hoffmann[1,2], Simon Trumm[1], Ralph Bergmann[1,2]
[1] Business Information Systems II, University of Trier, 54296 Trier, Germany
{malburgl,hoffmannm,s4sitrum,bergmann}@uni-trier.de,
http://www.wi2.uni-trier.de
[2] German Research Center for Artificial Intelligence (DFKI)
Branch University of Trier, Behringstraße 21, 54296 Trier, Germany
{lukas.malburg,maximilian.hoffmann,ralph.bergmann}@dfki.de

Abstract

The accelerated growth of available data causes case bases of increasing sizes and thus lowers efficiency during the case retrieval phase in *Case-Based Reasoning (CBR)* systems. Even though, many complex and data-intensive tasks are solved by using *Graphic Processing Units (GPUs)*, its application in CBR research has yet to advance past the early stage phase. In this paper, we present an approach to use CUDA-compatible GPUs for similarity assessment of structural, feature vector based cases. Our approach supports several syntactic and semantic similarity measures and is implemented in the open-source case-based reasoning framework ProCAKE. When comparing to current retrieval techniques that calculate similarities on the CPU, our GPU-based approach outperforms them by a factor of up to 37. In addition, our evaluation indicates that the performance gains increase with higher case complexity.

1 Introduction

Today, the rapidly growing amounts of data result in case bases of increasing sizes in *Case-Based Reasoning (CBR)* systems, which in turn multiplies the computation efforts during case retrieval. However, a critical factor for the efficiency of a CBR system is its performance during the aforementioned phase of case retrieval (Dalal, Athavale, and Jindal 2011) and high retrieval times cause lower user acceptance rates. Additionally, performance and accuracy of case retrieval also influences the speed and quality of learning adaptation knowledge from case bases (Hanney and Keane 1997).

Our previous work investigates the acceleration of retrieval by applying MAC/FAC techniques (Hoffmann et al. 2020; Klein, Malburg, and Bergmann 2019) and by improving heuristics used during case retrieval (Zeyen and Bergmann 2020). Due to the development of more powerful *Graphics Processing Units (GPUs)* in recent years, its utilization in both academia and industry is common. With GPUs playing an important role in various fields of computer science such as deep learning (Shi et al. 2016) or high performance computing (Kindratenko and Trancoso 2011), they are especially useful for highly parallelizable and possibly complex tasks. The procedure of similarity assessment during case retrieval can naturally be parallelized since the similarity computation between a query and a case from the case base is independent of other cases in the same case base. There exists some related work that uses GPUs for distributed CBR in the context of workflow monitoring (e. g., Agorgianitis et al. 2017) or tasks similar to case retrieval like k-nearest neighbors (e. g., Liang et al. 2009; Li and Amenta 2015). In this paper, we present and evaluate a novel GPU-based retrieval approach for the retrieval phase in CBR. The approach supports structural, feature vector based case representations and is implemented in the open-source CBR system ProCAKE (Bergmann et al. 2019).

In the following, previous work on using GPUs in CBR and other related fields is presented. Furthermore, the structural case representation in ProCAKE is described in detail. In Sect. 3, we present our approach for using GPUs to accelerate case retrieval in CBR. Here, we introduce how cases are pre-processed and how the data exchange is performed between CPU and GPU. An experimental evaluation of our approach is presented in Sect. 4. Finally, Sect. 5 concludes the results and discusses future work.

2 Foundations and Related Work

Current research on the acceleration of similarity-based retrieval in *Case-Based Reasoning (CBR)* mainly addresses slow retrieval times by using MAC/FAC (Forbus, Gentner, and Law 1995) techniques (e. g., Bach et al. 2016; Hoffmann et al. 2020; Klein, Malburg, and Bergmann 2019) or other index-based approaches (see Bergmann 2002). Since we tackle this problem by shifting the similarity calculation from CPU to GPU, we first present two important aspects of the similarity calculation, i. e., the structural case representation and the corresponding similarity measures. Afterwards, related work in the area of GPU-based acceleration of complex problems such as case retrieval in CBR is presented.

Case Representation and Similarity Assessment

In our approach, we restrict ourselves to structural case representations. Figure 1 shows an exemplary case in attribute-value fashion that represents a car. Each car

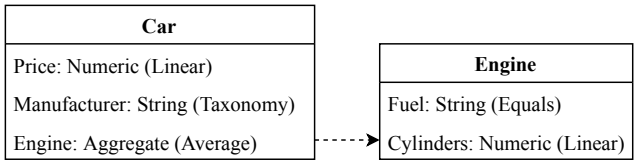

Figure 1: Exemplary Attribute-Value Representation of a Car.

has several attributes, e. g., the price, the manufacturer, or the type of fuel. These attributes are represented by *Atomic* data types, i. e., strings or numeric values, or by combining several atomic data types to composite data types such as *Aggregate* types. An *Aggregate* data type such as the attribute `Engine` consists in turn of one or more *Atomic* data types, i. e., `Fuel` and `Cylinders`. The similarity assessment of structural cases follows the local-global principle proposed by Richter (Richter 1998) where the local similarities of individual attributes on the lowest level of the case representation are first determined, i. e., `Fuel` and `Cylinders`, and aggregated afterwards. Here, the local similarities of the parent *Atomic* and *Composite* types are calculated, i. e., `Price`, `Manufacturer`, and the similarity for the *Composite* type `Engine`, which aggregates the similarities of `Fuel` and `Cylinders`. All similarity calculations are independent from each other and could also be performed partially simultaneously. Figure 2 illustrates an exemplary similarity calculation between a query Q and a case C. On the lowest level, the pairwise similarities

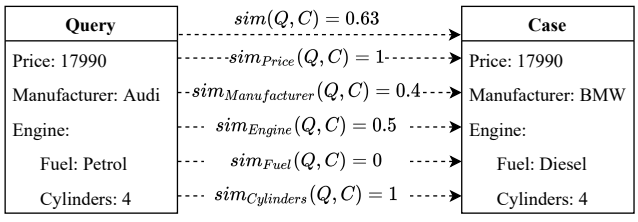

Figure 2: Similarity Assessment based on Local-Global Principle for a Query-Case Pair.

between the attributes `Cylinders` and `Fuel` are determined and aggregated with an equal weighting to the similarity for the attribute `Engine`, i. e., 0.5. Further, the similarity value of 0.5 is combined with the pairwise similarity values of the attributes `Manufacturer` and `Price` to form the global equally weighted average similarity value of 0.63 between query and case. The similarity measures used in this example are linear numeric measures and a string equality measure. For the attribute `Manufacturer`, a taxonomic similarity measure that is based on a semantic taxonomy (Bergmann 2002) of manually annotated similarities is applied.

Related Work

The application of GPUs besides its originally intended purpose of rendering images gains attraction in both academia and industry. GPUs are utilized in several research fields from an academic perspective, especially for highly parallelizable workloads. Liang et al. (2009) present an approach that uses the GPUs for a parallel *k-Nearest Neighbor (KNN)* search called *CUDA-based parallel KNN (CUKNN)*. They show in their evaluation that a speedup of 46.71 times is possible. However, they only use synthetic data and no real-world datasets from a practice-oriented point of view. Li and Amenta (2015) also introduce an approach that utilizes a truncated merge sort algorithm from a GPU library to find the k-nearest neighbors. The approaches are not specifically tailored for CBR applications and mainly focus on numerical data and not complex structural, feature vector based case representations. In context of CBR, GPU-based acceleration is widely used for combining CBR and *Deep Learning (DL)* techniques, e. g., for explainable AI research (e. g., Keane and Kenny 2019; Nugent and Cunningham 2005) or for learning similarity measures (e. g., Amin et al. 2019). Additionally, our own previous work (Hoffmann et al. 2020; Klein, Malburg, and Bergmann 2019) draws the computational power of GPUs to learn workflow embeddings with the application of deep learning techniques during the similarity-based retrieval in CBR. However, since GPU-based acceleration is part of the DL framework in these applications and it is not specifically customized for the CBR components, we clearly differentiate the approach shown here from the ones mentioned before. To the best of our knowledge, there is only one publication by Agorgianitis et al. (2017) that exploits the computational power of GPUs in the context of CBR. The goal is to monitor business processes by implementing a distributed CBR approach. The GPU performs the similarity calculation during the retrieval phase to assess the similarity between workflows. However, the approach focuses on decentralization and the use of a distributed infrastructure for workflow monitoring instead of the retrieval phase to determine the similarity of cases exclusively. In our approach, we use KNN similar to the presented related work for the retrieval phase in a CBR system. Finally, we consider several similarity measures and notes for their application on GPUs.

3 Similarity-Based Retrieval by Using Graphic Processing Units

In this section, we present our approach for performing a similarity-based retrieval on a GPU to accelerate retrieval in CBR. We describe our GPU retrieval framework, the specific data preparation for this retrieval context, and different types of similarity measures.

Framework Overview

Our approach uses CUDA (Nickolls et al. 2008; Nvidia 2020) as the programming interface for the GPU. Figure 3 illustrates an overview of the system components for our GPU approach. The relevant data for performing

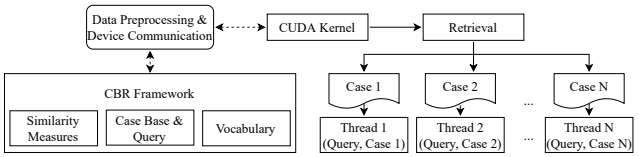

Figure 3: Framework for GPU-Based Retrieval in Case-Based Reasoning.

a retrieval, i. e., the case base and query, the vocabulary, and the used similarity measures, are modeled and stored in a CBR framework. The corresponding kernels that are executed during a retrieval are defined on the GPU. Both components, i. e., the CBR framework and CUDA, are connected via a data preparation component that handles encoding and transfer of information from CPU to GPU and back again. In our approach, a retrieval is run as follows: In an offline phase, the case base and the information regarding the similarity measures and the vocabulary are encoded, transferred to the GPU, and stored in the on-chip memory. We assume a static case base in our explanations but it is also possible to do dynamic changes to the case base in GPU memory, e. g., to add single cases. It may be necessary to further process the transferred data, e. g., to build a taxonomic data structure for some similarity measures (see Sect. 2). The case base and all additional information are stored on the GPU for the complete lifetime of the application. In this context, we assume that the data completely fits into the graphics RAM, which is usually no problem even for large case bases. The online phase starts when a query is submitted. First, the query case is created, encoded, and transferred to the GPU. After that, the similarity computation between the query case and all cases from the case base is started. Our approach utilizes the parallelization possibilities of GPUs by computing the similarity for each individual query-case pair (see Fig. 2) on a separate GPU thread (similar to Liang et al. 2009). Consequently, each thread generates a similarity for a respective case by using the given data about the similarity measures, including the aggregation of local similarity values to a single global similarity. The case similarity values are afterwards transferred back to the CPU. As a last step, the cases are sorted, the retrieval result is finalized, and the k-most similar cases are returned to the user by the CBR framework.

Data Preparation for GPU Retrieval

The process of GPU-based retrieval, as introduced before, requires the case base and query data to be prepared for this specific scenario. That is mainly due to the step of data transfer from CPU's RAM to GPU's graphics RAM. Therefore, it is necessary to encode the abstract data to a simple format of n-dimensional arrays (Nvidia 2020) that can be transferred to the GPU via built-in CUDA methods. Those arrays can be quickly accessed and are efficient in terms of required storage space. An example of two encoded cases is shown in Fig. 4. Case data is split-up in two arrays where the first ar-

Numeric	17990	4	17990	4
String	"Audi"	"Petrol"	"BMW"	"Diesel"

Figure 4: Exemplary Encoded Case Data for Similarity Computation on the GPU.

ray stores numeric data as 32-bit floating point values and the second array stores string data. Although the figure depicts an array of strings, this array contains chunks of char data that are concatenated to one large array. This concatenated array can be divided again after transfer to the GPU by storing another array. The additional array holds numbers that indicate the character length of each individual string in the concatenated array. Only the *Atomic* values of case attributes are part of the numeric and the string array. Values of individual attributes of the case representation always have the same order in these arrays across all encoded cases. Thus, it is ensured that each case is encoded in the same way. The order is given by the structural meta data of the cases, defined in the vocabulary that is also transferred to the GPU.

The encoding example shown in Fig. 4 reuses the query-case pair from Fig. 2. Consequently, the encoded data refers to these two cases with the numeric values being a price of 17990 and 4 cylinders for the query and a price of 17990 and also 4 cylinders for the case. The string values express an Audi with petrol fuel for the query and a BMW with diesel as fuel for the exemplary case. These array entries are placed according to the vocabulary: it describes that the car's price is always located in the first numeric array position belonging to one case and the number of cylinders in the second position. Similarly, for the array of strings, the manufacturer is located in the first position and the type of fuel in the second position. A clear definition of the position of attribute values in these arrays is essential since similarity measures solely rely on this information to compute similarities. It is important to note that missing attribute values are represented by using a special *Atomic* type. Therefore, the attribute sequence is always guaranteed even if values in cases are missing.

GPU-Accelerated Similarity Measures

In order to compute similarities between a query and the corresponding cases, the GPU has to execute kernels that compute these similarities. As opposed to CPU-based applications where the case data is represented in a structural, object-oriented fashion, the data on the

GPU utilizes an array-based, feature vector based representation of case data. This means that we implemented all similarity measures necessary to work with this data structure. The global aggregated similarity of two cases is computed in a bottom-up approach (see Sect. 2 for an example). Starting with the local similarities of all atomic values, i. e., the similarities of individual attributes, the similarities are iteratively aggregated to eventually result in one global similarity for the query-case pair. When applying this approach to the exemplary similarity assessment from Fig. 2 with the encoded data from Fig. 4, the similarity measures work as follows: First, the local similarities for the *Atomic* attributes `Fuel`, `Cylinders`, `Price`, and `Manufacturer` are computed (0, 1, 1, and 0.4). Afterwards, the local similarity of the aggregate attribute `Engine` is computed by taking the average of the similarities of `Fuel` and `Cylinders` (0.5). Finally, the global similarity of 0.63 is computed by averaging the equally weighted similarities of `Manufacturer`, `Price`, and `Engine`. The approach also supports other arbitrary attribute weightings. The previous example shows another important aspect in applying these similarity measures: When operating on the encoded case data, most of the similarity measures also require additional data to work properly, e. g., lower and upper bounds for a numeric measure or complex taxonomic hierarchies. For each of those cases, the individually encoded data might have to be processed on the GPU prior to executing the similarity measures.

4 Experimental Evaluation

To evaluate our approach, we measure the retrieval time of different retriever implementations with a specific focus on the comparison of retrievers that only use CPU computing and our retrieval approach that uses GPU computing. The evaluated retrievers comprise a serial CPU retriever that operates on a single thread (single thread CPU retriever; *SCR*), a parallel CPU retriever that uses all available threads of the CPU (parallel CPU retriever; *PCR*), and a GPU retriever that computes all similarities on GPU hardware (GPU retriever; *GR*). We investigate the following hypotheses in our experiment:

H1 Using *GR* outperforms all other retrievers that completely or partially operate on the CPU w. r. t. retrieval times.

H2 With increasing complexity of the case representation, the retrieval time of *GR* increases less than the retrieval time of the CPU approaches.

Experimental Setup

For our experiments, we implemented the previously presented approach in the ProCAKE CBR framework[1] (Bergmann et al. 2019), which supports structural and process-oriented CBR applications with cases represented as attribute-value pairs and cases represented as semantic graphs (Bergmann and Gil 2014), respectively. Since ProCAKE is implemented in Java we use the bindings of JCUDA (Yan, Grossman, and Sarkar 2009), but, however, the CUDA core can be flexibly integrated into other CBR frameworks as well. As already mentioned, we restrict ourselves to structural, feature vector based cases of cars and model two different case representations to define them. The cases are structured similarly to the example shown in Sect. 2 and they originate from several CSV datasets on Kaggle. The simple case representation (CR-I) has 12 string and 12 numeric attributes where 6 of the numeric attributes are part of an aggregate object inside the main aggregate case object. The more complex case representation (CR-II) has 308 string and 408 numeric attributes where 154 of the numeric attributes are part of an aggregate attribute. Those case representations are chosen to investigate trends in retrieval time for cases with different representation complexity. Throughout the experiment, we use 5 different case base sizes for each representation CR-I and CR-II, ranged from 10,000 to 50,000 cases with steps of 10,000 cases. We measure the retrieval time as the central performance metric. Therefore, first, an arbitrary query is created for both case representations CR-I and CR-II. Then, the retrieval time of each combination of retriever and case base is measured by executing a retrieval scenario with the respective data. This means that the time of each of the 3 retrievers is measured for each of the 5 case base sizes for the case representations CR-I and CR-II. To avoid distortions of individual retrieval times, we repeat each retrieval 10 times with the same query and take the average of the resulting values. The GPU retriever *GR* uses a case base and query that are allocated on the GPU memory in an offline phase that does not contribute to the measured time. It is important to note that the same applies to the CPU where the data is stored in memory directly. Additionally, we ensure correct computation results for *GR* by comparing its computed similarities on an attribute level and a case level with similarities that are computed on the CPU. All experiments are conducted on a computer with an Intel i7 6700 CPU (4 cores, 8 threads) with 48 GB RAM and an NVIDIA GTX 1080 GPU with 2560 CUDA cores and 8 GB graphics RAM, running Windows 10 64-bit. This represents a common configuration for a regular PC that enables reusing our approach in many fields.

Experimental Results

The most important result of our experiments concerns a runtime comparison of different retrievers. Figure 5 shows the averaged retrieval durations (vertical axis) of the evaluated retrievers for differently sized case bases (horizontal axis) with the simple case representation CR-I (solid lines) and the more complex case representation CR-II (dotted lines). In the diagram, *PCR* is depicted as a line with attached rectangles and *GR* as a line with attached circles. For reasons of clarity and

[1] See `http://procake.uni-trier.de`

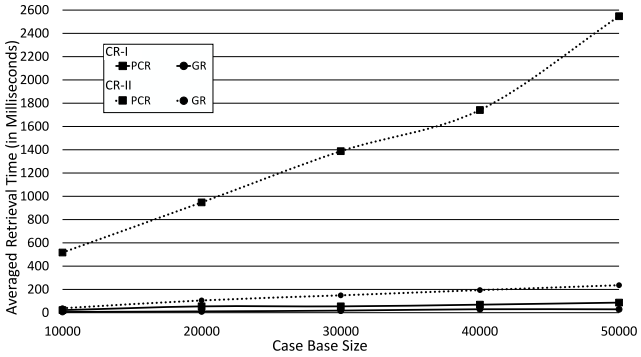

Figure 5: Retrieval Times of CPU- and GPU-Based Approaches for Case Representations CR-I and CR-II.

scale distortions for CR-II, we have not added the significantly slower SCR in the figure. For a detailed performance comparison of SCR with PCR and GR see Tab. 2 instead. It is apparent that *PCR* shows the longest retrieval time of the diagram for all case bases of case representation CR-I. For the same case representation, *GR* shows a much shorter retrieval time for all case base sizes and, thus, outperforms *PCR*. The results for the more complex case representation CR-II are very similar to those of case representation CR-I: *GR* outperforms *PCR* across all case bases. However, the retrieval times for CR-II are in general higher than those of CR-I, which is caused by the more complex case representation and the higher number of attributes. The retrieval times show an average speed-up of *GR* compared to *SCR* and *PCR* between 2 and 37 times depending on the retriever and the case representation (see Tab. 1). With a maximum of 1.3 GB of occupied graphics RAM, the approach is also efficient in terms of required storage capacity. Due to these results, Hypothesis H1 can be clearly accepted for all case base sizes of case representation CR-I and the more complex representation CR-II.

Table 1: Comparison of Speed-Up for Case Representations CR-I and CR-II.

Case Representation	Retriever		
	GR vs. PCR	GR vs. SCR	PCR vs. SCR
CR-I	2.04	10.98	2.95
CR-II	8.88	37.45	2.89

Since the retrieval scenarios in our experiments only represent a small number of all possible real-world scenarios, it is difficult to foresee how well our approach scales, especially when regarding larger case bases or cases with an even higher number of attributes. Therefore, in addition to measuring the absolute retrieval times, we investigate the increase of computation time for all retrievers by comparing retrieval times of equally sized case bases for the simple case representation CR-I and for the more complex case representation CR-II (see Tab. 2). For instance, the first value from the left in the row of *SCR* stems from the rate of time increase, i. e., $\frac{(1951-72)\cdot 100}{72}$, when comparing the difference in retrieval time of 10,000 cases structured according to CR-I or CR-II. The computed values show by how much the time increases, relative to the retrieval time for the simpler case representation CR-I. Since the number of attributes in the more complex case representation CR-II has increased by 2883 % compared to case representation CR-I, a perfectly proportional relationship is expressed by an increased relative retrieval time of 2883 %. In order to generate a single metric to compare across all retrievers, we averaged the values for each retriever (see "Avg." in Tab. 2). The average results

Table 2: Increase of Retrieval Time for a More Complex Case Base.

Retriever	Number of Cases (x1000)					Avg.
	10	20	30	40	50	
SCR	2598%	2677%	2436%	2250%	2232%	**2439%**
PCR	2250%	1644%	2454%	2424%	2839%	**2322%**
GR	505%	829%	728%	554%	714%	**666%**

in Tab. 2 show that *SCR* performs worst with a value of 2439 %. Increasing the number of attributes seems to only have a minor scaling effect for *SCR*. *PCR* shows an average value of 2322 %, which shows a slightly higher scaling effect than *SCR*. The best value is achieved by *GR* with an average increase in retrieval time of 666 %. This value shows great potential of our approach since a higher number of attributes and, thus, a computationally more complex problem seems to lead to great scaling effects in retrieval time. The reason for this might be the GPU's architecture that is highly specialized for parallel workloads, i. e., simultaneous similarity computations of query-case pairs, and large amounts of data, i. e., computation of local attribute similarities within a single thread. Furthermore, we observed in our experiments that numeric attributes are processed much faster on the GPU than on the CPU. For string attributes, the performance gain is smaller but still significant. Therefore, we clearly accept Hypothesis H2.

5 Conclusion and Future Work

We presented an approach to accelerate the similarity assessment during case retrieval in *Case-Based Reasoning (CBR)* by using *Graphic Processing Units (GPUs)*. The approach allows to use structural, feature vector based case representations with syntactic and semantic similarity measures. The acceleration of the retrieval might also affect other phases of CBR, e. g., the performance of learning adaptation knowledge. In our experimental evaluation, we measured retrieval times and could show that the presented approach clearly outperforms current retrieval techniques that assess the similarity on the CPU. We prototypically implemented the GPU-based retrieval in the open-source CBR framework ProCAKE in order that it can be used by others.

In future work, we intend to publish the CUDA core for integration into other CBR frameworks. Furthermore, the approach should be enhanced to support more complex case representations that exceed traditional attribute-value based representations, e. g., semantic graphs (Bergmann and Gil 2014) as cases. Especially the revealed scaling potential for more complex case representations (see Hypothesis H2) encourages the assumption that there are significant performance gains possible during the complex and NP-hard computations for subgraph matching (see Ontañón 2020). The use of machine learning frameworks such as TensorFlow[2] as a computational foundation for further performance improvements and more device independence is considered in future work. The combination of the presented approach with our already developed techniques (e. g., Hoffmann et al. 2020) or generally in parallel with CPU computing promises further enhancements during case retrieval in CBR systems.

References

Agorgianitis, I.; Kapetanakis, S.; Petridis, M.; and Fish, A. 2017. Business Process Workflow Monitoring Using Distributed CBR with GPU Computing. In *Proc. of the 30th Int. FLAIRS Conf.*, 495–498. AAAI Press.

Amin, K., and et al. 2019. Advanced Similarity Measures Using Word Embeddings and Siamese Networks in CBR. In *Proc. of the Intell. Syst. Conf., Vol. 2*, volume 1038 of *Adv. in Intell. Syst. Comput.*, 449–462. Springer.

Bach, K.; Szczepanski, T.; Aamodt, A.; Gundersen, O. E.; and Mork, P. J. 2016. Case Representation and Similarity Assessment in the selfBACK Decision Support System. In *Proc. of 24th ICCBR 2016*, volume 9969 of *LNCS*, 32–46. Springer.

Bergmann, R., and Gil, Y. 2014. Similarity assessment and efficient retrieval of semantic workflows. *Inf. Syst.* 40:115–127.

Bergmann, R.; Grumbach, L.; Malburg, L.; and Zeyen, C. 2019. ProCAKE: A Process-Oriented Case-Based Reasoning Framework. In *Workshop Proc. of 27th ICCBR 2019*, volume 2567, 156–161. CEUR-WS.org.

Bergmann, R., ed. 2002. *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*, volume 2432 of *LNCS*. Springer.

Dalal, S.; Athavale, V.; and Jindal, K. 2011. Case retrieval optimization of Case-based reasoning through Knowledge-intensive Similarity measures. *Int. J. Comput. Appl.* 34(3):12–18.

Forbus, K. D.; Gentner, D.; and Law, K. 1995. MAC/FAC: A Model of Similarity-Based Retrieval. *Cogn. Sci.* 19(2):141–205.

Hanney, K., and Keane, M. T. 1997. The Adaptation Knowledge Bottleneck: How to Ease it by Learning from Cases. In *Case-Based Reason. Res. and Dev.*, 359–370. Springer.

Hoffmann, M.; Malburg, L.; Klein, P.; and Bergmann, R. 2020. Using Siamese Graph Neural Networks for Similarity-Based Retrieval in Process-Oriented Case-Based Reasoning. In *Proc. of 28th ICCBR 2020*, volume 12311 of *LNCS*, 229–244. Springer.

Keane, M. T., and Kenny, E. M. 2019. How Case-Based Reasoning Explains Neural Networks: A Theoretical Analysis of XAI Using Post-Hoc Explanation-by-Example from a Survey of ANN-CBR Twin-Systems. In *Proc. of 27th ICCBR 2019*, volume 11680 of *LNCS*, 155–171. Springer.

Kindratenko, V., and Trancoso, P. 2011. Trends in High-Performance Computing. *Comput. Sci. Eng.* 13(3):92–95.

Klein, P.; Malburg, L.; and Bergmann, R. 2019. Learning Workflow Embeddings to Improve the Performance of Similarity-Based Retrieval for Process-Oriented Case-Based Reasoning. In *Proc. of 27th ICCBR 2019*, 188–203. Springer.

Li, S., and Amenta, N. 2015. Brute-Force k-Nearest Neighbors Search on the GPU. In *Similarity Search and Applications*, volume 9371 of *LNCS*. Springer. 259–270.

Liang, S.; Liu, Y.; Wang, C.; and Jian, L. 2009. A CUDA-based parallel implementation of K-nearest neighbor algorithm. In *Int. Conf. on Cyber-Enabled Distrib. Comput. and Knowl. Discov.*, 291–296. IEEE.

Nickolls, J.; Buck, I.; Garland, M.; and Skadron, K. 2008. Scalable Parallel Programming with CUDA. *ACM Queue* 6(2):40–53.

Nugent, C., and Cunningham, P. 2005. A Case-Based Explanation System for Black-Box Systems. *Artif. Intell. Rev.* 24(2):163–178.

Nvidia. 2020. CUDA Programming Guide. Version 11.0.

Ontañón, S. 2020. An overview of distance and similarity functions for structured data. *Artif. Intell. Rev.* 53(7):5309–5351.

Richter, M. M. 1998. Introduction. In *Case-Based Reason. Technol.*, volume 1400 of *LNCS*. Springer. 1–15.

Shi, S.; Wang, Q.; Xu, P.; and Chu, X. 2016. Benchmarking State-of-the-Art Deep Learning Software Tools. In *7th Int. Conf. on Cloud Comput. and Big Data*, 99–104. IEEE.

Yan, Y.; Grossman, M.; and Sarkar, V. 2009. JCUDA: A Programmer-Friendly Interface for Accelerating Java Programs with CUDA. In *Euro-Par 2009 Parallel Process.*, volume 5704 of *LNCS*. Springer. 887–899.

Zeyen, C., and Bergmann, R. 2020. A*-based Similarity Assessment of Semantic Graphs. In *Proc. of 28th ICCBR 2020*, volume 12311 of *LNCS*, 17–32. Springer.

[2] `https://tensorflow.org`