Investigando a Busca de Casos em Sistemas CBR via Algoritmos que Aceleram o *k*-NN

Raphael Lopes de Souza[1], Osvaldo Luiz de Oliveira[1]

[1]Centro Universitário Campo Limpo Paulista (UNIFACCAMP)
Rua Guatemala, 167, Jardim América – 13.231-230 – Campo Limpo Paulista – SP – Brasil

raphaellopesdesouza@gmail.com, osvaldo@faccamp.br

Abstract. *Case-based Reasoning Systems (CBR) use the solution of old problems to solve a new problem. The case concept is central to CBR systems and refers to a unit composed of a problem and a solution. To solve a new problem, CBR systems generally work with case base of large size and dimension. These systems often use k-NN-based (k-nearest neighbor) algorithms to search, on a case base, for an old case similar to a new problem. A known difficulty in the CBR systems design is that the k-NN algorithm is very time-consuming to perform searches in case base of large size and dimension. This article describes a work in progress that aims to conduct a theoretical and experimental study about the efficiency, memory use, accuracy and applicability, in the CBR systems design, of recent algorithms that were designed to accelerate the k-NN algorithm.*

Resumo. *Sistemas para raciocínio baseado em casos (do Inglês CBR, case-based reasoning) utilizam a lembrança da solução de problemas antigos para resolver um problema novo. O conceito de caso é central aos sistemas CBR e refere-se a uma unidade composta por um problema e sua solução. Para raciocinar a solução de um problema novo, sistemas CBR trabalham, de forma geral, com base de casos de alta dimensão e tamanho. Estes sistemas frequentemente utilizam algoritmos baseados no algoritmo k-NN (do Inglês, k-nearest neighbor) para buscar um caso antigo similar a um problema novo. Uma conhecida dificuldade no projeto de sistemas CBR é que o algoritmo k-NN consome muito tempo para realizar buscas em base de casos de alta dimensão e com muitos casos. Este artigo descreve um trabalho em desenvolvimento que objetiva realizar um estudo teórico e experimental sobre a eficiência, o consumo de memória, a acurácia e a aplicabilidade, no projeto de sistemas CBR, de algoritmos desenvolvidos para acelerar o algoritmo k-NN.*

1. Introdução

Sistemas para raciocínio baseado em casos (do Inglês, *case-based reasoning*) têm sido aplicados com sucesso em sistemas de suporte a tomada de decisão, classificação, recomendação, planejamento, entre outras áreas (Montani & Jain, 2010). A ideia que fundamenta e motiva o desenvolvimento de sistemas CBR é a de que problemas novos

podem ser resolvidos via a solução de problemas antigos. Kolodner (1993), em artigo seminal, argumenta que sistemas CBR se estabelecem sobre dois princípios básicos: (1) problemas similares possuem soluções similares; (2) problemas tendem a se repetir.

Sistemas CBR operam sobre casos armazenados em uma memória, frequentemente nomeada de base de casos, residente em disco, memória RAM ou outra mídia. Um caso é a unidade fundamental dos sistemas CBR, sendo composto por um problema e pela solução ao problema. Basicamente, o que um sistema CBR faz para desenvolver a solução de um novo problema é o seguinte: (1) busca, na base de casos, o caso cujo problema é mais similar ao problema novo; (2) utiliza a solução do caso antigo para resolver o problema novo, eventualmente adaptando a solução do problema antigo à solução do problema novo.

Uma estratégia comumente empregada por sistemas CBR é a de popular a base de casos com um número grande de casos de modo que, na presença de um problema novo, a chance de se encontrar um caso com problema muito similar é majorada e a necessidade de adaptação da solução antiga ao problema novo é minorada. No entanto, esta estratégia introduz o desafio da busca de casos, em um tempo factível, em bases de casos massivamente populadas e com alta dimensão (problemas descritos por muitos atributos). Este desafio de busca é um tópico "quente" da agenda de pesquisa atual sobre sistemas CBR (Bichindaritz, Marling & Montani, 2017; Goel & Diaz-Agudo, 2017).

Diversas soluções têm sido propostas para implementação do mecanismo de busca de casos em sistemas CBR. O uso do algoritmo *k*-NN está entre uma destas soluções. O mecanismo de busca do algoritmo *k*-NN é simples e consiste apenas na computação de uma função de similaridade entre o problema novo e todos os casos armazenados na base de casos para, ao final do processo, retornar $k \geq 1$ casos mais similares a um problema novo. Todavia, esta simplicidade impõe um grande consumo de tempo em sistemas CBR constituídos de bases de casos com grande tamanho e dimensão. Algoritmos têm sido projetados para acelerar o algoritmo *k*-NN e, assim, tornar factível o processo de busca em sistemas CBR. Este trabalho em desenvolvimento descreve os passos iniciais de um estudo teórico e experimental sobre algoritmos projetados para acelerar o algoritmo *k*-NN. O objetivo do estudo é investigar a eficiência, o consumo de memória, a acurácia e as condições em que tais algoritmos podem ser empregados no projeto de sistemas CBR.

O restante deste artigo está organizado da seguinte maneira. A Seção 2 descreve o que são e como funcionam os sistemas CBR. A Seção 3 descreve trabalhos relacionados com o mecanismo de busca empregado por sistemas CBR. A Seção 4 apresenta as ideias, ainda em fase de amadurecimento, do estudo teórico e experimental que é o objeto deste trabalho. A Seção 5 apresenta considerações finais deste trabalho em desenvolvimento.

2. Sistemas CBR

Sistemas CBR operam sobre casos. Um caso, unidade composta por problema e solução, é descrita em uma base de casos via a representação de um problema e de sua

solução. Um problema é representado por uma sequência de características (atributos[1]) que o expressam sobre uma certa perspectiva. Uma solução é comumente representada por um texto, em linguagem natural ou formal, descrevendo a solução do problema. A Figura 1 ilustra o ciclo típico dos sistemas CBR e é uma adaptação de uma ilustração clássica, presente em Aamodt & Plaza (1994). O ciclo compreende o seguinte, dado um problema novo, o sistema CBR: (1) recupera da base de casos um caso antigo[2] cujo problema é similar ao problema novo; (2) reusa a solução do caso antigo para produzir uma solução nova, eventualmente adaptando a solução antiga ao contexto do problema novo; (3) Revisa a solução nova com objetivo de confirmar se ela, ao ser aplicada no mundo real, soluciona ou não o problema novo; (4) Em caso da solução ser bem-sucedida no mundo real, retém o caso novo (problema novo e solução nova confirmada) na base de casos. É pouco comum um sistema CBR ser projetado para executar de maneira automática todas as fases do ciclo expresso na Figura 1. Frequentemente as fases de revisão e retenção são executadas totalmente ou parcialmente por usuários e especialistas do domínio do sistema CBR.

O foco deste trabalho está na fase de recuperação. É nesta fase que se dá a busca de um ou mais casos similares a um problema novo. A fase de reuso utiliza métodos de adaptação para reorganizar, adicionar, excluir ou combinar as características de soluções de problemas antigos visando o desenvolvimento de uma solução nova. Estratégias de adaptação são fortemente dependentes do domínio do sistema CBR e não é simples generalizar sobre elas. A forma mais simples de adaptação é a adaptação nula que, como o próprio nome sugere, significa não adaptar nada na fase de reuso. Quando a estratégia de adaptação nula é utilizada na programação de um sistema CBR, possíveis modificações são deixadas a cargo de usuários e especialistas no domínio do sistema CBR. A fase de revisão envolve: (1) a avaliação da solução gerada na fase de reuso; (2) o reparo da solução gerada na fase de reuso, caso tal solução não seja considerada bem-sucedida. A avaliação da fase de reuso é sempre algo externo ao sistema CBR, isto é, a solução do problema é avaliada no mundo real e, dependendo do sistema CBR, pode durar segundos, horas, dias ou meses. O propósito da fase de retenção é aperfeiçoar o desempenho do sistema CBR, via o aprendizado proporcionado pelo incremento de casos novos na base de casos.

O seguinte exemplo, adaptado de Von Wangenheim, Von Wangenheim & Rateke (2013), ajuda a concretizar os conceitos envolvidos em um ciclo típico de um sistema CBR. Considere um sistema CBR, digamos *S*, desenvolvido para dar suporte ao diagnóstico de falhas e ao reparo em impressoras do tipo jato de tinta. Considere também que o sistema CBR *S* opera sobre a base de casos *BC* composta, inicialmente, pelos dois casos apresentados na Figura 2. A representação usada na Figura 2, expressa cada caso entre parênteses angulares "< >". Dentro de cada caso, entre um par de parênteses, "( )", estão descritas as características do problema. O primeiro caso refere-se a uma impressora do modelo "RM800", cuja descrição é "Não imprime em preto", estando as luzes de estado do papel, nível de tinta preta e nível de tinta colorida,

[1] Neste artigo os termos "característica" e "atributo" são sinônimos. A literatura de Aprendizado de Máquina utiliza frequentemente o termo "atributo". Estamos dando preferência ao termo "característica" por ser mais comumente utilizado em textos específicos sobre CBR.

[2] Alguns sistemas CBR são desenvolvidos para operar recuperando dois ou mais casos da base de casos e não, apenas, um caso.

respectivamente, no estado “Apagada”, “Acesa” e “Apagada”. A solução deste primeiro caso está descrita em linguagem natural via a frase “O diagnóstico é de que o cartucho de tinta preta está vazio. O problema pode ser resolvido substituindo-se o cartucho de tinta preta”. O segundo caso expresso na Figura 2 pode ser entendido de forma análoga.

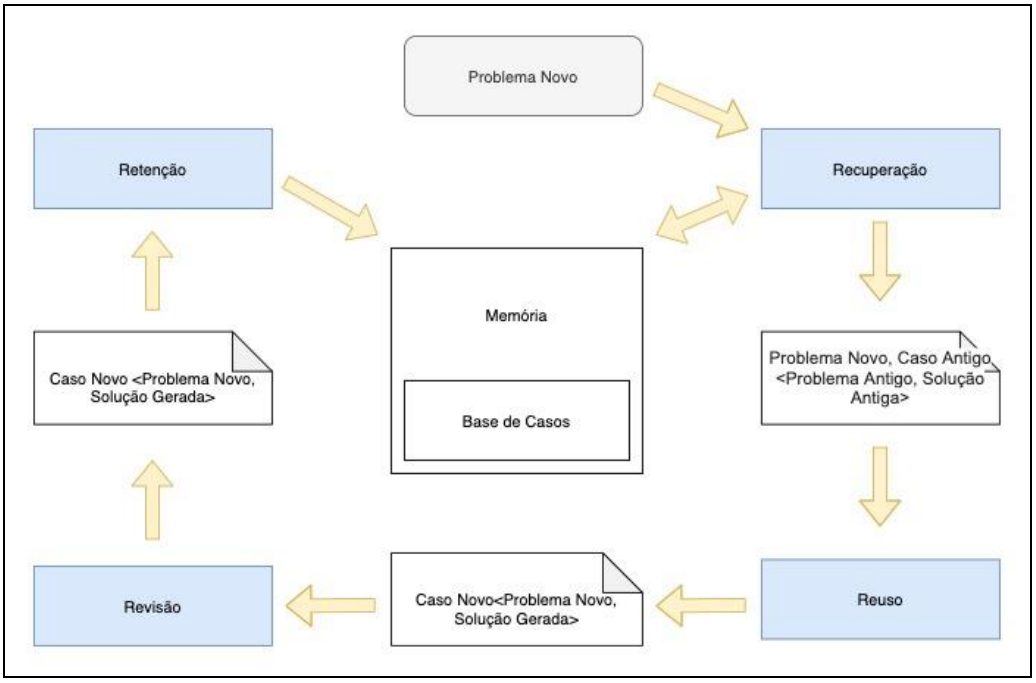

Figura 1 – Ciclo Típico de um Sistema CBR (adaptado de Aamodt & Plaza (1994))

Considere que o sistema CBR *S* recebeu, como entrada, um problema novo no qual uma impressora do modelo “RM800” não imprime em cor preta, estando as luzes de estado do papel, nível de tinta preta e nível de tinta colorida, respectivamente, no estado “Apagada”, “Acesa” e “Apagada”. Este problema é idêntico ao problema do primeiro caso expresso na Figura 2. Assim, o sistema CBR *S*: (1) na fase de recuperação, recupera o primeiro caso; (2) na fase de reuso, propõe como solução ao problema novo, a mesma solução do caso recuperado; (3) um usuário do sistema aplica a solução recomendada e, na fase de revisão, avalia que a solução resolveu o problema novo; (4) por se tratar de um caso idêntico a um caso já armazenado na base de casos, o sistema CBR *S* não retém o caso novo.

Considere, agora, que o sistema CBR *S* recebeu como entrada um problema novo em que uma impressora do modelo “RM800” não imprime em cores, estando as luzes de estado do papel, nível de tinta preta e nível de tinta colorida, respectivamente, no estado “Apagada”, “Apagada” e “Acesa”. O sistema CBR *S*: (1) na fase de recuperação, recupera o primeiro caso; (2) na fase de reuso, propõe uma adaptação da solução do problema antigo ao problema novo, recomendando a troca do cartucho de tinta colorida; (3) um usuário do sistema aplica a solução recomendada e, na fase de revisão, avalia que a solução resolveu o problema novo; (4) um especialista no domínio de suporte ao diagnóstico de falhas e reparo em impressoras jato de tinta decide reter o caso novo (problema novo, solução nova) na base de casos.

BC = {
< (“RM800”, “Não imprime em preto”, “Apagada”, “Acesa”, “Apagada”),
“O diagnóstico é de que cartucho de tinta preta está vazio. O problema pode ser resolvido substituindo-se o cartucho de tinta preta.” >,

< (“RM600”, “Impressora não liga”, “Apagada”, “Apagada”, “Apagada”),
“O diagnóstico é de que a fonte de alimentação está danificada. O problema pode ser resolvido substituindo-se a fonte de alimentação da impressora.” >
}

Figura 2 – Exemplo de uma Base de Casos de um Sistema CBR para Suporte ao Diagnóstico e Reparo de Falhas em Impressoras Jato de Tinta (adaptado de Von Wangenheim, Von Wangenheim & Rateke (2013))

3. Trabalho Relacionado

Este trabalho tem como foco os mecanismos de busca empregados na fase de recuperação de sistemas CBR. Diversas soluções têm sido propostas para isto: uso de índices, tal como em sistemas gerenciadores de banco de dados, para auxiliar (não resolver) o processo de busca (Von Wangenheim, Von Wangenheim & Rateke, 2013), uso de estruturas de dados especiais para orientar a busca (e.g., Ball-Tree (Liu, Moore & Gray, 2003), kdTree (Fukunaga & Narendra, 1975) e VP-Tree (Yianilos, 1993)), uso de algoritmos que são modificações (variantes) do algoritmo *k*-NN (Cover & Hart, 1967).

Mitchell (1997) situa o algoritmo *k*-NN (do Inglês, *k-nearest neighbor*), como sendo um exemplar da classe dos algoritmos baseados em instâncias. Algoritmos desta classe não operam sobre um modelo genérico (e.g., árvores de decisão) para orientar o processo de busca de um caso na base de casos. Dado um problema novo, digamos x, o *k*-NN itera sobre toda a base de casos calculando uma função de similaridade entre o problema x e os problemas subjacentes aos casos para, ao fim da iteração, entregar como saída $k \geq 1$ casos mais similares a x. O mecanismo de busca do *k*-NN é simples, mas impõe alta complexidade de tempo, especialmente em base de casos com grande número de casos e/ou problemas descritos por muitas características, o que torna o uso do *k*-NN infactível nestas condições. Variantes do algoritmo *k*-NN têm sido propostas para acelerar o mecanismo de busca do *k*-NN original e, com isto, torná-lo factível para o emprego em sistemas CBR.

As abordagens para acelerar o *k*-NN incluem métodos aproximados (e.g., Wang, Pan & Li (2019); Andoni & Indyk (2008)), em que “abre-se mão” da busca exata dos k casos mais similares a um problema x para diminuir o tempo de busca. Incluem também, métodos exatos que, em uma fase de pré-processamento, fazem uso do preenchimento de estruturas de dados para orientar a busca (Liu, Moore & Gray, 2003). Há também outros algoritmos que usam alguma propriedade (e.g., desigualdade triangular, distribuição estatística dos casos) da função de similaridade utilizada (Xueyi, 2011) para diminuir o tempo de busca.

4. O Estudo Proposto

Sistemas CBR não possuem um projeto padrão. Estes sistemas são projetados tendo em vista as particularidades do domínio em que se aplicam e, entre outros fatores, podem variar em tamanho da base de casos, duração do ciclo do sistema (segundos, horas,

meses etc.), risco da solução proposta, nível de automatização e relação entre o número de consultas realizadas e o número de retenção de casos novos. Estes fatores influenciam o projeto do mecanismo de busca da fase de recuperação de sistemas CBR, foco deste trabalho.

Este trabalho propõe um estudo teórico e experimental sobre algoritmos para busca em sistemas CBR. Especificamente, o trabalho propõe a investigação das condições de aplicação a sistemas CBR de algoritmos que são variantes que objetivam acelerar o algoritmo *k*-NN. No momento em que este artigo está sendo escrito, uma revisão da literatura ainda está em curso. Do trabalho de revisão da literatura já realizado, destacam-se os seguintes recentes algoritmos: kMkNN (Xueyi, 2011), KMC-OTI-FS (Pan *et al*., 2020) e Cell-LevelPQ (Wang, Pan & Li, 2019).

Todos os três algoritmos demandam uma fase de pré-processamento dos casos contidos na base de casos. O algoritmo kMkNN (do Inglês *k-Means for k-NN*) emprega o algoritmo de agrupamento *k*-Means (MacQueen, 1967) para criar $\sqrt{n}$ grupos de casos na fase de pré-processamento, onde *n* é o número de casos da base de casos. Na fase de busca o algoritmo utiliza o conceito de desigualdade triangular (Lima, 2013) para evitar cálculos de função de similaridade entre casos e, com isto, acelerar o *k*-NN tradicional. O algoritmo KMC-OTI-FS estende o algoritmo kMkNN via outra estratégia de emprego de desigualdade triangular que, segundo seus autores, reduz ainda mais o número de cálculos da função de similaridade no processo de busca, às custas de um aumento de uso de memória. O algoritmo Cell-LevelPQ utiliza uma técnica denominada de quantização de produto (Wang, Pan & Li, 2019) para calcular *k* casos aproximadamente similares a um problema dado como entrada. O algoritmo não garante a busca exata de casos similares, mas aposta em uma solução que é um compromisso entre a eficiência e a acurácia da busca.

O estudo proposto objetiva uma investigação teórica, baseada na Teoria da Complexidade de Algoritmos (Cormen, Leiserson & Rivest, 2009), e também experimental, baseada em uma coleção envolvendo conjuntos de dados sintéticos e conjuntos de dados usados em aplicações reais, sobre a eficiência, o consumo de memória, a acurácia e as condições de aplicação, ao projeto de sistemas CBR, de algumas variantes que objetivam acelerar o algoritmo *k*-NN. Estão sendo pesquisados no momento os conjuntos de dados, os índices de eficiência, acurácia e aplicabilidade a serem empregados nos experimentos.

5. Considerações Finais

Os estudos realizados até o momento, sugerem algoritmos que aceleram o *k*-NN cobrando custos computacionais extras em uma fase de pré-processamento. Em sistemas CBR isto significa um aceleramento da fase de recuperação, mas também significa um aumento da complexidade de tempo da fase de retenção de um caso novo. Felizmente, isto é aceitável para a maioria dos projetos de sistemas CBR, os quais possuem um número de buscas na fase de recuperação absolutamente maior do que o número de vezes em que um caso novo é retido.

O trabalho atual precisa amadurecer em relação à revisão da literatura, aos fatores que governam o projeto de sistemas CBR, alguns deles identificados na Seção 4, aos conjuntos de dados a serem utilizados nos experimentos e aos índices de eficiência de tempo, consumo de memória e acurácia.

References

Aamodt, A., Plaza. E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1), 39–59.

Andoni, A. & Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51, 117–122.

Bichindaritz, I., Marling, C., and Montani, S. (2017). Recent Themes in Case-Based Reasoning and Knowledge Discovery. *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference.* Marco Island, Florida. USA. 499–502

Cormen, T. H., Leiserson, C. E. & Rivest, R. L. (2009). *Introduction to Algoritms*. 3rd. ed. MIT Press.

Cover, T. M. & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*. 13 (1), 21–27.

Fukunaga K. & Narendra, P. M. (1975). A branch and bound algorithm for computing. *IEEE Trans. Comput.*, 100 (7), 750–753.

Goel, A. K, and Diaz-Agudo, B. (2017). What's Hot in Case-Based Reasoning. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (AAAI'17). San Francisco, California, USA. 5067–5069.

Kolodner. J. (1993). *Case-based reasoning*. Morgan Kaufmann.

Lima, E. L. (2013). *Espaços métricos*. 5ª ed. IMPA.

Liu T., Moore, A. W. & Gray, A, (2003). Efficient exact k-NN and nonparametric c classification in high dimensions, In *Proceedings of Neural Information Processing Systems*. Vancouver and Whistler, British Columbia, Canada.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, USA. 281–297.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Montani, S. & Jain, L. C. (2010). *Successful Case-Based Reasoning Applications*. Springer.

Pan, Y., Pan, Z., Wang, Y. & Wang, W. (2020). A new fast search algorithm for exact k-nearest neighbors based on optimal triangle-inequality-based check strategy. *Knowledge-Based Systems*, 189.

Von Wangenheim, C. G., Von Wangenheim, A. and Rateke, T. (2013). *Raciocínio Baseado em Casos com Software Livre e Aplicações Móveis*. 2ª. ed. Bookess.

Wang, Y., Pan, Z. & Li, R. (2019). A New Cell-Level Search Based Non-Exhaustive Approximate Nearest Neighbor (ANN) Search Algorithm in the Framework of Product Quantization. *IEEE Access*, 7, 37059–37070.

Xueyi, W. (2011). A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In *Proceedings Int. Joint Conf. Neural Netw*. 1293–1299.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM symposium on Discrete algorithm*. 311–321.