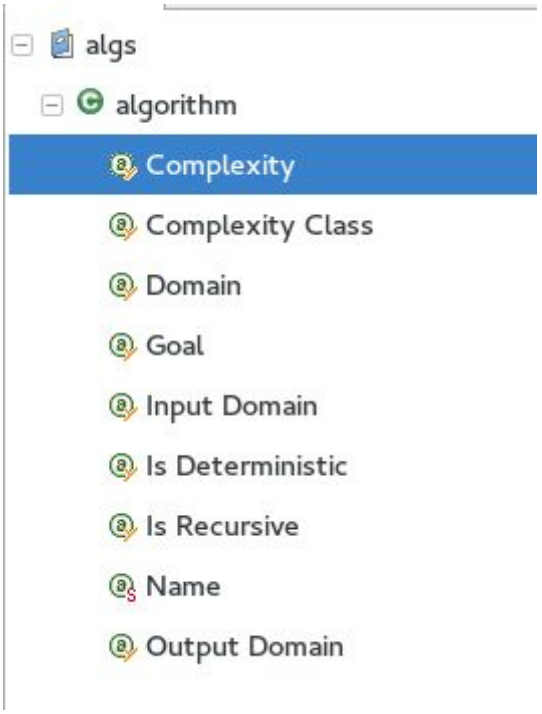
Ghabriel Nunes (14100829)
Marleson Graf (14100859)

Trabalho 4 – Raciocínio Baseado em Casos

Nossa base de dados contempla uma coleção de algoritmos de diferentes áreas da Ciência da Computação. Para cada algoritmo são especificados o nome, a complexidade, a classe de complexidade, o domínio do algoritmo, o objetivo, o domínio dos dados de entrada, o domínio dos dados de saída, se é determinístico e se é recursivo.



A seguir, são mostrados alguns exemplos de tuplas da base de dados. As colunas estão na mesma ordem exposta acima.

Dijkstra's Algorithm	O(|E| + |V|.log |V|)	Graph Theory	Minimization	no	yes	P	graph	path
A*	O(b^d)	Graph Theory	Minimization	no	yes	P	graph	path

AKS	O(log^(21/2) n)	Number Theory	Primality test	no	yes	P	positive integer	bool
Lucas primality test	O(log n)	Number Theory	Primality test	no	no	P	positive integer	bool
Binary search	O(log n)	Searching	Search	no	yes	P	list	list element

Para todas as colunas foi utilizado o tipo *Symbol* no myCBR. As únicas colunas na qual faria sentido utilizar outro tipo seriam *Is Deterministic* e *Is Recursive*, que poderiam ser do tipo *boolean*. Entretanto, devido ao myCBR não suportar a especificação dos valores de colunas *boolean* na sua ferramenta de busca, optamos por deixar o tipo *symbol*, com a função de similaridade *default*, isto é, a função identidade.

Para a coluna de complexidade, cada par de complexidades foi analisado e uma similaridade definida manualmente. Assim, complexidades como O(n) e O(n log n) podem ser definidas como bastante similares (foi utilizado o valor 0.9), visto que na prática os algoritmos com essas complexidades possuem um tempo de execução que escala de maneira semelhante.

Outra questão trata de complexidades para as quais a única diferença está na notação. Por exemplo, há algoritmos no dataset cuja complexidade é O(n^2) (por exemplo, Bubble Sort) e outros com complexidade O(|V|^2), como o algoritmo de Prim. É a mesma complexidade, mas, por serem de domínios diferentes, notações diferentes são utilizadas. Essa diferença é levada em consideração em tal similaridade, o que nos levou a atribuir peso 0.9 a tal combinação. Utilizando raciocínio semelhante, outros pares possíveis foram analisados e seus pesos escolhidos.

Para as classes de complexidade, foi considerado um peso de 0.5 entre NP-completo e NP-hard, visto que essas classes se intersectam, apresentando

problemas de complexidade similar. O peso de 0.2 para P e NP se baseia na esperança de que a conjectura P = NP possa ser verdade.

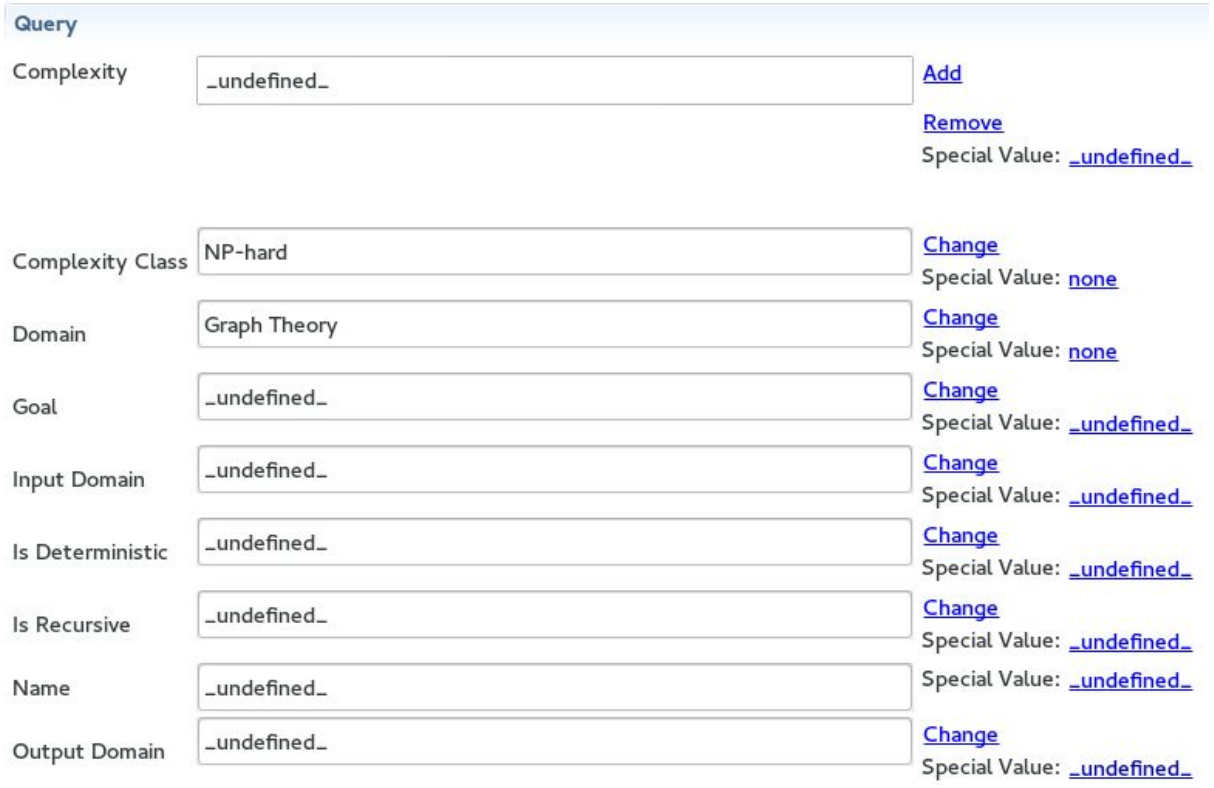
Para a coluna do domínio, foi atribuído peso 0.4 à relação graph theory-searching devido a essas áreas frequentemente estarem relacionadas e haver algoritmos que as intersectam. O peso 0.05 entre graph theory e sorting foi escolhido por um motivo semelhante. Apesar da interseção não ser tão comum quanto no caso anterior, algoritmos como “topological sort” se encaixam em ambas. Foi atribuído 0.2 a searching-sorting por frequentemente algoritmos desses domínios colaborarem para atingir um objetivo comum. Por exemplo, para o algoritmo “binary search” - um algoritmo de busca - é necessário que os dados de entrada estejam ordenados.

A coluna do objetivo relaciona objetivos que estão frequentemente associados. Por exemplo, alguém que pesquise pelo algoritmo de soma também poderá estar interessado no algoritmo de subtração, o qual tem peso 0.7, seguido do algoritmo de multiplicação, com peso 0.6.

Para os campos “input domain” e “output domain”, a seguinte lógica foi utilizada. O valor “list” deve possuir similaridade não-nula com valores que representam mais de um dado, como “2 positive integers”, “list of edges”, “path” (uma lista ordenada de vértices) e assim por diante. Além disso, pares como “positive integer” e “2 positive integers” também devem possuir certa similaridade (nesse caso, foi atribuído o peso 0.7).

A definição dos pesos dos atributos baseou-se, majoritariamente, no quanto cada atributo restringe o conjunto de algoritmos que cumprem com a característica especificada. O nome, por especificar precisamente qual algoritmo se está buscando, tem o maior peso (100), o qual não é superado pelo somatório de todos os outros pesos. O segundo atributo mais restritivo é o objetivo, com peso 20. Em seguida, temos o domínio, que representa um conjunto maior de algoritmos com objetivos diferentes e, portanto, tem peso 10. O domínio das entradas e saídas tem certa relação com o objetivo do algoritmo, recebendo peso 2. A complexidade relaciona-se com a estrutura do algoritmo, levando peso 2 também. A classe de complexidade, se é determinístico e se é recursivo são atributos mais genéricos, recebendo peso 1. Além disso, os valores foram estabelecidos de forma que as

somas de atributos menos significativos não ultrapasse o valor dos atributos mais significativos (nome, objetivo e domínio).



	algorithm3	algorithm9	algorithm10	algorithm4
Complexity	O(n^2.2^n);	O(1.657^n);	O(3^(n/3));	O(|E|.log |V|);
Complexity Class	NP-hard	NP-complete	NP-complete	P
Domain	Graph Theory	Graph Theory	Graph Theory	Graph Theory
Goal	Minimization	Coverage	Decomposition	Minimization
Input Domain	graph	graph	graph	graph
Is Deterministic	yes	yes	yes	yes
Is Recursive	no	no	no	no
Name	Travelling salesman problem	Hamiltonian Path	Clique Problem	Kruskal's Algorithm
Output Domain	path	path	list of subgraphs	list of edges

Nas imagens acima, vemos um exemplo de busca. O melhor resultado atende simultaneamente a todos os requisitos, como desejado. Já o segundo e terceiro não atendem à classe de complexidade desejada, mas possuem a mais próxima a ela: NP-complete. Por fim, o quarto resultado contém a classe de complexidade mais distante da desejada: P.