

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR)
DAINF - DEPARTAMENTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ALEX BENDER

**CONSULTAS POR SIMILARIDADE COM DIVERSIDADE
OTIMIZADAS PELA TÉCNICA OMNI EM SISTEMAS
GERENCIADORES DE BANCO DE DADOS RELACIONAIS**

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2021

ALEX BENDER

**CONSULTAS POR SIMILARIDADE COM DIVERSIDADE
OTIMIZADAS PELA TÉCNICA OMNI EM SISTEMAS
GERENCIADORES DE BANCO DE DADOS RELACIONAIS :**

**Similarity queries with diversity optimized by OMNI technology in
relational database management system**

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR), como requisito parcial para a obtenção do título de Bacharel.

Orientador: Dr. Ives Renê Venturini Pola
Universidade Tecnológica Federal do Paraná
(UTFPR)

PATO BRANCO
2021



Esta licença permite remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es) e que licenciem as novas criações sob termos idênticos. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

ALEX BENDER

**CONSULTAS POR SIMILARIDADE COM DIVERSIDADE OTIMIZADAS PELA
TÉCNICA OMNI EM SISTEMAS GERENCIADORES DE BANCO DE DADOS
RELACIONAIS**

Trabalho de Conclusão de Curso de Graduação
como requisito para obtenção do título de Bacharel
em Engenharia de Computação da Universidade
Tecnológica do Paraná (UTFPR).

Data de aprovação: 26 de Novembro de 2021

Ives Rene Venturini Pola
Doutor
Universidade Tecnológica Federal do Paraná (UTFPR)

Marco Antonio Barbosa
Doutor
Universidade Tecnológica Federal do Paraná (UTFPR)

Jefferson Tales de Oliveira
Doutor
Universidade Tecnológica Federal do Paraná (UTFPR)

Viviane Dal Molin de Souza
Doutor
Universidade Tecnológica Federal do Paraná (UTFPR)

**PATO BRANCO
2021**

AGRADECIMENTOS

Começo agradecendo aquele que é o autor de toda sabedoria, Nosso Senhor Jesus Cristo; sua santa mãe a Virgem Maria e todos os santos.

Agradeço ao meu professor orientador Dr. Ives Renê Venturini Pola pelos seus sábios conselhos e ajuda nas dificuldades.

Agradeço também a todos aqueles que me ajudaram a manter-me firme neste projeto. Ao Padre Sérgio Vieira da Rocha que me concitou a ter firmeza na decisão de desenvolver este trabalho, e toda a esta comunidade religiosa que me é muito cara; unida a força dos meus pais que me ajudaram a não desistir.

Agradeço a esta faculdade em todos os seus esforços em ensinar-me e aos professores nesta caminhada de formação.

Bem-aventurados os que choram, porque serão consolados (Mt 5,4).

RESUMO

BENDER, Alex. Consultas por similaridade com diversidade otimizadas pela técnica OMNI em Sistemas Gerenciadores de Banco de Dados Relacionais. 2021. 61 Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná (UTFPR). Pato Branco, 2021.

Os bancos de dados atuais recebem, além de dados convencionais como textos e números, novos dados conhecidos como dados complexos: imagens, áudios, vídeos, entre outros. Um dos problemas atuais nesta área, é encontrar uma resposta relevante em bases com dados redundantes e também em pesquisas ambíguas. A relevância desta resposta pode ser melhorada utilizando-se das *consultas com diversidade*. Estas consultas buscam uma diversificação de resultados, tornando as respostas mais satisfatórias. Em compensação, geram um outro problema: o tempo de processamento que é elevado pela complexidade dos algoritmos. Para acelerar estas consultas, é possível utilizar estruturas próprias dos bancos de dados ou ainda outros métodos, como a técnica OMNI. As consultas com diversidade utilizadas neste trabalho são tês: MMR, GMC e GNE. Um dos fatores que determinam o tempo de execução é a consulta por abrangência que traz os dados que estão salvos no banco de dados. Acelerando a consulta por abrangência com a utilização da técnica OMNI houve uma aceleração de mais de três mil por cento. Contudo, as consultas por abrangência perdem sua performance com o aumento do raio de consulta.

Palavras-chave: Banco de Dados relacional. Consultas com diversidade. Técnica OMNI. Heurística. PostgreSQL.

ABSTRACT

BENDER, Alex. . 2021. 61 f. Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná (UTFPR). Pato Branco, 2021.

The actual data bases has beyond convencional data like text and numbers, new datas known as complex data: images, audios, videos, etc. A current problem in this area are finding a relevant answer in a redundant data bases or ambiguous queries. The relevance inthis answer can be enchanced using result diversification queries. This queries seek for diversify the results, making them more satisfactory. On the other hand, it generates another problem: the processing time is increased by the complexity of the algorithm. To speed up this queries, it is possible to use databases structures or another methods, like OMNI technology. The result diversification queries used in this work are three: MMR, GMC and GNE. There is one key step to streamline this queries: the range query. The range query pulls data from the databases and is within the result diversification queries. Using the OMNI technology with the range queries, result diversification queries achieved three thousand percent speed, but by increasing the range of the range query, the performance of result diversification queries with OMNI technology fell apart.

Keywords: Relational Data Bases. Query result diversifation. OMNI Tech. Heuristic. Post-greSQL.

LISTA DE FIGURAS

Figura 1 – Áreas de qualificação pelo mbOr	9
Figura 2 – Exemplos de imagens da base de dados DOGS X CATS	19
Figura 3 – Resultado explain analyze	21
Figura 4 – Comparação do número de podas por número de focos utilizados na consulta	22
Figura 5 – Exemplo de retorno range Query	24
Figura 6 – Cor para os testes	36
Figura 7 – Resultados da <i>consulta com diversidade</i> MMR, segundo a variação do coeficiente de diversidade	37
Figura 8 – Comparativo da consulta MMR	38
Figura 9 – Resultados da consulta GMC	39
Figura 10 – Comparativo da consulta GMC	40
Figura 11 – Resultados da consulta GNE	41

LISTA DE TABELAS

Tabela 1 – Bases de dados	18
Tabela 2 – Porcentagem de tempo de execução do Rq nas consultas com diversidade	42
Tabela 3 – Tempo de execução da RQ	42
Tabela 4 – Tempo de execução das consultas com diversidade aplicado a técnica OMNI	43
Tabela 5 – Tempo da consulta por abrangência por variação do raio	43

LISTA DE ALGORITMOS

Algoritmo 1 – O algoritmo HF para determinar a base de focos na base de dados S .	10
Algoritmo 2 – <i>Range Query</i> para <i>OmniB-Forest</i>	12
Algoritmo 3 – <i>Maximal Marginal Relevance</i> - MMR	14
Algoritmo 4 – <i>Greedy Marginal Contribution</i> - GMC	15
Algoritmo 5 – GRASP	16
Algoritmo 6 – GNE-Construction	16
Algoritmo 7 – <i>GNE-LocalSearch</i>	17

SUMÁRIO

1 – INTRODUÇÃO	1
1.1 Considerações Iniciais	1
1.2 A técnica OMNI	2
1.3 Consultas com diversidade	2
1.4 Problemática	2
1.5 Objetivos	2
1.5.1 Objetivo geral	2
1.5.2 Objetivos específicos	3
1.6 Organização do trabalho	3
2 – Conceitos e definições para métricas, consultas e estruturas de indexação	4
2.1 Espaços métricos	4
2.2 Funções de distância	5
2.2.1 Distância Minkowski	5
2.3 Extratores de características	5
2.4 Consultas por similaridade	5
2.5 Estruturas de indexação	6
3 – Técnica OMNI	7
3.1 A técnica	7
3.2 O funcionamento da técnica	8
3.3 A cardinalidade da base de focos	9
3.4 Escolhendo os focos com o algoritmo <i>Hull of Foci (HF)</i>	10
3.5 Indexação das coordenadas OMNI	10
3.5.1 Indexação de coordenadas OMNI: A <i>OmniB-Forest</i>	11
3.5.1.1 <i>Range Query</i> e <i>OmniB-Forest</i>	11
4 – Consultas com diversidade	13
4.1 <i>Maximal Marginal Relevance</i> - MMR	13
4.2 <i>Greedy Marginal Contribution</i> - GMC	15
4.3 <i>Greedy Randomized with Neighborhood Expansion</i> - GNE	15
4.3.1 Fase de construção:	16
4.3.2 Fase de busca local:	17
5 – Materiais e Metodologia	18
5.1 Base de dados	18

5.1.1	Base DOGS X CATS	19
5.1.2	Base SIFT	20
5.2	Métricas	20
5.2.1	EXPLAIN ANALYZE	20
5.2.1.1	Tempo de Planejamento (<i>Planning Time</i>)	21
5.2.1.2	Tempo de Execução (<i>Execution Time</i>)	21
5.3	Escolha do número de focos da base de dados	21
5.4	<i>Consultas com diversidade</i> usando SQL	22
5.4.1	Cálculo de distância	22
5.4.2	O MMR	24
5.4.3	O MMC	26
5.4.4	O GMC	28
5.4.5	O GNE	29
5.4.5.1	Construtor	30
5.4.5.2	A busca local	32
5.4.5.3	O avaliador de conjunto	33
5.5	A técnica OMNI	34
5.5.1	A criação da base de focos	34
5.5.2	O cálculo da distância e a técnica OMNI	34
6	ANÁLISE E DISCUSSÃO DOS RESULTADOS	36
6.1	Resultados qualitativos	36
6.1.1	MMR	36
6.1.2	GMC	38
6.1.3	GNE	40
6.2	Resultados quantitativos	41
7	CONCLUSÃO	44
7.1	TRABALHOS FUTUROS	44
	Referências	46
	Apêndices	49
	APÊNDICE A –Configuração do banco de dados PostgreSQL	50
	APÊNDICE B –Criação da base de focos	51
	APÊNDICE C –Cálculo da distância entre dois elementos	53

APÊNDICE D–Avaliador de conjunto	54
APÊNDICE E–GNE	55
APÊNDICE F–Comparação do MMR com e sem o uso da técnica OMNI .	59
APÊNDICE G–Comparação do GMC com e sem o uso da técnica OMNI .	60
APÊNDICE H–Consulta por abrangência com uso da técnica OMNI	61

1 INTRODUÇÃO

Neste capítulo é apresentada uma introdução geral do assunto para o desenvolvimento deste trabalho, e também, quais são os objetivos a serem alcançados e o problema a ser resolvido.

1.1 Considerações Iniciais

Bancos de dados armazenam informações relevantes para empresas e serviços, tendo um grande impulso com a revolução da internet na déc. de 90 (SILBERSCHATZ, 2011). Em sua grande maioria, as bases de dados adotam o modelo relacional (DB-ENGINES, 2021) que busca representar os dados em tabelas que se relacionam entre si (SILBERSCHATZ, 2011). Estes dados comumente eram conhecidos por serem apenas letras, números, caracteres especiais. Há algum tempo, os bancos de dados armazenam novos formatos de dados como imagens, vídeos, documentos, impressões digitais entre outros, e estes novos dados são conhecidos como dados complexos (SANTOS, 2012).

Dados complexos não utilizam pesquisas com comparações usuais: $>$, $<$, $>=$, $<=$, $!$, $=$, mas sim, consultas por similaridade. As consultas por similaridade se baseiam em encontrar o que é parecido mas não igual (BARIONI et al., 2009).

Para determinar a similaridade entre estes dados complexos, é possível utilizar algumas funções, entre as mais famosas estão as *consultas em abrangência* (*Range Query* (R_q)) e aos *k-vizinhos mais próximos* (*kNN Query*). Essas funções se utilizam de um espaço métrico para calcular as distâncias entre os dados e determinar quão próximos um ao outro são. Uma forma de acelerar estas consultas, é a técnica *Omni*, que pode ser de 10 a 15 vezes mais rápida, com resultados equivalentes (TRAINA et al., 2007), e comparando ao método sequencial é possível obter um ganho de velocidade de 400% (MATSUI, 2018).

Mas nem sempre as pesquisas com similaridade podem satisfazer o usuário. Um exemplo que bem ilustra esse problema é: se em um site de buscas, um cliente pesquisa por CARRO AZUL, ele não espera que todos os carros retornados sejam do mesmo modelo e da mesma marca, mas em bases de dados deste tipo de aplicação, será comum encontrar uma grande quantidade de carros do mesmo modelo e marca, ou seja, uma redundância. Pesquisas puramente por similaridade, resultariam, em uma grande quantidade de carros azuis muito parecidos entre si. Entretanto, recentemente, uma abordagem tem buscado resolver este problema trazendo diversidade no retorno da pesquisa, aumentando a satisfação e a experiência do usuário (ZHENG et al., 2016). Esta abordagem tem um nome bem sugestivo: *consultas com diversidade*. As *consultas com diversidade* trariam carros azuis, mas com diferentes modelos e marcas.

Abaixo são apresentados: primeiramente uma pequena explicação do que é a técnica

Omni e após quais são os algoritmos de *consultas com diversidade*.

1.2 A técnica OMNI

A técnica OMNI tem como objetivo final acelerar o processo de busca no banco de dados. Alguns passos são necessários para que isso seja alcançado: são elegidos dados considerados, importantes, denominados focos. Esses focos auxiliam no cálculo das funções de distância ajudando a criar índices responsáveis por esse processo de aceleração (TRAINA et al., 2007).

1.3 Consultas com diversidade

São diversas funções de *consultas com diversidade* que podem ser implementadas como Relevância Marginal Máxima (*Maximal Marginal Relevance MMR*) (CARBONELL; GOLDS-TEIN, 1998); Dispersão *Max-Sum* (*Max-Sum Dispersion MSD*) (GOLLAPUDI; SHARMA, 2009); Método de Troca (*Swap Method*) (YU; LAKSHMANAN; AMER-YAHIA, 2009); Método BTroca (*BSwap Method*) (VIEIRA et al., 2011); *Motley* (JAIN; SARDA; HARITSA, 2004); Método Baseado em Agrupamento (*Clustering-Based Method*) (LEUKEN et al., 2009); Contribuição Marginal Gulosa (*Greedy Marginal Contribution GMC*) e Algoritmo guloso aleatório com expansão de vizinhos (*Greedy Randomized with Neighborhood Expansion - GNE*) (VIEIRA et al., 2011); *Result with influence diversification - BRID* (SANTOS et al., 2013); entre outras.

Todas essas técnicas retornam *consultas com diversidades* mas com conjuntos de respostas diferentes, pois utilizam diferentes funções para os cálculos. Apenas algumas das técnicas acima serão utilizadas.

1.4 Problemática

As *consultas com diversidade* buscam trazer novos conjuntos de resposta, mas tem um tempo de execução elevado, pois, normalmente, precisam buscar duas vezes na memória os dados (uma vez na busca por similaridade, outra por diversidade) e também tem uma complexidade de algoritmo alta. Este trabalho busca unir a técnica OMNI e as *consultas com diversidade*, diminuindo assim, o tempo de retorno do conjunto resposta.

1.5 Objetivos

Abaixo são apresentados os objetivos deste trabalho:

1.5.1 Objetivo geral

Este trabalho tem como objetivo unir a técnica de indexação de banco de dados OMNI e *consultas com diversidade* acelerando o processo de busca, criando diversidade nas

consultas em bases com grande redundância de dados e por fim aumentando a satisfação nas pesquisas.

1.5.2 Objetivos específicos

- Implementar os algoritmos de *consultas com diversidade*;
- Argumentar a qualidade da diversidade na resposta dos algoritmos implementados através da análise dos valores das funções;
- Unir a técnica OMNI e as *consultas com diversidade*;
- Comparar os algoritmos de *consultas com diversidade* sem a indexação e com a indexação da técnica OMNI as seguintes métricas:
 - Precisão e acurácia, sem e com a indexação;
 - Tempos de execução, sem e com a indexação.
- Documentar os algoritmos implementados para usos futuros.

1.6 Organização do trabalho

Este trabalho está dividido nos seguintes capítulos: No capítulo 2 são apresentados conceitos iniciais, no capítulo 3 é apresentada a técnica OMNI, no capítulo 4 é apresentado as funções e algoritmos das *consultas com diversidade*.

2 Conceitos e definições para métricas, consultas e estruturas de indexação

Toda consulta em banco de dados retorna um conjunto de respostas condizente com os parâmetros que são informados. Em bases com dados complexos é necessário uma forma de medir o grau de similaridade com que n elementos se relacionam e determinar, também, uma medida para diversidade entre os elementos.

Isto pode ser feito a partir de espaços métricos.

2.1 Espaços métricos

Uma forma de medida são os espaços métricos. É necessário partir da matemática para definir aquilo que é apresentado nesse capítulo, como domínios e funções. Um domínio, são os valores válidos para a entrada em uma função. E uma função é uma operação matemática que a partir do domínio, gera um contradomínio que são os valores gerados por uma função. Se temos uma função $y = x$, todos os valores em \mathbb{R} são válidos para o domínio.

A partir do que foi mencionado acima, partindo de um domínio \mathbb{S} , uma métrica possível é calculada com uma função $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$, que relaciona dois pontos (s_1, s_2) a um número real $d(s_1, s_2)$, chamado distância de s_1 a s_2 e que atende as propriedades que são próprias do espaço métrico (LIMA, 1977), é possível observar, que s_1, s_2 estão dentro de \mathbb{S} . O espaço métrico é o par $M = \langle \mathbb{S}, d \rangle$, onde \mathbb{S} é o domínio dos dados e d a distância entre os pontos e, necessita satisfazer a seguinte definição:

Definição 1. *Seja S um conjunto não-vazio de elementos e $d(s_1, s_2)$ uma métrica definida sobre $\mathbb{S} \times \mathbb{S}$. O par $\langle \mathbb{S}, d \rangle$ é chamado de espaço métrico desde que d satisfaça as seguintes propriedades:*

1. $d(s_1, s_2) = 0$ (identidade);
2. Se $s_1 \neq s_2$ então $d(s_1, s_2) > 0$ (não-negatividade);
3. $d(s_1, s_2) = d(s_2, s_1)$ (simetria);
4. $d(s_1, s_3) \leq d(s_1, s_2) + d(s_2, s_3)$ (desigualdade triangular)

onde $s_1, s_2, s_3 \in \mathbb{S}$.

Uma base de dados S está num espaço métrico se $S \in \mathbb{S}$.

A propriedade da desigualdade triangular teve um grande enfoque na técnica OMNI que será apresentada no capítulo 3, pois métodos de acessos métricos dividem, em geral, o banco em pequenos conjuntos ou nós, organizando-os em estruturas de árvore. Deste pequeno conjunto é escolhido um elemento para representá-lo, denominado elemento central s_q , e deste elemento são feitas as medidas para os outros dados. Usando esta propriedade é possível diminuir o número de cálculos de medidas, já que, se o cálculo para um item mais longínquo já foi feito, itens mais próximos estarão a uma distância menor.

2.2 Funções de distância

O espaço métrico apresentado acima está bem definido mas ainda é necessário definir a função de distância e como ela será usada. A função de distância é a responsável por determinar quão próximos elementos estão e quão diverso será o conjunto de resposta. A função de distância que será utilizada neste trabalho é a distância Minkowski.

2.2.1 Distância Minkowski

A distância Minkowski é a mais indicada para similaridade pois independe da origem do espaço dos conjuntos de dados (JAIN; DUBES, 1988).

Definição 2. *Sejam $s_1 = \{s_{11}, s_{12}, \dots, s_{1n}\}$ e $s_2 = \{s_{21}, s_{22}, \dots, s_{2n}\}$ dois vetores de dimensionalidade n pertencentes ao conjunto de elementos \mathbb{S} , a distância Minkowski entre esses dois elementos é dada por:*

$$d(s_1, s_2) = \sqrt[p]{\sum_{i=1}^n |s_{1i} - s_{2i}|^p} \quad (1)$$

Para cada valor de p a distância Minkowski tem um nome específico, para $p = 1$ distância City-Block(L_1), com $p = 2$ distância euclidiana(L_2) e com $p \rightarrow \infty$ distância Chebyshev(L_∞).

2.3 Extratores de características

Nas seções 2.1 e 2.2 são apresentados métodos para o calculo das distâncias entre elementos. Esses elementos são obtidos através de extratores de características, advindos dos dados complexos, e neste trabalho, em especial, imagens, nas quais podem ser obtidas características baseadas no aspecto como cor ou textura, de contorno ou regiões dimensionais. Com essas características é possível calcular a distância entre os elementos.

2.4 Consultas por similaridade

Com as características e as funções de distância é possível determinar quais são as formas das consultas por similaridade. Uma das formas mais comuns de consulta com similaridade é a *Range query* apresentadas pelo trabalho de Traina et al. (2007).

Definição 3. (*Range query - R_q*) *Dado um elemento $s_q \in \mathbb{S}$ e uma distância máxima r_q , a resposta é um subconjunto de S tal qual $R_q(r_q, s_q) = \{s_i | s_i \in S : d(s_i, s_q) \leq r_q\}$.*

O exemplo usado por Traina et al. (2007) é de que se uma base formada por palavras em inglês e um função de distância L_n , uma consulta *range query* seria $R_q('America', 3)$, ou

seja, palavras que tenham 3 unidades de distância da palavra América utilizando a função Minkowski de distância com grau n .

2.5 Estruturas de indexação

Indexar dados para encontrá-los de forma rápida e precisa sempre foi utilizado, como por exemplo, numa lista de chamada de uma escola, cada aluno sempre recebe um número que o identifica, e assim é possível facilmente encontrar os dados do aluno e qual ano/turma ele se encontra.

A mesma ideia pode ser usada em computação. Esta ideia começou com Knuth (1998) dando uma introdução a este tema, indexando arquivos. Alguns anos após isso, uma estrutura de indexação ganhou força a *B-Tree* (árvore binária), sendo utilizada de forma abrangente em banco de dados.

Criada por Bayer e McCreight (1972) consiste na generalização de uma árvore binária, em que cada nó pode guardar mais de dois filhos. A sua eficácia em acelerar as pesquisas advém da busca ser feita de forma binária, diminuindo o tempo de procura em comparação a métodos de busca em vetores de forma sequencial. A complexidade algorítmica cai de $O(n)$ para $O(\log n)$, e pode ser usada em especial em banco de dados.

3 Técnica OMNI

A partir dos conhecimentos construídos no Capítulo 2 é possível definir o que é a técnica OMNI e como utilizá-la. Toda a base desta técnica, foi construída nos trabalhos de (FILHO et al., 2001) e (TRAINA et al., 2007), e teve como uma de suas implementações por (MATSUI, 2018).

Esta técnica é constituída em gerar uma estrutura de indexação que permita acelerar as pesquisas de tal forma que, seja diminuído o número de acessos à memória (primária e secundária), e por consequência, o tempo para retorno de valores das consultas. Isto é possível através da diminuição do número dos cálculos de distância entre os elementos das consultas, utilizando a desigualdade triangular apresentado na 1 pela sua quarta propriedade.

3.1 A técnica

Usando métodos de acesso métrico, pequenos conjuntos são formados. Se neste conjunto apenas um elemento é escolhido como central, o método de acesso é dinâmico, se não, estático.

Definição 4. (Base de focos OMNI \mathcal{F}) Dado um espaço métrico $M = \langle \mathbb{S}, d \rangle$, uma base de OMNI-focos é um conjunto $\mathcal{F} = \{f_1, f_2, \dots, f_n | f_g, f_j \in \mathbb{S}, f_g \neq f_j, \forall g \neq j\}$

Nesta definição temos que n é o número de focos e cada f_g é um dado escolhido para foco de \mathbb{S} .

A técnica OMNI busca não mais dividir o banco de dados em pequenos conjuntos, mas a partir do conjunto completo de todos os dados, escolher alguns que sejam mais representativos. Estes dados escolhidos são chamados Omni *focus* e o conjunto de todos os focos, base de focos OMNI (*OMNI foci-base*).

Definição 5. (Coordenadas OMNI) Dado um objeto $s_i \in \mathbb{S}$ aonde $S \subseteq \mathbb{S}$ e a base de focos OMNI \mathcal{F} , as coordenadas OMNI $C(s_i)$ de um objeto s_i é um conjunto de distâncias do objeto até cada um dos focos $C(s_i) = \{ \langle f_g, d(f_g, s_i) \rangle, g = 1, 2, \dots, h \}$

Cada vez que um dado é inserido no banco uma nova coordenada é calculada e salva em um arquivo de acesso randômico (RAF - *Random Access File*), que contém além da distância do novo dado aos focos, um identificador para que possa ser encontrado o dado.

A partir das definições acima é possível entender a forma com que esta técnica funciona.

3.2 O funcionamento da técnica

Ao utilizar o tipo de consulta *Range Query* $R_q(s_q, r_q)$, sendo que s_q é o centro da consulta e r_q o raio, é normalmente feito em duas etapas: filtragem e refinamento. Na técnica OMNI não é diferente.

No passo de filtragem é feita uma *Range Query* com centro numa coordenada $C(s_q)$ num raio r_q , retornando um conjunto de candidatos a resposta da pesquisa. A propriedade da desigualdade triangular garante que falsas rejeições não ocorram, mas não impede falsos alarmes, que são corrigidos na segunda etapa, refinando os dados a partir do identificador do dado original no arquivo RAF. Desta forma é calculada a distância final entre os objetos e definido quem será o conjunto resposta. A definição matemática deste funcionamento pode ser encontrado abaixo:

Definição 6. (*Minimum-bounding-Omni-region mbOr*) Para uma base de focos $\mathcal{F} = \{f_1, f_2, \dots, f_n\}$, $\mathcal{F} \subset \mathbb{S}$, a $mbOr_{\mathcal{F}}(s_q, r_q)$ de uma consulta com centro em $s_q \in \mathbb{S}$ e um raio r_q , um subconjunto de objetos de S é:

$$mbOr_{\mathcal{F}}(s_q, r_q) = \bigcap_{g=1}^h I_g \quad (2)$$

Aonde I_g é um subconjunto de S que esta dentro dos limites superiores e inferiores denotados por:

$$I_g = \{s_i | I_g^{inf} \leq df_g(s_i) \leq I_g^{sup}, \forall s_i \in \mathbb{S}\} \quad (3)$$

Aonde $I_g^{inf} = df_g(s_q) - r_q$ se $df_g(s_q) \geq r_q$ ou $I_g^{inf} = 0$ e $I_g^{sup} = df_g(s_q) + r_q$.

A mbOr pode ser exibida de forma simplificada pela Figura 1.

Nesta figura é possível observar como a mbOr pode-se comportar. Na Figura 1a é possível observar o espaço \mathbb{S} sem nenhum tipo de filtragem, apenas com uma simples consulta por abrangência; na Figura 1b, ao utilizar um foco, é criado uma área de filtragem, em que os elementos fora dessa área são podados, e não entram nos cálculos distância da consulta por abrangência; nas outras sub-figuras apresentam o aumento do número de focos, diminuindo a mbOr, aumentando o número de podas.

Da Figura 1 existem alguns problemas, se o número de focos não for o ideal, há um aumento no número de falsos alarmes, que são os elementos dentro da área mbOr mas fora do raio de pesquisa. Assim é preciso ter um número de focos correto segundo a dimensionalidade intrínseca da base de dados.

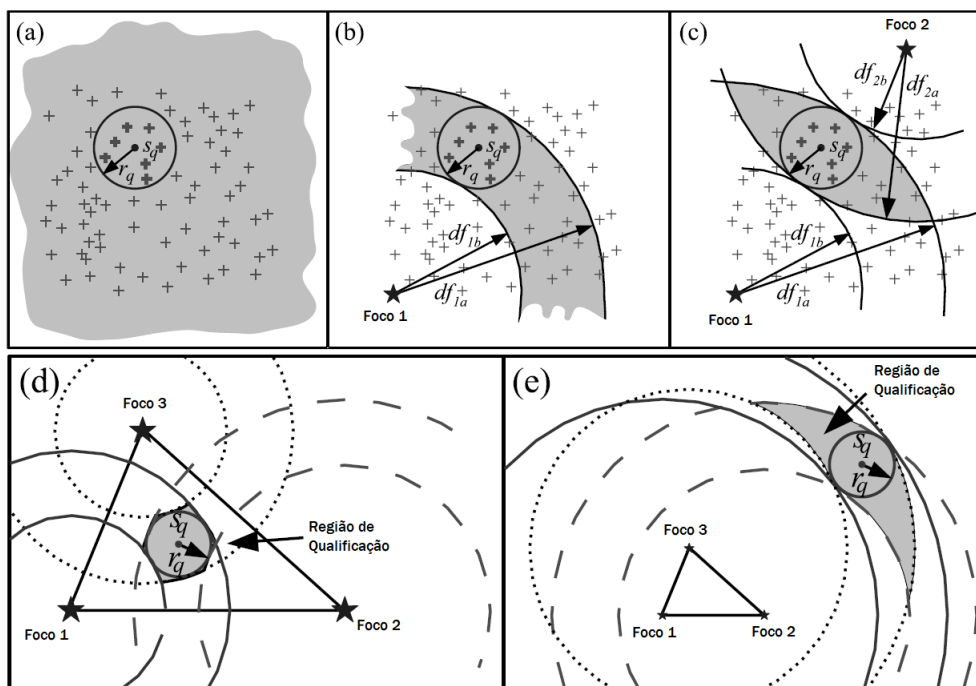


Figura 1 – Áreas de qualificação pelo mbOr

Fonte: Adaptado de (TRAINA et al., 2007)

3.3 A cardinalidade da base de focos

O problema da cardinalidade está fortemente relacionado com a base \mathcal{F} . É necessário escolher de forma otimizada o número de focos, diminuindo o número de cálculos de distância e o espaço utilizado para guardar todas essas informações.

Para uma boa escolha da base de focos é necessário escolher o número certo de focos que bem posicionados diminuam ao máximo o número de falsos positivos.

Ao usar as métricas da família Minkowski (Seção 2.2.1) é possível definir que o número de focos necessários é dado por $[D+1]$, dado que D é a dimensão intrínseca da base de dados acrescido de 1, pois, nem sempre os focos estarão bem posicionados para abranger de forma otimizada todas as consultas. Uma das formas possíveis de medir a dimensão da base é utilizando a métrica denominada *box-counting* (BLOCKAND; BLOH; SCHELLHUBER, 1990) e a prova que o número de focos é $[D+1]$ se encontra no trabalho de Traina et al. (2007).

No trabalho apresentado por Matsui (2018) não foi possível encontrar o número de focos através deste método, mas de forma empírica, através do número de dados que eram filtrados na primeira etapa.

3.4 Escolhendo os focos com o algoritmo *Hull of Foci (HF)*

Para determinar os focos que serão adicionados na base, usa-se o algoritmo a seguir, proposto por Traina et al. (2007):

Algoritmo 1: O algoritmo HF para determinar a base de focos na base de dados S

Entrada: a base de dados S e o número de focos h

Saída: a base de focos \mathcal{F}

1. Escolhe randomicamente um objeto $s_i \in S$.
 2. Encontre o dado f_1 mais distante de s_i . Insira f_1 em \mathcal{F} .
 3. Encontre o dado f_2 mais distante de f_1 . Insira f_2 em \mathcal{F} .
 4. Determine $borda = d(f_1, f_2)$.
 5. ENQUANTO houver focos a serem encontrados, FAÇA:
 6. PARA $s_i \in S, s_i \notin \mathcal{F}$: FAÇA
 $erro(s_i) = \sum_{g \in \mathcal{F}} |borda - d(f_g, s_i)|$
 7. Selecione $s_i \in S$ dado que $s_i \notin \mathcal{F}$ e o erro é o menor
 8. Insere s_i em \mathcal{F}
 9. FIM-ENQUANTO
-

O algoritmo tem como funcionamento alguns passos, sendo o primeiro passo, escolher de forma aleatória um dado da base de dados $s_i \in S$. Partindo de s_i é selecionado um novo dado que é o mais distante deste que será o primeiro foco denominado f_1 inserido a base de focos \mathcal{F} . O mesmo passo anterior se repete e o dado mais longe de f_1 será selecionado e inserido a base de focos \mathcal{F} sendo o foco f_2 . A distância de $d(f_1, f_2)$ é denominada borda.

O próximo passo desse algoritmo só ocorre se a cardinalidade da base de focos mais um, é maior que dois. Se for deve ser executados os seguintes passos: Testar para cada $s_i \in S$ e $s_i \notin \mathcal{F}$, calcule o erro. O dado com o menor erro é adicionado a base de focos \mathcal{F} .

É possível calcular o erro a partir da seguinte equação:

$$erro(s_i) = \sum_k^{k_{foco}} |borda - d(f_k, s_i)| \quad (4)$$

Este algoritmo precisa de apenas $h \times N$ cálculos, sendo h o número de focos da base de focos \mathcal{F} e N o número de dados na base de dados S , e mais $(h - 1) \times N$ cálculos para a segunda parte do algoritmo.

A criação da base de focos acontece anterior as consultas com o uso da técnica OMNI, sem aplicar um aumento de custo ao banco de dados a não ser na memória secundária. Mas não em tempo de execução de consultas.

3.5 Indexação das coordenadas OMNI

Até o momento, é apresentado o funcionamento geral da técnica OMNI sem ser mencionado a estrutura de indexação da base de dados que é utilizada. Nesta seção será

apresentado a indexação de coordenadas OMNI utilizando árvores B.

3.5.1 Indexação de coordenadas OMNI: A OmniB-*Forest*

Espaços métricos não tem noção de ordem em seus dados e, nesta estrutura de indexação, espera-se uma completa ordenação dos dados em seu domínio; portanto, não é possível indexar de forma direta.

Uma solução possível é usar as distâncias já calculadas, trocando o arquivo de coordenadas OMNI por uma nova estrutura de h árvores B, uma para cada foco; aonde todos os dados do banco estarão distribuídos nas folhas da árvore, e em cada folha os dados que estarão contidos são a $df_g(s_i)$ e $IOid(s_i)$. Unindo todas as árvores de um lado e o arquivo com as distâncias e os dados reais de outros tem-se um *OmniB-Forest* (TRAINA et al., 2007).

3.5.1.1 *Range Query* e *OmniB-Forest*

O algoritmo responsável pela *Range Query* se encontra abaixo, e consiste em resumo na intersecção de cada uma das h árvores B; por fim utiliza-se de um vetor para guardar os melhores $IOid(s_i)$ para o conjunto resposta.

O Algoritmo 2, tem como entrada o conjunto de todos os dados da base, as árvores B, o dado que está sendo pesquisado e o raio de abrangência. Com estas entradas é possível encontrar o conjunto de respostas R.

O primeiro passo é determinar a coordenada-OMNI do dado central para os focos da base \mathcal{F} ; o vetor V contará a quantidade de ocorrências de um determinado elemento nas consultas dentro do limiar I_g , sendo com que cada posição do vetor é o $IOid(s_i)$ armazenado na própria árvores B de cada foco. Se ao final da consulta da árvores B, se para cada elemento o número de contagens for igual ao número de focos o elemento está dentro da $mbOr$ e pode ser verificado sua qualidade em relação ao item central. Se a distância do elemento selecionado ao centro for menor que o raio de consulta ele é inserido no conjunto de saída.

Um problema que pode ocorrer é que, se a dimensionalidade intrínseca do banco de dados for muito grande, mas que normalmente não passa de dez (TRAINA et al., 2007) em vez da utilização de um vetor que ocuparia muito espaço na memória primária, pode-se contornar esse problema implementando uma árvore binária ou uma tabela *hash* para armazenar parte dos dados e utilizar um *merge join*, com o restante dos dados que estão na memória secundária.

Algoritmo 2: *Range Query para OmniB-Forest*

Entrada: O conjunto de dados S , o conjunto de todas as B -Trees, o dado central s_q e o raio da consulta r_q

Saída: o conjunto de saída R

1. Calcular as coordenadas Omni
 $C(s_q) = df_g(s_q), 1 \leq g \leq h$ do centro da consulta s_q ;
2. Seja V um vetor de contadores, para cada objeto $s_i \in S$, e todos os elementos de $V = 0$.
3. *PARA CADA* g , *ONDE* $1 \leq g \leq h$, *FAÇA* :
4. Determine o intervalo das chaves válidas:
 $I_g = [df_g(s_q) - r_q, df_g(s_q) + r_q]$.
5. Passe pela B -Tree correspondente ao foco g e procure por todas as coordenadas(chaves) que estão dentro do intervalo I_g , e:
6. Para cada objeto que se encaixa no item anterior, retorne seu $IOid(s_i)$ e incremente o valor em $V[IOid(s_i)]$.
7. *FIM-PARA-CADA*
8. Sequencialmente passe pelo vetor V e:
9. *PARA CADA* s_i *ONDE* $V[i] = h$, *FAÇA*:
10. Retorne o dado original de S , e:
11. *SE* $d(s_i, s_q) \leq r_q$ *ENTÃO*:
12. Insere em R .
13. *FIM-SE*.
14. *FIM-PARA-CADA*

Neste capítulo foi apresentada a técnica OMNI e como implementá-la. No próximo capítulo serão apresentadas as *consultas com diversidade* e como desenvolvê-las.

4 Consultas com diversidade

As *consultas com diversidade*, tem como objetivo buscar um balanço entre similaridade e diversidade dos elementos que compõem a consulta. Numa primeira etapa, itens são filtrados para um subconjunto menor $R \subseteq S$ e deste subconjunto R , um subconjunto R' com dados relevantes a consulta e diversos entre si (VIEIRA et al., 2011).

O problema, de forma matemática se resume a:

Seja $D = \{d_1, d_2, \dots, d_n\}$ um conjunto de n dados, d_q o dado central da consulta e $k \leq n$. Também seja a similaridade entre d_i e d_q um valor inteiro determinado por uma função de distância e a diversidade de dois elementos $d_i, d_j \in D, i \neq j$ por uma função de distância, que pode ser a mesma da similaridade. Ainda seja $0 \leq \lambda \leq 1$ o parâmetro da proporção entre similaridade e diversidade. O conjunto de resposta R é dado pela seguinte equação (SANTOS et al., 2013):

$$R = \operatorname{argmax} \mathcal{G}(d_q, R), \forall R \subseteq D, k = |R| \quad (5)$$

e aonde:

$$\mathcal{G}(d_q, R) = (k - 1)(1 - \lambda) \operatorname{sim}(d_q, R) + 2\lambda \operatorname{div}(R) \quad (6)$$

$$\operatorname{sim}(d_q, R) = \sum_{i=1}^k \delta_{\operatorname{sim}}(d_q, d_i), d_i \in R \quad (7)$$

$$\operatorname{div}(R) = \sum_{i=1}^{k-1} \sum_{j=i+1}^k \delta_{\operatorname{div}}(d_i, d_j), d_i, d_j \in R \quad (8)$$

Em termos de complexidade de algoritmo, tem-se uma combinação $C(n, k)$ resultando em uma complexidade de tempo de $O(n^k)$, além disso é necessário $O(k^2)$ cálculos de distância, totalizando uma complexidade de $O(n^k k^2)$, em casos aonde λ diferente de zero e um, se tornando um problema np-difícil (VIEIRA et al., 2011).

Uma das formas de resolver o problema é diminuindo o número de itens do subconjunto inicial, podendo aumentar os valores da função objetivo e melhorando a sua performance (SANTOS et al., 2013).

Abaixo é apresentado alguns algoritmos responsáveis por implementar as funções de *consultas com diversidade*. Algumas destas consultas utilizam a estratégia gulosa para preencher o conjunto inicial R vazio com dados que maximizem a função objetiva.

4.1 Maximal Marginal Relevance - MMR

É uma das primeiras formas de abordagem nas *consultas com diversidade*. Criada em 1998, buscava balancear a qualidade das respostas entre similaridade e diversidade (CAR-

BONELL; GOLDSTEIN, 1998), utilizando o método guloso. A função que demonstra o comportamento do mesmo está abaixo:

$$mmr(s_i) = (1 - \lambda)\delta_{sim}(s_i, q) + \frac{\lambda}{|R|} \sum_{s_j \in R} \delta_{div}(s_i, s_j) \quad (9)$$

Para a equação acima, s_i é um dado do conjunto S , que será inserido no sub-conjunto R se for relevante; λ é o parâmetro informado pelo usuário para a quantidade de similaridade e diversidade, q é a pesquisa que está sendo feita, $s_j \in R$ são todos os outros elementos contidos no subconjunto R .

O valor de λ influencia diretamente o retorno do dado, se $\lambda = 0$ tem-se uma consulta apenas de similaridade, se $\lambda = 1$ apenas uma consulta com diversidade, se λ está entre zero e um, existe, nesse caso, um problema NP-difícil mas pode ser reduzido ao clique máximo de um grafo (encontra sub-grafos completos com o maior número de vértices possíveis), que é encontrar objetos relacionados, como neste caso pela similaridade com o elemento central (HARTMANIS, 1982).

E o pseudo-algoritmo que implementa essa função é este:

Algoritmo 3: *Maximal Marginal Relevance* - MMR

Entrada: O conjunto de dados S , e o número k de elementos

Saída: o conjunto de saída $R \subseteq S, |R| \leq k$

1. $R \cup \emptyset$.
 2. $s_k \leftarrow$ o dado com o valor máximo da função MMR.
 3. $S \leftarrow S \cap s_k$.
 4. $R \cup s_k$.
 5. ENQUANTO $|R| \leq k$ FAÇA
 6. $s_k \leftarrow$ o dado com o valor máximo da função MMR.
 7. $S \leftarrow S \cap s_k$.
 8. $R \cup s_k$.
 9. FIM-ENQUANTO
-

O código tem como parâmetros o conjunto S que é base de dados, e quantos elementos devem ser retornados. Inicializa-se o código com um conjunto R vazio, utilizada para a saída. O primeiro passo é calcular o mmr e encontrar qual é o maior mmr na base de dados, este dado é salvo em uma elemento temporário s_k , este elementos é retirado do conjunto S e inserido no conjunto R . O próximo passo acontece se o número de elementos a serem retornados for maior que um. O algoritmo repete os mesmos passos até a cardinalidade do conjunto R for igual a k , ou seja, o conjunto de resposta for concluído.

Este código tem como grande vantagem a sua velocidade de execução, mas, em contrapartida, se o primeiro elemento s_k for de uma qualidade ruim, o conjunto resposta trará insatisfação ao usuário (SANTOS, 2012).

Alguns dos motivos para a escolha desse método: é um dos métodos primogênito de consultas com diversidade; é base para muitos outros algoritmos de consulta com diversidade;

é o mais rápido. Tendo como ponto negativo os piores resultados (VIEIRA et al., 2011).

4.2 Greedy Marginal Contribution - GMC

O GMC, método proposto por Vieira et al. (2011) juntamente com o método da Seção 4.3, também aborda o método guloso de selecionar os dados para o subconjunto R, o que difere o MMR para o GMC é a função objetiva que não é mais *mmr* mas *mmc*:

$$mmc(s_i) = (1 - \lambda)\delta_{sim}(s_i, q) + \frac{\lambda}{k-1} \sum_{s_j \in R_{p-1}} \delta_{div}(s_i, s_j) + \frac{\lambda}{k-1} \sum_{l=1}^{l \leq k-p} \delta_{div}^l(s_i, s_j) \quad (10)$$

Esta equação assim como no MMR tem a primeira parcela responsável pela similaridade, enquanto as outras duas parcelas são responsáveis pela diversidade, mudando a forma como o conjunto de resposta R é construído. R_{p-1} é o conjunto R parcial de tamanho $p-1$, sendo que p varia entre $1 \leq p \leq k$ e δ_{div}^l retorna o maior δ_{div} do conjunto $S - R_{p-1} - S_i$, ou seja, todos os dados que não estão no conjunto R e nem o dado que está selecionado.

O algoritmo que implementa esta consulta é dado por:

Algoritmo 4: Greedy Marginal Contribution - GMC

Entrada: O conjunto de dados S, e o número k de elementos

Saída: o conjunto de saída $R \subseteq S, |R| \leq k$

1. $R \leftarrow \emptyset$
 2. PARA $p \leftarrow 1$ ATÉ $p = k$ FAÇA
 3. $s_i \leftarrow \text{argmax}_{s_i \in S}(\text{mmc}(s_i))$
 4. $R_p \leftarrow R_{p-1} \cup s_i$
 5. $S \leftarrow S \setminus s_i$
 6. FIM-PARA
 7. $R \leftarrow R_p$
-

Em um primeiro momento, quando o conjunto R ainda está vazio, é selecionado o elemento baseado em sua similaridade com a consulta e a diversidade com os outros elementos de toda a base, nos próximos dados a serem inseridos o subconjunto R que está se formando também importa. Quanto mais próximo $|R|$ de k , menor a contribuição da diversidade em relação a S e maior em relação a R_{p-1} .

Este algoritmo aborda a solução gulosa, mas mudando a função objetiva, em relação ao MMR, houve uma melhora na qualidade da resposta em duas vezes e um tempo melhor que o GNE (VIEIRA et al., 2011).

4.3 Greedy Randomized with Neighborhood Expansion - GNE

Outro método baseado na abordagem gulosa utiliza o GRASP (*Greedy Randomized Adaptive Search Procedure*) (FEO; RESENDE, 1995). Segundo Vieira et al. (2011) é o primeiro

método com solução randômica para *consultas com diversidade*.

O GRASP funciona como uma busca em duas etapas: construção e pesquisa local, para encontrar soluções aproximadas em problemas de otimização combinatória.

O GNE diferente do GMC seleciona de um grupo de dados bem classificados randomicamente, e não apenas o melhor ranqueado. O algoritmo GRASP está abaixo:

Algoritmo 5: GRASP

Entrada: O conjunto de dados S , e o número k de elementos

Saída: o conjunto de saída $R \subseteq S, |R| \leq k$

1. $R \leftarrow \emptyset$.
 2. PARA $i \leftarrow 0$ ATÉ $i \leq i_{max}$ FAÇA
 3. $R' \leftarrow GNE - Construction()$
 4. $R' \leftarrow GNE - LocalSearch(R')$
 5. SE $\mathcal{G}(q, R') > \mathcal{G}(q, R)$ ENTÃO
 6. $R \leftarrow R'$
 7. FIM-SE
 8. FIM-PARA
-

As funções de construção e busca local estão abaixo:

4.3.1 Fase de construção:

O algoritmo da fase de construção é apresentado abaixo:

Algoritmo 6: GNE-Construction

Saída: o conjunto de saída candidato $R \subseteq S, |R| \leq k$

1. $R \leftarrow \emptyset$
 2. PARA $p \leftarrow 1$ ATÉ $p = k$ FAÇA
 3. $s_{max} \leftarrow \operatorname{argmax}_{s_i \in S}(mmc(s_i))$
 4. $s_{min} \leftarrow \operatorname{argmin}_{s_i \in S}(mmc(s_i))$
 5. $RCL \leftarrow \{s_i \in S | mmc(s_i) \geq s_{max} - \alpha(s_{max} - s_{min})\}$
 6. $s_i \leftarrow \operatorname{random}(RCL)$
 7. $R_p \leftarrow R_{p-1} \cup s_i$
 8. $S \leftarrow S \setminus s_i$
 9. FIM-PARA
 10. $R \leftarrow R_p$.
-

Este algoritmo cria um conjunto resultado possível de tamanho k que possa ser avaliado pela pesquisa local. O ponto chave está no subconjunto RCL, pois, é aquele que seleciona os melhores colocados da função mmc para os parâmetros informados e mais diretamente influenciado pelo parâmetro α . Se $\alpha = 0$ o algoritmo é basicamente o GMC, se $\alpha = 1$ é uma construção aleatória trazendo uma contribuição onde os elementos iniciais podem ser diferentes.

4.3.2 Fase de busca local:

O algoritmo da fase de busca local está abaixo:

Algoritmo 7: *GNE-LocalSearch*

Entrada: o conjunto de entrada candidato R de tamanho k

Saída: o conjunto de saída $R \subseteq \mathbb{S}$, $|R| \leq k$

1. $s_{max} \leftarrow \operatorname{argmax}_{s_i \in S}(\operatorname{mmc}(s_i))$
2. $s_{min} \leftarrow \operatorname{argmin}_{s_i \in S}(\operatorname{mmc}(s_i))$
3. $RCL \leftarrow \{s_i \in S \mid \operatorname{mmc}(s_i) \geq s_{max} - \alpha(s_{max} - s_{min})\}$
4. PARA CADA $s_i \in R$ FAÇA
5. PARA CADA $s_j \in R$, $s_j \neq s_i$ FAÇA
6. PARA $l \leftarrow 1$ ATÉ $l = k - 1$ FAÇA
7. $s_i^l \leftarrow \{s_i' \in S \mid \delta_{div}^l(s_i, s_i')\}$
8. SE $s_i^l \notin R$ ENTÃO
9. $R'' \leftarrow R' - s_j + s_i^l$
10. SE $\mathcal{G}(q, R'') > \mathcal{G}(q, R')$ ENTÃO
11. $R' \leftarrow R''$
12. FIM-SE
13. FIM-SE
14. END FOR
15. SE $\mathcal{G}(q, R') > \mathcal{G}(q, R)$ ENTÃO
16. $R \leftarrow R'$
17. FIM-SE
18. FIM-PARA-CADA
19. FIM-PARA-CADA

Este algoritmo gera um conjunto candidato a resposta final, analisando se o conjunto de entrada tem os melhores elementos de diversidade não selecionados, se isso for verdade, retira-se um elemento anteriormente selecionado e se insere este novo mais bem colocado.

A complexidade do GNE é de $O(kn^2)$ para o algoritmo de construção e $O(k^3)$ para a de busca local, feita i_{max} vezes tornando-o mais lento que o GMC mas em compensação os resultados são melhores.

Neste capítulo foi apresentada algumas técnicas de *consultas com diversidade*.

Com este capítulo e o capítulo 3 é finalizado os capítulos de referencial teórico. O próximo capítulo apresenta o desenvolvimento deste trabalho de conclusão de curso.

5 Materiais e Metodologia

Este capítulo tem por objetivo apresentar os materiais necessários para o desenvolvimento deste trabalho e bem como é desenvolvido, utilizando-se dos mesmos.

As tecnologias para o desenvolvimento deste trabalho são:

- o SGBD PostgreSQL 12.4, por ser o quarto banco de dados relacional mais utilizado segundo a [DB-ENGINES](#), o segundo nesta gratuito, mas é mais robusto que o outro banco de dados citado, em conjunto com o aplicativo pgAdmin 4 versão 4.24 que é a interface gráfica de administração e desenvolvimento de consultas em SQL, para o PostgreSQL padrão;
 - a extensão *cube*, própria da linguagem SQL do PostgreSQL (PLPGSQL) que agrupa dados em cubos de n dimensões para o cálculo de distâncias;
 - para o cálculo de distância euclidiana o operador $\langle - \rangle$, já desenvolvido em PLPGSQL.
- a linguagem de programação PYTHON na versão 3.8, pelos pacotes de processamento de imagens nativas do python, com o espaço de desenvolvimento Spyder 4.1.5;
 - o pacote opencv para o retorno visual das consultas;
 - o pacote psycopg2 para comunicação do PYTHON com o PostgreSQL.

O hardware utilizado tem como especificação:

- Processador AMD Athlon 3000G 3.50 GHz
- Memória RAM Corsair Vengeance 8GB 2400 MHZ

O Sistema operacional utilizado é o Windows 10 Education.

5.1 Base de dados

Neste trabalho foram utilizadas duas bases de dados distintas com dados complexos: uma base com imagens de cães e gatos e outra com características diversas extraídas de imagens. As imagens, como citado no Capítulo 1.1, são dados complexos. Um pequeno resumo das bases de dados serão apresentadas nesta seção. As próximas duas seções apresentarão com detalhes cada uma das bases de dados.

Tabela 1 – Bases de dados

Base de dados	Número de Imagens	Número de Características
Dogs vs. Cats	25000	12
SIFT	30607	128

Fonte: Autoria Própria

5.1.1 Base DOGS X CATS

Esta base de dados contém imagens de cães e gatos, numa divisão de imagens em treino e teste, disponível em <<https://www.kaggle.com/c/dogs-vs-cats>>. O Kaggle, muito conhecido na área de aprendizado de máquina, dispõe dessa base de dados. Com a intenção inicial de ser utilizada para um concurso, que procurava determinar, com aprendizado de máquina, se a imagem era de um cão ou de um gato. O algoritmo com a maior taxa de acerto era o vencedor. Esta base de dados, na área de treino, conta com 25000 imagens, 12500 de cada animal, das quais foram extraídas suas características para inserção no banco de dados.

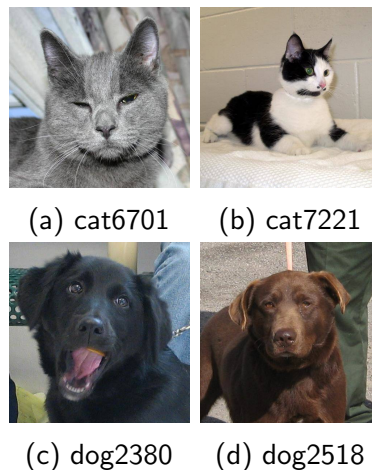


Figura 2 – Exemplos de imagens da base de dados DOGS X CATS

As características extraídas são:

- média e variância dos canais RGB das imagens;
- dissimilaridade, contraste e correlação para a textura, extraídos com a matriz de co-ocorrência de níveis de cinza (*gray level co-occurrence matrices (GLCMs)*), já implementado no python ([HARALICK](#); [SHANMUGAM](#); [DINSTEIN, 1973](#));
- excentricidade e razão da área sobre a área convexa em relação a forma.

Estas características foram retiradas apenas da região relevante da imagem, adquirida por meio de pré-processamentos nas imagens: na primeira etapa a remoção de ruídos com a suavização da imagem por um filtro de Gauss ([SONKA](#); [HLAVAC](#); [BOYLE, 2014](#)); na segunda etapa a segmentação do que é a imagem do animal e o fundo, sendo possível utilizando o limiar de Otsu ([OTSU, 1979](#)). O limiar de Otsu, de forma aplicada, parte de uma imagem em tons de cinza (que em cada um dos seus pixels tem um valor normalizado entre 0 e 1), encontrando um limiar que divide a imagem em duas classes. Se o limiar for bom, conseguirá dividir a imagem como um todo, em fundo e o objeto representado; o limiar de Otsu neste trabalho foi encontrado com uma função do python aplicada a imagem em graus de cinza, denominado *threshold_otsu*.

Toda essa extração de características é possível a partir de um algoritmo desenvolvido em Python, com o auxílio da biblioteca de manipulação de imagens *scikit-image*, que contém

todos estes métodos implementados. Estas características foram salvos em vetores e adicionados ao banco de dados.

5.1.2 Base SIFT

A base de dados SIFT fornecida pela *University of California Irvine*, disponível de forma pública em <<http://archive.ics.uci.edu/ml/datasets/SIFT10M#>> contém uma gama de 11164866 instâncias, cada uma com 128 atributos. Estes atributos, foram retirados de imagens de outra base de dados, a *CALTECH-256*, com uma ferramenta específica de código aberto a *VFeat*, que extrai características por visão computacional (VEDALDI; FULKERSON, 2008), em formato .MAT. Existem tantas instâncias às 30607 imagens da base *CALTECH-256*, pois cada imagem, é dividida em fragmentos de tamanho 41x41 e então suas características são extraídas.

Esta base tem como finalidade os testes de desempenho, e como as imagens não são disponibilizadas junto com a base SIFT, não é possível relacionar os resultados das consultas com as imagens originais, não sendo possível incluir os resultados dos testes qualitativos.

Os vetores de características foram lidos pela ferramenta Python versão 3.8, com o auxílio da biblioteca para leitura de arquivos MAT *h5py*.

5.2 Métricas

Para medir se as *consultas com diversidade* alcançaram as proposições deste trabalho foram utilizados dois métodos.

O primeiro é o teste qualitativo, determinando se as imagens retornadas são as mesmas, utilizando a técnica OMNI ou não, e se o número de imagens retornadas também é igual. O pgAdmin, apresenta o número de elementos retornados pelas consultas e, desta forma, é possível comparar, em primeira instância, se o número de elementos retornados são iguais e, em segundo nível, retornando as imagens de forma visual, num algoritmo em Python.

O segundo é a medida de desempenho. Esta medida de desempenho é retornada pela função *explain-analyze*, uma ferramenta própria do SGBD que retorna o tempo de planejamento e execução da consulta.

5.2.1 EXPLAIN ANALYZE

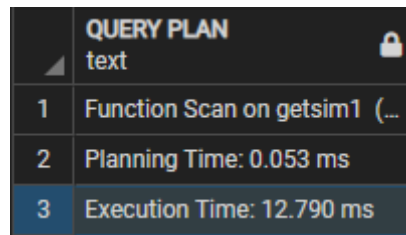
O comando *explain-analyze* é uma ferramenta nativa do SGBD PostgreSQL para medidas de desempenho do banco de dados, retornando em especial, o tempo de planejamento (*Planning Time*) e execução de uma consulta (*Execution Time*).

Este é o código de uma utilização do *explain-analyze*:

```
1 EXPLAIN ANALYZE SELECT * FROM MMR(255,255,255,10,0.7,10);
```

E o resultado:

Figura 3 – Resultado explain analyze



QUERY PLAN	
	text
1	Function Scan on getsim1 (...)
2	Planning Time: 0.053 ms
3	Execution Time: 12.790 ms

Fonte: Aatoria Própria

Outro ponto importante que esta ferramenta apresenta é: como o banco de dados faz o passo a passo para a execução das consultas. É assim possível, determinar configurações para a estrutura do banco que podem acelerar as consultas como um todo, mas em especial, as que se utilizam de métodos de indexação. Estas configurações são direcionamentos ao banco de dados dos recursos disponíveis em quesito de hardware, como por exemplo a quantidade de memória disponível, quantos núcleos existem no processador, qual é a forma de armazenamento. Um pequeno manual se encontra no apêndice [A](#) para estas configurações.

5.2.1.1 Tempo de Planejamento (*Planning Time*)

O tempo de planejamento é o tempo que o SGBD utiliza para determinar qual é a forma menos custosa de executar a consulta. Nos piores casos, a porcentagem de tempo de planejamento em relação ao tempo de execução é menor que 0,5%. Sendo assim, é possível descartar a influência nos testes de desempenho, do tempo de planejamento.

5.2.1.2 Tempo de Execução (*Execution Time*)

O tempo de execução consiste em mostrar o tempo em que uma consulta foi processada e o resultado foi retornado pelo banco de dados, executando a melhor forma planejada pelo banco de dados. Portanto esta é a medida de desempenho para os testes.

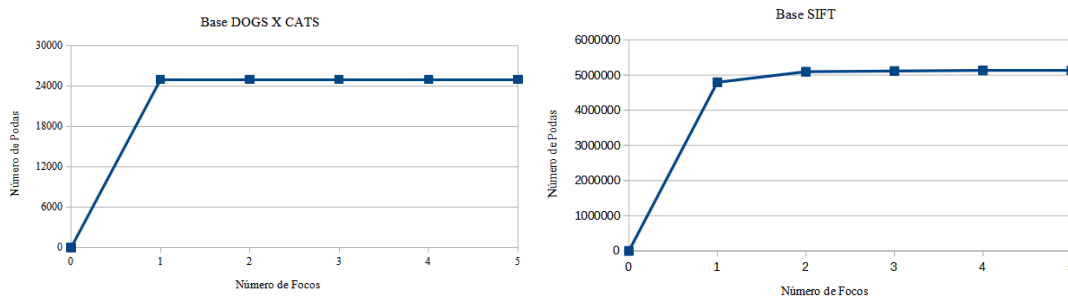
5.3 Escolha do número de focos da base de dados

Assim como apresentado no capítulo [3.3](#), a cardinalidade da base de focos é importante para diminuir ao máximo o número de cálculos desnecessários. Como as bases de dados apresentadas neste trabalho tem grande redundância, ou *clusters*, não é possível utilizar a técnica *box-counting* (MO; HUANG, 2012).

Outra forma de determinar o número de focos é pelo número de podas. O número de podas é o número de elementos que são descartados após a primeira etapa de filtragem da consulta com diversidade.

As seguintes figuras comparam o número de podas por número de focos.

Figura 4 – Comparação do número de podas por número de focos utilizados na consulta



Fonte: Autoria Própria

O número de focos ideal para cada base de dados, colhido de forma empírica, é quando não há aumento nas podas, aumentando o número de focos utilizados. Portanto, é possível perceber que na base DOGS X CATS, um foco é o número ideal já que não há um aumento no número de podas pelo aumento do número de focos; o mesmo acontece na base SIFT, com dois focos.

5.4 Consultas com diversidade usando SQL

As consultas com diversidade foram desenvolvidas em PLPGSQL, a linguagem própria do SGBD Postgres.

A programação na linguagem PLPGSQL segue um esqueleto:

- As palavras chaves CREATE OR REPLACE FUNCTION que garantem a criação da função em desenvolvimento;
- O nome da função;
- Seus parâmetros entre parênteses;
- O retorno com a palavra chave RETURNS e o tipo do retorno;
- Um bloco de declarações de variáveis, denotado pela palavra chave DECLARE;
- E a função em si, depois da palavra chave BEGIN até o END;
- E a discriminação da linguagem utilizada \$\$ LANGUAGE PLPGSQL.

5.4.1 Cálculo de distância

O cálculo da distância entre um elemento e outro é feito pela *range query* (consulta por abrangência). Esta é a primeira função a ser utilizada numa *consulta com diversidade*, criando um conjunto de elementos pré-selecionados. Alguns desses elementos podem ser retornados pela *consulta com diversidade*.

Segue abaixo na linguagem PLPGSQL:

```

1 CREATE OR REPLACE FUNCTION rangeqcl2(r integer, gr integer, b integer,
   radius float8)
2 RETURNS SETOF genq AS

```

```

3 $$
4 DECLARE
5   center_id integer;
6   feature_aux float8[];
7 BEGIN
8   SELECT id_ft INTO center_id FROM
9     (SELECT id_ft, (cube(array[$1, $2, $3]) <-> cube(array[feature[1],
10      feature[2], feature[3]])) ) as dist
11     FROM COLOR
12     ) AS Q
13   ORDER BY dist
14   FETCH FIRST ROW ONLY;
15
16   SELECT feature[1:3] into feature_aux FROM COLOR where id_ft =
17     center_id;
18
19   -----
20 RETURN QUERY
21   SELECT Sim1, red, green, blue, id_ft FROM
22     (SELECT (cube(feature_aux) <-> cube(C1.FEATURE[1:3])) AS Sim1,
23      c1.id_ft, c1.feature[1] as red, c1.feature[2] as green, c1.
24      feature[3] as blue
25     FROM COLOR c1
26     ) AS Q
27   WHERE Sim1 <= $4
28   ORDER BY Sim1;
29 END;
30 $$ LANGUAGE PLPGSQL

```

Esta consulta tem por objetivo encontrar os dados que estão a uma certa distância euclidiana ($\langle - \rangle$), que é menor ou igual a um raio (*radius*), com relação ao centro da consulta (*center_id*) conforme as características, neste caso de cor (*r*, *gr*, *b*).

O resultado, apresentado na figura acima consiste: na distância entre a imagem retornada e o centro da consulta (*dist*); o identificador único desta imagem (*id_ft*); e as características da imagem *RGB*, inseridas no banco (*red*, *green*, *blue*), por meio de uma tabela auxiliar (*genq*). Vale notar, que a primeira imagem retornada sempre será o próprio centro da consulta, conforme a figura abaixo.

Figura 5 – Exemplo de retorno range Query

	sim1 double precision	red double precision	green double precision	blue double precision	id_ft integer
1	0	74.01015375689005	52.83246301131419	31.781549173194083	41880
2	2.0489624004286062	74.40558079169371	51.327777588032376	30.44820519826497	39645
3	2.5987436517495333	72.49250426803243	51.59293640631669	33.488529662825435	96300
4	3.516320723374141	75.2092294233433	56.12531894729168	31.492017204928192	104775
5	3.777223621275514	75.0774939969439	54.37742850905916	35.05893909626719	92412

Fonte: Autoria Própria

A figura acima exemplifica a resposta da consulta *range query*, na primeira coluna (Sim1) é apresentada a distância das figuras a figura central, que é a primeira linha, pois a distância é igual a zero; as próximas três colunas (*red*, *green*, *blue*), são as características da média dos canais *rgb* das imagens; e por fim o identificador da imagem no banco de dados.

Esta consulta pode ser modificada para a utilização na base *SIFT* levando em conta apenas o id da imagem central(*center_id*) e o raio de abrangência.

Esta consulta tem aplicação com a técnica *OMNI*, pois é possível modificar a forma do cálculo da distância, utilizando-se da desigualdade triangular, para filtrar, ou podar, elementos que já estão fora da *mbOr*, diminuindo o número de cálculos.

5.4.2 O MMR

O *MMR* é a primeira consulta com diversidade desenvolvida neste trabalho por ser a mais simples e a primeira a ser desenvolvida.

```

1 CREATE OR REPLACE FUNCTION MMR(red integer, green integer,
2   blue integer, radius float8,
3   lambda float8, nElem int)
4 RETURNS INTEGER[] AS
5 $$
6 DECLARE
7   C1 CURSOR FOR SELECT * FROM rangeqc12($1, $2, $3, $4);
8   R INTEGER[];
9   auxiliar INTEGER;
10  auxiliar2 INTEGER;
11  sim2 double precision;
12  MMR_to_add double precision;
13  MMR_aux double precision;
14  i float8;
15 BEGIN
16   sim2 := 0;
17   MMR_to_add := 9999999;
18
19   WHILE cardinality(R) IS NULL OR cardinality(R) < $6 LOOP
20     FOR R1 IN C1 LOOP
21       IF cardinality(R) = 0 OR cardinality(R) IS NULL THEN
22         R := array_append(R, R1.id_ft);
23       ELSE
24         IF array_position(R, R1.id_ft) IS NULL THEN
25           FOREACH auxiliar2 IN ARRAY R LOOP
26             sim2 := sim2 + (SELECT * FROM getSim(R1.id_ft, auxiliar2));
27           END LOOP;
28           i := (1 / cardinality(R)::float8);
29           MMR_aux := (1 - $6) * R1.sim1 + $6 * i * sim2;
30           IF (MMR_aux < MMR_to_add) THEN
31             MMR_to_add := MMR_aux;
32             auxiliar := R1.id_ft;
33           END IF;
34         END IF;
35       END IF;
36       sim2 := 0;
37     END LOOP;
38     R := array_append(R, auxiliar);
39     MMR_to_add := 9999999;
40   END LOOP;
41   RETURN R;
42 END
43 $$ LANGUAGE PLPGSQL

```

O algoritmo *MMR* apresentado acima foi desenvolvido para ser amigável ao usuário, tendo como parâmetros para a função: as cores aos quais o usuário está pesquisando, denotado pelos três primeiros; logo após é requerido o raio de abrangência da consulta, a taxa de

diversidade e o número de elementos retornados.

Dentro da seção DECLARE existe uma estrutura de dados denominada cursor, que é uma pequena tabela temporária. Nela está o conjunto inicial para classificação segundo o MMR.

Após o BEGIN é dado início a *consulta com diversidade MMR*. O MMR tem como funcionamento, aceitar o primeiro elemento mais próximo do centro como elemento inicial, apresentado na linha 22.

O conjunto R retorna a resposta final. Se R já não está mais vazio, então é necessário classificar os próximos elementos que estão no cursor. As linhas 25-27 são responsáveis pelo cálculo da diversidade do conjunto, a diversidade é a soma das distâncias do elemento que está sendo avaliado a todos os outros elementos do conjunto R . A função *getSim* auxilia este processo, calculando a distância entre dois elementos, que pode ser encontrada no apêndice C. Por fim é calculado o valor MMR do elemento avaliado, na linha 29, utilizando-se: da distância do elemento avaliado até o centro da consulta, valor que já é um dos atributos de cada tupla no cursor; e a diversidade total do conjunto. O elemento com o menor MMR, é aceito como retorno dentro de R .

A *consulta com diversidade MMR* pode ser acelerada pela técnica OMNI de duas formas: na consulta por abrangência e no cálculo da diversidade, utilizando-se da desigualdade triangular, pela BTree OMNI, para encontrar o valor.

5.4.3 O MMC

O MMC apresentado na seção 4.2 é a função de avaliação para o GMC e o GNE as duas outras *consultas com diversidade*.

O algoritmo para o cálculo dessa função é este:

```

1 CREATE OR REPLACE FUNCTION getmmc(nElem integer, lambda FLOAT8, id_ft
   INTEGER, dist_P FLOAT8, R integer[], l integer)
2 RETURNS float8 AS
3 $$
4 DECLARE
5     sim2 float8; -- Valor da segunda parcela do MMC
6     simF float8; -- Valor da terceira parcela do MMC
7     ret float8;  -- Valor do MMC
8
9     cdAux integer; -- Variável auxiliar para percorrer os dados em R
10
11     k1 float8; -- Constante 1 do MMC
12     k2 float8; -- Constante 2 do MMC
13 BEGIN
14
15     sim2 := 0;
16     IF $5 IS NOT NULL THEN

```

```

17     FOR i IN 1..(CARDINALITY(R)-1) LOOP
18         FOR j IN (i+1)..CARDINALITY(R) LOOP
19             sim2 := sim2 + getSim(R[i], R[j]);
20         END LOOP;
21     END LOOP;
22 END IF;
23
24 IF $6 > 0 THEN
25     IF $5 IS NOT NULL THEN
26         SELECT SUM(dist) INTO simF FROM
27             (SELECT c1.id_ft as idc, (cube(c1.feature) <-> cube(c2.feature)
28 ) as dist
29             FROM COLOR C1, COLOR C2
30             WHERE (c2.id_ft != ALL ($5)) AND c1.id_ft = $3
31             ORDER BY dist DESC
32             LIMIT $6
33             ) AS Q
34         GROUP BY idc;
35     ELSE
36         SELECT SUM(dist) INTO simF FROM
37             (SELECT c1.id_ft as idc, (cube(c1.feature) <-> cube(c2.feature)
38 ) as dist
39             FROM COLOR C1, COLOR C2
40             WHERE c1.id_ft = $3
41             ORDER BY dist DESC
42             LIMIT $6
43             ) AS Q
44         GROUP BY idc;
45     END IF;
46 ELSE
47     simF := 0;
48 END IF;
49
50 k1 = 1 - $2;
51 k2 = $2 / ($1 - 1);
52
53 ret := k1 * $4 - (k2 * sim2) - ( k2 * simF );
54 RETURN ret;
55 END
56 $$ LANGUAGE PLPGSQL;

```

A função *MMC* tem como parâmetro: o número de elementos totais a existirem no conjunto R final ($nElem$), a taxa de diversidade λ , o elemento a ser avaliado, a distância do elemento que está sendo avaliado ao centro da consulta, o conjunto com os elementos a serem retornados na consulta (R) e l que é um valor para o cálculo da última parcela do *MMC* que é sempre $nElem - p$, sendo $p = |R| + 1$.

A primeira etapa é o cálculo da diversidade total do conjunto R , como no *MMR*, nas linhas 17-22.

A segunda etapa do algoritmo é o cálculo da terceira parcela do *MMC*, que leva em conta a diversidade dos elementos que ainda não foram adicionados a R , no conjunto S . Portanto, soma-se a distância do elemento avaliado a todos os outros elementos do universo S . Quando l for zero, então essa parcela do *MMC* é zero.

Por fim, retorna-se o valor do *MMC* do elemento avaliado.

5.4.4 O GMC

O GMC tem um funcionamento muito próximo do *MMR* mas, utilizando-se da função de avaliação *MMC*.

```

1 CREATE OR REPLACE FUNCTION getGMCred integer, green integer, blue
   integer
2   , radius float8, lambda float8, nElem integer)
3 RETURNS INTEGER[] AS
4 $$
5 DECLARE
6   C1 CURSOR FOR SELECT * FROM rangeqc12($1, $2, $3, $4);
7   MMC float8;
8   MMC_aux float8;
9   id_selecionado integer;
10  R INTEGER[];
11  p_mmc integer;
12 BEGIN
13   p_mmc = 1;
14   WHILE CARDINALITY(R) IS NULL OR CARDINALITY(R) < $6 LOOP
15     MMC = 999999;
16     FOR R1 IN C1 LOOP
17       IF array_position(R, R1.id_ft) IS NULL THEN
18         MMC_aux = getmmc($6, $5, R1.id_ft, R1.sim1, R, ($6 - p_mmc));
19         IF MMC_aux < MMC THEN
20           MMC = MMC_aux;
21           id_selecionado = R1.id_ft;
22         END IF;
23       END IF;
24     END LOOP;
25     R = array_append(R, id_selecionado);
26     p_mmc = p_mmc + 1;
27   END LOOP;
28
29   RETURN R;
30 END
31 $$ LANGUAGE PLPGSQL

```

Inicia-se o GMC, também, com uma busca inicial por elementos mais próximos do centro, que são selecionados dentro do cursor conforme a linha 5.

A linha 18 é a encarregada por avaliar cada um dos elementos do cursor pela função *MMC* e retornar o valor a uma variável auxiliar. O menor valor de *MMC* é salvo juntamente com o elemento correspondente a este valor. Ao serem avaliados todos os elementos, é salvo no vetor de retorno *R* o menor valor de *MMC*, até que o tamanho de *R* seja o número de elementos esperado.

5.4.5 O GNE

O GNE é uma *consulta com diversidade* com três etapas apresentadas na seção 4.3. Este é uma adaptação do *GRASP*, um dos algoritmos de aprendizado de máquina. O *GRASP* se desenvolve em repetir *i* vezes uma operação de busca a procura de uma melhora num conjunto. A melhora do conjunto é dado pela comparação dos valores retornados por uma função avaliadora, já que a construção deste conjunto é aleatória. O GNE utiliza-se da função avaliadora *MMC*.

O GNE em PLPGSQL apresenta-se da seguinte forma:

```

1 CREATE OR REPLACE FUNCTION GRASP_UserF(red integer, green integer,
2   blue integer, radius float8, lambda float8, nElem int,
3   nIt int, alfa float8)
4 RETURNS integer[] AS
5 $$
6 DECLARE
7   i int;
8
9   R_Line integer[];
10  R_Line2 integer[];
11  R_Line2_value float8;
12
13  R integer[];
14  R_value float8;
15 BEGIN
16  R_Value := -99999;
17  FOR i IN 0..$7 LOOP
18    R_Line := gne_constructor_userF($1, $2, $3, $4, $6, $5, $8);
19    R_Line2 := gne_localssearch_userF($1, $2, $3, $4, $6, $5, R_Line);
20    R_Line2_value := getValueF_Avaliation_userF($1, $2, $3, $4, $6, $5,
21    R_Line2);
22    IF cardinality(R) IS NULL THEN
23      R := R_Line2;
24      R_Value := R_Line2_value;
25    ELSIF R_Line2_value < R_value THEN
26      R := R_Line2;
27      R_Value := R_Line2_value;

```

```

27     END IF;
28     END LOOP;
29
30     return R;
31 END
32 $$ LANGUAGE PLPGSQL;

```

GRASP

O GNE tem como parâmetros:

- As cores RGB;
- O raio da consulta por abrangência (range query);
- A constante λ que apresenta a diversidade;
- O número de elementos a serem retornados;
- O número de iterações do GNE;
- A constante α que diminui ou aumenta o número de elementos a serem selecionados pelo construtor.

Com isto é possível construir um conjunto inicial, com elementos bem classificados pelo *MMC*, apresentado na linha 17. Logo após é possível fazer uma busca local pretendendo encontrar um ou mais elementos que melhorem o conjunto já bem selecionado pelo construtor, apresentado na linha 18. Por fim, é avaliado o conjunto para determinar se esse é o melhor conjunto seguindo a função exata, apresentado no capítulo 4, contido na linha 19.

É apresentado abaixo as funções que compõem o GNE: o construtor, a busca local e o avaliador de conjunto.

5.4.5.1 Construtor

O construtor tem a função de criar um conjunto, com uma boa avaliação. A avaliação dos elementos é dada pelo *MMC* e os melhores elementos avaliados são inseridos na lista *RCL*. O construtor, em PLPGSQL, está apresentado em íntegra no Apêndice E. Nesta seção serão explanados as principais funções que se observam.

A primeira etapa do processo de construção do conjunto de resposta *R* é calcular o *MMC* de cada elemento:

```

1 FOR R1 IN C1 LOOP
2     IF array_position(R, R1.id_ft) IS NULL THEN
3         id_ft := array_append(id_ft, R1.id_ft);
4         mmc := array_append(mmc,
5             getmmc_userF($5, $6, R1.id_ft, R1.sim1, R, ($5 - p_mmc)));
6
7         IF ( mmc[i] > mmc_max) THEN
8             mmc_max := mmc[i];
9         END IF;
10        IF ( mmc[i] < mmc_min) THEN

```

```

11     mmc_min := mmc[i];
12     END IF;
13     i := i + 1;
14     END IF;
15 END LOOP;

```

O cálculo do *MMC* é apresentado na linha 4-5, utilizando-se do mesmo algoritmo apresentado na Seção 5.4.3.

Nesta mesma etapa são salvos o maior e menor valor de *MMC*. Esta informação é importante para a construção da lista *RCL*, como apresentado na Seção 4.3.1.

A segunda etapa é filtrar os elementos segundo a constante α , e os valores de *MMC* máximo e mínimo salvos da etapa anterior, gerando a lista *RCL*.

```

1 k_construct := mmc_max - $7 * (mmc_max - mmc_min);
2
3 i := 1;
4 WHILE i <= cardinality(mmc) LOOP
5     IF (mmc[i] <= k_construct) THEN
6         RCL := array_append(RCL, id_ft[i]);
7     END IF;
8     i := i + 1;
9 END LOOP;

```

O primeiro passo é calcular o valor da constante que filtra os elementos. Esta constante é importante pois determina quais os elementos que serão adicionados a lista, sendo que os elementos só são adicionados a lista se o valor da função *MMC* for menor ou igual ao valor da constante. Se essa afirmação for verdadeira o elemento entra na lista *RCL* e poderá ser escolhido para o retorno no conjunto *R*.

E a última etapa é escolher um número de elementos aleatórios da lista *RCL*, conforme a quantidade informada pelo usuário, e construir assim o conjunto *R*.

```

1 IF CARDINALITY(RCL) IS NOT NULL AND CARDINALITY(RCL) > 1 THEN
2
3     SELECT random()*(CARDINALITY(RCL) - 1) + 1 INTO i;
4     R := array_append(R, RCL[i]);
5
6 ELSIF CARDINALITY(RCL) = 1 THEN
7
8     R := array_append(R, RCL[1]);
9 ELSE
10    RAISE NOTICE '--- ERROR ---';
11 END IF;

```

Se a lista tem mais de um elemento, de forma aleatória, um elemento é adicionado ao conjunto de retorno *R* apresentados na linha 1-4; se houver apenas um elemento na lista ele é adicionado automaticamente; por fim, se não houver elementos na lista *RCL*, há um erro, pois houve um problema com os parâmetros da função.

Com isto o primeiro conjunto R já está construído e pode ser melhorado com a consulta local, o próximo passo da *consulta com diversidade* GNE.

5.4.5.2 A busca local

A busca local tem por objetivo encontrar um novo conjunto R que tenha uma melhor avaliação que o conjunto formado pelo construtor.

Neste trabalho a busca local é feita pela interpolação de um elemento do conjunto R por um elemento do conjunto S' . O conjunto S' é proveniente de uma busca que retorna os n elementos mais diversos de um elemento r_i que está contido em R , pertencentes a S .

Esta busca é dada pelo algoritmo:

```

1 FOR i IN 1..CARDINALITY(R) LOOP
2   FOR j IN 1..CARDINALITY(R) LOOP
3
4     IF i != j THEN
5       R_LINHA := R;
6
7       FOR R1 IN C1(R_Linha[i]) LOOP
8         R_LINHA2 := R_LINHA;
9
10        R_LINHA2[J] := R1.id_ft_2;
11
12        fR_Linha2 := getValueF_Avaliation_userF($1, $2, $3, $4, $5, $6,
R_LINHA2);
13
14        IF ( fR_Linha2 < fR_Linha) THEN
15          R_LINHA := R_LINHA2;
16          fR_Linha := fR_Linha2;
17        END IF;
18      END LOOP;
19
20    END IF;
21  END LOOP;
22
23  IF fR_Linha < fR THEN
24    R := R_Linha;
25    fR := fR_Linha;
26  END IF;
27
28 END LOOP;

```

Um elemento r_i é fixado pois este elemento é a base do conjunto S' . Assim sendo S' será percorrido e cada elemento de R será trocado, como a linha 10 especifica. Esta troca gera um novo conjunto denominado R'' , que é avaliado pela função de valor exato. Se este novo conjunto for melhor que o anterior é salvo.

Ao fim de cada etapa é comparado o novo conjunto melhor avaliado R' com o conjunto inicial R , formado pelo construtor. Se a avaliação de R' for maior que de R , R' passa a ser o conjunto R . O algoritmo se finda até se esgotar todas as possibilidades de mudança em R .

5.4.5.3 O avaliador de conjunto

O avaliador de conjunto segue o cálculo da Equação 5 apresentado no início do capítulo 4. Para determinar o valor da equação, são feitos dois cálculos: a distância de todos os itens ao centro da consulta ($sim(d_q, R)$) e depois a soma total das distâncias entre cada um dos elementos em R ($div(R)$).

O algoritmo em PLPGSQL apresenta-se desta forma:

```

1 CREATE OR REPLACE FUNCTION getValueF_Avaliation_userF(red integer,
2   green integer, blue integer, radius float8,
3   nElem integer, lambda float8, R integer[] )
4 RETURNS float8 AS
5 $$
6 DECLARE
7   ret float8;
8
9   i integer;
10  j integer;
11
12  sim float8;
13  div float8;
14 BEGIN
15   sim := 0;
16   div := 0;
17
18   SELECT SUM(sim1) INTO sim FROM
19     (SELECT * FROM rangeqc12($1,$2,$3,$4)) AS Q
20     WHERE id_ft = ANY($7);
21
22   FOR i IN 1..(CARDINALITY(R)-1) LOOP
23     FOR j IN (i+1)..CARDINALITY(R) LOOP
24       div := div + getSim(R[i], R[j]);
25     END LOOP;
26   END LOOP;
27
28   ret := ($5 - 1)*(1 - $6)*sim - 2 * $6 * div;
29
30   RETURN ret;
31 END
32 $$ LANGUAGE PLPGSQL;

```

Avaliador de conjunto

O cálculo de $sim(d_q, R)$ se apresenta nas linhas 17-19, com um somatório da distância dos elementos em R ao centro da consulta, e o cálculo de $div(R)$ nas linhas 21-25, percorrendo todo o conjunto R e calculando as distâncias entre os elementos.

5.5 A técnica OMNI

Nesta seção serão apresentados as modificações nas consultas para que a técnica *OMNI* possa ser utilizada.

5.5.1 A criação da base de focos

Segundo os dados retirados de forma empírica, já é sabido o número de focos que devem ser utilizados em cada base de dados, conforme a Seção 5.3. Sabendo disso é possível criar tal base, que é de suma importância.

O algoritmo responsável pela criação desta base está apresentado no Apêndice B.

5.5.2 O cálculo da distância e a técnica OMNI

Para desenvolver uma união da consulta de abrangência com a estrutura OMNI foi necessário algumas mudanças pontuais, que não modificam a forma do cálculo da distância mas, restringem o número de cálculos, *i.e.*, as podas, diminuindo o tempo de execução e mantendo a qualidade da resposta.

Aplicado a base SIFT a consulta por abrangência, baseado na consulta desenvolvida em Matsui (2018), é a seguinte:

```

1  dist_fc = ARRAY( SELECT DISTINCT dist_l2 FROM sift_f_base WHERE
    sift_f_base.id_ft = $1);
2
3  RETURN QUERY
4  select * from (
5    SELECT idfT, (cube(feature_aux) <-> cube(S1.FEAT[1:99])) as dist
6    FROM
7    (
8      SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[1] - $2)) AND (SF.DIST_L2 < (dist_fc[1] + $2)) INTERSECT
9      SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[2] - $2)) AND (SF.DIST_L2 < (dist_fc[2] + $2)) INTERSECT
10     SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[3] - $2)) AND (SF.DIST_L2 < (dist_fc[3] + $2)) INTERSECT
11     SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[4] - $2)) AND (SF.DIST_L2 < (dist_fc[4] + $2)) INTERSECT
12     SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[5] - $2)) AND (SF.DIST_L2 < (dist_fc[5] + $2))
13     ) as fP
14     INNER JOIN SIFT S1 ON idfT = S1.ID_FT

```

```
15 ) as D
16 WHERE D.dist <= $2
17 ORDER BY D.dist;
```

A consulta completa apresenta-se no Apêndice H.

As principais mudanças com relação ao método sequencial apresentado na Seção 5.4.1 são: a adição da busca distância do item central aos focos para a aplicação da desigualdade triangular, apresentado na linha 1 e 2; e a modificação da consulta de retorno dos elementos.

Esta última mudança é o coração da técnica OMNI: podar os elementos que estão fora da mbOr, compreendida entre as linhas 8 até 12. Se o elemento estiver dentro da mbOr, este elemento está classificado à filtragem final, apresentada na linha 16, aceitando apenas os elementos que estão dentro do raio de distância do elemento central.

6 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Neste capítulo é tratada as respostas das *consultas com diversidade* e os resultados da aplicação da técnica OMNI.

6.1 Resultados qualitativos

Esta seção tem como objetivo apresentar a resposta visual das *consultas com diversidade*, com a alteração dos principais parâmetros e a resposta das mesmas ao uso ou não da técnica OMNI.

Nesta etapa, foram utilizadas as consultas que tem como parâmetro as cores das imagens a serem pesquisadas, criando um possível cenário real. A cor escolhida foi a seguinte, escolhida de forma aleatória:



Figura 6 – Cor para os testes

Esta cor no sistema *RGB*, utilizada nas *consultas com diversidade*, é: 75, 54, 33. As respostas contam com 5 imagens, e o raio da consulta por abrangência será de 25 unidades.

A variação do parâmetro nas consultas, acontece pela variação do λ , iniciando em 0 até 1, sendo que do 0,1 ao 0,9 o salto é de 0,2. Lembrando que o parâmetro de diversidade λ varia entre 0 e 1. Estes saltos foram escolhidos para apresentarem o comportamento das respostas das *consultas com diversidade*.

6.1.1 MMR

O MMR assim como apresentado nos capítulos anteriores, tem grande influência da primeira imagem selecionada. Isto pode ser constatado nas imagens abaixo:



Figura 7 – Resultados da *consulta com diversidade* MMR, segundo a variação do coeficiente de diversidade

Estas imagens foram extraídas advindas da *consulta com diversidade* MMR sem o uso da técnica *OMNI*. A imagem no canto superior esquerdo, representa, a mais próxima da característica esperada, neste caso a menos dissimilar. Quando λ é igual a 0 temos as imagens no banco mais próximas das características informadas e quando 1 as mais dissimilares entre si.

Como a imagem inicial tem grande influência no conjunto final, num certo grupo de fatores λ as imagens são quase sempre as mesmas.

De $\lambda = 0$ até $\lambda = 0,5$, a similaridade das imagens para com a característica prevalece, após esse limiar, a diversidade é tão ou mais importante que a similaridade.

Utilizando a técnica *OMNI* podemos verificar que a resposta é a mesma, garantindo a qualitatividade da consulta. Apresenta-se aqui três exemplos de comparação, com λ variando entre 0.1 até 0.5. Todas as comparações são encontradas no Apêndice F.

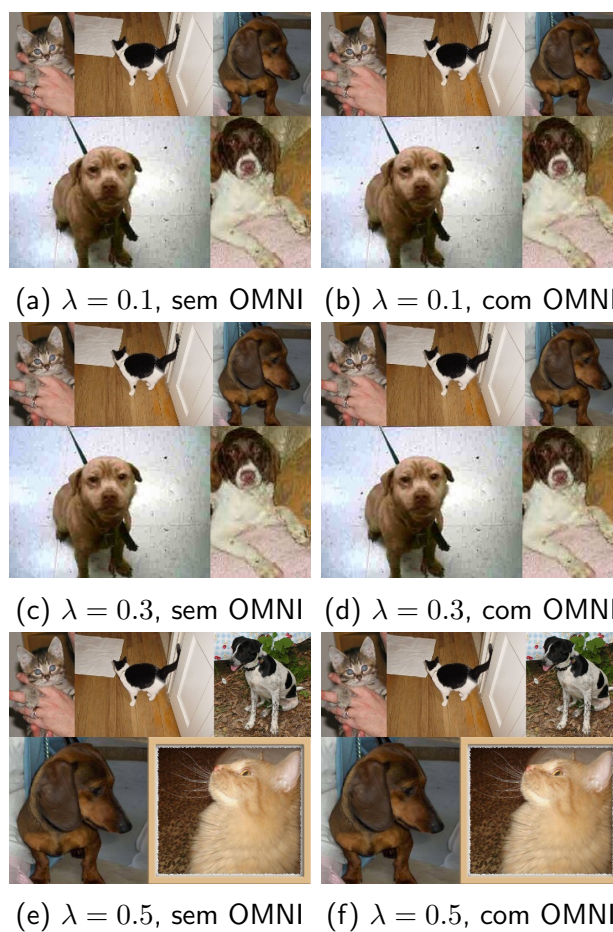


Figura 8 – Comparativo da consulta MMR

6.1.2 GMC

A resposta do *GMC* já apresenta diferenças quanto ao *MMR*, apresentado pela imagem abaixo:

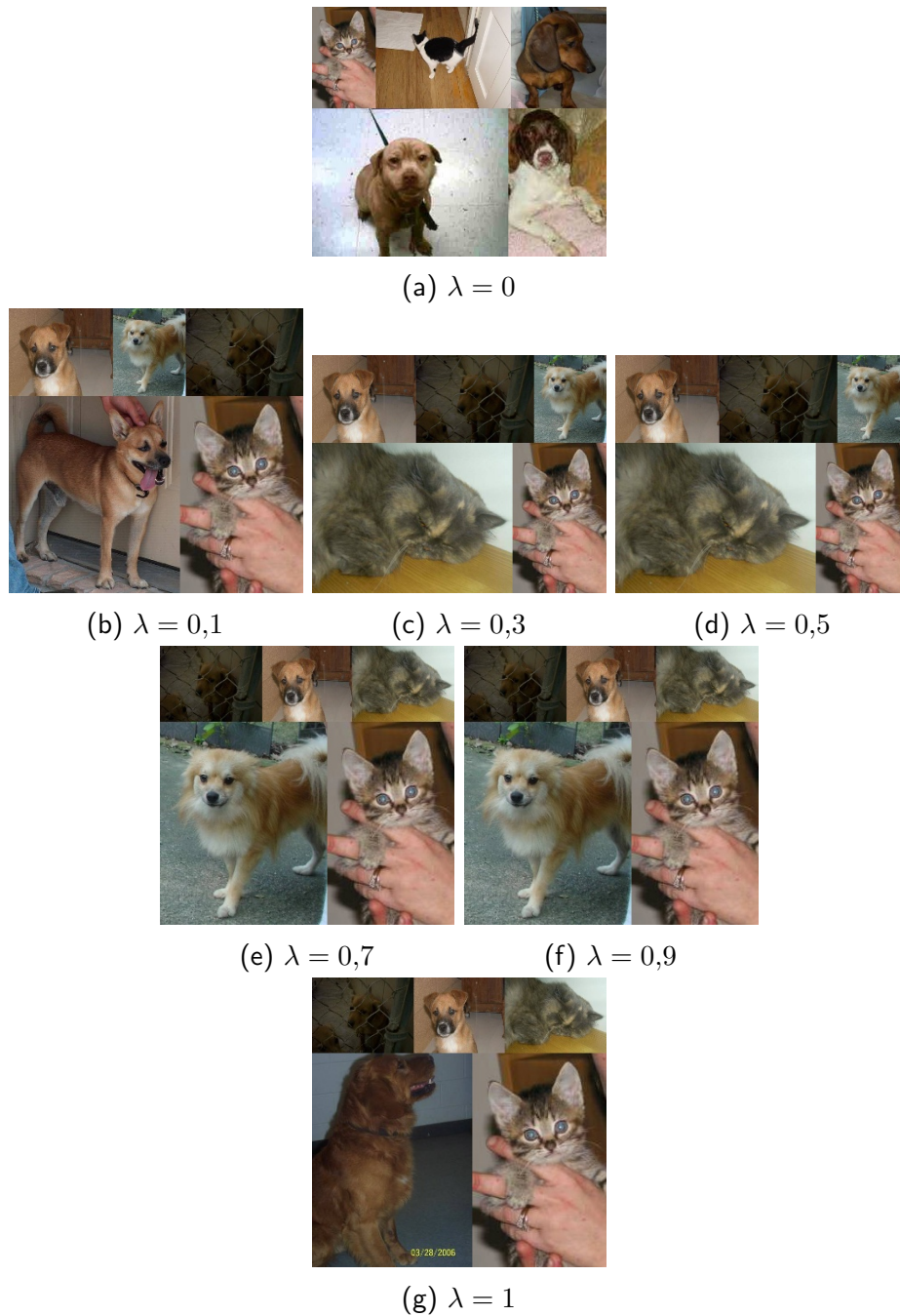


Figura 9 – Resultados da consulta GMC

A primeira imagem é igual ao do *MMR*, pois é levado em conta apenas a dissimilaridade por isto são retornadas apenas as imagens mais próximas do centro.

A partir da Figura 9b, começa-se a levar em conta a diversidade da imagem em relação ao conjunto total, o terceiro fator do *MMC*, por isso a imagem superior esquerda com $\lambda = 0.1$, se torna a última, ou seja, a direita inferior.

As mudanças nos conjuntos de respostas das Figuras 9c-9g, se dá pela mudança na diversidade do conjunto parcial, o segundo fator, já que quanto maior o fator λ maior a influência do fator de diversidade na função *MMC*.

Também houve a garantia de qualitatividade no *GMC*, conforme as comparações a seguir, da mesma forma que o *MMR*. O fator de diversidade λ varia de 0,1 até 0,5. A comparação com a variação de λ entre 0 e 1 está no Apêndice G.

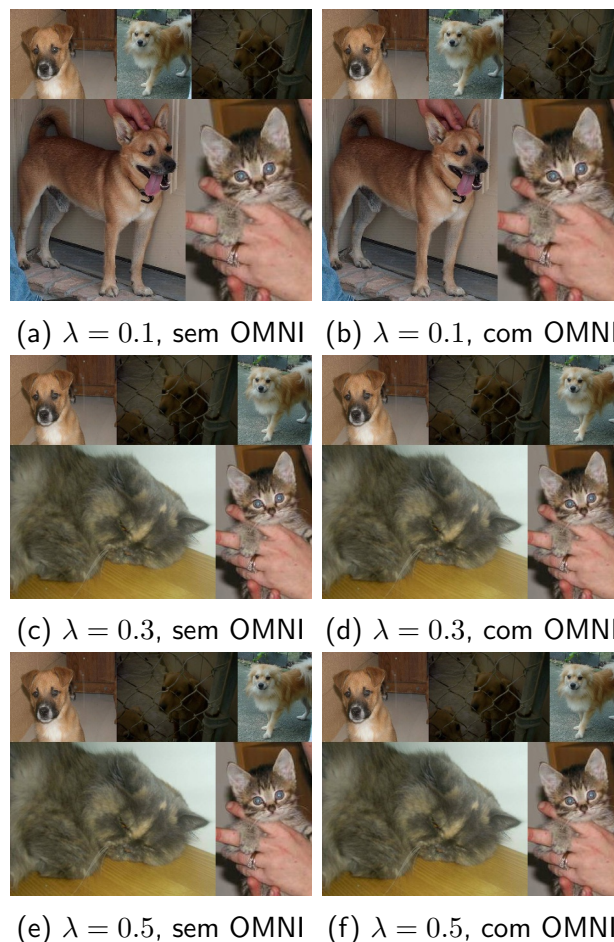


Figura 10 – Comparativo da consulta GMC

6.1.3 GNE

O *GNE* apresenta uma resposta única quanto as outras *consultas com diversidade* por não ser uma consulta determinística. A resposta desta consulta está na Figura 11.

A Figura 11a já não é igual as das outras duas consultas, já que o *GNE* cria uma lista com os melhores colocados na avaliação da função *MMC* e escolhe um elemento aleatoriamente. Este teste foi realizado com o número de iterações igual a 1, portanto não há uma garantia que este seja realmente o melhor resultado possível, mas é possível encontrar as imagens com certas nuances de similaridade com a cor proposta para os testes. O parâmetro α é 0, resultando numa lista *RCL* de todos os elementos, independente do limiar.

Assim sendo, não há como fazer um comparativo no teste qualitativo, entre o *GNE* utilizando a técnica *OMNI* e não a utilizando, já que não há determinismo na resposta, diferente das *consultas com diversidade* apresentadas anteriormente.



Figura 11 – Resultados da consulta GNE

6.2 Resultados quantitativos

Os testes quantitativos são realizados na base SIFT, que contém um maior número de elementos e atributos para cada elemento, vide apresentado no início desse capítulo, apresentando resultados mais próximos da realidade. Para os testes, um elemento de forma aleatória foi escolhido. O elemento aleatório tem como id = 1194630.

O primeiro teste apresenta o tempo de execução de cada um das consultas com diversidade, e a porcentagem de tempo em que a consulta por abrangência é executada, tendo

como raio de abrangência 5 unidades e 5 elementos retornados:

Tabela 2 – Porcentagem de tempo de execução do Rq nas consultas com diversidade

Consulta com diversidade	Tempo de Execução(s)	Tempo de execução do Rq (%)
MMR	157,34	26,00
GMC	286,71	16,64
GNE	292,74	16,24

Fonte: Autoria própria

É possível constatar que a consulta por abrangência é um dos fatores que determinam o tempo de execução das *consultas com diversidade*, sendo ao menos, um quinto do tempo de execução. Ao utilizar a técnica OMNI na consulta por abrangência, apresentada no fim do capítulo quinto, os tempos de execução por variação do número de focos é este:

Tabela 3 – Tempo de execução da RQ

Range Query	Tempo de Execução(s)
Sequencial	41,66
OMNI - 1 Foco	2,264
OMNI - 2 Focos	1,555
OMNI - 3 Focos	2,134
OMNI - 4 Focos	3,066
OMNI - 5 Focos	3,956

Fonte: Autoria própria

A técnica OMNI apresenta uma diminuição do tempo de execução da consulta por abrangência, e por consequência dos tempos de execução das *consultas com diversidade*. Assim os novos tempos de execução são os seguintes:

Tabela 4 – Tempo de execução das consultas com diversidade aplicado a técnica OMNI

Consulta com Diversidade	Tempo de Execução(s)	Desvio Padrão (s)	Aumento de Desempenho(%)
MMR	4,079	0,086	3856,64
GMC	65,912	2,795	434,99
GNE	159,711	21,844	183,29

Fonte: Autoria própria

Ao aumentar o raio de abrangência, há uma perda no desempenho das consultas por abrangência utilizando-se da técnica OMNI.

Tabela 5 – Tempo da consulta por abrangência por variação do raio

Raio	Sequencial(s)	OMNI(s)
5	39,709	1,555
10	40,444	2,043
25	42,442	31,916
50	38,678	189,782
100	39,419	228,981

Estes foram os resultados obtidos na utilização da técnica OMNI com as *consultas com diversidade*. O próximo capítulo apresentará as disposições finais deste trabalho em sua conclusão.

7 CONCLUSÃO

Este trabalho trata sobre as *consultas com diversidade* que são responsáveis por trazer maior satisfatibilidade aos usuários que fazem consultas a base de dados complexas - como por exemplo base de dados compostas de imagens, vídeos, impressões digitais, sequências matemáticas - que possuem grande redundância. Esses resultados são modelados matematicamente por uma função, que desenvolvida computacionalmente é muito custosa. As *consultas com diversidade* apresentadas neste trabalho já apresentam uma aceleração ao cálculo desta função, mas ainda continuam sendo custosas.

Uma forma de acelerar o processo de resposta é a utilização da técnica de indexação OMNI, que se utiliza de estruturas auxiliares para diminuir o número de elementos que podem formar o conjunto de resposta, sem mudança na complexidade do algoritmo. Isto se dá por uma estrutura chamada mbOr, que criam uma região em torno do centro da consulta por meio da desigualdade triangular.

O cálculo da desigualdade triangular é feito por um cálculo de distância. Neste trabalho o cálculo de distância optado foi o euclidiano, desenvolvido como uma consulta de abrangência.

Segundo os dados obtidos a consulta por abrangência é um fator determinante para a criação desta região e portanto, da diminuição do tempo de execução conforme apresentado no Capítulo 6.2.

A utilização da estrutura de indexação OMNI na consulta por abrangência obteve uma aceleração considerável nas *consultas com diversidade* alcançando uma aceleração de mais de 3500% com o MMR, quase 500% com o GMC, e quase 200% no GNE.

Esta técnica de indexação é muito positiva para um alcance de abrangência pequeno, mas aumentando o raio de consulta, a técnica de indexação OMNI perde performance. Isto também foi constatado por Matsui (2018).

Sendo assim é possível concluir que a técnica de indexação OMNI aplicada a consulta por abrangência tem sua aceleração mais efetiva, em raios de abrangência pequenos. O que acarreta a melhor utilização deste método utilizando-se de vários centros de consulta que possam englobar mais resultados em um pequeno raio.

Também foi possível constatar uma limitação na estrutura *cube* do *postgresql*, aceitando apenas 99 dimensões.

7.1 TRABALHOS FUTUROS

As *consultas com diversidade* tem sua utilidade na diversificação de dados para bancos que contêm grande redundância. Outra forma de acelerar tais consultas, além do uso da técnica de indexação OMNI seria a melhora na complexidade de algoritmo, utilizando-se de alguma meta-heurística para uma melhora na performance. Outra melhora que poderia ser procurada

é o cálculo das diversidades dentro de cada *consulta com diversidade* com a indexação OMNI.

De forma secundária, na questão da qualitatividade das consultas, poderia ser encontrada outra forma de retirar as características das imagens tornando o resultado mais sensível, como por exemplo, com o uso de redes neurais e a obtenção das características nas últimas fases.

Referências

- BARIONI, M. C. N. et al. Seamlessly integrating similarity queries in sql. **Software: Practice and Experience**, v. 39, n. 4, p. 355–384, 2009. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.898>>. Citado na página 1.
- BAYER, R.; MCCREIGHT, E. M. Organization and maintenance of large ordered indexes. **Acta Informatica**, Springer-Verlag, v. 1, p. 173–189, September 1972. ISSN 0001-5903. Disponível em: <<https://doi.org/10.1007/BF00288683>>. Citado na página 6.
- BLOCKAND, A.; BLOH, W. von; SCHELLNHUBER, H. J. Efficient box-counting determination of generalized fractal dimensions. **Phys. Rev. A**, American Physical Society, v. 42, p. 1869–1874, Aug 1990. Disponível em: <<https://link.aps.org/doi/10.1103/PhysRevA.42.1869>>. Citado na página 9.
- CARBONELL, J.; GOLDSTEIN, J. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In: **Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval**. New York, NY, USA: Association for Computing Machinery, 1998. (SIGIR '98), p. 335–336. ISBN 1581130155. Disponível em: <<https://doi.org/10.1145/290941.291025>>. Citado 2 vezes nas páginas 2 e 14.
- DB-ENGINES. **DB-Engines Ranking**. 2021. Disponível em: <<https://db-engines.com/en/ranking>>. Citado 2 vezes nas páginas 1 e 18.
- FEO, T.; RESENDE, M. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, p. 109–133, 03 1995. Citado na página 15.
- FILHO, R. F. S. et al. Similarity search without tears: The omni family of all-purpose access methods. In: **Proceedings of the 17th International Conference on Data Engineering**. USA: IEEE Computer Society, 2001. p. 623–630. ISBN 0769510019. Citado na página 7.
- GOLLAPUDI, S.; SHARMA, A. An axiomatic approach for result diversification. In: **Proceedings of the 18th International Conference on World Wide Web**. New York, NY, USA: ACM, 2009. (WWW '09), p. 381–390. ISBN 978-1-60558-487-4. Disponível em: <<http://doi.acm.org/10.1145/1526709.1526761>>. Citado na página 2.
- HARALICK, R. M.; SHANMUGAM, K.; DINSTEN, I. Textural features for image classification. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-3, n. 6, p. 610–621, 1973. Citado na página 19.
- HARTMANIS, J. Computers and intractability: A guide to the theory of np-completeness (michael r. Garey and david s. Johnson). **SIAM Review**, v. 24, n. 1, p. 90–2, 01 1982. Copyright - Copyright] © 1982 Society for Industrial and Applied Mathematics; Última atualização em - 2012-03-05; CODEN - SIREAD. Disponível em: <<https://search.proquest.com/docview/926173619?accountid=26636>>. Citado na página 14.
- JAIN, A.; SARDA, P.; HARITSA, J. R. Providing diversity in k-nearest neighbor query results. In: DAI, H.; SRIKANT, R.; ZHANG, C. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 404–413. ISBN 978-3-540-24775-3. Citado na página 2.

JAIN, A. K.; DUBES, R. C. **Algorithms for Clustering Data**. USA: Prentice-Hall, Inc., 1988. ISBN 013022278X. Citado na página 5.

KNUTH, D. E. **The Art of Computer Programming, Sorting and Searching**. 2 ed.. ed. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998. v. 3. ISBN 0-201-89685-0. Citado na página 6.

LEUKEN, R. H. van et al. Visual diversification of image search results. In: **Proceedings of the 18th International Conference on World Wide Web**. New York, NY, USA: ACM, 2009. (WWW '09), p. 341–350. ISBN 978-1-60558-487-4. Disponível em: <<http://doi.acm.org/10.1145/1526709.1526756>>. Citado na página 2.

LIMA, E. L. **Espaços métricos**. Instituto de Matemática Pura e Aplicada, CNPq, 1977. (Projeto Euclides). Disponível em: <<https://books.google.com.br/books?id=l1mGmQEACAAJ>>. Citado na página 4.

MATSUI, C. J. M. Consultas por similaridade em bases de dados complexos utilizando técnica omni em sgbdr. **Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco**, dec 2018. Disponível em: <<http://repositorio.utfpr.edu.br/jspui/handle/1/14606>>. Citado 5 vezes nas páginas 1, 7, 9, 34 e 44.

MO, D.; HUANG, S. Fractal-based intrinsic dimension estimation and its application in dimensionality reduction. **Knowledge and Data Engineering, IEEE Transactions on**, v. 24, p. 59 – 71, 02 2012. Citado na página 21.

OTSU, N. A threshold selection method from gray-level histograms. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 9, n. 1, p. 62–66, 1979. Citado na página 19.

SANTOS, L. F. D. **Explorando variedade em consultas por similaridade**. Dissertação (Mestrado) — USP - Univerdade de São Paulo, 2012. Citado 2 vezes nas páginas 1 e 14.

SANTOS, L. F. D. et al. Parameter-free and domain-independent similarity search with diversity. In: . New York, NY, USA: Association for Computing Machinery, 2013. (SSDBM). ISBN 9781450319218. Disponível em: <<https://doi.org/10.1145/2484838.2484854>>. Citado 2 vezes nas páginas 2 e 13.

SILBERSCHATZ, A. **DATABASE SYSTEM CONCEPTS**. 6. ed. New York: McGraw-Hill, 2011. ISBN 978-0-07-352332-3. Citado na página 1.

SONKA, M.; HLAVAC, V.; BOYLE, R. **Image Processing, Analysis, and Machine Vision**. Cengage Learning, 2014. ISBN 9781133593607. Disponível em: <<https://books.google.com.br/books?id=DcETCgAAQBAJ>>. Citado na página 19.

TRAINA, C. et al. The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. **The VLDB Journal**, v. 16, n. 4, p. 483–505, October 2007. ISSN 0949-877X. Disponível em: <<https://doi.org/10.1007/s00778-005-0178-0>>. Citado 7 vezes nas páginas 1, 2, 5, 7, 9, 10 e 11.

VEDALDI, A.; FULKERSON, B. **VLFeat: An Open and Portable Library of Computer Vision Algorithms**. 2008. <<http://www.vlfeat.org/>>. Citado na página 20.

VIEIRA, M. R. et al. On query result diversification. **IEEE 27th International Conference on Data Engineering**, p. 1163–1174, apr 2011. Citado 3 vezes nas páginas 2, 13 e 15.

YU, C.; LAKSHMANAN, L.; AMER-YAHIA, S. It takes variety to make a world: Diversification in recommender systems. In: **Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology**. New York, NY, USA: ACM, 2009. (EDBT '09), p. 368–378. ISBN 978-1-60558-422-5. Disponível em: <<http://doi.acm.org/10.1145/1516360.1516404>>. Citado na página 2.

ZHENG, K. et al. A survey of query result diversification. **Knowledge and Information Systems**, p. 1–36, 09 2016. Citado na página 1.

Apêndices

APÊNDICE A – Configuração do banco de dados PostgreSQL

O banco de dados PostgreSQL é robusto na questão de configurações quanto ao hardware. Quanta memória esta disponível, quantas conexões, qual é o número de núcleos do processador e assim por diante. Esta configuração é importantíssima para que se possa recolher os melhores resultados de desempenho. Este manual pode ser útil na configuração da base dados, que se baseia neste manual: [EDB Supercharges PostgreSQL](#).

O primeiro passo é encontrar o arquivo de configuração. Este arquivo se encontra no diretório de instalação do PostgreSQL/data/postgresql.conf.

Algumas das diretrizes de configuração e suas indicações:

- `shared_buffers`: é quantidade de memória alocada para retornar dados. Metade da memória disponível é o valor a ser configurado;
- `work_mem`: é responsável pelos *sorts* e *gather* do banco. Não deve ser um valor muito alto pois cada um dos *gather* em cada base utiliza essa quantidade de memória. O recomendado é $((\text{Memória disponível total} - \text{shared_buffers})) / (16 \times \text{Número de núcleos})$;
- `ranom_page_cost`: este parâmetro é importantíssimo ao utilizar-se de um SSD. O valor deve ser setado em 1.1;
- `effective_cache_size`: a quantidade efetiva de memória que o PostgreSQL pode utilizar. Um valor razoável é metade da memória disponível;
- `max_worker_processes`: o número de núcleos que as consultas podem se utilizar. O número de núcleos físicos e não lógicos
- `max_parallel_workers_per_gather`: o número de núcleos que cada *gather* pode utilizar. Um núcleo por *gather* parece suficiente
- `max_parallel_workers`: o número de núcleos que podem trabalhar em paralelo. A mesma configuração do `max_worker_processes`
- `max_parallel_maintenance_workers`: o número de núcleos para as operações de manutenção do banco. Um núcleo também é suficiente.

Apenas algumas configurações que, ajustadas, podem fazer que uma consulta passe de 50s de execução para 2s.

APÊNDICE B – Criação da base de focos

```

1 CREATE OR REPLACE FUNCTION createColorFocus(num integer)
2 RETURNS VOID AS $$
3 DECLARE
4     fprev_id integer;
5     fnext_id integer;
6     distance float8;
7     n_inserted integer = 0;
8     border float8;
9 BEGIN
10     PERFORM wipe_color_focus_base();
11
12     SELECT id_ft INTO fprev_id FROM color ORDER BY random() LIMIT 1;
13     LOOP
14         SELECT c2.id_ft, cube(c1.feature[1:3]) <-> cube(c2.feature
15         [1:3]) as dist
16             INTO fnext_id, distance
17             FROM color c1, color c2
18             WHERE c1.id_ft = fprev_id AND c2.is_focus = 'False'
19             ORDER BY dist DESC
20             LIMIT 1;
21
22         UPDATE color SET is_focus = 'True' WHERE id_ft = fnext_id;
23
24         n_inserted = n_inserted + 1;
25         num = num - 1;
26
27         RAISE NOTICE 'Num = %, n_inserted = %', num, n_inserted;
28         EXIT WHEN num = 0;
29
30         IF n_inserted = 2 THEN
31             SELECT cube(c1.feature[1:3]) <-> cube(c2.feature[1:3]) INTO
32             border
33             FROM color c1, color c2
34             WHERE c1.id_ft = fprev_id AND c2.id_ft = fnext_id;
35
36             fprev_id = fnext_id;
37
38             LOOP
39                 SELECT c2.id_ft, abs(border - (cube(c1.feature[1:3])
40                 <-> cube(c2.feature[1:3])))
41                 as err INTO fnext_id, distance
42                 FROM color c1, color c2
43                 WHERE c1.id_ft = fprev_id AND c2.is_focus = 'False'

```



```
41         ORDER BY err ASC
42         LIMIT 1;
43
44         UPDATE color SET is_focus = 'True' WHERE id_ft =
fnext_id;
45
46         fprev_id = fnext_id;
47         num = num - 1;
48
49         EXIT WHEN num <= 0;
50     END LOOP;
51
52     EXIT WHEN num = 0;
53
54 END IF;
55
56     fprev_id = fnext_id;
57 END LOOP;
58 PERFORM insertColorDistance();
59 END;
60 $$ LANGUAGE PLPGSQL;
```

APÊNDICE C – Cálculo da distância entre dois elementos

```
1 CREATE OR REPLACE FUNCTION getSim(ID_FT1 INTEGER, ID_FT2 INTEGER)
2 RETURNS double precision AS
3 $$
4 DECLARE
5     ret double precision;
6 BEGIN
7     SELECT Sim2 INTO RET FROM
8         (SELECT (cube(c1.feature[1:3]) <-> cube(c2.feature[1:3])) ) as Sim2
9         FROM COLOR C1, COLOR C2
10        WHERE C1.id_ft = $1 AND C2.id_ft = $2)
11     AS Q;
12
13 RETURN ret;
14 END;
15 $$ LANGUAGE PLPGSQL
```

APÊNDICE D – Avaliador de conjunto

```

1 CREATE OR REPLACE FUNCTION getValueF_Avaliation_userF(red integer ,
2   green integer , blue integer , radius float8 ,
3   nElem integer , lambda float8 , R integer [] )
4 RETURNS float8 AS
5 $$
6 DECLARE
7   ret float8;
8
9   i integer;
10  j integer;
11
12  sim float8;
13  div float8;
14 BEGIN
15   sim := 0;
16   div := 0;
17
18   SELECT SUM(sim1) INTO sim FROM
19     (SELECT * FROM rangeqc12($1,$2,$3,$4)) AS Q
20     WHERE id_ft = ANY($7);
21
22   FOR i IN 1..(CARDINALITY(R)-1) LOOP
23     FOR j IN (i+1)..CARDINALITY(R) LOOP
24       div := div + getSim(R[i], R[j]);
25     END LOOP;
26   END LOOP;
27
28   ret := ($5 - 1)*(1 - $6)*sim - 2 * $6 * div;
29
30   RETURN ret;
31 END
32 $$ LANGUAGE PLPGSQL;

```

Avaliador de conjunto

APÊNDICE E – GNE

```

1 CREATE OR REPLACE FUNCTION gne_constructor_userF(red integer,
2   green integer, blue integer, radius float8,
3   nElem int, lambda float8, alfa float8)
4 RETURNS int[] AS
5 $$
6 DECLARE
7   C1 CURSOR FOR SELECT * FROM rangeqc12($1, $2, $3, $4);
8
9   id_ft int[];
10  mmc float8[];
11  mmc_max float8;
12  mmc_min float8;
13
14  RCL int[];
15
16  p_mmc int;
17  k_construct float8;
18  i int;
19
20  R int[]; --O RETORNO DO CONSTRUTOR
21 BEGIN
22   p_mmc := 1;
23
24   WHILE cardinality(R) IS NULL OR cardinality(R) < $5 LOOP
25     mmc_max := -9999;
26     mmc_min := 9999;
27     i := 1;
28     id_ft := NULL;
29     mmc := NULL;
30     RCL := NULL;
31
32     FOR R1 IN C1 LOOP
33       IF array_position(R, R1.id_ft) IS NULL THEN
34         id_ft := array_append(id_ft, R1.id_ft);
35         mmc := array_append(mmc,
36           getmmc_userF($5, $6, R1.id_ft, R1.sim1, R, ($5 - p_mmc)));
37
38         IF ( mmc[i] > mmc_max) THEN
39           mmc_max := mmc[i];
40         END IF;
41
42         IF ( mmc[i] < mmc_min) THEN
43           mmc_min := mmc[i];

```

```

44     END IF;
45
46     i := i + 1;
47     END IF;
48     END LOOP;
49
50     k_construct := mmc_max - $7 * (mmc_max - mmc_min);
51
52     i := 1;
53     WHILE i <= cardinality(mmc) LOOP
54         IF (mmc[i] <= k_construct) THEN
55             RCL := array_append(RCL, id_ft[i]);
56         END IF;
57         i := i + 1;
58     END LOOP;
59
60     IF CARDINALITY(RCL) IS NOT NULL AND CARDINALITY(RCL) > 1 THEN
61
62         SELECT random()*(CARDINALITY(RCL) - 1) + 1 INTO i;
63         R := array_append(R, RCL[i]);
64
65     ELSIF CARDINALITY(RCL) = 1 THEN
66
67         R := array_append(R, RCL[1]);
68     ELSE
69         RAISE NOTICE '--- ERROR ---';
70     END IF;
71
72     p_mmc := p_mmc + 1;
73     END LOOP;
74
75     RETURN R;
76 END
77 $$ LANGUAGE PLPGSQL ;

```

Construtor GNE

```

1 CREATE OR REPLACE FUNCTION gne_localssearch_userF(red integer, green
2   integer,
3   blue integer, radius float8,
4   nElem int, lambda float8, R integer[])
5 RETURNS integer[] AS
6 $$
7 DECLARE
8   R_LINHA integer[];
9   R_LINHA2 integer[];
10
11   i integer;

```

```
11  j integer;
12
13  fR float8;
14  fR_Linha float8;
15  fR_Linha2 float8;
16
17  C1 CURSOR(ID_PARAMETRO INTEGER) FOR SELECT id_ft_2 FROM (
18    SELECT id_ft_2, (cube(c3.feature[1:3]) <-> cube(ft_Items)) AS
19    DIST_2 FROM
20    (SELECT id_ft_2, ft_Items FROM
21     (SELECT id_ft as id_ft_2, array[rangeqcl2.red, rangeqcl2.green,
22      rangeqcl2.blue] as ft_Items FROM rangeqcl2($1,$2,$3,$4)
23     ) AS Q
24     WHERE id_ft_2 != ALL($7)
25     ) AS itens_diversos_do_elemento_selecionado, COLOR C3
26     WHERE C3.id_ft = ID_PARAMETRO
27     ORDER BY DIST_2 DESC
28     LIMIT $5
29     ) AS os_n_elementos_mais_diversos;
30
31  R1 record;
32 BEGIN
33
34  fR := getValueF_Avaliation_userF($1, $2, $3, $4, $5, $6, $7);
35  fR_Linha := 9999;
36
37  FOR i IN 1..CARDINALITY($7) LOOP
38    FOR j IN 1..CARDINALITY($7) LOOP
39      IF i != j THEN
40        R_LINHA := R;
41
42        FOR R1 IN C1(R_Linha[i]) LOOP
43          R_LINHA2 := R_LINHA;
44
45          R_LINHA2[J] := R1.id_ft_2;
46          fR_Linha2 := getValueF_Avaliation_userF($1, $2, $3, $4, $5,
47            $6, R_LINHA2);
48          IF ( fR_Linha2 < fR_Linha) THEN
49            R_LINHA := R_LINHA2;
50            fR_Linha := fR_Linha2;
51          END IF;
52        END LOOP;
53      END IF;
54    END LOOP;
55
56  IF fR_Linha < fR THEN
57    R := R_Linha;
58    fR := fR_Linha;
```

```

55     END IF;
56
57     END LOOP;
58
59     RETURN R;
60 END
61 $$ LANGUAGE PLPGSQL;

```

LocalSearch GNE

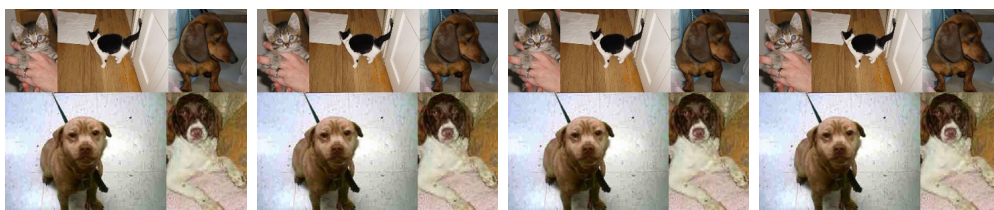
```

1 CREATE OR REPLACE FUNCTION GRASP_UserF(red integer, green integer,
2   blue integer, radius float8, lambda float8, nElem int,
3   nIt int, alfa float8)
4 RETURNS integer[] AS
5 $$
6 DECLARE
7   i int;
8
9   R_Line integer[];
10  R_Line2 integer[];
11  R_Line2_value float8;
12
13  R integer[];
14  R_value float8;
15 BEGIN
16   R_Value := -99999;
17   FOR i IN 0..$7 LOOP
18     R_Line := gne_constructor_userF($1, $2, $3, $4, $6, $5, $8);
19     R_Line2 := gne_localsearch_userF($1, $2, $3, $4, $6, $5, R_Line);
20     R_Line2_value := getValueF_Avaliation_userF($1, $2, $3, $4, $6, $5,
21       R_Line2);
22     IF cardinality(R) IS NULL THEN
23       R := R_Line2;
24       R_Value := R_Line2_value;
25     ELSIF R_Line2_value < R_value THEN
26       R := R_Line2;
27       R_Value := R_Line2_value;
28     END IF;
29   END LOOP;
30   return R;
31 END
32 $$ LANGUAGE PLPGSQL;

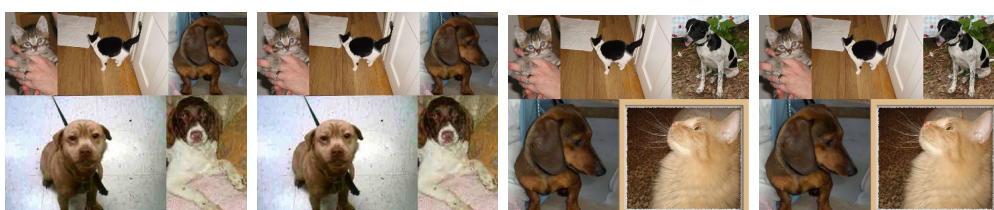
```

GRASP

APÊNDICE F – Comparação do MMR com e sem o uso da técnica OMNI



(a) $\lambda = 0$, sem OMNI (b) $\lambda = 0$, com OMNI (c) $\lambda = 0.1$, sem OMNI (d) $\lambda = 0.1$, com OMNI



(a) $\lambda = 0.3$, sem OMNI (b) $\lambda = 0.3$, com OMNI (c) $\lambda = 0.5$, sem OMNI (d) $\lambda = 0.5$, com OMNI



(a) $\lambda = 0.7$, sem OMNI (b) $\lambda = 0.7$, com OMNI (c) $\lambda = 0.9$, sem OMNI (d) $\lambda = 0.9$, com OMNI

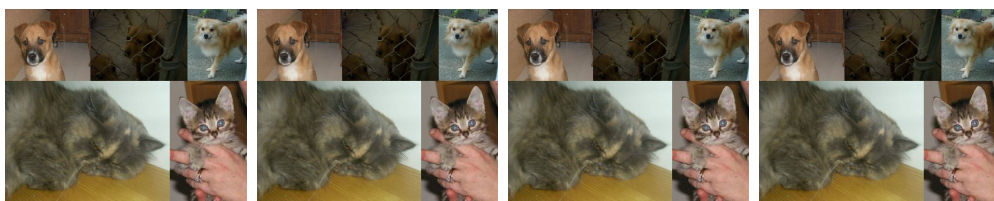


(a) $\lambda = 1$, sem OMNI (b) $\lambda = 1$, com OMNI

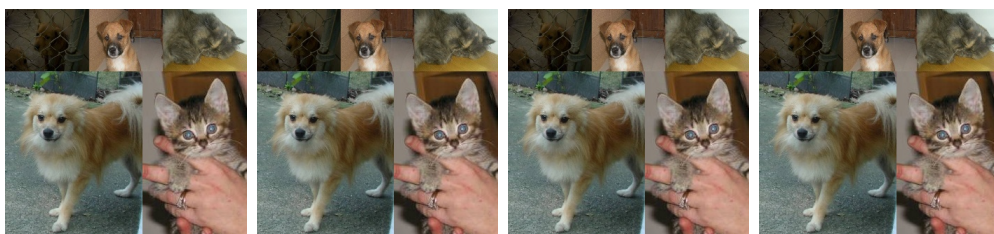
APÊNDICE G – Comparação do GMC com e sem o uso da técnica OMNI



(a) $\lambda = 0$, sem OMNI (b) $\lambda = 0$, com OMNI (c) $\lambda = 0.1$, sem OMNI (d) $\lambda = 0.1$, com OMNI



(a) $\lambda = 0.3$, sem OMNI (b) $\lambda = 0.3$, com OMNI (c) $\lambda = 0.5$, sem OMNI (d) $\lambda = 0.5$, com OMNI



(a) $\lambda = 0.7$, sem OMNI (b) $\lambda = 0.7$, com OMNI (c) $\lambda = 0.9$, sem OMNI (d) $\lambda = 0.9$, com OMNI



(a) $\lambda = 1$, sem OMNI (b) $\lambda = 1$, com OMNI

APÊNDICE H – Consulta por abrangência com uso da técnica OMNI

```

1 CREATE OR REPLACE FUNCTION rangeQ_L2_SIFT_OMNI (center_id integer ,
    radius FLOAT8)
2 RETURNS TABLE (id_ft INTEGER, dist double precision) AS $$
3 DECLARE
4     feature_aux FLOAT8 [];
5     dist_fc FLOAT8 [];
6 BEGIN
7     SELECT feat[1:99] INTO feature_aux FROM SIFT WHERE sift.id_ft = $1;
8
9     dist_fc = ARRAY( SELECT DISTINCT dist_l2 FROM sift_f_base WHERE
    sift_f_base.id_ft = $1);
10
11 RETURN QUERY
12     SELECT * FROM (
13         SELECT idfT, (cube(feature_aux) <-> cube(S1.feat[1:99])) as dist
14         FROM
15             (
16                 SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[1] - $2)) AND (SF.DIST_L2 < (dist_fc[1] + $2)) INTERSECT
17                 SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[2] - $2)) AND (SF.DIST_L2 < (dist_fc[2] + $2)) INTERSECT
18                 SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[3] - $2)) AND (SF.DIST_L2 < (dist_fc[3] + $2)) INTERSECT
19                 SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[4] - $2)) AND (SF.DIST_L2 < (dist_fc[4] + $2)) INTERSECT
20                 SELECT SF.id_ft as idfT FROM SIFT_F_BASE SF WHERE (SF.DIST_L2 >
    (dist_fc[5] - $2)) AND (SF.DIST_L2 < (dist_fc[5] + $2))
21             ) as fP
22         INNER JOIN SIFT S1 ON idfT = S1.ID_FT
23     ) as D
24     WHERE D.dist <= $2
25     ORDER BY D.dist;
26 END
27 $$ LANGUAGE PLPGSQL;

```