Análise de reprodutibilidade de artigo - GADIS

Marcelo P. R. Heredia, Eduardo C. Andrade, Michael L. S. Rosa, Duncan D. A. Ruiz

Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brazil

{marcelo.heredia, eduardo.a, michael.rosa}@edu.pucrs.br,
duncan.ruiz@pucrs.br

Abstract. *This article aims to describe the whole process of research reproduction of the article 'GADIS: A Genetic Algorithm for Database Index Selection' (Neuhaus et al., 2019). The reproduction of this research had support on the thesis 'Indexação Autônoma de Dados Utilizando Algoritmos Genéticos' (Miehe, 2019). One of the biggest worries of the Data Science area is the database performance. In order to lower cost and response time, several performance enhancing techniques have been developed, like indexing the tables. To help find out which columns from the tables to index, the reproduced article suggests using genetic algorithms to discover index combinations that offer the best balance between used space and gained query operation performance.*

Resumo. *Este artigo tem como objetivo descrever todo o processo de reprodução de pesquisa do artigo 'GADIS: A Genetic Algorithm for Database Index Selection' (Neuhaus et al., 2019). A reprodução dessa pesquisa teve apoio na dissertação 'Indexação Autônoma de Dados Utilizando Algoritmos Genéticos' (Miehe, 2019). Uma das maiores preocupações do ramo da ciência de dados é o desempenho do banco de dados. A fim de diminuir o custo e o tempo de resposta, foram desenvolvidas diversas técnicas de melhora de desempenho, como a indexação das tabelas. Para ajudar a saber quais colunas das tabelas indexar, o artigo reproduzido propõe o uso de algoritmos genéticos, usando-os para descobrir combinações de índices que ofereçam o melhor balanço entre espaço utilizado e desempenho de operações de consulta ganho.*

1. Introdução

O crescimento no volume de dados nos últimos anos vem se dando de forma exponencial. Muitos negócios se viram obrigados a buscar soluções para garantir um melhor desempenho de suas bases de dados, com isso surgiram diferentes estratégias, com objetivo de melhorar o desempenho dos sistemas. O baixo desempenho de um banco de dados pode acarretar não somente um desperdício de tempo para o negócio, mas também um alto custo computacional, fazendo com que sejam destinados mais recursos para a aplicação. Uma técnica usada para melhorar este desempenho da base de dados é a inclusão de índices, que tem como objetivo tornar uma consulta a um banco de dados mais rápida. Seu funcionamento se dá semelhante ao sumário de um livro, em que nele você encontra a página correta de acordo com o assunto e não precisa ficar folheando o livro até encontrar o assunto desejado.

Ao mesmo tempo que a criação de índices é capaz de aprimorar o desempenho de rotinas de consultas a bancos de dados, existem consequências. Criar índices no banco de dados aumenta o espaço em disco utilizado e, além disso, diminui o desempenho de

rotinas de inclusão/exclusão do banco de dados, já que cada uma dessas alterações deve ser refletida nos índices.

Tendo em vista o problema descrito, o objetivo do artigo reproduzido é otimizar o tempo de consulta a um banco de dados relacional, utilizando a técnica de indexação descrita anteriormente. Para que isso seja possível é usado o algoritmo genético, que tem como objetivo explorar as soluções de indexação buscando um conjunto de índices que reduza o tempo de respostas de conjuntos de consultas submetidas de forma frequente por um determinado período de tempo e, ao mesmo tempo, considerando a quantidade de memória secundária alocada. Adicionalmente, espera-se obter uma boa relação de custo-benefício entre a configuração de índices gerada e a sua utilização pelo otimizador do sistema no plano de execução das consultas.

Este artigo se propõe a reproduzir a pesquisa desenvolvida no artigo GADIS: A Genetic Algorithm for Database Index Selection (Neuhaus et al., 2019), para isto, a pesquisa terá apoio da dissertação Indexação Autônoma de Dados Utilizando Algoritmos Genéticos (Miehe, 2019). Para realizar a reprodução, faremos uma introdução do funcionamento do algoritmo genético e seus operadores. Descreveremos a solução proposta detalhando os experimentos realizados. Após isso, faremos uma análise comparando os resultados da nossa reprodução com os do artigo em si. Nosso escopo também inclui uma proposta de melhoria a ser desenvolvida no trabalho seguinte.

2. Algoritmo Genético

O algoritmo genético é uma técnica de busca inspirada na teoria de evolução natural de Charles Darwin, esta técnica é muito usada na computação para achar soluções aproximadas em problemas de busca e de otimização, este algoritmo simula o processo de seleção natural, onde os indivíduos mais aptos de uma população são selecionados para reprodução a fim de produzir descendentes da próxima geração.

O processo de seleção natural começa com a seleção dos indivíduos mais aptos de uma população, o cálculo da aptidão de um indivíduo é uma função definida de acordo com o objetivo do problema. Eles produzem descendentes que herdam as características dos pais e serão adicionados à próxima geração. Este processo continua até se obter a população desejada, até que o critério de parada seja atendido ou em caso de convergência, e então no final como resultado, temos uma geração com os indivíduos mais aptos.

Cinco etapas são consideradas em um algoritmo genético, sendo elas: geração de população inicial, função de avaliação (ou função de aptidão), seleção, cruzamento, mutação e critério de parada.

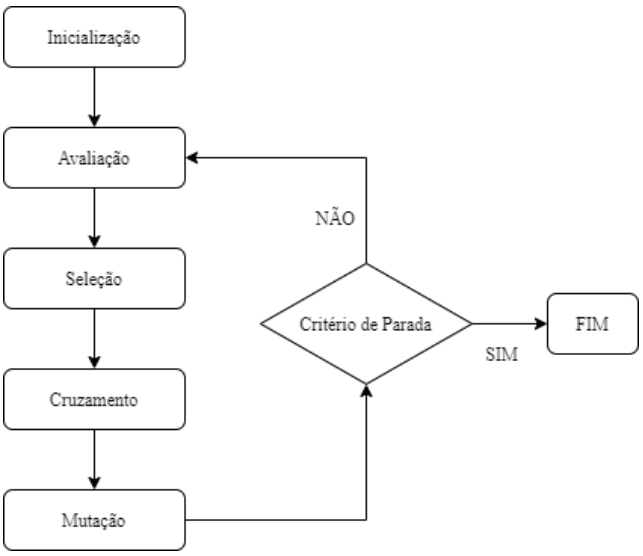

Figura 1. Fluxograma do Algoritmo Genético.
Fonte: Elaborada pelos autores.

2.1. População inicial

A inicialização da população começa com a criação de uma população de forma aleatória, onde cada membro dessa população é chamado de indivíduo e é uma proposta de solução para o problema que você deseja resolver.

Um indivíduo é caracterizado por um conjunto de parâmetros conhecidos como genes. Os genes são unidos em uma lista para formar um cromossomo.

Para aplicações em algoritmo genético, o cromossomo de um indivíduo é representado por uma string ou um vetor, e os genes normalmente são representados por valores binários (1 e 0).

2.2. Função de Avaliação

A função de avaliação, também conhecida como função de aptidão, determina o quão apto um indivíduo está para sobreviver e passar seus genes para as próximas gerações. Nesta função é dada uma pontuação de aptidão para cada indivíduo. A probabilidade de um indivíduo ser selecionado para reprodução é baseada em sua pontuação de aptidão.

2.3. Seleção

A ideia da etapa de seleção é selecionar os indivíduos mais aptos e fazer com que seus genes passem para as próximas gerações. Nesta, um número x de indivíduos (pais) são selecionados aleatoriamente e escolhidos para passar adiante. Indivíduos com alta aptidão têm mais chance de serem selecionados para reprodução.

2.4. Cruzamento

O cruzamento é a etapa mais importante em um algoritmo genético, pois é nela que vamos fazer a troca de genes entre indivíduos. Existem várias formas de se fazer o cruzamento, abaixo temos um exemplo de cruzamento por ponto de cruzamento:

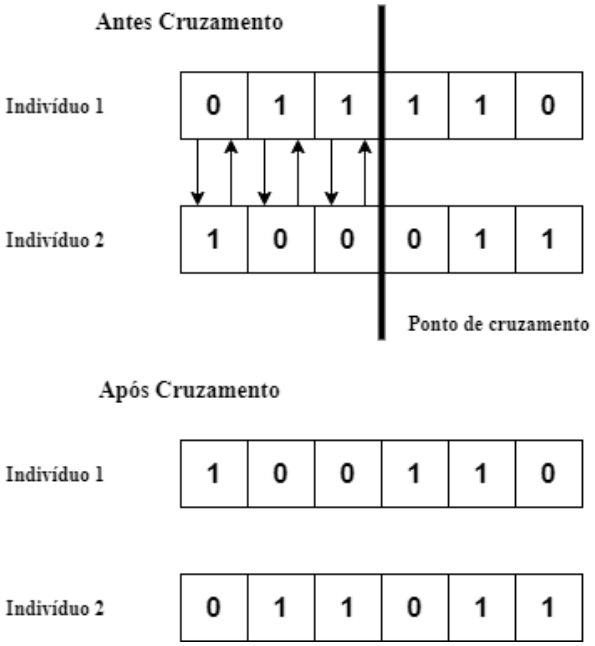

Figura 2. Processo de cruzamento através de ponto de cruzamento.
Fonte: Elaborada pelos autores.

2.5. Mutação

Após a formação de novos descendentes, alguns destes indivíduos podem ser submetidos a uma mutação com probabilidade aleatória. Isso implica que seu cromossomo pode sofrer pequenas alterações em seus genes, fazendo com que possam se diversificar mais facilmente. Existem diversas formas de mutação, abaixo temos uma das mais comuns:

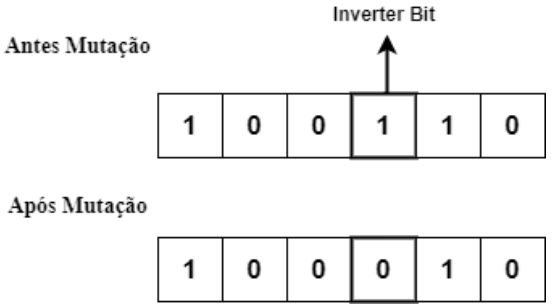

Figura 3. Processo de mutação por inversão de bit.
Fonte: Elaborada pelos autores.

2.6. Critério de parada

O critério de parada define quando o algoritmo deve encerrar sua execução e devolver seus resultados. Um algoritmo genético pode ter diversos critérios de parada, isso depende do problema a ser resolvido. Os principais critérios de parada são a parada após um número *n* de gerações e a parada após encontrar uma solução desejada (seja ela a solução ideal ou apenas uma aproximação boa o suficiente, a critério do desenvolvedor). Outro critério de parada muito usado é o por convergência, onde o algoritmo para caso não haja uma melhoria na aptidão após um número *n* de gerações.

3. Desenvolvimento

3.1. Abordagem da proposta de pesquisa

Conforme explicado, o artigo reproduzido propõe a criação de índices para otimizar o tempo de consulta, porém para isso, é necessário saber qual a melhor combinação de colunas de um esquema de banco de dados para indexação, buscando equilibrar o tempo de resposta de consulta, com o espaço ocupado na memória da base dados para criação destes índices.

O esquema de dados usado nesta pesquisa possui 45 colunas possíveis de serem indexadas, que gera 2^{45} combinações possíveis, para buscar a melhor combinação é usado o algoritmo genético, onde os indivíduos serão definidos com um vetor binário composto pelo número de colunas que podem ser indexadas, e cada gene representa se a coluna deve ser indexada ou não.

3.1.1. TPC-H

Para a geração da base foi usado um benchmark comercial gratuito da TPC, que é uma empresa sem fins lucrativos. O benchmark escolhido na pesquisa foi o TPC-H benchmark, ele possui um esquema de banco de dados, uma carga de trabalho e testes para métricas de desempenho. O tamanho do banco de dados varia de acordo com a escala escolhida, para reprodução desta pesquisa foi mantida a escala de 1, que gera um banco de dados de 1GB de tamanho.

O TPC consiste em um esquema de banco de dados, um conjunto de tabelas orientadas a um modelo comercial de negócios, além de modificações simultâneas de dados. Essas modificações simulam sistemas de suporte à decisão que usam um conjunto de dados massivo. As consultas provenientes desse benchmark são consideradas de alto grau de complexidade e fornecem vários aspectos da capacidade do sistema de processar consultas, como desempenho do sistema e complexidade do processamento de consultas simulando ambientes único e multi-usuários. (Miehe, 2019).

3.1.2. Função de Avaliação

Uma das etapas mais importantes do algoritmo genético é a escolha de uma boa função de avaliação, a pesquisa aborda uma avaliação multi-objetiva para definir a qualidade dos indivíduos. A função considera dois aspectos: o tempo de resposta sobre um conjunto de consultas e o espaço ocupado em disco pelos índices.

$$Objetivo1 = \frac{tempo\_individuo\_tudo\_indexado}{tempo\_individuo\_atual}$$ **(Miehe, 2019)**

O objetivo 1 tenta minimizar o tempo de resposta de um determinado conjunto de consultas.

$$Objetivo2 = \frac{tamanho\_estado\_inicial}{tamanho\_individuo\_atual}$$ **(Miehe, 2019)**

O objetivo 2 visa minimizar o espaço ocupado em disco pelos índices.

3.1.3. Operadores de Seleção

Uma importante etapa do algoritmo genético é a seleção dos indivíduos de uma população, a primeira etapa da seleção é o NSGA-II que é um algoritmo evolutivo para resolver problemas de otimização multiobjetivos. Após a definição das aptidão para os dois objetivos de cada indivíduo, o NSGA-II é utilizado para estabelecer uma dominância entre os indivíduos, facilitando o processo de escolha dos melhores.

A segunda etapa é a seleção em si, iniciando pelo torneio, neste processo são selecionados 5 indivíduos através do torneio, onde para cada seleção são testados 2 indivíduos e coletado o dominante dos dois.

O operador de cruzamento implementado consiste em alterar os indivíduos selecionados, seja através do torneio ou os que são simplesmente passados adiante. Para isto foi usado a técnica de cruzamento de dois pontos, esta técnica é bem similar a explicada anteriormente porém o indivíduo tem uma subdivisão a mais. A chance selecionada de cruzamento foi de 65%.

O operador de mutação selecionado é o de inversão de bit, o mesmo explicado anteriormente, a chance de mutação é de 20%, os indivíduos selecionados para sofrer mutação têm 2.2% de chance de mutação por gene/bit dos mesmos.

3.2.4. Critério de Parada

Para o critério de pesquisa optou-se pela abordagem k-iterações, onde o algoritmo finaliza de forma precoce sua execução, caso não haja melhoria nos valores obtidos com a função de avaliação após k iterações consecutivas. Como valor de k foi definido 10% de $nger$ por acreditar ser um bom valor, uma vez que não está claro qual o número de gerações que deve-se considerar como padrão na não melhoria dos resultados (Miehe, 2019). O total de gerações é 100, portanto, k vale 10.

3.2. Processo de reprodução da pesquisa

Para iniciar a reprodução da pesquisa, foi feito um estudo sobre algoritmos genéticos e seu funcionamento. Consultados, majoritariamente, os conteúdos do livro Algoritmos Genéticos (Linden, 2006). Além disso, houve um bom proveito da disciplina de Inteligência Artificial da PUCRS, lecionada pela professora Silvia W Moraes.

Após adquiridos os conhecimentos necessários para reprodução do algoritmo, foi inicializado um repositório para armazenar os códigos fonte e demais desenvolvimentos referentes a reprodução do artigo (Andrade et al., 2021).

O processo de reprodução se dividiu em duas etapas iniciais, são eles a construção dos códigos fonte do algoritmo genético e a criação e estruturação de um banco de dados utilizando o TPC-H.

3.2.1. Construção do algoritmo genético

Para criação do algoritmo genético, foi utilizada a mesma biblioteca usada no artigo, DEAP (Fortin et al., 2012), programando na linguagem Python. Utilizando os conhecimentos adquiridos sobre algoritmos genéticos, foi possível desenvolver o código

seguindo as especificações dadas no artigo e usando como apoio algumas especificações que estão contidas apenas na dissertação (Miehe, 2019).

Para codificação do indivíduo, foi utilizado um vetor de tamanho 22, onde os 22 genes são as colunas que contém mais dados do esquema.

Para a função de avaliação, foram levados em conta apenas o tempo de consulta dos indivíduos e o espaço ocupado em disco pela criação de índices em função do indivíduo, não foi calculado o índice QPHH pois seria necessário um experimento diferente utilizando inserções e exclusões da base de dados, o que aumentaria muito o tempo de execução do algoritmo.

3.2.2. Banco de dados usando TPC-H

Para a criação do banco de dados nos baseamos no projeto "TPC-H Benchmark, specific for MYSQL" (Ribeiro, 2016), onde a autora descreve como utilizar o TPC-H para gerar os dados e incluí-los no MYSQL, além de conter as consultas corretas do TPC-H.

Para conseguir diminuir o tempo de execução, já que o original demorava em torno de uma semana, aumentamos os buffers do gerenciador de banco de dados. Com isso o tempo de execução foi reduzido para 6 horas por execução. Os valores utilizados constam na tabela 1.

Tabela 1. Configurações ajustadas no SGBD.
Fonte: Elaborada pelos autores.

Configuração	Valor inicial	Valor utilizado
innodb_buffer_pool_size	550M	2048M
read_buffer_size	64K	256K
read_rnd_buffer_size	256K	1024K
innodb_log_buffer_size	1M	4M
sort_buffer_size	256K	1024K
join_buffer_size	256K	1024K
innodb_log_file_size	48M	256M
max_allowed_packet	4M	16M

4. Resultados

Ao analisar os resultados obtidos com a nossa reprodução e comparar com os obtidos no artigo vemos que os tamanhos são diferentes, mas ainda seguindo o padrão do artigo, com o melhor indivíduo sendo maior que apenas as chaves e menor que todas as colunas indexadas. (Fig. 4)

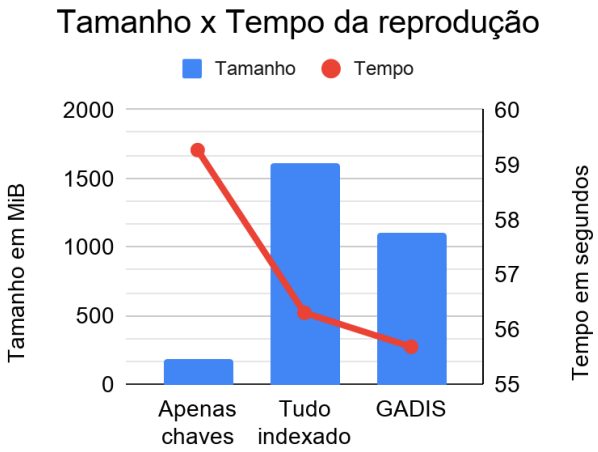

Figura 4. Gráfico do tamanho dos índices e o tempo médio de resposta.
Fonte: Elaborada pelos autores.

No caso do tempo de resposta das consultas, obtivemos pouca variação entre o pior e o melhor resultado, mas é possível notar que mesmo nessa faixa pequena de diferença, em todas as execuções, o melhor indivíduo tende a ser mais rápido a cada geração. (Fig. 5)

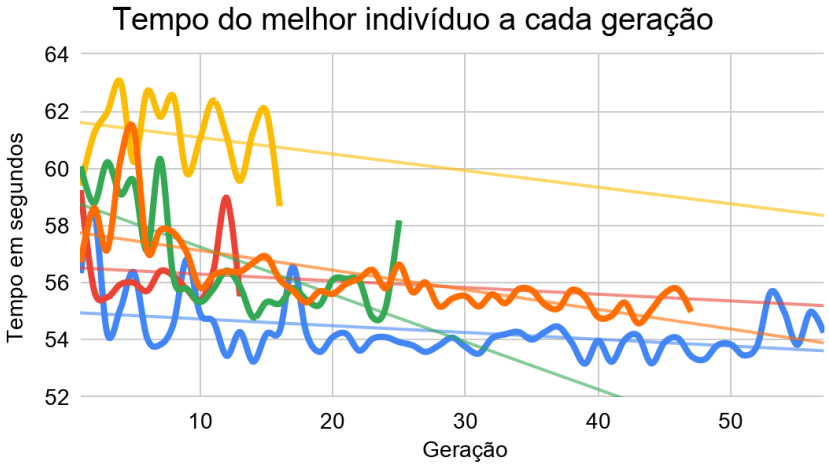

Figura 5. Tempo levado por cada geração do algoritmo, com tendências.
Fonte: Elaborada pelos autores.

No artigo, o tempo levado para o estado inicial (Fig. 6) é mais que o dobro do obtido em nossa reprodução. Acreditamos que seja devido à configuração de buffers que foi necessária para reproduzir o artigo dentro do tempo permitido, que pode ter diminuído todos os tempos de execução das consultas.

Methods	QPHH	Time	Index Size
Initial State	1678	149.3s	**599 MB**
Random Search	1864	145.3s	1196 MB
GADIS-[$\mathcal{Q}$]	2631	71.2s	*1193* MB
GADIS-[$\mathcal{T}$]	**3077**	**60.6s**	1600 MB

Figura 6. Resultados obtidos no artigo reproduzido.
Fonte: (Neuhaus et al., 2019).

Na dissertação o tempo também é diferente do nosso, porém os tamanhos estão mais próximos. Como o gerador de dados TPC-H não gera tabelas com o tamanho exato pedido, o tamanho final das tabelas e, portanto, o tamanho dos índices pode variar como visto na diferença entre dissertação, artigo e reprodução.

Métodos	QphH	Tempo de resposta (em segundos)	Memória em Disco (em Mb)
EDB	3044.05	60.59	755.18
POWA	3014.55	61.19	894.60
ITLCS	3136.11	59.35	**758.83**
Estado Inicial	1713.82	153.78	395.00
Tudo Indexado	3139.38	61.53	2050.00
Busca Aleatória	1690.67	150.09	1460.93
AG para SGBDs	**3175.58**	**59.19**	782.87

Figura 7. Resultados obtidos na dissertação.
Fonte: (Miehe, 2019).

5. Sugestão

A proposta da pesquisa reproduzida se propõe em pegar cada cromossomo, que representa a presença ou ausência de índices, para criar ou remover os índices para então executar, uma proposta de sugestão de melhoria seria ordenar os cromossomos antes de começar a executar o conjunto de consultas, de tal forma que minimize as criações e remoções desses índices, trazendo um melhor desempenho do algoritmo, fazendo com que minimize a criação e remoção de índices.

Para resolver o este problema de criação e remoção de índices, o grupo propõe usar um algoritmo de *Distância de Levenstein* também conhecido como distância de edição, que é dado pelo número mínimo de operações necessários para se transformar uma string em outra, operações como inserções, remoções, deleção ou substituição de um carácter.

O objetivo desta sugestão é fazer com que se ache a melhor ordem que minimize as distâncias de Levenshtein entre cada 2 cromossomos é obtido por programação dinâmica.

6. Conclusão

A indexação de banco de dados é uma otimização bastante eficaz e com alta complexidade de decisão, tornando-se um problema muito interessante de ser abordado com técnicas de Inteligência Artificial, como Algoritmos Genéticos ou Machine Learning, com objetivo de busca de soluções o mais otimizadas possíveis, ou dentro de critérios de aceitação.

O método proposto pelo artigo consegue sugerir seleções de índices que reduzem bastante o tempo de execução de consultas, levando de um estado inicial de 149 segundos para até 60 segundos de tempo de consulta. Além disso, essa otimização vem

sem um aumento tão grande no espaço em disco consumido pelos índices, de 2050 Mb com tudo indexado para 1460 Mb com a sugestão do algoritmo genético.

A reprodutibilidade foi considerada um ponto fraco do artigo, o motivo disso é que o artigo não acompanha os códigos fonte, tornando necessária a codificação da solução proposta a fim de reproduzir e poder, finalmente, comparar os resultados obtidos com os expostos no artigo. Além disso, algumas informações não ficam bem explicitadas no artigo em si, como as fórmulas da função de avaliação, quais consultas exatamente foram usadas, como as mesmas foram chamadas e como foram calculados os resultados de tempo de consulta e espaço em disco dos índices.

Ao concluir a codificação notamos alguns resultados com diferenças, como o espaço ocupado em disco sendo inferior ao do artigo quando a base não tem índices além de PKs e FKs, ou o tempo de execução tendo uma variação bem inferior: 52 a 64 segundos contra 59 a 154 segundos. Não podemos ter certeza que o problema não é apenas na codificação, pois temos apenas os dados do artigo para comparar, sem saber como foi feita a codificação.

7. Referências

Andrade, E., Herédia, M., & Rosa, M. (2021). *Reprodução de Pesquisa*. GitHub. https://github.com/tecnicasilegais/Integradora2

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012, jul). Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, *13*, 2171-2175. https://github.com/DEAP/deap

Linden, R. (2006). *Algoritmos Genéticos* (3rd ed.). Ciência Moderna. https://www.algoritmosgeneticos.com.br/

Miehe, P. (2019). Indexação Autônoma de Dados Utilizando Algoritmos Genéticos.

Neuhaus, P., Couto, J., Wehrmann, J., Ruiz, D., & Meneguzzi, F. (2019). GADIS: A Genetic Algorithm for Database Index Selection. *The 31st International Conference on Software Engineering and Knowledge Engineering*.

Ribeiro, C. (2016). *TPC-H Benchmark, specific for MYSQL*. GitHub. https://github.com/catarinaribeir0/queries-tpch-dbgen-mysql