UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

Algoritmos Genéticos de Auto Aprendizado

Evolução na Obtenção de Soluções em Roteamento e Otimização Contínua

Gustavo dos Reis Oliveira

UFG
UNIVERSIDADE
FEDERAL DE GOIÁS

A capa deste trabalho foi concebida para simbolizar de forma conceitual o funcionamento dos algoritmos genéticos de auto aprendizado, unindo elementos naturais e tecnológicos. A árvore, elemento central, representa os processos de adaptação e crescimento dinâmico, ajustando-se para captar mais energia solar, assim como os algoritmos genéticos se adaptam durante o processo de busca para otimizar e melhorar a qualidade das soluções.

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

GUSTAVO DOS REIS OLIVEIRA

Algoritmos Genéticos de Auto Aprendizado

Evolução na Obtenção de Soluções em Roteamento e Otimização Continua

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): GUSTAVO DOS REIS OLIVEIRA

Título do trabalho: Algoritmos Genéticos de Auto Aprendizado

Evolução na Obtenção de Soluções em Roteamento e Otimização Continua

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Gustavo Dos Reis Oliveira**, **Discente**, em 15/01/2025, às 17:33, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 16/01/2025, às 18:28, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5089776** e o código CRC **8FA574A5**.

Referência: Processo nº 23070.001586/2025-68 SEI nº 5089776

GUSTAVO DOS REIS OLIVEIRA

Algoritmos Genéticos de Auto Aprendizado

Evolução na Obtenção de Soluções em Roteamento e Otimização Continua

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.
Orientador: Prof. Dr. Fernando Marques Federson

Goiânia
2025

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

OLIVEIRA, GUSTAVO DOS REIS
Algoritmos Genéticos de Auto Aprendizado [manuscrito] : Evolução na Obtenção de Soluções em Roteamento e Otimização Continua / GUSTAVO DOS REIS OLIVEIRA. - 2025.
97 f.

Orientador: Prof. Dr. Fernando Marques Federson.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. inteligência artificial. 2. otimização. 3. metaheurísticas. I. Federson, Fernando Marques , orient. II. Título.

CDU 004

GUSTAVO DOS REIS OLIVEIRA

Algoritmos Genéticos de Auto Aprendizado

Evolução na Obtenção de Soluções em Roteamento e Otimização Contínua

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 17 de dezembro de 2024.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Prof. Dr. Sávio Salvarino Teles de Oliveira
(INF-UFG)

GUSTAVO DOS REIS OLIVEIRA

Algoritmos Genéticos de Auto Aprendizado

Evolução na Obtenção de Soluções em Roteamento e Otimização Continua

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Algoritmos Genéticos de Auto-Aprendizado**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: inteligência artificial, modelos grandes de linguagem, geração automática de datasets.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Self-Learning Genetic Algorithms**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

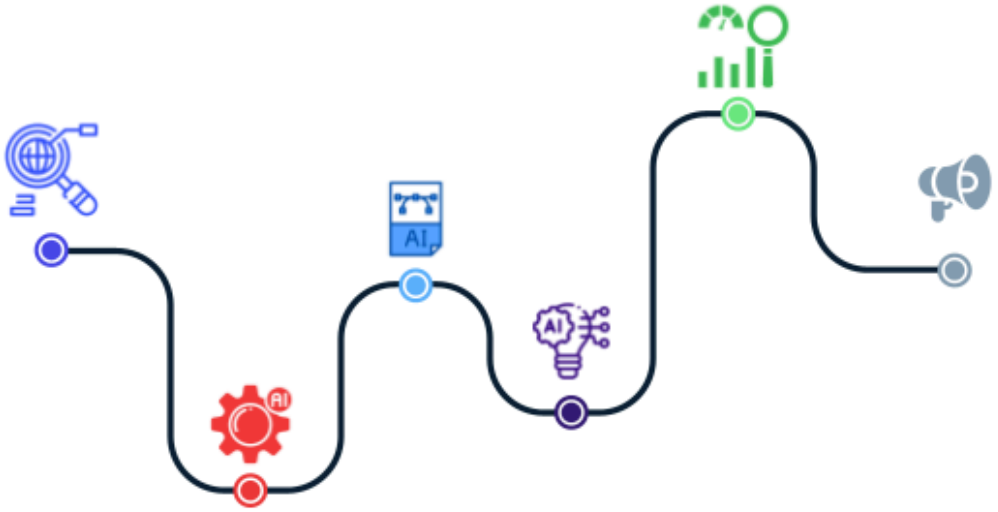
Keywords: artificial intelligence, large language models, automatic dataset generation.

Goiânia

2025

Minha Jornada

Gustavo dos Reis Oliveira



Especialista em:

Algoritmos Genéticos de Auto-Aprendizado

Semanas 1, 2 e 3

Definição da área de conhecimento em otimização com IA e estudo de artigos selecionados

Semanas 4 e 5

Exploração e uso de ferramentas para desenvolvimento em Algoritmos Genéticos, com foco em Auto Aprendizado.

Semana 6

Estudo da técnica de Auto Aprendizado LGA e definição de case de aplicação VRP

Semanas 7-8

Aplicação da técnica LGA ao VRP e simulações

Semana 9

Busca de benchmarks de otimização contínua e refinamento da técnica

Semana 10

Realização de teste e comparações de resultados

MINHA JORNADA

Nome: Gustavo dos Reis Oliveira

Especialidade: Algoritmos Genéticos de Auto-Aprendizado

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

Minha jornada começou com a busca por uma área de especialização em otimização, motivada pelo interesse que sempre tive no tema, mas que não explorei profundamente durante o curso. Decidi procurar problemas de otimização que achava interessantes. Na **Semana 1**, realizei um brainstorming de problemas de otimização e identifiquei vários temas relacionados a fluxos, consumo e distribuição. Apesar de tudo estar ainda muito abstrato para mim, senti vontade de trabalhar em soluções para esses desafios. Assim, comecei a pesquisar técnicas de inteligência artificial que pudessem ser aplicadas a esses problemas. Nesse processo, encontrei o artigo de revisão *Artificial Neural Networks Based Optimization Techniques: A Review*, que me ajudou a compreender melhor essas técnicas e suas limitações. A partir desse artigo, elaborei um esquema básico de organização dos algoritmos de inteligência artificial voltados para otimização. Além disso, descobri o *AI4OPT: AI Institute for Advances in Optimization Summary*, um instituto dedicado ao uso de inteligência artificial em avanços na área de otimização. Explorei o site do instituto, estudei os trabalhos publicados e procurei entender as linhas de pesquisa e técnicas utilizadas. Na **Semana 2**,

[1] Dez semanas, entre setembro de 2024 e dezembro de 2024.

continuei meus estudos aprofundando os artigos encontrados, com o objetivo de construir uma base sólida sobre otimização. Durante a **Semana 3**, concentrei meus esforços no estudo dos algoritmos genéticos, que chamaram minha atenção tanto pela inspiração nos conceitos de seleção natural quanto por suas diversas aplicações potenciais. Meu principal material de estudo foi o artigo *A Review on Genetic Algorithm: Past, Present, and Future*, que trouxe insights importantes sobre o funcionamento dos algoritmos genéticos, seus componentes, problemas e fluxos de execução. Essa etapa foi fundamental para construir uma base sólida sobre essa classe de algoritmos. Além disso, participei do evento *SVR 2024*, onde apresentei um trabalho e acompanhei tendências atuais na área. Os materiais relacionados a estas três Semanas podem ser encontrados no **Apêndice 1**.

Para aprofundar minha compreensão dos algoritmos genéticos, comecei explorando os desafios e trade-offs inerentes a essa abordagem, buscando identificar problemas relevantes que pudessem ser abordados. Na **Semana 4**, foquei nos problemas específicos e nas limitações das soluções baseadas em algoritmos genéticos, mas inicialmente tive dificuldades em encontrar artigos alinhados ao tema desejado. Dediquei tempo a ajustar os descritores de busca, o que resultou em materiais mais relevantes, incluindo artigos que discutiam a aplicação desses algoritmos a problemas do mundo real, os desafios de implementação e as perspectivas para pesquisas futuras. Paralelamente, procurei frameworks e ferramentas que facilitassem o desenvolvimento e a escalabilidade dessas abordagens. Na **Semana 5**, intensifiquei a busca por artigos recentes, especialmente em conferências renomadas como o *Congress on Evolutionary Computation* e a *Genetic and Evolutionary Computation Conference (GECCO)*, o que me permitiu identificar áreas contemporâneas de investigação. Esses estudos destacaram tópicos como otimização contínua e a integração de auto-adaptação e aprendizado em algoritmos genéticos, consolidando minha base teórica e prática para desenvolver soluções inovadoras. A partir disso, consolidei meu foco na pesquisa de algoritmos genéticos com capacidades de auto-adaptação e aprendizado. Os materiais relacionados a estas duas Semanas podem ser encontrados no **Apêndice 2**.

Na **Semana 6**, aprofundei-me nas técnicas de autoaprendizado e adaptação em algoritmos genéticos, estudando artigos recentes apresentados em conferências de otimização entre 2020 e 2024. Meu foco foi entender técnicas que tornassem a etapa de tunagem de parâmetros do algoritmo genético mais adaptativa e baseada em auto aprendizado em relação ao problema. Durante essa pesquisa, encontrei o artigo *Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization*, que apresentou o *Learn Genetic Algorithm (LGA)*. Esse algoritmo chamou minha atenção por utilizar mecanismos de atenção para aprendizado em algoritmos genéticos. Apesar disso, fiquei com dúvidas, pois o artigo parecia prometer muito, e ainda não tinha encontrado muitas referências a ele, embora tenha recebido o prêmio de melhor artigo na track de algoritmos genéticos na GECCO 2023. Testei o repositório oficial do artigo para avaliar a viabilidade de seguir com essa abordagem. Paralelamente, estudei o problema de roteamento de veículos, buscando compreender sua modelagem e as diferentes formas de abordá-lo em contextos variados. Esse problema me interessou particularmente por ser prático, representar um desafio complexo e estar alinhado com meu desejo de trabalhar em problemas de otimização significativos. Nessa mesma Semana, fiz algumas modelagens simples, aplicando algoritmos genéticos para resolver um problema de roteamento. Os materiais relacionados a esta Semana podem ser encontrados no **Apêndice 3**.

Na **Semana 7**, me aprofundei no estudo da técnica LGA, revisitei o artigo que a descreve, bem como o repositório oficial de sua implementação, buscando compreender seus detalhes teóricos e práticos. Paralelamente, explorei um framework do Google voltado para a implementação paralela e otimizada de algoritmos genéticos, o que ajudou a planejar experimentos mais eficientes. Optei por continuar com o framework do artigo . Em um dos testes, tentei aplicar o LGA ao problema de roteamento de veículos (VRP), mas enfrentei dificuldades devido a valores numéricos infinitos que surgiram durante a execução. Para lidar com esses problemas, investiguei mais a fundo a classe dos problemas NP-difíceis e suas características. Durante a **Semana 8**, simplifiquei o modelo do problema de roteamento para entender melhor o comportamento do LGA nesse contexto. Realizei também uma abordagem de força bruta para calcular a melhor solução possível, a fim de comparar com os resultados obtidos pelo LGA. Embora os ajustes tenham mitigado parcialmente a instabilidade numérica, ela continuou presente em algumas situações, indicando a

necessidade de mais refinamentos na abordagem, investiguei uma implementação mais customizada para tentar resolver o problema alterando algumas coisas no core do algoritmo. Essa Semana foi importante para mim, pois consegui explicar para os meus colegas melhor a área que estava estudando e o problema que queria resolver, consegui explicar de forma simples e eles entenderam. Os materiais relacionados a estas duas Semanas podem ser encontrados no **Apêndice 4**.

Dados os problemas contínuos com a implementação de uma solução para o problema de roteamento com LGA, na **Semana 9**, investiguei mais profundamente o funcionamento do LGA, percebi que ele é projetado especificamente para problemas de otimização contínua, o que explica sua baixa performance no problema de roteamento de veículos, que é um caso de otimização combinatória. Essa constatação me abalou, pois eu havia trabalhado nesse algoritmo nas últimas semanas e estava empolgado com seu potencial para os problemas que eu queria atacar, mas decidi continuar com essa técnica e busca por outros problemas. Tive uma reunião com a Professora Doutora Telma Woerle de Lima Soares, que é especialista em algoritmos genéticos, para obter direções de pesquisa e problemas que eu poderia abordar com otimização contínua. Após isso direcionei meu foco para testar o LGA em problemas mais alinhados ao seu propósito. Para isso, utilizei o benchmark de funções contínuas proposto no artigo original, complementando com a exploração de outros benchmarks amplamente discutidos em conferências da área. Essa abordagem permitiu avaliar o desempenho do LGA em cenários adequados, contribuindo para um refinamento mais direcionado da técnica. Os materiais relacionados a esta Semana podem ser encontrados no **Apêndice 5**.

Já na **Semana 10**, final do processo, eu estava um pouco triste ainda por não ter conseguido solucionar o problema com a técnica que estudei, mas prossegui com os experimentos e realizei testes comparativos do LGA utilizando o benchmark BBOB COCO, avaliando sua performance em relação a um GA comum implementado no repositório oficial. Os resultados indicaram que o LGA apresentou desempenho inferior ao GA padrão na maioria dos casos, reforçando a necessidade de investigar aplicações práticas para explorar melhor suas potencialidades.Dado que ainda tinha um tempo para experimentar algo novo, pensei em testar uma abordagem de seleção contínua de features, onde o LGA atribui

pesos às features de um dataset antes de passá-las para um modelo de redes neurais simples. Implementei essa estratégia, que mostrou-se promissora, resultando em um aumento significativo na acurácia do modelo, com um ganho percentual médio de 18,92% em comparação ao modelo sem a seleção de features implementada, isso renovou os ânimos na técnica estudada e me ajudou a fechar esse processo mais feliz. Os materiais relacionados a esta última Semana podem ser encontrados no **Apêndice 6**.

Quando comecei o processo da Residência, eu sabia pouco sobre otimização e menos ainda sobre algoritmos genéticos, mas decidi me aventurar nessa área em busca de explorar conhecimentos novos. Através do esforço, dedicação e curiosidade, desenvolvi habilidades que me ajudaram a entender melhor os desafios, encontrar problemas e pensar em soluções nessa área. Ao longo desse processo, passei por frustrações, mas também tive aprendizados muito valiosos. Isso me deu confiança para conversar com uma especialista e acompanhar bem o que era discutido. Mesmo não conseguindo aplicar a técnica ao problema de roteamento de veículos, que foi o que inicialmente despertou meu interesse, consegui usá-la em outros problemas e tive bons resultados. Esse aprendizado não só ampliou meu conhecimento técnico, como também me ajudou a adaptar técnicas a diferentes contextos, abrindo novas possibilidades tanto para pesquisa quanto para uso prático no futuro. Sou grato ao processo e a todas as pessoas envolvidas. O aprendizado adquirido me trouxe lições valiosas,uma é a importância de ter resiliência: continuar aprendendo, estudando e se adaptando, assim como os algoritmos genéticos se ajustam para encontrar as melhores soluções, precisamos nos adaptar na vida para alcançar nossos objetivos. Além disso, aprendi que, às vezes, é necessário dar um passo para trás para tomar impulso na direção certa.

Agradeço imensamente, nesta jornada, à minha noiva Meilin Miller, que sempre me deu suporte emocional e racional, carinhos e puxões de orelha nos momentos em que mais precisei e que me incentivou a embarcar em jornadas novas. Um agradecimento especial aos meus amigos Evellyn Nicole, Isadora Stéfany e Guilherme Henrique, por me apoiarem, pelos momentos de amizade, por compartilharem insights, darem dicas e ouvirem meus desabafos ao longo do curso. E também a Eduardo Stival e Thácio Breno, que, mesmo com os rumos distintos que a vida nos reservou, mantiveram o contato e continuaram oferecendo

apoio mútuo durante o curso e ao longo da vida. Por fim, gostaria de dedicar este trabalho ao meu falecido pai e à minha mãe. Ao meu pai, por ter sempre se entusiasmado com minhas conquistas e estudos e pelas longas conversas de madrugada no caminhão sobre a vida, o universo e tudo mais. À minha mãe, por ter sempre acreditado no meu potencial e me instigado e apoiado a persegui-lo. Ambos que sempre me apoiaram ao longo da minha vida e deram o seu melhor para que eu tivesse condições de perseguir meus sonhos e vontades, desde o primeiro computador até as comemorações das minhas conquistas ao longo dos anos. Essas conquistas não seriam possíveis sem o apoio e esforço deles e não tenho palavras suficientes para expressar a gratidão que sinto. Espero que minha jornada no Bacharelado em Inteligência Artificial e na vida, de algum modo, possa dar orgulho a eles.

APÊNDICE 1

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Termo de Aceite de Entrega 1

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 19 de set. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Eu ainda estou no processo de encontrar a área e escolher algo que une inteligência artificial e otimização.Para começar a busca fiz um brainstorming sobre as áreas que tenho interesse:

Problemas, Aplicações é tecnicas

Encontrei 3 artigos que me ajudaram a começar a entender melhor as sub-áreas e técnicas:

- Artificial Neural Networks Based Optimization Techniques: A Review
- A review on genetic algorithm: past, present, and future
- Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review

Para me auxiliar a descer o nível eu comecei a criar uma classificação por nível que separe as técnicas em alguns conjuntos maiores

Classificação de Niveis Otimização

Em uma das buscas eu encontrei o AI Institute for Advances in Optimization, que é um instituto de pesquisas que une os tópicos de IA e otimização. Fiz um breve sketch sobre

Estudo AI40P

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Terminar a classificação dos Níveis de otimização, incluindo os tipos de problema que cada um

resolve. Escolher uma área de fato, a partir dessa escolha fazer um levantamento do histórico e fundamentos desta para entendimento melhor da área.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Classificação Níveis de Otimização

algoritmo	Tipo	Solução	Natureza
Simulated Annealing (SA)	Físico inspirado	Meta-heurística	Estocástico
Gravitational Search Algorithm (GSA)	Físico inspirado	Meta-heurística	Estocástico
ANN	Bio inspirado	Aproximação função	Estocástico
Genetic Algorithm	Bio inspirado	Meta-heurística	Estocástico
Particle Swarm Optimization	Bio inspirado	Meta-heurística	Estocástico
Ant Colony	Bio inspirado	Meta-heurística	Estocástico
Bacterial Foraging Optimization	Bio inspirado	Meta-heurística	Estocástico
Artificial Bee Colony	Bio inspirado	Meta-heurística	Estocástico
Firefly Algorithm	Bio inspirado	Meta-heurística	Estocástico

Realizei essa classificação dos algoritmos de otimização mais populares para me ajudar a situar na área de otimização e definir uma área para seguir, esse catálogo foi importante para que eu entendesse o que existe e um pouco da história de cada um deles e suas inspirações. Na qual decidi continuar os estudos em algoritmos genéticos

Termo de Aceite de Entrega 2

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 26 de set. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Essa semana eu consegui olhar melhor os artigos levantados semana passada para me orientar, no processo de catalogação de alguns algoritmos de meta heurísticas.

Este processo está documentado em:

Classificação de Niveis Otimização

Além disso, com ajuda desse processo eu decidi pela área de pesquisa Algoritmos genéticos.

Dentro dessa área, levantei os seguintes papers para leitura e fundamentação da área:

- A hybrid genetic algorithm and particle swarm optimization for multimodal functions
- Hybrid Genetic Algorithms: A Review
- A review on genetic algorithm: past, present, and future

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- **Estudar os paper levantados**
- **Fazer um levantamento histórico da área de algoritmos genéticos**
- **Catalogar diferente abordagens de algoritmos genéticos e suas aplicações**

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Durante a semana 2 dei continuidade aos estudos sobre as áreas de otimização.

Termo de Aceite de Entrega 3

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 3 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Esta semana estudei os artigos propostos no gate passado e descrevi meus estudos em Anotações e estudos de Artigos .
- Aprofundamento nos algoritmos genéticos e catalogação destes, estudando sua origem, como funcionam, variações e aplicações Algoritmos Geneticos: Estrutura e variações
- Levantamento dos desafios de algoritmos genéticos: Algoritmos Geneticos desafios e futuro de pesquisa

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Estudar como os desafios da área estão sendo resolvidos no estado da arte e me aprofundar nessa parte.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Participei do SVR2024 durante essa semana

ACEITE DA ENTREGA:

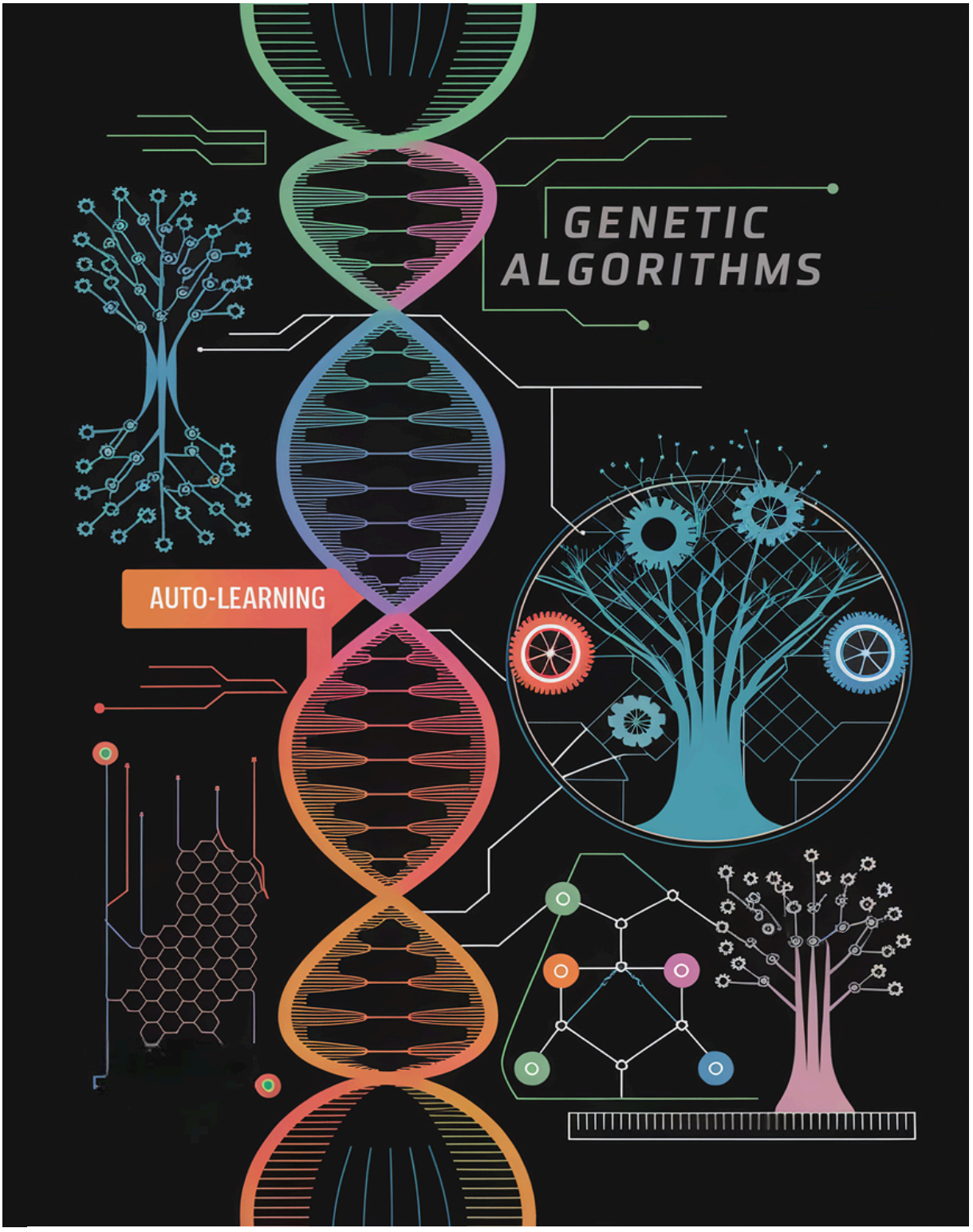
CEDRIC LUIZ DE CARVALHO: Go!

As semanas 2 e 3 foram fundamentais para minha base em algoritmos genéticos, nessa etapa eu busquei entender como eles funcionam em cada etapa da modelagem do problema a interpretação da solução encontrada, passando pelo seus controles internos. Além de pesquisar as direções futuras de pesquisa nessa área

Direções Futuras de Pesquisa

1. **Auto-organização para Taxa de Crossover e Mutação**: Investigar métodos de auto-organização para ajustar as taxas de crossover e mutação em função das características do problema.
2. **Redução da Convergência Prematura**: Desenvolver novas técnicas e abordagens para minimizar a convergência prematura em GAs.
3. **Simulação do Processo de Evolução do Sistema Imunológico Humano**: Explorar a aplicação de princípios da evolução do sistema imunológico humano para melhorar os algoritmos genéticos.
4. **Melhoria dos Esquemas de Codificação**: Focar na evolução e aprimoramento dos esquemas de codificação para lidar com diferentes tipos de problemas de forma mais eficaz.

APÊNDICE 2



MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Termo de Aceite de Entrega 4

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 10 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Esta Semana dei prosseguimento ao Gate passado estudando os problemas, desafios e aplicações de Algoritmos Genéticos. Meu processo está descrito em gate 10/10

Principais pontos do meu estudo:

- Entendimento dos problemas intrínsecos de Algoritmos Genéticos
- Levantamento de aplicações e casos do mundo real
- Levantamento de frameworks e suas capacidades

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- **Continuar o levantamento e entendimento dos problemas e limitações de Algoritmos Genéticos**
- **Testar a implementação de algum problema simples(a ser definido) nos frameworks encontrados**

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

A semana 4 foi interessante pois busquei por problemas que fossem resolvidos usando algoritmos genéticos para obter ideias de uma possível aplicação a ser implementadas , além disso também busquei por frameworks de algoritmos genéticos para poder começar a abordar os problemas de maneira mais prática

Como busca de aplicações de algoritmos genéticos em problemas do mundo real, realizei uma busca orientada por alguns descritores como

- “genetic algorithms problems”
- “genetic algorithms optimization”
- “genetic algorithms and deep learning”

Usando o buscador Google Scholar e bases como Science Direct, IEEE explorer, ACM digital library. E selecionei alguns artigos balanceando o número de citações com o ano de publicação, buscando ter um visão geral dos problemas que GAs são aplicados e alguns mais específicos também.

Genetic Algorithm- A Literature Review

- Ano 2019
- 223 Citações
- Autores: Annu Lambora, Kunal Gupta, Kriti Chopra

Network Routing Protocol

- GAs podem ser usados para otimização de rotas em um grafo encontrando as menores rotas

Image Processing

- Podem ‘tunar’ parâmetros de uma imagem de modo a melhorá-la com base em um preset definido de como deve ser a edição

Data Mining

- Pode ser usado para mineração de dados e extração de informações e controle, como em cadeias de produção, otimizando o custo da cadeia como um todo

Games

- Algoritmos genéticos podem otimizar a dificuldade de um jogo com base no padrão de jogadas dele, buscando manter um desafio para o jogador.

Comparative Approach of MRI-Based Brain Tumor Segmentation and Classification Using Genetic Algorithm

- Ano 2018
- 110 citações
- Autores: Nilesh Bhaskarrao Bahadure1,3 · Arun Kumar Ray1 · Har Pal Thethi2

Os autores utilizaram algoritmos genéticos para seleção de features e classificação de imagens MRI para detecção de tumores. Usando o algoritmo genético para selecionar quais espectros(features da imagem) eram melhores para a classificação e então para uma classificação. No paper conseguiram superar os resultados de outros modelos de classificação como ANFIS e K-NN.
Uma observação é que não testaram classificação com modelos neurais, e nem usaram GAs para seleção de features desse.

A genetic algorithm for the vehicle routing problem

- Ano 2013
- citado por 685
- Autores: CHENG Aiwen1,3 and YU Daben2

Os autores usaram algoritmos genéticos com codificação por números reais para resolver um problema de logística que consiste em uma frota de veículos realizar entregas para n usuários com restrições de tempo e capacidade de carga

Deep learning and genetic algorithm-based ensemble model for feature selection and classification of breast ultrasound images

- Ano 2024
- Citado por 0
- Autores: Mohsin Furkh Dar, Avatharam Ganivada *

Os autores propõem uma seleção de features no topo de features extraídas de uma imagem de ultrassom de peito, para detecção de câncer.

Eles usaram uma rede mobilenet para um pequeno fine tuning no dataset com algumas camadas congelada, para aprendizado de representações deste domínio, então eles usam esse modelo para extrair embedding que são de fato as features extraídas da imagem onde cada camada da rede corresponde a um embedding. Estas features são então passadas para um algoritmo genético fazer a seleção das melhores features a serem usadas por um modelo de classificação.

Usando esse método de feature extraction + feature selection(GA) eles conseguiram ótimos resultados nos datasets de teste disponíveis, demonstrando que a técnica é promissora

A multi-objective dual dynamic genetic algorithm-based approach for thermoelectric optimization of integrated urban energy systems

- Ano 2024
- Citado por -
- Autores: Yongbin Luo a, Shuo Yang a, Chenguang Niu a, Zhilei Hua b, Shiwen Zhang c

Os autores modelam o problema de distribuição de energia termoelétrica, considerando restrições de custos ambientais, sociais e operacionais e usaram um MOGA(algoritmo genético multi-objetivo) para otimizar essa distribuição considerando as dadas restrições e buscando reduzir os custos operacionais.

O método proposto consegue otimizar o dado sistema em várias métricas do contexto do problema, reduzindo custos e consumo

Busca por Frameworks inicial

Realizei um levantamento dos frameworks/bibliotecas que considerei mais interessantes e atrativos para modelagem e implementação de algoritmos genéticos. Para esta busca, levantei em fóruns da internet como Stack overflow, geeks for geeks, Kaggle os principais frameworks usados para implementação de algoritmos genéticos, selecionei estes para busca com base em sua popularidade e recomendações das comunidades. Além disso também pesquisei por alguns descritores no github como:

- genetic algorithms framework
- genetic algorithms library
- evolutionary algorithms framework
- genetic algorithms gpu implementation
- genetic algorithms tools

E fui lendo e selecionando os repositórios que se adequassem a um framework de fato. Eu instalei esses frameworks em um ambiente na minha máquina local e realizei um simples import para validar que a biblioteca foi instalada de fato.

PyGAD:

https://github.com/ahmedfgad/GeneticAlgorithmPython
1.9k Stars

462 Forks

- Biblioteca open source para modelagem de algoritmos genéticos e problemas de otimização
- Está em desenvolvimento ativo
- Dividido em módulos, cada qual com seu repositório
  - pygad.nn
  - pygad.gann
  - pygad.cnn
  - pygad.gacnn
  - pygad.kerasga
  - pygad.torchga
- Dividido em módulos, cada qual com seu repositório
- Suporte para os operadores mais conhecidos e para implementação de customizados
- Suporte com as bibliotecas keras e pytorch

DEAP

https://github.com/DEAP/deap
5.8k Stars
1.1k Forks

- Framework para prototipagem rápida e testes, tem foco em deixar claro os pontos de algoritmos e estrutura de dados.
- Muito usado em pesquisas e projetos de desenvolvimento.
- Grande suporte para várias estruturas de dados
- Suporte para otimização multiobjetiva
- Suporte para paralelização
- Benchmarks
- Suporte para checkpoints
- Customização de componentes

GeneAI

https://github.com/diogomatoschaves/geneal
62 Stars
10 Forks

- Biblioteca simples para modelagem de problemas binários e contínuos para serem resolvidos com algoritmos genéticos
- Possui customização para tipos de dados e função de fitness

- Foco na simplicidade de código balanceado customização e facilidade de implementação

Platypus

https://github.com/Project-Platypus/Platypus
566 Starts
153 Forks

- Foco em problemas de otimização multiobjetivos
- Possui algoritmos de otimização multi objetiva pré implementados bem como ferramentas de análise de soluções multi objetivas.

Geatpy2

https://github.com/geatpy-dev/geatpy
2k Stars
726 Forks

- Framework para algoritmos genéticos e evolucionários com foco em alta performance em python
- Vários algoritmos de solução para problemas multiobjetivos
- Varios esquemas de encoding pré definidos

Evox

https://github.com/EMI-Group/evox
200 Stars
35 Forks

- Framework para processamento distribuído em GPUs de algoritmos genéticos
- Suporte para vários algoritmos genéticos
- Foco em módulos para programação
- Suporte para customização de componentes

Desses frameworks encontrados quis me ater ao PyGad e ao DEAP por serem mais completos e populares, com uma comunidade maior a qual eu pudesse recorrer em caso de necessidade.

Termo de Aceite de Entrega 5

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 16 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Esta semana continuei a atividade do Stage passado de entender os problemas e limitações dos Algoritmos Genéticos, priorizando desta vez a data e a aplicação. Este processo está em: O que há de novo?

Implementação nos frameworks encontrados do problema de maximizar uma string binária, disponível em MaximizeBits.ipynb

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Continuar pesquisando self adaptation em Algoritmos Genéticos
- Modelar um Vehicle Route Optimization

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Durante a semana 5 eu busquei por paper mais novos sobre o tema de algoritmos genéticos, buscando principalmente em conferências e selecionando artigos que fossem mais atuais, datando de 2020 para frente, como forma de estar mais antenado com o estado da arte, eu tentei estruturar meu pensamento em como um cientista dessa área de 1996 se manteria atualizado até os dias de hoje em 2024.

Para isso eu procurei em conferências de computação evolutiva e algoritmos genéticos os paper que foram apresentados, os melhores papers da conferência e então consegui visualizar melhor como está o atual estado da arte.

Conferências Buscadas:

- Congress on Evolutionary Computation
- Genetic and Evolutionary Computation Conference (GECCO)

Buscando por alguns temas que achei interessantes nas Conferências:

- Design of new GA operators including representations, fitness functions, initialization, termination, parent selection, replacement strategies, recombination, and mutation
- Parameter tuning and control (including adaptation and meta-GAs)
- Evolutionary representation learning

Anotações dos artigos estudados

A Review of Crossover Methods and Problem Representation of Genetic Algorithm in
Recent Engineering Applications

Este trabalho faz um review dos operadores de crossover e representações de problema nos algoritmos genéticos, dado que a performance destes depende muito da escolha adequada de parâmetros e modelagem.

Crossovers:

- Crossover Binário
  - Random Respectful Crossover
  - Homologous Crossover
  - Count Preserving Crossover
  - One-Point Crossover
  - Two-Point Crossover
  - Uniform Crossover

    - Multi-Point Crossover
- Crossover Real
    - Simulated Binary Crossover
    - Convex Combination Crossover
    - Heuristic Normal Distribution Crossover
    - Gaussian Crossover
    - Blend Crossover Operator (BLX-α)
    - Average Crossover
- Crossover Geral
    - One-Point and Two-Point Crossover
    - Uniform Crossover
    - Multi-Point Crossover
    - Heuristic Crossover
    - Discrete Crossover

Representações:

Representação Binária

- Random Respectful
- Homologous
- Count Preserving
- One Point
- Uniform
- Multipoint

Representação Real

- Simulated Binary
- Convex Combination
- Heuristic Normal Distribution
- Gaussian
- BLX-α
- Average
- Multipoint

Promoting Two-sided Fairness in Dynamic Vehicle Routing Problem

Este trabalho apresenta o Two-sided Fairness-aware Genetic Algorithm (2FairGA). Enquanto o algoritmo genético original (GA) se concentra em gerar planos de roteamento em um ambiente estático, propomos o 2FairGA, que expande o GA para um ambiente dinâmico com múltiplos agentes, equilibrando três objetivos principais. No 2FairGA, a amostragem inicial foca em alocar provedores de serviço em diferentes localizações para otimizar a equidade de duas partes.

Para mitigar o conflito entre equidade bilateral e otimização de utilidade, evitando a dificuldade de buscar a solução ótima em uma função não convexa, o 2FairGA equilibrar os três objetivos em etapas distintas. Modificando as duas funções de aptidão no GA para considerar a equidade baseada em provedores de serviço e em clientes, incluindo um procedimento adicional de otimização local para maximizar a utilidade.

Accelerating Evolution Through Gene Masking and Distributed Search

Este trabalho propõe o método BLADE como uma nova abordagem eficaz para acelerar a evolução. A ideia central é usar uma "máscara", ou seja, um filtro, no genótipo para focar a busca em partes específicas do problema. Essas máscaras são construídas dinamicamente ao longo da busca, ajudando a concentrá-la nas partes mais relevantes, de forma similar ao funcionamento das "attention heads" nas redes neurais transformers. A junção das técnicas de mascaramento e busca distribuída forma a base do BLADE.

Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization

Inspirados pelo sucesso recente da arquitetura Set Transformer, os autores introduzem redes neurais para substituir operadores genéticos fundamentais, como a seleção e a adaptação da taxa de mutação. Essas operações são implementadas como módulos de atenção, que podem ser aplicados a problemas com dimensões e tamanhos de população variáveis. Utilizando a meta-black-box optimization (MetaBBO) para evoluir esses módulos em um conjunto representativo de problemas de otimização.
O estudo destaca o potencial de combinar parametrização flexível de algoritmos genéticos com evolução baseada em dados. Apesar da flexibilidade e interpretabilidade dos módulos de atenção, alguns mecanismos subjacentes ainda são opacos. Futuras investigações poderão explorar algoritmos genéticos mais transparentes e focar na manutenção da diversidade das soluções.

Dynamic Multi-objective Evolutionary Algorithm Based on Decomposition with Hybrid Prediction

Este algoritmo evolutivo dinâmico de múltiplos objetivos é baseado em decomposição, com uma abordagem híbrida de previsão para guiar a busca em problemas complexos.

An improved adaptive genetic algorithm

A adaptação das probabilidades de crossover e mutação no algoritmo genético é feita com base na aptidão dos indivíduos. Para o crossover, a probabilidade é aumentada para indivíduos com alta aptidão e reduzida para aqueles com baixa aptidão, favorecendo a

combinação de melhores soluções. Já a probabilidade de mutação é alta no início para promover diversidade e diminui à medida que o algoritmo se aproxima da solução ótima, evitando que ele se torne muito aleatório.

An improved class of real-coded Genetic Algorithms for numerical optimization

Este artigo apresenta uma nova classe de algoritmos genéticos e faz um review das técnicas de ce self adaptation para crossover e mutação.

- GA-DEx: Combina Algoritmos Genéticos com Evolução Diferencial para melhorar a exploração de soluções e evitar a convergência prematura.
- GA-DEx_SPS: Melhora a seleção de pais, guiando a busca para áreas mais promissoras e otimizando a eficiência
- GA-aDEx_SPS: Ajusta dinamicamente os parâmetros de cruzamento com base no desempenho anterior, evitando estagnação e promovendo uma melhor exploração.

ON THE SUITABILITY OF REPRESENTATIONS FOR QUALITY DIVERSITY OPTIMIZATION OF SHAPES

Este artigo explora as representações de encoding para algoritmos evolucionários tendo como foco a qualidade das soluções geradas a partir do encoding.

- CA (Cellular Automata): Usa autômatos celulares para gerar formas, onde regras simples locais podem produzir padrões complexos.
- CPPN (Compositional Pattern-Producing Networks): Uma rede neural que gera padrões complexos e repetitivos, modelando formas de maneira indireta.
- Direct (Direto): Cada ponto da solução é codificado diretamente, sem intermediários, o que pode dificultar a diversidade e a generalização.
- Parametric (Paramétrico): A forma é definida por um conjunto de parâmetros, permitindo ajustar diretamente características como tamanho e proporção.
- Dictionary (Dicionário): Usa uma base pré-definida de formas ou componentes, combinando elementos para gerar soluções mais complexas.

Destes artigos o que me chamou muita atenção foi o **Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization**, por combinar elementos de atenção e redes neurais para auto aprendizado de operados dos algoritmos genéticos.

Para começar me familiarizar com o desenvolvimento de soluções com algoritmos genéticos usando os frameworks previamente selecionados, implementei uma otimização de string binária que tem como objetivo maximizar todos os caracteres da string binária de 0 ou 1 para 1

Experimento prático

Após alguns testes iniciais com cada framework, decidi usar o pygad e o DEAP por hora por serem mais intuitivos de serem usados. Como problema inicial escolhi a maximização de uma string de N bits binários

```
!pip install pygad deap
```

```
import time
import pygad
import random
from deap import base, creator, tools, algorithms

NUM_BITS = 8
N_SETS = 1000
POPULATION_SIZE = 10
NUM_GENERATIONS = 50
MUTATION_PERCENT_GENES = 10  # Para o PyGAD
MUTATION_INDPB = 0.1  # Para o DEAP

# Função de fitness para o PyGAD
def fitness_func(ga_instance, solution, solution_idx):
    binary_solution = ''.join(map(lambda x: str(int(x)), solution))
    x = int(binary_solution, 2)
    return x ** 2

# Função de fitness para o DEAP
def eval_func(individual):
    x = int("".join(map(str, individual)), 2)
    return x ** 2,

# Rodando o PyGAD N_SETS vezes e medindo o tempo
pygad_total_time = 0
for _ in range(N_SETS):
    start_time = time.time()

    ga_instance = pygad.GA(num_generations=NUM_GENERATIONS,
                           num_parents_mating=2,
                           fitness_func=fitness_func,
                           sol_per_pop=POPULATION_SIZE,
                           num_genes=NUM_BITS,  # Definido como 8 bits
                           gene_space=[0, 1],
                           mutation_type="random",
```

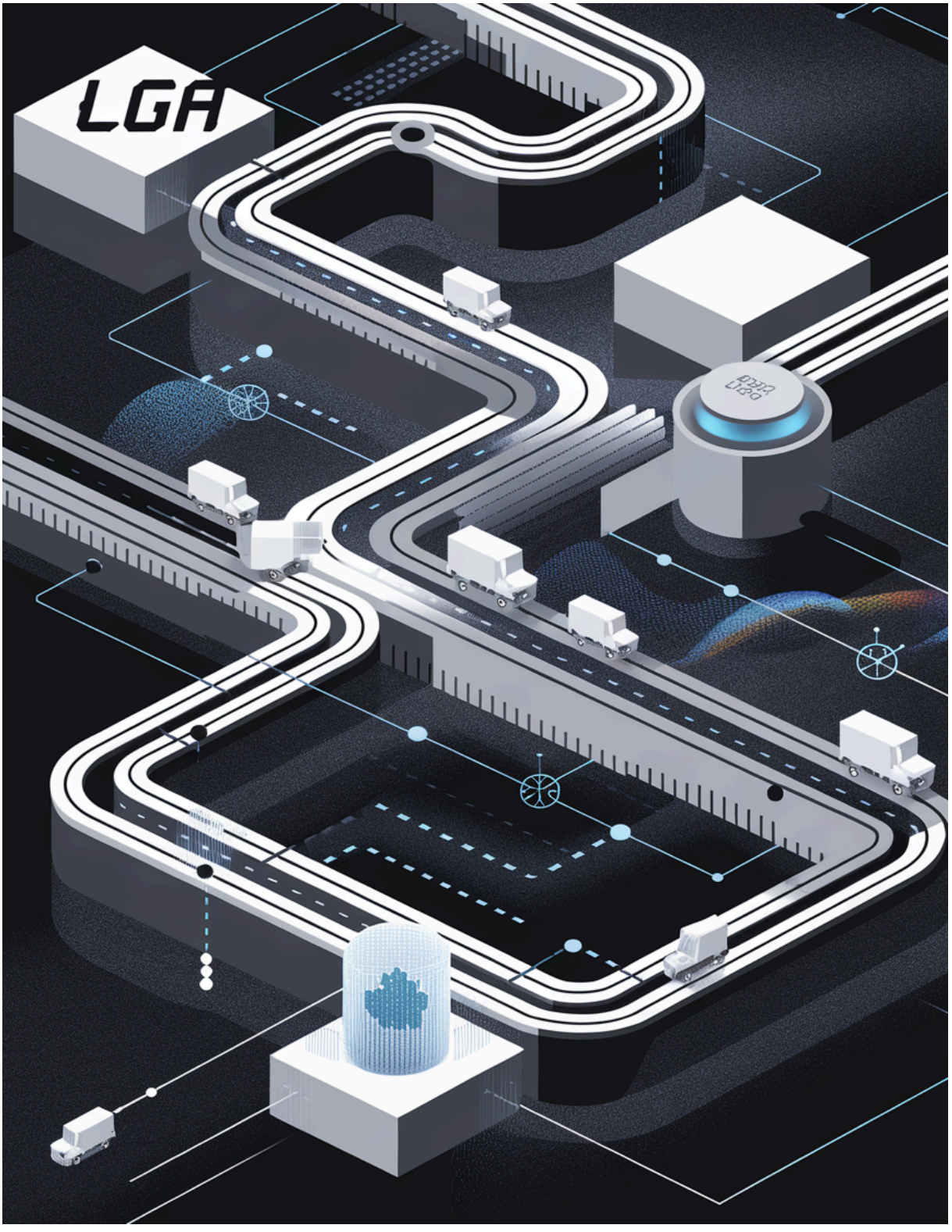
```
                                mutation_percent_genes=MUTATION_PERCENT_GENES)

    ga_instance.run()

    end_time = time.time()
    pygad_total_time += (end_time - start_time)

# Exibindo resultados do PyGAD
pygad_avg_time = pygad_total_time / N_SETS

# Configuração da estrutura para o DEAP
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual,
toolbox.attr_bool, NUM_BITS)  # 8 bits
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", eval_func)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=MUTATION_INDPB)
toolbox.register("select", tools.selTournament, tournsize=3)

# Rodando o DEAP N_SETS vezes e medindo o tempo
deap_total_time = 0
for _ in range(N_SETS):
    start_time = time.time()

    population = toolbox.population(n=POPULATION_SIZE)
    algorithms.eaSimple(population, toolbox, cxpb=0.5, mutpb=0.2,
ngen=NUM_GENERATIONS, verbose=False)

    end_time = time.time()
    deap_total_time += (end_time - start_time)

# Exibindo resultados do DEAP
deap_avg_time = deap_total_time / N_SETS

print(f"PyGAD - Tempo total: {pygad_total_time:.4f} segundos, Tempo médio
por execução: {pygad_avg_time:.4f} segundos")
print(f"DEAP - Tempo total: {deap_total_time:.4f} segundos, Tempo médio por
execução: {deap_avg_time:.4f} segundos")
```

```
PyGAD - Tempo total: 21.1599 segundos, Tempo médio por execução: 0.0212
segundos
```

DEAP - Tempo total: 12.7918 segundos, Tempo médio por execução: 0.0128 segundos

Para um mesmo problema, mantidos os parâmetros e rodando por 100 vezes, o deap conseguiu executar quase que na metade do tempo total e médio do pygad, indicando que pode ser mais otimizado

APÊNDICE 3



Termo de Aceite de Entrega 6

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 31 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

O processo do gate dessa Semana: Processo Gate 311024. Me aprofundei em técnicas de self-adaptation e LGA, lendo os seguintes artigos :

- Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization
- A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem
- Improved Self-Learning Genetic Algorithm for Solving Flexible Job Shop Scheduling
- Automated Metaheuristic Algorithm Design with Autoregressive Learning
- Meta Learning Black-Box Population-Based Optimizers

Fichamento dos pontos importantes em : Self Adaptation, Learning - Genetic Algorithm
Estudei a classe de problemas de Vehicle Route Optimization Vehicle routing problem.

Modelei um breve problema e solução usando GA padrão: simple_vrp.ipynb.

Teste do framework do artigo: Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization: 01_classic_benchmark-EVOJAX.ipynb.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Continuar estudando mais sobre self adaptation e LGA
- Testar os frameworks dos artigos em modelagem de VRP

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Essa semana foi uma virada de chave no processo da minha residência, com base no estudado nas semanas anteriores eu defini minha área de pesquisa em algoritmos genéticos de auto aprendizado e auto adaptativos

Escolhi estudar e me aprofundar nessas técnicas pois gostei dessa interseção de IA + GA, onde a IA é usada para melhorar um processo interno do GA e melhorar a busca por soluções.
Durante essa busca esbarrei em alguns temas de pesquisa do estado da arte para o auto aprendizado e adaptação em algoritmos genéticos como:

- Aprendizado por atenção
- Auto adaptação por reinforcement learning
- Auto regressão para problemas de otimização contínua em GA

Levantamento dás técnicas de auto aprendizado

Auto-Adaptação nos Algoritmos Genéticos

- Em busca de melhorias, foram analisados operadores de auto-adaptação, como a mutação Gaussiana. Esse operador ajusta a distribuição de mutação automaticamente, proporcionando maior eficácia em ambientes de otimização contínuos e complexos. Além disso, a programação genética foi estudada como um método para ajustar dinamicamente os operadores, integrando um sistema de recompensas e punições baseado no desempenho de cada operador, aplicando uma abordagem de aprendizado por reforço dentro do GA

Meta-Aprendizagem e Modelos Autorregressivos para Otimização de Algoritmos

- Um aprofundamento na meta-aprendizagem revelou o uso de frameworks como o **ALDes** (Autoregressive Learning-based Designer), que utiliza redes auto regressivas para gerar algoritmos de metaheurística de forma sequencial. Este framework

permite que algoritmos sejam otimizados tanto em suas estruturas quanto em seus operadores, promovendo uma adaptação contínua e eficiente

Meta-Black-Box Optimization (MetaBBO):

- A MetaBBO otimiza os parâmetros dos operadores de GAs por meio de um processo de meta-aprendizado, onde o algoritmo aprende a configurar as operações de busca com base no feedback recebido. Cada iteração avalia e ajusta os operadores, promovendo um ajuste contínuo sem conhecimento prévio do espaço de busca

Interseção entre IA e Algoritmos Genéticos para Auto-Aprendizado

- Foram analisadas abordagens híbridas que combinam redes neurais e GAs para otimizar os parâmetros dos modelos de machine learning. Esse tipo de integração possibilita que GAs explorem amplos espaços de busca enquanto as redes neurais oferecem uma avaliação detalhada das soluções

Além disso fiz um levantamento de paper sobre os temas de interesse na área de algoritmos genéticos de auto aprendizado

Levantamento de artigos sobre self adaptation, learning genetic algorithms, Black Box Optimization

Título: Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization

Autores: Robert Tjarko Lange, Tom Schaul, Yutian Chen, Chris Lu, Tom Zahavy, Valentin Dalibard, Sebastian Flennerhag

Ano: 2023

Introdução
Os algoritmos genéticos tradicionais dependem de operadores projetados manualmente, o que pode resultar em adaptações inadequadas para certos problemas. A ideia central é que esses operadores possam ser aprendidos automaticamente, aumentando a generalização e aplicabilidade dos GAs

- Usando parâmetros baseados em atenção(dos transformers), a taxa de mutação, adaptação e seleção podem ser aprendidas em vez de definidas

- MetaBBO é usado para evoluir os parâmetros do LGA, ajustando-os em uma série de problemas de otimização de caixa preta, o que permite que o LGA desenvolva operadores eficazes e adaptáveis.

Black-Box Optimization

Em problemas de otimização de caixa preta, a função f(x) a ser minimizada não tem uma forma explícita ou derivada acessível. Logo, métodos tradicionais de otimização de gradiente não são aplicáveis, tornando os GAs uma solução viável.

Set Operations via Dot-Product Self-Attention

Para modelar operações em conjunto, o artigo usa Self-Attention baseada em produto escalar (Scaled Dot-Product Attention), que preserva a invariância à permutação dos elementos do conjunto, algo desejável para GAs. Isso permite que o LGA manipule conjuntos de soluções candidatas de forma independente da ordem, aprimorando a eficácia dos operadores genéticos.

Meta-Black-Box Optimization (MetaBBO)

O MetaBBO é uma técnica de meta-otimização onde operadores genéticos são ajustados automaticamente através de várias iterações em tarefas de otimização. Neste processo, o LGA ajusta seus operadores com base no desempenho em tarefas de treinamento, permitindo uma evolução contínua e generalização dos parâmetros.

Treinamento LGA

O treinamento do LGA foi feito usando funções de benchmarking de otimização com diferentes objetivos(separabilidade, multimodalidade), desenvolvendo operadores genéticos adaptados e robustos.

Os experimentos mostraram que operadores aprendidos podem ser usados como substitutos em GAs convencionais, melhorando o desempenho dos algoritmos. Isso indica que os operadores do LGA são robustos e podem servir como módulos transferíveis em outros sistemas de otimização

Titulo: A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem

Autores: Ronghua Chen, Bo Yang, Shi Li, Shilong Wang

Ano: 2020

Introdução

Motivação: Métodos tradicionais de algoritmos genéticos não conseguem ajustar parâmetros de forma dinâmica e eficiente, o que reduz a qualidade das soluções. Assim, é proposta uma abordagem de aprendizado para ajustar automaticamente os parâmetros do GA (taxas de cruzamento e mutação) usando Reinforcement Learning (RL). Em um cenário de Job-Shop-Scheduling, que é um problema NP-hard e altamente complexo.

- **Algoritmo Genético de Auto-Aprendizado (SLGA)**: Um GA que ajusta automaticamente parâmetros importantes, como taxas de cruzamento e mutação com base em RL, especificamente os algoritmos SARSA e Q-Learning.
- **Integração de SARSA e Q-Learning:** Para aproveitar as vantagens dos dois algoritmos, o SLGA utiliza SARSA nas fases iniciais para uma convergência mais rápida e depois muda para Q-Learning, que oferece um ajuste mais preciso na fase final.
- **Comparação Empírica:** O SLGA é comparado com métodos de referência (GA, GA-SARSA, GA-Q) e outros algoritmos do estado da arte em agendamentos, mostrando melhor desempenho em termos de eficiência e qualidade da solução.

Algoritmo Genético e Aprendizado por Reforço

- Genetic Algorithm (GA): Soluções representadas por cromossomos são evoluídas por operadores de cruzamento e mutação para otimizar o makespan. O SLGA adiciona um mecanismo de RL para adaptar esses operadores durante a execução.
- Reinforcement Learning (RL): Utiliza SARSA e Q-Learning para ajustar em tempo real com base na qualidade das soluções. O SARSA é usado inicialmente para aprendizado mais rápido, enquanto o Q-Learning, que utiliza o valor máximo futuro esperado, é ativado posteriormente para otimização fina.

Experimentos

Q-Learning e o SARSA são aplicados ao algoritmo genético para ajustar automaticamente as taxas de cruzamento e mutação durante o processo evolutivo.

- SARSA: Usado nas fases iniciais do GA para ajuste rápido e dinâmico. O SARSA escolhe as ações com base na política atual e permite que o agente (GA) ajuste as taxas de mutação e cruzamento para melhorar a aptidão da população conforme ela evolui. Esse método é escolhido no começo por sua rápida convergência, ajudando o GA a estabilizar rapidamente.
- Q-Learning: Aplicado nas fases finais do GA para refinar os ajustes das taxas de cruzamento e mutação. Diferente do SARSA, o Q-Learning atualiza a política baseada no valor máximo esperado da próxima ação, permitindo um ajuste mais preciso. Esse refinamento ajuda o GA a alcançar o melhor desempenho possível ao final da execução.

Os autores testaram o SLGA em dois conjuntos de dados de benchmark (Kacem e Brandimarte) com instâncias de tamanhos variáveis. Os resultados foram comparados com GA-SARSA, GA-Q e outros algoritmos de referência.

O SLGA superou consistentemente outros métodos em qualidade e tempo de convergência, apresentando RPD (Relative Percentage Deviation) inferior em 8 de 10 instâncias grandes.

Titulo: Improved Self-Learning Genetic Algorithm for Solving Flexible Job Shop Scheduling

Autores: Ming Jiang, Haihan Yu , Jiaqing Chen

Ano: 2023

Introdução

O artigo propõe um **Algoritmo Genético de Auto-Aprendizado Melhorado** para resolver o *Flexible Job Shop Scheduling Problem*, FJSP. Este algoritmo busca superar limitações dos GAs tradicionais, especialmente em relação à adaptação dinâmica de operadores, melhorando a eficiência e qualidade das soluções.

Auto-Aprendizado em Mutação:

Quatro operadores de mutação foram desenvolvidos com base nas características do FJSP: **máquina gananciosa (machine greedy)**, **seleção aleatória de máquina**, **mutação de troca multi-ponto no processo** e **mutação de dois pontos no processo**.

Esses operadores foram ajustados dinamicamente usando pesos que variam de acordo com o sucesso de cada operador durante o processo iterativo. Se uma mutação gera uma melhoria, o peso do operador correspondente é aumentado; caso contrário, é reduzido. Isso

permite que o algoritmo favoreça operadores que melhoram a solução ao longo do tempo-Aprendizado Variacional:

Para controlar as taxas de mutação e cruzamento, o algoritmo emprega um processo de ajuste baseado em uma estratégia de aprendizado em RL (Reinforcement Learning). Ele aprende a ajustar essas taxas conforme o algoritmo evolui, usando experiências passadas e previsões futuras para otimizar a exploração e intensificação em diferentes estágios .

Resultados

Algoritmo de auto-aprendizado apresentou desempenho superior em benchmarks, com maior velocidade de convergência e melhor capacidade de busca local em comparação com GAs tradicionais e outras variantes de referência.

Titulo: Automated Metaheuristic Algorithm Design with Autoregressive Learning

Autores: Qi Zhao, Tengfei Liu, Bai Yan, Qiqi Duan, Jian Yang, and Yuhui Shi

Ano: 2024

Introdução

O artigo apresenta o ALDes (Autoregressive Learning-based Designer), um método inovador para automatizar o design de algoritmos metaheurísticos usando redes de aprendizado autorregressivo. O ALDes gera algoritmos sequencialmente, permitindo a criação de soluções adaptáveis e eficientes que se ajustam a diferentes problemas de otimização. Em vez de definir operadores de algoritmos genéticos (GAs) manualmente, o ALDes os cria e ajusta automaticamente com base em experiências passadas, otimizando e personalizando o funcionamento dos GAs de forma dinâmica, sem intervenção humana

Modelo Autorregressivo:

Aprendizado Auto Regressivo: ALDes emprega um modelo autorregressivo para gerar algoritmos como sequências de tokens. Cada token representa um componente de um algoritmo, como operadores de seleção, mutação ou hiperparâmetros.

Modelo de Geração Sequencial: A geração sequencial permite que o ALDes crie algoritmos com diferentes estruturas e comprimentos, adequando-os a demandas específicas de problemas de otimização.

Representação de Sequência e Componentes do Algoritmo:

Tokens Representativos: Os algoritmos são representados por tokens que descrevem os componentes do design, como operadores de cruzamento e mutação, parâmetros de taxa de mutação e crossover, bem como ponteiros que guiam a execução.

Treinamento com Experiência Acumulada:

- Aprendizado Incremental: O ALDes armazena experiências de algoritmos anteriores para aprender padrões e características úteis. Essa experiência acumulada permite que o designer identifique combinações de operadores que foram bem-sucedidas em problemas similares e adapte esses conhecimentos a novos contextos.
- Armazenamento em Neurônios da Rede: Informações sobre a eficácia dos algoritmos são armazenadas em neurônios da rede autorregressiva, permitindo que o ALDes refine suas escolhas de design.

Experimentos::

- ALDes foi testado em um conjunto de 25 problemas de otimização, abrangendo uma ampla gama de complexidade e tipos de restrições. Este conjunto inclui problemas de agendamento, alocação de recursos, e roteamento.
- Os algoritmos projetados pelo ALDes foram comparados com metaheurísticas tradicionais projetadas manualmente, incluindo algoritmos de otimização com operadores de mutação e crossover fixos.
- A qualidade da solução e o tempo de convergência foram usados como métricas para avaliar a eficácia dos algoritmos gerados pelo ALDes em comparação com os projetados manualmente.

Resultados

Em 24 dos 25 problemas testados, os algoritmos gerados pelo ALDes superaram os algoritmos de referência em termos de qualidade da solução e eficiência de convergência. Esse resultado indica a capacidade do ALDes de aprender a construir algoritmos que se ajustam bem aos problemas específicos.

TItulo: Meta Learning Black-Box Population-Based Optimizers

Autores: Hugo Siqueira Gomes, Benjamin Léger, Christian Gagné

Ano:2021

O artigo propõe um framework de meta-aprendizagem que utiliza um processo de decisão parcialmente observável (*Partially Observable Markov Decision Process*, POMDP) para otimizar algoritmos populacionais em cenários de otimização de caixa-preta. Esta abordagem foca em fazer a auto adaptação de operadores genéticos conhecidos como mutação,cruzamento e seleção.

LTO-POMDP Framework:

O artigo propõe um framework de meta-aprendizagem, **Learning-to-Optimize POMDP (LTO-POMDP)**, que modela a busca como uma sequência de decisões em um POMDP. A ideia é que o algoritmo aprenda a adaptar seu comportamento de busca com base em observações sobre o desempenho de cada solução, semelhante a como um GA adapta sua população ao longo das gerações.

LTO-POMDP

(*Learning-to-Optimize Partially Observable Markov Decision Process*) é um framework de meta-aprendizagem para otimização de algoritmos baseados em população. Ele modela o processo de otimização como uma sequência de decisões observáveis de forma parcial, permitindo que o algoritmo aprenda a escolher ações de busca de maneira informada, mesmo sem conhecer completamente o espaço de solução.

Modelo de POMDP:

No LTO-POMDP, o problema de busca é tratado como um *POMDP*, onde o estado verdadeiro do espaço de solução não é completamente conhecido. A cada etapa, o algoritmo observa uma representação parcial do estado, que inclui o desempenho das soluções candidatas.

Meta-Aprendizagem com Decisões Sequenciais:

O framework utiliza aprendizado por reforço para decidir quais ações de busca (como variações nos parâmetros) aplicar em cada etapa. As ações são escolhidas com base em uma política aprendida, que é ajustada a partir das recompensas recebidas ao longo de várias iterações de busca.

Avaliação de Desempenho:

O LTO-POMDP ajusta-se dinamicamente ao problema, adaptando sua política de otimização ao longo do tempo. Isso significa que ele acumula conhecimento sobre o problema específico e aplica estratégias que maximizam o desempenho em problemas anteriores.

Comparação e Eficiência:

Nos experimentos, o método mostrou-se competitivo em relação a métodos populares como CMA-ES, alcançando resultados superiores na eficiência de busca sem necessidade de ajustes manuais para cada novo problema, o que evidencia a capacidade do sistema de meta-aprendizagem de generalizar para uma ampla classe de problemas

Fichamento sobre o Problema de Roteamento de Veículos (Vehicle Routing Problem - VRP)

Para além do estudo teórico nesse campo, de forma paralela eu estudei problemas de roteamento de veículos através de paper que definem essa classe de problemas , com ênfase no seguinte artigo do qual usei como base para meus conhecimentos

Selected Genetic Algorithms for Vehicle Routing Problem Solving

Autores: Joanna Ochelska-Mierzejewska, Aneta Poniszewska-Mara ´nda, Witold Mara ´nda

Ano: 2021

Definição do Problema

O Problema de Roteamento de Veículos (VRP) é um problema clássico de otimização combinatória, amplamente estudado na área de pesquisa operacional. O objetivo principal do VRP é determinar a maneira mais eficiente de roteirizar um ou mais veículos que partem de um ponto de depósito para atender a um conjunto de clientes, cada um com uma demanda específica, minimizando custos operacionais ou distâncias percorridas. O problema é de natureza NP-difícil, o que significa que não existem algoritmos polinomiais

conhecidos para resolvê-lo em todas as suas variantes, tornando-o um campo ativo de pesquisa.

Características Principais

- Objetivo: O foco é minimizar o custo total das rotas, que pode incluir fatores como a distância total percorrida, tempo de viagem, ou custos associados ao uso de veículos.
- Veículos: Um ou mais veículos são utilizados para atender os clientes. Cada veículo tem uma capacidade máxima que não pode ser excedida, o que limita a quantidade de demanda que pode ser atendida em uma única rota.
- Clientes: Os clientes têm demandas específicas que precisam ser atendidas. Cada cliente é associado a uma localização geográfica, a qual deve ser visitada pelos veículos.
- Localização: Os pontos de depósito e dos clientes são representados em um espaço geográfico, frequentemente modelado como um grafo onde os nós representam locais e as arestas representam as distâncias ou tempos de viagem entre eles.

Restrições: O VRP pode incluir várias restrições, como:

Limitação da capacidade dos veículos, garantindo que a soma das demandas atendidas em uma rota não exceda a capacidade do veículo.
Janelas de tempo, que definem os intervalos durante os quais os clientes devem ser atendidos.
Restrições de serviço, que podem incluir horários específicos em que os veículos devem estar em determinados locais.

Tipos de VRP

O VRP possui diversas variantes, cada uma com suas particularidades. As mais comuns incluem:

- Capacitated Vehicle Routing Problem (CVRP): Esta variante considera a capacidade limitada dos veículos, exigindo que as rotas sejam planejadas de forma a não exceder essa capacidade.
- Vehicle Routing Problem with Time Windows (VRPTW): Incorpora janelas de tempo, onde cada cliente deve ser atendido dentro de um intervalo específico, aumentando a complexidade do problema.

- Stochastic Vehicle Routing Problem (SVRP): Considera incertezas nas demandas dos clientes ou nos tempos de viagem, introduzindo variáveis estocásticas que devem ser geridas.
- Periodic Vehicle Routing Problem (PVRP): Trata do planejamento de rotas para um período de vários dias, permitindo que os veículos atendam a clientes em diferentes dias, o que pode otimizar as operações ao longo de um horizonte temporal.
- Multi-Depot Vehicle Routing Problem (MDVRP): Neste caso, os veículos podem partir de múltiplos depósitos, o que adiciona um nível adicional de complexidade ao planejamento das rotas.

Esquemas de Codificação para Algoritmos Genéticos

- Codificação por permutação: Cada veículo é associado a uma sequência de clientes que ele atende, em uma lista única onde separadores indicam a mudança de veículo
    - [veículo 1 [cliente1, cliente2], veículo 2 [cliente1 , cliente2], … vn [cn]]

- Codificação por matrizes: Aqui, as rotas são representadas em uma matriz, onde cada linha corresponde a um veículo e cada coluna a um cliente. Esta estrutura é especialmente útil em problemas com múltiplos veículos.
    - [[Veículo 1: [Cliente1, Cliente2, Cliente 3]
      Veículo 2: [Cliente 4, Cliente 5]]]

- Codificação por listas: Cada veículo tem sua própria lista de clientes, permitindo que as restrições de rota, como capacidades e janelas de tempo
      Veículo 1: [Cliente1, Cliente2]
      Veículo 2: [Cliente 3, Cliente 4]

- Codificação por grafo de rotas: Nesta abordagem, cada rota é representada por um sub grafo onde os nós correspondem aos clientes e as arestas representam as distâncias ou tempos de viagem
      Veículo 1: Grafo (Cliente1 - Cliente2 - Cliente 3)
      Veículo 2: Grafo (Cliente 4 - Cliente 5)

Selection:

A etapa de seleção na solução de um VRP por algoritmos genéticos, não possui um requisito formal específico para a solução do problema. As técnicas de seleção padrão visando o indivíduo com melhor fitness podem ser aplicadas.

Crossover:

É preciso um cuidado com os crossovers utilizados em relação ao encoding escolhido, dado a natureza das representações onde a ordem importa na solução, crossovers incorretos podem gerar duplicidade nos pontos da rota gerando soluções inválidas para o problema.

- Order Crossover: Escolhe dois pontos de corte nos pais, copia a subsequência entre esses pontos de um pai para o filho, e preenche os pontos restantes em ordem a partir do segundo pai. Evita duplicações ao garantir que os genes entre os cortes sejam fixos e que o restante siga a ordem do outro pai, mantendo a integridade da rota.
- Partially mapped Crossover: Seleciona dois pontos de corte, copiando parte de um pai. Em seguida, os elementos restantes do segundo pai são mapeados para o filho nas posições não ocupadas. Eficaz para manter uma boa quantidade de ordem e posições do segundo pai, equilibrando entre diversidade e manutenção de rota.
- Edge recombination crossover: Foca na preservação das arestas entre os clientes, promovendo a herança de bordas comuns aos dois pais, criando um vetor de vizinhança. Excelente para instâncias onde a relação entre os pontos na rota é crítica, como em problemas com restrições fortes, pois prioriza a continuidade das rotas
- Cycle crossover: Copia um ciclo de genes de um pai para o filho, garantindo que cada posição em um ciclo é herdada de um dos pais. Começa com um gene do primeiro pai, então a posição é determinada pelo segundo pai, alternando os pais até completar o ciclo. O CX garante que cada gene apareça exatamente uma vez, sendo útil para problemas onde a preservação exata da ordem relativa é desejada.

Mutation

A etapa de seleção não possui também requisitos específicos para a solução do problema de VRP podendo ser uma implementação padrão de outros algoritmos genéticos, adequando a mutation de acordo com o tipo de dado do encoding

Exemplos clássicos de VRP

- **Augerat Set A**: Proposto em 1995, com 27 instâncias geradas aleatoriamente em uma área de 100x100 unidades, cada ponto tem uma demanda média de 15, com

10% das coordenadas tendo a demanda triplicada. A capacidade dos veículos é limitada a 100.

- **Augerat Set B**: Também de 1995, possui 23 instâncias, onde os pontos estão distribuídos em clusters. A demanda é aleatória entre 1 e 30, com 10% dos pontos tendo a demanda triplicada, com um número de clusters maior que o número de veículos.
- **Augerat Set P**: Um conjunto de instâncias que modifica exemplos já conhecidos na literatura, também descrito por Augerat.
- **Christofides e Eilon Set E**: Publicado em 1969, é composto por 13 instâncias distribuídas uniformemente no espaço.
- **Fisher Set F**: Conjunto de três problemas reais usados por Fisher, incluindo dois dias de entregas de uma empresa de supermercados e dados de entrega de itens para postos de gasolina.
- **TSPLIB Ulysses Instances**: Exemplos clássicos do problema do caixeiro-viajante adaptados para VRP, representando a jornada mítica de Ulisses.

Evaluation

Dado os exemplos clássicos uma solução pode ser avaliada da seguinte forma: Para todas as instâncias testadas, os valores ideais eram conhecidos. Os resultados de uma solução incluem o valor ideal, a melhor solução encontrada e a diferença percentual dessa solução em relação ao ótimo.

Escolha e teste do framework para um VRP com algoritmos genéticos de auto aprendizado

Dados os estudos dessa semana decidi prosseguir com o algoritmo genético de auto aprendizado (LGA) proposto no paper *Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization.* O qual já possuía um repositório oficial com os códigos necessários para testes com essa técnica. Para validar a funcionalidade do repositório eu testei o código de introdução deste

01 - Simple ES Benchmark Function

[Last Update: March 2022][Colab](https://colab.research.google.com/github/RobertTLange/evosax/blob/main/examples/01_classic_benchmark.ipynb)

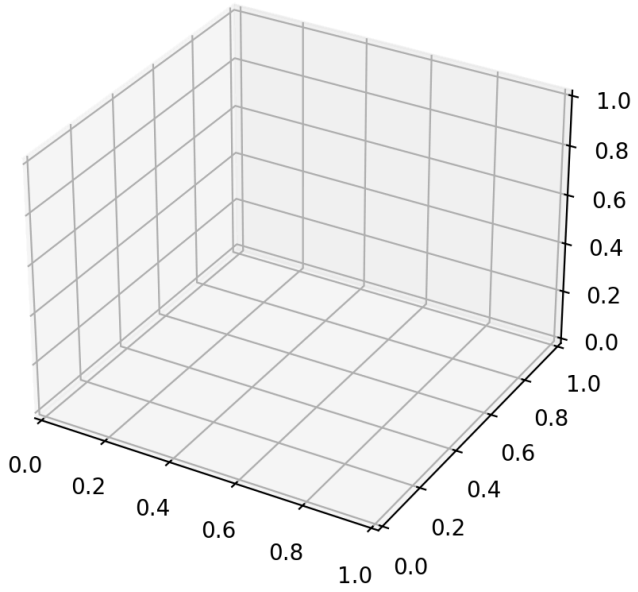
```
%matplotlib inline
%load_ext autoreload
%autoreload 2
%config InlineBackend.figure_format = 'retina'
```

2D Rosenbrock with CMA-ES

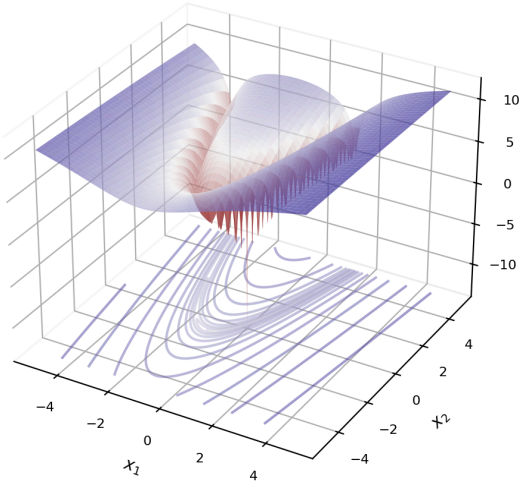
evosax implements a set of different classic benchmark functions. These include multi-dimensional versions of `quadratic`, `rosenbrock`, `ackley`, `griewank`, `rastrigin`, `schwefel`, `himmelblau`, `six-hump`. In the following we focus on the 2D Rosenbrock case, but feel free to play around with the others.

```
import jax
import jax.numpy as jnp
from evosax import CMA_ES
from evosax.problems import BBOBFitness

# Instantiate the problem evaluator
rosenbrock = BBOBFitness("RosenbrockOriginal", num_dims=2, seed_id=2)
rosenbrock.visualize(plot_log_fn=True)
```

```
An NVIDIA GPU may be present on this machine, but a CUDA-enabled jaxlib is
not installed. Falling back to cpu.
```



Log RosenbrockOriginal Function




```
# Instantiate the search strategy
rng = jax.random.PRNGKey(0)
strategy = CMA_ES(popsize=20, num_dims=2, elite_ratio=0.5)
es_params = strategy.default_params.replace(init_min=-2, init_max=2)

state = strategy.initialize(rng, es_params)
```

```
# Run ask-eval-tell loop - NOTE: By default minimization
for t in range(50):
    rng, rng_gen, rng_eval = jax.random.split(rng, 3)

    x, state = strategy.ask(rng_gen, state, es_params)
    fitness = rosenbrock.rollout(rng_eval, x)
    state = strategy.tell(x, fitness, state, es_params)

    if (t + 1) % 10 == 0:
        print("CMA-ES - # Gen: {}|Fitness: {:.5f}|Params: {}".format(
            t+1, state.best_fitness, state.best_member))
```

```
CMA-ES - # Gen: 10|Fitness: 0.11797|Params: [-0.24922164 -0.45996723]
CMA-ES - # Gen: 20|Fitness: 0.06408|Params: [-0.25254992 -0.44303328]
CMA-ES - # Gen: 30|Fitness: 0.00020|Params: [-0.00756088 -0.01385561]
CMA-ES - # Gen: 40|Fitness: 0.00000|Params: [-0.00087913 -0.00171672]
CMA-ES - # Gen: 50|Fitness: 0.00000|Params: [4.831762e-06 9.959638e-06]
```

2D Rosenbrock with Other ES

```
from evosax import Strategies
rng = jax.random.PRNGKey(0)

for s_name in ["SimpleES", "SimpleGA", "PSO", "DE", "Sep_CMA_ES",
               "Full_iAMaLGaM", "Indep_iAMaLGaM", "MA_ES", "LM_MA_ES",
               "RmES", "GLD", "SimAnneal", "GESMR_GA", "SAMR_GA"]:
    strategy = Strategies[s_name](popsize=20, num_dims=2)
    es_params = strategy.default_params
    es_params = es_params.replace(init_min=-2, init_max=2)
    state = strategy.initialize(rng, es_params)

    for t in range(30):
        rng, rng_gen, rng_eval = jax.random.split(rng, 3)
        x, state = strategy.ask(rng_gen, state, es_params)
        fitness = rosenbrock.rollout(rng_eval, x)
        state = strategy.tell(x, fitness, state, es_params)

        if (t + 1) % 5 == 0:
            print("{} - # Gen: {}|Fitness: {:.2f}|Params: {}".format(
                s_name, t+1, state.best_fitness, state.best_member))
    print(20*"=")
```

```
SimpleES - # Gen: 5|Fitness: 2.41|Params: [0.57494265 1.3363407 ]
SimpleES - # Gen: 10|Fitness: 0.05|Params: [-0.02394061 -0.06951423]
SimpleES - # Gen: 15|Fitness: 0.02|Params: [0.05201919 0.11855019]
```

```
SimpleES - # Gen: 20|Fitness: 0.02|Params: [0.05201919 0.11855019]
SimpleES - # Gen: 25|Fitness: 0.02|Params: [0.05201919 0.11855019]
SimpleES - # Gen: 30|Fitness: 0.02|Params: [0.05201919 0.11855019]
====================
SimpleGA - # Gen: 5|Fitness: 0.05|Params: [-0.23124486 -0.40957353]
SimpleGA - # Gen: 10|Fitness: 0.03|Params: [-0.13031453 -0.23322381]
SimpleGA - # Gen: 15|Fitness: 0.02|Params: [-0.11532616 -0.20849138]
SimpleGA - # Gen: 20|Fitness: 0.00|Params: [-0.00291445 -0.00076399]
SimpleGA - # Gen: 25|Fitness: 0.00|Params: [-0.00291445 -0.00076399]
SimpleGA - # Gen: 30|Fitness: 0.00|Params: [0.04794273 0.09704389]
====================
PSO - # Gen: 5|Fitness: 0.32|Params: [-0.01428866  0.02790421]
PSO - # Gen: 10|Fitness: 0.19|Params: [-0.32952124 -0.52207595]
PSO - # Gen: 15|Fitness: 0.12|Params: [-0.33950216 -0.56108046]
PSO - # Gen: 20|Fitness: 0.09|Params: [-0.30322647 -0.51587033]
PSO - # Gen: 25|Fitness: 0.06|Params: [-0.2369234 -0.4188014]
PSO - # Gen: 30|Fitness: 0.06|Params: [-0.2369234 -0.4188014]
====================
DE - # Gen: 5|Fitness: 0.37|Params: [-0.60689706 -0.8382895 ]
DE - # Gen: 10|Fitness: 0.03|Params: [-0.15622565 -0.29684156]
DE - # Gen: 15|Fitness: 0.00|Params: [-0.01819727 -0.03231793]
DE - # Gen: 20|Fitness: 0.00|Params: [0.00036369 0.00136669]
DE - # Gen: 25|Fitness: 0.00|Params: [0.00036369 0.00136669]
DE - # Gen: 30|Fitness: 0.00|Params: [0.0002749  0.00099869]
====================
Sep_CMA_ES - # Gen: 5|Fitness: 6.83|Params: [-2.4531186  1.021449 ]
Sep_CMA_ES - # Gen: 10|Fitness: 6.83|Params: [-2.4531186  1.021449 ]
Sep_CMA_ES - # Gen: 15|Fitness: 6.83|Params: [-2.4531186  1.021449 ]
Sep_CMA_ES - # Gen: 20|Fitness: 6.83|Params: [-2.4531186  1.021449 ]
Sep_CMA_ES - # Gen: 25|Fitness: 6.83|Params: [-2.4531186  1.021449 ]
Sep_CMA_ES - # Gen: 30|Fitness: 6.83|Params: [-2.4531186  1.021449 ]
====================
Full_iAMaLGaM - # Gen: 5|Fitness: 0.01|Params: [-0.10261798 -0.19658658]
Full_iAMaLGaM - # Gen: 10|Fitness: 0.00|Params: [-0.0065736  -0.01341228]
Full_iAMaLGaM - # Gen: 15|Fitness: 0.00|Params: [9.9909972e-05
1.8824372e-04]
Full_iAMaLGaM - # Gen: 20|Fitness: 0.00|Params: [2.0618314e-05
4.2214029e-05]
Full_iAMaLGaM - # Gen: 25|Fitness: 0.00|Params: [1.8419464e-06
3.4559771e-06]
Full_iAMaLGaM - # Gen: 30|Fitness: 0.00|Params: [-2.2521112e-07
-5.5985942e-07]
====================
Indep_iAMaLGaM - # Gen: 5|Fitness: 0.05|Params: [0.14939058 0.3042835 ]
Indep_iAMaLGaM - # Gen: 10|Fitness: 0.02|Params: [0.13289815 0.28522336]
```

```
Indep_iAMaLGaM - # Gen: 15|Fitness: 0.02|Params: [0.13289815 0.28522336]
Indep_iAMaLGaM - # Gen: 20|Fitness: 0.02|Params: [0.13289815 0.28522336]
Indep_iAMaLGaM - # Gen: 25|Fitness: 0.02|Params: [0.13289815 0.28522336]
Indep_iAMaLGaM - # Gen: 30|Fitness: 0.02|Params: [0.13289815 0.28522336]
====================
MA_ES - # Gen: 5|Fitness: 0.33|Params: [-0.5359075 -0.7642154]
MA_ES - # Gen: 10|Fitness: 0.33|Params: [-0.5359075 -0.7642154]
MA_ES - # Gen: 15|Fitness: 0.33|Params: [-0.5359075 -0.7642154]
MA_ES - # Gen: 20|Fitness: 0.33|Params: [-0.5359075 -0.7642154]
MA_ES - # Gen: 25|Fitness: 0.33|Params: [-0.5359075 -0.7642154]
MA_ES - # Gen: 30|Fitness: 0.33|Params: [-0.5359075 -0.7642154]
====================
LM_MA_ES - # Gen: 5|Fitness: 7.78|Params: [-2.7078676  1.8501627]
LM_MA_ES - # Gen: 10|Fitness: 7.45|Params: [-2.7272193  1.9920657]
LM_MA_ES - # Gen: 15|Fitness: 7.42|Params: [-2.7236936  1.9769124]
LM_MA_ES - # Gen: 20|Fitness: 7.42|Params: [-2.7237751  1.9702089]
LM_MA_ES - # Gen: 25|Fitness: 7.41|Params: [-2.721651   1.9702324]
LM_MA_ES - # Gen: 30|Fitness: 7.41|Params: [-2.7209196  1.9683607]
====================
RmES - # Gen: 5|Fitness: 0.37|Params: [-0.5904469 -0.8481015]
RmES - # Gen: 10|Fitness: 0.37|Params: [-0.5904469 -0.8481015]
RmES - # Gen: 15|Fitness: 0.11|Params: [-0.33058518 -0.54896724]
RmES - # Gen: 20|Fitness: 0.11|Params: [-0.33058518 -0.54896724]
RmES - # Gen: 25|Fitness: 0.11|Params: [-0.33058518 -0.54896724]
RmES - # Gen: 30|Fitness: 0.09|Params: [-0.28109023 -0.473656  ]
====================
GLD - # Gen: 5|Fitness: 0.01|Params: [-0.10305867 -0.19583313]
GLD - # Gen: 10|Fitness: 0.01|Params: [-0.07785731 -0.14456296]
GLD - # Gen: 15|Fitness: 0.00|Params: [-0.03990304 -0.08063482]
GLD - # Gen: 20|Fitness: 0.00|Params: [-0.03032007 -0.05793908]
GLD - # Gen: 25|Fitness: 0.00|Params: [-0.03103311 -0.06153297]
GLD - # Gen: 30|Fitness: 0.00|Params: [-0.01392278 -0.02847507]
====================
SimAnneal - # Gen: 5|Fitness: 114.55|Params: [-1.7434927   0.60876817]
SimAnneal - # Gen: 10|Fitness: 29.11|Params: [-1.990768    0.48308632]
SimAnneal - # Gen: 15|Fitness: 4.59|Params: [-2.1422088   0.30636737]
SimAnneal - # Gen: 20|Fitness: 4.36|Params: [-2.081329   0.1867277]
SimAnneal - # Gen: 25|Fitness: 4.25|Params: [-2.0612168   0.12882927]
SimAnneal - # Gen: 30|Fitness: 3.98|Params: [-1.9781699  -0.01724105]
====================
GESMR_GA - # Gen: 5|Fitness: 9.00|Params: [-1.181883  -1.2426326]
GESMR_GA - # Gen: 10|Fitness: 1.43|Params: [-1.1773776  -0.99034756]
GESMR_GA - # Gen: 15|Fitness: 0.43|Params: [-0.57461786 -0.78734726]
GESMR_GA - # Gen: 20|Fitness: 0.27|Params: [-0.5132652 -0.7691258]
GESMR_GA - # Gen: 25|Fitness: 0.24|Params: [-0.4847008  -0.73747873]
```

```
GESMR_GA - # Gen: 30|Fitness: 0.21|Params: [-0.46073768 -0.7048115 ]
====================
SAMR_GA - # Gen: 5|Fitness: 0.59|Params: [0.68800676 1.8831477 ]
SAMR_GA - # Gen: 10|Fitness: 0.57|Params: [0.64244306 1.7375212 ]
SAMR_GA - # Gen: 15|Fitness: 0.57|Params: [0.64244306 1.7375212 ]
SAMR_GA - # Gen: 20|Fitness: 0.57|Params: [0.64244306 1.7375212 ]
SAMR_GA - # Gen: 25|Fitness: 0.52|Params: [0.69069225 1.8796452 ]
SAMR_GA - # Gen: 30|Fitness: 0.52|Params: [0.69069225 1.8796452 ]
====================
```

Try out one of the many evosax algorithms!

```
Strategies.keys()
```

APÊNDICE 4

Termo de Aceite de Entrega 7

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 7 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Anteriormente: Estudo de artigos sobre Self Adaptation em GA e Learn Genetic algorithms. Estudo do VRP(vehicle routing problem), teste de frameworks de artigos.

Processo da semana em: Processo Gate 71124. Estudei a classe de problemas de análise combinatória, documentado em: Problemas-VRP

Reli os artigos sobre Self Adaptation e LGA. Selecionei o framework EvoSax, documentei a escolha em FrameWork - Evosax

Small VRP modelado:

- Número de Clientes: 10, distribuídos em uma área de 100x100 unidades.
- Demanda dos Clientes: Entre 1 e 20 unidades, definida aleatoriamente.
- Capacidade do Veículo: 100 unidades.
- Objetivo: Minimizar a distância total percorrida pelo veículo para atender todos os clientes sem ultrapassar sua capacidade de carga.
- Algoritmo: Algoritmo genético com 100 gerações e uma população de 50 soluções (rotas).

Implementei duas soluções para o VRP modelado usando o framework selecionado, disponíveis em:

- vrp-simple-ga-evosax.ipynb
- vrp_lga_evosax.ipynb

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Continuar a implementação do VRP com o EvoSax
- Modelagem do VRP massivo de acordo com a literatura e implementação deste

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Em análise!

Estudo do framework

Durante essa semana, aprofundei meus estudos sobre o artigo Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization, revisitando o paper e explorando o framework EvoSax, que é a base oficial desse trabalho
Eu também li o artigo do framework apesar de estar desatualizado em relação ao LGA

evosax : JAX-Based Evolution Strategies

autor: Robert Tjarko Lange
ano: 2022

O artigo apresenta o evosax, uma biblioteca de algoritmos de otimização evolutiva implementada em JAX. Estratégias evolutivas (ES) são técnicas de otimização de caixa preta, muitas vezes aplicadas a problemas complexos de aprendizado de máquina e sistemas dinâmicos, especialmente quando o cálculo de gradientes é inviável ou impraticável. O evosax foi desenvolvido para explorar os recursos de GPUs e TPUs, permitindo otimizações em larga escala com execução acelerada.

A biblioteca implementa 30 algoritmos de ES, incluindo abordagens populares como CMA-ES (Covariance Matrix Adaptation Evolution Strategy) e OpenAI-ES, com suporte para transformações em batch (jax.vmap) e paralelização em múltiplos dispositivos (jax.pmap).

As principais contribuições do evosax incluem:

- Aceleração por Hardware: Aproveita a compilação de JAX para executar estratégias evolutivas em GPUs e TPUs, o que é essencial para otimização em larga escala.

- Flexibilidade e Modularidade: Permite configurar diferentes estratégias evolutivas com ajustes finos, usando uma interface de API padrão, baseada nas etapas ask-evaluate-tell.
- Suporte Extensivo a Algoritmos: Com 30 algoritmos de ES, o evosax cobre desde estratégias clássicas a métodos mais modernos, facilitando a experimentação com diferentes abordagens.

O framework EvoSax funciona atŕaves de um paradigma de estruturação de problema comum a muitos algoritmos evolutivos:

- Ask: Gera uma população de soluções candidatas através dos operadores genéticos
- Evaluate: Avalia essa população através da função fitness
- Tell: Seleciona as melhores soluções candidatas e as passa para a próxima geração

Esse ciclo se repete, permitindo ao algoritmo evoluir a população de soluções para encontrar uma solução cada vez melhor

Abordagens usando o EvoSAx

Explorei o EvoSAX e suas aplicações para abordar o problema de roteamento de veículos (VRP), um desafio NP-hard que demanda um alto poder computacional e otimização de estratégias.

Esse processo tem me desafiado a compreender os fundamentos das estratégias evolutivas e adaptar o framework para lidar com as restrições específicas do VRP. Uma das maiores dificuldades que enfrentei foi lidar com instabilidades numéricas, como valores NaN nos cromossomos, que interrompiam a convergência do algoritmo. Para mitigar isso, passei a explorar ajustes nos parâmetros de mutação e métodos de controle de valores extremos, além de alternativas para contornar as limitações do EvoSax com grandes populações e alta dimensionalidade.

Problemas de VRP

A complexidade do VRP também me levou a aprofundar o entendimento dos problemas NP-difíceis, que exigem não só algoritmos eficazes, mas também um ajuste fino de parâmetros para obter resultados consistentes.

O problema de roteamento de veículos (VRP) é um problema de otimização combinatória da classe np-hard, onde o objetivo é encontrar a rota ideal para que veículos atendam locais específicos minimizando custos (como distância ou tempo), respeitando restrições como capacidade e janelas de tempo. A complexidade de resolução cresce exponencialmente com o aumento de locais e restrições.

O que é um problema NP-Hard?

> De acordo com Garey e Johnson (1979), problemas NP-difíceis são aqueles que são pelo menos tão difíceis quanto os problemas na classe NP. Isso significa que, embora seja possível verificar uma solução candidata em tempo polinomial, encontrar essa solução de maneira eficiente não é garantido, a menos que P=NPP = NPP=NP. Esses problemas incluem uma ampla variedade de desafios em otimização e combinatória, sendo caracterizados pela dificuldade de resolução exata em um tempo razoável.

Citação:

- Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Problemas similares ao VRP com características de otimização combinatória:

- Alocação de Tarefas: Atribuir tarefas a agentes para minimizar o custo total, garantindo que cada tarefa seja atribuída a um único agente.
- Mochila (Knapsack): Selecionar itens com pesos e valores variados para maximizar o valor total dentro de uma capacidade fixa, comum em otimização de recursos.
- Corte de Estoque: Cortar peças de um material contínuo para minimizar o desperdício, aplicável em indústrias como papel e vidro.
- Empacotamento de Caixas (Bin Packing): Colocar itens de tamanhos variados em contêineres de capacidade fixa, minimizando o número de contêineres.
- Sequenciamento de Tarefas: Ordenar tarefas em máquinas para minimizar o tempo total ou outras métricas de processamento.
- Cobertura de Conjuntos: Selecionar subconjuntos para cobrir todos os elementos de um conjunto maior, minimizando o número de subconjuntos escolhidos.
- Partição de Conjuntos: Dividir um conjunto em subconjuntos disjuntos que atendam a condições específicas.

VRP GA - EvoSax

A partir desses estudos passei pra parte prática, implementando inicialmente um vrp no EvoSax para validar seu uso

```
!pip install evosax
```

```
import jax
import jax.numpy as jnp
import numpy as np
from evosax import SimpleGA
import tqdm
import matplotlib.pyplot as plt

def euclidean_distance(pos1, pos2):
    return jnp.sqrt(jnp.sum((pos1 - pos2) ** 2))

def vrp_fitness_function(route, client_positions, client_demands,
vehicle_capacity):
    route = jnp.argsort(route)
    total_distance = 0.0
    current_load = 0.0

    def loop_body(i, loop_carry):
        total_distance, current_load = loop_carry
        client_a = client_positions[route[i]]
        client_b = client_positions[route[i + 1]]
        distance = euclidean_distance(client_a, client_b)
        total_distance += distance
        current_load += client_demands[route[i]]

        penalty = jax.lax.cond(current_load > vehicle_capacity,
                               lambda: 100.0, #penalidado acaso exceda a
carga
                               lambda: 0.0)
        total_distance += penalty
        return total_distance, current_load

    total_distance, _ = jax.lax.fori_loop(0, client_positions.shape[0] - 1,
loop_body, (total_distance, current_load))

    total_distance += euclidean_distance(client_positions[route[-1]],
client_positions[route[0]])

    return total_distance

# Parâmetros do VRP
num_generations = 100
pop_size = 50
num_clients = 10
```

```
vehicle_capacity = 100
client_demands = jnp.array(np.random.randint(1, 20, num_clients))
client_positions = jnp.array(np.random.rand(num_clients, 2) * 100)
```

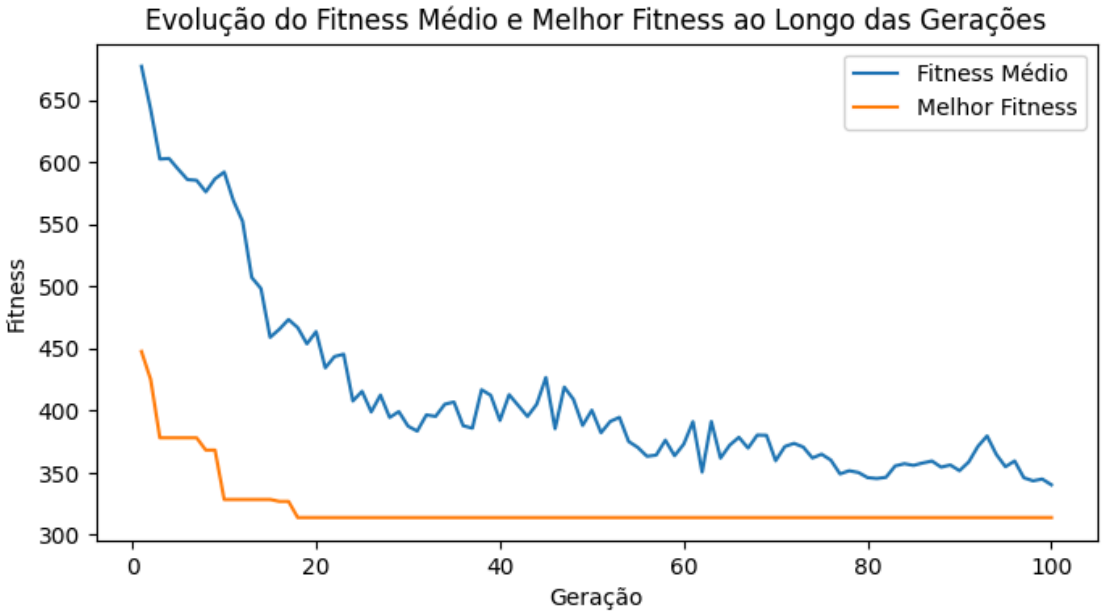
```
An NVIDIA GPU may be present on this machine, but a CUDA-enabled jaxlib is
not installed. Falling back to cpu.
```

```
sigma_init = 0.1
sigma_decay = 1.0
sigma_limit = 0.1
cross_over_rate = 0.5

rng = jax.random.PRNGKey(0)
num_dims = num_clients
strategy = SimpleGA(
    popsize=pop_size,
    num_dims=num_dims,
    sigma_init=sigma_init,
    sigma_decay=sigma_decay,
    sigma_limit=sigma_limit
)
es_params = strategy.default_params
# Atualizar o parâmetro de cruzamento diretamente no EvoParams
es_params =
strategy.params_strategy.replace(cross_over_rate=cross_over_rate)
es_state = strategy.initialize(rng, es_params)

# Loop principal do GA
best_fitness_history = []
average_fitness_history = []
for generation in tqdm.tqdm(range(num_generations)):
    rng, rng_gen = jax.random.split(rng)
    candidates, es_state = strategy.ask(rng_gen, es_state, es_params)
    fitness_scores = jax.vmap(lambda route: vrp_fitness_function(route,
client_positions, client_demands, vehicle_capacity))(candidates)
    es_state = strategy.tell(candidates, fitness_scores, es_state,
es_params)
    average_fitness = fitness_scores.mean()
    best_fitness = es_state.best_fitness
    average_fitness_history.append(average_fitness)
    best_fitness_history.append(best_fitness)


# Melhor rota final
best_solution = es_state.best_member
best_fitness = es_state.best_fitness
```

```
print("Melhor rota:", jnp.argsort(best_solution))
print("Melhor fitness (distância total):", best_fitness)
```

```
100%|██████████| 100/100 [00:08<00:00, 11.79it/s]

Melhor rota: [7 2 5 6 9 1 4 0 8 3]
Melhor fitness (distância total): 313.95752
```

```
# Plotar o fitness médio e o melhor fitness de cada geração
plt.figure(figsize=(8, 4))
plt.plot(range(1, num_generations + 1), average_fitness_history,
label='Fitness Médio')
plt.plot(range(1, num_generations + 1), best_fitness_history, label='Melhor
Fitness')
plt.xlabel('Geração')
plt.ylabel('Fitness')
plt.title('Evolução do Fitness Médio e Melhor Fitness ao Longo das
Gerações')
plt.legend()
plt.show()
```



VRP LGA - EvoSax

```
!pip install evosax
```

```
import jax
import jax.numpy as jnp
import numpy as np
from evosax import SimpleGA
import tqdm
import matplotlib.pyplot as plt
from evosax import  LGA

VRP
def euclidean_distance(pos1, pos2):
    return jnp.sqrt(jnp.sum((pos1 - pos2) ** 2))

def vrp_fitness_function(route, client_positions, client_demands,
vehicle_capacity):
    route = jnp.argsort(route)  # Organiza a sequência dos clientes
    total_distance = 0.0
    current_load = 0.0

    # Loop para calcular a distância percorrida e verificar a capacidade
    def loop_body(i, loop_carry):
        total_distance, current_load = loop_carry
        client_a = client_positions[route[i]]
        client_b = client_positions[route[i + 1]]
        distance = euclidean_distance(client_a, client_b)
        total_distance += distance
        current_load += client_demands[route[i]]

        # Aplica penalidade se a capacidade for excedida
        penalty = jax.lax.cond(current_load > vehicle_capacity,
                               lambda: 100.0,
                               lambda: 0.0)
        total_distance += penalty
        return total_distance, current_load

    # Executa o loop com fori_loop do JAX
    total_distance, _ = jax.lax.fori_loop(0, client_positions.shape[0] - 1,
loop_body, (total_distance, current_load))

    # Adiciona a distância de volta ao depósito
    total_distance += euclidean_distance(client_positions[route[-1]],
client_positions[route[0]])
```

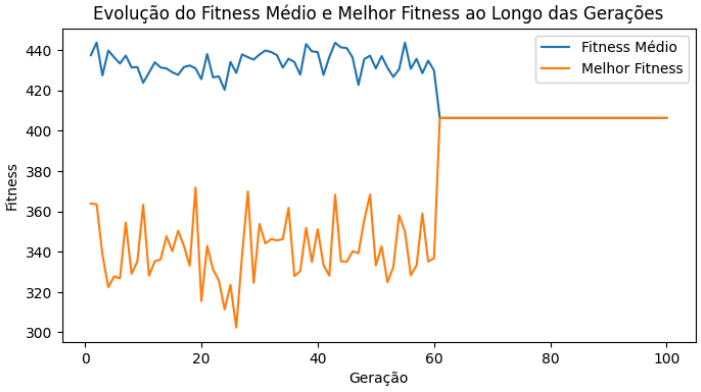
```
    return total_distance

# Parâmetros do VRP
num_generations = 100
pop_size = 50
num_clients = 10
vehicle_capacity = 100
client_demands = jnp.array(np.random.randint(1, 20, num_clients))
client_positions = jnp.array(np.random.rand(num_clients, 2) * 100)

LGA
num_dims = 10
rng = jax.random.PRNGKey(0)
pop_size=50
lga_strategy = LGA(popsize=pop_size, num_dims=num_dims)
lga_param = lga_strategy.default_params
lga_param = lga_strategy.params_strategy.replace(clip_min=-10.0)

lga_state = lga_strategy.initialize(rng, lga_param)


average_fitness_history = []
best_fitness_history = []
num_generations = 100

for generation in tqdm.tqdm(range(num_generations)):
    rng, rng_init, rng_ask, rng_eval = jax.random.split(rng, 4)
    candidates, lga_state = lga_strategy.ask(rng_ask, lga_state, lga_param)
    fitness_scores = jax.vmap(lambda route: vrp_fitness_function(route,
client_positions, client_demands, vehicle_capacity))(candidates)

    # Atualizar `lga_state` com os novos fitness scores
    lga_state = lga_strategy.tell(candidates, fitness_scores, lga_state,
lga_param)

    # Forçar atualização de `best_fitness` com o menor fitness da geração
    best_fitness = fitness_scores.min()

    # Guardar média e melhor fitness para plotagem
    average_fitness = fitness_scores.mean()
    average_fitness_history.append(average_fitness)
    best_fitness_history.append(best_fitness)
    if generation % 10 == 0:
```

```
      print(f"Geração {generation + 1}, Melhor fitness:
{fitness_scores.min()}")
      print(f'Amostra de solução candidata')
      print(candidates[0])



# Melhor rota final
best_solution = lga_state.best_member
best_fitness = lga_state.best_fitness
print("Melhor rota:", jnp.argsort(best_solution))
print("Melhor fitness (distância total):", best_fitness)


Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl

  2%|▌          | 2/100 [00:02<01:56,  1.19s/it]

Geração 1, Melhor fitness: 363.9293212890625
Amostra de solução candidata
[ 27.743494  -10.          1.4947082  -8.509466   -1.6227409  -9.502746
  -9.277853  -10.         23.217005   16.64156  ]

 11%|███▍       | 11/100 [00:05<00:28,  3.11it/s]

Geração 11, Melhor fitness: 328.149658203125
Amostra de solução candidata
[ 7.83834562e+05  2.21157900e+06  1.21990766e+05  4.44596836e+04
 -1.00000000e+01  1.06654600e+06  3.06340250e+05  9.72541938e+05
 -1.00000000e+01 -1.00000000e+01]

 21%|██▏        | 21/100 [00:07<00:20,  3.88it/s]

Geração 21, Melhor fitness: 342.92291259765625
Amostra de solução candidata
[ 1.6864456e+10  1.6518208e+09 -1.0000000e+01 -1.0000000e+01
  2.7744156e+10  3.9777124e+10  1.9931507e+10 -1.0000000e+01
  5.2761252e+10 -1.0000000e+01]

 31%|███▏       | 31/100 [00:09<00:17,  3.92it/s]

Geração 31, Melhor fitness: 344.11627197265625
Amostra de solução candidata
[-1.0000000e+01  4.6534204e+15  3.2757200e+16 -1.0000000e+01
 -1.0000000e+01 -1.0000000e+01 -1.0000000e+01  5.7214535e+16
  5.0810992e+16  2.0837229e+16]
```

```
 42%|████      | 42/100 [00:11<00:09,  6.23it/s]

Geração 41, Melhor fitness: 333.439697265625
Amostra de solução candidata
[ 1.0481242e+25  1.3946601e+25  9.7699156e+24 -1.0000000e+01
  1.3045469e+25  4.3673999e+25 -1.0000000e+01 -1.0000000e+01
 -1.0000000e+01  1.5628999e+25]

 52%|█████     | 52/100 [00:13<00:07,  6.07it/s]

Geração 51, Melhor fitness: 342.6730041503906
Amostra de solução candidata
[ 3.8494645e+31 -1.0000000e+01  4.7050108e+30  3.6517835e+30
  3.7705429e+31 -1.0000000e+01 -1.0000000e+01 -1.0000000e+01
 -1.0000000e+01  2.1659045e+31]

 62%|██████    | 62/100 [00:15<00:06,  5.68it/s]

Geração 61, Melhor fitness: 406.3946533203125
Amostra de solução candidata
[nan nan nan nan nan nan nan nan nan nan]

 72%|███████   | 72/100 [00:17<00:06,  4.20it/s]

Geração 71, Melhor fitness: 406.3946533203125
Amostra de solução candidata
[nan nan nan nan nan nan nan nan nan nan]

 82%|████████  | 82/100 [00:19<00:03,  5.69it/s]

Geração 81, Melhor fitness: 406.3946533203125
Amostra de solução candidata
[nan nan nan nan nan nan nan nan nan nan]

 91%|█████████ | 91/100 [00:21<00:02,  4.20it/s]

Geração 91, Melhor fitness: 406.3946533203125
Amostra de solução candidata
[nan nan nan nan nan nan nan nan nan nan]

100%|██████████| 100/100 [00:23<00:00,  4.21it/s]

Melhor rota: [1 3 4 5 6 7 2 0 9 8]
Melhor fitness (distância total): 302.40997
```

```
# Plotar o fitness médio e o melhor fitness de cada geração
plt.figure(figsize=(8, 4))
```

```
plt.plot(range(1, num_generations + 1), average_fitness_history,
label='Fitness Médio')
plt.plot(range(1, num_generations + 1), best_fitness_history, label='Melhor
Fitness')
plt.xlabel('Geração')
plt.ylabel('Fitness')
plt.title('Evolução do Fitness Médio e Melhor Fitness ao Longo das
Gerações')
plt.legend()
plt.show()
```



Atualmente, estou enfrentando dificuldades na implementação do VRP com o LGA no EvoSax, pois os valores dos cromossomos estão tendendo ao mínimo NaN da biblioteca, o que interrompe o processo de otimização. Estou explorando soluções para contornar esse problema.

Termo de Aceite de Entrega 8

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 14 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Tema: Algoritmos genéticos de self adaptation e auto aprendizado via atenção, aplicados em um problema VRP

Anteriormente: Definição de um Vehicle Routing Problem (VRP) simplificado, uso do Learn Genetic Algorithm (LGA) para solução.

Curiosidade: Para 25 clientes a quantidade de rotas possível é de $1{,}551121004 \times 10^{25}$ ou

15.511.210.043.330.985.984.000.000

O processo do Stage dessa Semana: Processo gate 141124
Ao longo da Semana, revisei a implementação do LGA para o problema de VRP e decidi validá-lo em uma versão mais simplificada do problema, utilizando força bruta para comparação de resultados. Realizei experimentos com o LGA e o GA, mantendo o máximo possível de parâmetros iguais entre os dois algoritmos.

Experimentos:

- 9 clientes
- 8 clientes
- 7 clientes
- 6 clientes

Planilha de resultados para comparação: Resultados VRP

Códigos em vrp_lga_x_ga.ipynb

Código em desenvolvimento: custom_lga.ipynb

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Continuar a implementação customizada do EvoSax
- Usa outro algoritmo de self adaptation e learn genetic algorithm

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Eu ia implementar um vrp massivo, mas dado o debug do código, eu precisei implementar um vrp pequeno

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Investigação de problemas

Durante a semana, concentrei meus esforços em investigar as razões pelas quais o **LGA (Learned Genetic Algorithm)** não estava funcionando como esperado. Um dos principais problemas observados foi o surgimento de valores **NaN** durante a execução do algoritmo, especialmente após a 100ª geração. Embora o algoritmo continuasse sendo executado, esses valores invalidavam os resultados, impedindo a convergência adequada e comprometendo a eficácia da otimização.

Sem saber exatamente qual era a causa raiz do problema, decidi revisar a modelagem do problema de roteamento de veículos (**VRP**) que estava sendo usada como base. Identifiquei que a função de fitness, em sua forma atual, ainda não considerava a capacidade dos veículos como uma restrição efetiva. Isso simplificava o problema, mas também tornava mais difícil diagnosticar erros, pois estava longe de representar um **VRP realista**.

Para facilitar o diagnóstico e compreender melhor o comportamento do algoritmo, optei por reduzir o escopo do problema. Diminuí o número de clientes e removi restrições desnecessárias, como a capacidade dos veículos, para criar um cenário mais básico e controlado. Além disso, implementei uma solução de força bruta para calcular todas as possíveis rotas e identificar a melhor solução manualmente. Isso me permitiu comparar

diretamente os resultados do LGA com a solução ótima e verificar se o algoritmo estava, de fato, convergindo para um resultado válido.

Rank de soluções encontradas: LGA e GA

Eu também realizei testes comparando as implementações do LGA e GA nesse set reduzido de problemas comparando o resultado com base no rank que eles conseguiam

Algoritmo	Número clientes	Número população	Número gerações	melhor fitness possível(VRP)	fitness Algoritmo	rank solução
LGA	9	10	500	14,57649	15,22677	71/362880
GA	9	10	500	14,57649	14,57649	19/362880
LGA	8	10	500	11,81256	11,81256	33/40320
GA	8	10	500	11,81256	11,81256	33/40320
LGA	7	10	500	11,43398	11,43398	19/5040
GA	7	10	500	11,43398	11,43398	19/5040
LGA	6	10	500	11,81256	11,81256	3/720
GA	6	10	500	11,81256	11,81256	3/720

Custom LGA e experimentação de parâmetros

Paralelamente, investiguei maneiras de lidar com os **NaN** gerados durante a execução do algoritmo. Ajustei parâmetros críticos, como os limites de mutação (**clip_min** e **clip_max**) e a inicialização dos valores de entrada. Também trabalhei para garantir que as rotas geradas sempre começassem no depósito, uma restrição fundamental do problema que poderia estar sendo violada durante as etapas de mutação e cruzamento.

Esses esforços não apenas ajudaram a entender o comportamento do LGA em cenários mais simples, mas também lançaram as bases para lidar com problemas mais complexos e realistas no futuro. Apesar dos desafios, o processo proporcionou um aprendizado significativo sobre algoritmos genéticos, a modelagem de problemas de otimização, e o uso prático de frameworks como o **EvoSax**.

```
!pip install evosax
```

```
from evosax import  LGA

import jax
import jax.numpy as jnp
import chex
from flax import struct
import tqdm

from evosax.learned_eo.lga_tools import (
    SelectionAttention,
    SamplingAttention,
    MutationAttention,
    tanh_age,
)

# Parâmetros do VRP
num_clients = 9
pop_size = 10
num_generations = 100
client_positions = jnp.array(jax.random.randint(jax.random.PRNGKey(0),
(num_clients, 2), 0, 5))
```

```
An NVIDIA GPU may be present on this machine, but a CUDA-enabled jaxlib is
not installed. Falling back to cpu.
```

```
@struct.dataclass
class EvoState:
    rng: chex.PRNGKey
    mean: chex.Array
    archive_age: chex.Array  # Parents: 'Age' counter
    archive_x: chex.Array  # Parents: Solution vectors
    archive_f: chex.Array  # Parents: Fitness scores
    archive_sigma: chex.Array  # Parents: Mutation strengths
    sigma_C: chex.Array  # Children: Mutation strengths
    best_member: chex.Array
    best_fitness: float = jnp.finfo(jnp.float32).max
    gen_counter: int = 0


@struct.dataclass
class EvoParams:
    net_params: chex.ArrayTree
    cross_over_rate: float = 0.0
    sigma_init: float = 1.0
    init_min: float = -5.0
```

```
    init_max: float = 5.0
    clip_min: float = -jnp.finfo(jnp.float32).max
    clip_max: float = jnp.finfo(jnp.float32).max

class CustomLGA(LGA):
    def ask_strategy(self, rng, state, params):
        """Sobrescreve o método `ask_strategy` para garantir que o depósito
esteja no início"""
        rng, rng_idx = jax.random.split(rng)
        elite_ids = jnp.arange(self.elite_popsize)

        # Amostragem de candidatos
        age_features = tanh_age(state.archive_age, state.gen_counter +
1e-10)
        F_E = self.fitness_features.apply(
            state.archive_x, state.archive_f, state.best_fitness
        )
        F_E = jnp.concatenate([F_E, age_features.reshape(-1, 1)], axis=1)
        p = self.sampling_layer.apply(params.net_params["sampling"], F_E)
        idx = jax.random.choice(rng_idx, elite_ids, (self.popsize,), p=p)

        # Selecionar candidatos
        X_C = state.archive_x[idx]
        f_C = state.archive_f[idx]
        sigma_C = state.archive_sigma[idx]

        # Adaptação da taxa de mutação
        F_C_tilde = self.fitness_features.apply(X_C, f_C,
state.best_fitness)
        sigma_C = self.mutation_layer.apply(
            params.net_params["mutation"], sigma_C, F_C_tilde
        )

        # Mutação Gaussiana escalada
        epsilon = jax.random.normal(rng, (self.popsize, self.num_dims))
        x = X_C + sigma_C * epsilon

        # Garantir que o depósito esteja no início
        x = jax.vmap(lambda route: jnp.concatenate([[0], route[1:]]))(x)

        return x, state.replace(sigma_C=sigma_C)

    def initialize_strategy(self,rng, params):
        """`initialize` the evolution strategy."""
        init_x = jax.random.uniform(
```

```
            rng,
            (self.elite_popsize, self.num_dims),
            minval=params.init_min,
            maxval=params.init_max,
        )

        # Garantir que o depósito esteja no início
        init_x = jax.vmap(lambda route: jnp.concatenate([jnp.array([0]),
route[1:]]))(init_x)

        init_sigma = jnp.ones((self.elite_popsize, 1)) * params.sigma_init

        return EvoState(
            rng=rng,
            mean=init_x[0],
            archive_x=init_x,
            archive_f=jnp.zeros(self.elite_popsize) + 5000000.0,
            archive_sigma=init_sigma,
            archive_age=jnp.zeros(self.elite_popsize),
            sigma_C=jnp.zeros((self.popsize, 1)),
            best_member=init_x[0],
        )

# Substitua LGA pela CustomLGA
lga_strategy = CustomLGA(
    popsize=pop_size,
    num_dims=num_clients
)

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl

def vrp_fitness_function_lga(route, client_positions):
    """Calcula o fitness (distância total) de uma rota no VRP."""
    route = jnp.argsort(route)  # Ordena a sequência dos clientes

    total_distance = 0.0
    current_load = 0.0

    for i in range(len(route) - 1):
        client_a = client_positions[route[i]]
        client_b = client_positions[route[i + 1]]
        distance = euclidean_distance(client_a, client_b)
        total_distance += distance
    depot_distance = euclidean_distance(client_positions[route[-1]],
client_positions[route[0]])
    total_distance += depot_distance
```

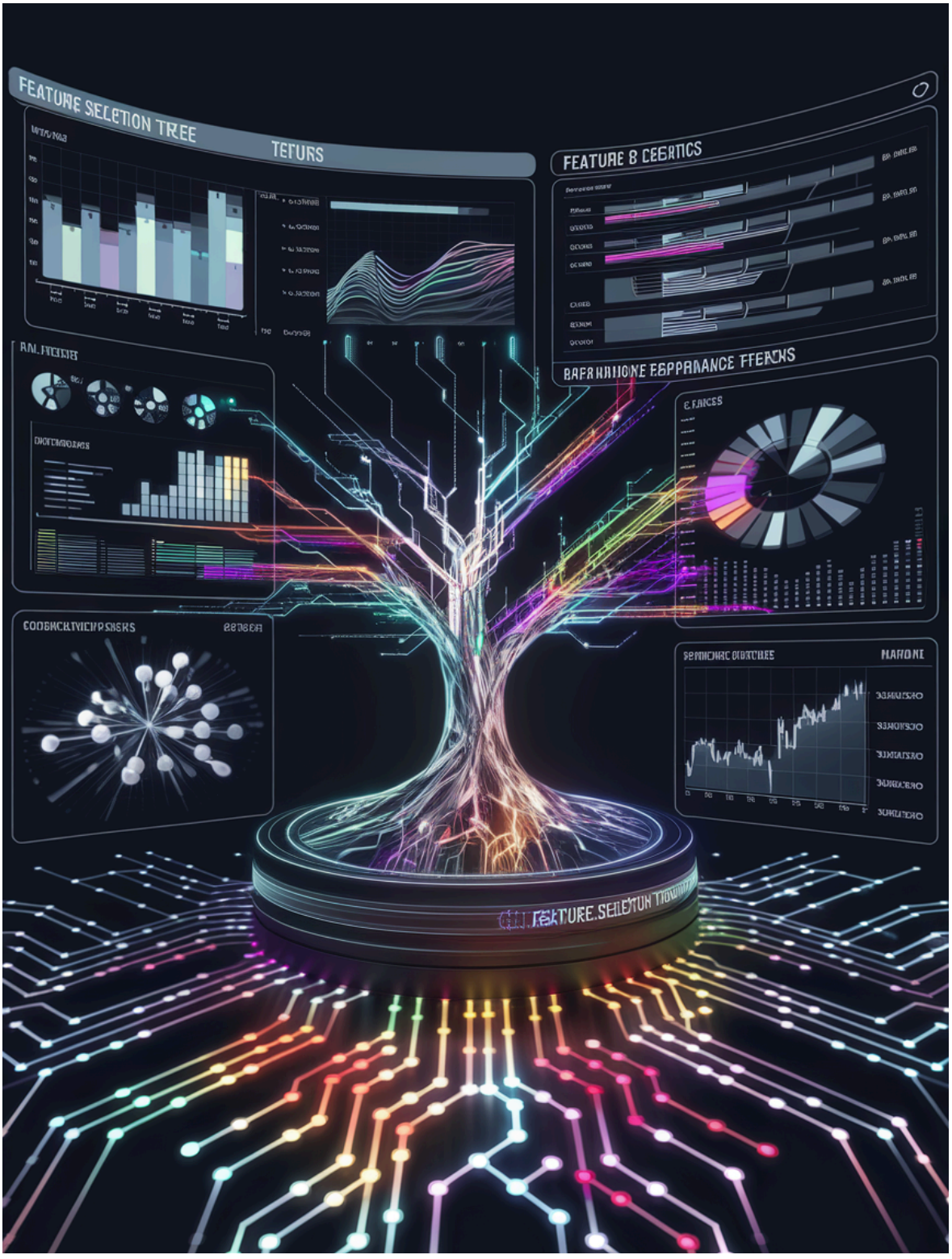
```
    return total_distance

num_dims = num_clients
average_fitness_history = []
best_fitness_history = []
rng = jax.random.PRNGKey(0)
lga_strategy = CustomLGA(
    popsize=pop_size,
    num_dims=num_clients
)

lga_param = lga_strategy.default_params
lga_param = lga_strategy.params_strategy.replace(clip_min=-100)

lga_state = lga_strategy.initialize(rng, lga_param)
for generation in tqdm.tqdm(range(num_generations)):
    rng, rng_init, rng_ask, rng_eval = jax.random.split(rng, 4)
    candidates, lga_state = lga_strategy.ask(rng_ask, lga_state, lga_param)
    if np.isnan(candidates).any():
        print(f"Erro: candidato contém NaN na geração {generation + 1}.
Parando a execução.")
        break

    fitness_scores = []
    for candidate in candidates:
        fitness = vrp_fitness_function_lga(candidate, client_positions)
        fitness_scores.append(fitness)

    fitness_scores = np.array(fitness_scores)
    # Atualizar `lga_state` com os novos fitness scores
    lga_state = lga_strategy.tell(candidates, fitness_scores, lga_state,
lga_param)

    # Guardar média e melhor fitness para plotagem
    average_fitness = fitness_scores.mean()
    average_fitness_history.append(average_fitness)
    best_fitness_history.append(fitness_scores.min())


best_solution_lga = lga_state.best_member
best_fitness_lga = lga_state.best_fitness
#print("Melhor rota:", jnp.argsort(best_solution_lga))
#print("Melhor fitness (distância total):", best_fitness_lga)
```

```
Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl

  0%|          | 0/100 [00:00<?, ?it/s]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[7], line 16
     14 for generation in tqdm.tqdm(range(num_generations)):
     15     rng, rng_init, rng_ask, rng_eval = jax.random.split(rng, 4)
---> 16     candidates, lga_state = lga_strategy.ask(rng_ask, lga_state,
lga_param)
     17     if np.isnan(candidates).any():
     18         print(f"Erro: candidato contém NaN na geração {generation +
1}. Parando a execução.")

    [... skipping hidden 11 frame]

File
~/miniconda3/envs/LGA/lib/python3.10/site-packages/evosax/strategy.py:101,
in Strategy.ask(self, rng, state, params)
     98     params = self.default_params
    100 # Generate proposal based on strategy-specific ask method
--> 101 x, state = self.ask_strategy(rng, state, params)
    102 # Clip proposal candidates into allowed range
    103 x_clipped = jnp.clip(x, params.clip_min, params.clip_max)

Cell In[4], line 56, in CustomLGA.ask_strategy(self, rng, state, params)
     53 x = X_C + sigma_C * epsilon
     55 # Garantir que o depósito esteja no início
---> 56 x = jax.vmap(lambda route: jnp.concatenate([[0], route[1:]]))(x)
     58 return x, state.replace(sigma_C=sigma_C)

    [... skipping hidden 3 frame]

Cell In[4], line 56, in CustomLGA.ask_strategy.<locals>.<lambda>(route)
     53 x = X_C + sigma_C * epsilon
     55 # Garantir que o depósito esteja no início
---> 56 x = jax.vmap(lambda route: jnp.concatenate([[0], route[1:]]))(x)
     58 return x, state.replace(sigma_C=sigma_C)

File
~/miniconda3/envs/LGA/lib/python3.10/site-packages/jax/_src/numpy/lax_numpy
.py:4623, in concatenate(arrays, axis, dtype)
   4621 if isinstance(arrays, (np.ndarray, Array)):
   4622   return _concatenate_array(arrays, axis, dtype=dtype)
-> 4623 util.check_arraylike("concatenate", *arrays)
```

```
   4624 if not len(arrays):
   4625   raise ValueError("Need at least one array to concatenate.")

File
~/miniconda3/envs/LGA/lib/python3.10/site-packages/jax/_src/numpy/util.py:3
16, in check_arraylike(fun_name, emit_warning, stacklevel, *args)
    313   warnings.warn(msg + " In a future JAX release this will be an
error.",
    314                 category=DeprecationWarning, stacklevel=stacklevel)
    315 else:
--> 316   raise TypeError(msg.format(fun_name, type(arg), pos))

TypeError: concatenate requires ndarray or scalar arguments, got <class
'list'> at position 0.
```

A implementação do LGA customizado para o problema de VRP acabou não funcionando, pois não consegui alterar o código de maneira funcional

APÊNDICE 5



Termo de Aceite de Entrega 9

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 27 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Tema: Algoritmos Genéticos de Self Adaptation e Auto Aprendizado via Atenção

Anteriormente: Uso do LGA em diferentes cenários de VRP, porém sem obter soluções boas.

O processo do Stage dessa Semana: Processo Gate 271124

Estudo de benchmarks de otimização contínua em busca de um problema: Benchmarks

- BBOB COCO
- CEC 2023
- HPO-B (Hyperparameter Optimization Benchmark)

Definição do subset do noisy BBOB COCO.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Testar LGA contra o benchmark BBOB e contra o Noisy BBOB.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Reunião com a professora Telma sobre algoritmos genéticos e problemas de otimização

ACEITE DA ENTREGA:

LEONARDO ANTÔNIO ALVES: Em análise!

Conclusão dos Erros

Nas últimas semanas, investiguei o uso do Learnable Genetic Algorithm (LGA) em problemas de otimização contínua, com o objetivo de aplicar essa abordagem ao Vehicle Routing Problem (VRP). Durante essa exploração, constatei que o LGA, embora eficiente em espaços contínuos e no ajuste de variáveis representadas por vetores de valores reais, não se adequa bem ao VRP. Ao longo do processo, identifiquei limitações importantes que me levaram a buscar problemas mais compatíveis com a natureza do LGA.

O VRP, apesar de poder ser modelado como um problema contínuo, é essencialmente discreto em sua essência. Durante minha investigação, percebi que, embora o LGA consiga representar rotas como vetores contínuos, a conversão desses vetores em rotas válidas introduz complexidade adicional e perda de eficiência. Além disso, os operadores de cruzamento e mutação do LGA têm dificuldade em garantir a viabilidade de soluções discretas. Outro ponto crítico foi a multimodalidade do espaço de busca do VRP, que dificulta a convergência do LGA para soluções ótimas. Por essas razões, algoritmos como os genéticos clássicos (GA) ou otimização por colônia de formigas tendem a superar o LGA nesse contexto, devido à sua adaptação natural ao espaço discreto.

Busca por problemas adequados

Com essa constatação, busquei redirecionar o foco para problemas mais alinhados à natureza contínua do LGA. Para isso, consultei a professora Doutora Telma Woerle de Lima Soares, especialista em algoritmos genéticos e problemas de otimização, em busca de sugestões de problemas que se beneficiaram de um método contínuo como o LGA.

Além disso realizei um screening de artigos sobre problemas de otimização contínua, o que me ajudou a compreender melhor o escopo e os desafios desses problemas.

A partir dessas análises e reflexões, identifiquei três problemas particularmente promissores. O primeiro é a otimização de redes neurais, um desafio contínuo. Hiperparâmetros como taxa de aprendizado, momentos, tamanho do batch e número de neurônios em camadas ocultas são variáveis contínuas que podem ser eficientemente ajustadas pelo LGA. Redes neurais frequentemente apresentam superfícies de perda complexas, e o LGA tem o potencial de explorar essas superfícies para identificar regiões promissoras. O objetivo seria reduzir o erro de validação da rede, ajustando

automaticamente seus hiperparâmetros, com aplicações em tarefas como classificação de imagens.

O segundo problema promissor é a otimização de parâmetros de um algoritmo genético (GA). Nesse caso, o LGA pode atuar como um "otimizador de meta-heurística", ajustando dinamicamente parâmetros como taxa de crossover, taxa de mutação, tamanho da população e pressão seletiva em torneios. Durante a execução do GA, o LGA pode monitorar a performance em tempo real e ajustar esses parâmetros para evitar convergência prematura. Isso tem o potencial de melhorar significativamente a eficiência do GA em problemas tanto combinatórios quanto contínuos.

O terceiro seria usar benchmarks de otimização contínua, que são ferramentas fundamentais para avaliar e comparar o desempenho de algoritmos de otimização em diferentes contextos. Entre os benchmarks mais utilizados estão o BBOB, o HPO-B e o CEC 2024, cada um com características específicas que os tornam adequados para diferentes tipos de problemas e configurações experimentais

BBOB (Black-Box Optimization Benchmark)

O BBOB, oferecido pela *COCO (Comparing Continuous Optimizers)*, é amplamente utilizado para testar algoritmos de otimização em problemas de caixa-preta contínuos. Ele inclui funções matemáticas bem definidas, que representam desafios comuns encontrados em otimização, como superfícies rugosas, vales estreitos, e plateaus. As funções do BBOB são categorizadas com base em características como separabilidade, escalabilidade e presença de simetrias.

O BBOB possui uma variante "noisy" que adiciona ruído às avaliações das funções objetivo. Isso é útil para simular problemas do mundo real, onde medições ou avaliações frequentemente contêm incertezas

HPO-B (Hyperparameter Optimization Benchmark)

O HPO-B é um benchmark mais recente, focado na otimização de hiperparâmetros para modelos de machine learning e deep learning. Ele reflete os desafios enfrentados ao ajustar hiperparâmetros em algoritmos de aprendizado, como redes neurais profundas, árvores de decisão ou modelos probabilísticos.

CEC 2023

A edição de 2023 trouxe uma ampla gama de problemas, incluindo funções multimodais, funções de alta dimensionalidade e problemas dinâmicos. Além disso, o benchmark focou em medir a capacidade dos algoritmos de explorar e explorar superfícies complexas de solução

Escolha BBOB:

Para prosseguir com meus teste com o LGA optei pelo benchmarking do BBOB do subset noisy. Isso pois o código do LGA já possui um wrapper para o benchmarking do BBOB COCO, tornando a configuração mais fácil, além disso o subset noisy tem as mesmas características do BBOB visto pelo LGA durante o treinamento tornando seu uso mais viavél

APÊNDICE 6

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião (“gate”) de aprovação: 5 de dez. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

GUSTAVO DOS REIS OLIVEIRA

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Tema: Algoritmos Genéticos com Self-Adaptation e Autoaprendizado via Atenção aplicados a problemas de otimização contínua e roteamento.

Anteriormente: Avaliação do desempenho do LGA em benchmarks de otimização contínua.

O processo do Stage dessa semana: Processo Gate 51224

Progresso da Semana:

- Aplicação do LGA, SimpleGA e LES no benchmark BBOB COCO para comparar estratégias evolutivas e avaliar o desempenho do LGA:
  Experimentos LGA/GA/LES - BBOB
  - O LGA apresentou desempenho inferior em relação às outras estratégias evolutivas, raramente obtendo a melhor solução nos problemas analisados.
- Implementação do LGA em um problema de seleção contínua de features aplicado a um MLP com dados simulados: lga_mlp_feature_selection.ipynb
  - alcançando um ganho percentual de 18,92% na acurácia em comparação ao MLP sem feature selection.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

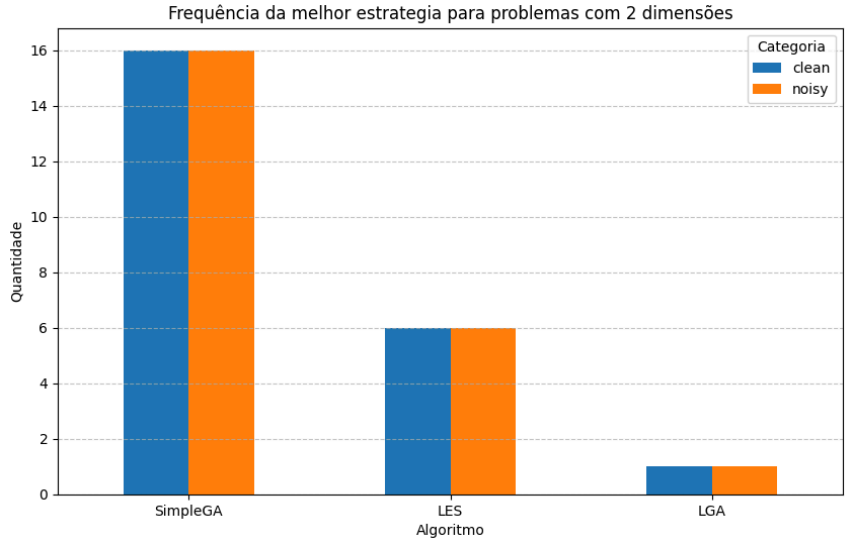
ACEITE DA ENTREGA:

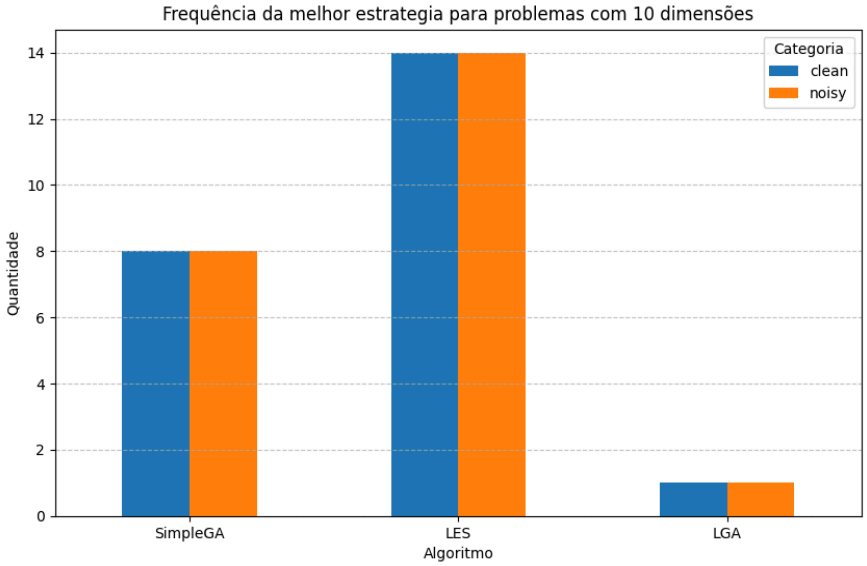
CEDRIC LUIZ DE CARVALHO: Go!

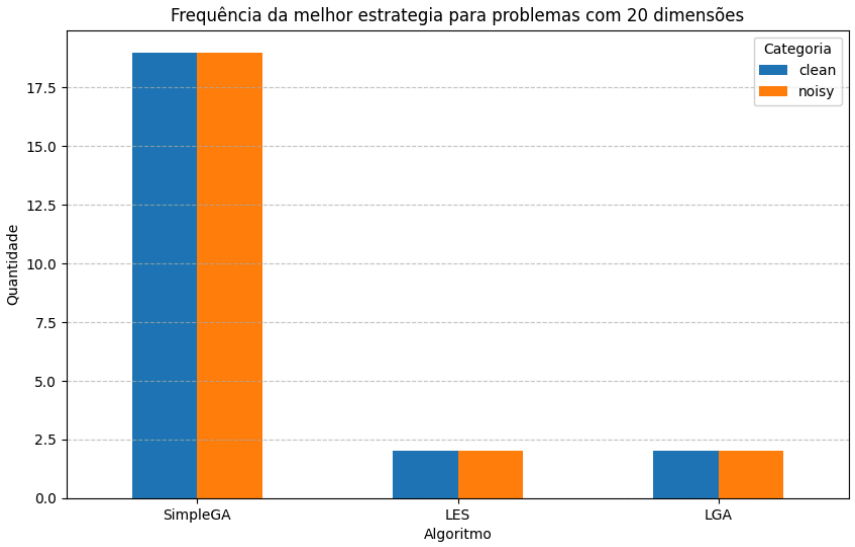
Benchmark das técnicas

Durante esta semana, foquei em avaliar o desempenho de três estratégias evolutivas — LGA (Learnable Genetic Algorithm), Simple GA e LES (Learning Evolution Strategy) — no benchmark COCO BBOB, que já possui uma pré-implementação na biblioteca evosax, utilizada para facilitar a configuração dos experimentos. Os testes seguiram as configurações descritas no paper "Discovering Self-Based Attention Genetic Algorithms", replicando o setup original do LGA para problemas de otimização contínua e comparando seu desempenho com as outras estratégias.

O objetivo foi analisar a eficácia do LGA em problemas de otimização contínua e verificar sua qualidade em relação às outras abordagens. Infelizmente, o LGA demonstrou desempenho inferior, raramente obtendo o melhor fitness nos casos testados, enquanto Simple GA e LES frequentemente superaram o LGA no conjunto de benchmarks analisados


Frequência da melhor estrategia para problemas com 2 dimensões
Categoria
clean
noisy
Quantidade
0
2
4
6
8
10
12
14
16
SimpleGA
LES
LGA
Algoritmo


Frequência da melhor estrategia para problemas com 10 dimensões
Categoria
clean
noisy
Quantidade
0
2
4
6
8
10
12
14
SimpleGA
LES
LGA
Algoritmo



Em poucas vezes o LGA foi a melhor solução para o conjunto de funções do benchmark avaliado

Seleção de features contínuas

Dado esse resultado, decidi explorar outro problema: seleção contínua de features para redes neurais, utilizando a capacidade do LGA de ajustar pesos atribuídos às features. Minha intenção inicial era aplicar essa técnica em imagens hiperespectrais, com foco em imagens médicas, dado seu potencial impacto. No entanto, devido à complexidade dos dados, tamanho elevado e dificuldade em encontrar datasets adequados, optei por desenvolver uma prova de conceito usando datasets sintéticos gerados com o scikit-learn.

Nesse experimento:

1. Criei um dataset de classificação sintético com adição de ruído, simulando um cenário mais desafiador para os modelos.
2. Comparei o desempenho de um MLP (Multi-Layer Perceptron) no dataset, com e sem a seleção de features contínuas realizada pelo LGA.
3. Realizei várias execuções com parâmetros bem definidos para garantir a robustez dos resultados e sua interpretabilidade.

Os resultados foram promissores. A aplicação do LGA para otimizar os pesos das features aumentou significativamente a acurácia do modelo, obtendo um ganho percentual médio de 18,92% em relação ao MLP sem seleção de features. Essa prova de conceito demonstrou o

potencial do LGA em cenários de seleção de features contínuas, abrindo possibilidades para futuras aplicações mais complexas em datasets reais.

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from evosax import LGA

import jax
import jax.numpy as jnp
import numpy as np
from evosax import SimpleGA
import tqdm
import matplotlib.pyplot as plt

import itertools
from evosax import  LGA
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from evosax import LGA

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Rodando no dispositivo: {device}")

Rodando no dispositivo: cuda

popsize = 10
num_generations = 50
num_features = 20
rng = np.random.default_rng(42)

def generate_data(num_samples=2000, num_features=20,
informative_fraction=0.33, noise_std=0.1, noise_range=0.05):
    X, y = make_classification(
        n_samples=num_samples,
        n_features=num_features,
        n_informative=int(num_features * informative_fraction),
        random_state=42
    )
    rng = np.random.default_rng(42)
```

```
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
    X_train_noisy = X_train + rng.normal(0, noise_std, X_train.shape)
    X_test_noisy = X_test + rng.uniform(-noise_range, noise_range,
X_test.shape)
    return X_train_noisy, X_test_noisy, y_train, y_test

# Modelo MLP
class MLP(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(MLP, self).__init__()
        self.layers = nn.Sequential(
            nn.Linear(input_dim, 32),
            nn.ReLU(),
            nn.Linear(32, 32),
            nn.ReLU(),
            nn.Linear(32, 16),
            nn.ReLU(),
            nn.Linear(16, output_dim)
        )

    def forward(self, x):
        return self.layers(x)

def apply_weights(X, weights):
    return X * weights

def evaluate_fitness(weights, X_train, y_train, X_test, y_test, model,
num_epochs=30, sparsity_penalty=0.1):
    weights_np = np.array(weights)
    X_train_np = np.array(X_train)
    X_test_np = np.array(X_test)

    X_train_weighted = apply_weights(X_train_np, weights_np)
    X_test_weighted = apply_weights(X_test_np, weights_np)

    X_train_tensor = torch.tensor(X_train_weighted,
dtype=torch.float32).to(device)
    X_test_tensor = torch.tensor(X_test_weighted,
dtype=torch.float32).to(device)
    y_train_tensor = torch.tensor(y_train, dtype=torch.long).to(device)
    y_test_tensor = torch.tensor(y_test, dtype=torch.long).to(device)

    model = model.to(device)
```

```
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    model.train()

    for epoch in range(num_epochs):
        optimizer.zero_grad()
        outputs = model(X_train_tensor)
        loss = criterion(outputs, y_train_tensor)

        sparsity_loss = sparsity_penalty * np.sum(np.abs(weights_np))
        total_loss = loss + sparsity_loss
        total_loss.backward()
        optimizer.step()

    model.eval()
    with torch.no_grad():
        outputs = model(X_test_tensor)
        predictions = torch.argmax(outputs, dim=1)
        accuracy = (predictions == y_test_tensor).float().mean().item()

    return accuracy

def mlp_train(num_features=20):
    X_train, X_test, y_train, y_test =
generate_data(num_features=num_features)

    # Inicializar pesos fixos (todos iguais a 1)
    weights = [1] * num_features

    # Criar modelo
    model = MLP(input_dim=num_features, output_dim=2)

    accuracy = evaluate_fitness(weights, X_train, y_train, X_test, y_test,
model)

    return accuracy

def optimize_features_with_lga(num_features=20, popsize=32,
num_generations=50):
    X_train, X_test, y_train, y_test =
generate_data(num_features=num_features)

    # Inicializar LGA
    rng = jax.random.PRNGKey(2)
    lga_strategy = LGA(popsize=popsize, num_dims=num_features,
```

```
maximize=True)
    lga_param = lga_strategy.params_strategy.replace(
        init_min=-10.0, init_max=10,
    )
    lga_state = lga_strategy.initialize(rng, lga_param)

    model = MLP(input_dim=num_features, output_dim=2)

    # Loop de otimização
    for generation in range(num_generations):
        rng_init, rng_ask= jax.random.split(rng, 2)
        candidates, lga_state = lga_strategy.ask(rng_ask, lga_state,
lga_param)
        candidates = (candidates - candidates.min(axis=1, keepdims=True)) /
(
            candidates.max(axis=1, keepdims=True) - candidates.min(axis=1,
keepdims=True) + 1e-8
        )
        # Avaliar cada conjunto de pesos
        fitness = np.array([
            evaluate_fitness(candidate, X_train, y_train, X_test, y_test,
model)
            for candidate in candidates
        ])

        # Atualizar o estado do LGA
        lga_state = lga_strategy.tell(candidates, fitness, lga_state,
lga_param)



    best_weights = lga_state.best_member


    return lga_state.best_fitness


n_trials = 10
mlp_acuracies = []
lga_acuracies = []

print("Iniciando os experimentos...\n")
for _ in range(n_trials):
    print(f"Executando experimento {_ + 1}/{n_trials}...\n")
```

```
    # LGA + MLP
    lga_accuracy = optimize_features_with_lga()
    lga_acuracies.append(lga_accuracy)

    # MLP básico
    mlp_accuracy = mlp_train()
    mlp_acuracies.append(mlp_accuracy)

mean_mlp = np.mean(mlp_acuracies)
std_mlp = np.std(mlp_acuracies)

mean_lga = np.mean(lga_acuracies)
std_lga = np.std(lga_acuracies)

print("\nResultados Finais:\n")
print(f"MLP Básico (sem feature selection):")
print(f"  Média da Acurácia: {mean_mlp:.4f}")
print(f"  Desvio Padrão: {std_mlp:.4f}\n")

print(f"LGA + MLP (com feature selection):")
print(f"  Média da Acurácia: {mean_lga:.4f}")
print(f"  Desvio Padrão: {std_lga:.4f}\n")

if mean_lga > mean_mlp:
    print("Conclusão: O uso de LGA para feature selection melhorou a
performance do MLP.")
else:
    print("Conclusão: O MLP básico teve performance comparável ou melhor.")

Iniciando os experimentos...

Executando experimento 1/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.7867
Executando experimento 2/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.8250
Executando experimento 3/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.8100
Executando experimento 4/10...
```

```
Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.8117
Executando experimento 5/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.7017
Executando experimento 6/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.7633
Executando experimento 7/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.8250
Executando experimento 8/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.7700
Executando experimento 9/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.8283
Executando experimento 10/10...

Loaded pretrained LGA model from ckpt: 2023_04_lga_v4.pkl
Acurácia obtida (MLP Básico, sem feature selection): 0.7683

Resultados Finais:

MLP Básico (sem feature selection):
  Média da Acurácia: 0.7890
  Desvio Padrão: 0.0376

LGA + MLP (com feature selection):
  Média da Acurácia: 0.9383
  Desvio Padrão: 0.0031

Conclusão: O uso de LGA para feature selection melhorou a performance do
MLP.

mean_mlp = np.mean(mlp_acuracies)
std_mlp = np.std(mlp_acuracies)

mean_lga = np.mean(lga_acuracies)
std_lga = np.std(lga_acuracies)
```

```
print("\nResultados Finais:\n")
print(f"MLP Básico (sem feature selection):")
print(f"  Média da Acurácia: {mean_mlp:.4f}")
print(f"  Desvio Padrão: {std_mlp:.4f}\n")
print(f"LGA + MLP (com feature selection):")
print(f"  Média da Acurácia: {mean_lga:.4f}")
print(f"  Desvio Padrão: {std_lga:.4f}\n")

if mean_lga > mean_mlp:
    print("Conclusão: O uso de LGA para feature selection melhorou a
performance do MLP.")
else:
    print("Conclusão: O MLP básico teve performance comparável ou melhor.")
```

```
Resultados Finais:

MLP Básico (sem feature selection):
  Média da Acurácia: 0.7890
  Desvio Padrão: 0.0376

LGA + MLP (com feature selection):
  Média da Acurácia: 0.9383
  Desvio Padrão: 0.0031

Conclusão: O uso de LGA para feature selection melhorou a performance do
MLP.
```

```
percentage_gain = ((mean_lga - mean_mlp) / mean_mlp) * 100
print(f"Ganho percentual em acurácia: {percentage_gain:.2f}%")
```

```
Ganho percentual em acurácia: 18.92%
```

```
results = {
    "Method": ["MLP Basic", "LGA + MLP"],
    "Mean_Accuracy": [mean_mlp, mean_lga],
    "Std_Deviation": [std_mlp, std_lga]
}

results_df = pd.DataFrame(results)

results_df.to_csv("benchmark_comparison_results.csv", sep=";", decimal=",",
index=False)
print("Arquivo salvo como 'benchmark_comparison_results.csv")
```