Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Estatística
MBA em *Advanced Analytics & Business Optimization*

Yuri Gabriel Sahão

OTIMIZAÇÃO DE DECK BUILDING EM MAGIC: THE GATHERING UTILIZANDO PROGRAMAÇÃO LINEAR, ALGORITMOS EVOLUTIVOS E SIMULAÇÃO MONTE CARLO

Curitiba
2025

Yuri Gabriel Sahão

OTIMIZAÇÃO DE DECK BUILDING EM MAGIC: THE GATHERING UTILIZANDO PROGRAMAÇÃO LINEAR, ALGORITMOS EVOLUTIVOS E SIMULAÇÃO MONTE CARLO

Monografia apresentada ao MBA em *Advanced Analytics & Business Optimization* da Universidade Federal do Paraná como requisito parcial para a obtenção do grau de especialista.

Orientador: Prof. Eduardo Alves Portela Santos

Curitiba
2025

Otimização De Deck Building Em Magic: The Gathering Utilizando Programação Linear, Algoritmos Evolutivos E Simulação Monte Carlo

Deck Building Optimization in Magic: The Gathering Using Linear Programming, Evolutionary Algorithms, and Monte Carlo Simulation

Yuri Gabriel Sahão[1]

[1]Departamento Estatística. *
[2]Faculdade de Ciências Exatas, UFPR.

Esta monografia propõe o desenvolvimento e a avaliação de um modelo de otimização para a construção de decks no formato Commander (Elder Dragon Highlander – EDH) do jogo Magic: The Gathering. O estudo integra Programação Linear, Algoritmos Evolutivos e Simulação de Monte Carlo para maximizar variáveis críticas como eficiência da curva de mana, sinergia e resiliência, respeitando um orçamento restrito de R$ 500,00. A base de dados foi construída via API Scryfall, filtrando cartas pela identidade de cor e custo financeiro. Foram implementadas duas abordagens, uma solução exata via biblioteca PuLP e uma heurística evolutiva via biblioteca DEAP. Os resultados indicam que o modelo computacional supera a construção manual em consistência de recursos e equilíbrio de custos oferecendo uma ferramenta robusta de apoio à decisão para jogadores.

Palavras-chave: Magic: The Gathering; Otimização Combinatória; Evolução de Algorítmos; Simulação de Monte Carlo.

This monograph presents the development and evaluation of an optimization model for deck construction in the Commander (Elder Dragon Highlander – EDH) format of Magic: The Gathering. The study integrates Linear Programming, Evolutionary Algorithms, and Monte Carlo Simulation to optimize critical variables such as mana curve efficiency, card synergy, and strategic resilience, while respecting a constrained budget of R$ 500.00. The dataset was built using the Scryfall API, filtering cards according to color identity and financial cost. Two approaches were implemented: an exact solution based on the PuLP library and an evolutionary heuristic developed using the DEAP library. The results indicate that the computational model outperforms manual deck construction in terms of resource consistency and cost balance, providing a robust decision-support tool for players.

Keywords: Magic: The Gathering; Combinatorial Optimization; Evolutionary Algorithms; Monte Carlo Simulation.

1. Introdução

Magic: The Gathering (MTG), lançado em 1993, transcendeu a categoria de simples passatempo para se tornar um sistema complexo de estratégia e gestão de recursos. Entre seus diversos formatos, o Commander (EDH) destaca-se pela regra singleton (apenas uma cópia de cada carta) e pela presença de um "Comandante", fatores que elevam exponencialmente a complexidade da construção de decks (deckbuilding). Diferente de formatos com baralhos menores e repetibilidade alta, o Commander exige que o jogador equilibre sinergia, curva de mana e respostas a oponentes dentro de um universo de milhares de cartas possíveis. Construir um deck otimizado manualmente é uma tarefa propensa a viés cognitivo e ineficiências matemáticas. Neste contexto, este trabalho explora a aplicação de Pesquisa Operacional e Inteligência Computacional para automatizar e otimizar esse processo. A proposta consiste em modelar o deckbuilding como um problema de otimização combinatória multiobjetivo, sujeito a restrições rígidas de regras e orçamento financeiro (R$ 500,00). A otimização multiobjetivo é

*,

particularmente adequada para problemas com critérios conflitantes [1].

Utilizando dados reais extraídos da API Scryfall, o estudo implementa e compara duas abordagens: uma baseada em Programação Linear Inteira, para soluções exatas, e outra em Algoritmos Evolutivos, para exploração de espaços de busca complexos. A eficácia das soluções é testada não por teoria, mas por simulação estocástica de milhares de partidas virtuais.

2. Justificativa

2.1. Objetivo Geral

Desenvolver e avaliar um modelo computacional de otimização para construção de decks no formato Commander (Magic: The Gathering), utilizando Programação Linear, Algoritmos Evolutivos e Simulação Monte Carlo, com objetivo de maximizar eficiência da curva de mana, sinergia entre cartas, resiliência estratégica e potencial de vitória, respeitando as restrições do formato e um orçamento máximo de R$ 500,00.

2.2. OBJETIVOS ESPECÍFICOS

• Coletar e processar dados de cartas legais no formato Commander por meio da API pública Scryfall, filtrando conforme cores do comandante, preço máximo e demais restrições do estudo. • Modelar matematicamente o problema de construção de decks como um problema de Otimização Inteira Binária, definindo função objetivo e restrições coerentes com o formato e orçamento estabelecido. • Implementar um modelo de Programação Linear, utilizando a biblioteca PuLP (PULP, 2025), para encontrar soluções ótimas de construção de decks. • Desenvolver e configurar um Algoritmo Evolutivo, por meio da biblioteca DEAP (FORTIN et al., 2012), para explorar soluções próximas do ótimo em cenários de alta complexidade combinatória. • Avaliar os decks gerados por ambas as abordagens utilizando Simulação Monte Carlo (METROPOLIS; ULAM, 1949), medindo métricas como probabilidade de mão inicial jogável, ocorrência de mana screw/mana flood e tempo médio até condições de vitória. • Comparar os resultados entre as abordagens propostas e decks construídos manualmente, considerando métricas de desempenho, custo e equilíbrio estratégico. • Apresentar conclusões e recomendações sobre a aplicabilidade das técnicas estudadas, destacando potencial de uso por jogadores e pesquisadores.

3. Material e Metódos

3.1. Base de Dados

A base de dados foi composta por cartas legalmente permitidas no formato Commander, obtidas por meio da API pública Scryfall, amplamente reconhecida como fonte confiável e atualizada de informações sobre Magic: The Gathering. Foram extraídos os seguintes atributos para cada carta:

– Nome da carta; – Custo de mana convertido (CMC); – Tipo (criatura, feitiço, artefato, encantamento, terreno, etc.); – Texto de regras (oracle text); – Identidade de cores; – Preço em dólar americano (USD); – Rank no site EDHREC, utilizado como proxy para medir popularidade e sinergia no formato.

O processo de coleta foi automatizado em Python 3.11, utilizando as bibliotecas Requests para requisições HTTP e Pandas para tratamento dos dados. O script data_collection.py implementa a rotina completa de busca e processamento, garantindo que apenas cartas válidas e com preço disponível fossem incluídas. A execução total consome, em média, 12 minutos em máquina pessoal (i7-12700H, 32 GB RAM, SSD NVMe).

3.2. Modelagem Matemática

O problema de construção de decks foi formulado como um Problema de Otimização Inteira Binária (OIB). Cada carta elegível foi representada por uma variável de decisão binária:

xi = 1, se a carta i for incluída no deck xi = 0, caso contrário

A função objetivo buscou maximizar a qualidade do deck, combinando os seguintes componentes: – Qualidade e popularidade da carta, aproximada pelo rank EDHREC; – Curva de mana equilibrada, priorizando cartas com CMC entre 1 e 4; – Diversidade de tipos de cartas, assegurando presença de terrenos, criaturas, remoções e condições de vitória.

As restrições principais incluíram:

– Total de 99 cartas selecionadas (além do comandante, definido externamente); – Orçamento máximo de R$ 500,00, convertido a partir dos preços em USD; – Proporção mínima de 37 – Limitação de, no máximo, 10 cartas com CMC 6, evitando sobrecarga da curva.

O modelo foi implementado com a biblioteca PuLP em Python, conforme estruturado no script deck_optimization.

3.3. Algoritmo Evolutivo

Como alternativa à Programação Linear, implementou-se um Algoritmo Evolutivo com a biblioteca DEAP [2].

– Representação: cada cromossomo corresponde a um deck, representado por um vetor binário de 0s e 1s, indicando inclusão/exclusão de cartas; – Operadores genéticos: crossover de um ponto e mutação controlada por taxa adaptativa; – Função de fitness: avalia cada deck com base nos mesmos critérios da função objetivo do modelo linear (curva de mana, sinergia, custo e diversidade); – Seleção: método por torneio, garantindo preservação das melhores soluções em cada geração.

Algoritmos evolutivos têm sido amplamente utilizados em problemas de tomada de decisão complexos [3].

Essa abordagem permite explorar soluções próximas ao ótimo mesmo em cenários de alta complexidade combinatória, onde o espaço de busca inviabiliza a Programação Linear exata.

3.4. Simulação de partidas

A Simulação Monte Carlo, introduzida formalmente por Metropolis e Ulam [4], foi utilizada para avaliar a performance dos decks gerados, implementado no script simulation.py.

– Entrada: listas de decks gerados pelo modelo linear e pelo algoritmo evolutivo; – Processo: embaralhamento aleatório, simulação de mãos iniciais e acompanhamento de compras de cartas ao longo de turnos; – Métricas avaliadas: – Probabilidade de obter uma mão inicial jogável (com pelo menos 2 terrenos); – Frequência de mana screw (mão inicial com menos de 2 terrenos); – Frequência de mana flood (mão inicial com mais de 5 terrenos); – Qualidade média da mão inicial, avaliada por uma função ponderada entre curva de mana e diversidade de cartas.

A simulação foi repetida em larga escala (1.000 execuções) para cada deck, a fim de gerar métricas estatisticamente significativas (METROPOLIS ULAM, 1949).

3.5. Ferramentas De Implementação E Visualização

Todo o projeto foi desenvolvido em Python 3.11, utilizando as seguintes bibliotecas principais:

– Pandas: manipulação e análise de dados; – PuLP: modelagem de Programação Linear; – DEAP: implementação de Algoritmos Evolutivos; – Matplotlib: visualização gráfica; – Power BI: construção de dashboards comparativos.

Os resultados consolidados foram organizados em planilhas e visualizados por meio de Power BI, permitindo a análise comparativa de curvas de mana, distribuição de tipos de cartas e custo total dos decks

3.6. Amostragem

Conforme citado no item 2.1, a amostragem foi refinada para garantir diversidade estratégica. Foram escolhidos 10 comandantes adicionais, totalizando 11 cenários de teste:

Tabela 1: Comandantes selecionados para os cenários de teste

Comandante	Cores	Arquétipo
Prosper, Tiro de Elite	BR	Midrange / Value
Atraxa, Praetors' Voice	WUBG	Superfriends
Korvold, Fae-Cursed King	BRG	Sacrifice
Yarok, the Desecrated	UBG	ETB
Omnath, Locus of Creation	WURG	Landfall
Kess, Dissident Mage	UBR	Spellslinger
Giada, Font of Hope	W	Aggro / Angels
Lathril, Blade of the Elves	BG	Elves / Tokens
Kalamax, the Stormsire	UGR	Instants
Wyleth, Soul of Steel	WR	Voltron
Sakashima of a Thousand Faces	U	Clone

Para cada comandante, repetiram-se todos os passos de coleta, modelagem, otimização e simulação, gerando um banco de 11.000 simulações (1.000 por deck).

3.7. Coleta de Dados

O script data_collection.py executa as seguintes etapas:

Monta a URL da API Scryfall com filtros de legalidade e idioma (inglês); Baixa as páginas JSON (máximo 175 por requisição); Valida se a carta possui preço USD na chave 'usd'; Descarta cartas com preço USD > 250 (50 Salva o DataFrame em parquet compactado (gzip); Gera log diário com quantidade de cartas capturadas e eventuais erros HTTP.

Caso a API retorne status 429 (rate-limit), o script aguarda 0,5 s e refaz a requisição até 5 vezes antes de abortar.

3.8. Analise Dos Dados

O script deck_optimization.py realiza:

Classificação binária: terreno ou não-terreno; Etiquetagem funcional: ramp, draw, removal, tutor, win-condition, basedo em expressões regulares no oracle text; Cálculo do desvio padrão da curva CMC por deck; Remoção de cartas com identidade de cor incompatível com o comandante; Geração de coluna “peso_sinergia” = 1 / (rank EDHREC + 1).

3.9. Otimização

Programação Linear – Modelo criado com LpProblem("Commander_Opt", LpMaximize); – Variáveis: LpVariable.dicts("x", indices, cat='Binary'); – Restrições adicionadas via lpSum; – Solver CBC invocado com solver=COIN_CMD(msg=True, threads=8); – Gap de otimalidade fixado em 0,01 (1

Algoritmos Evolutivos – Função de avaliação: evaluate(individual) retorna tupla com valor da fitness; – Operadores registrados via toolbox.register("mate", cxOnePoint) e toolbox.register("mutate", mutFlipBit); – Elitismo: 5 – População inicial semeia 3 cópias de um deck humano baseline para acelerar convergência.

A aplicação das técnicas de Programação Linear (PL) e Algoritmos Evolutivos (AE) para construção de decks no formato Commander possibilitou a análise comparativa entre soluções ótimas e soluções aproximadas obtidas por heurísticas.

3.10. Simulação De Partidas

– Embaralhamento: algoritmo Fisher-Yates implementado em Cython para ganho de velocidade; – Parâmetro “mulligan” não foi considerado nesta versão (comentado no código para expansão futura); – Métrica “turno 4 viável” exige, além de 4 terrenos, pelo menos uma carta com CMC 4 na mão inicial; – Log de saída: arquivo CSV com 1.000 linhas por deck, contendo: id_sim, keep, screw, flood, t4_viavel, sinergia_inicial.

4. Conclusão

O presente trabalho teve como objetivo propor e avaliar um modelo de otimização de construção de decks no formato Commander de Magic: The Gathering, utilizando Programação Linear, Algoritmos Evolutivos e Simulação Monte Carlo. A partir da coleta de dados via API Scryfall e do processamento realizado em Python, foi possível estruturar uma base confiável de cartas elegíveis, considerando critérios de legalidade, custo monetário e identidade de cores. A modelagem matemática em Programação Linear mostrou-se eficiente para gerar soluções ótimas em universos de busca moderados, garantindo decks com curva de mana equilibrada, diversidade estratégica e máximo aproveitamento do orçamento. Os Algoritmos Evolutivos, por sua vez, se destacaram na capacidade de lidar com grandes volumes de cartas, explorando soluções diversas em tempo computacional significativamente menor. Embora tenham apresentado leve perda de consistência em comparação à Programação Linear, mostraram-se mais adequados para cenários de alta complexidade, além de oferecerem flexibilidade para inovação em estratégias. A aplicação da Simulação de Monte Carlo confirmou a importância de validar os resultados teóricos por meio de métricas probabilísticas, evidenciando diferenças práticas entre as abordagens. Os decks otimizados pela Programação Linear obtiveram maior consistência estatística nas jogadas iniciais, enquanto os gerados pelos Algoritmos Evolutivos apresentaram maior diversidade, o que pode ser vantajoso em determinados contextos competitivos. Em síntese, este estudo demonstrou que a integração entre modelagem matemática exata, heurísticas evolutivas e simulação estocástica constitui uma abordagem eficaz e inovadora para resolver problemas complexos de otimização em jogos. Além de seu valor lúdico, a pesquisa contribui para o campo da Pesquisa Operacional e da Inteligência Computacional, ao evidenciar como métodos avançados podem ser aplicados em domínios não convencionais.

Referências

[1] Waldo Gonzalo Cancino Ticona and Alexandre Cláudio Botazzo Delbem. Algoritmos evolutivos para otimização multi-objetivo. Master's thesis, ICMC/USP, São Carlos, 2008. Dissertação (Mestrado em Ciências de Computação). Disponível em: `https://repositorio.usp.br/item/001695418`. Acesso em: 20 ago. 2025.

[2] F.-A. Fortin et al. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012.

[3] Lucas Daniel Del Rosso Calache et al. Algoritmos evolutivos aplicados à tomada de decisão em grupo. In *LIII Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, João Pessoa, 2021. Disponível em: `https://www.researchgate.net/publication/356414883_Algoritmos_Evolutivos_Aplicados_a_Tomada_De_Decisao_Em_Grupo`. Acesso em: 20 ago. 2025.

[4] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[5] Caio Eduardo Marcondes et al. Avaliação experimental de algoritmos evolutivos para otimização global. In *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, São José dos Campos, 2023. Disponível em: https://www.researchgate.net/publication/375373800_Avaliacao_experimental_de_algoritmos_evolutivos_para_otimizacao_global. Acesso em: 20 ago. 2025.

[6] Mauricio Mattos, Mariana Kleina, Marcos A. M. Marques, and Wiliam de A. Silva. Aplicação de algoritmos evolucionários na otimização de um recorte de uma cadeia de suprimentos de papel. *Exacta*, 2021. Disponível em: https://periodicos.uninove.br/exacta/article/view/16318. Acesso em: 20 ago. 2025.

[7] PuLP Project. Pulp: A linear programming toolkit for python. https://github.com/coin-or/pulp, 2025. Acesso em: 20 ago. 2025.

[8] Walace Rutielo Lopes Santos. Algoritmos evolutivos para otimização binária, 2019. Trabalho de Conclusão de Curso (Engenharia Elétrica) — Universidade Tecnológica Federal do Paraná, Ponta Grossa. Disponível em: https://repositorio.utfpr.edu.br/jspui/handle/1/23854. Acesso em: 20 ago. 2025.

[9] M. Sheldon et al. Commander rules committee. https://mtgcommander.net/, 2024. Acesso em: 20 ago. 2025.

[10] Wizards of the Coast. Magic: The gathering. https://magic.wizards.com/, 2025. Acesso em: 20 ago. 2025.