# Relatório Final Abrangente do Projeto CIFO EXTENDED

**Equipa CIFO**

**Data:** 15 de maio de 2025

## Sumário Executivo

Este relatório final consolida todas as fases do projeto CIFO EXTENDED, que visou a otimização da seleção de equipas desportivas através da aplicação e análise de algoritmos meta-heurísticos. O projeto evoluiu desde implementações em processador único, passando por otimizações de código, exploração de multiprocessamento com diferentes números de execuções, até uma análise aprofundada da variação de parâmetros para os algoritmos mais promissores. As principais conclusões apontam para o Hill Climbing (especificamente com 500 iterações) como a abordagem mais eficiente em termos de equilíbrio entre qualidade da solução, consistência e tempo de execução. Para cenários que exigem a garantia da melhor fitness possível com consistência perfeita, uma configuração otimizada do Algoritmo Genético (GA\_Config\_4 com taxa de mutação de 0.15) demonstrou ser superior, embora com um custo computacional mais elevado. Este documento detalha a metodologia, os resultados e as conclusões de cada fase, culminando com recomendações para trabalhos futuros e um diagrama do processo de análise empregue.

## 1. Introdução Geral ao Projeto

O desafio central do projeto CIFO EXTENDED foi desenvolver e avaliar métodos computacionais eficazes para o problema complexo de formar múltiplas equipas desportivas a partir de um conjunto de jogadores disponíveis. Este problema de otimização combinatória é caracterizado por diversas restrições, incluindo orçamentos por equipa, número de jogadores por equipa e requisitos posicionais específicos. O objetivo principal da otimização foi minimizar a variância da média de habilidade entre as equipas formadas, promovendo assim o equilíbrio competitivo.

Ao longo do projeto, foram explorados diversos algoritmos meta-heurísticos, incluindo Hill Climbing (HC), Simulated Annealing (SA) e Algoritmos Genéticos (AGs) com uma variedade de operadores. A investigação progrediu através de várias fases distintas:

1. **Fase de Processador Único:** Implementação inicial dos algoritmos, profiling para identificação de gargalos e otimizações de código (redução de deepcopy e vetorização).
2. **Fase de Multiprocessamento - 5 Execuções:** Avaliação inicial do desempenho dos algoritmos em paralelo para obter dados estatisticamente mais robustos e selecionar candidatos para análises mais profundas.
3. **Fase de Multiprocessamento - 30 Execuções com Algoritmos Promissores:** Análise estatística mais rigorosa dos algoritmos selecionados (HC, SA, e duas configurações de AG) com os seus parâmetros base.
4. **Fase de Variação de Parâmetros - 30 Execuções por Variação:** Investigação detalhada do impacto de diferentes hiperparâmetros (número de iterações para HC; taxa de mutação, tamanho da população, número de gerações para os AGs selecionados) no desempenho.

Este relatório visa apresentar uma visão consolidada de todo o processo, desde a conceção inicial até às conclusões finais e recomendações.

## 2. Fase de Processador Único: Implementação e Otimizações Iniciais

### 2.1. Arquitetura do Código de Processador Único (Versão Inicial)

A arquitetura do código para a fase de processador único foi concebida de forma modular para facilitar a implementação, teste e substituição de diferentes componentes dos algoritmos meta-heurísticos. Os principais ficheiros Python envolvidos nesta arquitetura são:

* solution.py: Define a representação de uma solução para o problema (uma atribuição de jogadores a equipas) e os métodos para avaliar a sua validade e qualidade (fitness). Inclui também métodos para gerar soluções vizinhas ou modificar soluções existentes.
* evolution.py: Contém as implementações dos principais algoritmos meta-heurísticos: Hill Climbing, Simulated Annealing e o Algoritmo Genético genérico.
* operators.py: Implementa os diferentes operadores genéticos utilizados pelo Algoritmo Genético, tais como operadores de seleção, cruzamento e mutação.
* main\_script\_sp.py: O script principal que orquestra a execução dos algoritmos, carrega os dados dos jogadores, configura os parâmetros dos algoritmos e recolhe/apresenta os resultados.

#### 2.1.1. Representação da Solução e Estruturas de Dados

O racional por detrás da escolha da representação da solução e das estruturas de dados foi a simplicidade e a eficiência para as operações necessárias.

* **Representação da Solução (LeagueSolution em solution.py):** Uma solução é representada como uma lista (ou, posteriormente, um array NumPy) onde o índice corresponde ao ID de um jogador e o valor nesse índice corresponde ao ID da equipa à qual esse jogador foi atribuído. Por exemplo, assignment[player\_id] = team\_id.
  + **Racional:** Esta representação é direta, fácil de manipular e permite um acesso rápido à equipa de qualquer jogador. É também compacta.
* **Dados dos Jogadores:** Os dados dos jogadores (ID, nome, posição, salário, habilidade) são inicialmente carregados a partir de um ficheiro CSV para uma estrutura de dados conveniente (como uma lista de dicionários ou um DataFrame Pandas) no script principal. Para as operações internas das classes de solução e algoritmos, estes dados são frequentemente passados ou convertidos para formatos mais otimizados para cálculo (como arrays NumPy para salários, habilidades e posições numéricas, como foi feito na fase de vetorização).
  + **Racional:** O CSV é um formato comum e fácil de usar para entrada de dados. Internamente, a conversão para arrays NumPy (especialmente após a vetorização) visa acelerar os cálculos numéricos.
* **Estrutura das Equipas:** Implicitamente, as equipas são coleções de jogadores. A representação da solução permite reconstruir facilmente a composição de cada equipa, filtrando os jogadores com base no team\_id que lhes foi atribuído.

#### 2.1.2. Algoritmos Implementados (Versão Inicial, Não Otimizada)

Foram selecionados três tipos principais de algoritmos meta-heurísticos para esta fase inicial, devido à sua popularidade e diferentes abordagens à exploração do espaço de soluções:

1. **Hill Climbing (HC):**
   * **Descrição:** Um algoritmo de busca local iterativo que começa com uma solução arbitrária e tenta encontrar uma solução melhor movendo-se para um vizinho com melhor fitness. Termina quando atinge um ótimo local, onde nenhum vizinho tem um fitness melhor.
   * **Racional da Seleção:** É um dos algoritmos de otimização mais simples de implementar e entender. Serve como uma boa linha de base para comparar com algoritmos mais complexos. É rápido, mas propenso a ficar preso em ótimos locais.
2. **Simulated Annealing (SA):**
   * **Descrição:** Uma técnica de otimização probabilística inspirada no processo de recozimento em metalurgia. Permite movimentos para soluções piores com uma certa probabilidade, que diminui à medida que o algoritmo progride (a “temperatura” baixa). Isto ajuda a escapar de ótimos locais.
   * **Racional da Seleção:** Oferece uma melhor capacidade de exploração global do que o Hill Climbing, com a capacidade de evitar ótimos locais. A sua eficácia depende da correta parametrização do esquema de arrefecimento.
3. **Algoritmos Genéticos (AGs):**
   * **Descrição:** Algoritmos de busca inspirados na evolução biológica. Mantêm uma população de soluções candidatas que evoluem ao longo de gerações através de operadores genéticos como seleção, cruzamento (recombinação) e mutação.
   * **Racional da Seleção:** São poderosos para problemas de otimização complexos e podem explorar eficazmente grandes espaços de soluções. A sua natureza baseada em população permite manter diversidade e explorar múltiplas regiões do espaço de busca simultaneamente.

#### 2.1.3. Seleção de Operadores para Teste (Algoritmos Genéticos)

Para os Algoritmos Genéticos, foi implementada uma variedade de operadores para permitir testar diferentes estratégias evolutivas:

* **Operadores de Seleção:** Determinam quais indivíduos da população atual são escolhidos para se reproduzirem.
  + selection\_tournament: Seleção por torneio.
  + selection\_roulette\_wheel: Seleção por roleta.
  + selection\_ranking: Seleção baseada no ranking.
  + selection\_boltzmann.
* **Operadores de Cruzamento (Crossover):** Combinam o material genético de dois pais para criar um ou mais filhos.
  + crossover\_one\_point: Cruzamento de um ponto.
  + crossover\_uniform: Cruzamento uniforme.
  + Versões \_prefer\_valid para tentar gerar filhos válidos mais diretamente.
* **Operadores de Mutação:** Introduzem pequenas alterações aleatórias nos indivíduos.
  + mutation\_random\_player\_team\_change: Muda aleatoriamente a equipa de um jogador.
  + mutation\_swap\_players\_between\_teams: Troca dois jogadores entre duas equipas.

A seleção inicial destes algoritmos e operadores visou cobrir um espectro de abordagens meta-heurísticas.

### 2.2. Resultados das Otimizações Iniciais (Processador Único)

Após a implementação inicial, procedeu-se a uma fase de profiling e otimização. As duas principais áreas de otimização focadas foram a redução do uso de deepcopy e a vetorização de funções críticas.

#### 2.2.1. Profiling Inicial e Identificação de Gargalos

A execução inicial do script (main\_script\_sp\_timing\_estimate.py) revelou que o tempo total era de aproximadamente **85.66 segundos**. O cProfile destacou:

* **Simulated Annealing (SA):** O mais lento, consumindo cerca de **73.17 segundos**.
* **copy.deepcopy:** Principal causa da lentidão, chamada mais de 23 milhões de vezes, totalizando **53.49 segundos**.
* **Funções de Solução:** get\_random\_neighbor, fitness, e is\_valid também consumiam tempo significativo.

#### 2.2.2. Otimização do Uso de deepcopy no Simulated Annealing

* **Alteração:** No SA, chamadas a deepcopy(neighbor\_solution) para atualizar current\_solution foram substituídas por atribuições diretas. A chamada a deepcopy para best\_solution foi mantida.
* **Impacto:** Redução do tempo total do script para **41.42 segundos** (melhoria de ~51.7%). Tempo do SA diminuiu para **28.91 segundos** (melhoria de ~60.5%).

#### 2.2.3. Vetorização das Funções is\_valid() e fitness()

* **Alterações:** Funções is\_valid() e fitness() em solution.py foram reescritas usando NumPy.
* **Impacto:** Tempo total do script com todas as otimizações foi de aproximadamente **40.96 segundos**. A vetorização não trouxe melhoria adicional significativa no tempo global neste contexto, mas tornou o código mais robusto.

### 2.3. Conclusões da Fase de Processador Único

A fase de processador único foi crucial. A otimização de deepcopy resultou numa redução significativa do tempo de execução. A vetorização modernizou a base de código, preparando-a para cenários mais exigentes. Estes resultados forneceram informações valiosas para a exploração do multiprocessamento.

## 3. Fase de Multiprocessamento - 5 Execuções: Avaliação Inicial

Esta secção do projeto focou-se na avaliação do desempenho dos algoritmos de otimização (Hill Climbing, Simulated Annealing e quatro configurações de Algoritmos Genéticos) quando executados múltiplas vezes em paralelo. O objetivo principal desta fase foi obter dados estatisticamente mais robustos sobre a eficácia e eficiência de cada abordagem, executando cada algoritmo 5 vezes. A utilização de multiprocessamento permitiu realizar estas execuções concorrentes, otimizando o tempo total de experimentação.

### 3.1. Metodologia Experimental

Os algoritmos avaliados foram: Hill Climbing (HC), Simulated Annealing (SA), e quatro configurações de Algoritmos Genéticos (GA\_Config\_1, GA\_Config\_2, GA\_Config\_3, GA\_Config\_4), variando operadores de mutação, cruzamento e seleção. Cada algoritmo/configuração foi executado 5 vezes em paralelo. As métricas coletadas incluíram melhor fitness geral, fitness média, desvio padrão da fitness e tempo médio de execução.

### 3.2. Resultados e Discussão (5 Execuções)

Após a execução do script main\_script\_mp.py para 5 execuções de cada algoritmo, os resultados foram:

* **Melhor Fitness Geral:** Todos os algoritmos (SA, GA\_Config\_1, GA\_Config\_2, GA\_Config\_3, e GA\_Config\_4) conseguiram alcançar o mesmo valor de Melhor Fitness Geral de aproximadamente **0.057143**. O Hill Climbing, embora tenha atingido este valor, apresentou uma Fitness Média ligeiramente superior (0.0671).
* **Consistência (Desvio Padrão da Fitness):** O Simulated Annealing e a GA\_Config\_4 destacaram-se com um Desvio Padrão da Fitness de **0.0**, indicando que convergiram para o mesmo valor ótimo em todas as 5 execuções.
* **Eficiência (Tempo Médio de Execução):**
  + Hill Climbing: ~0.47 segundos (mais rápido).
  + GA\_Config\_1: ~6.05 segundos.
  + GA\_Config\_4: ~8.19 segundos.
  + Simulated Annealing: ~17.60 segundos.
  + GA\_Config\_2 e GA\_Config\_3: ~61-62 segundos (mais lentas).

### 3.3. Conclusões Preliminares (5 Execuções)

Com base nos resultados de 5 execuções:

* O **Hill Climbing** mostrou-se extremamente rápido.
* O **Simulated Annealing** encontrou consistentemente a melhor solução, mas com custo de tempo moderado.
* A **GA\_Config\_4** emergiu como uma candidata forte, igualando o SA na melhor fitness e consistência, mas sendo mais rápida.
* A **GA\_Config\_1** foi notável pela sua rapidez entre os AGs, embora com menor consistência.

Estes resultados forneceram a base para selecionar algoritmos para a fase de 30 execuções.

## 4. Fase de Multiprocessamento - 30 Execuções com Algoritmos Promissores: Análise Estatística

Esta secção do projeto apresenta uma análise estatística detalhada dos quatro algoritmos identificados como mais promissores na fase anterior de 5 execuções (Hill Climbing, Simulated Annealing, GA\_Config\_1 e GA\_Config\_4). Cada algoritmo foi executado 30 vezes em paralelo para obter resultados estatisticamente mais robustos.

### 4.1. Metodologia

Os algoritmos foram executados com os seus parâmetros base (os mesmos da fase de 5 execuções). As métricas analisadas incluíram fitness média, desvio padrão da fitness, melhor fitness geral e tempo médio de execução.

### 4.2. Resultados e Análise Estatística (30 Execuções Iniciais)

* **Melhor Fitness Geral:** Todos os quatro algoritmos (HC, SA, GA\_Config\_1, GA\_Config\_4) conseguiram alcançar o mesmo valor de Melhor Fitness Geral de aproximadamente **0.0571**.
* **Fitness Média e Consistência:**
  + HC, SA, e GA\_Config\_4 apresentaram a mesma fitness média (0.0605) e o mesmo desvio padrão (0.0124).
  + GA\_Config\_1 teve um desempenho ligeiramente inferior em fitness média (0.0688) e maior variabilidade (desvio padrão 0.0210).
* **Eficiência (Tempo Médio de Execução):**
  + Hill Climbing: ~0.63 segundos (mais rápido).
  + GA\_Config\_1: ~5.97 segundos.
  + GA\_Config\_4: ~8.30 segundos.
  + Simulated Annealing: ~18.21 segundos (mais lento).

### 4.3. Conclusões (30 Execuções Iniciais)

* **Hill Climbing** emergiu como o algoritmo mais eficiente, alcançando a mesma qualidade de solução que SA e GA\_Config\_4, mas numa fração do tempo.
* HC, SA e GA\_Config\_4 foram igualmente consistentes.
* O HC ofereceu o melhor equilíbrio entre qualidade, consistência e eficiência.

Esta análise levou à decisão de focar a fase final de variação de parâmetros no Hill Climbing e nas duas configurações de AG mais promissoras (GA\_Config\_1 e GA\_Config\_4), descartando o SA devido à sua lentidão comparativa sem ganho de qualidade sobre o HC ou o GA\_Config\_4 otimizado.

## 5. Fase de Variação de Parâmetros - 30 Execuções por Variação: Análise Estatística Detalhada

Esta secção detalha a análise estatística dos resultados obtidos após a execução de 30 repetições para cada uma das 19 variações de parâmetros aplicadas ao Hill Climbing (HC) e às duas configurações de Algoritmos Genéticos (GA\_Config\_1 e GA\_Config\_4). O objetivo desta fase foi identificar as configurações de hiperparâmetros mais eficazes para cada algoritmo.

### 5.1. Justificação da Escolha dos Algoritmos Genéticos para Variação de Parâmetros

As configurações GA\_Config\_1\_SwapConst1PtPreferVTournVarK e GA\_Config\_4\_TargetExchUnifPreferVTournVarK\_k5 foram selecionadas para a fase de variação de parâmetros detalhada com base no seu desempenho promissor na ronda anterior de 30 execuções (com parâmetros base).

* GA\_Config\_1 foi a configuração de AG mais rápida nessa fase anterior que também demonstrou capacidade de alcançar a melhor fitness, embora com menor consistência que outras. A sua arquitetura, com mutate\_swap\_constrained e crossover\_one\_point\_prefer\_valid, representava uma abordagem mais clássica e menos disruptiva, cujo desempenho poderia ser afinado.
* GA\_Config\_4 destacou-se por alcançar a melhor fitness com uma consistência muito boa e um tempo de execução competitivo. A sua combinação de operadores (mutação direcionada mutate\_targeted\_player\_exchange e cruzamento uniforme crossover\_uniform\_prefer\_valid) sugeria um bom potencial para exploração e explotação do espaço de soluções, justificando uma investigação mais aprofundada dos seus parâmetros.

A expectativa era que, através da variação de parâmetros como taxa de mutação, tamanho da população e número de gerações, pudéssemos refinar ainda mais o desempenho destas duas arquiteturas de AG distintas.

### 5.2. Metodologia da Variação de Parâmetros

* **Hill Climbing (HC):** Variou-se o **Número Máximo de Iterações**: [500, 1000 (base), 1500].
* **GA\_Config\_1 e GA\_Config\_4:** Para cada uma, variou-se individualmente (mantendo os outros parâmetros base nos seus valores originais: PopSize=50, NumGen=100, MutRate=0.1):
  + **Taxa de Mutação (MutRate):** [0.05, 0.15, 0.25]
  + **Tamanho da População (PopSize):** [30, 75]
  + **Número de Gerações (NumGen):** [75, 150] Cada uma das 19 variações resultantes foi executada 30 vezes.

### 5.3. Resultados e Análise da Variação de Parâmetros

**Observação Chave Global:** Todas as 19 configurações testadas, ao longo das suas 30 execuções, foram capazes de encontrar pelo menos uma vez a mesma melhor fitness global de **0.0571**.

#### 5.3.1. Hill Climbing (HC) Otimizado

* **HC\_Iter\_500 (Máx. Iterações = 500):**
  + Fitness Média: 0.0588, Desvio Padrão: 0.0089, Tempo Médio: 0.41s.
* **Conclusão para HC:** A configuração HC\_Iter\_500 foi a mais eficaz, com a melhor fitness média, menor desvio padrão e tempo de execução extremamente baixo. Aumentar as iterações não trouxe melhorias.

#### 5.3.2. Algoritmo Genético - Configuração 1 (GA\_Config\_1) Otimizado

* **Melhores Variações:**
  + GA\_Config\_1\_mutation\_rate\_0.25 (Pop=50, Gen=100): Fitness Média: 0.0588, Desvio Padrão: 0.0089, Tempo: 6.13s.
  + GA\_Config\_1\_population\_size\_75 (MutRate=0.1, Gen=100): Fitness Média: 0.0588, Desvio Padrão: 0.0089, Tempo: 9.08s.
* **Conclusão para GA\_Config\_1:** Ambas as variações igualaram a qualidade e consistência do HC\_Iter\_500, mas com tempos de execução consideravelmente maiores. A variação com taxa de mutação de 0.25 foi mais eficiente em tempo.

#### 5.3.3. Algoritmo Genético - Configuração 4 (GA\_Config\_4) Otimizado

* **Melhores Variações (atingiram consistentemente fitness média de 0.0571 com desvio padrão ~0.0):**
  + GA\_Config\_4\_mutation\_rate\_0.15 (Pop=50, Gen=100): Tempo: 8.55s.
  + GA\_Config\_4\_mutation\_rate\_0.25 (Pop=50, Gen=100): Tempo: 8.90s.
  + GA\_Config\_4\_population\_size\_75 (MutRate=0.1, Gen=100): Tempo: 12.91s.
  + GA\_Config\_4\_generations\_150 (MutRate=0.1, Pop=50): Tempo: 12.44s.
* **Conclusão para GA\_Config\_4:** Demonstrou um potencial notável, com várias variações alcançando consistentemente a melhor fitness média de 0.0571 com desvio padrão nulo. A GA\_Config\_4\_mutation\_rate\_0.15 foi a mais eficiente em tempo entre estas configurações de topo.

### 5.4. Conclusões Gerais da Variação de Parâmetros

1. **Melhor Fitness Global Atingível:** 0.0571, alcançado por todas as variações.
2. **Hill Climbing Otimizado (HC\_Iter\_500):** Destacou-se pela extrema eficiência (0.41s), excelente fitness média (0.0588) e boa consistência.
3. **GA\_Config\_1 Otimizado (GA\_Config\_1\_mutation\_rate\_0.25):** Igualou a qualidade do HC otimizado, mas mais lento (6.13s).
4. **GA\_Config\_4 Otimizado (GA\_Config\_4\_mutation\_rate\_0.15):** Atingiu consistentemente a melhor fitness média (0.0571) com desvio padrão nulo, sendo a mais rápida (8.55s) entre as que alcançaram este nível de perfeição.

**Recomendação Final de Algoritmos e Configurações Pós-Variação:**

* **Para Eficiência Máxima com Excelente Qualidade:** HC\_Iter\_500.
* **Para Garantia da Melhor Fitness com Consistência Perfeita:** GA\_Config\_4\_mutation\_rate\_0.15.

## 6. Conclusões Finais e Recomendações

### 6.1. Síntese dos Resultados

Este projeto explorou a aplicação de algoritmos meta-heurísticos ao problema de otimização de seleção de equipas desportivas, progredindo desde implementações em processador único até análises detalhadas de variação de parâmetros. Os principais resultados podem ser sintetizados da seguinte forma:

1. **Otimizações de Código:** A redução do uso de deepcopy no Simulated Annealing resultou numa melhoria de desempenho de aproximadamente 51.7% no tempo total de execução.
2. **Multiprocessamento - 5 Execuções:** Todos os algoritmos conseguiram alcançar a mesma melhor fitness geral (0.057143), mas com diferenças significativas em consistência e tempo de execução. O Hill Climbing foi o mais rápido (0.47s), enquanto o Simulated Annealing e a GA\_Config\_4 foram os mais consistentes.
3. **Multiprocessamento - 30 Execuções:** Hill Climbing, Simulated Annealing e GA\_Config\_4 apresentaram desempenhos estatisticamente equivalentes em termos de qualidade da solução (fitness média 0.0605, desvio padrão 0.0124), mas com o Hill Climbing sendo significativamente mais rápido (0.63s vs. 18.21s para SA).
4. **Variação de Parâmetros - 30 Execuções por Variação:**
   * O Hill Climbing com 500 iterações (HC\_Iter\_500) emergiu como a configuração mais eficiente, com excelente fitness média (0.0588), boa consistência (desvio padrão 0.0089) e tempo extremamente baixo (0.41s).
   * A GA\_Config\_4 com taxa de mutação de 0.15 (GA\_Config\_4\_mutation\_rate\_0.15) alcançou consistentemente a melhor fitness média (0.0571) com desvio padrão nulo, sendo a mais rápida (8.55s) entre as configurações que garantiram este nível de perfeição.

### 6.2. Recomendações para Aplicação Prática

Com base nos resultados obtidos, recomendamos:

1. **Para Cenários com Restrições de Tempo:** Utilizar o Hill Climbing com 500 iterações (HC\_Iter\_500), que oferece um excelente equilíbrio entre qualidade da solução e eficiência computacional.
2. **Para Cenários que Exigem Garantia da Melhor Solução:** Utilizar o Algoritmo Genético GA\_Config\_4 com taxa de mutação de 0.15 (GA\_Config\_4\_mutation\_rate\_0.15), que garante consistentemente a melhor fitness possível, embora com um custo computacional mais elevado.
3. **Para Aplicações de Larga Escala:** Considerar a implementação de uma abordagem híbrida, iniciando com Hill Climbing para obter rapidamente uma boa solução, seguido de refinamento com GA\_Config\_4 otimizado para casos específicos onde a qualidade máxima é crítica.

### 6.3. Sugestões para Trabalhos Futuros

1. **Exploração de Hibridização de Algoritmos:** Investigar a combinação de Hill Climbing com Algoritmos Genéticos, potencialmente utilizando HC para refinar soluções geradas por AGs.
2. **Análise de Escalabilidade:** Testar o desempenho dos algoritmos otimizados em problemas de maior dimensão, com mais equipas ou mais jogadores.
3. **Paralelização Interna dos Algoritmos:** Além do multiprocessamento para múltiplas execuções, explorar a paralelização interna dos algoritmos, especialmente para os Algoritmos Genéticos (avaliação paralela de indivíduos).
4. **Exploração de Técnicas de Machine Learning:** Investigar o uso de técnicas de aprendizagem para prever boas configurações de parâmetros com base nas características do problema.

## 7. Diagrama do Processo de Análise

O diagrama abaixo ilustra o processo de análise seguido neste projeto, desde a implementação inicial até às recomendações finais:

┌─────────────────────────┐  
│ Implementação Inicial │  
│ (Processador Único) │  
└───────────┬─────────────┘  
 ▼  
┌─────────────────────────┐  
│ Profiling e Otimização │  
│ - Redução de deepcopy │  
│ - Vetorização │  
└───────────┬─────────────┘  
 ▼  
┌─────────────────────────┐  
│ Multiprocessamento │  
│ (5 Execuções) │  
│ - HC, SA, 4 configs AG │  
└───────────┬─────────────┘  
 ▼  
┌─────────────────────────┐  
│ Seleção de Algoritmos │  
│ Promissores │  
│ - HC, SA, GA1, GA4 │  
└───────────┬─────────────┘  
 ▼  
┌─────────────────────────┐  
│ Análise Estatística │  
│ (30 Execuções) │  
└───────────┬─────────────┘  
 ▼  
┌─────────────────────────┐  
│ Seleção para Variação │  
│ de Parâmetros │  
│ - HC, GA1, GA4 │  
└───────────┬─────────────┘  
 ▼  
┌─────────────────────────┐  
│ Variação de Parâmetros │  
│ (30 Execuções/Variação) │  
│ - 19 configurações │  
└───────────┬─────────────┘  
 ▼  
┌─────────────────────────┐  
│ Análise Final e │  
│ Recomendações │  
│ - HC\_Iter\_500 │  
│ - GA4\_MutRate\_0.15 │  
└─────────────────────────┘

Este processo metodológico permitiu uma exploração sistemática do espaço de algoritmos e parâmetros, culminando em recomendações bem fundamentadas para a aplicação prática.

## 6. Análise Quantitativa Consolidada e Recomendações Finais

Para complementar as análises detalhadas apresentadas nas secções anteriores, foi elaborado um documento dedicado à **Análise Quantitativa Detalhada dos Algoritmos do Projeto CIFO EXTENDED**. Este documento (quantitative\_analysis/pt/quantitative\_analysis\_pt.md) consolida e compara numericamente o desempenho dos algoritmos em todas as fases de experimentação, focando-se em métricas chave como fitness média, desvio padrão da fitness, melhor fitness geral e tempo médio de execução. As tabelas comparativas e as discussões aprofundadas presentes nesse documento fornecem a base numérica para as conclusões e recomendações que se seguem.

### 6.1. Síntese dos Resultados Quantitativos

A análise quantitativa global, conforme detalhado no documento mencionado, reforça várias observações chave:

* **Consistência na Obtenção da Melhor Solução:** Diversas configurações, incluindo o HC\_MaxIter\_500 e as versões otimizadas de GA\_Config\_1 e GA\_Config\_4, demonstraram capacidade de atingir consistentemente a melhor fitness conhecida de **0.057143**.
* **Eficiência Superior do Hill Climbing:** O HC\_MaxIter\_500 emergiu como o algoritmo mais rápido, encontrando a solução ótima em aproximadamente **0.44 segundos**, significativamente mais rápido que os Algoritmos Genéticos.
* **Desempenho dos Algoritmos Genéticos Otimizados:** Configurações como GA\_Config\_1\_mutation\_rate\_0.25 (aprox. 6.25s) e GA\_Config\_4\_mutation\_rate\_0.15 (aprox. 8.67s) também alcançaram a melhor fitness com consistência perfeita, embora com maior custo temporal. Estes oferecem uma exploração mais ampla do espaço de soluções.
* **Impacto da Otimização de Parâmetros:** A fase de variação de parâmetros foi essencial para refinar o desempenho dos AGs, com taxas de mutação mais elevadas e populações maiores geralmente resultando em melhor qualidade e consistência da solução.

### 6.2. Recomendações Finais

Com base na totalidade do trabalho desenvolvido e nos resultados quantitativos consolidados:

1. **Para Cenários com Prioridade na Velocidade:** Se o tempo de execução for o fator crítico e a solução ótima conhecida for aceitável, o **Hill Climbing com 500 iterações (HC\_MaxIter\_500)** é a recomendação primária devido à sua excecional rapidez e capacidade de encontrar consistentemente a melhor solução.
2. **Para Cenários que Exigem Máxima Qualidade e Exploração:** Se for necessária uma exploração mais exaustiva do espaço de soluções, com a garantia de atingir a melhor fitness possível, e o tempo de execução for uma consideração secundária, as configurações otimizadas dos Algoritmos Genéticos são preferíveis. Especificamente, **GA\_Config\_4 com taxa de mutação de 0.15 e GA\_Config\_1 com taxa de mutação de 0.25** são recomendadas, oferecendo um bom equilíbrio entre a qualidade da solução e um tempo de execução ainda gerível.
3. **Trabalhos Futuros:**
   * Explorar técnicas híbridas que combinem a velocidade do Hill Climbing com a capacidade de exploração dos Algoritmos Genéticos.
   * Investigar o impacto de diferentes representações de solução ou funções de fitness mais complexas.
   * Aplicar os algoritmos otimizados a conjuntos de dados maiores ou com diferentes características para avaliar a sua escalabilidade e generalização.
   * Desenvolver interfaces de utilizador mais amigáveis para facilitar a configuração e execução das simulações.

Este projeto demonstrou a eficácia das meta-heurísticas na resolução do problema de formação de equipas desportivas e forneceu um pipeline de análise robusto, cujos resultados e processos são detalhadamente documentados e reproduzíveis através dos notebooks fornecidos para cada fase.

(Secção de Apêndices e Diagrama do Processo de Análise a serem adicionados/referenciados posteriormente, conforme o plano.)

## Apêndices

Esta secção contém os documentos de suporte detalhados que foram produzidos e analisados durante o projeto CIFO EXTENDED.

### Apêndice A: Documentos da Fase de Processador Único (SP)

* Relatório da Fase de Processador Único: Arquitetura, Algoritmos e Operadores (reports\_sp\_phase/pt/01\_sp\_architecture\_algorithms\_operators.md)
* Documento de Arquitetura da Fase SP (reports\_sp\_phase/pt/00\_sp\_architecture.md)
* Documento de Revisão de Código da Fase SP (reports\_sp\_phase/pt/00\_sp\_code\_review.md)

### Apêndice B: Documentos da Fase de Multiprocessamento (MP)

* Relatório da Fase MP - Análise de 5 Execuções (reports\_mp\_phase/pt/02\_mp\_5\_runs\_analysis.md)
* Relatório da Fase MP - Análise Estatística de 30 Execuções (reports\_mp\_phase/pt/03\_mp\_30\_runs\_statistical\_analysis.md)
* Relatório da Fase MP - Análise da Variação de Parâmetros (reports\_mp\_phase/pt/04\_mp\_param\_var\_analysis.md)
* Relatório da Fase MP - Análise Estatística da Variação Final de Parâmetros (reports\_mp\_phase/pt/05\_mp\_final\_param\_var\_statistical\_analysis.md)
* Documento de Arquitetura da Fase MP (reports\_mp\_phase/pt/00\_mp\_architecture.md)
* Documento de Revisão de Código da Fase MP (reports\_mp\_phase/pt/00\_mp\_code\_review.md)

### Apêndice C: Análise Quantitativa e Dados

* Documento de Análise Quantitativa Detalhada (quantitative\_analysis/pt/quantitative\_analysis\_pt.md)
* Sumário dos Resultados - 5 Execuções MP (images\_mp/run\_5\_results/all\_algorithms\_summary\_mp\_5runs.csv)
* Sumário dos Resultados - 30 Execuções MP (Promissores) (images\_mp/run\_30\_results\_promising/all\_algorithms\_summary\_mp\_30runs.csv)
* Sumário dos Resultados - Variação de Parâmetros GA (images\_mp/param\_var\_results/all\_ga\_variations\_summary\_mp\_5runs.csv)
* Sumário dos Resultados - Variação Final de Parâmetros (30 Execuções) (images\_mp/final\_param\_var\_results/all\_algorithms\_summary\_final\_param\_var\_30runs.csv)

### Apêndice D: Notebooks de Análise (Localização)

* Todos os notebooks Jupyter utilizados para análise e geração de resultados estão localizados na diretoria /notebooks/ do repositório do projeto.
  + notebooks/01\_single\_processor\_analysis.ipynb (Exemplo, será atualizado/criado)
  + notebooks/02\_multiprocessing\_5\_runs\_analysis.ipynb (Exemplo, será atualizado/criado)
  + notebooks/03\_multiprocessing\_30\_runs\_analysis.ipynb (Exemplo, será criado)
  + notebooks/04\_multiprocessing\_param\_var\_analysis.ipynb (Exemplo, será criado)
  + notebooks/05\_multiprocessing\_final\_param\_var\_analysis.ipynb (Exemplo, será criado)

### Apêndice E: Diagrama do Processo de Análise

* (Referência à imagem do diagrama do processo de análise, que será incluída ou referenciada aqui)