Instituto Superior de Engenharia

Politécnico de Coimbra

Relatório do Trabalho Prático nº 2 de Introdução à Inteligência Artificial

Fábio Oliveira (2022145902)
Rafael Filipe Rodrigues Pereira (2022150534)

Licenciatura em Engenharia Informática

Departamento de Engenharia Informática e Sistemas

Instituto Superior de Engenharia de Coimbra

Coimbra, 15 de dezembro de 2024

Índice

Índice de Figuras

Índice de Tabelas

1 Introdução

Este trabalho prático consiste em desenvolver e testar um conjunto de algoritmos de otimização para o *Coin Change Problem*.

O objetivo principal é garantir soluções de qualidade para os diferentes desafios propostos.

Para concretizar o objetivo do trabalho, é necessário minimizar ao máximo o número de moedas utilizadas para atingir um valor objetivo.

2 Métodos de otimização

2.1 Algoritmo de pesquisa local

Para o algoritmo de pesquisa local foi escolhido o trepa-colinas. O funcionamento deste conta com uma função de avaliação que utiliza penalização ou reparação (escolha do utilizador) para lidar com soluções inválidas e dois tipos de vizinhanças (também escolha do utilizador).

2.1.1 Penalização

- Calcula o valor total da solução e o número de moedas utilizadas.
- Aplica uma penalização pesada caso o valor total seja diferente do valor objetivo.
- Devolve o número de moedas ajustado pela penalização, garantindo assim que soluções mais próximas do valor objetivo tenham uma penalização menor.

2.1.2 Reparação

- Calcula o valor total e tenta ajustar, ou seja, reparar, a solução para que esta atinja o valor objetivo.
  - Se o valor total for superior ao valor objetivo, reduz o número de moedas.
  - Se o valor total for inferior ao valor objetivo, aumenta o número de moedas.
- Penaliza soluções que, mesmo após a reparação, não alcancem o valor objetivo.

2.1.3 Vizinhanças

Cria uma nova solução a partir de uma solução existente.

- **CHANGE_ONE** - altera uma única moeda, aumentando ou diminuindo (aleatório).
- **CHANGE_TWO** - altera duas moedas em simultâneo, reduzindo uma e aumentando a outra.

2.2 Algoritmo Evolutivo

O algoritmo evolutivo foi desenvolvido para resolver o *Coin Change Problem* de forma a minimizar o número de moedas utilizadas. Os principais componentes são:

- **População inicial**: Indivíduos gerados aleatoriamente.
- **Seleção por torneio**: Seleciona $tsize$ indivíduos e escolhe o melhor (menor *fitness*).
- **Operadores genéticos**:
  - Recombinação de 1 ponto de corte (*Crossover*): Divide cromossomas dos pais e num ponto e combina segmentos.
  - Recombinação uniforme: Para cada gene, escolhe aleatoriamente o pai.
  - Mutação binária: Modifica aleatoriamente a quantidade de uma moeda.
  - Mutação por troca: Permuta os valores de duas posições no cromossoma.
- **Funções de penalização e reparação**: Garantem soluções viáveis. Foram herdadas do trepa-colinas.

2.2.1 Representação da Solução

Cada indivíduo é um array onde cada posição i indica a quantidade da moeda i. Por exemplo, para moedas $\{1, 0.5, 2\}$, o cromossoma $[3, 2, 1]$ representa a solução com 3 moedas de 1, 2 moedas de 0.5 e 1 moeda de 2. A função objetivo avalia a soma total gerada e o número de moedas usadas.

2.2.2 Parâmetros Principais

Os principais parâmetros são:

- Tamanho da população (popsize);
- Probabilidades de recombinação (pr) e mutação (pm);
- Tamanho do torneio (tsize);
- Número de gerações (numGenerations).

A combinação de operadores genéticos e funções de reparação torna o algoritmo eficiente para encontrar soluções viáveis e otimizadas.

2.3 Abordagem híbrida

A abordagem híbrida implementada combina as vantagens do algoritmo evolutivo com o método de trepa colinas, procurando obter melhores resultados através da sinergia entre pesquisa global e local.

O algoritmo híbrido opera em três fases distintas:

1. **Fase Inicial (Opcional)**
    - Gera-se uma população inicial aleatória
    - Aplica-se o algoritmo trepa colinas a cada indivíduo da população inicial
    - Esta fase permite começar com soluções localmente optimizadas
2. **Fase Evolutiva**
    - Executa-se o algoritmo evolutivo normal
    - Utiliza operadores genéticos (recombinação e mutação)
    - Realiza selecção por torneio
    - Decorre durante um número predefinido de gerações
3. **Fase Final (Opcional)**
    - Aplica-se o trepa colinas à população final
    - Permite refinar as melhores soluções encontradas
    - Otimização local das soluções promissoras

2.3.1 Vantagens da Abordagem

- **Exploração Global:** O algoritmo evolutivo permite explorar diversas regiões do espaço de pesquisa
- **Refinamento Local:** O trepa colinas otimiza localmente as soluções promissoras
- **Flexibilidade:** Permite ativar ou desativar as fases de otimização local
- **Equilíbrio:** Balanceia a diversificação da pesquisa com a intensificação local

3 Estudo experimental

3.1 Algoritmo de pesquisa local

Ficheiro	Métrica	100 it	1000 it	5000 it	10000 it
`file1.txt`	Melhor	2	2	2	2
	MBF	2	2	2	2
`file2.txt`	Melhor	20	20	20	20
	MBF	24.6	22.1	22.1	20.9
`file3.txt`	Melhor	102	102	102	102
	MBF	112.8	102.8	103.7	103.3
`file4.txt`	Melhor	861	276	108	108
	MBF	1277.9	933.7	564.3	205.8
`file5.txt`	Melhor	4	4	4	4
	MBF	4	4	4	4
Média/iterações		241.03	146.86	93.21	57.2
Média final: 134.575					

Tabela 1: Resultados do Trepa-Colinas com Vizinhança 2 + Reparação + Soluções de mesmo custo

A tabela apresenta a melhor configuração para o trepa-colinas, em comparação com as restantes (ver ficheiro excel). Os melhores resultados foram obtidos com 10000 iterações por repetição. Sendo assim, na abordagem híbrida será esse o valor utilizado quando o trepa-colinas for utilizado. A média final da tabela serviu para comparar com as restantes configurações testadas para concluir que esta era a melhor configuração para o trepa-colinas.

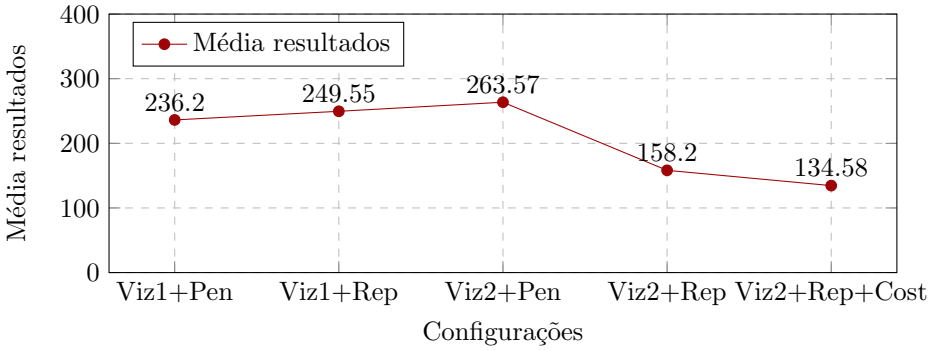

Figura 1: Média de resultados entre configurações para o trepa-colinas

O gráfico apresenta a comparação entre cinco variantes do trepa-colinas. As variantes incluem o uso de diferentes estratégias de vizinhança (Vizinhança 1 e Vizinhança 2), combinadas com métodos de penalização ou reparação para lidar com soluções inválidas. A última configuração, que incorpora Vizinhança 2, reparação e aceitação de soluções com o mesmo custo, demonstrou o melhor desempenho médio.

3.2 Algoritmo evolutivo

Os gráficos comparam quatro configurações:

a) Algoritmo base: Recombinação de 1 ponto de corte + Mutação binária + Penalização.

b) Recombinação de 1 ponto de corte + Mutação binária + Reparação.

c) Recombinação de 1 ponto de corte + Mutação por troca + Reparação.

d) Recombinação uniforme + Mutação por troca + Reparação.

3.2.1 file1.txt

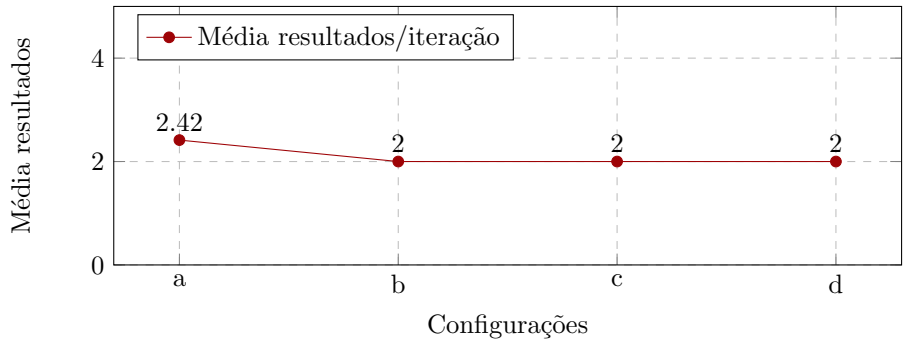

Figura 2: Média de resultados entre configurações evolutivas para file1.txt

O gráfico compara o desempenho de quatro variantes do algoritmo evolutivo aplicadas ao *file1.txt*.

Analisando os resultados notamos que, o desempenho uniforme reflete a simplicidade do problema e a baixa diversidade de soluções possíveis, já que a limitação no número de moedas facilita a convergência para a solução ótima.

A razão para a última combinação ser escolhida para realizar o método híbrido para este ficheiro, foi que esta última combinação foi a que demonstrou os melhores resultados para a maioria dos ficheiros.

3.2.2 file2.txt

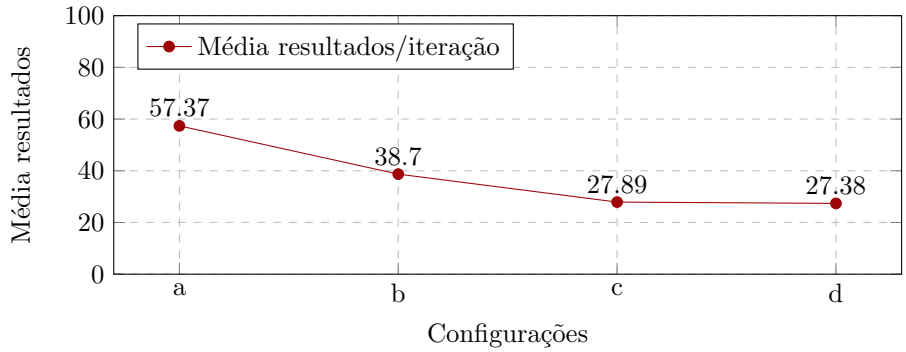

Figura 3: Média de resultados entre configurações evolutivas para file2.txt

Analisaremos agora o desempenho das quatro variantes do Algoritmo Evolutivo aplicadas ao *file2.txt*.

Com uma maior variabilidade em relação ao *file1.txt*, os últimos dois métodos (c e d) apresentaram os melhores resultados, com o último sendo ligeiramente melhor. Por essa razão, a última configuração foi escolhida para o Método Híbrido deste ficheiro.

Este desempenho reflete a eficácia da **Mutação por Troca** e da **Recombinação Uniforme** em problemas com uma diversidade de combinações média.

3.2.3 file3.txt

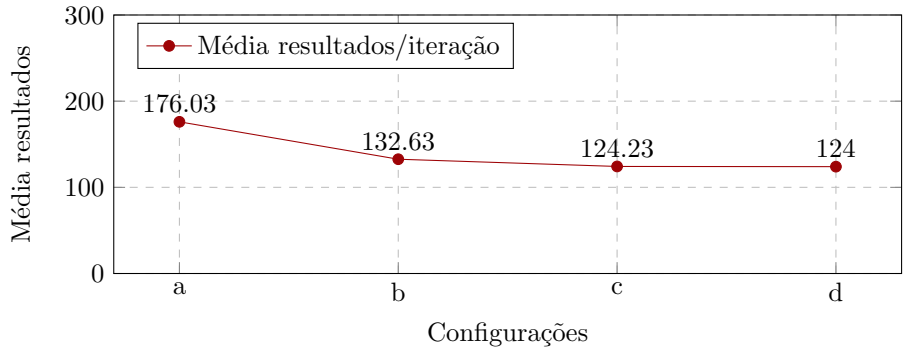

Figura 4: Média de resultados entre configurações evolutivas para file3.txt

Já no gráfico do ficheiro 3, este apresenta os resultados obtidos ao aplicar as quatro variantes do algoritmo evolutivo ao *file3.txt*.

Neste cenário, a diversidade das combinações possíveis foi ainda maior do que as no *file2.txt*, permitindo observar melhor a diferença de desempenho entre os métodos. O último método **(Recombinação uniforme + Mutação por troca + Reparação)** destacou-se como o mais eficaz, obtendo os melhores resultados. Apesar de sua vantagem sobre o **Recombinação de 1 ponto de**

corte + Mutação por troca + Reparação ser pequena, ela é suficiente para considerá-lo como o método preferido, reforçando sua escolha para integrar o Modo Híbrido.

Esses resultados evidenciam mais uma vez a robustez da **Recombinação Uniforme** aliada à **Mutação por Troca** em cenários com variabilidade média.

3.2.4 file4.txt

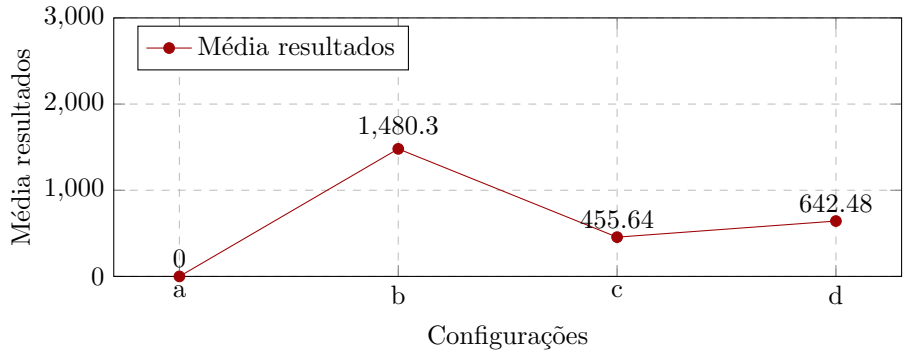

Figura 5: Média de resultados entre configurações evolutivas para file4.txt

Chegando ao gráfico do ficheiro 4, é possível analisar os resultados obtidos ao aplicar três das quatro variantes do algoritmo evolutivo ao *file4.txt*. Este cenário proporcionou a maior variação de resultados entre todos os ficheiros, devido à complexidade aumentada pela diversidade de moedas e pela precisão do objetivo.

Nesta experiência, o método **Recombinação de 1 ponto de corte + Mutação por troca + Reparação** destacou-se significativamente, apresentando uma diferença clara em relação aos outros métodos, consolidando-o como o mais eficaz neste contexto.

Assim, este foi o método escolhido para integrar o Modo Híbrido, evidenciando a sua superioridade em situações mais complexas e com maior variabilidade. O gráfico reflete essa diferença, demonstrando o desempenho superior do método escolhido em comparação às demais abordagens.

NOTA: A razão da primeira combinação de métodos estar a zero é devido aos resultados inválidos que obtivemos e invalidam a amostra de resultados.

3.2.5 file5.txt

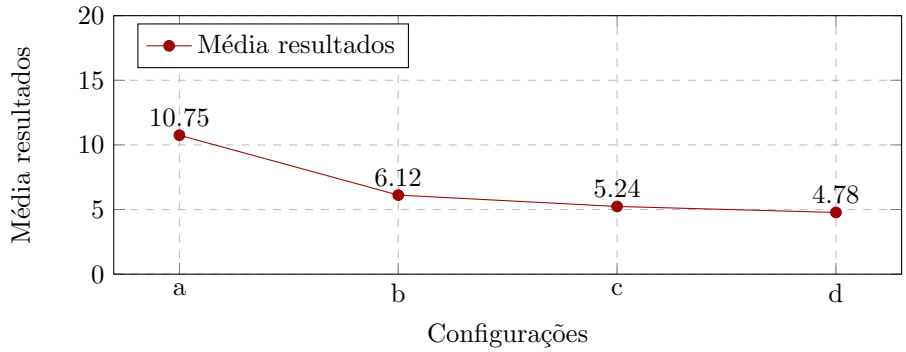

Figura 6: Média de resultados entre configurações evolutivas para file5.txt

Por fim temos gráfico que apresenta os resultados do *file5.txt*.

Apesar do objetivo ser pequeno, a quantidade de moedas adicionou complexidade ao problema, possibilitando uma ampla gama de soluções possíveis.

O método da **Recombinação Uniforme + Mutação por Troca + Reparação** destacou-se mais uma vez, apresentando os melhores resultados em comparação com as outras variantes. Este método, mais uma vez demonstra a sua capacidade de encontrar melhores soluções lidando com uma grande diversidade de moedas.

Assim como em outros cenários, esse método foi o escolhido para o Método Híbrido.

3.3 Abordagem híbrida

Os gráficos compara quatro configurações:

i) Algoritmo base híbrido - População inicial.

ii) Algoritmo base híbrido - População final.

iii) Algoritmo Base híbrido - Ambas.

3.3.1 file1.txt

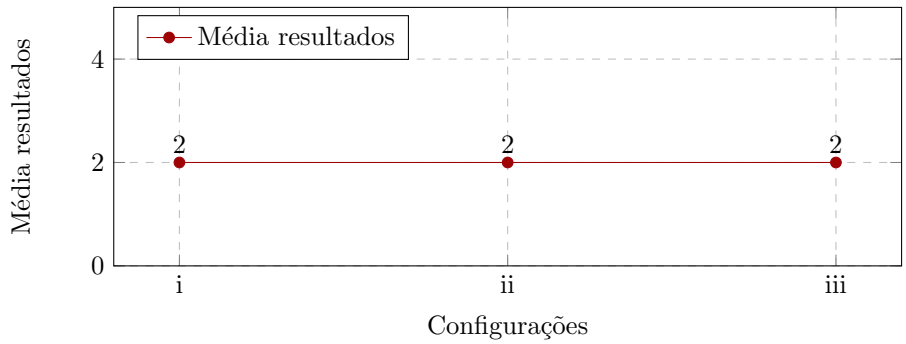

Figura 7: Média de resultados entre configurações híbridas para file1.txt

3.3.2 file2.txt

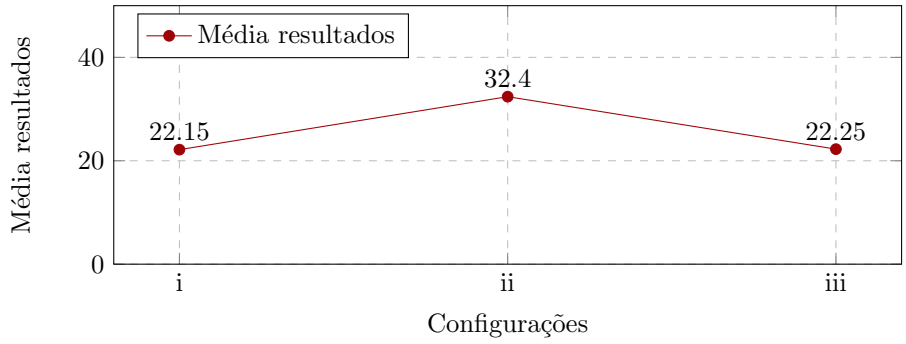

Figura 8: Média de resultados entre configurações híbridas para file2.txt

3.3.3 file3.txt

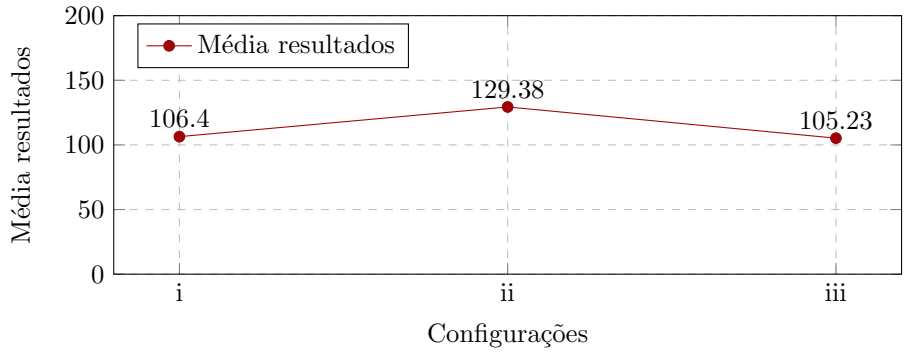

Figura 9: Média de resultados entre configurações híbridas para file3.txt

3.3.4 file4.txt

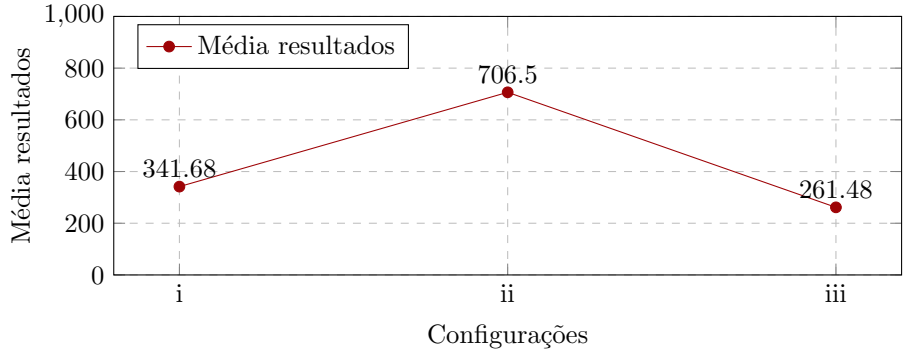

Figura 10: Média de resultados entre configurações híbridas para file4.txt

3.3.5 file5.txt

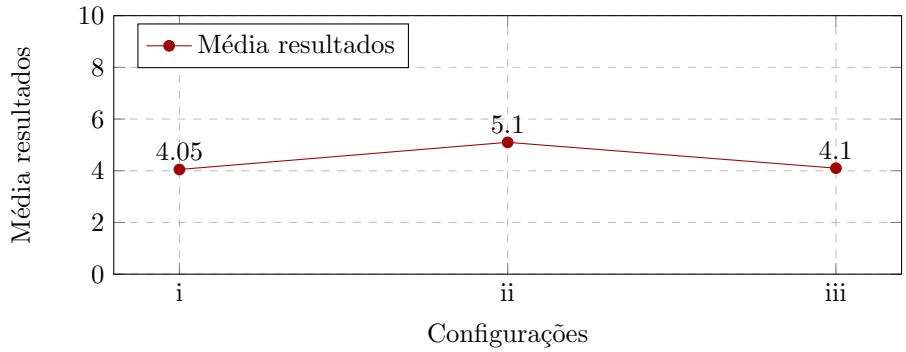

Figura 11: Média de resultados entre configurações híbridas para file5.txt

O método híbrido combinou os melhores elementos dos Métodos Evolutivos e do Trepa-Colinas, integrando as suas técnicas como a **Geração de Vizinhos 2**, a **Reparação das Soluções** e a **permissão para soluções iguais**. A combinação dessas abordagens, adaptadas a cada cenário específico, levou a uma performance superior em todas as experiências realizadas.

No *file1.txt*, o simples objetivo e o número reduzido de moedas não geraram variações significativas nos resultados, uma vez que três dos métodos atingiram resultados perfeitos.

No entanto, à medida que a complexidade aumentava nos outros ficheiros, o Método Híbrido destacou-se de forma clara e incontestável, superando os outros métodos em eficiência e precisão.

Nos *files2.txt, file3.txt, file4.txt, e file5.txt*, onde o número de moedas aumenta e o valor do objetivo é mais complexo, o híbrido demonstrou de forma consistente o porquê de ser o melhor, atingindo os melhores resultados em todos os cenários testados.

Ao combinar as melhores técnicas de cada ficheiro, o Método Híbrido solidificou seu lugar como o **método de melhor desempenho**.

4 Conclusão

Este trabalho abordou a implementação e avaliação de diferentes métodos de otimização para resolver o problema de encontrar combinações de moedas que somem a um valor alvo.

Através da experimentação com **Algoritmos Evolutivos** e o método do **Trepa-Colinas**, foi possível observar como cada abordagem se comporta em cenários de diferentes complexidades.

Os resultados demonstraram que, embora os **Métodos Evolutivos** e de **Trepa-Colinas** tenham mostrado eficiência em várias configurações, o **Método Híbrido**, que combinou as melhores características de ambas as abordagens, superou claramente os outros.

Em termos de análise, observou-se que, nos casos de problemas simples (como o *file1.txt*), onde o número de moedas era baixo e o objetivo simples, todos os métodos tiveram resultados próximos, mas foi em cenários mais complexos que o **Híbrido** realmente se destacou.

Em suma, o **Método Híbrido** mostrou ser a abordagem mais eficaz para o problema em questão. Este trabalho contribuiu para a compreensão e aplicação de técnicas de otimização em problemas combinatórios, com implicações tanto para a teoria dos Algoritmos Evolutivos quanto para sua aplicação prática em problemas de otimização de moedas e outras áreas relacionadas.