Instituto Superior
de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE / DEPARTMENT OF
INFORMÁTICA E SISTEMAS

Trabalho Prático nº02 – Introdução à Inteligência Artificial

Escolha um item. em / in Engenharia Informática

Autor / Author

José Xavier

António Pedroso

Unidade Curricular

Introdução à Inteligência Artificial

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, dezembro 2024

1 INTRODUÇÃO

O problema **Coin Change** é um desafio clássico de otimização que procura determinar o número mínimo de moedas necessário para atingir um valor-alvo específico, utilizando um conjunto pré-definido de denominações. Este problema é particularmente relevante no estudo da otimização computacional e dos algoritmos devido à sua aplicabilidade em cenários reais, como transações financeiras e alocação de recursos, bem como à sua complexidade combinatória.

O objetivo deste trabalho é implementar e analisar algoritmos de otimização capazes de fornecer soluções de alta qualidade para diferentes instâncias do problema **Coin Change**. Dado um conjunto de denominações de moedas e um valor-alvo, a tarefa consiste em explorar e identificar soluções válidas que minimizem o número de moedas utilizadas. No caso de soluções inválidas, em que a soma das moedas não corresponde ao valor-alvo, estas são penalizadas ou reparadas, garantindo um mecanismo robusto de avaliação.

Este projeto envolve a implementação e análise comparativa de três abordagens distintas de otimização:

1. **Algoritmo de Pesquisa Local:** Baseado na melhoria iterativa de soluções através da exploração de configurações vizinhas.
2. **Algoritmo Evolutivo:** Inspirado em princípios genéticos, utilizando operadores de recombinação e mutação para gerar novas soluções.
3. **Métodos Híbridos:** Combinam as vantagens das abordagens de pesquisa local e evolutiva, visando alcançar um desempenho superior na otimização.

O estudo experimental foca-se na avaliação da eficácia dos algoritmos implementados através de testes em instâncias diversificadas do problema. Os principais indicadores de desempenho incluem a qualidade das soluções, a eficiência computacional e a robustez face a diferentes parâmetros. Os resultados obtidos pretendem fornecer insights sobre a adequação e adaptabilidade das abordagens desenvolvidas na resolução de problemas combinatórios de otimização como o Coin Change.

2 DESCRIÇÃO DAS 5 INSTÂNCIAS FORNECIDAS

Os ficheiros fornecidos juntamente com o enunciado contêm diferentes instâncias do problema Coin Change. Cada instância apresenta um conjunto de moedas disponíveis e um valor-alvo a ser alcançado. Estas instâncias servem para testar a capacidade dos algoritmos de otimização em encontrar soluções eficientes. Segue a descrição de cada uma:

2.1 Ficheiro file1.txt

- **Descrição:** Uma instância simples com 3 tipos de moedas.
- **Valor-alvo: 1**.0 euro.
- **Moedas disponíveis:** {0.05, 0.2, 0.5}.
- **Complexidade:** Baixa. Devido ao número reduzido de moedas e ao valor-alvo pequeno, esta instância é adequada para validação inicial dos algoritmos.

2.2 Ficheiro file2.txt

- **Descrição:** Uma instância com 4 tipos de moedas.
- **Valor-alvo:** 10.0 euros.
- **Moedas disponíveis:** {0.02, 0.1, 0.2, 0.5}.
- **Complexidade:** Média. Com um número maior de combinações possíveis e um valor-alvo maior, esta instância requer maior capacidade de otimização.

2.3 Ficheiro file3.txt

- **Descrição:** Uma instância com 3 tipos de moedas.
- **Valor-alvo:** 100.6 euros.
- **Moedas disponíveis:** {0.1, 0.5, 1.0}.
- **Complexidade:** Alta. O valor-alvo elevado aumenta o espaço de soluções, exigindo maior precisão dos algoritmos para encontrar soluções ótimas.

2.4 Ficheirofile4.txt

- **Descrição:** Uma instância com 7 tipos de moedas.

- **Valor-alvo:** 105.56 euros.
- **Moedas disponíveis:** {0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0}.
- **Complexidade:** Muito alta. A diversidade de moedas combinada com um valor-alvo elevado resulta em um espaço de busca significativamente maior, desafiando a eficiência dos algoritmos.

2.5 Ficheiro file5.txt

- **Descrição:** Uma instância com 8 tipos de moedas.
- **Valor-alvo:** 0.66 euro.
- **Moedas disponíveis:** {0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0}.
- **Complexidade:** Moderada. Apesar do valor-alvo pequeno, a quantidade de moedas disponíveis cria várias combinações possíveis, o que torna esta instância interessante para análise.

2.6 Finalidade das Instâncias

Estas instâncias abrangem diferentes níveis de complexidade e características. Elas foram concebidas para avaliar:

- A precisão dos algoritmos em encontrar soluções válidas.
- A eficiência na minimização do número de moedas.
- A robustez face a variações no número de moedas e no valor-alvo.

Estes cenários permitem uma análise detalhada do desempenho dos métodos implementados, fornecendo uma base sólida para comparações e conclusões sobre a eficácia das abordagens propostas.

3 FUNÇÃO DE AVALIAÇÃO

A função **calculaFit** avalia a qualidade de uma solução para o problema Coin Change, considerando dois aspetos principais:

1. **Validade da solução:** Verifica se o valor total das moedas na solução corresponde ao valor-alvo (valorObjetivo).
2. **Qualidade da solução válida:** Avalia o número de moedas usadas, sendo soluções com menos moedas consideradas melhores.

3.1 Funcionamento

- A função calcula:
  - O **valor total** da solução, com base no número de moedas utilizadas e seus respetivos valores.
  - O **número total de moedas** usadas na solução.

  - Se o valor total for igual ao valor-alvo, a função retorna o número de moedas usadas.
  - Caso contrário, a solução é penalizada com:
  - Um valor base elevado (***10000.0f***) para diferenciar soluções inválidas.
  - A diferença absoluta entre o valor total e o valor-alvo, multiplicada por um fator grande para priorizar soluções mais próximas do objetivo.

3.2 Objetivo

Guiar os algoritmos de otimização a:

- Encontrar soluções válidas que atinjam o valor-alvo.
- Minimizar o número de moedas utilizadas.
- Evitar soluções inválidas ou menos eficientes.

Esta função é essencial para determinar a direção e eficácia do processo de otimização no problema.

4 ALGORITMOS E HEURÍSTICAS UTILIZADAS

4.1 Trepa-Colinas

O algoritmo de trepa colinas (***hill climbing***) é uma técnica de otimização heurística utilizada para encontrar uma solução satisfatória para um problema, especialmente em espaços de busca grandes e complexos. Este algoritmo é inspirado no processo de escalar uma colina, onde o objetivo é alcançar o ponto mais alto (ou o ponto de menor custo, dependendo do problema).

4.1.1 Funcionamento do Algoritmo

1. **Solução Inicial:** O algoritmo começa com uma solução inicial, que pode ser gerada aleatoriamente ou de forma heurística.
2. **Avaliação da Solução:** A solução atual é avaliada utilizando uma função de custo ou fitness, que quantifica a qualidade da solução.
3. **Geração de Vizinhos:** São geradas soluções vizinhas, que são pequenas modificações da solução atual.
4. **Seleção do Vizinho:** O vizinho com o melhor custo (ou fitness) é selecionado como a nova solução atual.
5. **Iteração:** O processo de avaliação e geração de vizinhos é repetido por um número fixo de iterações ou até que uma condição de parada seja satisfeita (por exemplo, não haver melhoria após várias iterações).

4.1.2 Vantagens e Desvantagens

4.1.2.1 Vantagens:

- **Simplicidade:** O algoritmo é fácil de implementar e entender.
- **Eficiência:** Pode encontrar soluções satisfatórias rapidamente para problemas de grande escala.

4.1.2.2 Desvantagens:

- **Ótimos Locais:** O algoritmo pode ficar preso em ótimos locais, que são soluções que são melhores do que as vizinhas, mas não são a melhor solução global.
- **Dependência da Solução Inicial:** A qualidade da solução final pode depender fortemente da solução inicial escolhida.

4.2 Algoritmo Evolutivo

O **algoritmo evolutivo** é uma técnica de otimização inspirada nos princípios da **evolução natural**. Esta abordagem trabalha com uma **população de soluções** que evolui ao longo de várias gerações, através da aplicação de operadores genéticos como seleção, crossover e mutação. O objetivo é encontrar soluções de qualidade superior de forma iterativa, utilizando princípios de seleção natural.

4.2.1 Métodos de Seleção

No presente trabalho, foram utilizados dois métodos de seleção distintos: o **Torneio Binário** e a **Roleta**. Ambos os métodos garantem que os melhores indivíduos **têm uma maior probabilidade de transmitir** os seus genes para a próxima geração,

impulsionando a evolução da população. No entanto, possuem características distintas que influenciam a diversidade genética e a eficiência do processo evolutivo.

4.2.1.1 Torneio Binário

O método de **Torneio Binário** consiste na seleção de dois indivíduos aleatórios da população. O **indivíduo com melhor fitness** (ou seja, o de menor valor de função objetivo) é escolhido como progenitor. Este processo é repetido até que o número necessário de "**pais**" seja alcançado.

Vantagens:

- **Simplicidade:** O método é direto e de fácil implementação.
- **Eficiência:** Garante a seleção rápida dos melhores indivíduos através de comparações simples.
- **Diversidade Genética:** Promove a competição entre pares aleatórios, permitindo que indivíduos com fitness moderado ainda tenham alguma probabilidade de serem selecionados, evitando a convergência prematura.

Motivação: O Torneio Binário foi escolhido devido à sua capacidade de combinar eficiência e simplicidade, permitindo um equilíbrio entre exploração (diversidade) e exploração (seleção dos melhores).

4.2.1.2 Roleta

No método da **Roleta**, a probabilidade de seleção de cada indivíduo é proporcional ao seu fitness. Neste contexto, indivíduos com menor fitness têm uma maior probabilidade de serem escolhidos como pais, uma vez que representam soluções mais aptas. Pode-se dizer que a "roleta é girada" repetidamente para selecionar o número necessário de "**pais**".

Vantagens:

- **Balanceamento de Seleção:** A roleta oferece uma seleção **mais equilibrada**, permitindo que indivíduos menos aptos ainda possam ser selecionados, preservando a diversidade genética.
- **Diversidade Genética:** Reduz o risco de convergência prematura, ao garantir que soluções com fitness ligeiramente inferior ainda participem na evolução.

Motivação: O método da Roleta foi adotado pela sua capacidade de **promover um balanceamento** entre indivíduos com diferentes níveis de aptidão. Desta forma, é possível manter uma maior diversidade na população, evitando que o algoritmo fique preso em mínimos locais.

4.2.2 Operadores de Recombinação (crossover)

O **crossover** é um operador genético responsável pela **recombinação genética** entre dois indivíduos (pais), com o objetivo de gerar **novos indivíduos** (filhos) que herdem características dos progenitores. Este processo é fundamental para a exploração do espaço de soluções, permitindo que a informação genética dos pais seja combinada para formar descendentes potencialmente mais aptos.

Neste projeto foram implementados dois métodos de recombinação:

4.2.2.1 Crossover de um ponto

No **crossover de um ponto**, é selecionado **um ponto de corte aleatório** no cromossomo dos pais. A partir desse ponto, os genes são trocados entre os dois progenitores para gerar dois novos filhos.

Funcionamento:

- Os genes **antes do ponto de corte** permanecem inalterados.
- Os genes **após o ponto de corte** são trocados entre os pais.

Vantagens:

- Simplicidade e eficiência na implementação.
- Permite uma rápida mistura genética, promovendo a recombinação de características vantajosas.

Limitações:

- Pode limitar a diversidade genética em problemas complexos, especialmente se os genes relevantes estiverem próximos no cromossomo.

4.2.3 Operadores de Mutação (mutation)

A **mutação** é o operador responsável por introduzir **pequenas alterações** nos indivíduos, com o objetivo de **manter a diversidade genética** na população e evitar a convergência prematura para mínimos locais. Este operador desempenha um papel essencial ao garantir que o algoritmo evolutivo continue a explorar novas regiões do espaço de soluções.

Neste projeto, foram aplicados dois métodos de mutação:

4.2.3.1 Mutação Simples

A **mutação simples** consiste em alterar aleatoriamente o valor de um gene no cromossomo do indivíduo. A alteração ocorre com base em uma probabilidade de mutação pré-definida.

Funcionamento:

- Um gene é selecionado aleatoriamente.
- O valor do gene é alterado de acordo com o domínio do problema (ex.: incremento, decremento ou substituição).

Vantagens:

- Simples e eficiente para manter a diversidade genética.
- Introduz pequenas variações nos indivíduos, promovendo a exploração local.

Limitações:

- Pode ser insuficiente em problemas de alta complexidade onde grandes variações são necessárias para escapar de mínimos locais.

4.3 Algoritmo Híbrido

O **algoritmo híbrido** combina as características de exploração global dos algoritmos evolutivos com a exploração local do algoritmo **Trepa-Colinas**, resultando em soluções mais eficientes e robustas:

- O algoritmo evolutivo destaca-se pela sua capacidade de explorar amplamente o espaço de soluções, garantindo uma diversidade populacional e evitando mínimos locais prematuros.
- O algoritmo Trepa-Colinas, por sua vez, é eficaz no refinamento local, ajustando soluções individuais de forma intensiva para encontrar soluções melhores numa região específica.

4.3.1 Variantes do Algoritmo Híbrido

Foram implementadas duas variantes distintas do algoritmo híbrido, que diferem no momento em que o **Trepa-Colinas** é aplicado em relação ao processo evolutivo:

4.3.2 Híbrido 1

No Híbrido 1, o algoritmo de Trepa-Colinas é aplicado à população inicial gerada pelo algoritmo evolutivo. Cada indivíduo da população inicial é refinado localmente antes do início do processo evolutivo.

Funcionamento:

1. O algoritmo evolutivo gera uma população inicial aleatória.
2. O Trepa-Colinas é aplicado individualmente a cada solução da população inicial.
3. Após o refinamento local, o processo evolutivo prossegue com a população melhorada.

Vantagens:

- A população inicial já parte de soluções refinadas, aumentando as chances de evolução mais eficiente ao longo das gerações.
- Reduz a necessidade de iterações longas do algoritmo evolutivo, uma vez que os indivíduos começam com uma qualidade inicial superior.

Limitações:

- Pode ser computacionalmente mais caro em problemas com populações grandes, devido à aplicação do Trepa-Colinas a todos os indivíduos iniciais.

Aplicação: O Híbrido 1 é ideal para problemas onde se deseja acelerar a convergência inicial do algoritmo evolutivo, eliminando soluções de baixa qualidade logo no início.

4.3.3 Híbrido 2

No Híbrido 2, o algoritmo de Trepa-Colinas é aplicado após a evolução da população, refinando as melhores soluções obtidas pelo algoritmo evolutivo.

Funcionamento:

1. O algoritmo evolutivo é executado até o final, gerando uma população final com soluções evoluídas.
2. As melhores soluções da população final são selecionadas.
3. O Trepa-Colinas é aplicado a essas soluções, com o objetivo de refiná-las localmente e melhorar a sua qualidade final.

Vantagens:

- Foca o refinamento local apenas nas soluções mais promissoras, otimizando o uso de recursos computacionais.
- Garante que o processo evolutivo explore amplamente o espaço de soluções antes do ajuste local.

Limitações:

- Pode demorar mais tempo a atingir soluções ótimas em comparação com o Híbrido 1, pois o refinamento local ocorre apenas no final do processo evolutivo.

Aplicação: O Híbrido 2 é ideal para cenários em que o foco está na qualidade final das soluções e em problemas onde o refinamento local pode proporcionar ganhos significativos em soluções já evoluídas.

5 RESULTADO DOS TESTES EFETUADOS E ANÁLISE

5.1 Análise dos Resultados do Algoritmo Trepa-Colinas

Os resultados apresentados na tabela permitem comparar o desempenho das diferentes estratégias de vizinhança utilizadas no algoritmo Trepa Colinas. As estratégias foram aplicadas aos ficheiros file1.txt a file5.txt com diferentes números de iterações (150, 500, 1000, 5000, 10000, 250000). Aqui está um resumo das observações e diferenças:

5.1.1 Estratégia 0 – Incrementar um tipo de moeda aleatório

Trepa Colinas (vizinhança 1 - Incrementa um tipo de moeda aleatório)							
Nome do Ficheiro		**150 Iter.**	**500 Iter.**	**1000 Iter.**	**5000 Iter.**	**10000 Iter.**	**250000 Iter.**
file1.txt	***Melhor***	8	5	5	5	2	8
	Média	4054.20	2023.00	2022.70	3058.70	3047.90	5047.30
file2.txt	***Melhor***	184	328	201	103	78	86
	Média	4332.90	8420.90	6306.50	7323.70	3254.00	8322.80
file3.txt	***Melhor***	308	878	752	224	179	396
	Média	4756.90	9810.80	7669.50	7591.40	6495.40	8678.60
file4.txt	***Melhor***	4119	7093	3181	10240	2681	7938
	Média	11409.90	13792.60	10964.20	17133.40	16172.60	13281.30
file5.txt	***Melhor***	60	37	51	16	55	33
	Média	9061.80	8054.30	9050.10	4041.50	7054.80	8045.60

Figura 1 - Estudo baseado no algoritmo Trepa-Colinas com Vizinhança 1 (Estratégia 0)

Tendências:

- Esta estratégia mostrou bom desempenho em alguns ficheiros, como file1.txt e file5.txt, alcançando valores baixos tanto no "Melhor" quanto na "Média" com 150 e 1000 iterações.
- Em ficheiros mais complexos, como file2.txt e file4.txt, a média dos custos aumentou significativamente, mostrando que esta abordagem pode não ser eficiente para problemas com maior diversidade de moedas.

Conclusão:

- É uma estratégia eficaz para instâncias simples, mas pode não explorar suficientemente o espaço de busca em instâncias mais complexas.

5.1.2 Estratégia 1 - Decrementar um tipo de moeda aleatório (se possível)

Trepa Colinas (vizinhança 1 - Decrementa um tipo de moeda aleatório)							
Nome do Ficheiro		150 Iter.	500 Iter.	1000 Iter.	5000 Iter.	10000 Iter.	250000 Iter.
file1.txt	***Melhor***	**20**	**10054**	**20**	**20**	**5**	**10060**
	Média	**9266.30**	**10250.40**	**9272.10**	**9213.90**	**8210.90**	**10222.20**
file2.txt	***Melhor***	**10296**	**10214**	**10150**	**10205**	**10392**	**10373**
	Média	**11008.50**	**10862.20**	**11221.80**	**10882.30**	**10892.30**	**10964.60**
file3.txt	***Melhor***	**741**	**10988**	**11728**	**11010**	**11281**	**12684**
	Média	**24444.00**	**15303.00**	**15935.90**	**28373.90**	**21278.60**	**30642.00**
file4.txt	***Melhor***	**14442**	**12725**	**13544**	**15072**	**12494**	**14679**
	Média	**17908.00**	**16906.30**	**16298.20**	**17814.00**	**18794.50**	**17475.20**
file5.txt	***Melhor***	**10035**	**10042**	**10024**	**10046**	**10037**	**10059**
	Média	**10089.10**	**10097.10**	**10079.60**	**10081.60**	**10090.00**	**10123.20**

Figura 2 - Estudo baseado no algoritmo Trepa-Colinas com Vizinhança 1 (Estratégia 1)

Tendências:

- Em ficheiros como file3.txt e file4.txt, esta estratégia obteve resultados menos consistentes, com médias elevadas e melhores valores significativamente piores do que as outras estratégias.
- Para instâncias mais simples (file1.txt e file5.txt), a estratégia demonstrou bom desempenho inicial, mas foi superada pela Estratégia 0 em iterações mais longas.

Limitações:

- A abordagem de decremento parece limitar a exploração do espaço de soluções, o que pode explicar os valores elevados nos ficheiros mais complexos.

Conclusão:

- É uma estratégia limitada para explorar soluções em problemas com maior diversidade e complexidade.

5.1.3 Estratégia 2 - Troca de quantidades de moedas

Trepa Colinas (vizinhança 1 - Troca quantidades de moedas)							
Nome do Ficheiro		150 Iter.	500 Iter.	1000 Iter.	5000 Iter.	10000 Iter.	250000 Iter.
file1.txt	Melhor	11	11	20	5	8	11
	Média	7132.80	6096.70	9196.20	9187.90	8146.80	9210.20
file2.txt	Melhor	10342	10079	10333	10289	388	10382
	Média	10674.30	10635.60	10674.80	10632.60	9962.80	10585.90
file3.txt	Melhor	11072	998	734	11006	11611	11064
	Média	15298.80	18843.90	18976.80	15031.60	23173.40	17243.60
file4.txt	Melhor	13560	15491	14827	13026	15110	15189
	Média	16519.80	17781.00	17852.40	16746.90	17647.90	17416.20
file5.txt	Melhor	29	10040	10033	10062	10016	10038
	Média	9059.50	10069.30	10067.50	10078.00	10062.60	10066.00

Figura 3 - Estudo baseado no algoritmo Trepa-Colinas com Vizinhança 1 (Estratégia 2)

Tendências:

- Esta estratégia mostrou ser a mais consistente entre as três, especialmente em ficheiros mais desafiadores como file2.txt e file3.txt, onde alcançou médias mais baixas em várias iterações.
- Em ficheiros simples, como file1.txt e file5.txt, obteve resultados competitivos, com melhores valores e médias próximas das melhores.

Vantagens:

- A troca de quantidades permite uma exploração mais ampla do espaço de busca, o que explica sua eficiência em instâncias mais complexas.

Conclusão:

- É a estratégia mais robusta, equilibrando exploração e refinamento em problemas simples e complexos.

5.1.4 Diferenças entre Estratégias

Soluções Simples (file1.txt, file5.txt):

- Todas as estratégias alcançaram bons resultados, mas a Estratégia 0 mostrou-se ligeiramente mais eficiente em termos de médias.

Soluções Complexas (file2.txt, file3.txt, file4.txt):

- A Estratégia 2 destacou-se como a mais consistente, enquanto a Estratégia 1 obteve os piores resultados, apresentando maiores médias e custos elevados.

Iterações Longas (250000):

- A Estratégia 2 continua a demonstrar vantagem em instâncias complexas, enquanto as Estratégias 0 e 1 apresentaram altos custos em média.

5.2 Análise dos Resultados do Evolutivo

Foram realizados diversos testes experimentais utilizando o algoritmo evolutivo, com diferentes combinações de operadores de seleção, recombinação (crossover) e mutação, em 5 instâncias de ficheiros (file1.txt, file2.txt, file3.txt, file4.txt e file5.txt). Cada teste visou avaliar o impacto das configurações de parâmetros e estratégias na qualidade das soluções obtidas.

file1.txt

Crossover + Torneio + Mutation				Crossover + Torneio + Swap Mutation				Crossover + Roleta + Mutation				Crossover + Roleta + Swap Mutation			
Parâmetros Fixos	**Parâmetros Variáveis**	**Melhor**	**Média**	**Parâmetros Fixos**	**Parâmetros Variáveis**	**Melhor**	**Média**	**Parâmetros Fixos**	**Parâmetros Variáveis**	**Melhor**	**Média**	**Parâmetros Fixos**	**Parâmetros Variáveis**	**Melhor**	**Média**
pop = 100	pr = 0.3	**2.0**	**4.5**	pop = 100	pr = 0.3	**2.0**	**2.0**	pop = 100	pr = 0.3	**2.0**	**4.6**	pop = 100	pr = 0.3	**2.0**	**2.0**
ger = 5500	pr = 0.5	**2.0**	**4.4**	ger = 5500	pr = 0.5	**2.0**	**2.0**	ger = 5500	pr = 0.5	**2.0**	**4.3**	ger = 5500	pr = 0.5	**2.0**	**2.0**
pm = 0.5	pr = 0.7	**2.0**	**4.0**	pm = 0.5	pr = 0.7	**2.0**	**2.0**	pm = 0.5	pr = 0.7	**2.0**	**4.2**	pm = 0.5	pr = 0.7	**2.0**	**2.0**
tsize = 2	pr = 0.9	**2.0**	**4.3**	tsize = 2	pr = 0.9	**2.0**	**2.0**		pr = 0.9	**2.0**	**4.5**		pr = 0.9	**2.0**	**2.0**
pop = 100	pm = 0.01	**2.0**	**5.0**	pop = 100	pm = 0.01	**2.0**	**3.2**	pop = 100	pm = 0.01	**2.0**	**4.5**	pop = 100	pm = 0.01	**2.0**	**2.7**
ger = 5500	pm = 0.1	**2.0**	**4.4**	ger = 5500	pm = 0.1	**2.0**	**2.1**	ger = 5500	pm = 0.1	**2.0**	**4.2**	ger = 5500	pm = 0.1	**2.0**	**2.0**
pr = BEST	pm = 0.5	**2.0**	**4.4**	pr = BEST	pm = 0.5	**2.0**	**2.0**	pr = BEST	pm = 0.5	**2.0**	**4.6**	pr = BEST	pm = 0.5	**2.0**	**2.0**
tsize = 2	pm = 0.9	**2.0**	**4.6**	tsize = 2	pm = 0.9	**2.0**	**2.0**		pm = 0.9	**2.0**	**4.6**		pm = 0.9	**2.0**	**2.0**
pop = 100	tsize = 1	**2.0**	**4.5**	pop = 100	tsize = 1	**2.0**	**2.0**	ger = 5500	pop = 50	**2.0**	**4.9**	ger = 5500	pop = 50	**2.0**	**2.0**
ger = 5500	tsize = 2	**2.0**	**4.6**	ger = 5500	tsize = 2	**2.0**	**2.0**	pr = BEST	pop = 100	**2.0**	**4.6**	pr = BEST	pop = 100	**2.0**	**2.0**
pm = BEST	tsize = 5	**2.0**	**3.8**	pm = BEST	tsize = 5	**2.0**	**2.0**	pm = BEST	pop = 250	**2.0**	**3.6**	pm = BEST	pop = 250	**2.0**	**2.0**
pr = BEST	tsize = 10	**2.0**	**4.0**	pr = BEST	tsize = 10	**2.0**	**2.0**		pop = 500	**2.0**	**3.1**		pop = 500	**2.0**	**2.0**
ger = 5500	pop = 50	**2.0**	**5.8**	ger = 5500	pop = 50	**2.0**	**2.0**	pop = BEST	ger = 2500	**2.0**	**2.6**	pop = BEST	ger = 2500	**2.0**	**2.0**
pr = BEST	pop = 100	**2.0**	**4.9**	pr = BEST	pop = 100	**2.0**	**2.0**	pr = BEST	ger = 5500	**2.0**	**2.8**	pr = BEST	ger = 5500	**2.0**	**2.0**
pm = BEST	pop = 250	**2.0**	**4.0**	pm = BEST	pop = 250	**2.0**	**2.0**	pm = BEST	ger = 7500	**2.0**	**2.6**	pm = BEST	ger = 7500	**2.0**	**2.0**
tsize = BEST	pop = 500	**2.0**	**2.4**	tsize = BEST	pop = 500	**2.0**	**2.0**		ger = 10000	**2.0**	**2.2**		ger = 10000	**2.0**	**2.0**
pop = BEST	ger = 2500	**2.0**	**3.1**	pop = BEST	ger = 2500	**2.0**	**2.0**								
pr = BEST	ger = 5500	**2.0**	**2.7**	pr = BEST	ger = 5500	**2.0**	**2.0**								
pm = BEST	ger = 7500	**2.0**	**2.5**	pm = BEST	ger = 7500	**2.0**	**2.0**								
tsize = BEST	ger = 10000	**2.0**	**3.1**	tsize = BEST	ger = 10000	**2.0**	**2.0**								

Crossover Uniforme + Torneio + Mutation				Crossover Uniforme + Torneio + Swap Mutation				Crossover Uniforme + Roleta + Mutation				Crossover Uniforme + Roleta +Swap Mutation			
Parâmetros Fixos	**Parâmetros Variáveis**	**Melhor**	**Média**	**Parâmetros Fixos**	**Parâmetros Variáveis**	**Melhor**	**Média**	**Parâmetros Fixos**	**Parâmetros Variáveis**	**Melhor**	**Média**	**Parâmetros Fixos**	**Parâmetros Variáveis**	**Melhor**	**Média**
pop = 100	pr = 0.3	**2.0**	**4.2**	pop = 100	pr = 0.3	**2.0**	**2.0**	pop = 100	pr = 0.3	**2.0**	**4.4**	pop = 100	pr = 0.3	**2.0**	**2.0**
ger = 5500	pr = 0.5	**2.0**	**3.9**	ger = 5500	pr = 0.5	**2.0**	**2.0**	ger = 5500	pr = 0.5	**2.0**	**4.3**	ger = 5500	pr = 0.5	**2.0**	**2.0**
pm = 0.5	pr = 0.7	**5.0**	**5.0**	pm = 0.5	pr = 0.7	**2.0**	**2.0**	pm = 0.5	pr = 0.7	**2.0**	**4.3**	pm = 0.5	pr = 0.7	**2.0**	**2.0**
tsize = 2	pr = 0.9	**2.0**	**4.0**	tsize = 2	pr = 0.9	**2.0**	**2.0**		pr = 0.9	**2.0**	**4.3**		pr = 0.9	**2.0**	**2.0**
pop = 100	pm = 0.01	**2.0**	**4.4**	pop = 100	pm = 0.01	**2.0**	**3.1**	pop = 100	pm = 0.01	**2.0**	**5.1**	pop = 100	pm = 0.01	**2.0**	**2.7**
ger = 5500	pm = 0.1	**2.0**	**4.0**	ger = 5500	pm = 0.1	**2.0**	**2.0**	ger = 5500	pm = 0.1	**2.0**	**4.5**	ger = 5500	pm = 0.1	**2.0**	**2.0**
pr = BEST	pm = 0.5	**2.0**	**4.3**	pr = BEST	pm = 0.5	**2.0**	**2.0**	pr = BEST	pm = 0.5	**2.0**	**4.3**	pr = BEST	pm = 0.5	**2.0**	**2.0**
tsize = 2	pm = 0.9	**2.0**	**4.6**	tsize = 2	pm = 0.9	**2.0**	**2.0**		pm = 0.9	**2.0**	**4.2**		pm = 0.9	**2.0**	**2.0**
pop = 100	tsize = 1	**2.0**	**4.9**	pop = 100	tsize = 1	**2.0**	**2.0**	ger = 5500	pop = 50	**5.0**	**5.8**	ger = 5500	pop = 50	**2.0**	**2.1**
ger = 5500	tsize = 2	**2.0**	**4.3**	ger = 5500	tsize = 2	**2.0**	**2.0**	pr = BEST	pop = 100	**2.0**	**4.6**	pr = BEST	pop = 100	**2.0**	**2.0**
pm = BEST	tsize = 5	**2.0**	**4.5**	pm = BEST	tsize = 5	**2.0**	**2.5**	pm = BEST	pop = 250	**2.0**	**3.4**	pm = BEST	pop = 250	**2.0**	**2.0**
pr = BEST	tsize = 10	**2.0**	**4.3**	pr = BEST	tsize = 10	**2.0**	**2.6**		pop = 500	**2.0**	**2.8**		pop = 500	**2.0**	**2.0**
ger = 5500	pop = 50	**2.0**	**4.5**	ger = 5500	pop = 50	**2.0**	**2.0**	pop = BEST	ger = 2500	**2.0**	**3.0**	pop = BEST	ger = 2500	**2.0**	**2.0**
pr = BEST	pop = 100	**2.0**	**4.5**	pr = BEST	pop = 100	**2.0**	**2.0**	pr = BEST	ger = 5500	**2.0**	**3.0**	pr = BEST	ger = 5500	**2.0**	**2.0**
pm = BEST	pop = 250	**2.0**	**3.0**	pm = BEST	pop = 250	**2.0**	**2.0**	pm = BEST	ger = 7500	**2.0**	**2.6**	pm = BEST	ger = 7500	**2.0**	**2.0**
tsize = BEST	pop = 500	**2.0**	**2.7**	tsize = BEST	pop = 500	**2.0**	**2.0**		ger = 10000	**2.0**	**2.7**		ger = 10000	**2.0**	**2.0**
pop = BEST	ger = 2500	**2.0**	**2.8**	pop = BEST	ger = 2500	**2.0**	**2.0**								
pr = BEST	ger = 5500	**2.0**	**2.7**	pr = BEST	ger = 5500	**2.0**	**2.0**								
pm = BEST	ger = 7500	**2.0**	**3.0**	pm = BEST	ger = 7500	**2.0**	**2.0**								
tsize = BEST	ger = 10000	**2.0**	**3.2**	tsize = BEST	ger = 10000	**2.0**	**2.0**								

Figura 4- Testes do Evolutivo (file1.txt)

5.2.1 Configurações Testadas

5.2.1.1 Métodos de Seleção

- Torneio
- Roleta

5.2.1.2 Operadores de Recombinação (Crossover):

- Crossover de um ponto
- Crossover uniforme

5.2.1.3 Operadores de Mutação

- Mutação Simples
- Mutação Uniforme

5.2.1.4 Parâmetros Fixos e Variáveis

- **pop:** Tamanho da população (100, 500, 750).
- **ger:** Número de gerações (2500, 5500, 10000).
- **pr:** Probabilidade de crossover (0.3, 0.5, 0.7, 0.9).
- **pm:** Probabilidade de mutação (0.01, 0.1, 0.5, 0.9).
- **tsize:** Tamanho do torneio (1, 2, 5, 10).

5.2.2 Análise dos Resultados

5.2.2.1 Instância file1.txt

Observações:

- Em todas as combinações, os valores finais obtidos foram muito baixos, com Melhor = 2.0 e médias a variar ligeiramente.
- Parâmetros Ideais: População 100, gerações 2500, probabilidade de crossover 0.5, mutação simples 0.1.

Conclusão: Esta instância revelou-se simples, com soluções de boa qualidade obtidas de forma consistente.

5.2.2.2 Instância file2.txt

Observações:

- A combinação de Crossover + Torneio + Mutação Simples apresentou resultados Melhor = 20.0 e Média = 20.0, indicando uma boa convergência.
- A Mutação Swap apresentou maior dificuldade em algumas configurações.
- Parâmetros Ideais: Probabilidade de crossover 0.5, probabilidade de mutação 0.1, e gerações 10000.

Conclusão: Esta instância apresentou complexidade moderada, exigindo configurações robustas e maior número de gerações para convergir.

5.2.2.3 Instância file3.txt

Observações:

- Esta instância apresentou grande estabilidade, com Melhor = 102.0 em quase todas as configurações.
- A combinação Crossover Uniforme + Torneio + Swap Mutation destacou-se pela consistência nos valores de melhor solução.
- Parâmetros Ideais: Probabilidade de crossover 0.3, tamanho do torneio 5, e gerações 5500.

Conclusão: O problema foi mais desafiador, mas o uso de mutação simples e crossover consistente permitiu alcançar resultados estáveis.

5.2.2.4 Instância file4.txt

Observações:

- A complexidade desta instância exigiu um maior número de gerações e configurações mais agressivas de crossover.
- A combinação Crossover + Roleta + Swap Mutation obteve Melhor = 108.0, mas com Médias mais altas devido à complexidade do problema.
- Parâmetros Ideais: Gerações 7500, população 500, e probabilidade de crossover 0.7.

Conclusão: A roleta mostrou-se eficaz em evitar convergência prematura, equilibrando exploração e exploração.

5.2.2.5 Instância file5.txt

Observações:

- Os resultados Melhor = 4.0 foram alcançados de forma consistente com todas as configurações, sugerindo que o problema é relativamente simples.
- O crossover uniforme combinado com mutação simples destacou-se pela eficiência em encontrar boas soluções rapidamente.
- Parâmetros Ideais: Probabilidade de crossover 0.5, tamanho da população 100, e probabilidade de mutação 0.1.

Conclusão: Esta instância apresentou baixa complexidade, com soluções ótimas obtidas com poucas gerações e configurações simples.

5.3 Análise dos Resultados do Híbrido

Foram realizados testes com o modelo híbrido, que combina o algoritmo evolutivo com o algoritmo Trepa-Colinas, aplicando três variantes do modelo híbrido:

1. Híbrido Antes: O Trepa-Colinas é aplicado antes do processo evolutivo, refinando a população inicial.
2. Híbrido Depois: O Trepa-Colinas é aplicado após o processo evolutivo, refinando as melhores soluções encontradas.
3. Híbrido Meio: O Trepa-Colinas é aplicado durante o processo evolutivo, em intervalos definidos.

Os testes foram conduzidos em cinco instâncias (file1.txt a file5.txt) com parâmetros específicos para cada instância.

5.3.1 Resultados Obtidos por Instância

5.3.1.1 file1.txt

Configuração:

- Crossover + Roleta + Swap Mutation
- Parâmetros: popsize = 50, pm = 0.1, pr = 0.3, numGenerations = 2500
- Resultados:
- Híbrido Antes: Melhor = 2, Média = 2.00
- Híbrido Depois: Melhor = 2, Média = 2.00
- Híbrido Meio: Melhor = 2, Média = 2.00
- Conclusão: O problema é simples, e todas as variantes do modelo híbrido obtiveram soluções idênticas com ótimo desempenho, apresentando resultados consistentes e rápidos.

5.3.1.2 file2.txt

Configuração:

- Crossover + Roleta + Swap Mutation
- Parâmetros: popsize = 50, pm = 0.5, pr = 0.3, numGenerations = 2500

Resultados:

- Híbrido Antes: Melhor = 20, Média = 20.00
- Híbrido Depois: Melhor = 20, Média = 20.00
- Híbrido Meio: Melhor = 20, Média = 20.00

Conclusão: Esta instância apresentou soluções consistentes em todas as variantes híbridas, com desempenho ótimo e rápido refinamento local.

5.3.1.3 file3.txt

Configuração:

- Crossover Uniforme + Roleta + Swap Mutation
- Parâmetros: popsize = 50, pm = 0.5, pr = 0.5, numGenerations = 5500
- Resultados:
- Híbrido Antes: Melhor = 102, Média = 102.00

- Híbrido Depois: Melhor = 102, Média = 102.00
- Híbrido Meio: Melhor = 102, Média = 102.00

Conclusão: A complexidade do problema exigiu um maior número de gerações, mas todas as variantes do híbrido apresentaram resultados idênticos, evidenciando a eficácia do refinamento.

5.3.1.4 file4.txt

- Configuração:
- Crossover Uniforme + Torneio + Swap Mutation
- Parâmetros: popsize = 500, pm = 0.9, pr = 0.5, tsize = 2, numGenerations = 10000

Resultados:

- Híbrido Antes: Melhor = 108, Média = 108.00
- Híbrido Depois: Melhor = 108, Média = 108.00
- Híbrido Meio: Melhor = 108, Média = 108.00

Conclusão: O problema revelou-se mais exigente, mas todas as variantes obtiveram valores ótimos. O modelo híbrido mostrou ser eficaz mesmo com populações grandes e um elevado número de gerações.

5.3.1.5 file5.txt

Configuração:

- Crossover + Torneio + Swap Mutation
- Parâmetros: popsize = 50, pm = 0.9, pr = 0.3, tsize = 1, numGenerations = 2500
- Resultados:
- Híbrido Antes: Melhor = 4, Média = 4.00
- Híbrido Depois: Melhor = 4, Média = 4.00
- Híbrido Meio: Melhor = 4, Média = 4.00

Conclusão: O problema foi simples, com todas as variantes híbridas a obterem soluções ótimas de forma rápida e consistente.

5.4 Conclusão

O presente trabalho apresentou um estudo detalhado e experimental sobre a aplicação de algoritmos evolutivos combinados com técnicas de exploração local, como o Trepa-Colinas, em problemas de otimização. Foram implementados diversos métodos de seleção, recombinação e mutação, bem como diferentes

configurações de parâmetros, com o objetivo de identificar as abordagens mais eficazes para resolver diferentes instâncias do problema proposto.

Os resultados demonstraram que:

- O algoritmo evolutivo é uma ferramenta robusta para exploração global do espaço de soluções, sendo capaz de evitar a convergência prematura através de mecanismos como a seleção por roleta e mutação aleatória.
- A utilização de crossover (em particular o crossover uniforme) e mutação simples destacou-se pela sua eficiência e consistência, sobretudo em instâncias com menor complexidade.
- Em problemas mais exigentes, como observado nas instâncias file2.txt e file4.txt, soluções mais competitivas foram obtidas com configurações robustas, como maiores populações, mais gerações e maior probabilidade de crossover.
- A introdução do algoritmo híbrido permitiu equilibrar a exploração global do algoritmo evolutivo com a exploração local do Trepa-Colinas, refinando as soluções e melhorando significativamente os resultados.

No decorrer deste trabalho, comprovou-se que a parametrização adequada desempenha um papel crucial na eficiência do algoritmo evolutivo. Foi possível observar que configurações equilibradas, combinando seleção eficiente com operações de crossover e mutação bem definidas, resultam em soluções de maior qualidade.

Por fim, os resultados obtidos evidenciam a relevância da hibridização de métodos e da escolha criteriosa dos operadores evolutivos, proporcionando um compromisso entre exploração e exploração local, essencial para a resolução eficaz de problemas de otimização complexos. O trabalho deixa ainda espaço para estudos futuros que possam explorar novas variantes híbridas, ajustes automáticos de parâmetros e a aplicação destas técnicas em problemas de maior escala e complexidade.

Instituto Superior
de Engenharia
Politécnico de Coimbra