Trabalho 2 - Heurísticas e Metaheurísticas

```
#define TRABALHO 2
#define PROFESSOR "Guilherme Pena"
#define ESTUDANTE "Gabriel de Paula"
```

🧠 Metaheurísticas

Duas metaheurísticas foram desenvolvidas para resolver dois problemas clássicos de otimização combinatória: o TSP (Problema do Caixeiro Viajante) e o KP (Problema da Mochila).

1) Algoritmo Genético

O algoritmo genético (AG) é uma técnica de otimização inspirada na evolução biológica. Ele busca soluções aproximadas para problemas complexos utilizando mecanismos similares à seleção natural, como seleção, cruzamento e mutação.

Parâmetros	População inicial	Gerações	Crossover	Probabilidade de Mutação	Taxa de Reprodução
Constantes	POPULACAO_INICIAL	GERACOES	CROSSOVER	MUTACAO	REPRODUCAO

🧬🔄🧬🔄🧬🔄🧬

2) Colônia de Formigas

O algoritmo de colônia de formigas (ACO) imita o comportamento das formigas na busca por alimento, usando feromônios para guiar a exploração de soluções, reforçando os melhores caminhos ao longo das iterações.

Parâmetros	Iterações	Formigas	Expoente Trilha	Expoente Caminho	Evaporação	Reforço
Constantes	ITERACOES	FORMIGAS	A	B	EVAPORACAO	REFORCO

🐜🐜🐜🐜🐜🐜

⚙️ Compilação

Compilar todos os executáveis em modo otimizado:

```
make
```

Compilar todos os executáveis em modo otimizado, imprimindo todas as soluções aceitas:

```
make debug
```

🛠️ Execução

Programas gerados:

Algoritmo	Problema	Programa
AG	TSP	ag-tsp
AG	KP	ag-kp
ACO	TSP	aco-tsp

Para executar um programa é necessário fornecer os valores numéricos dos parâmetros conforme indicado anteriormente.

```
./bin/<programa> <parametro1> <...> <parametroN>
```

Exemplo de como executar um programa 50 vezes para gerar relatórios:

```
for i in {1..50}; do ./bin/<programa> <parametro1> <...> <parametroN>; done
```

📊 Resultados

Foi realizada uma análise detalhada dos parâmetros de execução de todos os programas utilizando Grid Search, que explora o produto cartesiano de todas as combinações possíveis de parâmetros. Além do valor obtido, foi dada ênfase à análise do tempo decorrido.

Como ambos os algoritmos utilizam de aleatoriedade na geração de soluções, foi calculada a média dos resultados obtidos em 10 execuções, garantindo mais consistência nos dados.

Algumas combinações de parâmetros do ACO se fazem contraditórias, tendo em vista que são opostas, dessa forma foram testados como um conjunto.

> Solução ótima TSP (min) = **426**
>
> Solução ótima KP (max) = **9147**

Algoritmo Genético

1. POPULACAO_INICIAL = [10, 50, 100]
2. GERACOES = [50, 100, 500]
3. MUTACAO = [0.05, 0.33, 0.66, 1]

ag-tsp

É notável a dependência do algoritmo genético em relação ao número de iterações e à quantidade de gerações observando os testes, já que no começo as soluções são puramente aleatórias e vão se combinando até gerar soluções melhores. O problema disso é o tempo para finalizar a execução, que vai aumentando conforme as iterações aumentam.

Parâmetros	Valor médio	Tempo médio
(10, 50, 0.05)	967.83	0.5707 s
(10, 50, 0.33)	976.34	0.5358 s
(10, 50, 0.66)	1038.43	0.5571 s
(10, 50, 1)	1052.89	0.5205 s
(10, 100, 0.05)	1027.59	1.0991 s
(10, 100, 0.33)	933.88	1.0900 s
(10, 100, 0.66)	906.25	1.0985 s
(10, 100, 1)	916.21	1.0681 s
(10, 500, 0.05)	739.32	5.6350 s

Parâmetros	Valor médio	Tempo médio
(10, 500, 0.33)	658.01	5.5894 s
(10, 500, 0.66)	608.87	5.5910 s
(10, 500, 1)	796.43	5.6000 s
(50, 50, 0.05)	934.17	2.8528 s
(50, 50, 0.33)	900.78	2.8428 s
(50, 50, 0.66)	880.10	2.8453 s
(50, 50, 1)	1022.69	2.8512 s
(50, 100, 0.05)	752.44	5.7386 s
(50, 100, 0.33)	709.70	5.7128 s
(50, 100, 0.66)	786.89	5.7173 s
(50, 100, 1)	785.22	5.6811 s
(50, 500, 0.05)	510.86	29.4605 s
(50, 500, 0.33)	548.83	29.4370 s
(50, 500, 0.66)	527.67	29.5700 s
(50, 500, 1)	603.75	29.4122 s
(100, 50, 0.05)	823.83	5.6877 s
(100, 50, 0.33)	840.51	5.6726 s
(100, 50, 0.66)	902.59	5.6281 s
(100, 50, 1)	977.63	5.7044 s
(100, 100, 0.05)	677.17	11.3780 s
(100, 100, 0.33)	652.95	11.5339 s
(100, 100, 0.66)	792.25	11.3860 s
(100, 100, 1)	819.16	11.3923 s
(100, 500, 0.05)	**494.13**	58.5129 s
(100, 500, 0.33)	**487.53**	58.6289 s
(100, 500, 0.66)	**483.77**	59.5034 s
(100, 500, 1)	556.67	58.4693 s

Interessante o fato de que as mutações não são obrigatórias, mas também não são dispensáveis. É necessário dar atenção a esse parâmetro muitas vezes desprezado, pois, em alguns casos, coeficientes intermediários alcançam soluções melhores.

ag-kp

A resolução do problema da mochila, em sua implementação, possui uma execução mais performática devido às estruturas de dados utilizadas, sendo possível alcançar a solução ótima em um tempo menor. Observa-se também uma dependência dos parâmetros de população e gerações, atingindo as melhores soluções com pelo menos 100 gerações.

Parâmetros	Valor médio	Tempo médio
(10, 50, 0.05)	6774.00	0.0026 s
(10, 50, 0.33)	6207.00	0.0049 s
(10, 50, 0.66)	6543.00	0.0024 s
(10, 50, 1)	6439.00	0.0050 s
(10, 100, 0.05)	8497.00	0.0033 s
(10, 100, 0.33)	8349.00	0.0031 s
(10, 100, 0.66)	8247.00	0.0064 s
(10, 100, 1)	8315.00	0.0035 s
(10, 500, 0.05)	8929.00	0.0179 s
(10, 500, 0.33)	8817.00	0.0184 s
(10, 500, 0.66)	8940.00	0.0071 s
(10, 500, 1)	8929.00	0.0187 s
(50, 50, 0.05)	8658.00	0.0057 s
(50, 50, 0.33)	8179.00	0.0137 s
(50, 50, 0.66)	8017.00	0.0051 s
(50, 50, 1)	8331.00	0.0138 s
(50, 100, 0.05)	**9147.00**	0.0096 s
(50, 100, 0.33)	**9147.00**	0.0086 s
(50, 100, 0.66)	**9147.00**	0.0093 s
(50, 100, 1)	**9147.00**	0.0098 s
(50, 500, 0.05)	**9147.00**	0.0378 s
(50, 500, 0.33)	**9147.00**	0.0381 s
(50, 500, 0.66)	**9147.00**	0.0686 s
(50, 500, 1)	**9147.00**	0.0382 s
(100, 50, 0.05)	8690.00	0.0284 s

Parâmetros	Valor médio	Tempo médio
(100, 50, 0.33)	7983.00	0.0103 s
(100, 50, 0.66)	8745.00	0.0289 s
(100, 50, 1)	8940.00	0.0109 s
(100, 100, 0.05)	**9147.00**	0.0516 s
(100, 100, 0.33)	8810.00	0.0197 s
(100, 100, 0.66)	8940.00	0.0203 s
(100, 100, 1)	**9147.00**	0.0513 s
(100, 500, 0.05)	**9147.00**	0.0886 s
(100, 500, 0.33)	**9147.00**	0.0880 s
(100, 500, 0.66)	**9147.00**	0.1200 s
(100, 500, 1)	**9147.00**	0.1208 s

Nota-se também que, para esse problema em específico, manter o coefieciente de mutação baixo traz melhores soluções.

Colônia de Formigas

1. ITERACOES = [100, 1000];
2. FORMIGAS = [50, 100];
3. {A, B} = [{2, 1}, {1, 1}, {1, 2}];
4. {EVAPORACAO, REFORCO} = [{0.01, 1}, {0.1, 1}, {0.5, 1}];

aco-tsp

Novamente, algoritmos bioinspirados tendem a obter melhores resultados com o aumento da população e das iterações.

Parâmetros	Valor médio	Tempo médio
(100, 50, 2 1, 0.01 1)	520.49	1.0726 s
(100, 50, 2 1, 0.1 1)	550.41	1.0670 s
(100, 50, 2 1, 0.5 1)	565.88	1.0579 s
(100, 50, 1 1, 0.01 1)	614.13	1.0845 s
(100, 50, 1 1, 0.1 1)	508.59	1.0417 s
(100, 50, 1 1, 0.5 1)	829.49	1.0421 s
(100, 50, 1 2, 0.01 1)	467.38	1.0941 s
(100, 50, 1 2, 0.1 1)	456.19	1.0734 s

Parâmetros	Valor médio	Tempo médio
(100, 50, 1 2, 0.5 1)	584.05	1.0740 s
(100, 100, 2 1, 0.01 1)	483.39	2.0796 s
(100, 100, 2 1, 0.1 1)	575.10	2.0705 s
(100, 100, 2 1, 0.5 1)	497.55	2.0541 s
(100, 100, 1 1, 0.01 1)	598.68	2.0965 s
(100, 100, 1 1, 0.1 1)	554.41	2.0862 s
(100, 100, 1 1, 0.5 1)	843.50	2.0986 s
(100, 100, 1 2, 0.01 1)	466.37	2.0899 s
(100, 100, 1 2, 0.1 1)	447.29	2.0564 s
(100, 100, 1 2, 0.5 1)	545.25	2.0793 s
(1000, 50, 2 1, 0.01 1)	515.13	10.1908 s
(1000, 50, 2 1, 0.1 1)	503.60	10.2350 s
(1000, 50, 2 1, 0.5 1)	460.34	10.2861 s
(1000, 50, 1 1, 0.01 1)	**435.02**	10.3742 s
(1000, 50, 1 1, 0.1 1)	444.69	10.3313 s
(1000, 50, 1 1, 0.5 1)	767.05	10.4616 s
(1000, 50, 1 2, 0.01 1)	**429.74**	10.3158 s
(1000, 50, 1 2, 0.1 1)	**438.68**	10.3402 s
(1000, 50, 1 2, 0.5 1)	520.67	10.3790 s
(1000, 100, 2 1, 0.01 1)	482.65	20.1162 s
(1000, 100, 2 1, 0.1 1)	483.41	20.1574 s
(1000, 100, 2 1, 0.5 1)	**438.32**	20.4027 s
(1000, 100, 1 1, 0.01 1)	**432.96**	20.3740 s
(1000, 100, 1 1, 0.1 1)	**438.81**	20.3609 s
(1000, 100, 1 1, 0.5 1)	792.32	20.7825 s
(1000, 100, 1 2, 0.01 1)	**430.35**	20.4622 s
(1000, 100, 1 2, 0.1 1)	**433.74**	20.2529 s
(1000, 100, 1 2, 0.5 1)	474.11	20.5453 s

A importância da evaporação ser baixa é evidenciada nos resultados, onde em praticamente todas as situações aumentar o valor trouxe piora nas soluções.