Photo Slideshow

IART – Trabalho 1

João Praça – up201704748
Liliana Almeida – up201706908
Sílvia Rocha – up201704684

Especificação do Trabalho

Neste problema de otimização é pretendido que, dada uma lista de fotos e das *tags* associadas a cada foto, estas sejam organizadas num *slideshow* que obtenha a maior pontuação possível. A pontuação de um *slideshow* é dada como a soma dos fatores de interesse calculados para cada par de slides.

O fator de interesse corresponde ao mínimo entre o número de *tags* em comum entre os dois slides, o número de *tags* presente no primeiro slide mas não no segundo e o número de *tags* presente no segundo slide mas não no primeiro.

Formulação do Problema

Representação da solução

O estado inicial e a solução a cada iteração correspondem a um vetor de vetores, do tipo [[slide1], [slide2],...,[slideN]], otimizado de modo a que a pontuação total seja máxima. Cada slide corresponde a um vetor de imagens que pode ter tamanho 1 ou 2 (consoante a imagem seja vertical ou horizontal). A solução inicial obtida é um vetor ordenado conforme as imagens são lidas do ficheiro escolhido, sendo as imagens verticais, posteriormente, emparelhadas em slides conforme vão aparecendo no vetor.

Restrições rígidas

- Cada imagem poder ser usada 0 ou 1 vezes;
- Cada slide tem de conter uma imagem horizontal ou duas verticais;
- O slideshow tem de ter pelo menos 1 slide;
- Uma imagem não pode ser inserida num slide já cheio.

Abordagem

Função de vizinhança

A cada iteração dos algoritmos é chamada a função de vizinhança que será quem, em conjunto com a função de avaliação, irá gerar as soluções vizinhas do estado atual.

Na função de vizinhança implementada, são selecionados dois *slides* aleatórios. De seguida, é avaliado se os dois *slides* selecionados são de imagens verticais ou horizontais. Caso sejam *slides* com imagens horizontais é efetuado um *swap* entre eles. Caso os *slides* contenham imagens verticais, existe uma probabilidade de 50% de trocar os 2 *slides* ou de efetuar a troca entre duas imagens verticais dos dois *slides.*

Função de avaliação/heurística

A pontuação de dois slides adjacentes (fator de interesse) é calculada como o mínimo entre número de *tags* em comum, número de *tags* presente no primeiro mas não no segundo e o número de *tags* presente no segundo mas não no primeiro. A pontuação total do *slideshow* corresponde à soma de todos os fatores de interesse.

Algoritmos Implementados

Para a resolução do problema de otimização já apresentado foram implementados algoritmos iterativos. Estes algoritmos mantêm um único estado ou uma população de estados, tentando melhora-los a cada iteração.

Existem dois tipos de algoritmos iterativos : os baseados em indivíduos e os baseados em populações de soluções. Do primeiro tipo foram implementados os seguintes algoritmos:

Hill Climbing
Dentro deste tipo de algoritmo, existem três abordagens possíveis. A nossa implementação segue a versão aleatória do algoritmo. A execução começa com um estado inicial aleatório e a cada iteração é gerado um sucessor aleatório, caso este melhore a pontuação atual, é selecionado como novo estado. Caso contrário, a solução encontrada até então é mantida e avança para a iteração seguinte. O algoritmo termina quando atinge o número máximo de iterações parametrizado.

Simulated Annealing
Este algoritmo segue uma lógica semelhante à descrita acima no *Hill Climbing* com a diferença de que permite visitar vizinhos piores. A probabilidade de numa determinada iteração uma solução pior ser aceite é determinada através da temperatura que vai sendo reduzida de cada vez que isso acontece.

Algoritmos Implementados

Tabu Search

O algoritmo de *tabu search* é semelhante ao de *hill climbing* sendo que a cada iteração vai guardando numa fila os estados recentemente visitados.

Quanto aos algoritmos baseados em população, foi implementado o seguinte algoritmo:

Algoritmo Genético

Nos algoritmos genéticos é gerada uma população inicial que vai sendo alterada através de processos de seleção, *crossover* e mutação. Na implementação realizada, a fase de seleção é efetuada com recurso ao método *roulette wheel.* Este método seleciona os pais que irão sofrer *crossover* calculando probabilidades baseadas no *fitness score* de cada um dos indivíduos da geração. Na fase de *crossover*, é realizado um *Linear Order Crossover* (LOX) gerando duas *offsprings* candidatas. Destas, a que apresentar um melhor *fitness score* irá substituir o pai com o pior *fitness score.* A fase de mutação é realizada com recurso a uma probabilidade de 1%. A cada iteração, os *fitness scores* que foram alterados são atualizados e é efetuada uma nova seleção até atingir o número máximo de gerações.

Resultados

Ficheiro C – Memorable Moments

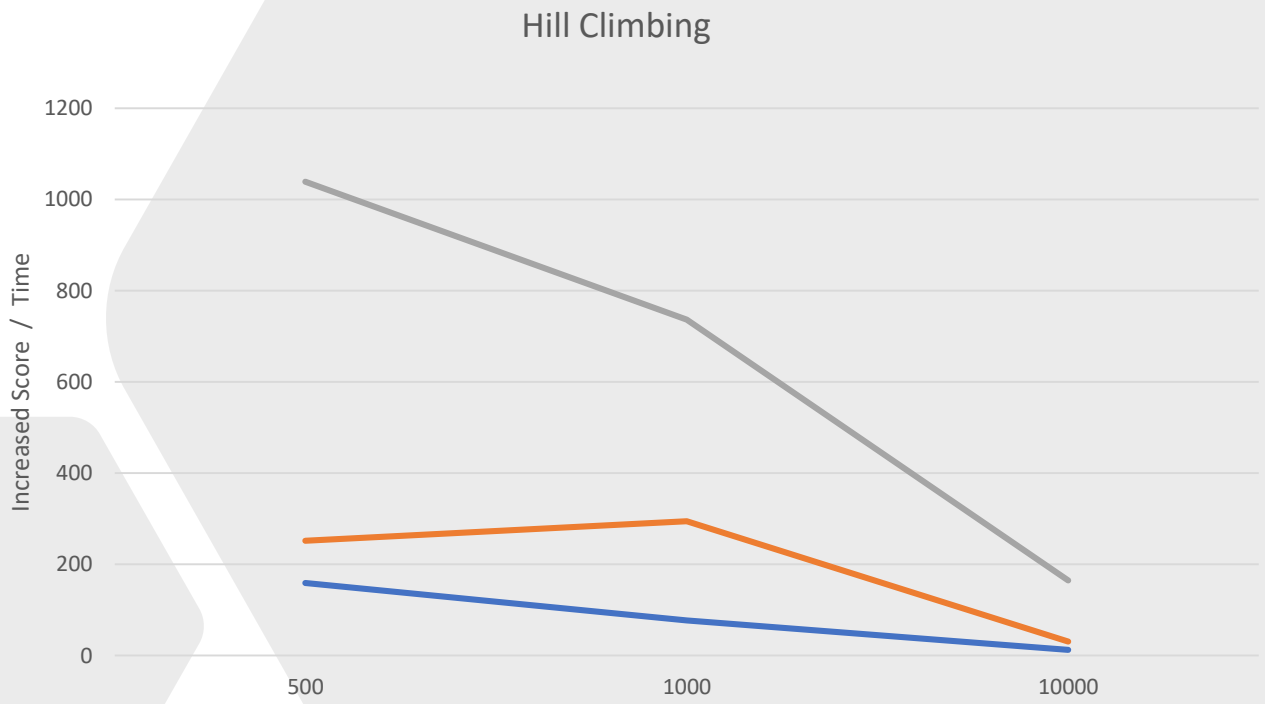
Número de iterações	Tempo	Pontuação
1000	5	769
5000	8	872
10000	47	1147

Ficheiro D – Pet Pictures

Número de iterações	Tempo	Pontuação
1000	7	3487
5000	6	3498
10000	65	3918

Ficheiro E – Shiny Selfies

Número de iterações	Tempo	Pontuação
1000	15	3372
5000	31	3376
10000	226	3975



Resultados

Simulated Annealing T=10

Ficheiro C – Memorable Moments

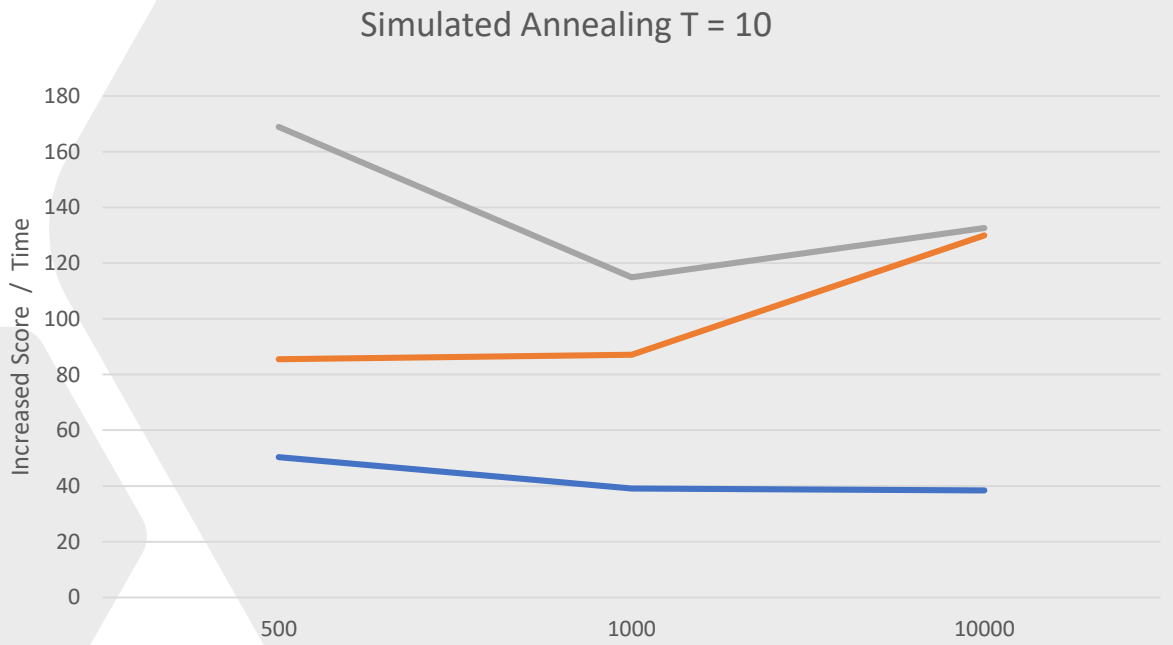
Número de iterações	Tempo	Pontuação
1000	9	225
5000	9	153
10000	8	157

Ficheiro D – Pet Pictures

Número de iterações	Tempo	Pontuação
1000	13	2197
5000	12	2067
10000	8	2056

Ficheiro E – Shiny Selfies

Número de iterações	Tempo	Pontuação
1000	23	1635
5000	27	1488
10000	27	1461



Resultados

Simulated Annealing T=50

Ficheiro C – Memorable Moments

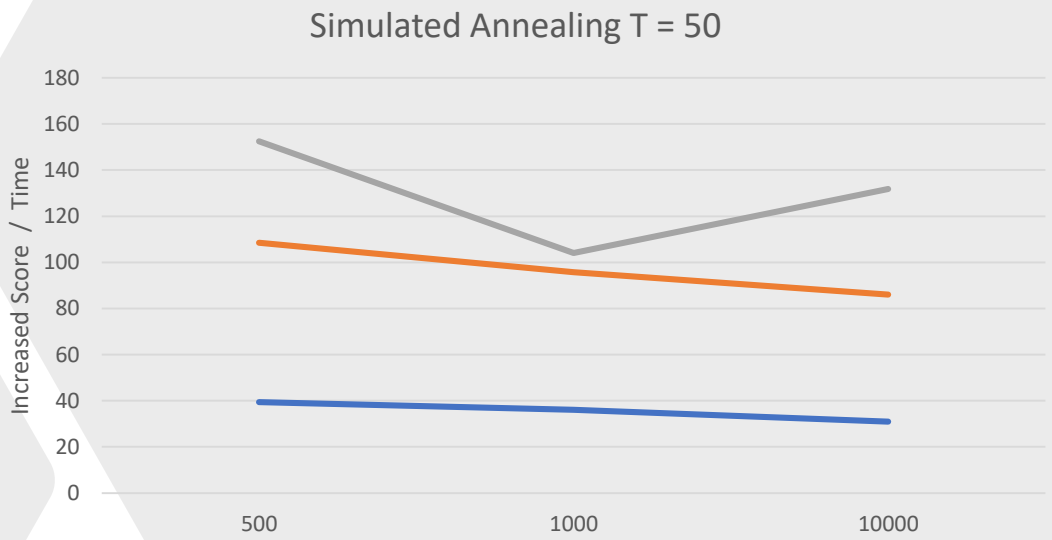
Número de iterações	Tempo	Pontuação
1000	9	203
5000	10	154
10000	8	156

Ficheiro D – Pet Pictures

Número de iterações	Tempo	Pontuação
1000	10	2145
5000	11	2081
10000	12	2042

Ficheiro E – Shiny Selfies

Número de iterações	Tempo	Pontuação
1000	31	1724
5000	29	1477
10000	36	1452



Resultados

Simulated Annealing T=100

Ficheiro C – Memorable Moments

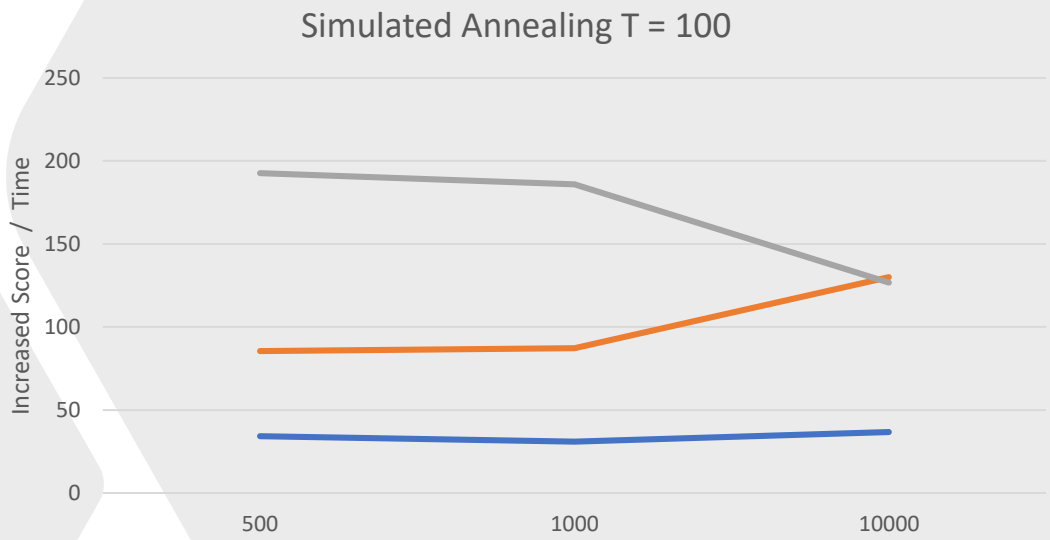
Número de iterações	Tempo	Pontuação
1000	8	228
5000	6	165
10000	8	150

Ficheiro D – Pet Pictures

Número de iterações	Tempo	Pontuação
1000	10	2259
5000	10	2070
10000	10	2046

Ficheiro E – Shiny Selfies

Número de iterações	Tempo	Pontuação
1000	36	1741
5000	34	1482
10000	28	1453



Resultados

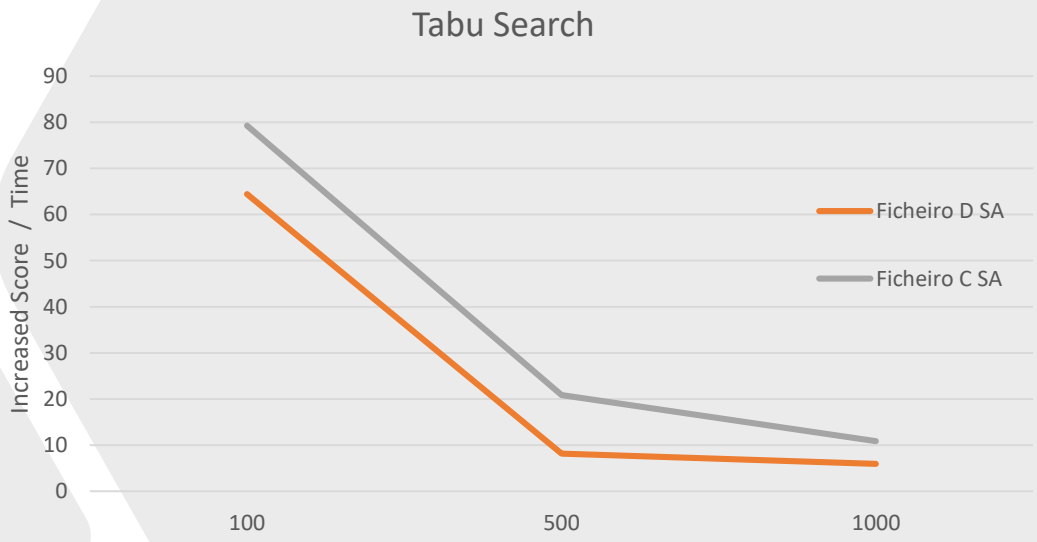
Tabu Search

Ficheiro C – Memorable Moments

Número de iterações	Tempo	Pontuação
1000	15	176
5000	77	238
10000	152	245

Ficheiro D – Pet Pictures

Número de iterações	Tempo	Pontuação
1000	17	2166
5000	129	2088
10000	183	2164



Resultados

Genetic Algorithm P=20

Ficheiro C – Memorable Moments

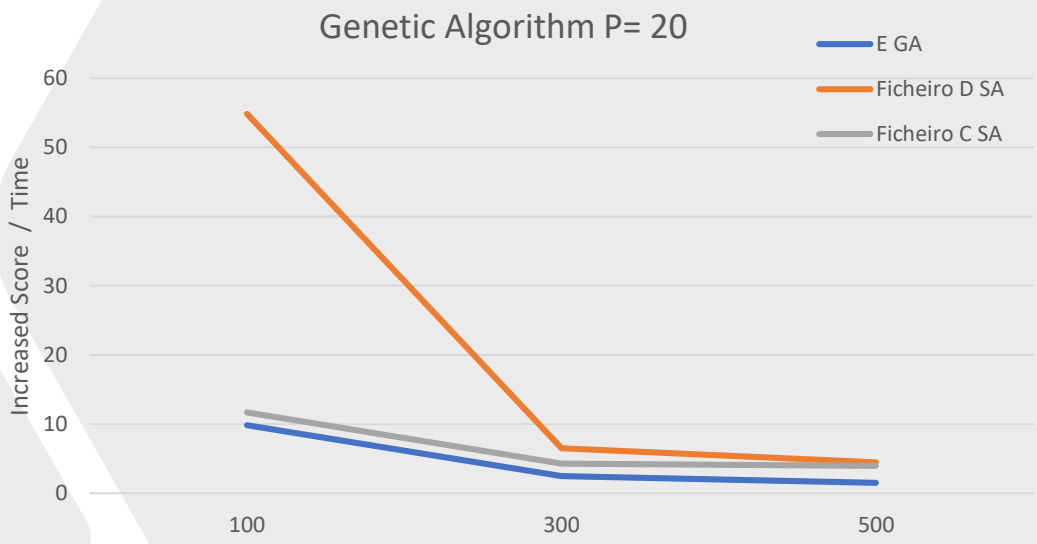
Número de iterações	Tempo	Pontuação
1000	96	166
5000	281	178
10000	327	191

Ficheiro D – Pet Pictures

Número de iterações	Tempo	Pontuação
1000	19	2060
5000	162	2090
10000	245	2166

Ficheiro E – Shiny Selfies

Número de iterações	Tempo	Pontuação
1000	106	1473
5000	428	1515
10000	735	1575



Resultados

Genetic Algorithm P=40

Ficheiro C – Memorable Moments

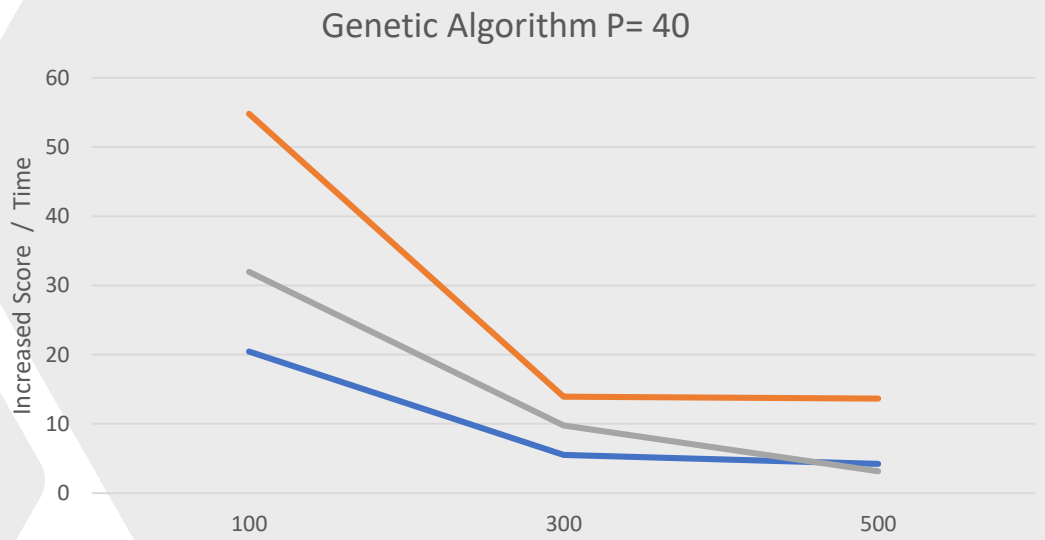
Número de iterações	Tempo	Pontuação
1000	33	156
5000	117	169
10000	361	169

Ficheiro D – Pet Pictures

Número de iterações	Tempo	Pontuação
1000	19	2058
5000	77	2121
10000	79	2131

Ficheiro E – Shiny Selfies

Número de iterações	Tempo	Pontuação
1000	51	1470
5000	193	1502
10000	249	1479



Resultados

Genetic Algorithm P=60

Ficheiro C – Memorable Moments

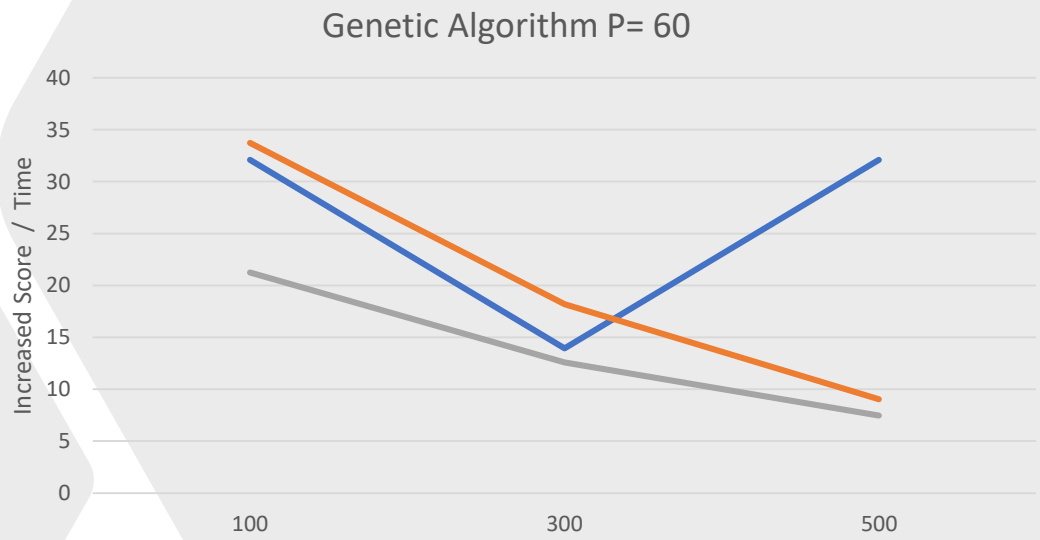
Número de iterações	Tempo	Pontuação
1000	49	154
5000	88	164
10000	155	171

Ficheiro D – Pet Pictures

Número de iterações	Tempo	Pontuação
1000	31	154
5000	58	164
10000	117	171

Ficheiro E – Shiny Selfies

Número de iterações	Tempo	Pontuação
1000	32	1449
5000	76	1495
10000	84	1490



Conclusões

No algoritmo *Hill Climbing*, a pontuação aumenta com o número de iterações, no entanto a eficiência calculada faz com que diminua.
No algoritmo *Simulated Annealing,* a pontuação diminui com o número de iterações, no entanto a eficiência calculada e o aumento da temperatura não apresentam resultados conclusivos.
No algoritmo *Tabu Search*, a pontuação aumenta com o número de iterações, no entanto a eficiência calculada e o aumento do número de vizinhos gerados resultam num decréscimo da mesma.
No algoritmo Genético, a pontuação aumenta com o número de gerações, diminui ligeiramente com o aumento da população, e a eficiência segue também uma tendência decrescente.
A eficiência calculada para avaliar o resultados foi calculada da seguinte forma: ((scoreFinal/scoreInicial)/tempo)*numImagensFicheiro.

Referências

https://towardsdatascience.com/genetic-algorithm-explained-step-by-step-65358abe2bf

https://en.wikipedia.org/wiki/Fitness_proportionate_selection

https://github.com/kobr4/g_hashcodes

https://github.com/AhmedSoli/Hashcode-2019

https://github.com/LJMUAstroecology/hashcode_2019

https://github.com/inaitana/hashcode-2019-qualification

https://medium.com/@danieleratti/how-we-placed-1st-in-italy-and-22nd-in-the-world-google-hashcode-2019-e59e52232b4e