Relatório do Primeiro Trabalho de Inteligência Artificial

Leandro Furlam Turi

Abstract

A organização de dados em estruturas que façam sentido é um problema comum em aprendizado de máquina. O agrupamento de dados numéricos multidimensionais possuem uma infinidade de aplicações, onde a mais bem-conhecida é a categorização das flores Íris em suas diferentes espécies. Neste trabalho, três diferentes metaheurísticas serão avaliadas a esta seara: Simulated Annealing, Genetic Algorithm e GRASP, visando observar e compreender o comportamento de cada uma nos problemas dados.

Keywords: Simulated Annealing, Genetic Algorithm, GRASP, Agrupamento.

1. Introdução

Como organizar dados observados em estruturas que façam sentido, ou como desenvolver taxonomias capazes de classificar dados observados em diferentes classes?. A análise ou agrupamento em *clusters* é a tarefa de agrupar um conjunto de itens de tal forma que os itens presentes em um mesmo grupo (*cluster*) são semelhantes. A análise em *cluster* em si não é um algoritmo específico, mas a tarefa geral a ser resolvida [1].

Metaheurísticas podem ser empregadas nesta tarefa. Elas consistem em métodos heurísticos para resolver de forma genérica problemas de otimização, através da utilização da combinação de escolhas aleatórias e conhecimento histórico dos resultados anteriores adquiridos pelo método para se guiarem e realizar suas buscas pelo espaço de pesquisa em vizinhanças dentro do espaço de pesquisa, o que evita paradas prematuras em ótimos locais [2].

Neste trabalho, analisaremos o comportamento de três destes métodos: Simulated Annealing, Genetic Algorithm e GRASP, visando analisar e avaliar o

comportamento de cada um deles em diferentes instâncias.

2. Problema

Dado um conjunto de dados X com N pontos $x_1, \cdots, x_N$, sendo que cada ponto $x_i = [x_{i1}, x_{i2}, \cdots, x_{id}]^t$ possui d coordenadas (dimensões), deseja-se encontrar K grupos $C_1, ..., C_K$, de tal forma que as seguintes condições sejam atendidas:

- $C_j \neq \emptyset$, $j = 1 : K$;
- $\cup_{j=1}^{K} C_j = X$;
- $C_i \cap C_j = \emptyset$, $i \neq j$, $i, j = 1 : K$.

Acerca da qualidade da divisão em grupos é aplicada a a soma das distâncias euclidianas quadradas (SSE):

$$SSE = \sum_{j=1}^{K} \sum_{x_i \in C_j} ||x_i - \mu_j||_2$$

onde $|| * ||_2$ é a norma euclideana.

O centróide $\mu_j = [\mu_{j1}, \mu_{j2}, \cdots, \mu_{jd}]^t$ é o ponto representativo do grupo C_j e é calculado como o centro de massa do grupo:

$$\mu_j = \frac{1}{n_j} \sum_{x_i \in C_j} x_i$$

onde n_j é o total de pontos pertencentes ao grupo C_j.

Neste problema busca-se a distribuição de pontos em grupos que minimiza a SSE.

3. Métodos utilizados

3.1. Representação do Descritor de Espaço de Estados

Itens são modelados como dicionários, exemplificados a seguir:

```
{'id': 1, 'coord': [7, 5.4, 6.32, 9]}
```

onde `id` é o identificador do item, e `coord` são as respectivas coordenadas.

Um *cluster* é um lista de itens:

```
[{'coord': [7, 5.4, 6.32, 9], 'id': 1},
 {'coord': [8.9, 5.8, 6, 9], 'id': 5},
 {'coord': [17, 32.3, 5, 9.99], 'id': 2}]
```

Enquanto um estado é um cluster avaliado através da função `evaluate_cluster()`, com os cômputos armazenados:

```
{'dist': [9.947, 8.951, 18.822],
 'itens': [{'coord': [7, 5.4, 6.32, 9], 'id': 1},
           {'coord': [8.9, 5.8, 6, 9], 'id': 5},
           {'coord': [17, 32.3, 5, 9.99], 'id': 2}],
 'mu': [10.966, 14.5, 5.773, 9.33],
 'sum_dist': 37.720}
```

onde `dist` são as distâncias de cada item ao centróide do cluster `mu`, `itens` são os respectivos itens e `sum_dist` a função objetivo, que no caso de interesse é a SSE.

3.2. GRASP

A metaheurística GRASP (Greedy Randomized Adaptive Search Procedure) é um algoritmo construtivo, privilegiando o ato de criar uma solução inicial de qualidade para depois efetuar uma busca local, realizando melhoras e objetivando o ótimo. Seu diferencial para outros métodos está na geração dessa solução inicial, baseada nas três primeiras iniciais de sua sigla em inglês: gulosa (*Greedy*), aleatória (*Randomized*) e adaptativa (*Adaptive*) [3]. O pseudocódigo

do algoritmo está apresentado no Algoritmo 1.

Algorithm 1: GRASP

Input: conjunto de itens

Output: solução ótima

```
1 while condição de parada não for satisfeita do
2 |   solução ← crie aleatoriamente uma solução de forma construtiva()
  |    solução ← busca local(solução)
3 |   if solução é a melhor solução até então conhecida then
4 |   |   grave(solução)
5 |   end
6 end
7 return solução ótima
```

A busca local foi realizada a partir de um *hill climbing* não determinístico. O primeiro requisito foi cumprido a partir de uma solução inicial aleatória e de uma escolha bem limitada de melhores elementos disponíveis, isto é, o elemento a ser inserido é escolhido aleatoriamente dentre os $n < ||itens||$ itens que tornam o atual *cluster* com o menor SSE. Após delimitar os melhores elementos, é sorteado um aleatoriamente sem verificar os demais, fazendo com que o algoritmo seja guloso e também míope. Ao adicionar um novo elemento na solução, ocorre a atualização de ambas as listas, tornando-o adaptativo. Como critério de parada, são utilizados parada por quantidade máxima de iterações e tempo de processamento máximo.

3.3. Simulated Annealing

O Simulated Annealing é uma metaheurística que consiste numa técnica de busca local probabilística, e se fundamenta numa metáfora de um processo térmico, dito recozimento ou *annealing*, utilizado em metalurgia. O processo consiste em duas etapas: na primeira, a temperatura do sólido é aumentada de forma que ocorra uma transformação de fase; e na segunda, um resfriamento controlado deve ser realizado lentamente até que o material se solidifique, permitindo aos átomos se organizarem numa configuração com menor energia in-

terna, reduzindo defeitos no material final [4]. O pseudocódigo do algoritmo está apresentado no Algoritmo 2.

Algorithm 2: Simulated Annealing

Input: solução inicial, t_0 (temperatura inicial), α (taxa de decaimento de temperatura), *num_iter* (número de iterações)

Output: solução ótima

```
1  while condição de parada não for satisfeita do
2      for 1 : num_iter do
3          Escolher aleatoriamente um vizinho s' de s
4          if f(s') < f(s) then
5              s ← s'
6              if f(s') < f(sm) then
7                  sm ← s'
8              end
9              else
10                 s ← s' com probabilidade e^{-(f(s')-f(s))/t}
11             end
12         end
13     end
14     t ← tα
15 end
16 return solução ótima
```

Acerca da geração da vizinhança, é escolhido um *cluster* mal avaliado (alto SSE) via roleta e pré-seleciona-se deste $||neighborhood||$ itens mais distantes do atual centróide. Dentre estes itens, remove-se aleatoriamente um item e o insere no *cluster* com a menor distância ao seu centróide. Em seguida, o novo conjunto é atualizado. Com estas etapas, o requisito de controle do resfriamento é cumprido. Como critério de parada, são utilizados por temperatura menor que um e tempo de processamento máximo.

3.4. Algoritmo Genético

O Algoritmo Genético (AG) é uma metaheurística inspirada pelo mecanismo de evolução natural, ao qual abrange conceitos como hereditariedade, mutação, seleção natural e recombinação [5]. O pseudocódigo do algoritmo está apresentado no Algoritmo 3.

Algorithm 3: Algoritmo Genético

Input: *pop_size* (tamanho da população), *cross_prob* (probabilidade de cruzamento), *mut_prob* (probabilidade de mutação)

Output: conjunto de soluções ótimas

```
1 sm ← s
2 while condição de parada não for satisfeita do
3 |   lista de pais ← seleção(população)
4 |   população ← reprodução(lista de pais) com probabilidade
  |    cross_prob
5 |   mutação(população) com probabilidade mut_prob
6 |   população ← descendência(população)
7 end
8 return população
```

A população inicial é construída de forma aleatória. O mecanismo de seleção é dado por torneio, onde uma quantidade de candidatos a pais são antepostos e apenas os dois melhores são selecionados para gerar uma prole. Dado que o mecanismo de cruzamento seleciona apenas dois indivíduos, esta etapa é repetida $||populacao||/2$ vezes, de forma que todos os indivíduos tenham igual chance de realizar a recombinação. O mecanismo de combinação utilizado também foi altamente inspirado na natureza, ao qual o pai doa um espermatozoide que será fecundado num óvulo doado pela mãe. Tecnicamente, o pai doa um *cluster* escolhido randomicamente e este é inserido na mãe. Em seguida, os itens presentes no *cluster* doado pelo pai são removidos do *cluster* mãe gerando um novo indivíduo. Caso ocorra de resultar no filho mais *clusters* do que o necessário, os com menor quantidade de itens são agrupados; caso contrário, os *clusters* com maior quantidade são separados ao meio. Esta técnica foi ado-

tada de forma que bons agrupamentos possam ser mantidos. O mecanismo de mutação aplicado é forte: caso o indivíduo seja selecionado, este é descartado e um novo indivíduo randômico é inserido na população. Note portanto que a mutação ocorre mais em âmbito populacional do que no indivíduo. Já acerca da descendência, apenas aos *pop_size* melhores indivíduos é permitido entrar em uma nova geração. Como critério de parada, são utilizados quantidade máxima de iterações, tempo de processamento máximo e se a população convergiu, ou seja, se todos os indivíduos estão idênticos.

4. Descrição dos Resultados dos Experimentos

4.1. Treino

Após realizar os experimentos com as configurações e bases de dados de treino, pode-se afirmar que o *Simulated Annealing* performou melhor e de maneira mais constante em relação aos demais métodos. Como pode ser avaliado na Figura 1, este quase não apresentou grandes variabilidades nos seus resultados, enquanto o GRASP (Figura 2) e o Algoritmo Genético (Figura 3) apresentaram. Ainda, pode-se observar que, embora o GRASP tenha atingido resultados mínimos menores que o *Simulated Annealing*, este apresentou grandes variabilidades e falta de inconstância nos resultados, fato este que também pode ser observado pelo Algoritmo Genético, cujas soluções ficaram visivelmente piores. Estes fatos podem ser confirmados através das cinco melhores configurações alcançadas via comparação por médias padronizadas, apresentadas nas Tabelas 1, 2 e 3 e pela melhor configuração por média e por rankeamento médio, apresentados nas Tabelas 4, 5 e 6.

Ressalta-se que acerca de parâmetros que não foram definidos pela especificação do trabalho, estes foram escolhidos empiricamente. A saber, no *Simulated Annealing*, o parâmetro `size_neighborhood` foi aplicado como sendo 5% do tamanho do conjunto de dados. Este parâmetro é aplicado nas operações entre as vizinhanças; e no Algoritmo Genético, foi utilizado como número máximo de 100 iterações.

Todos os resultados apresentados nesta sessão referem-se à média de 10 aplicações do método ao problema proposto.

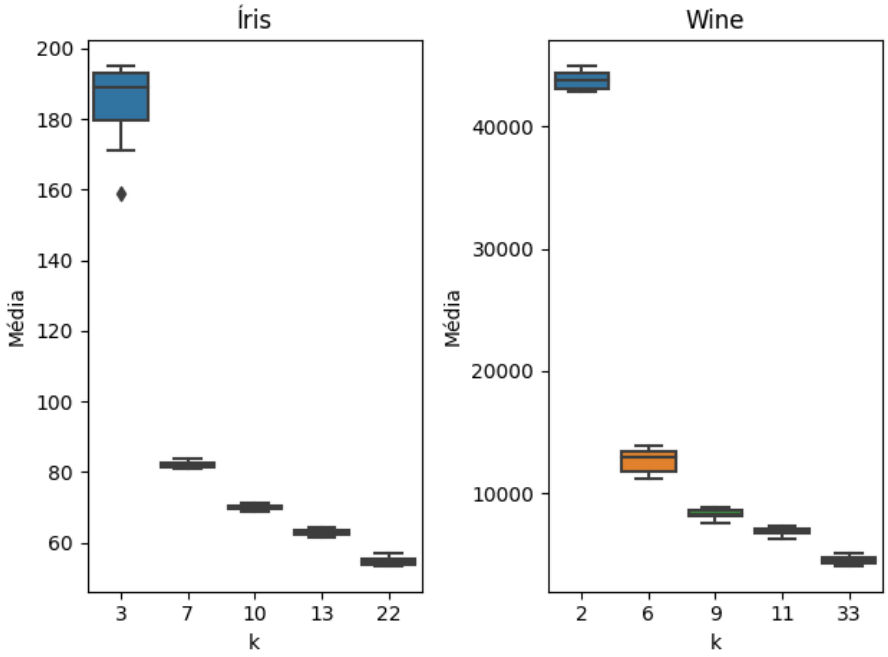

Figure 1: Comparação entre resultados de média para as bases Íris e Wine em diferentes valores de k, para o método *Simulated Annealing*.

Configuração			Resultados		
Problema	t_0	α	`iter_max`	Z-score	time [s]
Íris (k=3)	500	0.95	350	-2.57	1.23
Íris (k=10)	100	0.95	350	-2.33	1.33
Wine (k=11)	100	0.95	350	-2.14	1.21
Wine (k=9)	500	0.95	350	-1.94	1.15
Wine (k=6)	500	0.70	350	-1.83	1.22

Table 1: Valores de hiperparâmetros das cinco melhores configurações obtidas via comparação de médias padronizadas, para o método *Simulated Annealing*.

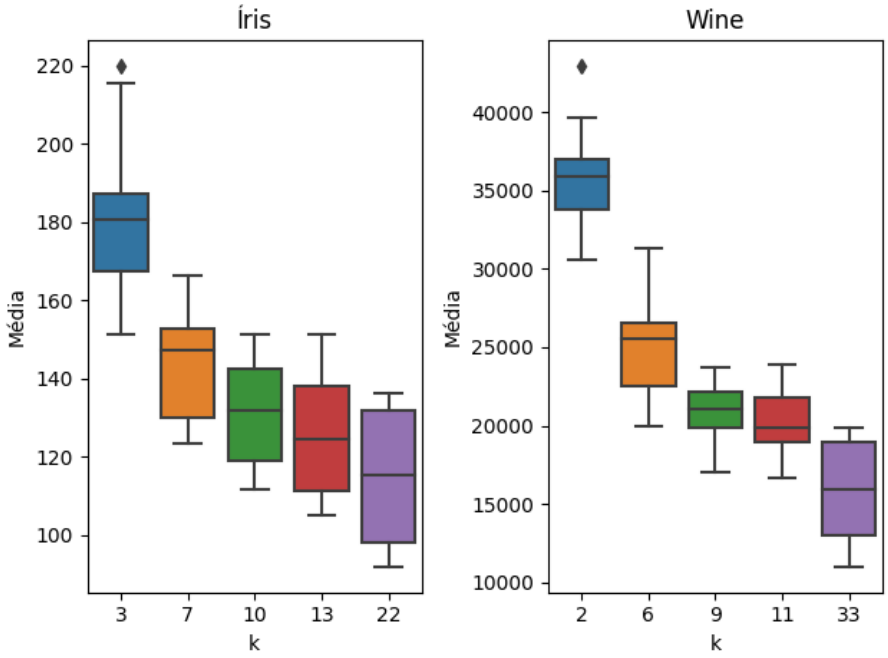

Figure 2: Comparação entre resultados de média para as bases Íris e Wine em diferentes valores de k, para o método GRASP.

Configuração			Resultados	
Problema	`iter_max`	`num_best`	Z-score	time [s]
Wine (k=9)	50	5	-2.03	1.66
Wine (k=11)	50	5	-1.89	1.41
Wine (k=2)	20	5	-1.85	2.80
Wine (k=33)	100	10	-1.73	1.25
Wine (k=6)	50	15	-1.69	1.12

Table 2: Valores de hiperparâmetros das cinco melhores configurações obtidas via comparação de médias padronizadas, para o método GRASP.

Ainda sobre as melhores configurações, observa-se que apenas o Algoritmo Genético apresentou a mesma configuração de parâmetros, o que pode ser car-

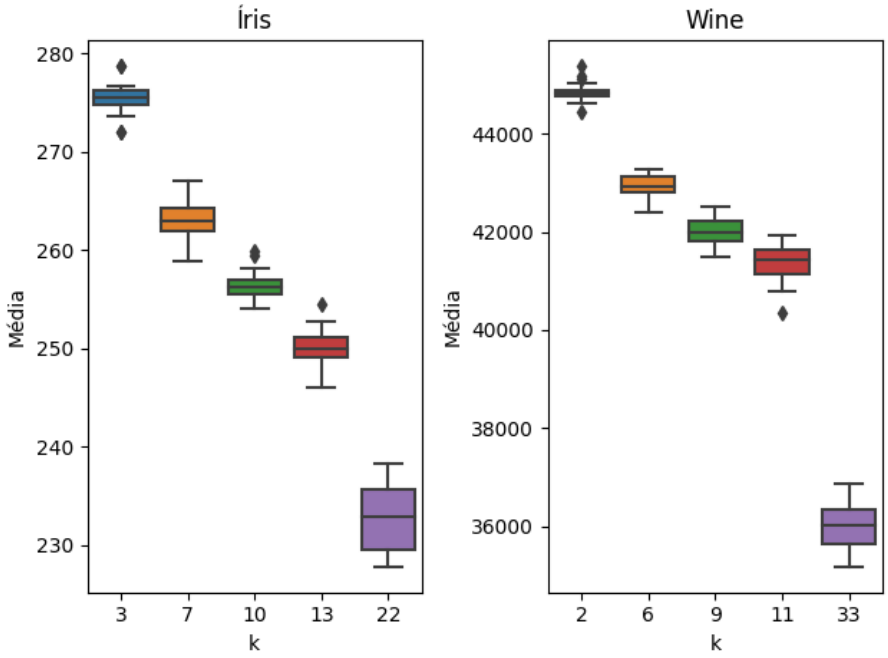

Figure 3: Comparação entre resultados de média para as bases Íris e Wine em diferentes valores de k, para o método Algoritmo Genético.

Configuração				Resultados	
Problema	`pop_size`	`cross_ratio`	`mut_ratio`	Z-score	time [s]
Wine (k=11)	50	0.95	0.1	-2.47	1.02
Wine (k=6)	50	0.85	0.1	-2.10	1.02
Íris (k=7)	10	0.75	0.1	-2.07	1.01
Wine (k=2)	50	0.95	0.1	-2.01	1.02
Wine (k=9)	10	0.75	0.1	-1.98	1.02

Table 3: Valores de hiperparâmetros das cinco melhores configurações obtidas via comparação de médias padronizadas, para o método Algoritmo Genético.

acterizado como coincidência, visto que, o efetuando novamente, este fato não foi observado.

	Configuração			Resultados		
	t_0	α	`iter_max`	mean	std	rank
média	100	0.85	350	7382.95	13136.38	6.80
rank	100	0.70	350	7388.91	13254.58	5.80

Table 4: Melhores configurações por média e por ranqueamento médio, para o método *Simulated Annealing*.

	Configuração		Resultados		
	`iter_max`	`num_best`	mean	std	rank
média	200	5	10273.20	11788.68	3.50
rank	50	5	10652.80	12135.71	3.30

Table 5: Melhores configurações por média e por ranqueamento médio, para o método GRASP.

	Configuração			Resultados		
	`pop_size`	`cross_ratio`	`mut_ratio`	mean	std	rank
média	50	0.75	0.20	20579.71	21554.80	4.70
rank	50	0.75	0.20	20579.71	21554.80	4.70

Table 6: Melhores configurações por média e por ranqueamento médio, para o método Algoritmo Genético.

Acerca dos tempos de processamento, o método Algoritmo Genético foi o que demonstrou mais constância, enquanto o GRASP demonstrou mais variabilidades, embora nenhum tenha ultrapassado muito mais que o tempo de processamento máximo previsto de $1s$. Isto pode ser observado pelas Figuras 4, 5, 6.

O rankeamento médio para cada configuração de teste é apresentado nas Tabelas 7, 8, 9. Comparando-se as linhas entre cada coluna, não é possível identificar nenhum padrão aparente ou configuração cujo rankeamento se apresentou constante. Isto confirma o acaso mencionado anteriormente.

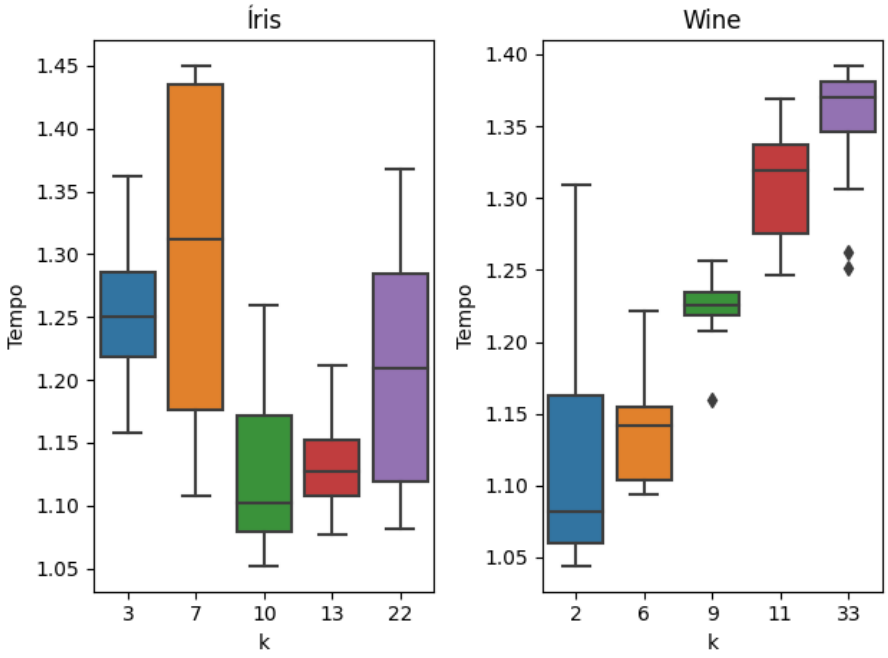

Figure 4: Comparação entre resultados de tempo de processamento para as bases Íris e Wine em diferentes valores de k, para o método *Simulated Annealing*.

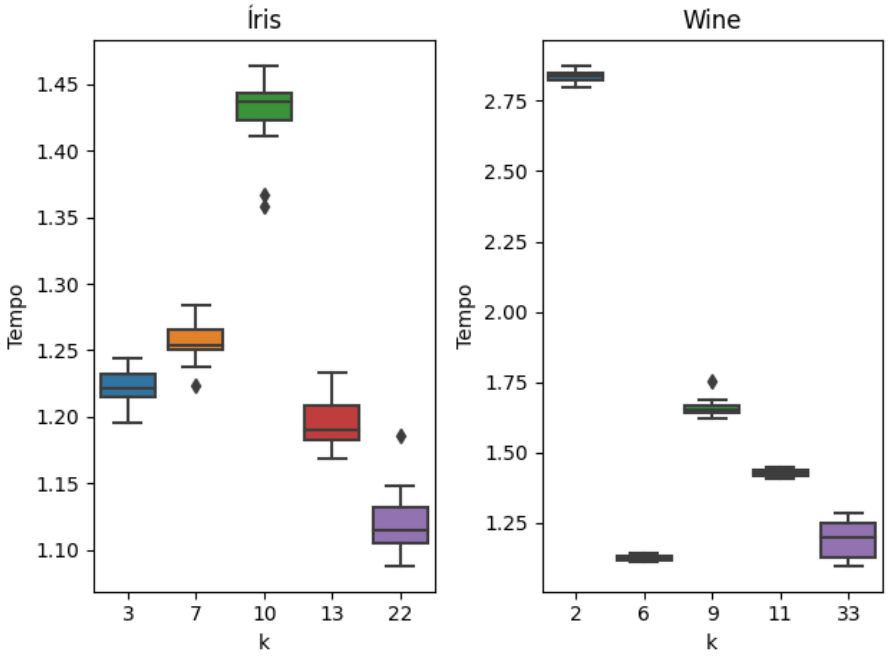

Figure 5: Comparação entre resultados de tempo de processamento para as bases Íris e Wine em diferentes valores de k, para o método GRASP.

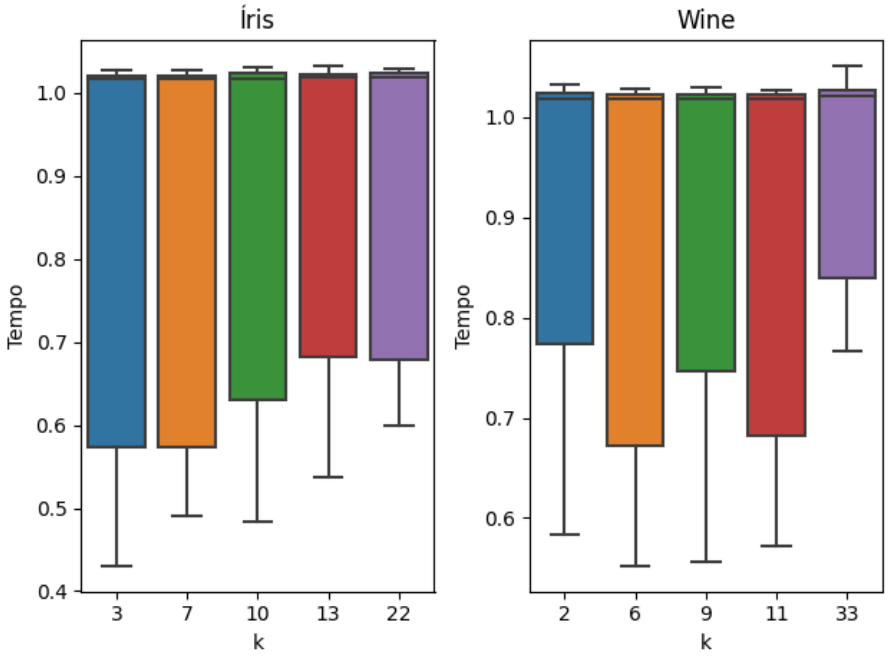

Figure 6: Comparação entre resultados de tempo de processamento para as bases Íris e Wine em diferentes valores de k, para o método Algoritmo Genético.

t	alfa	iter_max	rank	Problema									
				Íris					Wine				
		k		3	7	10	13	22	2	6	9	11	33
500.0	0.95	350.0	14.1	16.0	18.0	16.0	5.0	18.0	16.0	14.0	15.0	15.0	8.0
500.0	0.85	350.0	12.1	1.0	15.0	14.0	16.0	9.0	15.0	17.0	18.0	12.0	4.0
500.0	0.7	350.0	11.4	17.0	1.0	17.0	3.0	16.0	12.0	11.0	12.0	9.0	16.0
100.0	0.95	350.0	10.7	14.0	16.0	18.0	12.0	15.0	8.0	5.0	2.0	2.0	15.0
100.0	0.85	350.0	6.8	7.0	10.0	12.0	6.0	10.0	2.0	1.0	4.0	3.0	13.0
100.0	0.7	350.0	5.8	4.0	2.0	9.0	14.0	5.0	6.0	4.0	1.0	1.0	12.0
50.0	0.95	350.0	10.1	15.0	13.0	13.0	9.0	1.0	9.0	8.0	14.0	10.0	9.0
50.0	0.85	350.0	7.6	3.0	17.0	7.0	7.0	7.0	3.0	9.0	9.0	13.0	1.0
50.0	0.7	350.0	8.9	5.0	7.0	2.0	4.0	13.0	11.0	18.0	5.0	18.0	6.0
500.0	0.95	500.0	12.4	10.0	11.0	15.0	8.0	11.0	17.0	13.0	13.0	16.0	10.0
500.0	0.85	500.0	10.3	2.0	8.0	11.0	2.0	6.0	13.0	16.0	17.0	14.0	14.0
500.0	0.7	500.0	11.2	8.0	5.0	6.0	15.0	4.0	18.0	12.0	16.0	11.0	17.0
100.0	0.95	500.0	6.7	6.0	3.0	8.0	10.0	12.0	4.0	6.0	6.0	7.0	5.0
100.0	0.85	500.0	8.3	11.0	14.0	10.0	1.0	17.0	1.0	2.0	3.0	6.0	18.0
100.0	0.7	500.0	8.1	13.0	9.0	3.0	18.0	8.0	5.0	3.0	7.0	4.0	11.0
50.0	0.95	500.0	12.1	18.0	12.0	5.0	17.0	14.0	10.0	10.0	11.0	17.0	7.0
50.0	0.85	500.0	6.8	9.0	4.0	1.0	11.0	3.0	14.0	7.0	8.0	8.0	3.0
50.0	0.7	500.0	7.6	12.0	6.0	4.0	13.0	2.0	7.0	15.0	10.0	5.0	2.0

Table 7: Ranqueamento em cada problema, para o método *Simulated Annealing*.

iter_max	numBest	rank	Problema										
			Íris						Wine				
	k												
20	5	4.0	3.0	7.0	4.0	2.0	4.0	1.0	2.0	10.0	4.0	3.0
20	10	9.6	8.0	16.0	9.0	8.0	7.0	17.0	10.0	7.0	5.0	9.0
20	15	15.6	14.0	15.0	18.0	14.0	18.0	13.0	17.0	18.0	14.0	15.0
50	5	3.3	1.0	4.0	2.0	3.0	2.0	2.0	7.0	3.0	3.0	6.0
50	10	11.1	9.0	13.0	13.0	12.0	9.0	15.0	11.0	9.0	9.0	11.0
50	15	14.1	15.0	11.0	16.0	16.0	15.0	9.0	18.0	15.0	13.0	13.0
100	5	4.6	10.0	5.0	5.0	5.0	6.0	4.0	4.0	1.0	1.0	5.0
100	10	9.4	7.0	9.0	7.0	9.0	12.0	12.0	8.0	8.0	12.0	10.0
100	15	14.2	4.0	12.0	14.0	17.0	16.0	16.0	14.0	17.0	16.0	16.0
200	5	3.5	12.0	2.0	3.0	1.0	1.0	3.0	1.0	2.0	6.0	4.0
200	10	9.6	11.0	6.0	10.0	11.0	8.0	8.0	16.0	11.0	8.0	7.0
200	15	16.4	17.0	17.0	17.0	18.0	17.0	18.0	12.0	16.0	18.0	14.0
350	5	6.0	13.0	1.0	1.0	6.0	5.0	14.0	6.0	6.0	7.0	1.0
350	10	9.2	2.0	14.0	11.0	10.0	10.0	7.0	15.0	5.0	10.0	8.0
350	15	13.0	16.0	10.0	12.0	13.0	13.0	11.0	9.0	13.0	15.0	18.0
500	5	4.0	6.0	3.0	6.0	4.0	3.0	5.0	5.0	4.0	2.0	2.0
500	10	8.5	5.0	8.0	8.0	7.0	11.0	6.0	3.0	14.0	11.0	12.0
500	15	14.9	18.0	18.0	15.0	15.0	14.0	10.0	13.0	12.0	17.0	17.0

Table 8: Ranqueamento em cada problema, para o método GRASP.

pop_size	cross_ratio	mut_ratio	rank	Problema									
				Íris					Wine				
	k			3	7	10	13	22	2	6	9	11	33
10.0	0.75	0.1	13.1	15.0	11.0	17.0	18.0	2.0	10.0	15.0	15.0	16.0	12.0
10.0	0.85	0.1	13.9	8.0	17.0	18.0	15.0	16.0	18.0	9.0	18.0	2.0	18.0
10.0	0.95	0.1	13.9	18.0	14.0	16.0	8.0	17.0	5.0	17.0	13.0	17.0	14.0
30.0	0.75	0.1	8.7	3.0	6.0	4.0	11.0	13.0	13.0	4.0	8.0	14.0	11.0
30.0	0.85	0.1	9.3	11.0	7.0	12.0	13.0	9.0	4.0	11.0	7.0	11.0	8.0
30.0	0.95	0.1	14.0	17.0	10.0	10.0	10.0	18.0	17.0	18.0	9.0	18.0	13.0
50.0	0.75	0.1	9.9	13.0	16.0	5.0	9.0	8.0	16.0	16.0	6.0	6.0	4.0
50.0	0.85	0.1	12.3	9.0	13.0	11.0	12.0	12.0	9.0	14.0	17.0	9.0	17.0
50.0	0.95	0.1	13.6	6.0	18.0	9.0	17.0	14.0	14.0	13.0	14.0	15.0	16.0
10.0	0.75	0.2	8.5	5.0	8.0	15.0	5.0	10.0	12.0	10.0	11.0	4.0	5.0
10.0	0.85	0.2	5.3	4.0	4.0	7.0	2.0	4.0	3.0	3.0	16.0	7.0	3.0
10.0	0.95	0.2	7.0	16.0	5.0	6.0	1.0	6.0	11.0	6.0	12.0	5.0	2.0
30.0	0.75	0.2	5.2	1.0	15.0	2.0	7.0	3.0	2.0	1.0	3.0	3.0	15.0
30.0	0.85	0.2	7.5	14.0	9.0	1.0	3.0	11.0	6.0	5.0	4.0	13.0	9.0
30.0	0.95	0.2	9.4	12.0	2.0	14.0	14.0	5.0	15.0	7.0	10.0	8.0	7.0
50.0	0.75	0.2	4.7	7.0	3.0	13.0	4.0	7.0	1.0	8.0	2.0	1.0	1.0
50.0	0.85	0.2	7.8	10.0	12.0	8.0	6.0	1.0	8.0	12.0	5.0	10.0	6.0
50.0	0.95	0.2	6.9	2.0	1.0	3.0	16.0	15.0	7.0	2.0	1.0	12.0	10.0

Table 9: Ranqueamento em cada problema, para o método Algoritmo Genético.

A partir dos resultados apresentados, espera-se que o método *Simulated Annealing* apresente-se mais eficiente nos problemas de teste. Isto pode ocorrido devido à sua estrutura simplificada e mecanismo de geração de vizinhança eficiente, que pode ocorrer e performar de maneira adequada dentro do tempo proposto. O fato de o Algoritmo Genético ter performado mal pode ser devido à sua estrutura complexa, dependendo de mais tempo para realizar mais gerações, bem como ajustar mecanismos genéticos mais eficientes ao problema. Já o GRASP, sendo um algoritmo originalmente criado para resolver problemas combinatoriais, performou fora das expectativas, visto que houve considerável manifestação da parte aleatória.

Com os melhores resultados obtidos através das Tabelas 4, 5 e 6, será realizada a fase de testes.

4.2. Teste

Com os modelos treinados, agora estes serão testados e comparados ao modelo *KMeans* da biblioteca *Sklearn*[1]. A melhor configuração em cada método foi escolhida de acordo com os valores de médias padronizadas, visto que estas se apresentaram um desvio-padrão menor. Obviamente, no caso do Algoritmo Genético, ambos os melhores resultados, de média e rankeamento, apresentaram-se iguais e portanto este foi escolhido. Disto isso, os parâmetros utilizados foram:

- *Simulated Annealing*: $t_0 = 100$, $\alpha = 0.85$, `iter_max` $= 350$, `size_neighborhood` $= 0.05||\text{dataset}||$;
- GRASP: `iter_max` $= 200$, `num_best` $= 5$;
- Algoritmo Genético: `pop_size` $= 50$, `cross_ratio` $= 0.75$, `mut_ratio` $= 0.2$, `iter_max` $= 100$.

[1] https://scikit-learn.org/stable/

Todos os resultados apresentados nesta sessão referem-se à média de 20 aplicações do método ao problema proposto. Resultados iniciais estão apresentados na Tabela 10.

	Simulated Annealing	GRASP	Algoritmo Genético	*KMeans*
Média	3213.01	14389.63	13370.23	1876.61
Desvio-Padrão	6270.77	19551.55	18184.32	3449.57
Tempo [s]	1.21	3.73	1.03	0.09
Z-Score	-0.59	1.01	0.84	-1.26
Rank	2.06	4.00	2.93	1.00

Table 10: Resultados médios obtidos por cada método.

Já é sabido de demonstrações realizadas em sala de aula que apenas analisar as médias obtidas por cada método não é uma maneira adequada para se identificar qual performou melhor, embora já espera-se, de acordo com a Tabela 10, que o *KMeans* tenha melhores resultados. Mesmo assim, computou-se o teste de *t-student* entre os pares de variáveis, o qual está apresentado na Tabela 11. Ressalta-se ainda que é apresentado apenas a métrica do p-valor pois o número de amostras é pequeno (inferior a 30).

	Simulated Annealing	GRASP	Algoritmo Genético	KMeans
Simulated Annealing		0.004822	0.006150	0.318808
GRASP	0.004822		0.837825	0.001247
Algoritmo Genético	0.006150	0.837825		0.001451
KMeans	0.318808	0.001247	0.001451	

Table 11: Resultados de p-valores obtidos pelo teste de *t-student*.

Para um intervalo de significância de 95%, temos um valor α de 0.05, o que nos leva a rejeitar a hipótese nula (H_0) sobre a maior parte dos métodos implementados (*Simulated Annealing*, GRASP e Algoritmo Genético), isto é, há diferença significativa entre os resultados de p-valores apresentados por estes e α. Entretanto, em alguns casos o p-valor está levemente mais elevado. Testaremos se estão elevados o bastante com o teste de Wilcoxon pareado. Este teste é utilizado para comparar se as medidas de posição de duas amostras são iguais no caso em que as amostras são dependentes. Os resultados estão presentados na Tabela 12

	Simulated Annealing	GRASP	Algoritmo Genético	KMeans
Simulated Annealing		≈ 0	≈ 0	≈ 0
GRASP	≈ 0		≈ 0	≈ 0
Algoritmo Genético	≈ 0	≈ 0		≈ 0
KMeans	≈ 0	≈ 0	≈ 0	

Table 12: Resultados de p-valores obtidos pelo teste de Wilcoxon.

Novamente, como os p-valores estão virtualmente muito próximos de zero, podemos rejeitar nossa hipótese nula de que a mediana é virtualmente zero, e assim conseguiremos distinguir os quatro métodos e poder escolher o melhor.

Retornando portanto à Tabela 10 e observando o *Z-Score* e rank médios, podemos concluir que o *KMeans* apresentou-se como o melhor método em relação aos demais, e das metaheurísticas que foram implementadas, o *Simulates Annealing* foi o que melhor performou, conforme a nossa suposição inicial.

A título de curiosidade, aplicando-se os métodos aqui apresentados com os melhores parâmetros no banco de dados Íris apenas nas *features Sepal length* e *Sepal width*, com um $k = 3$, obtêm-se os resultados bidimensionais apresentados nas Figuras 7, 8, 9 e 10. Claramente a decisão de melhor método pode ser

afirmada.

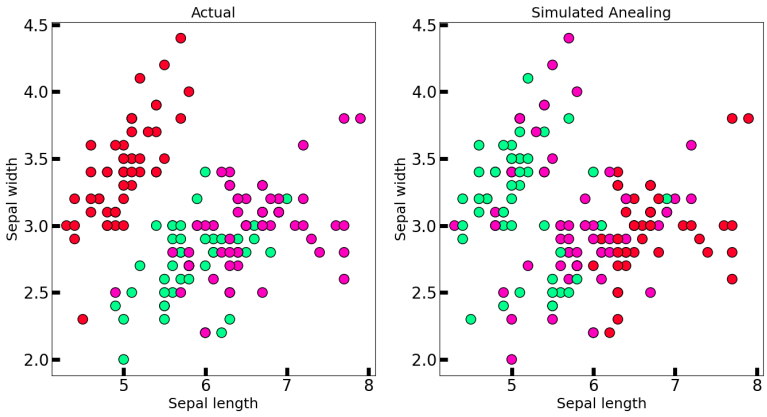

Figure 7: Comparação entre resultados equivalentes com $k = 3$ na base de dados Íris e *features* *Sepal length* e *Sepal width*, para o método *Simulated Annealing*.

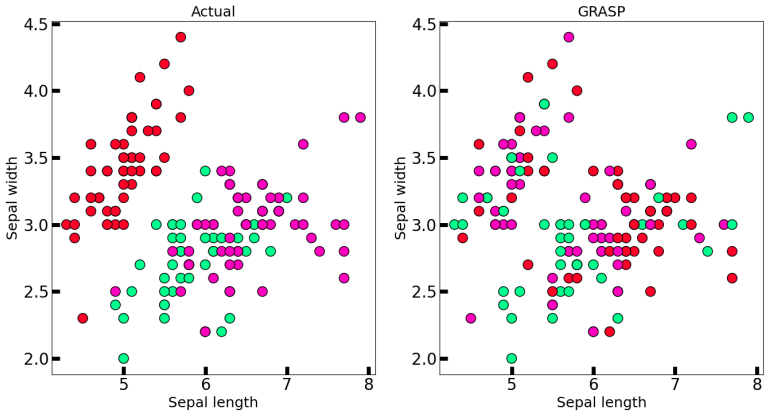

Figure 8: Comparação entre resultados equivalentes com $k = 3$ na base de dados Íris e *features* *Sepal length* e *Sepal width*, para o método GRASP

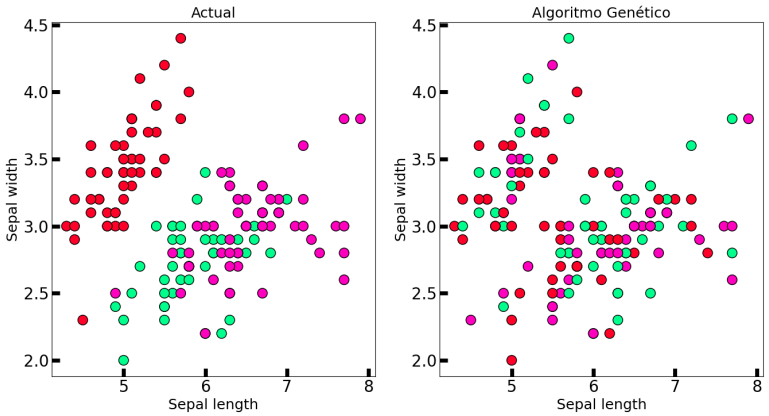

Figure 9: Comparação entre resultados equivalentes com $k = 3$ na base de dados Íris e *features Sepal length* e *Sepal width*, para o método Algoritmo Genético

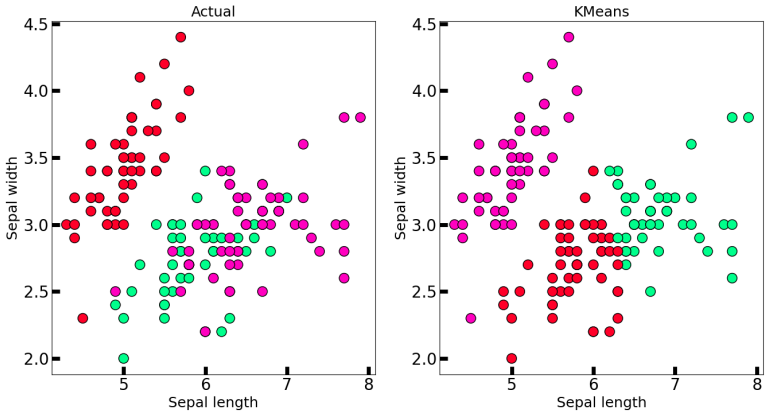

Figure 10: Comparação entre resultados equivalentes com $k = 3$ na base de dados Íris e *features Sepal length* e *Sepal width*, para o método KMeans.

5. Conclusões e possíveis melhorias

Após finalizar a análise, foi constatado que o método *KMeans* da biblioteca *Sklearn* apresentou-se superior às demais metaheurísticas implementadas para o problema do agrupamento. Isto ocorreu pois este método é, dentre os existentes na literatura, o mais sabido que melhor resolve esta classe de problemas, além

de possuir uma série de otimizações e técnicas de contornar erros que podem vir a aparecer.

Não obstante, destaca-se o bom desenvolvimento do *Simulated Annealing* quando comparado às demais metaheurísticas. Isto pode ter se dado ao fato de que neste método foi possível implementar uma técnica forte de construção de vizinhança, ao qual, além de a solução corrente ser melhorada ao retirar o item que atribui maior peso ao *cluster*, ainda procurou-se inseri-lo ao local mais adequado. Como grande parte da análise aqui presente era avaliar a aplicação de diferentes técnicas em diferentes metaheurísticas, o que foi realizado no *Simulated Annealing*, futuramente, poderia ser aplicada aos demais algoritmos. Ainda, melhorias de código objetivanto otimizar estruturas de dados quanto ao tempo de processamento e uso da memória também podem ser realizadas.

References

[1] Wikipédia, Clustering — Wikipédia, a enciclopédia livre, [Online, acessado em 16 mar. 2021] (2020).
URL `https://pt.wikipedia.org/w/index.php?title=Clustering&oldid=58054039`

[2] Wikipédia, Meta-heurística — Wikipédia, a enciclopédia livre, [Online, acessado em 16 mar. 2021] (2020).
URL `https://pt.wikipedia.org/w/index.php?title=Meta-heur%C3%ADstica&oldid=58878089`

[3] Wikipédia, GRASP — Wikipédia, a enciclopédia livre, [Online, acessado em 16 mar. 2021] (2017).
URL `https://pt.wikipedia.org/w/index.php?title=GRASP&oldid=48585662`

[4] Wikipédia, Simulated annealing — Wikipédia, a enciclopédia livre, [Online, acessado em 16 mar. 2021] (2019).

URL https://pt.wikipedia.org/w/index.php?title=Simulated_annealing&oldid=54388675

[5] Wikipédia, Algoritmo genético — Wikipédia, a enciclopédia livre, [Online, acessado em 16 mar. 2021] (2019).
URL https://pt.wikipedia.org/w/index.php?title=Algoritmo_gen%C3%A9tico&oldid=55993399