Q1) A figura 1 apresenta a resposta ao degrau para o controlador obtido ao final do treinamento (tendo fitness igual a 1). Visualmente, já é possível notar que o tempo de acomodação foi inferior a 0,5s, que era o tempo demandado. Considerando a definição de tempo de subida (tempo de 10% a 90% do set point), também fica evidente que o critério de 0,04s foi atendido.

A figura 2 apresenta qual indivíduo possuía melhor fitness a cada geração (série preta) e qual o fitness médio da população a cada geração. Indivíduos com fitness 1 são notados desde antes da décima geração, mas a população por completo não chega a atingir este nível, mesmo após as 50 gerações.

A figura 3 exibe o valor de cada um dos 3 ganhos do PID para o indivíduo de melhor fitness, onde pode-se notar que eles oscilam intensamente nas primeiras gerações e vão estabilizando próximos ao valor que dá o melhor resultado.

A figura 4 apresenta a intensidade da mutação não uniforme aplicada nas gerações. Esta intensidade varia de uma geração para outra e é determinante para encontrar novas soluções de bom desempenho.

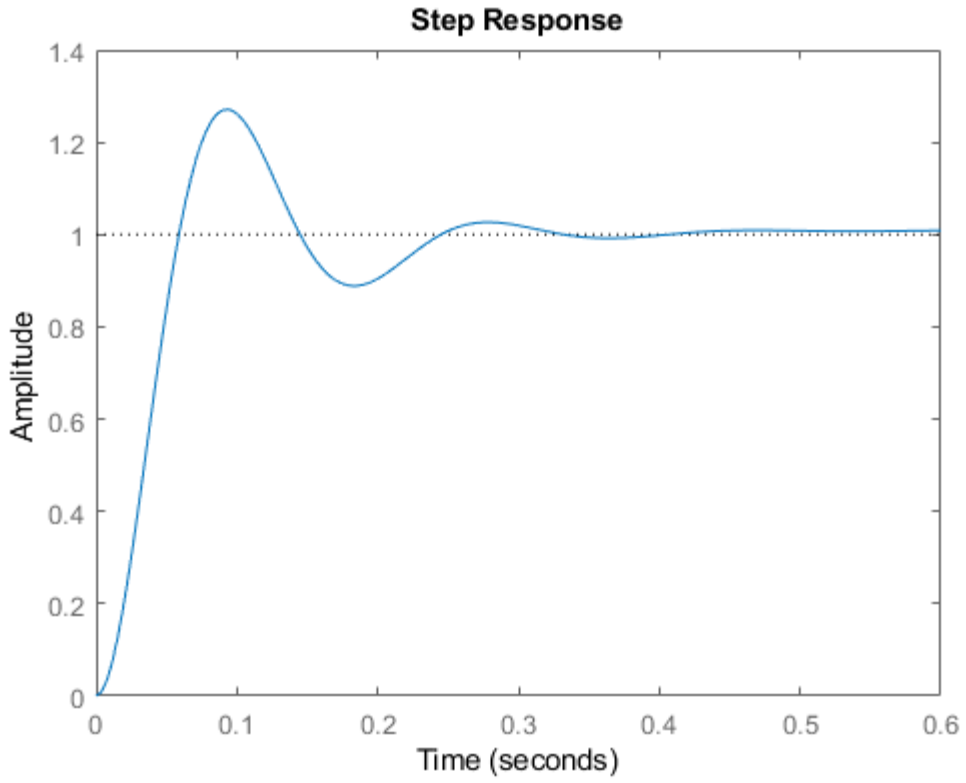

Figura 1: Amplitude x Tempo.

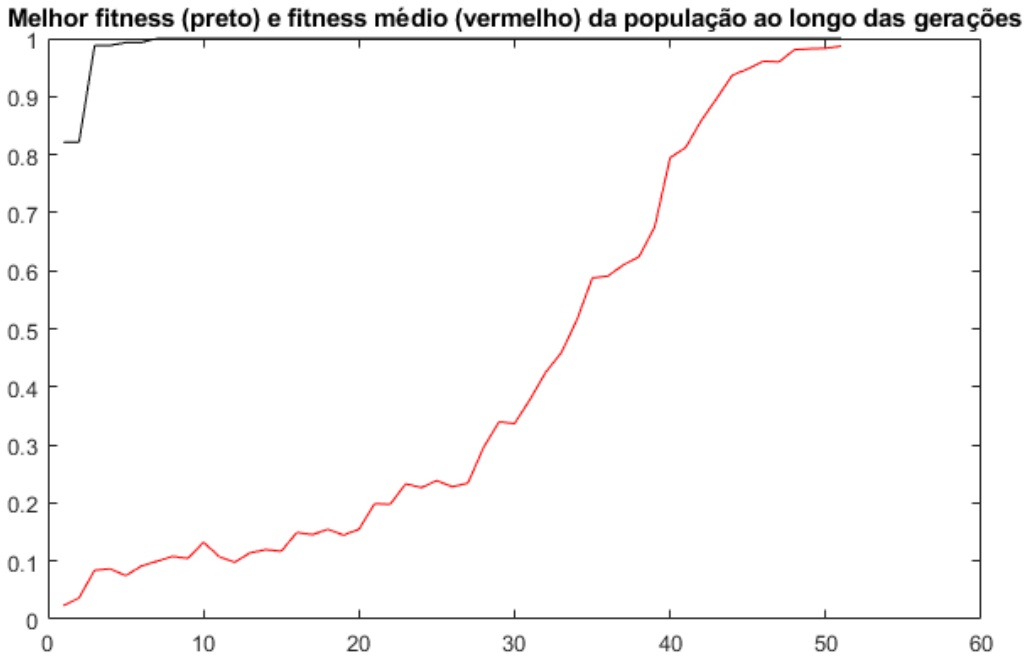

Figura 2: Indivíduo com melhor fitness e fitness médio x Geração.

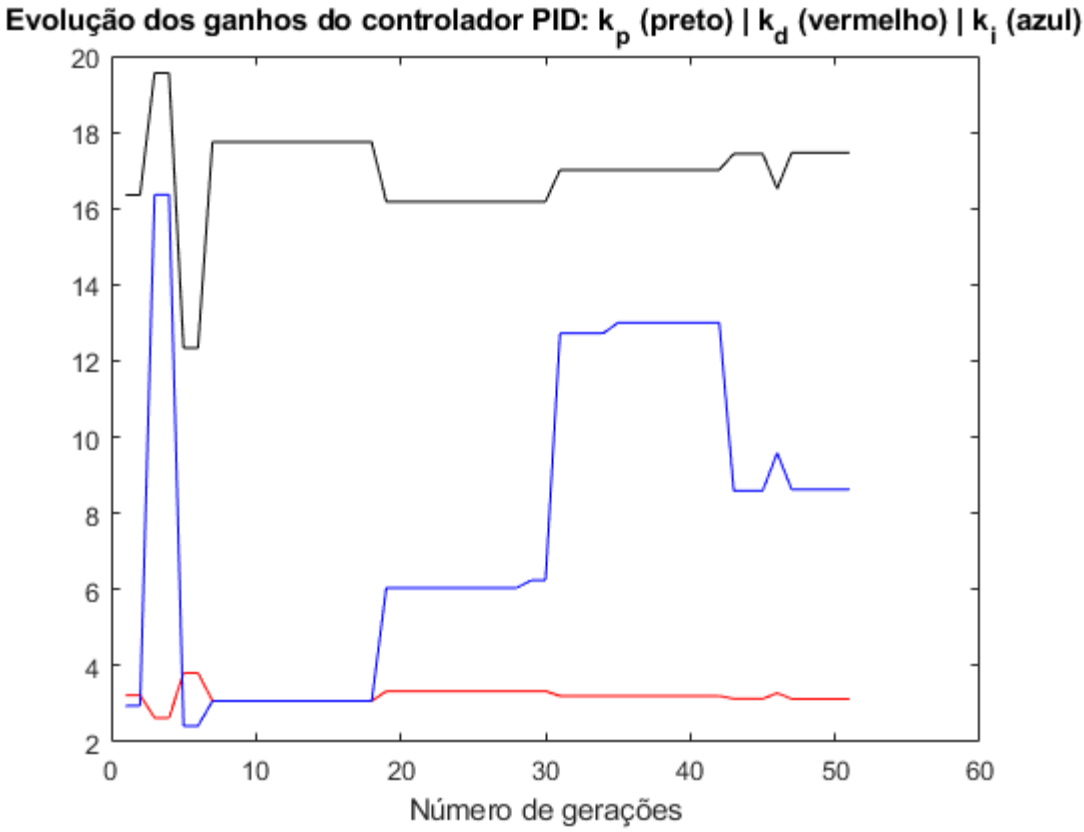

Figura 3: Ganhos do controlador PID a cada geração.

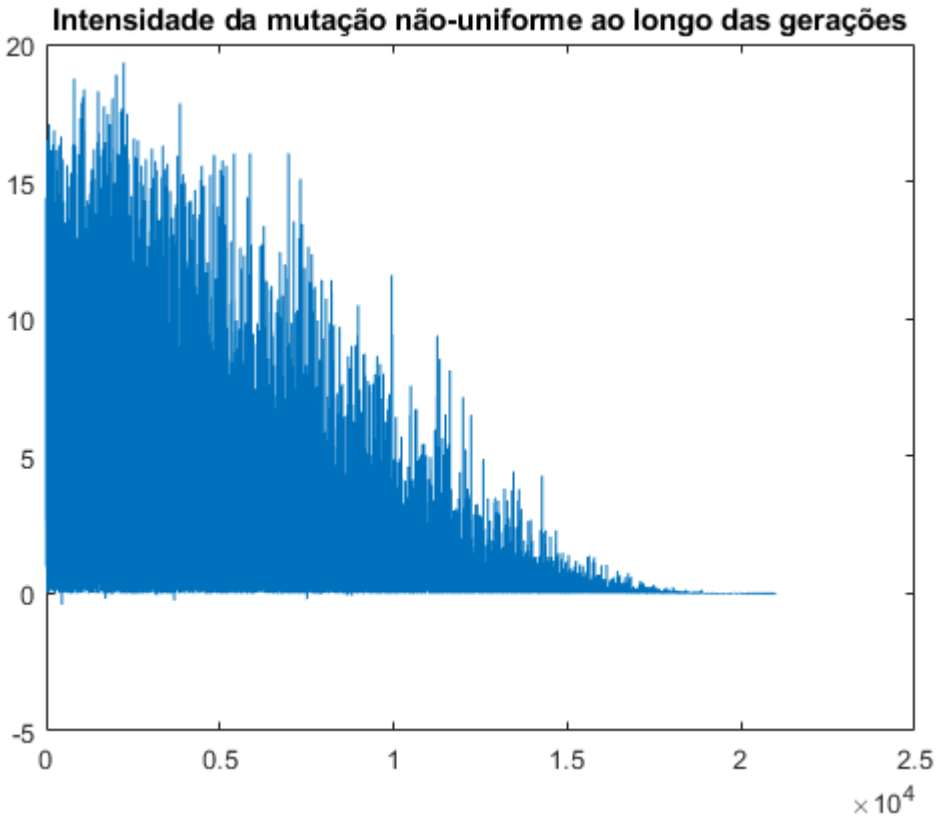

Figura 4: Mutação a cada geração.

Q3) Parte 1) Para a execução com o algoritmo de Kohonen, primeiro foi encontrada a solução para a cidade 1, com configuração inicial dos neurônios como mostrada na figura 5. Kohonen é um método de aprendizado não supervisionado, o que significa não ser necessário informar ao código valores ditos certos ou errados, a abstração se dá apenas pelos dados de entrada. Cada neurônio é ajustado para, no caso, a cidade de maior representatividade (mais próxima) e seus vizinhos também são ajustados para essa proximidade, com menos intensidade. O algoritmo insere 1 neurônio entre vizinhos de maneira a promover um neurônio representante para cada cidade. A solução final é mostrada na figura 6, tendo demandado 408 iterações e utilizado um total de 100 neurônios (como era de se esperar, já que haviam 100 cidades). O tamanho do percurso encontrado variou de 799.9697 a 821.8836 nas 5 execução realizadas, indicando que as soluções encontradas realmente podem não ser ótimas (pelo menos 4 das 5 não são), mas caracterizam boas propostas para realizar percursos eficientes.

Parte 2) Pseudo-código da solução evolutiva:

1. Primeiro, cria-se um vetor 2D com as posições de cada uma das M cidades;
2. Cria-se N rotas, cada uma iniciando em 1 cidade e conectando-se com a cidade mais próxima não percorrida até completar o percurso;
3. Usa-se as duas melhores rotas (menor percurso) para crossover e geram N/2 rotas filhas, as quais substituem as N/2 piores rotas;
4. Sob uma probabilidade P para cada uma das N/2 rotas filhas, ocorre mutação;
5. Repete-se o passo 3 até atingir o critério de parada.

Definições utilizadas no pseudo-código:

- M = 100;
- N = 20;
- P = 2%;

- Operador de Crossover: Sendo cada rota descrita pelo **índice** de cada cidade seguido da **ordem** em que ela é acessada (Ex: cidade A -> quinta, cidade B -> décima quarta...) a ser acessada, o operador pega as melhores rotas de duas a duas e troca 20% das **ordens de acesso** das cidades de uma com a outra. É importante ressaltar que não se deve trocar a ordem dos índices, mas sim as ordens de acesso nas cidades, senão poderia causar repetição de algumas cidades e não visitação de outras;

- Operador de Mutação: Há chance de 2% para cada rota “filha” gerada de que ocorra a mutação, consistindo em trocar a ordem de acesso a 10% das cidades dela mesma;

- Critério de Parada: Atingir 2000 gerações.

Nota-se que o algoritmo evolutivo aproxima-se da melhor solução encontrada logo no início da execução, antes de 250 gerações, e fica estabilizada assim até o final (2000ª geração). A figura 7 mostra a evolução do fitness ao longo das gerações.

A figura 8 exibe a melhor solução encontrada ao final do algoritmo (tamanho: 5811), enquanto que a figura 9 mostra a melhor solução ao longo da quinta geração (tamanho da rota: 14793) e deixa clara a melhora ocorrida ao longo da busca.

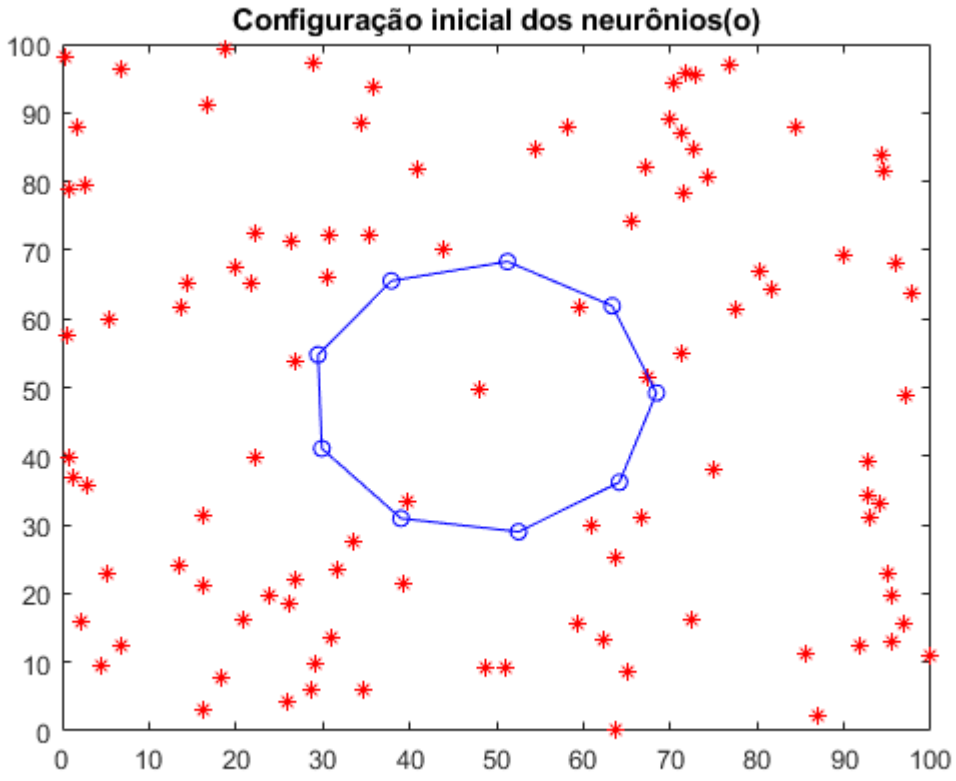

Figura 5: Configuração inicial dos neurônios distribuídos de maneira semi-uniforme.

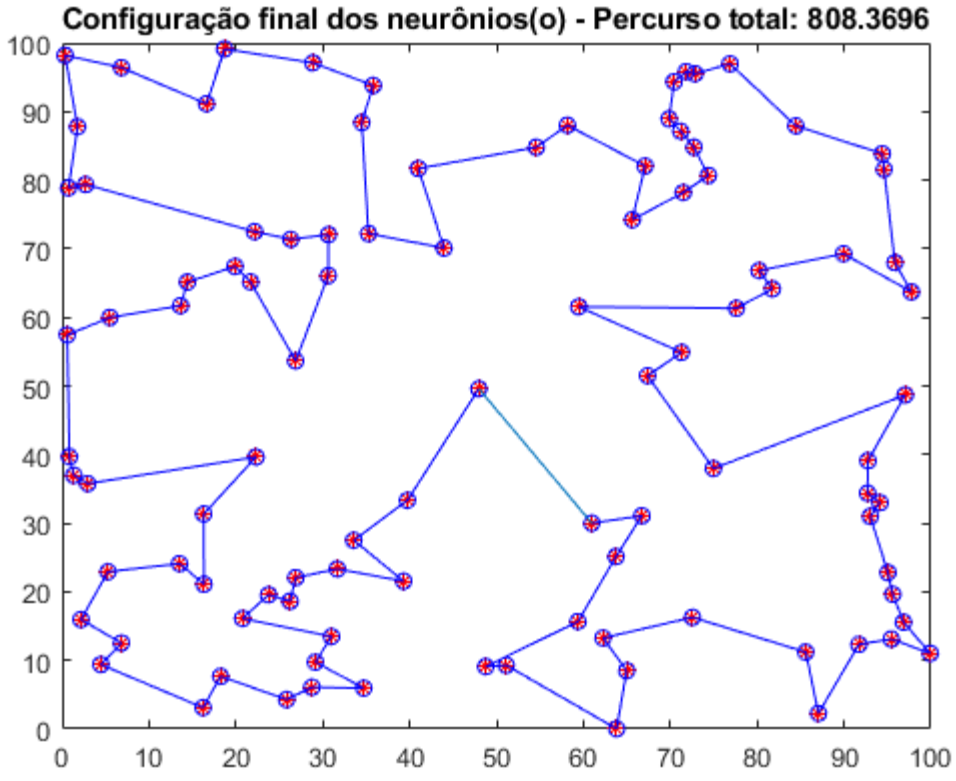

Figura 6: Configuração final dos neurônios com a solução por Kohonen.

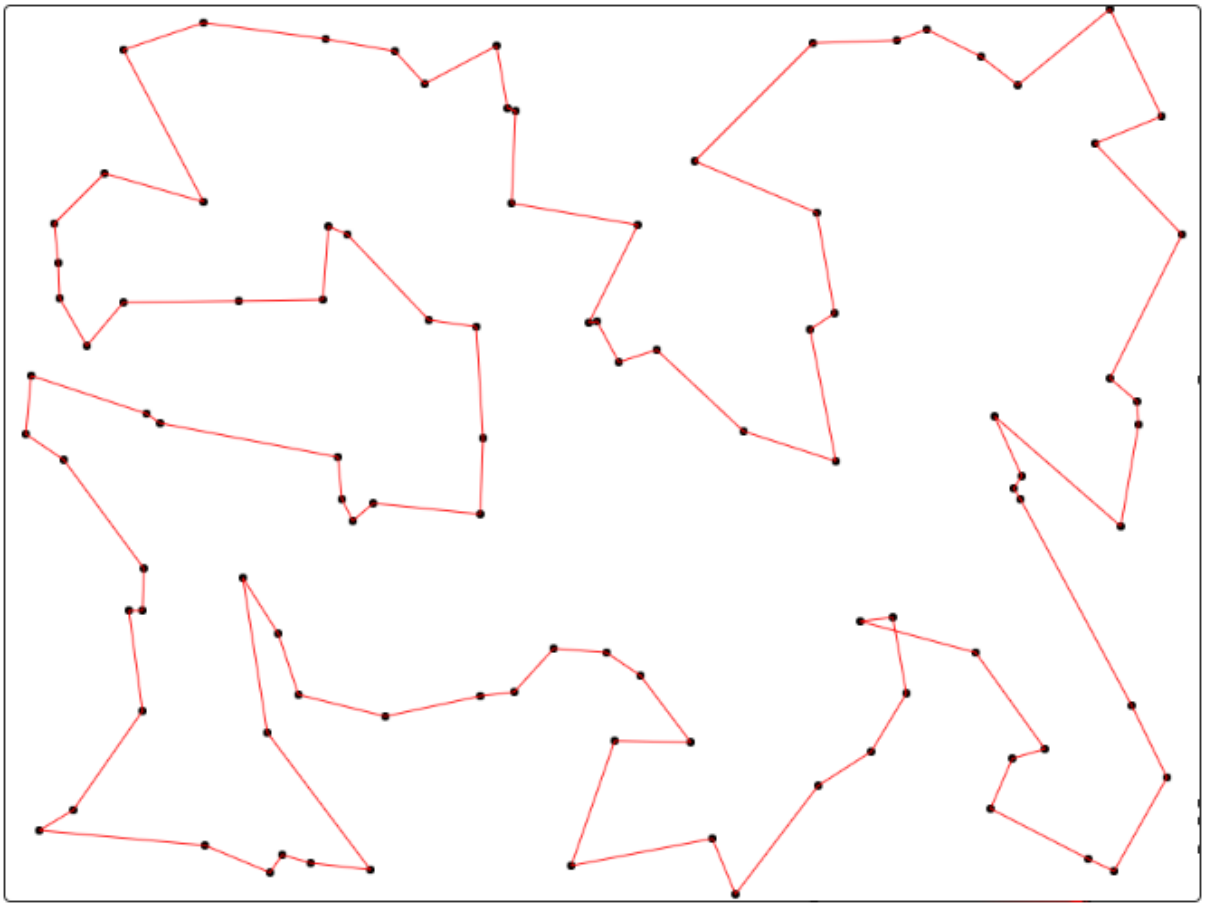

Figura 7: Melhor rota encontrada com o algoritmo evolutivo sobre 50 cidades.

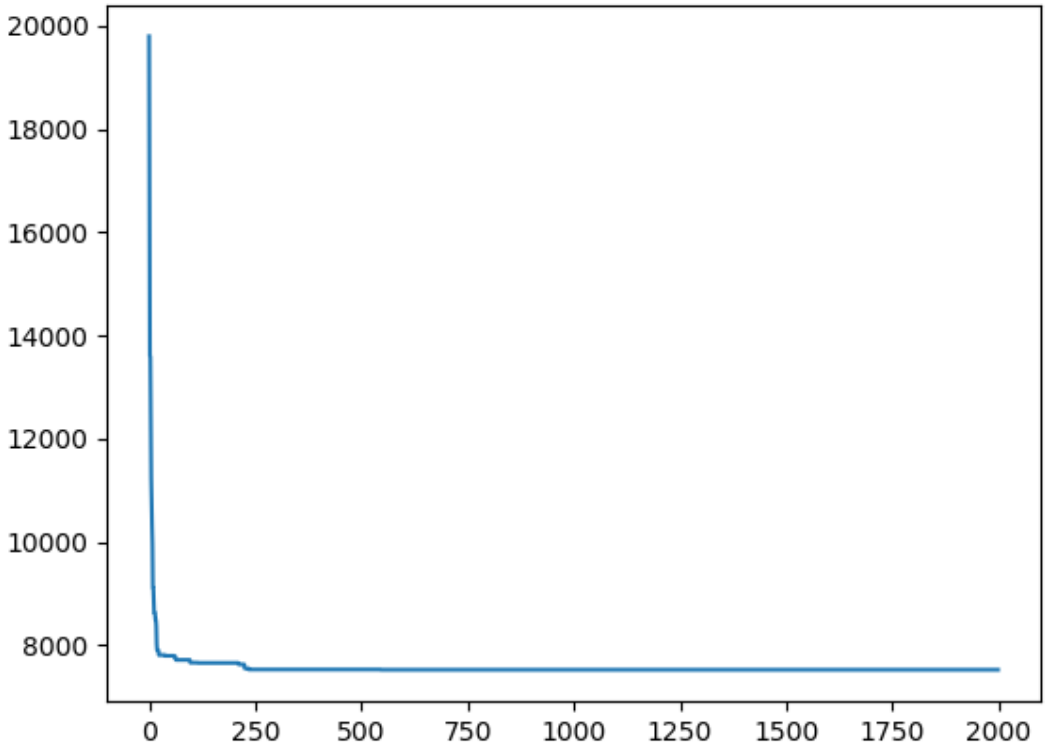

Figura 8: Melhor fitness da população por geração.

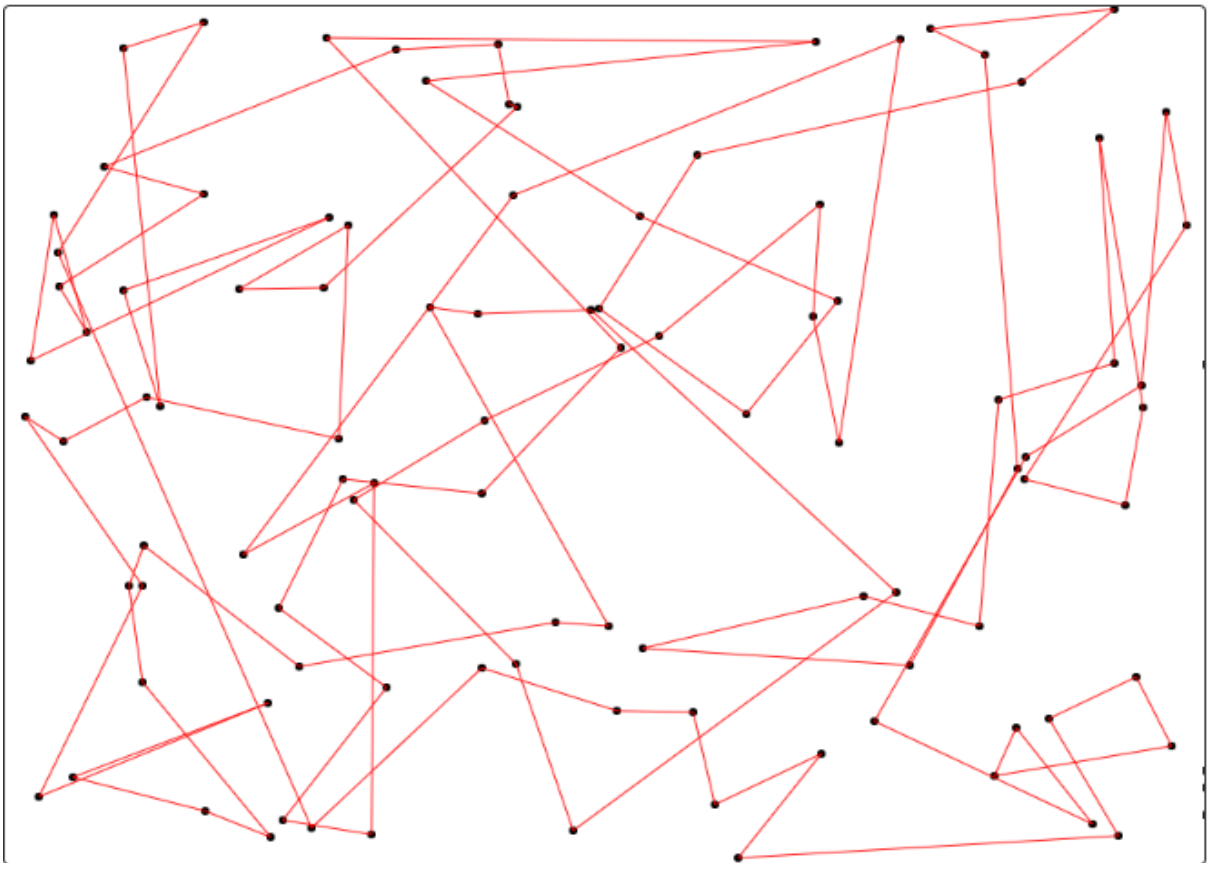

Figura 9: Melhor fitness na 5ª geração.

Q5) a) São 10000 amostras de 9 variáveis, sendo binárias (de 1 a 2), exceto a variável 2, que varia de 1 a 7.

b) Na figura 10, está a rede resultante dos dados utilizados. Nas figuras 11 e 12, estão as tabelas de probabilidades de cada nó.

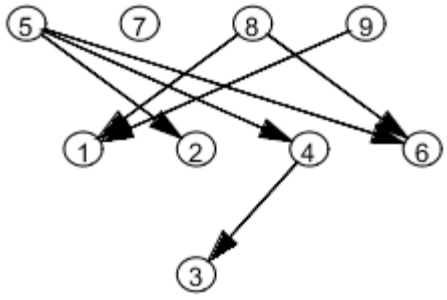

Figura 10: Rede Bayesiana resultante.

Tabela 1							
Entrada	FALSO	VERDADEIRO					
00	1	0					
01	0	1					
10	0	1					
11	1	0					
Tabela 2							
Entrada	1	2	3	4	5	6	7
0	0,2633	0,1904	0,1769	0,1641	0,1182	0,0655	0,0216
1	0,1159	0,1414	0,1283	0,1261	0,1657	0,1548	0,1677
Tabela 3							
Entrada	FALSO	VERDADEIRO					
0	0,3373	0,6627					
1	0,9802	0,0198					
Tabela 4							
Entrada	FALSO	VERDADEIRO					
0	0,5949	0,4051					
1	0,7509	0,2491					

Figura 11: Tabelas de probabilidades dos nós de 1 a 4 (de cima para baixo).

Tabela 5		
Entrada	FALSO	VERDADEIRO
----------	0,1481	0,8519
Tabela 6		
Entrada	FALSO	VERDADEIRO
00	0,8788	0,1212
01	0,5888	0,4112
10	0,0528	0,9472
11	0,1587	0,8413
Tabela 7		
Entrada	FALSO	VERDADEIRO
----------	0,5028	0,4972
Tabela 8		
Entrada	FALSO	VERDADEIRO
----------	0,482	0,518
Tabela 9		
Entrada	FALSO	VERDADEIRO
----------	0,5022	0,4978

Figura 12: Tabelas de probabilidades para os nós de 5 a 9 (de cima para baixo).

c) Pois é um atributo condicionalmente independente, não influenciando os demais nós.

d) Segundo a tabela da figura 11, 85,19%.

e) 41,12% de probabilidade para ser verdade.

f) “OU-Exclusivo”.

g) P(5 = 1 | 3 = 1) = P(5 = 1 ^ 3 = 1)/ P(3 = 1)

Numerador: **P(5 = 1 ^ 3 = 1)** = P(5 = 1 ^ 3 = 1 ^ 4 = 1) + P(5 = 1 ^ 3 = 1 ^ 4 = 0) = (0,0198*0,2491*0,8519) + (0,8519 * 0,7509 * 0,6627) = 0,428125420359.

Denominador: **P(3 = 1)** = P(5 = 1 ^ 3 = 1 ^ 4 = 1) + P(5 = 1 ^ 3 = 1 ^ 4 = 0) + P(5 = 0 ^ 3 = 1 ^ 4 = 1) + P(5 = 0 ^ 3 = 1 ^ 4 = 0) = (0,8519*0,2491*0,0198) + (0,8519*0,7509*0,6627) + (0,1481*0,4051*0,0198) + (0,1481*0,5949*0,6627) = 0,48770030556

Total: P(5 = 1 | 3 = 1) = P(5 = 1 ^ 3 = 1)/ P(3 = 1) = 0,877845 = **87,78%**

6)

item 3) Como atributos numéricos temos temperatura média, umidade média, altura da chuva mensal e precipitação 21 dias. Todos estes possuem como estatística o valor mínimo, máximo, médio e desvio padrão.

Como atributos categóricos, há local e classe, ambos possuindo estatística de classe e peso.

item 7) Taxa de acerto = 87,1429%

```
=== Confusion Matrix ===

   a  b   <-- classified as
   4  7 |  a = 0
   2 57 |  b = 1
```

Figura 13: Matriz de confusão.

A matriz de confusão representa quanta vezes um atributo da classe a (primeiro nó), na primeira coluna, foi classificado como sendo da classe a ou b. O mesmo ela apresenta para os atributos da classe b.

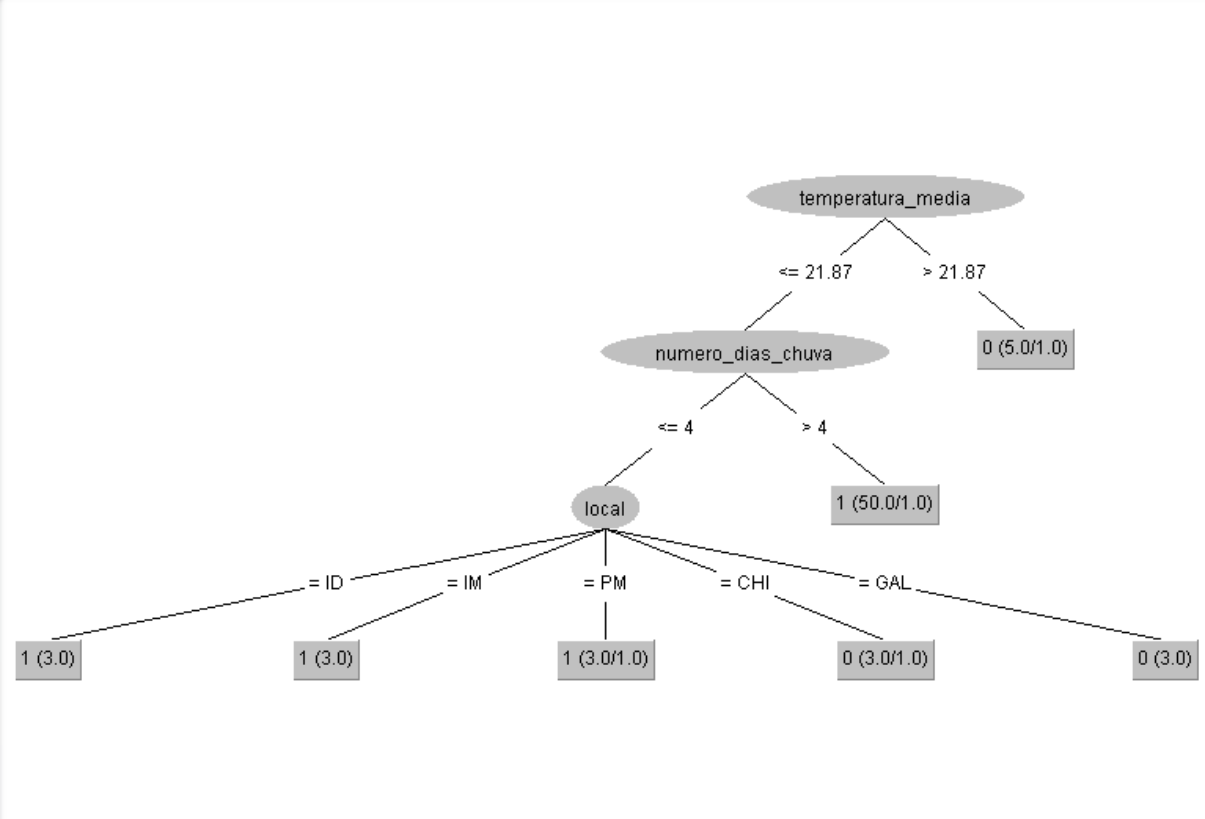

Figura 14: Árvore de decisão gerada.

item 8) De acordo com a árvore da figura 14,

Se a temperatura > 21,87, então classificada no nó 0; Se não:
Se numero dias de chuva > 4, ento nó 1; Se não:
Se CASO de ‘local’ [ID, IM, PM, CHI, GAL] respectivamente em nó [1, 1, 1, 0, 0]

Obs: Nó 1 representa que HÁ aquela espécie ali. Nó 0 representa que não há.

item 9) minNunObj: Instâncias mínimas por folha. Garante que, a cada divisão, pelo menos “minNunObj” dos ramos tenham o número mínimo de instâncias.

```
=== Confusion Matrix ===

  a  b   <-- classified as
  3  8 |  a = 0
  1 58 |  b = 1
```

Figura 15: Matriz de confusão ao alterar minNunObj para 2.

unpruned: Aplicar ou não o método de poda para redução de erro.

A árvore se tornou um único nó raiz (temperatura) e dois filhos com este método ativo, e a matriz de confusão ficou igual à figura 15. A ramificação para o numero de dias com chuva se tornou apenas uma classificação de nó 1 (há a espécie ali).

item 10) Utilizando um dataset para CPUs do próprio site do WEKA, foi gerada a árvore abaixo:

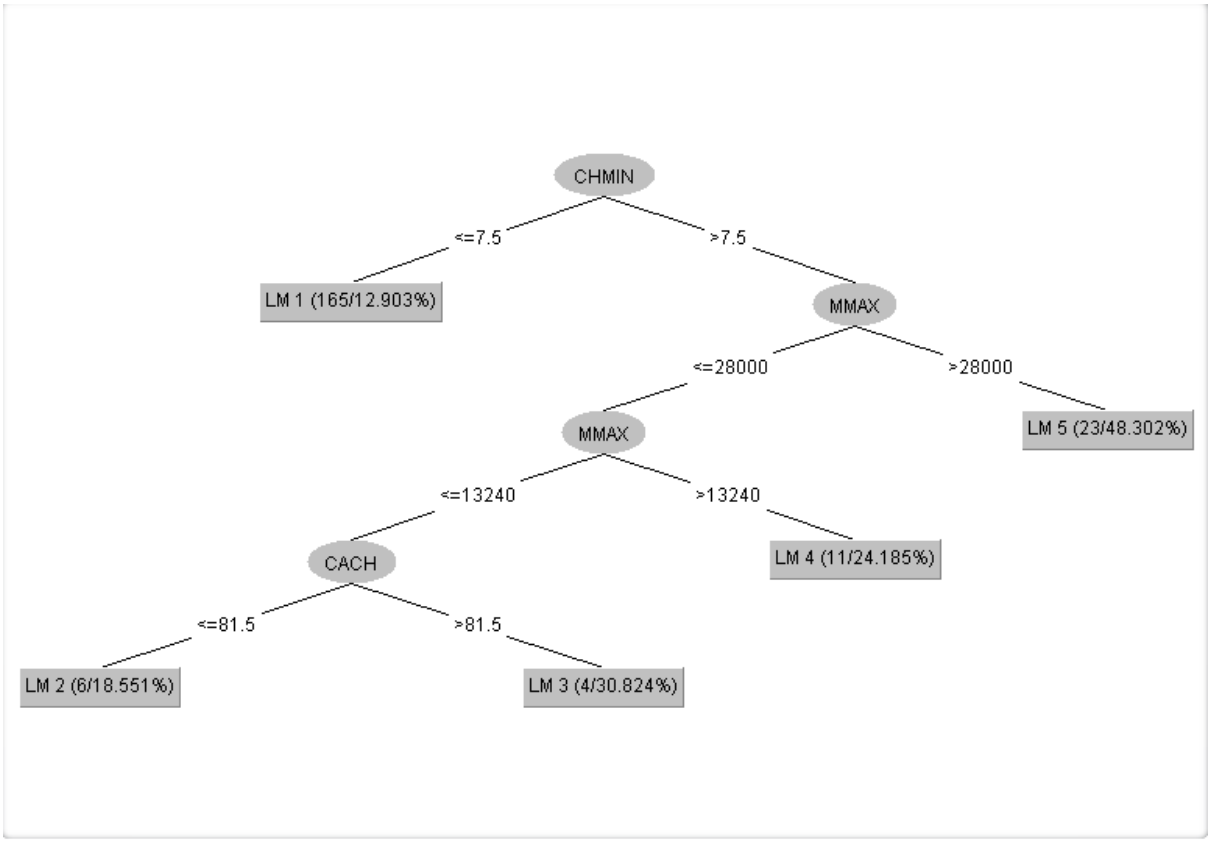

Figura 16: Árvore para definir a classe de desempenho de CPUs.

Obteve-se uma taxa de acertos de 92,74%, sendo um bom método para classificar as CPUs deste dataset quanto ao desempenho de acordo com os atributos, tais como memória máxima e memória cache.