

UNIVERSIDADE ESTADUAL PAULISTA JÚLIO DE MESQUITA FILHO – UNESP  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**COMPUTAÇÃO INSPIRADA PELA NATUREZA – TRABALHO 3**

MAYARA EID ORLANDINI – CCO220388

São José do Rio Preto - SP

2023

## Sumário

1. Questão 1 .....	3
1.1    Enunciado .....	3
1.2 Linguagem e IDE.....	3
1.3    Descrição lógica do algoritmo .....	3
1.4 Análise do algoritmo .....	4
1.4.1 Influência da quantidade de partículas em um enxame no processo de otimização de uma função bidimensional utilizando o PSO .....	4
1.4.2 Influência da limitação dos parâmetros de velocidade das partículas sob o processo de otimização de uma função bidimensional utilizando o PSO .....	8
1.4.3 Influência do valor das constantes cognitivas e sociais no processo de otimização de uma função bidimensional utilizando o PSO .....	12
1.4.3 Análise final do PSO .....	29
1.4.4 Comparação da performance do PSO e do Algoritmo Genético no processo de otimização da função bidimensional .....	31
2. Questão 2 .....	33
2.1    Enunciado .....	33
2.2 Linguagem e IDE.....	33
2.3    Descrição lógica do algoritmo .....	33
2.4 Análise do algoritmo .....	35
2.4.1 Influência da quantidade de iterações no processo de otimização do problema TSP utilizando ACO.....	35
2.4.2 Influência das variáveis $\alpha$ e $\beta$ no processo de otimização do problema TSP utilizando ACO.....	40
2.4.3 Influência da variável $\rho$ no processo de otimização do problema TSP utilizando ACO .....	51
2.4.4 Influência da variável Q no processo de otimização do problema TSP utilizando ACO .....	54

2.4.5 Influência da variável b no processo de otimização do problema TSP utilizando ACO .....	59
2.4.6 Análise final do ACO .....	63
3. Link para acesso dos algoritmos.....	66
3.1 Link de acesso ao exercício 1 .....	66
3.2 Link de acesso ao exercício 2 .....	66
4. Referências bibliográficas .....	67

## 1. Questão 1

Nessa seção, serão apresentados os tópicos relacionados a resolução da questão 1 proposta no trabalho 3 da matéria “Computação Inspirada pela Natureza”, referente ao Programa de Pós-graduação em Ciência da Computação da UNESP.

### 1.1 Enunciado

Utilize o PSO para minimizar a seguinte função no intervalo (-5, 5) e (-5, 5):

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

Inclua em seu relatório um gráfico que mostre o valor mínimo e médio de  $f(x, y)$  ao longo das iterações, além das configurações utilizadas e outros resultados que julgar interessantes.

Compare os resultados obtidos com aqueles obtidos com algoritmos genéticos no mesmo problema.

### 1.2 Linguagem e IDE

O código referente a questão 1 foi desenvolvido em *Python* [1], com suporte do *Google Colaboratory* [2].

### 1.3 Descrição lógica do algoritmo

Para o funcionamento do algoritmo, implementou-se as bibliotecas *random* [3] e *matplotlib* [4]. A primeira citada permite, justamente, trabalhar com números aleatórios e, a segunda, possibilita a geração de gráficos.

Iniciando a construção do código, parte-se para a definição de parâmetros iniciais, que delimitam o tamanho populacional, os limites de velocidade de cada partícula que compõe tal população, bem como a constante cognitiva e social, utilizadas no processo de atualização da velocidade das partículas do enxame [5].

Com tais parâmetros definidos, gera-se a população de modo aleatório, fazendo com que esta seja composta por partículas formadas por coordenadas x e y, conhecidas como tuplas. Para cada uma dessas partículas, atribui-se uma velocidade decimal aleatória, segmentada, também, em componentes x e y, que se relacionam às coordenadas anteriores.

Com o enxame formado, calcula-se a aptidão de cada partícula. Nesse caso, como o objetivo é otimizar uma função, minimizando-a, basta aplicar as coordenadas x e y geradas de maneira randômica e aplicá-las na equação em estudo. É a partir desse processo avaliativo, que algumas considerações são realizadas: i) o mínimo global da função é encontrado, bem como a partícula associada à tal mínimo; ii) os mínimos locais e as partículas associadas a estes são localizados.

Na sequência, cabe-se a realização de um processo iterativo, onde a lógica instituída permite que, a cada iteração, as melhores posições locais e global sejam atualizadas. É justamente com base nesses valores que as velocidades, e posteriormente as posições populacionais, são recalculadas, seguindo o precedente de respeitarem os limites de busca determinados em etapa anterior [5].

À fim de avaliar as novas soluções, são recalculadas as aptidões de cada indivíduo e o processo supracitado é reiniciado até que as iterações estabelecidas sejam concluídas.

#### **1.4 Análise do algoritmo**

A fim de realizar uma análise relacionada ao algoritmo desenvolvido, o código foi executado algumas vezes e, durante esse processo, alguns parâmetros foram alterados e resultados foram coletados e interpretados.

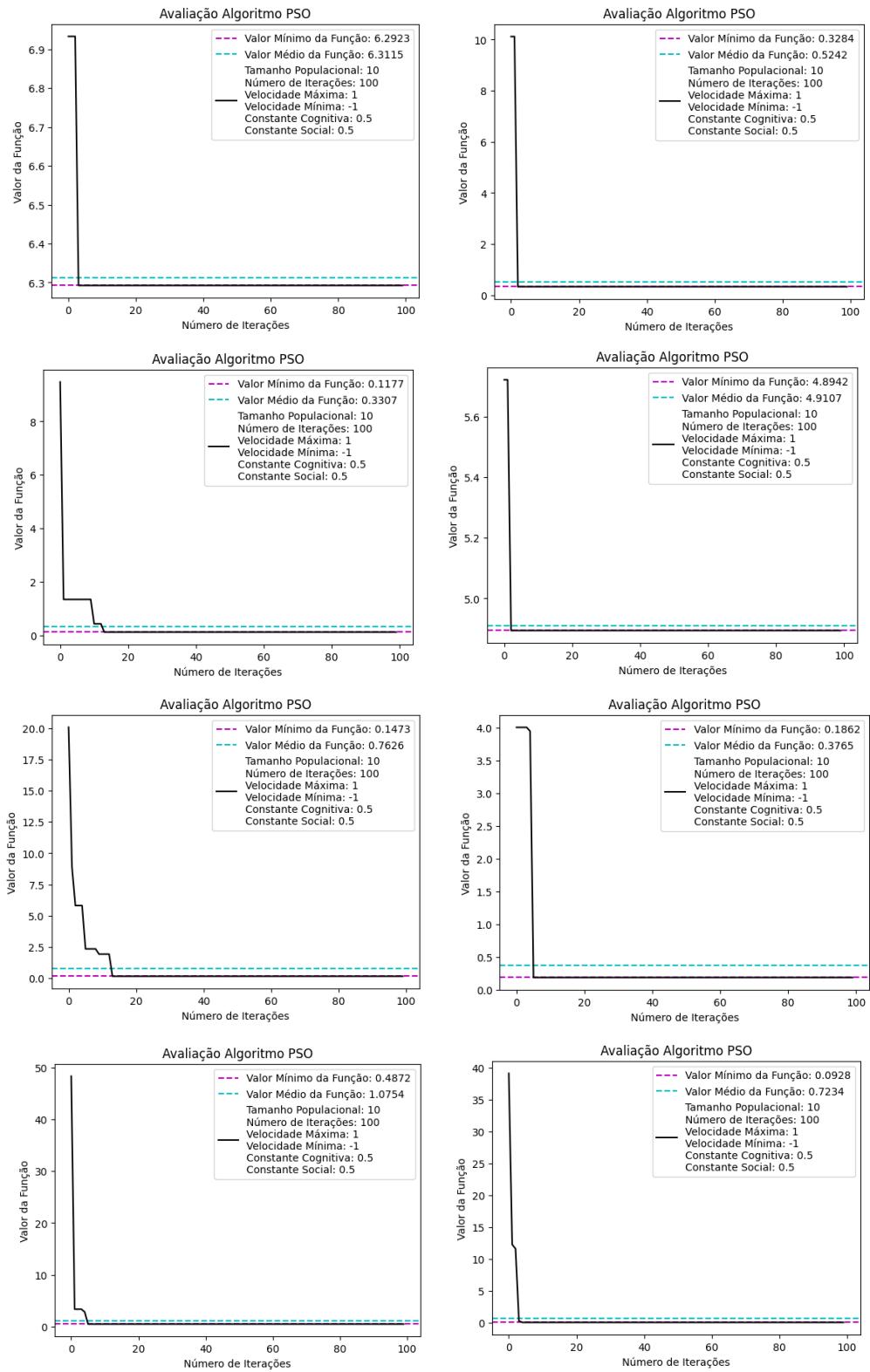
##### **1.4.1 Influência da quantidade de partículas em um enxame no processo de otimização de uma função bidimensional utilizando o PSO**

A primeira análise consistiu em analisar a influência da quantidade de partículas existentes no PSO em relação ao processo de convergência e atingimento de um ótimo resultado. Para isso, os outros parâmetros existentes – número de iterações, velocidades e constantes – foram mantidas e, somente o tamanho populacional sofreu alterações.

À título de complemento, considerou-se 100 iterações, as velocidades foram limitadas entre 1 e -1, e as constantes foram mantidas em 0,5. Todos esses valores podem ser observados nos gráficos plotados.

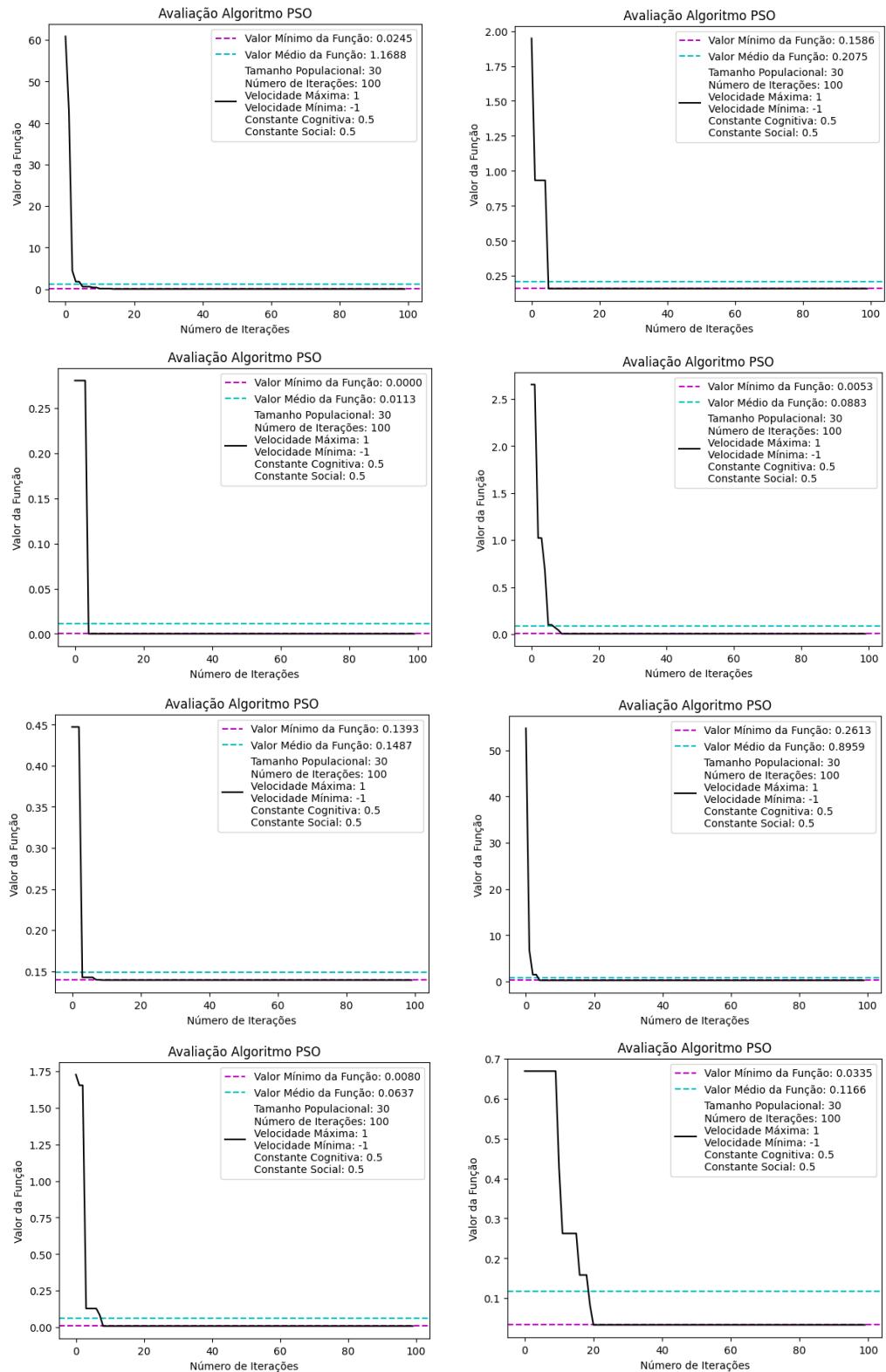
Considerou-se então, a análise de três tamanhos populacionais: 10, 30 e 50 partículas. Para a análise, o algoritmo foi iterado 8 vezes para cada configuração.

*Figura 1: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando 10 partículas em um enxame.*



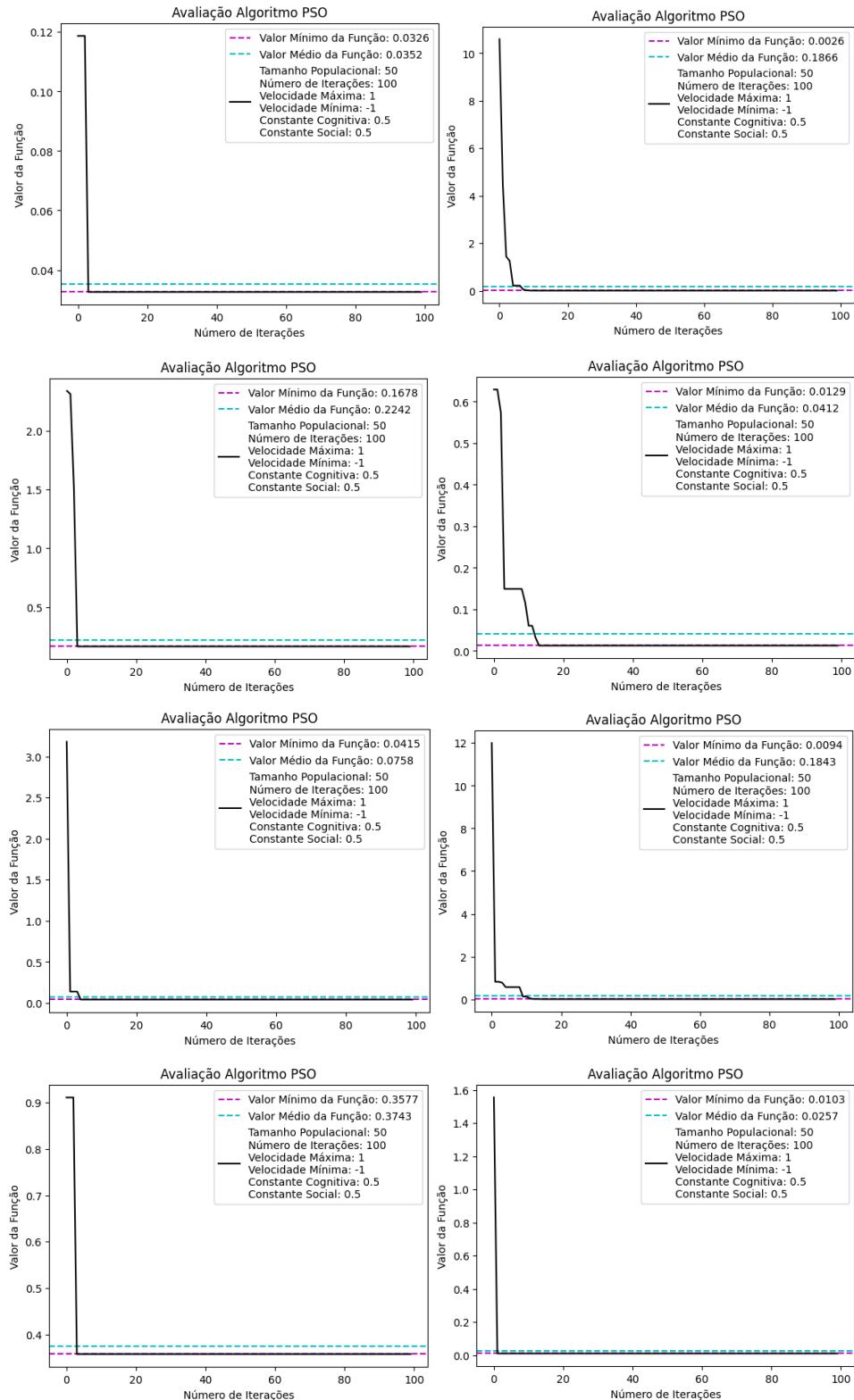
*Fonte: Elaborado pelo autor (2023).*

*Figura 2: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando 30 partículas em um enxame.*



*Fonte: Elaborado pelo autor (2023).*

Figura 3: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando 50 partículas em um enxame.



Fonte: Elaborado pelo autor (2023).

Analisando os gráficos apresentados nas Figuras 1, 2 e 3, é possível extrair que com um aumento populacional existe a tendência de que melhores resultados sejam atingidos. Isso é percebido à medida que o valor mínimo e médio da função otimizada em questão tende a reduzir-se com o aumento da quantidade de partículas (Tabela 1).

*Tabela 1: Resumo da influência a quantidade de partículas em um enxame no processo de otimização de uma função bidimensional utilizando o PSO.*

Avaliação da Influência do Tamanho Populacional no PSO			
Tamanho Populacional	10	30	50
Média do Valor Mínimo da Função (8 execuções)	1.5683	0.0788	0.0794
Média do Valor Médio da Função (8 execuções)	1.8769	0.3376	0.1434

*Fonte: Elaborado pelo autor (2023).*

O fato é justificado uma vez que, com uma maior quantidade de partículas no enxame, a diversidade populacional sofre um crescimento, permitindo uma exploração mais segura do espaço de busca pré-definido. Em contrapartida, mais partículas demandam de mais recursos, culminando em, possivelmente, um processo de convergência mais demorado.

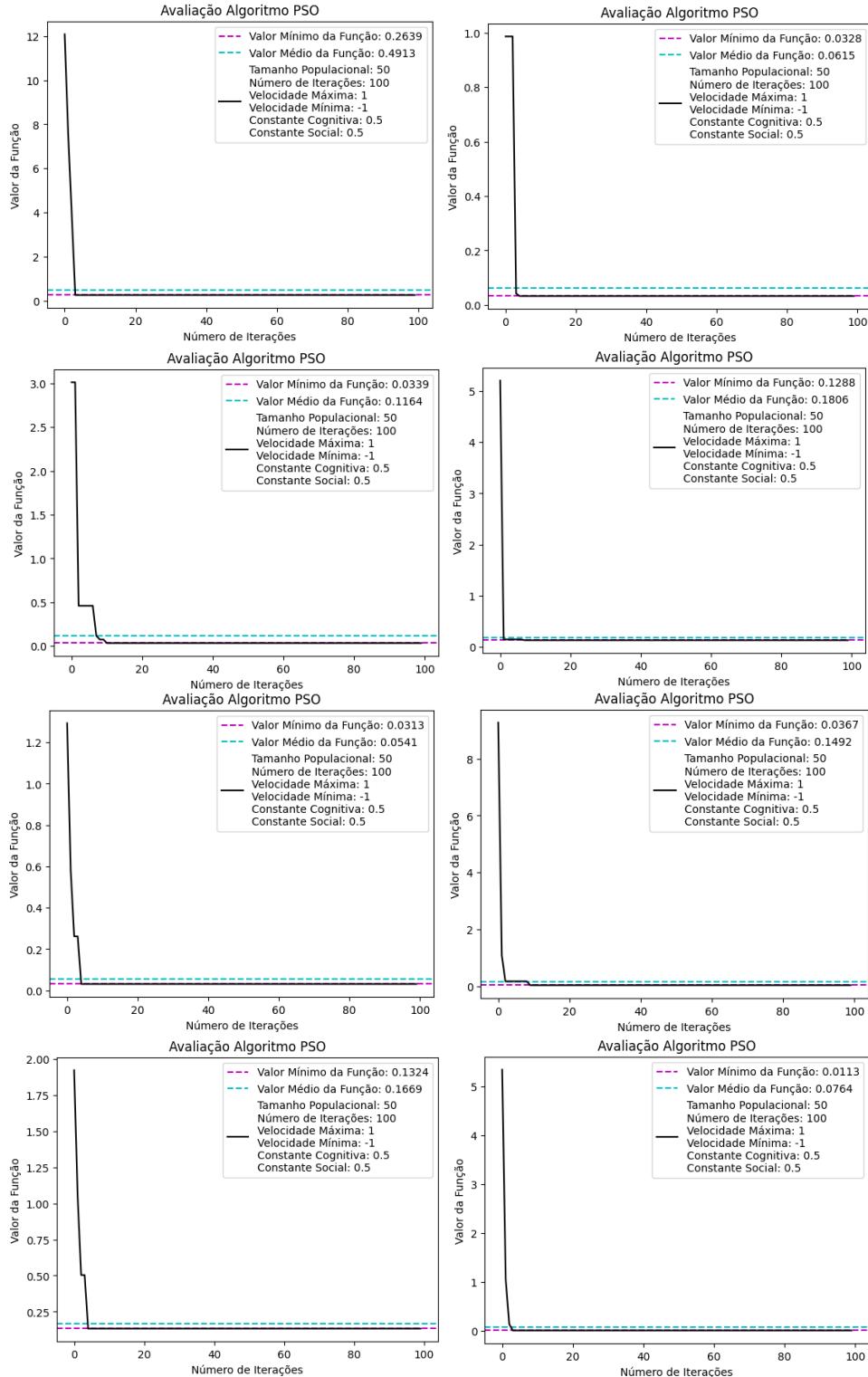
Nesse caso, optou-se por seguir com a população com 50 indivíduos, haja vista o atingimento de bons resultados sem grandes possibilidades de superexplorações ou gastos irreais de recursos computacionais.

#### **1.4.2 Influência da limitação dos parâmetros de velocidade das partículas sob o processo de otimização de uma função bidimensional utilizando o PSO**

O segundo experimento consistiu em analisar a influência do range de velocidade das partículas. Para isso, os parâmetros considerados na seção anterior foram mantidos, variando somente o que se entende por limite de velocidade máxima e mínima de cada indivíduo do enxame.

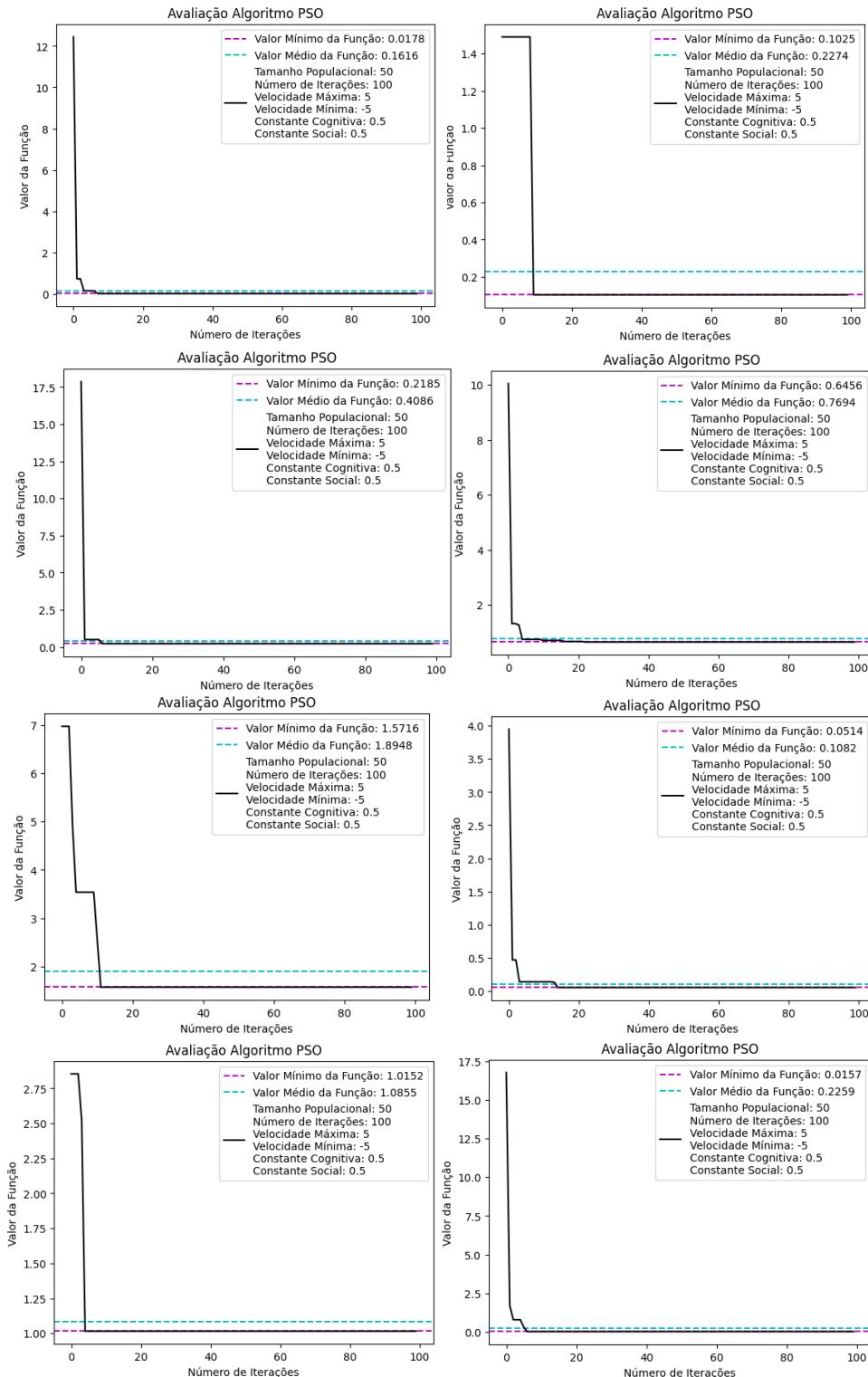
Adotou-se, então, limites simétricos de velocidades: -1 e 1, -5 e 5, -10 e 10. Para cada um dos casos executou-se o código 8 vezes.

Figura 4: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando os limites de velocidade de partículas como -1 e 1.



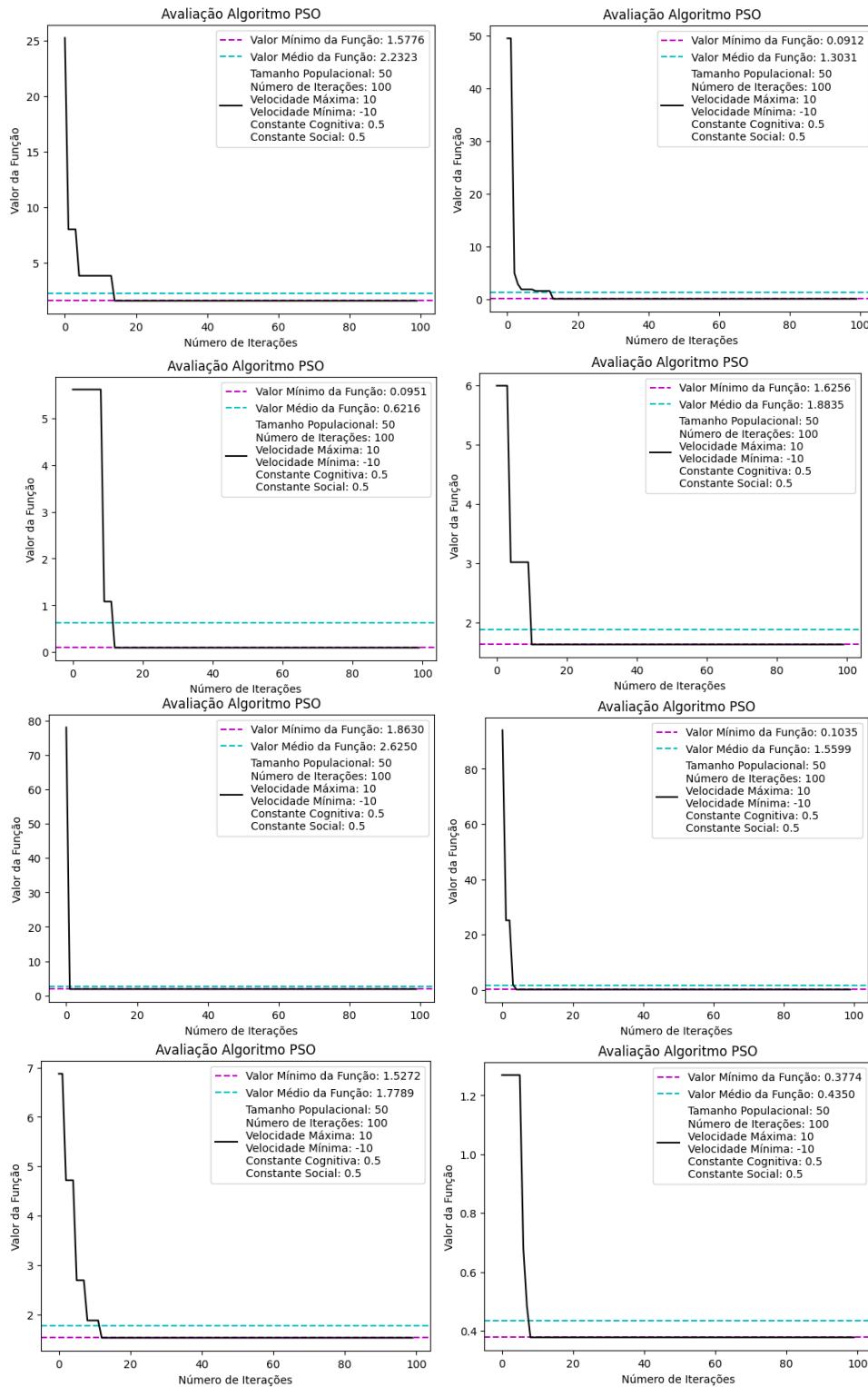
Fonte: Elaborado pelo autor (2023).

*Figura 5: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando os limites de velocidade de partículas como -5 e 5.*



*Fonte: Elaborado pelo autor (2023).*

Figura 6: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando os limites de velocidade de partículas como -10 e 10.



Fonte: Elaborado pelo autor (2023).

Analizando as informações gráficas contidas nessa seção, é possível perceber que os melhores resultados são obtidos quando a velocidade das partículas é limitada entre -1 e 1, vide Tabela 2 abaixo.

*Tabela 2: Resumo da influência da limitação da velocidade das partículas no processo de otimização de uma função bidimensional utilizando o PSO.*

<b>Avaliação da Influência do Limite de Velocidade no PSO</b>			
<b>Tamanho Populacional</b>	10	30	50
<b>Média do Valor Mínimo da Função (8 execuções)</b>	0.0839	0.4548	0.9076
<b>Média do Valor Médio da Função (8 execuções)</b>	0.1621	0.6102	1.5549

*Fonte: Elaborado pelo autor (2023).*

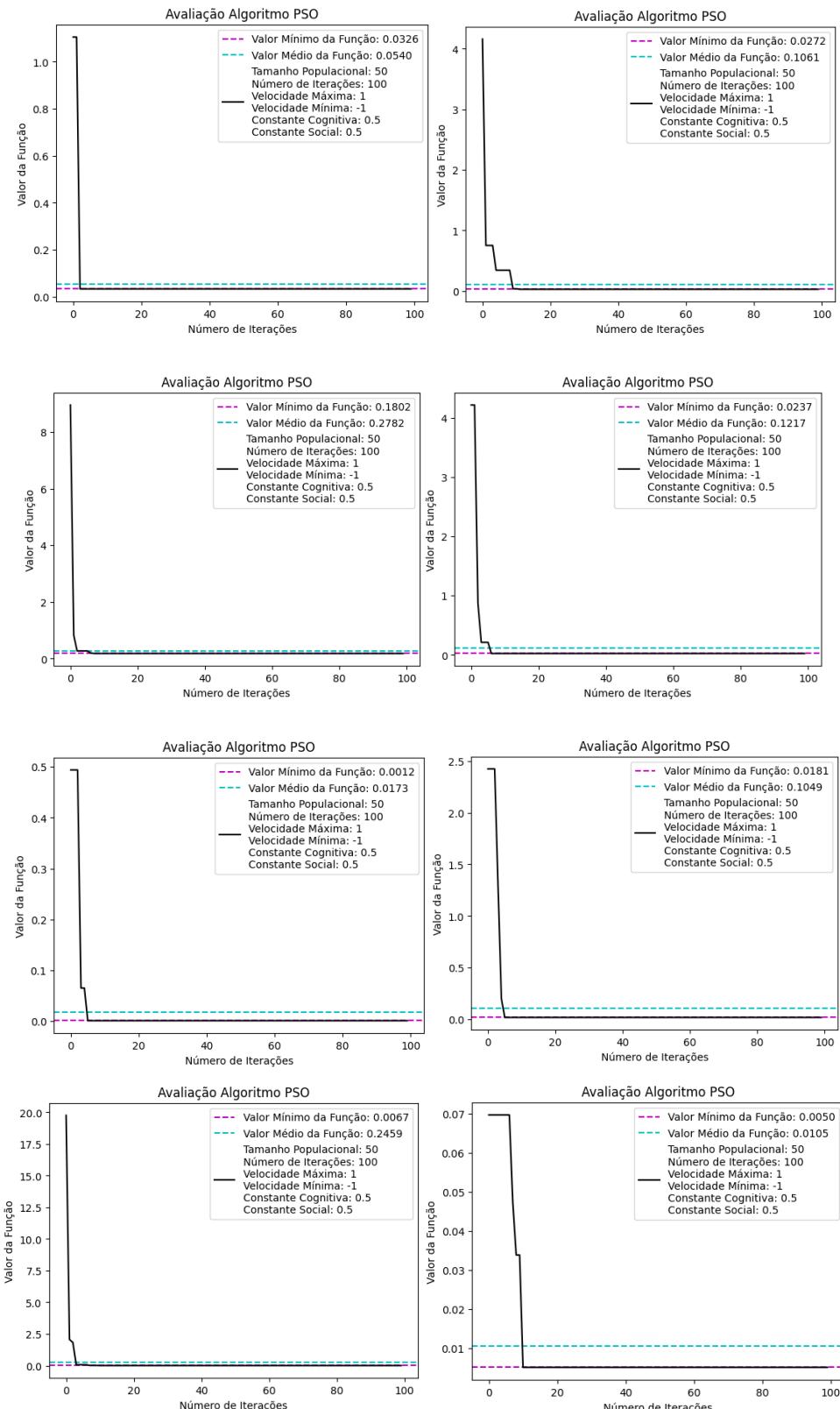
Interpreta-se que tal velocidade seja suficientemente ideal, uma vez que balanceia o processo de exploração e exploração. Adicionalmente, permite que as partículas mantenham certo grau de estabilidade que auxilia na busca por regiões próximas à ótimas soluções locais ou globais. Tal balanceamento otimiza a função adequadamente e, portanto, gera um comportamento não caótico.

Em outra perspectiva, velocidades mais lentas resultam em convergências mais lentas, porém, nesse caso, o grau de complexidade da situação proposta permite que baixos ranges de velocidades, como -1 e 1, sejam adotados.

#### **1.4.3 Influência do valor das constantes cognitivas e sociais no processo de otimização de uma função bidimensional utilizando o PSO**

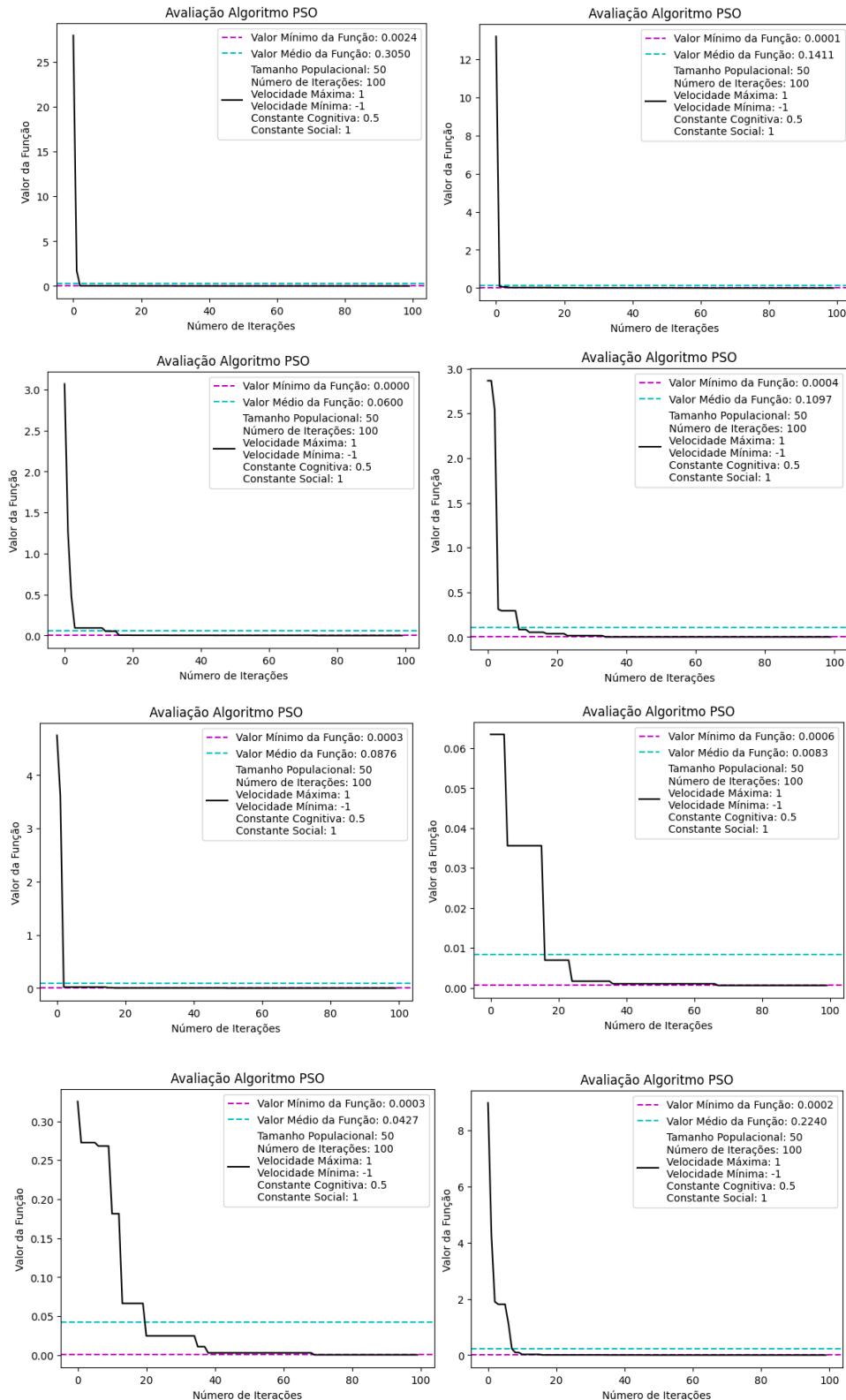
A terceira análise buscou entender a influência do valor das constantes de comportamento cognitivo bem como social das partículas que compõe o enxame. Assim como já realizado anteriormente, os parâmetros definidos em seções anteriores foram mantidos. As constantes, por sua vez, variaram, adotando valores de 0,5, 1, 1,5 e 2.

Figura 7: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 0.5 e constante social como 0.5.



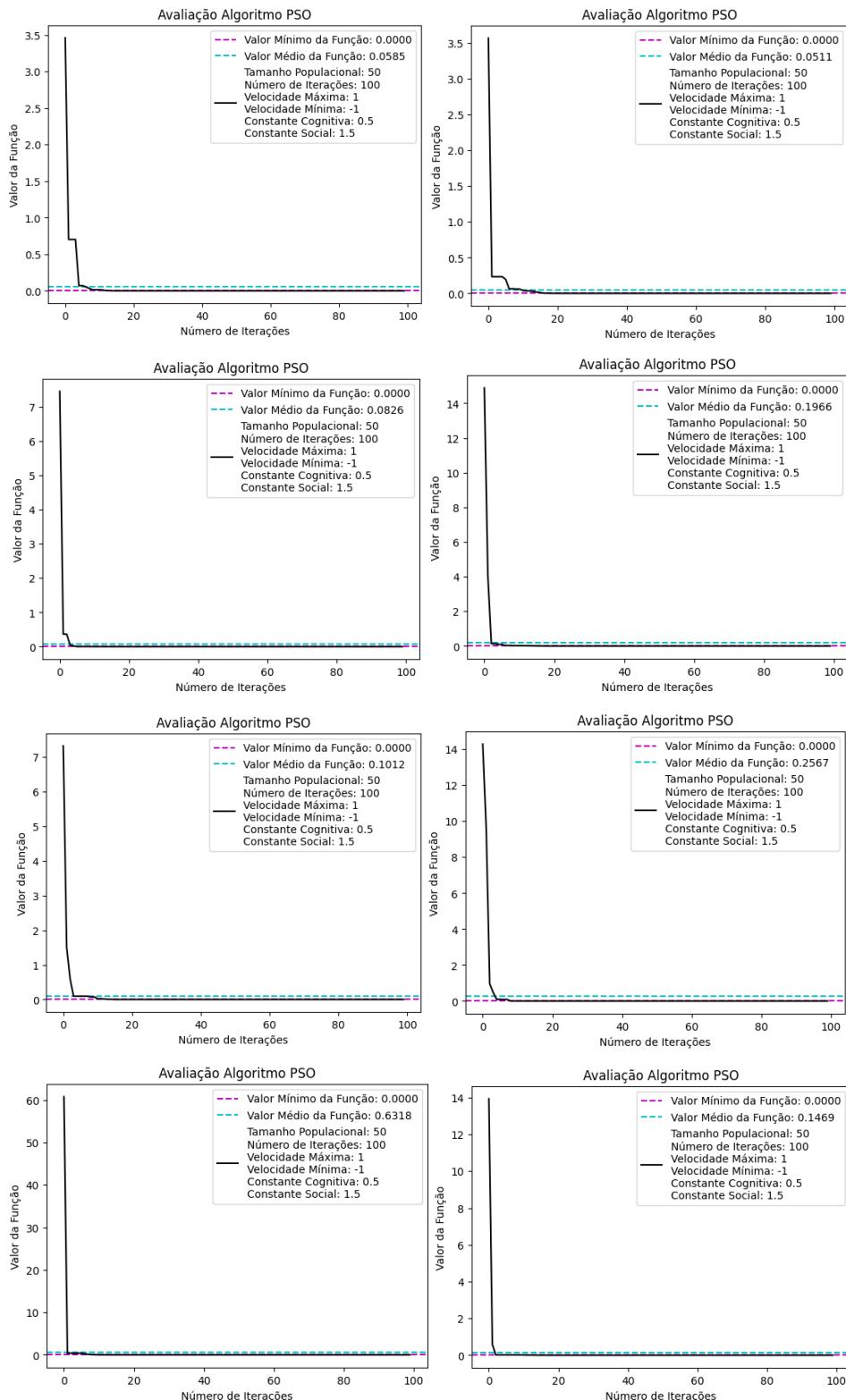
Fonte: Elaborado pelo autor (2023).

Figura 8: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 0.5 e constante social como 1.0.



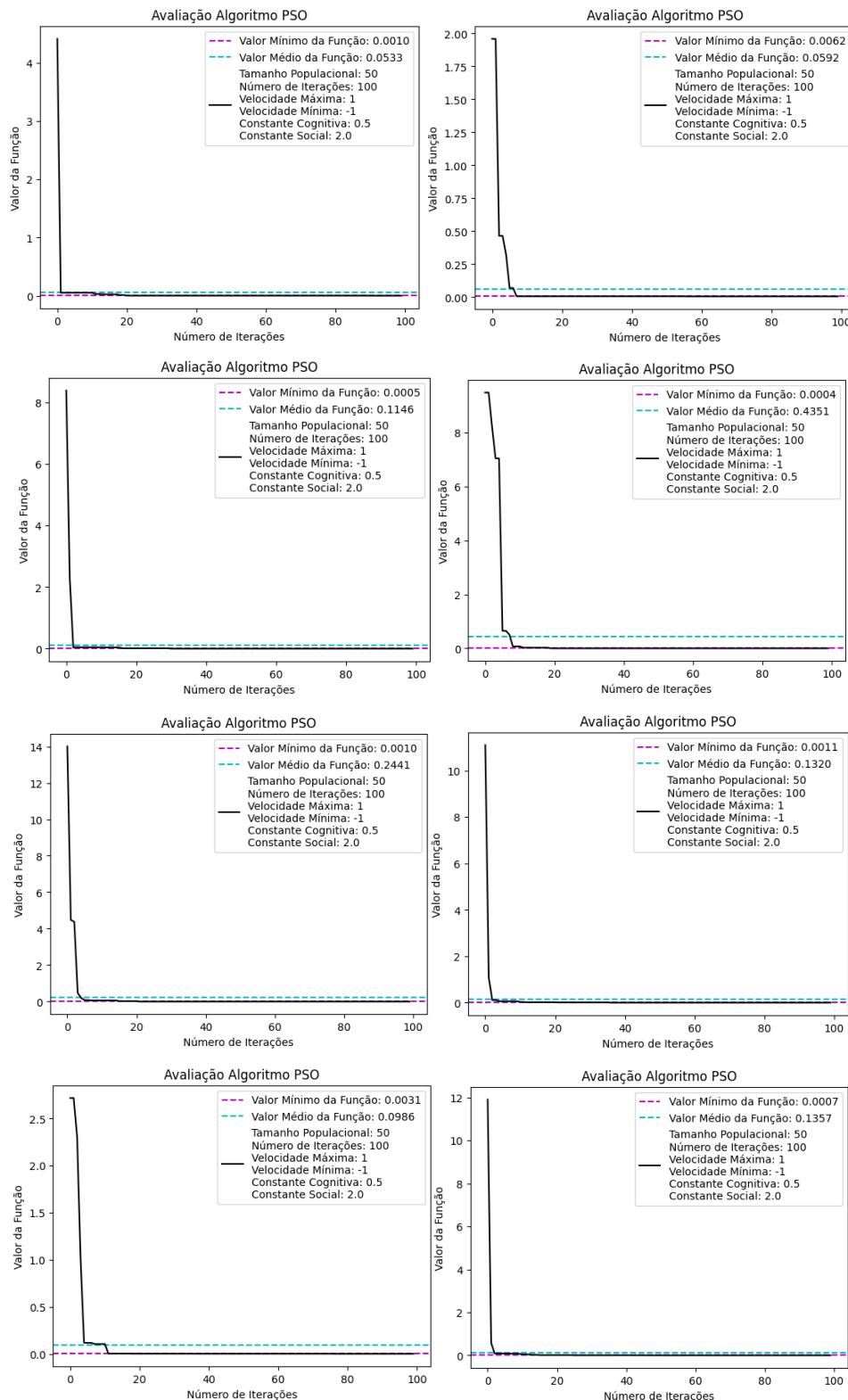
Fonte: Elaborado pelo autor (2023).

Figura 9: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 0.5 e constante social como 1.5.



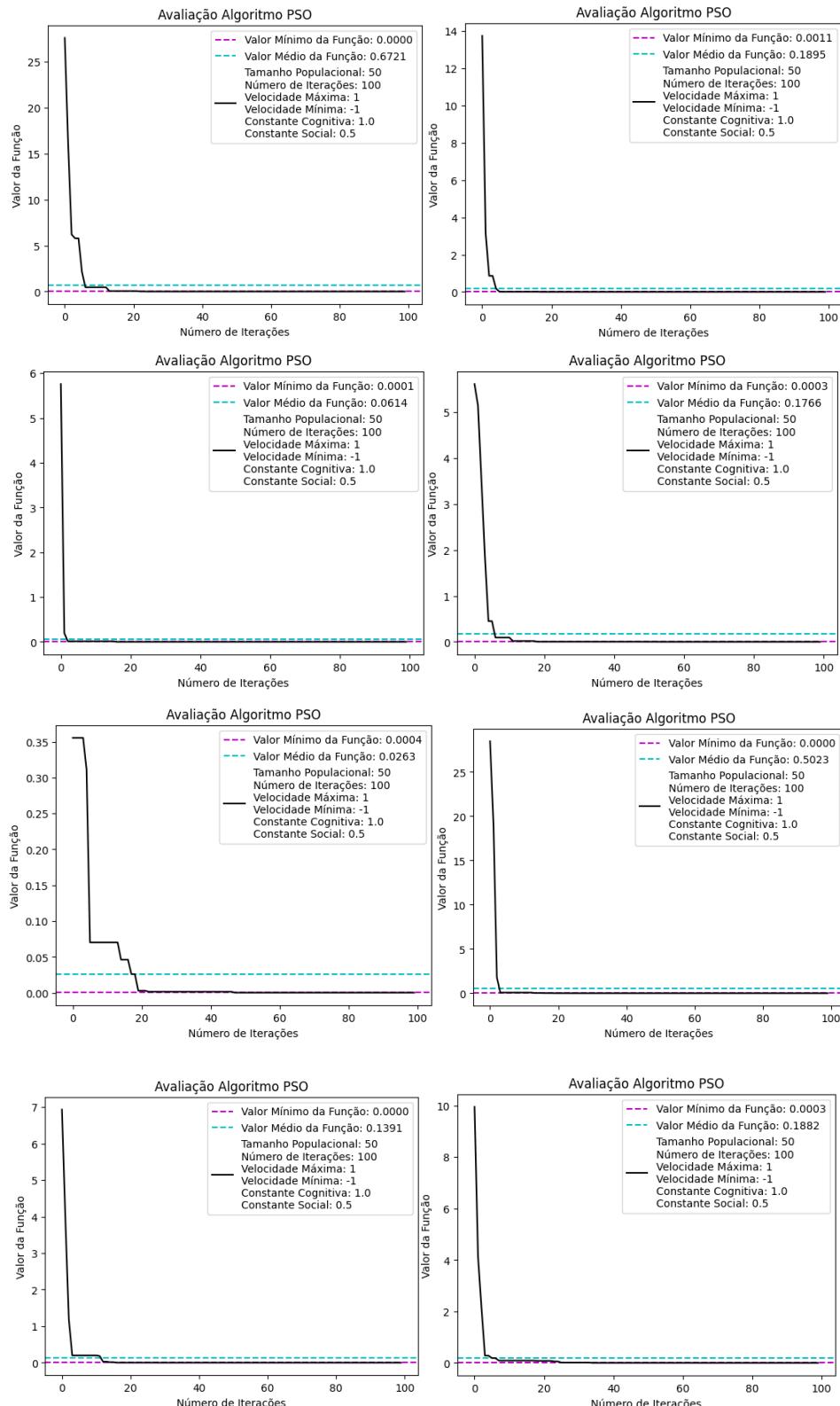
Fonte: Elaborado pelo autor (2023).

*Figura 10: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 0.5 e constante social como 2.0.*



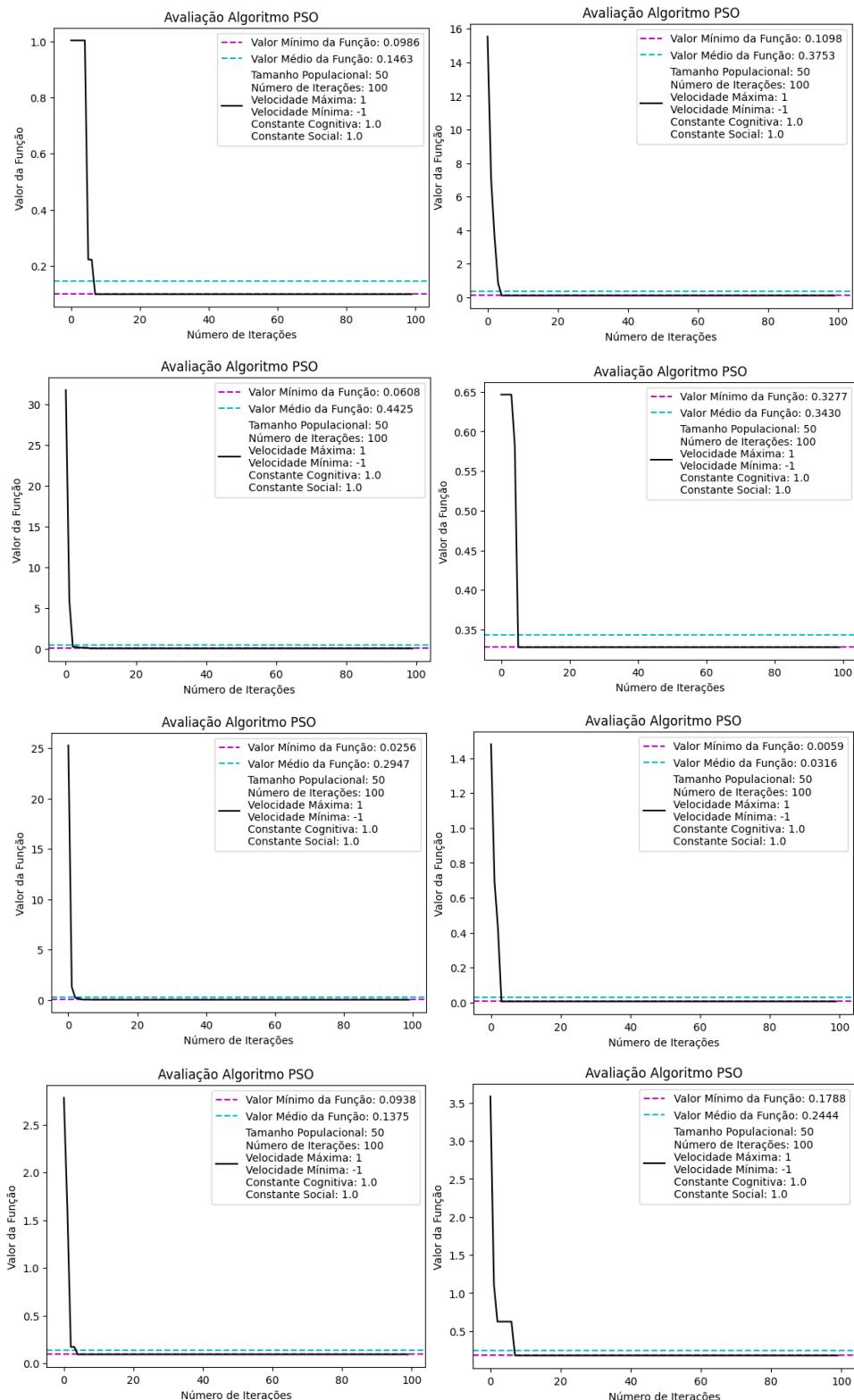
*Fonte: Elaborado pelo autor (2023).*

*Figura 11: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 1.0 e constante social como 0.5.*



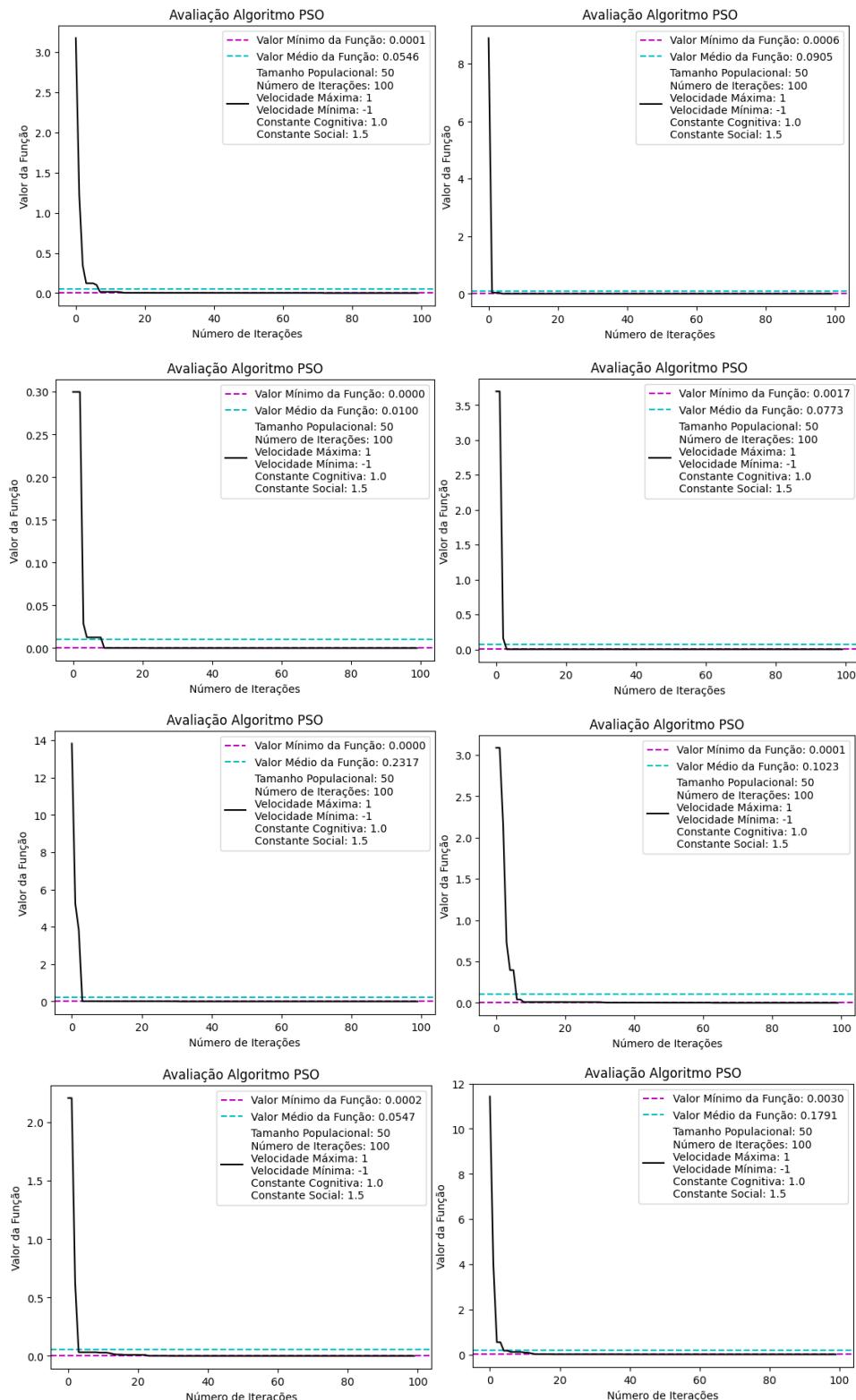
*Fonte: Elaborado pelo autor (2023).*

*Figura 12: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 1.0 e constante social como 1.0.*



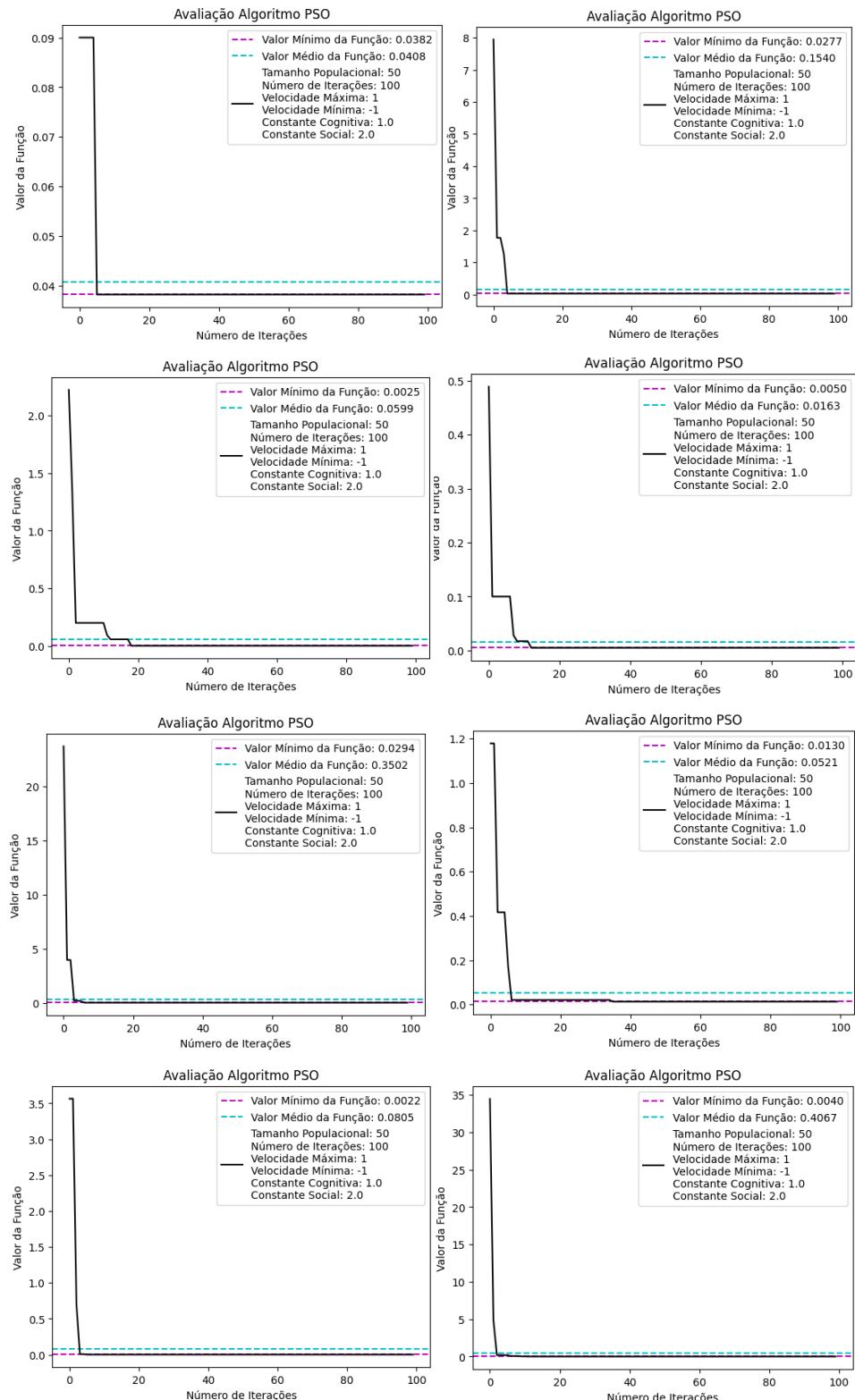
*Fonte: Elaborado pelo autor (2023).*

*Figura 13: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 1.0 e constante social como 1.5.*



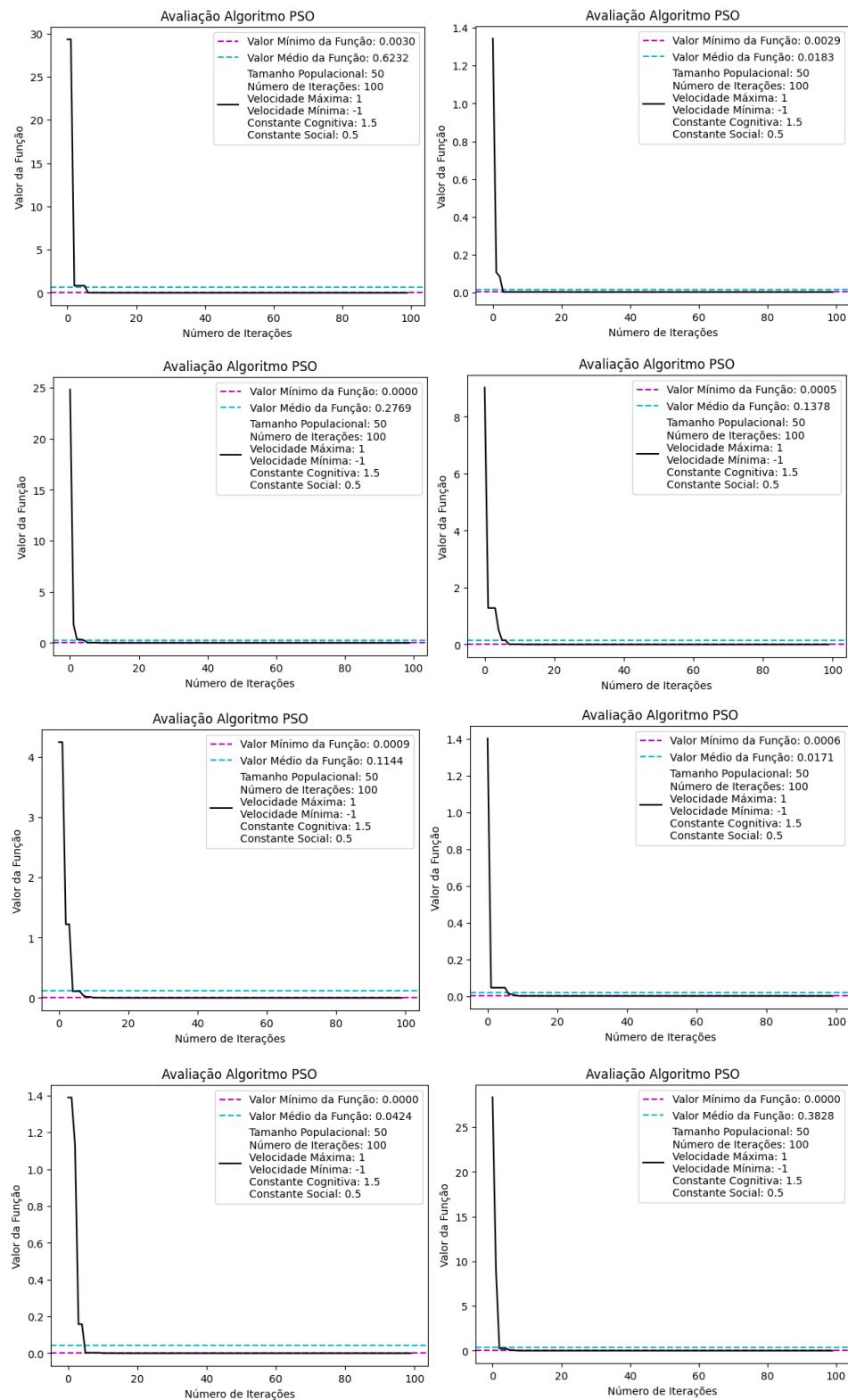
*Fonte: Elaborado pelo autor (2023).*

*Figura 14: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 1.0 e constante social como 2.0.*



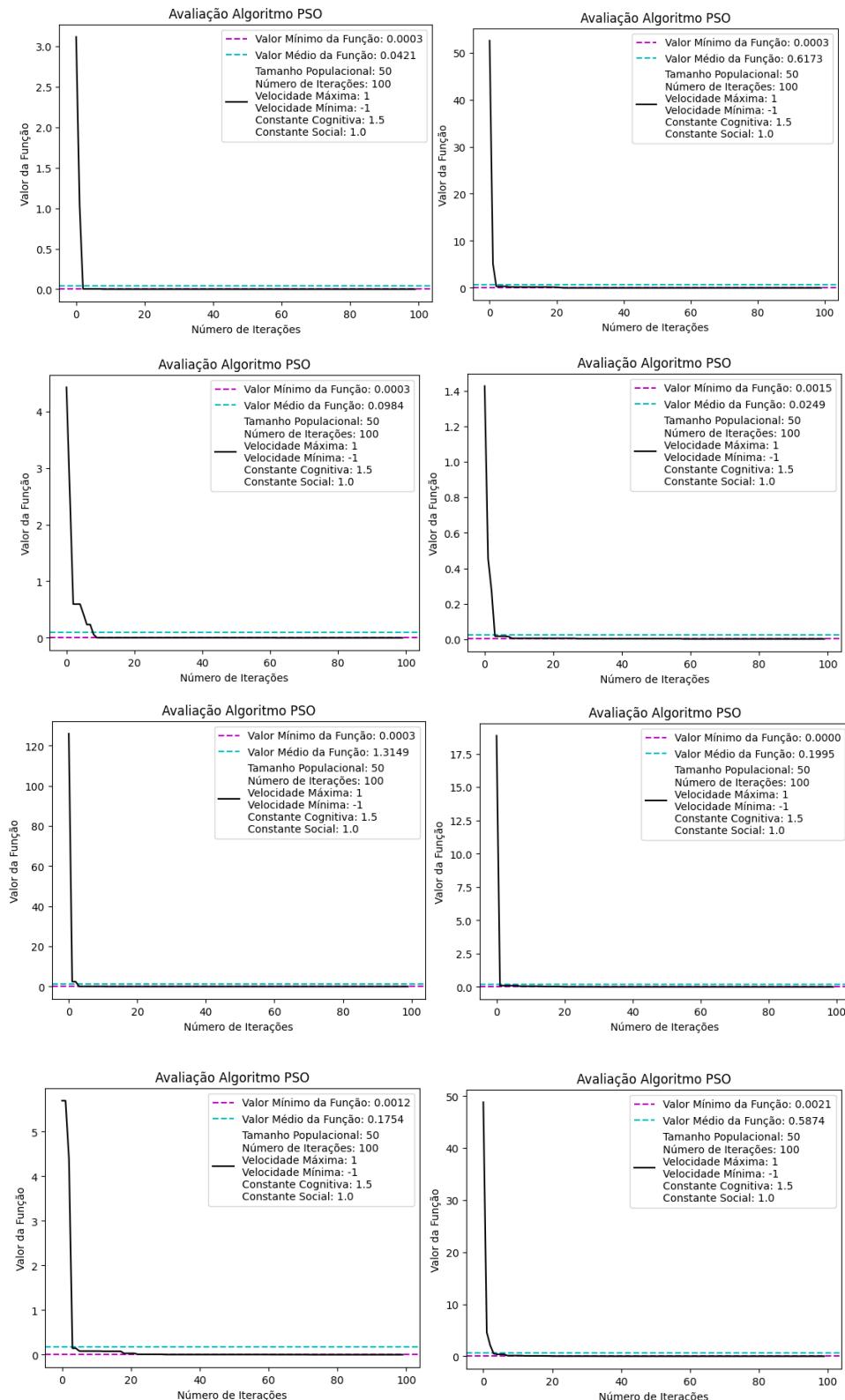
*Fonte: Elaborado pelo autor (2023).*

*Figura 15: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 1.5 e constante social como 0.5.*



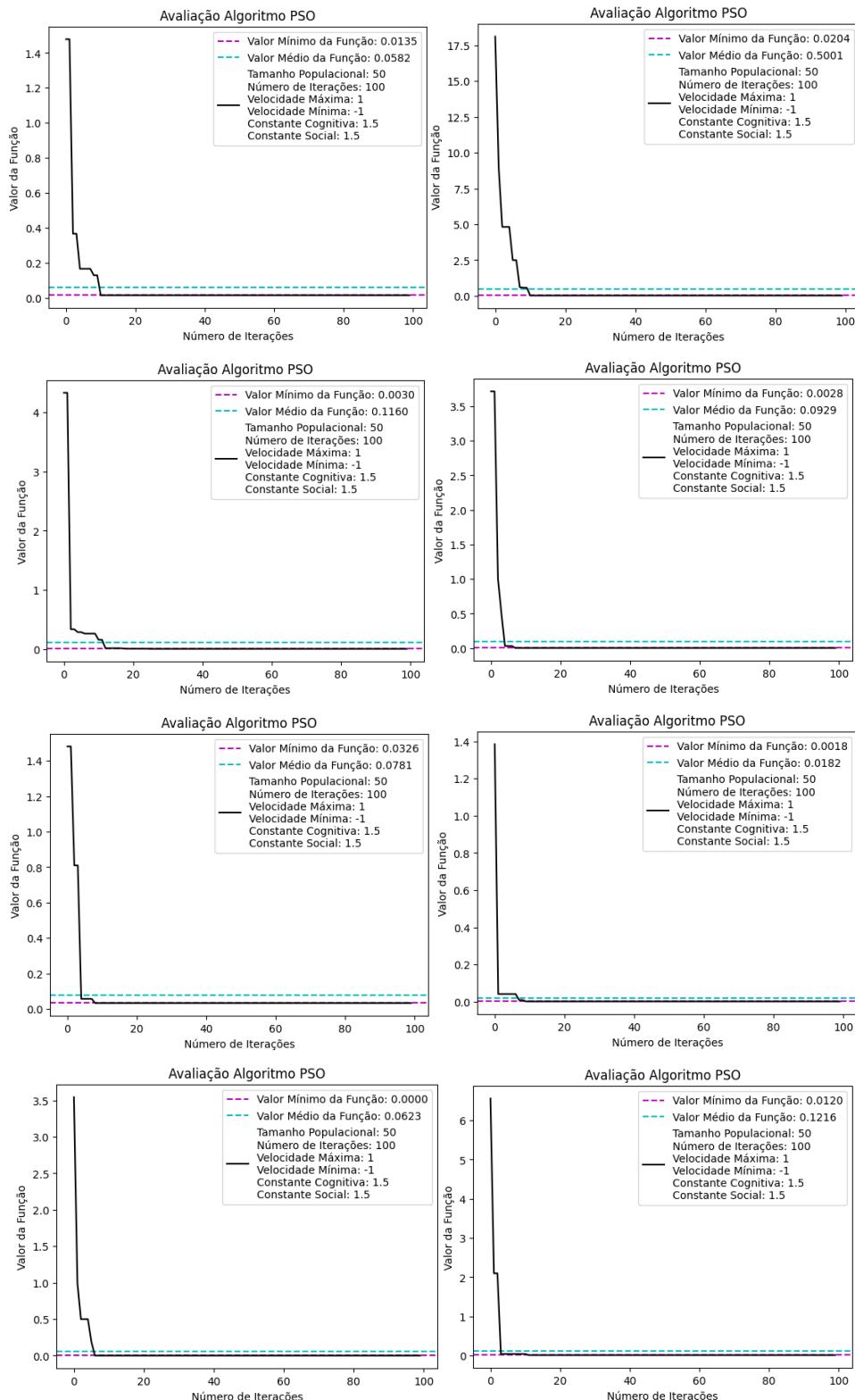
*Fonte: Elaborado pelo autor (2023).*

*Figura 16: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 1.5 e constante social como 1.0.*



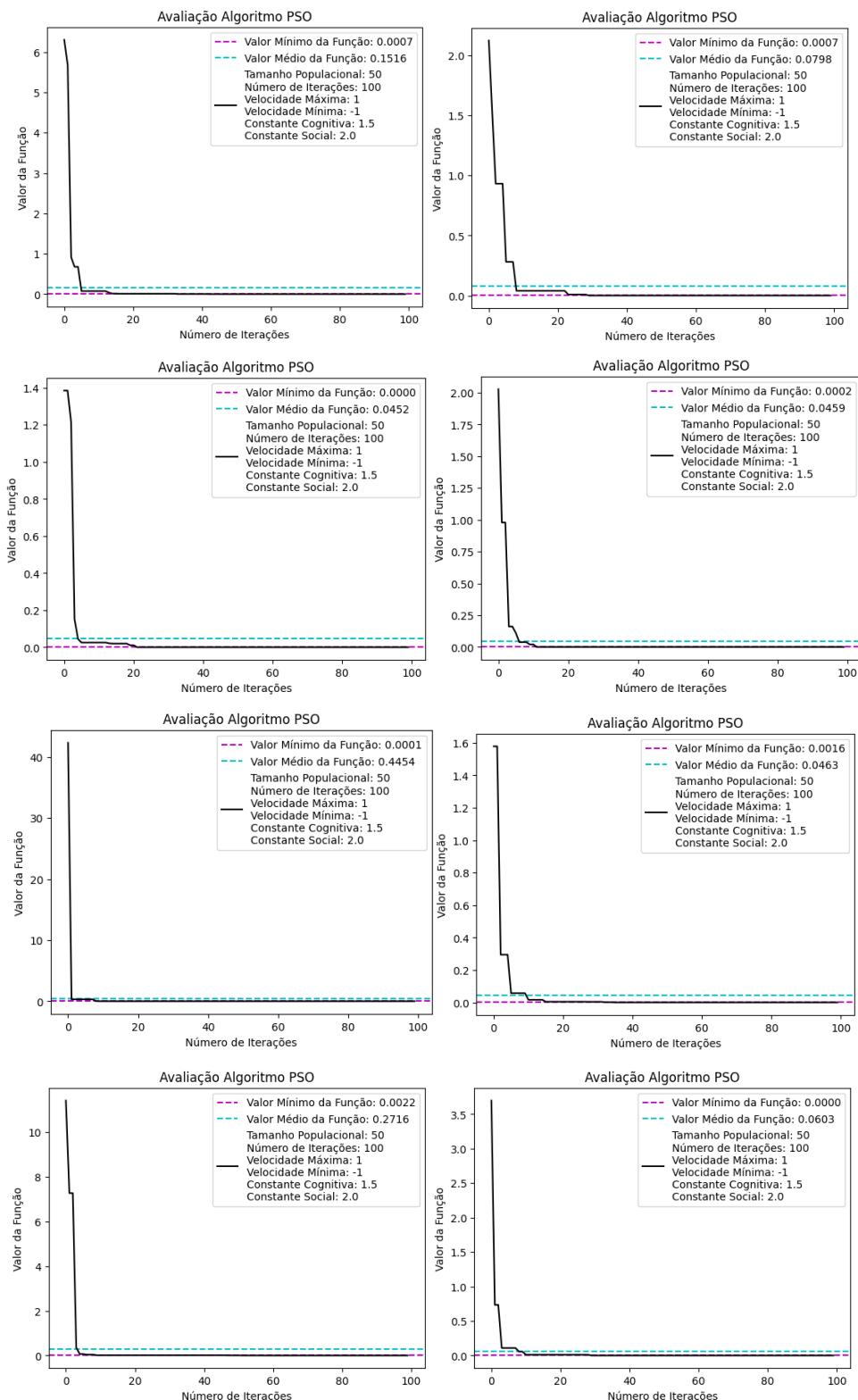
*Fonte: Elaborado pelo autor (2023).*

*Figura 17: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 1.5 e constante social como 1.5.*



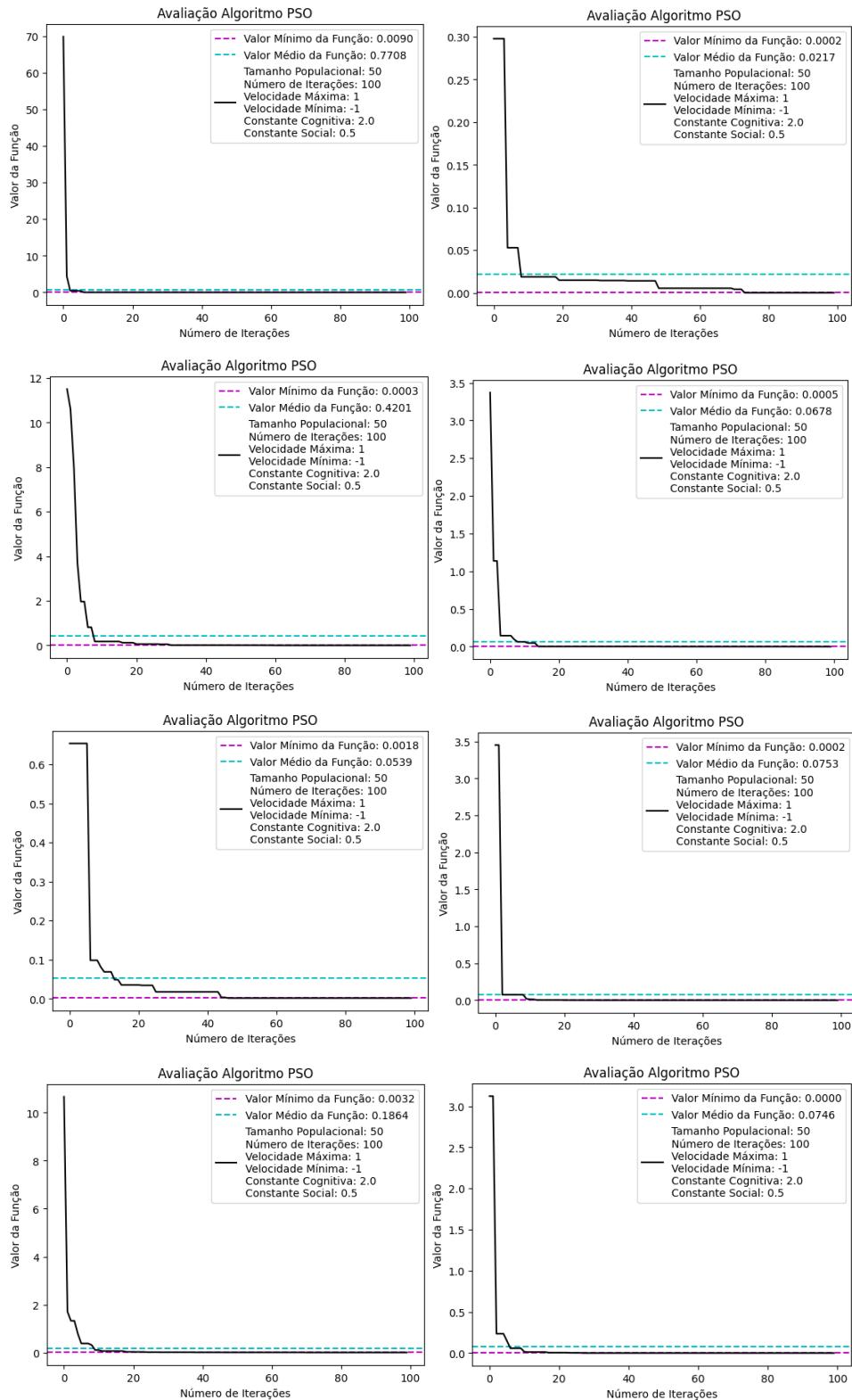
*Fonte: Elaborado pelo autor (2023).*

*Figura 18: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 1.5 e constante social como 2.0.*



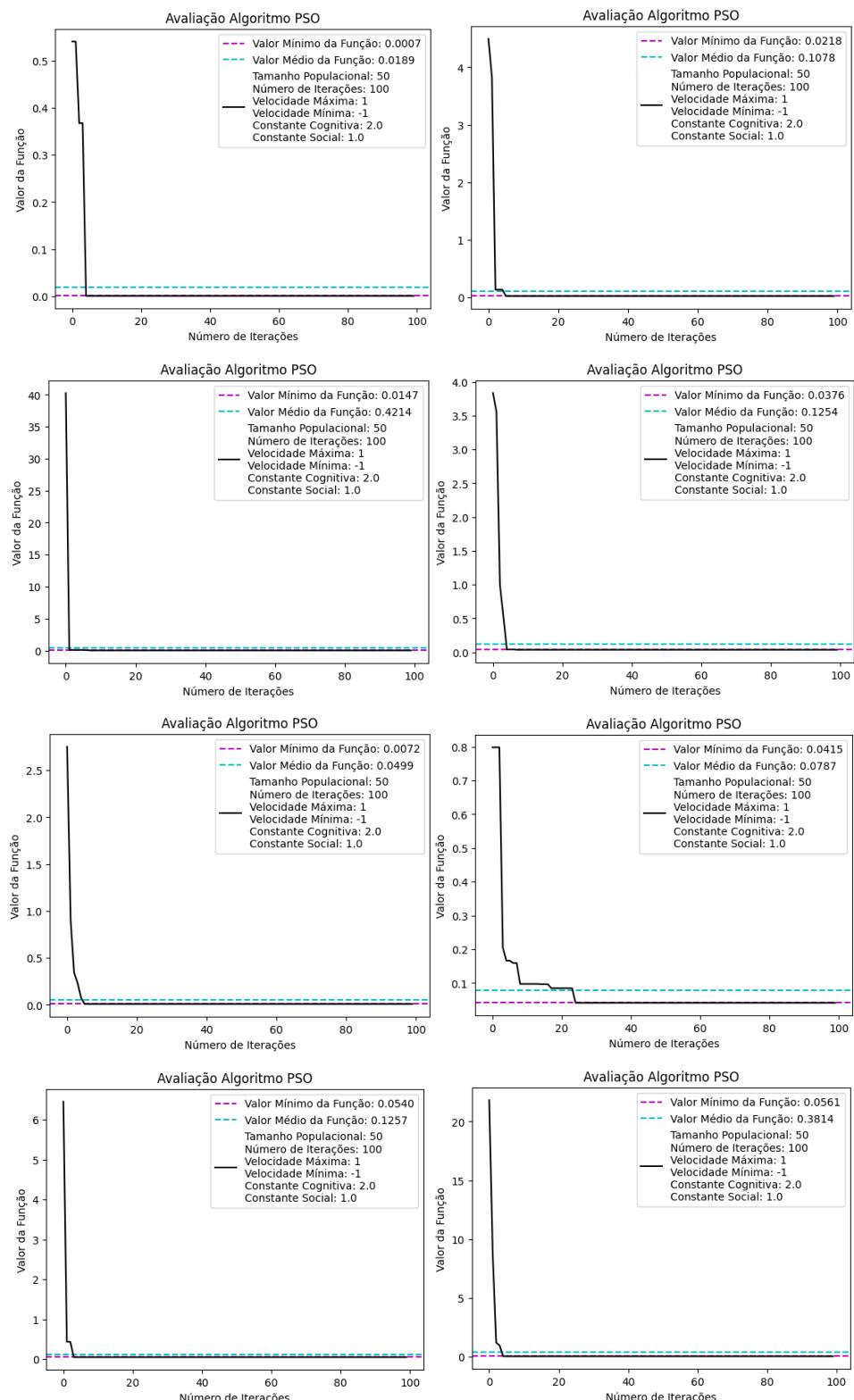
*Fonte: Elaborado pelo autor (2023).*

*Figura 19: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 2.0 e constante social como 0.5.*



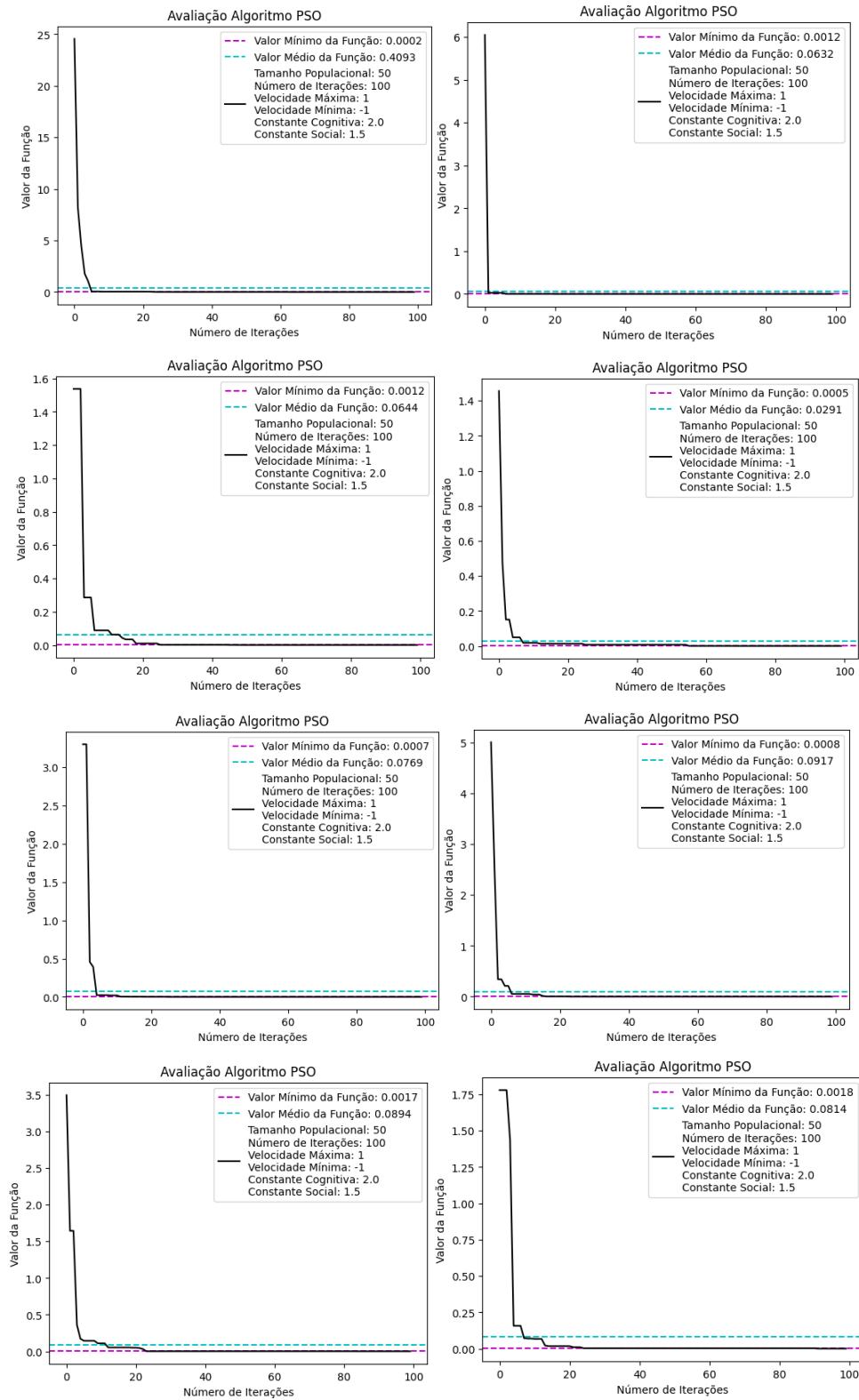
*Fonte: Elaborado pelo autor (2023).*

*Figura 20: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 2.0 e constante social como 1.0.*



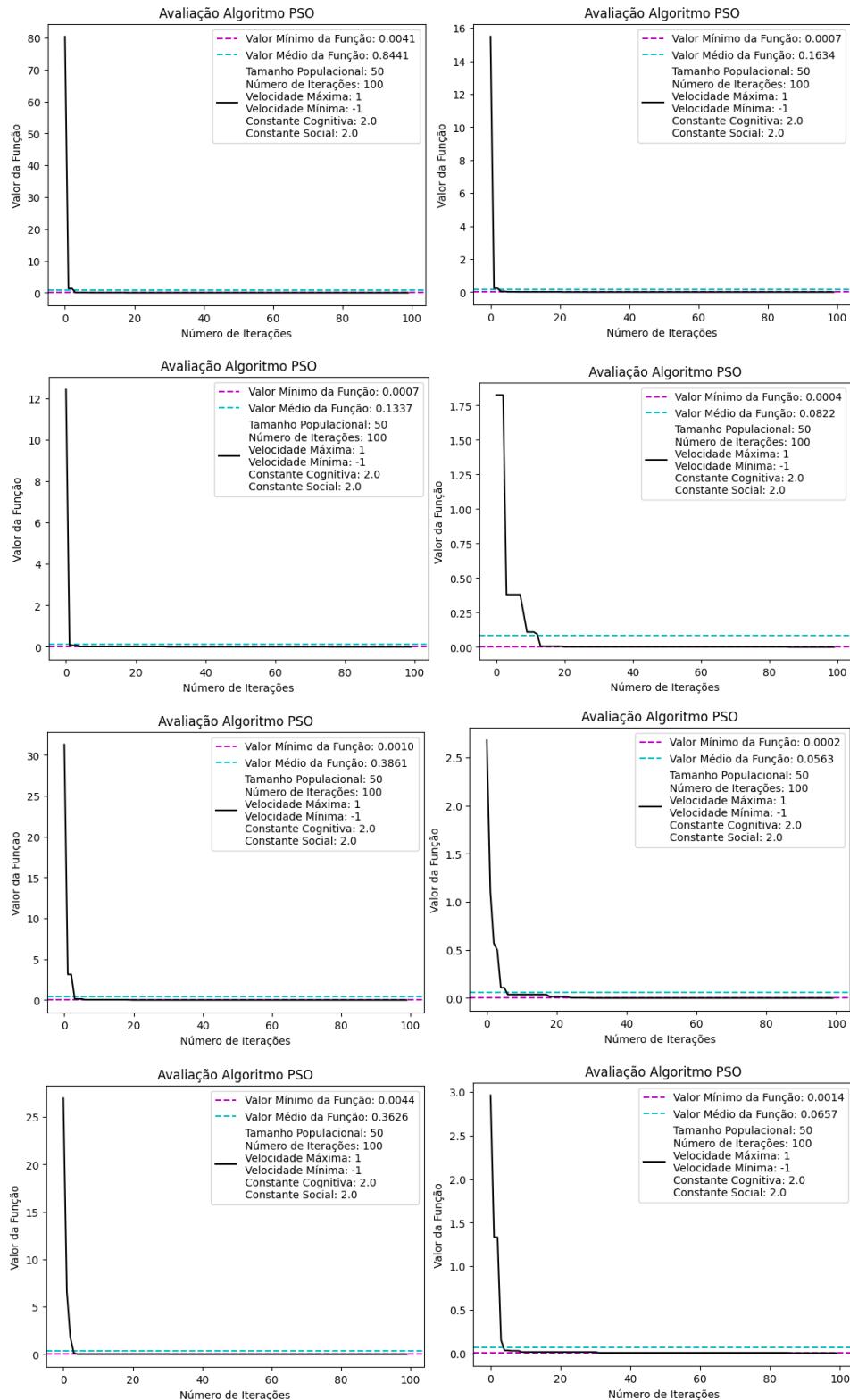
*Fonte: Elaborado pelo autor (2023).*

*Figura 21: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 2.0 e constante social como 1.5.*



*Fonte: Elaborado pelo autor (2023).*

*Figura 22: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional considerando a constante cognitiva como 2.0 e constante social como 2.0.*



*Fonte: Elaborado pelo autor (2023).*

Observando os gráficos plotados acima, é possível extrair as informações contidas na Tabela 3.

*Tabela 3: Resumo da influência das constantes sociais e cognitivas no processo de otimização de uma função bidimensional utilizando o PSO.*

		Média do Valor Mínimo da Função						Média do Valor Médio da Função			
		Cognitiva						Cognitiva			
		0.5	1.0	1.5	2.0			0.5	1.0	1.5	2.0
Social	0.5	0.0368	0.0003	0.0010	0.0019	Social	0.50	0.1173	0.2444	0.2016	0.2088
	1.0	0.0005	0.1126	0.0008	0.0292		1.00	0.1223	0.2519	0.3825	0.1637
	1.5	0.0000	0.0007	0.0108	0.0010		1.50	0.1907	0.1000	0.1309	0.1132
	2.0	0.0018	0.0153	0.0007	0.0016		2.00	0.1591	0.1451	0.1433	0.2493

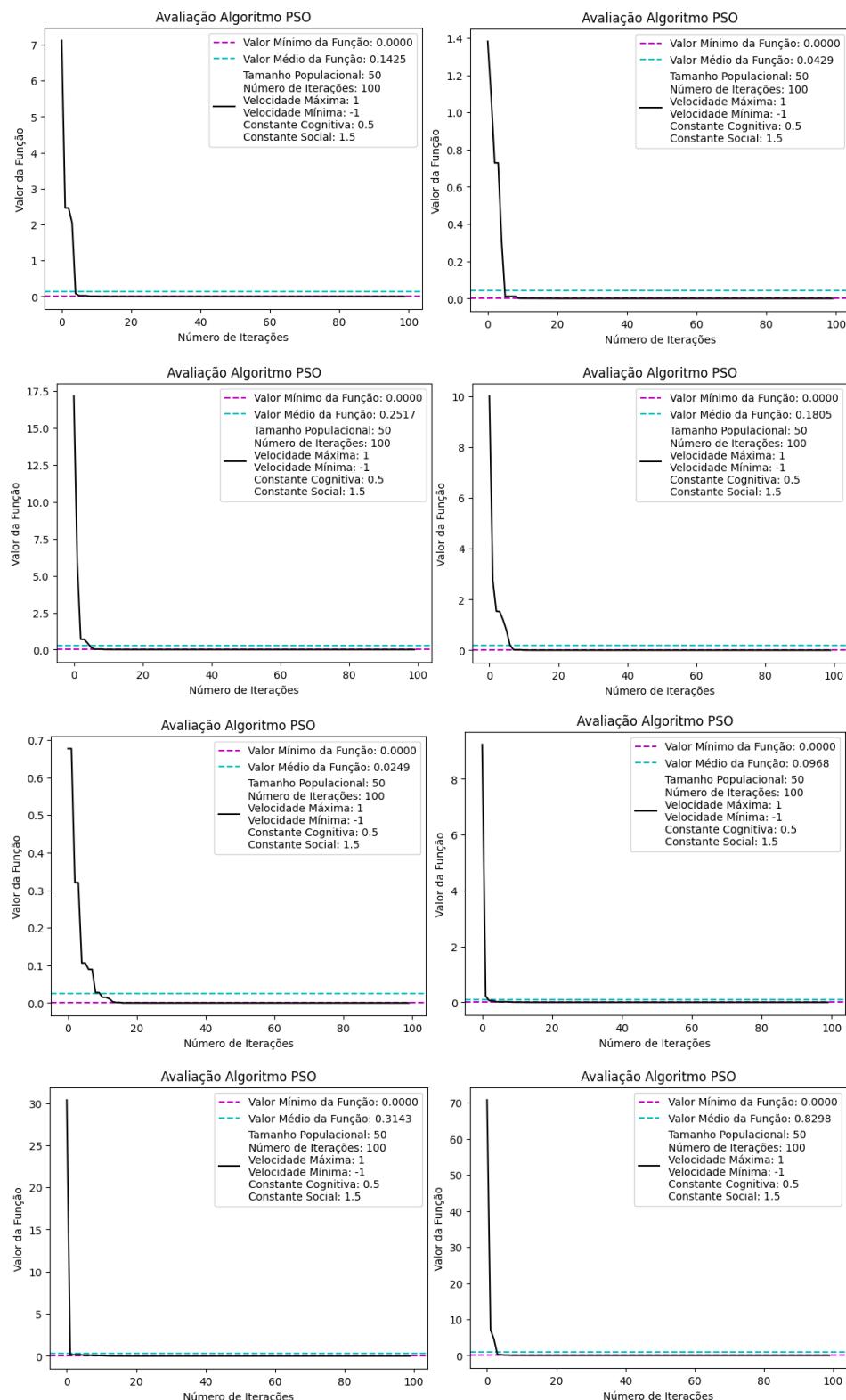
*Fonte: Elaborado pelo autor (2023).*

Por meio da Tabela 3, entende-se que o menor valor da função é atingido quando se utiliza uma constante cognitiva de 0,5 e uma constante social de 1,5. O valor médio da função, por outro lado, embora não seja o menor obtido, é satisfatório. Portanto, para essa situação, optou-se por seguir com as constantes cognitivas e sociais de 0,5 e 1,5, respectivamente.

#### 1.4.3 Análise final do PSO

À título de analisar o algoritmo com todos os parâmetros devidamente ajustados, com base nas experimentações supracitadas, executou-se o código 8 vezes, considerando 50 partículas, 100 iterações, velocidade limitada entre -1 e 1, constante cognitiva de 0,5 e constante social de 1,5.

*Figura 23: Gráficos para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional.*



*Fonte: Elaborado pelo autor (2023).*

Analizando os dados acima, extrai-se que, com as configurações estabelecidas dos hiperparâmetros envolvidos no PSO, considerando 100 iterações, o valor mínimo da função à ser minimizada é 0, bem como seu valor médio é consideravelmente baixo, o que comprova a eficácia do algoritmo em termos de resultado bem como de otimização e gasto de recursos computacionais.

*Tabela 4: Dados para analisar a resposta do algoritmo PSO no processo de otimização de uma função bidimensional.*

ID da execução	1	2	3	4	5	6	7	8	Média
Valor Mínimo da Função	0	0	0	0	0	0	0	0	0
Valor Médio da Função	0.1425	0.0429	0.2517	0.1805	0.0249	0.0968	0.3143	0.8298	0.23543

*Fonte: Elaborado pelo autor (2023).*

#### 1.4.4 Comparação da performance do PSO e do Algoritmo Genético no processo de otimização da função bidimensional

Buscando realizar uma análise comparativa entre a performance do PSO e do algoritmo genético no processo de minimização da função proposta, considerou otimizar os hiperparâmetros existentes nos dois códigos (Tabela 5).

*Tabela 5: Hiperparâmetros do algoritmo genético e PSO para otimização da função proposta.*

Hiperparâmetros	AG	Hiperparâmetros	PSO
Quantidade de Indivíduos	50	Quantidade de Partículas	50
Quantidade de Gerações	100	Quantidade de Iterações	100
Taxa de Crossover	60%	Range de Variação de Velocidade	-1 à 1
Taxa de Mutação	1%	Constante Cognitiva	0.5
Metodo de Seleção	Roleta	Constante Social	1.5

*Fonte: Elaborado pelo autor (2023).*

Em relação ao PSO, a performance gráfica do algoritmo pode ser observada na Figura 23. O desempenho do algoritmo genético, por sua vez, é obtido na Figura 24.

*Figura 24: Gráfico para analisar a resposta do algoritmo genético no processo de otimização de uma função bidimensional.*



*Fonte: Elaborado pelo autor (2023).*

Em resumo, extraí-se os valores presentes na Tabela 6.

*Tabela 6: Dados para comparar a resposta do PSO com o algoritmo genético no processo de otimização de uma função bidimensional.*

Avaliação da Performance	AG	Avaliação da Performance	PSO
Mínimo da Função	> 0	Mínimo da Função	0
Média da Função	6.3605	Média da Função	0.2354

*Fonte: Elaborado pelo autor (2023).*

Entende-se, analisando as informações expostas acima, que o desempenho do PSO é superior ao algoritmo genético para o caso em questão, uma vez que atinge um mínimo de função menor, bem como uma média mais baixa. Como, nesse caso, o objetivo é minimizar a função proposta, um menor global mais relevante é obtido utilizando, justamente, o PSO. Cabe esclarecer que a utilização dos algoritmos bem como suas respectivas performances variam conforme a necessidade e objetivo proposto.

## 2. Questão 2

Nessa seção, serão apresentados os tópicos relacionados a resolução da questão 2 proposta no trabalho 3 da matéria “Computação Inspirada pela Natureza”, referente ao Programa de Pós-graduação em Ciência da Computação da UNESP.

### 2.1 Enunciado

Utilize o ACO para encontrar o menor caminho no problema do Caixeiro Viajante com as 52 cidades do arquivo `berlin52.tsp`, disponível no Google Classroom, e representado na figura ao lado.

Inclua em seu relatório um gráfico que mostre a distância total do menor caminho encontrado ao longo das iterações, além das configurações e outros resultados que julgar interessantes.

Inclua também um gráfico mostrando as cidades em suas respectivas coordenadas e o traçado do menor caminho encontrado, para uma análise visual do resultado.

### 2.2 Linguagem e IDE

O código referente a questão 1 foi desenvolvido em *Python* [1], com suporte do *Google Colaboratory* [2].

### 2.3 Descrição lógica do algoritmo

Para a implementação do algoritmo, é necessário que algumas bibliotecas sejam importadas: *random* [3], *matplotlib* [4], *pandas* [6], *io* [7], *math* [8], *itertools* [9], *scipy* [10] e *numpy* [11]. De maneira resumida, o suporte das bibliotecas permite a manipulação com aleatoriedade, gráficos, banco de dados, iteradores e recursos matemáticos e vetoriais.

Na sequência, os hiperparâmetros são definidos com o intuito de possibilitar a execução bem como a otimização do algoritmo. i)  $\alpha$ , sendo o peso relacionado ao nível de feromônio da aresta; ii)  $\beta$ , sendo o peso atrelado à visibilidade das cidades; iii)  $\rho$ , sendo a taxa de evaporação do feromônio; iv) `num_formigas`, sendo o número de formigas existentes; v)  $Q$ , sendo a quantidade total de feromônio depositado; vi)  $b$ , o número de formigas elitistas. vii)  $\tau_0$ , sendo a taxa inicial de feromônio [12].

Com os hiperparâmetros definidos, tem-se a necessidade de ler o banco de dados composto por coordenadas x e y que representam a posição de 52 cidades de Berlim. Para que a leitura seja possibilitada, a biblioteca *pandas* [6] é acionada, convertendo o arquivo .xlsx, que contém as informações em questão, em um *data frame*. Feito isso, é realizada uma verificação para confirmar se todos os dados lidos são números e, caso sejam, estes são armazenados em um vetor, através do recurso de tuplas.

Posterior à leitura dos dados, as formigas são inseridas aleatoriamente, graças ao auxílio da biblioteca *random* [3], nas 52 cidades existentes.

Na sequência, utilizando a biblioteca *itertools* [9], é gerada uma lista de combinações que tem como intuito armazenar todas as possibilidades combinatórias entre 2 cidades. O intuito dessa etapa é, justamente, entender quais são os caminhos possíveis que as formigas podem seguir. Nessa lista, utilizando o recurso existente na biblioteca supracitada, combinações entre duas cidades iguais são consideradas e, como nesse caso, tais situações não representam cenários físicos para a resolução do problema em questão, uma vez que não possibilita deslocamento das formigas, estes são desconsiderados.

Com todas as combinações válidas definidas, calcula-se a distância entre cada uma das cidades bem como a visibilidade destas. Por visibilidade, uma vez que se almeja a minimização da função proposta, entende-se como o inverso das distâncias em análise.

Tendo as distâncias e visibilidades bem definidas, surge a necessidade de encontrar a probabilidade de que as formigas, inseridas no mapa de maneira aleatória, escolham determinados caminhos. Para isso, considera-se o valor de feromônio presente em cada rota bem como a visibilidade de cada uma das cidades. A estes, relacionam-se, respectivamente, os pesos definidos como  $\alpha$  e  $\beta$ .

Com as probabilidades já calculadas, parte-se para a lógica iterativa que permite que os melhores caminhos sejam escolhidos. Inicialmente, tem-se a necessidade de inicializar três variáveis: i) melhores\_distancias, responsável por armazenar as menores distâncias obtidas; ii) melhor\_distancia, responsável por armazenar o melhor valor global; iii) melhor\_rota, responsável por armazenar o melhor caminho encontrado.

Inicia-se, então, um loop iterativo. Nele, para cada formiga, é criada uma lista de controle que visa, justamente, entender e armazenar quais cidades já foram visitadas. Junto à esse processo, as formigas caminham entre as coordenadas existentes, com base na probabilidade de

visitação de cada uma das cidades, de modo a visitarem o mapa de Berlim em sua total completude. Essa rota, bem como a distância total de tal rota, é armazenada.

Quando todo o mapa é percorrido por todas as formigas existentes, cabe-se inicializar o processo de atualização de feromônios. A lógica é que as arestas mais visitadas apresentem maior quantidade de feromônios, estimulando a passagem dos outros insetos por lá. Por outro lado, as arestas que não foram percorridas tantas vezes, possuirão menos feromônio, atraindo menos insetos. O processo de atualização de feromônios consiste em depositar feromônio de maneira inversamente proporcional ao tamanho das arestas, estimulando que, dessa forma, os caminhos mais curtos sejam incentivados e priorizados. Como complemento, durante o procedimento de atualização de feromônios, é importante que os melhores globais sejam armazenados e, durante tal processo, sejam inseridas formigas elitistas, que potencializarão o depósito de feromônio nas rotas encolhidas. Adicionalmente, é importante que o feromônio evapore com o passar do tempo e, para isso, utiliza-se a variável  $\rho$  à título de reduzir a quantidade da substância nas arestas das cidades conforme o passar do tempo.

Na sequência, as probabilidades são atualizadas com base nos novos feromônios e o processo iterativo se reinicia. Por fim, gráficos que permitem entender a performance do ACO são plotados e avaliados.

## 2.4 Análise do algoritmo

A fim de realizar uma análise relacionada ao algoritmo desenvolvido, o código foi executado algumas vezes e, durante esse processo, alguns parâmetros foram alterados e resultados foram coletados e interpretados.

### 2.4.1 Influência da quantidade de iterações no processo de otimização do problema TSP utilizando ACO

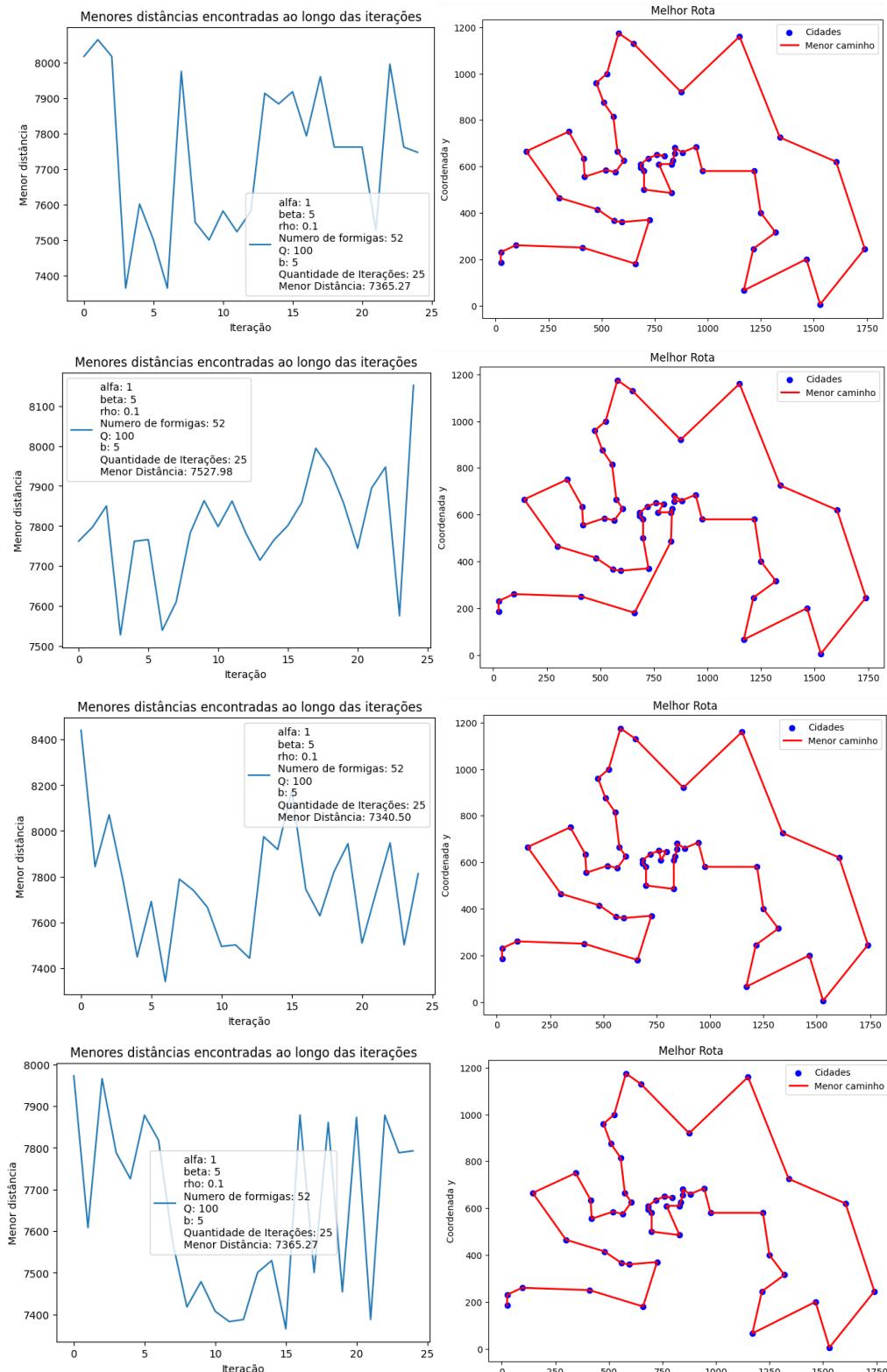
A primeira análise consistiu em verificar a influência da quantidade de iterações atreladas ao algoritmo. Para isso, os outros hiperparâmetros foram mantidos constantes, variando somente a quantidade de loops realizado por cada uma das formigas em questão.

À título de informação, considerou-se um  $\alpha$  de 1, um  $\beta$  de 5, um  $\rho$  de 0,1, 52 formigas, um  $Q$  de 100, um  $b$  de 5 e um  $\tau$  de  $10^{-6}$ . O primeiro parâmetro refere-se à um peso que se relaciona ao feromônio, o segundo, à visibilidade das cidades. O  $\rho$ , por sua vez, relaciona-se ao

processo de evaporação do feromônio. A quantidade de formigas foi definida almejando que fosse igual ao número de cidades disponíveis no problema de TSP. O Q é a quantidade de feromônio depositado, o b relaciona-se às formigas elitistas e, por fim, o  $\tau$  tem como objetivo armazenar as taxas de feromônios existentes em cada uma das arestas entre as cidades.

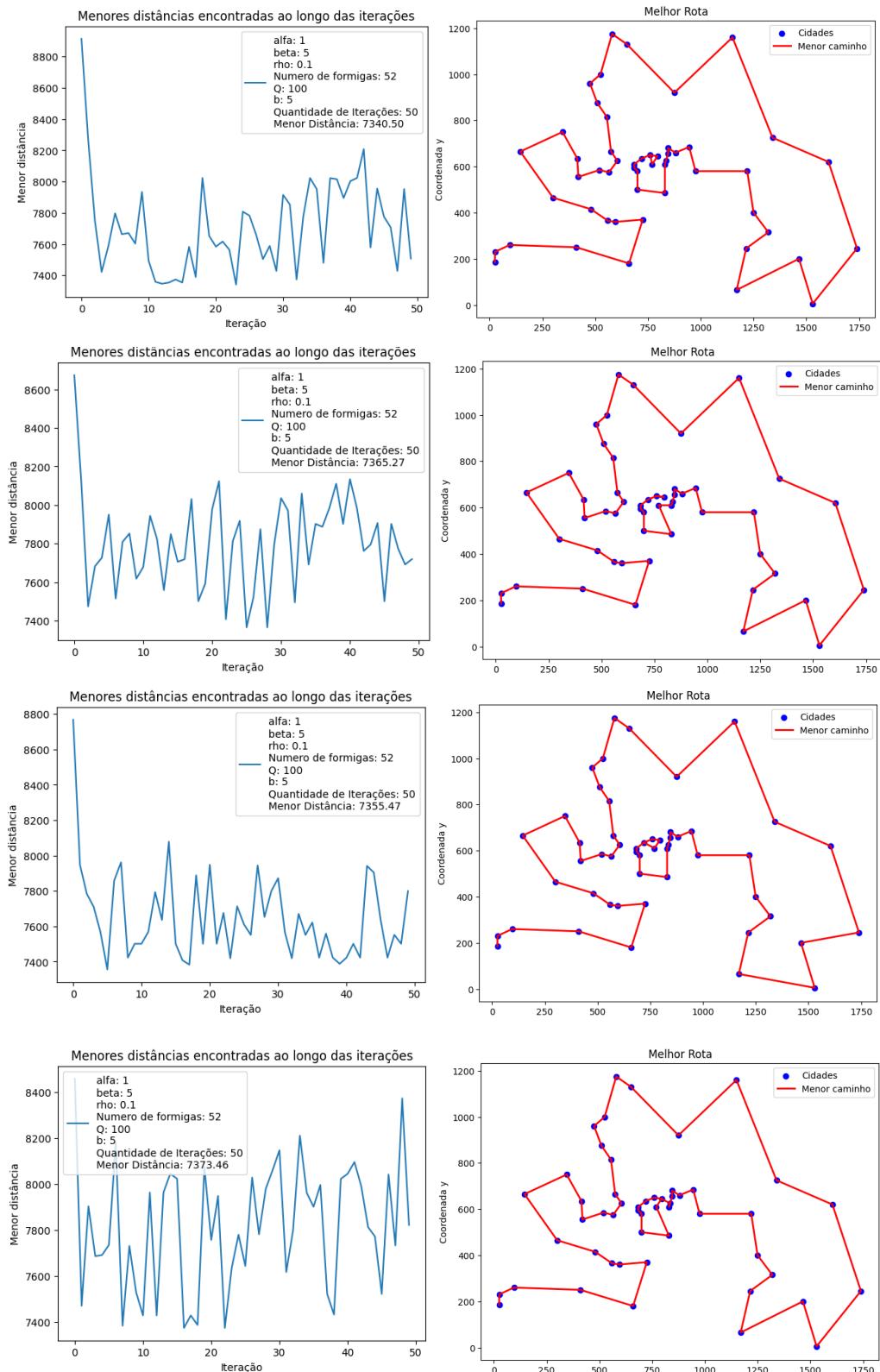
Com tais informações em mente, considerou-se 25, 50 e 100 iterações e, para cada um dos casos, simularam-se os resultados 4 vezes.

Figura 25: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando 25 iterações.



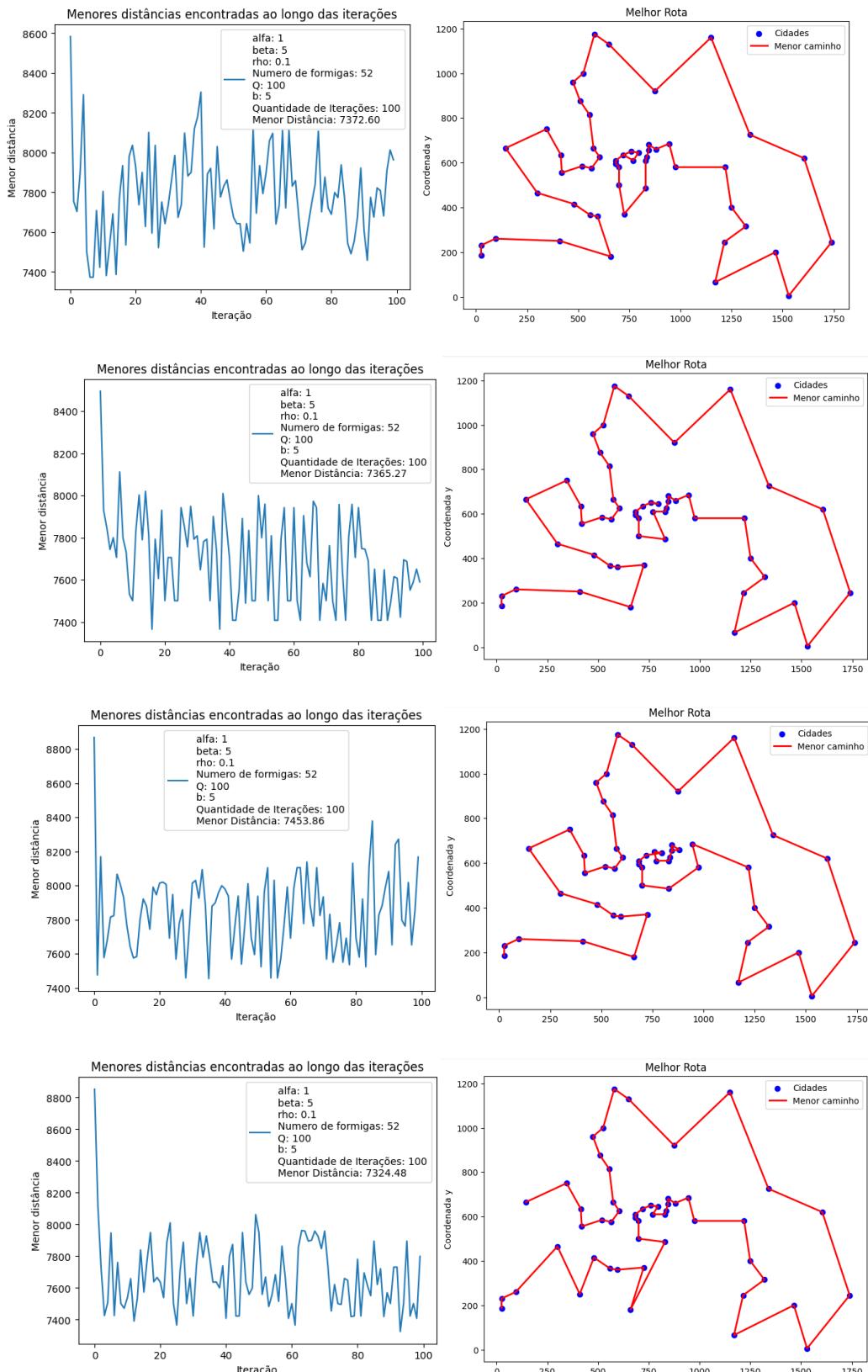
Fonte: Elaborado pelo autor (2023).

Figura 26: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando 50 iterações.



Fonte: Elaborado pelo autor (2023).

Figura 27: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando 100 iterações.



Fonte: Elaborado pelo autor (2023).

Extrai-se, das Figuras 25, 26, 27, as informações contidas na Tabela 7.

*Tabela 7: Resumo da influência a quantidade de iterações no processo de otimização do TSP utilizando o ACO.*

Influência da Quantidade de Iterações na Performance do ACO			
Quantidade de Iterações	25	50	100
Mínima Distância Encontrada	7399.76	7358.68	7379.05

*Fonte: Elaborado pelo autor (2023).*

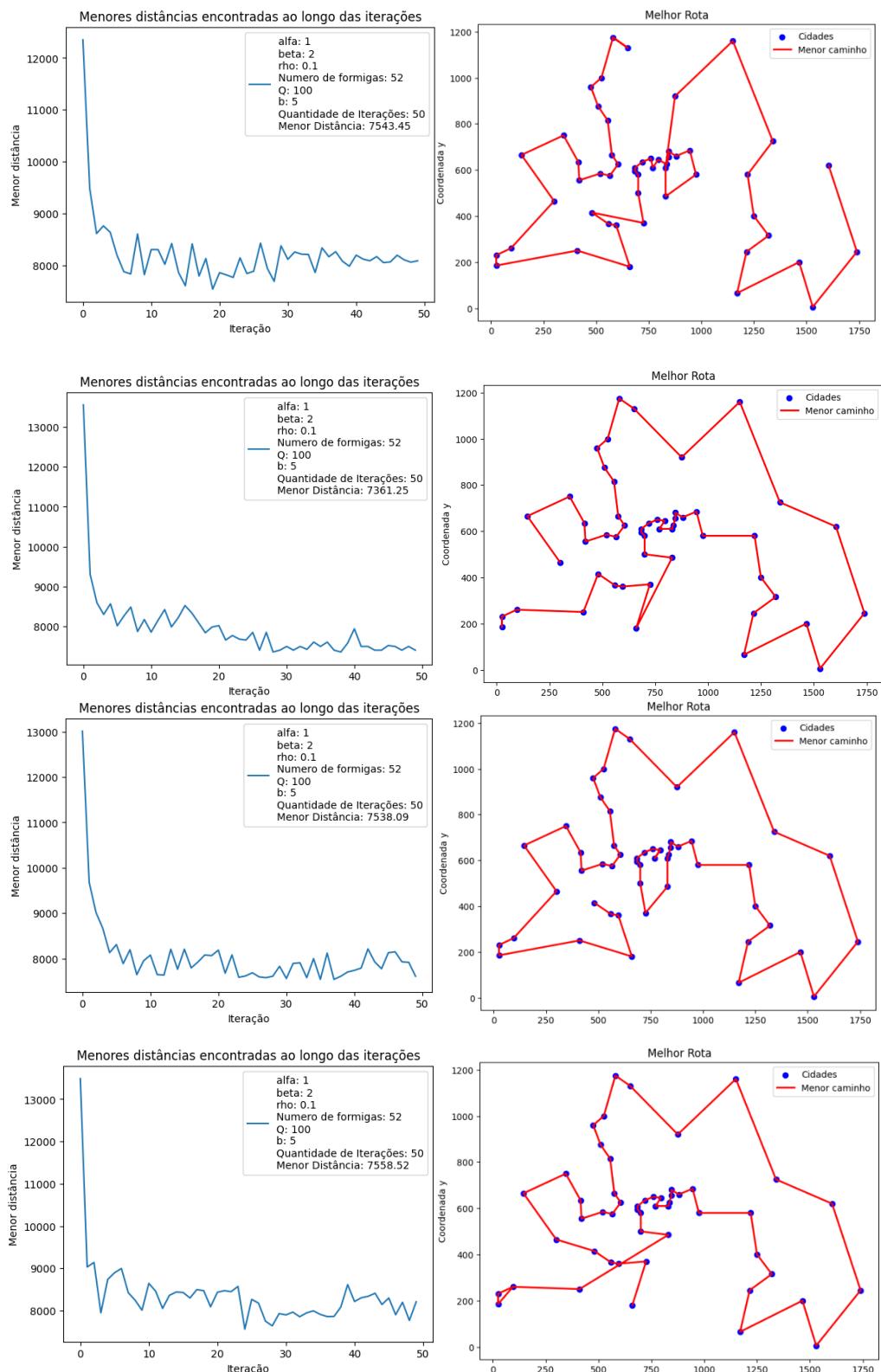
É importante citar que, no algoritmo proposto, com o aumento de iterações, o tempo de execução tende a aumentar e, portanto, é importante considerar tal evolução de gasto computacional e compará-lo com o aumento da performance obtida.

Considerando o exposto acima, percebe-se que 50 iterações são tidas como suficientes para a obtenção de um resultado satisfatório, atingido, inclusive, em tempo hábil. Portanto, esse número foi adotado e considerado ao longo dos outros experimentos.

#### **2.4.2 Influência das variáveis $\alpha$ e $\beta$ no processo de otimização do problema TSP utilizando ACO**

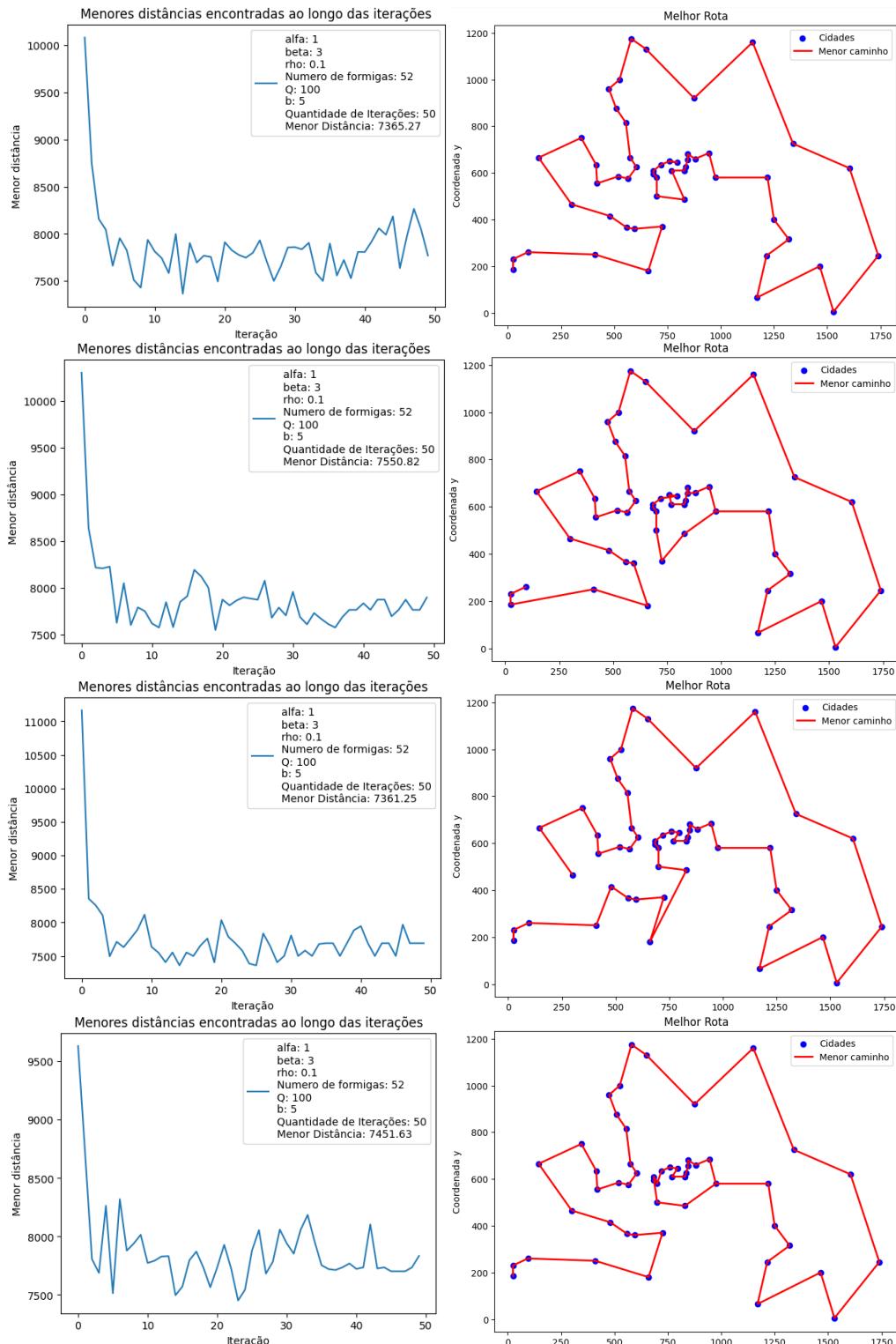
A segunda análise objetivou variar os parâmetros  $\alpha$  e  $\beta$ , que se associam, respectivamente, ao feromônio e a visibilidade das cidades. Nesse cenário, uma vez que o TSP visa, justamente, encontrar uma menor distância, considerou-se, em todas as simulações, um  $\alpha$  menor que  $\beta$ , de modo a priorizar, no processo de ajuste probabilístico e, consequentemente, na otimização do problema proposto, a distância envolvida entre as cidades do data set. Para cada um dos casos, foram realizadas 4 simulações e, em tais execuções, os hiperparâmetros adicionais à  $\alpha$  e  $\beta$  foram mantidos fixos.

Figura 28: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 1 e um  $\beta$  de 2.



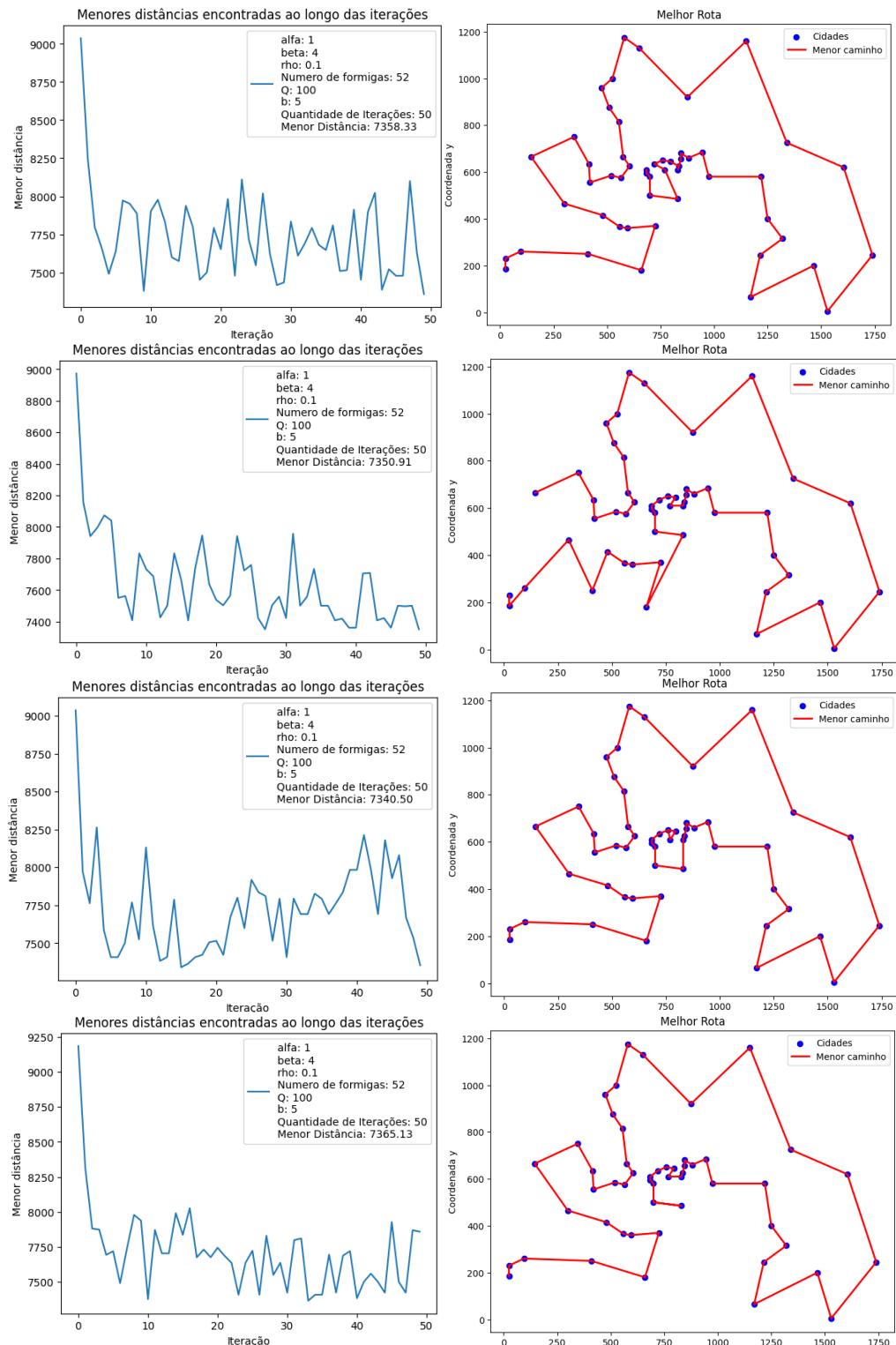
Fonte: Elaborado pelo autor (2023).

Figura 29: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 1 e um  $\beta$  de 3.



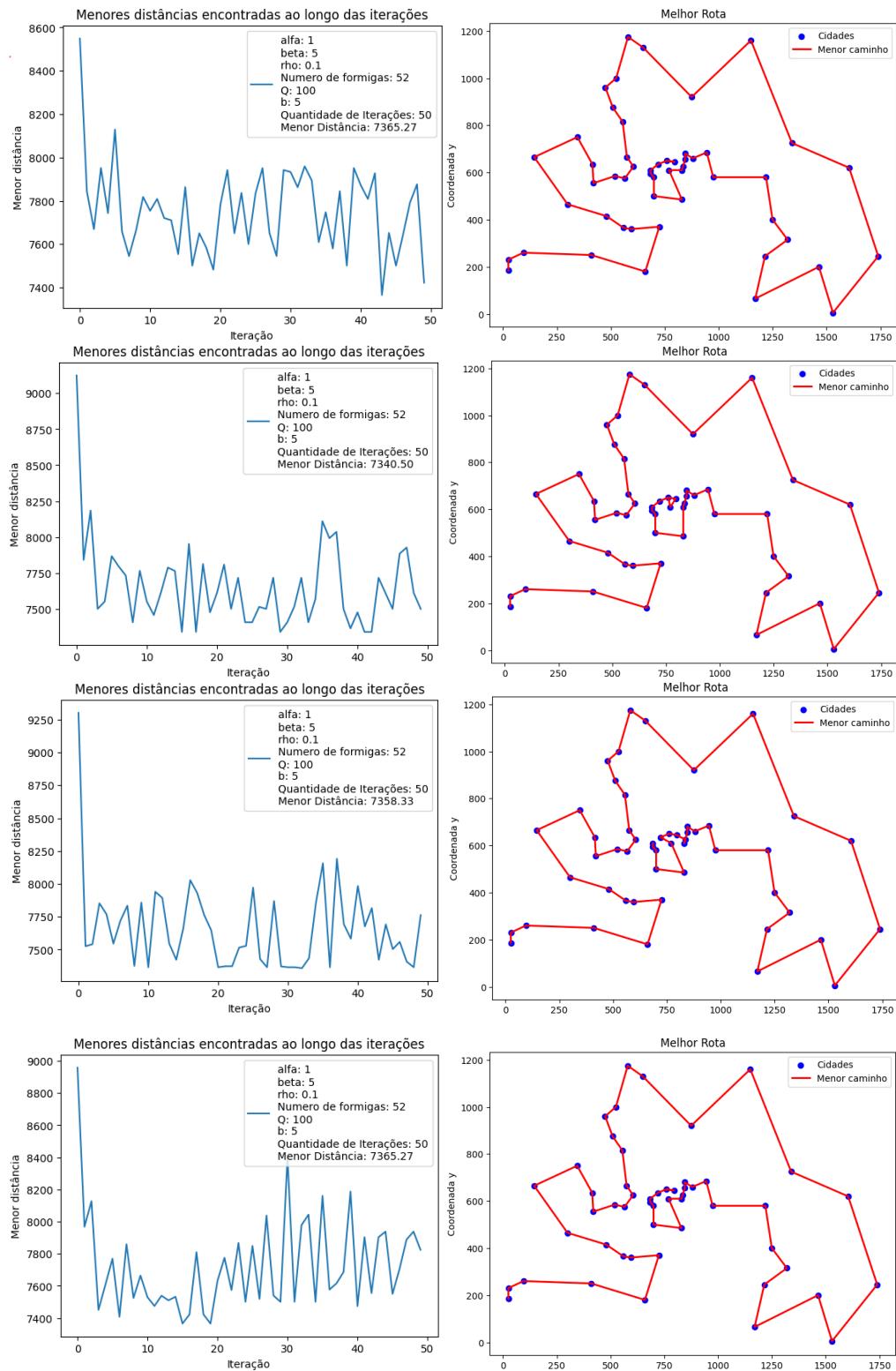
Fonte: Elaborado pelo autor (2023).

Figura 30: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 1 e um  $\beta$  de 4.



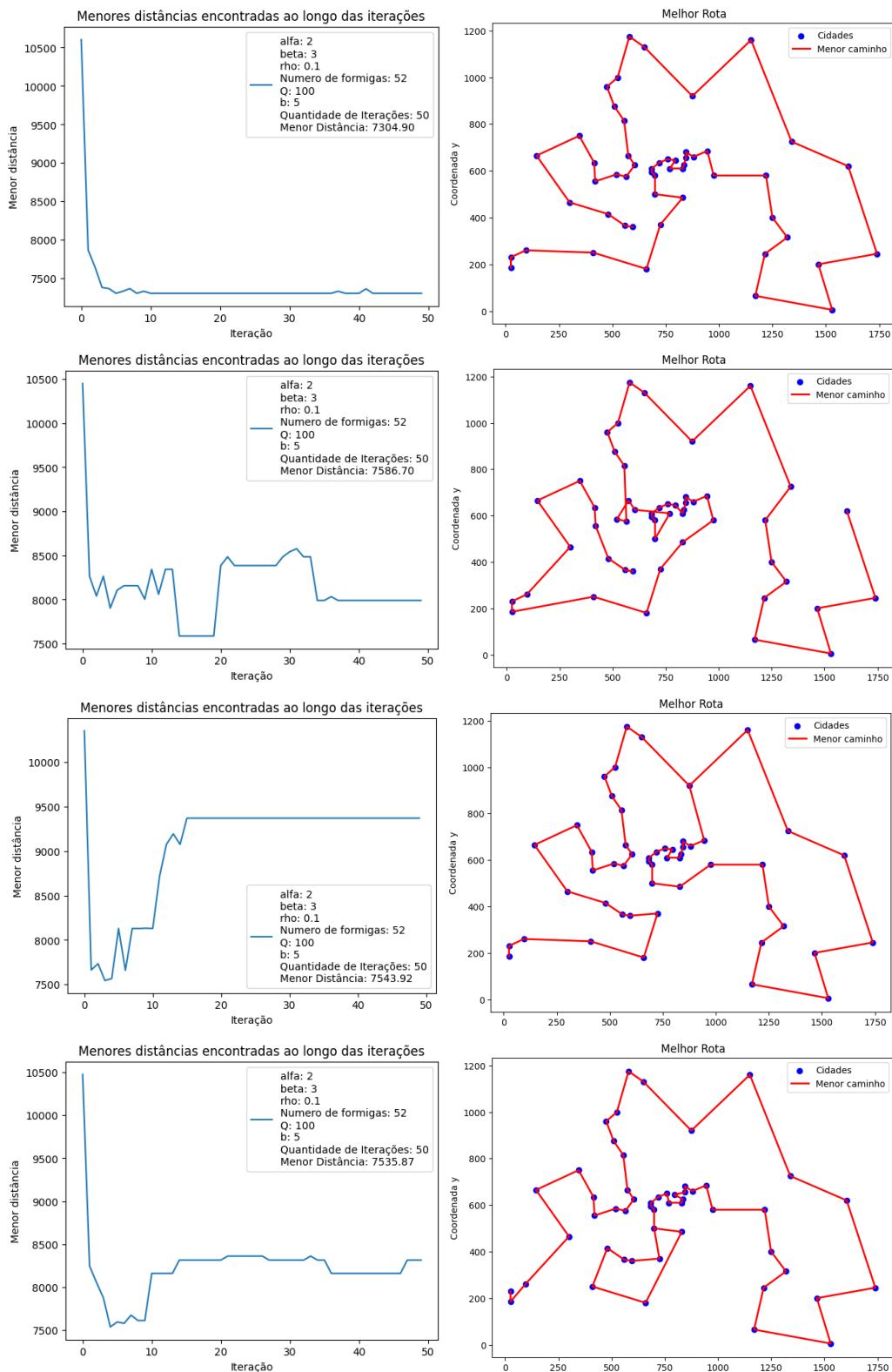
Fonte: Elaborado pelo autor (2023).

Figura 31: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 1 e um  $\beta$  de 5.



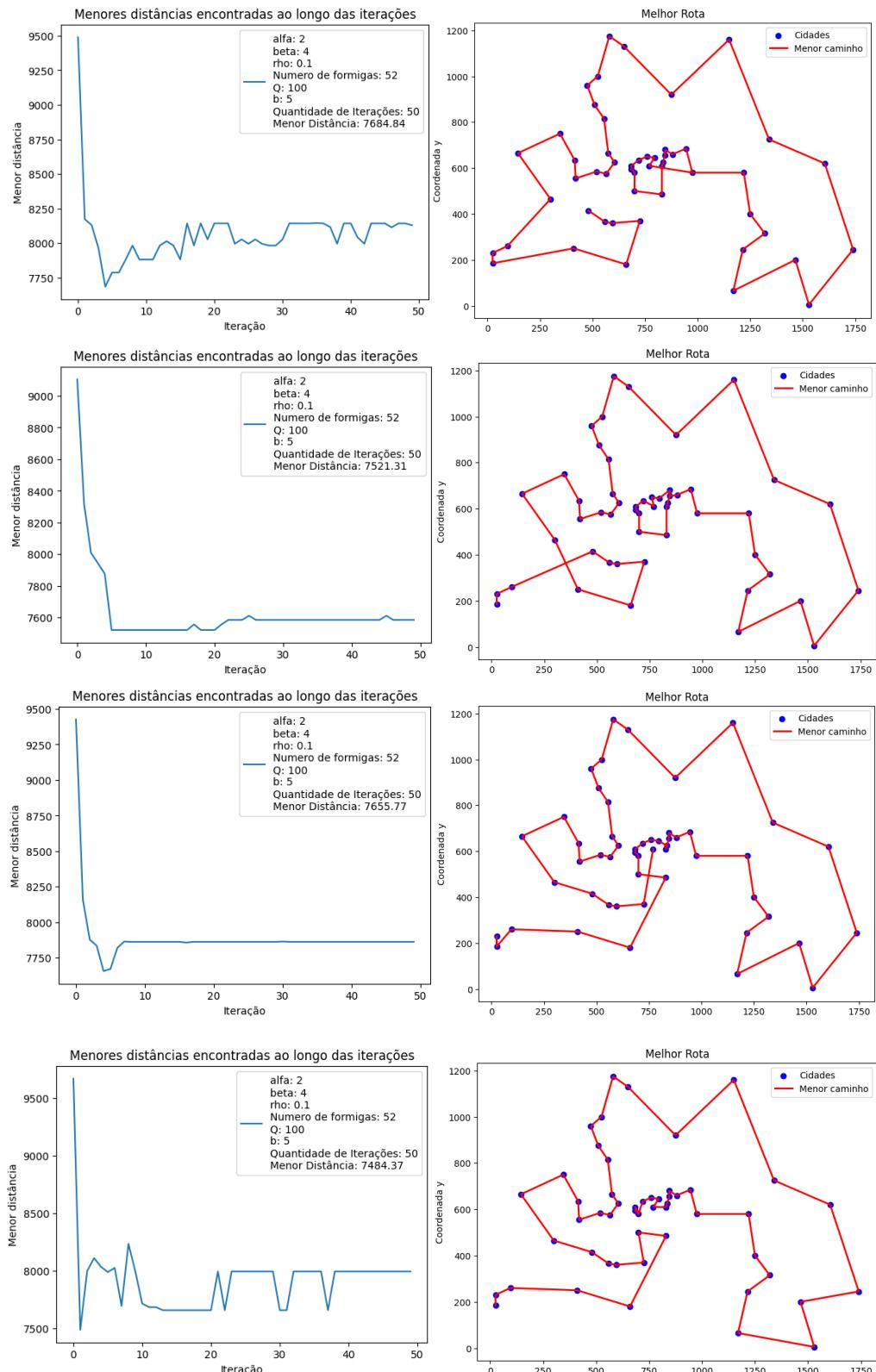
Fonte: Elaborado pelo autor (2023).

Figura 32: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 2 e um  $\beta$  de 3.



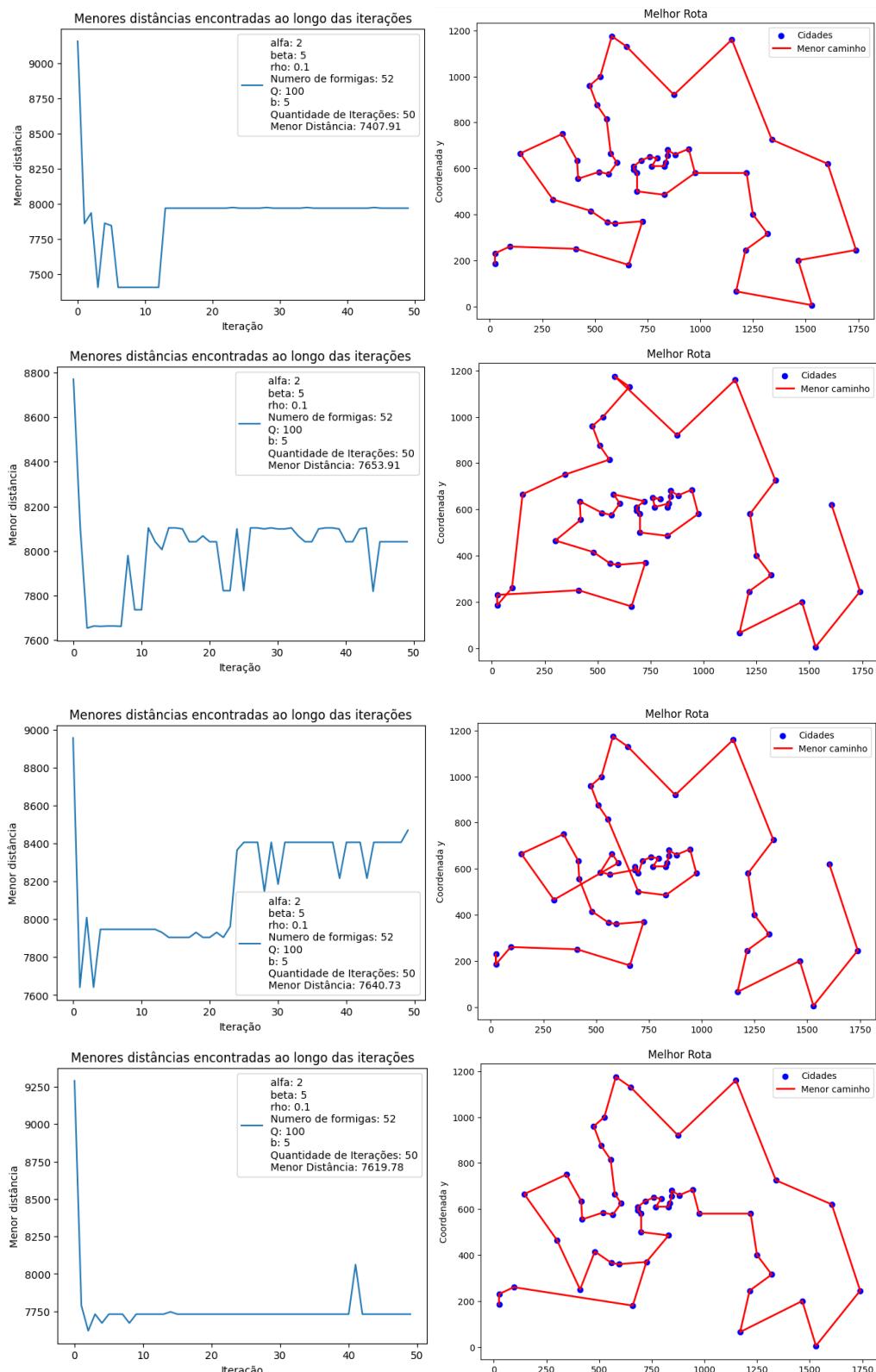
Fonte: Elaborado pelo autor (2023).

Figura 33: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 2 e um  $\beta$  de 4.



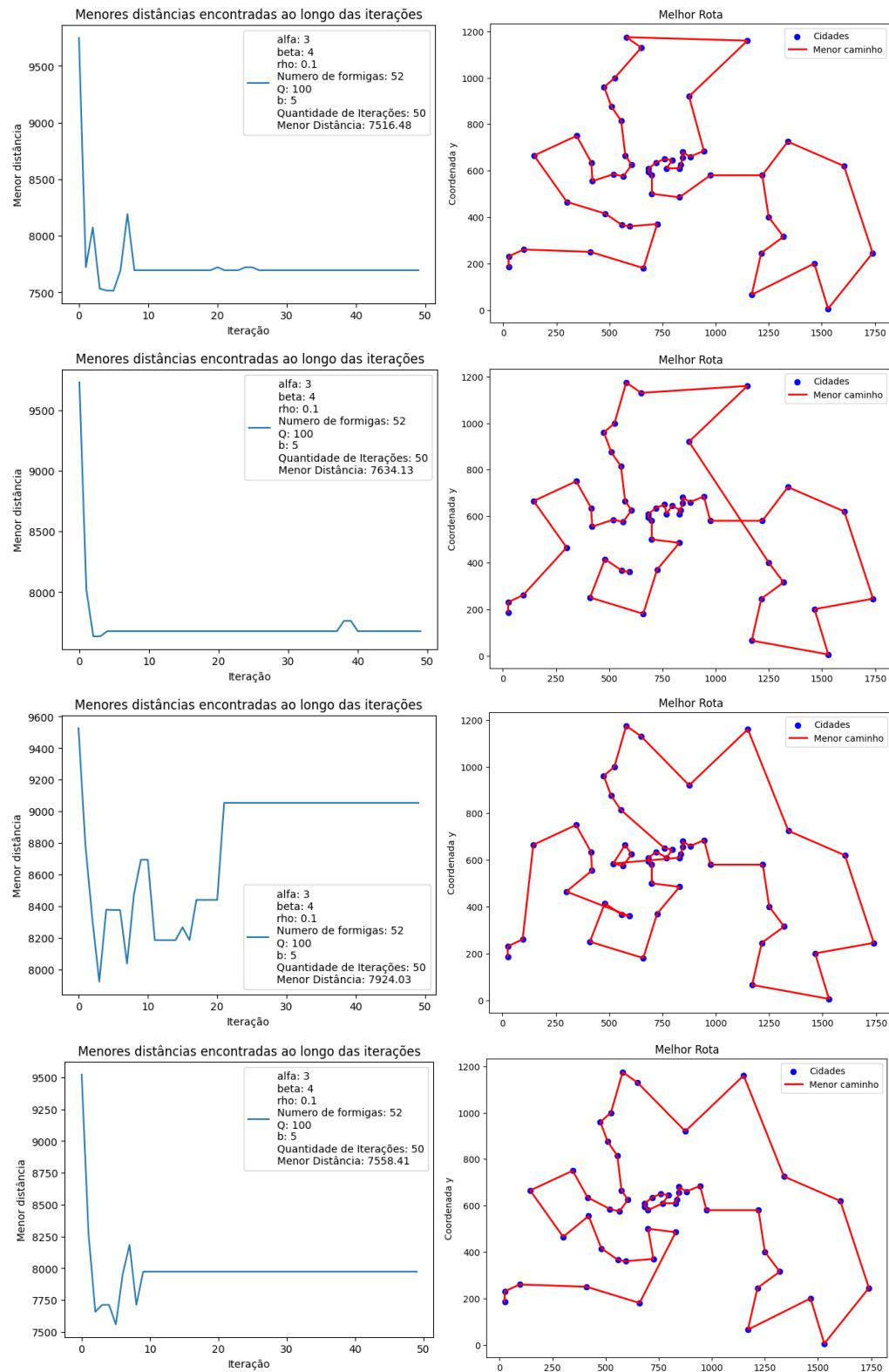
Fonte: Elaborado pelo autor (2023).

Figura 34: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 2 e um  $\beta$  de 5.



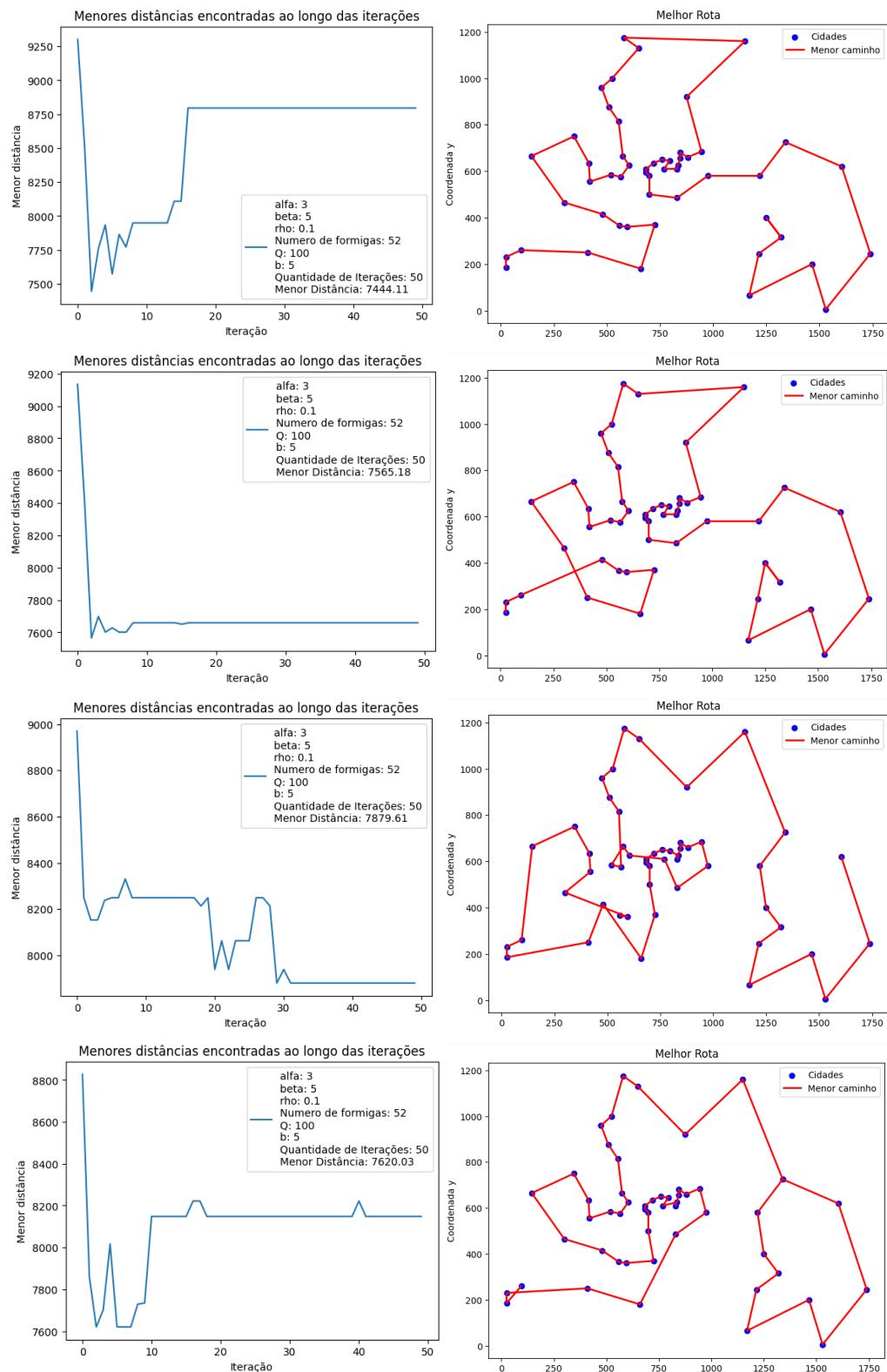
Fonte: Elaborado pelo autor (2023).

Figura 35: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 3 e um  $\beta$  de 4.



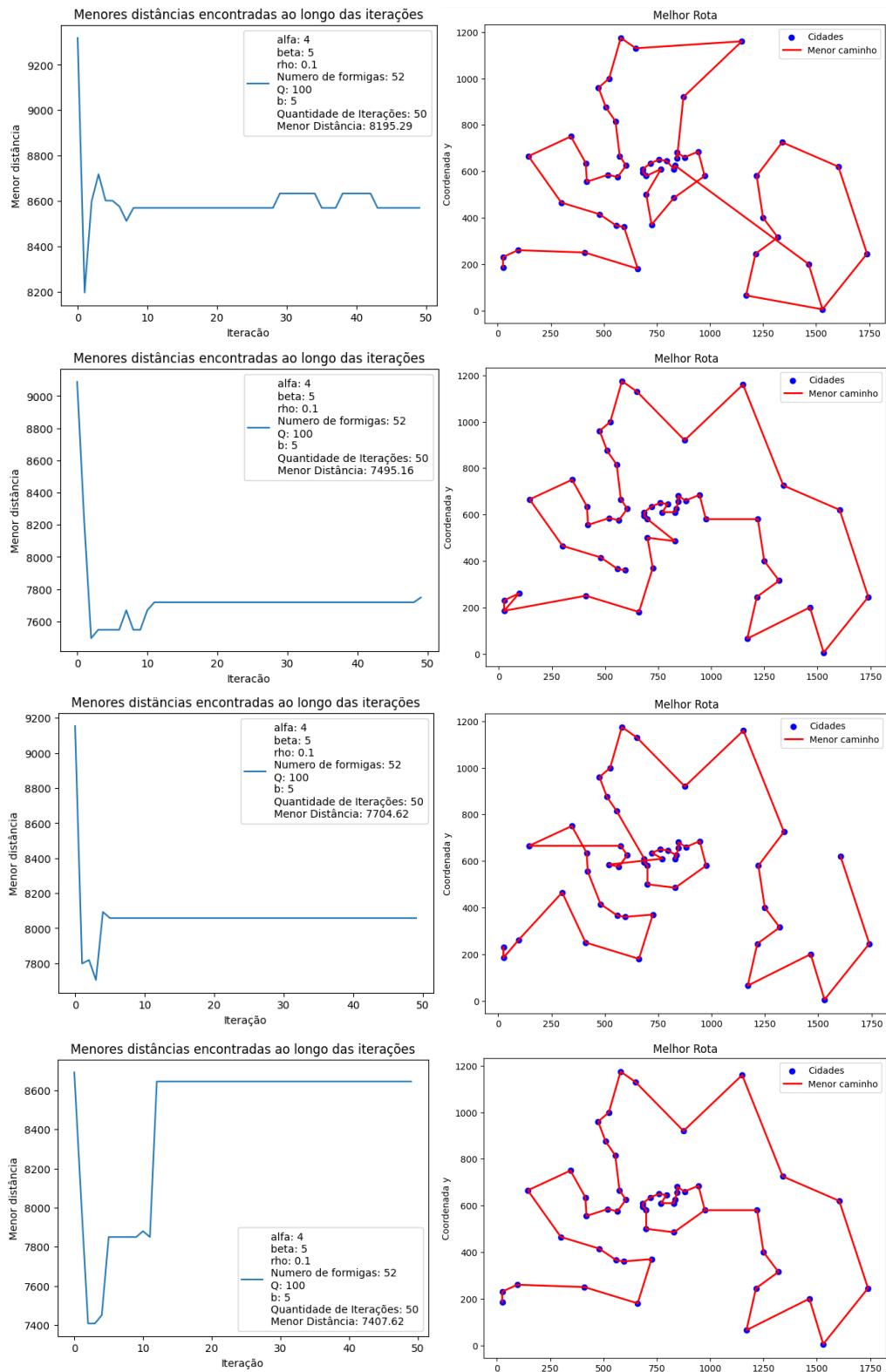
Fonte: Elaborado pelo autor (2023).

Figura 36: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 3 e um  $\beta$  de 5.



Fonte: Elaborado pelo autor (2023).

Figura 37: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\alpha$  de 4 e um  $\beta$  de 5.



Fonte: Elaborado pelo autor (2023).

Analizando as execuções apresentadas acima, tem-se a Tabela 8, abaixo, com as informações resumidas.

*Tabela 8: Análise da resposta do algoritmo ACO na resolução do TSP considerando variações de  $\alpha$  e  $\beta$ .*

		<b>Média de Distâncias Mínimas</b>			
		<b><math>\alpha</math></b>			
		<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b><math>\beta</math></b>	<b>2</b>	7500.33	X	X	X
	<b>3</b>	7432.24	7492.85	X	X
	<b>4</b>	7353.72	7586.57	7658.26	X
	<b>5</b>	7357.34	7580.58	7627.23	7700.67

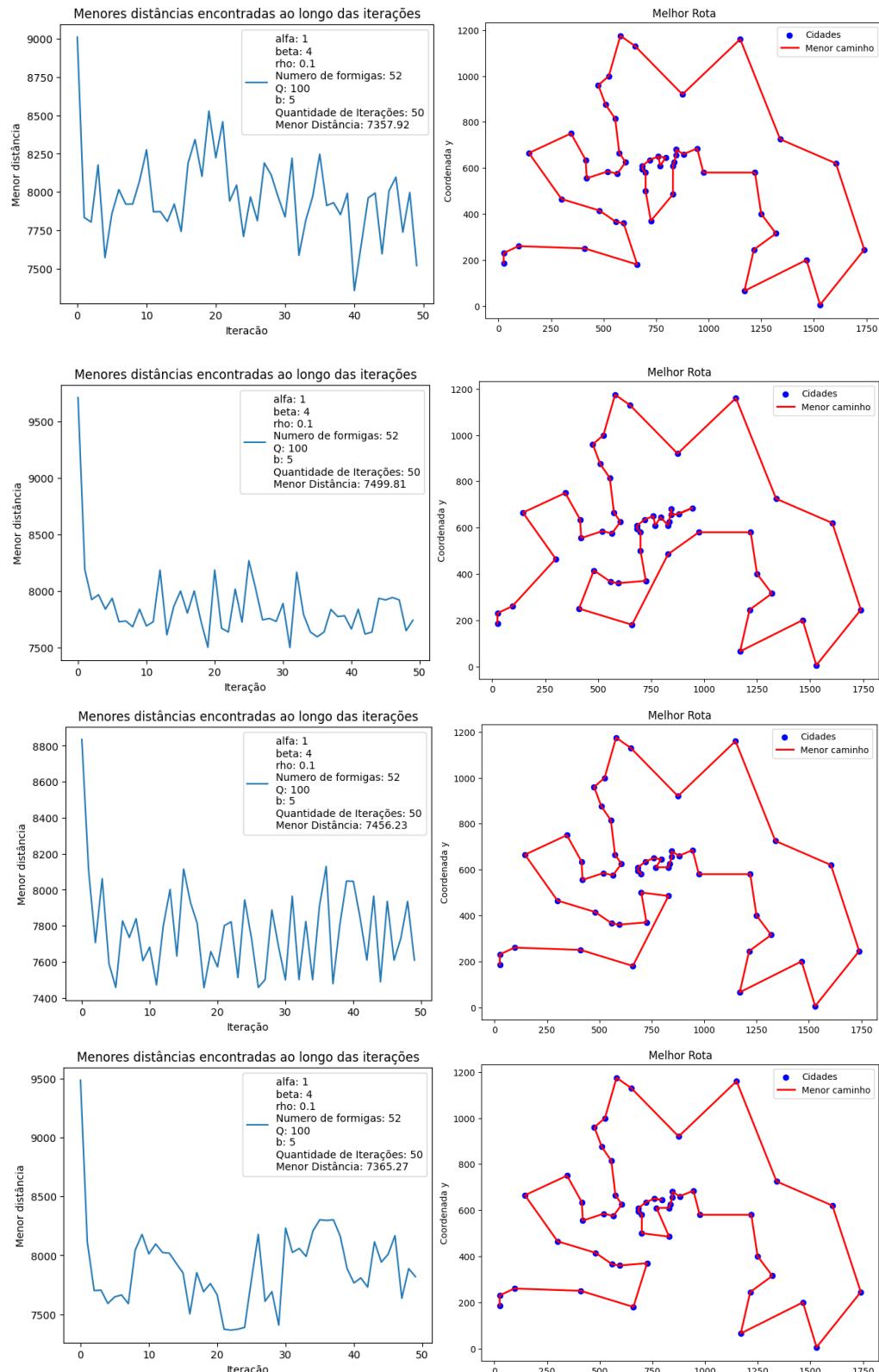
*Fonte: Elaborado pelo autor (2023).*

Conclui-se, portanto, que a melhor performance do ACO é obtida utilizando um  $\alpha$  de 1 e um  $\beta$  de 4, haja vista um atingimento de média de mínima distância menor do que os outros casos expostos. Tido isso, para os próximos experimentos foram considerados os valores explicitados acima.

#### **2.4.3 Influência da variável $\rho$ no processo de otimização do problema TSP utilizando ACO**

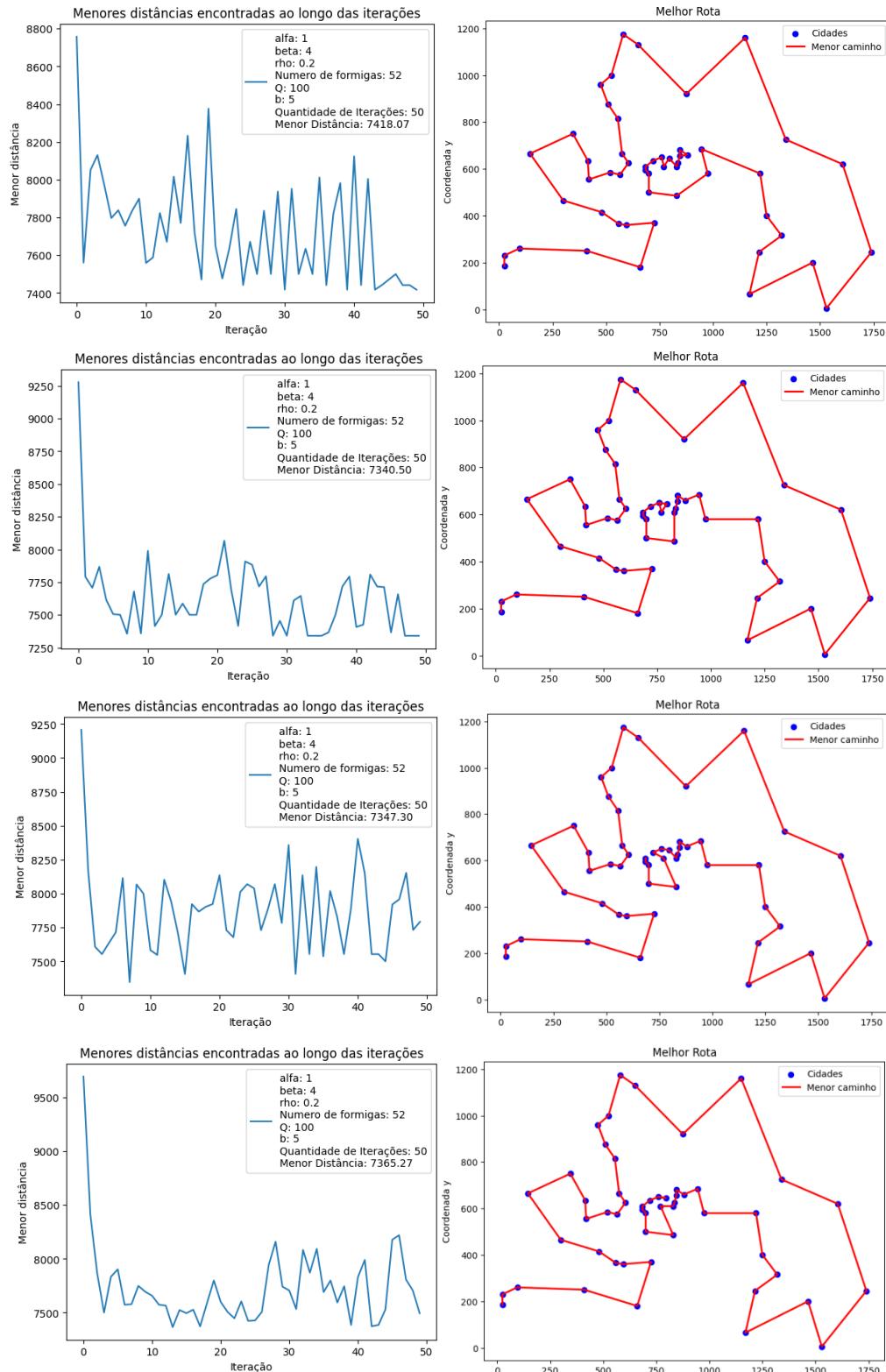
O terceiro experimento consistiu em variar o  $\rho$ , elemento atrelado à evaporação de feromônio no algoritmo de ACO para resolução do TSP proposto. O  $\rho$  foi variado entre 0,1 e 0,2 e, para cada um dos casos, o código foi executado 4 vezes. Esses valores foram considerados visando, justamente, uma não evaporação completa dos feromônios deixados nas arestas do percurso percorrido pelas formigas. Valores mais altos, instigariam uma evaporação mais rápida, favorecendo o processo anteriormente descrito.

Figura 38: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\rho$  de 0.1.



. Fonte: Elaborado pelo autor (2023).

Figura 39: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $\rho$  de 0.2.



. Fonte: Elaborado pelo autor (2023).

As informações das Figuras 38 e 39 são compiladas na Tabela 9.

*Tabela 9: Análise da resposta do algoritmo ACO na resolução do TSP considerando variações de  $\rho$ .*

<b>Média de Distâncias Mínimas</b>	
<b><math>\rho</math></b>	
<b>0.1</b>	<b>0.2</b>
7419.8075	7367.785

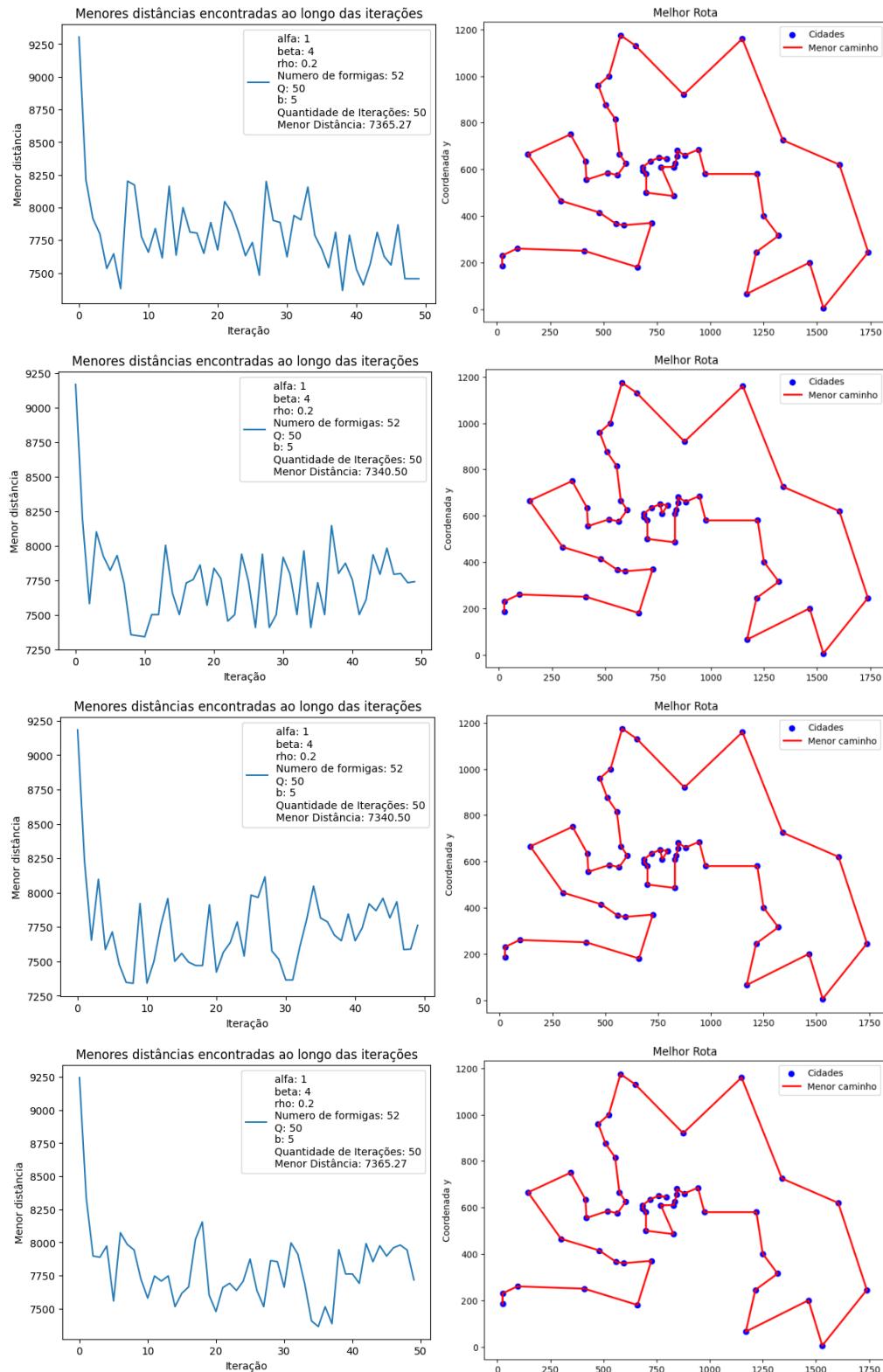
*Fonte: Elaborado pelo autor (2023).*

Observando a Tabela 9, percebe-se que os valores obtidos com a variação de  $\rho$  são bastante próximos um do outro podendo, inclusive, ser desconsiderado, haja vista a existência de poucas execuções do código. Nesse sentido, embora a diferença entre os dois casos apresentados seja irrelevante, optou-se por seguir com o  $\rho$  igual à 0,2.

#### **2.4.4 Influência da variável Q no processo de otimização do problema TSP utilizando ACO**

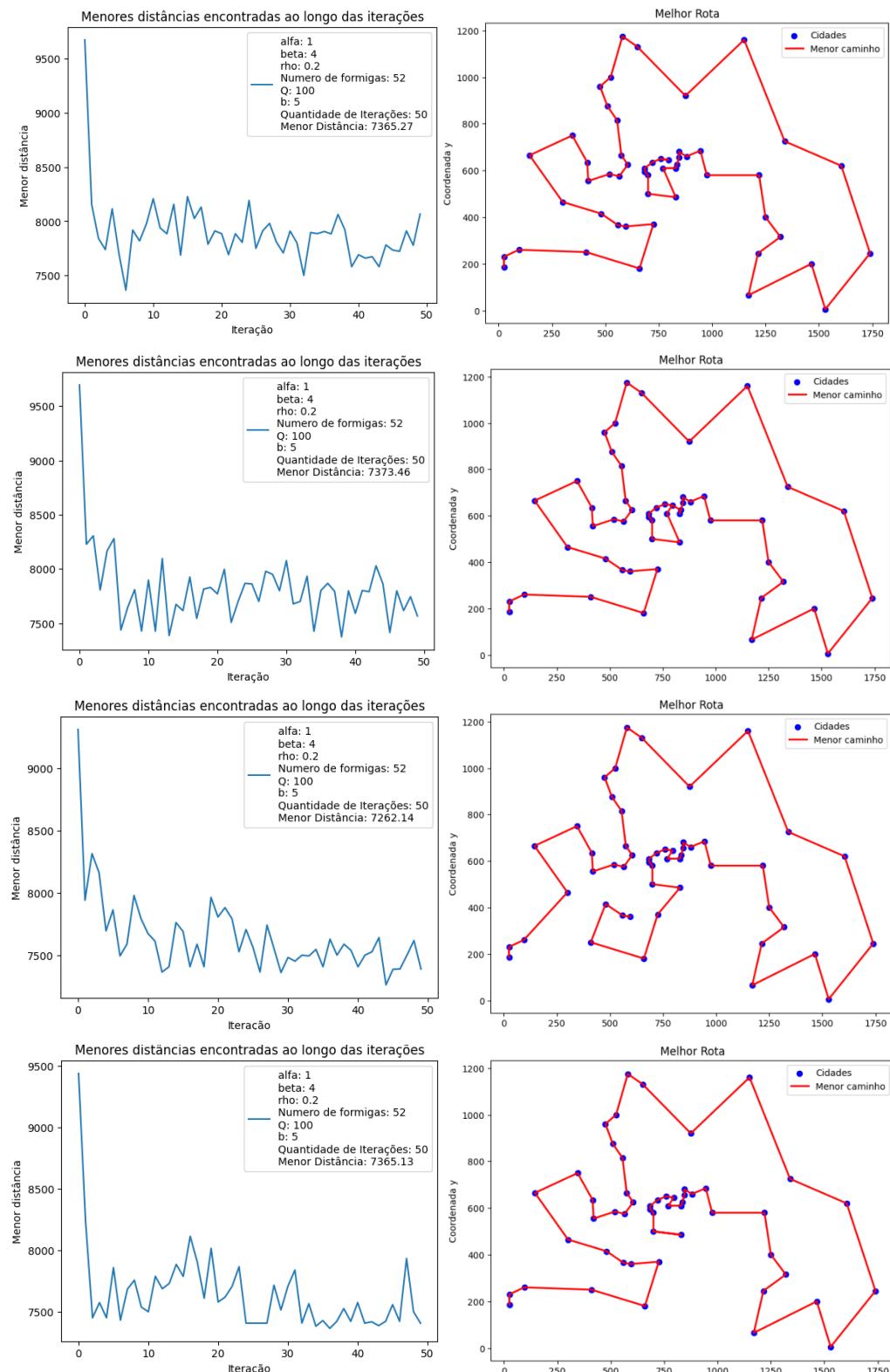
A variável Q relaciona-se a quantidade de feromônio depositado por cada formiga quando esta opta por seguir um trajeto. A fim de testar a quantidade ideal, variou-se Q em um range de 50 a 200. Para cada configuração, executou-se o código 4 vezes.

Figura 40: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $Q$  de 50.



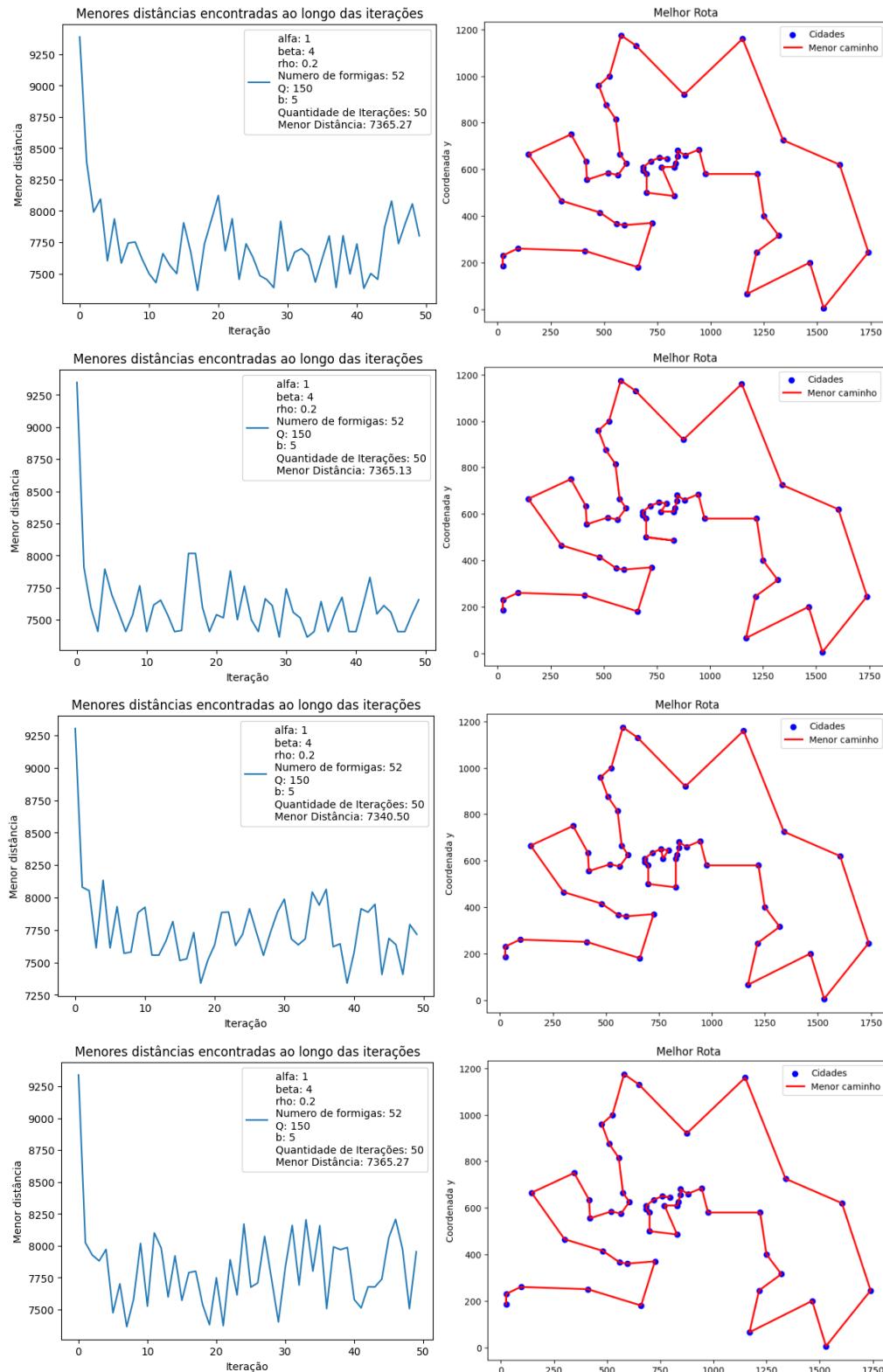
Fonte: Elaborado pelo autor (2023).

Figura 41: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $Q$  de 100.



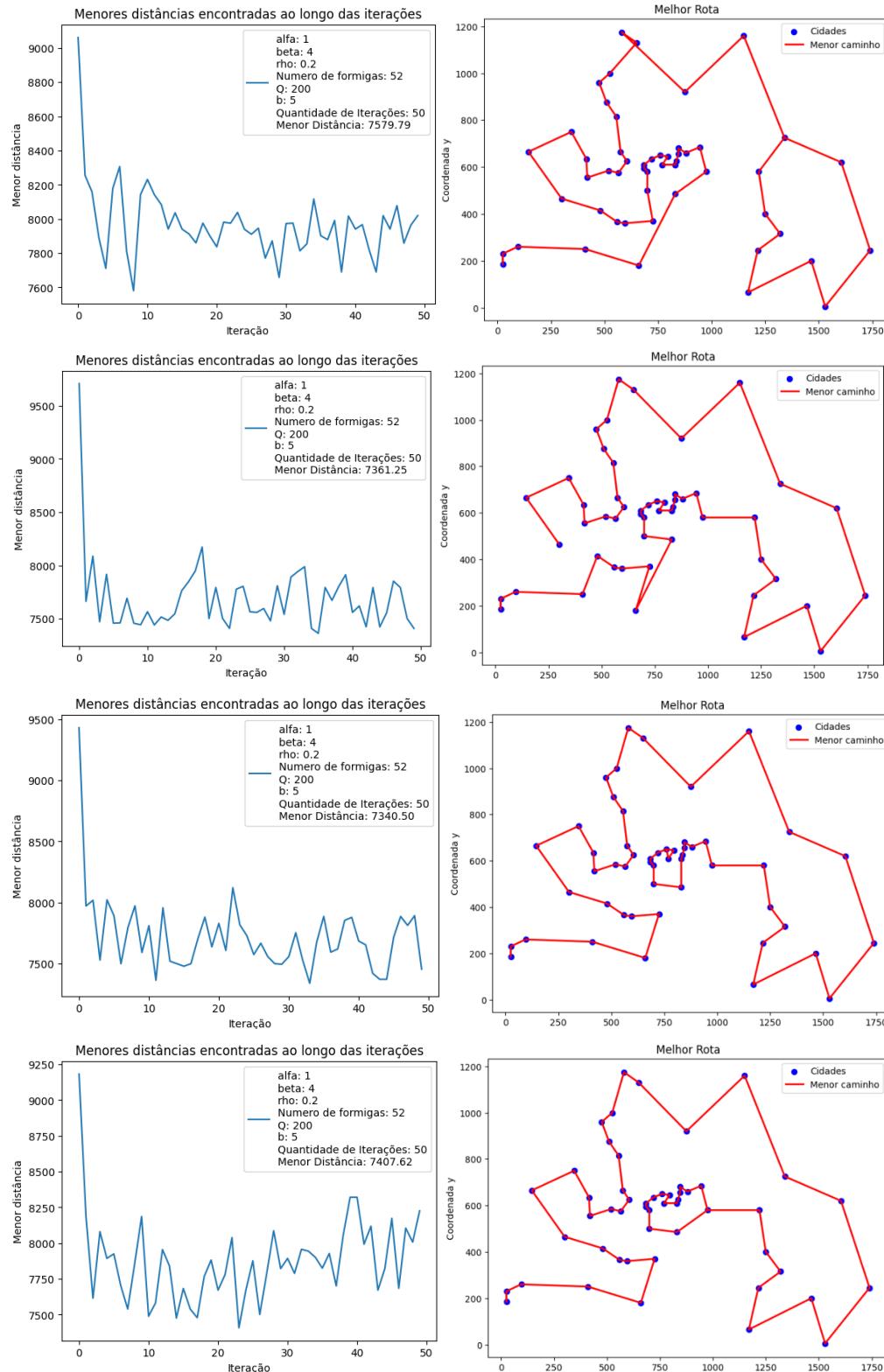
Fonte: Elaborado pelo autor (2023).

Figura 42: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $Q$  de 150.



Fonte: Elaborado pelo autor (2023).

Figura 43: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $Q$  de 200.



Fonte: Elaborado pelo autor (2023).

Na Tabela 10, tem-se as informações gráficas apresentadas acima expostas de maneira resumida.

*Tabela 10: Análise da resposta do algoritmo ACO na resolução do TSP considerando variações de Q.*

<b>Média de Distâncias Mínimas</b>			
<b>Q</b>			
<b>50</b>	<b>100</b>	<b>150</b>	<b>200</b>
7352.89	7341.5	23912.2	7422.29

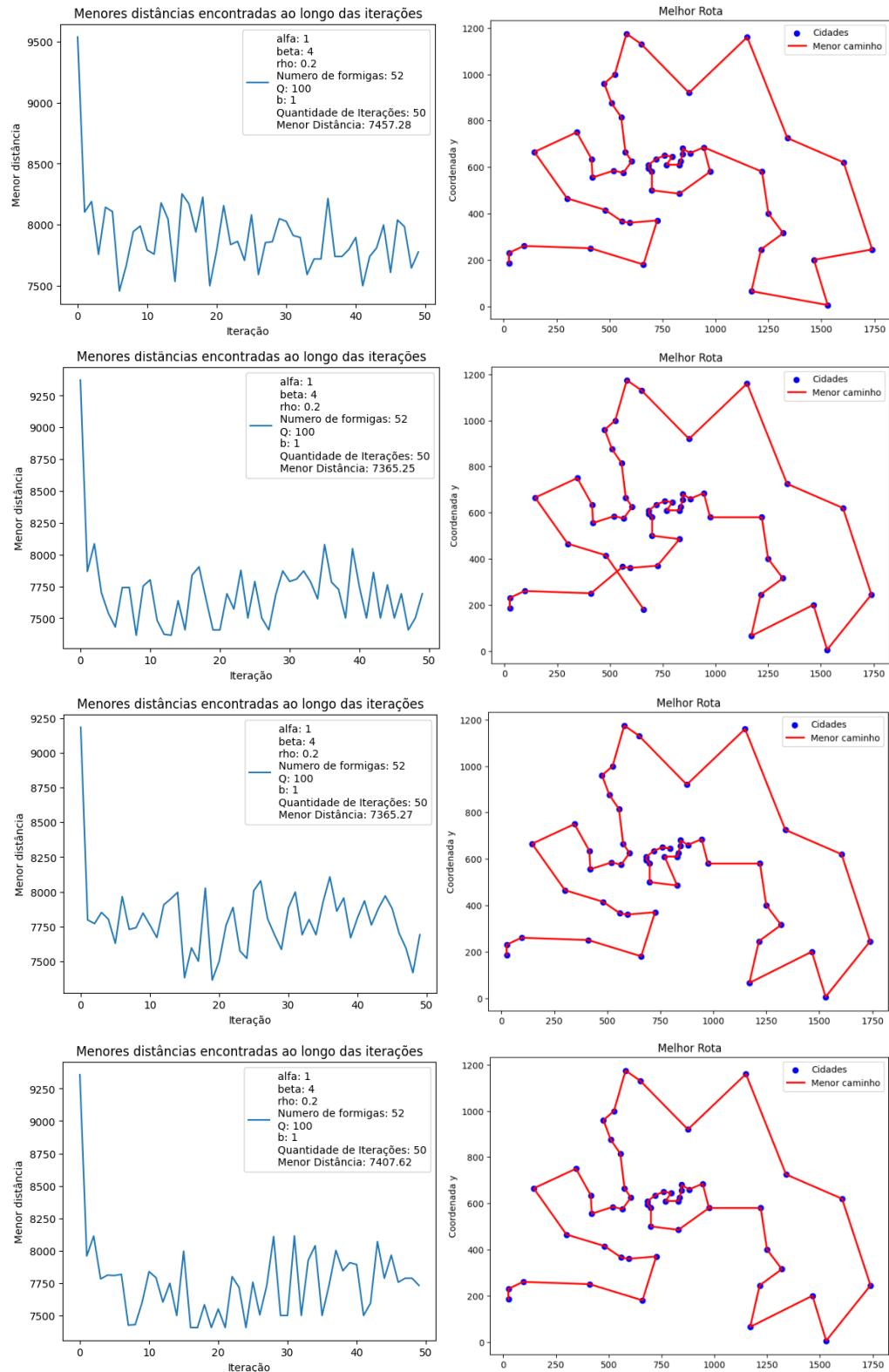
*Fonte: Elaborado pelo autor (2023).*

Considerando as informações apresentadas, tem-se a obtenção de resultados próximos. Pelo fato de as diferenças não serem significativamente expressivas, considerando o baixo número de execuções do algoritmo, optou-se por adotar um Q igual a 100. Esse resultado tende a balancear gasto computacional com resultados otimizados de maneira satisfatória.

#### **2.4.5 Influência da variável b no processo de otimização do problema TSP utilizando ACO**

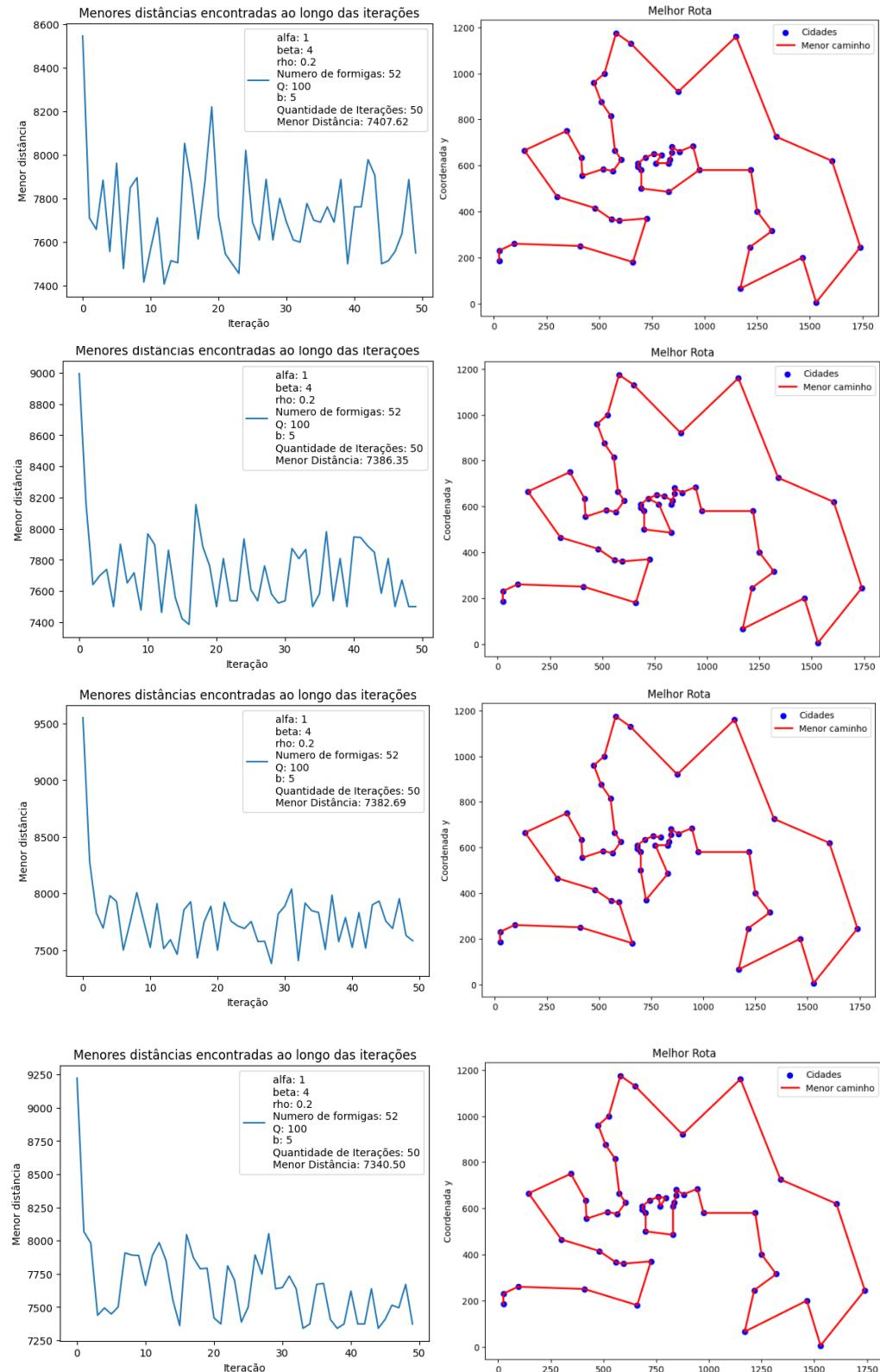
Nessa etapa, o objetivo é ajustar o hiperparâmetro que se relaciona às formigas elitistas. Para isso, realizaram-se 4 execuções considerando, respectivamente, 1, 5 e 10 formigas elitistas.

Figura 44: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $b$  de 1.



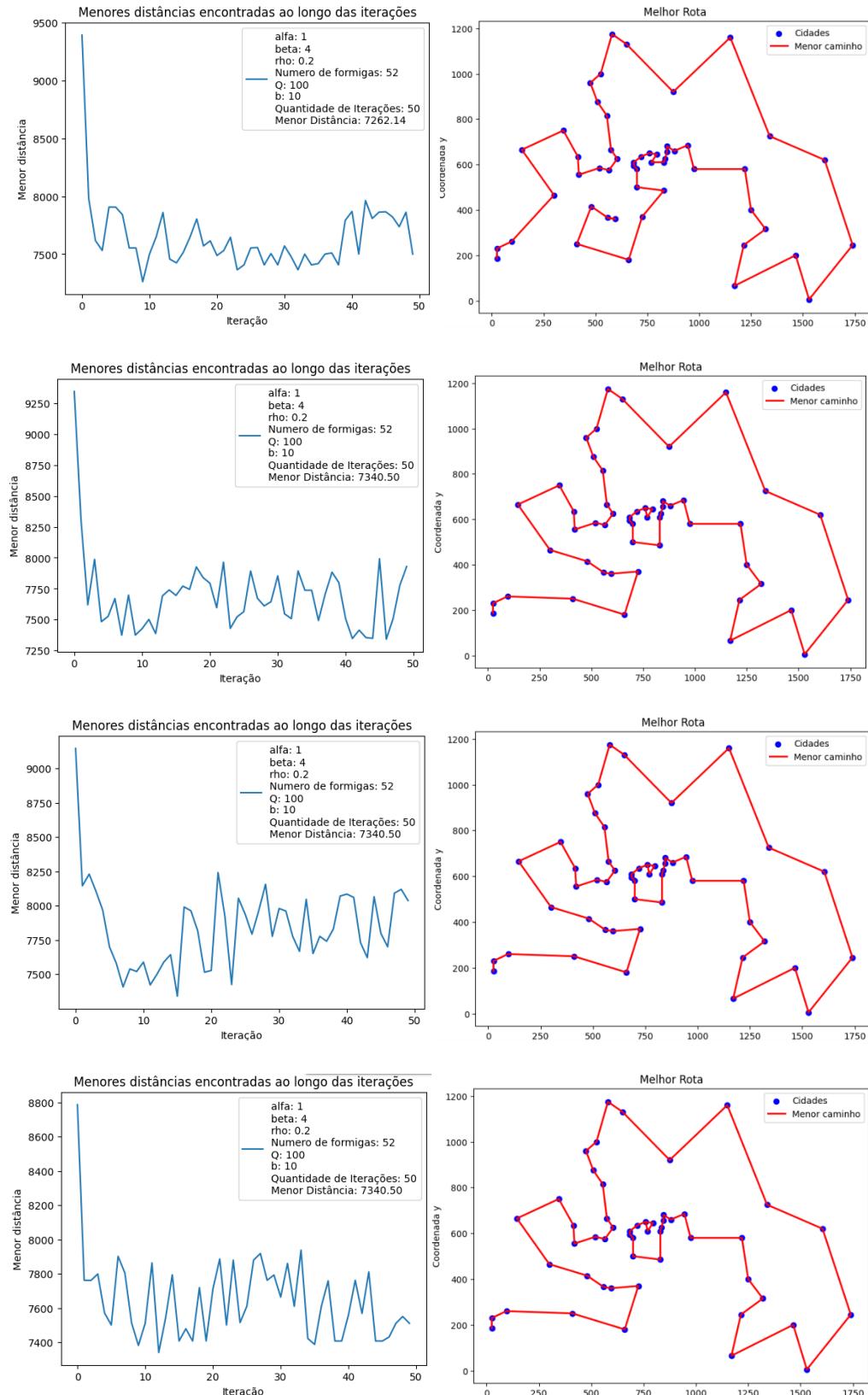
Fonte: Elaborado pelo autor (2023).

Figura 45: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $b$  de 5.



Fonte: Elaborado pelo autor (2023).

Figura 46: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP considerando um  $b$  de 10.



Fonte: Elaborado pelo autor (2023).

Com base nas Figuras 44, 45 e 46, constrói-se a Tabela 11.

*Tabela 11: Análise da resposta do algoritmo ACO na resolução do TSP considerando variações de b.*

<b>Média de Distâncias Mínimas</b>		
<b>b</b>		
<b>1</b>	<b>5</b>	<b>10</b>
7398.855	7379.29	7320.91

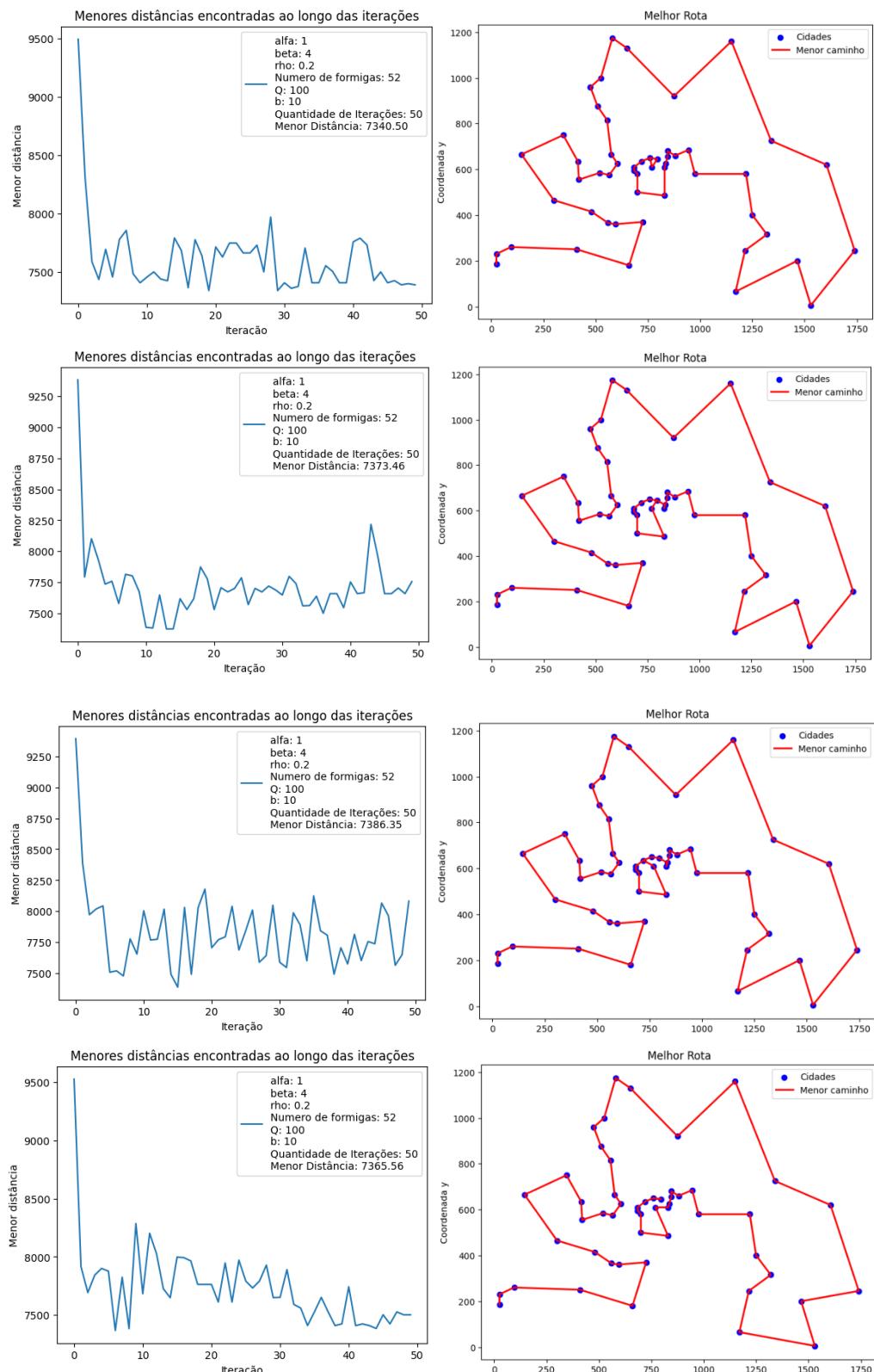
*Fonte: Elaborado pelo autor (2023).*

Analizando a Tabela 11 bem como as figuras supracitadas, conclui-se que quando o b assume o valor de 10, os resultados obtidos são extremamente satisfatórios. Nesse cenário, adotou-se tal valor como hiperparâmetro ideal para resolução do problema proposto.

#### **2.4.6 Análise final do ACO**

A título de entender o comportamento final do algoritmo com os hiperparâmetros devidamente ajustados com base nos experimentos apresentados acima, executou-se o código 4 vezes.

Figura 47: Gráficos para analisar a resposta do algoritmo ACO na resolução do TSP com os hiperparâmetros ajustados.



Fonte: Elaborado pelo autor (2023).

A análise final permitiu atingir uma média de valores mínimos de 7366,47. Tem-se tal valor como um resultado extremamente satisfatório, podendo considerá-lo como um ótimo global, confirmando, portanto, o correto ajuste dos hiperparâmetros.

### **3. Link para acesso dos algoritmos**

Nesta seção, encontram-se os links de acesso aos algoritmos que solucionam os exercícios propostos.

#### **3.1 Link de acesso ao exercício 1**

<https://colab.research.google.com/drive/12kOv4EeUBODp2g5FlXK-BnjQqpknBACI?usp=sharing>

#### **3.2 Link de acesso ao exercício 2**

<https://colab.research.google.com/drive/1IMFMYSikGgJ57g7EYbtuv63eyDd1t2SbO?usp=sharing>

#### 4. Referências bibliográficas

- [1] PYTHON SOFTWARE FOUNDATION. Python. Disponível em: <https://www.python.org/>.
- [2] GOOGLE. Colaboratory. Disponível em: <https://colab.research.google.com/>.
- [3] PYTHON SOFTWARE FOUNDATION. Random — Gerador de números pseudo-aleatórios. Python 3.9.7 documentation. Disponível em:  
<https://docs.python.org/3/library/random.html>.
- [4] MATPLOTLIB. Matplotlib: Visualization with Python: Disponível em: <https://matplotlib.org/>.
- [5] BREVE, Fabricio. Particle Swarm Optimization (PSO). 03 de maio de 2023. Apresentação de slides. UNESP.
- [6] PANDAS. Disponível em: <https://pandas.pydata.org/>.
- [7] PYTHON SOFTWARE FOUNDATION. IO – Core tools for working with streams. Disponível em: <https://docs.python.org/3/library/io.html>
- [8] PYTHON SOFTWARE FOUNDATION. Math - Funções matemáticas. Python 3.9.7 documentation. Disponível em: <https://docs.python.org/3/library/math.html>.
- [9] PYTHON SOFTWARE FOUNDATION. Itertools – Funções que criam iteradores para laços eficientes. Disponível em: <https://docs.python.org/pt-br/3.12/library/itertools.html>
- [10] SCIPY. SciPy - Fundamental algorithms for scientific computing in Python. Disponível em: <https://scipy.org/>
- [11] NUMPY. NumPy. San Francisco: NumPy, 2023. Disponível em: <https://numpy.org/.7>
- [12] BREVE, Fabricio. Ant Colony Optimization. 18 de maio de 2023. Apresentação de slides. UNESP.

\*O presente relatório foi embasado integralmente e desenvolvido com o suporte das notas de aula [5] e [12].