Implementation of Simulated Annealing Algorithm

```
import mlrose_hiive as mlrose

import numpy as np


def queens_max(position):

queennotattacking = 0

for i in range(len(position) - 1):

noattack = 0

for j in range(i + 1, len(position)):

if (position[j] != position[i]) and (position[j] != position[i] + (j - i)) and
(position[j] != position[i] - (j - i)):

noattack += 1

if noattack == len(position) - 1 - i:

queennotattacking += 1

return queennotattacking


# Take user input for the initial position

try:

user_input = input("Enter the initial position as 8 comma-separated integers
(e.g., '4,6,1,5,2,0,3,7'): ")

initialpos = np.array([int(x) for x in user_input.split(',')])

if len(initialpos) != 8 or any(x < 0 or x >= 8 for x in initialpos):

raise ValueError("Please enter exactly 8 integers between 0 and 7.")

except ValueError as e:

print(e)
```

```
exit()


# Define the problem and schedule

objective = mlrose.CustomFitness(queens_max)

problem = mlrose.DiscreteOpt(length=8, fitness_fn=objective, maximize=True,
max_val=8)

T = mlrose.ExpDecay()


# Run the simulated annealing algorithm

result = mlrose.simulated_annealing(problem=problem, schedule=T,
max_attempts=500, max_iters=5000, init_state=initialpos)


# Access the best state and best fitness from the result

best_state = result[0]  # Best state

best_fitness = result[1]  # Best fitness


print('The best position found is:', best_state)

print('The number of queens that are not attacking each other is:',
best_fitness+1)


# Print the diagram of the best state

print("\nBest State Diagram:")

board = [['.' for _ in range(8)] for _ in range(8)]

for row, col in enumerate(best_state):

board[col][row] = 'Q'  # Place queen
```

```
# Print the board
for row in board:
print(' '.join(row))
```

Output:

Enter the initial position as 8 comma-separated integers (e.g., '4,6,1,5,2,0,3,7'): 0,1,2,3,4,5,6,7

The best position found is: [5 3 1 7 4 6 0 2]

The number of queens that are not attacking each other is: 8.0

Best State Diagram:

. . . . . Q .

. . Q . . . . .

. . . . . Q

. Q . . . . . .

. . . . Q . . .

Q . . . . . .

. . . . . Q . .

. . . Q . . . .