```python
import mlrose_hiive as mlrose
import numpy as np

def hanoi_fitness(state):
    correct_disks = 0
    destination_peg = 2
    for i in range(len(state)):
        if state[i] == destination_peg:
            correct_disks += 1
        else:
            break
    return correct_disks

fitness_fn = mlrose.CustomFitness(hanoi_fitness)
problem = mlrose.DiscreteOpt(length=3, fitness_fn=fitness_fn, maximize=True, max_val=3)
schedule = mlrose.ExpDecay()

initial_state = np.array([0, 0, 0])
best_state, best_fitness, fitness_curve = mlrose.simulated_annealing(problem, schedule=schedule, max_attempts=1000, init_state=initial_state)

print("Best state (final configuration):", best_state)
print("Number of correct disks on destination peg:", best_fitness)

def print_hanoi_solution(state):
    print("\nTower of Hanoi Configuration:")
    pegs = {0: [], 1: [], 2: []}
    for disk, peg in enumerate(state):
        pegs[peg].append(disk)
    for peg in pegs:
        print(f"Peg {peg}: {pegs[peg]}")/.,mnbvx

print_hanoi_solution(best_state)
```

```
Best state (final configuration): [2 2 2]
Number of correct disks on destination peg: 3.0

Tower of Hanoi Configuration:
Peg 0: []
Peg 1: []
Peg 2: [0, 1, 2]
```

```
import mlrose_hiive as mlrose
import numpy as np

def hanoi_fitness(state):
    correct_disks = 0
    destination_peg = 2
    for i in range(len(state)):
        if state[i] == destination_peg:
            correct_disks += 1
        else:
            break
    return correct_disks

fitness_fn = mlrose.CustomFitness(hanoi_fitness)
problem = mlrose.DiscreteOpt(length=3, fitness_fn=fitness_fn, maximize=True, max_val=3)
schedule = mlrose.ExpDecay()

initial_state = np.array([0, 0, 0])
best_state, best_fitness, fitness_curve = mlrose.simulated_annealing(problem, schedule=schedule, max_attempts=1000, init_state=initial_st

print("Best state (final configuration):", best_state)
print("Number of correct disks on destination peg:", best_fitness)

def print_hanoi_solution(state):
    print("\nTower of Hanoi Configuration:")
    pegs = {0: [], 1: [], 2: []}
    for disk, peg in enumerate(state):
        pegs[peg].append(disk)
    for peg in pegs:
        print(f"Peg {peg}: {pegs[peg]}")/.,mnbvx

print_hanoi_solution(best_state)
```

```
Best state (final configuration): [2 2 2]
Number of correct disks on destination peg: 3.0

Tower of Hanoi Configuration:
Peg 0: []
Peg 1: []
Peg 2: [0, 1, 2]
```