```python
import numpy as np

# Step 1: Parameters
N = 10  # Population size (number of wolves)
T = 100  # Number of iterations
num_intersections = 3  # Number of traffic signal intersections
max_green_time = 60  # Max green light time in seconds
min_green_time = 30  # Min green light time in seconds

# Initialize the positions of alpha, beta, delta wolves (for fitness comparison)
alpha = np.inf  # Initial best fitness value (alpha)
beta = np.inf   # Second best (beta)
delta = np.inf  # Third best (delta)

# Fitness function: minimize total green light time for simplicity
def evaluate_fitness(wolf):
    # Here we assume the fitness is just the sum of green light times
    return np.sum(wolf)

# Initialize population with random green light times for each intersection
def initialize_population(N, num_intersections):
    return np.random.randint(min_green_time, max_green_time, size=(N, num_intersections))

# Update the wolf's position based on GWO strategy
def update_position(wolf, alpha_wolf, beta_wolf, delta_wolf):
    C1, C2, C3 = np.random.rand(3) * 2  # Random numbers for exploration
    A1, A2, A3 = np.random.rand(3)  # Random numbers for exploitation

    # Calculate the distance from the wolves
    D_alpha = np.abs(C1 * alpha_wolf - wolf)
    D_beta = np.abs(C2 * beta_wolf - wolf)
    D_delta = np.abs(C3 * delta_wolf - wolf)

    # Update position using the Grey Wolf Optimizer formula
    wolf_new = wolf + A1 * D_alpha + A2 * D_beta + A3 * D_delta

    # Ensure new wolf position is within valid range [min_green_time, max_green_time]
    return np.clip(wolf_new, min_green_time, max_green_time)

# Main loop for GWO algorithm
def gwo_traffic_signal_optimization(N, T, num_intersections):
    # Step 1: Initialize population (random green light times for each intersection)
    population = initialize_population(N, num_intersections)

    global alpha, beta, delta
    for t in range(T):
        # Step 2: Evaluate fitness for each wolf (traffic signal configuration)
        fitness_values = np.array([evaluate_fitness(population[i]) for i in range(N)])

        # Step 3: Update alpha, beta, and delta wolves based on fitness values
        sorted_indices = np.argsort(fitness_values)
        alpha = fitness_values[sorted_indices[0]]
        beta = fitness_values[sorted_indices[1]]
        delta = fitness_values[sorted_indices[2]]

        alpha_wolf = population[sorted_indices[0]]
        beta_wolf = population[sorted_indices[1]]
        delta_wolf = population[sorted_indices[2]]

        # Step 4: Update the positions of all wolves
        for i in range(N):
            population[i] = update_position(population[i], alpha_wolf, beta_wolf, delta_wolf)

    # Step 5: Return the best solution found
    return alpha_wolf, alpha

# Run the optimization
best_configuration, best_fitness = gwo_traffic_signal_optimization(N, T, num_intersections)

# Output the best configuration (green light times) and the fitness value
print("Best Traffic Signal Timings (Green Light Times for Each Intersection):")
print(best_configuration)
print("Total Waiting Time (Fitness Value):", best_fitness)
```

```
Best Traffic Signal Timings (Green Light Times for Each Intersection):
[60 60 60]
Total Waiting Time (Fitness Value): 180
```