```
import random

def greedy_subset_sum(set, k):
    if not set:
        print("Error: set is empty")
        return
    if sum(set) % k != 0:
        print("Error: sum of set is not divisible by number of subsets")
        return
    # calculate the target sum value of subsets
    target_sum = sum(set) // k
    # sort the set in descending order
    set = sorted(set, reverse=True)
    subsets = [[] for _ in range(k)]
    # initialize current sum of each subset
    subset_sum = [0 for _ in range(k)]
    # fill the subsets
    for i in range(len(set)):
        # find the subset with the smallest current sum
        min_subset = min(range(k), key=lambda x: subset_sum[x])
        # add the element to the subset
        subsets[min_subset].append(set[i])
        subset_sum[min_subset] += set[i]
    return subsets

def best_fit_subset_sum(set, k):
    if not set:
        print("Error: set is empty")
        return
    if sum(set) % k != 0:
        print("Error: sum of set is not divisible by number of subsets")
        return
    # calculate the target sum value of subsets
    target_sum = sum(set) // k
    # sort the set in descending order
    set = sorted(set, reverse=True)
    subsets = [[] for _ in range(k)]
    # initialize current sum of each subset
    subset_sum = [0 for _ in range(k)]
    # fill the subsets using best fit heuristic
    for i in range(len(set)):
        # find the subset with the closest current sum to the target sum
        closest_subset = min(range(k), key=lambda x: abs(subset_sum[x] - target_sum))
        # add the element to the subset
        subsets[closest_subset].append(set[i])
        subset_sum[closest_subset] += set[i]
    return subsets

def best_fit_hyperheuristic(set, k):
    if not set:
        print("Error: set is empty")
        return
    if sum(set) % k != 0:
        print("Error: sum of set is not divisible by number of subsets")
        return
    target_sum = sum(set) // k
    # Try both greedy and best fit heuristics
    greedy_subsets = greedy_subset_sum(set, k)
    greedy_deviation = sum([abs(target_sum - sum(subset)) for subset in greedy_subsets])
    best_fit_subsets = best_fit_subset_sum(set, k)
    best_fit_deviation = sum([abs(target_sum - sum(subset)) for subset in best_fit_subsets])
    # Select the best performing heuristic
    if greedy_deviation < best_fit_deviation:
        return greedy_subsets
    else:
        return best_fit_subsets

# example usage
set = [1, 2, 4, 11, 14, 18, 22, 29, 33, 37, 45, 47, 52, 53, 77, 82, 87, 92, 95, 99]
k = 4
print(best_fit_hyperheuristic(set, k))
```