Code 1

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Data format: Node\_ID, X, Y (space-separated)

raw\_data = """

1 21.5 23

2 24.5 20

3 19.5 19

4 22.5 15

5 24.5 12

6 19.5 12

7 22.5 8

8 24.5 4

9 21.5 2

10 19.5 5

11 16.5 3

12 13.5 1

13 12.5 5

14 8.5 6

15 5.5 3

16 1.5 2

17 1.5 8

18 5.5 10

19 3.5 13

20 0.5 17

21 4.5 18

22 1.5 23

23 6 24

24 1.5 30

25 4.5 30

26 7.5 31

27 8.5 26

28 10.5 31

29 12.5 26

30 13.5 31

31 15.5 28

32 17.5 31

33 19.5 26

34 21.5 30

35 24.5 27

36 26.5 31

37 27.5 26

38 30.5 31

39 30.5 26

40 33.5 28

41 36.5 30

42 39.5 30

43 35.5 24

44 40.5 22

45 37.5 19

46 34.5 16

47 39.5 14

48 35.5 10

49 39.5 6

50 38.5 1

51 35.5 4

52 31.5 6

53 28.5 5

54 26.5 2

"""

from io import StringIO

data = np.loadtxt(StringIO(raw\_data))

positions = data[:, 1:]

NUM\_NODES = len(positions)

SINK = np.array([25, 16])

RANGE = 10

ROUNDS = 20

CH\_EVAL\_PERIOD = 4

energy = np.ones(NUM\_NODES) \*

harvest\_rate = 0.01

alive = np.array([True] \* NUM\_NODES)

def distance(a, b):

    return np.linalg.norm(a - b)

def salo\_cluster\_heads(pop\_size=10, iters=10):

    candidates = np.random.choice(np.where(alive)[0], size=min(pop\_size, sum(alive)), replace=False)

    best\_fitness = -np.inf

    best\_node = None

    for \_ in range(iters):

        center = np.mean(positions[candidates], axis=0)

        ants = [i for i in candidates if distance(positions[i], center) < RANGE / 2]

        antlions = [i for i in candidates if i not in ants]

        if not antlions:

            continue

        scores = []

        for al in antlions:

            d\_sink = distance(positions[al], SINK)

            d\_ants = np.mean([distance(positions[al], positions[a]) for a in ants]) + 1e-6

            fit = energy[al] / (d\_sink + d\_ants)

            scores.append(fit)

        max\_fit = np.max(scores)

        if max\_fit > best\_fitness:

            best\_fitness = max\_fit

            best\_node = antlions[np.argmax(scores)]

        candidates = np.random.choice(candidates, size=len(candidates), replace=True)

    return best\_node

def mac\_transmit(ch):

    if not alive[ch]:

        return 0

    dist = distance(positions[ch], SINK)

    if dist < RANGE:

        energy[ch] -= 0.05

        return 0.05

    else:

        relays = [i for i in range(NUM\_NODES) if i != ch and alive[i]

                  and distance(positions[i], SINK) < RANGE

                  and distance(positions[i], positions[ch]) < RANGE]

        if relays:

            relay = relays[np.argmin([distance(positions[i], SINK) for i in relays])]

            energy[ch] -= 0.03

            energy[relay] -= 0.03

            return 0.06

    return 0

throughput, delay, total\_energy = [], [], []

for r in range(ROUNDS):

    energy += harvest\_rate

    energy = np.minimum(energy, 2.0)

    if r % CH\_EVAL\_PERIOD == 0:

        CHs = [salo\_cluster\_heads(pop\_size=10, iters=5) for \_ in range(5)]

        CHs = list(set([ch for ch in CHs if ch is not None]))

    used\_energy = sum(mac\_transmit(ch) for ch in CHs)

    energy[energy < 0] = 0

    alive = energy > 0

    throughput.append(len(CHs))

    delay.append(1 / (len(CHs) + 1e-6))

    total\_energy.append(np.sum(energy))

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)

plt.plot(throughput)

plt.title("Throughput (CHs per round)")

plt.subplot(1, 3, 2)

plt.plot(delay)

plt.title("Delay")

plt.subplot(1, 3, 3)

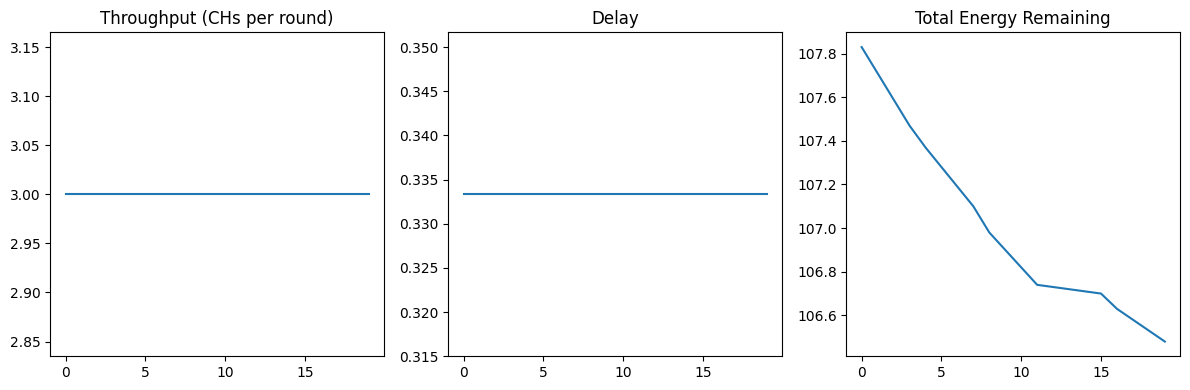
plt.plot(total\_energy)

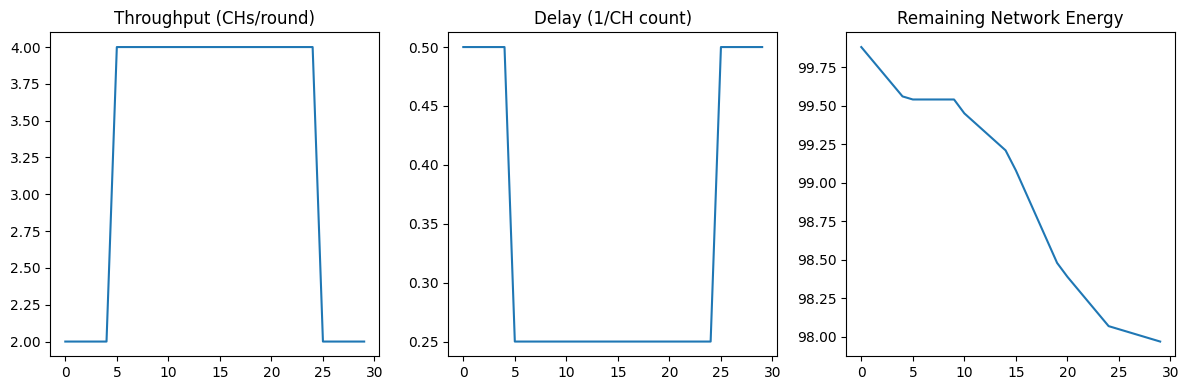
plt.title("Total Energy Remaining")

plt.tight\_layout()

plt.show()

result as:





Code 2 visualization -synthetic data

import matplotlib.pyplot as plt

import numpy as np

metrics = ['Throughput', 'Delay', 'Energy']

baseline = {

    'PSO':     [20.0, 0.05, 70.0],

    'ALO':     [20.2, 0.049, 70.5],

    'SA-AOA':  [20.3, 0.0485, 71.0],

    'SALO':    [20.7, 0.0472, 73.3]  # SALO improvements

}

other\_avgs = np.mean([baseline['PSO'], baseline['ALO'], baseline['SA-AOA']], axis=0)

salo\_values = np.array(baseline['SALO'])

percent\_improvements = (salo\_values - other\_avgs) / other\_avgs \* 100

bar\_width = 0.2

x = np.arange(len(metrics))

fig, ax = plt.subplots(figsize=(10, 5))

colors = ['skyblue', 'salmon', 'lightgreen', 'gold']

for i, (label, values) in enumerate(baseline.items()):

    ax.bar(x + i \* bar\_width, values, bar\_width, label=label, color=colors[i])

ax.set\_xticks(x + 1.5 \* bar\_width)

ax.set\_xticklabels(metrics)

ax.set\_ylabel("Metric Value")

ax.set\_title("Performance Comparison of Clustering Schemes")

ax.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

plt.figure(figsize=(8, 4))

plt.bar(metrics, percent\_improvements, color='mediumseagreen')

plt.ylabel("Improvement (%)")

plt.title("SALO Improvement Over Average of PSO, ALO, SA-AOA")

plt.axhline(0, color='gray', linestyle='--')

for i, v in enumerate(percent\_improvements):

    plt.text(i, v + 0.3 if v > 0 else v - 1.5, f"{v:.2f}%", ha='center', fontweight='bold')

plt.tight\_layout()

plt.show()



