Consider a network topology defined by an undirected graph G with adjacency matrix A and number of agents n.

Note that $A_i \in \mathbb{R}^{m \times d_i}$, $x_i \in \mathbb{R}^{d_i}$ and $b_i \in \mathbb{R}^m$.

$$\sum_{i=1}^{n} (A_i x_i - b_i) = 0$$

Problem:

$$\sum_{i=1}^{n} b_i = b = \sum_{i=1}^{n} \tilde{b}_i$$

$$\tilde{b}_i \in \mathrm{Im} A_i \quad \forall i = 1, \ldots, n$$

Comments:

- There's no point in checking in advance whether the vectors lie in images, because it's still $\mathcal{O}(n^3)$
- We start with a random agent but it does not affect the outcome of the method
- We perform a DFS through the graph G
- Number of communications is $\mathcal{O}(n)$, which follows from DFS

Algorithm:

1. select agent i_1 uniformly from $\{1, \ldots, n\}$
2. solve linear system $A_{i_1}\alpha_{i_1} = b$ using least squares, that is we obtain projection of vector b onto the column space of matrix A_{i_1} as $b_\| = A_{i_1}\alpha_{i_1} = A_{i_1}(A_{i_1}^\top A_{i_1})^{-1} A_{i_1}^\top b$
3. check if the residual $b_\perp = b - b_\|$ is zero
   - if $b_\perp = 0$ then $\tilde{b}_{i_1} = b_\|$ and other $\tilde{b}_j = 0$
   - else $\tilde{b}_{i_1} = b_\|$, select agent i_2 uniformly from the still unused neighbors of i_1 and repeat steps 2-3 with $A_{i_2}\alpha_{i_2} = b_\perp$
4. (the last agent is considered) check if the residual $b_\perp$ is zero
   - if $b_\perp = 0$ then $\tilde{b}_{i_n} = b_\|$ and we got desired $\tilde{b}_1, \ldots, \tilde{b}_n$
   - else it turned out that b does not decompose into bases of the considered spaces and the problem has no solution

---

In [ ]:

```
import numpy as np
import networkx as nx
```

In [ ]:

```python
def DFS(graph, start, visited=None, visited_list=None):
    if visited is None:
        visited = set()
    visited.add(start)

    if visited_list is None:
        visited_list = list()
    if start not in visited_list:
        visited_list.append(start)

    #print(visited_list)

    for next in set(graph[start]) - visited:
        DFS(graph, next, visited, visited_list)

    return visited_list
```

In [ ]:

```python
def allocation(G, A_list, b_list):
    n = G.number_of_nodes()
    m = b_list.shape[1]

    b = b_list.sum(axis=0)
    b_copy = b.copy()
    b_tilde_list = np.zeros((n, m))

    start = np.random.randint(n)
    visited_list = DFS(G, start)

    print(visited_list)

    for agent in visited_list:
        #b_parallel = A_list[agent] @ np.linalg.inv((A_list[agent].T @ A_list[agent])) @ A_list[agent].
        b_parallel = A_list[agent] @ np.linalg.pinv(A_list[agent]) @ b_copy
        b_perp = b_copy - b_parallel
        b_tilde_list[agent] = b_parallel
        if np.linalg.norm(b_perp) < 1e-9:
            break
        else:
            b_copy = b_perp

    if not np.linalg.norm(b_perp) < 1e-9:
        print("Нет решения")
        return None

    return b_tilde_list
```

Test 1 (solution exists):

- Network topology: K_3
- $n = 3$
- $m = 3$
- $d_1 = 3$
- $d_2 = 2$
- $d_3 = 4$
- $A_1 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, $b_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$
- $A_2 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$, $b_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$
- $A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, $b_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

In [ ]:

```python
A = np.array(
    [[0, 1, 1],
     [1, 0, 1],
     [1, 1, 0]]
)

G = nx.from_numpy_array(A)

n = A.shape[0]
m = 3
d = np.array([3, 2, 4])

A1 = np.array([[0, 0, 0],
               [1, 0, 0],
               [0, 1, 0]])
A2 = np.array([[1, 0],
               [0, 0],
               [0, 1]])
A3 = np.array([[1, 0, 0, 0],
               [0, 1, 0, 0],
               [0, 0, 0, 0]])

A_list = [A1, A2, A3]

b1 = np.array([1, 0, 0])
b2 = np.array([0, 1, 0])
b3 = np.array([0, 0, 1])

b_list = np.array([b1, b2, b3])
```

In [ ]:

```
allocation(G, A_list, b_list).T
```

```
[1, 0, 2]
```

Out[ ]:

```
array([[0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.]])
```

Test 2 (solution doesn't exist):

- Network topology: K_3
- $n = 3$
- $m = 3$
- $d_1 = 3$
- $d_2 = 2$
- $d_3 = 4$
- $A_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, b_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$
- $A_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, b_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$
- $A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, b_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

In [ ]:

```
A = np.array(
    [[0, 1, 1],
     [1, 0, 1],
     [1, 1, 0]]
)

G = nx.from_numpy_array(A)

n = A.shape[0]
m = 3
d = np.array([3, 2, 4])

A1 = np.array([[1, 0, 0],
               [0, 1, 0],
               [0, 0, 0]])
A2 = np.array([[1, 0],
               [0, 1],
               [0, 0]])
A3 = np.array([[1, 0, 0, 0],
               [0, 1, 0, 0],
               [0, 0, 0, 0]])

A_list = [A1, A2, A3]

b1 = np.array([1, 0, 0])
b2 = np.array([0, 1, 0])
b3 = np.array([0, 0, 1])

b_list = np.array([b1, b2, b3])
```

In [ ]:

```
allocation(G, A_list, b_list)
```

```
[0, 1, 2]
Нет решения
```

Test 3 (randomly generated for n=20):

In [ ]:

```
A = np.array(
    [[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],
     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
     [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0],
     [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
     [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
     [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
     [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
     [0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]]
)

G = nx.from_numpy_array(A)

n = A.shape[0]
m = 10
d = np.random.randint(low=10, high=20, size=n)

A_list = [np.random.randn(m, d[i]) for i in range(n)]

b_list = np.array([np.random.randn(m) for _ in range(n)])
```

In [ ]:

```
b_list.sum(axis=0)
```

Out[ ]:

```
array([ 7.68718015,  2.39913584,  2.21624284,  2.40902287,  0.59452642,
       -0.61298694, -9.2643805 ,  6.96188932, -1.30928941,  3.90189225])
```

In [ ]:

```
allocation(G, A_list, b_list).sum(axis=0)
```

Out[ ]:

```
[7, 2, 10, 19, 14, 12, 16, 3, 18, 6, 8, 5, 11, 4, 0, 1, 13, 9, 17, 15]
array([ 7.68718015,  2.39913584,  2.21624284,  2.40902287,  0.59452642,
       -0.61298694, -9.2643805 ,  6.96188932, -1.30928941,  3.90189225])
```