Decentralized System Implementation (Using MATLAB)

1. Algorithm Description

Centralized SOMP

Inputs: $\{y_l,\ B_l\}_{l=0}^{L-1}$**, and sparsity level** k
(1) Initialize $t = 1,\ \hat{u}(0) = \emptyset$**, residual vector** $r_{l,0} = y_l$
(2) Find the index $\lambda(t)$ **such that**

$$\lambda(t) = arg\max_{\omega} \sum_{l=1}^{L} \left|\langle r_{l,t-1},\ b_{l,\omega}\rangle\right|$$

(3) Set $\hat{u}(t) = \hat{u}(t-1) \cup \{\lambda(t)\}$
(4) Compute the orthogonal projection operator:

$$P_l(t) = B_l(\hat{u}(t)) \left[B_l(\hat{u}(t))^H B_l(\hat{u}(t))\right]^{-1} B_l(\hat{u}(t))^H$$

Update the residual: $r_{l,t} = [I - P_l(t)]y_l$
(5) Increment $t = t + 1$ **and go to step (2) if** $t \le k$**, otherwise,**
stop and set $\hat{u} = \hat{u}(t-1)$

We would like to modify the previous centralized SOMP method to Decentralized DC-OMP-TA Algorithm

DC-OMP-TA Algorithm as the following

At Sensor l
(1) Compute scores $f_{l,\omega}(t) = \left|\langle r_{l,t-1},\ b_{l,\omega}\rangle\right|$ **for** $\omega = 1,\ 2,\ \cdots N_G$**, and sort** $f_{l,\omega}(t)$ **in descending order, and denote the sorted scores as** $g_{l,i}$ **and their corresponding index as** $\omega_l(i)$
(2) Set $i = 1,$ **index set** $v_{0=}\emptyset,$ **sum score array** $s = \emptyset$
(3) Communication
Sensor l **send** $g_{l,i}$ **and** $\omega_l(i)$ **to its neighbors** $g\backslash\{l\}$
Sensor l **receive** $g_{m,i}$ **and** $\omega_m(i)$ **from its neighbors** $m \in g\backslash\{l\}$
(4) Update index set $v_{i=}v_{i-1} \cup \{\omega_1(i),\ \omega_2(i),\ \cdots,\ \omega_L(i)\}$
(5) Compute the threshold $\eta_i = \sum_{l=1}^{L} g_{l,i}$
(6) For any newly appearing index $\omega \in \overline{v_{i-1}} \cap \quad \{\omega_1(i),\ \omega_2(i),\ \cdots,\ \omega_L(i)\}$
Transmit $f_{l,\omega}$ **to** $g\backslash\{l\}$
Receive $f_{m,\omega}$ **from** $g\backslash\{l\}$
Update sum score array $s = [s\ \sum_{l=1}^{L} f_{l,\omega}]$
(7) If the cardinality $|s \ge \eta_i|$ **≥1, return the index with largest sum score, stop; otherwise,** $i = i + 1$**, go to step (3)**

For TA Algorithm

- **First, each sensor sorts its local $f_{l,\omega}$ values in descending order.**
- **Then TA goes down the sorted lists in parallel, one position at a time, and calculates the sum of the values at that position across all the lists.**
- **This sum is called "threshold'',.**
- **Every time a new state grid index (object) appears, TA looks up in all the lists to find its sum value (aggregate value) over the lists.**
- **The threshold sets an upper bound on the aggregate values of all the newly appearing objects at the current iteration**
- **TA stops when it finds k objects (indices) whose sum values are higher than the current threshold.**
- **In the worst case scenario, the communication cost per iteration is $O(L^2)$**

An example for TA Algorithm is shown below:

Position	Series 1	Series 2	Series 3
1	$\langle O_1, 10\rangle$	$\langle O_2, 10\rangle$	$\langle O_3, 10\rangle$
2	$\langle O_3, 8\rangle$	$\langle O_4, 9\rangle$	$\langle O_1, 9\rangle$
3	$\langle O_5, 8\rangle$	$\langle O_6, 8\rangle$	$\langle O_7, 8\rangle$
4	$\langle O_6, 8\rangle$	$\langle O_8, 6\rangle$	$\langle O_9, 7\rangle$
5	$\langle O_2, 7\rangle$	$\langle O_7, 5\rangle$	$\langle O_6, 6\rangle$
6	$\langle O_4, 5\rangle$	$\langle O_3, 2\rangle$	$\langle O_4, 5\rangle$
7	$\langle O_9, 1\rangle$	$\langle O_1, 1\rangle$	$\langle O_2, 1\rangle$

- **TA first checks objects in Row 1 of all lists, which are O1, O2, and O3, and set the row sum as threshold (30)**
- **It then finds objects' sum values over series/lists (aggregate values), e.g. $V(O_1) = 10 + 1 + 9 = 20$, $V(O_2) = 10 + 1 + 7 = 18$, $V(O_3) = 10 + 8 + 2 = 20$, all less than threshold (30), TA moves to Row 2**
- **In Row 2, new threshold is 26, O_4 is new, $V(O_4) = 9 + 5 + 5 = 19$, all aggregate values are less than 26, TA moves to Row 3**
- **TA finally stops at Row 5 and finds the top 2 objects are O_6 with value 22 and O_1 with value 20**

So, right now, the whole TA algorithm has the global standpoint of view.

2. Task Description

A MATLAB implementation is required. Actually, we already have a MATLAB version of the "distributed" DC-OMP-TA. But the global-variable-based method on a single computer is not strictly distributed. We need the algorithm to run on at least three computers (such as i5, i7 CPUs), which are connected via a WIFI router.

[Reference]

1. Joint-Sparse Heterogeneous Data Fusion for Target State Estimation with Weak Signals, R Niu, P Zulch, M Distasio, G Chen, D Shen, J Lu, 2020 IEEE Aerospace Conference, 1-11
2. Joint-Sparse Decentralized Heterogeneous Data Fusion for Target Estimation, R Niu, P Zulch, M Distasio, G Chen, D Shen, Z Wang, J Lu, 2019 IEEE Aerospace Conference, 1-10
3. Joint Sparsity Based Heterogeneous Data-Level Fusion for Multi-Target Discovery, JL R. Niu, P. Zulch, M. Distasio, E. Blasch, G. Chen, D. Shen, Z. Wang, 2018 IEEE Aerospace Conference,
4. Wimalajeewa, Thakshila, and Pramod K. Varshney. "OMP based joint sparsity pattern recovery under communication constraints." IEEE Transactions on Signal Processing 62.19 (2014): 5059-5072.