PUCCH CSI

1 Business Requirement

The domain of Radio Access Network Layer 2 or MAC Layer is dynamically developed together with 5G/5.5G Multi-User MIMO technologies where the Base Station communication has, as a physical phenomenon, the channel Tx x Rx complex value matrix, such that Tx can be 64, 128, etc. and Rx can be 4,8, etc. The channel matrix needs to be periodically estimated by sending Sounding Reference Signals, SRS. In the same time, more traditional, Channel State Information, CSI, measurement is also needed. Both type of measurements take some timeslots and this reduces time for sending data in Shared Channel but timely measuring allows for an efficient frequency resource measurements; this tradeoff is a standard challenge of MU-MIMO. Also, the transmission goes both Downlink and Uplink, and both for Shared Channel (data) and Control Channel. In what follows, we address the problem of efficient support of SRS and CSI measurement in Uplink Control Channel, PUCCH.

2 SRS and CSI Resource allocation at TDD

Timeslots The first dimension of the problem is time. In the system we have L time slots**.** Each slot is a downlink or an uplink slot, according to time division duplex (TDD) pattern. As example we may consider L = 320 and TDD pattern (8:2). In this case slots 0-7, 10-17, 20-27,…, 310-317 are downlink slots, slots 8-9, 18-19, 28-29,…, 318-319 are uplink slots. The most popular TDD patterns are (9:1), (8:2), (7:3), (4:1), (3:2).

Time Resource Requirements CSI and SRS require resources as periodic subsets of time slots. They are given by two numbers: **period** and **offset**, such that offset < period. For example, if we have L = 320 time slots and SRS resource is (40, 3), then CSI resource includes the slots (3, 43, 83,…, 283). In what follows we consider periods (5, 10, 20, 40, 80, 160, 320). In general, these periods both are of the shape 2^i or 5×2^i for integer i. For example, the pair (srsPeriod, csiPeriod) may be equal to (10, 80), (40, 20), (4, 16), (32, 4), but cases (4, 40) and (5, 8) are not allowed. For CSI resources there is one more requirement: CSI resource is a subset of uplink slots.

3 Mutual Dependencies of SRS and CSI timeslots

Consider SRS resource allocations SRS($period_1$, $offset_1$) and CSI resource CSI($period_2$, $offset_2$). Define a **distance** between them, *dist(CSI,SRS)* as follows:

- If $period_1 \geq period_2$, then the distance is computed by formula: $dist(CSI, SRS) = |offset_2 - (offset_1 \bmod period_2)|$
- If $|period_1 < period_2|$ then first define $r = period_2/period_1$, and then $\text{dist(CSI, SRS)} = \min_{0 \leq i < r}(|\text{offset}_2 - ((\text{offset}_1 + i \times \text{period}_1) \, mod \, \text{period}_2)|)$.
- If we have CSI resource CSI and multiple SRS resources $SRS_1, SRS_2, \dots, SRS_n$ then the distance between CSI resource and the union of SRS resources is defined by formula:

$$dist(CSI, \{SRS_1, SRS_2, \dots, SRS_n\}) = min_{1 \leq i \leq n}(dist(CSI, SRS_i)).$$

4 Frequency Selection

Frequency is the second dimension of the problem. We have R different RBs (resource blocks) and each of these can be used for each of L time slots. CSI resource allocation also has a parameter, the number rb of RB required for allocation of this CSI resource. So, the allocation of CSI resource requires rb of RBs. For several CSI resources they may be allocated simultaneously if for each pair (slot, frequency) no more than 1 CSI occurs. For example, if no uplink slots are occupied by CSI resources and we allocate CSI resource ($period$, $offset$) with $rb = 1$ to RB#17, another CSI resource with the same period and offset may not be allocated to RB#17, but may be allocated to other RBs.

5 Problem input

- Common constants: L = 320 – number of slots, R = 58 – number of RBs, TDD pattern. May be (9:1, 8:2, 7:3, 4:1, 3:2).
- Bool IsOccupied[R][L] – array of already occupied pairs (rb, slot). Initially each value is “false”. Each allocated user should occupy only free positions. When some user is allocated this array also should be updated: set “true” for occupied positions.
- N – number of users to be scheduled. Input data about each user:
- Current status of IsOccupied[R][L],
- CSIperiodsNum ≤ 7 – number of available CSI periods,
- CSIperiods[] $\subset \{5, 10, 20, 40, 80, 160, 320\}$ – CSI periods, available for user,
- RequiredRbNumForCSI = 1,
- SRSperiod $\in \{5, 10, 20, 40, 80, 160, 320\}$ – SRS period of the user,

- SRSoffsets…$\{0,1,\dots,\text{SRSperiod}-1\}$, $|\text{SRSoffsets}| \leq 5$. – offsets of existing SRS resourses. So there are $|\text{SRSoffsets}|$ SRS resources (SRSperiod, SRSoffsets[0]),…, (SRSperiod, SRSoffsets[$|\text{SRSoffsets}|-1$]) which already exist for user (SRS_i = (SRSperiod, SRSoffsets[i])).

6 Output, Constraints, and Objective Function

- Per user *i* output is the triple (*rb[i], CSIperiodFinal[i], CSIoffsetFinal[i]),* where (*CSIperiodFinal[i], CSIoffsetFinal[i])=: CSIfinal[i]* are the chosen CSI resource for user *i*, and *rb[i]* is the chosen RB for allocation of this CSI resource. If a user may not be allocated, then the output should be (0,640,0).
- Constraints:
- $CSIperiodFinal[i] \in CSIperiods[]$
- $IsOccupied[rb, (CSIperiodFinal[i]*I + CSIoffsetFinal[i]) \bmod L]$ = false for all *I*
- $(CSIperiodFinal[i]*I + CSIoffsetFinal[i]) \bmod L$ should be an uplink slot for all *I*
- The objective function:
- For a user *i* define $distFinal[i] = dist(CSIfinal[i],\{SRS_0[i], SRS_1[i],\dots, SRS_{|SRSoffsets[i]|-1}[i]\})$. If user is not allocated, then $distFinal[i] = 640$.
- For user define $quality[i] = distFinal[i]+CSIperiodFinal[i]$,
- Define $rbUsed$ as a number of RB, for which at least one slot is occupied after allocation procedure for all users,
- **The objective is to minimize** the function $rbUsed \cdot (quality[1] + \dots + quality[N])$.

7 Baseline description

For one user baseline may be described by following pseudocode:

1.distFinal = 2*L

2.periodFinal = 2*L

3. offsetFinal = 0

4. rbFinal = 0

5. TDD = (DL:UL); // TDD pattern DL – number of downlink slot, UL – number of uplink

slots

6. For (period in CSIperiods[])

   For (rb in [0,R-1])

      For (offset in [0, period -1])

         bool success = true;

         For (slot = offset; slot < L; slot += period){

            if (IsOccupied[rb][slot]) then success = false;

            if (slot % (DL+UL) < DL) then success = false;   // check that slot is uplink slot

             if (success)

                currCSI = (period, offset)

                distCurr = dist(currCSI, $\{SRS_0, SRS_1, \ldots, SRS_{|SRSoffsets|-1}\}$)

                if (distCurr < distFinal)

                    (rbFinal, periodFinal, offsetFinal, distFinal) = (rb, period, offset, dist)

7. For (slot = offsetFinal; slot < L; slot += periodFinal)

   IsOccupied[rb][slot] =   true;

8. Return (rbFinal, periodFinal, offsetFinal)

For several users this algorithm should be executed sequentially for them in the order in which they are represented in the input.

8 Data Generation and Some Remarks

- Data generation:
  - CSIperiods[], may be considered as a segment from array $\{5, 10, 20, 40, 80, 160, 320\}$. Also may expect, that it looks like (period, period*2,period*4, …, 320). For majority of the users such arrays should start from 40 or 80.
  - SRSperiod also should be 40 or 80 for the majority of the users.
  - SRSoffsets[] may be generated as arbitrary numbers in the required range.
  - IsOccupied[R][L] – initially all values are “false”, updated during the algorithm

- Algorithm should be tested for cases, in which baseline occupies 10%, 50%, 90% of all uplink slots. Also consider testcase, when 10% of users are not successfully allocated by baseline. Testcases may be obtained by changing of number of users (N).
- This problem may be considered as offline or online
  - For offline problem it is allowed to use information about all users for making coordinated allocation.
  - For online problem users should be scheduled sequentially according to their order in input.
  - In both cases it is necessary to take into account that in real system this process in total should take less than 0.5ms, so it will be better to reduce complexity of the algorithm as much as possible.