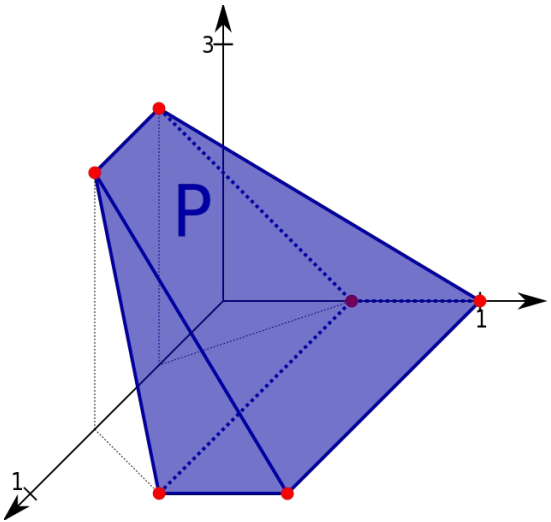
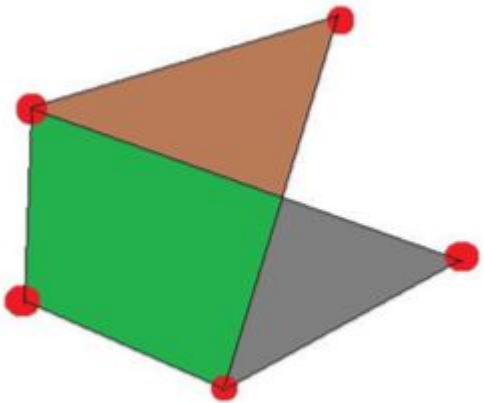
Comparison of Byzantine Vector Consensus Algorithms

Kyle Sung
Shiyang Cheng

Vector Consensus

In a lot of linear optimization problems, the feasible solutions lies inside a convex region.





Byzantine General Consensus

Works for one dimension

Agreement: non-faulty processes agree on the same value

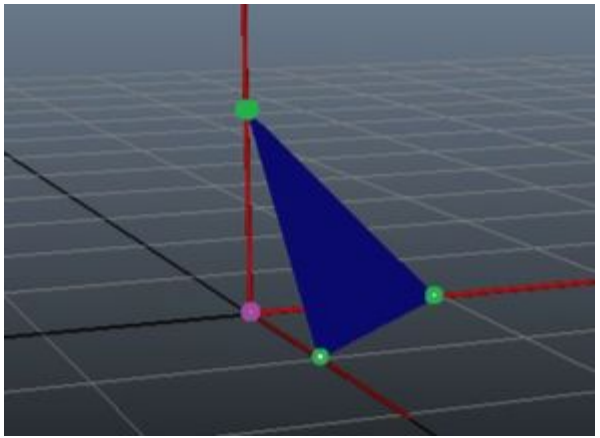
Non-triviality: each value need to be possible

Termination: non-faulty process will decide in finite time

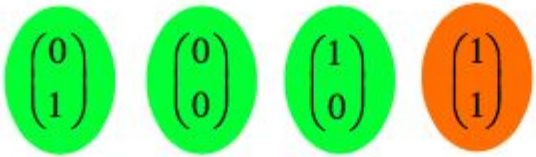
$n \geq 3f+1$

Naive Solution: BGA on each dimension

{(0,1,0),(1,0,0),(0,0,1)} => (0,0,0)



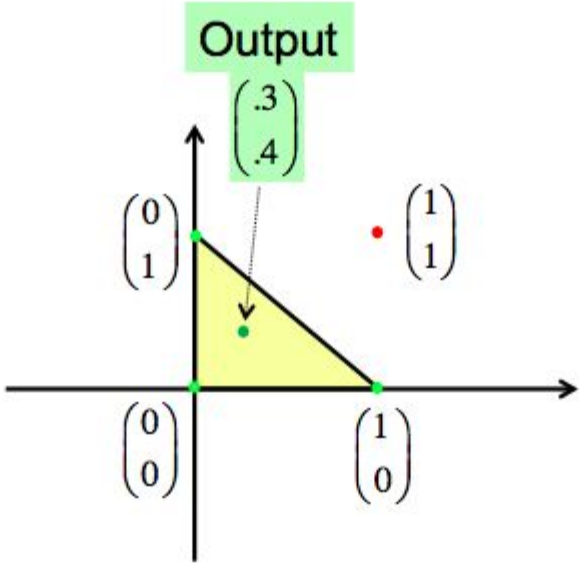
Exact Vector Consensus



Agreement: The decision vector at all the non-faulty processes must be identical.

Validity: The decision vector at each non-faulty process must be in the convex hull of the input vectors of all non-faulty processes.

Termination: Each non-faulty process must terminate after a finite amount of time.

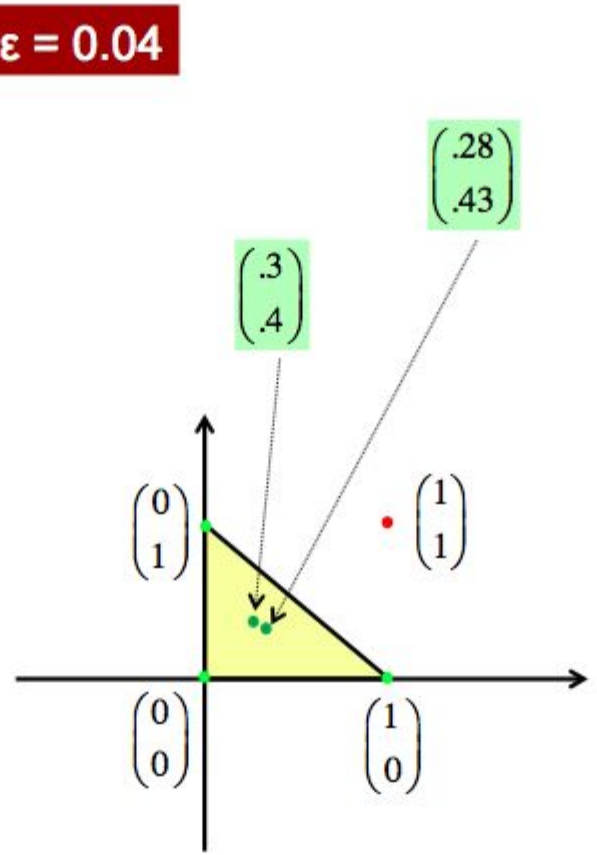


Approximate V consensus

ε-Agreement: Decision vectors at any two non- faulty processes must be within e of each other, where e > 0 is a pre-defined constant

Validity: The decision vector at each non-faulty process must be in the convex hull of the input vectors of all non-faulty processes.

Termination: Each non-faulty process must terminate after a finite amount of time.



Common techniques

Reliable broadcast

- Avoid Byzantine processes convey different contents to different processes in a single round of communication
  - P broadcast a decorated message M={p,r,c}
  - When others receive M, they echo it
  - When process receive n - f echo for M, they send ready
  - When process see f+ 1 ready for M, meaning a non-faulty process necessarily advocates the existence of M, they send ready message
  - When a process receives at least n - f ready message, the original message M is accepted

Common techniques

Witness technique

- Collect values of any two non-faulty processes suitably overlap in every round
- Make non-faulty processes have n-f common values
- As witness are obtained via reliable broadcast, any two non-faulty processes obtain n-2f witness in common, which at least one is non-faulty

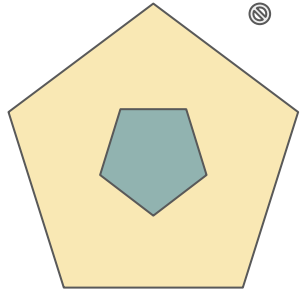
1. P reliably receives n - f messages from other processes, sorting them into Val
2. P reliably transmit its report, which contains the n-f messages first collected in Val, and reliably receives report from other processes, storing them into Rep
3. P collects reports in Rep until n - f witnesses are identified in Wit

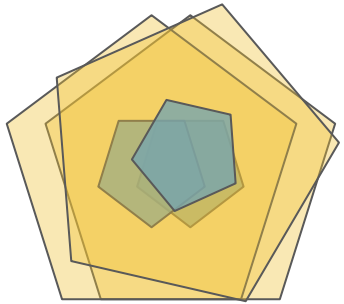
The Safe Area

- non-faulty processes exchange messages containing vectors
- Processes will compute one result in the safe area, which is the convex hull of all the input vectors
- Choose n - f values from n values and the intersection of all combination of n - f values are the safe area
- How to compute the safe area? -> Linear programming

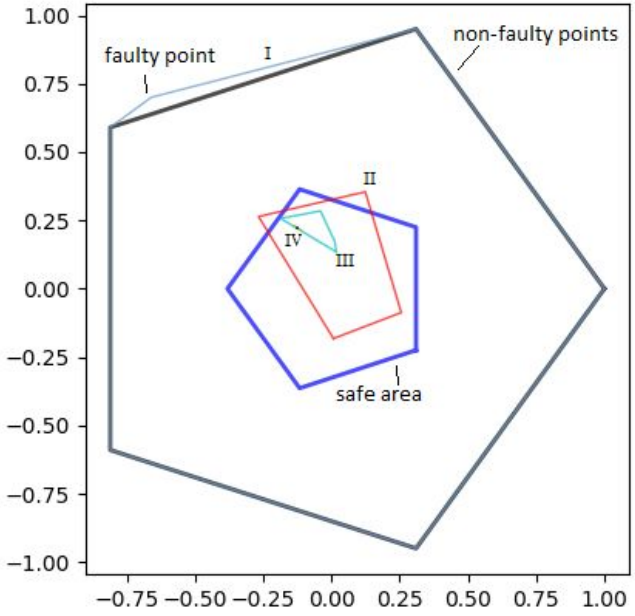
Vaidya-Garg Algorithm

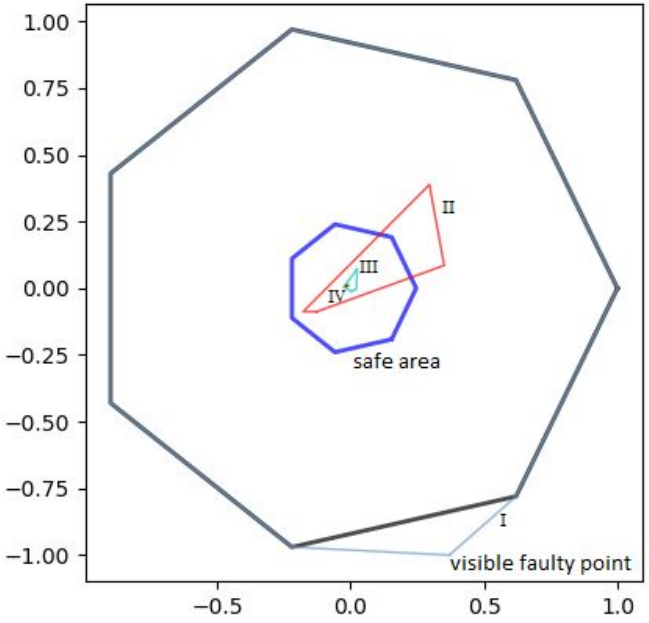
- For R rounds:
  - each process:
    - Collects n-f points from other processes
    - From the collected points, calculates the intersecting safe areas with comn-2f points
    - Pick a point from the safe area
    - Broadcast the point as the new coordinate of the process





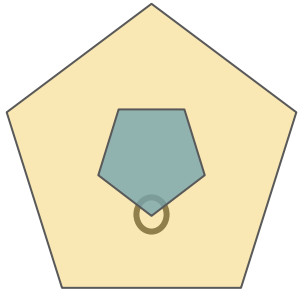
Example


faulty point
I
non-faulty points
II
IV
III
safe area
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

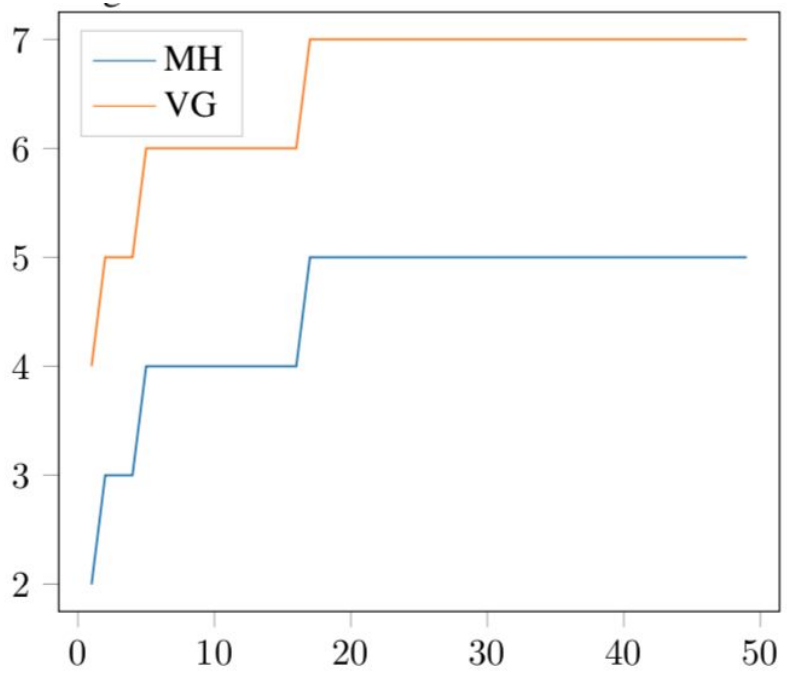

II
III
IV
safe area
I
visible faulty point
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−0.5
0.0
0.5
1.0

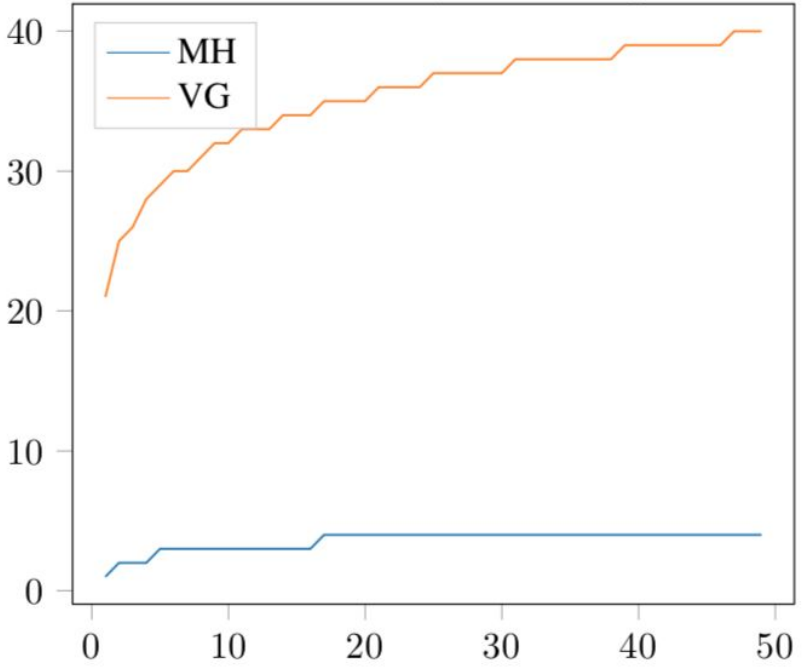
Mendes-Herlinhy Algorithm

- For each dimension:
  - Send:
    - For R times:
      - Collect n-f points from other processes
      - Calculate the safe area
      - Pick the “midpoint”
      - Broadcast the point to all other processes
    - Broadcast “Halt” message
  - Receive:
    - Collect f+1 Halt messages



Rounds of Iteration





Demo

Thanks