A comparative evaluation of approximate Byzantine Vector Consensus algorithms by Vaidya and Garg and by Hammurabi and Herlihy

Kai Qin
Dept. of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX
kai.qin@utexas.edu

Yi Han
Dept. of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX
yh5598@utexas.edu

***Abstract*—This document is for the final project of EE382N-Distributed System about A comparative evaluation of approximate Byzantine Vector Consensus algorithms by Vaidya and Garg and by Hammurabi and Herlihy.**

I. Introduction

We implemented the Mendes-Herlihy algorithm and the Vaidya-Garg algorithm shown in Algorithm 1 and 3. Both algorithms are for multidimensional byzantine approximate agreement problem. The problem we are trying to analyze in this project is “the $Multidimensional\ Byzantine\ Approximate\ Agreement$ (MBA) for asynchronous systems. The problem requires processes to decide on multiple d-dimensional vectors in R^d, all within a fixed Euclidean distance ϵ of each other and inside the convex hull of d-dimensional vectors that were given by the non-faulty processes” [**?**] In an asynchronous system. It is well known according to FLP [**?**], byzantine binary consensus is impossible under a single crash failure. In the approximate agreement problem, not all processes are required to agree on the same output, but they all need to be within a fixed Euclidean distance ϵ. The following result was obtained by this paper [**?**] for the MBA problem, in a fully connected distributed system with up to f Byzantine failures:
– In asynchronous systems, $n > (d + 2)f$ is necessary and sufficient to solve the multidimensional approximate agreement problem. The assumptions for this algorithm includes FIFO, reliable communication channels, and fully connected distributed network.

II. Implementation Setup

We choose to implement a robot convergence problem to examine the message and time complexity of each algorithm.We setup the same environment used to test both algorithms with the following properties:
(1) There are total of 6 servers with 1 byzantine server.
(2) Each server is fully connected to other servers with TCP connections.
(3) Each server has an input of 2 dimensional vector that initially form a convex shape.

Algorithm 1 The Mendes-Herlihy Algorithm

```
 1: $(R, v) \leftarrow$ CalculateRounds(I)
 2: for m $\rightarrow$ 1,....,d do
 3:     H$\leftarrow \varnothing$
 4:     r$\leftarrow$ 1
 5:     while $|H| \leqslant f$ do
 6:         RBSend$((p, m.r, v))$
 7:         upon V $\leftarrow$ RBReceiveWitness$(m.r)$ do
 8:             S $\leftarrow$ Safe$(V)$
 9:             v $\leftarrow$ v $\in$ S such that v$[m]$ = Midpoint$(S(m))$
10:             if r = R then
11:                 RBSend$((p, m.r, halt))$
12:             r $\leftarrow$ r + 1
13:         upon RBRecv$((p', m.r', halt))$, with $r' \geqslant$ r do
14:             H $\leftarrow$ H $\cup$ $((p', m.r', halt))$
15:     end while
16: end for
17: return v
```

Algorithm 2 p.CalculateRounds(I)

```
1: $RBSend((p, 0, I))$
2: $(V, W) \leftarrow (Val, Content(Wit)) from RBReceiveWitness(0)$
3: $U \leftarrow barycenter of Safe_f(W') : W' \in W$
4: $v \leftarrow barycenter of Safe_f(U)$
5: $R \leftarrow \lceil \log_2(\sqrt{d}/\epsilon) max(\sigma_U(m) : 1 \leq m \leq d)) \rceil$
6: return $(R, v)$
```

(4) The byzantine server will initially send random input vector values to all other servers.
(5) When implementing $if\ never\ sent$ in function $RBEcho_f$, we saved the sender’s pid as key in a hash table and save the number of time we sent the corresponding message as value in the hash table.
(6) Since we are working on the 2 dimensional case of the MBA problem, we wrote our $Safe_f$ function based on the Safe Area definition, where we take intersection (see fig 1

Algorithm 3 The Vaidya-Garg Algorithm

```
1: $R \leftarrow 1 + \log_{1/(1-\gamma)} \frac{\sqrt{d}(U-v)}{\epsilon}$
2: for r $\rightarrow$ 1,.... R do
3:     RBSend($(p, r, v)$)
4:     upon M $\leftarrow$ RBReceiveWitness($r$) do
5:         for $M' \subseteq M, |M'| = n - f$ do
6:             $S_{M'} \leftarrow Safe_f(M')$
7:             Z $\leftarrow$ Z $\cup$ DeterministicallyChoosePoint($S_{M'}$)
8:         end for
9:         v $\leftarrow$ $(\Sigma_{z \in Z} z)\ /|Z|$
10: end for
11: return $v$
```

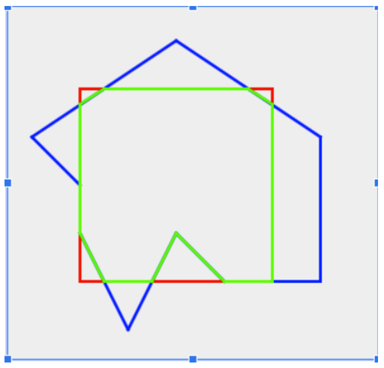

Fig. 1. Example results of take intersection of two convex hull.

of an exemple of how we took intersections [?])of any two $Restrict_f$ of the input message set and returns the vertices of the resulting convex hull.
(7) In line 9 of algorithm 6, the process computes, $S(m)$ the projection of the safe area on coordinate m, then choose a point in the safe area such that its m-th coordinate is in the midpoint of $S(m)$. Here we choose the other coordinate such that it is in the midpoint of the interval of the safe area as well.

III. Message Complexity Comparison

In MH algorithm, each non-faulty process spends $O(n^2)$ messages for reliable broadcast and the witness algorithm for each round. The overall message complexity for all the rounds is

$$O(n^2 d \log(d/\epsilon\ max(\sigma_{I_G}(m) : 1 \leq m \leq d)). \quad (1)$$

In VG algorithm the overall message complexity for all the rounds is

$$O(n^2 d \log_{1/(1-\gamma)}(d/\epsilon\ max(\sigma_{I_G}(m) : 1 \leq m \leq d)). \quad (2)$$

IV. Experimental Time Comparison

For live demo time result comparison, We setup a millisecond time counter that began at the initiation phase of running the algorithms, and logs the time elapsed. For the MH algorithm, initial calculate round function requires around 3000 milliseconds to complete. Both algorithms depend on reliable broadcast and receive witness, thus their time for each round is similar at around 3000 to 4000 milliseconds. The number of rounds needed for The VG algorithm is significantly higher than the MH algorithm, therefore the completion of MH algorithm is faster than the VG Algorithm. Since Our implementation of each process is thread based, accuracy of the logged elapsed time may not be true reflection of a real world implementation of these algorithms.

V. Algorithm Comparison Observations

According to the original paper, "MH [?] algorithm has an asymptotically constant convergence factor in relation to n, but requires certain geometrical operations too restrictive for applications such as robot navigation. In such application, instead of exchanging messages, processes obtain information via visual sensors, projecting them over private coordinate systems. The VG [?] algorithm is better suited for the application above, at the cost of a slower convergence,now asymptotically dependent on n,the number of processes."

VI. Future Works

While implementing the two algorithms, we found it useful to create more comparative methods to show the convergence factor differences. More visualization tools would be useful in further displaying convergence comparisons. During the implementation of this project, we focused on the 2-dimensional scenario, in the next step, we can further expand on the dimensions based on the linear programming algorithm provided in the appendix of the paper.
Robot [?] convergence is one of the more physically tangible application of byzantine vector consensus, other problems such as distributed voting can equally utilize the agreement.