Self-stabilizing alpha-maximal-partitioning

Group 43 - Shivneshwar Velayutham and Madhumitha Venkatesan

October 2022

Chapter 1

Pseudocode

1.1 processor()

while forever **do**
    tempset $\leftarrow$ *myset*
    if *tempset* does not contain *me* **then**
        tempset $\leftarrow$ *me*
    end if
    if size(*tempset*) $>$ *alpha* **then**
        tempset $\leftarrow$ *me*
    end if
    if checktree(*me*, *tempset*) = false **then**
        tempset $\leftarrow$ *me*
    end if
    for all *neighbors* **do**
        neighborset $\leftarrow$ *neighbor.myset*
        if (*neighbor* in *tempset*) AND (*me* not in *neighborset*) AND (size(*neighborset*) $>$ size(*tempset*)) **then**
            tempset $\leftarrow$ *me*
        end if
        if (*neighborset* has *me*) AND (size(*neighborset*) $>$ size(*tempset*)) **then**
            tempset $\leftarrow$ *neighborset*
        end if
        if (*neighbor* not in *tempset*) AND (*me* not in *neighborset*) AND (size(*neighborset*) + size(*tempset*) $\leq$ *alpha*) **then**
            tempset $\leftarrow$ *tempset* + *neighborset*
        end if
    end for
    myset $\leftarrow$ *tempset*
end while

1.2 checktree(root, set)

for p in set **do**
  $explored[p] \leftarrow false$
end for
$explored[root] \leftarrow true$
$q \leftarrow emptyqueue$
$q \leftarrow q.enqueue(root)$
while $q \neq NULL$ **do**
  $temp \leftarrow q.dequeue()$
  for all $neighbors$ of $temp$ **do**
    if ($neighbor$ in set) AND ($explored[neighbor] = false$) **then**
      $explored[neighbor] \leftarrow true$
      $q \leftarrow q.enqueue(neighbor)$
    end if
  end for
end while
if $explored$ contains a $false$ **then**
  $return \leftarrow false$
else
  $return \leftarrow true$
end if

Chapter 2

Proof

The following sections will prove each aspect of the problem.

2.1 Union of all partitions will result in the set of all processors

As we can see from the algorithm, at the minimum each processor will form a partition containing just itself. So no processor will be left out and the union will result in the set of all processors.

2.2 Size of partition cannot be greater than alpha

Each processor checks if size of its partition is greater than alpha and if that's the case, then removes this partition and creates a new one just containing itself.

2.3 Intersection of two partitions is always NULL

Each processor checks if it's trying to form a partition with a neighbor that has not been accepted by the neighbor. If that's the case and the neighbor's partition is bigger than it's, then it removes it's partition and starts again. And each processor checks if there's an incoming connection to form a partition from a neighbor who has a partition of size bigger than it's own. If that's the case, then the processor accepts the neighbor's partition. This way, it can be seen that any intersection will be eventually removed by the processors themselves making them self stabilizing.

2.4 Each partition is a connected tree

The checktree() method does a bfs search to check if the partition is a connected tree and returns false if it's not. When checktree() returns false, then the processor removes the partition and starts afresh with a new one just containing itself.

2.5 Proof that the partitioning is an alpha maximal partitioning

This is can be seen clearly that each processor checks with it's neighbors if they both have partitions that are mutually exclusive and if the sum of the partitions are less than or equal to alpha. If that's the case, then the processor forms a partition by adding both the partitions together making sure no partition can exist by doing union over two partitions. The neighbor will eventually notice this new partition and accept the incoming connection since its partition will be smaller than the neighbor's.

2.6 Proof that algorithm is self stabilizing

It's clear that each processor accepts a neighbor's partition if their partition is bigger than it's own and if the neighbor wants to form a partition with them. It's also clear that each processor will attempt to form a partition with a neighbor if the both the partitions are mutually exclusive and sum of the partitions is not greater than alpha. Thus using this principle, each processor will increase the size of it's partition and will attempt to create connections with other neighbors. If a neighbor instead formed a connection with some other neighbor, this can be verified with neighbor's partition is greater than it's own and hasn't accepted the request to form a partition then the processor resets it's own partition. Hopefully this makes it clear how the algorithm will self stabilize eventually and form an alpha maximal partitioning.

Even if the starting configuration was an configuration where partitions didn't form a connected tree, the checktree() method would help reset these illegal partitions. If the starting configuration was a connected tree, then each processor will see if it's connections has been accepted by other neighbors, if not it will reset the partition therefore proving that the algorithm will self stabilize.