Algorithm 5 : phase_2_abs(current node, opposite open list, dir)

```
nodeslist.push(currentnode)//expandingtheothersidefrontier
for node in opposite open list do
  if abstract_isExpandable(node) then
    expandablelist.push(node)
  end if
end for
while expandablelist ≠ {} do
  node = Pop(expandablelist)
  if abstract_isExpandable(node) then
    node →′ closed′
    for neighbour in expand(node) do
      child = Node(neighbour, direction(node), g + 1,′ open′)
      if check if child already exist in opposite open list then
        return gvalue(node) + gvalue(child)
      end if
      for node in nodes list do
        nodes list.push(child)
        check for duplicates and replace them if they have lower gvalue
      end for
      if abstract_isExpandable(child) then
        expandablelist.push(child)
      end if
    end for
  end if
end while
while True do
  if switcher == 0 then
    switcher = 1
    phase_3_abs(nodeslist, oppositeopenlist, highlimforward)
    if we find collision in phase 3 then
      return high lim foward + high lim backward
    end if
  else if switcher == 1 then
    switcher = 0
    phase_3_abs(oppositeopenlist, nodeslist, highlimbackward)
    if we find collision in phase 3 then
      return high lim foward + high lim backward
    end if
  end if
end while
```