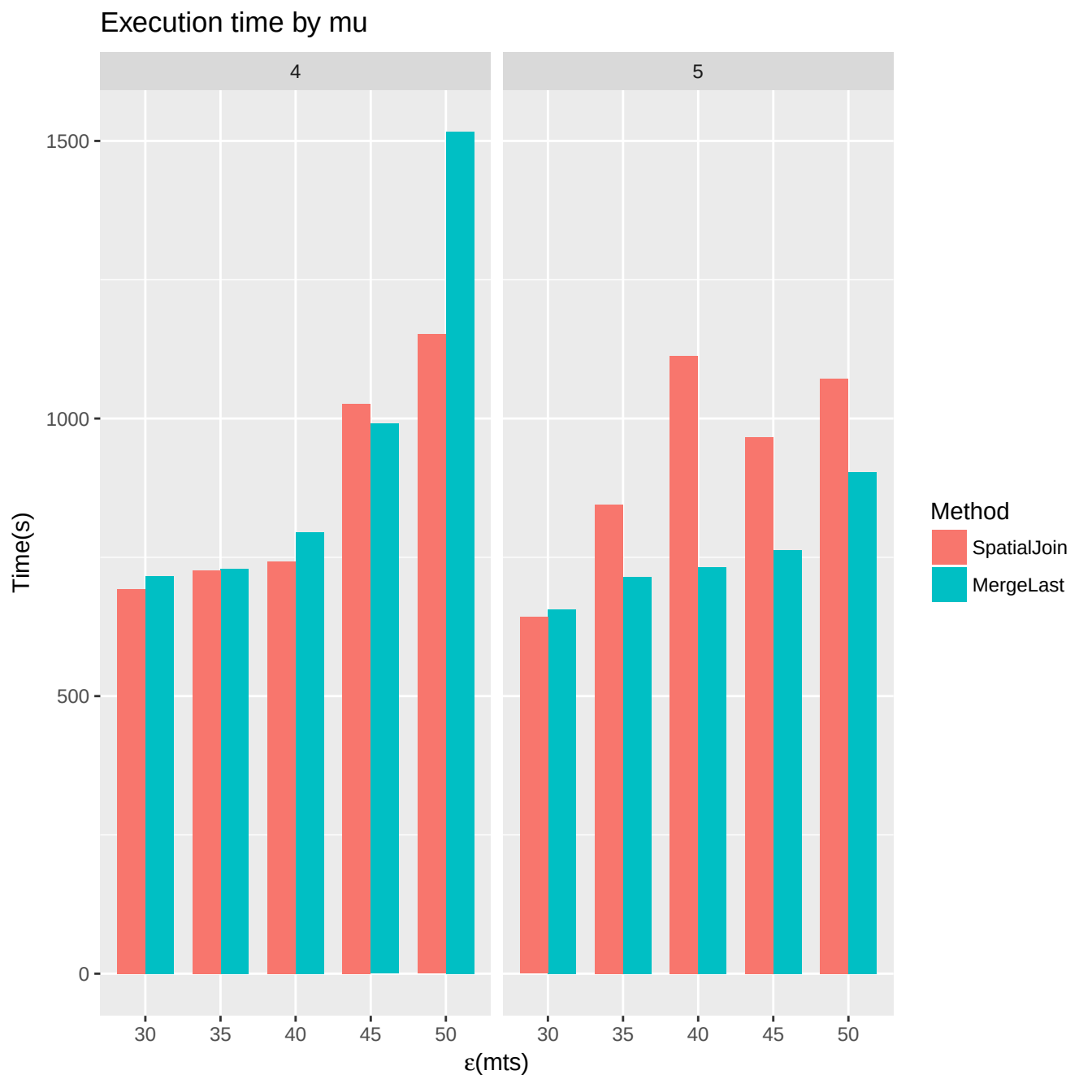
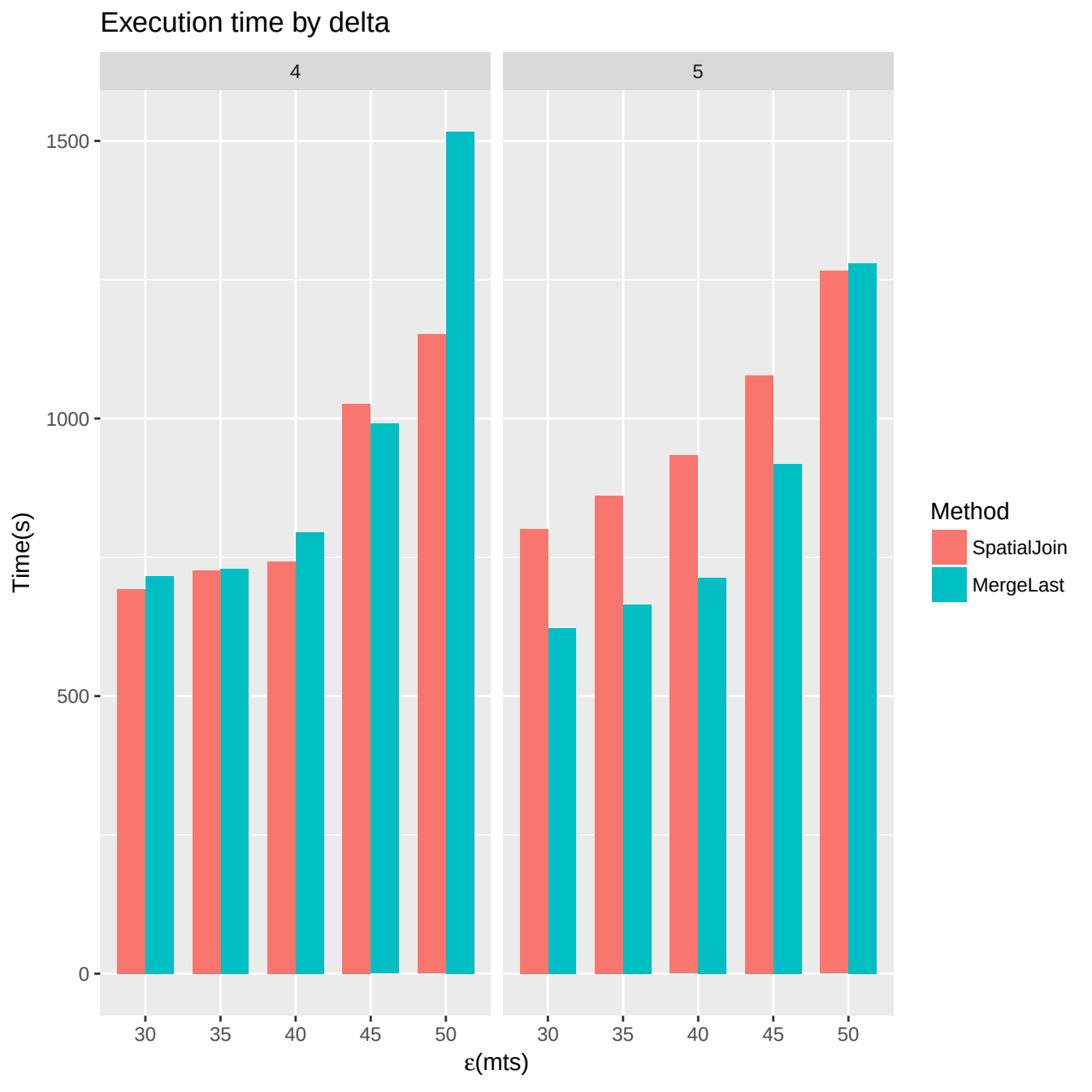
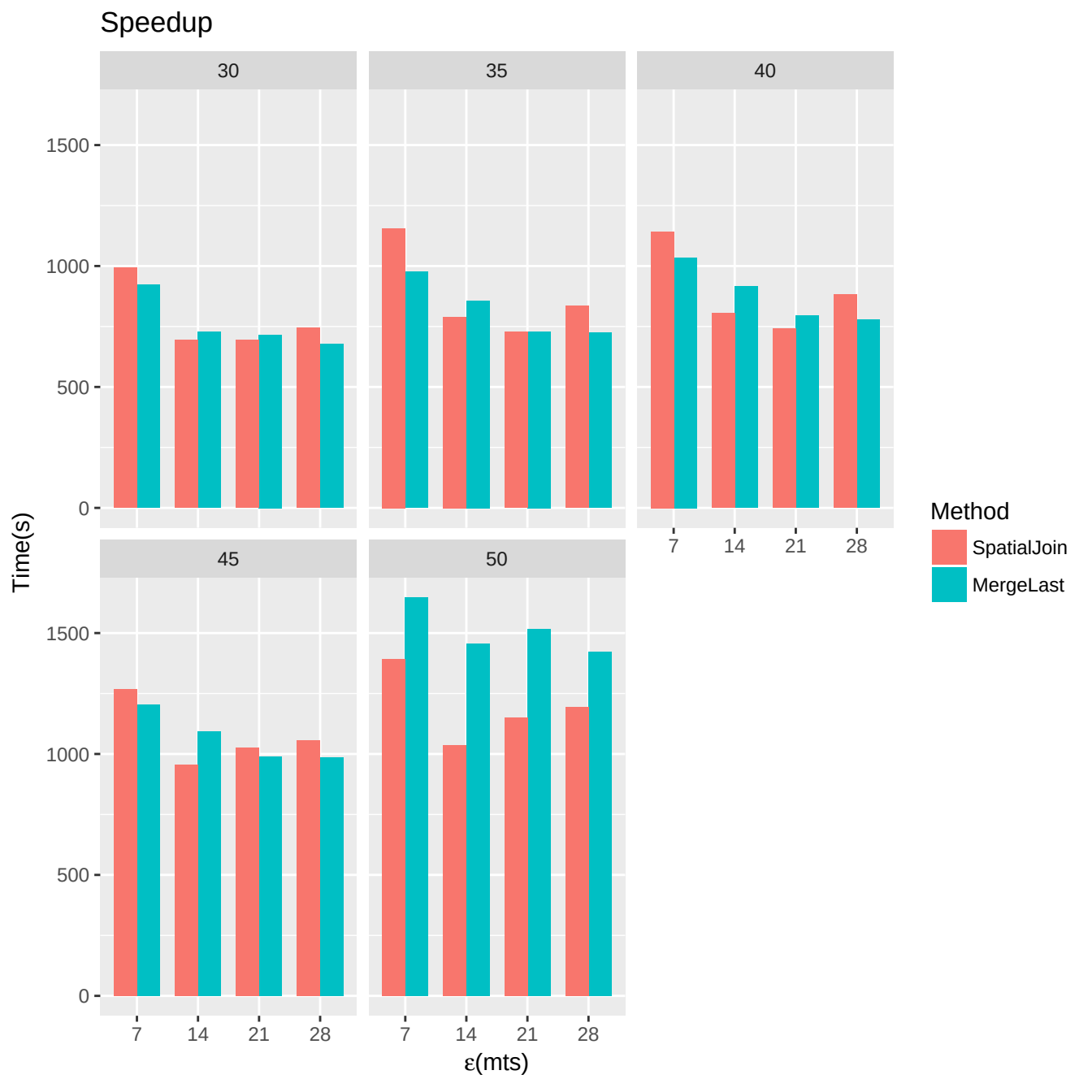
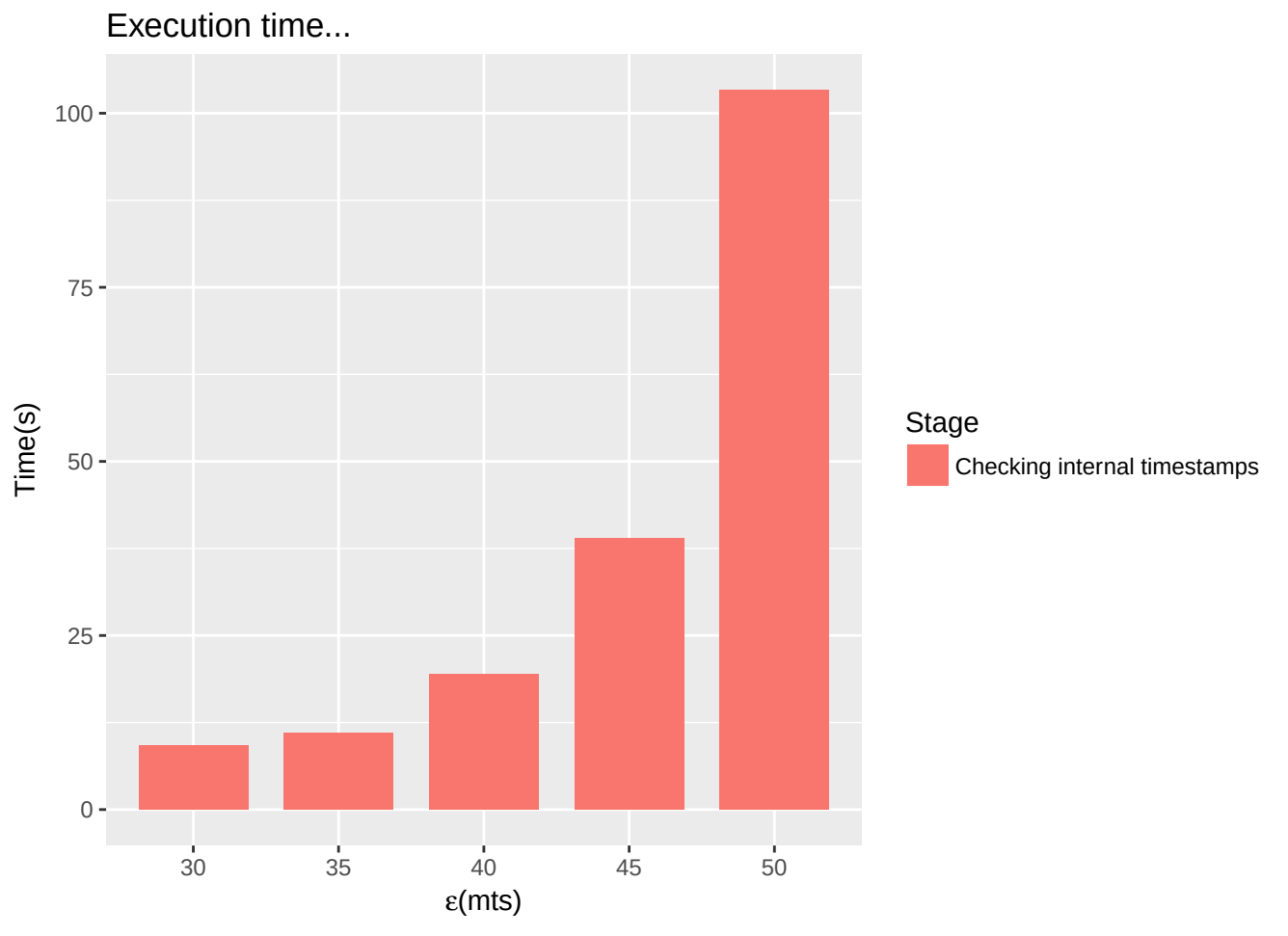
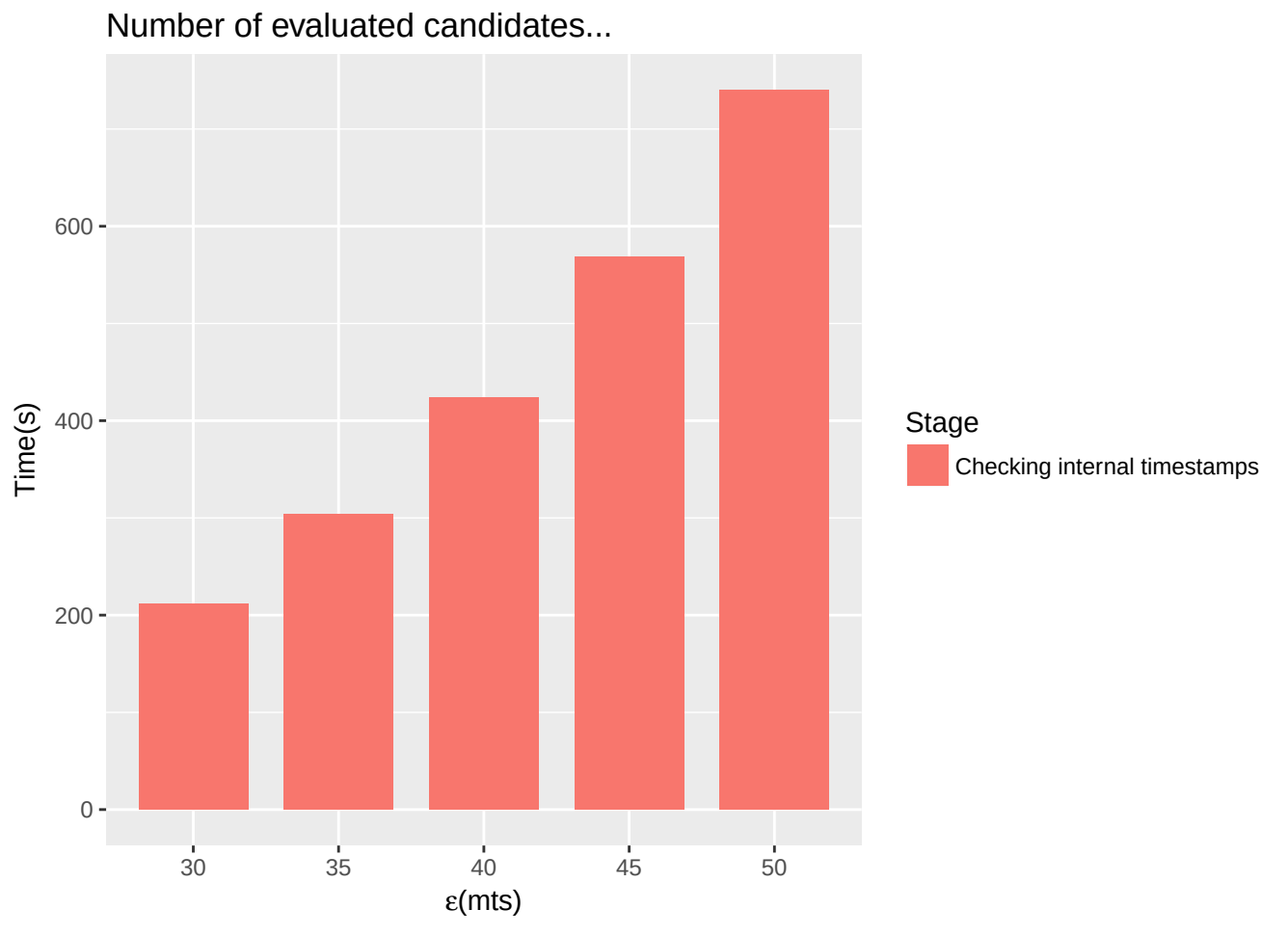

Execution time by mu
4
5
1500
1000
500
0
Time(s)
30
35
40
45
50
ε(mts)
Method
SpatialJoin
MergeLast


Execution time by delta
4
5
1500
1000
500
0
Time(s)
30
35
40
45
50
ε(mts)
Method
SpatialJoin
MergeLast


Speedup
30
35
40
45
50
1500
1000
500
0
Time(s)
7
14
21
28
ε(mts)
Method
SpatialJoin
MergeLast


Execution time...
Time(s)
100
75
50
25
0
30
35
40
45
50
ε(mts)
Stage
Checking internal timestamps


Number of evaluated candidates...
Time(s)
600
400
200
0
30
35
40
45
50
ε(mts)
Stage
Checking internal timestamps

Algorithm 1 MergeLast - Finding flock patterns looking δ timestamps ahead.

Input: Set of spatio-temporal points P, maximum distance ϵ, minimum size μ, minimum duration δ , maximum traveled distance τ
Output: Set of flock patterns F

```
T ← sorted set of timestamps in P
for each timestamp t_i in T − δ do
    P_i ← Set of points at timestamp t_i
    C_i ← getMaximalDisks(P_i, ϵ, μ)
    P_{i+δ} ← Set of points at timestamp t_{i+δ}
    C_{i+δ} ← getMaximalDisks(P_{i+δ}, ϵ, μ)
    if C_i ≠ Ø and C_{i+δ} ≠ Ø then
        F' ← C_i ⋈_τ C_{i+δ} {Distance join...}
        P' ← Set of points at timestamp t_j involved in F'
        for each timestamp t_j between t_i and t_{i+δ} do
            F ← Ø
            for each candidate flock f in F' do
                P'_j ← Set of points at timestamp t_j from P' involved in f
                d ← get maximum distance between points in P'_j
                if d < ϵ then
                    add f to F
                else
                    f' ← getMaximalDisks(P'_j, ϵ, μ)
                    add f' to F
                end if
            end for
            F' ← F
        end for
    end if
end for
return F
```

Algorithm 2 getMaximalDisks(T,ϵ,μ) - Finding maximal disks following a parallel approach.

Input: Set of points T, maximum distance ϵ and minimum size μ
Output: Set of maximal disks M

find the set of pairs of points P in T which are ϵ distance each other
$C \leftarrow \emptyset$
for each p_i in P **do**
  compute disks c_i^1 and c_i^2 of p_i using ϵ
  add c_i^1 and c_i^2 to C
end for
$D \leftarrow \emptyset$
for each c_i in C **do**
  find the set of points ρ_i which lie ϵ distance around c_i
  if $|\rho_i| \geq \mu$ **then**
    compute centroid ς_i of the MBR of ρ_i
    set $d_i.center$ as ς_i
    set $d_i.points$ as ρ_i
    if d_i not in D **then**
      add d_i to D {Pruning duplicate candidates...}
    end if
  end if
end for
build an R-Tree $disksRT$ using centers in D
$E \leftarrow \emptyset$
for each MBR in $disksRT$ **do**
  expand MBR to create an expanded MBR ε_i using a buffer of ϵ distance
  add ε_i to E
end for
for each d_i in D **do**
  for each ε_j in E **do**
    if $d_i.center \cap \varepsilon_j$ **then**
      add d_i to ε_j
    end if
  end for
end for
$M \leftarrow \emptyset$
for each ε_i in E **do**
  $\chi \leftarrow \emptyset$
  for each d_i in ε_i **do**
    add $d_i.points$ to χ
  end for
  find the set of maximal patterns F in χ
  for each f_i in F **do**
    compute centroid ς_i of the items in f_i
    if ς_i is not in the expansion area of ε_i **then**
      set $m_i.center$ as ς_i
      set $m_i.points$ as f_i
      add m_i to M
    end if
  end for
end for
return M