A Recursive Recomputation Approach to Smoothing in Nonlinear and Bayesian State-Space Modeling

Takashi Tsuchiya
(The Institute of Statistical Mathematics)

(joint work with Kazuyuki Nakamura
(The Institute of Statistical Mathematics and JST))

Modification of Slide Presented at Workshop on Advances in Optimization (Tokyo Inst. Tech.), Apr20, 2007

Contents of the talk

- In this talk, we deal with a computational issue in numerical smoothing in nonlinear and Bayesian state-space models.
- Link between optimization and today's talk: Automatic differentiation.

Paper

Kazuyuki Nakamura and Takashi Tsuchiya:

A Recursive Recomputation Approach for Smoothing in Nonlinear State Space Modeling --- An Attempt for Reducing Space Complexity ---

(Appeared in IEEE Transactions on Signal Processing)

State Space Model

- Basic model for analyzing events with one dimensional structure, e.g., time-series, signal, DNA sequence etc.
- Wide applications in statistics, machine learning, signal processing, bioinformatics etc.

State Space Model

Innovation of state

$$
\begin{aligned}
x_{t+1} &= f(x_t) + \varepsilon_t \\
y_t &= g(x_t) + \eta_t
\end{aligned}
$$

Observation

η_t, ε_t : (Non Gaussian) white noize

State Space Model

- $f(x),\ g(x)$ Vector functions of appropriate dimensions
- x_t : State vector
- y_t : Observation

State Space Model

- $Y_0, \ldots, Y_{T-1}$: Observation (Considered as realization of y_t .)
- T : Length of data
- Notation

$$s_{t_1:t_2} \equiv (s_{t_1}, \ldots, s_{t_2})$$

State Space Model

- Major Problem：
  Computing conditional distribution of x(t) given observation Y(・).

$p(x_{t+1}|Y_{0:t})$ ： Predictive distribution

$p(x_t|Y_{0:t})$ ： Filtering distribution

$p(x_t|Y_{0:T})$ ： Smoothing distribution

Applications of Smoothing

- Change point problems
- Estimation of a posteriori probability of models
- Target tracking
- Image processing
- Robotics
- Speech recognition etc. etc.

Scheme for computing filtering and predictive distribution

↓ ← Y_t Bayes Theorem

$p(x_t|Y_{1:t})$ Filtering Distribution

↓ Innovation of the State

$p(x_{t+1}|Y_{1:t})$ Predictive Distribution

↓ ← Y_{t+1} Bayes Theorem

$p(x_{t+1}|Y_{1:t+1})$ Filtering Distribution

Recursive formulas for predictive and filtering distributions

(Predictive distribution)

$$P(x_{t+1}|Y_{1:t}) = \int P(x_{t+1}|x_t)P(x_t|Y_{1:t})dx_t$$

(Filtering distribution)

$$P(x_{t+1}|Y_{t+1}) = \frac{P(Y_{t+1}|x_{t+1})P(x_{t+1}|Y_{1:t})}{\int P(Y_{t+1}|x_{t+1})P(x_{t+1}|Y_{1:t})dx_{t+1}}$$

Scheme for computing smoothing distributions

$$p(x_{t-2}|Y_{1:t-2})$$

$$p(x_{t-1}|Y_{1:T}) \leftarrow p(x_{t-1}|Y_{1:t-1})$$

$$p(x_t|Y_{1:T}) \leftarrow p(x_t|Y_{1:t})$$

$$p(x_{t+1}|Y_{1:T})$$

Filtering distribution

Smoothing distribution

Difficulty in computing smoothing distributions

- Recursion is executed in the reverse chronicle order

↓

- Need for preserving all filtering distributions !
- Storage is proportional to T

Difficulty in computing smoothing distributions

- Let M be the storage necessary for storing the filtering distribution at each time step.
- Then computation of smoothing requires O(MT) storage.
- M is very large (hi-dimensional state space case)

Representation of State Distribution

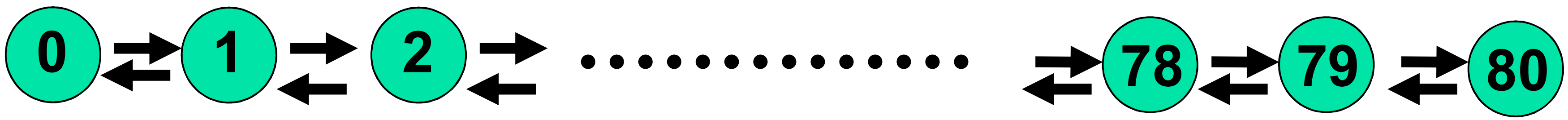
- Typically, state distribution is approximately represented as the set of huge number of particles in the state space. (Particle filter; major innovation in the last decade)

A Remedy: Recomputation Scheme

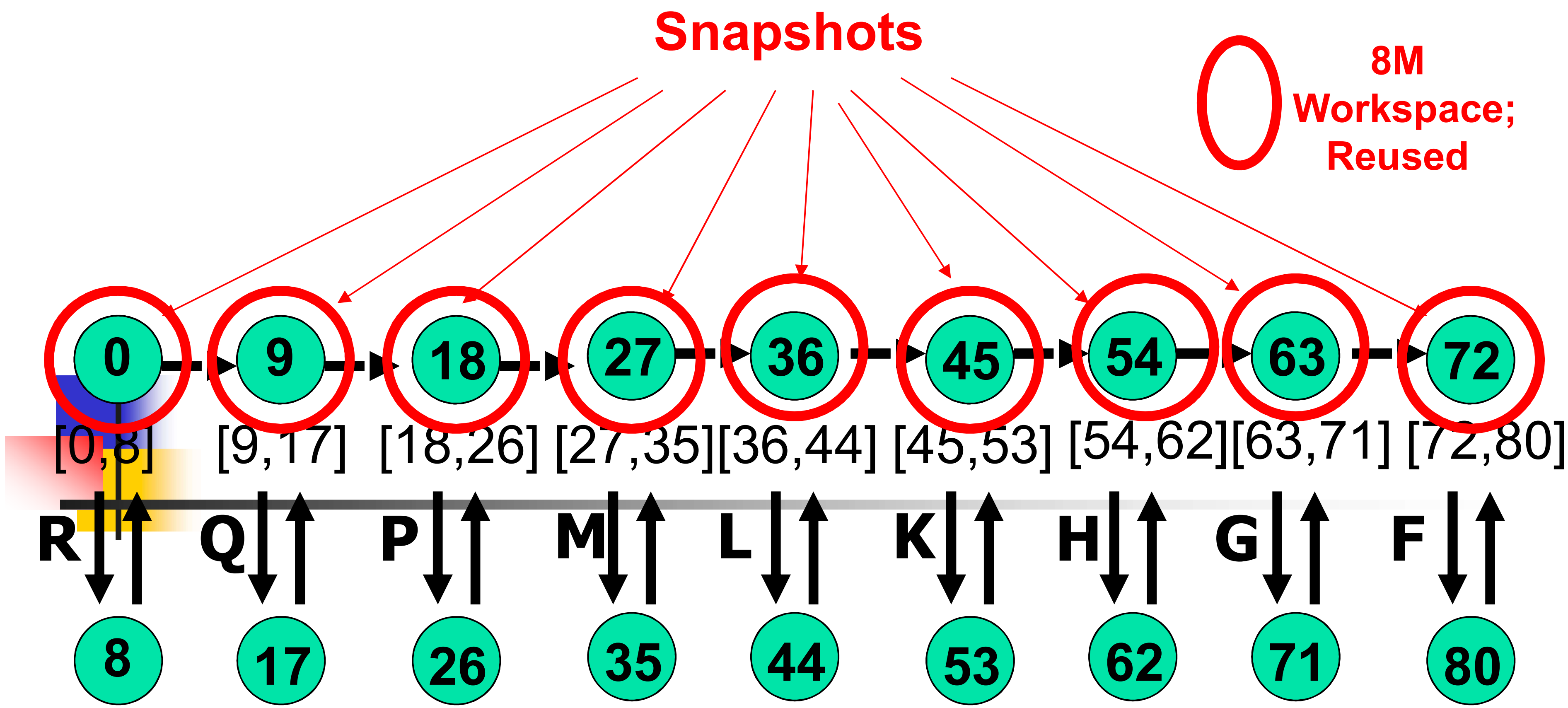
- Snapshot at t : the set of all variables necessary to reproduce all the computational process after t.
- Idea: Take snapshots at appropriate time points and then recompute filtering distributions as they are needed.

Elementary Recursive Recomputation Scheme

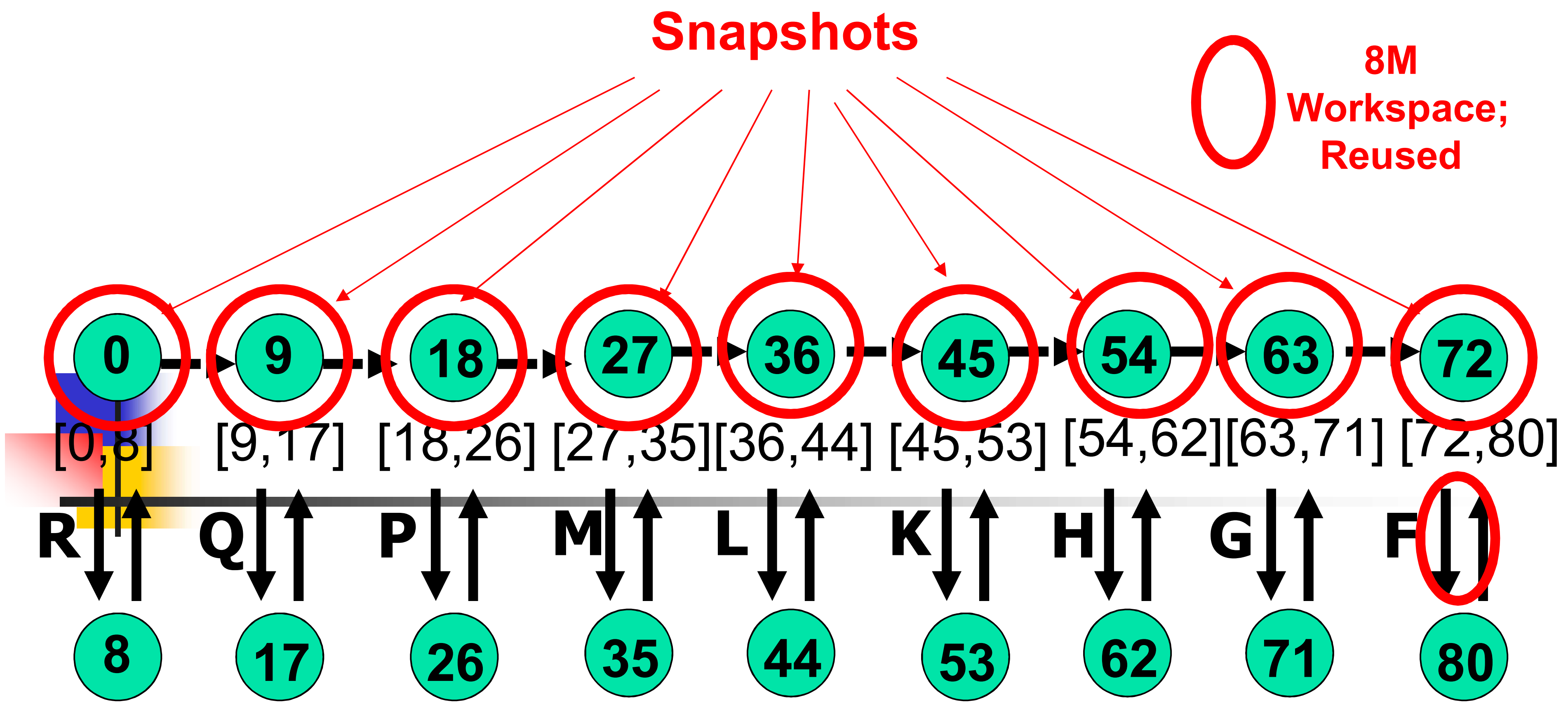
- We explain the method with T=81.



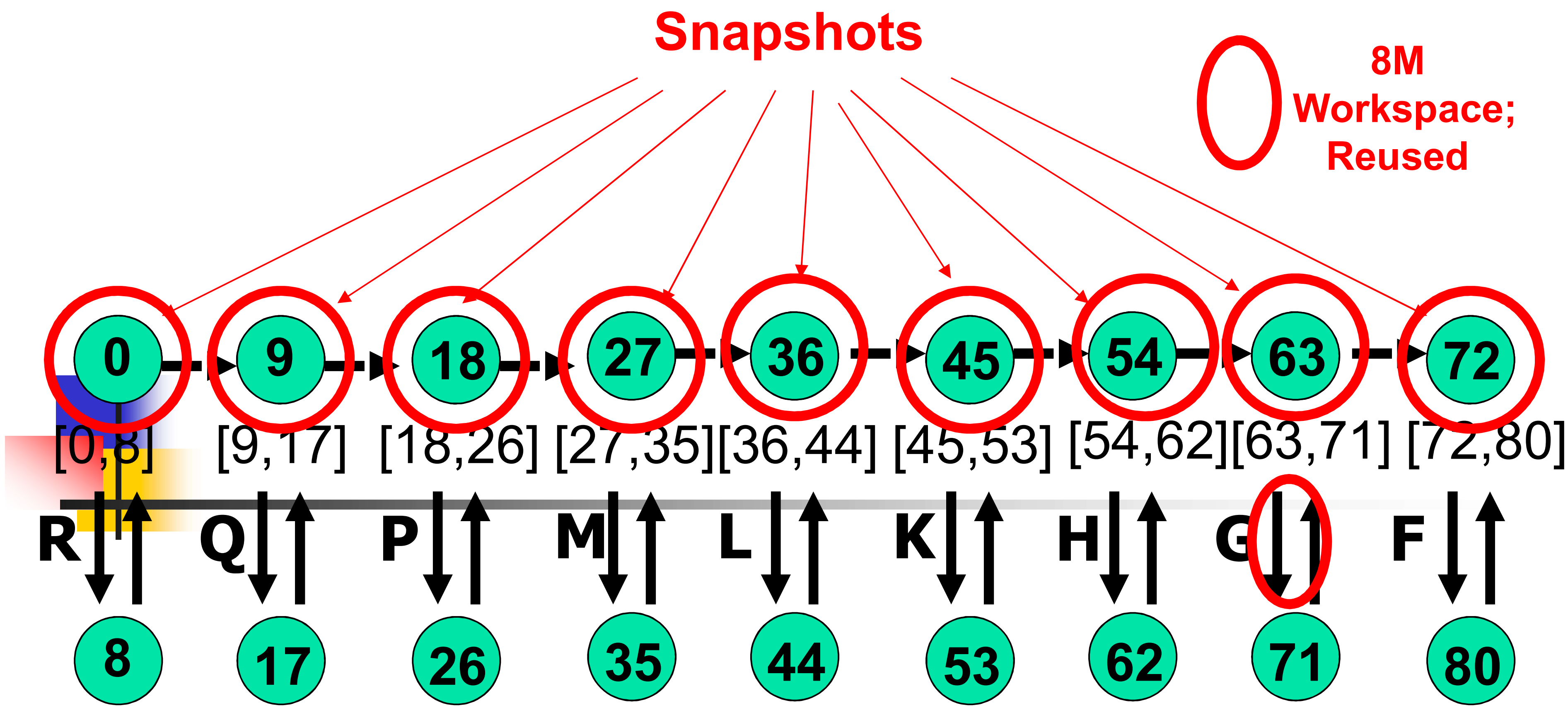
Ordinary, the storage of 81M is required to compute Smoothing distributions.



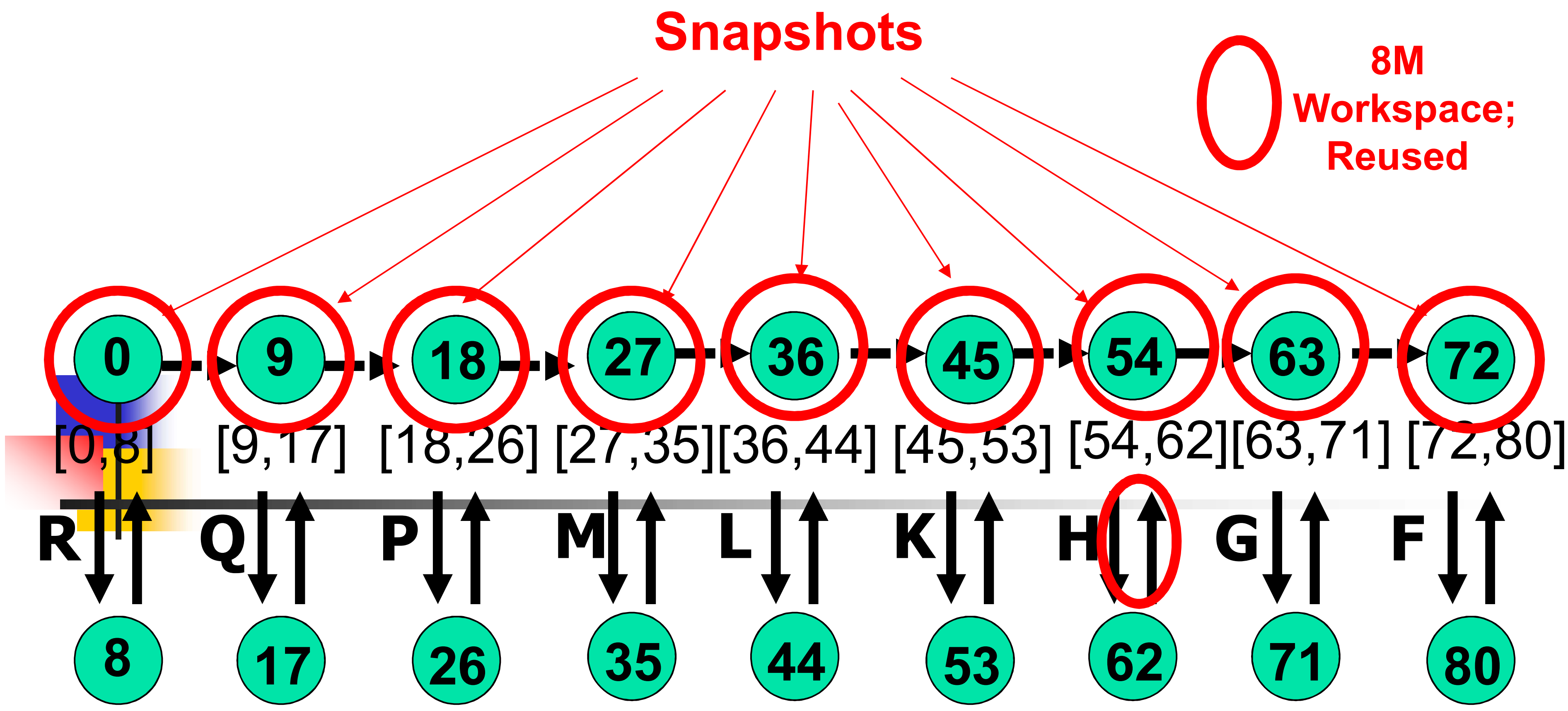
Perform computation of filtering distributions once up to t=72
Taking snapshots at t=0, 9, 18, … ,72

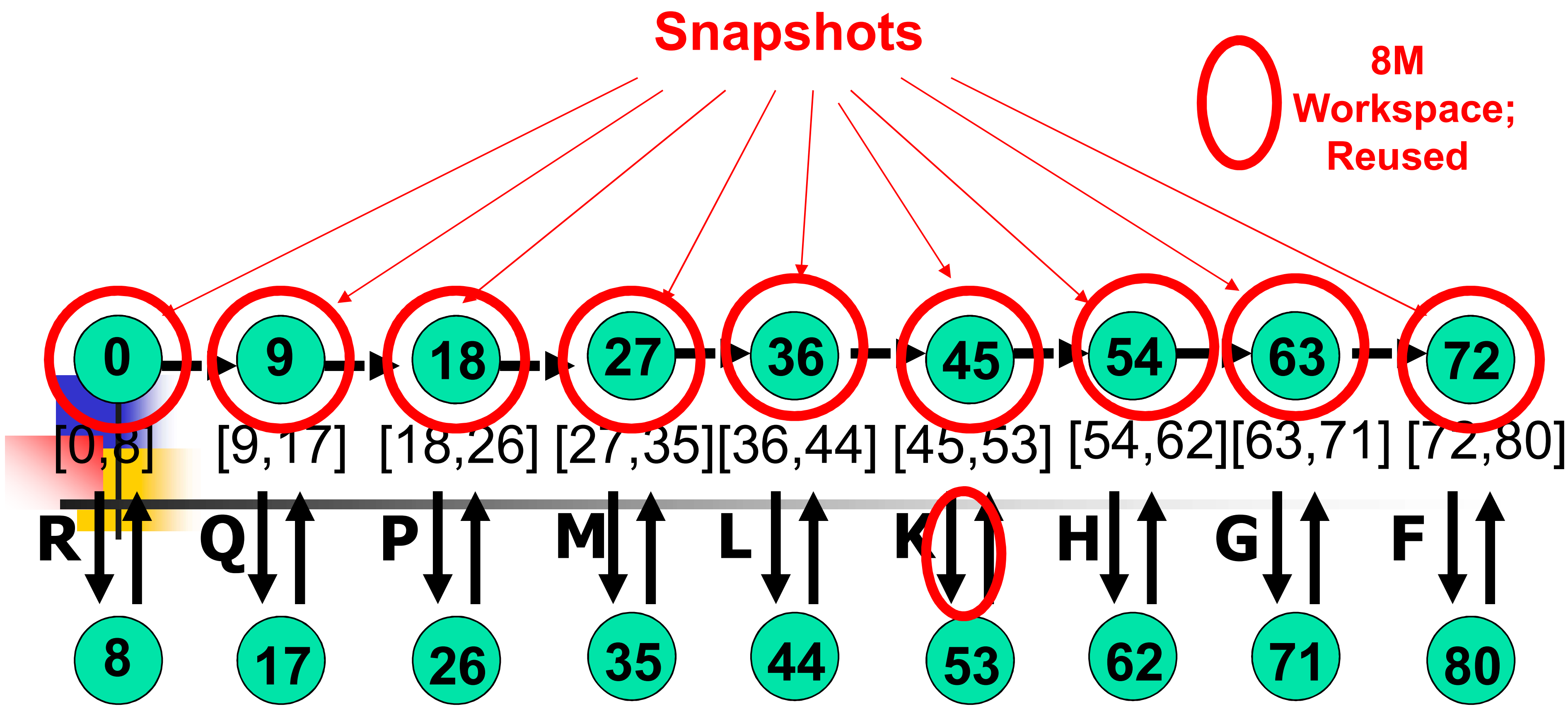


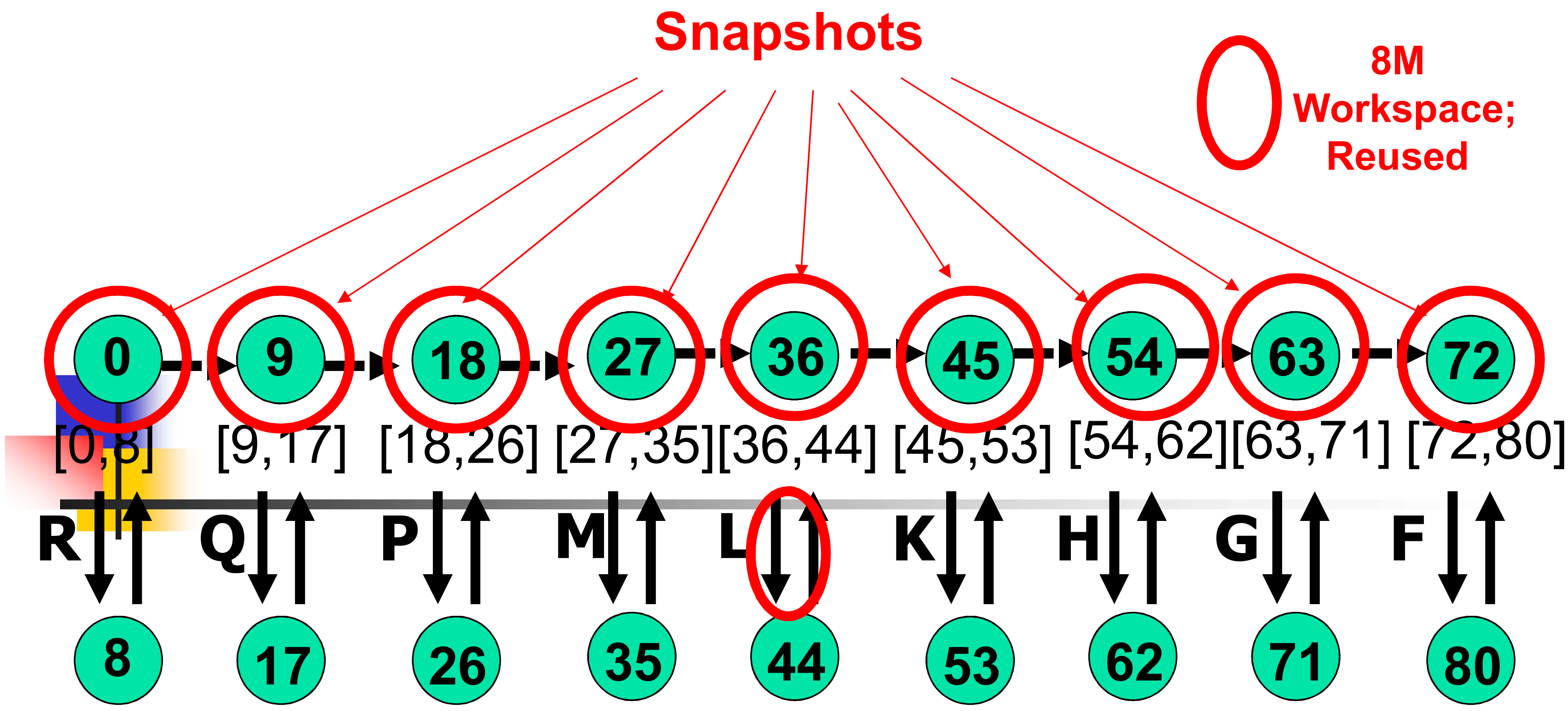
Compute the filtering distributions from t=72 to 80 and back. (Use 8M workspace which is going to be reused.)

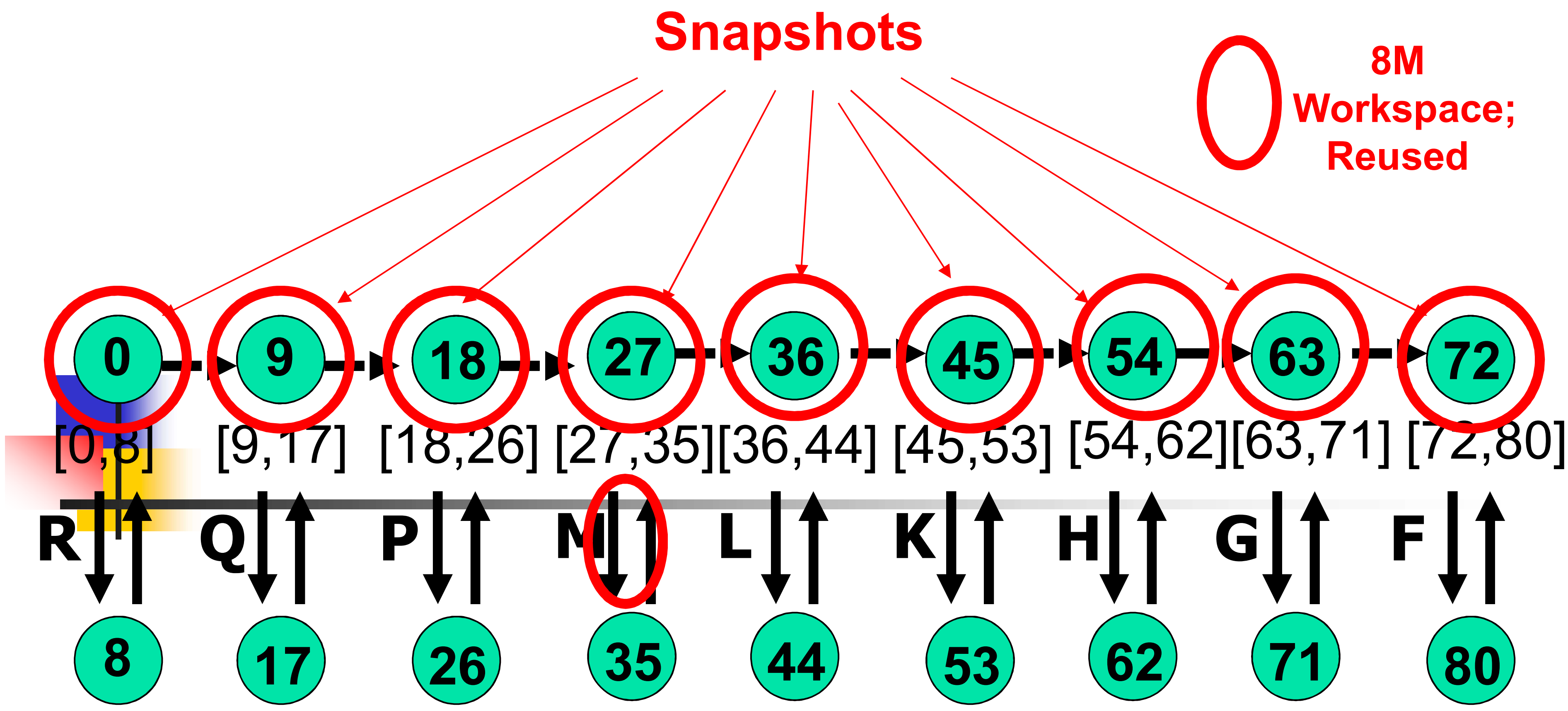


Compute the filtering distributions from t=63 to 71 and back. (8M workspace is reused.)

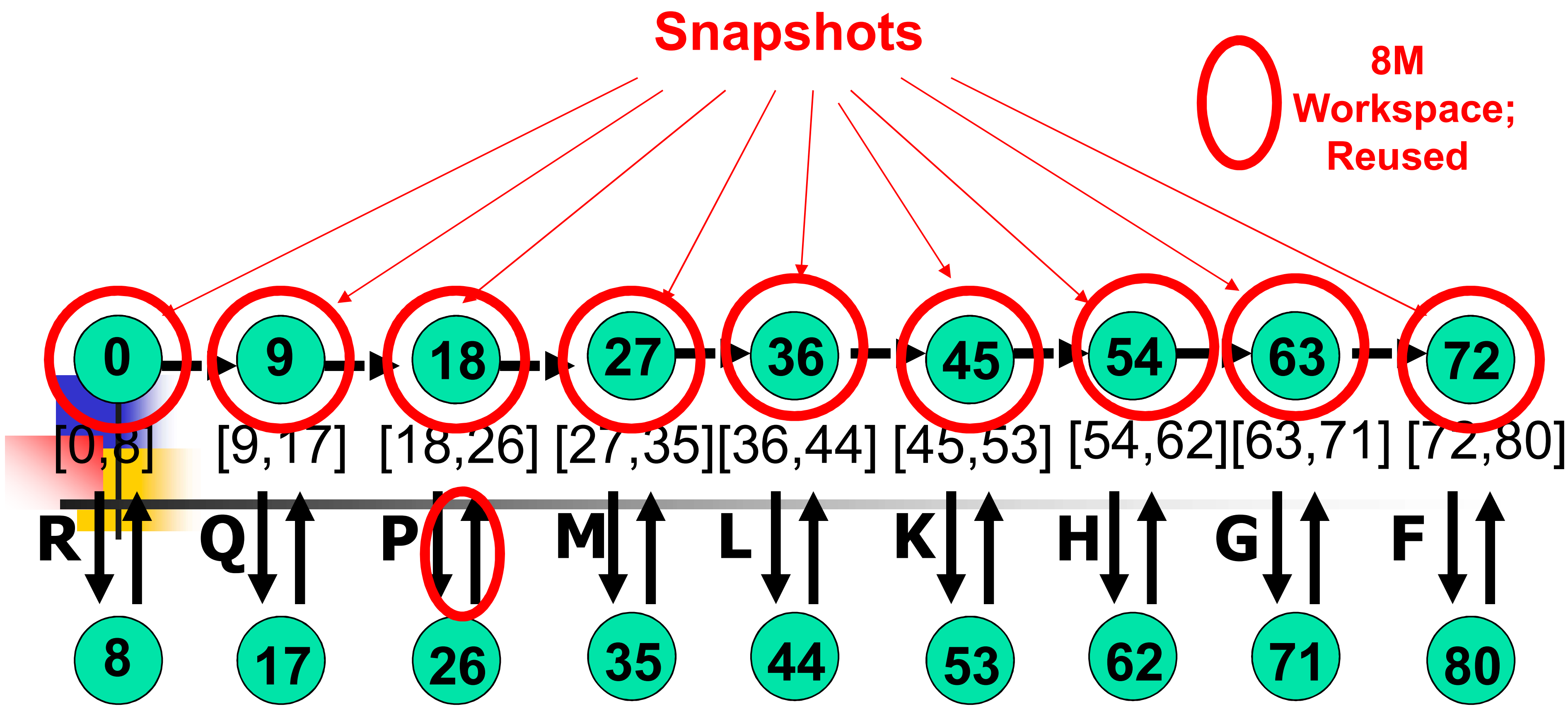

Snapshots
8M Workspace; Reused
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80

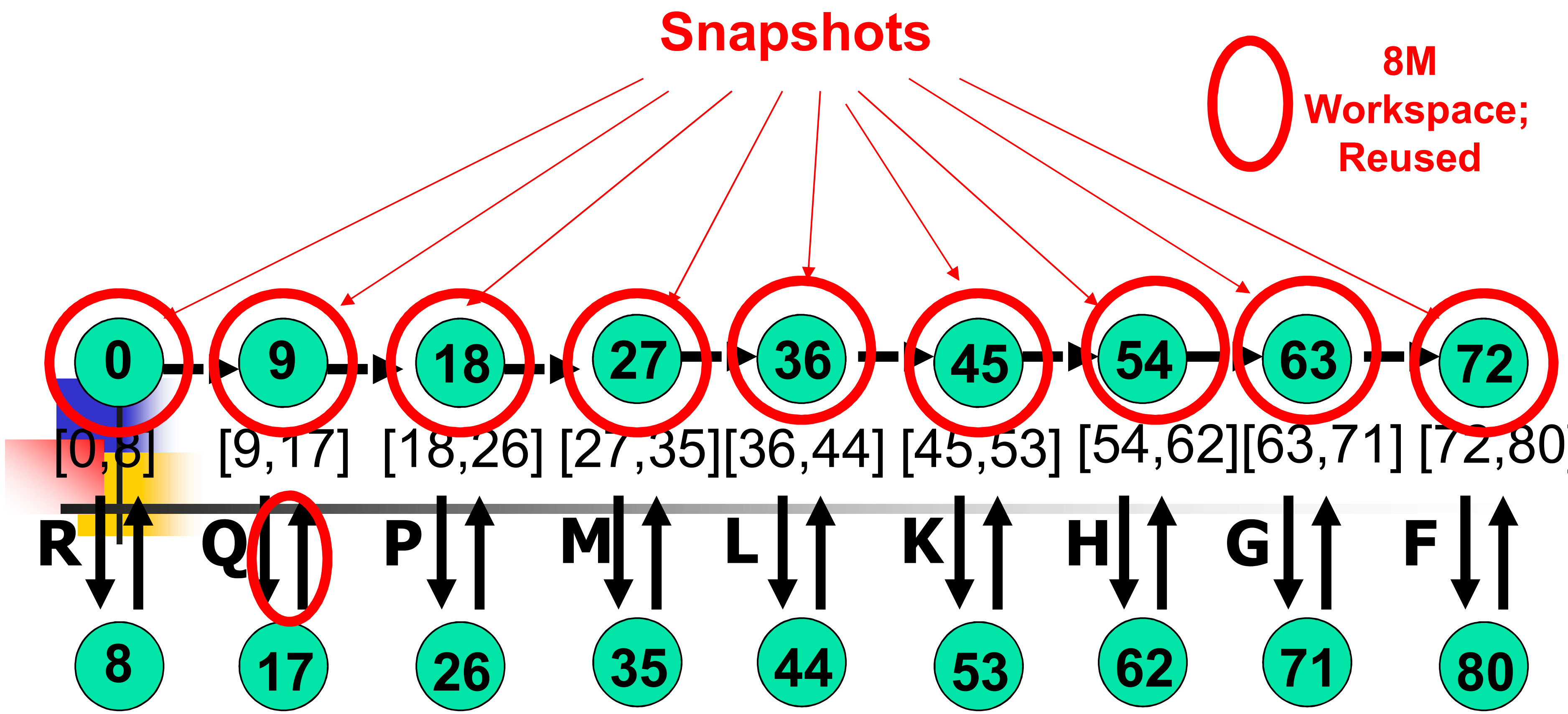

Snapshots
8M
Workspace;
Reused
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80

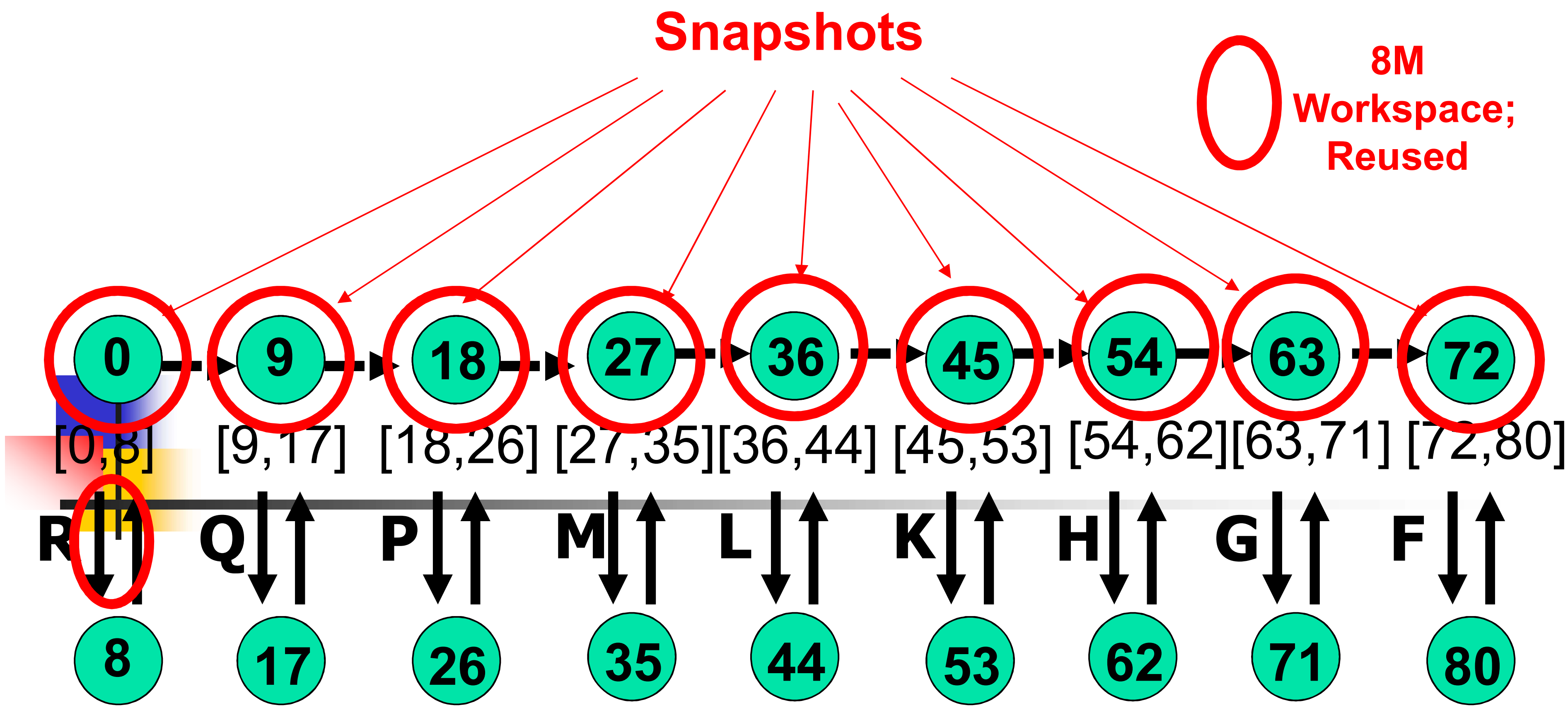

Snapshots
8M
Workspace;
Reused
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80



Snapshots
8M Workspace; Reused
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80


Snapshots
8M Workspace; Reused
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80


Snapshots
8M
Workspace;
Reused
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80



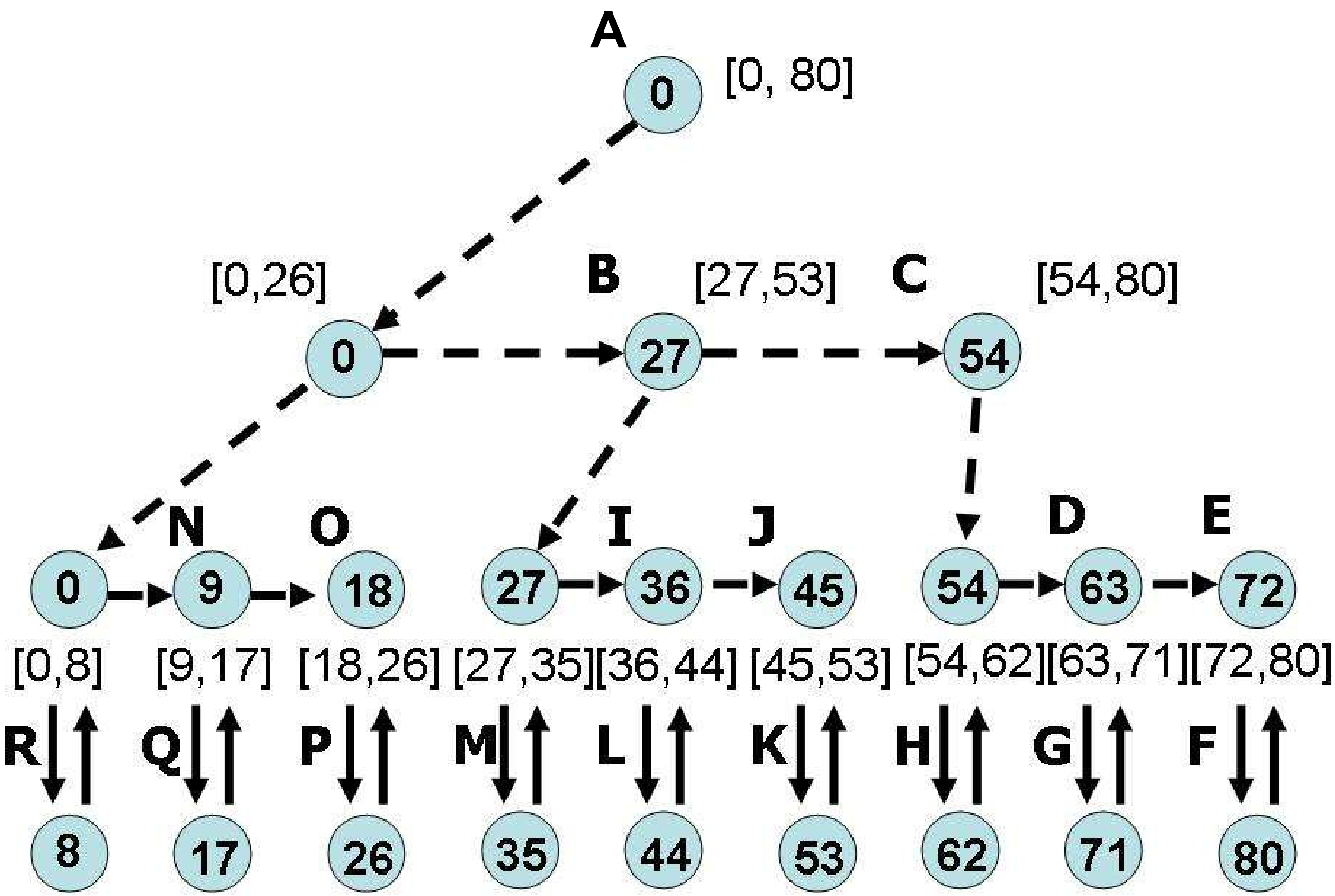
After all, we are able to compute smoothing distributions with 9+8 = 17 M storage.

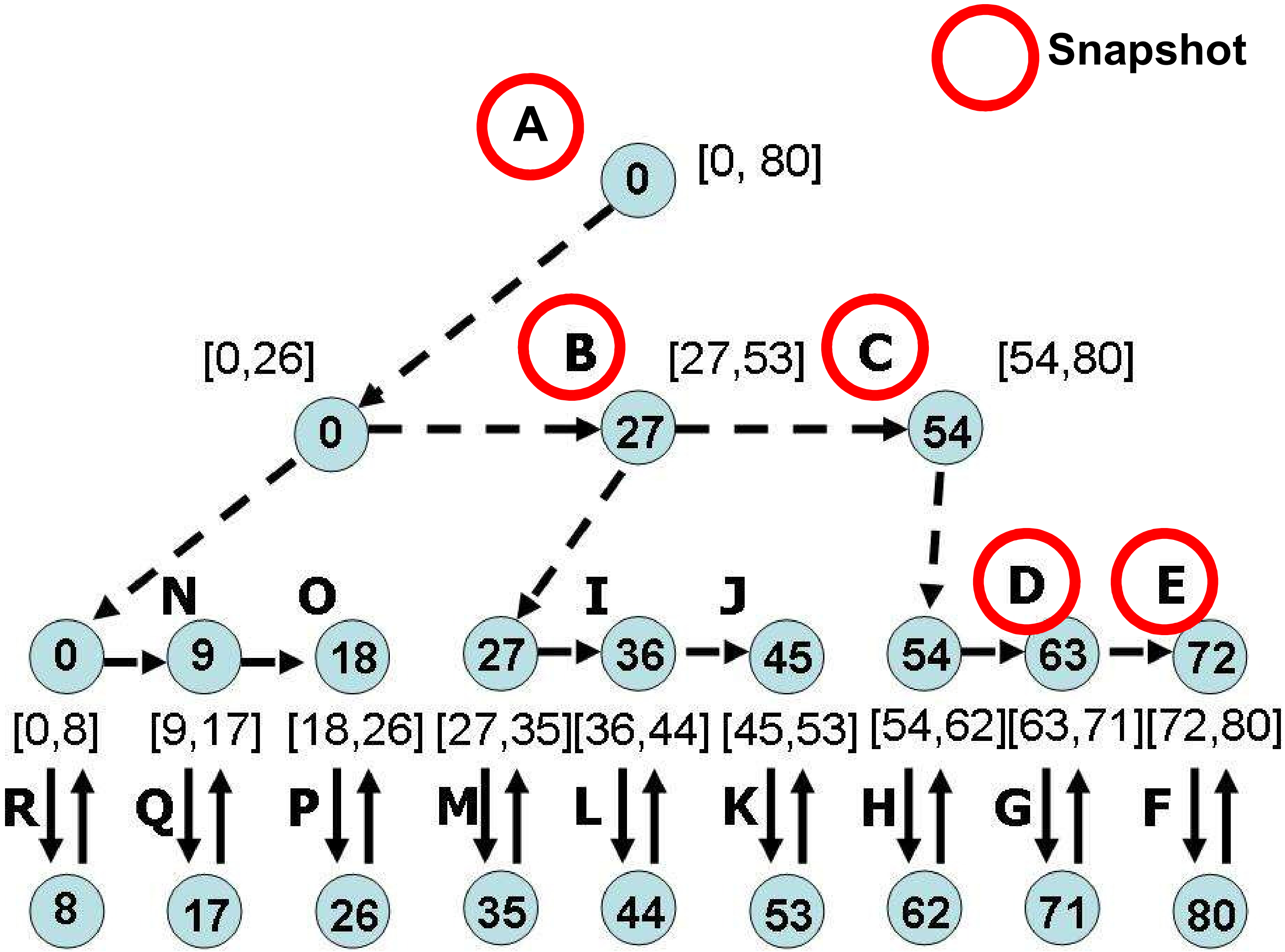
Elementary Recursive Recomputation Scheme

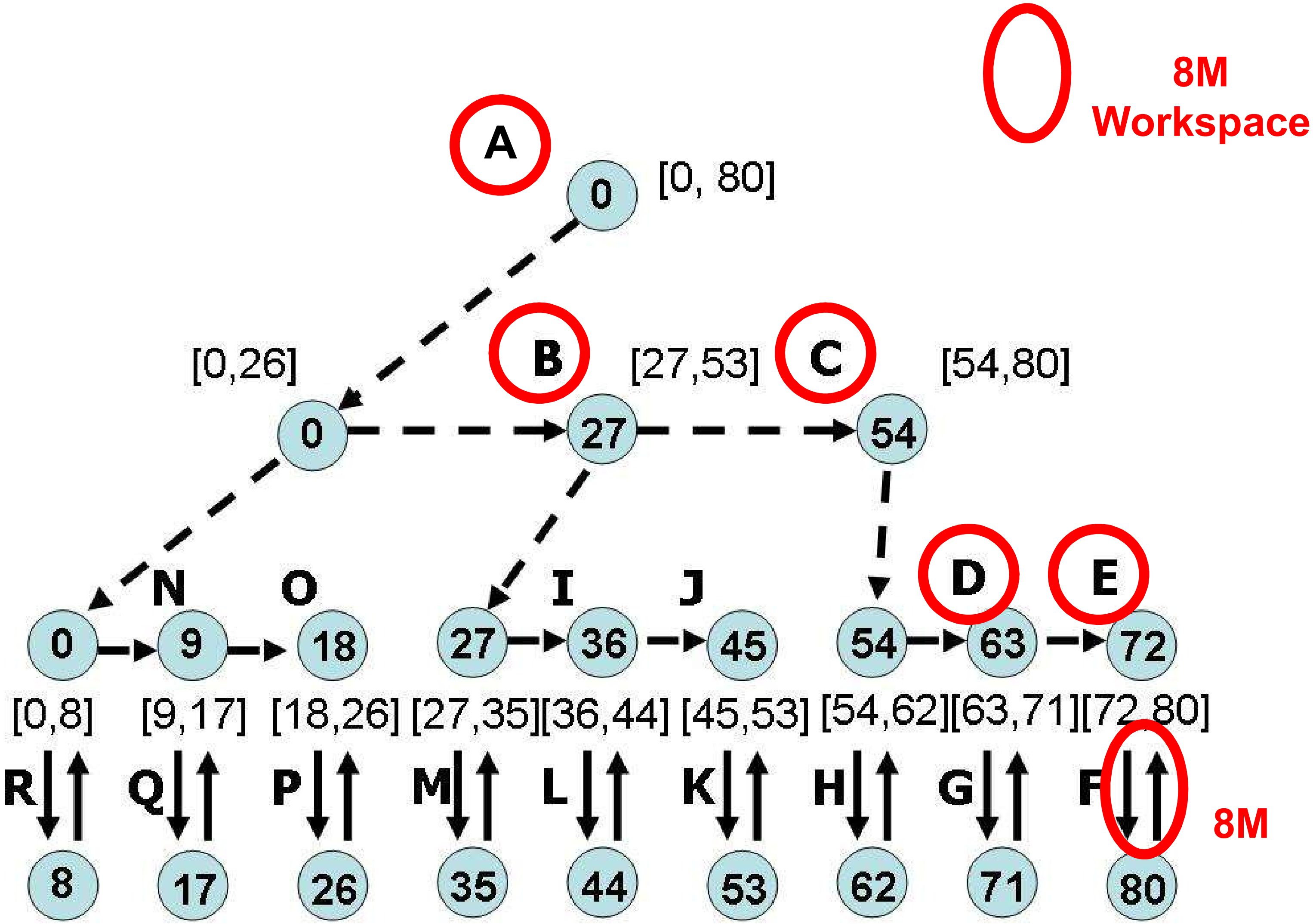
- Generally, if one takes a snapshot every $\sqrt{T}$ step, then we may reduce storage down to $2\sqrt{T}M$ by computing twice the whole filtering distributions.

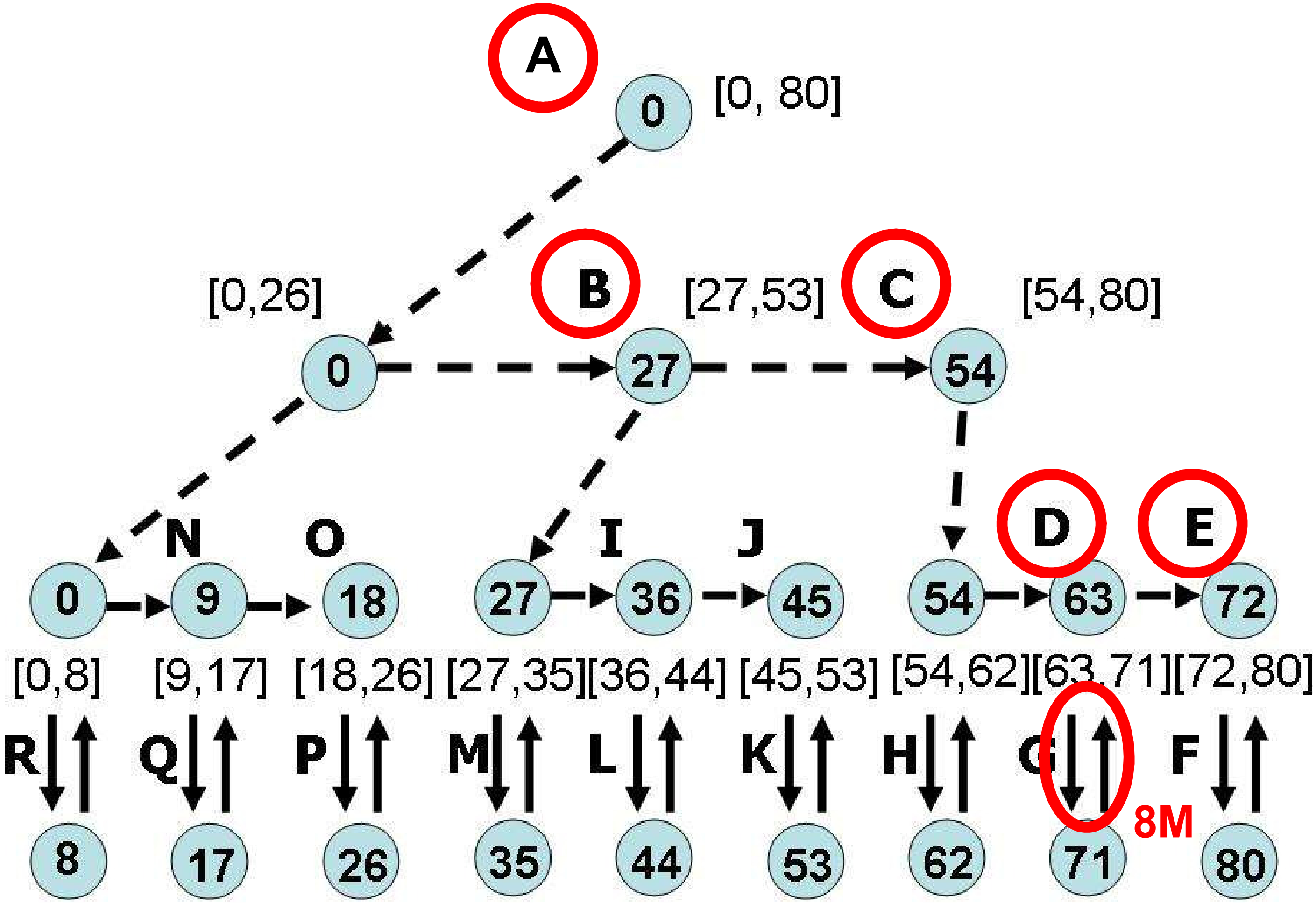
Recursive Recomputation Scheme

- Reduces further storage to O((log T)M) by applying the idea of recomputaiton recursively, at the cost of O(log T) times recomputation of whole filtering distributions
- We explain the method by applying to Smooth the series with T=81. (t=0~80)

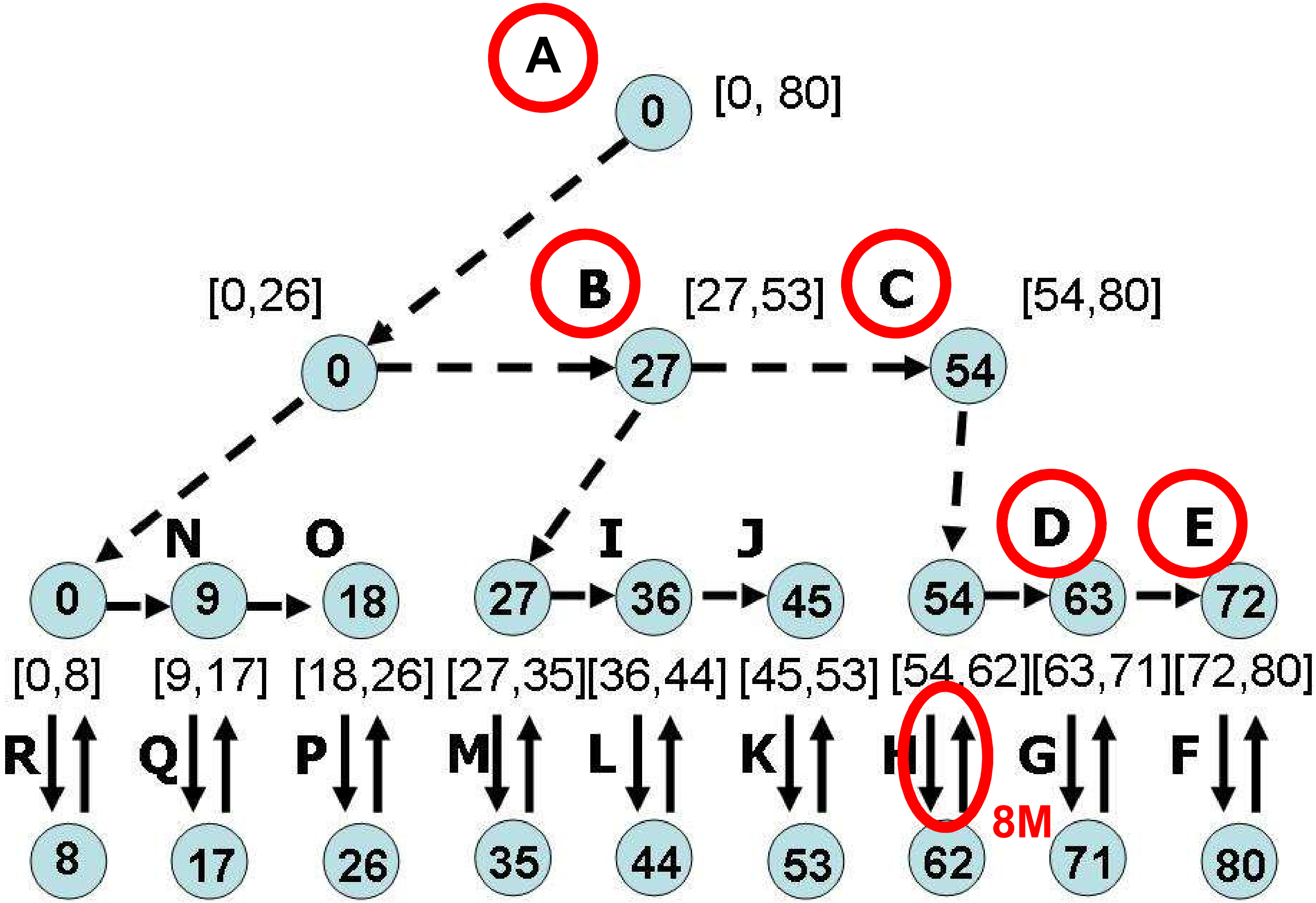

A
0
[0, 80]
[0,26]
0
B
27
[27,53]
C
54
[54,80]
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80

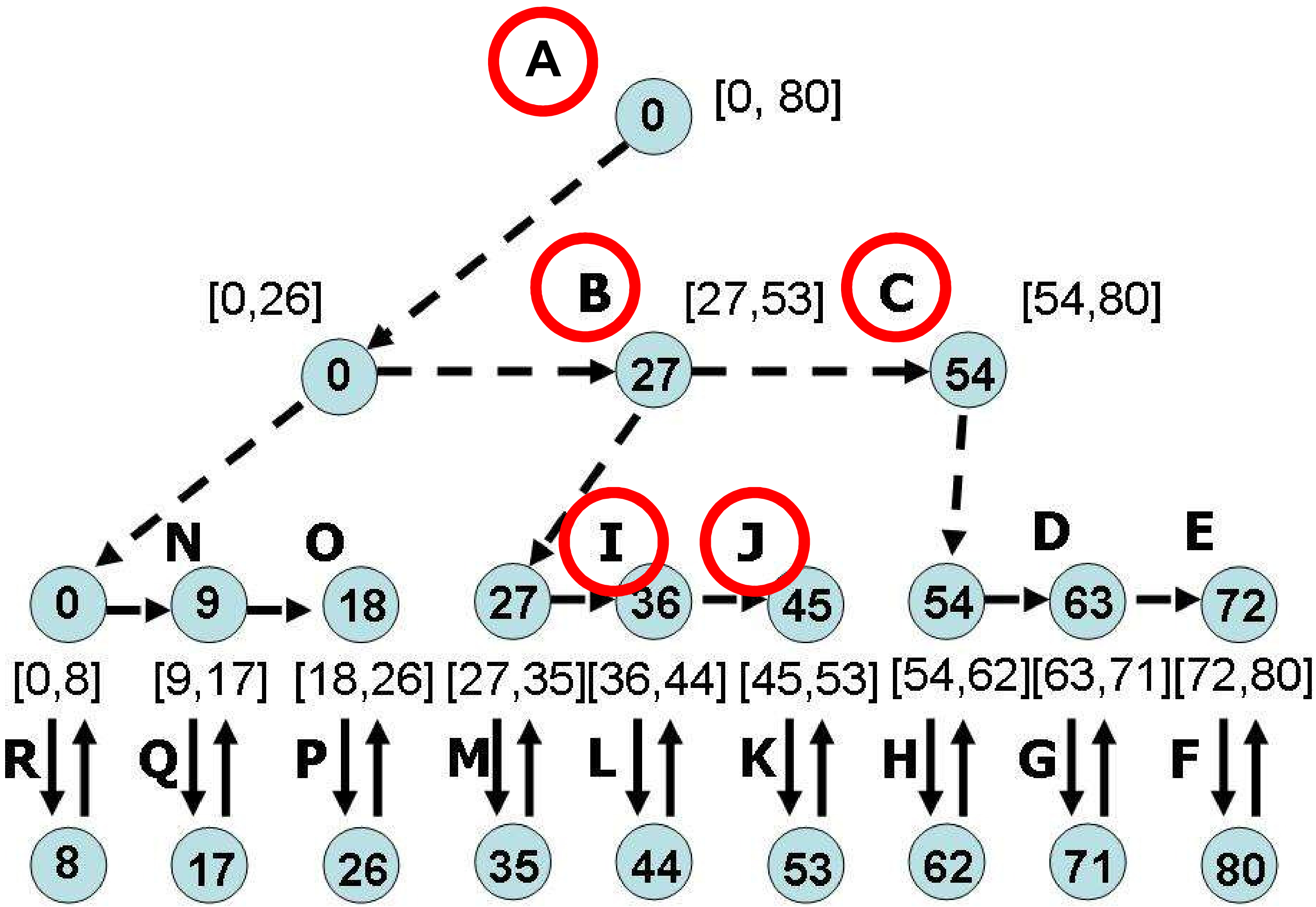

Snapshot
A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80


8M
Workspace
A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80

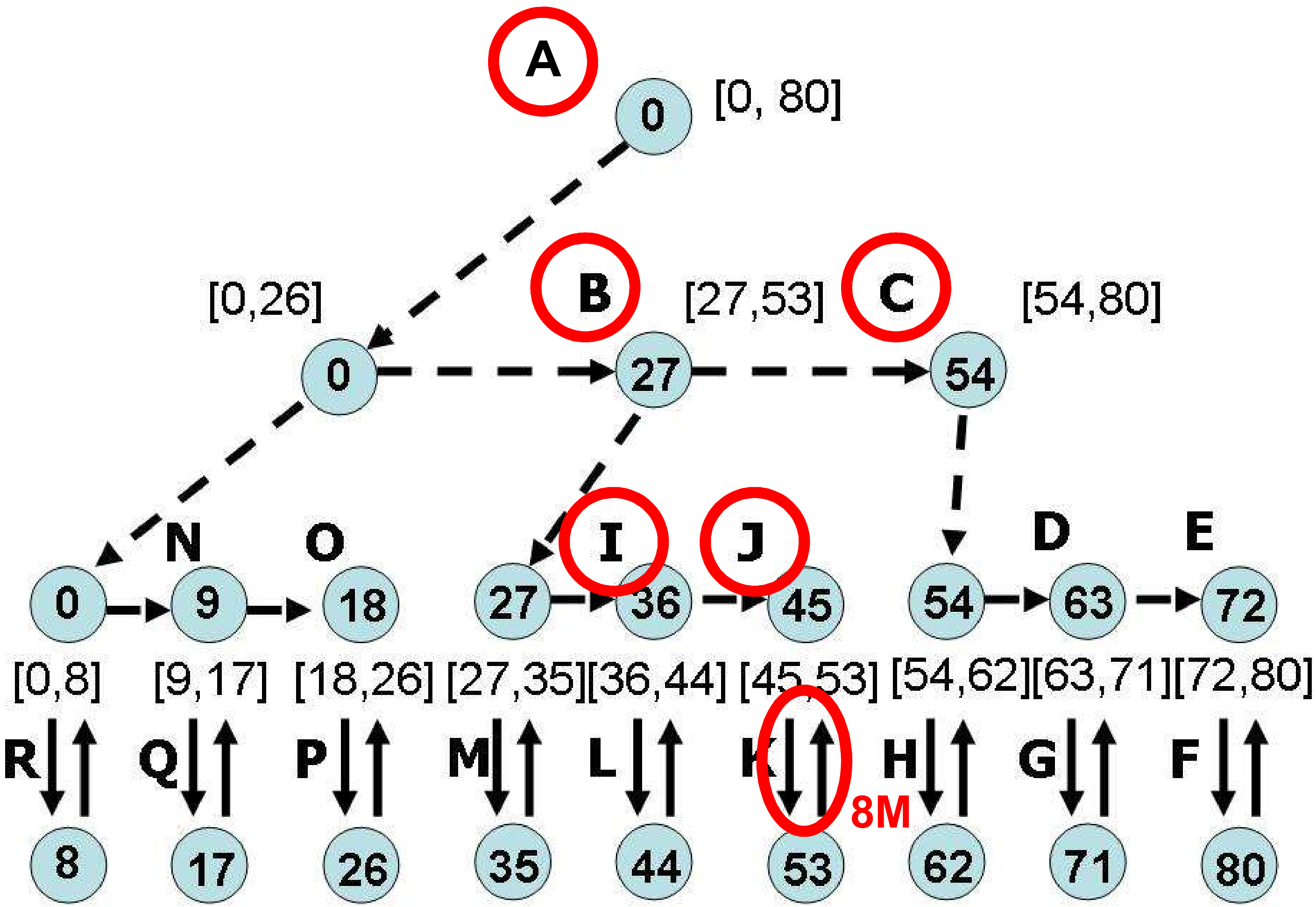


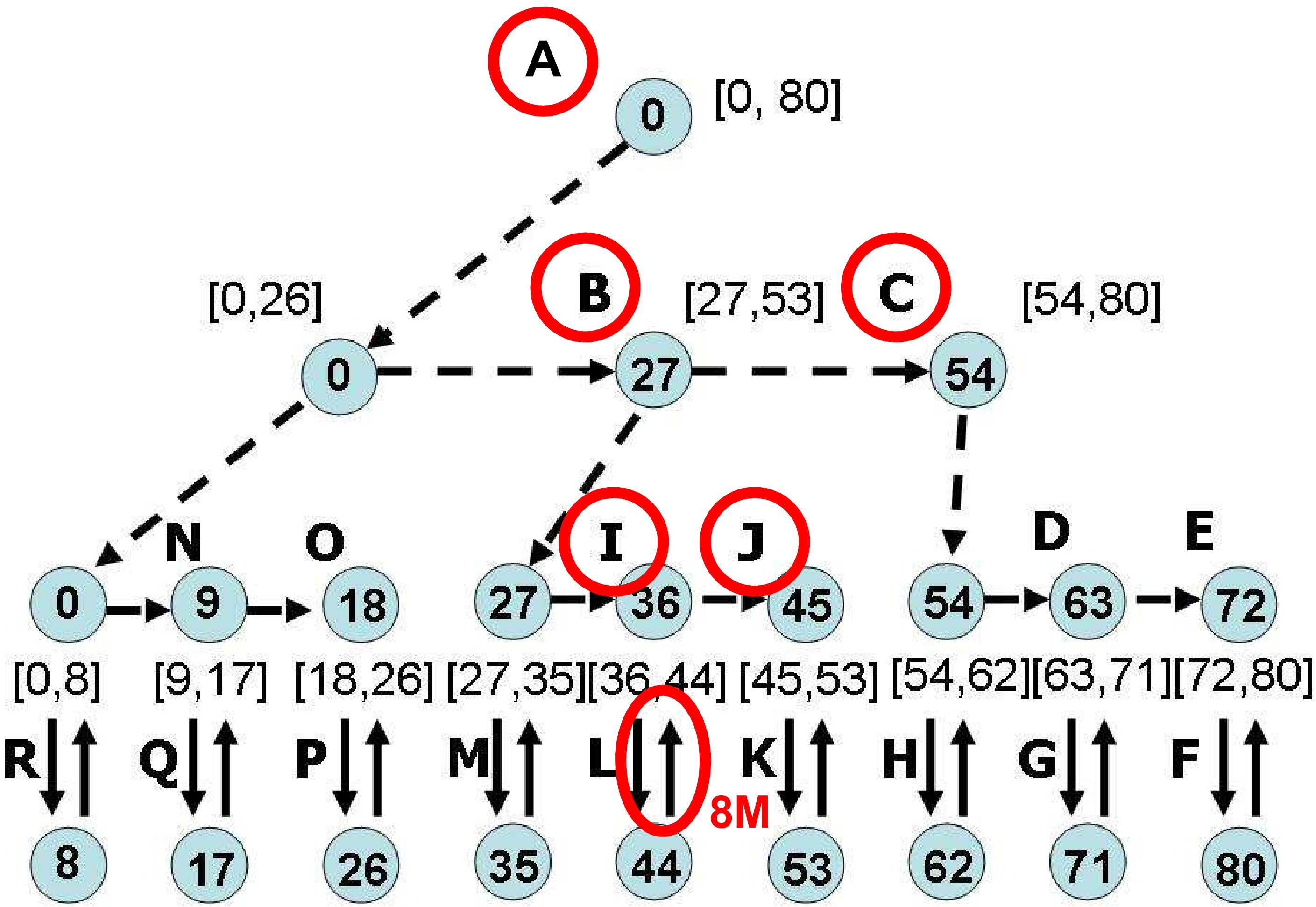
A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80


A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80

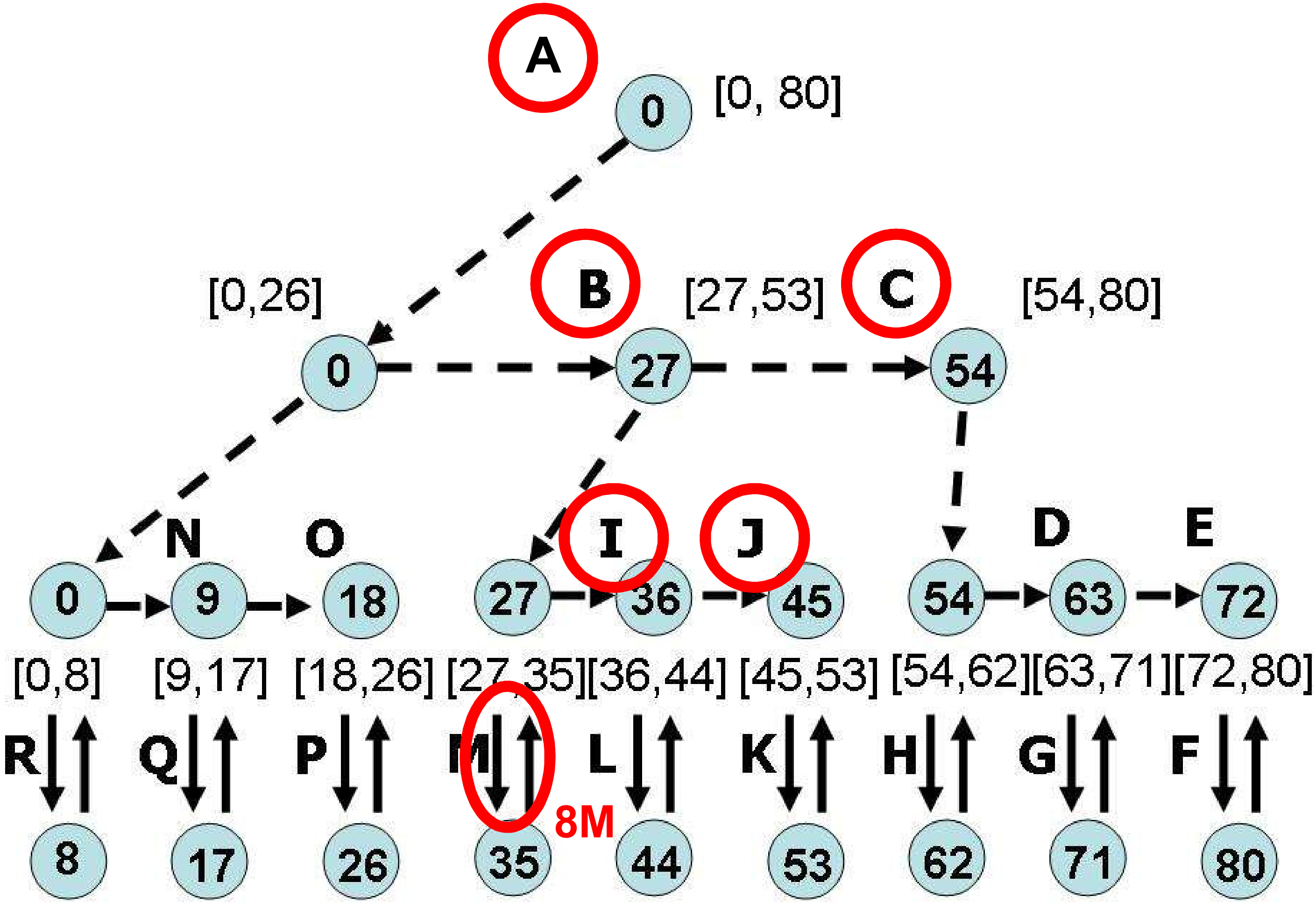


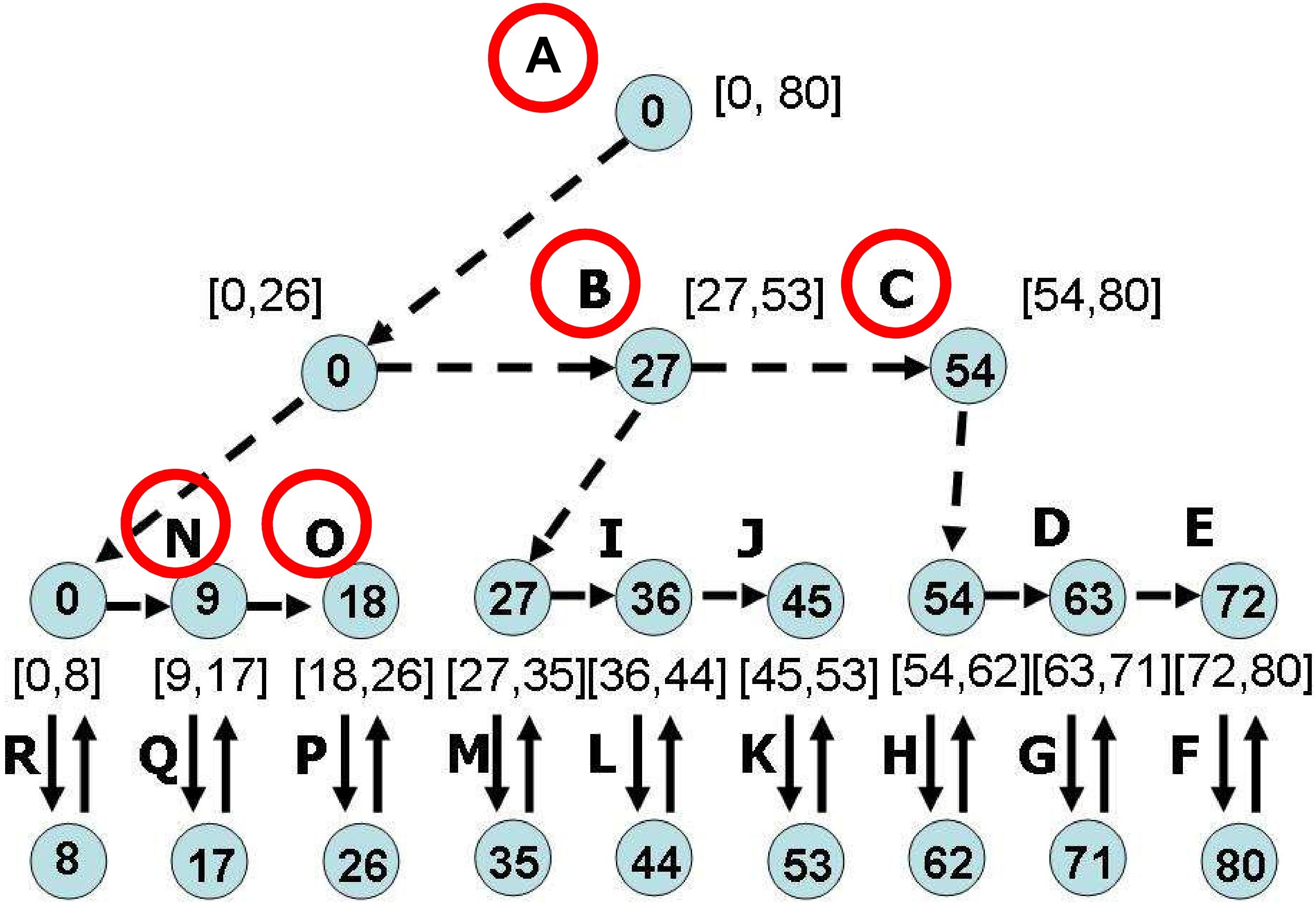
A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80

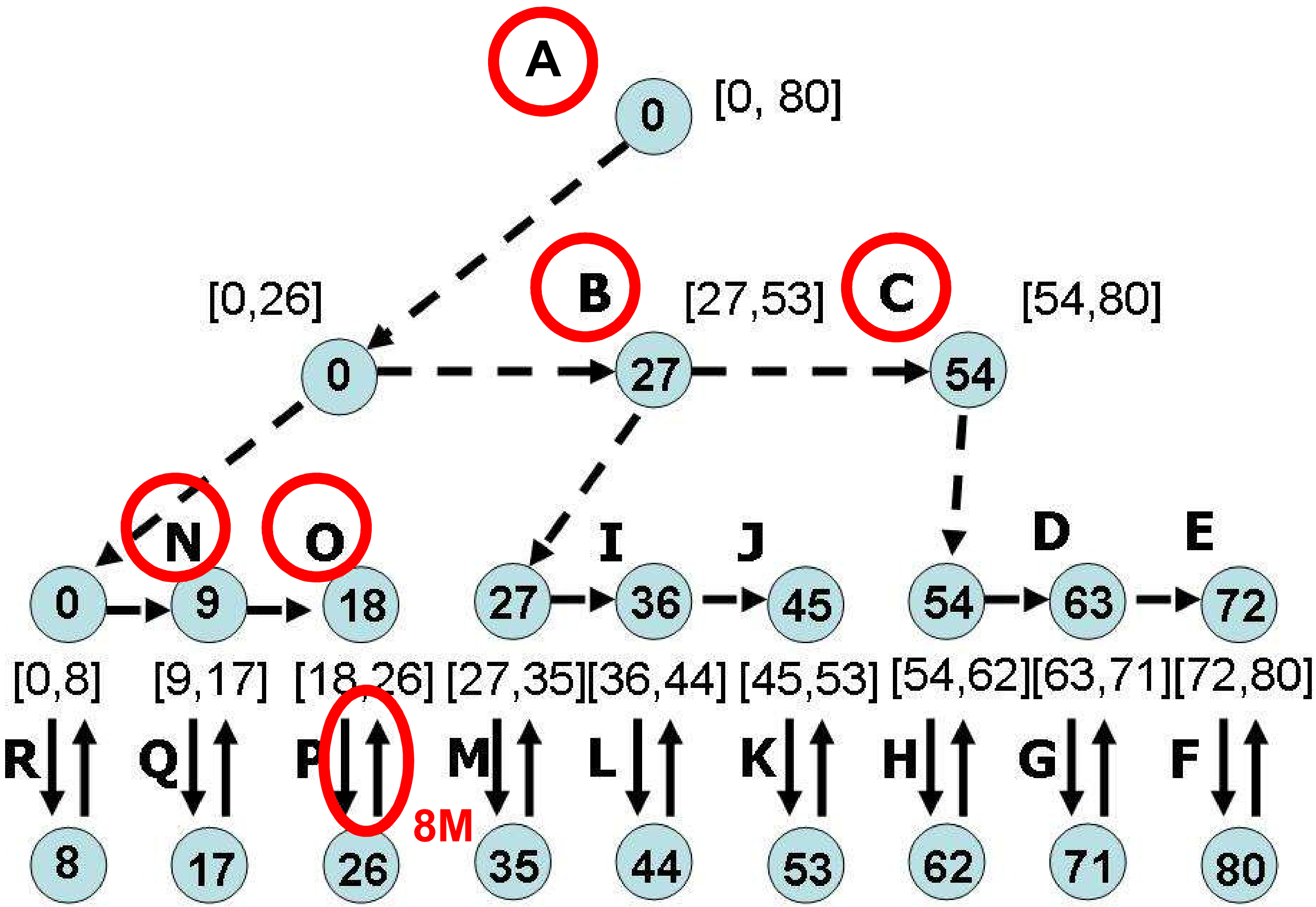

A
B
C
I
J
0
[0, 80]
[0,26]
[27,53]
[54,80]
0
27
54
N
O
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80

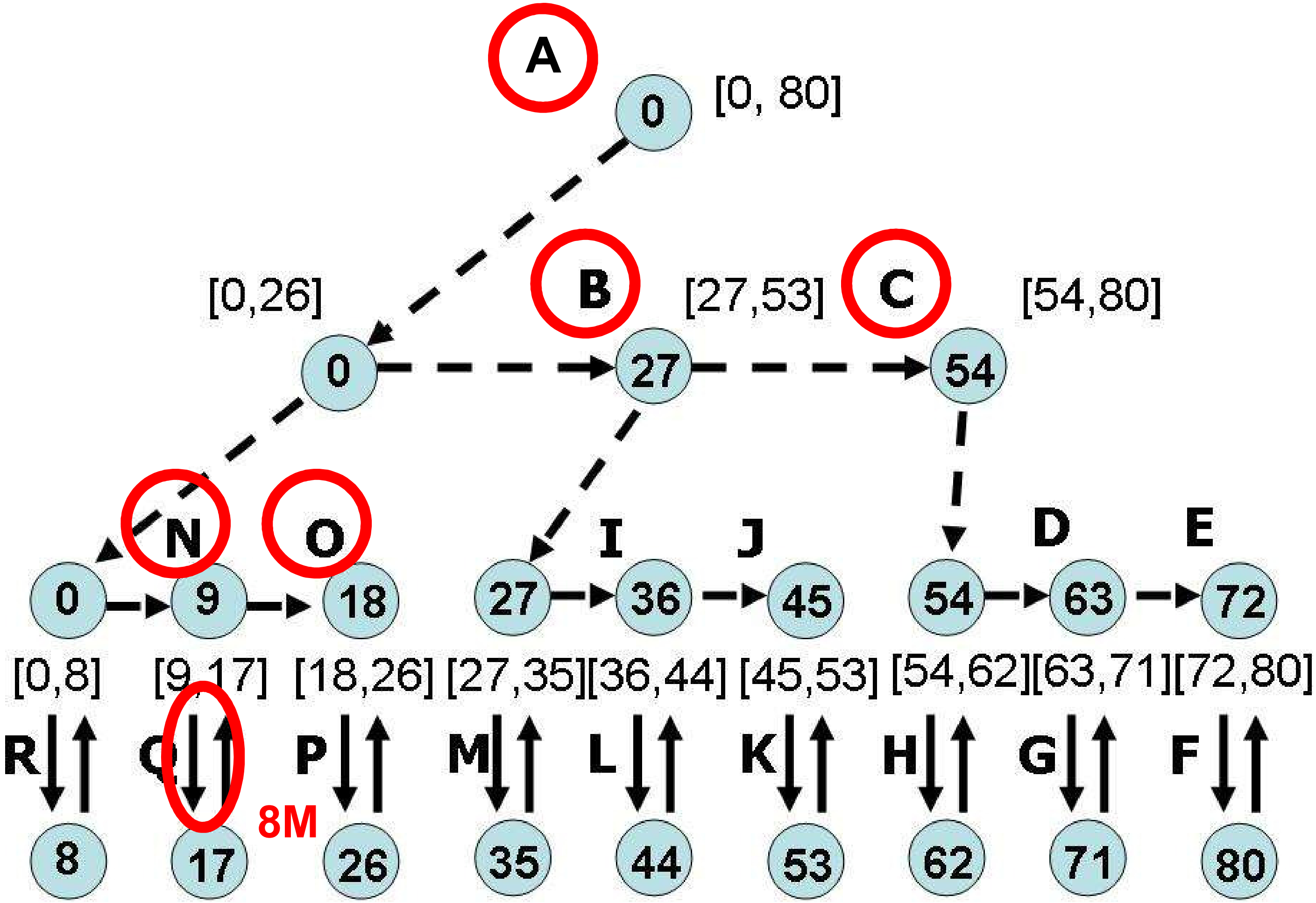


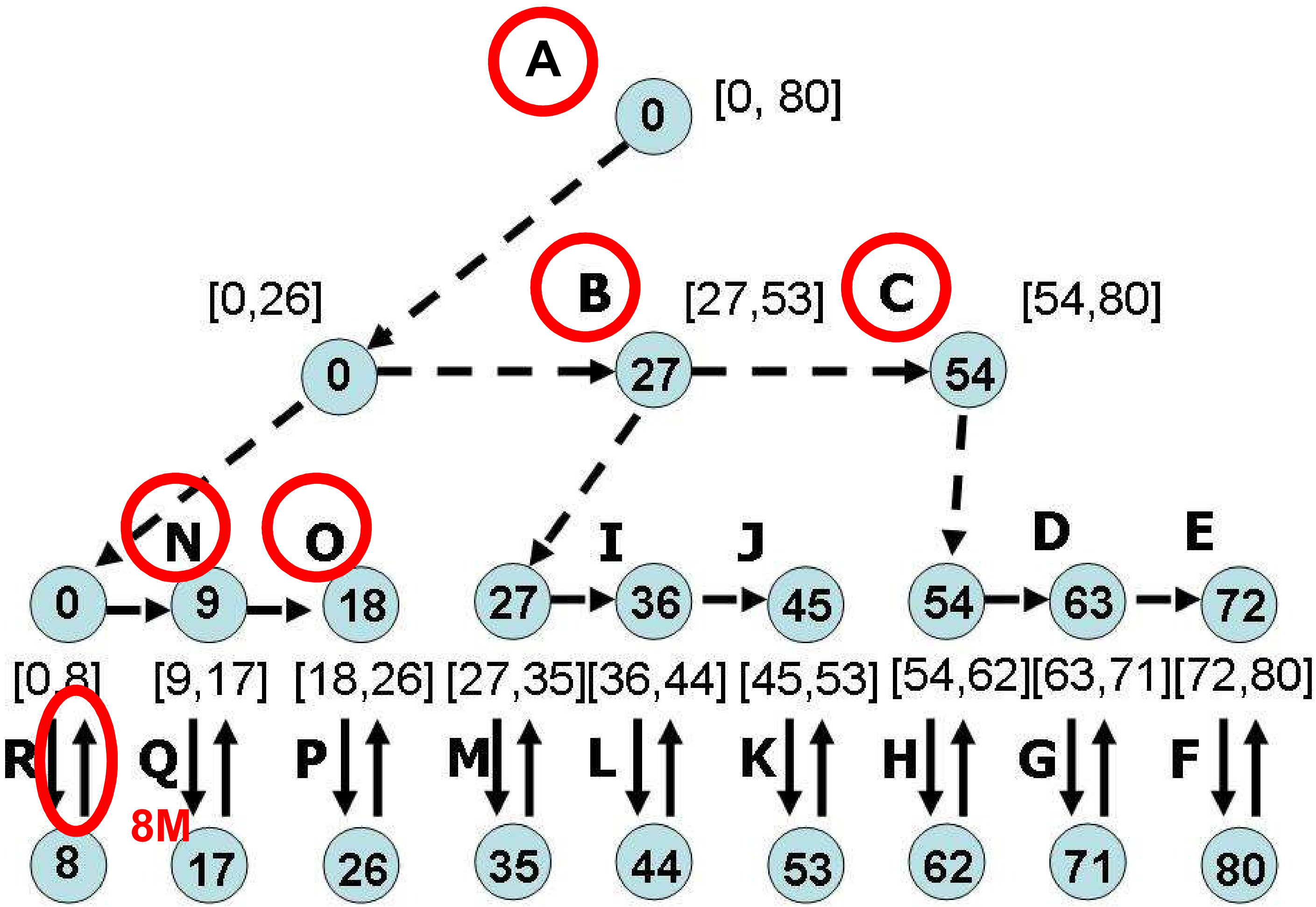
A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80


A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80


A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8
17
26
35
44
53
62
71
80


A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80


A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35]
[36,44]
[45,53]
[54,62]
[63,71]
[72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80


A
0
[0, 80]
[0,26]
B
[27,53]
C
[54,80]
0
27
54
N
O
I
J
D
E
0
9
18
27
36
45
54
63
72
[0,8]
[9,17]
[18,26]
[27,35][36,44]
[45,53]
[54,62][63,71][72,80]
R
Q
P
M
L
K
H
G
F
8M
8
17
26
35
44
53
62
71
80

Recursive Recomputation Scheme

- We are able to compute the smoothing distributions with 5+8 = 13 M storage (instead of 81 or 17).
- The number of whole filtering computation is 3 (or a bit less).

Recursive Recomputation Scheme (Summary)

- We can reduce the space complexity for smoothing from $O(MT)$ to $O(M \log T)$.

  (at the cost of $O(\log T)$ times computation of whole filtering distributions.)

Background (In connection with optimization)

- The idea of saving storage by recomputation was developed in automatic differentiation.
- Specifically, the idea of recursive recomputation was introduced by Griewank in 1992.

Easy Implementation

(Initialization)

- Execute a forward sweep taking snapshots under a certain rule (explained later).
- At the end of the sweep (t=T-1), the set of snapshots snap(T-1) is available.

Easy Implementation

(Reverse sweep with partial forward sweeps)

- If t = 0 then stop
- If not, execute the following procedure.

（We assume that the filtering distribution F(t) has been computed, and the set of snapshots snap(t) is available. We compute F(t-1) and snap(t-1) below.）

Easy Implementation

(To compute F(t-1))

- Execute partial forward sweep from the snapshot which is before t-1 and closest to t-1 to compute F(t-1). On the way, we take snapshots to construct snap(t-1).

Easy Implementation: Rule for snapshots:

- Provide areas snapshot(0), snapshot(1), …, snapshot(K) to store snapshots.
- If k lower bits are zero in the binary representation of t, then we store the snapshot F(t) at snapshot(k).

t=15 (After the forward sweep)

- Snapshot(0): t=(1111)_2 = 15 ←
- Snapshot(1): t=(1110)_2 = 14
- Snapshot(2): t=(1100)_2 = 12
- Snapshot(3): t=(1000)_2 = 8
- Snapshot(*): t=(0000)_2 = 0

t=14

- Snapshot(0):
- Snapshot(1): t=(1110)_2 = 14
- Snapshot(2): t=(1100)_2 = 12
- Snapshot(3): t=(1000)_2 = 8
- Snapshot(*): t=(0000)_2 = 0

t=13 (Recomp. from t=12)

- Snapshot(0): t=(1001)_2 = 13 ⇦
- Snapshot(1):
- Snapshot(2): t=(1100)_2 = 12
- Snapshot(3): t=(1000)_2 = 8
- Snapshot(*): t=(0000)_2 = 0

t=12

- Snapshot(0):
- Snapshot(1):
- Snapshot(2): t=(1100)_2 = 12
- Snapshot(3): t=(1000)_2 = 8
- Snapshot(*): t=(0000)_2 = 0

t=11 (Recomp. from t=8)

- Snapshot(0):
- Snapshot(1):
- Snapshot(2):
- Snapshot(3): t=(1000)_2 = 8
- Snapshot(*): t=(0000)_2 = 0

t=11 (Recomp. from t=8)

- Snapshot(0): t=(1001)_2 =   9
- Snapshot(1):
- Snapshot(2):
- Snapshot(3): t=(1000)_2 =   8
- Snapshot(*): t=(0000)_2 =   0

t=11 (Recomp. from t=8)

- Snapshot(0): t=(1001)_2 =   9
- Snapshot(1): t=(1010)_2 = 10
- Snapshot(2):
- Snapshot(3): t=(1000)_2 =   8
- Snapshot(*): t=(0000)_2 =   0

t=11 (Recomp. from t=8)

- Snapshot(0): t=(1011)_2 = 11
- Snapshot(1): t=(1010)_2 = 10
- Snapshot(2):
- Snapshot(3): t=(1000)_2 = 8
- Snapshot(*): t=(0000)_2 = 0

t=10

- Snapshot(0):
- Snapshot(1): t=(1010)_2 = 10
- Snapshot(2):
- Snapshot(3): t=(1000)_2 = 8
- Snapshot(*): t=(0000)_2 = 0

t=9 (Recomp. from t=8)

- Snapshot(0): t=(1001)_2 = 9
- Snapshot(1):
- Snapshot(2):
- Snapshot(3): t=(1000)_2 = 8
- Snapshot(*): t=(0000)_2 = 0

t=8

- Snapshot(0):
- Snapshot(1):
- Snapshot(2):
- Snapshot(3): t=(1000)_2 = 8 ⇐
- Snapshot(*): t=(0000)_2 = 0

t=7 (Recomp. from t=0)

- Snapshot(0): t=(0111)_2 = 7 ⇐
- Snapshot(1): t=(0110)_2 = 6
- Snapshot(2): t=(0100)_2 = 4
- Snapshot(3):
- Snapshot(*): t=(0000)_2 = 0

Application

- We applied the new method to smooth the Nikkei 225 stock price data with a stochastic volatility model with particle filters.
- We used the path-sampling smoothing algorithm of the particle filter by Kitagawa.

Path-sampling smoother

- Let N be the number of particles.

State Space Model

Innovation of state

$$x_{t+1} = f(x_t) + \varepsilon_t$$

$$y_t = g(x_t) + \eta_t$$

Observation

η_t, ε_t : (Non Gaussian) white noize

Path-sampling Smoother

- Traces the parents-children relation of particles.
- O(N) operations per time step.
- N can be millions (in view of computation time), but the limit of N comes from storage.
- Suffers from degeneracy.

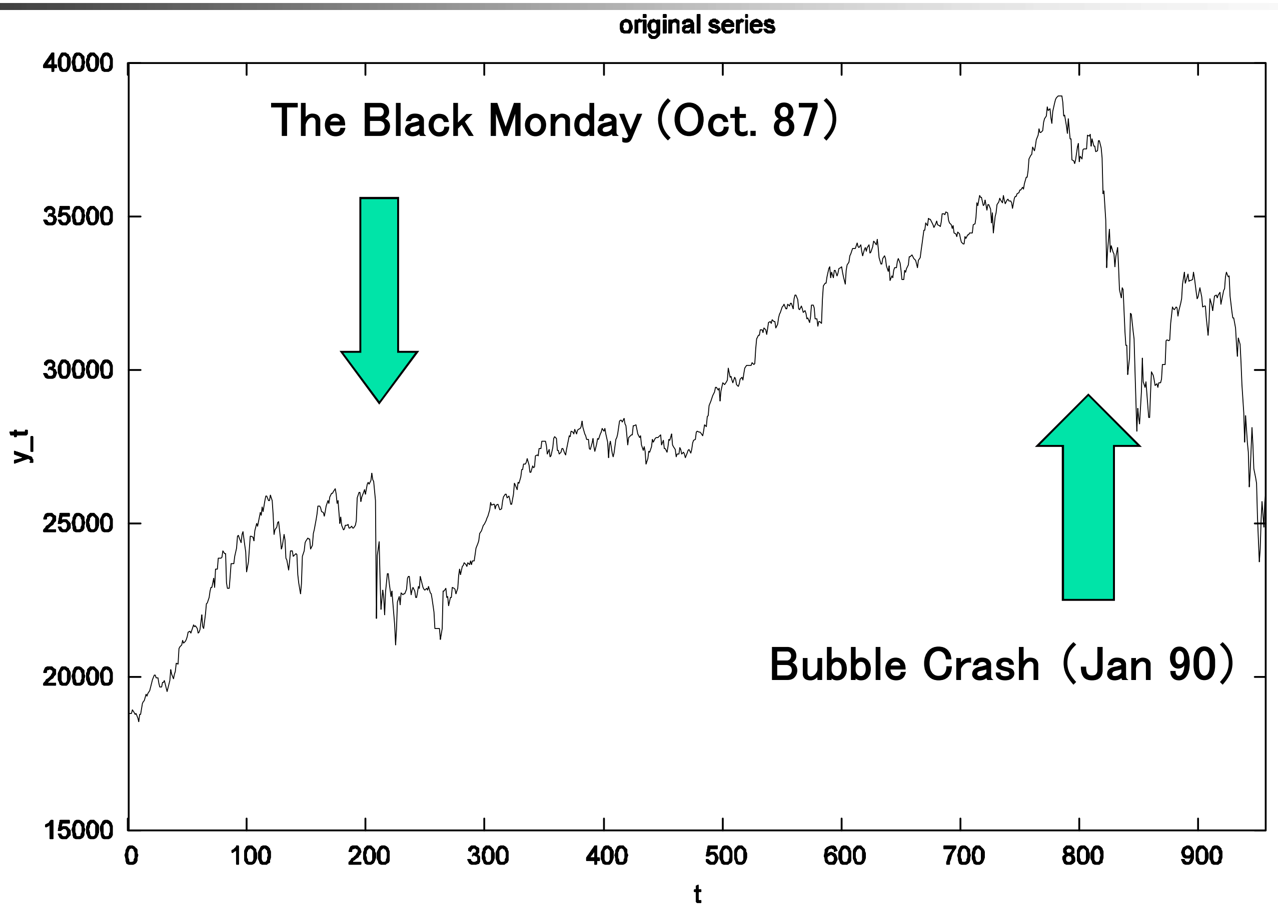
Path-sampling smoother

- What is degeneracy?

  The number of particles representing the smoothing distribution reduces quickly as t is traced back.

  (if you start with numerous and diverse particles, the particles can easily reduces 10 – 20 after a while.)

Nikkei 225 Data (From Jan. 87 to Aug. 90)



Model

Trend

$$x_{t+1} = -2x_t + x_{t-1} + \varepsilon_t,$$

$$y_t = x_t + \eta_t,$$

Stock price

Volatility

$$\varepsilon_t \sim \mathrm{Cauchy}(0, \tau_1)$$

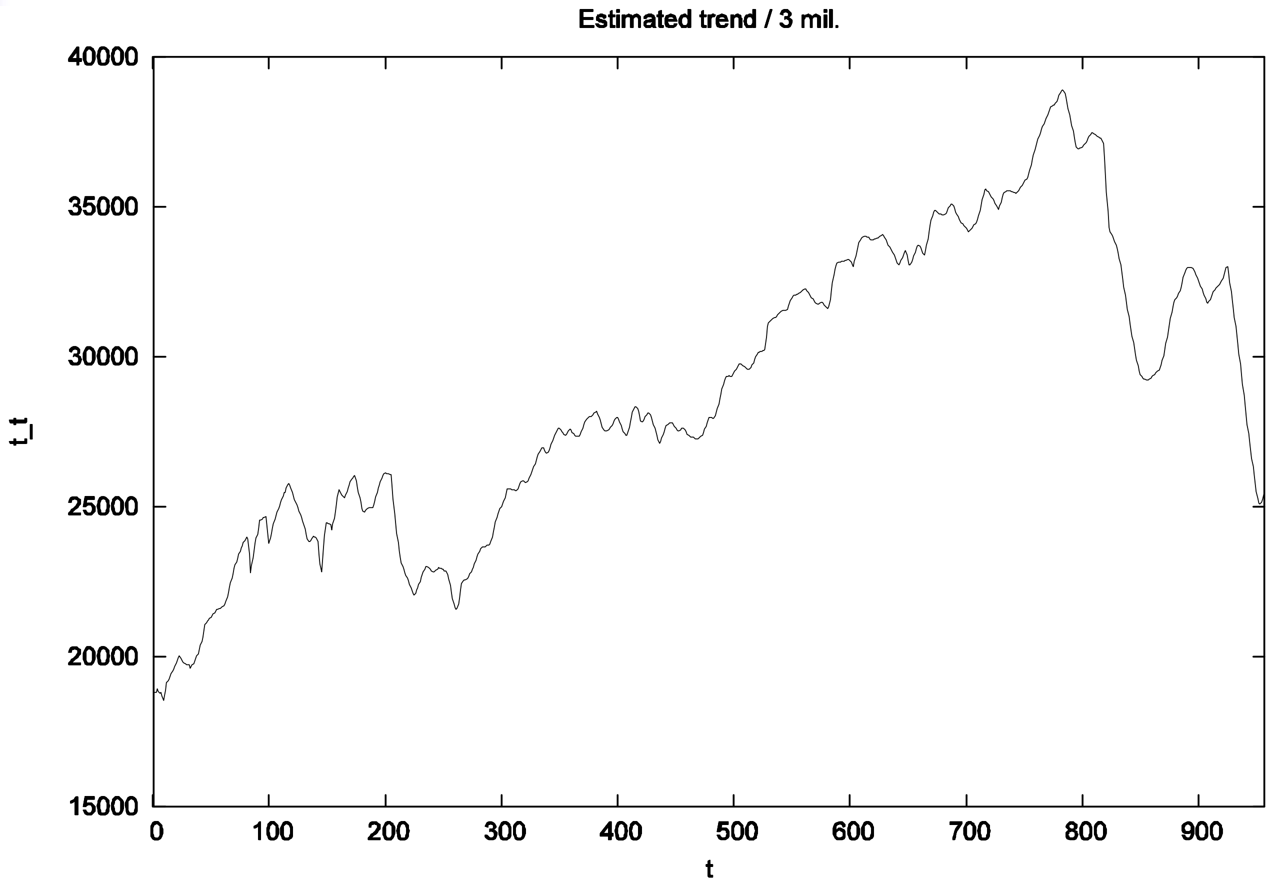
$$\eta_t \sim \mathrm{Normal}(0, \exp(p_t))$$

Model (II)

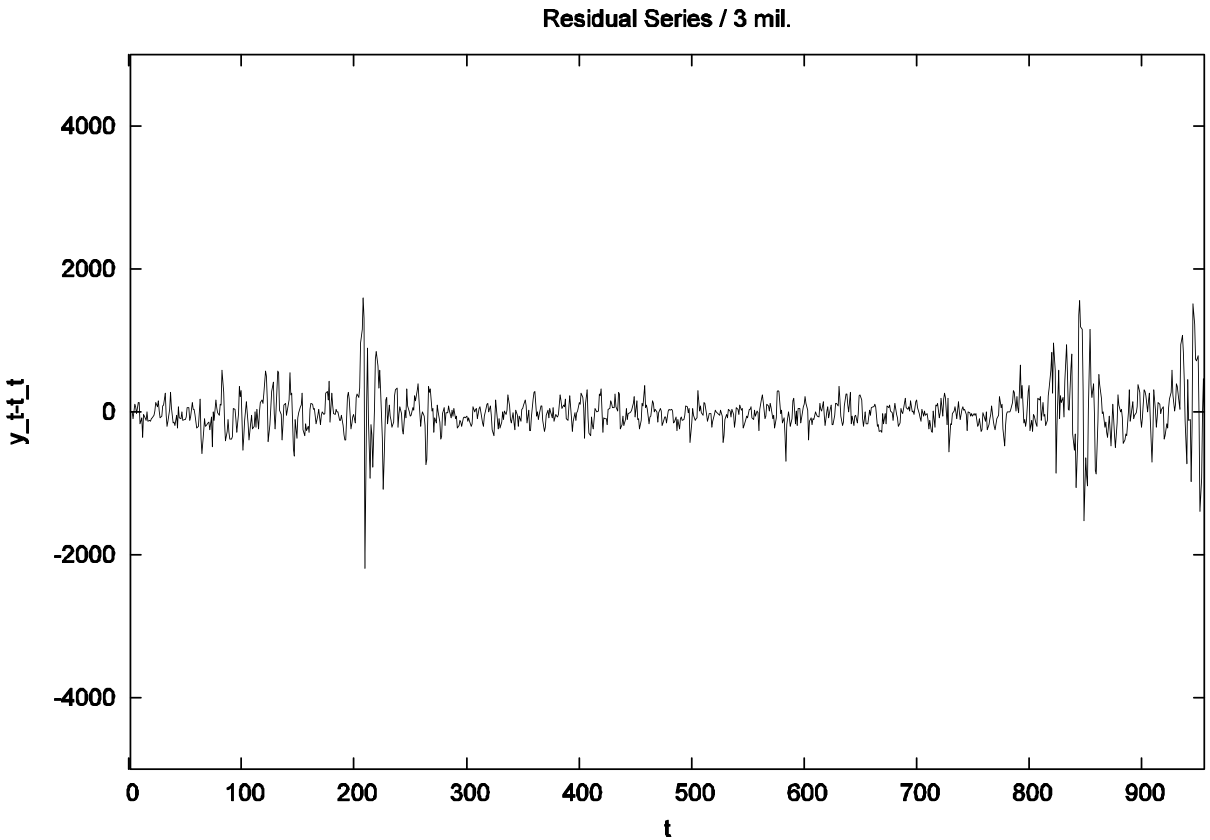
$$p_{t+1} = -2p_t + p_{t-1} + \delta_t,$$

$$\delta_t \sim \mathrm{Normal}(0, \tau_2)$$

Smoothed Trend (median)



Residual (Volatility is time-varying variance of the sequence)

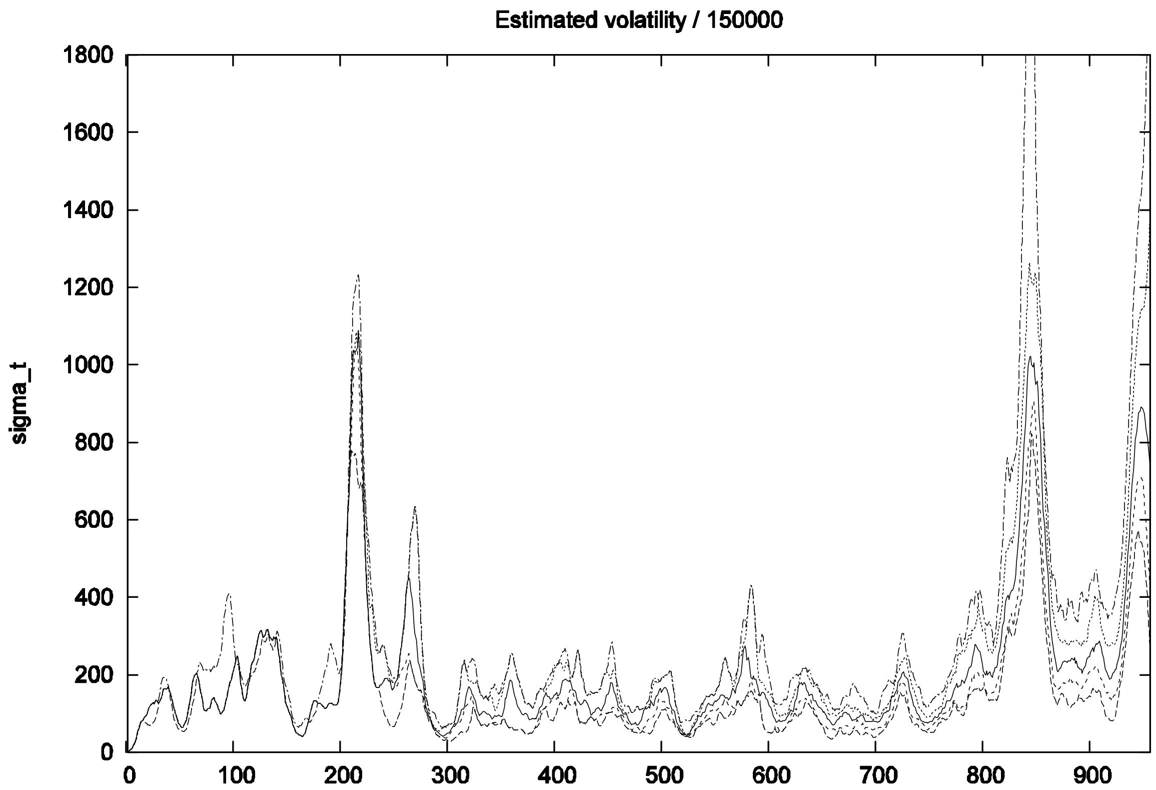


Smoothing with the Particle Filter

- Computer SGI Altix3700 (256CPU,Intel Itanium2 1.3GHz, Main Memory 1920GB)
- 1CPU, 5GB
- 150,000 particles can be used with smoothing by ordinary implementation.

(Comparable with high-end PC environment)

The smoothing distributions of volatility (150,000 particles)

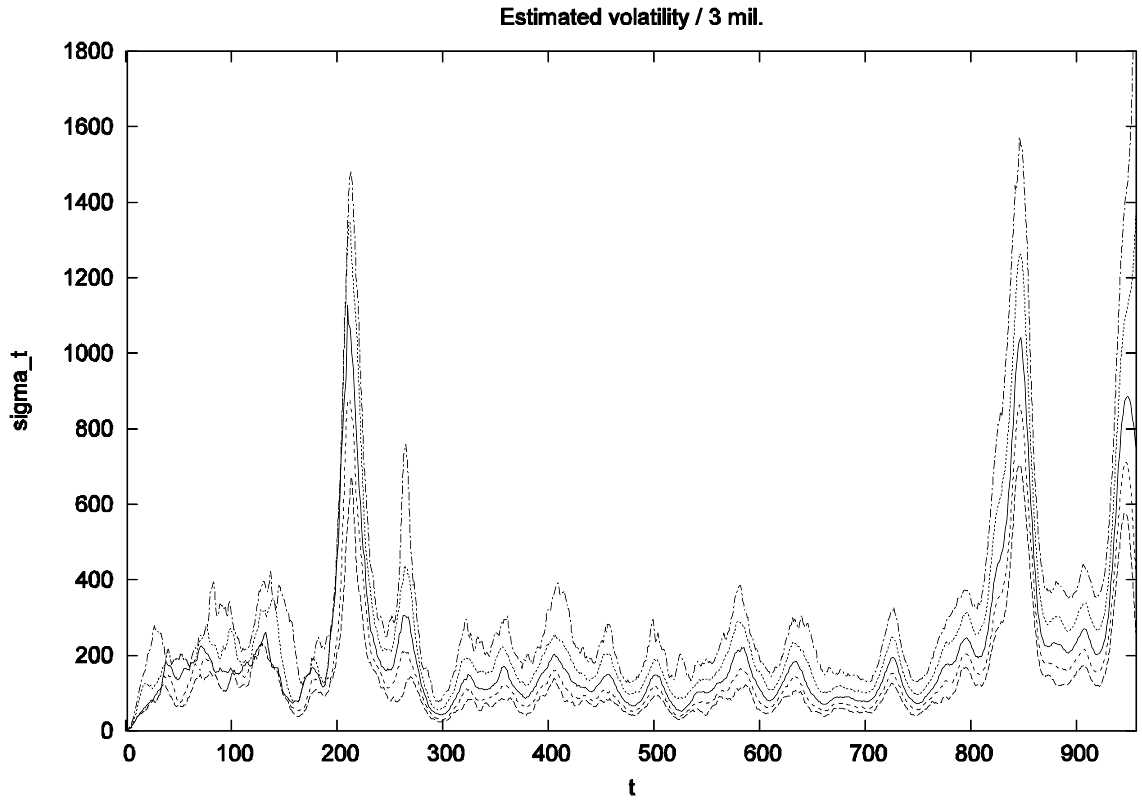


Each curve shows 2.3%, 15.9%, 50%(solid line), 84.1%, 97.7% points.

Smoothing with the particle filter

- 3,000,000 particles (20 times more than the ordinary implementation) can be used for smoothing with the recursive recomputation scheme.

Smoothing distributions of volatility (3 mil particles)



Each curve shows 2.3%, 15.9%, 50%(solid line), 84.1%, 97.7% points.

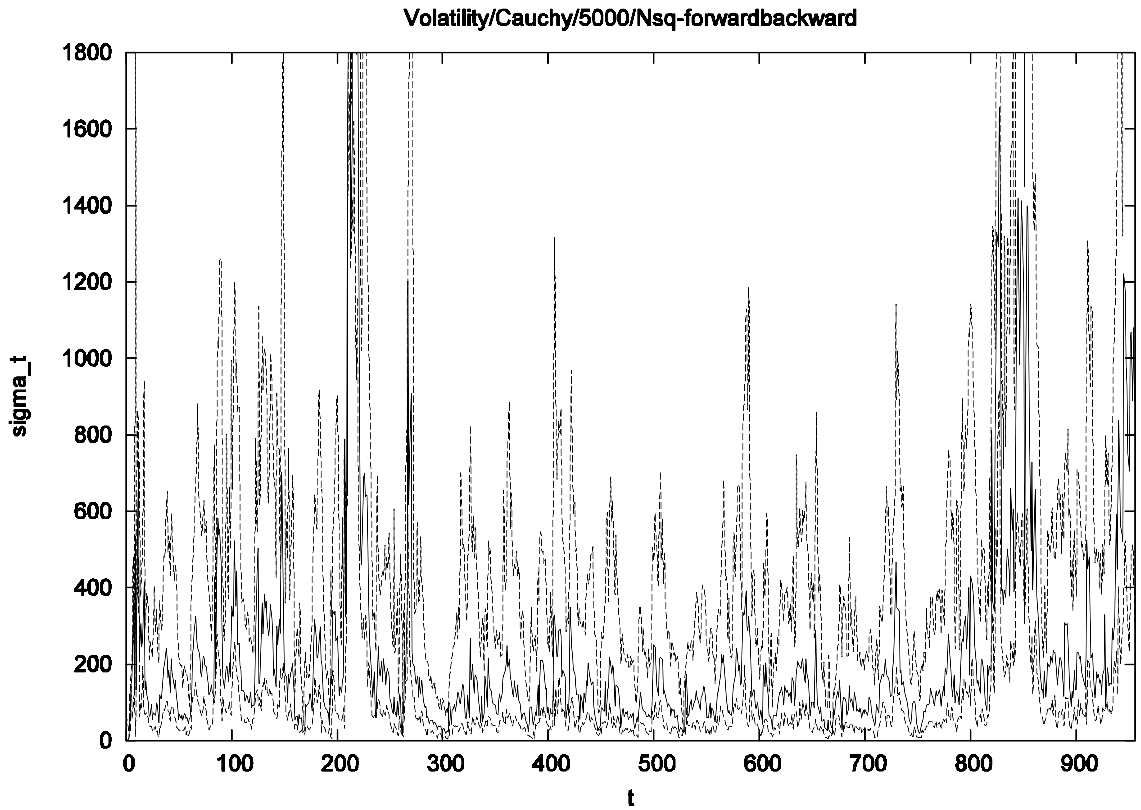
Comparison with other "standard particle smoothers"

- Forward-backward smoother
- Two-filter smoother
- These methods require $O(N^2)$ operations per time step. We cannot increase N more than 5000. (Since N is modest, there is no need for saving storage.)

Comparison with other "standard particle smoothers"

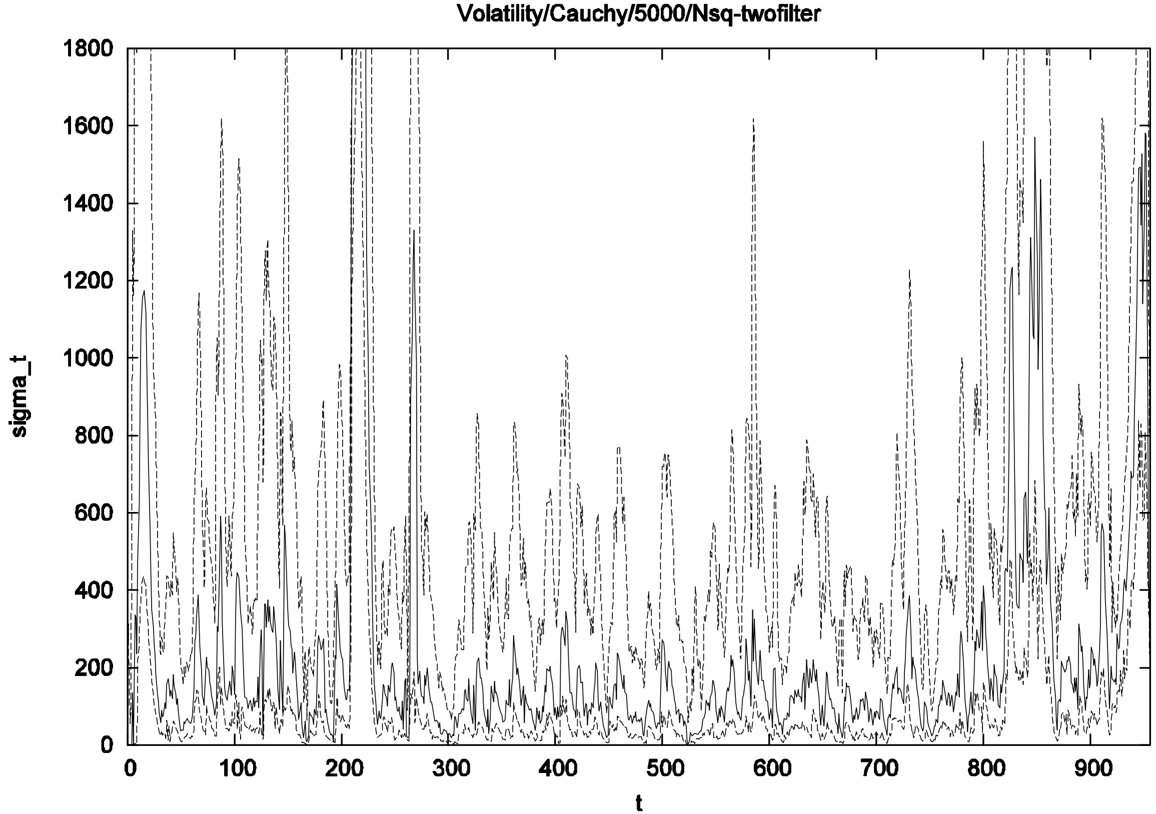
- Path-sampling Method is O(N) per step but suffers from degeneracy.
- In the following we compare the smoothing distributions computed in 7hours with N= 5000 (for the two $O(N^2)$ methods) and N=3000000 (for path-sampling method).

Forward-Backward Method



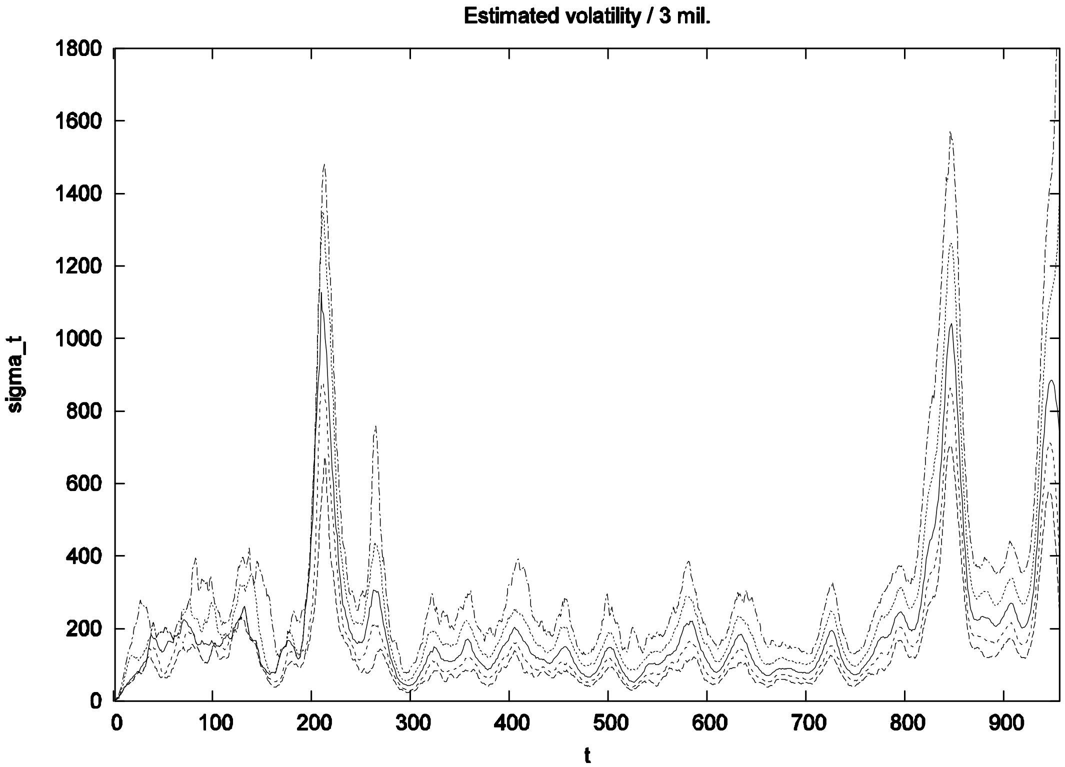
Each curve shows 2.3%, 50%(solid line), 97.7% points.

2 Filter Formulae



Each curve shows 2.3%, 50%(solid line), 97.7% points.

The New Method



Each curve shows 2.3%, 15.9%, 50%(solid line), 84.1%, 97.7% points.

Conclusion

- The path-sampling smoother with numerous particles performs the best. Using numerous particles helps.
- It is possible to extend the technique to fixed-lag smoothers.