Two Pass Strategy

Algorithm 1: Two Pass Strategy

```
Activated ← φ;
for Every event (k, v_k) do
    v(k) = v_k;
    for ∀j ∈ fanout of k do
        Update the value of j;
        Activated ← Activated ∪ {j};
    end
end
for j ∈ Activated do
    begin
        v′ = evaluate(j);
        if v′ ≠ lsv(j) then
            add event (j, v′) to the event list at time t + d(j);
            lsv(j) = v′;
        end
    end
end
```

One Pass Strategy

Algorithm 2: One Pass Strategy

```
for Every event (k, v_k) do
    v(k) = v_k;
    for ∀j ∈ fanout of k do
        Update the value of j;
        Instead of putting j to activated set, process it here;
        begin
            v' = evaluate(j);
            if v' ≠ lsv(j) then
                add event (j, v') to the event list at time t + d(j);
                lsv(j) = v';
            end
        end
    end
end
```

One Pass Strategy With Zero-Width Spikes

Algorithm 3: One Pass Strategy With Zero-Width Spikes

```
for Every event (k, v_k) do
    v(k) = v_k;
    for ∀j ∈ fanout of k do
        Update the value of j;
        Instead of putting j to activated set, process it here;
        begin
            v' = evaluate(j);
            if v' ≠ lsv(j) then
                t' = t + d(j);
                if t' = lst(j) then
                    cancel event (j, lsv(j)) at time t';
                end
                add event (j, v') to the event list at time t + d(j);
                lsv(j) = v';
                lst(j) = t';
            end
        end
    end
end
```