Heuristic Scheduling: Forced-directed Scheduling

Dr. Chandan Karfa

Department of Computer Science and Engineering

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

Force-directed scheduling

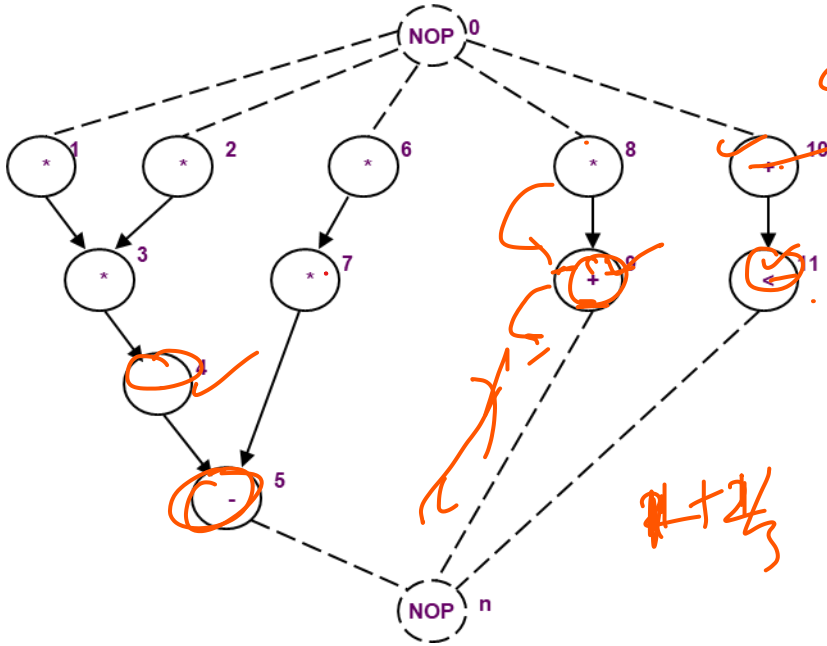
- Heuristic scheduling methods [by Paulin and Knight]:
  - Min *latency* subject to *resource bound*
    - Force-directed scheduling for MLRC
  - Min *resource* subject to *latency bound*
    - Force-directed scheduling for MRLC
- What is Force?

Force

- Mechanical analogy
- The force extend by an elastic spring is proportional to its displacement of its end points.
- F = K . X
  - X is the displacement
  - k is elastic constant
- Can we co-relate the selection of operation to be scheduled in a given time step with force?
- What is the force in scheduling context?

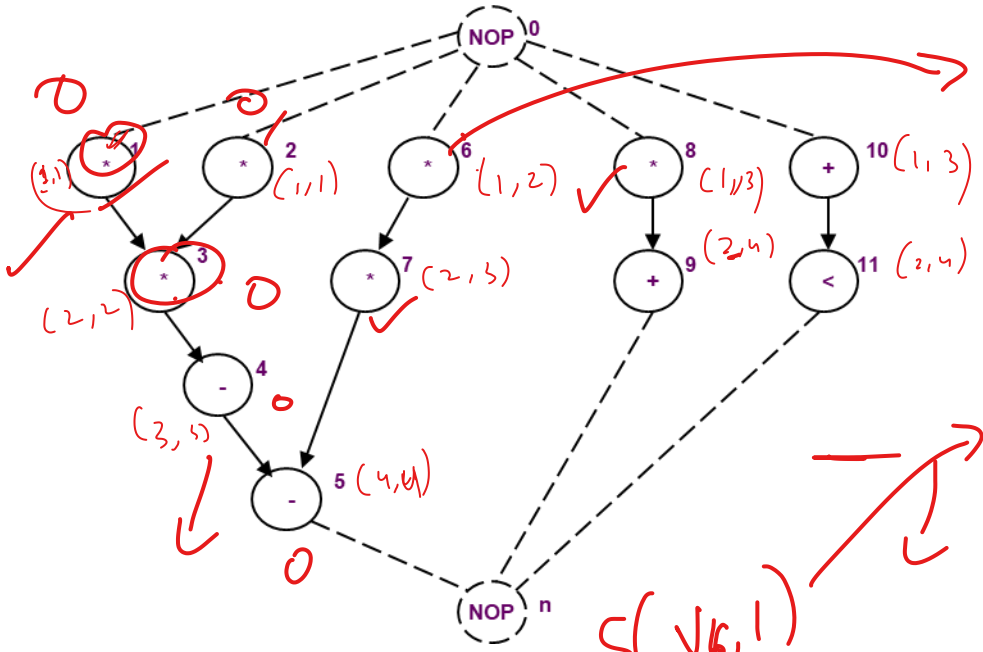
Force-directed scheduling

- **Operation *interval*:**
  - Mobility plus **one** (μ_i **+1**)
  - Computed by ASAP and ALAP scheduling [t^S , t^L]
- **Operation *probability* p_i *(l):***
  - Probability of executing in a given step
  - *1/ (μ_i + 1)* inside interval; *0* elsewhere

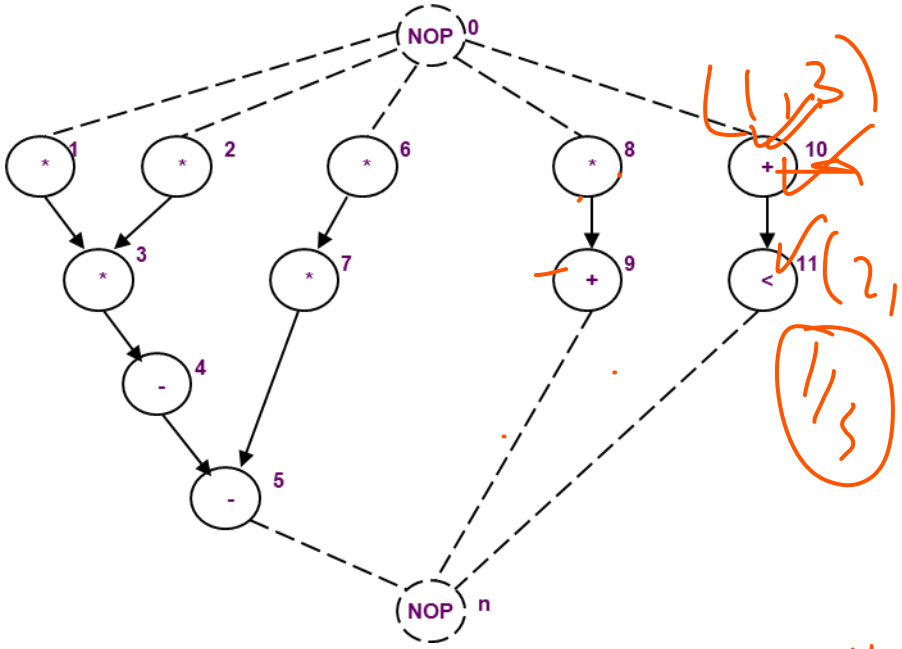


Force-directed scheduling

- ***Operation-type* distribution q_k *(l):***
  - Sum of the operation probabilities for each type



Example



- *v1* has zero mobility. p1(l) = 1, p1(2) = p1(3) = pl(4) = 0.
- p2(1) = 1, p2(2) = p2(3) = p2(4) = 0
- *v6* time frame is [1, 2]. p6(l) = p6(2) = 0.5 and p6(3) = p6(4) = 0.
- Operation *v8* timeframe is [1, 3]. Hence p8(l) = p8(2) = p8(3) = 0.3 and p8(4) = 0.

- Distribution graphs for multiplier and ALU

Analogy with Force

- Assignment of an operation to a control step corresponds to changing its probability.
  - Probability is 1 in that step and 0 elsewhere once the assignment has been done.
- Change in probability == displacement of a spring
- Type distribution $q_i(l)$ == elastic constant
- Force == concurrency of operations of a given type
- $F = K . X$

Force-directed scheduling

- Force is as *priority* function in during operation selection
- Force is related to concurrency:
  - The larger the force, the larger the concurrency
- Two types of force
  - Self force (relating operation to a time step)
  - Predecessor/successor force (related to dependencies)

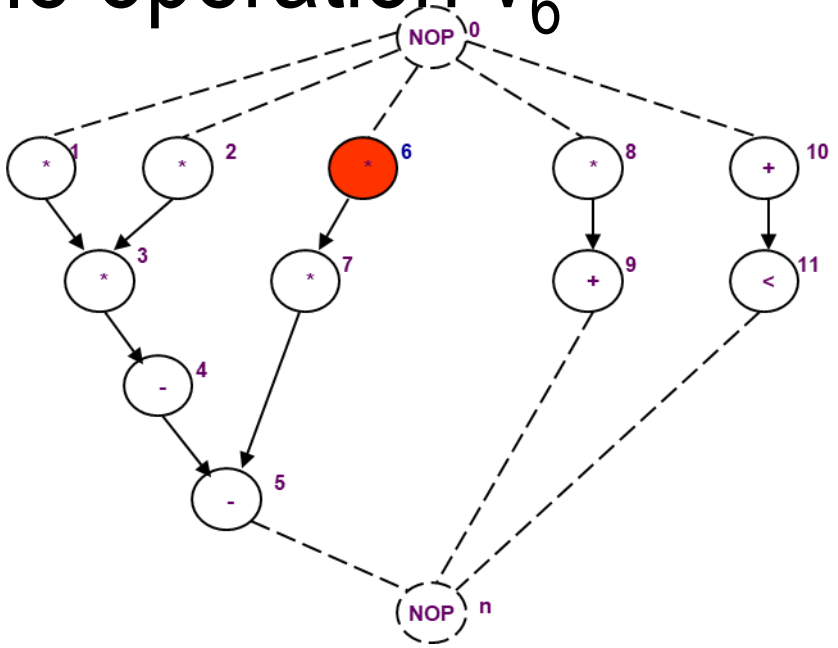
Self Force

- Consider an operation v_i of type k ($\tau(vi) = k$) when schedule in time step l, where $l = [t_i^s, t_i^l]$

$$\text{self-force}(i, l) = \sum_{m=t_i^s}^{t_i^L} q_k(m)(\delta_{lm} - p_i(m))$$

- The self force relating that operation to a step *m in* $[t^s, t^l]$ is equal to the type distribution $q_k(m)$ times the variation in probability (1 - $p_i(m)$),
- δ_{lm} denotes a Kronecker delta function (0 or 1)

Example
Schedule operation v_6

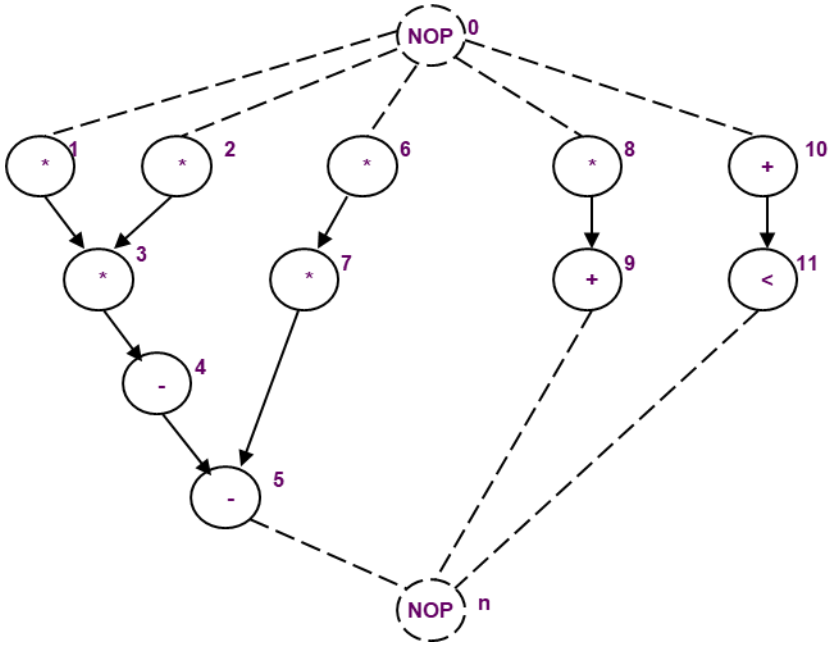


Operation v_6 can be scheduled in step 1 or step 2

Example: operation v_6

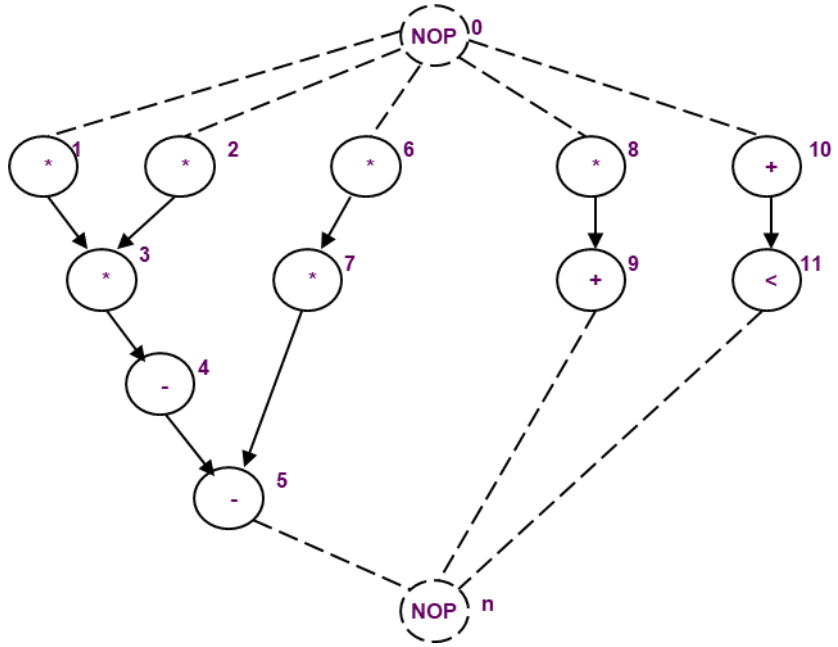
- Op v_6 can be scheduled in the first two steps

  p (1) = 0.5; p (2) = 0.5; p (3) = 0; p (4) = 0
- Distribution: q (1) = 2.8; q (2) = 2.3
- **Assign v_6 to step 1:**
  - variation in probability 1 – 0.5 = 0.5 for step 1
  - variation in probability 0 – 0.5 = -0.5 for step 2
- Self-force: 2.8 * 0.5 – 2.3 * 0.5 = + 0.25
- No successor force since it does not impact the schedule of node v_7



Example: operation v_6

- Assign v_6 to step 2:
  - variation in probability 0 – 0.5 = -0.5 for step 1
  - variation in probability 1 – 0.5 = 0.5 for step 2
- Self-force: - 2.8 * 0.5 + 2.3 * 0.5 = - 0.25
- Successor-force?
  - *Schedule of v7 is now impacted.*



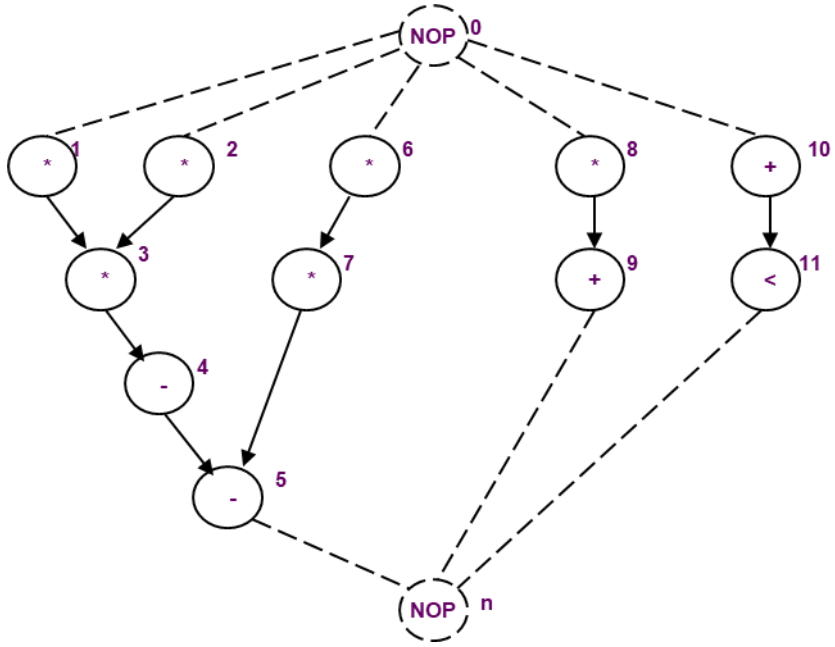
Predecessor/successor-force

- Fixing an operation timeframe restricts timeframe of predecessors/successors
- Delaying an operation implies delaying its successors

- How to calculate predecessor/successor force?

Example: operation v_6

- Assign v_6 to step 2:
  - variation in probability 0 – 0.5 = -0.5 for step 1
  - variation in probability 1 – 0.5 = 0.5 for step 2
- Self-force: - 2.8 * 0.5 + 2.3 * 0.5 = - 0.25
- Successor-force:
  - Operation v_7 assigned to step 3
  - Succ. force is 2.3 (0- 0.5) + 0.8 (1 – 0.5) = - .75
- Total force = self force + successor force = -1



Example: operation v_6

- Total force in step 1 = + 0.25
- Total force in step 2 = -1
- Conclusion:
  - Least force is for step 2
  - Assigning v_6 to step 1 improves concurrency
  - Assigning v_6 to step 2 reduces concurrency – less resource

Predecessor/successor-force

- Can be computed by evaluating the variation on the self forces of the predecessors due to restriction of their time frame.

$$\text{self-force}(i,l) = \sum_{m=t_i^S}^{t_i^L} q_k(m)(\delta_{lm} - p_i(m))$$

Can be rewritten as

$$\text{self-force}(i,l) = q_k(l) - \frac{1}{\mu_i + 1} \sum_{m=t_i^S}^{t_i^L} q_k(m)$$

Predecessor/successor-force

Let $[t_i^S, t_i^L]$ be the initial time frame and $[\tilde{t}_i^S, \tilde{t}_i^L]$ be the reduced one.

$$\text{ps-force}(i,l) = \frac{1}{\tilde{\mu}_i + 1} \sum_{m=t_i^S}^{\tilde{t}_i^L} q_k(m) - \frac{1}{\mu_i + 1} \sum_{m=t_i^S}^{t_i^L} q_k(m)$$

$$\text{self-force}(i,l) = q_k(l) - \frac{1}{\mu_i + 1} \sum_{m=t_i^S}^{t_i^L} q_k(m)$$

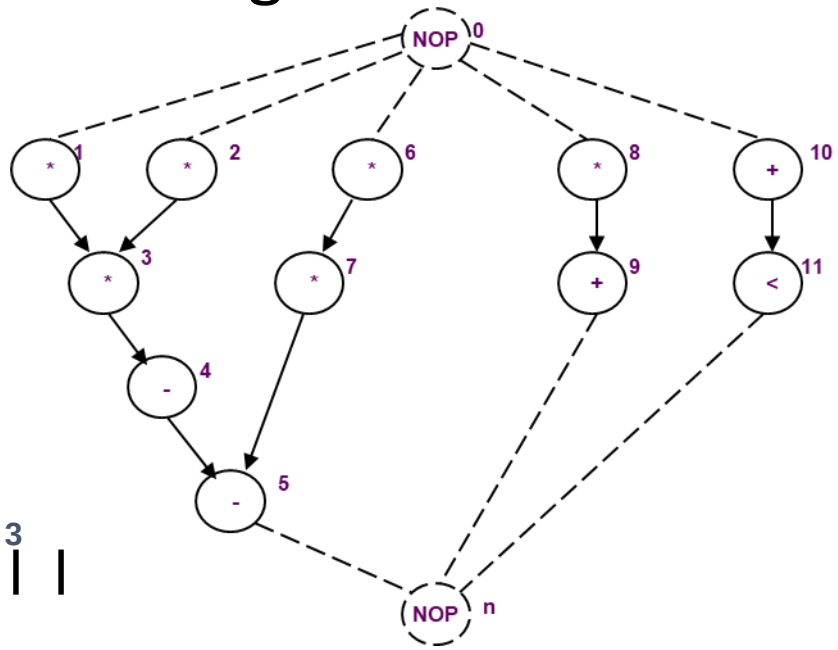
Example

- The assignment of operation v_8 to step 2 implies the assignment of operation v_9 to step 3 or 4.
- Therefore the variation on the force of v_9 is

$1/2(q_2(3) + q_2(4)) - 1/3(q_2(2) + q_2(3) + q_2(4))$

$= 0.5(2 + 1.6) - 0.3(1 + 2 + 1.6)$

$=0.3$



Complexity of Force calculation

- Total force of an operation related to schedule in one time step = self force (I, l) + ps-force (i, l) (of all predecessor/successor)
- Calculation of distribution graph: O(n)
- Self force: O(n)

  For each node

  in its time frame
- Predecessor/successor force: $O(n^2)$

  For each node

  for each predecessor/successor node

Recap -- List Scheduling Algorithm: ML-RC

Minimize latency under resource constraint

LIST_L (G(V,E), **a**) { // resource constraints specified by vector **a**

l = 1

repeat {

for each resource type k {

$U_{l,k}$ = candidate operations available in step l

$T_{l,k}$ = unfinished operations (in progress)

Select $S_k \subseteq U_{l,k}$ such that $|S_k| + |T_{l,k}| \leq a_k$

Schedule the S_k operations at step l

}

$l = l + 1$

} until v_n is scheduled

}

Select operations that increases local concurrency without violating the latency bound

List scheduling algorithm for MRLC - Recap

LIST_R(G(V, $\overline{E}$), λ) {

a = 1;

Compute the latest possible start times t^L by ALAP (G(V, E), λ);

if **($t_0 < 0$)$_L$**

return **(Ø);**

***l* = 1;**

repeat **{**

for each **resource type k = 1, 2, …, n_{res} {**

Determine ready operations $U_{l,k}$;

$T_{l,k}$ = unfinished operations (in progress)

Compute the slacks { $s_i = t_i - l$ for all $v_i \in U_{lk}$};

Select $S_k \subseteq U_{l,k}$ with zero slack and update a s.t. $|S_k| + |T_{l,k}| = a_k$;

Schedule the candidate operations in $U_{l,k}$ not needing additional resources;

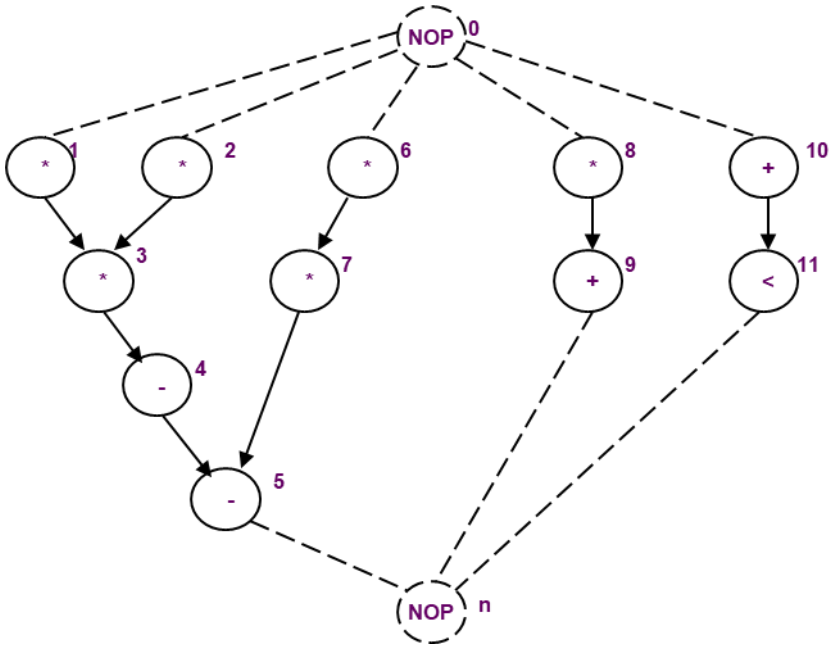
}

***l* = *l* + 1;**

} until (v_n is scheduled) ;

return (t, a);

}



V6
T1=.25
T2 = -1

V7
T2 = 1
T3 -.75

V8
T1 = 1.3
T2 = .93
T3 = -.75

V9:
T2 = .65
T3 = 1.1
T4 = 0.22

V10
T1 = -.69
T2 = .41
T3 = 1.23

V11
T2 = -1.07
T3 = .28
T4 = .022

V1
T1 = 0
V2
T1 = 0
V3
T2 = 0
V4
T3 = 0
V5
T4 = 0

FDS algorithm for MRLC

```
FDS ( G ( V, E ),  λ ) {
    repeat {
        Compute/update the time-frames;
        Compute the operation and type probabilities;
        Compute the self-forces, p/s-forces and total forces;
        Schedule the op. with least force in the corresponding time step;
    } until (all operations are scheduled)
    return (t);
}
```

FDS algorithm for MRLC

- Does not consider one iteration at a time
- The operation with the least force is scheduled in the corresponding step.
- The rationale is to minimize the local concurrency (related to the resource usage) while meeting the latency bound that is satisfied by construction because operations are always scheduled in their time frames

Summary

- Complexity: $O(n^3)$ - why?
- The results have been shown to be superior to list scheduling
- However, its run times tend to be long for large graphs, limiting its practical use for large designs.

Thank You