Variables Involved:

$G = (V, E)$ vertices, Edges, directed, finite
DAG

Each vertex represents a task.

$c(v)$ = computation time needed to execute task v

$e \in E = (a, b)$ where a = Source Task = $src(e)$
b = Destination Task = $dst(e)$

$w(e)$ = time to transfer intertask communication data from source to Dest.

$oe(v)$ = set of output edges for task v

$succ(v)$ = successor tasks of task v.

V_{exit}, V_{entry} } exit task doesnot have successors.

P = set of fully interconnected homogeneous processors.

$map(v)$ = task (v) to processor mapping.

S_v = Start time for execution.

V_d = set $\{v_p \mid v \in V$ and $p \in P\}$ represents all task duplications.

$X_{v_p} = 1$ if v_p is not redundant. } Binary variable.

$d_{a_i b_j} = 1 \Rightarrow b_j$ depends on a_i, $E_r = \{e_{ab_j} \mid e_{ab} \in E, j \in P\}$

Equation 1:

Schedule Length $\boxed{(SL) = \max_{v \in V} f_v}$ → maximum of finish times.

finish time $\boxed{(f_v) = S_v + C(v)}$ → Execution time

S_v → Start time

→ $\boxed{\forall a \neq b,\ map(a) = map(b),\ f_b \leq S_a \text{ or } f_a \leq S_b.}$

If two tasks are mapped to same processor, then finish time of a ≤ start time of b (or) vice versa.

→ Dependent tasks, b is dependent on a then $\boxed{f_a + w(e) \leq S_b}$ should hold. (i.e taking into account communication)

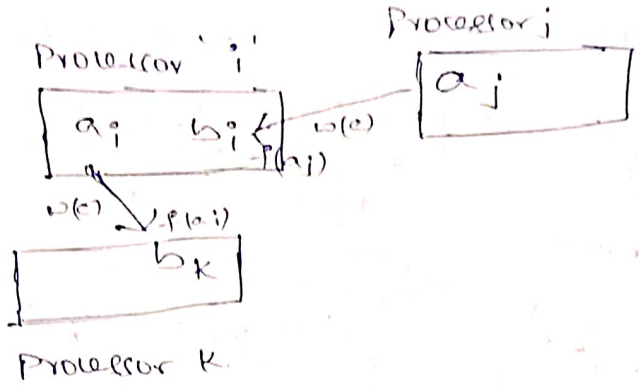
→ Every processor has unique set of tasks.

→ Theorm 1:- If b is dependent on a, in several duplications, 'b' will be exactly dependent on only one 'a'. (one to one) b a

→ Theorm 2:- No useless duplications (i.e) each duplication of its successors feeds data to atleast one duplication

→ Theorm 3:- Though a, b are mapped to diff processors, but 'b' will be mapped to some other 'a' if there is difference in execution time => satisfying dependency.

→ b_i executes first ⇒



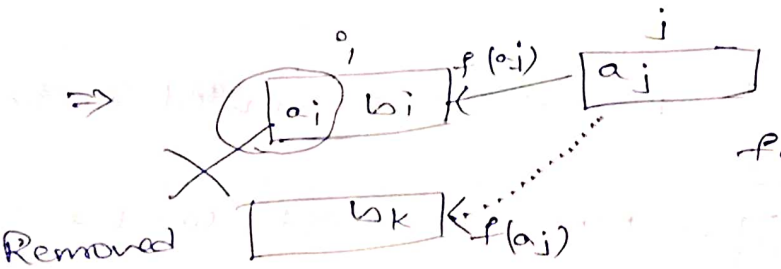
⇒ Data ready time drt_{b_k} is given by

$$\boxed{drt_{b_k} = f_{a_i} + w(e) \geq f(a_j) + w(e) + c(b) + c(a) + w(e).}$$

because b_i executes before a_i

$$\Rightarrow \boxed{f_{a_i} = f_{a_j} + w(e) + c(b) + c(a).}$$

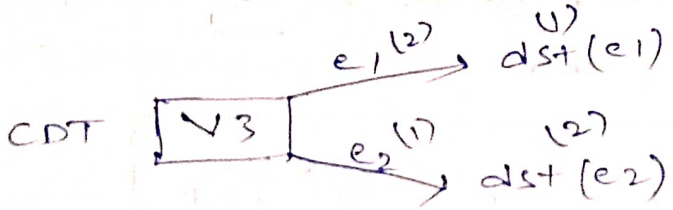
⇒



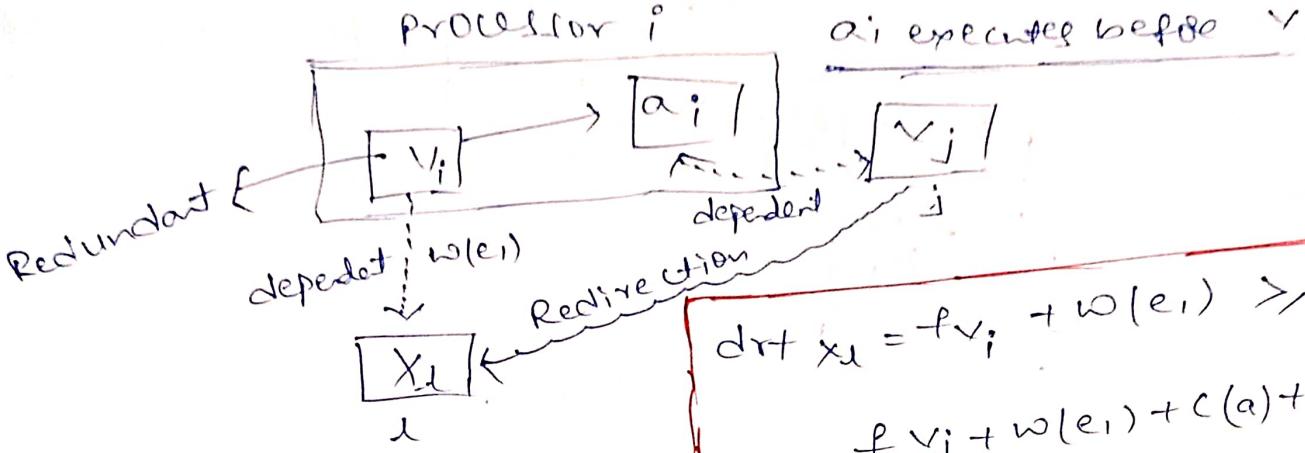
$f_{a_i} \gg f_{a_j}$, redirected & f_{a_i} can be removed.

→ **Theorm 4**:- The Data precedences between the CDT and all its successors are obeyed in partial schedule

→ If any of successor of CDT executes, then that CDT is redundant (∵ successor executed ⇒ all successors gets redirected to that parent). $\boxed{c(v) + w(e_2) + c(b) \geq w(e_1)}$



$$\boxed{c(v) + w(e_1) + c(dst(e_1)) \geq w(e_2)}$$



$$drt\ x_l = f_{v_i} + w(e_1) \geq f_{v_j} + w(e_1) + c(a) + c(v) + w(e_1).$$

∴ If any of successor executes, then Redirection is possible ⇒ $drt\ x_l' < drt\ x_l$ after redirection.

Task Execution :-

$\sum_{p \in P} x_{vp}$ = Duplication number of Task v.

① $\sum_{p \in P} x_{vp} = 1 \ \forall v \in V_{exit}$; Exit task executed only once.

② $\sum_{p \in P} x_{vp} \geq 1, \ \forall v \in V \mid V_{exit}$ ⇒ Non exit task can be executed ≥ 1 times.

③ There should be difference in execution times for tasks mapped to same processor.

$$\forall a \neq b, x_{a_i} = x_{b_i} = 1 \Rightarrow \begin{cases} f_{a_i} - s_{b_i} \leq 0 \\ \text{(or)} \\ f_{b_i} - s_{a_i} \leq 0. \end{cases}$$

④ For redundant task duplication ⇒ $x_{vp} = 0$, we force the start & end time $= 0$.

$s_{vp} < M * x_{vp}$ ($y = 0$) ; $f_{vp} = s_{vp} + c(v) * x_{vp}$ (y_0, y_0).

⑤ Linearizing point ⑦ for tasks on same processor,

$$\boxed{f_{v_p} - s_{a_p} \le M_1 * (2 - X_{v_p} - X_{a_p}) \quad \forall\, a \in succ(v)}$$

, $X_{v_p} = 1$, $X_{a_p} = 1$

, (v_p, a_p) is depend-ent on v

∴ ③ can be applied to task pairs that do not comply with theorem 3 & 4.

Data precedences:-

① If $\boxed{X_{a_i} = 1 \Rightarrow \sum_{b \in succ(a)} \sum_{j \in P} d_{a_i b_j} \ge 1}$ → atleast 1 successor should be there for irredundant task.

② $\boxed{d_{a_i b_j} \le X_{a_i}, \quad d_{a_i b_j} \le X_{b_j}}$ to remove the redundant $d_{a_i b_j}$ if initially it was defined greedily.

② ① + ② $\Rightarrow \boxed{\sum_{b \in succ(a)} \sum_{j \in P} d_{a_i b_j} \ge X_{a_i}}$ } There can be many successors ⇒ ≥

④ $\boxed{\sum_{i \in P} d_{a_i b_j} = X_{b_j} \quad \forall e_{a b_j} \in E_r}$ } b_j consumes from only 1 a_i.

⑤ $\boxed{f_{a_i} + const * w(e_{a b}) - s_{b_j} \le M_2 * (1 - d_{a_i b_j})}$

Satisfying b_j starts after a_i finishes.

If $d_{a_i b_j} = 1$ ⇒ b_j starts right after a_i.

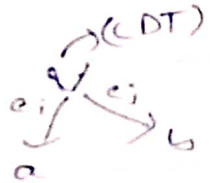
Objective Function:-

$\boxed{\min : sl.}$ minimum schedule length is objective function.

$\boxed{f_{v_p} \leq sl, \ \forall v_p \in V_d, v \in V_{exit}}$ } Largest finish time $\leq$ schedule length.

$\boxed{sl = \max_{v_p \in V_d, v \in V_{exit}} \{f_{v_p}\}}$ schedule length is the largest of all finish times.

Example:-

2 processors $\{i, j\}$, A simple dag 

is taken.

Now we see step by step imposing of constraints in obtaining optimal schedule length.

→ start times are $s_{v_i}, \ldots$

→ Task Duplication is indicated by $x_{v_i} \ldots$

→ Dependency variable $d_{v_i a_j} \ldots$

→ Binary Auxilary Variable z_{a,b_j}, $z_{a,b,j}$.

① Entry task should execute atleast once & exit task should execute ~~atleast~~ once only.

$x_{m_i} + x_{m_j} = 1, \ \forall m \in \{a, b\}$

$x_{v_i} + x_{v_j} \geq 1$

② Limit start time of duplicated redundant task to '0'.

$$s_{m_k} \le M * x_{m_k}, \forall m \in \{a,b\}, k \in P$$

③ As 'a', 'b' are successors of 'v', start time of a, b and finish time of 'v' should be perfectly ordered.

$$f_{v_k} - s_{m_k} \le M * (2 - x_{v_k} - x_{m_k}), \forall m \in \{a,b\}, k \in P$$

④ If a, b are mapped to same processor 'k' then their execution should be ordered.

$$f_{a_k} - s_{b_k} \le M * (2 - x_{a_k} - x_{b_k}) + M * z_{a,b,k} \quad \forall k \in P$$

$$f_{b_k} - s_{a_k} \le M * (2 - x_{a_k} - x_{b_k}) + M * (1 - z_{a,b,k}) \quad \forall k \in P$$

⑤ For m_l to depend on v_k, the dependency variable should be initialized.

$$d_{v_k m_l} \le x_{v_k}, \forall m \in \{a,b\}, k, l \in P.$$

⑥ Redundant duplications are removed by following constraints.

$$d_{v_i a_j} + d_{v_i a_j} + d_{v_i b_j} + d_{v_i b_j} \ge x_{v_i}$$

$$d_{v_j a_i} + d_{v_j a_j} + d_{v_j b_i} + d_{v_j b_j} \ge x_{v_j}$$

⑦ Duplicated successor task should depend only on one duplication of parent task.

$$d_{v_i m_i} + d_{v_j m_i} = x_{m_i}, \forall m \in \{a,b\}$$

$$d_{v_j m_j} + d_{v_i m_j} = x_{m_j}, \forall m \in \{a,b\}.$$

⑧ Since v is CDT & a, b are successors, their order of execution should be ordered.

$$f_{v_i} - s_{m_i} \leq M * (1 - d_{v_i m_i}), \forall m \in \{a, b\}$$

$$f_{v_j} - s_{m_j} \leq M * (1 - d_{v_j m_j}), \forall m \in \{a, b\}$$

$$f_{v_i} + w(e_1) - s_{a_j} \leq M * (1 - d_{v_i a_j})$$

$$f_{v_j} + w(e_1) - s_{a_i} \leq M * (1 - d_{v_j a_i})$$

$$f_{v_i} + w(e_2) - s_{b_j} \leq M * (1 - d_{v_i b_j})$$

$$f_{v_j} + w(e_2) - s_{b_i} \leq M * (1 - d_{v_j b_i}).$$

⑨ The schedule length must be equal to the maximum finish time.

$$f_{m_k} \leq s_1, \forall m \in \{a, b\}, k \in P.$$

Flow of constraints:-

Execute entry task ≥ 1
exit task = 1

↓

Limit Redundant task
start time = 0.

↓

Finish time (v), S(a, b) should
be ordered.

↓

If (v) S(b), (S(a), f(b)) should be ordered
if mapped to same processor.

↓

Dependency variable is initialised.

↓

Redundant duplications are removed
by duplication variable.

↓

Successor should be mapped to only
1 parent.

↓

For a CDT, its dependencies, order of
execution is ordered.

↓

Objective function
Schedule length = max finish time.