8.1

Project management applications

Katta G. Murty, IOE 612 Lecture slides 8

Large projects consist of many individual jobs (or activities) with a predecessor ordering among them.

Job i is a **predecessor** (or **ancestor**) of job j	if j can only be started after i is completed. Then j is called a **successor** of i.

i is an **immediate predecessor (IP)** of j	if i is a predecessor of j, and there is no other job for which i is a predecessor and j a successor. Then j is called an **immediate successor (IS)** of i. A job may have several IPs and several ISs. It can be started anytime after all its IPs are completed.
i is **unrelated** to j	if neither is a predecessor or succeessor of the other. All IPs of a job are unrelated.
Transitivity	If 1 is a predecessor of 2, & 2 is a predecessor of 3, then 1 is a predecessor of 3.

No job can be a predecessor of itself. Predecessor relationships define a **partial ordering** on the jobs.

Inconsistency in IP lists: Exists if transitivity $\Rightarrow$ a job precedes itself. Happens iff the project network to be defined later contains a circuit. A logical error that must be corrected.

Redundancy: if a distant ancestor is listed as an IP. Not a logical error, but increases the complexity of the network.

Example: Constructing an Office Bldg.

Job	IPs	Duration*
1. Site selection & purchase		30
2. Architectural plans (AP)		45
3. Revisions to AP		30
4. Arrange Financing		60
5. Construct outer frame		90
6. Plans for interior		25
7. Order furnishings (desks, etc.)		120
8. Order Computer equipment		120
9. Computer Network Design		70
10. Interior Construction		70
11. Instal computer equipment		90
12. Move-In		5

*Days

Construct single family home

Job	IPs	Duration*
0. Purchase site		16
1. Prepare blueprint	0	6
2. Excavation	1	4
3. Foundations & Basement walls	2	3
4. Frame	3	5
5. Electrical wiring	4	2
6. Insulate	4	1
7. Roofing	4	2
8. Ductwork & Plumbing	6	4
9. Drywall installation	5, 7, 8	6
10. Instal siding	7	4
11. Exterior painting	10	3
12. Interior painting	9	4
13. Carpeting	12	2
14.Instal interior fixtures	13	7
15. Instal Exterior fixtures	11	2
16. Move-in	14, 15	1

*weeks

Hydroelectric Power Station Building Project

No.	Job	IPs	Duration*
1.	Site Ecological survey		6.2
2.	Get EPA approval	1	9.1
3.	Economic feasibility study	1	7.3
4.	Prel. design and cost est.	3	4.2
5.	Project approval	2, 4	10.2
6.	Call quotations for elec. equip. (turbines, generators, ...)	5	4.3
7.	Select suppliers for elec. equip.	6	3.1
8.	Final design	5	6.5
9.	Select construction contractors	5	2.7
10.	Arrange materials supply	8, 9	5.2
11.	Dam building	10	24.8
12.	Power station building	10	18.4
13.	Power lines erection	7, 8	20.3
14.	Elec. equip. installation	7, 12	6.8
15.	Build up reservoir water level	11	2.1
16.	Commission the generators	14, 15	1.2
17.	Start supplying power[281]	13, 16	1.1

*months

Network Representation of Precedence

2 ways of representing predecessor relationships in network form.

1. AON (Activity On Node) network: is very easy to draw, but not very useful for project scheduling. Each job represented as a node. Arc (i, j) included iff i is an IP of j.

2. Activity On Arc (AOA) diagram: Also called Arrow diagram or project network. Each job represented by an arc. Each node corresponds to an Event, "jobs corresponding to all arcs incident into it, and all their predecessors, have been completed".

To portray predecessor relationships correctly in arrow diagram, sometimes necessary to introduce Dummy arcs corresponding to Dummy jobs (which always have 0 duration).

PROPERTIES SATISFIED BY ARROW DIAGRAM:

P1 (i, j) predecessor of (p, q) iff there is a chain from j to p (i.e. predecessor relationship represented through directed paths).

P2 (i, j) is an IP of (p, q) iff either $j = p$ or there is a chain from j to p consisting of dummy jobs only.

Need For Dummy Arcs

Dummy arcs are necessary whenever there is a subset of jobs A that have some but not all IPs in common. In this case we let arcs corresponding to common IPs of jobs in A to have same head node, and then add dummy arcs from that node to the tail nodes of arcs corresponding to jobs in A.

Example

Job	IPs
e_1	
e_2	
e_3	
e_4	
e_5	e_4
e_6	e_1, e_2, e_5
e_7	e_2, e_3, e_5

To avoid confusion, we follow convention that there should be no parallel arcs. So, if there are several jobs with same set of IPs, and same set of immediate successors, we represent them using dummy arcs.

How to Draw Arrow Diagram?

Put a Start node representing event of project commencement. Represent each job with no IP by an arc incident out of this start node.

Same way, at end, represent all jobs that have no successors by arcs incident into a single final node called Finish Node representing project completion.

If a job b has a single IP a, then b can be represented by an arc incident out of the head node of a.

If a job b has more than one IP, let $p_1, \ldots, p_r$ be the head nodes of IPs of job b:

either represent b by an arc incident out of one of the nodes $p_1, \ldots, p_r$, say p_1, and include dummy arcs from $p_2, \ldots, p_r$ into p_1.

or represent b by an arc incident out of a new node Q and include dummy arcs $(p_1, Q), \ldots, (p_r, Q)$.

If some jobs have identical sets of ISs, make the head node of arcs representing them the same.

If several jobs have some common IPs, represent these common IPs by arcs with the same head node, and include dummy arcs coming out of this node as necessary. Check:

- Precedence relationships not implied by original data are not introduced.
- Precedence relationships implied by original data are not omitted.
- See if no. of nodes & no. of dummy nodes can be reduced. For example, if there is a node with a single arc (which is a dummy arc) incident at it, then the 2 nodes on this dummy arc can be merged, and that dummy arc eliminated.

Complexity of drawing the smallest arrow diagram: Constructing AOA diagram with smallest no. of dummy arcs, or smallest no. of nodes is NP-hard. But in practice, the AOA produced by a reasonable heuristic is quite satisfactory.

Since no job can be its own predecessor, project network cannot contain any directed cycles, so project network must be an acyclic network.

Find acyclic numbering of nodes in project network. Node 1 = start node, node n = finish node.

Project Scheduling

To lay out jobs on time axis to minimize project duration. To answer questions like:

- Project start time = 0. How soon can the project be completed?
- How early would the event corresponding to each node materialize?
- What is the earliest time at which each job can be started (called Early Start Time of Job) or finished (Early finish time of job)?
- For each job, what is the latest time at which it can be started, without delaying project completion (Late start, Late finish times of a job)
- Total slack (or total float) of a job = it's late start time − early start time.
- What are critical jobs?

The Data

Node 1 = start node, node n = finish node; the project network $G = (\mathcal{N}, \mathcal{A}, (t_{ij}))$ where for each arc (i, j)

t_{ij} = Duration of job represented by arc (i, j)

= Length of arc (i, j)

Earliest Event Times

Define:

$t_i =$ Length of longest chain from node 1 to node i in project network.

$=$ earliest time at which event corresponding to node i can materialize.

$t_n =$ length of longest chain from start node 1 to finish node n = Minimum Project Duration.

A longest chain from node 1 to node n is called a Critical Path. There may be several critical paths.

Arcs on critical paths called Critical arcs, they represent Critical jobs.

To complete project in min. time, every critical job should be started at earliest time. Any delay in a critical job delays project completion.

Jobs not on any critical path called Slack jobs.

Early start time of job $(i, j) = t_i$

Early Finish Time of job $(i, j) = t_i + t_{ij}$.

Forward pass DP Algorithm to find Early Start Times: t_is

Data Structures

The Forward label on node i will be (L_i, t_i) where L_i called Predecessor Index of node i, is the previous node to i on a longest chain from 1 to i.

Boundary Conditions: $t_1 = 0$. Forward label node 1 with $(\emptyset, 0)$. This label indicates it is Origin node.

In this pass nodes labeled in the order 1 to n. The recurrence eq. for node r is:

$$t_r = \max\{t_i + t_{ir} : \quad i \in B(r)\}$$

L_r in the label for node r is an i that attains the maximum in the above eq.

The longest chain from node 1 to i can be traced using the predecessor indices backwards beginning with node i. It is one critical path.

To find all critical jobs & Latest Start Times:

The forward pass gives only one critical path. To find all critical arcs, we define:

$\mu_i =$ Latest point of time at which the event corresponding to node i has to materialize for the project to finish in minimum duration.

$\mu_n = t_n$ (Boundary Condition)

$\mu_i = \mu_n -$ length of the longest chain from node i to node n.

The Backward Pass

This pass determines the μ_i by DP. They are determined in the order $i = n, n-1, \ldots, 2, 1$.

$\mu_n = t_n$ Boundary condition.

$\mu_i = \min \{\mu_j - t_{ij}: \quad j \in A(i)\}$

The Late Finish of job $(i, j) = \mu_j$

The Late Start of job $(i, j) = \mu_j - t_{ij}$

The Total Slack of job (i, j) = Late start - early start = $\mu_j - t_{ij} - t_i$.

Free float or free slack of job $(i, j) = t_j - EF(i, j)$.

Job (i, j) is critical iff its total slack = 0. Node i on a critical path iff $\mu_i = t_i$.

Two nodes may be on a critical path, & yet the arc joining them may not be a critical job.

Project Shortening Cost Curve

Application of min cost flow algo. to determine which subset of jobs is to be expedited if project has to be finished before its min normal duration. For each job $(i, j) \in \mathcal{A}$

$\overline{\tau}_{ij}$ = normal time duration for completing job (i, j)

$\underline{\tau}_{ij}$ = the minimum, or crash time duration for completing job (i, j)

$\alpha_{ij} (\geq 0)$ = shortening cost in \$/unit time

If there are several critical paths, necessary to shorten at least one job on each. Problem is to determine opt. subset of jobs to shorten (& each by how much) to finish project in given duration λ. Here variables are:

t_{ij} = duration allowed for job (i, j).

t_i = forward pass distance with (t_{ij}) as arc lengths.

Problem is:

$$
\begin{aligned}
Q(\lambda) = \ \text{Maximum} \ \textstyle\sum(\alpha_{ij} t_{ij} \quad &: \quad \text{over } (i,j) \in \mathcal{A}) \\
\text{Subject to} \quad t_{ij} - (t_j - t_i) \ &\leq \ 0, \ \text{for all } (i,j) \in \mathcal{A} \qquad (1) \\
t_{ij} \ &\leq \ \overline{\tau}_{ij}, \ \text{for all } (i,j) \in \mathcal{A} \qquad (2) \\
-t_{ij} \ &\leq \ -\underline{\tau}_{ij}, \ \text{for all } (i,j) \in \mathcal{A} \qquad (3) \\
t_n - t_1 \ &\leq \ \lambda \qquad (4)
\end{aligned}
$$

Associate dual variables $f_{ij}, g_{ij}, h_{ij}, v$ to constraints in that order. The dual is:

$$\text{Min. } W(\lambda, v, f, g, h) = \lambda v \;+\; \sum_{(i,j)\in\mathcal{A}} (\overline{\tau}_{ij} g_{ij}) - \sum_{(i,j)\in\mathcal{A}} (\underline{\tau}_{ij} h_{ij})$$

$$\text{Subject to } f_{ij} + g_{ij} - h_{ij} \;=\; \alpha_{ij}, \text{ for all } (i,j) \in \mathcal{A}$$

$$f(i,\mathcal{N}) - f(\mathcal{N},i) \;=\; \begin{cases} 0, \text{ for all } i \neq 1 \text{ or } n \\ -v, \text{ for } i = n \end{cases} \qquad (5)$$

$$f, g, h, v \;\geq\; 0$$

A min cost flow problem in G with a PL convex obj. func.

To convert into a linear min cost flow, modify network to $G' = (\mathcal{N}, \mathcal{A}')$: replace each $(i,j) \in \mathcal{A}$ by:

One Type 1 arc only called $(i,j,1)$ if $\overline{\tau}_{ij} = \underline{\tau}_{ij} = \tau_{ij}$ say, & $\alpha_{ij} = 0$; i.e., no shortening possible for job (i,j).

A parallel pair (Types 1, 2) arcs $(i,j,1), (i,j,2)$, if $\overline{\tau}_{ij} > \underline{\tau}_{ij}$ & $\alpha_{ij} > 0$; i.e., job (i,j) can be shortened.

Data on arcs is LB $= 0$, Capacity $= k(i,j,r)$, cost coeff. $= c(i,j,r)$, in that order. Given (f, π) in G', the relative cost coeff. of $(i,j,r) = \bar{c}(i,j,r) = c(i,j,r) - (\pi_j - \pi_i)$.

Let $\underline{\lambda}$ denote the min project duration with crash durations, i.e., with $(t_{ij}) = (\underline{\tau}_{ij})$.

Parametric algo. with project duration λ as parameter:

1. Initialization: Using normal durations $(\overline{\tau}_{ij})$ let (L_i^1, t_i^1) be forward pass node labels. Define $(f^1 = 0, \pi^1 = t^1)$, $(t_{ij}^1) = (\overline{\tau}_{ij})$, $\lambda_1 = t_n^1 =$ min normal project duration.

2. Stage 1 labeling: Label 1 with $(\emptyset, 0)$, list $= \{1\}$.

2.1. Select node for Stage 1 Scanning: If list $= \emptyset$ go to 3. Otherwise delete the node i from top of list to scan.

2.2. Stage 1 Scanning: To scan i, label all unlabeled nodes j s. th. $(i, j, 1) \in \mathcal{A}'$ and $\bar{c}(i, j, 1) = 0$ with $(i, +, 1)$ and include it at the bottom of the list.

If n now labeled, there is an ∞-breakthrough $\Rightarrow$ current value of $\lambda = \lambda_p = \underline{\lambda} =$ crash project duration, TERMINATE.

If n not labeled yet, go back to 2.1.

3. Stage 2 Labeling: Make list = set of all labeled nodes now. Let $(f^p, \pi^p), v^p, \lambda_p$ be present quantities.

3.1. Select node for Stage 2 Scanning: If list = $\emptyset$, go to 5. Otherwise delete the node i from top of list to scan.

3.2. Stage 2 Scanning: To scan i:

Forward labeling: Label all unlabeled nodes j s. th. $(i, j, r) \in \mathcal{A}'$, $\bar{c}(i, j, r) = 0$ and $f^p(i, j, r) < k(i, j, r)$ for $r = 1$ or 2 or both, with $(i, +, r)$ and include them at the bottom of the list.

Reverse labeling: Label all unlabeled nodes j s. th. $(j, i, r) \in \mathcal{A}'$, $\bar{c}(j, i, r) = 0$ and $f^p(j, i, r) > 0$ for $r = 1$ or 2 or both, with $(i, -, r)$ and include them at the bottom of the list.

If node n now labeled, there is a finite breakthrough, go to 4. Otherwise go back to 3.1.

4. Flow augmentation: Trace FAP from node 1 to node n & carry flow augmentation. Erase all node labels & go back to 2.

5. Node Price Change: Let $X, \bar{X}$ be sets of labeled, unlabeled nodes. Define:

$$\begin{aligned} A^1 &= \{(i,j,r) \in (X, \bar{X}) : \bar{c}(i,j,r) > 0\} \neq \emptyset \\ A^2 &= \{(i,j,r) \in (\bar{X}, X) : \bar{c}(i,j,r) < 0\} \\ \delta &= \min\{|\bar{c}(i,j,r)| : (i,j,r) \in A^1 \cup A^2\} > 0 \end{aligned}$$

And for $0 \leq \mu \leq \delta$ define:

$$\begin{aligned} \pi_i^p(\mu) &= \begin{cases} \pi_i^p & \text{if } i \in X \\ \pi_i^p - \mu & \text{if } i \in \bar{X} \end{cases} \\ \lambda(\mu) &= \pi_n^p - \mu \\ t_{ij}^p(\mu) &= \min\{\bar{\tau}_{ij}, \pi_j^p(\mu) - \pi_i^p(\mu)\}, \text{ for } (i,j) \in \mathcal{A} \\ t_i^p(\mu) &= \pi_i^p(\mu), \text{ for } i \in \mathcal{N} \\ \pi^{p+1} &= \pi^p(\delta) \\ \lambda_{p+1} &= \lambda_p - \delta \end{aligned}$$

For $0 \leq \mu \leq \delta$, $(t_{ij}^p(\mu)), (t_i^p(\mu))$ is opt. when $\lambda = \lambda(\mu)$. With $(f^p, \pi^{p+1}), v^p, \lambda_{p+1}$ and list = set of all labeled nodes, go back to 2.1.

Resource constrained project scheduling

Jobs may require resources such as a crane, trained personnel etc. in limited supply. Scheduling jobs with resource constraints requires combinatorial opt. methods, not just network flow methods.