

620-362 Applied Operations Research

ASSIGNMENT 1

Due Date: Wednesday 3rd September 2003 5:00pm

Please deposit your assignment in Box 183 by the time indicated.

This assignment must be your own work; no discussion or collaboration is permitted. Please make an effort to write clearly, legibly and succinctly.

1. A construction project has been broken down into T tasks. With the resources currently allocated to each task, the duration of the i th task has been estimated to be d_i units of time, for each $i = 1, \dots, T$. While some tasks can be carried out in parallel, some cannot, due to dependencies. The project planners have determined for each task, the set of other tasks that must be completed before it can start. This information is recorded in a set of ordered pairs, P , which consists of the set of all pairs (i, j) such that the j th task cannot commence until after the completion of the i th task, for $i, j \in \{1, \dots, T\}$.

- (a) The project planners wish to minimize the completion time of the entire project. Thus if e_i denotes the completion time of the i th task, for $i = 1, \dots, T$, they wish to *minimize*

$$\max_{i=1, \dots, T} e_i.$$

- i. Show how the planners' problem of finding a project schedule (a start time for each task) can be modelled as a linear program. Clearly define all your variables, and explain the purpose of each constraint.

Solution: We use variables e_i to denote the completion time of task i , for each $i = 1, \dots, T$, and b to denote the project end time. Then the linear programming model is as follows.

$$\begin{array}{ll} \min & b \\ \text{s.t.} & b \geq e_i \quad \forall i = 1, \dots, T \\ & e_j - t_j \geq e_i \quad \forall (i, j) \in P \\ & e_i \geq 0 \quad \forall i = 1, \dots, T \end{array}$$

Alternatively, we can use additional variables s_i to denote the start time

of task i , for each $i = 1, \dots, T$, and have the following model.

$$\begin{aligned}
 \min \quad & b \\
 \text{s.t.} \quad & b \geq e_i \quad \forall i = 1, \dots, T \\
 & s_j \geq e_i \quad \forall (i, j) \in P \\
 & e_i = s_i + t_i, \quad \forall i = 1, \dots, T \\
 & s_i \geq 0 \quad \forall i = 1, \dots, T
 \end{aligned}$$

It is also possible to write the model entirely in terms of the s variables rather than the e variables.

ii. Implement your model from part 1(a)i as an AMPL model.

Solution:

The AMPL model file for the second model, with both task start and end time variables, is as follows.

```

set TASKS; # the set of tasks to be completed
set PRECEDENCES within {TASKS,TASKS}; # the set of precedences
# note (i,j) in PRECEDENCES implies
# that task i must be completed before
# task j.

param duration {TASKS} >=0; # the duration of each task

var StartTime {TASKS} >=0; # the time at which we will start each task
var EndTime {TASKS} >=0; # the time when each task finishes
var ProjectEndTime >=0; # the time when the project is finished

minimize AllDone: ProjectEndTime;

# each task may be finished prior to EndTime
subject to EndTimes { i in TASKS } :
    EndTime[i] <= ProjectEndTime;

# task may not start before its preceding tasks are finished
subject to Precedences { (i,j) in PRECEDENCES } :
    EndTime[i] <= StartTime[j];

# the end time of each task, is it's start time plus its duration.
subject to StartAndEndTimes { i in TASKS } :
    EndTime[i] = StartTime[i]+duration[i];

```

iii. Create an AMPL data file for your model using the following data. The list of tasks and expected duration of each task is given in Table 1. The set P is given by the precedence graph in Figure 1. Solve the problem using AMPL. What is the optimal project schedule? What is the overall project end time?

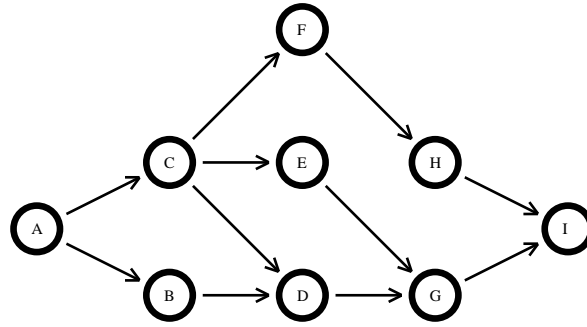


Figure 1: Task Precedence Graph

Task	A	B	C	D	E	F	G	H	I
Duration	2	3	4	5	2	3	4	5	6

Table 1: Task Durations

Solution:

The AMPL data file for the second model, with both task start and end time variables, is as follows.

```

set TASKS = A B C D E F G H I;
set PRECEDENCES =
    (A,B) (A,C)
    (B,D)
    (C,D) (C,E) (C,F)
    (D,G)
    (E,G)
    (F,H)
    (G,I)
    (H,I)
;

param duration :=
A      2
B      3
C      4
D      5
E      2
F      3
G      4
H      5
I      6
;
  
```

The optimal project schedule is to start each task at the times indicated in the table below.

Task	A	B	C	D	E	F	G	H	I
Start Time	0	2	2	6	6	6	11	9	15

The overall project end time is 21 units of time.

- (b) The planners realize that they can accelerate some tasks by acquiring more of a critical resource at casual rates (for example, general labour). After analysis, they find that for the i th task, they can reduce the duration of the task by up to r_i units of time, at a cost of c_i dollars per unit time reduced.

The planners also find that they can reduce costs by slowing down some tasks. Their analysis has found that for the i th task, they can reduce costs at a rate of p_i dollars per unit time that the duration of the task increases, up to a maximum excess duration of m_i units.

- i. Explain briefly why the cost the planners attribute to the actual duration of the i th task is given by the piecewise linear function

$$C(a_i) = \begin{cases} c_i(d_i - a_i) & d_i - r_i \leq a_i \leq d_i \\ p_i(d_i - a_i) & d_i < a_i \leq d_i + m_i \end{cases}$$

where a_i denotes the actual duration of the i th task. Give a sketch of the function C .

Solution:

If the actual duration of task i is a_i , then there are two cases. If $a_i \leq d_i$, then the task duration has been reduced, by $d_i - a_i$ units. This can occur by no more than r_i units, so it must be that $d_i - a_i \leq r_i$, and so in this case $d_i - r_i \leq a_i \leq d_i$. Each unit the duration is reduced costs c_i units, so in this case the cost is $c_i(d_i - a_i)$ units. The second case is that $a_i > d_i$, in which case the task duration is increased, by $a_i - d_i$ units. The increase cannot exceed m_i units, so $a_i - d_i \leq m_i$, and in this case $d_i < a_i \leq d_i + m_i$. Each unit of increase yields a benefit of p_i units, so the cost in this case is $-p_i(a_i - d_i) = p_i(d_i - a_i)$.

The function consists of two linear pieces, joined with a “kink” at the point $(d_i, 0)$. The first linear piece has slope $-c_i$, and runs from the point $(d_i - r_i, c_i r_i)$ to $(d_i, 0)$. The second linear piece has slope $-p_i$, runs from the point $(d_i, 0)$ to the point $(d_i + m_i, -p_i m_i)$.

- ii. For each unit of time the project runs over a given deadline D , the company is penalized P dollars. The planners wish to find a project schedule, including provision for reducing or increasing the durations of the tasks, that minimizes the total cost of carrying out the project, including task duration costs and late penalties. Give a condition on c_i relative to p_i that will ensure the problem can be modelled as a linear program.

Solution:

In order to model the problem as a linear program, the piecewise linear cost function (to be minimized) needs to be convex. For this to occur,

the slope of C_i must increase with a_i , and so it must be that $-p_i \geq -c_i$, i.e. $p_i \leq c_i$ for all $i = 1, \dots, T$.

- iii. Give a linear programming model of the planners' problem as described in part 1(b)ii. Clearly define all variables, and explain the purpose of each constraint.

Solution:

We modify the second linear programming model from Question 1(a). For each task $i = 1, \dots, T$, we introduce new variables f_i and h_i to denote the amount by which the task duration was increased, and decreased, respectively. We also introduce variable q to denote the amount by which the project end time exceeded the given deadline.

$$\begin{aligned}
 \min \quad & Pq + \sum_{i=1}^T (c_i f_i - p_i h_i) \\
 \text{s.t.} \quad & b \geq e_i && \forall i = 1, \dots, T \\
 & s_j \geq e_i && \forall (i, j) \in P \\
 & e_i - s_i = t_i - f_i + h_i, && \forall i = 1, \dots, T \\
 & q \geq b - D \\
 & 0 \leq f_i \leq r_i, && \forall i = 1, \dots, T \\
 & 0 \leq h_i \leq m_i, && \forall i = 1, \dots, T \\
 & s_i \geq 0 && \forall i = 1, \dots, T \\
 & q \geq 0
 \end{aligned}$$

The first constraint ensures that the project end time occurs no earlier than the end time of any task. The second constraint ensures precedence relations are obeyed, i.e. that the start of task j occurs no earlier than the end of task i , if i must precede j . The third constraint ensures that the actual duration of task i , given by $e_i - s_i$, is equal to the normal duration t_i , less any speed-up time f_i , and plus any delay time h_i . The fourth constraint, in conjunction with the non-negativity of q , makes sure that the amount the project is over its deadline, q , is at least $\max\{b - D, 0\}$. The fifth constraint ensures that the number of units of task i speed-up does not exceed the maximum allowed, r_i . Similarly the sixth constraint ensures that the number of units of task i delay does not exceed the maximum allowed, m_i .

- iv. Implement your linear program as an AMPL model.

Solution:

We first give the sets, parameters and variables used in the AMPL model.

```

set TASKS;                                # the set of tasks to be completed
set PRECEDENCES within {TASKS,TASKS};    # the set of precedences
                                           # Note that (i,j) in PRECEDENCES implies
                                           # that task i must be completed before task j.

param normal_duration TASKS >=0;         # the normal duration of each task
param max_fast_time TASKS >=0;          # the maximum amount a task may be sped up by
param cost_fast_time TASKS ;             # the increased cost per unit time faster
param max_slow_time TASKS >=0;          # the maximum amount a task may be slowed down by
param costred_slow_time TASKS ;         # the cost reduction per unit time slower

param deadline >=0;                       # the projects deadline
param cost_time_late >=0;                 # the penalty per unit time late

# check that the input data matches our assumptions about convexity
  check {i in TASKS} : cost_fast_time[i] >= costred_slow_time[i];

#check that tasks can't finish before they are begun.
  check { i in TASKS } : normal_duration[i] >= max_fast_time[i];

var StartTime TASKS >=0;                  # the time at which we will start each task
var EndTime TASKS >=0;                   # the time when each task finishes

var FastTime TASKS >=0;                  # how much faster than normal each task is completed
var SlowTime TASKS >=0;                 # how much slower than normal each task is completed

var ProjectEndTime >=0;                  # the time when the project is finished
var TimeLate >=0;                       # the amount by which the project is late.

```

We now give the body of the AMPL model.

```

minimize TotalCost : TimeLate*cost_time_late +
    sum {i in TASKS} (FastTime[i]*cost_fast_time[i]
        - SlowTime[i]*costred_slow_time[i]);

# each task must be finished prior to EndTime
subject to EndTimes { i in TASKS } :
    EndTime[i] <= ProjectEndTime;

# task may not start before its preceding tasks are finished
subject to Precedences { (i,j) in PRECEDENCES } :
    EndTime[i] <= StartTime[j];

# the end time of each task, is it's start time plus its duration.
subject to StartAndEndTimes { i in TASKS } :
    EndTime[i] = StartTime[i]+normal_duration[i]+SlowTime[i]-FastTime[i];

# not too much fast time on each task
subject to MaxFastTime { i in TASKS } :
    FastTime[i] <= max_fast_time[i];

#not too much slow time on any task
subject to MaxSlowTime { i in TASKS } :
    SlowTime[i] <= max_slow_time[i];

#ProjectEndTime is coupled too Early and late times
subject to ComputeProjectEndTime:
    ProjectEndTime <= deadline + TimeLate;

```

- v. Using the data from part 1a together with the values from Table 2, with a deadline $D = 15$ and Penalty $P = 8$ create an AMPL data file for your model. Note that the costs given satisfy your condition from part 1(b)ii. Solve the linear program using AMPL. What time should each task start and what is its optimal duration? What time will the project be completed?

Task	A	B	C	D	E	F	G	H	I
r	1	1	2	1	1	1	1	1	2
c	6	3	4	5	2	6	4	6	6
m	1	1	2	1	2	1	1	2	2
p	3	1	2	2	1	1	2	2	2

Table 2: Costs and bounds on modifying task duration.

Solution:

```

set TASKS = A B C D E F G H I;
set PRECEDENCES =
    (A,B) (A,C)
    (B,D)
    (C,D) (C,E) (C,F)
    (D,G)
    (E,G)
    (F,H)
    (G,I)
    (H,I)
;

param : normal_duration max_fast_time cost_fast_time max_slow_time costred_slow_time :=
A      2          1          6          1          3
B      3          1          3          1          1
C      4          2          4          2          2
D      5          1          5          1          2
E      2          1          2          2          1
F      3          1          6          1          1
G      4          1          4          1          2
H      5          1          6          2          2
I      6          2          6          2          2
;

param deadline := 15;
param cost_time_late := 8;

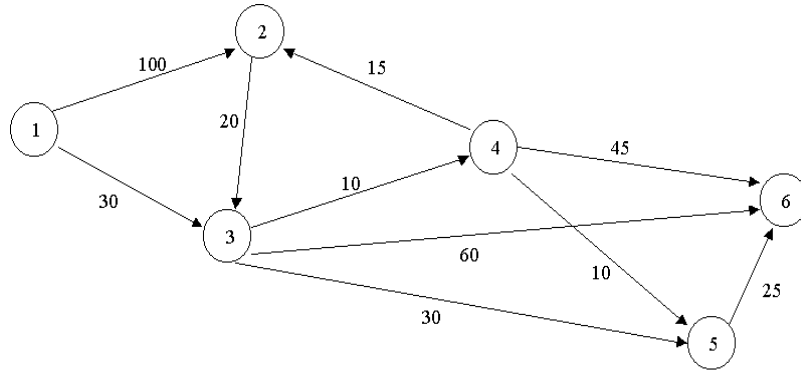
```

The optimal project schedule is to start each task at the times indicated in the table below, and speed up or slow down each task by the amounts indicated. The actual task durations are also given.

Task	A	B	C	D	E	F	G	H	I
Start Time	0	1	1	3	3	3	8	6	11
Speed-up	1	1	2	0	0	0	1	0	2
Delay	0	0	0	0	2	0	0	0	0
Actual Duration	1	2	2	5	4	3	3	5	4

The overall project end time is 15 units of time, and the total cost is 31 units.

2. Use the label-correcting algorithm with queue to find the shortest path from node 1 to node 6 in the network shown below. Show all working using an appropriate table.



- (a) What is the shortest path from node 1 to node 6, and what is its length?

Solution: The iterates of the label-correcting algorithm, with queue, for the shortest path between nodes 1 and 6 in this network produces the following table.

i	1	2	3	4	5	6	Q
	(0, -)	(∞ , -)	(∞ , -)	(∞ , -)	(∞ , -)	(∞ , -)	1
1	(0, -)	(100, 1)	(30, 1)	(∞ , -)	(∞ , -)	(∞ , -)	2,3
2	(0, -)	(100, 1)	(30, 1)	(∞ , -)	(∞ , -)	(∞ , -)	3
3	(0, -)	(100, 1)	(30, 1)	(40, 3)	(60, 3)	(90, 3)	4,5,6
4	(0, -)	(55, 4)	(30, 1)	(40, 3)	(50, 4)	(85, 4)	2,5,6
2	(0, -)	(55, 4)	(30, 1)	(40, 3)	(50, 4)	(85, 4)	5,6
5	(0, -)	(55, 4)	(30, 1)	(40, 3)	(50, 4)	(75, 5)	6
6	(0, -)	(55, 4)	(30, 1)	(40, 3)	(50, 4)	(75, 5)	ϕ

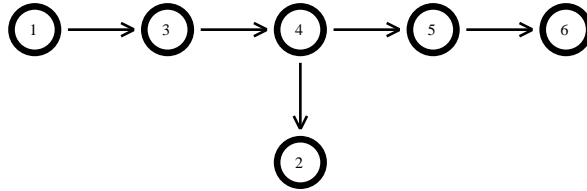
This identifies the following shortest path



The shortest path between nodes 1 and 6 has length 75.

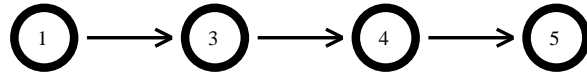
- (b) Give the shortest path tree.

Solution: The shortest path tree from node 1 is represented by the following figure.



- (c) Use the shortest path tree to find the shortest path from node 1 to node 5.

Solution: The shortest path from node 1 to node 5 is shown in the following figure.



- (d) Give two nodes in the network for which the shortest path from one to the other does *not* lie in the shortest path tree you found in part 2a.

Solution: There are many pair of nodes that have a shortest path from one to the other that does not lie in the shortest path tree from node 1. Some examples are from node 2 to node 5, from node 5 to node 4 or node 6 to any other node.