MTHSC 4420 Advanced Mathematical Programming Homework 2

Megan Bryant

October 30, 2013

1. Use Dijkstra's algorithm to solve the shortest path problem depicted below. For each iteration, list the node you mark permanent and the label and predecessor updates for each adjacent node. Count the total number of distance updates.

Table 1: Dijkstra's Algorithm

Node	Step 1	Step 2	Step 3	Step4	Step 5	Step 6
1	$\underline{0}$					
2	∞	$\underline{2}$				
3	∞	8	2+5 = 7	$2 + 3 + 1 = \underline{6}$		
4	∞	∞	$3 + 2 = \underline{5}$			
5	∞	∞	∞	$8 + 0 = 8$	$\underline{8}$	
6	∞	∞	∞	$2 + 3 + 6 = 11$	11	$\underline{11}$

2. Consider this uncapacitated facility location problem with five potential locations. The fixed costs for opening facilities at each location are $f = (4, 3, 4, 4, 7)$. The costs for satisfying the demand for client i (row) from location j (column) is given by c_{ij} where

$$C = \begin{bmatrix} 12 & 13 & 6 & 0 & 1 \\ 8 & 4 & 9 & 1 & 2 \\ 2 & 6 & 6 & 0 & 1 \\ 3 & 5 & 2 & 1 & 8 \\ 8 & 0 & 5 & 10 & 8 \\ 2 & 0 & 3 & 4 & 1 \end{bmatrix}.$$

(a) Formulate this problem as an AMPL model using the aggregated form of the fixed-charge constraint (that clients be served only from open facilities). Solve the LP relaxation (with nonnegative location variables instead of binary) and report your solution and objective function value.

Original Aggregate LP:

Let $N = \{1, 2, 3, 4, 5\}$ be the set of locations.

Let $M = \{1, 2, 3, 4, 5\}$ be the set of clients.

Let $x_{ij} = \begin{cases} 1 & \text{if location } j \text{ serves client } i \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in N, i \in M.$

Let $y_j = \begin{cases} 1 & \text{if location } j \text{ is opened} \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in N.$

Minimize: $\sum_{j \in N} f_j y_j + \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij}$

Subject to:

$\sum_{j \in N} x_{ij} = 1, \forall i \in M$

$\sum_{i \in M} x_{ij} \leq 5 y_j, \forall j \in N$

$x_{ij} \in \{0, 1\}, \forall i \in M, j \in N$

$y_j \in \{0, 1\}, \forall j \in N$

Original Aggregate LP AMPL

Model

```
param locations = 5;
param clients = 6;

param costs{i in 1..clients, j in 1..locations};
param fixed_costs{j in 1..locations};

var x{i in 1..clients,j in 1..locations} binary;
#1 if client i is served by location j

var y{j in 1..locations} binary;
#1 if location j is opened

minimize Operating_Costs:
sum{j in 1..locations}fixed_costs[j]*y[j] +
sum{i in 1..clients, j in 1..locations}costs[i,j]*x[i,j];
#Minimizing the total operating costs of opening
```

```
a location and servicing a client from a location

subject to Clients_Serviced {i in 1..clients}:
sum{j in 1..locations} x[i,j] = 1;
#Ensures that each client is serviced

subject to Available_Locations {j in 1..locations}:
sum{i in 1..clients} x[i,j] <= y[j] * clients;
#Ensures that clients can only
be serviced from open locations
```

Data

```
param costs : 1 2 3 4 5 :=
1 12 13 6 0 1
2 8 4 9 1 2
3 2 6 6 0 1
4 3 5 2 1 8
5 8 0 5 10 8
6 2 0 3 4 1;

param fixed_costs := 1 4 2 3 3 4 4 4 5 7;
```

Relaxed Aggregate LP:

Let $N = \{1, 2, 3, 4, 5\}$ be the set of locations.

Let $M = \{1, 2, 3, 4, 5\}$ be the set of clients.

Let $x_{ij} = \begin{cases} 1 & \text{if location } j \text{ serves client } i \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in N, i \in M.$

Let $y_j = \begin{cases} 1 & \text{if location } j \text{ is opened} \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in N.$

Minimize: $\sum_{j \in N} f_j y_j + \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij}$

Subject to:

$\sum_{j \in N} x_{ij} = 1, \forall i \in M$

$\sum_{i-1}^{5} x_{ij} \leq 6y_j, \forall j \in N$

$0 \leq x_{ij} \leq 1, \forall i \in M, j \in N$

$0 \leq y_j \leq 1, \forall j \in N$

Relaxed Aggregate LP AMPL

Model

```
param locations = 5;
param clients = 6;

param costs{i in 1..clients, j in 1..locations};
param fixed_costs{j in 1..locations};

var x{i in 1..clients,j in 1..locations} binary;
#1 if client i is served by location j

var y{j in 1..locations} binary;
#1 if location j is opened

minimize Operating_Costs:
sum{j in 1..locations}fixed_costs[j]*y[j] +
sum{i in 1..clients, j in 1..locations}costs[i,j]*x[i,j];
#Minimizing the total operating costs of
opening a location and servicing a client from a location

subject to Clients_Serviced {i in 1..clients}:
sum{j in 1..locations} x[i,j] = 1;
#Ensures that each client is serviced

subject to Available_Locations {j in 1..locations}:
sum{i in 1..clients} x[i,j] <= y[j] * clients;
#Ensures that clients can only
be serviced from open locations

option relax_integrality 1;
#Relaxes the integrality constraints
```

Data

```
param costs : 1 2 3 4 5 :=
1 12 13 6 0 1
2 8 4 9 1 2
3 2 6 6 0 1
4 3 5 2 1 8
5 8 0 5 10 8
6 2 0 3 4 1;

param fixed_costs := 1 4 2 3 3 4 4 4 5 7;
```

Relaxed Aggregated LP Solution:

```
ampl: model relaxedproblem2mod.mod;
ampl: data relaxedproblem2dat.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: optimal solution; objective 5.666666667
```

(b) Formulate this problem as an AMPL model using the disaggregated form of the fixed-charge constraint. Solve the LP relaxation and report your solution and objective function value.

Original Disaggregate LP:
Let $N = \{1, 2, 3, 4, 5\}$ be the set of locations.
Let $M = \{1, 2, 3, 4, 5\}$ be the set of clients.
Let $x_{ij} = \begin{cases} 1 & \text{if location } j \text{ serves client } i \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in N, i \in M.$
Let $y_j = \begin{cases} 1 & \text{if location } j \text{ is opened} \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in N.$

Minimize: $\sum_{j \in N} f_j y_j + \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij}$
Subject to:
$\sum_{j \in N} x_{ij} = 1, \forall i \in M$
$x_{ij} \leq y_j, \forall i \in M, j \in N$
$x_{ij} \in \{0, 1\}, \forall i \in M, j \in N$
$y_j \in \{0, 1\}, \forall j \in N$

Original Disaggregate LP AMPL
Model

```
param locations = 5;
param clients = 6;

param costs{i in 1..clients, j in 1..locations};
param fixed_costs{j in 1..locations};

var x{i in 1..clients,j in 1..locations} binary;
#1 if client i is served by location j

var y{j in 1..locations} binary;
#1 if location j is opened

minimize Operating_Costs:
sum{j in 1..locations}fixed_costs[j]*y[j] +
sum{i in 1..clients, j in 1..locations}costs[i,j]*x[i,j];
```

```
#Minimizing the total operating costs of opening
a location and servicing a client from a location

subject to Clients_Serviced {i in 1..clients}:
sum{j in 1..locations} x[i,j] = 1;
#Ensures that each client is serviced

subject to Available_Locations
{i in 1..clients, j in 1..locations}:
x[i,j] <= y[j];
#Ensures that clients can only be
serviced from open locations
```

Data

```
param costs : 1 2 3 4 5 :=
1 12 13 6 0 1
2 8 4 9 1 2
3 2 6 6 0 1
4 3 5 2 1 8
5 8 0 5 10 8
6 2 0 3 4 1;

param fixed_costs := 1 4 2 3 3 4 4 4 5 7;
```

Relaxed Disaggregate LP:
Let $N = \{1, 2, 3, 4, 5\}$ be the set of locations.
Let $M = \{1, 2, 3, 4, 5\}$ be the set of clients.
Let $x_{ij} = \begin{cases} 1 & \text{if location } j \text{ serves client } i \\ 0 & \text{otherwise} \end{cases} \forall j \in N, i \in M.$
Let $y_j = \begin{cases} 1 & \text{if location } j \text{ is opened} \\ 0 & \text{otherwise} \end{cases} \forall j \in N.$

Minimize: $\sum_{j \in N} f_j y_j + \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij}$
Subject to:
$\sum_{j \in N} x_{ij} = 1, \forall i \in M$
$x_{ij} \leq y_j, \forall i \in M, j \in N$
$x_{ij} \geq 0, \forall i \in M, j \in N$
$y_j \geq 0, \forall j \in N$

Relaxed Disaggregate LP AMPL
Model

```
param locations = 5;
param clients = 6;

param costs{i in 1..clients, j in 1..locations};
param fixed_costs{j in 1..locations};

var x{i in 1..clients,j in 1..locations} binary;
#1 if client i is served by location j

var y{j in 1..locations} binary;
#1 if location j is opened

minimize Operating_Costs:
sum{j in 1..locations}fixed_costs[j]*y[j] +
sum{i in 1..clients, j in 1..locations}costs[i,j]*x[i,j];
#Minimizing the total operating costs of opening
a location and servicing a client from a location

subject to Clients_Serviced {i in 1..clients}:
sum{j in 1..locations} x[i,j] = 1;
#Ensures that each client is serviced

subject to Available_Locations
{i in 1..clients, j in 1..locations}:
x[i,j] <= y[j];
#Ensures that clients can only
be serviced from open locations

option relax_integrality 1;
#Relaxes the integrality conditions
```

Data

```
param costs : 1 2 3 4 5 :=
1 12 13 6 0 1
2 8 4 9 1 2
3 2 6 6 0 1
4 3 5 2 1 8
5 8 0 5 10 8
6 2 0 3 4 1;

param fixed_costs := 1 4 2 3 3 4 4 4 5 7;
```

Relaxed Disaggregate LP Solution:

```
ampl: model relaxedproblem3mod.mod;
```

```
ampl: data relaxedproblem3dat.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: optimal solution; objective 9
6 simplex iterations
```

(c) Solve the original IP of each of your models (with binary location variables) and compare the solutions with each other and with the respective relaxation solutions.

Original Aggregate LP Solution:

```
ampl: model originalproblem2mod.mod;
ampl: data originalproblem2dat.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: optimal solution; objective 9
9 simplex iterations
```

Original Disaggregated LP Solution:

```
ampl: model originalproblem3mod.mod;
ampl: data originalproblem3dat.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: optimal solution; objective 9
12 simplex iterations
```

The relaxed aggregate LP had a solution of $5\frac{2}{3}$ and the relaxed disaggregate LP had a solution of 9. Comparing these to both the original aggregate and original disaggregate solutions of 9, we see that the relaxed disaggregate LP had a better bound and, in fact, found the optimal solution.

3. The QED Company must draw up a production program for the next nine weeks. Jobs last several weeks and once started must be carried out without interruption. During each week, a certain number of skilled workers are required to work full time on the job. Thus, if job i lasts p_i weeks, l_{iu} workers are required in week u for $u = 1, 2, \ldots, p_i$. The total number of workers available in week t is L_t. Typical job data is shown below.

		Week			
Job	Length	1	2	3	4
1	3	2	3	1	—
2	2	4	5	—	—
3	4	2	4	1	5
4	4	3	4	2	2
5	3	9	2	3	—

(a) Formulate the problem of finding a feasible schedule as an IP.

l_{iu} : number of workers required in the u-th week of doing job i
L_t : number of workers available in week t
p_i : number of weeks to complete job i

IP Feasible Solution:
Let $I = \{1, 2, 3, 4, 5\}$ be the set of all jobs
Let $T = \{1, \cdots, 9\}$ be the set of all weeks
Let $Y_i = \{t : (t + pi - 1) \in T\}$ be the set of possible starting weeks for job i
Let $x_{it} = \begin{cases} 1 & \text{if job } i \text{ starts in week } t \\ 0 & \text{otherwise} \end{cases} \forall i \in I, t \in Y_i.$
Let w_t be the number of workers used in week $t \in T$

$\sum_{t \in Y_i} x_{it} = 1, \forall i \in I$
$w_t = \sum_{i \in I} \sum_{u \in Y_i : u \leq t < (u+p_i)} l_{i(t-u+1)} x_{iu}, \forall t \in T$
$w_t \leq L_t, \forall t \in T$
$x_{it} \in \{0, 1\}, \forall i \in I, t \in Y_i$

(b) Formulate and solve when the objective is to minimize the maximum number of workers used during any of the nine weeks.

l_{iu} : number of workers required in the u-th week of doing job i
L_t : number of workers available in week t
p_i : number of weeks to complete job i

IP Program:
Let $I = \{1, 2, 3, 4, 5\}$ be the set of all jobs
Let $T = \{1, \cdots, 9\}$ be the set of all weeks

Let $Y_i = \{t : (t + pi - 1) \in T\}$ be the set of possible starting weeks for job i

Let $x_{it} = \begin{cases} 1 & \text{if job } i \text{ starts in week } t \\ 0 & \text{otherwise} \end{cases} \forall i \in I, t \in Y_i.$

Let w_t be the number of workers used in week $t \in T$

Let z = maximum number of workers in any single week

Minimize: z

Subject to: $\sum_{t \in Y_i} x_{it} = 1, \forall i \in I$

$w_t = \sum_{i \in I} \sum_{u \in Y i : u \leq t \leq (u + p_i)} l_{i(t-u+1)} x_{iu}, \forall t \in T$

$w_t \leq L_t, \forall t \in T$

$w_t \leq z, \forall t \in T$

$x_{it} \in \{0, 1\}, \forall i \in I, t \in Y_i$

IP AMPL Code

Model

```
param jobs = 5;
param total_weeks = 9;
param max_job_length = 4;
param max_workers;

param job_workers{i in 1..jobs, j in 1..max_job_length};
param job_length{i in 1..jobs};

var x{i in 1..jobs, t in 1..total_weeks} binary;
#1 if job i starts in week t,otherwise
var M;

set possible_start_weeks{i in 1..jobs}:=
{t in 1..total_weeks: 1 <= t + job_length[i] <= total_weeks};
set active_weeks{i in 1..jobs,t in possible_start_weeks[i]}:=
{u in possible_start_weeks[i]:u <= t < u + job_length[i]};

minimize Max_Workers_Week: M;

subject to M_def {t in 1..total_weeks}:
M >=
sum{i in 1..jobs, u in 1..max_job_length}x[i,t]*job_workers[i,u];

subject to Active_Job {i in 1..jobs}:
sum{t in possible_start_weeks[i]}x[i,t] = 1;
```

```
#Ensure that every job is completed

#subject to Workers_Used {t in 1..total_weeks}:
#sum{i in 1..jobs, u in active_weeks[i,t]}
job_workers[i,t-u+1]*x[i,u] <= max_workers[t];
#Ensure that the number of workers used in
week t is less than the number of workers available in week t

subject to Min_Max_Workers_Week{t in 1..total_weeks}:
sum{i in 1..jobs, u in 1..max_job_length}job_workers[i,u]*x[i,u] <= M;
#Minimizes the maximum number of workers per week
```

Data

```
param job_workers: 1 2 3 4 :=
1 2 3 1 0
2 2 4 5 0
3 2 4 1 5
4 3 4 2 2
5 9 2 3 0;

param job_length := 1 3 2 2 3 4 4 4 5 3;
```

IP AMPL Solution

```
ampl: model problem3bmod.mod;
ampl: data problem3bdat.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: optimal solution; objective 14
40 simplex iterations
plus 2 simplex iterations for intbasis
```

(c) Formulate and solve with the requirement that Job 1 must start at least two weeks before job 3.

l_{iu} : number of workers required in the u-th week of doing job i
L_t : number of workers available in week t
p_i : number of weeks to complete job i

IP Program:
Let $I = \{1, 2, 3, 4, 5\}$ be the set of all jobs
Let $T = \{1, \cdots, 9\}$ be the set of all weeks

Let $Y_i = \{t : (t + pi - 1) \in T\}$ be the set of possible starting weeks for job i

Let $x_{it} = \begin{cases} 1 & \text{if job } i \text{ starts in week } t \\ 0 & \text{otherwise} \end{cases} \forall i \in I, t \in Y_i.$

Let w_t be the number of workers used in week $t \in T$

Maximize: $\sum_{i \in T} \sum_{t \in Y_i} x_{it}$

Subject to: $\sum_{t \in Y_i} x_{it} = 1, \forall i \in I$

$w_t = \sum_{i \in I} \sum_{u \in Y i : u \leq t \leq (u + p_i)} l_{i(t-u+1)} x_{iu}, \forall t \in T$

$w_t \leq L_t, \forall t \in T$

$\sum_{u=1}^{t-1} x_{1u} \geq x_{3t}, t = 3, \cdots, 9$

$x_{31} = x_{32} = 0$

$x_{it} \in \{0, 1\}, \forall i \in I, t \in Y_i$

IP AMPL Code

Model

```
param jobs = 5;
param total_weeks = 9;
param max_job_length = 4;
param max_workers;

param job_workers{i in 1..jobs, j in 1..max_job_length};
param job_length{i in 1..jobs};

var x{i in 1..jobs, t in 1..total_weeks} binary;
#1 if job i starts in week t,otherwise
var M;

set possible_start_weeks{i in 1..jobs}:= {t in 1..total_weeks: 1 <= t + job_length[
set active_weeks{i in 1..jobs,t in possible_start_weeks[i]}:= {u in possible_start_

minimize Job1_Job3{i in 1..jobs}:
sum{t in 1..total_weeks,u in possible_start_weeks[i]}x[i,t];

subject to Active_Job {i in 1..jobs}:
sum{t in possible_start_weeks[i]}x[i,t] = 1;
#Ensure that every job is completed

#subject to Min_Max_Workers_Week{t in 1..total_weeks}:
```

```
#sum{i in 1..jobs, u in 1..max_job_length}job_workers[i,u]*x[i,u] <= M;
#Minimizes the maximum number of workers per week

subject to Job1_bigger_Job3{t in 3..total_weeks}:
sum{u in 1..t-1}x[1,u] >= x[3,t];

subject to Job_3:
x[3,1]-x[3,2]=0;
#Saying job 3 cannot start in weeks 1 or 2
```

Data

```
param job_workers: 1 2 3 4 :=
1 2 3 1 0
2 2 4 5 0
3 2 4 1 5
4 3 4 2 2
5 9 2 3 0;

param job_length := 1 3 2 2 3 4 4 4 5 3;
```

IP AMPL Solution

```
ampl: model problem3cmod.mod;
ampl: data problem3cdat.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: optimal solution; objective 6
Objective = Job1_Job3[1]
```

(d) Formulate and solve with the requirement that Job 4 must start no later than one week after Job 5.

l_{iu} : number of workers required in the u-th week of doing job i
L_t : number of workers available in week t
p_i : number of weeks to complete job i

IP Program:
Let $I = \{1, 2, 3, 4, 5\}$ be the set of all jobs
Let $T = \{1, \cdots, 9\}$ be the set of all weeks
Let $Y_i = \{t : (t + pi - 1) \in T\}$ be the set of possible starting weeks for job i

Let $x_{it} = \begin{cases} 1 & \text{if job } i \text{ starts in week } t \\ 0 & \text{otherwise} \end{cases} \forall i \in I, t \in Y_i.$
Let w_t be the number of workers used in week $t \in T$

Maximize: $\sum_{i \in T} \sum_{t \in Y_i} x_{it}$
Subject to: $\sum_{t \in Y_i} x_{it} = 1, \forall i \in I$
$w_t = \sum_{i \in I} \sum_{u \in Y i : u \leq t \leq (u+p_i)} l_{i(t-u+1)} x_{iu}, \forall t \in T$
$w_t \leq L_t, \forall t \in T$
$\sum_{u=1}^{t+1} x_{4u} \geq x_{5t}, t = 1, \cdots, 8$
$x_{it} \in \{0, 1\}, \forall i \in I, t \in Y_i$

IP AMPL Code
Model

```
param jobs = 5;
param total_weeks = 9;
param max_job_length = 4;
param max_workers;

param job_workers{i in 1..jobs, j in 1..max_job_length};
param job_length{i in 1..jobs};

var x{i in 1..jobs, t in 1..total_weeks} binary;
#1 if job i starts in week t,otherwise
var M;

set possible_start_weeks{i in 1..jobs}:= {t in 1..total_weeks: 1 <= t + job_length[
set active_weeks{i in 1..jobs,t in possible_start_weeks[i]}:= {u in possible_start_

minimize Job4_Job5{i in 1..jobs}:
sum{t in 1..total_weeks,u in possible_start_weeks[i]}x[i,t];

subject to Active_Job {i in 1..jobs}:
sum{t in possible_start_weeks[i]}x[i,t] = 1;
#Ensure that every job is completed

#subject to Min_Max_Workers_Week{t in 1..total_weeks}:
#sum{i in 1..jobs, u in 1..max_job_length}job_workers[i,u]*x[i,u] <= M;
#Minimizes the maximum number of workers per week
```

```
subject to Job4_bigger_Job5{t in 1..(total_weeks-1)}:
sum{u in 1..(t+1)}x[4,u] >= x[5,t];
```

Data

```
param job_workers: 1 2 3 4 :=
1 2 3 1 0
2 2 4 5 0
3 2 4 1 5
4 3 4 2 2
5 9 2 3 0;

param job_length := 1 3 2 2 3 4 4 4 5 3;
```

IP AMPL Solution

```
ampl: model problem3dmod.mod;
ampl: data problem3ddat.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: optimal solution; objective 6
Objective = Job4_Job5[1]
```

(e) Formulate and solve with the requirement that Jobs 1 and 2 both require the same machine and cannot be carried out simultaneously. (Hint: This means that either Job 1 must complete before Job 2 starts or Job 2 must complete before Job 1 starts.)

l_{iu} : number of workers required in the u-th week of doing job i
L_t : number of workers available in week t
p_i : number of weeks to complete job i

IP Program:
Let $I = \{1, 2, 3, 4, 5\}$ be the set of all jobs
Let $T = \{1, \cdots, 9\}$ be the set of all weeks
Let $Y_i = \{t : (t + pi - 1) \in T\}$ be the set of possible starting weeks for job i
Let $x_{it} = \begin{cases} 1 & \text{if job } i \text{ starts in week } t \\ 0 & \text{otherwise} \end{cases} \forall i \in I, t \in Y_i.$
Let w_t be the number of workers used in week $t \in T$
Let $a_{it} = \begin{cases} 1 & \text{if job } i \text{ is active in week } t \\ 0 & \text{otherwise} \end{cases} \forall i \in I$

Maximize: $\sum_{i \in T} \sum_{t \in Y_i} x_{it}$
Subject to: $\sum_{t \in Y_i} x_{it} = 1, \forall i \in I$
$w_t = \sum_{i \in I} \sum_{u \in Y i : u \leq t \leq (u+p_i)} l_{i(t-u+1)} x_{iu}, \forall t \in T$
$w_t \leq L_t, \forall t \in T$
$a_{1t} \geq \sum_{u=1}^{max\{1,(t-p_1+1)\}} x_{1u}, \forall t \in T$
$a_{2t} \geq \sum_{u=1}^{max\{1,(t-p_2+1)\}} x_{2u}, \forall t \in T$
$a_{1t} + a_{2t} \leq 1, \forall t \in T$
$x_{it} \in \{0,1\}, \forall i \in I, t \in Y_i$

IP AMPL Code
Model

```
param jobs = 5;
param total_weeks = 9;
param max_job_length = 4;
param max_workers;

param job_workers{i in 1..jobs, j in 1..max_job_length};
param job_length{i in 1..jobs};

param m1{t in 1..total_weeks}= max(1,t-job_length[1]+1);
param m2{t in 1..total_weeks}= max(1,t-job_length[2]+1);

var x{i in 1..jobs, t in 1..total_weeks} binary;
#1 if job i starts in week t,otherwise
var a{i in 1..jobs, t in 1..total_weeks} binary;

set possible_start_weeks{i in 1..jobs}:= {t in 1..total_weeks: 1 <= t + job_length[
set active_weeks{i in 1..jobs,t in possible_start_weeks[i]}:= {u in possible_start_

minimize Job1_Job3{i in 1..jobs}:
sum{t in 1..total_weeks,u in possible_start_weeks[i]}x[i,t];

subject to Active_Job {i in 1..jobs}:
sum{t in possible_start_weeks[i]}x[i,t] = 1;
#Ensure that every job is completed

#subject to Min_Max_Workers_Week{t in 1..total_weeks}:
#sum{i in 1..jobs, u in 1..max_job_length}job_workers[i,u]*x[i,u] <= M;
#Minimizes the maximum number of workers per week
```

```
subject to Job1_Requirements{t in 1..total_weeks}:
a[1,t]>=sum{u in 1..m1[t]}x[1,u];

subject to Job2_Requirements{t in 1..total_weeks}:
a[2,t]>=sum{u in 1..m2[t]}x[2,u];

subject to Job1_Job2_Relation{t in 1..total_weeks}:
a[1,t]+a[2,t] <= 1;
```

Data

```
param job_workers: 1 2 3 4 :=
1 2 3 1 0
2 2 4 5 0
3 2 4 1 5
4 3 4 2 2
5 9 2 3 0;

param job_length := 1 3 2 2 3 4 4 4 5 3;
```

IP AMPL Solution

```
ampl: model problem3emod.mod;
ampl: data problem3edat.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: infeasible
No primal variables returned.
Objective = Job1_Job3[1]
```

4. The 0-1 knapsack problem is *NP*-complete.

   (a) Formulate the decision form of the 0-1 knapsack problem.

   Let$N = \{1, \cdots, N\}$ be the set of possible items
   Let w_k = the weight of each type-k item, $\forall k \in N$
   Let r_k = the value associated with each type-k item, $\forall k \in N$
   Let c = the weight capacity of the knapsack
   Let $x_k = \begin{cases} 1 & \text{if item } k \text{ is packed} \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in N.$

   Is there an x such that,

$$\sum_{k=1}^{N} r_k x_k \geq v$$
$$\sum_{k=1}^{N} w_k x_k \leq c$$
$$x_k \in \{0, 1\}$$

(b) The 2-PARTITION problem is specified by n positive integers $(a_1, a_2, \ldots, a_n)$. The problem is to find a subset $S \subseteq N = \{1, 2, \ldots, n\}$ such that

$$\sum_{j \in S} a_j = \sum_{j \in N \setminus S} a_j$$

or prove that it is impossible. Show that 2-PARTITION is polynomially reducible to 0-1 KNAPSACK. Does this imply that 2-PARTITION is *NP*-complete?

Let$N = \{1, \cdots, N\}$ be the set of possible items
Let $w_k =$ the weight of each type-k item, $\forall k \in N$
Let $r_k =$ the value associated with each type-k item, $\forall k \in N$
Let $c =$ the weight capacity of the knapsack
Let $x_k = \begin{cases} 1 & \text{if item } k \text{ is packed} \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in N.$

Is there an x such that,

$$\sum_{k=1}^{N} w_k \leq c$$
$$\sum_{k=1}^{N} r_k \geq v$$
$$x_k \in \{0, 1\}$$

Which implies, $\sum_{j \in N} a_j \leq t$
$\sum_{j \in N} a_j \geq t$

Therefore, we get the following,

Reduction LP Model

$$\sum_{j \in N} a_j = t$$

This does not necessarily imply that 2-partition is NP-Complete. It would be NP-Complete if the $0-1$ knapsack problem is also polynomially reducible to 2-partition.

5. The problem SET COVERING is *NP*-complete. Show that SET COVERING is polynomially reducible to the uncapacitated facility location problem (UFL). (Hint: Think of UFL as minimizing the cost of selecting a subset S of locations, where the cost is given by

$$C(S) = \sum_{i=1}^{n} \min_{j \in S} c_{ij} + \sum_{j \in S} f_j.$$

That is, think about both problems as problems involving sets rather than trying to think about the integer programming formulations.

Basic IP for Set Covering:

Let $N = \{1, \cdots, n\}$
Let $M = \{1, \cdots, m\}$
Let $x_j = \begin{cases} 1 & \text{if activity } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}, \forall j \in N.$
Let $d_j =$ cost of activity $j \in N$.
Let $y_i j = \begin{cases} 1 & \text{if activity } j \text{ covers requirement } i \\ 0 & \text{otherwise} \end{cases}, \forall i \in M.$

Minimize:
$\sum_{j \in N} d_j x_j$
Subject to:
$\sum_{j \in N} y_{ij} x_j \geq 1,\ i \in M.$
$x_j \in \{0, 1\}, j \in N.$

Basic IP for UFL:

Let $N = \{1, \cdots, n\}$
Let $M = \{1, \cdots, m\}$

Let f_i be the fixed cost of opening facility i
Let c_{ij} be the cost of service from facility j to facility i
Let $x_j = \begin{cases} 1 & \text{if facility } j \text{ is opened} \\ 0 & \text{otherwise} \end{cases}, \forall j \in N.$
Let $a_{ij} = \begin{cases} 1 & \text{if facility } i \text{ is served by facility } j \\ 0 & \text{otherwise} \end{cases}, \forall j \in N.$

Minimize:
$\sum_{j \in N} f_i x_i + \sum_{i \in M} \sum_{j \in N} c_{ij} a_{ij}$
Subject to:
$\sum_{j \in N} a_{ij} = 1, i \in M$
$a_{ij} \leq a_j, i \in M, j \in N$
$a_{ij}, y_{ij} \in \{0, 1\}, i \in M, j \in N$

Let activity j correspond to facility j such that all clients in the set $S_j \subseteq M$.
Let $F_i \subseteq N$ be the subset of locations that can serve client i.
Let d_j, the cost of activity j, be the cost f_i of opening facility j
Let $c_{ij} = 0$, for all $i \in N, j \in M$
Let $a_{ij} = \begin{cases} 1 & \text{if } i \in S_j \\ 0 & \text{otherwise} \end{cases}, \forall j \in N.$
Let $b_{ij} = \begin{cases} 1 & \text{if facility} i \text{is satisfied by} j \\ 0 & \text{otherwise} \end{cases}, \forall j \in N.$

Reduction LP Model:

Minimize:
$\sum_{j \in N} f_i x_i + \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} = \sum_{j \in N} f_i x_i$
Subject to:
$\sum_{j \in F_i} b_{ij} = 1, i \in M$
$b_{ij}, x_j \in \{0, 1\}, i \in M, j \in N$