1. Doodle Scheduling [10 pts]

by Roumen Guha, on Sunday, February 19th, 2017

Doodle Inc. is looking to interview a candidate for a new software engineer position at their company. It works like this: the interview (10 AM to 3 PM) is divided into a number of 20-minute time slots that may be used for 1-on-1 meetings with the candidate. There is also a one-hour time slot in the middle of the day where 3 employees take the candidate out for lunch. It would be nice for all 15 senior employees to meet with the candidate at some point during the day, but everybody has a busy schedule so it's not clear whether this will be possible. A doodle poll (obviously) was sent to the 15 senior employees to figure out their availability.

	10:00	10:20	10:40	11:00	11:20	11:40	Lunch	1:00	1:20	1:40	2:00	2:20	2:40
Manuel	0	0	1	1	0	0	0	1	1	0	0	0	0
Luca	0	1	1	0	0	0	0	0	1	1	0	0	0
Jule	0	0	0	1	1	0	1	1	0	1	1	1	1
Michael	0	0	0	1	1	1	1	1	1	1	1	1	0
Malte	0	0	0	0	0	0	1	1	1	0	0	0	0
Chris	0	1	1	0	0	0	0	0	1	1	0	0	0
Spyros	0	0	0	1	1	1	1	0	0	0	0	0	0
Mirjam	1	1	0	0	0	0	0	0	0	0	1	1	1
Matt	1	1	1	0	0	0	0	0	0	1	1	0	0
Florian	0	0	0	0	0	0	0	1	1	0	0	0	0
Josep	0	0	0	0	0	0	1	1	1	0	0	0	0
Joel	1	1	0	0	0	1	1	1	1	0	0	1	1
Tom	1	1	1	0	1	1	0	0	0	0	0	1	1
Daniel	0	1	1	1	0	0	0	0	0	0	0	0	0
Anne	1	1	0	0	1	1	0	0	0	0	0	0	0

In the table, a 1 means that the employee is available at the indicated time, while a 0 means that they are unavailable. Determine whether a feasible interview schedule exists. If so, print out a calendar for the candidate that lists who they will be meeting at each time slot.

In [6]:
```julia
using NamedArrays

# import data set
raw = readcsv("1b.csv")       # A file I made
(m, n) = size(raw)

n_times = 2:n      # columns containing times
n_names = 2:m      # rows containing names

times = raw[1, n_times][:]   # the list of times (convert to 1-D array)
names = raw[n_names, 1][:]   # the list of names (convert to 1-D array)

# data[f,i] is the entire schedule
data = NamedArray(raw[n_names, n_times], (names, times), ("names", "times"))

show(IOContext(STDOUT, displaysize=(100, 1000)), data)
```

```
15×15 Named Array{Any,2}
names ╲ times │ 10:00  10:20  10:40  11:00  11:20  11:40  12:00  12:20  12:40  13:00  13:20  13:40  14:00  14:20  14:40
──────────────┼────────────────────────────────────────────────────────────────────────────────────────────────────────
Manuel        │     0      0      1      1      0      0      0      0      0      1      1      0      0      0      0
Luca          │     0      1      1      0      0      0      0      0      0      0      1      1      0      0      0
Jule          │     0      0      0      1      1      0      1      1      1      1      0      1      1      1      1
Michael       │     0      0      0      1      1      1      1      1      1      1      1      1      1      1      0
Malte         │     0      0      0      0      0      0      1      1      1      1      1      0      0      0      0
Chris         │     0      1      1      0      0      0      0      0      0      0      1      1      0      0      0
Spyros        │     0      0      0      1      1      1      1      1      1      0      0      0      0      0      0
Mirjam        │     1      1      0      0      0      0      0      0      0      0      0      0      1      1      1
Matt          │     1      1      1      0      0      0      0      0      0      0      0      1      1      0      0
Florian       │     0      0      0      0      0      0      0      0      0      1      1      0      0      0      0
Josep         │     0      0      0      0      0      0      1      1      1      1      1      0      0      0      0
Joel          │     1      1      0      0      0      1      1      1      1      1      1      0      0      1      1
Tom           │     1      1      1      0      1      1      0      0      0      0      0      0      0      1      1
Daniel        │     0      1      1      1      0      0      0      0      0      0      0      0      0      0      0
Anne          │     1      1      0      0      1      1      0      0      0      0      0      0      0      0      0
```

In [7]:
```
using JuMP

m = Model()

@variable(m, x[names, times] >= 0)

# the candidate can only meet with 1 employee at any 20-minute period
@constraint(m, a[j in times], sum(x[i,j] for i in names) <= 1)

# each employee can only meet with the candidate once
@constraint(m, b[i in names], sum(x[i,j] for j in times) <= 1)

@objective(m, Max, sum(x[i,j]*data[i,j] for i in names, j in times))

status = solve(m)
println(status)

schedule = NamedArray( [Int(getvalue(x[i,j])) for i in names, j in times], (names, times), ("Names", "Times"))

show(IOContext(STDOUT, displaysize=(100, 1000)), schedule)
```

```
Optimal
15×15 Named Array{Int64,2}
Names ╲ Times │ 10:00  10:20  10:40  11:00  11:20  11:40  12:00  12:20  12:40  13:00  13:20  13:40  14:00  14:20  14:40
──────────────┼──────────────────────────────────────────────────────────────────────────────────────────────────────────
Manuel        │     0      0      1      0      0      0      0      0      0      0      0      0      0      0      0
Luca          │     0      1      0      0      0      0      0      0      0      0      0      0      0      0      0
Jule          │     0      0      0      0      0      0      1      0      0      0      0      0      0      0      0
Michael       │     0      0      0      0      0      0      0      1      0      0      0      0      0      0      0
Malte         │     0      0      0      0      0      0      0      0      1      0      0      0      0      0      0
Chris         │     0      0      0      0      0      0      0      0      0      0      0      1      0      0      0
Spyros        │     0      0      0      0      1      0      0      0      0      0      0      0      0      0      0
Mirjam        │     0      0      0      0      0      0      0      0      0      0      0      0      0      1      0
Matt          │     0      0      0      0      0      0      0      0      0      0      0      0      1      0      0
Florian       │     0      0      0      0      0      0      0      0      0      0      1      0      0      0      0
Josep         │     0      0      0      0      0      0      0      0      0      1      0      0      0      0      0
Joel          │     1      0      0      0      0      0      0      0      0      0      0      0      0      0      0
Tom           │     0      0      0      0      0      0      0      0      0      0      0      0      0      0      1
Daniel        │     0      0      0      1      0      0      0      0      0      0      0      0      0      0      0
Anne          │     0      0      0      0      0      1      0      0      0      0      0      0      0      0      0
```

Note:

To account for the lunch period, I simply split the lunch hour into 3 separate 20-minute periods with identical data.

In [9]:
```
println("Schedule is as follows: ")
println()

for j = 1:15
    for i = 1:15
        if schedule[i,j] == 1
            print(times[j])
            print(": ")
            println(names[i])
        end
    end
end
```

```
Schedule is as follows: 

10:00: Joel
10:20: Luca
10:40: Manuel
11:00: Daniel
11:20: Spyros
11:40: Anne
12:00: Jule
12:20: Michael
12:40: Malte
13:00: Josep
13:20: Florian
13:40: Chris
14:00: Matt
14:20: Mirjam
14:40: Tom
```

This means that Jule, Michael and Malte get to join the candidate for lunch. Lucky them.

In [11]:
```
println(m)
```

```
Max x[Manuel,10:40] + x[Manuel,11:00] + x[Manuel,13:00] + x[Manuel,13:20] + x[Luca,10:20] + x[Luca,10:40] + x[Luca,13:20] + x[Luca,13:40] + x[Jule,11:00] + x[Jule,11:20] + x[Jule,1
2:00] + x[Jule,12:20] + x[Jule,12:40] + x[Jule,13:00] + x[Jule,13:40] + x[Jule,14:00] + x[Jule,14:20] + x[Jule,14:40] + x[Michael,11:00] + x[Michael,11:20] + x[Michael,11:40] + x[M
ichael,12:00] + x[Michael,12:20] + x[Michael,12:40] + x[Michael,13:00] + x[Michael,13:20] + x[Michael,13:40] + x[Michael,14:00] + x[Michael,14:20] + x[Malte,12:00] + x[Malte,12:20]
 + x[Malte,12:40] + x[Malte,13:00] + x[Malte,13:20] + x[Chris,10:20] + x[Chris,10:40] + x[Chris,13:20] + x[Chris,13:40] + x[Spyros,11:00] + x[Spyros,11:20] + x[Spyros,11:40] + x[Spy
ros,12:00] + x[Spyros,12:20] + x[Spyros,12:40] + x[Mirjam,10:00] + x[Mirjam,10:20] + x[Mirjam,14:00] + x[Mirjam,14:20] + x[Mirjam,14:40] + x[Matt,10:00] + x[Matt,10:20] + x[Matt,1
0:40] + x[Matt,13:40] + x[Matt,14:00] + x[Florian,13:00] + x[Florian,13:20] + x[Josep,12:00] + x[Josep,12:20] + x[Josep,12:40] + x[Josep,13:00] + x[Josep,13:20] + x[Joel,10:00] + x
[Joel,10:20] + x[Joel,11:40] + x[Joel,12:00] + x[Joel,12:20] + x[Joel,12:40] + x[Joel,13:00] + x[Joel,13:20] + x[Joel,14:20] + x[Joel,14:40] + x[Tom,10:00] + x[Tom,10:20] + x[Tom,1
0:40] + x[Tom,11:20] + x[Tom,11:40] + x[Tom,14:20] + x[Tom,14:40] + x[Daniel,10:20] + x[Daniel,10:40] + x[Daniel,11:00] + x[Anne,10:00] + x[Anne,10:20] + x[Anne,11:20] + x[Anne,11:
40]
Subject to
 x[Manuel,10:00] + x[Luca,10:00] + x[Jule,10:00] + x[Michael,10:00] + x[Malte,10:00] + x[Chris,10:00] + x[Spyros,10:00] + x[Mirjam,10:00] + x[Matt,10:00] + x[Florian,10:00] + x[Jos
ep,10:00] + x[Joel,10:00] + x[Tom,10:00] + x[Daniel,10:00] + x[Anne,10:00] <= 1
 x[Manuel,10:20] + x[Luca,10:20] + x[Jule,10:20] + x[Michael,10:20] + x[Malte,10:20] + x[Chris,10:20] + x[Spyros,10:20] + x[Mirjam,10:20] + x[Matt,10:20] + x[Florian,10:20] + x[Jos
ep,10:20] + x[Joel,10:20] + x[Tom,10:20] + x[Daniel,10:20] + x[Anne,10:20] <= 1
 x[Manuel,10:40] + x[Luca,10:40] + x[Jule,10:40] + x[Michael,10:40] + x[Malte,10:40] + x[Chris,10:40] + x[Spyros,10:40] + x[Mirjam,10:40] + x[Matt,10:40] + x[Florian,10:40] + x[Jos
ep,10:40] + x[Joel,10:40] + x[Tom,10:40] + x[Daniel,10:40] + x[Anne,10:40] <= 1
 x[Manuel,11:00] + x[Luca,11:00] + x[Jule,11:00] + x[Michael,11:00] + x[Malte,11:00] + x[Chris,11:00] + x[Spyros,11:00] + x[Mirjam,11:00] + x[Matt,11:00] + x[Florian,11:00] + x[Jos
ep,11:00] + x[Joel,11:00] + x[Tom,11:00] + x[Daniel,11:00] + x[Anne,11:00] <= 1
 x[Manuel,11:20] + x[Luca,11:20] + x[Jule,11:20] + x[Michael,11:20] + x[Malte,11:20] + x[Chris,11:20] + x[Spyros,11:20] + x[Mirjam,11:20] + x[Matt,11:20] + x[Florian,11:20] + x[Jos
ep,11:20] + x[Joel,11:20] + x[Tom,11:20] + x[Daniel,11:20] + x[Anne,11:20] <= 1
 x[Manuel,11:40] + x[Luca,11:40] + x[Jule,11:40] + x[Michael,11:40] + x[Malte,11:40] + x[Chris,11:40] + x[Spyros,11:40] + x[Mirjam,11:40] + x[Matt,11:40] + x[Florian,11:40] + x[Jos
ep,11:40] + x[Joel,11:40] + x[Tom,11:40] + x[Daniel,11:40] + x[Anne,11:40] <= 1
 x[Manuel,12:00] + x[Luca,12:00] + x[Jule,12:00] + x[Michael,12:00] + x[Malte,12:00] + x[Chris,12:00] + x[Spyros,12:00] + x[Mirjam,12:00] + x[Matt,12:00] + x[Florian,12:00] + x[Jos
ep,12:00] + x[Joel,12:00] + x[Tom,12:00] + x[Daniel,12:00] + x[Anne,12:00] <= 1
 x[Manuel,12:20] + x[Luca,12:20] + x[Jule,12:20] + x[Michael,12:20] + x[Malte,12:20] + x[Chris,12:20] + x[Spyros,12:20] + x[Mirjam,12:20] + x[Matt,12:20] + x[Florian,12:20] + x[Jos
ep,12:20] + x[Joel,12:20] + x[Tom,12:20] + x[Daniel,12:20] + x[Anne,12:20] <= 1
 x[Manuel,12:40] + x[Luca,12:40] + x[Jule,12:40] + x[Michael,12:40] + x[Malte,12:40] + x[Chris,12:40] + x[Spyros,12:40] + x[Mirjam,12:40] + x[Matt,12:40] + x[Florian,12:40] + x[Jos
ep,12:40] + x[Joel,12:40] + x[Tom,12:40] + x[Daniel,12:40] + x[Anne,12:40] <= 1
 x[Manuel,13:00] + x[Luca,13:00] + x[Jule,13:00] + x[Michael,13:00] + x[Malte,13:00] + x[Chris,13:00] + x[Spyros,13:00] + x[Mirjam,13:00] + x[Matt,13:00] + x[Florian,13:00] + x[Jos
ep,13:00] + x[Joel,13:00] + x[Tom,13:00] + x[Daniel,13:00] + x[Anne,13:00] <= 1
 x[Manuel,13:20] + x[Luca,13:20] + x[Jule,13:20] + x[Michael,13:20] + x[Malte,13:20] + x[Chris,13:20] + x[Spyros,13:20] + x[Mirjam,13:20] + x[Matt,13:20] + x[Florian,13:20] + x[Jos
ep,13:20] + x[Joel,13:20] + x[Tom,13:20] + x[Daniel,13:20] + x[Anne,13:20] <= 1
 x[Manuel,13:40] + x[Luca,13:40] + x[Jule,13:40] + x[Michael,13:40] + x[Malte,13:40] + x[Chris,13:40] + x[Spyros,13:40] + x[Mirjam,13:40] + x[Matt,13:40] + x[Florian,13:40] + x[Jos
ep,13:40] + x[Joel,13:40] + x[Tom,13:40] + x[Daniel,13:40] + x[Anne,13:40] <= 1
 x[Manuel,14:00] + x[Luca,14:00] + x[Jule,14:00] + x[Michael,14:00] + x[Malte,14:00] + x[Chris,14:00] + x[Spyros,14:00] + x[Mirjam,14:00] + x[Matt,14:00] + x[Florian,14:00] + x[Jos
ep,14:00] + x[Joel,14:00] + x[Tom,14:00] + x[Daniel,14:00] + x[Anne,14:00] <= 1
 x[Manuel,14:20] + x[Luca,14:20] + x[Jule,14:20] + x[Michael,14:20] + x[Malte,14:20] + x[Chris,14:20] + x[Spyros,14:20] + x[Mirjam,14:20] + x[Matt,14:20] + x[Florian,14:20] + x[Jos
ep,14:20] + x[Joel,14:20] + x[Tom,14:20] + x[Daniel,14:20] + x[Anne,14:20] <= 1
 x[Manuel,14:40] + x[Luca,14:40] + x[Jule,14:40] + x[Michael,14:40] + x[Malte,14:40] + x[Chris,14:40] + x[Spyros,14:40] + x[Mirjam,14:40] + x[Matt,14:40] + x[Florian,14:40] + x[Jos
ep,14:40] + x[Joel,14:40] + x[Tom,14:40] + x[Daniel,14:40] + x[Anne,14:40] <= 1
 x[Manuel,10:00] + x[Manuel,10:20] + x[Manuel,10:40] + x[Manuel,11:00] + x[Manuel,11:20] + x[Manuel,11:40] + x[Manuel,12:00] + x[Manuel,12:20] + x[Manuel,12:40] + x[Manuel,13:00] +
x[Manuel,13:20] + x[Manuel,13:40] + x[Manuel,14:00] + x[Manuel,14:20] + x[Manuel,14:40] <= 1
 x[Luca,10:00] + x[Luca,10:20] + x[Luca,10:40] + x[Luca,11:00] + x[Luca,11:20] + x[Luca,11:40] + x[Luca,12:00] + x[Luca,12:20] + x[Luca,12:40] + x[Luca,13:00] + x[Luca,13:20] + x[L
uca,13:40] + x[Luca,14:00] + x[Luca,14:20] + x[Luca,14:40] <= 1
 x[Jule,10:00] + x[Jule,10:20] + x[Jule,10:40] + x[Jule,11:00] + x[Jule,11:20] + x[Jule,11:40] + x[Jule,12:00] + x[Jule,12:20] + x[Jule,12:40] + x[Jule,13:00] + x[Jule,13:20] + x[J
ule,13:40] + x[Jule,14:00] + x[Jule,14:20] + x[Jule,14:40] <= 1
 x[Michael,10:00] + x[Michael,10:20] + x[Michael,10:40] + x[Michael,11:00] + x[Michael,11:20] + x[Michael,11:40] + x[Michael,12:00] + x[Michael,12:20] + x[Michael,12:40] + x[Michae
l,13:00] + x[Michael,13:20] + x[Michael,13:40] + x[Michael,14:00] + x[Michael,14:20] + x[Michael,14:40] <= 1
 x[Malte,10:00] + x[Malte,10:20] + x[Malte,10:40] + x[Malte,11:00] + x[Malte,11:20] + x[Malte,11:40] + x[Malte,12:00] + x[Malte,12:20] + x[Malte,12:40] + x[Malte,13:00] + x[Malte,1
3:20] + x[Malte,13:40] + x[Malte,14:00] + x[Malte,14:20] + x[Malte,14:40] <= 1
 x[Chris,10:00] + x[Chris,10:20] + x[Chris,10:40] + x[Chris,11:00] + x[Chris,11:20] + x[Chris,11:40] + x[Chris,12:00] + x[Chris,12:20] + x[Chris,12:40] + x[Chris,13:00] + x[Chris,1
3:20] + x[Chris,13:40] + x[Chris,14:00] + x[Chris,14:20] + x[Chris,14:40] <= 1
 x[Spyros,10:00] + x[Spyros,10:20] + x[Spyros,10:40] + x[Spyros,11:00] + x[Spyros,11:20] + x[Spyros,11:40] + x[Spyros,12:00] + x[Spyros,12:20] + x[Spyros,12:40] + x[Spyros,13:00] +
x[Spyros,13:20] + x[Spyros,13:40] + x[Spyros,14:00] + x[Spyros,14:20] + x[Spyros,14:40] <= 1
 x[Mirjam,10:00] + x[Mirjam,10:20] + x[Mirjam,10:40] + x[Mirjam,11:00] + x[Mirjam,11:20] + x[Mirjam,11:40] + x[Mirjam,12:00] + x[Mirjam,12:20] + x[Mirjam,12:40] + x[Mirjam,13:00] +
x[Mirjam,13:20] + x[Mirjam,13:40] + x[Mirjam,14:00] + x[Mirjam,14:20] + x[Mirjam,14:40] <= 1
 x[Matt,10:00] + x[Matt,10:20] + x[Matt,10:40] + x[Matt,11:00] + x[Matt,11:20] + x[Matt,11:40] + x[Matt,12:00] + x[Matt,12:20] + x[Matt,12:40] + x[Matt,13:00] + x[Matt,13:20] + x[M
att,13:40] + x[Matt,14:00] + x[Matt,14:20] + x[Matt,14:40] <= 1
 x[Florian,10:00] + x[Florian,10:20] + x[Florian,10:40] + x[Florian,11:00] + x[Florian,11:20] + x[Florian,11:40] + x[Florian,12:00] + x[Florian,12:20] + x[Florian,12:40] + x[Floria
n,13:00] + x[Florian,13:20] + x[Florian,13:40] + x[Florian,14:00] + x[Florian,14:20] + x[Florian,14:40] <= 1
 x[Josep,10:00] + x[Josep,10:20] + x[Josep,10:40] + x[Josep,11:00] + x[Josep,11:20] + x[Josep,11:40] + x[Josep,12:00] + x[Josep,12:20] + x[Josep,12:40] + x[Josep,13:00] + x[Josep,1
3:20] + x[Josep,13:40] + x[Josep,14:00] + x[Josep,14:20] + x[Josep,14:40] <= 1
 x[Joel,10:00] + x[Joel,10:20] + x[Joel,10:40] + x[Joel,11:00] + x[Joel,11:20] + x[Joel,11:40] + x[Joel,12:00] + x[Joel,12:20] + x[Joel,12:40] + x[Joel,13:00] + x[Joel,13:20] + x[J
oel,13:40] + x[Joel,14:00] + x[Joel,14:20] + x[Joel,14:40] <= 1
 x[Tom,10:00] + x[Tom,10:20] + x[Tom,10:40] + x[Tom,11:00] + x[Tom,11:20] + x[Tom,11:40] + x[Tom,12:00] + x[Tom,12:20] + x[Tom,12:40] + x[Tom,13:00] + x[Tom,13:20] + x[Tom,13:40] +
x[Tom,14:00] + x[Tom,14:20] + x[Tom,14:40] <= 1
 x[Daniel,10:00] + x[Daniel,10:20] + x[Daniel,10:40] + x[Daniel,11:00] + x[Daniel,11:20] + x[Daniel,11:40] + x[Daniel,12:00] + x[Daniel,12:20] + x[Daniel,12:40] + x[Daniel,13:00] +
x[Daniel,13:20] + x[Daniel,13:40] + x[Daniel,14:00] + x[Daniel,14:20] + x[Daniel,14:40] <= 1
 x[Anne,10:00] + x[Anne,10:20] + x[Anne,10:40] + x[Anne,11:00] + x[Anne,11:20] + x[Anne,11:40] + x[Anne,12:00] + x[Anne,12:20] + x[Anne,12:40] + x[Anne,13:00] + x[Anne,13:20] + x[A
nne,13:40] + x[Anne,14:00] + x[Anne,14:20] + x[Anne,14:40] <= 1
 x[i,j] >= 0 for all i in {Manuel,Luca,..,Daniel,Anne}, j in {10:00,10:20,..,14:20,14:40}
```

2. Car Rental [10 pts]

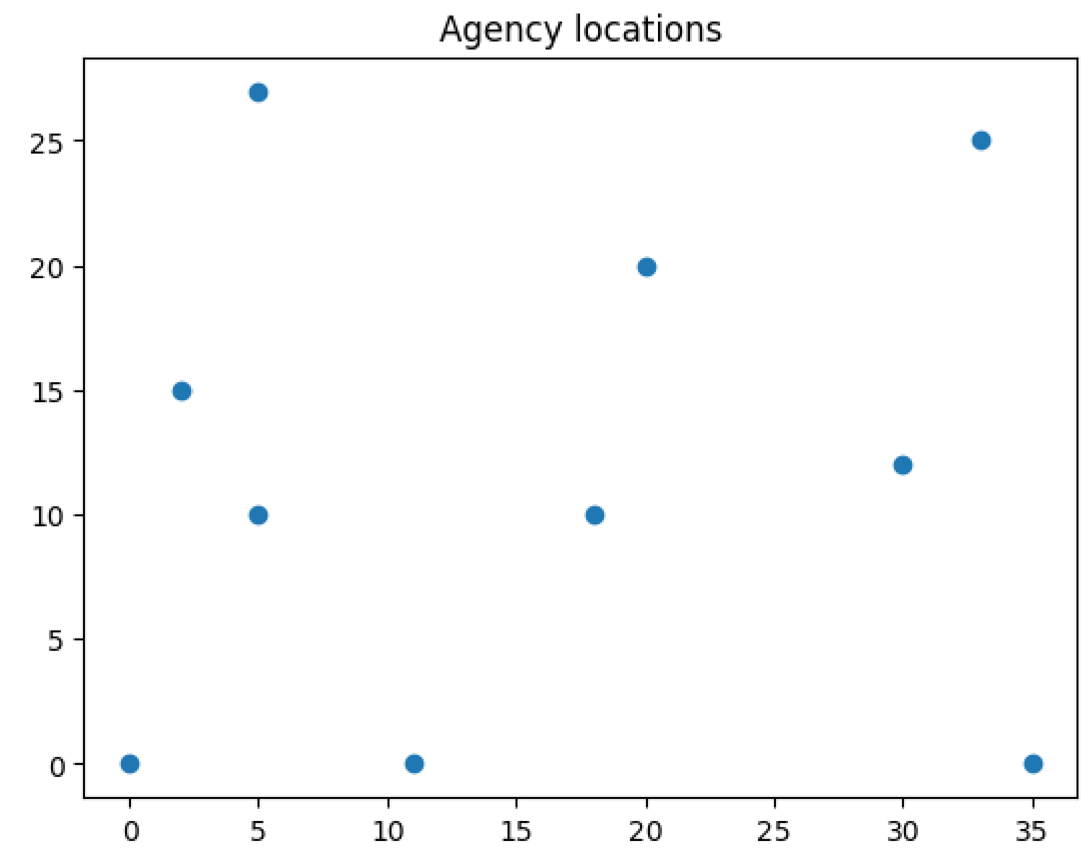
by Roumen Guha, on Sunday, February 19th, 2017

A small car rental company has a fleet of 94 vehicles distributed among its 10 agencies. The location of every agency is given by its geographical coordinates x and y in a grid based on miles. We assume that the road distance between agencies is approximately 1.3 times the Euclidean distance (as the crow flies). The following table indicates the coordinates of all agencies, the number of cars required the next morning, and the stock of cars in the evening preceding this day.

Agency number	1	2	3	4	5	6	7	8	9	10
x coordinate	0	20	18	30	35	33	5	5	11	2
y coordinate	0	20	10	12	0	25	27	10	0	15
Required cars	10	6	8	11	9	7	15	7	9	12
Cars present	8	13	4	8	12	2	14	11	15	7

Supposing the cost for transporting a car is $0.50 per mile, determine the movements of cars that allow the company to re-establish the required numbers of cars at all agencies, minimizing the total cost incurred for transport.

In [6]:
```
using NamedArrays, PyPlot

agencies = [1:10;]

coords =  [ 0 20 18 30 35 33  5  5 11  2
            0 20 10 12  0 25 27 10  0 15 ]

carsReq = [10  6  8 11  9  7 15  7  9 12]

carsPres = [8 13  4  8 12  2 14 11 15  7]

distMultiplier = 1.3
mileCost       = 0.5 # dollars
;

scatter(coords[1,:], coords[2,:])
title("Agency locations")
```



Out[6]:
```
PyObject <matplotlib.text.Text object at 0x0000000028ED4438>
```

In [101]:
```
displacements = zeros(10,10)

for source = 1:10
    for destination = 1:10
        # displacement = sqrt((x2 - x1)^2 + (y2 - y1)^2)
        displacements[source,destination] =  sqrt((coords[1,destination] - coords[1,source])^2 + (coords[2,destination] - coords[2,source])^2)
    end
end

distances = displacements * distMultiplier
costs = distances * mileCost
;
```

In [102]:
```
using JuMP

m = Model()

# movements[i,j] is the number of cars shipped from agency i to agency j
@variable(m, movements[agencies, agencies] >= 0)

# each agency must have as many cars as they require the next moring
@constraint(m, present[j in agencies], sum(movements[i,j] for i in agencies) == carsReq[j])
@constraint(m, required[i in agencies], sum(movements[i,j] for j in agencies) == carsPres[i])

@objective(m, Min, sum(movements[i,j]*costs[i,j] for i in agencies, j in agencies))

status = solve(m)
```

Out[102]:
```
:Optimal
```

In [103]:
```
raw = Int[getvalue(movements[i, j]) for i in agencies, j in agencies]
solution = NamedArray(raw, (agencies, agencies), ("from", "to"))
```

Out[103]:
```
10×10 Named Array{Int64,2}
from ╲ to │   1   2   3   4   5   6   7   8   9  10
──────────┼───────────────────────────────────────
1         │   8   0   0   0   0   0   0   0   0   0
2         │   0   6   1   0   0   5   1   0   0   0
3         │   0   0   4   0   0   0   0   0   0   0
4         │   0   0   0   8   0   0   0   0   0   0
5         │   0   0   0   3   9   0   0   0   0   0
6         │   0   0   0   0   0   2   0   0   0   0
7         │   0   0   0   0   0   0  14   0   0   0
8         │   0   0   0   0   0   0   0   6   0   5
9         │   2   0   3   0   0   0   0   1   9   0
10        │   0   0   0   0   0   0   0   0   0   7
```

In [105]:
```
for j = 1:10
    for i = 1:10
        if movement[i,j] >= 1e-5
            if i != j
                print(Int(movement[i,j]))
                println(" car(s) moved to agency ", j, " (from agency ", i, ")")
            end
        end
    end
end

println()
println("The total cost will be \$", getobjectivevalue(m))
```

```
2 car(s) moved to agency 1 (from agency 9)
1 car(s) moved to agency 3 (from agency 2)
3 car(s) moved to agency 3 (from agency 9)
3 car(s) moved to agency 4 (from agency 5)
5 car(s) moved to agency 6 (from agency 2)
1 car(s) moved to agency 7 (from agency 2)
1 car(s) moved to agency 8 (from agency 9)
5 car(s) moved to agency 10 (from agency 8)

The total cost will be $152.63901632295628
```

```
In [106]: println(m)
```

```
Min 18.38477631085024 movements[1,2] + 13.384319183283102 movements[1,3] + 21.002142747824564 movements[1,4] + 22.75 movements[1,5] + 26.91031400782979 movements[1,6] + 17.84838928
306978 movements[1,7] + 7.267220926874317 movements[1,8] + 7.15 movements[1,9] + 9.836284867774012 movements[1,10] + 18.38477631085024 movements[2,1] + 6.62872536767062 movements
[2,3] + 8.324061508662703 movements[2,4] + 16.25 movements[2,5] + 9.053452380169677 movements[2,6] + 10.759414482210452 movements[2,7] + 11.718041645257966 movements[2,8] + 14.2556
12929649851 movements[2,9] + 12.143002099975114 movements[2,10] + 13.384319183283102 movements[3,1] + 6.62872536767062 movements[3,2] + 7.907591289387685 movements[3,4] + 12.820003
900155413 movements[3,5] + 13.788582233137678 movements[3,6] + 13.910607463371253 movements[3,7] + 8.450000000000001 movements[3,8] + 7.934261150226907 movements[3,9] + 10.89598549
9256138 movements[3,10] + 21.002142747824564 movements[4,1] + 8.324061508662703 movements[4,2] + 7.907591289387685 movements[4,3] + 8.450000000000001 movements[4,5] + 8.67208164168
2118 movements[4,6] + 18.950593658247225 movements[4,7] + 16.30191706517979 movements[4,8] + 14.606933285258751 movements[4,9] + 18.304166192427342 movements[4,10] + 22.75 movement
s[5,1] + 16.25 movements[5,2] + 12.820003900155413 movements[5,3] + 8.450000000000001 movements[5,4] + 16.30191706517979 movements[5,6] + 26.234566891793737 movements[5,7] + 20.554
804791094465 movements[5,8] + 15.600000000000001 movements[5,9] + 23.561939648509416 movements[5,10] + 26.91031400782979 movements[6,1] + 9.053452380169677 movements[6,2] + 13.7885
82233137678 movements[6,3] + 8.672081641682118 movements[6,4] + 16.30191706517979 movements[6,5] + 18.246369501903658 movements[6,7] + 20.647094226549168 movements[6,8] + 21.646073
546950728 movements[6,9] + 21.17244671737303 movements[6,10] + 17.84838928306978 movements[7,1] + 10.759414482210452 movements[7,2] + 13.910607463371253 movements[7,3] + 18.9505936
58247225 movements[7,4] + 26.234566891793737 movements[7,5] + 18.246369501903658 movements[7,6] + 11.05 movements[7,8] + 17.978111691721132 movements[7,9] + 8.040055969954437 movem
ents[7,10] + 7.267220926874317 movements[8,1] + 11.718041645257966 movements[8,2] + 8.450000000000001 movements[8,3] + 16.30191706517979 movements[8,4] + 20.554804791094465 movemen
ts[8,5] + 20.647094226549168 movements[8,6] + 11.05 movements[8,7] + 7.580237463298891 movements[8,9] + 3.7901187316494456 movements[8,10] + 7.15 movements[9,1] + 14.2556129296485
1 movements[9,2] + 7.934261150226907 movements[9,3] + 14.606933285258751 movements[9,4] + 15.600000000000001 movements[9,5] + 21.646073546950728 movements[9,6] + 17.978111691721132
movements[9,7] + 7.580237463298891 movements[9,8] + 11.370356194948336 movements[9,10] + 9.836284867774012 movements[10,1] + 12.143002099975114 movements[10,2] + 10.895985499256138
movements[10,3] + 18.304166192427342 movements[10,4] + 23.561939648509416 movements[10,5] + 21.17244671737303 movements[10,6] + 8.040055969954437 movements[10,7] + 3.79011873164944
56 movements[10,8] + 11.370356194948336 movements[10,9]
Subject to
 movements[1,1] + movements[2,1] + movements[3,1] + movements[4,1] + movements[5,1] + movements[6,1] + movements[7,1] + movements[8,1] + movements[9,1] + movements[10,1] == 10
 movements[1,2] + movements[2,2] + movements[3,2] + movements[4,2] + movements[5,2] + movements[6,2] + movements[7,2] + movements[8,2] + movements[9,2] + movements[10,2] == 6
 movements[1,3] + movements[2,3] + movements[3,3] + movements[4,3] + movements[5,3] + movements[6,3] + movements[7,3] + movements[8,3] + movements[9,3] + movements[10,3] == 8
 movements[1,4] + movements[2,4] + movements[3,4] + movements[4,4] + movements[5,4] + movements[6,4] + movements[7,4] + movements[8,4] + movements[9,4] + movements[10,4] == 11
 movements[1,5] + movements[2,5] + movements[3,5] + movements[4,5] + movements[5,5] + movements[6,5] + movements[7,5] + movements[8,5] + movements[9,5] + movements[10,5] == 9
 movements[1,6] + movements[2,6] + movements[3,6] + movements[4,6] + movements[5,6] + movements[6,6] + movements[7,6] + movements[8,6] + movements[9,6] + movements[10,6] == 7
 movements[1,7] + movements[2,7] + movements[3,7] + movements[4,7] + movements[5,7] + movements[6,7] + movements[7,7] + movements[8,7] + movements[9,7] + movements[10,7] == 15
 movements[1,8] + movements[2,8] + movements[3,8] + movements[4,8] + movements[5,8] + movements[6,8] + movements[7,8] + movements[8,8] + movements[9,8] + movements[10,8] == 7
 movements[1,9] + movements[2,9] + movements[3,9] + movements[4,9] + movements[5,9] + movements[6,9] + movements[7,9] + movements[8,9] + movements[9,9] + movements[10,9] == 9
 movements[1,10] + movements[2,10] + movements[3,10] + movements[4,10] + movements[5,10] + movements[6,10] + movements[7,10] + movements[8,10] + movements[9,10] + movements[10,10]
 == 12
 movements[1,1] + movements[1,2] + movements[1,3] + movements[1,4] + movements[1,5] + movements[1,6] + movements[1,7] + movements[1,8] + movements[1,9] + movements[1,10] == 8
 movements[2,1] + movements[2,2] + movements[2,3] + movements[2,4] + movements[2,5] + movements[2,6] + movements[2,7] + movements[2,8] + movements[2,9] + movements[2,10] == 13
 movements[3,1] + movements[3,2] + movements[3,3] + movements[3,4] + movements[3,5] + movements[3,6] + movements[3,7] + movements[3,8] + movements[3,9] + movements[3,10] == 4
 movements[4,1] + movements[4,2] + movements[4,3] + movements[4,4] + movements[4,5] + movements[4,6] + movements[4,7] + movements[4,8] + movements[4,9] + movements[4,10] == 8
 movements[5,1] + movements[5,2] + movements[5,3] + movements[5,4] + movements[5,5] + movements[5,6] + movements[5,7] + movements[5,8] + movements[5,9] + movements[5,10] == 12
 movements[6,1] + movements[6,2] + movements[6,3] + movements[6,4] + movements[6,5] + movements[6,6] + movements[6,7] + movements[6,8] + movements[6,9] + movements[6,10] == 2
 movements[7,1] + movements[7,2] + movements[7,3] + movements[7,4] + movements[7,5] + movements[7,6] + movements[7,7] + movements[7,8] + movements[7,9] + movements[7,10] == 14
 movements[8,1] + movements[8,2] + movements[8,3] + movements[8,4] + movements[8,5] + movements[8,6] + movements[8,7] + movements[8,8] + movements[8,9] + movements[8,10] == 11
 movements[9,1] + movements[9,2] + movements[9,3] + movements[9,4] + movements[9,5] + movements[9,6] + movements[9,7] + movements[9,8] + movements[9,9] + movements[9,10] == 15
 movements[10,1] + movements[10,2] + movements[10,3] + movements[10,4] + movements[10,5] + movements[10,6] + movements[10,7] + movements[10,8] + movements[10,9] + movements[10,10]
 == 7
 movements[i,j] >= 0 for all i in {1,2,..,9,10}, j in {1,2,..,9,10}
```

3. Building a Stadium [10 pts]

by Roumen Guha, on Sunday, February 19th, 2017

A town council wishes to construct a small stadium in order to improve the services provided to the people living in the district. After the invitation to tender, a local construction company is awarded the contract and wishes to complete the task within the shortest possible time. All the major tasks are listed in the following table. Some tasks can only start after the completion of certain other tasks, as indicated by the "Predecessors" column.

Task	Description	Duration (in weeks)	Predecessors	Maximum reduction (in weeks)	Cost of reduction ($1k/wk)
1	Installing the construction site	2	none	0	–
2	Terracing	16	1	3	30
3	Constructing the foundations	9	2	1	26
4	Access roads and other networks	8	2	2	12
5	Erecting the basement	10	3	2	17
6	Main floor	6	4,5	1	15
7	Dividing up the changing rooms	2	4	1	8
8	Electrifying the terraces	2	6	0	–
9	Constructing the roof	9	4,6	2	42
10	Lighting of the stadium	5	4	1	21
11	Installing the terraces	3	6	1	18
12	Sealing the roof	2	9	0	–
13	Finishing the changing rooms	1	7	0	–
14	Constructing the ticket office	7	2	2	22
15	Secondary access roads	4	4,14	2	12
16	Means of signalling	3	8,11,14	1	6
17	Lawn and sport accessories	9	12	3	16
18	Handing over the building	1	17	0	–

And now, the problems:

a) What is the earliest possible date of completion for the construction? Note that the last two columns of the table are not relevant for this part of the problem.

In [17]:
```
tasks = [1:18;]

durations = Dict(zip(tasks, [2, 16, 9, 8, 10, 6, 2, 2, 9, 5, 3, 2, 1, 7, 4, 3, 9, 1]))
preds     = Dict(zip(tasks, ([],[1],[2],[2],[3],[4,5],[4],[6],[4,6],[4],[6],[9],[7],[2],[4,14],[8,11,14],[12],[17])))
maxReduc  = Dict(zip(tasks, [0, 3, 1, 2, 2, 1, 1, 0, 2, 1, 1, 0, 0, 2, 2, 1, 3, 0]))
reducCost = Dict(zip(tasks, [0, 30, 26, 12, 17, 15, 8, 0, 42, 21, 18, 0, 0, 22, 12, 6, 16, 0]))
;
```

In [18]:
```
using JuMP

m = Model()

@variable(m, tstart[tasks] >= 0)

@constraint(m, link[i in tasks, j in preds[i]], tstart[i] >= tstart[j] + durations[j])

@objective(m, Min, tstart[18] + durations[18])

println(solve(m))
println(getvalue(tstart))
println("The project can complete, at earliest, in ", Int(getobjectivevalue(m)), " weeks")
```

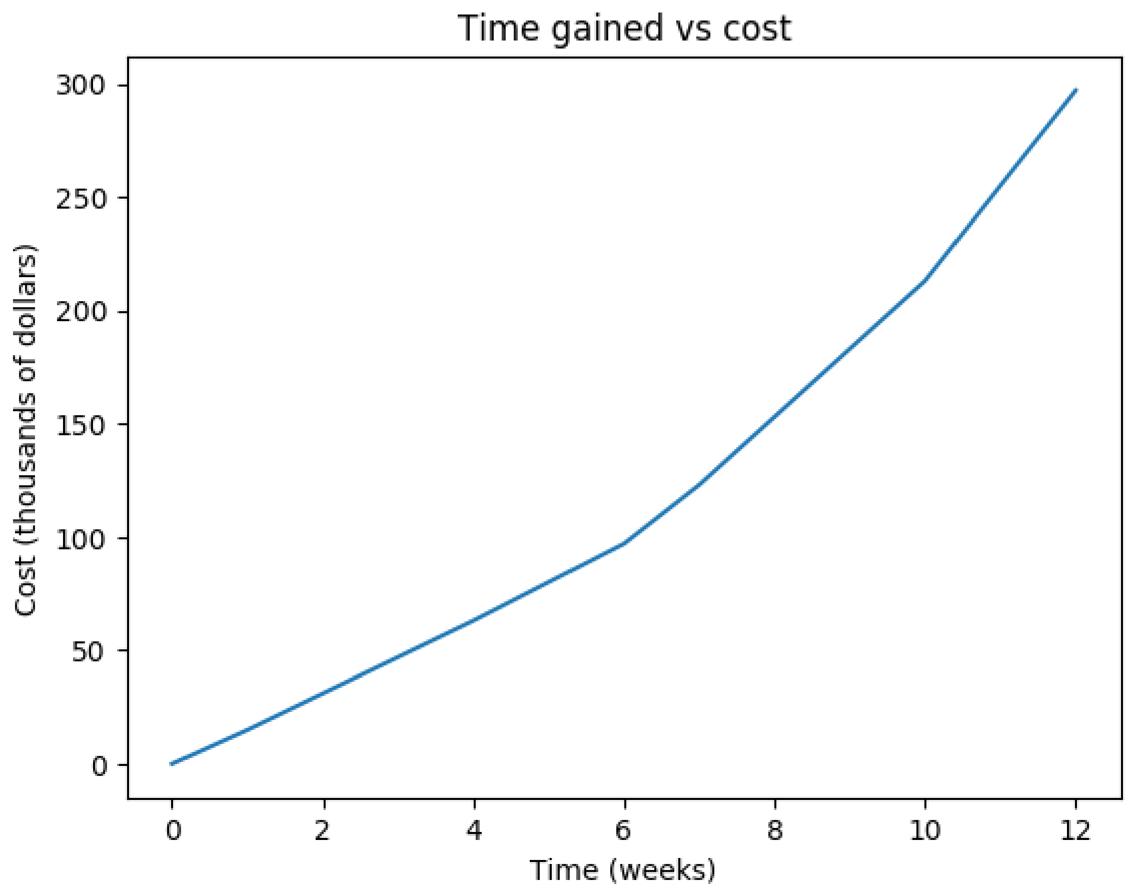
```
Optimal
tstart: 1 dimensions:
[ 1] = 0.0
[ 2] = 2.0
[ 3] = 18.0
[ 4] = 18.0
[ 5] = 27.0
[ 6] = 37.0
[ 7] = 26.0
[ 8] = 43.0
[ 9] = 43.0
[10] = 26.0
[11] = 43.0
[12] = 52.0
[13] = 28.0
[14] = 18.0
[15] = 26.0
[16] = 46.0
[17] = 54.0
[18] = 63.0

The project can complete, at earliest, in 64 weeks
```

```
In [19]: println(m)
```

```
Min tstart[18] + 1
Subject to
 tstart[2] - tstart[1] >= 2
 tstart[3] - tstart[2] >= 16
 tstart[4] - tstart[2] >= 16
 tstart[5] - tstart[3] >= 9
 tstart[6] - tstart[4] >= 8
 tstart[6] - tstart[5] >= 10
 tstart[7] - tstart[4] >= 8
 tstart[8] - tstart[6] >= 6
 tstart[9] - tstart[4] >= 8
 tstart[9] - tstart[6] >= 6
 tstart[10] - tstart[4] >= 8
 tstart[11] - tstart[6] >= 6
 tstart[12] - tstart[9] >= 9
 tstart[13] - tstart[7] >= 2
 tstart[14] - tstart[2] >= 16
 tstart[15] - tstart[4] >= 8
 tstart[15] - tstart[14] >= 7
 tstart[16] - tstart[8] >= 2
 tstart[16] - tstart[11] >= 3
 tstart[16] - tstart[14] >= 7
 tstart[17] - tstart[12] >= 2
 tstart[18] - tstart[17] >= 9
 tstart[i] >= 0 for all i in {1,2,..,17,18}
```

b) For some of the tasks, the builder may employ additional workers and rent more equipment to cut down on the total time. The last two columns of the table show the maximum number of weeks that can be saved per task and the associated additional cost per week incurred by the extra work. Plot a trade-off curve that shows extra cost as a function of the number of weeks early we wish the stadium to be completed.

In [20]:
```
using JuMP, PyPlot

extraCost = [0, 15, 31, 47, 63, 80, 97, 123, 153, 183, 213, 255, 297]
reducedTime = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
using PyPlot
plot(reducedTime, extraCost)
title("Time gained vs cost")
xlabel("Time (weeks)")
ylabel("Cost (thousands of dollars)")
```



Out[20]:
```
PyObject <matplotlib.text.Text object at 0x000000002DB3B0B8>
```

c) The town council wants the builder to expedite the project. As an incentive, the council will pay a bonus of $30k/week for each week the work finishes early. When will the project be completed if the builder is acting in a way that maximizes his profit?

In [21]:
```julia
using JuMP

m2 = Model()

@variable(m2, tstart[tasks] >= 0)
@variable(m2, reductions[tasks] >= 0)

for i in tasks
    # set each task to be limited by its max reductions
    @constraint(m2, reductions[i] <= maxReduc[i])
end

@constraint(m2, link[i in tasks, j in preds[i]], tstart[i] >= tstart[j] + durations[j] - reductions[j])

# profit is given by 30k * (difference between original predicted end date and actual end date) - costs incurred by builder from hiring additional employees and workers
@expression(m2, costs[i in tasks], sum(reductions[i]*reducCost[i]))
@objective(m2, Max, 30(64 - (tstart[18] + durations[18])) - sum(costs))

println(solve(m2))
println(getvalue(tstart))
println("If the builder acts to maximize profit, the project can complete, at earliest, in ", Int(getvalue(tstart[18]) + durations[18]), " weeks")
println("In the process, the builder will make ", Int(getobjectivevalue(m2)), "k more than he was previously making")
```

```
Optimal
tstart: 1 dimensions:
[ 1] = 0.0
[ 2] = 2.0
[ 3] = 18.0
[ 4] = 18.0
[ 5] = 26.0
[ 6] = 34.0
[ 7] = 26.0
[ 8] = 39.0
[ 9] = 39.0
[10] = 26.0
[11] = 39.0
[12] = 48.0
[13] = 28.0
[14] = 18.0
[15] = 26.0
[16] = 42.0
[17] = 50.0
[18] = 56.0

If the builder acts to maximize profit, the project can complete, at earliest, in 57 weeks
In the process, the builder will make 87k more than he was previously making
```

In [22]:
```
println(m2)
```

```
Max -30 tstart[18] - 30 reductions[2] - 26 reductions[3] - 12 reductions[4] - 17 reductions[5] - 15 reductions[6] - 8 reductions[7] - 42 reductions[9] - 21 reductions[10] - 18 redu
ctions[11] - 22 reductions[14] - 12 reductions[15] - 6 reductions[16] - 16 reductions[17] + 1890
Subject to
 reductions[1] <= 0
 reductions[2] <= 3
 reductions[3] <= 1
 reductions[4] <= 2
 reductions[5] <= 2
 reductions[6] <= 1
 reductions[7] <= 1
 reductions[8] <= 0
 reductions[9] <= 2
 reductions[10] <= 1
 reductions[11] <= 1
 reductions[12] <= 0
 reductions[13] <= 0
 reductions[14] <= 2
 reductions[15] <= 2
 reductions[16] <= 1
 reductions[17] <= 3
 reductions[18] <= 0
 tstart[2] - tstart[1] + reductions[1] >= 2
 tstart[3] - tstart[2] + reductions[2] >= 16
 tstart[4] - tstart[2] + reductions[2] >= 16
 tstart[5] - tstart[3] + reductions[3] >= 9
 tstart[6] - tstart[4] + reductions[4] >= 8
 tstart[6] - tstart[5] + reductions[5] >= 10
 tstart[7] - tstart[4] + reductions[4] >= 8
 tstart[8] - tstart[6] + reductions[6] >= 6
 tstart[9] - tstart[4] + reductions[4] >= 8
 tstart[9] - tstart[6] + reductions[6] >= 6
 tstart[10] - tstart[4] + reductions[4] >= 8
 tstart[11] - tstart[6] + reductions[6] >= 6
 tstart[12] - tstart[9] + reductions[9] >= 9
 tstart[13] - tstart[7] + reductions[7] >= 2
 tstart[14] - tstart[2] + reductions[2] >= 16
 tstart[15] - tstart[4] + reductions[4] >= 8
 tstart[15] - tstart[14] + reductions[14] >= 7
 tstart[16] - tstart[8] + reductions[8] >= 2
 tstart[16] - tstart[11] + reductions[11] >= 3
 tstart[16] - tstart[14] + reductions[14] >= 7
 tstart[17] - tstart[12] + reductions[12] >= 2
 tstart[18] - tstart[17] + reductions[17] >= 9
 tstart[i] >= 0 for all i in {1,2,..,17,18}
 reductions[i] >= 0 for all i in {1,2,..,17,18}
```

Comment: Thinking about this a little more, we can see that there is a way to actually reduce the time taken to construct this stadium by another 3 weeks.

Task 2, which would originally have taken 16 weeks (as it does even in this solution) can be reduced by 3 weeks. However, it looks like the solution did not include it because the profit from those extra 3 weeks (30k per week) would have completely negated the profit earned from those 3 weeks (30k for every week the building finishes early).

4. Dual Interpretation [10 pts]

by Roumen Guha, on Sunday, February 19th, 2017

Suppose t ∈ [0; 2π] is a parameter. Consider the following LP:

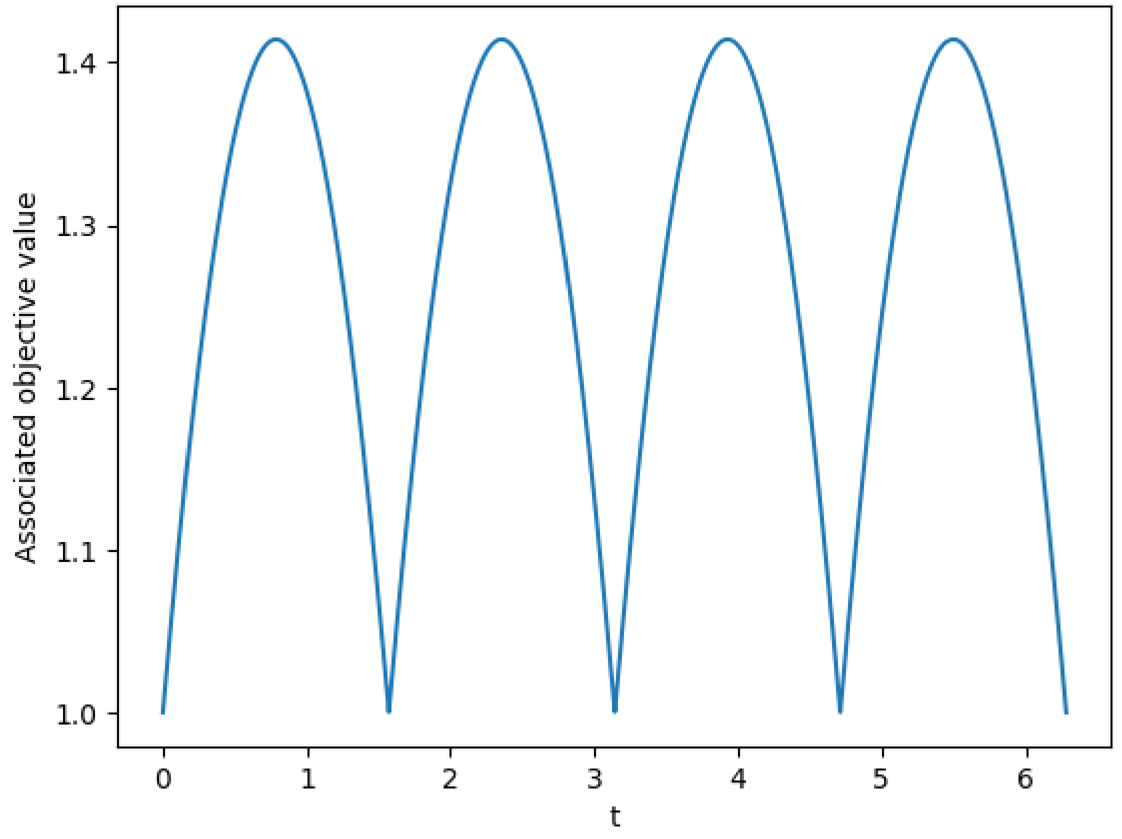
$$\begin{array}{rl} \text{minimize} & p+q+r+s \\ \text{subject to:} & p-r=cos(t) \\ & q-s=sin(t) \\ & p,q,r,s \geq 0 \end{array}$$

a) Plot the optimal objective of this LP as a function of t. Can you explain what you see? Hint: separately consider the cases where cos(t) and sin(t) are positive or negative (four cases).

In [5]:
```
using JuMP, Gurobi, Mosek

function solveLP(t)

    m = Model()

    @variable(m, p >= 0)
    @variable(m, q >= 0)
    @variable(m, r >= 0)
    @variable(m, s >= 0)

    @constraint(m, p - r == cos(t))
    @constraint(m, q - s == sin(t))

    @objective(m, Min, p + q + r + s)

    solve(m)

    return(getobjectivevalue(m))

end
```

Out[5]:
```
solveLP (generic function with 1 method)
```

In [35]:
```
using PyPlot

t_array = linspace(0, 2*pi, 10000)
obj_array = zeros(10000)

for i in 1:10000
    obj_array[i] = solveLP(t[i])
end

plot(t_array, obj_array)
xlabel("t")
ylabel("Associated objective value")
```



Out[35]:
```
PyObject <matplotlib.text.Text object at 0x0000000031196860>
```

There's not much to see, just that it is periodic with frequency pi/2 and never descends past 1, and also never increases past sqrt(2).

When both cos(t) and sin(t) are positive, that means that p >= r and q >= s.

When both cos(t) and sin(t) are negative, this means that p <= r and q <= s.

When cos(t) is positive and sin(t) is negative, we know that p >= r and q <= s.

When cos(t) is negative and sin(t) is positive, we know that p <= r and q >= s.

b) Find the dual LP and interpret it geometrically. Does this agree with the solution of part a)?

In [46]:
```
using JuMP, Gurobi, Mosek

m2 = Model()

@variable(m2, λ[1:2])

@constraint(m2, λ[1] <= -1)
@constraint(m2, λ[2] <= 1)

@objective(m2, Max, cos(t)*λ[1]+ sin(t)*λ[2]);
```

Primal

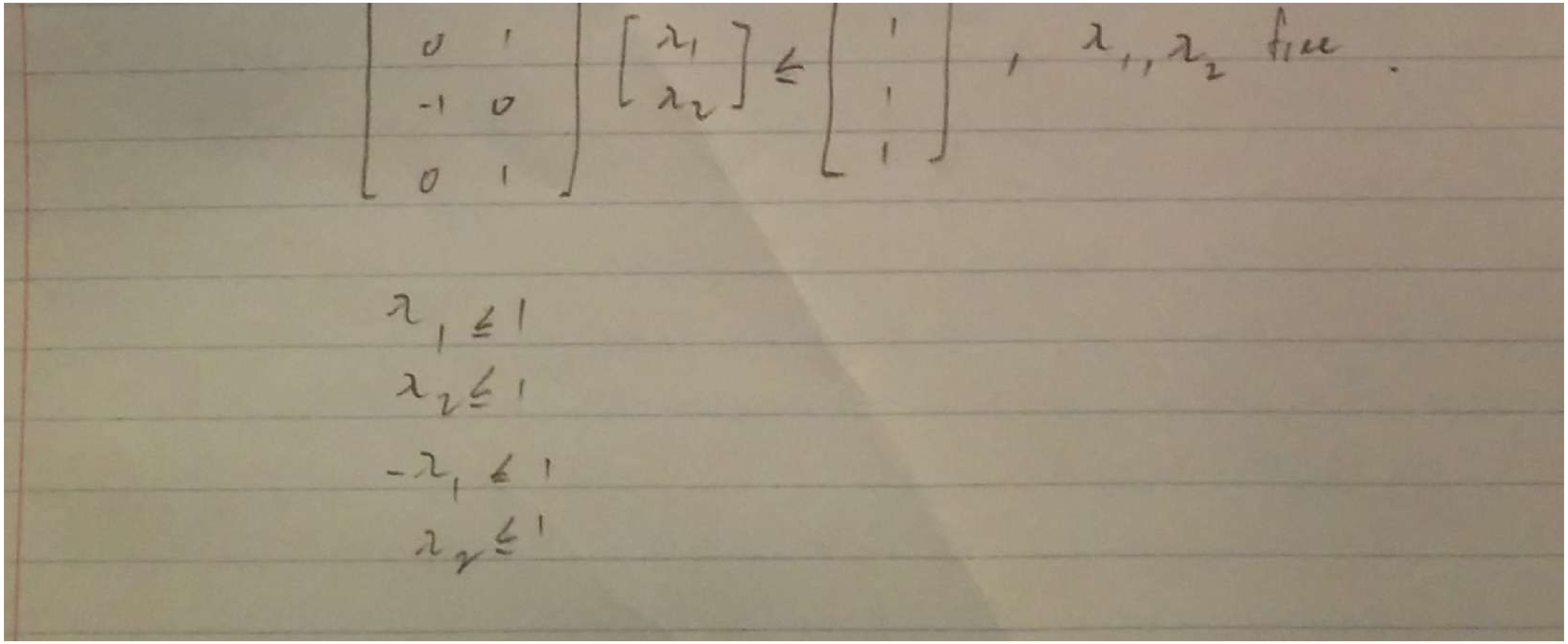
$$\min \quad \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \\ s \end{bmatrix}$$

$$\text{s.t} \quad \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \\ s \end{bmatrix} = \begin{bmatrix} \cos(t) \\ \sin(t) \end{bmatrix}, \quad p, q, r, s \geq 0$$

DUAL

$$\max \quad \begin{bmatrix} \cos(t) & \sin(t) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$$

s.t. $\begin{bmatrix} 1 & 0 \end{bmatrix}$ $\begin{bmatrix} 1 \end{bmatrix}$



$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \lambda_1, \lambda_2 \text{ free}.$$

$$\lambda_1 \leq 1$$
$$\lambda_2 \leq 1$$
$$-\lambda_1 \leq 1$$
$$\lambda_2 \leq 1$$

Note: I forgot to write this on the page, but since λ1 has two constraints, the lower of the two upper-bounds is the correct one. There is a mistake on the page, the last element of the A vector should be -1. Therefore the constraints are:

λ1 <= -1

λ2 <= -1

As for the geometric interpretation, we can see from this picture that the constraints that were present on p, q, r and s in the primal form of the problem have shifted into constraints on the lambdas which are naturally free variables. In fact, these constraints plot out the same vertices that the original problem did, meaning that we achieve the same planes of solutions. Yes, this dual formation agrees with the original primal formation.