HW 8: Integer programs and duality

1. ABC investments

The variables are

1. $S[i]$, (S is the short for selection), which only has 0 and 1 inside, indicating if a investment is made for the i_{th} option;
2. $I[i]$, (I is the short for investment), indicating how much investment is made for the i_{th} option;

The constraints are:

1. $\sum_i I_i = 80$;
2. S only contains 0 and 1;
3. $I_5 <= I_2 + I_4 + I_6$;
4. if $S_3 = 1$, then $S_6 = 1$, or, if $S_6 = 0$, then $S_3 = 0$;
5. $S_i Min_i \leq I_i \leq S_i Max_i$ where Min_i and Max_i is the minimum and maximum investment for the i_{th} option;

The obejective is to maximum $\sum_i I_i P_i$, where P_i is the return of the i_{th} option.

In [1]:

```
using JuMP, Gurobi
m1_test = Model(Gurobi.Optimizer);
set_silent(m1_test);
option = [3 27 13; 2 12 9; 9 35 17; 5 15 10; 12 46 22; 4 18 12];

select = @variable(m1_test, [1:6, 1:1], Bin);
invest = @variable(m1_test, [1:6,1:1]);

@constraint(m1_test, sum(invest) == 80);
@constraint(m1_test, invest[5]<=(invest[2] + invest[4] + invest[6]));

for i=1:6
   if i==3
      @constraint(m1_test, invest[i] <= select[i]*select[6]*option[i,2]); ## if opt6 is 0, then
      @constraint(m1_test, invest[i] >= select[i]*select[6]*option[i,1]); ## if opt6 is 1, then
   else
      @constraint(m1_test, invest[i] <= select[i]*option[i,2]);
      @constraint(m1_test, invest[i] >= select[i]*option[i,1]);
   end
end

@objective(m1_test, Max, sum(select .*invest .*option[:, 3] ./100));
optimize!(m1_test);

```

```
Academic license - for non-commercial use only
Academic license - for non-commercial use only
```

In [2]:

```
println("The maximum return is \$$(round.(objective_value(m1_test), digits =6)) million ");
println("The investment is $(round.(value.(invest), digits =6)) (million dollars) for these six
```

```
The maximum return is $13.5 million
The investment is [0.0; 0.0; 35.0; 5.0; 22.5; 17.5] (million dollars) for these six op
tions.
```

2. Lagrangian duality

a)

The variables are x_1 and x_2;
The constraint is $x_1 \geq 1$;
The objective is to minimize $\frac{1}{2}(x_1^2 + x_2^2)$;

In [3]:

```
using Gurobi, JuMP;
```

In [4]:

```
m2a = Model(Gurobi.Optimizer);
set_silent(m2a);
x = @variable(m2a, [1:2, 1:1]);
@constraint(m2a, x[1] >= 1 );
@objective(m2a, Min, (x[1]^2 + x[2]^2)/2);
optimize!(m2a);
primal_optimal = objective_value(m2a);
```

```
Academic license - for non-commercial use only
Academic license - for non-commercial use only
```

In [5]:

```
println("The optimal primal value p* is $(primal_optimal)");
```

```
The optimal primal value p* is 0.5
```

b)

The Lagrangian $L(x_1, x_2, \lambda) = \frac{1}{2}(x_1^2 + x_2^2) + \lambda(1 - x_1)$
$L(x_1, x_2, \lambda) = \frac{1}{2}(x_1^2 + x_2^2 - 2\lambda x_1) + \lambda = \frac{1}{2}(x_1^2 + x_2^2 - 2\lambda x_1 + \lambda^2) - \frac{1}{2}\lambda^2 + \lambda = \frac{1}{2}((x_1 - \lambda)^2 + x_2^2) - \frac{1}{2}\lambda$
The dual $g(\lambda) = min\frac{1}{2}((x_1 - \lambda)^2 + x_2^2) - \frac{1}{2}\lambda^2 + \lambda$
As $min\frac{1}{2}((x_1 - \lambda)^2 + x_2^2) = 0$, we have
$g(\lambda) = -\frac{1}{2}\lambda^2 + \lambda \quad (s.t \quad \lambda \geq 0)$

c)

c)

In [6]:

```
m2b = Model(Gurobi.Optimizer);
set_silent(m2b);
@variable(m2b, lambda >= 0);
@objective(m2b, Max, -0.5 * lambda^2 + lambda );
optimize!(m2b);
dual_optimal = objective_value(m2b);
```

```
Academic license - for non-commercial use only
Academic license - for non-commercial use only
```

In [7]:

```
println("The optimal dual value d* is $(dual_optimal)");
```

```
The optimal dual value d* is 0.5
```

d)

Slater condition is staisfied in this question. Let's look at the Lagrange function: $L(x_1, x_2, \lambda) = \frac{1}{2}(x_1^2 + x_2^2) + \lambda(1 - x_1)$ has two terms where the both of them are convex. Thus, this problem is convex; also, there exists x satisfying $1 - x < 0$, so this question is trictly feasible;

Thus the strong duality holds.

3. Laurent goes to the gym

In [8]:

```
using JuMP, Gurobi
Require = [310;100;355;130;395;160;375;160;355;160;330;160;310;160;290;160];

```

a)

The variable is n_{iw} , the PAIR number of plates with weight w in the i_{th} exercise.
The constraints are

1. n_{iw} is integer
2. $\sum_w 2 * n_{iw} w + 45 = r_i$, where r_i is the required weight for the i_{th} exercise, and 45 is the weight of the steel bar

The objective is to minimize $\sum_w n_{iw}$ for the i_{th} exercise

In [9]:

```
W = [2.5;5;10;25;45];

m3a = Model(Gurobi.Optimizer);
set_silent(m3a);

schedule_1 = zeros(length(W), length(Require));

for i = 1:length(Require)
  n_pairs = @variable(m3a, [1:length(W), 1:1], Int);
  @constraint(m3a, n_pairs .>= 0);
  @constraint(m3a, (2 * sum(n_pairs .* W) )+ 45== Require[i]);
  @objective(m3a, Min, sum(n_pairs));
  optimize!(m3a);
  schedule_1[:,i] = (value.(n_pairs)).*2;
end
```

```
Academic license - for non-commercial use only
Academic license - for non-commercial use only
```

In [10]:

```
using NamedArrays;

Exercises = collect(1:length(Require));
Weights = ["2.5", "5", "10", "25", "45"];
#NamedArray((schedule_1'), (Weights, Exercises), ("Plates", "Exercise"))
println(NamedArray(convert.(Int, schedule_1'), (Exercises, Weights), ("Exe", "Plates")));
println("The exercises are sorted by set. Every pair (e.g 3 and 4) are in the same set.");
```

```
16×5 Named Array{Int64,2}
Exe ╲ Plates │ 2.5    5   10   25   45
─────────────┼─────────────────────────
1            │   2    0    2    6    2
2            │   2    0    0    2    0
3            │   0    0    4    0    6
4            │   2    2    2    2    0
5            │   0    2    2    2    6
6            │   2    0    2    0    2
7            │   0    2    0    2    6
8            │   2    0    2    0    2
9            │   0    0    4    0    6
10           │   2    0    2    0    2
11           │   2    2    0    0    6
12           │   2    0    2    0    2
13           │   2    2    2    2    4
14           │   2    0    2    0    2
15           │   2    2    0    2    4
16           │   2    0    2    0    2
The exercises are sorted by set. Every pair (e.g 3 and 4) are in the same set.
```

b)

The variable is n_{iw}, the PAIR number of plates with weight w in the i_{th} exercise.
The constraints are

1. n_{iw} is integer

2. $\sum_w 2 * n_{iw} w + 45 = r_i$, where r_i is the required weight for the i_{th} exercise, and 45 is the weight of the steel bar
3. $n_{iw} \leq n_{w_{purchase}}$, where $n_{w_{purchase}}$ is the number of plates with weight w purchased.

The objective is to minimize $\sum_w n_{w_{purchase}}$.

In [11]:

```
W = [2.5;5;10;25;45];

m3b = Model(Gurobi.Optimizer);
set_silent(m3b);

#schedule_1 = zeros(length(W), length(Require));
n_purchase_pairs = @variable(m3b, [1:length(W), 1:1], Int);
n_pairs = @variable(m3b, [1:length(W), 1:length(Require)], Int);
for i = 1:length(Require)
   #@constraint(m3b, n_pairs[:,i])
   @constraint(m3b, n_pairs[:,i] .>= 0);
   @constraint(m3b, (2 * sum(n_pairs[:,i] .* W) )+ 45== Require[i]);
   @constraint(m3b, n_pairs[:,i] .<= n_purchase_pairs);
end

@objective(m3b, Min, sum(n_purchase_pairs));
optimize!(m3b);
convert.(Int,2*(value.(n_pairs)'))
```

```
Academic license - for non-commercial use only
Academic license - for non-commercial use only
```

Out[11]:

```
16×5 Array{Int64,2}:
 2  2  2  2  4
 2  0  0  2  0
 0  2  2  4  4
 2  2  2  2  0
 0  2  2  2  6
 2  0  2  0  2
 0  2  0  2  6
 2  0  2  0  2
 0  2  2  4  4
 2  0  2  0  2
 2  0  0  4  4
 2  0  2  0  2
 2  2  2  2  4
 2  0  2  0  2
 2  2  0  2  4
 2  0  2  0  2
```

In [12]:

```
Purchased = convert.(Int,value.(n_purchase_pairs) .*2);
println("The number of each bumber is :")
println(Weights);
println(Purchased);
```

```
The number of each bumber is :
["2.5", "5", "10", "25", "45"]
[2; 2; 2; 4; 6]
```

c)

The variable is n_{iw} , the PAIR number of plates with weight w in the i_{th} exercise.
The constraints are

1. n_{iw} is integer
2. $\sum_w 2 * n_{iw} w + 45 = r_i$, where r_i is the required weight for the i_{th} exercise, and 45 is the weight of the steel bar
3. $n_{iw} \leq n_{w_{purchase}}$, where $n_{w_{purchase}}$ is the number of plates with weight w purchased.

The objective is to minimize $\sum_w n_{iw}$ for the i_{th} exercise.

In [13]:

```
Purchased = value.(n_purchase_pairs);

W = [2.5;5;10;25;45];

m3c = Model(Gurobi.Optimizer);
set_silent(m3c);

schedule_2 = zeros(length(W), length(Require));

for i = 1:length(Require)
  n_pairs = @variable(m3c, [1:length(W), 1:1], Int);
  @constraint(m3c, n_pairs .>= 0);
  @constraint(m3c, n_pairs .<= Purchased);
  @constraint(m3c, (2 * sum(n_pairs .* W) )+ 45== Require[i]);
  @objective(m3c, Min, sum(n_pairs));
  optimize!(m3c);
  schedule_2[:,i] = (value.(n_pairs)).*2;
end
```

```
Academic license - for non-commercial use only
Academic license - for non-commercial use only
```

In [14]:

```
Exercises = collect(1:length(Require));
Weights = ["2.5", "5", "10", "25", "45"];
#NamedArray((schedule_1'), (Weights, Exercises), ("Plates", "Exercise"))
println(NamedArray(convert.(Int, schedule_2'), (Exercises, Weights), ("Exe", "Plates")));
println("The exercises are sorted by set. Every pair (e.g 3 and 4) are in the same set.");
```

```
16×5 Named Array{Int64,2}
Exe ╲ Plates │ 2.5   5  10  25  45
─────────────┼─────────────────────────────────────
1         │   2   2   2   2   4
2         │   2   0   0   2   0
3         │   0   2   2   4   4
4         │   2   2   2   2   0
5         │   0   2   2   2   6
6         │   2   0   2   0   2
7         │   0   2   0   2   6
8         │   2   0   2   0   2
9         │   0   2   2   4   4
10        │   2   0   2   0   2
11        │   2   0   0   4   4
12        │   2   0   2   0   2
13        │   2   2   2   2   4
14        │   2   0   2   0   2
15        │   2   2   0   2   4
16        │   2   0   2   0   2
The exercises are sorted by set. Every pair (e.g 3 and 4) are in the same set.
```

In [ ]:

```

```