Université Jean Monnet

Optimization and Operational Research

Practical Session Report

Constrained Optimization

Authors:
Allwyn Joeseph
Omar Elsabrout

Supervisor:
Dr. Amaury Habrard

March 21, 2018

Abstract

This document represents a report on the outcomings of a practical session in the optimization and operational research course. The goal of this practical session is to formulate some realistic problems as optimization problems and use the AMPL software to solve them.

1 Introduction

During studying mathematical optimization and constrained problems, it is natural to seek software to be able to solve real-world optimization problems in a practical and specialized technique. Thus, we use AMPL in our solution through out this report to solve the provided problems in the practical session PDF document.

In order to run our implementation, only a free-demo of the AMPL's IDE software solution is enough accompanied with our files available on the GitHub repository in the following link:
`https://github.com/Sabrout/Constraint-Optimization-AMPL`

To run our code, follow the following steps by executing them in the AMPL console:

1. Load the model file corresponding to the problem number by typing `model <filename>;`
2. Load the data file corresponding to the problem number by typing `data <filename>;` (if applicable)
3. Solve the current problem by typing `solve;`
4. View the solution by typing `display <Variable Name>`

2 Unconstrained optimization

2.1 Problem 1

The following problem entailed solving for the equation below:

$$\min_{x_1,x_2} \quad 4x_1^2 + 7(x_2 - 4)^2 - 3x_1 + 4x_2$$

The above equation can be modeled as the following in the AMPL solver:

```
reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;
var x1 >= 0;
var x2 >= 0;

minimize f: 4*(x1)^2 + 7*(x2-4)^2 - 3*x1 + 4*x2;

solve;
display x1;
display x2;
```

The resulting solution by the LOQO solver in AMPL is that `x1 = 0.375` and `x2 = 3.71429` . This was verified by differentiating the equation with respect to x1 and x2 to arrive at the solution output by the solver.

2.2 Problem 2

The following problem entailed solving for the equation below:

$$\min_{x_1,x_2} \quad (1-x_1)^2 + 100(x_2 - x_1^2)^2$$

The above equation can be modeled as the following in the AMPL solver:

```
reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;
var x1 >= 0;
var x2 >= 0;
let x1 := 1;   # tested for different values of x1
let x2 := -1;  # tested for different values of x2

minimize f: (1-x1)^2 + 100*(x2-(x1)^2)^2;

solve;
display x1;
display x2;
```

The problem equation was run four times with different starting values for `x1` and `x2` and the results were tabulated as seen below:

Parameters	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Initialized X1	0	1	5	10
Initialized X2	0	1	5	10
Output X1	1	1	2.8	10.1
Output X2	1	1	8.2	101.2
Dual Objective	-2.51e-09	0	3.48	84.72
Primal Objective	3.27e-15	-3.13e-08	4.75	89.2

It's clear from the iterations that the global minimum isn't always obtained, signaling that the optimisation process got stuck in some local minima. However, the optimum value for the equation can be found at `x1 = 1` and `x2 = 1`.

2.3 Problem 3

The following problem entailed solving for the equation below:

$$\min_{x_1,x_2} \quad 10 + x_1^2 + x_2^2 - 10(cos2\pi x_1 + cos2\pi x_2)$$

The above equation can be modeled as the following in the AMPL solver:

```
reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;
param pi := 4*atan(1);
var x1;
var x2;
let x1 := 0; # tested for different values of x1
let x2 := 0; # tested for different values of x2

minimize f: 10 + x1^2 + x2^2 - 10*(cos(2*pi*x1) + cos(2*pi*x2));

solve;
display x1;
display x2;
```

Parameters	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Initialized X1	-10	0	10	100
Initialized X2	-10	0	10	100
Output X1	-9.94	0	9.94	87.36
Output X2	-9.94	0	9.94	87.36
Dual Objective	188.98	-10	188.98	15286.25
Primal Objective	188.98	-10	188.98	15286.25

Different starting points does gives different minimas, signaling that the optimization parameters get stuck at some local minima. From the table above it is observed that as the initializations of `x1` and `x2` move away from zero the global minima is no longer achieved.

3 Modeling constrained problems

We now focus on modeling some kind of "real" problems.

3.1 Water resources

In order to solve the water resources problem, we need to introduce two variables that will be provide the solution later which is the amount of consumption of water sources. Those variables are "r" for the reservoir source of water and "s" for the stream source. The model is implemented as the following:

```
reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;

var r >= 0;
var s >= 0;

minimize f: 100*r + 50*s;
subject to c1: s <= 100;
subject to c2: 50*r + 250*s <= 100*(r+s);
```

```
subject to c3: r+s >= 500;
```

As indicated in the code, there are three constraints presented. The first constraint states the maximum limit of water used from the stream. Following a second constraint that limits the pollution to the standards accepted by the city so it will not exceed 100 ppm. The third constraint assures the basic needs of the city of 500,000 liters of water per day.

The resulting solution by the LOQO solver in AMPL is that `r = 400` and `s = 100` which means that we should get 400,000 liters from the reservoir and 100,000 liters from the stream.

3.2 Good-smelling perfume design

To solve the prefume design problem, we introduce four variables for the proportions of the four blends to be "b1", "b2", "b3" and "b4" respectively. The model is implemented as the following:

```
reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;

var b1 >= 0;
var b2 >= 0;
var b3 >= 0;
var b4 >= 0;

minimize f: 55*b1 + 65*b2 + 35*b3 + 85*b4;

subject to c0: b1+b2+b3+b4 = 1;
subject to c1: 0.05 <= b2 <= 0.2;
subject to c2: b3 >= 0.3;
subject to c3: 0.1 <= b1 <= 0.25;
subject to c4: 0.35*b1 + 0.6*b2 + 0.35*b3 + 0.4*b4 <= 0.5;
subject to c5: 0.08 <= 0.15*b1 + 0.05*b2 + 0.2*b3 + 0.1*b4 <= 0.13;
subject to c6: 0.3*b1 + 0.2*b2 + 0.4*b3 + 0.2*b4 <= 0.35;
subject to c7: 0.2*b1 + 0.15*b2 + 0.05*b3 + 0.3*b4 >= 0.19;
```

There are eight constraints used for the model. The first one basic as it indicates that all the proportions of the blends are percentages so they must

add to 100. Next, the following three constraints limit those proportions of the blends. Then, the four constraints after that are limiting the usages of essences of real flowers in the blends that are forming our new wanted blend.

The solution using the LOQO solver in AMPL is that `b1 = 0.14`, `b2 = 0.14`, `b3 = 0.3` and `b4 = 0.42`. This means to prepare the product with 14%, 14%, 30% and 42% in order for the four blends.

3.3 Roadway expenses

3.3.1 Rural/Urban case

In order to solve the Roadway expenses problem, we need to introduce two variables that will be provide the solution later which is the amount of cost for rural areas and the cost for urban areas. Those variables are "xr" for the reservoir source of water and "xu" representing those two costs. The model is implemented as the following:

```
reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;

var xu >= 0;
var xr >= 0;

maximize f: (7000*log(1 + xr)) + (5000*log(1 + xu)) - xr - xu;
subject to c1: xu + xr <= 200;
```

As indicated in the code, there is only one constraint to limit the expenses to 200 million euros. The benefit function is maximized to meet this upper bound.

The resulting solution by the LOQO solver in AMPL is that `xu = 83.1667` and `xr = 116.833` which means that we should spend 83 million euros on urban areas and 116.8 millions on rural areas.

3.3.2 General case

This part of the session is dependent on the previous part as we are trying to solve the same problem in a more dynamic manner with expanding the set of categories to be more than boolean if rural or urban. In this case, the

number of variables to provide the solution is dependent on the number of categories put in the data file and it represented by a set named "x" that will show the solution later. The model file is written as the following:

```
reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;

set items;
var x{i in items};
param C{i in items};
param budget;

maximize f: sum{i in items} (C[i]*log(1 + x[i]) - x[i]);
subject to c: sum{i in items} (x[i]) <= budget;
```

For such a case, we prepared to data files to test the model. The first one is `prob_3_3_2(old).dat` to solve the same problem again as in the Rural/Urban case and check whether we will have the same results. The data file is written like the following:

```
set items := rural urban;
param C := rural 7000 urban 5000;
param budget := 200;
```

After executing the model and the data file, it provides the same answer as before which indicates that the dynamic approach we selected is working successfully and provides the same answers as before. Another part of the question requires adding a third category in the set to test how dynamic is our model. Thus, we created another data file named `prob_3_3_2.dat` that is written to add the suburban category as the following:

```
set items := rural urban suburban;
param C := rural 7000 urban 5000 suburban 7000;
param budget := 200;
```

The result of such data is `rural 73.7895`, `suburban 73.7895` and `urban 52.4211`. Hence, France should spend 73.7 million euros on roadway improvements on rural areas, equally on suburban areas and 52.4 millions on urban areas.

3.4 Design you own optimization problem

Since this part is dedicated to designing our own optimization problem, we chose it to be expressive of a relative issue to our lives as students in the current time being which is maximizing our efficiency in studying, working and succeeding in our courses. Hence, we decided to propose a problem with multiple courses, grading schemes and working hours that are completely dynamic so they can fit every student's needs. Then, solving the optimization problem provides percentages of the student's total time so he or she can prioritize their course list and make the process as efficient as possible.

The structure of such problem is that we have four courses for the semester which are Computer Vision, Machine Learning, Data Mining and Optimization. Each course has a grading scheme which is divided into four duties such as a project, a report, a midterm exam and a final exam. The percentages of each duty with respect to the total grade differs from one course to another. Finally, we have the number of working hours that the student can afford for every course. The following table shows our problem.

Course	Computer Vision	Machine Learning	Data Mining	Optimization
Final Exam	30	20	40	25
Midterm	40	65	35	40
Project	20	15	5	30
Report	10	0	20	5
Hours	70	40	60	80

There are some constraints on the grades which will assume that the student is not aiming for the full grade for every course because ot the time limitations and the student's history of grades, scientific interest and curiosity, understanding of the teaching material and motivation and energy to work. For the machine learning course, the student is not willing to dedicate more than 20% of his time. However, he must dedicate more than 30% of his time to Data Mining in order to pass. Also, the student has some experience in Computer Vision so he only needs to dedicate from 10% to 25% of his time to it.

Moreover, practical parts are important for the student so the project parts of all courses can not be less than 18% in total. As the reports are essential for a good academic profile, reports must occupy between 8% and 13% of the total time. Finally, midterm exams secure a great part of the total grade but

it can take over other duties, so studying for midterms must occupy 50% at most of his or her time.
To solve the problem, we must find the least time consuming way of studying and working to make the student pass the semester with respect to his other concerns as provided before.

3.4.1 Problem Formalization

The decision variables are:

$$x_j = \text{Percent of time spent on a course in the semester, } j = 1 \text{ to } 4$$

The model is:

$$\text{Minimize time} = 70x_1 + 40x_2 + 60x_3 + 80x_4$$

subject to

$$x_1 + x_2 + x_3 + x_4 = 100$$

$$x_2 \leq 20,\ x_3 \geq 30$$

$$10 \leq x_1 \leq 25$$

$$0.2x_1 + 0.15x_2 + 0.05x_3 + 0.3x_4 \geq 18$$

$$8 \leq 0.1x_1 + 0.2x_3 + 0.05x_4 \leq 13$$

$$0.4x_1 + 0.65x_2 + 0.35x_3 + 0.4x_4 \leq 50$$

$$x_j \geq 0 \text{ for all } j = 1 \text{ to } 4$$

3.4.2 Model and Data

According to the formalized problem, we implemented a model in the file `prob_4_3.mod` that fits the mathematical representation of the problem as the following:

```
reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;


set FullGrade; # set of FullGrades
```

```
set MyGrade; # set of MyGradess
param n_percent{FullGrade,MyGrade}>=0;
# percentage of MyGrade to FullGrade
param work{FullGrade}>=0;
# hours for full grades
param u_FullGrade{FullGrade}>=0;
# upper bound for the percentage of each FullGrade in semester
param l_FullGrade{FullGrade}>=0;
# lower bound for the percentage of each FullGrade in semester
param u_MyGrade{MyGrade} >=0;
# upper bound for the percentage of each MyGrade in semester
param l_MyGrade{MyGrade} >=0;
# lower bound for the percentage of each MyGrade in semester
var WorkDivision{FullGrade}>=0;
# percentage of each CourseWork in semester
minimize Total_work:
sum{i in FullGrade} work[i]*WorkDivision[i];
subject to Work: sum{i in FullGrade} WorkDivision[i] = 100;
# total percentage should be 100%
subject to FullGrade_u{i in FullGrade}:
WorkDivision[i] <= u_FullGrade[i];
# FullGrade percentage <= upper bound
subject to FullGrade_l{i in FullGrade}:
WorkDivision[i] >= l_FullGrade[i];
# FullGrade percentage >= lower bound
subject to MyGrade_u{j in MyGrade}:
sum {i in FullGrade} WorkDivision[i]
* n_percent[i,j] / 100 <= u_MyGrade[j];
# MyGrades percentage <= upper bound
subject to MyGrade_l{j in MyGrade}:
sum {i in FullGrade} WorkDivision[i]
* n_percent[i,j] /100 >= l_MyGrade[j];
# MyGrades percentage >= lower bound
```

The model has comments explaining every step along the way. In addition, we implemented also the data file in order for the model to function in the file `prob_4_3.dat`. It is as follows:

```
set FullGrade := ComputerVision MachineLearning DataMining Optimization ;
```

```
set MyGrade := FinalExam MidtermExam Project Report;
param n_percent :
FinalExam MidtermExam Project Report :=
ComputerVision 30 40 20 10
MachineLearning 20 65 15 0
DataMining 40 35 5 20
Optimization 25 40 30 5 ;

param work :=
ComputerVision 70 MachineLearning 40 DataMining 60 Optimization 80 ;
param u_FullGrade :=
ComputerVision 25 MachineLearning 20 DataMining 100 Optimization 100 ;
param l_FullGrade :=
ComputerVision 10 MachineLearning 0 DataMining 30 Optimization 0 ;
param u_MyGrade :=
FinalExam 100 MidtermExam 50
Project 100 Report 13 ;
param l_MyGrade :=
FinalExam 0 MidtermExam 0
Project 18 Report 8 ;
```

After executing the file and solving the problem, we displayed the key variables in the console to find that :

```
WorkDivision [*] :=
ComputerVision  15
DataMining  30
MachineLearning  20
Optimization  35
;
```

Hence, the student should dedicate 15% of his or her time to the Computer Vision course, 30% to Data Mining, 20% to Machine Learning and 35% to the Optimization course. Of course this solution does not guarantee the highest grades possible for a student, it just shows the best possible grades given a limited time frame.

4 Data Analysis

In this section a linear regression objective function was analyzed using the AMPL solver. To do so, three regression objective scenarios were considered; simple, with L1 regression and with L2 regression. The minimas for each of these objectives for different values of lambda (L) were calculated, tablulated and compared with optimization trials conducted within the python frame work.

The model was defined as shown in the below:

```
# Note : code for all cases are the same, only the objective changes
# When running the program, please comment out non-targeted objectives

reset;
option solver loqo;
option loqo_options 'iterlim 30 dual', solver loqo;

param n >= 1;
param x1{1..n};
param x2{1..n};
param y{1..n};
var w1;
var w2;
var b;
param L = 10; # the value can be tuned

minimize f: sum{i in 1..n} (w1*x1[i] + w2*x2[i] + b - y[i])^2;
minimize f: sum{i in 1..n} (w1*x1[i] + w2*x2[i] + b - y[i])^2 + L*abs(w1 + w2);
minimize f: sum{i in 1..n} (w1*x1[i] + w2*x2[i] + b - y[i])^2 + L*(w1^2 + w2^2);

data prob_4_simple.dat;
solve;
display w1;
display w2;
```

Linear Regression	Lambda (L)	AMPL		PYTHON	
		w1	w2	w1	w2
Basic	-	0.0866	0.0192	0.0866	0.0192
with L1 regularizer	10	0.0756	-0.0187	0	0
	0	0.0866	0.0192	0.0866	0.0192
	0.1	0.0865	0.0188	0.0844	0.0116
	0.01	0.0866	0.0192	0.0864	0.0184
with L2 regularizer	100	0.0622	0.0169	0.0622	0.0169
	10	0.0833	0.0192	0.0833	0.0192
	0	0.0866	0.0192	0.0866	0.0192
	0.1	0.0866	0.0192	0.0866	0.0192

The .dat file was defined as below:

```
# Note: Please change value of n if number of instances are not equal to 10.
data;
param n := 10;
read {i in 1..n} (x1[i], x2[i], y[i]) < data.csv;
```

Note : The input data must be stored as a .csv file in the same directory in which the solver is running. Also, the model above is defined for only two input features. So the .csv file must consist of two features and a target.
The .csv file on which the models were tested can be found on our github repo.

From the above table we observe that the w1 and w2 values in case of simple regression is the same when solved in AMPL and Python. This is not the case with L1 regression. For some reason the values tend to deviate a bit for different values of Lambda (L). This could be due to different manner in which it's initialized on the python solver as opposed to the AMPL solver. This being said, the same situation isn't witnessed with L2 regression where values of w1 and w2 for different values of L are the same on AMPL and on Python.

5 Conclusion