Optimization assignment

Melvin Gode
Andrej Perković

February 2024

In this report, you will be able to find the general answers to the questions of the assignment. However, many of our answers being quite complicated to fully describe into text, we will only give the idea behind the answer and you will be able to find our implementation in the two notebooks handed along with the report. In particular, notebook *Projet Opti 5-office* describes our logic into more detail step by step with the implementation. Notebook *Projet Opti 13-office* follows the same ideas but without explanations and implements some more complicated techniques to adapt our solution to this larger version of the problem.
Have a nice read !

Question 1

1.1 Problem parameters

A few definitions.

- n : the number of offices
- p : the number of phases
- c : the number of colors

As suggested in the assignment, we are using a $p \times n \times n$ vector x (that can be represented as a matrix for easier readability) to represent the offices moves.

Ideally : $x_{kij} = \begin{cases} 1 & \text{if office } i \text{ moves to office } j \text{ in phase } k \\ 0 & \text{otherwise} \end{cases}$

Additionally, we are using a $p \times n \times c$ vector y to represent the color of the offices at each iteration.

Ideally : $y_{kij} = \begin{cases} 1 & \text{if office } i \text{ is of color } j \text{ in phase } k \\ 0 & \text{otherwise} \end{cases}$

To enforce correspondence between y values between phases, the formula we used in section (1.3) requires products of x and y. Thus, we have a third variable w of size $p \times n \times c \times n$[1] that is used to linearly represent these products.

1.2 Linear program

Since our constraints are challenging to formulate mathematically in one row, we will give the idea behind them here. You can find the corresponding implementation in the notebooks provided along with this file.

minimize number of changes s.t.

1. The initial assignment is respected.
   Fixed x and y entries for phase 0

2. Nobody moves to an office under construction.
   The sum of the columns of x corresponding to the offices in construction must be 0

3. No more than one office move to the same destination.
   All columns of x upper-bounded by 1

4. Occupied offices move somewhere and nothing comes out of an empty office.

   - If the office is occupied and not under construction, then the only spot where it can move is where it already is (except in the last phase where anybody can relocate anywhere).

   $\forall i$, the sum of row i must equal the sum of column i in the previous iteration

5. The final assignment is respected.
   For x : the sum of columns corresponding to assigned offices must be 1 and 0 for empty ones.
   For y : fixed entries for the last phase.

Additionally, for the 13 office version we added another natural inequality constraint

6. Office moved to at most one destination
   All rows upper-bounded by 1

The objective is implemented by a scalar product $c^T x$ where c is a size N vector full of ones except on the entries corresponding to the diagonals of the matrices which represent moves where an office stays where it already is.

[1] We only need $(p-1) \times n \times c \times n$ elements since there is no correspondence constraint for the first phase but we artificially extend w so that dimensions match with y which is needed for our implementation of the correspondence constraint.

1.3 Color logic

If (3.) is respected, then the color of an office in phase k is the color of the single office that moved to it in phase $k-1$. Thus

$$y_{k,i,j} = 1 \iff (y_{k-1,e,j} = 1 \wedge x_{k,e,i} = 1)$$

for some office e.

But since we are (ideally) dealing with boolean variables, this conjunction can be replaced with the product of the two variables. So we have

$$y_{k,i,j} = y_{k-1,e,j} \times x_{k,e,i}$$

Now if we generalize this to cover all offices with e, we have :

$$y_{kij} = \sum_{e=1}^{n} y_{k-1,e,j} \times x_{k,e,i}$$

We can use a sum thanks to (3.) which assures us that we only have a single one in column $x_{k,\cdot,i}$

This means that for every y past phase 1, we are going to need n w variables representing the product of the two variables we need.
To represent the product between the entries of x and y, the formula for w is the following :

$$w_{k,i,j,e} \leq x_{k,e,i}$$

$$w_{k,i,j,e} \leq y_{k-1,e,j}$$

$$w_{k,i,j,e} \geq y_{k-1,e,j} + x_{k,e,i} - 1$$

And with all that we finally have our simplified formula for y :

$$y_{kij} = \sum_{e=1}^{n} w_{k,i,j,e}$$

Question 2

Please, consult the notebook for the implementation of the LP.

Question 3

3.1 5-Office Version

For the 5-office plan, we obtain only ones and zeros in our x and y variables. Thus we can directly apply the principles defined in the definition of the variables (1.1).

To help with translating the result into a plan, we created two functions `plan_interpretation` and `detail_phase` respectively enumerating which offices should move where in each phase and detailing which departments occupy which offices in a given phase.

For example, the detailed plan output for this small problem is the following :

```
 Number of moves : 8

Phase 1 :
-Office 2 moves to office 1

Phase 2 :
-Office 3 moves to office 2

Phase 3 :
-Office 4 moves to office 3

Phase 4 :
-Office 5 moves to office 4

Phase 5 :
-Office 1 moves to office 2
-Office 2 moves to office 3
-Office 3 moves to office 4
-Office 4 moves to office 5
```

3.2 13-Office Version

Now, for this more complex version of the problem, the solver outputs some floating point values. This is the consequence of inherent lack of constraints in the given problem that would uniquely determine the transfers. What the solver then outputs is essentially a discrete uniform probability distribution of the transfer of a given office. For example, in phase 1 when the offices of wing B have to be moved to wing N, every such office can be placed in either of the three empty spots in wing N without violating the inherent constraints. Hence, it is "equally likely" to place an office in any of the 3 empty spots.

Our approach we would use to tackle this challenge would be to push "indecisive" entries to be ones or zeros by iteratively adding constraints. What we would do is that after the initial output with non integer values, we would go through the whole x array and find the first entry which has fractional value. We would then start a backtracking search where we add a constraint setting this variable to zero and repeating the process, backtracking to set it to one if no feasible solution was found in that branch.

We even made a very serious attempt to code this out, under the function `iter_res`, although the current form, after some 400 iterations, returns that the

problem is infeasible.

However, even though our solution contains floating point numbers, it is worth noting that these follow a certain logic. For example in phase 1, we see the offices of wing B moving into wing N by spreading $\frac{1}{3}$ of each office to each office of N. This simply means that instead of assigning a fixed office of N to each one of B, they can all move in any office of N. The same behavior repeats for all phases. Instead of giving us a precise office for each office to move to, this in fact gives us a range of offices in which several offices to move in.

Thus, while not as intuitive as in the 5-office version, the result is still logical and can be translated into a plan if we dive into these details.

Question 4

4.1 Constraint principle

To model this constraint we have constructed the graphs representing the layout of the problems (for both 5 and 13-office versions).

In every phase k, for every edge (u, w), we add a constraint $y_{k,u,3} + y_{k,w,4} \leq 1$ and another constraint $y_{k,u,4} + y_{k,w,3} \leq 1$ for symmetry of the edge.

Since 3 is the color representing the student association and 4 is the color representing the presidency, these constraints formulate that any pair of neighbor offices in any phase should not be of these two departments at the same time.

In other words, we can't have offices from the presidency neighboring offices from the student association.

For this section to work, it is important to note that we have to relax constraint (4.) to allow offices to move elsewhere even when their wing is not under construction.
For example, in phase 1 for the 13 office plan, we need to separate offices $P1$ and $C1$ (move them so they are not neighbors). This means we will have to move one of these offices elsewhere even though their wings are not under construction.

4.2 Results

As we can quickly verify for ourselves, these constraints are not satisfiable in additions to the other ones for the 5-office plan. The solver is consistent with this observation since it outputs that the problem is infeasible.

Question 5

All we have to do for this question is add the quadratic term described above to our objective function.

For this we have created a vector z^F which is a concatenation of 4 times the final color assignment (one time for each intermediate phase). And we simply

add to the objective $\lambda||z^F - y_{1:4}||^2$ with $y_{1:4}$ representing color assignments during all 4 intermediate phases.

Question 6

Please, consult the notebook for the implementation of the QP.

Question 7

For the 5-office plan, it seems like (and this is easily verifiable) there is a single plan satisfying all the constraints. Thus, no matter how much we might penalize it in the objective, we cannot do better than that and the solution always stays the same.

As for question 4.1, we need to relax constrain 4 in order to allow offices to move away even when they are not under construction. Otherwise we won't have enough flexibility to allow offices to move closer to their final assignment in intermediate phases.

Note that we had to change solver to `ECOS` to make the LP run with this new objective.

Question 8

We can easily replace z^F by z^I by using the same process of concatenating a vector 4 times, except this times it is the initial color assignment vector.

For the 5-office plan, we still have the same unique solution. For the 13-office plan, we get the matrix with fractional entries like in the linear program, but the execution is much faster.

Question 9

For this SDP, we can replace x and y with the u vector and w with the U matrix. We want the first $p \times n \times n$ entries of u to represent x while the other $p \times n \times c$ represent y (so u is of length $p \times n(n + c)$). However, as specified in the assignment, we want to go from $\{0, 1\}$ values to $\{-1, 1\}$ values. We can achieve this by doing a simple $\phi(t) = 2t - 1$ rescaling on x and y.

Now, since the matrix U already includes all x and y products, we don't need w anymore to simulate this.

We now look for a way to find the value of $y_{k-1,e,j} \times x_{k,e,i}$ as per the formula in 1.3. The entry of U corresponding to the product of the two (that we are going to denote by U_{xy} for simplicity of the indices) is not going to be enough because a value of 1 could be coming from the product of two 1s but also from the product of two -1s. Thus, we need to include the values from the vector u in the formula.

The sum $U_{xy} + u_{kn^2+en+i} + u_{pnn+(k-1)nc+ec+j} + 1$([2]) takes value 4 if both the entries of interest are 1 and 0. otherwise. Thus we can conclude that the resulting formula for color correspondence in our SDP is

$$u_{kij} = \frac{1}{2}(U_{xy} + u_{kn+en+i} + u_{pnn+(k-1)nc+ec+j} + 1) - 1, \forall k \geq 1$$

The rest of the constraints stay the same and are simply applied to the corresponding entries of u instead of x and y. We also have to replace target values in the vector b from 0 to $-1\times$ the number of entries taken into account, in order to match our new range of values.

Question 10

With this transformation, we got rid of our $p \times n \times c \times n$ entries for w. However, we added $(p \times n(n+c))^2$ new entries for U which is a lot more.

The real gain we make however, is on the constraints. We got rid of three constraints per entry of w to make it correspond to the product of the entries of x and y. Which means we have a net total of $3 \times p \times n \times c \times n$ less constraints.

In conclusion to this question, this new SDP model uses a lot more variables but reduces the number of constraints by a sizeable amount.

Question 11

Question 12

We can easily transfer the part of the objective from question 5 by using the entries of u corresponding to those of y (by simply adding $p \times n \times n$ to them to skip over the "x" entries). We also have to change the target vector z^F (or z^I) by changing the 0s into -1s. The range of differences is thus doubled and we can divide λ by two if we want to keep the exact same weight in the objective. Thus we have

$$\frac{\lambda}{2} \times ||(2z^F - 1) - u_{1:4}||^2$$

where $u_{1:4}$ is the vector of the u entries ($p\times n\times n+n\times c$) to ($p\times n\times n+(p-1)\times n\times c$).

[2]Note that u_{kn^2+en+i} and $u_{pnn+(k-1)nc+ec+j}$ are the entries of u corresponding respectively to $x_{k,e,i}$ and $y_{k-1,e,j}$.