Needle in a Haystack – Office Space Edition

Gary Sandine, Portland State University

May 27, 2022

1 Introduction

These notes and the corresponding presentation I will give in the PSU analysis seminar arose from a practical work situation I encountered in my job in the PSU Office of Information Technology (OIT). We moved into a new (for us) office space on the PSU campus, and office assignments were made with the assumption that all of our approximately 100 regular employees would be working on campus 3 or more days per week post-pandemic. A refined remote work program and capability was developed out of necessity during the pandemic, and it worked so well for us and our employees that it was determined that remote work was a possibility indefinitely for not all but many positions within OIT, and 20% or so of our staff has opted to work remotely most of the time. Thus, the initial office assignments resulted in a suboptimal usage of our office space based on who was actually coming to campus for work. A project was initiated to choose a new office assignment floor plan based on who was actually coming to campus for work and subject to optimality criteria and several constraints.

The best identified way to approach this task was to use an online program called Lucidchart with an image of the offices and to locate small boxes representing individuals on top of the office they would inhabit. This proved to be cumbersome and severely limits the number of potential floor plans that can easily be explored (it would take me about an hour to start from an empty diagram and generate *one* floor plan to fill in the space in a way that meets the objectives and constraints). Additionally, this approach naturally encourages one to fit ones own units with whom one has familiarity into space that accommodates those units and their needs well, leaving the remaining units and space to be assigned in a way that is naturally less optimal due to lack of familiarity with the intricacies of those other units.

After several false starts and a couple of weeks of intermittently thinking about the problem consciously (and subconsciously while sleeping and during periods of solitude and downtime), I awoke one Sunday morning with a clear way to formulate the problem that looks simple and obvious once written down but which provides tremendous flexibility with regard to optimizing for the objectives and meeting constraints. This offers us the additional ability to somewhat objectively compare floor plan proposals with regard to how well they meet our optimization criteria as long as we have captured those properly.

2 Linear Programming (LP)

Linear programming formalism appeared in the 1940s out of a need to address profoundly impactful WWII-era optimization problems, for example for planning expenditures to reduce costs and increase tactical effectiveness, and for allocating resources for maximal effectiveness with minimal financial and personnel losses.

The basic LP problem looks like

$$\begin{aligned} \text{maximize: } & c^T x \\ \text{subject to: } & Ax \le b \\ & x \in \mathbb{R}^n_+ \end{aligned}$$

where $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and the inequalities are component-wise. The function $x \mapsto c^T x$ is referred to as the objective function, and points x that satisfy the constraint inequality are referred to as feasible points. If there are no feasible points, the problem is called infeasible. A situation with no solution can also arise when the feasible region is unbounded in a direction of increase for the objective function.

There are methods for adding additional variables to turn the constraint inequality $Ax \le b$ into an equality, thereby transforming the problem into a canonical form that can be easily solved following the simplex algorithm which was developed by George B. Dantzig in the 1940s while tackling planning problems in the US Air Force. Other early pioneers include economists Leonid Kantorovich and T.C. Koopmans and mathematician John von Neumann who introduced duality methods for framing and solving LPs.

For an LP problem with a solution, the simplex method effectively starts on a vertex and traverses the feasible polyhedron from vertex to vertex improving the cost function at each step until it arrives at a point providing an optimal objective function value. This movement is called PIVOTING, and there are various pivot rules one can follow such as a greedy pivot rule that moves in the direction that offers the most rapid improvement to the objective function.

3 Integer Linear Programming (ILP)

An INTEGER LINEAR PROGRAMMING problem is an LP problem with the additional constraint that the variable x assumes integral values only:

$$\begin{aligned} \text{maximize: } & c^T x \\ \text{subject to: } & Ax \leq b \\ & x \in \mathbb{Z}^n_+. \end{aligned}$$

Whereas with an LP problem one can use the simplex method to solve it efficiently, the method used to solve an ILP problem is typically problem-dependent. Some common methods are brute force enumeration, implicit enumeration with a branch and bound method, rounding non-integer solutions to integers, or using a cutting plane method. The simplex method using any pivot rule generally completes in polynomial time for LP problems (in fact, it generally completes in *linear* time in the number of constraints m in practice), whereas there is no known polynomial time algorithm for solving a general ILP problem. In fact, the minimum vertex cover problem is NP-hard and it can be reduced to an ILP problem, meaning that ILP is NP-hard (it is at least as hard as min vertex cover) [2].

Implicit enumeration using branch and bound is applicable to the problem that motivated this talk, and we provide a small example that we could easily solve with brute force enumeration but that nicely demonstrates the features of branch and bound implicit enumeration. The idea is, relax the integral requirement and solve an identical LP. For a maximization problem, this puts an upper bound on the solution of the actual ILP (the nearest integer less than or equal to the solution to the relaxed LP problem), and from there, the problem "branches" into multiple subproblems by adding additional constraints bounding non-integer solutions by integers on either side of the non-integer values that were found as optimal points for the associated LP. Likewise, any integral objective-function evaluations within sub-problems place lower bounds on the maximum value of the actual ILP. These lower bounds allow us to discard relaxed LP branches associated with sub-problems whose optimal LP solution values are less than a previously identified lower bound for the actual ILP problem. Similarly, the upper bound found in the first relaxed LP allows one to stop, say when a lower bound attained along the way is within some percentage of the identified upper bound.

This is best demonstrated with a small example from [1]:

$$\begin{aligned} \text{maximize: } f(x_1, x_2) &= 5x_1 + 8x_2 = z \\ \text{subject to: } x_1 + x_2 &\leq 6 \\ 5x_1 + 9x_2 &\leq 45 \\ x_1, x_2 \geq 0,\ x_1, x_2 &\in \mathbb{Z} \end{aligned} \tag{1}$$

Fist, we relax the integral requirement and solve an identical LP using the simplex method – call this problem L_0:

$$\begin{aligned} L_0 - \text{maximize: } f(x_1, x_2) &= 5x_1 + 8x_2 \\ \text{subject to: } x_1 + x_2 &\leq 6 \\ 5x_1 + 9x_2 &\leq 45 \\ x_1, x_2 &\geq 0 \end{aligned}$$

The solution to problem L_0 is

$$(x_1, x_2) = (2.25, 3.75) \text{ maximal value } \overline{z}_0 = 41.25.$$

Now $\overline{z}_0$ is an upper bound for the actual integer programming problem since the actual maximal value for the problem (when the problem is bounded on the feasible set) will occur at a point (x_1^*, x_2^*) in the integral subset F of the feasible set $\overline{F}$ for the associated, relaxed LP. That is,

$$\sup_{(x_1,x_2)\in F} f(x_1, x_2) \leq \sup_{(x_1,x_2)\in \overline{F}} f(x_1, x_2) \text{ since } F \subset \overline{F}.$$

Such an upper bound for the initial LP relaxation of the problem can be useful in practice because for each evaluation of the objective function at an integral feasible point, one immediately has an estimate for how close one is to the true optimum value of the ILP. Since $x_1, x_2 \in \mathbb{Z}$, we now know that if z^* is the optimal value, then $z^* \leq 41$ since it too must be an integer, and then any z value from feasible $(x_1, x_2) \in \mathbb{Z}_+^2$ is within $41 - z$ of z^*.

Now we have the option to branch on x_1 or x_2 obtaining new relaxed LP problems L_1 and L_2 by adding two integral constraints for either x_1 or x_2 from the list

$$x_1 \leq 2,\ x_1 \geq 3,\ x_2 \leq 3, \text{ and } x_2 \geq 4$$

from the continuous LP solution $(x_1, x_2) = (2.25, 3.75)$ to L_0. We branch on x_2 obtaining relaxed linear programming problem L_1 by adding the constraint $x_2 \geq 4$ to problem L_0 and L_2 by adding the constraint $x_2 \leq 3$ to problem L_0:

$$\begin{aligned} L_1 \text{ – maximize: } f(x_1, x_2) &= 5x_1 + 8x_2 \\ \text{subject to: } x_1 + x_2 &\leq 6 \\ 5x_1 + 9x_2 &\leq 45 \\ x_2 &\geq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

$$\begin{aligned} L_2 \text{ – maximize: } f(x_1, x_2) &= 5x_1 + 8x_2 \\ \text{subject to: } x_1 + x_2 &\leq 6 \\ 5x_1 + 9x_2 &\leq 45 \\ x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Next, solve problem L_1 obtaining

$$(x_1, x_2) = (1.8, 4) \text{ maximal value } \overline{z}_1 = 41,$$

and solve problem L_2 obtaining

$$(x_1, x_2) = (3, 3) \text{ and maximal value } \overline{z}_2 = 39.$$

No further branching is required on problem L_2 – the maximum value over all real numbers in the feasible region for problem L_2 is obtained at $(x_1, x_2) = (3, 3)$, thus 39 is also a lower bound for z^*. We note here that $(3, 3)$ is the first integral feasible point at which we have evaluated the objective function.

However, we can branch on x_1 from problem L_1 adding the constraints $x_1 \geq 2$ and $x_1 \leq 1$ to get new problems L_3 and L_4:

$$\begin{aligned} L_3 \text{ – maximize: } f(x_1, x_2) &= 5x_1 + 8x_2 \\ \text{subject to: } x_1 + x_2 &\leq 6 \\ 5x_1 + 9x_2 &\leq 45 \\ x_2 &\geq 4 \\ x_1 &\geq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

$$\begin{aligned} L_4 \text{ – maximize: } f(x_1, x_2) &= 5x_1 + 8x_2 \\ \text{subject to: } x_1 + x_2 &\leq 6 \\ 5x_1 + 9x_2 &\leq 45 \\ x_2 &\geq 4 \\ x_1 &\leq 1 \\ x_1, x_2 &\geq 0 \end{aligned}$$

In L_3, the constraints $x_1 \geq 2$, $x_2 \geq 4$ (inherited from the upstream problem L_1), and $5x_1 + 9x_2 \leq 45$ (inherited from the upstream problem L_0) are contradictory and thus problem L_3 is infeasible. We can solve problem L_4 though obtaining

$$(x_1, x_2) = (1, 4.\overline{4}) \text{ and maximal value } \overline{z}_4 = 40.\overline{5}.$$

This tells us that z^* is bounded above by 40.

We branch one more time on x_2 adding the constraint $x_2 \leq 4$ to get problem

L_5 and adding the constraint $x_2 \geq 5$ to get problem L_6:

$$\begin{aligned} L_5 \text{ – maximize: } f(x_1, x_2) &= 5x_1 + 8x_2 \\ \text{subject to: } x_1 + x_2 &\leq 6 \\ 5x_1 + 9x_2 &\leq 45 \\ x_2 &\geq 4 \\ x_1 &\leq 1 \\ x_2 &\leq 4 \text{ (so } x_2 = 4 \text{ down this branch)} \\ x_1, x_2 &\geq 0 \end{aligned}$$

$$\begin{aligned} L_6 \text{ – maximize: } f(x_1, x_2) &= 5x_1 + 8x_2 \\ \text{subject to: } x_1 + x_2 &\leq 6 \\ 5x_1 + 9x_2 &\leq 45 \\ x_2 &\geq 4 \text{ (this constraint is no longer needed)} \\ x_1 &\leq 1 \\ x_2 &\geq 5 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Solving L_5 gives the solution

$$(x_1, x_2) = (1, 4) \text{ and maximal value } \overline{z}_5 = 37.$$

This is the second integral feasible point at which we evaluated the objective function, and it is not as good as the point $(3, 3)$ from problem L_2.

we next solve problem L_6 and get

$$(x_1, x_2) = (0, 5) \text{ and maximal value } \overline{z}_6 = 40.$$

We have thus found the maximal value $z^* = 40$ attained at $(x_1, x_2) = (0, 5)$. While there were 25 feasible integral (x_1, x_2) pairs, we only had to evaluate the objective function at 3 of them before we found an optimal pair while implicitly checking all 25 feasible integral pairs (although we did have to solve 7 linear programs).

4 Binary ILP – Implicit Brute Force

A special type of ILP problem is a BINARY INTEGER LINEAR PROGRAMMING problem, where the variables may assume the values 0 or 1 only:

$$\begin{aligned} \text{maximize: } & c^T x \\ \text{subject to: } & Ax \leq b \\ & x \in \{0,1\}^n. \end{aligned}$$

This type of problem can capture any yes-no decision problem, and the office assignment problem that motivated this talk can be captured in a binary ILP framework (yes or no to the assignment of office i to person j). When the objective function assumes a special form such that all 0s or all 1s provide an optimal value, it is possible to apply a special branch and bound procedure to solve these problems *without having to solve LP problems along the way!* The solution spaces for these problems quickly become intractable (there are 2^n 0-1 combinations for an n-variable problem) and for this method rather than dropping the integrality constraint, we maintain the 0-1 constraint but drop the linear inequality constraints and rule out large swaths of the solution space time and again until we arrive at a solution.

To apply this method, the objective function $x \mapsto c^T x$ takes on a special form where either $c \leq 0$ or $c \geq 0$ holds (component-wise). Then, for example, in the $c \leq 0$ case, the max occurs when $x_i = 0$ for every i and the min occurs when $x_i = 1$ for every i. Once we start developing bounds for the optimum value, we use this feature to rule out large swaths of the solution space. Each branch starts with a subset of variables that are fixed and the remaining are free to be set to all 0s or all 1s (depending on the objective function and if it is a min or a max problem). As we will see below, branches in this special branch and bound situation are implicitly fully explored if one of three conditions arise (here enumerated for $c \leq 0$ and a max problem):

1. there are no feasible points in a branch;
2. setting all free variables to 0 results in a feasible point (this provides the best possible objective function value in a branch); or
3. the objective value obtained by setting all free variables to 0 is no better than a feasible solution previously generated.

This too is nicely demonstrated with a small example, again from [1]:

$$\begin{aligned} \text{maximize: } & f(x) = -8x_1 - 2x_2 - 4x_3 - 7x_4 - 5x_5 = z \\ \text{subject to: } & -3x_1 - 3x_2 + x_3 + 2x_4 + 3x_5 \leq -2 \\ & -5x_1 - 3x_2 - 2x_3 - x_4 + x_5 \leq -4 \\ & x_i \in \{0,1\} \text{ for } i = 1, \dots, 5 \end{aligned} \tag{2}$$

Note that in this problem, if any variables are fixed at 0 or 1, then setting the remaining free variables to 0 provides an upper bound for the best one can do given the current fixed variables. This is the principle that will be used again and again in this special branch and bound process.

We begin by setting $x_i = 0$ for all i. While this x is not feasible, note that $f(x) = 0$ and this is an upper bound for the maximal value. Since it is not feasible, we begin our branching process starting (arbitrarily) with x_1 obtaining two branches, one for $x_1 = 1$ and one for $x_1 = 0$. As mentioned above, the best possible objective function value in each branch are attained by setting $x_2 = \cdots = x_5 = 0$.

Branch 1 ($x_1 = 1$): Here, $x_1 = 1$ and $x_2 = \cdots = x_5 = 0$ is a feasible point, thus the best objective function value in this entire branch is $\overline{z}_1 := -8$ attained at $x = (1, 0, 0, 0, 0)$. We have effectively checked all 16 possibilities down this branch by checking a single point. Additionally, since $(1, 0, 0, 0, 0)$ is a feasible point, we have that $\overline{z}_1$ is a lower bound for the optimum value, or $z^* \geq -8$. This will allow us to rule out additional branches whose best values are no better than $\overline{z}_1$.

Branch 2 ($x_1 = 0$): Setting $x_2 = \cdots = x_5 = 0$ results in an infeasible point $x = (0, 0, 0, 0, 0)$. However, in this branch we have

$$f(x) = 0 > -8 = \overline{z}_1$$

thus it is possible the objective function can be improved by branching further, so we will branch on x_2 (branches 3 and 4).

Branch 3 ($x_1 = 0, x_2 = 1$): Here too, $x = (0, 1, 0, 0, 0)$ is infeasible but

$$f(x) = -2 > -8 = \overline{z}_1$$

thus it is possible the objective function can be improved by branching further, so we will branch on x_3 (branches 5 and 6).

Branch 4 ($x_1 = 0, x_2 = 0$): In this branch, it is not possible to satisfy either of the inequality constraints, thus we have essentially checked all remaining 8 points and they are all infeasible. This branch ends here.

Branch 5 ($x_1 = 0, x_2 = 1, x_3 = 1$): Here, the point $x = (0, 1, 1, 0, 0)$ is feasible and $f(x) = -6 > \overline{z}_1$ so we have a new lower bound $z^* \geq \overline{z}_5 := -6$ and we have implicitly explored the rest of this branch.

Branch 6 ($x_1 = 0, x_2 = 1, x_3 = 0$): In this branch, $x = (0, 1, 0, 0, 0)$ is not feasible but

$$f(x) = -2 > -6 = \overline{z}_5$$

so we branch here on x_4 (branches 7 and 8).

Branch 7 ($x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$): Here, $x = (0, 1, 0, 1, 0)$ is not feasible and

$$f(x) = -9 < -6 = \overline{z}_5,$$

so there is no value attainable down this branch that exceeds the current lower bound $\overline{z}_5$. No further exploration is required down this branch.

Branch 8 ($x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 0$): Here, the point $x = (0, 1, 0, 0, 0)$ is not feasible, nor is the point corresponding to $x_5 = 1$. This branch ends here.

The whole space has implicitly been explored and we have found the maximum value

$$z^* = \overline{z}_5 = -6 \text{ is attained at } x^* := (0, 1, 1, 0, 0)$$

by essentially only evaluating 5 out of 32 potential x values.

This example is visualized in Figure 1.

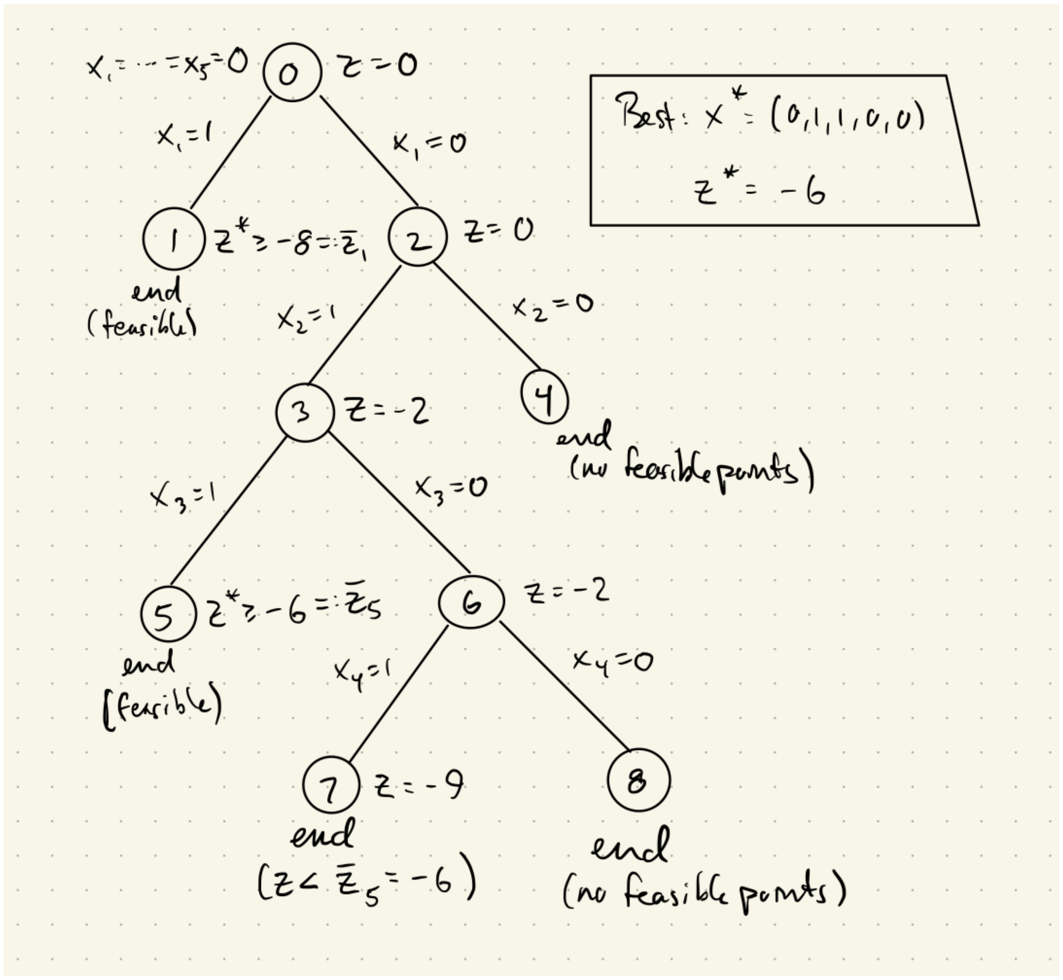

Figure 1: Visualization of binary ILP implicit brute force

5 Office Assignment Problem

5.1 Problem overview

Our desire is to place 80 staff members into 90 offices, 20 of which have windows. As there are a limited number of offices with windows, an assignment plan must ensure all 15 members of the OIT leadership team are assigned to window offices and as many of the remaining 10 OIT managers as possible are assigned to window offices (thus, 5). Additionally, it would be beneficial for individuals who work together often and who come to campus for work to have offices that are near each other. These boil down to the following optimality criterion:

- minimize the distance between individuals who work together frequently

and the following constraints:

- staff members who come to campus for work have their own office with floor to ceiling walls and a door that closes;
- all members of the OIT leadership team have offices with windows; and
- as many OIT managers as possible have offices with windows.

There are on the order of 10^{131} ways to assign 80 people into 90 offices ($90!/10!$), therefore a full brute force approach consisting of formulating an objective function, calculating the cost of all assignments, and choosing an assignment plan from among the lowest cost feasible assignment plans is out of the question. I also considered the possibility of formulating the problem as an integer program where the variables x_i represent people and their values are an office number between 1 and 90. I can imagine ways to capture the unique office assignment constraint but was unable to formulate the window constraint or capture the proximity objective in this format.

This problem has the flavor of a well understood job assignment type problem where we seek to populate an $m \times n$ matrix X corresponding to m jobs (I will say offices) and n people where $x_{ij} = 1$ if person j is assigned to office i, and there is a fixed cost matrix C where c_{ij} is the cost for assigning person j to office i. This does not capture this problem either, for in this case, c_{ij} would depend on where teammates were assigned and the distances between those offices.

This problem can be captured as a binary integer program – albeit with a nonlinear cost function – which I will detail in the next subsection.

5.2 Office assignments as an integer program

We begin with the variables we will use to capture the optimization criterion and the constraints.

- There are m offices and n people with $m \geq n$.
- For $k \in \mathbb{N}$, we use 1_k to represent the vector in $\mathbb{R}^k$ consisting of k 1s, and we use e_k to represent the kth standard unit vector in $\mathbb{R}^n$ ($e_k \in \mathbb{R}^n$ and all entries of e_k are 0 except for the kth entry which is 1).
- $w \in \mathbb{R}^m$ is a vector of 0s and 1s with $w_i = 0$ if office i does not have a window and $w_i = 1$ if it does.
- $D \in \mathbb{R}^{m \times m}$ is the nonnegative, symmetric OFFICE DISTANCE MATRIX where the i, jth entry d_{ij} is the distance from office i to office j.
- $\ell, \overline{\ell} \in \mathbb{R}^n$ are vectors of 0s and 1s with $\ell_j = 1$, $\overline{\ell}_j = 1$ if the jth person is in the OIT leadership team or is an OIT manager, respectively. Note that all members of the OIT leadership team are managers but not conversely, thus $\overline{\ell}_j = 1$ whenever $\ell_j = 1$ and $|\ell| < |\overline{\ell}|$ ($|\ell| = \sum_{j=1}^{n} \ell_j$).
- $A \in \mathbb{R}^{n \times n}$ is a symmetric AFFINITY MATRIX where the i, jth entry $a_{ij} = e_i^T A e_j$ is a weight for how important it is for people i and j to be near each other in an assignment plan.
- $X \in \mathbb{R}^{m \times n}$ is the binary ASSIGNMENT MATRIX where the i, jth entry x_{ij} is 1 if person j is in office i and is 0 otherwise. There are constraints on this matrix that must be captured in the problem constraints – namely, the rows must sum to 1 or 0 (depending on whether an office is assigned or not, and each office can be assigned only once) and the columns must sum to 1 (each person is assigned to 1 and only 1 office).
- $T := X^T D X \in \mathbb{R}^{n \times n}$ is the nonnegative, symmetric STAFF DISTANCE MATRIX, where the i, jth entry $t_{ij} = e_i^T T e_j$ is the distance from person i to person j according to the assignment plan represented by an assignment matrix X.

- For a cost function, we simply add up the products of the distances between individuals and their affinities:

$$\begin{aligned} f(X) &:= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(e_i^T A e_j \cdot e_i^T X^T D X e_j\right) \qquad (3) \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(e_i^T A e_j \cdot e_i^T T e_j\right) \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij} t_{ij}. \end{aligned}$$

We are now ready to state the binary integer programming problem:

$$\begin{aligned} \text{minimize:}\quad & f(X) \\ \text{subject to:}\quad & X 1_n \le 1_m && \text{(all staff have their own offices)} \\ & X\ell \le w && \text{(associate directors and up have windows)} \\ & X\overline{\ell} \ge w && \text{(as many managers as possible have windows)} \\ & X^T 1_m = 1_n && \text{(every person is assigned to an office)} \\ & x_{ij} \in \{0,1\} \text{ for all } i,j. \end{aligned}$$

While the objective function f is nonlinear, it does meet the criteria that all of the coefficients in the sum (3) are nonnegative, thus with a subset of the x_{ij} variables fixed, setting the remaining free variables to 0 does result in a minimal value for the objective function. Thus, there is hope that the special binary ILP implicit brute force method detailed in Section 4 can be applied to solve this problem (here for a min problem rather than for a max problem).

5.3 Refinements - pre-assignments and minimizing moves

It is possible to make some minor modifications to the problem that allow us to make some pre-assignments that will appear in any solution. Also, as it would be disruptive and costly to move *everyone*, we can also add a parametrized data fidelity term that penalizes moves that allows us to tune how many moves there are in an optimal solution. If we let $\overline{X}$ be a partial assignment matrix consisting of pre-assignments and let X_0 be an assignment matrix corresponding to the current office assignments, solving this modified problem will preserve the pre-assignments and the parameter λ can be increased to reduce the number of moves:

$$\begin{aligned}
\text{minimize: } & f(\overline{X}+X)+\lambda\|\overline{X}+X-X_0\|^2 & \\
\text{subject to: } & (\overline{X}+X)1_n \le 1_m & \text{(all staff have their own offices)} \\
& (\overline{X}+X)\ell \le w & \text{(associate directors and up have windows)} \\
& (\overline{X}+X)\overline{\ell} \ge w & \text{(as many managers as possible have windows)} \\
& (\overline{X}+X)^T 1_m = 1_n & \text{(every person is assigned to an office)} \\
& x_{ij}\in\{0,1\} \text{ for all } i,j. &
\end{aligned}$$

When $\lambda = 0$, $X = 0$ does still provide a minimal cost (but infeasible) solution so the binary implicit brute force method would work. However, that is no longer the case when $\lambda > 0$ and a different solution method would be needed.

References

[1] S.P. Bradley, A.E. Hax, A.C. Hax, and T.L. Magnanti. *Applied Mathematical Programming*. Addison-Wesley Publishing Company, 1977.

[2] C. Moore and S. Mertens. *The Nature of Computation*. OUP Oxford, 2011.