Computational Methods in Optimization

M1 IDD

Course project - 2023/2024

- The last version of this document can be found at: **https://www.lamsade.dauphine.fr/~croyer/ensdocs/CMO/ProjCMO.pdf**.
- Typos, questions, etc, can be sent to `clement.royer@lamsade.dauphine.fr`.
- Current version (includes minor edits): January 19, 2024.
- **Major updates to the document**
  - 2024.01.19: Clarified the example nature of the notations from Parts 1 and Parts 2. Fixed the formulation of the SDP in Part 3.

Assignment

- This project is a collaborative one. Students should organize in groups of 3 or 4.
- Students may submit their sources in either French or English. Those sources should include:
  - Their answers to the questions (in PDF or notebook format).
  - Their Python implementation (in .py files or in a notebook).
  - A Python script or a notebook to run the methods and reproduce the results.
  - A short report in PDF file explaining their approach to modeling and solving the optimization problems at hand.
- Please send your sources to `clement.royer@lamsade.dauphine.fr` under the form of a compressed folder. Those sources must include the last names of every member of the group.
- The deadline to send the sources is **February 2, 2023 AOE** (Anywhere On Earth).

Project: University office space reallocation

Introduction

Context Université Paris Dauphine-PSL is currently being renovated, with a new wing (thereafter called wing N) opening mid-2024. The next renovation phases are planned as follows:

1. Wing N opens and wing B is renovated;
2. Wing B re-opens and wing P is renovated;
3. Wing P re-opens and wings C/D are renovated;
4. Wings C/D re-open and wing A is renovated;
5. Wing A re-opens.

A key part of the renovation consists in planning the moves of personnel at every phase, so as to minimize the overall moving effort. Although solutions computed by hand have been proposed, the administration is working with researchers in LAMSADE to certify optimality of the solution according to several criteria.

Task In this project, we investigate a (greatly) simplified version of that problem, where the intended goal is to get from the current office configuration (called "Phase 0") to a desired office configuration in the final phase ("Phase 5"). We assume that all offices have the same size, therefore an office (and its occupants) can be moved to any other office that is not under renovation. Although our problem data will be defined according to a graph of the offices, we will not leverage the structure as much as we could (in particular, we will not take the distances between offices into account).

A lot of freedom is intentionally left in the modeling and implementation of the various problems and solutions. This freedom will be used to compare the solutions proposed by different groups. The project consists of three parts, that are meant to be completed in that order.

Problem data We proposed to use two sketch graphs representing situations one may encounter while considering the overall graph of Dauphine. We consider offices belonging to four entities : **LAMSADE**, **MIDO**, **Students association** and **Presidency**. To mimic the true office graph, the offices are designated by a code, and the first letter corresponds to the wing[1]. Still, the graph topologies are not essential for most of the project except Question 4.

The first graph, given in Figure 1, considers a very simple case where each entity has only one office, and the new wing adds another empty office. The final and initial affectations are identical, and the moving plan must manage to move offices that are under renovation while reaching the final affectation.

The second graph example involves different numbers of offices per entity. The final allocation differs from the initial one, though the number of offices associated with every entity remains constant throughout. Figure 2 details these allocations.

Depending on how the lab sessions go, other graph structures might be added to the project to further test the model. Nevertheless, the validation (and grading) will be based upon the two graphs from Figures 1 and 2.

[1] We ignore wing D for simplicity

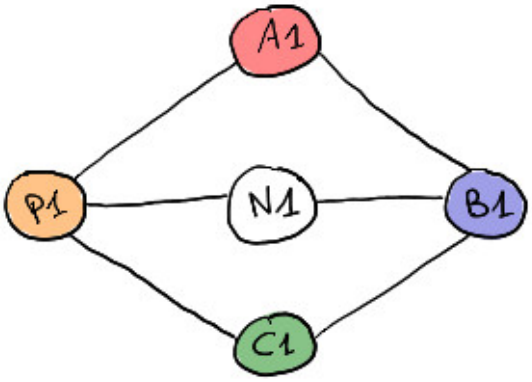

Figure 1: *A first graph example with same initial and final affectations. The office N1 (corresponding to the new wing) is empty at the final affectation. The office A1 is affected to Presidency, B1 is affected to MIDO, C1 is affected to Students association and P1 is affected to LAMSADE.*

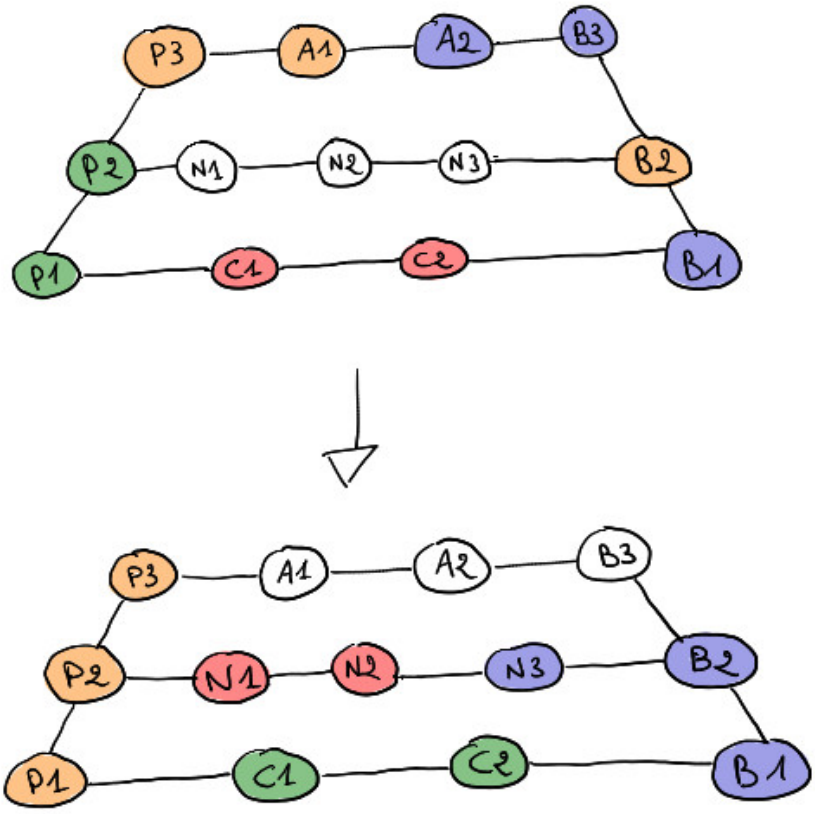

Figure 2: *A second graph example where the initial office allocation (top) differs from the final allocation (bottom). The colors correspond to Presidency, Students association, MIDO and LAMSADE.*

1 Linear programming relaxation

A linear relaxation of the proposed approach consists in replacing binary decision variables by continuous ones. As an example[2], given a set of offices $\{1, 2, \ldots, n\}$ and a set of phases $\{0, \ldots, 5\}$, consider x_{ijp} that indicates whether the current occupants of office i at phase p should move to office j during phase p. This is in essence a binary variable, and thus its value should be in $\{0, 1\}$. In this section, however, we consider a relaxation of the *integer* constraint $x_{ijp} \in \{0, 1\}$ into the continuous constraint $0 \leq x_{ijp} \leq 1$.

Question 1 Write a linear programming model for this problem. Indicate the number of variables, constraints, problem data. The use of the variables defined above is not mandatory, and additional variables will likely be necessary. The whole model should differ from a binary optimization problem only in the integer constraint, as explained above.

Question 2 Pass the problem to a solver such as `cvxpy`, or write your own interior-point implementation to solve the linear program.

Question 3 Propose a way to use the solution returned by the solver to suggest a moving plan.

Question 4 Can we avoid putting the students' association offices next to that of the presidency throughout the renovation? If so, adapt your model to enforce this guarantee.

2 Quadratic programming model

Since we know the final allocation of the offices, we may want to try to match the final allocation as early as possible in the renovation process. For this purpose, we add a term to the objective that quantifies how the allocation differs from the final one, of the form

$$\lambda \sum_{p=0}^{5} \|\boldsymbol{z}^p - \boldsymbol{z}^F\|^2 = \lambda (\boldsymbol{z}^p - \boldsymbol{z}^F)^{\mathrm{T}} (\boldsymbol{z}^p - \boldsymbol{z}^F) \tag{1}$$

where $\boldsymbol{z}^p$ represents the vector of office allocations at phase p[3], $\boldsymbol{z}^F$ is the corresponding vector of final affectations, and $\lambda > 0$ is a weight put on this penalty.

Question 5 Form a quadratic programming model based on the model of Part 1 that encodes this penalization.

Question 6 Pass the problem to a solver such as `cvxpy`, or write your own interior-point implementation to solve the quadratic program.

[2] Other variables can -and likely should- be used in the model. In addition, students are free not to use the variables x_{ijp}.

[3] This notation assumes that this information is stored in a vector. The formula should be adapted to fit the variable representation chosen in Part 1.

Question 7 Compute the solution for $\lambda = 1$ and $\lambda = 100$. Do you observe changes compared to the solution obtained in Section 1?

Question 8 Replace the vector $\boldsymbol{z}^F$ in (1) by $\boldsymbol{z}^I$, representing the initial affectation of the offices, and compute the solution of the resulting quadratic program for $\lambda = 1$ and $\lambda = 100$. Comment on your findings, and compare them with that of Question 7.

3 Semidefinite programming relaxation

We now come back to the problem studied in Section 1. Recall that we provided a linear programming relaxation of the binary constraints that applied to every variable. The goal of this section is to explore another relaxation based on semidefinite programming. As seen in class, optimizing over a vector of binary variables [4] $\boldsymbol{u} \in \{-1, 1\}^N$ amounts to considering a matrix $\boldsymbol{U} \in \mathbb{R}^{N \times N}$ with constraints

$$\boldsymbol{U} = \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}, \quad \boldsymbol{U}_{ii} = 1 \ \forall i = 1, \ldots, N.$$

Indeed, the constraints on the diagonal elements of $\boldsymbol{U}$ will imply that the elements of $\boldsymbol{u}$ have values in $\{-1, 1\}$.

The problem is then relaxed into

$$\boldsymbol{U} \succeq \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}, \quad \boldsymbol{U}_{ii} = 1 \ \forall i = 1, \ldots, N. \tag{2}$$

As seen in class, the system (2) corresponds to constraints of a semidefinite program.

Question 9 Form a semidefinite programming model of the problem based on the relaxation idea (2).

Question 10 Compare the number of variables/constraints for the model of Question 9 and that needed by the linear programming formulation of Question 1. Does the SDP formulation allow you to use fewer variables/constraints?

Question 11 Pass the problem to a solver such as `cvxpy`, or write your own interior-point implementation to solve the semidefinite program.

Question 12 Incorporate the quadratic function of Section 3 into the semidefinite programming formulation.

[4] Note that any vector $\boldsymbol{v} \in \{0, 1\}^N$ can be converted into $\boldsymbol{u} \in \{-1, 1\}^N$ through the linear transformation $\boldsymbol{u} = 2\boldsymbol{v} - 1$.