Pacific Paradise - Vaccination Distribution Strategy

Benjamin Kruger - 46465522 , James Seymour - 46417585,MATH3202 Assignment 1

March 26, 2022

1 Part A - Report to Boss

1.1 Sets

$$
\begin{aligned}
WEEKS &= \{\text{WEEK}_0, \text{WEEK}_1, \cdots, \text{WEEK}_5\} \\
IDs &= \{\text{ID-A, ID-B, ID-C}\} \\
LVCs &= \{\text{LVC}_0, \text{LVC}_1, \cdots, \text{LVC}_7\} \\
CCDs &= \{\text{CCD}_0, \text{CCD}_1, \cdots, \text{CCD}_24\}
\end{aligned}
$$

These sets represent the weeks in the project, the different ID centers, the difference LVC centers and the different CCDs respectively. Note we enumerate from 0 as this is how the code is written.

1.2 Data

IDtoLVC: The distances from each ID to each LVC.

CCDPop: The number of people in each CCD.

CCDtoLVC: The distances from each CCD to LVC.

ID_import_cost_per_dose: The cost for a vaccine at each ID.

COMM2_ID_Max: The maximum number of vaccines which can be sent to any ID.

COMM2_LVC_Max: The maximum number of vaccines to be given out at an LVC.

COMM3_LVC_Max: The maximum number of weekly vaccines to be given out at an LVC.

COMM4_Delay: The increase in cost for delaying administration of a vaccine 1 week.

COMM5_Ratio_tolerence: The limit of the difference in the fraction of the population vaccinated at each CCD, for each week.

1.3 Design Variables

We first have two intermediate design variables. The reason for this is we need to calculate the cost to get to each LVC from each ID and CCD respectively from the given data. The variables created for this aren't dependent on the week until later communications, but it is still useful to include this dimension in the variables to have this implementation ready.

We define the sets of variables: **IDtoLVC_cost_per_dose** = 0.2× **IDtoLVC**. This is the cost to transport one vaccine from an ID to an LVC.
CCDtoLVC_cost_per_dose = CCDtoLVC. This is the cost for a single person from a CCD to receive a vaccine at an LVC.
We reassign these variables to **IDtoLVC_costs** and **CCDtoLVC_costs** respectively, so that we can index these variables with our elements of our sets 1.1 rather than integers. For example, (week, ID, LVC) in **IDtoLVC_costs** uniquely identifies the cost of sending a vaccine from a specific ID to a specific LVC in a specific week.

ID_Vars: The set of variables containing the number of vaccines sent to each ID in each week.

CCDtoLVC_Vars: The set of variables containing the number of people who travel from each CCD to each LVC to receive a vaccine each week.

IDtoLVC_Vars: The set of variables containing the number of vaccines sent from each ID to each LVC each week.

CCD_ratio_vaccinated_vars: The ratio of people vaccinated in a CCD to the total population, each week (defined in terms of CCDtoLVC_vars).

We make a note that in CCDtoLVC_Vars, some combinations of (week, CCD, LVC) will not have a mapped variable as the CCD is not adjacent to the LVC. In these cases, indexing this variable simply returns 0, as summing over 0 will have no effect in any constraints.

1.4 Objective Function for Communication 1-3

We want to minimise the total cost.

$$\text{Total Cost} = \text{total\_ID\_cost} + \text{total\_IDtoLVC\_cost} + \text{total\_CCDtoLVC\_cost}$$

$$\begin{aligned}
\text{total_ID_cost} &= \sum_{w \in WEEKS} \sum_{i \in IDs} \text{ID_Vars}_{w,i} \times \text{ID_import_cost_per_dose}_{w,i} \\
\text{total_IDtoLVC_cost} &= \sum_{w \in WEEKS} \sum_{i \in IDs} \sum_{l \in LVCs} \text{IDtoLVC_Vars}_{w,i,l} \times \text{IDtoLVC_cost_per_dose}_{w,i,l} \\
\text{total_CCDtoLVC_cost} &= \sum_{w \in WEEKS} \sum_{c \in CCDs} \sum_{l \in LVCs} \text{CCDtoLVC_cost_per_dose}_{w,c,l} \times \text{CCDtoLVC_Vars}_{w,c,l}
\end{aligned}$$

So, we want to minimise

$$\begin{aligned}\text{Total Cost} = &\sum_{w\in WEEKS}\sum_{i\in IDs} \text{ID\_Vars}_{w,i} \times \text{ID\_import\_cost\_per\_dose}_{w,i}\\ &+ \sum_{w\in WEEKS}\sum_{i\in IDs}\sum_{l\in LVCs} \text{IDtoLVC\_Vars}_{w,i,l} \times \text{IDtoLVC\_cost\_per\_dose}_{w,i,l}\\ &+ \sum_{w\in WEEKS}\sum_{c\in CCDs}\sum_{l\in LVCs} \text{CCDtoLVC\_cost\_per\_dose}_{w,c,l} \times \text{CCDtoLVC\_Vars}_{w,c,l}\end{aligned}$$

1.5 Communication 1 Constraints

$$\sum_{w\in WEEKS}\sum_{l\in LVCs} \text{CCDtoLVC\_Vars}_{w,c,l} \leq \text{CCD\_Pops}, \quad \text{for all } c \in CCDs \tag{1}$$

$$\sum_{w\in WEEKS}\sum_{c\in CCDs} \text{CCDtoLVC\_Vars}_{w,c,l} \leq \sum_{w\in WEEKS}\sum_{i\in IDs} \text{IDtoLVC\_Vars}_{w,i,l}, \quad \text{for all } l \in LVCs \tag{2}$$

$$\sum_{w\in WEEKS}\sum_{l\in LVC} \text{IDtoLVC\_Vars}_{w,i,l} \leq \sum_{w\in WEEKS} \text{ID\_Vars}_{w,i}, \quad \text{for all } i \in IDs \tag{3}$$

Constraint (1) ensures that the number of people going to any accessible LVC from the each CCD (across all the weeks) cannot exceed the number of people in that CCD.
Constraint (2) ensures that the number of people who attend this LVC cannot exceed the number of vaccines being sent to each LVC, over all weeks.
Constraint (3) ensures that the number of vaccines being sent out to each LVC cannot exceed the number of vaccines sent to this ID over all weeks.

1.6 Communication 2 Constraints

$$\sum_{w\in WEEKS} \text{ID\_Vars}_{w,i} \leq \text{COMM2\_ID\_Max}, \quad \text{for all } i \in IDs \tag{4}$$

$$\sum_{w\in WEEKS}\sum_{i\in IDs} \text{IDtoLVC\_Vars}_{w,i,l} \leq \text{COMM2\_LVC\_Max}, \quad \text{for all } l \in LVCs \tag{5}$$

Constraint (4) ensures that the number of vaccines at each ID does not exceed the maximum bound over all weeks.
Constraint (5) ensures that the number of vaccines administered at each LVC does not exceed the maximum bound over all weeks.

1.7 Communication 3 Constraints

In Communication 3, we are now introduced to the notion of weeks. Therefore, in addition to the constraint given, we must apply the previous constraints that were built into the problem, now on a weekly basis.

$$\sum_{c \in CCDs} \text{CCDtoLVC\_Vars}_{w,c,l} \leq \sum_{i \in IDs} \text{IDtoLVC\_Vars}_{w,i,l}, \quad \text{for all } w \in WEEKS, \text{ for all } l \in lVCs \tag{6}$$

$$\sum_{l \in LVC} \text{IDtoLVC\_Vars}_{w,i,l} \leq \text{ID\_Vars}_{w,i}, \quad \text{for all } w \in WEEKS, \text{ for all } i \in IDs \tag{7}$$

$$\sum_{i \in IDs} \text{IDtoLVC\_Vars}_{w,i,l} \leq \text{COMM3\_LVC\_Max}, \quad \text{for all } w \in WEEKS, \text{ for all } l \in lVCs \tag{8}$$

Constraint (6) ensures that the number of people who attend this LVC cannot exceed the number of vaccines being sent to each LVC, for each week.
Constraint (7) ensures that the number of vaccines being sent out to each LVC cannot exceed the number of vaccines sent to this ID for each week.
Constraint (8) limits the number of vaccines that can be sent to each LVC in each week.

1.8 Communication 4 Objective Function

In Communication 4, it will cost $10 extra per week to delay administration of a vaccine. We attribute this to the CCDtoLVC_vars, and therefore the total_CCDtoLVC_cost changes.

$$\text{CCDtoLVC\_costs\_with\_delay} = \sum_{w \in WEEKS} \sum_{c \in CCDs} \sum_{l \in LVCs} \text{CCDtoLVC\_costs}_{w,c,l} + (w) \times \text{COMM4\_delay}$$

Note the (w) being multiplied refers to the integer index of the week - in code we have to separate "WEEKw" for $w \in \{0, 1, 2, 3, 4, 5\}$. Therefore,

$$\text{total\_CCDtoLVC\_with\_delay} = \sum_{w \in WEEKS} \sum_{c \in CCDs} \sum_{l \in LVCs} \text{CCDtoLVC\_Vars} \times \text{CCDtoLVC\_with\_delay}.$$

This makes the total cost we wish to minimise then,

$$\begin{aligned} \text{Total Cost} = & \sum_{w \in WEEKS} \sum_{i \in IDs} \text{ID\_Vars}_{w,i} \times \text{ID\_import\_cost\_per\_dose}_{w,i} \\ & + \sum_{w \in WEEKS} \sum_{i \in IDs} \sum_{l \in LVCs} \text{IDtoLVC\_Vars}_{w,i,l} \times \text{IDtoLVC\_cost\_per\_dose}_{w,i,l} \\ & + \sum_{w \in WEEKS} \sum_{c \in CCDs} \sum_{l \in LVCs} \text{CCDtoLVC\_Vars}_{w,c,l} \times \text{CCDtoLVC\_with\_delay}_{w,c,l} \end{aligned}$$

1.9 Communication 5 Constraints

$$\max \left\{ \frac{\sum_{l \in LVCs} \text{CCDtoLVC\_Vars}_{w,c,l}}{\text{CCD\_Pops}_c} \right\} - \min \left\{ \frac{\sum_{l \in LVCs} \text{CCDtoLVC\_Vars}_{w,c,l}}{\text{CCD\_Pops}_c} \right\} \leq \text{COMM5\_Ratio\_Tolerence}$$

$$, \quad \text{for all } w \in WEEKS, c \in CCDs$$

This constraint ensures that the ratio of people getting vaccinated in a week to the population of the CCD differs by no more than COMM5_Ratio_Tolerance across the CCDs.

2 Part B - Report to Pacific Paradise

Communication 1:

Dear Pacific Paradise,

Thankyou for contacting us. Given the data and problem description, we have designed an optimisation of the vaccine distribution strategy to minimise your costs. In the solution, we need to send enough vaccines to each LVC so as to not run out while administering vaccinations. We see that the optimal solution has,

ID	Vaccines sent to ID
ID-A	84657
ID-B	0
ID-C	0

We see that all vaccines are sent to ID-A to be distributed. Indeed, for each CCD, the entire population goes to a single accessible LVC in the optimal solution. The total costs are $11220506 for the distribution.
We have attached specific details on how to execute this distribution strategy, and are happy to discuss this further.

Please don't hesitate to get in contact for more information or additional needs!

Kind regards,
Ben Kruger and James Seymour, Operations Research.

Communication 2:

Dear Pacific Paradise,

Given these new requirements, we have updated our solution. We have implemented the requirements that the number of vaccines at each ID doesn't exceed the maximum bound of 34000 and that the number of vaccines administered at each LVC does not exceed the maximum bound of 15000. This led us to a new a distribution of vaccines,

ID	Vaccines sent to ID
ID-A	34000
ID-B	34000
ID-C	16657

The total cost of this solution is $13567747.

Please don't hesitate to get in contact for more information or additional needs!

Kind regards,
Ben Kruger and James Seymour, Operations Research.

Communication 3:

Dear Pacific Paradise,

We have implemented a weekly system for the vaccine roll-out, and have subsequently bounded the maximum number of weekly vaccines administered at each LVC to 2000. We have compiled a table of the number of vaccines sent to each ID week by week below. The total cost of this strategy is $13602425.

Week	**ID**	**Vaccines sent to ID**
WEEK0	ID-A	6300
WEEK0	ID-B	6300
WEEK0	ID-C	4200
WEEK1	ID-A	6795
WEEK1	ID-B	6300
WEEK1	ID-C	2100
WEEK2	ID-A	7591
WEEK2	ID-B	6300
WEEK2	ID-C	2100
WEEK3	ID-A	4200
WEEK3	ID-B	6300
WEEK3	ID-C	4057
WEEK4	ID-A	5413
WEEK4	ID-B	4200
WEEK4	ID-C	2100
WEEK5	ID-A	3701
WEEK5	ID-B	4600
WEEK5	ID-C	2100

Please don't hesitate to get in contact for more information or additional needs!

Kind regards,
Ben Kruger and James Seymour, Operations Research.

Communication 4:

Dear Pacific Paradise,

Given the cost for delaying administration of vaccinations, we have updated our model to account for this extra cost. We have attributed this cost to each person receiving a vaccination at an LVC in a given week. As expected, we encourage that more people are vaccinated earlier on in the program, while still adhering to the constraints given in the previous communications.

This total cost for this new solution is $15374778, with the relevant distribution below,

Week	**ID**	**Vaccines sent to ID**
WEEK0	ID-A	7596
WEEK0	ID-B	6300
WEEK0	ID-C	2904
WEEK1	ID-A	6300
WEEK1	ID-B	6300
WEEK1	ID-C	4200
WEEK2	ID-A	6300
WEEK2	ID-B	6300
WEEK2	ID-C	4200
WEEK3	ID-A	6300
WEEK3	ID-B	6300
WEEK3	ID-C	4200
WEEK4	ID-A	5404
WEEK4	ID-B	6300
WEEK4	ID-C	2100
WEEK5	ID-A	2100
WEEK5	ID-B	1456
WEEK5	ID-C	97

Please don't hesitate to get in contact for more information or additional needs!

Kind regards,
Ben Kruger and James Seymour, Operations Research.

Communication 5:

Dear Pacific Paradise,

It is very understandable to want to make sure the distribution of vaccines is fair throughout the weeks. We have introduced a condition that the proportion of people who are vaccinated at each CCD in a given week is fair across all CCDs. This was achieved by bounding the minimum and maximum proportion who received a vaccine by 10% each week. The distribution of vaccines to IDs is included below. If you would like to talk more about the full solution and how to implement it, we would be more than happy to set up a meeting.

The total cost of this solution is $15382855

Week	**ID**	**Vaccines sent to ID**
WEEK0	ID-A	6860
WEEK0	ID-B	6300
WEEK0	ID-C	3639
WEEK1	ID-A	6300
WEEK1	ID-B	6300
WEEK1	ID-C	4200
WEEK2	ID-A	6300
WEEK2	ID-B	6300
WEEK2	ID-C	4200
WEEK3	ID-A	6300
WEEK3	ID-B	6300
WEEK3	ID-C	4200
WEEK4	ID-A	6139
WEEK4	ID-B	6300
WEEK4	ID-C	2100
WEEK5	ID-A	2100
WEEK5	ID-B	720
WEEK5	ID-C	97

Please don't hesitate to get in contact for more information or additional needs!

Kind regards,
Ben Kruger and James Seymour, Operations Research.

```python
ID_import_cost_per_dose = [109, 160, 168]
IDtoLVC_cost_per_dose = [[0.2 * distance for distance in ID] for ID in IDtoLVC]
CCDtoLVC_cost_per_dose = CCDtoLVC
```

Design Variables (intermediate)

```python
# Exports
```

Sets

```python
weeks = [f"WEEK{i}" for i in range(6)]
IDs = ["ID-A", "ID-B", "ID-C"]
LVCs = [f"LVC{i}" for i in range(8)]
CCDs = [f"CCD{i}" for i in range(25)]
```

Data

```python
ID_costs = { (week, ID): ID_import_cost_per_dose[ID_i]  for week in weeks for ID_i, ID in enumerate(IDs) }
IDtoLVC_costs = { (week, ID, LVC): IDtoLVC_cost_per_dose[ID_i][LVC_i] for week in weeks for LVC_i, LVC in enumerate(LVCs) for ID_i, ID in enumerate(IDs) }
CCDtoLVC_costs = { (week, CCD, LVC): CCDtoLVC_cost_per_dose[CCD_i][LVC_i] for week in weeks for LVC_i, LVC in enumerate(LVCs) for CCD_i, CCD in enumerate(CCDs) }

CCD_pops = { CCD: CCDPop[CCD_i] for CCD_i, CCD in enumerate(CCDs) }

COMM2_ID_MAX = 34000
COMM2_LVC_MAX = 15000
COMM3_LVC_MAX = 2100
COMM4_DELAY_COST = 10
COMM5_RATIO_TOLERANCE = 0.1
```

```python
from typing import Dict
from gurobipy import Model, GRB, quicksum, max_, min_
from itertools import product, chain
from vaccine import (
    # Var definitions
    weeks,
    IDs,
    LVCs,
    CCDs,

    # Cost definitions
    ID_costs,
    IDtoLVC_costs,
    CCDtoLVC_costs,

    # Constraint definitions
    CCD_pops,

    # Communication constraint constants
    COMM2_ID_MAX,
    COMM2_LVC_MAX,
    COMM3_LVC_MAX,
    COMM4_DELAY_COST,
    COMM5_RATIO_TOLERANCE,
)
from helpers import map, Gurobi_Var_Dict, Gurobi_Constraint_Gen

## Setup model, variables and objective

model = Model()

variables: Dict[str, Gurobi_Var_Dict] = {
    "ID": model.addVars(weeks, IDs),
    "IDtoLVC": model.addVars(weeks, IDs, LVCs),
    "CCDtoLVC": model.addVars(filter(lambda var_key: CCDtoLVC_costs[var_key] > 0, product(weeks, CCDs, LVCs))), # Use sparse variable creation here, as a 0 cost CCDtoLVC is inaccessible
}

total_ID_cost = quicksum(ID_costs[var_key] * gurobi_var for var_key, gurobi_var in variables["ID"].items())
total_IDtoLVC_cost = quicksum(IDtoLVC_costs[var_key] * gurobi_var for var_key, gurobi_var in variables["IDtoLVC"].items())
total_CCDtoLVC_cost = quicksum(CCDtoLVC_costs[var_key] * gurobi_var for var_key, gurobi_var in variables["CCDtoLVC"].items())

total_cost = total_ID_cost + total_IDtoLVC_cost + total_CCDtoLVC_cost

model.setObjective(total_cost, GRB.MINIMIZE)
```

Objective func. #1

Communication 1

Contraint definitions

```python
def generate_base_CCD_constraints(CCDtoLVC_vars: Gurobi_Var_Dict) -> Gurobi_Constraint_Gen:
    """
        Main row-based constraint for the CCDtoLVC table.\n
        For each CCD, the relevant LVCs that can be accessed from this CCD must supply a total number of vaccines that is greater than or equal to population of that CCD.
    """

    for CCD in CCDs:
        total_vaccine_supply_for_accessible_LVCs = quicksum(CCDtoLVC_vars.get((CCD, LVC), 0) for LVC in LVCs)
        CCD_population = CCD_pops[CCD]
        yield total_vaccine_supply_for_accessible_LVCs >= CCD_population

def generate_base_LVC_constraints(CCDtoLVC_vars: Gurobi_Var_Dict, IDtoLVC_vars: Gurobi_Var_Dict) -> Gurobi_Constraint_Gen:
    """
        Main col-based constraint between the CCDtoLVC and IDtoLVC tables.\n
        For each LVC, the number of people who attend this LVC must be less than or equal to the number of vaccines supplied to this LVC.
    """

    for LVC in LVCs:
        total_LVC_attendance = quicksum(CCDtoLVC_vars.get((CCD, LVC), 0) for CCD in CCDs)
        total_vaccines_supplied_to_LVC = quicksum(IDtoLVC_vars[ID, LVC] for ID in IDs)
        yield total_LVC_attendance <= total_vaccines_supplied_to_LVC

def generate_base_ID_constraints(ID_vars: Gurobi_Var_Dict, IDtoLVC_vars: Gurobi_Var_Dict) -> Gurobi_Constraint_Gen:
    """
        Main row-based constraint between the ID and IDtoLVC tables.\n
        For each ID, the number of vaccines that this ID supplies to each of its LVCs is less than or equal to its total vaccine supply.
    """

    for ID in IDs:
        total_vaccines_supplied_to_ID = ID_vars[ID]
        total_vaccines_supplied_to_LVCs = quicksum(IDtoLVC_vars[ID, LVC] for LVC in LVCs)
        yield total_vaccines_supplied_to_LVCs <= total_vaccines_supplied_to_ID
```

Constraints
Comm 1

```python
# For Communication 1 and 2, we are agnostic of which week we are in because we don't have a notion of weeks yet.
# Therefore, instead of writing all of the code twice, we sum the gurobi vars over weeks,
# effectively removing the week dimension by grouping by it and aggregating with quicksum()

week_agnostic_IDs = {ID: quicksum(variables["ID"][week, ID] for week in weeks) for ID in IDs}
week_agnostic_IDtoLVCs = { (ID, LVC): quicksum(variables["IDtoLVC"][week, ID, LVC] for week in weeks) for ID, LVC in product(IDs, LVCs) }
week_agnostic_CCDtoLVCs = { (CCD, LVC): quicksum(variables["CCDtoLVC"].get((week, CCD, LVC), 0) for week in weeks) for CCD, LVC in product(CCDs, LVCs) }
```

```python
### Apply contraints

map(
    model.addConstr, # Using addConstr instead of addConstrs because gurobi complains about a missing index
    chain(
        generate_base_ID_constraints(week_agnostic_IDs, week_agnostic_IDtoLVCs),
        generate_base_LVC_constraints(week_agnostic_CCDtoLVCs, week_agnostic_IDtoLVCs),
        generate_base_CCD_constraints(week_agnostic_CCDtoLVCs),
    )
)

model.optimize()
```

Constraints
Comm 2.

```python
## Communication 2

### Constraint definitions

def generate_maximum_ID_constraint(ID_vars: Gurobi_Var_Dict, maximum: int) -> Gurobi_Constraint_Gen:
    """
        For each ID, limit the total vaccine supply to that ID by the given maximum
    """

    for ID in IDs:
        total_vaccines_supplied_to_ID = ID_vars[ID]
        yield total_vaccines_supplied_to_ID <= maximum

def generate_maximum_LVC_constraints(IDtoLVC_vars: Gurobi_Var_Dict, maximum: int) -> Gurobi_Constraint_Gen:
    """
        For each LVC, limit the total vaccine supply to that LVC by the given maximum
    """

    for LVC in LVCs:
        total_vaccines_supplied_to_LVC = quicksum(IDtoLVC_vars[ID, LVC] for ID in IDs)
        yield total_vaccines_supplied_to_LVC <= maximum

def generate_maximum_CCD_constraints(CCDtoLVC_vars: Gurobi_Var_Dict, maximum: int) -> Gurobi_Constraint_Gen:
    """
        For each LVC, limit the total attendance at the LVC by the given maximum
    """

    for LVC in LVCs:
        total_attendance_at_CCD = quicksum(CCDtoLVC_vars[CCD, LVC] for CCD in CCDs)
        yield total_attendance_at_CCD <= maximum
### Apply constraints

map(
    model.addConstr, # Using addConstr instead of addConstrs because gurobi complains about a missing index
    chain(
        generate_maximum_ID_constraint(week_agnostic_IDs, maximum=COMM2_ID_MAX),
        generate_maximum_LVC_constraints(week_agnostic_IDtoLVCs, maximum=COMM2_LVC_MAX),
    )
)

model.optimize()
```

Communication 3

Constraint definitions

```python
def generate_maximum_LVC_weekly_constraints(IDtoLVC_vars: Gurobi_Var_Dict, maximum: int) -> Gurobi_Constraint_Gen:
    """
        For each week, limit the number of weekly doses supplied to this LVC by the given maximum, using the IDtoLVC constraint generator above
    """

    for week in weeks:
        this_weeks_IDtoLVC_vars = { (ID, LVC): IDtoLVC_vars[week, ID, LVC] for ID, LVC in product(IDs, LVCs) }
        yield from generate_maximum_LVC_constraints(this_weeks_IDtoLVC_vars, maximum)

def generate_maximum_CCD_weekly_constraints(CCDtoLVC_vars: Gurobi_Var_Dict, maximum: int) -> Gurobi_Constraint_Gen:
    """
        For each week, limit the attendance to this CCD by the given maximum, using the CCDtoLVC constraint generator above
    """

    for week in weeks:
        this_weeks_CCDtoLVC_vars = { (CCD, LVC): CCDtoLVC_vars.get((week, CCD, LVC), 0) for CCD, LVC in product(CCDs, LVCs) }
        yield from generate_maximum_CCD_constraints(this_weeks_CCDtoLVC_vars, maximum)
```

Constraints
comm 3

Apply constraints

```python
map(
    model.addConstr, # Using addConstr instead of addConstrs because gurobi complains about a missing index
    chain(
        generate_maximum_LVC_weekly_constraints(variables["IDtoLVC"], maximum=COMM3_LVC_MAX),
        generate_maximum_CCD_weekly_constraints(variables["CCDtoLVC"], maximum=COMM3_LVC_MAX),
    )
)

model.optimize()
```

Communication 4

objective func #2.

Cost definitions

```python
CCDtoLVC_costs_with_delay = { (week, CCD, LVC): cost + COMM4_DELAY_COST * int(week[-1]) for (week, CCD, LVC), cost in CCDtoLVC_costs.items() } # Very hacky ik

total_CCDtoLVC_cost_with_delay = quicksum(CCDtoLVC_costs_with_delay[var_key] * gurobi_var for var_key, gurobi_var in variables["CCDtoLVC"].items())

updated_total_cost = total_ID_cost + total_IDtoLVC_cost + total_CCDtoLVC_cost_with_delay

model.setObjective(updated_total_cost, GRB.MINIMIZE)

model.optimize()
```

Design Variables

Constraints
Comm 5

```
## Communication 5

### Variable definitions

ratios = model.addVars(weeks, CCDs)
min_ratio_vars = model.addVars(weeks)
max_ratio_vars = model.addVars(weeks)

### Constraint definitions

def generate_maximum_distribution_constraints(CCDtoLVC_vars: Gurobi_Var_Dict, max_ratio_tolerance: float) -> Gurobi_Constraint_Gen:
    for week in weeks:

        yield from ((ratios[week, CCD] == quicksum(CCDtoLVC_vars.get((week, CCD, LVC), 0) for LVC in LVCs) / CCD_pops[CCD]) for CCD in CCDs)

        min_this_week, max_this_week = min_(ratios[week, CCD] for CCD in CCDs), max_(ratios[week, CCD] for CCD in CCDs)

        yield from (
            min_ratio_vars[week] == min_this_week,
            max_ratio_vars[week] == max_this_week,
            max_ratio_vars[week] - min_ratio_vars[week] <= max_ratio_tolerance,
        )

### Apply constraints

map(
    model.addConstr, # Using addConstr instead of addConstrs because gurobi complains about a missing index
    generate_maximum_distribution_constraints(variables["CCDtoLVC"], max_ratio_tolerance=COMM5_RATIO_TOLERANCE)
)

model.optimize()
```