IEOR4004 Project 1

Eliminating Child Care Deserts in New York State through Optimization

Jerry Shen, Mark Ma, Iris Wei, Seana Flood
zs2693, km4054, zw3072, sf3266
Nov 1, 2024

1 Abstract

In this paper, we seek to mitigate the prevalence of childcare deserts with the use of mixed integer linear programming. Since childcare demand and resources, among other considerations, can vary drastically between regions, the scope of this project is limited to childcare within the state of New York. To establish a baseline, we first consider ideal conditions in modeling childcare demands. Subsequently, we address additional factors that are relevant to modeling realistic conditions in the state of New York, such as cost constraints, distance limitations, and fairness measures. While considering these factors, the goal is to minimize the cost required to allocate child care resources efficiently. With the current constraints, we observe the ability to eliminate child care deserts while maintaining demand and distance restrictions, but not while maintaining the fairness measure.

2 Introduction

Childcare is an essential resource for many parents, yet it often proves to be a source of concern. In some cases, childcare can be cost prohibitive; in other cases, demand simply exceeds the supply leaving parents without a sure means of childcare. In turn, parents often must allocate a substantial portion of their salaries to childcare or alter their working hours. Therefore, it is important that child care resources are not only in place and available for parents, but managed efficiently. Most obviously, addressing this concern will start by benefiting the children, but the benefits will prove to extend beyond the children to both the parents and the state of New York as a whole.

3 Background

The issues surrounding childcare have been permeating for several years but were heightened during the pandemic leaving parents demanding action. When childcare services closed during the pandemic, the country quickly realized how dependent parents, and therefore businesses, are on childcare services. This paper focuses on the shortage of childcare facilities and spots to support the needs of the state, however the issue of childcare does not lie entirely in its lack of physical infrastructure. Defined as the "workforce behind the workforce," childcare workers are critical in the childcare ecosystem, but also in supporting New York's economic ecosystem as well (Scheid, 2021). Parents rely on childcare services in facilitating their ability to perform their daily work obligations.

While the availability of childcare workers is not considered in this analysis, these two issues do not live in isolation. In order to eliminate demand deserts, the facilities and spots must first be available, but there must also be a sufficient workforce available to support these facilities. By minimizing the expansion costs, the budget will be better suited to fund the remaining operations required of childcare. Thus, this analysis is just one step of a larger series that must be performed to properly address the comprehensive needs of childcare, but it is an essential step in getting started.

4 Methods

Before the model could be established, the datasets required cleansing. Given the large datasets, cleansing is essential not only to facilitate efficient modeling, but to simplify the computational steps required throughout the analysis.

Since the model is predicated on certain parameters such as population, child care deserts, and demand area, it was necessary to first define these parameters. Much of the analysis relies on the age ranges of children and thus involved the manipulation of the data that was collected and provided for the age ranges of 0-5 years, 5-9 years, and 10-14 years. As childcare is only relevant between the ages of 2 weeks and 12 years, the total population encompassing this range was estimated as a sum of the younger ages and a proportion of the population of 10-14 years:

$$\text{population}[0-5 \text{ years}] + \text{population}[5-9 \text{ years}] + \frac{3}{5}\text{population}[10-14 \text{ years}] \quad (1)$$

Summary statistics were also performed in order to calculate parameters such as average salary and employment rate. These parameters were subsequently used in determining whether or not a region, identified by zip code for the purposes of this analysis, is a high demand area or a normal demand area. A high-demand area is defined as one in which at least 60% of parents are employed or in which the average income is at most \$60,000. If an area does not meet either of these criteria, it is classified as a normal demand area. Outlining and defining these parameters ahead of time simplified the modeling process greatly.

The specifics surrounding the data modeling for each problem are similar throughout, but feature slight modifications depending on the problem at hand. The originating model and its modifications are detailed below.

Part 1:

To begin, we consider the demand and costs involved in allocating childcare resources. The number of childcare slots required is determined by demand type. In high-demand areas, the available slots must be more than half of the total population of children between the ages of 2 weeks to 12 years; in normal demand areas, this proportion decreases to one-third. For context, a childcare desert is defined as an area in which this minimum demand is not met.

Regardless of demand type, an additional condition requires that the number of slots available for children between the ages of 0-5 must be at least two-thirds of the total population of children between the ages of 0-5. Children between the ages of 0-5 also incur an additional \$100 per slot added. These requirements reflect the higher demand and cost typically associated with toddlers.

In order to meet demand, the state of New York proposes a combination of two approaches: expanding existing facilities and building new facilities. If a facility has an existing capacity larger than 500 slots, it cannot be expanded further. If it has an existing capacity under 500 slots, it can be expanded up to 20% or until capacity reaches 500 slots. It is also

worth noting that newly built facilities must remain at their preset capacity; in other words, newly built facilities cannot be expanded. Each approach has its own cost specifications.

Mathematically, we can represent this model and its considerations as follows:

Indices

- $t \in$ facility type $= \{small, medium, large\}$: potential facility build types
- $i \in$ facilities: existing facilities in New York State
- $j \in$ zip codes: zip codes in New York State

Parameters

- average salary$_j \in \mathbb{Q}^+$: average salary for zip code j
- employment rate$_j \in \mathbb{Q}^+$: average employment rate for zip code j
- baseline cost$_i \in \mathbb{Z}^+$: baseline cost for expanding facility i
- capacity based cost$_i \in \mathbb{Z}^+$: cost per spot added at facility i
- building cost$_t \in \mathbb{Z}^+$: cost to build a new facility of type t
- facility capacity$_i \in \mathbb{Z}^+$: current capacity at facility i
- population$_j \in \mathbb{Z}^+$: population of age ranges in zip code j

Decision Variables

- $x_{1,i} \in \mathbb{N}$: number of expanded spots at facility i
- $x_{2,i} \in \mathbb{N}$: number of expanded spots for children under age 5 at facility i
- $y_{t,j} \in \mathbb{N}$: number of newly built facilities of size t (small, medium, or large) in zip code j

Objective Function

Minimize the cost of funding expansion and creating new facilities to meet anticipated child care demand in the state of New York.

$$\text{Minimize } Z = Z_{\text{Expansion Cost}} + Z_{\text{Building Cost}}$$

where

$$Z_{\text{Expansion Cost}} = (200 \times x_{1,i}) + (100 \times x_{2,i}) + (20,000 \times \frac{x_{1,i}}{\text{childcare capacity}_i}) \quad \forall i \in I$$

$$Z_{\text{Building Cost}} = (65,000 \times y_{1,j}) + (95,000 \times y_{2,j}) + (115,000 \times y_{3,j}) \quad \forall j \in J$$

Constraints

- Expansion Constraint:
  - $x_{1,i} \leq 0.2 \times \text{facility capacity}_i \quad \forall i \in I$
  - $x_{2,i} - x_{1,i} \leq 0 \quad \forall i \in I$
  - $\text{facility capacity}_i + x_{1,i} \leq \max\{\text{facility capacity}_i, 500\} \quad \forall i \in I$
- High Demand Area:

  Let $l \in L_j$ represent the facilities within each zip code j.

$$\text{population}[2\text{w} - 12 \text{ years}]_j + (100 \times y_{1,j}) + (200 \times y_{2,j}) + (400 \times y_{3,j}) + \sum_{l \in L_j} x_{1,l} \geq ([\frac{1}{2} \times \text{population}[2\text{w} - 12 \text{ years}]_j]) + 1 \quad \forall j \in J \quad (2)$$

- Normal Demand Area:

  Let $l \in L_j$ represent the facilities within each zip code j.

$$\text{population}[2\text{w} - 12 \text{ years}]_j + (100 \times y_{1,j}) + (200 \times y_{2,j}) + (400 \times y_{3,j}) + \sum_{l \in L_j} x_{1,l} \geq ([\frac{1}{3} \times \text{population}[2\text{w} - 12 \text{ years}]_j]) + 1 \quad \forall j \in J \quad (3)$$

- Age Specific Constraint:

  Let $l \in L_j$ represent the facilities within each zip code j.

$$\text{population}[2\text{w} - 5 \text{ years}]_j + (50 \times y_{1,j}) + (100 \times y_{2,j}) + (200 \times y_{3,j}) + \sum_{l \in L_j} x_{2,l} \geq \frac{2}{3} \times \text{population}[0 - 5 \text{ years}]_j \quad \forall j \in J \quad (4)$$

- Non-Negativity Constraint:
  - $x_{1,i} \in \mathbb{N} \quad \forall i \in I$
  - $x_{2,i} \in \mathbb{N} \quad \forall i \in I$
  - $y_{t,j} \in \mathbb{N} \quad \forall t \in T,\ j \in J$

Part 2:

With the baseline model established, it can now be expanded to include more realistic cost approximations and distance limitations. Note, the model remains as defined in Part 1, unless specifically modified below. In Part 1, there is a baseline cost of \$20,000 and a consistent cost per added slot. Now, to model the natural tendency of rising expansion costs as the expansion

proportions increase, expansion costs are modeled as a piecewise function in which a different capacity cost is incurred depending on the expansion rate.

The baseline cost of $20,000 remains, but now the capacity based cost is $200, $400, or $1000 for expansion rates of 0-10%, 10-15%, and 15-20%. The $100 premium also remains in place for slots allocated for children under the age of 5 years. To represent the varying expansion costs, the existing decision variables are altered slightly and additional binary variables are added to represent whether or not each expansion increment is exceeded. More specifically, we denote $a_{1,i}$ to be 1 if the expansion rate reaches 10% and 0 if not. Likewise $a_{2,i}$ is defined as 1 if the expansion rate reaches 15% and 0 if not.

The state of New York also recommends a distance limitation of 0.06 miles to prevent facilities from being overcrowded and to ensure facilities are fairly distributed. With distance now being a consideration, we update index j to represent each potential location instead of an entire zip code and decision variable Y to be a binary variable representing whether or not a facility will be built at potential location j. Note, existing facilities are not subject to this constraint, meaning if two existing facilities are placed within 0.06 miles of each other, they may remain as is.

With 14,756 existing facilities and 102,300 potential facility locations to consider, this constraint is introduced using a two part approach in order to mitigate computational complexity. For all comparisons, the haversine Python module is used to calculate the distance between two locations given their latitudes and longitudes. First, each potential location is assessed in relation to existing facilities. If any of the potential locations are within 0.06 miles of an existing facility, a constraint is added that prevents a facility from being built at that potential location. We represent this as $y_{1,l} + y_{2,l} + y_{3,l} = 0$ at potential location, l. Then, the potential locations are compared against each other. If the distance between any two potential locations is less than 0.06, a constraint is added that allows a facility to be built in either location, but not both. Given potential locations k and l, we represent this constraint as $y_{1,l} + y_{2,l} + y_{3,l} + y_{1,k} + y_{2,k} + y_{3,k} = 1$

The updates to the model are summarized below.

Indices

- $j \in$ potential locations: potential locations to build a new facility

Decision Variables

- $x_{1,i} \in \mathbb{N}$: number of expanded spots at facility i in which the expansion is between 0% and 10%
- $x_{2,i} \in \mathbb{N}$: number of expanded spots at facility i in which the expansion is between 10% and 15%
- $x_{3,i} \in \mathbb{N}$: number of expanded spots at facility i in which the expansion is between 15% and 20%
- $x_{4,i} \in \mathbb{N}$: number of expanded spots for children under age 5 at facility i
- $y_{t,j}$ is a binary variable defined as:

$$y_{t,j} = \begin{cases} 1 & \text{if a facility of size } t \text{ is built at potential location } j \\ 0 & \text{otherwise} \end{cases}$$

- $a_{1,i}$ is a binary variable defined as:

$$a_{1,i} = \begin{cases} 1 & \text{if expansion exceeds 10\%} \\ 0 & \text{otherwise} \end{cases}$$

- $a_{2,i}$ is a binary variable defined as:

$$a_{2,i} = \begin{cases} 1 & \text{if expansion exceeds 15\%} \\ 0 & \text{otherwise} \end{cases}$$

Constraints

- Expansion Rate
  - $x_{1,i} + x_{2,i} + x_{3,i} \leq 0.2 \times \text{facility capacity}_i \quad \forall i \in I$
  - $x_{4,i} - x_{3,i} - x_{2,i} - x_{1,i} \leq 0 \quad \forall i \in I$
  - $\text{facility capacity}_i + x_{1,i} + x_{2,i} + x_{3,i} \leq \max\{\text{facility capacity}_i, 500\} \quad \forall i \in I$
- Expansion Cost
  - 0-10%:

$$[(a_{1,i} \times 0.1 \times \text{facility capacity}_i)] \leq x_{1,i} \leq [(0.1 \times \text{facility capacity}_i)]$$

  - 10-15%:

$$a_{2,i}([0.15 \times \text{facility capacity}_i] - [0.1 \times \text{facility capacity}_i]) \leq x_{2,i}$$
$$x_{2,i} \leq a_{1,i}([0.15 \times \text{facility capacity}_i] - [0.1 \times \text{facility capacity}_i])$$

  - 15-20%:

$$0 \leq x_{3,i} \leq a_{2,i}([0.2 \times \text{facility capacity}_i] - [0.15 \times \text{facility capacity}_i])$$

- High Demand Area

  Let $l \in L_m$ represent the facilities within each zip code m.

$$\text{population}[2\text{w} - 12 \text{ years}]_j + (100 \times y_{1,j}) + (200 \times y_{2,j}) + (400 \times y_{3,j}) + \sum_{l \in L_m} (x_{1,l} + x_{2,l} + x_{3,l}) \geq [\frac{1}{2} \times \text{population}[2\text{w} - 12 \text{ years}]_j] + 1 \quad \forall j \in J \quad (5)$$

- Normal Demand Area

  Let $l \in L_m$ represent the facilities within each zip code m.

$$\text{population}[2\text{w} - 12 \text{ years}]_j + (100 \times y_{1,j}) + (200 \times y_{2,j}) + (400 \times y_{3,j}) + \sum_{l \in L_m} (x_{1,l} + x_{2,l} + x_{3,l}) \geq [\frac{1}{3} \times \text{population}[2\text{w} - 12 \text{ years}]_j] + 1 \quad \forall j \in J \quad (6)$$

- Age Specific Constraint

  Let $l \in L_m$ represent the facilities within each zip code m.

$$\text{population}[2\text{w} - 5 \text{ years}]_j + (50 \times y_{1,j}) + (100 \times y_{2,j}) + (200 \times y_{3,j}) + \sum_{l \in L_m} x_{4,l} \geq \frac{2}{3} \times \text{population}[0 - 5 \text{ years}]_j \quad \forall j \in J \quad (7)$$

- Distance

  Let $k \in K_m$ represent the existing facilities within each zip code m. Let $l \in L_m$ represent the potential facilities within each zip code m.

  - Distance between Existing Locations and Potential Locations
    If haversine$((latitude_k, longitude_k), (latitude_l, longitude_l)) \leq 0.06$, then we impose:

$$y_{1,l} + y_{2,l} + y_{3,l} = 0 \quad \forall k \in K, \forall l \in L$$

  - Distance between two Potential Locations
    If haversine$((latitude_l, longitude_l), (latitude_{l+1}, longitude_{l+1})) \leq 0.06$, then we impose:

$$y_{1,l} + y_{2,l} + y_{3,l} + y_{1,l+1} + y_{2,l+1} + y_{3,l+1} = 1 \quad \forall l \in L$$

Objective Function

Minimize the cost of funding expansion and creating new facilities to meet anticipated child care demand in the state of New York.

$$\text{Minimize } Z = Z_{\text{Expansion Cost}} + Z_{\text{Building Cost}}$$

where

$$Z_{\text{Expansion Cost}} = (200 \times x_{1,i}) + (400 \times x_{2,i}) + (1,000 \times x_{3,i}) + (100 \times x_{4,i}) + (20,000 \times \frac{x_{1,i} + x_{2,i} + x_{3,i}}{\text{childcare capacity}_i}) \quad \forall i \in I$$

$$Z_{\text{Building Cost}} = (65,000 \times y_{1,j}) + (95,000 \times y_{2,j}) + (115,000 \times y_{3,j}) \quad \forall j \in J$$

Part 3:

While the distance limitation imposed in Part 2 ensures that facilities exist between regions, it does not take into account that the spots available at facilities are fairly distributed. For example, two facilities could be situated at least 0.06 miles apart. Yet, serving the same demand, one facility might accommodate 1,000 slots, while the other offers just 100. This disparity is addressed in Part 3 of the analysis.

Ideally, the state of New York would like to ensure that the difference in the proportion of slots available between any two facilities does not exceed 0.1. To model this constraint, we introduce two continuous variables that can take any value larger than or equal to 0 that represent the minimum ratio and the maximum ratio. After calculating the slot ratio for each zip code, we add three

constraints to ensure the difference proportion of slots does not exceed 0.1. It is worth noting, this approach reduces the operation from complexity $O(n^2) \to O(n)$, which is crucial when dealing with large datasets.

- slots ratio $\geq$ minimum ratio
- slots ratio $\leq$ maximum ratio
- maximum ratio $-$ minimum ratio ≤ 0.1

Consistent with previous analysis, children under the age of 5 are given twice the weighting of children over the age of 5 to account for the additional needs required of this age group. Given a budget of 1 billion dollars, the objective is no longer to minimize the cost allocated, but rather to maximize the social coverage on a state-level. The updates to the model are outlined below.

Indices

- $i \in$ facilities: existing facilities in New York State
- $j \in$ potential locations: potential locations to build a new facility
- $k \in$ zip codes: zip codes in New York State

Additional Decision Variables

(Although not frequent, the rates may exceed 1, which means in that region the number of facilities is larger than the population)

- minimum ratio$_k \geq 0$: minimum ratio for zip code k
- maximum ratio$_k \geq 0$: maximum ratio for zip code k

Additional Parameter

Let $l \in L_k$ represent the facilities within each zip code k.

- slots ratio$_k = \frac{\sum_{l \in L_k}(x_{1,l}+x_{2,l}+x_{3,l})+\sum_{l \in L_k}(100\times y_{1,l}+200\times y_{2,l}+400\times y_{3,l})}{\text{population}[2w-12 \text{ years}]_k} \quad \forall k \in K$

Constraints

- Fairness Constraints
  - slots ratio$_k \geq$ minimum ratio$_k \quad \forall k \in K$
  - slots ratio$_k \leq$ maximum ratio$_k \quad \forall k \in K$
  - maximum ratio$_k -$ minimum ratio$_k \leq 0.1 \quad \forall k \in K$
- Budget Constraint

$$\begin{aligned}(65,000 \times y_{1,j} + 95,000 \times y_{2,j} + 115,000 \times y_{3,j}) \quad & \forall j \in J \\ + ((200 \times x_{1,i}) + (400 \times x_{2,i}) + (1,000 \times x_{3,i}) + (100 \times x_{4,i}) & \\ + \left(20,000 \times \frac{x_{1,i}+x_{2,i}+x_{3,i}}{\text{childcare capacity}_i}\right)) \quad & \forall i \in I \leq 1,000,000,000 \end{aligned} \tag{8}$$

Objective Function

Maximize the social coverage index of child care facilities in New York.

$$\text{Maximize Z} = 2Z_{\text{Social Coverage[2w - 5 years]}} + Z_{\text{Social Coverage[2w - 12 years]}}$$

where

$$Z_{\text{SC[2w-5y]}} = \frac{\sum_{i \in I} \text{facility capacity}[2w - 5y]_i + \sum_{i \in I} x_{4,i} + \sum_{j \in J} 50y_{1,j} + 100y_{2,j} + 200y_{3,j}}{\text{population[2w - 5y]}}$$

$$Z_{\text{SC[2w-12y]}} = \frac{\sum_{i \in I} \text{facility capacity}[2w - 12y]_i + \sum_{i \in I} x_{1,i} + x_{2,i} + x_{3,i} + \sum_{j \in J} 100y_{1,j} + 200y_{2,j} + 400y_{3,j}}{\text{population[2w - 12y]}}$$

5 Results

Idealistic Conditions

Considering simply the budgetary constraints and demand, our analysis proposes the expansion of 1,951 out of the total 14,755 facilities and the creation of 2,687 new facilities. Of the 2,687 new facilities, 227 are small, 164 are medium, and 2,296 are large facilities. This approach costs $315,788,317.

Capacity Based Cost Expansion and Distance Limitations

With more realistic expansion costs and distance limitations, our analysis proposes the expansion of 2,239 existing facilities and the creation of 2,756 new facilities. Of the 2,756 new facilities, 213 are small, 167 are medium, and 2,376 are large. This approach costs $320,269,337.

Fairness Considerations

Considering the social coverage index and a restriction of no more than a 0.1 difference in indexes between two regions, the problem becomes infeasible due to the extreme variation in proportions among regions. For example, there are zip codes in which the slots-to-population ratio reaches as high as 10. Even when accounting for potential expansions or the creation of new facilities, achieving a ratio within 0.1 is impossible. In this case, the budget constraint is not the limiting factor; rather, it is the restrictive requirement on the difference in social coverage ratios that poses the challenge.

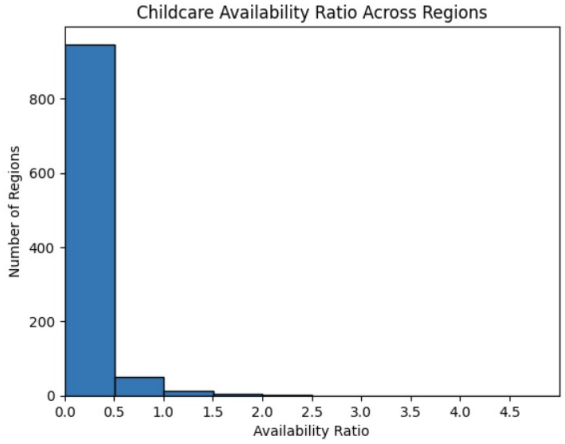
Overall Analysis

From examining the summary metrics from Parts 1 and 2, a few trends can be identified. To start, 85.4% and 86.2% of the newly built facilities are of large capacity. Since the objective in both cases was to minimize the cost, this aligns with our expectation as the cost per slot drops drastically as the facility size increases. While building new facilities involves a longer timeline

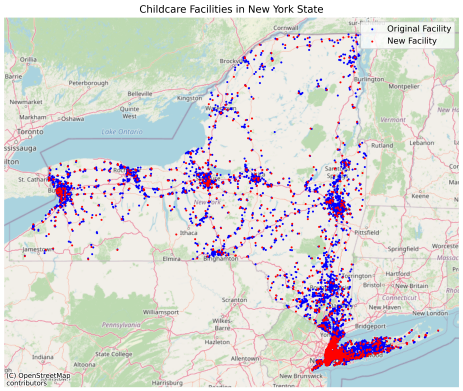
before resources become available compared to expanding existing ones, it is more economically sensible for New York to invest in large facilities to efficiently meet slot demand.

Also of note is that a significant proportion of expanded slots are allocated to children under the age of 5 years old. In both Parts 1 and 2, an overwhelming 94.1% and 94.8% of the expanded slots cater exclusively to this age group. This indicates that existing resources and the original allocation model may have overlooked the higher demand for childcare among the younger age group.

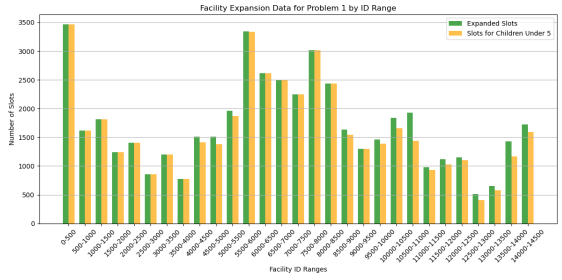
To analyze the recommended new locations for facility construction, we plotted the existing locations and the new locations. Varying colors were used to differentiate the existing and new locations and to facilitate ease of analysis. Examining the map of New York, it is evident that the majority of new facilities built are in areas in which facilities already exist. Conceptually, this makes sense as the original facilities were presumably established in response to population demand. This is further emphasized by the significant concentration of facilities placed in the metropolitan area. This suggests that there is a greater need for the reinforcement of existing resources as opposed to the redistribution and placement of entirely new resources.



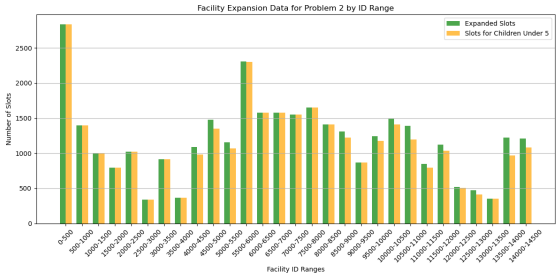
(a) Childcare Availability Ratio Across Regions



(b) Childcare Facilities



(c) Facility Expansion Problem 1



(d) Facility Expansion Problem 2

6 Conclusions

This project works to offer solutions for the issue of childcare deserts in the state of New York. By assessing demand, existing resources, and distance limitations, we were able to optimize the current childcare landscape and formulate recommendations for expanding resources efficiently. Based on our analysis, our recommendations focus on increasing the capacity for children under the age of 5 at existing facilities and building primarily new large facilities. Implementing our proposals will assist the state of New York in eliminating child care deserts, resulting in positive impacts that extend beyond simply the act of providing childcare. At a high level, it will enhance a child's social development, provide parents with greater freedom and time to work, and in the long run, generate positive economic benefits for the state as a whole. Based on our research, this analysis is just the first step in a broader series aimed at addressing the issue of eliminating childcare deserts. However, our work in allocating physical infrastructure to meet demand is a crucial step in initiating this process.

7 Bibliography (APA)

References

[1] Scheid, B. (2021, Oct 11). Child care shortage shuts millions of americans out of workforce. SNL Energy Coal Report Retrieved from http://ezproxy.cul.columbia.edu/login?url=https://www.proquest.com/wire-feeds/child-care-shortage-shuts-millions-americans-out/docview/2581103456/se-2

[2] Sipple, J. W., McCabe, L. A., & Casto, H. G. (2020). Child care deserts in New York State: Prekindergarten implementation and community factors related to the capacity to care for infants and toddlers. *Early Childhood Research Quarterly*, *51*, 167-177.

8 Appendix

Code Appendix

Nov 5, 2024

[17]:
```
import pandas as pd
import numpy as np
from gurobipy import Model, GRB, quicksum
from haversine import haversine, Unit
import warnings
warnings.filterwarnings("ignore")
```

1 Data Preprocessing

[18]:
```
#Data preprocessing
childcare = pd.read_csv(r'project1_new_datasets\new_child_care.csv')
employment = pd.read_csv(r'project1_new_datasets\new_employment.csv')
income = pd.read_csv(r'project1_new_datasets\new_income.csv')
population = pd.read_csv(r'project1_new_datasets\new_population.csv')
potential_loc = pd.read_csv(r'project1_new_datasets\new_potential_loc.csv')


population = population.iloc[:, :5].drop(['Total'], axis=1)
population['2w-12yrs'] = np.floor(population.iloc[:, 1:3].sum(axis=1)+3/
 ↪5*population.iloc[:, 3]).astype(int)
print(population)
demand_desert = pd.merge(population, employment, on='zip_code', how = 'outer')
demand_desert = pd.merge(demand_desert, income, on='zip_code', how = 'outer')
demand_desert['high_demand'] = (demand_desert['employment rate'] >= 0.
 ↪6)|(demand_desert['average income'] <= 60000)
demand_desert['high_demand'] = demand_desert['high_demand'].astype(int)
#Data cleaning
childcare=childcare[childcare['total_capacity']>0].reset_index(drop=True)
childcare_capacity = childcare.
 ↪groupby('zip_code')[['infant_capacity','toddler_capacity',
                                                    ␣
 ↪'preschool_capacity','school_age_capacity'
                                                    ,'children_capacity']].sum().
 ↪reset_index()
childcare_capacity['2w_5yr_cap'] = np.floor(childcare_capacity.iloc[:, 1:4].
 ↪sum(axis=1)+childcare_capacity['children_capacity']*5/12).astype(int)
```

```
childcare_capacity['2w_12yr_cap'] = np.floor(childcare_capacity.iloc[:, 1:6].
 ↪sum(axis=1)).astype(int)

demand_desert = pd.merge(demand_desert, childcare_capacity, on='zip_code', how␣
 ↪= 'outer')

demand_desert.reset_index(drop=True, inplace=True)

#def classify_desert(row):
#    if row['high_demand'] == 1:
#        return row['2w_12yr_cap'] <= row['2w-12yrs']*0.5
#    else:
#        return row['2w_12yr_cap'] <= row['2w-12yrs']*1/3
#
#demand_desert['desert'] = demand_desert.apply(classify_desert, axis=1).
 ↪astype(int)

demand_desert.to_csv(r'project1_new_datasets\demand_desert.csv', index=False)
```

```
      zip_code    -5   5-9  10-14  2w-12yrs
0        10001   744   784    942      2093
1        10002  2142  3046   3198      7106
2        10003  1440  1034    953      3045
3        10004   433   182    161       711
4        10005   484   204    229       825
…          …    …     …      …         …
1018     14767   101   219    168       420
1019     14770   137   197    223       467
1020     14772   256   253    224       643
1021     14805    31    16     29        64
1022     14806   127   111    154       330

[1023 rows x 5 columns]
```

[19]:
```
childcare.head()
```

[19]:
```
   zip_code  facility_id program_type facility_status  \
0     10001       837597         SACC    Registration
1     10001       661697         GFDC         License
2     10001       837329         SACC    Registration
3     10001       350076          FDC    Registration
4     10001       292419         SACC    Registration

                    facility_name      city school_district_name  \
0        I Have a Dream Foundation  New York          Manhattan 2
1   Chelsea Little Angels Day Care  New York          Manhattan 2
2  Bright Horizons at Hudson Yards  New York          Manhattan 2
```

```
3                    GRAMMAS HANDS  New York          Manhattan 2
4    The Hudson Guild @26th Street  New York          Manhattan 2

   infant_capacity  toddler_capacity  preschool_capacity  school_age_capacity  \
0                0                 0                   0                   84
1                0                 0                   0                    4
2                0                 0                   0                   17
3                0                 0                   0                    2
4                0                 0                   0                   79

   children_capacity  total_capacity   latitude  longitude
0                  0              84  40.748836 -73.999810
1                 12              16  40.748911 -74.001546
2                  0              17  40.752093 -74.002588
3                  6               8  40.748296 -74.001263
4                  0              79  40.749247 -74.001598
```

[20]:
```
demand_desert.head()
```

[20]:
```
   zip_code    -5   5-9  10-14  2w-12yrs  employment rate  average income  \
0     10001   744   784    942      2093         0.595097   102878.033603
1     10002  2142  3046   3198      7106         0.520662    59604.041165
2     10003  1440  1034    953      3045         0.497244   114273.049645
3     10004   433   182    161       711         0.506661   132004.310345
4     10005   484   204    229       825         0.665833   121437.713311

   high_demand  infant_capacity  toddler_capacity  preschool_capacity  \
0            0                0                 0                   0
1            1                0                 0                  18
2            0                0                 0                   0
3            0                0                 0                   0
4            1                0                 0                   0

   school_age_capacity  children_capacity  2w_5yr_cap  2w_12yr_cap
0                  585                 24          10          609
1                 4508                203         102         4729
2                 1995                  0           0         1995
3                  263                  0           0          263
4                   39                  0           0           39
```

2 The Problem of Budgeting

[21]:
```
#Problem 1
m1=Model("Budgeting")
x={}
y={}
```

```
#Decision variables
for i in range(len(childcare)):
    x[1,i]=m1.addVar(vtype=GRB.INTEGER,name=f"new slots at facility {i}")
    x[2,i]=m1.addVar(vtype=GRB.INTEGER,name=f"new slots for children under 5 at␣
 ↪facility {i}")
for j in range(len(demand_desert)):
    y[1,j]=m1.addVar(vtype=GRB.INTEGER,name=f"newly built small facilities in␣
 ↪{j}")
    y[2,j]=m1.addVar(vtype=GRB.INTEGER,name=f"newly built medium facilities in␣
 ↪{j}")
    y[3,j]=m1.addVar(vtype=GRB.INTEGER,name=f"newly built large facilities in␣
 ↪{j}")
#Objective function
m1.setObjective(
    quicksum(65000*y[1,j]+95000*y[2,j]+115000*y[3,j] for j in␣
 ↪range(len(demand_desert)))+
        quicksum(200*x[1,i]+100*x[2,i]+20000*(x[1,i]/
 ↪childcare["total_capacity"][i])
            for i in range(len(childcare))),GRB.MINIMIZE
)
#Constraints
for i in range(len(childcare)):
    m1.addConstr(x[1,i]<=0.2*childcare["total_capacity"][i],f"Maximum expansion␣
 ↪rate {i}")
    m1.addConstr(x[2,i]-x[1,i]<=0)
    m1.addConstr(childcare["total_capacity"][i]+x[1,i]<=
                 max(childcare["total_capacity"][i],500),f"Maximum slots {i}")
    m1.addConstr(x[1,i]>=0,f"non-negativity {i}_1")
    m1.addConstr(x[2,i]>=0,f"non-negativity {i}_2")

care_reg=childcare.groupby("zip_code")
for j in range(len(demand_desert)):
    m1.addConstr(y[1,j]>=0,f"non-negativity {j}_y_1")
    m1.addConstr(y[2,j]>=0,f"non-negativity {j}_y_2")
    m1.addConstr(y[3,j]>=0,f"non-negativity {j}_y_3")
    jcare=care_reg.get_group(demand_desert.iloc[j,0]).index
    #high demand or not(changing greater than to greater than or equal)
    if demand_desert["high_demand"][j]==1:
        m1.
 ↪addConstr(demand_desert["2w_12yr_cap"][j]+100*y[1,j]+200*y[2,j]+400*y[3,j]+
                        quicksum(x[1,l] for l in jcare)>=int(1/
 ↪2*(demand_desert["2w-12yrs"][j]))+1)
    else:
        m1.
 ↪addConstr(demand_desert["2w_12yr_cap"][j]+100*y[1,j]+200*y[2,j]+400*y[3,j]+
```

```
                        quicksum(x[1,l] for l in jcare)>=int(1/
  ↪3*(demand_desert["2w-12yrs"][j]))+1)
    m1.addConstr(demand_desert["2w_5yr_cap"][j]+50*y[1,j]+100*y[2,j]+200*y[3,j]+
                        quicksum(x[2,l] for l in jcare)>=2/
  ↪3*(demand_desert["-5"][j]))
m1.optimize()
```

```
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 78890 rows, 32579 columns and 127247 nonzeros
Model fingerprint: 0xc733106a
Variable types: 0 continuous, 32579 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+02]
  Objective range  [1e+02, 1e+05]
  Bounds range     [0e+00, 0e+00]
  RHS range        [6e-01, 1e+04]
Found heuristic solution: objective 3.605250e+08
Presolve removed 78600 rows and 32000 columns
Presolve time: 0.84s
Presolved: 290 rows, 579 columns, 1158 nonzeros
Found heuristic solution: objective 3.159789e+08
Variable types: 0 continuous, 579 integer (88 binary)
Found heuristic solution: objective 3.159489e+08

Root relaxation: objective 3.157883e+08, 289 iterations, 0.00 seconds (0.00 work
units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.1579e+08    0    1 3.1595e+08 3.1579e+08  0.05%     -    0s
H    0     0                     3.158804e+08 3.1579e+08  0.03%     -    0s
H    0     0                     3.158466e+08 3.1579e+08  0.02%     -    0s
H    0     0                     3.157883e+08 3.1579e+08  0.00%     -    0s
     0     0 3.1579e+08    0    1 3.1579e+08 3.1579e+08  0.00%     -    0s

Explored 1 nodes (289 simplex iterations) in 0.87 seconds (0.22 work units)
Thread count was 16 (of 16 available processors)

Solution count 6: 3.15788e+08 3.15847e+08 3.1588e+08 … 3.60525e+08
```

```
Optimal solution found (tolerance 1.00e-04)
Best objective 3.157883173882e+08, best bound 3.157883173882e+08, gap 0.0000%
```

[22]:
```python
# Initialize lists to store results for CSV
facility_expansion_data = []
new_facilities_data = []

# Assuming the optimization model has been solved, let's collect the data
if m1.status == GRB.OPTIMAL:
    # Gather facility expansion data
    for i in range(len(childcare)):
        facility_expansion_data.append({
            "Facility ID": i,
            "Expanded Slots": x[1, i].x,
            "Slots for Children Under 5": x[2, i].x
        })

    # Gather data on new facilities
    for j in range(len(demand_desert)):
        new_facilities_data.append({
            "Region Zip Code": demand_desert['zip_code'][j],
            "Small Facilities": y[1, j].x,
            "Medium Facilities": y[2, j].x,
            "Large Facilities": y[3, j].x
        })

    # Create DataFrames
    facility_expansion_df = pd.DataFrame(facility_expansion_data)
    new_facilities_df = pd.DataFrame(new_facilities_data)

    # Save to CSV files
    facility_expansion_df.to_csv("facility_expansion_p1.csv", index=False)
    new_facilities_df.to_csv("new_facilities_p1.csv", index=False)
else:
    print("No optimal solution found")
```

[23]:
```python
facility_expansion_df
```

[23]:

	Facility ID	Expanded Slots	Slots for Children Under 5
0	0	11.0	11.0
1	1	-0.0	0.0
2	2	-0.0	0.0
3	3	-0.0	0.0
4	4	0.0	-0.0
…	…	…	…
14750	14750	0.0	-0.0
14751	14751	0.0	0.0

```
14752        14752             0.0                        0.0
14753        14753            -0.0                        0.0
14754        14754             0.0                       -0.0

[14755 rows x 3 columns]
```

[24]:
```
new_facilities_df
```

[24]:

	Region Zip Code	Small Facilities	Medium Facilities	Large Facilities
0	10001	-0.0	-0.0	2.0
1	10002	-0.0	-0.0	3.0
2	10003	-0.0	-0.0	4.0
3	10004	1.0	-0.0	1.0
4	10005	0.0	0.0	2.0
…	…	…	…	…
1018	14767	0.0	1.0	-0.0
1019	14770	0.0	1.0	-0.0
1020	14772	0.0	0.0	1.0
1021	14805	1.0	-0.0	-0.0
1022	14806	0.0	1.0	-0.0

[1023 rows x 4 columns]

3 The Problem of Realistic Capacity Expansion and Distance

[25]:
```
#Problem 2
m2=Model("Realistic Capacity Expansion and Distance")
x={}
y={}
a={}
#Decision variables
#a:at the upper bound of section 1 and 2
for i in range(len(childcare)):
    x[1,i]=m2.addVar(vtype=GRB.INTEGER,name=f"new slots at facility {i} between␣
 ↪0% and 10%")
    x[2,i]=m2.addVar(vtype=GRB.INTEGER,name=f"new slots at facility {i} between␣
 ↪10% and 15%")
    x[3,i]=m2.addVar(vtype=GRB.INTEGER,name=f"new slots at facility {i} between␣
 ↪15% and 20%")
    x[4,i]=m2.addVar(vtype=GRB.INTEGER,name=f"new slots for children under 5 at␣
 ↪facility {i}")
    a[1,i]=m2.addVar(vtype=GRB.BINARY,name=f"10%")
    a[2,i]=m2.addVar(vtype=GRB.BINARY,name=f"15%")
for j in range(len(potential_loc)):
    y[1,j]=m2.addVar(vtype=GRB.BINARY,name=f"build a small facility in {j}")
    y[2,j]=m2.addVar(vtype=GRB.BINARY,name=f"build a medium facility in {j}")
```

```
    y[3,j]=m2.addVar(vtype=GRB.BINARY,name=f"build a large facility in {j}")
#Objective function
m2.setObjective(
    quicksum(65000*y[1,j]+95000*y[2,j]+115000*y[3,j] for j in␣
 ↪range(len(potential_loc)))+
      ␣
 ↪quicksum(200*x[1,i]+400*x[2,i]+1000*x[3,i]+100*x[4,i]+20000*((x[1,i]+x[2,i]+x[3,i])/
 ↪childcare["total_capacity"][i])
           for i in range(len(childcare))),GRB.MINIMIZE
)
#Constraints
for i in range(len(childcare)):
    m2.addConstr(x[1,i]+x[2,i]+x[3,i]<=0.
 ↪2*childcare["total_capacity"][i],f"Maximum expansion rate {i}")
    m2.addConstr(x[4,i]-x[1,i]-x[2,i]-x[3,i]<=0)
    m2.addConstr(childcare["total_capacity"][i]+x[1,i]+x[2,i]+x[3,i]<=
                 max(childcare["total_capacity"][i],500),f"Maximum slots {i}")

care_reg=childcare.groupby("zip_code")
new_pos=potential_loc.groupby("zip_code")
for j in range(len(demand_desert)):
    jcare=care_reg.get_group(demand_desert.iloc[j,0]).index
    jpos=new_pos.get_group(demand_desert.iloc[j,0]).index
    #high demand or not(changing greater than to greater than or equal)
    if demand_desert["high_demand"][j]==1:
        m2.
 ↪addConstr(demand_desert["2w_12yr_cap"][j]+quicksum(100*y[1,k]+200*y[2,k]+400*y[3,k]␣
 ↪for k in jpos)+
                        quicksum(x[1,l]+x[2,l]+x[3,l] for l in jcare)>=int(1/
 ↪2*(demand_desert["2w-12yrs"][j]))+1)
    else:
        m2.
 ↪addConstr(demand_desert["2w_12yr_cap"][j]+quicksum(100*y[1,k]+200*y[2,k]+400*y[3,k]␣
 ↪for k in jpos)+
                        quicksum(x[1,l]+x[2,l]+x[3,l] for l in jcare)>=int(1/
 ↪3*(demand_desert["2w-12yrs"][j]))+1)
    m2.
 ↪addConstr(demand_desert["2w_5yr_cap"][j]+quicksum(50*y[1,k]+100*y[2,k]+200*y[3,k]␣
 ↪for k in jpos)+
                        quicksum(x[4,l] for l in jcare)>=2/
 ↪3*(demand_desert["-5"][j]))

#Only one type at most
for j in range(len(potential_loc)):
    m2.addConstr(y[1,j]+y[2,j]+y[3,j]<=1,f"One type for {j}")
```

```
#piecewise
for i in range(len(childcare)):
    # Constraints for x[1,i]
    m2.addConstr(x[1,i]>=a[1,i]*int(0.1*childcare["total_capacity"][i]),␣
 ↪f"200_1")
    m2.addConstr(x[1,i]<=int(0.1*childcare["total_capacity"][i]),f"200_2")
    m2.addConstr(x[2,i]>=(int(0.15*childcare["total_capacity"][i])-int(0.
 ↪1*childcare["total_capacity"][i]))*a[2,i],f"400_1")
    m2.addConstr(x[2,i]<=(int(0.15*childcare["total_capacity"][i])-int(0.
 ↪1*childcare["total_capacity"][i]))*a[1,i],f"400_2")
    m2.addConstr(x[3,i]>=0,f"1000_1")
    m2.addConstr(x[3,i]<=(int(0.2*childcare["total_capacity"][i])-int(0.
 ↪15*childcare["total_capacity"][i]))*a[2,i],f"1000_2")

#Distance
for j in range(len(demand_desert)):
    jcare=care_reg.get_group(demand_desert.iloc[j,0]).index
    jpos=new_pos.get_group(demand_desert.iloc[j,0]).index
    #between original and new facilities
    for k in jcare:
        if np.isnan(childcare["latitude"][k]) or np.
 ↪isnan(childcare["longitude"][k]):
            continue
        for l in jpos:
            ␣
 ↪dist0=haversine((childcare["latitude"][k],childcare["longitude"][k]),
                              ␣
 ↪(potential_loc["latitude"][l],potential_loc["longitude"][l]),
                              unit=Unit.MILES)
            if dist0<0.06:
                m2.addConstr(y[1,l]+y[2,l]+y[3,l]==0)
    #between new facilities
    for k in jpos:
        for l in jpos:
            if l>k:
                ␣
 ↪dist1=haversine((potential_loc["latitude"][k],potential_loc["longitude"][k]),
                                  ␣
 ↪(potential_loc["latitude"][l],potential_loc["longitude"][l]),
                                  unit=Unit.MILES)
                #Can't be chosen at the same time if dist1<0.06
                if dist1<0.06:
                    m2.addConstr(y[1,k]+y[2,k]+y[3,k]+y[1,l]+y[2,l]+y[3,l]<=1)
m2.optimize()
```

```
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))
```

```
CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 282106 rows, 395430 columns and 1519042 nonzeros
Model fingerprint: 0x40b707a9
Variable types: 0 continuous, 395430 integer (336410 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+02]
  Objective range  [1e+02, 1e+05]
  Bounds range     [1e+00, 1e+00]
  RHS range        [6e-01, 1e+04]
Presolve removed 277075 rows and 387646 columns (presolve time = 50s) …
Presolve removed 277070 rows and 387643 columns
Presolve time: 49.96s
Presolved: 5036 rows, 7787 columns, 21580 nonzeros
Variable types: 0 continuous, 7787 integer (5686 binary)
Found heuristic solution: objective 3.320767e+08
Found heuristic solution: objective 3.311750e+08
Found heuristic solution: objective 3.293641e+08

Root simplex log…

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.0628423e+08   4.117375e+03   0.000000e+00     50s
    1943    3.2023925e+08   0.000000e+00   0.000000e+00     50s

Root relaxation: objective 3.202392e+08, 1943 iterations, 0.02 seconds (0.01
work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.2024e+08    0    8 3.2936e+08 3.2024e+08  2.77%     -   50s
H    0     0                     3.205571e+08 3.2024e+08  0.10%     -   50s
H    0     0                     3.203026e+08 3.2024e+08  0.02%     -   50s
H    0     0                     3.202693e+08 3.2024e+08  0.01%     -   50s

Explored 1 nodes (1943 simplex iterations) in 50.34 seconds (20.49 work units)
Thread count was 16 (of 16 available processors)

Solution count 6: 3.20269e+08 3.20303e+08 3.20557e+08 … 3.32077e+08

Optimal solution found (tolerance 1.00e-04)
Best objective 3.202693370780e+08, best bound 3.202392497221e+08, gap 0.0094%
```

[26]:
```
facility_expansion_data_p2 = []
new_facilities_data_p2 = []

# Assuming the optimization model has been solved, let's collect the data for␣
 ↪Problem 2
if m2.status == GRB.OPTIMAL:
    # Gather facility expansion data
    for i in range(len(childcare)):
        facility_expansion_data_p2.append({
            "Facility ID": i,
            "Expanded Slots": x[1, i].x + x[2, i].x + x[3, i].x,
            "Slots for Children Under 5": x[4, i].x
        })

    # Gather data on new facilities
    for j in range(len(potential_loc)):
        small_facility = int(y[1, j].x)
        medium_facility = int(y[2, j].x)
        large_facility = int(y[3, j].x)

        if small_facility == 1 or medium_facility == 1 or large_facility == 1:
            new_facilities_data_p2.append({
                "Region Zip Code": potential_loc['zip_code'][j],
                "Longitude": potential_loc['longitude'][j],
                "Latitude": potential_loc['latitude'][j],
                "Small Facility": small_facility,
                "Medium Facility": medium_facility,
                "Large Facility": large_facility
            })

    # Create DataFrames
    facility_expansion_df_p2 = pd.DataFrame(facility_expansion_data_p2)
    new_facilities_df_p2 = pd.DataFrame(new_facilities_data_p2)

    # Save to CSV files
    facility_expansion_df_p2.to_csv("facility_expansion_p2.csv", index=False)
    new_facilities_df_p2.to_csv("new_facilities_p2.csv", index=False)
else:
    print("No optimal solution found")
```

[27]:
```
facility_expansion_df_p2
```

[27]:

	Facility ID	Expanded Slots	Slots for Children Under 5
0	0	12.0	12.0
1	1	0.0	-0.0
2	2	0.0	-0.0
3	3	0.0	0.0

```
4                4            11.0                        11.0
…              …              …                           …
14750        14750              0.0                         0.0
14751        14751              0.0                         0.0
14752        14752              0.0                         0.0
14753        14753              0.0                        -0.0
14754        14754              0.0                        -0.0

[14755 rows x 3 columns]
```

[28]:
```
new_facilities_df_p2
```

[28]:
```
      Region Zip Code  Longitude   Latitude  Small Facility  Medium Facility  \
0               10001 -74.004994  40.740011               0                0
1               10001 -73.995710  40.744408               0                0
2               10002 -73.993451  40.707449               0                0
3               10002 -73.987162  40.720105               0                0
4               10002 -73.983491  40.723590               0                0
…                 …        …         …              …               …
2751            14767 -79.482863  42.064277               0                1
2752            14770 -78.327679  42.022032               0                1
2753            14772 -78.966816  42.169988               0                0
2754            14805 -76.724885  42.358652               1                0
2755            14806 -77.796393  42.150087               0                1

      Large Facility
0                  1
1                  1
2                  1
3                  1
4                  1
…                …
2751               0
2752               0
2753               1
2754               0
2755               0

[2756 rows x 6 columns]
```

4 The Problem of Fairness

[29]:
```
# Problem 3
m3=Model("Fairness")
x={}
y={}
a={}
```

```
# Decision variables
for i in range(len(childcare)):
    x[1, i] = m3.addVar(vtype=GRB.INTEGER, name=f"new slots at facility {i}␣
 ↪between 0% and 10%")
    x[2, i] = m3.addVar(vtype=GRB.INTEGER, name=f"new slots at facility {i}␣
 ↪between 10% and 15%")
    x[3, i] = m3.addVar(vtype=GRB.INTEGER, name=f"new slots at facility {i}␣
 ↪between 15% and 20%")
    x[4, i] = m3.addVar(vtype=GRB.INTEGER, name=f"new slots for children under␣
 ↪5 at facility {i}")
    a[1, i] = m3.addVar(vtype=GRB.BINARY, name=f"10%")
    a[2, i] = m3.addVar(vtype=GRB.BINARY, name=f"15%")

# Decision Variables for New Facilities
for j in range(len(potential_loc)):
    y[1, j] = m3.addVar(vtype=GRB.BINARY, name=f"build a small facility in {j}")
    y[2, j] = m3.addVar(vtype=GRB.BINARY, name=f"build a medium facility in␣
 ↪{j}")
    y[3, j] = m3.addVar(vtype=GRB.BINARY, name=f"build a large facility in {j}")

# Define the minimum and maximum slot ratio variables
min_ratio = m3.addVar(lb=0, vtype=GRB.CONTINUOUS, name="min_slot_ratio")
max_ratio = m3.addVar(lb=0, vtype=GRB.CONTINUOUS, name="max_slot_ratio")

# Objective Function: Maximize the Social Coverage Index
m3.setObjective(
        2 * (
            (demand_desert['2w_5yr_cap'].sum()+
            (quicksum(x[4, i] for i in range(len(childcare))) + quicksum(50 *␣
 ↪y[1, j] + 100 * y[2, j] + 200 * y[3, j] for j in␣
 ↪range(len(potential_loc))))) / demand_desert["-5"].sum()
        ) +
        (
            (demand_desert['2w_12yr_cap'].sum()+
            (quicksum(x[1, i] + x[2, i] + x[3, i] for i in␣
 ↪range(len(childcare))) + quicksum(100 * y[1, j] + 200 * y[2, j] + 400 * y[3,␣
 ↪j] for j in range(len(potential_loc))))) / demand_desert["2w-12yrs"].sum()
        ),
    GRB.MAXIMIZE
)

# Fairness Constraint: Calculate slot ratio dynamically using decision variables
care_reg=childcare.groupby("zip_code")
new_pos=potential_loc.groupby("zip_code")
# Calculate the slot ratios for each zip code and constrain them between␣
 ↪min_ratio and max_ratio
```

```
for i in range(len(demand_desert)):
    # Calculate the slot ratio for zip code i
    if demand_desert["2w-12yrs"][i] > 0:
        slots_ratio_i = (
            quicksum(x[1, l] + x[2, l] + x[3, l] for l in care_reg.
 ↪get_group(demand_desert.iloc[i, 0]).index) +
            quicksum(100 * y[1, k] + 200 * y[2, k] + 400 * y[3, k] for k in␣
 ↪new_pos.get_group(demand_desert.iloc[i, 0]).index)
        ) / demand_desert["2w-12yrs"][i]

        # Constrain each slot ratio to be within min_ratio and max_ratio
        m3.addConstr(slots_ratio_i >= min_ratio, f"Min ratio constraint for zip␣
 ↪{demand_desert.iloc[i,0]}")
        m3.addConstr(slots_ratio_i <= max_ratio, f"Max ratio constraint for zip␣
 ↪{demand_desert.iloc[i,0]}")
    else:
        print(f"Skipping zip code {demand_desert.iloc[i,0]} due to zero␣
 ↪population.")

# Enforce fairness by limiting the difference between max_ratio and min_ratio␣
 ↪to 0.1
m3.addConstr(max_ratio - min_ratio <= 0.1, "Fairness constraint")

# Budget Constraint
m3.addConstr(
    quicksum(65000 * y[1, j] + 95000 * y[2, j] + 115000 * y[3, j] for j in␣
 ↪range(len(potential_loc))) +
    quicksum(200 * x[1, i] + 400 * x[2, i] + 1000 * x[3, i] + 100 * x[4, i] +␣
 ↪20000 * ((x[1, i] + x[2, i] + x[3, i]) / childcare["total_capacity"][i]) for␣
 ↪i in range(len(childcare))) <= 1000000000,
    "Budget constraint"
)

# Constraint from Problem 2
for i in range(len(childcare)):
    m3.addConstr(x[1,i]+x[2,i]+x[3,i]<=0.
 ↪2*childcare["total_capacity"][i],f"Maximum expansion rate {i}")
    m3.addConstr(x[4,i]-x[1,i]-x[2,i]-x[3,i]<=0)
    m3.addConstr(childcare["total_capacity"][i]+x[1,i]+x[2,i]+x[3,i]<=
                 max(childcare["total_capacity"][i],500),f"Maximum slots {i}")

care_reg=childcare.groupby("zip_code")
new_pos=potential_loc.groupby("zip_code")
for j in range(len(demand_desert)):
    jcare=care_reg.get_group(demand_desert.iloc[j,0]).index
    jpos=new_pos.get_group(demand_desert.iloc[j,0]).index
```

```
    #high demand or not(changing greater than to greater than or equal)
    if demand_desert["high_demand"][j]==1:
        m3.
↪addConstr(demand_desert["2w_12yr_cap"][j]+quicksum(100*y[1,k]+200*y[2,k]+400*y[3,k]␣
↪for k in jpos)+
                        quicksum(x[1,l]+x[2,l]+x[3,l] for l in jcare)>=int(1/
↪2*(demand_desert["2w-12yrs"][j]))+1)
    else:
        m3.
↪addConstr(demand_desert["2w_12yr_cap"][j]+quicksum(100*y[1,k]+200*y[2,k]+400*y[3,k]␣
↪for k in jpos)+
                        quicksum(x[1,l]+x[2,l]+x[3,l] for l in jcare)>=int(1/
↪3*(demand_desert["2w-12yrs"][j]))+1)
    m3.
↪addConstr(demand_desert["2w_5yr_cap"][j]+quicksum(50*y[1,k]+100*y[2,k]+200*y[3,k]␣
↪for k in jpos)+
                        quicksum(x[4,l] for l in jcare)>=2/
↪3*(demand_desert["-5"][j]))

#Only one type at most
for j in range(len(potential_loc)):
    m3.addConstr(y[1,j]+y[2,j]+y[3,j]<=1,f"One type for {j}")

#piecewise
for i in range(len(childcare)):
    # Constraints for x[1,i]
    m3.addConstr(x[1,i]>=a[1,i]*int(0.1*childcare["total_capacity"][i]),␣
↪f"200_1")
    m3.addConstr(x[1,i]<=int(0.1*childcare["total_capacity"][i]),f"200_2")
    m3.addConstr(x[2,i]>=(int(0.15*childcare["total_capacity"][i])-int(0.
↪1*childcare["total_capacity"][i]))*a[2,i],f"400_1")
    m3.addConstr(x[2,i]<=(int(0.15*childcare["total_capacity"][i])-int(0.
↪1*childcare["total_capacity"][i]))*a[1,i],f"400_2")
    m3.addConstr(x[3,i]>=0,f"1000_1")
    m3.addConstr(x[3,i]<=(int(0.2*childcare["total_capacity"][i])-int(0.
↪15*childcare["total_capacity"][i]))*a[2,i],f"1000_2")

#Distance
for j in range(len(demand_desert)):
    jcare=care_reg.get_group(demand_desert.iloc[j,0]).index
    jpos=new_pos.get_group(demand_desert.iloc[j,0]).index
    #between original and new facilities
    for k in jcare:
        if np.isnan(childcare["latitude"][k]) or np.
↪isnan(childcare["longitude"][k]):
            continue
```

```
        for l in jpos:
            dist0=haversine((childcare["latitude"][k],childcare["longitude"][k]),
(potential_loc["latitude"][l],potential_loc["longitude"][l]),
                            unit=Unit.MILES)
            if dist0<0.06:
                m3.addConstr(y[1,l]+y[2,l]+y[3,l]==0)
    #between new facilities
    for k in jpos:
        for l in jpos:
            if l>k:
                dist1=haversine((potential_loc["latitude"][k],potential_loc["longitude"][k]),
(potential_loc["latitude"][l],potential_loc["longitude"][l]),
                                unit=Unit.MILES)
                #Can't be chosen at the same time if dist1<0.06
                if dist1<0.06:
                    m3.addConstr(y[1,k]+y[2,k]+y[3,k]+y[1,l]+y[2,l]+y[3,l]<=1)
m3.optimize()
```

```
Skipping zip code 11042 due to zero population.
Skipping zip code 12742 due to zero population.
Skipping zip code 13441 due to zero population.
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 284148 rows, 395432 columns and 2587516 nonzeros
Model fingerprint: 0xca0490b6
Variable types: 2 continuous, 395430 integer (336410 binary)
Coefficient statistics:
  Matrix range     [4e-05, 1e+05]
  Objective range  [4e-07, 5e-04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e-01, 1e+09]
Presolve removed 83711 rows and 24405 columns
Presolve time: 0.25s

Explored 0 nodes (0 simplex iterations) in 0.56 seconds (0.83 work units)
Thread count was 1 (of 16 available processors)

Solution count 0
```

```
Model is infeasible or unbounded
Best objective -, best bound -, gap -
```

[30]:
```
new_facilities_df_p2['Facility_total'] = new_facilities_df_p2.iloc[:, 3:6].
  ↪sum(axis=1)
new_facilities_df_p2
```

[30]:

	Region Zip Code	Longitude	Latitude	Small Facility	Medium Facility	\
0	10001	-74.004994	40.740011	0	0	
1	10001	-73.995710	40.744408	0	0	
2	10002	-73.993451	40.707449	0	0	
3	10002	-73.987162	40.720105	0	0	
4	10002	-73.983491	40.723590	0	0	
…	…	…	…	…	…	
2751	14767	-79.482863	42.064277	0	1	
2752	14770	-78.327679	42.022032	0	1	
2753	14772	-78.966816	42.169988	0	0	
2754	14805	-76.724885	42.358652	1	0	
2755	14806	-77.796393	42.150087	0	1	

	Large Facility	Facility_total
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1
…	…	…
2751	0	1
2752	0	1
2753	1	1
2754	0	1
2755	0	1

[2756 rows x 7 columns]

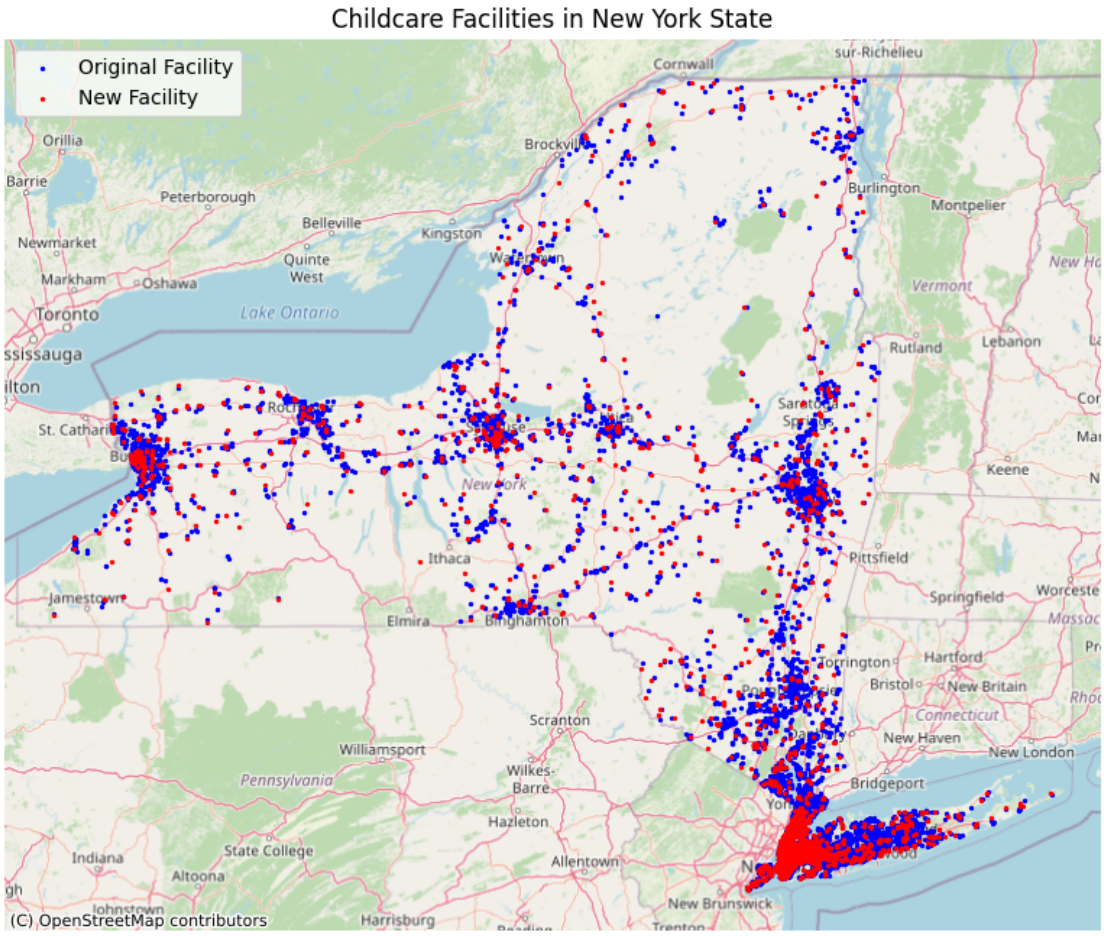
[31]:
```
import geopandas as gpd
import contextily as ctx
import matplotlib.pyplot as plt

# Convert the coordinates to a suitable CRS (usually EPSG:3857 for web maps)
gdf = gpd.GeoDataFrame(childcare, geometry=gpd.points_from_xy(childcare.
  ↪longitude, childcare.latitude), crs="EPSG:4326")
gdf_new = gpd.GeoDataFrame(new_facilities_df_p2, geometry=gpd.
  ↪points_from_xy(new_facilities_df_p2.Longitude, new_facilities_df_p2.
  ↪Latitude), crs="EPSG:4326")
gdf = gdf.to_crs(epsg=3857)  # Convert to web map projection
```

```
gdf_new = gdf_new.to_crs(epsg=3857)  # Convert to web map projection

# Plot the facilities with contextily basemap
fig, ax = plt.subplots(figsize=(10, 8))
gdf.plot(ax=ax, marker='o', color='blue', markersize=2, label="Original␣
 ↪Facility")
gdf_new.plot(ax=ax, marker='o', color='red', markersize=2, label="New Facility")
ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik)  # Google Maps␣
 ↪style basemap

ax.set_axis_off()
plt.title("Childcare Facilities in New York State")
plt.legend()
plt.show()
fig.savefig("childcare_facilities_map.png", dpi=500)
```



[32]:
```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the availability ratio
demand_desert['availability_ratio'] = demand_desert['2w_12yr_cap'] /␣
 ↪demand_desert['2w-12yrs']

# Replace infinite values (where denominator is zero) and drop NaN values
demand_desert['availability_ratio'].replace([np.inf, -np.inf], np.nan,␣
 ↪inplace=True)
demand_desert.dropna(subset=['availability_ratio'], inplace=True)

# Plotting the fairness metric comparison
plt.hist(demand_desert['availability_ratio'], bins=20, edgecolor='black')
plt.title("Childcare Availability Ratio Across Regions")
plt.xlabel("Availability Ratio")
plt.ylabel("Number of Regions")
plt.xticks(np.arange(0, 5, 0.5))
plt.xlim(0, 5)
plt.show()
```