In [1]:
```
import numpy as np
import pandas as pd
import gurobipy as gp
from gurobipy import GRB
```

In [2]:
```
#load population per LGA
data = pd.read_excel('2001-2022 census data.xlsx')
data
```

Out[2]:

	Year	S/T name	LGA name	Total persons
0	2001	New South Wales	Albury	45265
1	2001	New South Wales	Armidale	27906
2	2001	New South Wales	Ballina	37856
3	2001	New South Wales	Balranald	2751
4	2001	New South Wales	Bathurst	35504
...	...	...	...	...
12029	2022	Northern Territory	West Arnhem	7260
12030	2022	Northern Territory	West Daly	3435
12031	2022	Northern Territory	Unincorporated NT	7571
12032	2022	Australian Capital Territory	Unincorporated ACT	456844
12033	2022	Other Territories	Unincorp. Other Territories	2524

12034 rows × 4 columns

In [3]:
```
# Extracting NSW LGAs data
NSW_2001 = data[(data['Year'] == 2001) & (data['S/T name'] == 'New South Wales')]
NSW_2006 = data[(data['Year'] == 2006) & (data['S/T name'] == 'New South Wales')]
NSW_2021 = data[(data['Year'] == 2021) & (data['S/T name'] == 'New South Wales')]
LGA = NSW_2001['LGA name'].tolist()
```

NOTE:

- Since there are areas in Unincorporated New South Wales, I consider it as another Local Government Area (LGA).
- Therefore, I will be using 129 LGAs in this assignment.
- Please reload everything if there is a problem with 'Year' data (it sometimes happened).

Part 1

In [16]:
```
# Generate random distances
np.random.seed(1)
distance = {(i, j): np.random.randint(50, 500) if i != j else 0 for i in LGA for j in LGA}

# Print the generated distances
#for (i, j), d in distance.items():
#    print(f"Distance from {i} to {j}: {d} km")
```

In [9]:
```
# I created a dictionary to map age groups to population percentages and RDI
age_group_data = {
    '0-4 years old': {'population_percentage': 0.066283412, 'milk_intake': 500.0/100000.0},
    '5-9 years old': {'population_percentage': 0.069993856, 'milk_intake': 700.0/100000.0},
    '10-14 years old': {'population_percentage': 0.069843661, 'milk_intake': 1150.0/100000.0},
    '15-19 years old': {'population_percentage': 0.068525341, 'milk_intake': 1300.0/100000.0},
    '20-24 years old': {'population_percentage': 0.064145536, 'milk_intake': 1000.0/100000.0},
    '25-29 years old': {'population_percentage': 0.070077349, 'milk_intake': 1000.0/100000.0},
    '30-34 years old': {'population_percentage': 0.073531505, 'milk_intake': 1000.0/100000.0},
    '35-39 years old': {'population_percentage': 0.075803881, 'milk_intake': 1000.0/100000.0},
    '40-44 years old': {'population_percentage': 0.075696375, 'milk_intake': 1000.0/100000.0},
    '45-49 years old': {'population_percentage': 0.068784454, 'milk_intake': 1000.0/100000.0},
    '50-54 years old': {'population_percentage': 0.06481223,  'milk_intake': 1300.0/100000.0},
    '55-59 years old': {'population_percentage': 0.051058227, 'milk_intake': 1300.0/100000.0},
    '60-64 years old': {'population_percentage': 0.041913793, 'milk_intake': 1300.0/100000.0},
    '65-69 years old': {'population_percentage': 0.035787528, 'milk_intake': 1300.0/100000.0},
    '70 and above': {'population_percentage': 0.094235723, 'milk_intake': 1300.0/100000.0}
}


#calculate the total demand for each LGA

demand_2001 = []

for lga_index, lga_row in NSW_2001.iterrows():
    lga_name = lga_row['LGA name']
```

```
    # Initialize cumulative demand for the LGA to 0
    lga_cumulative_demand = 0.0

    # Iterate over age groups and calculate the demand for each LGA
    for age_group, data in age_group_data.items():
        population_percentage = data['population_percentage']
        milk_intake = data['milk_intake']

        # Calculate the demand for age group in the LGA
        age_group_demand = lga_row['Total persons'] * population_percentage * milk_intake

        # Add the age group demand to the cumulative demand for the LGA
        lga_cumulative_demand += age_group_demand

    # Append the cumulative demand for the LGA to the lga_demand list
    demand_2001.append((lga_name, (lga_cumulative_demand)))

#demand_2001
```

In [14]:

```
model_2001 = gp.Model("2001")

# Decision Variables
# x = whether to establish a center at LGA
x = model_2001.addVars(LGA, LGA, vtype=GRB.BINARY, name="x")
# Capacity of the centre
y = model_2001.addVars(LGA, vtype=GRB.CONTINUOUS, name="y")


# Objective Function
obj_2001 = (
    gp.quicksum(distance[i, j] * demand_2001[LGA.index(i)][1] * 0.1 for i in LGA for j in LGA)
    + gp.quicksum(y[i] * 1000 for i in LGA)
)

model_2001.setObjective(obj_2001, GRB.MINIMIZE)

# Constraints
# Constraint 1: Maximum 3 distribution centers
model_2001.addConstr(gp.quicksum(x[i, j] for i in LGA) <= 3)

# Constraint 2: Center capacity limit
model_2001.addConstrs(y[i] <= 0.5 * gp.quicksum(demand_2001[LGA.index(j)][1] for j in LGA) for i in LGA)

# Constraint 3: Demand fulfillment
model_2001.addConstrs(gp.quicksum(x[i, j] for i in LGA if i != j) >= 1 for j in LGA)

# Constraint 4: Linking variables
model_2001.addConstrs(x[i, j] <= y[i] for i in LGA for j in LGA)


# Optimize the model
model_2001.update()
model_2001.optimize()
```

```
Gurobi Optimizer version 10.0.2 build v10.0.2rc0 (mac64[x86])

CPU model: Intel(R) Core(TM) i5-5350U CPU @ 1.80GHz
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads

Optimize a model with 16900 rows, 16770 columns and 50052 nonzeros
Model fingerprint: 0x8b6196b3
Variable types: 129 continuous, 16641 integer (16641 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+03, 1e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+04]
Found heuristic solution: objective 2.406964e+08
Presolve removed 258 rows and 129 columns
Presolve time: 0.21s
Presolved: 16642 rows, 16641 columns, 49664 nonzeros
Variable types: 0 continuous, 16641 integer (16641 binary)

Root relaxation: objective 2.406144e+08, 16641 iterations, 0.97 seconds (0.42 work units)

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

     0     0 2.4061e+08    0 16641 2.4070e+08 2.4061e+08  0.03%     -    1s
H    0     0                     2.406954e+08 2.4061e+08  0.03%     -    1s
H    0     0                     2.406154e+08 2.4061e+08  0.00%     -    1s

Explored 1 nodes (16641 simplex iterations) in 1.67 seconds (0.73 work units)
Thread count was 4 (of 4 available processors)

Solution count 3: 2.40615e+08 2.40695e+08 2.40696e+08

Optimal solution found (tolerance 1.00e-04)
Best objective 2.406154300236e+08, best bound 2.406154300236e+08, gap 0.0000%
```

In [15]:
```python
if model_2001.status == GRB.OPTIMAL:
    print("Selected distribution centers:")
    for i in LGA:
        if y[i].x > 0.5:
            print(f"Distribution center at {i}")


    minimal_cost_2001 = model_2001.objVal
    print(f"Minimal Cost: $ {minimal_cost_2001:.2f}")
```

```
Selected distribution centers:
Distribution center at Warrumbungle
Distribution center at Yass Valley
Minimal Cost: $ 240615430.02
```

COMMENT:

The objective of this model is to create a sophisticated optimisation model that can address the difficult task of positioning up to three distribution centres for an effective supply of milk in New South Wales (NSW). The model is built according to:

- An actual approximation of population of each LGA in New South Wales (NSW) (Australian Bureau of Statistics, 2022).
- percenatge of population of NSW (Australian Bureau of Statistics, 2022).
- Random generated distances from LGAs to each LGAs.
- Recommended dietary intake of calcium (RDI) (note that I have generalised the approximate RDI for the given group to match the data I have obtained)
- The assumption that a litre of milk consists of 100000 mg of calcium

The variables above are going to be utilised to create the demand of milk in each LGA with the formulation of:

- demand of milk in LGA[i] : The sum of (LGA[i] population *every population percentage in age group* every milk intake in each age group)

After obtaining the demands for each LGA, optimisation model is formulated:

Variables:

- Let x[i,j] be a binary decision variable that indicates whether to establish a distribution centre at LGA[i] to service LGAj]. x[i,j] is a binary variable, and it takes value of 1 if a distribution centre is established, and 0 otherwise.
- Let y[i] be a continuous descision variable that indicates capacity of the distribution centre in LGA[i].

Objective:

- minimise the distribution cost of n litres of milk from distribution centres to LGAs (distance *demand* 0.1).
- minimise the cost of establishing a distibution (capacity *1000)* --> *(y[i]* 1000)

Constraints:

- x[i,j] <= 3 (a maximum of 3 distribution centre could be built)
- y[i] <= 0.5 * sum of entire state's demand (distribution centre should not be allowed to have more production capacity than 50% of the entire state's demand.)
- x[i,j] >= 1 (demand fulfilment/the demand of each LGA should be met)
- x[i,j] <= y[i] (Linking constraint --> ensures that if x[i, j] is 1, y[i] must be greater than or equal to 0)

Result:

- According to the model, 2 distribution centres is enough to supply NSW's demand for milk.
- Distribution center should be built in Warrumbungle and Yass Valley.
- the minimal cost for distrubution of milk to every LGA is $ 240615430.02

Part 2

In [11]:
```
years = [2002, 2003, 2004]  # Years 2002, 2003, and 2004

# Create an empty dictionary to store demand data for each year
demand_data = {}

for year in years:
    # Create a list to store the demand for the current year
    demand_year = []

    for lga_index, lga_row in NSW_2001.iterrows():
        lga_name = lga_row['LGA name']

        # population for the year with growth
        population = lga_row['Total persons'] * ((1 + 0.10) ** (year - 2001))

        # Initialize cumulative demand for the LGA to 0
        lga_cumulative_demand = 0.0

        # Iterate over age groups and calculate the demand for the LGA
        for age_group, data in age_group_data.items():
            population_percentage = data['population_percentage']
            calcium_intake = data['milk_intake']

            # Calculate the demand for age group in the LGA
            age_group_demand = population * population_percentage * milk_intake

            # Add the age group demand to the cumulative demand for the LGA
            lga_cumulative_demand += age_group_demand

        # Append the cumulative demand for the LGA to the demand_year list
        demand_year.append((lga_name, lga_cumulative_demand))

    # Store the demand data for the current year in the demand_data dictionary
    demand_data[year] = demand_year

# Loop through each year and optimize the distribution center locations
for year in demand_data:

    model_3 = gp.Model(f"distribution_center_{year}")


    x = model_3.addVars(LGA, LGA, vtype=GRB.BINARY, name="x")
    y = model_3.addVars(LGA, vtype=GRB.CONTINUOUS, name="y")

    # Objective Function
    obj = (
        gp.quicksum(distance[i, j] * demand_data[year][LGA.index(i)][1] * 0.1 for i in LGA for j in LGA)
        + gp.quicksum(y[i] * 1000 for i in LGA)
    )

    model_3.setObjective(obj, GRB.MINIMIZE)

    #constraints
    model_3.addConstr(gp.quicksum(x[i,j] for i in LGA) <= 3, name="Max_Centers")
    model_3.addConstrs(y[i] <= 0.5 * gp.quicksum(demand_data[year][LGA.index(j)][1] for j in LGA) for i in LGA)
    model_3.addConstrs(gp.quicksum(x[i, j] for i in LGA if i != j) >= 1 for j in LGA)
    model_3.addConstrs((x[i, j] <= y[i] for i in LGA for j in LGA))

    #printing results
    model_3.update()
    model_3.optimize()

    if model_3.status == GRB.OPTIMAL:
        print(f"Year {year} - Results:")
        print("Selected distribution centers:")
        for i in LGA:
            if y[i].x > 0.5:
                print(f"Distribution center at {i}")
```

```
        minimal_cost = model_3.objVal
        print(f"Minimal Cost for Year {year}: $ {minimal_cost:.2f}")
```

```
Gurobi Optimizer version 10.0.2 build v10.0.2rc0 (mac64[x86])

CPU model: Intel(R) Core(TM) i5-5350U CPU @ 1.80GHz
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads

Optimize a model with 16900 rows, 16770 columns and 50052 nonzeros
Model fingerprint: 0xc6cdca51
Variable types: 129 continuous, 16641 integer (16641 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+03, 1e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 5e+04]
Found heuristic solution: objective 3.235114e+08
Presolve removed 258 rows and 129 columns
Presolve time: 0.33s
Presolved: 16642 rows, 16641 columns, 49664 nonzeros
Variable types: 0 continuous, 16641 integer (16641 binary)

Root relaxation: objective 3.234294e+08, 16641 iterations, 0.63 seconds (0.42 work units)

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

     0     0 3.2343e+08    0 16641 3.2351e+08 3.2343e+08  0.03%     -    1s
H    0     0                     3.235104e+08 3.2343e+08  0.03%     -    1s
H    0     0                     3.234304e+08 3.2343e+08  0.00%     -    1s

Explored 1 nodes (16641 simplex iterations) in 1.31 seconds (0.73 work units)
Thread count was 4 (of 4 available processors)

Solution count 3: 3.2343e+08 3.2351e+08 3.23511e+08

Optimal solution found (tolerance 1.00e-04)
Best objective 3.234304013926e+08, best bound 3.234304013926e+08, gap 0.0000%
Year 2002 - Results:
Selected distribution centers:
Distribution center at Warrumbungle
Distribution center at Yass Valley
Minimal Cost for Year 2002: $ 323430401.39
Gurobi Optimizer version 10.0.2 build v10.0.2rc0 (mac64[x86])

CPU model: Intel(R) Core(TM) i5-5350U CPU @ 1.80GHz
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads

Optimize a model with 16900 rows, 16770 columns and 50052 nonzeros
Model fingerprint: 0x8bc0e61a
Variable types: 129 continuous, 16641 integer (16641 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+03, 1e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 5e+04]
Found heuristic solution: objective 3.558542e+08
Presolve removed 258 rows and 129 columns
Presolve time: 0.26s
Presolved: 16642 rows, 16641 columns, 49664 nonzeros
Variable types: 0 continuous, 16641 integer (16641 binary)

Root relaxation: objective 3.557722e+08, 16641 iterations, 0.51 seconds (0.42 work units)

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

     0     0 3.5577e+08    0 16641 3.5585e+08 3.5577e+08  0.02%     -    1s
H    0     0                     3.558532e+08 3.5577e+08  0.02%     -    1s
H    0     0                     3.557732e+08 3.5577e+08  0.00%     -    1s

Explored 1 nodes (16641 simplex iterations) in 1.10 seconds (0.73 work units)
Thread count was 4 (of 4 available processors)

Solution count 3: 3.55773e+08 3.55853e+08 3.55854e+08

Optimal solution found (tolerance 1.00e-04)
Best objective 3.557732415318e+08, best bound 3.557732415318e+08, gap 0.0000%
Year 2003 - Results:
Selected distribution centers:
Distribution center at Warrumbungle
Distribution center at Yass Valley
Minimal Cost for Year 2003: $ 355773241.53
Gurobi Optimizer version 10.0.2 build v10.0.2rc0 (mac64[x86])

CPU model: Intel(R) Core(TM) i5-5350U CPU @ 1.80GHz
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads

Optimize a model with 16900 rows, 16770 columns and 50052 nonzeros
```

```
Model fingerprint: 0xaeef7ce1
Variable types: 129 continuous, 16641 integer (16641 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+03, 1e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 6e+04]
Found heuristic solution: objective 3.914314e+08
Presolve removed 258 rows and 129 columns
Presolve time: 0.43s
Presolved: 16642 rows, 16641 columns, 49664 nonzeros
Variable types: 0 continuous, 16641 integer (16641 binary)

Root relaxation: objective 3.913494e+08, 16641 iterations, 0.86 seconds (0.42 work units)

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

     0     0 3.9135e+08    0 16641 3.9143e+08 3.9135e+08  0.02%     -    1s
H    0     0                     3.914304e+08 3.9135e+08  0.02%     -    1s
H    0     0                     3.913504e+08 3.9135e+08  0.00%     -    1s

Explored 1 nodes (16641 simplex iterations) in 1.87 seconds (0.73 work units)
Thread count was 4 (of 4 available processors)

Solution count 3: 3.9135e+08 3.9143e+08 3.91431e+08

Optimal solution found (tolerance 1.00e-04)
Best objective 3.913503656850e+08, best bound 3.913503656850e+08, gap 0.0000%
Year 2004 - Results:
Selected distribution centers:
Distribution center at Warrumbungle
Distribution center at Yass Valley
Minimal Cost for Year 2004: $ 391350365.69
```

Comment:

The population was anticipated to grow 10% per annum for the next 3 years. The distribution centre model was investigated in the sensitivity analysis With an additional consideration of population growth. Insights gained:

2002

- The model continued to choose Warrumbungle and Yass Valley as the distribution centres.
- This emphasises the stability of these distribution facilities, which seem to be equipped to effectively handle the increasing demand of 10% from 2001.
- The increase of of minimum cost is expected with a 10% population growth rate.
- The cost for 2002 become 323430401.39, which is approxinately 10% increase from the 2001 minimum cost.

2003

- The model still continued to choose Warrumbungle and Yass Valley as the distribution centres for 2003.
- This indicates that an increase of 10% population growth from 2002 still does not affect the the optimal solution.
- minimum cost become 355773241.53, approximately 10% increase from 2002's cost.
- The result indicate that the model is consistent in choosing the disrtibution centres.

2004

- As anticipated, the model still chooses the same distribution centres when population growth rates increased. The minimum price came to 391350365.69. This indicates the distribution center's adaptability to scenarios including rapid population increase.

Overall:

- the sensitivity analysis demonstrates the model's remarkable capacity to adjust to higher rates of population expansion. It always chooses the same distribution centres, demonstrating its dependability to handle increased demand. This may be due to:
  - Capactiy constraint: in this model we assume that the capacity of milk distribution model is less than or equal to 50% of the entire state's demand (instead of a fixed number), which means that the capacity will also increase as the demand grow.
- The increased of minimum cost is inevitable as it cost is related to demand, which means if demand incrase cost would increase as well.
- The model is a useful tool for the government in planning for the allocation of basic resources in the face of changing population dynamics because of its stability and adaptability.

With that said, government should always consider the population growth over time and choose the optimal location to build distribution centres to accomodate the increasing demand. Considering about capacity is also crucial. The chosen distribution centres should be able to grow their capacity as needed over time. The centres should be able to extend their operations as the population increases, even though the demand may initially be manageable. This will help to guarantee that the milk supply is consistent over the years.

Part 3

There are many factors that may affect the government's consideration of building milk distribution centres.

1. Unexpected events

- Recent global pandemic has brought attention to the need of being ready for such unexpected health events. The governments should consider how the distribution of milk might change in such circumstances.
- Events like wars or armed conflicts can have significant effects on a milk's demand, costs, and raw material availability (if imported), among other factors.
- natural disasters
- Another important external element that has the potential to have a big influence on milk distribution is disasters.

1. Economic flunctuations

- A distribution model's numerous components, such as product demand, operating expenses, and the distribution system's overall resilience, can be significantly impacted by economic swings, including both upturns and downturns.

1. Infrastructure development

- The accessibility and effectiveness of distribution networks may be impacted by infrastructure changes in a given area. New highways, trains, or other infrastructure additions may alter the distribution dynamics.

1. Market structure of milk industry

- The demand for milk may be impacted by modifications to the market structure, such as the arrival of new competitors or changes in customer preferences. To reflect these developments, government policies may need to change.

1. Alternative source of calcium

- Individuals does not need to be reliant on milk to fulfill the daily intake of calcium

1. A fixed capacity

- A fixed capacity of distribution centres would limit the optimisation model, which could also change the optimum locations and costs.

PART 4 (2006)

In [26]:
```
age_group_data_2006 = {
    '0-4 years old': {'population_percentage': 0.064196005, 'milk_intake': 500.0/100000.0},
    '5-9 years old': {'population_percentage': 0.065950882, 'milk_intake': 700.0/100000.0},
    '10-14 years old': {'population_percentage': 0.068185819, 'milk_intake': 1150.0/100000.0},
    '15-19 years old': {'population_percentage': 0.067162943, 'milk_intake': 1300.0/100000.0},
    '20-24 years old': {'population_percentage': 0.065940194, 'milk_intake': 1000.0/100000.0},
    '25-29 years old': {'population_percentage': 0.064764473, 'milk_intake': 1000.0/100000.0},
    '30-34 years old': {'population_percentage': 0.071290026, 'milk_intake': 1000.0/100000.0},
    '35-39 years old': {'population_percentage': 0.072479947, 'milk_intake': 1000.0/100000.0},
    '40-44 years old': {'population_percentage': 0.073774002, 'milk_intake': 1000.0/100000.0},
    '45-49 years old': {'population_percentage': 0.072563774, 'milk_intake': 1000.0/100000.0},
    '50-54 years old': {'population_percentage': 0.065520141,  'milk_intake': 1300.0/100000.0},
    '55-59 years old': {'population_percentage': 0.061369696, 'milk_intake': 1300.0/100000.0},
    '60-64 years old': {'population_percentage': 0.04849846, 'milk_intake': 1300.0/100000.0},
    '65-69 years old': {'population_percentage': 0.03884824, 'milk_intake': 1300.0/100000.0},
    '70 and above': {'population_percentage': 0.099455397, 'milk_intake': 1300.0/100000.0}
}

demand_2006 = []

for lga_index, lga_row in NSW_2006.iterrows():
    lga_name = lga_row['LGA name']

    # Initialize cumulative demand for the LGA to 0
    lga_cumulative_demand = 0.0

    # Iterate over age groups and calculate the demand for the LGA
    for age_group, data in age_group_data_2006.items():
        population_percentage = data['population_percentage']
        milk_intake = data['milk_intake']

        # Calculate the demand for age group in the LGA
        age_group_demand = lga_row['Total persons'] * population_percentage * milk_intake

        # Add the age group demand to the cumulative demand for the LGA
        lga_cumulative_demand += age_group_demand

    # Append the cumulative demand for the LGA to the lga_demand list
    demand_2006.append((lga_name, (lga_cumulative_demand)))
```

In [27]:
```
model_2006 = gp.Model("2006")
```

```
# Decision Variables
# x = whether to establish a center at LGA
x = model_2006.addVars(LGA, LGA, vtype=GRB.BINARY, name="x")
# amount of milk distributed
y = model_2006.addVars(LGA, vtype=GRB.CONTINUOUS, name="y")

# Objective Function
obj_2006 = (
    gp.quicksum(distance[i, j] * demand_2006[LGA.index(i)][1] * 0.1 for i in LGA for j in LGA)
    + gp.quicksum(y[i] * 1000 for i in LGA)
)

model_2006.setObjective(obj_2006, GRB.MINIMIZE)

# Constraints
# Constraint 1: Maximum 3 distribution centers
model_2006.addConstr(gp.quicksum(x[i, j] for i in LGA) <= 3, name="Max_Centers")

# Constraint 2: Center capacity limit
model_2006.addConstrs(y[i] <= 0.5 * gp.quicksum(demand_2006[LGA.index(j)][1] for j in LGA) for i in LGA)

# Constraint 3: Demand fulfillment
model_2006.addConstrs(gp.quicksum(x[i, j] for i in LGA if i != j) >= 1 for j in LGA)

# Constraint 4: linking constraints
model_2006.addConstrs((x[i, j] <= y[i] for i in LGA for j in LGA))

# Optimize the model
model_2006.update()
model_2006.optimize()
```

```
Gurobi Optimizer version 10.0.2 build v10.0.2rc0 (mac64[x86])

CPU model: Intel(R) Core(TM) i5-5350U CPU @ 1.80GHz
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads

Optimize a model with 16900 rows, 16770 columns and 50052 nonzeros
Model fingerprint: 0x92e581ef
Variable types: 129 continuous, 16641 integer (16641 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+03, 1e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+04]
Found heuristic solution: objective 2.529270e+08
Presolve removed 258 rows and 129 columns
Presolve time: 0.25s
Presolved: 16642 rows, 16641 columns, 49664 nonzeros
Variable types: 0 continuous, 16641 integer (16641 binary)

Root relaxation: objective 2.528450e+08, 16641 iterations, 0.51 seconds (0.42 work units)

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

     0     0 2.5284e+08    0 16641 2.5293e+08 2.5284e+08  0.03%     -    1s
H    0     0                     2.529260e+08 2.5284e+08  0.03%     -    1s
H    0     0                     2.528460e+08 2.5284e+08  0.00%     -    1s

Explored 1 nodes (16641 simplex iterations) in 1.13 seconds (0.73 work units)
Thread count was 4 (of 4 available processors)

Solution count 3: 2.52846e+08 2.52926e+08 2.52927e+08

Optimal solution found (tolerance 1.00e-04)
Best objective 2.528459816598e+08, best bound 2.528459816598e+08, gap 0.0000%
```

In [28]:

```
# Display results
if model_2006.status == GRB.OPTIMAL:
    print("Selected distribution centers:")
    for i in LGA:
        if y[i].x > 0.5:
            print(f"Distribution center at {i}")

    minimal_cost_2006 = model_2006.objVal
    print(f"Minimal Cost: $ {minimal_cost_2006:.2f}")

population_change_2001_2006 = sum(NSW_2006['Total persons']) / sum(NSW_2001['Total persons'])
print(f'Population change 2001-2006: {population_change_2001_2006:.3f}')
```

```
Selected distribution centers:
Distribution center at Warrumbungle
Distribution center at Yass Valley
Minimal Cost: $ 252845981.66
Population change 2001-2006: 1.033
```

Comments:

Comparing the results from 2002, 2003, 2004 (an anticipation 10% population growth) with the results gained from 2006 (real data), some insights gained:

1. Optimal distribution centres

- The 2001 model determined that Warrumbungle and Yass Valley were the best distribution centres, with an associated least cost of 240,615,430.02.
- Warrumbungle and Yass Valley were regularly chosen as distribution centres in the models for the predicted increase of 10% population growth per annum for 2002, 2003, and 2004.
- Warrumbungle and Yass Valley were still the chosen distribution centres for 2006, which is based on a real data.

1. Cost comparison

- The projected costs for 2002 (323430401.39), 2003 (355773241.53), and 2004 (391350365.69) indicate rising minimum costs over time, which represent the expense of adjusting for the predicted 10% increase in population growth.
- It is interesting to see that the costs for distributing milk in 2006 (352835708.82) are lower than the projected year 2002-2004.

1. population growth

- The interesting insights gained from the cost comparison can be explained by the population growth.
- Year 2002-2004 were based on an speculation growth of population by 10% per-annum, which may or may not be true.
- Year 2006 was based on the real data obtained from Australian Bureau of Statistics.
- It seems like the speculation for year 2002-2004 is not as accurate as the real data.
- According the population data obtained from Australian Bureau Statistics the populatio growth between 2001-2006 was only 3.3% instead 10% per annum, which could explain the lower distributing costs.

Overall:

- The government did a good job in anticipating populaton growth to accomodate the capacity of distribution centres.
- The opitmisation model continuously select Warrumbungle and Yass Valley, which indicates the stability of these distribution facilities.

Part 5 (2021)

In [29]:
```
# I created a dictionary to map age groups to population percentages and RDI
age_group_data_2021 = {
    '0-4 years old': {'population_percentage': 0.057983963, 'milk_intake': 500/100000},
    '5-9 years old': {'population_percentage': 0.062041611, 'milk_intake': 700/100000},
    '10-14 years old': {'population_percentage': 0.062081873, 'milk_intake': 1150/100000},
    '15-19 years old': {'population_percentage': 0.056725316, 'milk_intake': 1300/100000},
    '20-24 years old': {'population_percentage': 0.061468655, 'milk_intake': 1000/100000},
    '25-29 years old': {'population_percentage': 0.0688746, 'milk_intake': 1000/100000},
    '30-34 years old': {'population_percentage': 0.072602226, 'milk_intake': 1000/100000},
    '35-39 years old': {'population_percentage': 0.071874787, 'milk_intake': 1000/100000},
    '40-44 years old': {'population_percentage': 0.064788583, 'milk_intake': 1000/100000},
    '45-49 years old': {'population_percentage': 0.06403674, 'milk_intake': 1000/100000},
    '50-54 years old': {'population_percentage': 0.061944611, 'milk_intake': 1300/100000},
    '55-59 years old': {'population_percentage': 0.060721643, 'milk_intake': 1300/100000},
    '60-64 years old': {'population_percentage': 0.058426471, 'milk_intake': 1300/100000},
    '65-69 years old': {'population_percentage': 0.051596208, 'milk_intake': 1300/100000},
    '70 and above': {'population_percentage': 0.124832712, 'milk_intake': 1300/100000}
}


#calculate the total demand for each LGA

demand_2021 = []

for lga_index, lga_row in NSW_2021.iterrows():
    lga_name = lga_row['LGA name']

    # Initialize cumulative demand for the LGA to 0
    lga_cumulative_demand = 0.0

    # Iterate over age groups and calculate the demand for the LGA
    for age_group, data in age_group_data_2021.items():
        population_percentage = data['population_percentage']
        milk_intake = data['milk_intake']

        # Calculate the demand for age group in the LGA
        age_group_demand = lga_row['Total persons'] * population_percentage * milk_intake

        # Add the age group demand to the cumulative demand for the LGA
        lga_cumulative_demand += age_group_demand

    # Append the cumulative demand for the LGA to the lga_demand list
    demand_2021.append((lga_name, lga_cumulative_demand))
```

```
demand_2021

model_2021 = gp.Model("2021")

# Decision Variables
# x = whether to establish a center at LGA
x = model_2021.addVars(LGA, LGA, vtype=GRB.BINARY, name="x")
# amount of milk distributed
y = model_2021.addVars(LGA, vtype=GRB.CONTINUOUS, name="y")


# Objective Function
obj_2021 = (
    gp.quicksum(distance[i, j]  * demand_2021[LGA.index(i)][1]  * 0.1 for i in LGA for j in LGA)
    + gp.quicksum(y[i] * 1000 for i in LGA)
)

model_2021.setObjective(obj_2021, GRB.MINIMIZE)

# Constraints
# Constraint 1: Maximum 3 distribution centers
model_2021.addConstr(gp.quicksum(x[i,j] for i in LGA) <= 3, name="Max_Centers")

# Constraint 2: Center capacity limit
model_2021.addConstrs(y[i] <= 0.5 * gp.quicksum(demand_2021[LGA.index(j)][1] for j in LGA) for i in LGA)

# Constraint 3: Demand fulfillment
model_2021.addConstrs(gp.quicksum(x[i, j] for i in LGA if i != j) >= 1 for j in LGA)

# constraint 4: linking constraints
model_2021.addConstrs((x[i, j] <= y[i] for i in LGA for j in LGA))

# Optimize the model
model_2021.update()
model_2021.optimize()
```

```
Gurobi Optimizer version 10.0.2 build v10.0.2rc0 (mac64[x86])

CPU model: Intel(R) Core(TM) i5-5350U CPU @ 1.80GHz
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads

Optimize a model with 16900 rows, 16770 columns and 50052 nonzeros
Model fingerprint: 0x9f795922
Variable types: 129 continuous, 16641 integer (16641 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+03, 1e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+04]
Found heuristic solution: objective 3.076435e+08
Presolve removed 258 rows and 129 columns
Presolve time: 0.29s
Presolved: 16642 rows, 16641 columns, 49664 nonzeros
Variable types: 0 continuous, 16641 integer (16641 binary)

Root relaxation: objective 3.075615e+08, 16641 iterations, 0.57 seconds (0.42 work units)

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

     0     0 3.0756e+08    0 16641 3.0764e+08 3.0756e+08  0.03%     -    1s
H    0     0                    3.076425e+08 3.0756e+08  0.03%     -    1s
H    0     0                    3.075625e+08 3.0756e+08  0.00%     -    1s

Explored 1 nodes (16641 simplex iterations) in 1.20 seconds (0.73 work units)
Thread count was 4 (of 4 available processors)

Solution count 3: 3.07563e+08 3.07643e+08 3.07644e+08

Optimal solution found (tolerance 1.00e-04)
Best objective 3.075625099140e+08, best bound 3.075625099140e+08, gap 0.0000%
```

In [30]:
```
# Display results
if model_2021.status == GRB.OPTIMAL:
    print("Selected distribution centers:")
    for i in LGA:
        if y[i].x > 0.5:
            print(f"Distribution center at {i}")

    minimal_cost_2021 = model_2021.objVal
    print(f"Minimal Cost: $ {minimal_cost_2021:.2f}")
    print(f"Minimal Cost (inflation): $ {minimal_cost_2021*1.556:.2f}") #accounting for inflation of 55.6%

population_change_2001_2021 = sum(NSW_2021['Total persons']) / sum(NSW_2001['Total persons'])
print(f'Population change 2001-2021: {population_change_2001_2021:.3f}')
```

```
Selected distribution centers:
Distribution center at Warrumbungle
Distribution center at Yass Valley
Minimal Cost: $ 307562509.91
Minimal Cost (inflation): $ 478567265.43
Population change 2001-2021: 1.240
```

Comment:

Comparing the results from 2001, 2002-2004 (an anticipation 10% population growth), and 2006 (real data) with the results gained from 2021 (real data), some insights gained:

1. Optimal distrbution centre

- The model still chooses the same locations for the distribution centres: Warrumbungle and Yass Valley.
- This demonstrate that the capacity of distribution centres are able to handle the growing demands after 20 years, which is the government's goal wehn building the distribution centre in 2001.

1. Cost and population comparison

- As anticipated, the minimum cost increased to 307,562,509.91 in 2021. This rise can be linked to the increased demand for milk brought on by a 20-year increase in population.
- The distribution cost for 2021 is similar to 2003's anticipated costs.
- This happens because the actual population growth from 2001-2021 is approximately 24%, whereas 2003's population is only a prediction.

1. inflation impact

- The minimal cost increased to 478567265.43 after accounting for inflation. This significant rise underlines the need of taking into account how inflation would eventually reduce the buying power of a certain quantity of resources.

Overall:

- Costs significantly rise as a result of both population growth and inflation. When each of these variables were taken into account, the initial minimal cost of 240615430.02 in 2001 grew by around 98.6% during the subsequent 20 years. This shows how these impacts have a cumulative effect on the expenditure needed to maintain the distribution centres.
- The model's assessment of the costs of distribution centres over a 20-year period, which takes population growth and inflation into account, shows the need for ongoing assessments and adjustments. Governments and organisations must adopt a proactive strategy for infrastructure planning, resource allocation, and demand forecasting if they want to keep their distribution networks stable and efficient.

References

References Australian Bureau of Statistics. (2022, July 9). *Regionalpopulationbyageandsex*, 2021 | *AustralianBureauofStatistics*. Www.abs.gov.au. https://www.abs.gov.au/statistics/people/population/regional-population-age-and-sex/latest-release#data-downloads