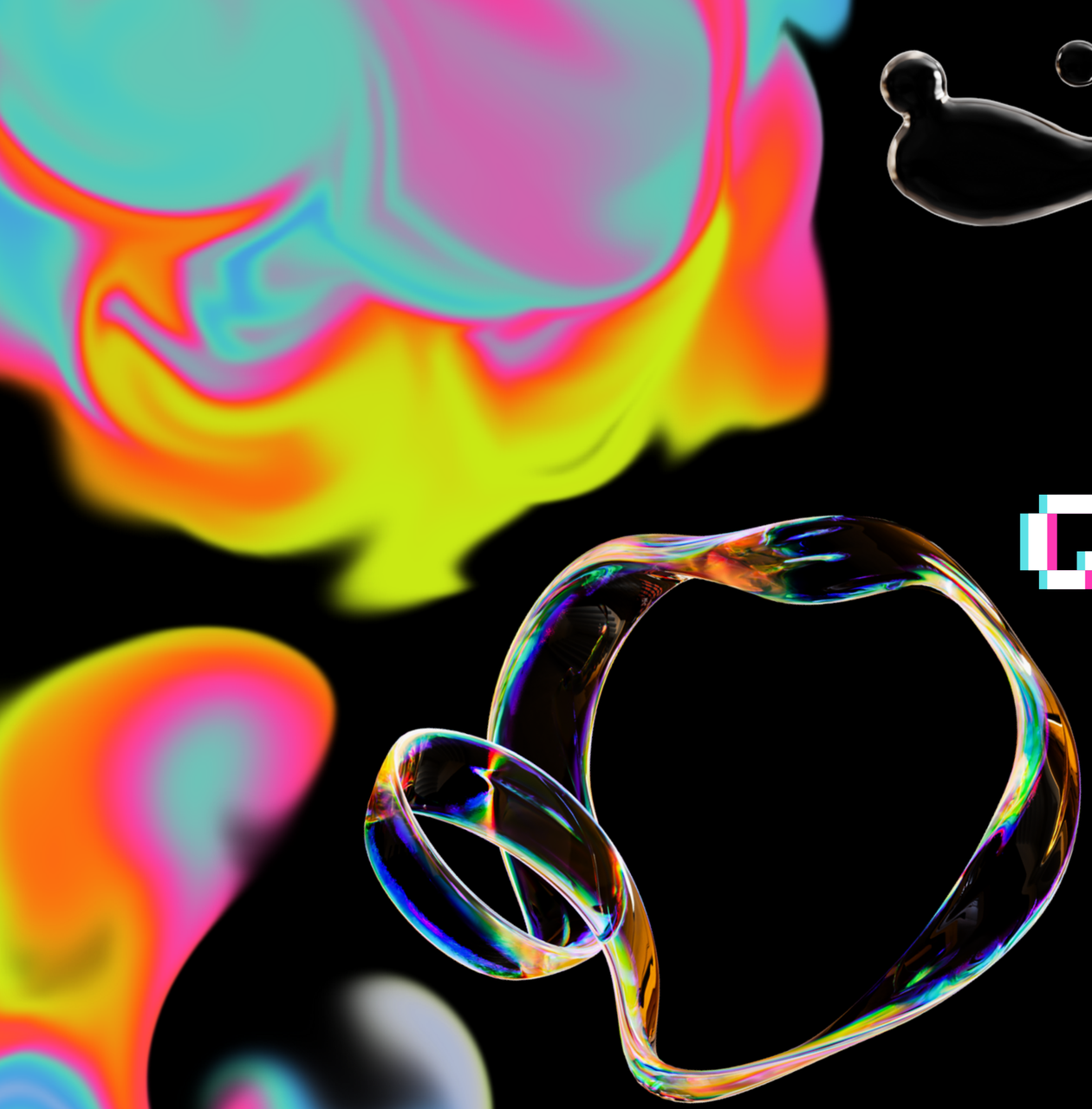


HACKATON QUANTUM IBM

HAPEN TEAM

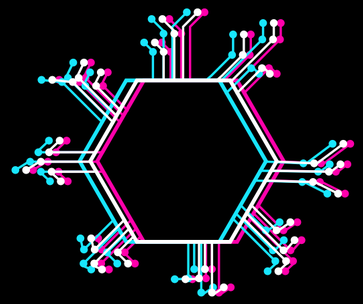
MARÍA DEL PILAR DÁVILA VERDUSCO - A01708943
ALEJANDRO LEMUS SALGADO - A01770848
JESUS EMERY BAUTISTA CEBALLOS - A01771304
VICTOR NOEL MADRID CASTILLO - A01562528
HOSSUE EDGARDO CEJA CARTAGENA - A01707761

WHAT'S THE CHALLENGE?

A delivery company operates a fleet of vehicles and serves a diverse customer base located in various regions for package delivery

OUR MISION?

Determine the collective shortest travel distance for all the trucks in order to minimize the overall cost and time.



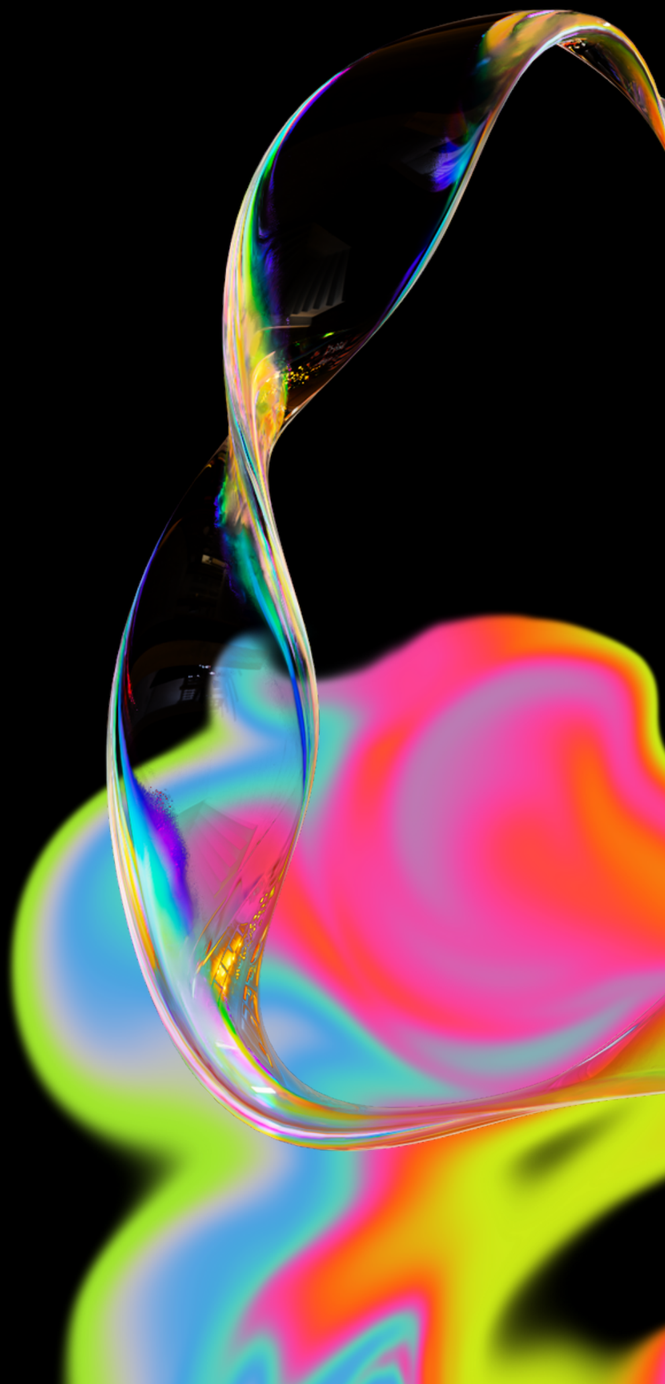
We should employ combinatorial optimization techniques to accomplish this objective.

PROBLEM MODELING

Model the shortest route problem for the trucks as a mathematical optimization problem.

We define:

- Decision variables
- Restrictions
- The objective function to be minimized (**total cost and time**)



INTO THE MODELING

For each truck, create a binary variable representing whether the truck is at a specific location at a given time. If we have N locations and T time moments, we will have N * T binary variables. These variables can be denoted as x(i, j, t), where:

- i represents the location.
- j represents the truck.
- t represents the time moment.

1.-Each truck must be at a single location at any given time: $\sum_{i=1}^{N} x(i,j,t) = 1$

2.-Each location must be visited exactly once by each truck: $\sum_{j=1}^{J} x(i,j,t) = 1$

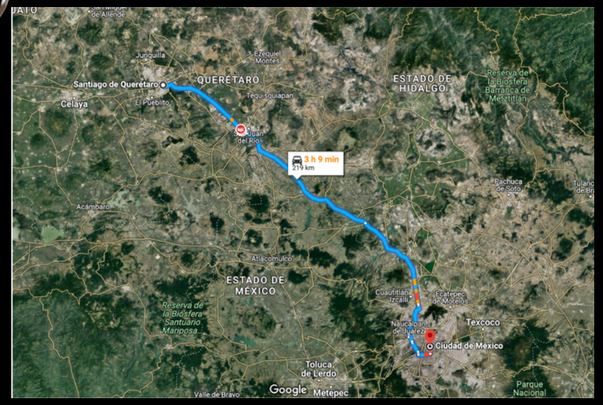
Finally, the objective function will aim to minimize both the total route length and the total cost, which can be a weighted combination of both metrics:

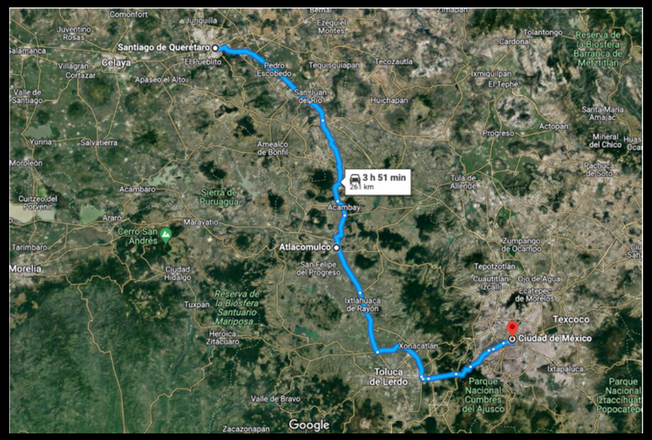
$$\sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{i=1}^{N} c(i,j,t) \cdot x(i,j,t)$$

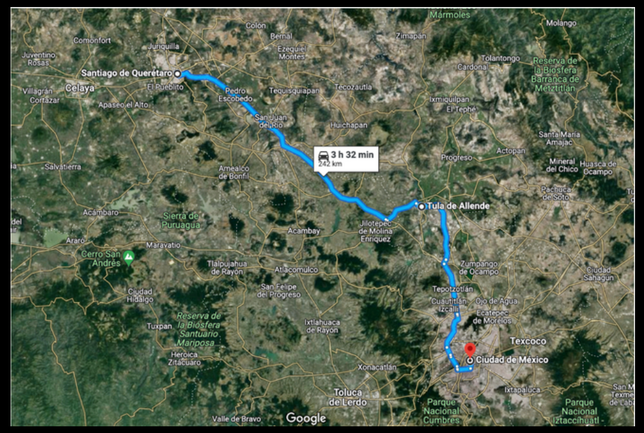
$c(i,j,t)$ Represents the cost associated with location i, truck j, and time moment t

WERE ARE WE?

We want to send three trucks loaded with packages from Mexico City to Santiago de Queretaro, Queretaro. We are considering two modes of transportation: air and land. Due to the large-scale concert event, we have chosen to transport them by land to minimize the cost, as using air transport for a roughly 20-minute flight would be expensive.







TRANSLATION INTO A QUANTUM FORMULATION

```python
# Parámetros de tu problema de optimización
from qiskit import QuantumCircuit
from qiskit.circuit import Parameter

def ansatz_circuit(params):
    num_locations = 4
    num_trucks = 2

    # Crear un circuito cuántico
    circuit = QuantumCircuit(num_locations + num_trucks)

    # Parámetros de rotación para ajustar la función de onda
    # Aquí, usamos los parámetros theta para definir las rotaciones
    theta_params = [Parameter(f'theta_{i}') for i in range(len(params))]

    # Aplicar compuertas de rotación X controladas por los parámetros
    for i in range(num_locations):
        circuit.rx(theta_params[i], i)

    # Aplicar compuertas de rotación Y controladas por los parámetros
    for i in range(num_locations, num_locations + num_trucks):
        circuit.ry(theta_params[i], i)
```

```python
    # Entrelazar los qubits de camiones y ubicaciones según las restricciones

    # Devuelve el circuito
    return circuit


# Construye un operador de Pauli ponderado basado en la función objetivo.
# Por ejemplo, puedes usar max_cut para problemas de corte máximo.
operator, offset = max_cut.get_operator(my_cost_matrix)

# Ahora, construye un circuito cuántico
num_qubits = operator.num_qubits
circuit = QuantumCircuit(num_qubits)

# La función de onda ansatz.
from qiskit import QuantumCircuit
from qiskit.circuit import Parameter

def ansatz_circuit(params):
    num_locations = 4
    num_trucks = 2
```

```python
    # Crear un circuito cuántico
    circuit = QuantumCircuit(num_locations + num_trucks)

    # Parámetros de rotación para ajustar la función de onda
    # Aquí, usamos los parámetros theta para definir las rotaciones
    theta_params = [Parameter(f'theta_{i}') for i in range(len(params))]

    # Aplicar compuertas de rotación X controladas por los parámetros
    for i in range(num_locations):
        circuit.rx(theta_params[i], i)

    # Aplicar compuertas de rotación Y controladas por los parámetros
    for i in range(num_locations, num_locations + num_trucks):
        circuit.ry(theta_params[i], i)

    # Entrelazar los qubits de camiones y ubicaciones según las restricciones

    # Devuelve el circuito
    return circuit
```

```python
# Configura y ejecuta el algoritmo de optimización cuántica (por ejemplo, VQE).
optimizer = COBYLA(maxiter=100)
algo = VQE(operator, optimizer)
result = algo.compute_minimum_eigenvalue()

# Imprime los resultados
print("Energía mínima:", result.eigenvalue.real + offset)
print("Solución óptima:", max_cut.sample_most_likely(result.eigenstate))
```

RESULTS ANALYSIS

OUR CONCLUSIONS ...