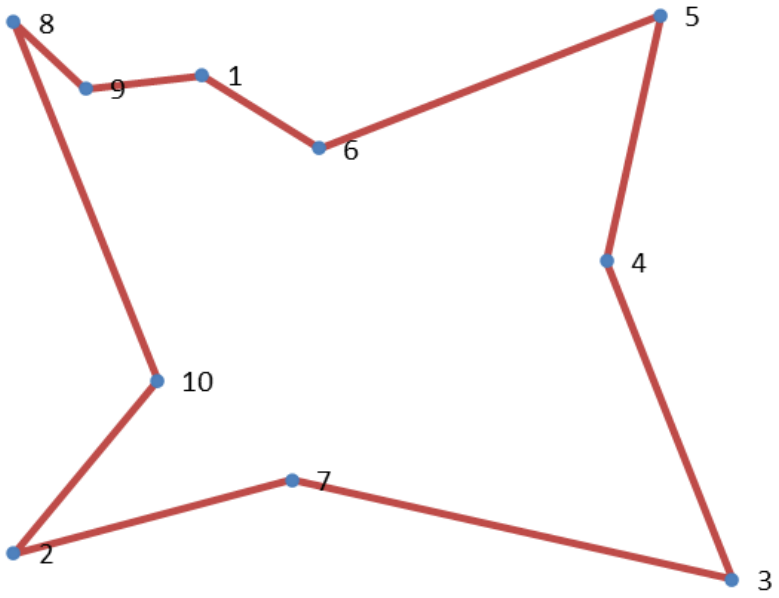
Modelización Matemática y Computacional del Problema de Ruteo de Vehículos

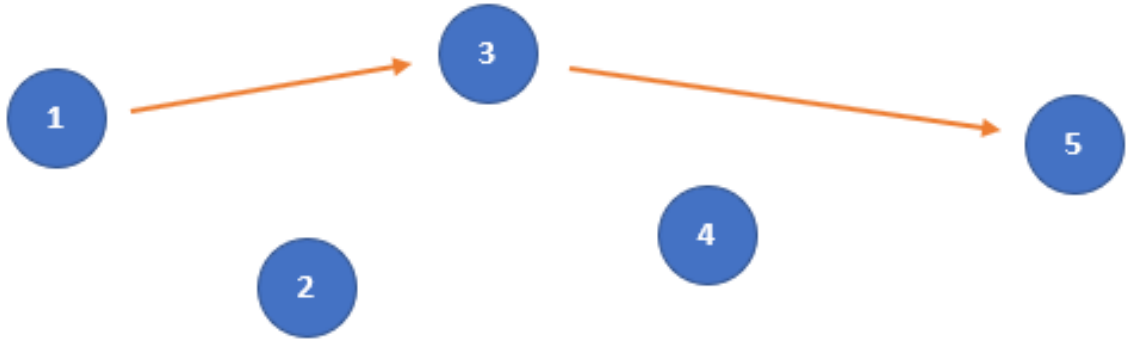
Rodrigo Maranzana

Logística

UTN BA

Camino más corto vs. Ruteo





¿Son problemas similares matemáticamente?

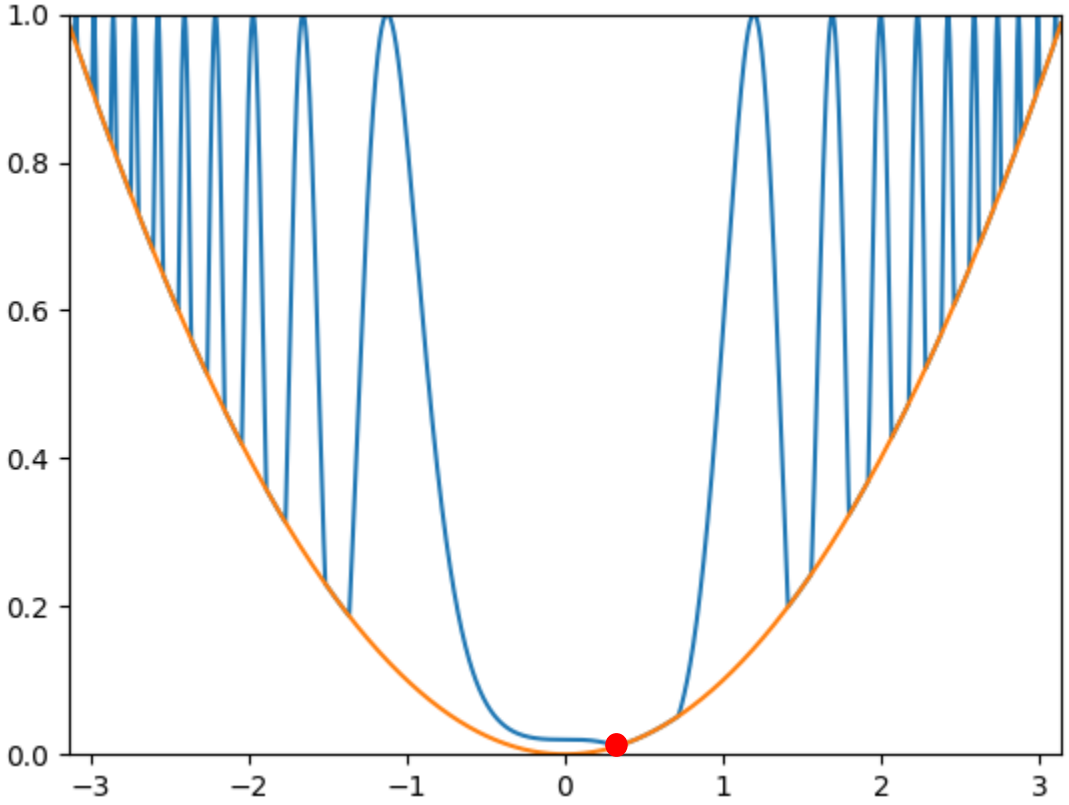
¿Son simples de resolver?

¿Se pueden resolver con SIMPLEX?

Repaso de Modelos de Optimización

Min $sin(X^3 + 3)^2$

s.t $objetivo \geq 0.1 * X^2$



El problema del viajante de comercio: TSP (Ruteo con 1 vehículo)

$$\min \sum_{ij \in A} c_{ij} x_{ij}$$

Min costo de recorrer todo el circuito

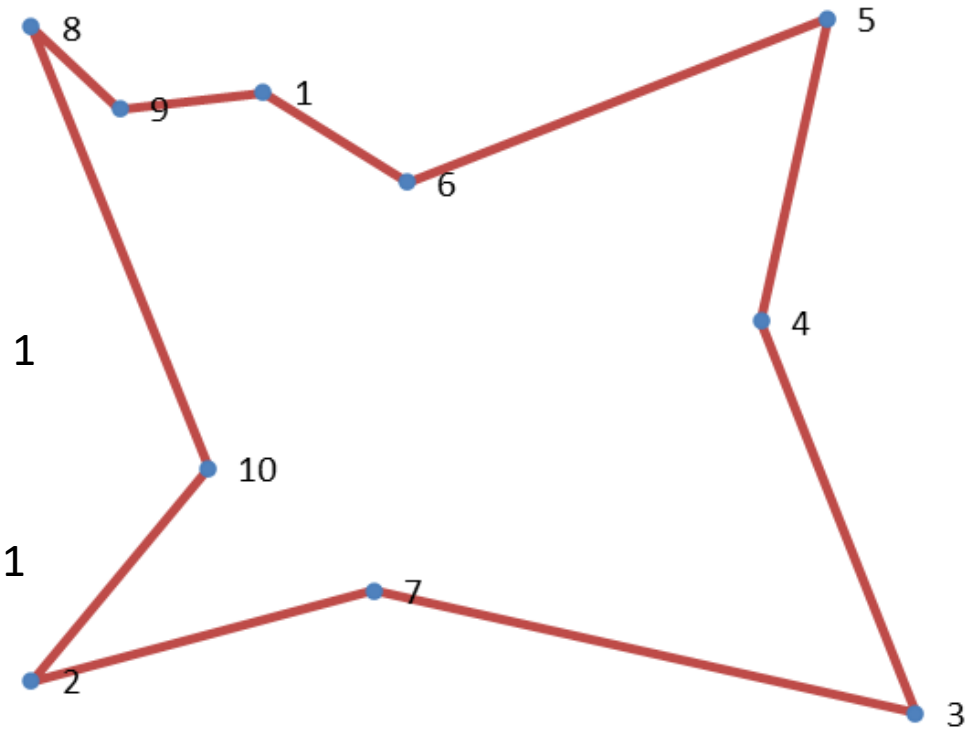
$$\sum_{j} x_{ij} = 1$$

Suma de arcos salientes de cada nodo i = 1

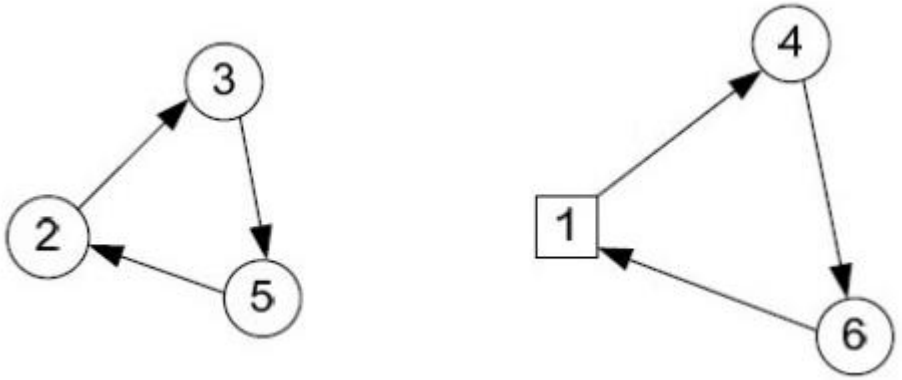
$$\sum_{i} x_{ij} = 1$$

Suma de arcos entrantes a cada nodo j = 1

$$x_{ij} \geq 0 \quad \forall ij \in A$$



El problema de los Subtours



Una aproximación para la eliminación de Subtours (Dantzig):

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1$$

Para todos los subconjuntos S que no contienen al 1 (depósito).
Restricciones crecen con 2^n

¿Cómo lo implementamos?

Software de Optimización con programación matemática:

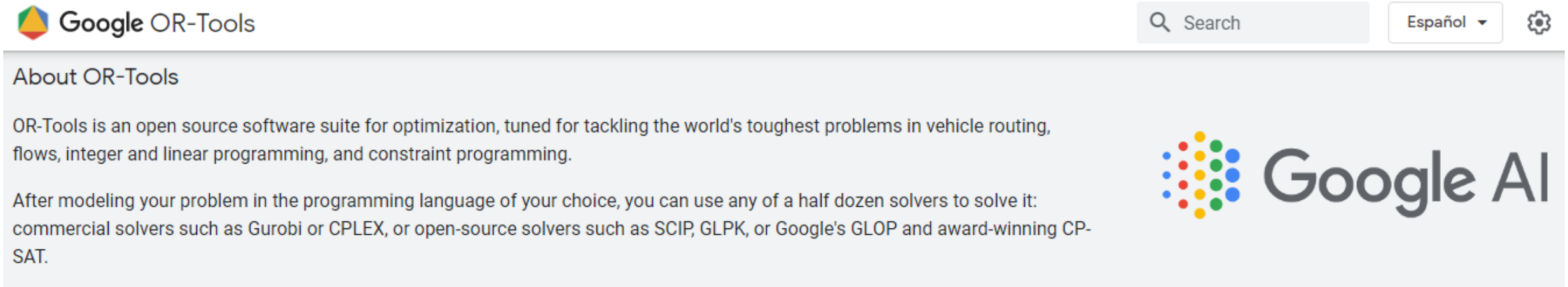
Interfaz:

- Python – scipy.optimize.linprog / PuLP
- MATLAB – LinProg / IntLinProg
- Julia – Jump.jl
- GAMS
- IBM ILOG
- Gusek
- Excel

Solvers:

- Gurobi
- CPLEX
- GLPK
- Xpress
- COIN-OR (Cbc)
- Clp
- Mosek

Ejemplo de paquete específico:



Ejemplo de Modelo Matemático en PuLP de Python (Programación matemática)

$$Max\ Z = (x - 45) + (y - 5)$$

sujeto a:

$$50x + 24y \leq 2400$$

$$30x + 33y \leq 2100$$

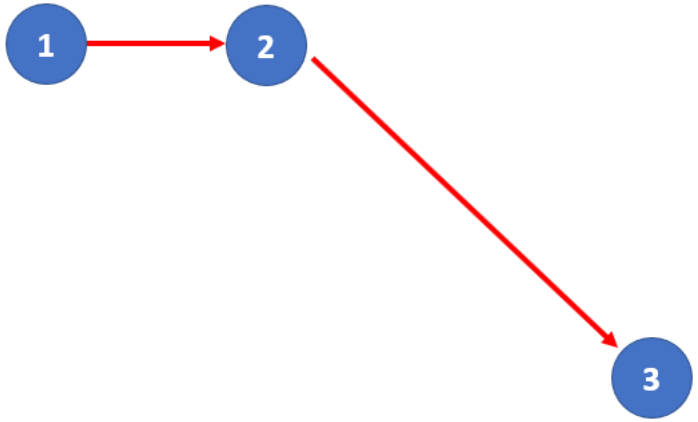
$$x \geq 45$$

$$y \geq 5$$

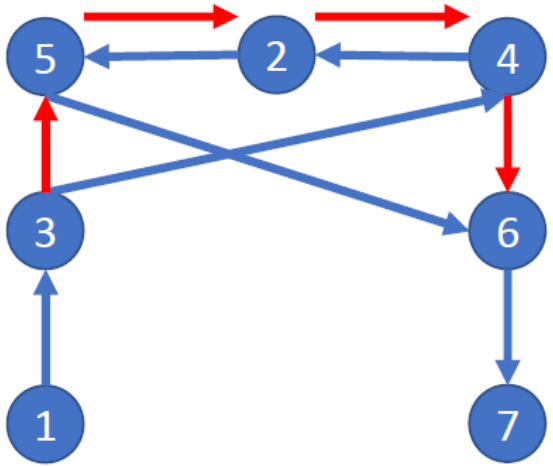
```
Optimal
x = 45.00
y = 6.25
1.25
```

```python
# Problema de producción
import pulp

lp01 = pulp.LpProblem("problema producción", pulp.LpMaximize)

# Variables:
x = pulp.LpVariable('x', lowBound=45, cat='Continuous')
y = pulp.LpVariable('y', lowBound=5, cat='Continuous')

# Función objetivo:
lp01 += (x - 45)  + (y - 5), "Z"

# Restricciones:
lp01 +=  50*x + 24*y <= 2400
lp01 +=  30*x + 33*y <= 2100

lp01.solve()
pulp.LpStatus[lp01.status]
print(pulp.LpStatus[lp01.status])

for variable in lp01.variables():
    print("%s = %.2f" % (variable.name, variable.varValue))
print(pulp.value(lp01.objective))
```

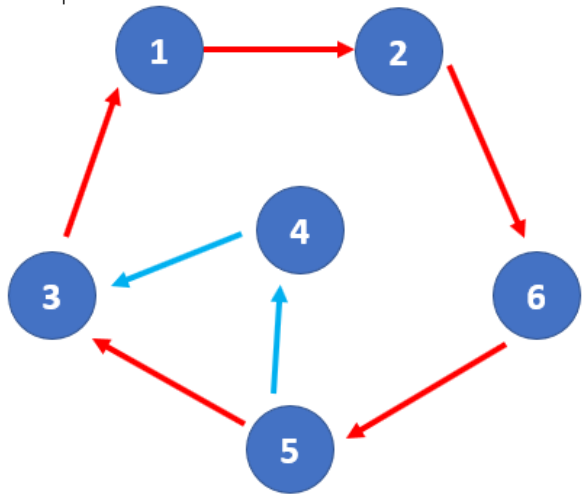
Resoluciones heurísticas:



Vecinos más cercanos



2-Opt

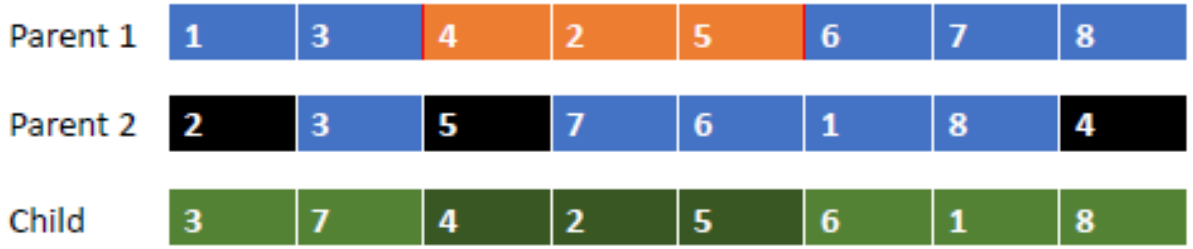


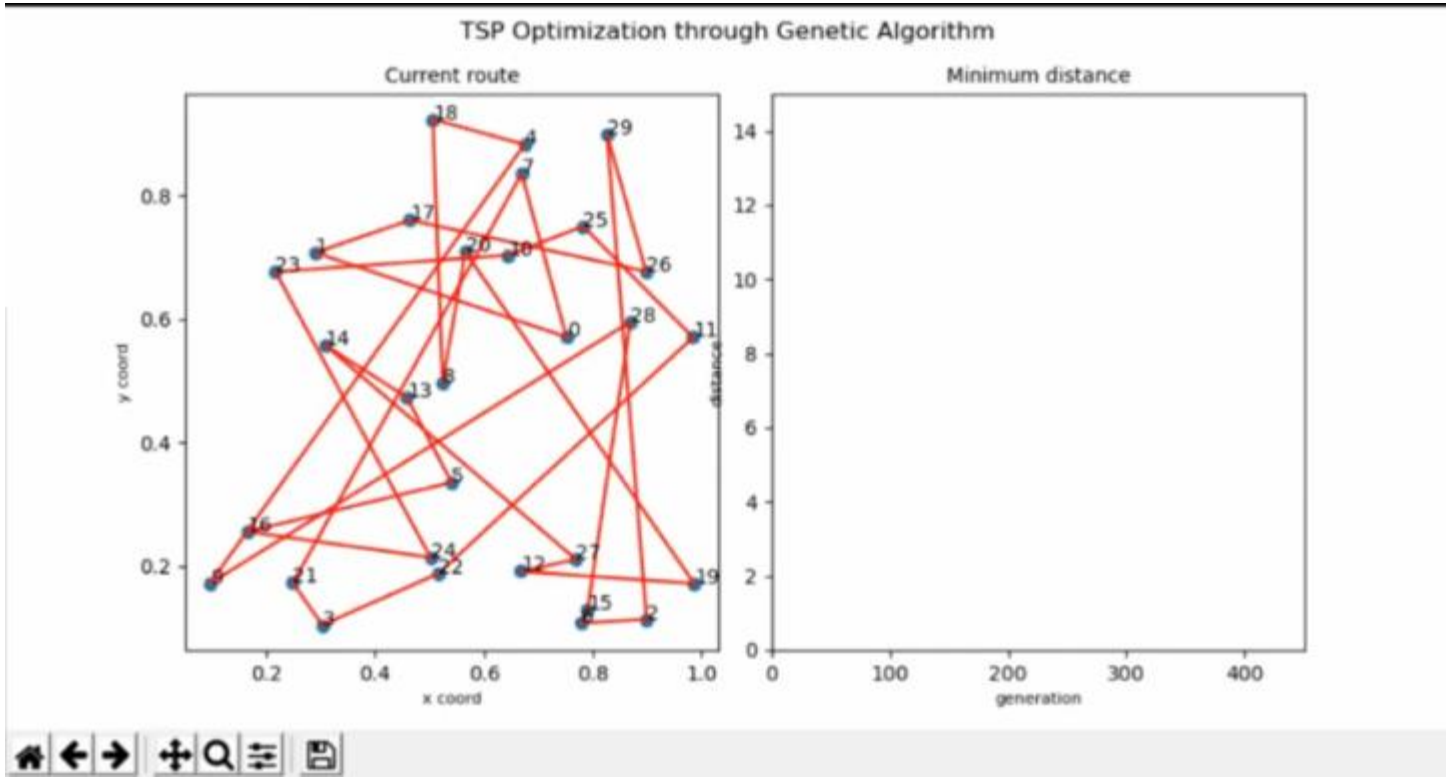
Convex Hull

Resoluciones heurísticas:

Algoritmo Genético

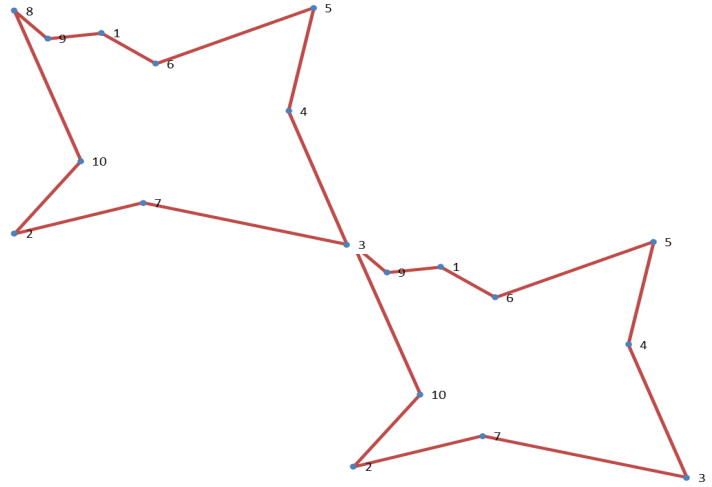
- Crear población Random
- Elegir mejores padres de acuerdo al costo mínimo
- Cruza de padres
- Selección de elite
- Iteración





El problema de Ruteo (TSP con N vehículos):

Variable y: arcos activados
Variable x: arcos asignados a camión k



$$\min \sum_{1\le k\le K} \sum_{(i,j)\in A} c_{ij}x_{ij}^{k}$$

Mínimo costo de recorrer todo el circuito para los k vehículos

$$\sum_{1\le k\le K} x_{ij}^{k} = y_{ij},$$

Cada nodo solo puede asignarse a 1 camión

$$\sum_{1\le j\le N} y_{ij} = 1,$$

Suma de arcos salientes de cada nodo i = 1

$$\sum_{1\le i\le N} y_{ij} = 1,$$

Suma de arcos entrantes a cada nodo j = 1

$$\sum_{1\le j\le N} y_{1j} = \sum_{1\le j\le N} y_{j1} = K,$$

El nodo 1 es el único permitido de tener K entrantes y K salientes.

$$\sum_{2\le i\le N} \sum_{1\le j\le N} d_i x_{ij}^{k} \le C,$$

Existe una capacidad máxima para cada recorrido (camión).
Hay una demanda d a cumplir en cada nodo.

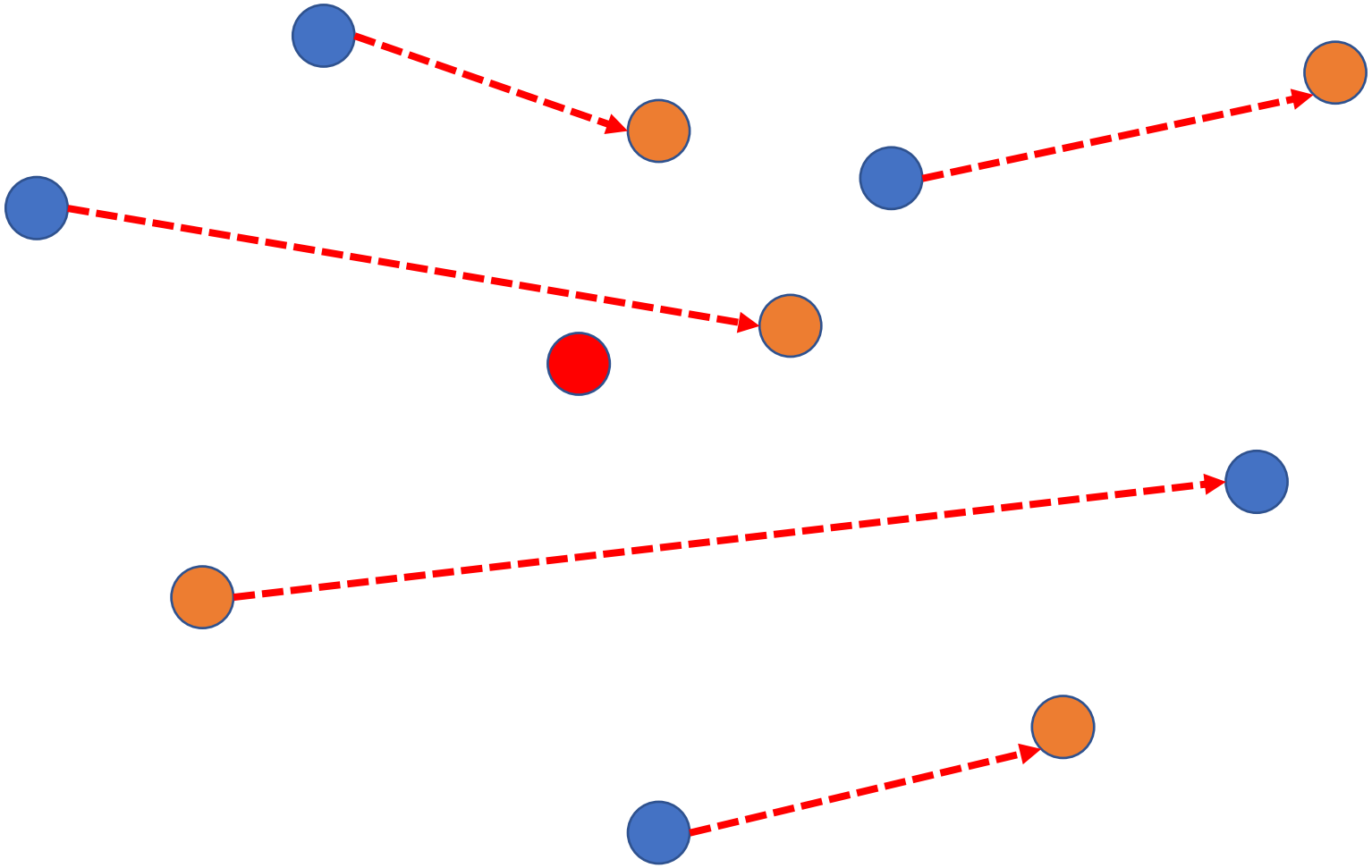
$$\sum_{i,j\in S} y_{ij} \le |S| - 1$$

Eliminación de subtours

→ *Capacitated Vehicle Routing Problem (CVRP)*

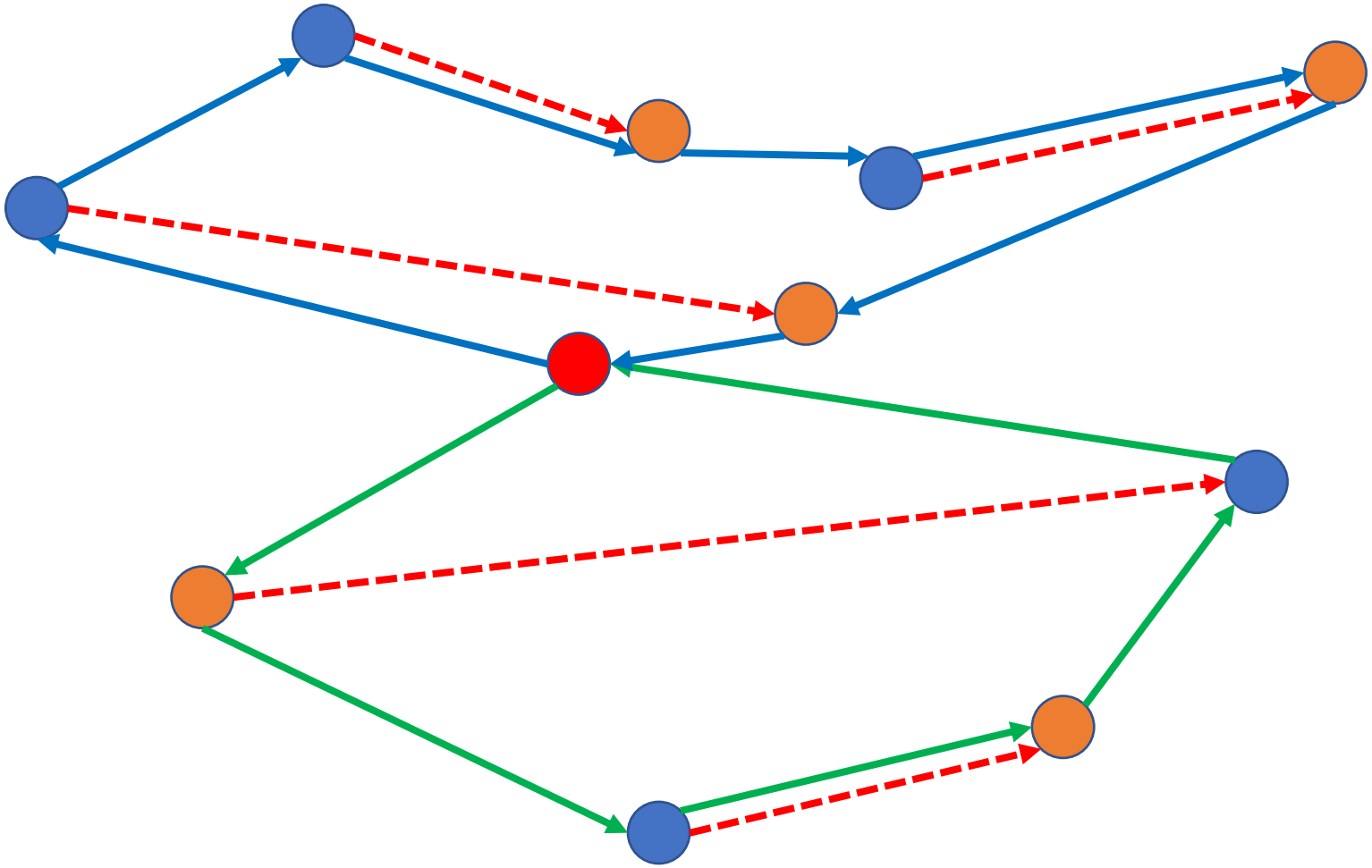
Ruteo con puntos de recogida y entrega

(Vehicle Routing With Pickups and Deliveries)



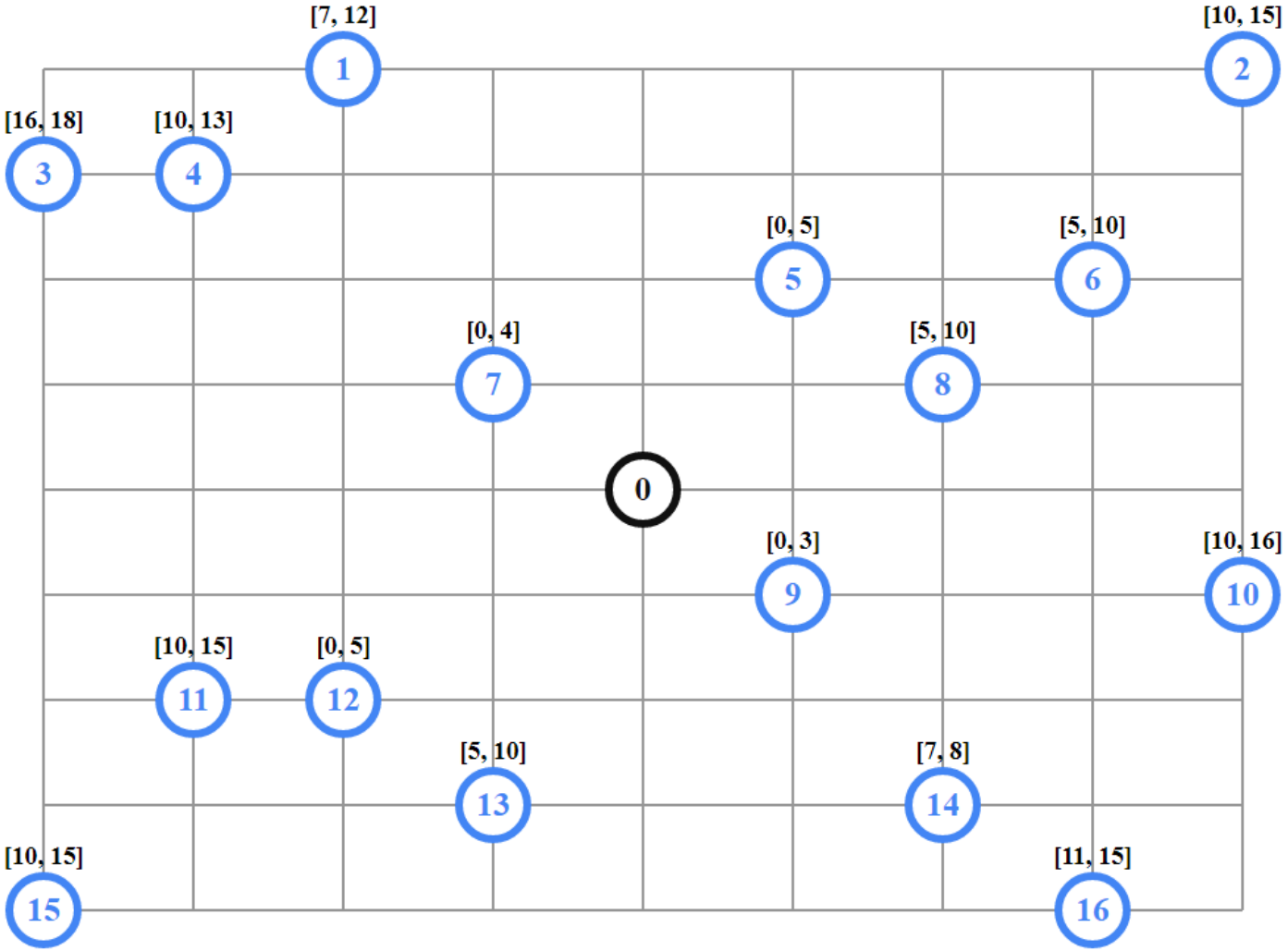
Ruteo con puntos de recogida y entrega

(Vehicle Routing With Pickups and Deliveries)



Ruteo con ventanas de tiempo

(Vehicle Routing With Time Windows - VRPTW)

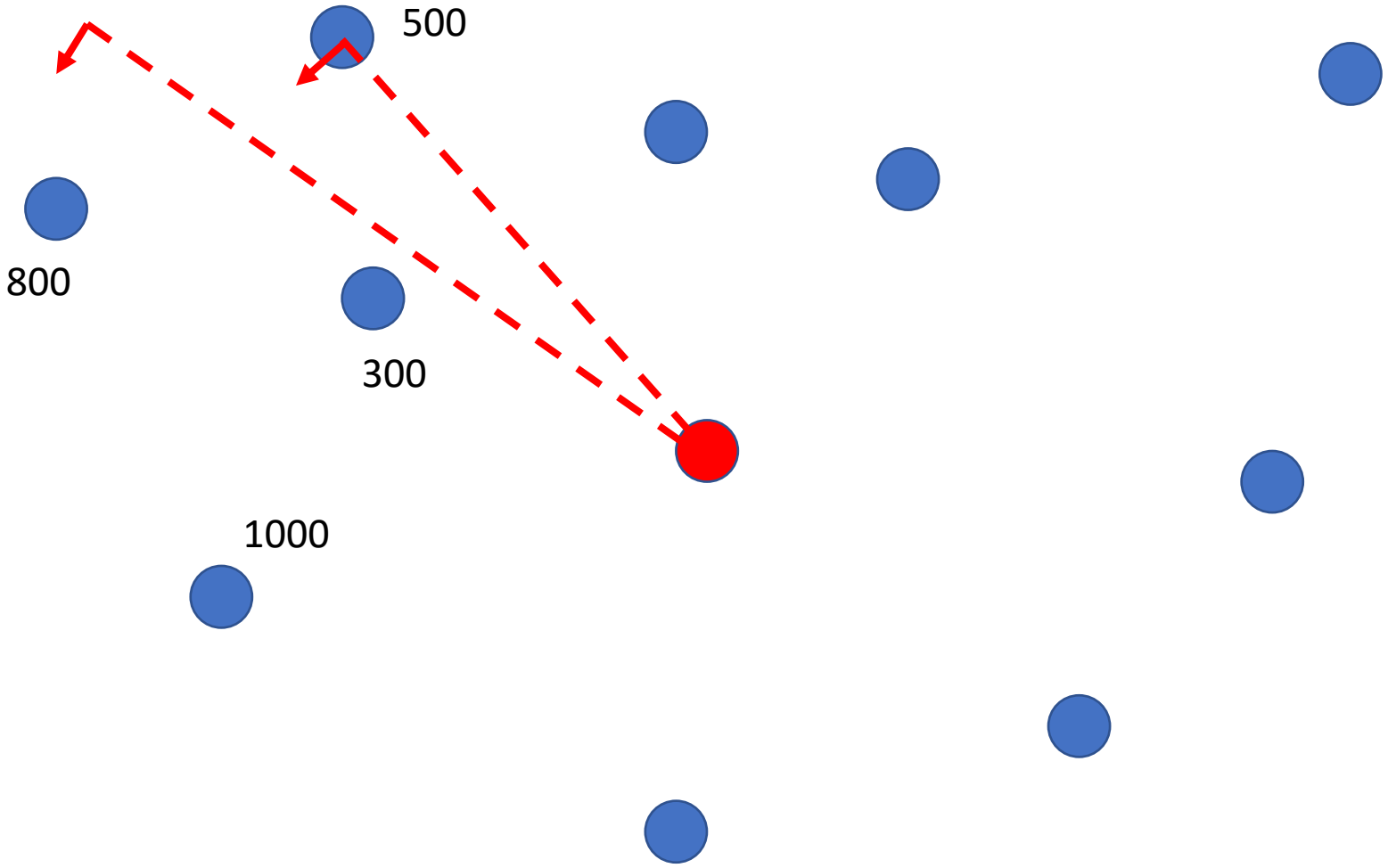


Ejemplo de **Google OR-Tools** (https://developers.google.com/optimization/routing/vrptw)

Más información y modelos: *Nasser A. El-Sherbeny (2010), Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods*

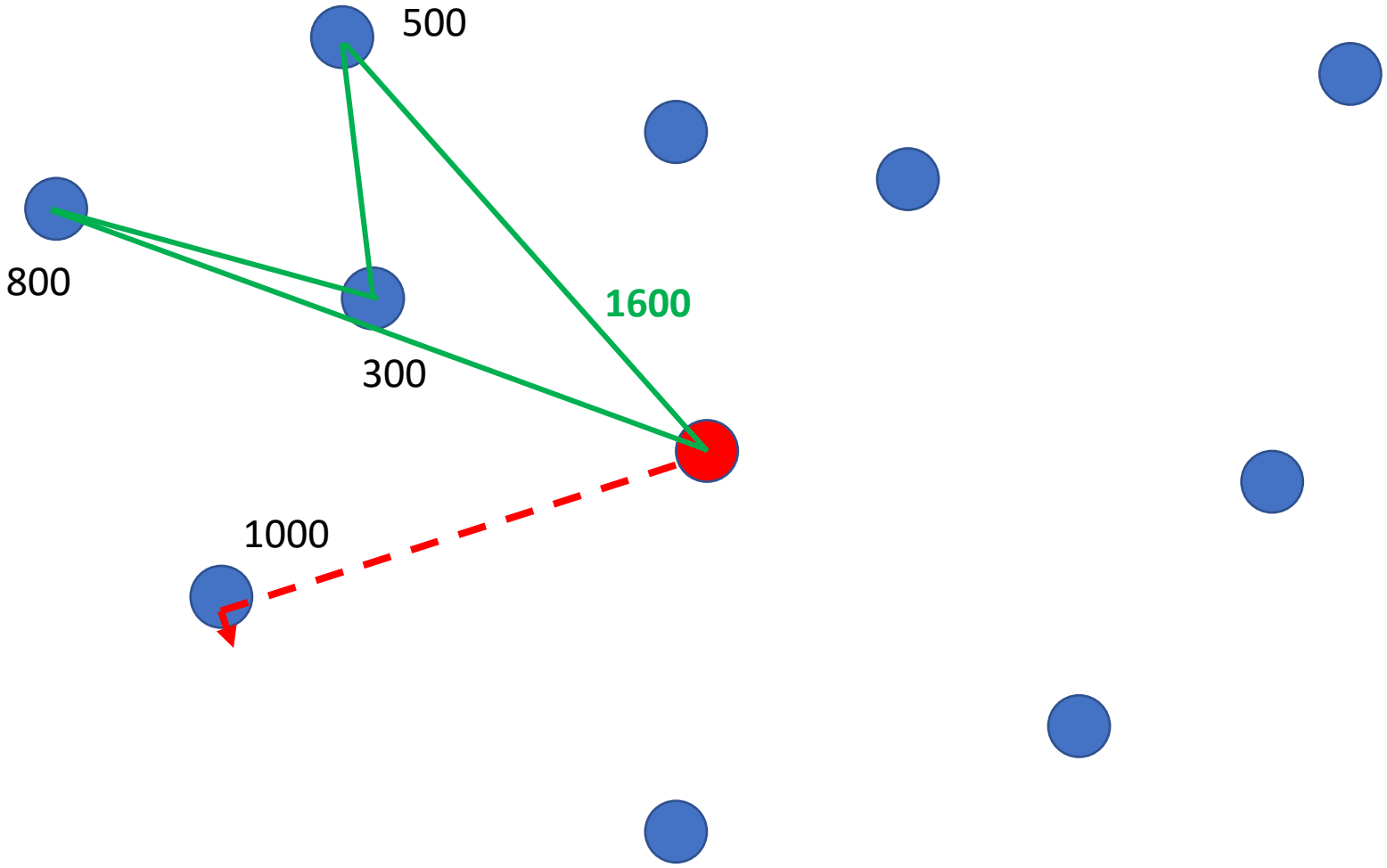
Heurística de ruteo CVRP: Sweep Algorithm (Gillett & Miller)

Capacidad de cada camión: 2000

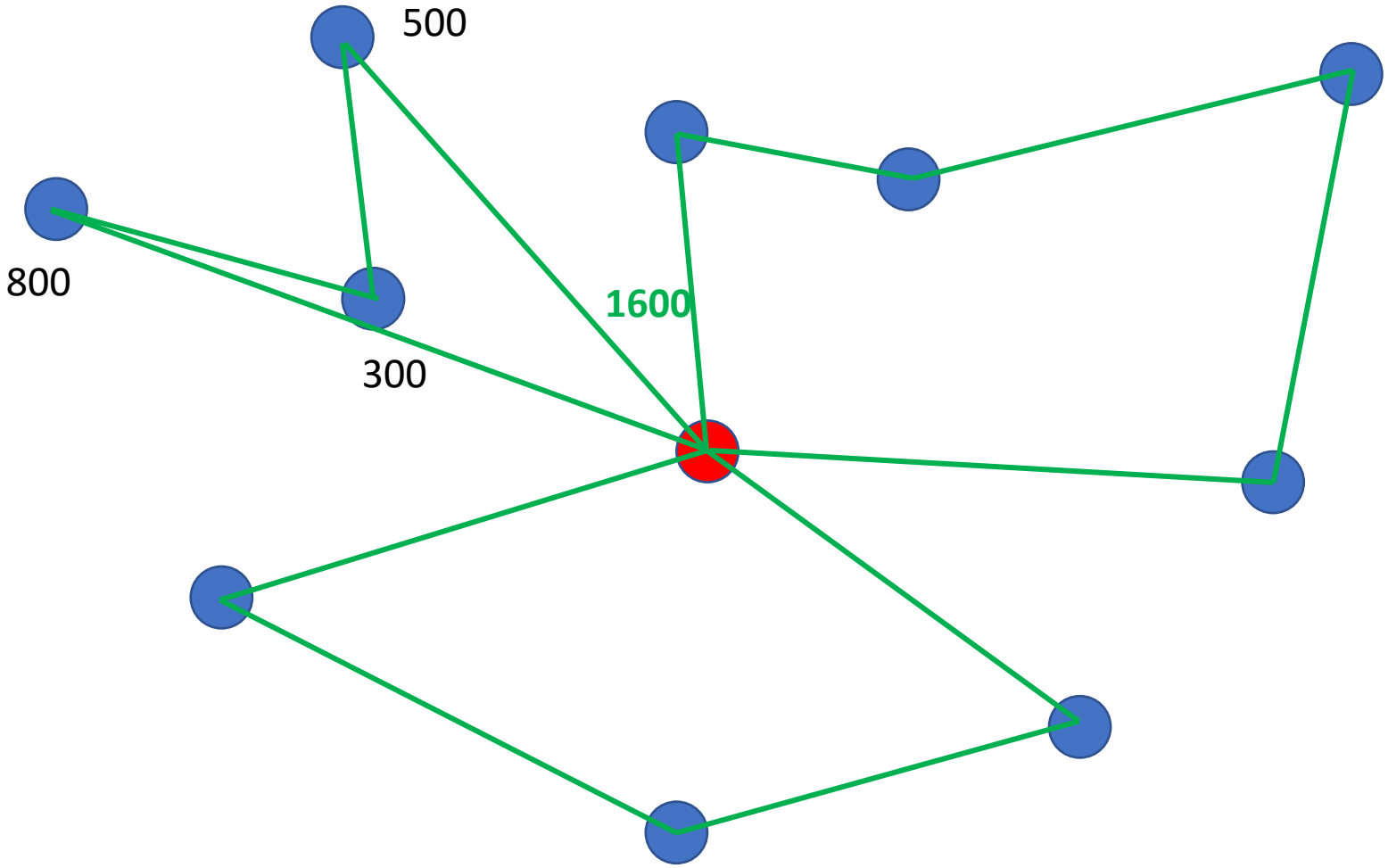


Heurística de ruteo: Sweep Algorithm (Gillett & Miller)

Capacidad de cada camión: 2000



Heurística de ruteo: Sweep Algorithm (Gillett & Miller)



Ejercicio: Implementemos Sweep Algorithm en Python

1- Vamos a la siguiente dirección:
https://colab.research.google.com/github/rmaranzana/logistica_utnba/blob/main/vrp_sweep_algorithm.ipynb
2- Se nos va a asignar un entorno virtual para ejecutar código de Python en la nube.