Algoritmos voraces (Greedy)

JOAQUÍN ARCILA PÉREZ
LAURA LÁZARO SORALUCE
CRISTÓBAL MERINO SÁEZ
ÁLVARO MOLINA ÁLVAREZ

ÍNDICE

I. Introducción

II. Desarrollo

1. Contenedores
    a. Problema
    b. Maximización del número de contenedores cargados
    c. Maximización del número de toneladas cargadas
2. Viajante de comercio (TSP)
    a. Vecino más cercano
    b. Inserción
    c. Algoritmo propio
    d. Estudio comparativo

III. Conclusión

INTRODUCCIÓN

DESARROLLO

Contenedores: Maximización contenedores cargados

```
vector<int> greedy1 (vector<int> pesos, int K){
    sort(pesos.begin(), pesos.end());
    vector<int> out;
    int suma=0;
     for(auto x : pesos){
          if((suma+=x)<=K)
               out.push_back(x);
       else break
    }
    return out;
    }
```

DESARROLLO

Contenedores: Maximización toneladas cargadas

```
vector<int> greedy2(vector<int> pesos, int K){
 int aux=0;          //carga montada
 vector<int> out;
 sort(pesos.begin(), pesos.end());
  for (auto it = pesos.end(); it != pesos.begin(); --it) {
    if((aux + *it)<=K){     //se puede meter en la carga
      aux += *it;
      out.push_back(*it);
   }
 }
 return out;
}
```

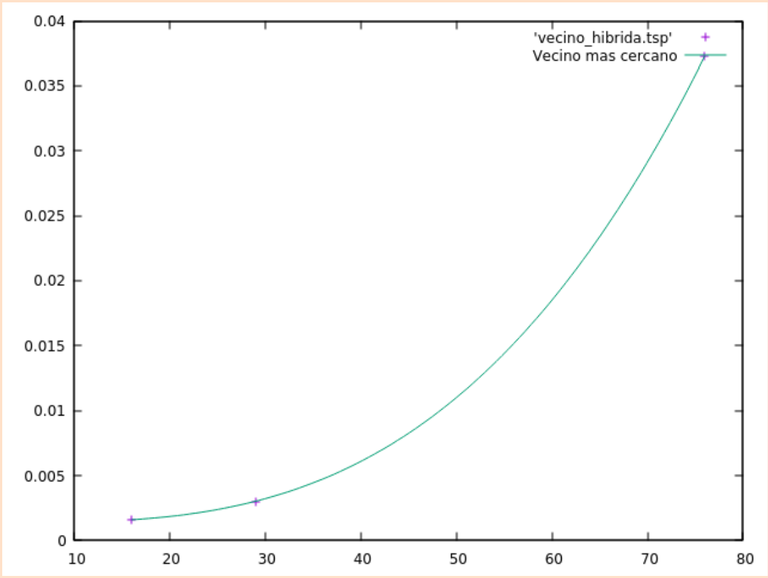
DESARROLLO

TSP: Matriz Adyacencia

0	5	5	3	10	8	7	0	11	7	25	5	5	5	6	1
5	0	1	4	16	14	13	6	17	13	31	11	10	11	12	6
5	1	0	4	16	13	12	5	16	12	30	10	10	10	12	5
3	4	4	0	12	11	10	2	14	11	28	8	7	8	8	4
10	16	16	12	0	4	5	10	7	8	15	6	6	6	4	10
8	14	13	11	4	0	1	8	4	3	17	3	3	2	2	7
7	13	12	10	5	1	0	7	4	2	18	2	2	2	3	6
0	6	5	2	10	8	7	0	11	8	25	5	5	5	6	2
11	17	16	14	7	4	4	11	0	3	15	6	6	7	7	10
7	13	12	11	8	3	2	8	3	0	18	3	3	4	5	6
25	31	30	28	15	17	18	25	15	18	0	20	20	20	19	24
5	11	10	8	6	3	2	5	6	3	20	0	0	0	3	4
5	10	10	7	6	3	2	5	6	3	20	0	0	0	2	4
5	11	10	8	6	2	2	5	7	4	20	0	0	0	2	4
6	12	12	8	4	2	3	6	7	5	19	3	2	2	0	6
1	6	5	4	10	7	6	2	10	6	24	4	4	4	6	0

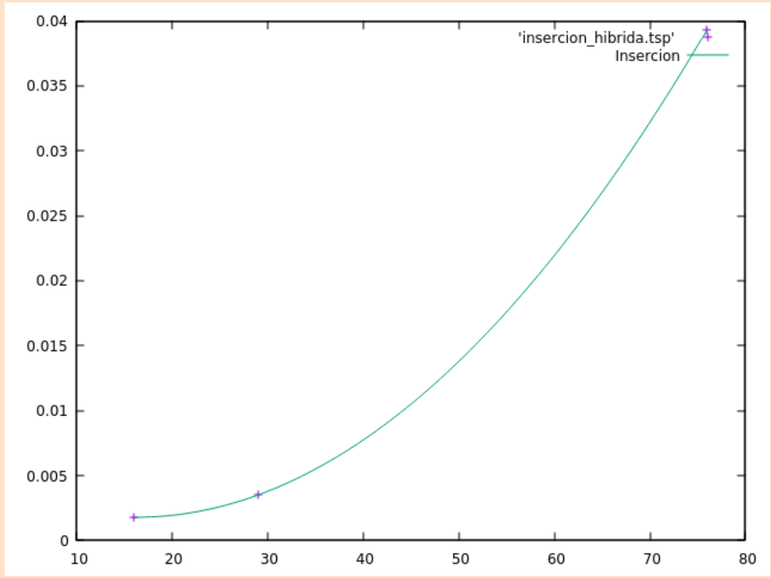
DESARROLLO
TSP: Eficiencia

- Vecino más cercano:



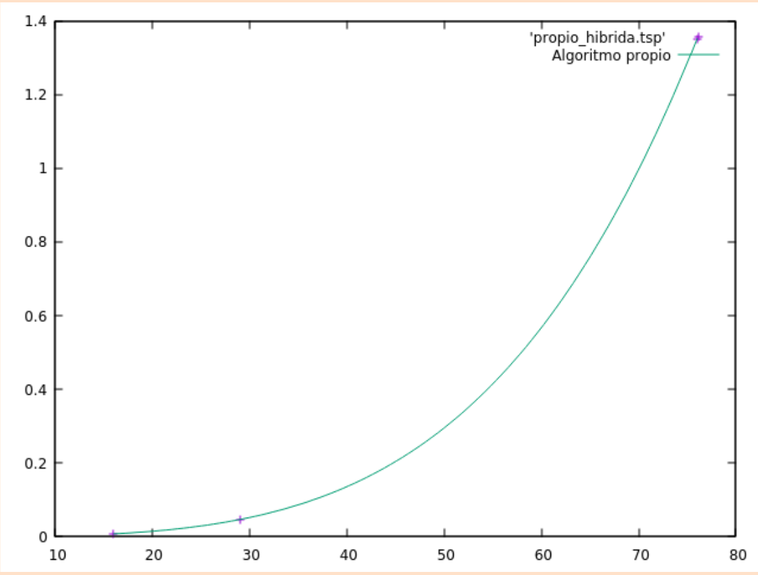
$O(n^3)$

- Inserción:



$O(n^2)$

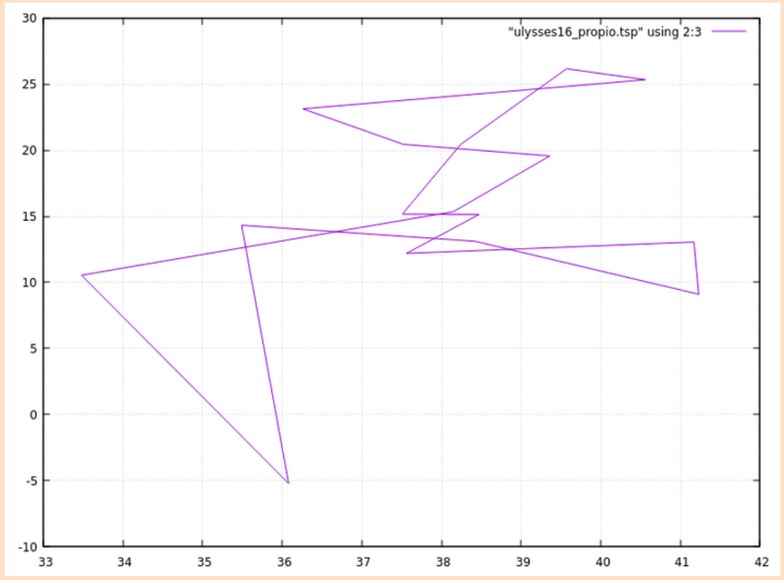
- Algoritmo propio:



$O(n^4)$

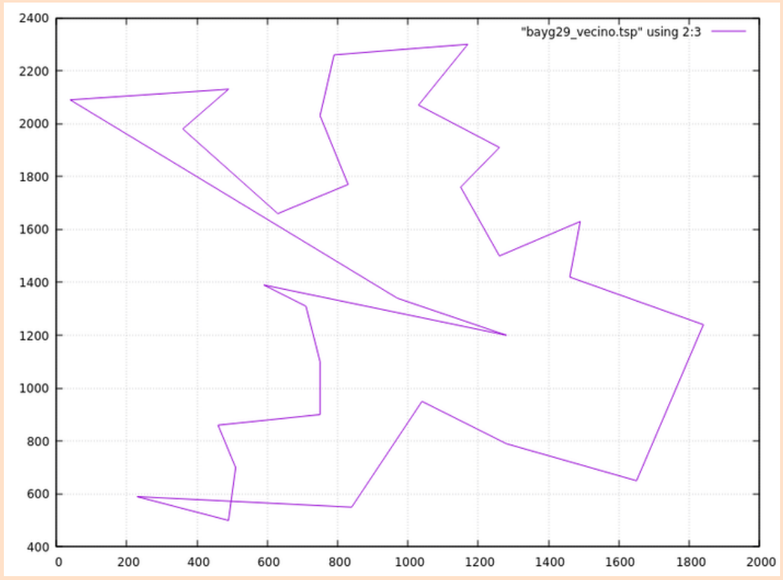
DESARROLLO
TSP: Vecino más cercano

- ulysses16.tsp:



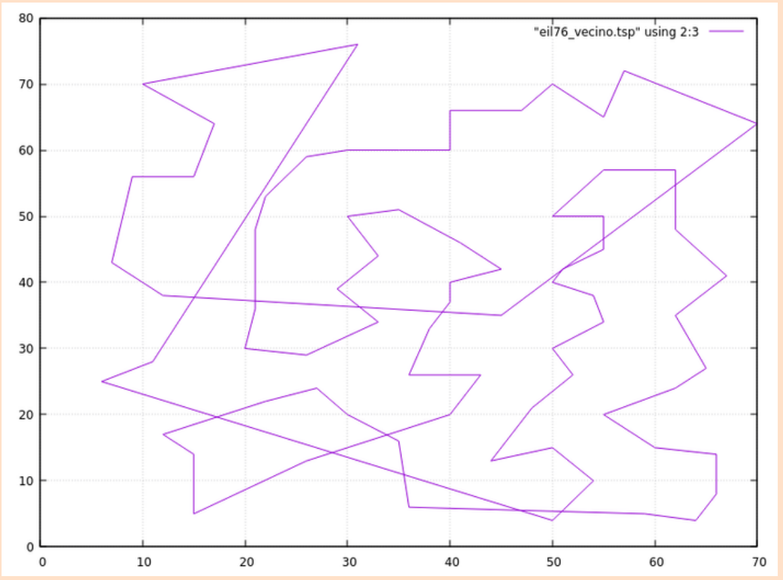
Longitud de ruta: 83

- bayg29.tsp:



Longitud de ruta: 12129

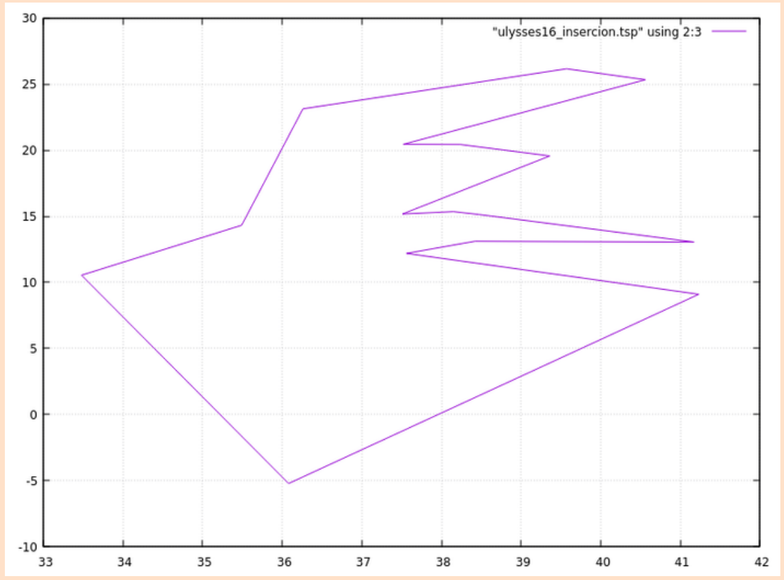
- eil76.tsp:



Longitud de ruta: 667

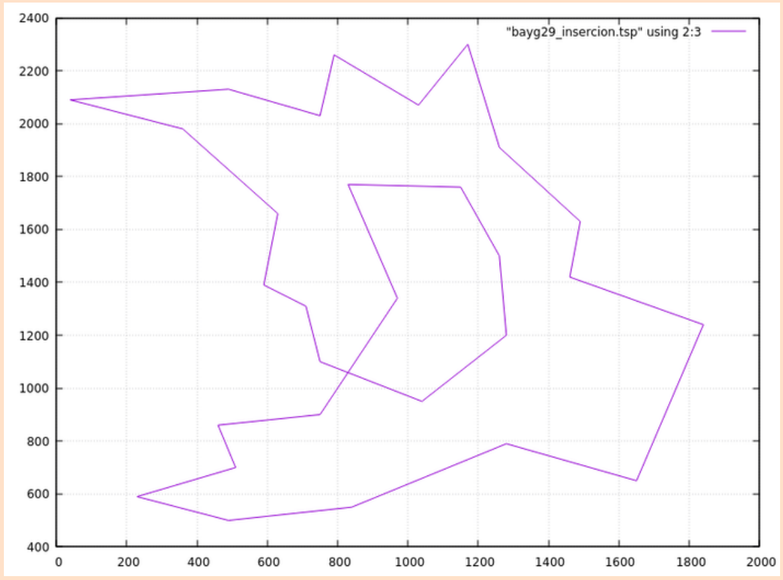
DESARROLLO
TSP: Inserción

- ulysses16.tsp:



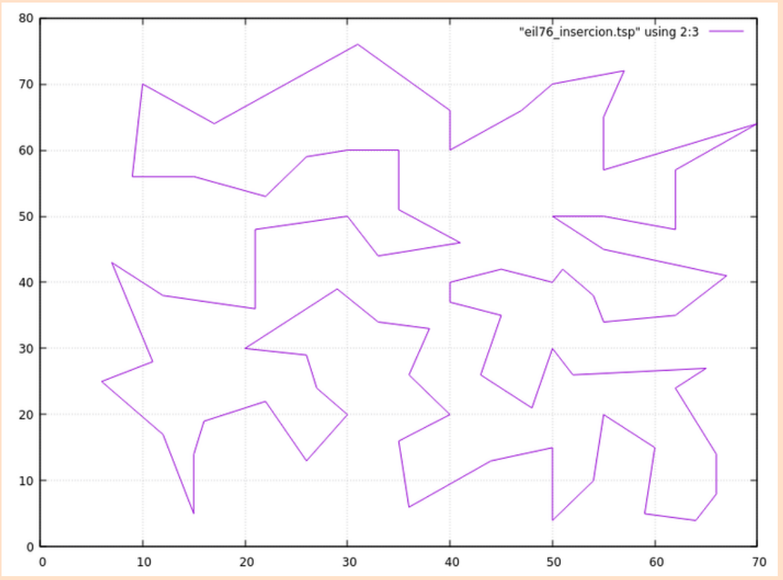
Longitud de ruta: 67

- bayg29.tsp:



Longitud de ruta: 9735

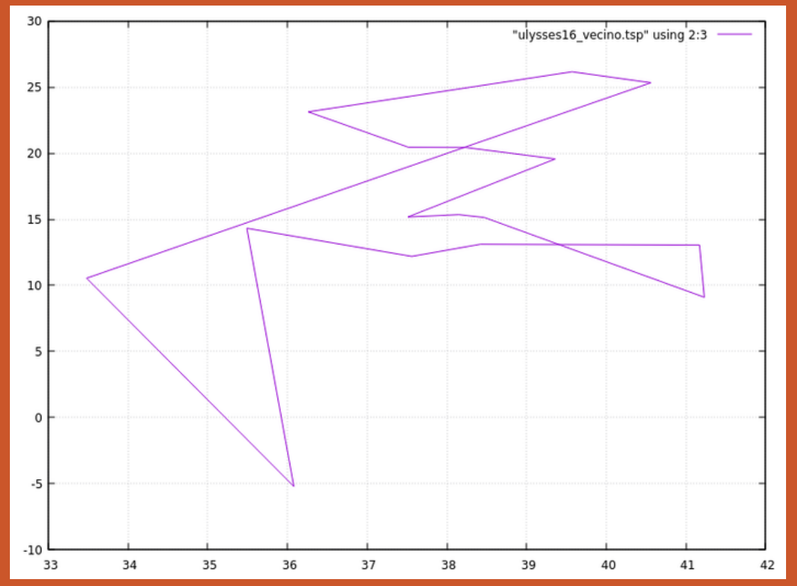
- eil76.tsp:



Longitud de ruta: 581

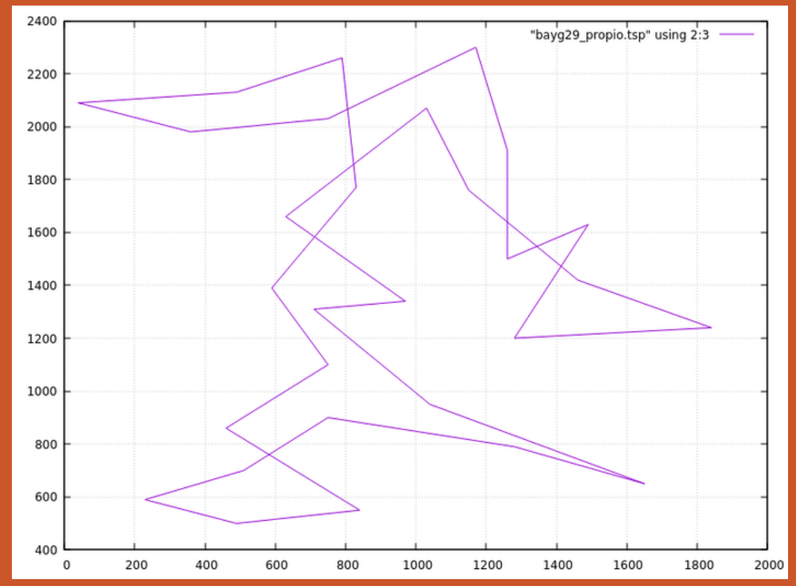
DESARROLLO
TSP: Algoritmo propio

- ulysses16.tsp:



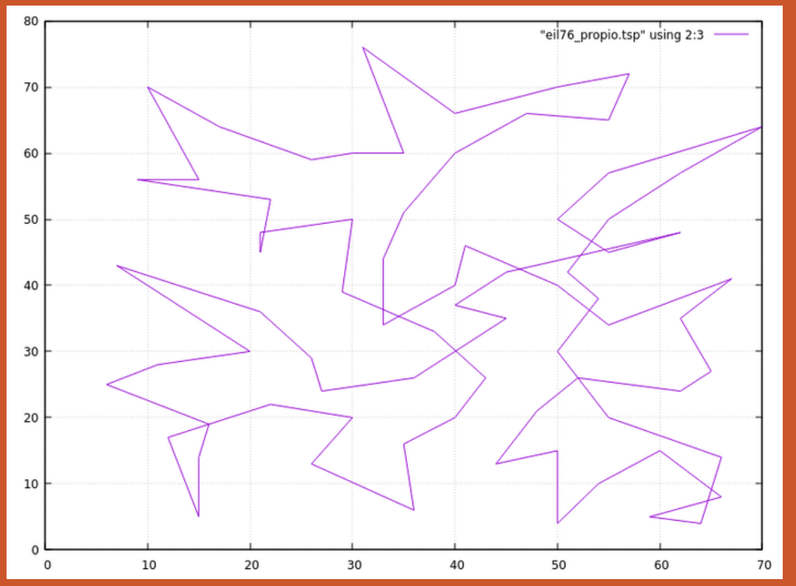
Longitud de ruta: 72

- bayg29.tsp:



Longitud de ruta: 12318

- eil76.tsp:



Longitud de ruta: 667

DESARROLLO
TSP: Estudio comparativo

- Vecino más cercano:

Ulysses: 0.001602 s	Bayg: 0.003028 s	Eil: 0.037258 s
Ruta: 83	Ruta: 12129	Ruta: 667

- Inserción:

Ulysses: 0.001791 s	Bayg: 0.003507 s	Eil: 0.039346 s
Ruta: 67	Ruta: 9735	Ruta: 581

- Propio:

Ulysses: 0.007957 s	Bayg: 0.046852 s	Eil: 1.35098 s
Ruta: 72	Ruta: 12318	Ruta: 667

CONCLUSIÓN

-El algoritmo greedy es útil en problemas que requieran maximizar o minimizar resultados.

-El algoritmo greedy es una buena opción para problemas con cierta dificultad.

- No siempre se obtienen los resultados más optimos

MUCHAS GRACIAS