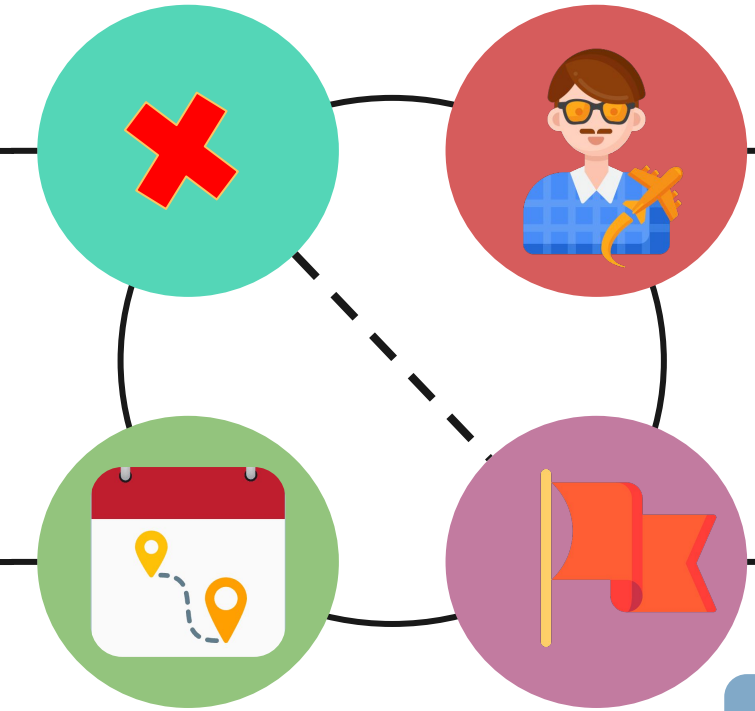


Solving Large Batches of TSP with Parallel and Distributed Computing

Computación Paralela y Distribuida
Departamento de Ciencia de la Computación

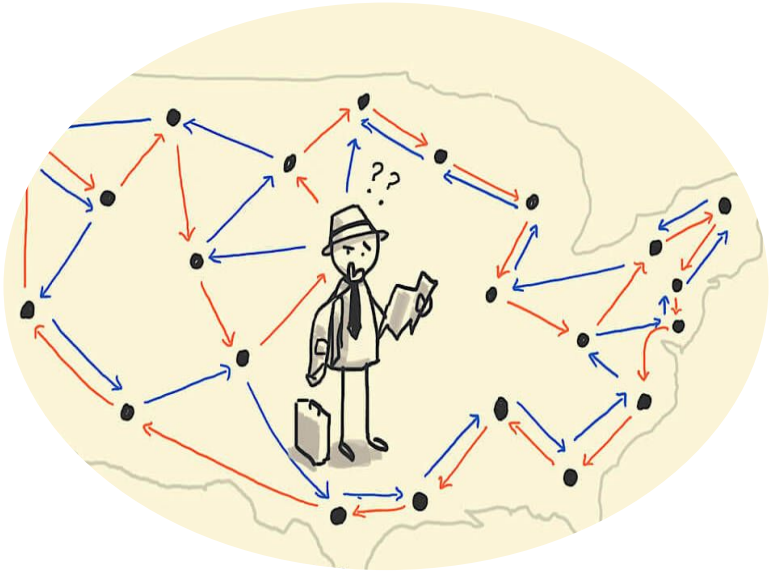
Christopher Yquira
christopher.yquira@ucsp.edu.pe

Yahaira Gomez
yahaira.gomez@ucsp.edu.pe

TABLA DE CONTENIDOS

INTRODUCCION

- **DEFINICIÓN:**
  - Problema NP-hard
- **OBJETIVO:**
  - Buscar reducción de costos
- **RESTRICCIONES:**
  - Puedo llegar a cada ciudad desde exactamente una ciudad
  - Salida en cada ciudad
  - Un solo recorrido que recorra todas las ciudades



TRAVEL SALESMAN PROBLEM

PROPUESTA

Comparar implementaciones de solvers TSP

Serial

Paralelo

Grandes lotes de TSP

Lin-Kernighan (LKH)

Concorde

Lin-Kernighan (LKH)

Generalización de 2-opt

Adaptativo

Intercambiar pares de sub-tours para hacer un nuevo tour

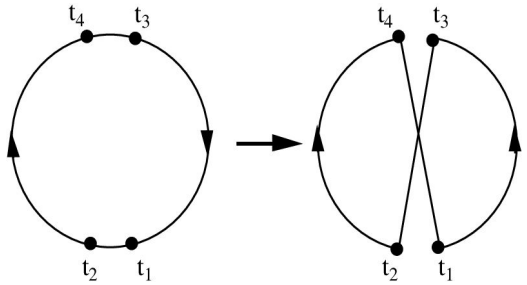


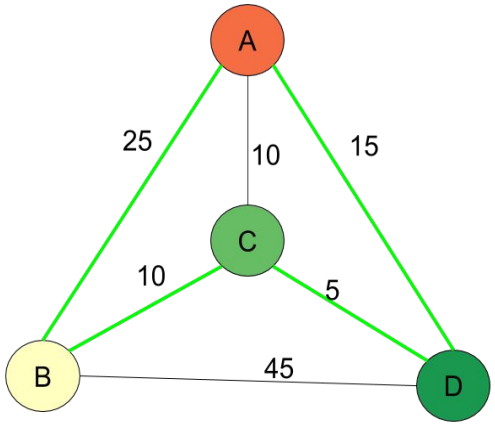
Figure 2.1 A 2-opt move

ALGORITMO LKH

1. Generate a random initial tour T.
2. Let i = 1. Choose t_1.
3. Choose $x_1 = (t_1,t_2) \in T$.
4. Choose $y_1 = (t_2,t_3) \notin T$ such that $G_1 > 0$.
   If this is not possible, go to Step 12.
5. Let i = i+1.
6. Choose $x_i = (t_{2i-1},t_{2i}) \in T$ such that
   (a) if t_{2i} is joined to t_1, the resulting configuration is a tour, T', and
   (b) $x_i \neq y_s$ for all $s < i$.
   If T' is a better tour than T, let T = T' and go to Step 2.
7. Choose $y_i = (t_{2i},t_{2i+1}) \notin T$ such that
   (a) $G_i > 0$,
   (b) $y_i \neq x_s$ for all $s \leq i$, and
   (c) x_{i+1} exists.
   If such y_i exists, go to Step 5.
8. If there is an untried alternative for y_2, let i = 2 and go to Step 7.
9. If there is an untried alternative for x_2, let i = 2 and go to Step 6.
10. If there is an untried alternative for y_1, let i = 1 and go to Step 4.
11. If there is an untried alternative for x_1, let i = 1 and go to Step 3.
12. If there is an untried alternative for t_1, then go to Step 2.
13. Stop (or go to Step 1).

EN RESUMEN...

CASO 1: Ir de ciudad A hacia D, pasando por todas las demás ciudades



	CityId	X	Y
2	0	316.8367391	2202.340707
3	1	4377.405972	336.6020822
4	2	3454.158198	2820.053011
5	3	4688.099298	2935.898056
6	4	1010.696952	3236.750989
7	5	2474.230877	1435.514651
8	6	1029.277795	2721.800952

Solucionar y evaluar varios TSP's ⟶ Serial y paralelo

Lin-Kernighan (LKH)

TRABAJO REALIZADO

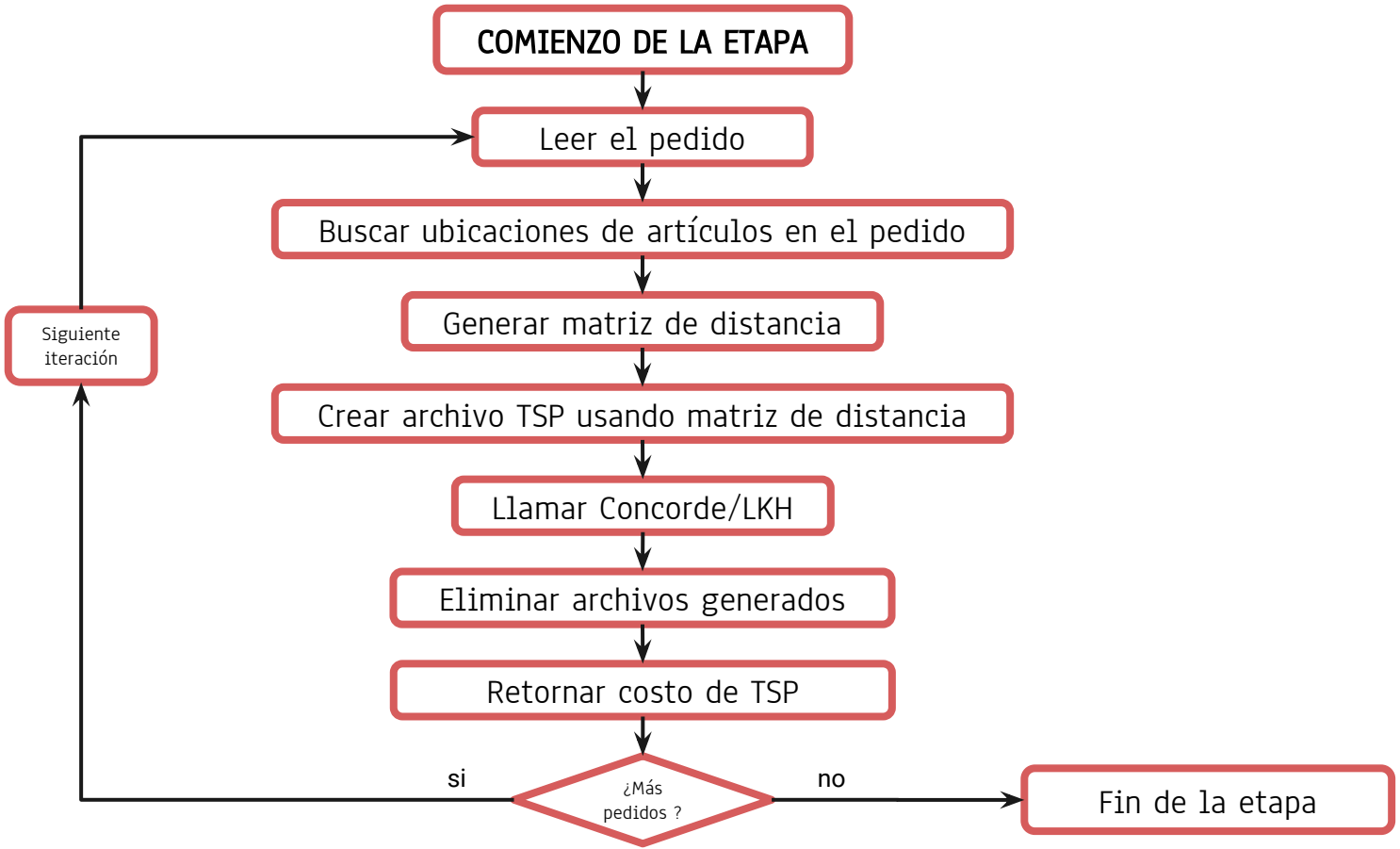
P1 Crear una red de distribución

P2

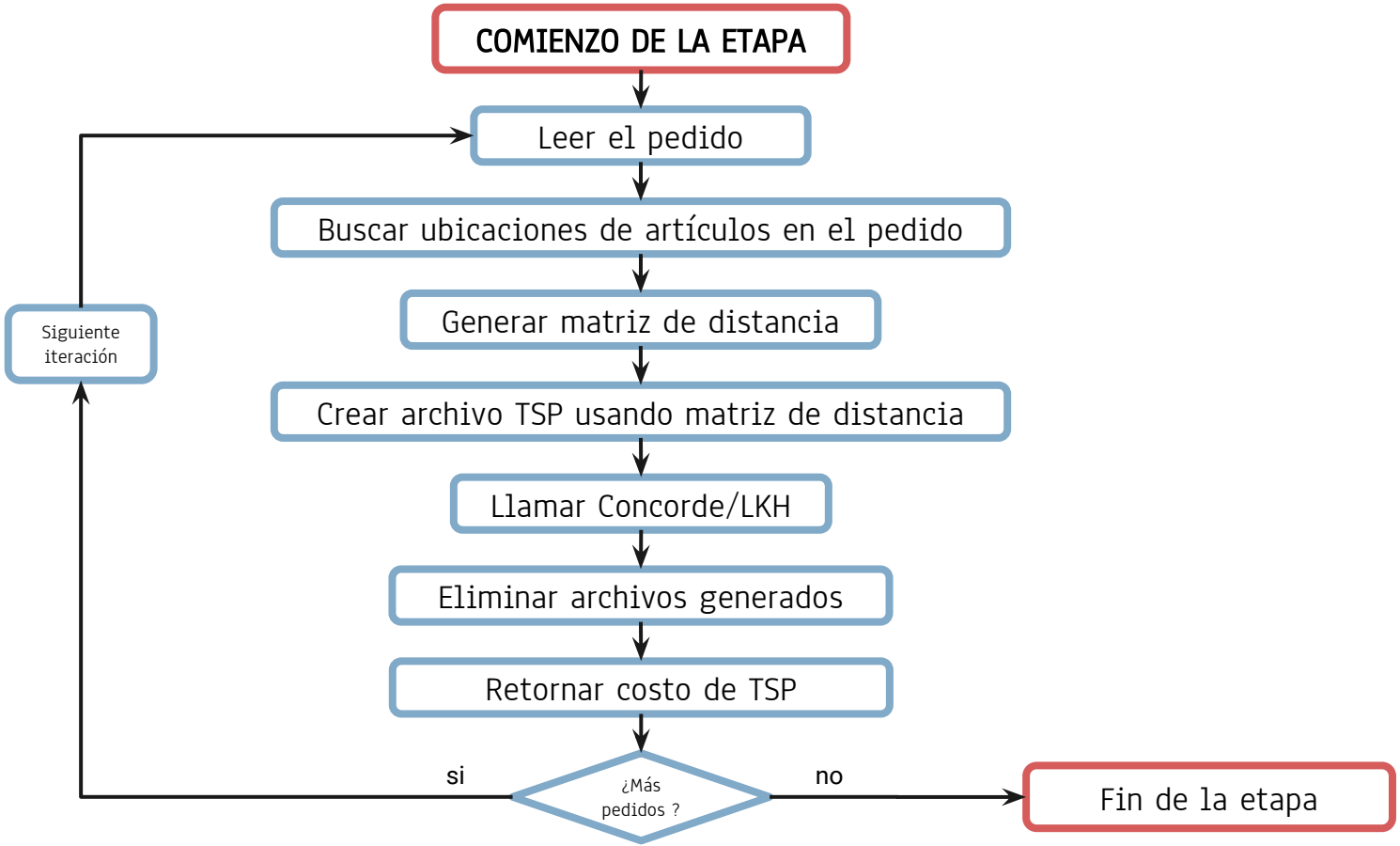
1 TSP por pedido

P3

IMPLEMENTACIÓN - SERIAL

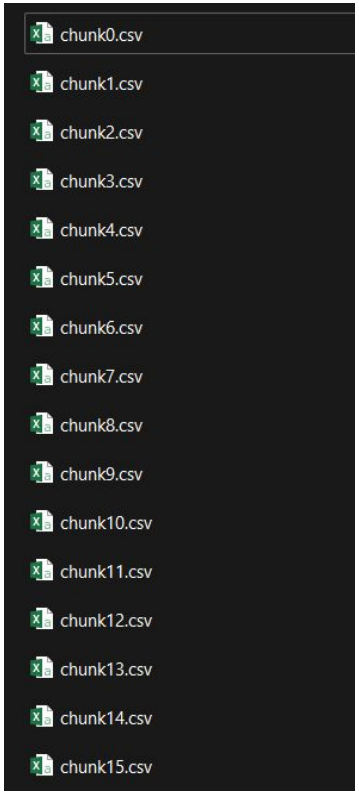


IMPLEMENTACIÓN - PARALELA



CÓDIGO SERIAL

CityId,X,Y
0,316.8367391,2202.340707
1,4377.405972,336.6020822
2,3454.158198,2820.053011
3,4688.099298,2935.898056
4,1010.696952,3236.750989
5,2474.230877,1435.514651
6,1029.277795,2721.800952
7,3408.887685,199.5857931
8,1304.006125,2657.427246
9,4211.525725,2294.595208
10,297.5754577,1952.548486
11,2052.1165,578.2935261
12,2645.077176,2717.838772
13,965.611152,1067.734281
14,454.9887227,1217.670993
15,3369.344927,1504.60374
16,4944.059453,2326.338189
17,4313.207563,2800.438423
18,2352.743647,2489.939529
19,3033.179607,515.2176131
20,3421.865966,1597.526985
21,4326.667571,1607.378707
22,1383.884494,3167.74721
23,3633.815728,2889.995167
24,3694.082279,734.9497566
25,4646.266998,2884.589219



INFORMACIÓN DE PC

- PROCESADOR → AMD Ryzen 7
- TARJETA GRÁFICA → Radeon RX Vega 10

create_dataset.py - C:\Users\Valeria\Documents\CCOMP1-2022\PARALELA\p_final\Large-TS...

File Edit Format Run Options Window Help

```
import os
import pandas as pd

def createDataset(numCities, batch):
  counter = 0
  for i,chunk in enumerate(pd.read_csv('cities.csv', chunksize=numCities)):
    chunk.to_csv('dataset/chunk{}.csv'.format(i), index=False)
    counter += 1
    if(counter == batch):
      break

createDataset(100, 16)
```

CÓDIGO SERIAL

```
void read_CSV(string argv, vector<string>* vectorX, vector<string>* vectorY){
    ifstream file(argv);
    string line;
    char delimitator = ',';
    getline(file, line);

    while(getline(file, line)){
        stringstream stream(line);
        string index, x, y;
        getline(stream, index, delimitator);
        getline(stream, x, delimitator);
        getline(stream, y, delimitator);

        //cout << "Index:" << index << endl;
        //cout << "x: " << x << endl;
        //cout << "y: " << y << endl;

        (*vectorX).push_back(x);
        (*vectorY).push_back(y);
    }
}
```

CÓDIGO SERIAL

```
void write_TSP(vector<string> vectorX, vector<string> vectorY){
    string filename("LKH-2.0.9/cities_tsp/cities.tsp");
    fstream outfile;

    outfile.open(filename, std::ios_base::out);
    if (!outfile.is_open()) {
        cout << "failed to open " << filename << '\n';
    } else {
        outfile << "NAME : traveling-santa-2018-prime-paths" << endl;
        outfile << "COMMENT : traveling-santa-2018-prime-paths" << endl;
        outfile << "TYPE : TSP" << endl;
        outfile << "DIMENSION : " << vectorX.size() << endl;
        outfile << "EDGE_WEIGHT_TYPE : EUC_2D" << endl;
        outfile << "NODE_COORD_SECTION" << endl;
        for(unsigned int i = 0; i < vectorX.size(); i++){
            outfile << i+1 << " " << vectorX[i] << " " << vectorY[i] << endl;
        }
        outfile << "EOF" << endl;
        //cout << "Done Writing!" << endl;
    }
}
```

CÓDIGO SERIAL

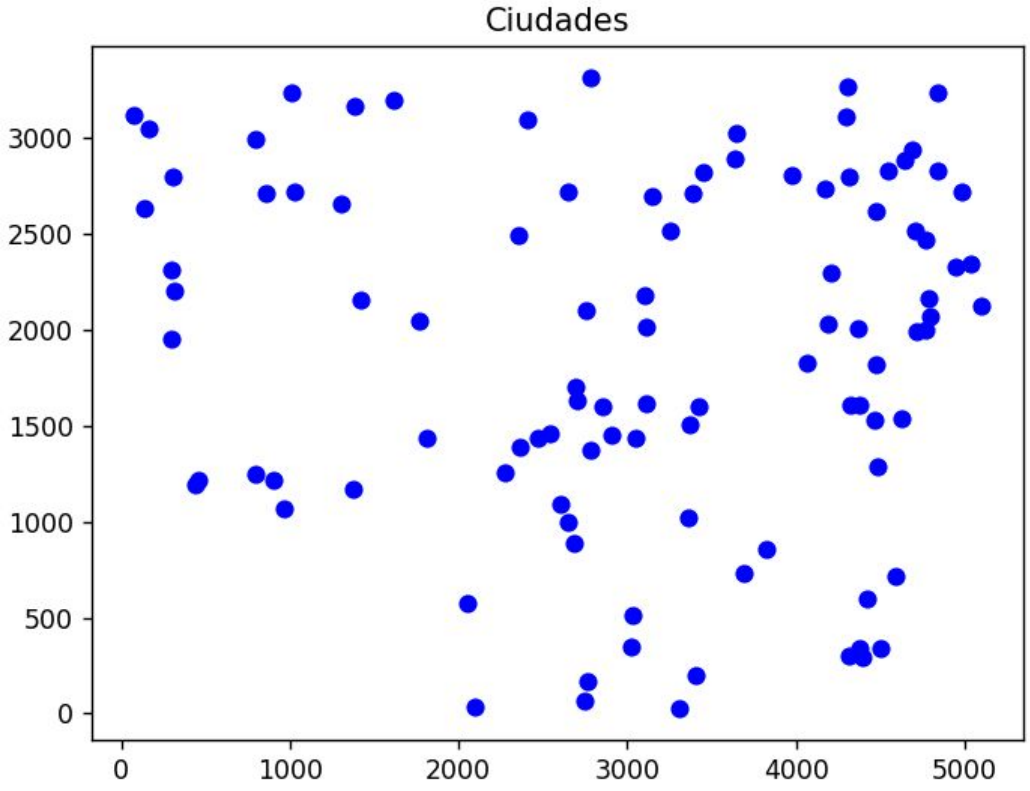
```cpp
double score_tour(string filename){
    double fiscore;
    ifstream file(filename);
    string line;
    string delimitator = "Length = ";
    getline(file, line);
    getline(file, line);
    string score_value = line.substr(19);
    //cout << score << endl;
    return stod(score_value);
}
```

```cpp
double serial_solver(vector<string> paths){
    double best_score = INT_MAX;
    for (unsigned int i = 0; i < paths.size(); i++){
        vector<string> x;
        vector<string> y;
        read_CSV(paths[i], &x, &y);
        write_TSP(x,y);
        double score = score_tour("LKH-2.0.9/solution_csv/tsp_solution.csv");
        //cout << "\nScore [" << i << "]: " << score << " ------------------here!"<< endl;
        // DETERMINA SI ES EL MEJOR SCORE
        if(score < best_score) best_score = score;
    }

    return best_score;
}
```

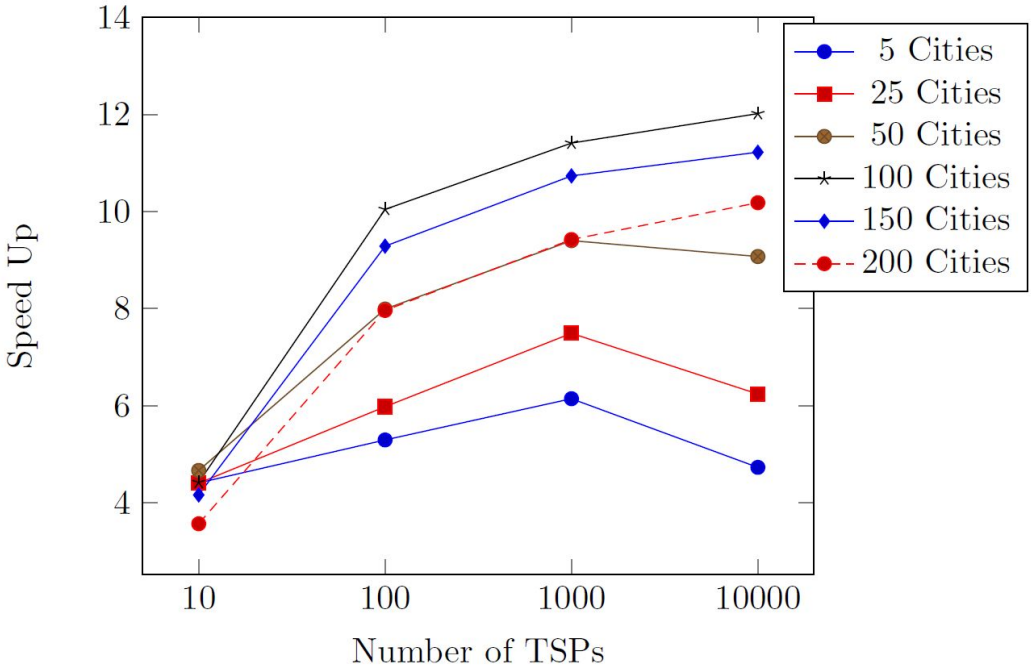
CÓDIGO PARALELO

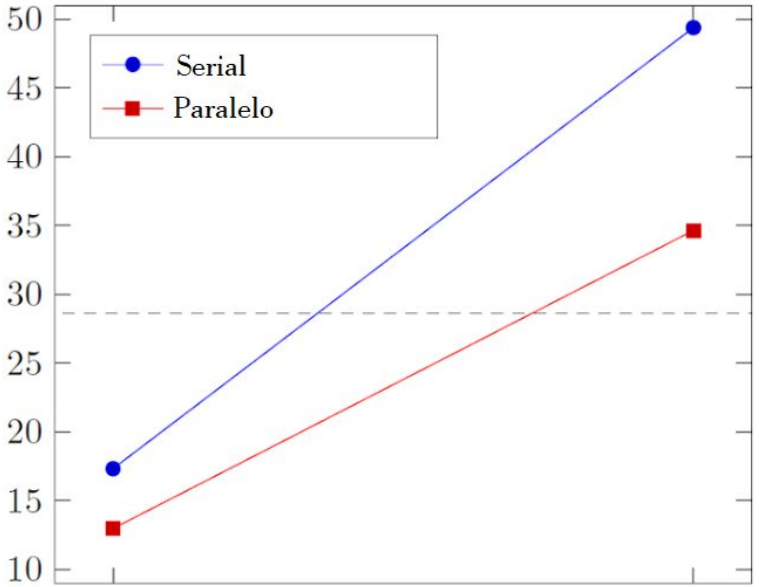
```
double parallel_solver(vector<string> paths, unsigned int num_threads){
    vector<thread> threadVect;
    unsigned int thread_spread = paths.size() / num_threads;
    double best_score = INT_MAX;
    cout<<paths.size()<<endl;


    for (unsigned int i = 0; i < num_threads; i++) {
        threadVect.emplace_back(solver, &paths, i, thread_spread, &best_score);
    }

    for (auto& t : threadVect) {
        t.join();
    }
    return best_score;
}
```

RESULTADOS

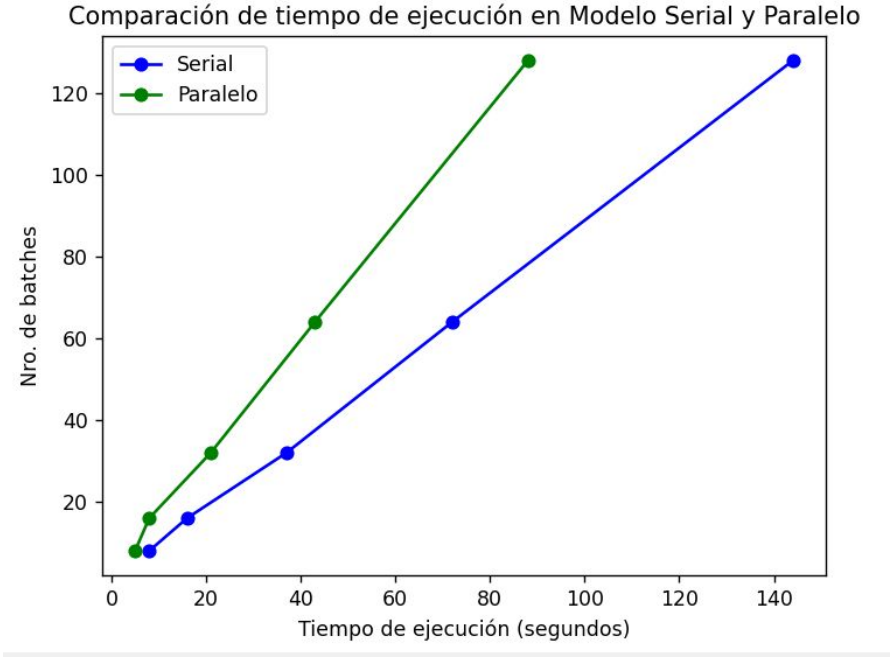

Ciudades
3000
2500
2000
1500
1000
500
0
0
1000
2000
3000
4000
5000

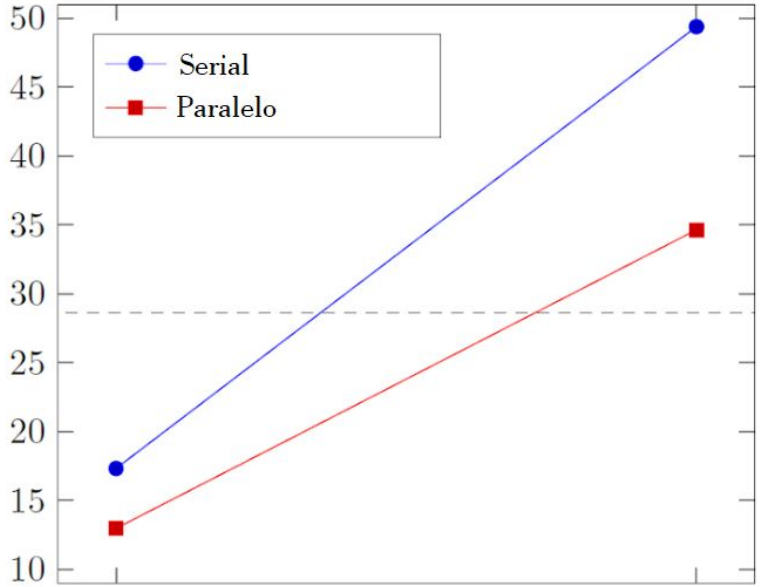
RESULTADOS PAPER





RESULTADOS ESTIMADOS





CONCLUSIONES

1

Paralelo

Reduce significativamente el tiempo en general

2

TSP

Tarea más demandante computacional

3

Paralela C++

Forma simple de resolver TSP

4

LPT

Funciona mejor que un EDR

5

Paralela y Distribuido

Trabajo a futuro

REFERENCIAS

- S.G.Ozdena, A.E.Smitha and K.R.Gue (2017) Solving large batches of traveling salesman problems with parallel and distributed computing
- D. Karapetyana, G. Gutina (2010) Lin-Kernighan heuristic