Results of ILS trials

Isaac Kosloski Oliveira[1]

[1]Faculdade de Computação
Universidade Federal de Mato Grosso do Sul (UFMS) Campo Grande – MS – Brazil

isaac.kosloski@ufms.br

1. Introduction

This write-up reports the results of an implementation of the Iterated Local Search algorithm (ILS), both sequential and parallel, for the Traveling Salesman Problem (TSP). The results are disposable in two phases. The first phase compares the evaluation measures of ILS on 1 and 12 cores (sequential and parallel, respectively, and optimal). The second phase compares the proposed parallel algorithm with the reported results of metaheuristics algorithms that were used to solve the TSP in the literature.

The implementation uses a pure *greedy algorithm* for the **Start** procedure, a *2-opt* as **Local Search**, one *double-bridge* for the **Perturbation** method, and the *better* for **Acceptance criteria**. The implementation can be seen in algorithm 1.

Algorithm 1 Iterated Local Search for Traveling Salesman Problem

```
PROCEDURE Iterated Local Search
    $s_0$ = greedySolution()
    $s^* = 2opt(s_0)$
    REPEAT
        $s' = doubleBridge(s^*, history)$
        $s^{*'} = 2opt(s')$
        $s^* = better(s^*, s^{*'}, history)$
    UNTIL Iterations $> 1000$
END
```

2. Experiment Setup

2.1. Implement Language and Frameworks

The implementation uses C++20 and OpenMP 5.2, compiled using GCC 12.2.0.

2.2. Testing Enviroment

The code was compiled and run on a machine x86_64, AMD Ryzen 5 5600GT with Radeon Graphics, 6 cores, and 12 threads. With Debian 12.2.0-14. The program was compiled using *g++ component.cpp node.cpp scanner.cpp functions.cpp mainExec.cpp -o ../bin/TspPar -fopenmp -Wall -pedantic*, and tested using a shell script.

2.3. Datasets - Inputs

For the datasets, nine TSP benchmarks were downloaded from TSPLIB [TspLib], which includes d198, a280, lin318, pcb442, rat783, u1060, pcb1173, d1291, and fl1577 have been used in this write-up for evaluating the proposed algorithm performance. The selected benchmarks varied in several cities where each city was represented by a 2D-Euclidian coordinate. The benchmarks' input was a *.tsp* file, with the format:

NAME : <benchmark name>

COMMENT : <benchmark description>

TYPE : TSP

DIMENSION : <dimension>

EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION

<*i*> <coordinate x_i > <coordinate y_i >

EOF

The table 1 displays benchmarks properties.

Benchmark name	Dimension	Description
d198.tsp	198 city	Drilling problem (Reinelt)
a280.tsp	280 city	drilling problem (Ludwig)
lin318.tsp	318 city	318-city problem (Lin/Kernighan)
pcb442.tsp	442 city	Drilling problem (Groetschel/Juenger/Reinelt)
rat783.tsp	783 city	Rattled grid (Pulleyblank)
u1060.tsp	1060 city	Drilling problem problem (Reinelt)
pcb1173.tsp	1173 city	Drilling problem (Juenger/Reinelt)
d1291.tsp	1291 city	Drilling problem (Reinelt)
fl1577.tsp	1577 city	Drilling problem (Reinelt)

Table 1. TSP benchmarks' properties

2.4. Data Analysis - outputs

The output data, *.sol* file, has the format:

Iterations: 1000

Time: <time in seconds> sec - <time in minutes> min - <time in hours> horas

<fraction of time in hours> h <fraction of time in minutes> min

Problem dimension: <dimension>

Total distance: <total path distance>

[v_1] [v_2] [v_3] ... [v_i]

Where v_i it's the city in the position i on the tour sequence. For each benchmark, there are 30 different output *.sol* files, to define the trial result statistically accurately.

2.5. Evaluation Measures

This section presents the evaluation metrics that are used for evaluating the proposed method.

1. Distance: the value of the best route found.
2. Accuracy: the percentage of retrieving the best route correctly. The optimal solution is provided in TSPLIB [TspLib].
3. Running time: is the duration between the end and the beginning of the main part (ILS algorithm itself) of the program running, using **chrono::high_resolution_clock::now()** for the sequential program and **omp_get_wtime()** for the parallel program.

$$RT = \text{finish\_time - start\_time} \tag{1}$$

4. Speedup: is the improvement in speed of execution of a task executed on two similar architectures with different resources:

$$SP = \frac{T_s}{T_p(n)} \tag{2}$$

   - T_s: Sequential running time;
   - $T_p(n)$: Parallel running time in function of n;
   - n: Number of cores.

5. Efficiency: represents the speedup divided by the number of cores

$$EF = \frac{SP}{n} \tag{3}$$

3. Experimental results

The results are taken by the average of 30 trials on each instance.

3.1. Comparison of Sequential and Parallel ILS

Table 2. Distance of each symmetric TSP benchmark, the Sequential and Parallel ILS, then the optimal solution.

Instance	Sequential Distance	Parallel Distance	Optimal
d198	1.628×10^4	1.721×10^4	1.578×10^4
a280	2.643×10^3	2.729×10^3	2.579×10^3
lin318	4.382×10^4	4.763×10^4	4.203×10^4
pcb442	5.172×10^4	5.348×10^4	5.078×10^4
rat783	9.271×10^3	9.659×10^3	8.806×10^3
u1060	2.317×10^5	2.416×10^5	2.241×10^5
pcb1173	0.000	6.498×10^4	5.689×10^4
d1291	0.000	5.690×10^4	5.080×10^4
fl1577	0.000	2.966×10^4	2.2249×10^4

Table 3. Accuracy of each symmetric TSP benchmark for Sequential and Parallel ILS.

Instance	Sequential Accuracy (%)	Parallel Accuracy (%)
d198	96.91	91.97
a280	97.58	94.50
lin318	95.91	88.25
pcb442	98.18	94.95
rat783	94.98	91.17
u1060	96.74	92.75
pcb1173	00.00	87.55
d1291	00.00	89.27
fl1577	00.00	75.02

Table 4. Running time, speedup, and efficiency of each symmetric TSP benchmark for Sequential and Parallel ILS.

Instance	Sequential Time (s)	Parallel Time (s)	Speedup	Efficiency (%)
d198	6.114×10^1	8.649	7.069	58.91
a280	1.755×10^2	2.411×10^1	7.279	60.66
lin318	2.775×10^2	3.941×10^1	7.041	58.68
pcb442	7.463×10^2	1.018×10^2	7.331	61.09
rat783	4.620×10^3	5.999×10^2	7.701	64.18
u1060	1.217×10^4	5.062×10^3	2.404	20.03
pcb1173	0.000	2.183×10^3	0.000	00.00
d1291	0.000	2.712×10^3	0.000	00.00
fl1577	0.000	5.797×10^3	0.000	00.00

4. Related work

4.1. Solving Traveling Salesman Problem Using Parallel River Formation Dynamics Optimization Algorithm on Multi-core Architecture Using Apache Spark

Year: 2024

It uses the parallel RFD algorithm for solving the TSP, comparing speedup, running time, and efficiency on 1 (sequential), 4, 8, and 16 cores. Then compare to three parallel water-based algorithms (*Water Flow, Intelligent Water Drops* and *Water Cycle*). Then, compare the proposed algorithm with the reported metaheuristics that were used to solve TSP in the literature.

Similar benchkmarks:

- d198;
- lin318;
- rat783.

Can be used for comparison:

→ Distance;
→ Accuracy;
→ Running time;
→ Speedup.

4.2. The AddACO: A bio-inspired modified version of the ant colony optimization algorithm to solve travel salesman problems

Year: 2024

It's a proposal to solve the TSP with the AddACO algorithm (it's a version of the Ant Colony Optimization method that is characterized by a modified probabilistic law at the basis of the exploratory movement of the artificial insects). In particular, the ant decisional rule is here set to the amount in a linear convex combination of competing behavioral stimuli and has, therefore, an additive form (hence the name of our algorithm), rather than the canonical multiplicative one. The AddACO intends to address two conceptual shortcomings that characterize classical ACO methods:

(i) the population of artificial insects is in principle allowed to simultaneously minimize/maximize all migratory guidance cues (which is implausible from a biological/ecological point of view).
(i) a given edge of the graph has a null probability to be explored if at least one of the movement traits is therein equal to zero, i.e., regardless of the intensity of the others (this in principle reduces the exploratory potential of the ant colony).

Similar benchmarks:

- lin318;

Can be used for comparison:

→ Distance;

4.3. A novel hybrid swarm intelligence algorithm for solving TSP and desired-path-based online obstacle avoidance strategy for AUV

Year: 2024

Similar benchmarks:

- none;

Can be used for comparison:

→ Distance;

4.4. Discrete artificial bee colony algorithm with fixed neighborhood search for traveling salesman problem

Year: 2024

It proposes a discrete artificial bee colony algorithm with a fixed neighborhood search for the traveling salesman problem (TSP), called DABC-FNS. The solution obtained by the discrete artificial bee colony algorithm is expressed by a positive integer coding method. Meanwhile, the local enhancement strategy and the 2-opt strategy with fixed neighborhood search are introduced to improve the solution accuracy of the ABC algorithm.

Similar benchmarks:

- d198;
- a280;
- rat783.

Can be used for comparison:

$\rightarrow$ Distance;

4.5. The Discrete Carnivorous Plant Algorithm with Similarity Elimination Applied to the Traveling Salesman Problem

Year: 2022

It uses a combination of six steps: first, the algorithm redefines subtraction, multiplication, and addition operations, which aims to ensure that it can switch from continuous space to discrete space without losing information; second, a simple sorting grouping method is proposed to reduce the chance of being trapped in a local optimum; third, the similarity-eliminating operation is added, which helps to maintain population diversity; fourth, an adaptive attraction probability is proposed to balance exploration and the exploitation ability; fifth, an iterative local search (ILS) strategy is employed, which is beneficial to increase the searching precision; finally, to evaluate its performance, DCPA is compared with nine algorithms.

Similar benchmarks:

- d198;
- lin318;
- pcb442;
- rat783;
- fl1577.

Can be used for comparison:

$\rightarrow$ Distance;

5. Comparison of ILS with other works

Table 5. Accuracy of each symmetric TSP benchmark for Sequential and Parallel ILS.

Instance	Sequential ILS (%)	Parallel ILS (%)	Sequential RFD	Parallel RFD
d198	96.91	91.97	92	98
lin318	95.91	88.25	95	98
rat783	94.98	91.17	92	97

Table 6. Speedup, and efficiency of each symmetric TSP benchmark for ILS and RFD.

Instance	ILS Speedup	ILS Efficiency (%)	RFD Speedup	RFD Efficiency (%)
d198	7.069	58.91	2.87	16
lin318	7.041	58.68	2.41	24.1
rat783	7.701	64.18	2.45	15

References

TspLib. Disponível em: `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`. Acesso em: 11 de setemrbo 2023.