

# INF-3201 Parallel Programming

## Shared Memory

Enrico Tedeschi  
ete011@post.uit.no

### I. INTRODUCTION

The goal of this assignment is to parallelize a piece of code using shared memory techniques preferably using **OpenMP**.

#### A. Requirements

- Choose a piece of software to parallelize
- Parallelize it with shared-memory techniques
- Evaluate speedup

### II. TECHNICAL BACKGROUND

- Concurrency and parallelism concepts
- Parallel programming concepts
- Basic programming approach
- Knowledge of C language
- Notion of design pattern principles
- Theory about software engineering
- Knowledge of git to manage the software versions

### III. ANALYSIS

The given **Markovian** code was found to be too long and quite hard to understand, in addition, it has no deterministic solution so it would have been really difficult to test.

The code chosen to be parallelized is a sequential version of the **TSP** (Traveller Salesman Problem). It asks the following question: *Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?* [1]

The sequential code was refined by adding a random function which generate distances between cities and also the possibility of run the programme with a size passed as a parameter was implemented.

To parallelize a sequential code is necessary to analyse which function involves the biggest amount of time. The c given program *tsp-seq.c* has been tested with *profile* and the following table is the result of this test:

	%	cumulative	self	
	time	seconds	calls	name
1	100.32	2.90	1	zRoute
2	0.00	0.00	196	zEuclidDist
3	0.00	0.00	28	random_at_most
4	0.00	0.00	2	second
5	0.00	0.00	1	zReadRoute

the function to parallelize is obviously *zRoute*. The function has a recursive structure for each possible successor of the city chosen as initial. For instance, if the function would have a *iSize* = 5 (number of cities) then the execution tree would look like the one in Fig 1. In the first 'level' of the tree the first city execute recursively the *zRoute* function

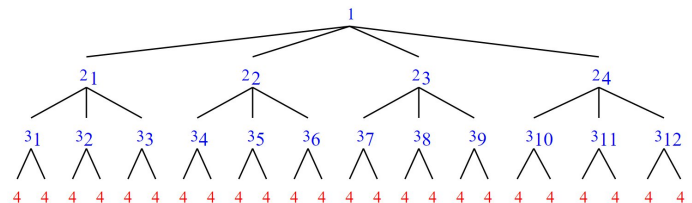


Fig. 1. Execution of the *zRoute* function

### IV. IMPLEMENTATION

#### A. Environment

#### B.

### V. RESULT AND BENCHMARKING

### VI. DISCUSSION

### VII. CONCLUSION

### REFERENCES

- [1] Wikipedia - tsp. [https://en.wikipedia.org/wiki/Travelling\\_salesman\\_problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem), 2015.