Documentation Laboratory 8

Hiticas Claudiu Emanuel
Gr 924 - Year 2

Requirements

Given a directed graph, find a Hamiltonean cycle, if one exists. Use multiple threads to parallelize the search.

Documentation

I solved the problems in C++. The performance is measured on my personal computer with the following config:

Computer specifications

MacBook Air (13-inch, Early 2015)

1.6 GHz Dual-Core Intel Core i5

8 GB 1600 MHz DDR3

Intel HD Graphics 6000 1536 MB

Algorithm

In general, the problem of finding a Hamiltonian cycle is NP-complete, so the only known way to determine whether a given general graph has a Hamiltonian cycle is to undertake an exhaustive search.

First, a hamiltonian cycle is actually a permutation p(0), p(1),..,p(n-1) such that from there is an edge between p(i) and p(i + 1 mod n) for every 0 <= i < n. Another observation comes from the fact that it does not matter from which node you start looking for a hamiltonian cycles, because eventually you will reach that node. An important fact is that the number of edges in the cycles will always be n. In other words, we are looking for a subset of exactly n edges that forms a hamiltonian cycle.

We start with the basic solution (backtracking). The solution basically starts with the first node, `1`, and recursively tries to add new nodes to the current path (based on the neighbours of the current node). When reached `n` nodes, we check if the there is an edge between the first and the last node in the path. Insted of having one thread, we introduce two threads, when choosing which next node to append from the list of nodes of current node. We also need to keep track of the nodes which we selected so far.

Synchronization method

Each thread makes two children threads, so it must wait for them in order to solve it's current problem. Apart from that, each thread is independent

Performance

Number of nodes	Number of threads	Time (ms)
6	2	`0.00710009`
6	4	`0.000988854`
6	8	`0.0145929`
30	2	`16.6115`
30	4	`15.6501`
30	8	**`11.8526`**