ESIGELEC – S8 Project

Overview of an article :

Comparaison d'algorithmes de plus courts chemins sur des graphes routiers de grande taille

In our group, everyone choosed a different orientation in order to get the maximum amount of ideas from various articles.
I choosed to compare various shortest path algorithms.
This research focus on large-sized road graphs and I choosed it in order to fit as much as possible into our own constraints.

Problematic

- First, this research distinguish 3 methods for shortest path :
  - Problem A : From node A to node B
  - Problem B : From node A to every node
  - Problem C : From every node to every node

  An algorithm which can do the problem C can also do the problem B.
  An algorithm which can do the problem B can also do the problem A, due to simplification.
- The research focus on large-sized graphs : The graph is composed of 5000 to 30000 nodes.
- This research is already constrainted to 9 algorithms, considered as efficient for shortest path resolution.
  - Dijkstra without stack
  - Dijkstra with stack
  - Dijkstra with Fibonacci's stack
  - Bucket 1
  - Bucket L
  - Ahuja
  - FIFO
  - Esopo
  - Thresh-X

Approach

The comparaison is divided into several parts.

The first test is based on perfect hexagonnals graph. This testis run with differents sizes (N = 1000, 5000, 10000 and 15000). For each size and for each algorithm, an average of time is made on 20 runs. Then the Dijkstra with stack is used has reference to compare the speed.
Unsurprisingly, the time increase for every algorithm with larger graph. However, this first test shows that Buckets algorithmes (BUCK1 and BUCKL) are quite faster than others.

The second test is based on 5000 nodes graphs but the graphs are randomly made. Each node is connected to a random amount of roads. The average amount of road is : 3, 5, 10, 15, 20 or 30 depending on the version of this test (In the last test with hexagonnals, the amount was always 3).
Dijkstra's algorithm stays slow but not vary that much. Others are disturbed, more or less depending on the algorithm. The BUCK1 stays the best one in this situation. BUCKL is too disturbed here and gets poor results. The Ahuja's algorithm and the Dijkstra's algorithm with Fibonacci's stack are a bit competitive in this context but are still slower tant BUCK1.

2 others tests are realized but without every algorithms considered before.

In the first test, the cost of each road (ie time/distance...) is changed. The previous maximum limit was 1000 and this test takes larger value. Dijkstra would not be influenced by this modification, in opposition to the others which are going to be tested, which are the buckets algorithms. The goal here is to see their weakness and how important it is.
BUCK1 loose against Dijkstra for a maximal cost around 7000 for 5000 nodes, and around 12000 for 10000 nodes.

The last test is used to compare Dijkstra's algorithm with Sedgewick-Vitter's algorithm. This algorithm was considered at first because it only solves the problem A for real graphs.

Eventuals contributions to our project

- BUCK1 is considered as the most performant one. Its major weakness is that it is quite affected by high costs. Another advantage is that it is easy to code.
- The problem C permit the construction of a « distancer », which is matrix NxN (N = the amount of nodes). It allows to find the shortest path between two nodes simply by looking in the matrix (assumed it's precomputed once). The only problem of the «distancer » is that it isn't memory friendly.
- Sedgewick-Vitter's algorithm is considered as the fastest for the problem A in a punctual way. But it is slow when it comes to others problems, so precomputations cannot be done with it.

Reference :
C. Prins, Comparaison d'algorithmes de plus courts chemins sur des graphes routiers de grande taille, *RAIRO Recherche Opérationnelle*