ECOPOINT: SELECTIVE GARBAGE COLLECTION

Bruno M. C. Barros, João P. M. Barbosa and Miriam C. M. C. Gonçalves
Algorithm Design and Analysis
Master in Informatics and Computing Engineering
Faculty of Engineering, University of Porto
Street Dr. Roberto Frias, s/n, 4200-465, Porto, Portugal
{up201405249, up201406241, up201403441}@fe.up.pt

KEYWORDS

graph, smart cities, eco point, Fibonacci heap, Dijkstra, best path, shortest path, weight, efficiency

ABSTRACT

The paper seeks to address the waste picking problem, by trying to find an optimal path passing through a set of garbage containers. In order to find the shortest path, we use Dijkstra's algorithm, modifying it to fit the context, thus enhancing efficiency.

INTRODUCTION

We live in a world where the concept of *Smart cities* is fundamental and universal. These cities try to promote core infrastructures and enhance the quality of life of its citizens, having in consideration a clean and sustainable environment. Currently, every city has a residual waste treatment network. In general terms, not only the countries are trying to reduce the amount of residual waste produced by their population and reintroduce materials into the economy through recycling and composting, but also increase the waste picking companies' productivity.

STUDY'S GOAL

The aim of this study is to discuss and propose a solution, relative to the garbage trucks' routes: how to choose the best path, in order to pick the waste from a set of garbage deposits.

MAIN DEFINITIONS

- Node – represents devices or data points on a larger network. In this study, it represents garbage deposits;
- Edge – connection between two nodes. In this study, it represents a city's street;
- Graph – data structure that consists on a finite set of nodes (or vertices), linked by edges;
- Path – set of edges and nodes that represents the route traveled by a truck;
- Section – a subset of the path;7

THE CONNECTION TO SMART CITIES

One of the *Smart cities* slopes is to improve the waste collection methods. So in order to be smart, these kind of cities must treat and manage the waste they generate, so they can create jobs, save resources, be environmentally friendly and reduce the costs of the picking and treating waste process. Nowadays, collecting waste is done using fixed routes and schedules that requires a lot of manual planning. Containers are collected on a set schedule whether they are full or not. Unnecessary costs, poor equipment utilization, excessive gas emissions and the fossil fuel spending should, therefore, be avoided. By putting sensor technology on containers (to transmit real time information on their level of occupation) and creating optimized routes to collect waste would certainly diminish the impact of the aspects referenced above, and would improve the quality of life of the citizens.

FINDING THE BEST PATH

When finding a route to be used by a garbage truck, our goal is not (only) to find the shortest path, but rather the best path. The "best path" depends on the criteria used: it can take in consideration distance and speed limit of the several roads to be traveled, for example. This way, path finding can be used in a dynamic fashion, that is, the routes used can change, depending on the conditions.

The method used to solve this problem is Dijkstra's algorithm. Initially, we used a binary heap to sort the graph's nodes. Using this method, the time complexity of the algorithm was $(|E| \cdot \log |V|)$ - assuming $|E| > |V|$.

However, a few changes were made to improve its efficiency:

- Instead of a regular binary heap, a Fibonacci heap was used to sort the nodes.
- Let s be the *source* node, and d be the *destination* node, on a certain section of the path. Let also E_i be an edge connecting to d, and $indegree = \sum_i^n E_i$. Meeting these requirements, it is possible to reduce Dijkstra's algorithm running time, by processing the nodes, only until d is visited $indegree$ times.

The use of a Fibonacci heap reduces time complexity to $O(|E| + |V| \cdot \log |V|)$.

The second tweak allows running time to be reduced, depending, however, on the graph layout.

When trying to find the best *section*, our method has the following structure:

Input

- S – starting (source) node
- D – ending (destination) node

Output

- W – set of nodes creating a path from S to D
- E – set of edges, connecting W

Temporary data

- V – set of nodes in the graph
- Adj(v) – set of nodes adjacent to each node v
- Indegree(D) – number of times D is on the destination end of an edge

Time performance

Best case: $O(1)$

Worst case: $O(|E| + |V| \cdot \log |V|)$

To find the best route, that processes n garbage deposits, the method above must be repeated n times - unless some of the containers are already in the section between two nodes. Therefore, in the worst case, time complexity will be $O(n \cdot |E| + n \cdot |V| \cdot \log |V|)$.

CONNECTIVITY

In our solution, there is no preemptive checking if the graph on the surrounding area of the garbage deposits to be picked is strongly connected, i.e. if for each container, there is a path to every other container on the given area. It is assumed that a route should be found, independently of the graph's connectivity.

When facing this situation, and there is no path connecting two nodes, *source* and *destination*, then the algorithm ignores the destination node, and attempts to generate a new path between the source node and the next node on the process list.

DIJKSTRA'S VS FLOYD-WARSHALL

While designing our solution, we had to choose, between several algorithms, the most efficient option. We believe the best candidates for satisfying this problem's requirements are Dijkstra's algorithm and Floyd-Warshall algorithm. The method we chose, as mentioned above, was the first one.

The Floyd-Warshall algorithm finds the lengths of the shortest paths between all pair of nodes, with a running time complexity of $O(|V|^3)$ independently of the graph arrangement. On the other hand, the Dijkstra's algorithm calculates the shortest path from one initial node to the others on the graph, having a running time complexity of $O(|E| + |V| \cdot \log |V|)$. When trying to find the optimal route for a truck to cover n eco points the running time complexity of Dijkstra's will increase to $O(n \cdot |E| + n \cdot |V| \cdot \log |V|)$. For $n = |V|$, the algorithm will perform with complexity $O(|V| \cdot |E| + |V|^2 \cdot \log |V|)$,

which is lower than $O(|V|^3)$ iff $|E| < |V|^2$. Note that this restriction will always be satisfied, because $|E| = |V| \cdot (|V| - 1)$, in a complete graph.

We can conclude from this analysis, that, overall, Dijkstra's algorithm will be theoretically faster - in practice, for $|E|$ close to $|V|^2$, Floyd-Warshall will generally have a better performance.

It should be noted that, although the running time complexity of Floyd-Warshall's is higher, a single execution of this algorithm would allow us to have the information of every shortest path for all the nodes of the graph elegantly, which would be ideal for a static model representation.

However, on a dynamic model (taking in consideration some external factors as traffic, weather, public constructions), the shortest path might vary, and so the algorithm should perform accordingly. In this representation, the Dijkstra's algorithm has better performance.

ROOM FOR IMPROVEMENT

Every city and country can be divided in high and low activity zones. In certain conditions, it would be reasonable to assume that some less crowded zones have a constant level of activity, i.e. external factors - amount of people, traffic flow, and overall movement - could be discarded. This means, we can approximate the zone to a static model, therefore, using Floyd-Warshall algorithm would become ideal.

Having in possession certain data of a given city or country, it would be possible to create a solution where the city could be divided in "dynamic" and "static", thus improving the overall efficiency of the path finding algorithms:

- In crowded areas, with a substantial traffic flow we would use a dynamic model. In other words, our solution would be based on Dijkstra's algorithm.
- In rural areas or less crowded zones, with a low level of activity, we would use a static model, which means we would use Floyd-Warshall's method.

Secondly, since we still run our algorithm on weakly connected sections of the graph, it is possible that the path generated is not always the most desirable: it depends on the order on which the nodes are being visited (if the first node being processed does not have any outgoing edges, then no path can be found from that point). Note that the route will always be the shortest, but may not always cover most of the containers – as said before, it is assumed that a route should be generated nonetheless, but, if needed, it would be possible to previously check if the route can visit all wanted deposits.

CONCLUSION

The aim of this study was to develop a solution to the picking waste problem of creating the most efficient and shortest route. Our solution was created having always in mind efficiency and usability.

We hope that our project and study will lead to further discussion and research so it can be turn into a reality in every single country, in order to cities to be smarter and more environmentally sustainable.

REFERENCES

Sedgewick, Robert; *Algorithms in C++ Part 5: Graph Algorithms*, 3/E, Addison-Wesley Professional, 2001. ISBN: 0201361183.

Gama, Rui; Fernandes, Ricardo; *Cidades inteligentes, inteligência territorial e criatividade em Portugal.* Cadernos de Geografia nº28/29 – 2009/10.

Gomes, Tiago Manuel Ribeiro; *WECO – Wireless Ecoponto*, Tese de Mestrado em Engenharia de Comunicações.

Andrew V. Golberg et al. *Efficient Point-to-Point Shortest Path Algorithms*.