Twice-around a Shortest-path Tree Significantly Increases the Solution Cost

Prof. Dr. Alexandre L. M. Levada, Lucas O. David

***Abstract*—In the Graph Theory field, one of the most important problems is the Traveling Salesman Problem. Many algorithms have been developed to solve or approximate this problem, with one of them being the Twice-around Algorithm. This paper describes the behavior of the Twice-around algorithm when it has as input a shortest-path tree instead of its default minimum spanning tree. Through empirical experimentation, we discovered that the modified Twice-around consistently presented poor solutions compared to the original algorithm.**

***Keywords*—Graph Theory, Traveling Salesman Problem, Twice-around.**

I. Introduction

In our world, it is possible to model a considerable set of problems using the Traveling Salesman Problem (TSP), making it one of the most popular existing algorithms. The study of methods that solve or satisfactorily approximate a solution is of great importance, as it would imply great development in many different areas of knowledge that depend on computational feasibility.

Currently, many different approaches are employed to solve the TSP. While some will seek the optimal answer through a smart (but possibly exhaustive) search, as Branch-and-bound or the Held-Karp algorithm, others will base themselves on heuristics to find a candidate-solution with reasonable cost (e.g. Lin-Kernighan or ant community optimization [2]).

This project aim to develop the basic algorithms involved (e.g. Dijkstra's, Prim's, Depth-First search, Twice-around), as well as an environment capable of creating or collecting instances of the problem from a database and running test between the original and the modified Twice-around algorithms. All results and necessary implementations can be found on this code repository: tsp.

II. The Traveling Salesman Problem

Let $G = (X, E)$ be the k-complete symmetric weighted graph, where X is a set of cities and E the set of edge that represents the distances between different cities. The Traveling Salesman Problem (TSP) has as goal find the Hamiltonian Circuit such that the sum of edges' weights is minimum.

III. The Twice-around Algorithm

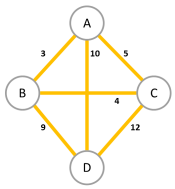
The Twice-around algorithm is an heuristic broadly employed when solving the TSP.

Alexandre L. M. Levada and Lucas O. David are with Universidade Federal de Sao Carlos.

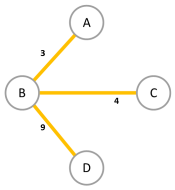
Let G be defined as in section II, the twice-around algorithm will transverse a minimum spanning tree extracted from G using a search algorithm such as **Depth-First Search** while records the vertices visited. It will then proceed to remove the repetitions in the recorded sequence, expect for last vertice [1].

A. Twice-around Execution Example

Let G be the graph defined in figure 1a, A the starting node and M the minimum spanning tree extracted from G.



(a) The graph G.



(b) The MST M.

M will be transversed and the visiting sequence $L_1 = (A, B, D, B, C, B, A)$ will be recorded. Finally, the repeated vertices are removed. This elimination process can be informally interpreted as "taking shortcuts". The following figure illustrates how removing the first repetition of B from L_1 will link the vertices C and D.

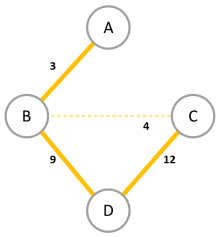


Fig. 2: MST M with the first B repetiton removed.

By eliminating all the repetitions, the algorithm will result in a Hamiltonian circuit candidate $L = (A, B, D, C, A)$ with cost $c_L = 100$:

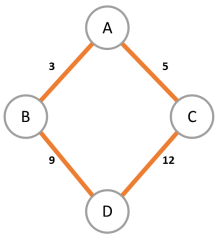


Fig. 3: The Hamiltonian circuit candidate L.

Theorem III.1. *$c_L \leq 2c_{L^\star}$, where c_P is the cost of walking on a circuit P, L is the Hamiltonian circuit candidate found and $c_{L^\star}$ is the optimal Hamiltonian circuit.*

Proof: Let T be the spanning tree resulted from removing any edge from $L^\star$ and M be the minimum spanning tree.

$$c_{L^\star} > c_T \geq c_M \iff 2c_{L^\star} > 2c_M$$

The triangular inequality $\delta_{uv} \leq \delta_{uw} + \delta_{wv}, \forall (u,v,w) \in V^3$ holds, as this graph was built over euclidean distances. Therefore,

$$2c_M \geq c_L$$
$$\therefore 2c_{L^\star} > c_L$$

■

It's also important to remember that any shortcut done in the MST m will not decrease the sum of costs from the edges of the resulting tree, compared to the sum of edges' costs from the edges in m. The example above demonstrates this: the cost of the MST is $3+4+9=16$, whereas the cost of the MST after the firsts shortcut is $5+9+12=26$.

Theorem III.2. *Let $m = (V,F)$ be the MST extracted from graph k-complete $G = (V,E)$, m' the path found by shortcutting all branches of m, δ_{AB} the distance between vertices A and B, and $c(p) = \sum \delta_{X_iY_i}, p = (V, \{X_iY_i\})$.*

$$c(m') \geq c(m)$$

Proof: If the MST m contains two edges $AB, AC \in F$ (a branch), leading to two different paths $B...B'$ and $C \dots C'$, $m'' = (V, F - \{AC\} + \{B'C\})$,

$$B'C \notin F, AC \in F \implies \delta_{B'C} \geq \delta_{AC}$$
$$c(m'') = c(m) - \delta_{AC} + \delta_{B'C} \geq c(m)$$

Now define $m' = (V,L) \mid L = F - \sum A_iC_i + \sum B'_iC_i$, where $B = \{(A_iB_i, A_iC_i)\}$ is the set with all branches in m.

$$\begin{aligned} c(m') &= c(F - \sum\{A_iC_i\} + \sum\{B'_iC_i\}) \\ &= c(V,F) - \delta_{A_0C_0} + \delta_{B'_0C_0} \\ &\quad - \delta_{A_1C_1} + \delta_{B'_1C_1} + \dots \\ &\quad - \delta_{A_nC_n} + \delta_{B'_nC_n} \end{aligned}$$

But $\delta_{A_iC_i} \leq \delta_{B'_iC_i}, \forall i \in [0, |B|)$, which implies

$$c(m') \geq c(V,F) = c(m)$$

B ■

IV. Dijkstra's Algorithm

Differently from the minimum spanning tree, the **shortest path tree** associated with a graph G and the node g_0 express the shortest path between the origin g_0 and all the other vertices. In the optimization context, the term "shortest" refers to the minimum cost required to travel on a path between two vertices.

Theorem IV.1. *Let $G = (V,E)$ be the k-complete graph and $E = \{\delta_{xy}\}, (x,y) \in V^2$ the set of egdes built over euclidean distances. The shortest-path tree with origin v_0 is the **star** S_{k-1} with center v.*

Proof: The shortest-path tree $T = (V_1, E_1)$ is such that the path cost between two vertices u and v is minimum. But $\delta_{uv} < \delta_{uw} + \delta_{wv}, \forall (u,v,w) \in V^3, u \neq v \neq w \neq u$ holds in euclidean spaces, so $T = (V_1, \{\delta_{v_0u}, \forall u \in V_1\})$, where V_1 is the set of all vertices in the connected cluster in which v_0 is placed. ■

V. Experiments

In order to observe the behavior of the Twice-around algorithm when it has as input a SPT instead of a MST, a analyzer was built. Given a set of S-TSP instances, it will solve all of them with both original and modified Twice-around. Information about the solutions is recorded in each iteration (e.g. costs, elapsed time, best performing algorithm). At the end, a report containing all data collected is issued.

The comparison between the two algorithms was done by three phases, with their respective set of problems.

A. Phase I: Randomly Generated Euclidean Graphs

Figure 4 illustrates that the modified Twice-around performs worse than the original one in all situations. Additionally, the cost disparity of the solutions quickly increased as the number of vertices grew.

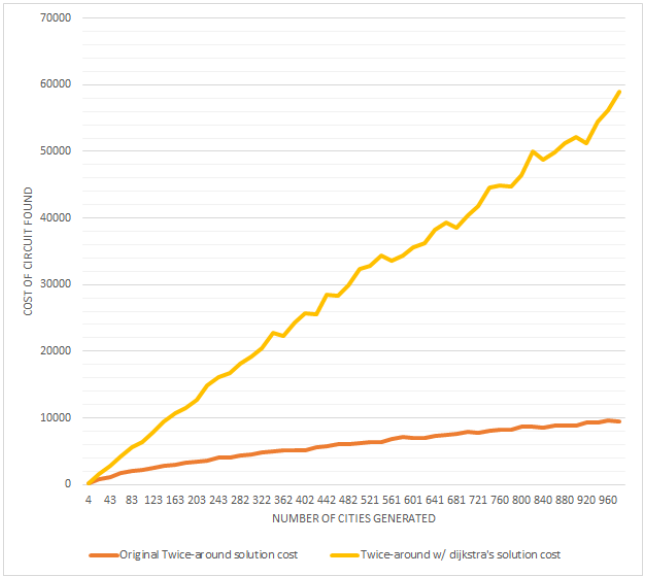


Fig. 4: Solutions' costs for random euclidean graphs.

B. Phase II: Real Euclidean Graphs

Figure 5 shows the disparity between costs found by both original and modified Twice-around when applied to real instances of the TSP problem. In all four cases, the original Twice-around presented a much better solution than the Twice-around with dijsktra's.

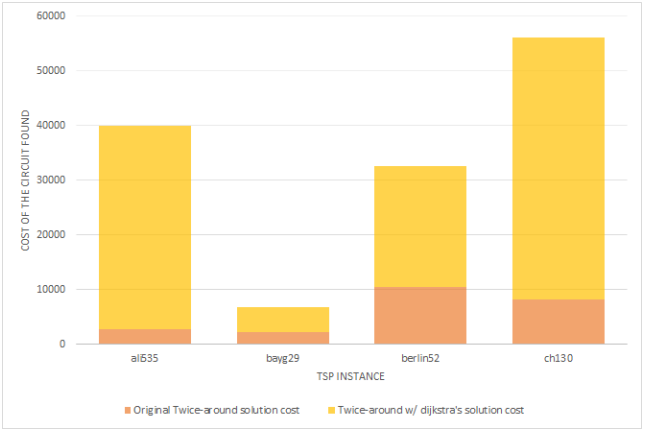


Fig. 5: Solutions' costs for real graphs.

C. Phase III: Randomly Generated Non-euclidean Graphs

Figure 6 illustrates that the modified Twice-around consistently presented a worse solution than the original algorithm in non-euclidean random graphs.

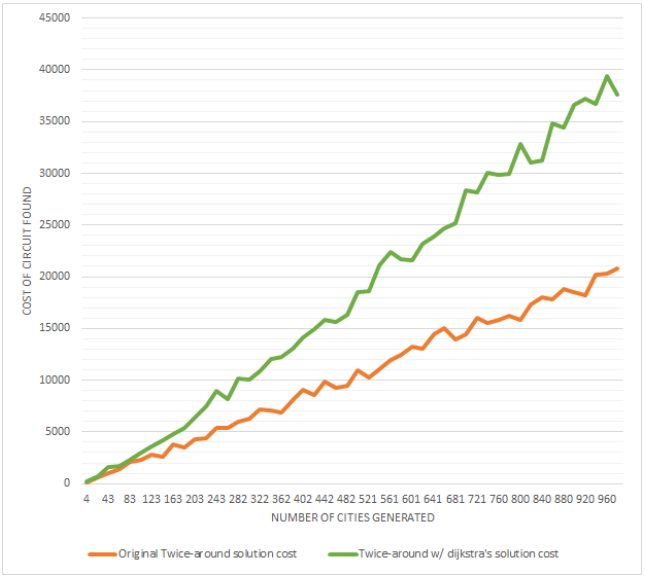


Fig. 6: Solutons's costs for non-euclidean random graphs.

VI. Conclusions

From IV.1, the shortest path tree found in euclidean graph is a star. The recorded visiting list is then be described as $L_1 = (A, B, A, C, A, D, A, E, ..., A)$. With the repetitions removed, the Hamiltonian circuit candidate is $L = (A, B, C, D, E, ..., A)$ with cost $c_L = \sum_{i=0}^{n-2} c(V_i, V_{i+1}) + c(V_{n-1}, V_0)$. Notice that no lower bound can be defined for cost c_L. As for shortest-path trees extracted from non-euclidean graphs: although they are not necessarily stars, we observed a tendency of those to have a high number of branches.

If the tree inputed in Twice-around has few branches, many sub-paths are preserved when building the circuit candidate (few shortcuts are required). In the other hand, as we are required to take many shortcuts in trees with many branches, such as the **star**, few sub-paths are preserved and the solution cost can drastically change (from III.2, shortcutting increases cost). We believe this to be the main reason for the poor performance of the modified Twice-around.

References

[1] C. Nilsson. Heuristics for the traveling salesman problem. Technical report, Tech. Report, Linköping University, Sweden, 2003.

[2] R. Soricone and M. Neville. Comparative analysis of genetic algorithm implementations. In *ACM SIGAda Ada Letters*, volume 24, pages 35–38. ACM, 2004.