An algorithm for finding a similar subgraph of all Hamiltonian cycles

R Wafdan\textsuperscript{1,3}, M Ihsan\textsuperscript{2,3} and D Suhaimi\textsuperscript{2,3}

\textsuperscript{1} Graduated Mathematics Program, Syiah Kuala University, Banda Aceh, Indonesia
\textsuperscript{2} Department of Mathematics, Syiah Kuala University, Banda Aceh, Indonesia
\textsuperscript{3} Multimedia Research Group, Department of Mathematics, Syiah Kuala University, Banda Aceh, Indonesia

E-mail: mahyus@unsyiah.ac.id

Abstract. This paper discusses an algorithm to find a similar subgraph called \textit{findSimSubG} algorithm. A similar subgraph is a subgraph with a maximum number of edges, contains no isolated vertex and is contained in every Hamiltonian cycle of a Hamiltonian Graph. The algorithm runs only on Hamiltonian graphs with at least two Hamiltonian cycles. The algorithm works by examining whether the initial subgraph of the first Hamiltonian cycle is a subgraph of comparison graphs. If the initial subgraph is not in comparison graphs, the algorithm will remove edges and vertices of the initial subgraph that are not in comparison graphs. There are two main processes in the algorithm, changing Hamiltonian cycle into a cycle graph and removing edges and vertices of the initial subgraph that are not in comparison graphs. The \textit{findSimSubG} algorithm can find the similar subgraph without using backtracking method. The similar subgraph cannot be found on certain graphs, such as an \(n\)-antiprism graph, complete bipartite graph, complete graph, \(2n\)-crossed prism graph, \(n\)-crown graph, \(n\)-möbius ladder, prism graph, and wheel graph. The complexity of this algorithm is \(O(m|V|)\), where \(m\) is the number of Hamiltonian cycles and \(|V|\) is the number of vertices of a Hamiltonian graph.

1. Introduction

The problem of finding Hamiltonian cycle is a case to find a cycle that contains all vertices of a Hamiltonian graph. This problem has become a challenge to mathematicians. Even though it is a classic and complicated problem, more than a few mathematicians choose this problem as their research objects. The Hamiltonian cycle problem is not only an important problem in graph theory but also can be used in real-life application such as telecommunication and computer network [1]. Several algorithms have been developed using backtracking or heuristic approach to find Hamiltonian cycles [2]. One of the algorithms that are able to find Hamiltonian cycle is \textit{findSegment} algorithm [3].

The \textit{findSegment} algorithm uses backtracking approach to find the solution. The \textit{findSegment} algorithm works by collecting edges that form Hamiltonian cycle called \textit{segments}. Furthermore, the algorithm will delete edges that are not needed to reduce the search space [3]. By using some modifications, the \textit{findSegment} algorithm can be used to find all Hamiltonian cycles of a Hamiltonian graph.

In addition to finding a Hamiltonian cycle, another thing needed to find a Hamiltonian cycle is finding a similar subgraph. The similar subgraph is a subgraph with a maximum number of edges, contains no isolated vertex and is contained in every Hamiltonian cycle of a Hamiltonian Graph. A
graph $G' = (V', E')$ is a subgraph of another graph $G = (V, E)$ if and only if the vertex $V'$ and edge $E'$ sets of $G'$ are, respectively, subsets of the vertex $V$ and edge $E$ sets of $G$ [4]. Finding the similar subgraph is an important case to be analyzed because it is often found in real life such as products distribution using box trucks. Box trucks distribute the products to the same consumers through different routes. However, on some occasions, the trucks use the same routes to distribute products to its destinations. The goal of this research is to find such routes. By finding the routes, the best policy can be made by authorities to avoid a traffic jam or to improve its quality.

In graph theory, if we consider the routes as a Hamiltonian cycle, the routes used by every box trucks can be seen as subgraphs of every Hamiltonian cycle. Applying subgraphs in transportation is not a new problem. Flinsenberg in his book on route planning algorithms for car navigation [5] used subgraphs called cells in his algorithm to divide road graph, that is, a road network represented by a multi-graph. Subgraph is also used by Yousefi and Zamani in their paper [6], they considered that a network graph was composed of several subgraphs. Learning the overall condition of the graph is possible by conducting several searches over each subgraph. This procedure helps discover the shortest path and avoid choosing paths randomly. In addition, Latora and Marchiori, who research about Boston Subway, also used the concept of subgraph [7]. They stated that the MBTA (Boston underground transportation system) is not a closed system, as it can be considered, after all, a subgraph of a wider transportation network, and they concluded that fault tolerance is not a critical issue. In this research, an algorithm to find subgraph that is on every Hamiltonian cycle will be proposed. If it is applied to transportation problem, the algorithm could find routes taken by all trucks of an industry to distribute its products, so the quality of such routes can be improved. This algorithm requires findSegment Algorithm to find Hamiltonian cycles before finding a similar subgraph.

2. Methodology
The algorithm of finding a similar subgraph takes the set of solution (every Hamiltonian cycle) of a Hamiltonian graph $H = (V, E)$ as its input. However, the following condition must be met:
The resulting subgraph contains no isolated vertex. Otherwise, the algorithm will not affect anything since every Hamiltonian cycle contains every vertex of a Hamiltonian graph.
The algorithm runs only on Hamiltonian graphs that contain more than one Hamiltonian cycle. On a graph with only one Hamiltonian cycle, the algorithm has no effect since the similar subgraph will be the Hamiltonian cycle itself.

For example, the Hamiltonian graph shown in Figure 1 contains four different Hamiltonian cycles, namely $(1,3,2,5,7,8,6,4,1)$, $(1,3,2,5,8,7,6,4,1)$, $(1,3,6,7,8,5,2,4,1)$, and $(1,3,6,8,7,5,2,4,1)$ as shown in Figure 2.

Figure 1. Hamiltonian graph.
Figure 2. Hamiltonian cycles.

From all four Hamiltonian cycles, a similar subgraph is obtained as shown in Figure 3.

Figure 3. Subgraph of Hamiltonian cycles.

The following is the finding algorithm of the similar subgraph of the graph as shown in Figure 1.

After taking the set of Hamiltonian cycles from findSegment algorithm, set index variable $i = 1$.

Change the $i$-th Hamiltonian cycle into cycle graph $HcG$ and then set the $i$-th Hamiltonian cycle as initial subgraph $SubG$ (see Figure 4).

| $i$ | Hamiltonian Cycle |
|-----|-------------------|
| 1   | (1,3,2,5,7,8,6,4,1) |
| 2   | (1,3,2,5,8,7,6,4,1) |
| 3   | (1,3,6,7,8,5,2,4,1) |
| 4   | (1,3,6,8,7,5,2,4,1) |

Figure 4. Initial subgraph.
- Set $i = i + 1$
- Change the $i$-th Hamiltonian cycle into cycle graph $HcG$.
- Check whether the initial subgraph $SubG$ is a subgraph of the cycle graph $HcG$ (see Figure 5).

![Image](image1.png)

**Figure 5.** Check Subgraph.

- If not, delete the vertices and edges of the initial subgraph that are not in the cycle graph of the $i$-th Hamiltonian cycle (see Figure 6).

![Image](image2.png)

**Figure 6.** The new initial subgraph.

- Otherwise, go to step 3 until every vertex has been compared.
- The result of the algorithm is an initial subgraph that is in every Hamiltonian cycle (see Figure 7).

![Image](image3.png)

**Figure 7.** The final initial subgraph.

3. **Results and Discussion**
The first step to finding a similar subgraph is finding every Hamiltonian cycle in Hamiltonian graph. The set of Hamiltonian cycles is called $HcList$. The set $HcList$ will then be the input to the algorithm.
The finding algorithm is called $\text{findSimSubG}$. In general, the subgraph finding algorithm of a Hamiltonian cycle works as follows.

\begin{verbatim}
findSimSubG(HcList: Set of Hamiltonian cycles): graph
begin
    set subG := convertToGraph(HcList[1])
    for i := 2 to |HcList| do
        set HcG := convertToGraph(HcList[i])
        if subG \ HcG then
            set subG := removeVE(subG, HcG)
        end
    end
    findSimSubG := subG
end;
\end{verbatim}

The $\text{findSimSubG}$ algorithm consists of two processes, $\text{convertToGraph}$, and $\text{removeVE}$. $\text{convertToGraph}$ is used to change a Hamiltonian cycle into cycle graph form. The algorithm takes Hamiltonian cycles as the input and outputs a cycle graph. In general, the process $\text{convertToGraph}$ can be described by the following algorithm.

\begin{verbatim}
convertToGraph (HamCy : Hamiltonian cycles) : graph
begin
    for i := 1 to |HamCy|-1 do
        set V := V \ {HamCy[i]}
    end
    for i := 2 to |HamCy| do
        set E := E \ {{HamCy[i-1], HamCy[i]}}
    end
    set HcG := (V, E)
    convertToGraph := HcG
end;
\end{verbatim}

The $\text{removeVE}$ process is a process to delete edges that are in $\text{subG}$ but not in graph $\text{HcG}$ and to delete vertices in $\text{subG}$ with degree 0. In the $\text{removeVE}$ process, the input of the algorithm is a subgraph $\text{subG}$ and a cycle graph $\text{HcG}$. The algorithm will output a subgraph. In general, the process $\text{removeVE}$ can be described as the following algorithm.

\begin{verbatim}
removeVE(subG : Subgraf, HcG : Graf) : graph
begin
    { Let subG = (Vs, Es) and HcG = (V, E) }
    for each (u, v) \in Es do
        if (u, v) \in E then
            set Es := Es \ {(u, v)}
        else (v, u) \in E then
            set Es := Es \ {(v, u)}
        end
    end
    for each v \in Vs do
        if deg(v) = 0
            set Vs := Vs \ {v}
        end
end
\end{verbatim}
The \textit{findSimSubG} algorithm is tested on eight well-known Hamiltonian graphs, namely \textit{n}-antiprism graph, complete bipartite graph \(K_{n,n}\), complete graph \(K_n\), \textit{2n}-crossed prism graph, \textit{n}-crown graph, \textit{n}-möbius ladder, prism graph \(Y_n\) and wheel graph \(W_n\). The aforementioned graphs have no the \textit{similar subgraph}.

The \textit{convertToGraph} algorithm consists of two \textit{for} loops, each \textit{for} loop executes \(|V|\) times and there are two processes on the last line of the algorithm. Thus, the total time for the \textit{convertToGraph} algorithm is \(O(2|V|+2) = O(|V|)\) where \(|V|\) is the number of vertices in the Hamiltonian graph. The \textit{removeVE} algorithm also consists of two \textit{for} loops, each \textit{for} loop executes \(|V|\) times and there is a process on the last line of the algorithm. Thus, the total time for the \textit{removeVE} algorithm is \(O(2|V|+1) = O(|V|)\). The \textit{findSimSubG} algorithm consists of calling \textit{convertToGraph} process one time outside the \textit{for} loop that is \(O(|V|)\), calling \textit{convertToGraph} process one time inside the \textit{for} loop that is \(O(m|V|)\) where \(m\) is the number of Hamiltonian cycles found in Hamiltonian graph, calling \textit{removeVE} process one time inside the \textit{for} loop that is \(O(m|V|)\), and there is a process on the last line of the algorithm. The total time for \textit{findSimSubG} is \(O(|V| + m|V| + m|V| + 1) = O(m|V|)\).

4. Conclusion

The \textit{findSimSubG} algorithm is able to find a \textit{similar subgraph} without using backtracking process. Not all Hamiltonian graphs contain the \textit{similar subgraph}. The test using the \textit{findSimSubG} algorithm on eight well-known Hamiltonian graph shows that the graphs have no the \textit{similar subgraph}. If the finding the \textit{similar subgraph} algorithm is applied to transportation problem, then the algorithm could find routes taken by all trucks of an industry to distribute its products, so the quality of such routes can be improved. The limitation of this algorithm is it only works on undirected graphs, while most of the real-life problems will likely be on directed graphs.

Acknowledgments

The authors express gratitude to Directorate of Research and Community Services for grant of Applied Products Research.

References

[1] Serra M and Kent K 2003 Proc. of Int. Symp. on Circuits and systems 3 228-231
[2] Riaz K and Khiyal M S H 2006 Information Technology Journal 5 851-859
[3] Ihsan M, Waldan R, Hafnani, Halfiani V and Ramli M 2016 Far East J. Math. Scie. 100 pp 573-587.
[4] Goodaire E G and Parmenter M M 2002 Discrete Mathematics with Graph Theory, (New Jersey: Prentice-Hall)
[5] Flinsenberg I C M 2004 Route Planning Algorithms for Car Navigation (Eindhoven: Technische Universiteit)
[6] Yousefi P and Zamani R 2013 Inter. J. Mach. Learn. Com. 3 p 44
[7] Latora V and Marchiori M 2002 Physica A: Statistical Mechanics and its Applications 314 l pp 109-113