A Development of Travel Itinerary Planning Application using Traveling Salesman Problem and K-Means Clustering Approach

Septia Rani, Kartika Nur Kholidah, Sheila Nurul Huda
Department of Informatics, Universitas Islam Indonesia
Yogyakarta, Indonesia
septia.rani@uii.ac.id, kartikakholidah@gmail.com, sheila@uii.ac.id

ABSTRACT

In this paper, an algorithm for making travel itinerary using traveling salesman problem (TSP) and k-means clustering technique is proposed. We employ the algorithm to develop a web based application that can help travelers to plan their travel itinerary. The developed application should be able to provide an optimal itinerary recommendation in terms of distance and travel time. We use initial assumption that the traveler has determined all the tourist destinations he/she wants to visit and also the number of days he/she will stay in the region. Our approach consists of two steps, macro grouping using k-means and micro tour arrangement using TSP. Yogyakarta city, one of the tourist city in Indonesia, is used as an example to illustrate how the proposed algorithm can help travelers make their itinerary. This approach works well in small to medium number points of interest. However, the application still need many improvements such as to make it run faster and to handle the additional constraints that exist when creating an itinerary.

CCS Concepts

• **Computing methodologies → Planning for deterministic actions**

Keywords

Travel itinerary; traveling salesman problem; k-means clustering.

1. INTRODUCTION

Itinerary is a planned route or journey. For travelers, travel itinerary usually contains a schedule of intended destinations and activities. After selecting the tourist destinations to be visited, the process of arranging the travel tour route (making itinerary) becomes the next challenging problem to be solved.

The travel itinerary planning problem is unstructured in real life, which means goals, decision criteria and other constraints of the problem are unable to completely predefined [1]. However, we can identify and define some of the problems that travelers often encounter when making an itinerary. One of them is when the traveler is not familiar or lack of information about the locations he/she wants to visit, then he/she must look for the tourist destination locations one by one to estimate the order of the visit. The searching and mapping process itself certainly can take a long time. Moreover, if the traveler decides to spend more than one day in the destination city, then he/she must also determine the schedule of tourist visits for each day.

In recent years, many computer-based applications are utilizing artificial intelligence (AI). Russell and Norvig [2] categorized the definitions of AI into four categories: systems that think like humans, systems that think rationally, systems that act like humans, and systems that act rationally. Some of the problems that are often solved using AI is optimization and decision making problem. The travel itinerary planning problem is one of the problems included in that category. Therefore, in this research, we will develop an application to solve the travel itinerary planning problem using artificial intelligence approach.

In this study, the developed application should be able to provide an optimal itinerary recommendation in terms of distance and travel time. We use initial assumption that the traveler has determined all the tourist destinations he/she wants to visit and also the number of days he/she will stay in the region. There are two sub-issues to solve. The first is how to cluster the tourist destinations that will be visited for each day. While the second is how to arrange the order of visits on a daily basis so that the total travel distance will be minimum.

The problem of grouping the visited tourist destinations in each day can be solved by employing k-means clustering, where the nearby tourist destinations will be grouped to be visited on the same day. While the problem of optimization of the visit order (tour arrangement) can be solved by using the traveling salesman problem (TSP) approach.

TSP can simply be stated as: if a traveling salesman wishes to visit exactly once each of a list of m cities (where the cost of traveling from city i to city j is c_{ij}) and then return to home city, what is the least costly route the traveling salesman can take? [3]. In this study, salesman can be analogized as a traveler and the visited cities are tourist destinations in a region. We use Yogyakarta city, one of the tourist city in Indonesia, as an example to illustrate how the proposed algorithm can help travelers make their itinerary. With the technique proposed in this study, it is expected that the application prototype will be able to help traveler in arranging a better travel itinerary, which is more efficient in terms of distance and travel time.

The rest of the paper is organized as follows. Section 2 describes some preliminaries about traveling salesman problem, k-means clustering, and review on travel itinerary planning application. Section 3 describes the methodology. Section 4 describes the

ICSCA 2018, February 8–10, 2018, Kuantan, Malaysia

ACM ISBN 978-1-4503-5414-1/18/02…$15.00
https://doi.org/10.1145/3185089.3185142

results. Finally, the conclusion and future work are described in section 5.

2. PRELIMINARIES

2.1 Traveling Salesman Problem

Traveling salesman problem (TSP) is one of the most famous problems in graph theory and combinatorics. Mathematical problems related to the TSP were firstly introduced in the 1800s by the Irish mathematician Sir William Rowan Hamilton and by the British mathematician Thomas Penyngton Kirkman [4]. TSP is an optimization problem and can simply be stated as: if a traveling salesman wishes to visit exactly once each of a list of m cities (where the cost of traveling from city i to city j is c_{ij}) and then return to home city, what is the least costly route the traveling salesman can take? [3]. According to graph theory, city can be represented as a graph node, while the road that connects between two cities can be represented as edge. The edge's weight represents the distance between two cities. TSP is none other than determining the Hamilton circuit that has the minimum weight on a connected graph. The Hamilton circuit itself is a closed path that pass each node/vertex in the graph exactly once, except the origin node (which is also the final node) that is passed twice.

Computationally, TSP has high time complexity. In any complete graph with n vertices ($n > 2$), the number of different Hamilton circuits is $(n-1)!/2$. We can imagine if the number of vertices to be visited are very huge, the time needed to evaluate the possible solutions (to obtain the best solution) will be very long. Therefore, beside using deterministic methods, many studies have developed methods for solving TSP using heuristic methods such as genetic algorithms [5][6] or swarm intelligence [7][8].

In a case of TSP with small number of vertices, a brute force (exhaustive enumeration) approach can be used to find the order of visits with a minimum total distance. Figure 1 shows an example of a TSP case with 4 vertices and also the solution using brute force method.

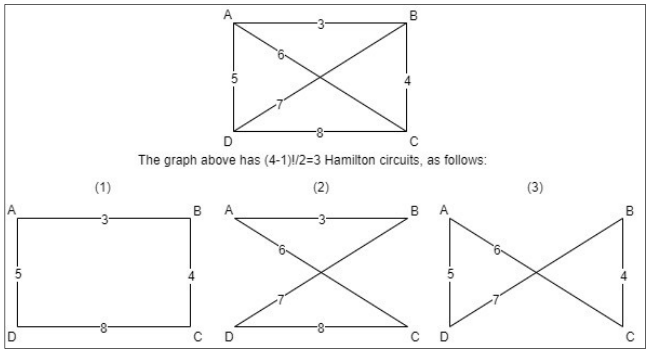

Figure 1. The example of TSP.

Based on the example in Figure 1, the shortest Hamilton circuit is shown by the leftmost graph. The leftmost graph has the order of visits as follows: (A, B, C, D, A) or (A, D, C, B, A) with the length of the circuit is $3 + 4 + 8 + 5 = 20$. This solution is relatively shorter when compared with the next two graphs that have the length of circuits 24 and 22. So, when a salesman visits each city according to the circuit on the leftmost graph, then he can save the cost of travel.

2.2 K-Means Clustering

K-means is an algorithm used for clustering which will divide data into several groups. K-means algorithm is one of the non-hierarchical clustering methods that can group data into several groups based on similarity of data. This mechanism allows data that have the same characteristics grouped into one cluster and data that have different characteristics grouped in other clusters.

To determine the cluster label of the data, the distance between data with each cluster center is calculated. There are several ways that can be used to perform distance calculations, such as Euclidean distance, Manhattan distance, and Chebichey distance. The k-means method aims to minimize the sum of squares of distance between all data points with the center of each cluster. The procedure of k-means consists of the following steps, as described in [9].

- Step 1: Select k out of the given n patterns as the initial cluster centers. Assign each of the remaining $n - k$ patterns to one of the k clusters; a pattern is assigned to its closest center/cluster.
- Step 2: Compute the cluster centers based on the current assignment of patterns.
- Step 3: Assign each of the n patterns to its closest center/cluster.
- Step 4: If there is no change in the assignment of patterns to clusters during two successive iterations, then stop; else, go to Step 2.

K-means algorithm has several advantages such as simple, fast, and easy to implement. Nevertheless, the output of k-means relies heavily on a randomly determined initial centroid. Therefore, in practical applications k-means must be run several times with different initial centroids to produce the final centroid that is considered the best.

2.3 Review on Travel Itinerary Planning Application

With the growing capabilities in technology nowadays, there are many research and applications to facilitate the travelers [10]. Most of them are applications that provide recommendations and reviews about destinations of interest to visit, such as TripAdvisor application. There are still rarely founded applications that can be used where the case is user has determined the destinations that he/she wants to visit and begin to arrange or schedule the order automatically. Usually the prospective travelers must arrange their own schedule and then record it in the applications that can help them keep the itinerary records.

Based on references found on the Internet, one of the itinerary planner application is Eightydays which is an application that helps travelers to design a travel plan to visit several cities in Europe [11]. This app gives tourists some available options when going on a trip to Europe. The app developer realizes that arranging travel plan manually is usually time consuming. By using Eightydays, user only needs to enter a city where the journey will begin and finish. Then user must also enter the date of departure, duration of travel and the number of cities they want to visit. The app will automatically generate itinerary along with flight information and accommodation information in each city.

Eightydays is considered as macro planner, which plans visit to several cities. Whereas in one city alone tourists still need to make itinerary. Therefore, it is necessary to develop additional applications that can solve this problem, so that prospective travelers no longer spend a lot of time to design an efficient itinerary.

Another more sophisticated tour planner system is Dynamic Tour Guide (DTG) [12]. Points of interest in DTG are called as Tour

Building Blocks (TBBs) and described by ontology. DTG calculates the similarity between TBBs and user profile to select which point of interest the user should visit during his/her stay. In our approach, the user can select the points of interest which he/she wishes to visit, then the points of interest will be clustered for each day planning.

3. METHODOLOGY

In this study, we choose Yogyakarta city, one of the tourist city in Indonesia, as an example to illustrate how the proposed algorithm can help travelers make their itinerary. Yogyakarta has a lot of tourism destinations. Moreover, based on statistical data of Yogyakarta Province Tourism Office [13], the number of tourists continues to increase from 2012 to 2016. This trend indicates that in the future Yogyakarta will attract more travelers. To encourage the development of Yogyakarta as a tourist destination, it should be supported not only by the government but also by the creative industry sector such as software industry.

Prior to modeling the system, we performed a system requirement analysis. We identify some inputs required by the application. They are the number of days desired for a vacation, the starting location of the trip and also the destination locations that user wants to visit. In this initial version of the app, it is assumed that the initial location and the final location of the trip are the same, which is in accordance with the TSP concept. This location can be filled with the address where user will stay in the tourist destination city. The app will also provide a list of tourist destination locations (points of interest) that can be visited, so user only needs to choose from the existing list. In making this application, the database for the list of points of interest is taken from YogYes.com which is a website travel portal to Yogyakarta.

The system framework is presented in Figure 2. Our approach consists of two main modules, the first one is clustering module using k-means method. All destination locations preferred by user will be grouped into k cluster with the number of k equals with the number of vacation days. Second one is finding TSP solution using all possible permutations.

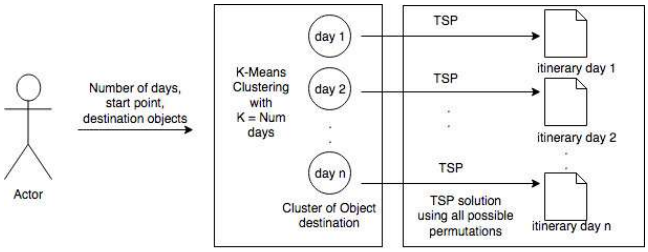

Figure 2. System framework.

Some processes that need to be done by the application are the process of accessing the location that the user wants as the initial location of the itinerary, the process of displaying the list of points of interest, the process of making itinerary, and the process of displaying itinerary result. In the end, user can download the itinerary result in the form of .pdf file in order to make it easier to print or save the itinerary result on other devices.

The next step is modeling the system. System modeling is made using flowchart diagram to describe the algorithm or the steps of problem solving. The general system flowchart can be seen in Figure 3.

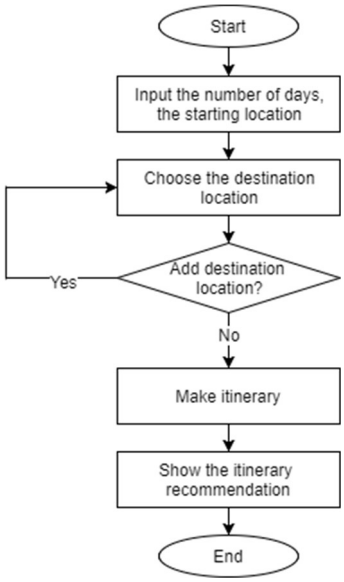

Figure 3. System flowchart.

Figure 3 illustrates the flow of the travel itinerary planning application. The process begins by entering the number of vacation days, the starting location of the trip and the destination locations to be visited. After system gets the inputs, the next step is to make the itinerary. The final step is to display the travel itinerary recommendation based on TSP solution approach and k-means clustering. The process of "make itinerary" is described in more detail in Figure 4.

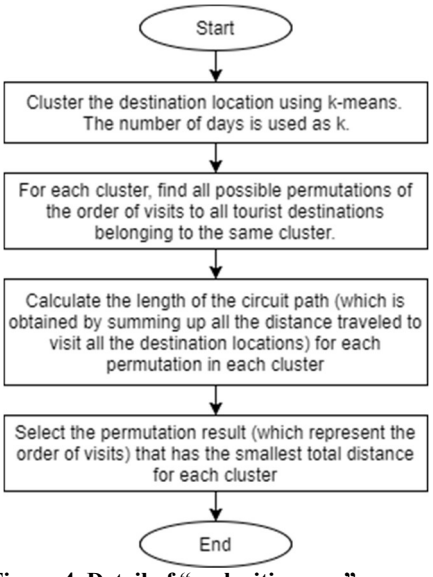

Figure 4. Detail of "make itinerary" process.

Figure 4 shows the detail process for getting a travel route. First of all, start by grouping the tourist destination locations by day. This grouping uses k-means clustering algorithm, where the number of days on vacation will be used as the value of k (number of clusters). With k-means clustering technique, the nearby tourist destinations will be grouped to be visited on the same day.

The next step is to apply the TSP approach to find the most efficient order of visits for each day. The process is done by searching all possible permutations of the order of visits to all tourist destinations belonging to the same cluster. Having

obtained the results of all possible permutations of the order of visits, then the next process is to take the distance between destinations based on data obtained from the Google Maps API. To accelerate this process, the distance between destinations is stored in the system database. Once obtained the distance, then calculate the length of the circuit path (which is obtained by summing up all the distance traveled to visit all the destination locations) for each possible permutation in each cluster. The last, select the permutation result (which represent the order of visits) that has the smallest total distance for each cluster. This permutation results are considered as the itinerary. From this result, the itinerary will be visualized into Google Maps.

4. RESULTS

The travel itinerary planning application is implemented as a web-based application. The application interface consists of two pages, the main page and the itinerary page. The main page is used to get input from user. On this page, user can determine the number of days of vacation, determine the starting location of the trip, and determine the desired tourist destinations. The number of the desired destination to visit can be more than one destination location. Figure 5 shows the example of implementation result from the main page.

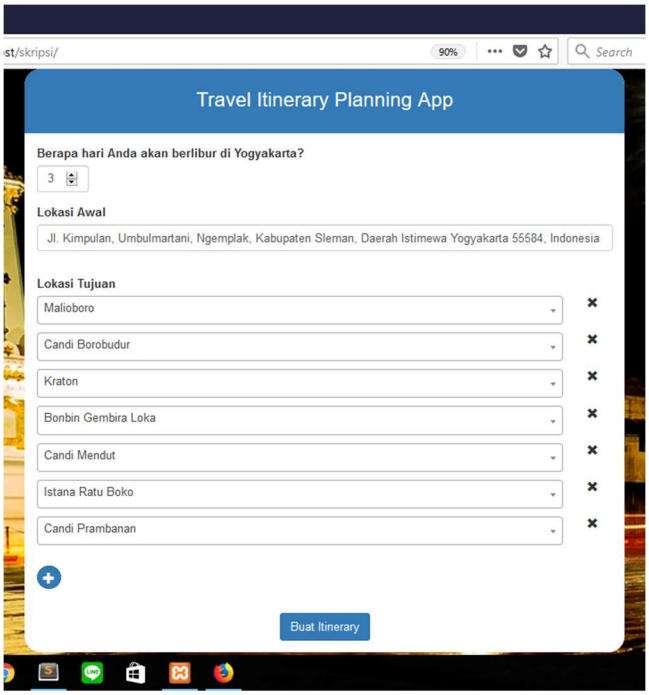

Figure 5. The main page.

In the example above, user inputs 3 days as the number of days on vacation, fills in the initial location of the trip and chooses 7 points of interest. After all the required data has been entered, the data will be processed using the algorithm as described in section 3 of this paper. User has to wait a while to get the result of the itinerary design from the system. An example of the itinerary result can be seen in Figure 6-8.

Figure 6 shows the itinerary recommendation for the first day, Figure 7 for the second day, and Figure 8 for the third day. If we analyze the result of the itinerary recommendation, we find that the places visited for each day are located close to each other. This result indicates that k-means clustering algorithm runs correctly. To ensure the clustering validity, we run Silhouette test on the results. Average Silhouette Coefficient, using 3 vacation days test case, results in 0.8374 and considered as strong classification according to [14].

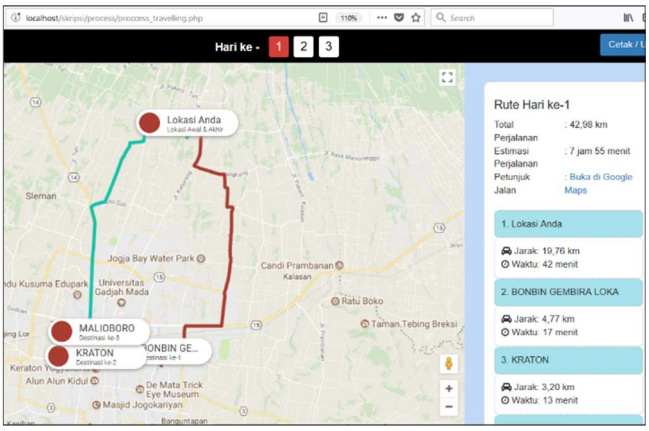

Figure 6. Itinerary for the 1st day.

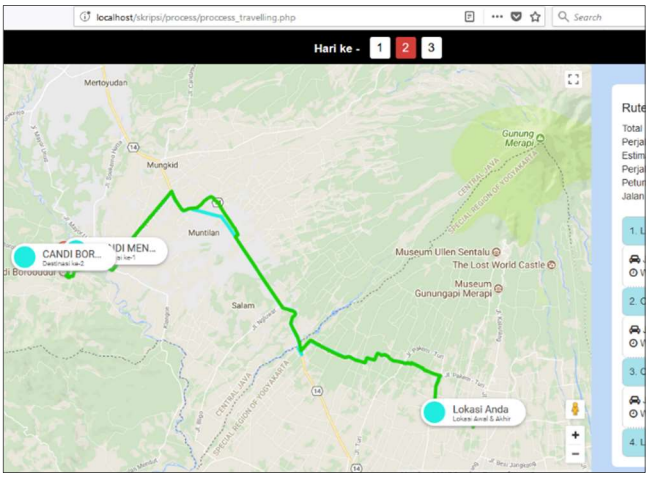

Figure 7. Itinerary for the 2nd day.

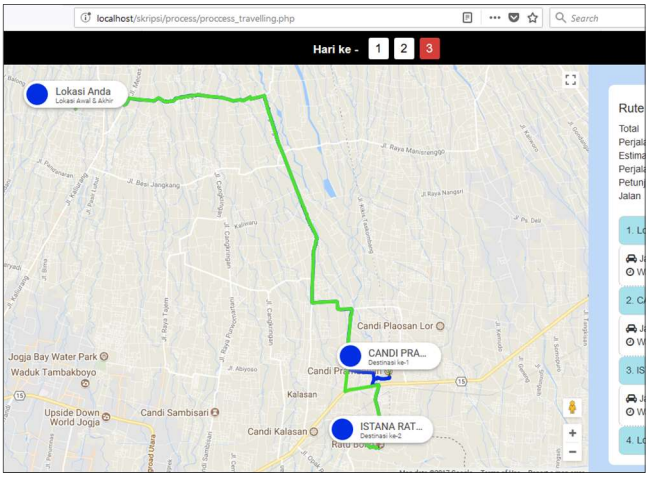

Figure 8. Itinerary for the 3rd day.

On the first day, user gets recommendation to visit 3 tourist attractions located in Yogyakarta's city center. On the second day it is recommended to visit 2 sights located in the north-west of Yogyakarta city, while on the third day it is recommended to visit 2 sights located in the east area of Yogyakarta.

Furthermore, the result of the order of visit has also indicated that the system's result is already the best route when viewed from the total distance traveled. On the itinerary page, in addition to the map presented, also presented the information about the estimated total distance and travel time on the right side of the map, as well as the detailed information about tour arrangement. User is also provided a link to access directions between locations that aim to facilitate the user in traveling between two locations. We obtain the location directions data from the Google Maps API. Finally, to facilitate the documentation, user can save and download the results of the itinerary design in the form of a .pdf file.

In this experiment, we tried to calculate and analyze the processing time of the application. For each scenario, we run it ten times then calculated the average value of the processing time. Although the scenario is same, the length of processing time may vary as it is influenced by several factors such as internet connection speed, memory, and also server processor speed. Unstable Internet connection will greatly affect the long of system's processing time. The results obtained can be seen in Table 1.

Table 1. The experimental results of the application processing time

No	Number of days	Number of destinations	Average of processing time (second)
1	3	6	2.0817
2	3	7	3.1725
3	3	8	3.5128
4	3	9	3.6899
5	3	10	4.6360
6	3	11	3.9075
7	3	12	6.4301
8	3	13	7.1961
9	3	14	7.7177
10	3	15	5.7449

Based on the experimental results, it can be seen that in general, the bigger the number of tourist destinations to visit, then the processing time of the system will be longer. This can happen because with the increasing number of tourist destinations, will prolong the clustering process. In addition, if the members of each cluster are larger, then the process of determining TSP solutions will also be longer. Nevertheless, in the real world, usually a traveler will not visit too many points of interest in a day. By considering that fact, the running time of this application can still be accepted by the user. For the future, the algorithm, especially the algorithm to find the solution of the order of visits needs to be improved to get faster run time.

5. CONCLUSION AND FUTURE WORK

In this paper, traveling salesman problem and k-means clustering approach has been applied to develop an automatic travel itinerary planning application. By using our application, it can help traveler in arranging a smart travel itinerary, which is efficient in terms of distance and travel time. However, we realize that this application still need many improvement, since the goals, decision criteria and other constrains of the travel itinerary planning problem are unable to completely predefined. In the future, in order to improve our algorithm, we plan to conduct User Acceptance Testing to gather feedback from prospective user.

6. ACKNOWLEDGMENTS

Our thanks to Department of Informatics Universitas Islam Indonesia for supporting this work.

7. REFERENCES

[1] Hsu, F. C. and Chen, P. 2000. Interactive genetic algorithms for a travel itinerary planning problem. *TSP*, 1, 13.

[2] Russell, S. and Norvig, P. 1995. *Artificial Intelligence A Modern Approach*. Prentice-Hall, Egnlewood Cliffs, 25, 27.

[3] Hoffman, K. L., Padberg, M., and Rinaldi, G. 2013. Traveling salesman problem. In *Encyclopedia of operations research and management science* (pp. 1573-1578). Springer US.

[4] Cook, W. 2007. *History of the TSP*. http://www.math.uwaterloo.ca/tsp/history/index.html.

[5] Moon, C., Kim, J., Choi, G., and Seo, Y. 2002. An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research*, 140(3), 606-617.

[6] Razali, N. M. and Geraghty, J. 2011. Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the world congress on engineering* (Vol. 2, pp. 1134-1139).

[7] Dorigo, M. and Gambardella, L. M. 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1), 53-66.

[8] Chen, S. M. and Chien, C. Y. 2011. Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, 38(12), 14439-14450.

[9] Murty, M. N. and Devi, V. S. 2011. *Pattern recognition: An algorithmic approach*. Springer Science & Business Media.

[10] Abbaspour, R. A. and Samadzadegan, F. 2011. Time-dependent personal tour planning and scheduling in metropolises. In *Expert Systems with Applications*, 38 (2011) 12439–12452.

[11] Rizzo, C. 2017. *This App Will Help You Effortlessly Plan a Multi-city European Vacation*. http://www.travelandleisure.com/travel-tips/mobile-apps/eightydays-app-plans-european-vacation.

[12] Hagen, K., Kramer, R., Hermkes, M., Schumann, B., and Mueller, P. 2005. Semantic matching and heuristic search for a dynamic tour guide. *Information and Communication Technologies in Tourism 2005*, pp.149-159.

[13] *Statistik Kepariwisataan 2016*. https://visitingjogja.com/10193/statistik-pariwisata-2016/.

[14] Kauffman, L. and Rousseeuw, P. J. 1990. *Finding Group in Data: An Introduction to Cluster Analysis*. Wiley, New York.