Developing an Android-Based City Tour App using Evolutionary Algorithm

https://doi.org/10.3991/ijim.v15i14.20275

Abidatul Izzah (✉), Irmala A. Kusuma, Yudi Irawan,
Toga A. Cinderatama, Benni A. Nugroho
PSDKU Politeknik Negeri Malang, Kediri, Indonesia
abidatul.izzah@polinema.ac.id

Abstract—Traveling around a city and making transit in certain areas is called a city tour. Furthermore, determining the optimal city tour route can be considered as a traveling salesman problem. There are many kinds of algorithms to solve this, one of which is the Genetic Algorithm (GA). In developing the City Tour application, a platform is needed to be taken to various places anywhere and anytime. Finally, we developed an application that runs on mobile devices. This application is built on the Android platform so that its use can be more efficient. Furthermore, it can be concluded that the GA applied to the Android-based City Tour Application is reliable to determine city tour routes; this is evidenced by comparing GA with the brute force method, where GA provides optimum results with less running time.

Keywords—Brute force, City tour, Genetic algorithm, Mobile application

1 Introduction

Travelling is an activity that many people do every day. It is taken to meet the needs of tourism and work. It also needs to be considered when we are in a foreign city or when we have several destinations in one trip. Furthermore, traveling around a city and making transit in certain areas is called a city tour. If we look at the computational discipline, determining the optimal city tour route can be considered as a traveling salesman problem. Traveling Salesman Problem is a category of NP-Hard Problem where the problem is difficult to solve so that there are many variations of methods that can be used [1][2][3]. The problem faced is how to build an optimal route by considering the rules on TSP, namely passing every location other than the initial location only once, to get the minimum total mileage to have an impact on saving transportation costs. Analytically, Traveling Salesman Problem (TSP) is a problem that is difficult to solve because many route combinations may occur along with the number of cities to be visited and must also pay attention to the applicable rules [4]. The number of studies that have been carried out has resulted in various completion methods that have been used. One method often used to solve NP-Hard problems is an algorithm adapted from nature or an evolutionary algorithm. There are many kinds of algorithms, one of which is the Genetic Algorithm (GA). GA has also developed and implemented both conventional and hybridized [1]–[3], [5]–[8]. Other studies have also developed this method in mobile devices so that the application of GA becomes more real [9]–[11]. However, the device built is still a general location; no route determination application specifically addresses tourist visits.

Therefore, in this study, an application will be developed to determine City Tour tourist locations in Kediri, Indonesia by implementing a Genetic Algorithm. The City Tour problem will be presented using the TSP approach. There have been many studies implementing Genetic Algorithms for solving TSP. In developing the City Tour application, a platform is needed to be taken to various places anywhere and anytime. Therefore, this application is developed on the mobile platform to be more efficient and easier to use anywhere. As we know, developments on mobile platforms are currently very rampant and have been implemented in many cases such as route search optimization, languages, retail, and others. As an example in a study [12], a mobile application was developed for the tour guide during Umrah. Other developments in banking and finance have also been carried out in [13], which produced a mobile application prototype that allows users to search for trilingual terms, namely Malay-Arabic-English. A study [14] presents the facts of a review about the factors affecting the use of mobile applications in retail. And there are many more application developments on smartphones today.

Furthermore, Android OS rules the smartphone market in Indonesia. Statcounter in [15] shows that 90 percent of smartphone users in Indonesia are Android users. Therefore, this study decided to implement GA for solving the city tour problem on Android-based. However, it does not rule out developing in other mobile OS versions such as iOS, Windows, Symbian, or another.

Finally, the output of this study is an application that is expected to provide the most optimal route desired by the user. As previously explained, the apps will be developed to solve city tour problems in Kediri, Indonesia, so that the apps will store data on tourist destinations. Thus, this study contributes to implementing GA in a TSP model and developing city tour apps to provide recommendations for travellers in Kediri, Indonesia.

2 Literature Review

2.1 City tour optimization as traveling salesman problem

Visiting the desired locations when doing a city tour will cost fuel prices, mileage, and travel time. The city tour trip would be better if the optimum route is available. Generally speaking, Traveling Salesman Problem(TSP) modeled the problem of determining the route. Traveling Salesman Problem is a problem that connects several cities with C_{ij} as the distance between city i and city j, the goal is to make a closed route by visiting each city once with the minimum total distance of all possible routes [4]. Several studies of determining the route in visiting specific locations have been conducted. A study mention that TSP is a classic problem that arises in the shipping business. In TSP, optimizations are carried out to find the shortest travel route passes the number of destinations with a specific path. As the result, each destination is passed once and the trip ends by returning to the starting place. Hence, TSP is a problem to find the most optimal route that can be taken, provided that each city must be visited once. If c_{ij} is the cost between location i and location j, then the cost matrix C can be written as follows:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \cdots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \cdots & c_{2n} \\ c_{31} & c_{32} & c_{33} & \cdots & c_{3n} \\ \cdots & \cdots & \cdots & \ddots & \vdots \\ c_{n1} & c_{n2} & c_{n3} & \cdots & c_{nn} \end{bmatrix}$$

Let x_{ij} represents whether there is a trip from location i to location j in a route, then the value of x_{ij} can be written as follows:

$$x_{ij} = \begin{cases} 1, & \text{if there is a trip from i to j} \\ 0, & \text{otherwise} \end{cases}$$

If Z is the TSP objective function, and $i, j = 1, 2, 3, \dots, N$ then the objective function Z is formulated by minimizing:

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

2.2 Genetic algorithm for city tour problem

Genetic Algorithm (GA) is an algorithm that adopts a natural selection process known as the evolutionary process proposed by Charles Darwin[16]. In the process of evolution, individuals continuously change genes to adapt to their living environment. "Only strong individuals can survive." GA may not always achieve the best results, but it often solves problems reasonably well. The genetic algorithm represents a solution to a problem as a chromosome. This chromosome will then regenerate into the optimal solution. There are several essential aspects in GA, including the definition of fitness function, definition and implementation of genetic representation, definition and genetic operations implementation. The three aspects above strongly support the performance of GA. The search algorithm on GA is based on natural selection mechanisms and biological evolution. The genetic algorithm combines a series of structures with the exchange of random information into a search algorithm with some human aptitude changes. A new set of individual sequences are created in each generation based on matches in the previous generation.

In its development, GA is widely used to solve high-complexity searching and optimization problems that often occur in dynamic programming such as TSP, Shortest Path, Minimum Spanning Tree, or Knapsack Problem[1], [17]–[19].

The GA method stages include chromosome coding, population initialization, calculation of fitness values, selection, crossover, and mutation. The following is the pseudocode of GA [16]:

```
Generate initial population P (0);
T = 0;
While has not met the criteria to quit
  Evaluate P(t);
  I(t) = Selection(P(t))
  If random < Pc //Probability of Crossover
    A(t) = Crossover(I(t));
    If random < Pm //Probability of Mutation
      A(t) = Mutation (I(t));
    EndIf
  EndIF
  t = t + 1
EndWhile
Output : the best individual P(t)
```

The coding of a GA chromosome can be done using the Binary Encoding, Integer Encoding, or Real-number Encoding methods. Furthermore, because the solution to this problem is the sequence of locations to be visited, the coding technique chosen is Integer Encoding, where an integer number symbolizes each location. The shape of the chromosomes can be seen in Figure 1 below:

1	3	2	9	2	4	0	8

Fig. 1. Genetic algorithm chromosome

In the case study of determining the shortest route, the value of the objective function ($f(x)$) of the chromosomes is the sum of the distance traveled from one to the next location, as in equation (1). Then if $f(x) = z$, then the fitness function is obtained from the following equation:

$$fit = \frac{1}{f(x)} \tag{2}$$

3 Method

3.1 System requirement

At this section, the requirement analysis contains a description of the software requirements to be developed. This requirement analysis includes functional and non-functional system requirements. Functional system requirements aim to register the access of a user as an application user. From this step, the user needs to select the location to be visited, delete unnecessary locations, view genetic algorithm calculations results to find the shortest route, and view the shortest route display on Google Map.

Furthermore, non-functional system requirements include the hardware and software requirements in application development. Hardware requirements used for the manufacture and testing of this application are a personal computer with an intel core i3 processor, 4GB minimum RAM, 14 "resolution 1366 x 768 monitor, 1 TB hard drive as storage media, and 3GB Android RAM phone. , OS 9.0 (Pie).

3.2 System design

After we finished the requirement analysis, the next stage is modeling the Unified Modeling Language (UML) with software modeling language. UML is a modeling language with an object-oriented approach. UML has various diagrams to explain the system to be developed, one of which is a use case diagram. Use cases describe the needs of each system user visually in interactions between systems and actors.

In this use case it will be known what functions are in the system being created. In the City Tour application, the system has a function that is more dominant than the user. Users can enter the destination location and view the optimal distance and route results, while the system has the task of displaying maps, looking for distance information per point, determining the optimal route with the Genetic Algorithm, and displaying the results. As shown in Figure 2 below:

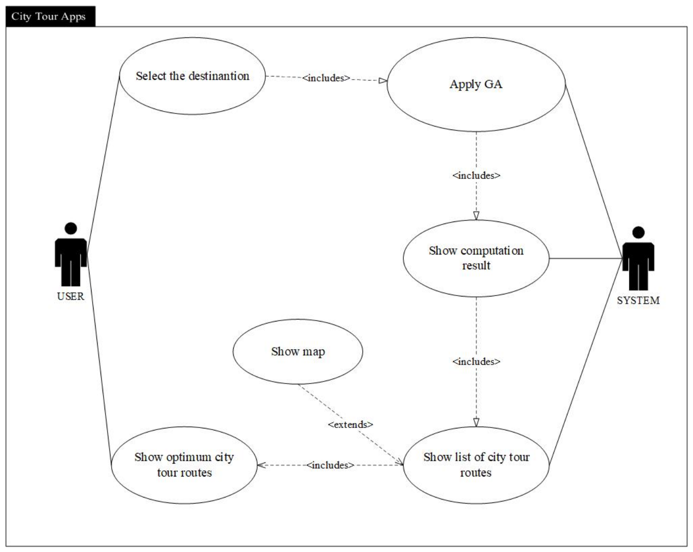


Fig. 2. Use case diagram

The next step is designing diagram architecture of the application. The diagram architecture is the structure of a system in the form of an image, which explains the concepts, principles, elements, and components, including how they work and instructions for use. There are three components in the City Tour Application, namely Cloud (Google Maps API), User, System. The Google Maps API acts as a source of information, the system acts as a calculation of the Genetic Algorithm, and the user acts as a brainware that inputs the location. As illustrated in Fig 3 below

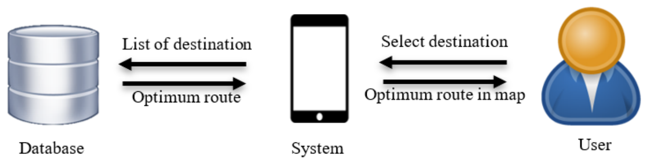


Fig. 3. Architecture Diagram

In the City Tour application, the tables are static, and some are dynamic. The following is a database design from City Tour Application:

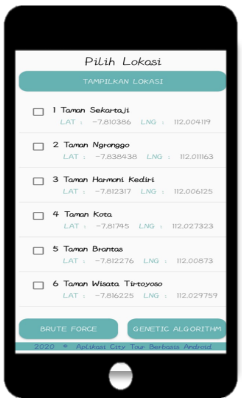
1. **Location Master Table**: Location master table functions for the location master data storage. The location master data will be displayed on the first page of the application, where users can select the desired location by checking the checkbox on the location they want to go to.
2. **Distance Master Table**: Distance master table works for the storage of distance master data between locations. Distance data is taken from retrieving information from Google Maps.
3. **Location Table**: The location table serves to store the location data selected from the master location table. Location tables are unlike master location and distance master tables, which are static. The location table is dynamic, which means it is always changing. The location table will be filled when the user checks the location on the first page, and all data will be erased when the user opens the application.
4. **Table Location2**: Table location2 serves to store locations. The location table is a complement to the location table by adding a column number. Column number will be used to connect table location2 with other tables. 5. Genetic Algorithm Table: The Genetic Algorithm table serves to store data on the Genetic Algorithm's evaluation results.
5. **Result Table**: The result table is used to store the optimal route results.
6. **Route Table**: The mix route table functions to store data on the longitude and latitude of the selected optimal route locations. Longitude and longitude data will be used to draw the route on maps.

4 Result and Discussion

4.1 Result

In this research work, we developed an application that runs on mobile devices. An Android-based City Tour application has one user. It has four layouts: Splash screen, List of destination, Computational Result, and Route Map. As shown in Fig. 4(a), the interface will appear when the user opens the application, and this page is called the splash screen.

(a) Splash Screen



Pilih Lokasi

TAMPILKAN LOKASI

1 Taman Sekartaji
LAT : -7.810386 LNG : 112.004119

2 Taman Ngronggo
LAT : -7.838438 LNG : 112.011163

3 Taman Harmoni Kediri
LAT : -7.812317 LNG : 112.006125

4 Taman Kota
LAT : -7.81745 LNG : 112.027323

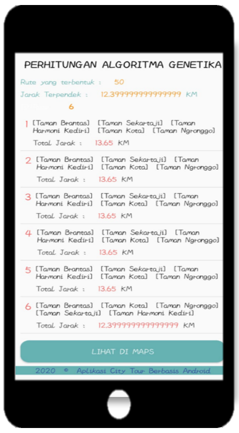
5 Taman Brantas
LAT : -7.812276 LNG : 112.00873

6 Taman Wisata Tirtoyoso
LAT : -7.816225 LNG : 112.029759

BRUTE FORCE | GENETIC ALGORITHM

2020 © Aplikasi City Tour Berbasis Android

(b) Choose the destination



PERHITUNGAN ALGORITMA GENETIKA

Rute yang terbentuk : 50
Jarak Terpendek : 12.399999999999999 KM
6

1 [Taman Brantas] [Taman Sekartaji] [Taman Harmoni Kediri] [Taman Kota] [Taman Ngronggo]
Total Jarak : 13.65 KM

2 [Taman Brantas] [Taman Sekartaji] [Taman Harmoni Kediri] [Taman Kota] [Taman Ngronggo]
Total Jarak : 13.65 KM

3 [Taman Brantas] [Taman Sekartaji] [Taman Harmoni Kediri] [Taman Kota] [Taman Ngronggo]
Total Jarak : 13.65 KM

4 [Taman Brantas] [Taman Sekartaji] [Taman Harmoni Kediri] [Taman Kota] [Taman Ngronggo]
Total Jarak : 13.65 KM

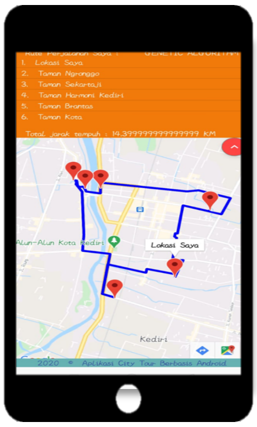
5 [Taman Brantas] [Taman Sekartaji] [Taman Harmoni Kediri] [Taman Kota] [Taman Ngronggo]
Total Jarak : 13.65 KM

6 [Taman Brantas] [Taman Kota] [Taman Ngronggo] [Taman Sekartaji] [Taman Harmoni Kediri]
Total Jarak : 12.399999999999999 KM

LIHAT DI MAPS

2020 © Aplikasi City Tour Berbasis Android

(c) Computational Result



Rute Perjalanan Saya : GENETIC ALGORITHM
1. Lokasi Saya
2. Taman Ngronggo
3. Taman Sekartaji
4. Taman Harmoni Kediri
5. Taman Brantas
6. Taman Kota

Total jarak tempuh : 14.399999999999999 KM

2020 © Aplikasi City Tour Berbasis Android

(d) Route Map

Fig. 4. City Tour Apps

After running for 3 seconds, the application will move to the select location page, as shown in Fig. 4 (b). On the choose location page, the user will select several locations to visit. After the user chooses several places, the next step is to press the Genetic Algorithm button to perform calculations using the genetic algorithm method to find the shortest route from several formed routes. In this application, we also code a brute force algorithm to evaluate and compare to GA's solution. Brute force is an algorithm that works by finding the best solution by generate and test. Thus, we provide two button options to find a solution for the city tour, using GA or brute force. These options also can be seen in the Fig(b), the brute force button is on the lower left and GA is next to it.

When the shortest route is formed, then the evaluation results of the genetic algorithm calculation will appear, which can be seen in Figure 4 (c). Finally, the application displays the shortest route shown on the Google Map, as in Figure 4 (d). This route will provide recommendations for the order of locations that must be visited first to start doing a city tour. The starting and ending points of the route are the points where the user is located.

GA did well in determining the shortest route of the city tour. GA's solution will converge and try to find the optimum solution at the end of the generation. The following is an example of GA's computation with PopSize = 10 and MaxGen = 50, which shows the solution has been found in the 6th generation.

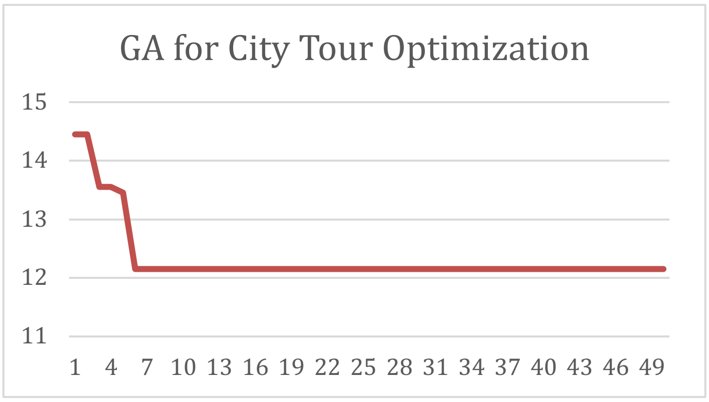


Fig. 5. Optimizing Process in GA

Moreover, three tests have been conducted: the black box testing method, the computational testing method, and the effectiveness solution. Black box testing is used to determine whether the application features are well developed or not by trying all the available features, as shown in the table below:

Table 1. Blackbox Testing Result

No	Scenario	Expected Result	Valid/ Not Valid
1	Open the application	The application shows the splash screen	Valid
2	Show locations list	List of locations successfully displayed	Valid
3	Choose the location	The location can be selected and stored in the database.	Valid
4	Delete location	The selected location can be removed from the view and database.	Valid
5	Genetic algorithm calculation	The system performs genetic algorithm calculations.	Valid
6	Displays the results of the calculation of the genetic algorithm	Successfully displays the results of the genetic algorithm calculations and displays the route with the shortest distance	Valid
7	View the selected route on the Map	Displays the shortest route into the map marked with a marker and a polyline	Valid

4.2 Discussion

Furthermore, computational testing is carried out to see the reliability of Genetic Algorithm, by comparing GA with the brute force method, which of these two methods has a higher level of effectiveness in terms of time and solution. The test is carried out by increasing the complexity of the search space. Apart from comparing time, the application is also evaluated for the algorithm capabilities. Brute force was chosen because it is considered a method that will solve the problem and can find the best solution even by trying one by one and calculating it (generate and test). Or in other words, the solution generated by the brute force is always optimum. Thus, if GA shows the same result with brute force's, the GA's solution is the optimum solution. The following is a comparison of the results of GA and brute force computational testing in complexity and capability:

Table 2. Experiment Result

Number of city	**Brute Force**		**Genetic Algorithm**	
	Computation time rate	***Solution***	***Computation time rate***	***Solution***
3	0,9 s	11,3 km	0,5 s	11,3 km
4	1,9 s	11,5 km	0,6 s	11,5 km
5	2,8 s	12,4 km	0,7 s	12,3 km
6	4,5 s	12,9 km	0,8 s	12,9 km
7	26 s	15.7 km	0,8 s	15.7 km

From the experiment result, the application can optimize regardless of the number of visited locations. However, with a value of n > 7, applications using the brute force method require a very high running time of more than one hour. Therefore, Table 2 only shows the time ratio up to n = 7. The difference in time required by the brute force and GA in determining city tour routes is obvious from the results shown above. The brute force method with a small dimension gives optimum value and shorter time. However, brute force becomes ineffective if the dimensions increase. While GA works well even though the dimensions have increased, but the running time does not increase significantly. Afterward, we also compare the effective solution that given by two algorithms. The application tested by solving the same tour problem using Brute Force and GA. The result shows that the solution given by both of them are similar. If any difference, it is less significant. So, we can conclude that an application using GA can provide a good one.

5 Conclusion

In this research work, we developed an application that runs on mobile devices. An Android-based City Tour application has one user. It has four layouts: Splash screen, List of destination, Computational Result, and Route Map. It developed by using java programming with Android Studio as editor. Furthermore, based on system testing, it can be concluded that the GA applied to the Android-based City Tour Application is reliable to determine city tour routes; this is evidenced by comparing GA with the brute force method, where GA provides optimum results with less running time. The application also has store tourist destinations data in Kediri city, Indonesia. Finally, this study successfully contributes by developing an android city tour app and implementing GA in a mobile device that can provide recommendations for someone who will travel in Kediri, Indonesia.

Further, this research is still limited. There is so much that can be extracted from this conclusion. The city tour application can be developed in areas that are not discussed here. It has a chance to develop in other mobile OS versions such iOS, windows, Symbian, or another. It can also use other evolutionary algorithms such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), or another.

6 Acknowledgement

The travel expense and registration of this paper to be published in I-JIM is fully sponsored by The Indonesian Ministry of Research and Higher Education through the Penelitian Kerjasama antar Perguruan Tinggi (PKPT) grant 2020.

7 References

[1] J. Grefenstette, R. Gopal, B. Roamaita, and D. Van Gucht, "Genetic algorithms for the traveling salesman problem," Evol. Comput. Foss. Rec., no. June 2014, pp. 532–540, 1998.

[2] C. M. White and G. G. Yen, "A hybrid evolutionary algorithm for traveling salesman problem," in Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), 2004, vol. 2, pp. 1473–1478. https://doi.org/10.1109/cec.2004.1331070

[3] M. Albayrak and N. Allahverdi, "Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms," Expert Syst. Appl., vol. 38, no. 3, pp. 1313–1320, 2011. https://doi.org/10.1016/j.eswa.2010.07.006

[4] G. Reinelt, "TSPLIB—A traveling salesman problem library," ORSA J. Comput., vol. 3, no. 4, pp. 376–384, 1991. https://doi.org/10.1287/ijoc.3.4.376

[5] Z. H. Ahmed, "Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator," Int. J. Biometrics Bioinforma., vol. 3, no. 6, pp. 96–105, 2010.

[6] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," Eur. J. Oper. Res., vol. 174, no. 1, pp. 38–53, 2006. https://doi.org/10.1016/j.ejor.2004.09.057

[7] G. Sun, C. Li, J. Zhu, Y. Li, and W. Liu, "An efficient genetic algorithm for the traveling salesman problem," Commun. Comput. Inf. Sci., vol. 107 CCIS, pp. 108–116, 2010.

[8] G. A. Jayalakshmi, S. Sathiamoorthy, and R. Rajaram, "a Hybrid Genetic Algorithm — a New Approach to Solve Traveling Salesman Problem," Int. J. Comput. Eng. Sci., vol. 02, no. 02, pp. 339–355, 2001. https://doi.org/10.1142/s1465876301000350.

[9] Ilhan \.Ilhan, "An Application On mobile devices with android and IOS operating systems using google maps APIs for the traveling salesman problem," Appl. Artif. Intell., vol. 31, no. 4, pp. 332–345, 2017. https://doi.org/10.1080/08839514.2017.1339983

[10] L. Helshani, "An Android Application for Google Map Navigation System, Solving the Travelling Salesman Problem, Optimization throught Genetic Algorithm The mathematical formulation throught graphs theory," pp. 89–102, 2015.

[11] T. Narwadi and Subiyanto, "An application of traveling salesman problem using the improved genetic algorithm on android google maps," AIP Conf. Proc., vol. 1818, no. March 2017, 2017. https://doi.org/10.1063/1.4976899
[12] M. S. Sahrir, M. F. Yahaya, T. Ismail, M. A. Zubir, W. Rusli, and W. Ahmad, "Development and Evaluation of i-Mutawwif: A Mobile Language Traveller Guide in Arabic for Mutawwif (Umrah Tour Guide)," pp. 54–68. https://doi.org/10.3991/ijim.v12i2.7708
[13] Z. Omar, "Prototype Development of Mobile App for Trilingual Islamic Banking and Finance Glossary of Terms via iOS and Android Based Devices Learning via Mobile Technology," vol. 11, no. 3, pp. 145–161. https://doi.org/10.3991/ijim.v11i3.6620
[14] A. Wohllebe and P. Dirrler, "Mobile Apps in Retail: Determinants of Consumer Acceptance – A Systematic Review," vol. 14, no. 20, pp. 153–164. https://doi.org/10.3991/ijim.v14i20.18273
[15] GlobalStats, "Mobile Operating System Market Share Indonesia | StatCounter Global Stats," Www.Gs.Statcounter.Com. 2020.
[16] D. Whitley, Computer Science a Genetic Algorithm Tutorial. 1993.
[17] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," IEEE Trans. Evol. Comput., vol. 6, no. 6, pp. 566–579, 2002. https://doi.org/10.1109/tevc.2002.804323
[18] S. Khuri, T. Bäck, and J. Heitkötter, "The zero/one multiple knapsack problem and genetic algorithms," in Proceedings of the 1994 ACM symposium on Applied computing, 1994, pp. 188–193. https://doi.org/10.1145/326619.326694
[19] G. Zhou and M. Gen, "Genetic algorithm approach on multi-criteria minimum spanning tree problem," Eur. J. Oper. Res., vol. 114, no. 1, pp. 141–152, 1999 https://doi.org/10.1016/s0377-2217(98)00016-2

8 Authors

Abidatul Izzah is a lecturer at State Polytechnic of Malang, PSDKU Kediri Indonesia. She received the master's degree from Sepuluh Nopember Institute of Technology, Indonesia. Her research mainly focused on Artificial Intelligence and Software Engineering. Email: abidatul.izzah@polinema.ac.id

Irmala Arin Kusuma Wardani is a student at State Polytechnic of Malang, PSDKU Kediri Indonesia. She is programmer and graphic designer.

Yudi Irawan is a student at State Polytechnic of Malang, PSDKU Kediri Indonesia.

Toga Aldila Cinderatama is a lecturer at State Polytechnic of Malang, PSDKU Kediri Indonesia. He received the master's degree from Chang Gung University, Taiwan. His research mainly focused on Networking, Information Systems, and Software Engineering.

Benni Agung Nugroho is a lecturer at State Polytechnic of Malang, PSDKU Kediri Indonesia. He received the master's degree from Gajah Mada University, Indonesia. His research mainly focused on Mobile Programming and Networking.

Article submitted 2020-12-04. Resubmitted 2021-05-24. Final acceptance 2021-05-25. Final version published as submitted by the authors.