International Journal of
Intelligent Systems and
Applications in Engineering
ISSN:2147-6799 www.atscience.org/IJISAE
Advanced Technology and Science
Original Research Paper

Solution for the Travelling Salesman Problem with a Microcontroller-based Instantaneous System

İlhan İlhan[*1]

Accepted 15th December 2016

Abstract: The travelling salesman problem (TSP) is one of the most frequently researched combinational optimization problems. Despite its trivial definition, the problem is very difficult to solve. Therefore, it is categorized as an NP-hard problem in research literature. It is used for the solution of many real-life problems like route planning, transportation and logistics applications. In this study, a microcontroller-based system was proposed for the solution of the TSP. In the proposed system, location information was imported instantaneously via a GPS module. The Ant Colony Optimization (ACO) algorithm was coded inside the microcontroller for the solution of the TSP. Various tests were performed on two different datasets using different parameter values. Tests showed that the only difference between the results for the microcontroller-based and the computer-based systems were the runtimes. Therefore, it was concluded that population-based algorithms like ACO could easily be used in current microcontrollers for various purposes in different areas.

Keywords: *Ant Colony Optimization, GPS Module, Microcontroller, Travelling Salesman Problem.*

1. Introduction

The travelling salesman problem (TSP) was introduced by Euler in 1759, and it was Menger who formulated it mathematically for the first time in the 1930s [1]. Currently, it is one of the most frequently researched combinational optimization problems. It is considered to be an NP-hard problem, due to its difficulty [2]. The TSP is applied to many real-life problems like route planning, transportation and logistics. Therefore, it gives an ideal testing environment for researchers, especially those working on optimization techniques [3].

Exact solution algorithms proposed for the TSP, like dynamic programming, branch-and-bound and branch-and-cut, can give successful results only up to a certain dimension [4][5][6][7]. In these algorithms, computation time increases with both the increasing number of points and dimensions of the problem, and therefore it becomes impossible to reach optimum results. On the other hand, heuristic and metaheuristic methods can give satisfactory results due to their shorter computation times, although they cannot guarantee an optimum result. Therefore, they are highly preferred by researchers in the field.

There are many studies using heuristic and metaheuristic algorithms. In the first set of these studies, computational experiments were carried out using benchmark data [8][9][10][11]. In the second set of studies, mobile devices with Acorn RISC Machine (ARM) architecture were used [12][13][14]. These devices have the ability to perform complicated computational tasks due to their ARM Cortex-A series application processors. The first application developed for mobile devices was the MyDisasterDroid, running on devices with the Android operating system [12]. This system was developed as a disaster management system for the Philippines. It was used to determine the optimum routes to transport rescue and aid services to the affected areas in the most efficient way in case of a disaster. To determine the optimum route, a genetic algorithm was used. The distances between geographical positions were computed with Euclidean distances. In a second application, a travel planner for the city of Eskişehir was developed [13]. Thirty points and 150 sub-points in the city were determined as places to see, and their location information was recorded to the application. The application was tested for two scenarios with a varying number of places to see: 5 and 10. As a result of the tests using the A* and Ant Colony Optimization (ACO) algorithms, it is seen that both algorithms identified the same route as the optimum one. In a later study, a test tool was developed that can run on both Android and IOS operating systems, to test eight different optimization algorithms (genetic algorithms, artificial immune systems, differential evolution algorithms, particle swarm optimization algorithms, simulated annealing algorithms, tabu search algorithms, artificial bee colony algorithms and ant colony optimization algorithms) [14]. On this test tool, Six-Hump Camel Back, Rastrigin, Shubert, Schwefel and Drop Wave functions were used as benchmark functions. It also allowed the comparison of algorithms for user-defined special functions. Comparisons based on different parameters can be represented graphically or as a table.

In this study, the proposed solution for the TSP employs a microcontroller-based route planning system that possesses an ARM Cortex-M4 processor. The proposed system imports location information in real time using a GPS module. For the solution of the TSP, the ACO algorithm was used and the related codes were stored in the microcontroller. Tests were run on two different datasets with different parameter values. Results obtained for the microcontroller-based and computer-based systems were compared in terms of optimum routes and running times. Despite the differences in running times, both systems produced almost the same optimum routes. This is an important result since it indicates

[1] *Necmettin Erbakan University, Faculty of Engineering and Architecture, Department of Mechatronic Engineering, Konya, Turkey*
** Corresponding Author: ilhan@konya.edu.tr*

that population-based algorithms like the ACO algorithm can easily be used with contemporary microcontrollers in different areas and for different purposes.

2. The Travelling Salesman Problem

The aim of the TSP is as follows: a salesman or a vehicle should start from a certain location, and during its travel, it should visit every other location in the system only once, returning to the starting point at the end. The total tour distance for the whole journey should be the shortest possible. If the total number of points is n then there are n-1 alternatives to visit from the first point and n-2 alternatives to visit from the second point. Here n is the dimension of the problem and, consequently, the total number of routes possible is (n-1)!. So, the number of alternative solutions increases rapidly with the increasing number of points and a great amount of time is needed to find the optimum solution. The mathematical formulation of the TSP is as follows [15].

$$y = \sum_{i=1}^{n} \sum_{j=1, i\neq j}^{n} x(i,j)d(i,j) \quad (1)$$

$$\sum_{j=1, j\neq i}^{n} x(i,j) = 1, \quad i = 1,2,..,n \quad (2)$$

$$\sum_{i=1, i\neq j}^{n} x(i,j) = 1, \quad j = 1,2,..,n \quad (3)$$

$$\sum_{i,j\in S, i\neq j}^{n} x(i,j) \leq |S| - 1, \quad \forall S \subset \{1,2,..,n\} \quad (4)$$

$$x(i,j) = \begin{cases} 1, & if\ the\ path\ goes\ from\ point\ i\ to\ point\ j \\ 0, & Otherwise \end{cases} \quad (5)$$

(Equation.1) is the aim function of the TSP. Here d(i, j) is the distance between points i and j and x(i, j) is the indicator of whether the journey from i to j was made. To ensure that every point was visited only once, (Equation.2) and (Equation.3) are used. To prevent any sub tours, a sub tour constraint is given by (Equation.4). If x(i, j) in (Equation.5) is equal to 1 that means that journey was undertaken from i to j, and if it is equal to 0 then that journey was not undertaken [16].

The points in this study are geographical locations on Earth and they are expressed in terms of their latitudes and longitudes. The distances between the points are given as air distances, therefore, they are calculated using the Haversine formula given in (Equation.6) [17].

$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat1).\cos(lat2).\sin^2\left(\frac{\Delta long}{2}\right)$$
$$c = 2.\arctan(\sqrt{a}, \sqrt{1-a}) \quad (6)$$
$$d = R.c$$

Here, the first point was represented by (lat1, long1) and the second point by (lat2, long2). According to these expressions, the difference in latitude and longitudes can be calculated as Δlat = lat2 - lat1 and Δlong = long2 - long1. R is the radius of the Earth and it is estimated as 6371 km for this study.

3. The Ant Colony Optimization Algorithm

In this study, the ACO algorithm [18] was used to determine the optimum route between the locations. This algorithm was coded into the microcontroller - with an ARM Cortex-M4 processor - and the optimum route was determined for given parameters. The ACO algorithm starts by determining the initial pheromone levels for roads between the points. Then the ant population is created based on the population size entered into the algorithm by the user. In this process, every ant is initially placed at a random location and then (Equation.7) and (Equation.8) are used to determine to which location the ant, k, should go to from that initial location, i among the alternatives with a total number of u.

$$j = \begin{cases} max_{u\in J_k(i)}[\tau(i,u)]^{\alpha} * [\eta(i,u)]^{\beta} & if\ q \leq q_0 \\ use\ Equation\ (8) & Otherwise \end{cases} \quad (7)$$

$$P_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha} * [\eta(i,j)]^{\beta}}{\sum_{u\in J_k(i)}[\tau(i,u)]^{\alpha} * [\eta(i,u)]^{\beta}} & if\ j \in J_k(i) \\ 0 & Otherwise \end{cases} \quad (8)$$

Here τ(i, u) is the pheromone trail on the (i, u) line. The term η(i, u) = 1/δ(i, u) is the inverse of the distance from point i to point u. The term $J_k(i)$ represents the points not visited by the ant, k, at the point i. The parameter α (α > 0) determines the importance of the pheromone level of the relevant road, and the parameter β (β > 0) determines the effect of the road length on the choice of the next destination. The parameter q_0 ($0 \leq q_0 \leq 1$) gives the relative importance within the solution space. Local pheromone levels are updated according to (Equation.9) and (Equation.10) after every ant has completed its tour.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t+1) \quad (9)$$

$$\Delta\tau_{ij}^{k}(t+1) = \begin{cases} 1/L^{k}(t+1) & if\ ant\ k\ used\ the\ path\ ij \\ 0 & Otherwise \end{cases} \quad (10)$$

Here, $\tau_{ij}(t)$ is the initial pheromone level and ρ ($0 \leq \rho \leq 1$) is the pheromone evaporation value entered by the user. The term $L^{k}(t+1)$ represents the total tour length of ant k. Global pheromone levels are updated according to (Equation.11) and (Equation.12) after every ant has completed its tour.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{k}(t+1) \quad (11)$$

$$\Delta\tau_{ij}^{k}(t+1) = \begin{cases} 1/L_{best}(t+1) & if\ ij\ belongs\ to\ the\ best\ tour \\ 0 & Otherwise \end{cases} \quad (12)$$

The best tour length obtained globally is $L_{best}(t+1)$. These processes in the ACO algorithm are repeated for a number of iterations determined by the user.

4. The Proposed System

A flowchart of the microcontroller-based real-time system proposed for the solution of the TSP is given in (Figure.1). As seen in this figure, the flowchart consists of two parts: the parts inside and outside the dashed line. The part outside the dashed line shows the data input-output operations of the computer. The part inside

the dashed line, in contrast, shows the flowchart of the algorithm run by the microcontroller. The interface used on the computer was developed with the Microsoft Visual C# development environment. The algorithm on the microcontroller was coded with the mikroC PRO for ARM programming tool.

The parameters needed for the ACO, and the longitude and latitude values of the points to be visited, were entered into the computer via the developed interface. (Figure.2a) shows the data input interface. The instantaneous geographical position information in that interface is imported from the GPS sensor connected to the microcontroller. The location data of the places to be visited can be determined either by clicking on the map or by entering latitude and longitude values with the keyboard. That information is sent to the microcontroller via USB communication. Then the relevant code in the microcontroller is run and the shortest distance and its related route are transferred to the computer again via USB. The resulting optimum route is shown on the map through the results interface. The result interface can be seen in (Figure.2b).

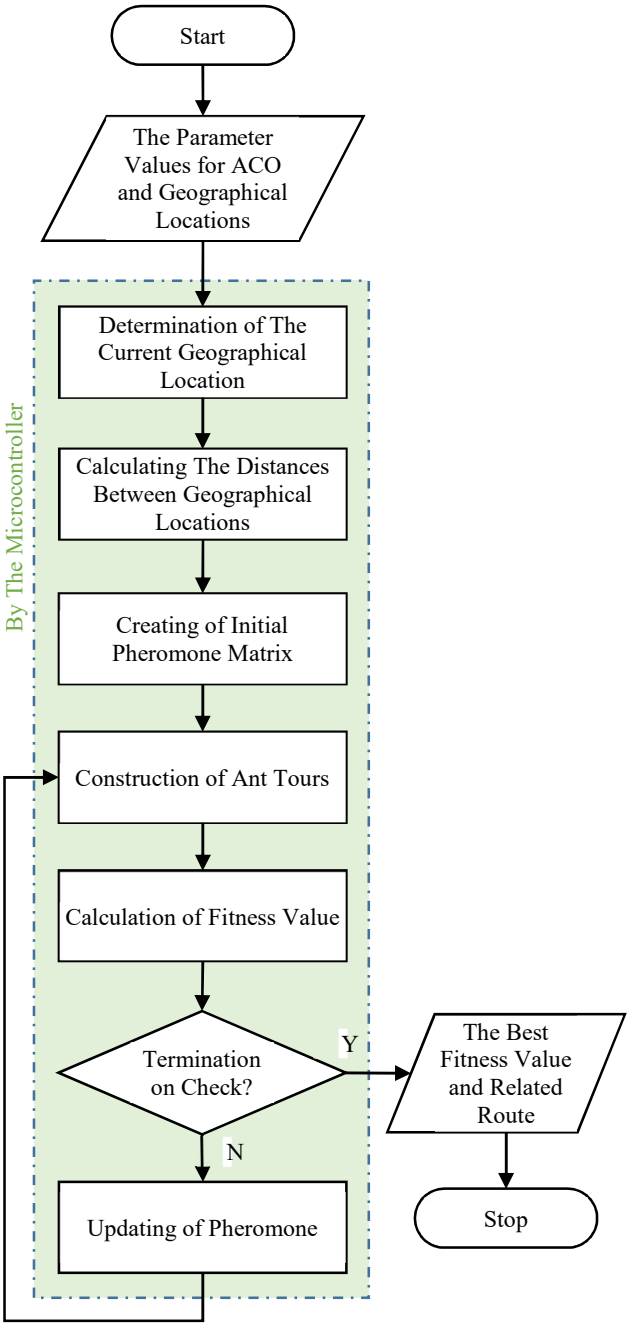
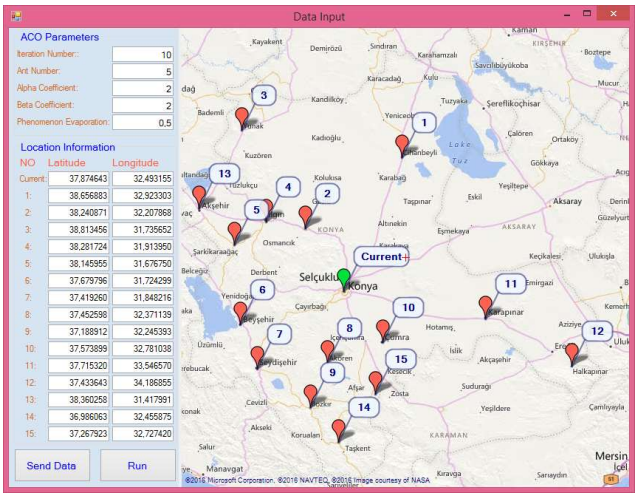


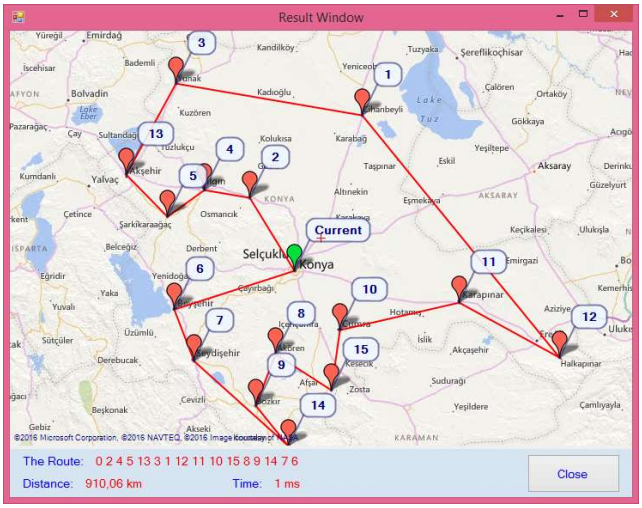
Figure 1. Flowchart of the proposed system

In this study, a 32-bit STM32F407VGT6 [19] with ARM Cortex-M4 architecture (from STMicroElectronics company) was used as the microcontroller. This microcontroller has an operating frequency up to 168 MHz. It contains 1 MByte of flash (programmable) memory and 192 KBytes of RAM memory. Its operating voltage is from 1.8 to 3.6 V. Additionally, it has 82 I/O ports with an external interrupt option. In order to program the STM32F407VGT6 microcontroller, and to run the code it contains, some mandatory connections (feed, crystal, programming, etc.) have to be made. Instead of making those connections separately, the STM32F4 Discovery Kit (from STMicroElectronics company) [20] was used, which contains the STM32F407VGT6 microcontroller and the relevant connections. That developer kit can be seen in (Figure.3a). In (Figure.3b), the GPS sensor (serial number GY-NEO6MV2 [21]) is shown, which detects the latitude and longitude values of the instantaneous geographical location. This sensor communicates with the microcontroller via a serial port. The microcontroller detects the meaningful latitude and longitude data from the set of numbers collected by the sensor.

As seen in the flowchart in (Figure.1), first the instantaneous location data is identified by the microcontroller. Then, the distances between the geographical points sent by the computer, including the actual location, are calculated using the Haversine formula given by (6) to create a distance matrix. Initial values are

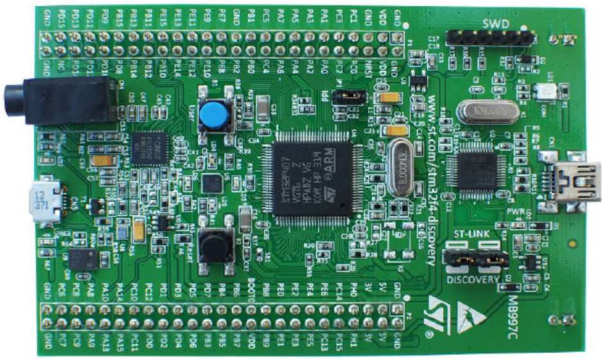


(a)



(b)

Figure 2. (a) Data Input interface (b) Result interface



(a)

(b)

Figure 3. (a) STM32F4 Discovery Kit (b) GY-NEO6MV2 GPS Module

assigned to the pheromone matrix and the ACO algorithm is run iteratively depending on the parameters sent by the computer, until the terminating criterion is met. The shortest distance and the corresponding route are sent by the microcontroller to the computer.

5. Results

The microcontroller-based, real-time system developed for the solution of the TSP was compared with a computer-based system. The computer-based system was developed using Microsoft Visual C# programming language. In that system, the flowchart shown in (Figure.1) was processed by the computer while the microcontroller was disabled. The computer used for this processing had 8 GB RAM, an i7 processor with an operating frequency of 2.40 GHz and Microsoft Windows 8.1 Home operating system.

The performance of the proposed microcontroller-based in the study, real-time system was tested on two datasets of different sizes. The first dataset was formed from 10 different points selected in Turkey. The second dataset consisted of 15 different points across Konya province. (Table.1) shows the instantaneous location and the latitude and longitude data of the points from both datasets.

The microcontroller-based system was tested on both datasets using the ACO algorithm and a comparison was made with the results obtained from the computer-based system. Different iteration numbers and population sizes were selected for the testing process. For the ACO algorithm, alpha and beta coefficients were assigned as 2 and the pheromone evaporation value was assigned as 0.5. ACO algorithm was run 30 times with the microcontroller and computer-based systems for both datasets. The results detailed in (Table.2) were determined by calculating average values. As seen in this table, both systems gave almost identical results in terms of the total distance to be travelled. For example, with an iteration number and population size of 20 and 10 respectively for the first dataset, both the microcontroller and computer-based systems gave 2569 km as the result. Similarly, for the second dataset, which has a greater number of points, both systems

Table 1. Latitude and longitude values from datasets

The First Dataset			***The Second Dataset***		
No	***Latitude***	***Longitude***	***No***	***Latitude***	***Longitude***
Current	37.8746429	32.4931554	Current	37.874643	32.493155
1	39.9333635	32.8597419	1	38.656883	32.923303
2	36.8968908	30.7133233	2	38.240871	32.207868
3	36.9914194	35.3308285	3	38.813456	31.735652
4	38.7204890	35.4825970	4	38.281724	31.913950
5	37.0659530	37.3781100	5	38.145955	31.676750
6	41.2797031	36.3360667	6	37.679796	31.724299
7	38.6742286	29.4058825	7	37.419260	31.848216
8	37.7830159	29.0963328	8	37.452598	32.371139
9	38.4237340	27.1428260	9	37.188912	32.245393
10	40.1885281	29.0609636	10	37.573899	32.781038
			11	37.715320	33.546570
			12	37.433643	34.186855
			13	38.360258	31.417991
			14	36.986063	32.455875
			15	37.267923	32.727420

Table 2. Results obtained for datasets.

Iteration Number	*Population Number*	*The First Dataset*				*The Second Dataset*			
		Microcontroller		*Computer*		*Microcontroller*		*Computer*	
		Distances (km)	*Elapsed Times (ms)*	*Distances (km)*	*Elapsed Times (ms)*	*Distances (km)*	*Elapsed Times (ms)*	*Distances (km)*	*Elapsed Times (ms)*
10	5	2727	51	2725	1	927	110	934	2
10	10	2620	104	2624	2	916	227	916	4
20	10	2569	209	2569	4	900	449	900	8
50	20	2566	1065	2567	20	886	2293	884	41
50	50	2566	2814	2566	44	884	5857	884	84
100	50	2566	5624	2566	92	884	11739	884	196

produced 900 km as the result when run with the same number of iterations and the same population size. When runtimes were considered it was seen that the computer-based system was much faster than the microcontroller-based system. For example, when the iteration number and population size were 20 and 10 respectively - for the first dataset - the microcontroller-based system runtime was 209 ms, whereas the computer-controlled system runtime was 4 ms. Similarly, for the second dataset with the same number of iterations and the same population size, the microcontroller-based system runtime was 449 ms, whereas the computer-controlled system runtime was 8 ms. These results showed that both systems produce similar numbers, but the computer-controlled system runs much faster. This is expected, considering the operating frequency of the computer processor (2.40 GHz) and the operating frequency of the STM32F407VGT6 microcontroller (168 Mhz).

Currently, systems like autonomous robots and unmanned aerial vehicles are being widely used for different purposes at different locations. Determination of the routes for such systems is a highly researched topic. This problem can be considered as a TSP in its most basic form. This study used a microcontroller-based system, which is commonly implemented on those systems and the solution for the TSP was studied, which is the basic form of routing problem. For this purpose, the ACO algorithm was used, which is a population-based metaheuristic algorithm. As seen in the results from (Figure.2), this algorithm can operate with the desired parameters, and the same results (optimum distance) can be obtained almost with the computer-based system. Of course, the RAM memory capacity and running speed of the microcontroller are worth mentioning as limiting factors.

6. Conclusion

In this study, a microcontroller-based route planning system possessing an ARM Cortex-M4 processor was proposed for the solution of the TSP. The proposed system imported location information via a GPS module in real-time. The ACO algorithm was used for the solution of the TSP and the related code was recorded in the microcontroller. Tests were carried out on two different datasets for different parameter values. The results obtained from the microcontroller- and computer-based systems were compared in terms of the optimum route and runtimes. Despite the differences in runtimes, both systems gave the same optimum routes. Thus, it has been shown that even population-based algorithms like ACO could easily be used with current microcontrollers for the determination of routes for systems like autonomous robots or unmanned aerial vehicles.

Acknowledgment

This study is supported by Necmettin Erbakan University Scientific Research Projects Coordinatorship, Konya, Turkey. The authors would like to thank the editors and anonymous reviewers of this manuscript for their very helpful suggestions.

References

[1] K. Menger (1932). Das botenproblem. In Ergebnisse eines Mathematischen Kolloquiums 2 (K. Menger, editor), Teubner, Leipzig.

[2] M. R. Garey and D. S. Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and co., New York.

[3] D. S. Johnson (1990). Local optimization and the traveling salesman problem. Automata, Languages and Programming, Springer Berlin Heidelberg, Pages. 446-461.

[4] M. Held and R. M. Karp (1992). A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics. Vol. 10. Pages. 196-210.

[5] N. Ascheuer, F. Matteo and G. Martin (2001). Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. Mathematical Programming, Vol. 90. Pages. 475-506.

[6] T. Volgenant and R. Jonker (1982) A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. European Journal of Operational Research, Vol. 9. Pages. 83-89.

[7] D. L. Applegate, R. E Bixby, V. Chvata and W. J Cook (2011). The traveling salesman problem: A computational study. Princeton University Press.

[8] Z. C. Hlaing and M. A. Khine (2011). Solving traveling salesman problem by using improved ant colony optimization algorithm. International Journal of Information and Education Technology, Vol. 1. Page. 404.

[9] B. Li, W. Lipo and W. Song (2008) Ant colony optimization for the traveling salesman problem based on ants with memory. Natural Computation, ICNC'08. Fourth International Conference on. Vol. 7.

[10] K. H. Hingrajiya, K. G. Ravindra and G. S. Chandel (2012). An ant colony optimization algorithm for solving travelling salesman problem. International Journal of Scientific and Research Publications, Vol. 2. Pages. 1-6.
[11] C. A. Silva and A. R. Thomas (2004). Ant Colony Optimization for dynamic Traveling Salesman Problems. ARCS Workshops.
[12] J. T. Fajardo and M. O. Carlos (2010). A mobile disaster management system using the android technology. WSEAS Transactions on Communications, Vol. 9. Pages. 343-353.
[13] A. Aydın and S. Telceken (2015). Artificial intelligence aided recommendation based mobile trip planner for Eskisehir city. Industrial Electronics and Applications (ICIEA), IEEE 10th Conference on.
[14] İ. İlhan (2016). Mobile Device Based Test Tool For Optimization Algorithms. Computer Applications In Engineering Education, Vol. 24. Pages. 744-754.
[15] G. Dantzig, R. Fulkerson and S. Johnson (1954). Solution of a large-scale traveling-salesman problem. Journal of The Operations Research Society of America, Vol. 2. Pages. 393-410.
[16] İ. İlhan (2016). An Application On Mobile Devices With Android And IOS Operating Systems Using Google Maps APIs For The Traveling Salesman Problem. Ciencia E Tecnica Vitivinicola, Vol. 31. Pages. 47-67.
[17] G. V. Glen (2013). Heavenly mathematics: The forgotten art of spherical trigonometry. Princeton University Press.
[18] M. Dorigo (1932). Optimization, learning and natural algorithms. Ph.D. Thesis, Politecnico di Milano, Italy.
[19] www.st.com/resource/en/datasheet/stm32f407vg.pdf (accessed 01.08.2016).
[20] www.st.com/resource/en/user_manual/dm00039084.pdf (accessed 01.08.2016).
[21] https://www.openimpulse.com/blog/products-page/product-category/gy-neo6mv2-gps-module/ (accessed 01.08.2016).

