A Comparison Between Optimization Heuristics of the Best Path Problem Applied to S-Route

Adriano dos Santos, Alex Barradas, Sofiani Labidi, Nilson Costa
Federal University of Maranhão
Sao Luiz, Brazil
{adriano.asr, barradas.alex, soflabidi, nilson2001}@gmail.com

Abstract - **The paper contextualizes the traveling salesman problem applied to S-Route, a system developed based upon the ant colony approach and used by the Analysis and Research in Petroleum Analytical Chemistry Laboratory - UFMA in order to generate routes in the process of fuel collection. Some heuristic approaches (constructive and improvement), such as Nearest Neighbor, Clarke and Wright, Mole and Jameson, 2-Opt, 3-Opt and Opt-K, are conceptualized and compared to the Ant Colony. Comparisons between the heuristics, and in some cases the combination of the constructive and improvement occurred through the S-TSP system. Like S-Route, the application was developed in a web environment and integrated with Google Maps API in order to facilitate visualization of the results from georeferenced data. Thus, the essay aims at identifying amongst the listed and/or combined heuristics, the best one regarding cost/benefit to be utilized by S-Route.**

Keywords*-Traveling salesman problem****;*** *S-Route; Fuel Quality Monitoring; ANP.*

I. Introduction

National Petroleum, Natural Gas and Biofuels Agency (ANP in Portuguese) is the Brazilian agency responsible for regulating, supervising and hiring all activities related to petroleum, natural gas and biofuels in Brazil [1]. In order to follow the general indicators of fuel quality traded in Brazil, ANP has the Liquid Fuels Quality Monitoring Program (PMQC in Portuguese), which is summarized in the following stages: Fuel Sample Collection (CAC in Portuguese); Sample Laboratory Analysis, Data Handling and Information Submission to ANP [1, 2].

In this context, the S-route system was developed in order to optimize part of CAC process. Having the principle of obtaining a path from a starting point going through all gas stations selected [2]. However, the systematic arrangement of this scenario can be associated with the graph theory and characterized by the Shortest Path Problem, precisely, the Travelling Salesman Problem(TSP) [3,4].

The purpose of the TSP is to find the lowest total cost of Hamiltonian cycle [5]. However, the TSP is classified as a NP–Complete problem, in other words, the execution time grows exponentially in accordance with the number of points in the route [4, 5]. In this scenario, it is suggested the utilization of heuristic methods that optimize the relationship between time and cost in order to find a solution.

TSP heuristic algorithms are an approach that do not offer the guarantee for the best solution, but seek to meet the standards through a good solution, which approximates the optimal solution and minimize time and cost execution [6].

S-Route system utilizes the ant colony heuristic method to elaborate routes [2]. Aiming to improve the performance of the S-Route system, this essay conducts a comparative study between the ant colony heuristic and other TSP heuristic algorithms: The nearest neighbor; Clarke and Wright (Saving); Mole and Jameson; Ant colony optimization; 2-Opt and 3-Opt e Lin and Kernighan (K-opt).

II. S-Route System

The S-Route System is a prototype of a web-based system that aims to automate part of the process of Fuel Sample Collection, the first phase of PMQC [2]. In State of Maranhão, the Federal University of Maranhão with its Analysis and Research in Petrol Analytical Chemistry Laboratory (LAPQAP / UFMA) is responsible by the PMQC– ANP in monitoring the fuel's quality in State.

The Maranhão State is divided in four regions by the LAPQAP called R1, R2, R3 and R4. This way, the laboratory has one week to collect the samples in each region [2].

The first week of the month is destined to the region R1 (Saint Louis city), so, the initial task is to make 10% (ten percent) of the fuel station in the group of towns of R1. The same thing happens in the others regions, even though needs to be kept the second week to R2, the third one to R3 and the forth one to R4 [2].

Therefore, the first stage of the system is a list of 10% of active gas stations in the data bank and randomly (drawing). Next, the administrator visualizes the layout of the gas stations, using a maps API and then, requests the generation of a route from the listed gas stations.

The first generated route takes into account the savings in time and distance between Federal University of Maranhão (starting point) and the drawn gas stations. Then, the path is displayed on a map, as well as the route description to be trafficked.

S-route application was developed through PHP programming languages [7] and JavaScript [8]. The use of PHP language, for business rules and information management, doesn't present a single or absolute justification in the development, but yet for implementation ease. Albeit the use of JavaScript language by the S-Route is not facultative, communication between maps API and application takes place via JavaScript and data structures in XML format.

III. HEURISTICS

The shortest path problem is originated in the purpose of obtaining the minimum route of an associated path using the graph theory [3, 4, 5]. In this case, a graph may represent a road network and geographical distance from one point to another or from an entire circuit. One of the trends of the shortest path problem is the Traveling Salesman Problem, which represents an optimization problem greatly studied by scholars from several areas, such as: logistics, genetics, production and others [9].

However, the resolution of Travelling Salesman Problem through exact methods or Brute Force algorithms, is not recommended, what is suggested is the utilization of polynomial complexity approximation algorithms that are called by Heuristic methods [10], which allow to obtain reasonable answers to the TSP.

Heuristic methods for TSP, according to the literature, can be classified into two types: Circuit Construction Methods and Circuit Improvement Methods [11].

In the first case, the circuits are built progressively, in other words, the nodes are sequentially inserted in the circuit, under the insertion conditions defined in the algorithm [11]. In the second case, the Circuit Improvement Methods aim to improve the existing Hamiltonian circuit through other methods applied [11].

Hereinafter, the heuristics which were used in the essay development and which comprise both types of methods will be presented.

A. Nearest Neighbor(NN)

The nearest neighbor heuristic starts with an empty circuit at a starting point in order to seek "the closest" point that is out of the circuit. For each subsequent interaction, the heuristic searches “the closest” point for the last point inserted in the circuit [12].

In summary, the path is constructed as per the shortest distance between these points, in other words, a point is added to a route based upon proximity in relation to the last point inserted. This distance is verified in the matrix, where d_{ij} is the distance between *i* and *j*. The metric applied to the NN approach can refer either on the spatial distance or temporal points.

In short, the NN heuristic is simple to implement, besides achieving good results for short distances, although for long distances it is not so recommended.

B. Clarke and Wright

Clarke and Wright method (CW) is based upon the concept of "gain" that can be achieved by connecting two knots in succession on a script [13]. The heuristic works similar to the Nearest Neighbor, differentiated by the search for better savings and NN searches the smallest edge.

The savings would be the cost of going and coming back to point 0 going through *a* and *b* without having to go through 0. Instead of going through *a*, getting back to 0, going to *b*, and coming back again to 0.

In essence, the algorithm computes all the savings amongst all pairs of possible vertexes using formula 1:

$$S_{ij} = C_{i0} + C_{0j} - C_{ij} \tag{1}$$

Sij represents the path savings of going and coming back to point 0 going through points i and j without having to return to 0 instead of going through i going back to 0, going to j e going back again to 0, as it is suggested by the initial routes that were previously created. *Cij* represents the cost of going from point *i* to *j*.

After calculating all the graph savings, a table of savings is created, with *i*, *j*, and the savings value. The table lines are ordered up, from the largest to the smallest savings. Then, the path is assembled, by using the vertexes of the table, from the beginning to the end of the table.

The advantage is that Clark and Wright algorithm computational complexity is O(n2), in other words, it is solved in polynomial time. The main contribution of this algorithm can be considered by the fact that it has paved the way to more powerful algorithms that emerged after this one, for example, the Mole and Jameson [14].

C. Mole and Jameson

Mole and Jameson heuristic [15] is an evolution of Clarke and Wright savings algorithm. The main difference between the two heuristics is the comparisons between the nodes and internal vertexes of the partial path, and allowance of insertions inside this path.

Mole and Jameson algorithm starts from the cost matrix that represents the route relationship, by selecting an initial vertex to build the path. After insertion of the first vertex into the route, the algorithm executes a loop that successively inserts the nodes in the path.

The previous action is performed according to two criteria: proximity and savings. The proximity criterion selects the node that is closer to the current route, according to the two distances calculated by formula 2 [14]:

$$e_{ilj} = C_{il} + C_{lj} - \mu C_{ij} \tag{2}$$

where *C* represents the cost between one vertex to another, *l* is the tested vertex to be inserted, the index *i* represents the beginning of the route *j* means the end of the route.

The savings Criterion selects the best place in the route to insert the chosen *I* vertex. This criterion follows the formula:

$$\sigma_{ilj} = \varphi C_{0l} - e_{ilj} \tag{3}$$

The place that presents the largest savings σ will be selected to receive l. The parameters φ and μ allow changing the behavior of the algorithm in several ways [15].

The literature indicates to follow the Gaskell criterion [16], where $1 \leq \varphi \leq 2$ e$\mu = \varphi - 1$, in which the following values $\varphi = 2$ and $\mu = 1$ are advisable. The algorithm ends when there are no more vertexes to be inserted in the route.

D. Ant Colony Optimization (ACO)

Ant colony optimization algorithm (ACO) [17] is a constructive meta-heuristic based upon a real behavior of ants using adaptation, cooperation and parallelism techniques [18].

The main idea of this algorithm is the agents indirect communication based on routes trailed by pheromones that are left by ants [17,18] and choice of the best route using the probability.

The ACO is based upon the probability of an ant k being in a point *i* to choose point *j* in an interaction *t*, following the formula below:

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}} \\ \kappa \in allowed_{\kappa} \\ 0, others\ cases \end{cases} \quad (4),$$

where the variable α is the pheromone weighting $(0 \leq \alpha \leq 1)$ and β is the heuristic information weighting $(0 \leq \beta \leq 1)$. $\tau_{ij}(t)$ is the pheromone present in the path between *i* and *j*, *t* being updated in every interaction. The value η_{ij} represents the ant's attractiveness at point i to visit point j, displayed in the following formula:

$$\eta_{ij} = {}^{1}/_{d_{ij}} \quad (5)$$

The pheromone deposit is calculated in a pheromone matrix τ , similar to the cost matrix, however the values of this matrix, τ_{ij}, are in constant variation.The utilization of the matrix τ values occurs after the completion of each route built by ants.

In the update, the pheromone is added to the path as below:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, if the \kappa - th\ ant use the track(i,j) \\ 0, others\ cases \end{cases} \quad (6)$$

where Q is a project constant and L_k is the length of the circuit of the K-th ant.

The pheromone is also decreased from the edges, simulating the evaporation through the following formula:

$$\tau_{ij}^{k}(t) = (1-\rho)\tau_{ij}^{k}(t) + \Delta\tau_{ij}^{k} \quad (7)$$

where ρ is the pheromone evaporation rate, defined in (0 <ρ ≤ 1).

In general terms, k ants are scattered by the nodes starting from the starting node i to j, where the choice of node j will have the highest probability according to formula 4, and thus they carry on building their paths. When all of them complete their paths, the level of pheromone on the paths is updated following the formulas 6 and 7.

Upon completion of the update, a new path search is performed until *t* interaction completion. The process ends with the end of the interactions or when all ants are on the same path.

E. 2-Opt e 3-Opt Improvement

The 2-Opt and 3-Opt heuristics are the most used and known path improvement methods in the literature. They were proposed respectively by Croes [19] and Lin [20], inaugurating the improvement of k-opt type. These methods work basically by removing k arcs from a script and replacing by other k arcs, with the purpose of reducing the total distance traveled.

To Laporte [21], the higher k value, the better method accuracy, but the higher computational effort. In this context, it highlights the preference of working with two or three arches, therefore, the use of 2-opt and 3-opt heuristics.

The 2-opt method works as follows: from a Hamiltonian path, the first step is to remove two edges from n edges, resulting in a pair of arcs. Then, the arcs are reconnected in an inverted way, the beginning of an arc is connected to the beginning of another arc and the end with the other arc's end, where the process repeats to the other arcs.

In 3-opt case three arcs are considered instead of two in order to assess the alterations in the connections between the nodes, which results in seven possible combinations.

F. Lin and Kernighan Improvement (LK)

The algorithm proposed by Lin and Kernighan [22] is based upon type k-opt improvements, differentiating itself by the k value variation during the improvement execution. In general terms, the algorithm determines which *k* to be utilized at each interaction without the need to indicate the *k* value before starting the execution.

Basically, the LK algorithm works as follows: The algorithm searches, in increasing values of *k*, which variation results in the shortest route and in each interaction the algorithm performs arc switching.

After a number of r switching, series of tests are performed in order to check if *r +1* must be considered until some stopping condition is satisfied.

Lin and Kernighan [22] defined rules to edge searching and switching, where only sequential switching is permitted, the search for an edge cannot exceed more than 5 nearest neighbors of the current point, and an edge previously excluded cannot be added to the route as well as an edge previously added cannot be excluded.

Starting from the Lin and Kernighan's original idea, other researchers have developed several modifications and improvements, always following the *K* variation line in algorithm execution. One of them was proposed by researchers Nguyen, Yoshihara, Yamamori and Yasunaga (LK-NYYY), in the algorithm called LK-NYYY [23].

LK-NYYY algorithm uses a sequence of 5-opt starting movements followed by 3-opt basic movements. By performing this sequence of movements, it's intended to balance the quality of solutions obtained and computational time.

IV. S-TSP IMPLEMENTATION

In order to find the most suitable heuristics to S-Route, it was necessary to implement each of the ones described previously. The S-TSP was responsible for embodying all heuristics equalizing all input conditions, facilitating the visualization of the results and using the same S-Route technologies [2].

The matching system allows the user to select the amount of gas stations that will be used. Then the next step consists in selecting one of the heuristics, Figure 1 ilustrates the application home screen.

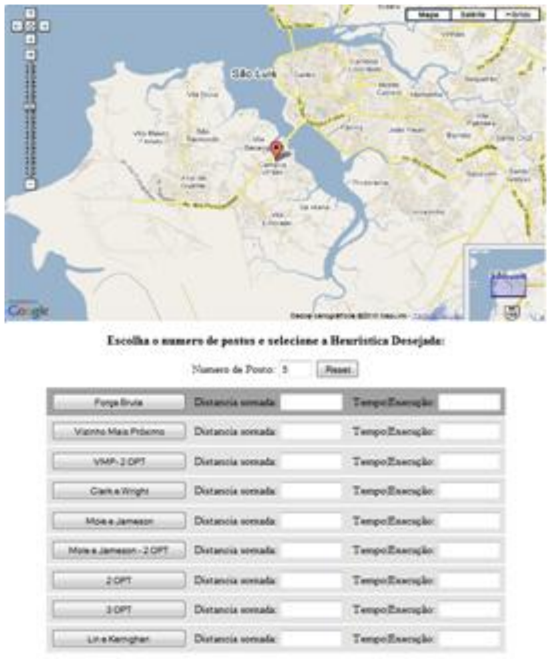

Figure 1. S-TSP.

The first text box represents the distance used to calculate the route, in other words, the sum of sections, in case of present costs in the cost matrix. The second text box is the heuristic execution time expressed in seconds.

Like in S-Route [2], the coordinates are transferred from the database to an XML file and then the JavaScript interacts with the data file. In the XML file there's the identification of each element and related latitude and longitude coordinates.

The class responsible for managing all heuristics, on equal terms, is denominated SuperMap. In this class the gas station sorting operation takes place in a vector in order to visualize the route via Google Maps API.

In order to perform the matching of algorithm performance the following heuristics were implemented in JavaScripts:

- Nearest Neighbor;
- Clark e Wright (Saving);
- Mole e Jameson;
- Ant Colony Optmization
- 2-Opt e 3-Opt;
- Lin e Kernighan(K-opt).

For results validation purposes the Brute force algorithm was also developed.

Each algorithm is represented by a SuperMap class method, where each method has the same parameters, the input criterion is the cost matrix, and the output is the route array. With these methods approach it is possible to use constructive heuristics alongside with improvement heuristics.

Some important descriptions regarding the implementations are listed below:

- Brute Force algorithm is implemented in a recursive performance in order to test all possible combinations of routes and verify which one is the shortest;
- Lin and Kernighan algorithm is implemented in Nguyen, Yoshihara, Yamamori and Yasunaga's technique. However, the algorithm has gone through some modifications in the initial movement sequence from 5-Opt to 3-opt and basic movements from 3-Opt to 2-opt;
- In Ant Colony algorithm implementation, ACO variables used the following values, all of them recommended in the literature [16]: α = 1.0, β = 1.0, ρ = 0.1, Number of ants = 10 and = Number of interactions = 10;
- All the other heuristics followed chapter III descriptions.

V. COMPARATIVE AND RESULTS

In order to find the best heuristic comparisons were carried out in two stages. At first the best constructive and improvement heuristics are verified independently. In the second stage, the best constructive algorithms as well as improvement algorithms are combined and compared.

A. *First Matching Stage.*

The first testing stage aims to find the best improvement and construction heuristic. At this stage, the heuristics are compared to each other by utilizing the total distance as per table 1.

TABLE I. CONSTRUCTIVE HEURISTICS

	10 stations	15 stations	20 stations	25 stations	30 stations
TSP	33.463	45.609	50.574	72.107	62.989
Clark and Wright	36.235	45.589	54.532	63.379	66.306
M&J	34.244	43.576	43.878	51.288	54.514
ACO	32.637	42.384	43.726	52.536	55.394
BruteForce	32.637	42.384	-	-	-

Comparing Table 1, the best results based upon distance are found by the Ant Colony (ACO) and with similar performance to it, the Mole & Jameson (M&J). The execution time was disregarded due to the fact that all the heuristics performed below 1 second.

In Table 2 the improvement heuristic performance is verified. For all improvement algorithms, the test uses as input the same array of the ordered gas stations.

TABLE II. IMPROVEMENT HEURISTICS

	10 stations	15 stations	20 stations	25 stations	30 stations
2-OPT	33.051	49.839	52.867	65.011	71.742
3-OPT	33.051	48.319	51.347	64.598	68.036
Lin and Kernighan	32.637	45.39	50.416	58.912	61.698
BruteForce	32.637	42.384	-	-	-

A performance better than Lin-Kernighan in 2-OPT and 3-OPT is noticed. Out of the OPT heuristics, the one that had the best performance was 3-OPT, which is justified by the greater amount of switching cycles.

However, Lin and Kernighan has a higher cost when compared to 2-OPT and 3-OPT, as you can see in Table 3.

TABLE III. EXECUTION TIME OF IMPROVEMENT HEURISTICS

Time/sec	2-OPT	3-OPT	Lin and Kernighan
10 stations	0.00199	0.00399	0.02399
15 stations	0.00299	0.01900	0.06599
20 stations	0.00600	0.02599	0.32200
25 stations	0.02000	0.07290	1.45299
30 stations	0.01100	0.10899	412.800

For the second matching stage the chosen improvement heuristics, picked from Table 2 and Table 3 results, were Lin and Kernighan and 3-OPT.

B. *Second Matching Stage.*

The second testing stage aims to find the best heuristic by verifying simple heuristics as well as combined heuristics. The combined heuristics are the construction ones which have their results improved by the improvement heuristics.

The heuristic defined in the first stage were tested with a larger amount of gas stations based upon distance as shown in table 4.

TABLE IV. HEURISTIC MATCHING

Stations	M&J	M&J + 3OPT	M&J + LK	ACO	ACO + 3OPT	ACO + LK
20	43.878	42.930	42.686	43.726	42.930	42.686
30	54.514	53.565	52.725	55.394	54.927	53.640
40	74.352	71.417	64.667	68.300	67.595	67.595
50	81.876	77.614	68.693	70.560	70.291	70.139
60	104.121	100.168	88.927	84.339	83.666	83.490
80	155.866	144.459	123.951	111.439	110.933	110.489
100	175.055	163.648	134.412	122.840	121.074	116.751

M&J obtained better performance over ACO regarding the preparation of route for 25 gas stations as shown in Table 1 and 30 gas stations as shown in table 4. ACO succeeded in constructing the route to the quantity of 10, 15, 20, 40, 50, 60, 80 and 100 gas stations (see Table 1 and Table 4).

With the combination of 3-OPT improvement heuristics and constructive heuristics (M&J and ACO), ACO continues to obtain better results over M&J except when the amount is equal to 30 gas stations.

By combining L&K to M&J and ACO heuristics, respectively, M&J showed better results (30, 40 and 50) than ACO except when the amount of gas stations was equal to 60 gas stations.

Table 5 shows the execution time of the heuristics as follows:

TABLE V. EXECUTION TIME(SEC) OF HEURISTICS

Stations	M&J	M&J + 3OPT	M&J + LK	ACO	ACO + 3OPT	ACO + LK
20	0.004	0.031	0.3329	0.025	0.0569	0.3209
30	0.023	0.027	41.789	0.032	0.069	4.210
40	0.037	0.207	15.700	0.085	0.230	23.200
50	0.070	0.500	101.500	0.052	0.480	102.290
60	0.088	0.976	300.430	0.115	1.006	301.530
80	0.223	2.790	1623	0.159	2.782	1731
100	0.364	7.935	7603	0.277	7.506	7570

In general, M&J has a lower computational cost ACO both in its original state as combined with the improvement heuristics. Albeit, the results do not show significant differences when they are compared to each other.

In this context, the combinations of L&K heuristics and constructive heuristics presented the best results in distance criterion albeit the execution time is high when compared to 3-OPT.

VI. CONCLUSION

The research addressed the traveling salesman problem applied to the S-Route system [2]. The objective was to compare the heuristics used originally by the S-Route, to ACO to other relevant heuristics in the literature.

Thus, the essay selected the following constructive heuristics: Nearest Neighbor, Clark and Wright (Saving); Mole and Jameson. For heuristic improvement, the research addressed: 2-OPT and 3-OPT; Lin and Kernighan (K-OPT).

The combination amongst L&K M&J and ACO heuristics, respectively, achieved the best results related to the distance.

However, the computational cost of L&K combinations was very high when compared to 3-OPT combinations, which somehow do not favor the utilization of L&K. In turn, the combination between 3-OPT + ACO got closer to the results of L&K + M&J and L&K + ACO, which did not occur to 3-OPT + M&J.

In this context, the essay suggests, for S-Route implementation purpose, the combination between ACO 3-OPT heuristic considering the cost benefit between distance and execution time.

ACKNOWLEDGMENT

Thank you very much to everyone who helped directly and indirectly and allowed the development of this research. Thanks in particular to PRH39-ANP and PPGEE-UFMA programs, and to LSI, LAPQAP and LPQA laboratories.

REFERENCES

[1] ANP, “Programa de Monitoramento”, Agência Nacional do Petróleo, Gás Natural e Biocombustíveis, 2011. Accessed september 26, 2011. Available at http://www.anp.gov.br/?pg=33970

[2] A. Barradas, A .Santos, S. Labidi, and N. Costa, “A Heuristic Approach Based on the Ant Colony Optimization for the Routes Elaboration on the Fuel Collection for the Brazilian Petrolium Agency” IARIA – GEOProcessing 2011, pp. 69-74, ISBN: 978-1-61208-118-2.

[3] N. Aras, B. J. Oommen, and I. K. Altinel, “The Kohonen network incorporating explicit statistics and its application to the traveling salesman problem”. Neural Networks, 12rd ed, vol. 9, 1999, pp. 1273–1284.

[4] K. Helsgaun, “An effective implementation of the Lin-Kernigham Traveling Salesman Heuristic”, European Journal of Operational Research, vol. 126, 2000, pp.106-130.

[5] C. H. Papadimitriou, “The Euclidean travelling salesman problem is NP-complete”. Theoretical Computer Science, 4, 1978, pp.237–244.

[6] P. H. Siqueira, M. T. A. Steiner, S. Scheer, “A new approach to solve the traveling salesman problem”. Amsterdam: ScienceDirect - Neurocomputing: A new approach to solve the traveling salesman problem, 2006.

[7] PHP, Manual, 2013. Accessed september 23, 2013. Available at http://www.php.net/manual/pt_BR/preface.php.

[8] J. Resig, “Pro JavaScript Techniques”, Apress, 2006.

[9] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook “The travelling salesman problem: a computational Study”. Princeton: Princeton University Press, 2006. ISBN 978-0-691-12993-8.

[10] A. Diaz, M. Laguna, P. Moscato, F.T. Tseng, F. Glover, and H. M. Ghaziri, “Optimización Heurística y Redes Neuronales”. Editorial Paraninfo, Madrid. 1996.

[11] D. E. Rosenkrantz, R. E. Stearns & P. M. Lewis II, “An analysis of several heuristics for the traveling salesman problem”. SIAM J. Comput., vol. 6, 1977, pp. 563-581.

[12] M. M. Solomon, “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints”. Operations Research, vol. 35, ed. 2, 1987, pp. 254-265.

[13] G. Clarke, and J. Wright, "Scheduling of vehicles from a central depot to a number of delivery points", Operations Research, vol. 12 ed. 4, 1964, pp. 568-581.

[14] M. R. Heinen, and F. S. Osório, “Algoritmos Genéticos Aplicados ao Problema de Roteamento de Veículos”. Hífen, Uruguaina. vol. 30, ed.58, 2006.

[15] R. H. Mole, and R. S. Jameson, “A sequential routing-building algorithm employing a generalized savings criterion”, Opl. Res Q, vol. 27, 1976, pp. 503–512.

[16] T. J. Gaskell, “Bases for the vehicle fleet scheduling”, Opl. Res. Q, ed. 18, 1967, pp. 281–294.

[17] M. Dorigo, “Optimization, Learning and Natural Algorithms”. M.. PhD thesis, Italy: Politecnico di Milano, 1992.

[18] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: optimization by a colony of cooperating agents”. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics vol. 26 ed.1, 1996, pp. 29–41.

[19] G. A. Croes. “A method for solving traveling salesman problems”. In: Operations Research n. 6, 1958, pp. 791-812.

[20] Lin, S. “Computer solutions of the traveling salesman problem”. In: Bell System. Technical. Jour nal. n. 44, 1965, pp. 2245-2269.

[21] G. Laporte, H. Mercure, and Y. Nobert, "A Branch and Bound algorithm for a class of asymmetrical vehicle routing problems". Journal of the Operational Research Society. vol. 43, ed. 5, 1992, pp. 469-481.

[22] S. Lin, and B. Kernighan, “An Effective Heuristic Algorithm for the Traveling Salesman Problem”. Operational Research, vol. 21, 1973, pp. 498-516.

[23] H. D. Nguyen, I. Yoshihara, M. Yamamori, “Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems”. IEEE Transaction on System, Man, and Cybernetics-PART B: Cybernetics, vol. 37, no. 1, 2007, pp. 92-99.