INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 6, Issue 3)
Available online at: www.ijariit.com

Heuristics and Meta-Heuristics optimization methods in solving Traveling Salesman Problem TSP

Jawaher A. Al-Ghamdi
ijawaher.a.al.ghamdi@gmail.com
King Khalid University, Saudi Arabia

Eyad R Al-Masalmeh
eyadmas.wo@gmail.com
Al Baath University, Syria

ABSTRACT

In modern societies there are increasingly more often problems of various kinds, and tests are needed to solve them in experimental ways. Although, develop a mathematical model that closely matches the reality to solve a real life problem is very complicated, since many of these models might has to contain very large number of variables (as a heuristic model that optimizes problems solving results). Furthermore, these shows as difficult problems in controlling subjec-tive behaviours, so They are making it even more complicated than these models resemble reality (wrong solving model leads to a more complex level). The purpose of this research is the study of combinatorial optimization problems using approximate methods. In particular, this work focuses on the analysis of meta-heuristics algorithms based on history and population related to the solu-tion of Travelling Salesman Problem (TSP) like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA) and many others, as well as hybrids, which allow efficiently to solve generic problems.

Keywords— PSO, ACO, TSP, Meta-heuristics, NP-Hard

1. INTRODUCTION

One of the most used optimization methods that aims to provide a higher level in solving NP-complete problems (more than one true solution to the same problem) mention from them Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA), Bees Algorithm, and many others. These optimization methods which represented as a heuristic type algorithm based on the amount of population in a specific problem solving, which simulates the natural behaviour of related algorithm, whose main characteristic is that they have memory and a specific number of iterations (Algorithm generations).

Bees, Ants and Swarms look at space in search of food and communicating with each other to see which of them has found more floral density and where. Then, each of them continues its search by modifying its search function into a better place found by the optimization method that each used in aims to find the best search result. Thus, the generation ever finds places with more efficient results (better results from optimization). In heuristics Methods, each generation is a potential solution to the problem, and has a current position and speed. The population that should be used by each generation is initialized randomly and then iteratively evolving to a better solution by updating the velocity and position of each particle in PSO, pheromone amount in ACO and Bees algorithms, and cooling degree in SA. First speed is updated by taking into account the best global position funded by the generation and the best position found so far by the generation itself, and then the position is updated by applying the rate of the founded optimized results.

The adopted solution is the best optimized result funded by the entire generations that con-sider as a heuristic solution. Although, it cannot be ensured the extracted solution is the optimal outcome, yet it might have to provide more generations in order to get more optimization (global-maximum point where there is no more optimization for a large number of iterations whereas it might find an optimization after this large number). This method as described, serves to search for optimized solutions in spaces of continuous search. However, the problem that we face is a discrete problem called the Travelling Salesman Problem (TSP) this problem will be adopted in order to show the benefit of using heuristic methods to provide an enhanced level of optimization. TSP is NP-Hard type which can be described as follows: A must visit and number of cities through all of them, without repeating any, and at last returning to the city of origin starting, so that named as a Hamiltonian Cycle (HM) will be generated. The objective of the problem is to find the optimum cycle to be performed by the traveller. The aim behind this problem is to find the best results in finding the path that involves the lower distance between all possible solutions.

The traveling salesman problem is to find the best Hamiltonian cycle (the minimum distance) that could make a traveler who has to pass n cities, and knowing the distances between them. For the Hamiltonian cycle it must be fulfill that: traveler must come from a

particular city, through all the other cities without repeating any, and return to the starting city or point. What adds to the original TSP is the distance from the city i to j does not have to be the same as the distance of the city j to i.

The difficulty of the resolution is that possible cycles can be carried out in increase exponentially with the size of the problem. Hence, try each of the possible solutions is not an option. Since this paper seeks to resolve in this paper is one of the versions of the Travelling Salesman Problem (TSP) and explain its structure, namely the asymmetric variant, also known as ATSPs, that belongs to NP-Hard problems optimization combinatorics, related section will begin introducing the characteristics of the decision problems and ranking them in order to describe what it is NP-complete problem (which belongs version of decision TSP) and then the NP-Hard, thereby class to contextualize the TSP. Finally, describe the TSP and the asymmetric version, which is the problem that really concerns us in this work in order to explain optimization methods.

1.1. TSP Main Subject Problem

The optimization problem involves both a minimization and maximization problem, each one has associated with a set of instances in the same cycle of the problem. Usually it consists of several parameters and variables without specific values. i.e, TSP problem is a general problem looking for a Hamiltonian cycle with a minimum cost, while an instance of TSP is a particular problem where it is already known the number of cities that the father cycle has, connections and the costs to go from one to another. Besides all optimization problems, which it does looks for the best possible solution of the problem (minimum travel in the case of TSP [1]), has several versions that meets NP problem definitions [2], such as: the evaluation version, and localization of decision.

The case of decision problems is where there is only two possible answers ”yes” or ”no”, so the decision version of the optimization problem looks like this: is there a possible solution for minor or major problem (as minimization or maximization) for a given number, and more specifically for the TSP would be: for a given number k, there is a Hamiltonian problem circuit less than

or equal to length k of classification class of decision problems, depending on the computational complexity involving its resolution sciences is responsible for the computing of the problem parts. For this purpose they consider all possible algorithms that can solve this problem and the resources required (time studying and memory which is the problem in its implementation) [1],[2].

2. TRAVELING SALESMAN PROBLEM (TSP)

The name that has been given to this problem today is a bit mysterious [3], since they found documents which the authors use this nomenclature, but none of them are given indications that the document author is the creator of the name ”Travelling Salesman Problem ”, but suggests that the name was already settled down. The TSP can be formulated as follows [4] A traveling or salesman must visit each and every one of the n cities given once, starting with one and returning to the same city of origin at the end of the route, so that a cycle is generated. Since there is to be met that each city goes exactly once, the resulting cycle is an Hamiltonian.

What is sought in the TSP is to choose the route or tour (the order in which the cities are visited) that optimizes the result, e.g. the least cost involving (generally use travel cost and can be referred to the distance, time, the cost of transporting, that depends on the objective of the problem. The most common in This type of problem is traveled) distance between all possible cities in the cycle. A solution for an instance of the TSP can be represented as a permutation of the induces of cities (a sort of cities). Happiness permutation is cyclic so the absolute position of a city in the tour is not relevant, what really matters is the sequence, e.g. the order relative between some cities and others. What this means is that it is not important to a city to be visited as a first or second city, which is what affects cities have visited before and after it, in other words, departure city does not affect the result of the problem (since it’s the source and the destination point in the cycle of visiting).

It is such complexity that the TSP is part of NP-Hard problems [5] in which it shows that Hamil-ton was NP-Complete cycle, which means that TSP is NP-Hard (it will be demonstrated below). In addition, it is an optimization problem (which involves looking for values for discrete variables. Hence, the optimal solution founded in regarding a given objective function) and undergoes the phenomenon of combinatorial explosion that means when the number of variables of the problem is increasing, the number of possible combinations and also computational efforts, in exponentially increment optimization way, from the previous TSP proved to be as a combinatorics optimization problem.

Theoretically the algorithm of “brute force” that delivered with the aim to test all possible routes of the problem (as a cycle), by this solution way it considers as an optimal and easy to way to be raised. The possible routes that can be taken, by giving in cities and passing exactly once for each one of them, are in!, though, because of the problem I it must specify the starting point (Although not specified, being a Hamiltonian cycle, the result of the problem is independent of the starting point specifying), while there is a possibility to calculate all possible routes of a TSP problem as (N-1) in the meeting with the original number of cities in the cycle, which is demonstrated by the increases of the problem space. Hence, it increases in an exponential cost with the time line increasing (rather than factorial increasing). As a formal description for TSP, it will introduce below the graph theory [6], then finally expose the problem formulation:

1. G is the graph of the problem and consists of V, A and C, then it’s representation: G = (V, A, C).
2. V is the set of vertices or nodes, which in TSP case are the cities that are the problem V = {1, 2, ..., n}. The size dimension we call V (n), which is equivalent to the number of cities that is the problem.
3. A is the set of edges or arcs of the problem, i.e., road or union there to get from one vertex to another. We talk about a complete graph if for every pair of vertices an edge that unites them.
4. C is the cost matrix, where c_{ij} is the cost of (usually in terms of distance) go city i to j, that is, the cost associated with the edge (i, j). C is usually nxn size, placing in the ranks the cities of origin and the destination columns. As each row is placed the city distance from that row to all others, and are n cities in altogether there are n columns and like any city can be taken as origin there are n rows.

When found a problem of TSP type, the only data that journeying resolution is those that form the graph of the problem, e.g. the vertices, edges and costs. In this document we assume that the graph that we provides is the complete graph, since being the most general case approach will help to do any particular case.

The graph is complete means that for each pair of cities, there is a connecting path between them. If we have a problem that does not bind directly two vertices, we created a fictitious edge binding which will cost equivalent to the cost of the shortest road linking the two cities. For example, in Figure 1 we could add a fictitious edge (1-7) cost equivalent to 9 roads (1-3-7).

A succession of edges (a_1, a_2, ..., a_k) where the final vertex of an edge is the same as the opening of the next ridge, it is a way. In addition, we call simple path if it meets the condition that does not go through the same vertex more once. A cycle is a path whose end vertex of a_k coincides with the initial vertex of a_1 and It is a simple cycle if the path is also composes.

When we talk about a simple loop that does not use all the vertices of the graph problem are talking about a sub tour. In the opposite case, when a cycle is simple and passes through each and every one of the vertices composing the graph which we are facing a problem or Hamiltonian cycle tour so you can denote as (a_1, a_2, ..., a_n). It is the latter type that is required to build on the TSP, further including the condition that it is of minimal cost. In Figure 1 see a graph problem where there are 8 vertices represented with numbered yellow circles, several edges that are recognized are straight binding between two vertices and the cost of each edge is represented with a number next to their respective edge. The road is marked in bold one possible Hamiltonian cycle in the graph.

One way of formulating TSP use in [7,22], which is represented as a binary variable used in integer linear programming model. While the objective function of TSP is to minimize the cost of the route that leads the traveller using genetic algorithm, and depending on the model, it looks like this:

$$Min \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} (c_{ij} x_{ij}) \tag{1}$$

Where n is the number of cities the problem, c_{ij} is the cost of going from city i x_{ij} the j and is the binary 0,1 the problem variable, taking the value of one if the Hamiltonian cycle holds that edge (i,j) and the value zero otherwise (so, logically, if the path is not going linking the city i with j, its cost is not added). In addition, the issue must meet the following restrictions:

$$\sum_{i=1, i\neq j}^{n} (x_{ij}) = 1 \qquad j = 1, 2, 3 \ldots n. \tag{2}$$

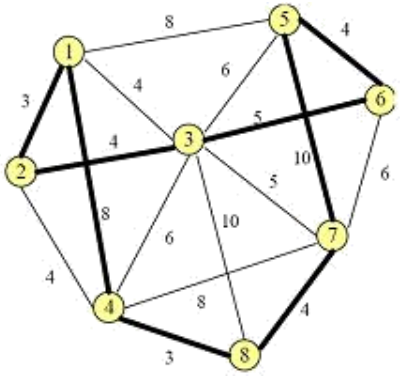

Fig. 1: Hamiltonian Cycle

The first constraint ensures that you cannot reach the same city from several and yes it comes to it, i.e. the arrival city has one and only one source. The second constraint ensures that you cannot go from one city to several and also all cities have a way out, forming part of the route, which means that each city has one and only one destination. The third restriction requires that is only one way which covers all cities, i.e., prevents two or more disjoint paths complement together covering the n cities. To this it ensures that all road has to go through the city 1. ???? variable is a free choice indicates the order in which the road is done, that is, if $u_i = t$ mean that the city ??i is accessed in step t, where t = the order 1, 2, ..., n. We define the travel to end in city 1 (which does not mean loss of generality since it is a Hamiltonian cycle, so that no starting point It affects the solution). It is true that $u_i - u_j \leq n-1$ since u_i cannot be greater than n ($u_i = n$ is the last city visited) and u_i cannot be less than 1 (as Very soon visit the first), in addition to the case where the variable $x_{ij} = 1$ we have $u_i - u_j + nx_{ij} = (t) - (t + 1)\, n = n - 1$. Compliance with the three constraints requires that the road is a cycle created and that is also Hamiltonian cycle.

Normally TSP is a real geometric optimization problem. a technique aim to reduce the searching space in TSP For instance, they snap input points to nearby grid points, then if doing so does not significantly affect the objective cost.) Recently, this technique was used by in the aim to optimize NP-Complete problems [21].

2.1. Computational complexity of TSP

It may not be practically possible to solve due to excess resources it requires. The main problems according to this consideration are [3]:

- **Class P:** It is polynomial complexity algorithms (P). Can be solved in polynomial time with a machine of deterministic Turing (given some input variables and the state is the computer, if there are several possible actions, the machine can only do one of

them, in the non-deterministic the ma-chine and continues replicates each for a different branch) and sequentially (performs each action after the previous finished). Polynomial time is the ideal time, which means that the execution time of the algorithm is below a certain calculated value (with a polynomial formula, which takes into account the data input, so the polynomial time is proportional too), so an efficient algorithm is one whose complexity is polynomial.

- **Class NP:** Decision problems are solved in polynomial time (P) with a non-deterministic Turing machine (N). We can say that the P class is a type within the NP class. This group includes many optimization problems and search which aims to find out if there is a certain solution, or if there is a better than known so far.
- **Class NP-Complete:** For a NP-Complete problem will have to meet two characteristics: it is an NP (full are the most difficult problems of class NP), and that all NP problems can be reduced (relate to problems, so that if the first can be solved by an algorithm ensures that there is an algorithm that solves the second) in each NP-Complete problems, e.g. all NP problem of polynomial transformation can become NP-Complete. All problems that meet the condition of the polynomial transformation, although not meet membership NP.
- **Class NP-Hard:** The NP-Hard class is not exclusive decision problems and not have to meet the condition of being NP, in fact, all NP-Complete belong to the NP-Hard, but not fulfilled the statement to the contrary. NP-Hard problems are those that meet all NP-Problems can be transformed into NP-Hard polynomially.

So far, it's not found any efficient algorithm (polynomially bounded) to provide the optimal solution for such problems, nor for the above. Therefore, how to cope with these problems is to perform an exhaustive search (enumeration of all possible solutions and choosing the best) if the inputs are "small" or otherwise find a near optimal solution through polynomial approximation algorithms.

These problems, at least as difficult as the NP because if we find an algorithm that resolved in polynomial time consider as one of NP-Hard problems (it is believed that there is no polynomial algorithm to solve them, but has never been proven) , then it could build a polynomial algorithm for any NP problem by reducing the problem in NP-Hard and subsequent execution of the algorithm that solves the NP-Hard. If this algorithm is found it means that P = NP and this would be very important for the theory of computation discovery. Given the characteristics of the NP-Hard class, yes it can assure that today is NP-Complete are the intersection of the NP with NP-Hard.

Figure 2 shows an Euler diagram which represents the relationship between the types of P problems discussed above. Due to the number of unknowns that exist on the subject today, we see the two main versions: left of the case has not demonstrated the existence of a polynomial algorithm that solves an NP-Hard problem and right, otherwise, so P is equal to NP and NP-Complete, being also all NP-Hard. [5]. Traveling salesman problem belongs to the NP-complete class (therefore also It is NP-hard).

To prove discuss below two conditions must meet any problem in order it belongs to that class [8]:

1. The TSP is an NP problem. To verify that a tour of the problem is valid, the first thing to do is check that you have all the vertices of the problem, then add the costs of all edges and check that the total cost of the path is the best. To do this process it can be done in polynomial time, so that the TSP is demonstrated belongs to the class of NP problems.
2. All problems can transform NP-Complete by polynomial transformation, e.g, the problem has to be NP-hard. To perform the test cycles take as reference of Hamiltonian cycle, since they are known to be NP-complete problems.

Start with G = (V, A), this being an example of Hamiltonian cycle. Starting hence create a complete graph G = (V, A), where A is the set of edges (i,j) for all i, j belonging to V (except i = j). The costs of these edges are 0 if the edge belongs to A (Hamiltonian cycle) and 1 if found one edge that does not belong to A. Once created the complete graph assume that h is a Hamiltonian

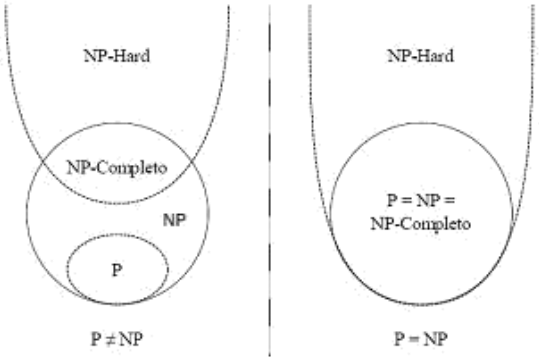

Fig. 2: Problems as computational complexity

cycle which exists in G, so the cost of each edge of h in G is 0, which makes the total cost of travel from h in G is 0. Hence, it means that if the graph G has a Hamiltonian cycle, then G will have a 0 cycle cost. Conversely assume that G has a cycle h with cost 0. This only it can occur if all the h edges are 0 cost and then it must belongs to A. Both assumptions imply that G has a Hamiltonian cycle if and only if G has a 0 cycle cost, so it is shown that the TSP is a NP-Full problem.

2.2. Applications of TSP

The TSP has a good application, especially in the fields of transportation and logistics, both used to optimize themselves routes, to serve as a tool (Sub-problem) to greater combinatorial optimization problems. But the truth is that it can be used in any situation requiring to select a number of nodes in a particular order to minimize the cost, though TSP goes far beyond into planning route issues, also used for other applications as they can be: minimization of displacement in manufacturing, robotised collection of

material storage, scheduling a drilling machine, scheduling a car move in the street, scheduling a CNC machine, organization of workstations, and many others. Then briefly explain some of the applications of the concerned problem [9]:

(a) Machine Scheduling Problems: One of the most studied problems is the organization and sequencing machines, where the vertices are the n tasks to be processed sequentially on a machine and the cost matrix is the set of c_{ij} being the cost of commissioning (configuration of changes tool) of changing the task i to j. When all tasks are processed the machine is reset, by returning it to its initial state, which will cost c_{n1}. What is sought in this application is to find the sequence of tasks that minimizes the total costs of starting up the machine tasks.

(b) Cellular Manufacturing: When family pieces that require to have a similar prosecution, which grouped and processed together in a specialized cell to improve efficiency and to re-duce costs. In most cases cells robots are used for handling the material and thus reduce the needed processing time. The sequencing of activities to be developed by the TSP robot can be considered.

An example of this problem is as follows: Robot sequence activities in a two machines robotic cell (M1 and M2), having n types of parts and having to be processed by M1 and M2 subsequently. The process of one piece is: the robot picks up the part of the initial stock, takes to the M1, M1 processes, the robot picks the M1 and M2 leads to the processes and finally M2 robot picks the M2 and leads to the end of the process.

For this problem are considered 2 cycles of possible robotic movements, in the first robot picks up a piece of M1, M2 leads it to wait till it processed, and leaves collected in the final stock, then goes to the initial, picks up another piece, it leads to M1, waits to be processed, leads to M2 and so on, while in the second cycle considered the robot picks a piece of M1, M2 and leads instead of waiting to be processed, then to another piece in order to reach the initial stock and leads to the M1, M2 again (wait if has not finished), take the piece to the end stock, will the M1 (wait if necessary), take the piece and brings it to M2 and so on.

What should be done is to select at each step (for this example, when it finishes processing the M1 piece) which is chosen robotic cycle to minimize the cost. The choice of most appropriate cycle for a given stage depends on the time which it takes for the robot to move between machines and stocks and time required for the machines 1 and 2 to perform the process of the piece.

(c) Frequency allocation Problems: This problem may be, for example, assigning a frequency to each transmitting of a com-munication network from a given set of possible frequencies that satisfy the restrictions of interference. The restrictions can be represented in the graph G = (V, A), where V are n number of transmitters and cost equivalent to the required tolerance, which is should choose a frequency (between possible), such that the frequency of i transmitter reflects j transmitter (in absolute value), in which it has to be greater than the tolerance c_{ij}. What is sought to minimize the frequency range used between possible frequencies, while meeting the restriction of tolerance.

2.3. TSP Types

Many different versions of this problem, which differ in a certain restriction added to the original problem. Some of the most studied in TSP are those presented below [7-10]:

1. Metric TSP or Delta-TSP: This type represents the costs (or distances) of the problem as inequality triangle, e.g. the cost of going from one city to another passing through an inter-mediate, it cannot be less than the cost of going directly $c_{uv} + c_{vw}$ c_{uw} for every vertex u, v and w belonging to problem vertex set V. When the cities are in a plane problem there are several ways to calculate the distance [7], mention below two of these ways:
   (a) Euclidean TSP: the distance between two cities is the Euclidean distance between two direct points in the plane of these cities.
   (b) Rectilinear TSP: To calculate the distance between two cities is Manhattan distance, e.g, the sum of the differences between the coordinates "x" and "y" of cities in the plane (in a case of undirected points might be applied into it). This is the case, for example, the transfer of a robot from one point to another, whose mode of movement is to perform a displacement first in a coordinate, and then the other.
2. Symmetric TSP (STSP): This is the case in which the costs of the problem or the distances in this problem are symmetrical, e.g. the cost of going from one city to another is equal in both ways: $c_{vw} = c_{wv}$ for every vertex v, w belonging to V. This feature makes the case of an undirected graph, so this version of the problem, it is immaterial whether the cycle runs in one direction or counter-clockwise, and therefore feasible routes for this kind of problems is reduced to half, being ((n-1)!) / 2 possible cycles.
3. Asymmetric TSP (ATSP): It is the opposite version to STSP, e.g. given two cities v and w, the cost going from v to w does not have to be the same as that which is going to v from w.

2.3.1. Asymmetric Travelling Salesman Problems, ATSPs: For the Travelling Salesman Problem Asymmetry can be the case in which that given two cities, there are no roads in both directions than the one directly or even the distance or cost about one city to another is not the same in one direction than the other, (but can also be the same). With that inequality of a single pair of vertices and it is a directed graph. Hence, the problem is already asymmetrical.

The ATSP is broader than the symmetrical case, in fact, a method which capable to solve ATSP it can also solve the symmetric version since this is a case asymmetrical concrete where possible solutions are reduced into half. For one problem ATSP with n cities, the number of possible cycles is (n-1)!.

The ATSP can be solved by converting it to a symmetrical TSP problem [17], since it is easier to be handled. For this reason, the option is double the size of the array, by having a matrix of cost for size and in the asymmetric problem, they became asymmetric cost matrix with size 2n as shown in Figure 3.

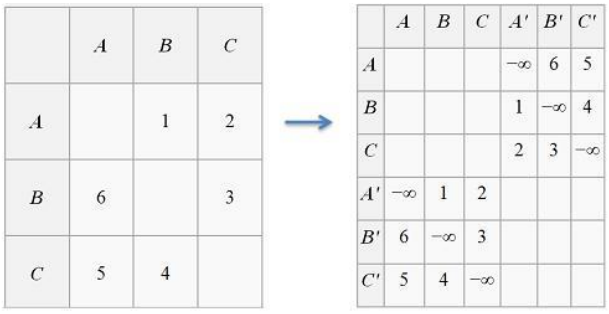

Fig. 3: Conversion of ATSP to the symmetric case

The procedure for converting is doubling every city in the problem by creating a second abstract node. Duplicated nodes are assigned with very low weights (Use -1, but it can also be zero, unless there is any edge of original graph worth 0, in which case it would have to be a smaller value for allow to a free jump between a node and its fictional) to allow routes to find a low cost that link the abstract node with the real node.

3. SOLVING METHODS FOR OPTIMIZATION PROBLEMS

Introducing the main methods used for solving a combinatorial optimization problem, as is the case of NP-hard, and therefore the Asymmetrical Travelling Salesman Problem (ATSP), these are: [13][9].

- Accurate Algorithms.
- Approximation Algorithms and Heuristic Methods.
- Hyper Heuristics.

The third type, called ”Hyper Heuristic” is used in some situations where a particular problem has a good resolution or by exact algorithms, or with heuristics, but it can break in several special cases or ”sub-issues” for which they are possibly best heuristics or exact algorithms, e.g. what the Hyper Heuristic Algorithms function is to create a new heuristic by combining parts of optimization methods, also compensating for the weaknesses of others [18]. The following briefly explains each of the first two types, for subsequently focus on approximation algorithms, specifically in meta-heuristic, Finally, it be described what that meta-heuristic.

3.1. Exact Algorithms

The exact algorithm is characterized by finding the optimal solution to the proposed problem. The fact that is always provided as the global optimal solution which consider as an advantage of the exact algorithm, while the runtime of these algorithms is their main disadvantage as it grows exponentially with the problem size. Although, the spent time to solve a problem NP-hard combinatorial optimization by an accurate, if there is the method in most cases is usually so great that it inapplicable. In fact, this type of algorithms is a problem even infeasible for the small amount of TSP cycle nodes “cities”.

This paper describes very briefly this type of algorithms since most engineering problems can-not be solved by exact methods, and it must resort to approximate as will be explained later. Some of the most important exact algorithms described below are [13] [19-20]:

- **Enumerative Techniques:** The most direct solution is to try all possible combinations of problem and find the target for each permutation function, choosing the one that achieves the best result (lower or higher value of the target function, depending on whether it is a minimization problem (ATSP case) or maximization). The problem with this solution is that the execution time is of order (n!), the number of cities to visit would be n, which implies a long time. Sometimes it can be reduced to some extent, though the time, eliminating computational possibilities due to the dominance, which is in what the branch and bound algorithm is based or branch and bound strategy. This algorithm involves the process of making a space, and partial list of all possible solutions, by generating a spanning tree. The tree is organized so that each node of level k represents a part of the solution, consisting of k stages that has been already made, and their children are possible extensions to add a new stage. The pruning process is to calculate a dimension (function) for each possible extension and if the dimension is worse than the best solution found so far, in order that branch (the whole pruning) is further explored. The method is carried out in 3 stages:
  - — Selecting the node between the set of living nodes (those that have not already been trimmed).
  - — Branch (construction of the possible children of the selected node).
  - — The algorithm ends when all living nodes are depleted, which it ensures that the left is the best of the problem (depth first exploration technique), since all which have been pruned had a dimension than this.
- Relaxation and Decomposition Approaches: Another well-known techniques dimension (with much success until that appeared dimensioning and pruning) is the Lagrangian and relaxation, where seek to eliminate bad restrictions, placing them in the objective function as a penalty (to them as a penalty also prevents break them to get a better result. The weight of them depends on Lagrangian multiplier that is applied). Moving these restrictions, the objective function, the resulting sub-problem is easier to solve. Is necessary to solve the sub-problem iteratively to find the optimal values for the multipliers.
- Cutting Plans Based on Polyhedra Combinations: It is based on the application of polyhedral theory to solve numeric problems. Given it as a theorem, if we can enumerate the set of linear inequalities that define the possible points and extreme rays of the problem, we can solve the integer programming problem using linear programming techniques.

As a summary exact method are very effective because they achieve the optimal solution, but on the other hand, they are not efficient because of the excessive time needed for it to deliver its goal. It is the second reason, which causes combinatorics optimization problems, in their vast majority, resolved through algorithms approach.

3.2. Heuristics Algorithms

ATSP try belongs to the category NP-hard combinatorial optimization (aim to find the maximum or minimum of the function target problem from a set of finite solutions, taking into account its decision variables to be a discrete values). NP-hard problems are "difficult to resolve" and algorithmic terms. Hence, can say that it means there is a probability that cannot guarantee to find the global optimum (best solution among all possible) for a reasonable time execution, so the exact methods are not efficient for this type of problems, which leads to approximation algorithms.

Heuristic algorithms are used in problems where the speed of implementation is as important as the quality of the obtained solution since it found a reasonable balance, giving a good solution (although not necessarily the optimum) in a computation time moderate [6] [14]. A heuristic method is a procedure for solving an optimization problem using a well-defined intuitive approach, in which problem structure is intelligently used for a good solution. There are several reasons for using heuristics:

1. There are some problems for which no exact method unknown we can solve them.
2. In some cases where the exact method does exist, the computer cost is so great that it makes it infeasible.
3. The flexibility of approximate methods is much higher than the accurate leaving, for example, add a complicated modelling condition.
4. The approximate method is usually a part of a global process that ensures the optimal prob-lem. There are two options for it:
   (a) The approximate method provides a good initial departure solution.
   (b) The approximate method is an intermediate step in the overall procedure, using for ex-ample to select variables in simplex method.

Heuristic algorithms depend on a specific problem for which they are designed, which is not true, for example, with branch and bound exact algorithm whose procedure is predetermined and is virtually independent of the problem in which it is used. Although, the approximate methods depend on the determined problem, they can be transferred to other some particularized techniques and specific ideas of the algorithm is structured to produce a specific output once its processed. Therefore, to apply a heuristic method must specify the particular problem which will be adopted to solve it.

3.2.1. Measuring the Quality of a Heuristic Algorithm: A good heuristic algorithm must have the following characteristics [6][12][13]:

(a) Efficiency: That the computational effort is realistic and has a consistent quality for the obtained solution.
(b) Goodness: The offered or the adopted solution should be good enough, that is, close to the global optimum of the problem (optimal or near optimal solution to the current problem situation).
(c) Robustness: It must have a very low probability of getting a bad solution (away from optimal). There are several ways to measure the quality of the optimization algorithms:
   (i) Compare it to an optimal solution with problem that applying an optimal solution is known and pre-defined (although the collection is reduced). Hence, comparison the percentage deviation of the value measured that offers heuristic methods when applied to the problem and the optimal method thereof problem. This measurement is made for various examples and is the average deviations.
   (ii) Compare it to a level where we do not have at even a limited set of examples of the problem that already defined. The goodness of this measure obviously depends on how good is the elevation of the chosen problem, i.e, the closeness of the dimension to optimum of the problem. Although, have no information of how good it is the dimension this measure without interest.
   (iii) Compared to a truncated exact method, such as that of Branch and bound if we set a limit of iterations a maximum execution time, or some other stopping criterion that prevents time problem these exact methods. Although, we know that so most likely not achieve the opti-mum, the value obtained serves dimension for comparison with our heuristic algorithm.
   (iv) Compare it with other heuristic algorithms. This comparison is the most commonly used for NP-hard problems because for them are known Heuristic good. The effectiveness of this comparison depends on what good the other heuristic method that comparing.
   (v) Analyse the behaviour of adopted algorithm in the worst case it might has e.g. implement the worst examples for algorithm and narrow, analytically, the maximum deviation from the optimum. The problem with this method is that by taking the worst case the results are not representative of the average behaviour of the algorithm.

3.2.2. Heuristic Classification and Approximate Methods: Due to the variety of existing heuristics (some even designed for specific problems without generalization option), it is very difficult to make a complete classification of them. A sort mode is given below [13][6], consisting of several categories:

(a) Decomposition Methods: These methods used in the aim to decompose the problem in easier way to solve it as a sub-problems, though not forgetting the needed time to address the goal, which all these divisions belong to the same problem.
(b) Inductive Methods: This type of methods uses the "opposite" idea to the group above decom-posed problem in the decomposition method type, since it is simpler to generalize versions of the full problem cases. It uses properties or valid techniques for easy analysis, in more complete problems.
(c) Reduction Methods: In type of algorithms it tries to identify and introduce some of restrictions problems to the properties that meet a good solutions also this problem must be already simplified. Because of this feature, such methods have restrict risk so that leaves out the optimum of the original problem.
(d) Construction Methods: These methods build a solution step by step for the treated problem, every step adding an element to complete a full solution. It is iterative methods and deter-ministic that select the best choice at each iteration of the algorithm. They are widely used in classical traveling salesman problems like web concerned.

(e) Local Search Methods: These methods are based on a solution problem, unlike the construc-tion that are gradually created. From this initial solution are gradually improving, getting at each step a new solution, the product of a movement of the solution before the new, better value. The method is terminated when for the existing solution is not found any other solution accessible to improve the current value.

The methods mentioned so far are a part of the traditional Heuristics. In recent years there has appeared a new category named as Meta-heuristics Methods that use the basis of the local search methods. Providing as a definition for it as the following:

- Meta-heuristics Methods: define a class of approximation algorithms combine traditional heuristics with efficient exploitation strategies in a search space. These methods are more recent, complex and emerged heuristics to improve the optimized results.

3.3. Meta-heuristics Methods

The meta-heuristics procedures are a type of heuristic procedure arise in the 70s, under the idea of combining various heuristic methods in order to reach a higher level that achieve efficient and effective exploration strategy over a specific search space. Until the time of working with meta-heuristics they were known as modern heuristic. Therefore, although some items are still used the terms "modern heuristics" and "classic heuristics", most used "heuristics" to refer to the classical heuristics and "meta-heuristics" to refer to this specific type of heuristic methods, which are more recent and complex, as a definition [6][13][14]:

The meta-heuristics procedures are a class of approximate methods which they are de-signed to solve difficult combinatorial optimization problems, in which the classic heuristics are not effective. Meta-heuristics provide a general framework to create new hybrid algorithms that combines different concepts derived from artificial intelligence field, biological evolution and statistical mechanisms. This intelligent design for general strategies creates or improve, in high yield, so heuristic procedures structured with the aim of solving a problem efficiently. A formal definition of the elements involved in the meta-heuristics' methods discussed below:

$$|M = (\tau, \mu, \lambda, \Theta, \Phi, \sigma, \Xi, \iota) \quad (3)$$

as a definitions for the previous parameters:

1. τ is the set of elements that manipulates the meta-heuristic, usually match the search space.
2. **M** is the number of structures (solutions) working with the meta-heuristic.
3. λ indicates the number of new structures generated in each meta-heuristic iteration, resulting from the adopted application Φ to Θ.
4. $\Theta = (\theta_1,....,\theta_\mu)$ are the variables that store meta-heuristic solutions, where all θ_i the set Θ belong to τ.
5. Φ: τ_μ x $\Xi \rightarrow \tau_\lambda$ is responsible for changing the operator structures.
6. $\Xi = (\xi_1, \xi_2, ..., \xi_n)$ state variables which have information on the evolution of the algorithm. The set has to have an element $\xi_1 = \theta^*$, in which it is the best solution to the encountered minimal problem.
7. τ: Φ x Ξ boolean parameter (true, false) function that determines the end of the algorithm (stopping criterion).

Operation of any meta-heuristic M could be defined formally as the following:

$$\Theta_{i+1} = \begin{cases} \theta^{*\mu} & if \quad \tau(\Theta_i \times \Xi) \rightarrow Result \\ \sigma(\Phi(\Theta_i) \times \Xi) & with \quad some \quad other \quad cases \end{cases} \quad (4)$$

3.3.1. Meta-heuristics Methods Features: The meta-heuristics methods possess certain fundamental characteristics, among which we highlight the following:

(a) These methods are templates or general strategies that what they do is to guide the search process, in order to efficiently carry out the exploration of the search space of the adopted problem to achieve an optimal result or pretty close to it.
(b) They are usually not deterministic algorithms, approximate, and cannot be absolutely certain that the solution they provide is the global optimum of the problem (which, if you get on the exact algorithms).
(c) Add restrictions to avoid exploring search spaces are not optimal.
(d) The meta-heuristic methods are not dependent problem for the which they are to be used, but its basic structure is general for all problem.
(e) Procedures algorithms can be easily transformed into parallel (execution of various parts of the algorithm at the same time, contrary to sequential), which allows a remarkable reduction of computational time.
(f) Meta-heuristics control a number of specific heuristics, by making the use of the characteris-tics of the problem for which they are designed, as the meta-heuristic is a high-level strategy, explores the search space.
(g) Since the meta-heuristic must be valid to address problems large search spaces, it is very important that there is a dynamic equilibrium between space exploration (search distant re-gions of space) and exploitation of specific areas (effort spent searching within the current region, to make a study of enough intense). This balance can quickly identify potential areas of global search space, while avoiding wasting time on studying regions already explored or those whose solutions are not of high quality.

Meta-heuristics algorithms expected to hold the following properties [9]:

(a) Easy understanding: as simplicity regarding the understanding of the method, it should be clear and simple, accuracy and concretion in the steps to create the algorithm, in order to allow easy implementation of same and consistency between the elements forming the meta-heuristic and principles thereof.

(b) High Performance: Effectiveness, efficiency and effectiveness. It must provides a high quality solutions and have a high probability of reaching them and maximizing the available resources (time computing and memory). To assess these properties must run a number of problems with known solutions, taking into account the optimal deviation as performance time.

(c) Comprehensive Implementation: General meta-heuristic utility, in which allows the algorithm to serve for many different problems, adaptability to different environments or variations of the apply model and robust enough for the result of the problem modifications to the model or environment to which it is applied.

(d) Usefulness of the Method: Interactivity that allows the user to go throw changing parts of the process to improve their performance, multiplicity in the provided solutions. It must provide a several solutions which close to optimum goal, in order that the best operation is chosen to prevent the user from this pending the program during execution.

3.3.2. Classification of Meta-heuristics Methods: There is no single way of classifying meta-heuristics techniques, since it depends on the property that we consider. Some of the classifications used today are linked to the following characteristics: if they are based on nature, if they have memory, if they have one or more neighbourhood structures, the nature of the objective function if each iteration is a single point of the search space it is or instead working with a population (set of ”points”). Thereby obtaining two distinct groups:

Meta-heuristics based on the trajectory. These techniques are, mostly, based on methods Local search with the addition of those trying to escape optimal premises to improve its outcome.

In such methods as a starting point we take the problem and each step of the algorithm same neighbours is explored, and goes updating the current point creating a path in space search problem. The search ends when the stopping condition is reached that can be: a maximum predefined number of iterations, reaching an acceptable quality, and detect a lack of progress. As given in the formal definition previously, this type of meta-heuristics characterized in that the element is 1, e.g. that it is working with a single structure, rather than using a population of one is the amount of structures (also called particles or individuals of population, depending on the method). A brief description of meta-heuristics classified within this set:

- **Simulated Annealing (SA):** This is one of the oldest meta-heuristics known and probably the first that tries to avoid local minima, allowing for it to choose solutions whose value of the objective function is worse than the others which provides current solution. Its operation is a simulation of the annealing process of crystals and metals, and what it does is to study the neighbourhood of the current particle and by certain criteria, choose one of their neighbours. Subsequently, if that neighbour provides a better fit than the current replaced directly, and if the fitness providing worse, the replacement is also performed but with a certain probability depending on the variation of fitness would cause and temperature (the probability is an analogy of the physical process of cooling, so the more iterations advance (approach the optimum), less temperature there, and therefore less likely to choose a worse fitness).
- **Iterated Local Search (ILS):** This is a very generic method in which first an initial solution to which is applied a local search method to improve, thus obtaining the final initial solution and then in each iteration a perturbation is performed on the solution is generated current and applies a local search method. To end the iteration, if the new candidate (already disturbed and improved) passes certain acceptance criteria (dependent on historical research, the new and current candidate) is replaced today by the new candidate. The most important in determining the performance of this algorithm process is the disturbance, since it must be neither so small as to not be able to leave a local optimum, nor so large as to cause us to fall into a local search method randomly reset.
- **Search with Variable Neighbourhood (SVN):** It is a very general algorithm and therefore flexible and adaptable to possible variations based on switch between different neighbourhoods throughout the search. First a set of neighbourhoods (k_{max} neighbourhoods) is defined and after each iteration three phases consisting randomly choose the candidate neighbourhood $n_k(s)$, where s is the current (starting with k = 1 are performed), improve the candidate selected by local search and finally if the result is better than the current movement is performed (replacing the current by the resulting candidate search and returning to the first phase with k = 1) for this new solution s, if the result is not better, no substitution is made and returns to the first phase, increasing k (until it reaches the value of k_{max}). Thus, we get the balance between intensity and diversity, as local search allows the exploitation of area while different neighbourhoods helps in exploration the search space.
- **Tabu Search (TS):** TS is one of the most successful meta-heuristics so far when solving combinatorial problems and is based on general principles of Artificial Intelligence as is the concept of memory. What it does is extract information from what happened and act accordingly, so we can say that it makes a smart search. This requires a tabu list (TL) is used short-term memory that allows us to avoid local optima, while preventing search in regions that have already been studied. That list recent moves are stored and discarded for possible future solutions. In each iteration the neighbourhood of the current position, N (s) is filtered, with excluded solutions (TL) obtaining a neighbourhood that satisfies the taboo condition, to which is added the solutions that meet the so-called aspiration criterion (the most common is to allow introducing solutions that have a better fitness so far, even if they belong to the TL). From this union we obtain the reduced $N_A(s)$ neighbourhood. Then choose the best fitness neighbour has within the $N_A(s)$, the taboo list is updated by adding this new movement and neighbouring selected by the current is replaced. In order to meet the importance of population-based meta-heuristics it will be discussed in the next section.

4. POPULATION-BASED META-HEURISTICS

The biggest difference between the different classified methods here lies in the way of manipulation of the population used. The following describes briefly the most common methods:

(a) Estimation of Distribution Algorithms (EDA): The first step, as in most of the algorithms is to create an initial population. Subsequently, in each iteration a set of individuals of the original population is selected, its probability distribution is studied and with it the new individuals who will form the new population is generated. Many authors classify this method as a variation of the evolutionary algorithms, or even as a type thereof, in which the reproductive stage is made in a special way (through the distribution of probability).

(b) Scatter Search (SS): This method is based on the principle that given two solutions; these can be combined into creating a new solution that enhances the two predecessors. The first step of the algorithm is to randomly generate the initial population, then with the most representative elements above a set reference set (consisting of solutions with high diversity and quality) is generated. Later iterations begin in several subsets of solutions S, from reference set are created, and solutions of these subsets are combined into creating new solutions. When we have created a new solution, they are improved by a local search algorithm chosen by the user and is updated reference set by replacing the part of the initial reference solutions set for some of the new solutions (only those that improve the initials). At the end of each iteration is checked whether reference set converges and if true, it rebuilds the entire population in a random form, the reference set is generated (in this case takes into account both the population created as the previous reference set) and continues iterating until the stop condition is reached. In this case the first thing it does is check whether converges reference set, if so, creates a new set of reference and otherwise generates subsets, combines and enhances individuals resulting from the previous recombination.

(c) Evolutionary Algorithms (EA): This type of evolutionary algorithms simulates the behaviour of beings in nature, which allows to adapt to different changes in their environment. A set of individuals, where each is a solution of the problem, and each iteration evolves to that population, thus obtaining a new and more adapted to the environment, which in our case means it has better fitness value is handled. First it is created either randomly or using a heuristic construction, an initial population and evaluated with the objective function (calculating therefore their fitness), then at each iteration three different phases are performed. The first is the selection of parents, where you create a new temporary population of size (parent population), from the original individuals, following the basis of natural selection (override those individuals who have better fitness). Second it is carried out reproduction during which operations are performed, with some probability, on the population of parents (couples or crossover recombination and then the mutation) thus obtaining a new population, the children side. To end the iteration the fitness of the population of children is evaluated and the replacement of the original population is made (There are two types which only takes into account the population of children, and also takes into account the old ones so that stay at the best and worst original individuals are replaced by the best sons). This is a general scheme based on which different evolutionary algorithms such as Evolutionary Programming (EP), Evolutionary Strategies (ES) and Genetic Algorithms (GA), which are the most known.

(d) Particle Swarm Optimization (PSO): This method simulates how swarms of bees looking for pollen in nature (or behaviour of flocks of birds and groups of fish). What they do is fly over an area looking for the area with the highest density of flowers, bearing in mind that always remember where they have displayed as much and also know what is the best site, found by the entire swarm. The bees continue to fly, looking for new best solutions, changing its direction depending on these two data. Meta-heuristic starts generating random, initial speed and position of each particle population form. The fitness of each individual, thereby knowing what the best solution the swarm is evaluated. On each iteration the velocity of considering the best own particle solution and the solution set is updated, the speed of particles moves, and finding the new position. To end the iteration, we evaluate the positions and update optimum values founded for each particle and the swarm.

(e) Ant Colony Optimization (ACO): The ACO algorithm is based on the natural behaviour of ants when looking for the shortest route between their food sources and the anthill way. Each ant starts searching at random (and at first no pheromone) and walking is deposited a trail of pheromone (chemical) that eventually evaporates, so the ant is on the shortest path it is the least time gives the pheromone to evaporate, so the way will have more trace than others, and as each ant chooses the next road to take depending on the amount of pheromone, each time they move closer to short cut. The meta-heuristic is based on a serial process, so the first step is to generate the initial population at random, then the pheromone matrix (formed by all the arcs possible paths) is updated and then start the iterations where it is updated the solution of each individual in the population considering a heuristic method of construction of trail pheromone. To end the iteration is calculated and fitness of the pheromone is updated (done both evaporations avoids falling into local optima, such as increasing the pheromone in the places that are a part of the best solutions for this iteration). Also, there is a probability to add to the method other operations that depend on the entire set of individuals, for example, by improving local search solutions.

5. CONCLUSION

Analysing the parts of optimization algorithms, we can conclude that those algorithms developed with the aim to optimize the already exsists results. Travelling Salesman Problem seems to work correctly with those optimization algorithms as discussed previously, getting very close to optimal value problems where the number of cities to be covered is small, even achieves the best result known so far. However, for large problems is far from optimum level. Hence, the using of hybrid optimization algorithms considers as a good solution reaching this level. Although, all optimization algorithms achieve near optimal solution to the TSP but these have drawbacks if implemented in its natural form (implement it with no problem), so they have to use specialized operators, as in the case of PSO, ACO and SA, which require specialized operators both to evaluate the algorithm functions in order to avoid generating infeasible routes (for solving TSP), and all used methods require that the movement takes feasibilities out does not occur, otherwise it is impossible to find the solution. However, it remains pending investigations future what would be the optimum value of these variables. It would advise also investigate the size of the population (in population base meta-heuristics algorithms) that optimizes the performance of the algorithm. As a future work our intentions is to provide a kind of optimization hybridization between two different kinds of optimization algorithms (e.g. Min-Max Ant Colony with Genetic Operations) in the aim to provide a new optimized solution for TSP problem.

6. ACKNOWLEDGEMENTS

This paper was made with the aim of preparing for the research problem in the Master's thesis. In the future it will have an extension in addressing one of the problems that has emerged in previous studies with the aim of improving and creating better outcomes.

7. REFERENCES

[1] Fan, H.: 'Discrete Particle Swarm Optimization for TSP based on Neighbourhood'. Journal of Computational Information Systems. 2010. pp. 3407-3413.

[2] Andrew, L.: 'Ising formulations of many NP problems',Department of Physics, Harvard Uni-versity, Cambridge, MA, USA 02138. 2014.
[3] Applegate, D. Bixby, R. Chavatal, V. y Cook, W.: 'The Travelling Salesman Problem: A Computational Study'. 2011. pp. 1-15.
[4] Lin, S.: 'Computer Solutions of the Travelling Salesman Problem'. The Bell System Technical Journal. 10(44). 1965. pp.2245-2269.
[5] Leticia, H. Jose, P, A. Alexander, M. Jose, L, A.:'A study on the complexity of TSP instances under the 2-exchange neighbour system'. Foundations of Computational Intelligence (FOCI), 2011 IEEE Symposium on. 2011. pp. 15-21.
[6] Yong, D. Yang, L. Deyun, Z.:'An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP'. Hindawi Publishing Corporation, Mathematical Problems in Engineer-ing. 2015. pp. 6.
[7] http://www.csd.uoc.gr/ hy583/papers/ch11.pdf, last access, Jun 2015.
[8] Abraham, P, P.: 'The Travelling Salesman Problem: Applications, Formulations and Varia-tions'. Elsiver, the series Combinatorial Optimization. 12. pp. 1-28.
[9] Karla, L, H. Manfred, P. Giovanni, R.:'The Travelling Salesman Problem'.Encyclopaedia of Operations Research and Management Science. 2016. pp. 1573-1578.
[10] Chiang, D. Potter, C.: 'Minimax non-redundant channel coding for vector quantization'. in Proceedings of the 1993 IEEE international conference on Acoustics, speech, and signal pro-cessing: image and multidimensional signal processing. USA. 1993. pp. 115-126.
[11] Gamboa, D. Rego, C. Glover, F.: 'Implementation analysis of efficient heuristic algorithms for the traveling salesman problem'. Computer Operational Research. 2006. 4(33). pp. 1025-1036.
[12] Blum, C. Roli, A.: 'Meta-heuristics in combinatorial optimization:Overview and conceptual comparison'. ACM Computer Studies. 2003. pp. 89-102.
[13] Alba, E. Luque, G.:'Parallel meta-heuristics:recent advances and new trends'. International Transactions in Operational Research.2013. 1(20). pp. 14-25.
[14] Advance Studies.: 'Relaxed tours and path ejections for the traveling salesman problem'. European Journal of Operational Research. 1998. 2(106). pp. 154-165.
[15] Pepper, W. Golden, B. Wasil, E.: 'Solving the traveling salesman problem with annealing-based heuristics: a computational study'. Trans. Sys. Man Cyber. 2002. 1(32). pp. 72-85.
[15] Ratnesh, K. Haomin, L.: 'On Asymmetric TSP: Transformation to Symmetric TSP and Per-formance Bound'. University of Kentucky.
[16] Burke, K. Kendall, G.: 'Search Methodologies'. Chapter 17: Hyper heuristics. Springer. 2005. pp.529-531.
[17] Hoffman, K.: 'Combinatorial optimization: Current successes and directions for the future'. Journal of Computational and Applied Mathematics. 2000. pp. 344-348.
[18] Muniswamy, V, V.: 'Design And Analysis Of Algorithms Book'. 2009.
[19] Kozma, L., and Momke, E. 'Maximum scatter TSP in doubling metrics'. In Proc. SODA 2017, pp 14-33.
[20] Aguayo, M., and Sherali, D. 'Single-commodity flow-based formulations and accelerated Benders algorithms for the high-multiplicity asymmetric traveling salesman problem and its extensions. Operation Research., 69(5), 2019. pp 736-742.

