Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences

Solving the Family Traveling Salesman Problem with Capacitated Agents

A thesis presented by

Kevin Alain Reyes Vega

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

In

Manufacturing Systems

Monterrey Nuevo León, June 5th, 2020

Dedication

To my parents… for their love, support and encouragement.

Acknowledgements

I would like to express my deepest gratitude to my parents for supporting me since my entrance into Tecnológico de Monterrey. My gratitude to all those who have been side by side with me during this project; in particular, to my sister and fiancée that have always supported me.

This would not have been possible without the support of Tecnológico de Monterrey and CONACyT. I want to thank both institutions for the scholarships that were granted to me. I would particularly like to thank Dr. José Luis González Velarde for advising me during the elaboration of this thesis. Thanks to Dra. Yasmín Águeda Ríos Solis, Dr. José Manuel Sánchez García and Eng. Saúl Domínguez Casasola, for their great support. I also want to thank Dr. Horacio Ahuett Garza for accepting me in the master´s program.

Lastly, I would like to recognize my friends support. Thanks for always be there.

Solving the Family Traveling Salesman Problem with Capacitated Agents

by
Kevin Alain Reyes Vega

Abstract

This thesis leads towards a new approach for the Family Traveling Salesman Problem (FTSP) using as an example a warehouse common problem. The enterprise owner of the warehouse wants to optimize the picking out total distance of the products, taking into account the available logistic resources.

This new extension of the standard FTSP is denoted as Family Traveling Salesman Problem with Capacitated Agents (FTSP-CA). The formulation of the problem is a single objective model, with binary variables. For the computational experimentation two methodologies were applied: Integer programming and Heuristics.

From a set of 21 benchmark FTSP instances, a new group of 36 adapted instances were created that consider the FTSP-CA parameters. For the integer programming technique, the Cplex solver is used to obtain optimal integer solutions. For the second methodology, a Biased Random-Key Genetic Algorithm (BRKGA) was implemented to improve the performance in time and while maintaining a high-quality value of solutions. Both techniques are compared to show the efficiency solving the FTSP-CA.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The Family Traveling Salesman Problem (FTSP), is an extension of the classic Traveling Salesman Problem (TSP). Both are well known due to their logistics applications to minimize the cost of travelling, providing better solutions for enterprises and governments. TSP is commonly used to optimize vehicle routes inside the city to deliver goods and products from distribution centers to retailers and costumers [6]. On the other hand, the FTSP is tailored to the flexibility of modern warehouses systems. In a *free-pick-and-drop* mechanism [19] the storekeeper can freely move inventory items using a RFID tracking system to locate each specific product in a virtual space. Consequently, different family types of items are placed randomly in the warehouse.

The FTSP was introduced by Morán-Mirabal et al. [13], considering a modern warehouse configuration. The problem consist in a warehouse picker, that must visit specific locations of the facility in order to pick-up different products. As mentioned before, the products belongs to an specific family, and are placed randomly within the warehouse. Consequently, a solution of the FTSP consist in a closed tour where the warehouse picker passes through each checkpoint once, starting and ending the tour in the same point.

As an extension of the FTSP, the Family Traveling Salesman Problem with Capacitated Agents (FTSP-CA) simulate the logistics operations in a warehouse where multiple agents (warehouse pickers) with finite capacity of products pick-ups are available. Taking into consideration that companies warehouses rely on a human workforce doing operations simultaneously. As a result, different warehouse pickers are going to work in parallel, to fulfill the demand of checkpoints required to visit.

The FTSP-CA is motivated by the fact that there is a necessity for enterprises to align the logistics inside a warehouse, to reduce the total cost in distance, time or money, generated by the daily operations of the warehouse pickers, providing an optimal integer solution for this scenario.

1.1 Problem Statement and Context

In the Supply Chain Management of a company, many different products are sorted randomly in a warehouse using a Radio Frecuency Identification (RFID) system. An inventory system classifies each product into a family number, according to their size, weight, color, etc. Due to the RFID system, similar family products are distributed randomly inside the warehouse.

A list of shipping products must be taken out by multiple agents (warehouse pickers). The product list is made up by a determined number of visits to the different families. In order to fulfill the order, multiple agents (warehouse pickers) must visit the complete family visits, starting the routing process and ending in the same point, which is called “depot”. However, agents inside the warehouse can only visit a node (product location) once, and must be limited to a maximum capacity of nodes. As mentionned this problem is the FTSP-CA.

1.2 Motivation

Logistics has been one of the most important areas for business competitive advantage to the possibility to speed up inbound companies processes and deliver goods to the costumers in the minimum time possible. Nowadays, enterprises demand intelligent solutions to improve their Supply Chain Management, and Inventory Control and Transportation Routing are very important to achieve that goal.

There are many studies about RFID Technology inside warehouse, that demonstrate the benefits to allocate products using this alternative [7] [8] [18]. Indeed, RFID is becoming a promising solution to avoid inventory inaccuracy. For that reason, the actual global tendency is to adopt this newer technology demanded by the retailers to provide a better response time between the different stages of the Supply Chain.

However, this new warehouse configuration demands complex alternatives to the standard traveling salesman problem to achieve a routing solution.

1.3 Contribution

Adopting RFID technology as an alternative to inventory system provides a justification to create a new routing control system inside the warehouses which adapts to the family sorting of products.

The main contribution of this thesis is to present a new problem as an extension of the standard family traveling salesman problem. This extension is called Family Traveling Salesman Problem with Capacitated Agents (FTSP-CA). Furthermore, the strategy to solve this NP- hard problem is presented, developing the heuristics to obtain an approximate value of the optimal integer solution, using a Biased- Random Key Genetic Algorithm (BRKGA).

1.4 Thesis Structure

The first chapter presents the main introduction of the problem and its context. Moreover, the motivation and contribution of the scientific research is described.

The second chapter presents a literature review, with the state-of-the-art and theoretical framework about the classic Traveling Salesman Problem and two important configurations: Generalized Traveling Salesman Problem (GTSP) and Family Traveling Salesman Problem (FTSP). At the end of the chapter, a review in Heuristics, specifically in Biased-Random Key Genetic Algorithms is shown.

The mathematical framework is presented in chapter three. First the paramaeters, decision variables and auxiliary decision variables are described, followed by the mathematical model.

In chapter four, two important methodologies are described which are the methodologies used in this thesis: Integer Programming and Heuristics.

The fifth chapter presents the complete computational experimentation. On the one hand, the computational characteristics are listed. On the other, the description of the different instances is presented and, finally, the final results are analyzed and discussed.

Finally, the sixth chapter is dedicated to the conclusions and future work of this thesis.

Chapter 2

Literature Review

In this chapter, a brief description of the concepts and algorithms used throughout the thesis is given. The first part describes the concept of Traveling Salesman Problem (TSP), and some of the current variations are presented. Second section is dedicated to the Family Traveling Salesman Problem (FTSP), the specific variation of the problem on which the thesis is focus on. Finally, a background of Biased- Random Key Genetic Algorithms (BRKGA) is presented.

2.1 Traveling Salesman Problem

The Traveling Salesman Problem is well known for being part of NP-hard problems (Karp, 1972). In terms of theoretical computational, this NP-hard computational class is a set of problems with high complexity for which no algorithm is capable to solve all instances in polynomial time. The TSP consists in a "salesman" person who is looking to travel by the most proficient sequence of cities or nodes within its territory, bearing in mind that the salesman can only visit a city once, and must finish the tour in the starting location. (Sathya, 2015).

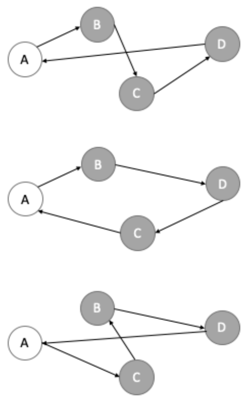


Fig.1 Representation of three different feasible solutions for the TSP.

The visual representation of the Traveling Salesman Problem consists in a graph, with a set of nodes. The arrow that connects two nodes shows the cost vector (time, distance, or other attribute), produced by the traveling movement.

Since 1934, when the problem was proposed by Hassler Whitney at Princeton University, the notation of the Traveling Salesman Problem has been evolving (Flood,1956).

Table 1. TSP Parameters

Parameter	Description
n	The number of nodes in the area
i, j	Indexes of variables relating nodes i *and* j
$x_{i,j}$	Binary variable equal to 1 when salesman goes from node i directly to node j and 0 otherwise.
$d_{i,j}$	Cost related to the distance traversed from node i to node j

2.2 Generalized Traveling Salesman Problem (GTSP)

The Traveling Salesman Problem has different variants. One specific configuration is the Generalized Traveling Salesman Problem, arising from the "clustering" of nodes, where the salesman or agent must visit only one node per cluster. Similar to the standard traveling salesman problem, the objective is to minimize the distance traveled, starting and finishing the tour in a determined node, and creating a Closed Hamiltonian Cycle.

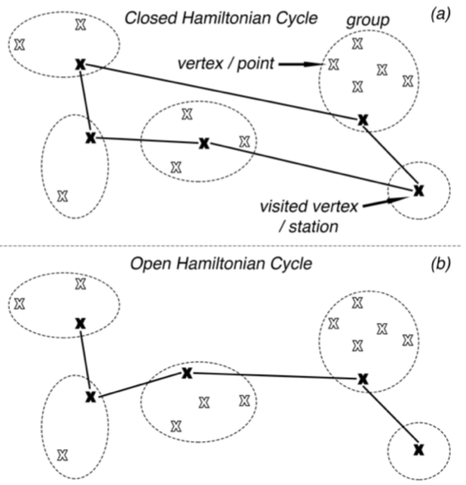
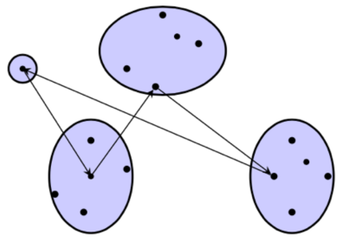


Fig.2 Representation of a Closed Hamiltonian Cycle (Zia, 2018).

The Generalized Traveling Salesman Problem has been introduced by Srivastava et al. (Srivastava, 1970). On the basis that standard traveling salesman problem is a NP-hard problem, the GTSP is still part of this class of problems. If all the clusters presented by a GTSP instance consist of a single node, a standard NP-Hard configuration is shown.

New approaches in the topic has been introduced by Yuan et al. (Yuan, 2020), using a branch-and-cut algorithm; Yang et al (Yang, 2008), proposed a genetic algorithm. Moreover, some alternatives to transform a GTSP to TSP have been applied by Zia et al. (Zia, 2018) without producing an extensive computational time solution. Importance of the Generalized Traveling Salesman Problem lies in its variety of different applications in logistics (Baniasadi, 2020) (Laporte, 1987).

According to Pintea et al. (Pintea, 2007) formulation, the Generalized Traveling Salesman Problem, can be interpreted by a graph $G = (N , E)$, where N is partitioned into p clusters, and the cost of an edge $e = \{i , j\}$ is denoted by $c_{i,j}$. The goal is to minimize the total cost of the Hamiltonian tour, where there is only one node per cluster visited.



(a) An example tour of GTSP

Fig.3 Example Tour of a GTSP (Pintea, 2007).

2.3 Family Traveling Salesman Problem (FTSP)

The FTSP is another configuration of the standard Traveling Salesman Problem (TSP), it was introduced by Morán-Mirabal et al. (Morán-Mirabal, 2014) as an application of routing solutions inside modern warehouse using RFDI technology. In this particular application the complete set of nodes are segmented into different families and a single agent must travel a determined number of family visits. Different from the GTSP, in the FTSP the agent can visit more than one node per cluster, in this case, per family.

The FTSP is modeled using a complete graph $G = (\mathcal{N} \cup \{0\}, E)$. In this formulation, $\{0\}$ is the depot (where all the routes start and end). $\mathcal{N}$ represents the set of nodes or cities, which displays a specific warehouse location, where an inventory product has been placed. The distance cost between a pair of nodes is denoted by $d_{ij} > 0$, , where $i \neq j$. All distances are indexed in the edge-set E .

For solving the FTSP, a prescribed number of nodes nf_l must be visited in order to pursue the total minimum distance tour. The nodes are selected from a set F_l, where l is the index of L families. Consequently, the total amount of visits V, is given by $V = \sum_{l=1 \to L} nf_l$. The warehouse is conditioned to a total of Kagents. Each agent has a finite capacity Q measured by the maximum number of nodes that an agent can visit.

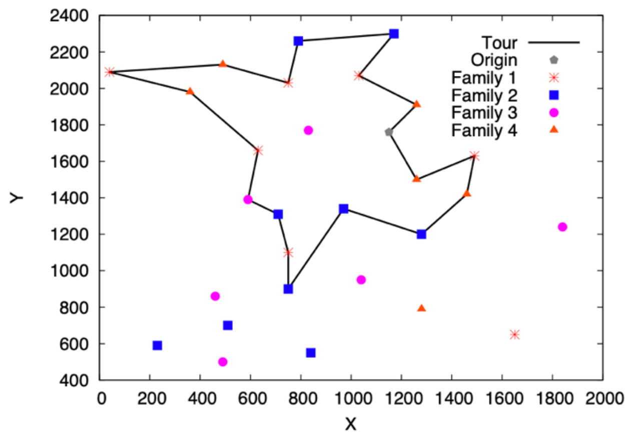


Figure 4. Example of FTSP Solution (Morán-Mirabal,2014).

Due to the importance of the Family Traveling Salesman Problem, different algorithm alternatives have been proposed by Bernardino and Paias such as: The neighborhood search procedure, the local search procedure, the perturbation method and the ILS algorithm (Bernardino and Paías, 2018).

2.4 Biased Random-Key Genetic Algorithms (BRKGA)

Genetic algorithms were introduced since the 60s to find feasible solutions to complex problem using a replication of the principles from Darwin´s natural evolution. A genetic algorithm starts with a constructed initial population of solutions. Similar to nature, the solutions pass on their codification to the next generation, using reproduction and crossover operations. These is described as the survival-of the-fittest, where the population became stronger over generations. The main idea is to reach a generation with a solution near the optimal.

BRKGA is an extension of the traditional RKGA proposed by Bean (Bean, 1994). As mentioned before, each individual of the total population has a *random key*, defined as "a string, or vector, of randomly generated real numbers in the interval [0,1]" (Gonçalves, 2011). The random-key codifies a solution of the optimization problem. However, in order to visualize the result and fitness of the codified individual, a *decoder* must be needed. The decoder is a deterministic algorithm that converts a codified individual with random key, to an objective value of the optimization problem.

The BRKGA assigns n_p vectors of random keys to an initial population P_0, each of size n_c. Each allele λ_i with $i \in 1, \dots, n_p$ (vector of random keys), is randomly generated in the real interval (0,1]. The alleles are translated into feasible solutions using a decoder. Fitness value of the feasible solution for the optimization problem is defined by $f(\lambda_i)$.

More recently, J.F. Gonçalves, M.G.C. Resende, and R.F. Toso (Gonçalves, 2012) present a differentiation between RKGA and BRKGA. The main difference between RKGA and BRKGA is the matting process. In RKGA both parents are selected at random from the entire population. In Biased-random algorithm the population is divided into elite and non-elite populations, each individual is generated by matting one element from the elite current population (a) and another from the non-elite class (b). To determine which parent will transfer the key to the next generation a biased coin toss is executed for n_c times. For each coin toss $j \in 1, \dots, n_c$ the offspring has a probability $p_e > .5$ to inherit the j-th key from a (vector of elite individuals), and a probability $p_{\bar{e}} = 1 - p_e$ to inherit from b .

Consequently, Biased Random-Key Algorithms have been used for multiple applications such as flowshop problems (Andrade et al., 2019), telecommunications and logistics (Resende, 2011). To provide more accurate solutions than the RKGA, in a shorter time.

Chapter 3

3.1 Mathematical Model

In the following chapter, a description of a mathematical integer linear programming is present for the FTPS-CA. First a table with the parameters, decision variables, and auxiliary decision variables is presented, followed by the complete mathematical framework. At the end of the chapter, some specific considerations are justified related with the configuration of the problem.

Table 2. Parameters, Decision Variables and Auxiliar Decision Variables

Parameters:	Description	
N	Set of nodes	
L	Set of families	
K	Set available agents	
V	Total number of visits	
Q	Agents capacity	
F_l	Nodes in the family l,	$l = 1,\ldots, L$
d_{ij}	Distance from node i to node j,	$i, j = 0, \ldots, N$
nf_l	Number of nodes of family l to visit,	$l = 1, \ldots,$ L
Decision Variables:	Description	
x_{ijk}	The variable is = 1, if the node i precedes node j, and its traversed by agent k,	$i, j,= 0, 1, \ldots, N$, $k = 1, \ldots.,$ K
Auxiliar Decision Variables:	Description	
$u_{ik} \geq 0$	Node i potential,	$i = 1, \ldots, k = 1, \ldots, K$

The parameters, decision variable and auxiliary decision variable for the mathematical model are shown in Table 2 . For the next part, the mathematical formulation of the problem is presented with the respective description.

$$\min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=0}^{K} d_{ij} x_{ijk} \tag{3.1}$$

Subject to:

$$\sum_{j=1}^{N} x_{0jk} = 1, \qquad k = 1,\ldots,K \tag{3.2}$$

$$\sum_{i=1}^{N} x_{i0k} = 1, \qquad k = 1,\ldots,K \tag{3.3}$$

$$\sum_{i=1}^{N} x_{i0k} = 1, \qquad k = 1,\ldots,K \tag{3.4}$$

$$\sum_{i=0}^{N} \sum_{k=1}^{K} x_{ijk} \le 1 \qquad j = 1,\ldots,N \tag{3.5}$$

$$\sum_{j=0}^{N} \sum_{k=1}^{K} x_{ijk} \le 1 \qquad i = 1,\ldots,N \tag{3.6}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} x_{ijk} = KN + K \tag{3.7}$$

$$\sum_{i=0}^{N} \sum_{j=1}^{N} x_{ijk} \le Q \qquad k = 1,\ldots,K \tag{3.8}$$

$$\sum_{i=0}^{N} \sum_{j \in F_l} \sum_{k=1}^{K} x_{ijk} = nf_l \qquad l = 1,\ldots,L \tag{3.9}$$

$$\sum_{i \in F_l} \sum_{j=0}^{N} \sum_{k=1}^{K} x_{ijk} = nf_l \qquad l = 1,\ldots,L \tag{3.10}$$

$$\sum_{i=0}^{N} x_{ijk} - \sum_{i=0}^{N} x_{jik} = 0, \qquad j = 0,\ldots,N, k = 1,\ldots,K \tag{3.11}$$

$$u_{ik} - u_{jk} + Qx_{ijk} \le Q - 1 \qquad i, j = 1,\ldots,N, i \ne j, k = 1,\ldots,K \tag{3.12}$$

The problem is formulated using a binary integer program (BIP), where the binary variable $x_{i,j,k} = 1$, when the arc from node $i \rightarrow j$ is traversed by the agent k, and $x_{i,j,k} = 0$ otherwise. The objective function (3.1) minimizes the total distance traveled for the selected tours of the agents. The following constraints (3.2) and (3.3) regulate that all the routes of the multiple agents start and end in the depot $\{0\}$, and all the agents must leave the origin. Constraints (3.4) and (3.5) specify that at most one arc must enter and leave each node, respectively. The total number of visits is formulated by constraints (3.6). The next constraints (3.7), establish a finite capacity for the K agents. In addition, constraints (3.8) and (3.9) establish the number of arcs to enter and leave each of the families. Constraints (3.10) regulates the flow conservation, while the sub-tour elimination strategy is denoted by constraints (3.11).

For the final consideration of the problem, the mathematical formulation assumes that all the agents must leave the deposit or starting point ((Constraints (3.2) and (3.3)).

Chapter 4

Resolution Methodologies

4.1 Integer Programming Method

The first methodology consists of an Integer Programming strategy using a formulation in AMPL software with CPLEX v.12.9 as a programming solver. The Family Traveling Salesman Problem with Capacitated Agents is an integer problem. It is well known, that most of scheduling, touring, routing, and sequencing problems are part the pure integer programming problems. Thus, the complete mathematical framework was translated into AMPL programming language to build the BIP of the FTSP-CA.

AMPL provides an optimization modeling lifecycle that enables the correct formulation of the FTSP-CA, and also tests the different benchmark instances with CPLEX as a solver. For version 12.9, CPLEX uses a branch-and-bound search to find a feasible best solution for the instance. The branch-and-bound was initially proposed by Land and Doig (Land and Doig, 1960), as an alternative to obtain optimal integer solutions to complex problems which cannot be solved in polynomial time.

According to Tomazella et al., the branch-and-bound method consists "of and implicit enumeration of the solution by creating partial sequences job per job and creating a tree that branches into complete solutions" (Tomazella, 2020). Due to the FTSP-CA complexity as a NP-hard problem, the Integer Programming technique will be applied in the computational experiments in order to find the optimal solutions or at least feasible solutions for the benchmark instances.

4.2 BRKGA Method

The second methodology consists of a Biased Random Key Genetic Algorithm (BRKGA) implementation. As mentioned in Chapter 2, BRKGA is an algorithm designed in the "survival of the fittest" in order to achieve complete optimal or near optimal solutions for complex optimization problems. A complete description of the algorithm is proposed by Bean (Bean,1994). In Figure 5, BRKGA is presented.

```
Data: $n_p, n_c, n_e, n_m, p_e$
Generate $\mathcal{P}_0$ with $n_p$ individuals having $n_c$ random-keys $\in (0,1]$;
$i = 0$;
while stopping criterion is not satisfied do
    Evaluate fitness of each new individual in $\mathcal{P}_i$;
    Partition $\mathcal{P}_i$ into $\mathcal{P}_i^e$ and $\mathcal{P}_i^{\bar{e}}$;
    Initialize next population: $\mathcal{P}_{i+1} \leftarrow \mathcal{P}_i^e$;
    Generate $n_m$ mutants $\mathcal{P}^m$ each having $n_c$ random-keys $\in (0,1]$;
    $\mathcal{P}_{i+1} \leftarrow \mathcal{P}_{i+1} \cup \mathcal{P}^m$;
    for $k \leftarrow 1$ to $n_p - n_e - n_m$ do
        Select parent $a$ at random from $\mathcal{P}_i^e$;
        Select parent $b$ at random from $\mathcal{P}_i^{\bar{e}}$;
        for $j \leftarrow 1$ to $n_c$ do
            Toss biased coin having probability $p_e > 0.5$ of heads;
            if Toss results in heads then $c[j] \leftarrow a[j]$;
            else $c[j] \leftarrow b[j]$;
        end
        $\mathcal{P}_{i+1} \leftarrow \mathcal{P}_{i+1} \cup \{c\}$;
    end
    $i++$;
end
return $\lambda^* \leftarrow \mathbf{argmin}\{f(\lambda_i) \mid \lambda_i \in \mathcal{P}_g\}$
```

Figure 5. BRKGA´s algorithm (Morán-Mirabal,2014).

To evolve the current population to the i-th generation of individuals, P_0 is partitioned in two types of individuals: elite individuals (P_i^e) and non-elite individuals ($P_i^{\bar{e}}$). Elite individuals are composed by n_e individuals, which are the evaluated to fit the optimization problem. Non-elite individuals are the remaining ones, composed by $n - n_e$, where $n_e < n_p - n_e$. After this segmentation process, all elite individuals in P_i^e are copied into the population of the following generation (P_{i+1}). Secondly, a mutant vector of random keys is inserted to the population P_{i+1} . A mutant is a set of P^m of n_m individuals, with the restriction that $2\,x\ \ n_e < n_p$ and $n_e + n_m \leq n_p$. Last group of individuals are found by matting n_o pairs of individuals from P_i where $n_o = n_p - n_e - n_m$. The crossover selects one individual from the elite group and another from the non-elite group. The matting process is randomly selected with replacement. However, different form traditional RKGA, in BRKGA, to determine which parent will transfer the key to the next generation a biased coin toss is executed for nc times. For each coin toss j∈1,…,nc the offspring has a probability pe $>$ .5 to inherit the j-th key from a (vector of elite individuals), and a probability $p\bar{e}$=1− pe to inherit from b .

The BRKGA is stopped until the acceptance criterion is reached. The total number of generations produced by the metaheuristic is denoted by g. Pg is returned as the final individual with the best fitness for the optimization problem.

From the BRKGA´s method description, a list of parameters are explained in Table 3. These parameters are included in Toso and Resende´s Application Programming Interface (Toso and Resende, 2012). They provided this API with the following set of necessary parameters.

Table 3. BRKGA´s Parameters Description

Parameter	Description
n	Total number of alleles per chromosome
p	Number of chromosomes in population
p_e	Size of elite set in population
p_m	Number of mutants to be introduced in population at each generation
p_e	Probability that an allele is inherited from the elite parent

For the FTS-CA experimental computation, the complete population was composed by 1000 chromosomes, *p= 1000* .The size of the elite population, $p_e = .20$. For each new generation a percentage of $p_m = .10$, of mutant individuals is added to to the population P_{i+1}. Finally, the probability to inherit from the elite parent in the crossover matting, is denoted by $p_e = .70$.

When the algorithm achieves the stopping condition, Pg is returned as the final individual with the best fitness for the optimization problem. To interpret this solution a *decoder* is used to retrieve the best feasible solution.

The feasible solution in the FTSP-CA is encoded as an *|N| + 2(KN)* vector λ of random keys, where $KN = {}_{j\in 1,\ldots,K}\, nv\, j$. The complete random vector key is divided in three segments for a better comprehension. The first part, |N|, produces a random number per each node in the FTSP-CA instance, this segment of the random key will be later use to determine which subset of nodes must be visited per family. For the second and third segments of the random key, each one of them haas a length of *KN* random keys. This parts of the random key are used to define the corresponding agent and the tour to be followed respectively.

In order to interpret the solution the decoder can be understood in three different stages according to the initial segmentation of the complete random key. Figure 6. represents the correct segmentation per decoding stages. In this example, a FTSPCA instance with the following parameters is considered: *N= 20 , L=4 , K=3, Q=4, F(1)=5, F(2)=4, F(3)=6, F(4)= 4, V(1)=3, V(2)=2, V(3)=4, V(4)=3.*

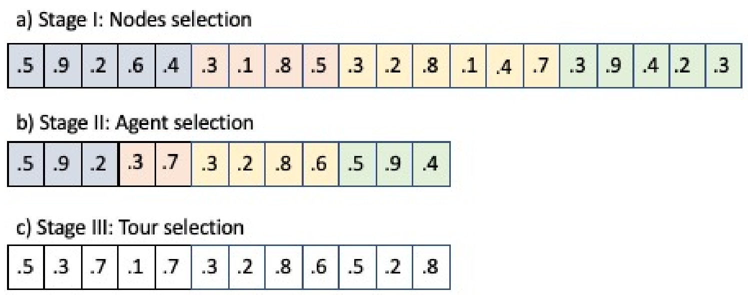


Figure 6. Decoding Stages of a FTSP-CA Random Key.

In order to transalate Pg into a feasible solution to the FTSP-CA, the decoding process stages are described, the nomenclature is based on the FTSP-BRKGA proposed by Morán-Mirabal et al. (Morán-Mirabal, 2014):

1. Stage I - Nodes Selection: The first segment of N random keys are used to determine the subset of nodes will be visited per family. The segment is divided into L sets of families Ri, $i = 1, 2, \ldots, L$, each of size nf_i. For each family random key, the nf_i keys are sorted in increasing order. Finally, select the nv_j smallest key indices where $j = 1, 2, \ldots, L$.

2. Stage II – Agent Selection: The first KN segment represents the set of family nodes obtained in stage I . The selected nodes are sorted in increasing order, where the first random key of this segment corresponds to the family 1 node with the lowest random key from stage 1. In order to match the random key with the agent, an interval segmentation of length $I = \frac{K}{KN}$ is stablished within [0,1] and a total of K segments. Finally, pair the random key value with the corresponding agent according its interval number.

3. Stage III – Tour Selection: The remaining segment of KN random keys are sorted in increasing order. The indices are used to define the sequence in which the nodes from stage II will be visited.

Chapter 5

Experimental Results

The experimental process was conducted by both methodologies: Integer programming and a biased random-key genetic algorithm. A group of 21 instances were used to test the performance of both models. Instances are divided into 7 different blocks depending on the number of nodes contained. The total number of nodes vary between 13 to 1001. Furthermore, the instances were taken from “Randomized Heuristics for the Family Traveling Salesman Problem” (Morán-Mirabal, 2014).

Original instances were adapted in terms of Family Traveling Salesman Problem with Capacitated Agents (FTSP-CA), adding two new blocks of parameters: K (agents) and Q (capacity). For each block of instances, there are two scenarios considered: Fixed-Agents and Fixed-Capacity. The fixed value varies from 2-18.

All the experimental tests were conducted using a laptop with a 2.4 Gigahertz Intel 8th Generation Core i5 with 8GB of RAM. For integer programming method AMPL software using CPLEX version 12.9 was used, and X-code version 10.2.1 for the BRKGA algorithm.

5.1 Instances Description

The complete list of benchmark instances for FTSP are described in Table 4. Instances are in the standard form for family traveling salesman problem. The table consist in a group of 21 instances. They are divided according to the total number of nodes corresponding to the problem in blocks of three instances. There are 7 major blocks of instances: Burma14, Bayg29, Att48, A280, Gr666 and Pr1002. The parameters for each instance are: Total number of arcs denoted as $|N|+1$, total number of families L, total number of visits V, the number of nodes per family F_l, and the number of nodes to visit per each family l, denoted as nf_l.

Table 4: Benchmark FTSP Instances [Morán-Mirabal, 2014]

Instance Name	$\lvert N\rvert+1$	L	V	F_l	nf_l
Burma14_3_1001_1001_2	14	3	6	[4, 5, 5]	[2, 2, 2]
Burma14_3_1001_1002_2			10		[4, 2, 4]
Burma14_3_1001_1003_2			4		[2, 1, 1]
Bayg29_4_1001_1001_2	29	4	16	[7, 9, 6, 6]	[6, 4, 5, 1]
Bayg29_4_1001_1002_2			17		[2, 9, 1, 5]
Bayg29_4_1001_1003_2			18		[6, 6, 1, 5]
Att48_5_1001_1001_2	48	5	34	[12, 9, 9, 7, 10]	[10, 4, 9, 7, 4]
Att48_5_1001_1002_2			25		[8, 2, 9, 1, 5]
Att48_5_1001_1003_2			15		[6, 1, 3, 3, 2]
Bier127_10_1001_1001_2	127	10	62		[10, 4, 13, 1, 12, 4, 6, 1, 5, 6]

Bier127_10_1001_1002_2			85	[12, 12, 14, 8, 13, 16, 13, 8, 17, 13]	[8, 2, 12, 7, 9, 9, 5, 5, 17, 11]
Bier127_10_1001_1003_2			60		[6, 1, 13, 3, 3, 13, 13, 2, 2, 4]
A280_20_1001_1001_2	280	20	179		[14, 10, 14, 4, 13, 9, 15, 4, 5, 14, 6, 7, 6, 8, 13, 7, 9, 13, 6, 2]
A280_20_1001_1002_2			156	15, 14, 16, 11, 19, 15, 18, 10, 17, 16, 16, 8, 7, 15, 24, 8, 11, 13, 15, 11]	[8, 2, 12, 9, 9, 5, 17, 6, 3, 9, 7, 2, 6, 11, 4, 6, 11, 7, 11, 11]
A280_20_1001_1003_2			141		[14, 14, 6, 1, 13, 3, 18, 3, 2, 4, 10, 4, 4, 8, 5, 4, 9, 4, 14, 1]
Gr666_30_1001_1001_2	666	30	357		[14, 10, 15, 4, 13, 9, 15, 4, 22, 5, 14, 6, 15, 30, 24, 7, 2, 1, 19, 5, 6, 13, 18, 9, 21, 10, 15, 2, 10, 19]
Gr666_30_1001_1002_2			328	[27, 24, 24, 17, 29, 19, 20, 17, 27, 24, 26, 15, 15, 30, 40, 11, 19, 28, 27, 20, 28, 22, 24, 14, 23, 15, 17, 18, 20, 25]	[8, 2, 15, 9, 21, 17, 14, 3, 9, 7, 10, 6, 11, 4, 39, 11, 11, 26, 7, 8, 1, 8, 14, 7, 19, 5, 6, 9, 9, 12]
Gr666_30_1001_1003_2			328		[6, 17, 13, 2, 4, 12, 4, 5, 12, 14, 15, 9, 4, 14, 33, 10, 17, 27, 17, 8, 6, 8, 2, 5, 8, 9, 17, 15, 6, 9]
Pr1002_40_1001_1001_2	1002	40	486		[14, 10, 15, 4, 13, 9, 15, 4, 22, 25, 5, 14, 6, 30, 24, 14, 13, 7, 25, 22, 2, 1, 19, 5, 6, 13, 18, 9, 15, 2, 22, 10, 19, 11, 1, 8, 3, 8, 6, 17]
Pr1002_40_1001_1002_2			538	[22, 28, 27, 30, 32, 24, 21, 22, 29, 30, 27, 16, 20, 30, 38, 16, 21, 23, 27, 28, 23, 25, 26, 26, 21, 24, 20, 30, 18, 25, 25, 27, 27, 21, 26, 24, 28, 28, 25, 21]	[8, 2, 15, 25, 9, 21, 17, 14, 22, 22, 3, 9, 7, 10, 6, 11, 4, 22, 27, 7, 11, 7, 8, 1, 8, 14, 19, 21, 6, 9, 9, 12, 26, 8, 23, 21, 8, 28, 18, 20]
Pr1002_40_1001_1003_2			463		[6, 17, 13, 19, 19, 18, 19, 2, 4, 26, 12, 4, 5, 12, 15, 9, 4, 14, 1, 15, 17, 17, 8, 6, 8, 2, 5, 8, 17, 15, 6, 9, 3, 20, 15, 5, 14, 26, 18, 10]

Benchmark instances were adapted to the form of FTSP-CA. There is a complete set of 36 new instances described in Table 5. Furthermore, each new instance contains a specific number of agents *K,* with a maximum capacity *Q*. All of the instances were placed in two different scenarios. Scenario A consist in a fixed value for the capacity and this value is higher than the number of available agents. On the other hand, scenario B alternates the value of capacity-agents. All the instances in scenario B have a higher number of agents than capacity. Fixed values vary from 2 – 18.

Table 5. List of FTSP-CA Adapted Instances

Instance Name	$\|N\| + 1$	L	V	A	Q	F_l
Burma1001A	14	3	6	2	3	[4, 5, 5]
Burma1001B			6	3	2	
Burma1002A			10	2	5	
Burma1002B			10	5	2	
Burma1003AB			5	2	2	
Bayg1001AB	29	4	16	4	4	[7, 9, 6, 6]
Bayg1002A			17	4	5	
Bayg1002B			17	5	4	
Bayg1003A			17	4	5	
Bayg1003B			18	5	4	
Att1001A	48	5	34	5	7	[12, 9, 9, 7, 10]
Att1001B			34	7	5	
Att1002AB			25	5	5	
Att1003A			15	3	5	
Att1003B			15	5	3	
Bier1001AB	127	10	62	8	8	[12, 12, 14, 8, 13, 16, 13, 8, 17, 13]
Bier1002A			85	8	11	
Bier 1002B			85	11	8	
Bier1003AB			60	8	8	
A1001A	280	20	179	12	15	15, 14, 16, 11, 19, 15, 18, 10, 17, 16, 16, 8, 7, 15, 24, 8, 11, 13, 15, 11]
A1001B			179	15	12	
A1002A			156	12	13	
A1002B			156	13	12	
A1002AB			141	12	12	
Gr1001A	666	30	357	16	23	[27, 24, 24, 17, 29, 19, 20, 17, 27, 24, 26, 15, 15, 30, 40, 11, 19, 28, 27, 20, 28, 22, 24, 14, 23, 15, 17, 18, 20, 25
Gr1001B			357	23	16	
Gr1002A			328	16	21	
Gr1002B			328	21	16	
Gr1003A			328	16	21	
Gr1003B			328	21	16	
Pr1001A	1002	40	486	18	27	[22, 28, 27, 30, 32, 24, 21, 22, 29, 30, 27, 16, 20, 30, 38, 16, 21, 23, 27, 28, 23, 25, 26, 26, 21, 24, 20, 30, 18, 25, 25, 27, 27, 21, 26, 24, 28, 28, 25, 21]
Pr1001B			486	27	18	
Pr1002A			538	18	30	
Pr1002B			538	30	18	
Pr1003A			463	18	26	
Pr1003B			463	26	18	

5.2 CPLEX results

The first part of the experimental analysis includes the results of the adapted FTSP-CA instances using AMPL system with CPLEX 12.9 solver. This integer programming solver is capable to evaluate the FTSP-CA instances directly.

As mentioned in the description of the adapted instances, the classification of the instances blocks is according to the total number of nodes. Instances from the first block, Burma1001-Burma1003, have 13 nodes. Second block (Bayg1001-Bayg1003) contains 28 nodes per instance. Instances Att1001-Att1003 have 47 nodes. Finally, the fourth block (Bier1001-Bier1003) includes 126 nodes per instance.

Table 6, shows the results of running CPLEX 12.9 on 19 FTSP-CA instances. The table contains the instance name, the total distance is presented as the final CPLEX solution. Furthermore, an explanation of the result condition is expressed. For Optimal Integer Solutions with MIP-gap, a relative MIP-Gap is presented. The relative MIP-Gap is presented which is the gap of the difference between the current upper and lower bounds on the optimal cost in the branch-and-bound procedure. When the relative MIP-Gap is zero, the result is an optimal integer solution (Angalakudati, 2014).

Table 6. List of CPLEX results

Instance Name	CPLEX Solution	Result Condition	Relative MIP-Gap	Time
Burma1001A	15.2479	Optimal Integer Solution	0.00	0.612863
Burma1001B	18.2279	Optimal Integer Solution	0.00	0.448693
Burma1002A	32.5140	I.S with MIP-gap	8.0571e-05	4.03052
Burma1002B	48.9479	O.I.S with MIP-gap	1.45163e-16	3.01468
Burma1003AB	13.630000	Optimal Integer Solution	0.00	1.00
Bayg1001AB	8304.8700	I.S with MIP-gap	9.99414e-05	887.247
Bayg1002A	8311.321494	I.S with MIP-gap	9.47143e-05	1419.53
Bayg1002B	9131.590974	I.S with MIP-gap	9.33979e-05	496.129
Bayg1003A	7687.72851	I.S with MIP-gap	9.87956e-05	2326.67
Bayg1003B	8457.99851	I.S with MIP-gap	9.65243e-05	184.758
Att1001A	**42677.802**	I.S with MIP-gap	.372425	5400*
Att1001B	**53313.4318**	I.S with MIP-gap	.475482	5400*
Att1002AB	**37216.58**	I.S with MIP-gap	.168075	3600*
Att1003A	**14859.3619**	I.S with MIP-gap	.312021	3600*
Att1003B	**20352.3683**	I.S with MIP-gap	.286678	3600*
Bier1001AB	**73067.1538**	I.S with MIP-gap	.584387	3600*
Bier1002A	**174083.9091**	I.S with MIP-gap	.545304	3600*
Bier1002B	**189226.0689**	I.S with MIP-gap	.538530	3600*
Bier1003AB	**103378.875**	I.S with MIP-gap	.648457	3600*

*The branch-and-bound reached the time limit.

Using CPLEX as a solver to find optimal solution for the FTSP-CA performs well using instances *Burma* and *Bayg*. In both cases the solution is either Optimal Integer Solution or

O.I.S with a low relative MIP gap. However, the performance of integer programming solver is not accurate for larger instances and requires an extensive amount of time if a feasible solution is desired. Bold solutions represent an integer solution with high relative MIP-gap due to the limitation in computing time. For that reason, a better solution technique is required for this type of NP-hard problem.

5.3 BRKGA results

For the second part of the computational experimentation, the performance of the Biased Random-Key Genetic Algorithm is compared with the result of the integer programming solver. The algorithm was coded in $C++$, using the API from Toso and Resende (Toso and Resende, 2012). All the experiments were executed with X-Code version 10.2 on a 2.4 Gigahertz Intel 8th Generation Core i5 processor with 8GB of RAM.

The BRKGA algorithm was calibrated with a population size $|P| = 1000$. The fraction of the population to be the elite set is denoted by $pe = .20$, and the percentage of the population to be replaced by mutants is denoted by $pm = .10$. The probability of an offspring to inherit the key from an elite parent during the crossover stage is $p_h = 0.7$. The algorithm runs for a maximum of 10000 generations. BRKGA will run until one of the ending criteria is fulfilled. First ending approach is reached when the algorithm provides 100 iterations without an improvement in the solution. Second approach is a time limitation of 7200 seconds.

The complete set of 36 adapted FTSPCA instances were solved with the BRKGA. Table 7 summarizes the experimental computation of an individual instance. The experiment was conducted running the algorithm 10 times for each instance. For each run the best solution in distance, total number of iterations and total time are display.

Table 7. Individual results for instance (Bayg1002B)

Bayg1002B			
Run Number	**Best Solution**	**Iterations**	**Time**
1	8521.188477	178	3
2	8491.591797	228	4
3	8477.118164	193	3
4	8521.188477	258	4
5	8477.118164	201	3
6	8477.118164	194	3
7	8651.52832	257	5
8	8477.118164	236	4
9	8477.118164	220	3
10	8491.59082	185	3
Avg.	8506.2678711	215.00	3.500
cMax.	8651.52832	258	5

Min.	8477.118164	185	3
σ	53.9806	29.13	0.71
C.V.	0.006	0.14	0.20

To compare the performance of the BRKGA and CPLEX, the average results of the 10-times running experiment for each instance are compared with the integer programming solution. Table 8 shows the complete arrangement of the 36 adapted instances using the BRKGA solver. The table is divided into family blocks and contains the specific instance name, the average cost of all runs, the best solution found which is the minimum cost value of the total runs, the standard deviation of the 10 experimental solutions, the coefficient of variation per instance, the average number of iterations and average time.

Table 8. List of BRKGA´s results

Instance Name	Avarage Cost	Best Solution	σ	C.V.	Iterations	Time
Burma1001A	15.251100*	15.251100*	0.00	0.00	103	1.00
Burma1001B	18.230801*	18.230801*	0.00	0.00	102	1.00
Burma1002A	32.514301*	32.514301*	0.00	0.00	123	1.10
Burma1002B	48.938400*	48.938400*	0.00	0.00	103	1.20
Burma1003AB	13.630000*	13.630000*	0.00	0.00	103	1.00
Bayg1001AB	8479.673145	8304.870117*	125.50	.015	189	2.80
Bayg1002A	8368.286426	8311.31543*	98.68	.012	233	3.90
Bayg1002B	9428.143554	9363.804688	40.20	.004	234	4.00
Bayg1003A	7731.651807	7687.728516*	45.38	.006	226	3.80
Bayg1003B	8506.267871	8457.118164*	53.98	.006	215	3.50
Att1001A	43780.2735	42242.19531	1369.98	.031	419	9.30
Att1001B	53142.596484	50978.86719	1753.55	.033	478	10.90
Att1002AB	36751.178125	35983.61719	566.61	.015	322	5.80
Att1003A	14923.035742	14859.34961	90.83	.006	198	3.00
Att1003B	20120.131445	20097.24805	55.03	.003	166	3.20
Bier1001AB	62571.777734	58377.09766	3400.04	.054	777	35.00
Bier1002A	155037.968750	145036.3125	8103.56	.052	418	72.80
Bier1002B	173146.039063	165589.9688	4914.33	.028	929	56.30
Bier1003AB	107131.796094	102284.0703	2275.27	.021	660	30.80
A1001A	7339.306836	6820.202637	377.13	.051	886	127.06
A1001B	8321.186621	7989.757812	387.02	.047	829	365.70
A1002A	6797.047559	6500.269043	330.20	.049	2199	320.50
A1002B	7112.204883	6701.203613	208.97	.029	2345	305.20
A1003AB	6512.760498	6055.009277	234.50	.036	1856	265.10
Gr1001A	7793.361133	7440.477539	249.87	.032	5818	3055.30
Gr1001B	8926.770996	8404.858398	314.23	.035	5628	2797.90
Gr1002A	6773.179688	6311.993652	326.91	.048	5479	2451.20
Gr1002B	7374.678711	6979.422852	245.47	.033	5811	2803.90
Gr1003A	6518.490283	5998.516113	243.72	.037	5344	2883.90
Gr1003B	7397.069000	6775.063965	310.74	.042	6261	3448.50
Pr1001A	918524.018750	877461.4375	28284.81	.031	7713	6894.50
Pr1001B	1091218.781250	1041818.1880	36901.70	.034	7278	6871.10
Pr1002A	1068720.562500	1019185.250	32579.67	.030	7537	7123.40
Pr1002B	1309075.600000	1251161.880	38170.69	.029	7258	6539.50
Pr1003A	907979.875000	861434.0625	35224.12	.039	7164	6012.20
Pr1003B	1058516.275000	1018187.813	26688.11	.025	6437	6140.40

The behavior of the BRKGA in the Burma´s block of instances is similar to the CPLEX solver. In both alternatives all of the instances reach an optimal integer solution with a non-significant relative gap value. For the Bayg´s group, CPLEX performs more accurate obtaining the lowest best solutions for all of the instances. However, the solutions provided by the BRKGA are also reliable. Although the variation of BRKGA, for Bayg´s instances 4 out of 5 instances reaches the optimal value at least once.

The main differences between both solution methods became visible in instances from block three and four. As mentioned in CPLEX results, the iteration process stopped due to limited time (3600-5400 seconds). For the BRKGA algorithm the program stopped when the criteria were reached, which takes from 3.00-72.80 seconds. In this short space of time BRKGA´s average results outperform CPLEX in 6 out of 9 instances. Furthermore, for individual best solution, the BRKGA develops a better or equal solution than the integer programming solver.

For instances above the 279 nodes (A, Gr, Pr), the BRKGA solutions behave similarly providing feasible solutions for the generous amount of running time; moreover, the solutions deviation is considerably low. The BRKGA results presents a coefficient of variation ranging from 0.004 – 0.054. According to Gomes (Gomes, 2009), the experiments with a coefficient of variation below 10% have a high precision. Due to the low coefficient of variation, the BRKGA method will provide similar and high-quality solutions for the FTSP-CA.

.

Chapter 6

Conclusions and Future Work

In this thesis work, the Family Traveling Salesman Problem with Capacitated Agents (FTSP-CA) is developed as a BIP formulation and evaluated by two different solving techniques. As an extension of the standard FTSP, the new adaptation of the problem adds two important factors that should be considered in logistics situations involving a family traveling salesman problem: agents and capacity.

The applied methodologies present feasible solutions for the FTSP-CA and can be used according to their strengths to deliver optimal integer solutions or optimal integer solutions with MIP-Gap. Integer programming strategy display an accurate alternative for solving smaller instances; however, the complexity of the problem does not allow to run instances above the fourth block of instances. Also, the amount of time used for solving instances from third and fourth block is limited to the computer specifications. On the basis of this constraints, the BRKGA methodology contributes for solving the FTSP-CA in a moderate time, providing feasible solutions with low variation.

Furthermore, for larger instances, the genetic algorithm outperforms the integer programming solver tool. There is no feasible way in time to provide an optimal integer solution with CPLEX for larger instances. The experimental computation displays a significant relative gap between the solution reached and the possible optimal integer solution.

In BRKGA methodology, two scenarios were evaluated in order to measure the impact of the parameters K (agents) and Q (capacity), in the optimal integer solution. As a result, the algorithm demonstrates that a configuration with higher number of agents (Scenario B), increases the total distance cost of the tour; however, increasing the number of agents provides a lower service time to complete the touring process.

In conclusion, solving the Family Traveling Salesman Problem with Capacitated Agents, requires the implementation of a heuristic solving technique, due to the complex structure of the NP-hard problem. The BRKGA heuristic performed as fast and accurate for the smaller instances compared to CPLEX solver, and for larger instances shows a good approximate value of the optimal integer solution, with lower coefficients of variation.

The presentation of the FTSP-CA enables future research for developing new specifications in the problem. As mentioned in the mathematical framework, the problem forces to all agents to leave the depot; however, in real life, some feasible solutions should not require employing all the available resources. This and other similar conditions can be applied to FTSP-CA in order to adjust to more demanding real-life parameters and constraints.

Bibliography

[1] Andrade, C. E., Silva, T., & Pessoa, L. S. (2019). Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. *Expert Systems with Applications*, *128*, 67-80.

[2] Angalakudati, M., Balwani, S., Calzada, J., Chatterjee, B., Perakis, G., Raad, N., & Uichanco, J. (2014). Business analytics for flexible resource allocation under random emergencies. *Management Science*, *60*(6), 1552-1573.

[3] Baniasadi, P., Foumani, M., Smith-Miles, K., & Ejov, V. (2020). A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. *European Journal of Operational Research*.

[4] Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, *6*(2), 154-160.

[5] Bernardino, R., & Paias, A. (2018). Solving the family traveling salesman problem. *European Journal of Operational Research*, *267*(2), 453-466.

[6] Chang, Y. S., & Lee, H. J. (2018). Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, *104*, 307-317.

[7] Fan, T., Tao, F., Deng, S., & Li, S. (2015). Impact of RFID technology on supply chain decisions with inventory inaccuracies. *International Journal of Production Economics*, *159*, 117-125.

[8] Fan, T. J., Chang, X. Y., Gu, C. H., Yi, J. J., & Deng, S. (2014). Benefits of RFID technology for reducing inventory shrinkage. *International Journal of Production Economics*, *147*, 659-665.

[9] Flood, M. M. (1956). The traveling-salesman problem. *Operations research*, *4*(1), 61-75.

[10] Gomes, F.P. Curso de estatística experimental. 15.ed. Piracicaba: Esalq, 2009. 477p.

[11] Gonçalves, J. F., & Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, *17*(5), 487-525.

[12] J.F. Gonçalves, M.G.C. Resende, and R.F. Toso. Biased and unibiased random key genetic algorithms: An experimental analysis. Technical report, AT&T Labs Research, Florham Park, NJ, 2012.

[13] Karp, R. M. (1975). On the computational complexity of combinatorial problems. *Networks*, *5*(1), 45-68.

[14] Land, A. H., & Doig, A. G. (2010). An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008* (pp. 105-132). Springer, Berlin, Heidelberg.

[15] Laporte, G., Mercure, H., & Nobert, Y. (1987). Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. *Discrete Applied Mathematics*, *18*(2), 185-197.

[16] Morán-Mirabal, L. F., González-Velarde, J. L., & Resende, M. G. (2014). Randomized heuristics for the family traveling salesperson problem. *International Transactions in Operational Research*, *21*(1), 41-57.

[17] Pintea, C. M., Pop, P. C., & Chira, C. (2007). The generalized traveling salesman problem solved with ant algorithms. *Journal of Universal Computer Science*, *13*(7), 1065-1075.

[18] Sathya, N., & Muthukumaravel, A. (2015). A review of the optimization algorithms on traveling salesman problem. *Indian Journal of Science and Technology*, *8*(29), 1-4.

[19] S. S. Srivastava, S. Kumar, R. C. Garg, and P. Sen. Generalized traveling sales- man problem through n sets of nodes. J. of the Canadian Operational Research Society, 7:97–101, 1970.

[20] Tomazella, C. P., & Nagano, M. S. (2020). A comprehensive review of Branch-and-Bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem. *Expert Systems with Applications*, 113556.

[21] R.F. Toso and M.G.C. Resende. A C++ application programming interface for biased random-key genetic algorithms. Technical report, AT&T Labs Research, Florham Park, NJ, 2012.

[22] Yang, J., Wu, C., Lee, H. P., & Liang, Y. (2008). Solving traveling salesman problems using generalized chromosome genetic algorithm. *Progress in Natural Science*, *18*(7), 887-892.

[23] Yuan, Y., Cattaruzza, D., Ogier, M., & Semet, F. (2020). A branch-and-cut algorithm for the generalized traveling salesman problem with time windows. *European Journal of Operational Research*.

[24] Zhang, L. H., Li, T., & Fan, T. J. (2018). Radio-frequency identification (RFID) adoption with inventory misplacement under retail competition. *European Journal of Operational Research*, *270*(3), 1028-1043.

[25] Zhou, W., Piramuthu, S., Chu, F., & Chu, C. (2017). RFID-enabled flexible warehousing. *Decision Support Systems*, *98*, 99-112.

[26] Zia, M., Cakir, Z., & Seker, D. Z. (2018). Spatial Transformation of Equality–Generalized Travelling Salesman Problem to Travelling Salesman Problem. *ISPRS International Journal of Geo-Information*, *7*(3), 115.

.