SOLVING A MODIFIED TSP PROBLEM BY A GREEDY HEURISTIC FOR COST MINIMIZATION

A Thesis

by

Murat Çal

Submitted to the

Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Industrial Engineering

Özyeğin University
June 2017

SOLVING A MODIFIED TSP PROBLEM BY A GREEDY HEURISTIC FOR COST MINIMIZATION

Approved by:

Assoc Prof Ali Ekici, Advisor,
Department Industrial Engineering
Özyeğin University

Asst Prof Enis Kayış,
Department Industrial Engineering
Özyeğin University

Asst Prof Fahrettin Eldemir,
Department Industrial Engineering
Yıldız Technical University

Date Approved: 14 August 2017

To My Mother and Father

ABSTRACT

Photographing a large area in an instant is only made possible by satellites. Satellites are able to help reducing the time required for the photographing/monitoring process, however, satellite images are not available all the time, and even if they exist, they are not easy to evaluate for decision makers. So decision makers use surveillance drones, also called unmanned aerial vehicles to take photos of the predefined region. Given a set of nodes defining an area, each node should be monitored and analyzed. In that sense, the problem may be defined as a variant of Traveling Salesman Problem. It is not practical, however, just to go to every node and take a shot. Instead, one can make use of the concept of relative heights, meaning if there is a node in a higher or more appropriate position than that of another node, drones can go to that higher positioned node, take a photo and are able to monitor the other node that is 'seen' by the current node.

In this study, we provide a mathematical model for this modified TSP, in which we should cover all the nodes either by photographing or physical visits and minimize the total travel cost. Then, we provide a greedy heuristic to find solutions and compare the values with optimal solutions as well as lower bounds to evaluate performance. We observe that in low photo costs, our algorithm may provide solutions within 1% of the solutions or lower bounds on hand, and in high photo costs the algorithm is still able to provide good solutions up to 5-10% of the solutions or lower bounds on hand.

ÖZET

Büyük bir alanı anında fotoğraflayabilmek yalnızca uydular tarafından mümkündür. Uydular, fotoğraf/izleme süreci için gerekli zamanı azaltmaya yardımcı olabilir, ancak uydu görüntüleri her zaman mevcut değildir ve bu görüntüler mevcut olsa dahi karar vericiler için değerlendirilmesi kolay değildir. Bu sebeple karar vericiler, önceden tanımlanmış bölgenin fotoğraflarını çekmek için insansız hava araçları da denilen gözetim uçakları kullanmaktadır. Bir alanı tanımlayan bir düğüm kümesi (ağ) göz önüne alındığında, her düğüm izlenmeli ve analiz edilmelidir. Bu anlamda sorun, Gezgin Satıcı Problemi olarak tanımlanabilir. Bununla birlikte, her düğüme gitmek ve fotoğraf çekmek pratik değildir. Bunun yerine göreceli yükseklik kavramından yararlanılabilir, yani başka bir düğümden daha yüksek ya da daha uygun bir konumda bir düğüm varsa, dronlar daha yüksek konumlandırılmış düğüme gidebilir, bir fotoğraf çekebilir ve mevcut düğüm tarafından görülen diğer düğümleri otomatik olarak izlemiş olur.

Bu çalışmada, yukarıdaki gibi modifiye edilmiş Gezgin Satıcı Problemi (MGSP) için, hepsi ziyaret edilmeksizin tüm düğümleri kapsayan ve toplam seyahat masrafı ile fotoğraf çekme maliyetini minimize eden bir matematiksel model sunulmaktadır. Ardından, çözüm bulmak için sezgisel bir yöntem önerilmekte ve performansı değerlendirmek için sezgisel tarafından bulunan değerler optimum çözümlerle ve alt sınır çözümleri ile karşılaştırılmaktadır. Düşük fotoğraf maliyeti olan durumlarda algoritmanın optimum çözümün ve alt sınır değerlerinin %1 fazlasına kadar iyi sonuçlar verdiği, yüksek maliyetli durumlarda ise nodlar arası görsel erişime bağlı olarak optimum çözümün veya alt sınır değerlerinin %5-10 fazlasına kadar sonuçlar üretebildiği görülmektedir.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Monitoring and photographing large areas for security, safety or other specific reasons requires careful and detailed planning since decision makers always have limited time and other resources. In a given amount of time, an area within a building, a zone within a region or even a region itself should be monitored to take precautions against different emergencies. To prevent fires in a bank or school, to set up cameras or alarms for burglary, to conduct periodical controls in forests for fire prevention or to monitor borders for intruders and terror attacks are among different motivations of field scanning.

For many years, Traveling Salesman Problem is approached in different ways by different researchers, and tremendous numbers of studies are created to effectively and efficiently solve the problem. Given the number of nodes, one should find an optimal way in terms of time or cost minimization or some other purpose together with contacting each node.

In our problem, an agent does not have to go to each node physically, rather, the agent is able to reach some nodes from the current node visited, and this saves time and money. In that case, TSP behaves like the so called Set Covering Problem (SCP), where given an entity set S, one should select a minimum number of entities to cover all entities in terms of an objective. For example, municipalities aim to select a minimum number of sub-districts to open fire stations to reach all sub-districts within a predefined amount of time. In that sense our problem is a mix of TSP and SCP, where we should physically or digitally reach all nodes by either visiting them or making connections with them. In doing this, we should minimize time and/or cost incurred. Accordingly, we define our problem in the next section.

1.1 Problem Definition and Regarded Models

As stated in the beginning of this chapter, our problem consists of- and is a mixed version of two different problems, namely the Set Covering Problem and the Traveling Salesman Problem. Given below are the simplest versions of both problems and some extensions used in various cases. Once an introduction is made to both problem types, we provide our model definition together with its mathematical model.

1.1.1 Set Covering Problem and Traveling Salesman Problem

Set covering problem, also known as set cover problem, concerns with minimizing the total cost of all selected elements or total number of selected elements itself given that these elements are required to cover a pre-given number of input sets. It has different versions such as edge covering problem and vertex covering problem, maximum coverage problem or geometric set cover. Even though the modified versions of this problem may be of mixed integer type (MIP), simplest version is a pure integer problem, whose formulation is pretty straightforward.

To illustrate the Set Covering Problem, suppose that a municipality wants to setup fire stations within a specified region, which is divided into sub-regions with the motivation of efficient management. Then, the municipality seeks to minimize the number of stations to be set up such that all sub-regions are within at most 15 minutes of travel time. In some cases, some locations may be more expensive than others and if that is the case, the municipality may want to minimize the total cost incurred by setting up these fire stations rather than minimizing their number.

The traditional formulation of TSP has sub-tour elimination constraints defined by subset S that denotes the set consisting of nodes in that particular subset of node set N and cardinality of S means the number of nodes in S, as also stated in [2] and as will be explained in literature search. For any subset to ever exist, there will be at least two nodes and a route, whenever there are nodes to form a subset due to cost optimality, it is prohibited in that the corresponding binary variables are not allowed to be equal to 1 all at the same time.

1.1.2 The Photo-Modified TSP

Our model is verbally defined as follows: Given node set N, edge set E that connects each and every node in N, a starting node where our agent is located, we should cover each node in N containing also the starting node, by either directly visiting that node or reaching that node by photographing it from a distance without physically going there. It is possible for the agent to visit a node just to cover that particular node and not to take a photograph. As such, the agent is not allowed to take photograph at any node to which the agent does not physically go. This operation should be performed at the least cost, considering the travel costs between all the nodes making up a complete route and also the photographing costs that enable connecting to a node within a distance but incurring an extra technology cost. Another parameter to be taken into account is the reachability issue related to distances between the nodes. The reachability of a node from another one, at which a photo is taken, is defined by some function that assigns a probability of being "reachable" depending on the distance.

The reason why we mention Set Covering Problem and also Traveling Salesman Problem is because the problem is a different version of TSP where all nodes should be covered without the necessity of visiting all of them one by one, rather, we have the opportunity to take a "photo" at a node to cover other nodes that are reachable from that particular node. SCP plays an important role in the heuristic algorithm we designed. In the first step, we divide the problem into two parts, and in the first part we define the photo nodes by which we are able to cover all nodes in the node set N. In that sense, the problem is similar to SCP and becomes relatively easier to solve.

As you will infer from the application areas section, there are lots of real life cases where the problem we are dealing with is considered. We name our problem as the photo-modified TSP for general purposes, and any digital connection or remote operation will be defined as photographing. Throughout the study, we will refer to means of transportation, called salesmen in traditional TSP or vehicles or unmanned aerial vehicles as "agent".

Reachability Issue: Since we are dealing with monitoring costs and fuel costs in relation to traveling of the agent, it would be irrelevant to consider the highest technology. The reason is explained as follows: There are cameras able to recognize smallest insects from thousands of meters, so we should take into account that our

camera is of moderate size, moderate cost and moderate capability. These moderate characteristics correspond to gimbal cameras that are able to prevent vibration and movement based disqualifications and mainly used in mini unmanned aerial vehicles for border monitoring, security management and precautionary motivations. Mini unmanned aerial vehicles, called mini İHA in Turkish, are one of those used for short range trips in Turkey. If we consider agents that are able to see each and every node regardless of how far the distance is, then there is no sense in defining such a problem because the agent will be able to reach every node from the starting node.

First and most important factor in a camera mounted on an unmanned aerial vehicle is the visibility range. Second factor in the camera tool's capability is lens selection. There are two important things to consider in selecting the lens in a camera, namely the conformance to camera's capabilities and conformance to environmental conditions. Another factor is the angle of view, determining how widely a camera can reach from its vertical focus and record videos or take photos. One should note that even though a point distanced vertically from a camera is seemingly qualified and enables easy detection, not all the points in the same angle of view may be of the same quality. We might as well use unattended ground sensors and improve the capability of the cameras, as Fargeas et al (2015) study in their work, however, such improvements are considered as future work since our target areas are deprived of such sensors and we want to maintain simplicity for our model structure. There can be lots of other details if we specify our problem to a real life case rather than generalizing it, thus we avoid giving here technical details regarding the visibility range of cameras or lens selection, for which we did additional research during the dissertation period. For simplicity, we assume that appropriate lens for our camera is used, that is, lens or any other factor such as focusing distance does not have quality-reducing effects on the camera itself and they are all selected appropriately. We also assume that after a distance from a node, that node becomes harder to "see", that is, the probability to cover a node from the photographing node decreases. We explain how we determined this probability in Section 3.2.1.

1.2 Photo-Modified TSP Problem Formulation

Our problem has already been defined in the introduction part with necessary assumptions. In this chapter, we provide our mathematical model with necessary notation.

1.2.1 Notation

Parameters:

$\mathbf{d_{ij}}$ Distance between node *i* and node *j*

$\mathbf{c_i}$ Photographing cost in node *i*

$\mathbf{r_{ij}}$ Binary number showing whether a photo taken at node *i covers node j*

N Set of nodes including starting node (0) such that N={0, 1, 2….n}

Decision Variables:

$\mathbf{x_{ij}}$ $\begin{cases} 1 \text{ if there is a direct path from node i to node j} \\ 0 \text{ otherwise} \end{cases}$ $\forall i, j \in N$

$\mathbf{y_i}$ $\begin{cases} 1 \text{ if a photograph is taken at node i} \\ 0 \text{ otherwise} \end{cases}$ $\forall i \in N$

$\mathbf{u_i}$ Order of node i in any complete route $\forall i \in N$

1.2.2 Objective Function

We seek to minimize the total cost incurred by traveling between the nodes and also photographing at certain nodes to consequently cover all nodes in N, as shown in **(1):**

$$\sum_{i=0}^{n}\sum_{j=0}^{n} d_{ij}x_{ij} + \sum_{i=0}^{n} c_i y_i \qquad \textbf{(1)}$$

1.2.3 Constraints

Node Coverage: All nodes should be covered. This can be done either by directly going to that node or taking a photo from a node that "covers" that particular node. This is mathematically expressed in **(2):**

$$\sum_{i=0, i\neq j}^{n} r_{ij}y_i + \sum_{i=0, i\neq j}^{n} x_{ij} \geq 1 \quad \forall \mathrm{j} \qquad \textbf{(2)}$$

Visited Nodes Are Left: If there is a node for taking photo, that node should be left to go to another node to eventually build up the route, as given in **(3):**

$$y_i \leq \sum_{j=0, j\neq i}^{n} x_{ij} \quad \forall \mathrm{i} \qquad \textbf{(3)}$$

Note in advance that there is also another constraint assuring that visited nodes are arrived, however, the next constraint we write includes this arrival constraint by incorporating it into a balance equality.

Node Balance: Node balance, together with **(3)** makes sure that any node will be left after physical or photographical visit to that node. The balance is given in **(4)**:

$$\sum_{j=0, j\neq i}^{n} x_{ij} - \sum_{j=0, j\neq i}^{n} x_{ji} = 0 \quad \forall \mathrm{i} \qquad \textbf{(4)}$$

Leaving the Source Node: Our agent is located at a starting node, so it should be leaving the starting node, shown in **(5):**

$$\sum_{j=0}^{n} x_{0j} = 1 \qquad \textbf{(5)}$$

Arriving at the Source Node: Similarly, our vehicle should return to starting node after all nodes are covered. This is made sure by **(6):**

$$\sum_{j=0}^{n} x_{i0} = 1 \qquad \textbf{(6)}$$

Sub-tour Elimination: The most important issue in solving our problem as well as solving a TSP is sub-tour elimination. This constraint correspondingly eliminates sub-tours and makes sure that additional decision variables are only allowed to reflect the order of the nodes within a feasible route. This is given in **(7)**:

$$u_i - u_j + n \cdot x_{ij} \leq n - 1 \quad \forall \mathrm{i} > 0, \mathrm{j} > 0, \ \ \mathrm{i} \neq \mathrm{j} \qquad \textbf{(7)}$$

CHAPTER II

PREVIOUS WORK

Much effort has been devoted to searching for best solutions for Traveling Salesman Problem. Karp (2010) shows that the problem is NP-complete, which leads to development of countless heuristic algorithms and also LP relaxation methods to efficiently solve the problem [6]. So TSP includes combinatorial constraints such as sub-tour eliminations and similar to vehicle routing problem, it is also NP-hard [3-4]. Papadimitriou (1977) concerns with Euclidean TSP and studies two different problems, namely the ordinary TSP, where one needs to go back to start node after visiting each node and the paths TSP, where one does not need to go back to original node after visiting all nodes [9]. In particular, Papadimitriou proves both problems' Euclidean versions are NP-complete, and restrictions on distance inputs do not make TSP easier.

This chapter continues with two subchapters explaining the solutions methods and different approaches for different versions of Set Covering and Traveling Salesman Problems previously studied.

2.1 Set Covering Problem

One of the algorithms that comes to mind immediately when trying to solve Set Covering Problem is the Greedy Approximate Algorithm, mainly defined as α-Approximation Algorithms since the results given by these algorithms cannot be worse than α times the objective function's optimal value. Greedy algorithms' working procedure is to find the node to be selected in such a way that it brings the maximum number of other nodes to be covered, and to do this job at minimum cost. This procedure is repeated until all nodes are covered. Mostly known values of α is 1.5, 2 and 3, after which an algorithm increasingly becomes unsuccessful. The most literarily used greedy algorithm finds a solution with at most $k \ln n$ sets of nodes, given originally n nodes to cover. It is important, however, to note that even though Set Covering Problem does not contain a combinatorial constraint like sub-tour elimination seen in TSP, number of nodes exponentially increases the solution time and this makes it necessary to develop algorithms to efficiently solve the problem. Exact and approximate algorithms developed for Traveling Salesman Problem are also employed in SCP. Modified genetic algorithms are used in [10], [11] and [12] especially to handle local optima and provide efficient solution procedures. In [13] several algorithms able to provide both exact and approximate solutions are outlined and sorted. In that sense, [13] provides a useful index for different approaches for SCP. Before this study, same researchers had also suggested a Lagrangian-based heuristic for crew scheduling in a broader sense in [14]. Dynamic pricing as well as column fixing are effectively used to obtain improved solutions, providing optima in 92 out of 94 instances.

2.2 Traveling Salesman Problem

Much progress has been made since the introduction of TSP into literature. Soon after the problem was defined, solution procedures together with the required effort were suggested.

2.2.1 Background and Pioneering Studies

Data plays an important role in TSP's progress. Researchers like Dantzig and Karp provided very good distance matrices to start with, however as the problem size grew, their applicability reduced very fast. So over time, people began to create test instances to computationally experiment with TSP. Reinelt's so-called TSPLIB, which contains examples up to 85900 cities is the mostly known test database. Other than TSPLIB, there are also National TSP Collection, VLSI TSP Collection, The World TSP, Mona Lisa TSP and United States TSP that can provide instances up to 1.904.711 cities, which cover all the cities and populated areas in the world. In our study, we generate our own data when we create our binary matrices of accessibility, however, when it comes to generating distance matrices or test our model in IBM ILOG CPLEX, we simply made use of some of these instances as well. Years after [2]'s highly inspiring infrastructure to handle TSP, many people tried the database to test and validate their own algorithms, and even hardest instances were tried to be solved [15].

The difficulty in defining and handling these constraints is handled quite in an elegant manner by Dantzig, Fulkerson and Johnson (1954), where they employ an estimation procedure to estimate upper bounds for a 49-city problem and where they illustrate the sub-tour concept by using cutting plane methods and optimality concept by orthogonality and feasibility [2]. Even though literature seeks to find optimal solutions for TSP just as this study did in 1954, we should definitely note that exact methods such as cutting planes and branch and bound methods do not provide quick solutions when the problem size is much larger. The study is a pioneer, however, in that it is denoted as the first time such a large problem instance is solved to optimality. Cutting plane concept are further investigated in [15] and [20] where authors deeply discuss the concept of a cut procedure that is used throughout the solution process. It is the content of the authors that given a solution not satisfying all the constraints which impose

another cut to be added, a cut based on tangled tours and combined with old fashion cutting plane process is employed to effectively define a plane. Same authors enumerate other methods derived from numerous studies in [21]. Another cutting plane algorithm is proposed in [22], that is based on a polyhedra strengthening the LP relaxation. In exchange, the linear program created has about twice the size of the usual LP relaxation. Therefore, they propose a lifting step to reduce the size for solvable complexity. In our study, our model's sub-tour elimination constraint makes our problem weaker in terms of LP relaxation, and in future studies, we may try to use this strengthening step in relaxing our model to find lower bounds.

2.2.2 Finding Lower Bounds

Laporte's work sheds light on exact and approximate algorithms and provides an extensive guide to make use of in finding lower bounds in [23]. In addition to integer programming formulations, the assignment lower bound and related branch and bound algorithms allow finding a lower bound by relaxing integer constraints, leading to an assignment problem that can be solved in $O(n^3)$ time [24]. Other exact algorithms that may be of interest are the shortest spanning arborescence bound and related algorithms as in [25], the shortest spanning tree bound and related algorithms and 2-matching lower bound and related algorithms as in [26] and [27]. As for approximate algorithms, of which we have talked and will be talking below, heuristics with guaranteed worst case performance that base on spanning trees and are improved by Christofides heuristics as in [28], heuristics with good empirical performance such as nearest neighbor which can further be divided into tour construction, tour improvement and composite heuristics studied in [29]. Another study compares the lower bound obtained by their model by the Held-Karp bound and the lower bounds obtained in [30] and [31]. It is shown that the Held-Karp bound is at least as tight as van der Veen, which is as least as tight as minimum weight tree. Both Held-Karp and can der Veen yielded best bounds, among which van der Veen prevailed since it provides values in linear time [32]. Another study in [33] handles the issue of finding lower bounds by a 1-tree learning based Lagrangian relaxation technique. Obtaining lower bounds for large instances of a problem that cannot be solved within reasonable times is also important especially when it comes to compare a heuristic algorithm with other exact or

approximate algorithms in terms of solution time and solution quality. Mathematical programming formulations such as single or multi-commodity flow can be used to formulate a minimum spanning tree (MST) and then applying branch and cut and price algorithms [129]. Our approach is similar to what is studied in [130], but it is different in that we incorporate photographing variables and make a coverage possible also by taking a photo from a node able to cover other nodes.

2.2.3 Mathematical Formulations

In addition to the constraint formulation suggested by [2] and also the introductory chapter, there are several other formulations to integer programming of TSP, as summarized and computationally compared in [34]. Four types of classes, namely conventional (C), sequential (S), flow based (F) and time based (T). Conventional formulation is what we see in Dantzig, Fulkerson and Johnson's work. It has $2^n + 2n - 2$ constraints and $n(n - 1)$ binary variables. To solve the problem in tolerable times, sub-tour elimination constraints are added as they are violated, thus reducing total computation time.

Second class is sequential formulation, which we used in CPLEX to effectively handle the sub-tour elimination constraints. Accordingly, the sub-tour constraint is replaced by what we write in constraint **(7)**. Even though the constraint **(7)** is written such that the variable u_i showing the sequence of node i visited in the route is defined as a continuous variable, it may be also be defined as integer to precisely notify the node sequence in a given route. Sequential formulation has $n^2 - n + 2$ constraints, $n(n-1)$ variables and $n-1$ continuous variables. This is known as the Miller-Tucker-Zemlin formulation, as stated in [35] and also compared in [36] and [37]. Main advantage of this formulation is the decreased problem size, and ease to manipulate the model in cases we want or prefer some cities to be visited earlier. It is also noted, however, that LP relaxation of this formulation compared to [17] is much weaker due to the need for elimination of continuous variables in relaxation process.

Third class is flow based formulation, which we also see in vehicle routing problems, studied in [38], [39] and [40]. This formulation may be subdivided into single flow, two flow and multi-commodity flow formulations, in each of which appropriate

indices are defined but the conceptual definitions remain the same. Single flow formulation incorporates a y_{ij} to traditional formulation to denote the flow in the corresponding i-j arc and adds the following constraints:

$$y_{ij} \leq (n-1)x_{ij} \quad \forall i, j \in N, i \neq j \tag{8}$$

$$\sum_{j, j \neq 1} y_{1j} = n - 1 \tag{9}$$

$$\sum_{i, i \neq j} y_{ij} - \sum_{k, i \neq k} y_{jk} = 1 \quad \forall \mathrm{j} \in \mathrm{N} - \{1\} \tag{10}$$

(9) and **(10)** allow n-1 units to flow into city 1 (starting node in our case) and 1 units from other cities, where any flow, as stated by **(8),** can exist if there is a direct path between the corresponding city i and city j. 2 flow and multi-commodity flow formulations can be looked and studied accordingly. As for multi-commodity TSP, [41] gives a formulation to model a transportation network. More formulations can be found in [42], [43], [44], [45], [46] and [47], where [42] compares the strengths of their LP relaxations and provides some base to select the best fit for conducted studies.

Fourth class is time based formulation, where we divide the route and moves into time-based intervals, thus modeling the problem in such a way that whenever there is a condition in time t at a particular node, we impose another condition in time t+1 in another node and so on. In addition to binary arc variable x_{ij}, we add the following to the problem:

$$y_{ij}^{\ t} = \begin{cases} 1 \text{ if arc i - j is traversed at stage t} \\ 0 \text{ otherwise} \end{cases}$$

such that

$$\sum_{i, j, t} y_{ij}^{\ t} = n \tag{11}$$

$$\sum_{j, t, t \geq 2} t y_{ij}^{\ t} - \sum_{k, t} t y_{kt}^{\ t} = 1 \quad \forall \mathrm{i} \in N - \{1\} \tag{12}$$

$$x_{ij} - \sum_{t} y_{ij}^{\ t} = 0 \tag{13}$$

This model, as stated in the study, has *n(n+2)* constraints and *n(n-1)(n+1)* binary variables. Even though we may claim that the formulation is a more compact one, its LP relaxation form is weak and therefore becomes slow in terms of running time. [48] deals with more time-based models and puts all advantages and pitfalls in the modelling

process. We do not prefer this modelling approach since we need another variable to denote photographing/connection decisions and incorporating this variable would cause another complexity in modelling due to inclusion of time stages.

2.2.4 Neural Networks

As for neural networks studies, [54] and [55] investigate different methods used in solving TSP. [54] compares three different approaches: integer linear programming to obtain optimal solutions without time consideration, Hopfield Neural Network as explained in [56], and Kohonen Self Organizing Feature Map. As stated in [56], Hopfield Network is able to compute good enough solutions by using analog inputs and uses these solutions as the network is extended. These 'milestones' to shed light on the optimization of the problem by spending less time is actually the what makes a network a neural network where some solutions are collectively computed by utilizing an energy function. Lots of other problem types are also solved by making use of neural networks [55], where TSP is the mostly investigated problem type and is characterized either by binary variables as in integer formulations or continuous variables as in [57]. Studies [58], [59], [60] and [61] also consider Hopfield neural network and provide extensions in terms of computational structures of the same algorithm. In [62] Hopfield network performance is compared with an ant colony system. For more information on neural networks, [65] can be consulted which classifies all neural network studies based on solving TSP and TSP with backhauls as in [66] and divides the work into three main groups, namely the Hopfield-Tank Network, elastic net algorithms and self-organizing maps. For more neural network applications employing Hopfield networks, reader may look at [67] and [68]. An interesting neural network application is the so-called Cuckoo Search Algorithm, introduced firstly by [69] and inspired by cuckoo behavior in nature [70].

2.2.5 Swarm Based Techniques

Another metaheuristics for solving TSP are swarm based optimization techniques, such as ant and bee colony algorithms [71], swarm particles or what we found in our thesis period, named African Buffalos. Particle swarm is developed by [72] and parametric extensions are created by the same logic in [73], [74]. Parameters like inertia weight or

discrete structures are also combined with the main concept to accelerate the method and obtain qualified solutions [75]. Precedence constraints in addition to sub-tour elimination constraints make TSP even more complex and this is where particle swarm optimization (PSO) comes in [76]. Convergence speed in such algorithms may be increased by eliminating crossovers, and re-designing subtraction operator contributes to possibility of solving even larger problems thanks to early convergence [48], [49], [77] and [78]. In [79], particle swarm techniques are combined with genetic simulated annealing ant colony system to solve 25 TSP instances obtained from TSPLIB, which is then compared with the self-organizing maps [53], and neural network approaches in [80] and [81]. [86] deals with TSP when there are time windows constraints. Time window constraint is handled by a beam ant colony optimization algorithm in [87], where stochastic sampling for differentiating between partial solutions is used. A combination of particle swarm optimization, ant colony optimization and 3-opt heuristics is employed to solve TSP in [88]. A bee algorithm in [90] investigates a deterministic case by the very same natural waggle dance behavior of bees. Different applications of swarm particle optimization approaches may be consulted in [92], [93] and [94].

A different aspect, namely an optimization technique called African Buffalo optimization, is concerned by [95] where the technique is compared with the Randomized Insertion Algorithm and performed better [96]. Buffalos run faster also in [97] and [98], where African Buffalos are compared to hybrid Honey Bees and Lin-Kernighan algorithms on TSPLIB95 benchmark data.

2.2.6 Simulated Annealing

Simulated annealing is used also in [82], where a local search algorithm combines greedy approaches with simulated annealing to solve TSP. Problem types that are fit for simulated annealing are mentioned in [83] together with a Metropolis algorithm firstly introduced in 1953. The algorithm is compared to simulated annealing in [84] and in [85],

2.2.7 Genetic Algorithm

[99] investigates different selection strategies in genetic algorithm (GA) approach to determine which parameter selection in terms of parent selection affects the solution time in the highest rate. As the problem size increases, it becomes more likely for tournament selection strategy to suffer from early convergence to some local optima [100]. As mentioned earlier, vehicle routing problem with precedence constraints is appropriate to be modeled as a TSP, which can produce invalid candidate solutions if solution representation is made order-based [101]. It is also important to prevent the algorithm to converge to some locally optimal solution by additional manipulation of results or preventive steps, such as evolutionary adaptation of simulated organisms [102]. Even though generalized chromosome genetic algorithms are claimed to solve standard TSPs as well as generalized TSPs (GTSP) where a tour is created by visiting single nodes in each clusters containing multiple nodes [103], generation of feasible and quality tours is an important aspect to achieve good enough solutions in the end, thus genetic algorithms are supported by local heuristics, such as the hybrid algorithm that produces solutions beforehand and then refines them by Lin-Kernighan local search [104]. It is also possible not to use a local search algorithm separately from the genetic algorithm but to convert local search heuristics into a genetic structure like it is done in [105]. [106] introduces a Memetic Algorithm that incorporates an extensive neighborhood search to generate new solutions using crossover. For the same problem, [107] again uses local search supported by genetic algorithm. Another genetic algorithm called the immune genetic algorithm (IGA) employs two ways to make use of an immune operator [108]. For more applications used in combinatorial problems, [109] classifies and enumerates previous studies based on scheme; [110],[111],[112] and [113] are extensive books providing traditional as well as combined approaches with respect to genetic algorithms, whereas [114], [115], [116] and [117] are the pioneering conference proceedings as cited in [107].

A different approach is used in [118], where symmetric and asymmetric TSPs are solved. Lots of different representations may be seen in [119]. Even creation of initial populations may be in tremendous numbers of ways, which in turn affect the solution quality and the rate at which the optimal solution is reached [120]. TSP with

precedence constraints is solved also by a genetic algorithm where the concept mainly bases on topological sort, namely the edges ordered in a directed graph [121]. As for mutation operators, a new one called Greedy Sub Tour Mutation (GSTM) is suggested in [122].

2.2.8 Other Approaches

Tabu search [123], owing its name to its property of holding previous solution information in memory, is also used in TSP problems. A balance between tabu conditions and aspiration criteria is important and well defined memory function play a vital role in reducing the total solution time. More information can be found in [123].

There are also algorithms that are motivated on their own and do not rely on Meta-heuristics, thus having no particular name like our algorithm. An example is given in [124], where edges are added to a tour progressively by selecting the less disturbing edge for the edges not yet selected. The algorithm is compared to famous Quick Method for evaluation.

A different version of TSP is maximum reward collection problem, studied in [125] and in [126]. A variant of the same problem with multiple agents is studied in [127]. Salesmen try to maximize their reward minus any costs by visiting the nodes which worsen as time passes. A penalty based heuristic is suggested in [125]. In [127], a Cluster and Route Algorithm (CRA) is suggested to find good enough solutions.

The maximum collection problem is studied also in [128], where this time a single constraint is imposed such that each nodes does not have to be visited exactly once, each node has a given reward and starting node should be returned to, maximizing the total reward within the given time. Similar to what we formulated in our heuristics, this study adds assignment problem's constraints to easily solve the problem by Lagrange relaxation.

CHAPTER III

SOLVING OUR PROBLEM BY MATHEMATICAL MODEL AND BY A GREEDY HEURISTIC

In this chapter, we solve our mathematical model by CPLEX, show that the problem is NP-Hard for higher numbers of nodes, explain the dynamics of a heuristic algorithm we designed to easily solve the problem and compare the results.

3.1 Solving the Model Mathematically and Evaluation of Results

In this section, we give information about how we generated our own problem instances and also our comments on computational results. To solve the problem, IBM ILOG CPLEX Optimization Studio 12.6.1 is used. The software is installed on Windows 7 Professional Operating System 64 bit, with Intel® Core™ i7-3630QM CPU @ 2.40 GHz processor and 8 GB RAM.

3.1.1 Generation of Different Problem Instances

Before explaining the methods we used for generating different problem instances, it is necessary to explain any assumptions we made for simplicity. In the construction of our distance matrix, distance is measured in Euclidean form. Since we want to maintain simplicity as much as possible, we directly took distance matrix as the traveling cost matrix, defining the traveling cost 1 monetary unit for 1 unit distance. We assumed a coordinate range between 0 and 1000, making the largest distance between any points at most $1000\sqrt{2}$.

Reachability Issue Revisited: In the introductory part, reachability was mentioned as one of the inputs to our model. To be able to adapt this characteristic into our model, we define a mathematical function through which we obtain our relative height (reachability) matrix. To reflect the decrease in capability as the distance increases we define the following function:

$$f(x) = \left(1 - \frac{d_{ij}}{d_{\max}}\right)^7 \qquad \textbf{(14)}$$

where d_{ij} denotes the distance between node i and node j and d_{max} denotes the maximum distance among all the pairs for a given set of nodes. As stated above, this function takes a value less than 0.1 after the integer value 392, a distance less than 400 unit distance when the maximum distance is 1400 units between two nodes. Note also that the coordinates of nodes vary between 0 and 1000, thus creating a square of 1000 x 1000 unit distances. The curve that shows the probability of 'seeing' a node is given in Figure 1. The maximum distance is determined as 1400. Note that if the distance is equal zero, the probability of seeing a node becomes 1. In other words, if we are at a node and we want to take a photo, we will definitely be able to cover that node. Note also that if two nodes are located by the maximum distance, the probability of them to see each other becomes zero.

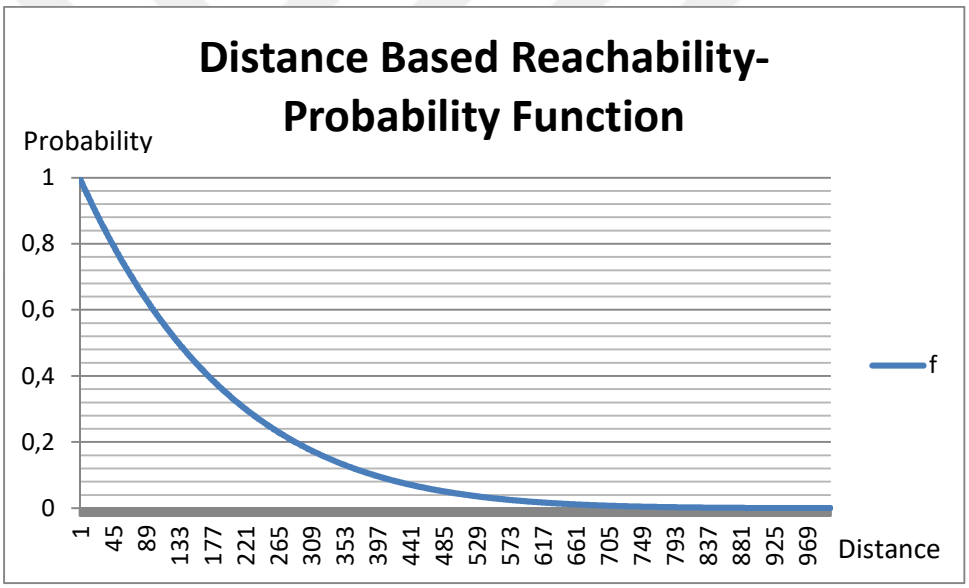

Figure 1. Distance-Probability Curve

The motivation behind the selection of this probability function is simply the case we are considering. Technical details we obtain from [5] and also the rule stating the reachability capability of a camera is taken as an example. For our particular case where we assumed a 1000 x 1000 grid, we want the nodes to contain limited numbers of 1s in the reachability matrix, thus allowing the problem to consider photographing opportunities. For this reason, we take the probability function to the seventh power to guarantee the steepness of the curve and be sure that a photo node does not cover "almost" all the nodes and does not make the problem meaningless since otherwise even

large numbers of nodes can easily be covered by photo nodes and the problem turns into SCP rather than TSP. For other cases where other factors play a role in reachability, this function may be defined based on those factors, for example, lens selection and weather conditions may be incorporated into the function appropriately. It should be noted that this function, at the end, is selected arbitrarily, the power of it could be 6 instead of 7, or 10, depending on how much reachability we want in our reachability matrix. We selected 7 as to make sure that we have enough number of coverages in our reachability matrix.

Next question to answer is how probabilities that a node covers another node are calculated. Depending on the distance, the function in **(14)** calculates a number between 0 and 1, whose graph has already been given in **Figure 1**. Taking the value as an input, Excel generates a random number between 0 and 1. If the generated number is greater than the value of the function, corresponding coverage value in the reachability matrix takes the value of 0, and it takes the value of 1 otherwise. The following formula is used in generating reachability matrices:

IF(RAND()>H406;0;1)

To solve different instances of the model, we modified both distance matrices and also the cost vector. Distance matrix is manipulated in two different ways:

1. Neighborhood Generation: We divide the distance matrix into numbers of neighborhoods which are defined by different distance ranges. For example, first neighborhood contains nodes whose coordinates are determined between the values 0 and 300, second neighborhood contains nodes whose coordinates are valued between 400 and 500 and third neighborhood contains coordinates between 800 and 1000 and so on. Depending on the instance, we also create neighborhoods together with a larger neighborhood containing the whole range. We create these neighborhoods to test and observe the solution structure of optimization software and heuristic framework. These two modifications are illustrated in Figure 2 and Figure 3, respectively. Numbers of neighborhoods are also increased in different instances and up to 5 neighborhoods are created.

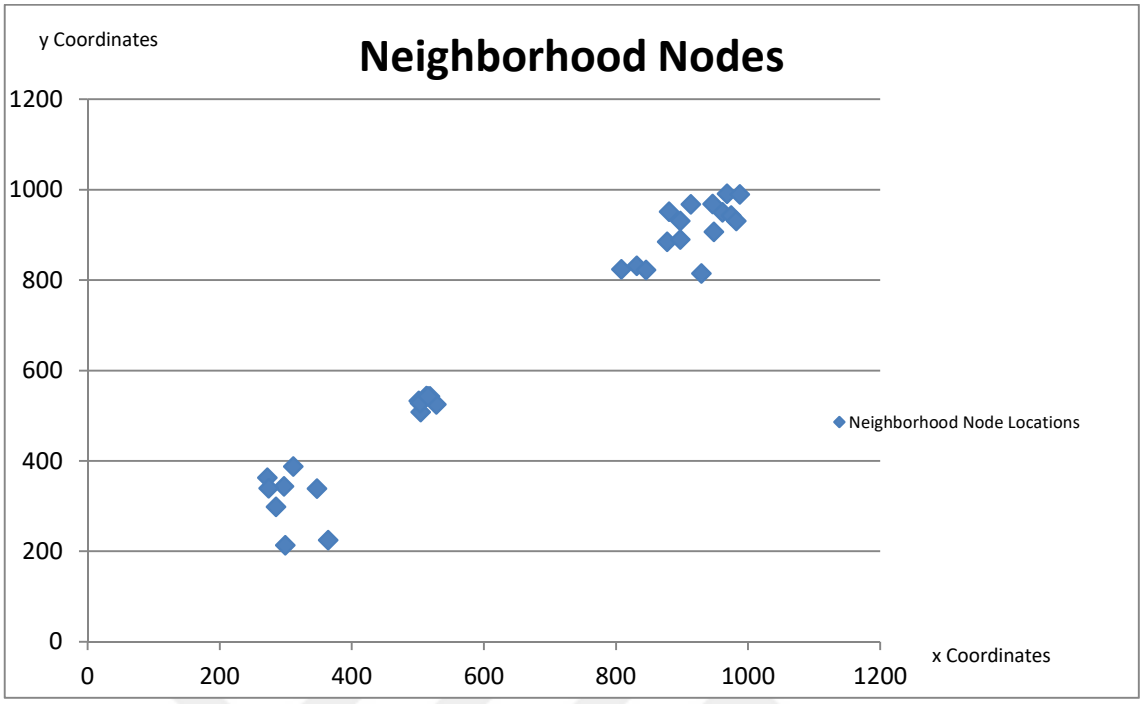

Figure 2. 3 Neighborhoods

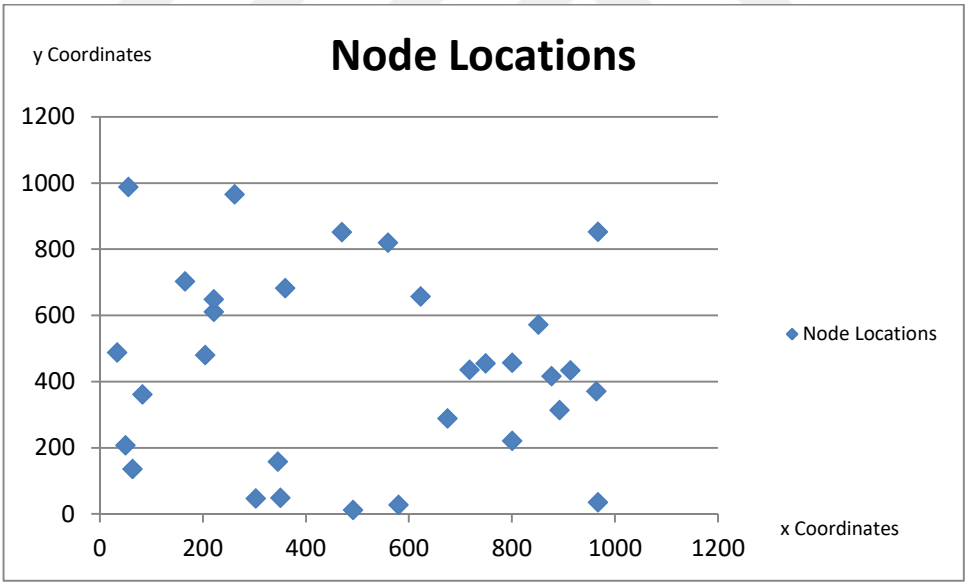

Figure 3. 3 Neighborhoods with Full Range Nodes

2. Distance Altering: Distance altering simply changes distances without changing the number or range of neighborhoods to measure the effect of different values in solution time and feasibility. Once the matrices are created, it is very easy to generate several versions.

While changing the distance matrix based on the above ways, cost vector is also manipulated by changing the ranges. The last input to our model is the cost vector showing photographing/digital monitoring cost of each node. If the model decides to take a photo at a particular node, the associated cost is automatically incurred in the objective function. We considered different ranges for these costs to alternatively determine optimal routes under diverse costs and to determine the dynamics where the model prefers taking photos instead of physically visiting the nodes. To illustrate, suppose that the costs of photographing in an instance are distributed between 1000 and 4000. Suppose further that the costs of photographing in another distance are distributed between 500 and 1000. It will definitely be preferable to consider photographing opportunities at a higher rate in [500-1000] case since there can be some nodes at which photographing process would cover more nodes than physically visiting that many nodes at the same cost. Thus, the dynamics of the photographing considerations should also be parametrically analyzed. In that sense, we divide the cost ranges accordingly and create problem instances to test our cost vector.

Based on these modifications, we create thirteen instances for numbers of nodes we solve in CPLEX. Each instance will be created based on the scheme given in Table 1.

Table 1. Instance Information

Instance Number	***Neighborhood Structure***	***Cost Vector Structure***
Instance 1A	No Neighborhood	CV ϵ [50,500]
Instance 1B	No Neighborhood	CV ϵ [250,1000]
Instance 1C	No Neighborhood	CV ϵ [1000,4000]
Instance 4A	NH1:x,y ϵ [0,100] NH2: x,y ϵ [900,1000]	CV ϵ [10,250]
Instance 4B	NH1: x,y ϵ [0,100] NH2: x,y ϵ [900,1000]	CV ϵ [250,750]
Instance 5A	NH1: x,y ϵ [0,250] NH2: x,y ϵ [400,600] NH3: x,y ϵ [750,1000]	CV ϵ [50,500]
Instance 5B	NH1: x,y ϵ [0,250] NH2: x,y ϵ [400,600] NH3: x,y ϵ [750,1000]	CV ϵ [50,2000]
Instance 6A	NH1: x,y ϵ [200,400] NH2: x,y ϵ [500,550] NH3: x,y ϵ [800,1000]	CV ϵ [10,400]
Instance 6B	NH1: x,y ϵ [200,400] NH2: x,y ϵ [500,550] NH3: x,y ϵ [800,1000]	CV ϵ [300, 850]
Instance 7A	NH1: x,y ϵ [0,200] NH2: x,y ϵ [200,400] NH3: x,y ϵ [400,600] NH4: x,y ϵ [600,800] NH5: x,y ϵ [800,1000]	CV ϵ [50,500]
Instance 7B	NH1: x,y ϵ [0,200] NH2: x,y ϵ [200,400] NH3: x,y ϵ [400,600] NH4: x,y ϵ [600,800] NH5: x,y ϵ [800,1000]	CV ϵ [500,1000]
Instance 8A	NH1: x,y ϵ [0,250] NH2: x,y ϵ [200,450] NH3: x,y ϵ [300,700] NH4: x,y ϵ [500,850] NH5: x,y ϵ [925,1000]	CV ϵ [50,500]
Instance 8B	NH1: x,y ϵ [0,250] NH2: x,y ϵ [200,450] NH3: x,y ϵ [300,700] NH4: x,y ϵ [500,850] NH5: x,y ϵ [925,1000]	CV ϵ [400, 800]

3.1.2 CPLEX Computational Results and Solution Times

Based on the problem instances created according to above procedures, a solution table with 8 nodes, 10 nodes, 12 nodes and 17 nodes is provided in Table 2. Note that the values are mean values for 5 runs for each node and instance number with different input data. Times are given in minutes, seconds and nanoseconds. Instances are named as number of nodes to solve and instance number to give as input. For example, 6-1A denotes a 6-node problem having input specified in instance 1A. All input data containing reachability matrices, cost vectors and distance matrices for CPLEX are stored in Excel files and read from Excel files by CPLEX. The nodes solved to optimality are given in Appendix A.

Table 2. CPLEX Results

Instance	*Average t*	*Instance*	*Average t*	*Instance*	*Average t*	*Instance*	*Average t*
8-1A	00:00,890	10-1A	00:00,910	12-1A	00:01,550	17-1A	00:01,580
8-1B	00:00,950	10-1B	00:00,930	12-1B	00:02,700	17-1B	00:01,150
8-1C	00:00,840	10-1C	00:01,310	12-1C	00:02,500	17-1C	00:01,150
8-4A	00:00,880	10-4A	00:01,220	12-4A	00:04,950	17-4A	00:52,120
8-4B	00:00,870	10-4B	00:01,600	12-4B	00:02,980	17-4B	01:45,720
8-5A	00:00,910	10-5A	00:01,180	12-5A	00:05,140	17-5A	03:46,700
8-5B	00:00,950	10-5B	00:02,550	12-5B	00:14,190	17-5B	02:57,520
8-6A	00:00,860	10-6A	00:02,590	12-6A	00:02,370	17-6A	26:23,000
8-6B	00:00,890	10-6B	00:02,440	12-6B	00:30,280	17-6B	09:34,720
8-7A	00:00,830	10-7A	00:01,150	12-7A	00:33,330	17-7A	44:31,820
8-7B	00:00,870	10-7B	00:01,550	12-7B	00:33,450	17-7B	16:25,170
8-8A	00:00,910	10-8A	00:05,230	12-8A	00:29,740	17-8A	00:17,580
8-8B	00:00,980	10-8B	00:03,550	12-8B	00:31,260	17-8B	00:02,980

Results differ in terms of solution times. It is expected and observed that an optimal route with the corresponding objective function value is always found for most of the problem instances defined. However, as seen in instances where neighborhoods are defined, reachability matrices contain more 1s, there is almost no reachability between neighborhoods, and we observe diverse solution times. We observe these diversities in instances 17-6 and 17-7. Even though instances of 17-5 and 17-6 are created based on same problem instance data and only the costs for photographing differ from each other, there is a large difference between their solution times. Therefore, we conclude that it is due to the structure of the convex hull that is created by the inputs, which is caused by reachability matrices and cost vectors together.

Other than the results for 17 nodes, we can draw the following conclusions and observe the following for overall outputs:

1. For small numbers of nodes, CPLEX solves the associated modified TSP in reasonable times, that is, it only takes seconds or couple of minutes to find the solution. The highest solution time is seen in instance 10-8A, which is around 5 seconds. As for 12, 15 and 17 nodes, solution times are slightly beginning to increase especially in instances with neighborhoods. In 12 nodes, solution times reach up to 35 seconds but they do not exceed 1 minute. It is also seen that solution times for high photo costs associated with instances with neighborhoods are higher than low photo cost instances. As we compute results for 15 nodes, solution times of instances with neighborhoods is again more than instances with no neighborhoods up to 5 times, as is the case with instance 15-6B against the first three instances. The longest time is observed in of the runs as 15 hours 13 minutes, corresponding to 913 minutes in instance 17-6A. Afterwards, CPLEX yielded more reasonable times and excluding this value yielded an average solution time of 26 minutes in 5 different runs. An average solution time of 16 minutes for instance 17-7B and 44 minutes for instance 17-7A are observed in 17 nodes computations. In Appendix A, we see other instances that belong to different node numbers. Even though in instances from 20-4 to 20-7 CPLEX showed memory errors, it seemed to approach to global optimal solutions as the gap in CPU diary always decreased and branching continued successfully.
2. To be able to determine the breaking point of our model, that is, to determine the node number after which our optimization software fails to find solutions due to incapacitated computer memory or very high solution times and to be able to reflect the increase in solution time depending on number of nodes, we select instance 1 to initiate another running process, and we run the model for the same instance by incrementally increasing the node number in each run. The results are as obtained in Table 3.

Table 3. Instance 3 (No Neighborhood) Node-Solution Time

Instance	z^{opt}	x^{opt}, y^{opt}*(shown by *)*	*t*
6	2290,341	0-5-4-1-3-6-2-0	00:08,950
7	2489,909	0-5-7-4-1-3-6-2-0	00:08,690
8	2514,839	0-5-7-4-1-8-3-6-2-0	00:08,840
9	2531,108	0-9-2-6-3-8-1-4-7-5-0	00:08,660
10	2573,539	0-2*-6-3-8-1-4-7-5-0	00:08,730
11	2658,788	0-2*-6-3-8-1-4-11*-5-0	00:08,940
12	2661,869	0-5-11*-4-1-8-3-12-6-22-0	00:09,000
13	2599,934	0-6-12-3-1-13*-5*-0	00:09,750
14	2715,279	0-5*-14-7-11-1-8-3-12-6-2-9-0	00:07,650
15	2851,261	0-6-12-3-15-13*-1-14-5*-0	00:07,920
16	2868,061	0-2*-6-12-3-15-13*-1-4-14-5-0	00:07,870
17	3055,662	0-5-14-4-1-13*-15-3-17-6-12-2*-0	00:09,080
18	3055,662	0-2*-12-6-17-3-15-13*-1-4-14-5-0	00:09,880
20	3381,822	0-5-14-4-19-1-13*-15-3-20-17-6-12-2*-0	00:12,980
22	3391,423	0-9-2-15-17*-3-20-8-22-1-19-11-7-14-5*-0	00:13,160
25	3803,031	0-2*-15-12-6-25-17-3-20-8-22-1-19-11-7-14-5*-10*-0	00:29,880
30	3958,999	0-23-29-28-12-6-25-17-3-20-8-22-1-1911-7-14-5*-26*-2*-0	00:36,190
35	4016,298	0-2*-27-28-31-25-17*-34-33*-32-22-1-19-11-7-14-5*-23*-0	01:34,260
36	3853,368	0-23*-5*-14-7-11-19-1-22-32-33*-34-17-36*-28-27-2*-0	01:40,710
37	3853,368	0-23*-5*-14-7-11-19-1-22-32-33*-34-17-36*-28-27-2*-0	01:34,570
38	3853,368	0-23*-5*-14-7-11-19-1-22-32-33*-34-17-36*-28-27-2*-0	03:26,000
39	3893,055	0-23*-5*-39-11-14-7-19-1-22-32-33*-34-17-36*-28-27-2*-0	09:05,430
40	3893,055	0-23*-5*-39-11-14-7-19-1-22-32-33*-34-17-36*-28-27-2*-0	01:25:13,500
41	3893,055	0-23*-5*-39-11-14-7-19-1-22-32-33*-34-17-36*-28-27-2*-0	33:53,970
42	3893,055	0-23*-5*-39-11-14-7-19-1-22-32-33*-34-17-36*-28-27-2*-0	01:11:05,650
43	4508,152	0-2*-27-28-36*-17-34-33*-43-32-22-1-19-7-14-11-39-5*23*-0	02:00:47,200
44	4508,152	0-2*-27-28-36*-17-34-33*-43-32-22-1-19-7-14-11-39-5*23*-0	02:03:32,400
45	4508,152	0-2*-27-28-36*-17-34-33*-43-32-22-1-19-7-14-11-39-5*23*-0	09:21:30,240
50	*Out of memory error*		

Up to 18 nodes, CPLEX finds the solution in less than 10 seconds. The times are close to each other and since there are no neighborhoods, finding the solution does not take as many amount of time as the other instances with neighborhoods do. Up to 39 nodes, CPLEX does not take more than 10 minutes to find the optimal solution, however, if there are 40 nodes or more, time increases tremendously even şf the optimal route and optimal solution remains the same. In 41 nodes, the time to find the solution again decreases, but in 42 nodes, it again increases by around 50%. This situation is explained again by different reachability matrices since all other cost vector and

distance matrix inputs remain the same. After 45 nodes we decided to solve the problem for 50 nodes, and received a memory error. The time taken to solve the different instances of our problem is shown in Figure 4. It is observed that as the number of nodes increases, the rate at which the time to solve the problem increases gets also larger. It should be noted that since these nodes belong to one instance, it is possible to have different solution times especially in larger numbers of nodes in different instances. Here we only deep delve into the speed for becoming NP-Hard for a non-neighborhood case, which has always shown smaller solution times as compared to neighborhood cases.

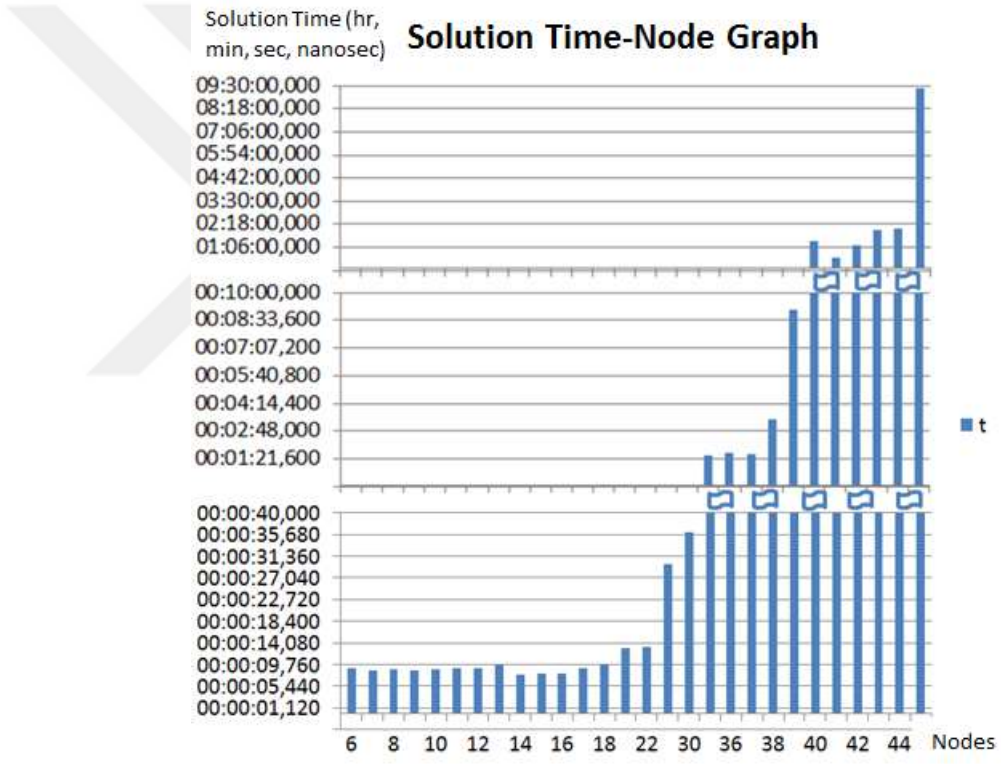

Figure 4. Solution Times versus Nodes

3.2 NP-Hardness, Lower Bounds and Need for a Heuristic Algorithm

The problem is a variant of TSP where a complete route should be found for the agent. If the reachability matrix has 1s only in its diagonal, it means that each node can be reached only from itself. In other words, each and every node should then be vsited physically, turning the problem into general TSP. Since TSP is NP-Hard, our problem is also NP-Hard and the time required for finding solutions for larger instances gets increasingly higher. To illustrate, we run CPLEX for 44 nodes and 45 nodes in Table 3, and even though the optimal route is exactly the same, the complexity of 45 nodes causes the problem to take a time of 9 hours and 21 minutes, which is 7 hours more from the time for 44 nodes. We ran the model for 45 nodes once again to see whether the instance will take the same amount of time, and after 5 hours we see that the problem is at the same place by looking at the gap amount and the best solution value obtained so far.

The high times to solve the problem create the motivation to design our heuristic algorithm. We seek to obtain relatively quicker solutions that are within acceptable percentages of the optimal solution. If we know the optimal solution of a particular instance, we define term “acceptable” by looking at the gaps between the optimal solution and the solution obtained by our heuristic algorithm. If an optimal solution of an instance is not known, however, then lower bounds should be found for these instances to be able to do comparisons with our heuristic algorithm. After 50 nodes, we tried our heuristic in all instances for 30 nodes, 50 nodes, 100 nodes and eventually for 400 nodes, for which we obtained lower bounds by a separate mathematical model. The model is defined below.

3.2.1 Notation

Parameters:

We use the same parameters from Section 1.2.1.

Decision Variables:

In addition to decision variables we used in our original model, we introduce an additional variable to denote whether a node is physically visited or not.

$\mathbf{z_j}$ $\begin{cases} 1 \text{ if the node j is physically visited} \\ 0 \text{ otherwise} \end{cases}$ $\forall i \in N$

3.2.2 Objective Function

Our objective is the same as in our modified TSP model. We aim to minimize the total photographing and visiting cost as we cover all nodes, shown in **(15):**

$$\sum_{i=0}^{n}\sum_{j=0}^{n} d_{ij}x_{ij} + \sum_{i=0}^{n} c_i y_i \quad \textbf{(15)}$$

3.2.3 Constraints

Node Coverage: As previously stated, all nodes should be covered either by taking a photo or directly visiting them, assured in **(16):**

$$\sum_{i=0, i\neq j}^{n} r_{ij}y_i + z_j \geq 1 \qquad \forall j \quad \textbf{(16)}$$

Photographing Requires Physical Visit: If a node is used for taking a photo, then that node should be visited by the agent itself. This is satisfied by **(17):**

$$z_i \geq y_i \qquad \forall i \quad \textbf{(17)}$$

Physical Visited Nodes Should Be Arrived and Left: To formulate this constraint, **(18)** is used.

$$2z_i = \sum_{j=0, j\neq i}^{n} (x_{ij} + x_{ji}) \quad \textbf{(18)}$$

Visits Imply Edges: If a node is visited, necessary route to that node consisting of the edges should be formed, shown in **(19):**

$$\sum_{i}^{n}\sum_{j}^{n} x_{ij} = \sum_{i}^{n} z_i \quad \textbf{(19)}$$

It is known that the spanning forest we obtain by solving the above mathematical model provides a lower bound for the problem instances which we cannot solve to optimality with the resources on hand. The lower bounds for 30 nodes, 50 nodes and 100 nodes will be used in Section 3.4 where we compare the optimal solutions as well as lower bounds with the solutions of our algorithm. We might also have tried to connect all node clusters in our initial solution to obtain complete MST, however, it would not then be sure that we get a lower bound, as the tree we obtain is not necessarily a minimum spanning tree. In fact, it is possible to find better solutions than what we could call a tree, which, in this case, means we cannot find the lower bound.

Another method to find the lower bounds is using CPLEX solver's solution diary to track the gap and best solution information obtained so far. Even though in different numbers of nodes this process cannot be pursued due to tremendous waiting time, it may still provide quality lower bounds considering also the gap, and these bounds can be compared to our heuristics solutions. In some instances where we are able to track the information, we waited until we are within 5-10% of gap, and we took the solution obtained so far as the lower bound. This makes sense because the lower bound we obtain by minimum spanning forest ignores connections between these forests, and since these forests, when connected, incur more cost, these costs are not included in our minimum spanning forest, which makes the lower bound relatively 'low'.

3.3 The Proposed Algorithm

Our algorithm is a combination of a route construction algorithm for TSP and a modified SCP, which are solved in a loop to first select a group of solutions and then try to find the best solution among them. The pseudocode of our algorithm is given as follows:

Step 0. Initialization
Read coordinates, cost vectors and reachability matrix from Excel
Take the reachability matrix and give it as input to modified SCP

Step 1. Determination of Base Node Set
Solve modified SCP to determine the nodes to cover all nodes in the network in the least cost
Create the base list BL_{SCP} such that all photo nodes in SCP are ordered in increasing order
If the starting node 0 is already in $L_{SCP,}$ go to Step 2
Else add the starting node 0 to L_{SCP} and go to Step 2

Step 2. For each L_{SCP} do
Approximate the associated TSP by Christofides Algorithm to obtain total route travel cost
Add z_{SCP} and z_{TSP} to obtain the cumulative cost z_{CUM} of photo-modified TSP
If no more exclusion is possible in BL_{SCP}, go to Step 4
Else go to Step 3

Step 3. For *each photo node* in BL_{SCP} do
Exclude the node from BL_{SCP}, solve SCP with remaining nodes to obtain new L_{SCP}, go to Step 2

Step 4. Obtain the best z_{CUM}
Record the route, cost and solution time of all other iterations
Stop

Here, BL_{SCP} denotes the list obtained by solving the modified SCP for the first time, and the main loop follows this list. L_{SCP} is the list obtained by solving SCP in each re-start of the loop. The values z_{SCP} and z_{TSP} refer to objective function values of SCP and TSP, respectively. The value z_{CUM} is the total cost associated with the corresponding step of the algorithm, best of which is reported after the algorithm is complete.

To illustrate our algorithm, we will solve instance 12-4A and compare the results. This is one of the instances having two neighborhoods, so it is reflected in the reachability matrix. The coordinates and also the cost vector are assigned random numbers based on the input structure. Our algorithm is then run on this instance.

Step 0: All input data is read by our algorithm, taken into memory by Excel files. The modified SCP differs from the original SCP in that it also incorporates physical visits to nodes in order to cover them, shown below:

$$\textbf{min} \quad \sum_{i=0}^{n}(c_{0i} \times z_i) + \sum_{i=0}^{n} f_i \times y_i \tag{20}$$

subject to

$$\sum_{i=0}^{n}(r_{ij} \times y_i) + z_j \geq 1 \qquad \forall \mathrm{j} \tag{21}$$

$$z_0 = 0 \tag{22}$$

The second part in **(20)** facilitates physical visits with the cost of arriving at that node from the center node. Even though an agent does not necessarily arrive at a node from the center node in practice, it is assumed to do so for computational purposes. **(21)** ensures that all nodes are covered either by photo taking or physical visits while **(22)** ensure that the center node is not physically visited during the tour.

Step 1: Accordingly, SCP uses reachability matrix and solves the resulting problem. The problem is straightforward and our heuristic yields the base node set as follows:

Base Node Set: 10*, 7*, 3* where * denotes a photo taking node

Accordingly BL_{SCP} becomes: 3*, 7*, 10*

Since the node 0 is initially not in the list, we add it to BL_{SCP}.

Step 2 and 3: We apply Christofides Algorithm in the following sub-steps to approximate TSP tour for L_{SCP}:

1. Insert Basic Information
2. Find Minimum Spanning Tree
3. Find Odd Degree Vertices
4. Minimum Weight Matching
5. Find Euler Cycle Path
6. Find TSP Cycle Path

We obtain the solution route as 0-10*-7*-3*-0 with z_{CUM} = 2841.14. Next the algorithm searches for any exclusion possibilities. Exclusion means elimination of a photo taking node from the base solution set BL_{SCP}, which is followed by re-solving the problem under the new constraints. In our case, we have three photo-taking nodes, meaning the loop will be called three times to solve the problem in absence of these photo taking nodes. First, the algorithm removes node 3, as given in BL_{SCP} and resolves the modified SCP. We obtain new L_{SCP} as: 5*, 7*, 11* with z_{CUM} = 2901.17, which is greater than our Base Node Set solution. Secondly, the algorithm removes node 7, allows node 3 again and re-solves the problem. We obtain L_{SCP} as: 5*, 10*, 12*, with also node 0 and with z_{CUM} = 2914.28, again greater than initial z value. Lastly, the algorithm removes node 10 and solves the problem again, obtaining L_{SCP} as 5*, 7*, 12* with z_{CUM} = 2987.032. Since there is no exclusion alternative left, the algorithm proceeds with Step 4.

Step 4: The best solution obtained is reported with z_{CUM} and the associated tour:

z_{CUM} = 2841.14 with tour 0-10*-7*-3*-0

3.4 Comparing the Greedy Heuristic to CPLEX Results

3.4.1 Objective Function Values

Results are given in terms of solution quality in Table 4, measured by the gap between the optimal objective function value or the lower bound:

$$\frac{(z_{heur} - z_{opt})}{z_{opt}}$$

where for instances for which we do not have a solution we simply replace the term z_{opt} by $z_{\text{lower bound}}$.

Table 4. Solution Gaps for 6, 8, 12 and 17 Nodes

Instance	Gap	Instance	Gap	Instance	Gap	Instance	Gap
6-1A	0,42	**8-1A**	0,29	**12-1A**	0,64	**17-1A**	0,38
6-1B	0,52	**8-1B**	0,60	**12-1B**	0,93	**17-1B**	0,66
6-1C	0,51	**8-1C**	0,57	**12-1C**	0,98	**17-1C**	1,10
6-4A	0,01	**8-4A**	0,07	**12-4A**	0,05	**17-4A**	0,04
6-4B	0,31	**8-4B**	0,50	**12-4B**	0,34	**17-4B**	0,28
6-5A	0,17	**8-5A**	0,19	**12-5A**	0,09	**17-5A**	0,29
6-5B	0,21	**8-5B**	0,75	**12-5B**	0,24	**17-5B**	0,36
6-6A	0,08	**8-6A**	0,09	**12-6A**	0,10	**17-6A**	0,18
6-6B	0,32	**8-6B**	0,23	**12-6B**	0,38	**17-6B**	0,71
6-7A	0,13	**8-7A**	0,27	**12-7A**	0,19	**17-7A**	0,29
6-7B	0,23	**8-7B**	0,70	**12-7B**	0,65	**17-7B**	1,18
6-8A	0,15	**8-8A**	0,26	**12-8A**	0,14	**17-8A**	0,28
6-8B	0,45	**8-8B**	0,86	**12-8B**	0,90	**17-8B**	0,69

The gaps of instance B of all nodes are higher than instance A. This means that when we increase the photographing costs of nodes, our algorithm begins to give worse solutions in terms of objective function value. Since our algorithm depends on photo-taking nodes in the beginning of its structure by solving a preliminary set covering problem, increasing photographing costs results in higher objective function values in exchange for speed for obtaining a solution.

We obtain solutions within 1-10% of the optimal solution in instances 4, 5, 6 and 7 of all nodes. Even though instance B solutions of the nodes are higher than those of

instance A, they go in parallel with instance A solutions, and there is consistency between their solution gaps. The gap averages for these nodes are provided in Table 5.

Table 5. Average Performances of 6, 8, 12, 17 Nodes

Node	Instance A	Instance B	Instance C
6	0.16	0.34	0.51
8	0.19	0.60	0.56
12	0.20	0.57	0.97
17	0.24	0.64	1.11

It is seen that as the numbers of nodes increase while increasing photographing costs, z value gaps also increase. Even if the algorithm provides good quality solutions for some instances, it performs increasingly worse as increasing photographing costs makes it preferable to physically visit the nodes rather than taking photos of them. Similarly, solution gaps for 20, 30, 50 and 100 nodes are provided in Table 6.

Table 6. Average Performances of 20, 30, 50 and 100 Nodes

Node	Instance A	Instance B	Instance C
20	0.62	0.98	1.05
30	0.67	0.91	0.55
50	0.51	0.55	0.97
100	0.47	0.60	1.67

The graphs of A and B instances again follow a consistent pattern, and the difference between them gets closer, however, this reduction is caused by two very important factors:

1. The average gaps of A instances increase at higher rates than B instances, which reduces the gap between two curves.
2. Since we are computing lower bounds in large numbers of nodes in order for a reasonable comparison and since these bounds are lower than their associated optimal objective function values, the gaps eventually increase, leading to reduced solution quality.

To understand the effects of photographing costs and also the reachability matrix on z value gaps, we re-computed the instances as to solve them to optimality and also to obtain heuristic solutions. We derive two conclusions about the solution quality being affected by photographing costs and reachability structure:

1. Under the condition that all input data remain the same, as photographing costs are decreased, the algorithm performs better because each candidate node for taking photos will then incur much less values to optimal objective function value. This is best illustrated in instances with multiple photo taking nodes and also high z value gaps. We re-solve 10 instances randomly by stepwise reduction of photographing costs and obtain the results shown in Table 7.

Table 7. Average Gaps under Decreasing Photo Costs

%Reduction	Gap
0	1,12
5	1,08
20	0,83
40	0,72
60	0,60
80	0,58
90	0,59

There is a decrease in z value gaps as the photo taking costs are decreased, however, after a certain value problem structure changes and the lower bound finder we developed in CPLEX provides different lower bounds, which, depending on the photo-taking nodes and also the distances, change at different rates. This in turn causes the gap increase to 0.71 and 0.76 in the last two cases. It should also be noted for cases having many photo nodes in the optimal solution that the amount of reduction in z value gap is much greater since any reduction in photo costs directly impacts objective function value. In addition, z value gap has a lower bound due to TSP part of the problem, so even if the photo costs are set to zero, MST lower bounds will be smaller than TSP routes that we obtain by our heuristic.

2. Increase in reachability promotes the algorithm's solutions and enhances solution quality in that the feasibility regions of instances increasingly begin to contain the extreme points where photo nodes that cover more nodes exist. We illustrate this concept best by resolving 10 instances again, where the optimal solution simply visits all nodes without taking any photos or takes at most 1-2 photos in the optimal solution, and our heuristic algorithm takes photos at multiple nodes, leading to an average z value gap of 0.26. We increase number of visual connections of each node by certain numbers and randomly, then investigate the change in z value gaps by resolving the problem, whose results are shown in Table 8.

Table 8. 30-1B Reachability-Gap Changes

Increase in Reachability	Gap
0	0,26
1	0,15
2	0,12
4	0,07
6	0,10
8	0,11

As is the case in photo costs, z gaps decrease significantly at first, and they increase after a certain point because of the change in feasibility region and also the stability of our heuristic to take into account these reachability changes. This graph also shows that our heuristic is able to produce solution within 5% of the optimal solution even in cases where photo costs are higher. The reachability issue that we explain in the very beginning to shed light on our selection of reachability function can now be exploited better. Decreasing the power of the function significantly enhances our algorithm, which makes sense in real life applications because if there is a hill or certain point higher in altitude than its neighbor points, that point will have visual access to all nearby points, which makes it reasonable to reduce the power of the reachability function.

So far we looked at reachability and photographing aspects separately. We now illustrate these concepts together in instance 30-1B, where we decrease photo costs by percentages while increasing reachability. We use the same increasing and decreasing structure and obtain the results summarized in Table 9.

Table 9. 30-1B Photo Cost and Reachability Impact on Solution Quality

% Reduction in Photo Costs	Increase in Reachability	Gap	% Reduction in Photo Costs	Increase in Reachability	Gap	% Reduction in Photo Costs	Increase in Reachability	Gap
0%	0	0,28	40%	0	0,34	80%	0	0,41
	1	0,13		1	0,22		1	0,06
	2	0,14		2	0,18		2	0,09
	4	0,06		4	0,23		4	0,24
	6	0,08		6	0,24		6	0,30
	8	0,13		8	0,11		8	0,18
20%	0	0,35	60%	0	0,47			
	1	0,31		1	0,13			
	2	0,26		2	0,10			
	4	0,28		4	0,18			
	6	0,27		6	0,30			
	8	0,14		8	0,12			

Reader should note that even though we judge based only on one instance, more instances should be investigated for better comparison. The results show that it is not necessarily true that we obtain better results as we provide more reachability and lower photo costs, rather, it causes more time to evaluate each alternative and thus increases overall solution time. In addition, although CPLEX solves each problem to optimality and obtains better solutions as we give better input, our algorithm is not able to find better solutions in some cases, as seen in 60% and 80% decreased photo costs, in particular the cases in which reachability is increased by 4 and 6 nodes. We already stated that these increases in z gaps are due to preliminary SCP that our algorithm solves. Even if the z gaps increase by some percent, however, the overall performance of the algorithm reaches 6%, 14%, 11%, 10% and 6% in respective photo cost reductions. Solution changes under both sensitivity analyses are shown in Figure 5.

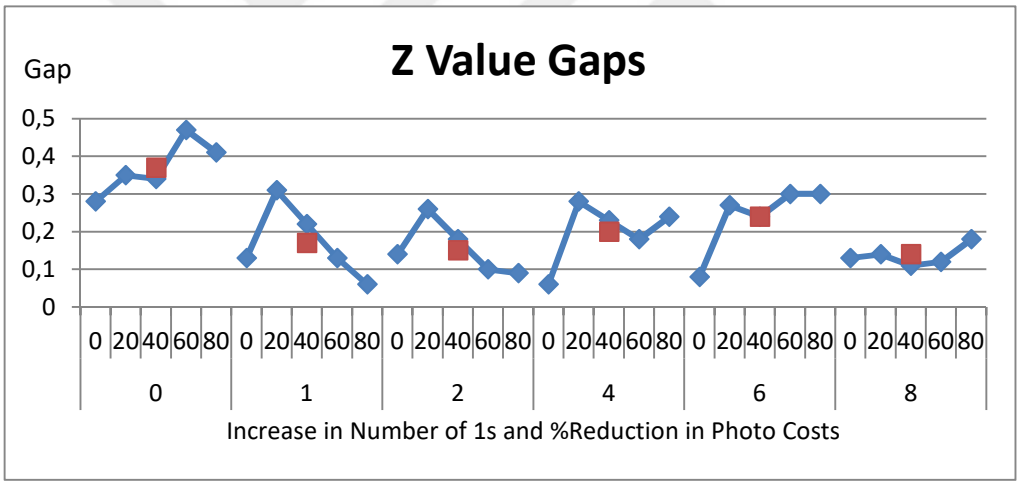

Figure 5. 30-1B Z Gaps under Reachability and Photo Cost Changes

In all cases, the largest gap does not exceed 0.47 that we obtained in initial solution. As we increase number of 1s in reachability matrix, average gaps first decrease significantly, then increase and then decrease again. In different numbers of 1s, z gaps increase when we decrease photo costs by 20%. They then increase or decrease depending on the convex hull we create by changing input values.

3.4.2 Solution Times

Reduction in solution times is the other objective in constructing our heuristic algorithm. During the solution process it is seen that when there are neighborhoods in reachability part of the problem, even 20 nodes show high level of complexity, making it necessary to solve the problem by heuristics. This is also observed in Table 9, where both decreasing photo costs and increasing reachability causes more time to solve the problem to optimality. Our heuristic algorithm, on the other hand, is not affected in terms of solution times and it provides solutions in less than 1 second. As numbers of 1s in reachability matrices are increased, a photo node increasingly begins to cover more nodes, and as the costs of photographing is reduced, photo nodes begin to become of alternatives to take photos. Both changes lead to a more complex structure since they enlarge the feasible region and number of extreme points. This in turn causes solution times to increase exponentially.

CPLEX is able to solve instances up to 17 nodes in almost no time, after which it increasingly spends time finding the optimal solution especially in neighborhood instances. Instances 20-4, 20-5 and 20-6 cannot be solved by CPLEX in reasonable times. Instance 20-7A and 20-7B takes around 5 hours while their no neighborhood versions take less than 10 seconds to solve. Next we give our algorithm's solution times for 17 and 20 nodes in Figure 6.

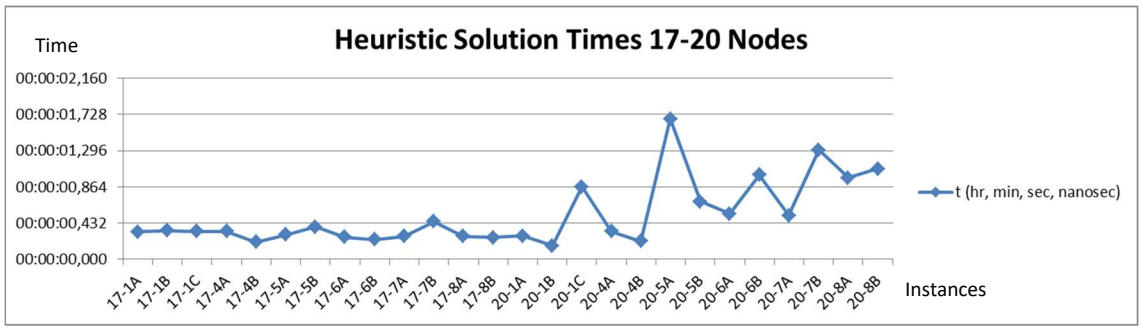

Figure 6. Heuristic Solution Times for 17-20 Nodes

The largest time is around 1.5 seconds for 20-5A, except which all instances are solved between 0.4 and 1.2 seconds. Our algorithm significantly reduces the amount of time to reach good enough solution for the problem. For 30 nodes and more, solution times differ between nanoseconds and 3 seconds up to 400 nodes.

CHAPTER IV

CONCLUDING REMARKS

Monitoring and photographing large areas for security, safety or other specific reasons requires careful and detailed planning since decision makers always have limited time and other resources. In a given amount of time, an area within a building, a zone within a region or even a region itself should be monitored to take precautions against different emergencies. To prevent fires in a bank or school, to set up cameras or alarms for burglary, to conduct periodical controls in forests for fire prevention or to monitor borders for intruders and terror attacks are among different motivations of field scanning.

For many years, Traveling Salesman Problem is approached in different ways by different researchers, and tremendous numbers of studies are created to effectively and efficiently solve the problem. In this study we aimed to provide optimal as well as heuristic solutions for a modified TSP problem, where agents do not have to go to each node physically, rather, they are able to reach some nodes from the current node visited, and this saves time and money. We provided a mathematical model for this problem, showed that it is a combination of TSP and SCP and is NP-Hard, provided a greedy heuristic algorithm to solve the problem in acceptable times and compared the results. We have seen that the problem structure is dependent on reachability and photo costs, and we tried to investigate these structures in various numbers of nodes and input schemes.

All in all, our algorithm is able to provide good solutions even within 1% of the optimal solution. We have observed that solution quality may change depending on the reachability probability function, and since our heuristic algorithm initializes by solving a preliminary SCP, it approaches to solutions with multiple photo taking nodes, thus failing to provide good enough solutions for the associated problem instance. Nevertheless, the algorithm is capable of providing solutions for each instance ranging from 6 to 400 nodes, and it take only nanoseconds to obtain a good solution. The algorithm provides solutions within %5-10 of the optimal solution on the average and performs flawlessly on digital framework.

There are several pitfalls of the algorithm, among which preliminary SCP's photo costs worsen the solution. Since a physically visited node should incur a cost

based on distance, and since we cannot possibly know the physically visited node's distance cost before it is visited, we simply made the assumption that any physically visited node incurs the cost of its distance from the center node. This assumption leaves the algorithm's performance for probabilistically changing levels due to random number generation in distances. If the center node is close to physical visit node, then there is less cost incurred to objective function value, and if there is a large distance between the visit node and the center node, the algorithm is not able to measure it correctly to select it. Suppose that in a good enough solution that far distanced node incurs a little cost because its neighborhood nodes are very close. However, since the SCP of the algorithm cannot measure it directly, it may automatically eliminate that node and may start with a worse solution to start searching. So in future studies it should be the objective of us to handle the SCP part of the algorithm so that photographing as well as physical visits are given the correct costs and the algorithm does not miss any solution in the beginning.

Another point of improvement can be made in the main loop of the algorithm where it removes any photo nodes and resolves the problem to look for better solutions. We consider the photo nodes only the Base Node Set, however, in any sub-set created by the removal of photo nodes, there can be other photo nodes not initially considered, and removal/addition of them may lead to better solutions. So our algorithm may be improved to take into account the multiple level photo node improvements instead basing on only the initial set.

APPENDIX A. CPLEX Solutions for Other Nodes

Instance	z^{opt}	x^{opt}, y^{opt}(shown by *)	*t*
6-1A	2419,645	0-3-1-4-6-2-5-0	00:01,410
6-1B	2419,645	0-3-1-4-6-2-5-0	00:01,260
6-1C	2419,645	0-3-1-4-6-2-5-0	00:01,160
6-4A	2759,711	0-2-6*-3-1-0	00:01,090
6-4B	2811,973	0-1-3-4-6-5-2-0	00:00,960
6-5A	2458,802	0*-3-6-5-4-1-0	00:00,910
6-5B	2577,136	0-2-1-4-5-6-3-0	00:01,030
6-6A	1771,107	0-2-1-4-3-5-6-0	00:01,120
6-6B	1771,107	0-2-1-4-3-5-6-0	00:00,960
6-7A	2796,662	0-4-5*-3-2-1-0	00:01,140
6-7B	2820,302	0-1-2-3-5-6-4-0	00:00,890
6-8A	2353,23	0-2-5-6-4-3-1-0	00:01,110
6-8B	2353,23	0-2-5-6-4-3-1-0	00:01,110
7-1A	2275,806	0-5-6-7-2-1-3-4-2	00:01,140
7-1B	2275,806	0-5-6-7-2-1-3-4-1	00:00,170
7-1C	2275,806	0-5-6-7-2-1-3-4-0	00:01,140
7-4A	2597,465	0-2-3-6-4*-1-0	00:00,810
7-4B	2636,868	0-2-3-4-6-7-5-1-0	00:01,910
7-5A	2624,015	0-1-3-5-6-7-4-2-0	00:00,970
7-5B	2624,015	0-1-3-5-6-7-4-2-0	00:00,940
7-6A	2148,844	0-1-4*-6-5-7-0	00:00,880
7-6B	2195,763	0-1-4-7-6-5-3-2-0	00:00,910
7-7A	2329,696	0-5-7-6-4-2*-0	00:00,940
7-7B	2335,472	0-1-5-7-6-4-3-2-0	00:00,960
7-8A	2713,544	0-1-3-4-7-6-5-2-0	00:00,890
7-8B	2713,544	0-1-3-4-7-6-5-2-0	00:00,960
9-1A	2444,961	0-6-8-9-5-2-1-3-7-4-0	00:00,910
9-1B	2444,961	0-6-8-9-5-2-1-3-7-4-0	00:00,890
9-1C	2444,961	0-6-8-9-5-2-1-3-7-4-0	00:00,870
9-4A	2593,477	0-4*-8-7*-0	00:01,050
9-4B	2775,331	0-9-5-6-8-7-3-4-1-2-0	00:01,110
9-5A	2590,607	0-2-5-7*-3-4-1-0	00:01,000
9-5B	2646,656	0-2-5-6-8-9-7-3-4-1-0	00:01,340
9-6A	2157,404	0-5-9*-3-4-2-1-0	00:01,670
9-6B	2186,642	0-5-6-8-9-7-3-4-2-1-0	00:01,050
9-7A	2623,664	0-3-5-7*-6-4-2-1-0	00:01,050
9-7B	2642,353	0-3-5-6-8-7-9-4-2-1-0	00:01,110
9-8A	2366,091	0*-4-3-5-9-8-7-6-0	00:01,020
9-8B	2397,169	0-1-2-6-7-8-9-5-3-4-0	00:01,080
15-1A	3129,343	0-2-7-1-5-13-12-6-9-15-4-14-8-3-10-11-0	00:15,580
15-1B	3129,343	0-2-7-1-5-13-12-6-9-15-4-14-8-3-10-11-0	00:06,970
15-1C	3129,343	0-2-7-1-5-13-12-6-9-15-4-14-8-3-10-11-0	00:06,000
15-4A	2664,904	0*-13*-0	00:04,440
15-4B	2793,892	0-3-14-15-11-9-12-10-13-6-7-8-2-4-5-1-0	00:06,620
15-5A	2720,731	0-3-2-9-6*-11*-12-7-10-4-1-0	01:34,620
15-5B	2870,999	0-3-2-9-5-6-8-11-13-15-14-12-7-10-4-1-0	00:45,110
15-6A	2134,442	0-2-1-9-7-8-6-12*-11-13*-5-4-3-0	02:35,650
15-6B	2249,49	0-2-1-9-7-8-6-12-14-11-10-15-13-5-4-3-0	05:38,960
15-7A	2954,179	0-4-7-11-13-15-10*-8-6-5-1*-0	01:42,450
15-7B	2998,71	0-4-7-9-11-13-15-14-10-12-8-6-5-2-3-1-0	00:32,780
15-8A	2747,465	0-1-2-3-8-12-11-15-14-13-10-7-9-5-6-4-0	00:57,470
15-8B	2747,465	0-1-2-3-8-12-11-15-14-13-10-7-9-5-6-4-0	00:17,430
20-1A	4036,494	0*-20-17-18-19-9-10-3-13-1-7-5-4-14-16-2-0	00:06,480
20-1B	4036,494	0*-20-17-18-19-9-10-3-13-1-7-5-4-14-16-2-0	00:05,040
20-1C	4071,03	0-15-6-2-8-16-14-4-5-7-1-13-3-10-9-19-18-17-20-11-12-0	00:03,050
20-7A	2332,931	0-3-4-5-6-7-12-11-9-20*-16-13-15-14-10-8-1-2-0	05:16:33,147
20-7B	2848,931	0-3-4-5-6-7-12-11-9-20*-16-13-15-14-10-8-1-2-0	04:57:00,040

Instance	z^{opt}	x^{opt}, y^{opt}*(shown by *)*	*t*
20-8A	2989,167	0-4-3*-7-11-13-15-16-14-17*-10-12-5-6-2-0	00:17:28,070
20-8B	3194,28	0-1-4-3-7-6-5-12-10-11-13-15-16-14-18-19-20-17-9-8-2-0	01:51:26,310
30-1A	3327,206	0*-18*-10*-27*-11-7*-19*-26-20-6-23-0	00:00:06,670
30-1B	4400,607	0-22-23-8-16-13-30-18-27*-12-15-4-19-26-29-20-9-1-6-0	00:00:08,610
30-1C	4982,442	0-6-1-9-20-29-26-4-19-28-15-12-2-25-10-27-5-21-3-14-24-8-30-13-16-8-23-22-0	00:00:08,660
50-1A	4673,481	0-5*-18*-13-6-20-9-49-34-16*-45-32-1-24-11*14-46-22-19*-17-26*-12*-0	00:04:24,350
50-1B	5707,910	all nodes visited except 13 and 5's coverages	00:02:00,380
50-1C	5786,210	all nodes visited	00:03:12,430

BIBLIOGRAPHY

[1] R. Necula, M. Breaban, and M. Raschip, “Performance Evaluation of Ant Colony Systems for the Single-Depot Multiple Traveling Salesman Problem,” Springer International Publishing, 2015, pp. 257–268.

[2] G. B. Dantzig, D. R. Fulkerson, and S. Johnson, “Solution of a Large-Scale Traveling-Salesman Problem,” *J. Oper. Res. Soc. Am.*, vol. 2, no. 4, pp. 393–410, 1954.

[3] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, and M. R. Reddy, “Chapter 1 Applications of network optimization,” *Handbooks Oper. Res. Manag. Sci.*, vol. 7, no. Network Models, pp. 1–83, 1995.

[4] M. Jünger, G. Reinelt, and G. Rinaldi, “Chapter 4 The traveling salesman problem,” *Handbooks Oper. Res. Manag. Sci.*, vol. 7, no. Network Models, pp. 225–330, 1995.

[5] J. A. Ratches, “Review of current aided / automatic target acquisition technology for military target acquisition tasks technology for military target acquisition tasks,” *Opt. Eng.*, no. Army Research Lab, 2016.

[6] R. M. Karp, “Reducibility among combinatorial problems,” *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. pp. 219–241, 2010.

[7] Y. Nobert, “Exact Algorithms for the Vehicle Routing Problem,” *North-holl. Math. Stud.*, vol. 132, pp. 147–184, 1987.

[8] J. K. Lenstra and A. H. G. R. Kan, “Complexity of vehicle routing and scheduling problems,” *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[9] C. H. Papadimitriou, “The Euclidean travelling salesman problem is NP-complete,” *Theor. Comput. Sci.*, vol. 4, no. 3, pp. 237–244, 1977.

[10] J. . Beasley and P. . Chu, “A genetic algorithm for the set covering problem,” *Eur. J. Oper. Res.*, vol. 94, no. 2, pp. 392–404, Oct. 1996.

[11] M. Solar, V. Parada, and R. Urrutia, “A parallel genetic algorithm to solve the set-covering problem,” *Comput. Oper. Res.*, vol. 29, no. 9, pp. 1221–1235, 2002.

[12] U. Aickelin, “An indirect genetic algorithm for set covering problems,” *J. Oper. Res. Soc.*, vol. 53, no. 10, pp. 1118–1126, Oct. 2002.

[13] A. Caprara, P. Toth, and M. Fischetti, “Algorithms for the Set Covering Problem,” *Ann. Oper. Res.*, vol. 98, no. 1/4, pp. 353–371, 2000.

[14] A. Caprara, M. Fischetti, and P. Toth, “A Heuristic Method for the Set Covering Problem,” *Oper. Res.*, vol. 47, no. 5, pp. 730–743, Oct. 1999.

[15] D. Applegate, R. Bixby, V. Sek, and C. Atal, “Finding Cuts in the TSP (A preliminary report),” 1995.

[16] M. Grötschel and L. Nemhauser, George, “George Dantzig’s contributions to integer programming,” *Discret. Optim.*, no. 5, pp. 168–173, 2008.

[17] S. J. G. Dantzig, R. Fulkerson, “SOLUTION OF A LARGE-SCALE TRAVELING-SALESMAN PROBLEM,” *J. Oper. Res. Soc. Am.*, vol. 2, no. 4, pp. 393–410, 1954.

[18] R. E. Gomory, “Outline of an algorithm for integer solutions to linear programs,” *Bull. Am. Math. Soc.*, no. 64, pp. 275–278, 1958.

[19] R. Bellman and Richard, “Dynamic Programming Treatment of the Travelling Salesman Problem,” *J. ACM*, vol. 9, no. 1, pp. 61–63, Jan. 1962.

[20] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, “TSP Cuts Which Do Not Conform to the Template Paradigm.”

[21] D. Applegate, R. Bixby, C. Vasek, and W. Cook, “On the Solution of Traveling Salesman Problems,” *Doc. Math.*, vol. Extra Volu, pp. 645–656, 1998.

[22] E. Balas, S. Ceria, and G. Cornuéjols, “A lift-and-project cutting plane algorithm for mixed 0–1 programs,” *Math. Program.*, vol. 58, no. 1–3, pp. 295–324, Jan. 1993.

[23] G. Laporte, “The Traveling Salesman Problem: An overview of exact and approximate algorithms,” *Eur. J. Oper. Res.*, vol. 59, pp. 231–247, 1992.

[24] E. Balas and N. Christofides, “A restricted Lagrangean approach to the traveling salesman problem,” *Math. Program.*, vol. 21, no. 1, pp. 19–46, Dec. 1981.

[25] R. E. Tarjan, “Finding optimum branchings,” *Networks*, vol. 7, no. 1, pp. 25–35, 1977.

[26] K. Helbig Hansen and J. Krarup, “Improvements of the Held—Karp algorithm for the symmetric traveling-salesman problem,” *Math. Program.*, vol. 7, no. 1, pp. 87–96, Dec. 1974.

[27] T. H. C. Smith and G. L. Thompson, “A Lifo Implicit Enumeration Search Algorithm for the Symmetric Traveling Salesman Problem Using Held and Karp’s 1-Tree Relaxation,” *Ann. Discret. Math.*, vol. 1, pp. 479–493, 1977.

[28] N. Christofides, “Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem.” 1976.

[29] M. Gendreau, A. Hertz, and G. Laporte, “New Insertion and Postoptimization Procedures for the Traveling Salesman Problem,” *Oper. Res.*, vol. 40, no. 6, pp. 1086–1094, Dec. 1992.

[30] E. de Klerk, D. V. Pasechnik, and R. Sotirov, “On Semidefinite Programming Relaxations of the Traveling Salesman Problem,” *SIAM J. Optim.*, vol. 19, no. 4, pp. 1559–1573, Jan. 2009.

[31] J. Van der Veen, “Solvable cases of the traveling salesman problem with various objective functions,” 1992.

[32] E. Klerk and C. Dobre, “A comparison of lower bounds for the symmetric circulant traveling salesman problem,” *Discreete Appl. Math.*, vol. 159, pp. 1815–1826, 2011.

[33] R. Zamani and S. K. Lau, “Embedding learning capability in Lagrangean relaxation: An application to the travelling salesman problem,” *Eur. J. Oper. Res.*, vol. 201, no. 1, pp. 82–88, 2010.

[34] A. J. Orman and H. P. Williams, “A survey of different integer programming

formulations of the travelling salesman problem," London, 2005.

[35] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer Programming Formulation of Traveling Salesman Problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, Oct. 1960.

[36] G. Pataki, "Teaching Integer Programming Formulations Using the Traveling Salesman Problem *," vol. 45, no. 1, pp. 116–123.

[37] G. Pataki, "The bad and the good-and-ugly: formulations for the travelling salesman problem," 2000.

[38] M. Teresa Godinho, L. Gouveia, T. L. Magnanti, P. Pesneau, and J. Pires, "On the Unit Demand Vehicle Routing Problem : Flow Based Inequalities implied by a Time - dependent Formulation."

[39] M. T. Godinho, L. Gouveia, T. L. Magnanti, P. Pesneau, and J. Pires, "On Time-Dependent Models for Unit Demand Vehicle Routing Problems."

[40] B. Gavish and S. C. Graves, "The travelling salesman problem and related problems," Cambridge, Working Paper OR-078-78, 1978.

[41] B. Gendron, T. G. Crainic, and A. Frangioni, "Multicommodity Capacitated Network Design," in *Telecommunications Network Planning*, Boston, MA: Springer US, 1999, pp. 1–19.

[42] T. Öncan, İ. K. Altınel, and G. Laporte, "A comparative analysis of several asymmetric traveling salesman problem formulations," *Comput. Oper. Res.*, vol. 36, no. 3, pp. 637–654, 2009.

[43] S. C. Sarin, H. D. Sherali, and A. Bhootra, "New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints," 2005.

[44] A. Langevin, F. Soumis, and J. Desrosiers, "Classification of travelling salesman problem formulations," *Oper. Res. Lett.*, vol. 9, no. 2, pp. 127–132, Mar. 1990.

[45] A. Claus, "A New Formulation for the Travelling Salesman Problem," *SIAM J. Algebr. Discret. Methods*, vol. 5, no. 1, pp. 21–25, Mar. 1984.

[46] T. Ibaraki, "Integer programming formulation of combinatorial optimization problems," *Discrete Math.*, vol. 16, no. 1, pp. 39–52, Sep. 1976.

[47] H. D. Sherali, S. C. Sarin, and P.-F. Tsai, "A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints," *Discret. Optim.*, vol. 3, no. 1, pp. 20–32, 2006.

[48] L. Gouveia and S. Voß, "A classification of formulations for the (time-dependent) traveling salesman problem," *Eur. J. Oper. Res.*, vol. 83, no. 1, pp. 69–82, 1995.

[49] U. Pferschy and R. Staněk, "Generating subtour elimination constraints for the TSP from pure integer solutions," *Cent. Eur. J. Oper. Res.*, pp. 1–30, 2016.

[50] M. Drexl, "A note on the separation of subtour elimination constraints in elementary shortest path problems," *Eur. J. Oper. Res.*, vol. 229, no. 3, pp. 595–598, 2013.

[51] M. Grötschel and O. Holland, "Solution of large-scale symmetric travelling salesman

problems," *Mathematical Programming*, vol. 51, no. 1–3. pp. 141–202, 1991.

[52] M. Hahsler and K. Hornik, "TSP – Infrastructure for the Traveling Salesperson Problem," *J. Stat. Softw.*, vol. 1, no. 1, pp. 1–21, 2007.

[53] B. Angeniol, G. Croix Vaubois, and J. Y. Le Texier, "Self-Organizing Feature Maps and the Travelling Salesman Problem," *Neural Networks*, vol. 1, pp. 289–293, 1988.

[54] B. F. J. La Maire and V. M. Mladenov, "Comparison of neural networks for solving the travelling salesman problem," in *11th Symposium on Neural Network Applications in Electrical Engineering*, 2012, pp. 21–24.

[55] C.-K. Looi, "Neural network methods in combinatorial optimization," *Comput. Oper. Res.*, vol. 19, no. 3, pp. 191–208, 1992.

[56] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[57] R. Durbin and D. Willshaw, "An analogue approach to the travelling salesman problem using an elastic net method," *Nature*, vol. 326, no. 6114, pp. 689–691, Apr. 1987.

[58] D. E. V. den Bout and T. K. Miller, "Graph partitioning using annealed neural networks," *IEEE Transp. Neural Networks*, no. 1, pp. 192–203, 1990.

[59] S. U. Hegde, J. L. Sweet, and W. B. Levy, "Determination of Parameters in a Hopfield/Tank Computational Network," *IEEE Transp. Neural Networks*, vol. 2, pp. 291–298, 1988.

[60] J. Balicki, Z. Kitowski, and A. Stateczny, "Extended Hopfield models of neural networks for combinatorial multiobjective optimization problems," in *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, vol. 2, pp. 1646–1651.

[61] B. W. Lee and B. J. Sheu, "Combinatorial optimization using competitive Hopfield neural network," *Biol. Cybernet*, no. 62, pp. 415–423, 1990.

[62] S. M. Abdel-Moetty, "Traveling Salesman Problem Using Neural Network Techniques," in *Informatics and Systems (INFOS), 2010 The 7th International Conference*, 2010.

[63] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *Biosystems*, vol. 43, no. 2, pp. 73–81, 1997.

[64] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[65] J.-Y. Potvin, "The Traveling Salesman Problem: A Neural Network Perspective."

[66] H. Ghaziri and I. H. Osman, "A neural network algorithm for the traveling salesman problem with backhauls," *Comput. Ind. Eng.*, vol. 44, no. 2, pp. 267–281, 2003.

[67] J.-C. Créput and A. Koukam, "A memetic neural network for the Euclidean traveling salesman problem," *Neurocomputing*, vol. 72, no. 4, pp. 1250–1264, 2009.

[68] M. . Saadatmand-Tarzjan, M. . Khademi, M.-R. . Akbarzadeh-T., and H. A. Moghaddam, "A Novel Constructive-Optimizer Neural Network for the Traveling Salesman Problem,"

IEEE Trans. Syst. Man Cybern. Part B, vol. 37, no. 4, pp. 754–770, Aug. 2007.

[69] X.-S. Yang and Suash Deb, "Cuckoo Search via Lévy flights," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, pp. 210–214.

[70] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Comput. Appl.*, vol. 24, no. 7–8, pp. 1659–1669, Jun. 2014.

[71] D. Karaboga and B. Gorkemli, "A combinatorial Artificial Bee Colony algorithm for traveling salesman problem," in *2011 International Symposium on Innovations in Intelligent Systems and Applications*, 2011, pp. 50–53.

[72] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43.

[73] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, pp. 4104–4108.

[74] T. Zeugmann *et al.*, "Particle Swarm Optimization," in *Encyclopedia of Machine Learning*, Boston, MA: Springer US, 2011, pp. 760–766.

[75] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pp. 69–73.

[76] M. F. F. Ab Rashid, N. M. Zuki, and A. N. M. Rose, "Optimization of Traveling Salesman Problem with Precedence Constraint using Particle Swarm Optimization," *J. Sci. Res. Dev.*, vol. 2, no. 13, pp. 212–216, 2015.

[77] X. H. Shi, Y. Zhou, L. M. Wang, Q. X. Wang, and Y. C. Liang, "A discrete particle swarm optimization algortihm for travelling saelsman problem," in *Computational Methods*, Springer International Publishing, 2006, pp. 1063–1068.

[78] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and Q. X. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP," 2007.

[79] S. M. Chen and C. Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," *Expert Syst. Appl.*, vol. 38, pp. 14439–14450, 2011.

[80] S. Somhom, A. Modares, and T. Enkawa, "A self-organising model for the travelling salesman problem," *J. Oper. Res. Soc.*, vol. 48, no. 9, pp. 919–928, Sep. 1997.

[81] T. A. S. Masutti and L. N. de Castro, "A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem," *Inf. Sci. (Ny).*, vol. 179, no. 10, pp. 1454–1468, 2009.

[82] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," *Appl. Soft Comput.*, vol. 11, no. 4, pp. 3680–3689, 2011.

[83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing,"

Sci. New Ser., vol. 220, no. 4598, pp. 671–680, 1983.

[84] K. Meer, "Simulated Annealing versus Metropolis for a TSP instance," *Inf. Process. Lett.*, vol. 104, no. 6, pp. 216–219, Dec. 2007.

[85] I. Wegener, "Simulated Annealing Beats Metropolis in Combinatorial Optimization," Springer, Berlin, Heidelberg, 2005, pp. 589–601.

[86] C.-B. Cheng and C.-P. Mao, "A modified ant colony system for solving the travelling salesman problem with time windows," *Math. Comput. Model.*, vol. 46, no. 9, pp. 1225–1235, 2007.

[87] M. L. Ibanez and C. Blum, "Beam-ACO for the travelling salesman problem with time windows," *Comput. Oper. Res.*, vol. 37, no. 9, pp. 1570–1583, 2010.

[88] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem," *Appl. Soft Comput.*, vol. 30, pp. 484–490, 2015.

[89] Z. Wang, J. Guo, M. Zheng, and Y. Wang, "Uncertain multiobjective traveling salesman problem," *Eur. J. Oper. Res.*, vol. 241, no. 2, pp. 478–489, 2015.

[90] L.-P. Wong, M. Y. H. Low, and C. S. Chong, "A Bee Colony Optimization Algorithm for Traveling Salesman Problem," in *2008 Second Asia International Conference on Modelling & Simulation (AMS)*, 2008, pp. 818–823.

[91] Y. Marinakis and M. Marinaki, "A Hybrid Multi-Swarm Particle Swarm Optimization algorithm for the Probabilistic Traveling Salesman Problem," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 432–442, 2010.

[92] Wei Pang, Kang-ping Wang, Chun-guang Zhou, and Long-jiang Dong, "Fuzzy discrete particle swarm optimization for solving traveling salesman problem," in *The Fourth International Conference onComputer and Information Technology, 2004. CIT '04.*, pp. 796–800.

[93] M. Clerc, "Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem," Springer Berlin Heidelberg, 2004, pp. 219–239.

[94] Kang-Ping Wang, Lan Huang, Chun-Guang Zhou, and Wei Pang, "Particle swarm optimization for traveling salesman problem," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, pp. 1583–1585.

[95] J. B. Odili, M. N. M. Kahar, S. Anwar, and M. A. K. Azrag, "A Comparative Study of African Buffalo Optimization and Randomized Insertion Algorithm for Asymmetric Travelling Salesman's Problem," in *4th International Conference on Software Engineering and Computer Systems*, 2015, pp. 90–95.

[96] J. B. Odili, M. N. M. Kahar, and S. Anwar, "African Buffalo Optimization: A Swarm-Intelligence Technique," *Procedia Comput. Sci.*, vol. 76, pp. 443–448, 2015.

[97] J. B. Odili and M. N. Mohmad Kahar, "Solving the Traveling Salesman's Problem Using the African Buffalo Optimization," *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–12, 2016.

[98] J. B. Odili, M. N. Kahar, and A. Noraziah, "Solving Traveling Salesman's Problem Using

African Buffalo Optimization, Honey Bee Mating Optimization & Lin-Kerninghan Algorithms," *World Appl. Sci. J.*, vol. 34, no. 7, pp. 911–916, 2016.

[99] N. M. Razali and J. Geraghty, "Genetic Algorithm Performance with Different Selection Strategies in Solving TSP," in *The 2011 International Conference of Computational Intelligence and Intelligent Systems*, 2011.

[100] H. Braun, "On solving travelling salesman problems by genetic algorithms," in *Parallel Problem Solving from Nature*, Berlin/Heidelberg: Springer-Verlag, 1990, pp. 129–133.

[101] N. M. Noraini, "An Efficient Genetic Algorithm for Large Scale Vehicle Routing Problem Subject to Precedence Constraints," *Procedia - Soc. Behav. Sci.*, vol. 195, pp. 1922–1931, 2015.

[102] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," *Biol. Cybern.*, vol. 60, no. 2, pp. 139–144, Dec. 1988.

[103] J. Yang, C. Wu, H. P. Lee, and Y. Liang, "Solving traveling salesman problems using generalized chromosome genetic algorithm," *Prog. Nat. Sci.*, vol. 18, no. 7, pp. 887–892, 2008.

[104] R. Baraglia, J. I. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 5, no. 6, pp. 613–622, 2001.

[105] N. L. J. Ulder, E. H. L. Aarts, H.-J. Bandelt, P. J. M. van Laarhoven, and E. Pesch, "Genetic local search algorithms for the traveling salesman problem," Springer Berlin Heidelberg, 1991, pp. 109–116.

[106] B. Bontoux, C. Artigues, and D. Feillet, "A Memetic Algorithm with a large neighborhood crossover operator for the Generalized Traveling Salesman Problem," *Comput. Oper. Res.*, vol. 37, no. 11, pp. 1844–1852, 2010.

[107] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *Eur. J. Oper. Res.*, vol. 174, no. 1, pp. 38–53, 2006.

[108] Licheng Jiao and Lei Wang, "A novel genetic algorithm based on immunity," *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 30, no. 5, pp. 552–561, 2000.

[109] J.-Y. Potvin, "Genetic algorithms for the traveling salesman problem," *Ann. Oper. Res.*, vol. 63, no. 3, pp. 337–370, Jun. 1996.

[110] A. M. S. Zalzala and P. J. Fleming, *Genetic algorithms in engineering systems*. London: The Institution of Electrical Engineers, 1997.

[111] G. Winter, M. Periaux, P. Galan, and P. Cuesta, *Genetic algorithms in engineering and computer science*. New York: Wiley Subscription Services, Inc., A Wiley Company, 1995.

[112] M. Kaufmann, *Foundations of genetic algorithms*. San Mateo, 1991.

[113] K. F. Man, K. S. Tang, and B. W. S. Wah, *Genetic algorithms: concepts and designs*. New York: Springer, 1999.

[114] J. T. Alander, "Proceedings of the First Nordic Workshop on Genetic Algorithms and their Applications (1NWGA)," 1995.

[115] W. Banzhaf *et al.*, "Proceedings of the genetic and evolutionary computation

conference," 1999.

[116] J. J. Grefenstette, Naval Research Laboratory (U.S.), and Texas Instruments Incorporated., *Proceedings of the First International Conference on Genetic Algorithms and Their Applications : July 24-26, 1985 at the Carnegie-Mellon University, Pittsburg, PA*. .

[117] J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., "Genetic programming: proceedings of the first annual conference," 1996.

[118] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 616–621.

[119] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators," *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 129–170, 1999.

[120] Y. Deng, Y. Liu, and D. Zhou, "An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP," *Math. Probl. Eng.*, vol. 2015, 2015.

[121] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *Eur. J. Oper. Res.*, vol. 140, no. 3, pp. 606–617, 2002.

[122] M. Albayrak and N. Allahverdi, "Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 1313–1320, 2011.

[123] F. Glover, "Artificial intelligence, heuristic frameworks and tabu search," *Manag. Decis. Econ.*, vol. 11, no. 5, pp. 365–375, 1990.

[124] J. M. Basart and L. Huguet, "An approximation algorithm for the TSP," *Inf. Process. Lett.*, vol. 31, no. 2, pp. 77–81, Apr. 1989.

[125] E. Erkut and J. Zhang, "The maximum collection problem with time-dependent rewards," *Nav. Res. Logist.*, vol. 43, no. 5, pp. 749–763, Aug. 1996.

[126] E. Balas, "The prize collecting traveling salesman problem," *Networks*, vol. 19, no. 6, pp. 621–636, 1989.

[127] A. Ekici and A. Retharekar, "Multiple agents maximum collection problem with time dependent rewards," *Comput. Ind. Eng.*, vol. 64, no. 4, pp. 1009–1018, 2013.

[128] S. Kataoka and S. Morito, "An Algorithm for Single Constraint Maximum Collection Problem," *Oper. Res. Soc. Japan*, vol. 31, no. 4, pp. 515–530, 1988.

[129] A. M. Chwatal, G. R. Raidl, and nther R., "Solving the Minimum Label Spanning Tree Problem by Mathematical Programming Techniques," *Adv. Oper. Res.*, vol. 2011, pp. 1–38, 2011.

[130] N. Fan and M. Golari, "Integer Programming Formulations for Minimum Spanning Forests and Connected Components in Sparse Graphs," Springer, Cham, 2014, pp. 613–622.

VITA

SUMMARY OF PROFESSIONAL EXPERIENCE

Researcher with 5 years of experience in TUBITAK, mainly working in R&D projects in logistics and transportation as well as strategic management. Actively working in data analysis, logistics, mathematical modelling applications, network scheduling and ORER planning. Strategic planning, project management and process management are the secondary fields. Working also as instructor conducting strategical management trainings, project management trainings and PMP trainings. Took part in leadership programs to host and train managers from other countries. Developed business strategies with foreign managers from European countries, in particular EU-Horizon projects. Designed German and English trainings from beginner to advance level, and gave courses in mathematics, statistics, data management and optimization methods. Giving special trainings such as operations research, simulation and statistics.

Currently working as a strategy developer and data analyzer as well as reporter in DOKAP's (Black Sea Regional Development Authority) project named "Development of Entrepreneurship and Innovation Ecosystem of SMEs" and working as process consultor in "Integrated Management System Project" of PTT (Post and Telegraph Organization).

CONFERENCES, NOTICES & PUBLICATIONS

- M.ÇAL, İ. ÖNDEN, F.ELDEMİR, M. SAMASTI "Istanbul Traffic Problem: Expert Opinions", ISMA 13th Strategic Management Conference, 2017
- F.ELDEMİR, M.ÇAL and A.ÜNAL, "Istanbul Metrobus System Demand Forecasting Models" TRANSİST 2013 Notice Booklet, pp.311–315, 201
- M. ÇAL, A. ÜNAL, F. ELDEMİR, "The Optimization Of Drivers In Bus Rapid Transit System: A Case For Istanbul Metrobus System, 2015
- M.ÇAL, "General Driver Assignment Model in a Metrobus System", 2014
- M.ÇAL, ITU Quality in Healthcare Conferences: Ventilator Associated Pneumonia and VAP Bundle, Industrial Applications, 2014
- Umraniye Municipality Wunderkinds Leadership Seminars, 2014