



Faculty of Sciences
Department of Mathematics & Statistical Sciences

Urban Traffic Flow Management Problem through a Bilevel Optimisation approach

by

Bame Menyatso

Reg. No: 18100034
BSc (Mathematics), UB, Botswana

A Dissertation/Thesis Submitted to the Faculty of Sciences in Partial Fulfilment of the
Requirements for the Award of the Degree of Master of Science in Pure and Applied
Mathematics of BIUST

Supervisor(s):

Prof Semu M. Kassa
Department of Mathematics & Statistical Sciences,
Faculty of Sciences, BIUST
E-mail: kassas@biust.ac.bw

February 13, 2023

DECLARATION REGARDING THE WORK AND COPYRIGHT

Candidate Bame Menyatso

Student ID: 18100034

Thesis Titled: Urban Traffic Flow Management Problem through a bilevel optimisation approach

I **Bame Menyatso**, certify that the Thesis is all my own original work and that I have not obtained a degree in this University or elsewhere on the basis of any of this work.

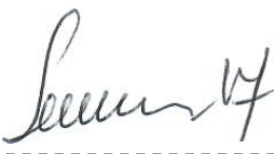
This dissertation/thesis is copyright material protected under the Berne Convention, the Copyright and Neighbouring Rights Act, Act. No. 8 of 2000 and other international and national enactments, in that behalf, on intellectual property. It must not be reproduced by any means, in full or in part, except for short extracts in fair dealing; for researcher private study, critical scholarly review or discourse with an acknowledgement, without the written permission of the office of the Postgraduate School, on behalf of both the author and the BIUST.

Signed: 

Date: December 7, 2022

Primary Supervisor Prof Semu M. Kassa

I **Prof Semu M. Kassa**, hereby confirm that I have inspected the above titled thesis and, to the best of my knowledge, it is based on the original work of the candidate.

Signed: 

Date: December 9, 2022

Abstract

There are various strategies used to alleviate the problem of traffic congestion. These include increasing the capacity, using coordinated traffic signal systems, and building new infrastructure. Traffic flow models have many applications and can assist the traffic authority in planning purposes. In this study we present a deterministic model formulation of a hierarchical traffic flow management problem. The study considers the traffic authority minimising the total travel time taken by the traffic that flows over the whole network system to clear while the driver who intend to minimise their own flow, is part of the constraints. There are two decision makers in this practical problem. On the upper level we have the traffic authority who decides the free flow speed and cycle length to set on each traffic light. On the other hand the drivers are free to choose the route from their origin to the destination of their choice. We deal with a two level optimisation problem, with a single leader and multiple followers which is later converted to a single leader and single follower problem. In this problem we consider a signal controlled network only and the travel time function used is a volume delay function called the Bureau of Public Roads (BPR) formula, we combine it with the modified Webster equation which computes the waiting time on a traffic signal. Bilevel problems are classified as NP-hard problems and are difficult to solve. In this problem we work with a non-linear travel cost function. The lower level problem is a binary-type problem, and it is not possible to find its derivative, therefore we cannot use the derivative optimisation methods to solve the lower level problem. In addition, some variables of both the lower and upper level objectives are implicit, making it difficult to find its derivative. In this regard we choose to employ a meta heuristic algorithm that can solve deterministic bilevel problems effectively. In particular we use Binary Particle Swarm Optimisation (BPSO) technique to solve the lower level problem after choosing the leader's actions using the systematic sampling technique. The results indicate that this model can be useful to the traffic authority as it can be applied in urban areas that have signal controlled intersections in their roadways. The model has been implemented and a particular example is used to simulate the results.

Key words: Urban traffic flow network, Deterministic hierarchical model, Bilevel programming, Meta-heuristic solution algorithms, Binary Particle Swarm Optimization.

Dedications

I dedicate this thesis to my lovely daughter Realeboga Victoria Menyatso.



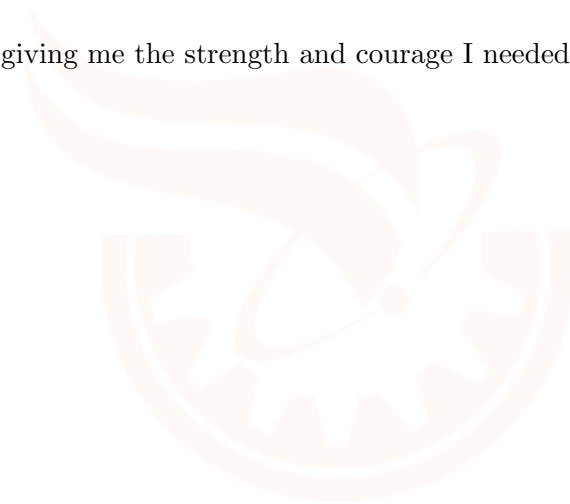
Acknowledgements

First and foremost I thank the Botswana International University of Science and Technology for giving me an opportunity to be a student in their institution.

I would like to express my sincere gratitude to my supervisor Professor Semu Kassa for giving me guidance and support when I was working on this thesis. He has contributed immensely with many constructive ideas towards my work. I thank him for always encouraging me to work harder and to put more effort into my work and for allowing me the opportunity to be an independent researcher. Without him I would not have been able to complete this thesis.

I wish to acknowledge the assistance and support that was provided by my fellow students and colleagues in the department of Mathematics and Statistics by creating a conducive office space. The assistance provided by my family and friends is also highly appreciated.

Lastly, I thank God for giving me the strength and courage I needed to complete this thesis.



Contents

Contents	vi
List of Figures	vi
List of Tables	vii
1 Introduction	1
1.1 Statement of the problem	3
1.1.1 Main Objective	3
1.1.2 Outline	3
2 Preliminaries	4
2.1 Graphs and Subgraphs	4
2.1.1 Special Subgraphs	5
2.1.2 Trees	7
2.1.3 Networks	8
2.2 Classical Optimisation problem	9
2.2.1 Some special classes of optimisation problems	9
2.2.2 Conditions for Local minimiser	10
2.3 Stackelberg Games	11
2.4 Bilevel Programming Problems	12
2.5 The General Bilevel Programming Problem	12
2.6 KKT- reformulation to reduce the bilevel problem to single level	14
3 Literature Review	16
4 Model Formulation	20
4.1 The traffic network	20
4.2 Description of the cost function	21
4.3 Lower level decision maker: The Driver's problem	23
4.4 Upper level decision maker: The Traffic Authority's Problem	24
4.5 The Bilevel problem	25
5 Solution Approach	27
5.1 A systematic sampling numerical method for the bilevel model	27
5.1.1 Partitioning the decision space of the upper level problem	27
5.1.2 The lower level problem	30
5.1.3 The bilevel optimisation problem	31
5.2 Binary particle swarm optimisation	31

5.3	A particular example of the Authority and Driver's problem	35
5.3.1	The lower level problem (Driver's problem)	37
5.3.2	Lower level objective	37
5.3.3	The upper level problem (Traffic Authority's problem)	42
5.4	Numerical Results	44
5.5	Case 1: 100 vehicles from the source nodes	44
5.6	Case 2: The number of vehicles from the source nodes with different proportions	47
5.7	Case 3: 500 vehicles from the source nodes	48
6	Conclusion and Recommendations	50
	Appendix	52
A	MatLab Codes	52
A.1	The main partitioning MatLab code	52
A.2	The binary particle swarm optimisation MatLab code	70
	Appendix	74



List of Figures

2.1	Directed graph	4
2.2	Road network	6
2.3	Weighted graph	7
2.4	A tree	8
4.1	Weighted graph	20
5.1	The Leader's decision space diagram.	28
5.2	Partitioning decision space of the leader.	28
5.3	Example of a network.	36



List of Tables

4.1	Description of variables and Parameters.	22
5.1	Binary PSO parameters	33
5.2	Parameters and variables - lower level.	37
5.3	Parameters and variables - upper level level.	42
5.4	Lower level objective function (case1)	45
5.5	The upper and lower level best cost (case 1)	45
5.6	The upper level optimal solution (case 1)	46
5.7	The upper level optimal solution (case 1)	46
5.8	Case 1	47
5.9	Case 2	47
5.10	The upper level optimal solution (case 2)	47
5.11	The upper level optimal solution (case 2)	48
5.12	The upper level optimal solution (case 3)	48
5.13	The upper level optimal solution (case 3)	49

1. Introduction

Adequate transport planning and management is an important aspect of traffic control. Traffic flow models provide researchers and engineers with important information concerning delays incurred by traffic due to congestion, the travel time on a road section, and the behaviour of vehicles in general. This information can be useful to the traffic authority for planning purposes to help create more desirable traffic conditions [1]. Traffic flow modeling was introduced in the 1930's by Bruce D. Greenshields [2] who is considered as the pioneer of traffic flow theory. He studied the relationship between traffic density, speed and volume [3]. In the 1950's new models were discovered and they were used to describe that traffic flows change over time or are dynamic. The invention of new technologies such as computers in the 1990's brought about models which are still in use in the present day. Due to growing economies and technological advancements, nowadays researchers use efficient algorithms for more accurate results which can also be attained in a reasonable amount of time. The traffic flow model used in this study is a combination of the Bureau of Public Roads (BPR) [4] formula which is a volume delay function and the modified Webster equation [5] [6]. It is designed to calculate the travel time of a vehicle in a link and the waiting time spent by a vehicle at each signalised intersection. The BPR formula calculates the link travel time while the Webster formula calculates the waiting time at a signalised intersection. A summation of the link travel time and the delay at the intersection will give the total travel time for a vehicle's whole journey.

The first electric or automated traffic light was invented by Lester Farnsworth Wire (1887-1958) in 1912 [7]. Ever since then due to the invention and increased use of computers, traffic lights have been controlled systematically to manage flow and congestion at intersections. Previous research shows that congestion at intersections can be reduced by optimising the cycle length of traffic lights [8]. Although there are many factors which contribute to traffic congestion and traffic jams such as poor infrastructure and poor planning one of the main problems is inefficient management of the traffic signal optimisation [9]. Different studies have indicated that the use of heuristic algorithms such as genetic algorithm and particle swarm optimisation can be used for traffic light control [10] [11].

When modelling traffic management in urban areas the main parameters used in the model include but are not limited to the green interval and the cycle length of the traffic light. Most models use only one of these parameters [12]. In [13] Eriskin et al considered the green interval as the only parameter. In our study the model parameters we considered are the cycle length and the free flow speed. In [12] Pavlova et al considered a city network, they managed the flow by optimising the cycle length of the traffic signals. They used the bilevel approach to handle a large number of variables and constraints. They indicate in their research that the use of two objective functions leads to better results. In [14] bilevel programming model of urban public transport network under fairness constraints is considered. Congestion remains a challenge in cities and towns and model predictive control is applied in [15] in order to reduce

the travel time thereby improving flow in the traffic network. A store and forward model is presented by Aboudolas et al in [16] as they consider the problem of designing traffic signal control strategies for a large congested urban network.

Traffic congestion is one of the main problems that is faced by towns and city councils in the present time. Without proper management of traffic, road users will experience delay, unpleasant traffic conditions, road accidents and increased fuel consumption due to spending too much time in traffic. In the past years the population has increased significantly and this has led to increased road traffic congestion. Different methods of managing traffic include the use of roundabouts, signal controlled intersections (static and vehicle actuated), strict bus lanes and cycle lanes. Managing road traffic flow using traffic signals is more cost effective than increasing the capacity of a road network. Therefore we formulate a bilevel programming model in order to address the problem of traffic congestion by managing the flows at signalised intersections.

In this study the traffic authority aims to minimise the travel time for the whole network while the driver who seeks to minimise their own travel time is also a constraint. We chose to adopt the bilevel modeling approach in which the decision process is in a hierarchical manner. After analysing their set of feasible solutions the leader will make their decision first while the follower will respond accordingly to the decision made by the leader. We assume that the network is in an urban area and that its intersections are managed by traffic signals only. The problem of traffic congestion is influenced by many parameters such as weather, the type of vehicles, type of intersections (roundabout or stop-sign), capacity and many other factors which makes it a complex problem to pursue [17]. In this particular problem we focus on the optimisation of the cycle length of the traffic signal and the maximum permitted speed on the roadway.

Bilevel programming problems are two level hierarchical problems where there is what we call a leader (in the upper level problem) and a follower (in the lower level problem). In a bilevel problem the lower level problem is a constraint that itself is also an optimisation problem [18] [19]. These problems are very difficult to solve and they are classified as NP-hard, even the simplest bilevel problem with a linear objective function and constraints can be hard to solve. Consequently it is necessary to do a solid research before proposing an effective and efficient algorithm to solve a bilevel programming problem. Many algorithms which are already in existence cannot solve all bilevel problems since the solution methods are based on the structure of the problem itself and convergence to a solution is not usually guaranteed.

The goal of this research is for the traffic authority to find a cycle length and the maximum speed for each link which minimises the travel time for the whole network while the driver who is a follower as well as a constraint for the upper level problem seeks to minimise his or her own travel time. In the proposed solution approach the leader's actions are selected in a systematic manner after first partitioning the decision space of the leader. We then apply a binary particle swarm optimisation technique (BSOP) to acquire a global minimum for the follower corresponding to each of the leader's actions. Particle swarm optimisation (PSO) technique is based on how animals behave when they are in a group such as a school of fish or a flock of birds. This technique was first introduced by Kennedy and Eberhart [20] in 1995. Since the PSO is used

for problems in the continuous space, however in this work we use the Binary Particle Swarm Optimisation (BPSO) technique. We have chosen this approach because our model equation is not fully convex, the involved functions are inductively defined with mixed integer variables, and the chosen algorithm is a derivative free method and is easier to implement.

1.1 Statement of the problem

Drivers prefer to spend as less time as possible on the roadways and the traffic authority should be able to route the traffic in as short time as possible. However different authorities around the world are experiencing the problem of traffic congestion and delays. Many studies have focused on a single level approach on the problem of optimisation of traffic signals concerned only with the cycle length and the green interval. With all those studies conducted the problem of traffic congestion still prevails. Consequently it is necessary for this study to propose a bilevel model for the traffic authority to manage traffic flows in a signal controlled network.

1.1.1 Main Objective

The goal of this research is to develop a bilevel optimisation model, which improves the urban traffic flow management by minimising the travel time for the whole network system.

Specific Objectives

- To formulate a model for the traffic authority which determines the optimal cycle length for the traffic signals at each signalised intersection in the network.
- To formulate a model for the traffic authority which can be used to set the optimal maximum speed on the roadways of a network in a town.
- To propose an efficient traffic flow management plan for the authority which also promotes safety for the road users.

1.1.2 Outline

This thesis is organized as follows. In the next chapter we present the preliminaries, and layout definitions and theorems which will be used in this thesis. This chapter is followed by the literature review which gives an overview of the past and present literature. In Chapter 4 we formulate the bilevel optimisation model- the objectives for the upper and lower level with their constraints and the description of the cost function. In Chapter 5 we give a discussion of the solution approach with details of the algorithms which will be used and the software used for the analysis. Last but not least Chapter 6 is a summary of the conclusion and recommendations for the traffic authority.

2. Preliminaries

This chapter comprises of an overview of the terminology and concepts used in the rest of the text. This will be from the fields of graph theory, classical optimisation and bilevel optimisation.

2.1 Graphs and Subgraphs

The applications of Graph Theory include telecommunications such as determining the minimum cost of laying telephone cables, for chemical nomenclature, computer science and engineering. Graphs are also used to represent communication networks, road networks and can be used to study the flow-rate of water in a pipe network [21][22][23].

A graph G consists of a set of nodes V which are represented by unique points and a set of edges E of pairs of these nodes which are represented by arcs. A pair of nodes (i, j) is connected by an edge $e = \{i, j\}$ and i and j are called the endpoints of the edge. A relation called incidence associates each edge with a pair of nodes. The endpoints i and j of the edge are said to be adjacent. The edge is also incident to each of the nodes, and vice versa. Two edges of a graph are adjacent if they are incident to a common node.

Figure 2.1 below shows the graph of a road network of a town, where the edges (arcs) represent the road segments and the nodes (points) represent the different intersections on the road network.

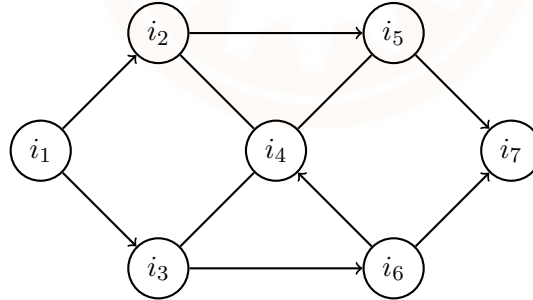


Figure 2.1: Directed graph

We now formally define a graph and some other terms used in graph theory.

Definition 1. A **graph** $G = (V, E)$ is an ordered pair which consists of a non-empty set V of vertices (also called nodes) and a set E of edges consisting of pairs of nodes.

For example for a set of nodes $V = \{i_1, i_2, i_3, i_4, i_5, i_6, i_7\}$, the set of edges is $E = \{\{i_1, i_2\}, \{i_3, i_4\}, \dots\}$. In this case $\{i_1, i_2\}$ is the edge connecting node i_1 to node i_2 and $\{i_3, i_4\}$ is the edge connecting node i_3 to node i_4 .

Definition 2. A **directed graph** is a graph in which the edges may only be considered to traverse in one direction. Edges in a simple directed graph may be specified by an ordered pair

(i, j) of the two vertices that the edge connects. In this case, we say that i is adjacent to j and j is adjacent from i .

Definition 3. A graph H is a **subgraph** of a graph G if all the vertices of H are a subset of the set of vertices of G and all the edges in H are also a subset of the set of edges of G .

When two edges share a common vertex we say that they are **incident**. Similarly, vertices can be **incident** with an edge when the vertex is one of its end points, that is, if the edge connects to the vertex we say the edge is **incident** to the vertex. Incidence is used in directed graphs to calculate the degree of a vertex.

Definition 4. An edge with only one endpoint or an edge with identical ends is called a **loop**; and if the edge has distinct endpoints then it is called a **link**.

Two or more edges with the same endpoints are called **multiple** or **parallel** edges, these edges have the same head and tail vertex.

Definition 5. A **simple** graph is a graph with no loop edges or multiple edges. In a simple graph the edges may be specified by a set of two elements of two vertices which the edge connects for example (i_1, i_2) is an edge from node i_1 to node i_2 .

In a directed graph two or more vertices are said to be **adjacent** if they are connected by the same edge, that is for a graph G if we let $a, b \in V$ then a and b are adjacent if and only if $\{a, b\} \in E$

2.1.1 Special Subgraphs

Paths, cycles and walks

A **walk** consists of a sequence of edges and vertices, following one after another. This sequence is of the form $i_0 e_1 i_1 e_2 \cdots e_k i_k$, where i is a vertex and e is an edge. There are different types of walks and in some walks a vertex may be repeated while in a simple graph no vertices may be repeated. The length of a walk is the number of edges in the walk. Some walks may be finite while others are infinite. A walk in which no vertex appears more than once is called a **path**. A path may be in a directed graph or an undirected graph. In a path the vertices $i_0, i_1, i_2, \dots, i_k$ and the edges $e_1, e_2, \dots, e_k$ are distinct. A graph in which there is a path joining any pair of vertices is called a **connected** graph that is for a pair of vertices i_1 and i_2 there is a sequence of vertices and edges joining the two vertices. A **cycle** is a sequence of edges and vertices in which the only repeated vertices are the first and the last. A cycle may also be referred to as a closed path. In a cycle the sequence is of the form $i_0 e_1 i_1 e_2, \dots, i_k$ where $i_0 = i_k$. A graph which does not have a cycle is called **acyclic** or **simple** graph.

In Fig 2.2 $T \rightarrow U \rightarrow X$ is a walk of length 2, and $T \rightarrow V \rightarrow Y \rightarrow W \rightarrow X$ is a walk of length 4. Note that $T \rightarrow U \rightarrow X$ is a path. A walk of the form $T \rightarrow U \rightarrow X \rightarrow W \rightarrow T$ is called a **cycle**.

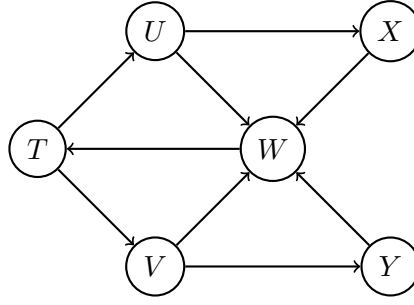


Figure 2.2: Road network

Shortest path and Weighted Graphs

Definition 6. A **weight** on the edge of a graph is a numerical value on the edge of a graph. We usually refer to the weight on an edge as the “cost” and these values are generally nonnegative.

Definition 7. A **weighted graph** is a graph which has numerical weights on its edges. The weights are usually nonnegative numerical values.

The shortest path refers to as a path which carries the minimum weight in a weighted graph between two vertices O (origin) and D (destination). The weights on the edges represent the cost of travel between the vertices which usually is the distance or travel time hence are consequently nonnegative.

One way of finding the shortest path in a weighted graph is through the use of the Dijkstra’s Algorithm [24] [25]. This algorithm was discovered in 1959 by a scientist named E. W Dijkstra of Dutch origin[24][21]. The Dijkstra’s algorithm is still used by many researchers to this date. This algorithm solves the shortest path problem where there is only one origin (O) or source node and one destination (D) node (sink). In this case the source node s is fixed while the destination node can be any vertex on the graph. The Dijkstra’s algorithm is used in a weighted graph where the weights are non-negative.

The Dijkstra’s algorithms calculates the length of the shortest path between the source node s and any other vertex on a graph. The algorithm may be described as follows:

1. Choose the source vertex s which is the initial node.
2. Dijkstra’s algorithm will assign some initial values for the distances from node s to all the other nodes in the network.
3. Dijkstra’s algorithm operates in steps, where at each step it improves the distance values.
4. At each step, the shortest distance from node s to another node is determined.
5. The status label of a node is “Permanent” if its distance value is equal to the shortest distance from node s
6. Otherwise, the status label of a node is “Temporary”
7. The algorithm maintains and step-by-step updates the states of the nodes. At each step one node is designated as “current”

8. If all nodes that can be reached from node s have been permanently labeled we stop.
9. If we cannot reach any temporary labeled node from the current node, then all the temporary labels become permanent- we are done.
10. Otherwise, go to Step 3.

Let $G = (V, E)$ be a graph, the algorithm begins at the origin vertex (node) which is the starting point of the journey. From this point the driver will observe all the possible edges they can choose to travel on. The driver will then choose the one with the minimum weight attached to it to get to the next vertex. They do this while obviously moving in the direction towards their destination. The same thing will happen at the next vertex and this will go on until we reach the destination vertex [24] [21].

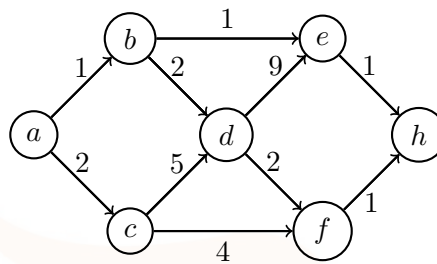


Figure 2.3: Weighted graph

2.1.2 Trees

Trees are a very important class of graphs with various applications. We see the application of trees in different fields of science, for instance, in Chemistry, trees are used for drawing diagrams of molecules. Two molecules may have the same chemical formula but different structures, therefore, in this regard their tree diagrams are used to differentiate the two [26]. More applications of trees include drawing “tree diagrams” when calculating probability, modelling social relationships such as a corporate hierarchy in the work place and many more [27][28].

Definition 8. The **degree** of a vertex is the number of edges incident to the vertex.

Definition 9. A **tree** is an undirected graph that has no cycles or loops. A vertex of degree 1 in a tree is called a **leaf**.

Shown below in Figure 2.4 is an example of a tree.

Theorem 1. A graph G is a tree if and only if there is a unique path between any two vertices in G .

Proof. (see [29]) $\Rightarrow$ Assume that G is a tree. We want to show that there is a unique path between any two vertices of G . Since G is a tree then every two vertices of G are connected by a path. Suppose G has two vertices which are connected by two different paths, then a cycle will be produced, which is a contradiction.

$\Leftarrow$ Suppose there is a unique path between any two vertices in G . We want to show that G is a tree. If G is a tree then it is connected, so between any two vertices there is at least one path.

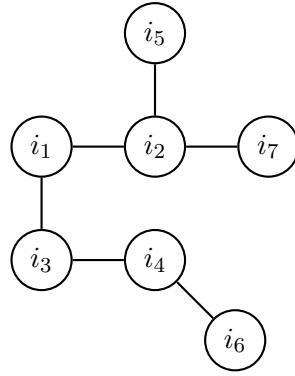


Figure 2.4: A tree

For a contradiction, suppose G has a cycle. Then there are two different paths from vertex i to vertex j , a contradiction! Therefore G is acyclic hence a tree.

2.1.3 Networks

Distance, Flow and Capacity

Definition 10. The **capacity** of an edge (i, j) is a mapping $c : E \rightarrow \mathbb{R}$ and it represents the maximum amount of flow which can pass through an edge.

Let $G = (V, E)$ be a weighted and directed graph. A “flow network” is a directed graph in which an edge $e = (i, j) \in E$ has a capacity $c(i, j) \geq 0$. The edges and the vertices or nodes represent the roadways and the intersections respectively. In a flow network, the origin and destination vertices are referred to as the source and sink respectively. Let an edge $e = (i, j)$ be an edge which connects the vertices i and j , then the flow is a mapping $f : E \rightarrow \mathbb{R}$ which satisfy the following properties [24]:

- For all $i \in V$, $f(i, j) \leq c(i, j)$ (the **capacity constraint**).
- For all $i \in V$, $\sum_{j \in V} f(i, j) = \sum_{j \in V} f(j, i)$ (the **conservation law**).

It is important to note that whenever $f(i, j) = c(i, j)$ in an edge $e = (i, j)$ we say that the edge is **f-saturated**, and it is said to be **f-zero** when $f(i, j) = 0$.

In a flow network the **distance** is given by the weights on the graph which may also be referred to as the cost. Since the vertices or nodes on the network represent intersections then the weights on an edge $e = (i, j)$ can be used to represent the distance from i to j , or the time it takes to travel from i to j [30]. In a graph where there is a single source and a sink the distance between any two vertices i and j is the length of the shortest path between those two vertices. In this thesis we are more concerned with the travel time between two vertices i and j and the travel time will be regarded as the travel cost on the link ij between those nodes.

2.2 Classical Optimisation problem

In this section we describe the classical optimisation problem with only one objective function and some concepts associated with this problem. We also define the notion of convex sets and convexity.

In this section, we consider the optimisation problem,

$$\min f(\mathbf{x}) \text{ s.t } \mathbf{x} \in \Omega = \{\mathbf{x} : \mathbf{g}(\mathbf{x}) \leq 0, \mathbf{h}(\mathbf{x}) = 0\} \subseteq \mathbb{R}^n \quad (2.1)$$

where,

- the function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ which we seek to minimise is a real valued function and is called the objective function.
- $\mathbf{x} = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$ is a vector of independent variables. The variables $x_1, x_2, \dots, x_n$ are called the decision variables.
- The functions $\mathbf{g}$ and $\mathbf{h}$ are referred to as the functional constraints.

2.2.1 Some special classes of optimisation problems

- Linear programming problem

Linear programming is concerned with minimising a linear objective function subject to linear equality and inequality constraints. George Dantzig (1946–1947) developed the general linear programming formulation and introduced the simplex algorithm to solve linear programs [32][33][34]. The algorithm's success led to a vast array of specialisations and generalisations that have dominated practical operations research for half a century. It is the most widely used optimisation problem. It is of the form:

$$\max_{x \in \mathbb{R}^n} c^T x : a_i^T x \leq b_i, i = 1, \dots, m, \quad (2.2)$$

where

$$c \in \mathbb{R}^n, a_i \in \mathbb{R}^n, b_i \in \mathbb{R} \ (i = 1, \dots, m).$$

Linear programming applications include economics, business, manufacturing and transportation [34].

- Least-squares problem

This problem was discovered by Karl Friedrich Gauss in 1795 who was inspired by astronomical studies [35]. This problem has two categories namely, the linear least squares and nonlinear least squares depending on the linearity of the residuals in all unknown variables. This problem takes the following form:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 \quad (2.3)$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $\|\cdot\|_2$ denotes the Euclidean norm.

This method is used in the present day to solve problems in the areas of machine learning, bioinformatics, computer vision and neuroinformatics [36].

- Quadratic programming problem

The quadratic programming problem deals with minimising or maximising a quadratic objective function subject to linear constraints [37]. The Markowitz mean-variance portfolio optimisation problem is a widely used quadratic programming problem [38]. This problem can be used to solve problems in finance such as portfolio analysis, economics and crop selection in agriculture [39] and is formulated as,

$$\min_{x \in \mathbb{R}^n} (\|x\|_2^2 + c^T x) \quad : \quad a_i x^T \leq b_i, i = 1, \dots, m \quad (2.4)$$

where $c \in \mathbb{R}^n$, $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ ($i = 1, \dots, m$) and $\|\cdot\|_2$ denotes the Euclidean norm.

Definition 11. The set $\Omega = \{x : g(x) \leq 0, h(x) = 0\} \subseteq \mathbb{R}^n$ is the **constraint set** or **set of feasible solutions** of (2.1).

If no constraints are specified that is if $\Omega = \mathbb{R}^n$ then problem (2.1) is said to be unconstrained.

Definition 12. The point $x^* \in \mathbb{R}^n$ is the **optimal solution** of the optimisation problem (2.1) if there exist $x^* \in \Omega$ such that $f(x^*) \leq f(x)$ for all $x \in \Omega$.

Definition 13. Suppose that $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a real-valued function defined on some set $\Omega \subset \mathbb{R}^n$. A point $x^* \in \Omega$ is a **local minimiser** of f over Ω if there exists $\epsilon > 0$ such that $f(x) \geq f(x^*)$ for all $x \in \Omega \setminus \{x^*\}$ and $\|x - x^*\| < \epsilon$.

Definition 14. A point $x^* \in \Omega$, is a **global minimiser** of f over Ω if $f(x) \geq f(x^*)$ for all $x \in \Omega \setminus \{x^*\}$.

In the definitions of local minimiser (Definition 13) and global minimiser (Definition 14), if we replace “ $\geq$ ” with “ $>$ ” then we have a strict local minimiser and a strict global minimiser, respectively.

2.2.2 Conditions for Local minimiser

In this section we derive the conditions for x^* to be a local minimiser. Before we proceed we need to define **feasible directions**.

Definition 15. A vector $d \in \mathbb{R}^n$, $d \neq 0$, is a **feasible direction** at $x \in \Omega$ if there exists $\alpha_0 \geq 0$ such that $x + \alpha d \in \Omega$ for all $\alpha \in [0, \alpha_0]$.

Remark: If the function f has continuous partial derivatives of order p then we write $f \in \mathcal{C}^p$

Definition 16. The matrix of second derivatives of f is called the **Hessian** matrix of f at the point x , and it is denoted by $\nabla^2 f(x)$.

Definition 17. A real symmetric matrix A is **positive definite** if and only if $x^T A x$ is positive for all non zero vectors x of $\mathbb{R}^n$.

Theorem 2. First order necessary condition

Let Ω be a subset of $\mathbb{R}^n$, and $f \in C^1$ be a real valued function on Ω . If x^* is a local minimiser of f over Ω , then for any feasible direction d at x^* , we have

$$d^T \nabla f(x^*) \geq 0, \quad (2.5)$$

where $\nabla f(x^*)$ represents the gradient of f at x^* .

For the special case where x^* is in the interior of Ω . We have the following result

Corollary 1 (Interior case)

Let Ω be a subset of $\mathbb{R}^n$ and $f \in C^1$ a real valued function on Ω . If x^* is a local minimiser of f over Ω and if x^* is an interior point of Ω then

$$\nabla f(x^*) = 0. \quad (2.6)$$

Theorem 3. Second order necessary condition

Let $\Omega \subset \mathbb{R}^n$, $f \in C^2$ a function on Ω , x^* a local minimiser of f over Ω , and d a feasible direction at x^* . If for all feasible direction vectors d at x^* , $d^T \nabla f(x^*) \geq 0$, then $d^T \nabla^2 f(x^*) d \geq 0$.

For the special case where x^* lies in the interior of Ω , we have the following result.

Corollary 2 (Interior case)

Let x^* be an interior point of $\Omega \subset \mathbb{R}^n$. If x^* is a local minimiser of $f : \Omega \rightarrow \mathbb{R}$, $f \in C^2$ then,

$\nabla f(x^*) = 0$ and the Hessian matrix $\nabla^2 f(x^*)$ is positive semi-definite.

Definition 18. A set C is said to be a **convex set** if for any $x_1, x_2 \in C$ we have that $\lambda x_1 + (1 - \lambda)x_2 \in C$ for all real λ such that $0 \leq \lambda \leq 1$.

Definition 19. A function f is said to be **convex** in Ω if for all $x_1, x_2 \in \Omega$, $\lambda \in [0, 1]$, we have $\lambda f(x_1) + (1 - \lambda)f(x_2) \geq f(\lambda x_1 + (1 - \lambda)x_2)$; and is called **quasiconvex** if for any $x_1, x_2 \in \Omega$ and $0 \leq \lambda \leq 1$, $\max\{f(x_1), f(x_2)\} \geq f(\lambda x_1 + (1 - \lambda)x_2)$. If we replace “ $\geq$ ” with “ $>$ ” then we say that f is strictly convex (quasiconvex, respectively).

2.3 Stackelberg Games

Heinrich von Stackelberg (1905-1946) was one of the most influential economists in Germany[40]. Stackelberg used a hierarchical model to describe real market situations. Stackelberg games

are non-symmetric games where one player or specified group of players have the privilege position to make a decision before the other players. The player who makes the decision first is called a leader and the other player is a follower, and this can be extended to multiple players (leaders/followers). The follower has knowledge of the leader's strategy and the leader is aware of this fact. The follower studies the leader's strategy and then chooses his own strategy based on the leader's actions[41][42][43][44]. A good example of a Stackelberg game is that of a security guard and a robber. A robber will study the security guard's movements and their patrol strategy before they attack. Following the robber's attack the security guard will be forced to beef up their security thereby changing their strategy and it will continue like this in a sequential manner. The Stackelberg games problem is formulated as follows: Let $x \in X$ and $y \in Y$ denote the set of admissible strategies of the leader and the follower respectively. The leader chooses their strategy x . The follower will then choose their strategy which is $y(x)$, where $y(x)$ is an element of the set of admissible strategies for the follower in response to the leader's choice of strategy x . The challenge from here now is to find the leader's optimal strategy taking into consideration that the follower acts in a rational manner and also wants to optimise their objective function. In this study we consider the leader as the traffic authority and we have multiple followers who are the drivers.

2.4 Bilevel Programming Problems

When we have two or more decision makers (players) involved in a situation, competition comes into play. The player who has the privilege to make the decision first is called the leader while the player who makes a decision based on the leader's actions is called a follower. Both players' decisions are codependent since the leader must make his choices bearing in mind how the follower may respond to his actions. Bilevel optimisation is an area of optimisation which deals with problems having a hierarchical structure where two decision makers are involved that is the leader and the follower. Bilevel programming has many applications which include, but not limited to transportation, network design and trip demand estimation[45], management- credit allocation[46], environmental policies[47] and machine learning[48].

2.5 The General Bilevel Programming Problem

Bilevel problems are of a hierarchical structure and involve two decision markers (players) a leader and a follower. The constraints of a bilevel problem are determined by another optimisation problem.

Definition 20. For $x \in X \subset \mathbb{R}^n$, $y \in Y \subset \mathbb{R}^m$, $F : X \times Y \rightarrow \mathbb{R}^1$ and $f : X \times Y \rightarrow \mathbb{R}^1$, $G : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$, $g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^q$, the general bilevel programming problem (BLPP) is of the

form,

$$\begin{aligned}
 & \min_{x \in X} F(x, y) \\
 & G(x, y) \leq 0, \text{ and } y \text{ solves} \\
 & \min_{y \in Y} f(x, y) \\
 & g(x, y) \leq 0.
 \end{aligned} \tag{2.7}$$

The sets X and Y are additional restrictions on the variables, such as upper and lower bounds or integrality requirements. Once a leader selects an x the first term in the follower's objective function becomes a constant. It is assumed that the sets X and Y are convex and compact and that F and f are continuous on $X \times Y$. Also the assumption is that f has a unique solution $y(x)$. That is $f(x, y)$ becomes $f(x)$ then y can be perceived as a function of x that is $y = y(x)$. The functions F, f, G, g are assumed to be continuous and twice differentiable.

Definition 21. The *Extended Constraint region* of the BLPP is the set of points $S = \{(x, y) : x \in X, y \in Y, G(x, y) \leq 0 \text{ and } g(x, y) \leq 0\}$.

Definition 22. The *follower's feasible set* for each $x \in X$ is given by $S(x) = \{y \in Y : g(x, y) \leq 0\}$ and the *follower's rational reaction set* is defined as

$$R(x) = \{y \in Y : y \in \operatorname{argmin}[f(x, \hat{y}) : \hat{y} \in S(x)]\},$$

where

$$\operatorname{argmin}[f(x, \hat{y}) : \hat{y} \in S(x)] = [y \in S(x) : f(x, y) \leq f(x, \hat{y}), \text{ for all } \hat{y} \in S(x)]$$

Definition 23. The *induced region* $\mathcal{R} = \{(x, y) \in S, y \in R(x)\}$ represents the feasible region of problem (2.7).

Definition 24. We say the point (x^*, y^*) is the *complete optimal solution* of problem (2.7) if and only if there exist $(x^*, y^*) \in S$ such that $F(x^*, y^*) \leq F(x, y^*)$ and $f(x^*, y^*) \leq f(x^*, y)$ for all $(x, y) \in S$.

It is important to note that the complete optimal solution does not always exist which leads us to the definition of the Pareto optimal solution.

Definition 25. We say the point (x^*, y^*) is a *Pareto optimal solution* of problem (2.7) if and only if there does not exist $(x, y) \in S$ such that $F(x, y) \leq F(x^*, y^*)$, $f(x, y) \leq f(x^*, y^*)$ and $F(x, y) \neq F(x^*, y^*)$ or $f(x, y) \neq f(x^*, y^*)$.

The following assumptions were made by Bard [49] to ensure that problem (2.7) has a Pareto optimal solution.

- S is non-empty and compact

- The follower's rational reaction set $R(x)$ is non-empty
- The follower's rational reaction map $R(x)$ is a point to point map.

2.6 KKT- reformulation to reduce the bilevel problem to single level

In this section we consider the following bilevel problem:

$$\begin{aligned} \min_{x,y} & F(x, y) \\ \text{s.t. } & y \in S(x), \\ & G_k(x, y) \leq 0, \quad k \in K, \end{aligned} \tag{2.8}$$

with $S(x)$ denoting the set of feasible solutions of the lower level problem:

$$\begin{aligned} \min_y & f(x, y) \\ \text{s.t. } & g_i(x, y) \leq 0, \quad i \in I, \end{aligned} \tag{2.9}$$

where F, f, G_k and g_i are continuously differentiable functions on $\mathbb{R}^n \times \mathbb{R}^m$ with finite index sets $I = \{1, 2, \dots, p\}$, $K = \{1, 2, \dots, q\}$.

The **Lagrangian** of problem (2.9) is given by,

$$\mathcal{L}(x, y, \lambda) = f(x, y) + \sum_{i \in I} \lambda_i g_i(x, y).$$

Definition 26. Let $x \in \Omega$, where Ω is the domain of the vector function $g = (g_1, \dots, g_p)$. Then the set

$$I_{ac} := \{1 \leq i \leq p \mid g_i(x) = 0\}$$

is said to be the **set of active inequality constraints** at x .

Definition 27. Linear independence constraint qualification

Let $I_{ac}(x^*), x^* \in \Omega$, be the set of active inequality constraints. Then the linear independence constraint qualification is satisfied at x^* if the set of active constraints gradients

$$\{\nabla h_1(x^*), \dots, \nabla h_m(x^*), \nabla g_i(x^*), i \in I_{ac}(x^*)\}$$

is linearly independent.

Then using the Lagrangian function, and assuming that the linearly independent constraint qualifications hold:

the KKT conditions of (2.9) becomes

$$\begin{aligned}\nabla_y \mathcal{L}(x, y, \lambda) &= 0 \\ \lambda_i g_i(x, y) &= 0, \quad i \in I \\ \lambda_i &\geq 0, \quad \text{with } g_i(x, y) \leq 0, \quad i \in I.\end{aligned}\tag{2.10}$$

If the lower level problem (2.9) is convex it can be replaced in (2.8) as follows

$$\begin{aligned}\min_{x, y, \lambda} & F(x, y) \\ & G_k(x, y) \leq 0, \quad k \in K \\ & g_i(x, y) \leq 0, \quad i \in I \\ & \nabla_y \mathcal{L}(x, y, \lambda) = 0 \\ & \lambda_i g_i(x, y) = 0, \quad i \in I \\ & \lambda_i \geq 0, \quad i \in I,\end{aligned}\tag{2.11}$$

where,

$\nabla_y \mathcal{L}(x, y, \lambda) = 0$ is the Lagrangian constraint and

$\lambda_i g_i(x, y) = 0$ is the complementary slackness condition for each $i \in I$.

Replacing the lower level problem with the Lagrangian results in a more practical way to solve the linear bilevel problem. This converts the bilevel problem into a nonlinear program which can now be solved by existing algorithms [50]. However for a non linear case, solving the transformed problem depends on the complexity of the involved functions. Particularly, for the problems with mixed integer constraints many of the existing solution methods fail to work.

3. Literature Review

Transportation problem models are divided into various categories and these are the Traffic Equilibrium problem, the Network Design problems and the Signal Setting problem. Traffic equilibrium problems are concerned with finding the user equilibrium and system optimum that is prediction of demand, flow and travel times. The Network design models deal with improving the network flow by increasing the road capacity and even sometimes by removing some links. In the signal setting problem the traffic lights are used to improve the flow without altering the capacity of the network. Although formulating models to solve the problem of congestion for dynamic flows is a difficult task, managing traffic congestion through construction of new infrastructure or widening of existing roadways to increase system capacity is expensive and unsustainable. Hence researchers focus on developing models for optimising traffic lights to route traffic efficiently on the existing roadways.

An important measure of effectiveness of a traffic signal is the level of delay created on the network due to the signal. Therefore, it is of great importance to develop models which estimate the delay. Estimating the delay is a concern to most researchers since travelers do not want to lose their time on the roadways. Transportation costs such as fuel consumption are also considered and most importantly reducing the delay contributes positively to the livelihood of travellers in general. At a signalised intersection vehicles may arrive in a uniform or random fashion. Also, accidents may occur, so this problem is both stochastic and deterministic in nature thereby making it a really difficult problem to solve. Wardrop developed a delay model for signalised intersection in 1952 [51] in which the author assumed uniform arrivals at the intersection. This was one of the first models for the delay at a signalised intersection. After six years Webster in 1958 developed a model for the delay at intersections and introduced the traffic signal settings [52]. Ever since this publication many researchers around the world have showed interest in working on signalised intersection network models. Researchers have applied this knowledge in various mechanisms and analysis. In this model Webster assumes random arrivals at a signalised intersection and uniform departure. Many theoretical analysis have been developed in order to predict the travel time functions in signalised intersections, queue length and waiting time at junctions in a quest to improve congestion. In addition this development by Webster also led to various microscopic models and algorithms to solve the traffic management problem for intersection control [53]. These models include among others the Miller model which was developed in 1963 [54], Hutchinson model in 1972 [55] and the Akcelik model which was introduced in 1980 [56]. All these models were formulated for homogeneous traffic conditions.

In their study Eriskin et al [13] used the eliminating pairing system to optimise the green interval for over saturated junctions. Stoilova et al in [57] used the bilevel model in the traffic management problem. They used the cycle length and green interval to model the problem. The results in their study indicate reduced queue lengths in front of junctions. The bilevel model remains a more efficient model to use as compared to the classical single optimisation model.

The authors state in their paper that they chose to use the bilevel model to investigate more parameters. They conclude that the bilevel model allows for a fair distribution of the waiting time between the vehicles in the network.

In [58] Stoilova et al formulated a bilevel model for intelligent traffic management. They focus on traffic lights green interval and the traffic lights cycle as the decision parameters of the upper level. The authors chose to employ the bilevel approach as opposed to the classical optimisation method to manage the traffic flows. It has been shown in their study that this approach proved better transportation conditions in the simulation results. In [59] The bilevel model is used to investigate mixed land use and urban road network. Feng and Wen formulate a fuzzy two goal function model to control traffic in a disaster affected area. They state that this approach allows them to model a real situation and to come up with appropriate traffic control measures to mitigate the problem. In [61] Zhao et al, developed a two goal function model to optimise intersection control and trajectory. The results in their paper show that using this approach can achieve better intersection traffic control. Patriksson and Rockafellar in [62] analysed upper level and lower level variables and formulated a descent algorithm to manage traffic flows.

On the other hand Chiou studied a signal controlled network in [63]. A bilevel model is formulated to solve the transportation problem in their study and they used a heuristic in their solution approach which involves a local search step and a global search heuristic. Whereas Cipriani and Fusco give a discussion on the global signal setting problem in [64]. In their paper they remarked that the signal setting problem is a complicated and non-convex optimisation problem, and they used the Armijo rule [65] for optimal step size in their estimation.

It is highlighted in [66] and [67] that meta-heuristic algorithms can be used for intersection control. This solution approach is more suited for some optimisation problems which involve signal control and which have a large solution space. Recent studies found out that the bilevel optimisation approach is efficient and effective because it allows for the use of a large number of variables, parameters and constraints that can be used in the problem [12][57]. The use of an increased number of control variables, parameters and goal functions will consequently lead to better results.

The delay at a signalised intersection is influenced by many factors and variables, therefore estimating the delay remains a difficult problem to solve. In [68] Preethi et al formulated a model for the delay at signalised intersections and they observed heterogeneous conditions. In their model formulation they modified the Webster delay formula by adding a field delay term for the inclusion of heterogeneous conditions. In their paper Hoque and Imran [69] modify the Webster formula for heterogeneous traffic conditions and they investigate non-lane based road traffic. Viti and Zuylen [70] develop a queue Markov model at signalised intersections in their paper, the mean, standard deviation and variance of the queue are determined and the authors state that these variables have emerged to be influential in the queue dynamics. In [71] Shariff and Faheem analysed the delay at signalised intersections under mixed traffic conditions.

Numerous studies have been performed to investigate the transportation problem with a single objective function with the goal to minimise the travel time and transportation cost[72][73].

There is a need to do more research on the transportation problem which explores optimisation models with multiple objective functions. A limited number of researchers have investigated this type of problem.

Previous studies have focused on using green interval or cycle length as a parameter [12][13][57]. Despite decades of research, the bilevel problem is still a complex problem to solve [74]. Although there are some studies which have previously addressed this topic, it remains that the research in the use of different parameters combined remains limited.

While many studies have focused on green interval and cycle length as the decision variable for the upper level some researchers have studied the optimisation of the offsets as a critical parameter in signal timing [75][76][77]. The offset is defined as the time between the start of the “green light” at one intersection and the start of “green light” at another intersection, and it defines the movement of traffic along major road. For example in [78] Kamijo et al investigate the offset optimisation of traffic signals to maximise green time overlap. Amini et al in [79] discuss a case study on optimising offsets in signalised traffic networks. Amin et al remark that optimising the offsets in an urban network reduces the delay incurred by vehicles.

This study addresses the consideration of different parameters which so far has been indicated by previous research that they have not been used and also the use of the bilevel model to solve the optimisation problem since many studies have addressed the transportation problem with only one objective function. One of the big challenges in this domain is solving the bilevel problem, as a result most researchers have focused on functions with properties which are more desirable to deal with, for example objective and constraint functions which are linear, smooth or convex [74].

TRANSYT software is one of the most widely used tool for optimising signal timings. TRANSYT is a time dependent macroscopic model which is also a group based optimisation of fixed-time signals. Different researchers have used TRANSYT software for traffic control at a single intersection and for large signal controlled road networks [80][81][82]. TRANSYT software has an inbuilt optimisation function which is based on performance index(PI). The performance index is a weighted average of delays, stops, progression and fuel consumption. This software uses manually entered fixed flows and assumes vertical queuing of traffic at the stop line hence it cannot accurately model spill-backs. Although this software is widely used, other researchers in the past who have studied the bilevel traffic signal control problem have chosen to use different algorithms and implementations for their results analysis [83][84][85][86]. However in our study we assume that individual vehicles (drivers) make their decisions based on prior knowledge, these decisions will also affect the capacity of the road. We use a bilevel model considering the decision of each driver in choosing their route and the control measures are implemented by the authorities.

The objective functions adopted in this study are partially convex and non linear functions. A different solution approach that involves systematic selection of the actions of the leader and using them to get a response from the lower level is used in this study. We solve the lower level problem by employing the BPSO- a global technique which unlike most optimisation techniques

does not require gradient information and is easier to implement.



4. Model Formulation

The model formulated in this study is a deterministic model. It is made up of a combination of discrete and continuous variables. A directed graph is used to represent the urban traffic network. The edges of the graph represent the links on the road network whilst the vertices represent the signal intersections. The cost function we use to calculate the travel time in both the upper and lower level is a non-linear function which is partially convex. Hence our two objective functions are both non-linear and non-convex.

4.1 The traffic network

Let graph $G = (N, A)$ represent the transportation network with N as the set of nodes and A as the set of links. The set of nodes N represents the street intersections. Each link $l_{ij} \in A$ is governed by a cost function. The cost function defines the delay incurred by the traffic due to the density on l_{ij} . Factors that determine the amount of time spent on a route include the following: traffic density, capacity and the free flow travel time. The cost function t_{ij} is also a function of the flow x_{ij} . We assume that the drivers behave in a rational manner, that is, they will choose to travel on the path with minimum cost given the prevailing traffic conditions due to the other road users. Thus, the drivers want to minimise the function $t_{ij}(x_{ij})$ for their own benefit. The traffic authority on the other hand wishes to minimise the total cost given by

$$\sum_{i,j}^{o,d} t_{ij}(x_{ij})x_{ij},$$

which represents the total time the traffic clears the road network.

In Figure 4.1, t_{ij} denotes the cost or the travel time on the link connected by n_i and n_j where $i, j = 1, \dots, 7$. Let the origin node o and destination node d be n_1 and n_7 respectively. Then the trip cost (travel time) from o to d is given by the minimum of f_{17} , where f is the objective of the driver.

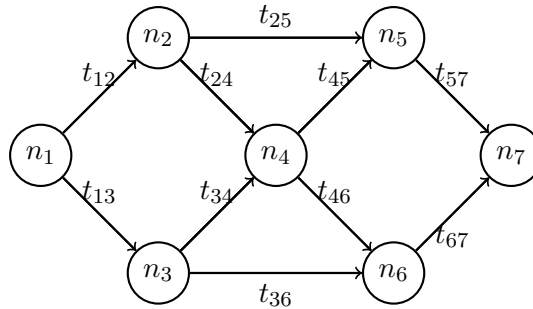


Figure 4.1: Weighted graph

Below we define some terms used in this study.

1. Traffic capacity(ca_{ij}): is the maximum number of vehicles that can fit on a segment of a road of length l_{ij} , measured in vehicles per kilometre (veh/km).
2. Free flow speed(u_{ij}): is the desired average speed that a motorist chooses to travel at if there is no congestion or other adverse conditions, it occurs under very low volumes of vehicles.
3. Volume(V_{ij}): there are many ways of defining traffic volume, but in this study traffic volume is defined as the number of vehicles passing at a specified point in a given time interval.
4. Traffic flow: is the study of the interactions between the drivers, the vehicles and the road infrastructure while we define the flow x_{ij} as the number of vehicles from node i to node j at a given time.
5. The offset is defined as the time between the start of the “green light” at one intersection and the start of “green light” at another intersection.
6. Cycle length (C_i): the cycle length at node i is the time it takes to complete one sequence of signal indications of a traffic light system. The cycle length also includes the offsets.
7. Traffic congestion: It occurs when the number of the incoming vehicles exceeds the number of the outgoing vehicles that is, when $\sum_{o,d} \delta_{i,j}^{o,d} > \sum_{o,d} \delta_{j,k}^{o,d}$ where $k > j$. In terms of capacity ca_{ij} congestion occurs when the flow exceeds the capacity that is, $x_{ij} > ca_{ij}$.

Table 4.1 gives a brief description of the parameters, variables and notations used in this study.

4.2 Description of the cost function

The travel cost function t_{ij} is a combination of the Bureau of Public Roads (BPR) formula [4] and the modified Webster equation [5] [6]. The BPR function computes the travel time on a link with respect to the flow and the capacity on that link, where the capacity ca_{ij} is the actual capacity on the road. The modified Webster equation estimates the delay at a signalised intersection. As a result the cost function computes the congested link travel time plus the waiting time at a signalised intersection. The cost function t_{ij} which we considered in this study is as follows:

$$t_{ij}(C_i, u_{ij}, x_{ij}) = \frac{l_{ij}}{u_{ij}} (1 + 0.15(y_{ij})^4) + \frac{9}{10} \left(\frac{C_i \cdot (1 - \lambda_i)^2}{2 \cdot (1 - \lambda_i \cdot y_{ij})} \right). \quad (4.1)$$

In the formula given in equation (4.1) the term $\frac{C_i \cdot (1 - \lambda_i)^2}{2 \cdot (1 - \lambda_i \cdot y_{ij})}$ gives the delay at the traffic signal when traffic is said to be arriving uniformly where C_i is the cycle length of traffic signal at node i . In this formula λ_i is the proportion of the green interval with respect to the cycle length of the traffic signal at node i and is given by $\lambda_i = \frac{g_i}{C_i}$, where g_i is given as the length of the green interval in seconds, the constant $\frac{9}{10}$ is the fraction of random vehicle headway [5][6].

It is essential to note that in equation (4.1) $\lambda_i \cdot y_{ij} < 1$. This is because $\lambda_i = \frac{g_i}{C_i}$ and $g_i < C_i$ hence

Variable/ Parameter	Description
ij	the direction from node i to node j on road ij
V_{ij}	the volume on link ij
ca_{ij}	the capacity of link ij (constant-given in veh/km)
y_{ij}	the volume to capacity ratio for the lane group ie $\frac{x_{ij}}{ca_{ij}}$
u_{ij}	free flow speed on link ij
l_{ij}	the section length (km) from node i to node j
C_i	cycle length for traffic signal i (in seconds)
g_i	the length of the green interval(in seconds) at traffic signal i
λ_i	is the proportion of the effective green with respect to the cycle length that is: $\frac{g_i}{C_i}$
$x_{i,j}$	flow (number of cars) on road ij from node i to node j
$\delta_{i,j}^{o,d}$	$\delta_{i,j}^{o,d} = 1$ if the link chosen lies on the path taken by the driver otherwise $\delta_{i,j}^{o,d} = 0$
o	origin
d	destination
$\sum_{o,d} \delta_{i,j}^{o,d}$	the total number of vehicles from the origin node o to the destination node d
S_o	supply at the source node
D_d	demand at the destination node
N_o	set of origin node
N_d	set of destination nodes
N_k	set of nodes at the k^{th} position for $k = 1, 2, 3, \dots$
$r_{o,d}$	the ratio of vehicles whose origin is o that will be destined to d

Table 4.1: Description of variables and Parameters.

$\lambda_i < 1$, while $y_{ij} = \frac{x_{ij}}{ca_{ij}}$ since x_{ij} , the flow, cannot exceed the capacity we have that $y_{ij} \leq 1$. Now since both λ_i and y_{ij} are less than 1 we conclude that the product $\lambda_i \cdot y_{ij}$ is less than 1, that is $1 - \lambda_i \cdot y_{ij} \neq 0$. Hence the travel time function t_{ij} is continuous.

We make an assumption that $g_i = \frac{1}{4}C_i$ hence equation (4.1) becomes

$$t_{ij}(C_i, u_{ij}, x_{ij}) = \frac{l_{ij}}{u_{ij}} (1 + 0.15(y_{ij})^4) + \frac{81}{160} \left(\frac{C_i}{2 - \frac{1}{2}y_{ij}} \right). \quad (4.2)$$

The term $\frac{l_{ij}}{u_{ij}}$ computes the free flow travel time that is the time it takes to traverse a link when a vehicle is traveling at a speed permitted by the traffic Authority when there is no congestion or adverse conditions.

The volume to capacity ratio for the lane group is given by $y_{ij} = \frac{x_{ij}}{ca_{ij}}$ where x_{ij} and ca_{ij} is given as the flow and the actual capacity on a link ij respectively. In the BPR formula $\frac{l_{ij}}{u_{ij}} (1 + 0.15(y_{ij})^4)$ the constant 0.15 which is referred to as the scale parameter is the ratio of the speed at practical capacity to that at free flow travel time. The index 4 which is referred to as the shape parameter and it determines how fast the curve increases when the capacity is reached [87]. The flow is a function of $\delta_{ij}^{o,d}$ and it is calculated by the formula

$$x_{ij} = \sum_{o,d} \delta_{ij}^{o,d} r_{o,d},$$

where $r_{o,d}$ is the ratio of all vehicles that originate from the o are destined to d .

In this study, interactions between the drivers and traffic authority in the traffic road network are described by a Stackelberg leader-follower game. The traffic authority makes all laws and policies and the drivers are compelled to comply with these rules and regulations.

4.3 Lower level decision maker: The Driver's problem

In this problem the driver is a follower, and they seeks to minimise their own travel time and so they behave in a rational manner. The driver will start his journey at any of the source nodes to their preferred destination nodes. When choosing which path to use, the driver will choose to travel on the link ij with the least travel cost. When traveling through a link from node i to node j the travel cost function t_{ij} will compute the travel time required for the driver to traverse the link. In addition the waiting time at node j will also be calculated. The number of cars traveling on the same link ij is given by the summation

$$\sum_{o,d} \delta_{ij}^{o,d},$$

where $\delta_{ij}^{o,d}$ is the choice of a driver, whose origin is at o and destined at node d , to use the link ij , which takes the value 1 if the driver chooses to use the link and 0 otherwise. We assume

that all the drivers are independent and this assumption will allow us to add the cost of the drivers in the objective function for the lower level [88]. The driver is much more interested in reaching their destination in the shortest possible time not considering other drivers on the roadway and does not cooperate with the other drivers. It is also assumed that the driver can calculate the cost on each link perfectly, so they make decisions on which route to use based on that knowledge. Therefore the driver seeks to minimise:

$$\begin{aligned}
 \min_{\delta_{i,j}^{o,d}} f(\delta_{i,j}^{o,d}, x_{ij}) &= \sum_{o,d} \sum_{i,j} t_{ij}(C_i, u_{ij}, x_{ij}) \delta_{i,j}^{o,d} \\
 \text{subject to} \\
 \sum_{i \in N_1} \delta_{o,i} &= 1 \quad \text{for each } o \text{ (from origin node to intermediate nodes)} \\
 \sum_{i \in N_{k-1}} \delta_{i,k} &= \sum_{j \in N_{k+1}} \delta_{k,j} \quad \forall i = 1, 2, 3, \dots, k-1 \\
 &\quad \forall j = k+1, k+2, k+3, \dots, m \\
 &\quad \text{(intermediate nodes)} \\
 \sum_{i \in N_k} \delta_{i,d} &= 1 \quad \forall i = 1, 2, 3, \dots, k \text{ (from intermediate to destination nodes)} \\
 &\quad \text{for each } d \text{ (where } d \text{ is the destination node)}
 \end{aligned} \tag{4.3}$$

4.4 Upper level decision maker: The Traffic Authority's Problem

The traffic authority in this problem is considered to be the leader. The authority decides on and controls the cycle length of the traffic signals at the different intersections of the network. They also decide on the maximum speed limit on each roadway. These factors influence the rate of flow in the network. Hence the authority indirectly influences the choices of the drivers. The objective of the traffic authority is to minimise the total travel time in the whole network to make the conditions as smooth and as safe as possible for the drivers.

The authority computes all the travel times from all the source nodes to the destination nodes within the network, in particular, the travel time on the links and the waiting time at the nodes in the whole network using the travel cost function t_{ij} . The travel cost in each link will be summed up and this summation will be multiplied by the corresponding flow x_{ij} on the links. Now since we know that the authority seeks to minimise the total cost, the question is how does the authority achieve this goal. The answer to this question is that the authority will minimise the total cost by minimising the total travel time $t_{ij}(x_{ij})x_{ij}$ of the whole network. What this means is that the authority's desire is for all the drivers to spend minimum time on the roadway and to make the flow as smooth as possible. The authority will do this by controlling or varying the cycle length of the traffic signal at node i and setting the optimal maximum speed permissible to the driver. In this sense, the traffic authority seeks to minimise

the following objective function:

$$\begin{aligned}
 \min_{C_i, u_{ij}} F(C_i, u_{ij}, x_{ij}) &= \sum_{i,j} t_{ij}(C_i, u_{ij}, x_{ij}) x_{ij} \\
 \text{subject to} \\
 u_{ij}^{lower} \text{ km/hr} &\leq u_{ij} \leq u_{ij}^{upper} \text{ km/h}, & \text{for each } i, j \\
 C_i^{lower} \text{ seconds} &\leq C_i \leq C_i^{upper} \text{ seconds} & \text{for each } i \\
 \sum_{j \in N_1} x_{oj} &= S_o, \text{ for each } o \in N_o \\
 \sum_{i \in N_k} x_{id} &= D_d, \text{ for each } d \in N_d \\
 \sum_o S_o &= \sum_d D_d, \text{ for each } o, d \text{ (conservation laws)},
 \end{aligned} \tag{4.4}$$

where $\delta_{i,j}^{o,d}$ is the solution of the lower level problem and the flow x_{ij} is defined as:

$$\begin{aligned}
 x_{ij} &= \sum_{o,d} \delta_{i,j}^{o,d} r_{o,d} \quad \forall i \in N_i \\
 &\quad \forall j \in N_j \text{ with } j = i + 1.
 \end{aligned} \tag{4.5}$$

4.5 The Bilevel problem

In this formulation, the objective function of the traffic authority is to minimise the travel time for the whole network system while the drivers objective function is treated as a constraint. The bilevel problem is a mixture of discrete and continuous variables. The cost function t_{ij} is also not a fully convex function making this problem even more difficult to solve. We cannot apply discrete or continuous methods only to solve this problem hence it will be difficult. As a result we employ a meta-heuristic method to solve this problem which is elaborated in detail in Chapter 5.

We have the following bilevel model formulation for the authority and the driver:

$$\begin{aligned}
 \min_{C_i, u_{ij}} F(C_i, u_{ij}, x_{ij}) &= \sum_{i,j} t_{ij}(C_i, u_{ij}, x_{ij}) x_{ij} \\
 \text{subject to} \\
 u_{ij}^{lower} \text{ km/hr} &\leq u_{ij} \leq u_{ij}^{upper} \text{ km/h,} & \text{for each } i, j \\
 C_i^{lower} \text{ seconds} &\leq C_i \leq C_i^{upper} \text{ seconds} & \text{for each } i \\
 \sum_{j \in N_1} x_{oj} &= S_o, \text{ for each } o \in N_o \\
 \sum_{i \in N_k} x_{id} &= D_d, \text{ for each } d \in N_d \\
 \sum_o S_o &= \sum_d D_d, \text{ for each } o, d \text{ (conservation laws),}
 \end{aligned} \tag{4.6}$$

and $\delta_{i,j}^{o,d}$ solves the following problem

$$\begin{aligned}
 \min_{\delta_{i,j}^{o,d}} f(\delta_{i,j}^{o,d}, x_{ij}) &= \sum_{o,d} \sum_{i,j} t_{ij}(C_i, u_{ij}, x_{ij}) \delta_{i,j}^{o,d} \\
 \text{subject to} \\
 \sum_{i \in N_1} \delta_{o,i} &= 1 & \text{for each } o \text{ (from origin node to intermediate nodes)} \\
 \sum_{i \in N_{k-1}} \delta_{i,k} &= \sum_{j \in N_{k+1}} \delta_{k,j} & \forall i = 1, 2, 3, \dots, k-1 \\
 & & \forall j = k+1, k+2, k+3, \dots, m \\
 & & \text{(intermediate nodes)} \\
 \sum_{i \in N_k} \delta_{i,d} &= 1 & \forall i = 1, 2, 3, \dots, k \text{ (from intermediate to destination nodes)} \\
 & & \text{for each } d \text{ (where } d \text{ is the destination nodes),}
 \end{aligned} \tag{4.7}$$

where the flow x_{ij} is defined as:

$$\begin{aligned}
 x_{ij} &= \sum_{o,d} \delta_{i,j}^{o,d} r_{o,d} \quad \forall i \in N_i \\
 & \quad \forall j \in N_j \text{ with } j = i + 1.
 \end{aligned} \tag{4.8}$$

5. Solution Approach

In the solution approach we use a meta-heuristic algorithm to solve the bilevel problem. A global optimisation technique that does not require differentiability of a function has been adopted to solve the lower level problem in this study. We chose this technique because the lower level problem is binary and therefore it will not be possible to find its derivative thus we cannot apply the derivative method solution approach. In particular we use systematic sampling approach to select the leader's actions from the partitioned decision space. The proposed method for solving the lower level problem is the binary particle swarm optimisation (BPSO) technique. The two methods are elaborated in this chapter.

5.1 A systematic sampling numerical method for the bilevel model

The systematic sampling method for bilevel models was first introduced by Goshu and Kassa [89]. We have adopted this method because the cost function t_{ij} is not linear. Our problem is a mixed integer model and the derivative of the cost function t_{ij} is not easy to determine and in addition the functions involved in t_{ij} are not explicit. The flow x_{ij} is implicit in y_{ij} and the term $\delta_{i,j}$ in the lower level problem is implicit in x_{ij} . Our problem is an inductive-type problem. Furthermore, the lower level problem is binary so it is not differentiable and the cost functions are functions of mixed integer variables. Using the derivative method solution approach is difficult to implement, since the cost functions are non-linear, which are defined inductively with more than one layer, and the lower level problem is made up of binary variables which makes the entire problem non-differentiable. Therefore we opted to use this method.

5.1.1 Partitioning the decision space of the upper level problem

In the upper level problem the traffic authority desires to improve the transportation conditions by minimising the travel time of the entire network. The total travel time is found by summing the travel time in each link ij and the waiting time at each node. To find the flow in a link the travel time is multiplied by the total number of vehicles per unit time in that link, and this gives us the flow at that instance. We compute this total number of vehicles by evaluating the sum $\sum_{o,d} \delta_{i,j}^{o,d}$.

Figure 5.1 shows a hyperbox with dimensions of the variables of the leader and Figure 5.2 shows how the leader's variables will be partitioned so as to choose randomly actions for the leader from each partition. We suppose that the decision variables of the leader are in a closed box region in a finite dimensional Euclidean space, as shown on Figure 5.1. The leader may pick any point in the hyper-box on Figure 5.1 resulting in a vector of the form (C_i, u_{ij}) for each i and for each j depending on the number of links and intersections in the network investigated.

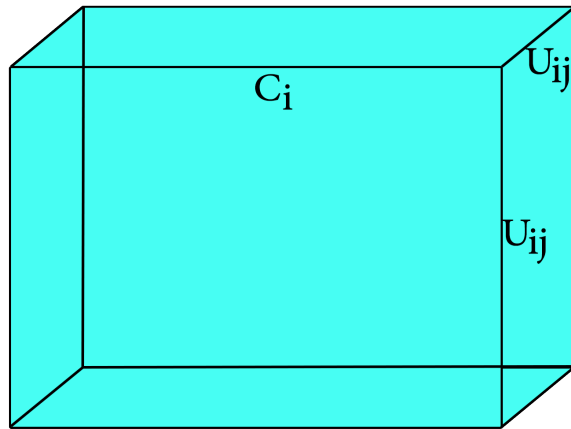


Figure 5.1: The Leader's decision space diagram.

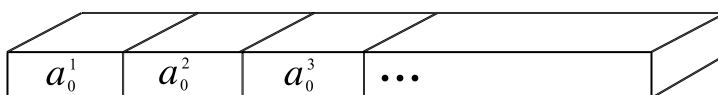


Figure 5.2: Partitioning decision space of the leader.

The procedure is explained in the following steps:

1. When using this procedure the first step is to partition the decision space of the leader in to small hyper-box cells $a_0^1, a_0^2, \dots$ as shown on Figure 5.2.
2. A point x_0^i which represents the actions of the leader is chosen randomly from each of the partitions a_0^i .
3. Each one of these points (x_0^i) which represent the actions of the leader are then substituted in the lower level objective function to get a response from the lower level.
4. For each the of these points (x_0^i) the lower level optimization problem is solved (using an appropriate optimization method) to get a corresponding response (y_0^i) from the lower level.
5. Next the value of upper level objective function will then be evaluated and the point $(\bar{x}_0^i, \bar{y}_0^i)$ which gives the best upper level objective will be selected and stored.
6. Then another set of different points from the same partitions is randomly selected again and the procedure above is repeated.
7. At iteration in step 5, the values of the upper level objective function will be compared with the value from step 3.
8. The point which gives the best cost is selected and stored.
9. This procedure is repeated until the maximum number of iterations is reached.

The accuracy of this method can be improved by subdividing a_0^i into more partitions as shown on Figure 5.2 and increasing the maximum number of iterations. This improves the results as the leader will have more actions to choose from. Another way of improving the results is by reducing the value of parameter beta in the MATLAB code. The downside to having more partitions and an increased number of iterations is that the algorithm will take a longer time to converge. To take care of the other constraints in the upper level problem (4.6) we define the augmented Lagrangian for the upper problem,

$$\begin{aligned} \mathcal{L}(C_i, u_{ij}, x_{ij}, \mu) = & \sum_{i,j} t_{ij}(C_i, u_{ij}, x_{ij})x_{ij} + \mu_1 \left(\sum_{j \in N_1}^m x_{oj} - S_o \right) + \\ & \mu_2 \left(\sum_{i \in N_k}^n x_{id} - D_d \right) + \mu_3 \left(\sum_o S_o - \sum_d D_d \right) + \mu_4 \left(x_{ij} - \sum_{o,d} \delta_{i,j}^{o,d} r_{o,d} \right) \quad (5.1) \end{aligned}$$

where,

μ_1, μ_2, μ_3 and μ_4 are Lagrange multipliers.

5.1.2 The lower level problem

In this problem the driver desires to take as less time as possible when traveling. Therefore the lower level objective is to minimise the cost function where the cost is the time it takes to traverse a link plus the waiting time at a node. Hence the cost is basically the driver's travel time. We compute this by finding the travel time of a vehicle from the origin node- o to the destination node d , and assign these vehicles the objective function $f_{o,d}$. The total number of vehicles on this route from o to d , is found by computing the sum $\sum_{o,d} \delta_{i,j}^{o,d}$.

In the lower level problem we compute the travel time for the objective function f_{n_1,n_8} on the path (n_1, n_8) up to the last objective f_{n_3,n_9} which is on the path (n_3, n_9) . Since we know the travel time for one origin to one destination the traffic authority can now make informed decisions. While the vehicles may have different origins and destinations, it is important to note that there are some instances where they will share some links (paths). This is explained more in the example given in Section 5.3.

The augmented Lagrangian for the lower level problem (4.7) is defined by equation (5.2):

$$\begin{aligned} \mathcal{L}(\delta_{i,j}^{o,d}, x_{ij}, \gamma) = & \sum_{o,d} \sum_{i,j} t_{ij}(C_i, u_{ij}, x_{ij}) \delta_{i,j}^{o,d} + \sum_o \gamma_o \left(\sum_{i \in N_1} \delta_{o,i}^{o,d} - 1 \right) \\ & + \sum_k \gamma_k \left(\sum_{i \in N_{k-1}} \delta_{i,k}^{o,d} - \sum_{j \in N_{k+1}} \delta_{k,j}^{o,d} \right) + \sum_d \gamma_d \left(\sum_{i \in N_k} \delta_{i,d}^{o,d} - 1 \right), \quad (5.2) \end{aligned}$$

for all $i = 1, 2, \dots, k-1, k, k+1, \dots, m$, and for each o and d , where, γ_o, γ_k , and γ_d are lagrange multipliers for each given index.

Instead of minimising the constrained problem (4.7) we minimise the augmented Lagrangian (5.2) using binary particle swarm optimisation, a technique which we elaborated in Section 5.2.

The following assumptions are made for the lower level problem.

A1: The driver does not have prior knowledge of the route choice of the others, therefore the drivers optimise their travel times independently[90].

A2: The drivers are rational and they can calculate the cost on each link perfectly.

A3: The route choice of one driver is not affected by the route choice of another driver hence the drivers optimise their travel times independently.

A4: All drivers know their destinations before they depart.

A5: We assume that the total travel time from an origin node to the destination node for a certain path is the same for each driver.

5.1.3 The bilevel optimisation problem

The bilevel problem is continuous and deterministic. Let there be a hyper box. The hyper box is three dimensional and made up of the decision variables of the leader (C_i, u_{ij}) for each i and each j depending on the number of intersections and links on the network. These points are all feasible solutions for the traffic authority. After picking a point from the hyper box using systematic random sampling, what remains is to now solve the lower level problem. We substitute the chosen value into the lower level problem to get the driver's response. We use a global optimisation technique called Binary Particle Swarm Optimisation to get a response from the driver. The process explained here is carried out for different actions of the leader. The results of each action of the leader are compared to get the best optimal solution which is the response that will yield the minimum value for the upper level objective function.

5.2 Binary particle swarm optimisation

Particle swarm optimisation (PSO) is a global optimisation technique for unconstrained problems. Unlike many optimisation techniques, this method may be used for problems which are non-convex or not differentiable or non-smooth. It does not require gradient information. It is easier to implement and is useful when dealing with complex and nonlinear problems. This method is also suitable for solving discrete and combinatorial problems. This technique was first presented by James Kennedy and Russel Eberhart in 1995 [20] who were inspired by Grenander and Heppner (1990) [91]. These authors were working on computational intelligence and they did not only want to focus on the cognitive skills but rather also on the social interaction. PSO is based on how animals behave when they are in a group, for example, a school of fish or a flock of birds. They were inspired by the way the birds move together and the distance they keep from each other while searching for food and how they separate from the swarm and eventually regroup again.

PSO algorithm is described below

1. Initialise constants c_1, c_2 and set $k = 0$
2. Initialise the particles x_i^k
3. Initialise randomly particles velocities v_i^k
4. Evaluate each particles position x_i^k considering the objective function that is $f(x_i^k)$.
5. Initialise the particles's best position x_i^{best} and the global best value of the swarm x_g^{best}
6. If a particle's present position is better than its previous best position, update it, that is if:

$$f(x_i^k) \leq f(x_i^{best}) \text{ then } f(x_i^{best}) = f(x_i^k)$$

7. Find the best particle (according to the particle's last best places).
8. If the particle's current position is better than the swarm global best update it. That is

if:

$$f(x_i^k) \leq f(x_g^{best}) \text{ then } f(x_g^{best}) = f(x_i^k)$$

9. Update particles' velocities using (5.3).
10. Move particles to their new positions using: $x_i^{k+1} = x_i^t + v_i^{k+1}$
11. Set $k = k + 1$
12. Go to step 2 until the stopping criteria is satisfied.
13. Terminate

The PSO method makes use of the velocity vector to update the current position of each particle. The position of each particle is updated based on the behaviour of the members of the swarm as a whole and the memory of each particle. This process is stochastic in its nature. The implementation of the PSO begins with an initial population size. The positions of the members of the swarm are randomly distributed with a random initial velocity vector. Then the velocity vector is calculated and the position of each particle (member of the swarm) is updated using the previous position and the new velocity vector. This process is repeated until the algorithm converges to a solution.

The BPSO which is a variant of the PSO is used to solve the lower level problem. This method was introduced by Kennedy and Eberhart in 1997 [92]. The BPSO is a binary version of the PSO and it is applicable for binary cases. It differs with the PSO in that only the velocity is used when updating the position of the BPSO, that is the position of the BPSO does not rely on the current position unlike in the PSO [93]. On the other hand the BPSO's personal best and global best is updated in the same manner they are updated in the continuous PSO [94]. The position of a particle in BPSO is updated to either 1 or 0 depending on a probability value which is calculated by transforming the velocity by a sigmoid function. Under the BPSO the velocity can only take values within the closed interval $[V_{\min}, V_{\max}]$, where $V_{\max}$ is the maximum velocity, $V_{\min}$ is the minimum velocity and $V_{\min} = -V_{\max}$. A piece-wise linear function that ranges in this interval is used to keep the velocity values within this range [95][96].

According to [67]:

- the new velocity is given by,

$$v_i^{\bar{d}}(\bar{k} + 1) = wv_i^{\bar{d}}(\bar{k}) + c_1\bar{r}_1(pbest_i^{\bar{d}}(\bar{k}) - \bar{x}_i^{\bar{d}}(\bar{k})) + c_2\bar{r}_2(gbest^{\bar{d}}(\bar{k}) - \bar{x}_i^{\bar{d}}(\bar{k})), \quad (5.3)$$

where the description of the variables and parameters used in equation (5.3) is given in Table 5.1.

The velocity in equation (5.3) is converted to a probability value using equation (5.4).

$$\bar{s}(v_i^{\bar{d}}(\bar{k} + 1)) = \frac{1}{1 + \exp(-v_i^{\bar{d}}(\bar{k} + 1))} \quad (5.4)$$

Variables/Parameter	Description
$\bar{i}$	the order of the particle in the population
$\bar{x}$	the position of the particle
$\bar{d}$	the dimension of the search space
w	inertia of the particle
$\bar{r}_1, \bar{r}_2$	random numbers between 0 and 1
c_1, c_2	are acceleration factors
$pbest$	personal best solution
$gbest$	global best solution for the whole population

Table 5.1: Binary PSO parameters

- The position of each particle is updated as follows,

$$\bar{x}_{\bar{i}}^{\bar{d}}(\bar{k} + 1) = \begin{cases} 1, & \text{if } rand < \bar{s}(v_{\bar{i}}^{\bar{d}}(\bar{k} + 1)) \\ 0, & \text{otherwise} \end{cases} \quad (5.5)$$

where $rand$ is a random number uniformly distributed between 0 and 1.

- The personal best given by $pbest$ and global best given by $gbest$ of the swarm are calculated using:

$$pbest_{\bar{i}}(\bar{k} + 1) = \begin{cases} \bar{x}_{\bar{i}}(\bar{k} + 1), & \text{if } rand < \phi(\bar{x}_{\bar{i}}(\bar{k} + 1)) < \phi(pbest_{\bar{i}}(k)) \\ pbest_{\bar{i}}(k), & \text{otherwise} \end{cases} \quad (5.6)$$

and

$$gbest(\bar{k} + 1) = \begin{cases} pbest_{\bar{i}}(\bar{k} + 1), & \text{if } < \phi(pbest_{\bar{i}}(\bar{k} + 1)) < \phi(gbest(k)) \\ gbest(k), & \text{otherwise} \end{cases} \quad (5.7)$$

where ϕ is the fitness function [67] which is the objective function value for a particle [97] and it measures how close or how “fit” a given solution is to the optimal solution, k is the number of iterations.

The binary particle swarm optimisation algorithm is given in **Algorithm 3**.

Algorithm 1: Main pseudo code for solving the bilevel problem

- 1: Select leader's strategy using systematic sampling technique; use Algorithm 2.
 - 2: Search potential optimal reaction for each of the leader's action.; use Algorithm 3.
 - 3: Substitute back the optimal reactions from 2 for each of the leader's actions and compute the optimal solution for the bilevel problem.
 - 4: Repeat steps 1 to 3 up to the stopping criteria. The stopping criteria is the maximum number of iterations.
-

Algorithm 2: Pseudo code for solving the realised bilevel programming problem.

- 1: Consider a bilevel programming problem with bounded decision variables input:
 $c_i^{lower}, c_i^{upper}, u_{ij}^{lower}, u_{ij}^{upper}, \omega$
 Algorithm parameters:
 β - partition parameter value
 $\bar{b}_1$ - number of partitions on $b_1^j(k)$ (where $b_1^j(k)$ is formed by merging any two consecutive a_0^i)
- 2: set iteration $\bar{k}=1$
- 3: Partition leader's decision space given by the hyperbox on Figure 5.2.
- 4: **for** $q = 3 : 2 : 2\bar{b}_1 - 1$ **do**
- 5: $x_1^j(q) = x_1^j(q - 2) + 2\beta$
- 6: $x_1^j(q + 1) = x_1^j(q - 1) + 2\beta$
- 7: For each leaders actions (C_i, u_{ij}) Find the optimal reaction $(\delta_{i,j}^{o,d})$ using the appropriate global technique (BPSO)
- 8: Select the best solution using leader's objective function from equation (5.1) such that:

$$F(C_i^*, u_{ij}^*, \delta_{i,j}^{o,d*}) \leq F(C_i, u_{ij}, \delta_{i,j}^{o,d})$$

Then set the best solution $F(C_i^*, u_{ij}^*, \delta_{i,j}^{o,d*})$ as the solution for iteration k .

- 9: Update the best solution at each iteration by comparing it with the previous one
 - 10: Terminate if $k = \omega$ ▷ maximum number of iterations
 - 11: Else
 - 12: Set $k = k + 1$ and go to step 3
 - 13: Output= $(\delta_{i,j}^{o,d*}, u_{ij}^*, C_i^*)$
-

Algorithm 3: Binary particle swarm optimisation algorithm(BPSO)

1: Initialisation

Consider the follower's problem with binary decision variables.

Input: Npop, ω

Algorithm parameters: c_1, c_2, r_1, r_2, w

- Set $k = 0$
- Initialise the positions of the particles $\bar{x}_i^{\bar{d}}$
- Initialise the velocities of the particles $v_i^{\bar{d}}$
- Initialise algorithm constants $c_1, c_2, \bar{r}_1$ and $\bar{r}_2$

2: Evaluate objective function for each particle $\bar{x}_i^{\bar{d}}$ 3: Initialise the particles best position $pbest$ and the whole population global best $gbest$

4: Update iteration counter

- $k = k + 1$

5: Calculate the change in velocity $v_i^{\bar{d}}(\bar{k} + 1)$ using (5.3) and then update the velocity for each vector using the piece-wise linear function equation.

- $v_i^{\bar{d}}(\bar{k} + 1) = wv_i^{\bar{d}}(\bar{k}) + c_1\bar{r}_1(pbest_i^{\bar{d}}(\bar{k}) - \bar{x}_i^{\bar{d}}(\bar{k})) + c_2\bar{r}_2(gbest^{\bar{d}}(\bar{k}) - \bar{x}_i^{\bar{d}}(\bar{k}))$

6: Update the particles positions using the sigmoid function (5.5)

7: Evaluate the objective function for each particle $\bar{x}_i^{\bar{d}}(\bar{k} + 1)$ 8: Update $pbest$ and $gbest$ using (5.6) and (5.7) respectively.9: Terminate if $k = \omega$

▷ maximum number of iterations.

10: Else

11: Set $k = k + 1$ and go to step 4.

5.3 A particular example of the Authority and Driver's problem

In this example we have two decision makers. The Traffic authority which we call the upper level decision maker (or a leader) and the drivers which we call the lower level decision maker (or followers). In Figure 5.3 we have a network which represents the section of a town. There are three origin nodes which can be viewed as the starting points of the drivers' journey or the entry points into the town. And we have two end points which are the drivers' destination points or the exit points of the town section. In the middle of the town there are some points as shown on Figure 5.3 which connect the entry points with the exit points, the drivers are not allowed to park at these points, and we assume that there is a traffic signal at each of these points. In this problem the Traffic authority sets and controls the maximum speed allowed on the roadways including the cycle length at each traffic signal. We call these two characterising values the decision variables of the upper level problem. The traffic authority desires to come up with the best value of these two decision variables for a "smooth" running of traffic flow, better traffic conditions and to minimise the travel time for the entire network. The driver will in turn respond to the traffic authority's actions by choosing which routes they want to use based on the fact that they want to spend as minimum travel time on the roadways as possible. After evaluating all the responses of the driver the traffic authority will then settle for the representative which gives them the most desired response from the driver. This is what we call a bilevel optimisation problem.

Figure 5.3 of a transshipment network will be used to elaborate the problem for the driver and the authority. The figure is made up of 9 nodes represented by the vertices and 14 links represented by the edges. There are 3 source nodes coloured in yellow and 2 sink nodes in green and 4 intermediate nodes in red and blue. In a case where one considers traffic on a two-way road (street), the flow on a link for both directions will be equal. The only difference would be the direction of the vehicles, this difference will then be indicated by putting a negative sign on the flow-quantity

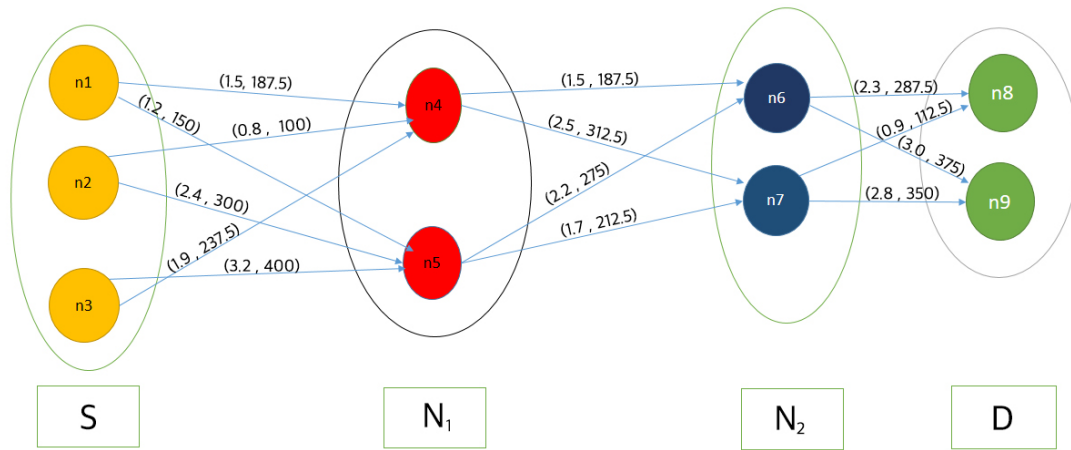


Figure 5.3: Example of a network.

When calculating the capacity (veh/km) for each link we assumed that a vehicle is on average 4 meters long while there is a space of 2 meters on the front and behind every vehicle. In this example, we consider any given number of cars leaving the origin nodes- n_1, n_2 and n_3 , then by the conservation laws we have the same amount of vehicles destined to the sink (destination) nodes n_8 and n_9 . In this regard we assign the origin nodes, n_1, n_2 and n_3 with the following proportion of vehicles: $n_1 = \alpha_1, n_2 = \alpha_2$ and $n_3 = \alpha_3$, where α_1, α_2 and α_3 are the number of vehicles departing from the origin nodes respectively. In the results section, three different cases are observed. These cases are varied by focusing on the total number of vehicles departing the origin nodes and different proportions.

The demand at nodes n_8 and n_9 obeys the conservation laws and so the summation of the vehicles at the source nodes equals the summation of the vehicles at the sink nodes. The length and the capacity of each link (l_{ij}, ca_{ij}) has been given in Figure 5.3 such that, for example, on the link connecting n_1 to n_4 , we have (1.5, 187.5) indicating that the link which connects n_1 to n_4 has a length of $l_{14} = 1.5$ km and a capacity of $ca_{14} = 187.5$ veh/km. Similarly, on the link connecting n_1 to n_5 , we have (1.2, 150) which indicates that the length of the road section $l_{15} = 1.2$ km and the capacity $ca_{15} = 150$.

5.3.1 The lower level problem (Driver's problem)

From the given example and the network provided in Figure 5.3, we have drivers who are traveling from any one of the origin nodes - n_1, n_2 or n_3 to any destination nodes n_8 or n_9 , we assume that the drivers can calculate perfectly the travel cost of the links they intend to use. The drivers are free to choose any route they intend to use and their decisions are not dependent on the decisions of the other drivers. In this regard we assume that the driver behaves in a rational manner and so they seek to minimise their own travel time. In the given example in Section 5.3, f_{n_1, n_8} is the objective function for the driver whose origin node is n_1 and destination node is n_8 .

5.3.2 Lower level objective

As mentioned before, each of the drivers seek to minimise their own travel time and it is assumed that they are independent decision makers. That means anyone of the routes they decide to travel on is not affected by the decision of the other drivers concerning which routes they choose to use.

The cost function t_{ij} of the lower level problem is given (5.8),

$$t_{ij}(C_i, u_{ij}, x_{ij}) = \frac{l_{ij}}{u_{ij}} (1 + 0.15(y_{ij})^4) + \frac{81}{160} \left(\frac{C_i}{2 - \frac{1}{2}y_{ij}} \right). \quad (5.8)$$

When we consider the link (n_1, n_4) in this particular example, we have $l_{14} = 1.5$ and $y_{14} = \frac{x_{14}}{187.5}$, and the equation for calculating the cost of this link connecting n_1 to n_4 consequently becomes (5.9),

$$t_{14}(C_i, u_{ij}, x_{ij}) = \frac{1.5}{u_{14}} \left(1 + 0.15\left(\frac{x_{14}}{187.5}\right)^4 \right) + \frac{81}{160} \left(\frac{C_4}{2 - \frac{1}{2}\frac{x_{14}}{187.5}} \right). \quad (5.9)$$

Here, x_{14} which is the value of the flow on link (n_1, n_4) can be obtained by equation (5.10).

$$x_{14} = \alpha_1 \cdot r_{1,8} \delta_{1,4}^{1,8} + \alpha_1 \cdot r_{1,9} \delta_{1,4}^{1,9}, \quad (5.10)$$

where α_1 is the total number of vehicles originated at node n_1 , $r_{1,8}$ and $r_{1,9}$ is the ratio of vehicles destined to nodes n_8 and n_9 respectively.

Table 5.2 shows the variables and parameters used in the lower level problem.

Variables	Parameters
$\delta_{i,j}^{o,d}$ - number of cars from the origin node to the destination node	ca_{ij} - capacity on road ij
	l_{ij} - section length (km) from node i to node j

Table 5.2: Parameters and variables - lower level.

The objective of each driver is to choose a route that minimises the travel time from their origin to the respective destination subject to the required constraints. We assume that the total travel time from an origin node to the destination for a certain path is the same for each driver. The driver optimises the following objective functions subject to the given constraints depending on the path they choose to take as shown on Figure 5.3.

$$\begin{aligned} \min_{\delta^{1,8}} f_{n1,n8} = & (t_{n1,n4}\delta_{n1,n4}^{1,8} + t_{n1,n5}\delta_{n1,n5}^{1,8} + t_{n4,n6}\delta_{n4,n6}^{1,8} + t_{n5,n6}\delta_{n5,n6}^{1,8} \\ & + t_{n4,n7}\delta_{n4,n7}^{1,8} + t_{n5,n7}\delta_{n5,n7}^{1,8} + t_{n6,n8}\delta_{n6,n8}^{1,8} + t_{n7,n8}\delta_{n7,n8}^{1,8}) \end{aligned}$$

subject to:

$$\delta_{n1,n4}^{1,8} + \delta_{n1,n5}^{1,8} = 1$$

$$\delta_{n1,n4}^{1,8} = \delta_{n4,n6}^{1,8} + \delta_{n4,n7}^{1,8}$$

$$\delta_{n1,n5}^{1,8} = \delta_{n5,n6}^{1,8} + \delta_{n5,n7}^{1,8}$$

$$\delta_{n6,n8}^{1,8} = \delta_{n4,n6}^{1,8} + \delta_{n5,n6}^{1,8}$$

$$\delta_{n7,n8}^{1,8} = \delta_{n4,n7}^{1,8} + \delta_{n5,n7}^{1,8}$$

$$\delta_{n6,n8}^{1,8} + \delta_{n7,n8}^{1,8} = 1$$

$$\begin{aligned} \min_{\delta^{1,9}} f_{n1,n9} = & (t_{n1,n4}\delta_{n1,n4}^{1,9} + t_{n1,n5}\delta_{n1,n5}^{1,9} + t_{n4,n6}\delta_{n4,n6}^{1,9} + t_{n5,n6}\delta_{n5,n6}^{1,9} \\ & + t_{n4,n7}\delta_{n4,n7}^{1,9} + t_{n5,n7}\delta_{n5,n7}^{1,9} + t_{n6,n9}\delta_{n6,n9}^{1,9} + t_{n7,n9}\delta_{n7,n9}^{1,9}) \end{aligned}$$

subject to:

$$\delta_{n1,n4}^{1,9} + \delta_{n1,n5}^{1,9} = 1$$

$$\delta_{n1,n4}^{1,9} = \delta_{n4,n6}^{1,9} + \delta_{n4,n7}^{1,9}$$

$$\delta_{n1,n5}^{1,9} = \delta_{n5,n6}^{1,9} + \delta_{n5,n7}^{1,9}$$

$$\delta_{n6,n9}^{1,9} = \delta_{n4,n6}^{1,9} + \delta_{n5,n6}^{1,9}$$

$$\delta_{n7,n9}^{1,9} = \delta_{n4,n7}^{1,9} + \delta_{n5,n7}^{1,9}$$

$$\delta_{n6,n9}^{1,9} + \delta_{n7,n9}^{1,9} = 1$$

$$\begin{aligned} \min_{\delta_{i,j}^{2,8}} f_{n2,n8} = & (t_{n2,n4}\delta_{n2,n4}^{2,8} + t_{n2,n5}\delta_{n2,n5}^{2,8} + t_{n4,n6}\delta_{n4,n6}^{2,8} + t_{n5,n6}\delta_{n5,n6}^{2,8} \\ & + t_{n4,n7}\delta_{n4,n7}^{2,8} + t_{n5,n7}\delta_{n5,n7}^{2,8} + t_{n6,n8}\delta_{n6,n8}^{2,8} + t_{n7,n8}\delta_{n7,n8}^{2,8}) \end{aligned}$$

subject to:

$$\delta_{n2,n4}^{2,8} + \delta_{n2,n5}^{2,8} = 1$$

$$\delta_{n2,n4}^{2,8} = \delta_{n4,n6}^{2,8} + \delta_{n4,n7}^{2,8}$$

$$\delta_{n2,n5}^{2,8} = \delta_{n5,n6}^{2,8} + \delta_{n5,n7}^{2,8}$$

$$\delta_{n6,n8}^{2,8} = \delta_{n4,n6}^{2,8} + \delta_{n5,n6}^{2,8}$$

$$\delta_{n7,n8}^{2,8} = \delta_{n4,n7}^{2,8} + \delta_{n5,n7}^{2,8}$$

$$\delta_{n6,n8}^{2,8} + \delta_{n7,n8}^{2,8} = 1$$

$$\begin{aligned} \min_{\delta_{i,j}^{2,9}} f_{n2,n9} = & (t_{n2,n4}\delta_{n2,n4}^{2,9} + t_{n2,n5}\delta_{n2,n5}^{2,9} + t_{n4,n6}\delta_{n4,n6}^{2,9} + t_{n5,n6}\delta_{n5,n6}^{2,9} \\ & + t_{n4,n7}\delta_{n4,n7}^{2,9} + t_{n5,n7}\delta_{n5,n7}^{2,9} + t_{n6,n9}\delta_{n6,n9}^{2,9} + t_{n7,n9}\delta_{n7,n9}^{2,9}) \end{aligned}$$

subject to:

$$\delta_{n2,n4}^{2,9} + \delta_{n2,n5}^{2,9} = 1$$

$$\delta_{n2,n4}^{2,9} = \delta_{n4,n6}^{2,9} + \delta_{n4,n7}^{2,9}$$

$$\delta_{n2,n5}^{2,9} = \delta_{n5,n6}^{2,9} + \delta_{n5,n7}^{2,9}$$

$$\delta_{n6,n9}^{2,9} = \delta_{n4,n6}^{2,9} + \delta_{n5,n6}^{2,9}$$

$$\delta_{n7,n9}^{2,9} = \delta_{n4,n7}^{2,9} + \delta_{n5,n7}^{2,9}$$

$$\delta_{n6,n9}^{2,9} + \delta_{n7,n9}^{2,9} = 1$$

$$\begin{aligned} \min_{\delta_{i,j}^{3,8}} f_{n3,n8} = & (t_{n3,n4}\delta_{n3,n4}^{3,8} + t_{n3,n5}\delta_{n3,n5}^{3,8} + t_{n4,n6}\delta_{n4,n6}^{3,8} + t_{n5,n6}\delta_{n5,n6}^{3,8} \\ & + t_{n4,n7}\delta_{n4,n7}^{3,8} + t_{n5,n7}\delta_{n5,n7}^{3,8} + t_{n6,n8}\delta_{n6,n8}^{3,8} + t_{n7,n8}\delta_{n7,n8}^{3,8}) \end{aligned}$$

subject to:

$$\delta_{n3,n4}^{3,8} + \delta_{n3,n5}^{3,8} = 1$$

$$\delta_{n3,n4}^{3,8} = \delta_{n4,n6}^{3,8} + \delta_{n4,n7}^{3,8}$$

$$\delta_{n3,n5}^{3,8} = \delta_{n5,n6}^{3,8} + \delta_{n5,n7}^{3,8}$$

$$\delta_{n6,n8}^{3,8} = \delta_{n4,n6}^{3,8} + \delta_{n5,n6}^{3,8}$$

$$\delta_{n7,n8}^{3,8} = \delta_{n4,n7}^{3,8} + \delta_{n5,n7}^{3,8}$$

$$\delta_{n6,n8}^{3,8} + \delta_{n7,n8}^{3,8} = 1$$

$$\begin{aligned} \min_{\delta_{i,j}^{3,9}} f_{n3,n9} = & (t_{n3,n4}\delta_{n3,n4}^{3,9} + t_{n3,n5}\delta_{n3,n5}^{3,9} + t_{n4,n6}\delta_{n4,n6}^{3,9} + t_{n5,n6}\delta_{n5,n6}^{3,9} \\ & + t_{n4,n7}\delta_{n4,n7}^{3,9} + t_{n5,n7}\delta_{n5,n7}^{3,9} + t_{n6,n9}\delta_{n6,n9}^{3,9} + t_{n7,n9}\delta_{n7,n9}^{3,9}) \end{aligned}$$

subject to:

$$\delta_{n3,n4}^{3,9} + \delta_{n3,n5}^{3,9} = 1$$

$$\delta_{n3,n4}^{3,9} = \delta_{n4,n6}^{3,9} + \delta_{n4,n7}^{3,9}$$

$$\delta_{n3,n5}^{3,9} = \delta_{n5,n6}^{3,9} + \delta_{n5,n7}^{3,9}$$

$$\delta_{n6,n9}^{3,9} = \delta_{n4,n6}^{3,9} + \delta_{n5,n6}^{3,9}$$

$$\delta_{n7,n9}^{3,9} = \delta_{n4,n7}^{3,9} + \delta_{n5,n7}^{3,9}$$

$$\delta_{n6,n9}^{3,9} + \delta_{n7,n9}^{3,9} = 1$$

We have made the assumption that the drivers at the lower level of the decision process are at the same level and therefore they optimise their travel times independently and the choices they make on which routes to use do not depend on the other drivers' decisions. Therefore we compute the sum of the 6 objective functions in the lower level problem and we convert the lower level problem from a multiple follower problem to a single follower problem with only one objective function f under the lower level problem[90].

$$\min_{\delta_{ni,nj}^{o,d}} f = f_{n1,n8} + f_{n1,n9} + f_{n2,n8} + f_{n2,n9} + f_{n3,n8} + f_{n3,n9}$$

subject to

$$\delta_{n1,n4}^{1,8} + \delta_{n1,n5}^{1,8} = 1$$

$$\delta_{n1,n4}^{1,8} = \delta_{n4,n6}^{1,8} + \delta_{n4,n7}^{1,8}$$

$$\delta_{n1,n5}^{1,8} = \delta_{n5,n6}^{1,8} + \delta_{n5,n7}^{1,8}$$

$$\delta_{n6,n8}^{1,8} = \delta_{n4,n6}^{1,8} + \delta_{n5,n6}^{1,8}$$

$$\delta_{n7,n8}^{1,8} = \delta_{n4,n7}^{1,8} + \delta_{n5,n7}^{1,8}$$

$$\delta_{n6,n8}^{1,8} + \delta_{n7,n8}^{1,8} = 1$$

$$\delta_{n1,n4}^{1,9} + \delta_{n1,n5}^{1,9} = 1$$

$$\delta_{n1,n4}^{1,9} = \delta_{n4,n6}^{1,9} + \delta_{n4,n7}^{1,9}$$

$$\delta_{n1,n5}^{1,9} = \delta_{n5,n6}^{1,9} + \delta_{n5,n7}^{1,9}$$

$$\delta_{n6,n9}^{1,9} = \delta_{n4,n6}^{1,9} + \delta_{n5,n6}^{1,9}$$

$$\delta_{n4,n2}^{1,9} = \delta_{n4,n7}^{1,9} + \delta_{n5,n7}^{1,9}$$

$$\delta_{n6,n9}^{1,9} + \delta_{n7,n9}^{1,9} = 1$$

$$\delta_{n2,n4}^{2,8} + \delta_{n2,n5}^{2,8} = 1$$

$$\delta_{n2,n4}^{2,8} = \delta_{n4,n6}^{2,8} + \delta_{n4,n7}^{2,8}$$

$$\delta_{n2,n5}^{2,8} = \delta_{n5,n6}^{2,8} + \delta_{n5,n7}^{2,8}$$

$$\delta_{n6,n8}^{2,8} = \delta_{n4,n6}^{2,8} + \delta_{n5,n6}^{2,8}$$

$$\delta_{n7,n8}^{2,8} = \delta_{n4,n7}^{2,8} + \delta_{n5,n7}^{2,8}$$

$$\delta_{n6,n8}^{2,8} + \delta_{n7,n8}^{2,8} = 1$$

$$\delta_{n2,n4}^{2,9} + \delta_{n2,n5}^{2,9} = 1$$

$$\delta_{n2,n4}^{2,9} = \delta_{n4,n6}^{2,9} + \delta_{n4,n7}^{2,9}$$

$$\delta_{n2,n5}^{2,9} = \delta_{n5,n6}^{2,9} + \delta_{n5,n7}^{2,9}$$

$$\delta_{n6,n9}^{2,9} = \delta_{n4,n6}^{2,9} + \delta_{n5,n6}^{2,9}$$

$$\delta_{n7,n9}^{2,9} = \delta_{n4,n7}^{2,9} + \delta_{n5,n7}^{2,9}$$

$$\delta_{n6,n9}^{2,9} + \delta_{n7,n9}^{2,9} = 1$$

$$\delta_{n3,n4}^{3,8} + \delta_{n3,n5}^{3,8} = 1$$

$$\delta_{n3,n4}^{3,8} = \delta_{n4,n6}^{3,8} + \delta_{n4,n7}^{3,8}$$

$$\delta_{n3,n5}^{3,8} = \delta_{n5,n6}^{3,8} + \delta_{n5,n7}^{3,8}$$

$$\delta_{n6,n8}^{3,8} = \delta_{n4,n6}^{3,8} + \delta_{n5,n6}^{3,8}$$

$$\delta_{n7,n8}^{3,8} = \delta_{n4,n7}^{3,8} + \delta_{n5,n7}^{3,8}$$

$$\delta_{n6,n8}^{3,8} + \delta_{n7,n8}^{3,8} = 1$$

$$\delta_{n3,n4}^{3,9} + \delta_{n3,n5}^{3,9} = 1$$

$$\delta_{n3,n4}^{3,9} = \delta_{n4,n6}^{3,9} + \delta_{n4,n7}^{3,9}$$

$$\delta_{n3,n5}^{3,9} = \delta_{n5,n6}^{3,9} + \delta_{n5,n7}^{3,9}$$

$$\delta_{n6,n9}^{3,9} = \delta_{n4,n6}^{3,9} + \delta_{n5,n6}^{3,9}$$

$$\delta_{n7,n9}^{3,9} = \delta_{n4,n7}^{3,9} + \delta_{n5,n7}^{3,9}$$

$$\delta_{n6,n9}^{3,9} + \delta_{n7,n9}^{3,9} = 1$$

5.3.3 The upper level problem (Traffic Authority's problem)

The objective of the traffic authority is to minimise the total travel time of the whole network. This can be expressed as the sum of the product of the travel time on each link and the total flow on the corresponding link. The traffic authority cannot choose which routes the driver can use. However the traffic authority has control over the speed limit which they can set on the different roadways and the cycle length which they can set on the different intersections. This can have some influence on the driver concerning which routes to take. In the upper level problem, the driver's problem is treated like a constraint, therefore, we have another optimisation problem as a constraint in upper level. The same cost function t_{ij} for the upper level is maintained.

Table 5.3 shows the variables and parameters used in the upper level problem.

Variables	Parameters
u_{ij} - the free flow speed	ca_{ij} - the capacity on road ij l_{ij} - section length (km) from node i to node j
C_i - cycle length	

Table 5.3: Parameters and variables - upper level level.

Suppose the minimum free flow speed is 20 km/hr and the maximum speed is 80 km/hr. Similarly suppose the minimum cycle length is $\frac{1}{30}$ hrs and the maximum cycle length is $\frac{1}{15}$ hrs.

Then, it follows that the traffic authority minimises the following function:

$$\min_{C_i, u_{ij}} F = (f_{n1,n8} \cdot x_{n1,n8} + f_{n1,n9} \cdot x_{n1,n9} + f_{n2,n8} \cdot x_{n2,n8} + f_{n2,n9} \cdot x_{n2,n9} + f_{n3,n8} \cdot x_{n3,n8} + f_{n3,n9} \cdot x_{n3,n9}) \quad (5.11)$$

subject to :

$$20 \text{ km/h} \leq u_{ij} \leq 80 \text{ km/h} \quad (5.12)$$

$$\frac{1}{30} \text{ hours} \leq C_i \leq \frac{1}{15} \text{ hours} \quad (5.13)$$

$$x_{1,4} + x_{1,5} = S_1 \quad (5.14)$$

$$x_{2,4} + x_{2,5} = S_2 \quad (5.15)$$

$$x_{3,4} + x_{3,5} = S_3 \quad (5.16)$$

$$x_{6,8} + x_{7,8} = D_1 \quad (5.17)$$

$$x_{6,9} + x_{7,9} = D_2 \quad (5.18)$$

$$S_1 + S_2 + S_3 = D_1 + D_2, \quad (5.19)$$

where the flow is defined as follows:

$$\delta_{n4,n6}^{1,8} r_{n1,n8} + \delta_{n4,n6}^{1,9} r_{n1,n9} + \delta_{n4,n6}^{2,8} r_{n2,n8} + \delta_{n4,n6}^{2,9} r_{n2,n9} + \delta_{n4,n6}^{3,8} r_{n3,n8} + \delta_{n4,n6}^{3,9} r_{n3,n9} = x_{n4,n6}$$

$$\delta_{n4,n7}^{1,8} r_{n1,n8} + \delta_{n4,n7}^{1,9} r_{n1,n9} + \delta_{n4,n7}^{2,8} r_{n2,n8} + \delta_{n4,n7}^{2,9} r_{n2,n9} + \delta_{n4,n7}^{3,8} r_{n3,n8} + \delta_{n4,n7}^{3,9} r_{n3,n9} = x_{n4,n7}$$

$$\delta_{n5,n6}^{1,8} r_{n1,n8} + \delta_{n5,n6}^{1,9} r_{n1,n9} + \delta_{n5,n6}^{2,8} r_{n2,n8} + \delta_{n5,n6}^{2,9} r_{n2,n9} + \delta_{n5,n6}^{3,8} r_{n3,n8} + \delta_{n5,n6}^{3,9} r_{n3,n9} = x_{n5,n6}$$

$$\delta_{n5,n7}^{1,8} r_{n1,n8} + \delta_{n5,n7}^{1,9} r_{n1,n9} + \delta_{n5,n7}^{2,8} r_{n2,n8} + \delta_{n5,n7}^{2,9} r_{n2,n9} + \delta_{n5,n7}^{3,8} r_{n3,n8} + \delta_{n5,n7}^{3,9} r_{n3,n9} = x_{n5,n7}$$

5.4 Numerical Results

In this study the simulations of the bilevel problem is implemented using the MATLAB programming language (see Appendix), and hypothetical data was used for the traffic's network parameters. The MATLAB code was run using R2022a version of MATLAB on an iMac with a 1.4 GHz Dual-Core Intel Core i5 processor and an 8GB 1600MHz DDR3 memory.

Numerical simulations were carried out to find the upper and lower level best cost and to also compute the optimal solution for the upper level and lower level, respectively. The lower level is estimated using the BPSO which uses a probability (sigmoid) function to obtain the binary values. The main code was run thirty times to obtain the possible lowest value since the BPSO uses a probability function that generates random numbers. Due to the probability function in the BPSO, a different value appears each time the code is run. Therefore it was run as many times as possible to increase the chance of getting a better solution than the previous one.

The formulated code runs for 50 iterations and a partitioning parameter value of $\beta = 0.02$ was used. In the BPSO the cognitive and social parameters were both assigned a value of 2, an inertia weight of a value of 5 and a velocity limiting parameter of 0.75. The population size of the BPSO was taken as Npop=60 with a maximum of 60 iterations.

In this study the upper level problem consists of many variables which are to be partitioned, and due to this reason, it takes several hours for the code to converge to a solution. The main partitioning code systematically selects feasible solutions of the leader from each hyper-box using **Algorithm2** then sends it to the lower level to get the best response, and the BPSO will compute the lower level best response using **Algorithm3**. These responses are then used in the leader's objective to get a solution. The next step is to compare the solutions and the best solution is selected and stored. This process is repeated at every iteration for 50 times, each time taking a different set of sample points from the leader's decision space.

5.5 Case 1: 100 vehicles from the source nodes

In the particular example given in Section 5.3 for case 1, we have a total of 100 vehicles departing from the source nodes, with 25 vehicles departing from n_1 , 35 from n_2 and 40 vehicles from n_3 . The 100 vehicles are destined for two nodes- n_8 and n_9 .

The results in Table 5.4 show the minimum time the driver will take when traveling from his origin node to his respective destination node, where the three origin nodes- n_1, n_2 and n_3 are the sources and the destination nodes- n_8 and n_9 are the sinks.

Table 5.4 indicates that the travel time is approximately 0.2943 hours for a vehicle whose origin is n_1 and is destined to n_8 , the same can be deduced for f_{n_1, n_9} and the rest of the lower level objectives. The results in Table 5.4 show that the route/path with the shortest travel time is f_{n_2, n_8} with a travel time of 0.0360 hours or 2.16 minutes while the route with the longest travel time is f_{n_1, n_8} with a travel time of 0.2943 hours or 17.658 minutes.

Objective Function	Travel time (hours)
f_{n_1,n_8}	0.2943
f_{n_1,n_9}	0.1576
f_{n_2,n_8}	0.0360
f_{n_2,n_9}	0.0628
f_{n_3,n_8}	0.2051
f_{n_3,n_9}	0.1067
f	0.8624

Table 5.4: Lower level objective function (case1)

In Table 5.5 the upper level best cost is indicated as 16.3967 hours at the end of the 50 iterations. This means that it would take the traffic authority 16.3967 hours to clear all the traffic from the network. This outcome is due to the small capacities which have been assigned to the links in the given example in addition the cars are in a “bumper to bumper” motion. This result indicates that increasing the capacity in the links would lead to a shorter time taken for the traffic authority to clear off all traffic in the network. The lower level best cost as shown in Table 5.5 is 0.86262 hours. It is a summation of the six lower level objectives f_{n_1,n_8} to f_{n_3,n_9} but the interest is in the individual cost which are indicated in Table 5.4.

Table 5.5 shows the results of the MatLab code for the best optimal solution.

Iteration 1: Upper Best Cost = 24.0299: Lower Best Cost = 1.0278
Iteration 2: Upper Best Cost = 24.0299: Lower Best Cost = 1.0278
Iteration 3: Upper Best Cost = 24.0299: Lower Best Cost = 1.0278
Iteration 4: Upper Best Cost = 24.0299: Lower Best Cost = 1.0278
Iteration 5: Upper Best Cost = 24.0299: Lower Best Cost = 1.0278
⋮ ⋮ ⋮
Iteration 21: Upper Best Cost = 20.1355: Lower Best Cost = 0.95927
Iteration 22: Upper Best Cost = 20.1355: Lower Best Cost = 0.95927
Iteration 23: Upper Best Cost = 16.3967: Lower Best Cost = 0.86262
Iteration 24: Upper Best Cost = 16.3967: Lower Best Cost = 0.86262
Iteration 25: Upper Best Cost = 16.3967: Lower Best Cost = 0.86262
⋮ ⋮ ⋮
Iteration 46: Upper Best Cost = 16.3967: Lower Best Cost = 0.86262
Iteration 47: Upper Best Cost = 16.3967: Lower Best Cost = 0.86262
Iteration 48: Upper Best Cost = 16.3967: Lower Best Cost = 0.86262
Iteration 49: Upper Best Cost = 16.3967: Lower Best Cost = 0.86262
Iteration 50: Upper Best Cost = 16.3967: Lower Best Cost = 0.86262

Table 5.5: The upper and lower level best cost (case 1)

The optimal maximum speed from Table 5.6 and the optimal cycle length from Table 5.7 were substituted in the lower level objective function to obtain the travel time for each respective

path in the given network.

Table 5.6 and 5.7 is a summary of the best optimal solution for the traffic authority.

Link	Speed
u_{14}	52.192km/hr
u_{15}	52.12km/hr
u_{24}	52.349km/hr
u_{25}	52.167km/hr
u_{34}	52.189km/hr
u_{35}	52.3km/hr
u_{46}	52.081km/hr
u_{47}	52.392km/hr
u_{56}	52.2km/hr
u_{57}	52.068km/hr
u_{68}	52.377km/hr
u_{69}	52.175km/hr
u_{78}	52.363km/hr
u_{79}	52.354km/hr

Table 5.6: The upper level optimal solution (case 1)

Table 5.6 and Table 5.7 indicate the optimal solution of traffic authority with its respective best costs on Table 5.5. From Table 5.6, $u_{14}= 52.192$ km/hr is the maximum speed that the traffic authority should set on the link (n_1, n_4) while Table 5.7 indicates that $C_4= 235$ seconds is the cycle length which should be set on the traffic light at node 4 in order to achieve the given best costs. The remaining values can be interpreted in the same manner. The intermediate nodes n_4, n_5, n_6 and n_7 are the only nodes with signalised intersections.

Traffic light	Cycle length (seconds)
C_4	235
C_5	234
C_6	235
C_7	235

Table 5.7: The upper level optimal solution (case 1)

The binary values δ_1^* shows the lower level optimal solution for case 1.

0 1 1 0 1 1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1

Lower level Optimal solution- δ_1^* (case 1)

5.6 Case 2: The number of vehicles from the source nodes with different proportions

In this case the number of vehicles departing from the origin nodes is presented as follows: $n_1 = 10, n_2 = 50, n_3 = 40$. The results indicate the upper level best cost to be equal to 17.7385 hrs while the lower best cost was found to be 1.1138 hrs, with both costs slightly higher than the costs for case 1, which was expected since the number of vehicles departing from node 1 was extremely small compared with the other two source nodes. The values for the travel times for case 1 and case 2 are compared using Tables 5.8 and 5.9, respectively. And we deduce that the travel time for objective f_{n_2, n_8} is three times more for case 2 than for case 1. This indicates a possible traffic jam on this route due to the small number of vehicles assigned at node 1 for case 2.

Obj Funtn	Travel-time(hrs)
f_{n_1, n_8}	0.3027
f_{n_1, n_9}	0.1610
f_{n_2, n_8}	0.0391
f_{n_2, n_9}	0.0657
f_{n_3, n_8}	0.2119
f_{n_3, n_9}	0.1088
f	0.8892

Table 5.8: Case 1

Obj Funtn	Travel-time(hrs)
f_{n_1, n_8}	0.1997
f_{n_1, n_9}	0.1540
f_{n_2, n_8}	0.1018
f_{n_2, n_9}	0.0369
f_{n_3, n_8}	0.3580
f_{n_3, n_9}	0.2634
f	1.1138

Table 5.9: Case 2

Table 5.10 shows the results for the upper level optimal solution for case 2.

Link	Speed
u_{14}	48.781km/hr
u_{15}	48.589km/hr
u_{24}	48.46km/hr
u_{25}	48.779km/hr
u_{34}	48.695km/hr
u_{35}	48.411km/hr
u_{46}	48.554km/hr
u_{47}	48.702 km/hr
u_{56}	48.47km/hr
u_{57}	48.565km/hr
u_{68}	48.5km/hr
u_{69}	48.753km/hr
u_{78}	48.423 km/hr
u_{79}	48.752km/hr

Table 5.10: The upper level optimal solution (case 2)

The binary values δ_2^* shows the optimal solution for the lower level problem. These binary values are used to obtain the values of the driver's objective functions.

Traffic light	Cycle length (seconds)
C_4	221
C_5	222
C_6	222
C_7	222

Table 5.11: The upper level optimal solution (case 2)

0 1 1 0 0 1 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1

Lower level Optimal solution- δ_2^* (case 2)

5.7 Case 3: 500 vehicles from the source nodes

In this case, the number of vehicles departing from the origin nodes is presented as follows: $n_1 = 190, n_2 = 150, n_3 = 160$. The results indicate the upper best cost to be equal to 80.8643 hrs while the lower best cost was found to be 0.93022 hrs. This indicates that the traffic authority will take a long time trying to clear off the traffic. Such a traffic load on the traffic network will create a big traffic jam and therefore, the authority should be ready for such an incident. This means, unless an alternative route is created the network is not sufficient for such load.

Tables 5.12 and 5.13 show the results for the upper level optimal solution for case 3 for the free flow speed in each link and the cycle length, respectively.

Link	Speed
u_{14}	52.679km/hr
u_{15}	52.505km/hr
u_{24}	52.571/hr
u_{25}	52.538km/hr
u_{34}	52.704 km/hr
u_{35}	52.46km/hr
u_{46}	52.625 km/hr
u_{47}	52.514 km/hr
u_{56}	52.59km/hr
u_{57}	52.439km/hr
u_{68}	52.478km/hr
u_{69}	52.72km/hr
u_{78}	52.744km/hr
u_{79}	52.584km/hr

Table 5.12: The upper level optimal solution (case 3)

The binary values δ_3^* shows the optimal solution for the lower level problem. These binary values are used to obtain the values of the driver's objective functions.

In this study, we propose a flow management transportation model which can be used to calculate the travel time of a path. This model can also be used to calculate the value of the maximum velocities on each link and the value of the cycle length on each traffic light which will minimise

Traffic light	Cycle length (seconds)
C_4	236
C_5	237
C_6	236
C_7	236

Table 5.13: The upper level optimal solution (case 3)

1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0

Lower level Optimal solution- δ_3^* (case 3)

the travel time on a path. This model was formulated for the traffic authority and we propose this traffic management plan to the traffic authority for use in managing the flow of a section of network or the network of a single town. The traffic authority can use the formulated model to set the maximum permitted speed on the roadway and the cycle length of traffic lights to an appropriate value which can help to reduce traffic jams, improve the flow and hence reduce the amount of time spent by drivers traveling.

The bilevel method compared to the classical single optimisation method is more accurate since it incorporates more constraints, parameters and objective functions in addition leading to improved traffic behavior. When applied in a proper manner this model can be useful to paramedics, ambulance drivers, police officers, fire fighters and logistics companies and other emergency vehicles on the best routes to use since the travel time for the different routes will be known.

6. Conclusion and Recommendations

In this study a bilevel optimisation model was proposed for managing flow in an urban network. In the upper level the decision space of the leader was partitioned and the leader's actions were chosen using a systematic sampling technique. On the other hand, for the lower level problem a global optimisation technique called binary particle swarm optimisation was proposed for solving the lower level problem. The optimal response from the driver was obtained using the BPSO. This process was done until the maximum number of iterations was reached.

A meta-heuristic algorithm was adopted because it is derivative free and easier to implement than other optimisation techniques. The problem we dealt with is a two level deterministic problem with a single leader and multiple followers. This problem was later converted to a single leader and single follower problem. A signal controlled net work only was considered. It is important to note that in this problem the driver is independent and freely chooses the routes that they want to use. The routes that the driver wishes to use cannot be decided by the authority.

By solving the optimisation procedure we have described above, we obtained a solution of the link which makes the flow to be optimal. This value gives the optimum route for the drivers to follow and the actual flow on each link. This information is of importance to the authority since they now know which links to manage and pay close attention to. The higher the flow, the higher the possibility of a traffic jam.

If one wants to expand the network, and work on a bigger scale problem, it means that this complex problem will have a large number of optimisation variables, both for the lower and upper level problems. There will also be a large number of constraints on both levels to consider. Expanding the model means that one would need a larger optimisation model and MATLAB code, because they will now be working with a larger number of variables.

The limitations of the model include the assumptions that were made while creating the model. This includes the assumption that all the green intervals and cycle length at each traffic signal have the same length interval, which is not usually the case. One other limitation is that only a signal controlled network was examined as opposed to a network made up of different types of intersections such as roundabouts and stop-signs.

This model can be very useful to the traffic authorities as it can be applied in urban areas which have signal controlled intersections to manage and control traffic flow.

Future Work

Some ideas for extending the thesis work are elaborated below.

In future there is a need to engage in a study that looks at a larger urban network scale. This would lead to more parameters both at the lower and upper level. A possible extension of this research also concerns the inclusion of the green interval and offsets in the bilevel and optimisation model as one of the decision variables in upper level. Other types of intersections such as roundabouts and stop signs, can also be considered. This approach, that is the inclusion of the offsets and green interval will result in an increased number of decision variables for the leader, hence an appropriate MATLAB code and algorithms will be needed to solve the problem.



A. MatLab Codes

A.1 The main partitioning MatLab code

```
1 %% Systematic Evolutionary Algorithm %%
2 clc; clear ; clear var;
3
4 %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 %% Problem Definition %%
6 %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8 % ObjFunU = @(x) ObjectiveFunctionU; % First level Decision maker
   objective function
9 % ConsFunU = @(x) myconstraintU;      % First level Decision maker
   constraint function
10 % ObjFunL = @(x) ObjectiveFunctionL; % Second level Decision maker
   objective function
11 % ConsFunL = @(x) myconstraintL;      % Second level Decision maker
   constraint function
12
13
14 MinVar(1) = 0.02;                      % Minimum value of leader
   variable u14
15 MaxVar(1) = 0.08;                      % Maximum value of leader
   variable u14
16 MinVar(2) = 0.02;                      % Minimum value of leader
   variable u15
17 MaxVar(2) = 0.08;                      % Maximum value of leader
   variable u15
18 MinVar(3) = 0.02;                      % Minimum value of leader
   variable u24
19 MaxVar(3) = 0.08;                      % Maximum value of leader
   variable u24
20 MinVar(4) = 0.02;                      % Minimum value of leader
   variable u25
21 MaxVar(4) = 0.08;                      % Maximum value of leader
```

```

    variableMin u25
22 MinVar(5) = 0.02; % Minimum value of leader
    variable u34
23 MaxVar(5) = 0.08; % Maximum value of leader
    variable u34
24 MinVar(6) = 0.02; % Minimum value of leader
    variable u35
25 MaxVar(6) = 0.08; % Maximum value of leader
    variable u35
26 MinVar(7) = 0.02; % Minimum value of leader
    variable u46
27 MaxVar(7) = 0.08; % Maximum value of leader
    variable u46
28 MinVar(8) = 0.02; % Minimum value of leader
    variable u47
29 MaxVar(8) = 0.08; % Maximum value of leader
    variable u47
30 MinVar(9) = 0.02; % Minimum value of leader
    variable u56
31 MaxVar(9) = 0.08; % Maximum value of leader
    variable u56
32 MinVar(10) = 0.02; % Minimum value of leader
    variable u57
33 MaxVar(10) = 0.08; % Maximum value of leader
    variable u57
34 MinVar(11) = 0.02; % Minimum value of leader
    variable u68
35 MaxVar(11) = 0.08; % Maximum value of leader
    variable u68
36 MinVar(12) = 0.02; % Minimum value of leader
    variable u69
37 MaxVar(12) = 0.08; % Maximum value of leader
    variable u69
38 MinVar(13) = 0.02; % Minimum value of leader
    variable u78
39 MaxVar(13) = 0.08; % Maximum value of leader
    variable u78
40 MinVar(14) = 0.02; % Minimum value of leader
    variable u79
41 MaxVar(14) = 0.08; % Maximum value of leader
    variable u79
42 MinVar(15) = 0.033; % Minimum value of leader

```

```

    variable C4
43 MaxVar(15) = 0.067; % Maximum value of leader
    variable C4
44 MinVar(16) = 0.033; % Minimum value of leader
    variable C5
45 MaxVar(16) = 0.067; % Maximum value of leader
    variable C5
46 MinVar(17) = 0.033; % Minimum value of leader
    variable C6
47 MaxVar(17) = 0.067; % Maximum value of leader
    variable C6
48 MinVar(18) = 0.033; % Minimum value of leader
    variable C7
49 MaxVar(18) = 0.067; % Maximum value of leader
    variable C7
50
51 noVar_U = 18; % Number of upper level
    variables , note y(3) = BT - (y(1)+y(2)).
52 VarSize_U = [1 noVar_U]; % Decision space
53 Range = MaxVar - MinVar; % Range of decision space(Length
    of upper problem decision space(?))
54
55 nvar_L = 48; % Number of variables in the
    lower level problem
56 Npop = 60; % Number of population for the
    PSO
57 Max_iter_PSO = 60; % Maximum iteration of the PSO
58
59
60 MaxIter = 50; % Maximum number of iteration for
    the partition section
61
62
63 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
64 %% Parameter Definition %%
65 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
66
67 %% Sampling Parameter for partitioning decision space at each

```

```

    Iteration
68  epsilon = 0.00;           % Tolerance of a reaction
69  DELTA = 0.015;           % delta
70  a(1) = 2*ceil(Range(1)/(20*DELTA)); % Number of spies on first level
71  a(2) = 2*ceil(Range(2)/(20*DELTA)); % Number of spies on first level
72  a(3) = 2*ceil(Range(3)/(20*DELTA)); % Number of spies on first level
73  a(4) = 2*ceil(Range(4)/(20*DELTA)); % Number of spies on first level
74  a(5) = 2*ceil(Range(5)/(20*DELTA)); % Number of spies on first level
75  a(6) = 2*ceil(Range(6)/(20*DELTA)); % Number of spies on first level
76  a(7) = 2*ceil(Range(7)/(20*DELTA)); % Number of spies on first level
77  a(8) = 2*ceil(Range(8)/(20*DELTA)); % Number of spies on first level
78  a(9) = 2*ceil(Range(9)/(20*DELTA)); % Number of spies on first level
79  a(10) = 2*ceil(Range(10)/(20*DELTA)); % Number of spies on first
    level
80  a(11) = 2*ceil(Range(11)/(20*DELTA)); % Number of spies on first
    level
81  a(12) = 2*ceil(Range(12)/(20*DELTA)); % Number of spies on first
    level
82  a(13) = 2*ceil(Range(13)/(20*DELTA)); % Number of spies on first
    level
83  a(14) = 2*ceil(Range(14)/(20*DELTA)); % Number of spies on first
    level
84  a(15) = 2*ceil(Range(15)/(20*DELTA)); % Number of spies on first
    level
85  a(16) = 2*ceil(Range(16)/(20*DELTA)); % Number of spies on first
    level
86  a(17) = 2*ceil(Range(17)/(20*DELTA)); % Number of spies on first
    level
87  a(18) = 2*ceil(Range(18)/(20*DELTA)); % Number of spies on first
    level
88
89
90
91  b(1) = floor(a(1)/2); % Number of subintervals on first
    level
92  b(2) = floor(a(2)/2); % Number of subintervals on first
    level
93  b(3) = floor(a(3)/2); % Number of subintervals on first
    level
94  b(4) = floor(a(4)/2); % Number of subintervals on first
    level
95  b(5) = floor(a(5)/2); % Number of subintervals on first

```

```

    level
96  b(6) = floor(a(6)/2);           % Number of subintervals on first
    level
97  b(7) = floor(a(7)/2);           % Number of subintervals on first
    level
98  b(8) = floor(a(8)/2);           % Number of subintervals on first
    level
99  b(9) = floor(a(9)/2);           % Number of subintervals on first
    level
100 b(10) = floor(a(10)/2);          % Number of subintervals on
    first level
101 b(11) = floor(a(11)/2);          % Number of subintervals on
    first level
102 b(12) = floor(a(12)/2);          % Number of subintervals on
    first level
103 b(13) = floor(a(13)/2);          % Number of subintervals on
    first level
104 b(14) = floor(a(14)/2);          % Number of subintervals on
    first level
105 b(15) = floor(a(15)/2);          % Number of subintervals on
    first level
106 b(16) = floor(a(16)/2);          % Number of subintervals on
    first level
107 b(17) = floor(a(17)/2);          % Number of subintervals on
    first level
108 b(18) = floor(a(18)/2);          % Number of subintervals on
    first level
109
110
111
112 Delta = DELTA/MaxIter;           % Step size of the partition of
    the first subinterval
113 perm = randperm(MaxIter);        % Random representative of the
    first subintervals
114 empty_Spies= [];
115 lambda = 1000000000000;           % Penalty Factor
116
117
118
119
120
121 %

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
122 %% Main Iteration  %%
123 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

124 for IT=1:MaxIter
125
126 %% Initialization
127 % Distribution of partition by first decision variables on first
    level
128 % Distribution of partition by first decision variables on first
    level

129
130 flsf(1) = MinVar(1)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition

131 if flsf(1) < 0
132     flsf(1) = 0;
133 end
134 flsf(2) = MinVar(1)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
135 if flsf(2) < 0
136     flsf(2) = 0;
137 end
138 for q=3:2:2*b(1)-1
139     flsf(q) = flsf(q-2)+2*DELTA;
140     if flsf(q) > MaxVar(1)
141         flsf(q) = MaxVar(1);
142     end
143     flsf(q+1) = flsf(q-1)+2*DELTA;
144     if flsf(q+1) > MaxVar(1)
145         flsf(q+1) = MaxVar(1);
146     end
147 end
148
149 flss(1) = MinVar(2)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition

150 if flss(1) < 0
151     flss(1) = 0;
152 end
153 flss(2) = MinVar(2)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
154 if flss(2) < 0
155     flss(2) = 0;

```

```

156 end
157 for q=3:2:2*b(2)-1
158     flss(q) = flss(q-2)+2*DELTA;
159     if flss(q) > MaxVar(2)
160         flss(q) = MaxVar(2);
161     end
162     flss(q+1) = flss(q-1)+2*DELTA;
163     if flss(q+1) > MaxVar(2)
164         flss(q+1) = MaxVar(2);
165     end
166 end
167
168
169 flsK(1) = MinVar(3)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition
170 if flsK(1) < 0
171     flsK(1) = 0;
172 end
173 flsK(2) = MinVar(3)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
174 if flsK(2) < 0
175     flsK(2) = 0;
176 end
177 for q=3:2:2*b(3)-1
178     flsK(q) = flsK(q-2)+2*DELTA;
179     if flsK(q) > MaxVar(3)
180         flsK(q) = MaxVar(3);
181     end
182     flsK(q+1) = flsK(q-1)+2*DELTA;
183     if flsK(q+1) > MaxVar(3)
184         flsK(q+1) = MaxVar(3);
185     end
186 end
187
188
189 flsl(1) = MinVar(4)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition
190 if flsl(1) < 0
191     flsl(1) = 0;
192 end
193 flsl(2) = MinVar(4)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
194 if flsl(2) < 0
195     flsl(2) = 0;

```

```

196 end
197 for q=3:2:2*b(4)-1
198     flsl(q) = flsl(q-2)+2*DELTA;
199     if flsl(q) > MaxVar(4)
200         flsl(q) = MaxVar(4);
201     end
202     flsl(q+1) = flsl(q-1)+2*DELTA;
203     if flsl(q+1) > MaxVar(4)
204         flsl(q+1) = MaxVar(4);
205     end
206 end
207
208 flsm(1) = MinVar(5)+(perm(IT)-1)*Delta+rand(1)*Delta;
                % fls: first level partition
209 if flsm(1) < 0
210     flsm(1) = 0;
211 end
212 flsm(2) = MinVar(5)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
213 if flsm(2) < 0
214     flsm(2) = 0;
215 end
216 for q=3:2:2*b(5)-1
217     flsm(q) = flsm(q-2)+2*DELTA;
218     if flsm(q) > MaxVar(5)
219         flsm(q) = MaxVar(5);
220     end
221     flsm(q+1) = flsm(q-1)+2*DELTA;
222     if flsm(q+1) > MaxVar(5)
223         flsm(q+1) = MaxVar(5);
224     end
225 end
226
227
228 flsn(1) = MinVar(6)+(perm(IT)-1)*Delta+rand(1)*Delta;
                % fls: first level partition
229 if flsn(1) < 0
230     flsn(1) = 0;
231 end
232 flsn(2) = MinVar(6)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
233 if flsn(2) < 0
234     flsn(2) = 0;
235 end

```

```

236 for q=3:2:2*b(6)-1
237     flsn(q) = flsn(q-2)+2*DELTA;
238     if flsn(q) > MaxVar(6)
239         flsn(q) = MaxVar(6);
240     end
241     flsn(q+1) = flsn(q-1)+2*DELTA;
242     if flsn(q+1) > MaxVar(6)
243         flsn(q+1) = MaxVar(6);
244     end
245 end
246
247
248
249 flso(1) = MinVar(7)+(perm(IT)-1)*Delta+rand(1)*Delta;
                % fls: first level partition
250 if flso(1) < 0
251     flso(1) = 0;
252 end
253 flso(2) = MinVar(5)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
254 if flso(2) < 0
255     flso(2) = 0;
256 end
257 for q=3:2:2*b(7)-1
258     flso(q) = flso(q-2)+2*DELTA;
259     if flso(q) > MaxVar(7)
260         flso(q) = MaxVar(7);
261     end
262     flso(q+1) = flso(q-1)+2*DELTA;
263     if flso(q+1) > MaxVar(7)
264         flso(q+1) = MaxVar(7);
265     end
266 end
267
268
269 flsp(1) = MinVar(8)+(perm(IT)-1)*Delta+rand(1)*Delta;
                % fls: first level partition
270 if flsp(1) < 0
271     flsp(1) = 0;
272 end
273 flsp(2) = MinVar(8)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
274 if flsp(2) < 0
275     flsp(2) = 0;

```

```

276 end
277 for q=3:2:2*b(8)-1
278     flsp(q) = flsp(q-2)+2*DELTA;
279     if flsp(q) > MaxVar(8)
280         flsp(q) = MaxVar(8);
281     end
282     flsp(q+1) = flsp(q-1)+2*DELTA;
283     if flsp(q+1) > MaxVar(8)
284         flsp(q+1) = MaxVar(8);
285     end
286 end
287
288
289
290 flsC(1) = MinVar(9)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition
291 if flsC(1) < 0
292     flsC(1) = 0;
293 end
294 flsC(2) = MinVar(9)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta; %
295 if flsC(2) < 0
296     flsC(2) = 0;
297 end
298 for q=3:2:2*b(9)-1
299     flsC(q) = flsm(q-2)+2*DELTA;
300     if flsC(q) > MaxVar(9)
301         flsC(q) = MaxVar(9);
302     end
303     flsC(q+1) = flsm(q-1)+2*DELTA;
304     if flsC(q+1) > MaxVar(9)
305         flsC(q+1) = MaxVar(9);
306     end
307 end
308
309
310 flsR(1) = MinVar(10)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition
311 if flsR(1) < 0
312     flsR(1) = 0;
313 end
314 flsR(2) = MinVar(10)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
           %

```

```

315 if flsR(2) < 0
316     flsR(2) = 0;
317 end
318 for q=3:2:2*b(10)-1
319     flsR(q) = flsR(q-2)+2*DELTA;
320     if flsR(q) > MaxVar(10)
321         flsR(q) = MaxVar(10);
322     end
323     flsR(q+1) = flsR(q-1)+2*DELTA;
324     if flsR(q+1) > MaxVar(10)
325         flsR(q+1) = MaxVar(10);
326     end
327 end
328
329
330 flsd(1) = MinVar(11)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition
331 if flsd(1) < 0
332     flsd(1) = 0;
333 end
334 flsd(2) = MinVar(11)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
           %
335 if flsd(2) < 0
336     flsd(2) = 0;
337 end
338 for q=3:2:2*b(11)-1
339     flsd(q) = flsd(q-2)+2*DELTA;
340     if flsd(q) > MaxVar(11)
341         flsd(q) = MaxVar(11);
342     end
343     flsd(q+1) = flsd(q-1)+2*DELTA;
344     if flsd(q+1) > MaxVar(11)
345         flsd(q+1) = MaxVar(11);
346     end
347 end
348
349
350 flst(1) = MinVar(12)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition
351 if flst(1) < 0
352     flst(1) = 0;
353 end

```

```

354 flst(2) = MinVar(12)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
      %
355 if flst(2) < 0
356     flst(2) = 0;
357 end
358 for q=3:2:2*b(12)-1
359     flst(q) = flst(q-2)+2*DELTA;
360     if flst(q) > MaxVar(12)
361         flst(q) = MaxVar(12);
362     end
363     flst(q+1) = flst(q-1)+2*DELTA;
364     if flst(q+1) > MaxVar(12)
365         flst(q+1) = MaxVar(12);
366     end
367 end
368
369
370 flsu(1) = MinVar(13)+(perm(IT)-1)*Delta+rand(1)*Delta;
      % fls: first level partition
371 if flsu(1) < 0
372     flsu(1) = 0;
373 end
374 flsu(2) = MinVar(13)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
      %
375 if flsu(2) < 0
376     flsu(2) = 0;
377 end
378 for q=3:2:2*b(13)-1
379     flsu(q) = flsu(q-2)+2*DELTA;
380     if flsu(q) > MaxVar(13)
381         flsu(q) = MaxVar(13);
382     end
383     flsu(q+1) = flsu(q-1)+2*DELTA;
384     if flsu(q+1) > MaxVar(13)
385         flsu(q+1) = MaxVar(13);
386     end
387 end
388
389
390 flsv(1) = MinVar(14)+(perm(IT)-1)*Delta+rand(1)*Delta;
      % fls: first level partition
391 if flsv(1) < 0

```

```

392     flsv(1) = 0;
393 end
394 flsv(2) = MinVar(14)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
395     %
396     if flsv(2) < 0
397         flsv(2) = 0;
398     end
399     for q=3:2:2*b(14)-1
400         flsv(q) = flsv(q-2)+2*DELTA;
401         if flsv(q) > MaxVar(14)
402             flsv(q) = MaxVar(14);
403         end
404         flsv(q+1) = flsv(q-1)+2*DELTA;
405         if flsv(q+1) > MaxVar(14)
406             flsv(q+1) = MaxVar(14);
407         end
408     end
409     flsw(1) = MinVar(15)+(perm(IT)-1)*Delta+rand(1)*Delta;
410     % fls: first level partition
411     if flsw(1) < 0
412         flsw(1) = 0;
413     end
414     flsw(2) = MinVar(15)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
415     %
416     if flsw(2) < 0
417         flsw(2) = 0;
418     end
419     for q=3:2:2*b(15)-1
420         flsw(q) = flsw(q-2)+2*DELTA;
421         if flsw(q) > MaxVar(15)
422             flsw(q) = MaxVar(15);
423         end
424         flsw(q+1) = flsw(q-1)+2*DELTA;
425         if flsw(q+1) > MaxVar(15)
426             flsw(q+1) = MaxVar(15);
427         end
428     end
429     flse(1) = MinVar(16)+(perm(IT)-1)*Delta+rand(1)*Delta;

```

```
                                % fls: first level partition
431 if flse(1) < 0
432     flse(1) = 0;
433 end
434 flse(2) = MinVar(16)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
        %
435 if flse(2) < 0
436     flse(2) = 0;
437 end
438 for q=3:2:2*b(16)-1
439     flse(q) = flse(q-2)+2*DELTA;
440     if flse(q) > MaxVar(16)
441         flse(q) = MaxVar(16);
442     end
443     flse(q+1) = flse(q-1)+2*DELTA;
444     if flse(q+1) > MaxVar(16)
445         flse(q+1) = MaxVar(16);
446     end
447 end
448
449
450
451
452
453 flsy(1) = MinVar(17)+(perm(IT)-1)*Delta+rand(1)*Delta;
                                % fls: first level partition
454 if flsy(1) < 0
455     flsy(1) = 0;
456 end
457 flsy(2) = MinVar(17)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
        %
458 if flsy(2) < 0
459     flsy(2) = 0;
460 end
461 for q=3:2:2*b(17)-1
462     flsy(q) = flsy(q-2)+2*DELTA;
463     if flsy(q) > MaxVar(17)
464         flsy(q) = MaxVar(17);
465     end
466     flsy(q+1) = flsy(q-1)+2*DELTA;
467     if flsy(q+1) > MaxVar(17)
468         flsy(q+1) = MaxVar(17);
```

```

469     end
470 end
471
472
473
474 flsz(1) = MinVar(18)+(perm(IT)-1)*Delta+rand(1)*Delta;
           % fls: first level partition
475 if flsz(1) < 0
476     flsz(1) = 0;
477 end
478 flsz(2) = MinVar(18)+DELTA+(MaxIter-perm(IT)+1)*Delta+rand(1)*Delta;
           %
479 if flsz(2) < 0
480     flsz(2) = 0;
481 end
482 for q=3:2:2*b(18)-1
483     flsz(q) = flsz(q-2)+2*DELTA;
484     if flsz(q) > MaxVar(18)
485         flsz(q) = MaxVar(18);
486     end
487     flsz(q+1) = flsz(q-1)+2*DELTA;
488     if flsz(q+1) > MaxVar(18)
489         flsz(q+1) = MaxVar(18);
490     end
491 end
492
493
494
495 % Destribution of first level candidate solution as a Matrix
496 for i=1:a(1)
497     for j=1:a(2)
498         for K=1:a(3)
499             for l=1:a(4)
500                 for m=1:a(5)
501                     for n=1:a(6)
502                         for o=1:a(7)
503                             for p=1:a(8)
504                                 for C=1:a(9)
505                                     for R=1:a(10)
506                                         for d=1:a(11)
507                                             for t=1:a(12)
508                                                 for u=1:a(13)

```

```
,K,l,m,n,o,p,C,R,d,t,u,v,w,e,y,z)
(K) flsl(l) flsm(m) flsn(n) flso(o)
(d) flst(t) flsu(u) flsv(v) flsw(w)
%do it 18times
```

```

534 end
535 %end
536
537
538
539
540
541
542 % Distribution of the above candidate solution as a vector
543 A = sortrows(cat(1,Candidate01.Sol));
544
545 kk=1;
546 for i=1:a(1)*a(2)*a(3)*a(4)*a(5)*a(6)*a(7)*a(8)*a(9)*a(10)*a(11)*a
    (12)*a(13)*a(14)*a(15)*a(16)*a(17)*a(18) % number of
    partition on the first level (18 times)
547
548     BeST(i).SolU = A(i,:);
549     if i == 1
550         [xL,fL] = PSO_Menyatso(BeST(i).SolU,nvar_L,Npop,
            Max_iter_PSO);
551
552         Best(i).SolL = xL;
553         Best(i).SolU = BeST(i).SolU;
554         Best(i).CostL = fL;
555         Best(i).CostU = ObjectiveFunctionU(Best(i).SolU,Best(i).
            SolL)+...
556             lambda*(sum(max(myconstraintU_INEQ(Best(i).SolU),0)
                ));
557     end % if i == 1;
558     if (i<=2^18 & i >= (2^18-254)) | (mod(i,128) == 0)
559
560         % Call the PSO to obtain the corresponding best Lowervariable
            for each choice of Best(i).Sol
561
562         [xL,fL] = PSO_Menyatso(BeST(i).SolU,nvar_L,Npop,Max_iter_PSO);
563
564
565         Best(i).SolL = xL;
566         Best(i).SolU = BeST(i).SolU;
567         Best(i).CostL = fL;
568         Best(i).CostU = ObjectiveFunctionU(Best(i).SolU,Best(i).SolL)
            +...

```

```

569         lambda*(sum(max(myconstraintU_INEQ(Best(i).SolU),0)));
570     if i == 1
571         Best_T = Best(1); % Current
572         iteration best upper
573 %         elseif i~= 1 && Best(i).CostU <= Best(i-1).CostU
574         elseif i~= 1 && Best(i).CostU <= Best(kk).CostU
575             Best_T = Best(i);
576
577         else
578             Best_T = Best(kk);
579 %             Best_T = Best(i-1);
580
581         end % if i==1
582         kk = i;
583     end %if mod ...
584 end % for i=1:a(1)*a(2)
585
586     BEST(IT).COSTU= ObjectiveFunctionU(Best_T.SolU,Best_T.SolL);
587     BEST(IT).COSTL= Best_T.CostL; %only lower level constraints
588     BEST(IT).SOLU = Best_T.SolU;
589     BEST(IT).SOLL = Best_T.SolL;
590
591 % Selecting best solution for the current iteration
592
593     if IT == 1
594         BEst = BEST(IT);
595     elseif IT~= 1 && BEST(IT).COSTU <= GBEST(IT-1).COSTU
596         BEst = BEST(IT);
597
598     else
599         BEst = GBEST(IT-1);
600
601     end
602
603     GBEST(IT) = BEst;
604
605 % Displaying results on Command Window
606
607     disp(['Iteration ' num2str(IT) ': Upper Best Cost = ' num2str(GBEST
        (IT).COSTU) ': Lower Best Cost = ' num2str(GBEST(IT).COSTL)]); %
        ': Optimal Solution = ' num2str(GBEST(IT).SOLU)]); %

```

```

608
609 end % for IT=1:MaxIter
610
611
612 disp(['The Optimal Solution for the upper level is = ' num2str(
    GBEST(MaxIter).SOLU) ': For the lower level problem is = '
    num2str(GBEST(MaxIter).SOLL)]);

```

A.2 The binary particle swarm optimisation MatLab code

```

1
2 %% Adapted and modified from the code produced by
3 % Jogendra Karthika Sri Sai Balaji Dulipala
4 % Last updated:– 19/4/2020
5 % contact details:– balajijogendra@gmail.com
6 %% Non–linear Inequality and equality constrained PSO, by Bame
    Menyatso & Semu M. Kassa
7 function [Xmin,Fmin]=PSO_Menyatso(y,nvars,Npop,max_it)
8 %function [Xmin,Fmin,SUM_Constraints]=PSO_non_linear_constraint(
    objective_function ,constraints ,constraints_eq ,LB,UB,nvars ,Npop,
    max_it)
9 %%
10 % Objective_function      function to be optimized
11 % nvars                   number of variables to be optimized
12 % npop                    number of populations
13 % LB                      Lower bounds of the problem
14 % UB                      Upper bounds of the problem
15 % max_it                  Maximum iterations
16 % constraints              Non–linear or linear inequality
    constraints
17 % constraints_eq           Non–linear or linear equality constraints
18 %%
19 format long
20     w=5;
21     c1=2;
22     c2=2;
23     alpha=0.75;
24     UppVar = y;
25
26     epss=1e–6;
27     for i=1:Npop
28         Tmp1 = rand(1,nvars);
29         pop(i).position=(rand(1,nvars)<Tmp1);

```

```

30     pop(i).cost=ObjectiveFunctionL(UppVar,pop(i).position);
31     c = [abs(myconstraintL_Eq(pop(i).position));
32          abs(myconstraintU(pop(i).position))];
33     %             myconstraint_U_Ineq(UppVar,pop(i).position)];
34     %     c = [constraints(pop(i).position);
35     %         abs(constraints_eq(pop(i).position))];
36     pop(i).const=sum(c(c>epss));
37     pop(i).velocity=zeros(1,nvars);
38 end
39 %—————Sort to selection the global best and local best for
    1st iteration —————
40 X_Minus=[];
41 aa=[pop.const];
42 COST_MINUS=[pop(aa<=epss).cost];
43 if ~isempty(COST_MINUS)
44     X_Minus=pop(aa<=epss);
45     [~,INDEX_M]=sort(COST_MINUS);
46     X_Minus=X_Minus(INDEX_M);
47 end
48 X_PLUS=[];
49 SUM_C_PLUS=aa(aa>epss);
50 COST_PLUS=[pop(aa>epss).cost];
51 if ~isempty(SUM_C_PLUS)
52     AD=unique(SUM_C_PLUS);
53     if size(AD,2)==Npop
54         X_PLUS=pop(aa>epss);
55         [~,INDEX_P]=sort(SUM_C_PLUS);
56     else
57         X_PLUS=pop(aa>epss);
58         [OR,INDEX]=sort(SUM_C_PLUS);
59         COST_PLUS=COST_PLUS(INDEX);
60
61         kk=1;
62         N=0;
63
64         for m=1:size(AD,2)
65             B=length(find(AD(m)==OR));
66             [~,IND]=sort(COST_PLUS(kk:N+B));
67             INDEX_P(kk:N+B)=IND;
68             kk=B+1;
69             N=N+B;
70         end

```

```

71     end
72     X_PLUS=X_PLUS(INDEX_P);
73 end
74 pop=[X_Minus;X_PLUS];
75 %————— Forming the global best
76
77
78 %————— Main Loop of PSO
79
80 % disp('***** Particle Swarm Optimization (PSO)
81 % disp('***** Iterations Function Values Sum_Const
82 % disp('*****
83 % disp('*****
84 % disp('*****
85 % disp('*****
86 % disp('*****
87 % disp('*****
88 % disp('*****
89 % disp('*****
90 % disp('*****
91 % disp('*****
92 % disp('*****
93 % disp('*****
94 % disp('*****
95 % disp('*****
96 % disp('*****
97 % disp('*****
98 % disp('*****
99 % disp('*****
100 % disp('*****
101 % disp('*****
102 % disp('*****

```

```

103     for j=1:Npop
104         if (global_best.const<=epss && pop(j).const<=epss && pop(j).
            cost<global_best.cost) || (global_best.const>epss && pop(j).
            const<=epss)
105
106             global_best=pop(j);
107
108         elseif global_best.const>epss && pop(j).const>epss &&
            global_best.const>pop(j).const
109
110             global_best=pop(j);
111
112         end
113     end
114
115 end
116
117
118 Xmin=global_best.position;
119 Fmin=ObjectiveFunctionL(UppVar,Xmin);
120
121 end

```

Bibliography

- [1] Kessels, F. (2019). Traffic flow modelling. Springer International Publishing, Switzerland.
<https://doi.org/10.1007/978-3-319-78695-7>
- [2] Dhingra, S. L., and Gull, I. (2011). Traffic flow theory historical research perspectives. Transportation Research Circular, E-C149: 45-62.
- [3] Disbro, J. E., and Frame, M. (1989). Traffic flow theory and chaotic behavior (No. Special report 91). New York (State). Dept. of Transportation. Available from <https://rosap.ntl.bts.gov/view/dot/15604>, Accessed on March 11,2022.
- [4] Tan, H. F., Yang, Y., and Zhang, L. R. (2017). Improved BPR function to counter road impedance through OD matrix estimation of freight transportation. Journal of Highway and Transportation Research and Development, 11(2): 97-102.
- [5] Xie, C., Cheu, R. L., and Lee, D. H. (2001). Calibration-free arterial link speed estimation model using loop data. Journal of Transportation Engineering, 127(6): 507-514.
- [6] Kajalić, J., Čelar, N., and Stanković, S. (2018). Travel time estimation on urban street segment. Promet-Traffic and Transportation, 30(1): 115-120.
- [7] Chong, H. F., and Ng, D. W. K. (2016, December). Development of IoT device for traffic management system. In 2016 IEEE Student Conference on Research and Development (SCoReD) (pp. 1-6). IEEE.
- [8] Ramirez-Polo, L., Jimenez-Barros, M. A., Narvaez, V. V., and Daza, C. P. (2022). Simulation and Optimization of Traffic Lights For Vehicles Flow in High Traffic Areas. Procedia Computer Science, 198: 548-553.
- [9] Zhao, D., Dai, Y., and Zhang, Z. (2011). Computational intelligence in urban traffic signal control: A survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(4): 485-494.
- [10] Wiering, M. A., Veenen, J. V., Vreeken, J., and Koopman, A. (2004). Intelligent traffic light control. Available from https://www.academia.edu/es/2739749/Intelligent_traffic_light_control, Accessed on March 11, 2022.
- [11] Jamal, A., Tauhidur Rahman, M., Al-Ahmadi, H. M., Ullah, I., and Zahid, M. (2020). Intelligent intersection control for delay optimization: Using meta-heuristic search algorithms. Sustainability, 12(5): 1896.
- [12] Stoilova, K., Stoilov, T., and Pavlova, K. (2019). Traffic Management of Urban Network by Bi-Level Optimization. Journal Information Technologies and Control, (4): 12-21.

- [13] Eriskin, E., Karahancer, S., Terzi, S., and Saltan, M. (2017). Optimization of traffic signal timing at oversaturated intersections using elimination pairing system. *Procedia Engineering*, 187: 295-300.
- [14] Hao, J., Liu, X., Shen, X., and Feng, N. (2019). Bilevel Programming Model of Urban Public Transport Network under Fairness Constraints. *Discrete Dynamics in Nature and Society*, 2019. <https://doi.org/10.1155/2019/2930502>
- [15] Kamal, M. A. S., Imura, J. I., Ohata, A., Hayakawa, T., and Aihara, K. (2012). Control of traffic signals in a model predictive control framework. *IFAC Proceedings Volumes*, 45(24), 221-226. <https://doi.org/10.3182/20120912-3-BG-2031.00044>
- [16] Aboudolas, K., Papageorgiou, M., and Kosmatopoulos, E. (2009). Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, 17(2): 163-174.
- [17] Zhao, D., Dai, Y., and Zhang, Z. (2011). Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4): 485-494.
- [18] Dempe, S. (2002). *Foundations of bilevel programming*. Springer Science and Business Media, Dordrecht.
- [19] Dempe, S., and Zemkoho, A. B. (2013). The bilevel programming problem: reformulations, constraint qualifications and optimality conditions. *Mathematical Programming*, 138(1): 447-473.
- [20] Eberhart, R., and Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural network*, 4: 1942-1948.
- [21] West, D. B. (2001). *Introduction to graph theory*. Pearson Education, Inc. Delhi, India.
- [22] Balaban, A. T. (1985). Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, 25(3): 334-343.
- [23] Prathik, A., Uma, K., and Anuradha, J. (2016). An Overview of application of Graph theory. *International Journal of ChemTech Research*, 9(2): 242-248.
- [24] Joyner, D., Van Nguyen, M., and Cohen, N. (2010). *Algorithmic graph theory*. Available from <https://google-code-archive-downloads.storage.googleapis.com/v2/code.google.com/graphbook/latest-r1984.pdf> Accessed on July 11, 2021.
- [25] Thulasiraman, K., Swamy, M. N. (2011). *Graphs: theory and algorithms*. John Wiley Sons. New York.
- [26] Chartrand, G., Lesniak, L., and Zhang, P. (1996). *Graphs and digraphs*. Chapman and hall. London.
- [27] Gross, J. L., Yellen, J., and Anderson, M. (2018). *Graph theory and its applications*. Chapman and Hall. London.

- [28] Foulds, L. R. (2012). Graph theory applications. Springer Science and Business Media. New York.
- [29] Chartrand, G., and Zhang, P. (2013). A first course in graph theory. Dover publications, Inc. Mineola, New York.
- [30] Jungnickel, D., and Jungnickel, D. (2005). Graphs, networks and algorithms. Springer. Berlin.
- [31] Chong, E. K., and Zak, S. H. (2004). An introduction to optimization. John Wiley and Sons. Toronto, Canada.
- [32] Albers, D. J., and Reid, C. (1986). An interview with George B. Dantzig: The father of linear programming. The college mathematics journal, 17(4): 292-314.
- [33] Eiselt, H. A., Sandblom, C. L., and Sandblom, C. L. (2007). Linear programming and its applications. Berlin: Springer.
- [34] Nash, J. C. (2000). The (Dantzig) simplex method for linear programming. Computing in Science and Engineering, 2(1): 29-31.
- [35] Sorenson, H. W. (1970). Least-squares estimation: from Gauss to Kalman. IEEE spectrum, 7(7): 63-68.
- [36] Mehmood, T., and Ahmed, B. (2016). The diversity in the applications of partial least squares: an overview. Journal of Chemometrics, 30(1): 4-17.
- [37] Mangasarian, O. L. (1994). Nonlinear programming. Society for Industrial and Applied Mathematics.
- [38] Cottle, R. W., and Infanger, G. (2010). Harry Markowitz and the early history of quadratic programming. In Handbook of Portfolio Construction (pp. 179-211). Springer, Boston, MA. Chicago .
- [39] McCarl, B. A., Moskowitz, H., and Furtan, H. (1977). Quadratic programming applications. Omega, 5(1): 43-55.
- [40] Senn, P. R. (1996). Heinrich von Stackelberg in the history of economic ideas. Journal of Economic Studies. 23(5/6): 15-39. <https://doi.org/10.1108/01443589610154027>
- [41] Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordonez, F., and Kraus, S. (2008, May). Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems. 2: 895-902.
- [42] Nie, P. Y., and Zhang, P. A. (2008, July). A note on Stackelberg games. In 2008 Chinese Control and Decision Conference. IEEE. 1201-1203.
- [43] Simaan, M., and Cruz, J. B. (1973). On the Stackelberg strategy in nonzero-sum games. Journal of Optimization Theory and Applications, 11(5): 533-555.

- [44] Yin, Z., Korzhyk, D., Kiekintveld, C., Conitzer, V., and Tambe, M. (2010, May). Stackelberg vs. Nash in security games: interchangeability, equivalence, and uniqueness. In *AA-MAS*, 10: 6.
- [45] Ben-Ayed, O., Blair, C. E., Boyce, D. E., and LeBlanc, L. J. (1992). Construction of a real-world bilevel linear programming model of the highway network design problem. *Annals of Operations Research*, 34(1): 219-254.
- [46] Sadatrasou, S., Gholamian, M., and Shahanaghi, K. (2015). An application of data mining classification and bilevel programming for optimal credit allocation. *Decision Science Letters*, 4(1): 35-50.
- [47] Whittaker, G., Fare, R., Grosskopf, S., Barnhart, B., Bostian, M., Mueller-Warrant, G., and Griffith, S. (2017). Spatial targeting of agri-environmental policy using bilevel evolutionary optimization. *Omega*, 66: 15-27.
- [48] Bennett, K. P., Kunapuli, G., Hu, J., and Pang, J. S. (2008, June). Bilevel optimization and machine learning. In *IEEE World Congress on Computational Intelligence* (pp. 25-47). Springer, Berlin, Heidelberg.
- [49] Bard, J. F. (2013). *Practical bilevel optimization: algorithms and applications*. Springer Science and Business Media. Boston.
- [50] Ye, J. J. (2006). Constraint qualifications and KKT conditions for bilevel programming problems. *Mathematics of Operations Research*, 31(4): 811-824.
- [51] Allsop, R. E. (1972). Delay at a fixed time traffic signal-I: Theoretical analysis. *Transportation Science*, 6(3): 260-285.
- [52] Webster, F. V. (1958). Traffic signal settings, road research technical paper no. 39. Road Research Laboratory.
- [53] Eom, M., and Kim, B. I. (2020). The traffic signal control problem for intersections: a review. *European transport research review*, 12(1): 1-20.
- [54] Miller, A. J. (1963). Settings for fixed-cycle traffic signals. *Journal of the Operational Research Society*, 14(4): 373-386.
- [55] Hutchinson, T. P. (1972). Delay at a fixed time traffic signal-II: Numerical comparisons of some theoretical expressions. *Transportation Science*, 6(3): 286-305.
- [56] Akcelik, R. (1980). Time-dependent expressions for delay, stop rate and queue length at traffic signals. Internal report AIR 367-1. Australian road research board. Vermont south, Australia.
- [57] Stoilova, K., Stoilov, T., and Dimitrov, S. (2021). Bi-Level Optimization Model for Urban Traffic Control. *Cybernetics and Information Technologies*, 21(3): 108-126.
- [58] Stoilova, K., Stoilov, T., and Ivanov, V. (2017). Bi-level optimization as a tool for implementation of intelligent transportation systems. *Cybernetics and Information Technologies*, 17(2): 97-105.

- [59] Pang, M., Chen, C., and Ma, L. (2021). Bilevel Mixed Land Use–Transportation Model Based on Urban Road Network Balance. *Journal of Urban Planning and Development*, 147(4): 04021048.
- [60] Feng, C. M., and Wen, C. C. (2005). A fuzzy bi-level and multi-objective model to control traffic flow into the disaster area post earthquake. *Journal of the Eastern Asia Society for Transportation Studies*, 6: 4253-4268.
- [61] Zhao, W., Liu, R., and Ngoduy, D. (2021). A bilevel programming model for autonomous intersection control and trajectory planning. *Transportmetrica A: transport science*, 17(1): 34-58.
- [62] Patriksson, M., and Rockafellar, R. T. (2002). A mathematical model and descent algorithm for bilevel traffic management. *Transportation Science*, 36(3): 271-291.
- [63] Chiou, S. W. (2014). Optimal signal-setting for road network with maximum capacity. *Information Sciences*, 273: 287-303.
- [64] Cipriani, E., Fusco, G. (2004). Combined signal setting design and traffic assignment problem. *European Journal of Operational Research*, 155(3): 569-583.
- [65] Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1): 1-3.
- [66] Jamal, A., Tauhidur Rahman, M., Al-Ahmadi, H. M., Ullah, I., and Zahid, M. (2020). Intelligent intersection control for delay optimization: Using meta-heuristic search algorithms. *Sustainability*, 12(5): 1896.
- [67] Too, J., Abdullah, A. R., and Saad, N. (2019). A new co-evolution binary particle swarm optimization with multiple inertia weight strategy for feature selection, *Informatics*, 6: 21.
- [68] Preethi, P., Varghese, A., and Ashalatha, R. (2016). Modelling delay at signalized intersections under heterogeneous traffic conditions. *Transportation research procedia*, 17: 529-538.
- [69] Hoque, M. S., and Imran, A. (2007). Modification of Webster’s delay formula under non-lane based heterogeneous road traffic condition. *Journal of civil engineering (IEB)*, 35(2): 81-92.
- [70] Viti, F., and van Zuylen, H. J. (2004). Modeling queues at signalized intersections. *Transportation Research Record*, 1883(1): 68-77.
- [71] Shariff, M. A., and Faheem, M. I. (2012). Delay analysis at signalized intersections under mixed traffic conditions. *International journal for engineering research and industrial applications*, 215.
- [72] Seshan, C. R., and Tikekar, V. G. (1980, May). On Sharma-Swarup algorithm for time minimising transportation problems. In *Proceedings of the Indian Academy of Sciences-Mathematical Sciences*. Springer India, 89(2): 101-102
- [73] Puri, M. C. (2004). Two level hierarchical time minimizing transportation problem. *Top*, 12(2): 301-330.

- [74] Colson, B., Marcotte, P., and Savard, G. (2007). An overview of bilevel optimization. *Annals of operations research*, 153(1): 235-256.
- [75] Improta, G., and Sforza, A. (1982). Optimal offsets for traffic signal systems in urban networks. *Transportation Research Part B: Methodological*, 16(2):143-161.
- [76] Ouyang, Y., Zhang, R. Y., Lavaei, J., and Varaiya, P. (2020). Large-scale traffic signal offset optimization. *IEEE Transactions on Control of Network Systems*, 7(3), 1176-1187.
- [77] Abbas, M., Bullock, D., Head, L. (2001). Real-time offset transitioning algorithm for coordinating traffic signals. *Transportation Research Record*, 1748(1):26-39.
- [78] Aoki, N., Niimi, H., and Kamijo, S. (2013). Offset optimization of traffic signals to maximize green time overlap considering the vehicle arrival profile for two-dimensional networks. In *16th International IEEE Conference on Intelligent Transportation Systems*:738-743.
- [79] Amini, Z., Coogan, S., Flores, C., Skabardonis, A., Varaiya, P. (2018, August). Optimizing offsets in signalized traffic networks: A case study. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*: 614-619
- [80] Wong, S. C., Wong, W. T., Leung, C. M., and Tong, C. O. (2002). Group-based optimization of a time-dependent TRANSYT traffic model for area traffic control. *Transportation Research Part B: Methodological*, 36(4):291-312.
- [81] Robertson, D. I. (1969). 'TRANSYT'METHOD FOR AREA TRAFFIC CONTROL. *Traffic Engineering and Control*, 8(8).
- [82] Robertson, D. I. (1969). TRANSYT: a traffic network study tool.
- [83] Singh, L., Tripathi, S., and Arora, H. (2009). Time optimization for traffic signal control using genetic algorithm. *International Journal of Recent Trends in Engineering*, 2(2):4.
- [84] Dabiri, S., and Abbas, M. (2016). Arterial traffic signal optimization using Particle Swarm Optimization in an integrated VISSIM-MATLAB simulation environment. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*:766-771
- [85] Jing, P., Huang, H., Chen, L. (2017). An adaptive traffic signal control in a connected vehicle environment: A systematic review. *Information*, 8(3):101.
- [86] Xu, M., An, K., Vu, L. H., Ye, Z., Feng, J., and Chen, E. (2019). Optimizing multi-agent based urban traffic signal control system. *Journal of Intelligent Transportation Systems*, 23(4): 357-369.
- [87] Manzo, S., Nielsen, O. A., and Prato, C. G. (2014). Effects of Uncertainty in Speed-Flow Curve Parameters on a Large-Scale Model: Case Study of the Danish National Model. *Transportation Research Record*, 2429(1): 30-37.
- [88] Kassa, A. M., and Kassa, S. M. (2017). Deterministic solution approach for some classes of nonlinear multilevel programs with multiple followers. *Journal of Global Optimization*, 68(4): 729-747.

- [89] Goshu, N. N., and Kassa, S. M. (2020). A systematic sampling evolutionary (SSE) method for stochastic bilevel programming problems. *Computers and Operations Research*, 120: 104942. <https://doi.org/10.1016/j.cor.2020.104942>
- [90] Kassa, A. M., and Kassa, S. M. (2017). Deterministic solution approach for some classes of nonlinear multilevel programs with multiple followers. *Journal of Global Optimization*, 68(4): 729-747.
- [91] Heppner, H and Grenander, U., (1990). Stochastic non-linear model for coordinated bird flocks, In: Krasner, S (Ed.), *The ubiquity of chaos*. 233-238. Washington.
- [92] Kennedy, J., and Eberhart, R. C. (1997, October). A discrete binary version of the particle swarm algorithm. In *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*. 5: 4104-4108.
- [93] Lee, S., Soak, S., Oh, S., Pedrycz, W., and Jeon, M. (2008). Modified binary particle swarm optimization. *Progress in Natural Science*, 18(9): 1161-1166
- [94] Nezamabadi-pour, H., Rostami-Shahrbabaki, M., and Maghfoori-Farsangi, M. (2008). Binary particle swarm optimization: challenges and new solutions. *CSI J Comput Sci Eng*, 6(1): 21-32.
- [95] Tasgetiren, M. F., and Liang, Y. C. (2003). A binary particle swarm optimization algorithm for lot sizing problem. *Journal of Economic and Social Research*, 5(2): 1-20.
- [96] Khanesar, M. A., Teshnehlab, M., and Shoorehdeli, M. A. (2007, June). A novel binary particle swarm optimization. In *2007 Mediterranean conference on control and automation*. 1-6. IEEE.
- [97] Santibanez-Gonzalez, E., Montoya-Torres, J., and Huaccho Huatuco, L. (2016). A Binary Particle Swarm Optimization (BPSO) algorithm to solve a reverse logistics supply chain problem. *InImpact: The Journal of Innovation Impact*, 7(2):524.