Chapter 1

Introduction

This thesis proposes to simultaneously optimise five power-aware metrics for energy efficiency in maximising mobile ad-hoc network (MANET) lifetime, taking in consideration a realistic mobility model, using an ant colony optimisation (ACO) approach. The ACO approach addresses a hardware independent routing protocol implementation within MANETs. Using the proposed algorithms, a set of optimal solutions, the Pareto-optimal set, is found based on ACO. Section 1.1 introduces MANETs. Section 1.2 emphasizes the importance of reducing energy consumption for MANETs. Section 1.3 briefly describes how this thesis solves the energy consumption problem for MANETs. Section 1.4 states the primary objectives of this thesis. Section 1.5 explains the contribution of this thesis, and Section 1.6 gives the thesis outline.

1.1 Mobile Ad Hoc Network

Mobile devices coupled with wireless network interfaces will become an essential part of future computing environments consisting of infrastructured and infrastructure-less mobile networks. Wireless local area networks based on IEEE 802.11 technology are the most prevalent infrastructured mobile networks, where a mobile node communicates with a fixed base station, and thus a wireless link is limited to one hop between the node and the base station. A MANET is an infrastructure-less multi-hop network where each node communicates with other nodes directly or indirectly through intermediate nodes [35, 117]. Thus, all nodes in a MANET basically function as mobile routers participating in some routing protocol required for deciding and maintaining the (potentially) dynamically changing routes. Since MANETs are infrastructure-less, self-organizing, rapidly deployable wireless networks, they are highly suitable for applications such as:

- Military tactical operations [86, 100] for fast and possibly short term establishment of military communications for troop deployments in hostile and/or unknown environments.

- Search and rescue missions [114] for communication in areas with little or no wireless infrastructure support.

- Disaster relief operations [162] for communication in environments where the existing infrastructure is destroyed or left inoperable.

- Law enforcement [201] for secure and fast communication during law enforcement operations.

- Commercial use [26] for creating communications in exhibitions, conferences, and large gatherings.

Building such ad hoc networks poses a significant technical challenge due to multiple constraints imposed by the environment [36, 82]. As a result, the device used in the field must weigh as little as possible. Furthermore, since mobile devices are battery operated, they need to be energy conserving in order to maximise battery lifetime. Several technologies are in the process of being developed with the aim of achieving energy conservation by targeting specific components of the computer and optimising the energy consumption of these components. For example, low-power displays, algorithms to reduce the power consumption of disk drives, low-power I/0 devices, and low-power central processing units (CPUs) all contribute to overall energy savings.

As a result of the highly dynamic and distributed nature of MANETs, routing tends to be one of the key issues in MANETs [48, 174]. In particular, energy efficient routing may constitute the most important design criterion for MANETs since mobile nodes are powered by batteries with limited capacity. Power failure of a mobile node not only affects the node itself, but also the ability of the node to forward packets on behalf of other nodes and, thus, the overall network lifetime.

A mobile node not only consumes its battery energy when it is actively sending or receiving packets, but it also consumes battery energy when idle and listening to the wireless medium for any possible communication requests from other nodes. Thus, energy efficient routing protocols minimise either the active communication energy which is required to transmit and receive data packets or the energy consumed during inactive periods. In terms of protocols that belong to the former category, the active communication energy may be reduced by adjusting the radio power of each node just enough to reach the receiving node, and no more. This transmission power control approach may be extended to determine the optimal routing path that will minimise the total transmission energy required to deliver data packets to their destinations. In terms of those protocols that minimise the energy consumed for data transfer during inactive periods, each node may save the inactivity energy by switching its mode of operation to sleep/power-down mode or by simply turning the mode of operation off when there is no data to be either transmitted or received. This will result in substantial energy savings, especially in cases where the network environment is characterised by a low duty cycle of communication activities. However, a well-designed routing protocol is required to guarantee data delivery even if most of the nodes sleep and do not forward packets to other nodes.

Another important approach to the optimisation of active communication energy is the load distribution approach [111]. While the primary focus of the above two approaches is to minimise the energy consumption of individual nodes, the main goal of the load distribution method is to balance the energy usage amongst the nodes and to maximise the network lifetime by avoiding over-utilised nodes when selecting a routing path. While it is not clear whether any particular algorithm or class of algorithms is the best for all scenarios, there are definite advantages and disadvantages to each protocol, and each protocol is suited for certain situations. However, it is possible to combine and integrate the existing solutions and metrics for energy efficiency in order to offer a more energy efficient routing mechanism.

1.2 Reducing Energy Consumption for MANETs

The research focus in MANETs, in the past years, has been on developing strategies for reducing the energy consumption of the communication subsystem and increasing the lifetime of the nodes. Recent studies have stressed the need for designing medium access control (MAC) and routing protocols to ensure longer battery life. Much research has been done on designing protocols that increase the lifetime of nodes and the network [33, 109]. The research and developed algorithms were done with reference to the MAC, network and transport layers. Power-aware MAC protocols such as the power-aware multi-access protocol (PAMAS) [185] have been designed for battery energy savings that intelligently turn off radios when they can not transmit or can not receive packets.

The mobile ad-hoc network routing problem is difficult because of node mobility [7, 105, 206, 215]. With mobility, physically available routes may become invalid due to the topology change by node movement or link failure (i.e. may not be found by the routing algorithm), causing packets to be dropped and leading to throughput degradation and increasing control overhead. When two nodes previously within the transmission range move far away, the connection is lost. Vice versa, when two nodes move into the transmission range, a connection is gained. Thus, two conflicting goals are encountered: on the one hand, in order to optimise routes, frequent topology updates are required, while on the other hand, frequent topology updates result in higher message overhead.

Routing algorithms for mobile networks have been presented that attempt to optimise routes while attempting to keep message overhead small [33, 78, 97, 107, 116, 125, 138, 139, 157, 160, 182, 204]. Different routing protocols use one or more of a small set of metrics to determine optimal paths. Some of these metrics, however, have a negative impact on node and network life by inadvertently overusing the energy resources of a small set of nodes in favour of others.

Conserving power and carefully sharing the cost of routing packets will ensure that node and network life are increased. Singh *et al.* [184] presented several power-aware metrics that do result in energy-efficient routes. These metrics are

1. to minimise energy consumed per packet,
2. to maximise time to network partition,
3. to minimise variance in node power levels,
4. to minimise cost per packet, and
5. to minimise maximum node cost.

Many real-world problems require the simultaneous optimisation of a number of objective functions, referred to as *multi-objective optimisation problems* (MOP) [37, 147]. Some of these objectives may be in conflict with one another. A typical MOP simultaneously involves some competing objectives. The solution to a MOP requires a suitable definition of optimality (usually called *Pareto optimality*). MOPs normally have not one, but an infinite set of solutions, which represent possible trade-offs among the objectives (such solutions constitute the so-called *Pareto-optimal set* [155]). For MOPs the main task is to optimise a vector function, say $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x}))$. A typical way to approach these problems is to transform the MOPs into single-objective (or scalar) problems (e.g. by using a linear aggregating function). This approach does, indeed, make sense if the functions $f_1, f_2, ..., f_n$ are of the same type and expressed in the same units, but otherwise (for instance, if f_1 denotes distance, f_2 denotes time, and so on) the scalarised problem might be meaningless. Also transforming the MOPs into a single-objective problem might not be feasible when there are trade-offs among the sub-objectives.

1.3 Solving Multi-Objective Power-Aware Metrics with Ant Colony Optimisation Algorithms

This research presents five multi-objective algorithms for simultaneously optimising the five power-aware metrics used for determining routes in wireless ad hoc networks, whilst, at the same time, taking into account the reference point group mobility (RPGM) model [29].

The proposed algorithms constitute new versions of the multi-objective ant colony system [53], the max-min ant system [195], and the multiple colony ant system [76].

1.4 Objectives

The primary objectives of this thesis can be summarised as follows:

- To develop and test multi-objective ant optimisation algorithms, in order to simultaneously optimise the five power-aware routing metrics described in Section 1.2 while taking in consideration a realistic mobility model.
- To obtain empirical results to support the predictions offered by the proposed algorithms.

1.5 Contributions

The main contributions of this work are:

- Development of the first dynamic multi-objective ant optimisation algorithms to simultaneously optimise five power-aware objectives for energy efficiency and the MANET's lifetime, taking in consideration a realistic mobility model.
- The evaluation of the scalability of the five dynamic multi-objective ant optimisation algorithms with different network sizes.
- The experimental evaluation of the response of the five algorithms to varying node mobility.

1.6 Thesis Outline

Chapter 2 contains an introduction to MANETs which is, in turn, followed by a comprehensive review of recent work conducted which addresses energy efficient and low-power design within the network layer. The chapter also examines different mobility models.

Chapter 3 starts with an introduction to combinatorial optimisation. An overview of the foraging behaviour of real ants is then presented, and this is, in turn, followed by a discussion on ant colony optimisation algorithms.

Chapter 4 commences with a theoretical overview of the multi-objective problem (MOP), which is followed by a discussion of the concepts of Pareto-optimal set and Pareto-optimal front. In view of the fact that ACO methods for MOO problems form the basis of the work presented in this thesis the focus then shifts to the application of ACO algorithms to MOO problems. A brief introduction of the evolutionary algorithms for solving multi-objective optimisation problems is then presented, and the NSGA-II algorithm is described in detail. Performance metrics used to compare multi-objective algorithms are discussed.

Chapter 5 discusses the concept of optimisation within dynamic environments since the optimisation problem considered in this thesis is within the context of a dynamic environment. A formal definition of a dynamic optimisation problem (DOP) is presented, followed by an overview of the main characteristics of DOPs. The ant algorithms for DOP are discussed, the performance metrics for DOP are described, and dynamic multi-objective optimisation examined.

Chapter 6 describes in detail the five metrics for power-aware routing, and the multi-objective optimisation problem is reformulated. This is followed by a description of the new multi-objective ant colony optimisation algorithms for simultaneously optimising the five power-aware routing metrics. These new algorithms are adaptations of multi-objective ant colony optimisation algorithms.

Chapter 7 presents an empirical analysis of the behaviour of the multi-objective ant colony optimisation algorithms which were introduced in Chapter 6. The results are presented and evaluated.

Chapter 8 presents a summary of the findings of this thesis. Topics for future research are also discussed.

Appendix A and appendix B respectively list abbreviations and symbols used in this thesis along with their explanations.

Appendix C displays symbols used in this thesis to formulate the multi-objective optimisation problem for power-aware routing metrics.

Appendix D and appendix E respectively contain tables and graphs to visualise the results of the empirical analysis of the ant-based algorithm control parameters.

Appendix F displays the algorithms results for different scenarios.

Appendix G presents a three dimensional graphs to illustrate the influence of change frequency and change severity on the performance metrics.

Appendix H contains the results of the Mann-Whitney U test for each pair of algorithms to be compared.

Appendix I summarises the optimisation criteria results.

Appendix J presents FluxViz graphs to illustrate the influence of change frequency and change severity on the optimisation criteria for different number of nodes.

Chapter 2

Energy Efficient Network Protocols for Mobile Ad Hoc Networks

This chapter provides a review of mobile ad hoc networks (MANETs) and their main components. A discussion of multi-hop MANETs is also presented. An overview of the different mobility models is given. This is followed by a survey of different energy efficient protocols within the network layer.

2.1 Introduction

A mobile ad hoc network (MANET) refers to a collection of wireless mobile nodes which form a self-configuring network without using any existing, fixed infrastructure [35, 117]. As wireless networks become an integral component of the modern communication infrastructure, energy efficiency becomes an important design consideration in view of the limited battery life of mobile terminals. Power conservation techniques are commonly used in the hardware design of such systems. Since the network interface is a significant consumer of power, there has been much research conducted into low-power design of the entire network protocol stack of wireless networks in an effort to enhance energy efficiency. This chapter presents a comprehensive summary of recent work done on energy efficient and low-power design within the network layer. Table 2.1 illustrates the open systems interconnection (OSI) based protocol stack for a generic wireless network.

The mobility model plays a very important role in determining the protocol performance in MANETs [47, 123, 174]. Thus, it is essential to study and analyse various mobility models and their effect on MANET protocols. This chapter surveys and examines different mobility models which have been proposed in recent research literature.

A great body of knowledge about MANETs has been produced and many researchers in the field are now trying to apply this knowledge to the field of wireless sensor networks

Table 2.1: Protocol stack for a generic wireless network

Application and Services
OS and Middleware
Transport
Network
Data link (LLC, MAC)
Physical

(WSNs) because MANETs and WSNs are very similar. Both are distributed wireless networks and routing between two nodes may involve the use of intermediate relay nodes. Both networks are usually battery-powered and therefore there is a big concern on minimizing power consumption. Both MANETs and WSNs use a wireless channel and finally, self-management is necessary because of the distributed nature of both networks.

The remainder of this chapter is organised as follows. Section 2.2 provides a description of MANETs and their main components. Section 2.3 discusses multi-hop MANETs. Section 2.4 surveys and examines different mobility models. Section 2.5 discusses power-aware protocols within the network layer. Section 2.6 discusses bio-inspired routing for MANETs and, finally, Section 2.7 summarises and concludes the chapter.

2.2 Power Consumption and Communication for MANETs

In ad hoc mobile wireless networks, energy consumption is an important issue as most mobile hosts operate on limited battery resources. Conservation of energy is, therefore, critical in order to prolong the lifetime of the network. Instruction level modelling mobile systems run on the limited energy which is available within a battery. Thus the energy consumed by the system, or by the software running on the system, determines the duration of battery life.

There are two main consumers of energy on a MANET node, namely, the central processing unit and the radio (transmitter/receiver). These energy consumers are described in the following subsections.

2.2.1 Central Processing Unit

The microprocessor draws a current each time a program is executed. The average power consumed by a microprocessor while running a certain program is given by $P_o = I_c * V_s$, where P_o is the average power, I_c is the average current, and V_s the supply voltage [202]. Since power is the rate at which energy is consumed, the energy consumed by a program is given by $E = P_o * T_e$ where T_e is the execution time of the program. T_e, in turn, is given by $T_e = N_c * c_p$ where N_c is the number of clock cycles utilised by the program and c_p is the clock period.

Thus, together with the power cost of the hardware component it is important to also estimate the power cost of the software component. In order to systematically analyse the power cost of the software component it is necessary to estimate the power cost of the individual instructions.

Power Cost Measurement Method

It is obvious that a good instruction-level energy model is essential in order to evaluate software in terms of the power metric and also to help search the design space for low power software implementations. The instruction-level power analysis technique was first developed at Princeton University [128, 202, 203]. The technique is based on measurements of the current drawn by the processor as it executes certain instructions repeatedly. Power models for the Intel 486DX2, the Fujitsu SPARClite 934, and the Fujitsu DSP processor have been developed using this method. In order to model the energy consumption of the microprocessor, individual instructions must be considered. Each instruction involves specific processing demands across various units of the CPU. In terms of this model each instruction in the instruction set is assigned a fixed energy cost which is termed the base energy cost. The variation in the base costs of a given instruction is then quantified as a result of different operands and address values. The base energy cost of a program is based on the sum of the base energy costs of each instruction executed. However, during the execution of a program, certain inter-instruction effects occur of which the energy contribution is not accounted for if only the base costs are taken into account. The circuit state constitutes the first type of inter-instruction effect, while the second type is related to the resource constraints that may lead to stalls and cache misses. The energy cost of these effects is also modelled and used to obtain the

Table 2.2: Subset of the base cost table for the 486DX2

Number	*Instruction*	*Base Cost(mA)*	*Cycles*
1	NOP	275.7	1
2	MOV DX,BX	302.4	1
3	MOV DX,[BX]	428.3	1
4	MOV DX,[BX][DI]	409.0	2
5	MOV [BX],DX	521.7	1
6	MOV [BX][DI],DX	451.7	2
7	ADD DX,BX	313.6	1
8	ADD DX,[BX]	400.1	2
9	ADD [BX],DX	415.7	3
10	SAL BX,1	300.8	3
11	SAL BX,CL	306.5	3
12	LEA DX,[BX]	364.4	1
13	LEA DX,[BX][DI]	345.2	2
14	JMP label	373.0	3
15	JZ label	375.7	3
16	JZ label	355.9	1
17	CMP BX,DX	298.2	1
18	CMP [BX],DX	388.0	2

total energy cost of a program.

Certain instructions involve multiple cycles within a given pipeline stage. The base energy cost of the instruction is merely the observed average current value multiplied by the number of cycles taken by the instruction in that specific stage. Table 2.2 [203] summarises the CPU base costs for certain 486DX2 instructions.

Methodologies for analysing the energy consumption of embedded software help to verify whether an embedded design meets the energy constraints. These methodologies may also be used to guide the design of embedded software in such a way that the design meets these constraints.

Operating Modes

The CPU has four operating modes, namely, power down, power save, active, and idle. Power down shuts down the processor while the external interrupts (switch, button) remain on. Power save shuts down the processor while external interrupts and an asynchronous timer (external oscillator) remain on. The active mode leaves everything on, while the idle mode shuts down the processor while the peripheral universal asynchronous receiver transmitter (UART), serial peripheral interface (SPI), and analogue to digital converter (ADC) remain on.

2.2.2 Radio

Radios have three operating modes, namely, transmit, receive, and power off. Thus, the total power consumption at a node is dependent on the operating modes of the two subsystems, i.e. the CPU and radio. All two subsystems must be used sparingly in order to prolong the lifetime of the network. As has been proved by several wireless network researchers [12, 50, 164, 187] the radio consumes the most energy of these two subsystems. At the communication distances which are typical in MANETs, the receiving and transmitting data involve similar costs [166]. Therefore, it is essential that protocols that account explicitly for receive power be developed. The primary cost of radio power consumption does not come from the number of packets transmitted but from the time spent by the nodes in a state of idle listening.

Idle listening is the time spent listening while waiting to receive packets. Stemm and Katz [189] observed that idle listening dominates the energy costs of network interfaces in hand-held devices. Overhearing constitutes a secondary cost of radio power consumption. Since radios are broadcast mediums, nodes receive all communications, including those destined for other nodes. Clearly, in order to reduce power consumption in radios, the radio must be switched off during idle times.

An important challenge for the communication block unit is the design of a wakeup radio – a low-power radio that is able to receive very simple communication and, in particular, is able to detect whether communication with its own node is desired. In such a case the wakeup radio may power up the main radio that receives the actual communication. Unfortunately, switching the radio off means that a neighbouring node that detects an interesting event is not able to wake up the radio's node. This may lead to missed events and packets, thus both increasing latency and wasting energy. Accordingly, a challenge to radio technology is to develop an ultra low power communication channel which is able to wake up neighbouring nodes on demand. Currently, such wakeup radios still constitute an area of active research in chip design and communications research [131, 209].

Researchers are investigating protocols across software layers for controlling radio on/off times. Current approaches, which have been designed and implemented for real-world TinyOS applications, have focused on MAC-layer and application-layer techniques. Great Duck Island (GDI) for habitat monitoring [137] uses MAC-layer low-power-listening [197], while TinyDB [136] uses application-layer duty cycling [28] when deployed.

2.3 Multi-Hop MANETs

Multi-hop topologies play a significant role in MANETs. There are two main reasons for the importance of multi-hops. Firstly, the MANET itself has no wired or power-rich infrastructure. Therefore, in order to connect to the outside world, data must travel hop by hop to the nearest access point. Secondly, in terms of wireless communication, it is more energy efficient to transmit over several short distances than over a few long distances. Short distances also have better signal-to-noise ratios (because the environment is more homogenous), and this results in fewer retransmissions per hop due to packet loss [164].

However, there are inherent problems in multi-hop MANETs. Asynchronous events may trigger sudden bursts of traffic that could lead to collisions, congestion, and channel capture [214]. The problems that arise for wireless multi-hop networks are as a result of hidden nodes and exposed nodes. A hidden node refers to a node within the interfering range of the intended destination but out of the sensing range of the sender. Hidden nodes cause collisions at the destination when they transmit during the destination's reception. An exposed node refers to a node which is within the sensing range of the sender but out of the interfering range of the destination. Exposed nodes cease transmitting despite the fact that no collisions will take place at the destination.

In view of the fact that the MAC is a shared and scarce resource in a wireless multi-hop ad hoc network, efficient control of access to this shared media tends to become complicated. There has been considerable effort expended in designing MAC layer protocols, and several possible MAC layer protocols have been proposed [3, 217, 221]. The widely adopted IEEE 802.11 distributed coordination function (DCF) MAC protocol does not work well in wireless multi-hop networks primarily because DCF was designed for single communication cell networks [217]. The basic DCF access mechanism is carrier sense multiple access/collision avoidance (CSMA/CA) which uses physical carrier sense and the request-to-send/clear-to-send (RTS/CTS) handshake for collision avoidance [99]. The latter is extremely effective in avoiding hidden nodes in single communication cells. However, the hidden node problem still exists in multi-hop networks. There is no scheme which addresses the exposed node problem which is far more harmful in multi-hop networks. In order to understand the reason why this is the case, it must be stated that in a carrier sense wireless network,

- the communication (transmitting) range and sensing (receiving) range are not symmetric,
- the interfering range and sensing range are much larger than the communication range, and
- collisions occur at the receiver and not the transmitter.

The larger interfering and sensing ranges are the cause of severe unfairness while end-to-end packets yield problems in multi-hop networks. While larger interfering ranges exacerbate the hidden node problem, larger sensing ranges exacerbate the exposed node problem.

2.4 Mobility Models

Performance evaluation of a protocol for a MANET should test the protocol under realistic conditions including, but not limited to, a sensible transmission range, limited buffer space for the storage of messages, representative data traffic models, and realistic movements of the mobile users (i.e. a mobility model).

A mobility model should attempt to mimic the movements of real mobile nodes (MNs). Changes in the speed and direction of MNs must occur and within reasonable time slots; for example, it is not desirable for MNs to travel in straight lines at constant speeds throughout the course of the entire simulation because real MNs generally do not travel in such a restricted manner. Different entity and group mobility models for ad hoc networks have been developed [13, 29, 124]. This research uses the reference point group mobility model (RPGM) [29].

In the remainder of this section the reference point group mobility model is described. Reference point group mobility model (RPGM) is a group mobility model, where group movements are based upon the path traveled by a logical centre.

At the beginning of a simulation, the RPGM model divides mobile nodes into groups. Each group has a logical centre whose movement defines the entire group's motion behavior including location, speed, direction and acceleration.

Each individual node has one reference point which movement is determined by that of the group. The motion of each node is determined by two vectors, a group motion vector and an individual motion vector with respect to the node's reference point. The net motion vector of each node is the sum of the two vectors. The group motion is defined by specifying a sequence of check points along the path corresponding to given time intervals. As time goes by, a group moves from one check point to the next on a continuing basis. By proper selection of check points, many realistic situations can easily be modeled, where a group must reach predefined destinations within given time intervals to accomplish the group's task. There are different ways to create various moving scenarios by changing the pattern of check points [101].

This thesis generates group motion patterns using the random waypoint model. Every time the group reaches its destination, all nodes inside the group pause for a certain time and then restart the moving process.

The group motion vector maps out the location of the reference centre, while the node-dependent random motion vectors, added to the group motion vector, give the positions of the node. The RPGM model describes the group membership of a mobile node by its physical displacement from the group reference center. For example, at time t the location of the i-th node in the j-th group is given by

$$\mathbf{x}_{j,i}(t) = \mathbf{y}_{j,i}(t) + \mathbf{z}_{j,i}(t) \tag{2.1}$$

where $\mathbf{y}_{j,i}(t)$ is the reference location and $\mathbf{z}_{j,i}(t)$ is the local displacement.

The node-dependent local displacement or random motion vector, $\mathbf{z}_{j,i}(t)$, denotes the effect of the mobile nodes having their own localised movements while following the general group motion defined by the reference centre. RPGM is illustrated in Figure 2.1.

In Figure 2.1, five MNs are initially placed in the lower left-hand corner of the simulation area. A black square represents the group centre while the circles near the group centre represent the MNs in the group. One circle in Figure 2.1 is grey in order to distinguish it from the other MNs in the group. The movement of the grey circle will be examined. RPGM first calculates the reference point of each MN using the group motion

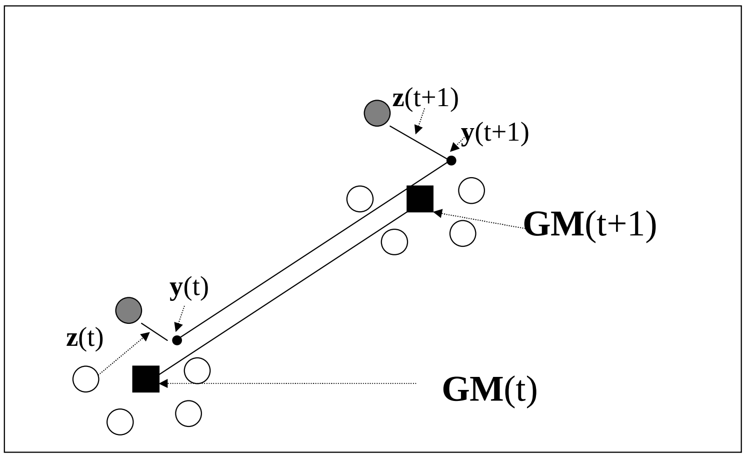

Figure 2.1: Movements of MNs using RPGM

vector, $\mathbf{GM}(t)$. $\mathbf{GM}(t)$ may be randomly chosen or predefined. The current reference point of the grey MN, $\mathbf{y}(t)$, moves towards the right-hand corner of the simulation area alongside the group centre. This location becomes the new reference point, $\mathbf{y}(t+1)$, for the grey MN. Finally, the new position of the grey MN is calculated by summing a random motion vector, $\mathbf{z}(t+1)$, with the new reference point, $\mathbf{y}(t+1)$. The length of $\mathbf{z}(t+1)$ is uniformly distributed within a specified radius centred at $\mathbf{y}(t+1)$ while its direction is uniformly distributed between 0 and 2π. This process is repeated for each MN in the group.

The RPGM model is a very good approach for realizing group mobility in tactical scenarios, because relative positions of nodes inside the groups can be modeled explicitly using an appropriate choice of parameters.

2.5 Network and Power Saving Routing Protocols

Routing of packets and congestion control are the main functions of the network layer [133]. In wireless mobile networks the network layer has the added functionality of routing under mobility constraints and mobility management including user location update. This section discusses energy efficient routing algorithms which were developed for wireless networks.

A switch is a device with several inputs and outputs that acts like a traffic junction to forward messages. Switches consist of a fabric and a set of ports which allow packets to flow from one link to another. Switches act as intersections as they forward messages arriving from one input link to a different output link. The advantage of a packet switched data network is its ability to scale beyond single hop communication. By connecting switches the network may be expanded to allow messages to be sent over multiple hops to non-adjacent nodes. Packet switched networks contain many pathways and this creates the problem of how to find a route from one node to another. A route connects nodes via one or more intermediate switches. Routes are discovered by exchanging information about links in order to construct route tables which contain directions to each node in the network. A packet switched network may be represented as a graph of nodes which are connected by links or edges. Each edge is assigned a cost which is derived from the characteristics of the link. The aim of a routing algorithm is to find the least cost path between two non-adjacent nodes in the network and to create a table which maps each destination to one of the output ports of the switches.

Ad hoc routing algorithms may be categorised into two types: table driven and on-demand [168]. Table driven algorithms send periodical broadcasts in order to maintain a route table. Algorithms classified as on-demand construct routes only when the routes are needed. Both types of algorithms use controlled flooding to find routes. The difference between these two approaches is the frequency with which flooding takes place. In terms of an on-demand protocol, flooding takes place only when a node desires a route to a new destination. On the other hand, because routing information is constantly propagated and maintained in table driven routing protocols, a route to every other node in the ad hoc network is always available, regardless of whether or not it is needed. This feature, although useful for datagram traffic, incurs substantial flooding traffic and power consumption.

Ad hoc routing algorithms may also be classified as active or reactive routing algorithms [216]. Routing algorithms that attempt to determine routes in advance are classified as active routing algorithms. Routing algorithms which take action only when a link is broken are classified as reactive.

Subsections 2.5.1 to 2.5.7 respectively discuss power efficient data gathering and aggregation protocols, dynamic source routing, distance vector routing, routing for maximum system lifetime, temporally ordered routing algorithms, volcano routing schemes, and destination sequenced distance vector, power-aware protocols. Subsection 2.5.8 presents a routing algorithm for network capacity maximisation (CMAX) in a wireless network. Subsection 2.5.9 presents the online maximum lifetime (OML) routing algorithm to maximise lifetime in a wireless network. Subsection 2.5.10 discusses other power-aware routing algorithms with respect to different metrics while Subsection 2.5.11 discusses power-aware routing algorithms for networks with frequent topological changes. These routing protocols employ power efficient methods for data gathering, aggregation and sending in order to achieve long network lifetimes.

2.5.1 Power Efficient Data Gathering and Aggregation Protocol

Tan and Korpeoglu [200] proposed two algorithms for enhancing network lifetime, namely, the power efficient data gathering and aggregation protocol (PEDAP) and the power efficient data gathering and aggregation protocol power-aware (PEDAP-PA). Both are routing protocols based on optimal minimum spanning trees (MST) [34]:

Definition 2.5.1. Minimum spanning tree:

Given a connected, undirected, graph $G = (V, L)$ a spanning tree, T_s, denotes an acyclic subset of edges, $T_s \subseteq L$, that connects all the vertices together.

Assuming G is weighted, the cost of a spanning tree, T_s, is the sum of edge weights, C_s, in the spanning tree, given as

$$C_s(T_s) = \sum_{(u,w)\,\in\,T_s} c_{s,uw} \tag{2.2}$$

where $(u, w) \in T_s$ is an edge and $c_{s,uw}$ is the weight of edge (u, w). A MST is a spanning tree of minimum weight.

PEDAP minimises the total energy expended in the system in a round of communication in which each round corresponds to an aggregation of data which is transmitted from different MANET nodes to the sink by computing a minimum spanning tree over the MANET. The data packets are then routed to the base station over the edges of the computed minimum spanning tree. PEDAP prolongs a satisfactory lifetime for the first node while providing a satisfactory lifetime for the last node. PEDAP constructs minimum energy consuming routing tables for each round of communication.

PEDAP-PA extends PEDAP by balancing the energy consumption among the nodes. PEDAP-PA provides a near optimal lifetime for the first node although the lifetime of the last node is slightly decreased.

Both algorithms consider that locations of wireless nodes are fixed and that the base station knows all the locations of the nodes a priori. The nodes are in direct communication range of each other and are able to transmit to and receive from the base station. The nodes aggregate or fuse the data received (via the minimum spanning tree) from the other nodes with their own data, and produce one packet only regardless of the number of packets received. In each round a special node is randomly selected to assume the responsibility for sending the fused data to the base station. Both algorithms assume that the quality of the system dramatically decreases after the first node dies. However, this is not always the case if the redundancy of a wireless network is considered.

2.5.2 Dynamic Source Routing

Dynamic source routing (DSR) [107] uses one or more of a small set of metrics to determine optimal paths. The most common metric used is shortest-hop routing [173]. DSR is a routing protocol for ad hoc networks that uses dynamic source routing of packets between hosts that wish to communicate. Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which to forward the packet. The sender lists this route explicitly in the header of the packet and identifies each forwarding hop by the address of the next node to which it must transmit the packet on its way to the destination host. The sender uses a route discovery algorithm to discover a route to the destination host dynamically. A route maintenance procedure is used to inform the sender of any routing errors. DSR uses no periodic routing advertisement messages, thereby reducing network bandwidth overhead, particularly during periods when little or no significant host movement is taking place. Battery power is also conserved on the mobile hosts – both by not sending the advertisements and by having no need to receive the advertisements. DSR adapts quickly to routing changes when host movement is frequent, yet requires little or no overhead during periods in which hosts move less frequently.

2.5.3 Distance Vector Routing

Distance vector routing (DVR) [97] refers to a decentralised algorithm which applies a distance vector to each route. A distance vector has two components – magnitude and direction. The magnitude represents the cost or distance of the route while the direction identifies the output port which leads to the destination. Switches which use the distance vector algorithm maintain local route tables containing a cost and next hop address for each destination in the network. Messages are forwarded by consulting the route table for the correct destination. Routes are selected by competing for the lowest cost path until a stable state is reached. Occasionally, routes do not stabilise and this causes loops to be formed.

2.5.4 Routing for Maximum System Lifetime

Chang and Tassiulas [33] presented several proposals on ways in which to model the system lifetime of ad hoc networks when the total energy in the network limits lifetime. They proposed routing algorithms with which to select routes and the corresponding power levels so as to maximise the elapsed time until the batteries of the nodes are depleted. Instead of minimising the energy consumed, the focus of the routing algorithms is on maximising the lifetime of the system. In order to achieve this, instead of minimising the absolute power consumed, traffic is routed in such a way that the energy consumption is balanced among the nodes in proportion to the available energy of the nodes. The proposals of Chang and Tassiulas are applicable to networks which are either static or else to networks with slowly changing topology to the extent that there is enough time to balance the traffic optimally during the periods between successive topology changes.

2.5.5 Temporally Ordered Routing Algorithm

Park and Corson [156, 157] presented a new distributed routing protocol for mobile multi-hop wireless networks known as temporally ordered routing algorithm (TORA). TORA is extremely quick in creating and maintaining loop-free multi-path routing to destinations for which routing is required while simultaneously minimising communication overhead. TORA adapts speedily to topological changes and has the ability to detect network partitions and to erase all invalid routes within a finite time. TORA is a link reversal algorithm which routes messages by assigning a height value to each node. Heights impose a temporary order on the set of nodes from highest to lowest. Messages may flow only via nodes in descending height order. Heights are negotiated between immediate neighbours on demand by sending a query message, and, thus, triggering further queries which elevate the height of the originator. Intermediate nodes are assigned heights in descending order toward the destination. A query packet consists of a quintuplet which contains the time of a link request, the identifier address of the originator, the reference height of the originator, and the forwarding node height. A sequence number orders the queries, thus reducing the chances of a loop being created by older requests. At any given time the same quintuplet with node identifier i is associated with each node i. This quintuplet represents the height of the node. Route maintenance is performed when a broken link is detected. The height of a node at the point of failure is lowered, thus causing a link reversal. If it is not possible to reach a destination, then routes toward the destination are flushed from the network.

This protocol is best-suited for use in large, dense mobile networks in which the reaction of the protocol to link failures typically involves a localised single pass of the distributed algorithm only. This capability is unique among protocols which are stable in the face of network partitions, and results in the high degree of adaptability of the protocol. In order to verify convergence in a MANET protocol it is necessary to consider states which are reachable not only in terms of the events of the protocol itself but also in terms of changes of topology [219, 220].

The TORA protocol guarantees that no loops will occur, provides multiple routes and minimal communication overhead even in highly dynamic environments. TORA aims to minimise routing discovery overhead, and, in doing so, prefers instant routes to optimal routes. The protocol supports source-initiated, on-demand routing for networks with a high rate of mobility as well as destination oriented, proactive routing for networks with lesser mobility.

2.5.6 Volcano Routing Scheme

Ganjali and McKeown [78] proposed the volcano routing scheme (VRS) algorithm which routes packets successfully even if the topology changes extremely rapidly. VRS does not need to discover routes, or exchange routing information. It merely balances the load locally between adjacent pairs of nodes. Ganjali and McKeown demonstrated that VRS keeps the system stable for various models of mobility, different communication patterns, and different volumes of flow in the network. The VRS is a potential-based routing scheme. The key principle in potential-based routing is to define K scalar fields on the network one for each destination node. More formally, a single-valued potential, denoted by P_u^f, is associated with any flow f at a given node u. At each node u of the network, packets destined to D_f, the destination of the f-th flow, are routed in the direction (i.e. the next hop) that the potential field decreases the most for flow f.

Normally, the potential function depends on the topology of the network and it is chosen in such a way that each packet is directed toward its destination. In VRS, the potential function is totally different, and is simply based on the number of packets buffered at each node of the network and not on the connectivity of the network. At a given node, u, and for a given flow, f, the potential function, P_u^f, is equal to the number of packets of flow that reside at node u. VRS forwards packets from nodes which have more buffered packets to those nodes which have fewer buffered packets.

The performance of VRS is based on several metrics, namely, packet loss, distribution of queue size, and the length of the path taken by packets. Simulations suggest that, when the network is not highly loaded, the average and maximum queue sizes do not change with the communication range, the number of nodes, and the mobility process [78]. However, when the network becomes highly loaded as a result of reducing the communication range or decreasing the number of nodes, the queue sizes increase. For a fixed number of nodes in the network expansion of the communication range of each node increases the average degree of each node, and this, in turn, enhances the connectivity of the network and reduces the packet loss ratio. If the communication range of each node is fixed then increasing the number of nodes in the network will reduce the packet loss.

2.5.7 Destination Sequenced Distance Vector

Destination sequenced distance vector (DSDV) [160] is a proactive protocol which models mobile computers as routers. The mobile computers cooperate to forward packets as needed to each other. Packets are transmitted between the stations (mobile computers) of the network by using routing tables which are stored at each station of the network. Each routing table, at each of the stations, lists all the available destinations as well as the number of hops to each destination. Each route table entry is tagged with a sequence number. This sequence number is generated by the destination station. In order to maintain the consistency of the routing tables in a dynamically varying topology, each station periodically transmits routing table updates and transmits updates immediately when significant new information becomes available. Older routes will be discarded in favour of newer or cheaper routes (with fewer hop counts). The routing table updates may be sent in two ways:

1. a full dump which sends the full routing table to the neighbours and is able to span several packets, or
2. an incremental update which sends only those entries with a metric (hop count) change since the last update.

Route updates must be sent sufficiently frequently in order to locate all the nodes in the network, thus creating a high communication overhead when the topology is dynamic. Temporary routing loops may be caused by a delay in the propagation of accurate route information. The aim of DSDV is to prevent the propagation of false or out of date information by appending a sequence number to distance vector routing. DSDV has a moderate memory requirement of $O(n)$, where n is the number of nodes. No simulation studies have yet been performed in order to examine the convergence of the algorithm.

2.5.8 Routing for Network Capacity Maximisation in Energy-Constrained Ad Hoc Networks

Kar *et al.* [115] developed a capacity-competitive algorithm known as CMAX. CMAX carries out admission control, i.e. the algorithm may occasionally reject messages that are deemed to be too detrimental to the residual capacity of the network. Network capacity is defined as the total volume of message data that is successfully carried by the network.

Before describing the CMAX algorithm, the following terminology will be defined: The wireless network is modelled as a directed graph, $G = (V, L)$. V is the set of nodes in the network with $n = |V|$ the number of nodes. L is the edge set. There is a directed edge, $(u, w) \in L$, from node u to node w if and only if a single-hop transmission from u to w is possible. Let $E_u > 0$ be the initial energy of node u and let $e_u^c \geq 0$ denote the current energy of node u. For each $(u, w) \in L$ let $E_{uw} > 0$ denote the energy required to do a single-hop transmission from node u to node w. Following a single-hop message transmission from u to w the current energy in node u becomes $e_u^c - E_{uw}$. Note that this single-hop transmission is possible only if $e_u^c \geq E_{uw}$.

To calculate E_{uw}, the most common model for power attenuation is used. In this model signal power attenuates at the rate $\frac{c_a}{d^{h_d}}$, where c_a is a media dependent constant, d is the distance from the signal source, and h_d is another constant between 2 and 4 [170]. Therefore, $E_{uw} = E_{wu} = c * d_{uw}^{h_d}$, where d_{uw} is the Euclidean distance between nodes u and w, and c is a constant.

Let $\kappa(u) = 1 - e_u^c/E_u$ be the fraction of initial energy of node u that has been used so far, where E_u is the initial energy of node u. Let ζ and σ represent two constants. CMAX changes the weight of every edge (u, w) from E_{uw} to $E_{uw} * (\zeta^{\kappa(u)} - 1)$. The shortest source-to-destination path, P, in the resulting graph is determined using the new weight. If the length of this path is more than σ, then the routing request is rejected (admission control); otherwise, path P is accepted as the route. Algorithm 1 summarises the CMAX algorithm.

Algorithm 1 CMAX Algorithm

```
Step 1: {Initialize}
    Eliminate from $G$ every edge $(u, w)$ for which $e_u^c < E_{uw}$;
    Change the weight of every remaining edge $(u, w)$ to $E_{uw} * (\zeta^{\kappa(u)} - 1)$;
Step 2: {Shortest Path}
    Let $P$ be the shortest source-to-destination path in the modified graph.
Step 3: {Wrap Up}
    If no path is found in Step 2, the route is not possible;
    If the length of $P$ is more than $\sigma$, reject the route;
    Otherwise, use $P$ for the route;
```

2.5.9 The Online Maximum Lifetime Heuristic

In order to maximise lifetime, it is necessary to delay the depletion of the energy of a node to a level below that needed to transmit to its closest neighbour for as long as possible [94]. This objective may be attained by using a two-step algorithm, namely, the online maximum lifetime (OML) heuristic algorithm. The OML is used to find a path for each routing request, $r_i = (s_i, d_i)$, where s_i is the source node and d_i is the destination node. In the first step, OML removes from G all edges (u, w) such that $e_u^c < E_{uw}$ as these edges require more energy than is available for a transmit. Let the resulting graph be $G^{'} = (V, L^{'})$. In the next step OML determines the minimum energy path, $P_i^{'}$, from s_i to d_i in the pruned graph $G^{'}$. This may be done by using Dijkstra's shortest path algorithm [175]. In cases in which there is no s_i to d_i path in the pruned graph $G^{'}$ then the routing request r_i will fail. Assume that such a $P_i{'}$ exists. Using $P_i{'}$, OML computes the residual energy, e_u^r as $e_u^r = e_u^c - E_{uw}$, for all edges $(u, w) \in P_i^{'}$. The minimum residual energy, e_{min}^r, is then calculated as

$$e_{min}^r = min\{e_u^r | u \in P_i{'} \text{ AND } u \neq d_i\} \tag{2.3}$$

Let $G^{''} = (V, L^{''})$ be obtained from $G^{'}$ by removing all edges $(u, w) \in L^{'}$ with $e_u^c - E_{uw} < e_{min}^r$. That is, all edges whose use would result in a residual energy below e_{min}^r are pruned from $L^{'}$. This pruning is an attempt to prevent the depletion of energy from nodes that are low on energy.

The second step finds the path to be used to route request r_i. In order to find the path, OML begins with $G^{''}$ as above and assigns weights to each $(u, w) \in L^{''}$. The weight assignment is done to balance the desire to minimise total energy consumption as well as the desire to prevent the depletion of node energy. Let $e_u^m = min\{E_{uw} | (u, w) \in L^{''}\}$ be the energy needed by node u to transmit a message to its nearest neighbour in $G^{''}$. Let $\phi(u, w)$ a function whose use prevent the depletion of the energy of node u, below that needed to transmit to the closest neighbour of u; $\phi(u, w)$ is defined as

$$\phi(u, w) = \begin{cases} 0 & \text{if } e_u^c - E_{uw} > e_u^m \\ c & \quad \text{otherwise} \end{cases} \tag{2.4}$$

where c is a non-negative constant. For each $u \in V$, define

$$\kappa(u) = \frac{e_{min}^r}{e_u^c} \tag{2.5}$$

where $\kappa(u)$ is the fraction of u's initial energy that has been used so far.
The weight, $E_{uw}^{''}$, assigned to edge $(u, w) \in L^{''}$ is

$$E_{uw}^{''} = (E_{uw} + \phi(u, w))(\zeta^{\kappa(u)} - 1) \tag{2.6}$$

where ζ is another non-negative constant.

From equation (2.6), the weighting function, through ϕ, assigns a high weight to edges whose use on a routing path causes a node's residual energy to become low. Also, all edges emanating from a node whose current energy is small relative to e^r_{min} are assigned a high weight because of the $\zeta^{\kappa(u)}$ term. Thus the weighting function discourages the use of edges whose use on a routing path is likely to result in the failure of a future route.

Algorithm 2 gives the OML algorithm to select a path for request r_i.

Algorithm 2 General Procedure of OML

```
Step 1: **Compute** $G''$;
    $G' = (V, L')$ where $L' = L - \{(u,w) | e^c_u < E_{uw}\}$;
    Let $P'_i$ be a shortest path from $s_i$ to $d_i$ in $G'$ using Dijkstra's algorithm;
    If there is no such $P'_i$, the route request fails and the algorithm terminates;
    Compute the minimum residual energy $e^r_{min}$ for all nodes other than $d_i$ on $P'_i$
                using equation (2.3);
    Let $G'' = (V, L'')$ where $L'' = L' - \{(u,w) | e^c_u - E_{uw} < e^r_{min}\}$;
Step 2: **Find route path**
    Compute the weight $E''_{uw}$ for each edge of $L''$ using equation (2.6);
    Let $P''_i$ be a shortest path from $s_i$ to $d_i$ in $G''$;
    Use $P''_i$ to route from $s_i$ to $d_i$;
```

2.5.10 Other Power-Aware Routing Algorithms and Metrics

Energy conservation is a critical issue in wireless networks for node and network lifetime, as the nodes are powered by batteries [211]. This subsection surveys recent routing protocols for wireless networks in which energy awareness is an essential consideration. The most interesting research issue in respect of these power-aware routing protocols consists of ways in which to optimise different metrics.

Singh *et al.* [186] addressed routing of unicast traffic (unicast refers to the sending of information packets to a single destination) with respect to battery power consumption. Their research focused on the design of protocols to reduce energy consumption and to increase the lifetime of each mobile node, thus also increasing network lifetime. This goal may be attained by minimising the energy of mobile nodes, not only during active communication, but also when the mobile nodes are inactive.

Transmission power control and load distribution are two approaches to minimising the active communication energy. Sleep/power-down mode is used to minimise energy during inactivity. In order to minimise the active communication energy, five different metrics are defined from which to study the performance of power-aware routing protocols. These energy-related metrics have been used to determine an energy efficient routing path instead of finding the shortest paths. The metrics are [186]:

- **Energy consumed per packet:**

  This metric is useful in order to provide the minimum power path through which the overall energy consumption for delivering a packet is minimised. Each wireless link is annotated with the link cost in terms of transmission energy over the link. The minimum power path is that path which minimises the sum of the link costs along the path. However, a routing algorithm using this metric may result in unbalanced energy spending among mobile nodes. Nodes that are unfairly burdened in order to support several packet-relaying functions consume more battery energy and run out of energy earlier than other nodes, thus disrupting the overall functionality of the ad hoc network.

- **Time to network partition:**

  This metric maximises the network lifetime. Given a network topology, a minimal set of MNs exists so that the removal of these MNs would cause the network to partition. The traffic in these MNs should be divided in such a way that their power is depleted at equal rates. Given alternative routing paths, this metric favours the selection of that path which will result in the longest network operation time.

- **Variance in node power levels:**

  This metric is based on the premise that all MNs in the network operate at the same priority level, thus ensuring that all mobile nodes are equal and that no single mobile node is penalised or advantaged over any other. All mobile nodes in the network remain powered on for as long as possible.

- **Cost per packet:**

  Metrics other than energy consumed per packet need to be adopted in order to maximise the lifetime of all the mobile nodes in the network. The cost per packet metric creates routes in such a way that mobile nodes with depleted energy reserves do not form part of many routes.

- **Maximum node cost:**

  This metric attempts to minimise the cost experienced by a mobile when routing a packet. By minimising the cost per mobile node significant reductions in the maximum mobile cost are obtained. Also, the maximum node cost metric delays mobile failure, and reduces variance in mobile power level.

In order to conserve energy all metrics need to be minimised, except the time to network partition metric which needs to be maximised. As a result, a minimal cost routing protocol with respect to the five energy efficient metrics is more appropriate instead of a shortest hop routing protocol. Thus, although packets may be routed through longer paths, the paths contain mobile nodes that have greater energy reserves. Also, routing traffic through lightly loaded mobile nodes conserves energy because contention and retransmission are minimised.

The above approach to routing in wireless ad hoc networks requires that every mobile node has knowledge of the locations of every other mobile node and the links between them. This creates significant communication overhead and increases delays. Stojmenovic and Lin [191] addressed this issue by proposing a power, cost, and power-cost global positioning system (GPS) which is based on a localised routing algorithm, where nodes make routing decisions based solely on the location of their neighbours and on the location of the destination.

The power-aware localised routing algorithm attempts to minimise the total power needed to route a message between a source and a destination. The loop-free localised power efficient routing algorithm may be described as follows: The source (or an intermediate node), S, should select one of its neighbours, A, to forward packets toward destination, D, with the goal of reducing the total power needed for the packet transmission. Only those neighbours that are closer to the destination than S are considered. Node A becomes the source and the algorithm proceeds recursively until the destination is reached – if possible.

The cost-aware localised routing algorithm is aimed at extending the worst case lifetime of batteries. The loop-free localised cost aware routing algorithm may be described as follows: The cost, $c(A)$, of a route from S to D via neighbouring node A is the sum of the cost, $f(A)$, of node A and the estimated cost of the route from A to D. If the destination is one of the neighbours of node S, which currently holds the packet, then the packet will be delivered to D. Otherwise, S will select that neighbour, A, which minimises $c(A)$. The algorithm proceeds until the destination is reached, if possible, or until a node has no neighbouring node to the destination other than itself.

Combined power-cost algorithms both minimise the total power needed and maximise remaining battery lifetime.

Stojmenovic and Lin [191] proved empirically that these localised power, cost, and power-cost efficient routing algorithms are loop-free. These routing algorithms achieve very high delivery rates for dense networks, but low delivery rates for sparse networks. Control messages are used to update the positions of all nodes in order to maintain the efficiency of the routing algorithms. These control messages also consume power which means that the most advantageous trade-off for moving nodes must be established.

These routing algorithms were tested on static networks with high connectivity [191]. Power efficient methods tend to select well positioned neighbouring nodes in forwarding messages while cost efficient methods favour nodes with more power remaining.

Shah and Rabaey [181] presented an energy aware routing algorithm for low energy wireless networks. Network survivability was considered as the primary metric. They demonstrated that network lifetimes may be increased up to 40% as against comparable schemes such as directed diffusion routing. Energy aware routing builds per-sink cost fields to direct data delivery. A sender probabilistically picks a receiver to which each packet has to be forwarded. The basic principle is that, in order to occasionally increase the survival of networks, it may be necessary to use sub-optimal paths. This will ensure that the power of the optimal path is not depleted, and that the network degrades as a whole rather than being partitioned. To prevent energy depletion of the optimal path, multiple paths are found between the source and the destination, and each path is assigned a certain probability of being selected, depending on the energy metric. In order for each packet to be sent from the source to the destination one of the paths is randomly chosen, depending on the probabilities. This means that none of the paths is used all the time, thus preventing energy depletion. Also, different paths are tried continuously, hence improving tolerance to nodes moving around in the network.

Energy aware routing is also a reactive routing protocol. It is also a destination-initiated protocol in terms of which the data consumer initiates the route request and subsequently maintains the route.

2.5.11 Power-Aware Routing Algorithms for Networks with Frequent Topological Changes

Routing in mobile ad hoc networks is difficult because the topology may change rapidly [105, 215]. By the time new paths have been discovered the network could change again. In extreme cases packets may circulate endlessly, thus causing the system to become unstable. It is difficult to describe and to constrain mobility for frequently changing networks. There will not be enough time to balance traffic optimally between successive topology changes. For this reason existing routing algorithms are of little use. However, existing algorithms such as the distance vector routing [160] have proved to be effective in situations in which topology changes slowly.

Gafni and Bertsekas [71] considered the problem of maintaining communication between the nodes of a data network and a central station in the presence of frequent topological changes as, for example, in mobile packet radio (PR) networks. They proposed distributed algorithms for generating loop-free routes, and thus forming the basis for the development of contingency routing algorithms.

These distributed algorithms have the following properties:

1. They work on unknown communication topologies.

2. They implement a directed acyclic graph (DAG) over the network topology, where all directed paths lead to the destination.

3. In the event of changes to the topology of the network, a new DAG is created using an iterative method.

In such distributed algorithms each PR is assigned a generalised number (i.e. an element of a suitable totally ordered set). Link directions will always be oriented from higher to lower numbers. This prevents loops from forming and provides reliable secondary routes that may be used for transmitting connectivity information and data to the station when the primary routes fail. When a PR loses all its routes to the station, a reversal process is executed whereby, on the basis of the numbers of its neighbours, the PR selects a number according to the rules of one of the algorithms proposed. This number is broadcast to all the PR's neighbours informing them of any reversals in the direction of communications that affect them.

There is a possibility that some numbers may become too large, for example, if the network becomes disconnected. Accordingly, an error detection scheme is required to ensure that each PR operates on the basis of correct numbers for all its neighbours.

These distributed algorithms – referred to as Gafni-Bertsekas (GB) algorithms – are designed for operation in connected networks. GB algorithms do exhibit instability in parts of the network which have become partitioned from the destination.

Stergaard [190] presented a distributed algorithm for a network with dynamic changing topology. This distributed algorithm is termed the efficient distributed hormone graph gradient (EDHGG). EDHGG provides information on topological distance in communication networks with dynamic changing topologies. From a functional point of view this algorithm is an improved version of the algorithm which was presented by Gafni and Bertsekas [71], with two additional features:

1. From a given node, the length of any directed path to the destination is equal to the shortest undirected path to the destination.
2. Each node knows its topological distance from the destination.

EDHGG is based on topological distance (i.e. the number of hops between two nodes) and is capable of dealing with changing topologies in the connectivity graph using only local information, without global synchronisation. The algorithm generates messages only when the topology of the graph changes.

2.6 Bio-inspired Routing for MANETs

In addition to the classical MANET routing algorithms, the focus of the MANET research community has also been on the application of nature inspired engineering approaches to solve the MANET routing problem [31]. The term bio-inspired has been introduced to demonstrate the strong relation between a particular system or algorithm, which has been proposed to solve a specific problem, and a biological system, which follows a similar procedure or has similar capabilities.

A number of MANET routing protocols have been designed [2], which deal with the extremely dynamic nature of MANET networks [69]. These protocols are mainly inspired by ant colony behaviours, resulting in distributed, self-organizing, and adaptive algorithms. The first ACO routing algorithms developed were the ant based control (ABC) algorithm [178] for circuit-switched telephone networks and AntNet [31] for connectionless IP data networks. ACO implementations for different routing protocols developed since then have based their design on either AntNet [14, 44, 113, 180, 210] or ABC [196].

ABC is a proactive algorithm designed for load balancing in a circuit switched system. ABC is used for call controlling (distributes the calls over multiple switches) and maintains only one routing table whose rows show the destination. Calls between nodes are routed as a function of the pheromone distributions at each intermediate node.

AntNet is an adaptive, distributed, mobile agents-based algorithm which was based on the stigmergic communication found in natural ant colonies. The operation of AntNet is based on two types of agents:

- forward ants who gather information about the state of the network, and
- backward ants who use the collected information to adapt the routing tables of nodes on their path.

The routing tables of each visited node are updated based on trip times. AntNet has been used to simultaneously optimize the *throughput* (delivered bits/sec), *average delay* for data packets (sec), and *network's capacity usage*. However, AntNet has a scalability problem with larger scale networks, because each node has to generate many ants for updating routing tables. In large networks, both overhead and loss of protocol packets grow for distant destinations. Furthermore, ants may carry outdated information for long travel times. Another problem which may arise when implementing AntNet on a real network is the synchronization of the internal clocks of the nodes in the network.

The popularity of MANETs has lead to an increasing need to address MANETs security issues. There are a number of proposals for secure MANETs that are based on artificial immune systems (AISs) [39]. Artificial immune systems (AIS) are algorithms and systems that use the human immune system as inspiration. Sarafijanovic and Boudec [176] presented an AIS security solution that can detect misbehavior in the dynamic source routing (DSR) protocol [107]. Mazhar and Farooq [145] addressed anomaly detection in MANETs using AIS, while Mazhar [144] proposed two security frameworks for securing MANET protocols based on the AIS approach, i.e BeeAIS based on self non-self discrimination from adaptive immune system and BeeAIS-DC based upon the danger theory [39]

.

2.7 Summary

As a result of the power constraint characteristics of MANETs, it is vital to enhance the longevity of the nodes in the network in order to prolong network lifetime. Low power design as applied to the implementation of all protocols, and in particular (in reference to this thesis), the routing protocols, remains one of the most important research areas in terms of MANETs. This chapter provided a survey of existing algorithms at the network layer that address the energy efficiency of MANETs. The next chapter discusses the ant colony optimisation meta-heuristic approach for solving combinatorial optimisation problems. In late chapters, energy aware optimisation algorithms are developed to based on the ant colony optimisation meta-heuristic.

Chapter 3

Combinatorial Optimisation and Ant Colony Optimisation Meta-Heuristic

This chapter reviews basic definitions of concepts related to optimisation. The ant colony meta-heuristic, viewed in the general context of combinatorial optimisation, is then discussed. This is followed by a detailed description as well as a guide to three major ant colony optimisation algorithms, namely, the ant system, the ant colony system, and the MAX-MIN ant system.

3.1 Introduction

Ant colony optimisation (ACO) [54, 56] refers to a recent meta-heuristic approach for solving difficult combinatorial optimisation problems. ACO was inspired by the pheromone trail laying and following behaviour of real ants which use pheromones as a communication medium. With reference to this biological example, ACO is based on the indirect communication of a colony of simple agents, termed artificial ants, which is mediated by (artificial) pheromone trails. The pheromone trails in ACO algorithms serve as distributed, numerical information which the ants use to construct probabilistic solutions to the problem under investigation and which the ants adapt during the algorithm's execution to reflect their search experience.

The ant system (AS) was the first ACO algorithm [56] developed to solve the travelling salesman problem (TSP) [127]. Despite initial encouraging results, AS was not able to compete with state-of-the-art algorithms for the TSP [57]. Nevertheless, AS did play an important role in stimulating further research on algorithmic variants which did obtain far more improved computational performance [53, 56, 73, 195]. Motivated by this success, the ACO meta-heuristic was proposed [54, 56] as a common framework for the existing applications and algorithmic variants. Algorithms which follow the ACO meta-heuristic are generically referred to as ACO algorithms for the remainder of this thesis.

The rest of chapter 3 is organised as follows: Section 3.2 provides an overview of computational complexity, Section 3.3 discusses meta-heuristics in general, Section 3.4 gives an overview of the ant colony optimisation meta-heuristic, while Section 3.5 provides a detailed description of three major ant colony optimisation algorithms.

3.2 Computational Complexity

An important criterion for the classification of problems is the time required by algorithms to find a solution to the given problem. This issue is addressed by the theory of computational complexity [154] and, in particular, by the theory of NP-completeness [80]. The subject of computational complexity theory is dedicated to classifying problems in terms of their degree of complexity.

The time-complexity of an algorithm is measured by a time-complexity function that offers, depending on the size of the problem instance, the maximal run-time needed by the algorithm to solve an instance of that problem. The size of a problem instance reflects the volume of data required to encode an instance in a compact form. An intuitive understanding of the size of an instance of the problem often suffices; for example, the size of TSP [127] instance may be measured by the number of cities to be visited. Typically the time-complexity is detailed in terms of the number of elementary operations required to solve the problem, for example, the number of value assignments or comparisons. Time complexity is formalised by the $O(.)$ notation: Let $f : N \rightarrow N$ and $g : N \rightarrow N$ be two functions. Then, $f(n) = O(g(n))$ if there are positive integers c and n_0 such that for all $n > n_0$, $f(n) \leq c.g(n)$.

An algorithm runs in polynomial time if the runtime is bounded by a polynomial. If it is not possible for the runtime to be bounded by a polynomial then the algorithm is said to be an exponential time algorithm. In complexity theory a distinction is made between efficiently solvable problems (easy problems) and inherently intractable problems (difficult problems). Usually, a problem is considered to be efficiently solvable if it is possible to find a solution in a number of steps bounded by a polynomial of the input size. If the number of steps needed to solve an instance grows super-polynomially then that problem is considered to be inherently intractable.

The theory of NP-completeness distinguishes between two basic classes of problems: the class NP of tractable problems and the NP-complete class of problems. The class NP consists of those problems that may be solved by a nondeterministic polynomial-time algorithm (composed of a guessing stage and a checking stage). The NP-complete class is the class comprising the most difficult problems. If it is possible to solve an NP-complete problem by a polynomial time algorithm, then all problems in NP may be solved in polynomial time.

Many problems of practical and theoretical importance within the fields of artificial intelligence and operations research are of a combinatorial nature [9, 32, 127, 135, 205]. Combinatorial optimisation problems involve finding values for discrete variables such that certain conditions are satisfied. Problems of combinatorial nature may be classified either as optimisation or constraint satisfaction problems. The goal of combinatorial optimisation problems is to find an optimal arrangement, grouping, ordering, or selection of discrete objects.

The TSP [127] is probably the most widely known combinatorial optimisation problem, where the goal is to find a closed tour through a set of cities. Other examples of combinatorial optimisation problems are assignment [32], scheduling [9], and vehicle routing problems [205]. Constraint satisfaction problems (CSP) [135] require a solution to be found that satisfies a given set of constraints. An important special case of the CSP is the well-known satisfiability problem in propositional logic [122]. Other CSP problems are graph coloring [25], temporal and spatial reasoning [5], as well as resource allocation [151].

Combinatorial optimisation problems are often extremely difficult to solve. For example, no algorithm exists for finding the optimal solution to a TSP within polynomial time [153]. Similarly, no algorithm is guaranteed to decide in polynomial time whether a given CSP instance is satisfiable or not. These problems are classified as NP-complete. The class of NP-complete problems has the important distinction that no polynomial time algorithm for any of its members exists to date and, consequently, these problems are considered as inherently intractable from a computational point of view. Thus, in the worst case, any algorithm that endeavours to solve an NP-complete problem will require exponential execution time. In particular, the TSP and the CSP belong to this class and are thus among the most difficult combinatorial problems.

Algorithmic approaches to combinatorial optimisation problems may be classified as either exact or approximate [96]. Exact algorithms are guaranteed to find an optimal solution in finite time by systematically searching the solution space. However, due to the NP-completeness of many combinatorial optimisation problems, the time needed to solve these problems may grow exponentially in the worst case. As a result of the fact that there are several problems for which exact algorithms display poor performance, several types of approximate algorithms have been developed that provide high quality solutions to combinatorial problems in short computation time [62, 108].

Approximate algorithms may be classified into two main types: construction algorithms and local search algorithms [193]. Construction algorithms generate solutions from scratch by adding solution components step by step. The best known examples are greedy construction heuristics [24]. The main advantage of the greedy heuristics is speed: the algorithms are very quick and return reasonably good solutions. However, these solutions are not guaranteed to be optimal with respect to small local changes and solutions may be further improved by a local search.

Local search algorithms start from some given solution and try to find a better solution within an appropriately defined neighbourhood of the current solution. Should a better solution be found, this solution then replaces the current solution and the local search is continued from this point. The most basic local search algorithm, termed iterative improvement [59], applies these steps repeatedly until it is no longer possible to find a better solution in the neighbourhood of the current solution and stops in a local optimum. A disadvantage of this algorithm is that it may stop at poor quality local minima. In order to avoid these disadvantages, many generally applicable extensions of local search algorithms have been proposed [81, 118, 192, 193]. Local search algorithms may be improved by either accepting worse solutions and, thus, allowing the local search to escape from local optima, or by generating good starting solutions for local search algorithms which guide towards better solutions. In the latter case, the experience gained during the run of the algorithm is often used to guide the search in subsequent iterations.

General-purpose techniques have been designed to allow for a further improvement in solution quality. These methods are termed meta-heuristics [152]. A meta-heuristic is defined as a general heuristic method which is used to guide an underlying construction or local search algorithm towards promising regions of the search space containing high quality solutions. In other words, a meta-heuristic may be seen as a general algorithmic framework which may be applied to different combinatorial optimisation problems with relatively few modifications if given some underlying, problem specific method. The problem considered in this thesis is a combinatorial optimisation problem.

The next section describes meta-heuristics in more detail.

3.3 Meta-Heuristics

Meta-heuristics are high-level strategies which guide an underlying, more problem specific heuristic, in enhancing the performance of this heuristic. Meta-heuristics are applicable to a wide range of different combinatorial optimisation problems [20]. The main goal is to avoid the disadvantages of iterative improvement and, in particular, multiple descent by allowing the local search to escape from local optima. Many of the meta-heuristic methods may be interpreted as introducing a bias so that high quality solutions are produced quickly. This bias may take various forms and it may be cast as descent bias (based on the objective function), memory bias (based on previous decisions), or experience bias (based on prior performance) [193]. Many of the meta-heuristic approaches rely on probabilistic decisions which were made during the search [193]. However, the main difference in terms of pure random search is that meta-heuristics do not use randomness blindly, but in an intelligent, biased form.

When applied to combinatorial optimisation problems, the main aim of meta-heuristic algorithms is to provide efficient solution techniques in order to yield high quality solutions within a reasonable amount of time.

The next section discusses one of the most popular and efficient meta-heuristics, namely, the ant colony optimisation (ACO) meta-heuristic. The algorithms proposed in this thesis are using an ACO approach.

3.4 Ant Colony Optimisation Meta-Heuristic

Ant colony optimisation (ACO) is a meta-heuristic proposed by Dorigo [51].

The inspiring source of ACO is the foraging behaviour of real ants. The foraging behaviour [45] enables the ants to find shortest paths between food sources and their nests. While walking from food sources to the nest and vice versa, ants deposit a substance called pheromone on the ground. In this way a pheromone trail is formed. When they decide about a direction to go they choose, in probability, paths marked by strong pheromone concentrations. This basic behaviour is the basis for a cooperation interaction which leads to the emergence of shortest paths.

One of the basic principles of ACO is the use of an algorithmic counterpart of the pheromone trail as a medium for cooperation and communication among a colony of artificial ants. This medium of communication is guided by positive feedback.

Artificial ants posses characteristics of the real ants foraging behaviour. Artificial ants are also enriched with additional capabilities to make them more effective and efficient.

The rest of the section is organised as follows. Subsection 3.4.1 discusses ant algorithms and the foraging behaviour of real ants. Subsection 3.4.2 describes the relation between natural and artificial ants. Subsection 3.4.3 discusses a general framework for the ant colony optimisation meta-heuristic.

3.4.1 Ant Algorithms and Foraging Behaviour of Real Ants

Ant algorithms were first proposed by Dorigo *et al.* [51, 55] as a multi-agent approach to difficult combinatorial optimisation problems, e.g. the travelling salesman problem and the quadratic assignment problem (QAP). There is currently much ongoing research in the scientific community with the aim of extending and applying ant-based algorithms to the many different discrete optimisation problems [21, 38]. Recent applications of ant-based algorithms cover problems such as vehicle routing [27, 205], sequential ordering [66, 75], graph colouring [4], routing in communications networks [98], amongst others.

Ant algorithms were inspired by observations of real ant colonies [45, 141]. Ants are social insects, that is, insects that live in colonies and whose behaviour is directed more to the survival of the colony as a whole than to that of a single individual component of the colony. Social insects have captured the attention of many scientists [54, 58] because of the high structuration level such colonies are capable of, especially when compared to the relative simplicity of the individuals within the colony.

An important and interesting form of behaviour on the parts of ants is their foraging behaviour [56], and, in particular, the way in which ants find the shortest paths between food sources and their nest. Ants are able to detect pheromone and probabilistically choose the next path to follow based on pheromone concentrations. Pheromone trails enable ants to find their way back to the food source (or to the nest). Also, pheromone trails may be used by other ants to find the location of food sources which have been discovered by other ants. It has been demonstrated experimentally [45] that this pheromone trail following behaviour may give rise to the emergence of shortest paths. In other words, when more paths are available from the nest to a food source, a colony of ants may be able to exploit the pheromone trails left by the individual ants in order to discover the shortest path from the nest to the food source and back again.

In order to study the foraging behaviour of ants under controlled conditions, Deneubourg *et al.* [45] set up the binary bridge experiment. In this laboratory experiment, as illustrated in Figure 3.1, the nest was separated from the food source by a bridge with two equally long branches. Initially, both branches were free of any pheromones. After a finite time period, one of the branches was selected, with most of the ants following the path, despite the fact that both branches were of the same length. This selection of one of the branches is as a result of random fluctuations in path selection which cause higher concentrations on the one path.

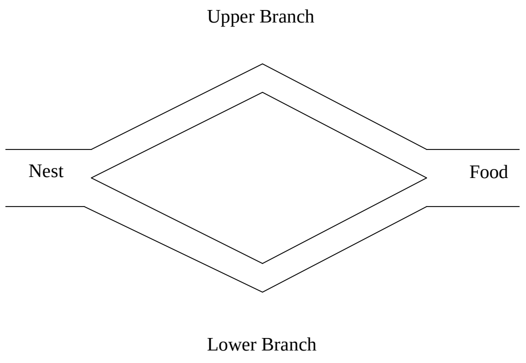

Figure 3.1: Binary bridge experiment

From this experiment, referred to as the binary bridge experiment, a simple, formal model was developed to characterise the path selection process [45, 159]. This probabilistic model makes the assumption that ants deposit the same amount of pheromone. This assumption implies that pheromone evaporation is not taken into account. It is also assumed that the amount of pheromone on a branch is proportional to the number of ants that used the branch in the past. The probability of choosing a branch at a certain time depends on the total amount of pheromone on the branch, which is, in turn, proportional to the number of ants that have used the branch until that time. More precisely, if U_m and L_m respectively denote the numbers of ants that have used the upper and lower branches after m ants have crossed the bridge, then $U_m + L_m = m$. Pasteels *et al.* [159] found empirically that the probability, $P_U(m)$, with which the $(m+1)$-th ant chooses the upper branch is given as

$$P_U(m) = \frac{(U_m + z)^h}{(U_m + z)^h + (L_m + z)^h} \tag{3.1}$$

where z quantifies the degree of attraction of an unexplored branch, and h is the bias to using pheromone deposits in the decision process. The probability, $P_L(m)$, that an ant chooses the lower branch is $P_L(m) = 1 - P_U(m)$.

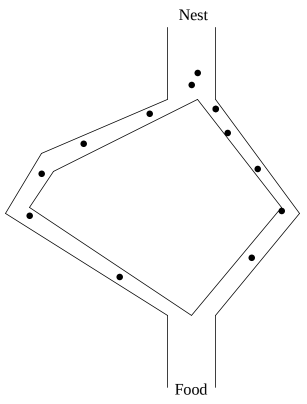

Figure 3.2: Ants start exploring the double bridge

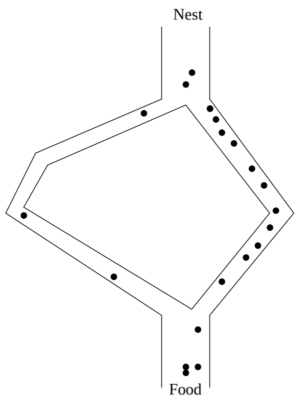

Figure 3.3: Shortest path selection by forager ants

Goss *et al.* [83] extended the binary bridge experiment such that one of the branches of the bridge is longer than the other, as illustrated in Figures 3.2 and 3.3. Dots in these figures indicate ants. Initially, paths are chosen randomly with approximately the same number of ants following both paths. However, over time, more and more ants follow the shorter path, since ants that follow the shortest path return to the nest earlier than those ants on the longer path. The pheromone on the shorter path is, therefore, reinforced sooner than that on the longer path.

The above process represents a form of distributed optimisation mechanism to which each single ant makes a very small contribution. It is interesting to note that, although a single ant is, in principle, capable of constructing a solution (i.e. of finding a path between the nest and food reservoir), it is only the ensemble of ants, that is the ant colony, which presents the shortest path finding behaviour. In a sense, this behaviour may be viewed as an emergent property of the ant colony. It is also interesting to note that ants are able to perform this specific behaviour using a simple form of indirect communication mediated by pheromone laying which is known as stigmergy [84].

Stigmergetic communication is at work via the pheromone that ants deposit on the ground while walking. Correspondingly, artificial ants simulate pheromone laying by modifying appropriate pheromone variables associated to problem states visited, while building solutions to the optimisation problem under consideration. Also, according to the stigmergetic communication model, artificial ants have local access only to these pheromone variables.

Another important aspect of the foraging behaviour of real ants, that is exploited by artificial ants, is the coupling between the autocatalytic (positive feedback) mechanism [55] and the implicit evaluation of solutions. Implicit solution evaluation is the fact that shorter paths (which correspond to lower cost solutions) will be completed earlier than longer paths and therefore, shorter paths will receive pheromone reinforcement more quickly. Implicit solution evaluation coupled with autocatalysis may be very effective: the shorter the path, the sooner the pheromone is deposited by the ants and the greater the number of ants that use the shorter path. As a result of the stronger attraction of ants via the high pheromone concentrations, autocatalysis causes the shortest paths to be favoured. This may cause premature convergence (stagnation) [51, 56], where ants converge too quickly on the same path. Consequently, ants barely explore, and exploit too much. The result of stagnation is that a suboptimal solution may be found. Pheromone trail evaporation and stochastic (based on random choice) state transitions are used to address these autocatalysis drawbacks. Evaporation causes pheromone concentrations on paths to decrease, thereby increasing the probability of selecting longer paths.

3.4.2 Relation Between Natural and Artificial Ants

The artificial ant is a simple, computational agent that builds feasible solutions to the problem being optimised by exploiting the available pheromone trails and heuristic information [52, 54]. Real and artificial ant colonies share a number of characteristics. The most important of these characteristics are summarised as follows [54]:

- Natural and artificial ants both use a colony of individuals that interact and collaborate in order to solve a given task.
- Natural and artificial ants both modify their "environment" through stigmergic communication based on pheromones. In the case of artificial ants the *(artificial) pheromone trail* consists of numeric information which is only locally available.
- Natural and artificial ants both share a common task, i.e. the search for the shortest path from an origin (the ant nest) to a certain goal state (the food).
- As do real ants, artificial ants build solutions iteratively by applying a local stochastic transition policy in order to move between adjacent states.

However, these characteristics alone do not allow for the development of efficient algorithms for difficult combinatorial problems. Artificial ants live in a discrete world and possess additional capabilities:

- Artificial ants make use of heuristic information in the stochastic transition policy used to select next links in the path being constructed.
- Artificial ants possess a memory that stores the path followed.
- The amount of pheromone deposited by artificial ants is a function of the quality of the solution discovered.
- Pheromone evaporation in ACO algorithms is different to the pheromone evaporation in nature, since the inclusion of an evaporation mechanism is a key question to avoid the algorithm becoming stuck in local optima.

3.4.3 General Framework for Ant Colony Optimisation Meta-Heuristic

Several ACO algorithms exist, which are collectively referred to as ant colony optimisation meta-heuristics. These algorithms model and exploit the behaviour of ants to be applied to solve graph-based, NP-hard, combinatorial optimisation problems.

The main characteristics of ACO are:

1. **Autocatalytic**: The system uses positive feedback as a way of auto-reinforcement. This process is iterative, and soon almost all ants will be choosing the best path.

2. **Distributed computation**: A number of agents are searching for the best solution.

3. **Constructive heuristics**: A "greedy force" which balances the decisions of ants in terms of shorter paths or paths with more pheromone.

The problem to solve with an ACO algorithm is represented by a graph, $G = (V, L)$, where V denotes the set of nodes and L is a matrix which represents the connections between nodes. The graph has $N_G = |V|$ nodes.

ACO algorithms are essentially construction algorithms: For each algorithm iteration, every artificial ant constructs a solution to the problem by travelling on a construction graph. Each link, $(i, j) \in L$, of the graph, which represents a possible step that an ant may make, possesses two kinds of information in order to guide ant movements:

- **Heuristic information** which measures the heuristic preference of moving from node i to node j. This heuristic information is denoted by η_{ij}.

- **Artificial pheromone trail information** which measures the "learned desirability" of the movement and mimics the real pheromone that natural ants deposit. This pheromone information is denoted by τ_{ij}.

The ACO meta-heuristic framework consists of three parts, namely, Ants_generation-_and_activity(), EvaporatePheromone(), and DaemonActions() (refer to Algorithm 3).

Ants_generation_and_activity() After initialising each link of the problem graph with a very small amount of pheromone and defining the starting node of each ant, each ant iteratively constructs a solution during each iteration.

The transition rule assigns a probability to each possible link leading from the current node. This probability expresses the desirability of each link and is calculated as a function of pheromone concentrations on the link and heuristic information. While moving, an ant keeps in memory the partial solution it has built in terms of the path the ant was walking on the construction path. As ants construct paths (i.e. solutions to the optimisation problem), a pheromone update rule is executed to update pheromone concentrations on links. The exact way in which the pheromone update rule is implemented differs based on the specific ACO algorithm used. Depending on the algorithm used, all ants can update pheromone on all links along the traversed path (referred to as the online step-by-step pheromone trail update). Once every ant has generated a solution, the ant can deposit an amount of pheromone which is a function of the quality of the ant's solution (referred to as the global or online delayed pheromone trail update).

EvaporatePheromone() Pheromone evaporation is used to increase exploration of alternative paths, thereby reducing the chance of premature stagnation. For each link, (i, j), pheromone evaporation is implemented using

$$\tau_{ij}(t) = (1 - \rho)\tau_{ij}(t), \quad \forall (i, j) \in L \tag{3.2}$$

where $\rho \in [0, 1]$ specifies the rate at which pheromones evaporate.

DaemonActions() Since not all ACO algorithms make use of daemon actions, the daemon actions are indicated as optional. Daemon actions can be used to implement centralised actions which cannot be performed by single ants. An example daemon action is the use of a local optimisation procedure applied to the solutions built by the ants. Pheromone updates performed by the daemon are called *o*ffline pheromone updates.

A generic ACO meta-heuristic algorithm is summarised in Algorithm 3. The high-level description in Algorithm 3 consists of the above three main components of ACO algorithms, namely, ant generation and activation, pheromone evaporation, and daemon actions gathered in the schedule_activities construct. The schedule_activities construct does not specify how these three activities are scheduled and synchronised. This is up to the algorithm design.

Action *ants_generation_and_activity* creates a new ant and activates that ant. The procedure *new_active_ant* is called for each ant to construct a path from source to destination node. The procedure refers to an ant routing table which is maintained for each ant. Each entry in an ant routing table is a value obtained by a functional composition of the pheromone and heuristic values of the link of the corresponding nodes. The abstract specification of *new_active_ant()* allows for the implementation of any ACO instances.

Algorithm 3 Generic ACO Meta-Heuristic

```
procedure ACO_Meta_Heuristic()
while termination_criterion_not_satisfied do
    begin schedule_activities
      ants_generation_and_activity();
      pheromone_evaporation();
      daemon_actions(); {optional}
    end schedule_activities
end while
end procedure

procedure ants_generation_and_activity()
while available_resources do
    schedule_the_creation_of_a_new_ant();
    new_active_ant();
end while
end procedure

procedure new_active_ant() {ant lifecycle}
initialise_ant();
$M$= update_private_ant_memory();
$i$ = starting_node;
$D$ = target_node;
$T = \emptyset$; {ant solution}
while i $\neq$ D do
  Build set of neighbours for $i$; {using $M$}
  $A$ = read_local_ant_routing_table(); {problem specific heuristic information and pheromone information}
  Assign probability $p_{ij}$ to each neighbour node $j$ based on $A, M$, and problem_constraints;
  Select next node $j$ according to transition policy based on $p_{ij}$ and problem_constraints;
  $T = T \cup j$;
  $i = j$;
  if online_step_by_step_pheromone_update then
    deposit_pheromone_on_the_visited_arc();
    update_ant_routing_table();
  end if
  $M$= update_private_ant_memory();
end while
if online_delayed_pheromone_update then
  evaluate_solution();
  deposit_pheromone_on_all_visited_arcs();
  update_ant_routing_table();
end if
die(); {when ant finish building a solution and depositing pheromone the ant is deleted from the system}
end procedure
```

3.5 Ant Colony Optimisation Algorithms

The term ACO meta-heuristic (ACO-MH) is used to represent all instances of ACO variants [52, 54]. ACO refers to an algorithm which is a specific instance of the generic algorithm presented in Algorithm 3.

The ACO-MH comprises a wide class of algorithms that may manifest very different implementations. However, the ACO-MH is not sufficiently general to cover the full family of ant algorithms. The fast ant system [198, 199] is an example of an ant algorithm which is not covered by the ACO-MH. The fast ant system is a construction algorithm based on the operation of a single ant without using explicit pheromone evaporation.

The next subsections discuss three different instances of the ACO-MH, namely, the ant system [56], the ant colony system [53], and the MAX-MIN ant system [194]. The algorithms presented in this thesis are adaptations of these three ACO algorithms. The complete algorithm for each of these ACO instances is given. The three algorithms vary in their transition rules and their pheromone trail update rules.

3.5.1 Ant System

The ant system (AS), developed by Dorigo *et al.* [56], was the first algorithm inspired by the behaviour of real ants. Even though AS was developed to solve the travelling salesman problem (TSP) [74], it can be applied to a more general class of combinatorial optimisation problems [27, 67, 75, 140]. The AS deviates from the natural metaphor in that artificial ants possess a degree of memory. Memory is achieved by maintaining a tabu list of already visited cities. This is used to prevent revisiting cities. The tabu list is selected using the ant's private memory, M (refer to the generic ACO meta-heuristic in Algorithm 3).

In AS, at each construction step an ant k chooses to go to the next node with a probability that is computed as:

$$p_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha(\eta_{ij}(t))^\beta}{\sum_{u\in N_i^k(t)}(\tau_{iu}(t))^\alpha(\eta_{iu}(t))^\beta} & \text{if} \quad j \in N_i^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where $N_i^k(t)$ is the set of feasible nodes for ant k which is currently located at node i; $N_i^k(t) = V(i)\backslash TL^k(t)$, where $TL^k(t)$ is the tabu list for ant k, storing the ant's partial tour and $V(i)$ is the set of neighbouring nodes for ant k which is currently located at node i; τ_{ij} represents the *a posteriori effectiveness* of the move from node i to node j, as expressed in the pheromone intensity of the corresponding link, (i, j); η_{ij} represents the *a priori effectiveness* of the move from i to j (i.e. the attractiveness, or desirability of the move) computed using a specific heuristic; α is a positive constant used to amplify the influence of pheromone concentrations; β is an adjustable parameter that controls the relative influence of the attractiveness, $\eta_{ij}(t)$, of node j. The heuristic information, η_{ij}, adds an explicit bias towards the most attractive solutions and is, therefore, a problem dependent function.

The transition probability as given in equation (3.3) balances pheromone intensity, τ_{ij}, and heuristic information, η_{ij}. The best balance between exploration and exploitation is achieved through proper selection of the parameters α and β. Therefore, the transition probability is a trade-off between heuristic (which indicates that close nodes should be chosen with high probability, thus implementing a greedy constructive heuristic) and pheromone trail intensity at time t (which indicates that links with a high traffic load are more desirable than links with a low traffic load, thus implementing the autocatalytic process). The ant's decision table, A_i, for node i (refer to Algorithm 3), is obtained by the composition of the local pheromone trail values with the local heuristic values as in equation (3.3).

Pheromone evaporation is implemented as given in equation (3.2). Once all ants have constructed a solution, pheromones are laid on the links, and the amount of pheromone on the trails is calculated using

$$\tau_{ij}(t+1) \leftarrow \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{3.4}$$

where $\tau_{ij}(t)$ represents the pheromone concentration associated with link (i, j), and $\Delta_{ij}(t)$ is the total amount of pheromone deposited by all ants on link (i, j), defined as

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t) \tag{3.5}$$

with

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^k(t)} & \text{if} \quad \text{link } (i,j) \text{ occurs in path described by } TL^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where Q is a parameter that specifies the amount of pheromones ant k has to distribute throughout its trail, $L^k(t)$ is the tour length of ant k or the quality of the complete path constructed by the ant k, and n_k is the number of ants. Ants with a minimum tour length deposit a greater amount of pheromone on the links that form part of their trails, while longer tours receive smaller pheromone deposits.

The AS algorithm is summarised in Algorithm 4.

3.5.2 Ant Colony System Optimisation

This section describes the ant colony system (ACS) – an ACO meta-heuristic introduced by Dorigo and Gambardella [53] in order to improve the performance of AS. ACS differs from AS in four aspects:

1. a different transition rule is used,
2. a different pheromone update rule is defined,
3. local pheromone updates are introduced, and
4. candidate lists are used to provide additional local heuristic information.

Each of these differences is discussed next.

ACS transition rule

The ACS transition rule, also referred to as a *pseudo-random-proportional* action rule [74], was developed explicitly to balance the exploration and the exploitation abilities of the algorithm. Ant k, currently located at node i, selects the next node, j, to move to using the rule,

$$j = \begin{cases} Arg\ Max_{u \in N_i^k(t)}\{\tau_{iu}(t)\eta_{iu}^{\beta}(t)\} & \text{if } r \leq r_0, \\ J & \text{otherwise} \end{cases} \tag{3.7}$$

Algorithm 4 General Procedure of Ant System Algorithm

```
$t = 0$;
$TL^{best}(t) = \emptyset$; {shortest path}
$L^{best}(t) = 0$; {tour length of the shortest path}
Initialise all parameters, i.e. $\alpha$, $\beta$, $\rho$, $Q$, $n_k$, $\tau_0$;
Place all ants, $k = 1, ..., n_k$;
for each link $(i, j)$ do
  $\tau_{ij}(t) = \tau_0$; {Initialise pheromones to the small value $\tau_0$}
  $\eta_{ij}(t) = \frac{1}{d_{ij}}$; { $d_{ij}$ represents the distance between the nodes $i$ and $j$ }
end for
repeat
  for all ant $k = 1, ..., n_k$ do
    $TL^k(t)$ = starting_node of ant $k$;
    $i$ = starting_node of ant $k$;
    repeat
      From current node $i$, select next node $j$ with probability as defined in equation
      (3.3);
      Add $j$ to the ordered list $TL^k(t)$;
      $i = j$;
    until full path has been constructed
    $L^k(t)$=length of the tour described by $TL^k(t)$;
    if $L^k(t) < L^{best}(t)$ then
      $TL^{best}(t) = TL^k(t)$;
    end if
  end for
  for each link $(i, j)$ do
    Apply evaporation using equation (3.2);
    Calculate $\Delta\tau_{ij}(t)$ using equation (3.5);
    Update pheromone using equation (3.4);
  end for
  for each link $(i, j)$ do
    $\tau_{ij}(t + 1) = \tau_{ij}(t)$;
  end for
  $t = t + 1$;
until Stopping condition is true
Return $TL^{best}(t)$;
```

where r is a real random variable uniformly distributed in the interval $[0, 1]$, r_0 is a tuneable parameter $(0 \leq r_0 \leq 1)$, and $J \in N_i^k(t)$ is a node that is randomly selected according to probability,

$$p_{iJ}^k(t) = \frac{\tau_{iJ}(t)\eta_{iJ}^{\beta}(t)}{\sum_{u \in N_i^k(t)} \tau_{iu}(t)\eta_{iu}^{\beta}(t)} \tag{3.8}$$

where $N_i^k(t)$ is a set of valid nodes to visit and β is an adjustable parameter that controls the relative influence of the attractiveness, $\eta_{ij}(t)$, of node j. The ACS transition rule uses $\alpha = 1$ and may be omitted. If $\beta = 0$, only pheromone amplification is at work, and this will, in turn, lead to the rapid selection of tours which may prove to be far from optimal. When $r \leq r_0$ exploitation is facilitated. The selection of the next node is then heavily influenced by the distances between nodes and existing pheromone concentrations by choosing the best local compromise between distance and pheromone concentration. When $r > r_0$ exploration is favoured. Greater emphasis may be placed on exploitation instead of exploration by increasing the value of r_0.

Pheromone global update

The ant that performed the best tour is allowed to update the concentrations of pheromone on the corresponding links globally. Gambardella and Dorigo [53] implemented two methods of selecting the best tour (best path), $TL^{bt}(t)$, namely

- **iteration-best** in terms of which $TL^{bt}(t)$ represents the best path found during the current iteration, t, denoted as $TL^{ib}(t)$, and
- **global-best** in terms of which $TL^{bt}(t)$ represents the best path globally which has been found from the beginning of the trial, denoted as $TL^{gb}(t)$.

The pheromone concentration, $\tau_{ij}(t)$, is then modified by an amount, $\Delta\tau_{ij}(t)$, as follows:

$$\Delta\tau_{ij}(t) = \begin{cases} \frac{1}{L^{bt}(t)} & \text{if } (i,j) \in TL^{bt}(t) \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

where $L^{bt}(t)$ represents the length of the tour described by $TL^{bt}(t)$.

Only the best tour is reinforced through the global update as follows (including pheromone evaporation),

$$\tau_{ij}(t+1) \leftarrow (1-\rho_g)\tau_{ij}(t) + \rho_g \Delta\tau_{ij}(t) \tag{3.10}$$

where $\rho_g (0 \leq \rho_g \leq 1)$ is a parameter governing global pheromone decay.

The ACS global update rule results in the search being more directed by encouraging ants to search in the vicinity of the best solution found thus far. This strategy favours exploitation and is applied after all ants have constructed a solution.

Local update

When, while performing a tour, ant k is on node i and selects node $j \in N_i^k(t)$ as the next node to hop to, the pheromone concentration of (i, j) is immediately reinforced by a fixed amount τ_0. The pheromone decays simultaneously using

$$\tau_{ij}(t) \leftarrow (1-\rho_l)\tau_{ij}(t) + \rho_l \tau_0 \tag{3.11}$$

where $\rho_l (0 \leq \rho_l \leq 1)$ is a parameter which governs local pheromone decay, and τ_0 is a small positive constant. Experimental results demonstrated that $\tau_0 = 1/(n_G L_{nn})$ provided good results [53]; n_G is the number of nodes in graph G, and L_{nn} is the tour length produced by the nearest neighbour heuristic [172].

Candidate list

A candidate list is a list of preferred nodes to be visited from a given node. When an ant is in node i, instead of examining all the unvisited neighbours of i, the ant chooses the node to move to among those in the candidate list. Only if no candidate list node has unvisited status then other nodes are examined. The candidate list contains $n_c < N_i^k(t)$ nodes ordered by increasing cost, and the list is scanned sequentially and according to the ant tabu list to avoid already visited cities.

ACS is summarised in Algorithm 5.

Algorithm 5 General Procedure of Ant Colony System Algorithm

```
$t = 0$; Initialise parameters $\beta$, $\rho_g$, $\rho_l$, $r_0$, $n_k$, $\tau_0$;
Place all ants, $k = 1, ..., n_k$;
for each link $(i, j)$ do
    $\tau_{ij}(t) = \tau_0$;
    $\eta_{ij}(t) = \frac{1}{d_{ij}}$; { $d_{ij}$ represents the distance between the nodes $i$ and $j$ }
end for
$TL^{gb}(t) = \emptyset$;
$L^{gb}(t) = 0$;
repeat
    for all ants $k = 1, ..., n_k$ do
        $TL^k(t)$ = starting_node of ant $k$;
        $i$ = starting_node of ant $k$;
        repeat
            From current node $i$, select next node $j \in N_i^k(t)$ using equations (3.7) and (3.8);
            Add $j$ to the ordered list $TL^k(t)$;
            $i = j$;
            Apply local update using equation (3.11);
        until full path has been constructed;
        $L^k(t)$=length of the tour described by $TL^k(t)$;
    end for
    for $k = 1, ..., n_k$ do
        if $L^k(t) < L^{gb}(t)$ then
            $TL^{gb}(t) = TL^k(t)$;
            $L^{gb}(t) = L^k(t)$;
        end if
    end for
    for each link $(i, j) \in TL^{gb}(t)$ do
        Apply global update using equation (3.10);
    end for
    for each link $(i, j)$ do
        $\tau_{ij}(t + 1) = \tau_{ij}(t)$;
    end for
    $TL^{gb}(t + 1) = TL^{gb}(t)$;
    $L^{gb}(t + 1) = L^{gb}(t)$;
    $t = t + 1$;
until Stopping condition is true;
Return $TL^{gb}(t)$ as the solution;
```

3.5.3 MAX-MIN Ant System

St*ü*tzle and Hoos [194] introduced the max-min ant system (MMAS) in order to address the premature stagnation problem of AS. The main difference between MMAS and AS is that pheromone intensities are restricted within given intervals, $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$ for all τ_{ij}, where τ_{min} and τ_{max} are two tunable parameters. By choosing appropriate values for τ_{min} and τ_{max} stagnation is reduced. Additional differences with AS as stated in [194] are:

- After each iteration only the best ant is allowed to deposit pheromones following the ACS model.
- Trails are initialised with the highest possible volume of pheromone τ_{max} to incite high exploration of trails at the commencement of the search process.

In order to limit the stagnation of the search a direct influence on the pheromone limits is exerted by restricting the allowed range of the possible pheromone strength. Pheromone strength is bounded by an upper and lower limit (thus MAX-MIN). A range $[\tau_{min}, \tau_{max}]$ is imposed to all τ_{ij} components. A bound on the upper level is given as [195]

$$\tau_{max}(t) = \left(\frac{1}{1-\rho}\right)\frac{1}{L^{ib}(t)} \tag{3.12}$$

where ρ is the evaporation factor and $L^{ib}(t)$ is the cost of the iteration-best path at iteration t (alternatively the global-best path). The upper level is, therefore, time-dependent. Before the first iteration the pheromone strength on all links is set to a certain high value to ensure that, after the first iteration, the pheromones correspond to τ_{max}. The lower bound may be calculated as [195]

$$\tau_{min}(t) = \frac{\tau_{max}(t)(1 - p_{best}^{1/(n_G-1)})}{(\frac{n_G}{2} - 1)p_{best}^{1/(n_G-1)}} \tag{3.13}$$

where n_G denotes the number of nodes in graph, G, and p_{best} is the probability at which the best solution is constructed. The term, $\frac{n_G}{2} - 1$, represents the average number of nodes from which an ant has to choose.

The transition rule is the same as that of AS while the global update is the same as that of ACS. As a result of the evaporation coefficient, all pheromones are decreased at the end of each iteration while the pheromones on the links pertaining to the best solution are increased. The pheromone upper bound helps to avoid search stagnation by preventing only one trail from accumulating high values of pheromone. By limiting the amount of pheromone in a given trail, the probability of an ant choosing that particular trail is also limited.

The max-min algorithm is summarised in Algorithm 6. The iteration-best solution is used to update pheromone concentrations.

3.6 Summary

Ant colony optimisation (ACO) has been used for the last decade to design effective algorithms to solve combinatorial optimisation problems.

This chapter provided an overview of the basic principles of combinatorial problems. ACO meta-heuristics for solving combinatorial optimisation problems were discussed, including the most popular ACO algorithms. The ACO algorithms discussed in this chapter are stochastic, population-based search algorithms. Optimal solutions are incrementally constructed by a number of agents working cooperatively by exchanging information obtained about the search space.

The ACO algorithms discussed in this chapter were developed to solve

- static and well defined combinatorial optimisation problems; that is, problems for which all the necessary information is both available and does not change during the optimisation process, and
- single-objective optimisation problems.

In order to apply ACO algorithms to dynamic problems, the algorithms have to be adapted to ensure good exploration abilities during the entire optimisation process. Similarly, ACO algorithms have to be adapted to solve multi-objective optimisation problems (MOPs).

Chapter 4 shows how ACO algorithms can be adapted to solve MOPs, while Chapter 5 explains how ACO algorithms can be adapted to solve dynamic optimisation problems.

Algorithm 6 General Procedure of Max-Min Ant System Algorithm

```
Initialise parameters $\alpha$, $\beta$, $\rho$, $n_k$, $p_{best}$, $\tau_{min}$, $\tau_{max}$;
$t = 0$, $\tau_{max}(0) = \tau_{max}$, $\tau_{min}(0) = \tau_{min}$;
Place all ants, $k = 1, ..., n_k$;
for each link $(i, j)$ do
  $\tau_{ij}(t) = \tau_{max}(0)$;
  $\eta_{ij}(t) = \frac{1}{d_{ij}}$; { $d_{ij}$ represents the distance between the nodes $i$ and $j$ }
end for
repeat
  $TL^{ib}(t) = \emptyset$; {$TL^{ib}(t)$ is the iteration-best solution}
  $L^{ib}(t) = 0$;
  for all ant $k = 1, ..., n_k$ do
    $TL^k(t)$ = starting_node of ant $k$;
    $i$ = starting_node of ant $k$;
    repeat
      From current node $i$, select next node $j$ with probability as defined in equation (3.3);
      Add $j$ to the ordered list $TL^k(t)$;
      $i = j$;
    until full path has been constructed;
    $L^k(t)$=length of the tour described by $TL^k(t)$;
    if k=1 OR $L^k(t) < L^{ib}(t)$ then
      $TL^{ib}(t) = TL^k(t)$;
      $L^{ib}(t)$=length of the tour described by $TL^{ib}(t)$;
    end if
  end for
  for each link $(i, j) \in TL^{ib}(t)$ do
    Apply global update using equation (3.10);
  end for
  for each link $(i, j)$ do
    Constrict $\tau_{ij}(t)$ to be in $[t_{min}(t), \tau_{max}(t)]$;
  end for
  $TL^{ib}(t + 1) = TL^{ib}(t)$;
  $L^{ib}(t + 1) = L^{ib}(t)$;
  $t = t + 1$;
  Update $\tau_{max}(t)$ using equation (3.12 );
  Update $\tau_{min}(t)$ using equation (3.13 );
until Stopping condition is true;
Return $TL^{ib}(t)$ as the solution;
```