Georg-August-Universität
Göttingen
Institute of Computer Science

A Multi-objective Ant Colony Optimisation-based Routing Approach for Wireless Sensor Networks Incorporating Trust

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"
der Georg-August-Universität Göttingen

vorgelegt von
Ansgar Kellner
aus Hannover

Göttingen 2012

Referent:	Prof. Dr. Dieter Hogrefe
Korreferent:	Prof. Dr. Xiaoming Fu
Tag der mündlichen Prüfung:	21. Juni 2012

Abstract

In the near future, Wireless Sensor Networks (WSNs) are expected to play an important role for sensing applications, in the civilian as well as in the military sector. WSNs are autonomous, distributed, self-organised networks consisting of multiple sensor nodes. Usually, the limited radio range of the nodes, arising from energy constrains, is overcome by the cooperation of nodes.

As the Combinatorial Optimisation Problem (COP) of routing is computationally hard, often approximation algorithms are preferred, which are capable of finding near optimal solutions within polynomial time. A simple but robust way of solving the routing COP is the application of Ant Colony Optimisation (ACO)-based routing algorithms. When multiple (conflicting) objectives should be considered, ACO algorithms can be extended to Multi-objective Ant Colony Optimisation (MOACO) algorithms that are capable of considering multiple objectives at the same time within the optimisation process.

Normally, the routing in WSNs is susceptible to adversaries due to their deployment in unattended or in hostile environments. Particularly, attacks from compromised nodes (*insider attacks*) are a severe problem in WSNs. As insider attacks cannot be alleviated by classical security measures, often soft security measures (trust and reputation) are applied to mitigate the impact of these attacks.

In this thesis, the idea of using trust as security measure against insider attacks is seized and interweaved with an MOACO-based routing approach. The Multi-objective Ant Colony Optimisation Routing Framework for WSNs (MARFWSN) is developed, a routing framework for WSNs that provides an interface for the docking of MOACO-based algorithms that can be used for the routing. Different MOACO-based algorithms – including `ASMOACO`, `MMASMOACO`, `ACSMOACO` and `SRMOACO` – are implemented and tested, considering multiple objectives incorporating trust. In the first step, several pre-experiments are conducted to find the best input parameters for the MOACO-based WSN routing algorithms. Subsequently, further experiments aim for comparing the MOACO-based routing algorithms to the well-known Dynamic Source Routing (DSR) protocol regarding several performance criteria. Different topologies and traffic patterns are taken into account to obtain heterogeneous results, depending on the specific application area of the tested routing algorithms.

The simulation results show that the MOACO-based routing algorithms perform similarly compared to the DSR-based protocol, implicating the possibility for their utilisation in WSNs. The additional benefit of the mitigation of insider attacks for the MOACO-based algorithms, though result in a slightly higher routing overhead and end-to-end delay.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

ACS Ant Colony System 63

ACO Ant Colony Optimisation 9

AIS Artificial Immune System 52

ANN Artificial Neural Network 52

AODV Ad hoc On-Demand Distance Vector Routing 90

AS Ant System 63

AS$_{elite}$ Elitist Ant System 63

AS$_{rank}$ Rank-based Ant System 63

ASFA Achievement Scalarizing Function Approach 49

ATM Asynchronous Transfer Mode 103

BIA Biologically-inspired Algorithm 51

CAGR Compound Annual Growth Rate 1

CBR Constant Bit Rate 166

COP Combinatorial Optimisation Problem 9

CSMA/CA Carrier Sense Multiple Access/Collision Avoidance ... 170

DDR Distinct Packet Delivery Ratio 178

DM Decision Maker 44

DSR Dynamic Source Routing 90

DoS Denial of Service 21

EA Evolutionary Algorithm 52

EAS Elitist Ant System 65

e-commerce electronic commerce 23

eCM ϵ-Constraint Method 49

EP Evolutionary Programming 52

ES Evolution Strategies 52

GA Genetic Algorithm 53

GBAS Graph–based Ant System 73

GP	Genetic Programming . . . 52
GPS	Global Positioning System . . . 13
GSA	Gravitational Search Algorithm . . . 53
IEEE	Institute of Electrical and Electronics Engineers . . . 15
ILS	Iterated Local Search . . . 51
IP	Internet Protocol . . . 150
IWD	Intelligent Water Drops . . . 53
LAN	Local Area Network . . . 2
MAC	Media Access Control . . . 21
MANET	Mobile Ad Hoc Network . . . 10
MARFWSN	Multi-objective Ant Colony Optimisation Routing Framework for WSNs . . . 117
MMAS	Max-Min Ant System . . . 63
MCDM	Multi-Criteria Decision Making . . . 46
MCOP	Multi-objective Combinatorial Optimisation Problem . . . 9
MOACO	Multi-objective Ant Colony Optimisation . . . 9
MPI	Message Passing Interface . . . 102
NIC	Network Interface Card . . . 125
OLSR	Optimised Link State Routing Protocol . . . 89
PDR	Packet Delivery Ratio . . . 89
PSO	Particle Swarm Optimisation . . . 53
P2P	Peer-to-Peer . . . 23
QAP	Quadratic Assignment Problem . . . 70
QoS	Quality of Service . . . 15
RBE	Reliable Best-Effort Data Traffic . . . 167
RFD	River Formation Dynamics . . . 53
RTM	Real-time Multimedia Data Traffic . . . 167
SA	Simulated Annealing . . . 39
SDS	Stochastic Diffusion Search . . . 53

SI	Swarm Intelligence . . . 53
SIA	Swarm Intelligence Algorithm . . . 53
TnR	Trust and Reputation . . . 22
TRS	Trust and Reputation System . . . 23
TS	Tabu Search . . . 39
TSP	Travelling Salesman Problem . . . 35
UCTP	University Course Timetabling Problem . . . 70
VoIP	Voice over IP . . . 15
WLAN	Wireless Local Area Network
WMM	Weighted Metrics Model . . . 48
WPM	Weighted Product Model . . . 47
WSM	Weighted Sum Model . . . 46
WSN	Wireless Sensor Network . . . 9
WTM	Weighted Tchebycheff Metric . . . 49
PDA	Personal Digital Assistant . . . 14

1

Introduction

> Technology has become as ubiquitous as the air we breathe, so we are no longer conscious of its presence.
>
> Godfrey Reggio (1940 -)

Wireless Sensor Networks (WSNs) are expected to play an important role in the upcoming era of pervasive computing, i.e. the integration of information processing into every object and activity. Due to advances in wireless communication and micro-electronics, the development of small low-cost, low-power wireless communication devices, so called sensors, became focus of attention. These sensors are small-sized lightweight devices that are capable of interacting with their environment by sensing or controlling physical parameters, performing simple computations and communicating with other nodes typically via radio frequency channel.

The main goal of sensor networks is to detect events or phenomena, collect data about those events, process and aggregate the data and finally transmit the data to a final destination such as a database server or an interested end-user. To achieve this goal, sensor nodes collaborate by forming a highly distributed network for sensing tasks that a single node could not do. The sensor nodes can be deployed either in a random fashion, e.g. the sensors are dropped from an air-plane, or placed manually, e.g. fire alarm sensors in a building.

The interest in sensor applications and the use of sensors is increasing so that the market forecast of BCC Research from March 2011 estimated the global market of sensors at \$56.3 billion in 2010, increasing to \$62.8 billion in 2011 and already about \$91.5 billion by 2016 reaching a Compound Annual Growth Rate (CAGR) of 7.8 % [14].

Sensor networks have certain characteristics that distinguish them from other (wireless) networks. The fundamental characteristics of WSNs are the resource constraints, the self-organisation capabilities and the use of

the wireless broadcast medium for the communication. The sensor nodes are strictly constrained in their use of resources in terms of low computational power, small memory, limited transmission range and severe energy constraints. Although the sensors in WSNs are often statically deployed, i.e. without later movement, the topology of a WSN needs to be adapted dynamically to changing environment. This is mainly because of failing nodes that are running out of energy, getting destroyed in hostile environments or being temporary unable to communicate due to interferences. The use of the wireless broadcast medium as communication channel leads, on the one hand, to an unreliable communication that is prone to interferences and, on the other hand, to security issues because every node in the radio range can eavesdrop the network traffic. While the single characteristics of WSNs might not be completely new, the combination of small sensor nodes that form a self-configuring network to fulfil a certain sensing task under strong resource constraints can be seen as rather novel and challenging.

Mobile Ad Hoc Networks (MANETs), Peer-to-Peer (P2P) overlay networks and also WSNs, which are the main focus in this thesis, are the most prominent approaches of highly dynamic wireless networks, which resulted from the paradigm shift from centralised static wired networks, such as Local Area Networks (LANs), to distribute, highly dynamic wireless networks. In these highly dynamic wireless networks, each network entity acts autonomously and self-organised so that networks can be formed in a distributed fashion and on-demand. This is essential in most of today's dynamic environments because the network topology may change frequently due to several reasons such as moving, joining and leaving of nodes, but also due to environmental changes. Furthermore, the number of participating network nodes in such networks is increasing, as a result scalability is becoming a major concern.

As highlighted by Abraham et al. [1, p. 50], there is however a trade-off between scalability and determinism that has to be taken into account by the paradigm shift from systems with centralised control over distributed systems to self-organising systems, which is depicted in figure 1.1. In general, it can be stated that while the scalability is increasing towards self-organising systems, the determinism decreases.

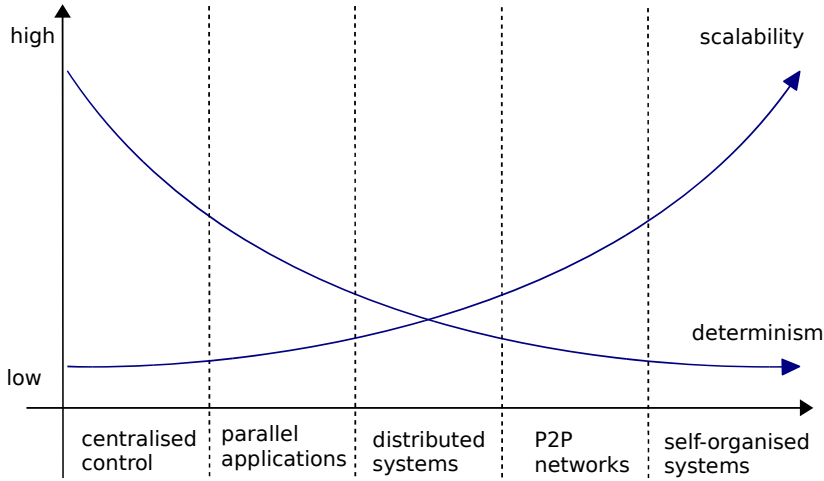


Figure 1.1: Scalability vs. determinism (cf. [1, p. 50])

One of the important research questions in the field of WSNs is how data packets can be transmitted from a source to a destination node, particularly when the sensor nodes are deployed in a random fashion and the radio range of single nodes is strictly limited. Or to be more precise: what makes a routing algorithm effective that meets all challenges of WSNs.

As the desired key factors for WSN routing algorithms, such as self-organisation, robustness, adaptivity and scalability, can also be observed in nature in several contexts, biologically inspired approaches are a promising starting point to get inspiration from for the development of routing algorithms. Although the idea of using Biologically-inspired Algorithms (BIAs) in the context of information technologies is not new, the key-success of those properties in nature makes them interesting for further considerations in the area of networking. There are several research areas in which BIAs are successfully applied to solve Combinatorial Optimisation Problems (COPs), such as the TSP [15], the knapsack problem [16], the vehicle routing problem [17], scheduling problems [18] etc. Due to the fact that routing algorithms can be considered as COP slightly modified BIAs could be applied to the area of routing in WSNs.

One of the most interesting approaches that seems to be directly trans-

ferable from nature to the area of networking are ACO algorithms [11]: the basic observation that can be made in nature is that ants are capable of finding the shortest path between their nest and food source by cooperating with each other via stigmergy. This idea can be transferred to the area of networking in which artificial ants are sent from a source node to a destination node in the network finding the shortest path. When more than one objective should be considered, ACO algorithms can be extended to MOACO algorithms [19] that are capable of considering multiple (conflicting) objectives at the same time within the optimisation process.

1.1 Research Question

The distributed nature of WSNs, their severe resource constraints as well as their application in hostile and unattended areas lead to the need of new security concepts to protect the network, particularly the routing of data messages. Common security mechanisms that are applied in wired networks, but also wireless networks, cannot be applied one-to-one to WSNs without modifications. Particularly, malicious nodes that are already part of the network are a serious problem that should be considered in WSNs. While common security mechanisms, such as cryptography and authentication, can help to protect the network against external attacks, those mechanisms cannot be used alone to deal with internal adversaries or faulty nodes that are a 'valid' part of the network. As a result, novel counter measures need to be found that can deal with these types of internal attacks, particularly to guarantee a sensible and economic routing in WSNs. One security mechanism that may help to solve this problem is the use of *trust* mechanisms.

One of the main challenges in WSNs is to combine resource-efficient routing with the use of secure and trustworthy routes, while obtaining a good performance. As the application of MOACO algorithms for the routing in WSN seems to be reasonable, one can pose the following question: why the idea of ant-based routing cannot be combined with the incorporation of trust?

The resulting main hypothesis of this thesis is that

MOACO-based routing algorithms, which are capable of considering multiple optimisation criteria at the same time, can solve the problem of trusted routing in WSNs, while providing a reasonable performance that is comparable to existing routing protocols.

1.2 Thesis Contributions

The main contributions of this thesis can be summarised as follows:

- Identification of security issues of WSN routing protocols, particularly considering malicious sensor nodes that are already part of the network, the so called *insider attacks*.
- Analysis of the current state of the art biologically-inspired algorithms that are suitable for the routing in WSNs with a special focus on ant-based routing approaches, in particular algorithms that are capable of dealing with multiple objectives, the so called MOACO algorithms.
- Review of the use of ACO-based algorithms for the routing in wired networks, MANETs and WSNs.
- Development and implementation of a MOACO-based framework for the routing in WSN that is able to optimise routes regarding multiple objectives, including trust.
- Experimental analysis and evaluation of the implemented MOACO-based routing framework for WSNs regarding several performance metrics and trust.

1.3 Thesis Scope

This thesis is divided into the following chapters (see figure 1.2):

Chapter 2 gives an overview of the theoretical foundations of the thesis that will provide a common basis to make the remaining chapters of the thesis comprehensible. After reading this chapter you will

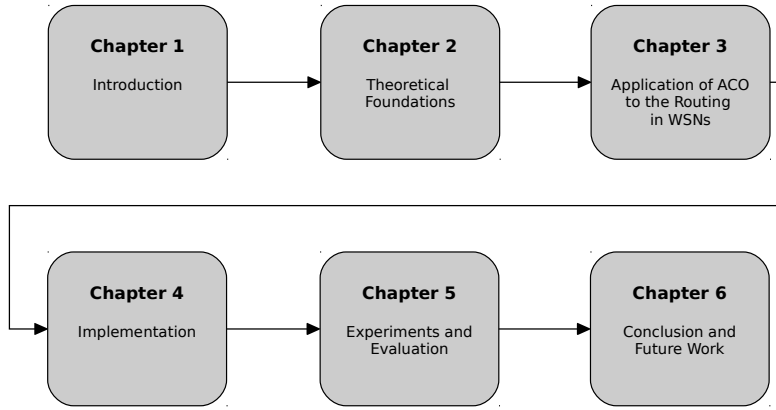


Figure 1.2: Overview of chapters of the thesis

have a basic understanding of Wireless Sensor Networks (WSNs), Trust and Reputation (TnR), Combinatorial Optimisation Problems (COPs), Multi-objective Combinatorial Optimisation Problems (MCOPs), Decision Maker (DM), Ant Colony Optimisation (ACO) and Multi-objective Ant Colony Optimisation (MOACO).

Chapter 3 discusses the use of ACO algorithms in the area of routing in WSNs. The chapter covers a general introduction to the general routing challenges in WSNs as well as the network layer attacks that can influence the routing performance in WSNs. Moreover, related works from the research area of ACO-based routing algorithms are presented, including their origin in wired networks, their use in MANETs and finally, the current state of the art in WSNs. After that latest findings in the area of MOACO-based routing algorithms are presented.

Chapter 4 presents the implementation of the Multi-objective Ant Colony Optimisation Routing Framework for WSNs (MARFWSN), a MOACO-based routing framework for WSNs that is implemented in the network simulator OMNeT++ as part of the conducted research. The used methodology is presented and a detailed description of the imple-

mented application layer and network layer will be given. Moreover, the parameters that influence the implemented layers are discussed.

Chapter 5 explains the configuration of MARFWSN simulation scenarios and the conducted experiments using the network simulator OMNeT++. The simulation results are analysed and evaluated regarding several network metrics.

Chapter 6 summarises the conducted research and draws conclusions of the results and findings from the conducted experiments. Besides, future work is presented that can enhance this research in the future.

Readers that are familiar with the research area of WSNs and MOACO algorithms in general may skip the second chapter and continue directly with chapter 3.

2

Theoretical Foundations

> You can't build a reputation on what you are going to do.
>
> Henry Ford (1863 - 1947)

In this chapter, the theoretical foundations of the thesis will be presented that should enable the reader to get the required background knowledge on which the following considerations of this thesis will be based on. Starting with a general introduction about the Wireless Sensor Network (WSN) basics, including challenges and corresponding security issues, the concept of soft security will be introduced, which is closely linked to the terms of trust and reputation. Due to the fact that routing can be considered as an optimisation problem, the theoretical foundations of Combinatorial Optimisation Problems (COPs) and their extensions Multi-objective Combinatorial Optimisation Problems (MCOPs) will be introduced. Subsequently, the metaheuristic of Ant Colony Optimisation (ACO), a biologically-inspired approach, is introduced that can be used to solve COP problems as well as its extension Multi-objective Ant Colony Optimisation (MOACO), which is capable of taking multiple objectives into account. Finally, the presented theoretical foundations will be summarised.

2.1 Wireless Sensor Networks

A Wireless Sensor Network (WSN) is a network that consists of small low-cost, low-power wireless communication devices, so called *sensors* or *sensor nodes*, which are capable of interacting with their environment by sensing or controlling physical parameters and performing simple computations on them. The main idea of WSNs is that sensors detect events or phenomena at a certain place or in a certain region, collect, process, aggregate and finally transmit these data to the final destination in the network, e.g. an interested end-user's notebook. As the sensors are be-

coming smaller and smaller and thus also cheaper, WSNs are expected to play an important role in the upcoming era of *pervasive computing* [20], i.e. the integration of information processing in everyday things and activities.

The communication between sensors in a WSN is realised using the wireless channel. Due to the fact that the radio range of each sensor is limited because of strict energy constraints the sensor nodes need to cooperate with each other to transport data via multiple hops from the source to the destination by using intermediate nodes. Also considered here as one of the main research questions in this thesis is the *routing* in WSNs, i.e. how data can successfully be transported from a source to a destination node in an efficient and secure way through the network.

In the following section some basics about the application areas of WSNs, the sensor node hardware and the network architecture of WSNs will be discussed in more detail. Thereafter, a short comparison between WSNs and Mobile Ad Hoc Networks (MANETs) is drawn because both network types are often put on the same level, though there are significant differences. Finally, the unique constraints and challenges of WSNs are summarised.

2.1 Application Areas of WSNs

There are several application areas in which WSNs can be used. An overview of basic application areas of WSNs is shown in the following[1] (cf. [21]).

- **Disaster relief applications**: A typical scenario is a wildfire detection in which sensor nodes are deployed over a certain area by an air plane. Sensors collect temperature information and produce collaboratively a temperature map, which then can be read out by fire fighters equipped with a smart phone.
- **Environment control and biodiversity mapping**: WSNs can be used, for example, to monitor the erosion processes on the ground

[1] Though, here only peaceful applications of WSNs from the civilian sector are listed, WSNs are also used in the military sector, e.g. sensors could be dropped from planes to monitor hostile areas, which are too dangerous to access by humans. However, in this thesis a peaceful use of WSNs is assumed.

of the ocean. Closely related is biodiversity mapping in which a number of plants or animals in a certain region is monitored.

- **Intelligent buildings**: To reduce the energy use of buildings, WSNs could be deployed to measure temperature, humidity and airflow, which then could be used to adapt the temperature within the building automatically. Also sensors could be used to monitor the mechanical stress level of buildings, such as bridges, to find out the likelihood of a collapse.

- **Facility management**: In larger facilities with multiple buildings sensors could be used to track vehicles in that area or to detect intruders. Another application could be the deployment of sensors in a chemical plant to detect leaking chemicals.

- **Machine surveillance and preventive maintenance**: Sensor can also be placed at industrial machinery, such as industrial robots, that cannot be accessed easily by humans. Vibrations could be measured to find out the need for maintenance.

- **Precision agriculture**: By placing humidity and soil sensors in the field, the amount of irrigation and fertiliser can be precisely determined.

- **Medicine and health care**: Sensors can be used to monitor patients after an operation. Also long-term surveillance of patients, e.g. for elderly people, which triggers an alarm in the case of emergency could be a useful application.

- **Logistics**: Sensors can be placed at certain items, such as parcels, to allow a simple tracking of objects during transportation or within a warehouse.

- **Telematics**: Sensors can be embedded into streets to collect information about current traffic conditions. These sensors could be read out by passing cars so that they could get alarmed in case of dangerous traffic situations.

In WSN applications there are two common interaction patterns: *event-driven* and *time-driven*. *Time-driven* WSNs provide snapshots of relevant

data at periodic intervals, i.e. the sensor nodes periodically switch on their sensors, sense the specified attributes and transmit the result in certain time intervals to other interested nodes. In contrast, *event-driven* WSNs react immediately to occurrences of specified events that are monitored. Also *hybrid* networks using a combination of both types are possible, for example, an event triggers a periodic reporting of a certain attribute.

2.1 Sensor Node Hardware Components

Depending on the application of the WSN, the hardware components of the sensor nodes have to be chosen carefully regarding several properties such as size, cost and energy consumption. In figure 2.1 the general components of a sensor node are depicted.

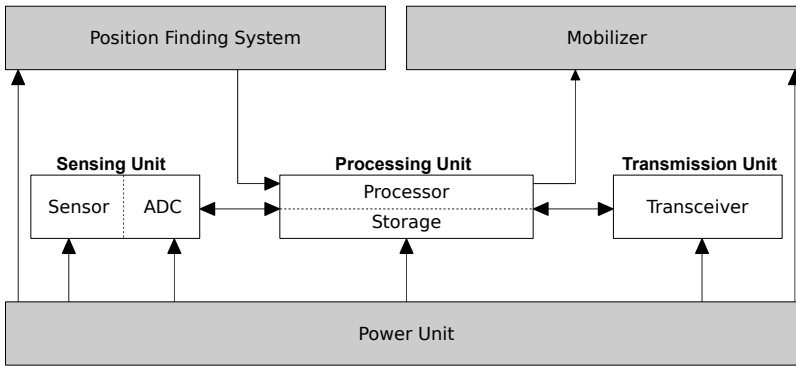


Figure 2.1: Components of a sensor node (cf. [2])

Basically, a sensor node consists of a *sensing unit*, a *processing unit*, a *transmission unit* and a *power unit*. The *sensing unit* is used for the measurement of one or more physical phenomena such as temperature, humidity, pressure, acceleration, visual, infra-red, acoustic, vibration, magnetism etc. The sensing unit includes an ADC that converts the continuous quantity of the measured parameter to a discrete digital number. The *processing unit* temporarily stores the converted digital numbers in its memory and processes the data, e.g. by aggregating the data with data of other nodes. Using the *transmission unit*, the aggregated data can be transferred to neighbouring nodes or correspondingly data can be

received from surrounding nodes. Additionally to the basic components, a *position finding system*, such as the Global Positioning System (GPS), can be attached to the sensor node as well as, in special application cases, a *mobiliser* if the sensor node should be able to move. The *position finding system* of the nodes can be a useful tool for location-based routing. However, the disadvantages of an additional GPS are its costs and the limited indoor functionality. Besides, a GPS device uses additional energy that cannot be neglected. All components of a sensor node are powered by a power unit, in most cases a simple battery. Due to the fact that sensor nodes are often deployed in unattended or hostile environments, the power source cannot be replaced. As a result, the sensor's lifetime is a crucial design factor in hard and software design of WSNs. A detailed investigation and discussion of current sensor node hardware can be found in references [22–24].

2.1 Network Architecture of WSNs

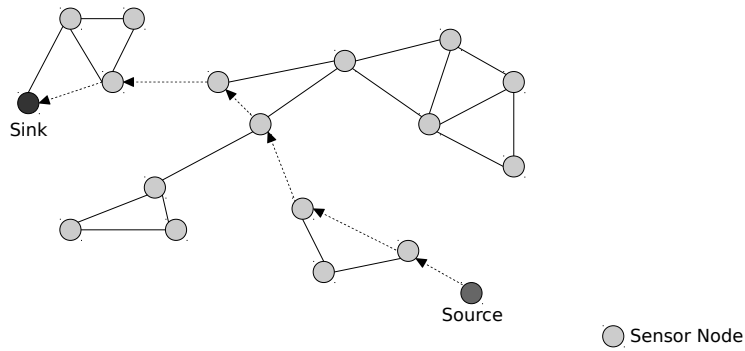


Figure 2.2: Architecture of a WSN

In figure 2.2 the basic architecture of a WSN is depicted. The sensor nodes are scattered in a certain area which is to be monitored. Neighbouring nodes that are within each other's radio range can communicate with each other through wireless channel. Apart from the 'normal' sensor nodes, which are measuring physical parameters, there are two types of special nodes in the network: the *source* and the *sink*.

The *source* is an entity in the network, typically a sensor node, that provides certain information about an event. In contrast, the *sink* is an entity that has an interest in a particular information in the network[2].

The sink can be either a part of the network or an entity outside the WSN. In the former case, the sensor node is just a 'normal' sensor node in the network having the same features as all the other sensing nodes. In the latter case, the data of the WSN is accessed by a device which is located outside the network, such as a Personal Digital Assistant (PDA) or smartphone. The device is normally directly connected to one of the sensor nodes in the network to obtain data. Furthermore, the sink can also be a device at the edge of the WSNs, such as a gateway, to allow remote connections to the WSN via other networks such as the Internet.

As consequence of power restrictions and the use of the radio communication, the communication between the sensor nodes is limited so that adequate distance between the sending and receiving node has to be considered. Also, obstacles or interferences have to be taken into account, which may additionally decrease the possible communication range. As a result, it is in most cases not possible to establish a direct connection between source and sink. To overcome the limitation of radio range, the transmission of data via multiple hops, operating in a store-and-forward fashion, is an obvious solution. A side effect of using multiple hop communication is that the sensor nodes can operate more energy efficient regarding their transmission power. This is based on the fact that the attenuation of radio signals is at least quadratic in most environments. However, the additional energy that is required for store-and-forward needs also be taken into account because if the distance between the nodes is getting too small, the energy for storing and forwarding is bigger than the energy saved by the reduced transmission power (cf. Min and Chandrakasan [25]). Nevertheless, the multiple hop approach also allows to focus large raw-data streams by doing in-network processing so that information can be aggregated.

[2]However, due to the fact that in other networks, such as LANs, WLANs and MANETs, the end points of a connection are referred to as *source* and *destination*, in the following a *sink-source-pair* is used synonymously with a *source-destination-pair* so that processes in the network can be described understandably for the 'common' reader with networking background.

2.1 Difference between MANETs and WSNs

MANETs and WSNs are often mentioned in the same context or even said to be equivalent, but although there are some similarities, there are some differences that make both types of networks clearly distinguishable:

While MANETs are rather made for the interaction with humans, WSNs are made for the interaction with the environment. In general, in a MANET a 'normal' TCP/IP network stack is used that enables the transmission of data packets between network participants allowing diverse applications. In contrast, WSNs are rather tight to exactly one application fulfilling a certain application task. However, due to the predefined application task the traffic in WSNs is more predictable than in MANETs in which traffic burst and resulting congestion may become an issue. While for WSNs best-effort services are in most cases sufficient, in MANETs some application need some sort of Quality of Service (QoS), e.g. for Voice over IP (VoIP). For the lower layers in MANETs Institute of Electrical and Electronics Engineers (IEEE) 802.11 is used, where in contrast for WSNs often IEEE 802.15.4 is utilised.

In general, it can be stated that WSNs are used for answering questions instead of just transferring data so that WSNs can be considered as data-centric, while MANETs can be considered as node-centric. As a consequence, a single sensor in a WSN is not as important as a node in a MANET. Due to the application case of WSNs, WSNs are working unattended and the number of nodes as well as the node density is much higher than in MANETs. WSNs are often deployed in a random fashion and also redundant so that hundreds or even thousands of sensors per network are no rarity. Thus, scalability issues are more severe in WSNs than in MANETs.

The network nodes in MANETs are normally more powerful devices, such as smart phones, PDAs or notebooks, whereas in WSNs special sensor nodes are utilised that have strong resource constraints. As future vision often Smart Dust [26] is mentioned, in which sensor nodes of milimeter-scale are assumed as long-term goal. Although the self-organisation of the network topology is a feature MANETs and WSNs share, there are different reasons why the topology is changing: while in MANETs the

topology changes due to joining/leaving nodes as well as the movement of the nodes, in WSNs the topology changes due to node failures, temporary interferences or the sleep-awake cycles of the sensor nodes. In most WSN scenarios the nodes are deployed in a static fashion so that movement is not an influence factor of the network topology. In WSNs the energy is a major concern because normally each sensor node is equipped with a battery that has to last the sensor's entire life, i.e. months or even years. In contrast, nodes in MANETs are normally connected to a stronger energy source with a direct connection to power supply or with the possibility of recharging the battery so that energy constraints are not really existing. Although multi-hop is a feature of both network types, multi-hop is a stronger concern in WSNs because the radio range of sensor nodes is significantly smaller than in MANETs so that more hops are required to span the same distance.

2.1 Unique Constraints and Challenges of WSNs

Due to the wide range of different applications it is impossible to specify a single WSN implementation that satisfies all needs. Nevertheless, there are several characteristics that should be taken into account for WSN applications:

- **Type of service**: The purpose of traditional communication networks is to transfer raw bits from a source to a destination. In contrast, WSNs are expected to perform better in terms of delivering meaningful information for a given task instead of just transferring raw bits – as Steven Glaser, UC Berkeley points out "People want answers, not numbers" [27]. For that reason, new ways of providing services in WSNs have to be found taking this additional requirement into account.

- **Quality of service**: The QoS in WSNs depends also heavily on the application area of the network: for periodic measurements with small data transmission some packets lost or delay may not be crucial, whereas in applications that have real-time requirements those packet losses could cause fatal results. On the whole, the amount and quality of information received at the sink are important.

- **Fault tolerance**: Node failures are quite likely in WSNs because nodes may run out of energy, get destroyed or the wireless communication between nodes may be disturbed for a long period of time. Thus, fault tolerance is an important challenge that needs to be taken into account for WSNs.
- **Lifetime**: Due to the fact that sensors in WSNs are normally battery-powered, the sensors are severely limited in their energy source. In most application scenarios it is not possible to exchange the battery, e.g. if the sensors are deployed in an unattended or hostile environment. For that reason, an energy-efficient operation of sensor nodes is required to maximise the nodes lifetime and thus, the network's lifetime.
- **Scalability**: The architecture and protocols that are used in WSNs have to take scalability into account because for most application scenarios, it is envisioned that hundreds up to thousands of nodes are deployed for an application.
- **Density**: Particularly, if nodes are placed in a random fashion, e.g. dropped from an air-plane, the density varies heavily. Furthermore, the sensor density is permanently changing due to node failures or node movements. Therefore, WSNs need to be capable of adapting to varying sensor densities.
- **Programmability**: During the operation of a WSN, its requirements may change, i.e. another task may get more important than the previous one. Thus, a fixed way of information processing is not suitable for WSNs so that it is useful to provide programmability to react to the changes whenever required.
- **Maintainability**: A WSN must be able to adapt to both: changes in the environment as well as changes in the WSN itself such as node failures, new tasks etc. For that reason, the WSN must be capable of monitoring itself and automatically change certain parameters. The remote management functionality should be minimised so that the network is rather self-maintaining.
- **Unreliable communication**: Due to the use of the wireless medium as communication channel, the transfer of data packets is

unreliable, i.e. packets can get damaged and lost because of channel errors or packets are dropped due to congestions. Also the probability of packet collisions increases in dense networks. Another issue is the latency of transmissions caused by the multi-hop routing, node processing as well as network congestions.

- **Unattended operation**: WSNs are often deployed in unattended and hostile areas so that the sensor nodes are vulnerable against physical attacks of adversaries, but also severe weather or other natural disasters. Due to the fact that the sensors are normally not accessible by the owner, physical damage or tampering of sensor nodes can hardly be detected.

As a result, several characteristics of WSNs have to be considered:

If data need to be transferred over long distances using wireless links a huge transmission power is required. To reduce the transmission power often *multihop communication* is used, i.e. intermediate nodes are relaying the data until it reaches the destination.

As stated before, sensor nodes are strongly energy limited. Thus, *energy-efficient operation* is a key factor to achieve long lifetime.

Due to the vast amount of sensor nodes in the network, the changing environment, the issues of node failures and the fact that the locations of sensor nodes are in most cases unclear before the deployment, the sensors should configure autonomously in an *auto-configuration* fashion to form a functioning network, i.e. without any additional human interaction.

Another special characteristic of WSNs is that the sensors *collaborate* to do certain tasks a single node could not do on its own. Moreover, *in-network processing* is often used to aggregate data, reducing the amount of data that needs to be transferred through the network.

In contrast to traditional communication networks in which the transfer of data is rather address-centric, most of the WSNs are shifting to a new *data-centric* addressing paradigm. In a data-centric network the data itself is important rather than the node the data is coming from. The huge amount of nodes and thus, the redundancy of nodes supports this approach in WSNs.

To support scalability the principle of *locality* is often used, i.e. the nodes consider only surrounding nodes that are located in their direct neighbourhood for making decisions. This approach limits the amount of information that needs to be stored at single nodes and transmitted to other nodes and therefore, leads to more efficient protocols.

Finally, there are always *trade-offs* that have to be considered, such as higher energy requirement for more accuracy or longer network lifetime against the lifetime of individual nodes etc.

2.1 Security in WSNs

Security plays an important role in WSNs due to the fact that the sensors are often deployed for mission critical tasks in unattended and hostile environments. In such environments a certain level of security is required. In contrast to traditional wired computer networks, the implementation of security for WSNs is more difficult: the major problem is the use of the shared wireless medium for communication which leads to privacy issues and provides adversaries with an easy target. In most cases it is not possible to apply traditional security techniques that have been in wired networks for years, such as cryptographic techniques, directly without any modification to WSNs.

The main implementation difficulties of security mechanisms for WSNs are the constrained hardware resources in terms of limited energy, computational power and memory. The main goals for WSNs security implementations are low communication costs and low resource utilisation to minimise the energy consumption of the sensor nodes.

Besides, the problem of physical attack needs to be considered, if the sensors are deployed in hostile environments. In this case, adversaries can easily access the sensor nodes physically that are deployed on their own terrain. Also the scalability of security mechanisms must be taken into account, for instance, the distribution of keys has to be considered if cryptographic measures are to be applied. All in all, the implementation of security mechanisms in WSNs is always a trade-off between the mutually exclusive factors of security maximisation and energy minimisation.

In the following, first, the basic security requirements for WSNs are dis-

cussed. After that, the term 'soft security' is introduced. Then, the fundamentals of trust and reputation are presented.

Basic Security Requirements

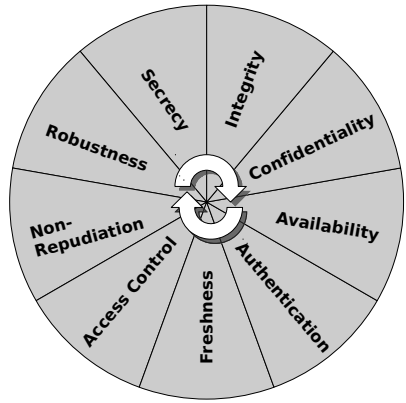


Figure 2.3: Basic security requirements for WSNs

To achieve secure WSNs there are several basic requirements that need to be taken into account (cf. [28–30]). The major requirements that should be considered are depicted in figure 2.3. In the following, these basic requirements are explained briefly:

- **Confidentiality**: The sensed data as well as any other control information, such as routing information, which is transported through the network needs to be protected from disclosure. Moreover, information that is stored on each sensor node, such as public or symmetric keys, needs to be protected against adversaries.

- **Integrity**: The integrity of data that is transported in the network should be ensured so that any modification, insertion, deletion or replay of data can be detected. This should also cover shared keys needed for cryptographic measures.

- **Authentication**: To achieve authentication it should be assured that a communicating sensor is the one that it claims to be and that the origin of the data received by another node is as claimed.

- **Authorisation/ access control**: Unauthorised use of network resources has to be prevented by assigning access rights to valid network participants. Sensor nodes that cannot be successfully authorised should be isolated form the network.

- **Availability**: The service provided by the WSN should be permanently available, i.e. unnecessary processing has to be minimised. For that reason, WSNs need to be protected against attacks that target the network availability such as Denial of Service (DoS) attacks. Also the security mechanisms should be available all the time, particularly single points of failure should be avoided.

- **Robustness**: The structure of the WSN should be robust so that single node failures or interferences do not affect the overall network functionality. Also counter measures against more sophisticated attacks e.g. against the Media Access Control (MAC) layer, for instance intentionally transmitting while the other node is transmitting, need to be taken into account. Spread-spectrum techniques [31] could be used to make the WSN less susceptible to those types of attacks.

- **Freshness**: The freshness of data has to be guaranteed in WSNs because sensed values that are already outdated are useless in almost every application scenario. For security reasons this should be considered as well to avoid replay attacks, e.g. by reusing shared keys.

- **Secrecy**: In more sophisticated WSNs it should also be ensured that leaving sensor nodes cannot read any further messages in the network. Likewise new joining nodes should not be able to read any messages that been transferred before their joining.

- **Non-repudiation**: For some application scenarios it may be required to explictly prove the authorship of a particular message.

Ideally all of them should be considered, but it is more likely that due to latency issues or energy constraints only certain requirements are chosen regarding the application area and the level of security that is required for the WSN.

2.2 Trust and Reputation

Trust and Reputation (TnR) are two supportive tools that can be used to improve the decision making by reducing the risk of interactions and transactions between entities. From the social perspective, since the beginning of mankind trust and reputation were two important factors to assess the trustworthiness between human beings. With the forming of civilisation the original concept of interpersonal trust was then extended and transferred to a commercial context so that trust relationships between sellers and buyers could be assessed using these tools.

Thousands of years later, in the age of computer science, new challenges arose with the development of electronic devices and corresponding communication networks in terms of finding means to assess the trustworthiness of devices or virtual agents before exchanging information. How trust and reputation can be used in the context of WSNs will be discussed in the following.

2.2 Hard vs. Soft Security

Most of earlier researches in the area of networking assumed that all nodes of a network are cooperative and trustworthy. Rasmusson and Jansson [32] took a more differentiated look at security by introducing the distinction between *hard security* and *soft security*.

While traditional security mechanisms, such as authentication and access control, are classified as hard security, a new perspective of security, the social control component, was introduced as *soft security*. In contrast to hard security mechanisms, which try to keep adversaries out of the system by all available means, soft security mechanism accept that there might be intruders in the system. However, the goal of the social control component is that it should identify the intruders and try to restrain them from causing harm to the system.

Jøsang et al. [33] take up the point of soft security and define it as "collaborative enforcement of, and adherence to common ethical norms by participants in a community". To identify misbehaving members in a community, i.e. members breaching ethical norms, soft security mecha-

nisms use collaborative methods for assessing the members' behaviour.

One of the mechanisms that implement the idea of soft security is the use of Trust and Reputation Systems (TRSs), which enable participating entities in a community to find misbehaving ones based on trust and reputation. There are several applications of TRSs in the area of electronic commerce (e-commerce) and communication networks such as MANETs and Peer-to-Peer (P2P) networks.

2.2 Notion of Trust and Reputation

In the area of computer science the notion of trust and reputation is ambiguous due to the fact that there are various definitions by several authors that differ in their meaning and complexity. A detailed discussion of this issue can be found in the research of McKnight et al. [34].

For a consistent view of trust and reputation and related terms in this thesis the basic terms are briefly introduced again in the following paragraphs:

Trust In this thesis, the following definition of trust is used, as defined in the Oxford English Dictionary [35]:

> Trust is the "confidence in or reliance on some quality or attribute of a person or thing, or the truth of a statement"

Basically, trust is a directional relationship between two parties or entities, the *trustor* and the *trustee*. The trustee can be anything from a person, to an organisation or to a physical entity. The trustor makes assessments and decisions based on received information and past experience (cf. [33]).

A trust relationship is *context sensitive*, i.e. trust is always applied in a certain *scope* or *context*. Consequently, a trust relationship is applied to a specific purpose or domain of action, such as "being the person claimed to be" or "providing correct information". If both parties of a trust relationship trust each other within the same scope, this is referred to as *mutual trust*.

In the first place, trust is *unidirectional*, i.e. an entity can assess another entity and can have trust in it based on the gathered knowledge about that entity. However, this relationship does not necessarily has to be reciprocal because the other entity may have another opinion about the other party.

Trust is a personal and *subjective* phenomenon based on various factors or evidences with different weightings. In general, personal experiences have more weight than recommendations from others. The subjectivity of trust is based on the fact that an entity's trustworthiness does not only depend on its behaviour, but rather on how its behaviour is perceived by other entities.

There are two main interpretations of trust: *Reliability trust* and *decision trust*. Reliability trust is the reliability of something or somebody independently of any actual commitment. In contrast, decision trust is the willingness of a party to depend on something or somebody in a given situation, even knowing negative consequences may occur. Both, reliability trust and decision trust, are a positive belief about something on which the trustor depends for his welfare (cf. [33]).

Uncertainty Trust plays an important role, particularly in environments in which entities depend on each other to reach a certain goal, but in which at the same time *uncertainty* is present (cf. [36]). Uncertainty is defined in the Oxford English Dictionary [35] as

> "The state of not being definitely known or perfectly clear; doubtfulness or vagueness"

In a communication network uncertainty means that an entity is unclear about the environment and communication partners so that it cannot anticipate or accurately predict the outcome of interactions.

Uncertainty originates from two sources: *information asymmetry* and *opportunism*. Information asymmetry may occur if a party does not have all of the information it needs, while opportunism describes the fact that both parties behave opportunistically to serve their self-interest.

Transitivity of Trust A trust relationship can be *transitive*, though it does not necessarily has to be. For example, if Alice trusts Bob, and Bob trusts Carol, then Alice can also trust Carol (see figure 2.4). This *chained trust* is possible due to the assumption that Alice trusts Carol because of Bob's *recommendation* on Carol.

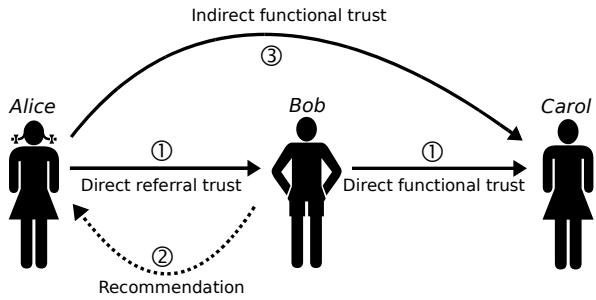


Figure 2.4: Trust transitivity

As highlighted by Jøsang and Pope [37], trust is *conditionally transitive*, i.e. trust is only transitive under certain semantic constraints: one crucial factor for trust transitivity is that trust can only be derived if the scope of trust is the same between all involved parties. Based on this assumption recursive 'referral' trust paths of arbitrary length can be established, also referred to as transitive *trust chains*.

The ability to refer trust to a third party is called *referral trust* and that is what makes trust become transitive in the first place. However, at the end of each trust chain there has to be *functional trust*, i.e. the trust that a certain function can be fulfilled by this party.

Based on the *direct* referral trust between Alice and Bob as well as the direct functional trust between Bob and Carol, an *indirect* functional trust between Alice and Carol can be established (see figure 2.4).

Multidimensionality of Trust Due to the complexness and multidimensionality of trust, there are multiple research streams dealing with this topic. The major research streams include *calculus-based trust*, *knowledge-*

based trust and *institution-based trust*.

The idea of calculus-based trust is, for example, used in economic transactions in which trust is developed by the participating parties in a calculative manner [38]. One party calculates the costs and benefits of other party's cheating or cooperating in a transaction. If the probability that a party is performing a beneficial action to the first party is high, trust can be developed.

In first researches in the area of knowledge-based trust, trust is developed by aggregating trust related knowledge of involved parties [39]. The knowledge is either accumulated first-hand or second-hand. Gefen et al. [40] introduce the concept of familiarity, i.e. considering the experience 'with what', 'who', 'how', and 'when of what' is happening. With familiarity the social uncertainty can be reduced through knowledge gained from previous interactions.

In institutional-based trust a party believes that the necessary impersonal structures are in place to enable acting in anticipation of a successful future endeavor [41]

Reputation Strongly linked with the concept of trust is the term *reputation*. The Oxford English Dictionary [35] describes the term 'reputation' as:

> "the condition, quality, or fact of being highly regarded or esteemed; credit, fame, distinction; respectability, good report."

In the context of communication networks, reputation can be seen as global perception of a node's trustworthiness in a network (cf. [36]). In contrast to trust, reputation can be considered as collective measure of trustworthiness (reliability) based on recommendations from members of a community.

Reputation can be either *group related* or *individual related*. For example, the group reputation can be computed by the arithmetic mean of a group members' individual reputations.

2.2 Classes of Trust

Grandison and Sloman discuss in their survey classes of trust [42] based on literature review. Though, the authors emphasize that the presented taxonomy is not exhaustive, it provides a useful tool for the classification of trust. The following five major classes of trust were identified by Grandison and Sloman:

- **Access trust**: A trustor trusts a trustee to access a resource, the trustor owns or controls. Access trust is strongly related to the security paradigm of access control.
- **Provision trust**: A trustor trusts a trustee to provide a service. This does not include the access to the trustor's resources.
- **Certification trust**: The trustworthiness of a trustee is based on certification by a third party, i.e. a set of certificates is presented by the trustee to the trustor.
- **Delegation trust**: A trustor trusts a trustee to make decisions on its behalf, with respect to a resource or service that the trustor owns or controls. As Grandison and Sloman point out, delegation trust can be seen as a special form of provision trust.
- **Infrastructure trust**: The trustor must trust the basic infrastructure, such as a network or a server, to support transactions with a certain level of security.

In this thesis, the class of provision trust is mainly considered because for the routing of packets in WSNs, the sensor nodes need to trust the other nodes in the network that provide the service of packet forwarding.

2.2 Trust in WSNs

While most researches consider trust in WSNs from a communication-centric point of view, it should be noted that the major functionality of WSNs is the sensing of data. Therefore, additionally to the communication angle of trust, also the trust of sensed data should be taken into account. Figure 2.5 shows an extended version of how trust in WSNs can be classified based on the idea presented by Srinivasan et al. [3].

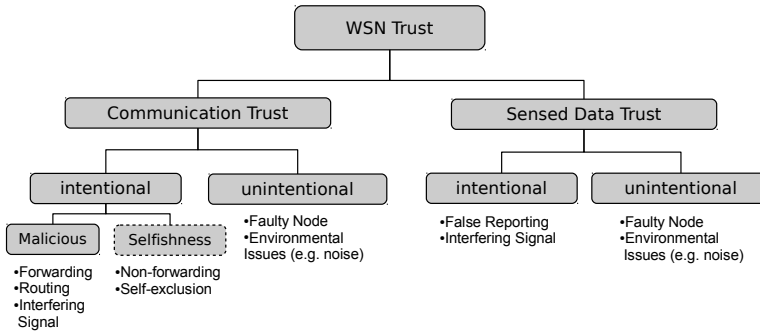


Figure 2.5: Trust in WSNs (cf. [3])

Basically, trust in WSNs can be subdivided into *communication trust* and *sensed data trust*. While communication trust covers the correct routing and forwarding of packets in the network, sense data trust deals with the correctness of the sensed data provided by the sensor nodes. Both, the communication trust as well as the sensed data trust, are affected by intentional and unintentional issues.

In the case of communication trust, the intentional misbehaviour can be sub-classed into *malicious* and *selfish* behaviour. Malicious behaviour includes adversary attacks on the forwarding and routing mechanisms in the WSN such as dropping, modifying or fabricating packets (see section 2.1.6 for network layer attacks). In contrast, a node that behaves selfishly tries to maximise its own benefit in terms of saving power, CPU cycles memory etc. Though, selfish nodes act on their own, they are harming the network and it is difficult to figure out whether a node is acting selfishly or 'just' harming the network. However, in most scenarios selfishness of nodes can be neglected due to the fact that the sensor nodes are deployed by a single entity to fulfil a certain task so that nodes which have not been modified never act selfishly, except when they are programmed to do so. There are also some exogenous factors that have an effect on the communication such as faulty nodes or environmental issues such as thermal noise.

On the other hand, the sensed data trust is affected by intentional false

reporting on sensed data or interfering signals sent out by adversaries, who try to manipulate the overall result. Moreover, environmental issues may take influence on sensed data.

2.3 Combinatorial Optimisation Problems Fundamentals

As routing can be considered a Combinatorial Optimisation Problem (COP), in this section the fundamental concepts that are related to COPs are introduced and discussed.

2.3 Global Optimisation

In many practical fields, such as advanced engineering design, data analysis, risk management, communications and others, problems need to be solved that involve *global optimisation*, i.e. finding optimal parameters for complex systems.

But what does the term 'optimal' in this context mean? From the view of applied mathematics global optimisation for a single-objective optimisation problem is the determination of the global optimum of a continuous or discrete cost function, the so called *single-objective function*, of an arbitrary number of independent variables in the presence of multiple local optima.

More formal an optimisation problem can be defined as follows:

> **Definition 1 (Optimisation problem)** *An optimisation problem can be represented as tuple (S, f) in which S is as set of feasible solutions and f is a cost function that assigns to each feasible solution an objective function value from $\mathbb{R}$.*
>
> $$f : S \to \mathbb{R} \tag{2.1}$$
>
> *The goal of the optimisation process is to find an optimum $s^* \in S$ that minimises or respectively maximises the cost function.*

Normally, optimisation problems are defined as *minimisation problems*

due to the fact that *Maximisation problems* can be solved by minimising f's negation $(-f)$[3].

In the next step, the local and global minimum as well as the global optimum for the single-objective function f are formally introduced:

Definition 2 (Local minimum) *An objective function f is said to have a local minimum at the point x^* in a certain ϵ-neighbourhood if:*

$$\forall x \in S, \exists \epsilon > 0 : |x - x^*| < \epsilon, f(x^*) \leq f(x) \tag{2.2}$$

Definition 3 (Global minimum) *An objective function f is said to have a global minimum at the point x^* if:*

$$\forall x \in S : f(x^*) \leq f(x) \tag{2.3}$$

An example of local and global maxima and minima is depicted in figure 2.6.

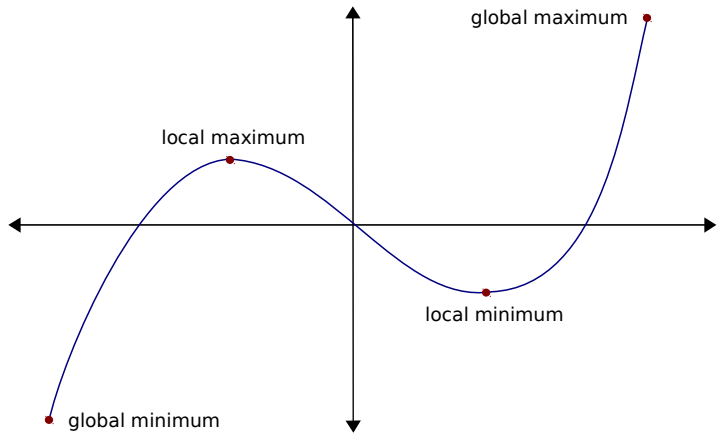


Figure 2.6: Local and global maxima and minima

[3]Without loss of generality in the following only the *minimisation problem* is considered

Definition 4 (Global Optimum) *A global optimum x^* of an objective function f is, depending on the optimisation problem, either a global maximum or respectively a global minimum.*

The major problem of global optimisation is the ubiquity of local optima, which are increasing with the size of the problem. For that reason, one of the basic requirements of global optimisation algorithms is that they must avoid to remain in local minima and instead, continue to search for the best possible optimum. [43]

Phases of Optimisation

Mathematical optimisation consists of two phases: *modelling* and *solving* (see figure 2.7). In the modelling phase three major aspects have to be considered: the selection of design variables, the selection of an objective function according to the optimisation goal and the specification of constraints that should be taken into account.

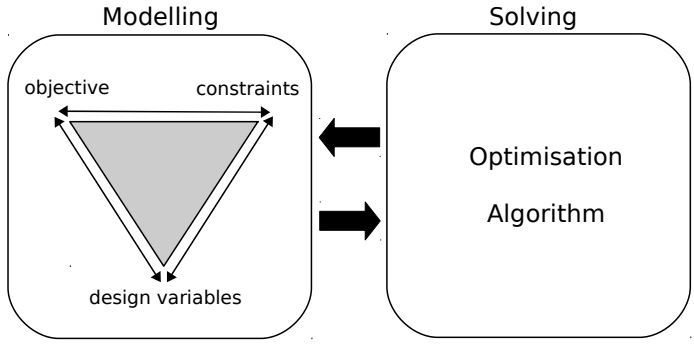


Figure 2.7: The concept of mathematical optimisation: modelling and solving

There is a close relationship between the different aspects of the modelling of an optimisation problem: the design variables should quantify the objective function as well as the constraints. The constraints and the objective function are often exchangeable e.g. it can be optimised for 'cost' with the constraint of 'quality' or *vice versa*. Subsequently to the

modelling phase there is the problem solving phase in which an optimisation algorithm is applied to the modelled optimisation problem. It should be highlighted that both phases are strongly related: on the one hand, a suitable optimisation algorithm needs to be chosen for a specific optimisation problem and, on the other hand, the optimisation problem needs to be adjusted to the optimisation algorithm that should be applied. [44]

2.3 Combinatorial Optimisation Problems

A special subset of global optimisation problems are COPs, which are concerned with finding optimal solutions for discrete problems. The term 'optimal solution' refers to the best among all feasible solutions of a given problem according to a given cost function. The set of solutions of a COP is finite and any solution has some combinatorial property such as a permutation, an arrangement of objects or a tree/graph which indicates a relationship between those objects [45].

Formally, an *instance of a COP* can be defined as follows:

Definition 5 (Instance of a combinatorial optimisation problem) *An instance of a combinatorial optimisation problem can be denoted as* $P = (S, \Omega, f)$ *with*

- $S = \{s_1, s_2, ..., s_k\}, k \in \mathbb{N}$, *a finite set of feasible solutions*
- $\Omega(t)$, *set of constraints among the variables*
- $f(t)$, *objective function* $f : S \to \mathbb{R}$, *which assigns a real-valued cost* $f(s,t)$ *to each feasible solution* $s \in S$

The purpose of the optimisation process is to find the element $s^* \in S$, *which minimises the function* f *and satisfies all constraints in* Ω, *the so called global optimum of this instance.*

$$s^* = \arg\min_{s \in S} f(s) \tag{2.4}$$

The constraints $\Omega(t)$ *as well as the objective function* f *can be time-dependent on* t.

Figure 2.8 depicts an example of the mapping between the decision vector x and the objective function f.

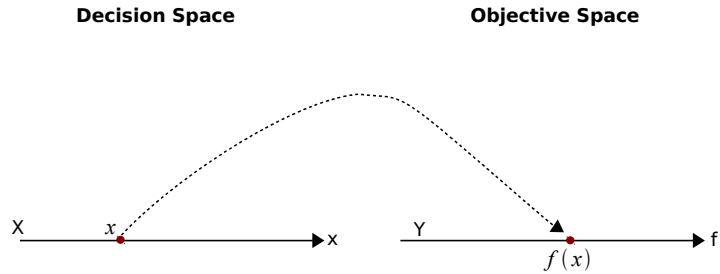


Figure 2.8: Mapping from decision space to objective space

Based on that the definition of a combinatorial optimisation problem can be derived:

> **Definition 6 (Combinatorial optimisation problem)** *A combinatorial optimisation problem is a set of instances of an optimisation problem.*

However, instead of using the introduced notation, in practice, often a rather compact representation of instances of a combinatorial problem is used, which is based on *solution components*.

> **Definition 7 (Compact representation of an instance of a COP)** *A compact representation of an instance of a combinatorial problem can be denoted as triple* $< C, \Omega, J >$ *with*
>
> - $C = \{c_1, c_2, ..., c_{n_C}\}, n_C \in \mathbb{N}$, *finite set of variables (decisions) to select, the so called solution components*
> - Ω, *finite set of constraints*
> - $J(S)$ *real-valued cost function*
>
> *The set of feasible solutions* S *is a subset of solution components that are satisfying the relations in* Ω*:* $S \subseteq \mathcal{P}(C) \cap \Omega(C)$*. The instance of*

> *such a combinatorial problem can be solved by finding the element* s^* *such that*
>
> $$s^* = \underset{s \in \{\mathcal{P}(C) \cap \Omega(C)\}}{\arg\min} J(s) \tag{2.5}$$

In the following a short side note on problem complexity is given, which is a worthwhile topic to discuss for solving COPs.

A side note: Problem complexity

In context of solving an optimisation problem, the complexity of a problem should be considered. Therefore, in following a brief side note on this topic will be made:

If a computational problem is solved by an algorithm, it is interesting to determine its efficiency to make it comparable to other algorithms. There are two aspects that affect an algorithm's efficiency: (i) the amount of time that is required to execute the algorithm and (ii) the memory space it consumes. However, as Garey and Johnson already emphasised in 1979:

> "Efficiency concentrates primarily on time as it usually dominated over other computing resources." [46]

Therefore, normally just the required time for an algorithm is considered. This is most often done by determining the *time complexity function* for an algorithm, which measures the time requirement of the algorithm. Due to the fact that the execution time of an algorithm depends heavily on the input, generally the *worst-case time complexity*, denoted as $T(n)$, is used that measures the maximum amount of time needed by the algorithm to solve a problem instance of size n.

The general *execution time* of an algorithm is an estimate of the number of operations performed on a particular number of input values. This can be determined for an algorithm by counting primitive operations, such as arithmetic operations, assignments etc. However, usually the *order of growth* or *rate of growth* of an algorithm is a more interesting concept of complexity analysis, i.e. instead of considering the exact number of

arithmetic operations the rate of growth of the complexity function is considered as the size of the problem n increases.

One of the most popular notations in the complexity analysis that is used to compare complexity functions of two algorithms is the *Big-O notation*. It is defined as follows:

> **Definition 8 (Big O notation)** *For the two non-negative functions $f(n)$ and $g(n)$ of the non-negative variable n it can be stated that $f(n)$ has the order of $g(n)$, if the following equation is fulfilled:*
>
> $$f(n) = \mathcal{O}(g(n)) \Leftrightarrow \exists n_0 \in \mathbb{N} \wedge c > 0, \forall n \geq n_0 : \ |f(n)| \leq c|g(n)| \quad (2.6)$$
>
> *It can also be stated that $g(x)$ is an* asymptotic upper bound *of the function $f(x)$.*

If the execution time of an algorithm is bounded by a polynomial in the size of the input n for the algorithm, i.e. $T(n) = \mathcal{O}(n^k)$ for a constant k, the algorithm is called *polynomial*. Algorithms that cannot be bounded that way are called at least *exponential*. Problems that cannot be solved within polynomial-time complexity are called *intractable*.

Algorithms for Solving COPs

Many combinatorial optimisation problems are hard to solve. The problem of 'hardness' is part of the research area of computational complexity theory, which deals with the classification of complexity. The set of problems that are hard to solve are collected in the complexity class of $\mathcal{NP}$-hard. This class contains problems of which the worst-case time complexity of a problem's instance is at least exponential.

A famous example of a combinatorial problem, which is $\mathcal{NP}$-hard, is the Travelling Salesman Problem (TSP) which is studied quite often in operations research and theoretical computer science. The goal of the salesman is to find the shortest possible tour that includes all cities, each exactly once, from a given list, i.e. finding a Hamiltonian circuit[4]. The

[4]named after Sir William Rowan Hamilton (1805 - 1865), Irish physicist, astronomer, and mathematician

frequent use of the TSP in the context of testing algorithms for COPs derives from the fact that the problem is easy to understand so that the researcher can concentrate on the algorithm that should be applied rather than being distracted by complicated details of the problem. Furthermore, a lot of algorithms, that have been tested on the TSP and that have been proven to be efficient, showed also good results on various related problems.

In contrast to small-scale problems for which exact methods are a good choice, for large instances approximation methods are better suited to obtain near optimal results at low computational costs. In the area of mathematics and computer science often heuristics are used to find feasible solutions for (multi-objective) COP. Therefore, in the following the terms 'heuristic methods' and 'metaheuristic' are discussed.

Heuristic Methods *Heuristic methods*[5] in computer science are used to provide good solutions for a specific problem at low computational costs and within a reasonable computation time. While traditional algorithms try to guarantee optimal solutions within an optimal runtime, heuristic methods reduce one or both of these requirements. Instead heuristic methods try to find a good trade-off between computational costs and accuracy of the solution of a problem so that a 'good' solution can be provided within a reasonable time without promising an optimal solution. To reach this goal techniques, such as estimation, trial-and-error methods, 'rule of the thumb', educated guess, intuitive judgement or common sense, are applied.

Heuristics can be divided into: *constructive algorithms* and *iterative improvement algorithms*.

1. **Improvement Heuristics** Start with 'some' solution and continue to change it into a better one as long as possible.

2. **Construction Heuristics** Start building a solution from scratch.

[5] Heuristic, from greek *heuriskein* "to discover" or "to find"

Definition 9 (Construction Heuristics) *A* construction algorithm *builds a solution to an instance of a COP, as defined in Definition 7, in an incremental way. Starting with an empty partial solution $x_0 = \emptyset$, step-by-step a solution component $c \in C$ is added to the partial solution until a complete solution $s \in S$ is generated. Listing 2.1 shows python-inspired pseudo-code of a generic construction algorithm.*

```
def generic_construction_algorithm:
    s = []
    while (s not in complete_solutions) or abort_criterion:
        c = select_component()
        s.append(c)

    return s
```

Listing 2.1: Generic construction algorithm

The starting point is the initial empty partial solution s. At each step, the function select_component *returns a solution component c, which is then added to the partial solution s. Afterwards, it is checked if the solution is complete or an abort criterion is met such as the current partial solution cannot be converted to a complete feasible solution. Finally the function returns either a feasible solution $s \in S$ or a partial solution $s \notin S$, if the abort criterion is met.*

Due to the fact that constructive algorithms are, on the one hand, quite fast, but on the other hand, their solutions are not of high quality, constructive algorithms are often used to generate an initial solution on which afterwards local search methods are applied.

A graphical tool for visualisation and analysis, which can be used to represent the sequence of decisions in a construction or decision process, is the *construction graph*, firstly introduced by Dorigo et al. [47]. Formally

a construction graph can be defined as follows:

Definition 10 (Construction graph) *A given COP* $< C, \Omega, J >$ *can be represented by a completely connected finite directed graph, the so called* construction graph*:*

$$G_C = (C, L) \tag{2.7}$$

with

- $C = \{c_1, c_2, ..., c_n\}$, $n \in \mathbb{N}$, *as finite set of nodes*
- $L = \{l_{c_i c_j} | (c_i, c_j) \in \tilde{C}\}$ *as finite set of edges that are connecting the nodes, the so called connections or transitions*
- $\tilde{C} \subseteq C \times C$ *set of possible connections*

A solution of the COP can be expressed as feasible path on the Graph G_C*.*

For example, in the Travelling Salesman Problem (TSP)[6], one of the most often studied problems in the area of COPs, is C the set of cities, L the set of edges connecting the cities and a solution $s \in S$ is an Hamiltonian circuit (see figure 2.9).

Metaheuristic

A *metaheuristic* is a set of algorithmic concepts that can be applied to a wide set of various problems. For that reason a metaheuristic can be seen as 'general-purpose heuristic'. Normally, only a very few modifications have to be made to adapt the general algorithmic framework to a specific problem. A metaheuristic optimises a problem by iteratively improving a candidate solution with regard to a measure of quality. Local search and other high level strategies are often used to find global near-optimal

[6] In the TSP a salesman is given a list with cities and their pairwise distances. Based on this the salesman is supposed to find the shortest possible tour that visits all cities, but each city exactly once. The TSP is also often used as benchmark for different optimisation methods.

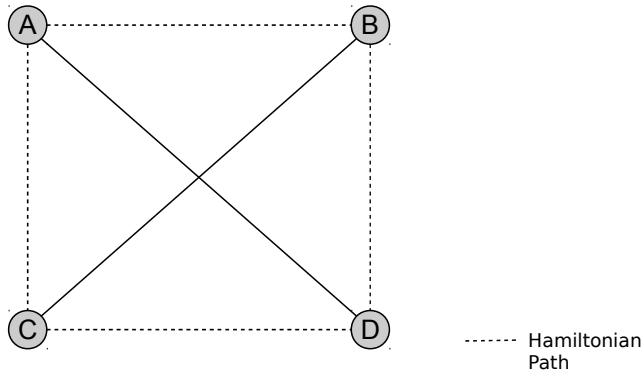


Figure 2.9: Four cities Travelling Salesman Problem (TSP)

solutions instead of local optimum solutions. The term 'meta-heuristic' was introduced by Glover to describe that a heuristic is "superimposed on another heuristic" [48].

There are several famous metaheuristics that are used to find good solutions to difficult, but relevant combinatorial optimisation problems. Well-known metaheuristics are: iterated local search [49], variable neighbourhood search [50], Simulated Annealing (SA) [51, 52], shortest path finding algorithm A^* [53], Tabu Search (TS) [54, 55], Ant Colony Optimisation (ACO) [56, 57] etc. A sub-class of those metaheuristics, the nature-inspired approaches, will be discussed in more detail later in this chapter.

2.3 Multi-objective Combinatorial Optimisation Problems

In the real-world optimisation problems are often dependent on multiple, conflicting, objectives. Traditional approaches often try to combine all objectives in a single objective function that can be optimised regarding the resulting single cost value. However, due to the fact that the different objectives have often varying representations and meanings, this combined objective function is a rather artificial construct that may not consider the overall problem adequately.

Another option, as used here, is to consider this sort of problem as MCOP, i.e. a combinatorial optimisation problem in which two or more (conflicting) objectives that are subject to certain constraints are simultaneously optimised. As in COPs the set of feasible solutions is finite and they own a combinatorial property. An example of an MCOP would be to optimise a travel in terms of minimising the cost and at the same time the duration of journey.

Formally an MCOP can be defined as follows (cf. [58]):

Definition 11 (Multi-objective combinatorial optimisation problem) *A multi-objective combinatorial optimisation problem maps a tuple of parameters (decision variables) to a tuple of objectives. This can be formally denoted as:*

$$\vec{x} \in X \subseteq \mathbb{R}^k, \vec{f} \in Y \subseteq \mathbb{R}^n, n \geq 2$$
$$\arg\min_{\vec{x}} \vec{f}(\vec{x}) \tag{2.8}$$

with

- $X \subseteq \mathbb{R}^k$*, the multi-dimensional space of feasible solutions constituted by the decision variables of* $\vec{x}$*, the so called* k*-dimensional* decision space
- $Y \subseteq \mathbb{R}^n$*, the multi-dimensional space that is constituted by the objective functions from* $\vec{f}$*, the so called* n*-dimensional* objective space
- $\vec{x} = (x_1, x_2, ..., x_k) \in X$*, the vector of decision variables, the so called* decision vector
- $\vec{f} = (f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x})) \in Y$*, the vector of objective functions, the so called* objective vector

The main goal of an MCOP is to optimise the objective vector $\vec{f}$*.*

Figure 2.10 depicts an example of the mapping between the k-dimensional decision vector and the n-dimensional objective vector.

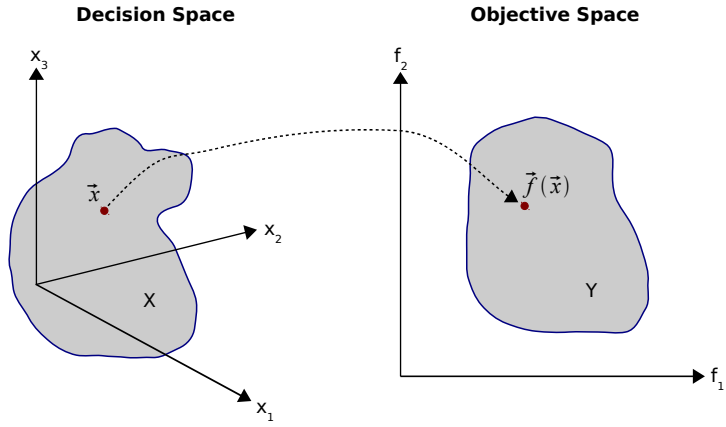


Figure 2.10: Mapping from decision space to objective space

In contrast to single-objective COP, for an MCOP there is typically no single best solution, but instead an MCOP's solution is composed of a set of solutions that represent the best possible trade-off among those objectives. For such a solution it is necessary to find a quality criterion that takes all objective functions into account to find a compromise among those objectives. Such a criterion, the Pareto Optimality, is discussed in the following subsection.

Pareto Optimality

A notion of optimality that is most commonly adopted for multi-dimensional objective-spaces is *Pareto optimality*, named after Vilfredo Pareto[7].

In the following a few definitions are introduced that are helpful in the context of Pareto optimality:

Definition 12 (Pareto domination) *An objective vector* $\vec{f}^* \in Y$ *dominates another* $\vec{f} \in Y$ *in terms of* Pareto's domination*, in short* $\vec{f}^* \succ \vec{f}$*, if it is better for one criterion and better or equal for the*

[7] Vilfredo Pareto (1848 - 1923), Italian engineer, sociologist, economist, and philosopher

others. Formally this can be denoted as follows:

$$\begin{aligned} \forall i \in \{1, 2, ..., n\} : f_i^* \leq f_i \\ \wedge \exists j \in \{1, 2, ..., n\} : f_j^* < f_j \end{aligned} \tag{2.9}$$

Accordingly, it can be stated that the decision vector $\vec{x}^* \in X$ dominates *the decision vector* $\vec{x} \in X$*, in short* $\vec{x}^* \succ \vec{x}$*, if* $f(\vec{x}^*) \succ f(\vec{x})$*.*

Definition 13 (Pareto optimal) *A decision vector* $\vec{x}^* \in X$ *is called* Pareto-optimal*,* efficient *or* non-dominated*, if and only if, there is no vector* $\vec{x} \in X$ *such that* $\vec{x}$ *dominates* $\vec{x}^*$*.*

Definition 14 (Weakly Pareto optimal) *A decision vector* $x^* \in X$ *is* weakly Pareto optimal *if there is no other decision vector* $\vec{x} \in X$ *for which all the components are better. Formally this can be denoted as:*

$$\nexists \vec{x} \in X, \forall i \in \{1, 2, ..., n\} : f_i(\vec{x}) < f_i(\vec{x}^*) \quad \vec{x}^* \in X \tag{2.10}$$

Definition 15 (Pareto set) *The set of all Pareto-optimal solutions, which is a subset of the decision space, is named* Pareto set *or* efficient set*. The Pareto set can be formally denoted as:*

$$\begin{aligned} X^* \subseteq X \\ X^* := \{\vec{x}^* \in X | \nexists \vec{x} \in X : \vec{x} \succ \vec{x}^*\} \end{aligned} \tag{2.11}$$

Definition 16 (Pareto front) *The set of elements in the objective space that is corresponding to the set of elements of the Pareto set in the decision space is named* Pareto front*,* Pareto frontier *or* Non-dominated set*. Formally denoted as follows:*

$$Y^* = f(X^*) \subseteq Y \tag{2.12}$$

Figure 2.11 depicts an example of the mapping between Pareto set in the decision space to Pareto front in the objective space.

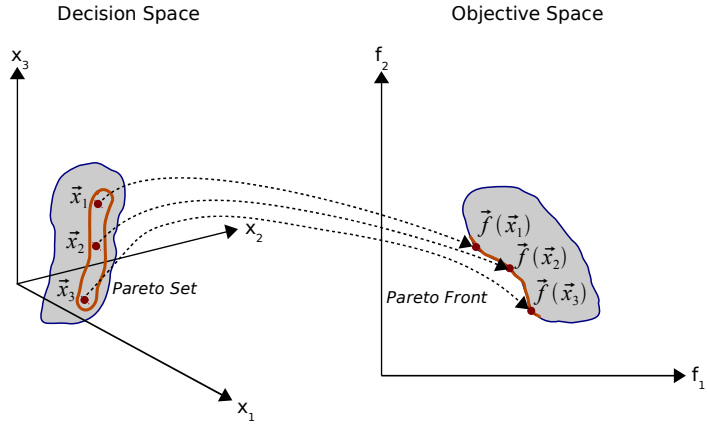


Figure 2.11: Mapping from Pareto set in the decision space to Pareto front in the objective space

In the case, in which an MCOP has multiple competing objective functions, there is always more than one Pareto optimal solution[8]. Therefore, solving an MCOP in terms of Pareto optimality, means finding the Pareto set. Although, the Pareto set contains multiple solutions, in general one is only interested in exactly one solution as in single-objective COPs. How to select a single solution from the Pareto set will be discussed in the following.

2.3 Decision Making

Due to the fact that each element of the Pareto front is a valid solution, as compromise between all objective functions, the question arises which single solution to choose (see figure 2.12)?

In the simplest case automatically a simple algorithm is applied that is capable of finding the final single solution. This approach is also referred

[8]Note: correlating multiple-objective functions are resulting in a single Pareto optimal solution, which makes them identical to COPs with a single-objective function, which are not considered here

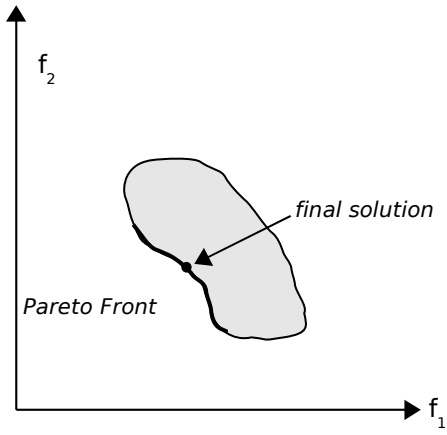


Figure 2.12: Finding the final solution

to as *no-preference method* because the algorithm can not be influenced by any given preferences. For more details of this trivial approach see [59].

For more complex approaches to select a final single solution from the set of Pareto optimal solutions some sort of additional information is required that is not part of the objective functions. This can be achieved by adding an additional component, the so called Decision Maker (DM), to the process, which can claim desired preferences to find the best solution regarding those. The DM is assumed to know the problem and must be able to give some input, in terms of additional information to the problem regarding objectives and/or solutions, so that a preferred solution can be found.

Basically, there are three possibilities to incorporate the decision maker's preferences into the optimisation process (cf. [4, 10, 58, 60]) (see figure 2.13):

1. In *a priori approaches* the decision maker specifies the relative importance of certain criteria before the optimisation process is started. Only a single optimisation is required to obtain a single solution, which makes this approach really fast. However, in this case the time for the modelling has to be taken into account.

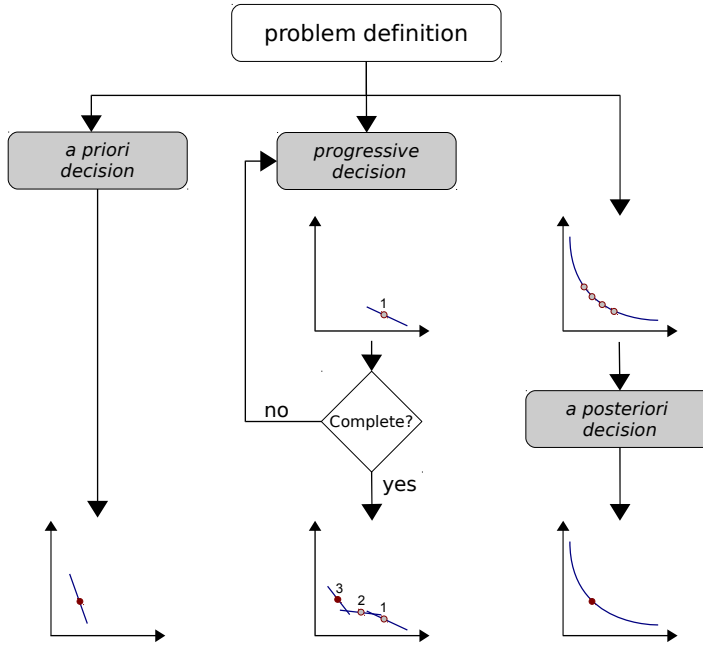


Figure 2.13: Three approaches to incorporate the DM in the decision making process (cf. [4])

At the worst, the decision maker is not satisfied with the solution found then the whole modelling and optimisation process needs to be started again.

2. In *progressive approaches* the decision maker expresses the desired preferences during the optimisation process. The expressed preferences are immediately used to guide the search. In this case, the desired preferences can be accurately taken into account, but a lot of interaction is required during the optimisation process.

3. In *a posteriori approaches* the decision maker obtains a set of (non-dominated) solutions generated by a chosen optimisation method and then, chooses the final solution that is most suitable. An advantage of this approach is that the time-consuming modelling phase of

preferences before starting the optimisation process can be omitted. However, those sort of approaches require a lot of computational effort.

Depending on the specific problem one of the presented approaches should be carefully chosen[9]. For MCOPs mostly multi-objective metaheuristics are traditionally used so that *a posteriori* approaches are often applied.

Multi-Criteria Decision Making

A research area that addresses the assistance of one or multiple decision makers in taking an optimal decision, i.e. finding an optimal solution, is Multi-Criteria Decision Making (MCDM). The basic idea of MCDM models is to rank the performance of a finite set of alternatives, the so called decision criteria, i.e. finding an optimal solution among existing solutions.

A comprehensive overview about this topic is given by Triantaphyloou [61] who is presenting multi-criteria decision making methods as well as a comparative study of those models. Due to the fact that MCDM is one of the fastest growing research areas that is touching various application areas, there is a vast number of researches available, proposing several MCDM models. It is clear that not all MCDM models can be presented in depth in this thesis. However, a short overview about some classical MCDM approaches that are widely used in practice is given in the following, focussing mainly on *a posteriori* approaches:

Weighted Sum One of the most commonly used MCDM models is the Weighted Sum Model (WSM), originally proposed by Fishburn [61, 62]. In this model a weighting coefficient is assigned to each objective function and then, the corresponding weighted sum of objectives is minimised. For a decision problem the best solution satisfies the following equation:

$$A^*_{WSM-score} = \arg\min_i \sum_{j=1}^{n} w_j f_{ij}, \quad i = 1, 2, 3, ..., m \tag{2.13}$$

where

[9] In some cases also multiple approaches can be chosen to form a hybrid approach.

- $A^*_{WSM-score}$ is the alternative with the best WSM score
- n number of decision criteria
- m number of alternatives
- w_j relative weight of importance of the j^{th} criterion with $0 \leq w_j \leq 1$; $\sum_{j=1}^{n} w_j = 1$
- f_{ij} objective of the j^{th} criterion and the i^{th} alternative

In the case of a single-dimension, i.e. all units of measurements are identical, (e.g. km, s, \$), the WSM method can be simply applied. However, if a multi-dimensional problem should be solved, in which different units are used, WSM cannot be applied because the additivity utility assumption will be violated otherwise[10].

Weighted Product Another method, which is very similar to the WSM, is the Weighted Product Model (WPM). The main difference between both models is that instead of an addition, as used in WSM, a multiplication is used in WPM. The first reference to this model can be found in Bridgman [63] as well as Miller and Starr [64].

$$P(A_i) = \prod_{j=1}^{n} (f_{ij})^{w_j}, \quad i = 1, 2, 3, ..., m \tag{2.14}$$

where

- $P(A_i)$ the (absolute) performance value of alternative A_i considering all criteria
- n number of criteria
- m number of alternatives
- w_j weight of importance of the j^{th} criterion with $0 \leq w_j \leq 1$; $\sum_{j=1}^{n} w_j = 1$
- f_{ij} objective function of the j^{th} criterion and the i^{th} alternative

[10] The additive utility assumptions implies that the total value of each alternative is equal to the sum of the products given in equation 2.13

In contrast to WSM, WPM is dimensionless, i.e. any units of measure are eliminated due to the structure of the formula. For that reason, WPMs can be applied for single and multidimensional decision problems.

Weighted Metrics The goal of Weighted Metrics Model (WMM) [59] is to minimise the distance between the ideal criterion vector z^*, whose entries consists of the optima of the objective functions, and the solutions on the criteria space. If the ideal criterion vector is from the admissible solutions space, the best solution will be the ideal criterion vector itself. However, this is only possible if there are only non-conflicting criteria. The weighted approach is also referred to as *comprise programming* [65]. A multi-objective problem with m objectives can be transformed to a single objective problem by applying the following L_p-metric:

$$s_p^* = \arg\min_{x \in S} \left(\sum_{i=1}^{m} w_i \left| f_i(x) - z_i^* \right|^p \right)^{1/p} \tag{2.15}$$

- m number of objectives
- $1 \leq p < \infty$; Commonly used values for p are:
  $p = 1$ (*Manhatten*), $p = 2$ (*Euclidean*) and $p = \infty$ (*Tchebycheff*)
- S is the feasible decision variable space
- $z^* \in \mathbb{R}^n$ ideal objective vector, obtained by minimising each objective z_i^* separately; z^* is not a feasible vector because the objectives are conflicting.
- w_i as the i^{th} component of the weight vector
  with $0 \leq w_j \leq 1$; $\sum_{j=1}^{n} w_j = 1$
- f_i objective function of the i^{th} criterion

The solution of 2.15 is Pareto optimal, if all weights are positive or the solution is unique.

For $p = \infty$ the equation 2.15 can also be written as:

$$s_\infty^* = \arg\min_{x \in S} \max_{i=1,2,3,..,m} \left[w_i \left| f_i(x) - z_i^* \right| \right] \tag{2.16}$$

This method is also referred to as Weighted Tchebycheff Metric (WTM). The solution of 2.16 is weakly Pareto optimal for positive weights. Furthermore, there is at least one Pareto optimal solution for 2.16.

Achievement Scalarizing Function Approach: Wierzbicki [66, 67] introduced the Achievement Scalarizing Function Approach (ASFA) that utilises a special type of scalarizing functions, the so called *achievement (scalarizing) functions*. This sort of functions are of the form $s_{\bar{z}} : Z \to \mathbb{R}$, where $\bar{z} \in \mathbb{R}^n$ is an arbitrary reference point. Due to the fact that Z is not known explicitly, instead the function $s_{\bar{z}}(f(\vec{x}))$ with $\vec{x} \in S$ is minimised. The idea of the approach is to project the reference point $\bar{z}$ consisting of desirable aspiration levels onto the set of Pareto optimal solutions.

Achievement functions can be formulated in different ways, as an example the following function is considered:

$$s(f(\vec{x})) = \max_{i=1,\dots,n} [w_i(f_i(\vec{x}) - \bar{z}_i)] + \rho \sum_{i=1}^{n} w_i(f_i(\vec{x}) - \bar{z}_i) \tag{2.17}$$

where

- w_i fixed normalising factor
- ρ augmentation multiplier (small positive scalar)

An advantage of this approach is that the decision maker can obtain different Pareto optimal solutions by just moving the reference point.

ϵ-Constraint Method: In the ϵ-Constraint Method (eCM), originally proposed by Haimes et al. [68], a multi-objective problem is transformed into several single-objective problems with constraints. In each single-objective problem one objective function is optimised as follows:

$$\begin{aligned} \arg\min \quad & f_l(x), && l \in \{1, 2, 3, ..., k\} \\ & f_j \leq \epsilon_j, \quad \forall j = 1, 2, 3, ..., k \quad j \neq l, \\ & x \in S \end{aligned} \tag{2.18}$$

with

- ϵ_j upper bounds for the objective l

- S feasible region

To ensure Pareto optimality, either k different problems have to be solved or a unique solution needs to be found. Due to the fact that uniqueness of solutions is not easy to verify, Chankong and Haimes [69] propose systematic ways of perturbing the upper bounds to obtain different Pareto optimal solutions.

Multi-Criteria Decision Summary: In table 2.1 a summary of the described MCDM methods is given.

	WSM$^{\clubsuit}$	**WPM**	**WMM**$^{\heartsuit}$	**ASFA**$^{\diamondsuit}$	**eCM**$^{\spadesuit}$
a priori method	√	√		√	√
a posteriori method	√	√	√	√	√
any Pareto optimal solution is found			(√)	√	√
solutions always Pareto optimal	(√)		(√)	(√)	(√)
type of preference					
weights	√	√			
bounds					√
reference point				√	

Table 2.1: Summary of described MCDM methods (cf. [10, p. 22])

$\clubsuit$
- $\oplus$ solution is Pareto optimal, if it is unique or all weights $w_i > 0$
- $\ominus$ small change in weights may change solution dramatically
- $\ominus$ evenly distributed weights do not produce evenly distributed representation of Pareto optimal set

$\spadesuit$
- $\oplus$ a unique solution is Pareto optimal
- $\oplus$ a solution x^* is Pareto optimal if $\epsilon_j = f(x^*)$ $(i = 1, .., k \wedge j \neq l)$ for all objectives to be minimised
- $\ominus$ it may be difficult to specify the upper bounds

$\heartsuit$
- $\oplus$ solution is Pareto optimal, if it is unique or all weights are positive
- $\ominus$ not all Pareto optimal solutions may be found

$\diamondsuit$
- $\oplus$ if lexicographical ordering is used (see [70]) Pareto optimality can be guaranteed

2.4 Ant Colony Optimisation

In the following, Biologically-inspired Algorithms (BIAs) that can be used for solving COPs will be discussed with a special focus on the ACO metaheuristic. Therefore, in the first step, a short introduction to the idea of using nature inspired algorithms to solve COPs will be given and the most prominent examples of BIAs will be presented. Afterwards, the special case of Ant Colony Optimisation (ACO) will be discussed in depth, starting from its natural origin to the derivation of artificial ants to the most famous ACO algorithms and their application areas.

2.4 Biologically-inspired Algorithms for Optimisation Problems

Apart from successful metaheuristics, such as Tabu Search (TS) [54, 55], Iterated Local Search (ILS) [49, 71, 72] and Simulated Annealing (SA) [51, 52], there is a class of metaheuristics that is inspired by the studying of life and living organisms in nature, the so called Biologically-inspired Algorithms (BIAs) [73, 74], in short *bio-inspired algorithms*.

Due to the wide variety of BIAs it is hard to find a strict classification of those approaches. Besides, there are some approaches that combine different BIAs in hybrid approaches. Nevertheless, in figure 2.14 a basic taxonomy of BIAs is depicted that covers the most important approaches.

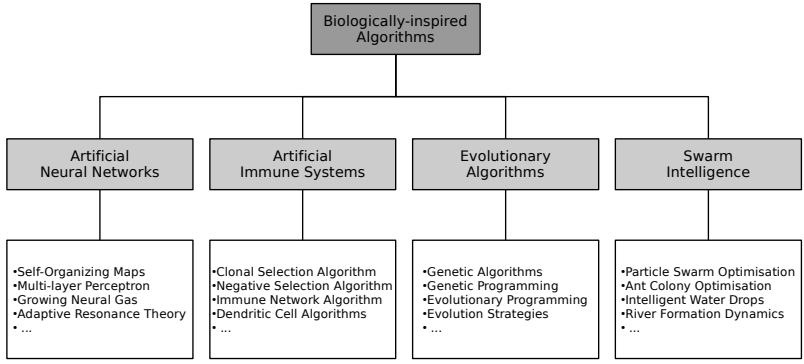


Figure 2.14: Taxonomy of biologically-inspired algorithms

It is clear that this thesis cannot discuss all BIAs in depth; however, a

short overview about the different approaches is provided in the following:

Artificial Neural Networks: Artificial Neural Networks (ANNs) in the area of BIAs imitate the behaviour of networks or circuits of biological neurons as can be found in the nervous system of humans and animals. This is achieved by composing and interconnecting artificial neurons, i.e. algorithmic functions that simulate the properties of their biological ancestors, to a network. While biological neurons are highly complex, artificial neurons are simplified by focussing on the information processing part that is relevant in this context. Today, most of the used ANNs are making strong use of control theory, statistical estimation as well as classification optimisation. Important approaches in the area of ANNs are self-organising maps, multi-layer perceptrons, growing neural gas and adaptive resonance theory. More details about ANNs can be found in [75, 76].

Artificial Immune Systems: Artificial Immune Systems (AISs) are BIAs that are inspired by the principles and processes observed in the immune system of vertebrates. Basically, an immune system is in the continuous process of adapting antibodies to the antigens that should be combated. From the computational point of view the two main characteristics *immune learning* and *memory* are significant. The most important approaches in the area of AISs are clonal selection, negative Selection, dendritic cell algorithms and immune network algorithms (more details can be found in [77]).

Evolutionary Algorithms: Evolutionary Algorithms (EAs) are based on the theory of evolution, i.e. the survival of the fittest. EAs are inspired by mechanisms of biological evolution, utilising mechanisms such as *reproduction*, *mutation*, *recombination* and *selection*. Candidate solutions are mapped to individuals of a population and a fitness function determines the environment the population lives in. Due to the repetitive application of the fitness function an evolution of the population takes place. The most important Evolutionary Algorithm (EA) paradigms are: Evolutionary Programming (EP) [78–80], Evolution Strategies (ES) [81–83],

Genetic Programming (GP) [84–87] and Genetic Algorithm (GA) [88–90].

Swarm Intelligence: Swarm Intelligence (SI)[11] [92], i.e. the decentralised, but self-organising and collective behaviour of autonomous individuals acting by following simple rules to form a powerful community, can be observed in bird flocks, animal herds, ant colonies, bacterial growth and fish schooling. Based on studies of social animals and insects several computational models of SI, so called Swarm Intelligence Algorithm (SIA), were developed that imitate the observed swarm behaviour to solve complex problems. Various approaches make use of the concept of SI. The most famous representatives of this sort of algorithms are Particle Swarm Optimisation (PSO) [93–95] and Ant Colony Optimisation (ACO) [11, 56, 96]. Other interesting approaches in this area are River Formation Dynamics (RFD) [97], Stochastic Diffusion Search (SDS) [98], Intelligent Water Drops (IWD) [99] and Gravitational Search Algorithm (GSA) [100].

BIAs, such as GA, EP, PSO and ACO, share some common characteristics: all approaches work on a set of feasible solutions for a particular problem. The solutions need to be evaluated regarding a fitness value to obtain some kind of quality measure. Based on this measure a selection mechanism is used to select the solution with the best fitness value. The selected solution is subsequently used to generate new solutions by varying them.

BIAs have been applied in the last few years to several combinatorial and continuous optimisation problems in the area of scheduling, robotics, communication networks etc. Particularly SI approaches were successfully applied to the area of communication networks [101].

In the following a special case of BIA, the Ant Colony Optimisation (ACO) algorithm, will be discussed in detail as it is the foundation of the research conducted in this thesis.

[11] The term 'swarm intelligence' was firstly introduced by Beni and Wang [91] in the area of computer science in the context of cellular robotic systems.

2.4 Ant Colony Optimisation

Ant Colony Optimisation (ACO) was inspired by the observation of the behaviour of real ants. Thus, in the following, first, its biological origin will be discussed. Afterwards, the formal basis of ACO will be explained as well as some applications of ACOs will be presented.

Biological Origins of Ant Colony Optimisation

Ants are social insects that live together in colonies. The colonies are highly organised, i.e. ants behave collaboratively in a complex way to solve difficult tasks, a single ant, as a simple individual, could not solve. One of the interesting and important behaviours that can be observed in ant colonies is the *foraging*, particularly, how ants can find the shortest path between the nest and food sources.

One of the first researchers who investigated the social behaviour of insects was the French zoologist Pierre-Paul Grassé[12]. During his research trips to Africa he focused on studying termites and became one of the experts on those insects. He discovered that the two studied species of termites (*Cubitermes sp.* and *Bellicosotermes natalensis*) react to 'significant stimuli' [102] while rebuilding a nest. The response to stimuli leads to reactions by the termites that "[..] explain(s) the synchronisation of various individual tasks". For the indirect communication, which Grassé describes as "[..] stimulation of workers by the very performances they have achieved", he introduced the term *stigmergy*, which was later used for similar phenomena by other social insects.

According to Dorigo et al. [12] stigmergy differs from other forms of communication in two main characteristics:

- Stigmergy is an indirect, non-symbolic form of communication mediated by the environment → insects exchange information by modifying their environment
- Stigmergic information is local → it can only be accessed by those insects that visit the locus in which it was re-

[12] Pierre-Paul Grassé (1895 - 1985), editor of the 28-volume "Traite de Zoologie", chair of evolutionary biology at Sorbonne University and ex-president of the French Academy of Sciences.

leased (or its immediate neighbourhood)

In many ant colonies of different ant species stigmergy can be observed: ants that are moving between the nest and food sources are depositing a chemical substance, so called *pheromone*, on the ground; thus, modifying the environment. Ants that are perceiving the deposited pheromone are influenced by its presence. While without pheromone trails the movements of the ants is essentially random, ants that are perceiving a pheromone trail are tending to follow the paths with higher pheromone concentrations. As a result, the ants can effectively make use of the shortest paths between the nest and the food sources.

The pheromone laying is complemented in the natural environment by *evaporation*, i.e. the decaying of the pheromone after some time. The evaporation of pheromone depends on several factors such as the chemical composition of the terrain, the ant species, the weather conditions and the amount of pheromone deposited etc. However, pheromone can stay for several hours or even months depending on the given circumstances. As consequence of evaporation less promising paths that are not used frequently will disappear over time.

The self-organising behaviour of ants is based on *autocatalysis*, i.e. positive feedback is exploited to control the overall behaviour of the ant colony. The risk generally involved with autocatalysis, i.e. premature convergence (*stagnation*), is mitigated by evaporation and the additional stochastic choice of paths by the ants. As a result, the exploration of new paths is never ceased.

Basically, the *trail-laying* and *trail-following* behaviour can be summarised as follows:

- When an ant moves it lays a pheromone trail
- Each ant moves as follows:
  - when no pheromone trail is perceived → random walking
  - when an ant perceives a pheromone trail → random walk, but stochastic decision is biased by the amount of pheromone

for the trail, i.e. higher likelihood to follow trails with high pheromone concentration

- Pheromone evaporates over time → Unused paths will vanish.

Several researchers investigated the pheromones laying and following patterns of ants experimentally and quantifying it to gain deeper insights. Two of the most famous exerimental works in this research area are discussed in the following.

Experiments with two bridges: To investigate the ants' exploratory trails Deneubourg et al. [5] setup an experiment with a "Diamond-shaped bridge" that connects the nest with an unexplored area, the so called arena, via two bridges of the same length. The bridges provide the ants a binary choice, i.e. between the upper or lower bridge at each decision point 1 and 2 (see figure 2.15).

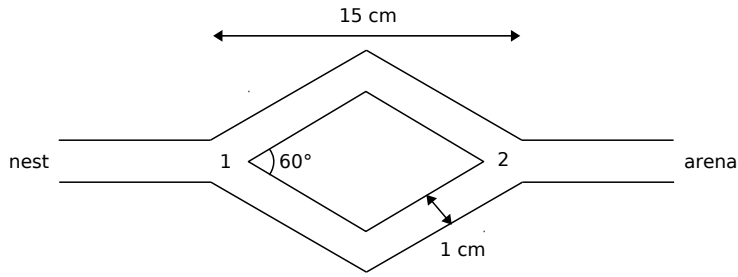


Figure 2.15: Experimental setup of the diamond-shaped bridge experiment (cf. [5])

The ants had a free choice using the bridges to get from the nest to the arena and *vice versa*. The ratio of the ants using the upper and the lower bridge were observed over time. For the evaluation of the experiment two simplifications were made: first, the evaporation of pheromone was ignored because the experiments were run shorter than the mean lifetime of the pheromone; and second, an average amount of pheromone that is laid by the ants was assumed, instead of considering the individual

amount of pheromone.

The main result of the experiment observations was that, while initially both branches were chosen equally, after a certain time, the ants preferred one of the branches. This result can be explained as follows: at the beginning there are no pheromones on the two bridges so that the ants select their way randomly. However, each ant that is passing one of the bridges deposits some pheromone on the used bridge. Due to random fluctuation a few ants will select the same bridge so that the density of pheromone is increased on this bridge. Consequently, the following ants will be influenced by the deposited pheromone so that the probability of choosing the path with the higher pheromone concentration is increased. Finally, the described mechanism leads to a convergence of using one of the bridges, breaking the symmetry. In multiple experiments Deneubourg et al. found out that each of the two bridges was used in about 50% of the cases.

A similar experiment was conducted by Goss et al. [6] in which ants were given access to a food source that was connected by a bridge consisting of two identical modules, each having two branches of different length providing the ants at each module a choice between a longer and a short path at each decision point 1 and 2 (see figure 2.16).

The results of the experiment show in this case, after a while, the ants chose the shortest path between the nest and the food source. The results of the second experiment can be explained as follows: similar to the previous experiment the ants have to choose at each decision point one of the branches of a bridge, i.e. either the short or the long branch. When at the beginning there is no pheromone on the bridges the decision is taken randomly – as in the first experiment, one of the path may be preferred over time. However, in the second experiment another factor accelerates the creation of the shortest path: ants that select the shortest branch will first reach the food source and then head home to the nest. When the ant reaches a decision point, its decision is biased by the pheromone concentration, which is obviously higher on the short branch each time. Due to the more frequently used short branches the pheromone concentration on the short branches increased rapidly, making the majority of following ants use the shortest path between the nest and the food source. Addi-

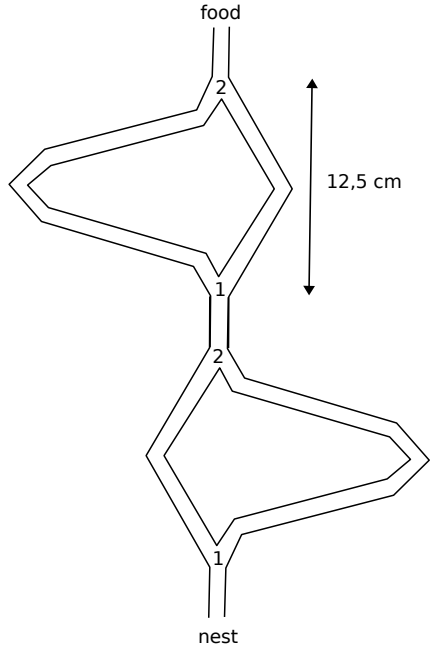


Figure 2.16: Experimental setup of the bridge experiment with two modules in which each module has two branches of different lengths (cf. [6])

tionally to the autocatalytic mechanism observed in the first experiment, in the second experiment an *implicit evaluation of solutions* can be found, i.e. shorter paths will be completed earlier and thus, these paths are reinforced quicker with higher pheromone concentration. In comparison to the first experiment, the initial random fluctuation is reduced.

Stochastic Ant Colony Model: Goss et al. [5, 6] proposed a simple stochastic model that describes the dynamics of the observed ant colony behaviour in the double bridge experiments.[13]

As described in the experiment, the ants cross the bridges selecting at each decision point either the lower or the upper branch. An ant with a constant speed of v cm/s traverses the lower branch of length l_l in a

[13] As in the real experiment the evaporation of the pheromone was ignored for the stochastic model due to the fact that the mean lifetime of pheromone in the real experiment was $\simeq 30$ min.

time of $t_l = l_l/v$, while the ant requires $t_u = rt_l$ seconds with $r = \frac{l_l}{l_u}$ for traversing the upper branch.

$p_{ib}(t)$ is the probability that at the time of t an ant arriving at the decision point $i \in \{1, 2\}$ is selecting the branch $b \in \{u, l\}$, i.e. the upper or respectively lower branch. The total amount of pheromone at branch b at the time of t is $\varphi_{ib}(t)$, which is proportional to the number of ants that used the branch up to that time t.

The probability $p_{il}(t)$, i.e. of choosing the lower branch at the decision point i is given by

$$p_{il}(t) = \frac{(t_l + \varphi_{il}(t))^\alpha}{(t_l + \varphi_{il}(t))^\alpha + (t_u + \varphi_{iu}(t))^\alpha} \tag{2.19}$$

while, p_{iu}, the probability of choosing the upper branch at decision point i, is $p_{iu} = 1 - p_l$.

A comparison of Monte Carlo simulations of the model and real measured data showed that the results are corresponding. The suitability for the model described by equation (2.19) for the double bridge experiment was found in the case of the empirically fitted value of $\alpha \approx 2$ [6]. For two branches of equal length $r = 1$, the modelled results show the observed behaviour as in the real experiment that one of the branches is selected randomly after a while.

In the considered model, ants deposit pheromone on their forward as well as the backward paths. This turns out to be a necessary behaviour to converge to the shortest path between nest and food source.

2.4 From Real Ants to Artificial Ants

As the two experiments with real ants have shown, ant colonies have some kind of optimisation capability to find the shortest path between two locations based on probabilistic rules using local information. Inspired by this observation of real ant colonies, the Ant Colony Optimisation (ACO) *metaheuristic* was developed, which imitates the behaviour of real ant colonies by artificial ant colonies to solve combinatorial optimisation problems.

Based on the descriptions of Dorigo et al. and Cordon et al. [56, 57] the most important similarities and differences between real and artificial ants are discussed in the following.

Similarities Between Real and Artificial Ants

The main characteristics of real ant colonies that are leveraged by ACOs algorithms can be summarised as follows (cf. [56, 57]):

Colony of cooperating individuals: As in real ant colonies, ACO algorithms make use of artificial ant colonies, consisting of simple autonomous agents that are working cooperatively together to find a good solution for a specific problem. Though, a single artificial ant is capable of finding a feasible solution for a problem, just as a real ant can find a path between the nest and a food source, a high quality solution can be obtained due to the cooperation of all ants of the colony.

Pheromone trail and stigmergy: The communication between real as well as artificial ants is enabled by the use of stigmergy, i.e. by modifying certain aspects of the environment. While real ants are depositing a chemical substance called pheromone at places they have visited, artificial ants are modifying some kind of numeric information that is locally stored at the problem states they are visiting. This numeric information contains details about the current agents performance/history. Each ant that is visiting the same problem state can access, i.e. read and write, this additional state information. As analogy to real ants, this numeric information is also referred to as *artificial pheromone trail*.

In nature, pheromone evaporation leads to a slow decay of the pheromone over time and thus, the forgetting of paths found in the past. To alleviate the effect of past decisions in artificial ant colonies, a similar mechanism is often implemented to allow the exploration of new problem solutions.

Shortest path searching and local movement: Both, artificial as well as real ants share the common task of finding the shortest path, in terms of minimum costs, between a source and a destination. While

real ants walk through the surrounding adjacent terrain to find the shortest path between the nest and a food source, artificial ants iteratively construct a minimum cost solution by moving between adjacent states, starting from a start state they reach a final state. Due to the various application areas of artificial ants the terms 'state' and 'adjacent' differ from problem to problem.

Stochastic decision policy for state transition: In natural as well as in artificial ant colonies, the ants use a probabilistic decision policy to move from one to another adjacent state. In both cases, the probabilistic decision policy solely makes use of local information without any sort of look-ahead functionality to predict subsequent states. As a result, the probabilistic decision policy is completely local in space and time. For the artificial ants the probabilistic decision policy can be formally seen as function of the *a priori* defined information given by the problem's specifications and the *a posteriori* local modifications of the environment caused by the ants that passed the same environment.

What Makes Artificial Ants Different?

However, there are also some special characteristics of artificial ants that have no counterpart for real ants in nature (cf. [56, 57]).

Heuristic information: Compared with real ants, artificial ants can make use of heuristic information in addition to the pheromone information in the probabilistic decision process. The heuristic information is problem specific.

Discrete world: In contrast to real ants, artificial ants are moving in a discrete world. A move for an artificial ant is the application of the transition rule, which is the transition from one discrete problem state to another discrete problem state from a set of feasible adjacent states in the neighbourhood. The transition rule is a function of locally available pheromone, heuristic information, the ant's internal state and the problem constraints.

Internal state: Each artificial ant has an internal state, also referred to as memory, to store information about the visited states so far. The memory can be used for a couple of things such as building a feasible solution, evaluate a generated solution and to trace back a path from the current state to the initial state.

Quality of solution: Artificial ants can make use of weighting to prefer good solutions, i.e. the artificial ants are adjusting the amount of pheromone at certain states depending on the quality of a solution found.

Timing of pheromone deposit: In contrast to real ants, the laying of pheromones of artificial ants is problem dependent. For example, artificial ants often do not lay any pheromone until they constructed a feasible solution and then, they start to deposit pheromone on their way back to the origin.

Extra capabilities: Additionally to the highlighted characteristics, artificial ants can be enhanced with further features such as look-ahead, local optimisation, backtracking etc. However, the use of this sort of features is only worthwhile if the chosen feature matches the problem to be solved and the costs for the feature is lower than the cost for just doing the next move.

2.4 Ant Colony Optimisation Algorithms

Each ACO algorithm is based on one or more colonies of *artificial ants*, i.e. autonomous agents that are working together in a cooperative to solve a combinatorial problem (see section 2.3).

In the last few years several ACO algorithms have been proposed. Table 2.2 shows a selection of pioneering ACO algorithms (cf. [11, 96]).

In the following, the first ACO algorithm *Ant System* will be discussed as well as its direct descendants *Elitist Ant System* and *Rank-based Ant System*. After that the two most successful approaches $\mathcal{MAX} - \mathcal{MIN}$ Ant System and *Ant Colony System* are presented and the reasons why they outperform other ACO approaches are discussed. To have a common

ACO Algorithm	Authors	Year
Beam-ACO [103, 104]	Blum	2004
Population-based ACO [105, 106]	Guntsch, Middendorf	2002
Hyper-cube ACO (HC-ACO) [107, 108]	Blum et al.	2001
Best-Worst Ant System [109–111]	Cordón et al.	2000
ANTS [112]	Maniezzo	1999
Rank-based Ant System (AS_{rank}) [113]	Bullnheimer et al.	1997
Max-Min Ant System (MMAS) [114–116]	Stützle, Hoos	1996
Ant Colony System (ACS) [117–119]	Dorigo, Gambardella	1996
Ant-Q [120, 121]	Gamardella, Dorigo	1995
Elitist Ant System (AS_{elite}) [122, 123]	Dorigo	1992
Ant System (AS) [122–125]	Dorigo, Maniezzo, Colorni	1991

Table 2.2: Selection of pioneering ACO algorithms (cf. [11])

basis and to make the different ACO algorithms comparable, the basic ideas of the algorithms will be explained as application on the TSP.

Ant System

The first ant algorithm in the research area of ACO algorithms was proposed by Dorigo [122–125], the so called Ant System (AS)[14]. The basic idea of using artificial ants to solve combinatorial problems and the first prototype implementation triggered a lot of related researches dealing with various interesting ACO algorithms and their applications.

The main idea of the first AS algorithm was that m artificial ants should build concurrently a solution of the TSP by traversing the construction graph, i.e. making probabilistic decisions from vertex to vertex or correspondingly from city to city.

The basic AS algorithm can be subdivided into two different phases: the construction of a solution and the pheromone update. At the beginning of the construction phase, each ant is assigned to a random city. At each step, each ant needs to decide which city should be visited next. This is done by applying the *random proportional* rule that is specifying the probability of the k-th ant moving from city i to city j. To avoid ants visiting the same city multiple times, each ant stores a list of all cities visited so far. The *random proportional rule* can be formally defined as

[14] In the original research paper [125] three different ant algorithms were proposed: *ant density*, *ant quantity* and *ant cycle* algorithm. However, due to the fact that experiments have shown that ant-cycle is superior to the other two algorithms in succeeding publications the ant cycle algorithm is referred to as Ant System (AS).

follows:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

where

- m number of ants
- τ_{ij} is the pheromone value that is changing during construction of solutions
- $\eta_{ij} = 1/d_{ij}$ is a heuristic value that is know a priory (d_{ij} is the distance between the cities i and j)
- α and β are two parameters that define the relative influence of pheromone and heuristic information
- $\mathcal{N}_i^k$ is the feasible neighbourhood of ant k, i.e. the set of cities that can be reached from city i and that have not been visited yet by ant k

The probability of choosing a certain arc(i, j) increases with the value of the associated pheromone information τ_{ij} and the value of the associated heuristic information η_{ij}. The relative value of the parameters α and β define the relative importance of the pheromone and corresponding heuristic.

After all ants have constructed their tour the second phase, the pheromone update, can be triggered. The special characteristic of AS is that all ants are involved in the pheromone updating process. The pheromone update can be formally denoted as:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{2.21}$$

where

- m number of ants
- $\rho \in (0, 1]$ evaporation rate

- $\Delta\tau_{ij}^{k}$ quantity of pheromone on the arc (i, j)

To avoid local minima, pheromone evaporation is used. The evaporation is applied uniformly to all arcs with the evaporation rate ρ.

The $\Delta\tau_{ij}^{k}$, the quantity of pheromone on the arc (i, j), is defined as follows:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

where

- Q is a constant (in most cases $Q := 1$)
- L_k is the length of the tour build by the k-th ant, i.e. the sum of all the length of arcs in the tour

As a result, arcs that are part of short ants' tours will obtain more pheromone than the arcs that are part of longer tours so that pheromone is increased proportional to the quality of the tour. This impact is intensified due to the fact that arcs that are part of short tours and are used by many ants, will have more pheromone and thus, following ants are biased by choosing the same arcs with high pheromone concentration in the future.

Elitist Ant System

One of the first improvements of AS was proposed by Dorigo et al. [122, 123] with the Elitist Ant System (EAS). The modified EAS algorithm provides an additional reinforcement mechanism that deposits additional pheromone on the arcs that are belonging to the *best-so-far tour* found by the ants.

The additional adding of pheromone is achieved by sending a group of elitist ants along the best-so-far tour and placing a certain amount additional pheromone at each arc they visit. Formally, the pheromone updating equation from AS (see equation 2.21) is adapted to:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{best} \tag{2.23}$$

where

- e is the number of elitist ants
- $\Delta\tau_{ij}^{best}$ quantity of pheromone on the arc $(i,\ j)$ of the best-so-far tour

The additional amount of pheromone $\Delta\tau_{ij}^{best}$ that is deposited by the elitist ants is defined as follows:

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{Q}{L_{best}} & \text{if edge } (i,j) \text{ is part of the best-so-far tour} \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

where

- Q is a constant (in most cases $Q := 1$)
- L_{best} is the length of the best-so-far tour build by an ant

As a result, the best-so-far tour reinforced by elitist ants influences all other ants with a certain probability to construct solutions using edges of the best-so-far tour. Simulation results [122, 123] showed that with a well-balanced amount of elitist ants, better tours can be found quicker than with the AS algorithm.

Rank-based Ant System

Another modification of AS was proposed by Bullnheimer et al. [113], the so called Rank-based Ant System (AS_{rank}). In this variation, the amount of pheromone that is deposited by the ants is depending on the rank of the ant. Moreover, just as in the EAS algorithm, the ant that has found the best-so-far tour deposits the largest amount of pheromone at each iteration.

To obtain the rank of each ant, the ants are sorted in descending order by their tour length. The quantity of pheromone that is deposited by an ant is weighted by the ant's rank. Formally, the pheromone updating equation from AS (see equation 2.21) is adapted to:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{best} \tag{2.25}$$

where

- r is the rank of the ant
- w is the number of ants that are updating the pheromone information
- $\Delta\tau_{ij}^{r}$ quantity of pheromone on the arc (i, j) of r-ranked ant

The quantity of pheromone $\Delta\tau_{ij}^{r}$ that is deposited by the r-ranked ant is defined as follows:

$$\Delta\tau_{ij}^{r} = \begin{cases} \frac{Q}{L_{rank}} & \text{if edge } (i,j) \text{ is part of the r-ranked ant's tour} \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

where

- Q is a constant (in most cases $Q := 1$)
- L_{rank} is the length of the r-ranked ant's tour

As a result, at each iteration the $(w-1)$ best-ranked ants as well as the ant that has found the best-so-far tour[15] are updating the pheromone information. The rank of an ant does directly influence the amount of pheromone by the weight $(w-r)$.

Bullnheimer et al. [113] showed in their experiments that AS_{rank} seems to provide a good compromise between exploitation, in terms of the reinforcement of good paths, and exploration, in terms of reinforcing several good paths instead of only the best. As a result, AS_{rank} significantly outperforms AS and is slightly better than AS_{elite}.

$\mathcal{MAX} - \mathcal{MIN}$ Ant System

The Max-Min Ant System (MMAS) proposed by Stützle and Hoos [114–116,126], is the first major modification of the original AS algorithm. The proposed changes aim, on the one hand, at a better exploitation of the

[15] note: the best-so-far tour does not necessarily has to be found in the current iteration.

best solutions found to direct the ants to high quality solutions and, on the other hand, at avoiding premature convergence of ants' searches.

Basically, MMAS differs from AS in the following three main points:

- **Emphasising best solutions**: In MMAS the best solution found is heavily exploited because after each iteration only the best ant is allowed to add pheromone to its path. The 'best ant' can either be the *iteration-best ant*, i.e. the ant that found the best solution in the current iteration, or the *best-so-far ant* that found the overall best solution until now. In each iteration one of the modes is used exclusively, i.e. either the iteration-best ant or the best-so-far ant is allowed to update the pheromone. Nevertheless, in some applications a mixture of the iteration-best ant and the best-so-far ant is used.

- **Restriction of pheromone values**: Due to the first modification the algorithm may enter quickly a stagnation state in which only good, but not optimal, paths are used. This effect is mitigated by introducing a limit for the pheromone values so that a pheromone value has to be within the range $[\tau_{min}, \tau_{max}]$.

- **High exploration at bootstrapping phase**: To achieve a high exploration of different solutions in the bootstrapping phase of the algorithm, the initial pheromone value is set to τ_{max}, the upper boundary of the introduced interval.

In MMAS the pheromone is updated as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}^{best} \tag{2.27}$$

with

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{Q}{L_{best}} & \text{if edge } (i,j) \text{ is part of the best tour} \\ 0 & \text{otherwise} \end{cases} \tag{2.28}$$

where

- Q is a constant (in most cases $Q := 1$)

- L_{best} is the length of the best tour, i.e. in case of *iteration-best* ant L_{best} is assigned the length of the tour L_{ib} found by the iteration-best ant ($L_{best} := L_{ib}$) or, in case of *best-so-far* ant, L_{best} is assigned the length of the tour L_{bs} found by the *best-so-far* ant ($L_{best} := L_{bs}$)

The frequency of the application of the different pheromone updating rules, which are depending on the *iteration-best* ant or *best-so-far* ant, influences the greediness of the search: while the pheromone updating rule, based on the *best-so-far* ant, rapidly concentrates on the search close to the best-so-far tour; the pheromone updating rule, based on the *iteration-best* ant, provides a more exploratory fashion by updating different good edges. Experiments have shown that the best results can be obtained when using both pheromone updating rules with a gradual increase of the frequency of the *iteration-best* ant's rule [116, 126].

Stützle and Hoos suggest to determine the lower bound τ_{min} and upper bound τ_{max} experimentally for the specific problem. Nevertheless, they also present an approach to calculate these boundaries analytically. To get an idea of the analytical approach a shortened version is presented here (for details cf. [116]):

If the length of the optimal tour L_{opt} is known, the upper bound τ_{max} can be simply calculated as follows:

$$\tau_{max} = \frac{1}{\rho L_{opt}} \tag{2.29}$$

Due to the fact that in most cases the optimum tour length is unknown, often the *best-so-far* tour length L_{best} is used as approximation of the optimum tour length. So every time an improved L_{best} is found τ_{max} is updated as follows:

$$\tau_{max} = \frac{1}{\rho L_{best}} \tag{2.30}$$

The lower boundary τ_{min} has to be chosen carefully because it has a huge impact on the performance of the algorithm. The presented analytic calculation of τ_{min} is based on the probability p_{best}, which is the probability for choosing the 'right' edge in every step, resulting in the best possible solution. When p_{dec} is the probability of choosing the 'right' edge directly,

then if n-times the 'right' decision is made $p_{dec} = \sqrt[n]{p_{best}}$.[16]. Based on this and the fact that an ant needs to choose on average among avg edges, τ_{min} can be calculated as follows:

$$\tau_{min} = \frac{\tau_{max}(1 - p_{dec})}{avg\ p_{dec}} \tag{2.31}$$

To increase the exploration of new solutions, in MMAS occasionally the pheromone trails are reinitialised, i.e. $\tau_{ij} := \tau_{max}$. The reinitialisation is triggered when the algorithm enters the stagnation phase, e.g. determined by analysing additional statistics, or if a certain number of iterations did not lead to any improvement of the best-so-far solution [115, 116, 126].

Finally, after each iteration it needs to be guaranteed that τ_{ij} is within the defined range $[\tau_{min}, \tau_{max}]$. Therefore, the following rule is applied to set a valid value for τ_{ij}:

$$\tau_{ij} = \begin{cases} \tau_{min} & \text{if } \tau_{ij} < \tau_{min} \\ \tau_{ij} & \text{if } \tau_{min} \leq \tau_{ij} \leq \tau_{max} \\ \tau_{max} & \text{if } \tau_{ij} > \tau_{max} \end{cases} \tag{2.32}$$

Due to the fact that MMAS is outperforming the original AS significantly, MMAS is one of the most studied algorithms in the research area of ACO algorithms. It has also been applied to several application areas starting from the original TSP [115] to further areas such as Quadratic Assignment Problem (QAP) [116], University Course Timetabling Problem (UCTP) [127] etc.

Ant Colony System

Ant Colony System (ACS), proposed by Dorigo and Gambardella [117–119], is another ACO algorithm that was inspired by the original AS algorithm. However, in contrast to the previously discussed ACO algorithms, the modifications made in ACS are crucial so that ACS can be rather seen as a novel ACO algorithm, instead of a direct descendant of AS.

The key features of ACS can be summarised as follows:

[16] assumptions: heuristic information is neglected; p_{dec} is constant during tour construction

- Pseudo-random proportional rule
- global pheromone update
- Local pheromone update

The *pseudo-random proportional rule* specifies how an ant k, currently located at city i, can move to city j. Formally the pseudo-random proportional rule can be defined as follows:

$$j = \begin{cases} \arg\max_{l \in \mathcal{N}_i^k} \{\tau_{il}[\eta_{il}]^\beta\} & \text{if } q \leq q_0 \text{ exploitation} \\ J & \text{otherwise (biased exploration)} \end{cases} \tag{2.33}$$

where

- q is a random variable uniformly distributed in $[0, 1]$
- q_0 is a parameter ($0 \leq q_0 \leq 1$), which determines the relative importance of exploitation vs. exploration. Higher q_0 leads to greedier decisions.
- J is a random variable selected according to the probability distribution in equation 2.20

Based on the pseudo-random proportional rule, an ant can make the best possible move, in terms of learned pheromone and heuristic information, with a probability of q_0 or it explores new arcs with a probability of $(1 - q_0)$. Thus, the parameter q_0 allows to set the degree of exploitation and correspondingly the degree of exploration for the algorithm, i.e. it can be adjusted whether the ants should concentrate on finding better tours near the best-so-far solution or rather explore new tours instead.

The *global pheromone update* in ACS is triggered after each iteration by the best-so-far ant only. Formally the update in ACS can be denoted as follows:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{best} \tag{2.34}$$

with

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{Q}{L_{best}} & \text{if edge } (i, j) \text{ is part of the best tour} \\ 0 & \text{otherwise} \end{cases} \tag{2.35}$$

where

- Q is a constant (in most cases $Q := 1$)
- L_{best} is the length of the best-so-far tour

In contrast to AS, in ACS the pheromone update as well as the evaporation does solely affect the best-so-far tour[17] so that the computational complexity for updating the pheromone trails is dramatically reduced. Furthermore, in ACS the new pheromone trail is a weighted average between the old pheromone value and the newly deposited amount of pheromone.

Additionally to the global pheromone update, in ACS *local pheromone update* is applied that is triggered during the tour construction each time an ant passes an arc (i, j). Every time an ant k adds a component to its partial solution, the pheromone value is decreased by applying the following formula:

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0 \tag{2.36}$$

with

- τ_0 is the initial value of pheromone trails (normally a small constant)
- ξ $(0 < \xi < 1)$ is the pheromone decay coefficient that controls the ants' degree of exploration

As a result, the pheromone concentration of traversed edges is decreased to encourage following ants to choose another edge than the previous ant. This is done to increase the exploration of different solutions so that stagnation can be avoided.

Finally, it should be mentioned that ACS is based on the earlier algorithm Ant-Q [120, 121]. The main difference of the algorithms is the setting of the parameter τ_0. Because it was found that if τ_0 was set to a small value, ACS provides similar results as Ant-Q with less complexity, Ant-Q was abandoned.

[17] Dorigo and Gambardella did also experiments with the iteration-best tour, but the results showed that the best-so-far tour outperformed the iteration-best tour variant so that only the best-so-far tour was used in subsequent experiments [118].

2.4 Theoretical Works on Ant Colony Optimisation Algorithms

The first works on ACO algorithms were experimentally driven with the goal to show the successful use of this sort of algorithms. Starting from these practical works, the researchers started to have a look on the theoretical foundations of ACO algorithms.

One of the first questions that was raised by researchers was whether ACO algorithms are converging to an optimal solution or not, i.e. whether there is a situation in which the ACO algorithms generate the same optimal solution over and over again.

The first work on ACO algorithms' convergence was presented by Gutjahr, who worked on a formal extension of AS, the so called Graph–based Ant System (GBAS), a metaheuristic capable of handling arbitrary static combinatorial optimisation problems. For GBAS Gutjahr proved its convergence with a probability $\geq 1 - \epsilon$ to the optimal solution with an arbitrary small ϵ [128] and later, its convergence to an optimal solution with probability exactly one [129]. However, due to the fact that the presented proof was Graph–based Ant System (GBAS) specific, the results could not be generalised to other ACO algorithms.

For the two best experimentally performing ACO algorithms, MMAS and ACS, Stützle and Dorigo could show their convergence to an optimal solution [96, 130].

Nevertheless, all presented convergence proofs for ACO algorithms are not saying anything about the speed of convergence, i.e. the computationally required time, to find an optimal solution.

While convergence proofs are providing a mathematical insights on the ACO algorithms' properties, best-practices for their implementations cannot be derived directly.

2.4 Applications of Ant Colony Optimisation Algorithms

There are various application areas in which ACO algorithms can be applied. Table 2.3 shows a nice overview about those main application areas of ACO algorithms. Although it is clear that this list is not comprehensive, it shows the diversity of application areas in which ACO algorithms

can be used.

Problem Type	Problem Name	Authors
Routing	Travelling Salesman	Dorigo et al. [124]
		Dorigo, Gambardella [117]
		Stützle, Hoos [115, 116]
	Vehicle Routing	Gambardella et al. [131]
		Reimann et al. [132]
	Sequential Ordering	Gambardella, Dorigo [133]
	Communication Networks	Di Caro, Dorigo [134]
	Routing in MANETs	Ducatelle et al. [135]
		Di Caro et al. [136]
Assignment	Quadratic Assignment	Stützle, Hoos [116]
		Maniezzo [112]
	Course Timetabling	Socha et al. [127, 137]
	Graph Colouring	Costa, Hertz [138]
Scheduling	Project Scheduling	Merkle et al. [139]
	Total Weighted Tardiness	Den Besten et al. [140]
		Merkle, Middendorf [141]
	Open Shop	Blum [104]
Subset	Set Covering	Lessing et al. [142]
	k-Cardinality Trees	Blum, Blesa [143]
	Multiple Knapsack	Leguizamón, Michalewicz [144]
	Maximum Clique	Fenet, Solnon [145]
Other	Constraint Satisfaction	Solon [146, 147]
	Classification Rules	Parpinelli et al. [148]
		Martens et al. [149]
	Bayesian Networks	Campos et al. [150, 151]
	Protein Folding	Shmygelska, Hoos [152]
	Protein-Ligand Docking	Korb et al. [153]

Table 2.3: Selected applications of ACO algorithms (cf. [12])

Although, there are a lot of variations of ACO algorithms in different application contexts, it can be stated that MMAS and ACS are belonging to the best known ACO algorithms so far.

2.5 Multi-objective Ant Colony Optimisation

Recently, different researches propose the extension of ACO algorithms to support multi-objective problems, the so called Multi-objective Ant Colony Optimisation (MOACO) algorithms. Most of the proposed MOACO algorithms tend to be based on well-known single-objective ACO algorithms such as ACS or MMAS.

In the following a taxonomy of MOACO algorithms is discussed that can be used to classify these algorithms by their main criteria. Besides, the

identified main components of the MOACO algorithms will be later used in the implementation part of the thesis (see chapter 4).

2.5 Taxonomy of Multi-objective Ant Colony Optimisation Algorithms

There are some approaches to classify MOACO algorithms by their main characteristics including the researches of López-Ibáñes and Stützle [154–156], Garcia-Martinez et al. [157], Angus and Woodward [19] and Angelo and Barbosa [158]. To get an insight into these characteristics of MOACO algorithms and to ease the identification of the different components, which are required for the implementation of a MOACO algorithm, in the following a taxonomy of MOACO algorithms is presented based on those researches:

Pheromone and Heuristic Information: For MOACOs pheromone information needs to be stored in some kind of pheromone structure. Without loss of generality in the following it is assumed that the pheromone information is stored in a matrix. There are basically two options how to store pheromone information:

- **Single matrix**: If a single pheromone matrix is used, the algorithm works similar to a traditional single-objective ACO algorithm. However, due to the fact that multiple objectives should be managed at the same time, those objectives needs to be aggregated to obtain a combined value, which is then stored in the matrix.

- **Multiple matrices**: If multiple pheromone matrices are used, typically each objective is assigned to one matrix. In this case, each pheromone matrix reflects the solution components' advantage for a certain objective.

The same applies to the heuristic information, which either can be stored in a single matrix or in multiple matrices.

Number of Ant Colonies Another important characteristic of MOACO algorithms is the number of ant colonies that is used[18]. In general, there are two options:

- **One colony** This is the trivial case in which one ant colony is optimising multiple objectives at the same time.
- **Multiple colonies** If multiple ant colonies are used, normally each colony is responsible for the optimisation of one objective. Consequently, in most cases each ant colony operates on its own pheromone matrix. However, a disadvantage of using multiple ant colonies is that the overhead is increased.

When the multiple ant colonies should cooperate with each other, either solutions found can be exchanged via a common shared archive of solutions to find the best solution; or, solutions found by one ant colony affect other ant colonies by modifying the pheromone information of the other ant colonies. Each ant colony can have a certain number of ants that is belonging to a colony.

Aggregation of Pheromone and Heuristic Information: In the case of using multiple pheromone/heuristic matrices, these matrices need to be aggregated. Different aggregation strategies can be used for combining these matrices, the most common strategies are the following:

- **Weighted sum**: The matrices are aggregated by a weighted sum: $\sum_{l=1}^{n} \lambda_l \tau_{ij}^l$
- **Weighted product**: The matrices are aggregated by weighted product: $\prod_{l=1}^{n} (\tau_{ij}^l)^{\lambda_l}$
- **Random**: At each construction step, randomly one of the matrices is chosen.

[18]The term 'ant colony' is not used consistently in the literature. In this thesis the definition introduced by Iredi et al. [159] is used which defines ant colonies as multiple instances of single-colony algorithms. Consequently, each colony acts independently, using separate pheromone information and a separate group of ants. To exchange information, the ant colonies are able to communicate with each other.

The number of weights that is used in the aggregation is normally corresponding to the number of ants or the number of objectives. The weights λ_l per objective can be set either *fixed*, i.e. the weights for each objective are set *a priori* based on prior knowledge and remain unchanged the entire time; or, the weights are set *dynamically*, i.e. different objectives can be weighted differently at different times. For the latter case, for instance, different ants can assign different weights at different iterations.

Pheromone Update: The updating of pheromone information can be done in several ways. In most algorithms only the backward ants are allowed to update the pheromone information. Basically there are five main ways to specify which ants are allowed to update the pheromone information:

- **Elite solution**: If only a single matrix is used for all objectives, a similar approach as for the normal ACO algorithms can be used by only allowing the *best-so-far* ant or the *iteration-best* to update the pheromone matrix.
- **Best-of-objective solutions**: If multiple matrices are used, also an elite like strategy can be used by choosing the *best-so-far* and or the *iteration-best* ant with respect to each objective.
- **Best-of-objective-per-weight solutions**: An elite solution is picked in respect to each objective, while considering the weight λ.
- **Non-dominated solutions**: In this case the ants store all non-dominated solutions in the set. Only the ants that provided a non-dominated solution are allowed to update the pheromone. Additionally, an elite solution of the non-dominated solutions could be picked by specifying additional knowledge of how to find the best solution from this set.
- **All solutions**: This is the trivial case, in which ants of all solutions found are allowed to update the pheromone information.

Hybrid solutions are also conceivable, for example, all ants deposit pheromone on their way, while the *best-so-far* ant adds an additional

amount of pheromone to the path of the best solution.

Evaluation of Solutions: Although in the area of MCOPs there are several methods to evaluate a solution regarding different criteria, in the area of MOACO algorithms it is important whether a Solution is Pareto-optimal or not. The following distinction for a solution can be made:

- **Pareto solutions**: For finding Pareto solutions, the domination criterion (see section 2.3.3) is considered to find the best solutions taking all objectives equally into account.
- **Non-Pareto solutions**: For finding non-Pareto Solutions either the solutions take only one of the objectives into account or by aggregating all objectives and considering the combined value.

Archival of Solutions: When a solution is found its storage needs to be considered. Basically, there are four options how a solution can be stored:

- **Offline storage**: When a new solution is found, it is used to update the pheromone information. Afterwards, it is added to an offline storage, also referred to as archive. When looking for Pareto solutions, all non-Pareto solutions are removed from the archive. The archive does not influence any further decisions, but it is used to store historic data by time in the 'hall of fame'. At the end of the algorithm, the hall of fame is returned as list of solutions.
- **Online storage**: When a new solution is found, it is directly added to the population of solutions. The change of the population of solutions triggers directly the pheromone update procedure using the improved solution set. In the case of Pareto solutions, all dominated solutions are removed. Consequently, the population of solutions contains always the best solutions found[19]. The last population of solutions is returned at the end of the algorithm as set of final solutions.

[19] In some cases the Pareto set is only used to return the final solutions after the completion of the algorithm, while during the online storage non-Pareto solutions are considered.

- **Elite storage**: Only a single solution, the elite solution, is stored. The stored elite solution can be used to update the pheromone information and it can also be returned as best-so-far solution after completing the algorithm.
- **No storage**: New solutions are used to update the pheromone information, but are discarded subsequently. Only when ending the algorithm the last solution found is considered as final solution.

Table 2.4 shows an overview of all components derived from the presented characteristics of the MOACO algorithms taxonomy.

Component	Options
Pheromone Information ($[\tau]$)	one matrix, multiple matrices
Heuristic Information ($[\eta]$)	one matrix, multiple matrices
Number of Ant Colonies	single colony (*single*), multiple colonies (*multiple*)
Aggregation of Pheromone/ Heuristic Information	weighted sum ($\sum$), weighted product ($\prod$), random (*rand*)
Pheromone Update	elite solution (*es*), best-of-objective solutions (*boo*), best-objective-per-weight solutions (*bopw*), non-dominated solutions (*nd*), all solutions (*all*)
Evaluation of Solutions	Pareto solutions (*p*), non-Pareto solutions (*np*)
Archival of Solutions	offline storage (*offline*), online storage (*online*), elite storage (*elite*), no storage (*none*)

Table 2.4: Taxonomy-based components of MOACO algorithms

Based on the identified components of the provided taxonomy, the most common MOACO algorithms can be classified as shown in table 2.5.

2.5 Performance Metrics

When considering different MOACO algorithms the question arises how different solutions can be compared regarding their performance? While the performance analysis of ACO algorithms is rather straight forward, because only the single best solution needs to be considered, the analysis of performance metrics of MOACO algorithms is more difficult because in most cases an entire set of solutions is returned.

In other researches often *unary quality metrics*, also referred to as *unary quality indicators*, are used to compare different solution sets. Unary quality metrics are able to assign quality values (real numbers) to solution sets to make them comparable with each other. However, although unary

Algorithm	$[\tau]$	$[\eta]$	Col.	Aggr.	Weigh.	τ Upd.	Eval.	Arch.
MOAQ [160]	1	d	*multiple*	$\prod$,$\sum$	d	*nd*	*p*	*offline*
MACS-VRPTW [131]	d	1	*multiple*	–	–	*boo*	*np*	*elite*
BicriterionAnt [159]	d	d	*single*	$\prod$	m	*nd*	*p*	*offline*
COMPETants [161]	d	d	*multiple*	$\sum$	$d+1$	*boo*	*p*	*none*
ACOAMO [162]	1	1	*single*	–	–	*all*	*np*	*elite*
SACO [163]	1	1	*single*	–	–	*elite*	*np*	*none*
MACS [164]	1	d	*single*	$\prod$	m	*nd*	*p*	*online*
MONACO [165]	d	1	*single*	$\prod$	d	*all*	*np*	*offline*
PACO-MO [106]	d	d	*single*	$\sum$	m	*all*	*p*	*online*
P-ACO [166]	d	d	*single*	$\sum$	m	*boo*	*np*	*offline*
M3AS [167]	1	d	*single*	$\prod$	d	*nd*	*p*	*offline*
MOA [168]	1	1	*single*	$\prod$	d	*nd*	*p*	*offline*
CPACO [169]	d	d	*single*	$\sum$	d	*nd*	*p*	*online*
MOACSA [170]	1	1	*single*	$\prod$	d	*elite*	*np*	*none*
mACO-1 [171]	d	d	*multiple*	*rand*	$d+1$	*bopw*	*p*	*offline*
mACO-2 [171]	d	d	*multiple*	$\sum$	$d+1$	*bopw*	*p*	*offline*
mACO-3 [171]	1	1	*single*	–	–	*nd*	*p*	*offline*
mACO-4 [171]	d	1	*single*	*rand*	d	*boo*	*p*	*offline*

Table 2.5: Taxonomy of MOACO algorithms with d as number of objectives, m number of ants

quality indicators seem to be a suitable tool to assess solution sets, those indicators are difficult to develop. One of the main problems, as Zitzler et al. [172] highlight, is that for MCOPs the optimisation goal itself consists of multiple objectives:

- The distance of the resulting non-dominated set to the Pareto-optimal front should be minimized.
- A good (in most cases uniform) distribution of the solutions found is desirable (may be based on certain distance metric).
- The extent of the obtained non-dominated front should be maximized, i.e., for each objective a wide range of values should be covered by the non-dominated solutions.

Due to the fact that in this thesis not all quality indicators can be discussed comprehensively, in the following, just a short introduction to the main approaches will be given:

Zitzler et al. [173] and Knowles et al. [174] present in their works several unary quality indicators and discuss their limitations and the problems that can occur and may lead to false or misleading assessments of the quality of solution sets. Based on their findings Knowles et al. [174] recommend three widely accepted unary quality indicators that can be applied to assess the quality of Pareto sets:

1. The hypervolume indicator I_H [175]
2. The unary Epsilon Indicators I^1_ϵ and $I^1_{\epsilon+}$ [173]
3. The I^1_{R2} and I^1_{R3} indicators [176]

Those three unary quality indicators are described briefly in the following:

Hypervolume Indicator

The hypervolume indicator (I_H) [175], proposed by Zitzler and Thiele, measures the hypervolume spanned by the non-dominated front and a bounding point z that is at least weakly dominated by all points (see figure 2.17).

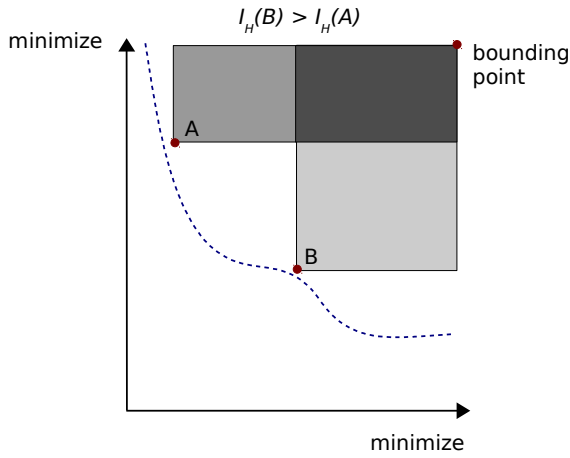


Figure 2.17: Example of hypervolume indicator

The hypervolume is to be maximised. The hypervolume difference of the solution set A can also be considered to a reference set R, then the hypervolume indicator is defined as follows:

$$I_H^- = I_H(R) - I_H(A) \tag{2.37}$$

However, as While et al. [177] showed with an increasing number of objectives the computational cost of this indicator grows exponentially.

Unary Epsilon Indicators

The unary epsilon indicator [173], proposed by Zitzler et al., exists in a multiplicative (I_ϵ^1) and in an additive ($I_{\epsilon+}^1$) version. Both unary epsilon indicators can be directly derived from their binary versions. Thus, in the first step, the binary version of the I_ϵ-indicator is defined as follows:

$$I_\epsilon(A, B) = \inf_{\epsilon \in \mathbb{R}_+} \{\forall b \in B \; \exists a \in A | a \preceq_\epsilon b\} \tag{2.38}$$

with the ϵ-relation ($\preceq_\epsilon$) for the multiplicative case defined as

$$a \preceq_\epsilon b \Leftrightarrow \forall i \in \mathbb{N} : f_i(a_i) \leq \epsilon \cdot f_i(b_i) \tag{2.39}$$

The binary multiplicative epsilon indicator $I(A, B)$ provides the minimum ϵ-factor by which the objective vector associated with B can be multiplied such that it is weakly dominated by the objective vector associated with A, which represents the approximation of the Pareto front.

Correspondingly, for the additive case, the binary version of the $I_{\epsilon+}$-indicator can be defined using the following relation:

$$a \preceq_{\epsilon+} b \Leftrightarrow \forall i \in \mathbb{N} : f_i(a_i) \leq \epsilon + f_i(b_i) \tag{2.40}$$

From the binary version, the unary indicators $I^1_\epsilon = I_\epsilon(A, R)$ and $I^1_{\epsilon+} = I_{\epsilon+}(A, R)$ can be derived by using the reference set R. The binary as well as the unary epsilon indicators are to be minimised.

R2 and R3 Indicators

The I^1_{R2} and I^1_{R3} indicators [176], proposed by Hansen and Jaszkiewicz, are based on a set of utility functions. A utility function u is defined as a mapping from the set $\mathbb{R}^n$ of the n-dimensional objective vectors to the set of real numbers, in short:

$$u : \mathbb{R}^n \mapsto \mathbb{R} \tag{2.41}$$

If the decision maker's preferences are given as parametrised utility function u_λ with $\lambda_1, ..., \lambda_n \in \Lambda$ as vector of weights, the binary quality indicators I_{R2} and I_{R3} can be derived from this family of utility functions as follows:

$$I_{R2}(A, B) = \frac{\sum_{\lambda \in \Lambda} u^*(\lambda, A) - u^*(\lambda, B)}{|\Lambda|} \tag{2.42}$$

$$I_{R3}(A, B) = \frac{\sum_{\lambda \in \Lambda} [u^*(\lambda, B) - u^*(\lambda, A)]/u^*(\lambda, B)}{|\Lambda|} \tag{2.43}$$

where u^* is the maximum value reached by the utility function u_λ with the weight vector λ on the solution set A, in short:

$$u^*(\lambda, A) = \max_{z \in A} u_\lambda(z) \tag{2.44}$$

From the binary indicators, the unary indicators I^1_{R2} and I^1_{R3} can be derived using a reference set, i.e. $I^1_{R2}(A) = I_{R2}(A, R)$ and $I^1_{R3}(A) =$

$I_{R3}(A, R)$.

For the utility function u_λ there are several options. Often a *weighted linear function*, the *nonlinear weighted Tchebycheff function* or the combination of both the *augmented Tchebycheff function* is used.

Remarks on Quality Indicators

In some of the discussed approaches reference points or reference sets need to be chosen. Based on the research of Knowles et al. [174] the following guidelines can be derived for finding such reference points or sets:

Finding reference points: If the bounds of the objective space are known, use these as reference points. Otherwise, all solution sets can be combined and then the ideal and Nadir points can be computed. The found ideal or nadir point can then be shifted until it strictly dominates all points in the solution sets.

Finding reference sets: If the Pareto front is known, use it as reference set. Otherwise, a reference set from the literature can be chosen, if it is known as good approximation of the Pareto front. Also the 50% attaintment surface of random search can be used, i.e. half of all points in the decision space will weakly dominate this surface. However, some software tool is needed to generate this.

As final remark, it can be stated that due to the fact that each quality indicator takes different preference information into account, no 'best' quality indicator can be highlighted. Knowles et al. recommend to take multiple quality indicators into account to get the most information. Besides, the selection of reference points or reference sets has an influence on the quality indicators. For further information about the presented quality indicators see [174].

2.6 Summary

In this chapter the theoretical foundations of the thesis were discussed that should enable the reader to get the sufficient theoretical background

information that is required to understand the following chapters of this thesis. Furthermore, these basics should help to understand the decisions that will be made later in the implementation of this thesis (see chapter 4).

In the first step, the basic idea of Wireless Sensor Networks (WSNs) was introduced starting with the various application areas of WSNs, the hardware components of the sensor nodes and the distributed network architecture of WSNs. Due to the fact that MANETs and WSN are often mentioned in the same breath, afterwards, the differences between MANETs and WSNs were elaborated for a clear distinction between both network types. Moreover, the unique constraints and challenges of WSNs were discussed that need to be considered in the research area of WSNs. Subsequently, the basic security requirements of WSNs were emphasised that should be taken into account for a secure WSN.

In the next step, the terms of trust and reputation were introduced, highlighting the difference between hard and soft security. Afterwards, the formal notion of trust, uncertainty, transitivity of trust, multidimensionality of trust and reputation were presented. Subsequently, different classes of trust were discussed as well as the meaning of TnR in WSNs.

Due to the fact that routing in WSNs can be treated as COP, in the next step, the fundamental concepts of COPs were introduced. Starting with a basic introduction to optimisation problems and the general phases of the optimisation process, COPs were formally introduced. After that, the problem of complexity was discussed and basic algorithms were presented that can be used to solve COPs.

COPs were extended to MCOPs, i.e. optimisation problems considering multiple (conflicting) objectives at the same time, which are subjects to certain constraints. MCOPs were formally introduced and the idea of Pareto optimality was explained. To select a final solution from the Pareto set, the concept of decision making was discussed, specifically the approach of Multi-Criteria Decision Making (MCDM), covering some basic methods that are widely used for different decision problems.

In the next step, a closer look was taken on biologically-inspired algorithms that can be applied to solve optimisation problems. A special

focus was laid on Ant Colony Optimisation (ACO), a swarm intelligence based metaheuristic, which can be applied to solve COPs. The covered topics included the natural origin of ACO, the transition from natural ants to artificial ants as well as the presentation of well known ACO algorithms, such as AS, MMAS and ACS.

To be able to consider multiple (conflicting) objectives at the same time, ACO was extended to MOACO, i.e. ACO-based algorithms that are capable of solving MCOPs. The main characteristics of MOACO algorithms were discussed and a taxonomy was created in which existing, well-known MOACO algorithms were classified. Besides, a selection of performance metrics was introduced that can be used to compare solution sets of different MOACO algorithms.

In the following chapter 3, the application of ACO-based as well as MOACO-based algorithms to the routing in WSNs will be discussed.

3

Application of Ant Colony Optimisation to the Routing in WSNs

> Ants are good citizens, they place group interests first.
>
> Clarence Day (1874 - 1935)

In this chapter, the application of ACO algorithms to the routing in WSNs is discussed. Therefore, in the first step, the general characteristics of the routing in WSNs will be explained, including the basic routing challenges in WSNs, a taxonomy of WSN routing algorithms and the basic requirements for WSN routing algorithms. Subsequently, in the second step, the idea of ACO-based routing algorithms will be presented with a special focus on WSNs. Starting with the basic idea of ACO-based routing and its origin in wired networks, a special focus is set on the current use of ACO-based algorithms in wireless networks such as MANETs and WSNs. Afterwards, recent researches from the area of MOACO-based routing algorithms will be discussed, which extend the idea of ACO-based algorithms by considering multi-objectives at the same time during the routing process. Finally, the related works in the area of ACO-based and MOACO-based routing algorithms will be summarised.

3.1 Routing in WSNs

Due to the discussed resource constraints of the sensor nodes and the use of the wireless channel as communication medium for the exchange of information, the routing in WSNs is quite challenging. In the following, the characteristics of the routing in WSNs are discussed. After that, a taxonomy of WSN routing algorithms is presented that can be used to classify routing algorithms regarding certain criteria. Derived from the routing characteristics some routing requirements are presented that can

be used as guidelines for 'good' WSN routing algorithms. Finally, attacks on the network layer are presented, a routing protocol developer should be aware of.

3.1 Characteristics of the Routing in WSNs

Because of the sensors' energy constraints the radio range of each sensor node is limited. Consequently, the sensor nodes need to cooperate with each other to forward packets via multiple hops from a source to a remote destination node. Therefore, WSN routing protocols should be capable of forwarding packets via multiple hops.

Due to the use of the wireless channel, which is a broadcast medium shared by all radio devices, additional challenges arise that are affecting the routing in WSN: for example, if multiple devices are sending data at the same time, packet collisions or interference can occur. Furthermore, interferences from the environment can attenuate the wireless signal so that it cannot be received correctly at the receiver side. This can be either unintentional due to environmental conditions or intentional due to adversaries that try to influence or jam the signal. Though, the basic negotiation of the medium access as well as the retransmission of packets in the case of transmission errors should be handled by the MAC-Layer, such as IEEE 802.15.4 [178] or IEEE 802.11 [179], a WSNs routing protocol should be capable of re-routing packets around problematic network regions.

Also from the security point of view the routing in WSNs has several challenges:

For instance, due to the fact that the wireless channel is a broadcast medium, everything that is transmitted can be simply eavesdropped by any adversary that is in radio range. Also the modification or injection of packets is a problem in this sort of environment. Depending on the application, cryptographic measures can be used, on the one hand, to achieve *confidentiality* so that data packets cannot be read in plain text and, on the other hand, to achieve *message integrity* e.g by using message digests.

Also the use of multiple hops to reach a remote destination is a security

challenge because an adversary could compromise a sensor on a route or add additional sensors to the network to capture the network communication. Directly related to this area are insider attacks, i.e. compromised nodes that are seen as 'valid' nodes by their neighbours, but which are misbehaving in some way. While some sort of *authentication* mechanism should be used to ensure the validity of nodes' identities, TRSs can be used as counter measure against insider attacks by providing trusted routes.

The discussed security challenges should also be taken into account while designing secure routing protocols for WSNs.

Additionally, the performance as well as the reliability of a routing protocol plays an important role, particularly if time-critical applications are considered. In general, the end-to-end delay as well as the Packet Delivery Ratio (PDR) are important metrics that can be used to evaluate WSN routing protocols.

3.1 Taxonomy of WSN Routing Algorithms

Routing algorithms for WSNs have several characteristics that depend on the specific application area they are being used in, or more precise the services they are providing. The main characteristics of WSN routing algorithms are discussed in the following (cf. [2, 180–182]):

Proactive vs. Reactive Routing: In a *proactive* – also referred to as *table-driven* – routing approach, routing information is periodically monitored and then distributed among the network nodes. A disadvantage of this approach is that the amount of data caused by the update of routing information is rather high so that additional routing overhead is created. Furthermore, the reaction to topology or traffic changes is slow because new routing information needs a while to spread out in the network. An example of a well-known proactive routing protocol is Optimised Link State Routing Protocol (OLSR) [183].

In a *reactive* – also referred to as *on-demand* – routing approach, routes are only established on request. In the meantime the nodes are idle in terms of routing functionality so that energy can be saved. Normally, in reactive approaches the discovery of a path is implemented with some sort

of broadcasting functionality. A disadvantage of this approach is that the latency is increased because each path has to be established before any data can be sent. Also extensive flooding can become a problem in dense networks leading to congestions. Examples of famous reactive routing protocols are Ad hoc On-Demand Distance Vector Routing (AODV) [184] and Dynamic Source Routing (DSR) [185].

Hybrid routing approaches try to combine the advantages of proactive and reactive routing: initially, some known routes are distributed proactively, while later on demand new routes are added reactively.

Due to the frequently changing topology and the data on request functionality reactive or hybrid routing approaches are preferred for most WSN application scenarios.

Centralised vs. Distributed Routing: In a *centralised* routing approach a central entity manages the routing table of all sensor nodes in the network. Based on the global view of the network, the central point is responsible for distributing routing information to the nodes in the network so that all routing decisions within the network rely on the central entity. A disadvantage of this approach is that the central entity is a bottleneck and single point of failure, i.e. if the central entity fails or cannot be reached, no routing decisions can be taken and thus, no data can be transferred at all. Furthermore, the routing updates sent by the central entity are causing overhead and delays.

On the contrary, in a *decentralised* network the nodes can take decisions autonomously so that the node itself is responsible for any sort of routing decisions. The routing tables, which normally contain only routing decisions for the neighbouring nodes, are managed locally at each node.

The decentralised approach is often preferred for WSNs because of the better scalability and the robustness in case of node failures. In this thesis only the decentralised approach is considered.

Static vs. Dynamic Routing Tables: When *static routing tables* are used, network connections are *a priori* defined by optimising the network regarding some cost-function; such as end-to-end delay, number of hops

etc.; in an offline fashion. Once the routing entries of the routing table are created, these connections are used for the entire lifetime of the network. However, the static routing table approach has some disadvantages, e.g. *a priori* knowledge is required for the creation of the routing table. Moreover, static routing tables cannot react to topology or traffic changes during the lifetime of the network.

In contrast, *dynamic routing tables* are created based on current events that are occurring in the network, i.e. traffic changes and topology changes. During the lifetime of the network, those routing entries can change and adapt to new situations in the network, i.e. the routing tables are updated in an online fashion.

Data Centric vs. Address Centric Routing: In *address centric routing* the routing is based on the fact that communication should take place between two endpoints identified by unique addresses, i.e. data packets are transferred from one addressable node, the source, to another addressable node, the destination.

In contrast, in *data centric routing* the focus is set on the data itself, i.e. it does not matter from which endpoint the data comes, but rather just to get the data from some node in the network to answer the posed question. In most cases queries with attribute-value pairs are used to get named data from the network.

Although in WSNs address centric routing can be used for several application scenarios, the data centric routing approach is often better suited because the sink that poses the question does not necessarily know which source can answer the question. Consequently, it is impossible for the sink to address the source directly that can answer the posed question.

Flat vs. Hierachical Routing: In *hierarchical routing* an additional layer of abstraction is added to the routing, normally by forming clusters. Among the nodes in the cluster a cluster head is chosen that is responsible for the nodes in the cluster and can be used for aggregation of data of the cluster's nodes. The transmission of data between two clusters can either be done via gateway nodes, i.e. nodes of two different clusters

which are neighbours, or via cluster heads. While for node to cluster head communication the radio range can be reduced to save energy, for cluster head to cluster head communication the radio range must be larger, resulting in a higher energy consumption. Often, cluster rotation is used, i.e. a new cluster head is elected after a while to distribute the higher energy consumption. Although, an hierarchical routing approach has the advantage that data aggregation leads to less transmissions and therefore to less energy consumption, the cluster heads require significantly more energy for managing the cluster.

In *flat routing* each network node is a peer with exactly the same functionality as all other nodes in the network. Due to the fact that all nodes have the same capabilities, single node failures do not have a huge impact. Moreover, the management and thus overhead is low because each node just needs to communicate with the nodes in its neighbourhood.

Single Path vs. Multiple Paths: In *single path* routing approaches, only a single path between the source and the destination is used. In general, this single path is the optimal path between source and destination, which provides the best performance regarding the chosen criteria.

In contrast, in *multi path* routing approaches multiple paths are used concurrently between the source and the destination. In most cases, multiple paths are found in the route discovery process by using some sort of flooding mechanism. The found paths can then be used to achieve a better load distribution, a higher data throughput or a better failure recovery.

One-to-One vs. Many-to-One vs. Many-to-One: Additionally to the common traffic patterns such as *one-to-one*, which is a simple connection between a source and a destination, and *one-to-many* traffic, i.e. one source is multicasting to a group of destination nodes, in WSNs the *many-to-one* traffic pattern is used, i.e. multiple sources are reporting events to a single sink that is interested in certain information.

Quality of Service vs. Best Effort: Depending on the application scenario the routing algorithm may have certain demands in terms of

Quality of Service (QoS). This is particularly important if real time data, such as audio or video streams, are transferred. In other cases a best effort service is sufficient, i.e. the delivery of packets without guarantee.

It is clear that the discussed routing characteristics are neither comprehensive nor mutually exclusive, but this overview can highlight the most important features and design choices that need to be considered when developing a routing protocol for WSNs.

3.1 WSN Routing Requirements

The described characteristics of WSNs leads to several requirements that should be taken into account for successful WSN routing algorithms (cf. [186]):

- **Optimality**: a (near) optimal path should be found between any source-destination pair regarding the desired criteria such as fastest transmission or most secure transmission. If no optimal path can be found between source and destination at least one path should be found (if the network is not partitioned) to guarantee the reachability within the network.
- **Robustness**: the routing algorithm needs to be robust so that unforeseen situations such as failures caused, on the one hand, unintentionally by nodes' movement, environmental changes or transmission issues and, on the other hand, failures caused intentionally by adversaries, cannot harm the overall functionality of the routing.
- **Distributed**: ideally the routing algorithm should be distributed so that no central entity is required. A flat hierarchy of nodes as peers make the routing decentralised.
- **Self-organised**: each node should work as a simple agent that can join the network to cooperate in the overall structure of the network.
- **Simplicity**: the routing algorithm should be simple in terms of low resource utilisation at each node and in terms of low protocol overhead.

- **Scalability**: due to the fact that in WSNs the number of sensors may become high, e.g. a few hundreds, the routing algorithm should be well-performing even when the amount of nodes is increasing.

- **Adaptivity**: the routing algorithm needs to be flexible so that it can adapt to a variety of different network situations in terms of network topology, but also data traffic changes. Both may even change during the network's lifetime.

- **Multi-hop**: due to the limited radio range of each sensor node the routing algorithm should be capable of making use of multiple hops to transfer a data packet between the source and the destination.

- **Multipath**: the routing algorithm can use multiple paths so that, on the one hand, the data transfer rate can be increased and, on the other hand, the data load is spread among the network nodes.

- **Energy efficiency**: due to the fact that the sensor nodes are restricted in terms of available energy resources, the routing algorithm itself should take the energy consumption into account.

- **Security**: the routing algorithm should also consider security mechanisms to mitigate attacks from adversaries. This is particularly important because data is transferred via wireless broadcast medium.

3.1 WSN Network Layer Attacks

Basically, attacks on WSNs can be classified into one or more of the general categories shown in table 3.1 (cf. [28, 30]).

As stated before, routing in WSN is one of the crucial services that should be protected. Therefore, in the following the most important attacks on the network layer in WSNs will be discussed:

There are several attacks that can be launched against the network layer in WSNs. Most of the attacks on the network layer can be classified into one of the following categories:

- *Information disclosure*: Disclosure of routing information by passive or active participation in the WSN

Attack	Description
Outsider vs. Insider attack:	In an *outsider attack*, a malicious node harms the WSN without being part of it. In contrast, in an *insider attack* the malicious node harms the WSN as (authorised) participant of it.
Physical vs. remote attack:	In a *physical attack* an adversary physically accesses the sensor node that should be harmed by tampering or destroying the sensor's hardware. In contrast, a *remote attack* is implemented from a (large) distance, e.g. by emitting a high-energy signal to interrupt the communication.
Passive vs. active attack:	In a *passive attack* an adversary just eavesdrops or monitors the communication within the WSN. In contrast, in an *active attack* the adversary directly influences the communication in the WSN by modifying, fabricating or suppressing packets.
Laptop-class vs. mote-class attack:	A *mote-class attack* is an attack against a WSN that is implemented from a mote, i.e. the attacking device is of same type of hardware as the sensor nodes that are targeted by the attack. In contrast, in a *laptop-class attack*, the adversary utilises a device which is superior, in terms of computational power and transmission power, to the sensor nodes of the attacked network.

Table 3.1: General types of attacks in WSNs

- *Physical attack*: Unauthorised access to sensor node through physical intervention
- *Energy exhaustion*: Intentional waste of energy resources by adversaries, e.g. by requesting unnecessary routes
- *Denial of service*: Flooding of the network with unnecessary routing requests
- *Spoofed, altered or replayed routing information*: Changing the routing behaviour by spoofing, altering or replaying routing information
- *Routing table overflow*: Flooding of the routing table by creating multiple non-existing routes to make the routing algorithm collapse
- *HELLO flood attack*: Intentionally inject bogus HELLO messages to remote nodes to confuse the routing protocol
- *Sybil attack*: Creating a large number of pseudonymous entities to gain a greater influence on the network

- *Sinkhole/ blackhole attack*: Trying to obtain all network packets in a certain network area by 'looking attractive' to surrounding nodes
- *Wormhole attack*: Making to nodes believe that they are neighbours, though they are far apart by tunnelling packets using a low latency link
- *Selective forwarding*: Forwarding only certain packets in the network to save resources

For an in-depth study of WSN network layer attacks see [30, 187–192].

3.2 Routing Based on Ant Colony Optimisation

As highlighted before, the desired key factors for WSN routing algorithms, such as self-organisation, robustness, adaptivity and scalability, can also be observed in nature in several contexts. Due to the fact that routing algorithms can be viewed as COPs, it seems to be a reasonable option to apply BIAs to the routing problem in WSNs. One of the most promising approaches of BIAs are ACO-based algorithms, which have been applied successfully to several COPs, such as the TSP, the knapsack problem, scheduling problems etc. As routing can be seen as COP, ACO-based algorithms can also be used to solve the problem of routing in WSNs.

To understand the growing interest in ACO-based routing algorithms in the following, first, a small introduction to the basic idea of ACO-based routing algorithms in WSNs is given. Afterwards, related works in the area of ACO-based routing algorithms are discussed, followed by MOACO-based routing approaches, i.e. ant-based approaches that take the optimisation of multiple-objectives into account.

3.2 Basic Idea of ACO-based Routing Algorithms in WSNs

For the application of ACO-based algorithms in the area of WSN routing there is some sort of mapping required, i.e. how the terms used in ACO for COPs can be transferred to the area of routing: first, the ants are modelled as data packets on the network layer. The moving of an ant from one node to another node is modelled as the sending of an ant

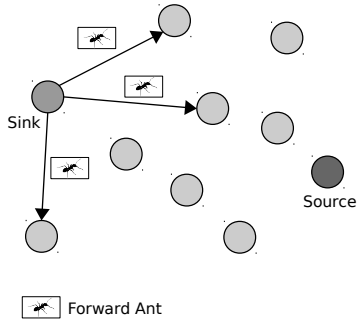
packet from one node to another node. The deposition of pheromone is done by modifying one or multiple pheromone matrices. Due to the fact that WSNs are acting completely distributed there are no central pheromone matrices, instead each sensor node needs to manage its own pheromone matrices for its neighbouring nodes. Furthermore, to avoid loops each ant needs to memorise the nodes it has visited already. This is done by storing a list of visited nodes in each ant packet at each time an ant arrives at the next node. The choice of the next hop is implemented on the network layer by applying the probabilistic rule provided by the ACO-based algorithm.

The basic idea of ACO-based routing algorithms in WSNs, i.e. how data packets can be transferred through WSN by using ants, is depicted in figure 3.1. The general process is as follows:

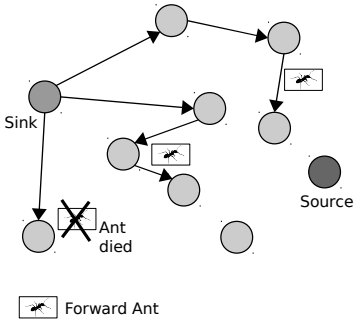
If a data packet should be transported from the sink to the source, but the sink does not have a route to the source yet, forward ants are sent out to discover new routes in the network (see 3.1a). Each forward ant moves by a probability rule through the network, i.e. each ant chooses next hops randomly biased by the pheromone along the path. The ant appends each visited node to the list of visited nodes, which is stored in the ant packet. As a result, at any time each ant is aware of the route it took so far. Ants can die due to interferences or running into a loop (see 3.1b)[1]. When all ants arrived at the destination[2], i.e. when the iteration is over, the 'optimal' route regarding a cost function is chosen from all routes received by the ants (see 3.1c). Depending on the chosen approach, one or more ants are converted to backward ants and sent back on the route they came from (see 3.1d). During the return, each ant places pheromone along the route it is using. In the following iterations this pheromone influences the forward ants that are sent out for discovery. To improve the quality of routes and to remove broken routes from time to time ants can be sent out in a new iteration.

[1] Another option would be that the ants that run into a loop would simply forget this part of the route and continue with the path discovery. However, to obtain a shorter end-to-end delay the dying of single ants is preferred in the presented approach.

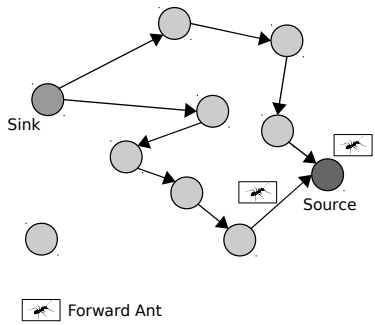
[2] When ants are dying, not all ants can reach the destination so it is reasonable to implement some other mechanism e.g. a timer that completes the iteration after a certain time.



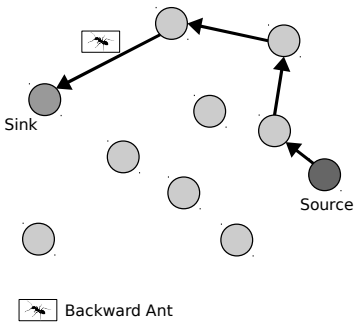
(a) Forward ants are sent from the sink towards the source to discover a route



(b) Forward ants are randomly choosing next hops biased by pheromone. Ants may die, if they run into a loop or due to interferences



(c) Of all forward ants reaching the destination, the 'optimal' path regarding a cost function is chosen



(d) One or more forward ants are converted to a backward ant, which is then sent back on the same way the forward ant came to the sink.

Figure 3.1: Basic idea of MOACO-based routing algorithms

In the following the related works in the area of ACO-based and MOACO-based routing are discussed, starting from its origin in wired networks to wireless networks such as MANETs and WSNs.

3.2 Wired Networks

The first ideas of using ACO-based algorithms for the routing in telecommunication networks emerged at the end of the Nineties. Due to the fact that at this time the majority of telecommunication networks were wired, first approaches dealt with applying ACO-based routing approaches to wired networks. The ACO-based routing algorithms were applied to circuit-switched as well as packet-switched telecommunication networks. Due to the fact that since then there were at lot of improvements on these first approaches, just for the sake of completeness the 'great grandfathers' of ACO-based routing are described in the following.

Circuit-switched Networks

One of the first ACO-based routing approaches in the area of circuit-switched telecommunication networks was the Ant-based Control (ABC) algorithm, proposed by Schoonderwoerd et al. [193]. The original idea of this approach was to achieve a decentralised and adaptive control system for the load balancing in circuit-switched telecommunication networks. A small number of experiments were conducted to test the basic idea of the proposed algorithm and as it showed some promising first results the ACO-based routing idea was picked up by other researchers.

Packet-switched Networks

With the huge growth of the Internet in the Nineties, packet-switched networks came to the fore. Dorigo and Di Caro [194] proposed with AntNet one of the first ACO-based routing approaches for packet-switched networks. The goal of the approach was to create a new adaptive routing algorithm for packet-switched communication that is robust and shows good performance. Simulations were conducted to compare AntNet to five (at this time) state-of-the-art shortest path algorithms (OSPF [195], BF [196], SPF [197], SPF_1F [197] and Daemon, an approximation of an ideal algorithm). The simulation results showed that AntNet was always

among the best performing algorithms. In contrast to the other algorithms, AntNet showed a robust behaviour by rapidly reaching a stable level of performance. As one of the first ant-based algorithms for packet-switched communication networks, AntNet was often used as a starting point for following researches and also for comparison purposes.

Almost at the same time Subramanian et al. [198] proposed two routing algorithms that are based on ACO. The first algorithm targets at telephone call routing in telephone networks with symmetric path costs, while the second algorithm is a multi-path routing algorithm for data networks with symmetric or asymmetric path costs. Convergence theorems for both algorithms were presented as well as an empirical evaluation. The simulation results show that the proposed algorithms provide good results compared to Distance Vector (DV) and Link State (LS) routing protocols.

Lu et al. [199] proposed an adaptive ant-based dynamic routing (ADR) algorithm for packet-switched wired communication networks to improve the QoS in the network and to reduce congestions. The algorithm is based on the idea of ACO. Simulations are conducted to compare ADR to AntNet [194]. The simulation results show that ADR outperforms AntNet [194] in terms of less delay and higher throughput, while having faster convergence.

3.2 Wireless Networks

With the spreading of wireless networks, the focus of research shifted from ACO-based routing for wired networks to the domain of wireless networks. At the beginning of the twenty-first century MANETs emerged as one of the pioneering technologies in this domain for which ACO-based routing algorithms were adapted and evaluated. A little later first applications of ACO-based routing algorithms for WSNs were proposed.

In the following, a selection of important related researches in the area of ACO-based routing algorithms for MANETs as well as WSNs are discussed.

ACO Routing Algorithms in MANETs

The idea of using ACO-based algorithms for the routing in MANETs matured during the last years so that there is a huge amount of existing researches that deal with this topic. In the following a selection of those approaches is discussed:

Güneş et al. [200] present the Ant-colony-based Routing Algorithm (ARA), an on-demand routing algorithm for MANETs. The main goal of this ACO-based routing approach is to reduce the overhead of routing. The simulation results show that ARA performs almost as good as DSR [185], while outperforming AODV [184] and DSDV [201]. An advantage of ARA is its small routing overhead, which is equal or smaller than the DSR routing overhead.

Marwaha et al. [202] propose Ant-AODV a hybrid routing technique for MANETs that is combining the on-demand capability of AODV [184] with a distributed topology discovery mechanism based on ant-like agents. The simulation results show that Ant-AODV can provide a low end-to-end delay and a high connectivity compared to AODV [184], which make Ant-AODV suitable for the transmission of real-time data and multimedia content. However, the low end-to-end delay and high connectivity comes with the cost of higher overhead caused by the ants. Although the goodput is almost the same as for AODV [184].

Hussein and Saadawi [203] discuss in their paper the Ant Routing Algorithm for Mobile Ad-hoc networks (ARAMA), which is a ACO algorithm-based routing approach considering the optimisation of more than one QoS parameter to achieve good network performance. The routing is based on a local normalised node index that is a weighted sum of the chosen optimisation parameters such as number of hops and remaining energy. The presented OPNet simulation scenario shows some interesting results, although there is no comparison to existing MANET routing algorithms. However, in a later paper Hussein et al. [204] test ARAMA in five scenarios with two different configurations considering the number of hops and the remain energy of the battery as input parameter. The results of the experiments show that in comparison to AODV [184], in ARAMA the first node failure time is higher. Also the maximum energy

standard deviation is about 60% lower in the case of ARAMA. Finally, the number of delivered packets is higher and the network failure time is lower than in AODV so that ARAMA outperforms AODV [184] in all tested scenarios.

Islam et al. [205] propose an on-demand routing algorithm called source update for MANET that is based on the ACO metaheuristic. The presented approach considers the MANET routing problem as irregular application of parallel computing. Thus, a parallelized version of the routing algorithm is developed and executed on a network of workstations using the Message Passing Interface (MPI). The experiment, considering all-pair routing between the nodes, shows that the parallel algorithm based on source update scales well for the increasing number of nodes and processors. Best results, in terms of execution time, were observed when the number of ants equals the number of processors. Another interesting observation is that the percentage of communication among the ants is much higher than the percentage of computation by each ant.

Heissenbüttel and Braun [206] present an ant-based routing algorithm for large-scale MANETs in terms of a large number of nodes and a huge coverage area. Before the ant-based routing algorithm is used the Topology Abstracting Protocol (TAP) is applied to obtain 'logical routers' and 'logical links' (can span multiple hops). On top of this overlay structure Mobile Ants Based Routing (MABR) is run to determine logical paths. Finally Straight Packet Forwarding (SPF) is used to transmit packets over such a logical link. However, in this paper the approach is discussed theoretically only; simulations were not conducted.

Baras and Mehta [207] propose the ant-based routing approach Probabilistic Emergent Routing Algorithm (PERA) for MANETs. In simulations PERA was compared to AODV [184] regarding throughput, goodput and end-to-end delay. The simulation results show that while the end-to-end delay was rather low for PERA, AODV [184] had a higher goodput particularly for scenarios with high mobility. The throughput for both algorithms was almost the same, though PERA performce slightly weaker.

Jawahar [208] proposes a probabilistic multi-path routing algorithm for MANETs inspired by the colony of harvester ants observed in nature. The

main focus of the approach is to optimise the following factors: routing efficiency, congestion avoidance and load balancing. The proposed routing is based on the pheromone count of a route which is defined as route efficiency metric (i.e. the product of all node congestion values of all intermediate nodes between source and sink) by the power of the length of the route. The presented experiment shows that in comparison to DSR [185], the presented approach has a similar performance in terms of delivery ratio and average hops, while the package overhead is higher. However, the congestion and network load is better for the new ant-based approach.

Zheng et al. [209] discuss a novel ant-based distributed route algorithm for ad hoc networks (ADRA). For their algorithm, the quality of the link as well as congestion is taken into account. To mitigate the impact of congestions an anti-ant is introduced that can be sent by intermediate nodes in the direction of the upstream to reduce the pheromone of the route, if a certain threshold is reached. The simulation results show that in contrast to DSR [185], the novel ADRA approach has a lower end-to-end delay. For dynamic networks ADRA performs better than DSR [185] in terms of a better packet delivery ratio, while the overhead in ADRA is less than in DSR [185].

Di Caro [210] discusses in his dissertation ACO and its application to adaptive Routing in telecommunication networks. This covers a wide spectrum of different applications presented by four different routing algorithms: two delivering best-effort traffic in wired IP networks, one that deals with QoS traffic in Asynchronous Transfer Mode (ATM) networks and one for best-effort traffic in MANETs. The two wired IP based algorithms were extensively tested and compared to popular state-of-the-art algorithms as well as the MANET approach, which is however still under development. For the QoS approach no results are reported because it has not been fully tested yet. In general, all tested ACO-based algorithms provide good results, particularly for the wired IP networks the ACO-based algorithms, which outperform the state-of-the-art-algorithms or perform on a similar level.

Di Caro et al. [136, 211, 212] propose AntHocNet a routing algorithm for MANETs that is based on the idea of ACO. The new approach is a hy-

brid algorithm that combines reactive components (for the path setup) and proactive components (for the path maintenance): while in the initial phase a purely ant-based approach is used to find a path between the source and the destination, for the path maintenance and improvement a combination of path sampling and slow-rate pheromone diffusion is used. To update the routing information the sampled information is distributed by ants using a special bootstrapping mechanism. The simulation experiments show that in comparison to AODV [184], AntHocNet has a better performance in terms of delivery ratio, average end-to-end delay and average jitter. The measured communication overhead is comparable between both algorithms.

Ahmed [213] proposes a routing protocol for MANETs that combines ant colony behaviour and queuing network analysis to evaluate end-to-end packet delay. A small experiment is conducted to test the proposed algorithm. Though the results show that the algorithm can deal with the tested dynamic mobility models, a general statement about its use cannot be made due to a missing comparison to other existing algorithms.

Kamali and Opatrny [214] present in their paper a Position based Ant Colony Routing Algorithm (POSANT) a new reactive routing algorithm for MANETs that combines the idea of ACO and the use of information about the nodes' positions as heuristic values to increase the routing efficiency. Additionally POSANT is developed as multipath routing algorithm, in contrast to most of the other position-based algorithms. The simulation results show that POSANT has a higher delivery rate with a shorter average packet delay than GPSR [215]. Furthermore, POSANT shows a faster stable behaviour in comparison to ANTNET [194].

In his thesis Ducatelle [216] considers an improved version of AntHocNet [136], an ACO-based hybrid approach that is combining reactive and proactive mechanisms. In contrast to the previous version of AntHocNet, multiple routes are set up in the reactive route setup process. Furthermore, a special mechanism is added that make ants choose as first hop only a node that has not been visited yet by another ant. This results in the creation of more disjoint paths and thus, in a more robust routing in the case of failures. However, to reduce the overhead in later experiments the reactive routing setup was restricted to find only a single route,

while for the proactive route maintenance multiple routes are obtained. Several simulations were conducted in different environments and evaluated regarding several performance metrics. The simulation results show that AntHocNet can outperform important reference algorithms such as AODV [184] and OLSR. Besides, some simulations in a realistic urban environment were carried out as well as the implementation of AntHocNet in a real world testbed is discussed.

Woo et al. [217] propose an ant-based routing algorithm for MANETs, which is based on AntHocNet [136]. The main goal of the proposed routing algorithm is to support multipath routes and to reduce the overhead that is required for the management of the ants. The simulation results show that the proposed routing algorithm can achieve a faster end-to-end delay and an improved packet delivery ratio with a reasonable control overhead in comparison to AntHocNet so that it outperforms AntHocNet for all tested metrics, particularly when the mobility of the nodes is high.

Liu et al. [218] propose a parallel ant colony algorithm (PACO) to establish multiple paths between a source and a destination node to improve the packet delivery ratio. The results of the simulations show that PACO reduces the route discovery latency and the end-to-end delay while providing a high connectivity. In the simulations PACO outperforms the routing protocols AODV [184] and DSR [185]. The parallel ant colony algorithm improves the probability of route discovery and the constringency speed.

Asokan et al. [219] propose a QoS Dynamic Source Routing protocol based on ACO, the so called Ant DSR protocol (ADSR). ADSR is a routing protocol that takes three QoS parameters into account namely delay, jitter and energy. In the simulations ADSR is compared to DSR [185] regarding several performance metrics such as end-to-end delay, throughput, routing overhead, jitter and residual energy. The simulation results show that ADSR outperforms DSR [185] in terms of delay, energy, jitter and throughput.

Yu et al. [220] propose a new hybrid routing algorithm for MANETs that is based on ACO, the so called ACO-AHR algorithm. It uses a hybrid approach in terms of combining a reactive path setup with proactive path probing and maintaining mechanisms. Simulations are conducted to com-

pare ACO-AHR to AODV [184]. The results of the simulations show that ACO-AHR outperforms AODV [184] in terms of end-to-end delay, packet delivery ratio and throughput.

Sivajothi and Naganathan [221] propose AntHocNetM, a routing algorithm based on ACO for multimedia communication in MANETs. Reactive as well as proactive components are applied in the algorithm. A reactive path setup is used to establish multiple paths between the source and the destination, which are subsequently used for the transmission of multimedia data. During the transmission the paths are monitored and improved in a proactive fashion. The simulation results for different scenarios show that AntHocNetM outperforms AODV [184] in terms of end-to-end delay, packet delivery ratio and jitter.

Wang et al. [222] propose HOPNET, a hybrid routing algorithm for MANETs based on ACO and ideas from the zone routing protocol (ZRP) [223]. The algorithm is based on the idea that ants hop from one zone to the next using local proactive route discovery, while reactive communication is used between neighbouring nodes. HOPNET is simulated and compared to AODV [184] and AntHocNet [136]. The simulation results show that HOPNET has a lower end-to-end delay, but higher control overhead and lower packet delivery ratio than AODV [184]. In contrast to AntHocNet [136], HOPNET performs much better, particularly for large networks.

Attia et al. [224] present two routing algorithms for MANETs that are inspired by ACO, a Hybrid Multi-Ant routing algorithm (HMAnt) and a Hybrid Multi-Ant algorithm with QoS provision (HMAnt-QoS). HMAnt utilises multiple paths that are established reactively, while their maintenance is done proactively. HMAnt-QoS additionally deals with certain QoS requirements of the incoming traffic. The simulation results show that HMAnt outperforms AODV [184], AntNet [194] and AntHocNet [136] in terms of end-to-end delay and packet delivery ratio, while having an acceptable routing overhead. HMAnt-QoS achieves the goal of providing paths that fulfil the request QoS constraints for real time applications.

Kadono et al. [225] propose an ACO-based routing algorithm for MANETs incoporating GPS. The focus of the routing approach is the

robustness of paths which is evaluated by each ant using GPS information of visited nodes. Corresponding to the degree of robustness pheromone is deposited. Furthermore, a prediction mechanism based on GPS information is used to predict possible disconnections. Based on the prediction pheromone will be transferred from links that are likely to be disconnected to other links so that alternative paths can be found more quickly. The simulation results show that in comparison to LAR [226] and AntHocNet [136] the proposed algorithm provides a higher packet delivery ratio with lower communication costs.

ACO Routing Algorithms in WSNs

Lately, there have also been several researches in the area of WSNs that make use of ACO algorithms for routing purposes. A selection of interesting approaches is presented in the following:

Zhang et al. [227] show in their paper why existing ACO-based algorithms do not work well for WSNs. Subsequently, three new ant-routing algorithms; Sensor-driven and cost-aware ant routing (SC), Flooded forward ant routing (FF) and Flooded piggybacked ant routing (FP); are presented. The simulation results show that the proposed ant-based algorithm perform well and each of them has certain advantages: while SC is energy efficient, FF has shorter delays and FP has the highest success rates.

Camilo et al. [228] present the Energy Efficient Ant-Based Routing Algorithm (EEABR), a WSN routing protocol based on ACO and optimised regarding the energy constrains of WSNs. In EEABR paths are optimised regarding the distance and the energy level. Simulations are conducted in which EEABR is compared to the basic ant-based routing algorithm (BABR) and the improved ant-based routing algorithm (IABR). In comparison to BABR and IABR, EEABR performs better in the static as well as in the mobility scenario in terms of minimising the communication load, while maximising the energy savings. However, due to the fact that EEABR is not compared to well-known routing protocols it is hard to say whether the presented results are significant.

Chen et al. [229] propose an improved ant-based routing algorithm for

WSNs that should overcome the flaws of conventional ant-based data-centric routing algorithms, such as dead-locks of forward ants or slow convergence. An additional ant type is introduced, the so called search ant, that provides prior information to following ants. To accelerate the convergence of the algorithm a global pheromone update strategy is proposed as well as a retry rule to avoid protocol dead-locks. Simulations are conducted with a Java implementation, but only different scenarios for the proposed routing algorithms are tested instead of comparing the results to existing routing protocols.

Iyengar et al. [230] investigate in biologically inspired mechanisms and associated techniques that can be used for the routing in WSNs including ant-based approaches and genetic approaches. The mathematical background of biological computations in the area of WSNs is presented as well as some ideas for a generalised routing framework for WSNs. However, the paper has a survey-like character so that rather existing approaches and ideas are discussed, but no experiments were conducted.

Ghasem Aghaeu et al. [231] present two adaptive routing algorithms based on swarm intelligence: Adaptive Routing (AR) and Improved Adaptive Routing (IAR). The originally for packet-switched communication developed routing algorithm [199] AR was slightly modified and combined with the routing algorithm proposed in [227] to make it suitable for WSNs. For IAR some modifications were made to the AR algorithm including the addition of a coefficient that takes the cost between the neighbouring node and the destination node into account. In simulations AR and IAR were compared to Basic Ant Routing [194], Sensor-driven Cost-aware Ant Routing, Flooded Piggybacked Ant Routing [227]. The simulation results show that AR has a good performance and IAR an even better performance in terms of low energy consumption, high energy efficiency and less latency, while having a high success rate.

Wang et al. [232] propose ACLR, a novel adaptive intelligent routing scheme for WSNs that is based on ACO. In contrast to other ACO-based routing algorithm the search range of an ant is limited to a certain subsets of neighbouring nodes for selecting the next hop. A new probability transition rule, which is used by the ants to select the next hop, is introduced that is based on the residual energy and the global as well as

the local location information of the nodes. Simulations were conducted in which ACLR was compared to Basic Ant Routing (BAR), Sensor-driven Cost-aware Ant Routing (SCAR), Flooded Piggybacked Ant Routing (FPAR) [227] as well as IAR [231]. The simulation results show that ACLR has a better performance in terms of energy consumption, energy efficiency and packet delivery latency.

Ren et al. [233] propose a multipath routing approach based on Ant Colony System [117] (MACS) for WSNs with the purpose of increasing the overall lifetime of the network. The proposed routing algorithm is compared to Directed Diffusion (DD) [234], ACS and Max-Min Ant System (MMAS). The simulation results show that MACS has the lowest energy consumption compared to the other algorithms as well as the lowest average transmission delay.

Wen et al. [235] propose a novel Energy*Delay model based on ant algorithms (E&D ANTS) with the goal of providing a real-time data transmission service instead of maximising the lifetime of the network. A reinforcement learning algorithm is used to train the model regarding the trade-off between energy and delay. Simulations are conducted to compare the proposed E&D ANTS algorithm to AntNet [194] and AntChain [236]. The simulation results show that in the Energy*Delay comparison E&D ANTS performs better than AntNet [194] and AntChain [236]. Furthermore, the routing load for E&D ANTS is the smallest of all tested algorithms.

Okdem and Karaboga [237] propose an ACO-based routing algorithm for WSNs consisting of stable nodes. To consider the nodes' energy consumption the heuristic value of the ACO-based algorithm is used to take the energy consumption into account. Simulations were conducted to compare the proposed routing algorithm to EEABR [228] using MICAz' mote specifications for the physical layer. The proposed routing approach outperforms EEABR [228] in terms of a significant reductions of the energy consumption. Furthermore, the ACO-based routing algorithm was implemented on a small sized hardware component as a router chip. The hardware implementation showed some promising results in terms of quick response times of the router chip.

Hui et al. [238] propose an ACOs-based routing approach for WSNs con-

sidering path delay, node energy and the frequency on which a node is acting as router. The energy level of the nodes is expressed as heuristic value. The main goal of the approach is a dynamic and adaptive routing behaviour that can effectively balance the power consumption in the WSN to increase the overall lifetime of the network. The proposed routing algorithm was compared to EAR [239] and SPEED [240] regarding average energy consumption and node operational time. The simulation results show that the proposed ACO-based algorithm has the lowest average energy consumption and the highes node operational time for all tested scenarios so that the life time of the network could be improved significantly.

Saleem et al. [241] present an overview of recently proposed ACO-based routing protocols for WSNs. Based on this a new ACO-based routing algorithm for WSNs is proposed that takes delay, energy and velocity as relevant routing features into account. The main goal of the proposed algorithm is to improve the overall data throughput, e.g. for real-time traffic, while the energy consumption of the routing should be minimised. Furthermore, a reinforcement approach is used to get a superior optimal decision and a loop avoidance mechanism is introduced to avoid deadlocks. Although a simulation section is provided, the results are rather limited to a special scenario and are not compared to other existing routing protocols. In [242] the basic idea of this approach is extended to support multipath routing in WSNs with the goal of maximising the data throughput, while minimising the data loss rate.

Wang et al. [243] propose EAQR, a new routing protocol for WSNs that is based on ACO. The main goal of the proposed routing protocol is the provision of QoS and a balanced energy consumption over the whole network. The proposed algorithm takes the minimum path energy as well as the distance in terms of number of hops into account. Furthermore two heuristics are proposed: one for real-time (RT) traffic and the other for best-effort (BE) traffic. Different simulations were conducted to test the different variations of the algorithm and compare them to EEABR [228]. The simulation results show that EAQR outperforms EEABR [228]: a lower delay for real-time traffic could be achieved and the network lifetime could be increased.

Saleem et al. [244] propose BIOSARP, a ACO-based routing approach for WSNs based on a cross layer design. The link quality, the energy level and the velocity are taken into account to find the optimal route. Additionally, the signal strength, the remaining power and time stamp metrics are used as information from the physical layer. The proposed routing protocol is targeting the improvement of the overall data delivery ratio, particularly for real time traffic. Simulations were conducted to compare BIOSARP to RTLD [245], a protocol that considers the same parameters to make next hop decisions in WSNs. The simulation results show that BIOSARP has a higher delivery ratio than RTLD [245], but consumes a bit more energy. Besides, a small test-bed scenario was created with 6 Telosb nodes. The results of the experiment show that the simulated packet delivery ratio is slightly higher (5%) than the delivery ratio measured in the test bed.

Jiang and Hong [246] propose the ACO-based Energy-Balance Routing Algorithm (ABEBR) for WSNs to balance the energy consumption in the network. A new pheromone update operator was introduced that includes energy consumption and the number of hops into routing decisions. Experiments were conducted to compare ABEBR to LEACH [247], Directed Diffusion (DD) [234] and Flooding. The simulation results show that ABEBR provides the lowest energy consumption rate for all tested network scales. Also the lifetime of ABEBR networks is higher than that of the compared algorithms.

Yang et al. [248] propose a multipath routing protocol (MRP) for WSNs based on clustering and ACO. The goal of the routing approach is to maximise the network's lifetime and to reduce its energy consumption. The proposed routing algorithm was compare to TEEN [249], MP [250] and MACS [233]. The simulation results show that in comparison to the other algorithms MRP could efficiently balance the energy consumption so that the network's lifetime could be increased significantly, while MRP has the lowest energy consumption.

Okazaki and Fröhlich [251] propose the Ant-based Dynamic Zone Routing Protocol (AD-ZRP), a routing protocol for WSNs that is based on HOPNET [222]. The proposed routing protocol combines the idea of ACO-based routing with the utilisation of a dynamic zone-based approach. In comparison to HOPNET [222], AD-ZRP uses dynamic zones to minimize

the latency while reducing the network overhead. Furthermore, instead of using two routing tables, one for intra and one for inter zone communication, in AD-ZRP only one routing table is used to reduce the complexity. In the conducted experiments AD-ZRP was compared to the original HOPNET [222] routing algorithm. The simulation results show that AD-ZRP outperforms HOPNET [222] in terms of better data delivery ratio, smaller routing overhead and a better congestion avoidance for dynamic topologies.

3.2 Multi-objective Ant Colony Optimisation Algorithms for Routing

As the diversity of presented approaches has shown, the idea of using ACO-based routing algorithms for dynamic communication networks seems to have become rather popular over the last years. In contrast, the routing approaches that optimise multiple objectives simultaneously are still rare when considering the 'real' MOACO-based approaches that are not simply combining multiple objectives into a single objective function as used in many other approaches e.g. [221, 252–254] and a lot more.

A few researches that touch the research area of MOACO-based routing are discussed in the following:

Cardoso et al. [165] are some of the first researchers that proposed to use ACO algorithms for multi-objective network optimisation. The authors propose a computational model named MONACO that is based on the classic AS approach which was extended to support multiple pheromone matrices, for each objective one matrix. Each ant, represented by a message, chooses the next hop by a probabilistic rule that takes into account the multiple pheromone values as well as a single heuristic information. MONACO was implemented in C++ as proof-of-concept to test the general idea. Small tests with 18 nodes and 27 edges as well as with 25 nodes and 40 edges were conducted that show that the basic idea seems to be working, although no further generalised conclusions can be drawn from these small tests.

Sim and Sun [255] present in their paper the Multiple Ant Colony Optimization (MACO) algorithm in which multiple ant colonies are used for routing and load balancing. However, just the basic idea is discussed, but

there are no simulation results presented.

In the closely related domain of multicast traffic distribution, Pinto et al. [256] present a new ACO-based algorithm for the multicast traffic engineering problem in networks that takes multiple objectives into account. The proposed algorithm is inspired by the MOACS approach presented by Baran et al. [164]. Three objectives are considered the cost of the multicast tree, the average delay and the maximum end-to-end delay. Experiments were conducted to test the proposed MOACS-inspired algorithm as well as MMA [257] on a NTT network [258] with 55 nodes and 144 links. The simulation results show that MOACS is able to find 69.9 % of the best solutions in average, while MMA [257] found 42.1 %. In proceeding researches [167, 259] the experiment was extended by considering additionally the maximum link utilisation of the tree as a fourth objective. Furthermore, M-MMAS is presented, an MMAS-inspired [114] approach, which is then tested and compared to the previously mentioned approaches. The conducted experiments show that the MOACS found better solutions in average than M-MMAS and MMA [257].

3.3 Summary

In this chapter, the application of ACO-based algorithms in the area of WSNs routing was discussed. In the first step, the basic topic of WSNs routing was presented, including the basic characteristics of the routing in WSNs, a taxonomy of WSN routing algorithms as well as general requirements for the routing in WSNs. After that, some network layer attacks were discussed that can be launched in WSNs to harm the network and thus, should be considered when developing a WSN routing algorithm. In the second section, related works that make use of ACO-based routing algorithms were discussed, starting from the origin of ant-based routing algorithms in wired networks to their extensions to the wireless domain, particularly to the area of MANETs and WSNs. Finally, related works that make use of MOACO-based routing algorithms were discussed.

Concluding, from the findings in the related works it can be said that the use of ACO-based algorithms for the routing became popular over the last few years, starting from the wired to the wireless domain. A general

observation that can be made on the investigated related works is that in most cases ACO-based routing algorithms outperform the well-known routing approaches, such as AODV, DSR etc., in terms of end-to-end delay and packet delivery ratio. On the other hand, ACO-based routing is causing a higher control overhead in comparison to the other routing protocols. This is often based on the fact that ACO-based routing algorithms are mainly relying on unicast communication, while other routing protocols, such as DSR, are for the most part based on broadcast mechanisms. In contrast, in the area of MOACO-based routing algorithms only a very few related works can be found. Consequently, the answering of the posed research questions in this thesis can contribute to the body of knowledge in this research area.

4

Implementation

> Plan specifically so you can implement flexibly.
>
> Dallin H. Oaks (1932 -)

In this chapter, the implementation of the Multi-objective Ant Colony Optimisation Routing Framework for WSNs (MARFWSN) and its related protocols will be discussed. First, the problem definition that is precisely specifying the problem that should be solved in this thesis will be presented. Afterwards, the used methodology and the basics of the chosen simulation environment OMNeT++ [260] will be explained. Subsequently, the implementation of the application layer and the network layer will be discussed in detail. The latter includes an in-depth discussion of MARFWSN as well as a DSR-based routing protocol, which is used for comparison. Finally, the implementation chapter will be summarised.

4.1 Problem Definition

The main problem that is considered in this thesis is how the routing in dynamic WSN networks[1] can be optimised, while at the same time the network can be protected against insider attacks.

In the pursued approach the routing in WSNs is considered as MCOP taking several objectives, such as energy, duration and trust, into account. While the energy objective is responsible for a long-lasting liftetime of the WSN, the duration in terms of end-to-end delay aims at using the fastest routes, resulting in quick response times. The trust objective is meant to influence the finding of trustworthy routes so that misbehaving nodes are preferably avoided in the routes. To solve the routing MCOP different MOACO-based algorithms are examined for their suitability to optimise

[1] The term 'dynamic' in this context includes the joining and leaving of nodes and temporarily unavailable nodes, rather than moving nodes because the nodes' positions are static in the considered scenarios.

the routing in WSNs.

It is assumed that the MOACO-based algorithms have a similarly good performance as existing WSN routing algorithms, while having slightly more overhead due to the continuous use of unicast messages. At the same time, it is expected that more trustworthy routes will be found due to the consideration of trust as optimisation objective.

4.2 Methodology

Due to the fact that a physical deployment of sensor nodes make the reproducibility of experiments too difficult and the use of an algebraic analysis approach would have too many unrealistic and incomplete assumptions about the WSN, in this thesis a simulation-based approach is pursued to test the proposed Multi-objective Ant Colony Optimisation Routing Framework for WSNs (MARFWSN).

4.2 Simulation Environments for WSNs

Different simulation environments with their corresponding frameworks that can be used for the simulation of WSNs were compared [13]. An overview of these simulation environments is given in table 4.1.

Simulation Environment	Version	License	Programming Language	WSN support
GloMoSim/ QualNet	2.0 (Dec 2000)/ 5.0 (Nov 2009)	free for academic research/ commercial	C and Parsec	GloMoSim: basic mobility and radio propagation models; 802.11; QualNet: additionally battery and energy model; ZigBee → GloMoSim seems to be outdated; QualNet seems to be more up-to-date, but commercial

OPNET Modeler Wireless Suite	16.0 (Dec 2009)	commercial	configuration by GUI; internals C++	Different propagation models; 802.11, ZigBee; some MANET protocols, but no special WSN support → powerful tool with a nice GUI, but expensive
TOSSIM (part of TinyOS)	2.1.1 (Apr 2010)	BSD	nesC	All TinyOS-based WSN protocols can be simulated with TOSSIM without modifications → good approach especially if implementation should also be used with TinyOS-based nodes
OMNeT++	4.0 (March 2009)	Academic Public License	basic modules C++; larger structures NED	Several frameworks that add WSN functionality to OMNet++ such as MiXiM, Castalia, etc. → active project with a huge user base; Eclipse-based IDE for development
NS-2	2.34 (Jun 2009)	GPL	C++; configuration OTcl	Huge amount of protocols available contributed by NS-2 users → complex configuration; unclear situation due to large number of different user contributed implementations

Avrora	Beta 1.7.106 (Aug 2008)	BSD	AVR microcontroller binaries	Particularly for programs written for AVR microcontroller with support for Mica2 and MicaZ → very special application area; project seems to be still active - still changes in CVS
J-Sim	1.3 + patch4 (Jul 2006)	BSD-like	Java; configuration Tcl/-Java	Includes sensor network package containing models such as propagation, battery, radio model and sensor protocol stack including AODV and 802.11 → project seems to be abandoned
ATEMU	0.4 (2004)	BSD	AVR microcontroller binaries	Complete emulation of the AVR instruction set with partial Mica2 support; TinyOS based code can be run → very special application area; slow simulation speed; project seems to be abandoned
EmStar	2.5 (Oct 2005)	*unknown*	C	Provides network functionality for wireless embedded systems; EmTOS can be used to run TinyOS applications as EmStar module → project seems be abandoned (download links broken)

SENS	jan31-2005b (Jan 2005)	*unknown*	C++	Provides very basic network and physical layer support. Source can be compiled for TinyOS. → project does not seem to be developed any further
SENSE	3.1 (Nov 2008)	BSD-like	C++	Includes battery and power models, MAC layers (802.11) as well as network protocols (AODV, DSR, SSR, SHR) → does not seem to be developed any further
Shawn	Continuous SVN development (May 2010)	BSD	C++	Algorithmic approach that concentrates on lower layers, no special WSN protocols → very active project - lot of recent changes in SVN

Table 4.1: Overview of simulation environments for WSNs (cf. [13])

Based on this comparison OMNeT++ was selected for the implementation of the experiments in this thesis. As addtional framework MiXiM [261, 262] was used to support wireless and mobile functionality in OMNeT++. In the following a short introduction to OMNeT++ as well as the MiXiM framework will be given.

4.2 OMNeT++

OMNeT++ [7, 260, 263, 264] is an object-oriented discrete network simulation framework that can be used to simulate various network scenarios. The architecture of OMNeT++ is rather general so that various problem domains can be simulated such as protocol modelling, validation of hardware architectures and modelling of wired and wireless communica-

tion networks. It is highly portable so that it can be run on the most common operating systems such as Windows, Linux and Mac OSX. The current stable version 4.2 of OMNeT++ was released in November 2011, though new test versions are released every few months. OMNeT++ is free for academic and non-profit use – for commercial purposes OMNEST, a commercially supported version, can be licensed by Simulcraft Inc [265].

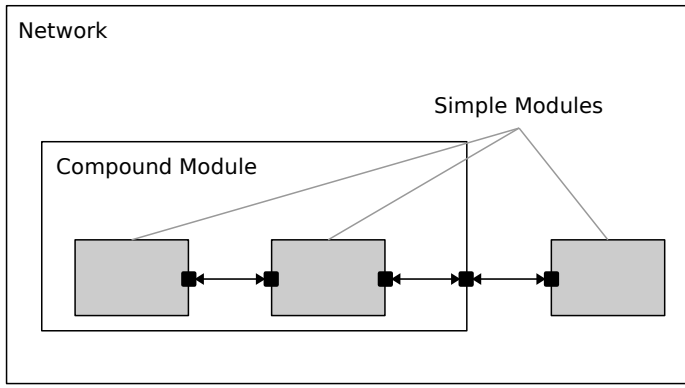


Figure 4.1: OMNeT++: simple and compound modules (cf. [7])

One of the fundamentals of the OMNeT++ framework is its component-based architecture for simulation models: a model can be combined in various ways from reusable components, so called *modules*. The modules can be connected using *gateways* and multiple modules can be combined to form a *compound module* (see figure 4.1) – the nesting depth is unlimited.

The communication between modules is done via message passing, where each message can contain arbitrary data structures. The messages can be passed either using pre-defined paths via gateways or a direct connection to the destination. Each module can have several parameters to customise its behaviour and/ or customise the topology of the model. The modules at the lowest level of the hierarchy, also referred to as *simple modules*, are programmed in C++ and make use of the *simulation library*. The larger components are assembled using the high-level language NED.

A simulation with OMNeT++ can be run utilising various interfaces:

from a simple command-line interfaces, which well-suits batch execution, to a tcl-based graphical animated user interface, which can be used for debugging or demonstration purposes. The parameters of each module can be easily adjusted by the `omnetpp.ini` configuration file, which is read for each simulation run. As a result, different configurations can be tested without recompiling the simulation again. Starting from version 4.0, OMNeT++ provides an Eclipse-based simulation IDE (see figure 4.2) to replace the previous standalone GUI programs `gned`, `scalars` and `plove`.

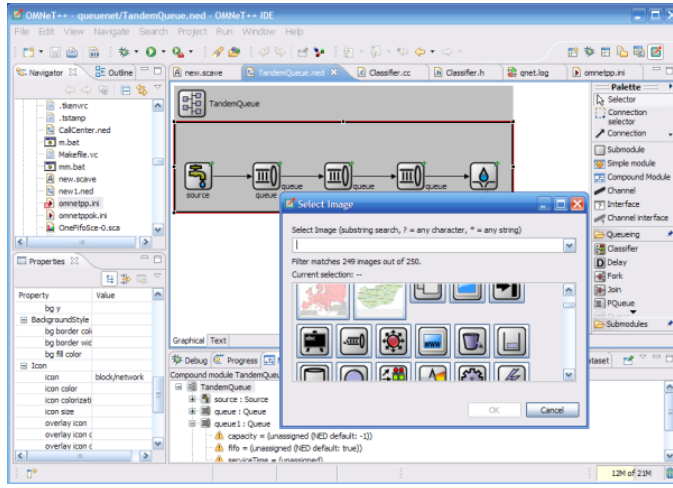


Figure 4.2: OMNeT++ 4.0 IDE [8]

OMNeT++ also supports the execution of parallel distributed simulations by providing several communication mechanisms between partitions. To run a parallel distributed simulation no special modifications to the models are necessary – only the configuration has to be modified correspondingly.

There are a couple of simulation frameworks that enable OMNeT++ to be used for WSNs. The most common of these frameworks are discussed in [13]. One of the most up-to-date frameworks is the MiXiM [261] framework that provides some mechanisms to support the lower layers of WSNs. Therefore, in the following subsection the MiXiM framework for OMNeT++ will be discussed in detail, which is then used for the implementation of the simulations in this thesis.

MiXiM framework for OMNeT++

MiXiM [261, 262], the abbreviation of **mix**ed s**im**ulator, is a framework for OMNeT++ that provides wireless and mobile functionality. MiXiM combines multiple existing frameworks that were developed in this area to a common approach by providing detailed models, protocols and the supporting infrastructure. The general structure of MiXiM is used from the Mobility Framework (MF) [266], the radio propagation models from CHannel SIMulator (Chsim) [267] and the protocol library from MAC Simulator & Positif [268] as well as the Mobility Framework (MF) [266]. The current stable version of MiXiM 2.2, released in November 2011, was recently updated to support OMNeT++ 4.2.

The MiXiM framework can be divided into five basic components (cf. [262]):

- **Environment model**: Provides 'relevant' parts of the real world, such as obstacles, that influence wireless communication
- **Connectivity and mobility**: Tracks the movement of the nodes and the corresponding connectivity among neighbouring nodes
- **Reception and collision**: Takes care of changes of the signal on the way from the sender to receivers; also considering multiple sending nodes
- **Experiment support**: Supports different evaluation methods
- **Protocol library**: Includes a set of already implemented protocols that enable researchers to compare results based on common protocols

Additionally to the combination of existing approaches MiXiM was extended to support features such as full 3D support, models for walls and obstacles, support for attenuation of radio signals, different frequencies and transmission media, full multi-channel support in space and frequency (enabling Orthogonal Frequency Division Multiplexing (OFDM) and Multiple Input and Multiple Output (MIMO) simulations) as well as many MAC protocols including IEEE 802.11 [269] and IEEE 802.15.4 [270].

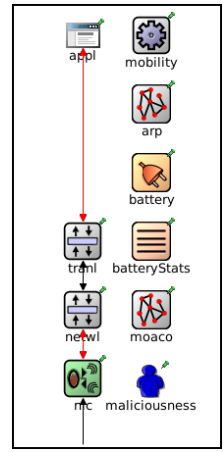


Figure 4.3: MiXiM: base node

The general structure of a node using the MiXiM framework (see figure 4.3) is corresponding to the common OSI network model [271], i.e. the node utilises the following layers: application layer (`appl`), transport layer (`tranl`), network layer (`netwl`) as well as the physical layer and the MAC layer which are combined in the Network Interface Card (NIC) module (`nic`) to enable a strong coupling regarding the used communication technique[2]. The adjacent layers are connected by `gates`, one for data messages and the other for control messages. Control messages are used to trigger certain actions in adjacent layers, for instance, the MAC layer can request the physical layer to perform carrier sensing.

Furthermore, the node contains a mobility module (`mobility`) for the movement of nodes and objects, a battery module (`battery`) and a battery statistics module (`batteryStats`) dealing with energy related issues and an arp module (`arp`) for the address resolution between the NIC and the network layer.

Additionally to the described layers that are provided by the MiXiM framework, a MOACO module (`moaco`) and a maliciousness module (`maliciousness`) are introduced by the presented approach. The MOACO module is a supportive module that provides the MOACO-based

[2]As a consequence, a node could have multiple NICs, e.g. 802.11 and Bluetooth, which is however not pursued in the presented approach.

algorithms that can be used by the network layer. The maliciousness module assigns at the beginning of the simulation a trust value, from the range between 0 and 1, to the node, where 1 means completely trustworthy and 0 means untrustworthy[3]. If a node has a trust value below 0.5 the node will be malicious, i.e. it will drop incoming packets with the specified probability. Besides, the maliciousness module has influence on the MOACO-based algorithms in terms of being one of the considered optimisation objectives.

Figure 4.4 shows the `BaseNetwork` scenario that is provided as one of the examples in the MiXiM framework. The screenshot of the `tkenv`-GUI shows the network topology including the `BaseNodes`; the `World utility module` that provides the global environment parameters, e.g. size of the simulated area; and the `ConnectionManager` that dynamically adapts the connections between nodes. Moreover, it is possible to add objects, such as `ObjectHouse` or `ObjectWall`, to the scenario and an `ObjectManager` that decides which objects are interfering. However, for reasons of simplicity additional objects and their interferences are omitted in the following simulations of this thesis.

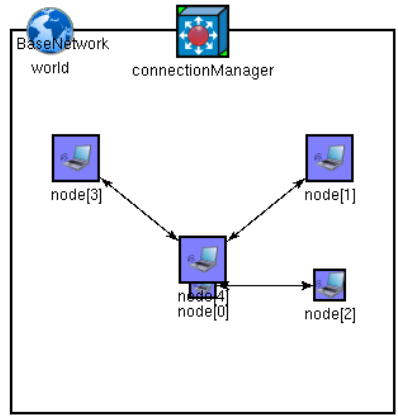


Figure 4.4: MiXiM: base network

The protocol library that is provided by the MiXiM framework can be divided into two parts: the base framework and the protocol library.

[3] In the future a TRS could be implemented that adapts the trust values dynamically depending on the behaviour of the nodes.

While the base framework provides general functionality; such as the connection management, mobility and wireless channel modelling; the protocol library provides a set of standard protocols such as 802.11b/g and 802.15.4 for the MAC layer or the 'Ley Line Routing Protocol' for the network layer.

4.3 Implementation of the Application Layer and the Network Layer

In this section the implementation of the application layer and the network layer are described in a top-down fashion, as implemented in OMNeT++ using the MiXiM framework. The description of the application layer includes the basic application protocol that is used for the simulations as well as the traffic generator that is used to imitate the querying of multiple sources by a sink node. For the network layer Multi-objective Ant Colony Optimisation Routing Framework for WSNs (MARFWSN), which includes the different MOACO-based algorithms, is discussed as well as the DSR protocol, which is used for comparison in the later experiments. Both protocols on the network layer use the assumption of symmetric connections, i.e. if a sensor node A can receive messages from sensor node B, then B can also receive messages from A respectively. Due to the fact that the lower layers are used from the MiXiM framework (`Nic802154_TI_CC2420`) without any modifications, no further details are provided here. For more information about the lower layers see the MiXiM documentation [261].

The rest of this section is organised as follows: first, the details of the application layer; and then, the details of the network layer are discussed.

4.3 Application Layer

To test the WSN routing functionality of the proposed MARFWSN, on top of the network layer a simple application layer is developed that imitates a simple WSN scenario application namely the querying of multiple sources for data by a sink node.

Basic Functionality

The basic functionality of the application protocol is to allow a sink, which has an interest in certain information, to send a query to one or more sensors which have some knowledge about this information, the data sources. As a consequence, the sink sends a request message (`GetDataReq`) in the direction of each data source. The request message is forwarded, as presented later in this chapter, by the underlying routing algorithm until it reaches the data source. On receiving the request, the data source creates a response message (`DataResp`) with the desired information and sends this response message back towards the source.

Application Message Flow

Figure 4.5 shows a simple request-response application message flow between a sink and a source using multiple intermediate nodes.

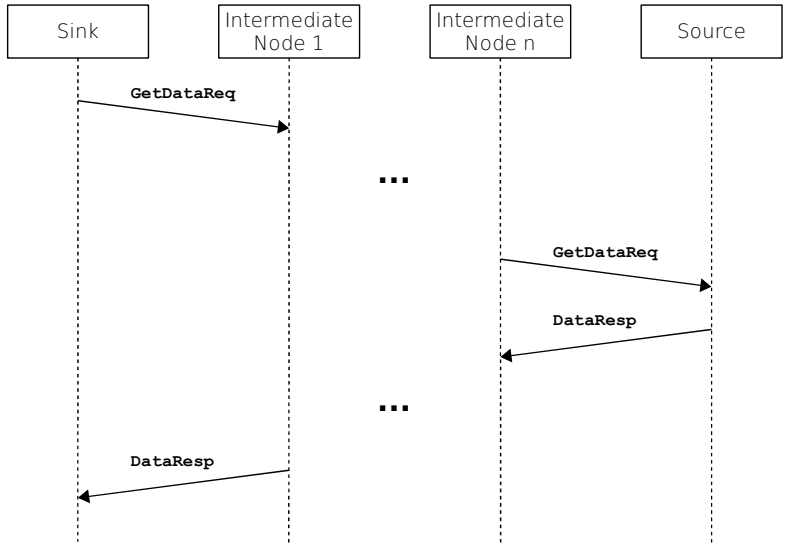


Figure 4.5: Application protocol message flow

```
packet ApplPkt
{
    int destAddr = -1; // destination address
    int srcAddr = -1;  // source address

}
```

Listing 4.1: `ApplPkt` message

Application Messages

The application messages `GetDataReq` and `DataResp` are both derived from the basic application packet class (`ApplPkt`) that is provided by the MiXiM framework (see figure 4.6).

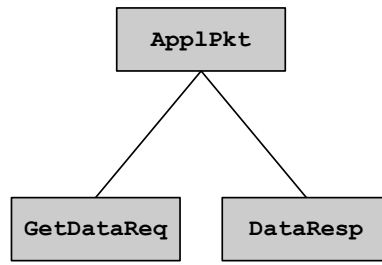


Figure 4.6: Application messages

The parent class `ApplPkt` packet (listing 4.1) provides just the basic packet fields `destAddr` and `srcAddr` for the destination and the source address of the packet, both as application layer addresses.

```
packet GetDataReq extends ApplPkt
{
    int interval;        //interval in which data should be
        sent
    simtime_t expiresAt; //time when get data request should
        expire
}
```

```
packet DataResp extends ApplPkt
{
    int dataValue; //simple data value that is the payload
    int payload[]; //some additional payload data to make it
        more realistic
}
```

Listing 4.3: `DataResp` message

Listing 4.2: `GetDataReq` message

The derived `GetDataReq` packet (listing 4.2) provides additionally two fields: the `expiresAt` and the `interval` field. The `expiresAt` field specifies the point in time until which data should be sent from the source to the sink. The `interval` field specifies in which interval those data should be sent.

The derived `DataResp` packet (listing 4.3) provides the `dataValue` field and the `payload[]` field. The `dataValue` field contains the data that should be sent back to the sink, which provides some human-readable value. Furthermore, the `payload[]` field is used to simulate some additional payload bytes that may occur in a real application, e.g. if more complex data such as arrays of measurements should be transferred.

Traffic Generator

To generate network traffic with the described application layer protocol, a traffic generator is created, which can be assigned to each node in the WSN to handle the application traffic. The traffic generator has basically two usages: the generation of `GetDataReq` messages and the generation of `DataResp` messages.

Generation of GetDataReq Messages: The generation of `GetDataReq` messages is activated for nodes that are interested in certain data of one or multiple other nodes, i.e. the sink. It can be simply enabled by setting the `active` flag to `true`. An overview of the other parameters that influence the generation of `GetDataReq` messages is shown in table 4.2.

Parameter	Description
`payloadSize`	Number of random bytes that are generated as payload
`activated`	Flag to turn the traffic generator either on or off
`initTime`	Initial time of waiting before the generation of `GetDataReq` messages begins.
`getDataReqGenerationInterval`	Interval the `GetDataReq` messages should be generated
`dataReqInterval`	Interval for sending `DataResp` messages from the final destination, resulting from `GetDataReq` message arriving at the final destination
`dataReqDuration`	Duration of sending `DataResp` messages from the final destination, resulting from `GetDataReq` message arriving at the final destination
`dstAddresses`	List of destination addresses to which the sink should send `GetDataReq` messages, leave this empty for random destinations
`randomDstAddressesCount`	Number of random destinations (used in random mode)
`trafficType`	Type of the traffic that is generated. This can be one of the following: `uniform`, `exponential`, `periodic`

Table 4.2: Traffic generator parameters for the generation of `GetDataReq` messages

Generation of DataResp Messages: All nodes were assigned a traffic generator to answer `GetDataReq` messages, i.e. behave like a data source: if such a node receives a `GetDataReq` message, it reacts to it by sending a `DataResp` message for the duration and with the frequency specified in the received `GetDataReq` message[4].

[4]In the current implementation the 'requested data' is an integer value that is increased by one each time a new `DataResp` message is sent. The payload field is filled with random data to imitate a larger payload.

4.3 Network Layer: Multi-objective Ant Colony Optimisation Routing Framework for Wireless Sensor Networks

The network layer is responsible for the forwarding of packets via multiple hops from the sink to the source and *vice versa*. In this thesis an MOACO-based routing approach is presented that makes use of ant-based routing algorithms that are capable of taking multiple objectives into account in the route optimisation process. The routing problem in WSNs is considered here as MCOP in terms of a minimisation problem.

Instead of implementing a specific MOACO-based routing algorithm, the Multi-objective Ant Colony Optimisation Routing Framework for WSNs (MARFWSN) is implemented, a framework for MOACO-based WSN routing algorithms. The framework provides the fundamental ant-based routing functionality and a docking site to which different MOACO algorithms can be docked and then used in the context of WSNs routing.

The details of the implementation of MARFWSN as well as the related challenges, such as finding a node's neighbours, will be discussed in the following.

Generic Multi-objective Ant Colony Optimisation Algorithm

In the first step, based on the identified components of MOACO algorithms (see table 2.4) and inspirations from the general MOACO framework proposed by Lopez and Stützle [155], a generic MOACO algorithm is developed that takes Pareto optimal solutions into account (see listing 4.4).

The proposed generic MOACO algorithm supports multiple colonies and correspondingly multiple pheromone and heuristic information that are assigned to each ant colony. The multiple colonies are interacting with each other, on the one hand, by exchanging solutions to update pheromone information and, on the other hand, by using a common archive of non-dominated solutions.

In the `initialization` function the pheromone and heuristic matrices are set to their initial values. Furthermore, the set of weights is defined that is used in the algorithm.

```
initialization()
while (termination condition not met):
    for colony in colonies:
        for ant in colony.get_ants():
            weight = next_weight(weights, ant, iteration)
            tau = aggregation(weight, tau_matrices)
            eta = aggregation(weight, eta_matrices)
            s   = probabilistic_solution_construction(tau, eta
                )
            s   = weighted_local_search(weight, s) #optional
            global_solutions.append(s)

    for colony in colonies:
        s_c        = multi_colony_update(global_solutions)
        s_c_update = selection(sc)
        update_pheromones(s_c_update, n_update)
```

Listing 4.4: Generic MOACO algorithm

Each ant colony constructs probabilistically a solution biased by pheromone and heuristic information. To store the pheromone and respectively heuristic information one or multiple matrices can be used. In the case of multiple matrices those matrices need to be aggregated. This is done by the `aggregation` function, which is incorporating the use of a weight. The sequence of weights that is applied is determined by the `next_weight` function which takes the current ant as well as the current iteration into account.

Based on the aggregated pheromone and heuristic information a solution can be constructed probabilistically by the `probabilistic_solution_construction` function. Optionally, a local search (`weighted_local_search`) can be applied to improve the solution. After that the constructed solution can be added to the set of global solutions.

When all ant colonies have finished constructing solutions, the solutions can be exchanged between the colonies by the `multi_colony_update` function. From all solutions the `selection` function chooses the solution that will be used for updating the pheromone information (`update_pheromones`).

The MOACO algorithm continues until the termination condition is met, e.g. a certain time span exceeds or a certain number of iterations is reached.

Basic Idea of the Multi-objective Ant Colony Optimisation Routing Framework

Based on the presented generic MOACO algorithm MARFWSN is developed, a framework for MOACO-based WSN routing algorithms. It provides the basic packet forwarding functionality using ants as well as the management of the corresponding data structures for heuristic values and pheromones. Additionally, MARFWSN provides a flexible interface for MOACO algorithms so that specific functionality, such as the deposition of pheromone or the movement of ants by some sort of probabilistic rule, can be simply exchanged by selecting another MOACO algorithm. As a result, a very flexible framework is created that enables the testing of different MOACO algorithms in the context of WSN routing by simply docking them to the MOACO-based network layer.

The MARFWSN pursues a *hybrid approach*, i.e. on the one hand, a *reactive approach* is used to find new routes on demand; and, on the other hand, a proactive approach is used for the maintenance and improvement of existing routes. For this reason, two different types of ants are used: *forward ants* and *backward ants*. While forward ants are used in the exploration phase, in terms of route discovery and route maintenance phase, backward ants are used to exploit routes previously found by the forward ants. Backward ants can be either used to notify the node that was requesting a route about a route found by the forward ants, or backward ants can be used to transport an application packet, i.e. data payload, from a source to a destination node.

Implementation of Multi-objective Ant Colony Optimisation Algorithms

Basically, all MOACO algorithms are implemented via `IMOACO` interface that is provided by `MOACONetwLayer`, the network layer of MARFWSN. The provided interface allows to exchange the MOACO algorithm of a sensor node simply by implementing a corresponding MOACO algorithm and assigning this algorithm in the `omnetpp.ini` configuration file to the node.

To avoid the need for a complete reimplementation of the basic MOACO functionality for each MOACO algorithm, a `GenericMOACO` class is implemented that provides this fundamental MOACO functionality. This includes the management of possible routes with corresponding heuristic and pheromone matrices as well as the updating of the pheromone matrices and the management of solutions. The `GenericMOACO` class is quite flexible, i.e. it can deal with multiple objectives, multiple ant colonies as well as multiple pheromone and heuristic matrices.

The `GenericMOACO` class provides the virtual functions `update_pheromone`, `local_update_pheromone`, `get_bwd_ants` and `get_next_hop` that need to be overridden by every derived MOACO algorithm class. Although providing the basic functionality, the `GenericMOACO` class is not a MOACO algorithm itself, it can be rather seen as some kind of abstract class[5]. The basic functionality of those virtual functions is shown in table 4.3.

Virtual Function	Description
`get_bwd_ants`	Returns one or multiple backward ants, i.e. the solutions found by the current MOACO algorithm, which should be sent back to the original source node that started the route request. Depending on the algorithm this can be the best-so-far ant, the iteration best ant, all ants or any other subset of ants that reached the destination.
`get_next_hop`	Returns the next hop to get to the final destination node, based on the probabilistic choice of the MOACO algorithm taking the current pheromone and heuristics matrices into account.
`update_pheromone`	Updates the pheromone for a certain objective of a node, based on the information stored in the backward ant.
`local_update_pheromone`	Local updates of the pheromone for a certain objective of a node based on the information stored in a forward and/or backward ant.

Table 4.3: Virtual functions of the `GenericMOACO` class

Moreover, several parameters can be configured to set up an MOACO algorithm. An overview of all parameters for `GenericMOACO` is shown in table 4.4.

[5]note: strictly `GenericMOACO` is not an abstract class as defined by the C++ `abstract` keyword; however, this results from the fact that `GenericMOACO` is an OMNeT++ class instead of a pure C++ class, which needs to be able to fit into the basic OMNeT++ module structure.

Parameter	Description
`antColonyCount`	Number of ant colonies
`antsPerColony`	Number of ants per colony
`objectiveCount`	Number of objectives considered by the MOACO
`aggregationType`	Type of aggregation used for combining pheromone and heuristic matrices. This can be a weighted sum (`weighted_sum`), a weighted product (`weighted_product`) or a random choice (`random`)
`heuristicType`	Type of heuristic that should be used, i.e. one per objective (`one_heuristic_per_objective`) or one for all objectives (`one_heuristic_for_all_objectives`)
`pheromoneHeuristicWeightsType`	Type of weights that should be used in the aggregation of pheromone and heuristic matrices. This can be either one weight per iteration (`one_weight_per_iteration`) or all weights per iteration (`all_weights_per_iteration`)
`pheromoneHeuristicWeights`	List of weights separated by space that are used for aggregation of pheromone and heuristic matrices
`decisionMakerType`	Type of decision maker (currently only `weighted_product_metric` is available)
`decisionMakerWeights`	Weights separated by space that are used for the decision maker
`pheromoneEvaporationRate`	Rate in which the pheromone will evaporate
`initialPheromoneValue`	Pheromone value that is used for the initialisation of the pheromone matrices
`initialHeuristicValue`	Heuristic value that is used for the initialisation of the heuristic matrices
`alpha`	Specifies the relevance of the pheromone matrices in the MOACO algorithm.
`beta`	Specifies the relevance of the heuristic matrices in the MOACO algorithm.
`localPheromoneUpdate`	If flag is set to `true`, the local pheromone update is activated
`onlyParetoOptimalSolutions`	If flag is set to `true`, only Pareto optimal solutions are considered as valid solutions of the MOACO

Table 4.4: Parameters for the `GenericMOACO` class

From the `GenericMOACO` class the following specific MOACO algorithms were derived (see figure 4.7): `ASMOACO`, `MMASMOACO`, `ACSMOACO` and `SRMOACO`. The main differences between those implemented MOACO algorithms are discussed briefly in the following:

- **Ant System Multiple-objective Ant Colony Optimisation**: The `ASMOACO` implements the basic idea of the AS [122–125] while considering multiple objectives. After an iteration is completed, every forward ant is converted to a backward ant and then sent back to the source. On the way back, each backward ant updates the pheromone value at each node it is visiting. No local updating is performed in this MOACO algorithm.

- **Min-Max Ant System Multiple-objective Ant Colony Optimisation**: The `MMASMOACO` implements basically the MMAS [114–116] approach for multiple objectives. Globally only the best route found is emphasised by depositing pheromone. No local updating is performed in this MOACO algorithm. Furthermore, the pheromone value is restricted to a certain range by the parameters `minimumPheromoneValue` and the `maximumPheromoneValue`. At the beginning the initial pheromone value is set to the maximal possible value from this range (`maximumPheromoneValue`). By setting the `antType` parameter either the best-so-far ant or the iteration-best ant can be used to update the pheromone.

- **Ant Colony System Multiple-objective Ant Colony Optimisation**: The `ACSMOACO` implements the idea of the ACS [117–119] approach, a modification of AS regarding three points: a better exploitation of search experience, pheromone deposition and evaporation on best-tour and a local update rule that removes pheromone each time an ant moves from the current node to the next node. Consequently, at the end of an iteration, only the ant that found the best route is allowed to deposit pheromone on the way back to the source. Additionally, a local pheromone update is triggered each time an ant is sent from the current to the next hop[6]. This leads

[6]While in the original proposal the pheromone is updated on the arrival of the ant at the next node, in the current implementation the local pheromone update is triggered in advance at the current node, i.e. before ant is sent to the next node. This

to a faster evaporation of the pheromone at visited nodes resulting in also using unvisited nodes (stronger exploration). Besides, the parameter `q0` specifies the ratio between exploration of new routes and exploitation of existing routes.

- **Simple Random Multiple-objective Ant Colony Optimisation** `SRMOACO`: provides a very simple random ant-based algorithm that was created just as reference point to compare it to the other MOACO-based algorithms. Basically, the concept of sending forward ants to the source and receiving backward ants with the requested information is kept, while the depositing of pheromone is completely omitted. As a consequence, the forward ants always select the next hop randomly. Neither global nor local updating is performed in this approach.

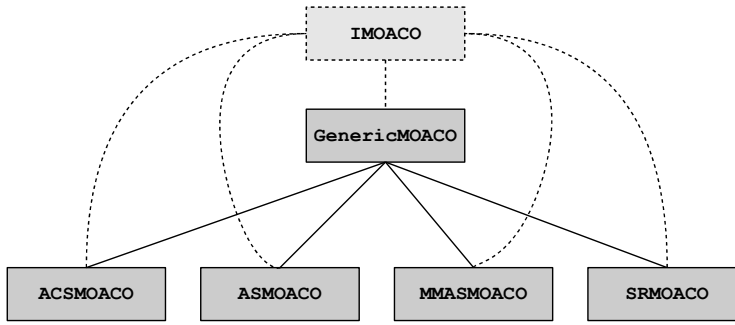


Figure 4.7: Derived classes from `GenericMOACO` all implementing the `IMOACO` interface

Ant Messages

MARFWSN is based on the exchange of ants, represented by `Ant` packets. `Ant` packets are derived from the `NetwPkt` class (listing 4.5) provided by the MiXiM framework.

is because the pheromone should influence future decisions at this node, instead of influencing the decision at the next hop.

```
packet NetwPkt
{
    int destAddr;  //destination address
    int srcAddr;   //source address
    int ttl = 1;   //time to live field (IP)
    unsigned long seqNum = 0;  //sequence number
}
```

Listing 4.5: `NetwPkt` packet

The `NetwPkt` class provides basic fields such as the `destAddr` and `srcAddr` for the destination and source address in terms of network address. Moreover, a `ttl` field for the 'time to live' information as well as a field `seqNum` for the sequence number is provided[7].

Forward as well as backward ants are represented by the `Ant` class (listing 4.6). The `Ant` class adds additionally an original source address (`originSrcAddr`) and a final destination address (`finalDestAddr`) to the message. This is required to distinguish the network addresses used for the next hop (`srcAddr` and `destAddr`) and the overall source and destination addresses (`originSrcAddr` and `finalDestAddr`) of a multiple hops route.

Additionally, each `Ant` is identified by the colony it belongs to (`antColonyNo`), the number of the ant in this colony (`antNo`) and the iteration the ant belongs to (`iterationNo`). Due to the fact that ants are converted from forward to backward ants, the backward ants need an additional identification number, the `bwdAntNo` field. The `antType` specifies whether the ant is a forward ant (`FORWARD_ANT`) or a backward ant (`BACKWARD_ANT`).

The rank of the ant (`antRank`) is used in the pheromone update process because certain MOACO algorithms give certain updating privileges to certain ants. The rank of an ant can characterise the ant as 'normal' ant

[7] In the current implementation the `seqNum` field as well as the `ttl` field are not used. Consequently, both could be omitted to save some bandwith; however, to have a consistent MiXiM-based design the `Ant` packet is nevertheless derived from the `NetwPkt` packet.

```
enum ANT_TYPE {
    FORWARD_ANT  = 1;
    BACKWARD_ANT = 2;
}

enum ANT_RANK {
    DEFAULT_ANT          = -3;
    ITERATION_BEST_ANT   = -2;
    BEST_SO_FAR_ANT      = -1;
}

packet Ant extends NetwPkt {
    int originSrcAddr;  //origin network address
    int finalDestAddr;  //final destination network address
    int antColonyNo;    //ant colony number
    int antNo;          //ant number in the ant colony
    int iterationNo;    //number of the iteration
    int bwdAntNo;       //bwd ant number
    int antType enum(ANT_TYPE);  //type of the ant, either
        FORWARD_ANT
                                 //or BACKWARD_ANT
    int antRank enum(ANT_RANK);  //rank of the ant, either
                                 //DEFAULT_ANT,
                                     ITERATION_BEST_ANT,
                                 //BEST_ANT_SO_FAR or rank >=
                                     0
    int visitedNodes[];  //list of nodes that were visited by
        the ant
    double objectives[]; //list of objectives
}
```

Listing 4.6: `Ant` packet

(DEFAULT_ANT), best ant in the iteration (ITERATION_BEST_ANT) or best ant over all iterations so far (BEST_SO_FAR_ANT). In addition to the predefined ranks, other ranks can be introduced by assigning a positive value to the antRank field, if required by the MOACO algorithm.

The visitedNodes[] field is used to store all nodes that have been visited by an ant. While in the discovery phase forward ants add each node they visit to that list, backward ants just use this list to return to the original source. Besides, the visitedNodes[] list is used to avoid loops[8]. Due to the fact that multi-objective optimisation problems are considered, ants are capable of transporting information about multiple objectives, represented by double values, in the objectives[] list.

Finding the Best Solutions within an Iteration

Apart from the first ACO algorithm, AS, in which all ants are contributing to the process of depositing pheromone, the later ACO algorithms rely on emphasising only certain (best) paths. Depending on the chosen ACO algorithm, only the best-so-far ant, the iteration-best ant or the k-best-ranked ants are allowed to place pheromone along the way.

While finding out the best-so-far ant is a minor problem, finding the iteration-best or the k-best ranked ants is a major problem in a WSN scenario due to its dynamic nature. In contrast to many other applications of ACO algorithms, the considered WSN scenario is not turn-based, but rather time-dependent. Moreover, the use of the wireless channel as transmission medium leads to additional problems such as packet loss resulting in 'dying ants' that never arrive at their final destination. For that reason, it has to be specified for the WSN scenario when an iteration starts and when it is completed. The problem of finding the beginning and the end of an iteration in a non-round-based network is tackled as follows:

The start of a route discovery is triggered with the sending of the first ant of the iteration at the original source. However, due to the fact that

[8] In the current implementation forward ants that are visiting a node that is already in the visitedNodes[] list, are simply dying, i.e. the ant message is dropped by that node. An alternative would be that the ant forgets the loop and continues with the exploration – this is however not pursued here to keep the end-to-end delay low.

finding solutions is the main goal of the ACO algorithms, the first arriving ant at the final destination marks the start of the iteration.

In the next step, the end of the iteration has to be found, which is the most difficult thing in this context. To find out whether an iteration has ended at the final destination there are three options:

1. All ants of the current iteration reach the final destination
2. An iteration timer expires
3. An ant of the following iteration arrives at the final destination

The first case, in which all ants that were sent out by the original source are reaching the final destination, is the ideal and trivial case[9]. However, due to the already discussed issues that are leading to packet losses, this case is rather unlikely. To address the issue of ants that are never arriving at the final destination, an iteration timer is introduced. This timer starts with the arrival of the first ant at the destination for the current iteration and lasts for the given time span. If the timer expires before all ants of the current iteration have reached the destination, the iteration will be completed anyway. The third case occurs when an ant of the following iteration reaches the final destination before the previous iteration has been completed. In this case, the first ant of the following iteration that is arriving at the destination node completes the previous iteration and subsequently, triggers the beginning of the next iteration.

Objectives and Ant Colonies

In the implementation of MARFWSN there are two possibilities considered regarding the number of objectives and the number of corresponding ant colonies (see table 4.5). As a result, either one ant colony is managing all objectives or for each objective a separate ant colony can be used.

Although, the MARFWSN is quite flexible regarding the number of objectives and the number of ant colonies, for the implementation one ant colony is used to update all objectives. This is justified in the fact that

[9] To find out whether the last ant has arrived the final destination, a counter checks the incoming ants of the iteration and compares the counter to the total number of ants in the colony.

Objectives	Ant Colonies	Comment
n	1	All objectives are updated by a single ant colony.
n	n	Each objective is updated by an independent ant colony

Table 4.5: Number of objectives and number of ant colonies (n number of objectives)

the overhead should be minimised that is required for storing the matrices as well as the overhead caused by the ants that need to be sent for each colony.

The following three objectives are considered in the implementation:

1. Residual energy
2. Duration of transmission
3. Trust value

The *residual energy* (abbr. *energy*) is the current energy of a sensor node that is remaining before it dies. Due to the fact that the routing problem is considered as minimisation problem the reciprocal of the residual energy is used. The residual energy values are combined by multiplication at each node.

The *duration of transmission* (abbr. *duration*) is calculated at each node as the current time minus the timestamp from the packet, i.e. the time it was sent.

The *trust value* (abbr. *trust*) is the value that was assigned to the node[10]. Again the reciprocal of the trust value is considered as part of the minimisation optimisation problem. The trust values are combined by multiplication at each node.

[10]In the current work there is no exchange of trust information so that the trust value is assigned by the maliciousness module at the start of the simulation. More sophisticated TRSs are thinkable to be used in this context, but due to the complexity the management of trust is neglected here.

Flowcharts of the Behaviour of Forward and Backward Ants

As described before, if a route to a certain destination node is unknown, forward ants are sent out to discover a route to that destination. The basic approach is depicted in the flowchart in figure 4.8.

On the way back, depending on the chosen MOACO-based routing algorithm, one or multiple backwards ants are sent back to the original source node. The general approach is depicted in the flowchart in figure 4.9.

Data Structures

The `GenericMOACO` class uses the `MOACONodeManager` to manage the neighbours of the node with its corresponding heuristic and pheromone structures. For each destination node a `MOACONodeMapEntry` is provided that contains the corresponding pheromone and heuristic values. For the aggregation of pheromone/heuristic matrices an aggregation function is provided that considers the current ant, the current iteration as well as the assigned weights. Furthermore, the solutions found so far are managed by `MOACONodeManager` using the `SolutionManager` class.

Figure 4.10 depicts an overview of the main data structures that are used by the MOACO-based algorithms in an example. Basically, a forward ant that should be transmitted by the forwarding *node 1* checks the destination of the packet and selects the corresponding destination node e.g. *node 10*. Based on the chosen destination node, the possible neighbours (*node 2*, *node 3* and *node 4*) are stored as possible next hops. For each neighbour there is a pheromone and a heuristic vector that contains the corresponding values for each objective. Depending on the chosen MOACO algorithm a probabilistic choice is made based on the given pheromone and heuristic values for each neighbouring node. The ant is then forwarded to the chosen next hop neighbour.

Know your neighbourhood

When the sensor nodes are deployed in a random fashion, the nodes will not know the other nodes in their neighbourhood. Consequently, the nodes are not capable of sending unicast messages to certain nodes in the neighbourhood, as required for the MOACO-based routing algorithms.

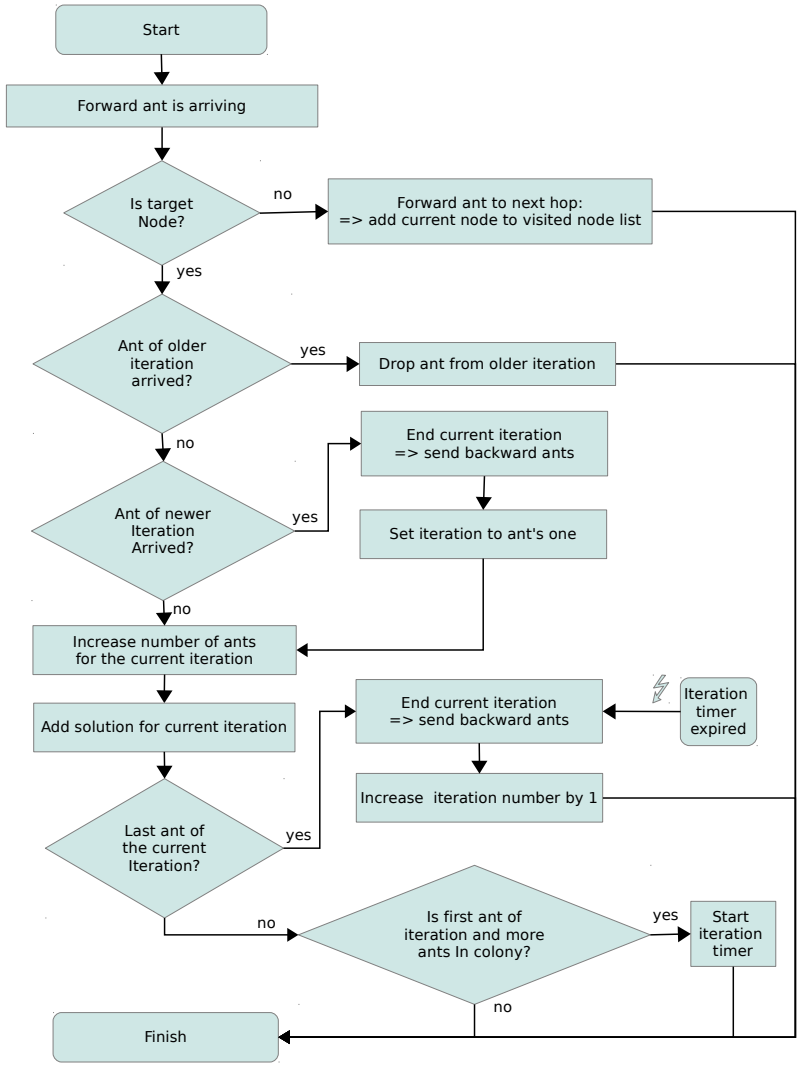


Figure 4.8: Flowchart of the route discovery with forward ants

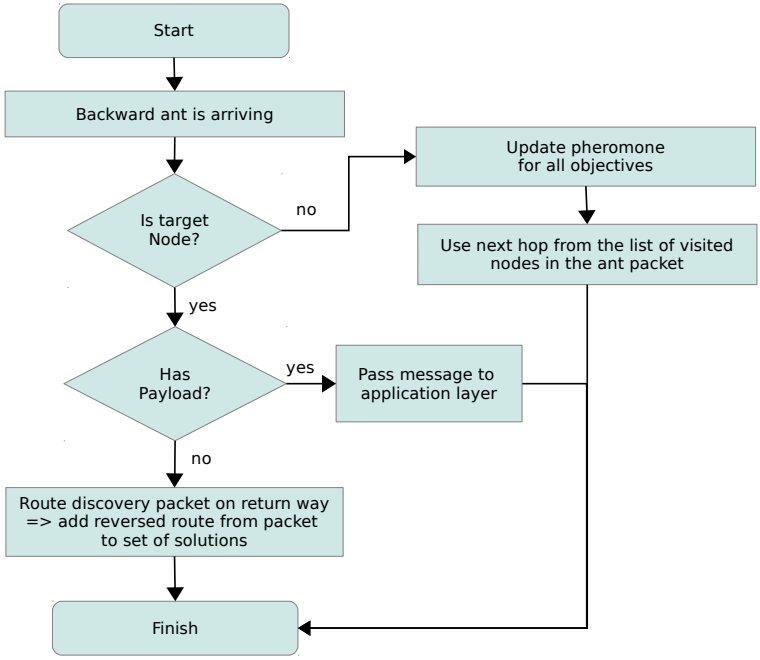


Figure 4.9: Flowchart of transporting packets with backward ants

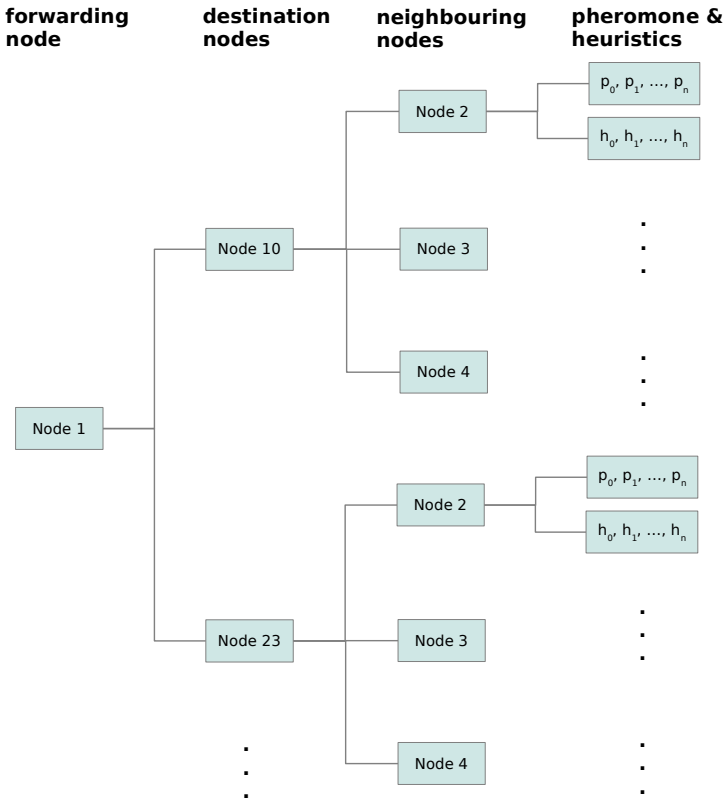


Figure 4.10: Overview of the main data structure for MOACO-based algorithms in an example

To solve that problem, a simple broadcast mechanism is introduced to let nodes announce their presence in the neighbourhood: in defined intervals a `HELLO` message is broadcasted by each node so that nodes in the corresponding neighbourhood are aware of the other nodes' presences. When a node receives a `HELLO` message, the address of the sending node is stored in the table of neighbours for later use, i.e. as prospective next-hop destination.

In case that a node disappears, e.g. due to running out of energy or other interferences, the node needs to be removed from the neighbours list of the nodes in the neighbourhood. This is achieved by a timer which checks in certain time intervals for each known neighbour whether a `HELLO` message was received within the specified time interval, if not the node is deleted from the list of neighbours. Of course, if a node reappears it will be added again to the list of neighbours.

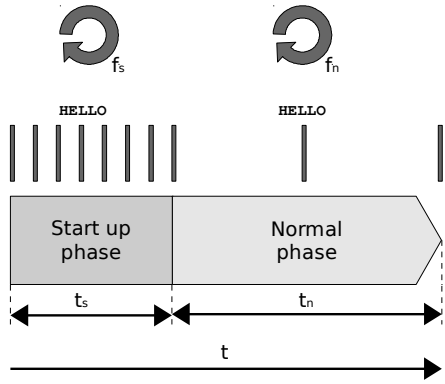


Figure 4.11: Two phases of HELLO messages

To reduce the data traffic of `HELLO` messages a two phase strategy is chosen (see figure 4.11). The entire simulation time t is divided into two phases: the *start up phase* and the *normal phase*. Due to the fact that after the deployment of the WSN the nodes' neighbourhoods are unclear, the frequency of `HELLO` messages f_s is at the beginning rather high. After completing the start up phase, i.e. after the duration of t_s, the normal phase is started for the duration of t_n, i.e. to the end of the simulation time. In the normal phase `HELLO` messages are send with a lower frequency t_n because the neighbourhood of each node should be clear in general so

that each node just needs to send an 'I am alive' signal from time to time. This approach results in a lower data traffic caused by `HELLO` messages.

Simulation Parameter	Description
`helloTimerStartUpDuration`	Duration of the start up phase.
`helloTimerStartUpFrequency`	Frequency of `HELLO` messages in the start up phase.
`helloTimerFrequency`	Frequency of `HELLO` messages in the normal phase.
`helloTidyUpTimerFrequency`	Frequency of the tidy up mechanism that removes neighbours that did not sent a `HELLO` within this time.

Table 4.6: Simulation parameter for `HELLO` messages used in the simulation

An overview of the parameters that are affecting the broadcasting of `HELLO` messages is shown in table 4.6.

Application Message Queue

In general, a network layer receives packets from the application layer, which then should be routed through the network from a source node to a destination node. However, a problem arises when the application layer is sending packets, which cannot be routed to the destination because the route to the destination node has not been found yet. To solve this problem, an *application packet message queue* is introduced that deals with the buffering of application packets that cannot be routed immediately.

When an application packet arrives from the application layer at the network layer, the application packet is added to the corresponding application packet queue for the destination node (see figure 4.12). If the maximum queue length is exceeded (`maxQueueLength`), the oldest packet will be removed from the queue and the newly arrived packet will be added. For each destination node an application packet queue timer is started. When this timer is triggered the application packet at the front of the queue is tried to be delivered to the destination. If there is still no route available the timer will be started again with an exponential back-off mechanism. If the maximum number of retries is exceeded, the application packet at the front of the queue will be removed. When there is still a packet in the application packet queue, the timer is restarted again for the packet that recently became the first packet in the queue.

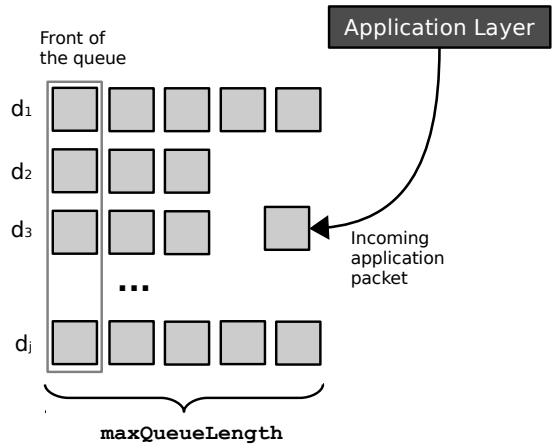


Figure 4.12: Basic idea of the application message queue

Table 4.7 shows the parameters that are influencing the behaviour of the application packet queue.

Simulation Parameter	Description
`appPktMaxQueueLength`	Maximum Number of application packets in the queue per destination node.
`appPktMaxRetryAttempts`	Maximum number of retry attempts for transmitting an application packet from the queue.
`appPktExponentialBackoffSlot`	Time slot that is used for the exponential-backoff mechanism

Table 4.7: Simulation parameter for the application packet queue used in the simulation

4.3 Network Layer: Dynamic Source Routing

For the comparison of MARFWSN to an existing routing protocol for WSNs, additionally an DSR-based protocol is implemented. The implemented protocol is inspired by the original DSR protocol [185], but removes the requirement of Internet Protocol (IP) so that it can run directly as network layer, making it directly comparable to MARFWSN using the same application protocol on top. At the current state the

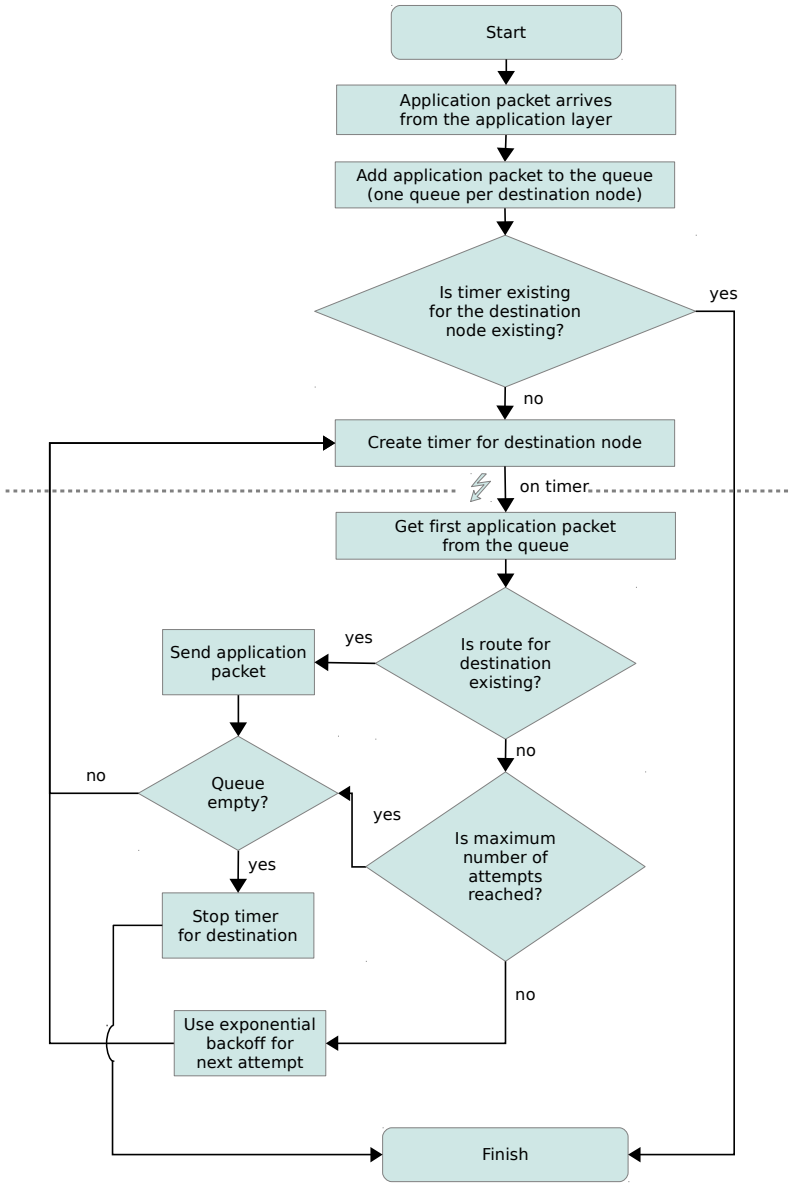


Figure 4.13: Flowchart of the application message queue

DSR-based protocol has just the basic functionality without special features such as salvaging etc. In the following the basic DSR functionality will be explained.

Dynamic Source Routing Messages

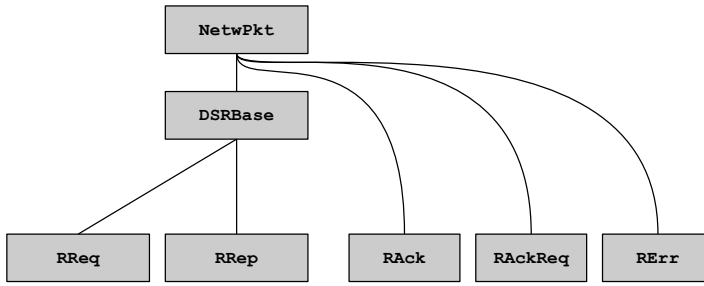


Figure 4.14: Derived DSR message classes

Figure 4.14 depicts all messages that are used in the DSR-based routing protocol. As in the MARFWSN network layer, all messages are derived from the `NetwPkt` class (see listing 4.5). For the two main message classes `RReq` and `RRep` the common `DSRBase` class was introduced as parent message class to group some of the common fields that are required in both message classes. In contrast, the `RAck`, the `RAckReq` and the `RErr` messages that are used for the route maintenance are directly derived from the `NetwPkt` class.

The `DSRBase` class contains some important fields that are required in the `RReq` as well as the `RRep` messages. This is, on the one hand, the `initialSrcAddr` that specifies the origin of the message and `finalDestAddr`, the address of the final destination. The `routeRecord[]` array stores all nodes that have been visited by the packet. The `trust` field is neither part of the original DSR nor it is used in any way for the DSR-based protocol, but it is rather used to make the trust values comparable to the trust values of MARFWSN.

The `RReq` message, derived from the `DSRBase` message class, adds only the additional field of `requestId` that is used to make `RReq` messages clearly

```
packet DSRBase extends NetwPkt{
    int initialSrcAddr; //initial source that started the
        request
    int finalDestAddr;  //final destination address
    int routeRecord[];  //list of nodes that were visited
    float trust;        //trust value for statistics [NOT part
         of DSR]
}
```

Listing 4.7: `DSRBase` packet

```
packet RReq extends DSRBase {
    int requestId; //unique request id
}
```

Listing 4.8: `DSRReq` packet

identifiable in the broadcasting process.

The `RRep` message is also derived from the `DSRBase` message class; however, no additional fields are added so that it provides exactly the same fields as the `DSRBase` message class. Nevertheless, this message class was added to make this packet type better distinguishable among the other message classes.

The `RAck`, `RAckReq` and the `RErr` message classes are both used for the route maintenance.

```
packet RRep extends DSRBase {
}
```

Listing 4.9: `DSRRep` packet

```
packet RAckReq extends NetwPkt {
    int ackId; //acknowledgement id
}
```

Listing 4.10: `RAck` packet

```
packet RAck extends NetwPkt {
    int ackId; //acknowledgement id
}
```

Listing 4.11: `RAck` packet

The `RAckReq` message just has an additional `ackId` field so that the acknowledgement messages can be clearly distinguished.

The `RAck` message usess the same `ackId` field that is corresponding to the same field in the `RAckReq` message.

The `RErr` message provides an aditional `errorType` field in which one of the defined error types can be used. In the case of a `NODE_UNREACHABLE` error, the `unreachableNodeAddr` can be used to specify the unreachable node.

Route Discovery and Route Maintenance

The DSR-based routing protocol can be divided into two phases: the route discovery phase and the route maintenance phase. In the route discovery phase, new routes are discovered for a given destination to which no routes exist in the routing table yet. In the route maintenance phase, existing routes are checked for availability. In the case of unavailable routes links can be marked as 'broken' and all routes that are containing the broken link are removed. Both phases are explained in more detail in the following:

```
enum ErrorTypes {
     NODE_UNREACHABLE = 1;
     FLOW_STATE_NOT_SUPPORTED = 2;
     OPTION_NOT_SUPPORTED = 3;
}

packet RErr extends NetwPkt {
    int errorType enum(ErrorTypes); //type of error
    int unreachableNodeAddr; //used in the case of
        NODE_UNREACHABLE
}
```

Listing 4.12: `RErr` packet

Route Discovery: If an application packet should be send from a source to a destination node, but until now no existing route is known, the route discovery process is triggered. This is done by the source node by simply broadcasting a `RREQ` message for the destination. A node receiving the `RREQ` message has several actions to take, as depicted in figure 4.15.

In the first step, the receiving node updates its routing table by routes and partial routes that are stored in the record list of the `RREQ` message. Based on the unique ID of the `RREQ` message, the receiving node checks if the packet was seen recently – in this case the `RREQ` message is simply discarded. Also in the case that the `RREQ` message has travelled already to many hops, the `RREQ` message is discarded. In the other case, the node checks if the destination in the `RREQ` message is the node itself or if the node knows a route to the destination. If this is the case, a `RREP` message is send back on the way the `RREQ` message came. Otherwise, the node appends its address to the route stored in the record list of the `RREQ` message and then broadcasts the packet again to the nodes in its neighbourhood.

When a node receives a `RREP` message, the node has to check several options, as depicted in figure 4.16:

In the first step, the routing table of the node is updated by the route

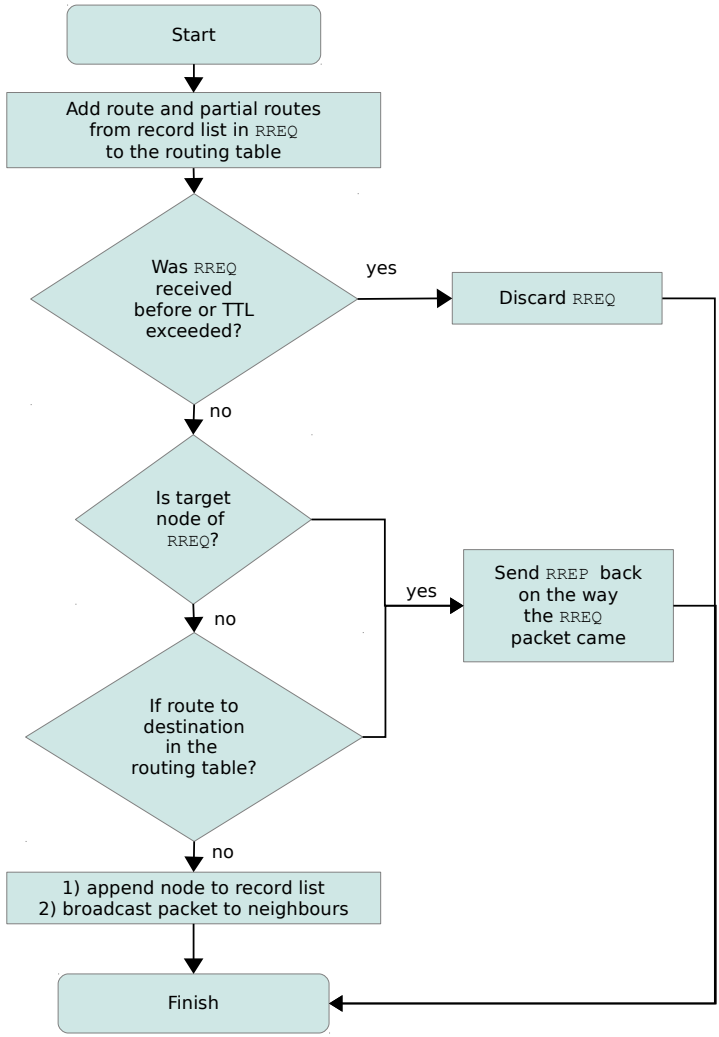


Figure 4.15: DSR route discovery: on receiving `RREQ`

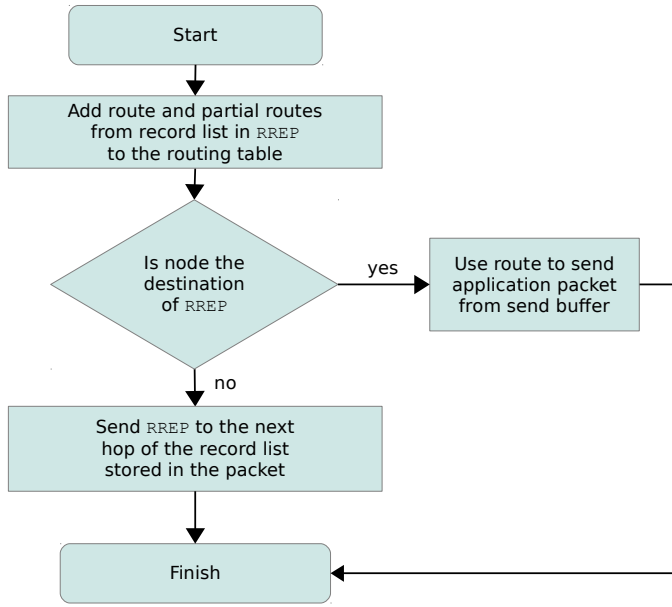


Figure 4.16: DSR route discovery: on receiving RREP

and partial routes that can be derived from the record list in the `RREP` message. Subsequently, the node checks if it is the destination node – in this case, the found route can be used to transfer the application packet from the send buffer. If the node is not the destination node, the node sends the `RREP` message to the next hop of the route from the record list that is stored in the `RREP` message.

Figure 4.17 shows an example of the route discovery process in the DSR-based protocol.

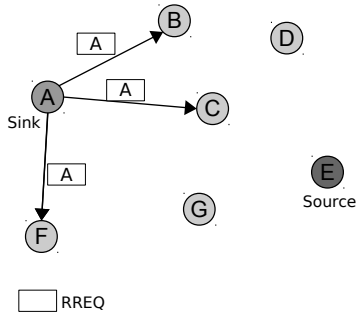
Route Maintenance: Additionally, a route maintenance mode can be activated in the DSR-based routing algorithm that is capable of detecting broken links. The soft acknowledge mechanism provides the `RAckReq` message to ask a certain node for an acknowledgement. If the requested node responses with an `RAck` message everything is impeccable, i.e. the link is available. For a certain time span the requested node will not be asked again.

However, if the requested node is not responding, the requesting node uses an exponential back-off mechanism to repeat the sending of the `RAckReq` message again. If the number of maximum request attempts is exceeded and the node does not respond with a `RAck` message, the link will be marked as broken. Subsequently, the routing table of the node will be checked for routes that contain the broken link. All routes that include the broken link will be removed from the routing table.

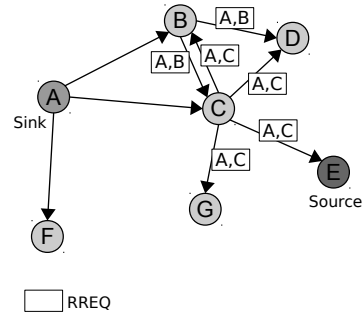
The route maintenance mode in DSR is optional because the underlying layers can often provide a mechanism for the detection of broken links.

4.4 Summary

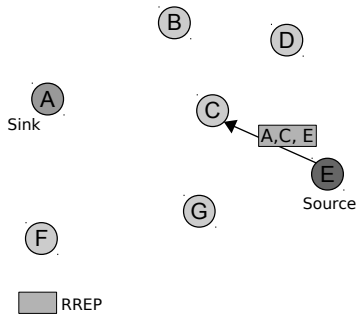
In this chapter, the implementation of MARFWSN and its related protocols were discussed. Taking up the research question, the basic problem definition was transferred to the implementation task developed for this thesis. After that, the used methodology was presented including the justification of the chosen simulation environment OMNeT++ and the MiXiM framework for the support of WSN scenarios. Subsequently, the



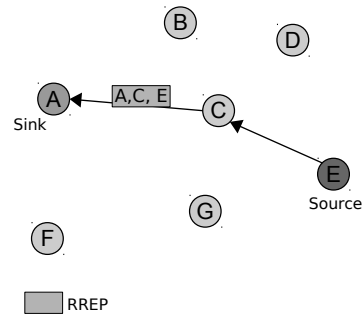
(a) Sink node A want to send an application packet to source node E. Due to the fact that there is no route know between those node, it sends a `RREQ` message to discover a route to source node E



(b) `RREQ` messages are broadcasted through the network. Each node adds its address to the list of recorded nodes in the `RREQ` message.



(c) After a while some routes between node A and node E are discovered. The shortest route is chosen and a `RREP` message is sent back towards the sink



(d) `RREP` message is sent back along the path stored in the recorded list of visited nodes until it reaches the destination. The found route can then be used to send an application packet from node A to node E

Figure 4.17: Basic idea of the DSR routing algorithm

developed application layer was presented including the underlying protocol and the traffic generator as sample application. Furthermore, two different network layers were discussed, on the one hand, MARFWSN, a routing framework that supports multiple MOACO-based routing algorithms; and, on the other hand, DSR, an adaption of the famous DSR protocol that can be used in WSN. While MARFWSN targets directly the research question by supporting the MOACO-based routing algorithms `ASMOACO`, `MMASMOACO`, `ACSMOACO` and `SRMOACO`; the latter `DSR` was implemented as well-known routing algorithm for comparison purposes.

In the following chapters, the presented implementation will be used in several simulation scenarios and then, the outcomes will be analysed and evaluated.

5

Experiments and Evaluation

> A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it.
>
> Albert Einstein (1879 - 1955)

In this chapter, the implemented WSN routing approaches will be examined and compared in various simulation experiments. As described in the previous chapter 4, the routing algorithms were implemented for the network simulator OMNeT++ [260] using the MiXiM framework [261]. In several experiments, the performance of the implemented routing algorithms will be tested for different simulation scenarios with configurations. For this reason, in the first step, the general experiment settings will be described, including the used scenario configuration and sensor node configuration. Subsequently, the configuration of the routing algorithms, i.e. the MARFWSN-based routing algorithms as well as the DSR-based routing algorithm, will be discussed. Moreover, the performance metrics that will be used to compare the different routing algorithms will be explained. In the next step, the different experiments will be presented: this includes some pre-experiments, which deal with finding the best parameters for the MOACO-based algorithms for the different scenarios, and subsequently, the comparison of the MARFWSN-based routing algorithms to the DSR-based routing algorithm, applying the best parameters of the pre-experiments found. The simulation results will be evaluated and conclusions will be drawn.

5.1 Experiment Settings

In this section, the general settings of the experiments are discussed, which are common to all tested simulation scenarios. This includes the configuration of the simulation scenario in terms of used parameters and

the configuration of the sensor nodes.

5.1 Simulation Scenario Configuration

In the following, the configuration of the simulation scenarios is discussed, i.e. the basic parameters that apply to all conducted experiments such as the playground, the network topologies, the data traffic pattern etc.

The Playground

For all simulation scenarios a square area, also referred to in OMNeT++ as *playground*, is used. Based on the IRIS Mote specifications [9] and some simulation test-runs, a playground area of 1000 m × 1000 m was defined for 10 nodes, which provides a reasonable connectivity for different random topologies.

Based on this definition the *node density*, i.e. number of nodes per area, can be calculated as follows:

$$node\ density = \frac{10\ nodes}{1000\ m * 1000\ m} = \frac{1}{100000} \tag{5.1}$$

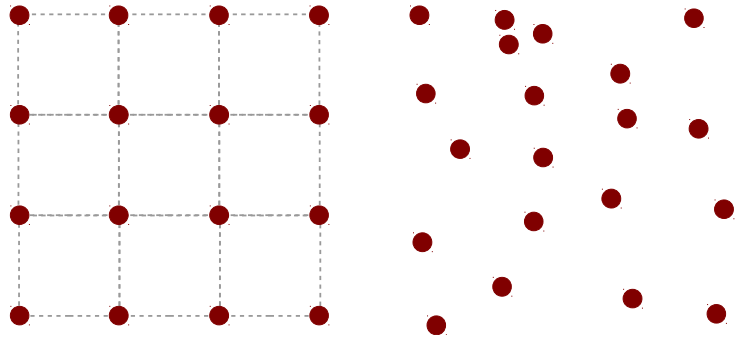
For all simulations with different amount of nodes the node density should be kept the same so that the side length (x) of the playground area can be calculated as follows, depending on the number of nodes:

$$\begin{aligned} node\ density &= \frac{10\ nodes}{1000\ m \times 1000\ m} = \frac{1}{100000} \\ x^2 &= \#nodes \times 100000 \\ x &= \sqrt{\#nodes \times 100000} \end{aligned}$$

Depending on the chosen topology the exact playground sizes are presented later in table 5.1 for the grid topology and in table 5.2 for the random topology.

Network Topologies

For the different experiments two different types of network topology are used: the *grid topology* and the *random topology*.



(a) Grid topology (b) Random topology

Figure 5.1: Examples of a grid and a random topology

For the first set of experiments a *grid topology* is used to obtain a uniform distribution of the sensor nodes in a grid (see figure 5.1a). This limits one of the factors of randomness in the scenarios so that less fluctuations in the results are expected. For each scenario the number of nodes is chosen based on square numbers $(9, 16, 25, ..., 100)$ allowing the alignment of the nodes in a uniform, fully occupied grid in the squared playground area. Consequently, the direct neighbours of each node are always in the same distance, and it is clear that the WSN does not fall into multiple partitions because every node has the fixed amount of surrounding neighbours in the grid.

As highlighted in the previous section, the node density is kept consistent for all experiments. The resulting playground sizes for the corresponding number of nodes in the grid topology are shown in table 5.1.

For the second set of experiments a *random topology* is used, i.e. each sensor node is randomly placed on the playground (see figure 5.1b). As a result, the distances between the nodes' neighbours vary and thus, also the number of neighbours per node due to the limited radio range of each node. Though, this scenario seems to be more realistic, it has the disadvantage that at worst the network gets partitioned and no valid route

#Nodes	Playground Size
9	949 m × 949 m
16	1265 m × 1265 m
25	1581 m × 1581 m
36	1897 m × 1897 m
49	2214 m × 2214 m
64	2530 m × 2530 m
81	2846 m × 2846 m
100	3162 m × 3162 m

Table 5.1: Playground sizes for different number of nodes using a grid topology

can be found between the source and the destination nodes.

For each experiment a different number of nodes is used, starting from 10 up to 60 nodes in steps of five $(10, 15, 20, ..., 60)$. Again a constant node density is used in the experiments. The resulting playground sizes for the different number of nodes are presented in table 5.2.

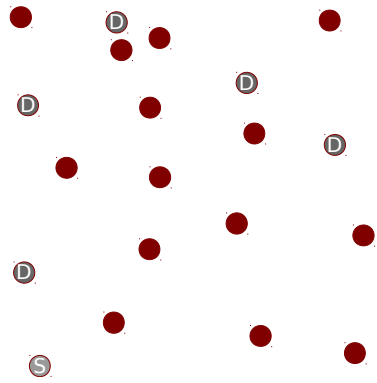
#Nodes	Playground Size
10	1000 m × 1000 m
15	1225 m × 1225 m
20	1414 m × 1414 m
25	1581 m × 1581 m
30	1732 m × 1732 m
35	1871 m × 1871 m
40	2000 m × 2000 m
45	2121 m × 2121 m
50	2236 m × 2236 m
55	2345 m × 2345 m
60	2449 m × 2449 m

Table 5.2: Playground sizes for different number of nodes using a random topology

Data Traffic Patterns

Instead of using a Constant Bit Rate (CBR) traffic, a simple *request-response protocol* is used that imitates a common network traffic pattern in WSNs. The basic flow of the protocol in the tested scenarios is as follows: a source node starts to send out data requests to one of multiple possible destination nodes. When one of the destination nodes receives such a request, it answers with the corresponding data response. For each simulation scenario a traffic pattern with one source and five destinations

is examined (see figure 5.2a).



(a) One source – five destinations

Figure 5.2: Distribution of source and destinations

The node placed first ($node_0$) is defined as source node (S), while the multiple destination nodes ($D_0, ..., D_5$) are chosen randomly, before the scenario is started.[1]

Furthermore, two traffic types are considered in the experiments: Real-time Multimedia Data Traffic (RTM) and Reliable Best-Effort Data Traffic (RBE). While RTM data traffic is simulating video and audio data, in which a fast and continuous transmission of multimedia streams is important; RBE data traffic simulates a reliable transfer of data in which the arrival of data is more important than the time window within which it arrives.

Further Simulation Settings

The duration for each simulation is set to 1200 s (≙20 minutes). Per simulation scenario 25 repetitions for the pre-experiments and 50 repetitions for the experiments were simulated, which were then averaged to obtain a more generalised view on the results. Furthermore, each scenario is tested

[1] It should be remarked that due to the fact that the destination nodes are chosen in a random fashion, in some scenarios the source cannot reach the destination(s). In the case that no traffic is arriving at any destination node during the entire simulation time, the meaningfulness of the results is limited so that in this case these results are omitted. Due to the fact that the scenarios are repeated multiple times and the results are averaged, single omitted results are not crucial for the overall result.

with a different percentage of malicious nodes (0 %, 25 %, 50 %, 75 %)[2]. As discussed before, a grid and a random topology is used in the different scenarios. No mobility is considered so that the sensor nodes stay at the same position for the entire simulation time after their deployment.

Table 5.3 shows an overview of the main parameters that were used for the different scenarios.

Parameter	Value
Simulation Time	1200 s
Number of Repetitions per Scenario	25 (pre-experiments), 50 (experiments)
Number of Malicious Nodes	0 %, 25 %, 50 %, 75 %
Deployment	grid and random
Mobility	static, i.e. no mobility

Table 5.3: Scenario parameters overview table

5.1 Sensor Node Configuration

The sensor node configuration includes the parameters that affect each node individually. As discussed in section 4.2.2, a network node in MiXiM uses a layered approach. Figure 5.3 shows the layers of the network stack as used by the sensor nodes in the simulation scenarios.

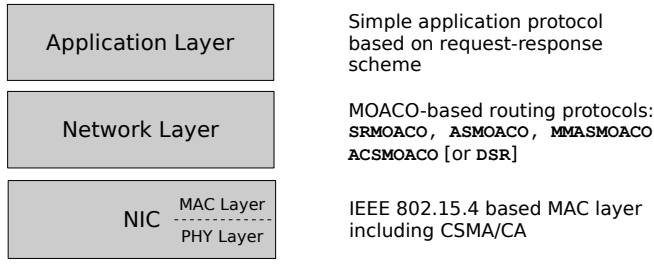


Figure 5.3: Network stack of a sensor node

From a top-down perspective, the main parameters of each layer are discussed briefly in the following:

[2]The term 'malicious node' refers here to a node which is dropping packets with a likelihood of 50 %; correspondingly, the trust value is set to a positive random value below 0.5, marking the node as untrusted.

Application Layer

As stated before, a simple request-response protocol is used on the application layer to imitate a realistic WSN data traffic pattern. For the generation of data traffic a traffic generator is used, as described in section 4.3.1. Table 5.4 shows the basic parameters that were used for the traffic generator in the different simulation scenarios.

Parameter	Value
`initTime`	5 s
`getDataReqGenerationInterval`	10 s
`dataReqInterval`	1 s
`dataReqDuration`	6 s
`randomDstAddressesCount`	5
`trafficType`	`periodic`

Table 5.4: Traffic generator parameters

Depending on the role of the sensor node it must be distinguished between the *sink* that is generating data requests and the *sources* that generate data response messages on request:

The *generation of data requests* is managed by the sink in terms of generating and sending of `GetDataReq` messages to one of the *a priori* selected sources. This is done as follows: after an initial start up time of 5 s, the generation of data request messages will start. From this point in time data requests are sent out periodically in intervals of 10 s to one of the destinations until the simulation stops.

When a data request arrives at one of the sources, the *data response generation* is triggered. From this point in time `DataResp` messages are periodically send out back to the sink. The sending of data response messages is repeated every second and lasts for an overall time of 6 s.

Network Layer

For the routing of the application packets through the network, two different network layers were used in the experiments: a MARFWSN-based network layer (see section 4.3.2) and a DSR-based network layer (see section 4.3.3). While the MARFWSN-based routing algorithms, as main contribution of this thesis, are used for the examination of different MOACO-based routing algorithms; the DSR-based routing algorithm is used solely

for comparison purposes. Due to the fact that the configuration of the network layer is the most important part, it is discussed in-depth in the next section 5.2.

Physical and Media Access Control Layer

In the simulation scenarios the IEEE 802.15.4 standard [270] is used, which specifies the physical and the Media Access Control (MAC) layer for low-rate wireless personal area networks. The MAC layer utilises a Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) (in the non-beacon variant) to avoid collisions. IEEE 802.15.4 is provided by the MiXiM framework for OMNeT++ [261] so that those layer were used without any modification.

The simulation parameters that are related to the hardware are based on the specification of the IRIS Mote [9]. The IRIS mote is a 2.4 GHz Mote module (see figure 5.4) that can be used for the creation of low-power WSNs.

Figure 5.4: IRIS mote [9]

As successor of the famous MICA Motes [272] it provides a three times improved radio range and twice the program memory. An overview of the most important parameters for the simulation scenarios that were derived from the IRIS Mote datasheet is shown in table 5.5.

For all other parameters in this layer the default values were used as provided by the MiXiM framework.

Parameter	Value
Frequency band	2405 MHz to 2480 MHz (ISM band)
Transmit (TX) data rate	250 kbps
RF power	3 dBm
Receive sensitivity	-101 dBm
Current Draw Receive Mode	16 mA
Current Draw Transmit Mode	10 mA
Battery	2x AA batteries (à 1250 mAH and 1.5 V)

Table 5.5: Hardware related parameters based on IRIS mote [9]

5.2 Routing Algorithms

In this section the simulation parameters of the MARFWSN-based and the DSR-based network layer are discussed, which are common to all simulated scenarios.

5.2 MARFWSN-based Network Layer

As discussed in section 4.3.2, the *MARFWSN-based network layer* provides an interface so that different MOACO-based algorithms can be docked to it and hence, easily adapted for their use in WSN routing. For the experiments four different MOACO-based algorithms were implemented: `SRMOACO`, `ASMOACO`, `MMASMOACO` and `ACSMOACO`.

Due to their common origin the MOACO-based algorithms have, on the one hand, some common parameters and, on the other hand, some unique parameters depending on the modifications made to the original algorithms. Both sorts of parameters are discussed in the following:

General Parameters

For all MOACO-based routing algorithm three objectives are considered in the conducted simulations:

1. *Residual energy*
2. *Duration*
3. *Trust*

The *residual energy* is the product of all residual energy values collected by an ant at all visited nodes on the route from the source to a destination node. The *duration* is the overall time an ant needs to get from a source to a destination node. The *trust* value is the product of all trust values collected at the visited nodes on the path from a source to a destination node. For each objective a separate pheromone matrix is used. To reduce the overhead caused by the ants only a single ant colony is used that manages all pheromone matrices at the same time. To combine the pheromone value a *weighted sum* is used.

When the route discovery mechanism is activated every 5 s, new forward ants are sent out to discover new or to improve existing routes. An iteration timer makes sure that if not all ants arrive within 0.8 s at the destination, the iteration is ended automatically. This is necessary because ants can get lost or run into a loop, and thus never arrive at the destination to complete the iteration.

An overview of the general parameters that were used for all tested MOACO-based routing algorithms is shown in table 5.6.

Parameter	Value
Number of objectives	3 (→ Energy, Duration, Trust)
Number of ant colonies	1
Type of aggregation	Weighted sum
`routeMaintenanceTimerFrequency`	5 s
`iterationTimerDuration`	0.8 s

Table 5.6: General parameters for the MOACO algorithms

MOACO Algorithm Specific Parameters

Table 5.7 shows an overview of the main parameters that influence the different MOACO algorithms. Depending on the chosen MOACO algorithm the available parameters slightly differ.

MOACO algorithm	#ants	α	β	ρ	ξ	τ_0	τ_{min}	τ_{max}	q_0
`ASMOACO`	✓	✓	✓	✓		✓			
`MMASMOACO`	✓	✓	✓	✓		✓	✓	✓	
`ACSMOACO`	✓	✓	✓	✓	✓	✓			✓
`SRMOACO`	✓								

Table 5.7: Parameters for MOACO algorithms

with

- #ants is the number of ants per colony
- α relative influence of the pheromone information
- β relative influence of the heuristic information
- ρ evaporation rate used for global pheromone updating
- ξ pheromone decay coefficient used for local pheromone updating
- τ_0 initial pheromone value
- τ_{min} minimal pheromone value

- τ_{max} maximal pheromone value
- q_0 relative importance of exploitation vs. exploration

Due to the fact that SRMOACO is acting completely randomly, i.e. without considering any pheromone or heuristic values, no parameters take influence on this algorithm, except the number of ants. In contrast, the common parameters α, β, ρ and τ_0 are used in ASMOACO, MMASMOACO and ACSMOACO due to their common origin. While MMASMOACO introduces the additional parameters τ_{min} and τ_{max} for limiting the pheromone values by a lower and an upper boundary, ACSMOACO introduces q_0 to specify the relative importance of the exploitation and the local pheromone decay coefficient ξ .

Although, MARFWSN is capable of handling heuristic values and combining them in the probabilistic rule with the pheromone values, in the current implementation the heuristic value is not considered. This results from the fact that it is difficult to find a meaningful heuristic value for a WSN routing scenario with random deployment[3]. Deactivating the heuristic part is achieved by setting α to 1 and β to 0 for all MOACO-based routing algorithms so that only pheromone information is taken into account in the ants' decision process. As a result, it is expected that at the beginning the route discovery phase will take a little bit longer because there is no additional guidance from the heuristic values available. However, in the long run, this should not be a problem.

Due to the assumption that the recommended parameters from other COPs, such as TSP or QAP, cannot be directly transferred to the unique area of WSN routing, several pre-experiments will be conducted to find the optimal parameters for the tested scenarios. For the MOACO algorithms ASMOACO, MMASMOACO and ACSMOACO the available parameters are tested in different combinations to figure out which parameters will deliver the best results. In the end, the best parameters for each MOACO-based algorithm will be chosen based on the findings of the pre-experiments and then compared to each other and the DSR-based routing algorithm.

[3]If the distances between the sensor nodes are known, e.g. by GPS or by manual deployment, the inverse of the distance could be used as heuristic, as often applied in the TSP. However, this sort of assumptions are not considered in the following.

Application Packet Queue Parameters

As mentioned before, an application packet queue is used to buffer application packets that cannot be sent directly to the destination node. An exponential back-off mechanism is used that tries to deliver the same packet for three times. When the application packet cannot be delivered successfully it is removed from the packet queue. Moreover, when there are more than five packets in the queue, the oldest packet will be discarded on the arrival of a new packet.

The application packet queue parameters are shown in table 5.8.

Parameter	Value
appPktMaxQueueLength	5
appPktMaxRetryAttempts	3
appPktExponentialBackoffSlot	1 s

Table 5.8: Application packet queue parameters for MARFWSN

Neighbourhood Discovery Parameters

As mentioned in section 4.3.2, for the MOACO-based network layer a neighbourhood discovery mechanism is required so that each sensor node is aware of the other nodes in its neighbourhood. In the start up phase of the neighbourhood discovery, every 0.5 s a `HELLO_MESSAGE` is broadcasted by each node so that all nodes in the neighbourhood are made aware of its existence. To reduce the data traffic load, after 5 s the start up phases ends and from this point in time only every second a new `HELLO_MESSAGE` is broadcasted. Every 20 s a tidy up timer is called that removes all neighbours which did not sent a `HELLO_MESSAGE` within this timespan.

The configuration parameters for the neighbourhood discovery mechanism are summarised in table 5.9.

Simulation Parameter	Value
helloTimerStartUpDuration	5 s
helloTimerStartUpFrequency	0.5 s
helloTimerFrequency	1 s
helloTidyUpTimerFrequency	20 s

Table 5.9: Simulation parameter for the neighbourhood discovery mechanism

5.2 DSR-based Network Layer

For comparison purposes the *DSR-based network layer* was implemented, as presented in section 4.3.3. There are couple of parameters that are used for the basic configuration of the DSR-based algorithm, which are discussed in the following:

General DSR Parameters

To reduce the broadcasting in the DSR networking layer the maximum number of hops is set to 10. Per destination node 5 routing entries are managed, and as maximum there are 3 messages in the route request table. If no route can be found for a destination, an exponential back-off mechanism is triggered that tries 10 repetitions with a 5 s back-off slot. If still no route could be found the route finding request is omitted. The route maintenance functionality is activated; and, `ack` messages are repeated every second. When no `ack` message arrived within 5 s, the ack request is repeated. When within 3 repetitions no `ack` message arrived the link will be marked as broken.

The general parameters that were used for the DSR-based network layer can be found in table 5.10.

Parameter	Value
`ttl`	10
`routingTableSizePerDst`	5
`routeRequestTableSize`	3
`routeRequestTableMaximumAttempts`	10
`exponentialBackoffSlot`	5 s
`routeMaintenance`	*true*
`ackRequestInterval`	1 s
`ackRequestMaximumUncheckedInterval`	5 s
`ackRequestMaximumAttempts`	3

Table 5.10: General parameters for the DSR algorithm

Application Packet Queue Parameters

As in the MOACO-based routing approach the same application packet queue is used. The queue stores five packets at the maximum before the oldest packet will be dropped and replaced by the newly arriving one. Up to three retransmissions with an exponential back-off mechanism will be

tried to send the application packet again, using an exponential-backoff timeslot of 1 s. If the packet can still not be delivered, the application packet is simply dropped.

The summary of application packet queue parameters is shown in table 5.11.

Parameter	Value
appPktMaxQueueLength	5
appPktMaxRetryAttempts	3
appPktExponentialBackoffSlot	1 s

Table 5.11: Application packet queue parameters for DSR

5.2 Performance Metrics

To compare the routing algorithms with each other and to say something about their performance, it is necessary to agree on a set of network goodness measures, so called *metrics*.

For the experiments the following metrics are considered:

- **Average end-to-end delay**: The average end-to-end delay is the average time (in s) that is needed to deliver an application packet from a source to a destination node. This value should be as small as possible.
- **Average hop count**: The average hop count is the average number of subsequent intermediate nodes (hops) that is used along the path from a source to a destination node. This value should be as small as possible.
- **Routing overhead**: The routing overhead is the ratio of the number of network control packets that are required for a successful delivery of an application packet (in %). This value should be as small as possible.
- **Average residual energy**: The (relative) average residual energy is the average of remaining energy of all nodes in the WSN (in %) at the end of the simulation. This value should be as near as possible to the 100 % mark.

- **Trust**: The trust value is the product of all trust values that were obtained at each hop of the route between a source and a destination node[4]. This value should be as close as possible to 1.
- **Packet Delivery Ratio (PDR)**: The PDR is defined as the ratio of the number of successfully delivered packets to the total number of packets sent from a source to a destination. The PDR should be as close as possible to 1.
- **Distinct Packet Delivery Ratio (DDR)**: The DDR is defined as the ratio of the number of different application response packets that arrive at the original source, in comparison to the application request packets that were sent. This value should be as large as possible.

Packet Sizes and Data Types

For the calculations of the packet sizes, which are used in various metrics, some assumptions have been made regarding the used data types. Those assumptions are summarised in table 5.12.

Data Type	Size
`int`	4 byte
`double`	8 byte
`simtime_t`	8 byte (= 64-bit `integer`)

Table 5.12: Assumptions for calculating the packet sizes

Metrics and Data Traffic Patterns

The variety of different (conflicting) performance metrics makes it hard to define how well a routing algorithm performs. Instead of considering each metric individually, in the following, the metrics are considered in the context of data traffic patterns, i.e. depending on the type of data traffic that is used (RTM or RBE) a subset of the presented metrics is consulted. Table 5.13 shows an overview of the relevance of each metric as used for the analysis of the experiments, where ✓ means important for this traffic pattern class.

[4]Due to the fact that each node's trust value is in the interval of $(0, 1]$ the resulting product will also be in this range.

Metric	RTM	RBE
Average end-to-end delay	✓	
Average hop count	✓	
Routing Overhead	✓	✓
Average Residual Energy	✓	✓
Trust	✓	✓
PDR		✓
DDR		✓

Table 5.13: Relevance of the metrics depending on the application

5.3 Experiments and Evaluation

In the following, various experiments with different configurations are conducted to draw comparisons between the routing algorithms' performances and to answer the question, if the MOACO-based algorithms of MARFWSN are suitable for the routing in WSNs. Due to the fact that the MOACO-based algorithms have several configuration parameters themselves, in the first step, some pre-experiments are conducted to find the best parameters for the MOACO-based algorithms. After that, further experiments are conducted to compare the MARFWSN-based algorithms to the DSR-based routing algorithms, considering different network topologies and traffic patterns. Finally, the results of the experiments are discussed and conclusions are drawn.

5.3 Pre-Experiments: Finding Optimal Parameters for MOACO Algorithms

As discussed before, it is expected that the recommended ACO parameters from other COPs, such as TSP or QAP, cannot be directly transferred to the area of WSN routing. Therefore, a couple of pre-experiments were conducted to test in which WSN routing scenarios which MOACO parameters perform best[5]. An overview of the tested parameters is shown in table 5.14. The pre-experiments cover both, the *grid* and the *random* topology. For each topology the traffic patterns RTM and RBE were simulated with *one source* and *five destinations*. For all pre-

[5]Due to the fact that each simulation run needs highly computational power and takes multiple days for completion, not every possible combination of parameters can be tested. However, for each parameter a couple of different values were tested, which were derived from literature research and seemed to be reasonable.

experiments a balanced weighting of the considered objectives is used, i.e. $[energy = 0.33,\ duration = 0.33,\ trust = 0.33]$.

Parameter	Value
#ants per colony	$5, 10, 15, 20, 25, 30$ (in % of nodes)
ρ	$0.05, 0.1, 0.15, 0.2, 0.25$
τ_0	$0, 0.1, 0.25, 0.5, 0.75, 1$
τ_{min}	$0, 0.15, 0.2, 0.25, 0.3$
τ_{max}	$0.25, 0.5, 0.75, 1$
ξ	$0.05, 0.1, 0.15, 0.2, 0.25$
q_0	$0.5, 0.6, 0.7, 0.8, 0.9, 0.98$

Table 5.14: MOACO-related input parameters for the pre-experiments

Depending on the traffic type that is used in the simulation scenario, the corresponding metrics are considered, as discussed in the previous section. For each of the metrics from the corresponding subset of metrics, the best value of the considered parameter is chosen. To obtain the final aggregated result parameter, simply all selected parameters are averaged.[6]. It is clear that averaging the output parameters is not ideal, but some sort of trade-off needs to be made to choose the parameters for the next experiments.

Evaluation of the Pre-Experiments

As stated before, the pre-experiments aim at finding the best parameters for the MOACO-based routing algorithms. Because of the large amount of conducted pre-experiments and due to the fact that these pre-experiments are not the main focus of this research, in the following the results of the pre-experiments are only discussed briefly. Table 5.15 summarises the results of the conducted pre-experiments, which are subsequently used as parameters for the following experiments.

Number of Ants: The number of ants plays a central role for all MOACO-based routing algorithms because the more ants are used, the more ant packets for the routing need to be sent through the network[7].

[6]The averaging of the best parameters is done for simplicity reasons, of course other criteria could be used to find the final 'best' parameter. Additionally, it needs to be highlighted that in some cases the best parameter for a metric cannot be clearly identified. In this case, the parameter is ignored and not considered in the average.

[7]To be more precise, the number of ants (#ants) is given as percentage of the total number of nodes, e.g. for a scenario with 50 nodes, $\#ants = 10$ means 10 %

Topo.	Traffic	Dsts.	MOACO Alg.	Parameters
grid	RTM	5	SRMOACO	$\#ants = 5$
			ASMOACO	$\#ants = 5,\ \rho = 0.1,\ \tau_0 = 0$
			MMASMOACO	$\#ants = 5,\ \rho = 0.1,\ \tau_0 = 0.25,\ \tau_{min} = 0.3,\ \tau_{max} = 0.5$
			ACSMOACO	$\#ants = 10,\ \rho = 0.25,\ \tau_0 = 0.5,\ \xi = 0.15,\ q_0 = 0.8$
	RBE	5	SRMOACO	$\#ants = 20$
			ASMOACO	$\#ants = 10,\ \rho = 0.1,\ \tau_0 = 0.1$
			MMASMOACO	$\#ants = 10,\ \rho = 0.2,\ \tau_0 = 0.25,\ \tau_{min} = 0.15,\ \tau_{max} = 0.75$
			ACSMOACO	$\#ants = 20,\ \rho = 0.2,\ \tau_0 = 0.25,\ \xi = 0.2,\ q_0 = 0.8$
random	RTM	5	SRMOACO	$\#ants = 10$
			ASMOACO	$\#ants = 5,\ \rho = 0.1,\ \tau_0 = 0.1$
			MMASMOACO	$\#ants = 10,\ \rho = 0.1,\ \tau_0 = 0.25,\ \tau_{min} = 0.3,\ \tau_{max} = 0.5$
			ACSMOACO	$\#ants = 20,\ \rho = 0.15,\ \tau_0 = 0.5,\ \xi = 0.2,\ q_0 = 0.98$
	RBE	5	SRMOACO	$\#ants = 20$
			ASMOACO	$\#ants = 10,\ \rho = 0.1,\ \tau_0 = 0$
			MMASMOACO	$\#ants = 10,\ \rho = 0.2,\ \tau_0 = 0.25,\ \tau_{min} = 0.15,\ \tau_{max} = 0.75$
			ACSMOACO	$\#ants = 20,\ \rho = 0.2,\ \tau_0 = 0.25,\ \xi = 0.2,\ q_0 = 0.8$

Table 5.15: Results of the pre-experiments (note: #ant in % of nodes)

However, each transmission of an ant packet requires energy for its transmission so that for a large number of ants more energy is required which is not desirable for WSNs. Additionally, the more ant packets are sent, the likelihood of packet collisions will increase so that the overall performance of the network will be reduced by handling those on the lower layers. Consequently, a trade-off needs to be found between good routing performance, energy requirements and low number of packet collisions.

In general, it can be stated that for all tested MOACO-based algorithms the best results of the pre-experiments were obtained when the number of ants was between 5 and 20. For the grid topology, the RBE traffic pattern requires about twice as much ants for getting the optimal performance in comparison to the RTM pattern. For the random topology this observation can be a little bit weakened, though still for almost all MOACO-based routing algorithms more ants perform better in the RBE case.

On average, slightly more ants are required in the random topology than in the grid topology to obtain an optimal performance. This may be related to the fact that for the random topology it is more difficult to find optimal routes, in contrast to the uniformly distributed grid topology. Another interesting observation that can be made is that although `SRMOACO` does not rely on any other parameters, the algorithm performs best with a similar number of ants as for the other MOACO-based routing algorithms.

Evaporation Rate (ρ): To avoid a rapid convergence to a local optima, i.e. a globally suboptimal path, the evaporation of pheromone is utilised, controlled by the evaporation rate parameter ρ. However, a fine balance needs to be found between the depositing of new pheromone and the decaying of pheromone: on the one hand, if the evaporation rate is too high, the pheromone will disappear too quickly so that in the end the effect of the positive feedback tends to zero and, on the other hand, if the evaporation rate is too low, the amount of newly deposited pheromone will be too high so that the pheromone evaporation will have no influence

of 50 nodes, which are effectively 5 ants or more precise 5 ant packets. However, for simplicity reasons in the following just the term 'number of ants' is used.

at all.

In the pre-experiments the best results could be obtained by setting the evaporation rate between 0.1 and 0.25. For the experiments with grid topology the optimal evaporation rate is on average smaller or equal to the evaporation rate for the experiments with random topology. The requirement of a smaller evaporation rate for the grid topology may result from the fact that the routes in the grid can be found more easily due to the uniform distribution of the nodes so that the forgetting of suboptimal routes is not as important as in the random topology case. For both topologies, on average, the RTM traffic pattern performs better with a lower evaporation rate in comparison to the slightly higher ρ values for the RBE traffic pattern. It can be assumed that in the case of the RBE traffic pattern, the rapid forgetting of suboptimal routes leads to better optimised routes in terms of diversity, and thus to better results.

The evaporation rate for `ASMOACO` is for all tested scenarios set to 0.1 as optimal value. This may result from the fact that all forward ants are converted to backward ants, which then contribute to the updating of the pheromone. Consequently, a small evaporation rate is sufficient to obtain good results because all backward ants are contributing to the pheromone updating process by depositing new pheromone, also considering its evaporation. In contrast, for `MMASMOACO` and `ACSMOACO` the evaporation rate is in most cases higher than the 0.1 of `ASMOACO`, i.e. both perform best with roughly the same evaporation rate. Only two times, 0.25 performed best, while in the other cases a value between 0.1 and 0.2 was sufficient as evaporation rate.

Initial Pheromone (τ_0): The initial pheromone influences the ants that start with the first iteration. τ_0 is strongly linked to the evaporation rate ρ that is influencing the pheromone decay. When a small evaporation rate is set in combination with large initial pheromone, there is only a small difference on each pheromone update so that the exploration phase at the start is extended.

The best results in the pre-experiments were obtained by setting τ_0 to a value from the range between 0 and 0.5. In contrast to the pre-

experiments conducted with the random topology, on average, a better performance was obtained for the grid topology, when a higher τ_0 value was set. Consequently, it can be assumed that the longer exploration rate seems to be better exploitable in the grid topology. For the most experiments a larger τ_0 value leads to better results if only one destination is considered, particularly in the grid topology. The RTM and the RBE traffic pattern behave quite similar; however, for most cases RBE performs better with smaller τ_0 values.

Maximum Pheromone (τ_{max}): `MMASMOACO` limits the amount of pheromone by the parameter τ_{max} as pheromone upper bound. This avoids that the pheromone value is getting too high, resulting in a too strong convergence to the best-so-far route without considering any alternatives. The results of the pre-experiments show that the best results can be obtained when the value of τ_{max} is between 0.5 and 0.75.

The type of topology does not seem to have an influence on the performance, when setting the τ_{max} value. The only observation that can be made is that depending on the chosen traffic pattern a different τ_{max} value performs better: while in case of RTM a τ_{max} value of 0.5 provides the best simulation results, for RBE the best results could be obtained by setting τ_{max} to 0.75.

Another interesting observation of the conducted pre-experiments is that, differently from what was proposed in the original paper of MMAS, setting the initial pheromone value to τ_{max}, i.e. $\tau_0 = \tau_{max}$, did not result in the best performance of `MMASMOACO` for the tested scenarios.

Minimum Pheromone (τ_{min}): Additionally to the upper bound, `MMASMOACO` limits the pheromone by the parameter τ_{min} as lower bound. Depending on the chosen τ_{min} value, it can be avoided that the pheromone value becomes smaller than a certain value. The best results of the pre-experiments could be obtained in the case of setting τ_{min} in the range of 0.15 and 0.3

In comparison to the RTM traffic pattern, the RBE traffic performs on average better with smaller values of τ_{min}. All in all, the results of the

pre-experiments show that all tested MOACO-based algorithms perform best if τ_{min} is set to a value larger than 0, i.e. all the time there is some amount of pheromone in the network that is influencing the ants' decisions.

Local Pheromone Update Rate (ξ): The local pheromone update rate is only utilised by `ACSMOACO` to update each path locally, i.e. directly after using it, the amount of pheromone is reduced. Consequently, a fast convergence to certain paths is avoided so that a more explorative behaviour is encouraged. The best performance values in the pre-experiments were obtained for ξ values in the range between 0.15 and 0.25.

For almost all tested scenarios `ACSMOACO` performs best if the ξ value is smaller or equal to the ρ value. This make sense because if the ξ value is too high, the pheromone evaporation gets unbalanced so that the influence of ρ tends to zero. In the tested grid topology experiments `ACSMOACO` performs best for slightly smaller ξ values than in the random topology experiments. While for the RTM traffic pattern a higher ξ value leads to better results, for the RBE slightly lower values perform best.

Relative Importance of Exploitation vs. Exploration (q_0): In `ACSMOACO` the relative importance of exploitation vs. exploration can be set by the parameter q_0 in the range between 0 and 1, in which the values near 1 lead to a faster convergence to existing routes. Due to the fact that q_0 counteracts ξ, a fine balance between these parameters has to be found. The best results of the experiments for `ACSMOACO` could be obtained by setting q_0 to a value from the range of 0.6 and 0.98. More precise, when not considering the pre-experiment with grid topology and the RTM traffic pattern, all other pre-experiments perform best for ξ values from the range 0.8 and 0.98. All in all, obviously, the searching of good paths near the best-so-far paths results in a better performance than searching for completely new routes instead.

5.3 Comparison of MARFWSN-based and DSR-based Routing Algorithms

In the experiments, the MOACO-based routing algorithms of MARFWSN are compared to the DSR-based routing algorithm. The parameters gained from the pre-experiments are applied correspondingly to the MOACO-based algorithms. All routing algorithms are compared on different topologies (grid and random topology), with different traffic patterns (RTM and RBE traffic pattern) and with different percentages of malicious nodes (0 %, 25 %, 50 %, 75 %)[8].

Grid with Real-time Multimedia Data Traffic

In the following the experiments with a grid topology and the RTM data traffic pattern are discussed:

Application Packet End-to-End Delay: Figure 5.5 shows the average application packet end-to-end delay with standard deviation for the grid topology with the RTM traffic pattern. The graphs show the end-to-end delay in seconds on the y-axis and the number of nodes on the x-axis.

For the scenario with 0 % of malicious nodes (see figure 5.5a), it can be observed that the delay for the DSR-based routing algorithm is much smaller (< 0.1 s) than for the MOACO-based algorithms (between 0.05 s and 0.5 s) for all tested network sizes. The reason for this is that both routing approaches are based on different route discovery phases: while in DSR the first route that was found is used for the lifetime of the network, the MOACO-based algorithms trigger periodically the route discovery process so that the routes can be improved over several iterations. This results in a larger average end-to-end delay for the MOACO-based algorithms. `MMASMOACO` has a similarly low average end-to-end delay as the DSR-based algorithm. While `SRMOACO` and `ACSMOACO` are very slowly increasing (staying below 0.25 s), `ASMOACO` is increasing strongly starting from 64 nodes on, going up to 0.5 s. On average, the standard deviation of

[8] Note: for each used network metric the same scale is applied to all diagrams of the same type. The main reason for this is to ease the comparison between the different simulation scenarios.

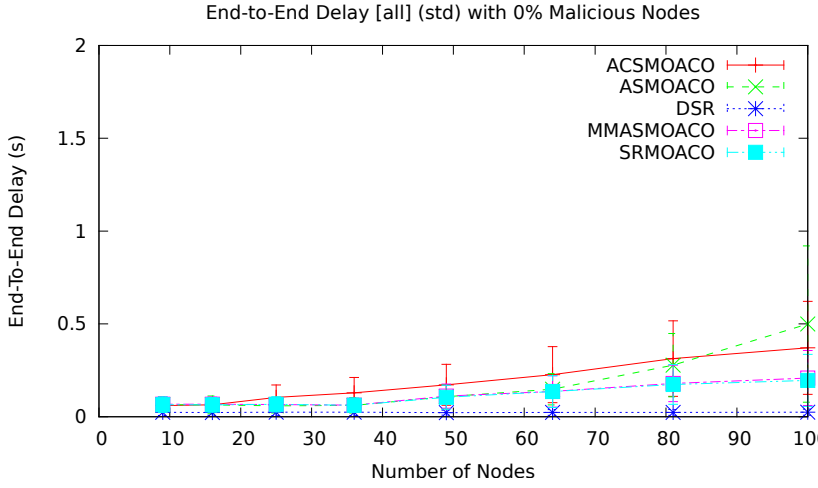
ACSMOACO is much higher than for the other MOACO-based algorithms. This can be explained by the fact that the local updating in ACSMOACO leads to a more explorative behaviour and hence, to a greater variety in the end-to-end delay.

If the percentage of malicious nodes in the network is increased to 25 %, to 50 % and to 75 % a similar end-to-end delay can be observed as for 0 % malicious nodes. Only the average end-to-end delay of ASMOACO is decreasing so that it converges to a similar average end-to-end delay as SRMOACO.

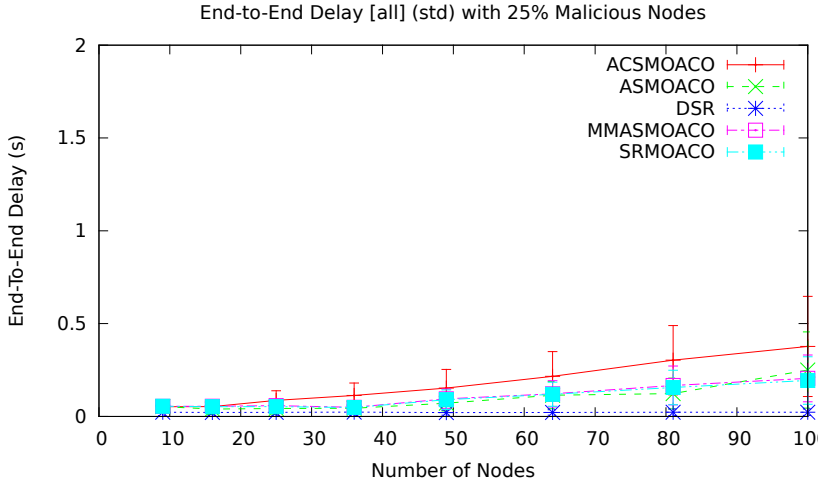
Application Packet Delivery Ratio: Figure 5.6 shows the average application PDR with standard deviation for the grid topology with the RTM traffic pattern. The graphs show the PDR on the y-axis and the number of nodes on the x-axis.

When there are no malicious nodes in the network (see figure 5.6a), it can be observed that from 9 to 36 nodes DSR outperforms all MOACO-based algorithms; only SRMOACO performs almost as good as the DSR-based algorithm. From 49 to 100 the MOACO-based algorithms, except ASMOACO, outperform the DSR-based algorithm in terms of average application PDR. ASMOACO shows the worst average application PDR for a low as well as for a high number of nodes, only for 36 and 49 nodes it performs similarly good as the other routing algorithms.

As expected, with the increasing of the percentage of malicious nodes to 25 %, to 50 % and to 75 %, it can be observed that the average application PDR is decreasing for all tested routing algorithms (see figures 5.6b, 5.6c and 5.6d). The average application PDR of the DSR-based and the MOACO-based algorithms becomes closer with each increasing of the malicious nodes. The only real difference can be observed between 9 and 49 nodes, in which the DSR-based algorithm performs best, followed by SRMOACO, ACSMOACO and MMASMOACO, and finally, ASMOACO as worst. From 49 nodes on all routing algorithms have a similar average application PDR so that they are not distinguishable any more, particularly if there is a high number of malicious nodes in the network. For most MOACO-based algorithms it can be observed that the lower the number of ants

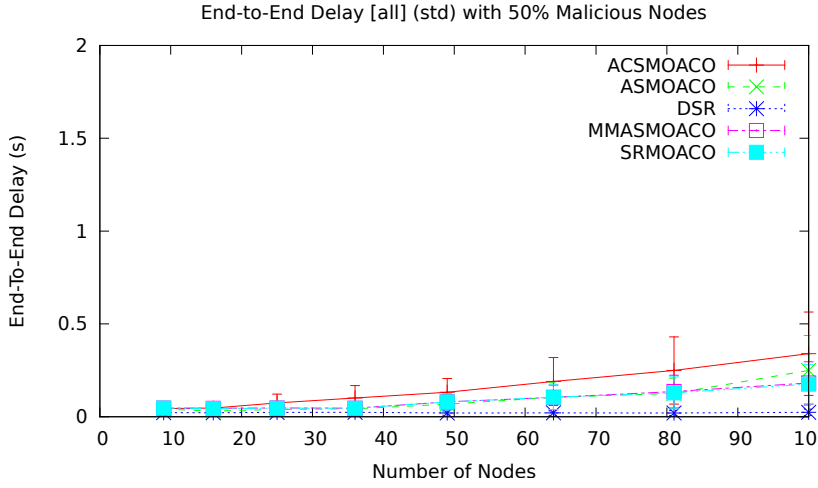


(a) 0% Malicious Nodes

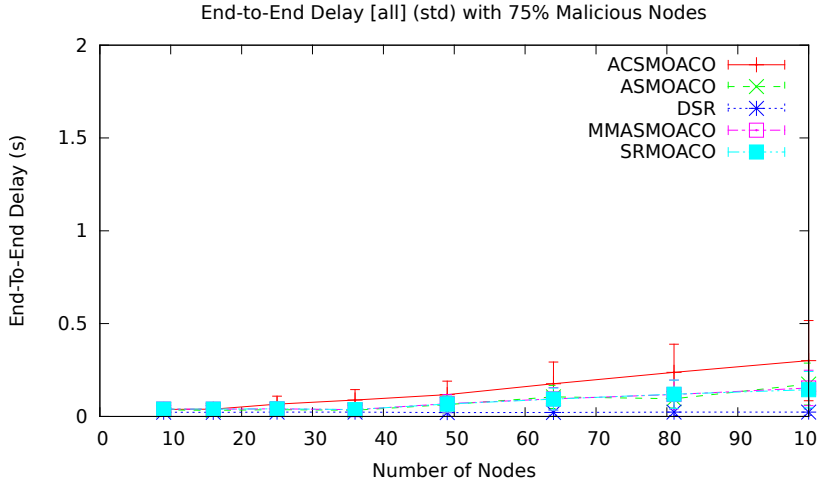


(b) 25% Malicious Nodes

Figure 5.5: Application packet average end-to-end delay with different percentages of malicious nodes for grid topology with RTM traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.5: Application packet average end-to-end delay with different percentages of malicious nodes for grid topology with RTM traffic pattern

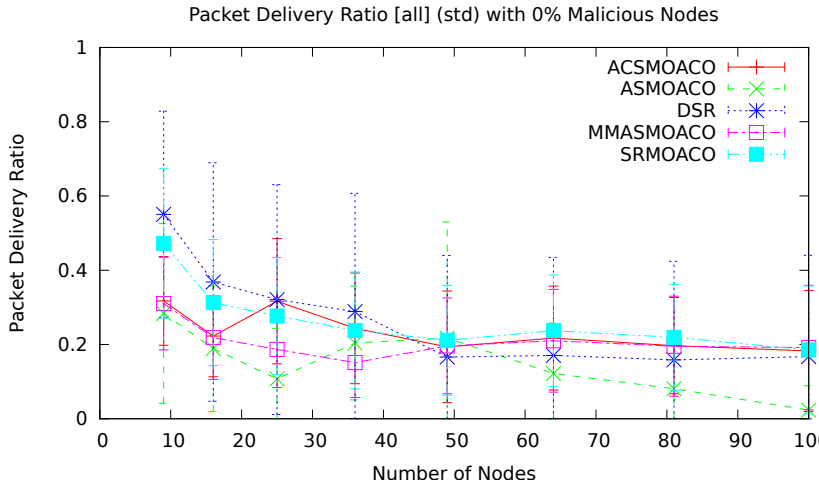
used for the algorithm, the lower the average application PDR. However, SRMOACO performs in this case astonishingly good, though it is just based on random movement with a small number of ants.

Number of Hops: Figure 5.7 shows the average number of hops with standard deviation for the grid topology with the RTM traffic pattern. The graphs show the number of hops on the y-axis and the number of nodes on the x-axis.

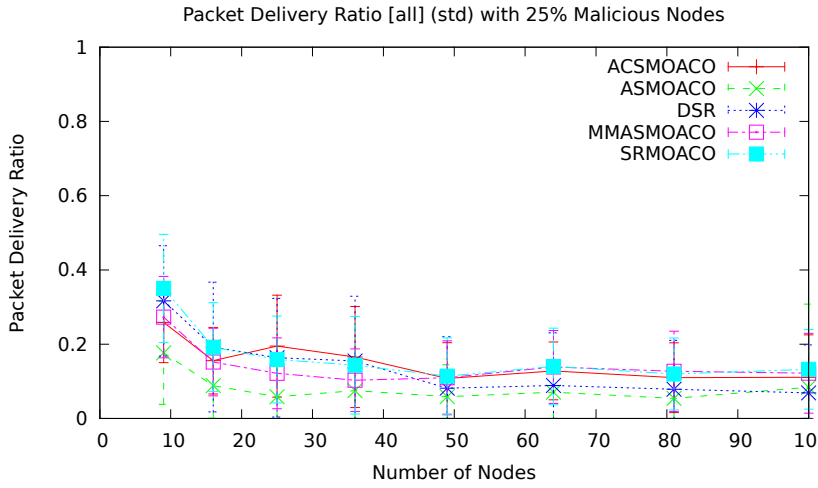
For 0 % of malicious nodes (see figure 5.7a) it can be observed that all MOACO-based algorithms required on average between ~ 2.3 and ~ 3.2 hops. In contrast, for the DSR-based algorithm up to 36 nodes only between ~ 1.5 and ~ 2.6 hops are required, for more than 36 nodes the number of hops stays around 3.

When the percentage of malicious nodes is increased to 25 %, to 50 % and to 75 %, it can be observed that the average number of hops for MOACO-based algorithms is slightly decreased so that they are, on average, located rather constantly around 2 hops. In contrast, the average number of hops for the DSR-based algorithm is only very slightly decreased so that it needs a little bit less than 3 hops on average. However, the distance between the average number of hops for the DSR-based algorithm and the MOACO-based algorithms is larger, the more malicious nodes are in the network, for network sizes with more than 25 nodes. The observation that the average number of hops of the MOACO-based algorithms is rather constant can be explained by the fact that the ant-based algorithms seem to converge much faster to (near) optimal routes because the diversity of routes is decreased by loosing ants on routes with malicious nodes, while the good routes are reinforced by pheromone. This is also supported by the decreasing standard deviation of the MOACO-based algorithms the more malicious nodes are in the network.

Residual Energy: Figure 5.8 shows the average residual energy with standard deviation for the grid topology with the RTM traffic pattern. The graphs show the residual energy in % on the y-axis and the number of nodes on the x-axis.

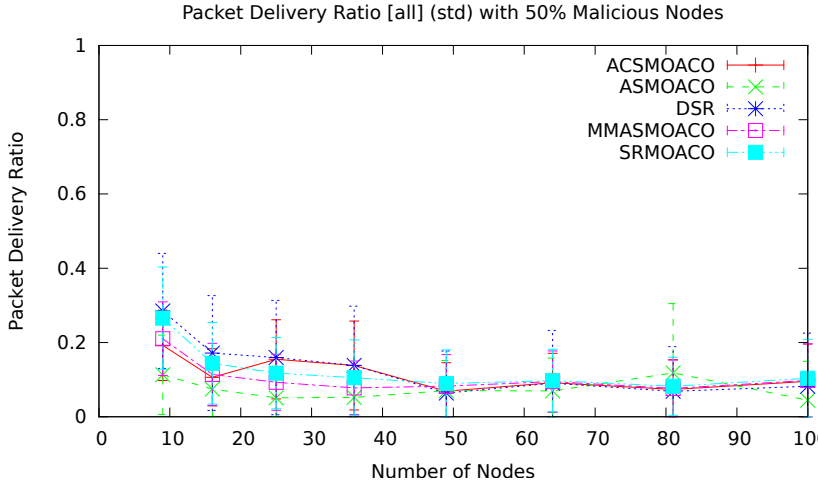


(a) 0% Malicious Nodes

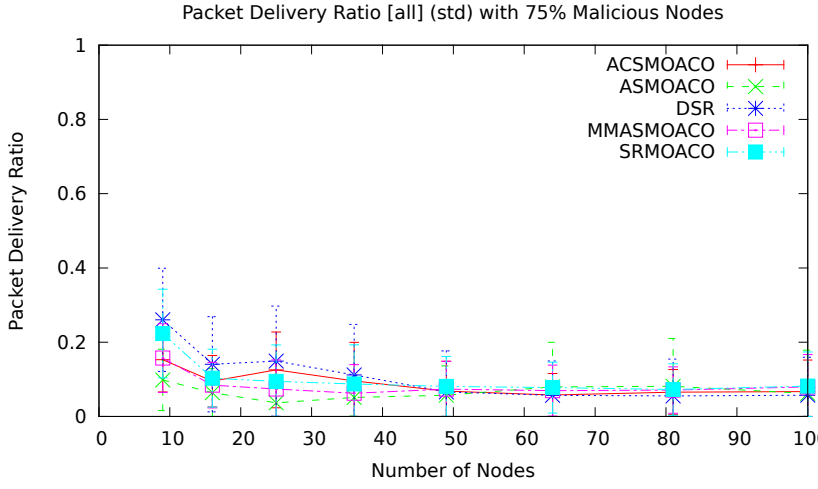


(b) 25% Malicious Nodes

Figure 5.6: Application PDR with different percentages of malicious nodes for grid topology with RTM traffic pattern

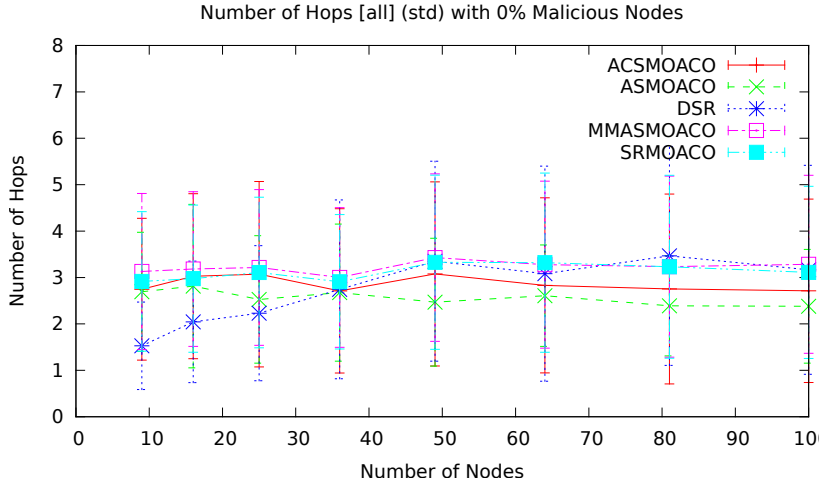


(c) 50% Malicious Nodes

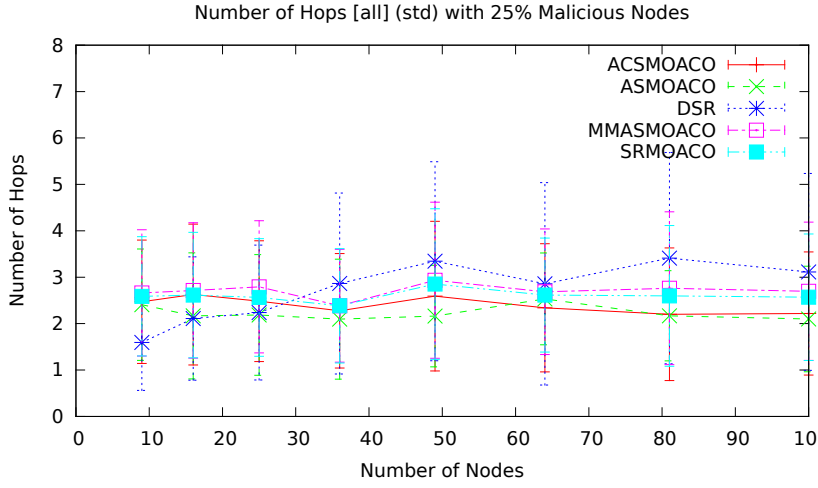


(d) 75% Malicious Nodes

Figure 5.6: Application PDR with different percentages of malicious nodes for grid topology with RTM traffic pattern

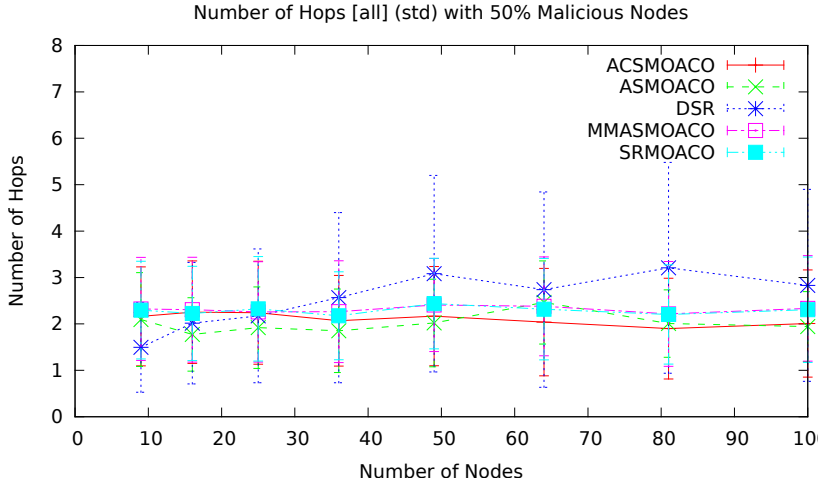


(a) 0% Malicious Nodes

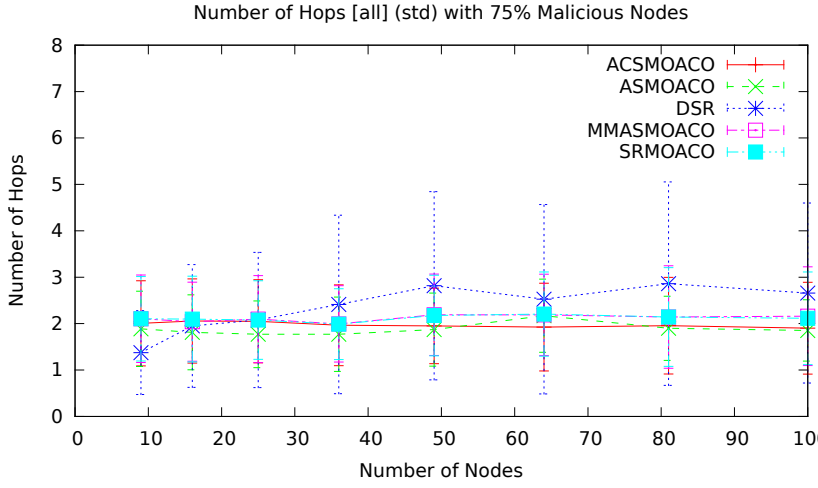


(b) 25% Malicious Nodes

Figure 5.7: Number of hops with different percentages of malicious nodes for grid topology with RTM traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.7: Number of hops with different percentages of malicious nodes for grid topology with RTM traffic pattern

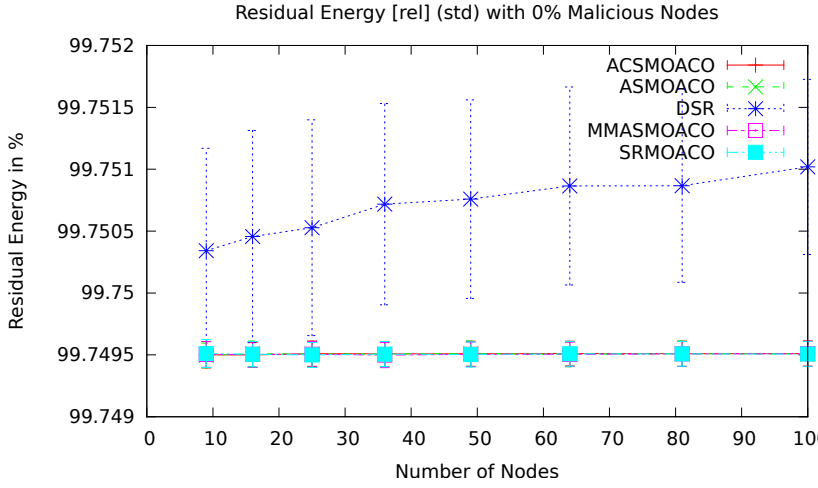
The results show that for the DSR-based routing algorithm more residual energy is left than for the MOACO-based algorithms for all tested scenarios. While the residual energy for the DSR-based algorithm is slightly increased, when increasing the number of nodes in the network, for the MOACO-based routing algorithms the residual energy stays quite constant for all tested scenarios. However, the standard deviation in the DSR-based routing algorithm is much higher than for the MOACO-based algorithms. Interestingly, the MOACO-based algorithms can not be distinguished, although they are influenced by different parameters and act differently. That slightly more energy is required for the MOACO-based algorithms can be explained by the fact that these algorithms use several iterations of sending multiple forward ants in the discovery process, while the DSR-based algorithm is based on a single broadcasting and there are no further iterations for improving a route, when already one working route was found. Besides, the required broadcasting of `HELLO`-messages that is required to indentify neighbours in the MOACO-based routing algorithms, is requiring additional energy. Increasing the percentage of malicious nodes does not seem to have a great impact on the residual energy for all tested routing algorithms.

Although, the residual energy for the DSR-based algorithm looks much more than for the MOACO-based algorithms in the diagrams, it has to be highlighted that this difference is for the absolute values is very small[9].

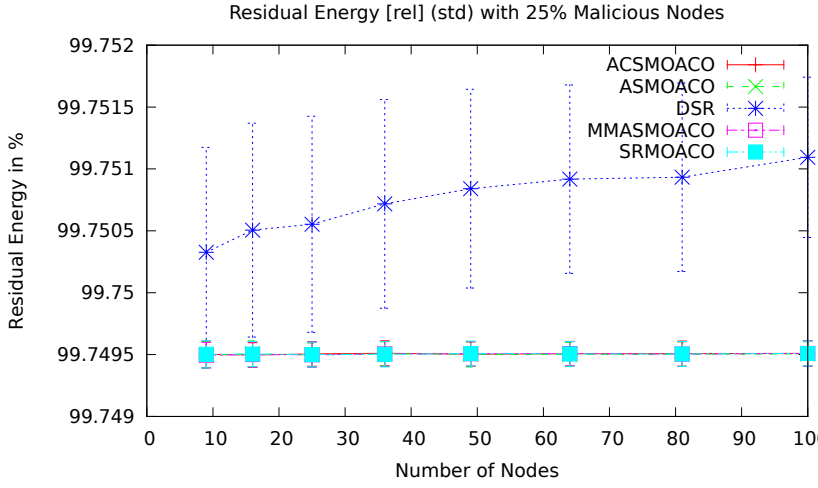
Routing Overhead: Figure 5.9 shows the average routing overhead with standard deviation for the grid topology with the RTM traffic pattern. The graphs show the routing overhead in % on the y-axis and the number of nodes on the x-axis.

For 0 % of malicious nodes (see figure 5.9a) it can be observed, that the DSR-based routing algorithm has a slightly lower average overhead ($\sim$ 5 %) in comparison to the MOACO-based routing algorithms (between $\sim$ 5 % and $\sim$ 23 %). The average routing overhead for the MOACO-based algorithms is slightly increased, when the number of nodes in network is increased, while the DSR-based algorithm stays on a constantly low level.

[9]For a better evaluation of the energy, in the future, the simulation time needs to be increased to obtain more significant values. The results of the current simulations should be rather seen as tendency.

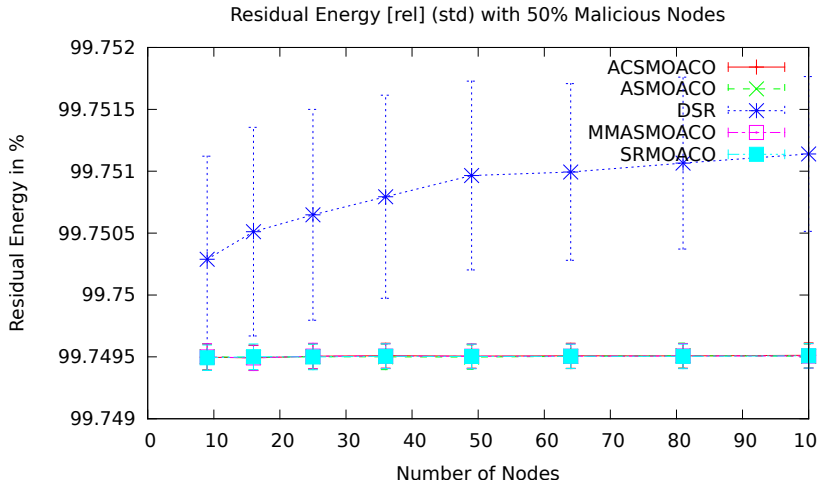


(a) 0% Malicious Nodes

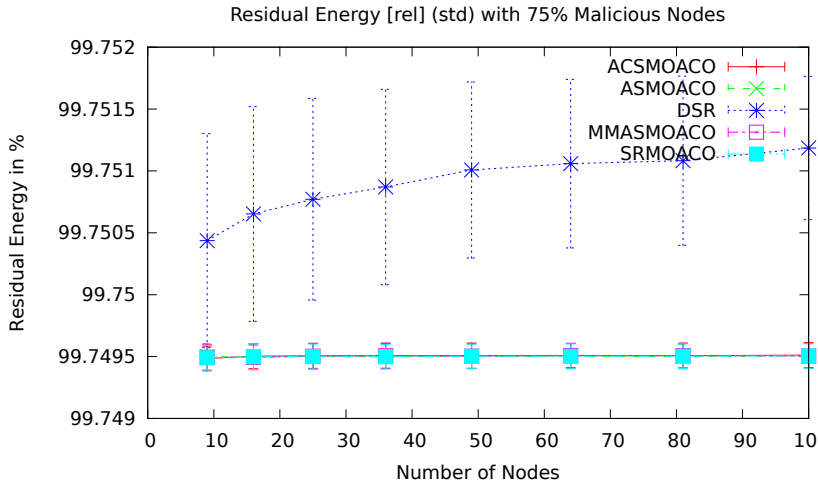


(b) 25% Malicious Nodes

Figure 5.8: Residual energy with different percentages of malicious nodes for grid topology with RTM traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.8: Residual energy with different percentages of malicious nodes for grid topology with RTM traffic pattern

While MMASMOACO and SRMOACO show a maximum average overhead of ∼ 10 %, ASMOACO and ACSMOACO have an average overhead around 23 %, with ASMOACO having the highest average overhead of the tested algorithms.

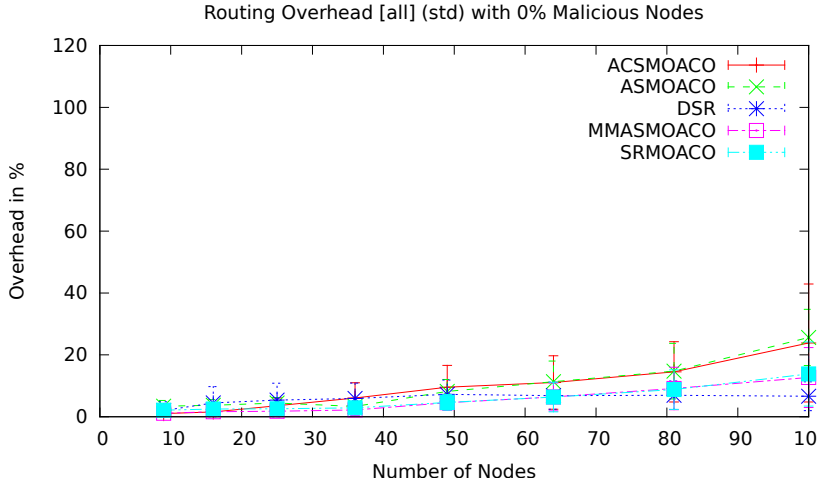
For 25 % of malicious nodes (see figure 5.9b) it can be observed that the average overhead of MMASMOACO and SRMOACO is only very slightly increasing, while the average overhead for ASMOACO and ACSMOACO is increasing, up to 30 % for ACSMOACO and up to 25 % for ASMOACO.

Interestingly, for 50 % and 75 % of malicious nodes (see figures 5.9c and 5.9d), the average overhead of MMASMOACO and SRMOACO increases up to ∼ 18 %, while the average overhead for ASMOACO stays almost the same, and the overhead of ACSMOACO increases again up to ∼ 40 % in worst case. The average overhead of the DSR-based routing algorithm stays low, when the percentage of malicious nodes is increased. The strongest rise can be observed for ACSMOACO in comparison to the other MOACO-based algorithms.

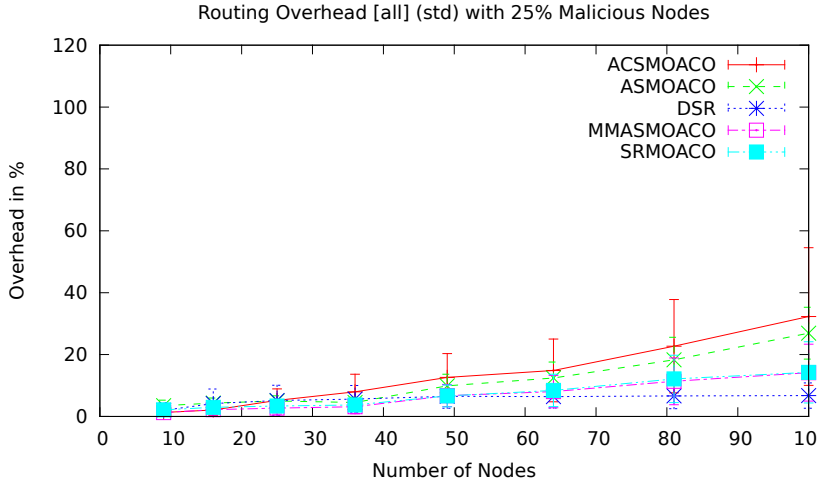
The great difference between the DSR-based and the MOACO-based routing algorithms can be explained by the fact that the MOACO-based algorithm use several iterations to improve the routes so that the ratio between network layer packets and application layer packets becomes worse, in contrast to the DSR-based algorithm that is using the first route found in the route discovery process. The difference in the overhead of the MOACO-based algorithms is mainly based on the number of ants that are used in which algorithm.

Trust: Figure 5.10 shows the average trust value with standard deviation for the grid topology with the RTM traffic pattern. The graphs show the trust value on the y-axis and the number of nodes on the x-axis.

For 0 % of malicious nodes (see figure 5.10a) it can be observed that all MOACO-based routing algorithms have an average trust value between 0.5 and 0.7. Considering the different number of nodes, ASMOACO provides the best average trust values for almost all tested scenarios, except for 64 and 100 nodes ACSMOACO performs better. The DSR-based algorithm performs significantly worse in comparison to the MOACO-based algorithms. This result is as expected because the DSR-based routing algorithm does

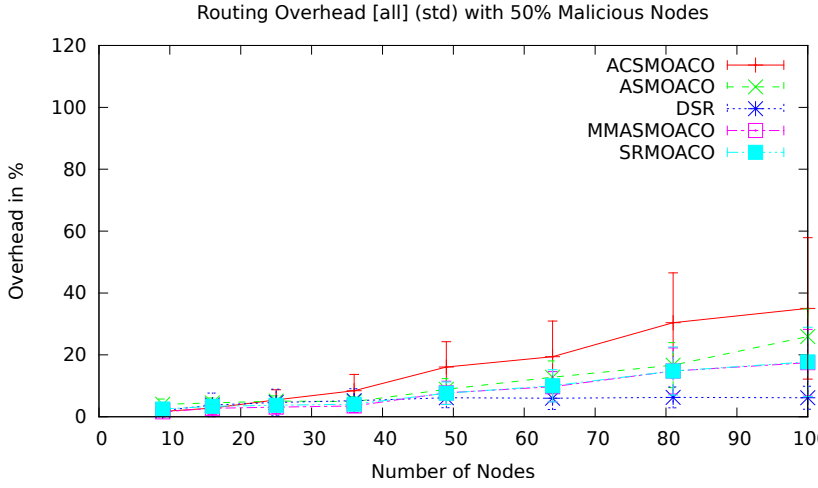


(a) 0% Malicious Nodes

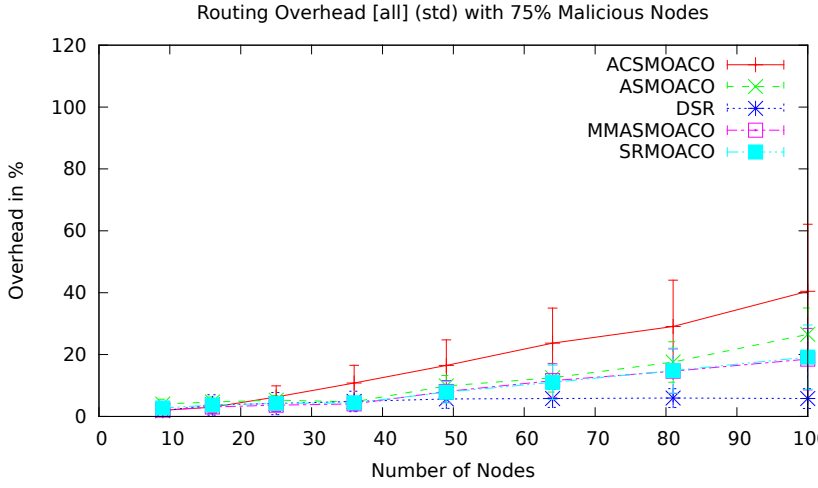


(b) 25% Malicious Nodes

Figure 5.9: Routing overhead with different percentages of malicious nodes for grid topology with RTM traffic pattern



(c) 50% Malicious Nodes



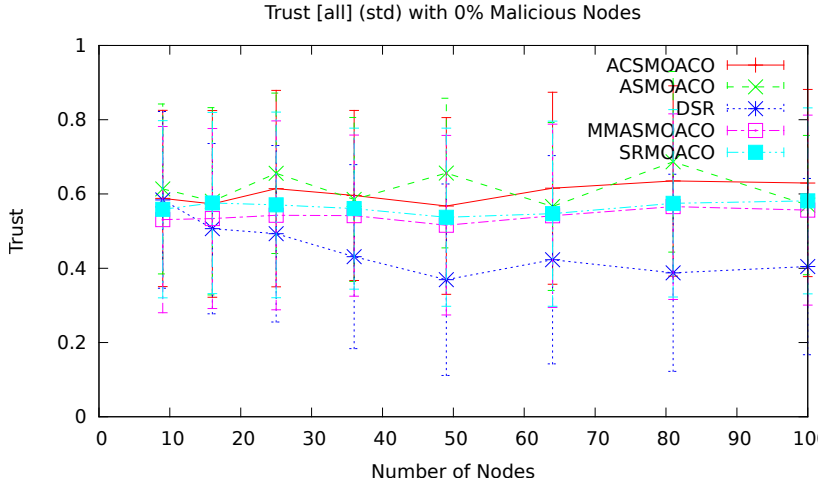
(d) 75% Malicious Nodes

Figure 5.9: Routing overhead with different percentages of malicious nodes for grid topology with RTM traffic pattern

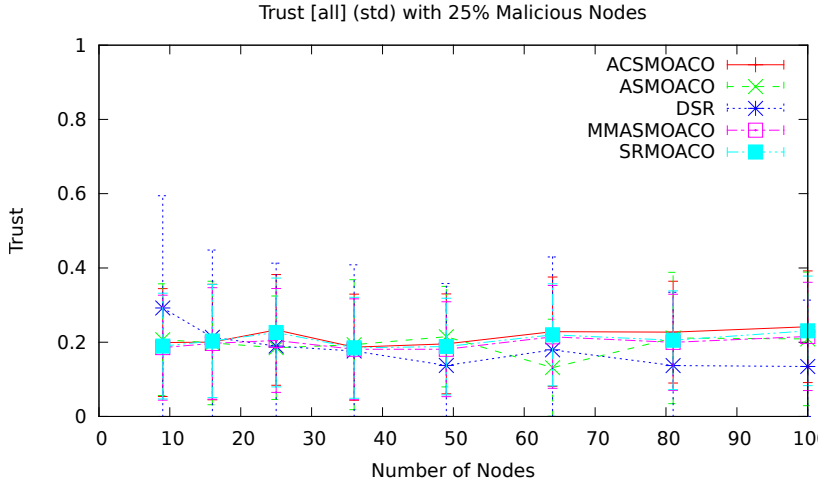
not optimise its route regarding trust, while for the MOACO-based algorithms this is one of the desired optimisation features.

With 25 % and 50 % of malicious nodes (see figures 5.10b and 5.10c), the average trust values of all tested routing algorithms get significantly worse and are moving closer together. The average trust values are then in the range between 0.1 and 0.3. For more than 36 nodes the DSR-based algorithms performs worst (except the outlier of `ASMOACO` for 64 nodes).

With 75 % of malicious nodes (see figure 5.10d) the average trust value of `DSR` is again reduced (~ 0.1) on average, while the MOACO-based algorithms perform similar to 25 % or 50 % of malicious nodes. As expected, the MOACO-based algorithms outperform the DSR-based algorithm regarding the trust value, particularly in networks with many malicious nodes.

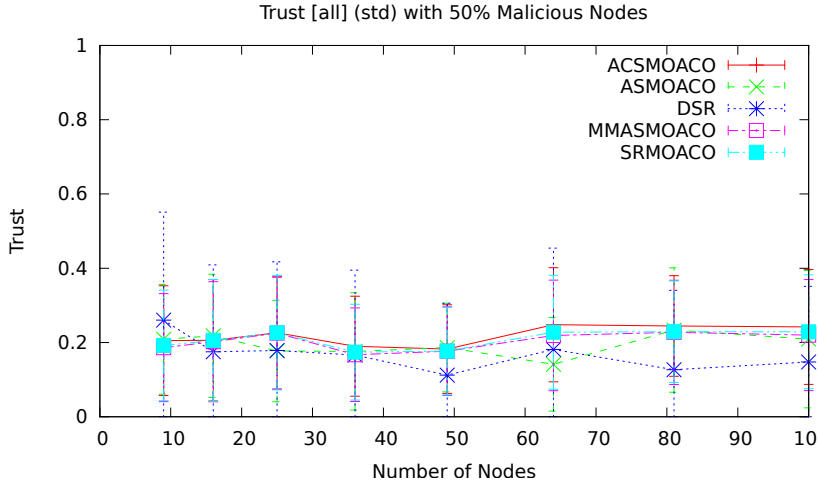


(a) 0% Malicious Nodes

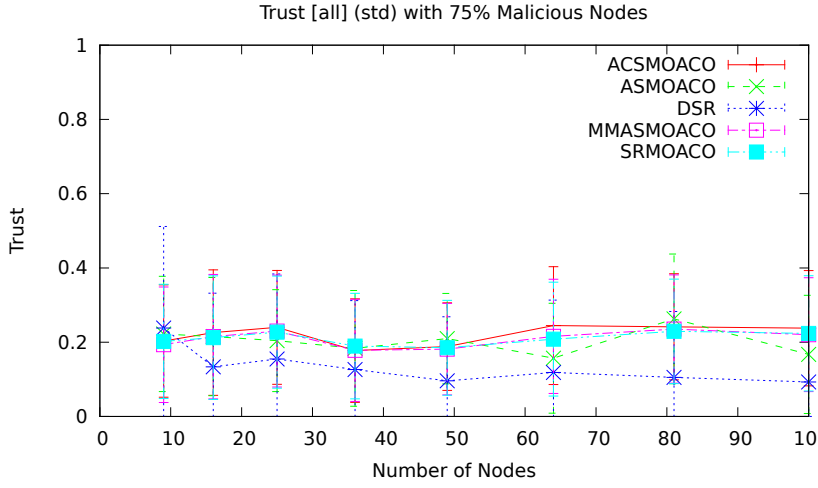


(b) 25% Malicious Nodes

Figure 5.10: Trust with different percentages of malicious nodes for grid topology with RTM traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.10: Trust with different percentages of malicious nodes for grid topology with RTM traffic pattern

Grid with Reliable Best-Effort Data Traffic

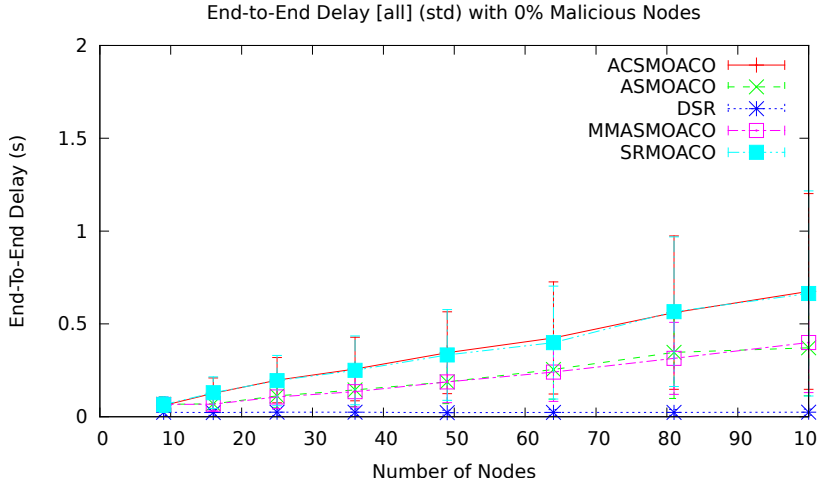
In the following the experiments with a grid topology and the RBE data traffic pattern are discussed:

Application Packet End-to-End Delay: Figure 5.11 shows the average application packet end-to-end delay with standard deviation for the grid topology with the RBE traffic pattern. The graphs show the end-to-end delay in seconds on the y-axis and the number of nodes on the x-axis.

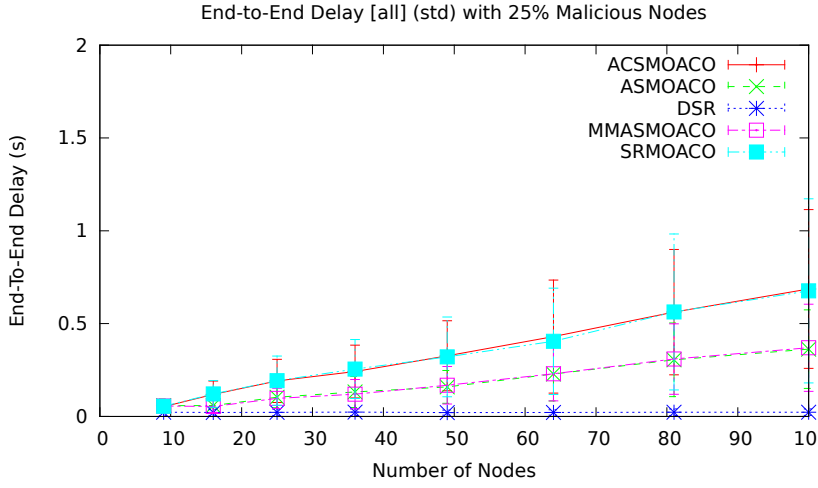
In comparison to the RTM traffic pattern, the average end-to-end delay of the MOACO-based algorithms for the RBE traffic pattern on the grid topology with 0 % malicious nodes is slightly higher, whereas the DSR-based routing algorithm stays on a constantly low level (see figure 5.11a). The largest increase can be observed for `ACSMOACO` and `SRMOACO` which have a worst average end-to-end delay of ~ 0.6 s. `MMASMOACO` and `ASMOACO` have a slightly better average end-to-end delay reaching an average application packet end-to-end delay of ~ 0.4 s at worst. The increase in the average application packet end-to-end delay of the MOACO-based algorithms can be explained by the increased amount of ants that is used in the RBE scenario compared to the previous RTM scenario (see table 5.15).

When the percentage of malicious nodes is increased to 25 %, to 50 % and to 75 %, the average end-to-end delay of the MOACO-based algorithm is slightly decreased in each step (see figures 5.11b, 5.11c and 5.11d). The worst case average end-to-end delay for `SRMOACO` and `ACSMOACO` is decreased to ~ 0.52 s and for `MMASMOACO` and `ASMOACO` to ~ 0.25 s. The reduced average end-to-end delay for the MOACO-based algorithms may result from the fact that more packets are dropped and hence, the convergence to (near) optimal paths is faster. Besides, when packets are dropped, less packets need to be forwarded, resulting in less packet collisions, which cause additional delay.

Application Packet Delivery Ratio: Figure 5.12 shows the average application PDR with standard deviation for the grid topology with the

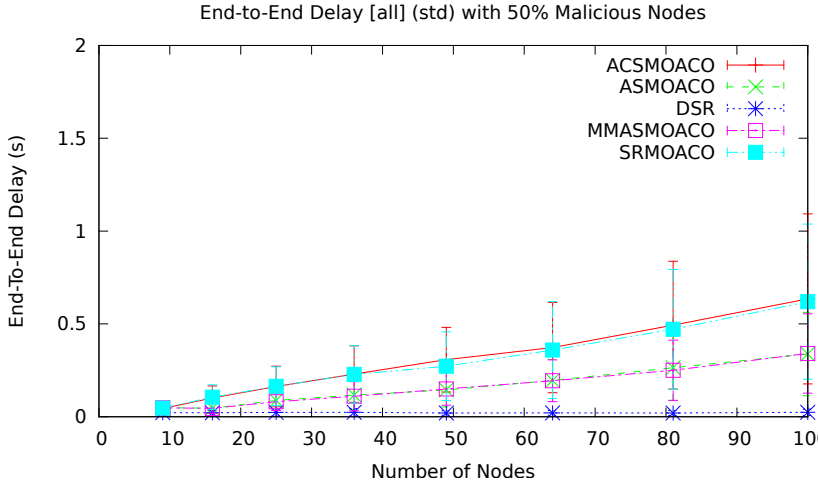


(a) 0% Malicious Nodes

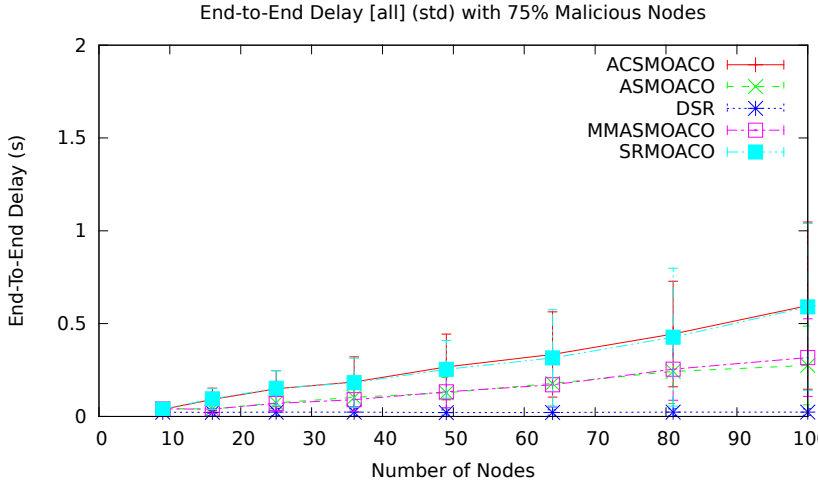


(b) 25% Malicious Nodes

Figure 5.11: Application packet average end-to-end delay with different percentages of malicious nodes for grid topology with RBE traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.11: Application packet average end-to-end delay with different percentages of malicious nodes for grid topology with RBE traffic pattern

RBE traffic pattern. The graphs show the PDR on the y-axis and the number of nodes on the x-axis.

For the scenario with 0 % of malicious nodes, it can be observed that for ASMOACO the average application PDR is really high compared to the other algorithms (see figure 5.12a). Moreover, it can be observed that starting from 25 all other routing algorithms perform on a similar level, and starting from 50 nodes all MOACO-based algorithms even outperform the DSR-based algorithm in terms of average application PDR.

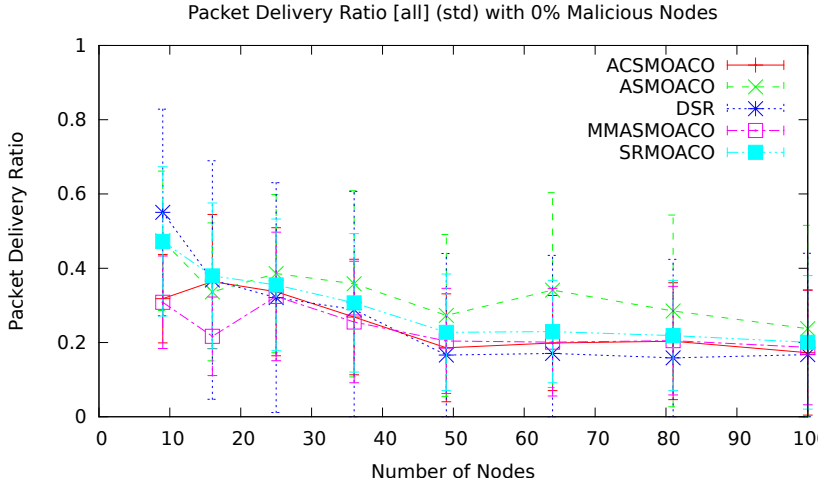
When the number of malicious nodes is increased to 25 % (see figure 5.12b) the average application PDR of all tested routing algorithms becomes worse, but all MOACO-based algorithms outperform the DSR-based routing algorithm for network sizes with more than 25 nodes. After a decrease from ~ 0.38 for 9 nodes to ~ 0.18 for 36 nodes, the MOACO-based algorithms stay almost constant around ~ 0.1 from 50 nodes on.

For 50 % and 75 % of malicious nodes, the average application PDR is again a little bit decreased for all tested routing algorithms (see figures 5.12c and 5.12d). Except ASMOACO, the other routing algorithms perform similarly bad so that they cannot be distinguished any more.

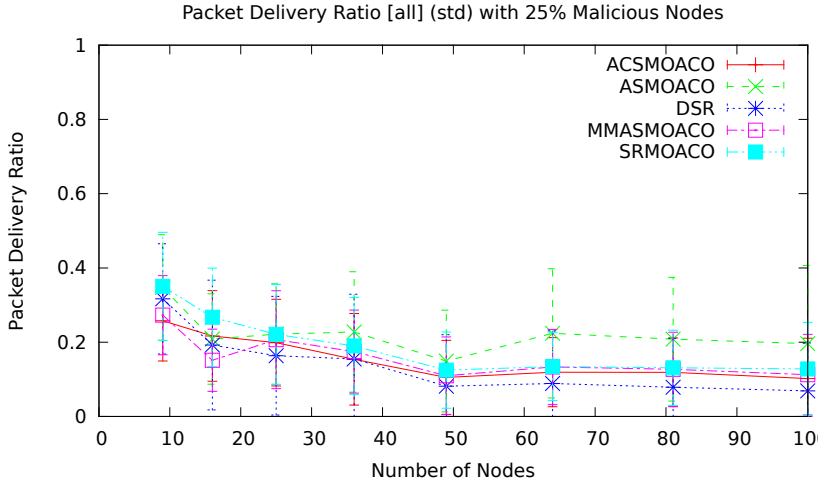
As expected, the tuning of the MOACO-based algorithms' parameters lead to a slight improvement of the average PDRs at the cost of sending more ants. The largest effect could be observed for ASMOACO, which shows a significant improvement for the average application PDR in case of the RBE traffic pattern.

Number of Hops: Figure 5.13 shows the average number of hops with standard deviation that are required for the grid topology with the RBE traffic pattern. The graphs show the number of hops on the y-axis and the number of nodes on the x-axis.

Similar to the previous experiment with the grid topology and the RTM traffic pattern, the average number of hops for the MOACO-based algorithms stays almost constant when the number of nodes in the network is increased (compare figure 5.7). On average the most hops are required by ASMOACO (~ 3.5), while the least hops are required by ACSMOACO (~ 2.8) . In comparison the MOACO-based algorithms, the DSR-based algorithm

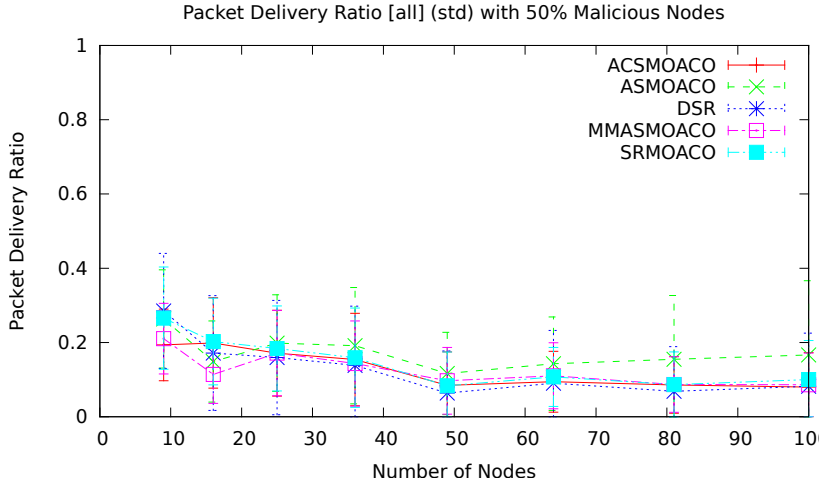


(a) 0% Malicious Nodes

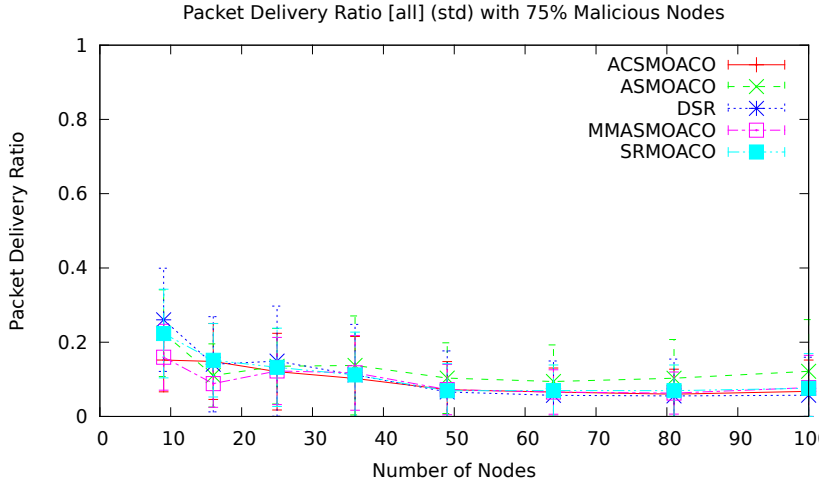


(b) 25% Malicious Nodes

Figure 5.12: Application PDR with different percentages of malicious Nodes for grid topology with RBE traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.12: Application PDR with different percentages of malicious Nodes for grid topology with RBE traffic pattern

has the lowest average number of hops up to 25 nodes (between $\sim$ 1.5 and $\sim$ 2.1) and then, it raises further so that it is on a similar level as the other MOACO-based algorithms (between $\sim$ 2.8 and $\sim$ 3.3).

When the percentage of malicious nodes is increased, from 25 % to 50 % to 75 %, the average number of hops for the MOACO-based algorithms is slightly reduced so that they are getting closer together (between 2.5 and 2 hops), while the average number of hops for the DSR-based algorithms stays between 1.5 and 3 hops. It can be observed that the more malicious nodes are in the network, the larger gets the gap between the MOACO-based algorithms and the DSR-based algorithms in terms of average number of hops.

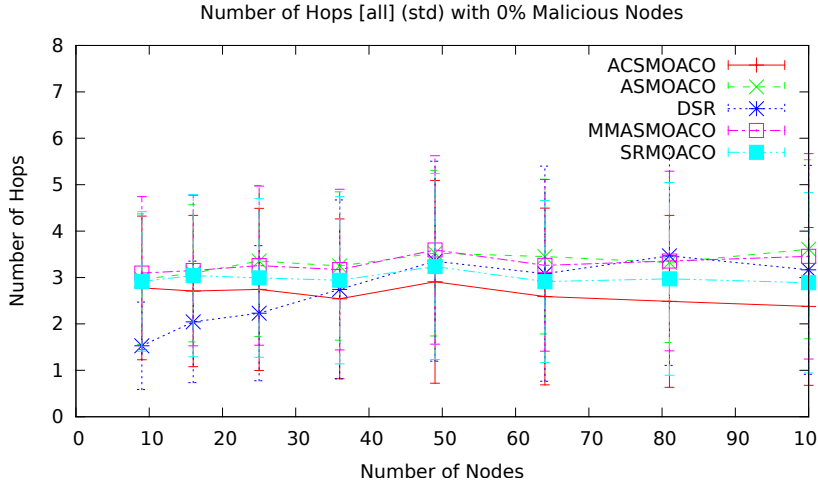
As explained before, the rather constant behaviour of the MOACO-based algorithms can be explained by the fact that the MOACO-based algorithms converge to (near) optimal solutions with a small number of hops, while the broadcasting of the DSR-based algorithm also utilises longer routes with multiple hops.

Residual Energy: Figure 5.14 shows the average residual energy with standard deviation for the grid topology with the RBE traffic pattern. The graphs show the residual energy in % on the y-axis and the number of nodes on the x-axis.

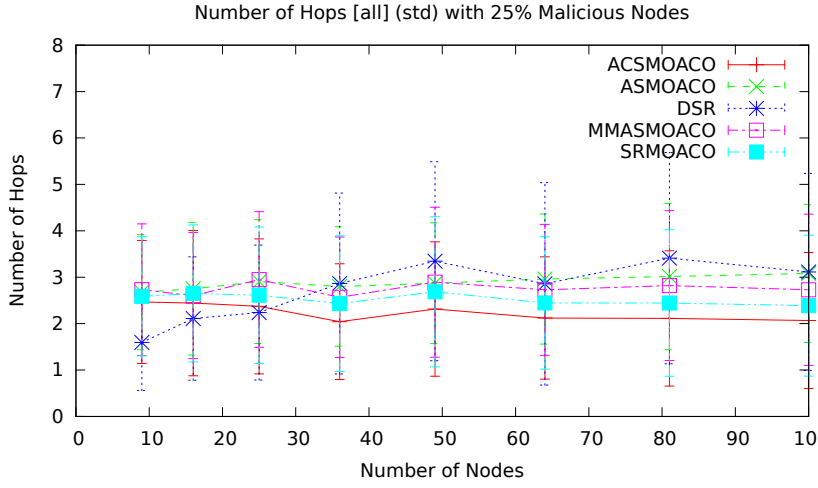
Again, as in the previous scenario based on the grid topology, very similar values of the average residual energy can be observed (compare figure 5.8) so that it can be assumed that the traffic pattern for the grid topology does not have a great influence on the residual energy of the nodes.

Routing Overhead: Figure 5.15 shows the average routing overhead with standard deviation for the grid topology with the RBE traffic pattern. The graphs show the routing overhead in % on the y-axis and the number of nodes on the x-axis.

For the scenario with 0 % of malicious nodes (see figure 5.15a), the average routing overhead of the MOACO-based algorithms is between $\sim$ 2 % and $\sim$ 52 %, in worst case. `SRMOACO` and `ACSMOACO` increase a little bit more strongly (up to $\sim$ 52 %), while `MMASMOACO` and `ASMOACO` increase more

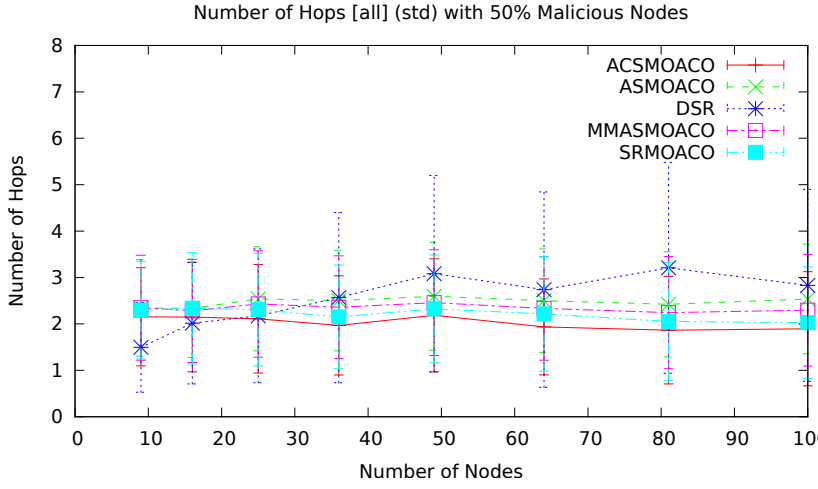


(a) 0% Malicious Nodes

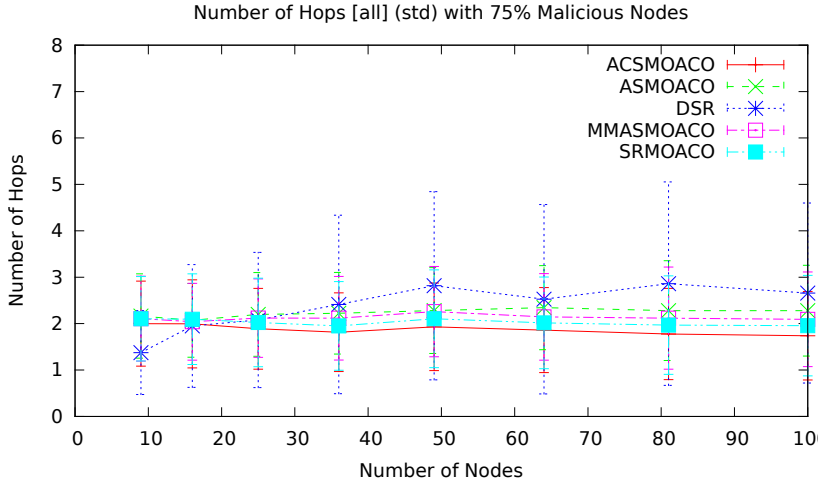


(b) 25% Malicious Nodes

Figure 5.13: Number of hops with different percentages of malicious nodes for grid topology with RBE traffic pattern

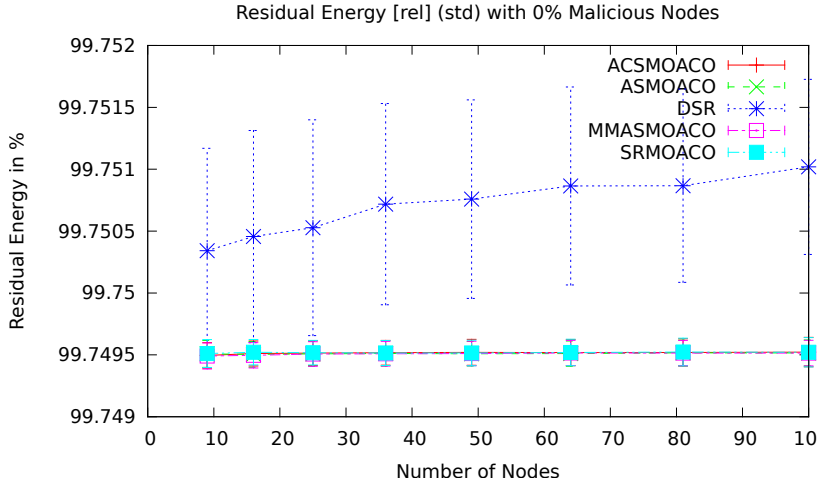


(c) 50% Malicious Nodes

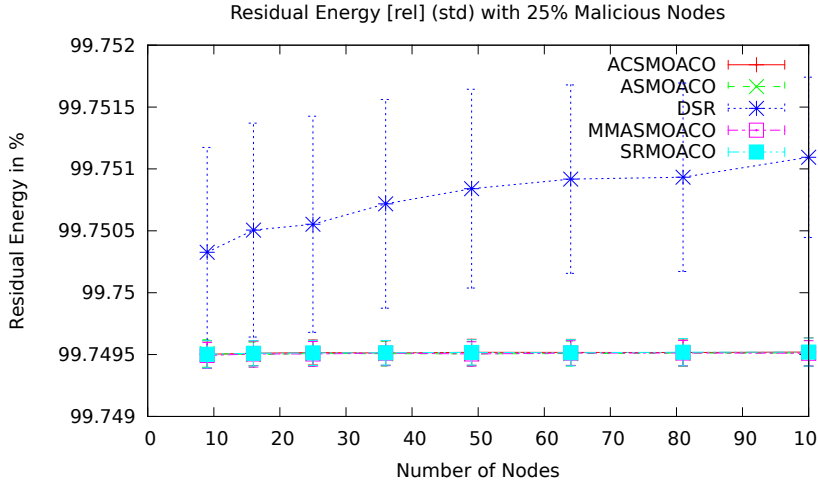


(d) 75% Malicious Nodes

Figure 5.13: Number of hops with different percentages of malicious nodes for grid topology with RBE traffic pattern

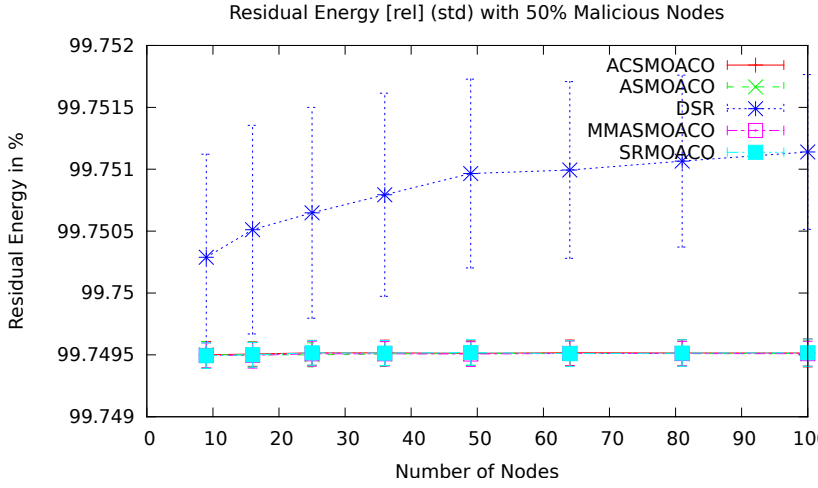


(a) 0% Malicious Nodes

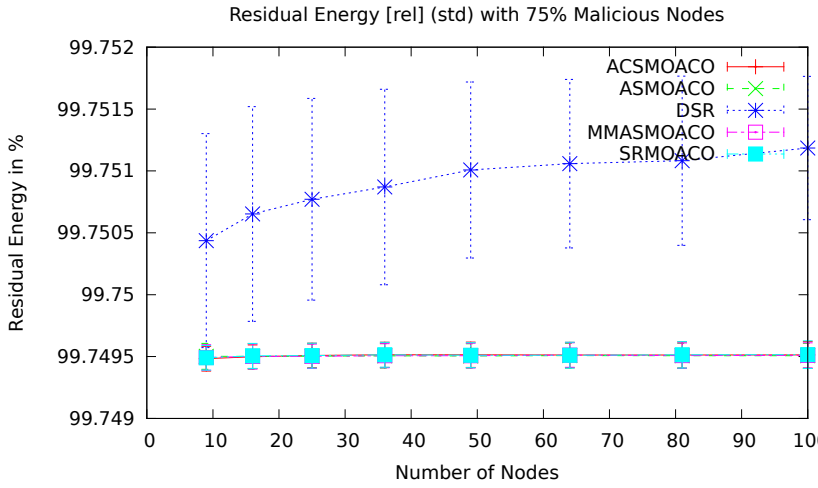


(b) 25% Malicious Nodes

Figure 5.14: Residual energy with different percentages of malicious nodes for grid topology with RBE traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.14: Residual energy with different percentages of malicious nodes for grid topology with RBE traffic pattern

slowly (up to $\sim 21\,\%$). The DSR-based routing algorithms has the lowest average overhead around $7\,\%$ and stays rather constant starting from 25 nodes.

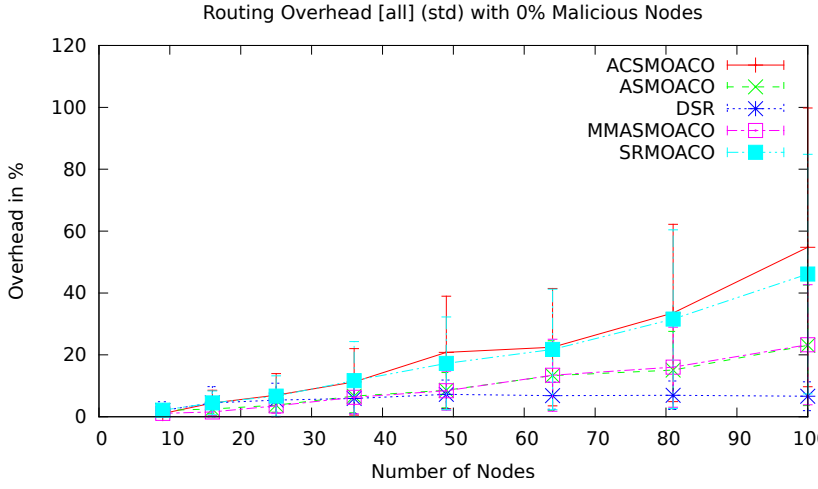
By increasing the number of malicious nodes in the network from $25\,\%$ to $50\,\%$ to $75\,\%$ (see figures 5.15b, 5.15c and 5.15d), the routing overhead of the MOACO-based algorithms is additionally increased. At worst, `ACSMOACO` and `SRMOACO` have the highest overhead around $78\,\%$. `MMASMOACO` and `ASMOACO` are the MOACO-based algorithms with the lowest overhead with around $38\,\%$ in worst case. In contrast, the DSR-based algorithm stays constant, when the percentage of malicious nodes is increased.

Basically, it can be observed that the increase in the number of ants for the MOACO-based algorithms leads to a larger routing overhead, in contrast to the previous scenario with the RTM traffic pattern. The more ants are used by the MOACO-based algorithms, the higher is the average routing overhead.

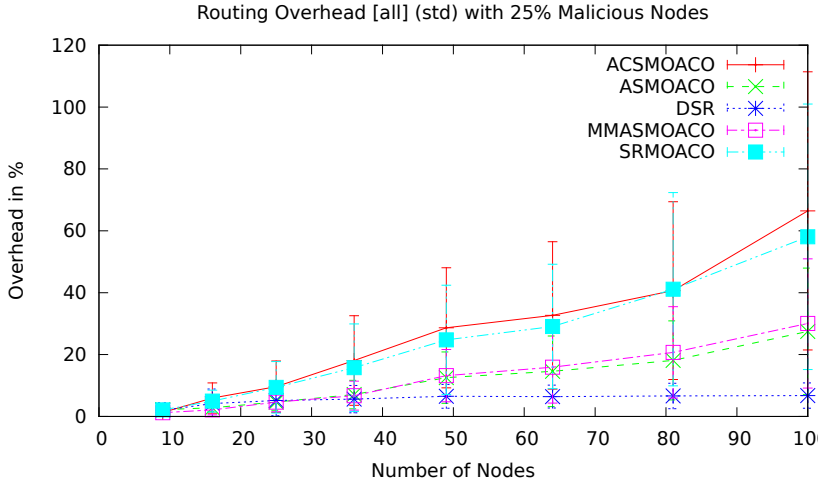
Trust: Figure 5.17 shows the average trust value with standard deviation for the grid topology with the RBE traffic pattern. The graphs show the trust value on the y-axis and the number of nodes on the x-axis.

The average trust values that can be observed in the scenario with the RBE traffic pattern are similar to the average trust values from the RTM traffic pattern (compare figure 5.10). The main difference is that the tested MOACO-based algorithms are a little bit clearer separated from each other. For $0\,\%$ of malicious nodes, the following ranking, from highest to lowest trust value, can be created: `ACSMOACO`, `SRMOACO`, `MMASMOACO` and `ASMOACO`. The DSR-based algorithm has again the lowest trust values.

When increasing the number of malicious nodes from $25\,\%$ to $75\,\%$ all average trust values decrease and get closer together (~ 0.2). The DSR-based algorithms has on average, as expected, always the lowest average trust values, when the number of malicious nodes is increased. The more ants are used by the MOACO-based algorithm, the better the obtained average trust value. This may be explained by the fact that more ants lead to a greater diversity of routes, which results then in finding more

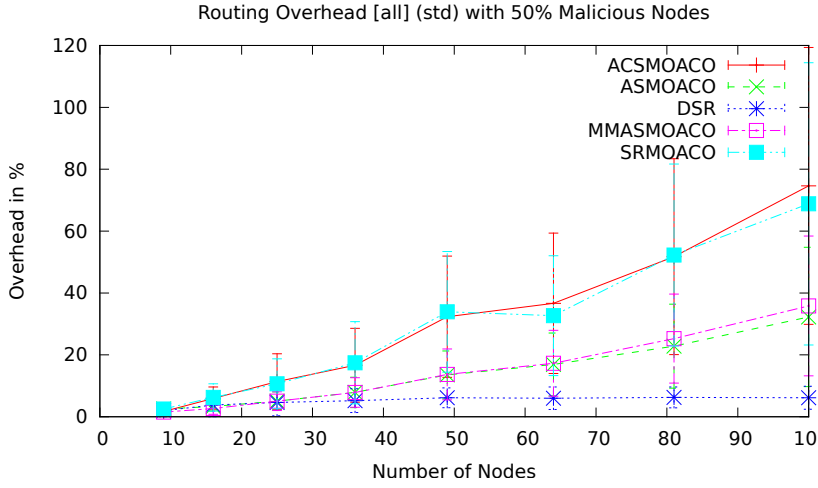


(a) 0% Malicious Nodes

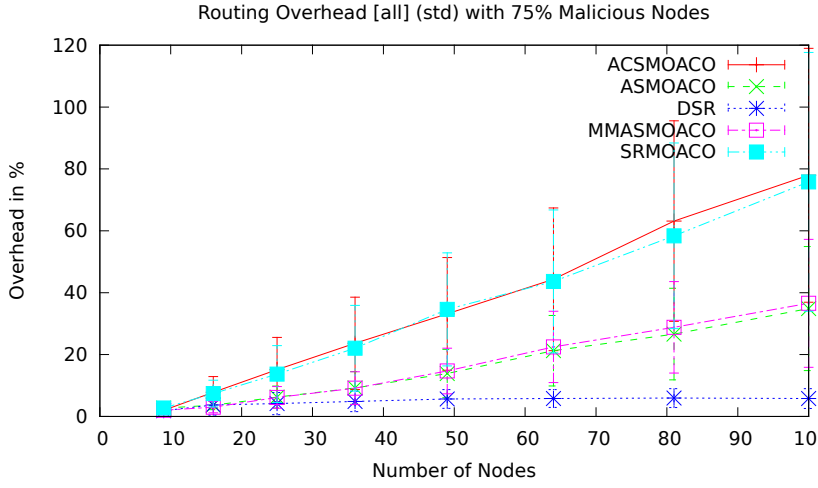


(b) 25% Malicious Nodes

Figure 5.15: Routing overhead with different percentages of malicious nodes for grid topology with RBE traffic pattern



(c) 50% Malicious Nodes

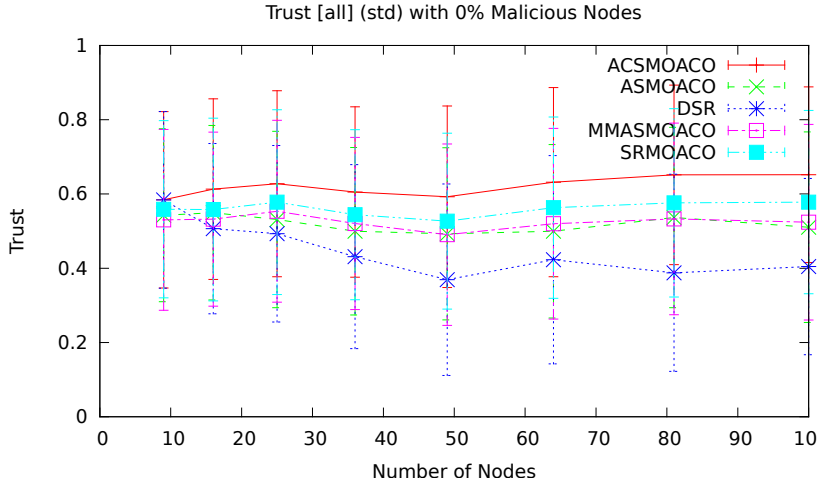


(d) 75% Malicious Nodes

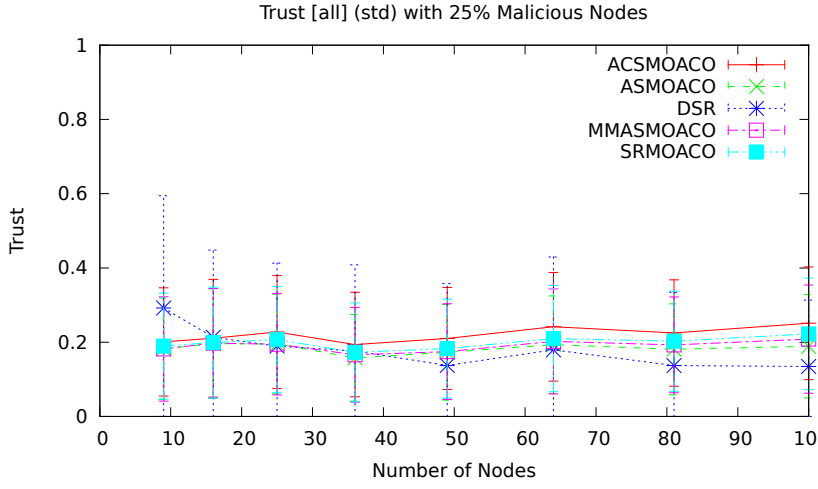
Figure 5.15: Routing overhead with different percentages of malicious nodes for grid topology with RBE traffic pattern

trusted routes.

As stated for the previous scenario, the difference between the MOACO-based algorithms and the DSR-based algorithms regarding the trust value can be explained by the fact that the MOACO-based algorithms take trust into account, while the DSR-based algorithm is not aware of this.

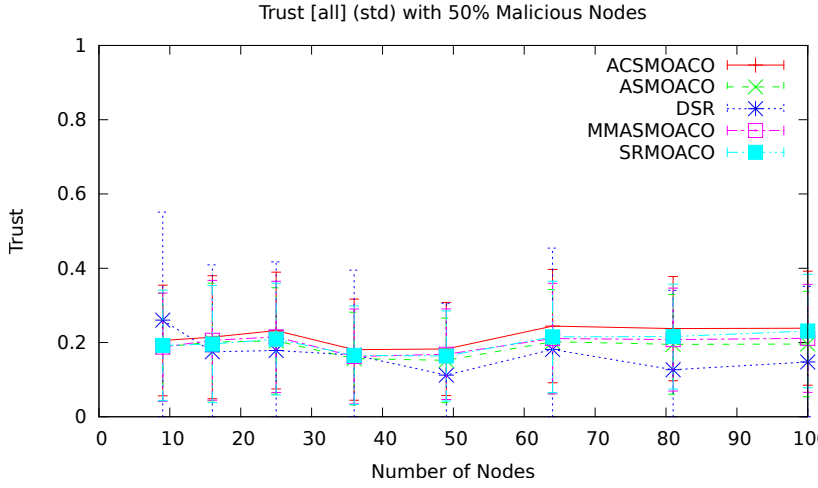


(a) 0% Malicious Nodes

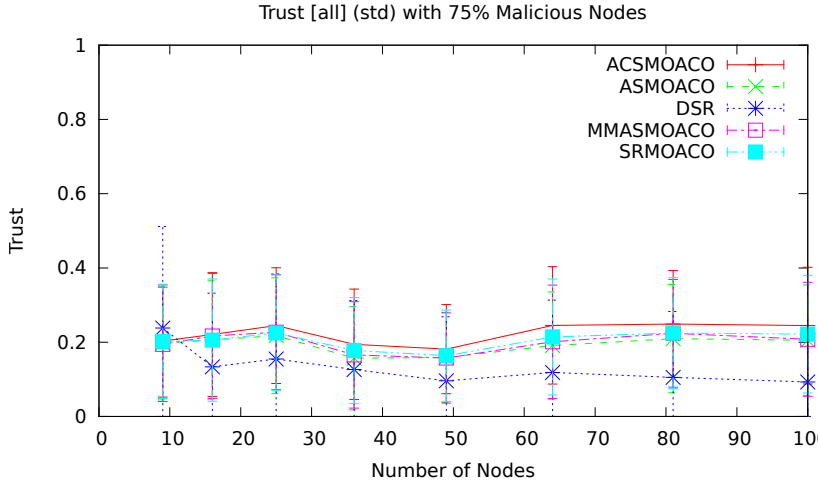


(b) 25% Malicious Nodes

Figure 5.16: Trust with different percentages of malicious nodes for grid topology with RBE traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.16: Trust with different percentages of malicious nodes for grid topology with RBE traffic pattern

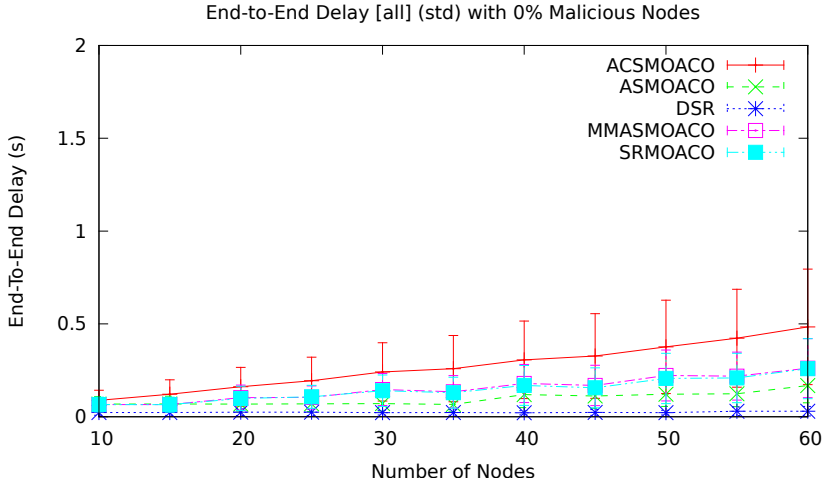
Random with Real-time Multimedia Data Traffic

In the following the experiments with a random topology and the RTM data traffic pattern are discussed:

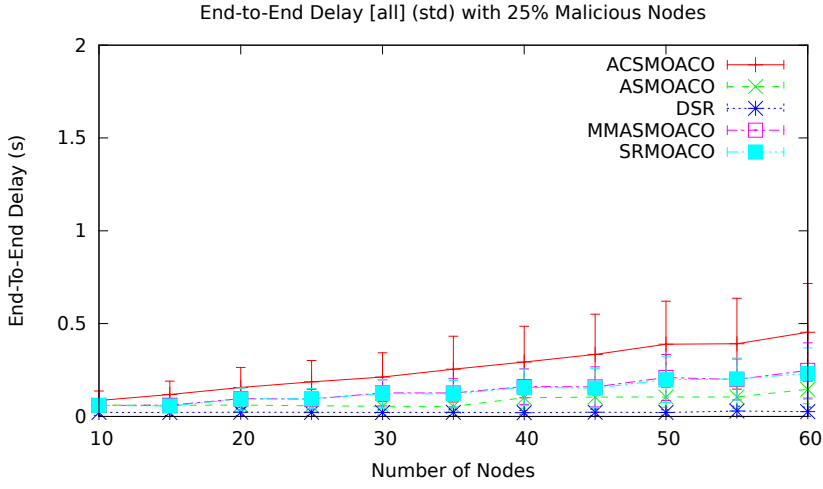
Application Packet End-to-End Delay: Figure 5.17 shows the average application packet end-to-end delay with standard deviation for the random topology with the RTM traffic pattern. The graphs show the end-to-end delay in in seconds on the y-axis and the number of nodes on the x-axis.

For the scenario with 0 % of malicious nodes (see figure 5.17a), it can be observed that the average application packet end-to-end delay for the DSR-based routing algorithm is smaller (< 0.1 s) than for the MOACO-based algorithms (between 0.1 s and 0.5 s) for all tested network sizes. The reason why the average application packet end-to-end delay of the DSR-based algorithm is much smaller in comparison to the MOACO-based algorithms is based on the different approaches in the route discovery phase: while in the DSR-based algorithm the first route that was found is used, the MOACO-based algorithms wait until all forward ants reached the destination or the expiration timer exceeds, causing the difference in the end-to-end delay. In comparison to the grid topology with the RTM traffic pattern, the average application packet end-to-end delay is for the random topology with the RTM traffic pattern slightly increased. This may be based on the fact that the nodes in the random topology are not as uniformly distributed as in the grid topology hence longer delays are more likely. Besides, slightly more ants are used in the MOACO algorithms as for the random topology causing some additional delay.

When the percentage of malicious nodes is increased to 25 % to 50 % to 75 %, a slightly lower average application packet end-to-end delay can be observed for the MOACO-based algorithms as for the scenario with 0 % of malicious nodes. This can be explained by the fact that the diversity of routes is reduced by malicious nodes dropping packets so that the ants converge faster to (near) optimal routes with lower end-to-end delay.

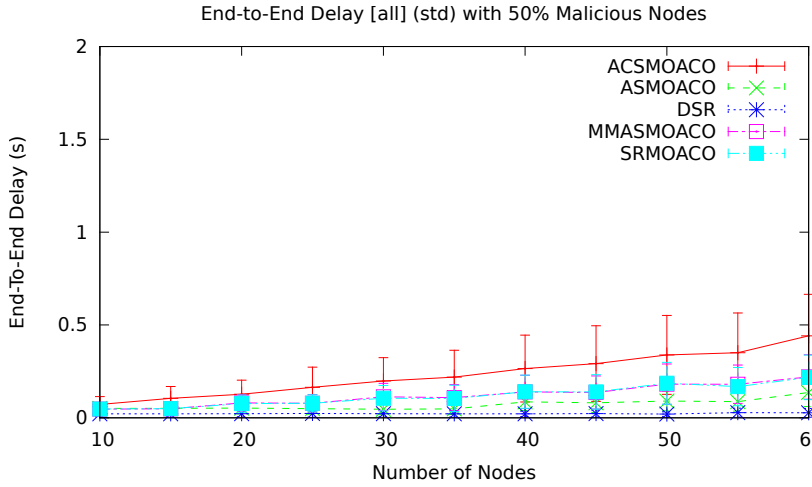


(a) 0% Malicious Nodes

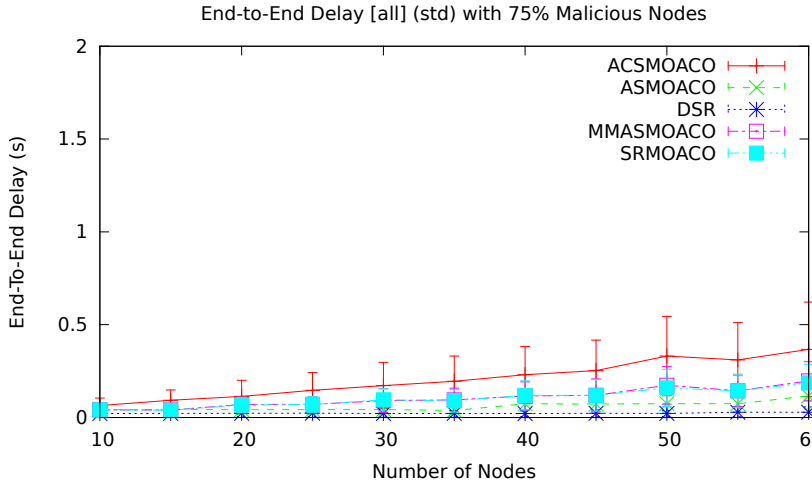


(b) 25% Malicious Nodes

Figure 5.17: Trust with different percentages of malicious nodes for grid topology with RBE traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.17: Application packet average end-to-end delay with different percentages of malicious nodes for random topology with RTM traffic pattern

Application Packet Delivery Ratio: Figure 5.18 shows the average application PDR with standard deviation for the random topology with the RTM traffic pattern. The graphs show the PDR on the y-axis and the number of nodes on the x-axis.

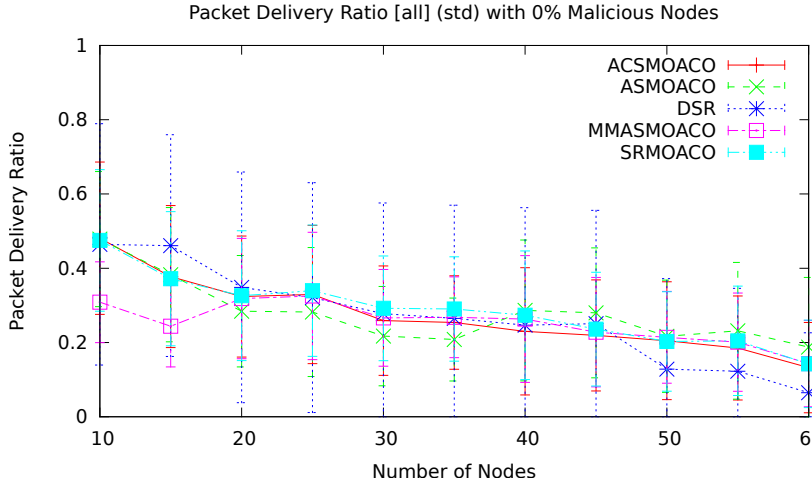
When there are no malicious nodes in the network (see figure 5.18a), it can be observed that starting from 20 nodes all MOACO-based algorithms as well as the DSR-based algorithms are close to each other, except `ASMOACO` which is slightly oscillating. When the network size is increased, the average application PDR is decreasing for all tested algorithms. While the DSR-based algorithm is strongly decreasing, the MOACO-based algorithms decrease more slowly so that for network size with more than 45 the MOACO-based algorithms outperform the DSR-based algorithm.

As expected, with the increasing of the percentage of malicious nodes to 25 % to 50 % to to 75 % it can be observed that for all tested routing algorithms the average application PDR is decreasing. The general trend of the routing algorithms is similar to the one observed with 0 % of malicious nodes, but here the PDR values are closer together so that no real difference between the algorithms can be observed, the more malicious nodes are in the network.

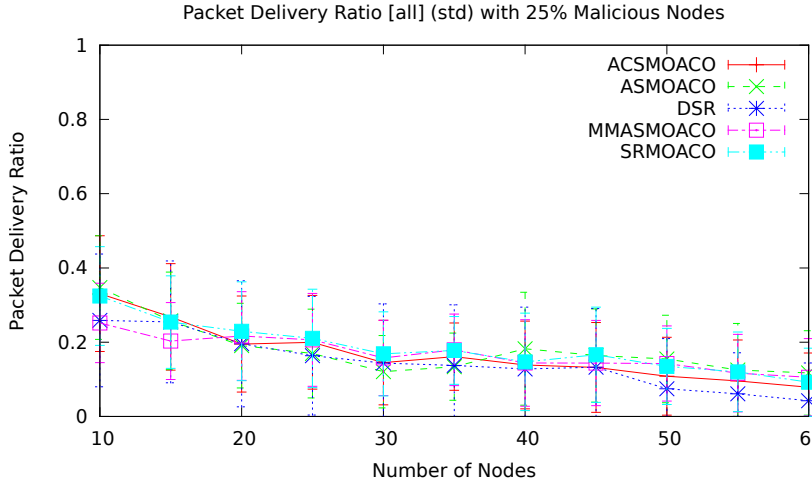
Number of Hops: Figure 5.19 shows the average number of hops with standard deviation required for the random topology with the RTM traffic pattern. The graphs show the number of hops on the y-axis and the number of nodes on the x-axis.

For the scenario with 0 % of malicious nodes in the network (see figure 5.19a) it can be observed that all MOACO-based algorithms require on average between 2 and 3.8 hops. In contrast, for the DSR-based algorithm less hops are required between 10 and 25 nodes, while a similar average number of hops as for the MOACO-based algorithms is required starting from 30 nodes.

When the number of malicious nodes is increased to 25 %, to 50 % and to 75 %, the average number of hops for the MOACO-based algorithms is decreased so that it is located around 2 hops. Although, the average number of hops is also decreased for the DSR-based algorithm, the de-

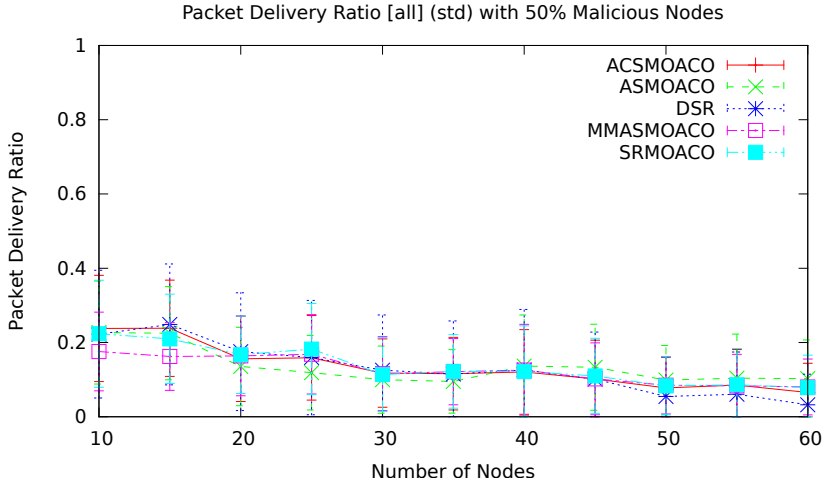


(a) 0% Malicious Nodes

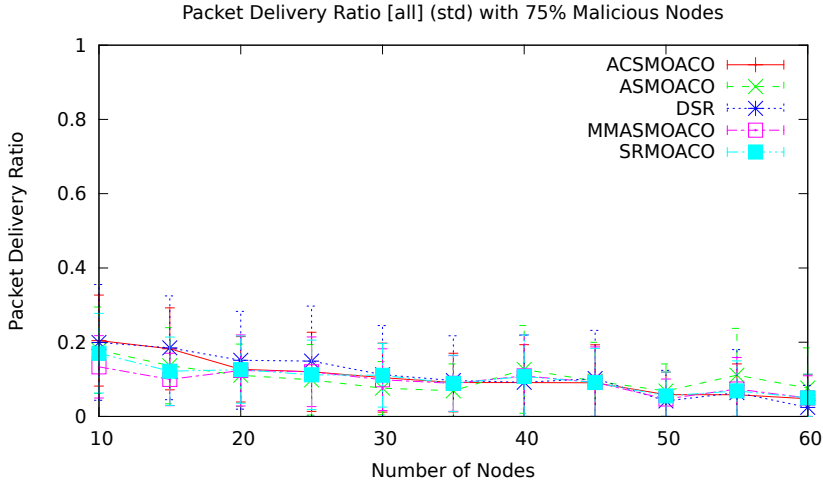


(b) 25% Malicious Nodes

Figure 5.18: Application PDR with different percentages of malicious nodes for random topology with RTM traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.18: Application PDR with different percentages of malicious nodes for random topology with RTM traffic pattern

crease is smaller: with already 25 % of malicious nodes in the network, the number of hops for the DSR-based algorithm, for more than 30 nodes, is always higher than the average number of hops for the MOACO-based algorithms. This basic tendency can be explained by the fact that the MOACO-based algorithms converge after a while to the (near) optimal route with a few hops, while the broadcasting mechanism in the discovery phase of the DSR-based algorithm leads to routes with multiple hops.

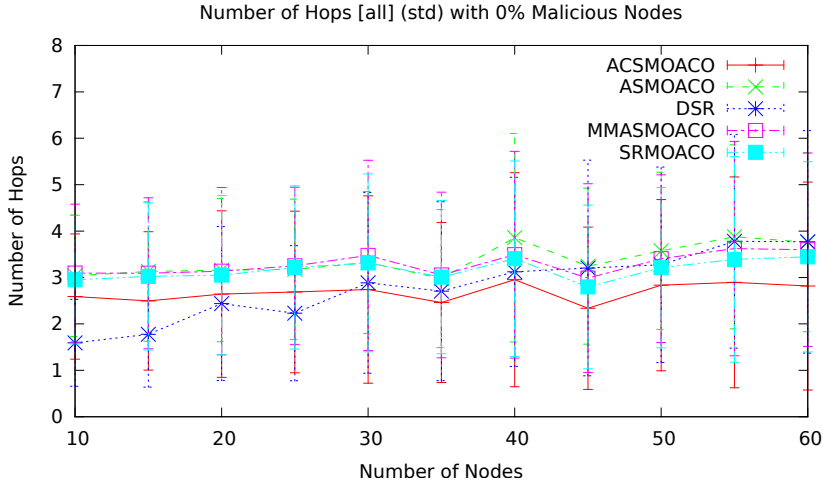
Residual Energy: Figure 5.20 shows the average residual energy with standard deviation for the random topology with the RTM traffic pattern. The graphs show the residual energy in % on the y-axis and the number of nodes on the x-axis.

The average residual energy in the random scenario is similar to the one observed in the grid scenario with the RTM traffic pattern, i.e. the MOACO-based routing algorithms have a slightly higher energy consumption than the DSR-based algorithm. When the number of malicious nodes is increased in the network, the average residual energy stays almost as it was.

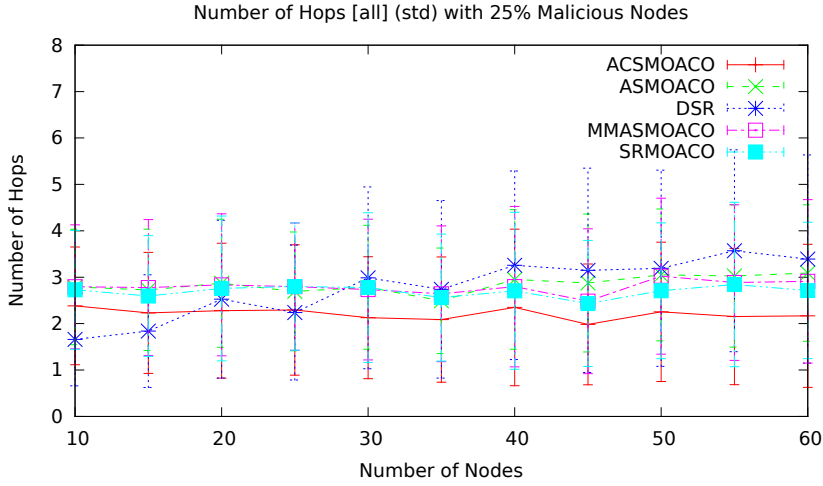
Routing Overhead: Figure 5.21 shows the average routing overhead for the random topology with the RTM traffic pattern. The graphs show the routing overhead on the y-axis and the number of nodes on the x-axis.

When there are no malicious nodes in the network (see figure 5.21a), it can be observed that `ASMOACO` has the lowest average routing overhead, followed by the DSR-based routing algorithm. Slightly more average routing overhead have the MOACO-based algorithms `SRMOACO` and `MMASMOACO`, which both increase up to $\sim$ 17 % in worst case. Significantly more overhead can be observed for `ACSMOACO`, increasing more strongly up to $\sim$ 32 %. The stronger increase of `ACSMOACO` can be explained by the fact that it uses the largest number of ants in the scenario.

By increasing the percentage of malicious nodes in the network from 25 %, to 50 %, to 75 %, it can be observed that the average routing overhead for the DSR-based algorithm stays almost constant on the same level, while the average application routing overhead for the MOACO-based

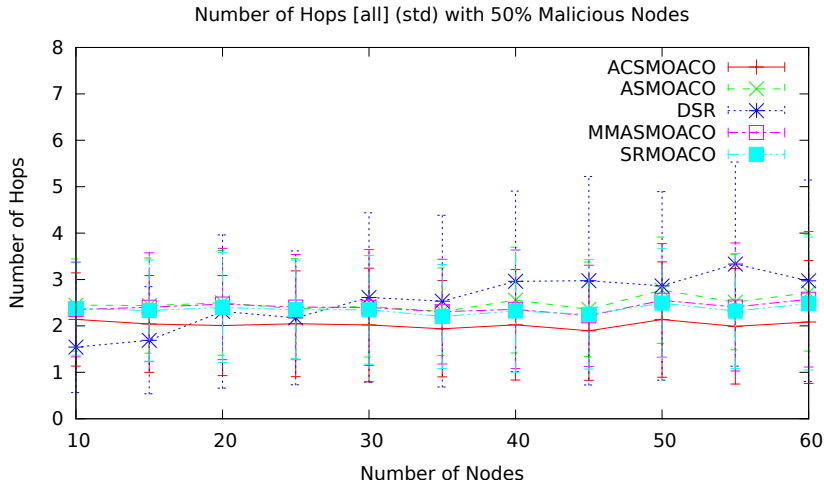


(a) 0% Malicious Nodes

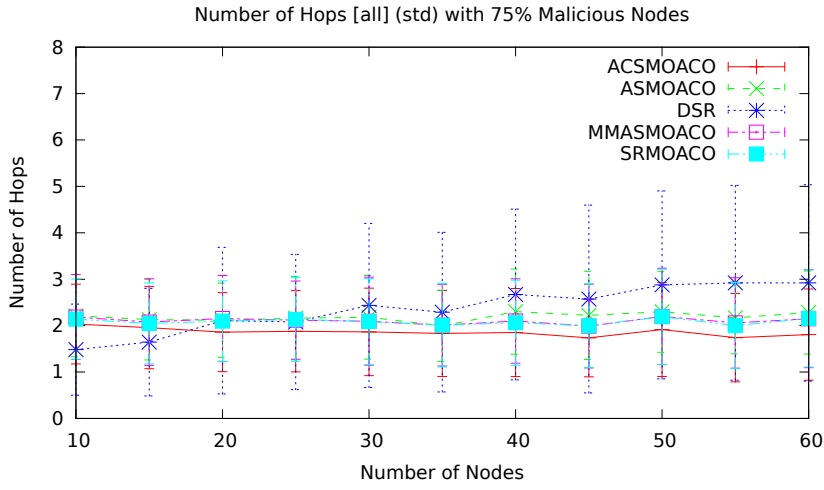


(b) 25% Malicious Nodes

Figure 5.19: Number of hops with different percentages of malicious nodes for random topology with RTM traffic pattern

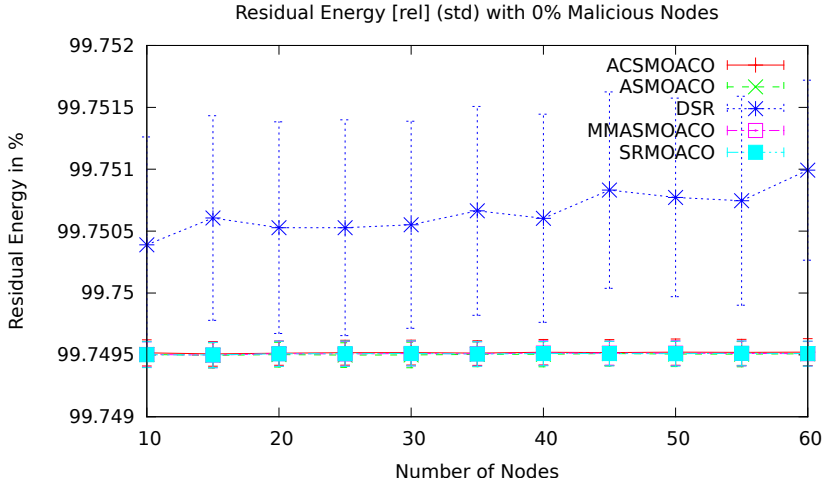


(c) 50% Malicious Nodes

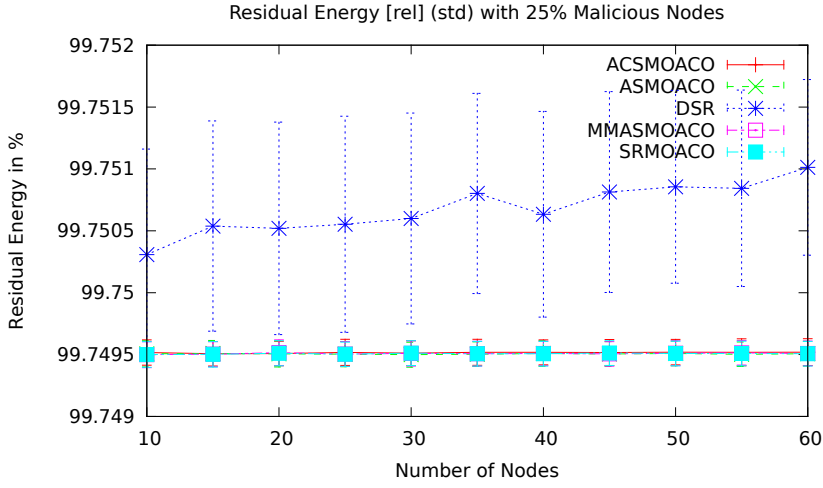


(d) 75% Malicious Nodes

Figure 5.19: Number of hops with different percentages of malicious nodes for random topology with RTM traffic pattern

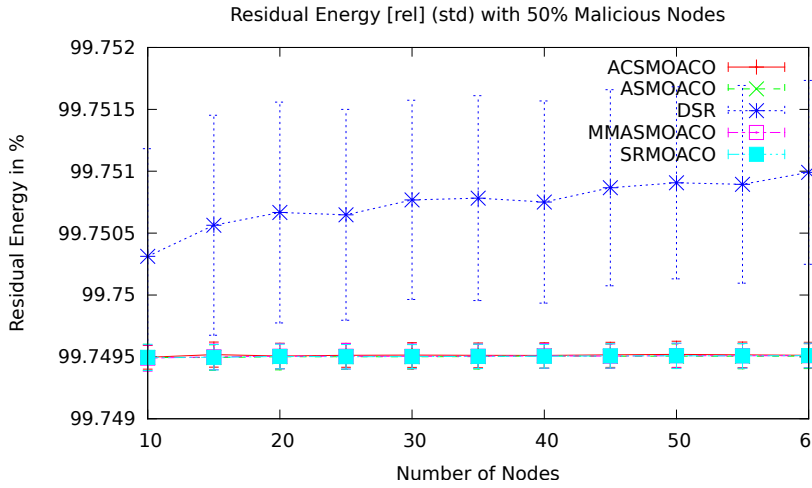


(a) 0% Malicious Nodes

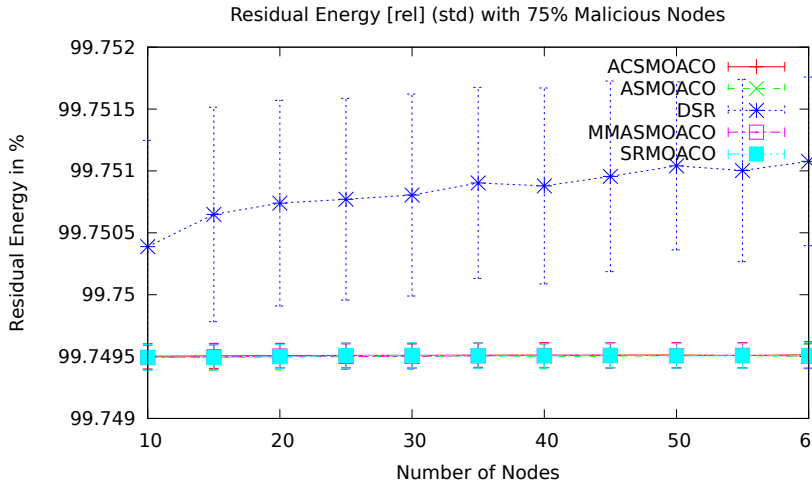


(b) 25% Malicious Nodes

Figure 5.20: Residual energy with different percentages of malicious nodes for random topology with RTM traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.20: Residual energy with different percentages of malicious nodes for random topology with RTM traffic pattern

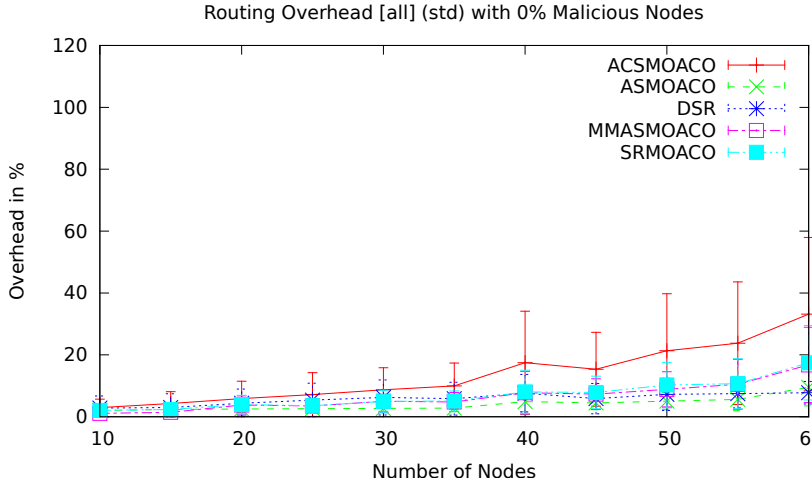
algorithm is increased. The strongest rise can be observed for ACSMOACO, which reaches $\sim 53\,\%$ in worst case, whereas SRMOACO and MMASMOACO go up to $\sim 25\,\%$ and ASMOACO up to $10\,\%$ at worst (see figure 5.21d).

The increase of the average routing overhead of the MOACO-based algorithms in the case of an increase of malicious nodes can be explained by the fact that the more ants are dropped by malicious nodes, the worse becomes the ratio between routing and application packets. Due to the fact that the DSR-based algorithm in the tested scenarios only starts the route discovery once, the influence of dropped packets in the routing packet to application packet ratio is not as crucial as for the MOACO-based algorithms.

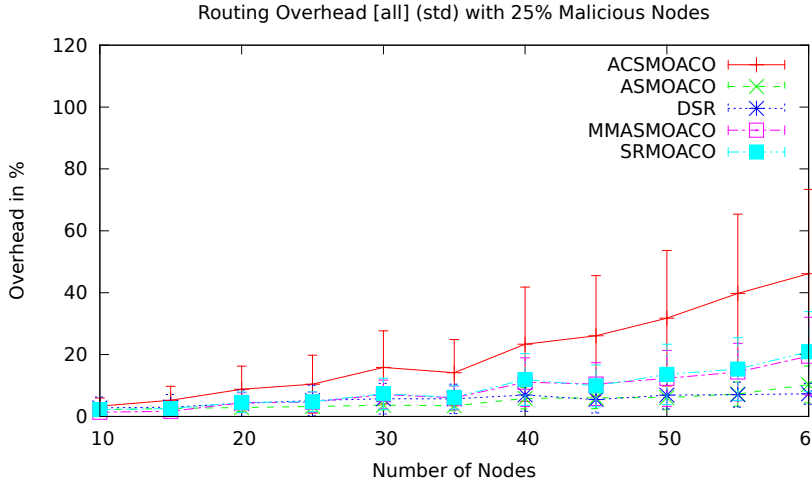
Trust: Figure 5.22 shows the average trust value with standard deviation for the random topology with the RTM traffic pattern. The graphs show the trust value on the y-axis and the number of nodes on the x-axis.

In the case of $0\,\%$ of malicious nodes in the network (see figure 5.22a), it can be observed that the MOACO-based routing algorithms outperforms the DSR-based routing algorithm for all tested scenarios in terms of the average trust value. This is as expected because the DSR-based algorithms does not consider any sort of trust in its decision making process, while for the MOACO-based algorithms this is one of the optimisation critera. The best MOACO-based algorithms is ACSMOACO with an average trust value around 0.7, followed by SRMOACO, MMASMOACO and ASMOACO with a trust value between ~ 0.6 and ~ 0.45). The DSR-based algorithm performs significantly weaker (around 0.4 on average). An interesting fact is that the number of nodes in the network does not seem to have a large influence on the average trust value.

When the percentage of malicious nodes is increased up to $25\,\%$, on average, the trust value of all approaches is halved. Still ACSMOACO outperforms all other algorithms for most network sizes, followed by the other MOACO-based algorithms and then, the DSR-based algorithm, which is only slightly weaker. For $50\,\%$ and $75\,\%$ of malicious nodes, the MOACO-based algorithms performs similar to the scenario with $25\,\%$ of malicious nodes, while the average trust value for the DSR-based algorithm is de-

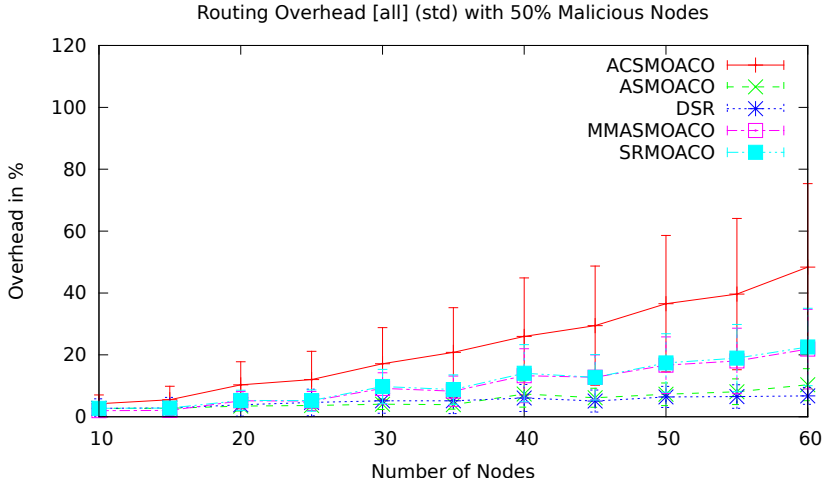


(a) 0% Malicious Nodes

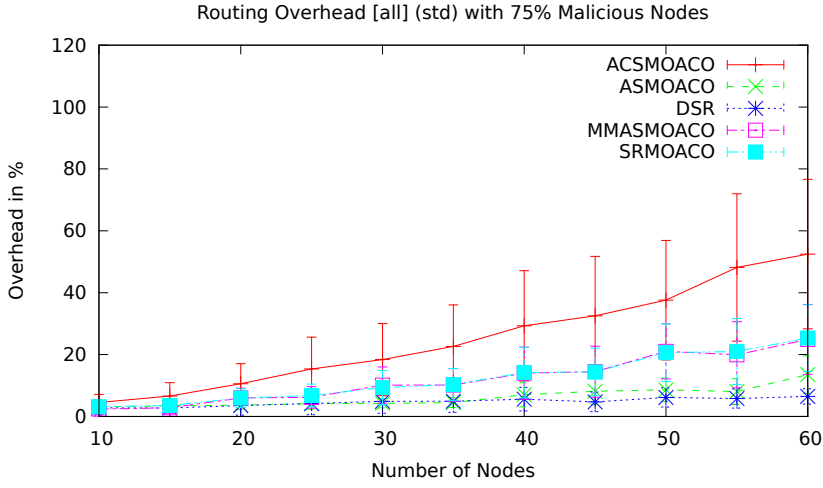


(b) 25% Malicious Nodes

Figure 5.21: Routing overhead with different percentages of malicious nodes for random topology with RTM traffic pattern



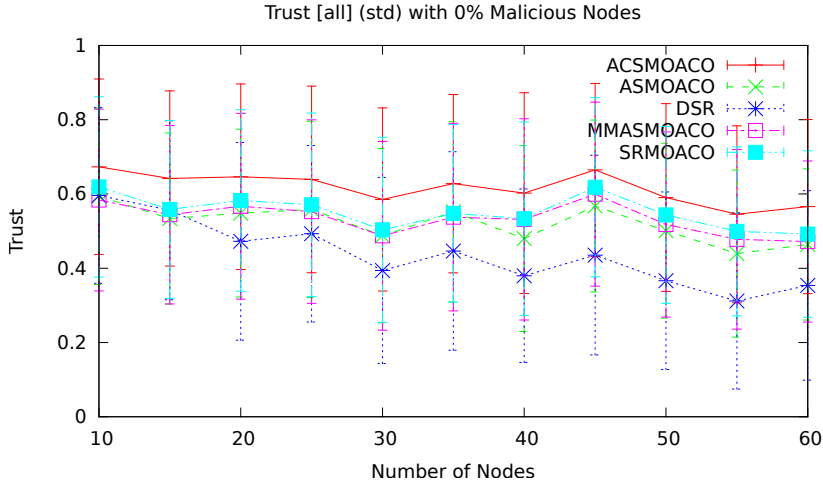
(c) 50% Malicious Nodes



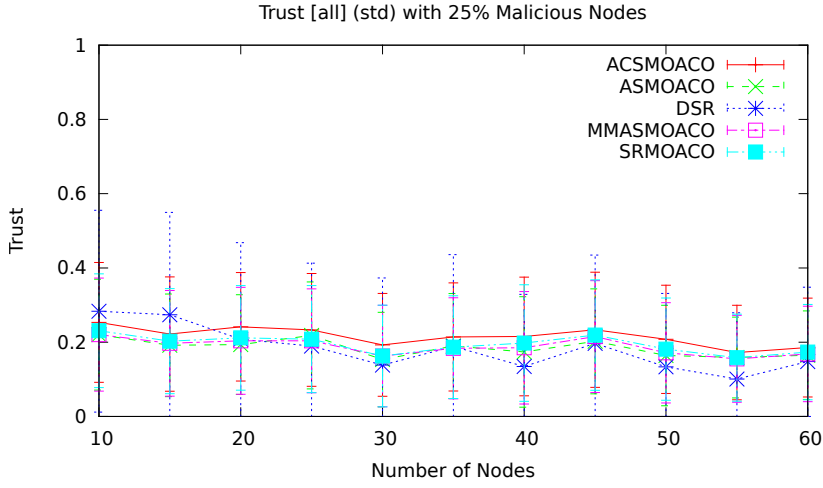
(d) 75% Malicious Nodes

Figure 5.21: Routing overhead with different percentages of malicious nodes for random topology with RTM traffic pattern

creasing dramatically so that a significant gap between the average trust values can be observed.

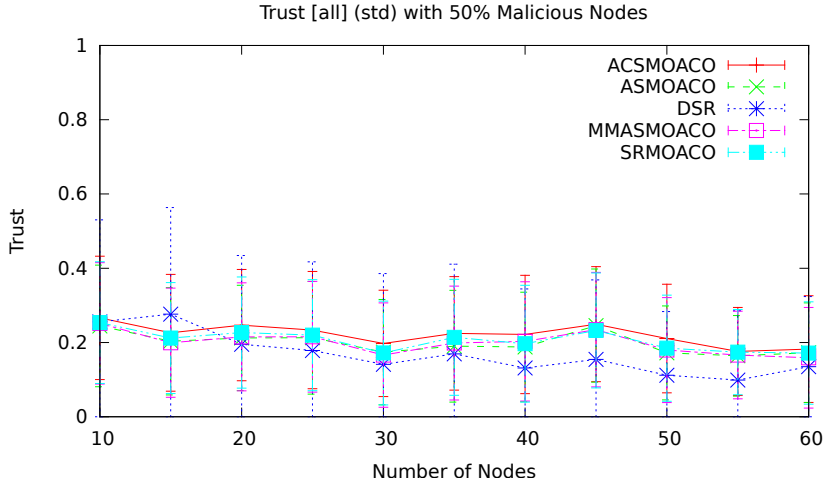


(a) 0% Malicious Nodes

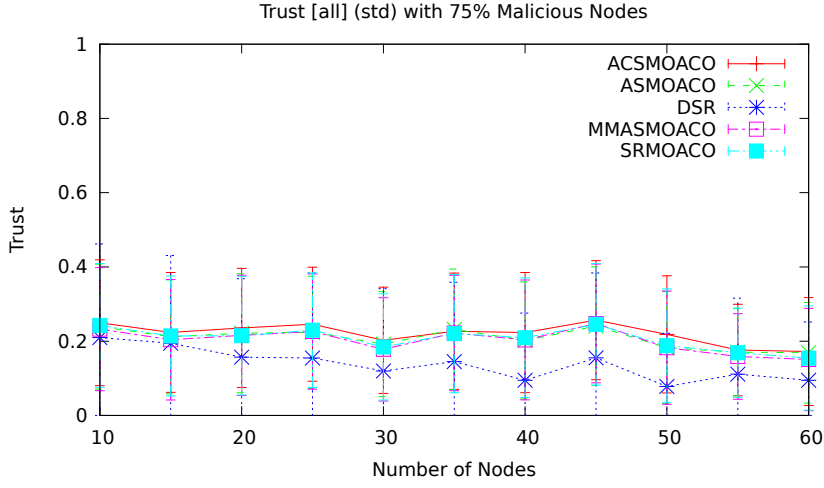


(b) 25% Malicious Nodes

Figure 5.22: Trust with different percentages of malicious nodes for random topology with RTM traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.22: Trust with different percentages of malicious nodes for random topology with RTM traffic pattern

Random with Reliable Best-Effort Data Traffic

In the following the experiments with a random topology and the RBE data traffic pattern are discussed:

Application Packet End-to-End Delay: Figure 5.23 shows the average application packet end-to-end delay with standard deviation for the random topology with the RBE traffic pattern. The graphs show the end-to-end delay in seconds on the y-axis and the number of nodes on the x-axis.

When there are no malicious nodes in the network (see figure 5.24a), the MOACO-based algorithms have on average a higher application packet end-to-end delay (up to 1.1 s than the DSR-based algorithm ($\leq$ 0.1 s). When not considering the outlier of `ASMOACO` for 100 nodes, all MOACO-based algorithm stay below 0.5 s. The difference between the DSR-based and the MOACO-based algorithms can be explained by the fact that the MOACO-based algorithms are sending regularly new ants for the discovery of new routes or the improvement of existing routes, while the DSR-based algorithms sticks with the first route that was found. On average, `ACSMOACO` and `SRMOACO` have the highest application packet end-to-end delays around 0.5 s. A significantly lower end-to-end delay can be observed for `MMASMOACO`, which requires in worst case 0.25 s. `ASMOACO` slightly oscillates between `MMASMOACO` and the other two MOACO-based algorithms, except for 100 nodes where the application packet end-to-end delay gets much higher than for the other tested algorithms. The slightly longer delay for `ACSMOACO` and `SRMOACO` can be mainly explained by the fact that both use two times more ants than the other two MOACO-based algorithms.

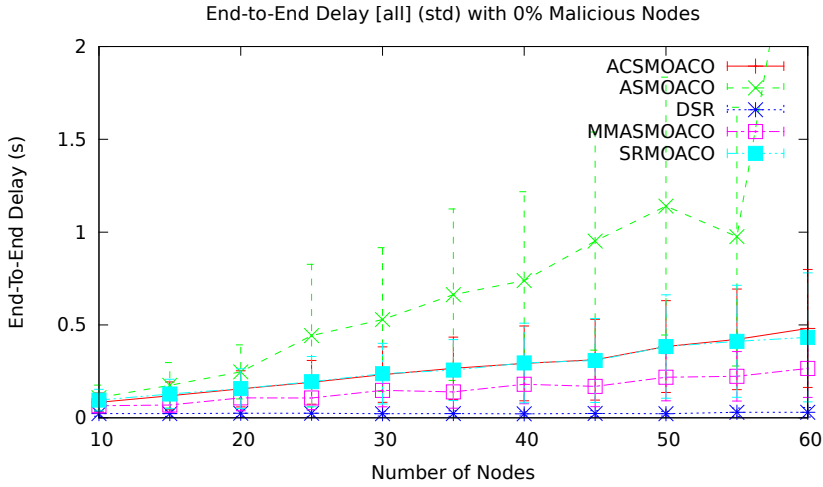
When the percentage of malicious nodes is increased to 25 % the average application packet end-to-end delay of the tested algorithms does not change much, except for `ASMOACO`, which has in this case a lower end-to-end delay than all other MOACO-based algorithms at a similar level as `MMASMOACO` (see figure 5.23b). For 50 % and 75 % of malicious nodes, the average application packet end-to-end delay of all MOACO-based algorithms is slightly reduced. (see figure 5.23c and 5.23d). The slightly

decreasing application packet average end-to-end delay for the MOACO-based algorithms can be explained by the fact that the ants converge faster to (near) optimal routes because the other routes cannot be used due to malicious nodes. Furthermore, the dropping of packets leads to less data traffic in the network and thus, less packet collisions.

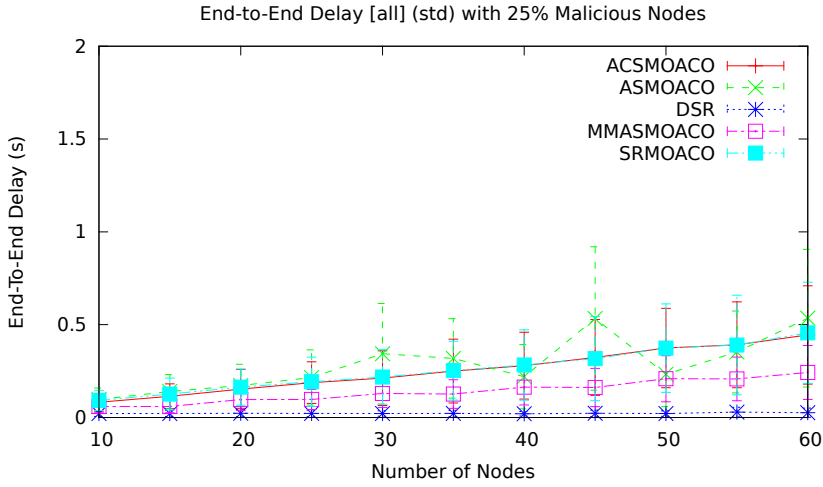
Application Packet Delivery Ratio: Figure 5.24 shows the average of the application PDR for the random topology with the RBE traffic pattern. The graphs show the PDR on the y-axis and the number of nodes on the x-axis.

When there are 0 % of malicious nodes in the network all tested routing algorithms have a similar average application PDR, except `ASMOACO`, which performs significantly worse (see figure 5.24a). With an increase of the nodes in the network, the average PDR is decreasing. The maximum average PDR is reached by `SRMOACO` ($\sim$ 0.5), while the minimum is reached by `ASMOACO` ($\sim$ 0.05). It can be observed that in comparison to the MOACO-based algorithms (except `ASMOACO`) the DSR-based algorithm is strongly decreasing starting from 50 on.

With the increase of the percentage of malicious nodes to 25 %, 50 % and 75 % the average PDR is decreased for all tested algorithm so that it varies in the range of $\sim$ 0.02 and $\sim$ 0.38 (see figures 5.24b - 5.24d). For 75 % of malicious nodes, all algorithms have an average application PDR below 0.2. The measurements of all tested algorithms are coming closer together and the standard deviation is getting smaller so that for almost all network sizes the algorithms can hardly be distinguished. Consequently, it can be assumed that similar routes, namely the only routes that are working without malicious nodes, are used by all algorithms, resulting in a similar average application PDR. Except `ASMOACO`, which performs worst for all scenarios, the MOACO-based algorithms perform on a similar level than the DSR-based algorithm and starting from 50 nodes in the network, the MOACO-based algorithms even outperform the DSR-based algorithm so that it would be interesting to test scenarios with more nodes in the future to see whether this tendency is ongoing.

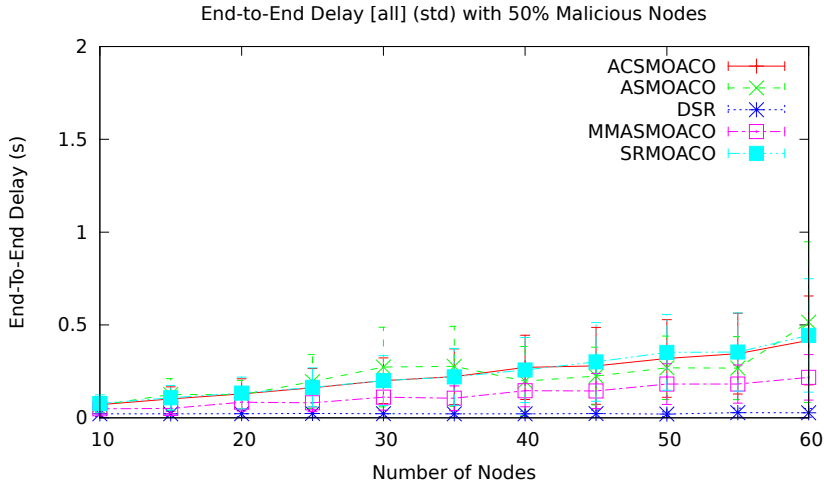


(a) 0% Malicious Nodes

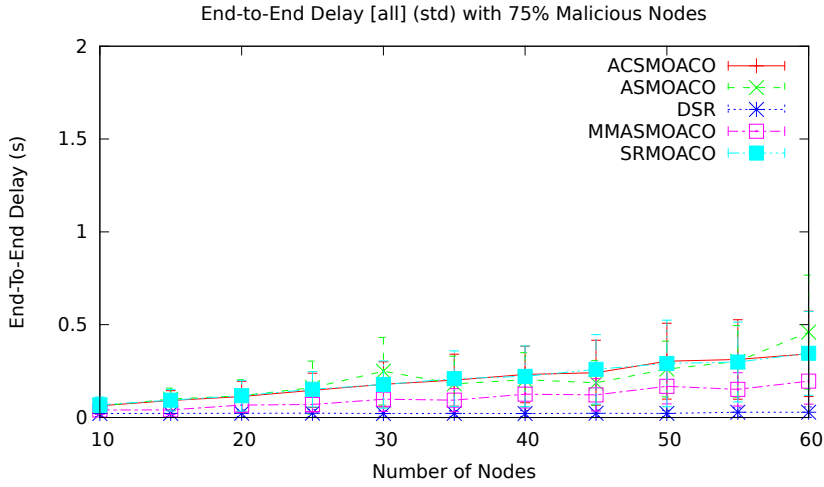


(b) 25% Malicious Nodes

Figure 5.23: Application packet average end-to-end delay with different percentages of malicious nodes for random topology with RBE traffic pattern

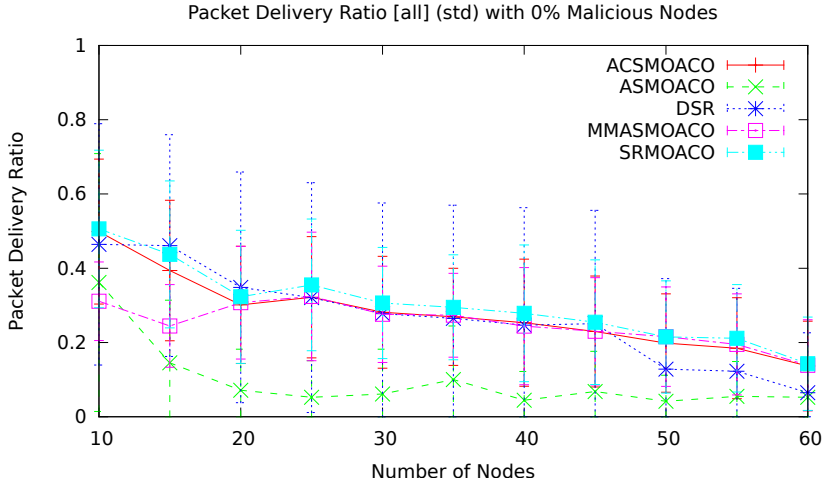


(c) 50% Malicious Nodes

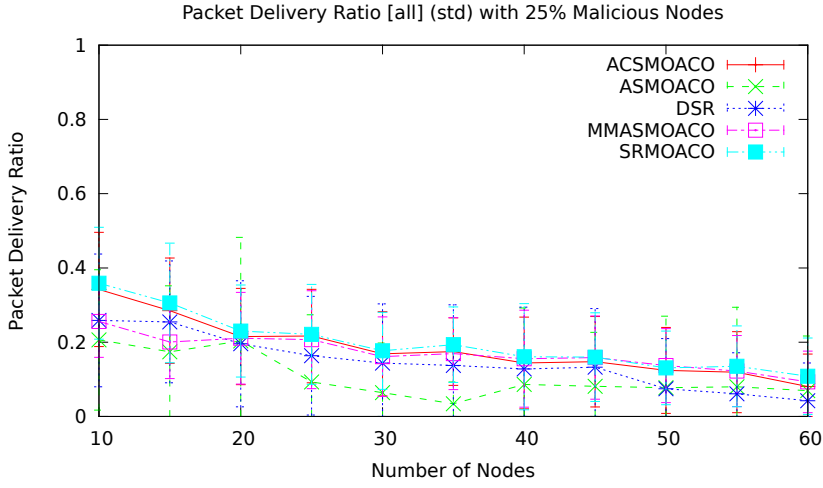


(d) 75% Malicious Nodes

Figure 5.23: Application packet average end-to-end delay with different percentages of malicious nodes for random topology with RBE traffic pattern

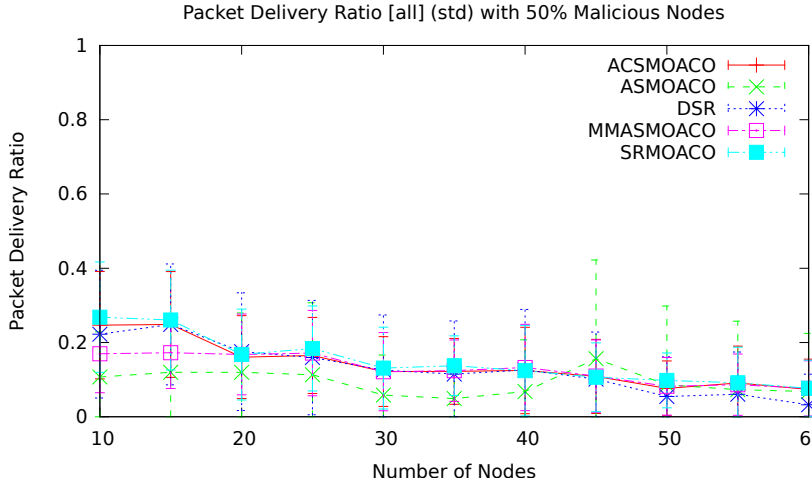


(a) 0% Malicious Nodes

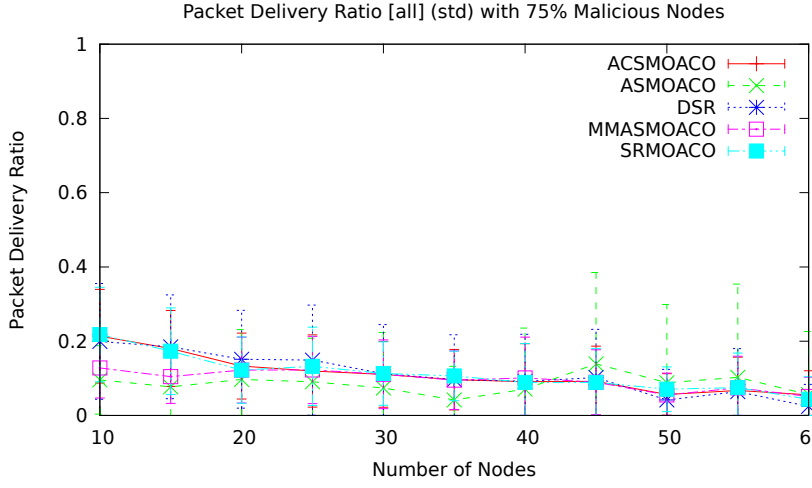


(b) 25% Malicious Nodes

Figure 5.24: Application PDR with different percentages of malicious nodes for random topology with RBE traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.24: Application PDR with different percentages of malicious nodes for random topology with RBE traffic pattern

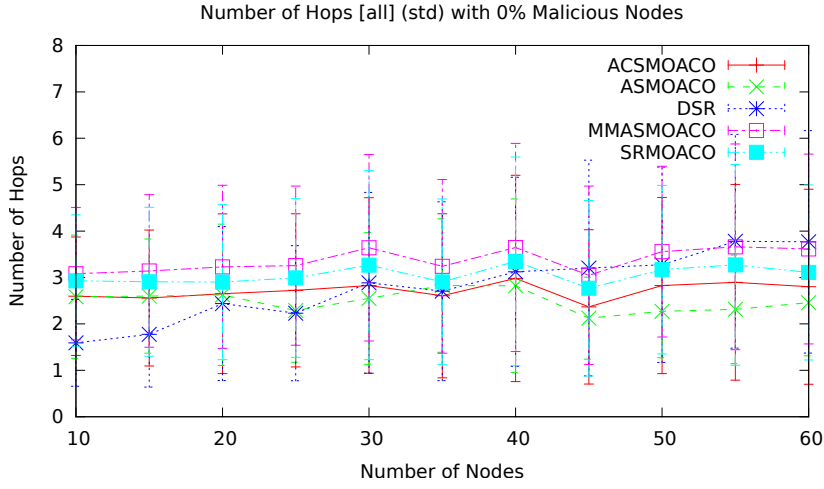
Number of Hops: Figure 5.25 shows the average number of hops with standard deviation required for the random topology with the RBE traffic pattern. The graphs show the number of hops on the y-axis and the number of nodes on the x-axis.

When there are no malicious nodes in the network (see figure 5.25a), the number of hops for the MOACO-based algorithms varies between ~ 2.25 and ~ 3.5. Although, the average number of hops for the MOACO-based algorithms are close together, it can be stated that on average the `MMASMOACO` requires the most hops, while `ASMOACO` and `ACSMOACO` require the least hops. While for networks with 10 or 15 nodes, the number of hops for the DSR-based algorithm is lower than for the MOACO-based algorithms, for more than 25 nodes the average number of hops for the DSR-based algorithm is in a similar range as for the MOACO-based algorithms.

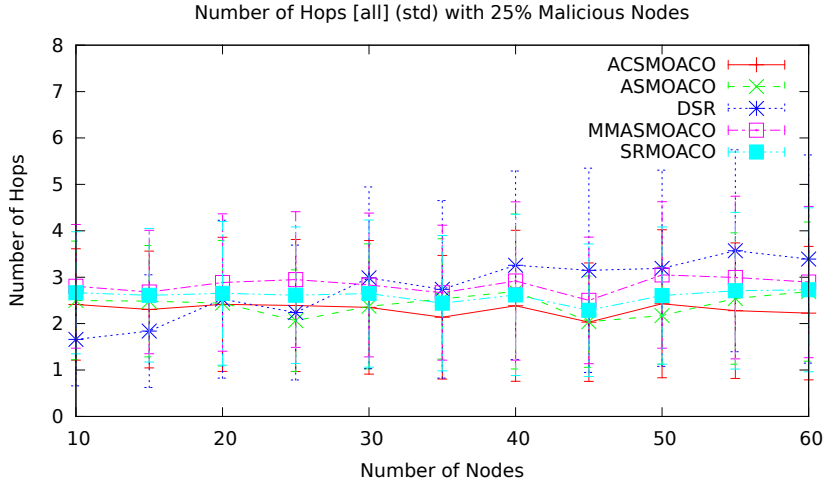
When the percentage of malicious nodes is increased up to 25 %, to 50 % and to 75 % of malicious nodes, it can be observed that the average number of hops for the MOACO-based routing algorithms is slightly decreasing (down to 2.2 hops) (see figures 5.25b, 5.25c and 5.25d), while average number of hops for the DSR-based algorithm is rather constant, resulting in a larger gap between the DSR-based algorithm and the MOACO-based algorithms. This can be explained by the fact that the MOACO-based algorithm are biased by the pheromone to stick to the (near) optimal paths found with less hops, while the DSR-based routing algorithm will always have unnecessary routes with more hops due to the broadcasting mechanism used in the route discovery phase. All in all, the average number of hops is similar to the ones observed in the previous scenarios.

Residual Energy: Figure 5.26 shows the average residual energy with standard deviation for the random topology with the RBE traffic pattern. The graphs show the residual energy in % on the y-axis and the number of nodes on the x-axis.

No big difference can be observed for the average residual energy in comparison to the previously evaluated scenarios: while the DSR-based routing algorithm has slightly more residual energy with a greater standard

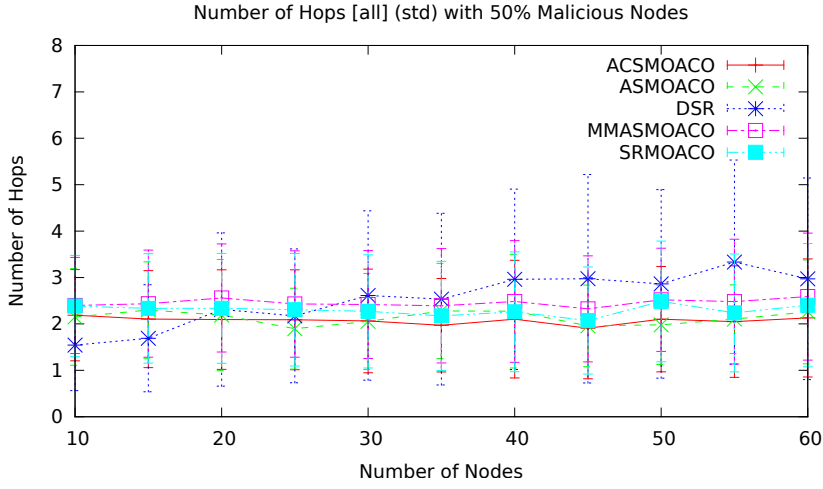


(a) 0% Malicious Nodes

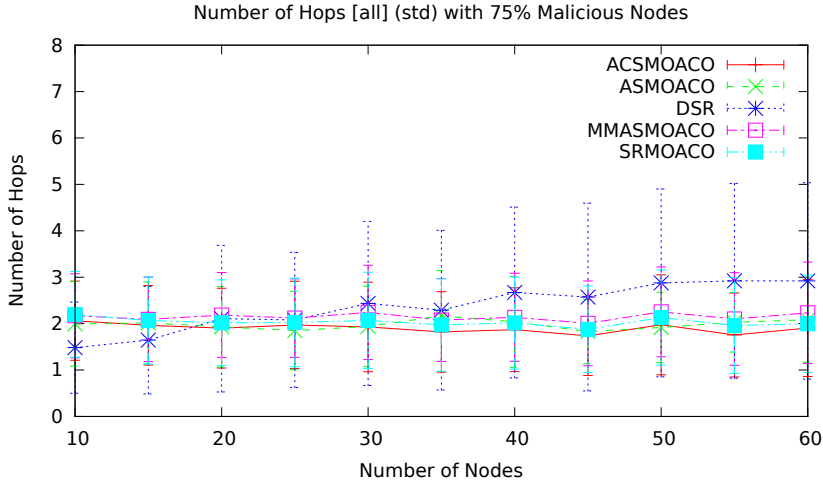


(b) 25% Malicious Nodes

Figure 5.25: Number of hops with different percentages of malicious nodes for random topology with RBE traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.25: Number of hops with different percentages of malicious nodes for random topology with RBE traffic pattern

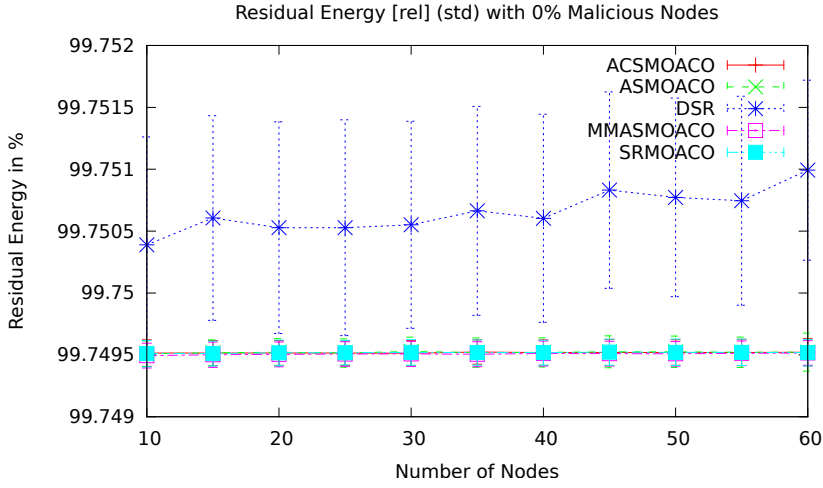
deviation, the MOACO-based routing algorithms have a little less residual energy with a smaller standard deviation. The percentage of malicious nodes does not seem to influence the residual energy much within this short runtime of the scenario. For a better analysis of the residual energy, further simulations with a longer runtime need to be conducted.

Routing Overhead: Figure 5.27 shows the routing overhead with standard deviation for the random topology with the RBE traffic pattern. The graphs show the routing overhead in % on the y-axis and the number of nodes on the x-axis.

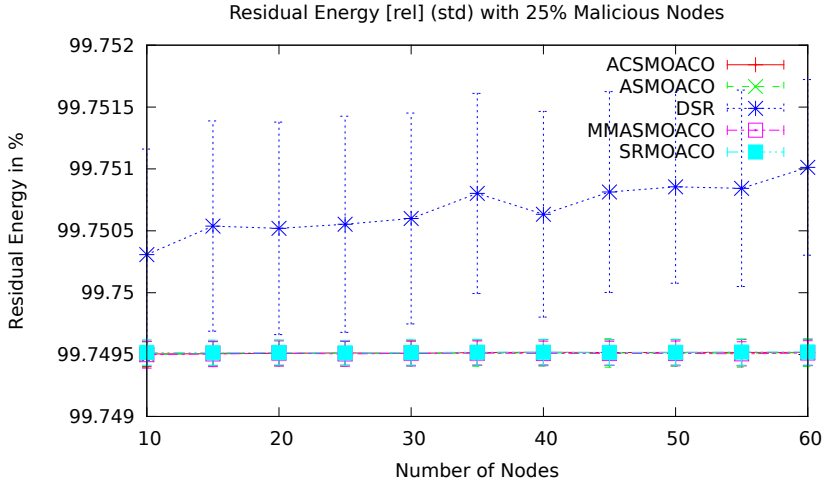
When there are no malicious nodes in the network (see figure 5.27a), the average routing overhead of the MOACO-based algorithms is slightly increased for all MOACO-based algorithms, except `ACSMOACO`, compared to the random scenario with the RTM traffic pattern. `ACSMOACO`, `ASMOACO` and `SRMOACO` have very similar overhead ($\sim$ 38 % at worst). In contrast, `MMASMOACO` has only a little bit more routing overhead ($\sim$ 16 % in worst case) so that for scenarios with more than 50 nodes it has slightly more overhead than the DSR-based routing algorithm, which stays rather constant around 8 %.

When the number of malicious nodes in the network is increased to 25 % to 50 % to 75 %, (see figures 5.27b, 5.27c and 5.27d), the average routing overhead of all MOACO-based algorithms is increased. The worst average routing overhead can be observed for `SRMOACO` ($\sim$ 55 %) for the scenario with 75 % of malicious nodes – a similar overhead can be observed for `ACSMOACO` ($\sim$ 50 %). Slightly less routing overhead has `ASMOACO` (up to 30 %) and `MMASMOACO` (up to 22 %) for 60 nodes. The overhead for the DSR-based algorithm stays constantly low no matter how many malicious nodes are in the network.

The additional overhead for the MOACO-based algorithms can be explained by the fact that for almost all MOACO-based algorithms, the number of ants was doubled, resulting in an additional overhead. An interesting observation can be made for `MMASMOACO`, which has an higher overhead as with the RTM traffic pattern on the same random topology, though the number of ants is the same: it is likely that the larger overhead

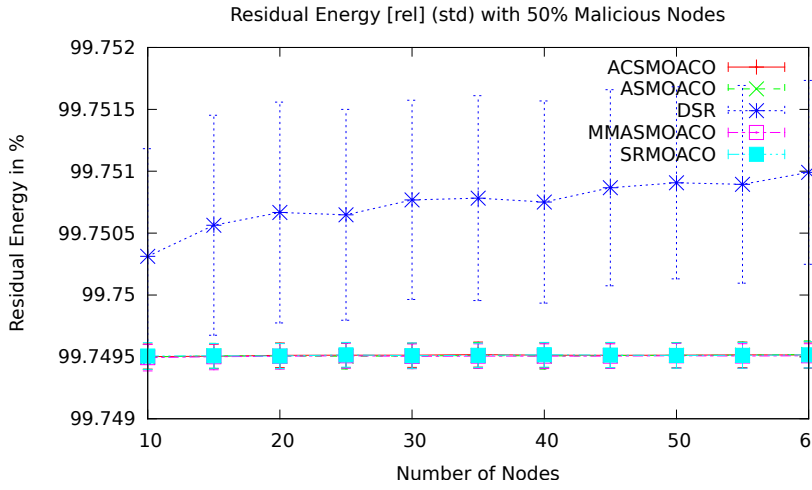


(a) 0% Malicious Nodes

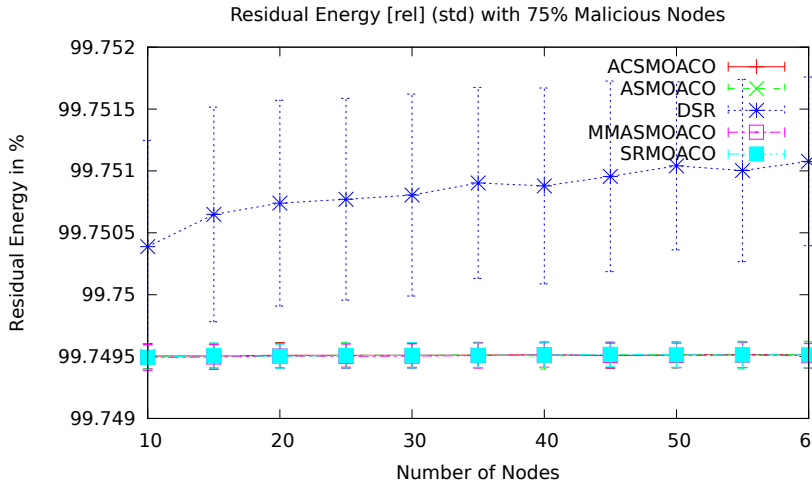


(b) 25% Malicious Nodes

Figure 5.26: Residual energy with different percentages of malicious nodes for random topology with RBE traffic pattern



(c) 50% Malicious Nodes



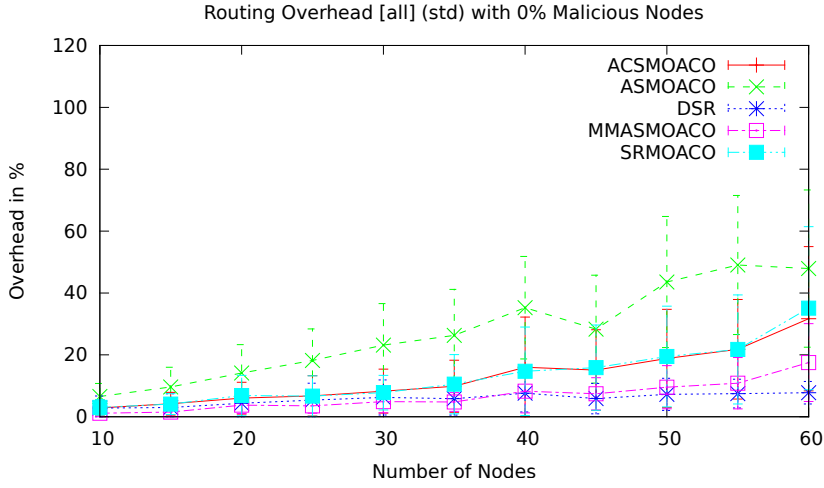
(d) 75% Malicious Nodes

Figure 5.26: Residual energy with different percentages of malicious nodes for random topology with RBE traffic pattern

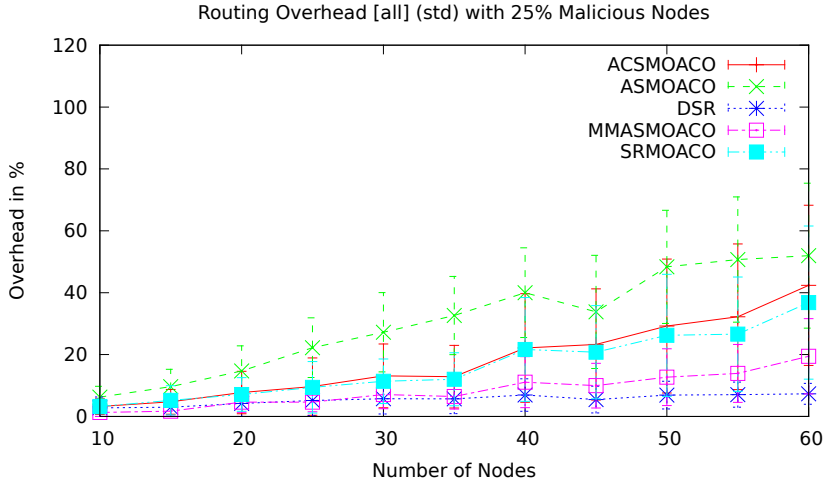
is resulting from the tuned evaporation rate, pheromone boundaries, but also the randomness of the scenario.

Trust: Figure 5.28 shows the trust value with standard deviation for the random topology with the RBE traffic pattern. The graphs show the trust value on the y-axis and the number of nodes on the x-axis.

The average trust values that can be observed for the RBE traffic pattern for the different scenarios with different percentages of malicious nodes are very similar to those observed for the RTM traffic pattern on the same topology (see figures 5.28 and 5.22). The only obvious difference is the volatile behaviour of `ASMOACO`, which seems to be strongly influenced by the randomness of the scenario. The large changes can be explained by the fact that `ASMOACO` is the only MOACO-based algorithms that sends all forward ants back as backward ants, all contributing to the average trust value. Consequently, high trust values are amplified if trustworthy routes are found by many backward ants, while , inversely, low trust values are amplified by untrustworthy routes. This amplification seems to be particularly the case for the scenarios with 35 and 50 nodes.

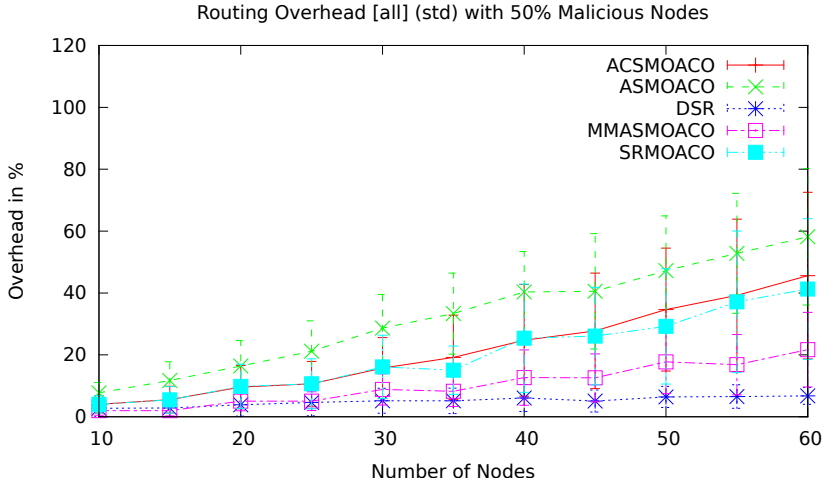


(a) 0% Malicious Nodes

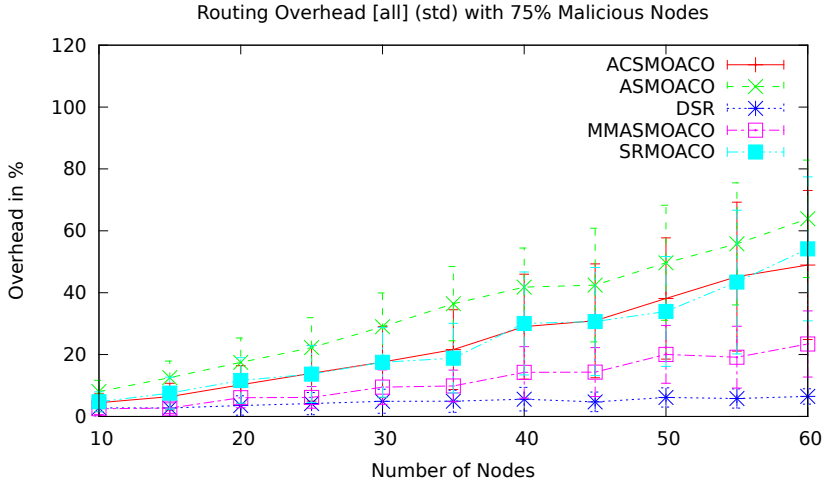


(b) 25% Malicious Nodes

Figure 5.27: Routing overhead with different percentages of malicious nodes for random topology with RBE traffic pattern

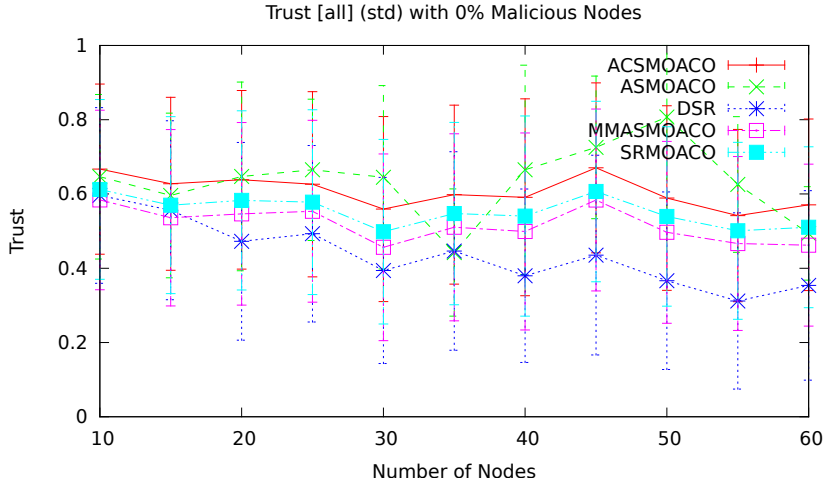


(c) 50% Malicious Nodes

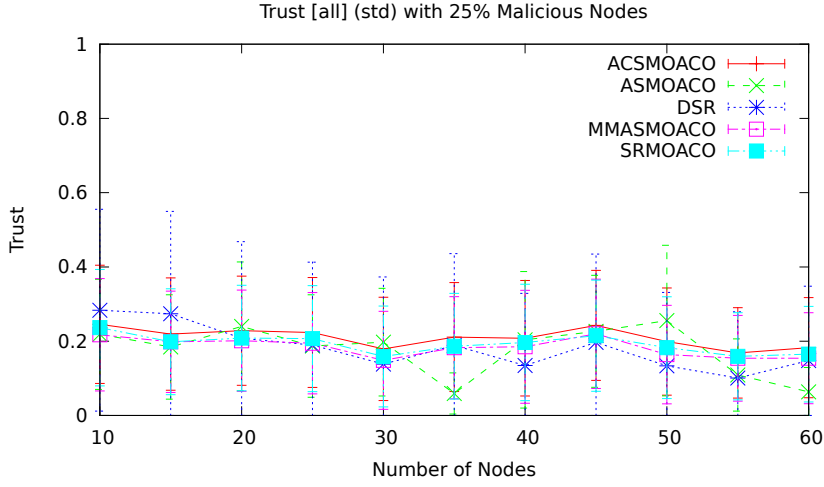


(d) 75% Malicious Nodes

Figure 5.27: Routing overhead with different percentages of malicious nodes for random topology with RBE traffic pattern

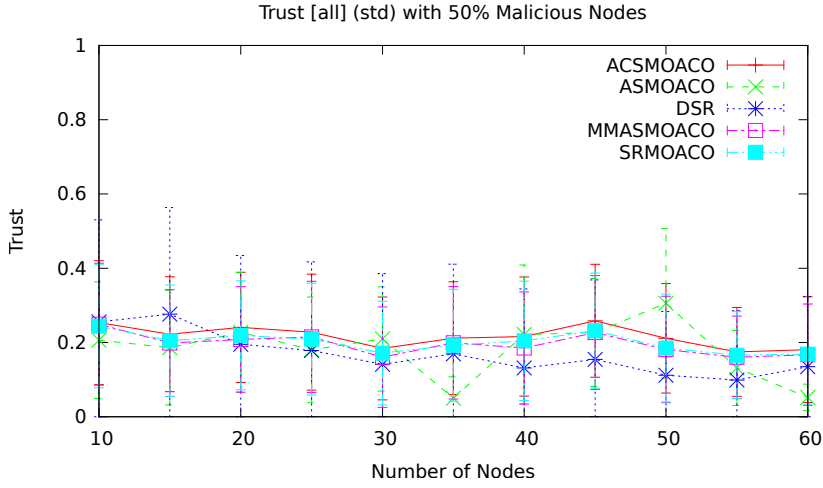


(a) 0% Malicious Nodes

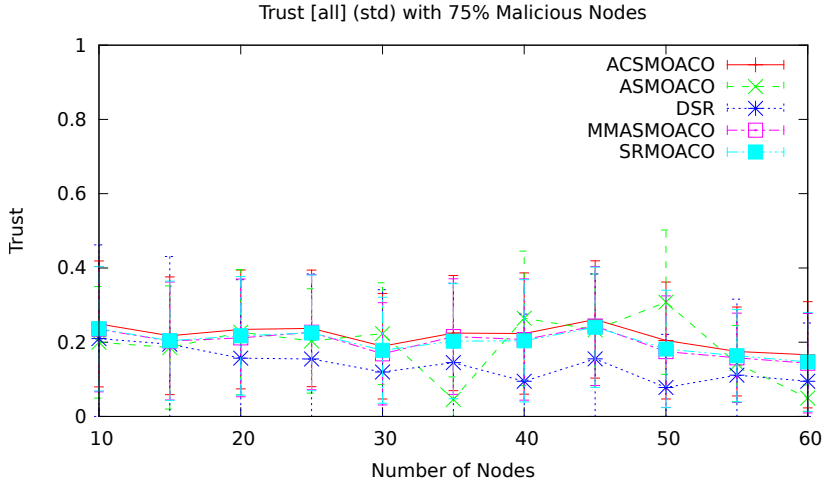


(b) 25% Malicious Nodes

Figure 5.28: Trust with different percentages of malicious nodes for random topology with RBE traffic pattern



(c) 50% Malicious Nodes



(d) 75% Malicious Nodes

Figure 5.28: Trust with different percentages of malicious nodes for random topology with RBE traffic pattern

5.3 Remarks

In this subsection a few remarks on the conducted experiments are made that should help to put the results in a certain context:

Pre-Experiments

As stated before, it was assumed that the parameters from existing MOACO researches could not be transferred directly to the area of WSNs due to their unique constraints. Consequently, in a set of pre-experiments the 'optimal' parameters for the MOACO-based WSN routing algorithms should be found, which however involved several challenges:

Firstly, it must be stated that finding the 'optimal' parameters for the MOACO-based algorithms is an MCOP itself, namely finding the optimal parameters out of all possible parameter combinations. This involves also dependencies between certain parameters that are not obvious at first glance. Due to the fact that the simulation of each MOACO-based algorithm scenario took multiple days, it is clear that only a subset of parameter combinations could be tested in an experimental manner. Therefore, the conducted simulation results cannot give an exhaustive answer to the optimal MOACO parameters for the routing in WSNs, but rather some sort of tendency.

Another challenge was the aggregation of values obtained from the pre-experiments to a final best parameter because for different metrics, different parameter values performed best. The used averaging of the parameter values, leads to some final results, but depending on the desired focus other parameter values may perform better for certain application scenarios.

Besides, it should be highlighted that the obtained values for the MOACO parameters look static, but it can be assumed that, depending on the actual network situation, a dynamic finding of parameters that adapts to the current network situation may be reasonable.

Experiment

Also the subsequent experiment involved several challenges:

A flaw in the experiments is also that at the current implementation state, the iterations of the MOACO-based routing algorithms are triggered by a timer, causing in scenarios that are less dynamic a too large overhead. In the future one could imagine a more adaptive approach, which starts new iterations triggered by certain events.

A general problem of the conducted experiments is the quantity of randomness involved in the experiments: in the experiments, the connections between source and destinations, the topology (in case of random topology), the malicious nodes, the packet dropping likelihood and the probabilistic behaviour of ants are based on randomness so that even on the same topology an almost unlimited number of different results can be created. Particularly, the selection of the source and destinations has a strong influence on the performance of the algorithms because in best case, source and destination are direct neighbours, while in worst case multiple hops are required to reach the destination. The only possibility to generalise these experiments is to repeat them for several runs and then, average the results – as done in this thesis. Although, the number of repetitions for each experiment is with 50 runs already quite high, for more stable results more repetitions are required, which however, need significantly more simulation time and result in large amounts of output data that need to be analysed. Nevertheless, the conducted experiments shows some interesting results for a set of realistic scenarios, which have a more generalisable significance than providing results for completely pre-defined simulation scenarios.

5.3 Conclusions

For the application packet end-to-end delay it can be stated that the MOACO-based algorithms are suitable for both, the grid and the random topology, as well as for the tested traffic patterns, RTM and RBE. Although, the DSR-based algorithm has a lower application packet end-to-end delay for almost all scenarios, the MOACO-based algorithms stay even on average below 0.6 s. The difference in the delay between the DSR-based and the MOACO-based algorithms can be mainly explained by the different route discovery approaches: while the MOACO-based algorithms always need to wait until all forward ants have reached the destination

or a timer exceeded (required if ants get lost), whereas the DSR-based algorithm uses broadcasting and the first route found is used. As shown by the experiments, the number of ants used by the MOACO-based algorithms plays an important role for the application packet end-to-end delay because more ants result in more network packets, a higher likelihood of packet collisions and thus, delayed transmissions. As the expiration timer of the MOACO-based routing algorithms can be controlled by a timer, the application packet end-to-end delay might even be reduced in the future by optimising this timer depending on the network size.

For the application PDR it can be observed that in the grid topology with the RTM traffic pattern, the DSR-based algorithm outperforms all MOACO-based algorithms up to 36 nodes, afterwards the DSR-based algorithm is outperformed by all MOACO-based algorithms, except `ASMOACO`. For the RBE traffic pattern on the grid topology the DSR-based algorithm is outperformed by all MOACO starting from 49 nodes on, also when the percentage of malicious nodes is increased. For the random topology it can be observed that all routing algorithms have on average a similar PDR for networks with more than 35 nodes. However, in comparison to the MOACO-based algorithms, the DSR-based algorithm seems to decrease dramatically for higher number of nodes (> 45). It seems to be likely that this effect continues so that the MOACO-based algorithms will outperform the DSR-based algorithm for larger network sizes; however, this should be examined in future experiments. A not so good application PDR can only be observed for `ACSMOACO` in the random topology with the RBE traffic pattern. However, it is likely that this results from the randomness of the scenario so that future experiments should be conducted to examine this assumption. In general it can be observed that, as expected, with the increasing number of malicious nodes the overall average PDR is reduced, while the algorithms get closer together so that a real difference between the routing algorithms cannot be observed. However, on average the MOACO-based algorithms have a similarly good application PDR as the DSR-based algorithm, particularly for larger networks and also when there are many malicious nodes in the network.

The average number of hops for the DSR-based and the MOACO-based routing algorithms behave quite similar no matter which topology is cho-

sen and no matter which traffic pattern is selected: up to 50 nodes, the number of hops for the DSR-based algorithm is slightly increasing from ~ 1.5 up to ~ 3.2, after that the average number of hops stays around 3. In contrast, the average number of hops for the MOACO-based algorithms is between 2.5 and 3, depending on the chosen MOACO-based algorithm. The number of hops stays almost constant for the MOACO-based algorithms when the number of nodes in the network is increased. For all tested scenarios it can be observed that when the number of malicious nodes is increased, the average number of hops of the MOACO-based algorithm is slightly decreased, while the DSR-based algorithm stays approximately on the same level so that the MOACO-based algorithms outperform the DSR-based algorithms for scenarios with more than 35 nodes. The constant number of hops for the MOACO-based algorithms can be explained by the fast convergence of the ants to (near) optimal routes with less hops due to their biased used of routes with pheromone.

For the average residual energy of the tested routing algorithms it can be observed that for the different topologies and traffic patterns similar results are obtained: while the average residual energy for the DSR-based algorithm is slightly increasing the more nodes are in the network, the MOACO-based algorithms have a little less residual energy. However, the standard deviation of the DSR-based algorithm is much higher than the one observed for the MOACO-based algorithm. Interestingly, all MOACO-based algorithms have the same residual energy so that the different algorithms cannot be distinguished regarding the average residual energy. The reason why the MOACO-based algorithms require slightly more energy than the DSR-based algorithm is based on two facts: *beaconing* and *iterations*. Due to the fact that the MOACO-based algorithms use unicast messages for the communication, each sensor node needs to be aware of its neighbours; therefore, additional broadcast beaconing messages are used to make all nodes in the radio range aware of the nodes' existence, which, however, require additional energy. The other thing is the number of iterations used in the MOACO-based algorithms: while the DSR-based algorithm just uses one broadcasting mechanism to discover the routes, the MOACO-based algorithms try to improve their routes after a certain timespan by sending out forwarding ants for discovery purposes again, leading to multiple iterations and thus, more sending and receiving

power.

In general it can be stated that the overhead of the DSR-based routing algorithm is lower than for the MOACO-based algorithms for all tested topologies and traffic patterns. This can again be explained by the fact that the DSR-based routing algorithm is doing exactly one route discovery, while the MOACO-based algorithms use several iterations. However, for the grid topology with the RTM traffic pattern, the maximum overhead is around 20 % for the worst MOACO-based algorithm, when there are no malicious nodes in the network. Due to the fact that for the RBE traffic pattern more ants are used, the overhead increases on the same topology to $\sim$ 51 % in worst case. A similar overhead can be observed for the random topology; however, in case of RTM traffic the `ACSMOACO` performs worst and in case of RBE traffic `ASMOACO` performs worst, while all other MOACO algorithms are close to the DSR-based algorithm, particularly up to 35 nodes. In general it can be observed that when the percentage of malicious nodes in the network is increased, the average overhead of MOACO-based algorithms is increased: the more malicious nodes, the stronger the increase in the overhead. Particularly, if a lot of ants are utilised by a MOACO-based algorithm and a lot of them gets dropped by the malicious nodes, the overhead gets worst.

As expected, the average trust value of the MOACO-based routing algorithms is significantly higher (between $\sim$ 0.45 and $\sim$ 0.7) than of the DSR-based (between 0.38 and 0.6) when there are no malicious nodes in the network for all tested scenarios; though, the trust value interval seems to be similar, the MOACO-based algorithms the trust value of the DSR-based routing algorithm is decreasing, while the trust value of the MOACO-based algorithms stays constant or oscillates around one value. The only outlier can be observed for `ASMOACO` for the random topology with the RBE traffic pattern, which has an oscillating behaviour, which is likely to result from the discussed amplification problem created by the backward ants. The difference between the MOACO-based algorithms and the DSR-based algorithm can be explained by the fact, that the DSR-based algorithm is not aware of the trust values, while for the MOACO-based algorithms trust is one of the objectives used for route optimisation. As expected, when the number of malicious nodes in the network is in-

creased, the trust values for all routing algorithms is decreased by around two third. Since there are not so large differences for the MOACO-based algorithms when increasing the percentage of malicious nodes to 25 % to 50 % to 75 %, the DSR-based algorithm decreases more and more when increasing the number of nodes in the network resulting in a larger gap between the MOACO-based algorithms and the DSR-based algorithm.

Additionally, it should be highlighted that for almost all scenarios the number of ants was increased from the RTM to the RBE traffic pattern. This leads, on the one hand, to a better reliability and a higher PDR for the RBE traffic, but, on the other hand, the application packet end-to-end delay was increased as well as the overhead. Consequently, one of the most important factors for the MOACO-based algorithms is the number of ants that is used, because each ant equals a routing packet and thus, increases the overall data traffic in the network. In dense networks the huge amount of routing packets leads to collisions and therefore, to additional delay and a higher likelihood of lost packets.

Although, the MOACO-based algorithms perform as good as the DSR-based scenario or even better in some cases, it should be again stated that the DSR-based routing algorithm was only utilising exactly one discovery phase, while the MOACO-based routing algorithms was using multiple iterations. For more dynamic networks, it can be assumed that the DSR broadcasting discovery process needs to be triggered more often so that the distance between the MOACO-based algorithms and the DSR-based algorithm, particularly regarding the overhead, will be reduced. Further simulations should be conducted in the future to get an answer to the question what will happen in more dynamic network scenarios.

5.4 Summary

In this chapter, the experiments were conducted and evaluated, which should deliver some empirical data to answer the question if the MARFWSN-based routing algorithms, based on MOACO algorithms, are suitable for the use in WSNs in terms of performing as good as existing routing protocols, while additionally mitigating the effect of insider attacks.

Therefore, in the first step, the settings of the experiments were explained including the general simulation scenario configuration such as the playground, the network topologies and the data traffic pattern. Afterwards, the general sensor node configuration was discussed including the used network stack containing the application layer, the network layer, the MAC and the physical layer. Subsequently, a more detailed look was taken on the configuration of the routing algorithms as most crucial part of the experiments. This included the MARFWSN-based routing approaches (`SRMOACO`, `ASMOACO`, `MMASMOACO` and `ACSMOACO`) and the DSR-based routing algorithm, used for comparison purposes. Furthermore, the performance metrics were discussed that were used to compare the different routing approaches depending on the traffic pattern. In the next step, the configuration of the pre-experiments and their results were discussed, which was conducted to find the best parameters for the MOACO-based parameters. Then, based on the pre-experiments, a set of experiments was conducted to compare the different MOACO-based routing algorithms using MARFWSN to the DSR-based routing algorithm in different scenarios. In the experiments two different network topologies (grid and random topology) as well as two different traffic patterns (RTM and RBE) were considered, leading to four sub-experiments. The results of the experiments were discussed and complemented by some remarks that should explain the context of the results, which should be considered when interpreting the experimental results. Finally, the results of the experiments were summarised and conclusions were drawn showing the suitability of the MOACO-based routing algorithms in WSNs as well as the challenges that need to be met.

6

Conclusions and Future Work

> Science never solves a problem without creating ten more.
>
> George Bernard Shaw (1856 - 1950)

In this chapter the conclusions of the thesis will be drawn and possible future work will be discussed:

In this thesis MARFWSN was proposed, a routing framework for WSNs that utilises MOACO-based algorithms for the optimisation of routes in WSNs. The MOACO-based algorithms are a subset of biologically-inspired algorithms that are based on the idea of using artificial ant colonies to solve MCOPs, i.e. combinatorial problems with multiple (conflicting) objectives. As the routing in WSNs can be seen as such a problem, the robust features of the MOACO-based algorithms seem to be well-suited.

As insider attacks, i.e. compromised nodes that are a valid part of the network, are one of the big challenges in WSNs, the proposed routing algorithms consider trust as soft security measure to mitigate the impact of these attacks. This is achieved by using the trust value of each node as one of the objectives the MOACO-based algorithm is optimising for.

6.1 Contributions

In the following the contributions of this thesis are summarised:

6.1 Foundations

In the first step, the security issues of WSN routing protocols were identified, particularly considering insider attacks. Subsequently, the idea of BIAs were discussed as suitable approaches for the routing in WSNs with a special focus on ant-based algorithms. The origin of ACO and

its extension MOACO were explained as well as their applications. The related works in this field were reviewed for the area of wired and wireless networks, including WSNs and MANETs.

6.1 Multi-objective Ant Colony Optimisation Routing Framework for WSNs

Based on the findings MARFWSN, a MOACO-based routing framework for WSN, was designed and implemented. MARFWSN aims to be a common framework that enables the testing of different MOACO-based algorithms in the context of WSNs routing.

Design Goals

The main design goals that were considered in the development of MARFWSN are as follows:

- **Deployment**: Based on the layered approach, the framework should integrate seamlessly as network layer into the WSN network stack without the need for modification of other network layers.
- **Dockability**: The framework should provide a docking interface so that different MOACO algorithms can be easily added, exchanged and examined, without the need to modify the framework itself.
- **Extendability**: The framework should be extendable in terms of that additional MOACO algorithms can be easily added to test their performance in WSN routing.
- **Flexibility**: The framework should not be restricted to a certain amount of objectives so that any number of objectives should be usable by the MOACO-based algorithms.
- **Common features**: The framework itself should deal with the functionality that is common to all MOACO algorithms such as sending of ants, management of pheromone and heuristic information etc.
- **Soft security**: The framework should support some integration of soft-security to mitigate the impact of insider attacks.

Implementation

Based on the design goals, MARFWSN was implemented in the simulation tool OMNeT++ [260] using the MiXiM framwork [261] that provides wireless and mobile functionality. Four MOACO-based algorithms were implemented for MARFWSN (`SRMOACO`, `ASMOACO`, `MMASMOACO` and `ACSMOACO`) based on their ACO ancestors. Besides, a traffic generator was implemented that uses a simple request-response protocol imitating common WSN application network traffic, i.e. the querying of a source by a sink node.

6.1 Experiments and Evaluation

MARFWSN, with the four implemented MOACO-based algorithms, was tested in several WSN scenarios with different parameter settings. The conducted experiments considered different topologies (random and grid) as well as different traffic types (RTM and RBE). Several network sizes were tested each for different percentages of malicious nodes (0 %, 25 %, 50 % and 75 %). By comparing the MARFWSN-based network layer to the DSR-based network layer conclusions were drawn regarding the performance differences and the general use of MOACO-based routing algorithms in WSNs.

The main outcomes of the evaluations can be summarised as follows:

- The pre-experiments showed that the parameters for MOACO-based algorithms are unique for WSNs and thus, cannot be directly transferred from other MCOPs, such as TSP or QAP.
- The number of ants used in the MARFWSN-based routing algorithms significantly influenced the performance of the MOACO-based algorithms because each ant is equivalent to one network packet that needs to be sent.
- The end-to-end delay of the MARFWSN-based network layer is on average slightly higher than for the DSR-based routing protocol because of the iterations and corresponding timer used in the MOACO-based algorithms.
- In terms of application PDR, the MARFWSN-based routing proto-

cols perform on average similarly good as the DSR-based protocol; in some cases MARFWSN even outperforms the DSR-based protocol for large network sizes. More ants lead to a better application PDR, but also to more overhead and delay.

- While the average number of hops of the MARFWSN-based network layer stays rather constant for all network sizes due to the convergence by the pheromone, the number of hops of the DSR-based protocol is less for small network sizes, but similar or even larger for networks with many sensor nodes.

- The residual energy for the DSR-based protocol was better than for the MARFWSN-based protocol, which uses multiple iterations and additionally beacons for neighbourhood discovery. Longer simulations need to be conducted to examine this tendency.

- Using trust as third objective allowed the MARFWSN-based network layer to find more trustful routes in the network in comparison to the DSR-based routing protocol so that the impact of insider attacks can be mitigated for the MARFWSN-based network layer.

- The DSR-based routing algorithms outperform the MARFWSN-based routing algorithms, utilising several iterations, in terms of overhead, particularly for larger networks.

All in all, it can be stated that the MARFWSN-based routing algorithms are basically suitable for the use in WSNs by performing similarly good as the DSR-based protocol in the tested scenarios. The idea of incorporating trust in the routing process for MARFWSN showed promising results to mitigate insider attacks.

However, the results of the conducted simulation experiments should be seen as first tendency because more repetitions and a longer duration of the simulations could lead to more robust results, particularly for the scenarios with high randomness. Due to the long simulation time a vast number of additional hours need to be spent to improve the simulations results.

6.2 Future Work

In this thesis at several points limitations, challenges and ideas were presented that could lead to following researches. Therefore, in this section only a few ideas for future work will be discussed briefly, which go beyond the scope of this thesis, but can be used as starting point for further research.

Parameters

In general, it would be interesting to find out more about the parameters that can be used in the MOACO-based routing algorithms and their interactions. Additional values of parameters could be checked and analysed to gain new insights into their influence on the WSN routing process. However, as stated before, finding the optimal parameters for the MOACO-based algorithms is a MCOP itself so that it would also be interesting to investigate in a more formalised approach of 'solving' this problem. It may also be worthwhile to conduct research on adaptive parameter settings that are capable of reacting to changes in the network.

Scalability

Due to the limited time frame, only a limited subset of different network sizes could be simulated. However, based on the obtained results and the tendencies that could be observed for the different routing algorithms towards the upper limit of the network size, it would be interesting to investigate how the routing algorithms perform if the number of nodes in the network is increased up to multiple hundreds of sensor nodes. It is expected that the MOACO-based algorithms perform better or more efficient, while the DSR-based protocol becomes inefficient because of the used broadcasting mechanism.

Network Attacks

Regarding the security in the WSN it would also be interesting to make the malicious behaviour in the WSN more sophisticated. As a result, it could be investigated how the performance changes when 'real' WSN networks attacks emerge, such as wormhole attacks or Sybil attack, in-

stead of just observing the performance in the network when packets are dropped, as done in the simplified attacker model used in this thesis.

Trust and Reputation

As one of the main features of the presented MOACO-based algorithms is the consideration of trust as one of the objectives in the route optimisation process, it should be investigated in a real TRS that deals with the management and exchange of trust values in the WSN. Based on such a system, MARFWSN should be modified so that the trust values of the next hop can already be obtained at the current hop by each ant. This should lead to an even better performance of the MOACO-based routing algorithm because less ants will get lost by visiting malicious nodes, as this is often the case in the simplified current implementation. However, when adding a TRS it needs to be considered that the overhead of such a system should be kept small in terms of the computational power and memory required at each node as well as the amount of messages that needs to be additionally sent, which are increasing the overall network traffic load. Also the TRS itself can be an interesting target for adversaries so that new challenges will arise to protect these systems.

Hardware Implementation

As the configuration of the sensor nodes used in the simulations were already based on the Iris Mote specifications [9], as next logical step it would be interesting to implement MARFWSN on real hardware sensor nodes in a test-bed and then, to compare these results to the obtained results of the experiments conducted in OMNeT++.

References

[1] A. Abraham, A. Hassanien, and V. Snášel, *Computational social network analysis: trends, tools and research advances*. Springer-Verlag New York Inc, 2009.

[2] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE wireless communications*, vol. 11, no. 6, pp. 6–28, 2004.

[3] A. Srinivasan, J. Teitelbaum, H. Liang, J. Wu, and M. Cardei, "Reputation and Trust-based Systems for Ad Hoc and Sensor Networks," *Algorithms and Protocols for Wireless Ad Hoc and Sensor Networks*, 2006.

[4] F. Streichert and M. Tanaka-Yamawaki, "A new scheme for interactive multi-criteria decision making," in *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 2006, pp. 655–662.

[5] J. Deneubourg, S. Aron, S. Goss, and J. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of Insect Behavior*, vol. 3, no. 2, pp. 159–168, 1990.

[6] S. Goss, S. Aron, J. Deneubourg, and J. Pasteels, "Self-organized shortcuts in the Argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.

[7] A. Varga, "OMNeT++ Discrete event simulation system. User Manual," *Technical University of Budapest, Dept. of Telecommunications*, 2006.

[8] OMNeT++ Community. (2010, May) OMNeT++ 4.0 IDE (picture). [Online]. Available: http://omnetpp.org/doc/omnetpp40/ide-overview/pictures/img1.png

[9] Crossbow Technology, Inc. (2011, Aug.) IRIS Mote Datasheet. [Online]. Available: http://www.dinesgroup.org/projects/images/pdf_files/iris_datasheet.pdf

[10] J. Branke, K. Deb, and K. Miettinen, *Multiobjective optimization: Interactive and evolutionary approaches*. Springer-Verlag New York Inc, 2008, vol. 5252.

[11] M. Dorigo and T. Stützle, "Ant Colony Optimization: Overview and Recent Advances," *Handbook of Metaheuristics*, pp. 227–263, 2010.

[12] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[13] A. Kellner, K. Behrends, and D. Hogrefe, "Simulation Environments for Wireless Sensor Networks," Institute of Computer Science, Georg-August-Universität Göttingen, Germany, Technical Report No. IFI-TB-2010-04, June 2010, ISSN 1611-1044. [Online]. Available: http://filepool.informatik.uni-goettingen.de/publication/tmg/2010/AK_KB_2010_01.pdf

[14] BCC Research Market Forecasting. (2011, Mar.) Sensors: Technologies and Global Markets (IAS006D). [Online]. Available: http://www.bccresearch.com/report/IAS006D.html

[15] Z. Hao, H. Huang, and R. Cai, "Bio-inspired algorithms for tsp and generalized tsp," *Travelling Salesman Problem*, p. 35, 2008.

[16] M. Sotelo-Figueroa, R. Baltazar, and M. Carpio, "Application of the bee swarm optimization bso to the knapsack problem," *Soft Computing for Recognition Based on Biometrics*, pp. 191–206, 2010.

[17] J. Potvin, "A review of bio-inspired algorithms for vehicle routing," *Bio-inspired algorithms for the vehicle routing problem*, pp. 1–34, 2009.

[18] R. Pagliari, Y. Hong, and A. Scaglione, "Bio-inspired algorithms for decentralized round-robin and proportional fair scheduling," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 4, pp. 564–575, 2010.

[19] D. Angus and C. Woodward, "Multiple objective ant colony optimisation," *Swarm intelligence*, vol. 3, no. 1, pp. 69–85, 2009.

[20] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century," *Computer*, vol. 36, no. 3, pp. 25–31, 2003.

[21] H. Karl and A. Willig, *Protocols and architectures for wireless sensor networks*. Wiley-Interscience, 2007.

[22] M. Hempstead, M. Lyons, D. Brooks, and G. Wei, "Survey of hardware systems for wireless sensor networks," *Journal of Low Power Electronics*, vol. 4, no. 1, p. 11, 2008.

[23] R. Bischoff, J. Meyer, and G. Feltrin, "Wireless Sensor Network Platforms," *Encyclopaedia of structural health monitoring. Chichester: John Wiley & Sons*, pp. 1229–1238, 2009.

[24] A. García-Hernando, J. Martínez-Ortega, J. López-Navarro, A. Prayati, and L. Redondo-López, "Hardware Platforms for WSNs," *Problem Solving for Wireless Sensor Networks*, pp. 1–34, 2008.

[25] R. Min and A. Chandrakasan, "MobiCom poster: top five myths about the energy consumption of wireless communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 1, pp. 65–67, 2003.

[26] K. Pister, J. Kahn, B. Boser *et al.*, "Smart dust: Wireless networks of millimeter-scale sensor nodes," *Highlight Article in*, p. 2, 1999.

[27] G. Huang, "Casting the wireless sensor net," *TECHNOLOGY REVIEW-MANCHESTER NH-*, vol. 106, no. 6, pp. 50–57, 2003.

[28] E. Shi and A. Perrig, "Designing secure sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 38–43, 2004.

[29] J. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security: A survey," *Security in distributed, grid, mobile, and pervasive computing*, p. 367, 2007.

[30] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys and Tutorials*, vol. 8, no. 2, pp. 2–23, 2006.

[31] J. Zhang, P. Orlik, Z. Sahinoglu, A. Molisch, and P. Kinney, "UWB systems for wireless sensor networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 313–331, 2009.

[32] L. Rasmusson and S. Jansson, "Simulated social control for secure Internet commerce," in *Proceedings of the 1996 workshop on New security paradigms*. ACM New York, NY, USA, 1996, pp. 18–25.

[33] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[34] D. McKnight and N. Chervany, "The meanings of trust," *Trust in Cyber-Societies-LNAI*, vol. 2246, pp. 27–54, 2001.

[35] O. Dictionary, "Oxford English Dictionary," *Notes and Queries*, 2004.

[36] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, "A Survey of Trust and Reputation Management Systems in Wireless Communications," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1755–1772, 2010.

[37] A. Jøsang and S. Pope, "Semantic constraints for trust transitivity," in *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*. Australian Computer Society, Inc., 2005, pp. 59–68.

[38] P. Dasgupta, "Trust as a Commodity," *Trust: Making and Breaking Cooperative Relations, electronic edition, Department of Sociology, University of Oxford*, pp. 49–72, 2000.

[39] R. Lewicki and B. Bunker, "Trust in relationships: A model of development and decline," *Conflict, cooperation, and justice: Essays inspired by the work of Morton Deutsch*, pp. 133–173, 1995.

[40] D. Gefen, E. Karahanna, and D. Straub, "Trust and TAM in online shopping: An integrated model," *Mis Quarterly*, vol. 27, no. 1, pp. 51–90, 2003.

[41] D. McKnight, L. Cummings, and N. Chervany, "Initial trust formation in new organizational relationships," *Academy of Management Review*, vol. 23, no. 3, pp. 473–490, 1998.

[42] T. Grandison and M. Sloman, "A survey of trust in internet applications," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 4, pp. 2–16, 2000.

[43] P. Pardalos and H. Romeijn, *Handbook of global optimization. Volume 2*. Kluwer, 2002.

[44] P. Papalambros and D. Wilde, *Principles of optimal design: modeling and computation*. Cambridge Univ Pr, 2000.

[45] L. Paquete and T. Stützle, "Stochastic local search algorithms for multiobjective combinatorial optimization: A review," *Handbook of Approximation Algorithms and Metaheuristics. Chapman & Hall/CRC*, pp. 29–1, 2007.

[46] M. Garey and D. Johnson, *Computers and intractability*. Freeman San Francisco, 1979, vol. 174.

[47] M. Dorigo and G. Di Caro, "The ant colony optimization metaheuristic," *New ideas in optimization*, pp. 11–32, 1999.

[48] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[49] H. Lourenco, O. Martin, and T. Stützle, "Iterated local search," *Handbook of metaheuristics*, pp. 320–353, 2003.

[50] N. Mladenovi and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[51] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of Statistical Physics*, vol. 34, no. 5, pp. 975–986, 1984.

[52] V. Černỳ, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of optimization theory and applications*, vol. 45, no. 1, pp. 41–51, 1985.

[53] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[54] F. Glover, "Tabu search, Part I," *ORSA Journal on Computing*, vol. 3, pp. 190–206, 1989.

[55] ——, "Tabu search, Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[56] M. Dorigo, G. Caro, and L. Gambardella, "Ant algorithms for discrete optimization," *Artificial life*, vol. 5, no. 2, pp. 137–172, 1999.

[57] O. Cordon, F. Herrera, and T. Stützle, "A review on the ant colony optimization metaheuristic: Basis, models and new trends," in *Mathware & Soft Computing*. Citeseer, 2002.

[58] C. Coello, *Advances in Multi-objective Nature Inspired Computing*. Springer Verlag, 2010.

[59] K. Miettinen, *Nonlinear multiobjective optimization*. Springer, 1999, vol. 12.

[60] C. Hwang, A. Masud, S. Paidy, and K. Yoon, *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Springer Berlin, 1979.

[61] E. Triantaphyllou, *Multi-criteria decision making methods: a comparative study*. Springer Netherlands, 2000.

[62] P. Fishburn, "Additive utilities with incomplete product set: Applications to priorities and assignments, operations research," 1967.

[63] P. Bridgman, *Dimensional analysis*. Yale University Press, 1922.

[64] D. Miller and M. Starr, *Executive decisions and operations research*. Prentice-Hall, 1960.

[65] J. Cochrane and M. Zeleny, *Multiple criteria decision making*. Univ of South Carolina Pr, 1973.

[66] A. Wierzbicki, "A mathematical basis for satisficing decision making," *Mathematical modelling*, vol. 3, no. 5, pp. 391–405, 1982.

[67] ——, "On the completeness and constructiveness of parametric characterizations to vector optimization problems," *OR spectrum*, vol. 8, no. 2, pp. 73–87, 1986.

[68] Y. Haimes, L. Lasdon, and D. Wismer, "On a bicriterion formulation of the problems of integrated system identification and system optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 1, no. 3, pp. 296–297, 1971.

[69] V. Chankong and Y. Haimes, *Multiobjective decision making: theory and methodology*. North-Holland New York, 1983, vol. 8.

[70] P. Fishburn, "Lexicographic orders, utilities and decision rules: A survey," *Management Science*, vol. 20, no. 11, pp. 1442–1471, 1974.

[71] H. Lourenço, O. Martin, and T. Stützle, "Iterated local search: framework and applications," *Handbook of Metaheuristics*, pp. 363–397, 2010.

[72] H. Lorenço, O. Martin, and T. Stützle, "Iterated local search," *ISORMS*, vol. 57, p. 321, 2002.

[73] S. Olariu and A. Zomaya, *Handbook of bioinspired algorithms and applications*. CRC Press, 2006, vol. 7.

[74] W.-P. Tsang, "Bio-inspired algorithms for single and multi-objective optimization," *HKU Theses Online (HKUTO)*, 2009.

[75] C. Peterson and B. Soderberg, *Artificial neural networks*. Blackwell Scientific Publications, Oxford, 1993.

[76] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2004.

[77] L. De Castro and J. Timmis, *Artificial immune systems: a new computational intelligence approach*. Springer Verlag, 2002.

[78] L. Fogel, A. Owens, and M. Walsh, *Artificial intelligence through simulated evolution*. John Wiley, 1966.

[79] L. Fogel, "Intelligence through simulated evolution: Four decades of evolutionary programming," 1999.

[80] A. Eiben and J. Smith, *Introduction to evolutionary computing*. Springer Verlag, 2003.

[81] I. Rechenberg, "Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution, frommann-holzboog," *Stuttgart. German*, 1973.

[82] H. Schwefel, *Numerische optimierung von computer-modellen mittels der evolutionsstrategie*. Birkhäuser, 1977.

[83] H. Beyer, *The theory of evolution strategies*. Springer Verlag, 2001.

[84] J. Koza, *Genetic programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[85] ——, *Genetic programming II: automatic discovery of reusable programs*. MIT Press, 1994.

[86] J. Koza and R. Poli, "Genetic programming," *Search Methodologies*, pp. 127–164, 2005.

[87] Z. Michalewicz and Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-verlag Berlin, 1992, vol. 19.

[88] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional, 1989.

[89] L. Schmitt, "Theory of genetic algorithms," *Theoretical Computer Science*, vol. 259, no. 1-2, pp. 1–61, 2001.

[90] M. Vose, *The simple genetic algorithm: foundations and theory*. The MIT Press, 1999, vol. 12.

[91] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, vol. 102, pp. 703–703, 1993.

[92] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems.* Oxford University Press, USA, 1999.

[93] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4. IEEE, 1995, pp. 1942–1948.

[94] R. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the 2001 congress on evolutionary computation*, vol. 1. Piscataway, NJ, USA: IEEE, 2001, pp. 81–86.

[95] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999.

[96] M. Dorigo and T. Stützle, *Ant colony optimization.* MIT press, 2004.

[97] P. Rabanal, I. Rodríguez, and F. Rubio, "Using river formation dynamics to design heuristic algorithms," *Unconventional Computation*, pp. 163–177, 2007.

[98] J. Bishop, "Stochastic searching networks," in *Artificial Neural Networks, 1989., First IEE International Conference on (Conf. Publ. No. 313).* IET, 1989, pp. 329–331.

[99] H. Shah-Hosseini, "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm," *International Journal of Bio-Inspired Computation*, vol. 1, no. 1, pp. 71–79, 2009.

[100] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[101] F. Ducatelle, G. Di Caro, and L. Gambardella, "Principles and applications of swarm intelligence for adaptive routing in telecommunications networks," *Swarm Intelligence*, vol. 4, no. 3, pp. 173–198, 2010.

[102] P. Grassé, "La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs," *Insectes sociaux*, vol. 6, no. 1, pp. 41–80, 1959.

[103] C. Blum, *Theoretical and practical aspects of ant colony optimization*. Akademische Verlagsfesellschaft, 2004.

[104] ——, "Beam-ACO–Hybridizing ant colony optimization with beam search: An application to open shop scheduling," *Computers & Operations Research*, vol. 32, no. 6, pp. 1565–1591, 2005.

[105] M. Guntsch and M. Middendorf, "Applying population based ACO to dynamic optimization problems," *Ant Algorithms*, pp. 97–104, 2002.

[106] ——, "Solving multi-criteria optimization problems with population-based aco," in *Evolutionary Multi-Criterion Optimization*. Springer, 2003, pp. 71–71.

[107] C. Blum, A. Roli, and M. Dorigo, "HC–ACO: The hyper-cube framework for Ant Colony Optimization," in *Proceedings of MIC*, vol. 2. Citeseer, 2001, pp. 399–403.

[108] C. Blum and M. Dorigo, "The hyper-cube framework for ant colony optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 2, pp. 1161–1172, 2004.

[109] O. Cordón, I. de Viana, F. Herrera, and L. Moreno, "A new ACO model integrating evolutionary computation concepts: the best-worst ant system," *Proceedings of Ants2000*, pp. 22–29, 2000.

[110] O. Cordón, I. de Viana, and F. Herrera, "Analysis of the best-worst ant system and its variants on the TSP," *Mathware and Soft computing*, vol. 9, no. 2/3, pp. 177–192, 2002.

[111] ——, "Analysis of the best-worst ant system and its variants on the QAP," *Ant Algorithms*, pp. 139–152, 2002.

[112] V. Maniezzo, "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem," *INFORMS Journal on Computing*, vol. 11, no. 4, pp. 358–369, 1999.

[113] B. Bullnheimer, R. Hartl, and C. Strauß, "A new rank based version of the ant system-a computational study," in *Central European Journal for Operations Research and Economics*. Citeseer, 1997.

[114] T. Stützle and H. Hoos, "Improving the Ant System: A Detailed Report on the MAX-MIN Ant System," Technical University of Darmstadt, Tech. Rep., 1996.

[115] ——, "MAX-MIN ant system and local search for the traveling salesman problem," in *Evolutionary Computation, 1997., IEEE International Conference on*. IEEE, 1997, pp. 309–314.

[116] T. Stützle, H. Hoos *et al.*, "MAX-MIN ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[117] M. Dorigo and L. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–82, 1997.

[118] ——, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.

[119] L. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. IEEE, 1996, pp. 622–627.

[120] L. Gambardella, M. Dorigo *et al.*, "Ant-Q: A reinforcement learning approach to the traveling salesman problem," in *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer, 1995, pp. 252–260.

[121] M. Dorigo and L. Gambardella, "A study of some properties of Ant-Q," *Parallel Problem Solving from Nature—PPSN IV*, pp. 656–665, 1996.

[122] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, pp. 29–41, 1996.

[123] M. Dorigo, "Optimization, Learning and Natural Algorithms (in Italian)," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[124] M. Dorigo, V. Maniezzo, A. Colorni, and V. Maniezzo, "Positive Feedback as a Search Strategy," Dipartimento di Elettronica - Politecnico di Milano, Tech. Rep., 1991.

[125] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: An autocatalytic optimizing process," *TR91-016, Politecnico di Milano*, 1991.

[126] T. Stützle, *Local Search Algorithms for Combinatorial Problems - Analysis, Improvements and New Applications.* Ios Pr Inc, 1999.

[127] K. Socha, J. Knowles, and M. Sampels, "A max-min ant system for the university course timetabling problem," *Ant Algorithms*, pp. 63–77, 2002.

[128] W. Gutjahr, "A graph-based ant system and its convergence," *Future Generation Computer Systems*, vol. 16, no. 9, pp. 873–888, 2000.

[129] ——, "ACO algorithms with guaranteed convergence to the optimal solution," *Information Processing Letters*, vol. 82, no. 3, pp. 145–153, 2002.

[130] T. Stützle and M. Dorigo, "A short convergence proof for a class of ant colony optimization algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 4, pp. 358–365, 2002.

[131] L. Gambardella, É. Taillard, and G. Agazzi, "Macs-vrptw: A multiple colony system for vehicle routing problems with time windows," in *New ideas in optimization.* Citeseer, 1999.

[132] M. Reimann, K. Doerner, and R. Hartl, "D-Ants: savings based ants divide and conquer the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 4, pp. 563–591, 2004.

[133] L. Gambardella and M. Dorigo, "An ant colony system hybridized with a new local search for the sequential ordering problem," *INFORMS Journal on Computing*, vol. 12, no. 3, pp. 237–255, 2000.

[134] G. Di Caro and M. Dorigo, "AntNet: Distributed stigmergetic control for communications networks," *Journal of Artificial Intelligence Research*, vol. 9, no. 1, pp. 317–365, 1998.

[135] F. Ducatelle, G. Di Caro, and L. Gambardella, "Using ant agents to combine reactive and proactive strategies for routing in mobile ad-hoc networks," *International Journal of Computational Intelligence and Applications*, vol. 5, no. 2, pp. 169–184, 2005.

[136] G. Di Caro, F. Ducatelle, and L. Gambardella, "AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks," *European Transactions on Telecommunications*, vol. 16, no. 5, pp. 443–455, 2005.

[137] K. Socha, M. Sampels, and M. Manfrin, "Ant algorithms for the university course timetabling problem with regard to the state-of-the-art," *Applications of evolutionary computing*, pp. 334–345, 2003.

[138] D. Costa and A. Hertz, "Ants can colour graphs," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 295–305, 1997.

[139] D. Merkle, M. Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 4, pp. 333–346, 2002.

[140] M. Den Besten, T. Stützle, and M. Dorigo, "Ant colony optimization for the total weighted tardiness problem," in *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 611–620.

[141] D. Merkle and M. Middendorf, "Ant colony optimization with global pheromone evaluation for scheduling a single machine," *Applied Intelligence*, vol. 18, no. 1, pp. 105–111, 2003.

[142] L. Lessing, I. Dumitrescu, and T. Stützle, "A comparison between ACO algorithms for the set covering problem," *Ant Colony, Optimization and Swarm Intelligence*, pp. 105–122, 2004.

[143] C. Blum and M. Blesa, "New metaheuristic approaches for the edge-weighted k-cardinality tree problem," *Computers & Operations Research*, vol. 32, no. 6, pp. 1355–1377, 2005.

[144] G. Leguizamón and Z. Michalewicz, "A new version of ant system for subset problems," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2. IEEE, 1999.

[145] S. Fenet and C. Solnon, "Searching for maximum cliques with ant colony optimization," *Applications of Evolutionary Computing*, pp. 291–302, 2003.

[146] C. Solnon, "Solving permutation constraint satisfaction problems with artificial ants," in *ECAI*. Citeseer, 2000, pp. 118–122.

[147] ——, "Ants can solve constraint satisfaction problems," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 4, pp. 347–357, 2002.

[148] R. Parpinelli, H. Lopes, and A. Freitas, "Data mining with an ant colony optimization algorithm," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 4, pp. 321–332, 2002.

[149] D. Martens, M. De Backer, R. Haesen, B. Baesens, C. Mues, and J. Vanthienen, "Ant-based approach to the knowledge fusion problem," *Ant Colony Optimization and Swarm Intelligence*, pp. 84–95, 2006.

[150] L. De Campos, J. Fernandez-Luna, J. Gámez, and J. Puerta, "Ant colony optimization for learning Bayesian networks," *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 291–311, 2002.

[151] L. De Campos, J. Gámez Martín, and J. Puerta, "Learning bayesian networks by ant colony optimisation: searching in two different spaces," *Mathware & soft computing*, vol. 9, no. 3, pp. 251–268, 2008.

[152] A. Shmygelska and H. Hoos, "An ant colony optimisation algorithm for the 2 D and 3 D hydrophobic polar protein folding problem," *BMC bioinformatics*, vol. 6, no. 1, p. 30, 2005.

[153] O. Korb, T. Stützle, and T. Exner, "PLANTS: Application of ant colony optimization to structure-based drug design," *Ant Colony Optimization and Swarm Intelligence*, pp. 247–258, 2006.

[154] M. López-Ibáñez and T. Stützle, "An analysis of algorithmic components for multiobjective ant colony optimization: A case study on the biobjective TSP," *Artifical Evolution*, pp. 134–145, 2010.

[155] ——, "Automatic Configuration of Multi-Objective ACO Algorithms," *Swarm Intelligence*, pp. 95–106, 2010.

[156] ——, "The impact of design choices of multiobjective antcolony optimization algorithms on performance: an experimental study on the biobjective TSP," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 71–78.

[157] C. Garcia-Martinez, O. Cordón, and F. Herrera, "A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP," *European Journal of Operational Research*, vol. 180, no. 1, pp. 116–148, 2007.

[158] J. Angelo and H. Barbosa, "Ant Colony Algorithms for Multiobjective Optimization, Ant Colony Optimization - Methods and Applications," *InTech Publications*, 2011.

[159] S. Iredi, D. Merkle, and M. Middendorf, "Bi-criterion optimization with multi colony ant algorithms," in *Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 359–372.

[160] C. Mariano and E. Morales, "A multiple objective Ant-Q algorithm for the design of water distribution irrigation networks," in *First International Workshop on Ant Colony Optimization ANTS'98*. Citeseer, 1998.

[161] K. Doerner, R. Hartl, and M. Reimann, "Are COMPETants more competent for problem solving? The case of a multiple objective transportation problem." *Report Series SFB "Adaptive Information Systems and Modelling in Economics and Management Science"*, 2001.

[162] P. McMullen, "An ant colony optimization approach to addressing a jit sequencing problem with multiple objectives," *Artificial Intelligence in Engineering*, vol. 15, no. 3, pp. 309–317, 2001.

[163] V. T'kindt, N. Monmarche, F. Tercinet, and D. Laugt, "An ant colony optimization algorithm to solve a 2-machine bicriteria flow-shop scheduling problem," *European Journal of Operational Research*, vol. 142, no. 2, pp. 250–257, 2002.

[164] B. Baran and M. Schaerer, "A multiobjective ant colony system for vehicle routing problem with time windows." in *21 st IASTED International Multi-Conference on Applied Informatics*, 2003, pp. 97–102.

[165] P. Cardoso, M. Jesus, and A. Márquez, "Monaco-multi-objective network optimisation based on an aco," *Proc. X Encuentros de Geometria Computacional, Seville, Spain*, 2003.

[166] K. Doerner, W. Gutjahr, R. Hartl, C. Strauss, and C. Stummer, "Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection," *Annals of Operations Research*, vol. 131, no. 1, pp. 79–99, 2004.

[167] D. Pinto and B. Barán, "Solving multiobjective multicast routing problem with a new ant colony optimization approach," in *Proceeding of 3rd International IFIP/ACM Latin American Conference on Networking*, 2005, pp. 11–19.

[168] P. Gardel, B. Baran, H. Estigarribia, U. Fernandez, and S. Duarte, "Multiobjective reactive power compensation with an ant colony optimization algorithm," in *AC and DC Power Transmission, 2006. ACDC 2006. The 8th IEE International Conference on.* IET, 2006, pp. 276–280.

[169] D. Angus, "Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem," in *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on.* IEEE, 2007, pp. 333–340.

[170] B. Yagmahan and M. Yenisey, "A multi-objective ant colony system algorithm for flow shop scheduling problem," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1361–1368, 2010.

[171] I. Alaya, C. Solnon, and K. Ghedira, "Ant colony optimization for multi-objective optimization problems," in *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, vol. 1. IEEE, 2007, pp. 450–457.

[172] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[173] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, 2003.

[174] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," *Tik report*, vol. 214, 2006.

[175] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

[176] M. Hansen and A. Jaszkiewicz, *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1998.

[177] L. While, L. Bradstreet, L. Barone, and P. Hingston, "Heuristics for optimizing the calculation of hypervolume for multi-objective optimization problems," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3. IEEE, 2005, pp. 2225–2232.

[178] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards," *Computer communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[179] P. Ferrari, A. Flammini, D. Marioli, and A. Taroni, "Ieee802. 11 sensor networking," *Instrumentation and Measurement, IEEE Transactions on*, vol. 55, no. 2, pp. 615–619, 2006.

[180] L. Feeney, "A taxonomy for routing protocols in mobile ad hoc networks," *SICS Research Report*, 1999.

[181] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad hoc networks*, vol. 3, no. 3, pp. 325–349, 2005.

[182] A. Boukerche, *Algorithms and protocols for wireless and mobile ad hoc networks*. Wiley-IEEE Press, 2009, vol. 77.

[183] T. Clausen and P. Jaqcquet, "Optimized link state routing protocol (olsr)," *IETF-Request-for-Comments, rfc3626.txt*, 2003.

[184] C. Perkins, E. Belding-Royer, S. Das *et al.*, "Ad hoc on-demand distance vector (aodv) routing," *IETF-Request-for-Comments, rfc3561.txt*, 2003.

[185] D. Johnson, "The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4," *IETF-Request-for-Comments, rfc4728.txt*, 2007.

[186] H. Wedde and M. Farooq, "A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks," *Journal of Systems Architecture*, vol. 52, no. 8-9, pp. 461–484, 2006.

[187] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys and Tutorials*, vol. 8, no. 2, 2006.

[188] W. Znaidi, M. Minier, and J. Babau, "An Ontology for Attacks in Wireless Sensor Networks," INRIA, Tech. Rep., 2008.

[189] A. Pathan, H. Lee, and C. Hong, "Security in wireless sensor networks: issues and challenges," in *The 8th International Conference Advanced Communication Technology, 2006. ICACT 2006*, vol. 2, 2006.

[190] T. Zia and A. Zomaya, "Security Issues in Wireless Sensor Networks," in *Systems and Networks Communications, 2006. ICSNC'06. International Conference on*, 2006, pp. 40–40.

[191] N. Ahmed, S. Kanhere, and S. Jha, "The holes problem in wireless sensor networks: a survey," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 2, pp. 4–18, 2005.

[192] N. W. Group, "A Security Framework for Routing over Low Power and Lossy Networks (draft-tsao-roll-security-framework-01), Expires: March 24, 2010," IETF, Tech. Rep., 2009.

[193] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, "Ant-based load balancing in telecommunications networks," *Adaptive behavior*, vol. 5, no. 2, p. 169, 1997.

[194] G. Di Caro and M. Dorigo, "Antnet: A mobile agents approach to adaptive routing," *Artificial Intelligence Research*, vol. 9, pp. 317–365, 1997.

[195] J. Moy, "Rfc 1583: Ospf version 2," *Request For Comments, March*, 1994.

[196] D. Bertsekas and R. Gallager, "Data networks. 1992," 1992.

[197] A. Khanna and J. Zinky, "The revised arpanet routing metric," *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 4, pp. 45–56, 1989.

[198] D. Subramanian, P. Druschel, and J. Chen, "Ants and reinforcement learning: A case study in routing in dynamic networks," in *International Joint Conference on Artificial Intelligence*, vol. 15. Citeseer, 1997, pp. 832–839.

[199] Y. Lu, G. Zhao, and F. Su, "Adaptive ant-based dynamic routing algorithm," in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 3. IEEE, 2004, pp. 2694–2697.

[200] M. Gunes, U. Sorges, and I. Bouazizi, "Ara-the ant-colony based routing algorithm for manets," in *Parallel Processing Workshops, 2002. Proceedings. International Conference on.* IEEE, 2002, pp. 79–85.

[201] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 234–244, 1994.

[202] S. Marwaha, C. Tham, and D. Srinivasan, "Mobile agents based routing protocol for mobile ad hoc networks," in *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, vol. 1. IEEE, 2002, pp. 163–167.

[203] O. Hussein and T. Saadawi, "Ant routing algorithm for mobile ad-hoc networks (arama)," in *Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International*. IEEE, 2003, pp. 281–290.

[204] O. Hussein, T. Saadawi, and M. Lee, "Probability routing algorithm for mobile ad hoc networks' resources management," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 12, pp. 2248–2259, 2005.

[205] M. Islam, P. Thulasiraman, and R. Thulasiram, "A parallel ant colony optimization algorithm for all-pair routing in manets," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*. IEEE, 2003, pp. 8–pp.

[206] M. Heissenbüttel and T. Braun, "Ants-based routing in large scale mobile ad-hoc networks," *Kommunikation in verteilten Systemen (KiVS03)*, 2003.

[207] J. Baras and H. Mehta, "A probabilistic emergent routing algorithm for mobile ad hoc networks," *WiOpt03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[208] A. Jawahar, "Ant colony optimization for mobile ad-hoc networks," *Rutgers University*, 2004.

[209] X. Zheng, W. Guo, and R. Liu, "An ant-based distributed routing algorithm for ad-hoc networks," in *Communications, Circuits and Systems, 2004. ICCCAS 2004. 2004 International Conference on*, vol. 1. IEEE, 2004, pp. 412–417.

[210] G. Di Caro, "Ant Colony Optimization and its application to adaptive routing in telecommunication networks," Ph.D. dissertation, Université Libre de Bruxelles, Faculté des Sciences Appliquées, 2004.

[211] G. Di Caro, F. Ducatelle, and L. Gambardella, "Swarm intelligence for routing in mobile ad hoc networks," in *Proceedings of the IEEE Swarm Intelligence Symposium.* Citeseer, 2005.

[212] F. Ducatelle, G. Di Caro, and L. Gambardella, "An analysis of the different components of the anthocnet routing algorithm," *Ant Colony Optimization and Swarm Intelligence*, pp. 37–48, 2006.

[213] T. Ahmed, "Simulation of mobility and routing in ad hoc networks using ant colony algorithms," in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, vol. 2. IEEE, 2005, pp. 698–703.

[214] S. Kamali and J. Opatrny, "Posant: A position based ant colony routing algorithm for mobile ad-hoc networks," in *Wireless and Mobile Communications, 2007. ICWMC'07. Third International Conference on.* IEEE, 2007, pp. 21–21.

[215] B. Karp and H. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking.* ACM, 2000, pp. 243–254.

[216] F. Ducatelle, "Adaptive routing in ad hoc wireless multi-hop networks," Ph.D. dissertation, PhD thesis, Università della Svizzera Italiana, Istituto Dalle Molle di Studi sullIntelligenza Artificiale, 2007.

[217] M. Woo, N. Dung, and W. Roh, "An efficient ant-based routing algorithm for manets," in *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, vol. 2. IEEE, 2008, pp. 933–937.

[218] C. Liu, L. Li, and Y. Xiang, "Research of multi-path routing protocol based on parallel ant colony algorithm optimization in mobile

ad hoc networks," in *Fifth International Conference on Information Technology: New Generations*. IEEE, 2008, pp. 1006–1010.

[219] R. Asokan, A. Natarajan, and C. Venkatesh, "Ant based dynamic source routing protocol to support multiple quality of service (qos) metrics in mobile ad hoc networks," *International Journal of Computer Science and Security (IJCSS)*, vol. 2, no. 3, p. 48, 2008.

[220] W. Yu, G. Zuo, and Q. Li, "Ant colony optimization for routing in mobile ad hoc networks," in *Machine Learning and Cybernetics, 2008 International Conference on*, vol. 2. IEEE, 2008, pp. 1147–1151.

[221] M. Sivajothi and E. Naganathan, "An ant colony based routing protocol to support multimedia communication in ad hoc wireless networks," *IJCSNS*, vol. 8, no. 7, p. 21, 2008.

[222] J. Wang, E. Osagie, P. Thulasiraman, and R. Thulasiram, "Hopnet: A hybrid ant colony optimization routing algorithm for mobile ad hoc network," *Ad Hoc Networks*, vol. 7, no. 4, pp. 690–705, 2009.

[223] Z. Haas, M. Pearlman, and P. Samar, "The zone routing protocol (zrp) for ad hoc networks," *IETF Internet Draft*, 2002.

[224] R. Attia, R. Rizk, and M. Mariee, "A hybrid multi-path ant qos routing algorithm for manets," in *Wireless and Optical Communications Networks, 2009. WOCN'09. IFIP International Conference on*. IEEE, 2009, pp. 1–5.

[225] D. Kadono, T. Izumi, F. Ooshita, H. Kakugawa, and T. Masuzawa, "An ant colony optimization routing based on robustness for ad hoc networks with gpss," *Ad Hoc Networks*, vol. 8, no. 1, pp. 63–76, 2010.

[226] Y. Ko and N. Vaidya, "Location-aided routing (lar) in mobile ad hoc networks," *Wireless Networks*, vol. 6, no. 4, pp. 307–321, 2000.

[227] Y. Zhang, L. Kuhn, and M. Fromherz, "Improvements on ant routing for sensor networks," *Ant Colony, Optimization and Swarm Intelligence*, pp. 289–313, 2004.

[228] T. Camilo, C. Carreto, J. Silva, and F. Boavida, "An energy-efficient ant-based routing algorithm for wireless sensor networks," *Ant Colony Optimization and Swarm Intelligence*, pp. 49–59, 2006.

[229] G. Chen, T. Guo, W. Yang, and T. Zhao, "An improved ant-based routing protocol in wireless sensor networks," in *2006 International Conference on Collaborative Computing: Networking, Applications and Worksharing.* IEEE, 2006, p. 65.

[230] S. Iyengar, H. Wu, N. Balakrishnan, and S. Chang, "Biologically inspired cooperative routing for wireless mobile sensor networks," *Systems Journal, IEEE*, vol. 1, no. 1, pp. 29–37, 2007.

[231] R. GhasemAghaei, A. Rahman, W. Gueaieb, and A. El Saddik, "Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks," in *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE.* IEEE, 2007, pp. 1–6.

[232] X. Wang, Q. Li, N. Xiong, and Y. Pan, "Ant colony optimization-based location-aware routing for wireless sensor networks," *Wireless Algorithms, Systems, and Applications*, pp. 109–120, 2008.

[233] X. L. Ren, H. W. Liang, and Y. Wang, "Multipath routing based on ant colony system in wireless sensor networks," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 3. IEEE, 2008, pp. 202–205.

[234] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking.* ACM, 2000, pp. 56–67.

[235] Y. Wen, Y. Chen, and M. Pan, "Adaptive ant-based routing in wireless sensor networks using energy* delay metrics," *Journal of Zhejiang University-Science A*, vol. 9, no. 4, pp. 531–538, 2008.

[236] S. Tarannum, V. Anitha, A. Priya, R. Adrakatti, L. Nalini, K. Venugopal, and L. Patnaik, "Self-healing antchain for increasing lifespan in wireless sensor networks," in *Information and Communica-*

tion Technology in Electrical Sciences (ICTES 2007), 2007. ICTES. IET-UK International Conference on. IET, 2008, pp. 908–915.

[237] S. Okdem and D. Karaboga, "Routing in wireless sensor networks using an ant colony optimization (aco) router chip," *Sensors*, vol. 9, no. 2, pp. 909–921, 2009.

[238] X. Hui, Z. Zhigang, and Z. Xueguang, "A novel routing protocol in wireless sensor networks based on ant colony optimization," in *2009 International Conference on Environmental Science and Information Application Technology.* IEEE, 2009, pp. 646–649.

[239] L. Keong, L. Huan, and P. Yi, "An efficient and reliable routing protocol for wireless sensor networks," in *Proceedings of the First International IEEE WoWMoM Workshop on Autonomic Communications and Computing (ACC'05)-Volume 02.* IEEE Computer Society, 2005, pp. 512–516.

[240] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "Speed: A stateless protocol for real-time communication in sensor networks," in *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on.* IEEE, 2003, pp. 46–55.

[241] K. Saleem, N. Fisal, S. Hafiza, S. Yusof, S. Kamilah, A. Rashid *et al.*, "Ant based self-organized routing protocol for wireless sensor network," *International Journal of Communication Networks and Information Security*, vol. 1, no. 2, pp. 42–46, 2009.

[242] K. Saleem, N. Fisal, S. Hafizah, S. Kamilah, and R. Rashid, "A self-optimized multipath routing protocol for wireless sensor networks," *International Journal of Recent Trends in Engineering*, vol. 2, no. 1, 2009.

[243] J. Wang, J. Xu, and M. Xiang, "Eaqr: an energy-efficient aco based qos routing algorithm in wireless sensor networks," *Chinese Journal of Electronics*, vol. 18, no. 1, pp. 113–116, 2009.

[244] K. Saleem, N. Fisal, M. Baharudin, S. Hafizah, S. Kamilah, and R. Rashid, "Ant colony inspired self-optimized routing protocol based on cross layer architecture for wireless sensor networks," in

Proceedings of the 14th WSEAS international conference on Communications. World Scientific and Engineering Academy and Society (WSEAS), 2010, pp. 178–183.

[245] A. Ali, L. Latiff, and N. Fisal, "Simulation-based real-time routing protocol with load distribution in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 10, no. 7, pp. 1002–1016, 2010.

[246] X. Jiang and B. Hong, "Aco based energy-balance routing algorithm for wsns," *Advances in Swarm Intelligence*, pp. 298–305, 2010.

[247] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*. IEEE, 2002, pp. 10–pp.

[248] J. Yang, M. Xu, W. Zhao, and B. Xu, "A multipath routing protocol based on clustering and ant colony optimization for wireless sensor networks," *Sensors*, vol. 10, no. 5, pp. 4521–4540, 2010.

[249] A. Manjeshwar and D. Agrawal, "Teen: a routing protocol for enhanced efficiency in wireless sensor networks," in *Parallel and Distributed Processing Symposium., Proceedings 15th International*. IEEE, 2001, pp. 2009–2015.

[250] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 4, pp. 11–25, 2001.

[251] A. Okazaki and A. Frohlich, "Ad-zrp: Ant-based routing algorithm for dynamic wireless sensor networks," in *Telecommunications (ICT), 2011 18th International Conference on*. IEEE, 2011, pp. 15–20.

[252] R. Huang, J. Zhu, X. Yu, and G. Xu, "The ant-based algorithm for the optimal many-to-one routing in sensor networks," in *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, vol. 3. IEEE, 2006, pp. 1532–1536.

[253] B. McBride, C. Scoglio, and S. Das, "Distributed biobjective ant colony algorithm for low cost overlay network routing," *Proceedings of ICAI 2006*, 2006.

[254] H. Wang, Z. Shi, A. Ge, and C. Yu, "An optimized ant colony algorithm based on the gradual changing orientation factor for multi-constraint qos routing," *Computer Communications*, vol. 32, no. 4, pp. 586–593, 2009.

[255] K. Sim and W. Sun, "Ant colony optimization for routing and load-balancing: survey and new directions," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 33, no. 5, pp. 560–572, 2003.

[256] D. Pinto, B. Barán, and R. Fabregat, "Multi-objective multicast routing based on ant colony optimization," in *Proceeding of the 2005 conference on Artificial Intelligence Research and Development*, 2005, pp. 363–370.

[257] J. Crichigno and B. Barán, "Multiobjective multicast routing algorithm for traffic engineering," in *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on.* IEEE, 2004, pp. 301–306.

[258] B. Baran and R. Sosa, "A new approach for antnet routing," in *Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on.* IEEE, 2000, pp. 303–308.

[259] B. Baran and D. Pinto, "Multiobjective multicast routing with ant colony optimization," *IFIP Advances in Information and Communication Technology (AICT)*, vol. 213, no. 213, pp. 101–115, 2011.

[260] OMNeT++ Community. (2011, Dec.) OMNeT++. [Online]. Available: http://www.omnetpp.org/

[261] MiXiM developers. (2010, May) MiXiM project. [Online]. Available: http://mixim.sourceforge.net/index.html

[262] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. Haneveld, T. Parker, O. Visser, H. Lichte, and S. Valentin, "Simulating wireless and mobile networks in omnet++ the mixim vision," in *Pro-*

ceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–8.

[263] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference (ESM'2001)*, 2001, pp. 319–324.

[264] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops table of contents*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) ICST, Brussels, Belgium, Belgium, 2008.

[265] Simulcraft, Inc. (2011, Dec.) OMNEST. [Online]. Available: http://www.omnest.com/

[266] Telecommunication Networks Group (TKN), TU Berlin. (2010, May) Mobility Framework for OMNeT++. [Online]. Available: http://mobility-fw.sourceforge.net/

[267] Research Group Computer Networks, Universität Paderborn. (2010, May) Chsim: Wireless channel simulator for OMNeT++. [Online]. Available: http://www.cs.uni-paderborn.de/en/fachgebiete/research-group-computer-networks/projects/chsim.html

[268] TU Delft and University of Twente. (2010, May) MAC Simulator and Positif for OMNeT++. [Online]. Available: http://www.consensus.tudelft.nl/software.html

[269] IEEE Computer Society - Wireless LAN Working Group . (2011, Aug.) IEEE 802.11: WIRELESS LOCAL AREA NETWORKS (LANs). [Online]. Available: http://standards.ieee.org/about/get/802/802.11.html

[270] IEEE Computer Society - 802.15.4 Task Group. (2011, Aug.) IEEE 802.15: WIRELESS PERSONAL AREA NETWORKS (PANs) . [Online]. Available: http://standards.ieee.org/about/get/802/802.15.html

[271] T. Handel and M. Sandford, "Hiding data in the osi network model," in *Information Hiding*. Springer, 1996, pp. 23–38.

[272] Crossbow Technology, Inc. (2011, Aug.) MICA Mote Datasheet. [Online]. Available: http://stomach.v2.nl/docs/Hardware/DataSheets/Sensors/MICA_data_sheet.pdf

Index