Jiří Trdlička

On Distributed and Real-Time Routing in Sensor Networks

May 2011

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Control Engineering

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

On Distributed and Real-Time Routing in Sensor Networks

Doctoral Thesis

Jiří Trdlička

Prague, June 2011

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics

Supervisor: ***Doc. Dr. Ing. Zdeněk Hanzálek***

Jiří Trdlička
www.trdlicka.cz
jiri@trdlicka.cz

To my parents, family and friends.

Acknowledgements

First of all, I would like to thank my adviser Zdeněk Hanzálek for his guidance and patience during my work. Without Zdeněk this work would never be created. I would also like to thank Mikael Johansson for his inspiration and time I have spend at the KTH in Stockholm. It was a great time.

I am grateful to my family and all my friends for being here and for their persistent believe that I will finish this work one day. I am especially grateful to my friend Petr Macejko, who arranged two virtual machines for me. Without him, the presented experiments would take ages to compute.

This work has been supported by the Ministry of Education of the Czech Republic under Research Programme MSM6840770038, by the Czech Science Foundation under Research Programme 5 P103/10/0850 and by the European Commission under project FRESCOR IST 034026.

Czech Technical University in Prague *Jiří Trdlička*
June 2011

Nomenclature

General:

$\mathcal{R}^L$	Set of all real vectors with L components
$\mathcal{R}_+^L$	Set of all non-negative real vectors with L components
$\mathcal{Z}_+^0$	Set of all non-negative integer numbers
$\vec{x}^T$	Transposition of vector $\vec{x}$
$\vec{0}$	Column vector with all elements equal to zero
I	Identity matrix
$[\vec{x}]^+$	Non-negative value in each component of vector $\vec{x}$
$[A\vec{x}_k - \vec{b}]_i$	i-th component of the resulting vector
$\max_{\vec{\theta}}$	Maximization with respect to variable $\vec{\theta}$
$\min_{\vec{x}}$	Minimization with respect to variable $\vec{x}$
∇L	Gradient of L
$\nabla_x L$	Gradient with respect to $\vec{x}$
$\nabla^2 L$	Hessian matrix of L
$\nabla_{xy}^2 L$	Hessian matrix of L with respect to $\vec{x}$ and $\vec{y}$

All vectors are a column vectors and are marked as $\vec{x}$

Multi-commodity flow:

$\mathcal{O}(n)$	Set of links leaving node n
$\mathcal{I}(n)$	Set of links incoming to node n
A^+	Incidence matrix of the incoming links
A^-	Incidence matrix of the outgoing links
$\vec{x}^{(m)}$	Flow vector of communication demand m for all links
$\vec{s}_{in}^{(m)}$	Vector of flow for communication demand m incoming into the network
$\vec{s}_{out}^{(m)}$	Vector of flow for communication demand m leaving the network
$\vec{t}$	Vector of total flow over all communication demands
D	Capacity constraints matrix

$\vec{\mu}$	Capacity constraints values (e.g. link or node communication capacities)
$\vec{c}$	Communication costs per transmitted data unit for all links
$\mathcal{M}$	Set of all communication demands
M	Number of communication demands
m	Index for communication demand ($m \in \mathcal{M}$)
N	Number of nodes in the network
n	Index for node $n = 1, \ldots, N$
L	Number of links in the network
l	Index for link $l = 1, \ldots, L$
l^+	End node of link l
l^-	Start node of link l
$x_l^{(m)}$	l-th component of $\vec{x}^{(m)}$, i.e. flow in link l
$s_{out,n}^{(m)}$	n-th component of $\vec{s}_{out}^{(m)}$, i.e. flow leaving the network in node n

Real-time routing:

$d^{(m)}$	Deadline for communication demand m (i.e. maximum communication delay)
w	Index for communication delay (number of communication hops)
$\vec{x}^{(m,w)}$	Flow vector of communication demand m with delay w
$\vec{s}_{out}^{(m,w)}$	Vector of flow for communication demand m leaving the network with delay w
$\mathcal{M}'$	Set of non-real-time communication demands
m'	Index of non-real-time communication demands

Periodic messages routing:

$p^{(m)}$	Transmission period for communication demand m
P	Network period defined as a least common multiple of periods $p^{(m)} \quad \forall m \in \mathcal{M}$

$o^{(m)}$	Transmission offset for communication demand m (start time within the network period P)
$\mathcal{T}(q)$	Set of pairs (m, w) routed through network in time q (see definition (2.4.4))
$\vec{t}^{(q)}$	Vector of total flow over all communication demands in time q
$\vec{y}^{(m,w)}$	Flow vector of communication demand m with delay w which is not transmitted between nodes

Distributed routing algorithms:

α	Constant step size for gradient algorithms
ε	Constant proximal-point parameter
$\vec{x}_k$	Vector $\vec{x}$ in k-th iteration of an algorithm
$\vec{\theta}^{(m)}$	Dual variable corresponding to equality constraints
$\vec{\lambda}$	Dual variable corresponding to inequality constraints
$\vec{x}'', \vec{z}'', \vec{s}''$	Proximal-point variables
L	Lagrangian function
W	Dual function
U	Dual problem
$\vec{x}$	Generalized flow variables (see (3.3.3))
$\vec{\theta}$	Generalized dual variables (see (3.3.14))
$\vec{b}$	Generalized constraints vector (see (3.3.2))
$\vec{z}$	Slack variables (see Section 3.3.1)
$\vec{s}^{(m,w)}$	the same definition as $\vec{s}_{out}^{(m,w)}$ (used in Chapter 4 for more compact description)

Abbreviations

TDMA	Time Division Multiple Access
CSMA	Carrier Sense Multiple Access
IEEE 802.15.4	Standard for low-rate wireless networks
GTS	Guaranteed Time Slots
MMCF	Minimum-Cost Multi-Commodity Flow Problem
NUM	Network Utility Maximization
NP	Non-Deterministic Polynomial-Time
TLDRA	Two Loops Distributed Routing Algorithm
OLDRAi	One Loop Distributed Routing Algorithm with Incremental flow update
OLDRAo	One Loop Distributed Routing Algorithm with Optimal flow update

On Distributed and Real-Time Routing in Sensor Networks

Ing. Jiří Trdlička
Czech Technical University in Prague, 2011

Thesis Advisor: Doc. Dr. Ing. Zdeněk Hanzálek

The thesis is focused on in-network distributed and real-time routing algorithms for multi-hop sensor networks with linear cost functions and constant communication delays. The work aims mathematically derived algorithms which are based on the convex optimization theory and which are capable of computing an exact optimal solution e.g. in terms of the energy consumption. The work consists of three parts.

The first part is focused on centralized algorithms for real-time routing in sensor networks. Two routing algorithms are developed in this part. The first algorithm addresses problem with continuous data streams with a real-time constraints on communication delay. The second algorithm addresses problem, where the real-time data are send in messages with transmission periods significantly bigger than one hop communication delay. The both algorithms are based on a minimum-cost multi-commodity network flow model and use a network replication to include the real-time constraints. Solved by Linear Programming, they exhibit a very good performance, as is shown in our experiments. Surprisingly, the performance does not degrade even in the presence of an integral flow constraint, which makes the problems NP hard.

The second part of this work is focused on in-network distributed energy optimal routing algorithms for non-real-time data flow with linear objective functions. Three distributed routing algorithms are mathematically derived in this part: *Two Loops Distributed Routing Algorithm* (TLDRA), *One Loop Distributed Routing Algorithm with Incremental flow update* (OLDRAi) and *One Loop Distributed Routing Algorithm with Optimal flow update* (OLDRAo). The algorithms are based on the proximal-point method and the dual decomposition of convex optimization problem. The algorithms compute an exact energy optimal routing in the network without any central node

or the knowledge about the whole network structure, using only peer-to-peer communication between neighboring nodes. In contrast to other works in this area, the presented approach is not limited to strictly convex objective functions and it even handles linear objective functions which makes the algorithm and proof of its convergence more difficult. Proofs of the algorithms convergence are presented.

The third part of this work derives a distributed routing algorithm for real-time data streams with constant communication delays. The algorithm is based on the OLDRAo and on the routing algorithm for continuous data streams with real-time constraints from the two previous parts.

The behaviors of all presented algorithms have been evaluated on benchmarks for energy optimal routing in multi-hop sensor networks, using Matlab.

Goal and Objectives

The main goal of this thesis is to bring a new knowledge into the area of in-network distributed and real-time routing algorithms for multi-hop sensor networks. The work aims mathematically derive algorithms which are based on the convex optimization theory and which are capable of computing an exact optimal solution e.g. in terms of the energy consumption.

The objectives of the thesis are:

1. Develop a centralized algorithm for real-time routing in multi-hop sensor networks.
2. Derive in-network distributed routing algorithm for data flow in multi-hop sensor networks.
3. Derive in-network distributed routing algorithm for real-time data flow in multi-hop sensor networks.
4. Evaluate behavior of all derived algorithms on benchmarks for energy optimal routing in multi-hop sensor networks.

Contents

Goals and Objectives **xiii**

1 Introduction **1**
1.1 Sensor Networks 2
1.2 Convex Optimization 4
1.3 Outline and Contribution 5

2 Centralized Algorithms for Real-Time Routing **7**
2.1 Introduction 7
2.1.1 Problem Formalization 8
2.1.2 Related Works 8
2.2 Multi-Commodity Network Flow Model 10
2.3 Routing of Continuous Data Flow with Real-Time Constraints 14
2.3.1 Mathematical Model 15
2.3.2 Numerical Experiments 20
2.4 Routing of Periodic Messages with Real-Time Constraints . . 30
2.4.1 Mathematical Model 31
2.4.2 Numerical Experiments 36
2.5 Summary 43

3 Distributed Routing Algorithm **45**
3.1 Introduction 45
3.1.1 Related Works 46
3.1.2 Multi-Commodity Network Flow Model 47
3.1.3 Proximal-Point Method 48
3.2 Two Loops Distributed Routing Algorithm 49
3.2.1 Mathematical Derivation 49

3.2.2 Experiments 56
3.3 One Loop Distributed Routing Algorithm with Incremental flow update 65
3.3.1 Mathematical Derivation 65
3.3.2 Experiments 70
3.3.3 Proof of the Algorithm Convergence 75
3.4 One Loop Distributed Routing Algorithm with Optimal flow update 79
3.4.1 Mathematical Derivation 79
3.4.2 Experiments 84
3.4.3 Proof of the Algorithm Convergence 91
3.4.4 Open Issues 95
3.5 Summary 98

4 Distributed Algorithms for Real-Time Routing 101
4.1 Introduction 101
4.1.1 Related Works 101
4.2 Mathematical Derivation 102
4.2.1 Dual Problem 104
4.2.2 Dual Gradient Algorithm 105
4.2.3 Variables Initialization 108
4.3 Experiments 109
4.3.1 Example 109
4.3.2 Algorithm Convergence 111
4.3.3 Number of Iterations 112
4.3.4 Network Change 113
4.4 Proof of the Algorithm Convergence 114
4.5 Summary 117

5 Conclusions 119
5.1 Summary and Contributions 119
5.2 Open Issues and Future Work 121

Bibliography 128

List of Author's Publications 129

List of Figures

1.1.1 Multi-hop communication 3

2.2.1 Example: Graph of basic network. 11
2.2.2 Example: An optimal data flow routing with capacity constraints. 13
2.3.1 Example: An optimal data flow routing with capacity and deadline constraints. 16
2.3.2 Intuitive presentation for the graph replication. 17
2.3.3 Data collection problem: Network structure 21
2.3.4 Data collection problem: Optimal real-time routing 22
2.3.5 Data collection problem: Optimal real-time routing in node-delay space 23
2.3.6 Experiment: Time complexity when progressively decreasing the link capacities - fragmented flow 25
2.3.7 Experiment: Time complexity when progressively decreasing the link capacities - integral flow 25
2.3.8 Experiment: Comparison of computation times for fragmented and integral flow 28
2.3.9 Experiment: Time complexity in relation to the number of communication demands - fragmented flow 28
2.3.10 Experiment: Time complexity in relation to the number of communication demands - integral flow 29
2.4.1 Intuitive presentation for the graph replication. 33
2.4.2 Gantt Charts for the network links 35
2.4.3 Data Collection Problem: Optimal real-time routing 37
2.4.4 Data Collection Problem: Optimal real-time routing in node-delay space 38

2.4.5 Data Collection Problem: Optimal real-time routing in node-time space . . . 38
2.4.6 Experiment: Time complexity in relation to number of nodes - fragmented flow . . . 39
2.4.7 Experiment: Time complexity in relation to number of nodes - integral flow . . . 40
2.4.8 Experiment: Time complexity in relation to network period - fragmented flow . . . 41

3.2.1 TLDRA: Iteration algorithm . . . 51
3.2.2 TLDRA: Optimal data flow routing . . . 57
3.2.3 TLDRA: Progress of Lagrangian function (3.2.2) . . . 58
3.2.4 TLDRA: Number of proximal-point iterations . . . 59
3.2.5 TLDRA: Number of gradient iterations in 1st proximal-point iteration . . . 59
3.2.6 TLDRA: Number of gradient iterations in relation to index of proximal-point iteration . . . 60
3.2.7 TLDRA: Lagrangian progress for incomplete gradient iterations. . . . 61
3.2.8 TLDRA: Number of proximal-point iterations in case of incomplete gradient iteration . . . 62
3.2.9 TLDRA: Number of proximal-point iterations for node capacities . . . 63
3.2.10 TLDRA: Number of gradient iterations in 1st proximal-point iteration for node capacities . . . 64
3.2.11 TLDRA: Number of gradient iterations in relation to index of proximal-point iteration for node capacities . . . 64
3.3.1 OLDRAi: Optimal data flow routing . . . 71
3.3.2 OLDRAi: Progress of the Lagrangian function (3.3.5) . . . 72
3.3.3 OLDRAi: Algorithm convergence with random starting points. 73
3.3.4 OLDRAi: Number of iterations in relation to the number of nodes. . . . 74
3.3.5 OLDRAi: Number of iterations in relation to the number of communication demands. . . . 74
3.4.1 OLDRAo: Optimal data flow routing . . . 85
3.4.2 OLDRAo: Progress of the Lagrangian function (3.3.5) . . . 86
3.4.3 OLDRAo: Algorithm convergence . . . 87

3.4.4 OLDRAo: Number of iterations in relation to number of nodes 88
3.4.5 OLDRAo: Number of iterations in relation to number of communication demands 89
3.4.6 OLDRAo: Algorithm convergence for network change simulation. 90
3.4.7 OLDRAo: Algorithm convergence 97

4.3.1 Distributed, real-time routing: Optimal data flow routing . 110
4.3.2 Distributed, real-time routing: Optimal data flow routing in node-delay space. 110
4.3.3 Distributed, real-time routing: Algorithm convergence . . . 111
4.3.4 Distributed, real-time routing: Number of iterations in relation to number of nodes 112
4.3.5 Distributed, real-time routing: Algorithm convergence for network change simulation. 113

List of Tables

2.3.1 Computation times for 5 communication demands [s] 26
2.3.2 Computation times for 20 communication demands [s] 27

3.2.1 TLDRA: Dual Gradient Algorithm 52
3.2.2 TLDRA: Distributed Routing Algorithm executed in node n 54
3.3.1 OLDRAi: Distributed Routing Algorithm executed in node n 69
3.4.1 OLDRAo: Distributed Routing Algorithm executed in node n 83

4.2.1 Distributed, Real-Time Routing Algorithm 107

Chapter 1

Introduction

The communication systems and networks are one of the most important phenomena of the today's world. The almost whole word is connected and the communication systems are still evolving. Many communication systems are well known by the general public, like the Internet, Cell phones networks, satellite networks etc. On the other side, many communication systems are not so well known, but they affect our lives not less, maybe more. There are e.g. the industrial communication networks, car communication systems, building control systems etc.

The development in the recent years is slowly changing the configuration and the control of the communication systems. In many areas, the networks control is changing from the centralized systems, where one master device is controlling the whole network, to distributed control systems, where all the devices (or subset of the devices) are equal and they are controlling the network cooperatively. Simultaneously, many communication systems are changing into so called multi-hop or switched communication, which increases the data throughput and the communication speed. Examples of such systems are e.g. the industrial protocol Profinet, the switched Ethernet for the Internet connection or the Sensor Networks.

In this chapter, we briefly mention the sensor networks, routing problem and convex optimization, which are basis of this work. At the end of this chapter a short outlines and work contributions are mentioned.

1.1 Sensor Networks

The sensor networks is a technology of future, which can significantly change the world as we know it. (for more detail see e.g. [Edga 03]) The rise of this technology is based on the recent development in the micro-electronic area. The main idea is that we are able to create a small and cheap devices, which are equipped with a microprocessor, a radio transceiver, some components to interact with the environment (typically sensors) and an energy source (typically battery). The sensor networks are meant to consist of hundreds or thousands of devices (usually called *motes* or *nodes*), which cooperatively work on given tasks. Primarily, the development was motivated by military applications such as battlefield monitoring. Today, they are used in many industrial and consumer applications, such as industrial process monitoring and control, environment monitoring, home automation, traffic control etc.

The main challenge in the sensor networks area is the strictly resource limitation such as limited energy, limited communication bandwidth or limited computational power. The limited resources are the main reason for the optimization in all levels of the research. In this work we focus on the communication problems, especially on routing algorithms.

Routing in Sensor Networks

To save energy during the communication the nodes in sensor networks use multi-hop communication. The multi-hop communication means, that the messages are routed to their destinations through some other nodes, in order to decrease the transmission distance and save the energy (see illustration in Figure 1.1.1).

The routing algorithms can be categorized as on-line or off-line algorithms. The on-line algorithms choose the massage routing direction according to actual situation in the time of the message transmission. The off-line algorithms use precomputed routing rules. The on-line algorithms are usually more simple, more robust in the case of network damage and need less memory. On the other side the off-line algorithms are usually more effective in terms of energy consumption and communication bandwidth utilization [Karl 05].

A different categorization of the routing algorithms, important for this work, is into classes of centralized and distributed algorithms. The central-

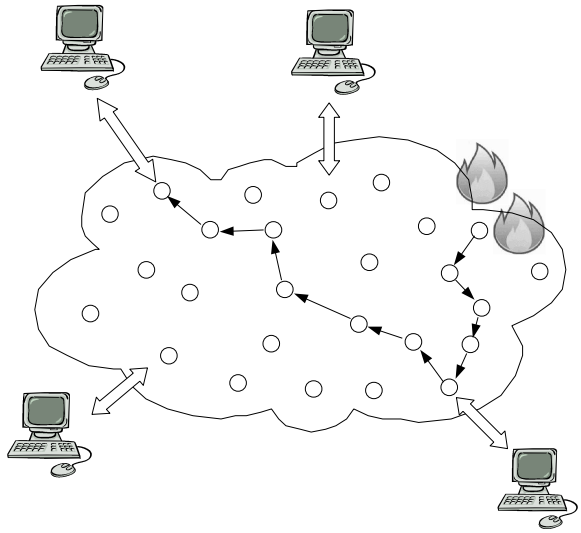


Fig. 1.1.1: Multi-hop communication

ized algorithms are computed in a central computational point and then the routing rules are distributed into the network (e.g. [Chen 10]). The distributed algorithms are based on the communication and cooperation of the individual nodes. The distributed algorithms are more robust against network damage. On the other side, to achieve the same optimality as the centralized algorithms they are more complicated and they are the challenges of the current research. The off-line routing algorithms can be both distributed or centralized.

Real-Time Routing

In many applications in sensor networks area such as industrial process monitoring and control or fire detection, which are time-critical, a real-time communication is required. The objective of the real-time communication is to ensure that all the routed messages are delivered to their destinations before their deadlines.

There are two main methods to model the communication delay in the sensor networks. A queuing delay, which is more suitable for networks us-

ing CSMA and similar medium access mechanism, where the communication delay is a function of data flow volume (e.g.[Piro 04]). A constant communication delay, which is more suitable for networks with TDMA based medium access mechanism, where the communication delay depends on the number of communication hops and time slots schedule [Chen 10]. In this work, we focus on the constant communication delay, which is often used in the industry communication systems.

1.2 Convex Optimization

Convex optimization is a special class of mathematical optimization problems, which includes least-squares and Linear Programming problems. It is widely used in many areas such as statistic and finance, automatic control, estimations and signal processing, communications and networks, electronics, modeling etc. There are great advantages to recognizing or formulating a problem as a convex optimization problem. The most basic advantage is that the problem can then be solved, very reliably and efficiently, using interior-point methods or other special methods for convex optimization. These solution methods are reliable enough to be embedded in a computer-aided design or analysis tool, or even a real-time reactive or automatic control system. There are also theoretical or conceptual advantages of formulating a problem as a convex optimization problem. The associated dual problem, for example, often has an interesting interpretation in terms of the original problem, and sometimes leads to an efficient or distributed method for solving it. [*Cited: Stephen Boyd, Lieven Vandenberghe* 2004 in [Boyd 04]]

For detailed description about convex optimization see e.g. [Boyd 04, Bert 99]. We define several terms for future use according to [Boyd 04] in this section.

A set $\mathcal{C}$ is convex if the line segment between any two points in $\mathcal{C}$ lies in $\mathcal{C}$, i.e., if for any $\vec{x}, \vec{y} \in \mathcal{C}$ and any ξ with $0 \leq \xi \leq 1$, we have

$$\xi\vec{x} + (1-\xi)\vec{y} \in \mathcal{C} \tag{1.2.1}$$

A function $f : \mathcal{R}^n \rightarrow \mathcal{R}$ is convex if $\boldsymbol{dom}f$ is a convex set and if for all $\vec{x}, \vec{y} \in \boldsymbol{dom}f$, and ξ with $0 \leq \xi \leq 1$, we have

$$f(\xi\vec{x} + (1-\xi)\vec{y}) \leq \xi f(\vec{x}) + (1-\xi)f(\vec{y}) \tag{1.2.2}$$

A function $f : \mathcal{R}^n \to \mathcal{R}$ is strictly convex if $\boldsymbol{domf}$ is a convex set and if for all $\vec{x}, \vec{y} \in \boldsymbol{domf}$, and ξ with $0 \leq \xi \leq 1$, we have

$$f(\xi\vec{x} + (1-\xi)\vec{y}) < \xi f(\vec{x}) + (1-\xi)f(\vec{y}) \tag{1.2.3}$$

And the general convex optimization problem is:

$$\begin{array}{rcll} \min\limits_{x} & f_0(\vec{x}) & & \\ \textbf{subject to:} & & & \\ f_i(\vec{x}) & \leq & 0 & i = 1, \ldots, L \\ \vec{a}_i^T \vec{x} & = & b_i & i = 1, \ldots, N \end{array} \tag{1.2.4}$$

where $f_0, \ldots, f_m$ are convex functions.

1.3 Outline and Contribution

This thesis is focused on the real-time and distributed routing problems in the sensor networks. The work is divided into three parts. Each part consists of problem introduction with related works, mathematical derivation of the presented problem, experimental section and summary.

The first part, which is presented in Chapter 2, is focused on centralized algorithms for real-time routing in sensor networks. Two routing algorithms, which are based on Linear Programming theory, are developed in this chapter. The first algorithm addresses problem with continuous data streams (flow) with a real-time constraints on communication delay (Section 2.3). The second algorithm addresses problem, where the real-time data are send in messages with transmission periods significantly bigger than one hop communication delay (Section 2.4).

The second part of this work, which is presented in Chapter 3, is focused on distributed routing algorithms for non-real-time data flow. Three distributed routing algorithms are mathematically derived in this chapter: *Two Loops Distributed Routing Algorithm* (TLDRA) in Section 3.2, *One Loop Distributed Routing Algorithm with Incremental flow update* (OLDRAi) in Section 3.3 and *One Loop Distributed Routing Algorithm with Optimal flow update* (OLDRAo) in Section 3.4. The algorithms are based on the dual decomposition of a convex optimization problem and proofs of their convergence are presented.

The third part of this work, which is presented in Chapter 4, is focused on distributed routing algorithm for real-time data flow. The algorithm is based on the OLDRAo algorithm and on the routing algorithm for continuous data streams (flow) with real-time constraints from previous two chapters.

The main contributions of this thesis are:

1. Formulation of a real-time multi-commodity network flow problem and its solution by Linear Programming based on graph replication.

2. Discovery of a surprisingly good Integer Linear Programming performance for the above mentioned problem with an integral data flow constraint, which makes the problem NP hard.

3. Introduction of a new distributed algorithm based on dual decomposition of routing problem formulated in node-link form.

4. Presentation of novel approach to distribute the linear optimization problem by dual decomposition as an in-network distributed algorithm. (Other works using the dual decomposition on the routing problems are limited to strictly convex objective functions and fail in the linear case.)

5. Introduction of new mathematically derived, distributed algorithm for energy optimal real-time routing based on network replication and dual decomposition.

6. Performance evaluation of all presented algorithms on benchmarks for energy optimal routing in sensor networks.

Chapter 2

Centralized Algorithms for Real-Time Data Routing in Sensor Networks

2.1 Introduction

Our work in this chapter is focused on centralized algorithms for data flow routing through the multi-hop sensor network, where all data has to be delivered to the destinations in time. The objective is to optimize the energy consumption for the data transfer and we assume the following constraints: link capacities, node capacities and different deadlines for each value sensed. All data has to be delivered before their deadlines. We assume a TDMA (Time Division Multiple Access) protocol (e.g. GTS allocation in IEEE 802.15.4 [Koub 06a, Koub 06b]) which ensures collision-free communication and causes communication delay. This approach is quite often in industry and time-critical systems. Due to the TDMA mechanism assumed, the worst-case delay from the source node to the sink node is the sum of the particular delays for each of the hops, assumed to be an integer (derived from the parameters like TDMA period, worst-case execution time of the communication stack...). In a particular setting, we may assume a unit hop delay (the same TDMA period, negligable influence of the transmission delay on the physical layer...). In this work we assume the unit hop delay (the deadlines are expressed as the number of communication hops between devices) which is very

transparent for the reader and furthermore it can be generalized to integer delays that may differ for each hop [Trdl 07].

In Section 2.2 of this chapter, we briefly introduce the multi-commodity network flow problem which is used during the whole thesis. Section 2.3 is focused on continuous flow problem which is well suitable for data streams or in the case of periodically sent data with a transmission period close to one hop communication delay. In Section 2.4 we focused on the case where the transmission period is significantly bigger than the hop communication delay. Sections 2.3 and 2.4 are concluded with extensive experimental simulations to evaluate the algorithm behavior and performance.

2.1.1 Problem Formalization

We model the data flow as a multi-commodity network flow problem in node-link form, where each commodity represents a different communication demand in the network. The network topology is represented by a directed graph where the nodes represent the devices and the oriented edges represent the oriented communication links between the devices. The communication capacities of the network and the communication cost (e.g. energy consumption) are associated with the edges of the graph. Each communication demand has source nodes, sink nodes (multi-source, multi-sink) and corresponding volumes. (i.e. problems such as data gathering can be defined in this way) We extend this problem by a real-time constraint, i.e. the problem is feasible when each data flow from the source node to the sink node obeys the deadline of the corresponding communication demand.

To solve the real-time version of the energy optimal routing, we adapt the model to solve the minimum-cost multi-commodity flow problem (MMCF) [Piro 04, Bert 98] with additional constraints on the number of communication hops. The number of communication hops corresponds to the worst-case of the communication delay in the network and its constraints ensure the real-time behavior.

2.1.2 Related Works

Traditionally, routing problems for data networks are often formulated as linear or convex multi-commodity network flow routing problems e.g.

[Bert 04, Chia 05, Piro 04, Bulb 07] for which many efficient solution methods exist [Bert 98, Ouor 00, Boyd 04, Joha 06b].

In [Xiao 04], the multi-commodity problem formulation is used for simultaneous routing and resource allocation (e.g. node related bandwidth), which finds more efficient routing than the case of separated algorithms. One of the advantages of this method is that several objective functions and constraints can be put together. Using the same underlying model, we can easily combine the solution of different works focused on partial problems.

Several papers have been performed in the area of real-time routing in multi-hop wireless sensor networks. In [He 03], a well known soft real-time communication protocol SPEED, is presented. The protocol uses the speed of the message propagation to set priorities of the messages. Several works use relation between message propagation speed and transmitting energy to balance trade-off between energy consumption and communication delay. In [Chip 06], a protocol called RPAR is presented, in [Brav 09] a protocol called EDEM is presented or in [Xian 10] a distributed cross-layer routing mechanism is presented.

There are papers which modify the geographical routing into time aware form [Ahme 08, Chen 06]. In [Li 07] an algorithm based on direct diffusion which balances node energy utilization is presented. In [Jurk 08] the authors deal with real-time communications over cluster-tree sensor networks, where they evaluate the end-to-end communication delay. In [Cacc 03], the authors assume nodes in hexagonal cells and use inter-cell and intra-cell communication in single directions to ensure the real-time behavior. The protocol presented in [Watt 05] uses the distance from the last transmitting node to avoid data collisions and the data is sent in communication waves. However, none of these algorithms can ensure real-time and energy optimal routing, especially in high loaded networks.

We use the multi-commodity network flow model, because it does not need any particular network structure, like a hexagonal structure in [Cacc 03], or a tree topology in [Jurk 08]. This approach can handle any network topology with any number of sources and destinations. In contrast with all the papers about real-time routing in sensor networks referenced here, our approach ensures the real-time and energy optimal routing for all communication demands even in high loaded networks.

2.2 Multi-Commodity Network Flow Model

In this section, we briefly summarize the basic terminology and specify the multi-commodity network flow model used in this work. A simple example is introduced during this and the next section for better understanding.

Network Structure

The network is represented by a directed graph, where for each device able to send or receive data, a node of the graph exists. The nodes are labeled as $n = 1, \ldots, N$. Directed communication links are represented as ordered pairs (n_1, n_2) of distinct nodes. The presence of a link (n_1, n_2) means that the directed communication, from node n_1 to node n_2, is possible. The links are labeled as $l = 1, \ldots, L$. We define the set of the links l leaving the node n as $\mathcal{O}(n)$ and the set of the links l incoming to node n as $\mathcal{I}(n)$. The network structure is described with two incidence matrices in node-link form. The matrix of the incoming links is denoted A^+ and the matrix of the outgoing links is denoted A^-.

$$A^+_{n,l} = \begin{cases} 1, & l \in \mathcal{I}(n) \text{ (link } l \text{ enters node } n) \\ 0, & \text{otherwise} \end{cases} \tag{2.2.1}$$

$$A^-_{n,l} = \begin{cases} 1, & l \in \mathcal{O}(n) \text{ (link } l \text{ leaves node } n) \\ 0, & \text{otherwise} \end{cases} \tag{2.2.2}$$

Example: An example of a simple graph with 4 nodes and 5 links is shown in Figure 2.2.1. The numbers in parenthesis stand for the node and link labels. The values associated to the links stand for the communication costs. The matrices A^- and A^+ for this graph are:

$$A^- = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad A^+ = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

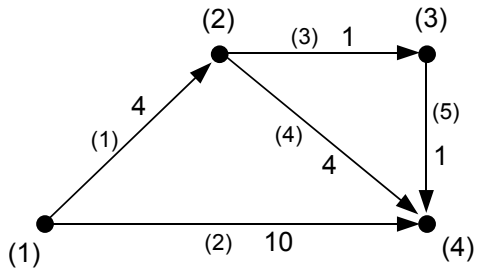


Fig. 2.2.1: Example: Graph of basic network.

Multi-Commodity Flow

In the multi-commodity network flow model, each node can send different pieces of data to any node. Each requested data transfer through the network is called the communication demand m and the set of all communication demands is labeled as $\mathcal{M}$. From the nature of the multi-commodity flow model, the data flow of each communication demand can be fragmented into more paths across the network.

The communication demands are represented as various flow commodities entering/leaving the network in some nodes. Each demand can come into the network in more than one node and leave the network in more nodes (multi-source, multi-sink). Let us denote the flow volume of demand m coming into the network in node n as $s_{in,n}^{(m)} \geq 0$ and similarly the flow leaving the network in node n as $s_{out,n}^{(m)} \geq 0$. We define the vectors of the flow of demand m leaving the network as $\vec{s}_{out}^{(m)} \in \mathcal{R}_+^N$ and the flow incoming into the network as $\vec{s}_{in}^{(m)} \in \mathcal{R}_+^N$ over all nodes. From the nature of the routing problem, it has to hold $\sum_{n=1}^{N} s_{out,n}^{(m)} = \sum_{n=1}^{N} s_{in,n}^{(m)}$ for all $m \in \mathcal{M}$.

Let $x_l^{(m)} \geq 0$ be the flow of demand m routed through the link l and $\vec{x}^{(m)} \in \mathcal{R}_+^L$ the flow vector for demand m. The flow vector and the leaving/incoming flow have to satisfy the flow conservation law for each communication demand:

$$A^- \vec{x}^{(m)} + \vec{s}_{out}^{(m)} = A^+ \vec{x}^{(m)} + \vec{s}_{in}^{(m)} \qquad \forall m \in \mathcal{M} \qquad (2.2.3)$$

Total volume of the flow in the link l over all communication demands is $t_l = \sum_{m \in \mathcal{M}} x_l^{(m)}$ and vector $\vec{t} \in \mathcal{R}_+^L$ denotes the total flow for each link over the network. It has to satisfy the capacity constraint $D\vec{t} \leq \vec{\mu}$. Where, $\vec{\mu}$ is the limit of the constraints and the matrix D represents the constraints structure. When there is a separate capacity for each link, matrix D is the identity matrix of size $[L \times L]$ and $\vec{\mu} \geq \vec{0}$ consists of the link capacities.

In summary, our network flow model imposes the following group of constraints on the network flow variables $\vec{x}^{(m)}$:

$$\begin{array}{rcll} A^- \vec{x}^{(m)} + \vec{s}_{out}^{(m)} & = & A^+ \vec{x}^{(m)} + \vec{s}_{in}^{(m)} & \forall m \in \mathcal{M} \\ \vec{t} & = & \sum\limits_{m \in \mathcal{M}} \vec{x}^{(m)} & \\ D\vec{t} & \leq & \vec{\mu} & \\ \vec{x}^{(m)} & \geq & \vec{0} & \forall m \in \mathcal{M} \end{array} \tag{2.2.4}$$

This model describes the average behavior of the data transmission, i.e., the average data rates on the communication links, and ignores packet-level details of transmission protocols. The link layer communication protocol (e.g. TDMA) should set the bandwidths for each demand according to the flow vectors $\vec{x}^{(m)}$. The link capacity should be defined appropriately, taking into account packet loss and retransmission, so the flow conservation law holds with sufficient probability.

Example (continued): Let all the link capacities in our example be equal to 1. Then the capacity constraints matrix D is the identity matrix of size $[5 \times 5]$ and $\vec{\mu} = (1, 1, 1, 1, 1)^T$.

Let there be two communication demands both with flow volume equal to 1. The first is routed from node 1 to node 4 and the second from node 2 to node 4. Therefore, we have: $\vec{s}_{in}^{(1)} = (1, 0, 0, 0)^T$, $\vec{s}_{in}^{(2)} = (0, 1, 0, 0)^T$, $\vec{s}_{out}^{(1)} = \vec{s}_{out}^{(2)} = (0, 0, 0, 1)^T$.

Routing Optimization

In our work, we focus on the total energy minimization over the whole network. The task is to minimize the objective function by setting the flow vectors $\vec{x}^{(m)}$ for all $m \in \mathcal{M}$

$$f_{total\ cost} = \vec{c}^T \vec{t} \tag{2.2.5}$$

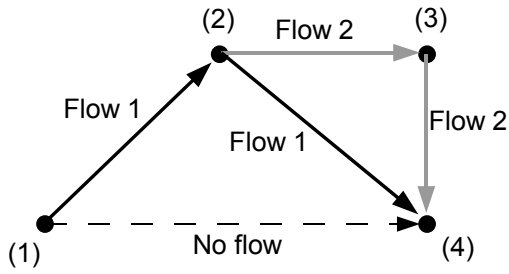


Fig. 2.2.2: Example: An optimal data flow routing with capacity constraints.

The vector $\vec{c}$ is a column vector of the energy consumption per data unit transmitted. The components of the vector $\vec{c}$ correspond with the energy consumption for the individual links in the network. Their values are usually determined directly from the transmission energy level in the nodes, which is needed for a reliable connection. This is done by the MAC layer of the communication protocol.

Another target application of this model is a sensor network, where the energy supplies of the nodes can be recharged, with different maintenance costs (e.g. depending on the node reachability). In such application the recharge price is included in the vector $\vec{c}$ and we minimize the network long-term maintenance cost.

Any Linear Programming solver can be used to find the optimal flow vectors $\vec{x}^{(m)}$ for all $m \in \mathcal{M}$.

Example (continued): Let the communication cost in our example be the same as in Figure 2.2.1 (the numbers associated to the edges). $\vec{c} = (4, 10, 1, 4, 1)$

One of the optimal solutions for this example is: $\vec{x}^{(1)} = (1, 0, 0, 1, 0)^T$, $\vec{x}^{(2)} = (0, 0, 1, 0, 1)^T$, which means that the first flow is routed through the nodes $(1 \rightarrow 2 \rightarrow 4)$ and the second flow is routed through the nodes $(2 \rightarrow 3 \rightarrow 4)$. The total flow is the sum of the routings over all demands $\vec{t} = (1, 0, 1, 1, 1)^T$ and the communication cost is equal to 10. This energy optimal routing is shown in Figure 2.2.2.

2.3 Energy Optimal Routing of Continuous Data Flow with Real-Time Constraints in Multi-hop Sensor Networks

This section is focused on the continuous flow problem which is well suitable for data streams or in the case of periodically sent data with a transmission period close to one hop communication delay.

We extend the multi-commodity flow model by real-time constraints, which guarantee sufficient routing delay through the network. Each communication demand has its own deadline and the communication delay of this demand has to be shorter than the deadline. We model the hop delay as an integer value associated to each communication link. For transparent model derivation, we assume the same communication delay over the entire network (i.e. each communication hop causes delay equal to one). However, the model can be easily extended to the general form (see [Trdl 07] for more details), where the communication delays can be different integer values for each link and demand.

First, we assume the value of the data flow to be a continuous quantity and we allow the data flow fragmentation to go to more routing paths. This approach is usually used for problems like video or voice data streaming to increase the robustness of the data routing at the price of possible quality decrease (see e.g. [Zhan 08]). Thanks to data flow continuity, the problem is solvable in polynomial time. Furthermore we focus on two types of problems with integral data flow that are NP hard. The first type is a problem, where the data flow can be split into more routing paths, however each fragment of the data flow has to have an integral volume. The second type is a problem, where the data flow cannot be split at all and all data flow of one communication demand has to follow the same routing path.

Example (continued): We added the deadline constraints into our example. Namely, the first communication demand has the deadline equal to 2 communication hops and the second communication demand has the deadline equal to 1 communication hop. With these new constraints, the solution shown in Figure 2.2.2 is not feasible.

2.3.1 Mathematical Model of Real-Time Routing for Continuous Data Flow

Let vector $\vec{x}^{(m,w)} \in \mathcal{R}_+^L$ denote the flow of communication demand m with integer communication delay w in the network. Then the flow vector $\vec{x}^{(m)}$ independent of the flow delay of demand m is equal to the sum of the flow vectors over all acceptable delays: $\vec{x}^{(m)} = \sum_{w=0}^{d^{(m)}} \vec{x}^{(m,w)}$. Where $d^{(m)}$ denotes the deadline of the communication demand m. Using this equation, we can rewrite the equation for total flow vector from the system of inequalities (2.2.4) into a new form:

$$\vec{t} = \sum_{m \in \mathcal{M}} \sum_{w=0}^{d^{(m)}} \vec{x}^{(m,w)} \tag{2.3.1}$$

Vector $\vec{s}_{out}^{(m,w)} \in \mathcal{R}_+^N$ stands for the flow of the demand m leaving the network with communication delay w and vector $\vec{s}_{in}^{(m,w)} \in \mathcal{R}_+^N$ denotes the flow of demand m coming into the network with initial delay w. As usual, the flow of each demand may come into the network and leave it in more nodes. If all flow fragments of one demand coming into the network in different nodes have the same initial delay, then $\vec{s}_{in}^{(m,0)} = \vec{s}_{in}^{(m)}$, and $\vec{s}_{in}^{(m,w)} = 0$ for $w > 0$. The flow of demand m leaving the network prior the deadline is:

$$\vec{s}_{out}^{(m)} = \sum_{w=0}^{d^{(m)}} \vec{s}_{out}^{(m,w)} \qquad \forall m \in \mathcal{M} \tag{2.3.2}$$

Through Equations (2.3.1, 2.3.2), we have converted the real-time constraint (i.e. the delay has to be shorter than the deadline) to the structural constraint. Only the flow, whose delay is shorter than the deadline, is represented. The flow, which does not meet the deadline, causes that the flow conservation law does not hold and then the network flow constraints are not satisfied, i.e. this solution is not feasible.

If the flow is sent through the network, the flow delay is increased by each communication hop. The flow of demand m coming into node n with communication delay w has to either leave the network in node n with the same delay w or reach the neighbor node with delay $w + 1$. The flow conservation

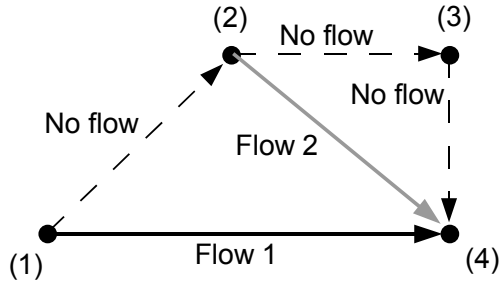


Fig. 2.3.1: Example: An optimal data flow routing with capacity and deadline constraints.

law from Equation (2.2.3) can be written in the delay awareness form as:

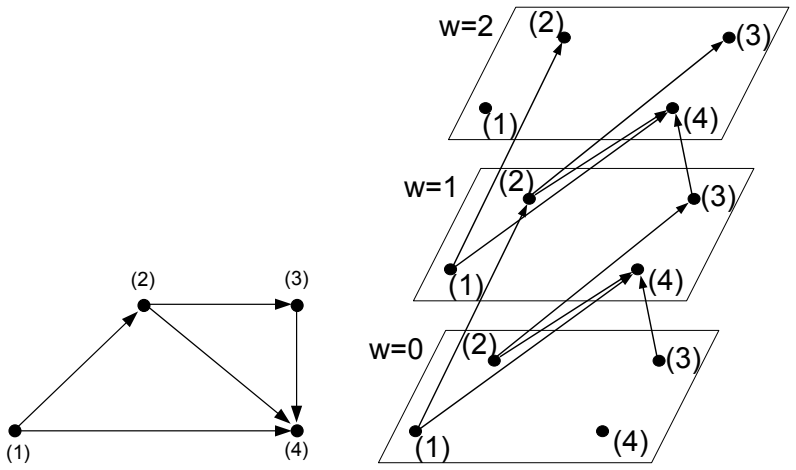
$$A^{-}\vec{x}^{(m,w+1)} + \vec{s}_{out}^{(m,w)} = A^{+}\vec{x}^{(m,w)} + \vec{s}_{in}^{(m,w)} \qquad \forall m \in \mathcal{M}, 0 \le w \le d^{(m)} \tag{2.3.3}$$

In summary, the constraints of the real-time multi-commodity flow routing problem can be written as:

$$\begin{aligned}
A^{-}\vec{x}^{(m,w+1)} + \vec{s}_{out}^{(m,w)} &= A^{+}\vec{x}^{(m,w)} + \vec{s}_{in}^{(m,w)} && \forall m \in \mathcal{M},\ 0 \le w \le d^{(m)} \\
\vec{s}_{out}^{(m)} &= \sum_{w=0}^{d^{(m)}} \vec{s}_{out}^{(m,w)} && \forall m \in \mathcal{M} \\
\vec{t} &= \sum_{m \in \mathcal{M}} \sum_{w=0}^{d^{(m)}} \vec{x}^{(m,w)} \\
D\vec{t} &\le \vec{\mu} \\
\vec{x}^{(m,w)} \ge \vec{0};\quad \vec{s}_{in}^{(m,w)} &\ge \vec{0};\quad \vec{s}_{out}^{(m,w)} \ge \vec{0} && \forall m \in \mathcal{M},\ 0 \le w \le d^{(m)} \\
\vec{x}^{(m,0)} = \vec{x}^{(m,d^{(m)}+1)} &= \vec{0} && \forall m \in \mathcal{M}
\end{aligned} \tag{2.3.4}$$

All feasible routings, which obey the deadlines and capacity constraints and realize all communication demands are described by the system of inequalities (2.3.4). To choose the cheapest one in terms of the objective function (e.g. 2.2.5) we can use Linear Programming with these constraints.

Example (continued): Solving the example with the deadline constraints according to (2.3.4) and objective function (2.2.5) we get the new



(a) Original graph (b) Expanded graph for 2 hops

Fig. 2.3.2: Intuitive presentation for the graph replication.

solution: $\vec{x}^{(1,1)} = (0,1,0,0,0)^T$, $\vec{x}^{(1,2)} = (0,0,0,0,0)^T$, $\vec{x}^{(2,1)} = (0,0,0,1,0)^T$, $\vec{x}^{(1,0)} = \vec{x}^{(2,0)} = (0,0,0,0,0)^T$. This means that the first flow is routed through nodes $(1 \to 4)$ and the second flow through nodes $(2 \to 4)$.

$\vec{s}_{out}^{(1,1)} = (0,0,0,1)^T$ and $\vec{s}_{out}^{(1,0)} = \vec{s}_{out}^{(1,2)} = (0,0,0,0)^T$, which means that the first flow leaves the network in node 4 with communication delay 1 and no part of the first flow leaves the network with communication delays 0 and 2. Similarly for the second flow $\vec{s}_{out}^{(2,1)} = (0,0,0,1)^T$ and $\vec{s}_{out}^{(2,0)} = (0,0,0,0)^T$. $\vec{s}_{out}^{(2,2)}$ has no sense for the second flow because its deadline is equal to 1.

The total load of the links is: $\vec{t} = (0,1,0,1,0)^T$ and the communication cost is equal to 14. The energy optimal real-time routing is shown in Figure 2.3.1.

2.3.1.1 Intuitive Presentation of Extended Graph

In this section, we illustrate in an intuitive way the graph transformation, which has been discussed in the previous Section 2.3.1 by mathematical equa-

tions. New variables have appeared for each communication link as well as new constraining equations for each node. These variables and constraints can be seen as virtual layers of the network where each layer represents a different communication delay w. The number of the network layers is equal to the integer deadline of the demand m plus one (the number of allowed communication hops plus a zero layer). As consistent with the structure of Equation (2.3.3), all communication links are redirected to the nodes in the higher layer, which means that the flow is routed not only in node space but also in delay space. Because the number of layers is limited by the deadline and the flow can leave the network only in virtual nodes of the sink nodes, all possible routings through this transformed network hold the deadlines. An example of the expanded graph is shown in Figure 2.3.2.

2.3.1.2 Simultaneous Real-Time and non-Real-Time Routing

The network communication problems are often comprised with real-time and non-real-time communication demands. The non-real-time routing demands can be taken as if they would have no communication delays. Therefore, they can be solved as the common multi-commodity routing problem presented in Section 2.2. The only resources where the non-real-time and the real-time flow interact are guarded by the capacity constraints. Let $\mathcal{M}'$ be a set of non-real-time demands and vector $\vec{x}^{(m')}$ be a non-real-time flow of demand $m' \in \mathcal{M}'$.

A new equation of flow conservation for the non-real-time flow has to be added to the system of inequalities (2.3.4):

$$(A^+ - A^-)\vec{x}^{(m')} = \vec{s}_{out}^{(m')} - \vec{s}_{in}^{(m')} \qquad m' \in \mathcal{M}' \tag{2.3.5}$$

and the equation for the total flow vector (2.3.1) has to be changed to consider the non-real-time flow:

$$\vec{t} = \sum_{m \in \mathcal{M}} \sum_{w=0}^{d^{(m)}} \vec{x}^{(m,w)} + \sum_{m' \in \mathcal{M}'} \vec{x}^{(m')} \tag{2.3.6}$$

With these changes in the system of inequalities (2.3.4), simultaneous real-time and non-real-time routing is possible.

2.3.1.3 Time Complexity

A big advantage of this approach is the polynomial-time complexity. The exact time complexity of the computation depends on a specific solver used for the linear optimization and on the number of variables and constraints. We denote N as the number of nodes, L as the number of communication links, W_{max} as the maximum of deadlines $d^{(m)}$ and M as the number of communication demands. The number of variables no_{var} is the sum of the flow variables x, s_{out} and t:

$$no_{var} = L(\sum_{m=1}^{M}(d^{(m)})) + N\sum_{m=1}^{M}(d^{(m)}+1) + L \tag{2.3.7}$$

If we consider that $L \leq N(N-1)$ and that the number of allowed communication hops is smaller than the number of nodes ($W_{max} \leq N-1$), we can limit the worst-case number of the variables as:

$$O_{var}(MN^3) \tag{2.3.8}$$

Please notice that the worst-case number of the variables for the non-real-time multi-commodity flow problem in the node-link form is:

$$O_{var}(MN^2) \tag{2.3.9}$$

2.3.1.4 Integral Flow

The major practical inconvenience of the model presented up to now is flow fragmentation, which is not suitable for all applications (i.e. the flow from the source node can be split and each fragment of the flow can reach the sink node by a different path, even though a non-fragmented solution exists which satisfies the capacity constraints). The multi-commodity routing problem with integral flow is typically solved by an Integer Linear Programming solver through algorithms based on the Branch-and-Bound mechanism, Cutting planes, Lagrangian relaxation, Evolution algorithms, etc., where the continuous problem is used to derive a partial solution (see [Bert 98, Piro 04]).

Two types of the integral flow problems in applications, where the flow cannot be fragmented, can be defined. The first type is a problem (further called *Splittable integral flow*), where the flow can be split into more routing

paths, however each fragment of the flow has to have an integral volume (e.g. the flow volume denotes the number of data packets, or number of bytes). The second type is a problem (further called *Unsplittable flow*), where the flow cannot be split at all and all flow of one communication demand has to follow the same routing path (e.g. because of the increase of the protocol overhead). Both types of the integral multi-commodity network flow problems are NP hard even for two communications demands (see [Gare 79]). In this section, we briefly describe the modifications of the system of inequalities (2.3.4) to solve integral flow problems.

Splittable integral flow: To solve the real-time routing for splittable integral flow, the system of inequalities (2.3.4) has to be extended by a new constraint: the variables $\vec{x}^{(m,w)}$ have to be an integer for all communication delays w and for the communication demands m, which are supposed to be integral. The simultaneous routing of integral and fragmented flow is possible. The Linear Programming algorithm has to be substituted by Integer Linear Programming. This problem can be transformed to an unsplittable flow problem by dividing the splittable communication demands into more unsplittable communication demands, however at the cost of a huge increase in the number of the communication demands.

Unsplittable flow: To solve the real-time routing problem with an unsplittable flow, we have to adapt the system of inequalities (2.3.4). The variables $\vec{x}^{(m,w)}$ have to be an integer for all communication delays w and for all communication demands m with unsplittable flow. Moreover, the volume of the unsplittable flow is set to one ($s_{in}^{(m)} = 1$) and the equation for the total link flow (2.3.1) is changed as follows:

$$\vec{t} = \sum_{m \in \mathcal{M}} v^{(m)} \sum_{w=0}^{d^{(m)}} \vec{x}^{(m,w)} \tag{2.3.10}$$

where the $v^{(m)}$ denotes the real volume of the communication demand m.

2.3.2 Numerical Experiments

To demonstrate the benefits and correctness of our approach for real-time routing, we have simulated the routing problems in Matlab. We have demon-

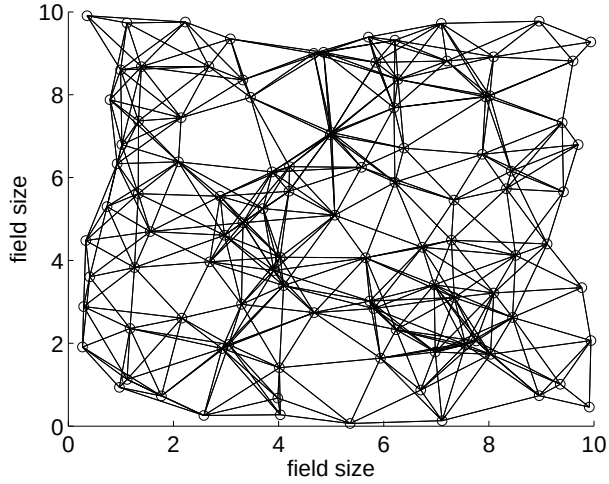


Fig. 2.3.3: Topology of a randomly generated wireless network with 100 nodes and 1066 directed communication links.

strated the data collection problem in the network with 100 nodes for two communication demands and a set of experiments to demonstrate the time complexity of the computation algorithm. The simulation has been run using a commercial solver CPLEX 9.1 on a computer with an AMD processor Opteron 248 at 2200 MHz, 2GB RAM DDR at 400MHz.

2.3.2.1 Data Collection Problem

For the data collection problem (i.e. multi-source, mono-sink problem) we consider a network field of size $[10 \times 10]$ and divide it into 100 subsquares of size $[1 \times 1]$. One node is randomly placed into each subsquare and the communication distance is set to 2 (i.e. node A can communicate with node B within one hop, if and only if their Euclidean distance is less than 2). Each node, which is not on the border of the network field, has at least three communication links to its neighbors. Please notice, that our network is close to the "unit-disk network". An example of such a random network topology is shown in Figure 2.3.3. There are two communication demands in the network. Each node has a 50% probability that it will send data of the first communication demand to the first sink node and a 50% probability that

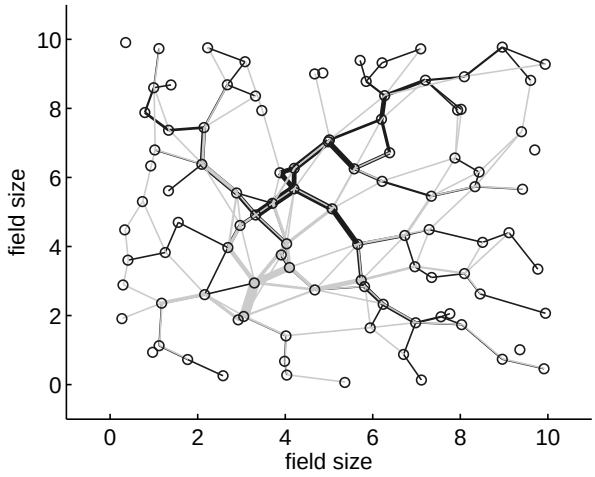


Fig. 2.3.4: Optimal real-time data flow routing in a network with 100 nodes and 2 communication demands for the data collection problem.

it will send data of the second communication demand to the second sink node (i.e. the data flow coming into the network in node n of the demand 1 is $s^{(1)}_{in,n} \in \{0,1\}$ and the data flow of the demand 2 is $s^{(2)}_{in,n} \in \{0,1\}$). The deadline of the first communication demand is set to 8 communication hops and the deadline of the second communication demand is set to 6 communication hops. The link capacities are set to 30. The communication costs per transmitted data flow unit are set as the power of the distance between the nodes.

The resulting data flow routing through the network is shown in Figure 2.3.4, where only the used links are shown and the links width is a logarithmic function of the volume of the data flow routed through the link. The first type of data flow is in black and the second data flow is in grey. Each routing path of the first type of data flow has at maximum 8 communication hops and each routing path of the second type of data flow has at maximum 6 communication hops.

The routing in the node-delay space is shown in Figure 2.3.5. On the horizontal axis, the single nodes are placed and on the vertical axis, the integer delays are placed. The data flow, which is produced by the nodes,

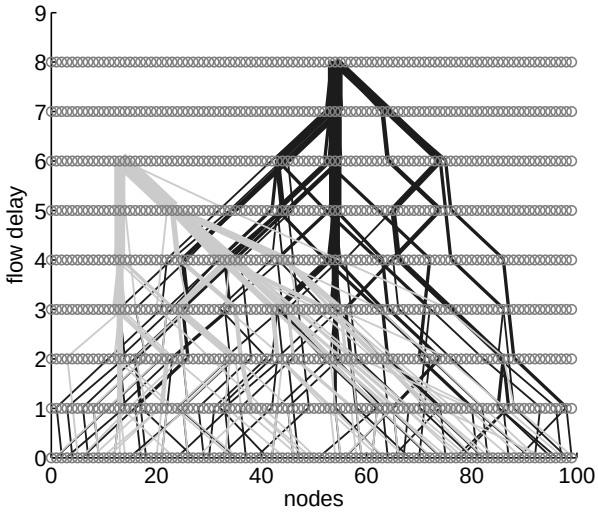


Fig. 2.3.5: Optimal real-time data flow routing in node-delay space for two communication demands with deadlines of 8 and 6 communication hops for the data collection problem. The sink nodes are nodes number 54 and 13.

starts with the delay equal to zero and through each communication hop the delay is increased. The lines width is a logarithmic function of the volume of the data flow. The vertical lines in node 54 and in node 13 represent the data flows reaching these two sink nodes.

2.3.2.2 Time Complexity

To demonstrate the time complexity of our approach, we have varied the experiment of data collection presented in Section 2.3.2.1. The size of the square field is gradually set from $[2 \times 2]$ to $[10 \times 10]$ with as the number of nodes from 4 to 100 as well. The communication distance is kept at 2 and the communication costs are kept as a power of the node's distance. The number of communication demands is set to 5 (and to 20 for the second experiment). Each node sends data flow to 5 different sink nodes (or to 20 in the second experiment). For each demand m and for each node n the volume of data flow coming into the network is equal to one ($s_{in,n}^{(m)} = 1$). The data flow volume of demand m, which is leaving the network in sink node i, is equal to

N ($s^{(m)}_{out,i} = \sum_{n=1}^{N} s^{(m)}_{in,n} = N$). The link capacities are set as $\mu = 0.25NM/3$ where N denotes the number of the nodes in the network and M denotes the number of communication demands. The deadline for each communication demand is set as $d^{(m)} = \sqrt{N}$.

The simulation has been run 500 times for each number of nodes on randomly constructed networks for both the fragmented data flow problem (2.3.4) and for the integral flow problem ((2.3.4) augmented by Section (2.3.1.4)). The minimum, average and maximum computational times for 5 communication demands and for 20 communication demands are shown in Table 2.3.1 and in Table 2.3.2, respectively.

Opposite to our expectations, the computation times for integral flow are not much higher than the computation times for the fragmented flow, even though the integral flow problem is NP hard and the fragmented flow problem is polynomial. The performance of Integer Linear Programming is surprisingly good, since the Integer Linear Programming usually solves the problems with 1000 integer decision variables at most, while in our case we solve the problems with up to 220000 integer variables. Let us remind everyone that the matrix representing constraints in our Integer Linear Programing formulation is not totally unimodular, thus it does not imply an integer solution in polynomial time [Bert 98]. The difference between integral and fragmented flow was just slightly bigger, while using a non-commercial GLPK solver. Therefore, we have performed several attempts to make the problem more difficult.

2.3.2.3 Time Complexity when Progressively Decreasing the Link Capacities

To demonstrate time complexity for the instances on the limit of feasibility, we have solved the data collection problem and we have gradually decreased the link capacities.

The network has been constructed in the same way as in the previous experiment. The size of the network has been set to 64 nodes ($[8 \times 8]$ field), the starting link capacities have been set to 31, the communication distance to 2 and the communication costs as the power of the node's distance. The number of communication demands has been set to 10, and the data flow volume of each node for each communication demand is equal to 1 (i.e. for all communication demands m and all nodes n the data flow coming into

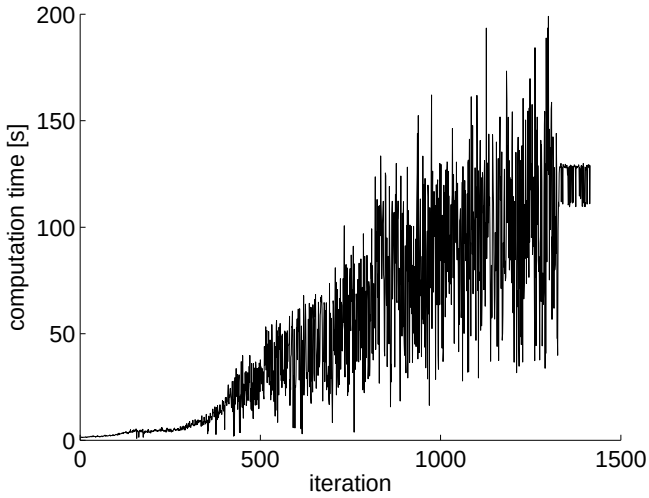


Fig. 2.3.6: Computation times when progressively decreasing the link capacities for 10 communication demands, fragmented flow, network with 64 nodes ([8 × 8] field)

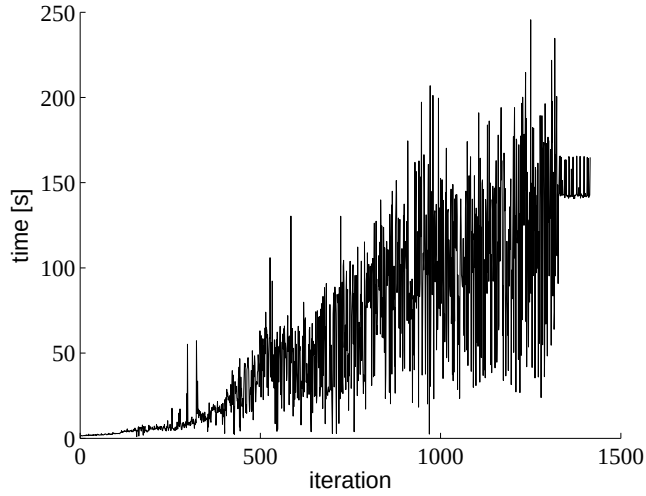


Fig. 2.3.7: Computation time when progressively decreasing the link capacities for 10 communication demands, integral flow, network with 64 nodes ([8 × 8] field)

	Fragmented flow [s]			Integral flow [s]		
Nodes	min	avg	max	min	avg	max
4	0.01	0.01	0.02	0.01	0.01	0.02
9	0.01	0.02	0.03	0.02	0.02	0.03
16	0.02	0.02	0.03	0.03	0.04	0.05
25	0.04	0.05	0.07	0.06	0.07	0.13
36	0.08	0.11	0.14	0.11	0.14	0.19
49	0.15	0.18	0.29	0.20	0.25	0.45
64	0.26	0.29	0.42	0.35	0.40	0.56
81	0.45	0.56	0.90	0.59	0.70	0.97
100	0.85	1.23	2.02	1.04	1.42	2.35

Table 2.3.1: Computation times for 5 communication demands [s]

the network is $s_{in,n}^{(m)} = 1$). The deadline for each communication demand has been set to 8 communication hops.

The test decreasing the link capacities works as follows: We solve the data collection problem, then we pick the link with the maximum data flow and we decrease the link capacity to half of the volume of the data flow routed through the link ($\mu_l = t_l/2$; where t_l is the routed data flow in the link and μ_l is the new link capacity). Then we solve the new problem. If the problem becomes infeasible, we set the link capacity to the previous value and mark it, so that it cannot be decreased. In the next iteration we choose the non marked link with the maximum data flow. We repeat the test until there is no link, able to decrease its capacity.

The computation times for fragmented flow and for integral flow are shown in Figure 2.3.6 and Figure 2.3.7, respectively. The computation time increases while decreasing the capacity of the links. However, even the maximum computational times are acceptable and there is little difference between the integral and fragmented flow. The computation times at the end of the graph correspond to the phase, when the test was decreasing the capacity of the links with zero data flow, which does not change the optimal routing.

In order to better compare the computation times of fragmented and integral flow we have filtered them as an average value with a floating window. The comparison with the floating window of 100 samples is shown in Figure

	Fragmented flow [s]			Integral flow [s]		
Nodes	min	avg	max	min	avg	max
4	0.01	0.02	0.03	0.01	0.01	0.02
9	0.03	0.03	0.04	0.04	0.05	0.21
16	0.06	0.09	0.12	0.10	0.13	0.18
25	0.16	0.22	0.34	0.26	0.32	0.48
36	0.33	0.43	0.64	0.51	0.62	0.82
49	0.67	0.87	1.51	0.95	1.20	1.76
64	1.27	1.63	2.77	1.82	2.20	3.21
81	2.41	2.96	4.91	3.21	3.83	5.73
100	4.09	5.14	9.85	5.29	6.39	9.28

Table 2.3.2: Computation times for 20 communication demands [s]

2.3.8 (i.e. the first sample of the graph is the average value of the computation times from iteration 1 to iteration 100, the second sample is the average value of the computation times from iteration 2 to iteration 101, ...).

2.3.2.4 Time Complexity in Relation to the Number of Communication Demands

This experiment demonstrates time complexity depending on the number of communication demands and the algorithm behavior in the case of networks with a large number of communication demands.

The experiment is performed in a network with 100 nodes (square field $[10 \times 10]$), communication distance equal to 2, communication costs equal to the power of the nodes distance, deadlines equal to 10 communication hops and the link capacities equal to 2. The communication demands are set between two nodes (single-source, single-sink) and the volume of the data flow for each demand is equal to 1. In each iteration of this test we add a random communication demand (random source and sink). When the problem becomes infeasible, we change the last communication demand. We have run the experiment on 508 random networks and the average, maximum and minimum computation times are shown in Figure 2.3.9 for fragmented flow and in Figure 2.3.10 for integral flow.

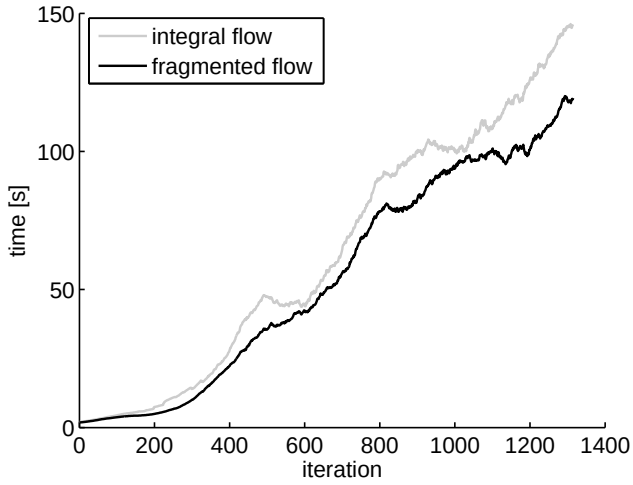


Fig. 2.3.8: Comparison of the average computation times for fragmented and integral flow. Averaging with floating window size of 100.

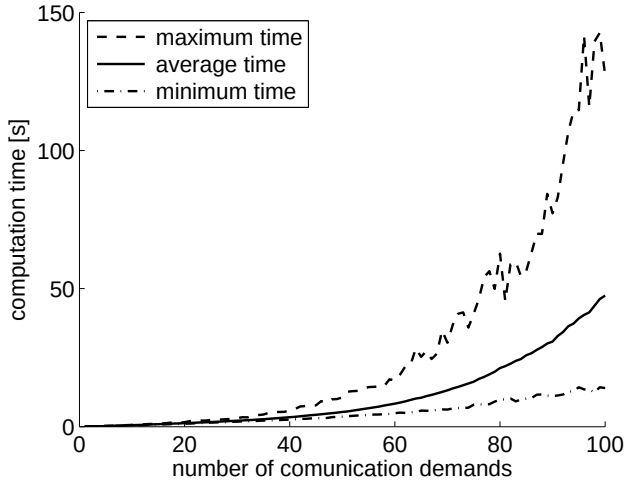


Fig. 2.3.9: Computation time for fragmented flow in a network with 100 nodes depending on the number of communication demands.

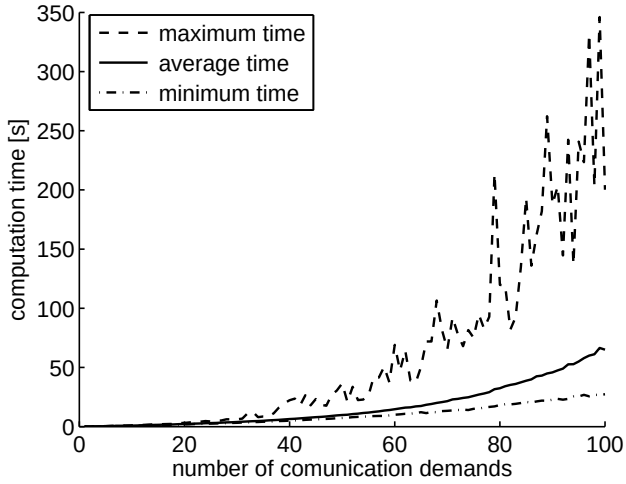


Fig. 2.3.10: Computation time for integral flow in a network with 100 nodes depending on the number of communication demands.

2.4 Energy Optimal Routing of Periodic Messages with Real-Time Constraints in Multi-hop Sensor Networks

In previous Section 2.3 we have derived a multi-commodity network flow model to compute the real-time routing for sensor networks which models the communication demands as a continuous data flow. It is well suitable for data streams, or communication demands with transmission period close to one hop communication delay. However in the case when the periods of data transmissions are much bigger than the one hop communication delay, the flow model in Section 2.3 leads to waste of the link capacities. This section deals with the case when the transmission periods are much bigger than the one hop communication delay.

The problem definition is based on the same underlying model as in previous Section 2.3, which is introduced in Section 2.2. Moreover, we define for each communication demand the transmission period and the transmission offset (start of the periodic transmission). The data volume for each communication demand is redefined as data volume which is transmitted during every transmission period. Let us remind that a data transfer from several source nodes to one sink node with the same deadlines can be described by one communication demand (multi-source). In a similar way, the model allows to describe a problem with several sink nodes (multi-sink).

In this approach the transmission offset is strictly given in the setting of the problem and it is not an object of the optimization (no transmission offset phasing). This situation occurs e.g. in applications with synchronized measurement or in applications designed for measurement of some discrete process (e.g. flashes of light, short sound intensity etc.). There are applications where the transmission offset (time of the data measurement) can be shifted for each device independently. For these applications it is possible to achieve better solutions by inclusion of the transmission offset into the optimization. However solving any of these problems is NP hard and we will not investigate this problem in this work (see e.g. [Such 07]). The approach presented in this chapter can be used as a heuristic to evaluate solutions gained e.g. by Genetic Algorithms for the problem with the phasing.

For more transparent model derivation, we assume the same communication delay over the entire network (i.e. each communication hop causes delay

equal to one).

2.4.1 Mathematical Model of Real-Time Routing for Periodic Messages

Let vector $\vec{x}^{(m,w)} \in \mathcal{R}_+^L$ denotes the flow of communication demand m with integer communication delay w in the network.

If the flow is sent through the network, the flow delay is increased by each communication hop. The flow of demand $m \in \mathcal{M}$ coming into node n with communication delay w has to either stay in the node n and increase its delay to $w+1$ or reach the neighbor node with delay $w+1$. The flow comes into the network with zero delay ($w = 0$) and leaves the network with delay $d^{(m)}$, the deadline of the demand m.

The flow conservation law from Equation (2.2.3) can be rewritten in a delay awareness form as:

$$\begin{aligned} A^- \vec{x}^{(m,w)} + \vec{y}^{(m,w)} &= A^+ \vec{x}^{(m,w-1)} + \vec{y}^{(m,w-1)} \\ &\quad \forall m \in \mathcal{M}, \quad 2 \leq w \leq d^{(m)} \end{aligned} \tag{2.4.1}$$

Where the $\vec{y}^{(m,w)} \in \mathcal{R}_+^L$ denotes the flow of communication demand $m \in \mathcal{M}$ with communication delay w which is not transmitted to other node and waits to the next transmission period with the communication delay $w+1$. Slightly different form of Equation (2.4.1) holds for $w = 1$ and for $w = d^{(m)} + 1$, which describe the first communication hop of the flow and the flow leaving the network:

$$A^- \vec{x}^{(m,1)} + \vec{y}^{(m,1)} = \vec{s}_{in}^{(m)} \quad \forall m \in \mathcal{M} \tag{2.4.2}$$

$$\vec{s}_{out}^{(m)} = A^+ \vec{x}^{(m,d^{(m)})} + \vec{y}^{(m,d^{(m)})} \quad \forall m \in \mathcal{M} \tag{2.4.3}$$

Where $\vec{x}^{(m,w)} \geq 0$ and $\vec{y}^{(m,w)} \geq 0$ for all $m \in \mathcal{M}$ and all $1 \leq w \leq d^{(m)}$.

Through Equations (2.4.1, 2.4.2, 2.4.3), we have converted the real-time constraint (i.e. the delay has to be shorter than the deadline) to the structural constraint. Only the flow, whose delay is shorter than the deadline, is represented. The flow, which does not meet the deadline, causes that the flow conservation law does not hold and then the network flow constraints are not satisfied, i.e. this solution is not feasible.

Let $p^{(m)} > 0$ denotes the transmission period and $o^{(m)} \geq 0$ denotes the transmission offset (start time within the network period P) of communication demand $m \in \mathcal{M}$ and P denotes a network period defined as a least common multiple of periods $p^{(m)}$ of all communication demands $m \in \mathcal{M}$.

We use variable $q \in \mathcal{N}$ to denote a discrete time within the network period $1 \leq q \leq P$. To satisfy the communication capacities we have to distinguish the flow in the communication links according to the time q within the network period P. To do so, we define a set $\mathcal{T}(q)$ of pairs (m, w). If pair (m, w) is in set $\mathcal{T}(q)$ it means that the flow of demand $m \in \mathcal{M}$ with delay w is routed through the network in time q.

$$\begin{aligned} \mathcal{T}(q) \;=\; \Big\{ & (m,w) \mid m \in \mathcal{M};\; 1 \leq w \leq d^{(m)}; \\ & q = \; (w + u \cdot p^{(m)} + o^{(m)}) \mod P; \\ & 0 \leq u < P/p^{(m)};\; u \in \mathcal{Z}_+^0 \Big\} \end{aligned} \tag{2.4.4}$$

Where the operand mod denotes a remainder of integer division. The expression $(u \cdot p^{(m)} + o^{(m)})$ is equal to the time when the flow of demand m is coming into the network, where u is a segment number of the transmission of the demand m within the network period P.

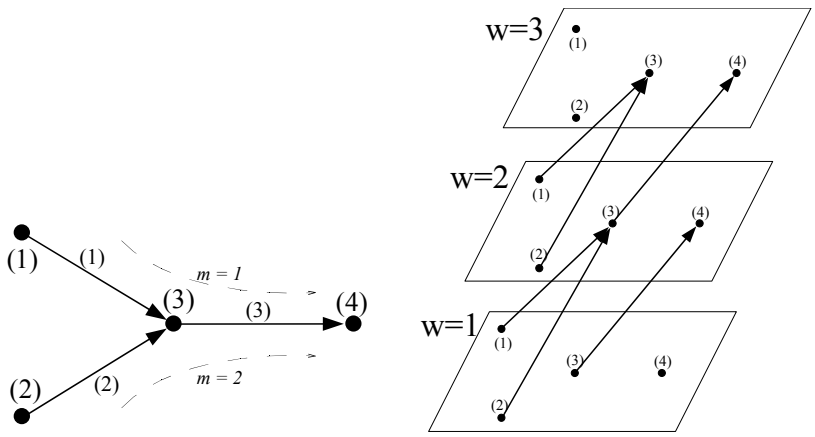
Now, we can define an equation for the flow in the communication links in time q as:

$$\vec{t}^{(q)} \;=\; \sum_{(m,w)\in\mathcal{T}(q)} \vec{x}^{(m,w)} \qquad \forall 1 \leq q \leq P \tag{2.4.5}$$

In every time q within the network period P the link capacities has to satisfy $(D\vec{t}^{(q)} \leq \vec{\mu})$.

In summary, the constraints of the real-time multi-commodity flow routing problem can be written as:

$$\begin{aligned} A^- \vec{x}^{(m,w)} + \vec{y}^{(m,w)} &= A^+ \vec{x}^{(m,w-1)} + \vec{y}^{(m,w-1)} \\ &\qquad \forall m \in \mathcal{M},\; 2 \leq w \leq d^{(m)} \\ A^- \vec{x}^{(m,1)} + \vec{y}^{(m,1)} &= \vec{s}_{in}^{(m)} \qquad \forall m \in \mathcal{M} \\ A^+ \vec{x}^{(m,d^{(m)})} + \vec{y}^{(m,d^{(m)})} &= \vec{s}_{out}^{(m)} \qquad \forall m \in \mathcal{M} \\ \sum_{(m,w)\in\mathcal{T}(q)} \vec{x}^{(m,w)} &= \vec{t}^{(q)} \qquad \forall 1 \leq q \leq P \\ D\vec{t}^{(q)} &\leq \vec{\mu} \qquad \forall 1 \leq q \leq P \\ \vec{x}^{(m,w)} \geq \vec{0};\quad \vec{y}^{(m,w)} &\geq \vec{0} \qquad \forall m \in \mathcal{M},\; 0 \leq w \leq d^{(m)} \end{aligned} \tag{2.4.6}$$



(a) Original graph (b) Expanded graph for 2 hops

Fig. 2.4.1: Intuitive presentation for the graph replication.

All feasible routings, which obey the deadlines and capacity constraints and realize all communication demands are described by the system of inequalities (2.4.6). To choose the cheapest one in terms of the objective function $f_{total\ cost} = \vec{c}^T \vec{t}$ we can use Linear Programming with these constraints.

2.4.1.1 Example and Intuitive Presentation

For a more transparent presentation of the presented approach, we show a simple example. Assume a simple graph with 4 nodes and 3 oriented links according to Figure 2.4.1(a). The numbers in the parenthesis stand for the node and link indexes. The link capacities are equal to $\vec{\mu} = (1, 1, 1)$, D is the identity matrix of size $[3 \times 3]$ and the communication costs are $\vec{c} = (1, 1, 1)$. We assume 2 communication demands in the network:

- The first communication demand $m = 1$ from node 1 to node 4, with deadline $d^{(1)} = 3$, period $p^{(1)} = 2$, offset $o^{(1)} = 0$; $\vec{s}_{in}^{(1)} = (1, 0, 0, 0)$ and $\vec{s}_{out}^{(1)} = (0, 0, 0, 1)$
- The second communication demand $m = 2$ from node 2 to node 4, with $d^{(2)} = 2$, $p^{(2)} = 4$, $o^{(2)} = 0$; $\vec{s}_{in}^{(2)} = (0, 1, 0, 0)$ and $\vec{s}_{out}^{(2)} = (0, 0, 0, 1)$

The intuitive presentation of the network replication of the graph from Figure 2.4.1(a) is presented in Figure 2.4.1(b) for the deadline equal to two communication hops. This replication is an effect of Equations (2.4.1, 2.4.2, 2.4.3) which convert the real-time constraint to the structural constraint. Each layer of the expanded graph represents a different communication delay and the links are redirected to the higher layer (i.e. each communication hop cause a unit delay).

The network period is $P = 4$ and the sets $\mathcal{T}(q)$ for $1 \leq q \leq P$ are:

$$\begin{aligned} \mathcal{T}(1) &= \big\{(1,1),(1,3),(2,1)\big\} \\ \mathcal{T}(2) &= \big\{(1,2),(2,2)\big\} \\ \mathcal{T}(3) &= \big\{(1,1),(1,3)\big\} \\ \mathcal{T}(4) &= \big\{(1,2)\big\} \end{aligned}$$

The sets are created according to Equation (2.4.4) and means that e.g. in time $q = 2$ the variables $\vec{x}^{(1,2)}$ and $\vec{x}^{(2,2)}$ denote the flow in the links.

Than the optimal routing for first communication demand is $\vec{x}^{(1,1)} = (1,0,0)$, $\vec{x}^{(1,2)} = (0,0,0)$, $\vec{x}^{(1,3)} = (0,0,1)$, $\vec{y}^{(1,1)} = \vec{y}^{(1,3)} = (0,0,0,0)$, $\vec{y}^{(1,2)} = (0,0,1,0)$ and for the second communication demand is $\vec{x}^{(2,1)} = (0,1,0)$, $\vec{x}^{(2,2)} = (0,0,1)$, $\vec{y}^{(2,1)} = \vec{y}^{(2,2)} = (0,0,0,0)$ (i.e. the flow of the first communication demand is routed through the links 1, 2 and waits in node 3, the flow of the second communication demand is routed through the links 2, 3 and waits in no node). Gantt Chart of the flow routed through the links in time within the network period ($1 \leq q \leq P$) is presented in Figure 2.4.2. Where, the first communication demand is in the light gray and the second in the dark gray.

The solution is constrained by:

- real-time constraint (maximum communication delay). It used Equations (2.4.1, 2.4.2, 2.4.3) and can be seen as a network replication (Figure 2.4.1(b)). Only flow which satisfies the real-time constrain is represented.
- the link capacities using the sets $\mathcal{T}(q)$ in every time $1 \leq q \leq P$ (Equations 2.4.4 and 2.4.5).

Please notice, that the presented transformation to solve the real-time routing problem preserves the problem in the minimum-cost multi-commodity network flow formulation. Due to this fact, the well known methods from this area can be used even on the real-time problems.

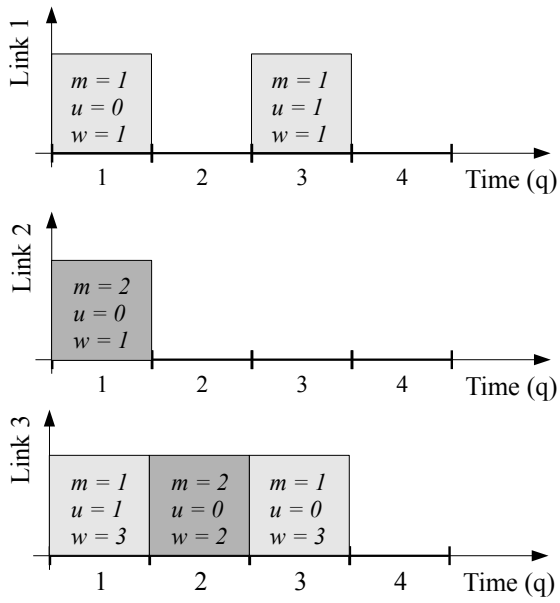


Fig. 2.4.2: Gantt Charts for the network links

2.4.1.2 Time Complexity

A big advantage of this approach is the polynomial-time complexity. The exact time complexity of the computation depends on a specific solver used for the linear optimization and on the number of variables and constraints. We denote N as the number of nodes, L as the number of communication links, W_{max} as the maximum of deadlines $d^{(m)}$ and M as the number of communication demands. The number of variables no_{var} depends on number of flow variables $x_l^{(m,w)}$, $y_n^{(m,w)}$ and $t_l^{(q)}$:

$$no_{var} = L \sum_{m=1}^{M} (d^{(m)}) + N \sum_{m=1}^{M} (d^{(m)}) + PL \tag{2.4.7}$$

If we consider that $L \leq N(N-1)$ and that the number of allowed communication hops is smaller than the number of nodes ($W_{max} \leq N-1$), we can limit the worst-case number of the variables as:

$$O_{var}(MN^3 + PN^2) \tag{2.4.8}$$

The number of constraints no_{cons} is:

$$no_{cons} = N \sum_{m=1}^{M} (d^{(m)}) + 2PL \tag{2.4.9}$$

Under the same assumptions the worst-case number of the constraints is:

$$O_{cons}\big(N^2(M+P)\big) \tag{2.4.10}$$

Please notice that the worst-case number of the variables for the non-real-time multi-commodity flow problem in the node-link form is: $O_{var}(MN^2)$ and the worst-case number of the constraints for the non-real-time multi-commodity flow problem in the node-link form is: $O_{var}\big(N(M+N)\big)$

2.4.2 Numerical Experiments

To demonstrate our real-time routing algorithm for messages with transmission period bigger than one hop communication delay, we have performed several simulations in Matlab. We have demonstrated the data collection problem in the network with 64 nodes for two communication demands and a set of experiments to evaluate the time complexity of the computation algorithm. The simulation has been run using YALMIP [Lfbe 04] and commercial solver CPLEX 9.1 on a computer with an AMD processor Opteron 248 at 2200 MHz, 2GB RAM DDR at 400MHz.

2.4.2.1 Data Collection Problem

For the data collection problem (i.e. multi-source, mono-sink problem) we consider a network field of size [8×8], divided into 64 sub-squares of size [1×1]. One node is randomly placed into each sub-square and the communication distance is set to 2 (i.e. node A can communicate with node B within one hop, if and only if their Euclidean distance is less than 2). There are two communication demands in the network. Each node has a 50% probability that it will send data of the first communication demand to the first sink node and a 50% probability that it will send data of the second communication demand to the second sink node (i.e. $s_{in,n}^{(1)} \in \{0,1\}$ and $s_{in,n}^{(2)} \in \{0,1\}$). The deadline of the first communication demand is set to $d^{(1)} = 4$ communication hops and the deadline of the second communication demand is set to $d^{(2)} = 5$

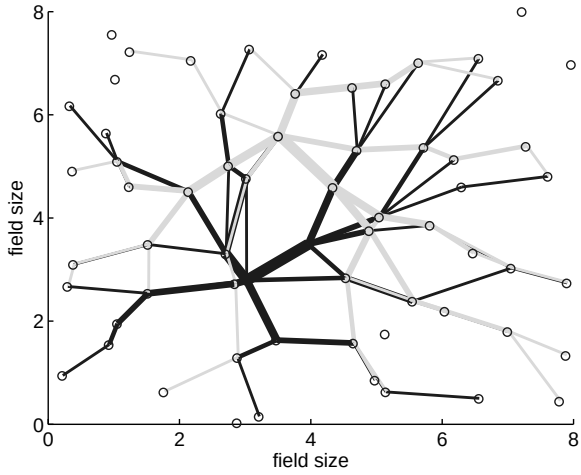


Fig. 2.4.3: Optimal real-time data routing in a network with 64 nodes and 2 communication demands for the data collection problem.

communication hops. The transmission periods are $p^{(1)} = 4$ for the first communication demand and $p^{(2)} = 6$ for the second communication demand. The transmission offsets are set to zero $o^{(1)} = o^{(2)} = 0$. The network period P is 12. The link capacities are set to $\mu = \frac{1}{2} \max_{m \in \mathcal{M}} \sum_{n=1}^{N} s_{in,n}^{(m)}$. The communication costs per transmitted data flow unit are set as the power of the distance between the nodes.

The resulting data routing through the network is shown in Figure 2.4.3, where only the used links are shown and the links width is a logarithmic function of the sum of the data flow volume routed through the link over the network period. The first type of data flow is in black and the second data flow is in gray. Each routing path of the first type of data flow has at maximum 4 communication hops and each routing path of the second type of data flow has at maximum 5 communication hops.

The routing in the node-delay space is shown in Figure 2.4.4. On the horizontal axis, the single nodes are placed and on the vertical axis, the integer delays are placed. The data flow, which is produced by the nodes, starts with the delay equal to zero and through each communication hop the delay is increased. The lines width is a logarithmic function of the volume

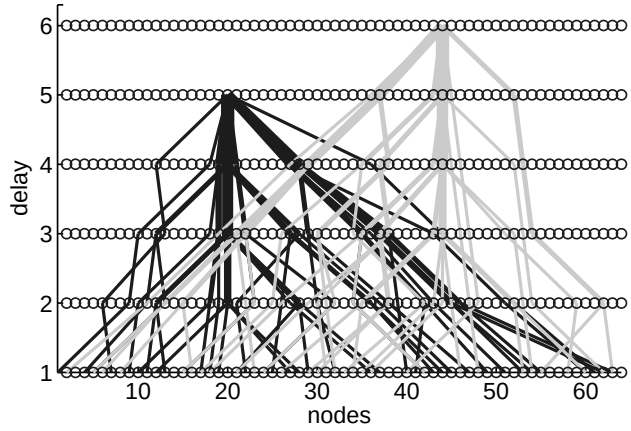


Fig. 2.4.4: Optimal real-time data routing in node-delay space for two communication demands with deadlines of 4 and 5 communication hops for the data collection problem. The sink nodes are nodes number 20 and 44.

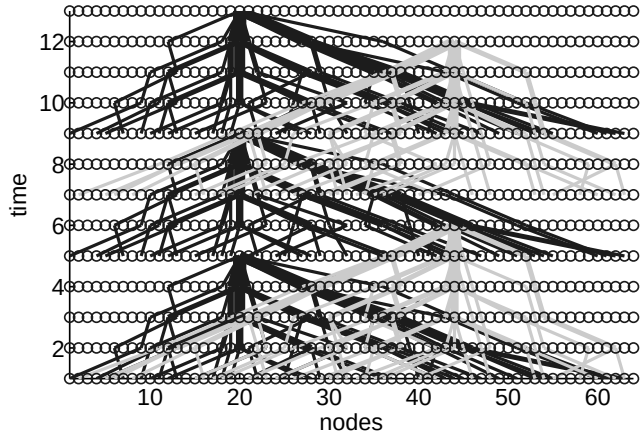


Fig. 2.4.5: Optimal real-time data routing in node-time space for two communication demands with deadlines of 4 and 5 communication hops and transmission periods 4 and 6 for the data collection problem. The sink nodes are nodes number 20 and 44.

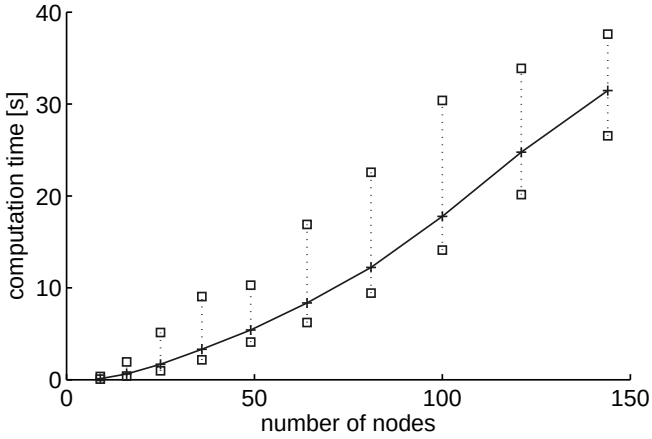


Fig. 2.4.6: Computation times in dependence on number of nodes in the network for fragmented flows.

of the data flow. The vertical lines in node 20 and in node 44 represent the data flow reaching these two sink nodes and remaining in buffer until whole message is received.

The routing in node-time space is shown in Figure 2.4.5. On the horizontal axis, the single nodes are placed and on the vertical axis, the integer time of one network period is placed. It is seen how the messages transmission is repeated during the network period and which data are meeting in the links and how they affect the link capacities. The lines width is a logarithmic function of the volume of the data flow. Please notice that the last layer of the nodes (the 13^{th} layer) is the first layer of next network period and is drown for more transparent visualization.

2.4.2.2 Time Complexity in Relation to the Number of Nodes

To demonstrate the time complexity of our approach, we have varied the experiment of data collection presented in Section 2.4.2.1. The field *size* is gradually set from $[2\times2]$ to $[12\times12]$ (i.e. the number of nodes from 4 to 144). The communication distance is kept at 2 and the communication costs are kept as a power of the node's distance. The number of communication demands is set to 10. Each node has a 50% probability that it will send data of each

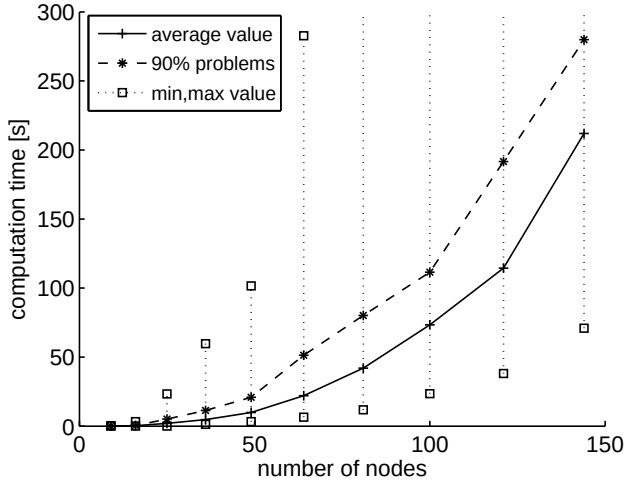


Fig. 2.4.7: Computation times in dependence on number of nodes in the network for integral flow.

communication demand to the corresponding sink node. The link capacities are set as $\mu = \frac{1}{8} \max_{m \in \mathcal{M}} \sum_{n=1}^{N} s_{in,n}^{(m)}$. The deadline for each communication demand is set as $d^{(m)} = \sqrt{N}$. The transmission periods $p^{(m)}$ of the ten communication demands are set in order as: $p^{(1-10)} = [4, 4, 6, 6, 8, 8, 12, 12, 24, 24]$ and the transmission offsets are set to zero $o^{(1-10)} = 0$. The network period is 24.

The simulation has been run 300 times for each number of nodes on randomly constructed networks. The minimum, average and maximum computational times are shown in Figure 2.4.6 for the fragmented data flow and in Figure 2.4.7 for the integral splittable data flow according to Section 2.3.1.4. The maximum values which are not inside the graph axis for the integral flow, are in range from 1420 to 11097 seconds. Moreover, there are presented the solution times which are needed to solve 90% of the problems.

It is seen that in the contrast to the experiments in Section 2.4.2, the solution time for the integral data flow increases much faster than for the fragmented data flow in this case. However, the solution times are still in an acceptable range for a practical application.

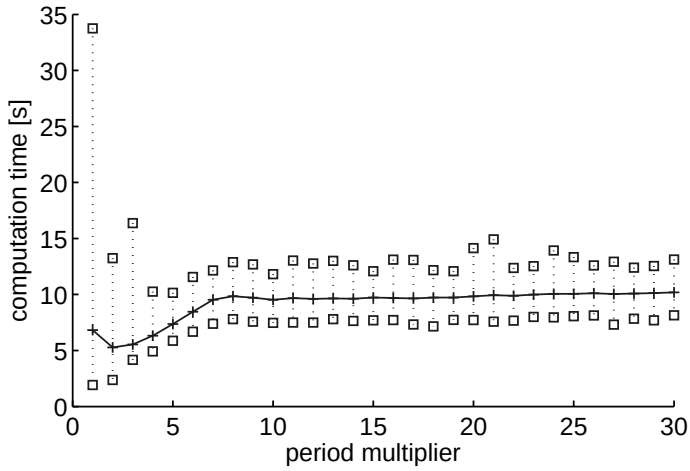


Fig. 2.4.8: Computation times in dependence on the network period for fragmented flows.

2.4.2.3 Time Complexity in Relation to the Network Period

This experiment demonstrates time complexity depending on the network period, which is defined as a least common multiple of periods $p^{(m)}$ over all $m \in \mathcal{M}$.

The experiment is performed in a network with 64 nodes (field *size* [8×8]), communication distance equal to 2, communication costs equal to the power of the nodes distance. Each node has a 50% probability that it will send data of each communication demand to the corresponding sink node. Deadlines are equal to 10 communication hops and the link capacities are set as $\mu = \frac{1}{8} \max_{m \in \mathcal{M}} \sum_{n=1}^{N} s_{in,n}^{(m)}$.

There are 8 communication demands with transmission periods equal to $p^{(1-8)} = [1, 1, 2, 2, 3, 3, 6, 6] \times i$ in the network. The transmission offsets are set to zero $o^{(1-8)} = 0$. During the experiment the multiplier i is gradually increasing from 1 to 30 with as the network period P from 6 to 180.

The progress of computation time is presented in Figure 2.4.8. The interesting result of this experiment is that for multiplier bigger than 9 the computation times are not increasing and stay constant. This means that the presented approach is usable even in case when the transmission periods

of individual demands are much bigger than the hop communication delay. In this case this approach can provide more effective solutions than the approach using continuous flow approximation from Section 2.3. The constant computation time in this experiment is caused by the fact that all transmission periods are increasing together. So, the number of transmissions during the network period is constant during the experiment. When the transmission period increases over the deadline of the communication demand, the next increase of the transmission period do not increase the number of capacity constraints in (2.4.6).

The bigger computational times for small multipliers i are caused by the small transmission periods. For small transmission periods is more suitable the continuous data flow mode in Section 2.3 (especially for transmission period equal to 1 because it is exactly the case of continuous data flow).

2.5 Summary

In this chapter, we have focused on real-time routing in multi-hop networks for both, the problems with continuous data flow (small transmission period) and the problems of periodic messages (the transmission period is bigger than the one hop communication delay). We have used the multi-commodity network flow routing problem and extended it to solve real-time routing in a general multi-hop network. Such an extension is quite simple from the theoretical point of view, but very powerful from the practical point of view since the multi-commodity network flow problem has no notion of time, thus it cannot be used in many applications where the response time is the subject of constraints. Solved by Linear Programming, it exhibits a very good performance, as was shown in our experiments.

Surprisingly, the performance does not degrade even in the presence of an integral flow constraint, which makes the problem NP hard. Being aware of the weaknesses of randomly generated test instances, we have performed several steps making the problem more difficult, but the time complexity of the Integer Linear Programming solution of the integral flow problem was still comparable to the Linear Programming solution of the fragmented flow problem. Up to now, we do not have a satisfactory explanation of this phenomenon, except the conjecture that integral flow is rather simple to achieve in our model, which makes the model a very good candidate for applications.

The other advantage of the presented algorithms is, that the real-time routing is described as a multi-commodity network flow problem with side constraints. The side constraints of the problem are in form, which allow future problem distribution as an in-network algorithm (see Chapter 4).

Chapter 3

Distributed Routing Algorithm

3.1 Introduction

This chapter is focused on a distributed algorithm for data flow routing through the multi-hop wireless sensor network. The objective is to optimize the energy consumption for the data transfer (minimal possible energy consumption over the whole network), while constrained by communication capacities (maximum data volume which can be transferred by link, or device per a time unit).

There are many communication protocols designed for the data routing in wireless sensor networks however, to achieve the exact energy optimal routing which complies with the communication capacities, the system usually needs a central computational point with the knowledge of the actual network structure and parameters (e.g. [Xiao 04]). The existence of such a computational point decreases the robustness of the system against the network damage and increases the communication load of the network. Furthermore, the routing of such information (the actual network structure and parameters) has to be solved in the case of the centralized algorithm.

In this chapter, we propose three distributed algorithms, which compute the energy optimal routing without the need of any central computational or data point. The algorithms suppose that each node knows only the cost (energy consumption per sent data unit) of its outgoing communication links

and the data which it is supposed to send and receive. The algorithms are mathematically derived using the convex optimization theory and proof of their convergence is presented.

The main purpose of this work is to present the principle of new distributed routing algorithms rather than to present application ready algorithms. We believe that the presented approach can lead to new efficient and highly adaptive routing algorithms for sensor networks. Moreover, the approach used in this work can be adapted for general distribution of convex optimization problems into the network.

According to our knowledge, this work is first, which solves the routing problem with linear cost functions, using the dual decomposition.

3.1.1 Related Works

There are several works, which focus on the decomposition of network optimization problems described by strictly convex optimization. A systematic presentation of the decomposition techniques for network utility maximization (NUM) is presented in [Palo 06b, Palo 06a, Chia 07]. The authors present several mathematical approaches to structural decomposition of the NUM problems and classify them. In [Joha 06a, Joha 05, Nama 06] the authors use the dual decomposition to decompose cross-layer optimization problems into optimization of separated layers. The presented approaches lead to structural decomposition (e.g. to routing layer, capacity layer...) which is not suitable for derivation of the in-network distributed algorithm. In [Joha 08] a general distributed algorithm for strictly convex optimization problems with common parameter for all nodes is presented.

The decomposition of an optimal routing problem is presented e.g. in [Tsit 86, Low 99], where the authors have focused on the node-path formulation of the routing problem and use the dual decomposition to find the distributed algorithm. The presented algorithms can be described as a negotiation between the source node and the path load. This approach is suitable for problems with a small number of communication paths. However, in sensor networks routing problems, where many possible communication paths exist, we have to find a different way to distribute the routing algorithm.

In [Zhen 10] the authors independently derived a distributed routing algorithm, which is based on the node-link problem formulation and so it uses similar mechanisms as the algorithm presented in this chapter. However, the

algorithm is limited to strictly convex objective function.

All the algorithms referenced in this section are limited to strictly convex objective functions and fail in the case of linear objective functions which make the problems and convergence proof more difficult. According to our knowledge, this work is the first one, which addresses and solves the problem of the dual decomposition of NUMs for problems with linear objective functions.

3.1.2 Multi-Commodity Network Flow Model

During this chapter, we use the multi-commodity network flow model formulated as the Linear Programming to describe the basic routing problem. The model is identical to the model introduced in Section 2.2 and we write it as:

$$
\begin{aligned}
&\min_{x} \quad \vec{c}^{T} \sum_{m\in\mathcal{M}} \vec{x}^{(m)} \\
&\textbf{subject to:} \\
&A^{-}\vec{x}^{(m)} + \vec{s}_{out}^{(m)} = A^{+}\vec{x}^{(m)} + \vec{s}_{in}^{(m)} \qquad \forall m \in \mathcal{M} \qquad (3.1.1)\\
&D \sum_{m\in\mathcal{M}} \vec{x}^{(m)} \leq \vec{\mu} \\
&\vec{x}^{(m)} \geq \vec{0} \qquad \forall m \in \mathcal{M}
\end{aligned}
$$

The vectors and matrices are defined as in Section 2.2. The vector $\vec{c} > 0$ is a column vector of the energy consumption per sent data unit. The column vector $\vec{s}_{in}^{(m)} \geq \vec{0}$ denotes the flow coming into the network, the $\vec{s}_{out}^{(m)} \geq \vec{0}$ denotes the flow leaving the network and the $\vec{x}^{(m)} \geq \vec{0}$ denotes the flow routed through the network for demand m. The matrices A^{+} and A^{-} are incidence matrices for incoming and leaving links defined according Equation (2.2.1) and (2.2.2). The matrix D and the column vector $\vec{\mu}$ describe the capacity constraints. If there is a separate capacity for each communication link, matrix D is the identity matrix of size $[L \times L]$ and $\vec{\mu} \geq \vec{0}$ consists of the link capacities. For the node capacities the $\vec{\mu} \geq \vec{0}$ is column vector of the node capacities and matrix D is $D_{n,l} = 1$ for $l \in \mathcal{O}(n)$ (link l leaves node n) and $D_{n,l} = 0$ otherwise. (i.e. $D = A^{-}$)

3.1.3 Proximal-Point Method

To decompose the routing algorithm into the network, we use a gradient optimization method to solve its dual problem. However, the linearity of the objective function of the routing problem 3.1.1 would cause oscillations in the algorithm and prevents us finding the optimal solution. We use the proximal-point method (for details see [Bert 98]) to modify the problem into strictly convex form, which allows the usage of the gradient method.

The proximal-point method is an iterative method described as:

$$\vec{x}_{k+1} = \arg\min_{\vec{x}\in\mathcal{C}} \Big(f(\vec{x}) + \varepsilon\big(\vec{x} - \vec{x}_k\big)^T \big(\vec{x} - \vec{x}_k\big) \Big) \tag{3.1.2}$$

Where $\mathcal{C}$ is a convex set, $f(\vec{x})$ is a convex function and the $\varepsilon > 0$ is a constant. Please note, that each solution x_k of the iteration (3.1.2) is a feasible solution and that for each $k \in \mathcal{N}$ holds $f(\vec{x}_k) \geq f(\vec{x}_{k+1})$. If $k \to \infty$ then $\min_{\vec{x}\in\mathcal{C}}(f(\vec{x}) + \varepsilon(\vec{x} - \vec{x}_k)^T(\vec{x} - \vec{x}_k)) \to \min_{\vec{x}\in\mathcal{C}} f(\vec{x})$. The iteration (3.1.2) is called the outer iteration or the proximal-point iteration in this work.

3.2 Two Loops Distributed Routing Algorithm

We derive a *Two Loops Distributed Routing Algorithm* (TLDRA) in this section. The derivation is based on the multi-commodity network flow model (3.1.1) for the case with separate capacity for each communication link, where the matrix D is an identity matrix.

3.2.1 Mathematical Derivation of the Two Loops Distributed Routing Algorithm

By applying the proximal point method (3.1.2) to the energy optimal routing problem (3.1.1), substituting $\vec{x}_k$ by $\vec{x''}$ and $A = (A^+ - A^-)$, we can write the problem for one proximal-point iteration in form:

$$\begin{aligned}
&\min_{x} \sum_{m\in\mathcal{M}} \left(\vec{c}^T\vec{x}^{(m)} + \varepsilon(\vec{x}^{(m)} - \vec{x''}^{(m)})^T(\vec{x}^{(m)} - \vec{x''}^{(m)})\right) \\
&\textbf{subject to:} \\
&\qquad A\vec{x}^{(m)} = \vec{s}_{out}^{(m)} - \vec{s}_{in}^{(m)} \qquad \forall m \in \mathcal{M} \\
&\sum_{m\in\mathcal{M}} \vec{x}^{(m)} \leq \vec{\mu} \\
&\qquad \vec{x}^{(m)} \geq \vec{0} \qquad \forall m \in \mathcal{M}
\end{aligned} \tag{3.2.1}$$

Where $\vec{x''}$ is the proximal-point variable with a fixed value from the proximal-point iteration.

3.2.1.1 Dual Problem

Please note that according to Slater's conditions (see e.g. [Boyd 04]), the strong duality holds for problem (3.2.1) (i.e. the optimal values of the dual and the primal problem are equal).
The Lagrangian function of the system of inequalities (3.2.1) is:

$$\begin{aligned}
L(\vec{x}^{(m)}, \vec{x''}^{(m)}, \vec{\lambda}, \vec{\theta}^{(m)}) &= \sum_{m\in\mathcal{M}} \left(\vec{c}^T\vec{x}^{(m)} + \varepsilon(\vec{x}^{(m)} - \vec{x''}^{(m)})^T(\vec{x}^{(m)} - \vec{x''}^{(m)})\right) \\
&+ \sum_{m\in\mathcal{M}} \vec{\theta}^{(m)^T}(A\vec{x}^{(m)} - \vec{s}_{out}^{(m)} + \vec{s}_{in}^{(m)}) + \vec{\lambda}^T(\sum_{m\in\mathcal{M}} \vec{x}^{(m)} - \vec{\mu})
\end{aligned} \tag{3.2.2}$$

Where $\vec{x}^{(m)} \geq \vec{0}$ is the primal variable, $\vec{\lambda} \geq \vec{0}$ and $\vec{\theta}^{(m)}$ are the dual variables.

The dual function W is:

$$W(\vec{x}''^{(m)}, \vec{\lambda}, \vec{\theta}^{(m)}) = \min_{\vec{x}^{(m)} \geq \vec{0}} L(\vec{x}^{(m)}, \vec{x}''^{(m)}, \vec{\lambda}, \vec{\theta}^{(m)}) \tag{3.2.3}$$

Differentiation of the Lagrangian function (3.2.2) gives:

$$\frac{\partial L}{\partial \vec{x}^{(m)}} = \vec{c} + \vec{\lambda} + A^T \vec{\theta}^{(m)} + 2\varepsilon \vec{x}^{(m)} - 2\varepsilon \vec{x}''^{(m)} \tag{3.2.4}$$

And the minimizer $\vec{x}^{(m)}_{min}$ of the dual function (3.2.3):

$$\vec{x}^{(m)}_{min} = \left[\ \vec{x}''^{(m)} - \frac{1}{2\varepsilon}(\vec{c} + \vec{\lambda} + A^T \vec{\theta}^{(m)}) \ \right]^+ \tag{3.2.5}$$

Where $[..]^+$ denotes a positive or zero value in each component $[..]^+ = max(\vec{0}, ..)$.
Then the dual problem of (3.2.1) is:

$$U(\vec{x}''^{(m)}) = \max_{\vec{\theta}^{(m)}, \vec{\lambda} \geq 0} W(\vec{x}''^{(m)}, \vec{\lambda}, \vec{\theta}^{(m)}) \tag{3.2.6}$$

And the dual function gradients are:

$$\frac{\partial W}{\partial \vec{\lambda}} = \sum_{m \in \mathcal{M}} \vec{x}^{(m)}_{min} - \vec{\mu} \tag{3.2.7}$$

$$\frac{\partial W}{\partial \vec{\theta}^{(m)}} = A\vec{x}^{(m)}_{min} - \vec{s}^{(m)}_{out} + \vec{s}^{(m)}_{in} \tag{3.2.8}$$

3.2.1.2 Dual Gradient Algorithm

To solve the routing problem, we put together the proximal-point method (3.1.2) and the dual problem (3.2.6). The created algorithm consists of two nested iterations. The internal iteration is the gradient iteration which solves the dual problem (3.2.6). The outer iteration is the proximal-point iteration corresponding to Equation (3.1.2). The algorithm structure is presented in Figure 3.2.1.

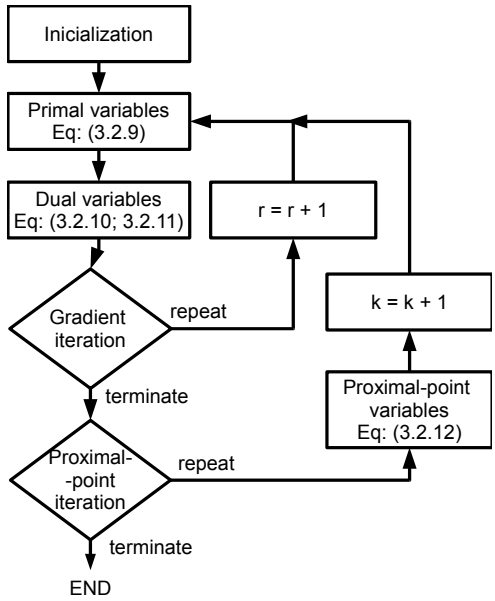


Fig. 3.2.1: TLDRA: Iteration algorithm

The termination of the iterations could be done through sophisticated methods based on the progress of the dual variables in combination with the global communication. However, for the verification of the concept of this algorithm it is more practical to use a constant number of proximal-point iterations based on a heuristic experiences. To the same way we set the number of the gradient iterations to decrease proportioning to the index of the proximal-point iterations.

We denote a counter of the gradient iteration as r and a counter of the proximal-point iteration as k. The constants $R(k)$ and K denote the number of cycles of the corresponding iterations. The constant $\alpha > 0$ is the step size of the gradient algorithm. The dual gradient algorithm is presented in Table 3.2.1.

In the first step of the dual gradient algorithm in Table 3.2.1, the variables are set to arbitrarily initial values. The closer the values are to the final solution, the faster the convergence of the algorithm becomes. This property can be used in the case of minor changes of the network structure during

1. Initialize the dual and the proximal-point variables:

$$r := 1, \quad k := 1, \quad \vec{\theta}^{(m)} := \vec{\theta}^{(m)}_{start},$$
$$\vec{\lambda} := \vec{\lambda}_{start}, \quad \vec{x}''^{(m)} := \vec{x}''^{(m)}_{start}$$

2. Evaluate the primal variables $\vec{x}^{(m)}_{min}$:

$$\vec{x}^{(m)}_{min} := \Big[\ \vec{x}''^{(m)} - \frac{1}{2\varepsilon}\big(\vec{c} + \vec{\lambda} + A^T\vec{\theta}^{(m)}\big)\ \Big]^+ \tag{3.2.9}$$

3. Modify the dual variables according to the gradient of the dual function (3.2.7)

$$\vec{\lambda} := \Big[\vec{\lambda} + \alpha\big(\sum_{m\in\mathcal{M}} \vec{x}^{(m)}_{min} - \vec{\mu}\big)\Big]^+ \tag{3.2.10}$$

$$\vec{\theta}^{(m)} := \vec{\theta}^{(m)} + \alpha\big(A\vec{x}^{(m)}_{min} - \vec{s}^{(m)}_{out} + \vec{s}^{(m)}_{in}\big) \tag{3.2.11}$$

4. $r := r + 1$; Go to step 2 and start a new cycle of the gradient iteration. Repeat $R(k)$-times.
5. Start new cycle of the proximal-point iteration:
   - Set the iteration and the proximal-point variables:

$$r := 1, \quad k := k + 1$$
$$\vec{x}''^{(m)} := \vec{x}^{(m)}_{min} \tag{3.2.12}$$

   - Go to step 2.
     Repeat the proximal-point iteration K-times.
6. Read the result routing: $\vec{x}^{(m)} = \vec{x}^{(m)}_{min}$

Table 3.2.1: TLDRA: Dual Gradient Algorithm

its operation or in case of a pre-computed routing e.g. based on Dijkstra's algorithm. However, we will not investigate this problem further in this section. During the experiments in this section we initiate the variables to zero.

3.2.1.3 Distributed Algorithm

The presented distributed algorithm is running on each node in the network. The algorithm is synchronized by the communication between the nodes. The structure of the computation of the distributed algorithm is the same

as the structure of the Dual Gradient Algorithm in Figure 3.2.1, only the communication is added.

We use $x_{min,i}^{(m)}$, $x_i''^{(m)}$, λ_i, c_i etc. to denote the i-th component of the corresponding vector.

Due to the structure of matrix A (see Section 2.2) we rewrite the expression (3.2.9) in order to compute the flow of the communication demand m in the link l as:

$$x_{min,l}^{(m)} = \Big[\ x_l''^{(m)} - \frac{1}{2\varepsilon}\big(c_l + \lambda_l + \theta_{l^+}^{(m)} - \theta_{l^-}^{(m)}\big)\ \Big]^+ \tag{3.2.13}$$

Where the expression l^- denotes index of the start node of the link l and l^+ denotes index of the end node of the link l. I.e. $l \in \mathcal{O}(l^-)$ and $l \in \mathcal{I}(l^+)$ Similarly we can rewrite the expressions (3.2.10, 3.2.11):

$$\lambda_l \ = \ \Big[\lambda_l + \alpha\big(\sum_{m\in\mathcal{M}} x_{min,l}^{(m)} - \mu_l\big)\Big]^+ \tag{3.2.14}$$

$$\theta_n^{(m)} \ = \ \theta_n^{(m)} + \alpha\big(\sum_{i\in\mathcal{I}(n)} x_{min,i}^{(m)} - \sum_{i\in\mathcal{O}(n)} x_{min,i}^{(m)} - s_{out,n}^{(m)} + s_{in,n}^{(m)}\big) \tag{3.2.15}$$

Each node n is responsible for computation of the flow volume of the links starting in the node n and for the corresponding dual variables. Therefore, node n computes $x_{min,l}^{(m)}$ for all $l \in \mathcal{O}(n)$ and all $m \in \mathcal{M}$, λ_l for all $l \in \mathcal{O}(n)$ and $\theta_n^{(m)}$ for all $m \in \mathcal{M}$.

In (3.2.13), node n computes $x_{min,l}^{(m)}$ for all links leaving node n. It is a function of the local variables x'', λ, θ and the variables θ of the neighbor nodes. Similarly, the computation of λ_l and $\theta_n^{(m)}$ in (3.2.14, 3.2.15) is a function of the local variables and the variables of the neighbor nodes that are within a one hop communication distance.

The algorithm for node n is presented in Table 3.2.2. In step 1, the algorithm initializes the variables. Similarly as in Table 3.2.1, the closer the values are to the optimal solution the less cycles of the iterations are needed. In steps 2 and 3, the node communicates the values of the proximal-point variables and the dual variables. In step 4, the node computes $x_{min,l}^{(m)}$ for the links leaving the node. In step 5, the node computes $x_{min,l}^{(m)}$ for the links

1. Initialize the variables:
$$\begin{aligned} x_l^{\prime\prime(m)} &:= x_{start,l}^{\prime\prime(m)} && \forall m \in \mathcal{M} \quad \forall l \in \mathcal{O}(n) \\ \theta_n^{(m)} &:= \theta_{start,n}^{(m)} && \forall m \in \mathcal{M} \\ \lambda_l &:= \lambda_{start,l} && \forall l \in \mathcal{O}(n) \end{aligned}$$

2. Send/receive the proximal-point variables to/from the neighbors.
$$\begin{aligned} \textbf{Send:} \quad & x_l^{\prime\prime(m)} && \forall m \in \mathcal{M}, \forall l \in \mathcal{O}(n) \\ \textbf{Receive:} \quad & x_l^{\prime\prime(m)} && \forall m \in \mathcal{M}, \forall l \in \mathcal{I}(n) \end{aligned}$$

3. Send/receive the dual variables to/from the neighbors.
$$\begin{aligned} \textbf{Send:} \quad & \lambda_l && \forall l \in \mathcal{O}(n) \\ & \theta_n^{(m)} && \forall m \in \mathcal{M} \\ \textbf{Receive:} \quad & \lambda_l && \forall l \in \mathcal{I}(n) \\ & \theta_{l^-}^{(m)} && \forall m \in \mathcal{M}, \forall l \in \mathcal{I}(n) \end{aligned}$$

4. Evaluate the primal variables $x_{min,l}^{(m)}$ for $\forall l \in \mathcal{O}(n)$ and $\forall m \in \mathcal{M}$:
$$x_{min,l}^{(m)} := \Big[\; x_l^{\prime\prime(m)} - \frac{1}{2\varepsilon}\big(c_l + \lambda_l + \theta_{l^+}^{(m)} - \theta_n^{(m)}\big) \;\Big]^+$$

5. Evaluate the neighbors primal variables $x_{min,l}^{(m)}$ for $\forall l \in \mathcal{I}(n)$ and $\forall m \in \mathcal{M}$:
$$x_{min,l}^{(m)} := \Big[\; x_l^{\prime\prime(m)} - \frac{1}{2\varepsilon}\big(c_l + \lambda_l + \theta_n^{(m)} - \theta_{l^-}^{(m)}\big) \;\Big]^+$$

6. Modify the dual variables.
$$\begin{aligned} \lambda_l &:= \Big[\lambda_l + \alpha\big(\sum_{m\in\mathcal{M}} x_{min,l}^{(m)} - \mu_l\big)\Big]^+ && \forall l \in \mathcal{O}(n) \\ \theta_n^{(m)} &:= \theta_n^{(m)} + \alpha\big(\sum_{i\in\mathcal{I}(n)} x_{min,i}^{(m)} - \sum_{i\in\mathcal{O}(n)} x_{min,i}^{(m)} - s_{out,n}^{(m)} + s_{in,n}^{(m)}\big) && \forall m \in \mathcal{M} \end{aligned}$$

7. Go to step 3 and start a new cycle of the gradient iteration. Repeat $R(k)$-times.

8. Start new cycle of the proximal-point iteration:
   - Preserve the dual variables $\theta_n^{(m)}$ and λ_l for $\forall l \in \mathcal{O}(n)$ and $\forall m \in \mathcal{M}$ and set the proximal-point variables:
   $$x_l^{\prime\prime(m)} := x_{min,l}^{(m)} \qquad \forall l \in \mathcal{O}(n), \forall m \in \mathcal{M}$$
   - Go to step 2. Repeat the proximal-point iteration K-times.

9. Each node n knows the routing of the outgoing flow.
   $x_l^{(m)} = x_{min,l}^{(m)}$ for $\forall l \in \mathcal{O}(n)$ and $\forall m \in \mathcal{M}$.

Table 3.2.2: TLDRA: Distributed Routing Algorithm executed in node n

entering the node from its neighbors. In step 6, the node modifies the dual variables. Steps 7 and 8 start the new iteration cycles of the algorithm. Please notice, in the step 5 we compute the variables $x_{min,l}^{(m)}$ which has already been computed in neighboring nodes in step 4. Due to this computation, we can save a communication of the primal variables.

3.2.1.4 Node Communication Capacities

In some applications, especially in wireless networks, the node communication capacity (maximum data volume which can be transmitted by a node per a time unit) is more appropriate than the link communication capacity, used in this section. The link communication capacity has been used during the algorithm derivation because of the more transparent presentation and because of the more general behavior of the algorithm (e.g. in case of the GTS slots allocation in IEEE 802.15.4 where each link has its capacity assigned). The node capacity can be easily transformed to the link capacity by the graph transformation where each node is replaced by two nodes connected by a link with the given capacity. However, in this section we present a direct transformation of the equations of the presented algorithm to implement the node capacities more efficiently.

To describe the node capacity constraints, we define the new matrix D as:

$$D_{n,l} = \begin{cases} 1, & l \in \mathcal{O}(n) \text{ (link } l \text{ leaves node } n) \\ 0, & \text{otherwise} \end{cases} \tag{3.2.16}$$

A new form of the communication capacity constraints is:

$$D \sum_{m \in \mathcal{M}} \vec{x}^{(m)} \leq \vec{\mu} \tag{3.2.17}$$

where $\vec{\mu} \in \mathcal{R}^N$ is a vector of the node capacities for all the nodes in the network.

These changes are directly projected into Equations in 3.2.1, 3.2.2, 3.2.4 and 3.2.7). The changes in the dual gradient algorithm in Table 3.2.1 are in Equations (3.2.9 and 3.2.10). The new form of Equation (3.2.9) is:

$$\vec{x}_{min}^{(m)} := \left[\; \vec{x}''^{(m)} - \frac{1}{2\varepsilon}\left(\vec{c} + D^T\vec{\lambda} + A^T\vec{\theta}^{(m)}\right) \; \right]^+ \tag{3.2.18}$$

and the new form of Equation (3.2.10) is:

$$\vec{\lambda} := \Big[\vec{\lambda} + \alpha\big(\sum_{m\in\mathcal{M}} D\vec{x}^{(m)}_{min} - \vec{\mu}\big)\Big]^{+} \tag{3.2.19}$$

Similarly, the changes of the distributed routing algorithm executed in node n in Table 3.2.2 are the following:
In step (4):

$$x^{(m)}_{min,l} := \Big[\ x''^{(m)}_{l} - \frac{1}{2\varepsilon}\big(c_l + \lambda_n + \theta^{(m)}_{l^+} - \theta^{(m)}_{n}\big)\ \Big]^{+} \tag{3.2.20}$$

In step (5):

$$x^{(m)}_{min,l} := \Big[\ x''^{(m)}_{l} - \frac{1}{2\varepsilon}\big(c_l + \lambda_{l^-} + \theta^{(m)}_{n} - \theta^{(m)}_{l^-}\big)\ \Big]^{+} \tag{3.2.21}$$

In step (6):

$$\lambda_n := \Big[\lambda_n + \alpha\big(\sum_{l\in\mathcal{O}(n)}\sum_{m\in\mathcal{M}} x^{(m)}_{min,l} - \mu_n\big)\Big]^{+} \tag{3.2.22}$$

While applying these changes, the algorithm uses one variable λ_n for each node instead of one variable λ_l for each link. Using a similar transformation of the equations, the algorithm can be updated to implement many different communication constraints (e.g. both oriented link capacities, node incoming and transmitting capacities...) and their combinations.

3.2.2 Experiments

We have performed several experiments in Matlab, to present a behavior of the presented algorithm. We have focused on a data collection problem in this section (i.e. multi-commodity multi-source, mono sink problem). The experiments are based on the same network structure as in previous Chapter 2.

The random networks have been constructed as follows: We consider a square field of size $[size \times size]$, where the $size$ is changing during the experiments. The field is divided into sub-squares of size $[1 \times 1]$. One node is randomly placed into each sub-square and the communication distance is set to 2 (i.e. node A can communicate with node B, if and only if their Euclidean distance is less than 2). The network is close to the "unit-disk network" [Rese 06]. The communication costs per transmitted data flow unit have been set as the power of the distance between the nodes.

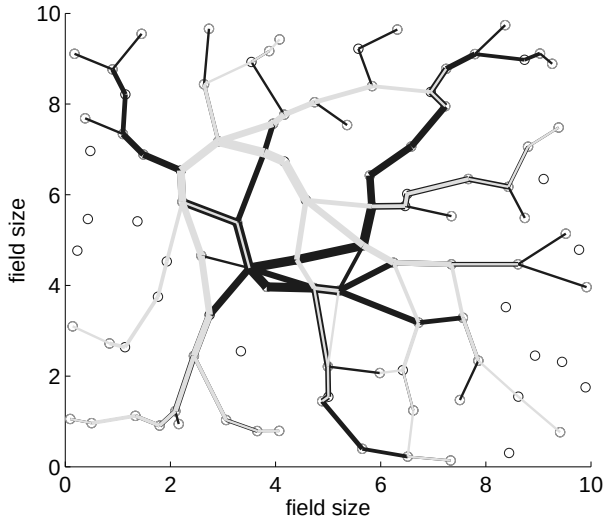


Fig. 3.2.2: TLDRA: Optimal data flow routing

3.2.2.1 Algorithm Presentation

To present the resulting optimal data flow routing in the network and the progress of the Lagrangian function during the computation, we have performed an experiment based on the network described above. The field *size* has been set to 10 (i.e. there are 100 nodes in the network). There are two communication demands in the network, each of them with a different sink node. Each node has a 60% probability that it will send data of the first communication demand of volume 1 to the first sink node and a 40% probability that it will send data of the second communication demand of volume 1 to the second sink node (i.e. $s_{in,n}^{(1)} \in \{0,1\}$ and $s_{in,n}^{(2)} \in \{0,1\}$). The link capacities have been set to $\mu_l = \frac{1}{2} \max_{m \in \mathcal{M}} \sum_{n=1}^{N} s_{in,n}^{(m)}$. The constants of the algorithm have been set as: $\alpha = 0.05$, $\epsilon = 0.2$. The initial values $x_{start,l}^{\prime\prime(m)}$, $\theta_{start,n}^{(m)}$, $\lambda_{start,l}$ have been set to 0.

The optimal data flow routing is shown in Figure 3.2.2, where the first communication demand is in black and the second communication demand in

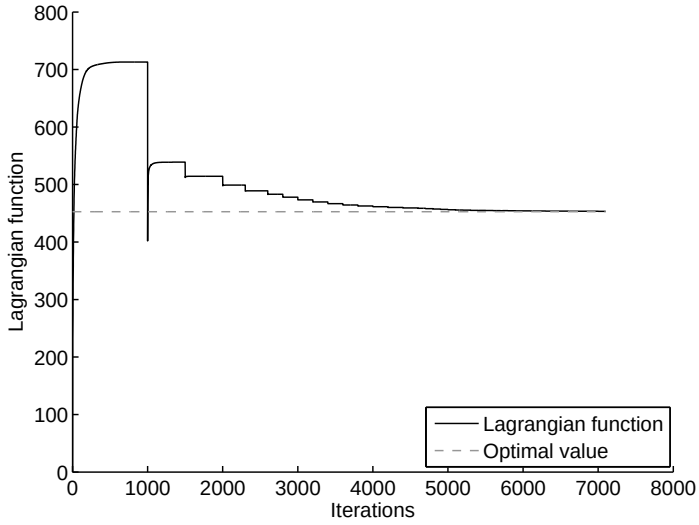


Fig. 3.2.3: TLDRA: Progress of Lagrangian function (3.2.2)

gray. The progress of the Lagrangian function (3.2.2) is presented in Figure 3.2.3. The Lagrangian function is in black and its final value in gray. (The final value was computed separately by a centralized algorithm for evaluation purposes only.) The nesting of the gradient iteration and the proximal-point iteration can be seen in Figure 3.2.3. The internal iteration (i.e. the gradient iteration) maximize the Lagrangian function using the dual variables $\theta_n^{(m)}$ and λ_l with constant variable $x_l''^{(m)}$. The outer iteration (i.e. the proximal-point iteration) minimize the Lagrangian function using the variable $x_l''^{(m)}$.

3.2.2.2 Number of Iterations

To demonstrate the statistical behavior of the algorithm, we have performed several tests in networks of different size. We have set the field *size* gradually from 3 to 10 (i.e. from 9 to 100 nodes), the number of communication demands have been set to 10 (multi-source, mono-sink) with random sinks. Each node has a 50% probability to send the data flow for each communication demand with volume 1. The link capacities have been set to

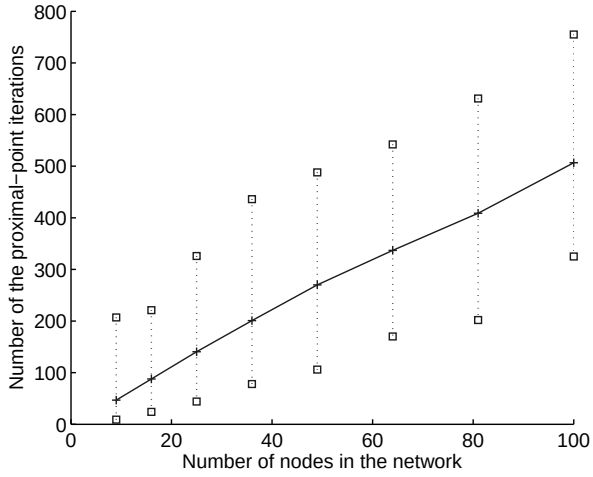


Fig. 3.2.4: TLDRA: Number of proximal-point iterations needed to achieve 0.01% deviation from the optimal value.

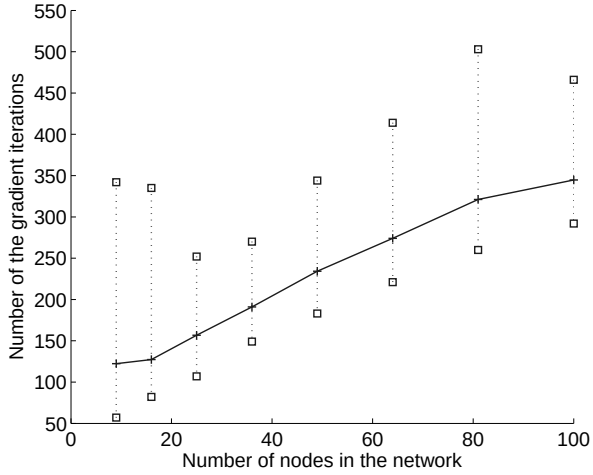


Fig. 3.2.5: TLDRA: Number of gradient iterations in the 1st proximal-point iteration needed to achieve 0.01% deviation from the optimal value.

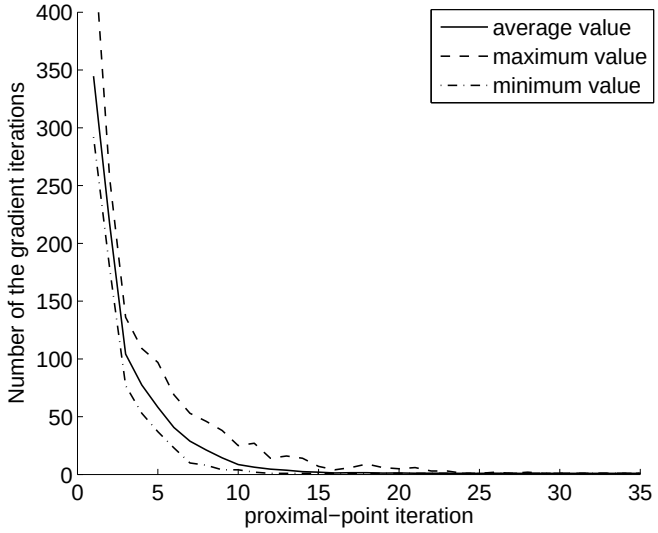


Fig. 3.2.6: TLDRA: Number of the gradient iterations in dependence on the index of the proximal-point iteration for the field *size* 10.

$\mu_l = \frac{1}{2} \max_{m \in \mathcal{M}} \sum_{n=1}^{N} s_{in,n}^{(m)}$. The constants of the algorithm have been set as: $\alpha = 0.1$, $\epsilon = 0.6$. The initial values $x_{start,l}^{\prime\prime(m)}$, $\theta_{start,n}^{(m)}$, $\lambda_{start,l}$ have been set to 0. The computation has been repeated, on random networks, 100 times for each field *size*. The numbers of the iterations $R(k)$ and K have been set to sufficiently large values in order to ensure that each gradient iteration achieves an optimal solution for the given proximal-point variables and that the final solution is the energy optimal routing. The results have been evaluated as a maximum, average and minimum number of iterations needed to achieve a 0.01% deviation of the Lagrangian function from the optimal value. (the optimal value was computed separately by a centralized algorithm for evaluation purposes only)

In Figure 3.2.4, the number of needed proximal-point iterations, in dependence on the number of nodes in the network, is presented.

In Figure 3.2.5, the number of needed repetitions of the gradient algorithm in the first cycle of the proximal-point iteration, in dependence on the number

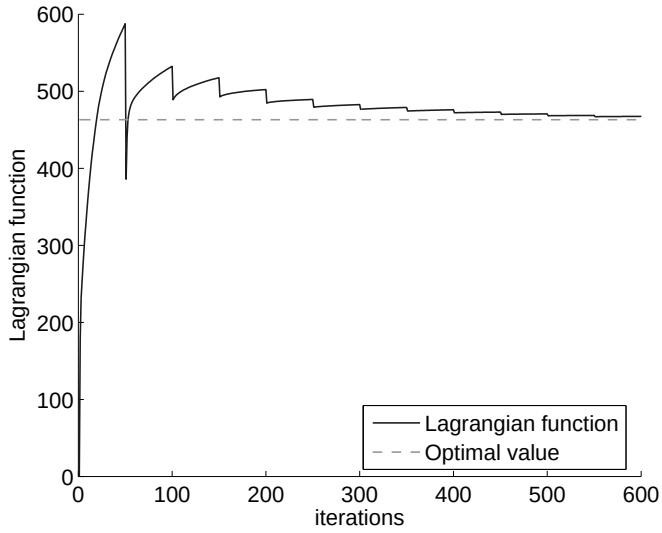


Fig. 3.2.7: TLDRA: Lagrangian progress for incomplete gradient iterations.

of nodes in the network, is presented.

In Figure 3.2.6, the number of needed repetitions of the gradient algorithm, in dependence on the index of the proximal-point method for field *size* 10, is presented. The number of the needed gradient iterations decreases rapidly during the computation. Only the first 35 proximal-point iterations are presented, because the next values are equal to 1.

3.2.2.3 Robustness

The presented approach requires that the gradient iteration ends with an optimal solution for a given proximal-point variables. Due to the distribution of the algorithm, the termination of the gradient iteration is problematic and in this chapter we use heuristic constants for the number of repetitions. Being aware of the problems with the algorithm termination we have performed several simulations to evaluate the robustness of the approach. We have focused on the case when the gradient iterations do not reach the optimal solution for the given proximal-point variables in the given number of

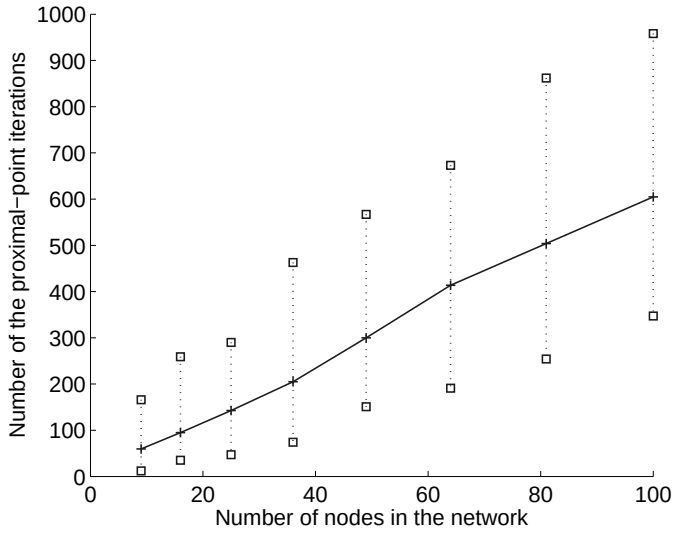


Fig. 3.2.8: Number of proximal-point iterations needed to achieve 0.01% deviation from the optimal value in the case of an incomplete gradient iteration.

iterations. To ensure this behavior, we have set the number of repetitions of the gradient iteration to 50 ($R(k) = 50 \quad \forall 0 < k < K$). An example of progress of the Lagrangian function for the first 12 proximal-point iteration (i.e. 600 gradient iterations) for $R(k) = 50$ is presented in Figure 3.2.7. All parameters for this simulation were the same as in the experiment in Section 3.2.2.1, except $R(k)$. In Figure 3.2.7, it is seen that some of the first gradient iterations do not reach the optimal solution for the given proximal-point variables within the 50 repetitions. However the algorithm still converges to the final optimal solution. Moreover, it converges even faster, than in Figure 3.2.3 since it does not spend that much time while searching for the optimal solution for the given proximal-point variables.

The experiment has been repeated 50 times for each field *size* and the results have been evaluated as a maximum, average and minimum number of iterations needed to achieve a 0.01% deviation of the Lagrangian function from the optimal value. The number of needed proximal-point iterations, in dependence on the number of nodes in the network, is presented in Figure

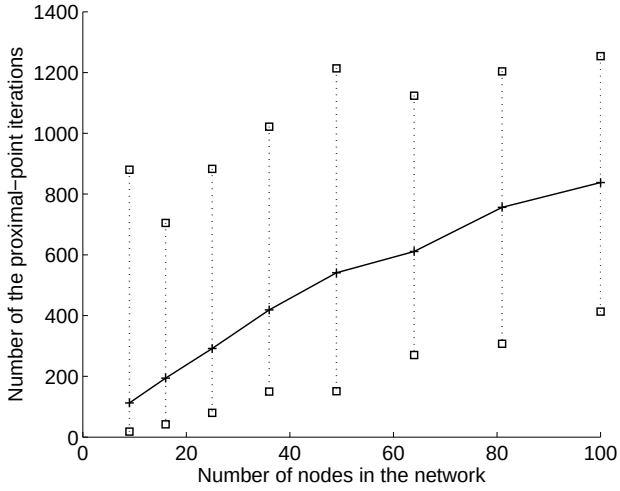


Fig. 3.2.9: Number of proximal-point iterations needed to achieve 0.01% deviation from the optimal value for the node communication capacities.

3.2.8. It is seen that the number of the needed iterations is slightly bigger than the number in the previous Section 3.2.2.2 (see Figure 3.2.4). However the algorithm still converges to the final optimal solution.

3.2.2.4 Node Communication Capacities

To demonstrate the behavior of the algorithm in the case of the node communication capacities, described in Section 3.2.1.4, we have performed an experiment with the same parameters as in Section 3.2.2.2, except for the capacity constraints. The link capacities have been replaced by the node capacities $\mu_l = 3 \max_{m \in \mathcal{M}} \sum_{n=1}^{N} s_{in,n}^{(m)}$. The computation has been repeated on random networks 100 times for each field *size*.

The statistical results are presented in Figure 3.2.9, 3.2.10 and 3.2.11 . It is seen, that the progress of the number of the iterations of the algorithm for the node capacities is similar to the problem with the link capacities.

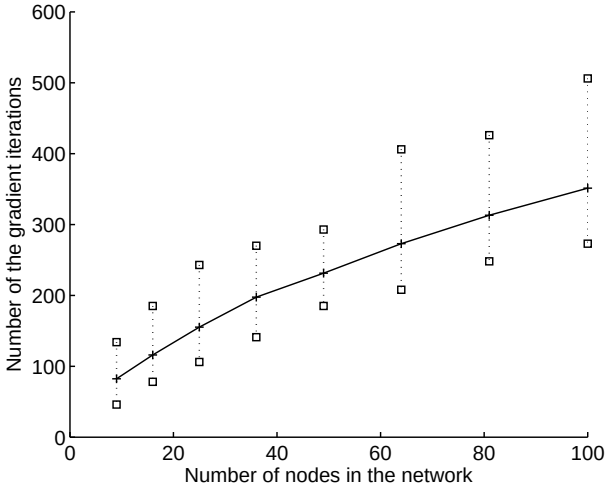


Fig. 3.2.10: Number of gradient iterations in the 1st proximal-point iteration needed to achieve 0.01% deviation from the optimal value for the node capacities.

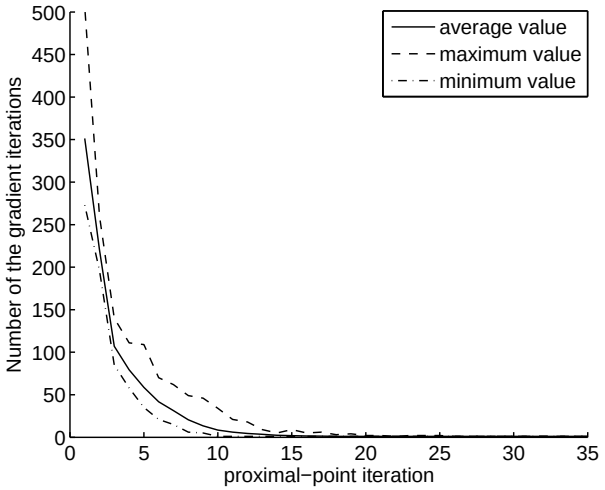


Fig. 3.2.11: Number of the gradient iterations in dependence on the index of the proximal-point iteration for the field *size* 10, for the node capacities.

3.3 One Loop Distributed Routing Algorithm with Incremental flow update

In this section, we derive a distributed algorithm, which consist of only one iteration loop and we prove its convergence. The algorithm is called *One Loop Distributed Routing Algorithm with Incremental flow update* (OLDRAi) and it is based on a subgradient algorithm with three nested iteration loops. In contrast to the algorithm TLDRA derived in previous Section 3.2 the OLDRAi do not set the flow variable x to the optimal value, but x is updated incrementally. We focus on the link capacity case in this section.

3.3.1 Mathematical Derivation of the One Loop Distributed Routing Algorithm with Incremental flow update

Without loss of generality, we rewrite the routing problem into the equality form for a more transparent presentation.

$$\begin{aligned} &\min_{\vec{x}} \quad \vec{c}^T\vec{x} \\ &\textbf{subject to:} \\ &A\vec{x} \;=\; \vec{b} \\ &\vec{x} \;\geq\; \vec{0} \end{aligned} \tag{3.3.1}$$

Where

$$A = \begin{pmatrix} A' & 0 & 0 & 0 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & A' & 0 \\ I & I & I & I \end{pmatrix} \quad \vec{b} = \begin{bmatrix} \vec{s}_{out}^{(1)} - \vec{s}_{in}^{(1)} \\ \vdots \\ \vec{s}_{out}^{(M)} - \vec{s}_{in}^{(M)} \\ \vec{\mu} \end{bmatrix} \tag{3.3.2}$$

$$\vec{x} = \begin{bmatrix} \vec{x}^{(1)} \\ \vdots \\ \vec{x}^{(M)} \\ \vec{z} \end{bmatrix} \quad \vec{c} = \begin{bmatrix} \vec{c'} \\ \vdots \\ \vec{c'} \\ \vec{0} \end{bmatrix} \tag{3.3.3}$$

I is the identity matrix, $A' = (A^+ - A^-)$ and $\vec{c'}$ is the original vector of the communication costs per transmitted data unit over all links in the network (it was marked as $\vec{c}$ in previous Section 3.2 and in Chapter 2).

Similarly as in previous Section 3.2 we use the proximal-point method (3.1.2) to modify the problem into strictly convex form, which allows the usage of the gradient method. The modified problem is:

$$\begin{array}{rl} \min\limits_{\vec{x}''} \min\limits_{\vec{x}} & \vec{c}^T\vec{x} + \varepsilon(\vec{x}-\vec{x}'')^T(\vec{x}-\vec{x}'') \\ \textbf{subject to:} & \\ A\vec{x} & = \vec{b} \\ \vec{x} & \geq \vec{0} \end{array} \tag{3.3.4}$$

where $\varepsilon > 0$.

The set of optimal solutions for problem (3.3.4) is the same as for problem (3.3.1). For optimal solution of problem (3.3.4) holds $\vec{x} = \vec{x}''$. The routing problem has been separated into two nested subproblems. The internal subproblem is minimization over the variable $\vec{x}$ and it is strictly convex. The outer subproblem minimize the internal one by the proximal-point variable $\vec{x}''$.

3.3.1.1 Dual Problem

As in Section 3.2.1.1, to solve the internal subproblem of (3.3.4) (minimization over the variable $\vec{x}$) we present its dual problem to derive the distributable gradient algorithm. According to Slater's conditions (see e.g. [Boyd 04]) the optimal solution of the dual and primal problems are equal.

The Lagrangian function of problem (3.3.4) is:

$$L(\vec{x},\vec{x}'',\vec{\theta}) = \vec{c}^T\vec{x} + \varepsilon(\vec{x}-\vec{x}'')^T(\vec{x}-\vec{x}'') + \vec{\theta}^T(A\vec{x}-\vec{b}) \tag{3.3.5}$$

Where $\vec{x} \geq \vec{0}$ is the primal variable and $\vec{\theta}$ is the dual variable. The dual function W is:

$$W(\vec{x}'',\vec{\theta}) = \min_{\vec{x}\geq\vec{0}} L(\vec{x},\vec{x}'',\vec{\theta}) \tag{3.3.6}$$

Differentiation of the Lagrangian function (3.3.5) gives:

$$\nabla_x L = \vec{c} + A^T\vec{\theta} + 2\varepsilon(\vec{x}-\vec{x}'') \tag{3.3.7}$$

$$\nabla_{x''} L = -2\varepsilon(\vec{x}-\vec{x}'') \tag{3.3.8}$$

The dual problem of (3.3.4) is:

$$U(\vec{x}'') = \max_{\vec{\theta}} W(\vec{x}'',\vec{\theta}) \tag{3.3.9}$$

And the dual function gradient is:

$$\nabla_{\vec{\theta}} W \quad = \quad A\vec{x} - \vec{b} \tag{3.3.10}$$

3.3.1.2 Dual Gradient Algorithm

Using the dual problem (3.3.9) and the dual function (3.3.6) we rewrite the routing problem (3.3.4) into form:

$$\min_{\vec{x}''} \max_{\vec{\theta}} \min_{\vec{x} \geq \vec{0}} \quad L(\vec{x}, \vec{x}'', \vec{\theta}) \tag{3.3.11}$$

In contrast to Section 3.2.1.2, we define a gradient algorithm which consists of 3 nested loops (one loop for each variable). It is created from Equation (3.3.11). The internal loop solves the subproblem (3.3.6) using the gradient of the Lagrangian function (3.3.7). The middle loop solves the dual problem (3.3.9) using its gradient (3.3.10). The outer loop minimizes over the proximal-point variable $\vec{x}''$ using the gradient (3.3.8).

$$\begin{array}{l}
\text{LOOP 1} \\
\quad \text{LOOP 2} \\
\quad\quad \text{LOOP 3} \\
\quad\quad\quad \vec{x} = \left[\; \vec{x} - \alpha \nabla_x L \; \right]^+ \\
\quad\quad \text{END 3} \\
\quad\quad \vec{\theta} = \vec{\theta} + \alpha \nabla_{\vec{\theta}} W \\
\quad \text{END 2} \\
\quad \vec{x}'' = \vec{x}'' - \alpha \nabla_{x''} L \\
\text{END 1}
\end{array} \tag{3.3.12}$$

The $\alpha > 0$ is a constant step size of the algorithm.

We join all the loops of the gradient algorithm into one loop only, where we update all the variables $\vec{x}'', \vec{\theta}, \vec{x}$ simultaneously. Using Equations (3.3.7), (3.3.8) and (3.3.10) we derive one iteration of the algorithm:

$$\begin{array}{rcl}
\vec{x}_{k+1} & = & \left[\vec{x}_k - \alpha \big(\vec{c} + A^T \vec{\theta}_k + 2\varepsilon (\vec{x}_k - \vec{x}''_k) \big) \; \right]^+ \\
\vec{x}''_{k+1} & = & \vec{x}''_k + \alpha 2\varepsilon (\vec{x}_k - \vec{x}''_k) \\
\vec{\theta}_{k+1} & = & \vec{\theta}_k + \alpha (A\vec{x}_k - \vec{b})
\end{array} \tag{3.3.13}$$

k denotes the iteration number and symbol $[..]^+$ denotes a positive or zero value in each component of the vector $[..]^+ = max(\vec{0}, ..)$.

The correctness of such algorithm is not seen directly form its derivation and has to be proven. The proof of the algorithm convergence is not a trivial problem and its simplified version is presented in Section 3.3.3. A necessary condition for the algorithm convergence is $\alpha \leq 1/2\varepsilon$. Moreover, we have performed several simulation experiments to test the algorithm convergence in Section 3.3.2.

The variables $\vec{x}_0$, $\vec{x}''_0$ and $\vec{\theta}_0$ are set to arbitrary initial values. As in Chapter 3.2, the closer the values are to the final solution, the faster the algorithm converges. We do not investigate this problem further in this section. During the experiments in Section 3.3.2 we initiate the variables to zero.

3.3.1.3 Distributed Algorithm

The system of equations (3.3.13) is a description of one iteration of the distributable routing algorithm. However, to define the distributed algorithm for each node, we have to rewrite Equations (3.3.13) using (3.3.2), (3.3.3) and define variables $\vec{\theta}_k$ and $\vec{x}''_k$:

$$\vec{\theta}_k = \begin{bmatrix} \vec{\theta}_k^{(1)} \\ \vdots \\ \vec{\theta}_k^{(M)} \\ \lambda_k \end{bmatrix} \quad \vec{x}''_k = \begin{bmatrix} \vec{x}_k''^{(1)} \\ \vdots \\ \vec{x}_k''^{(M)} \\ \vec{z}''_k \end{bmatrix} \tag{3.3.14}$$

The presented distributed algorithm is running on each node in the network and it is synchronized by the communication between the nodes. The algorithm for node n is presented in Table 3.3.1.

We use $x_{k,i}^{(m)}$, $x_{k,i}''^{(m)}$, $\theta_{k,i}^{(m)}$, c_i etc. to denote the i-th component of the corresponding vector.

Due to the structure of the matrix A and vectors $\vec{x}_k$, $\vec{x}''_k$, $\vec{b}$ and $\vec{\theta}_k$ we rewrite the expressions (3.3.13) in order to compute the flow of the communication demand m in the link l into form of Equation (3.3.15). Where the expression l^- denotes index of the start node of the link l and l^+ denotes index of the end node of the link l.

1. Initialize the variables:

$$\begin{aligned}
x_{0,l}^{(m)} &= x_{start,l}^{(m)} && \forall m \in \mathcal{M} \quad \forall l \in \mathcal{O}(n) \\
x_{0,l}^{\prime\prime(m)} &= x_{start,l}^{\prime\prime(m)} && \forall m \in \mathcal{M} \quad \forall l \in \mathcal{O}(n) \\
\lambda_{0,l} &= \lambda_{start,l} && \forall l \in \mathcal{O}(n) \\
\theta_{0,n}^{(m)} &= \theta_{start,n}^{(m)} && \forall m \in \mathcal{M} \\
z_{0,l} &= z_{start,l} && \forall l \in \mathcal{O}(n) \\
z_{0,l}^{\prime\prime} &= z_{start,l}^{\prime\prime} && \forall l \in \mathcal{O}(n) \\
k &= 0
\end{aligned}$$

2. Send/receive the variables to/from the neighbors.

$$\begin{aligned}
\textbf{Send:} \quad & x_{k,l}^{(m)} && \forall m \in \mathcal{M}, \forall l \in \mathcal{O}(n) \\
& \theta_{k,n}^{(m)} && \forall m \in \mathcal{M} \\
\textbf{Receive:} \quad & x_{k,l}^{(m)} && \forall m \in \mathcal{M}, \forall l \in \mathcal{I}(n) \\
& \theta_{l^-}^{(m)} && \forall m \in \mathcal{M}, \forall l \in \mathcal{I}(n)
\end{aligned}$$

3. Evaluate equations for $k+1$ and for $\forall l \in \mathcal{O}(n)$ and $\forall m \in \mathcal{M}$:

$$\begin{aligned}
x_{k+1,l}^{(m)} &= \Big[x_{k,l}^{(m)} - \alpha\big(c_l' + \theta_{k,l^+}^{(m)} - \theta_{k,n}^{(m)} + \lambda_{k,l} + 2\varepsilon(x_{k,l}^{(m)} - x_{k,l}^{\prime\prime(m)})\big) \Big]^+ \\
x_{k+1,l}^{\prime\prime(m)} &= x_{k,l}^{\prime\prime(m)} + 2\alpha\varepsilon(x_{k,l}^{(m)} - x_{k,l}^{\prime\prime(m)}) \\
z_{k+1,l} &= \Big[z_{k,l} - \alpha\big(\lambda_{k,l} + 2\varepsilon(z_{k,l} - z_{k,l}^{\prime\prime})\big) \Big]^+ \\
z_{k+1,l}^{\prime\prime} &= z_{k,l}^{\prime\prime} - \alpha\big(2\varepsilon(z_{k,l} - z_{k,l}^{\prime\prime})\big) \\
\theta_{k+1,n}^{(m)} &= \theta_{k,n}^{(m)} + \alpha\Big(\sum_{i \in \mathcal{I}(n)} x_{k,i}^{(m)} - \sum_{i \in \mathcal{O}(n)} x_{k,i}^{(m)} - \bar{s}_{out,n}^{(m)} + \bar{s}_{in,n}^{(m)} \Big) \\
\lambda_{k+1,l} &= \lambda_{k,l} + \alpha\Big(\sum_{m \in \mathcal{M}} x_{k,l}^{(m)} + z_{k,l} - \mu_l \Big)
\end{aligned} \tag{3.3.15}$$

4. $k = k+1$, go to step 2 and start a new iteration loop.

Table 3.3.1: OLDRAi: Distributed Routing Algorithm executed in node n

Each node n is responsible for computation of the flow volume of the links leaving the node n and for computation of the corresponding dual variables. Therefore, node n computes $x_{k+1,l}^{(m)}$ and $x_{k+1,l}^{\prime\prime(m)}$ for all $l \in \mathcal{O}(n)$ and all $m \in \mathcal{M}$, $z_{k+1,l}$ and $z_{k+1,l}^{\prime\prime}$ and $\lambda_{k+1,l}$ for all $l \in \mathcal{O}(n)$ and $\theta_{k+1,n}^{(m)}$ for all $m \in \mathcal{M}$.

The algorithm for node n is presented in Table 3.3.1. In step 1, the algorithm initializes the variables. In steps 2 the node communicates the

variables to the neighbor nodes. In step 3 the node computes new values of the k-th iteration.

In (3.3.15), node n computes $x_{k+1,l}^{(m)}$ for all links leaving node n. It is a function of the local variables $x_{k,l}^{(m)}$, $x_{k,l}^{\prime\prime(m)}$, $\lambda_{k,l}$, $\theta_{k,n}^{(m)}$ and the variables $\theta_{k,l^+}^{(m)}$ of the neighbor nodes. Similarly, the computation of the others variables is a function of the local variables and the variables of the neighbor nodes that are within one hop communication distance.

3.3.2 Experiments

To demonstrate the behavior of OLDRAi and to experimentally verify its convergence, we present several experiments experiments in Matlab. The experiments are based on the same experimental model, as the previous experiments, with some small adjustments for the OLDRAi algorithm. We have focused on problem, where for each communication demand one node is supposed to send data flow to one sink node (i.e. multi-commodity mono-source, mono-sink problem). $s_{in,n_1}^{(m)} = 1$ for the source node n_1 and $s_{out,n_2}^{(m)} = 1$ for the sink node n_2 of the communication demand m.

The random networks for the experiments have been constructed as follows: We consider a square field of size $[size \times size]$. The $size$ is changing during the experiments. One node is randomly placed into each $[1 \times 1]$ sub-square and the communication distance is set to 1.5 (i.e. node n_1 can communicate with node n_2, if and only if their Euclidean distance is less than 1.5). Such a network is close to the "unit-disk network" [Rese 06]. The communication costs $\vec{c}$ per transmitted data flow unit have been set as the power of the distance between the nodes. The link capacities have been set to one $\mu_l = 1$. The constants of the algorithm have been set as: $\alpha = 0.03$, $\epsilon = 0.3$. The initial values $x_{start,l}^{(m)}$, $x_{start,l}^{\prime\prime(m)}$, $\theta_{start,n}^{(m)}$ have been set to 0 and $z_{start,l} = \mu_l$ and $z_{start,l}^{\prime\prime} = \mu_l$ for all experiments except 3.3.2.2. Only feasible problems are used.

During the experiments we evaluate the number of iterations k needed to achieve the optimal solution as a number of iterations needed to achieve less than 0.01% deviation of the objective function from the optimal value, less than 0.01% communication capacity violation and less than 0.01% flow conservation violation during last 100 iterations. (the optimal value was computed separately by a centralized algorithm for evaluation purposes only)

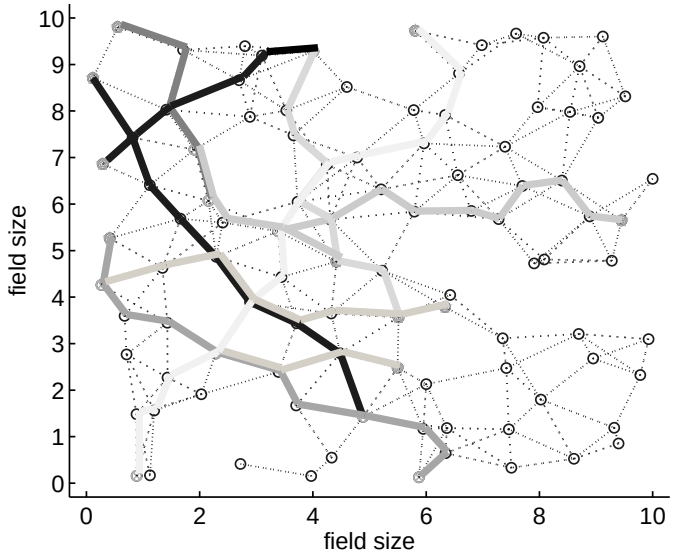


Fig. 3.3.1: OLDRAi: Optimal data flow routing (multi-commodity, mono-source, mono-sink problem)

3.3.2.1 Example

To present the final routing and the progress of the Lagrangian function during the computation of OLDRAi, we have performed an experiment based on the network described above. The field *size* has been set to 10 (i.e. 100 nodes in the network) and the number of communication demands has been set to 10.

The optimal data flow routing is shown in Figure 3.3.1. The progress of the Lagrangian function (3.3.5) and its optimal value are presented in Figure 3.3.2.

On the progress of the Lagrangian function, the algorithm convergence can be observed as the difference from its optimal value. Unfortunately, the progress of the Lagrangian function is not generally monotonic, which makes the proof of the algorithm convergence more difficult.

The difference between TLDRA and OLDRAi algorithms is visible from progress of the Lagrangian functions in Figures 3.3.2 and 3.2.3.

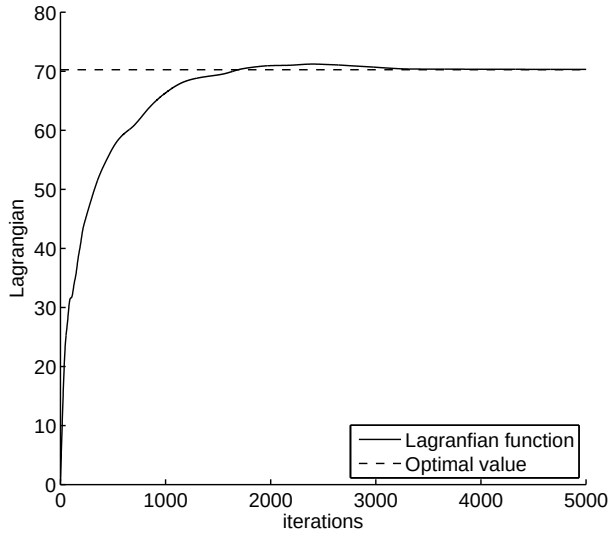


Fig. 3.3.2: OLDRAi: Progress of the Lagrangian function (3.3.5)

3.3.2.2 Algorithm Convergence

To experimentally verify the algorithm convergence, we have preformed a set of experiments with random starting points on random networks.The field *size* has been set to 10. There are 10 communication demands in the network. The initial values have been set randomly from intervals: $x^{(m)}_{start,l} \in \langle 0, 2\rangle$, $x''^{(m)}_{start,l} \in \langle 0, 2\rangle$, $z_{start,l} \in \langle 0, 2\rangle$, $z''_{start,l} \in \langle 0, 2\rangle$, $\theta^{(m)}_{start,n} \in \langle -20, 20\rangle$ and $\lambda^{(m)}_{start,l} \in \langle -20, 20\rangle$. The intervals have been chosen as double value of maximum/minimum of typical optimal values.

The algorithm has been run 2800 times on random networks and the results are presented in Figure 3.3.3. There is number of iterations placed on the horizontal axis and number of experiments which has been finished before the number of iterations on the vertical axis.

This experiment provides an important practical verification of the theoretical proof of the algorithm convergence. It can be seen, that approximately 95% of the experiments have been finished in 20000 iterations. It is obvious, that the number of the iterations is too big for the practical implementation

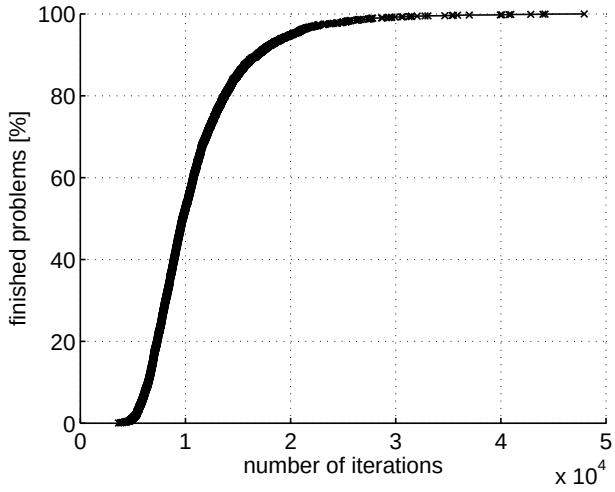


Fig. 3.3.3: OLDRAi: Algorithm convergence with random starting points.

of the algorithm. However, the main aim of this work was to introduce a new approach for distributed routing algorithms.

3.3.2.3 Number of Iterations

To demonstrate the statistical behavior of the algorithm, we have performed two tests. In the first one we have gradually increased the field *size* from 3 to 13 (i.e. from 9 to 169 nodes) for 10 communication demands. In the second one we have gradually increased the number of communication demands for field *size* 10. The computation has been repeated, on random networks, 300 times for each field *size* and each number of demands.

The results have been evaluated as a maximum, average and minimum number of iterations needed to achieve the optimal value and it is presented in Figure 3.3.4 for variable field *size* and in Figure 3.3.5 for variable number of demands.

The important outcome of this experiment is the observation, that the number of the iterations is approximately linear. It follows, that the algorithm is well applicable to a big networks with many communication demands.

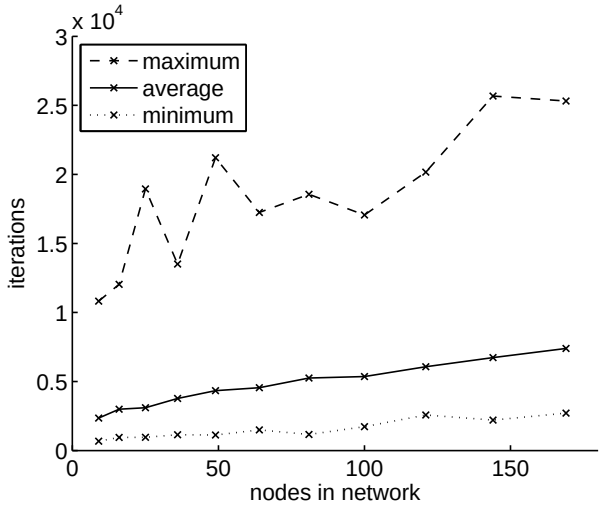


Fig. 3.3.4: OLDRAi: Number of iterations in relation to the number of nodes.

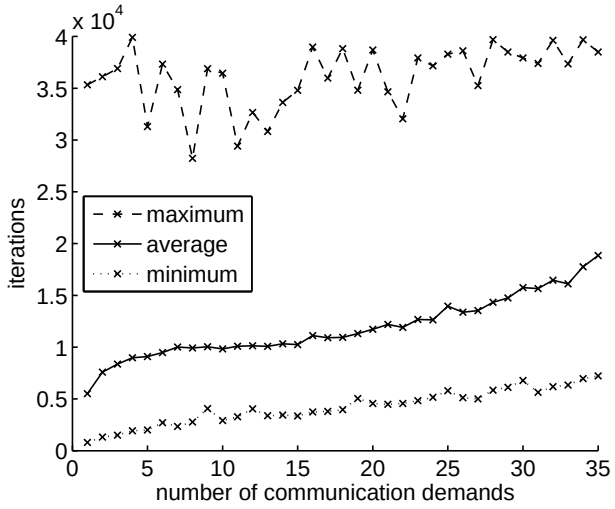


Fig. 3.3.5: OLDRAi: Number of iterations in relation to the number of communication demands.

3.3.3 Proof of the Algorithm Convergence

To prove the convergence of the algorithm presented in Table 3.3.1 for $\alpha \to 0$ and $\alpha > 0$ we proceed as follows. We formally rewrite Equations (3.3.13) into more suitable form. Then we define a merit function P_k such that $P_k = 0$ for optimal solution and show that P_k is non-increasing. Next we show for feasible problems that, if P_k is non-decreasing for all $k \geq k_0$, the solution in k_0-th iteration is the optimal solution of the problem. We use a marking $\pi_{k,i}$, $\nabla_x \tilde{L}_{k,i}$, $\left[A^T(A\vec{x}_k - \vec{b})\right]_i$ to denote i-th component of vectors. Let us remind that $\varepsilon > 0$, $\vec{x} \geq \vec{0}$, $\vec{c} \geq \vec{0}$ and that the Slater's conditions holds for problem (3.3.4). First we define variable $\vec{\pi}_k$:

$$\vec{\pi}_k = \frac{\partial L}{\partial \vec{x}} - \frac{\vec{x}_k}{\alpha} = \vec{c} + A^T\vec{\theta}_k + 2\varepsilon(\vec{x}_k - \vec{x}''_k) - \frac{\vec{x}_k}{\alpha} \tag{3.3.16}$$

We rewrite Equations (3.3.13) into a more suitable form as:

$$\begin{aligned} \vec{x}_{k+1} &= \vec{x}_k - \alpha\nabla_x \tilde{L}_k \\ \vec{x}''_{k+1} &= \vec{x}''_k - \alpha\nabla_{x''} \tilde{L}_k \\ \vec{\theta}_{k+1} &= \vec{\theta}_k + \alpha(A\vec{x}_k - \vec{b}) \end{aligned} \tag{3.3.17}$$

$$\nabla_x \tilde{L}_{k,i} = \begin{cases} c_i + [A^T\vec{\theta}_k]_i + 2\varepsilon(x_{k,i} - x''_{k,i}) & \text{for: } \pi_{k,i} \leq 0 \\ \frac{x_{k,i}}{\alpha} & \text{for: } \pi_{k,i} > 0 \end{cases} \tag{3.3.18}$$

$$\nabla_{x''} \tilde{L}_{k,i} = -2\varepsilon(x_{k,i} - x''_{k,i}) \tag{3.3.19}$$

It can be easily verify, that Equations (3.3.17) are identical with Equations (3.3.13). We can define a column change vector $\vec{d}_k$ as $\vec{d}_k = \alpha[-\nabla_x \tilde{L}_k^T, -\nabla_{x''} \tilde{L}_k^T, (A\vec{x}_k - \vec{b})^T]^T$. For future use we express the second differentiation of the $\tilde{L}_k$:

$$\nabla^2_{xx} \tilde{L}_{k,i} = \begin{cases} 2\varepsilon & \text{for: } \pi_{k,i} \leq 0 \\ \frac{1}{\alpha} & \text{for: } \pi_{k,i} > 0 \end{cases} \qquad \nabla^2_{xx''} \tilde{L}_{k,i} = \begin{cases} -2\varepsilon & \text{for: } \pi_{k,i} \leq 0 \\ 0 & \text{for: } \pi_{k,i} > 0 \end{cases} \tag{3.3.20}$$

$$\nabla^2_{x''x''} \tilde{L}_{k,i} = +2\varepsilon \qquad \nabla^2_{x''x} \tilde{L}_{k,i} = -2\varepsilon \tag{3.3.21}$$

We define the merit function P_k as:

$$P_k = 0.5|\nabla_x \tilde{L}_k|^2 + 0.5|\nabla_{x''} \tilde{L}_k|^2 + 0.5|A\vec{x}_k - \vec{b}|^2 \tag{3.3.22}$$

According to Karush-Kuhn-Tucker conditions (see e.g.[Boyd 04], [Bert 99]) if $P_k = 0$ the solution in the k-th iteration of the algorithm is the optimal solution of problem (3.3.4). Gradient of the merit function P_k is:

$$\nabla P_k = \begin{bmatrix} \nabla^2_{xx} \tilde{L}_k \nabla_x \tilde{L}_k + \nabla^2_{x''x} \tilde{L}_k \nabla_{x''} \tilde{L}_k + A^T(A\vec{x}_k - \vec{b}) \\ \nabla^2_{xx''} \tilde{L}_k \nabla_x \tilde{L}_k + \nabla^2_{x''x''} \tilde{L}_k \nabla_{x''} \tilde{L}_k \\ A\nabla_x \tilde{L}_k \end{bmatrix} \tag{3.3.23}$$

For $\alpha \to 0$ we can express one iteration step of Equation (3.3.17) and modify using Equations (3.3.20) and (3.3.21):

$$\begin{aligned} \vec{d}_k^T \nabla P_k &= \alpha\big(- \nabla_x \tilde{L}_k^T \nabla^2_{xx} \tilde{L}_k \nabla_x \tilde{L}_k - \nabla_x \tilde{L}_k^T \nabla^2_{x''x} \tilde{L}_k \nabla_{x''} \tilde{L}_k \\ &\quad -\nabla_x \tilde{L}_k^T A^T(A\vec{x}_k - \vec{b}) - \nabla_{x''} \tilde{L}_k^T \nabla^2_{xx''} \tilde{L}_k \nabla_x \tilde{L}_k \\ &\quad -\nabla_{x''} \tilde{L}_k^T \nabla^2_{x''x''} \tilde{L}_k \nabla_{x''} \tilde{L}_k + (A\vec{x}_k - \vec{b})^T A \nabla_x \tilde{L}_k\big) = \\ &= -2\alpha\varepsilon\big(\nabla_x \tilde{L}_k^T I'_k \nabla_x \tilde{L}_k - \nabla_x \tilde{L}_k^T I'_k \nabla_{x''} \tilde{L}_k - \nabla_{x''} \tilde{L}_k^T I'_k \nabla_x \tilde{L}_k \\ &\quad +\nabla_{x''} \tilde{L}_k^T I'_k \nabla_{x''} \tilde{L}_k\big) \\ &\quad -\alpha\big(\tfrac{1}{\alpha} \nabla_x \tilde{L}_k^T I''_k \nabla_x \tilde{L}_k - 2\varepsilon \nabla_x \tilde{L}_k^T I''_k \nabla_{x''} \tilde{L}_k + \nabla_{x''} \tilde{L}_k^T I''_k \nabla_{x''} \tilde{L}_k\big) = \\ &= -2\alpha\varepsilon\big(\nabla_x \tilde{L}_k - \nabla_{x''} \tilde{L}_k\big)^T I'_k \big(\nabla_x \tilde{L}_k - \nabla_{x''} \tilde{L}_k\big) \\ &\quad -2\alpha\varepsilon\big(\nabla_x \tilde{L}_k^T I''_k \nabla_x \tilde{L}_k - \nabla_x \tilde{L}_k^T I''_k \nabla_{x''} \tilde{L}_k + \nabla_{x''} \tilde{L}_k^T I''_k \nabla_{x''} \tilde{L}_k\big) \\ &\quad -\big(1 - 2\alpha\varepsilon\big) \nabla_x \tilde{L}_k^T I''_k \nabla_x \tilde{L}_k \end{aligned} \tag{3.3.24}$$

$$I'_{k,i,j} = \begin{cases} 1, & i = j \text{ and } \pi_{k,i} \le 0 \\ 0, & \text{otherwise} \end{cases} \qquad I''_{k,i,j} = \begin{cases} 1, & i = j \text{ and } \pi_{k,i} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.3.25}$$

From Equation (3.3.24) results that $\vec{d}_k^T \nabla P_k \le 0$ and then the merit function P_k is non-increasing during the algorithm for some $\alpha > 0$ and $\alpha \le 1/(2\varepsilon)$. For non-decreasing merit function P_k (i.e. $\vec{d}_k^T \nabla P_k = 0$ for $\forall_{k \ge k_0}$) we can write from Equation (3.3.24):

$$\begin{aligned} \nabla_x \tilde{L}_{k,i} - \nabla_{x''} \tilde{L}_{k,i} = 0 \quad \text{for} \quad \pi_{k,i} \le 0 \\ \nabla_x \tilde{L}_{k,i} = \nabla_{x''} \tilde{L}_{k,i} = 0 \quad \text{for} \quad \pi_{k,i} > 0 \end{aligned} \tag{3.3.26}$$

From Equations (3.3.26), (3.3.19) and (3.3.17) holds:

$$\nabla_x \tilde{L}_k = \nabla_{x''} \tilde{L}_k = \nabla_{x''} \tilde{L}_{k+1} = \nabla_x \tilde{L}_{k+1} \tag{3.3.27}$$

It means, that if $\exists_{k \geq k_0} : \pi_{k,i} > 0$ then $\nabla_x \tilde{L}_{k,i} = \nabla_{x''} \tilde{L}_{k,i} = 0$ for $\forall_{k \geq k_0}$. From Equations (3.3.27), (3.3.18) and (3.3.17) we can write for all components i where holds $\pi_{k,i} \leq 0$:

$$\left[A^T(A\vec{x}_k - \vec{b})\right]_i = 0 \tag{3.3.28}$$

From Equations (3.3.16) and (3.3.17)

$$\pi_{k+1,i} = \pi_{k,i} + \alpha \nabla_x \tilde{L}_{k,i} + \left[A^T(A\vec{x}_k - \vec{b})\right]_i \tag{3.3.29}$$

and according to Equations (3.3.28) and (3.3.27) it folows:

$$\nabla_x \tilde{L}_k \leq \vec{0} \qquad \forall_{k \geq k_0} \tag{3.3.30}$$

We can write:

$$\begin{aligned} \nabla_x \tilde{L}_k A^T \left(A\vec{x}_{k+1} - \vec{b}\right) &= \nabla_x \tilde{L}_k A^T \left(A\vec{x}_k - \vec{b} - \alpha A \nabla_x \tilde{L}_k\right) = \\ &= \nabla_x \tilde{L}_k A^T \left(A\vec{x}_k - \vec{b}\right) - \alpha \nabla_x \tilde{L}_k^T A^T A \nabla_x \tilde{L}_k = 0 \end{aligned} \tag{3.3.31}$$

Using Equations (3.3.26) and (3.3.28) we get: $\nabla_x \tilde{L}_k A^T \left(A\vec{x}_k - \vec{b}\right) = 0$ and then result of Equation (3.3.31) is:

$$\nabla_x \tilde{L}_k^T A^T = 0 \tag{3.3.32}$$

Using Equations (3.3.18), (3.3.26) (3.3.19) (3.3.32) we can proceed:

$$0 \leq \nabla_x \tilde{L}_k^T \nabla_x \tilde{L}_k = \nabla_x \tilde{L}_k^T \left(\vec{c} + A^T \vec{\theta}_k + 2\varepsilon(\vec{x}_k - \vec{x}''_k)\right) = \nabla_x \tilde{L}_k^T \left(\vec{c} - \nabla_x \tilde{L}_k\right) \leq 0 \tag{3.3.33}$$

From Equation (3.3.33) follows that for non-decreasing merit function P_k (i.e. $\vec{d}_k^T \nabla P_k = 0$ for $\forall_{k \geq k_0}$) holds:

$$\nabla_x \tilde{L}_k = \nabla_{x''} \tilde{L}_k = 0 \tag{3.3.34}$$

Consider problem (3.3.4) with both optimization variables x, x''. Under the condition (3.3.34) we can write a dual function of this problem as:

$$\begin{aligned} g(\vec{\theta}_k) &= \min_{\vec{x} \geq \vec{0}, \vec{x}''} L(\vec{x}, \vec{x}'', \vec{\theta}_k) = \\ &= \sum_{m \in \mathcal{M}} \left(\vec{c}^T \vec{x}_{k_0} + \varepsilon(\vec{x}_{k_0} - \vec{x}''_{k_0})^T (\vec{x}_{k_0} - \vec{x}''_{k_0}) + \vec{\theta}_k^{(m)^T} (A\vec{x}_{k_0} - \vec{b})\right) \end{aligned} \tag{3.3.35}$$

Then for $k+1$ iteration holds:

$$g(\vec{\theta}_{k+1}) = g(\vec{\theta}_k) + \alpha \sum_{m \in \mathcal{M}} (A\vec{x}_{k_0} - \vec{b})^T (A\vec{x}_{k_0} - \vec{b}) \tag{3.3.36}$$

From Equation (3.3.36) and (3.3.34) follows that for $(A\vec{x}_{k_0} - \vec{b}) \neq 0$ holds: if $k \to \infty$ then $g(\vec{\theta}_k) \to \infty$. According to the duality gap theorem (see e.g. [Bert 99]) holds $\max_{\vec{\theta}} g(\vec{\theta}) \leq \min_{\vec{x},\vec{x}'' \in \mathcal{S}} f(\vec{x}, \vec{x}'')$, where $f(\vec{x}, \vec{x}'')$ represents the objective function and $\mathcal{S}$ a set of feasible solutions. It follows, if $(A\vec{x}_{k_0} - \vec{b}) \neq 0$ and the merit function (3.3.23) is not-decreasing then the original problem is not feasible. It follows for feasible problems:

$$(A\vec{x}_{k_0} - \vec{b}) = 0 \quad \forall_{k \geq k_0} \tag{3.3.37}$$

We have presented a merit function P_k which is not-increasing during the algorithm for some small α and which is equal to zero $P_k = 0$ for optimal feasible solution. Next, we have shown for feasible problems that if the merit function P_k is not-decreasing then the solution $\vec{x} = \vec{x}''$ is the optimal solution of the original problem.

3.4 One Loop Distributed Routing Algorithm with Optimal flow update

This section introduces a different one loop distributed routing algorithm, called *One Loop Distributed Routing Algorithm with Optimal flow update* (OLDRAo). It is based on the algorithm TLDRA from Section 3.2 and it is adjusted by a similar approach as the OLDRAi in Section 3.3. The algorithm differs from the OLDRAi in the update of the flow variable x, which is directly set to the optimal value in each iteration of the algorithm. We focus on the node capacity case in this section.

3.4.1 Mathematical Derivation of the One Loop Distributed Routing Algorithm with Optimal flow update

We use the same equality form for the problem description (3.3.1) as in previous Section 3.3, which has been adjusted by the proximal-point method (3.1.2):

$$\begin{aligned} \min_{\vec{x}''} \min_{\vec{x}} \quad & \vec{c}^T\vec{x} + \varepsilon(\vec{x} - \vec{x}'')^T(\vec{x} - \vec{x}'') \\ \textbf{subject to:} & \\ A\vec{x} &= \vec{b} \\ \vec{x} &\geq \vec{0} \end{aligned} \tag{3.4.1}$$

where $\varepsilon > 0$.

Only matrix A differs from the (3.3.2). It is changed for the node capacity problem:

$$A = \begin{pmatrix} A' & 0 & 0 & 0 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & A' & 0 \\ D & D & D & I \end{pmatrix} \tag{3.4.2}$$

where D is defined according to (3.2.16) and $A' = (A^+ - A^-)$.

Again, for optimal solution of problem (3.4.1) holds $\vec{x} = \vec{x}''$. The routing problem has been separated into two nested subproblems. The internal subproblem is minimization over the variable $\vec{x}$ and it is strictly convex. The

outer subproblem minimizes the internal one by the proximal-point variable $\vec{x}''$.

3.4.1.1 Dual Problem

To solve the internal subproblem of (3.4.1) (minimization over the variable $\vec{x}$) we present its dual problem, which allows to derive the distributable gradient algorithm. The Slater's conditions (see e.g. [Boyd 04]) hold for problem (3.4.1).

The dual problem $U(\vec{x}'')$, dual function $W(\vec{x}'', \vec{\theta})$ and Lagrangian function $L(\vec{x}, \vec{x}'', \vec{\theta})$ are the same as in Section 3.3.1.1:

$$U(\vec{x}'') = \max_{\vec{\theta}} \; W(\vec{x}'', \vec{\theta}) = \max_{\vec{\theta}} \min_{\vec{x} \geq \vec{0}} \; L(\vec{x}, \vec{x}'', \vec{\theta}) \tag{3.4.3}$$

According to gradient of Lagrangian function (3.3.7), the minimizer of $L(\vec{x}, \vec{x}'', \vec{\theta})$ over variable x is:

$$\vec{x}_{min} = \left[\vec{x}'' - \frac{1}{2\varepsilon} (\vec{c} + A^T \vec{\theta}) \right]^+ \tag{3.4.4}$$

where symbol $[..]^+$ denotes a positive or zero value in each component of vector $[..]^+ = max(\vec{0}, ..)$.

The gradients of the dual function $W(\vec{x}'', \vec{\theta})$ and the dual problem $U(\vec{x}'')$ are:

$$\nabla_{\theta} W(\vec{x}'', \vec{\theta}) = A\vec{x}_{min} - \vec{b} \tag{3.4.5}$$

$$\nabla_{x''} U(\vec{x}'') = \nabla_{x''} L(\vec{x}_{min}, \vec{x}'', \vec{\theta}_{max}) = -2\varepsilon(\vec{x}_{min} - \vec{x}'') \tag{3.4.6}$$

Variable $\vec{\theta}_{max}$ maximizes the dual function $W(\vec{x}'', \vec{\theta})$ for a given $\vec{x}''$.

3.4.1.2 Dual Gradient Algorithm

The routing problem (3.4.1) can be written as:

$$\min_{\vec{x}''} \; \max_{\vec{\theta}} \; \min_{\vec{x} \geq \vec{0}} \; L(\vec{x}, \vec{x}'', \vec{\theta}) \tag{3.4.7}$$

Similarly as in Section 3.2.1.2 we define a gradient algorithm with two nested loops. The internal loop solves the dual problem $U(\vec{x}'')$ using the gradient

(3.4.5). The outer loop minimizes problem (3.4.7) over the proximal-point variable $\vec{x}''$ using the gradient (3.4.6).

$$\begin{array}{l} \text{LOOP 1} \\ \quad \text{LOOP 2} \\ \quad\quad \vec{x} = \left[\vec{x}'' - \frac{1}{2\varepsilon}(\vec{c} + A^T\vec{\theta})\right]^+ \\ \quad\quad \vec{\theta} = \vec{\theta} + \alpha(A\vec{x} - \vec{b}) \\ \quad \text{END 2} \\ \quad \vec{x}'' = \vec{x}'' + 2\alpha\varepsilon(\vec{x} - \vec{x}'') \\ \text{END 1} \end{array} \tag{3.4.8}$$

$\alpha > 0$ is a constant step size of the algorithm.

To derive the distributed routing algorithm, we join both loops of the gradient algorithm into only one loop, where we update all the variables $\vec{x}'', \vec{\theta}, \vec{x}$ simultaneously. Using Equations (3.4.4), (3.4.5), (3.4.6) we derive one iteration of the algorithm as:

$$\begin{array}{lcl} \vec{x}_k & = & \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]^+ \\ \vec{\theta}_{k+1} & = & \vec{\theta}_k + \alpha(A\vec{x}_k - \vec{b}) \\ \vec{x}''_{k+1} & = & \vec{x}''_k + 2\alpha\varepsilon(\vec{x}_k - \vec{x}''_k) \end{array} \tag{3.4.9}$$

where variable k denotes the iteration number. The proof of the algorithm convergence is not a trivial problem and its simplified version is presented in Section 3.4.3. A necessary condition for the algorithm convergence assumed in the proof is $\alpha < 1/2\varepsilon$. We have performed several simulation experiments to test the algorithm convergence in Section 3.4.2.2.

There are two main differences between the OLDRAo and OLDRAi, which can be seen from Equations (3.4.9) and (3.3.13). The flow variable $\vec{x}$ is set directly to optimal value in OLDRAo while in OLDRAi it is changed incrementally. The second difference is in the update of the variables $\vec{\theta}$ and $\vec{x}''$. While the algorithm OLDRAi uses the variable $\vec{x}$ from previous iteration, the algorithm OLDRAo uses the variable $\vec{x}$ which has been computed in the actual iteration. The OLDRAo algorithm needs less iterations than OLDRAi to converge, according to our experiments (see Section 3.4.2.2 and 3.4.2.4).

As for the OLDRi algorithm, the variables $\vec{x}''_0$, $\vec{\theta}_0$ are set to arbitrary initial values for the OLDRAo. The closer the values are to the final solution,

the faster the algorithm converges. This property can be used in the case of minor changes of the network structure during its operation or in case of a precomputed routing e.g. based on Dijkstra's algorithm. For variable initialization see Section 3.4.1.4 and for experimental verification see Section 3.4.2.3.

3.4.1.3 Distributed Algorithm

To derive the distributed algorithm for one node, we proceed as in Section 3.3.1.3. We rewrite Equations (3.4.9) using definitions (3.3.2), (3.3.3) and (3.3.14).

The presented distributed algorithm is running on each node in the network and it is synchronized by the communication between the neighboring nodes. The algorithm for node n is presented in Table 3.4.1.

We use marking $x_{k,i}^{(m)}$, $x_{k,i}^{\prime\prime(m)}$, $\theta_{k,i}^{(m)}$, c_i etc. to denote the i-th component of the corresponding vector.

Due to the structure of matrix A and vectors $\vec{x}_k$, $\vec{x}_k^{\prime\prime}$, $\vec{b}$, $\vec{\theta}_k$ we rewrite the expressions (3.4.9) in order to compute the flow of the communication demand m in link l into form of Equations (3.4.10) and (3.4.11). The expression l^- denotes the index of the start node of link l and l^+ denotes the index of the end node of link l.

Each node n is responsible for the computation of the flow volume of the links leaving node n and for the computation of the corresponding dual variables. Therefore, node n computes $x_{k,l}^{(m)}$ and $x_{k+1,l}^{\prime\prime(m)}$ for all $l \in \mathcal{O}(n)$ and all $m \in \mathcal{M}$, $\theta_{k+1,n}^{(m)}$ for all $m \in \mathcal{M}$ and $z_{k,n}$, $z_{k+1,n}^{\prime\prime}$ and $\lambda_{k+1,n}$.

The algorithm for node n (Table 3.4.1) works as follows: In step 1, the algorithm initializes the variables. In steps 2 and 4 the node communicates the variables with the neighbor nodes. In steps 3 and 5 the node computes the new values of the k-th iteration.

In (3.4.10), node n computes $x_{k,l}^{(m)}$ for all links leaving node n. It is the function of the local variables $x_{k,l}^{\prime\prime(m)}$, $\lambda_{k,n}$, $\theta_{k,n}^{(m)}$ and the variables $\theta_{k,l^+}^{(m)}$ of the neighbor nodes. Similarly, the computation of the other variables in (3.4.11) is the function of the local variables and the variables of the neighbor nodes that are within one hop communication distance.

1. Initialize the variables:

$$\begin{aligned} x_{0,l}^{\prime\prime(m)} &= x_{start,l}^{\prime\prime(m)} && \forall m \in \mathcal{M} \quad \forall l \in \mathcal{O}(n) \\ \lambda_{0,n} &= \lambda_{start,n} \\ \theta_{0,n}^{(m)} &= \theta_{start,n}^{(m)} && \forall m \in \mathcal{M} \\ z_{0,n}^{\prime\prime} &= z_{start,n}^{\prime\prime} \\ k &= 0 \end{aligned}$$

2. Send/receive the variables to/from the neighbors.

$$\begin{aligned} \textbf{Send:} \quad & \theta_{k,n}^{(m)} \quad \forall m \in \mathcal{M} \\ \textbf{Receive:} \quad & \theta_{l^+}^{(m)} \quad \forall m \in \mathcal{M}, \forall l \in \mathcal{O}(n) \end{aligned}$$

3. Evaluate equations for $x_{k,l}^{(m)}$ and $z_{k,n}$ for $\forall l \in \mathcal{O}(n)$, and $\forall m \in \mathcal{M}$:

$$\begin{aligned} x_{k,l}^{(m)} &= \left[x_{k,l}^{\prime\prime(m)} - \tfrac{1}{2\varepsilon}\left(c_l^{\prime} + \theta_{k,l^+}^{(m)} - \theta_{k,n}^{(m)} + \lambda_{k,n}\right) \right]^+ \\ z_{k,n} &= \left[z_{k,n}^{\prime\prime} - \tfrac{1}{2\varepsilon}\lambda_{k,n} \right]^+ \end{aligned} \tag{3.4.10}$$

4. Send/receive the variables to/from the neighbors.

$$\begin{aligned} \textbf{Send:} \quad & x_{k,l}^{(m)} \quad \forall m \in \mathcal{M}, \forall l \in \mathcal{O}(n) \\ \textbf{Receive:} \quad & x_{k,l}^{(m)} \quad \forall m \in \mathcal{M}, \forall l \in \mathcal{I}(n) \end{aligned}$$

5. Evaluate equations for $k+1$ and for $\forall l \in \mathcal{O}(n)$ and $\forall m \in \mathcal{M}$:

$$\begin{aligned} \theta_{k+1,n}^{(m)} &= \theta_{k,n}^{(m)} + \alpha\Big(\sum_{i \in \mathcal{I}(n)} x_{k,i}^{(m)} - \sum_{i \in \mathcal{O}(n)} x_{k,i}^{(m)} - \bar{s}_{out,n}^{(m)} + \bar{s}_{in,n}^{(m)}\Big) \\ \lambda_{k+1,n} &= \lambda_{k,n} + \alpha\Big(\sum_{i \in \mathcal{O}(n)} \sum_{m \in \mathcal{M}} x_{k,i}^{(m)} + z_{k,n} - \mu_n\Big) \\ x_{k+1,l}^{\prime\prime(m)} &= x_{k,l}^{\prime\prime(m)} + 2\alpha\varepsilon\big(x_{k,l}^{(m)} - x_{k,l}^{\prime\prime(m)}\big) \\ z_{k+1,n}^{\prime\prime} &= z_{k,n}^{\prime\prime} + 2\alpha\varepsilon\big(z_{k,n} - z_{k,n}^{\prime\prime}\big) \end{aligned} \tag{3.4.11}$$

6. $k = k+1$, go to step 2 and start a new iteration loop.

Table 3.4.1: OLDRAo: Distributed Routing Algorithm executed in node n

3.4.1.4 Variables Initialization

As we have mentioned in Section 3.4.1.2 the number of iterations needed to reach the optimal solution depends on the initial variables $\vec{x}_0^{\prime\prime}$, $\vec{\theta}_0$. We can say that the closer the initial variables are to the final solution the less iterations

are needed and we can base an initial heuristic on this fact.

First, let us suppose some initial variable $\vec{\theta}_0$ which satisfies condition:

$$\left(\vec{c} + A^T \vec{\theta}_0\right) \geq \vec{0} \tag{3.4.12}$$

According to Equations (3.4.9), such initial variables do not cause any changes of the algorithm variables on its own. (i.e. there is no change of the variables if $\vec{x}''_0 = \vec{0}$ and $\vec{b} = \vec{0}$)

According to the complementary slackness condition (for details see e.g. [Bert 98]) for the optimal solution holds: if $\theta^{(m)}_{k,l^+} - \theta^{(m)}_{k,l^-} < c'_l$ then $x^{(m)}_{k,l} = 0$ and if $\theta^{(m)}_{k,l^+} - \theta^{(m)}_{k,l^-} = c'_l$ then $x^{(m)}_{k,l} \geq 0$. Moreover, if $\theta^{(m)}_{k,l^+} - \theta^{(m)}_{k,l^-} = c'_l$ link l is a part of the shortest path for demand $m \in \mathcal{M}$. This fact leads us directly to the Dijkstra's algorithm which can be used in distributed way. If for all sink nodes n_{out} we set $\theta^{(m)}_{0,n_{out}} = 0$ and for the other nodes n we set $\theta^{(m)}_{0,n} =$ *the shortest distance to the sink node*, we get an initial setting, which satisfies the condition (3.4.12). Moreover, this initial setting is much closer to the optimal solution than setting $\theta^{(m)}_{0,n} = 0$ for all $\forall n, m$. In Section 3.4.2.3 we show several experiments to present the heuristic behavior in comparison with the zero initial setting. Please notice, that due to the capacity constraints, the heuristic solution is usually not equal to the final optimal solution. According to our experiments it rapidly increases the algorithm convergence.

3.4.2 Experiments

This section aims to present the behavior of OLDRAo algorithm, to experimentally verify its convergence and to compare the OLDRAo and the OLDRAi algorithms. The OLDRAi algorithm has been adjusted for the node capacity constraints to be comparable with the OLDRAo algorithm. The experiments have been performed in Matlab. We have focused on one-to-one communication problem in this section (i.e. multi-commodity mono-source, mono-sink problem). Therefore, $s^{(m)}_{in,n_1} = 1$ for the source node n_1 and $s^{(m)}_{out,n_2} = 1$ for the sink node n_2 of communication demand m.

The networks are constructed as in Sections 3.2.2 and 3.3.2: One node is randomly placed into each $[1 \times 1]$ sub-squares of a $[size \times size]$ field. The $size$ is changing during the experiments. The communication distance is set to 1.5 (i.e. node n_1 can communicate with node n_2, if and only if

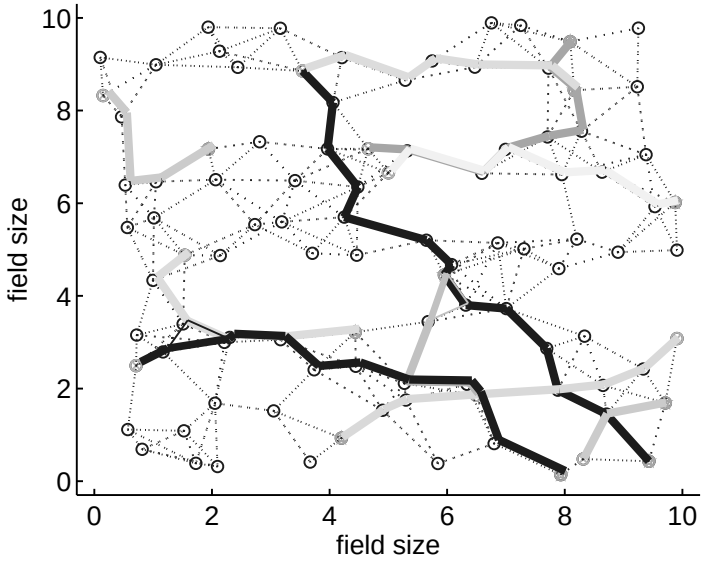


Fig. 3.4.1: OLDRAo: Optimal data flow routing (multi-commodity, mono-source, mono-sink problem)

their Euclidean distance is less than 1.5). The communication costs $\vec{c'}$ per transmitted data flow unit have been set as the square power of the distance between the nodes. In contrast to Sections 3.2.2 and 3.3.2, we use a node communication capacity in this experiments. The node capacities have been set as $\mu_l = 2$. The constants of the algorithm have been set as: $\alpha = 0.03$ and $\epsilon = 0.3$. The initial values $x''^{(m)}_{start,l}$, $\theta^{(m)}_{start,n}$ have been set to 0 and $z''_{start,n} = \mu_l$ for all experiments except 3.4.2.2 and 3.4.2.4.

The number of algorithm iterations k is evaluated in the experiments as a number of iterations needed to achieve less than 1% deviation of the objective function from the optimal value, less than 1% communication capacity violation and less than 1% flow conservation violation during last 500 iterations. (the optimal value was computed separately by a centralized algorithm for evaluation purposes only)

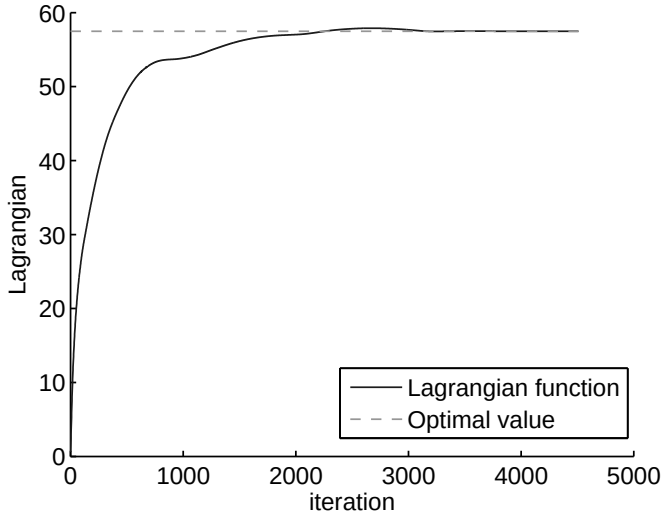


Fig. 3.4.2: OLDRAo: Progress of the Lagrangian function (3.3.5)

3.4.2.1 Example

To present the OLDRAo algorithm routing and the progress of the Lagrangian function, we have performed an experiment based on the network described above. The field *size* has been set to 10 (i.e. 100 nodes in the network) and the number of communication demands has been set to 10.

The optimal data flow routing is shown in Figure 3.4.1. The progress of the Lagrangian function (3.3.5) and its final optimal value are presented in Figure 3.4.2.

As for the OLDRAi algorithm, on the progress of the Lagrangian function, the algorithm convergence can be observed as the difference from its optimal value. Unfortunately, the progress of the Lagrangian function is not generally monotonic, which makes the proof of the algorithm convergence more difficult.

Several videos for presentation of the algorithm progress in time can be found on author web-pages [Trdl 11].

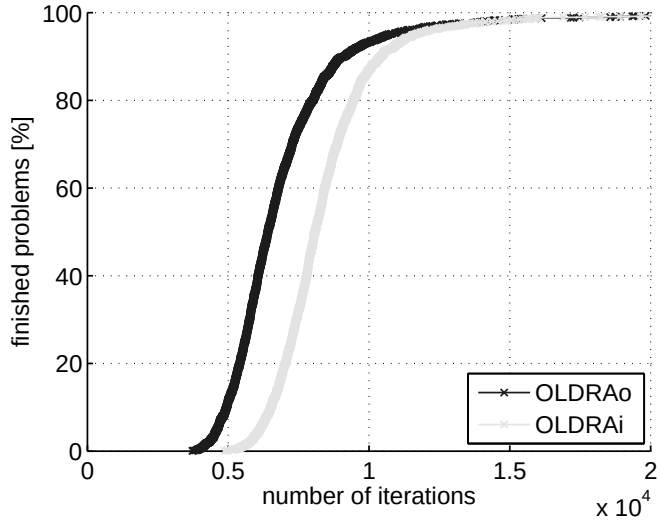


Fig. 3.4.3: OLDRAo: Algorithm convergence with random initial variables.

3.4.2.2 Algorithm Convergence

We have performed a set of experiments with random initial variables on random networks to evaluate the algorithm convergence and to compare the OLDRAi and the OLDRAo algorithms. The field *size* has been set to 10. There are 10 communication demands in the network. The initial values have been set randomly from intervals: $x''^{(m)}_{start,l} \in \langle 0, 2\rangle$, $\theta^{(m)}_{start,n} \in \langle -20, 20\rangle$, $\lambda^{(m)}_{start,n} \in \langle -20, 20\rangle$ and $z''_{start,l} =\in \langle 0, 2\rangle$. The intervals have been chosen as double the value of maximum/minimum of typical optimal values.

The algorithm has been run 2000 times on random networks for the algorithm OLDRAo and for the algorithm OLDRAi. The results are presented in Figure 3.4.3. There is the number of iterations placed on the horizontal axis and the number of experiments which has been finished before the number of iterations on the vertical axis.

This experiment provides an important practical verification of the theoretical proof of the algorithm convergence. 95% of the experiments have been finished in 10688 iterations for OLDRAo and in 11600 iterations for the OLDRAi. The average improvement over all experiments of the algorithm is

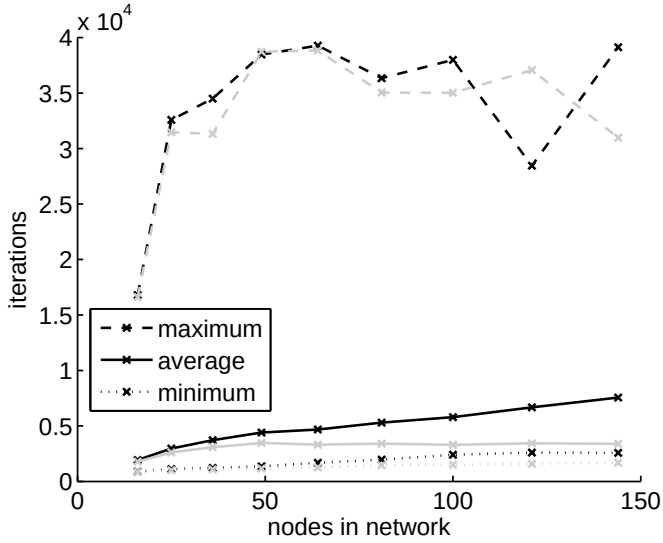


Fig. 3.4.4: OLDRAo: Number of iterations in relation to the number of nodes. In black, for zero initial variables. In gray, for initialization according to Section 3.4.1.4.

1513 iterations.

3.4.2.3 Number of Iterations

To demonstrate the statistical behavior of the algorithm, we have performed two tests. In the first one we have gradually increased the number of nodes from 16 to 144 (i.e. field *size* from 4 to 12) for 10 communication demands. In the second one we have gradually increased the number of communication demands for field *size* 10 from 1 to 19. The computation has been repeated, on random networks, 300 times for each number of nodes and number of demands.

The results have been evaluated as a maximum, average and minimum number of iterations needed to achieve the optimal value and it is presented in Figure 3.4.4 for the variable number of nodes and in Figure 3.4.5 for the variable number of demands. The data for the basic algorithm without the initial heuristic (the initial variables have been set to zero) is in black and the data for the algorithm with the initial heuristic from Section 3.4.1.4 is in

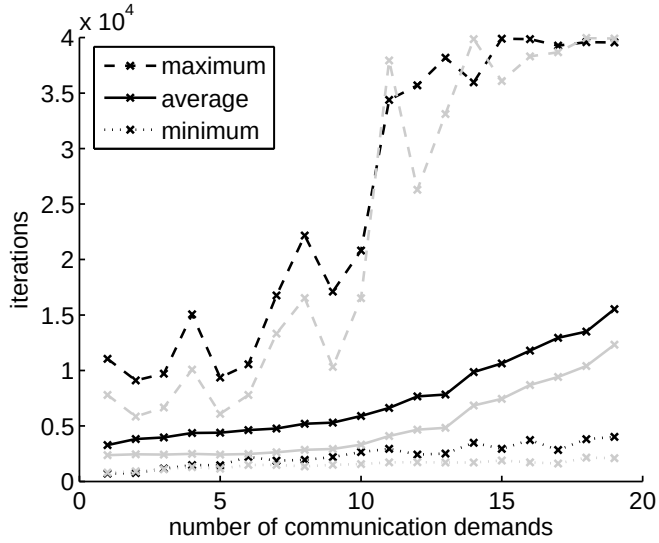


Fig. 3.4.5: OLDRAo: Number of iterations in relation to the number of communication demands. In black, for zero initial variables. In gray, for initialization according to Section 3.4.1.4.

gray.

The important outcome of this experiment is the observation, that the number of the iterations is approximately linear in relation to the number of the nodes. It follows, that the algorithm is easy applied to big networks with many nodes. The improvement gained by the initial heuristic can see in Figures 3.4.4, 3.4.5.

3.4.2.4 Network Change

The advantage of the one-loop algorithm presented in this work is that it can automatically adjust the routing in case of network structure changes. To evaluate the algorithm behavior in this case, we simulated a dying node as follows:

1. We generated a random network with communication demands and found the optimal solution

2. In the original problem from step 1, we removed one node and measured

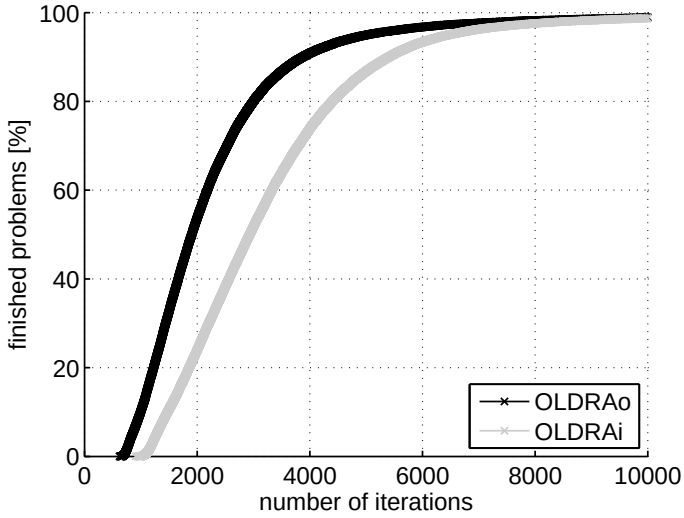


Fig. 3.4.6: OLDRAo: Algorithm convergence for network change simulation.

the number of iterations to find the new optimal solution

3. We repeated step 2 on the original problem for 30 different nodes with the largest data flows

We have performed this simulation for 2000 random networks (i.e. $2000 \times 30 = 60000$ experiments). The network *size* has been set to 10 and there have been 10 communication demands.

The results are presented in Figure 3.4.6. The number of iterations is placed on the horizontal axis and the number of experiments which have been finished before the number of iterations is placed on the vertical axis. There is a comparison of the algorithm OLDRAo (in black) and the algorithm OLDRAi (in gray).

The average difference in the number of iterations is 986. 95% of the experiments have been finished in 5000 iterations for the algorithm OLDRAo and in 6500 iterations for the algorithm OLDRAi. Moreover, only 29.0% of the dual variables $\theta_{k,n}^{(m)}$, and 5.0% of the primal variables $x_{k,l}^{(m)}$ have been changed during the experiments for OLDRAo. It can decrease the amount of transmitted data.

3.4.3 Proof of the Algorithm Convergence

We prove the convergence of the algorithm presented in Table 3.4.1 for $\alpha \to 0$ and for $0 < \alpha < 1/(2\varepsilon)$ as follows. First, we define a merit function P_k such that $P_k \geq 0$ and $P_k = 0$ for the optimal solution and we show that P_k is non-increasing during the algorithm computation. Next, we assume the merit function P_k to be non-decreasing for all $k \geq k_0$ and show, for feasible problems, that for some $k_1 \geq k_0$ we get the optimal solution. We use marking $x_{k,i}$, $\nabla_x L_{k,i}$, $[A^T(A\vec{x}_k - \vec{b})]_i$ to denote the i-th component of the vectors. We simplify the notation of L_k, P_k, $\vec{d}_k$ etc. instead of $L_k(x_k, x''_k, \theta_k)$, $P_k(x_k, x''_k, \theta_k)$ etc. for a more compact and transparent description. Let us remind the reader that $\varepsilon > 0$, $\vec{x}_k \geq \vec{0}$, $\vec{c} \geq \vec{0}$ and that according to Slater's conditions [Boyd 04] the duality gap is zero for our problem.
The Lagrangian function of problem (3.4.1) is:

$$L_k = \vec{c}^T\vec{x}_k + \varepsilon(\vec{x}_k - \vec{x}''_k)^T(\vec{x}_k - \vec{x}''_k) + \vec{\theta}_k^T(A\vec{x}_k - \vec{b}) \tag{3.4.13}$$

We rewrite the algorithm Equations (3.4.9) into:

$$\begin{array}{lcl} \vec{x}_k & = & \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]^+ \\ \vec{\theta}_{k+1} & = & \vec{\theta}_k + \alpha\nabla_{\theta}L_k \\ \vec{x}''_{k+1} & = & \vec{x}''_k - \alpha\nabla_{x''}L_k \end{array} \tag{3.4.14}$$

It can be easy verify, that Equations (3.4.14) are identical with Equations (3.4.9). We define two diagonal matrices I'_k and I''_k as:

$$I'_{k,i,j} = \begin{cases} 1, & i = j \text{ and } \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.4.15}$$

$$I''_{k,i,j} = \begin{cases} 1, & i = j \text{ and } \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]_i \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.4.16}$$

Now we can write $\vec{x}_k = I'_k\left(\vec{x}''_k - \frac{1}{2\varepsilon}(\vec{c} + A^T\vec{\theta}_k)\right)$. Then by differentiations of the Lagrangian function (3.4.13) we get:

$$\nabla_{x''}L_k = -2\varepsilon(\vec{x}_k - \vec{x}''_k) = I'_k(\vec{c} + A^T\vec{\theta}_k) + I''_k 2\varepsilon\vec{x}''_k \tag{3.4.17}$$

$$\nabla_{\theta}L_k = A\vec{x}_k - \vec{b} \tag{3.4.18}$$

$$\begin{array}{lcllcl} \nabla^2_{x''x''}L_k & = & I''_k 2\varepsilon & \nabla^2_{\theta x''}L_k & = & AI'_k \\ \nabla^2_{x''\theta}L_k & = & I'_k A^T & \nabla^2_{\theta\theta}L_k & = & -\frac{1}{2\varepsilon}AI'_k A^T \end{array} \tag{3.4.19}$$

We define merit function P_k as:

$$P_k = 0.5|\nabla_{x''}L_k|^2 + 0.5|\nabla_{\theta}L_k|^2 \tag{3.4.20}$$

According to the Karush-Kuhn-Tucker conditions (for details see [Boyd 04], [Bert 99]) if $P_k = 0$ the solution in the k-th iteration of the algorithm is the optimal solution of problem (3.4.1). The gradient of the merit function P_k is:

$$\nabla P_k = \begin{bmatrix} \nabla^2_{x''x''}L_k \nabla_{x''}L_k + \nabla^2_{x''\theta}L_k \nabla_{\theta}L_k \\ \nabla^2_{\theta x''}L_k \nabla_{x''}L_k + \nabla^2_{\theta\theta}L_k \nabla_{\theta}L_k \end{bmatrix} = \begin{bmatrix} I''_k 2\varepsilon L_k \nabla_{x''}L_k + I'_k A^T(A\vec{x}_k - \vec{b}) \\ AI'_k \nabla_{x''}L_k - \frac{1}{2\varepsilon}AI'_k A^T(A\vec{x}_k - \vec{b}) \end{bmatrix} \tag{3.4.21}$$

We can define a column change vector as $\vec{d}_k = \alpha[-\nabla_{x''}L_k^T, (A\vec{x}_k - \vec{b})^T]^T$. For $\alpha \to 0$ we can express one iteration step of the merit function (3.4.20):

$$\begin{array}{rcl} \vec{d}_k^T \nabla P_k & = & \alpha\Big(- \nabla_{x''}L_k^T 2\varepsilon I''_k \nabla_{x''}L_k - \nabla_{x''}L_k^T I'_k A^T(A\vec{x}_k - \vec{b}) \\ & + & (A\vec{x}_k - \vec{b})^T AI'_k \nabla_{x''}L_k - \frac{1}{2\varepsilon}(A\vec{x}_k - \vec{b})^T AI'_k A^T(A\vec{x}_k - \vec{b})\Big) \\ & = & -\alpha\Big(\nabla_{x''}L_k^T 2\varepsilon I''_k \nabla_{x''}L_k + \frac{1}{2\varepsilon}(A\vec{x}_k - \vec{b})^T AI'_k A^T(A\vec{x}_k - \vec{b})\Big) \leq 0 \end{array} \tag{3.4.22}$$

From Equation (3.4.22), it results that $\vec{d}_k^T \nabla P_k \leq 0$. I.e. the merit function P_k is non-increasing during the algorithm.

Let us assume, that there is a k_0, such as the merit function P_k. is non-decreasing for all $\forall_{k \geq k_0}$. (i.e. $\vec{d}_k^T \nabla P_k = 0$ for $\forall_{k \geq k_0}$) From Equation (3.4.22), we can write for $\forall_{k \geq k_0}$:

$$\begin{array}{rcll} I''_k \nabla_{x''}L_k & = & \vec{0} & \forall_{k \geq k_0} \\ I'_k A^T(A\vec{x}_k - \vec{b}) & = & \vec{0} & \forall_{k \geq k_0} \end{array} \tag{3.4.23}$$

And it follows:

$$I''_k \vec{x}''_k = \vec{0} \qquad \forall_{k \geq k_0} \tag{3.4.24}$$

Let us suppose for some $k \geq k_0$ that $I'_k \neq I'_{k+1}$ and for some index i holds:

$$i : \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]_i > 0 \text{ and } \left[\vec{x}''_{k+1} - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_{k+1}\right)\right]_i \leq 0 \quad (3.4.25)$$

Then we can write from (3.4.25):

$$\left[\vec{x}''_{k+1} - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_{k+1}\right)\right]_i = \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right) - \alpha\left(\vec{c} + A^T\vec{\theta}_k\right)\right]_i \leq 0 \quad (3.4.26)$$

From (3.4.26) follows:

$$\left[\vec{c} + A^T\vec{\theta}_k\right]_i > 0 \quad (3.4.27)$$

Under the condition $\alpha < 1/(2\varepsilon)$ we write from Equations (3.4.25), (3.4.17) and (3.4.14):

$$0 < \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]_i < \left[\vec{x}''_k - \alpha\left(\vec{c} + A^T\vec{\theta}_k\right)\right]_i = \left[\vec{x}''_k - \alpha\nabla_{x''}L_k\right]_i = \vec{x}''_{k+1,i} \quad (3.4.28)$$

It means $\vec{x}''_{k+1,i} \neq 0$, which is in contradiction with the conditions (3.4.24) and (3.4.23). It follows that the condition (3.4.25) cannot be satisfy for any $k \geq k_0$.

Because there is no i satisfying condition (3.4.25) for $\forall_{k\geq k_0}$, we can write:

$$rank|I'_k| \leq rank|I'_{k+1}| \qquad \forall_{k\geq k_0} \quad (3.4.29)$$

Moreover for all i which satisfy $i : \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]_i > 0$ for some $k \geq k_0$, we get from Equation (3.4.28):

$$\left[\vec{c} + A^T\vec{\theta}_k\right]_i \leq \vec{0} \qquad \forall_{k\geq k_0} \quad (3.4.30)$$

And from (3.4.17) under the condition (3.4.30) and (3.4.24) we get:

$$\nabla_{x''}L_k \leq \vec{0} \qquad \forall_{k\geq k_0} \quad (3.4.31)$$

The result of Equation (3.4.29) is that, there is a finite number of changes

of the matrices I'_k and I''_k such as $I'_k \neq I'_{k+1}$ and $I''_k \neq I''_{k+1}$ for $k \geq k_0$. So for some $k_1 \geq k_0$ holds $I'_k = I'_{k+1}$ for all $\forall_{k \geq k_1}$. Based on this fact we can express from Equations (3.4.14) $\vec{x}_{k+1}$ using (3.4.23) for all $\forall_{k \geq k_1}$:

$$\begin{aligned} \vec{x}_{k+1} &= I'_{k+1}\Big[\vec{x}''_{k+1} - \tfrac{1}{2\varepsilon}\big(\vec{c} + A^T\vec{\theta}_{k+1}\big)\Big] \\ &= I'_k\Big[\vec{x}''_k - \tfrac{1}{2\varepsilon}\big(\vec{c} + A^T\vec{\theta}_k\big) - \alpha\nabla_{x''}L_k - \tfrac{\alpha}{2\varepsilon}A^T(A\vec{x}_k - \vec{b})\Big] \\ &= \vec{x}_k - \alpha\nabla_{x''}L_k \end{aligned} \tag{3.4.32}$$

and from (3.4.23) and (3.4.32) follows for all $\forall_{k \geq k_1}$:

$$\begin{aligned} 0 &= \nabla_{x''}L_k^T A^T(A\vec{x}_{k+1} - \vec{b}) = \nabla_{x''}L_k^T\big(A^T(A\vec{x}_k - \vec{b}) - \alpha A^T A\nabla_{x''}L_k\big) \\ &= \alpha\nabla_{x''}L_k^T A^T A\nabla_{x''}L_k \end{aligned} \tag{3.4.33}$$

From (3.4.33) we get a condition, which says that for all $\forall_{k \geq k_1}$ the change of variable $\vec{x}''_k$ is a circulation in the network:

$$A\nabla_{x''}L_k = \vec{0} \qquad \forall_{k \geq k_1} \tag{3.4.34}$$

Based on (3.4.34), (3.4.23), (3.4.31) and $\vec{c} \geq \vec{0}$ we write:

$$0 \leq \nabla_{x''}L_k^T\nabla_{x''}L_k = \nabla_{x''}L_k^T(\vec{c} + A^T\vec{\theta}_k) = \nabla_{x''}L_k^T\vec{c} \leq 0 \tag{3.4.35}$$

And the result of (3.4.35) is:

$$\nabla_{x''}L_k = \vec{0} \qquad \forall_{k \geq k_1} \tag{3.4.36}$$

To express $(A\vec{x}_{k_1} - \vec{b})$ consider problem (3.4.1) with both optimization variables x, x''. According to the Karush-Kuhn-Tucker conditions and Equation (3.4.36) we can write a dual function of this problem. Please notice that $\vec{x}_k\nabla_x L_k = 0$ for all $\forall_{k \geq k_1}$ according to (3.4.36) and (3.4.17).

$$g(\vec{\theta}_k) = \min_{\vec{x} \geq \vec{0}, \vec{x}''} L(\vec{x}, \vec{x}'', \vec{\theta}_k) = \vec{c}^T\vec{x}_{k_1} + \varepsilon(\vec{x}_{k_1} - \vec{x}''_{k_1})^T(\vec{x}_{k_1} - \vec{x}''_{k_1}) + \vec{\theta}_k^T(A\vec{x}_{k_1} - \vec{b}) \tag{3.4.37}$$

Then for $k+1$ iteration holds:

$$g(\vec{\theta}_{k+1}) = g(\vec{\theta}_k) + \alpha(A\vec{x}_{k_1} - \vec{b})^T(A\vec{x}_{k_1} - \vec{b}) \tag{3.4.38}$$

From Equation (3.4.38), it follows that for $(A\vec{x}_{k_1} - \vec{b}) \neq 0$ holds: if $k \to \infty$ then $g(\vec{\theta}_k) \to \infty$. According to the duality gap theorem (see e.g. [Bert 99]), it holds: $\max_{\vec{\theta}} g(\vec{\theta}) \leq \min_{\vec{x},\vec{x}'' \in \mathcal{S}} f(\vec{x}, \vec{x}'')$, where $f(\vec{x}, \vec{x}'')$ represents the primal function and $\mathcal{S}$ a set of feasible solutions. It follows, if $(A\vec{x}_{k_1} - \vec{b}) \neq 0$ and the merit function (3.4.20) is non-decreasing then the original problem is not feasible. It follows for feasible problems:

$$(A\vec{x}_k - \vec{b}) = 0 \quad \forall_{k \geq k_1} \tag{3.4.39}$$

We have presented a merit function P_k which is non-increasing during the algorithm for some $\alpha \to 0$ and which is equal to zero $P_k = 0$ for the optimal feasible solution. Next, we have shown for feasible problems that if the merit function P_k is non-decreasing for $\forall_{k \geq k_1}$ then according to (3.4.36) and (3.4.39) the merit function $P_k = 0$ and the solution $(\vec{x}_k, \vec{x}''_k, \vec{\theta}_k)$ is the optimal solution of the original problem (3.4.1).

3.4.4 Open Issues

The distributed routing algorithms, which have been derived in this chapter, are based on the sub-gradient method with a constant optimization step α. In this section we briefly present a possible extension of the derived algorithms, which is based on the Newton's method and which can significantly improve the algorithms convergence for dense networks, where each node has many communication links. As the dense networks are quite common in the sensor networks area, this extension can have a significant impact. We present our preliminary experiments to show the extension potential.

Unfortunately, the Newton's method cannot be directly used in this case, because it would not allow deriving the in-network distributed algorithm. However, the algorithm still can be adjusted by heuristic and approximations based on the Newton's method such as Regula falsi method (e.g. [Sigl 02]), quasi-Newton methods, or diagonal approximation (e.g. [Bert 99]).

In this section we focus on the algorithm extension based on the *diagonal approximation to Newton's method* [Bert 99] and on an experimental verification of the algorithm convergence and its comparison to OLDRAo algorithm.

We start with Equations (3.4.9) of the OLDRAo algorithm and adjust them into new form:

$$\begin{array}{lcl} \vec{x}_k & = & \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]^+ \\ \vec{\theta}_{k+1} & = & \vec{\theta}_k + Q_k^{-1}(A\vec{x}_k - \vec{b}) \\ \vec{x}''_{k+1} & = & \vec{x}''_k + 2\alpha\varepsilon(\vec{x}_k - \vec{x}''_k) \end{array} \tag{3.4.40}$$

where the matrix $Q_k \in \mathcal{R}^{N\times N}$ is a diagonal matrix:

$$Q_{k,n,n} = \max\left(\alpha^{-1}, \quad \frac{[(AI'_kA^T)]_{n,n}}{2\varepsilon}\right) \tag{3.4.41}$$

where I'_k is defined as:

$$I'_{k,i,j} = \begin{cases} 1, & i = j \text{ and } \left[\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right)\right]_i \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.4.42}$$

Please note, that the expression $\frac{(AI'_kA^T)}{2\varepsilon} = \nabla_{\theta\theta}W(\vec{x}'', \vec{\theta})$ and therefore the diagonal components of the matrix Q_k are equal to the diagonal components of the Hessian matrix $\nabla_{\theta\theta}W(\vec{x}'', \vec{\theta})$.

Moreover, according to the properties of the *Diagonally Dominant Matrices* (for details see e.g. [Saad 03]) we can write for dense networks:

$$\left(\text{DIAG}\left(\frac{[(AA^T)]_{n,n}}{2\varepsilon}\right)\right)^{-1} \cdot \frac{(AA^T)}{2\varepsilon} \to I$$

where DIAG(..) is a diagonal matrix.

The individual components of the matrix Q_k can be easily computed directly in the corresponding nodes as the number of "opened" communication links (i.e. $\vec{x}''_k - \frac{1}{2\varepsilon}\left(\vec{c} + A^T\vec{\theta}_k\right) \geq 0$).

We have performed a set of experiments with random initial variables with the same parameters as in Section 3.4.2.2. The $\alpha = 0.1$ for the extended OLDRAo.

The algorithm has been run 2000 times on random networks for both OLDRAo and for the extended OLDRAo. The results are presented in Figure 3.4.7. There is the number of iterations placed on the horizontal axis and the number of finished experiments on the vertical axis.

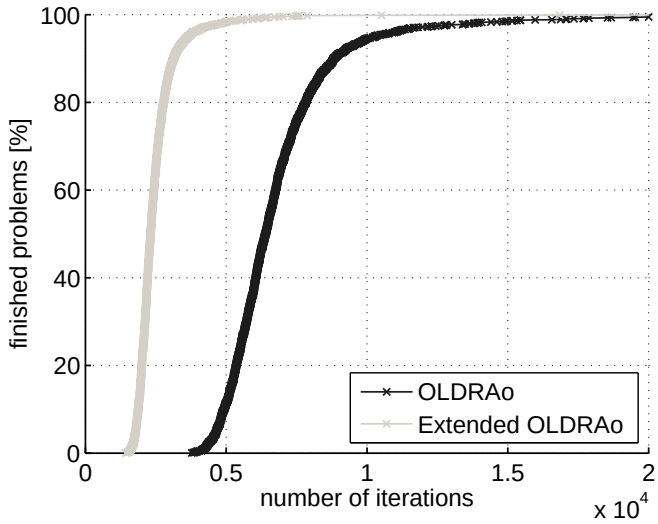


Fig. 3.4.7: Extended OLDRAo: Algorithm convergence with random initial variables and its comparison with OLDRAo.

This experiment provides practical verification of the algorithm convergence. 95% of the experiments have been finished in 3637 iterations for extended OLDRAo and in 10283 iterations for the OLDRAo. The improvement of the algorithm for 95% of experiments is more than 2.8 times. The average improvement over all experiments is 4326 iterations, which is an improvement bigger than 2.7 times.

According to these experiments, the algorithms convergence can be significantly improved in the future work. However, a proper mathematical derivation, proof of the algorithm convergence and more precise experimental verification are needed in this area. This section has been included into this work to present the potential of the derived algorithm in terms of convergence improvement and to suggest the future work in this area.

3.5 Summary

In this chapter we have presented three distributed algorithms for the energy optimal data flow routing in sensor networks. The algorithms differ in derivation approaches which lead to their different variants. The most advance algorithm is the OLDRAo which uses the basics from the other two algorithms derivation and which has presented the best convergence.

We have described the routing problem as a multi-commodity network flow optimization problem and used the dual decomposition method to mathematically derive the distributed algorithms. The algorithms do not need any central computational node with knowledge about the whole network structure. This rapidly increases the robustness of the algorithms in the case of partial network damage. The algorithms use only peer-to-peer communication between the neighboring nodes which allows the routing update using only the local communication. We have performed several simulation experiments to evaluate the algorithms behavior and to test their convergence. The mathematical proofs of the algorithms convergence are presented.

The main purpose of this chapter was to present the basic concept of new in-network distributed routing algorithms. From the experimental section it is seen that the presented algorithms are not application ready because of the high number of iterations, which would lead to high number of communications. However the main strength of the algorithms is not to find the whole optimal routing in unknown networks, but to adapt the existing routing in case of local network changes (dead/new node, loss of connection, etc.) where the number of data communications could be significantly decreased. Moreover, the algorithm can easily adapt to slow network changes such as slow communication costs or capacity changes and data volume changes.

As mentioned in Introduction, the other works from the area of network utility maximization (NUM) concentrate only on the strictly convex optimization problems, or they approximate the linear problems as strictly convex. They fail in the case of linear objective functions. According to our knowledge, this is the first work, which addresses and solves the problem of the dual decomposition of NUMs for problems with linear objective functions. Pleas note, that the algorithm for problems with the linear objective function is more complex and that the linearity of the objective functions makes the proof of the algorithm convergence more difficult in comparison to the strictly convex cases.

Unfortunately, as the presented approach of dual decomposition of the routing problems in node-link form is a new technique, there are only few works in this area at this moment. The most similar work can be found in [Zhen 10]. However, it is focused on slightly different problems. It uses a different energy consumption model and it is limited only on problems with strictly convex objective functions. Our work is focused on the problems with linear objective functions and presents different quality then [Zhen 10]. Due to the differences in the used models and algorithms derivations the experimental comparison would have a disputable contribution and would be strongly dependent on the chosen problems.

According to our preliminary experiments the number of iterations can be significantly decreased in future work. The algorithm can be extended by heuristics based on the partial knowledge about the network structure (e.g. node geographical position) and heuristics based on Newton's method. The results of our preliminary testing in Section 3.4.4 indicate that the Newton's method based heuristics can decrease the number of iterations more then 2.7 times.

Considering the fact, that the algorithms are based on Linear programming formulation, we believe that the principle of the algorithms and the approaches used to their derivation can be used to solve many different problems in the sensor networks area, like resource sharing, network localization, object tracking, etc.

Chapter 4

Distributed Algorithms for Real-Time Routing

4.1 Introduction

In this chapter, we propose a distributed algorithm, which computes the energy optimal real-time routing without the need of any central computational or data point. It is based on the modified multi-commodity network flow model for real-time routing which is described in Chapter 2. It uses the fact that the problem definition after the network replication stays in the form of network flow problem with side constraints. And that the side constraints are in form, which allows the problem distribution as an in-network algorithm. We apply the distribution approach (Chapter 3) on the real-time routing problem in this chapter. According to our knowledge, this work is first, which solves the real-time routing problem with linear cost functions and constant communication delays, using the dual decomposition.

4.1.1 Related Works

Beside the ad-hoc real-time routing algorithms like SPEED, RPAR and others referenced in Section 2.1.2 which route the data according to actual parameters (like time remaining to the deadline, message priority, network load etc.), there are several works, which focus on the algorithm distributed from their centralized mathematical description and use the precomputed routing

paths. Some of the easy algorithms are based e.g. on the Dijkstra's algorithm, or on the network flooding principles (see e.g. [Karl 05]). The more sophisticated algorithms are based on the convex optimization theory and use the dual decomposition to derive their distributed version.

In [Zhen 10] the authors use the network flow problem formulation in node-link form with strictly convex objective function to derive the distributed routing algorithm. Further, the authors extend the algorithm to optimize a queuing delay which is strictly convex function of the total flow routed through the links. This approach ignores the constant communication delay, which is independent on the volume of the routed flow. In case of a high flow fragmentation this approach cannot ensure the messages deadlines satisfaction. Moreover, the queuing delay optimization cannot be used in the case of pre-scheduled communication based on the TDMA principles (e.g. GTS allocation in IEEE 802.15.4).

In [Anas 08] the authors derive a distributed routing algorithm, where they minimize communication delay, which is caused by computation in the nodes. The authors focus on the time needed for the messages decoding and encoding in the nodes to check and regenerate the corrupted data. The objective function of this problem is linear. The authors use the quadratic approximation of the objective function to derive the distributed algorithm.

The distributed algorithms based on the node-path routing formulation [Tsit 86, Low 99], which have been mentioned in Section 3.1.1 can be directly adopted for the real-time routing problems. The algorithms optimize the data flow routed through selected paths. If only paths with acceptable communication delay are chosen, the algorithms can ensure the maximum communication delay. However, the node-path problem formulation is well usable only for networks with small number of possible routing paths. For the multi-hop networks with high density like sensor networks a different approach has to be derived.

4.2 Mathematical Derivation of the Real-Time Distributed Routing Algorithm

To derive the distributed algorithm for Real-time routing, we use the multi-commodity network flow model for real-time routing which is derived in Chapter 2 and its constraints are described by Equations (2.3.4). The model

is written as a Linear Programming problem:

$$\min_{\vec{x},\vec{s}} \quad \vec{c}^T \sum_{m\in\mathcal{M}} \sum_{w=1}^{d^{(m)}} \vec{x}^{(m,w)}$$

subject to:

$$\begin{aligned}
A^- \vec{x}^{(m,w+1)} + \vec{s}^{(m,w)} &= A^+ \vec{x}^{(m,w)} + \vec{s}_{in}^{(m,w)} && \forall m \in \mathcal{M},\ 0 \le w \le d^{(m)} \\
D \sum_{m\in\mathcal{M}} \sum_{w=1}^{d^{(m)}} \vec{x}^{(m,w)} &\le \vec{\mu} \\
\sum_{w=0}^{d^{(m)}} \vec{s}^{(m,w)} &= \vec{s}_{out}^{(m)} && \forall m \in \mathcal{M} \\
\vec{x}^{(m,w)} \ge \vec{0};\quad \vec{s}^{(m,w)} &\ge \vec{0} && \forall m \in \mathcal{M},\ 0 \le w \le d^{(m)} \\
\vec{x}^{(m,0)} = \vec{x}^{(m,d^{(m)}+1)} &= \vec{0} && \forall m \in \mathcal{M}
\end{aligned} \tag{4.2.1}$$

To decompose the routing algorithm, we use the proximal-point method (3.1.2) to modify the problem into strictly convex form, which allows the usage of the gradient methods. Moreover, we rewrite the problem into equality form for a more transparent presentation. The new model can be described as a convex optimization problem:

$$\min_{\vec{x}'',\vec{z}'',\vec{s}''} \min_{\vec{x},\vec{z},\vec{s}} \quad g(\vec{x},\vec{x}'',\vec{s},\vec{s}'',\vec{z},\vec{z}'')$$

subject to:

$$\begin{aligned}
A^- \vec{x}^{(m,w+1)} + \vec{s}^{(m,w)} &= A^+ \vec{x}^{(m,w)} + \vec{s}_{in}^{(m,w)} && \forall m \in \mathcal{M},\ 0 \le w \le d^{(m)} \\
D \sum_{m\in\mathcal{M}} \sum_{w=1}^{d^{(m)}} \vec{x}^{(m,w)} &= \vec{\mu} - \vec{z} \\
\sum_{w=0}^{d^{(m)}} \vec{s}^{(m,w)} &= \vec{s}_{out}^{(m)} && \forall m \in \mathcal{M} \\
\vec{x}^{(m,w)} \ge \vec{0};\quad \vec{s}^{(m,w)} &\ge \vec{0} && \forall m \in \mathcal{M},\ 0 \le w \le d^{(m)} \\
\vec{x}^{(m,0)} = \vec{x}^{(m,d^{(m)}+1)} &= \vec{0} && \forall m \in \mathcal{M} \\
\vec{z} &\ge \vec{0};
\end{aligned} \tag{4.2.2}$$

where the objective function $g(\vec{x}, \vec{x}'', \vec{s}, \vec{s}'', \vec{z}, \vec{z}'')$ is:

$$\begin{aligned} g(\vec{x}, \vec{x}'', \vec{s}, \vec{s}'', \vec{z}, \vec{z}'') &= \vec{c}^T \sum_{m \in \mathcal{M}} \sum_{w=1}^{d^{(m)}} \vec{x}^{(m,w)} \\ &+ \varepsilon (\vec{z} - \vec{z}'')^T (\vec{z} - \vec{z}'') \\ &+ \varepsilon \sum_{m \in \mathcal{M}} \sum_{w=1}^{d^{(m)}} (\vec{x}^{(m,w)} - \vec{x}''^{(m,w)})^T (\vec{x}^{(m,w)} - \vec{x}''^{(m,w)}) \\ &+ \varepsilon \sum_{m \in \mathcal{M}} \sum_{w=0}^{d^{(m)}} (\vec{s}^{(m,w)} - \vec{s}''^{(m,w)})^T (\vec{s}^{(m,w)} - \vec{s}''^{(m,w)}) \end{aligned} \tag{4.2.3}$$

We have added slack variables $\vec{z} \geq \vec{0}$ into problem (4.2.1) to convert the problem into the equality form. The variables $\vec{x}''$, $\vec{s}''$ and $\vec{z}''$ in Equation (4.2.3) are the proximal-point variables, which has been added to convert the objective function from linear to strictly convex. Please notice that the set of optimal solutions for problem (4.2.2) is the same as for the original problem (4.2.1). For the optimal solution of problem (4.2.2) holds $\vec{x} = \vec{x}''$, $\vec{s} = \vec{s}''$, $\vec{z} = \vec{z}''$. In this way the routing problem has been separated into two nested subproblems. The internal subproblem is the minimization over the variables $\vec{x}$, $\vec{s}$, $\vec{z}$ and it is strictly convex. The outer subproblem minimizes the internal one by the proximal-point variables $\vec{x}''$, $\vec{s}''$, $\vec{z}''$.

4.2.1 Dual Problem

To solve the internal subproblem of (4.2.2) (minimization over the variables $\vec{x}, \vec{z}, \vec{s}$) we present its dual problem, which allows one to derive the distributable gradient algorithm. According to Slater's conditions (see e.g. [Boyd 04]) the optimal solutions of the dual and primal problems have the same optimal values in this case.
The Lagrangian function of problem (4.2.2) is:

$$\begin{aligned} L(\vec{x}, \vec{x}'', \vec{s}, \vec{s}'', \vec{z}, \vec{z}'', \vec{\theta}, \vec{\lambda}, \vec{\gamma}) &= g(\vec{x}, \vec{x}'', \vec{s}, \vec{s}'', \vec{z}, \vec{z}'') \\ &+ \sum_{m \in \mathcal{M}} \sum_{w=0}^{d^{(m)}} \vec{\theta}^{(m,w)^T} (A^- \vec{x}^{(m,w+1)} - A^+ \vec{x}^{(m,w)} + \vec{s}^{(m,w)} - \vec{s}_{in}^{(m,w)}) \\ &+ \vec{\lambda}^T (D \sum_{m \in \mathcal{M}} \sum_{w=1}^{d^{(m)}} \vec{x}^{(m,w)} + \vec{z} - \vec{\mu}) \\ &+ \vec{\gamma}^{(m)^T} (\sum_{w=0}^{d^{(m)}} \vec{s}^{(m,w)} - \vec{s}_{out}^{(m)}) \end{aligned} \tag{4.2.4}$$

where $\vec{x} \geq \vec{0}$, $\vec{s} \geq \vec{0}$ and $\vec{z} \geq \vec{0}$ are the primal variables and $\vec{\theta}$, $\vec{\lambda}$ and $\vec{\gamma}$ are the dual variables. The dual function W is:

$$W(\vec{x}'', \vec{s}'', \vec{z}'', \vec{\theta}, \vec{\lambda}, \vec{\gamma}) = \min_{\vec{x},\vec{s},\vec{z}\geq\vec{0}} L(\vec{x}, \vec{x}'', \vec{s}, \vec{s}'', \vec{z}, \vec{z}'', \vec{\theta}, \vec{\lambda}, \vec{\gamma}) \tag{4.2.5}$$

The minimizers of the dual function (4.2.5) are:

$$\begin{aligned}
\vec{x}_{min}^{(m,w)} &= \left[\vec{x}''^{(m,w)} - \quad \tfrac{1}{2\varepsilon}(A^{-T}\vec{\theta}^{(m,w-1)} - A^{+T}\vec{\theta}^{(m,w)} + D^T\vec{\lambda} + \vec{c})\right]^+ \\
\vec{z}_{min} &= \left[\vec{z}'' - \tfrac{1}{2\varepsilon}\vec{\lambda}\right]^+ \\
\vec{s}_{min}^{(m,w)} &= \left[\vec{s}''^{(m,w)} - \tfrac{1}{2\varepsilon}\vec{\gamma}^{(m)}\right]^+
\end{aligned} \tag{4.2.6}$$

where symbol $[..]^+$ denotes a positive or zero value in each component of vector $[..]^+ = max(\vec{0}, ..)$ and $\vec{x}^{(m,0)} = \vec{x}^{(m,d^{(m)}+1)} = \vec{0}$.
The dual problem of the internal subproblem of (4.2.2) is:

$$U(\vec{x}'', \vec{s}'', \vec{z}'') = \max_{\vec{\theta},\vec{\lambda},\vec{\gamma}} W(\vec{x}'', \vec{s}'', \vec{z}'', \vec{\theta}, \vec{\lambda}, \vec{\gamma}) \tag{4.2.7}$$

The gradients of the dual function (4.2.5) are:

$$\begin{aligned}
\nabla_{\theta} W^{(m,w)} &= A^{-}\vec{x}_{min}^{(m,w+1)} - A^{+}\vec{x}_{min}^{(m,w)} + \vec{s}_{min}^{(m,w)} - \vec{s}_{in}^{(m,w)} \\
\nabla_{\lambda} W^{(m)} &= D \sum_{m\in\mathcal{M}} \sum_{w=1}^{d^{(m)}} \vec{x}_{min}^{(m,w)} + \vec{z}_{min} - \vec{\mu} \\
\nabla_{\gamma} W &= \sum_{w=0}^{d^{(m)}} \vec{s}_{min}^{(m,w)} - \vec{s}_{out}^{(m)}
\end{aligned} \tag{4.2.8}$$

The gradients of dual problem (4.2.7) are:

$$\begin{aligned}
\nabla_{x''} U^{(m,w)} &= -2\varepsilon(\vec{x}_{min}^{(m,w)} - \vec{x}''^{(m,w)}) \\
\nabla_{\vec{z}''} U &= -2\varepsilon(\vec{z}_{min} - \vec{z}'') \\
\nabla_{\vec{s}''} U^{(m,w)} &= -2\varepsilon(\vec{s}_{min}^{(m,w)} - \vec{s}''^{(m,w)})
\end{aligned} \tag{4.2.9}$$

4.2.2 Dual Gradient Algorithm

Using the dual problem (4.2.7) and the dual function (4.2.5) we rewrite the routing problem (4.2.2) into form:

$$\min_{\vec{x}'',\vec{z}'',\vec{s}''} \max_{\vec{\theta},\vec{\lambda},\vec{\gamma}} \min_{\vec{x},\vec{s},\vec{z}\geq\vec{0}} L(\vec{x}, \vec{x}'', \vec{s}, \vec{s}'', \vec{z}, \vec{z}'', \vec{\theta}, \vec{\lambda}, \vec{\gamma}) \tag{4.2.10}$$

A gradient algorithm created from problem (4.2.10) consists of 2 nested loops and it is described as (4.2.11). The internal loop solves the dual problem (4.2.7) using the gradients (4.2.8). The outer loop minimizes problem (4.2.10) over the proximal-point variables using gradients (4.2.9).

$$
\begin{array}{l}
\text{LOOP 1} \\
\quad \text{LOOP 2} \\
\qquad \text{Compute the } \vec{x}_{min},\ \vec{z}_{min},\ \vec{s}_{min}^{(m,w)} \\
\qquad\quad \text{according to Equations (4.2.6),} \\
\qquad\quad \text{and compute:} \\
\qquad \vec{\theta}^{(m,w)} = \vec{\theta}^{(m,w)} + \alpha \nabla_{\vec{\theta}} W^{(m,w)} \\
\qquad \vec{\gamma}^{(m)} = \vec{\gamma}^{(m)} + \alpha \nabla_{\vec{\gamma}} W^{(m)} \\
\qquad \vec{\lambda} = \vec{\lambda} + \alpha \nabla_{\vec{\lambda}} W \\
\quad \text{END 2} \\
\quad \vec{x}''^{(m,w)} = \vec{x}''^{(m,w)} + \alpha \nabla_{\vec{x}''} U^{(m,w)} \\
\quad \vec{z}'' = \vec{z}'' + \alpha \nabla_{\vec{z}''} U \\
\quad \vec{s}''^{(m,w)} = \vec{s}''^{(m,w)} + \alpha \nabla_{\vec{s}''} U^{(m,w)} \\
\text{END 1}
\end{array}
\tag{4.2.11}
$$

$\alpha > 0$ is a constant step size of the algorithm.

The important property of the algorithm (4.2.11) is that it can be distributed as in-network algorithm. Each node is responsible for computation of the flow volume of the links leaving the node and for computation of the other corresponding variables. All the components of the variable vectors are function of the node local variables and of the variables of the neighboring nodes that are within one hop communication distance. Only peer-to-peer communication during the algorithm is needed.

However, the distributed version of such algorithm would have problems with the termination of the internal loop and with the synchronization of the loops between the nodes. Moreover the nested loops would cause an increase in the iterations.

To derive one loop algorithm, we join both loops of the gradient algorithm into only one loop, where we update all the variables simultaneously. Using Equations (4.2.6), (4.2.8) and (4.2.9) we derive the iterative algorithm which is presented in Table 4.2.1. The k denotes the iteration number. Proof of the algorithm convergence is presented in Section 4.4.A necessary condition for the algorithm convergence assumed in the proof is $\alpha < 1/2\varepsilon$. We have

1. Initialize variables $\vec{x}_0''^{(m,w)}$, $\vec{s}_0''^{(m,w)}$, $\vec{z}_0''^{(m,w)}$, $\vec{\theta}_0^{(m,w)}$, $\vec{\lambda}_0$, $\vec{\gamma}_0^{(m)}$.
2. Compute primal variables $\vec{x}_k$, $\vec{z}_k$, $\vec{s}_k$ according to:

$$\vec{x}_k^{(m,w)} = \left[\vec{x}_k''^{(m,w)} - \tfrac{1}{2\varepsilon}(A^{-T}\vec{\theta}_k^{(m,w-1)} - A^{+T}\vec{\theta}_k^{(m,w)} + D^T\vec{\lambda}_k + \vec{c})\right]^+$$
$$\vec{z}_k = \left[\vec{z}_k'' - \tfrac{1}{2\varepsilon}\vec{\lambda}_k\right]^+$$
$$\vec{s}_k^{(m,w)} = \left[\vec{s}_k''^{(m,w)} - \tfrac{1}{2\varepsilon}\vec{\gamma}_k^{(m)}\right]^+$$
$$\vec{x}_k^{(m,0)} = \vec{x}_k^{(m,d^{(m)}+1)} = \vec{0}$$

3. Send/Receive the primal variables $\vec{x}_k$, $\vec{z}_k$, $\vec{s}_k$ to/from neighboring nodes.
4. Compute dual variables $\vec{\theta}_{k+1}$, $\vec{\lambda}_{k+1}$, $\vec{\gamma}_{k+1}$

$$\vec{\theta}_{k+1}^{(m,w)} = \vec{\theta}_k^{(m,w)} + \alpha(A^-\vec{x}_k^{(m,w+1)} - A^+\vec{x}_k^{(m,w)} + \vec{s}_k^{(m,w)} - \vec{s}_{in}^{(m,w)})$$
$$\vec{\lambda}_{k+1} = \vec{\lambda}_k + \alpha(D\sum_{m\in\mathcal{M}}\sum_{w=1}^{d^{(m)}}\vec{x}_k^{(m,w)} + \vec{z}_k - \vec{\mu})$$
$$\vec{\gamma}_{k+1}^{(m)} = \vec{\gamma}_k^{(m)} + \alpha(\sum_{w=0}^{d^{(m)}}\vec{s}_k^{(m,w)} - \vec{s}_{out}^{(m)})$$

5. Compute proximal-point variables $\vec{x}_{k+1}''$, $\vec{s}_{k+1}''$, $\vec{z}_{k+1}''$

$$\vec{x}_{k+1}''^{(m,w)} = \vec{x}_k''^{(m,w)} + \alpha(-2\varepsilon(\vec{x}_k^{(m,w)} - \vec{x}_k''^{(m,w)}))$$
$$\vec{z}_{k+1}'' = \vec{z}_k'' + \alpha(-2\varepsilon(\vec{z}_k - \vec{z}_k''))$$
$$\vec{s}_{k+1}''^{(m,w)} = \vec{s}_k''^{(m,w)} + \alpha(-2\varepsilon(\vec{s}_k^{(m,w)} - \vec{s}_k''^{(m,w)}))$$

6. Send/Receive the dual variables $\vec{\theta}_{k+1}$, $\vec{\lambda}_{k+1}$, $\vec{\gamma}_{k+1}$ to/from the neighboring nodes.
7. Set $k = k + 1$ and start new iteration in step 2.

Table 4.2.1: Distributed, Real-Time Routing Algorithm

performed several simulation experiments to test the algorithm convergence in Section 4.3.

The initial variables $\vec{x}_0''^{(m,w)}$, $\vec{s}_0''^{(m,w)}$, $\vec{z}_0''^{(m,w)}$, $\vec{\theta}_0^{(m,w)}$, $\vec{\lambda}_0$, $\vec{\gamma}_0^{(m)}$ are set to arbitrary initial values. The closer the values are to the final solution, the faster the algorithm converges. This property is used in Section 4.2.3 for variable initialization.

4.2.3 Variables Initialization

As we have mentioned in Section 4.2.2 the number of iterations depends on the initial variables $\vec{x}_0''^{(m,w)}$, $\vec{s}_0''^{(m,w)}$, $\vec{z}_0''^{(m,w)}$, $\vec{\theta}_0^{(m,w)}$, $\vec{\lambda}_0$, $\vec{\gamma}_0^{(m)}$. The closer the initial variables are to the final solution the less iterations are needed. In this work, we focus only on the dual variables. The proximal-point variables $\vec{x}_0''^{(m,w)}$, $\vec{s}_0''^{(m,w)}$, $\vec{z}_0''^{(m,w)}$ are set to zero. The variables $\vec{\lambda}_0$, $\vec{\gamma}_0^{(m)}$ cannot be estimated without the knowledge of the final routing. We set them to zero.

Let us suppose some initial variable $\vec{\theta}_0$ which satisfy condition:

$$A^{-T}\vec{\theta}_0^{(m,w-1)} - A^{+T}\vec{\theta}_0^{(m,w)} + D^T\vec{\lambda}_0 + \vec{c} \geq \vec{0} \tag{4.2.12}$$

According to Equations (4.2.6), such initial variables do not cause any changes of the algorithm variables on its own. (i.e. no variable changes if $\vec{s}_{out}^{(m)} = \vec{s}_{in}^{(m)} = 0$)

According to the complementary slackness condition (for details see e.g. [Bert 98]) for the optimal solution holds for two neighboring nodes: if $(\theta_{k,l^+}^{(m,w-1)} - \theta_{k,l^-}^{(m,w)}) < c_l$ then $x_{k,l}^{(m,w)} = 0$, if $(\theta_{k,l^+}^{(m,w-1)} - \theta_{k,l^-}^{(m,w)}) = c_l$ then $x_{k,l}^{(m,w)} \geq 0$ and the link l is a part of the shortest path for the demand $m \in \mathcal{M}$. (we use the index l to denote the vector component corresponding to the link l, l^+ to denote end node of the link and l^- to denote the start node of the link). This fact leads us directly to the Dijkstra's algorithm which can be used in distributed way. If for all sink nodes n_{out} we set $\theta_{0,n_{out}}^{(m,w)} = 0$ for all $\forall m \in \mathcal{M},\ 0 \leq w \leq d^{(m)}$ and for the other nodes n we set $\theta_{0,n}^{(m,w)} =$ *the shortest distance to sink node*, we get an initial setting, which satisfies the condition (4.2.12). Moreover, this initial setting is much closer to the optimal solution than the setting $\theta_{0,n}^{(m,w)} = 0$ for all $\forall n, m, w$. In Section 4.3.3 we present several experiments to evaluate the heuristic behavior in comparison with the zero initial setting. Please notice, that due to the capacity constraints, the heuristic solution does not need to be equal to the final optimal solution. According to our experiments this heuristic rapidly decreases the number of iterations.

4.3 Experiments

To demonstrate the behavior and the correctness of the distributed real-time routing algorithm, we have performed several experiments in Matlab. We have focused on a problem, where for each communication demand one node sends data flow to one sink node (i.e. a multi-commodity mono-source, mono-sink problem). I.e. $s^{(m)}_{in,n_1} = 1$ for the source node n_1 and $s^{(m)}_{out,n_2} = 1$ for the sink node n_2 of communication demand m.

The random networks for the experiments have been constructed as follows: We consider a square field of size $[size \times size]$, where the $size$ is changing during the experiments. The field is divided into sub-squares of size $[1 \times 1]$. One node is randomly placed into each sub-square and the communication distance is set to 1.7 (i.e. node n_1 can communicate with node n_2, if and only if their Euclidean distance is less than 1.7). Please notice, that our network is close to the "unit-disk network" [Rese 06]. The communication costs $\vec{c}$ per transmitted data flow unit have been set as the square power of the distance between the nodes. The node communication capacities have been set to two $\mu_n = 2$ and the maximum number of communication hops to $d^{(m)} = 6$ for all $\forall m \in \mathcal{M}$. The constants of the algorithm have been set as: $\alpha = 0.03$ and $\epsilon = 0.3$. The initial values $\vec{x}''^{(m,w)}_0$, $\vec{s}''^{(m,w)}_0$, $\vec{\theta}^{(m,w)}_0$, $\vec{\lambda}_0$, $\vec{\gamma}^{(m)}_0$ have been set to 0 and $z''_{0,n} = \mu_n$ for all experiments except 4.3.2 and 4.3.4. Only feasible problems are used.

During the experiments we evaluate k as a number of iterations needed to achieve less than 1% deviation of the objective function from the optimal value, less than 1% capacity violation and less than 1% flow conservation law violation during the last 500 iterations. The 500 iterations are included in the statistics. (the optimal value was computed separately by a centralized algorithm for evaluation purposes only)

4.3.1 Example

To present the resulting optimal data flow routing in the network we have performed an experiment based on the network described above. The field $size$ has been set to 9 (i.e. 81 nodes in the network) and the number of communication demands has been set to 8. The link capacities have been set to $\mu_n = 4$ for this problem. The initial variables $\vec{\theta}_0$ have been set according Section 4.2.3.

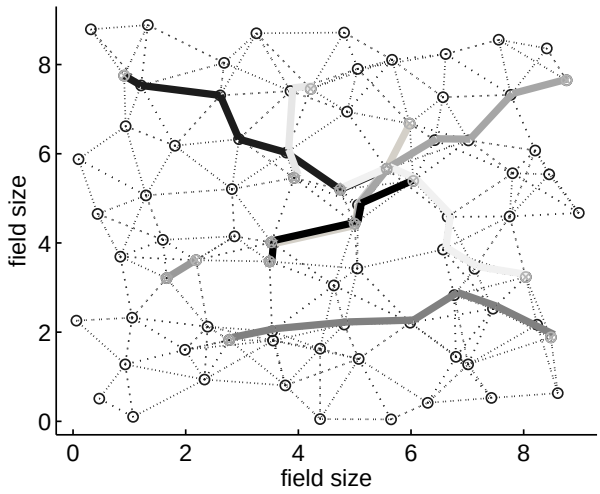


Fig. 4.3.1: Distributed, real-time routing: Optimal, real-time data flow routing (multi-commodity, mono-source, mono-sink problem).

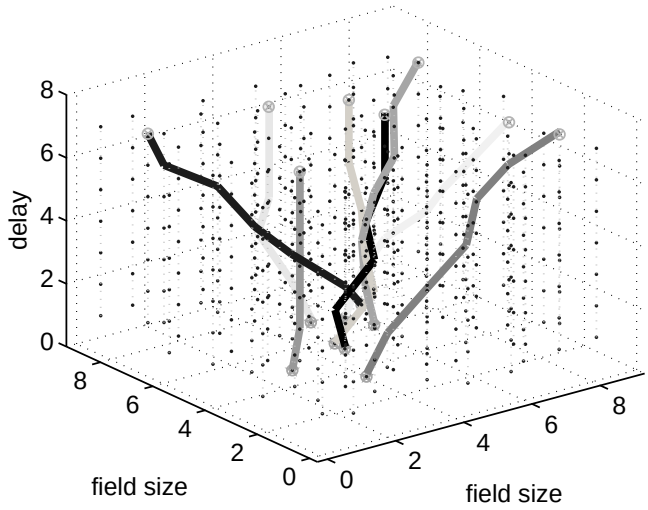


Fig. 4.3.2: Distributed, real-time routing: Optimal data flow routing in node-delay space.

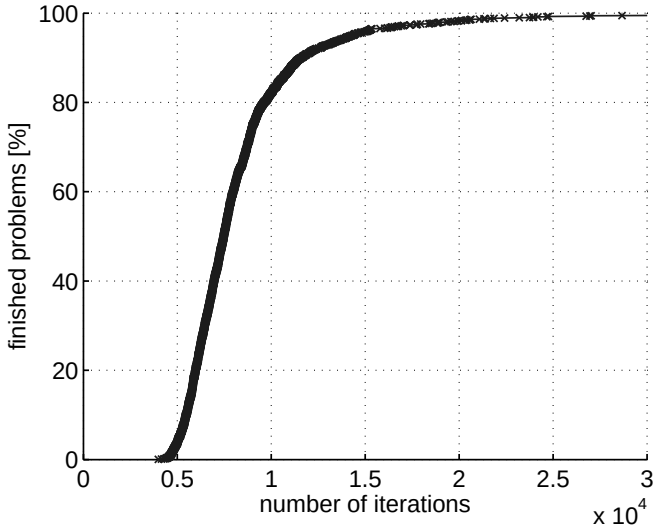


Fig. 4.3.3: Distributed, real-time routing: Algorithm convergence with random initial variables.

The optimal data flow routing in the network is shown in Figure 4.3.1.The 3D routing model in the node-delay space is presented in Figure 4.3.2 where the vertical axis represents the communication delay. No communication demand routing has more than 6 communication hops.

Several videos for presentation of the algorithm progress in time can be found on author web-pages [Trdl 11].

4.3.2 Algorithm Convergence

We have performed a set of experiments with random initial variables on random networks and we have evaluated the algorithm convergence. The field *size* has been set to 7. There are 8 communication demands in the network. The initial values have been set randomly from intervals: $\vec{x}_0^{\prime\prime(m,w)}, \vec{s}_0^{\prime\prime(m,w)}, \vec{z}_0^{\prime\prime(m,w)} \in \langle 0, 2\rangle$ for proximal-point variables, and $\vec{\theta}_0^{(m,w)}, \vec{\lambda}_0, \vec{\gamma}_0^{(m)} \in \langle 0, 20\rangle$ for dual variables. The intervals correspond to the double values of typical optimal values.

The algorithm has been run 2000 times on random networks. The results

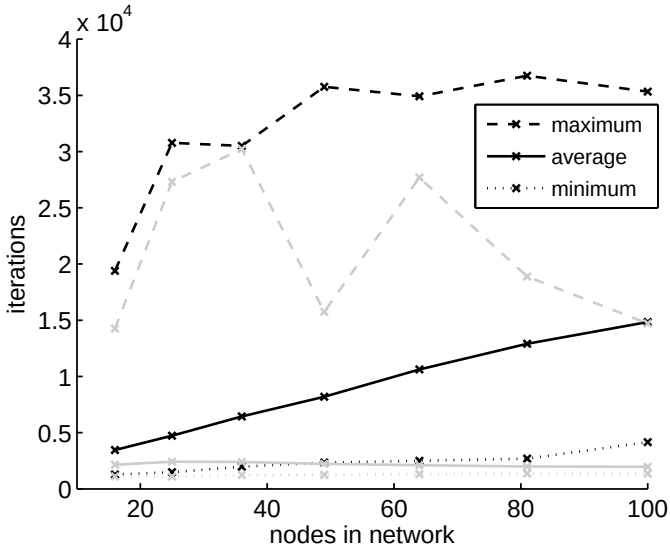


Fig. 4.3.4: Distributed, real-time routing: Number of iterations in relation to the number of nodes. In black, for zero initial variables. In gray, for initialization according to Section 4.2.3.

are presented in Figure 4.3.3. There is the number of iterations placed on the horizontal axis and the number of experiments, which has been finished before the number of iterations, on the vertical axis.

This experiment provides an important practical verification of the theoretical proof of the algorithm convergence. 95% of the experiments have been finished before 14300 iterations.

4.3.3 Number of Iterations

To demonstrate the statistical behavior of the algorithm, we have gradually increased the number of nodes from 16 to 100 (i.e. the field *size* from 4 to 10) for 8 communication demands. The computation has been repeated, on random networks, 1000 times for each number of nodes.

The results have been evaluated as a maximum, average and minimum number of iterations needed to achieve the optimal value and it is presented in Figure 4.3.4. There is presented the experiment progress for the basic algorithm without the initial heuristic in black (the initial variables have

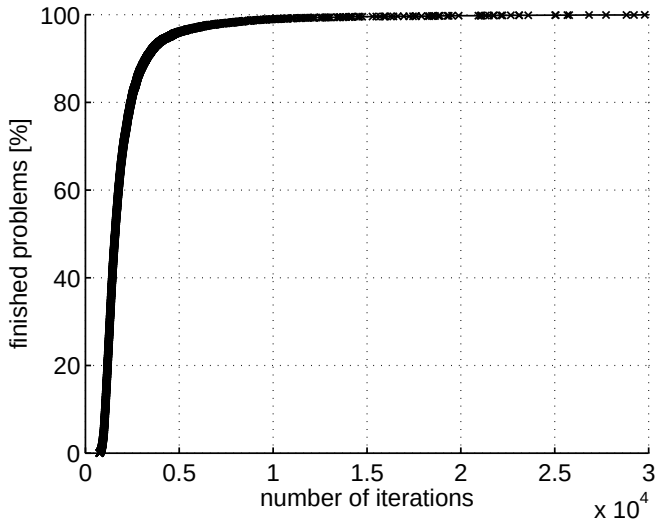


Fig. 4.3.5: Distributed, real-time routing: Algorithm convergence for network change simulation.

been set to zero) and for the algorithm with the initial heuristic from Section 4.2.3 in gray.

The number of the iterations is approximately linear, which means that the algorithm can be used in case of networks with many nodes. We can see a significant improvement of the initial heuristic in the figures.

4.3.4 Network Change

The advantage of the one-loop algorithm presented in this work is that it can automatically adjust the data routing in case of network changes. In order to evaluate the algorithm behavior in this case, we have simulated a dying node in the network as follow:

1. We generated a random network with communication demands and found the optimal solution

2. In the original problem from step 1, we removed one node and measured the number of iterations to find the new optimal solution

3. We repeated step 2 for 15 different nodes with largest data flow.

The simulation has been performed for 942 random original networks with field *size* set to 7 and with 8 communication demands. (i.e. $942 \times 15 = 14130$ experiments)

The results are presented in Figure 4.3.5. The number of iterations is placed on the horizontal axis and the number of experiments, which have been finished before the number of iterations are seen on the vertical axis.

95% of the experiments have been finished before 4466 iterations. Moreover, less than 15.0% of the dual variables, and 2.0% of the primal variables have been changed during the experiments, which significantly decrease the amount of transmitted data.

4.4 Proof of the Algorithm Convergence

To prove the convergence of the distributed real-time routing algorithm from Table 4.2.1, we rewrite the original real-time routing problem (4.2.2) into more compact form and we show that the algorithm equations can be written in the same form as Equations (3.4.14). According to the convergence proof in Section 3.4.3 the algorithm for Real-time routing converges to optimal solution.
We rewrite problem (4.2.2) into more transparent form:

$$\begin{array}{rcl} \min\limits_{\vec{x}''} \min\limits_{\vec{x}} & \vec{c}'^T\vec{x} + \varepsilon(\vec{x}-\vec{x}'')^T(\vec{x}-\vec{x}'') & \\ \textbf{subject to:} & & \\ \tilde{A}\vec{x} & = & \vec{b} \\ \vec{x} & \geq & \vec{0} \end{array} \tag{4.4.1}$$

An example of the matrix and vectors transformation is:

$$\tilde{A} = \begin{pmatrix} \tilde{B} & \tilde{I} & 0 \\ 0 & \tilde{S} & 0 \\ \tilde{D} & 0 & I \end{pmatrix} \quad \tilde{B} = \begin{pmatrix} \tilde{B}^1 & 0 & 0 \\ & \ddots & \\ 0 & 0 & \tilde{B}^M \end{pmatrix}$$

$$\tilde{B}^m = \begin{pmatrix} A^+ & A^- & 0 & 0 \\ 0 & A^+ & A^- & 0 \\ & & \ddots & \\ 0 & 0 & A^+ & A^- \end{pmatrix} \quad \tilde{S} = \begin{pmatrix} \tilde{S}^1 & 0 & 0 \\ & \ddots & \\ 0 & 0 & \tilde{S}^M \end{pmatrix}$$

$$\tilde{S}^m = \begin{pmatrix} I & I & \dots & I \end{pmatrix} \qquad \tilde{I} = \begin{pmatrix} I & 0 & 0 \\ & \ddots & \\ 0 & 0 & I \end{pmatrix}$$
$$\tilde{D} = \begin{pmatrix} D & D & \dots & D \end{pmatrix}$$

$$\vec{x}_k = \begin{bmatrix} \vec{x}_k^{(1,0)} \\ \vdots \\ \vec{x}_k^{(1,d^{(1)})} \\ \vdots \\ \vec{x}_k^{(M,d^{(M)})} \\ \vec{s}_k^{(1,0)} \\ \vdots \\ \vec{s}_k^{(1,d^{(1)})} \\ \vdots \\ \vec{s}_k^{(M,d^{(M)})} \\ \vec{z}_k \end{bmatrix} \qquad \vec{x}''_k = \begin{bmatrix} \vec{x}''^{(1,0)}_k \\ \vdots \\ \vec{x}''^{(1,d^{(1)})}_k \\ \vdots \\ \vec{x}''^{(M,d^{(M)})}_k \\ \vec{s}''^{(1,0)}_k \\ \vdots \\ \vec{s}''^{(1,d^{(1)})}_k \\ \vdots \\ \vec{s}''^{(M,d^{(M)})}_k \\ \vec{z}''_k \end{bmatrix} \qquad \vec{c}' = \begin{bmatrix} \vec{c} \\ \vdots \\ \vec{c} \\ \vdots \\ \vec{c} \\ \vec{0} \\ \vdots \\ \vec{0} \\ \vdots \\ \vec{0} \\ \vec{0} \end{bmatrix}$$

$$\vec{\theta}_k = \begin{bmatrix} \vec{\theta}_k^{(1,0)} \\ \vdots \\ \vec{\theta}_k^{(1,d^{(1)})} \\ \vdots \\ \vec{\theta}_k^{(M,d^{(M)})} \\ \vec{\gamma}_k^{(1)} \\ \vdots \\ \vec{\gamma}_k^{(M)} \\ \vec{\lambda}_k \end{bmatrix} \qquad \vec{b} = \begin{bmatrix} \vec{s}_{in}^{(1,0)} \\ \vdots \\ \vec{s}_{in}^{(1,d^{(1)})} \\ \vdots \\ \vec{s}_{in}^{(M,d^{(M)})} \\ \vec{s}_{out}^{(1)} \\ \vdots \\ \vec{s}_{out}^{(M)} \\ \vec{\mu} \end{bmatrix}$$

where I is identity matrix. The sizes of new vectors and matrices are:

$$\begin{aligned}
\vec{x}_k, \vec{x}''_k, \vec{c}' &: [N \sum_{m\in\mathcal{M}} d^{(m)} + MN + N \times 1] \\
\vec{\theta}_k, \vec{b} &: [L \sum_{m\in\mathcal{M}} d^{(m)} + N \sum_{m\in\mathcal{M}} d^{(m)} + N \times 1] \\
\tilde{A} &: [L \sum_{m\in\mathcal{M}} d^{(m)} + N \sum_{m\in\mathcal{M}} d^{(m)} + N \times N \sum_{m\in\mathcal{M}} d^{(m)} + MN + N] \\
\tilde{B}^m &: [Nd^{(m)} \times Ld^{(m)}] \\
\tilde{B} &: [N \sum_{m\in\mathcal{M}} d^{(m)} \times L \sum_{m\in\mathcal{M}} d^{(m)}] \\
\tilde{S}^m &: [N \times Nd^{(m)}] \\
\tilde{S} &: [NM \times N \sum_{m\in\mathcal{M}} d^{(m)}] \\
\tilde{D} &: [N \times L \sum_{m\in\mathcal{M}} d^{(m)}] \\
\tilde{I} &: [N \sum_{m\in\mathcal{M}} d^{(m)} \times N \sum_{m\in\mathcal{M}} d^{(m)}]
\end{aligned}$$

The Lagrangian function of problem (4.4.1) is:

$$L_k = \vec{c}'^T \vec{x}_k + \varepsilon(\vec{x}_k - \vec{x}''_k)^T(\vec{x}_k - \vec{x}''_k) + \vec{\theta}_k^T(\tilde{A}\vec{x}_k - \vec{b}) \tag{4.4.2}$$

and the algorithm equations from Table 4.2.1 can be expressed as:

$$\begin{aligned}
\vec{x}_k &= \left[\vec{x}''_k - \tfrac{1}{2\varepsilon}\left(\vec{c} + \tilde{A}^T\vec{\theta}_k\right)\right]^+ \\
\vec{\theta}_{k+1} &= \vec{\theta}_k + \alpha\nabla_{\vec{\theta}} L_k \\
\vec{x}''_{k+1} &= \vec{x}''_k - \alpha\nabla_{\vec{x}''} L_k
\end{aligned} \tag{4.4.3}$$

Equations (4.4.3) are the same as (3.4.14). Using the same procedure as in the proof of the OLDRAo algorithm from Section 3.4.3 we prove the convergence of the real-time routing from Table 4.2.1.

4.5 Summary

In this chapter we have presented a distributed algorithm for the real-time energy optimal routing in sensor networks for systems with linear cost functions and constant communication delays. Such networks are very often in the industrial environment, where TDMA derived mechanisms or mechanism with significant stack computation delay are used. We have used the centralized description of the real-time routing problem based on the network replication of the multi-commodity network flow optimization problem, which has been described in Chapter 2. Thanks to the fact, that the real-time model stay in the form of multi-commodity network flow optimization problem with side constraints and that the side constraints are in the form, which allow the problem distribution, we use the distribution approach presented in Chapter 3. The final algorithm computes the real-time routing using only peer-to-peer communication between the neighboring nodes, even for the problems with linear objective functions. In contrast to the other works this algorithm uses constant communication delay, independent on the routed flow volume, which allows to use this algorithm in industrial networks with TDMA like mechanism. The other works in this area use the queuing delay, which is strictly convex function of the routed flow volume and it is not well suitable for problems with hard real-time constraints. Several experiments to evaluate the algorithm behavior and proof of the algorithm convergence have been presented.

In this chapter we have focused on the distribution of the real-time routing problem for the continuous data streams, which has been described in Section 2.3. Please notice, that the same approach can be used for the real-time routing problem with transmission period bigger than one hop communication delay described in Section 2.4.

Chapter 5

Conclusions

In this thesis we have focused on the in-network distributed and real-time routing problems in the multi-hop sensor networks. The work is divided into three parts. The first part deals with a centralized algorithm for the real-time routing, which fulfills the work objective 1. The second part deals with a general distributed algorithm for the non-real-time routing problems described by multi-commodity network flow model, which fulfills the work objective 2. The third part joints the works from the previous two parts and introduces a distributed algorithm for real-time routing in the sensor networks, which fulfills the work objective 3. According to the work objective 4, all presented algorithms have been evaluated on benchmarks for energy optimal routing, using Matlab. Further in this section, we describe the contributions of each part in more details and we mention open issues and future work.

5.1 Summary and Contributions

In the first part of this thesis, we have introduced centralized algorithm for the real-time routing in the sensor networks for systems with linear cost functions and constant communication delays. Such networks are very often in the industrial environment, where TDMA derived mechanisms or mechanism with significant stack computation delay are used. The algorithm is based on the minimum-cost multi-commodity network flow model described as a Linear Programming problem. We have used the network replication to model the constant communication delay in the network. The derived

real-time model stays in the form of multi-commodity network flow problem with side constraints. The structure of the side constraints allow us to derive the distributed algorithm in Chapter 4. Solved in centralized way by Linear Programming, it exhibits a very good performance, as was shown in our experiments. Surprisingly, the performance does not degrade even in the presence of an integral flow constraint, which makes the problem NP hard. It follows that the model is very powerful from the practical point of view and can be used in many applications where the response time is the subject of constraints.

In the second part of this thesis, we have developed three in-network distributed routing algorithms, which are based on the dual decomposition of minimum-cost multi-commodity network flow problem. The algorithms derivations use the Linear Programming model in node-link form, which leads to unique peer-to-peer distributed algorithms. Only the communication between the neighboring nodes is needed during the computation. Moreover the algorithms compute the energy optimal routing for problems with linear objective functions. All other distributed routing algorithms based on the dual decomposition focus only on the problems with strictly convex objective functions and fail in the case of the linear objective functions. According to the fact, that the presented algorithms are based on the general minimum-cost multi-commodity network flow problem, it can be easily adapted for many other problems in the sensor network area, like resource sharing, network localization, object tracking, etc.

In the third part of this thesis, we have used the results of the previous two parts and developed an in-network distributed real-time routing algorithm for systems with linear cost functions and constant communication delays. As mentioned above the centralized real-time routing problem is described as minimum-cost multi-commodity network flow, which allow us to use the general distributed approach form Chapter 3. The resulting algorithm finds the energy optimal routing with real-time constraints even for the problems with linear objective functions, using only the peer-to-peer communication between the neighboring nodes. In contrast to the other works, this algorithm uses constant communication delay independent on the routed flow volume. It allow using this algorithm e.g. in industrial networks with TDMA like mechanism. The other works in this area use the queuing delay, which is strictly convex function of the routed flow volume and it is not well suitable

for problems with hard real-time constraints.

The contributions of this thesis are summarized in Section 1.3. The main contributions of this work are:

1. Formulation of a real-time multi-commodity network flow problem and its solution by Linear Programming based on graph replication.

2. Discovery of a surprisingly good Integer Linear Programming performance for the above mentioned problem with an integral data flow constraint, which makes the problem NP hard.

3. Introduction of a new distributed algorithm based on dual decomposition of routing problem formulated in node-link form.

4. Presentation of novel approach to distribute the linear optimization problem by dual decomposition as an in-network distributed algorithm. (Other works using the dual decomposition on the routing problems are limited to strictly convex objective functions and fail in the linear case.)

5. Introduction of new mathematically derived, distributed algorithm for energy optimal real-time routing based on network replication and dual decomposition.

6. Performance evaluation of all presented algorithms on benchmarks for energy optimal routing in sensor networks.

5.2 Open Issues and Future Work

In Chapter 2, we have presented a good performance of Integer Linear Programming for the centralized real-time routing problems. The performance did not decrease even for integral flow constraints, which makes the problem NP hard. We have considered networks with up to 100 nodes in our experiments. In future work, it would be interesting to develop a heuristic sub-optimal algorithm for the real-time routing problem and evaluate its performance for significantly bigger networks.

The main drawback of the distributed routing algorithms derived in Chapter 3, is the high number of iterations which are needed to achieve the optimal solution. The high number of iterations leads to high number of

communication, which is not suitable for application in the sensor networks area. We believe that the number of iterations can be significantly decreased by several extensions in the future work. We mention several ideas, which should according to our knowledge and experiments significantly decrease the number of iterations:

1. The dual variables corresponding to the capacity constraints of the problems are in some cases fixed only to one node (e.g. in the case of link or node capacity constraints). This dual variables can be exactly evaluated in each iteration (instead of their incremental update, which is used in this work). This approach would lead to a less general model, where the capacity constraints are not relaxed during the dual decomposition. However, the number of iteration of such algorithm should be significantly decreased.

2. The algorithms use a sub-gradient method with a constant step α. The algorithm convergence could be increase by using a more sophisticated method for the dual variables update. We suggest application of methods based on the Newton's method, like the Regula falsi method (e.g. [Sigl 02]), or quasi-Newton methods. According to our preliminary experiments presented in Section 3.4.4 the Newton's method based heuristics can decrease the number of iterations more than 2.7 times.

3. The computation of a new flow variable x_{k+1} could be extended by a nonlinear function, which is aware of the flow saturation ($[..]^+$) and can compute more efficient values.

The next logical step for the distributed routing algorithm for real-time data flow is to perform several experiments in a sensor networks simulator such as the NS-2 or RTNS [Paga 10]. The RTNS, beside the communication aspects, allows to simulate the computational delays in the nodes. Then we can also experiment with an asynchronous version of the distributed routing algorithm, where the nodes are not synchronized by the communication and each node runs independently on the others.

Other future extension of the presented routing algorithms is to join them with other works in the sensor networks area such as topology management and control. E.g. in [LoBe 09] authors present an adaptive topology management approach for time bounded data, which seems well suitable for this task.

Bibliography

[Ahme 08] A. A. Ahmed and N. Fisal. "A real-time routing protocol with load distribution in wireless sensor networks". *Computer Communications*, Vol. 31, pp. 3190–3203, September 2008.

[Anas 08] M. P. Anastasopoulos, A. D. Panagopoulos, and P. G. Cottis. "A distributed routing protocol for providing QoS in Wireless Mesh Networks operating above 10 GHz". *Wireless Communications and Mobile Computing*, Vol. 8, No. 10, p. 1233–1245, December 2008.

[Bert 04] D. P. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 2004.

[Bert 98] D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[Bert 99] D. P. Bertsekas. *Nonlinear Programming*. Massachusetts Institute of Technology, 1999.

[Boyd 04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[Brav 09] G. Bravos and A. G. Kanatas. "Integrating power control with routing to satisfy energy and delay constraints in sensor networks". *European Transactions on Telecommunications*, Vol. 20, No. 2, pp. 233–245, 2009.

[Bulb 07] K. Bulbul and O. Ercetin. "Maximum precision-lifetime curve for joint sensor selection and data routing in sensor networks".

European Transactions on Telecommunications, Vol. 18, No. 7, p. 815, 2007.

[Cacc 03] M. Caccamo and L. Zhang. "The capacity of implicit EDF in wireless sensor networks". *Proceedings of 15th Euromicro Conference on Real-Time Systems, 2003.*, pp. 267– 275, July 2003.

[Chen 06] W. Cheng, L. Yuan, Z. Yang, and X. Du. "A Real-time Routing Protocol with Constrained Equivalent Delay in Sensor Networks". In: *11th IEEE Symposium on Computers and Communications (ISCC'06)*, pp. 597–602, 2006.

[Chen 10] D. Chen, M. Nixon, and A. Mok. *WirelessHART(TM): Real-Time Mesh Network for Industrial Automation.* Springer, 2010.

[Chia 05] M. Chiang. *Geometric programming for communication systems.* Vol. 2, Foundations and Trends in Communications and Information Theory, August 2005.

[Chia 07] M. Chiang, S. Low, A. Calderbank, and J. Doyle. "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures.". In: *Proceedings of the IEEE.*, pp. 255–312, January 2007.

[Chip 06] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. A. Stankovic, and T. F. Abdelzaher. "Real-time Power Aware Routing in Wireless Sensor Networks". *The 14th IEEE International Workshop on Quality of Service*, June 2006.

[Edga 03] J. Edgar, H. Callaway. *Wireless Sensor Networks: Architectures and Protocols.* Auerbach Publications, 2003.

[Gare 79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman , Oxford, UK, 1979.

[He 03] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. "SPEED: a stateless protocol for real-time communication in sensor networks". *International Conference on Distributed Computing Systems (ICDCS 2003)*, pp. 46– 55, May 2003.

[Joha 05] B. Johansson and M. Johansson. “Primal and Dual Approaches to Distributed Cross-layer Optimization.”. In: *16th IFAC World Congress*, Prague, Czech Republic, 2005.

[Joha 06a] B. Johansson, P. Soldati, and M. Johansson. “Mathematical decomposition techniques for distributed cross-layer optimization of data networks.”. *IEEE Journal on Selected Areas in Communications*, Vol. 24, pp. 1535 – 1547, August 2006.

[Joha 06b] M. Johansson and L. Xiao. “Cross-layer optimization of wireless networks using nonlinear column generation”. *IEEE Transactions on Wireless Communications*, Vol. 5, pp. 435–445, February 2006.

[Joha 08] B. Johansson, C. Carretti, and M. Johansson. “On Distributed Optimization Using Peer-to-Peer Communications in Wireless Sensor Networks”. In: *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, San Francisco, CA, 2008.

[Jurk 08] P. Jurčík, R. Severino, A. Koubâa, M. Alves, and E. Tovar. “Real-Time Communications over Cluster-Tree Sensor Networks with Mobile Sink Behaviour”. *14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Kaohsiung, Taiwan*, 2008.

[Karl 05] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. Wiley, 2005.

[Koub 06a] A. Koubaa, M. Alves, and E. Tovar. “GTS Allocation Analysis in IEEE 802.15.4 for Real-Time Wireless Sensor Networks”. *The 14th International Workshop on Parallel and Distributed Real-Time Systems, Rhodes, Greece*, 2006.

[Koub 06b] A. Koubaa, M. Alves, and E. Tovar. “i-GAME: An Implicit GTS Allocation Mechanism in IEEE 802.15.4”. *The 18th Euromicro Conference on Real-Time Systems (ECRTS’06), Dresden, Germany*, July 2006.

[Lfbe 04] J. Löfberg. "YALMIP : A Toolbox for Modeling and Optimization in MATLAB". In: *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[Li 07] P. Li, Y. Gu, and B. Zhao. "A Global-Energy-Balancing Real-Time Routing in Wireless Sensor Networks". In: *Proceedings of the The 2nd IEEE Asia-Pacific Service Computing Conference*, pp. 89–93, 2007.

[LoBe 09] L. LoBello and E. Toscano. "An Adaptive Approach to Topology Management in Large and Dense Real-Time Wireless Sensor Networks". *IEEE Transactions on Industrial Informatics*, Vol. 5, No. 3, pp. 314–324, August 2009.

[Low 99] S. Low and D. Lapsley. "Optimization flow control. I. Basic algorithm and convergence.". *IEEE/ACM Transactions on Networking (TON)*, Vol. 7, pp. 861 – 874, December 1999.

[Nama 06] H. Nama, M. Chiang, and N. Mandayam. "Utility-Lifetime Trade-off in Self-regulating Wireless Sensor Networks: A Cross-Layer Design Approach". *ICC '06. IEEE International Conference on Communications*, Vol. 8, pp. 3511–3516, June 2006.

[Ouor 00] A. Ouorou, P. Mahey, and P. Vial. "A Survey of Algorithms for Convex Multicommodity Flow Problems". *Management Science*, Vol. 46, No. 1, pp. 126–147, January 2000.

[Paga 10] P. Pagano, M. Chitnis, G. Lipari, C. Nastasi, and Y. Liang. "Simulating Real-Time Aspects ofWireless Sensor Networks". *EURASIP Journal on Wireless Communications and Networking*, 2010. Article ID 107946.

[Palo 06a] D. Palomar and M. Chiang. "Alternative Decompositions for Distributed Maximization of Network Utility: Framework and Applications.". In: *25th IEEE International Conference on Computer Communications*, pp. 1 – 13, 2006.

[Palo 06b] D. Palomar and M. Chiang. "A tutorial on decomposition methods for network utility maximization". *IEEE Journal on Selected Areas in Communications*, Vol. 24, pp. 1439 – 1451, August 2006.

[Piro 04] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann, 2004.

[Rese 06] M. G. C. Resende and P. M. Pardalos. *Handbook of Optimization in Telecommunications.* Springer, 2006.

[Saad 03] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, 2003.

[Sigl 02] L. Sigler. *Fibonacci's Liber Abaci.* Springer-Verlag, 2002.

[Such 07] P. Šůcha and Z. Hanzálek. "Deadline constrained cyclic scheduling on pipelined dedicated processors considering multiprocessor tasks and changeover times". *Mathematical and Computer Modelling*, Vol. 47, No. 9–10, pp. 925–942, May 2007. Springer.

[Trdl 07] J. Trdlička. "Real-Time Network Flow Routing". Tech. Rep. 0.2, Department of Control Engineering, Faculty of Electrical Engineering, CTU Prague, Czech Republic, 2007. http://rtime.felk.cvut.cz/wiki/images/4/46/Real-time_routing_0.2.pdf.

[Trdl 11] J. Trdlička. "Video Presentation for Distributed Routing Algorithms.". Tech. Rep., CTU in Prague, Czech Republic, 2011. http://www.trdlicka.cz/research/distributed-routing.

[Tsit 86] J. Tsitsiklis and D. Bertsekas. "Distributed asynchronous optimal routing in data networks.". *IEEE Transactions on Automatic Control*, Vol. 31, pp. 325 – 332, April 1986.

[Watt 05] T. Watteyne and I. Auge-Blum. "Proposition of a hard real-time MAC protocol for wireless sensor networks". *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium*, pp. 533–536, September 2005.

[Xian 10] K. Xiang and Y. Zeng. "A distributed cross-layer real-time routing in wireless sensor networks". In: *2nd International Conference on Signal Processing Systems (ICSPS)*, Dalian, 2010.

[Xiao 04] L. Xiao, M. Johansson, and S. Boyd. "Simultaneous routing and resource allocation via dual decomposition". *IEEE Transactions on Communications*, Vol. 52, No. 7, pp. 1136–1144, July 2004.

[Zhan 08] L. Zhang, M. Hauswirth, L. Shu, Z. Zhou, V. Reynolds, and G. Han. "Multi-priority Multi-path Selection for Video Streaming in Wireless Multimedia Sensor Networks". In: *5th International Conference Ubiquitous Intelligence and Computing*, pp. 439–452, Oslo, Norway, June 2008.

[Zhen 10] M. Zheng, W. Liang, H. Yu, and Y. Xiao. "Cross layer optimization for energy-constrained wireless sensor networks: Joint rate control and routing". *Computer Journal*, Vol. 53, No. 10, pp. 1632–1642, December 2010.

List of Author's Publications

Journal Papers

J. Trdlicka and Z. Hanzálek. "Distributed Multi-Commodity Network Flow Algorithm for Energy Optimal Routing in Wireless Sensor Networks". *Radioengineering*, Vol. 19, December 2010.

International Conference Papers

J. Trdlicka and Z. Hanzálek. "Distributed Algorithm for Energy Optimal Multi-Commodity Network Flow Routing in Sensor Networks". In: *IEEE International Conference on Wireless Communications and Signal Processing (WCSP 2010)*, Suzhou, 2010.

J. Trdlicka and Z. Hanzálek. "Algorithm for Energy Optimal Routing of Periodic Messages with Real-Time Constraints in Sensor Networks". In: *30th IFAC Workshop on Real-Time Programming and 4th International Workshop on Real-Time Software*, Mragowo, 2009.

I. Sigh, J. Trdlicka, and Z. Hanzálek. "ITEM - Implementation of Integrated TDMA and E-ASAP Module". In: *Proceedings Work-In-Progress Session of 20th Euromicro Conference on Real-Time Systems (ECRTS 08)*, pp. 64–67, Prague, 2008.

J. Trdlicka, M. Johansson, and Z. Hanzálek. "Optimal Flow Routing in Multi-hop Sensor Networks with Real-Time Constraints through Linear Programming". In: *12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 07)*, pp. 439–452, Patras, 2007.

J. Trdlicka. "Real-Time Flow Routing in Sensor Networks". In: *11th International Student Conference on Electrical Engineering, Poster 2007*, Prague, 2007. **Best Paper Award**.

Papers Submitted to International Journals

J. Trdlicka and Z. Hanzálek. "In-Network Distributed Algorithm for Energy Optimal Routing based on Dual Decomposition of Linear Programming". *IEEE Transactions on Communications*. **Major Revision**.

J. Trdlicka and Z. Hanzálek. "Distributed Algorithm for Real-Time Energy Optimal Routing based on Dual Decomposition of Linear Programming". In: *International Journal of Distributed Sensor Networks*. Submitted to.

Other Publications

J. Trdlicka and Z. Hanzálek. "Proof of algorithm convergence for Distributed Real-Time routing". Tech. Rep., CTU in Prague, Czech Republic, 2011. http://dce.felk.cvut.cz/hanzalek/DistRouting/ConProof2.pdf.

J. Trdlicka and Z. Hanzálek. "Proof of the algorithm convergence". Tech. Rep., Department of Control Engineering, Faculty of Electrical Engineering, CTU Prague, Czech Republic, 2010.

J. Trdlicka. "Distribuovaný algoritmus pro smerování toku dat v bezdrátových senzorových sítích". In: *Kolokvium vestavených systému*, Praha, 2009.

M. Sojka, M. Molnár, J. Trdlička, P. Jurčík, P. Smolík, and Z. Hanzálek. "Wireless networks - documented protocols, demonstration". Tech. Rep., Department of Control Engineering, Faculty of Electrical Engineering, CTU Prague, Czech Republic, 2008. Deliverable: D-ND3v2, FRESCOR.

J. Trdlicka and Z. Hanzálek. "Real-Time Data Flow Routing in Multi-Hop Sensor Networks". In: *Proceedings of Workshop 2008*, Prague, 2008.

J. Trdlicka. "Real-Time smerování toku dat v senzorových sítích". In: *Kolokvium vestavených systému*, Praha, 2007.

J. Trdlicka. "Optimal Data Flow Routing in Multi-hop Sensor Networks". Tech. Rep., CTU in Prague, Czech Republic, 2007.

J. Trdlicka. "Real-Time Network Flow Routing". Tech. Rep. 0.2, Department of Control Engineering, Faculty of Electrical Engineering, CTU Prague, Czech Republic, 2007. http://rtime.felk.cvut.cz/wiki/images/4/46/Real-time_routing_0.2.pdf.

The main goal of this thesis is to bring a new knowledge into the area of in-network distributed and real-time routing algorithms for multi-hop sensor networks. The work aims mathematically derive algorithms which are based on the convex optimization theory and which are capable of computing an exact optimal solution e.g. in terms of the energy consumption.

The main contributions of this work are:

Formulation of a real-time multi-commodity network flow problem and its solution by Linear Programming based on graph replication.

Discovery of a surprisingly good Integer Linear Programming performance for the above mentioned problem with an integral data flow constraint, which makes the problem NP hard.

Introduction of a new distributed algorithm based on dual decomposition of routing problem formulated in node-link form.

Presentation of novel approach to distribute the linear optimization problem by dual decomposition as an in-network distributed algorithm. (Other works using the dual decomposition on the routing problems are limited to strictly convex objective functions and fail in the linear case.)

Introduction of new mathematically derived, distributed algorithm for energy optimal real-time routing based on network replication and dual decomposition.

Performance evaluation of all presented algorithm on benchmarks for energy optimal routing in sensor networks.