Managing Large Distributed Networks – An Algorithmic Perspective

Habilitation Thesis

Tamás Lukovszki

Eötvös Loránd University
H-1117 Budapest, Hungary

Contents

1 Introduction **3**

2 Wireless Ad Hoc Networks **4**
2.1 Topology Control and Routing 6
2.1.1 Resource efficient maintenance of wireless network topologies 6
2.1.2 Worst case mobility in ad hoc networks 10

3 Mobile Sensor Networks, Autonomos Robots **13**
3.1 Localized sensor self-deployment for focused coverage 16
3.1.1 Focused coverage (F-coverage) 16
3.1.2 The equiliteral triangle tessellation 17
3.1.3 Problem 17
3.1.4 Our contribution 17
3.2 Pattern formation for anonymous, position-aware robots 18
3.2.1 Our contribution 18
3.3 Uniform Dispersal of Myopic Robots 19
3.3.1 Our contribution 20

4 Large Scale Content Distribution **22**
4.1 Data compression and decompression 23
4.1.1 Space efficient Burrows-Wheeler backtransformation 23
4.2 Content distribution in P2P networks 27
4.2.1 Topology management and data distribution in tracker based networks 27
4.2.2 Network coding for robust data distribution in BitTorrent-like networks 28
4.2.3 Resource constrained BitTorrent-like networks 29

5 Network Function Virtualisation, Software Defined Networks **30**
5.1 On-line admission control and embedding of service chains 30
5.2 Middlebox Deployment 32

1 Introduction

We consider algorithmic aspects of managing large scale distributed networks, such as wireless ad hoc networks, sensor networks, peer-to-peer (P2P) networks, and we also address network function deployment and service chain embedding problems arising in software defined networking (SDN) and network function virtualization (NFV). The main goal of this work is to solve practical problems and develop exact theory which is related but not restricted to the solution of the practical problem.

Wireless ad hoc networks are formed by mobile terminals which use a wireless communication interface and establish communication between nodes using other terminals as intermediate relay stations. Control and databases are distributed over the network; a fixed infrastructure is not needed. Thus, they can be used in emergency situations when infrastructure is not available or (partly) destroyed, or the nodes are highly dynamic. In order to effectively operate such networks it is essential to use efficient localized strategies for topology management and for routing. A localized strategy means only to use information about nearby nodes within the communication radius and guarantee global properties of the network topology. The goal is to construct and maintain networks guaranteeing short paths, low energy paths between the communicating nodes, low congestion path systems, and support for efficient routing strategies without using global knowledge about the network. Fault tolerance, self-(re)construction, and dealing with mobility are of key importance.

Mobile sensor networks (MSN) consist of nodes, where each node has sensing, computation, and locomotion capabilities. By assuming a large scale sensor network with unpredictable sensor failure, limited sensing and communication ranges, decentralized and localized sensor self-deployment methods are more beneficial and scale invariant than centralized solutions. Each sensor/robot is autonomus and makes independent decisions using neighborhood information only. Self-deployment and pattern formation problems are fundamental problems in mobile sensor networks and autonomous robots. The sensors must cover a dedicated area, a border, an area around a so called Point of Interest (POI), or form a given connected pattern.

The simplicity of creating, distributing and communicating content causing an enormous load on the Internet. Peer-to-peer (P2P) systems provide a very popular and efficient way of content distribution. They generate a significant part of the overall Internet traffic. In tracker-based P2P networks, like BitTorrent (BT), the effectiveness of downloading data largely depends on the neighbors of the client and on the distribution and selection of the pieces of the file. We model those systems, derive upper and lower bouns on the distribution time, and develop efficient methods for the data distribution.

Data compression and decompression are essential tools to decrease bandwith demand and increase throughput during data transmission. It is of high interest

for distributing large or real-time data. A particular focus of our research is data (de-)compression on resource constrained devices, such sensor nodes or embedded devices. The Burrows-Wheeler transformation is used for effective data compression, e.g., in the well known program bzip2. We present algorithms that reduce the memory need without sacrificing speed too much.

Modern computer networks provide a rich set of in-network functions, including access control, firewall, intrusion detection, network address translation, traffic shaping and optimization, caching, among many more. While such functionality is traditionally implemented in hardware middleboxes, computer networks become more and more virtualized: Network Function Virtualization (NFV) enables a flexible instantiation of network functions on network nodes, e.g., running in a virtual machine on a commodity x86 server. Modern computer networks also offer new flexibilities in terms of how traffic can be routed through such network functions. In particular, using Software-Defined Networking (SDN) technology, networks also enables the definition and fast deployment of novel network services called service chains: sequences of virtualized network functions (e.g., firewalls, caches, traffic optimizers) Such advanced network services open an interesting new market for Internet Service Providers, which can become "miniature cloud providers", specialized for in-network processing. We present solutions for incremental middlebox deployment and online servive chain embedding.

Outline: In Section 2 we overview our results on wireless ad hoc networks. In Section 3 we present our work on mobile sensor networks and autonomous robots. Section 4 summarize our achievments in large scale peer-to-peer content distribution. In Section 5 we present our results in the area of NFV and SDN.

2 Wireless Ad Hoc Networks

Multihop wireless ad hoc networks received considerable attention recently due to their potential wide applications in various areas and particularly the ubiquitous computing and sensor networks as they do not require any fixed infrastructure. Such networks are self organizing and self administrating.

Wireless ad hoc networks need special treatment because of their own characteristics and some limitations compared with wired networks. For example, such networks are often powered by batteries and they often have limited storage capacity. A transmission of a wireless device is often received by many nodes within its transmission range, which possibly causes signal interferences at these nodes. One of the most important properties of wireless ad hoc networks is that the topology may change suddenly and frequently. Due to the above intrinsic properties, it is more challenging to design a network topology for wireless ad hoc networks, which is suitable for an efficient routing scheme, saves energy and storage at the nodes, and keeps the interferences low, than in traditional (wired)

networks. As overviews on algorithmic aspects of ad hoc networks we refer to [119, 92, 65].

Applications: Wireless ad hoc networks can be applied as independent communication networks in an area where no infrastructure is available and emergency operations are taking place, for instance, due to a natural disaster.

- **Emergency services:** In an area, where the infrastructure is damaged by a disaster, emergency troups can use an ad hoc network during their operations. Beside a communication platform this ad hoc network may also provide other services. As the importance of the Internet grows continuously, the loss of connectivity during such a natural disaster may cause more noticable problems in the affected area. Furthermore, network applications become increasingly important also for emergency services, and thus it will be necessary to reach these applications even if the infrastrure elments are destroyed.

- **Extension for cellular networks:** In the area of telecommunication networks ad hoc networks have been also proposed as supplement to existing cellular radio networks, to provide connectivity in an area without cellular coverage, such as a road tunnel, an underground tunnel, an area, where the deployment of base stations is too expensive or not desired, or a region affected by a natural disaster, where emergency or rescue operations take place. Allowing the nodes a multihop connection to the base stations – using intermediate mobile nodes as relay stations – the ubiquity can be attained.

- **Conferencing:** Consider the scenario of a mobile conference, in which mobile computer users meet outside their everyday office space, where the normal network infrastructure is missing. Even in such situations collaborative computing might be important. The goal of such a meeting might be to make progress on a certain project. Thus, creating an ad hoc network is quite helpful. The establishment of an ad hoc network seems to be useful even if a fixed infrastructure is available in a conference, since routing back and forth between widely separated office environments may cause a large overhead.

- **Sensor networks:** Sensor networks are likely to be widely deployed in the future because they essentially improve our ability to monitor and control the physical environment from remote locations and can significantly improve the accuracy of information obtained via collaboration of sensor nodes. Some low cost sensors could be rapidly deployed (e.g. by dropping them from an airplane) in an area. Then they can be used for a surveillance application, for object tracking; for fire detection in a forest or to analyze

the motion of a tornado. Sensors are developed which can be used for inventory control, product quality monitoring, establishing smart office spaces, and for interfaces for disabled. Corresponding to different application areas we have different constraints on the mobility, the energy, computation power and memory storage, and communication resources of the sensors.

- **Inter-vehicle networks:** Sensors can also be attached to taxi cabs or other vehicles to collect traffic information, such as data on braking sent from a preceding car, data on the traffic flow on a route, and data about sites located along a route. This data can be used for avoidance of accidents and for planning routes more efficiently.

- **Rooftop networks:** Self-configuring, densely deployed wireless networks, called rooftop networks, has been proposed in the 90's in metropolitan area as an alternative to traditional infrastructure offered by telecommunication providers. The name refers to an antenna (access point) on each building's roof. Such a network also provides an alternate infrastructure, if the conventional network fails, as after a disaster. A further benefit is that the energy consumption of the nodes can be reduced, since they can use lower transmission power. The nodes of a rooftop network are not mobile, but a self-configuring routing system for hundreds of thousands of nodes, which resists against failure of nodes, represents a great scaling challenge.

2.1 Topology Control and Routing

Wireless ad hoc networks often have limited energy and storage resources. A transmission of a wireless device is received by many nodes within its transmission range, which causes interferences at these nodes. In order to save energy and to keep the interferences at an acceptable level, each node of an ad hoc network can potentially change the network topology by adjusting its transmission range and/or selecting certain nodes to forward messages, i.e. controlling the set of its neighbors. The primary goal of topology control is to maintain network connectivity, optimize network lifetime, interferences, throughput, and to support (power) efficient routing. As excellent surveys on topology control in ad hoc networks we refer to [125, 126, 159, 173, 174].

2.1.1 Resource efficient maintenance of wireless network topologies

Multiple hop routing in mobile ad hoc networks can minimize energy consumption and increase data throughput. Yet, the problem of radio interferences remain. However if the routes are restricted to a basic network based on local neighborhoods, these interferences can be reduced such that standard routing algorithms can be applied. We compare different network topologies for these basic networks, i.e. the Yao-graph (aka. θ-graph) and some also known related

models, which will be called the SymmY-graph (aka. YS-graph), the SparsY-graph (aka. YY-graph) and the BoundY-graph. Further, we present a promising network topology called the Hieralchical Layer graph (HL-graph). We compare these topologies regarding degree, spanner-properties, and communication features. We investigate how these network topologies bound the number of (uni- and bidirectional) interferences and whether these basic networks provide energy-optimal or congestion-minimal routing. Then, we compare the ability of these topologies to handle dynamic changes of the network when radio stations appear and disappear. For this we measure the number of involved radio stations and present distributed algorithms for repairing the network structure.

Model and quality measures: We assume that the nodes V of the network are given in the two dimensional Euclidean plane $\mathbb{R}^2$, and each node has the same maximum transmission range r. The wireless nodes define the so-called *unit disk graph* $U(V)$, in which two nodes $u, v \in V$ are connected by an edge if and only if they are within the maximum transmission range of each other. The nodes V of the network are given as points in $\mathbb{R}^2$. By scaling, we assume that the maximum transmission range r is equal to the unit. The edge set of the unit disk graph is $E = \{\{u, v\} : ||u, v||_2 \leq 1\}$, where $||u, v||_2$ denotes the Euclidean distance between u and v.

In network design is is often useful to approximate the unit disk graph by certain subgraphs. Obviously, a necessary condition for the connectivity of the ad hoc network is the connectivity of the unit disk graph and of its approximation. Of course, not every connected subgraph of the unit disk graph plays the same important role in the network design. In the following we describe several quality measures for wireless network topologies.

Sparse topologies, low node degree: Maintaining a sparse topology and a low node degree is an important optimization aspect in mobile wireless networks. Dense topologies or a high node degree result in high per node storage requirement, time-consuming updates if the topology changes frequently, and expensive packet forwarding schedules.

Short paths – spanner graphs: A very important requirement of topology control is to construct a subgraph of $U(V)$, such that for each pair of nodes, the shortest path in the subgraph is not much longer than the shortest path in the unit disk graph. This property is captured by the **stretch factor** of the subgraph G, which is defined as the maximum ratio between the legth of the shortest path G and the length of the shortest path in $U(V)$ over all pairs of nodes. A graph with stretch factor $t \geq 1$ is called a **t-spanner**. More formally, let $G = (V, E)$ be a subgraph of $U = U(V)$. The length of a path P from a node u to node v in a graph G is the sum of the Euclidean lengths of edges in P. Let $d_G(u, v)$ and $d_U(u, v)$ be the length of the shortest path from u to v in G and U respectively. The stretch factor t of a subgraph $G \subseteq U$ is defined as

$t = \max\{\frac{d_G(u,v)}{d_U(u,v)} : u, v \in V\}$. A subgraph with a constant stretch factor t is called a *t-spanner*. A t-spanner is called a *sparse t-spanner* if the number of links is linear in $|V|$.

Paths with low energy consumption – power spanners: Energy consumption is a critical issue in ad hoc networks. Using variable transmission power reduces the energy consumption, and thus, increases the life time of the network on the one side; and it decreases the interferences, and thus, increases the throughput on the other side. Li et al. [123] introduced the notion of ***t*-power spanners** to capture the property of a graph G that between each pair of nodes it contains a path, which can be established using at most t times more energy as the minimum energy path between that pair of nodes in $U(V)$. More formally, this is the following. Based on a power-attenuation model, where the power $pow(e)$ needed to support a link $e = (u, v)$ is $||u, v||_2^\beta$, where $\beta \geq 2$ (for the free space propagation model $\beta = 2$, for the so-called two ray model, which also considers multipath fading, $\beta = 4$), Li et al. [123] defined the *power stretch factor* of a subgraph $G = (V, E) \subseteq U(V)$ as follows. Let P be a path from u to v in G. Let $pow(P) = \sum_{e \in P} pow(e)$. The distance $d_{G,\beta}(u, v)$ between a pair of nodes $u, v \in V$ is then defined as $\min\{pow(P) : P$ is a path between u and v in $G\}$. The graph G is called a (t, β)*-power spanner*, if for each pair of nodes $u, v \in V$, $d_{G,\beta}(u, v) \leq t \cdot d_{U,\beta}(u, v)$ for a constant t. The constant t is called the *power stretch factor* of G.

We proved in [22, 40] a general relation between Euclidean spanners and power spanners. We have shown that each t-spanner is a (t^β, β) power spanner.

Low congestion path system – weak spanners: A further useful geometric property of graphs is the **weak spanner property** [19, 22, 40], which is closely related to the congestion. This means that using a weak spanner as underlying topology allows low congestion routing in the wireless network. A graph $G = (V, E)$ is called a weak t-spanner if for each pair of nodes $u, v \in V$ there is a path from u to v such that for each node w on the path $||u, w||_2 \leq ||u, v||_2$ holds. In [141] it was shown that this property is closely related to the congestion of paths systems in the network.

Yao graphs: Popular topologies for wireless ad hoc networks is the class of Yao graphs [178]. These graphs are not necessarily planar but they are very simple to construct in a distributed manner and they have a couple of other nice properties. The **Yao graph** [178] with an angle $\theta > 0$ is defined as follows. We divide the plane around each point $u \in V$ into cones (called sectors), each with an angle at most θ at the apex u.

For each sector we join u with its closest neighbor – if any – by an edge. Formally, the Yao graph (aka. Θ-graph) $Yao_\theta(V)$ is defined by the following set of directed edges: $E := \{(u, v) \mid \forall w \neq u \ : \ sector(u, v) = sector(u, w) \ \Rightarrow ||u, v||_2 \leq ||u, w||_2\}$, where $sector(u, v)$ denotes the sector around u containing v

In [120, 158] it is shown that the Yao graph is a t-spanner for V with $t = \frac{1}{1-2\sin(\theta/2)}$ if $\theta < \pi/3$. The proofs in [120, 158] also construct a t-spanner path which immediately gives us a so-called position based routing strategy. Li et al. [123] proved that $Yao(V)$ is a (t, β) power spanner with $t = \frac{1}{1-(2\sin(\theta/2))^{\beta}}$.

Yao type graphs with bounded degree:

- The *Symmetric Yao graph* [124] contains an edge (u, v) if and only if both (u, v) and (v, u) are contained in the Yao graph.
- The *Sparsified Yao graph* contains an edge (u, v) if and only if (u, v) is contained in the Yao graph and it is the shortest incoming edge of v in a sector.
- The *Bounded Degree Yao graph* results form substituting the set of incoming edges in a sector by certain tree according to a a general transformation by Arya et al. [50].

We proved in [22, 40] that the stretch factor of $SymmY_{\theta}(V)$ can be $\Omega(n)$ in the worst-case hence it is not a t-spanner. Table 1 (see [40]) summarizes the properties of the Yao type topologies.

Topology	Degree	Spanner	Energy approx. factor	Congestion approx. factor
Yao	$O(n)$	yes	$O(1)$	—
SymmY	$O(1)$	no, but connected	—	—
SparsY	$O(1)$	weak and power	$O(1)$	$O(\log n)$
BoundY	$O(1)$	yes	$O(1)$	—

Table 1: Basic properties of the Yao type topologies

Entering or leaving of nodes: If a node enters of leaves the network, a certain set of other nodes must update their neighborhood. Since the maximum degree D of a node in the Yao graph is $n - 1$ in the worst case, the number of affected nodes is $\Omega(n)$ if this node enters or leaves. We defined the edge set of the SymmY, SparsY, and BoundY graphs with help of the edge set of the Yao graph. If we store each of the incoming Yao-edges to compute the edges of the SymmY, SparsY, or BoundY graph, then the number of involved nodes can not be improved in the worst case. In [40] we have shown how to update the edges SymmY, SparsY, and BoundY graphs in $O(D)$ time.

2.1.2 Worst case mobility in ad hoc networks

We consider two models to find a reasonable restriction on the worst-case mobility. In the pedestrian model we assume a maximum speed v_{max} of the radio stations, while in the vehicular model we assume a maximum acceleration a_{max} of the points. Our goal is to maintain persistent routes with nice communication network properties like hop-distance, energy-consumption, congestion and number of interferences. A route is persistent, if we can guarantee that all edges of this route can be uphold for a given time span Δ, which is a parameter denoting the minimum time the mobile networks needs to adopt changes, i.e. update routing tables, change directory entrees, etc... We present distributed algorithms based on a grid clustering technique and a high-dimensional representation of the dynamical start situation. We measure the optimality of the output of our algorithm by comparing it with the optimal choice of persistent routes under the same circumstances with respect to pedestrian or vehicular worst-case movements.

For the first model we assume a large number of pedestrians using mobile wireless communication devices in a rather small area. The maximum speed is bounded by a small constant. The standard approach in a static ad hoc network scenario is to build up connections between nearest neighbors. If the mobility is very high, like on a crowded sidewalk, this leads to short communication links, that survive for only short time periods. Although it is possible to build up these connections and transmit some data, it is nearly impossible to maintain packet routes or maintain directories for efficient location of users. Therefore, we need communication links to sustain for some time span Δ to enable the routing layer to keep up with the dynamical changes. We can guarantee that a communication link between two moving stations sustains for this period if we adjust the transmission range to a value, which covers all possible distances the communication partners can reach in time Δ. Since, we know the maximum speed, this implies that the transmission power must be chosen such that the transmission range is at least $2\Delta v_{max}$ larger than the distance at the beginning of the time interval. The task is now to appropriately build up the basic connection links such that the routing algorithm can choose routes with low energy or low congestion, while the number of edges and interfering edges is small.

A motivating example for the acceleration bounded model is given by vehicles of high speed, like cars, trains, or aircrafts. E.g., consider trains where each wagon carries a mobile radio station. Now consider a scenario, where two such trains pass each other in opposite directions. If we take a snapshot in this moment and build a static ad hoc network using the temporary positions, then this static approach may lead to many links joining nodes in the different trains. But such connection links can be upheld only for a short time period since the trains move with high speed. Therefore this static network design is not a good choice.

Previous research: Many mobility models have been proposed as a basis for simulation of cellular and ad hoc networks. Most of them use a random process to vary speed or direction of the moving objects, like the *random walk* model and its variants, that describe mobility as a stop-and-go motion between cells. In the *fluid flow* model every object moves with a randomly chosen speed and direction for a predefined time interval. In the *Gauss-Markov* model [130] speed and direction are changed with an adjustable amount of randomness, ranging from completely random to predictable, linear motion. In the *random waypoint* model [117] the objects move between randomly chosen positions where they pause for a certain time interval. Their speed is uniformly distributed between zero and a maximum. The speed chosen for the next motion period does not depend on the speed of the previous period. Thus sharp turns and sudden stops are possible, i.e. the acceleration is not bounded. There are mobility models that regard mobility of a group of objects, e.g. the *reference point group mobility* model [112] that defines for groups of objects a logical center that determines direction, speed, and acceleration of each object. A survey of the mobility models mentioned above can be found in [76, 68].

Another way to deal with mobility in ad hoc networks is to consider what happens to the underlying topology when the nodes are moving. This leads to the *adversarial network model* [53] in which all communication links are under control of an adversary. A worst case for mobility corresponds with the topological changes the adversary may perform within some predefined restrictions.

In the context of computational geometry Basch et al. introduced the concept of *kinetic data structures* (KDS) [66] that describes a framework for analyzing algorithms on mobile objects. In their model the mobility of objects is described by pseudo-algebraic functions of time and fully or partially predictable. The analysis of a KDS is done by counting the combinatorial changes of the geometric structure that is maintained by the KDS. Another approach that captures unpredictable mobility is the concept of *soft kinetic data structures* (SKDS) [85]. These data structures maintain an approximate geometric structure that is updated by property testing and reorganization. The idea of kinetic data structures is also used in [108] to maintain a clustering of moving objects. This approach is used in [109] to determine the head of each cluster in a mobile network. In each cluster the nodes are directly connected to the head.

Own contributions: Assuming that a fixed time interval of length Δ is given, in [42] we describe how we construct the mobile ad hoc network for a set of stations to solve routing problems. Further, we innovate the pedestrian and the vehicular mobility as two worst-case mobility models. Further, we introduce a measure called crowdedness that states a lower bound on the number of radio interferences. We concentrate on the distributed computation of the network at MAC and physical layer and show the main result that a mobile spanner hosts a

path system that polylogarithmically approximates the optimal congestion. We give techniques how to construct such mobile spanners with small congestion, small interference number, small energy-consumption, and small degree. We present a Hierarchical Grid based on a grid-cluster technique and prove that its interference number can be upper bounded by a logarithmic term if we assume the crowdedness to be logarithmic. One assumption in our approach is that all positioning information is available to all nodes. We present two solutions: the first is based on a positioning system and the second uses distances as location information. This yields to a very dynamic data structure, the so-called Mobile Hierarchical Layer graph that fulfills all our requirements.

Model: we consider a fixed set S of n mobile stations $s_1, \ldots, s_n$ in the Euclidean plane. We denote by $s_i(t)$ the coordinates of a mobile station s_i at time point t and by $s_i'(t) = ds_i(t)/dt$ its speed vector. Furthermore, $s_i''(t) = ds_i'(t)/dt$ denotes the acceleration of s_i at time t, i.e. the change of the speed.

We allow adjustable transmission power for each connection, which is high enough such that all mobile stations never leave the maximum transmission range of a mobile station. The mobile stations use omni-directional radio antennae, i.e. all mobile stations inside a disk with the sender as center and the transmission distance as radius can receive the message or will be disturbed while receiving data on a different connection. We assume bidirectional communication on a single frequency with time-multiplexing, i.e. using different time slots.

We try to keep all connections alive for at least a fixed time interval of length Δ. It should be chosen sufficiently large to set up the communication links between neighbors, to update routing tables, and deliver some amount of data. For the theoretical analysis we assume synchronous round model.

The **pedestrian mobility** model is a worst case approach relying on all mobile stations obeying a speed limit of $v_{\max}$. In this velocity bounded model the starting position $s_i := s_i(0)$ is known and for the speed vector $s_i'(t) = ds_i(t)/dt$ it holds $|s_i'(t)| \leq v_{\max}$. This implies for the relevant time interval Δ that all mobile stations remain in a disk with radius $v_{\max} \cdot \Delta$ around the starting position s_i, i.e. for $t \in [0, \Delta] \ : \ |s_i(t) - s_i| \ \leq \ v_{\max} \ \Delta$.

The **vehicular mobility** model describes the movement of n stations with bounded acceleration $a_{\max}$. Let $s_i''(t) = ds_i'(t)/dt$ denote the acceleration vector of a mobile station s_i, then we assume that for all mobile stations $|s_i''(t)| \leq a_{\max}$. Now, the starting speed vector $s_i' := s_i'(0)$ at the beginning of the time interval Δ can be arbitrarily large. Yet, we assume that at the beginning of the time interval $[0, \Delta]$ we know all locations $s_1, \ldots, s_n$ and all speed vectors $s_1', \ldots, s_n'$. Then, we can estimate the position of station i at time point $t \in [0, \Delta]$ by

$$|s_i(t) - ts_i' - s_i| \ \leq \ \frac{1}{2} a_{\max} t^2 \ \leq \ \frac{1}{2} a_{\max} \Delta^2 \ .$$

As a technical condition we require a polynomial bound on the maximum

distances and relative speed differences for both models, i.e. for some constant k we claim $|s_i - s_j| = O((v_{\max}\Delta)^k)$ in the pedestrian model and $|s_i - s_j| + |s'_i - s'_j| = O((a_{\max}\Delta)^k)$ in the vehicular model.

We introduce a network independent measure, called **crowdedness**. In the velocity bounded model we define $Crowd_{\mathsf{v}}(u)$ of a node u by the set of all other nodes in distance $2v_{\max}\Delta$. Its cardinality defines $crowd_{\mathsf{v}}(u)$, the crowdedness of u.

In the acceleration bounded model we define the crowd of a node u by

$$\begin{aligned} Crowd_{\mathsf{a}}(u) := \{w \in S \setminus \{u\} \ : \ |u - w| \ &\leq \ \frac{1}{2}a_{\max}\Delta^2 \\ \text{and } |u' - w'| \ &\leq \ \frac{1}{2}a_{\max}\Delta\} \ , \end{aligned}$$

where u, w denote the starting positions, and u', w' the starting vector of mobile stations for the time interval $[0, \Delta]$. The crowdedness $crowd_{\mathsf{a}}(u)$ is defined by its cardinality.

In [42] we have presented a distributed algorithm to build a mobile network, which allows small congestion, few interferences, low energy data routes, small degree and small diameter. This can be summarized as follows.

There exist distributed algorithms that construct mobile networks for the velocity bounded and the acceleration bounded mobility model with the following properties:

- The interference number of the mobile network approximates the optimal interference number by a factor of $O(\log n)$.
- Energy-optimal routes can be approximated by a constant factor.
- The degree is bounded by $O(crowd_\alpha(S) + \log n)$ and the diameter is at most $O(\log n)$, where $\alpha \in \{\mathsf{v}, \mathsf{a}\}$.
- The mobile networks allow data routes on this mobile network inducing a congestion of at most $O(\log^2 n)$ times the congestion of the optimal routing.

3 Mobile Sensor Networks, Autonomos Robots

Mobile sensor networks (MSN) are distributed collections of nodes, where each node has sensing, computation, and locomotion capabilities.

By assuming a large scale sensor network with unpredictable sensor failure, limited sensing and communication ranges, decentralized or localized sensor self-deployment methods are more beneficial and scale invariant than centralized solutions. In this context localized means that each sensor makes independent decisions using neighborhood information only.

Self-deployment and pattern formation problems are fundamental problems in mobile sensor networks and autonomous robots. The sensors must cover a dedicated area, a border, an area around a so called Point of Interest (POI), or form a given connected pattern.

In the field of mobile sensor networks sensor self-deployment problem has been an important research topic that deals with autonomous coverage formation.

In the article of Gage et al. [107] three type of formation was introduced. However it was a military oriented article, from the perspective of the F-coverage only the *blanket* formation is relevant. In this formation the nodes form static connected group in order to maximizes the detection rate of targets appearing within the coverage area.

The most common sensor self-deployment method is the vector or virtual-force-based approach. The algorithms which rely on this approach use potential fields, generated around the sensors which moves the neighbors by attract or repulse them (depending on the distance). The first work which used this approach was published by Howard et al. [113].

Large amount of research deals with sensor deployment algorithms for coverage formation over a Region of Interest (ROI). An excellent summary can be found in the works of Nayak et al. [145] and Brass et al. [72].

Cortes et al. [84] proposed Voronoi diagram based sensor self-deployment method for the coverage of the ROI. The main idea of self-deployment with Voronoi diagrams is to move sensors to minimize their local uncovered areas (equivalently speaking, to maximize their sensing-effective areas) by aligning their sensing range with their Voronoi regions as much as possible.

Li at al. [129], [127], [128] introduced the F-coverage problem. They solved the problem in a discrete case on an equilateral triangle tessellation. Collision of sensors during the deployment was allowed, i.e. more than one sensors can occupy the same triangle vertex at the same time. They presented a proof of the convergence of their solution within finite time. The convergence time, energy consumption and number of collisions has been evaluated by simulations.

In the work of Yang et al. [177] a distributed load-balancing sensor self-deployment algorithm was presented which partitions the plane into a 2D mesh, and treats nodes as load. By this algorithm, nodes in each cell form a cluster covering the cell and are managed by an elected cluster head. This approach also requires dense network coverage and inter-agent communication.

Bartolini et al. [64] have presented a localized algorithm on a hexagonal grid map in which the entities simultaneously use the *snap* and the *spread* activities in order to cover the given area. The nodes are dispersing from their initial position while occupying the free hexagons. On each occupied hexagon only the occupier allowed to stay, which forwards the others towards the borderline of the covered area.

Cord-Landwehr et al. [82] studied the problem of gathering mobile robots with an extent at a fixed position as dense as possible to form a disk of minimum radius

around the gathering point. The authors present an algorithm for the continuous case and the discrete case, where the robots are moving on a grid. They prove an $O(nR)$ upper bound for the gatheringg time, where n is the number of robots and R is the distance of the farthest robot from the gathering point. They empirically studied the continuous case, where in they report a few deadlock situations in the simulations.

Cohen and Peleg [80] presented an asynchronous algorithm to gather oblivios robots at the center of gravity. Their algorithm uses the LCM (Look-Compute-Move) discrete cycle based model to move their robots. They mathematically proved upper and lower bounds on the convergence speed of their solution.

Cord-Landwehr et al. [83] described an easy-to-check property of target functions that guarantee convergence and gives upper time bounds. This property holds for the target function in [80] and improves the upper bound on the speed of the convergence.

Czyczowicz et al. [87] considered the gathering problem for few fat robots, where the robots are modeled by unit disks. The goal was to gather the robots, such that the union of the unit disks is connected at the end. Collisions of the robots are not allowed during the gathering. A main problem which had to be solved here is that the line of sight of a robot may be blocked by the extent of other robots.

For the gathering problem of mobile robots many different variants exist differing in levels of synchronization, computational power of the robots, memory, range of visibility, agreement on coordinate system. For a survey we refer to [79].

Another related problem in distributed robotics is the Pattern Formation problem, where a group of mobile robots have to form a desired geometric pattern. The pattern can be given as set of points in the plane (by their coordinates) or as a predicate (e.g. "form a circle"). A common requirement is that the robots have distinct initial positions and that the number of points in the pattern and the number of robots are the same. Suzuki and Yamashita [170, 171] investigated the question what kinds of patterns can be formed by a group of autonomous, anonymous and homogenous mobile robots that do not communicate, but they are able to observe each others movements. In [170, 171] the authors have shown that without agreeing a common coordinate system, a pattern can be formed if and only if it is purely symmetrical, i.e., a regular polygon (or a point), or a set of regular concentric polygons. They also have shown that by agreeing on a coordinate system, the robots can form any geometric pattern. Flocchini et al. [105] have shown that if each robot has a compass needle that indicates North (the compass needles are parallel), then any odd number of robots can form an arbitrary pattern, but an even number, in the worst case, cannot. If each robot has two independent compass needles, say North and East, then any set of robots can form any pattern. Pattern Formation by robots with limited visibility has been studied in [176].

Another related fundamental problem is the Filling problem (see [61]), in

which a given region must be covered by robots. In this problem the robots are initially not in the region, they enter the the space one by one, from a point called "door". When a robot enters the door, it must disperse itself in the region. The goal is to cover the entire region. Barrameda et al. [61] have proven that the Filling problem can be solved with limited visibility, for any simple orthogonal space, i.e., a polygonal region without holes with sides either parallel or orthogonal, with a single door, by finite-state robots with a common coordinate system and common unit of distance in finite time.

For an excellent overview on distributed computing by mobile robots we refer to the the book by Flocchini et al. [102].

We investigate the self-deployment problems in synchronous Look-Compute-Move model and provide upper and lower bounds and efficient solutions [37, 4, 6, 5, 3].

3.1 Localized sensor self-deployment for focused coverage

We consider the focused coverage self-deployment problem in mobile sensor networks, where an area with maximum radius around a Point of Interest (POI) must be covered without sensing holes. Li et al. [128, 129] described several algorithms solving this problem. They showed that their algorithms terminate in finite time. In [4] we have presented a modified version of the Greedy-Rotation-Greedy (GRG) algorithm by Li et al., which drive sensors along the equilateral triangle tessellation (TT) graph to surround a POI. We have proven that our modified GRG (mGRG) algorithm is collision free and always ends up in a hole-free network around the POI with maximum radius in $O(D)$ steps, where D is the sum of the initial distances of the sensors from the POI. This significantly improves the previous bound on the coverage time. The theoretical results have been also validated by simulations.

3.1.1 Focused coverage (F-coverage)

We follow the terminology of Li et al. [129]. The *coverage region* of a sensor network is the region which is enclosed by the outer boundary of the network. If the coverage is not complete there will be still *sensing* (or coverage) *holes*. Sensing holes are closed areas inside the coverage region which are not not covered by the sensing range of the sensors.

The *coverage radius* (or *radius of an F-coverage*) is the radius of the maximal hole-free disc enclosed by sensors and centered as POI. The optimal F-coverage has maximized coverage radius. If the number of sensors is unlimited and the sensing radius of the sensors approaching zero then the maximum hole-free disc has a circular shape. Since the sensing radius of the sensors is finite, we consider a discrete variant of coverage radius measured by *layer distance*. Layer distance, also called convex layers in computational geometry represents the number of

successive complete convex polygons adjacently surrounding POI. More precisely, we consider a discrete set of convex polygons $P_i, (i = 1, 2, ...)$ composed of sensors, centered at POI, and having a diameter of $i \cdot d$ for some constant d. Then the coverage radius is the maximum value k, such that P_k is completely in the coverage region.

3.1.2 The equiliteral triangle tessellation

The equiliteral triangle tessellation is a tiling of the plane in equiliteral triangles with no overlaps and no gaps. The equilateral triangle tessellation (TT) maximizes the coverage area of a given number of sensors without coverage gap when sensor separation is equal to $r_s\sqrt{3}$, where r_s is the sensing radius of the sensors [57], [134]. If the communication radius r_c of the sensors is at least $r_s\sqrt{3}$, the deployment of the sensors corresponding to a TT layout guarantees the connectivity of the network. The convex polygons defining the layers and the layer distance of the F-coverage are hexagons centered at the POI.

3.1.3 Problem

We are given n mobile sensors with communication radius r_c and sensing radius is r_s of each, $r_c \geq r_s\sqrt{3}$. We assume that the n mobile sensors are initially placed at the vertices of the TT, such that each sensor is placed in a different vertex. This is an unrealistic assumption if the sensors are dropped from a plane. In that case the sensors can perform the Snap and Spread algorithm by Bartolini et al. [64] to achieve the above condition. The sensors may be disconnected at the beginning. All sensors have a common coordinate system and they all know the location of the POI. Without loss of generality, the POI at the origin of the coordinate system. Furthermore, the sensors only have information about their 1- and 2-hop neighbors. The sensors are able to move only on the edges of the TT graph. They all move synchronously with uniform speed, s.t. they travel an edge of the TT in one time unit.

The sensors operate corresponding to the *Look-Compute-Move* model. In one cycle, a sensor takes a snapshot of the current configuration (Look), makes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case makes an instantaneous move to this neighbor (Move).

The motion ends when the sensors uniformly surround the POI by forming hole-free network with maximized coverage radius. From now on we will use the terms node and sensor interchangeably.

3.1.4 Our contribution

In [4] we have presented a modified version of the GRG/CV algorithm of Li et al. [129]. We have proven that our modified GRG (mGRG) algorithm guarantees, that after $O(D)$ steps each node reaches its final layer, where D is the sum of

initial hop distances of the nodes from the POI in the TT. The theoretical results have been also validated with simulations.

An important difference between the requirements of the GRG of Li et al. [129] and our mGRG algorithm is that the GRG in [129] uses the knowledge about the 1-hop neighborhood of the sensors, while our mGRG algorithm needs the knowledge of the 2-hop neighborhood. We give examples, that show that the knowledge about the 2-hop neighborhood of the sensors is necessary to avoid collision situations and make the deployment process faster.

3.2 Pattern formation for anonymous, position-aware robots

We consider various pattern formation problems by n identical autonomous robots on a 2D grid. They are memoryless (or use only $O(1)$ bits of persistent memory) and operate without explicit communication. They have computation and locomotion capabilities and limited visibility range. They are represented by discs of unit diameter. Each robot r knows its position $p_r \in \mathbf{Z}^2$ but not the position of the other robots. In addition, each robot knows the connected pattern F to be formed. F may be given by a set of n points in $\mathbf{Z}^2$, or as a predicate, e.g., "form a horizontal line segment", or may be only partially described, e.g., by "form a connected pattern", or "build a connected formation with minimum diameter" (Collisonless Gathering). All robots have a common coordinate system. Each robot has a visibility range of 2 units, i.e. it can see the robots within its local range of 2 units. With other words, the robots only have information about their 1- and 2-hop grid neighbors. The robots are able to move only on the edges of the grid. They all move synchronously with unit speed, s.t. they travel an edge of the grid in one time unit.

The robots operate corresponding to the *Look-Compute-Move* (LCM) model. In one cycle, a robot takes a snapshot of its current visibility range (Look), makes a decision to stay idle or to move to one of the neighboring vertices (Compute), and in the latter case makes an instantaneous move to this neighbor (Move). We assume that the LCM cycles are synchronous at each robot. Collisions are not allowed during the algoritms, i.e. in each time step each vertex of $\mathbf{Z}^2$ can be occupied by at most one robot. The motion ends when the robots form the connected pattern F. From now on we will use the terms node and robot interchangeably.

3.2.1 Our contribution

In [37] first we consider a helpful intermediate *gathering* (or *point formation*) problem - we call it the *Lemmings problem* - where collision at one single point g, known to all robots, is allowed. The goal is that all robots gather at g. We consider *oblivious* robots, i.e. the robots do not remember results from any

of the previous computations. We present an algorithm solving this problem, called x-y-routing, where the robots only need local knowledge about their 2-hop neighborhood in the grid $\mathbf{Z}^2$. We show that the x-y-routing method can be used to guarantee the gathering of all robots at g in $2n + D - 1$ time steps, where D is the maximum initial hop distance of a robot from g. We prove that this time bound is optimal up to a constant factor.

After this we investigate the gathering problem of n oblivious robots, where no collision is allowed at g and the robots have to form a connected configuration containing g. We show that the x-y-routing solves this problem in $n+D-1$ time steps. This significantly improves the previous upper bound of $O(nD)$ on this problem presented in [82].

After this we consider *finite state* robots, i.e., the robots can use $O(1)$ bits of persistent memory for the computation. We show, how the set of n robots can be arranged to form a connected axis parallel line segment containing a given point g, known to all robots, in $3n + D + 3$ steps.

Finally, for finite state robots, we show how an arbitrary connected pattern F, known to all robots, can be formed in time $O(n + D^*)$, where D^* denotes the diameter of the point set consisting of the initial configuration and F. In case when all robots know n, this solution can also be applied for solving the focused coverage problem on the 2D grid. This results in $O(n + D)$ covering time. If the number of robots n is not known for the robots, then best known upper bound on this problem is $O(S)$, presented in [4], where S is the sum of initial distances of the mobile sensors from g.

3.3 Uniform Dispersal of Myopic Robots

In swarm robotics, a huge number of simple, tiny robots can perform complex tasks collectively without central control. Such systems are scalable, reliable, and can provide a high grade of fault tolerance. Many distributed protocols have been developed for a wide range of problems, like gathering, flocking, pattern formation, dispersing, filling, coverage, exploration (e.g., [46, 52, 59, 62, 63, 81, 86, 114]; see [73, 103, 104] for recent surveys).

We consider the problem of filling an unknown area represented by a connected graph of n vertices by autonomous mobile robots. In this problem, the robots enter the graph one-by-one through specific vertices, called the doors, and they have to cover all vertices of the graph while avoiding collisions. The robots are anonymous and make decisions driven by the same local rule of behavior. They have limited persistent memory and limited visibility range. We investigate the Filling problem in the synchronous and asynchronous Look-Compute-Move (LCM) model.

The Filling (or Uniform Dispersal) problem was introduced by Hsiang et al. [114] for an orthogonal area, where the area is subdivided into square-shaped pixels. The robots are placed at the same entry point one-by-one and have to

occupy all the pixels. The entry point is called the door. In a step, a robot can move to a neighboring cell sharing a common side or stay at its current position. During the algorithm, the collision of the robots must be avoided, i.e., each pixel can be occupied by at most one robot at any time. In the final configuration, all pixels must be occupied by exactly one robot. When more than one door is present in the area, the problem is called *multiple door filling* or *k-door filling*.

The main question asked by Barrameda et al. [62, 63] is to determine the minimum hardware requirement of the robots to solve the Filling problem. They propose algorithms for orthogonal regions by robots with constant visibility radius, constant communication range, and a constant number of bits of persistent memory. In [62], common top-down and left-right directions and externally visible colors were assumed for the multiple door filling.

Barrameda et al. [63] presented two methods for filling an unknown orthogonal area in presence of obstacles (holes) in the asynchronous (ASYNC) model. Their first method, called TALK, requires a visibility range of 2 hops[1] if the robots have explicit communication. The other method in [63], called MUTE, does not use explicit communication between the robots, but it requires a visibility range of 6. Both methods need $O(1)$ bits of persistent memory and terminate in finite time.

The model of luminous robots was introduced by Peleg [149]. Subsequently, significant amount of research has been carried for a plenty of problems using this model (e.g. [47, 69, 71, 98, 106, 118, 132, 147, 165, 164, 166]). Das et al. [88, 89] considered the model, where the robots can move in the continuous Euclidean plane, and they proved that the asynchronous model with a constant number of colors ASYNC$^{O(1)}$ is strictly more powerful than the semi-synchronous model SSYNC, i.e. ASYNC$^{O(1)}$ > SSYNC. Das et al. [89] also prove that there are problems that robots cannot solve without lights, even if they are fully synchronous, but can be solved by asynchronous luminous robots with $O(1)$ colors.

D'Emidio et al. [90] have shown that on graphs one task can be solved in the fully synchronous model FSYNC but not in the asynchronous lights-enhanced model, while for other tasks, the converse holds. In this work, we show that the Filling problem can be solved in both models by robots with 1 hop visibility range and $O(1)$ bits of persistent memory.

3.3.1 Our contribution

Synchronous model: In [24] and [26] we investigate the Filling problem in the synchronous model. In [24] we present an algorithm, called *SDIR* for the Filling problem in connected orthogonal regions, where the region is subdivided into square-shaped cells, so-called pixels. The visibility range of the robots is

[1] In [63] it is assumed that the robot sees all eight sourrounding cells and able to communicate with robots at that eight cells. Assuming orthogonal movements, a cell sharing only one corner with the current cell of the robot are reachable in two hops.

one hop, and they have $O(1)$ bits of persistent memory, which is optimal and improves previous bounds. The Filling time in the synchronous model is $O(n)$, where n is the number of cells in the region. First, we present an algorithm for the single door case, then we extend the solution for multiple doors, where the robots are entering the area in k different doors. For the k-door case, the visibility range of the robots is still 1 hop, and the running time is $O(n)$, and the memory requirement is still $O(1)$.

In [26] we invetigate the general problem where the area is represented by an arbitrary connected graph. The robots have a visibility range of 1 hop, i.e., the robots can see adjacent vertices. In one time step, a robot can move to an adjacent vertex or stay in place while avoiding collisions. In the final configuration, all vertices of the graph must be occupied by exactly one robot. We present a method, called the *Virtual Chain Method* (*VCM*), for the single door Filling problem by a set of autonomous anonymous robots with a visibility range of 1 hop in $O(\Delta \cdot n)$ time in the synchronous computational model, where n is the number of vertices of the graph with a maximum degree of Δ. The robots require $O(\Delta)$ bits of persistent memory. Then we consider the multiple door case, when the robots enter in $k > 1$ doors, and we generalize the VCM algorithm for solving this problem. The robots need a visibility range of 1 hop, $O(\Delta + \log k)$ bits of persistent memory. The algorithm terminates in $O(k \cdot \Delta \cdot n)$ time.

The algorithms in [26] are optimal in terms of visibility range. This follows from the fact that with a visibility range of less than 1, the robots cannot even distinguish between occupied and unoccupied neighbors. For constant k and constant Δ, our algorithm is asymptotically optimal in the size of the memory. This follows from the result by Barrameda et al. [62], they proved that oblivious (memoryless) robots cannot deterministically solve the problem. Moreover, for constant k and constant Δ, our algorithm is asymptotically optimal in running time. The asymptotic optimality of the running time $O(n)$ follows from the fact that we can place one robot per round in the single door case, and n robots must be placed.

By extending the visibility range of the robots to 2 hops and coloring them by k different colors the running time can be improved to $O(\log \Delta \cdot n)$ rounds.

In [28] we present a modification of the VCM algorithm, which improves the running time to $O(k + \Delta) \cdot n)$

In [23] we investigate the 3D orthogonal problem. In [27] we study randomized solutions.

Asynchronous model, luminous robots: In [25] we describe a method, called PACK, which solves the problem by robots with 1 hop visibility range, $O(\log \Delta)$ bits of persistent memory, and $\Delta + 4$ colors for the single Door case, including the color when the light is off. We analyze the algorithm in terms of asynchronous rounds, where a round means the smallest time interval in which each robot, which has not yet finished the algorithm, has been activated at least once. We show that this algorithm needs $O(n^2)$ asynchronous rounds. Regarding

asynchronous algorithms for the Filling problem, former works only guarantee termination within finite time. Our analysis provides the first asymptotic upper bound on the running time in terms of asynchronous rounds.

We show how the number of colors can be reduced to $O(1)$ at the cost of running time. The algorithm with 1 hop visibility range, $O(\log \Delta)$ bits of persistent memory, and $O(1)$ colors needs $O(n^2 \log \Delta)$ rounds.

We show how the running time can be significantly improved by robots with a visibility range of 2 hops, with no communication, $O(\log \Delta)$ bits of persistent memory, and $\Delta + 4$ colors, by presenting the algorithm called BLOCK. This algorithm needs $O(n)$ rounds.

We extend the BLOCK algorithm for solving the k-Door Filling problem, $k \geq 2$, by using $O(\log \Delta)$ bits of memory and $\Delta + k + 4$ colors, including the color when the light is off. The visibility range of 2 hops is optimal for the k-Door case.

4 Large Scale Content Distribution

The simplicity of creating, distributing and communicating content causing an enormous load on the Internet. Peer-to-peer (P2P) systems provide a popular and efficient way of content distribution. They generate a significant part of the overall Internet traffic. In tracker-based P2P networks, like BitTorrent (BT), the effectiveness of downloading data largely depends on the neighbors of the client and on the distribution and selection of the pieces of the file. We model BitTorrent-like systems, derive upper and lower bouns on the distribution time, and develop efficient methods for the data distribution.

Data compression and decompression are essential tools to decrease bandwith demand and increase throughput during data transmission. It is of high interest for distributing large or real-time data. A particular focus of our research is data (de-)compression on resource constrained devices, such sensor nodes or embedded devices. The goal is to reduce the memory need of the devices without sacrificing speed too much.

We give a summary of our results on data (de-)compression [34, 32], which also lead to the international patent [33]. Then we present our results on tracker-based, BitTorrent-like peer-to-peer (P2P) data distribution [30, 31] and on applying network coding for more efficient download [29, 1, 35, 2, 17, 14]. Further results on related topics, especially on content distribution using so called mobile based social networks appeared in [7, 18, 13, 11, 15, 16, 12, 14, 10, 9, 8]. Results on I/O efficient algorithms for processing large data that are much larger than the size of the main memory and require secondary storage (e.g., the data is stored in in disks, storage networks, or even in multiply data centers) are presented in [21, 20, 36].

4.1 Data compression and decompression

The Burrows-Wheeler transformation is used for effective data compression, e.g., in the well known program bzip2. We are focusing on resource constrained devices, such sensor nodes or embedded devices.

4.1.1 Space efficient Burrows-Wheeler backtransformation

The Burrows-Wheeler transformation (BWT) [75] is at the heart of modern, very effective data compression algorithms and programs, e.g., bzip2 [161]. BWT-based compressors usually work in a block-wise manner, i.e., the input is divided into blocks and compressed block by block. Larger block sizes tend to result in better compression results, thus bzip2 uses by default a block size of 900,000 bytes and in its low memory mode still 100,000 bytes. The standard algorithm for decompression (reverse BWT) needs auxiliary memory of 4 bytes per input character, assuming 4-byte computer words and thus $n < 2^{32}$. This may pose a problem in embedded systems (say, a mobile phone receiving a software patch over the air interface) where RAM is a scarce resource. In such a scenario, space requirements for compression ($8n$ bytes when a suffix array [135] and the standard algorithm in [135] or the algorithm in [121] is used)[2] is not an issue, as compression is done on a full fledged host. In the target system, however, cutting down memory requirements may be essential.

The BWT Backtransformation: We will not go into details of the BW-transformation here, as it has been described in a number of papers [48, 58, 75, 99, 101, 136] and tutorials [45, 146] nor do we give a proof of the reverse BWT algorithm. Instead, we give the bare essentials needed to understand the problem we solve in the following sections. The BWT (conceptually) builds a matrix whose rows contain n copies of the n character input string, row i rotated i steps. The n strings are then sorted lexicographically and the last column is saved as the result, together with the "primary index", i.e., the index of the row that contains - after sorting - the original string. The first column of the sorted matrix is also needed for the backtransformation, but it needs not to be saved, as it can be reconstructed by sorting the elements of the last column. (Actually, as we will see, the first column is also needed only conceptually).

Figure 2 shows the first and last columns resulting from the input string "ALIBABA". The arrow indicates the primary index. Note that we have numbered the occurrences of each character in both columns, e.g., row 2 contains the occurrence 0 of character "A" in L, row 3 contains occurrence 1, and row 6 contains occurrence 2. We call these numbers the *rank* of the character within column L.

[2] For the construction of the suffix array using less than $8n$ bytes algorithms are presented e.g. in [67, 74, 115, 137, 143, 162]. For a comprehensive survey of suffix array constructions, we refer to [152].

		F	L rank	base
	0	A_0	B_0	A: 0
	1	A_1	B_1	B: 3
$\longrightarrow$	2	A_2	A_0	I: 5
	3	B_0	A_1	L: 6
	4	B_1	I_0	
	5	I_0	L_0	
	6	L_0	A_2	

Table 2: Last (L) and first (F) column for the input string "ALIBABA"

To reconstruct the input string, we start at the primary index in L and output the corresponding character, "A", whose rank is 0. We look for A_0 in column F, find it at position 0 and output "B". Proceeding in the same way, we get "A", "B", "I", "L", and eventually "A", i.e., the input string in reverse order. The position in F for a character/rank pair can easily be found if we store for each character of the alphabet the position of its first occurrence in F; these values are called *base*.

This gives us a simple algorithm when the vectors *rank* and *base* are available:

```
int h = primary_index;
for (int i = 0; i < n; i++) {
  char c = L[h];
  output(c);
  h = base[c] + rank[h];
}
```

The *base*-vector and *rank* can easily be calculated with one pass over L and another pass over all characters of the alphabet.

```
for (int i = 0; i < 256; i++) base[i] = 0;
for (int i = 0; i < n; i++) {
  char c = L[i];
  rank[i] = base[c];
  base[c]++;
}

int total = 0;
for (int i = 0; i < 256; i++) {
  int h = base[i];
  base[i] = total;
```

```
    total  += h;
  }
```

These algorithms need $O(n)$ space (n words for the *rank*-vector) and $O(n)$ time. Alternatively, we could do without precalculation of rank-values and calculate $rank[h]$ whenever we need it, by scanning L and counting occurrences of $L[h]$. This would give us $O(1)$ space and $O(n^2)$ time.

The question, now, is: is there a data structure that needs significantly less than n words without increasing run time excessively?

In this paper we present efficient data structures and algorithms solving following problems:

Rank searching: The input must be preprocessed into a data structure, such that for a given index i, it supports a query for $rank(i)$. This query is referred to as rank-query.

Rank-position searching (Select): The input must be preprocessed into a data structure, such that for a given character c and rank r, it supports a query $select(c, r)$ for index i, such that $L[i] = c$ and $rank(i) = r$. This query is referred to as select-query. (This allows traversing L and F in the direction opposite to that discussed so far, producing the input string in forward order).

Computation model: As computation model we use a random access machine (RAM) (see e.g., in [51]). The RAM allows indirect addressing, i.e., accessing the value at a relative address, given by an integer number, in constant time. In this model it is also assumed that the length of the input n can be stored in a computer word. Additionally, we assume that the size $|A|$ of the alphabet A is a constant, and particularly, $|A| - 1$ can be stored in a byte. Furthermore, we assume that a bit shift operation in a computer word, word-wise *and* and *or* operations, converting a bit string stored in a computer word into an integer number and vice-versa and algebraic operations on integer numbers ('+', '-', '*', '/', 'mod', where '/' denotes the integer division with remainder) are possible in constant time.

Previous results: In [163] Seward describes a slightly different method for the reverse BWT by handling the so-called transformation vector in a more explicit way. He presents several algorithms and experimental results for the reverse BWT and answering rank-queries (more precisely, queries "how many symbols x occur in column L up to position i?", without the requirement $L[i] = x$). A rigorous analysis of the algorithms is omitted in [163]. The algorithms described in [163], `basis` and `bw94` need $5n$ bytes of memory storage and support a constant query time; algoritm `MergedTL` needs $4n$ bytes if n is limited to 2^{24} and supports

a constant query time. The algorithm `indexF` needs $2.5n$ bytes if $n < 2^{20}$ and $O(\log |A|)$ query time. The algorithms `tree` and `treeopt` build 256 trees (one for each symbol) on sections of the vector L. They need $2n$ and $1.5n$ bytes, respectively, if $n < 2^{20}$ and support $O(\log(n/\Delta) + c_x\Delta)$ query time, where Δ is a given parameter depending on the allowed storage and c_x is a relatively big multiplicator which can depend on the queried symbol x.

Beside the computation of the reverse BWT, there is a large amount of recent literature on BWT-based compressed full-text indexes, where rank- and select-queries are used for searching patterns in the compressed file as well as for retrieving (portions of) the original file. For an extensive survey on compressed full-text indexes, see [144]. In [100] bucketing schemes are presented for the computation of rank-queries. In this solution the BWT of the string is partitioned into superbuckets and each superbucket is partitioned into buckets. For each superbucket the number of occurences of every character in the previous superbuckets is stored. The buckets are compressed and stored in a bucket directory. Each compressed bucket also includes a header containing the number of occurences of every character since the beginning of the superbucket. The rank-queries are answered by retrieving and decompressing the appropriate bucket and counting the number of occurences of the corresponding character from the beginning of the bucket. For the compression of buckets unary coding, hierarchical 3-level coding, arithmetic coding, and Huffman coding has been used.

Our contributions: In [34, 32] we present a data structure which supports answering a rank-query $Q(i)$ in $O(1)$ time using $n(\frac{\ell-1}{8} + \frac{w|A|}{2^\ell})$ bytes, where w denotes the length of a computer word in bytes, and $|A|$ is the size of the alphabet. If $|A| \leq 256$ and $w = 4$ (32 bit words), by setting $\ell \in \{12, 13\}$, we obtain a data structure of $\frac{13}{8}n$ or 1.625 bytes. For $w = 2$ we get a data structure of $\frac{25}{16}n$ or 1.5625 bytes. Thus, the space requirement is strictly less than that of the trivial data structure, which stores the rank for each position as an integer in a computer word and that of the methods in [163] with constant query time. The preprocessing needs $O(n)$ time and $O(|A|)$ working storage.

We also present data structures of n bytes, where we allow at most $L = 2^9$ sequential accesses to a data block of L bytes. Because of caching and hardware prefetching mechanism of todays processors, with this data structure we obtain a reasonable query time.

Furthermore, we present a data structure, which supports answering a rank-query $Q(i)$ in $O(t)$ time using t random accesses and $c \cdot t$ sequential accesses to the memory storage, where c is a constant, which can be chosen, such that the speed difference between non-local (random) accesses and sequential accesses is utilized optimally. The data structure needs $\frac{n(8+|A|\log ct)}{8ct} + \frac{n|A|w}{ct^2}$ bytes. For $t = \omega(1)$, this results in a sub-linear space data structure, e.g., for $t = \Theta(n^{1/d})$ we obtain a data structure of $\frac{1}{d}n^{1-1/d}|A|(1 + o(1))$ bytes. The preprocessing needs $O(n)$ time and

$O(|A|)$ working storage.

After this, we turn to the inverse problem, the problem of answering select-queries. We present a data structure of $n(\frac{|A|(\ell+8w)}{2^\ell} + \ell)$ bits, which supports answering select-queries in $O(\log(n/2^\ell))$ time. The preprocessing needs $O(n)$ time and $O(|A| + 2^\ell)$ working storage. For $\ell = 13$, we obtain a data structure of $14\frac{3}{8} \cdot n$ bits.

In [34, 32] we also present experimental results, that show that our algorithms perform quite well in practice. Thus, they give significant improvement for decompression in embedded devices for the mentioned scenarios.

4.2 Content distribution in P2P networks

The simplicity of creating, distributing and communicating content causing an enormous load on the networks. P2P systems provide a very popular and efficient way of content distribution. In tracker-based P2P networks, like BitTorrent, the effectiveness of downloading data largely depends on the neighbors of the client and on the distribution and selection of the pieces of the file. We give an overwiew of our results on neighbor selection and topology management in BitTorrent-like networks and on using network coding for more efficient content distribution.

4.2.1 Topology management and data distribution in tracker based networks

In the past decade, tracker-based peer-to-peer networks like BitTorrent (BT) [43] and Tribler [44] have emerged as popular solutions in the area of not only simple file-sharing, but video-on-demand services as well. These applications are still showing an increasing interest, generating a significant part of the overall Internet traffic.

The selection of neighbors is an important design decision of peer-to-peer systems. In tracker-based peer-to-peer networks, each peer that enters the network, first has to connect to a central component called tracker to obtain a peer set representing the initial neighborhood of the joining client. The tracker maintains a list of all nodes in the system, called the swarm, and returns a random subset of the existing nodes. This random neighbor selection may lead to suboptimal overlay topologies. In order to optimize the network, various neighbor selection strategies can be found in the literature that considers different aspects from locality [172] and load balancing [49, 111] to quality of experience [150].

The performance of BT like peer-to-peer systems has been widely analyzed in the past few years from theoretical and practical aspects as well. The empirical results [116, 70] show that the simple routing policy applied by the original BT is quite effective even in case of a flash crowd setting when a great deal of peers join the network almost at the same time. Besides empirical evidences, this heavy loaded case has already been investigated from a theoretical perspective as well.

In [49], several algorithms are demonstrated that share b data blocks among n clients in a network of diameter d and degree D in $O(D(b+d))$ steps with high probability[3], where in one time step, each client can upload one data block to, and download one block from one of its neighbor. For a network used by BT it results in a time bound of $O(b \ln n)$ time steps. They propose a neighbor selection strategy which improves this bound to a near-optimal $O(b + (\ln n)^2)$ steps.

In [30, 31] we improve the neighbor selection strategy resulting in a time bound of $O(b + \ln n)$ steps, which is optimal up to a constant factor.

Our method uses the idea of multiple choice [55] and takes into account not only the actual load of the peers, but the possibility as well that a client will be selected in the future. This will ensure an overlay network of constant degree and logarithmic diameter with high probability. The constructed overlay topologies are examined from both theoretical and practical perspectives as well. We model these overlay networks as a graph, whose vertices are the peers and neighboring peers are connected by an edge. We analyze the key graph properties of the proposed network, showing that the maximum degree in our overlay topology is $O(1)$, with high probability, while its diameter still remains logarithmic in the number of peers n. In such a network, the randomized upload policy will share b data blocks among n clients in $O(b+\ln n)$ time steps with high probability, which is optimal in networks of n vertices, in which the degree of the vertices is bounded by a constant. Besides the theoretical analysis thorough simulations have been performed to validate the different properties of the constructed overlay networks. In addition to the analysis of the degree and diameter distributions, we also show how our balanced algorithm could work in a tracker-based file sharing system such as BitTorrent to accelerate the delivery of the data blocks in the whole network. The theoretical results are backed up by simulations. The show that our balanced overlay construction could work not only in theory, but in practice as well.

4.2.2 Network coding for robust data distribution in BitTorrent-like networks

In the BitTorrent protocol the file is divided into pieces and the clients upload and download the pieces to each other. The tit-for-tat rule of the protocol enforces the cooperation between selfish peers. In the original BitTorrent protocol sometimes the neighbors of a peer have no piece that the peer does not already have, which inhibits the download. Rare pieces can cause long waiting times. An elegant way of solve these problems by Locher et al. [131] is the extension of the original protocol by network coding. This coding method increases the diversity of the pieces in the network which accelerates tit- for-tat piece exchange and leads to faster downloads. In [1, 2] we propose a novel deterministic source coding method.

[3] An event E is said to occur with high probability, if given $n > 1$, $Pr[E] > 1 - 1/n^c$, where $c > 1$ is a constant [142].

The main advantages of our method compared to state of the art random coding methods are the reduced traffic overhead and the guarantee of decodability of the original file after downloading d different coded pieces, where d is the number of pieces of the file. We analyze the deterministic method theoretically. The theoretical results are backed up by simulations.

In [35] we conduct further simulations for the extension of BitTorrent by random network coding and the deterministic coding. In [17] we provide a special version of random coding and present simulation results.

4.2.3 Resource constrained BitTorrent-like networks

In [29] we present a solution that makes BitTorrent content transfer for mobile device more energy efficient. The main idea is that instead of downloading the content via BitTorrent directly to the mobile phone, an intermediate proxy is used which sends the data to the phone in high speed bursts. This results in smaller energy footprint compared with regular BitTorrent data transfer. Furthermore, we focus on how the proxy can be hosted on memory limited broadband routers which are available in almost every home. We define an analytical model which can be used to analyze the memory allocation strategies of the proxy peers and predict how proxy peers influence the P2P community performance. We verify our model via simulations. In [29] we also present measurement results with real life torrents using our prototype system running on home routers mobile phones.

Main contributions: In [29] our explicit interest is how the system can deliver a torrent to a battery powered, wirelessly connected mobile device energy-efficiently. So far the key performance metric of most BitTorrent research has been download time.

Then we mathematically model and analye how the limited resources of the proxy peer influence BitTorrent performance. The ability to store and share pieces that a peer has already downloaded is one of the key concepts of BitTorrent. If the storage space of the proxy is limited, downloaded pieces have to be discarded after they have been transmitted to the mobile, and thus cannot be shared any longer. However, there is no method in the BitTorrent protocol to announce that a formerly shared piece has been discarded. We ensure that these solutions are compatible with the existing BitTorrent clients and that they do not harm the downloading performance of regular peers.

On the practical side, we have shown that using broadband routers as proxies for BitTorrent downloads is feasible and results in energy savings and user experience improvements via shorter download times.

On the theoretical side, we have investigated how BitTorrent works on memory-limited devices. Only a relatively small number of pieces are required to achieve full upload utilization and reach good download speeds. Most of the

available memory should be allocated to serving other peers. This memory allocation can be done statically using our analytical formulas or with an adaptive algorithm. The amount of data that is sent to the mobile device in one pass is another important parameter influenc- ing the performance. In general a bigger chunk size can save energy but it may increase the download time.

5 Network Function Virtualisation, Software Defined Networks

Today's computer networks provide a rich set of in-network functions, including access control, firewall, intrusion detection, network address translation, traffic shaping and optimization, caching, among many more. While such functionality is traditionally implemented in hardware middleboxes, computer networks become more and more virtualized [93, 160]: *Network Function Virtualization (NFV)* enables a flexible instantiation of network functions on network nodes, e.g., running in a virtual machine on a commodity x86 server.

Modern computer networks also offer new flexibilities in terms of how traffic can be routed through such network functions. In particular, using *Software-Defined Networking (SDN)* [139] technology, traffic can be steered along arbitrary routes, i.e., along routes which depend on the application [110], and which are not necessarily shortest paths or destination-based, or not even loop-free [56, 155, 95, 153].

These trends enable the realization of interesting new in-network communication services called *service chains* [94, 154, 148, 168]: sequences of network functions which are allocated and stitched together in a flexible manner. For example, a service chain c_i could define that traffic originating at source s_i is first steered through an intrusion detection system for security (1^{st} network function), next through a traffic optimizer (2^{nd} network function), and only then is routed towards the destination t_i. Such advanced network services open an interesting new market for Internet Service Providers, which can become "miniature cloud providers" [169], specialized for in-network processing.

In this section we present our results on incremental middlebox deployment [38] and online servive chain embedding [41].

5.1 On-line admission control and embedding of service chains

The virtualization and softwarization of modern computer networks enables the definition and fast deployment of novel network services called *service chains*: sequences of virtualized network functions (e.g., firewalls, caches, traffic optimizers) through which traffic is routed between source and destination. In [41] we

attend to the problem of admitting and embedding a maximum number of service chains, i.e., a maximum number of source-destination pairs which are routed via a sequence of ℓ to-be-allocated, capacitated network functions. We consider an Online variant of this maximum Service Chain Embedding Problem, short *OSCEP*, where requests arrive over time, in a worst-case manner. Our main contribution is a deterministic $O(\log \ell)$-competitive online algorithm, under the assumption that capacities are at least logarithmic in ℓ. We show that this is asymptotically optimal within the class of deterministic and randomized online algorithms. We also explore lower bounds for offline approximation algorithms, and prove that the offline problem is APX-hard for unit capacities and small $\ell \geq 3$, and even Poly-APX-hard in general, when there is no bound on ℓ. These approximation lower bounds may be of independent interest, as they also extend to other problems such as Virtual Circuit Routing. Finally, we present an exact algorithm based on 0-1 programming, implying that the general offline SCEP is in NP and, by the above hardness results, it is NP-complete for constant ℓ.

Scope: In [41], we study the problem of how to optimally admit and embed service chain requests. Given a redundant distribution of network functions and a sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)$, where each $\sigma_i = (s_i, t_i)$ for $i \in [1, k]$ defines a source-destination pair (s_i, t_i) which needs to be routed via a sequence of network function instances, we ask: Which requests σ_i to admit and where to allocate their service chains c_i? The service chain embedding should respect capacity constraints as well as constraints on the length (or stretch) of the route from s_i to t_i via its service chain c_i.

Our objective is to maximize the number of admitted requests. We are particularly interested in the *Online Service Chain Embedding Problem (OSCEP)*, where σ is only revealed over time. We assume that a request cannot be delayed and once admitted, cannot be preempted again. Sometimes, we are also interested in the general (offline) problem, henceforth denoted by SCEP.

Our contribution: We formulate the online and offline problems OSCEP and SCEP, and make the following contributions:

1. We present a deterministic online algorithm ACE^4 which, given that node capacities are at least logarithmic, achieves a competitive ratio $O(\log \ell)$ for OSCEP. This result is practically interesting, as the number of to be traversed network functions ℓ is likely to be small in practice. In our analysis, we adapt a proof strategy known from virtual circuit routing [151]. Note however that in contrast to virtual circuit routing, where the end nodes have to be connected by a path in the network, in the SCEP, the path must traverse a sequence of ℓ nodes, such that the ith node of this sequence hosts

[4] **A**dmission control and **C**hain **E**mbedding.

network function f_i. Furhermore, in the SCEP, the path length must be bounded by r hops. So far, only heuristic and offline approaches to solve the service chain embedding problem have been considered [60, 91, 140, 168].

2. We prove that the competetive ratio of our online algorithm is asymptotically optimal in the class of both deterministic and randomized online algorithms, which can be achieved in polynomial time, by adapting a proof strategy from virtual circuit routing in [54]. Moreover, we initiate the study of lower bounds for the offline version of our problem, and show that no good approximation algorithms exist, unless $P = NP$: for unit capacities and already small ℓ, the offline problem SCEP is APX-hard. For arbitrary ℓ, the problem can even become Poly-APX-hard. These results also apply to the offline version of classic online call control problems, which to the best of our knowledge have not been studied before.

3. We present a 0-1 program for SCEP, which also shows that SCEP is in NP for constant ℓ and, taking into account our hardness result, that SCEP is NP-complete for constant ℓ. More precisely, if the number of all possible chains that can be constructed over the network function instances is polynomial in the network size n, then the number of variables in the 0-1 program is also polynomial, and thus the problem is in NP. If m_i is the number of instances of network function f_i in the network, $i = 1, ..., \ell$, and $m = \max_i\{m_i\}$, then the size of the 0-1 program is polynomial for $m^\ell = \text{poly}(n)$. For example, this always holds for constant ℓ. When m is constant, then it holds for $\ell = O(\log n)$.

5.2 Middlebox Deployment

Virtualization of network function and softwareization of modern computer networks offer new opportunities for the simplified management and flexible placement of middleboxes as e.g. firewalls and proxies. In [38] we initiate the study of algorithmically exploiting the flexibilities present in virtualized and software-defined networks. Particularly, we are interested in the initial as well as the incremental deployment of middleboxes. We present a deterministic $O(\log(\min\{n, \kappa\}))$ approximation algorithm for n-node computer networks, where κ is the middlebox capacity. The algorithm is based on optimizing over a submodular function which can be computed efficiently using a fast augmenting path approach. The derived approximation bound is optimal: the underlying problem is computationally hard to approximate within sublogarithmic factors, unless $P = NP$ holds. We additionally present an exact algorithm based on integer programming, and complement our formal analysis with simulations. In particular, we consider the number of used middleboxes and highlight the benefits of the approximation algorithm in incremental deployments. Our approach

also finds interesting applications, e.g., in the context of incremental deployment of software-defined networks.

Scope: Middleboxes are ubiquitous in modern computer networks, which provide a wide spectrum of in-network functions to ensure security, performance, and policy compliance. In fact, the number of middleboxes in enterprise networks can be of the same order of magnitude as the number of routers [167].

While in-network functions were traditionally implemented in specialized hardware appliances and middleboxes, computer networks in general and middleboxes in particular become more and more software-defined and virtualized [93]: network functions can be implemented in software and deployed fast and flexibly on the virtualized network nodes. Virtualization is also attractive for its potential cost savings [157].

Modern computer networks also offer new flexibilities in terms of how traffic can be *routed* through middleboxes and virtualized data plane appliances (often called Virtual Network Functions, short *VNFs*) [156]. In fact, Openflow, the standard SDN protocol today, not only introduces a more flexible routing, but itself allows to implement basic middlebox functionality, on the switches [96]: an Openflow switch can match, and perform actions upon, not only layer-2, but also layer-3 and layer-4 header fields.

However, not much is known today about how to exploit these flexibilities algorithmically. A particularly interesting problem regards the question of where to deploy a minimal number of middleboxes such that basic routing and capacity constraints are fulfilled. Intuitively, the smaller the number of deployed network functions, the longer the routes via these functions, and a good tradeoff between deployment costs and additional latency must be found. Moreover, ideally, middleboxes should be *incrementally deployable*: when additional middleboxes are deployed, existing placements do not have to be changed. This is desirable especially in deployment scenarios with budget constraints.

Novelty and Related work: Interestingly, the middlebox deployment problem has not been studied before in the algorithms literature. As we show, the problem can be seen as a novel covering problem [78, 97, 175] where: (1) we are interested in the distance *between communicating pairs*, *via* the covering nodes, and not *to* the covering nodes; (2) we aim to support incremental deployments: middlebox locations selected earlier in time as well as the supported communication pairs should not have to be changed when deploying additional middleboxes; (3) we consider a *capacitated* setting where the number of items which can be assigned to a node is bounded by κ.

Nevertheless, we can adapt algorithmic concepts introduced in prior work. Our lower bounds build upon hardness results on uncapacitated covering problems [133], and our upper bound builds upon Wolsey's study of vertex and set

covering problems with hard capacities [175]. An elegant alternative proof to Wolsey's dual fitting approach, based on combinatorial arguments, is due to Chuzhoy and Naor [77]. The authors also show that using LP-relaxation approaches is generally difficult, as the integrality gap of a natural linear program for the weighted and capacitated vertex and set covering problems is unbounded.

Our contributions: In [38] and [39] we initiate the study of the problem of placing a minimum number of middleboxes or network functions, such that distance constraints between communicating node pairs as well as capacity constraints on the network nodes are satisfied.

Our main contribution is a deterministic and greedy $O(\log(\min\{\kappa, n\}))$-approximation algorithm for the middlebox placement problem in n-node networks where capacities are bounded by κ. Our algorithm supports both deployments *from scratch* as well as *incremental deployments*: It does not require any changes to the locations of existing middleboxes or the preemption of previously served communication pairs when additional middleboxes are deployed.

The approximation bound is optimal in the sense that, as we show, neither the capacitated nor the uncapacitated problem admits a sublogarithmic polynomial-time approximation, unless $P = NP$ holds.

We also present a (non-polynomial) optimal algorithm based on a 0-1 integer linear program which, together with our hardness result, implies that the problem is NP-complete: 0-1 programming is one of Karp's 21 NP-complete problems.

We complement our formal analysis with a simulation study, where we investigate the tradeoff between routing flexibilities (in terms of path stretch) and number of required middleboxes, and also highlight the benefit of incremental deployments.

We believe that our model and approach has ramifications beyond middlebox deployment. For instance, our algorithm can also be used to solve the problem of incremental SDN deployment [122, 138].

References

Own Publications

[1] Ádám Agócs, Zoltán Ács, Attila Balaton, and Tamás Lukovszki. Towards a faster bittorrent. *Annales Univ. Sci. Budapest., Sect. Comp.*, 37:65–80, 2012.

[2] Attila Balaton, Tamás Lukovszki, and Ádám Agócs. A new deterministic source coding method in peer-to-peer systems. In *12th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, 2011.

[3] László Blázovics and Tamás Lukovszki. Surrounding robots – a localized solution for the intruder problem. In *Proc. 3rd IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, 2012.

[4] László Blázovics and Tamás Lukovszki. Fast localized sensor self-deployment for focused coverage. In *Algorithms for Sensor Systems - 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2013, Revised Selected Papers*, pages 83–94. Springer, LNCS Vol. 8243, 2013.

[5] László Blázovics, Tamás Lukovszki, and Bertalan Forstner. Target surrounding solution for swarm robots. In *Information and Communication Technologies - 18th EUNICE/ IFIP WG 6.2, 6.6 International Conference, EUNICE*, pages 251–262. Springer, LNCS Vol. 7479, 2012.

[6] László Blázovics, Tamás Lukovszki, and Bertalan Forstner. Surrounding robots - A discrete localized solution for the intruder problem. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 18(3):315–319, 2014.

[7] Péter Eker and Tamás Lukovszki. The analysis of similarities and registration delay in phonebook centric social networks. *International Journal on Advances on Networks and Applications*, 4(1-2):199–208, 2011.

[8] Péter Ekler and Tamás Lukovszki. Calculating the total number of similarities in phonebook-centric social networks. In *Proc. Automation and Applied Computer Science Workshop (AACS), ISBN: 978-963-420-978-2*, 2009.

[9] Péter Ekler and Tamás Lukovszki. Learning methods for similarity handling in phonebook-centric social networks. In *10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics (CINTI)*, 2009.

[10] Péter Ekler and Tamás Lukovszki. Similarity distribution in phonebook-centric social networks. In *Proc. 5th International Conference on Wireless and Mobile Communications (ICWMC)*, pages 359–364, 2009.

[11] Péter Ekler and Tamás Lukovszki. The accuracy of power law based similarity model in phonebook-centric social networks. In *Proc. 6th International Conference on Wireless and Mobile Communications (ICWMC)*, pages 209–214, 2010.

[12] Péter Ekler and Tamás Lukovszki. The accuracy of similarity calculation of phonebook-centric social networks. In *Proc. 11th International Carpathian Control Conference (ICCC)*, 2010.

[13] Péter Ekler and Tamás Lukovszki. Examining registration delays in mobile based social networks. In *11th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, 2010.

[14] Péter Ekler and Tamás Lukovszki. Experiences with network coding in mobile bittorrent environment. In *Proc. 24th MicroCAD Conference, ISBN: 978-963-661-925-1*, 2010.

[15] Péter Ekler and Tamás Lukovszki. Experiences with phonebook-centric social networks. In *Proc. 7th IEEE Consumer Communications and Networking Conference (CCNC)*, pages 412–416, 2010.

[16] Péter Ekler and Tamás Lukovszki. Modelling the variance of power law distrbution considering phonebook-centric social networks. In *Proc. Automation and Applied Computer Science Workshop (AACS)*, 2010.

[17] Péter Ekler and Tamás Lukovszki. Extending mobile bittorrent environment with network coding. In *Proc. 8th IEEE Consumer Communications and Networking Conference (CCNC)*, pages 529–530, 2012.

[18] Péter Ekler, Tamás Lukovszki, and Hassan Charaf. Evaluating dynamically evolving mobile-based social networks. *Acta Cybernetica*, 19(4):735–748, 2010.

[19] Matthias Fischer, Tamás Lukovszki, and Martin Ziegler. Geometric searching in walkthrough animations with weak spanners in real time. In *Algorithms - ESA '98, 6th Annual European Symposium*, pages 163–174. Springer, LNCS Vol. 1461, 1998.

[20] Sathish Govindarajan, Tamás Lukovszki, Anil Maheshwari, and Norbert Zeh. I/O-efficient well-separated pair decomposition and its applications. In *Algorithms - ESA 2000, 8th Annual European Symposium*, pages 220–231. Springer, LNCS Vol. 1879, 2000.

[21] Sathish Govindarajan, Tamás Lukovszki, Anil Maheshwari, and Norbert Zeh. I/O-efficient well-separated pair decomposition and applications. *Algorithmica*, 45(4):585–614, 2006.

[22] Matthias Grünewald, Tamás Lukovszki, Christian Schindelhauer, and Klaus Volbert. Distributed maintenance of resource efficient wireless network topologies (distinguished paper). In *Euro-Par 2002, Parallel Processing, 8th International Euro-Par Conference*, pages 935–946. Springer, LNCS Vol. 2400, 2002.

[23] Attila Hideg, Blázovics László, Tamás Lukovszki, and Bertalan Forstner. Uniform dispersal of cheap flying robots in the presence of obstacles. *Acta Polytechnica Hungarica*, 18(1):13–28, 2021.

[24] Attila Hideg and Tamás Lukovszki. Uniform dispersal of robots with minimum visibility range. In *13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2017, Revised Selected Papers*, pages 155–167. Springer, LNCS 10718, 2017.

[25] Attila Hideg and Tamás Lukovszki. Asynchronous filling by myopic luminous robots. In *16th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2020, Revised Selected Papers*, pages 108–123. Springer, LNCS 12503, 2020.

[26] Attila Hideg, Tamás Lukovszki, and B. Forster. Filling arbitrary connected areas by silent robots with minimal visibility range. In *14th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2018, Revised Selected Papers*, pages 193–205. Springer, LNCS 11410, 2018.

[27] Attila Hideg, Tamás Lukovszki, and Bertalan Forster. Uniform dispersal of silent oblivious robots. In *IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 73–76. IEEE, 2017.

[28] Attila Hideg, Tamás Lukovszki, and Bertalan Forstner. Improved runtime for the synchronous multi-door filling. *Periodica Polytechnica Electrical Engineering and Computer Science*, 66(1):12–19, 2021.

[29] Imre Kelényi, Jukka K. Nurminen, Akos Ludanyi, and Tamás Lukovszki. Modeling resource constrained bittorrent proxies for energy efficient mobile content sharing. *Peer-to-Peer Networking and Applications*, 5(2):163–177, 2012.

[30] Sándor Laki and Tamás Lukovszki. Balanced neighbor selection for bittorrent-like networks. In *Algorithms - ESA 2013 - 21st Annual European Symposium*, pages 659–670. Springer, LNCS Vol. 8125, 2013.

[31] Sándor Laki and Tamás Lukovszki. On a balanced neighbor selection strategy for tracker-based peer-to-peer networks. In *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013*, pages 1–9, 2013.

[32] Ulrich Lauther and Tamás Lukovszki. Space efficient algorithms for the Burrows-Wheeler backtransformation. In *Algorithms - ESA 2005, 13th Annual European Symposium*, pages 293–304. Springer, LNCS Vol. 3669, 2005.

[33] Ulrich Lauther and Tamás Lukovszki. Method for carrying out an inverse Burrows-Wheeler transform with efficient memory use / Verfahren zur speichereffizienten Durchführung einer Burrows-Wheeler-Rücktransformation. In *International Patent PCT/EP2006/065263*, 2006.

[34] Ulrich Lauther and Tamás Lukovszki. Space efficient algorithms for the Burrows-Wheeler backtransformation. *Algorithmica*, 58(2):339–351, 2010.

[35] Akos Ludanyi, Tamás Lukovszki, and Péter Ekler. Simulating network coding for accelerating tit-for-tat in peer-to-peer content sharing. In *Information and Communication Technologies - 18th EUNICE/ IFIP WG 6.2, 6.6 International Conference, EUNICE*, pages 408–411. Springer, LNCS Vol. 7479, 2012.

[36] Tamás Lukovszki, Anil Maheshwari, and Norbert Zeh. I/O-efficient batched range counting and its applications to proximity problems. In *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science, 21st Conference*, pages 244–255. Springer, LNCS Vol. 2245, 2001.

[37] Tamás Lukovszki and Friedhelm Meyer auf der Heide. Fast collisionless pattern formation by anonymous, position-aware robots. In *Principles of Distributed Systems - Proc. 18th International Conference, OPODIS*, pages 248–262. Springer, LNCS Vol. 8878, 2014.

[38] Tamás Lukovszki, Matthias Rost, and Stefan Schmid. It's a match!: Near-optimal and incremental middlebox deployment. *ACM SIGCOMM Computer Communications Review (CCR)*, 46(1):30–36, 2016.

[39] Tamás Lukovszki, Matthias Rost, and Stefan Schmid. Approximate and incremental network function placement. *Journal of Parallel and Distributed Computing (JPDC)*, 120:159–169, 2018.

[40] Tamás Lukovszki, Christian Schindelhauer, and Klaus Volbert. Resource efficient maintenance of wireless network topologies. *J. UCS Journal of Universal Computer Science*, 12(9):1292–1311, 2006.

[41] Tamás Lukovszki and Stefan Schmid. Online admission control and embedding of service chains. In *Structural Information and Communication Complexity - 22nd International Colloquium, SIROCCO, Post-Proceedings*, pages 104–118. Springer, LNCS Vol. 9439, 2015.

[42] Christian Schindelhauer, Tamás Lukovszki, Stefan Rührup, and Klaus Volbert. Worst case mobility in ad hoc networks. In *SPAA 2003: Proc. 15th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 230–239, 2003.

Other References

[43] Bittorrent [online], available: http://bittorrent.com.

[44] Tribler [online], available: http://www.tribler.org.

[45] J. Abel. Grundlagen des Burrows-Wheeler-Kompressionsalgorithmus (in german). *Informatik - Forschung und Entwicklung*, 2003. http://www.data-compression.info/JuergenAbel/Preprints/Preprint_Grundlagen_BWCA.pdf.

[46] Susanne Albers, Klaus Kursawe, and Sven Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32(1):123–143, 2002.

[47] Aisha Aljohani, Pavan Poudel, and Gokarna Sharma. Complete visitability for autonomous robots on graphs. In *IPDPS 2018*, pages 733–742, 2018.

[48] Z. Arnavut. Generalization of the BWT transformation and inversion ranks. In *Proc. IEEE Data Compression Conference (DCC)*, page 447, 2002.

[49] D. Arthur and R. Panigrahy. Analyzing bittorrent and related peer-to-peer networks. In *Proc. 17th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 961–969, 2006.

[50] S. Arya, G. Das, D. Mount, J. Salowe, and M. Smid. Euclidean spanners: short thin and lanky. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 489–498, 1995.

[51] M.J. Atallah, editor. *Algorithms and Theory of Computation Handbook*. CRC Press, 1999.

[52] John Augustine and William K. Moses, Jr. Dispersion of mobile robots: A study of memory-time trade-offs. In *Proc. 19th Int. Conf. on Distributed Computing and Networking, ICDCN 2018*, pages 1:1–1:10, 2018.

[53] B. Awerbuch, P. Berenbrink, A. Brinkmann, and C. Scheideler. Simple Routing Strategies for Adversarial Systems. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 158–167, 2001.

[54] Baruch Awerbuch, Yossi Azar, and Serge A. Plotkin. Throughput-competitive on-line routing. In *Proc. 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 32–40, 1993.

[55] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.

[56] M. Bagaa, T. Taleb, and A. Ksentini. Service-aware network function placement for efficient traffic handling in carrier cloud. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2402–2407, 2014.

[57] Xiaole Bai, Santosh Kumar, Dong Xuan, Ziqiu Yun, and Ten-Hwang Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *MobiHoc*, pages 131–142, 2006.

[58] B. Balkenhol and S. Kurtz. Universal data compression based on the Burrows-Wheeler transformation: Theory and practice. *IEEE Trans. on Computers*, 23(10):1043–1053, 2000.

[59] Evangelos Bampas, Leszek Gasieniec, Nicolas Hanusse, David Ilcinkas, Ralf Klasing, Adrian Kosowski, and Tomasz Radzik. Robustness of the rotor-router mechanism. *Algorithmica*, 78(3):869–895, 2017.

[60] Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, and Raouf Boutaba. On orchestrating virtual network functions in NFV. *CoRR*, 2015.

[61] Eduardo Mesa Barrameda, Shantanu Das, and Nicola Santoro. Deployment of asynchronous robotic sensors in unknown orthogonal environments. In *Algorithmic Aspects of Wireless Sensor Networks, Fourth International Workshop, (ALGOSENSORS), Revised Selected Papers*, pages 125–140. Springer, LNCS Vol. 5389, 2008.

[62] Eduardo Mesa Barrameda, Shantanu Das, and Nicola Santoro. Deployment of asynchronous robotic sensors in unknown orthogonal environments. In *Algorithmic Aspects of Wireless Sensor Networks, 4th Int. Workshop, ALGOSENSORS 2008, Revised Selected Papers*, volume 5389 of *LNCS*, pages 125–140. Springer, 2008.

[63] Eduardo Mesa Barrameda, Shantanu Das, and Nicola Santoro. Uniform dispersal of asynchronous finite-state mobile robots in presence of holes. In *Algorithms for Sensor Systems - 9th Int. Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2013, Revised Selected Papers*, volume 8243 of *LNCS*, pages 228–243. Springer, 2013.

[64] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri. Snap and spread: A self-deployment algorithm for mobile sensor networks. In *Proc. 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 451–456. Springer-Verlag, 2008.

[65] Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic (ed.). *Mobile Ad Hoc Networking*. Wiley-IEEE Press, 2004.

[66] J. Basch, L. J. Guibas, and J. Hershberger. Data Structures for Mobile Data. *Journal of Algorithms*, 31(1):1–28, 1999.

[67] J. L. Bentley and R. Sedgewick. Fast algorithms for sorting and searching strings. In *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 360–369, 1997.

[68] C. Bettstetter. Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks. In *ACM Int. Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM'01)*, pages 19–27, 2001.

[69] Subhash Bhagat and Krishnendu Mukhopadhyaya. Optimum algorithm for mutual visibility among asynchronous robots with lights. In *SSS 2017*, pages 341–355. LNCS 10616, 2017.

[70] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Understanding and Deconstructing BitTorrent Performance. Technical Report MSR-TR-2005-03, Microsoft Research, 2005.

[71] Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary, and Buddhadeb Sau. Arbitrary pattern formation by asynchronous opaque robots with lights. In *SIROCCO 2019*, pages 109–123. LNCS 11639, 2019.

[72] Peter Brass. Bounds on coverage and target detection capabilities for models of networks of mobile sensors. *ACM Trans. Sen. Netw.*, 3(2), 2007.

[73] Francesco Bullo, Jorge Cortés, and Sonia Martínez. Distributed algorithms for robotic networks. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 7712–7728. Springer, 2009.

[74] S. Burkhardt and J. Kärkkäinen. Fast lightweight suffix array construction and checking. In *Proc. 14th Symposium on Combinatorial Pattern Matching (CPM)*, pages 55–69, 2003.

[75] M. Burrows and D.J. Wheeler. A block-sorting lossless data compression algorithm. *Technical report 124, Digital Equipment Corporation*, 1994. http://gatekeeper.research. compaq.com/pub/DEC/SRC/research-reports/abstracts/src-rr-124.html.

[76] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[77] Julia Chuzhoy and Joseph Naor. Covering problems with hard capacities. *SIAM J. Comput.*, 36(2):498–515, 2006.

[78] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[79] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by mobile robots: Gathering. *SIAM J. Comput.*, 41(4):829–879, 2012.

[80] Reuven Cohen and David Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM J. Comput.*, 34(6):1516–1528, June 2005.

[81] Reuven Cohen and David Peleg. Local spreading algorithms for autonomous robot systems. *Theoretical Computer Science*, 399(1):71 – 82, 2008.

[82] Andreas Cord-Landwehr, Bastian Degener, Matthias Fischer, Martina Hüllmann, Barbara Kempkes, Alexander Klaas, Peter Kling, Sven Kurras, Marcus Märtens, Friedhelm Meyer Auf Der Heide, Christoph Raupach, Kamil Swierkot, Daniel Warner, Christoph Weddemann, and Daniel Wonisch. Collisionless gathering of robots with an extent. In *Proceedings of the 37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2011)*, pages 178–189. Springer Verlag, LNCS, 2011.

[83] Andreas Cord-Landwehr, Bastian Degener, Matthias Fischer, Martina Hüllmann, Barbara Kempkes, Alexander Klaas, Peter Kling, Sven Kurras, Marcus Märtens, Friedhelm Meyer auf der Heide, Christoph Raupach, Kamil Swierkot, Daniel Warner, Christoph Weddemann, and Daniel Wonisch. A new approach for analyzing convergence algorithms for mobile robots. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, pages 650–661. Springer Verlag, LNCS Vol. 6756, 2011.

[84] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *Robotics and Automation, IEEE Transactions on*, 20(2):243–255, 2004.

[85] A. Czumaj and C. Sohler. Soft Kinetic Data Structures. In *Symposium on Discrete Algorithms (SODA)*, pages 865–872, 2001.

[86] Jurek Czyzowicz, Dariusz Dereniowski, Leszek Gasieniec, Ralf Klasing, Adrian Kosowski, and Dominik Pajak. Collision-free network exploration. *J. Comput. Syst. Sci.*, 86:70–81, 2017.

[87] Jurek Czyzowicz, Leszek Gasieniec, and Andrzej Pelc. Gathering few fat mobile robots in the plane. *Theor. Comput. Sci.*, 410(6-7):481–499, 2009.

[88] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: Synchronizing asynchronous robots using visible bits. In *ICDCS 2012*, pages 506–515, 2012.

[89] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. Autonomous mobile robots with lights. *Theoretical Computer Science*, 609:171–184, 2016.

[90] Mattia D'Emidio, Daniele Frigioni, and Alfredo Navarra. Synchronous robots vs asynchronous lights-enhanced robots on graphs. *Electronic Notes in Theoretical Computer Science*, 322:169–180, 2016.

[91] D. Dietrich, A. Abujoda, and P. Papadimitriou. Network service embedding across multiple providers with nestor. In *Proc. IFIP Networking*, 2015.

[92] K. Erciyes. *Distributed Graph Algorithms for Computer Networks*. Series: Computer Communications and Networks, Springer Verlag, 2013.

[93] A. Gember-Jacobson et al. OpenNF: Enabling innovation in network function control. In *Proc. ACM SIGCOMM*, 2014.

[94] ETSI. Network functions virtualisation (nfv); use cases. *WWW*, 2014.

[95] Seyed Kaveh Fayazbakhsh, Vyas Sekar, Minlan Yu, and Jeffrey C. Mogul. Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions. In *Proc. ACM HotSDN*, 2013.

[96] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to SDN. *Queue*, 11(12), 2013.

[97] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[98] Caterina Feletti, Carlo Mereghetti, and Beatrice Palano. Uniform circle formation for swarms of opaque robots with lights. In *SSS 2018*, pages 317–332. LNCS 11201, 2018.

[99] P. Fenwick. Block sorting text compression – final report. Technical report, Department of Computer Science, The University of Auckland, 1996. ftp://ftp.cs.auckland.ac.nz/ pub/staff/peter-f/TechRep130.ps.

[100] P. Ferragina and G. Manzini. An experimental study of an opportunistic index. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 269–278, 2001.

[101] P. Ferragina and G. Manzini. Compression boosting in optimal linear time using the Burrows-Wheeler transform. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 655–663, 2004.

[102] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.

[103] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.

[104] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *LNCS*. Springer, 2019.

[105] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theor. Comput. Sci.*, 407(1-3):412–447, 2008.

[106] Paola Flocchini, Nicola Santoro, and Koichi Wada. On memory, communication, and synchronous schedulers when moving and computing. In *OPODIS 2019*, pages 25:1–25:17, 2019.

[107] D. W. Gage. *Command control for many-robot systems*. Naval Command Control and Ocean Surveilance Center RDT and E Div San Diego CA, 1992.

[108] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete Mobile Centers. In *Proc. of the 17th Symposium on Computational Geometry (SoCG)*, pages 188–196, 2001.

[109] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric Spanner for Routing in Mobile Networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBICOM'01)*, pages 45–55, 2001.

[110] Arpit Gupta, Laurent Vanbever, Muhammad Shahbaz, Sean P. Donovan, Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, and Ethan Katz-Bassett. Sdx: A software defined internet exchange. In *Proc. ACM SIGCOMM*, pages 551–562, 2014.

[111] F. V. Hecht, T. Bocek, and B. Stiller. B-tracker: Improving load balancing and efficiency in distributed p2p trackers. In *Proc. 11th IEEE Int. Conf. on Peer-to-Peer Computing (P2P)*, pages 310 –313, 2011.

[112] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A Group Mobility Model for Ad Hoc Wireless Networks. In *Proc. of the 2nd ACM int. workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 53–60, 1999.

[113] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Auton. Robots*, 13(2):113–126, 2002.

[114] Tien-Ruey Hsiang, Esther M. Arkin, Michael A. Bender, Sándor P. Fekete, and Joseph S. B. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. *Algorithmic Foundations of Robotics V, Springer Tracts in Advanced Robotics*, 7:77–93, 2004.

[115] H. Itoh and H. Tanaka. An efficient method for in-memory construction of suffix arrays. In *Proc. 6th International Sympisium on String Processing and Information Retrieval (SPIRE)*, pages 81–88, 1999.

[116] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice. Dissecting bittorrent: Five months in a torrent's lifetime. In *PAM '04*, pages 1–11, 2004.

[117] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[118] Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, Sebastien Tixeuil, and Koichi Wada. Gathering on rings for myopic asynchronous robots with lights. In *OPODIS 2019*, pages 27:1–27:17, 2019.

[119] Holger Karl and Andreas Willig. *Protocols and architectures for wireless sensor networks*. Wiley, 2005.

[120] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete and Computational Geometry*, 7:13–28, 1992.

[121] N. J. Larsson and K. Sadakane. Faster suffix sorting. *Theoretical Computer Science*, 387(3):258–272, 2007.

[122] Dan Levin, Marco Canini, Stefan Schmid, Fabian Schaffert, and Anja Feldmann. Panopticon: Reaping the benefits of incremental sdn deployment in enterprise network. In *Proc. USENIX ATC*, 2014.

[123] X.-Y. Li, P.-J. Wan, and Y. Wang. Power efficient and sparse spanner for wireless ad hoc networks. In *Proc. IEEE International Conference on Computer Communications and Networks (ICCCN)*, pages 264–267, 2001.

[124] X.-Y. Li, P.-J. Wan, Y. Wang, and O. Frieder. Sparse power efficient topology for wireless networks. In *Proc. Hawaii International Conference on System Sciences (HICSS)*, 2002.

[125] Xiang-Yang Li and Yu Wang. *Applications of Computational Geometry in Wireless Ad Hoc Networks*. Book Chapter of Ad Hoc Wireless Networking, edited by XiuZhen Cheng, Xiao Huang, and Ding-Zhu Du, Springer, 2004.

[126] Xiang-Yang Li and Yu Wang. *Wireless Sensor Networks and Computational Geometry*. Book Chapter of Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, edited by Mohammad Ilyas et al., CRC Press, 2004.

[127] Xu Li, H. Frey, N. Santoro, and I. Stojmenovic. Localized sensor self-deployment for guaranteed coverage radius maximization. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1 –5, 2009.

[128] Xu Li, Hannes Frey, Nicola Santoro, and Ivan Stojmenovic. Focused-coverage by mobile sensor networks. In *6th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 466–475, 2009.

[129] Xu Li, Hannes Frey, Nicola Santoro, and Ivan Stojmenovic. Strictly localized sensor self-deployment for optimal focused coverage. *IEEE Trans. Mob. Comput.*, 10(11):1520–1533, 2011.

[130] B. Liang and Z. J. Haas. Predictive Distance-Based Mobility Management for PCS Networks. In *Proceedings of IEEE INFOCOM'99*, pages 1377–1384, 1999.

[131] Thomas Locher, Stefan Schmid, and Roger Wattenhofer. Rescuing tit-for-tat with source coding. In *Proc. 7th IEEE Int. Conf. on Peer-to-Peer Computing (P2P)*, pages 3–10, 2007.

[132] G.A. Di Luna, P. Flocchini, S. Gan Chaudhuri, F. Poloni, N. Santoro, and G. Viglietta. Mutual visibility by luminous robots without collisions. *Information and Computation*, 254(3):392–418, 2017.

[133] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[134] Ming Ma and Yuanyuan Yang. Adaptive triangular deployment algorithm for unattended mobile sensor networks. *IEEE Trans. Computers*, 56(7), 2007.

[135] U. Manber and E. Meyers. Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 22:935–948, 1993.

[136] G. Manzini. An analysis of the Burrows-Wheeler transform. *Journal of the ACM*, 48(3):407–430, 2001.

[137] G. Manzini and P. Ferragina. Engineering a lightweight suffix array construction algorithm. *Algorithmica*, 40(1):33–50, 2004.

[138] Michael Markovitch and Stefan Schmid. Shear: A highly available and flexible network architecture: Marrying distributed and logically centralized control planes. In *Proc. 23rd IEEE International Conference on Network Protocols (ICNP)*, 2015.

[139] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[140] Sevil Mehraghdam, Matthias Keller, and Holger Karl. Specifying and placing chains of virtual network functions. In *Proc. 3rd IEEE International Conference on Cloud Networking (CloudNet)*, pages 7–13, 2014.

[141] F. Meyer auf der Heide, C. Schindelhauer, K. Volbert, and M. Grünewald. Energy, congestion and dilation in radio networks. *Theory of Computing Systems (TOCS)*, 37:343–370, 2004.

[142] R. Motwani and P. Raghavan. Randomized algorithms. *ACM Comput. Surv.*, 28(1):33–37, 1996.

[143] Joong Chae Na. Linear-time construction of compressed suffix arrays using $o(nlogn)$-bit working space for large alphabets. In *Proc. 16th Symposium on Combinatorial Pattern Matching (CPM '05)*, pages 56–67, 2005.

[144] G. Navarro and V. Mäkinen. Compressed full text indexes. *ACM Computing Surveys*, 39(1), 2007.

[145] Amiya Nayak and Ivan Stojmenovic. *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. Wiley-Interscience, New York, NY, USA, 2010.

[146] M. Nelson. Data compression with the Burrows-Wheeler transform. *Dr. Dobb's Journal*, 9, 1996.

[147] Fukuhito Ooshita and Sebastien Tixeuil. Ring exploration with myopic luminous robots. In *SSS 2018*, pages 301–316. LNCS 11201, 2018.

[148] P. Skoldstrom et al. Towards unified programmability of cloud and carrier infrastructure. In *Proc. European Workshop on Software Defined Networking (EWSDN)*, 2014.

[149] David Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In *IWDC 2005*, pages 1–12. LNCS 3741, 2005.

[150] M. Picone, M. Amoretti, and F. Zanichelli. An evaluation criterion for adaptive neighbor selection in heterogeneous peer-to-peer networks. In *Proc. 12th IFIP/IEEE Int. Conference on Management of Multimedia and Mobile Networks and Services: Wired-Wireless Multimedia Networks and Services Management*, pages 144–156, 2009.

[151] Serge A. Plotkin. Competitive routing of virtual circuits in ATM networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1128–1136, 1995.

[152] S. J. Puglisi, W. F. Smyth, and A. H. Turpin. A taxonomy of suffix array construction algorithms. *ACM Computing Surveys*, 39(2), 2007.

[153] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu:. SIMPLE-fying middlebox policy enforcement using SDN. In *Proc. ACM SIGCOMM*, pages 27–38, 2013.

[154] R. Hartert et al. Declarative and expressive approach to control forwarding paths in carrier-grade networks. In *Proc. ACM SIGCOMM*, 2015.

[155] W. Rankothge, Jiefei Ma, F. Le, A. Russo, and J. Lobo. Towards making network function virtualization a cloud computing service. In *Proc. IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 89–97, 2015.

[156] Saqib Raza, Guanyao Huang, Chen-Nee Chuah, Srini Seetharaman, and Jatinder Pal Singh. Measurouting: A framework for routing assisted traffic monitoring. *IEEE/ACM Trans. Netw.*, 20(1):45–56, 2012.

[157] Ted Ritter. Network-based firewall services: Extending the firewall into the cloud. In *Nemertes White Paper N0496*, 2009.

[158] J. Ruppert and R. Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proc. Canadian Conference on Computational Geometry (CCCG)*, pages 207–210, 1991.

[159] C. Scheideler. *Overlay Networks for Wireless Systems*. Book Chapter of Performance Analysis of Mobile and Ad Hoc Networks, Wireless Networks and Mobile Computing Series, Vol. 7, edited by Chansu Yu, Nova Science Publishers, 2007.

[160] Vyas Sekar, Sylvia Ratnasamy, Michael K. Reiter, Norbert Egi, and Guangyu Shi. The middlebox manifesto: Enabling innovation in middlebox deployment. In *Proc. HotNets*, pages 21:1–21:6, 2011.

[161] J. Seward. Bzip2 manual, http://www.bzip.org/1.0.3/bzip2-manual-1.0.3.html.

[162] J. Seward. On the performance of BWT sorting algorithms. In *Proc. IEEE Data Compression Conference (DCC '00)*, pages 173–182, 2000.

[163] J. Seward. Space-time tradeoffs in the inverse B-W transform. In *Proc. IEEE Data Compression Conference (DCC '01)*, pages 439–448, 2001.

[164] Gokarna Sharma, Ramachandran Vaidyanathan, and Jerry L. Trahan. Constant-time complete visibility for asynchronous robots with lights. In *SSS 2017*, pages 265–281. LNCS 10616, 2017.

[165] Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L. Trahan, Costas Busch, and Suresh Rai. Complete visibility for robots with lights in O(1) time. In *SSS 2016*, pages 327–345. LNCS 10083, 2016.

[166] Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L. Trahan, Costas Busch, and Suresh Rai. O(log N)-time complete visibility for asynchronous robots with lights. In *IPDPS 2017*, pages 513–522, 2017.

[167] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. Making middleboxes someone else's problem: Network processing as a cloud service. In *Proc. ACM SIGCOMM*, pages 13–24, 2012.

[168] Robert Soulé, Shrutarshi Basu, Parisa Jalili Marandi, Fernando Pedone, Robert Kleinberg, Emin Gun Sirer, and Nate Foster. Merlin: A language for provisioning network resources. In *Proc. 10th ACM International on Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 213–226, 2014.

[169] Radu Stoenescu, Matei Popovici, Vladimir Olteanu, Joao Martins, Roberto Bifulco, Felipe Huici, Mohamed Ahmed, Georgios Smaragdakis, Mark Handley, and Costin Raiciu. In-net: Enabling in-network processing for the masses. In *Proc. ACM EuroSys*, 2015.

[170] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots. In *Proc. 3rd International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 313–330, 1996.

[171] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999.

[172] H. Wang, J. Liu, B. Chen, K. Xu, and Z. Ma. On tracker selection for peer-to-peer traffic locality. In *Proc. 10th IEEE Int. Conf. on Peer-to-Peer Computing (P2P'10)*, pages 1 –10, 2010.

[173] Yu Wang. *Topology Control for Wireless Sensor Networks*. Book Chapter of Wireless Sensor Networks and Applications, Series: Signals and Communication Technology, edited by Li, Yingshu; Thai, My T.; Wu, Weili, Springer-Verlag, 2008.

[174] Yu Wang. *Three-Dimensional Wireless Sensor Networks: Geometric Approaches for Topology and Routing Design*. Book Chapter of The Art of Wireless Sensor Networks, Series: Signals and Communication Technology, pp 367-409, edited by Habib M. Ammari, Springer-Verlag, 2014.

[175] L.A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[176] Yukiko Yamauchi and Masafumi Yamashita. Pattern formation by mobile robots with limited visibility. In *20th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Revised Selected Papers*, pages 201–212. Springer, LNCS Vol. 8179, 2013.

[177] Shuhui Yang, Minglu Li, and Jie Wu. Scan-based movement-assisted sensor deployment methods in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(8):1108–1121, 2007.

[178] A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11:721–736, 1982.