Mixtures & Monitoring
(work in progress)

Dennis Schieferdecker (schiefer@ira.uka.de)

ITI Sanders, University of Karlsruhe (TH)

04.12.2008
# Table of Contents

## 1 Gaussian Mixture Reduction
- Introduction
- Clustering Algorithm
- Results

## 2 Area Monitoring
- Motivation
- Problem Formulation
- Approximation Algorithm

## 3 Conclusion
Gaussian Mixture Reduction

approximating functions
Introduction

Motivation

Gaussian Mixtures (GMs)

- weighted sum of gaussians: \( G = \sum_{i=1}^{N} w_i \cdot g_i \)
- universal function approximator
- versatile applications:
  - machine learning
  - density estimation
  - ...

Problems in Application

- recursive processing of GMs
- number of components grows rapidly (exponential)
**Introduction**

**Motivation**

**Gaussian Mixtures (GMs)**

- weighted sum of gaussians: \( G = \sum_{i=1}^{N} w_i \cdot g_i \)
- universal function approximator
- versatile applications:
  - machine learning
  - density estimation
  - ...

**Problems in Application**

- recursive processing of GMs
- number of components grows rapidly (exponential)
**Introduction**

**Motivation**

**Gaussian Mixtures (GMs)**

- weighted sum of gaussians: \( G = \sum_{i=1}^{N} w_i \cdot g_i \)
- universal function approximator
- versatile applications:
  - machine learning
  - density estimation
  - ...

**Problems in Application**

- recursive processing of GMs
- number of components grows rapidly (exponential)
### Introduction

#### Related Work

**Classic Algorithms**

- West, Williams, Runnalls, ...
  - top-down approaches
  - greedy merging of components
  - slow or good quality

**Modern Algorithm**

- Progressive Gaussian Mixture Reduction
  - bottom-up method
  - successive construction of approximated mixture
  - average fast and better quality
Introduction

Related Work

Classic Algorithms

- West, Williams, Runnalls, ...
- top-down approaches
- greedy merging of components
- slow or good quality

Modern Algorithm

Progressive Gaussian Mixture Reduction

- bottom-up method
- successive construction of approximated mixture
- average fast and better quality
**Clustering Algorithm**

*Problem formulation*

---

**Gaussian Mixture Reduction**

- given:
  - gaussian mixture $G = \sum_{i=1}^{N} w_i \cdot g_i$

- wanted:
  - approximation $A = \sum_{j=1}^{K} w_j \cdot a_j$
  - minimizing a distance measure to $G$ (i.e. ISD)

(joint work with Marco Huber at ISAS)
**Clustering Algorithm**

**Initial Ideas**

**Approach with Algorithmics - P.o.V.**

Idea: clustering / facility location

- Consider $g_1..N$, $a_1..K$ as points on a plane
- Separate $g_1..N$ into $K$ groups (clustering), each with an associated cluster center $a_j$

Goal:

- Minimize distance measure $d(g_i, a_j)$ within each group $j$ for all group members $g_i$
**Clustering Algorithm**

*Initial Ideas*

**Approach with Algorithmics - P.o.V.**

**Idea:** clustering / facility location

- Consider $g_1..N, a_1..K$ as points on a plane
- Separate $g_1..N$ into $K$ groups (clustering), each with an associated cluster center $a_j$

**Goal:**

- Minimize distance measure $d(g_i, a_j)$ within each group $j$ for all group members $g_i$
Clustering Algorithm

Description of the algorithm - 1

Initialisation

- generate initial approximation \( A = \{a_j\} \)
- yields initial cluster centers
  (here: using Runnalls’ algorithm)
- compute initial assignments of Gaussians \( g_1 \ldots N \)
  to cluster centers \( a_1 \ldots K \): \( \text{center(\cdot)}: g_i \rightarrow a_j \)
  (here: using Kullback-Leibler discrimination)

Clustering algorithm

- loosely based on Llyod’s algorithm
Clustering Algorithm

Description of the algorithm - 2

Program Flow

- iterate through all Gaussians $g_1..N$, for each
  - compute distance $d(g_i, a_j)$ to all centers $a_1..K$
  - assign to center $a_j$ with minimal distance

- update cluster center $a_j$
  - Gaussian with weight, mean and variance equal to $\sum_{center(g_i=1..N)=a_j} w_i \cdot g_i$

Comment

- Iteration order is important!
**Clustering Algorithm**

*Description of the algorithm - 2*

**Program Flow**

- **iterate** through all Gaussians $g_{1..N}$, for each
  - compute distance $d(g_i, a_j)$ to all centers $a_{1..K}$
  - assign to center $a_j$ with minimal distance

- **update** cluster center $a_j$
  - Gaussian with weight, mean and variance equal to $\sum_{\text{center}(g_i=1..N)=a_j} w_i \cdot g_i$

**Comment**

- Iteration order is important!
**Clustering Algorithm**

*Description of the algorithm - 2*

**Program Flow**

- **iterate** through all Gaussians $g_1..N$, for each
  - compute distance $d(g_i, a_j)$ to all centers $a_1..K$
  - assign to center $a_j$ with minimal distance

- update cluster center $a_j$
  - Gaussian with weight, mean and variance equal to $\sum_{center(g_i=1..N)=a_j} w_i \cdot g_i$

**Comment**

- Iteration order is important!
**Clustering Algorithm**

*Description of the algorithm - 2*

**Program Flow**
- iterate through all Gaussians $g_1..N$, for each
  - compute distance $d(g_i, a_j)$ to all centers $a_1..K$
  - assign to center $a_j$ with minimal distance
- update cluster center $a_j$
  - Gaussian with weight, mean and variance equal to $\sum_{center(g_i=1..N)=a_j} w_i \cdot g_i$

**Comment**
- Iteration order is important!
**Clustering Algorithm**

*Description of the algorithm - 3*

**Termination**
- repeat iterating until
  - quality of approximation is adequate
  - center assignments stay constant

**Postprocessing**
- Solution usually not optimal
- apply Newton approach to improve solution
  - Clustering algorithm yields solution close to a local optimum
  - Newton approach reaches local optimum
**Clustering Algorithm**

*Description of the algorithm - 3*

### Termination
- repeat iterating until
  - quality of approximation is adequate
  - center assignments stay constant

### Postprocessing
- Solution usually **not optimal**
- apply **Newton approach** to improve solution
  - Clustering algorithm yields solution close to a local optimum
  - Newton approach reaches local optimum
**Clustering Algorithm**

*Description of the algorithm - 3*

- Distance measure $d(g_i, a_j)$ in use:
  - ISD between *original mixture density* $G$ and an approximation $A'$
  - $A'$ equals the current approximation, only $g_i$ is reassigned to center $a_j$

**ISD:** normalized Integrated Squared Distance between two functions
**Clustering Algorithm**

*Description of the algorithm - 3*

- **ISD** between original mixture density $G$ and an approximation $A'$
- $A'$ equals the current approximation, only $g_i$ is reassigned to center $a_j$

**ISD**: normalized *Integrated Squared Distance* between two functions
Clustering Algorithm

**Description of the algorithm - 3**

- **ISD** between original mixture density $G$ and an approximation $A'$
- $A'$ equals the current approximation, only $g_i$ is reassigned to center $a_j$

**ISD:** normalized Integrated Squared Distance between two functions
**Clustering Algorithm**

*Description of the algorithm - 3*

**Distance measure \(d(g_i, a_j)\) in use**

- **ISD** between original mixture density \(G\) and an approximation \(A'\)
- \(A'\) equals the current approximation, only \(g_i\) is reassigned to center \(a_j\)

**ISD**: normalized Integrated Squared Distance between two functions
**Clustering Algorithm**

*Description of the algorithm - 3*

- **ISD** between original mixture density $G$ and an approximation $A'$
- $A'$ equals the current approximation, only $g_i$ is reassigned to center $a_j$

**ISD**: normalized Integrated Squared Distance between two functions
**Clustering Algorithm**

*Description of the algorithm - 3*

**Distance measure \( d(g_i, a_j) \) in use**

- ISD between original mixture density \( G \) and an approximation \( A' \)
- \( A' \) equals the current approximation, only \( g_i \) is reassigned to center \( a_j \)

**ISD**: normalized Integrated Squared Distance between two functions
**Clustering Algorithm**

*Description of the algorithm - 3*

- **Distance measure** $d(g_i, a_j)$ in use
  - **ISD** between original mixture density $G$ and an approximation $A'$
  - $A'$ equals the current approximation, only $g_i$ is reassigned to center $a_j$

**ISD:** normalized Integrated Squared Distance between two functions
**Clustering Algorithm**

*Description of the algorithm - 3*

**Distance measure** $d(g_i, a_j)$ in use

- ISD between original mixture density $G$ and an approximation $A'$
- $A'$ equals the current approximation, only $g_i$ is reassigned to center $a_j$

**ISD:** normalized Integrated Squared Distance between two functions
**Clustering Algorithm**

*Description of the algorithm - 3*

**Distance measure** $d(g_i, a_j)$ *in use*

- **ISD** between original mixture density $G$ and an approximation $A'$
- $A'$ equals the current approximation, only $g_i$ is reassigned to center $a_j$

**ISD**: normalized Integrated Squared Distance between two functions
## Preliminary results

### Numbers

#### Running times and approximation quality

<table>
<thead>
<tr>
<th>algorithm</th>
<th>N</th>
<th>K</th>
<th>running time</th>
<th>approx. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering</td>
<td>100</td>
<td>5</td>
<td>0.79s</td>
<td>3.10 ± 2.60</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>10</td>
<td>1.09s</td>
<td>0.88 ± 0.74</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>5</td>
<td>2.37s</td>
<td>1.99 ± 0.96</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>10</td>
<td>3.04s</td>
<td>0.67 ± 0.32</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>5</td>
<td>8.15s</td>
<td>1.30 ± 0.34</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>10</td>
<td>9.53s</td>
<td>0.54 ± 0.11</td>
</tr>
<tr>
<td>West*</td>
<td>200</td>
<td>10</td>
<td>0.04s</td>
<td>3.81</td>
</tr>
<tr>
<td>Williams*</td>
<td>200</td>
<td>10</td>
<td>57.92s</td>
<td>1.03</td>
</tr>
<tr>
<td>PGMR*</td>
<td>200</td>
<td>5</td>
<td>4.78s</td>
<td>0.64</td>
</tr>
</tbody>
</table>

* results taken from [HuberHa08]
Preliminary results

Graphical Analysis

Function Plots

original gaussian mixture density (10 components)
Preliminary results

Graphical Analysis

Function Plots

Clustering approximation (7 components) - error: 1.18%
**Preliminary results**

*Graphical Analysis*

*Function Plots*

PGMR approximation (7 components) - error: 0,22%
Preliminary results

Graphical Analysis

Function Plots

comparison of PGMR and Clustering
Area Monitoring

energy-efficient, sensor-based
Motivation

Overview - 1

Area Monitoring

- **Task:**
  Permanent monitoring of an area (e.g. temperature, intrusion detection, ...)

- **Tools:**
  Wireless sensor nodes
  - limited power supply
  - more sensors spread than necessary

- **Idea:**
  Activate only as many sensors as necessary
  \(\rightarrow\) maximize lifetime
Motivation

Overview - 1

Area Monitoring

- Task:
  Permanent monitoring of an area
  (e.g. temperature, intrusion detection, ...)

- Tools:
  Wireless sensor nodes
  - limited power supply
  - more sensors spread than necessary

- Idea:
  Activate only as many sensors as necessary
  \[\rightarrow\text{maximize lifetime}\]
**Motivation**

*Overview - 1*

**Area Monitoring**

- **Task:**
  Permanent monitoring of an area
  (e.g. temperature, intrusion detection, ...)

- **Tools:**
  Wireless sensor nodes
  - limited power supply
  - more sensors spread than necessary

- **Idea:**
  Activate only as many sensors as necessary
  $ightarrow$ maximize lifetime
**Motivation**

*Overview - 1*

**Area Monitoring**

- **Task:**
  Permanent monitoring of an area
  (e.g. temperature, intrusion detection, ...)

- **Tools:**
  Wireless sensor nodes
  - limited power supply
  - more sensors spread than necessary

- **Idea:**
  Activate only as many sensors as necessary
  → maximize lifetime

![Diagram of sensor nodes and coverage areas]
Motivation

Overview - 2

Example

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$
   $\implies$ after $t_{total} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,
   then, let BC active for $t = 0.5$,
   then, let CA active for $t = 0.5$
   $\implies t_{total} = 1.5$, lifetime increased by 50%

Problem Denotation

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
Motivation

Overview - 2

Example

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$
   $\implies$ after $t_{total} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,
   then, let BC active for $t = 0.5$,
   then, let CA active for $t = 0.5$
   $\implies t_{total} = 1.5$, lifetime increased by 50%

Problem Denotation

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
**Motivation**

**Overview - 2**

**Example**

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$  
   $\rightarrow$ after $t_{total} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,  
    then, let BC active for $t = 0.5$,  
    then, let CA active for $t = 0.5$  
   $\rightarrow t_{total} = 1.5$, lifetime increased by 50%

**Problem Denotation**

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)  
(see [BermanCa04] for previous work)
**Motivation**

*Overview - 2*

**Example**

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$
   - $t_{total} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,
   - then, let BC active for $t = 0.5$,
   - then, let CA active for $t = 0.5$
   - $t_{total} = 1.5$, lifetime increased by 50%

**Problem Denotation**

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
Motivation

Overview - 2

Example

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for \( t = 1.0 \)
   \( \Rightarrow \) after \( t_{total} = 1.0 \), no further covers possible

(a) Let AB be active for \( t = 0.5 \),
   then, let BC active for \( t = 0.5 \),
   then, let CA active for \( t = 0.5 \)
   \( \Rightarrow t_{total} = 1.5 \), lifetime increased by 50%

Problem Denotation

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
**Motivation**

*Overview - 2*

**Example**

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$
   - after $t_{total} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,
   - then, let BC active for $t = 0.5$,
   - then, let CA active for $t = 0.5$
   - $t_{total} = 1.5$, lifetime increased by 50%

(b) optimal solution

**Problem Denotation**

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
**Motivation**

*Overview - 2*

---

**Example**

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$
   $\rightarrow$ after $t_{total} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,
   then, let BC active for $t = 0.5$,
   then, let CA active for $t = 0.5$
   $\rightarrow t_{total} = 1.5$, lifetime increased by 50%

---

**Problem Denotation**

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
**Motivation**

*Overview - 2*

**Example**

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$
   - after $t_{\text{total}} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,
   - then, let BC active for $t = 0.5$,
   - then, let CA active for $t = 0.5$
   - $t_{\text{total}} = 1.5$, lifetime increased by 50%

**Problem Denotation**

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
Motivation

Overview - 2

Example

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$
   $\Rightarrow$ after $t_{total} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,
   then, let BC active for $t = 0.5$,
   then, let CA active for $t = 0.5$
   $\Rightarrow t_{total} = 1.5$, lifetime increased by 50%

Problem Denotation

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
**Motivation**

*Overview - 2*

**Example**

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for $t = 1.0$
   → after $t_{total} = 1.0$, no further covers possible

(a) Let AB be active for $t = 0.5$,
   then, let BC active for $t = 0.5$,
   then, let CA active for $t = 0.5$
   → $t_{total} = 1.5$, lifetime increased by 50%

**Problem Denotation**

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
**Motivation**

*Overview - 2*

**Example**

- Sensors A, B, C with equal capacity
- 3 possible covers: AB, BC, AC

(a) Let AB be active for \( t = 1.0 \)
   - after \( t_{\text{total}} = 1.0 \), no further covers possible

(a) Let AB be active for \( t = 0.5 \)
   - then, let BC active for \( t = 0.5 \)
   - then, let CA active for \( t = 0.5 \)
   - \( t_{\text{total}} = 1.5 \), lifetime increased by 50%

**Problem Denotation**

Scheduling of nodes for Lifetime maximization of area Coverage (SLC)
(see [BermanCa04] for previous work)
**Problem Formulation**

**Model Description**

**Given:**
- arbitrary area
- \( N \) sensor nodes \( \{s_i\} \), with
  - fixed position
  - circular monitoring area with radius \( r \)
  - limited capacity \( c_i \)

**Wanted:**
- **Maximum time** \( T \), the whole area can be monitored (lifetime)
- A feasible solutions includes:
  - grouping of sensors into \( M \) covers \( \{C_j\} \), each monitoring the whole area
  - durations \( \{t_j\} \), for which each cover \( C_j \) is active (scheduling)
**Problem Formulation**

Formulation with Linear Programming (LP)

**LP formulation**

maximize: lifetime \( T \)

\[
T = \max\{1^T t | t \in \mathbb{R}^M\}
\]

subject to: limited node capacities \( \{c_i\} \)

\[
\sum_{j=1}^{M} A_{i,j} t_j \leq c_i \quad i = 1, \ldots, N
\]

- \( t_j \): duration for which cover \( C_j \) is active
- \( A_{i,j} \): 1, if node \( s_i \) in cover \( C_j \) is active, 0 otherwise
- \( c_i \): capacity of node \( s_i \)
**Problem Formulation**

Remaining problems

**Solution by CPLEX**

- **General conception:**
  problem phraseable as LP → solvable with CPLEX

- **But:**
  \( \#\text{covers} \ M \ \text{is exponential in} \ \#\text{nodes} \ N \)

- **Thus:**
  approximation algorithm required for large instances \((N > 100)\)

**Hardness of SLC**

- Problem is NP-complete
  (remains so for squared areas)

- Proof by reduction of Minimum Dominating Set
Problem Formulation

Remaining problems

Solution by CPLEX

- **General conception:**
  problem phraseable as LP → solvable with CPLEX

- **But:**
  #covers $M$ is exponential in #nodes $N$

- **Thus:**
  approximation algorithm required for large instances ($N > 100$)

Hardness of SLC

- **Problem is NP-complete**
  (remains so for squared areas)

- **Proof by reduction of Minimum Dominating Set**
Approximation Algorithm

Preliminaries

Approach

- **Relax** two attributes to provide a fast approximation algorithm
  - sensor radii \( r \)
  - maximum lifetime \( T \)

Naming Conventions

- \( T_r \): feasible solution of an SLC instance with sensor radii \( r \)
- \( T_r = opt_r \): optimal solution
**Approximation Algorithm**

*First Relaxation*

**Sensor Radii**

- Relocation of all sensor nodes to a grid of size $r \cdot \delta/2$
- Let algorithm $A$ provide an $\alpha$-approximation for this reduced problem

$A$ yields solution for the general problem with $T_r \geq \alpha \cdot \text{opt}(1 - \delta)r$

**Results**

- Possibly constant approximation factor by increasing $r$ by $1/(1 - \delta)$

Sensor nodes
Approximation Algorithm

First Relaxation

Sensor Radii

- Relocation of all sensor nodes to a grid of size $r \cdot \delta/2$
- Let algorithm $\mathcal{A}$ provide an $\alpha$-approximation for this reduced problem
  
  $\Rightarrow \mathcal{A}$ yields solution for the general problem with $T_r \geq \alpha \cdot \text{opt}(1-\delta)r$

Results

- Possibly constant approximation factor by increasing $r$ by $1/(1-\delta)$
**Approximation Algorithm**

*First Relaxation*

**Sensor Radii**

- Relocation of all sensor nodes to a grid of size $r \cdot \delta/2$
- Let algorithm $A$ provide an $\alpha$-approximation for this reduced problem

$A$ yields solution for the general problem with $T_r \geq \alpha \cdot \text{opt}(1-\delta)r$

**Results**

- Possibly constant approximation factor by increasing $r$ by $1/(1-\delta)$
**Approximation Algorithm**

**First Relaxation**

**Sensor Radii**
- Relocation of all sensor nodes to a grid of size $r \cdot \delta/2$
- Let algorithm $A$ provide an $\alpha$-approximation for this reduced problem
  \[ \rightarrow A \text{ yields solution for the general problem with } T_r \geq \alpha \cdot \text{opt}(1-\delta)r \]

**Results**
- Possibly constant approximation factor by increasing $r$ by $1/(1-\delta)$
Approximation Algorithm

First Relaxation

Sensor Radii

- Relocation of all sensor nodes to a grid of size $r \cdot \delta/2$
- Let algorithm $A$ provide an $\alpha$-approximation for this reduced problem
  \[ A \text{ yields solution for the general problem with } T_r \geq \alpha \cdot \text{opt}(1-\delta)r \]

Results

- possibly constant approximation factor by increasing $r$ by $1/(1-\delta)$
Approximation Algorithm

Second Relaxation - 1

Maximum Lifetime

- Generate tiling $\mathcal{T}$ of the area with width $k = \lceil 10/\epsilon \rceil$
- Generate shiftings $\mathcal{T}_i$ of $\mathcal{T}$ by $(i, i)$ with $i \in \mathbb{Z}_k$

Observations for $r = 1$

- each monitoring area
  - is cut by at most 2 of the tilings $\mathcal{T}_i$,
  - intersects at most 4 squares

sensor nodes
Approximation Algorithm

Second Relaxation - 1

Maximum Lifetime
- Generate tiling $\mathcal{T}$ of the area with width $k = \lceil 10/\varepsilon \rceil$
- Generate shiftings $\mathcal{T}_i$ of $\mathcal{T}$ by $(i, i)$ with $i \in \mathbb{Z}_k$

Observations for $r = 1$
- each monitoring area
  - is cut by at most 2 of the tilings $\mathcal{T}_i$,
  - intersects at most 4 squares
Approximation Algorithm

Second Relaxation - 1

Maximum Lifetime

- Generate tiling $\mathcal{T}$ of the area with width $k = \lceil \frac{10}{\epsilon} \rceil$
- Generate shiftings $\mathcal{T}_i$ of $\mathcal{T}$ by $(i, i)$ with $i \in \mathbb{Z}_k$

Observations for $r = 1$

- each monitoring area
  - is cut by at most 2 of the tilings $\mathcal{T}_i$
  - intersects at most 4 squares
Approximation Algorithm

Second Relaxation - 1

Maximum Lifetime

- Generate tiling $T$ of the area with width $k = \lceil 10/\epsilon \rceil$
- Generate shiftings $T_i$ of $T$ by $(i, i)$ with $i \in \mathbb{Z}_k$

Observations for $r = 1$

- each monitoring area
  - is cut by at most 2 of the tilings $T_i$,
  - intersects at most 4 squares
**Approximation Algorithm**

*Second Relaxation - 1*

---

**Maximum Lifetime**

- Generate tiling \( \mathcal{T} \) of the area with width \( k = \lceil \frac{10}{\epsilon} \rceil \)
- Generate shiftings \( \mathcal{T}_i \) of \( \mathcal{T} \) by \((i, i)\) with \( i \in \mathbb{Z}_k \)

---

**Observations for \( r = 1 \)**

- each monitoring area
  - is cut by at most 2 of the tilings \( \mathcal{T}_i \),
  - intersects at most 4 squares
Approximation Algorithm

Second Relaxation - 1

Maximum Lifetime

- Generate tiling \( \mathcal{T} \) of the area with width \( k = \lceil 10/\epsilon \rceil \)
- Generate shiftings \( \mathcal{T}_i \) of \( \mathcal{T} \) by \((i, i)\) with \( i \in \mathbb{Z}_k \)

Observations for \( r = 1 \)

- each monitoring area
  - is cut by at most 2 of the tilings \( \mathcal{T}_i \),
  - intersects at most 4 squares
**Approximation Algorithm**

*Second Relaxation - 2*

- Let algorithm $\mathcal{A}$ provide an $\alpha$-approximation for an area of size $k \times k$.
- Run $\mathcal{A}$ for each square of $T_i$:
  - $T_1 = \alpha \cdot \text{opt}_1$
  - at most $4\times$ excess use of each node
- Combine solutions of all $T_i$ weighted by $\frac{1-\epsilon}{k}$:
  - $T_1 = (1 - \epsilon) \cdot \alpha \cdot \text{opt}_1$
  - no violation of capacity constraints

**Results**
- possibly constant approximation factor by reducing $T_1$ by $(1 - \epsilon)$
Approximation Algorithm

Second Relaxation - 2

- Let algorithm $\mathcal{A}$ provide an $\alpha$-approximation for an area of size $k \times k$
- Run $\mathcal{A}$ for each square of $T_i$:
  - $T_1 = \alpha \cdot \text{opt}_1$
  - at most 4x excess use of each node
- Combine solutions of all $T_i$ weighted by $\frac{1-\epsilon}{k}$:
  - $T_1 = (1 - \epsilon) \cdot \alpha \cdot \text{opt}_1$
  - no violation of capacity constraints

Results
- possibly constant approximation factor by reducing $T_1$ by $(1 - \epsilon)$
Approximation Algorithm

Joined Relaxations

Combination of both Relaxations

- Let algorithm $A$ provide an $\alpha$-approximation for
  - square areas of width $k \times k$, and
  - sensor positions restricted to a $\delta/2$-grid

Observation

- Only $O(1/\delta^2 \epsilon^2)$ contributing nodes per square
  → independent of $N$!
**Approximation Algorithm**

*Joined Relaxations*

**Combination of both Relaxations**

- Let algorithm $A$ provide an $\alpha$-approximation for
  - square areas of width $k \times k$, and
  - sensor positions restricted to a $\delta/2$-grid

**Observation**

- Only $O(1/\delta^2 \epsilon^2)$ contributing nodes per square
  → independent of $N$!
Approximation Algorithm

Results - 1

Approximation guarantee

\[ T_{1/(1-\delta)} \geq (1 - \epsilon) \cdot \alpha \cdot opt_1 \]

- \( (1 - \epsilon) \): Segmentation of the area into smaller squares
- \( \alpha \): Approximation guarantee of algorithm \( A \)
- \( opt_1 \): Restriction of sensor positions to a grid

Applied relaxations

- Sensor radii can be larger than \( r \)
- Maximum lifetime can be smaller than the optimum
Approximation Algorithm

Results - 1

Approximation guarantee

\[ T_{1/(1-\delta)} \geq (1 - \epsilon) \cdot \alpha \cdot opt_1 \]

- \((1 - \epsilon)\): Segmentation of the area into smaller squares
- \(\alpha\): Approximation guarantee of algorithm \(A\)
- \(opt_1\): Restriction of sensor positions to a grid

Applied relaxations

- Sensor radii can be larger than \(r\)
- Maximum lifetime can be smaller than the optimum
Approximation Algorithm

Results - 1

Approximation guarantee

\[ \frac{T_1}{(1-\delta)} \geq (1 - \epsilon) \cdot \alpha \cdot opt_1 \]

- \((1 - \epsilon)\): Segmentation of the area into smaller squares
- \(\alpha\): Approximation guarantee of algorithm \(A\)
- \(opt_1\): Restriction of sensor positions to a grid

Applied relaxations

- Sensor radii can be larger than \(r\)
- Maximum lifetime can be smaller than the optimum
**Approximation Algorithm**

*Results - 1*

**Approximation guarantee**

\[ T_{1/(1-\delta)} \geq (1 - \epsilon) \cdot \alpha \cdot opt_1 \]

- \((1 - \epsilon)\): Segmentation of the area into smaller squares
- \(\alpha\): Approximation guarantee of algorithm \(A\)
- \(opt_1\): Restriction of sensor positions to a grid

**Applied relaxations**

- Sensor radii can be larger than \(r\)
- Maximum lifetime can be smaller than the optimum
Approximation Algorithm

Results - 1

Approximation guarantee

\[ \frac{T_1}{1 - \delta} \geq (1 - \epsilon) \cdot \alpha \cdot opt_1 \]

- \((1 - \epsilon)\): Segmentation of the area into smaller squares
- \(\alpha\): Approximation guarantee of algorithm \(A\)
- \(opt_1\): Restriction of sensor positions to a grid

Applied relaxations

- Sensor radii can be larger than \(r\)
- Maximum lifetime can be smaller than the optimum
Approximation Algorithm

Results - 2

Asymptotic running time

\[ O \left( N + \frac{1}{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{o_{pt^1}} \cdot f \left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right) \right) \]

- \( O(N) \): relocation of sensor nodes to grid points
- \( O(1/\epsilon) \): Number of tilings \( T_i \)
- \( O\left( \frac{\epsilon^2 \cdot N}{o_{pt^1}} \right) \): Number of squares to be considered per tiling
- \( O(f \left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right)) \): Running time of algorithm \( \mathcal{A} \)

Observation

- Overall running time is linear in \( N \)
- Running time of \( \mathcal{A} \) independent of \( N \), allowed to take exponential time
**Approximation Algorithm**

**Results - 2**

**Asymptotic running time**

\[
O \left( N + \frac{1}{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{opt_1} \cdot f \left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right) \right)
\]

- **\(O(N)\):** relocation of sensor nodes to grid points
- **\(O(1/\epsilon)\):** Number of tilings \(T_i\)
- **\(O(\frac{\epsilon^2 \cdot N}{opt_1})\):** Number of squares to be considered per tiling
- **\(O(f \left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right))\):** Running time of algorithm \(\mathcal{A}\)

**Observation**

- Overall running time is linear in \(N\)
- Running time of \(\mathcal{A}\) independent of \(N\), allowed to take exponential time
**Approximation Algorithm**

**Results - 2**

**Asymptotic running time**

\[ O \left( N + \frac{1}{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{\text{opt}_1} \cdot f\left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right) \right) \]

- \( O(N) \): relocation of sensor nodes to grid points
- \( O(1/\epsilon) \): Number of tilings \( T_i \)
- \( O\left( \frac{\epsilon^2 \cdot N}{\text{opt}_1} \right) \): Number of squares to be considered per tiling
- \( O(f\left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right)) \): Running time of algorithm \( \mathcal{A} \)

**Observation**

- Overall running time is linear in \( N \)
- Running time of \( \mathcal{A} \) independent of \( N \), allowed to take exponential time
Approximation Algorithm

Results - 2

Asymptotic running time

\[ O \left( N + \frac{1}{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{\text{opt}_1} \cdot f \left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right) \right) \]

- \( O(N) \): relocation of sensor nodes to grid points
- \( O(1/\epsilon) \): Number of tilings \( T_i \)
- \( O(\frac{\epsilon^2 \cdot N}{\text{opt}_1}) \): Number of squares to be considered per tiling
- \( O(\frac{1}{\epsilon}) \): Running time of algorithm \( \mathcal{A} \)

Observation

- Overall running time is linear in \( N \)
- Running time of \( \mathcal{A} \) independent of \( N \), allowed to take exponential time
Approximation Algorithm

Results - 2

Asymptotic running time

\[ O \left( N + \frac{1}{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{\text{opt1}} \cdot f\left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right) \right) \]

- \( O(N) \): relocation of sensor nodes to grid points
- \( O(1/\epsilon) \): Number of tilings \( T_i \)
- \( O(\frac{\epsilon^2 \cdot N}{\text{opt1}}) \): Number of squares to be considered per tiling
- \( O(f(O(\frac{1}{\delta^2 \epsilon^2}))) \): Running time of algorithm \( \mathcal{A} \)

Observation

- Overall running time is linear in \( N \)
- Running time of \( \mathcal{A} \) independent of \( N \), allowed to take exponential time
Approximation Algorithm

Results - 2

Asymptotic running time

\[ O \left( N + \frac{1}{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{\text{opt}_1} \cdot f \left( O \left( \frac{1}{\delta^2 \epsilon^2} \right) \right) \right) \]

- \( O(N) \): relocation of sensor nodes to grid points
- \( O(1/\epsilon) \): Number of tilings \( T_i \)
- \( O(\epsilon^2 \cdot N/\text{opt}_1) \): Number of squares to be considered per tiling
- \( O(f(O(1/\delta^2 \epsilon^2))) \): Running time of algorithm \( A \)

Observation

- Overall running time is linear in \( N \)
- Running time of \( A \) independent of \( N \), allowed to take exponential time
Conclusion
Summary and Outlook
Summary and Outlook

Gaussian Mixture Reduction

Summary

- provided point of view of algorithmics
  (up to now only approaches from numerics)
- fast clustering approach
  (PGMR yields better approximations but takes longer)

Outlook

- apply different initialization algorithms (hierarchical approach?)
- evaluate different distance measures (speed-up?)
- student thesis in cooperation with Marco Huber (ISAS)
**Summary and Outlook**

**Area Monitoring**

**Summary**

- Proof of **NP completeness**
  (omitted in this talk)

- Linear-time, constant-factor approximation scheme
  ([BermanCa04]: $O(n \log n)$ algorithm with similar approximation guarantees)

**Outlook**

- **Implementation** of approximation algorithm
  (applying exact algorithm implemented with CPLEX)

- **Generalisation** to arbitrary (convex) monitoring areas and general metriks
  (David Steurer - Princeton University)
Time for questions

Thank you, for your attention!
References
