Mixtures & Monitoring

(work in progress)

Dennis Schieferdecker (schiefer@ira.uka.de)

ITI Sanders, Universtiy of Karlsruhe (TH)

04.12.2008

Table of Contents

– I –

Gaussian Mixture Reduction

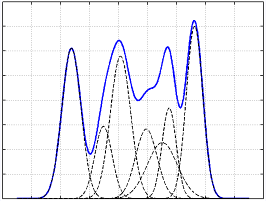
approximating functions

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

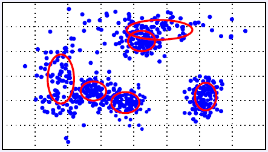
- recursive processing of GMs
- number of components grows rapidly (exponential)

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

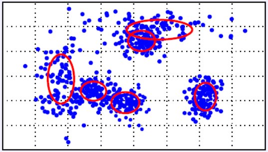
- recursive processing of GMs
- number of components grows rapidly (exponential)

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)

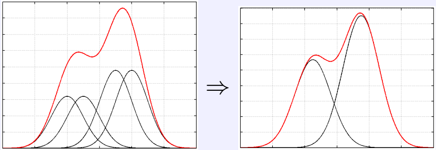
Introduction

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

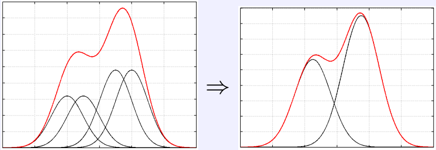
Introduction

Related Work

Classic Algorithms

West, Williams, Runnalls, . . .

- top-down approaches
- greedy merging of components

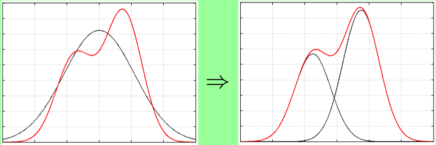


- slow or good quality

Modern Algorithm

Progressive Gaussian Mixture Reduction

- bottom-up method
- successive construction of approximated mixture



- average fast and better quality

Clustering Algorithm

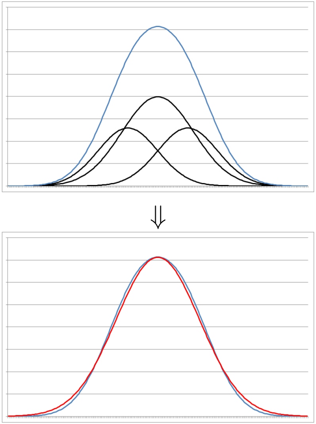
Problem formulation

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

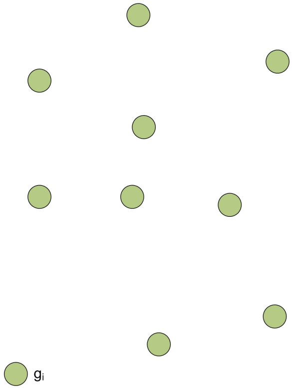
Approach with Algorithmics - P.o.V.

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

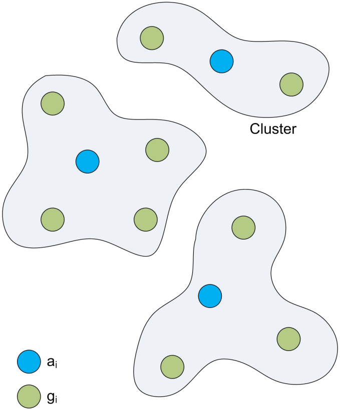
Approach with Algorithmics - P.o.V.

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

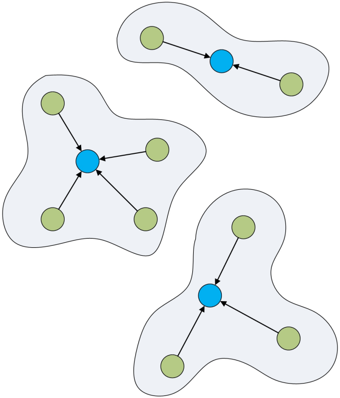
Description of the algorithm - 1

Clustering algorithm

- loosely based on Llyod's algorithm

Initialisation

- generate initial approximation $A = \{a_j\}$
  $\hookrightarrow$ yields initial cluster centers
  (**here:** using Runnalls' algorithm)
- compute initial assignments of Gaussians $g_{1..N}$ to cluster centers $a_{1..K}$: $center(\cdot) : g_i \to a_j$
  (**here:** using Kullback-Leibler discrimination)



Clustering Algorithm

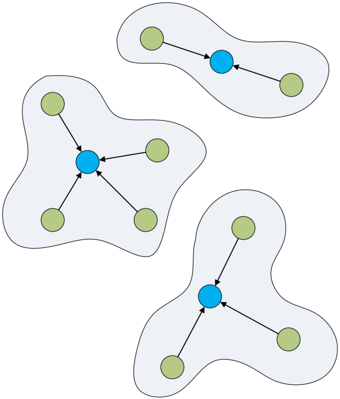
Description of the algorithm - 2

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!

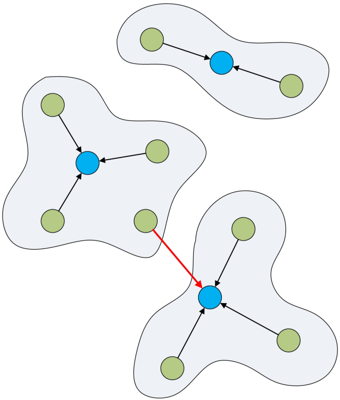


Clustering Algorithm

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

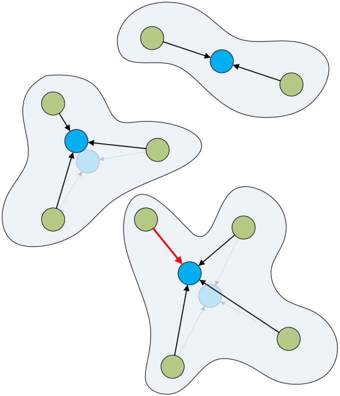
Description of the algorithm - 2

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 $\sum_{center(g_{\mathbf{i}=\mathbf{1}..\mathbf{N}})=a_{\mathbf{j}}} w_i \cdot g_i$

Comment

- Iteration order is important!



Clustering Algorithm

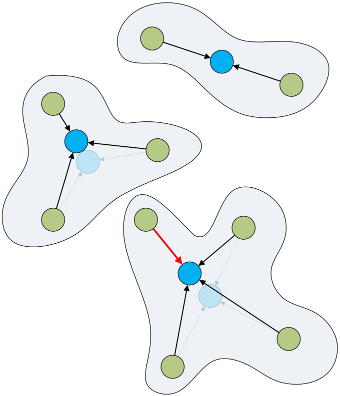
Description of the algorithm - 2

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

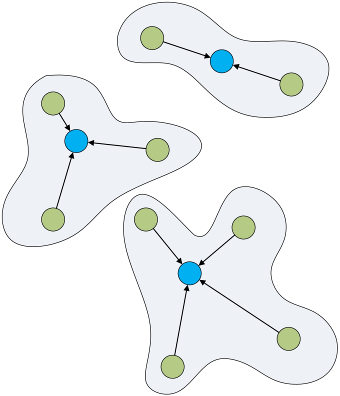
Description of the algorithm - 3

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

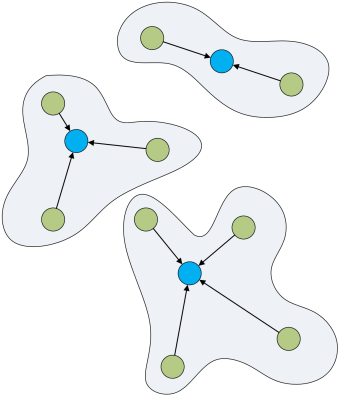
Description of the algorithm - 3

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



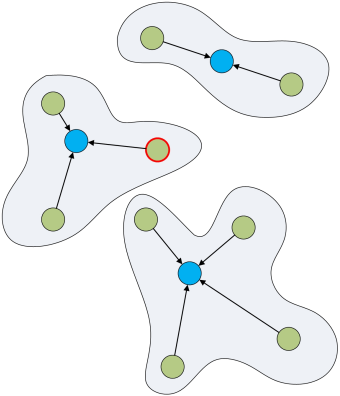
Clustering Algorithm

Description of the algorithm - 3

Distance measure $\mathrm{d}(\mathrm{g_i}, \mathrm{a_j})$ 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

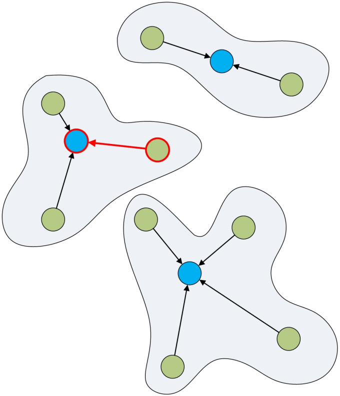


Clustering Algorithm

Description of the algorithm - 3

Distance measure $\mathrm{d(g_i, a_j)}$ 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

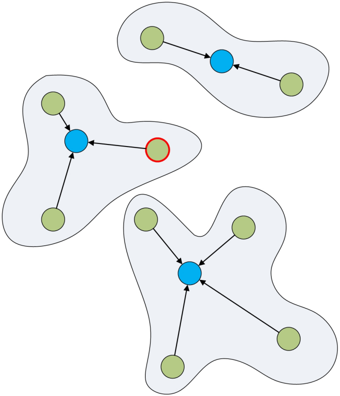
Clustering Algorithm

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

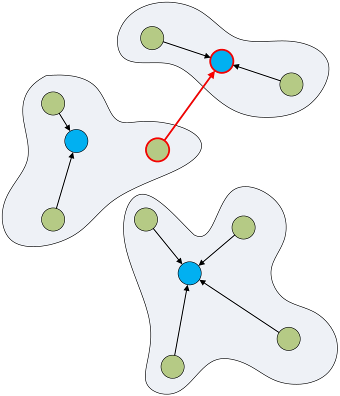


Clustering Algorithm

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



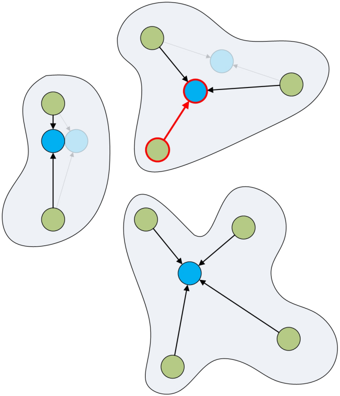
ISD: normalized Integrated Squared Distance between two functions

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



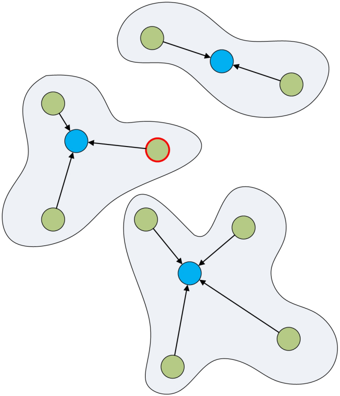
ISD: normalized Integrated Squared Distance between two functions

Clustering Algorithm

Description of the algorithm - 3

Distance measure $\mathrm{d}(\mathrm{g_i}, \mathrm{a_j})$ 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

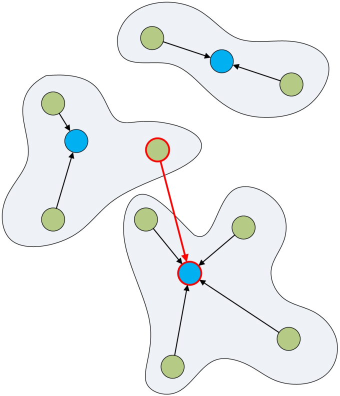
Clustering Algorithm

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



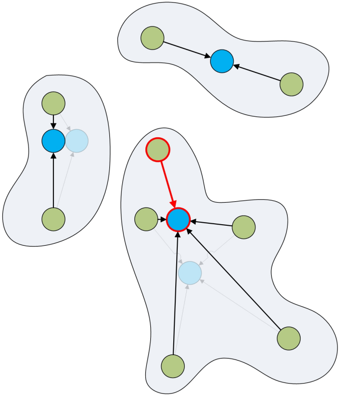
Clustering Algorithm

Description of the algorithm - 3

Distance measure $\mathrm{d}(\mathrm{g_i}, \mathrm{a_j})$ 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

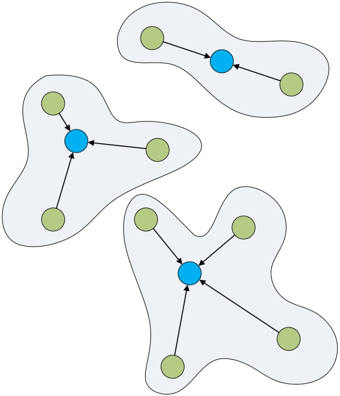


Clustering Algorithm

Description of the algorithm - 3

Distance measure $\mathrm{d}(\mathrm{g_i}, \mathrm{a_j})$ 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

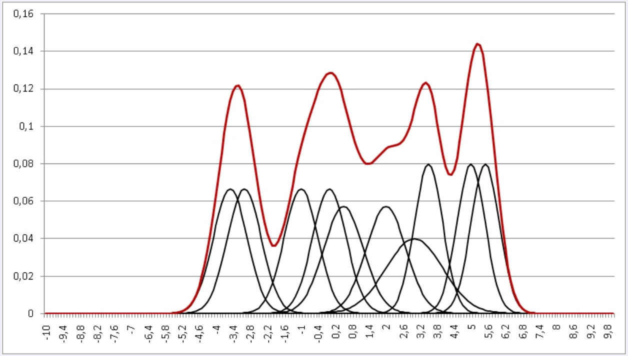
algorithm	N	K	running time	approx. error
Clustering	100	5	0.79s	3.10 ± 2.60
	100	10	1.09s	0.88 ± 0.74
	200	5	2.37s	1.99 ± 0.96
	200	10	3.04s	0.67 ± 0.32
	400	5	8.15s	1.30 ± 0.34
	400	10	9.53s	0.54 ± 0.11
West*	200	10	0.04s	3.81
Williams*	200	10	57.92s	1.03
PGMR*	200	5	4.78s	0.64

* results taken from [HuberHa08]

Preliminary results

Graphical Analysis

Function Plots

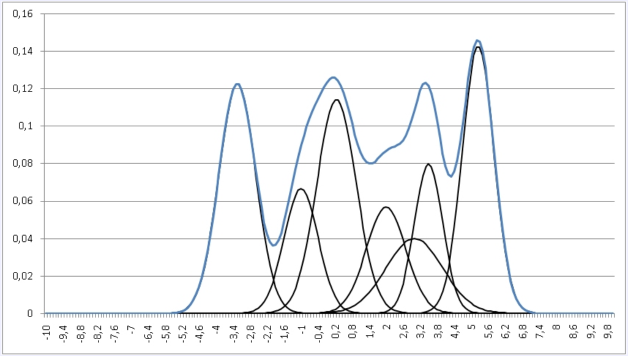


original gaussian mixture density (10 components)

Preliminary results

Graphical Analysis

Function Plots

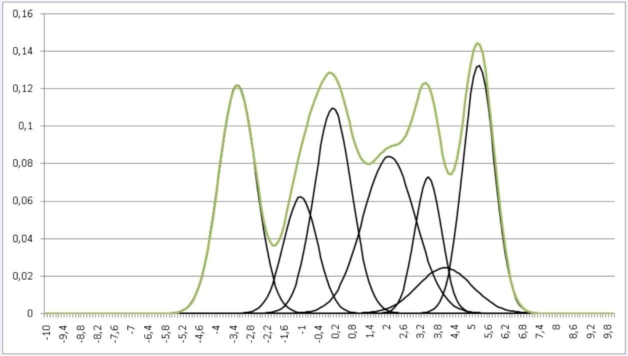


Clustering approximation (7 components) - error: $1,18\%$

Preliminary results

Graphical Analysis

Function Plots

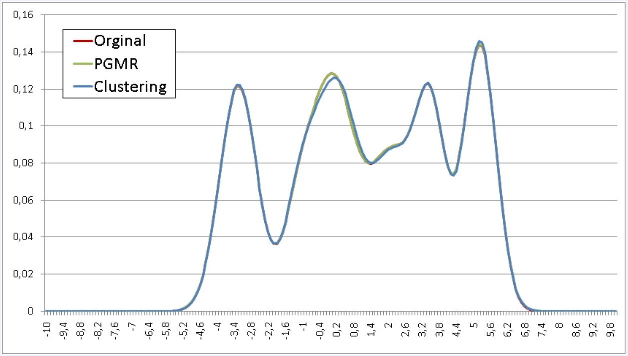


PGMR approximation (7 components) - error: $0,22\%$

Preliminary results

Graphical Analysis

Function Plots



comparison of PGMR and Clustering

– II –
Area Monitoring

energy-efficient, sensor-based

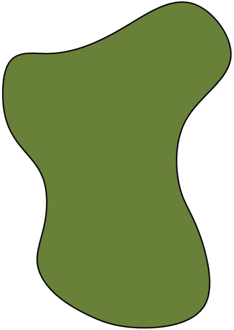
Motivation

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

↪ maximize lifetime



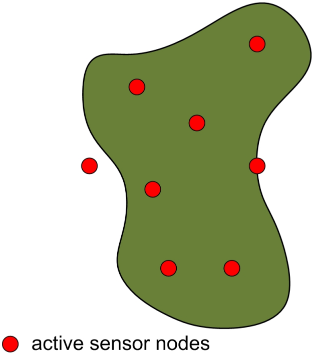
Motivation

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

$\hookrightarrow$ maximize lifetime



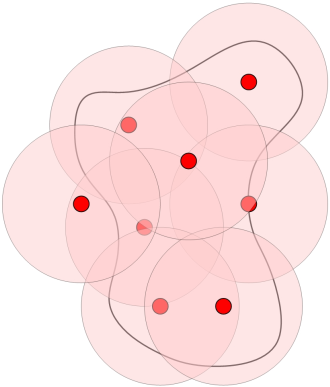
Motivation

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

$\hookrightarrow$ maximize lifetime



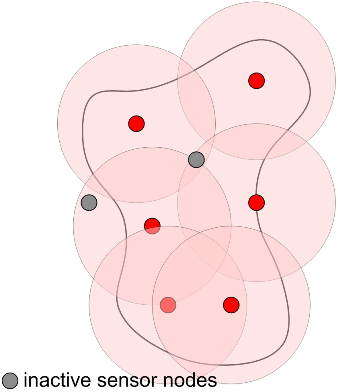
Motivation

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

  $\hookrightarrow$ maximize lifetime



Motivation

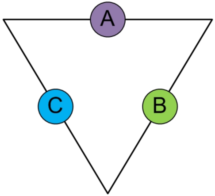
Overview - 2

Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%



Problem Denotation

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

Motivation

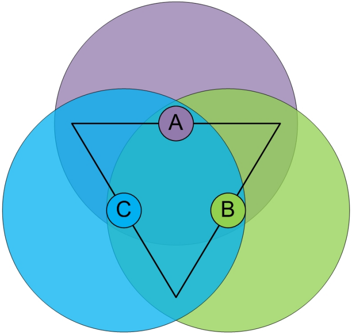
Overview - 2

Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%



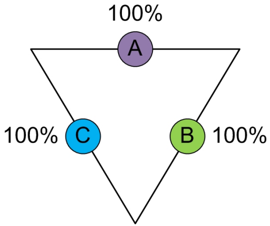
Problem Denotation

Scheduling of nodes for **L**ifetime maximization of area **C**overage **(SLC)**
(see [BermanCa04] for previous work)

Motivation

Overview - 2

(a) trivial solution



Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%

Problem Denotation

Scheduling of nodes for **L**ifetime maximization of area **C**overage **(SLC)**
(see [BermanCa04] for previous work)

Motivation

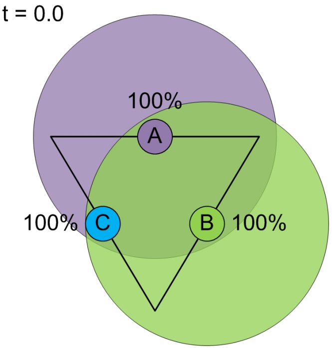
Overview - 2

Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%



Problem Denotation

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

Motivation

Overview - 2

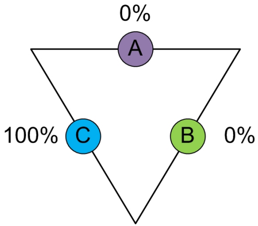
Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%

t = 1.0



Problem Denotation

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

Motivation

Overview - 2

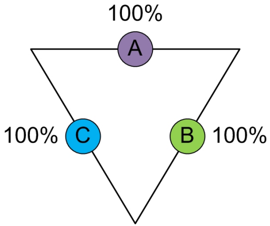
Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, 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

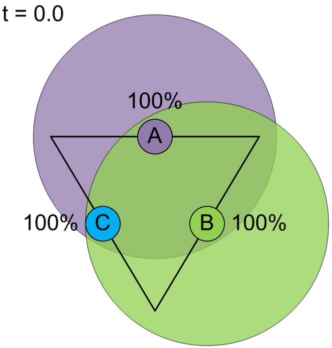
Overview - 2

Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%



Problem Denotation

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

Motivation

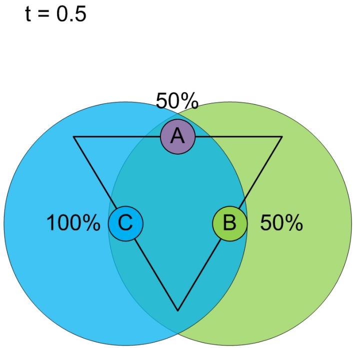
Overview - 2

Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%



Problem Denotation

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

Motivation

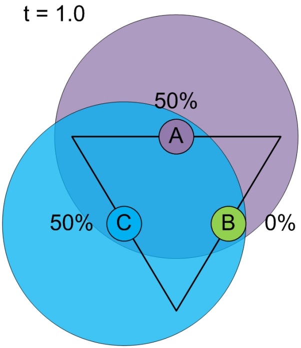
Overview - 2

Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%



Problem Denotation

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

Motivation

Overview - 2

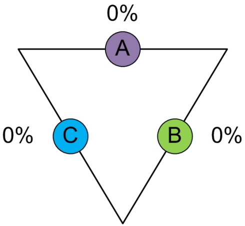
Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%

t = 1.5



Problem Denotation

Scheduling of nodes for **L**ifetime maximization of area **C**overage **(SLC)**
(see [BermanCa04] for previous work)

Motivation

Overview - 2

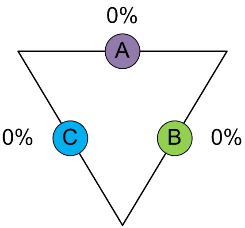
t = 1.5

Example

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

(a) Let AB be active for $t = 1.0$
$\hookrightarrow$ 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 $t = 0.5$
$\hookrightarrow$ $t_{total} = 1.5$, lifetime increased by 50%



Problem Denotation

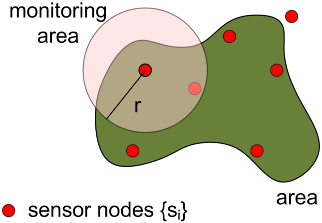
Scheduling of nodes for **L**ifetime maximization of area **C**overage **(SLC)**
(see [BermanCa04] for previous work)

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 $\{\mathcal{C}_j\}$, each monitoring the whole area
  - durations $\{t_j\}$, for which each cover $\mathcal{C}_j$ is active (scheduling)

Problem Formulation

Formulation with Linear Programming (LP)

LP formulation

maximize: lifetime T

$$T = max\{\mathbf{1}^T \mathbf{t} | \mathbf{t} \in \mathbb{R}^M\}$$

subject to: limited node capacities $\{c_i\}$

$$\sum_{j=1}^{M} A_{i,j}\ t_j \leq c_i \qquad i = 1, \dots N$$

- t_j: duration for which cover $\mathcal{C}_j$ is active
- $A_{i,j}$: 1, if node s_i in cover $\mathcal{C}_j$ is active, 0 otherwise
- c_i: capacity of node s_i

Problem Formulation

Remaining problems

Solution by CPLEX

- **General conception:**
  problem phraseable as LP $\rightarrow$ 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

Problem Formulation

Remaining problems

Solution by CPLEX

- **General conception:**
  problem phraseable as LP $\rightarrow$ 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

Prelimenaries

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

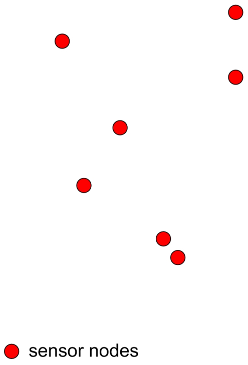
Sensor Radii

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

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

Results

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



Approximation Algorithm

First Relaxation

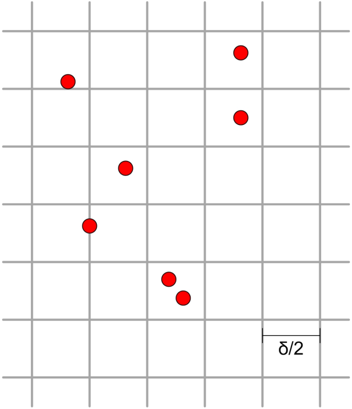
Sensor Radii

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

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

Results

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



Approximation Algorithm

First Relaxation

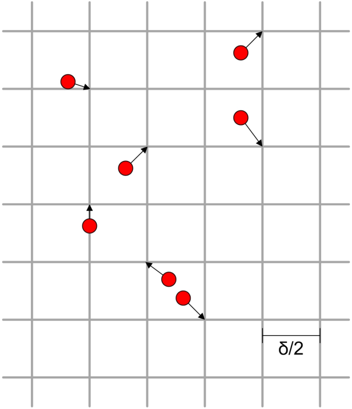
Sensor Radii

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

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

Results

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



Approximation Algorithm

First Relaxation

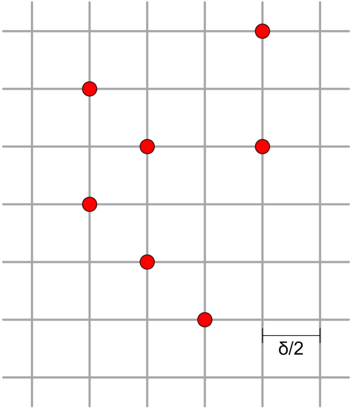
Sensor Radii

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

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

Results

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



Approximation Algorithm

First Relaxation

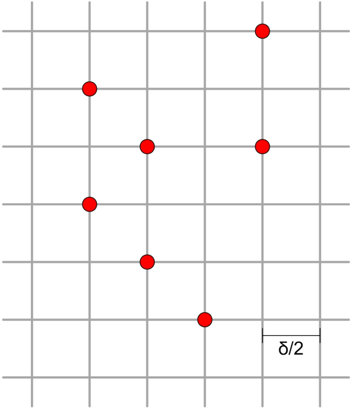
Sensor Radii

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

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

Results

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



Approximation Algorithm

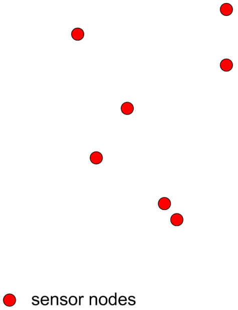
Second Relaxation - 1

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



Approximation Algorithm

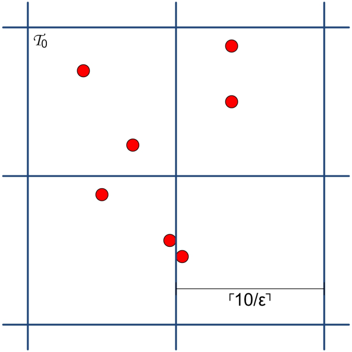
Second Relaxation - 1

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

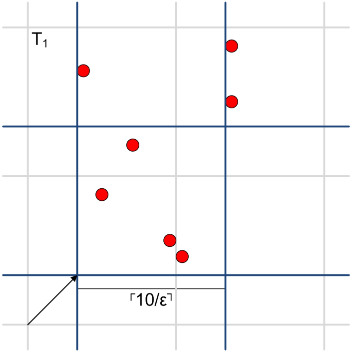
Second Relaxation - 1

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

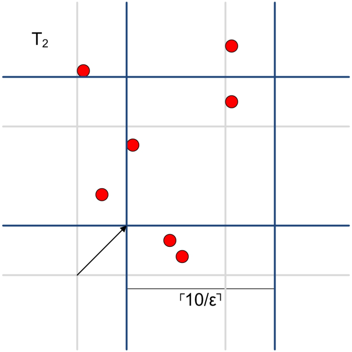
Second Relaxation - 1

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

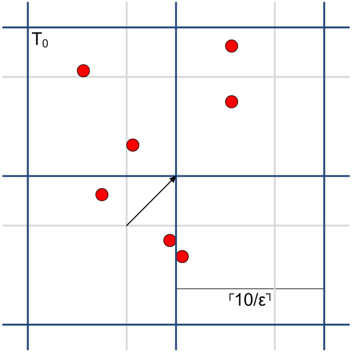
Second Relaxation - 1

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

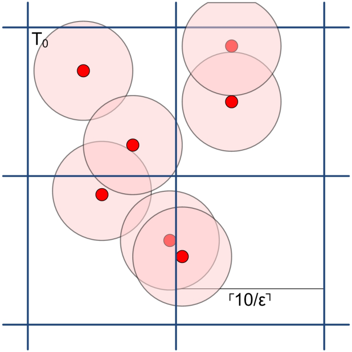
Second Relaxation - 1

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

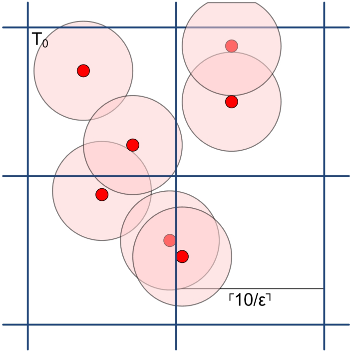
- each monitoring area
  - is cut by at most 2 of the tilings $\mathcal{T}_i$,
  - intersects at most 4 squares



Approximation Algorithm

Second Relaxation - 2

- Let algorithm $\mathcal{A}$ provide an α-approximation for an area of size $k \times k$
  - Run $\mathcal{A}$ for each square of $\mathcal{T}_i$:
    - $T_1 = \alpha \cdot opt_1$
    - at most $4x$ excess use of each node
  - Combine solutions of all $\mathcal{T}_i$ weighted by $\frac{1-\epsilon}{k}$:
    - $T_1 = (1-\epsilon) \cdot \alpha \cdot opt_1$
    - no violation of capacity constraints



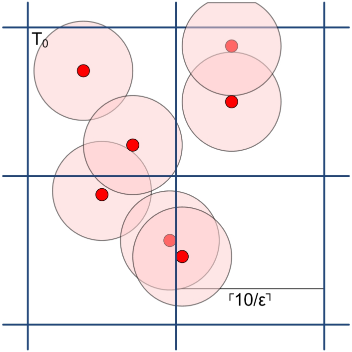
Results

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

Approximation Algorithm

Second Relaxation - 2

- Let algorithm $\mathcal{A}$ provide an α-approximation for an area of size $k \times k$
  - Run $\mathcal{A}$ for each square of $\mathcal{T}_i$:
    - $T_1 = \alpha \cdot opt_1$
    - at most $4x$ excess use of each node
  - Combine solutions of all $\mathcal{T}_i$ weighted by $\frac{1-\epsilon}{k}$:
    - $T_1 = (1-\epsilon) \cdot \alpha \cdot opt_1$
    - no violation of capacity constraints



Results

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

Approximation Algorithm

Joined Relaxations

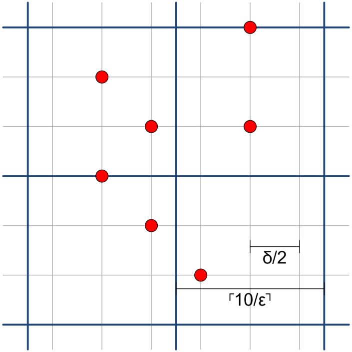
Combination of both Relaxations

- Let algorithm $\mathcal{A}$ provide an α-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

$\longrightarrow$ independent of N !



Approximation Algorithm

Joined Relaxations

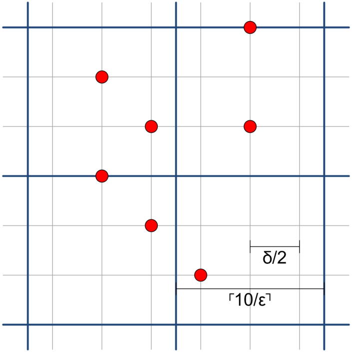
Combination of both Relaxations

- Let algorithm $\mathcal{A}$ provide an α-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

$\longrightarrow$ 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
- α: Approximation guarantee of algorithm $\mathcal{A}$
- 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
- α: Approximation guarantee of algorithm $\mathcal{A}$
- 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
- α: Approximation guarantee of algorithm $\mathcal{A}$
- 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
- α: Approximation guarantee of algorithm $\mathcal{A}$
- 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
- α: Approximation guarantee of algorithm $\mathcal{A}$
- 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

$$O\left(N + {}^{1}/_{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{opt_1} \cdot f\left(O\left({}^{1}/_{\delta^2\epsilon^2}\right)\right)\right)$$

- $O(N)$: relocation of sensor nodes to grid points
- $O({}^{1}/_{\epsilon})$: Number of tilings $\mathcal{T}_i$
- $O(\frac{\epsilon^2 \cdot N}{opt_1})$: Number of squares to be considered per tiling
- $O\left(f\left(O\left({}^{1}/_{\delta^2\epsilon^2}\right)\right)\right)$: 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

$$O\left(N + {}^{1}\!/_{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{opt_1} \cdot f\left(O\left({}^{1}\!/_{\delta^2 \epsilon^2}\right)\right)\right)$$

- $O(N)$: relocation of sensor nodes to grid points
- $O({}^{1}\!/_{\epsilon})$: Number of tilings $\mathcal{T}_i$
- $O(\frac{\epsilon^2 \cdot N}{opt_1})$: Number of squares to be considered per tiling
- $O\left(f\left(O\left({}^{1}\!/_{\delta^2 \epsilon^2}\right)\right)\right)$: 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

$$O\left(N + {}^{1}/_{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{opt_1} \cdot f\left(O\left({}^{1}/_{\delta^2\epsilon^2}\right)\right)\right)$$

- $O(N)$: relocation of sensor nodes to grid points
- $O({}^{1}/_{\epsilon})$: Number of tilings $\mathcal{T}_i$
- $O(\frac{\epsilon^2 \cdot N}{opt_1})$: Number of squares to be considered per tiling
- $O\left(f\left(O\left({}^{1}/_{\delta^2\epsilon^2}\right)\right)\right)$: 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

$$O\left(N + {}^{1}/_{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{opt_1} \cdot f\left(O\left({}^{1}/_{\delta^2\epsilon^2}\right)\right)\right)$$

- $O(N)$: relocation of sensor nodes to grid points
- $O({}^{1}/_{\epsilon})$: Number of tilings $\mathcal{T}_i$
- $O(\frac{\epsilon^2 \cdot N}{opt_1})$: Number of squares to be considered per tiling
- $O\left(f\left(O\left({}^{1}/_{\delta^2\epsilon^2}\right)\right)\right)$: 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

$$O\left(N + {}^{1}/_{\epsilon} \cdot \frac{\epsilon^2 \cdot N}{opt_1} \cdot f\left(O\left({}^{1}/_{\delta^2\epsilon^2}\right)\right)\right)$$

- $O(N)$: relocation of sensor nodes to grid points
- $O({}^{1}/_{\epsilon})$: Number of tilings $\mathcal{T}_i$
- $O(\frac{\epsilon^2 \cdot N}{opt_1})$: Number of squares to be considered per tiling
- $O\left(f\left(O\left({}^{1}/_{\delta^2\epsilon^2}\right)\right)\right)$: 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

$$O\left(N + {}^1\!/_\epsilon \cdot \frac{\epsilon^2 \cdot N}{opt_1} \cdot f\left(O\left({}^1\!/_{\delta^2\epsilon^2}\right)\right)\right)$$

- $O(N)$: relocation of sensor nodes to grid points
- $O({}^1\!/_\epsilon)$: Number of tilings $\mathcal{T}_i$
- $O(\frac{\epsilon^2 \cdot N}{opt_1})$: Number of squares to be considered per tiling
- $O\left(f\left(O\left({}^1\!/_{\delta^2\epsilon^2}\right)\right)\right)$: 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

– III –
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

[HuberHa08] M. Huber and U. D. Hanebeck, "Progressive Gaussian Mixture Reduction", in *Proceedings of the 11th International Conference on Information Fusion (Fusion 2008)*, Cologne, Germany, July, 2008

[BermanCa04] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, *Power efficient monitoring management in sensor networks*, in "Wireless Communications and Networking Conference", 2004