Strategic Deployment

Starting from a basis camp with a set of agents
Take over and control some settlements
Resistance in the outback
Enough agents for the movements between the settlements and for controlling the settlements

Task: Move efficiently around and occupy the settlements

Historic examples:
Gaius Julius Ceasar: Conquer of the Gauls (58 to 51 B.C.)
Alexander the Great (356 B.C. to 323 B.C.): Alexander's campaign
Modeled by an edge and vertex-weighted graph

Elmar Langetepe
Strategic deployment in graphs
Starting form a basis camp with a set of agents
Strategic Deployment

- Starting from a basis camp with a set of agents
- Take over and control some settlements
Strategic Deployment

- Starting from a basis camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
Strategic Deployment

- Starting from a basis camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
- Enough agents for the movements between the settlements and for controlling the settlements
Strategic Deployment

- Starting from a basis camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
- Enough agents for the movements between the settlements and for controlling the settlements
- Task: Move *efficiently* around and occupy the settlements
Strategic Deployment

- Starting from a basic camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
- Enough agents for the movements between the settlements and for controlling the settlements
- Task: Move *efficiently* around and occupy the settlements
- Historic examples!
  - Gaius Julius Caesar: Conquer of the Gauls (58 to 51 B.C.)
  - Alexander the Great (356 B.C. to 323 B.C.): Alexander’s campaign
Strategic Deployment

- Starting from a base camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
- Enough agents for the **movements between the settlements** and for **controlling the settlements**
- Task: Move *efficiently* around and occupy the settlements
- Historic examples!
  - Gaius Julius Caesar: Conquer of the Gauls (58 to 51 B.C.)
  - Alexander the Great (356 B.C. to 323 B.C.): Alexander’s campaign
- Modeled by an edge an vertex-weighted graph
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known

- Rules for the movement:
  1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
  3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.

Interesting computational questions:
- How many agents are required in total?
- How long does this take?
- $k$ agents given: How many settlements can we get?
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:

1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$.
2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.

Interesting computational questions:
- How many agents are required in total?
- How long does this take?
- $k$ agents given: How many settlements can we get?
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:
  1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$. 

Interesting computational questions:
- How many agents are required in total?
- How long does this take?
- $k$ agents given: How many settlements can we get?
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:
  1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.

Interesting computational questions:
- How many agents are required in total?
- How long does this take?
- $k$ agents given: How many settlements can we get?
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:
  1. Edge $e$, weight $w_e$: *Lower bound* on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
  3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.

Interesting computational questions:
- How many agents are required in total?
- How long does this take?
- $k$ agents given: How many settlements can we get?
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:
  1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
  3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
Model of the Problem

• Edge- and vertex-weighted graph $G$, Graph is fully known
• Some start vertex $v_s$
• Rules for the movement:
  1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
  3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.
• Visit and occupy the vertices accordingly
• Interesting computational questions:
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:
  1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
  3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:
  How many agents are required in total?
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:
  1. Edge $e$, weight $w_e$: *Lower bound* on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
  3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:
  How many agents are required in total?
  How long does this take?

Elmar Langetepe
Strategic deployment in graphs
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:
  1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
  3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:
  How many agents are required in total?
  How long does this take?
  $k$ agents given: How many settlements can we get?
Model of the Problem

- Edge- and vertex-weighted graph $G$, Graph is fully known
- Some start vertex $v_s$
- Rules for the movement:
  1. Edge $e$, weight $w_e$: Lower bound on the number of agents required for traversal of $e$.
  2. Vertex $v$, weight $w_v$: Number of agents that have to be placed at the vertex.
  3. First visit of $v$: Full amount $w_v$ have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:
  How many agents are required in total?
  How long does this take?
  $k$ agents given: How many settlements can we get?
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required.
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!

![Graph Diagram]

$1 \leq v_3 \leq 20$
$1 \leq v_2 \leq 7$
$1 \leq v_1 = v_s \leq 23$
$1 \leq v_4 \leq 25$
$1 \leq v_5 \leq 15$

Is the problem clear?

Elmar Langetepe
Strategic deployment in graphs
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!

Is the problem clear?
Example: Minimal number of agents required is 23!

1. At least \( w_e \) for traversing edge \( e \) are required
2. At the first visit of \( v \) exactly \( w_v \) have to be placed and will never by removed!

\[
\begin{array}{c}
v_1 = v_s \\
1 \\
1 \\
1 \\
15 \\
22 \\
7 \\
25 \\
1 \\
1 \\
1 \\
21 \leftarrow 23
\end{array}
\]
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required.
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!

Elmar Langetepe
Strategic deployment in graphs
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never be removed!

$v_1 = v_s$

$v_2$

$v_3 \leftarrow 23$

$v_4$

$v_5$

20 $\leftarrow$ 23

1

1

1

25

7

22

1

20

15

1

1

1

1

23

Is the problem clear?

Elmar Langetepe

Strategic deployment in graphs
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!

Elmar Langetepe
Strategic deployment in graphs
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required.
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!

![Graph Diagram]

Elmar Langetepe
Strategic deployment in graphs
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!

\[ v_1 = v_s \]

\[ 19 \leftarrow 23 \]

\[ v_3 \]
\[ v_5 \]
\[ v_1 = v_s \]
\[ v_2 \]
\[ v_4 \]
\[ v_5 \]
\[ v_3 \]
\[ v_5 \]
\[ v_1 \]
\[ v_2 \]
\[ v_4 \]
\[ v_5 \]
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!

![Diagram of a graph with nodes labeled $v_1, v_2, v_3, v_4, v_5$ and edges labeled with weights. The node $v_1$ is labeled as $v_s$. The edges $v_1-v_2$ and $v_1-v_3$ are marked with weights 1 and 7, respectively. The edge $v_3-v_5$ is marked with a weight of 22. The weight of the edge $v_1-v_4$ is 25. The node $v_2$ is connected to $v_3$ with a weight of 20. The node $v_4$ is connected to $v_5$ with a weight of 15. The node $v_5$ is connected to the edge $v_1-v_3$ with a weight of 19. The node $v_3$ is connected to the edge $v_1-v_2$ with a weight of 1. The node $v_1$ is placed at the start of the graph.]

4 agents remain unsettled! No other strategy is better!!

Is the problem clear?
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never by removed!

4 agents remain unsettled! No other strategy is better!!
Example: Minimal number of agents required is 23!

1. At least $w_e$ for traversing edge $e$ are required
2. At the first visit of $v$ exactly $w_v$ have to be placed and will never be removed!

4 agents remain unsettled! No other strategy is better!!

Is the problem clear?
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  - $w_{\text{max}} := \max\{w_e|e \in E\}$

$N + w_{\text{max}}$ on $G$ is sufficient!

Strategy $S$:
- $w_S := \max\{w_e|e \text{ was visited by } S\}$
- $S$ requires at most $N + w_S$
- $S$ requires at least $\max\{N, w_S\}$

Minimum Spanning Tree (MST),
- $w_{\text{MST}} := \max\{w_e|e \in \text{MST}\}$
- $N + w_{\text{MST}}$ on MST is sufficient

Any Strategy $S$ on $G$ requires at least $\max\{N, w_{\text{MST}}\}$

Lemma:
- Optimal Strategy for MST gives 2-Approximation for $G$
Number of agents required: Simple bounds!

- $G = (V, E), \ N := \sum_{v \in V} w_v$
- $w_{\text{max}} := \max\{w_e | e \in E\}$

Strategy $S$:
- $w_S := \max\{w_e | e \text{ was visited by } S\}$
- $S$ requires at most $N + w_{\text{max}}$ on $G$
- $S$ requires at least $\max\{N, w_S\}$

Minimum Spanning Tree (MST), $w_{\text{MST}} := \max\{w_e | e \in \text{MST}\}$
- $N + w_{\text{MST}}$ on MST is sufficient

Any Strategy $S$ on $G$ requires at least $\max\{N, w_{\text{MST}}\}$

Lemma: Optimal Strategy for $\text{MST}$ gives 2-Approximation for $G$
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\text{max}} := \max\{w_e | e \in E\}$
- $N + w_{\text{max}}$ on $G$ is sufficient!
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
- $w_{\text{max}} := \max\{w_e | e \in E\}$
- $N + w_{\text{max}}$ on $G$ is sufficient!
- Strategy $S$:

\[ w_S := \max\{w_e | e \text{ was visited by } S \} \]

$S$ requires at most $N + w_S$

$S$ requires at least $\max\{N, w_S\}$

Minimum Spanning Tree (MST),

\[ w_{\text{MST}} := \max\{w_e | e \in \text{MST}\} \]

$N + w_{\text{MST}}$ on MST is sufficient

Any Strategy $S$ on $G$ requires at least $\max\{N, w_{\text{MST}}\}$

Lemma: Optimal Strategy for MST gives 2-Approximation for $G$
Number of agents required: Simple bounds!

- **$G = (V, E)$**, $N := \sum_{v \in V} w_v$
  - $w_{\text{max}} := \max\{w_e | e \in E\}$
- $N + w_{\text{max}}$ on $G$ is sufficient!
- **Strategy $S$:**
  - $w_S := \max\{w_e | e \text{ was visited by } S\}$

![Diagram of a graph with vertices and weights]

- $N = 19$
- $w_{\text{max}} = 25$
Number of agents required: Simple bounds!

- \( G = (V, E), \quad N := \sum_{v \in V} w_v \)
  \( w_{\max} := \max\{w_e | e \in E\} \)
- \( N + w_{\max} \) on \( G \) is sufficient!
- Strategy \( S \):
  \( w_S := \max\{w_e | e \text{ was visited by } S\} \)
  \( S \) requires at most \( N + w_S \)
  \( S \) requires at least \( \max\{N, w_S\} \)

Minimum Spanning Tree (MST),
\( w_{\text{MST}} := \max\{w_e | e \in \text{MST}\} \)
\( N + w_{\text{MST}} \) on MST is sufficient

Any Strategy \( S \) on \( G \) requires at least \( \max\{N, w_{\text{MST}}\} \)

Lemma:
Optimal Strategy for MST gives 2-Approximation for \( G \)

\[ N = 19 \quad w_{\max} = 25 \]
\[ v_1 = v_s \]
\[ v_2 \]
\[ v_3 \]
\[ v_4 \]
\[ v_5 \]

\[ N = 19 \quad w_{\max} = 25 \]
\[ v_1 = v_s \]
\[ v_2 \]
\[ v_3 \]
\[ v_4 \]
\[ v_5 \]
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\text{max}} := \max\{w_e| e \in E\}$

- $N + w_{\text{max}}$ on $G$ is sufficient!

- **Strategy** $S$:
  $w_S := \max\{w_e | e \text{ was visited by } S\}$
  $S$ requires at most $N + w_S$
  $S$ requires at least $\max\{N, w_S\}$

- Minimum Spanning Tree (MST),
  $w_{\text{MST}} := \max\{w_e | e \in \text{MST}\}$

$N = 19$
$w_{\text{max}} = 25$

$
\begin{array}{c}
\text{v}_1 = \text{v}_s \\
1 \\
1 \\
1 \\
1
\end{array}
$

$
\begin{array}{c}
\text{v}_2 \quad 1 \\
\text{v}_3 \quad 1 \\
\text{v}_4 \quad 25 \\
\text{v}_5 \quad 15
\end{array}
$

Elmar Langetepe
Strategic deployment in graphs
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
- $w_{\text{max}} := \max\{w_e | e \in E\}$
- $N + w_{\text{max}}$ on $G$ is sufficient!
- **Strategy $S$:**
  - $w_S := \max\{w_e | e \text{ was visited by } S\}$
  - $S$ requires at most $N + w_S$
  - $S$ requires at least $\max\{N, w_S\}$
- Minimum Spanning Tree (MST),
  - $w_{\text{MST}} := \max\{w_e | e \in \text{ MST}\}$

![Graph](image_url)

- $N = 19$
- $w_{\text{max}} = 25$
- $w_{\text{MST}} = 20$

Elmar Langetepe  
Strategic deployment in graphs
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  - $w_{\text{max}} := \max\{w_e | e \in E\}$
- $N + w_{\text{max}}$ on $G$ is sufficient!
- Strategy $S$:
  - $w_S := \max\{w_e | e \text{ was visited by } S\}$
  - $S$ requires at most $N + w_S$
  - $S$ requires at least $\max\{N, w_S\}$
- Minimum Spanning Tree (MST),
  - $w_{\text{MST}} := \max\{w_e | e \in \text{MST}\}$
- $N + w_{\text{MST}}$ on MST is sufficient

\[ N = 19 \]
\[ w_{\text{max}} = 25 \]
\[ w_{\text{MST}} = 20 \]
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\text{max}} := \max\{w_e | e \in E\}$
- $N + w_{\text{max}}$ on $G$ is sufficient!
- Strategy $S$:
  $w_S := \max\{w_e | e \text{ was visited by } S\}$
  $S$ requires at most $N + w_S$
  $S$ requires at least $\max\{N, w_S\}$
- Minimum Spanning Tree (MST),
  $w_{\text{MST}} := \max\{w_e | e \in \text{ MST}\}$
- $N + w_{\text{MST}}$ on MST is sufficient
  Any Strategy $S$ on $G$ requires at least $\max\{N, w_{\text{MST}}\}$
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\text{max}} := \max \{w_e | e \in E\}$

- $N + w_{\text{max}}$ on $G$ is sufficient!

- **Strategy $S$:**
  $w_S := \max \{w_e | e \text{ was visited by } S\}$
  $S$ requires at most $N + w_S$
  $S$ requires at least $\max\{N, w_S\}$

- Minimum Spanning Tree (MST),
  $w_{\text{MST}} := \max \{w_e | e \in \text{ MST}\}$

- $N + w_{\text{MST}}$ on MST is sufficient
  Any Strategy $S$ on $G$ requires at least $\max\{N, w_{\text{MST}}\}$

- **Lemma:** Optimal Strategy for MST gives 2-Approximation for $G$

![Graph with nodes and edges labeled with $N = 19$, $w_{\text{max}} = 25$, $w_{\text{MST}} = 20$.]
(No-return) It suffices to fill the vertices as required, no agents have to return to the start vertex.
(No-return) It suffices to fill the vertices as required, no agents have to return to the start vertex.

(Return) Finally some agents have to return to the start vertex and report the success of the whole operation.
Variants: Return or No-Return!

(No-return) It suffices to fill the vertices as required, no agents have to return to the start vertex.

(Return) Finally some agents have to return to the start vertex and report the success of the whole operation.

Comparable to *routes* (round-trips) and *tours* (open paths) in TSP.
Variants: Return or No-Return!

(No-return) It suffices to fill the vertices as required, no agents have to return to the start vertex.

(Return) Finally some agents have to return to the start vertex and report the success of the whole operation.

Comparable to *routes* (round-trips) and *tours* (open paths) in TSP

Reporting the success formally means:
Set, $M$, of agents return to $v_s$, the union of all vertices visited by the members of $M$ equals $V$. 

Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

Optimal strategy: \( n + 1 \) agents, visit vertices in order of decreasing edge weights: \( n, n - 1, n - 2, \ldots, 2, 1 \).

Any other order will increase the number!

Example: Visit \( n - 2 \) before \( n \).

Lemma: Computational lower bound \( O(n \log n) \) by sorting (both variants, but real weights)!
Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

Optimal strategy: \( n + 1 \) agents, visit vertices in order of decreasing edge weights: \( n, n-1, n-2, \ldots, 2, 1 \)
Computational lower bound and algorithmic idea! Example!

Optimal strategy: \( n + 1 \) agents, visit vertices in order of decreasing edge weights: \( n, n-1, n-2, \ldots, 2, 1 \)

Any other order will increase the number!
Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

Optimal strategy: $n + 1$ agents, visit vertices in order of decreasing edge weights: $n, n-1, n-2, \ldots, 2, 1$

Any other order will increase the number!

Example: Visit $n - 2$ before $n$
Computational lower bound and algorithmic idea! Example!

Optimal strategy: $n + 1$ agents, visit vertices in order of decreasing edge weights: $n, n-1, n-2, \ldots, 2, 1$

Any other order will increase the number!

Example: Visit $n - 2$ before $n$
Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

Optimal strategy: \( n + 1 \) agents, visit vertices in order of decreasing edge weights: \( n, n-1, n-2, \ldots, 2, 1 \)

Any other order will increase the number!

Example: Visit \( n - 2 \) before \( n \)
Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

Optimal strategy: $n + 1$ agents, visit vertices in order of decreasing edge weights: $n, n-1, n-2, \ldots, 2, 1$

Any other order will increase the number!

Example: Visit $n-2$ before $n$
Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

Optimal strategy: $n + 1$ agents, visit vertices in order of decreasing edge weights: $n, n - 1, n - 2, \ldots, 2, 1$

Any other order will increase the number!
Example: Visit $n - 2$ before $n$
Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

Optimal strategy: $n + 1$ agents, visit vertices in order of decreasing edge weights: $n, n-1, n-2, \ldots, 2, 1$

Any other order will increase the number!

Example: Visit $n-2$ before $n$

**Lemma:** Computational lower bound $O(n \log n)$ by sorting (both variants, but real weights)!

![Diagram of a tree with labeled vertices and edges](attachment:tree_diagram.png)
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leaves and return!
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leaves and return!
Collect the leaves in subtrees w.r.t. dominating edge along path!
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leafs and return!
Collect the leafs in subtrees w.r.t. dominating edge along path!
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leaves and return!
Collect the leaves in subtrees w.r.t. dominating edge along path!

![Diagram of a tree with nodes and edges labeled with values and arrows indicating traversal]

Elmar Langetepe  Strategic deployment in graphs
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leaves and return!
Collect the leaves in subtrees w.r.t. dominating edge along path!
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leaves and return!
Collect the leaves in subtrees w.r.t. dominating edge along path!
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leaves and return!
Collect the leaves in subtrees w.r.t. dominating edge along path!
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leafs and return!
Collect the leafs in subtrees w.r.t. dominating edge along path!
Have to visit all leafs and return!
Collect the leafs in subtrees w.r.t. dominating edge along path!

**Optimal strategy:** Collected subtrees in decreasing of dominating edges. Simple DFS inside the coll. subtrees!
$O(n \log n)$ Algorithm for Trees: Return Variant!

Have to visit all leafs and return!
Collect the leafs in subtrees w.r.t. dominating edge along path!
**Optimal strategy:** Collected subtrees in decreasing of dominating edges. Simple DFS inside the coll. subtrees!
Theorem: The optimal strategy for the return variant in a tree visits (and then fully explores (by DFS)) the collected subtrees in the order of the dominating edges weights. The number of required agents can be computed in $O(n \log n)$ time.
Amortized $O(n \log n)$ Algorithm for No-Return Variant!

- Important question: In which leaf $b$ should we finally end?

$$T(b_7, b_6, b_4, b_2, b_3, b_1, b_5, b_0)^{0,41,0} =$$
$$[T(b_7, b_6), T(b_4, b_2, b_3), T(b_1), T(b_5), T(b_0)]$$
Amortized $O(n \log n)$ Algorithm for No-Return Variant!

- Important question: In which leaf $b$ should we finally end?
- Recursively: Choose collected subtree visited last, remaining subtrees in decreasing order.

$$T(b_7, b_6, b_4, b_2, b_3, b_1, b_5, b_0)^{0,41,0} =$$

$$[T(b_7, b_6), T(b_4, b_2, b_3), T(b_1), T(b_5), T(b_0)]$$

Elmar Langetepe
Strategic deployment in graphs
Amortized $O(n \log n)$ Algorithm for No-Return Variant!

- Important question: In which leaf $b$ should we finally end?
- Recursively: Choose collected subtree visited last, remaining subtrees in decreasing order.
- Datastructure comparing all alternatives, amortized $O(n \log n)$

$$T(b_7, b_6, b_4, b_2, b_3, b_1, b_5, b_0)^{0,41,0} =$$

$[T(b_7, b_6), T(b_4, b_2, b_3), T(b_1), T(b_5), T(b_0)]$

$T(b_5)^{7,9,0} = [T(b_5)]$

$T(b_7, b_6)^{12,8,6} = [T(b_7), T(b_6)]$

$T(b_4, b_2, b_3)^{10,8,0} = [T(b_4), T(b_2, b_3)]$

$T(b_2, b_3)^{3,4,0} = [T(b_2), T(b_3)]$

$T(b_2)^{2,2,1}$

$T(b_3)^{1,1,0}$

Elmar Langetepe  Strategic deployment in graphs
**Theorem:** For any strategy of the strategic deployment problem on a graph $G$ and the minimum number of agents required there is always an optimal strategy that let the optimal number of agents run in a single group!
Given: Finite set $X$ of $3n$ items and a collection $F = \{F_1, F_2, \ldots, F_m\}$ of subsets $m$ of $X$ with $|F_i| = 3$.

Question: Is there a subset $F_c \subseteq F$ with $X = \bigcup_{F_i \in F_c} F_i$ and $|F_c| = n$?
Given: Finite set $X$ of $3n$ items and a collection $F = \{F_1, F_2, \ldots, F_m\}$ of subsets $m$ of $X$ with $|F_i| = 3$.

Question: Is there a subset $F_c \subseteq F$ with $X = \bigcup_{F_i \in F_c} F_i$ and $|F_c| = n$?

Example: $X = \{a_1, a_2, \ldots, a_{12}\}$

$F_1 = \{a_1, a_2, a_3\}, F_2 = \{a_1, a_2, a_4\}, \ldots, F_6 = \{a_9, a_{11}, a_{12}\}$
**Given:** Finite set $X$ of $3n$ items and a collection $F = \{F_1, F_2, \ldots, F_m\}$ of subsets $m$ of $X$ with $|F_i| = 3$.

**Question:** Is there a subset $F_c \subseteq F$ with $X = \bigcup_{F_i \in F_c} F_i$ and $|F_c| = n$?

**Example:** $X = \{a_1, a_2, \ldots, a_{12}\}$, $F_1 = \{a_1, a_2, a_3\}$, $F_2 = \{a_1, a_2, a_4\}$, $\ldots$, $F_6 = \{a_9, a_{11}, a_{12}\}$

$N = m + 3n$

$m - n = 2$
Given: Finite set $X$ of $3n$ items and a collection $F = \{F_1, F_2, \ldots, F_m\}$ of subsets $m$ of $X$ with $|F_i| = 3$.

Question: Is there a subset $F_c \subseteq F$ with $X = \bigcup_{F_i \in F_c} F_i$ and $|F_c| = n$?

Example: $X = \{a_1, a_2, \ldots, a_{12}\}$

$F_1 = \{a_1, a_2, a_3\}, F_2 = \{a_1, a_2, a_4\}, \ldots, F_6 = \{a_9, a_{11}, a_{12}\}$
Conclusion

- Novel deployment problem on graphs with security constraints
- Many interesting variants and questions
- Optimal number of agents
  - NP-hard in general
  - 2-Approximation by MST
  - Optimal solution for trees in $\Theta(n \log n)$ (both variants!)
- Open questions: Combined measures (steps/number), Better approximations
- Joint work with Bernd Brüggemann (FKIE) and Andreas Lenerz (Univers. Bonn)