Covering problems
Prototype problems: Covering problems

• Setting:
  • Universe of \( N \) elements \( U = \{U_1, \ldots, U_N\} \)
  • A set of \( n \) sets \( S = \{s_1, \ldots, s_n\} \)
  • Find a collection \( C \) of sets in \( S \) (\( C \) subset of \( S \)) such that \( U_{c \in C} \) contains many elements from \( U \)

• Example:
  • \( U \): set of documents in a collection
  • \( s_i \): set of documents that contain term \( t_i \)
  • Find a collection of terms that cover most of the documents
Prototype covering problems

• **Set cover problem:** Find a small collection $C$ of sets from $S$ such that all elements in the universe $U$ are covered by some set in $C$

• **Best collection problem:** find a collection $C$ of $k$ sets from $S$ such that the collection covers as many elements from the universe $U$ as possible

• Both problems are NP-hard

• Simple approximation algorithms with provable properties are available and very useful in practice
Set-cover problem

- Universe of \( N \) elements \( U = \{U_1, \ldots, U_N\} \)
- A set of \( n \) sets \( S = \{s_1, \ldots, s_n\} \) such that \( U_i s_i = U \)

**Question:** Find the smallest number of sets from \( S \) to form collection \( C \) (\( C \) subset of \( S \)) such that \( U_{c \in C} c = U \)

- The set-cover problem is **NP-hard** (what does this mean?)
Trivial algorithm
Trivial algorithm

• Try all subcollections of $S$
Trivial algorithm

• Try all subcollections of $S$
Trivial algorithm

• Try all subcollections of \( S \)

• Select the smallest one that covers all the elements in \( U \)
Trivial algorithm

• Try all subcollections of $S$

• Select the smallest one that covers all the elements in $U$
Trivial algorithm

• Try all subcollections of $S$

• Select the smallest one that covers all the elements in $U$

• The running time of the trivial algorithm is $O(2^{|S||U|})$
Trivial algorithm

• Try all subcollections of $S$

• Select the smallest one that covers all the elements in $U$

• The running time of the trivial algorithm is $O(2^{|S||U|})$
Trivial algorithm

• Try all subcollections of $S$

• Select the smallest one that covers all the elements in $U$

• The running time of the trivial algorithm is $O(2^{|S||U|})$

• This is way too slow
Greedy algorithm for set cover

• Select first the largest-cardinality set \( s \) from \( S \)

• Remove the elements from \( s \) from \( U \)

• Recompute the sizes of the remaining sets in \( S \)

• Go back to the first step
As an algorithm

- \( X = U \)
- \( C = \{\} \)
- while \( X \) is not empty do
  - For all \( s \in S \) let \( a_s = |s \cap X| \)
  - Let \( s \) be such that \( a_s \) is maximal
  - \( C = C \cup \{s\} \)
  - \( X = X \setminus s \)
How can this go wrong?

• No global consideration of how good or bad a selected set is going to be
How good is the greedy algorithm?
How good is the greedy algorithm?

• Consider a minimization problem
  • In our case we want to minimize the cardinality of set \( C \)

• Consider an instance \( I \), and cost \( a^*(I) \) of the optimal solution
  • \( a^*(I) \): is the minimum number of sets in \( C \) that cover all elements in \( U \)

• Let \( a(I) \) be the cost of the approximate solution
  • \( a(I) \): is the number of sets in \( C \) that are picked by the greedy algorithm

• An algorithm for a minimization problem has approximation factor \( F \) if for all instances \( I \) we have that
  \[
a(I) \leq F \times a^*(I)
  \]

• Can we prove any approximation bounds for the greedy algorithm for set cover?
How good is the greedy algorithm for set cover?
How good is the greedy algorithm for set cover?

• *(Trivial?) Observation:* The greedy algorithm for set cover has approximation factor $F = s_{\text{max}}$, where $s_{\text{max}}$ is the set in $S$ with the largest cardinality.
How good is the greedy algorithm for set cover?

- *(Trivial?)* Observation: The greedy algorithm for set cover has approximation factor \( F = s_{\text{max}} \), where \( s_{\text{max}} \) is the set in \( S \) with the largest cardinality.

- Proof:
  - \( a^*(I) \geq N / |s_{\text{max}}| \text{ or } N \leq |s_{\text{max}}| a^*(I) \)
  - \( a(I) \leq N \leq |s_{\text{max}}| a^*(I) \)
How good is the greedy algorithm for set cover?
A tighter bound

• The greedy algorithm for set cover has approximation factor \( F = O(\log |s_{\text{max}}|) \)

• **Proof**: (From CLR “Introduction to Algorithms”)
Best-collection problem

- Universe of $N$ elements $U = \{U_1, \ldots, U_N\}$
- A set of $n$ sets $S = \{s_1, \ldots, s_n\}$ such that $U_i s_i = U$

**Question:** Find the a collection $C$ consisting of $k$ sets from $S$ such that $f(C) = |U_{c \in C} c|$ is maximized

- The best-collection problem is NP-hard

- Simple approximation algorithm has approximation factor $F = (e-1)/e$
Greedy approximation algorithm for the best-collection problem

• $C = \{\}$
• for every set $s$ in $S$ and not in $C$ compute the gain of $s$:
  \[ g(s) = f(C \cup \{s\}) - f(C) \]
• Select the set $s$ with the maximum gain
• $C = C \cup \{s\}$
• Repeat until $C$ has $k$ elements
Basic theorem

• The **greedy** algorithm for the best-collection problem has approximation factor \( F = \frac{e-1}{e} \)

• \( C^* \): optimal collection of cardinality \( k \)
• \( C \): collection output by the **greedy** algorithm
• \( f(C) \geq \frac{e-1}{e} \times f(C^*) \)
Reference
Reference
Reference

- Finding team of experts in a social network
  - [ T. Lappas, K. Liu, E. Terzi KDD 2009]
Reference

- Finding team of experts in a social network
  - [T. Lappas, K. Liu, E. Terzi KDD 2009]
Reference

- Finding team of experts in a social network
  - [T. Lappas, K. Liu, E. Terzi KDD 2009]
Reference

- Finding team of experts in a social network
  - [ T. Lappas, K. Liu, E. Terzi KDD 2009]
Reference

- Finding team of experts in a social network
  - [T. Lappas, K. Liu, E. Terzi KDD 2009]

  Expertise location in social networks: “How do I find an effective team of people that collectively can perform a given task”
Setting

- **Experts (defining the set \( V \), with \(|V|=n|\):**
  - Every expert \( i \) is associated with a **set of skills** \( X_i \)

- **Tasks**
  - Every task \( T \) is associated with a set of skills (\( T \)) **required** for performing the task

- **A social network of experts (\( G=(V,E) \))**
  - Edges between experts indicate ability to work well together

### Team Formation

<table>
<thead>
<tr>
<th></th>
<th>Team Formation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experts’ skills</td>
<td><strong>Known</strong></td>
</tr>
<tr>
<td>Participation of experts in teams</td>
<td><strong>Unknown</strong></td>
</tr>
<tr>
<td>Network structure</td>
<td><strong>Known</strong></td>
</tr>
</tbody>
</table>
Group–Formation Problem
Group–Formation Problem

- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task.
Group–Formation Problem

- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task.
Group–Formation Problem

- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task

- Task: set of required skills
Group–Formation Problem

- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task.

- Task: set of required skills.
Group–Formation Problem

- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task

- **Task**: set of required skills

- **Expert**: has a set of skills
Group–Formation Problem

- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task

- Task: set of required skills

- Expert: has a set of skills
Group–Formation Problem

- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task

- **Task**: set of required skills

- **Expert**: has a set of skills

- **Network**: represents strength of relationships
Expertise networks

- Collaboration networks (e.g., DBLP graph, coauthor networks)
- Organizational structure of companies
- LinkedIn
- Geographical (map) of experts
What makes a team effective for a task?

- $T = \{\text{algorithms, java, graphics, python}\}$

- **Alice**
  - algorithms

- **Bob**
  - python

- **Cynthia**
  - graphics, java

- **David**
  - graphics

- **Eleanor**
  - graphics, java, python
What makes a team effective for a task?

- \( T = \{\text{algorithms, java, graphics, python}\} \)

Alice \{algorithms\}

Bob \{python\}

Cynthia \{graphics, java\}

David \{graphics\}

Eleanor \{graphics, java, python\}
What makes a team effective for a task?

- $T = \{\text{algorithms, java, graphics, python}\}$

Coverage: For every required skill in $T$ there is at least one team member that has it.
Problem definition – v.0

- Given a task and a set of individuals, find the subset (team) of individuals that can perform the given task.
Is coverage enough?

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Cynthia</td>
<td>David</td>
</tr>
<tr>
<td>{algorithms}</td>
<td>{python}</td>
<td>{graphics, java}</td>
<td>{graphics}</td>
</tr>
<tr>
<td>Eleanor</td>
<td>T={algorithms, java, graphics, python}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{graphics, java, python}</td>
<td></td>
</tr>
</tbody>
</table>
Is coverage enough?

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

Alice \{algorithms\}  
Bob \{python\}  
Cynthia \{graphics, java\}  
David \{graphics\}  
Eleanor \{graphics, java, python\}

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ T = \{ \text{algorithms, java, graphics, python} \} \]
Is coverage enough?

\[ T = \{ \text{algorithms, java, graphics, python} \} \]
Is coverage enough?

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

Alice
{algorithms}

Bob
{python}

Cynthia
{graphics, java}

David
{graphics}

Eleanor
{graphics, java, python}

A, E could perform the task if they could communicate
Is coverage enough?

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

A, B, C form an effective group that can communicate.
Is coverage enough?

\[ T = \{\text{algorithms, java, graphics, python}\} \]

**Communication:** the members of the team must be able to **efficiently communicate** and work together.
Problem definition – v.1
Problem definition – v.1

- Given a task and a social network of individuals, find the subset (team) of individuals that can effectively perform the given task.
Problem definition – v.1

- Given a task and a social network of individuals, find the subset (team) of individuals that can effectively perform the given task.
Problem definition – v.1

- Given a task and a social network of individuals, find the subset (team) of individuals that can effectively perform the given task.
Problem definition – v.1

- Given a task and a social network of individuals, find the subset (team) of individuals that can effectively perform the given task.

- **Thesis:** Good teams are teams that have the necessary skills and can also communicate effectively.
How to measure effective communication?

- **Diameter** of the subgraph defined by the group members
How to measure effective communication?

- Diameter of the subgraph defined by the group members

  The longest shortest path between any two nodes in the subgraph
How to measure effective communication?

- **Diameter** of the subgraph defined by the group members

Diameter: The longest shortest path between any two nodes in the subgraph.
How to measure effective communication?

- **Diameter** of the subgraph defined by the group members
How to measure effective communication?

- **Diameter** of the subgraph defined by the group members

The longest shortest path between any two nodes in the subgraph

\[
\text{diameter} = \infty
\]
How to measure effective communication?

- Diameter of the subgraph defined by the group members

The longest shortest path between any two nodes in the subgraph
How to measure effective communication?

- **Diameter** of the subgraph defined by the group members

The longest shortest path between any two nodes in the subgraph

![Diagram](image.png)

\[ \text{diameter} = 1 \]
How to measure effective communication?

- MST (Minimum spanning tree) of the subgraph defined by the group members
How to measure effective communication?

- MST (Minimum spanning tree) of the subgraph defined by the group members

  The total weight of the edges of a tree that spans all the team nodes
How to measure effective communication?

- MST (Minimum spanning tree) of the subgraph defined by the group members

The total weight of the edges of a tree that spans all the team nodes
How to measure effective communication?

- MST (Minimum spanning tree) of the subgraph defined by the group members

The total weight of the edges of a tree that spans all the team nodes.
How to measure effective communication?

- MST (Minimum spanning tree) of the subgraph defined by the group members

The total weight of the edges of a tree that spans all the team nodes

\[ \text{MST} = \infty \]
How to measure effective communication?

- MST (Minimum spanning tree) of the subgraph defined by the group members

The total weight of the edges of a tree that spans all the team nodes
How to measure effective communication?

- MST (Minimum spanning tree) of the subgraph defined by the group members

The total weight of the edges of a tree that spans all the team nodes

\[ \text{MST} = 2 \]
Problem definition – v.1.1
Problem definition – v.1.1

- Given a task and a social network $G$ of experts, find the subset (team) of experts that can perform the given task and they define a subgraph in $G$ with the minimum diameter.

- Problem is NP-hard
Algorithms for minimizing the diameter: **RarestFirst**

- Find **Rarest** skill $\alpha_{\text{rare}}$ required for a task
- $S_{\text{rare}}$ group of people that have $\alpha_{\text{rare}}$
- Evaluate star graphs, centered at individuals from $S_{\text{rare}}$
- Report cheapest star
Algorithms for minimizing the diameter: **RarestFirst**

- Find **Rarest** skill $\alpha_{\text{rare}}$ required for a task
- $S_{\text{rare}}$ group of people that have $\alpha_{\text{rare}}$
- Evaluate star graphs, centered at individuals from $S_{\text{rare}}$
- Report cheapest star

**Running time:** **Quadratic** to the number of nodes
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ \{ \text{graphics, python, java} \} \quad \{ \text{algorithms, graphics} \} \]

\[ \{ \text{algorithms, graphics, java} \} \quad \{ \text{algorithms, graphics} \} \]

\[ \{ \text{python, java} \} \quad \{ \text{python} \} \]

\[ \alpha_{\text{rare}} = \text{algorithms} \]

\[ S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \]
The **RarestFirst** algorithm

\[
T = \{ \text{algorithms, java, graphics, python} \}
\]

\[
\{ \text{graphics, python, java} \} \quad \{ \text{algorithms, graphics} \}
\]

\[
\{ \text{algorithms, graphics, java} \} \quad \{ \text{algorithms, graphics} \}
\]

\[
\{ \text{python, java} \} \quad \{ \text{python} \}
\]

\[
\alpha_{\text{rare}} = \text{algorithms}
\]

\[
S_{\text{rare}} = \{ \text{Bob, Eleanor} \}
\]
The **RarestFirst** algorithm

\[
T = \{\text{algorithms, java, graphics, python}\}
\]

\[
\{\text{graphics, python, java}\} \quad \{\text{algorithms, graphics}\}
\]

\[
\{\text{algorithms, graphics, java}\} \quad \{\text{python, java}\}
\]

\[
\{\text{python}\}
\]

\[
\alpha_{\text{rare}} = \text{algorithms}
\]

\[
S_{\text{rare}} = \{\text{Bob, Eleanor}\}
\]
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

Skills:
- algorithms
- graphics
- java

\[ \alpha_{\text{rare}} = \text{algorithms} \]

\[ S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \]
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ \alpha_{\text{rare}} = \text{algorithms} \]

\[ S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \]
The **RarestFirst** algorithm

\[
T = \{ \text{algorithms, java, graphics, python} \}
\]

Skills:

- algorithms
- graphics
- java
- python

\[
\alpha_{\text{rare}} = \text{algorithms}
\]

\[
S_{\text{rare}} = \{ \text{Bob, Eleanor} \}
\]

Diameter = 2
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ \{ \text{graphics, python, java} \} \]

\[ \{ \text{algorithms, graphics} \} \]

\[ \{ \text{algorithms, graphics, java} \} \]

\[ \{ \text{python, java} \} \]

\[ \{ \text{python} \} \]

\[ \alpha_{\text{rare}} = \text{algorithms} \]

\[ S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \]
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms}, \text{java}, \text{graphics}, \text{python} \} \]

\[ \{ \text{graphics}, \text{python}, \text{java} \} \]

\[ \{ \text{algorithms}, \text{graphics} \} \]

\[ \{ \text{algorithms}, \text{graphics}, \text{java} \} \]

\[ \{ \text{python}, \text{java} \} \]

\[ \{ \text{python} \} \]

\[ \alpha_{\text{rare}} = \text{algorithms} \]

\[ S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \]
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ \{ \text{graphics, python, java} \} \]
\[ \{ \text{algorithms, graphics} \} \]
\[ \{ \text{algorithms, graphics, java} \} \]
\[ \{ \text{python, java} \} \]
\[ \{ \text{python} \} \]

\[ \alpha_{\text{rare}} = \text{algorithms} \]

\[ S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \]
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ \{ \text{graphics, python, java} \} \]

\[ \{ \text{algorithms, graphics} \} \]

\[ \{ \text{algorithms, graphics, java} \} \]

\[ \{ \text{python, java} \} \]

\[ \{ \text{python} \} \]

\[ \alpha_{\text{rare}} = \text{algorithms} \]

\[ S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \]

Skills:

- algorithms
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\{graphics, python, java\}    \{algorithms, graphics\}

\{algorithms, graphics, java\} \{algorithms, graphics, java\}

\{python, java\}    \{python\}

\( \alpha_{\text{rare}} = \text{algorithms} \)

\( S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \)
The **RarestFirst** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

**Skills:**
- algorithms
- graphics
- java

\[ \alpha_{\text{rare}} = \text{algorithms} \]

\[ S_{\text{rare}} = \{ \text{Bob, Eleanor} \} \]
The **RarestFirst** algorithm

\[
T = \{ \text{algorithms, java, graphics, python} \}
\]

\[
T = \{ \text{graphics, python, java} \} \quad \{ \text{algorithms, graphics} \} \quad \{ \text{algorithms, graphics, java} \} \quad \{ \text{python, java} \} \quad \{ \text{python} \}
\]

**Skills:**
- algorithms
- graphics
- java
- python

\[\alpha_{\text{rare}} = \text{algorithms}\]

\[S_{\text{rare}} = \{ \text{Bob, Eleanor} \}\]
The **RarestFirst** algorithm

$$\alpha_{\text{rare}} = \text{algorithms}$$

$$S_{\text{rare}} = \{\text{Bob, Eleanor}\}$$

$$T = \{\text{algorithms, java, graphics, python}\}$$
The **RarestFirst** algorithm

\[
T = \{\text{algorithms, java, graphics, python}\}
\]

Skills:
- algorithms
- graphics
- java
- python

\[
\alpha_{\text{rare}} = \text{algorithms}
\]

\[
S_{\text{rare}} = \{\text{Bob, Eleanor}\}
\]

Diameter = 1
Analysis of the RarestFirst algorithm (metric graphs)
Analysis of the **RarestFirst** algorithm (metric graphs)
Analysis of the \textbf{RarestFirst} algorithm (metric graphs)
Analysis of the **RarestFirst** algorithm (metric graphs)
Analysis of the **RarestFirst** algorithm (metric graphs)

- \( D = \max \{d_1, d_k, d_{\ell k}\} \)
Analysis of the **RarestFirst** algorithm (metric graphs)

\[ D = \max \{ d_\ell, d_k, d_{\ell k} \} \]
Analysis of the **RarestFirst** algorithm (metric graphs)

- $D = \max \{d_\ell, d_k, d_{\ell k}\}$
- Fact: $\text{OPT} \geq d_\ell$
Analysis of the **RarestFirst** algorithm (metric graphs)

- $D = \max \{d_1, d_k, d_{\ell k}\}$
- Fact: $\text{OPT} \geq d_\ell$
Analysis of the RarestFirst algorithm (metric graphs)

- D = max \{d_\ell, d_k, d_{\ell k}\}
- Fact: OPT ≥ d_\ell
- Fact: OPT ≥ d_k
Analysis of the **RarestFirst** algorithm (metric graphs)

- $D = \max \{d_1, d_k, d_{\ell k}\}$
- Fact: $\text{OPT} \geq d_1$
- Fact: $\text{OPT} \geq d_k$
Analysis of the **RarestFirst** algorithm (metric graphs)

- \( D = \max \{d_{\ell}, d_k, d_{\ell_k}\} \)
- Fact: \( \text{OPT} \geq d_{\ell} \)
- Fact: \( \text{OPT} \geq d_k \)
- \( D \leq d_{\ell_k} \leq d_{\ell} + d_k \leq 2 \times \text{OPT} \)
Problem definition – v.1.2
Problem definition – v.1.2

- Given a task and a social network $G$ of experts, find the subset (team) of experts that can perform the given task and they define a subgraph in $G$ with the minimum MST cost.

- Problem is NP-hard
The Steiner Tree problem

- Graph $G = (V, E)$
- Partition of $V$ into $V = \{R, N\}$
- Find $G'$ subgraph of $G$ such that $G'$ contains all the required vertices (R) and $\text{MST}(G')$ is minimized
The **EnhancedSteiner** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]
The **EnhancedSteiner** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

![Diagram showing set membership of elements in T]

- **A** \{graphics, python, java\}
- **B** \{algorithms, graphics\}
- **C** \{python, java\}
- **D** \{python\}
- **E** \{algorithms, graphics, java\}

 Enlightenment **Steiner** algorithm
The **EnhancedSteiner** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

![Diagram showing the EnhancedSteiner algorithm with nodes labeled A, B, C, D, and E connected by edges with sets of languages such as \{graphics, java\} and \{algorithms, graphics\}].
The EnhancedSteiner algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]
The **EnhancedSteiner** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]
The EnhancedSteiner algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

Graph:

- Node E with labels: python, algorithms, graphics
- Nodes A, B, C, D connected to E with dashed lines
The EnhancedSteiner algorithm

\[ T = \{\text{algorithms, java, graphics, python}\} \]
The EnhancedSteiner algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]
The EnhancedSteiner algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

\[ \text{MST Cost} = 1 \]
Other ways of exploiting the SteinerTree problem

- Graph $G(V,E)$
- Partition of $V$ into $V = \{R,N\}$
- Find $G'$ subgraph of $G$ such that $G'$ contains all the required vertices ($R$) and $\text{MST}(G')$ is minimized
The **CoverSteiner** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]
The **CoverSteiner** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

1. Solve **SetCover**
The **CoverSteiner** algorithm

$$T = \{ \text{algorithms}, \text{java}, \text{graphics}, \text{python} \}$$

1. Solve **SetCover**
2. Solve **Steiner**
The **CoverSteiner** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

1. Solve **SetCover**
2. Solve **Steiner**
The **CoverSteiner** algorithm

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

1. Solve **SetCover**
2. Solve **Steiner**

Diagram:

- A \{graphics, python, java\}
- B \{algorithms, graphics\}
- C \{python, java\}
- D \{python\}
- E

- A to E \{algorithms, graphics, java\}
- B to E \{algorithms, graphics\}
- C to E \{python, java\}
- D to E \{python\}
The **CoverSteiner** algorithm

\[
T = \{ \text{algorithms, java, graphics, python} \}
\]

1. Solve **SetCover**
2. Solve **Steiner**

\[
\begin{align*}
\text{E} & \quad \{\text{algorithms, graphics, java}\} \\
\text{A} & \quad \{\text{graphics, python, java}\} \\
\text{B} & \quad \{\text{algorithms, graphics}\} \\
\text{C} & \quad \{\text{python, java}\} \\
\text{D} & \quad \{\text{python}\}
\end{align*}
\]

**MST Cost = 1**
How good is CoverSteiner algorithm?
How good is the CoverSteiner algorithm?

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

Diagram:
- Node A: \{graphics, python, java\}
- Node B: \{algorithms, graphics\}
- Node C: \{python, java\}
- Node D: \{python\}
- Node E: \{algorithms, graphics, java\}
How good is CoverSteiner algorithm?

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

1. Solve SetCover
How good is CoverSteiner algorithm?

$$T = \{ \text{algorithms, java, graphics, python} \}$$

1. Solve SetCover
How good is CoverSteiner algorithm?

1. Solve SetCover
2. Solve Steiner

T={algorithms, java, graphics, python}
How good is **CoverSteiner** algorithm?

\[ T = \{ \text{algorithms, java, graphics, python} \} \]

1. Solve **SetCover**
2. Solve **Steiner**

MST Cost = Infty
Experiments – Cardinality of teams

Dataset

**DBLP** graph (DB, Theory, ML, DM)

~6000 authors

~2000 features

**Features:** keywords appearing in papers

**Tasks:** Subsets of keywords with different cardinality $k$
Example teams (I)

- S. Brin, L. Page: The anatomy of a large-scale hypertextual Web search engine
  - Paolo Ferragina, Patrick Valduriez, H. V. Jagadish, Alon Y. Levy, Daniela Florescu
    Divesh Srivastava, S. Muthukrishnan
  - P. Ferragina, J. Han, H. V. Jagadish, Kevin Chen–Chuan Chang, A. Gulli, S.
    Muthukrishnan, Laks V. S. Lakshmanan
Example teams (II)

- J. Han, J. Pei, Y. Yin: Mining frequent patterns without candidate generation
  - F. Bronchi
- A. Gionis, H. Mannila, R. Motwani
Extensions

- Skill attribution

<table>
<thead>
<tr>
<th></th>
<th>Team</th>
<th>Skill Attribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experts’ skills</td>
<td>Known</td>
<td>Unknown</td>
</tr>
<tr>
<td>Participation of experts in</td>
<td>Unknown</td>
<td>Known</td>
</tr>
<tr>
<td>Network structure</td>
<td>Known</td>
<td>Irrelevant</td>
</tr>
</tbody>
</table>

- Team chemistry as a factor of success
Example teams (II)

- J. Han, J. Pei, Y. Yin: Mining frequent patterns without candidate generation
  - F. Bronchi
- A. Gionis, H. Mannila, R. Motwani