Online Social Networks and Media

Team Formation in Social Networks

Thanks to Evimari Terzi

ALGORITHMS FOR TEAM FORMATION

Team-formation problems

- 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 and potentially a budget
- Expert: has a set of skills and potentially a price
- Network: represents strength of relationships

Security expert

Electronics expert

Insider

Mechanic

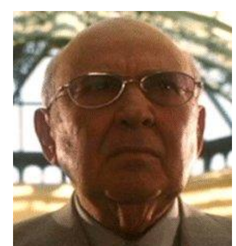
Pick-pocket thief

Organizer

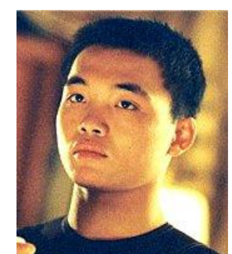
Co-organizer

Mechanic

Explosives expert



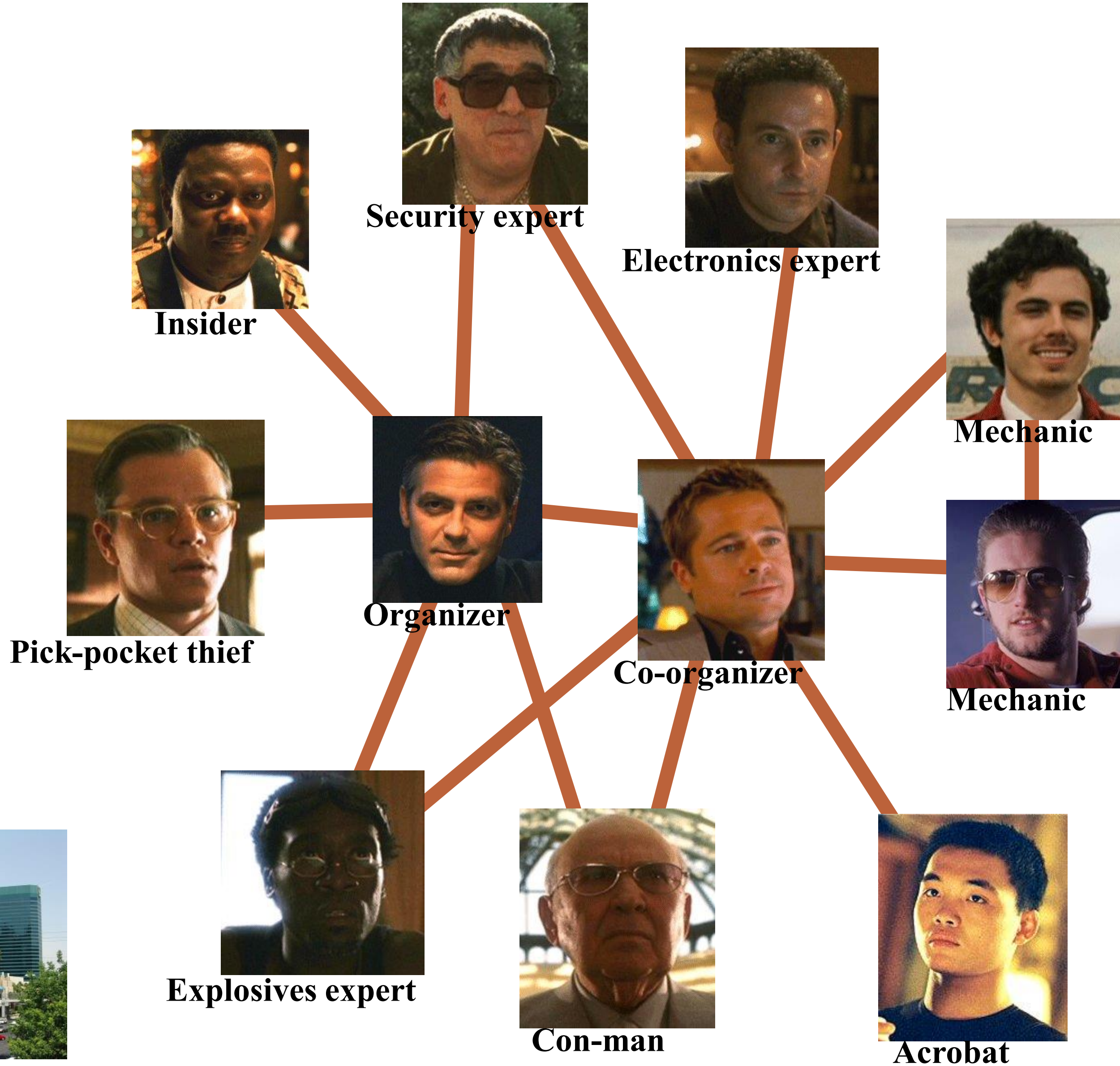
Con-man



Acrobat


george clooney
matt damon
andy garcia
brad pitt
julia roberts
OCEAN'S ELEVEN
Place your bets.


MIRAGE
GRAND


Security expert
Electronics expert
Insider
Mechanic
Pick-pocket thief
Organizer
Co-organizer
Mechanic
Explosives expert
Con-man
Acrobat

Applications

- Collaboration networks (e.g., scientists, actors)
- Organizational structure of companies
- LinkedIn, UpWork, FreeLance
- Geographical (map) of experts

Simple Team formation Problem

- Input:
  - A task T, consisting of a set of skills
  - A set of candidate experts each having a subset of skills

T = {algorithms, java, graphics, python}

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

- Problem: Given a task and a set of experts, find the smallest subset (team) of experts that together have all the required skills for the task

Set Cover

- The Set Cover problem:
  - We have a universe of elements $U = \{x_1, \dots, x_N\}$
  - We have a collection of subsets of U, $\boldsymbol{S} = \{S_1, \dots, S_n\}$, such that $\bigcup_i S_i = U$
  - We want to find the smallest sub-collection $\boldsymbol{C} \subseteq \boldsymbol{S}$ of $\boldsymbol{S}$, such that $\bigcup_{S_i \in \boldsymbol{C}} S_i = U$
    - The sets in $\boldsymbol{C}$ cover the elements of U

Coverage

- The Simple Team Formation Problem is a just an instance of the Set Cover problem
  - Universe U of elements = Set of all skills
  - Collection $\boldsymbol{S}$ of subsets = The set of experts and the subset of skills they possess.

T = {algorithms, java, graphics, python}

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

Complexity

- The Set Cover problem are NP-complete
  - What does this mean?
  - Why do we care?

- There is no algorithm that can guarantee finding the best solution in polynomial time
  - Can we find an algorithm that can guarantee to find a solution that is close to the optimal?
  - Approximation Algorithms.

A simple approximation ratio for set cover

- Any algorithm for set cover has approximation ratio $\alpha = |S_{max}|$, where S_{max} is the set in $\boldsymbol{S}$ with the largest cardinality

- Proof:
  - $OPT(X) \geq N/|S_{max}| \Rightarrow N \leq |S_{max}|OPT(X)$
  - $ALG(X) \leq N \leq |S_{max}|OPT(X)$

- This is true for any algorithm.
- Not a good bound since it may be that $|S_{max}| = O(N)$

An algorithm for Set Cover

- What is the most natural algorithm for Set Cover?

- Greedy: each time add to the collection $\boldsymbol{C}$ the set S_i from $\boldsymbol{S}$ that covers the most of the remaining uncovered elements.

The GREEDY algorithm

GREEDY(U,S)

X= U

C = {}

while X is not empty do

For all $S_i \in \boldsymbol{S}$ let $\text{gain}(S_i) = |S_i \cap X|$

Let S_* be such that $gain(S_*)$ is maximum

C = C U {S_*}

X = X\ S_*

S = ***S***\ S_*

The number of elements covered by S_i not already covered by C.

Greedy is not always optimal

Alice
C, C++, Unix

Eleanor
Python, Joomla

Required Skills
C, C++, Unix, php, Java, Python, Joomla

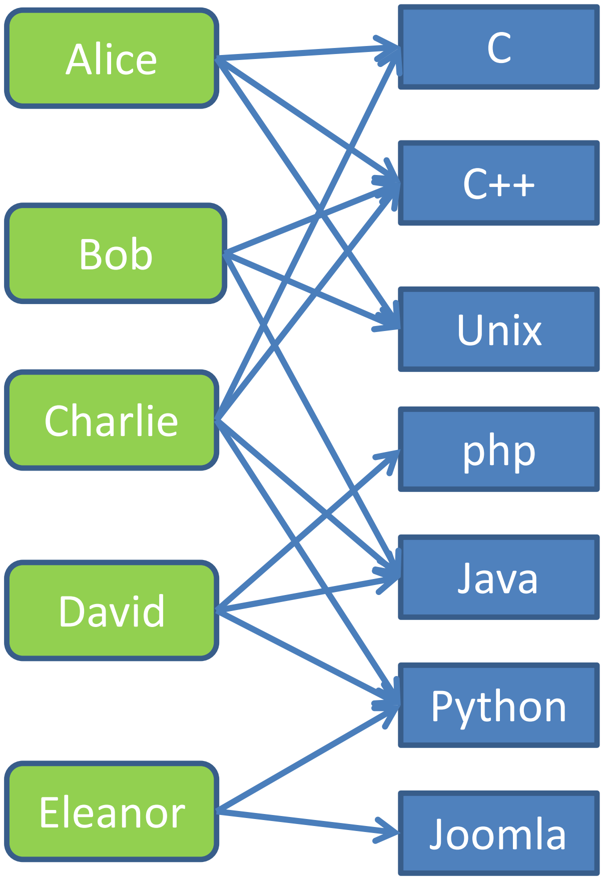
Bob
C++, Unix, Java

David
php, Java, Python

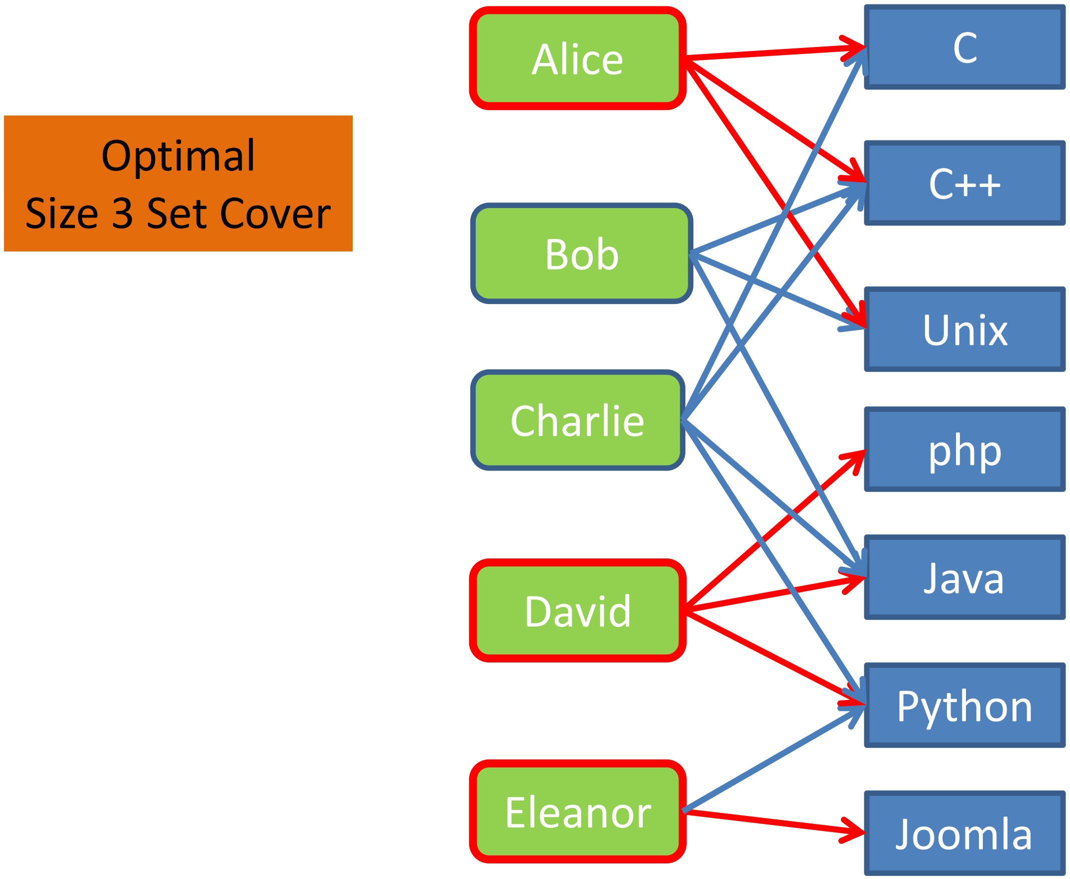
Charlie
C, C++, Java, Python

Greedy is not always optimal

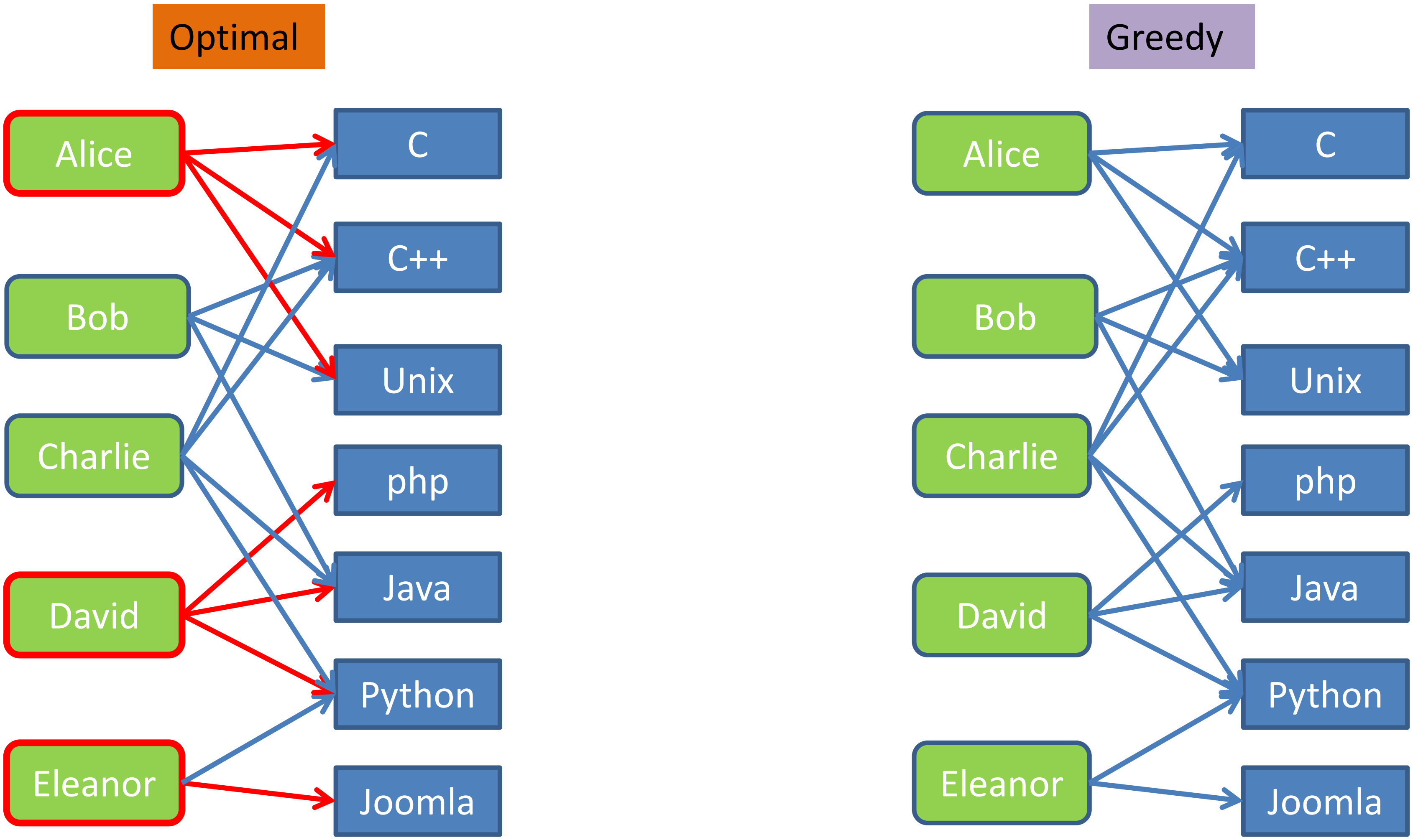
A different representation



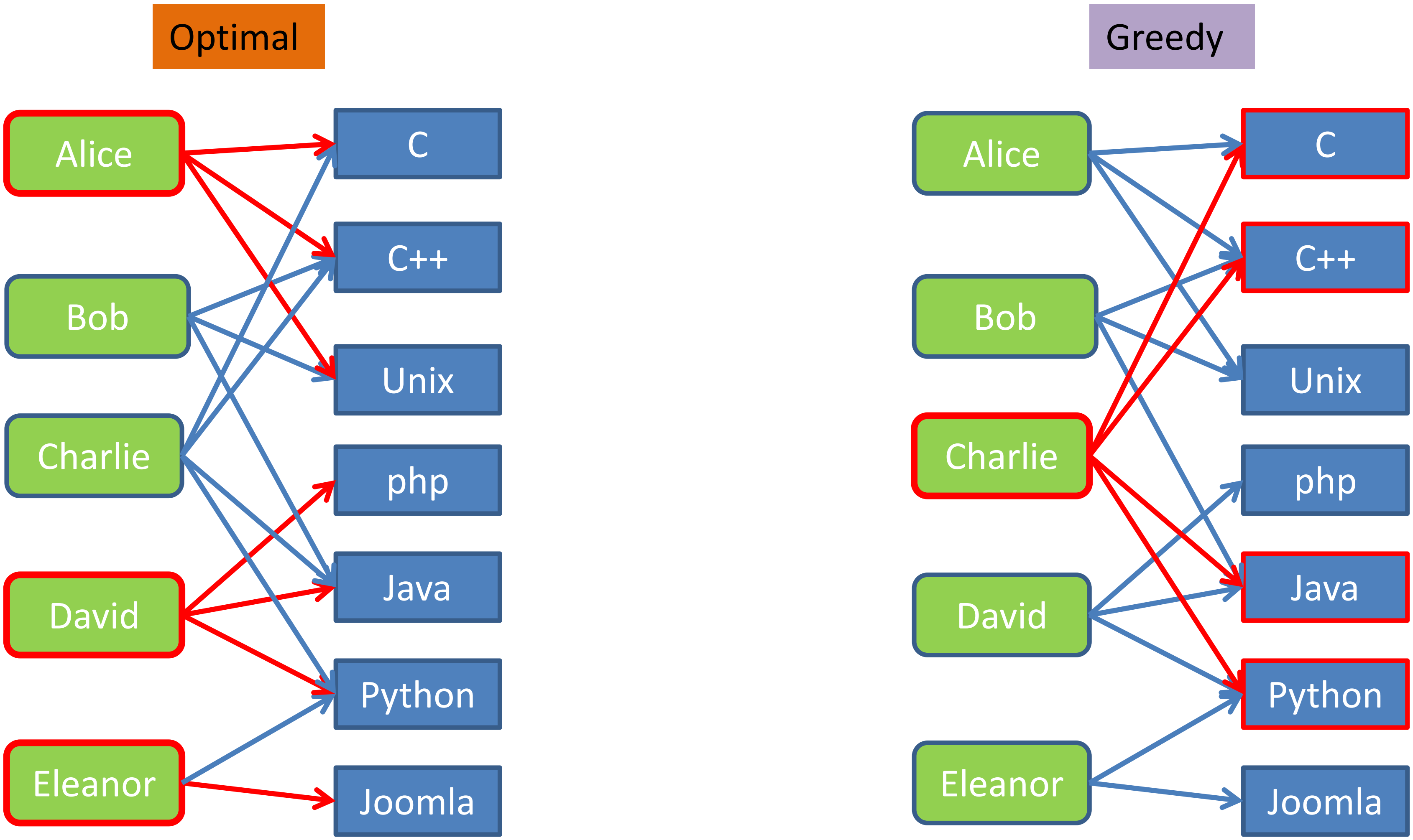
Greedy is not always optimal



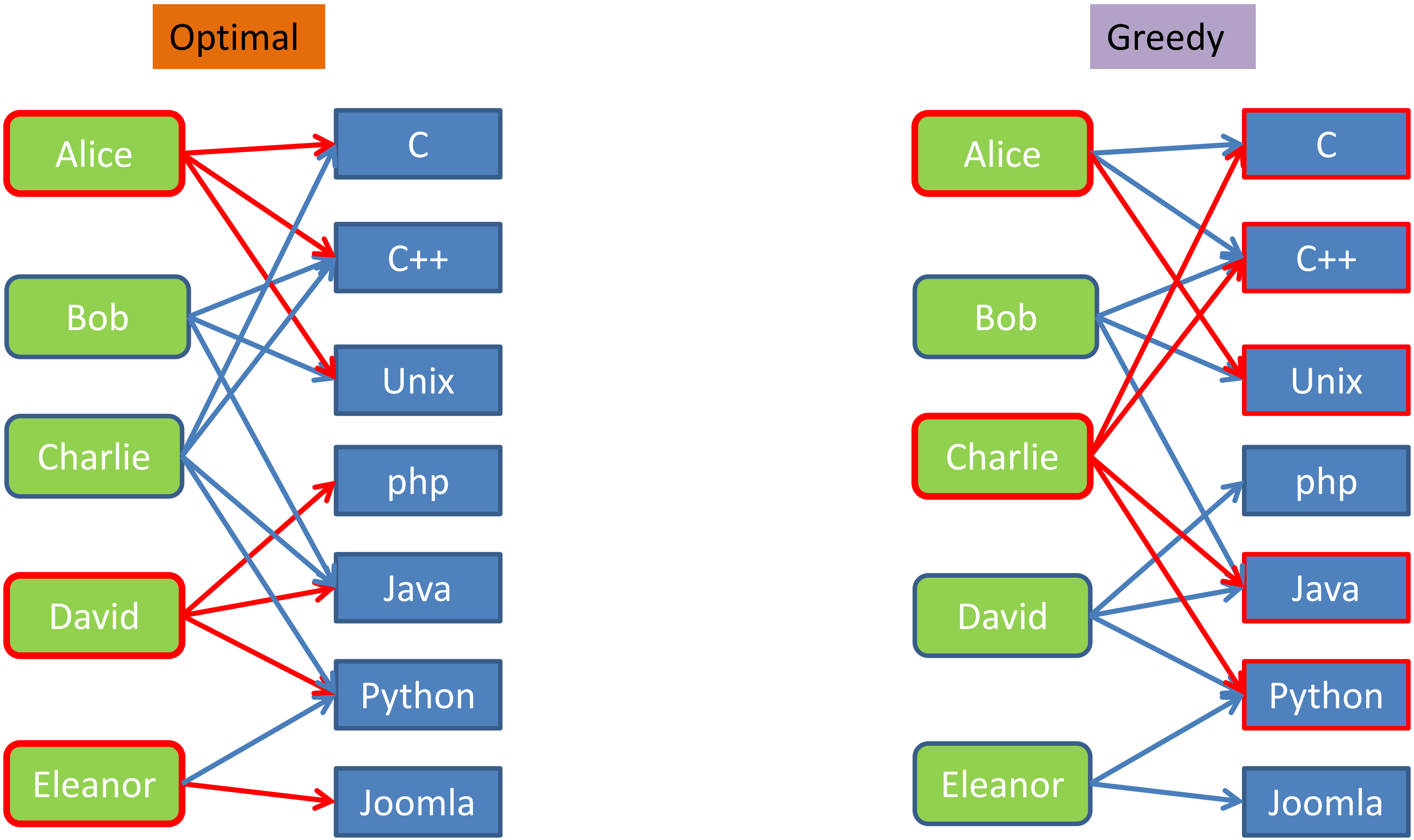
Greedy is not always optimal



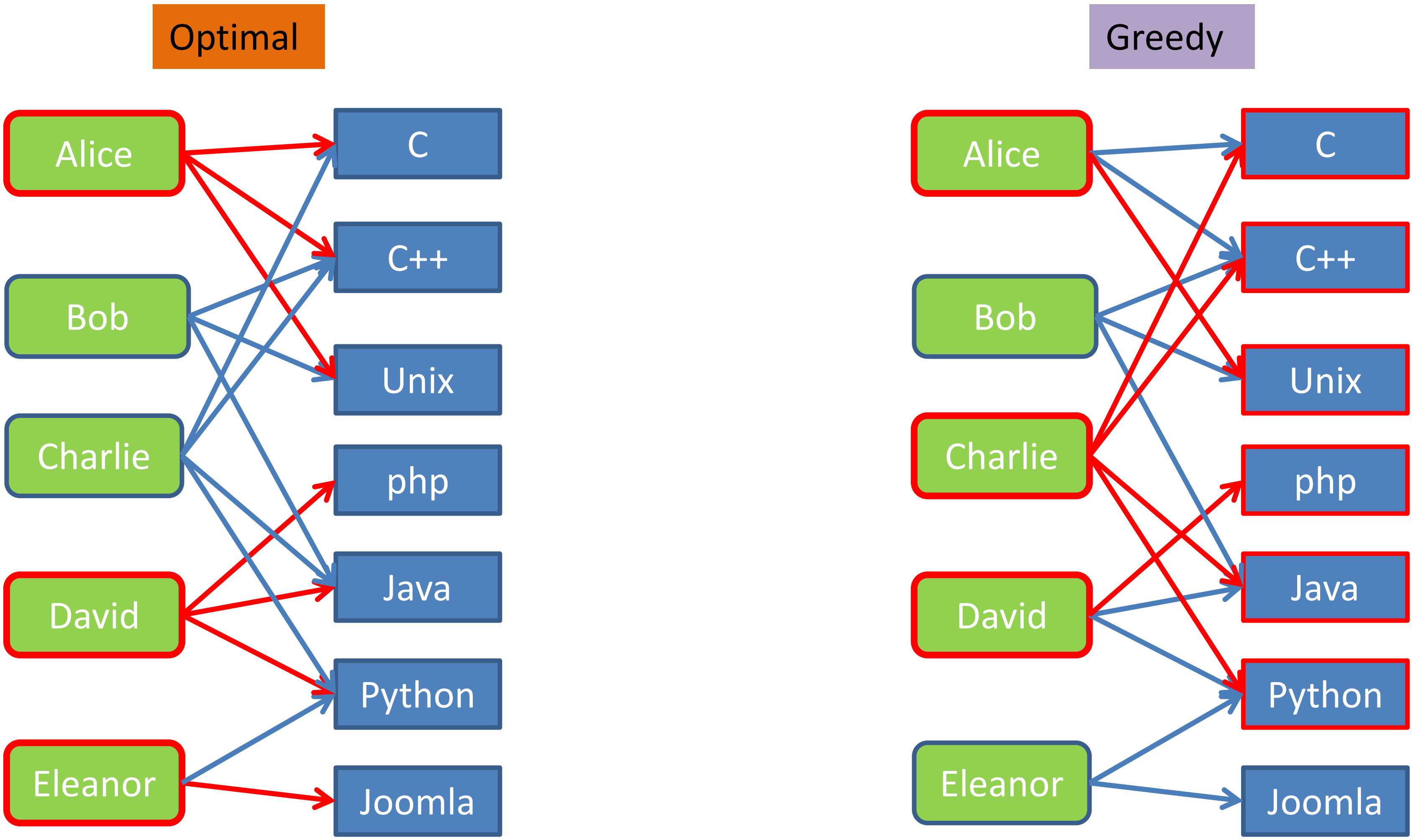
Greedy is not always optimal



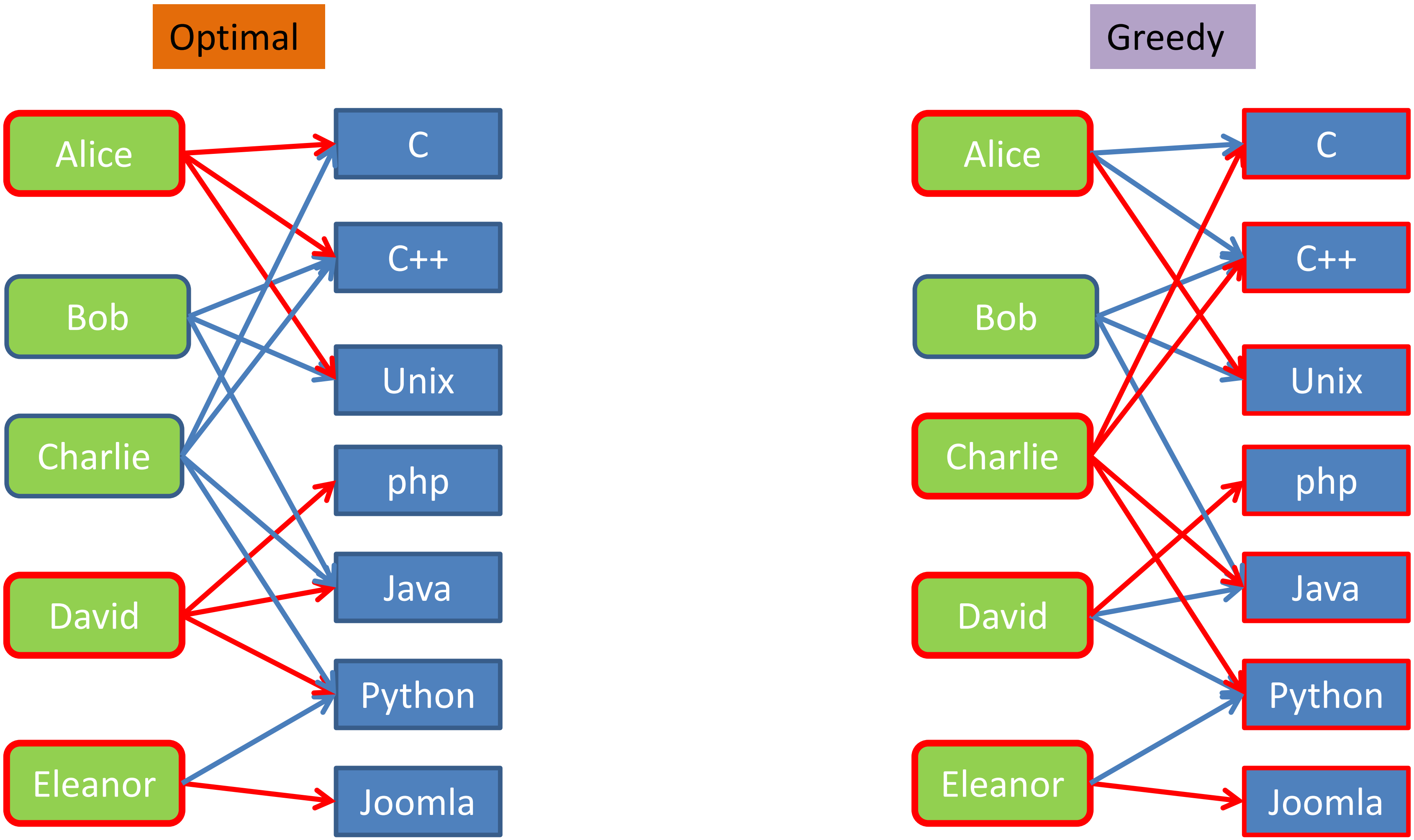
Greedy is not always optimal



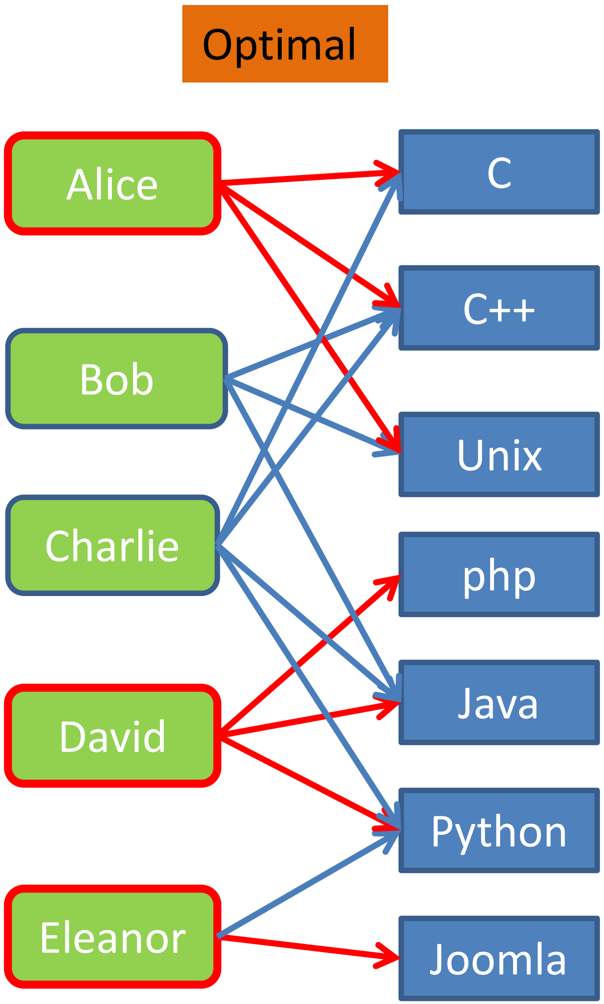
Greedy is not always optimal



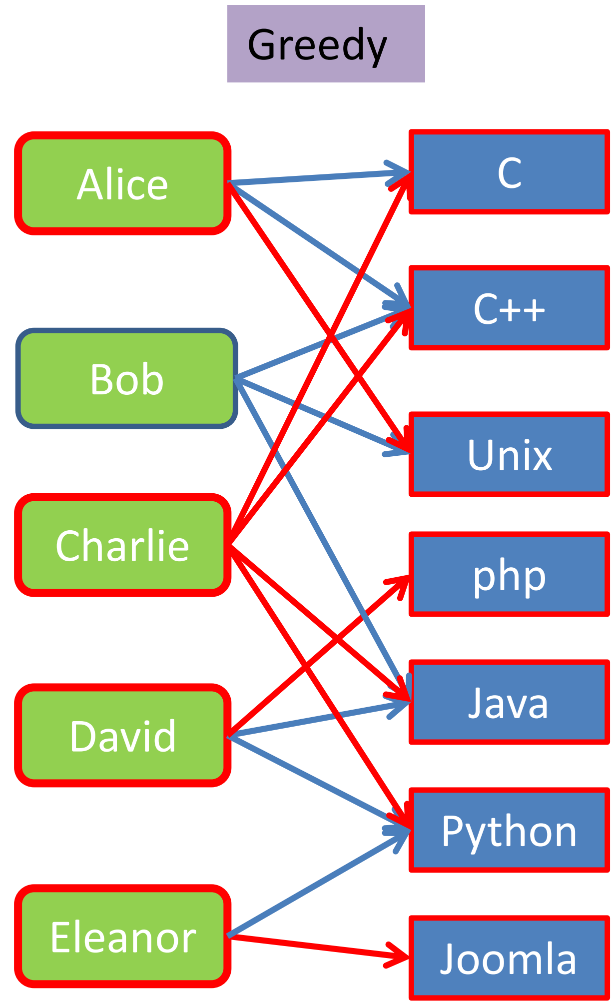
Greedy is not always optimal



Greedy is not always optimal



- Selecting Charlie is useless since we still need Alice and David
- Alice and David cover together a superset of the skills covered by Charlie



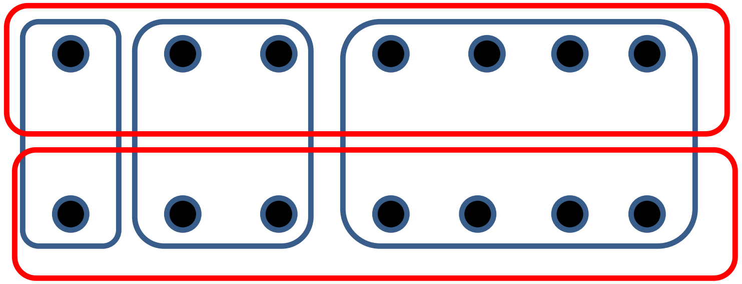
Approximation ratio of GREEDY

- Good news: GREEDY has approximation ratio:

$$\alpha = H(|S_{\max}|) = 1 + \ln|S_{\max}|, \qquad H(n) = \sum_{k=1}^{n} \frac{1}{k}$$

$$GREEDY(X) \leq (1 + \ln|S_{\max}|)OPT(X), \text{ for all X}$$

- The approximation ratio is tight up to a constant
  - Tight means that we can find a counter example with this ratio



OPT(X) = 2
GREEDY(X) = logN
α=½logN

Team formation in the presence of a social network

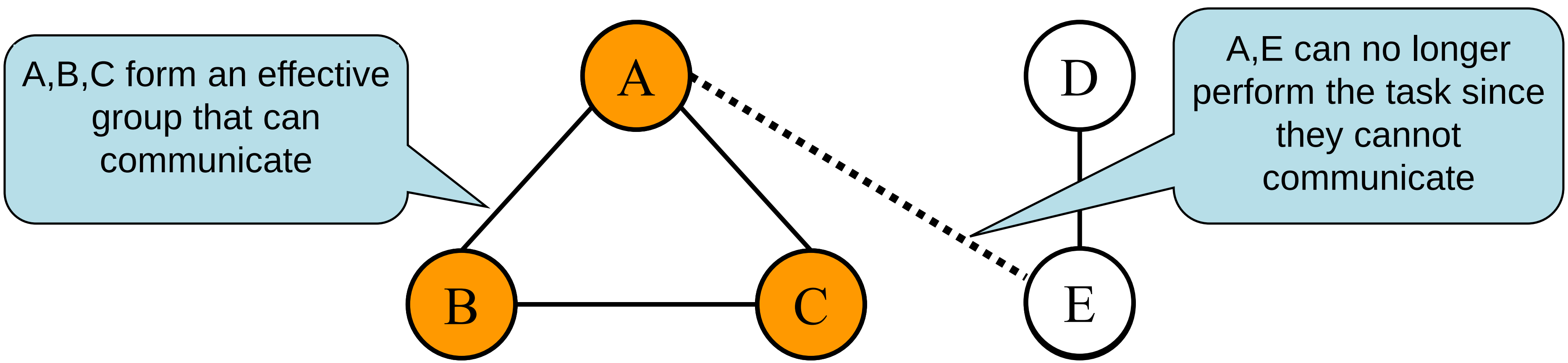
- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task
- Task: set of required skills
- Expert: has a set of skills
- Network: relationships and their strength
- Effectively: There is good communication between the team members
  - What does good mean? E.g., all team members are connected.

Coverage is NOT enough

T={algorithms,java,graphics,python}

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

Alice and Eleanor are the smallest team that covers all skills

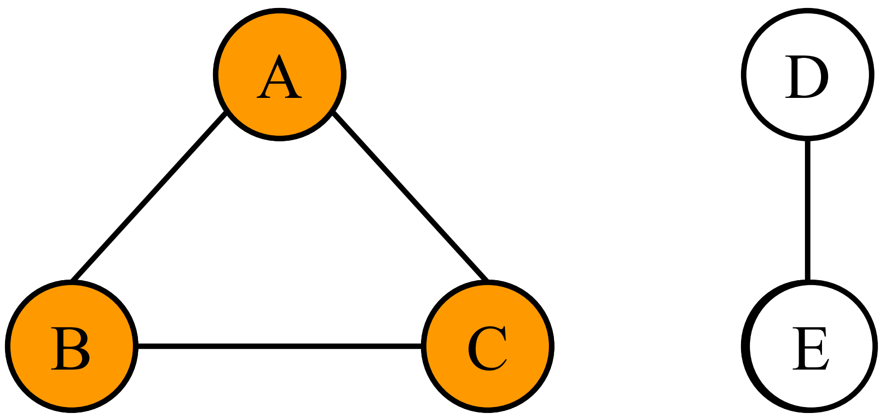


Communication: the members of the team must be able to efficiently communicate and work together

How to measure effective communication?

The longest shortest path between any two nodes in the subgraph

- Diameter of the subgraph defined by the group members

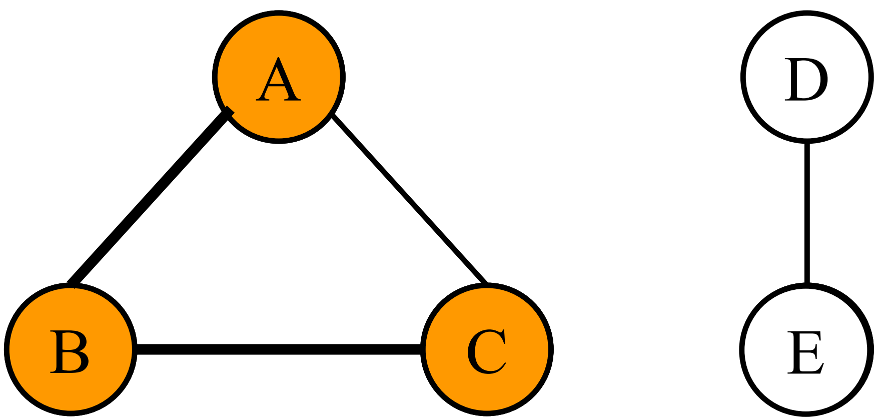


diameter = 1

How to measure effective communication?

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

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



MST = 2

Problem definition (MinDiameter)

- 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 G' in G with the minimum diameter.

- Problem is NP-hard
- Equivalent to the Multiple Choice Cover (MCC)
  - We have a set cover instance $(U, \boldsymbol{S})$, but we also have a distance matrix D with distances between the different sets in $\boldsymbol{S}$.
  - We want a cover that has the minimum diameter (minimizes the largest pairwise distance in the cover)

The RarestFirst algorithm

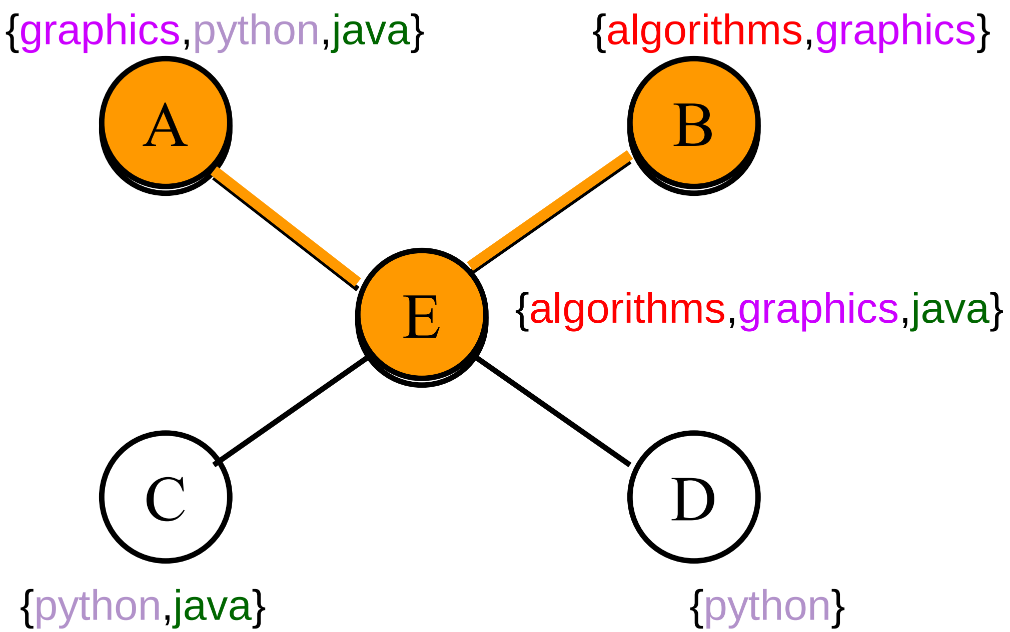
- Compute all shortest path distances in the input graph G and create a new complete graph G_C
- Find Rarest skill α_{rare} required for a task
- S_{rare} = group of people that have α_{rare}
- Evaluate star graphs in G_C, centered at individuals from S_{rare}
- Report cheapest star

Running time: Quadratic to the number of nodes

Approximation factor: 2×OPT

The RarestFirst algorithm

T={algorithms,java,graphics,python}



Skills:

- algorithms
- graphics
- java
- python

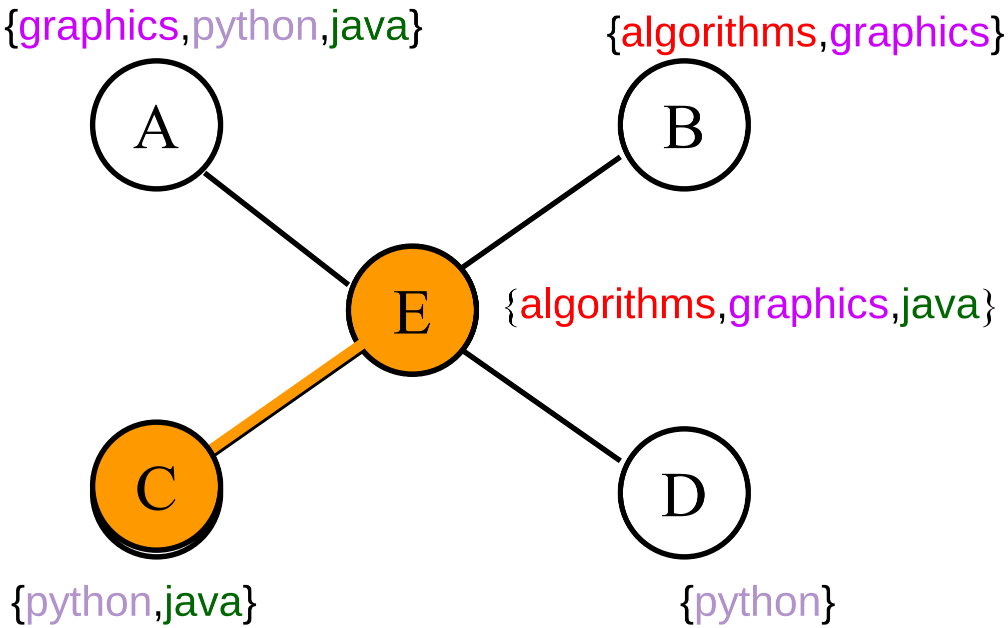
α_{rare} = algorithms

S_{rare} ={Bob, Eleanor}

Diameter = 2

The RarestFirst algorithm

T={algorithms,java,graphics,python}



Skills:

algorithms

graphics

java

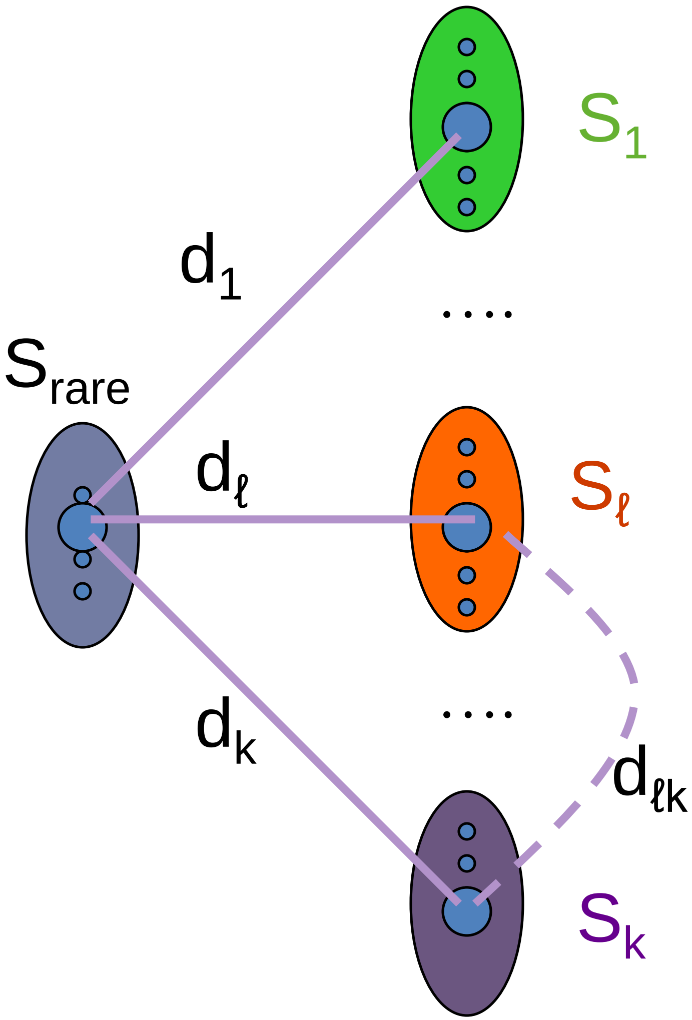
python

α_{rare} = algorithms

S_{rare} ={Bob, Eleanor}

Diameter = 1

Analysis of RarestFirst



- The diameter is
  - either $D = d_k$, for some node k,
  - or $D = d_{\ell k}$ for some pair of nodes ℓ, k
- Fact: $OPT \geq d_k$
- Fact: $OPT \geq d_\ell$
- $D \leq d_{\ell k} \leq d_\ell + d_k \leq 2{*}OPT$

Problem definition (MinMST)

- 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 G' in G with the minimum MST cost.

- Problem is NP-hard
- Follows from a connection with Group Steiner Tree problem

The SteinerTree problem

- Graph G(V,E)

Required vertices

- Partition of V into V = {R,N}

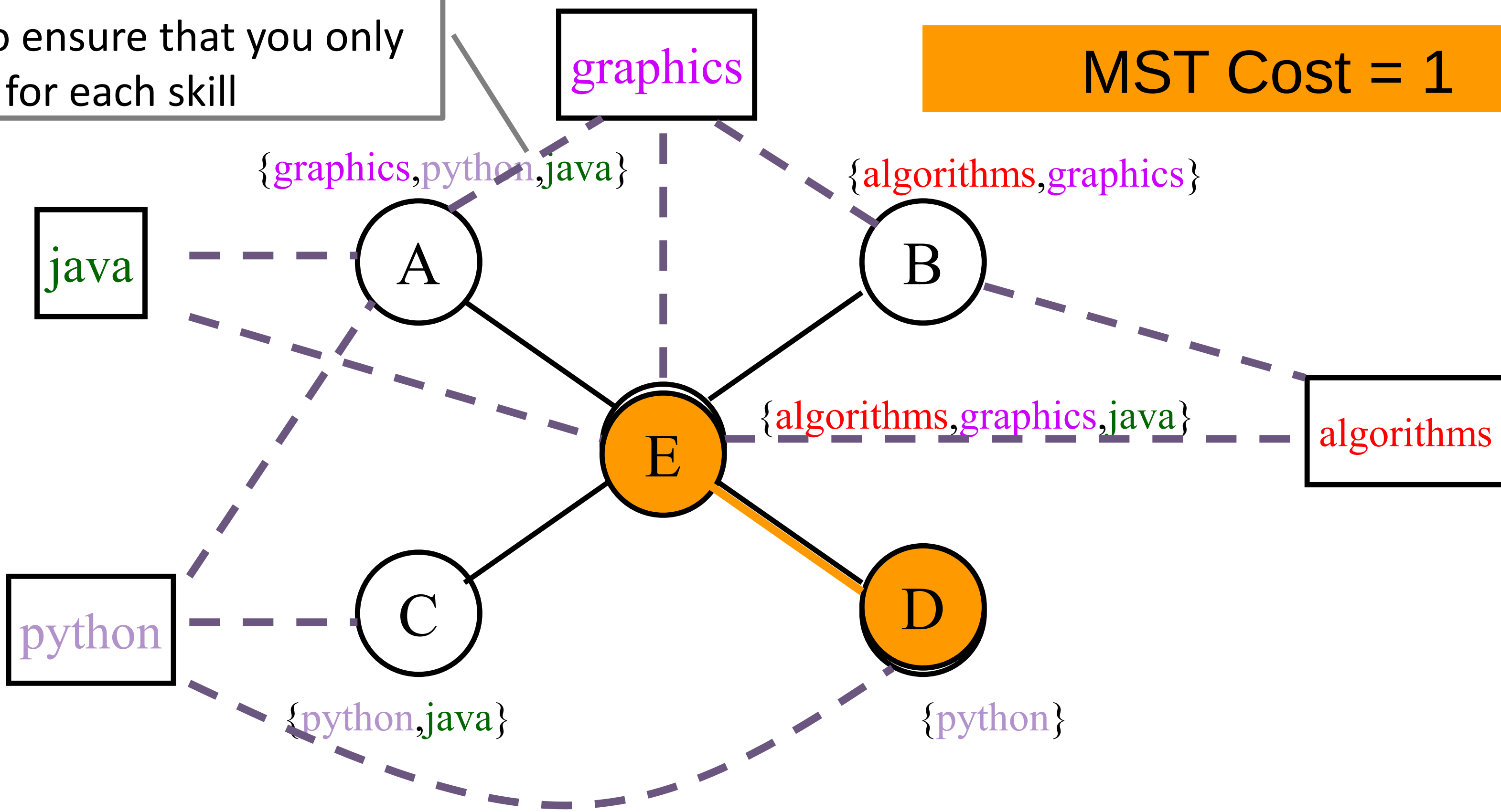
- Find G' subgraph of G such that G' contains all the required vertices (R) and MST(G') is minimized
  - Find the cheapest tree that contains all the required nodes.

The EnhancedSteiner algorithm

Put a large weight on the new edges (more than the sum of all edges) to ensure that you only pick one for each skill

T={algorithms,java,graphics,python}

MST Cost = 1



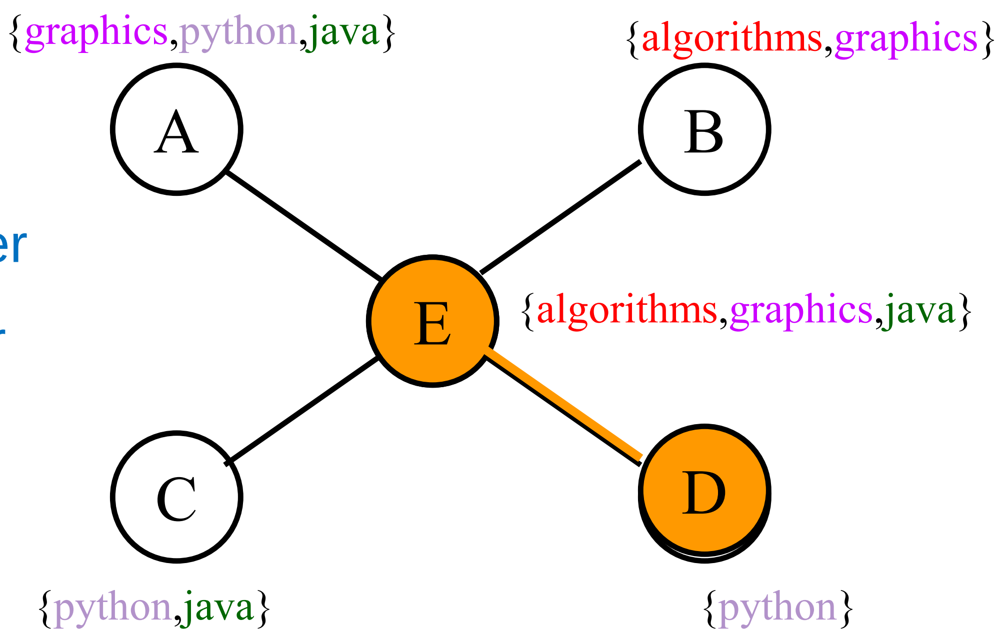
Add the skills as new nodes in the graph, connected to the graph nodes that have the skill

Solve the Steiner Tree on this graph, with the skill nodes being required

The CoverSteiner algorithm

T={**algorithms**,**java**,**graphics**,**python**}

1. Solve SetCover
2. Solve Steiner

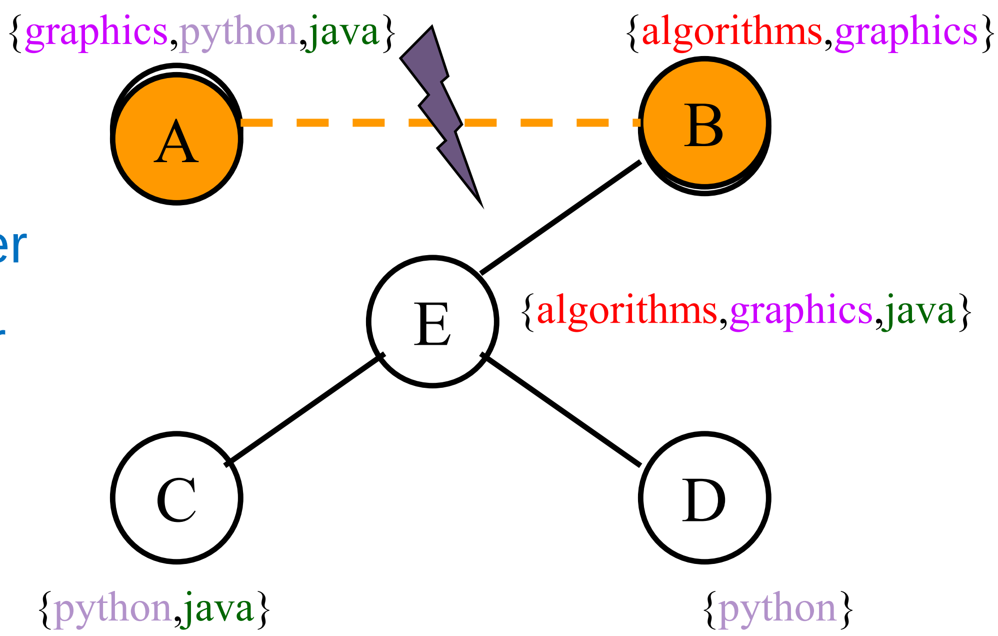


MST Cost = 1

How good is CoverSteiner?

T={algorithms,java,graphics,python}

1. Solve SetCover
2. Solve Steiner



MST Cost = Infty

References

Theodoros Lappas, Kun Liu, Evimaria Terzi, Finding a team of experts in social networks. KDD 2009: 467-476