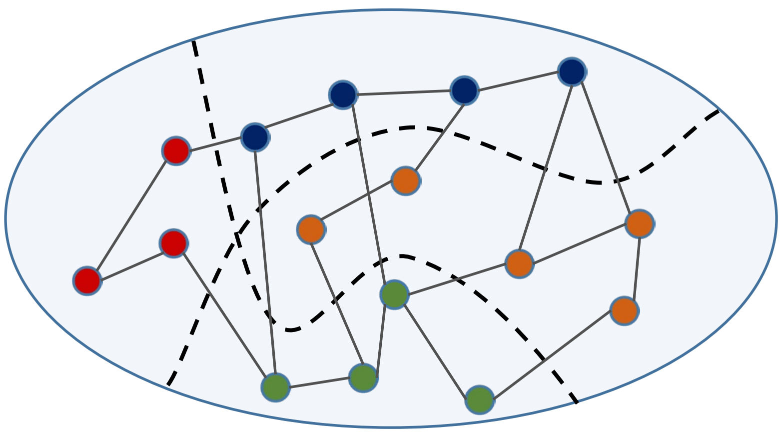
Algorithms for perturbation resilient problems

Instructor: Yury Makarychev, TTIC

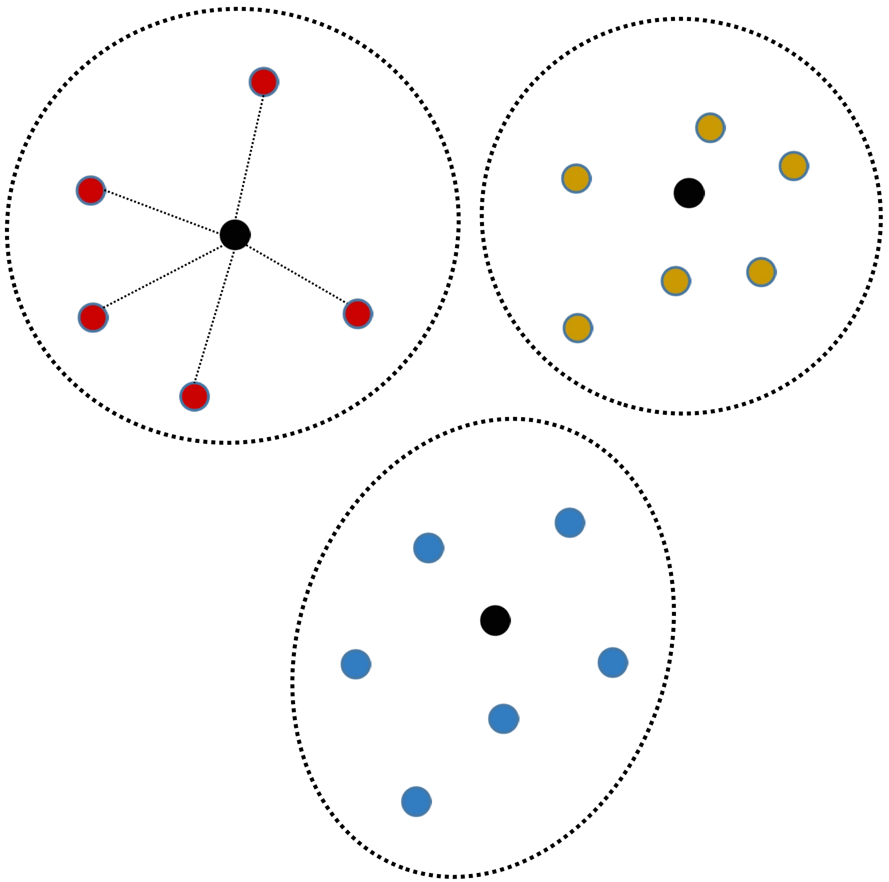
Max Planck ADFOCS 2023

Practice

Need to solve combinatorial optimization and clustering problems



Multiway Cut



k-means

Theory

Many of these problems are NP-hard and cannot be solved exactly in polynomial time.

Traditional approach

- Don't make any assumptions about the input.
- Design an approximation algorithm for the worst case.

Recall: an algorithm has an α-approximation if

$ALG \geq OPT/\alpha$ for a maximization problem

$ALG \leq \alpha\, OPT$ for a minimization problem

Beyond-Worst-Case Analysis

- Real-life instances appear to be much easier than worst-case instances.
- Heuristics used in practice often get much better approximation than it is theoretically possible for worst-case instances.

➢ Why is it the case?

➢ Create good models for real-life instances.

➢ Design algorithms that solve instances from these models.

Two Approaches to Modelling Real-life Instances

Assume that an instance satisfies certain structural properties:

- Perturbation Resilience
- Assumptions of the graph, weights, etc

Generative models. Assume that an instance is generated in a certain way:

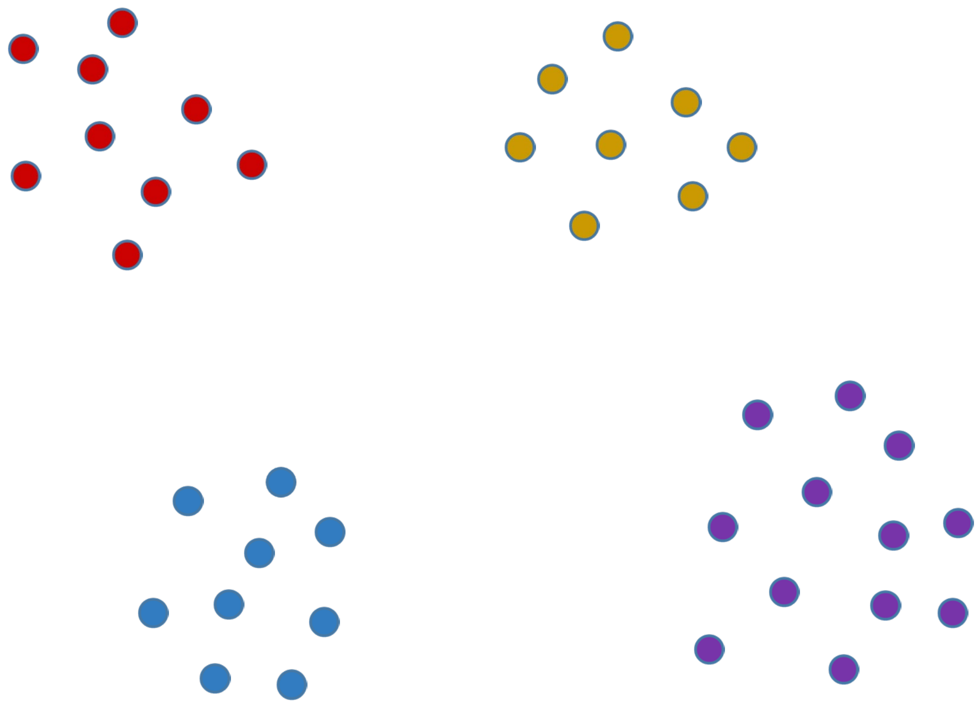
- Random models: e.g. G is a $G(n, p)$ graph
- Semirandom models: random + adversarial choices

Perturbation Resilience

Bilu and Linial ‘10

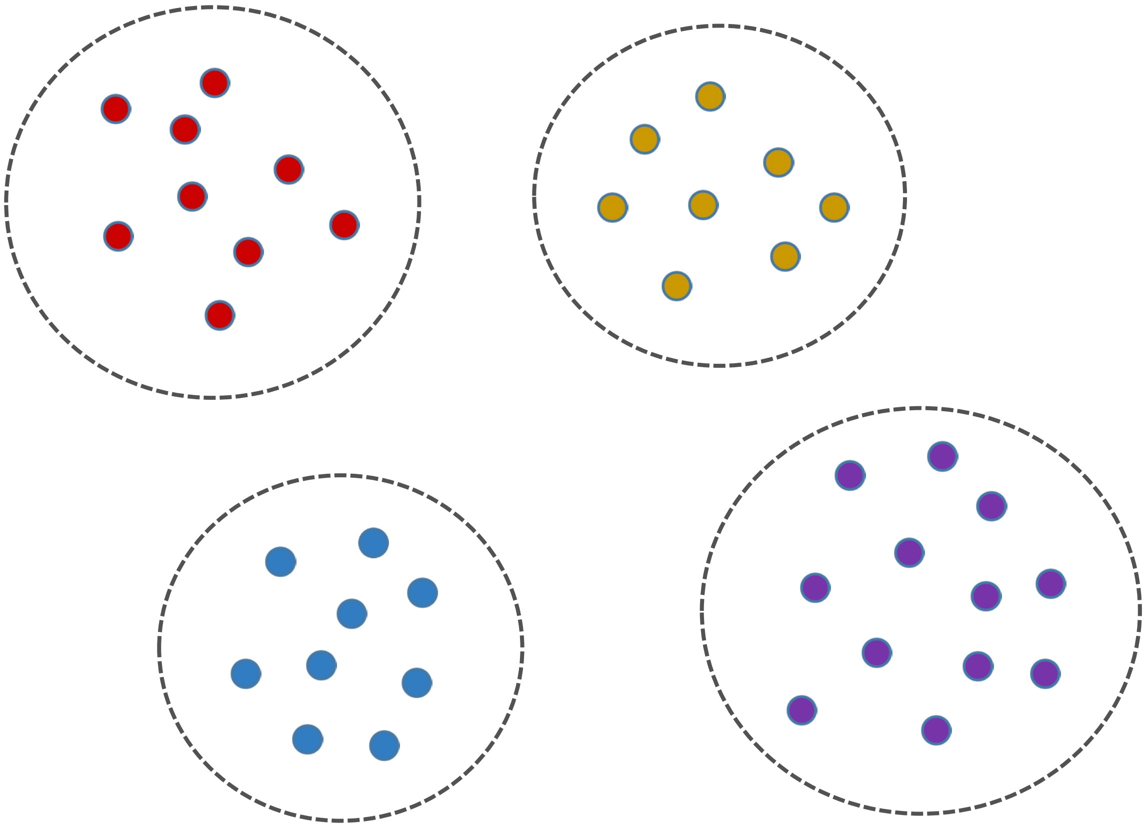
Warm up

Cluster the following data set.



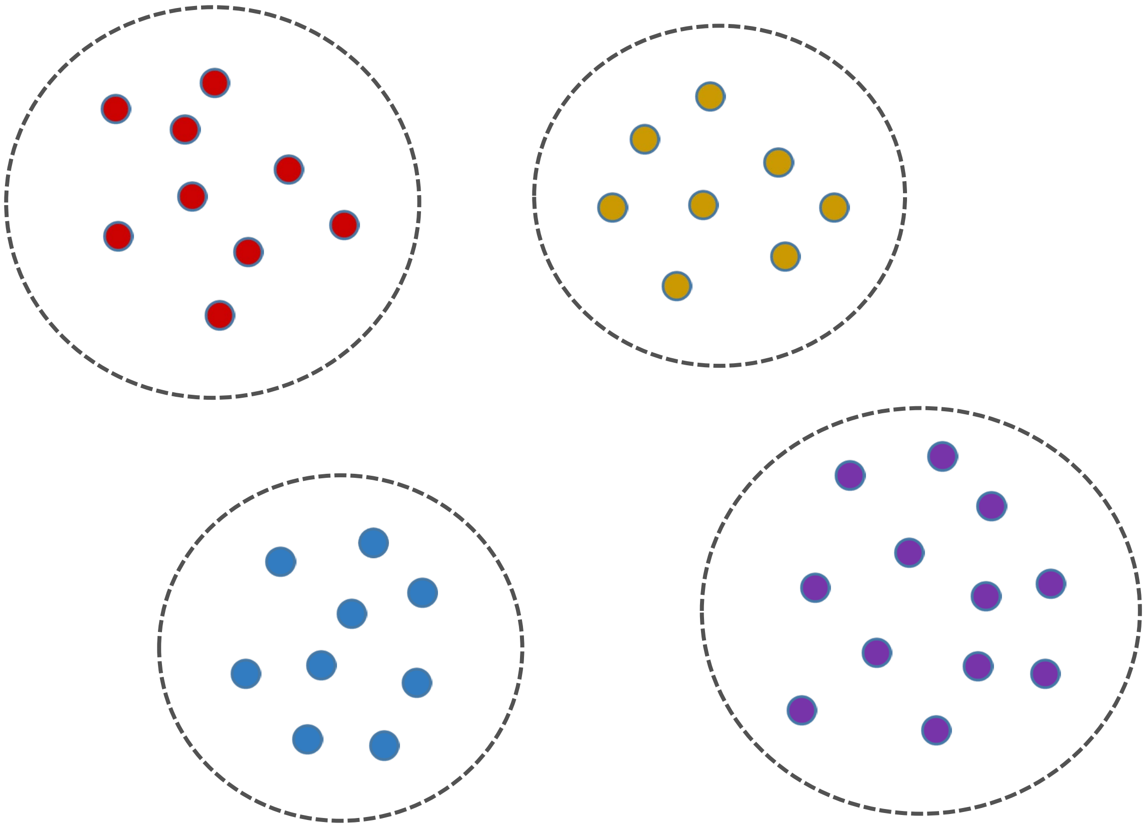
Warm up

Cluster the following data set.



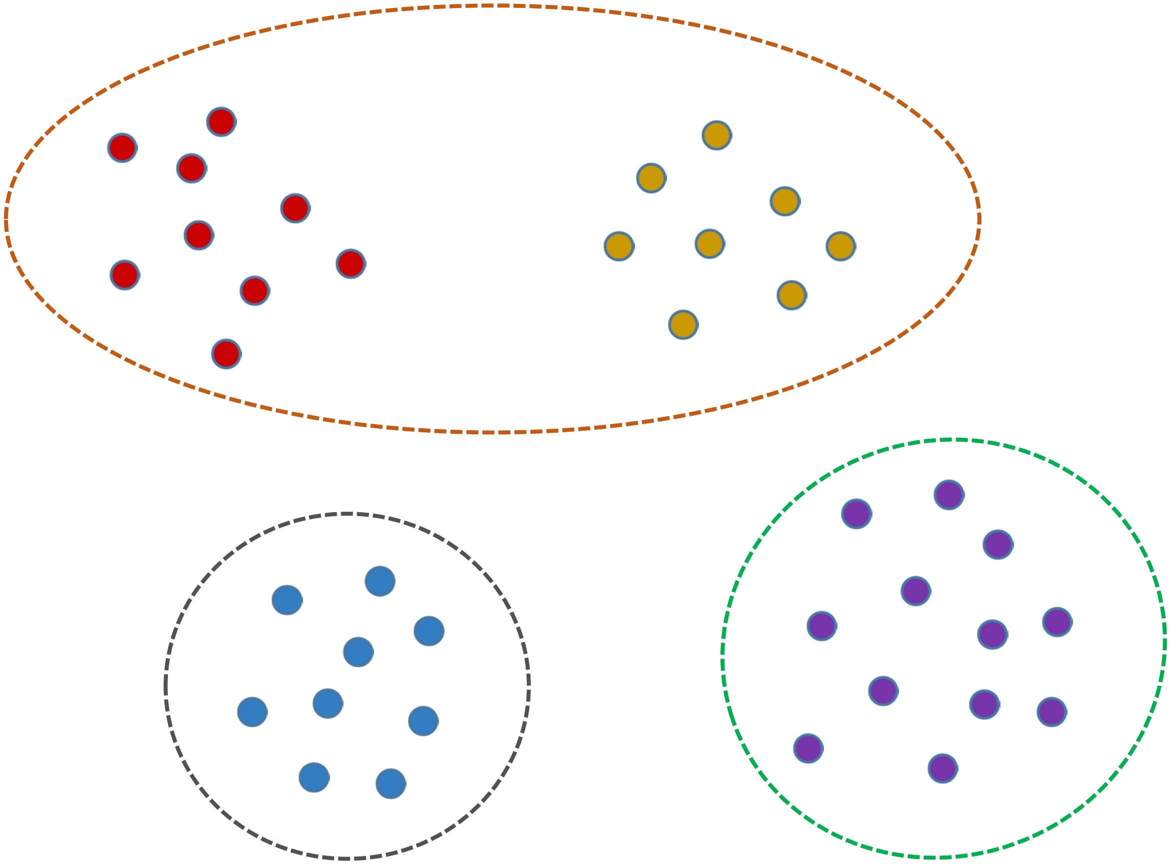
Warm up

Cluster the following data set in 3 groups.



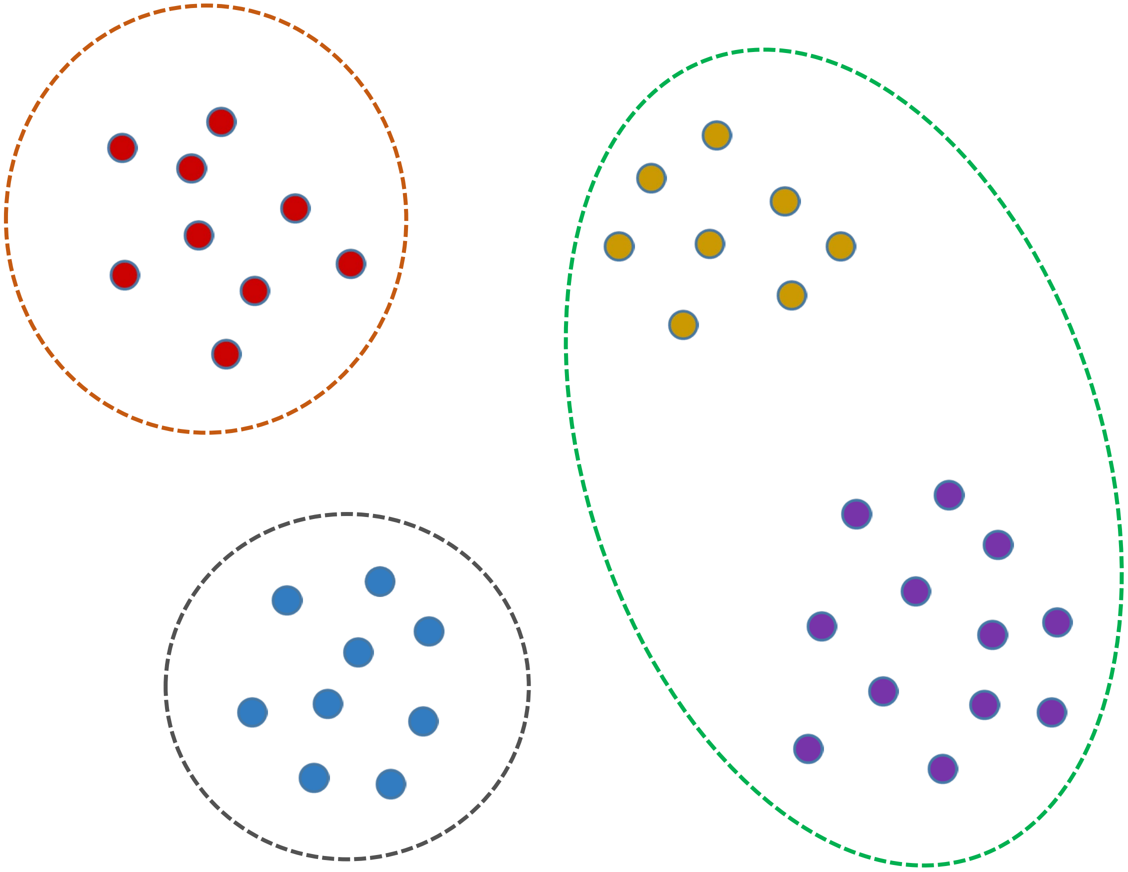
Warm up

Cluster the following data set in 3 groups.



Warm up

Cluster the following data set in 3 groups.



"Clustering is difficult only when it does not matter."

Daniely, Linial, Saks

When do solutions matter?

Bilu and Linial '10:

> Optimal solutions matter when they are unique and stand out among other solutions.

When do solutions matter?

Bilu and Linial '10:

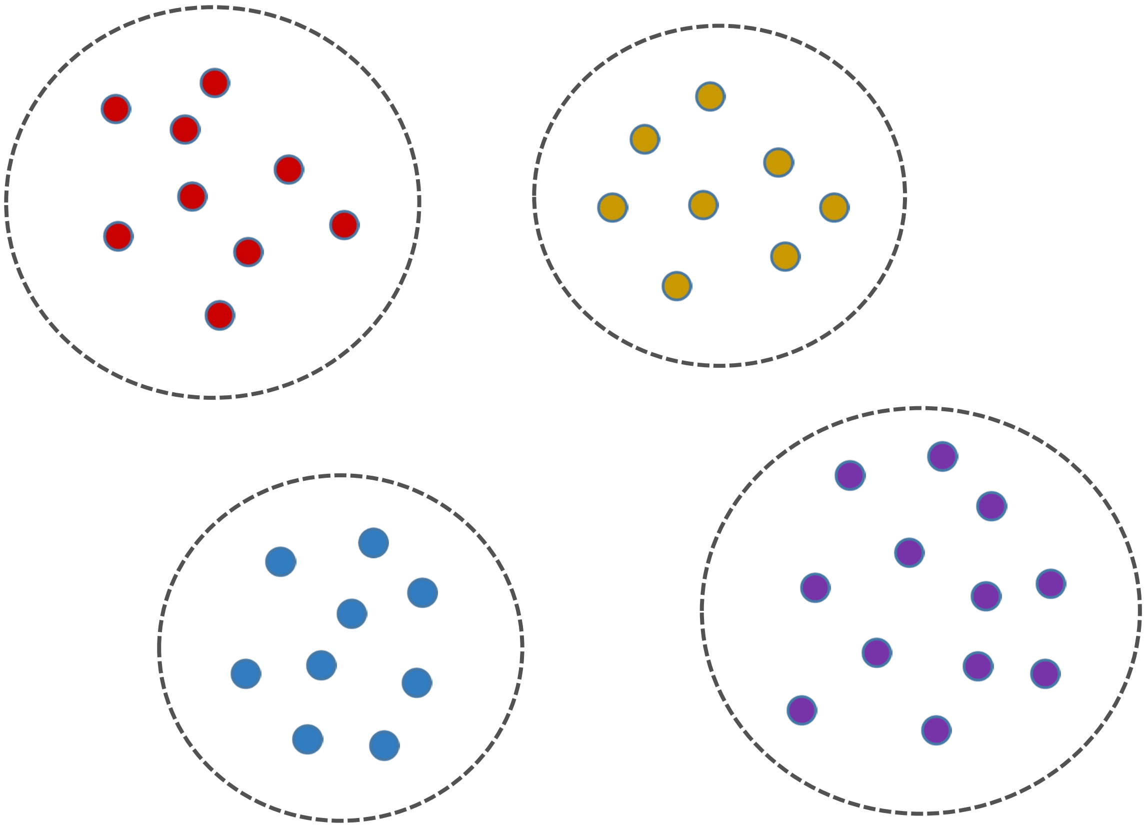
> Optimal solutions matter when they are unique and stand out among other solutions.

An instance of a problem is perturbation resilient if

> the optimal solution remains the same when
> we perturb the instance.

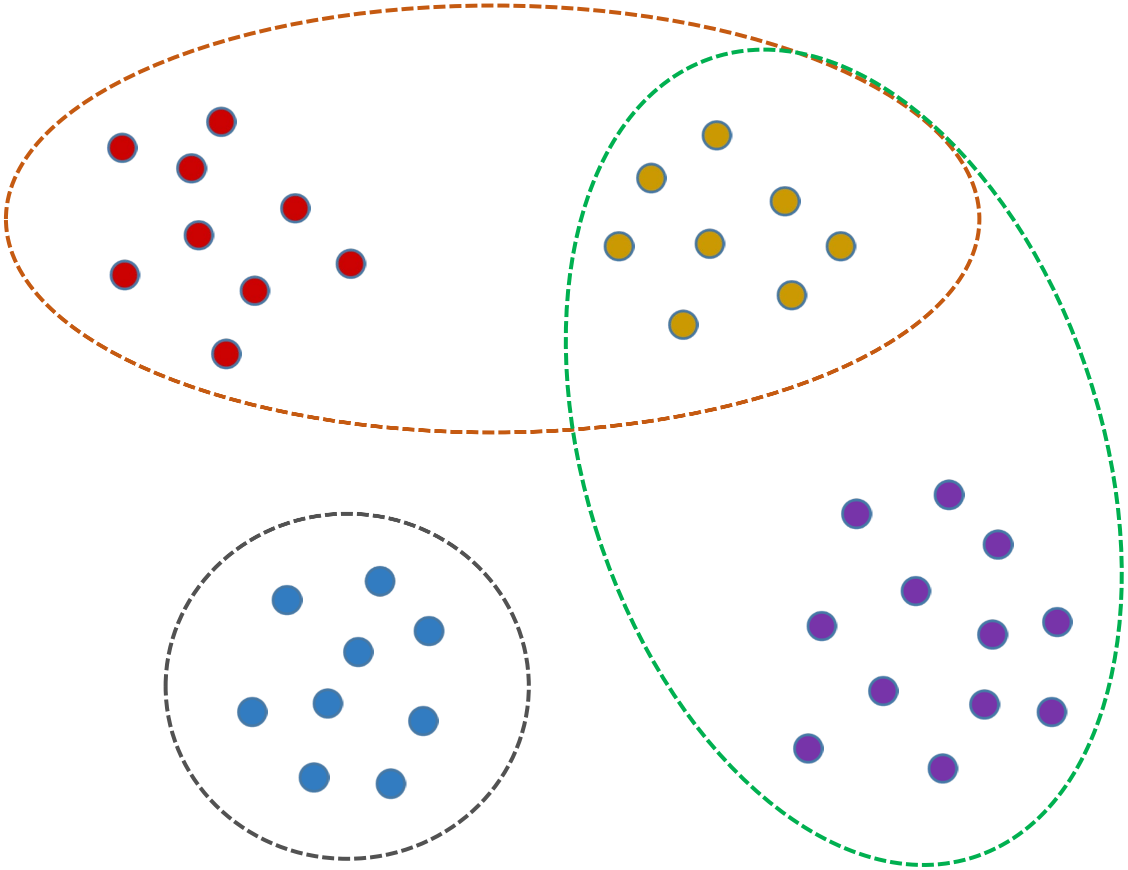
Perturbation-resilient Instance

Cluster the following data set in 4 groups.



Non-PR Instance

Cluster the following data set in 3 groups.



Perturbation-resilience

- Consider an instance $\mathcal{I}$ of an optimization or clustering problem. Assume that it has a number of parameters

$$p_1, \ldots, p_m > 0$$

The parameters may be edge, vertex, or constraint weights, or distances between points.

- $\mathcal{I}'$ is a *γ-perturbation* of $\mathcal{I}$ if it can be obtained from $\mathcal{I}$ by "perturbing the parameters" — multiplying each p_i by a number from 1 to γ.

$$p_i \leq p_i' \leq \gamma \cdot p_i$$

Perturbation-resilience

[Bilu and Linial '10] An instance $\mathcal{I}$ of an optimization or clustering problem is γ-perturbation-resilient if the optimal solution remains the same when we perturb the instance:

every γ-perturbation $\mathcal{I}'$ has the same optimal solution as $\mathcal{I}$

(the value/cost of the solution may be different)

Perturbation-resilience

Every γ-perturbation $\mathcal{I}'$ has the same optimal solution as $\mathcal{I}$.

- Empirical evidence shows: the optimal solution often "stands out" among all other solutions [Bilu, Linial]
- In ML, we want to find the "true" solution.
    - Make many somewhat arbitrary choices; e.g. choose one similarity function among several options
    - If the instance is not p.r., the optimal solution will be different from the true solution.

Weak perturbation-resilience

[Makarychev, M, Vijayaraghavan '14]

An instance $\mathcal{I}$ of an optimization or clustering problem is γ-weakly perturbation-resilient if the optimal solution for every γ-perturbation $\mathcal{I}'$ of $\mathcal{I}$ is "close" to the optimal solution for $\mathcal{I}$.

Goal: Exact algorithms

- Design exact algorithms for γ-perturbation resilient instances.
- Design an algorithm that finds a solution "close" to an optimal solution for weakly γ-perturbation resilient instances.
- We want γ to be small.

k-means and k-median

Given a set of points X, distance $d(\cdot,\cdot)$ on X, and k

Partition X into k clusters $C_1, \ldots, C_k$ and find a "center" c_i in each C_i so as to minimize

$$\sum_{i=1}^{k} \sum_{u \in C_i} d(u, c_i) \qquad (k\text{-median})$$

$$\sum_{i=1}^{k} \sum_{u \in C_i} d(u, c_i)^2 \qquad (k\text{-means})$$

Results

Results (clustering)

$\gamma \geq 3$	**k-center, k-means, k-median**	[Awasthi, Blum, Sheffet \`12]
$\gamma \geq 1 + \sqrt{2}$	**k-center, k-means, k-median**	[Balcan, Liang \`13]
$\gamma \geq 2$	**sym. /asym. k-center**	[Balcan, Haghtalab, White \`16]
$\gamma \geq 2$	**k-means, k-median**	[Angelidakis, Makarychev, M \`17]

Results (optimization)

$\gamma \geq cn$	**Max Cut**	[Bilu, Linial `10]
$\gamma \geq c\sqrt{n}$	**Max Cut**	[Bilu, Daniely, Linial, Saks `13]
$\gamma \geq c\sqrt{\log n}\log\log n$	**Max Cut**	[Makarychev, M, Vijayaraghavan `13]
$\gamma \geq 2 - 2/k$	**Multiway**	[AMM `17]

Results (optimization)

Our algorithms are robust.

- Find the optimal solution, if the instance is p.r.
- Find an optimal solution or detects that the instance is not p.r., otherwise.
- Never output an incorrect answer.

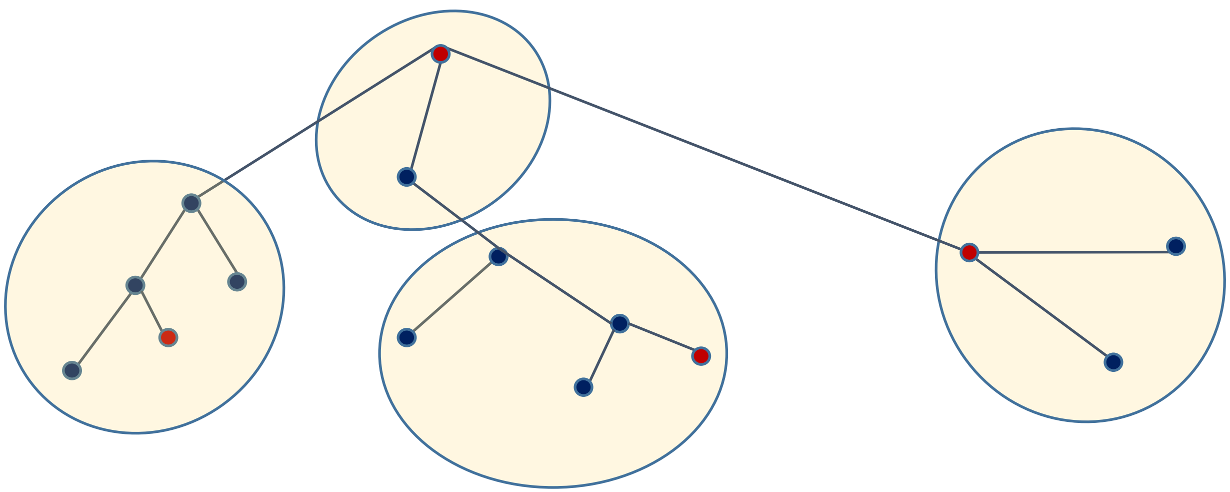
Solve weakly p.r. instances.

Algorithm for Clustering Problems

Plan [AMM `17]

i. γ-perturbation resilience ⇒ γ-center proximity

ii. 2-center proximity ⇒ each cluster is a subtree of the MST

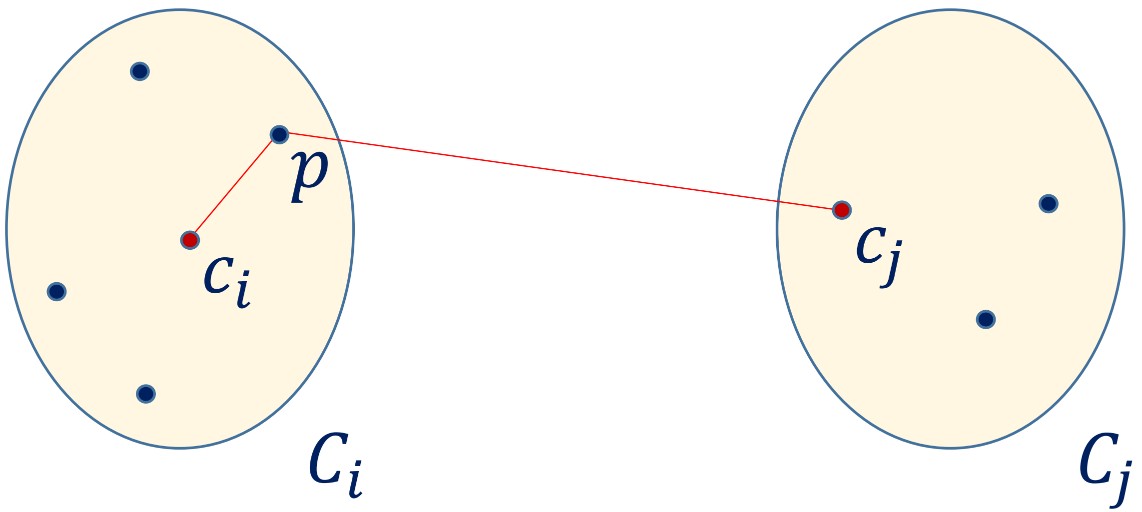


iii. use single-linkage + DP to find $C_1, \ldots, C_k$

Center proximity property

[Awasthi, Blum, Sheffet `12] A clustering $C_1, \ldots, C_k$ with centers $c_1, \ldots, c_k$ satisfies the center proximity property if for every $p \in C_i$:

$$d(p, c_j) > \gamma\, d(p, c_i)$$



Perturbation resilience ⇒ center proximity

Perturbation resilience: the optimal clustering doesn't change when we perturb the distances.

$$d(u,v)/\gamma \leq d'(u,v) \leq d(u,v)$$

[ABS `12] $d'(\cdot,\cdot)$ doesn't have to be a metric

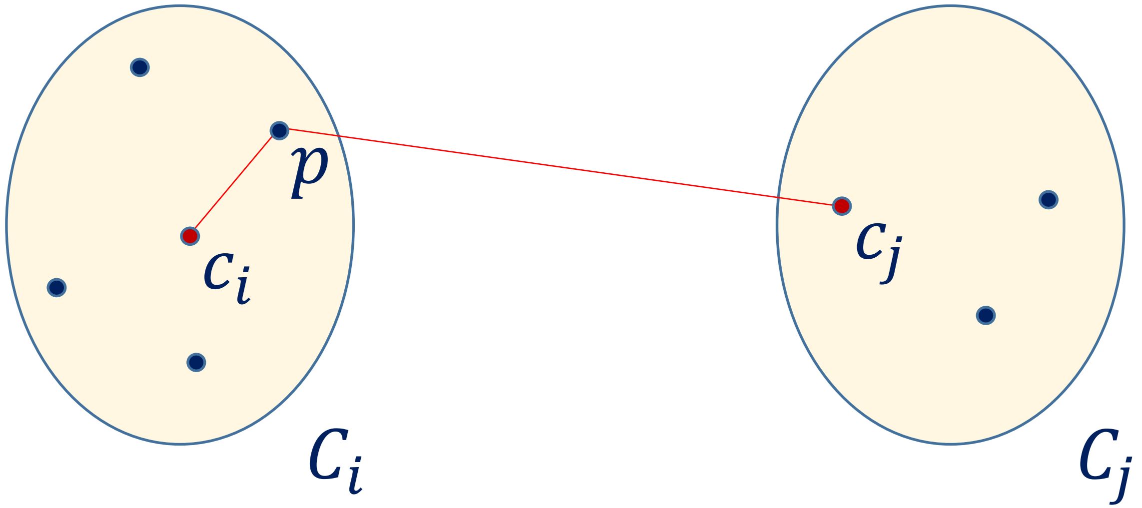
[AMM `17] $d'(\cdot,\cdot)$ is a metric

Metric perturbation resilience is a more natural notion.

Perturbation resilience ⇒ center proximity [ABS `12, AMM `17]

Assume center proximity doesn't hold.

Then $d(p, c_j) \leq \gamma\, d(p, c_i)$.

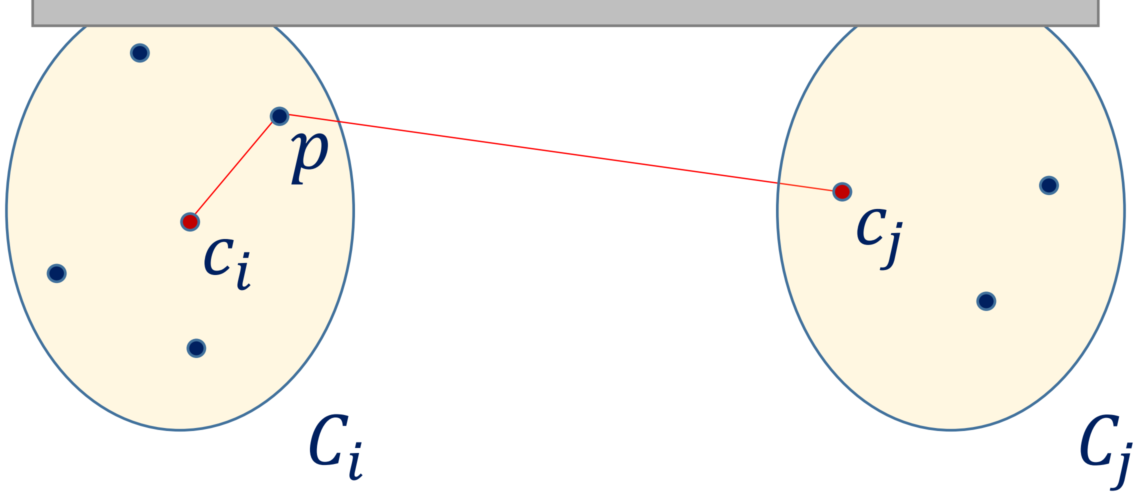


Perturbation resilience ⇒ center proximity [ABS `12, AMM `17]

Assume center proximity doesn't hold.

- Let $d'(p, c_j) = d(p, c_i) \geq \gamma^{-1} d(p, c_j)$.
- Don't c
- Consic

This is a γ-perturbation.

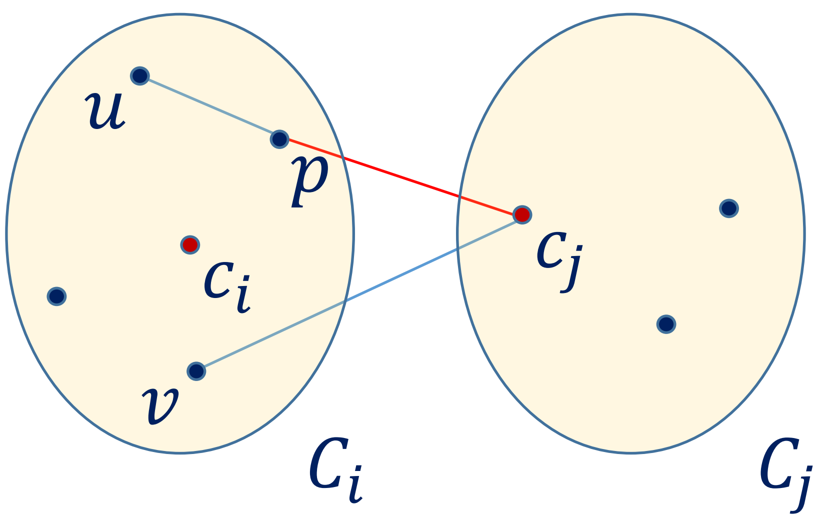


Perturbation resilience ⇒ center proximity [ABS `12, AMM `17]

Distances inside clusters C_i and C_j don't change.

Consider $u, v \in C_i$.

$$d'(u,v) = \min\begin{pmatrix} d(u,v), \\ d(u,p) + d'(p,c_j) + d(c_j,v) \end{pmatrix}$$

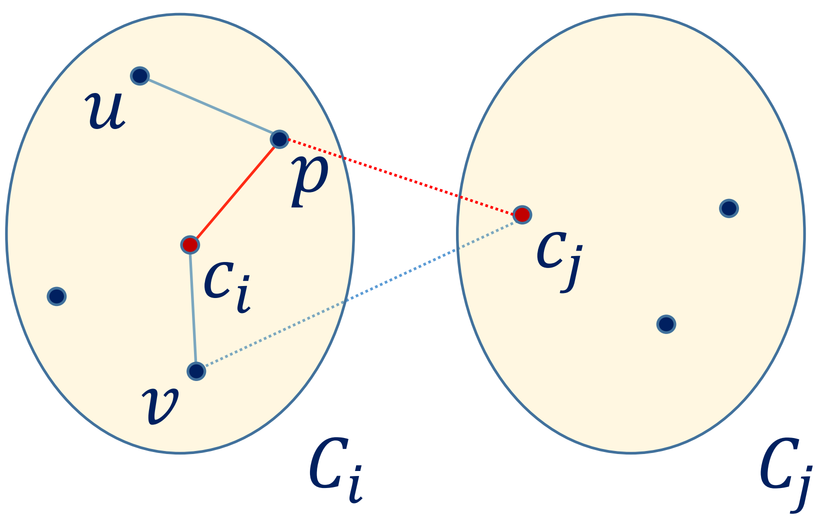


Perturbation resilience ⇒ center proximity [ABS `12, AMM `17]

Distances inside clusters C_i and C_j don't change.

Consider $u, v \in C_i$.

$$d'(u,v) = \min\begin{pmatrix} d(u,v), \\ d(u,p) + d'(p,c_j) + d(c_j,v) \end{pmatrix}$$

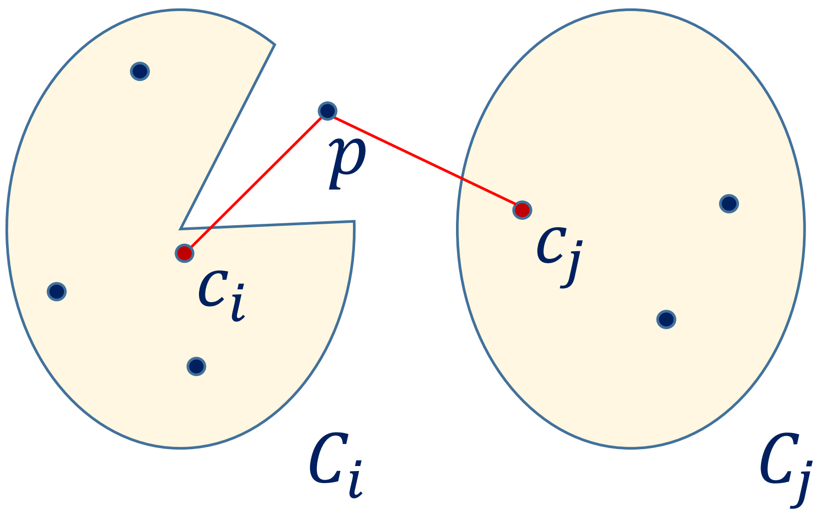


Perturbation resilience ⇒ center proximity [ABS `12, AMM `17]

Since the instance is γ-p.r., $C_1, \ldots, C_k$ must be the unique optimal solution for distance d'.

Still, c_i and c_j are optimal centers for C_i and C_j.

$$d'(p, c_i) = d'(p, c_j) \Rightarrow \text{can move } p \text{ from } C_i \text{ to } C_j$$

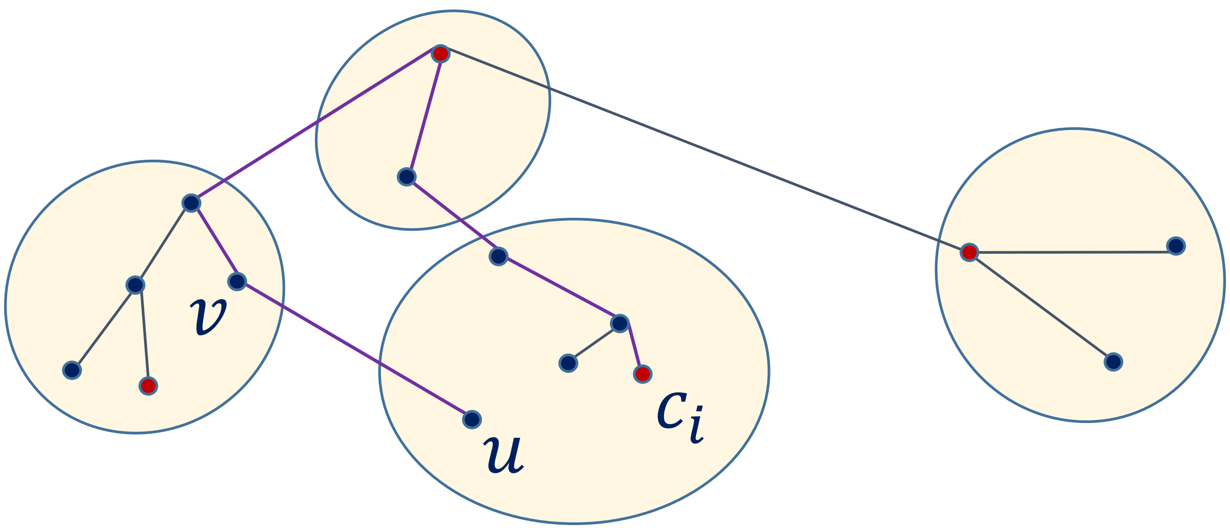


Each cluster is a subtree of MST

[ABS `12] 2-center proximity ⇒

every $u \in C_i$ is closer to c_i than to any $v \notin C_i$

Assume the path from $u \in C_i$ to c_i in MST, leaves C_i.

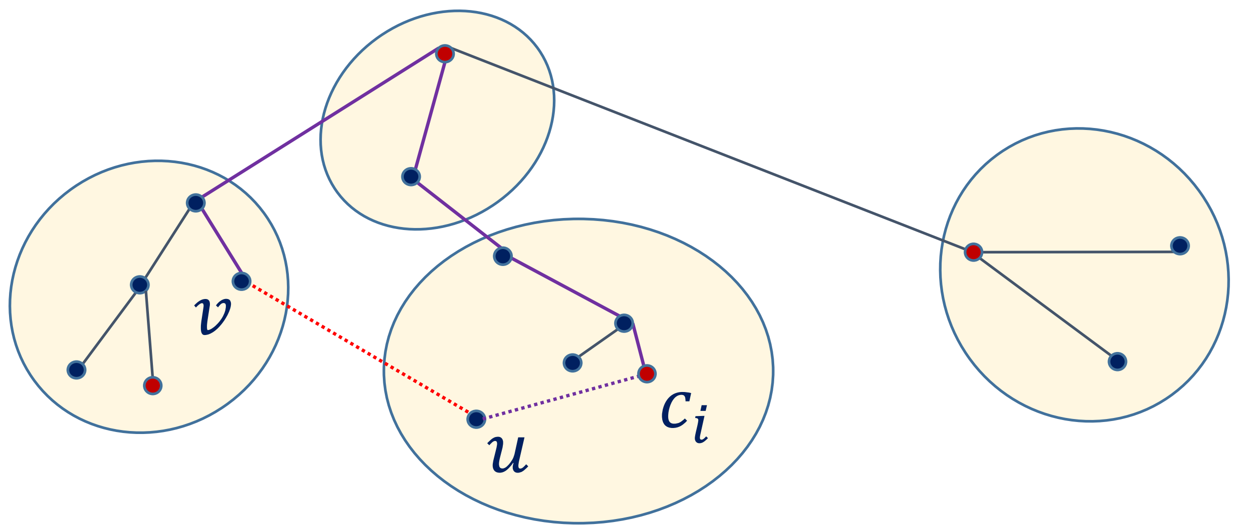


Each cluster is a subtree of MST

[ABS `12] 2-center proximity ⇒

every $u \in C_i$ is closer to c_i than to any $v \notin C_i$

Assume the path from $u \in C_i$ to c_i in MST, leaves C_i.

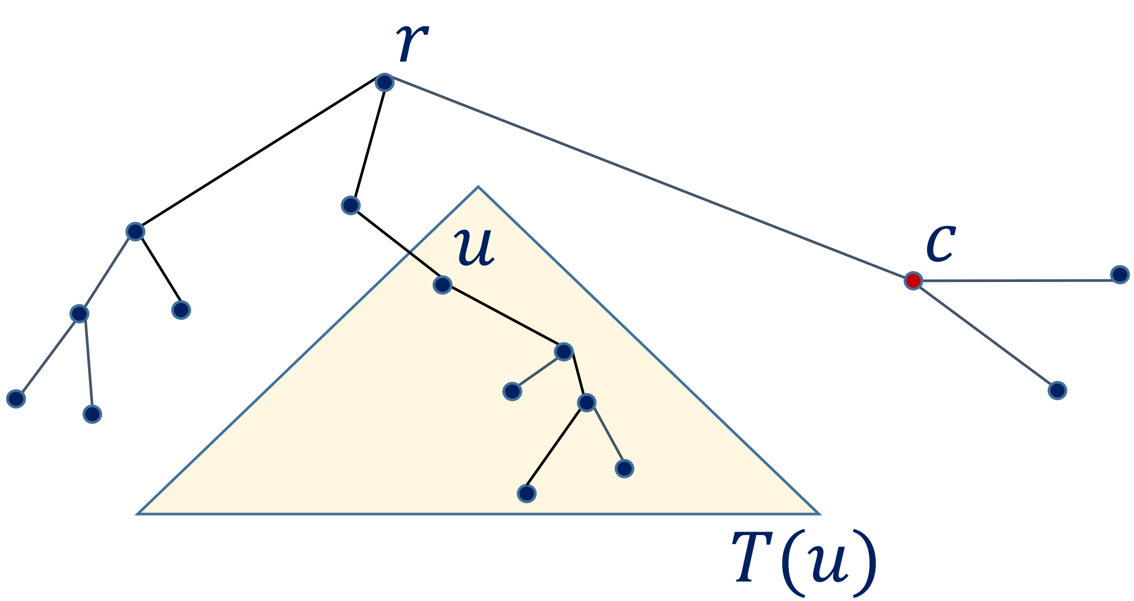


Dynamic programming algorithm

Root MST at some r. $T(u)$ is the subtree rooted at u.

$\mathrm{cost}_u(j, c)$: the cost of partitioning $T(u)$

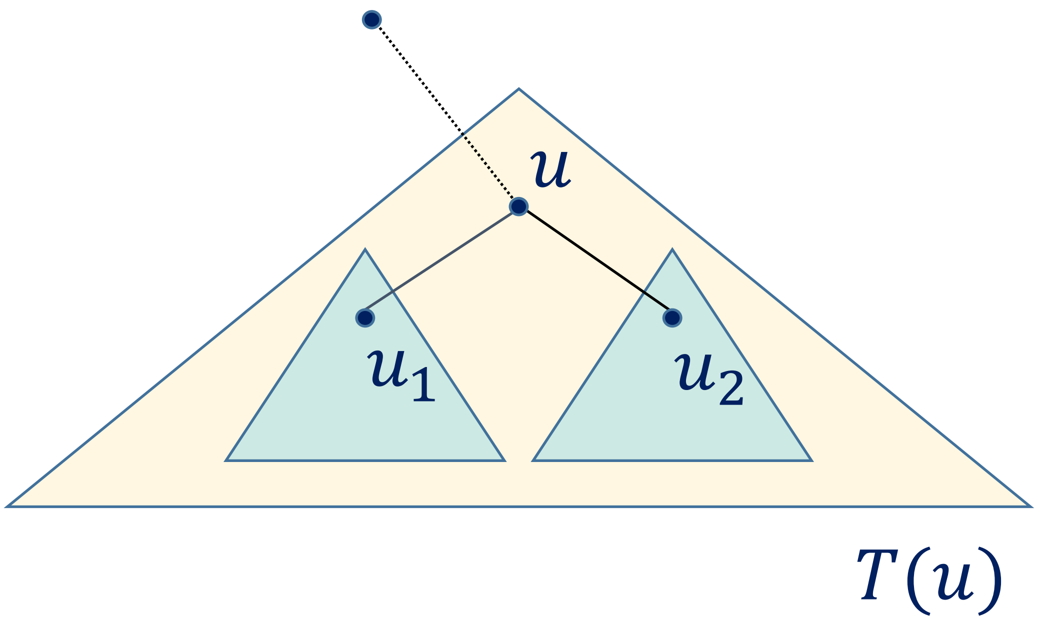
- into j clusters (subtrees)
- so that c is the center of the cluster containing u.



Dynamic programming algorithm

Fill out the DP table bottom-up.

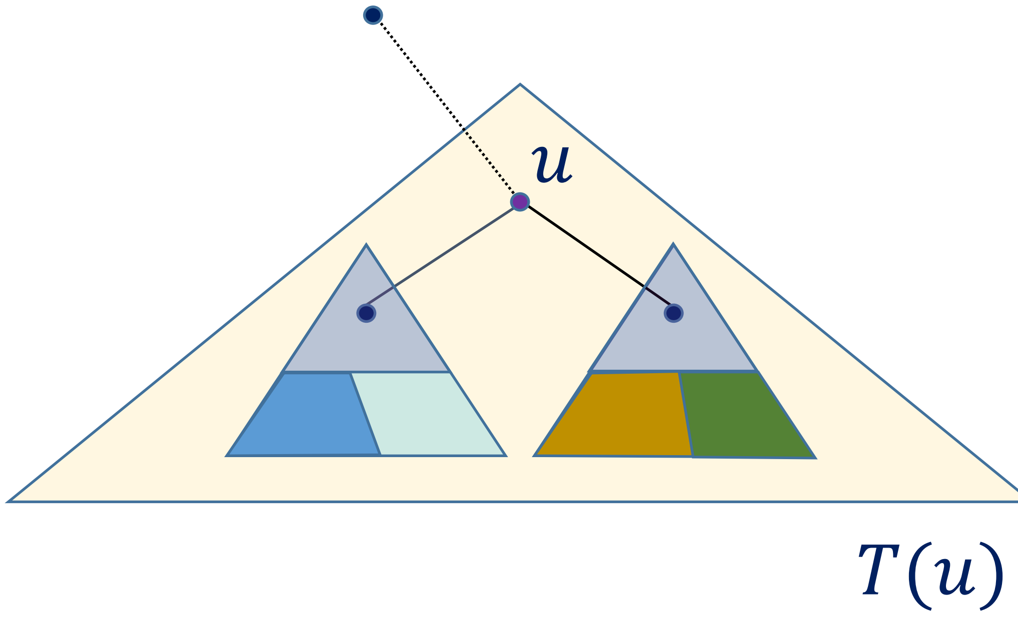
Example: k-median, u has 2 children u_1 and u_2.



Dynamic programming algorithm

Fill out the DP table bottom-up.

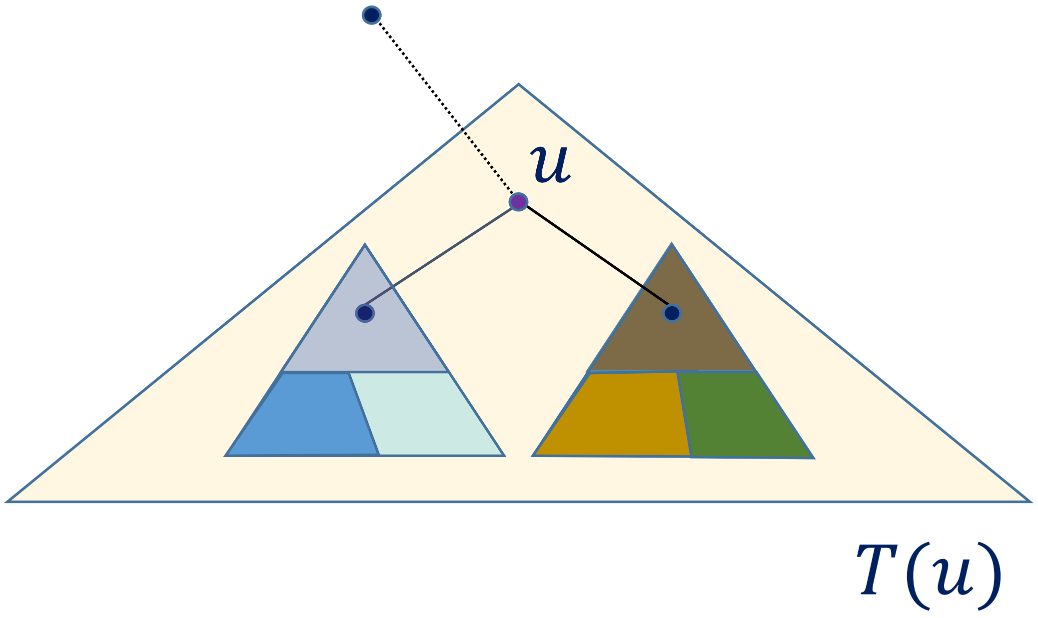
Example: k-median, u has 2 children u_1 and u_2.



Dynamic programming algorithm

Fill out the DP table bottom-up.

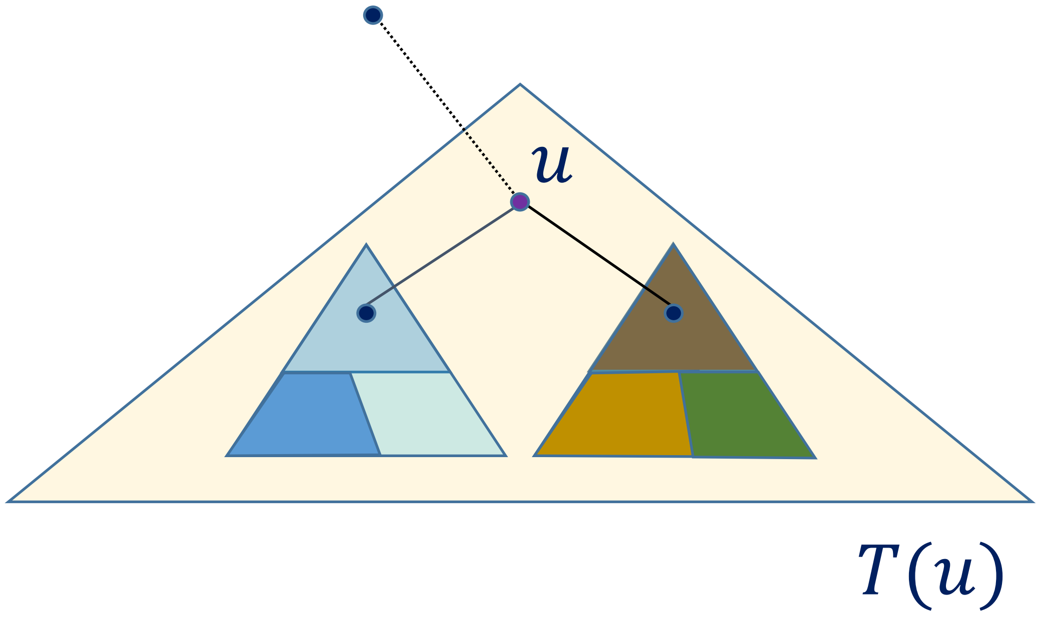
Example: k-median, u has 2 children u_1 and u_2.



Dynamic programming algorithm

Fill out the DP table bottom-up.

Example: k-median, u has 2 children u_1 and u_2.



Dynamic programming algorithm

u, u_1, u_2 lie in the same cluster

$$\mathrm{cost}_u(j, c) = d(u, c) + \mathrm{cost}_{u_1}(j_1, c) + \mathrm{cost}_{u_2}(j_2, c)$$

where $j_1 + j_2 = j + 1$

u, u_1, u_2 lie in different clusters

$$\mathrm{cost}_u(j, c) = d(u, c) + \mathrm{cost}_{u_1}(j_1, c_1) + \mathrm{cost}_{u_2}(j_2, c_2)$$

where $j_1 + j_2 = j - 1$, $c_1 \in T(u_1)$, $c_2 \in T(u_2)$

u, u_1 lie in the same clusters, u_2 in a different

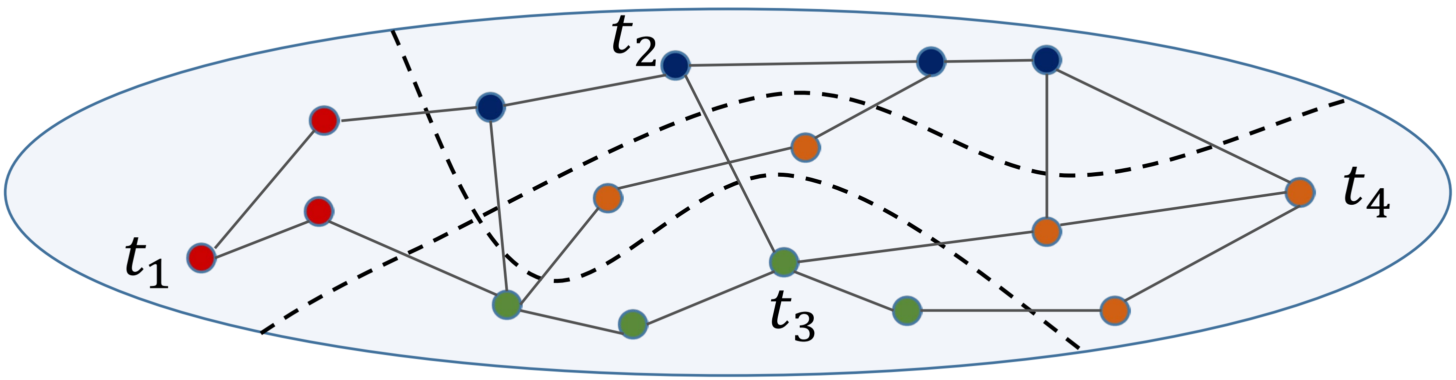
$$\mathrm{cost}_u(j, c) = d(u, c) + \mathrm{cost}_{u_1}(j_1, c) + \mathrm{cost}_{u_2}(j_1, c_2)$$

where $j_1 + j_2 = j$, $c_2 \in T(u_2)$

Multiway Cut

Given

- a graph $G = (V, E, w)$
- a set of terminals $t_1, \ldots, t_k$



Find a partition of V into sets $S_1, \ldots, S_k$ that minimizes the weight of cut edges s.t. $t_i \in S_i$.

Algorithms for Max Cut and Multiway Cut [MMV `13]

Write an SDP or LP relaxation for the problem.

Show that it is integral if the instance is γ-p.r.

```
solve the relaxation
if the SDP/LP solution is integral
    return the solution
else
    return that the instance is not γ-p.r.
```

The algorithm is *robust*: it *never* returns an incorrect answer.

Multiway Cut

Write the relaxation for Multiway Cut by Călinescu, Karloff, and Rabani [CKR `98]

To get an α-approximation, we would design a rounding scheme with

$$\Pr[(u, v) \text{ is cut}] \leq \alpha \, d(u, v)$$

Then

$$\mathbb{E}[\text{weight of cut edges}] \leq \alpha \sum_{(u,v)\in E} w_{uv} d(u, v)$$

Multiway Cut: complementary objective

If we want to maximize the weight of uncut edges, we would we would design a rounding scheme with

$$\Pr[(u, v) \text{ is not cut}] \geq \beta\,(1 - d(u, v))$$

Then

$$\mathbb{E}[\text{wt. of uncut edges}] \geq \beta \sum_{(u,v)\in E} w_{uv}(1 - d(u, v))$$

General approach to solving p.r. instances of graph partitioning

Write an LP or SDP relaxation for the problem.

Design a rounding procedure s.t.

$$\Pr[(u,v) \text{ is cut}] \leq \alpha\, d(u,v)$$ minimization

$$\Pr[(u,v) \text{ is not cut}] \geq \beta\big(1 - d(u,v)\big)$$

or

$$\Pr[(u,v) \text{ is cut}] \geq \beta\, d(u,v)$$ maximization

$$\Pr[(u,v) \text{ is not cut}] \leq \alpha\big(1 - d(u,v)\big)$$

! Then the relaxation for γ-p.r. is integral, when $\gamma \geq \alpha/\beta$

Solving Max Cut [MMV `13]

Use the Goemans–Williamson SDP relaxation with ℓ_2^2-triangle inequalities.

Design a rounding procedure with

$$\frac{\alpha}{\beta} = O\left(\sqrt{\log n} \log \log n\right),$$

which is a combination of two algorithms:

- the algorithm for Sparsest Cut with Nonuniform Demands by Arora, Lee, and Naor `08,
- the algorithm for Min Uncut by Agarwal, Charikar, Makarychev, M `05

Solving Multiway Cut [AMM `17]

Rounding procedures for Multiway Cut by

- Sharma and Vondrák `14
- Buchbinder, Schwartz, and Weizman `17

are highly non-trivial.

We need a rounding procedure only for LP solutions that are almost integral.

Design a simple rounding procedure with

$$\frac{\alpha}{\beta} = 2 - \frac{2}{k} .$$

Summary

- Algorithms for 2-perturbation-resilient instances of problems with a natural center-based objective: k-means, k-median, facility location.

- Robust algorithms for $O\left(\sqrt{\log n}\log\log n\right)$-p.r. instanced of Max Cut and $\left(2-\frac{2}{k}\right)$-p.r. instances of Multiway Cut.

- Negative results for p.r. instances of Max Cut, Multiway Cut, Max k-Cut, Multi Cut, Set Cover, Vertex Cover, Min 2-Horn Deletion.