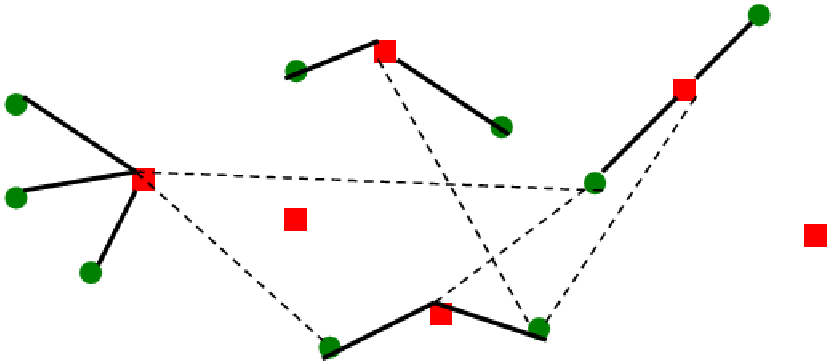
# The Euclidean k-Supplier Problem

Viswanath Nagarajan (IBM)
Baruch Schieber (IBM)
Hadas Shachnai (Technion)

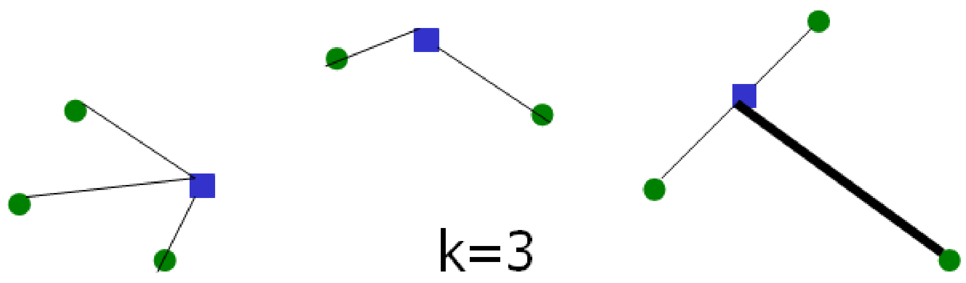
# Facility location problems

- Applications
  - plant location
  - placing servers in network
  - clustering…
- Clients and potential facilities
- Metric distances



# k-Center

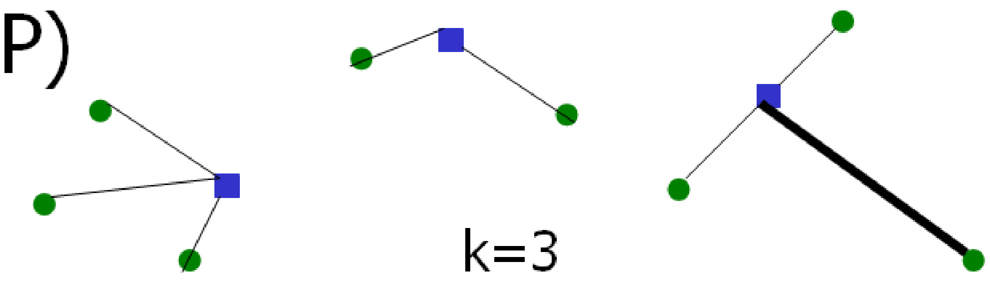
- Metric (V,d), $d : V \times V \to R_+$
- Open k centers S
  - Each vertex connects to nearest center
- Minimize $\max_{u \in V} d(u,S)$



# k-Center

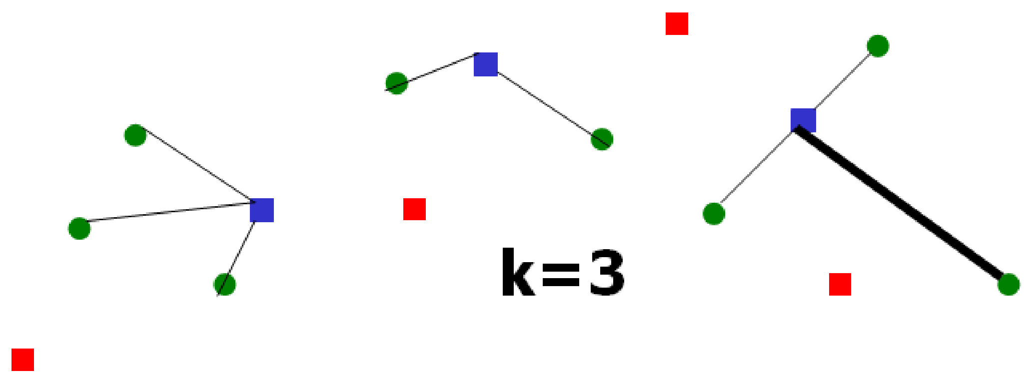
- Metric (V,d), $d : V \times V \to R_+$
- Open k centers S
  - Each vertex connects to nearest center
- Minimize $\max_{u \in V} d(u,S)$
- 2-approximation algorithms

  [Hochbaum, Shmoys '85] [Gonzalez '85]
  - Best possible (P≠NP)



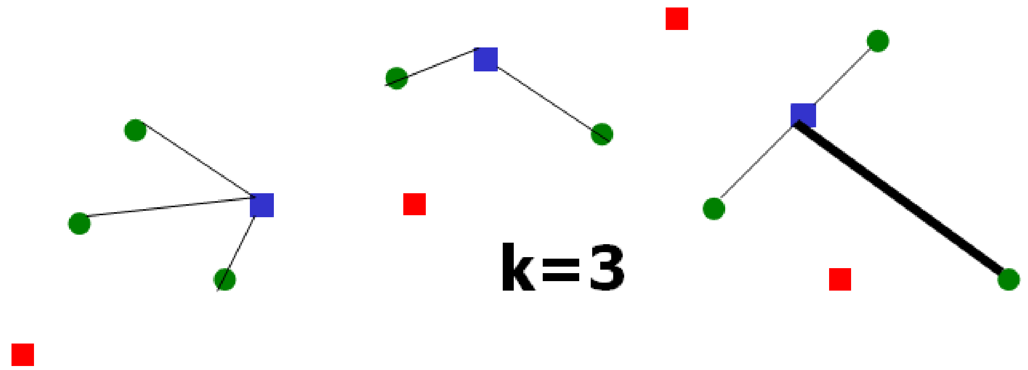
# k-Supplier

- Distinct sets of clients C, facilities F
  - metric ($C \cup F$, d)
- Open k facilities $S \subseteq F$ as centers
- Minimize $\max_{u \in C} d(u,S)$



# k-Supplier

- Distinct sets of clients C, facilities F
  - metric ($C \cup F$, d)
- Open k facilities $S \subseteq F$ as centers
- Minimize $\max_{u \in C} d(u,S)$
- 3-approximation algorithm [Hochbaum, Shmoys '86]
  - Best possible ($P \neq NP$)



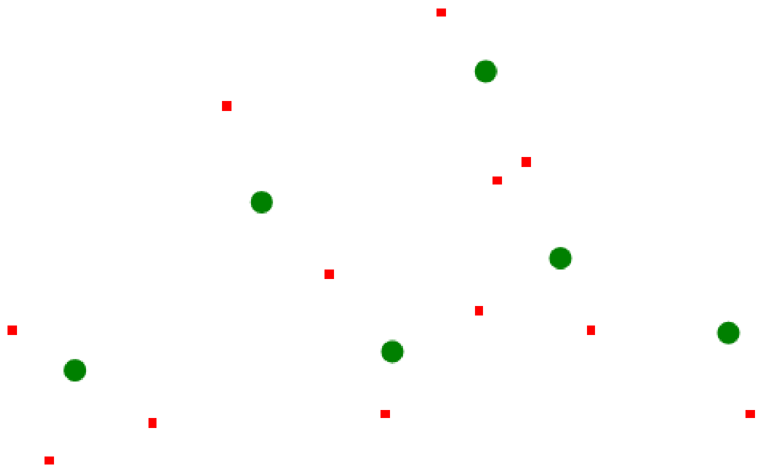
# Outline

- k-Supplier on general metrics
- Euclidean k-Supplier
- Fast approximation algorithm

# Algorithm for k-Supplier

[Hochbaum, Shmoys '86]

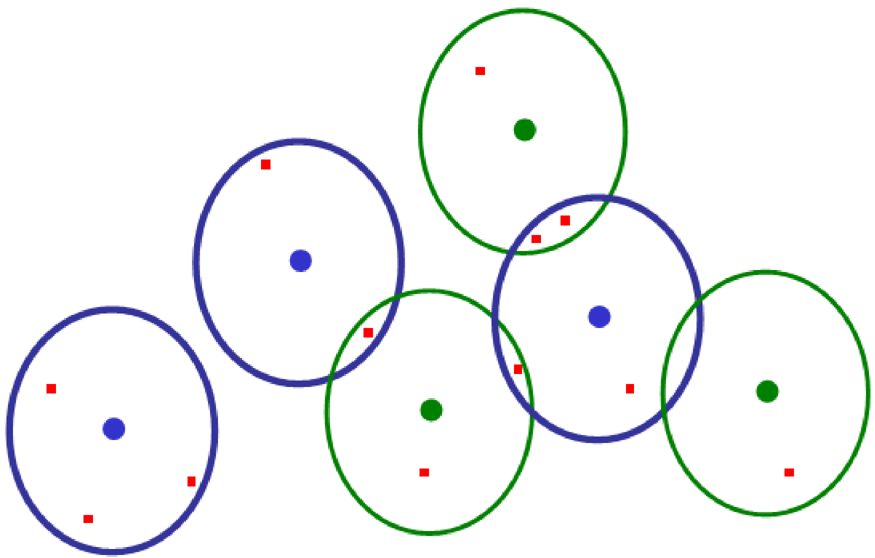
- Guess optimal value L



# Algorithm for k-Supplier

[Hochbaum, Shmoys '86]

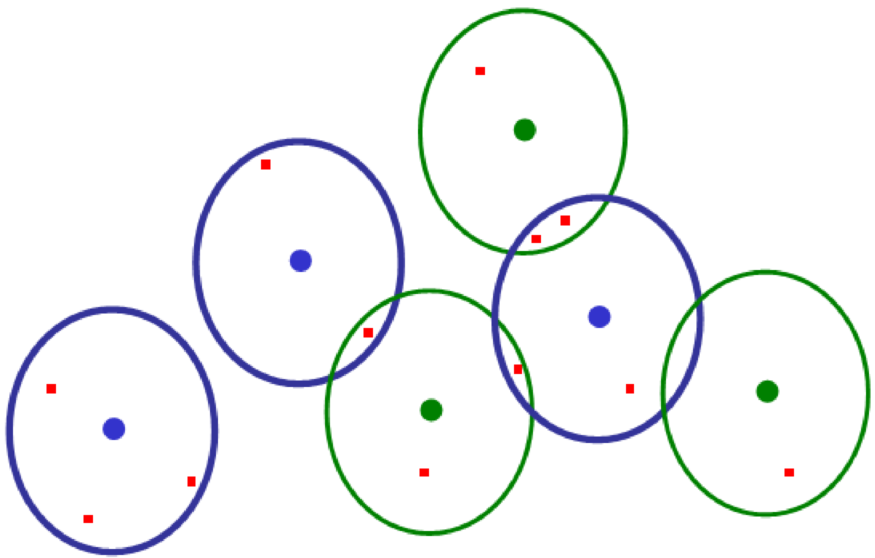
- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > 2L
  - Disjoint *balls* of radius L



# Algorithm for k-Supplier

[Hochbaum, Shmoys '86]

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > 2L
  - Disjoint *balls* of radius L

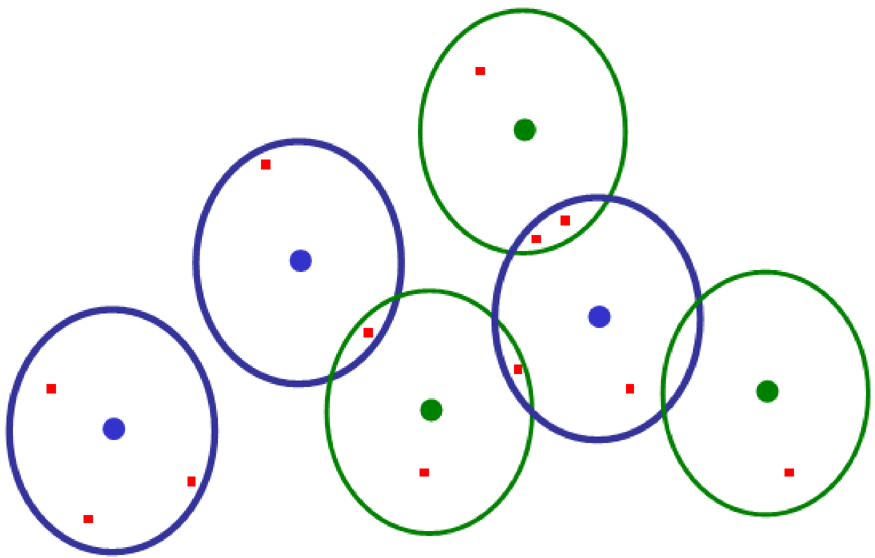


If |**S**| > k : declare **OPT** > L

# Algorithm for k-Supplier

[Hochbaum, Shmoys '86]

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > 2L
  - Disjoint *balls* of radius L



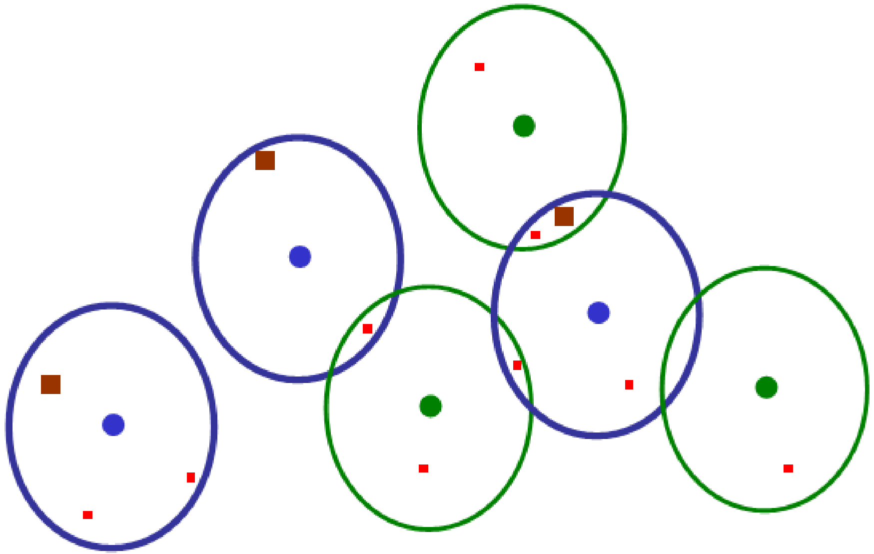
If |**S**| > k : declare **OPT** > L

If |**S**| $\leq$ k : declare **ALG** $\leq$ 3L

# Algorithm for k-Supplier

[Hochbaum, Shmoys '86]

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > 2L
  - Disjoint *balls* of radius L



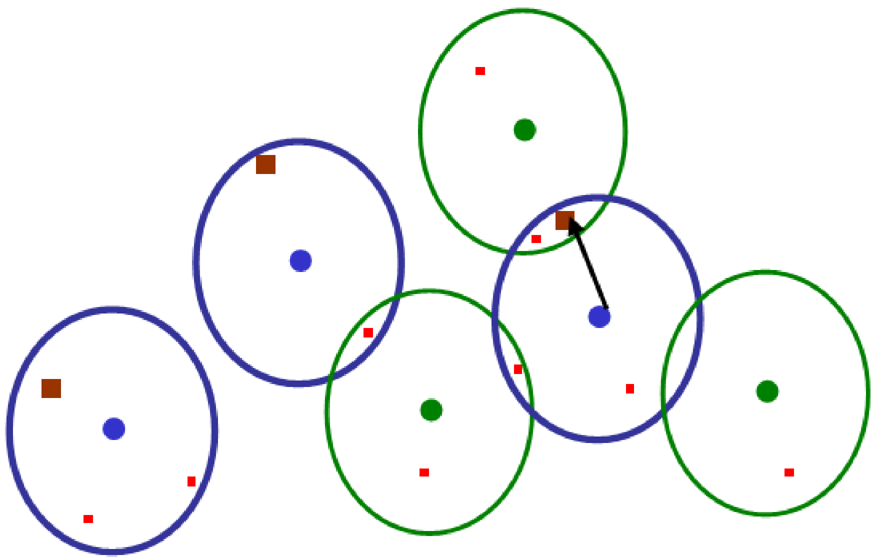
If |**S**| > k : declare **OPT** > L

If |**S**| $\leq$ k : declare **ALG** $\leq$ 3L
ALG = one facility in each **S** ball

# Algorithm for k-Supplier

[Hochbaum, Shmoys '86]

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > 2L
  - Disjoint *balls* of radius L



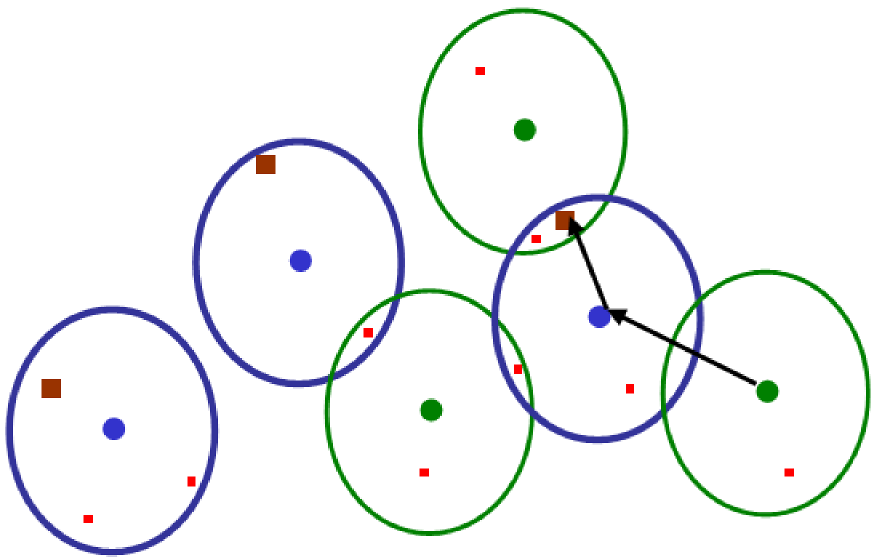
If |**S**| > k : declare **OPT** > L

If |**S**| ≤ k : declare **ALG** ≤ 3L
ALG = one facility in each **S** ball

# Algorithm for k-Supplier

[Hochbaum, Shmoys '86]

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > 2L
  - Disjoint *balls* of radius L



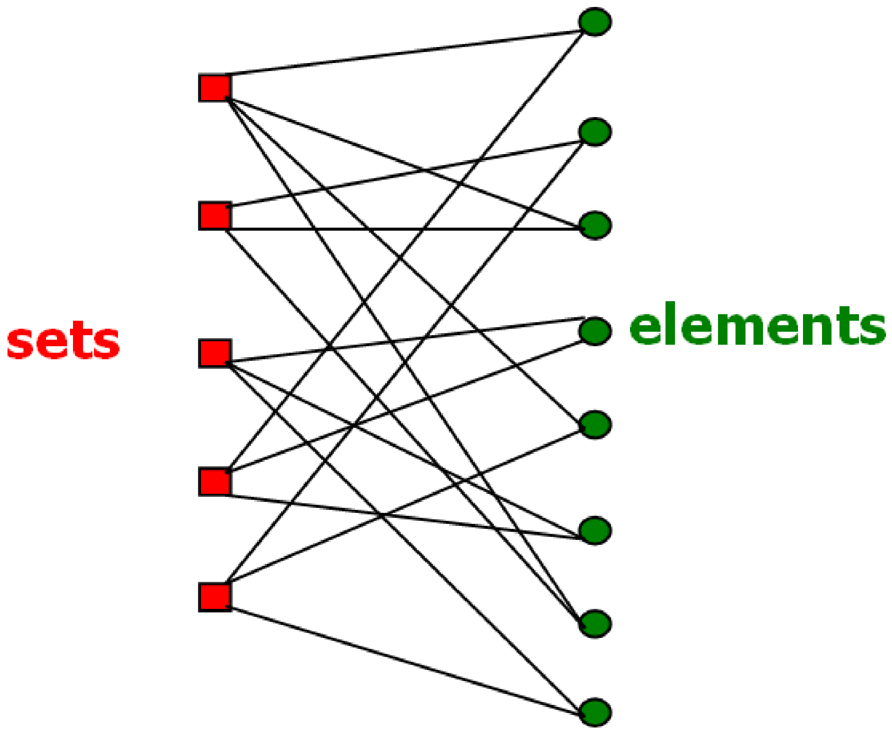
If |**S**| > k : declare **OPT** > L

If |**S**| $\leq$ k : declare **ALG** $\leq$ 3L
ALG = one facility in each **S** ball

# Hardness for k-Supplier

[Hochbaum, Shmoys '86]

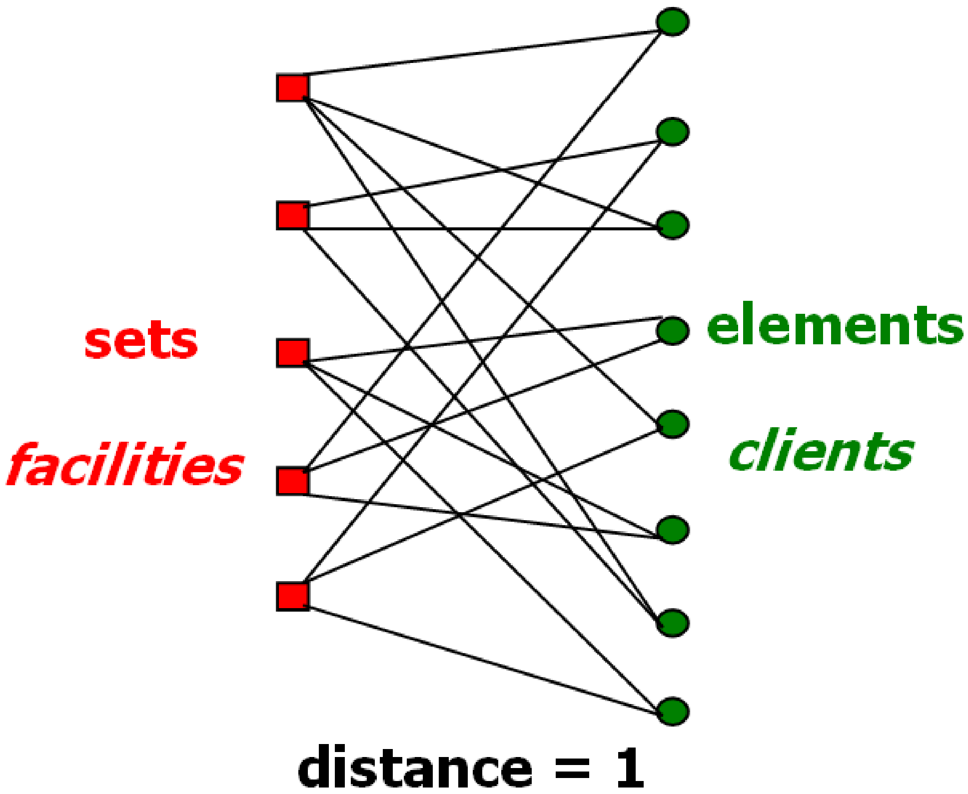
- From Set Cover



# Hardness for k-Supplier

[Hochbaum, Shmoys '86]

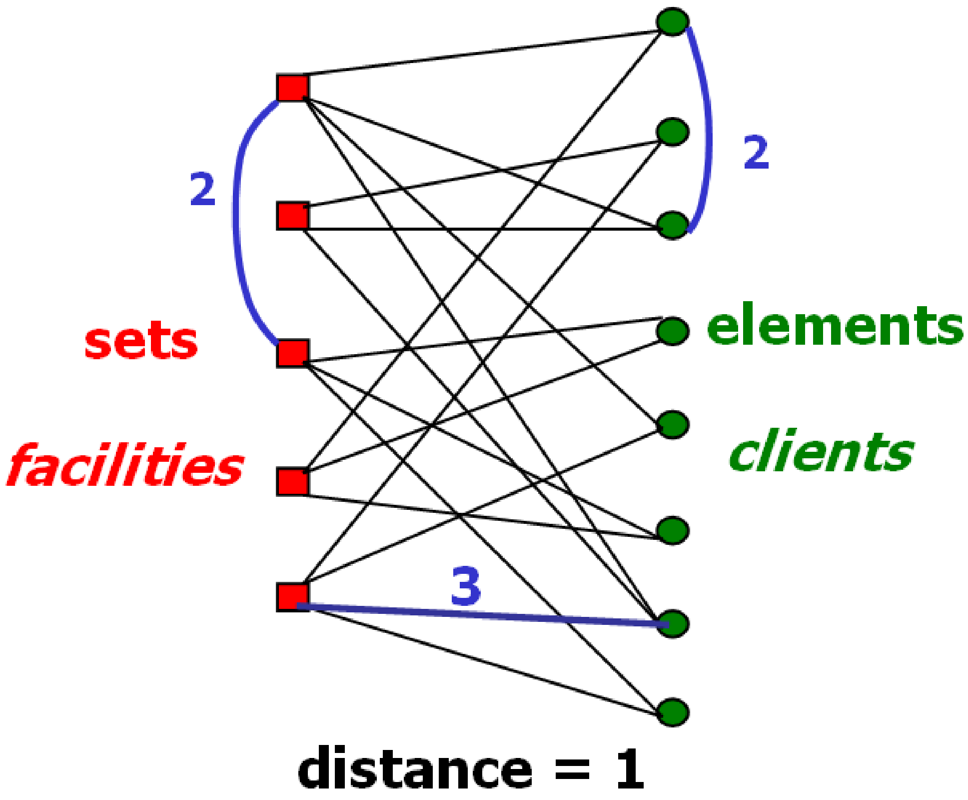
- From Set Cover



# Hardness for k-Supplier

- From Set Cover

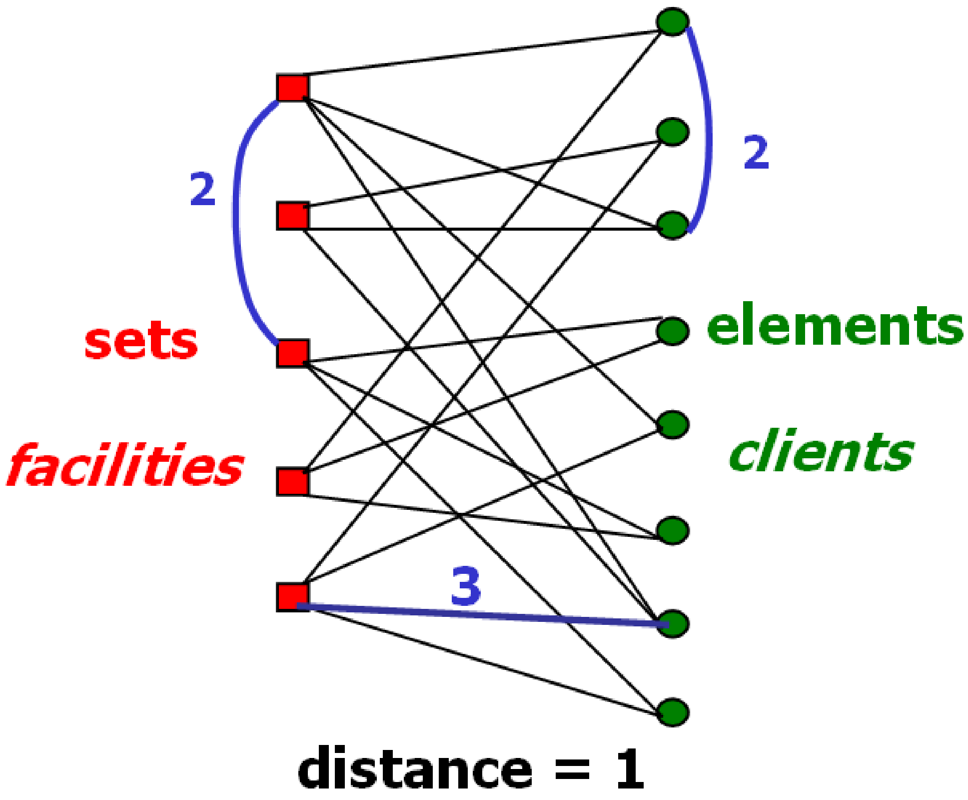
[Hochbaum, Shmoys '86]



# Hardness for k-Supplier

- From Set Cover

[Hochbaum, Shmoys '86]

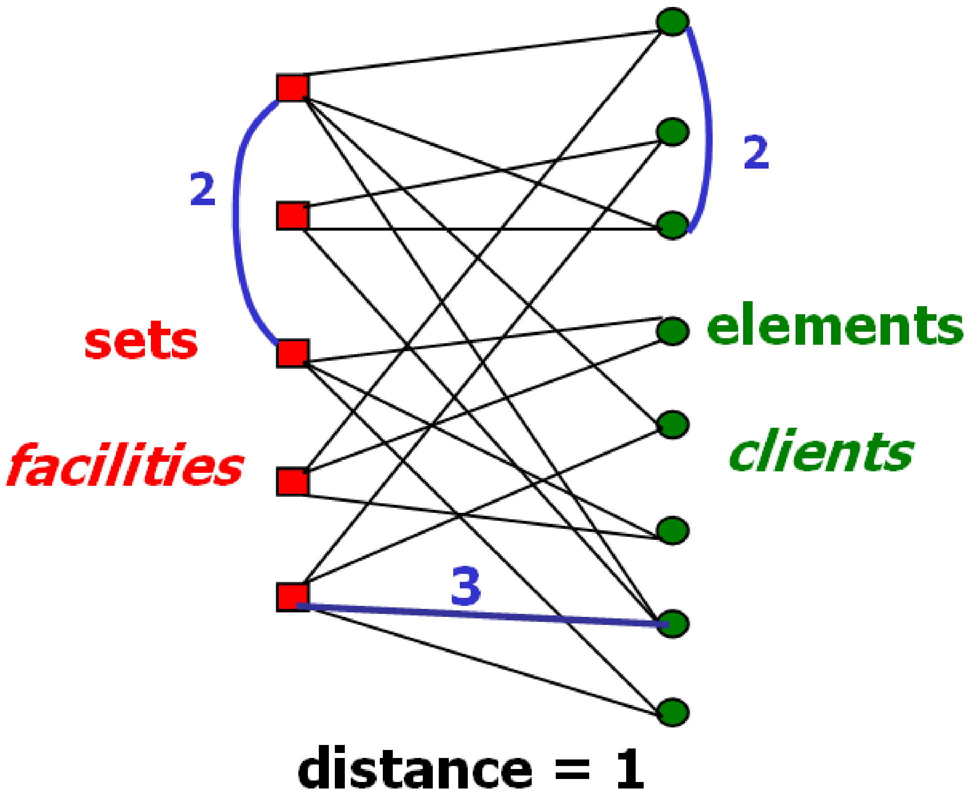



$$\mathrm{OPT}(\mathrm{SC}) \leq k \Rightarrow \mathrm{OPT}(\textit{k-Supp}) = 1$$

# Hardness for k-Supplier

- From Set Cover

[Hochbaum, Shmoys '86]




$\mathrm{OPT(SC)} \leq k \Rightarrow \mathrm{OPT}(\textit{k-Supp}) = 1$

$\mathrm{OPT(SC)} > k \Rightarrow \mathrm{OPT}(\textit{k-Supp}) = 3$

# Euclidean k-Supplier

- Points in $R^d$

- Euclidean distance
  - $d(u,v) = \sqrt{(u_1-v_1)^2 + (u_2-v_2)^2 + \ldots + (u_d-v_d)^2}$

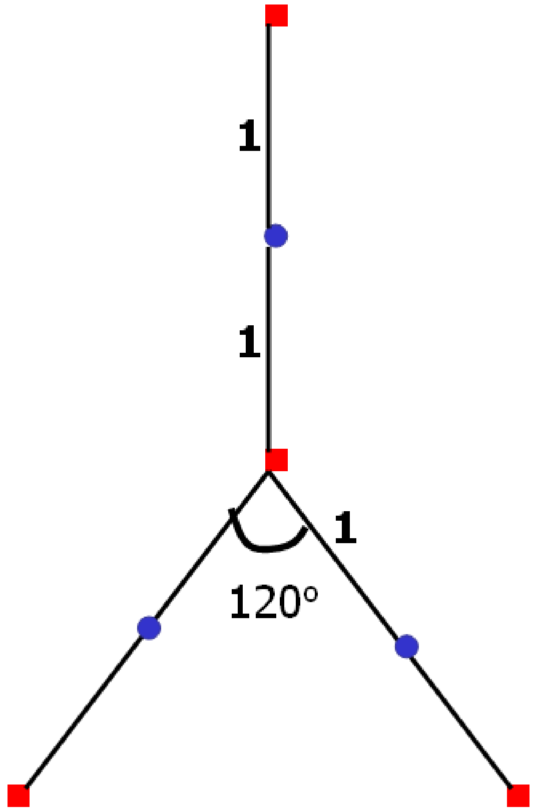
- Natural metric in many applications

*Can we do any better?*

# Euclidean k-Supplier Hardness

**Thm**: NP-hard to approximate better than $\sqrt{7}$

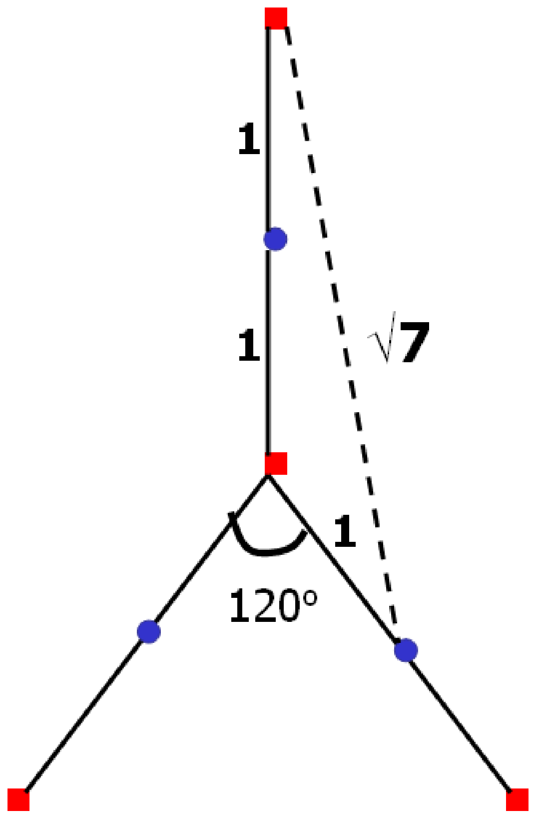
- From degree 3 planar vertex cover [Feder, Greene '88]



# Euclidean k-Supplier Hardness

**Thm**: NP-hard to approximate better than √7

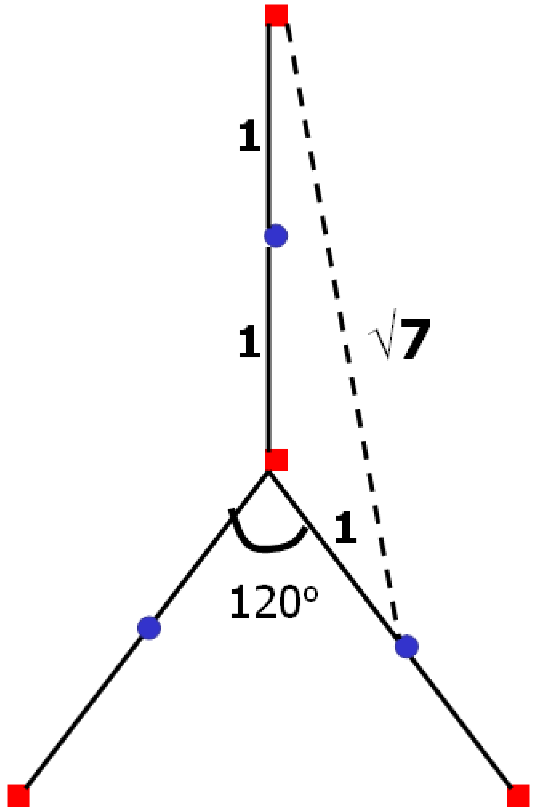
- From degree 3 planar vertex cover [Feder, Greene '88]



# Euclidean k-Supplier Hardness

**Thm**: NP-hard to approximate better than √7

- From degree 3 planar vertex cover [Feder, Greene '88]




OPT(VC) $\leq$ k $\Rightarrow$ OPT(k-Supp) = 1

OPT(VC) **>** k $\Rightarrow$ OPT(k-Supp) $\geq$ √7

# Results

## Euclidean k-Supplier

- $\sqrt{3}+1 < 2.74$ approximation algorithm
  - Any number of dimensions
  - Running time ~$O(d \cdot n^2)$
  - $\sqrt{7} > 2.64$ hardness [Feder, Greene '88]

# Results

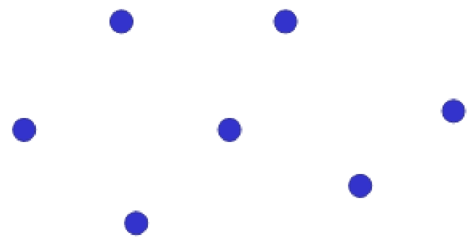
Euclidean k-Supplier

- $\sqrt{3}+1 < 2.74$ approximation algorithm
  - Any number of dimensions
  - Running time $\sim O(d \cdot n^2)$
  - $\sqrt{7} > 2.64$ hardness [Feder, Greene '88]

- Fast 2.965 approximation algorithm
  - Time $O(n \cdot \log^2 n)$, $d=O(1)$ dimensions
  - Previously 3-approx in $O(n \cdot \log k)$

# Algorithm

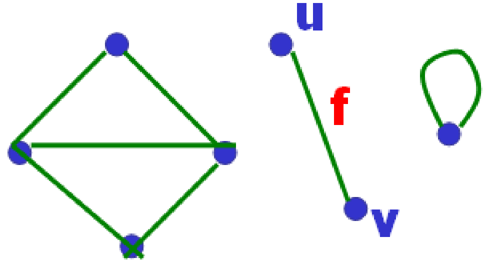
- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > $\sqrt{3}$ L



# Algorithm

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > $\sqrt{3}$ L
- Define graph **G** on vertices **S**
  - edge (u,v) *iff* $\exists$ facility f : d(u,f) and d(v,f) $\leq$ L

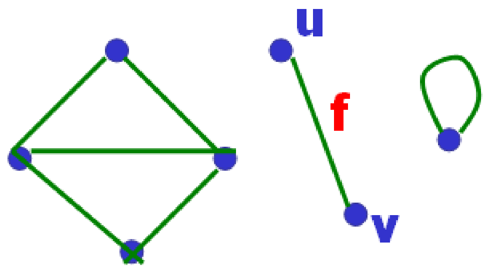
**graph G**



# Algorithm

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance $> \sqrt{3}\,L$
- Define graph **G** on vertices **S**
  - edge (u,v) *iff* $\exists$ facility f : d(u,f) and d(v,f) $\leq$ L
- Solve *min edge cover* on **G**

**graph G**



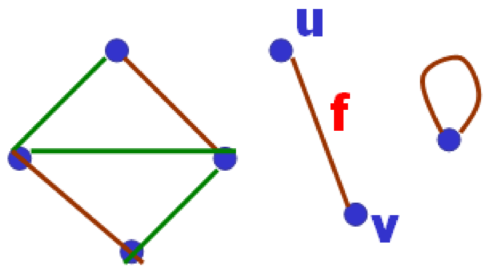

**Edge cover**: min number of edges to cover all vertices

Poly-time algorithm [Edmonds '65]...

# Algorithm

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > $\sqrt{3}$ L
- Define graph **G** on vertices **S**
  - edge (u,v) *iff* $\exists$ facility f : d(u,f) and d(v,f) $\leq$ L
- Solve *min edge cover* on **G**

**graph G**



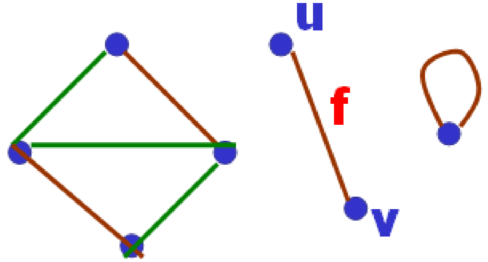

**Edge cover**: min number of edges to cover all vertices

Poly-time algorithm [Edmonds '65]...

# Algorithm

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > $\sqrt{3}$ L
- Define graph **G** on vertices **S**
  - edge (u,v) *iff* $\exists$ facility f : d(u,f) and d(v,f) $\leq$ L
- Solve *min edge cover* on **G**
  - EC* > k $\Rightarrow$ OPT > L
  - EC* $\leq$ k $\Rightarrow$ ALG $\leq$ ($\sqrt{3}$+1) $\cdot$ L

**graph G**




**Edge cover:** min number of edges to cover all vertices

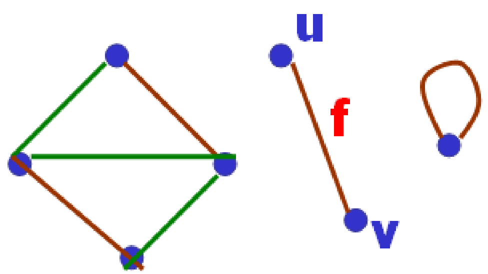
Poly-time algorithm [Edmonds '65]...

# Algorithm

- Guess optimal value L
- *Maximal* clients **S** of pairwise distance > √3 L
- Define graph **G** on vertices **S**
  - edge (u,v) *iff* ∃ facility f : d(u,f) and d(v,f) ≤ L
- Solve *min edge cover* on **G**

***To show***
  - EC* > k ⇒ OPT > L
  - EC* ≤ k ⇒ ALG ≤ (√3+1) · L

**graph G**



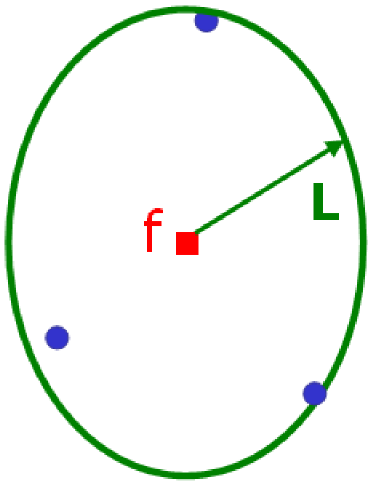

**Edge cover:** min number of edges to cover all vertices
Poly-time algorithm [Edmonds '65]...

# Analysis

- Main property: any facility f can "cover" *at most two* clients in S

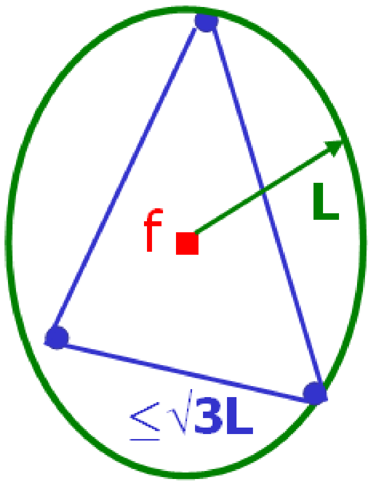
# Analysis

- Main property: any facility f can "cover" *at most two* clients in S



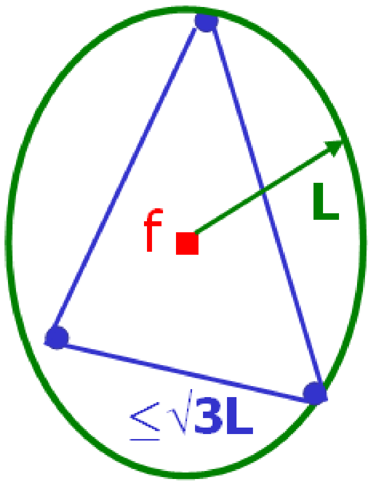
# Analysis

- Main property: any facility f can "cover" *at most two* clients in S



# Analysis

- Main property: any facility f can "cover" *at most two* clients in S

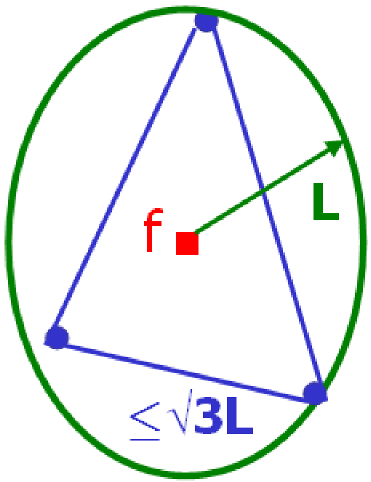



Contradicts choice of "net" **S** !

# Analysis

- Main property: any facility f can "cover" *at most two* clients in S

- k-Supplier[S] of value L ≡ edge cover on G




Contradicts choice of "net" **S** !

# Running time

- Construct graph G in $O(dn^2)$
  - naïve method
- Solve edge cover in $O(n^{1.5})$
  - $O(E\sqrt{V})$ algorithm [Micali, Vazirani '80]
- Overall: $O(dn^2 \cdot \log n)$

# Running time

- Construct graph G in $O(dn^2)$
  - naïve method
- Solve edge cover in $O(n^{1.5})$
  - $O(E\sqrt{V})$ algorithm [Micali, Vazirani '80]
- Overall: $O(dn^2 \cdot \log n)$

- Improved runtime for $d=O(1)$

# Faster implementation

- Approximate nearest neighbor
  - [Arya, Mount, Netanyahu, Silverman, Wu '98]
  - 1+$\epsilon$ approx. nearest neighbors
  - O(log n) time per query: NN, add, delete
  - O(1) dimensions

# Faster implementation

- Approximate nearest neighbor
  - [Arya, Mount, Netanyahu, Silverman, Wu '98]
  - $1+\epsilon$ approx. nearest neighbors
  - O(log n) time per query: NN, add, delete
  - O(1) dimensions

$\Rightarrow$ Construct G in $O(n \cdot \log n)$

# Faster implementation

- Approximate nearest neighbor
  - [Arya, Mount, Netanyahu, Silverman, Wu '98]
  - $1+\epsilon$ approx. nearest neighbors
  - O(log n) time per query: NN, add, delete
  - O(1) dimensions

  $\Rightarrow$ Construct G in $O(n \cdot \log n)$
- Edge cover in $O(n^{1.5})$

$\sqrt{3}+1+\epsilon$ approx. in $\sim O(n^{1.5})$ time

# Faster implementation

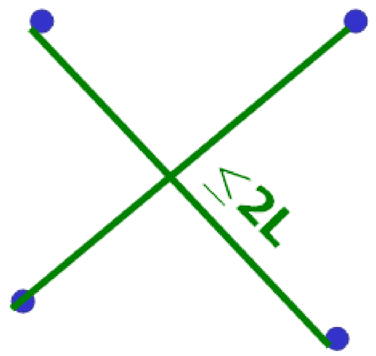
- Approximate nearest neighbor
  - [Arya, Mount, Netanyahu, Silverman, Wu '98]
  - $1+\epsilon$ approx. nearest neighbors
  - O(log n) time per query: NN, add, delete
  - O(1) dimensions

  $\Rightarrow$ Construct G in $O(n \cdot \log n)$
- Edge cover in $O(n^{1.5})$
  - Use additional structure in **G** ?

$\sqrt{3}+1+\epsilon$ approx.
in $\sim O(n^{1.5})$ time

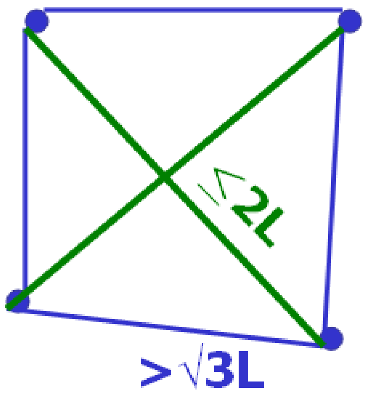
# Faster implementation d=2

- Here G is planar



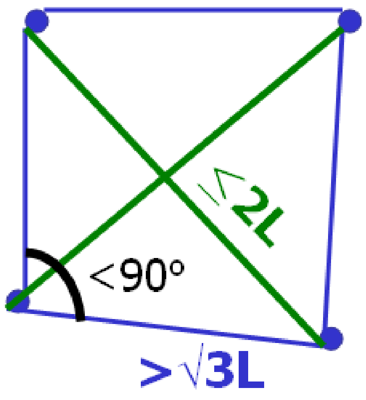
# Faster implementation d=2

- Here G is planar



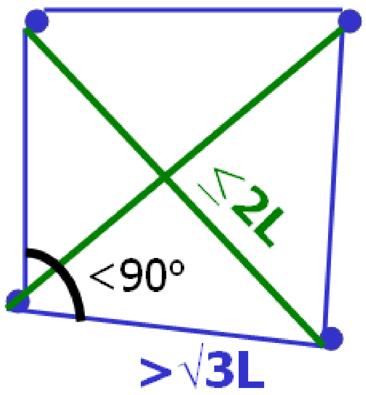
# Faster implementation d=2

- Here G is planar



# Faster implementation d=2

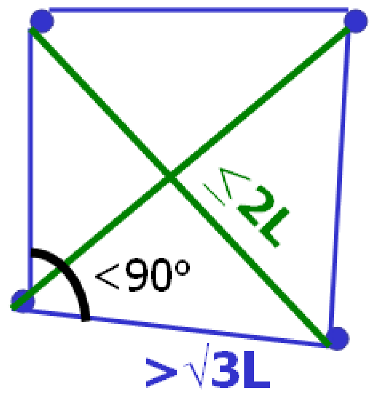
- Here G is planar




Not possible in 2d !

# Faster implementation d=2

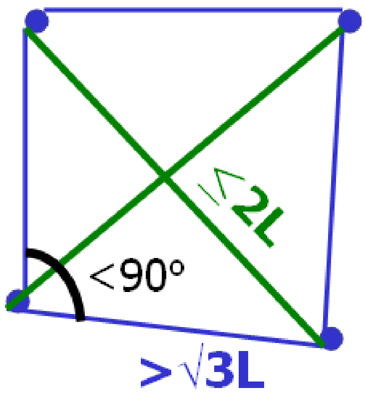
- Here G is planar
- Use $O(n^{\omega/2})$ algorithm [Mucha, Sankowski '06]
  - $\omega < 2.38$ is matrix multiplication exponent




Not possible in 2d !

# Faster implementation d=2

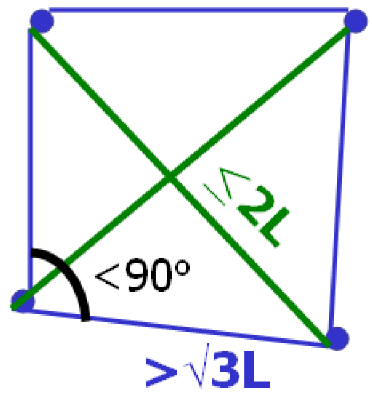
- Here G is planar
- Use $O(n^{\omega/2})$ algorithm [Mucha, Sankowski '06]
  - $\omega < 2.38$ is matrix multiplication exponent
- Don't know further useful structure of G




Not possible in 2d !

# Faster implementation d=2

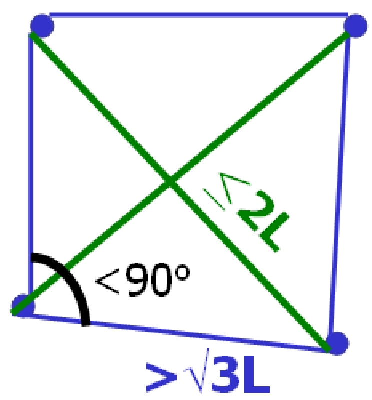
- Here G is planar
- Use $O(n^{\omega/2})$ algorithm [Mucha, Sankowski '06]
  - $\omega < 2.38$ is matrix multiplication exponent
- Don't know further useful structure of G
  - For d=2, G can be *any* degree 3 planar




Not possible in 2d !

# Faster implementation d=2

- Here G is planar
- Use $O(n^{\omega/2})$ algorithm [Mucha, Sankowski '06]
  - $\omega < 2.38$ is matrix multiplication exponent
- Don't know further useful structure of G
  - For d=2, G can be *any* degree 3 planar
  - For d$\geq$3, G does not exclude any minor




Not possible in 2d !

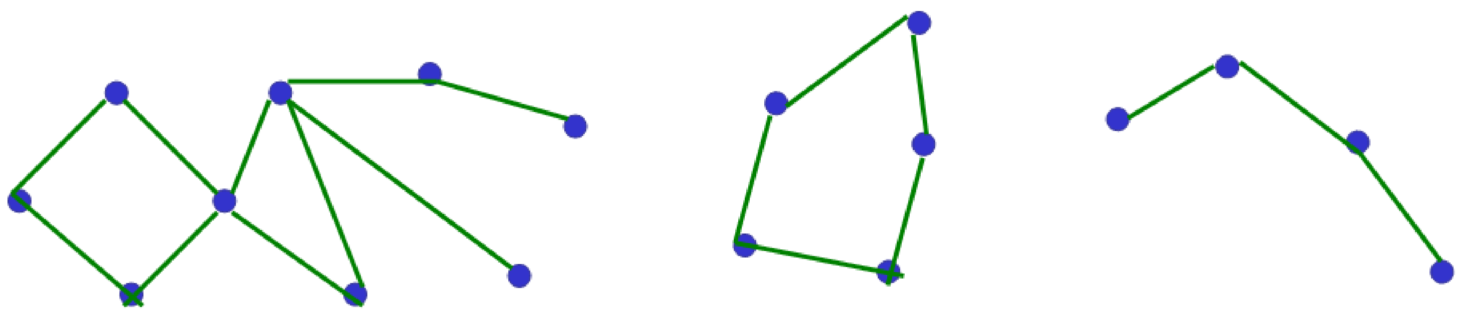
# Nearly linear time algorithm

# Near-linear time

- Idea: reduce to edge cover on special graphs

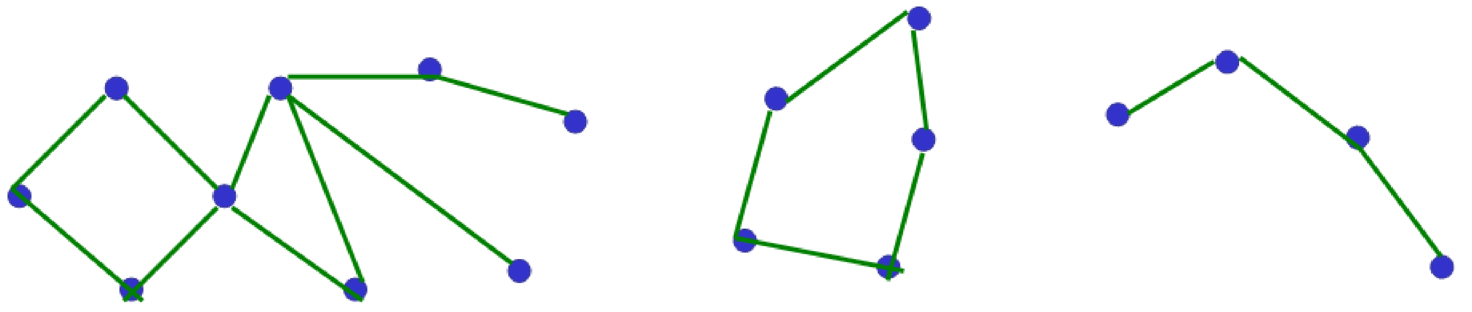
# Near-linear time

- Idea: reduce to edge cover on special graphs
  - *Cactus* (linear cycles)
  - Solvable in linear time



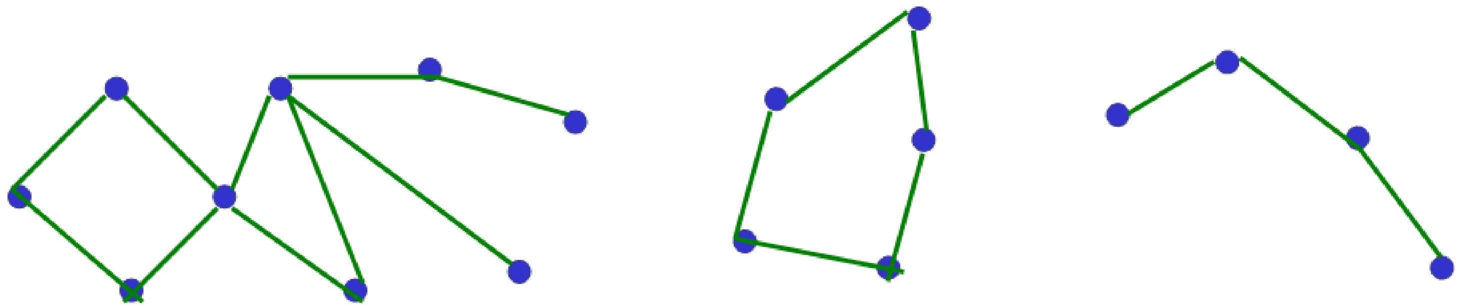
# Near-linear time

- Idea: reduce to edge cover on special graphs
  - *Cactus* (linear cycles)
  - Solvable in linear time
- Constructing G more complex ☹
  - Also worse approx. ratio 2.965 ☹



# Near-linear time

- Idea: reduce to edge cover on special graphs
  - *Cactus* (linear cycles)
  - Solvable in linear time
- Constructing G more complex ☹
  - Also worse approx. ratio 2.965 ☹
  - But time ~O(n) using ANN ☺



# Constructing G

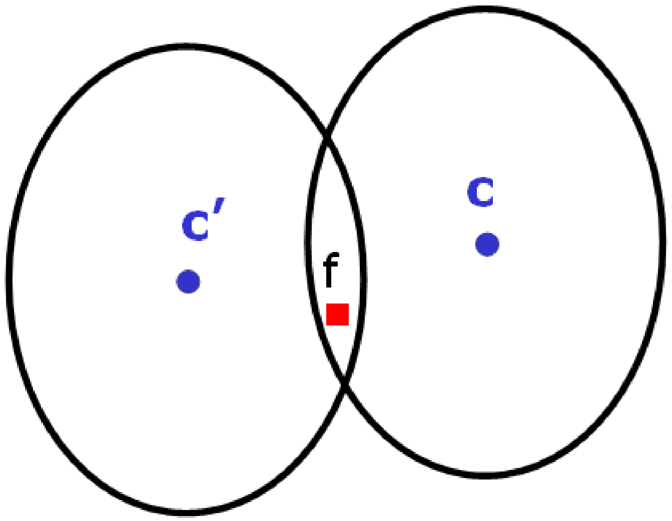
- Graph G has
  - vertices $S \subseteq$ clients
  - edges $E \subseteq$ facilities
- Use larger separation in S + more geometry

scale L = 1

# Constructing G

- Graph G has
  - vertices $S \subseteq$ clients
  - edges $E \subseteq$ facilities
- Use larger separation in S + more geometry

scale L = 1



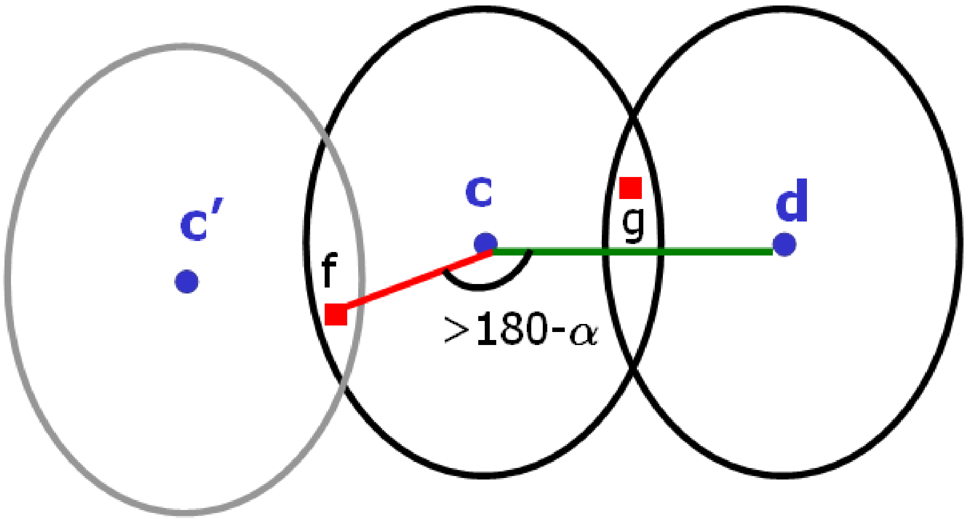

c′ and c ***fringe*** intersect

# Constructing G

- Graph G has
  - vertices S $\subseteq$ clients
  - edges E $\subseteq$ facilities

scale L = 1

- Use larger separation in S + more geometry

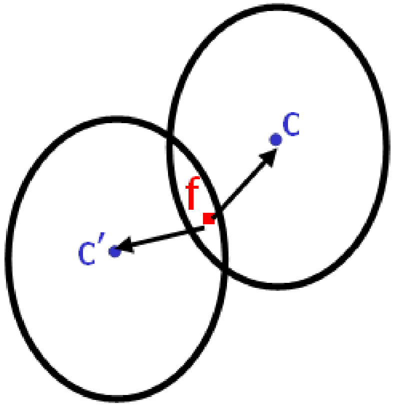



c′ and c ***fringe*** intersect

d ***antipode*** intersects <f,c>
(i) c & d fringe intersect
(ii) d is "too far" from f

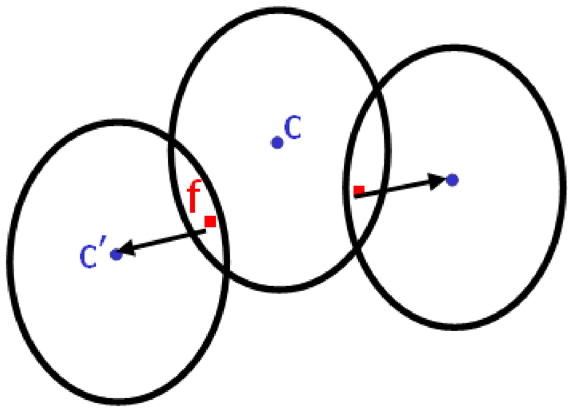
# Constructing G

- Find next antipode intersection using ANN
  - Maintain pair of “end points”



# Constructing G

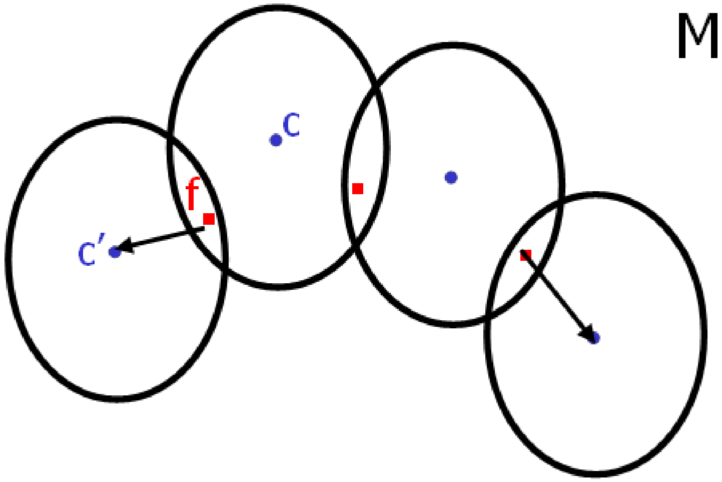
Find next antipode intersection using ANN




Maintain pair of "end points"

# Constructing G

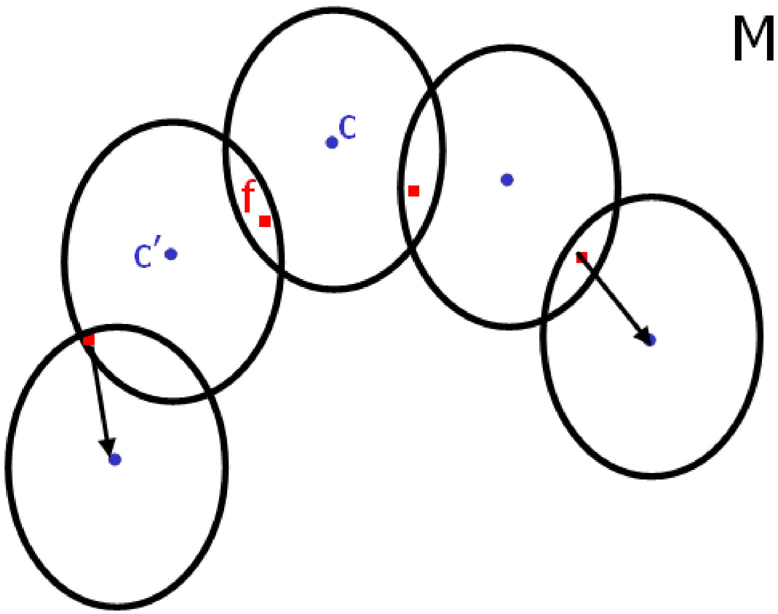
- Find next antipode intersection using ANN
  - Maintain pair of "end points"



# Constructing G

Find next antipode intersection using ANN

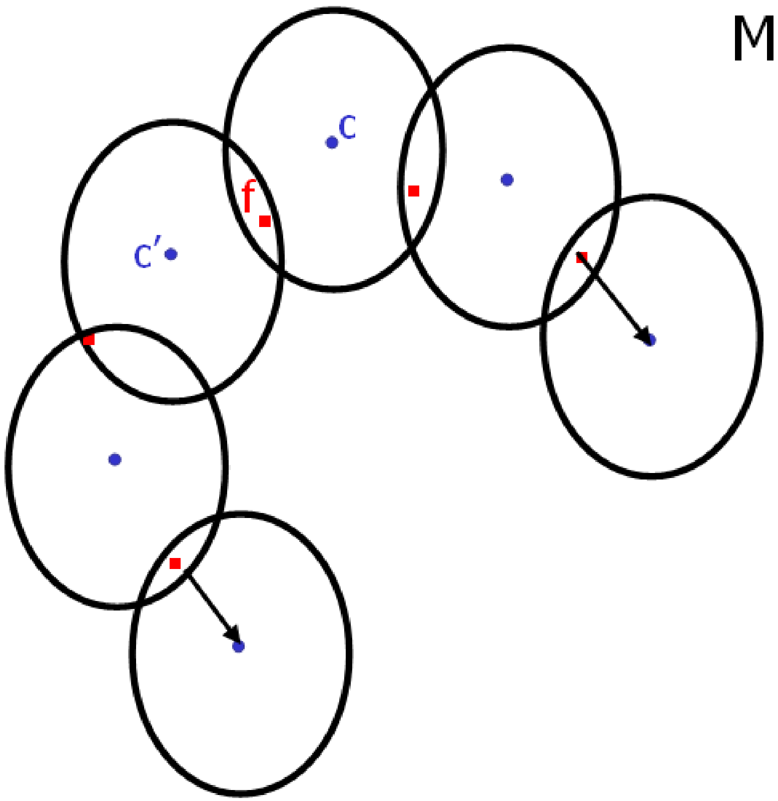
Maintain pair of "end points"



# Constructing G

Find next antipode intersection using ANN

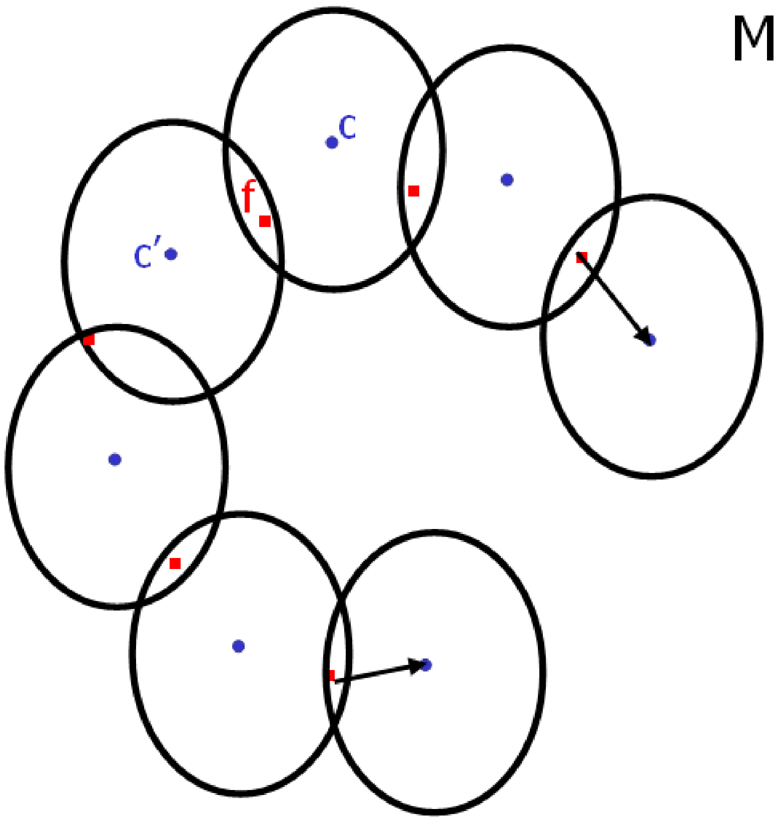
Maintain pair of "end points"



# Constructing G

Find next antipode intersection using ANN

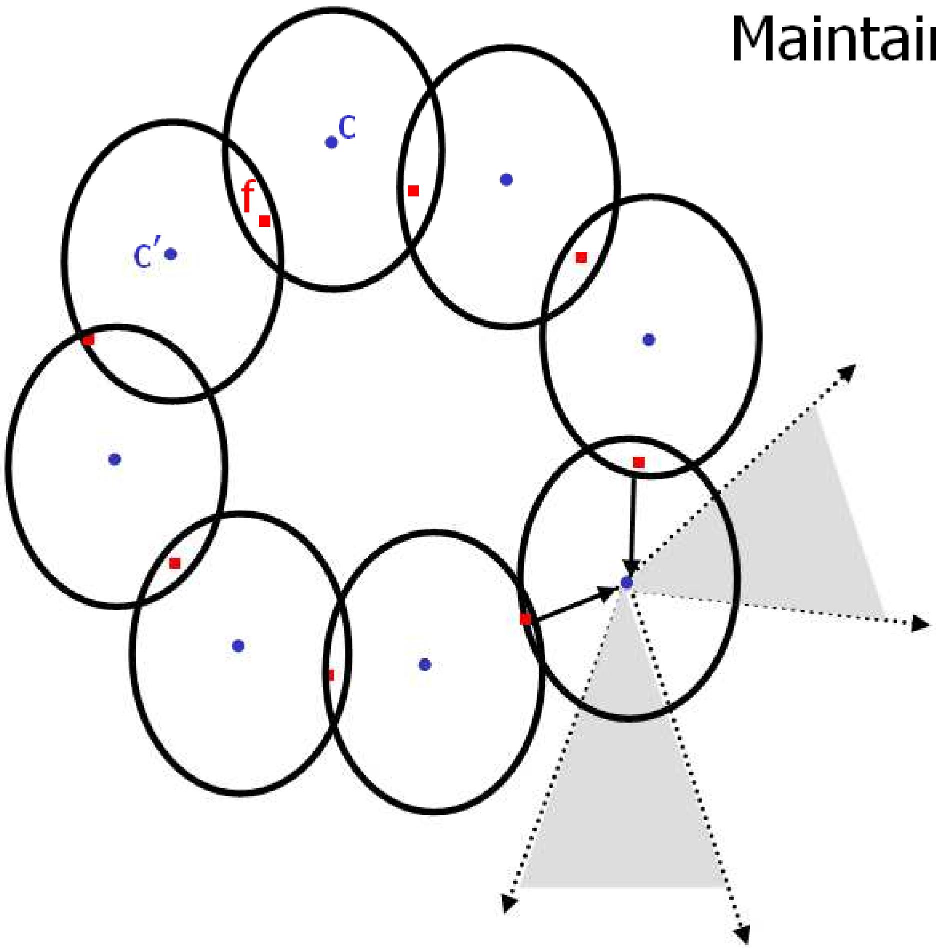
Maintain pair of "end points"



# Constructing G

Find next antipode intersection using ANN

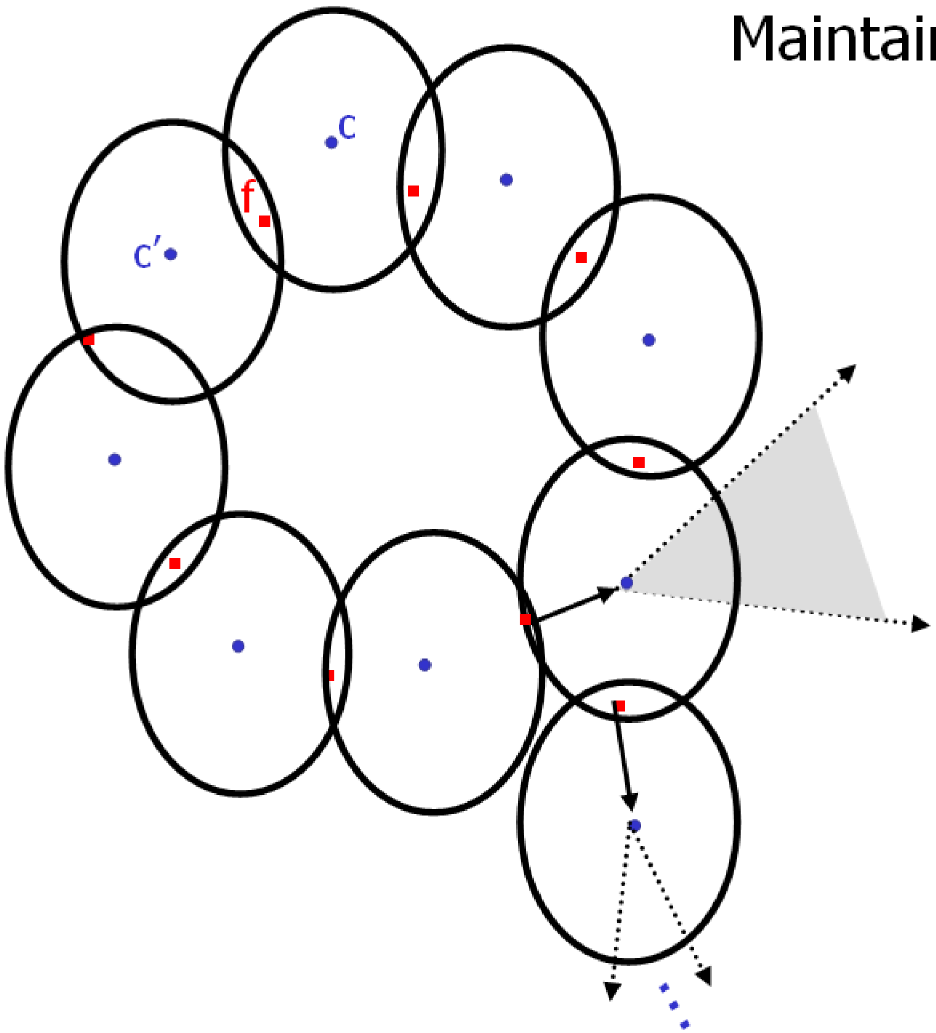
Maintain pair of “end points”



# Constructing G

Find next antipode intersection using ANN

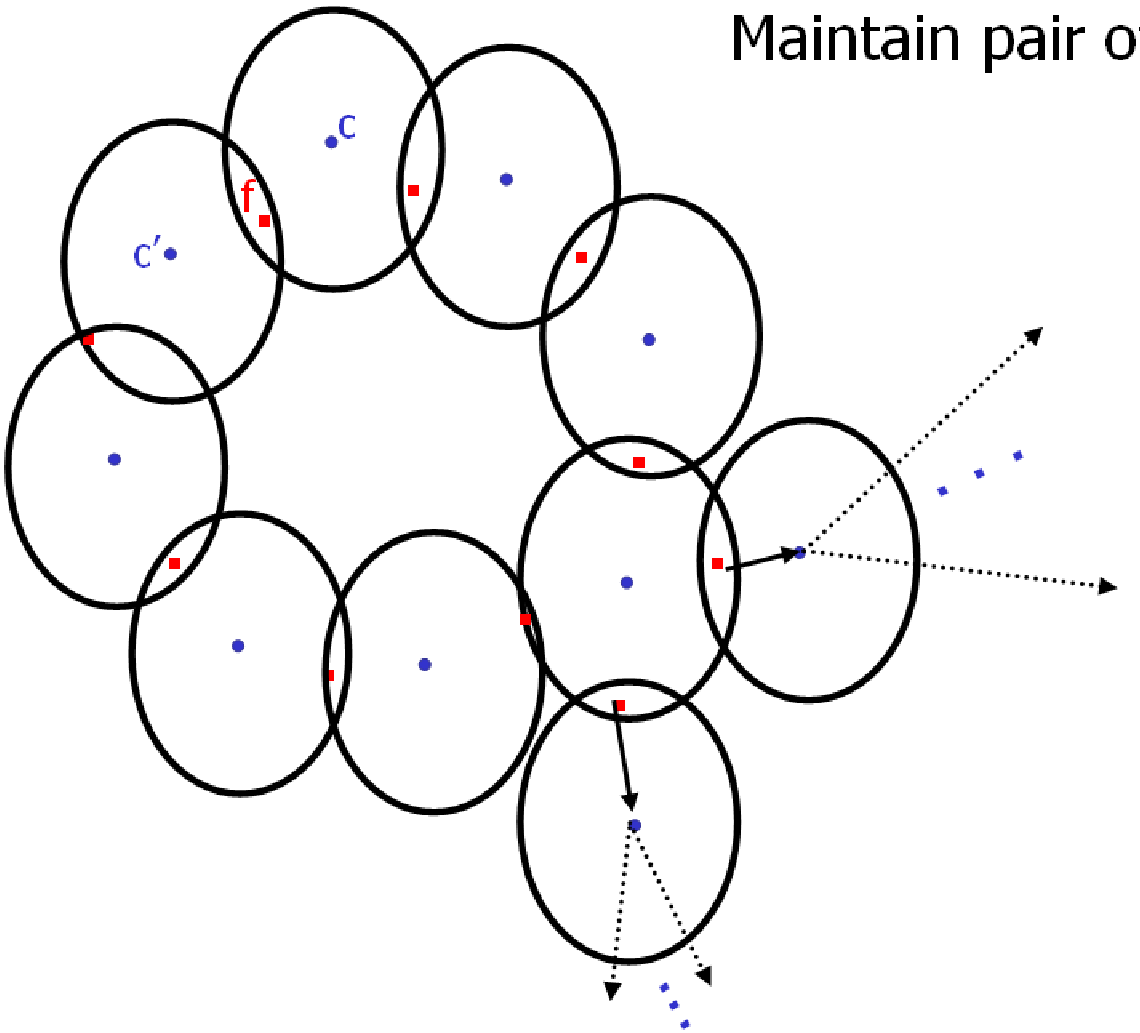
Maintain pair of "end points"



# Constructing G

Find next antipode intersection using ANN

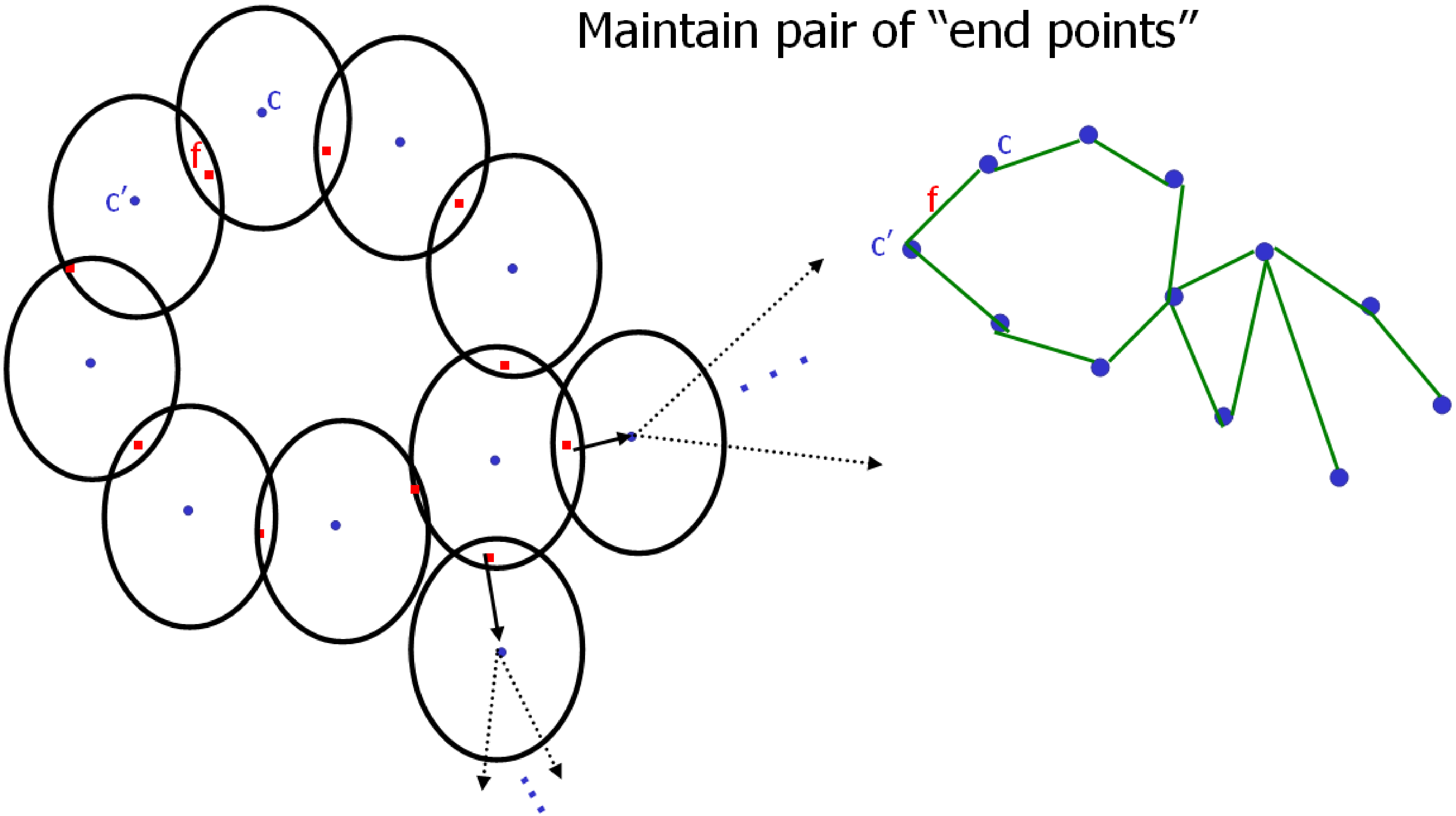
Maintain pair of "end points"



# Constructing G

Find next antipode intersection using ANN

Maintain pair of "end points"

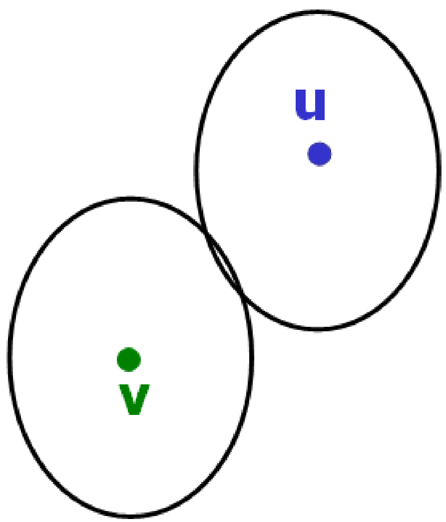


# Analysis outline

- Edge cover covers clients in **G** within 1
  - guess of OPT

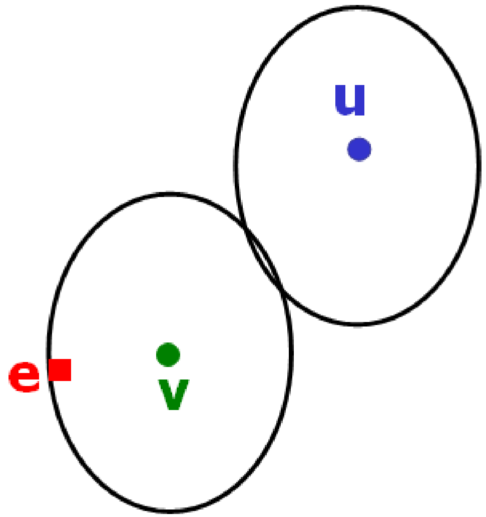
# Analysis outline

- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client u $\notin$ V(G) intersects some v $\in$ V(G)



# Analysis outline

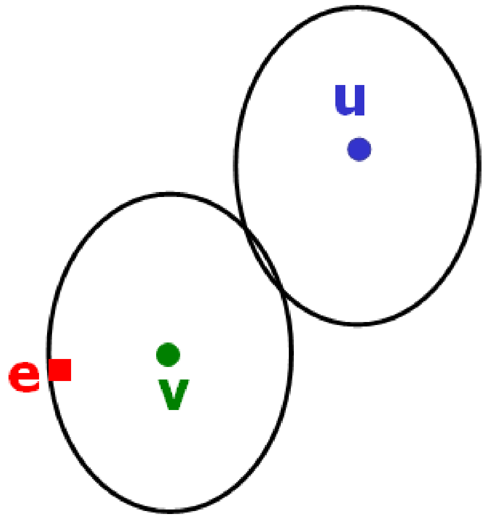
- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client u $\notin$ V(G) intersects some v $\in$ V(G)
  - Facility e $\in$ Ball(v,1) in edge cover



# Analysis outline

- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client u $\notin$ V(G) intersects some v $\in$ V(G)
  - Facility e $\in$ Ball(v,1) in edge cover

**Lemma**: d(u,e) $\leq$ (3-$\rho$) *or* d(u,V(G)) $\leq$ (2-$\rho$).

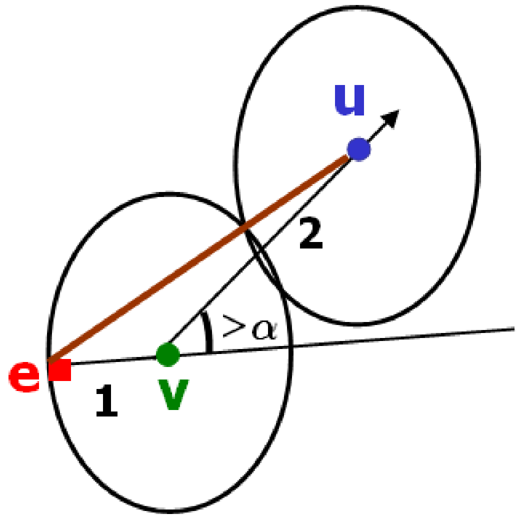


# Analysis outline

- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client u $\notin$ V(G) intersects some v $\in$ V(G)
  - Facility e $\in$ Ball(v,1) in edge cover

**Lemma**: d(u,e) $\leq$ (3-$\rho$) *or* d(u,V(G)) $\leq$ (2-$\rho$).

*u not antipode <e,v>*

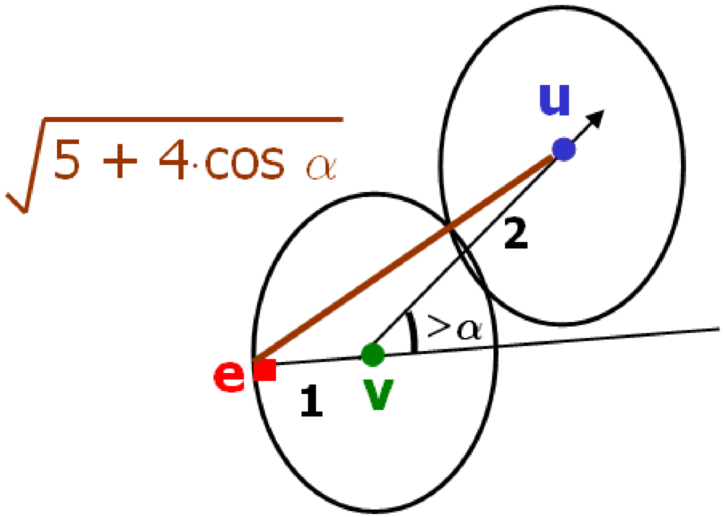


# Analysis outline

- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client u $\notin$ V(G) intersects some v $\in$ V(G)
  - Facility e $\in$ Ball(v,1) in edge cover

**Lemma**: d(u,e) $\leq$ (3-$\rho$) *or* d(u,V(G)) $\leq$ (2-$\rho$).

*u not antipode <e,v>*

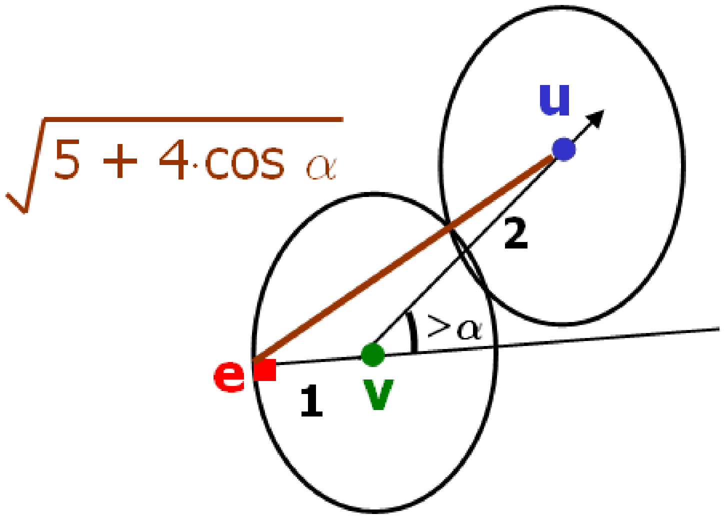


# Analysis outline

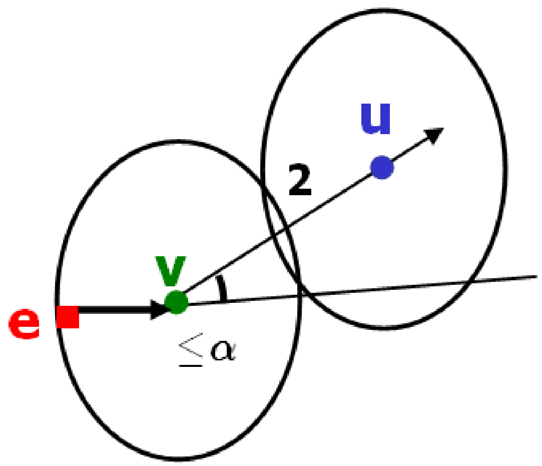
- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client u $\notin$ V(G) intersects some v $\in$ V(G)
  - Facility e $\in$ Ball(v,1) in edge cover

**Lemma**: d(u,e) $\leq$ (3-$\rho$) *or* d(u,V(G)) $\leq$ (2-$\rho$).

*u not antipode <e,v>*




*u antipode <e,v>*

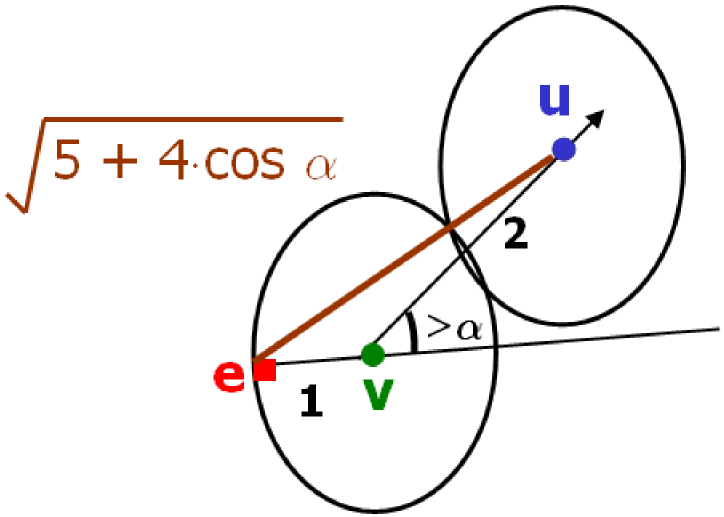


# Analysis outline

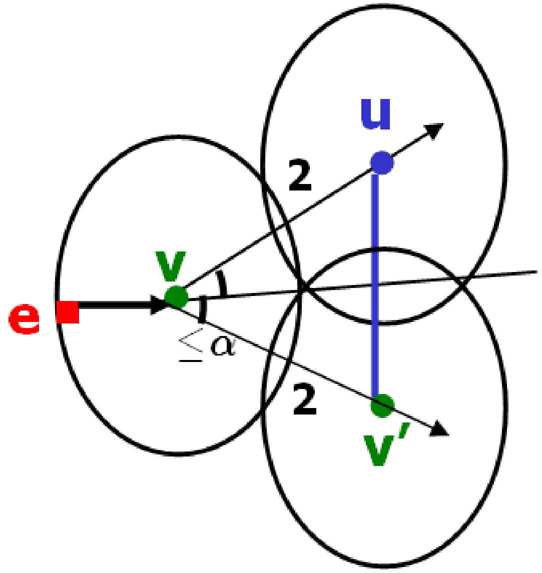
- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client $u \notin V(G)$ intersects some $v \in V(G)$
  - Facility $e \in Ball(v,1)$ in edge cover

**Lemma**: $d(u,e) \leq (3-\rho)$ *or* $d(u,V(G)) \leq (2-\rho)$.

*u not antipode <e,v>*




*u antipode <e,v>*

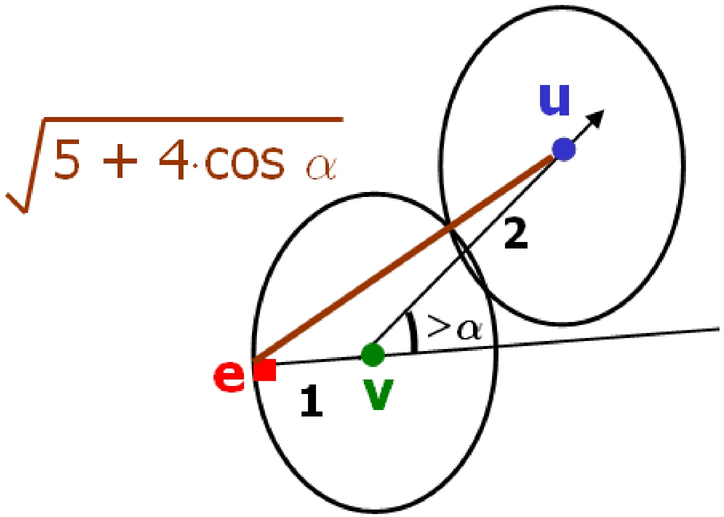


# Analysis outline

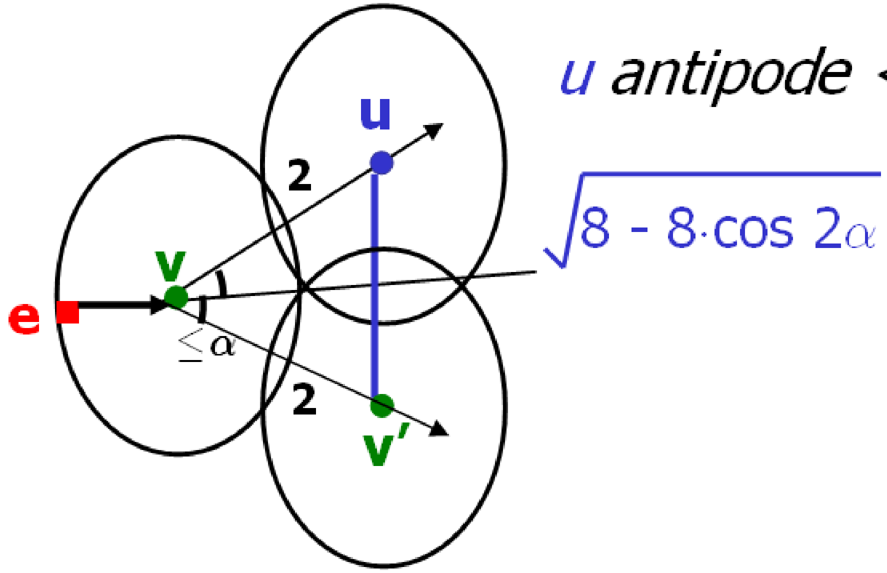
- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client $u \notin V(G)$ intersects some $v \in V(G)$
  - Facility $e \in$ Ball($v$,1) in edge cover

**Lemma**: $d(u,e) \leq (3-\rho)$ *or* $d(u,V(G)) \leq (2-\rho)$.

*u not antipode <e,v>*




*u antipode <e,v>*

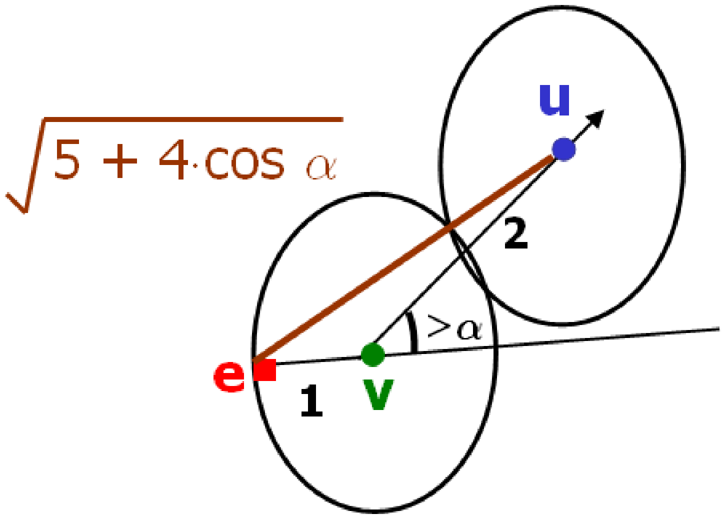


# Analysis outline

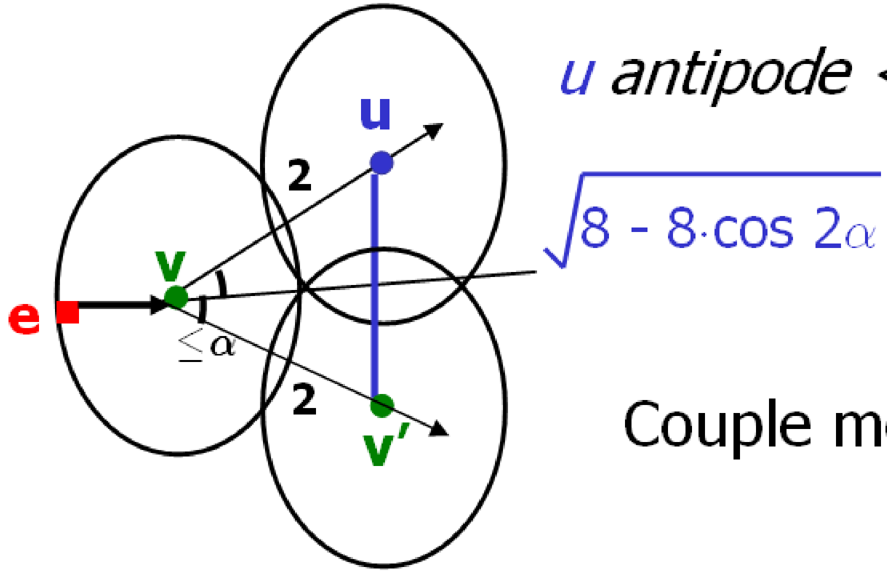
- Edge cover covers clients in **G** within 1
  - guess of OPT
- Client u $\notin$ V(G) intersects some v $\in$ V(G)
  - Facility e $\in$ Ball(v,1) in edge cover

**Lemma**: d(u,e) $\leq$ (3-$\rho$) *or* d(u,V(G)) $\leq$ (2-$\rho$).

*u not antipode <e,v>*




*u antipode <e,v>*




Couple more cases..

# Open Questions

- Tight approximation ratio ?
  - $\sqrt{7} \leq \alpha \leq 1+\sqrt{3}$
- Linear time $1+\sqrt{3}$ approximation ?
- Euclidean *k-Center* better than 2 ?
  - $\sqrt{3} \leq \alpha \leq 2$

# Open Questions

- Tight approximation ratio ?
  - $\sqrt{7} \le \alpha \le 1+\sqrt{3}$
- Linear time $1+\sqrt{3}$ approximation ?
- Euclidean *k-Center* better than 2 ?
  - $\sqrt{3} \le \alpha \le 2$
- "Subset" k-Supplier better than 3 ?
  - facilities $\subseteq$ clients

# Open Questions

- Tight approximation ratio ?
  - $\sqrt{7} \leq \alpha \leq 1+\sqrt{3}$
- Linear time $1+\sqrt{3}$ approximation ?
- Euclidean *k-Center* better than 2 ?
  - $\sqrt{3} \leq \alpha \leq 2$
- "Subset" k-Supplier better than 3 ?
  - facilities $\subseteq$ clients

**Thank You !**