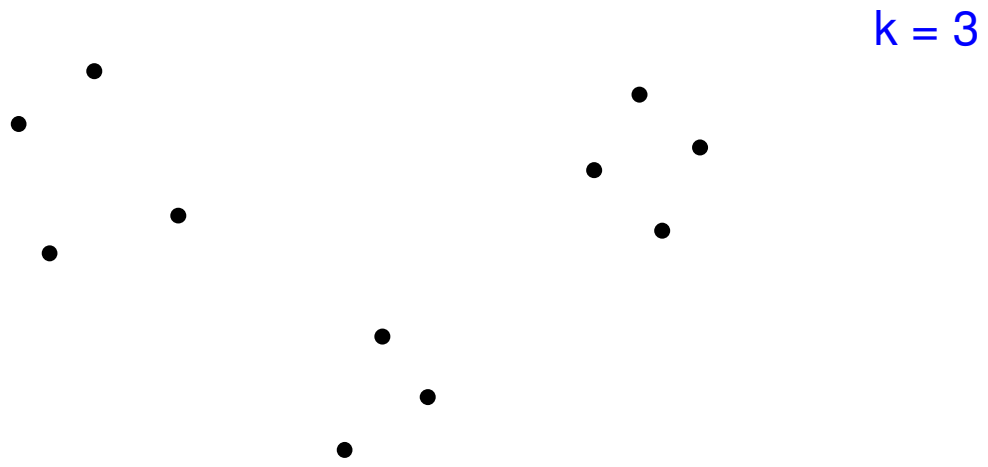
Streaming algorithms for *k*-center clustering with outliers and with anonymity

Richard Matthew McCutchen and Samir Khuller
University of Maryland
{rmccutch,samir}@cs.umd.edu

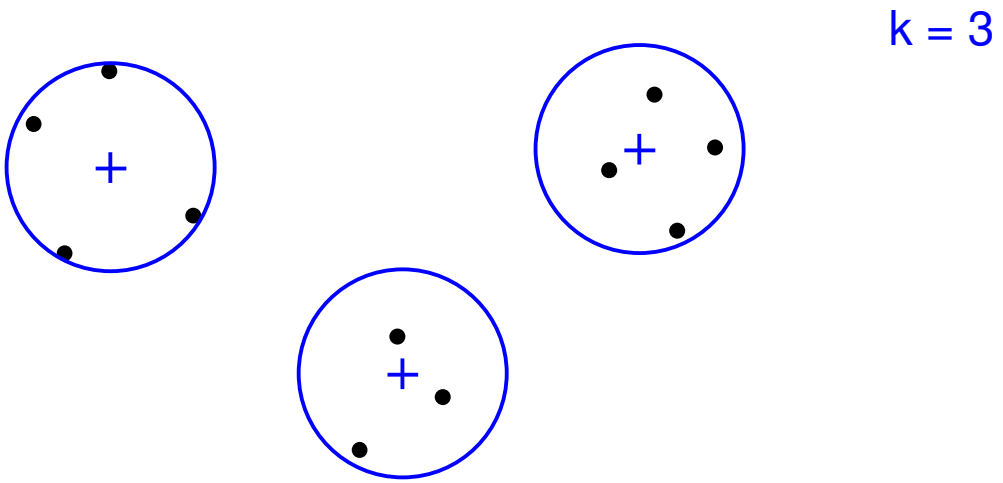
k-center clustering problem

- Input: *n* points in an arbitrary metric space.
- Goal: Partition them into *k* clusters and assign each a center point to minimize the maximum distance from an input point to its cluster center.



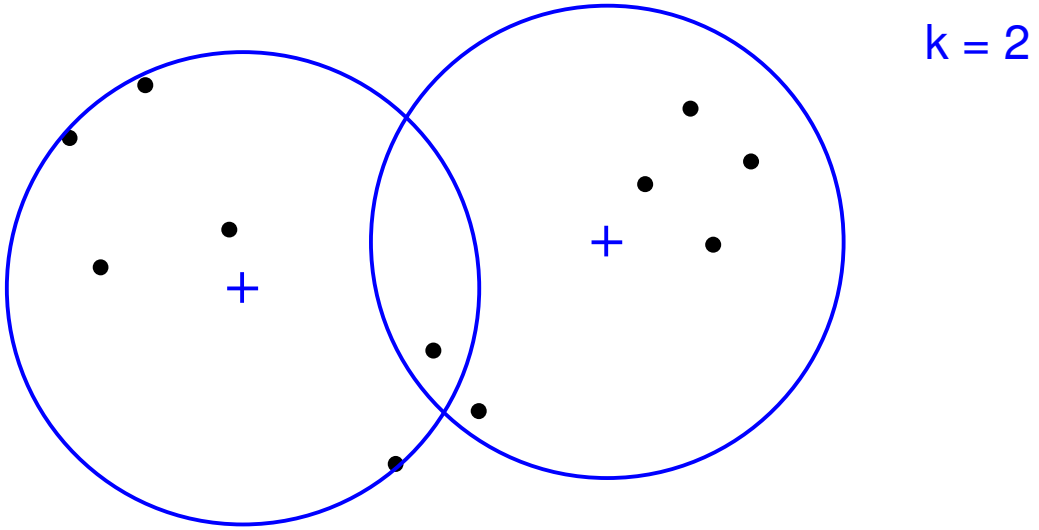
k-center clustering problem

- Input: *n* points in an arbitrary metric space.
- Goal: Partition them into *k* clusters and assign each a center point to minimize the maximum distance from an input point to its cluster center.



k-center clustering problem

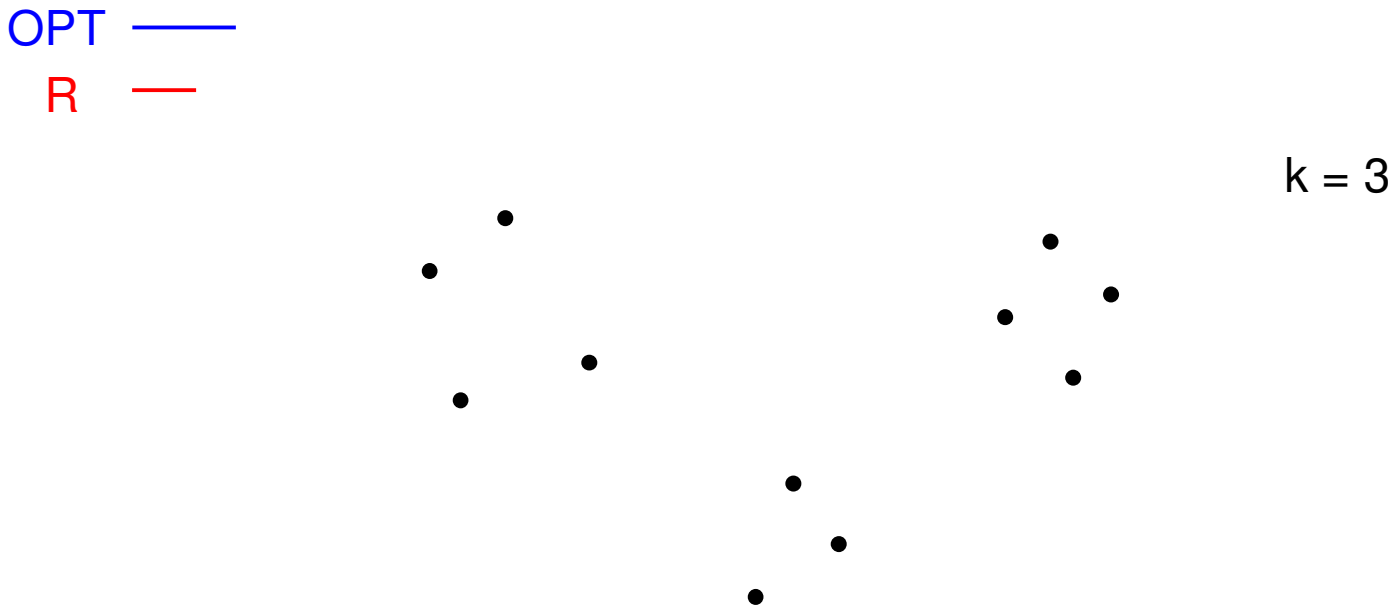
- Input: *n* points in an arbitrary metric space.
- Goal: Partition them into *k* clusters and assign each a center point to minimize the maximum distance from an input point to its cluster center.



Greedy 2-approximation

(Hochbaum and Shmoys, 1985)

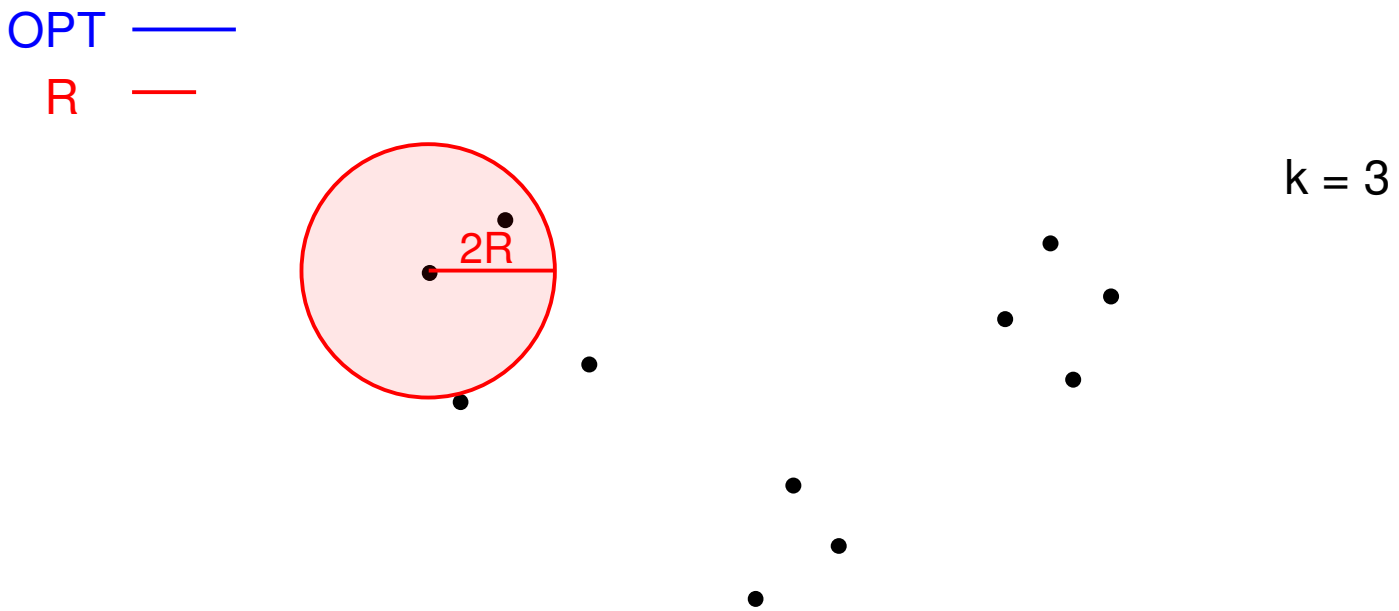
- Greedily make clusters of radius 2R centered at uncovered points
- Take smallest R for which ≤ k clusters suffice



Greedy 2-approximation

(Hochbaum and Shmoys, 1985)

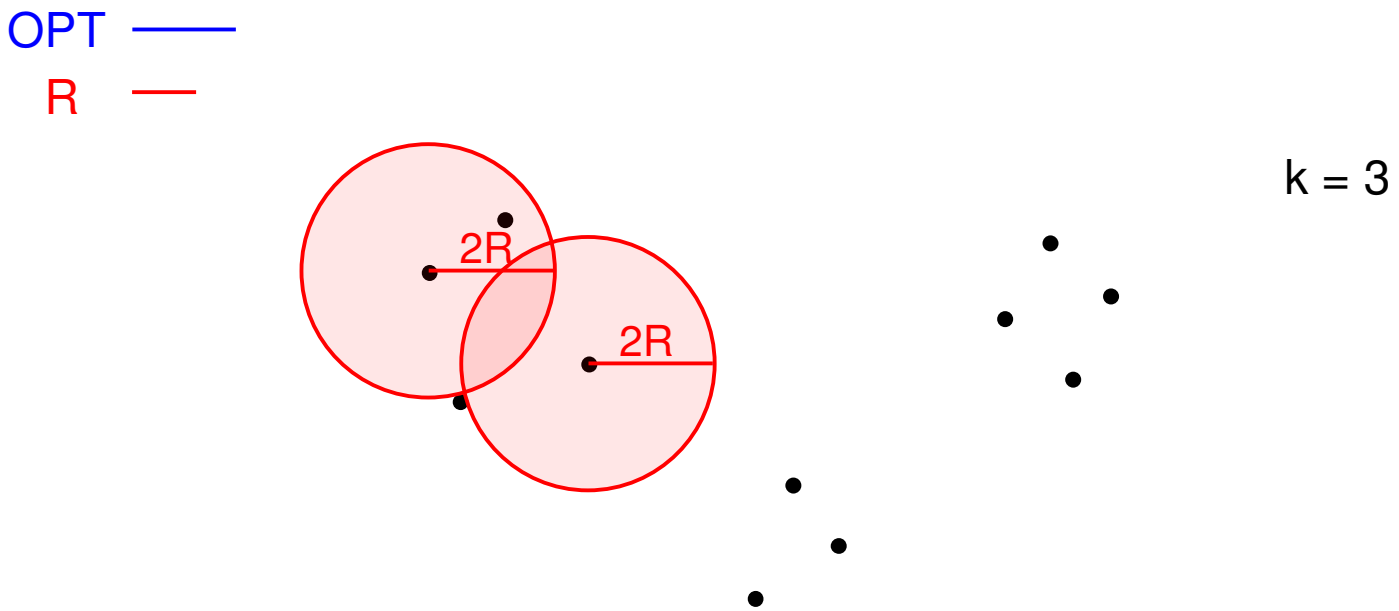
- Greedily make clusters of radius 2R centered at uncovered points
- Take smallest R for which ≤ k clusters suffice



Greedy 2-approximation

(Hochbaum and Shmoys, 1985)

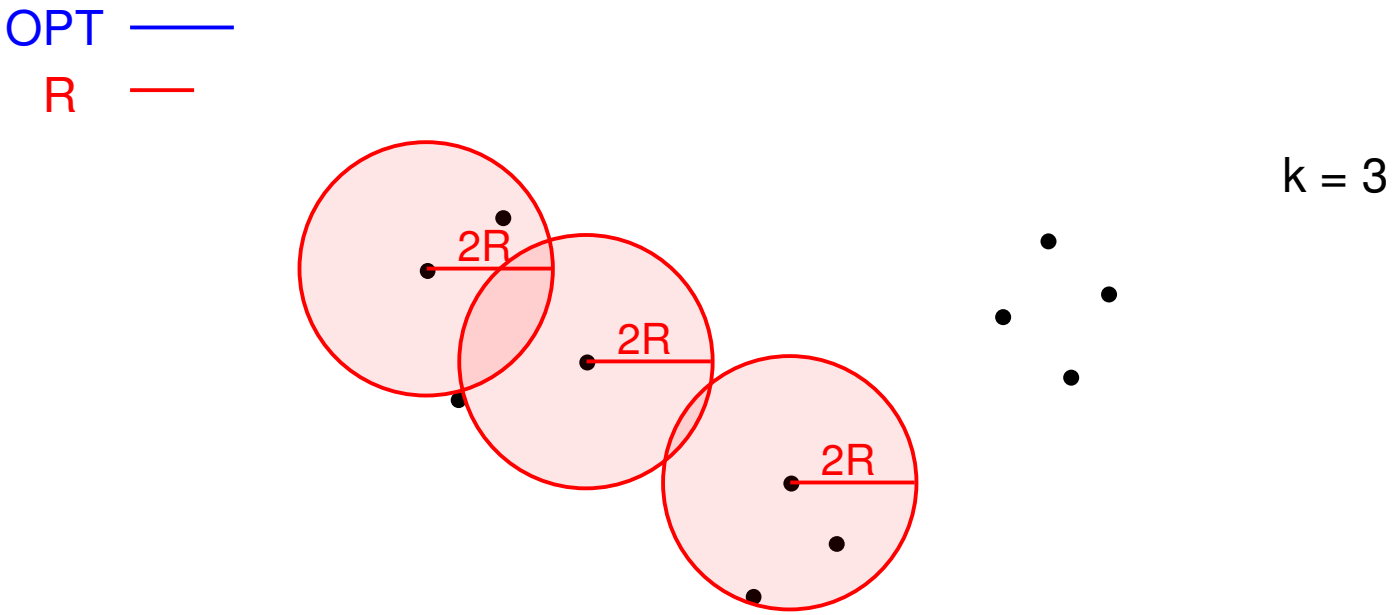
- Greedily make clusters of radius 2R centered at uncovered points
- Take smallest R for which ≤ k clusters suffice



Greedy 2-approximation

(Hochbaum and Shmoys, 1985)

- Greedily make clusters of radius 2R centered at uncovered points
- Take smallest R for which ≤ k clusters suffice

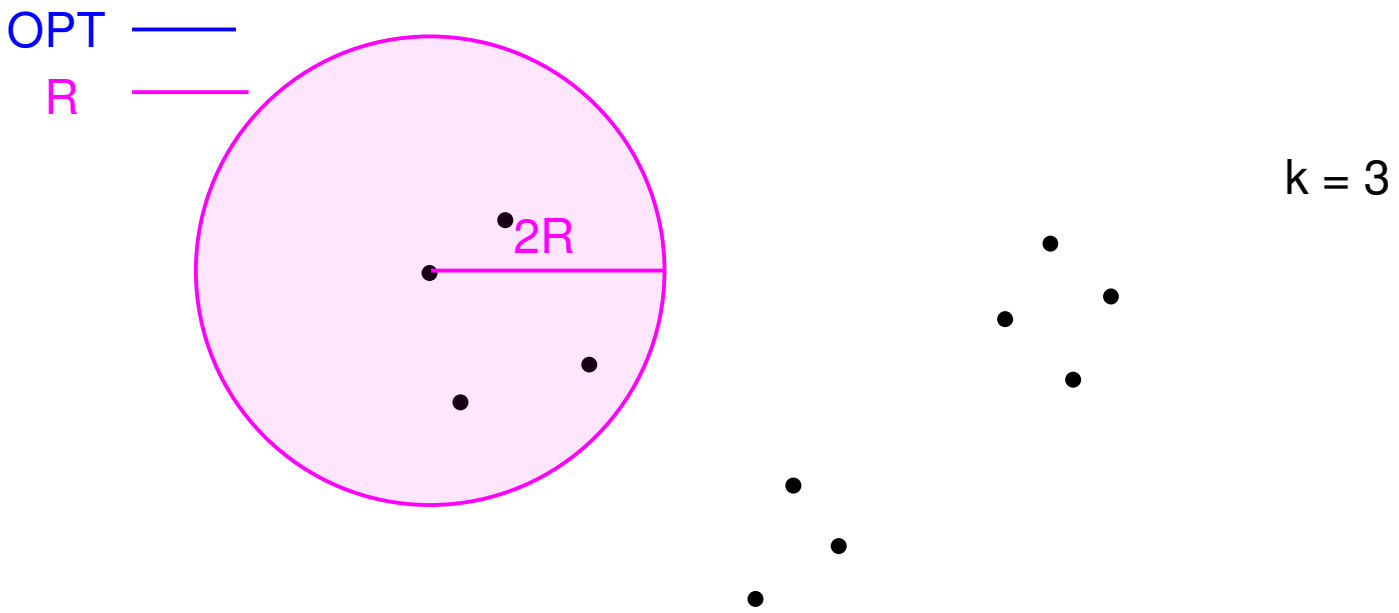


k clusters, but points left uncovered ⇒ R too small. Start over w/ bigger guess.

Greedy 2-approximation

(Hochbaum and Shmoys, 1985)

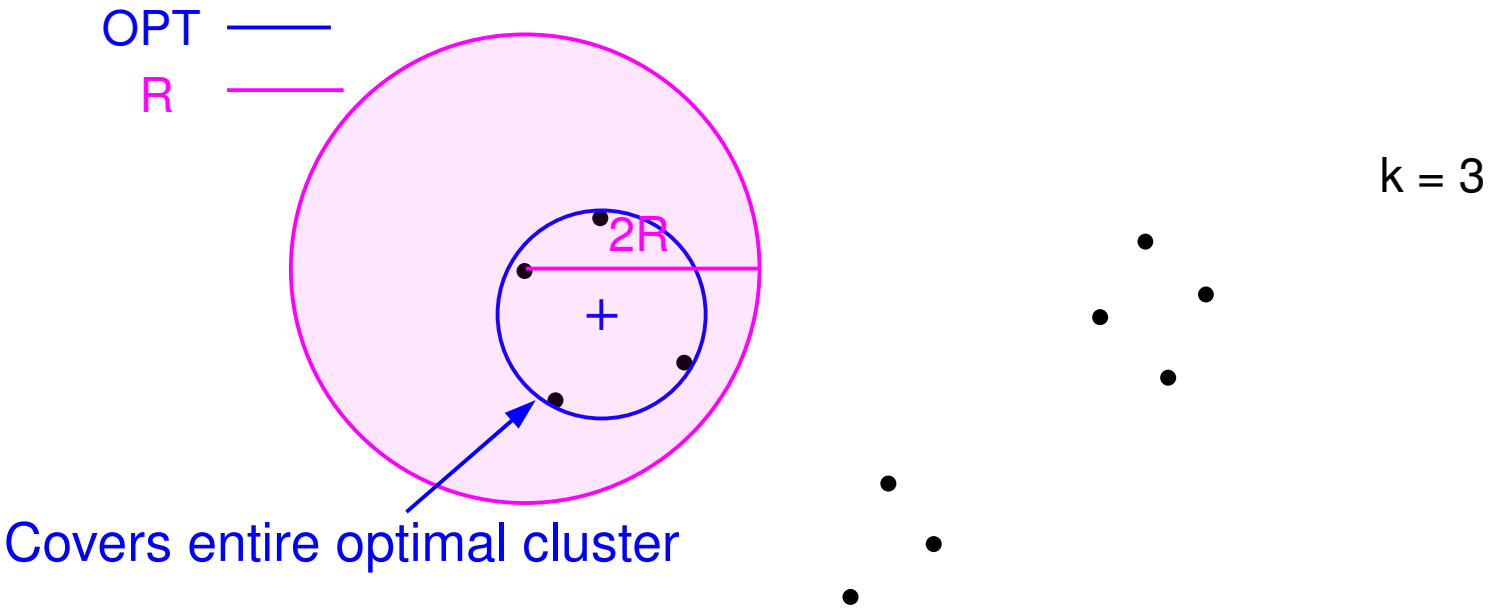
- Greedily make clusters of radius 2R centered at uncovered points
- Take smallest R for which ≤ k clusters suffice



Greedy 2-approximation

(Hochbaum and Shmoys, 1985)

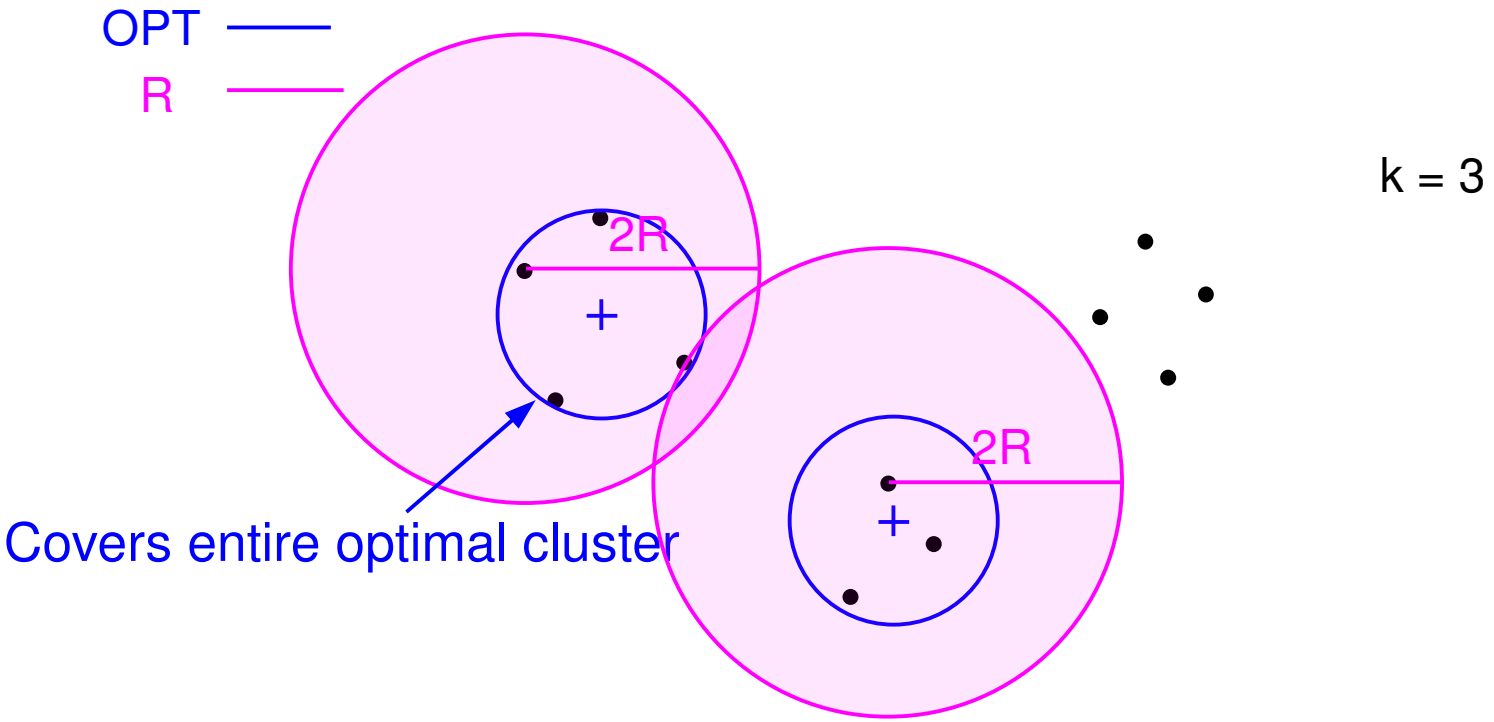
- Greedily make clusters of radius 2R centered at uncovered points
- Take smallest R for which ≤ k clusters suffice



Greedy 2-approximation

(Hochbaum and Shmoys, 1985)

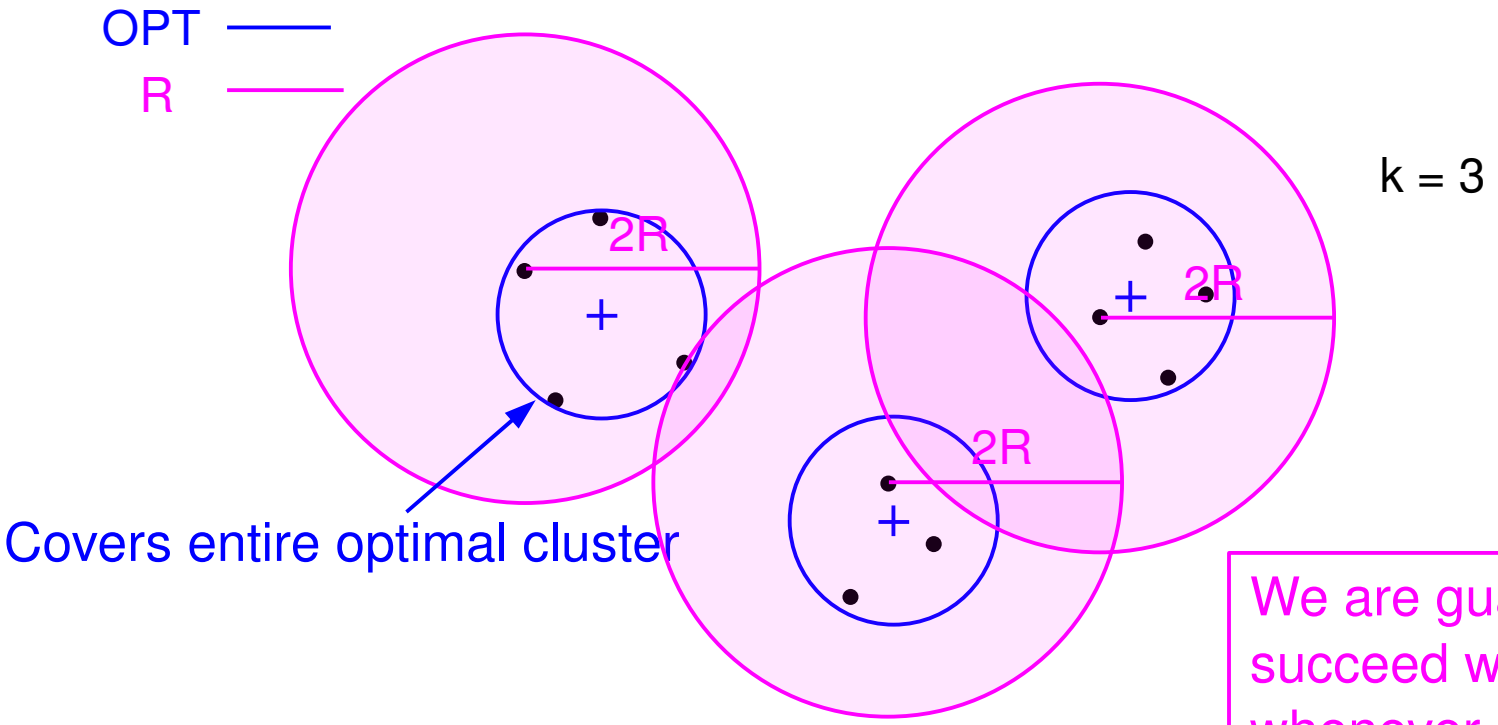
- Greedily make clusters of radius 2R centered at uncovered points
- Take smallest R for which ≤ k clusters suffice



Greedy 2-approximation

(Hochbaum and Shmoys, 1985)

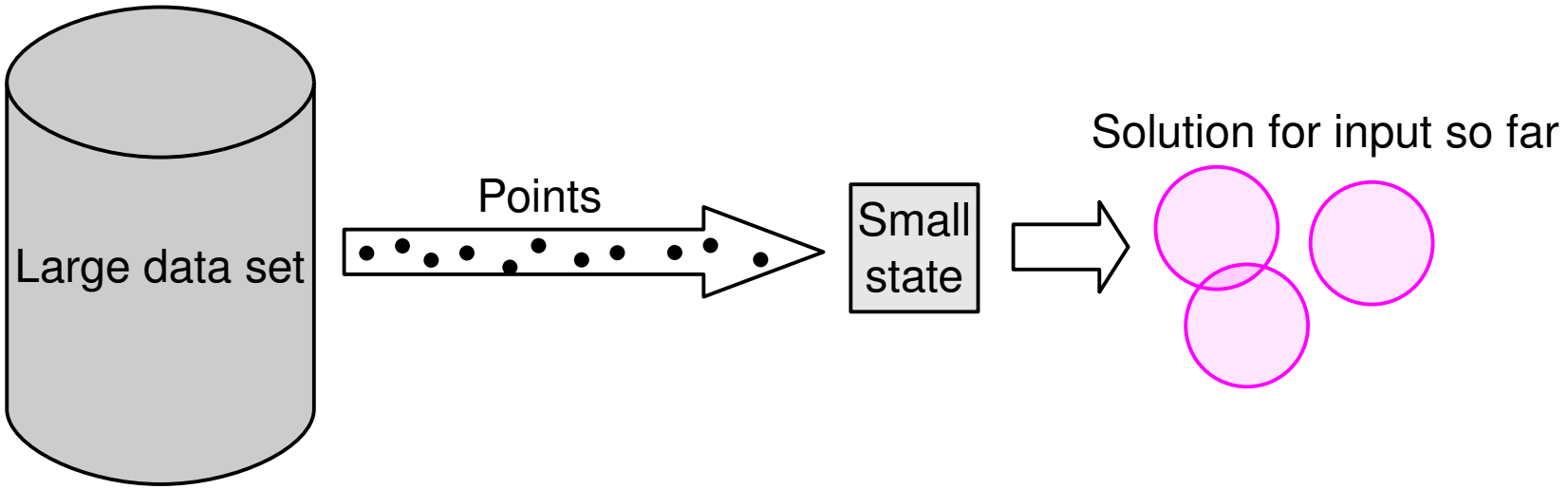
- Greedily make clusters of radius 2R centered at uncovered points
- Take smallest R for which ≤ k clusters suffice



We are guaranteed to succeed w/ ≤ k clusters whenever R ≥ OPT no matter how badly we choose centers, hence we'll take R ≤ OPT.

Streaming model

- Data set too large to fit in memory
- Receive points one at a time (can't start over!)
- Maintain small state, incl. solution for input so far
- Return solution when end of input is reached



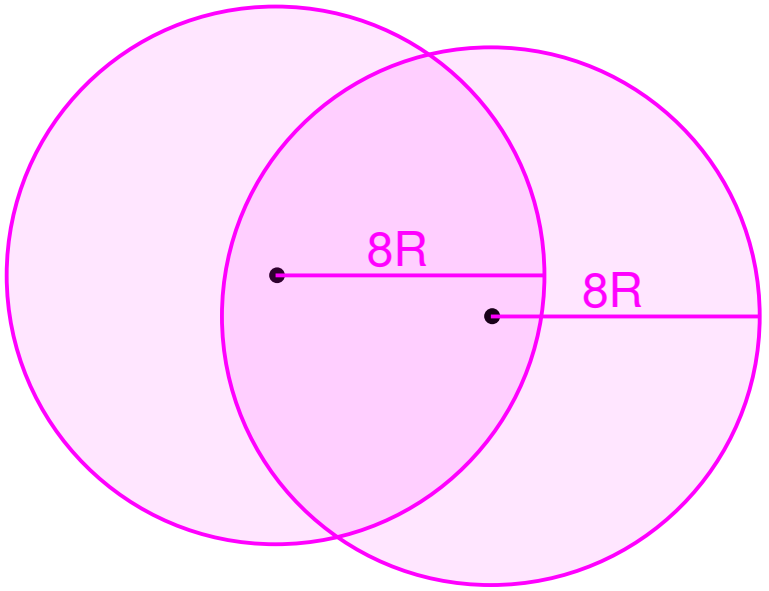
Doubling Algorithm

(Charikar et al., STOC 1997)

- State:
  - Lower bound R on optimal radius
  - ≤ k "stored centers" such that every input point read so far is within 8R of a stored center
  - ⇒ Stored centers give an 8-approximation at any time
- If an input point is within 8R of a stored center, then drop it, otherwise store it.

R –

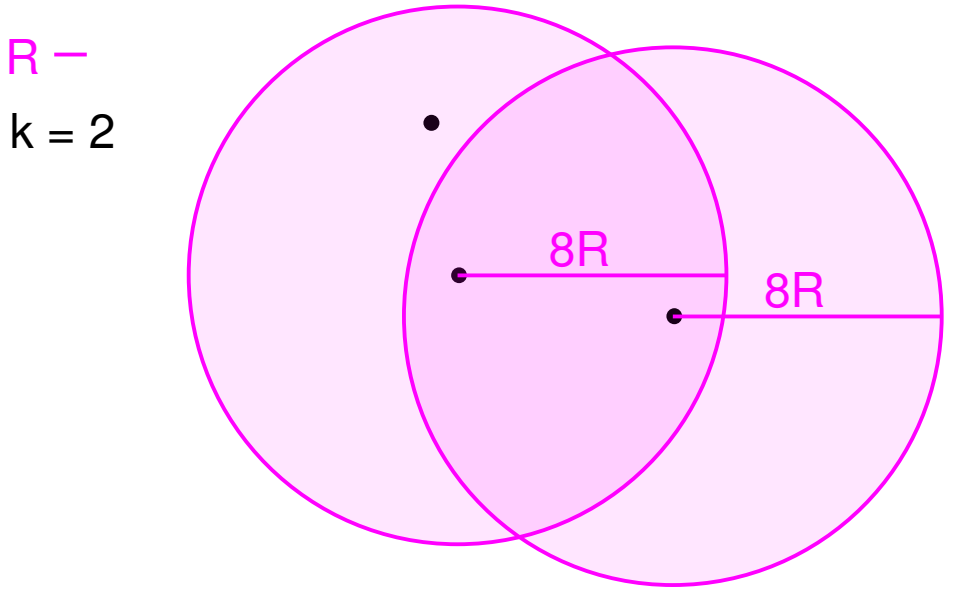
k = 2



Doubling Algorithm

(Charikar et al., STOC 1997)

- State:
  - Lower bound R on optimal radius
  - ≤ k "stored centers" such that every input point read so far is within 8R of a stored center
  - ⇒ Stored centers give an 8-approximation at any time
- If an input point is within 8R of a stored center, then drop it, otherwise store it.



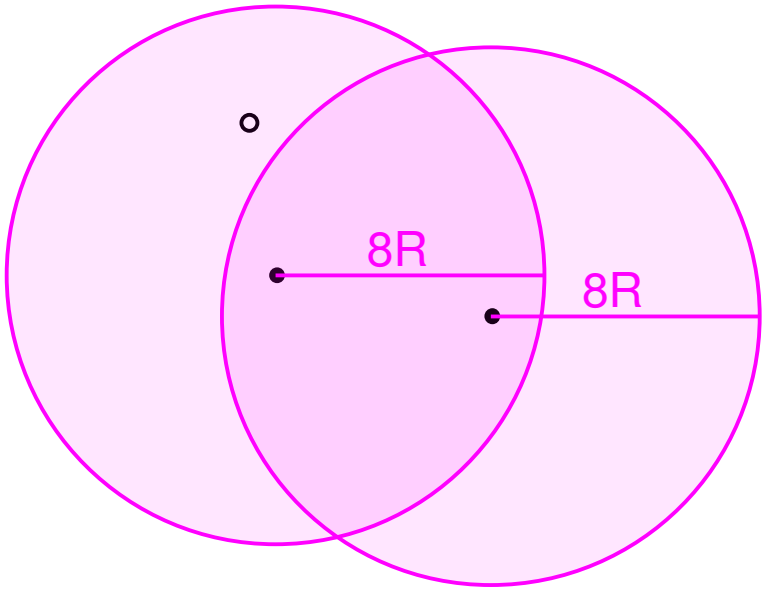
Doubling Algorithm

(Charikar et al., STOC 1997)

- State:
  - Lower bound R on optimal radius
  - ≤ k "stored centers" such that every input point read so far is within 8R of a stored center
  - ⇒ Stored centers give an 8-approximation at any time
- If an input point is within 8R of a stored center, then drop it, otherwise store it.

R –

k = 2



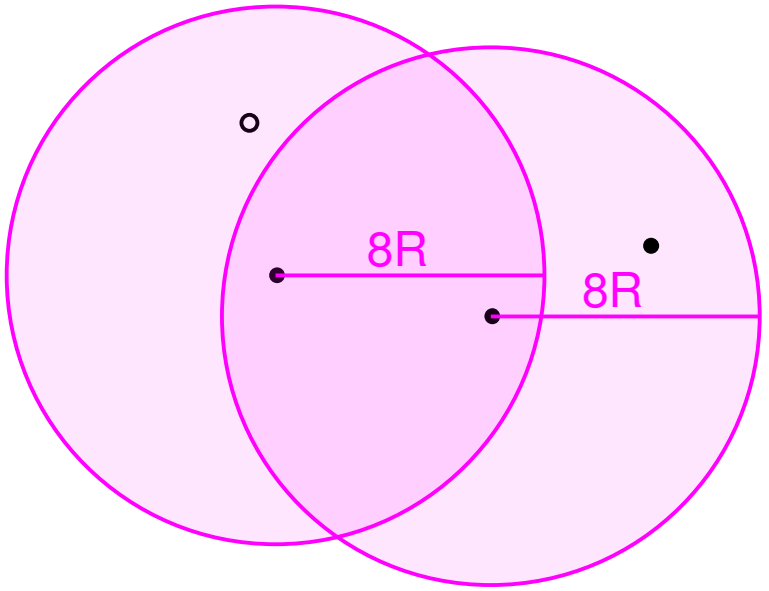
Doubling Algorithm

(Charikar et al., STOC 1997)

- State:
  - Lower bound R on optimal radius
  - ≤ k "stored centers" such that every input point read so far is within 8R of a stored center
  - ⇒ Stored centers give an 8-approximation at any time
- If an input point is within 8R of a stored center, then drop it, otherwise store it.

R –

k = 2



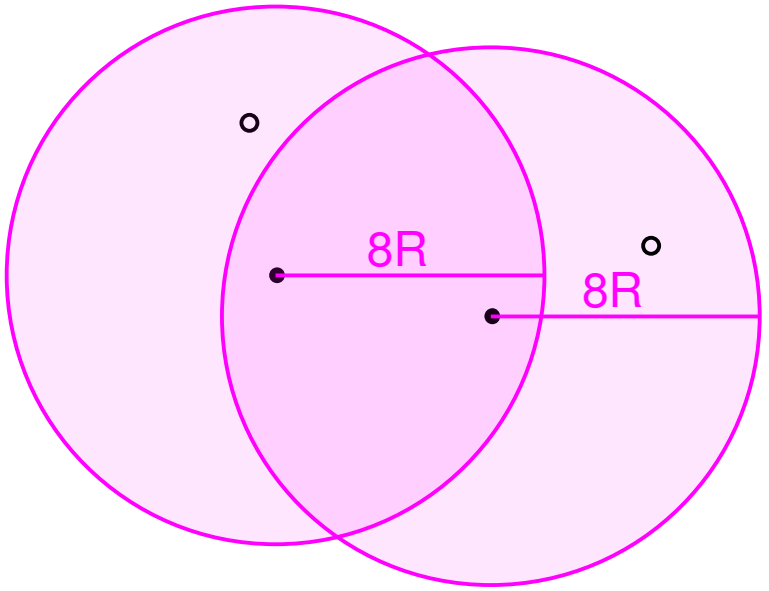
Doubling Algorithm

(Charikar et al., STOC 1997)

- State:
  - Lower bound R on optimal radius
  - ≤ k "stored centers" such that every input point read so far is within 8R of a stored center
  - ⇒ Stored centers give an 8-approximation at any time
- If an input point is within 8R of a stored center, then drop it, otherwise store it.

R –

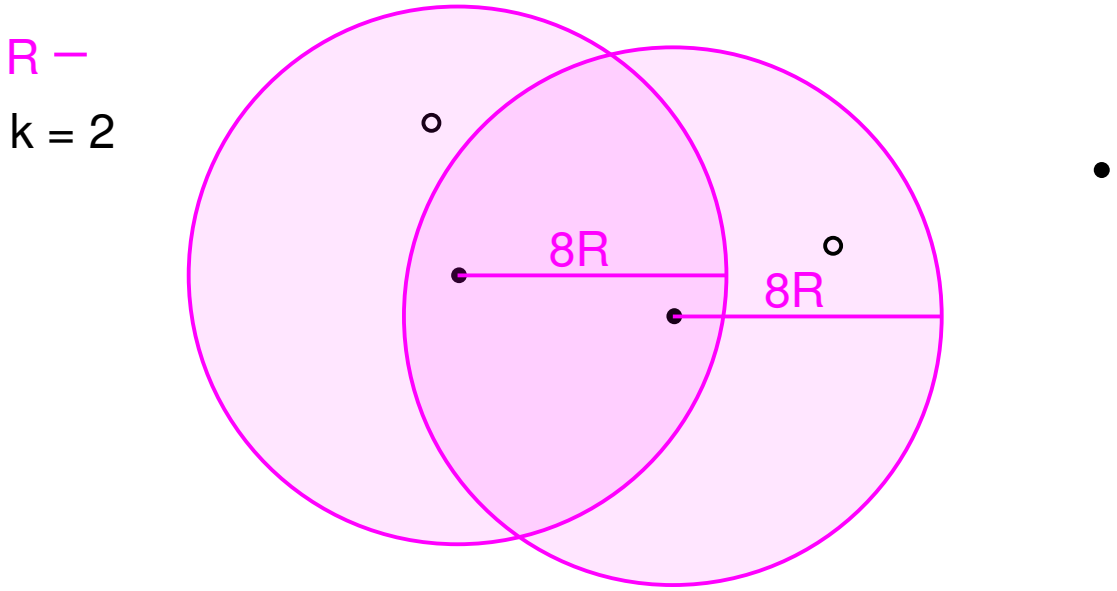
k = 2



Doubling Algorithm

(Charikar et al., STOC 1997)

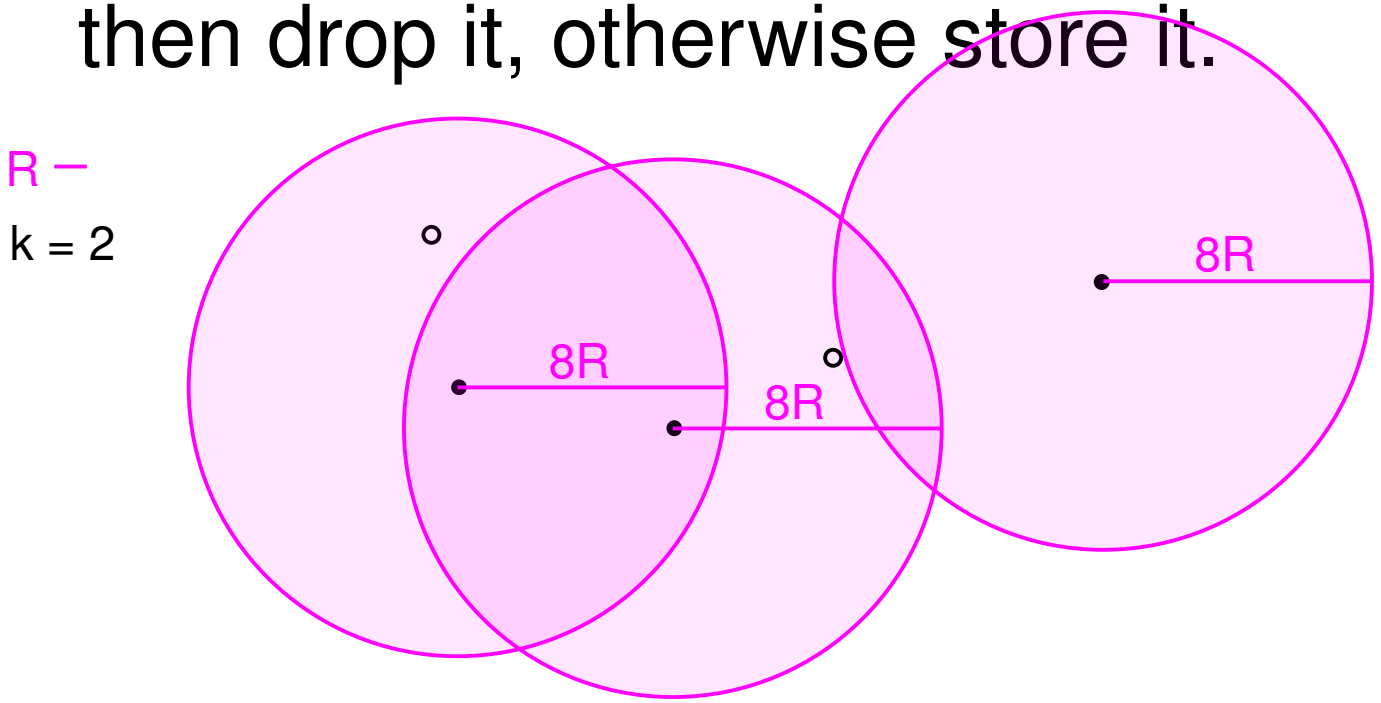
- State:
  - Lower bound R on optimal radius
  - ≤ k "stored centers" such that every input point read so far is within 8R of a stored center
  - ⇒ Stored centers give an 8-approximation at any time
- If an input point is within 8R of a stored center, then drop it, otherwise store it.



Doubling Algorithm

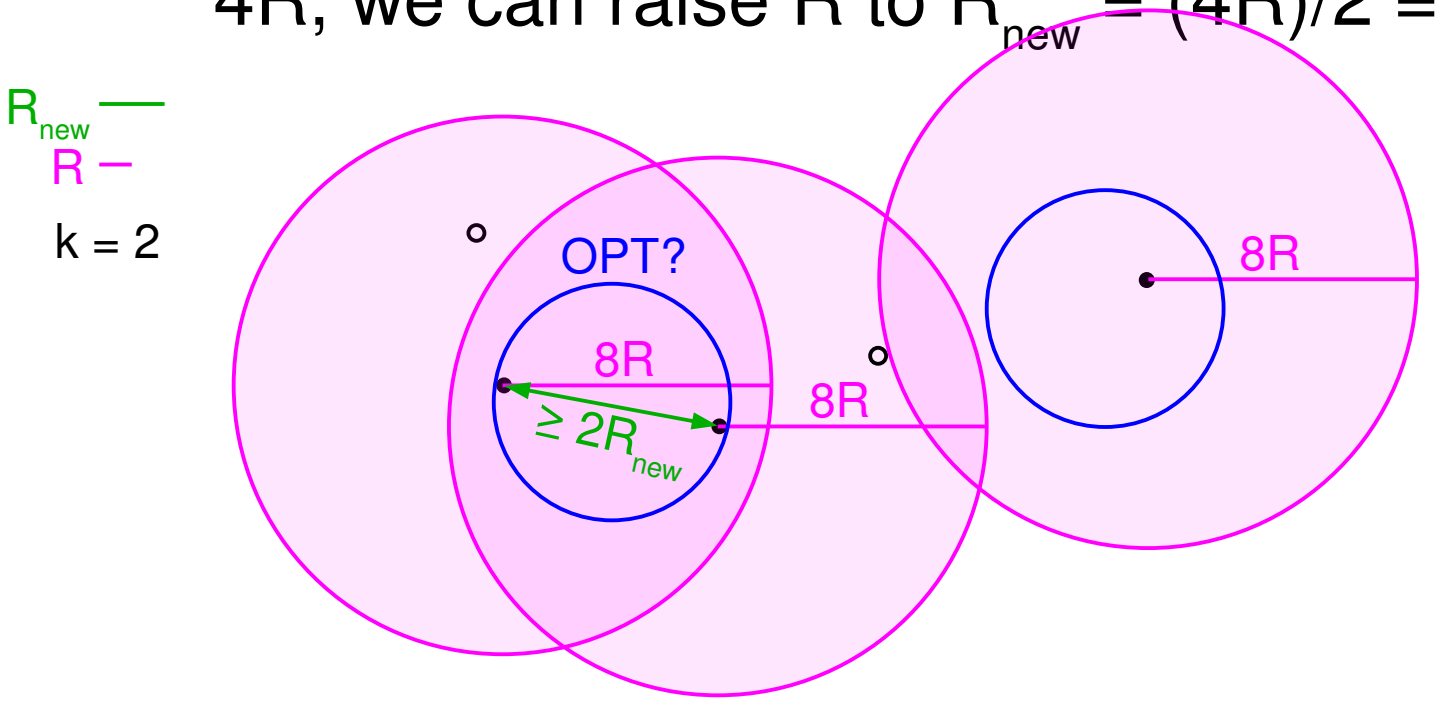
(Charikar et al., STOC 1997)

- State:
  - Lower bound R on optimal radius
  - ≤ k "stored centers" such that every input point read so far is within 8R of a stored center
  - ⇒ Stored centers give an 8-approximation at any time
- If an input point is within 8R of a stored center, then drop it, otherwise store it.



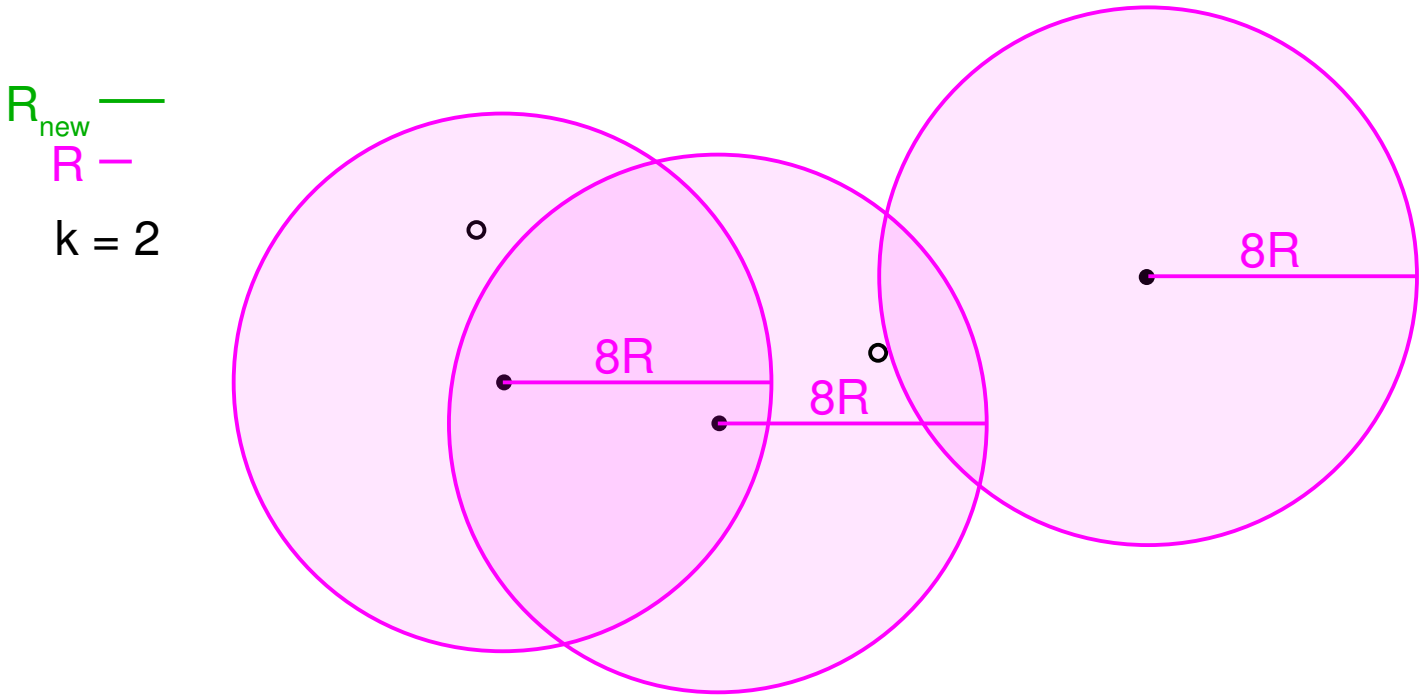
Doubling Algorithm: raising R

- Oops, we have > k stored centers!
  - Must drop some *and* account for the input points they covered within distance 8R.
  - Obs: Some optimal cluster must cover two stored centers, so OPT ≥ (shortest pairwise distance)/2.
  - *Assuming* that stored centers are always separated by 4R, we can raise R to $R_{new} = (4R)/2 = 2R$.



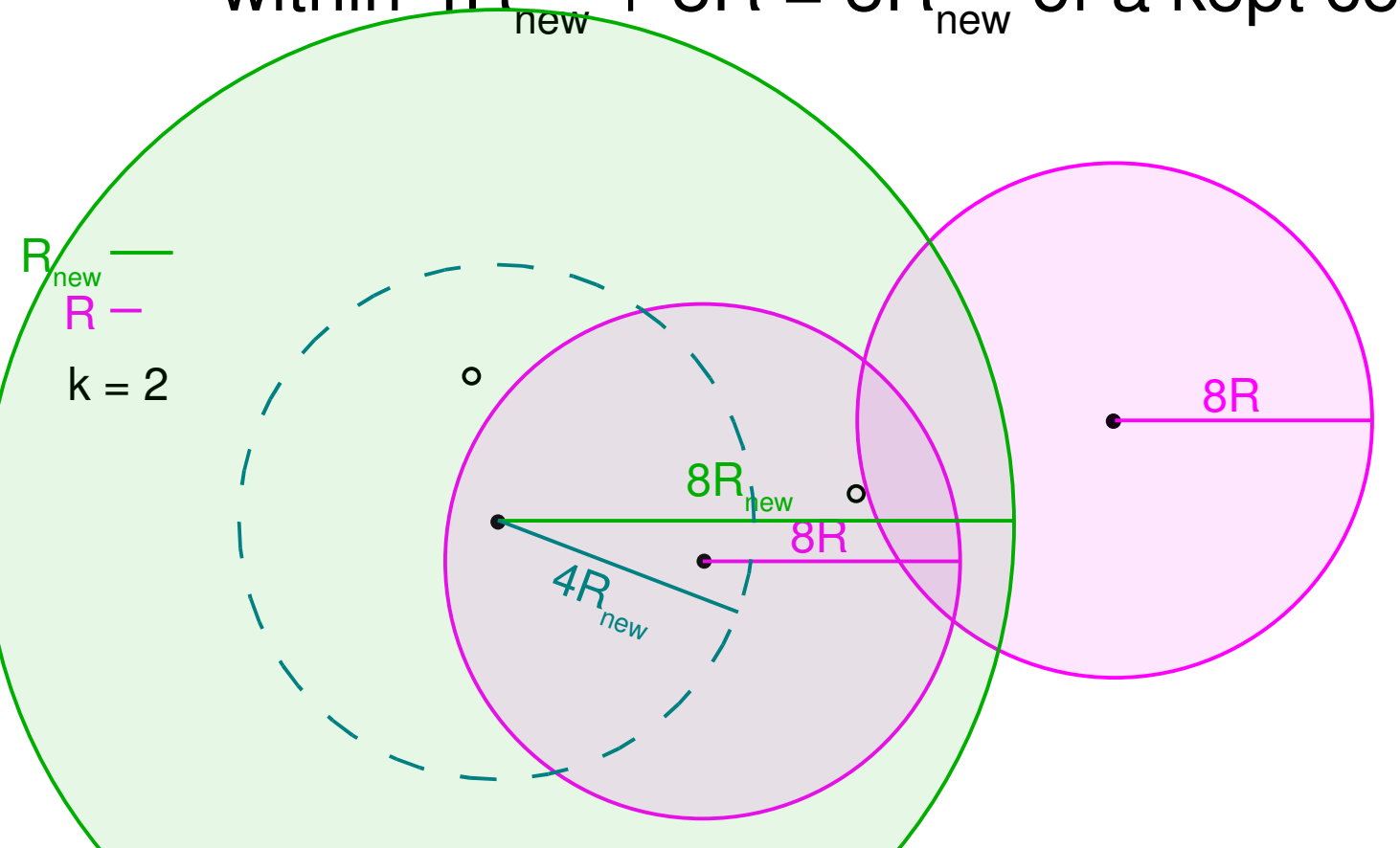
Doubling Algorithm: merging step

- Oops, we have > k stored centers!
  - Restore separation invariant by letting each center greedily subsume others within $4R_{new}$.
  - Every input point belonging to a subsumed center is within $4R_{new} + 8R = 8R_{new}$ of a kept center $\Rightarrow$ covered.



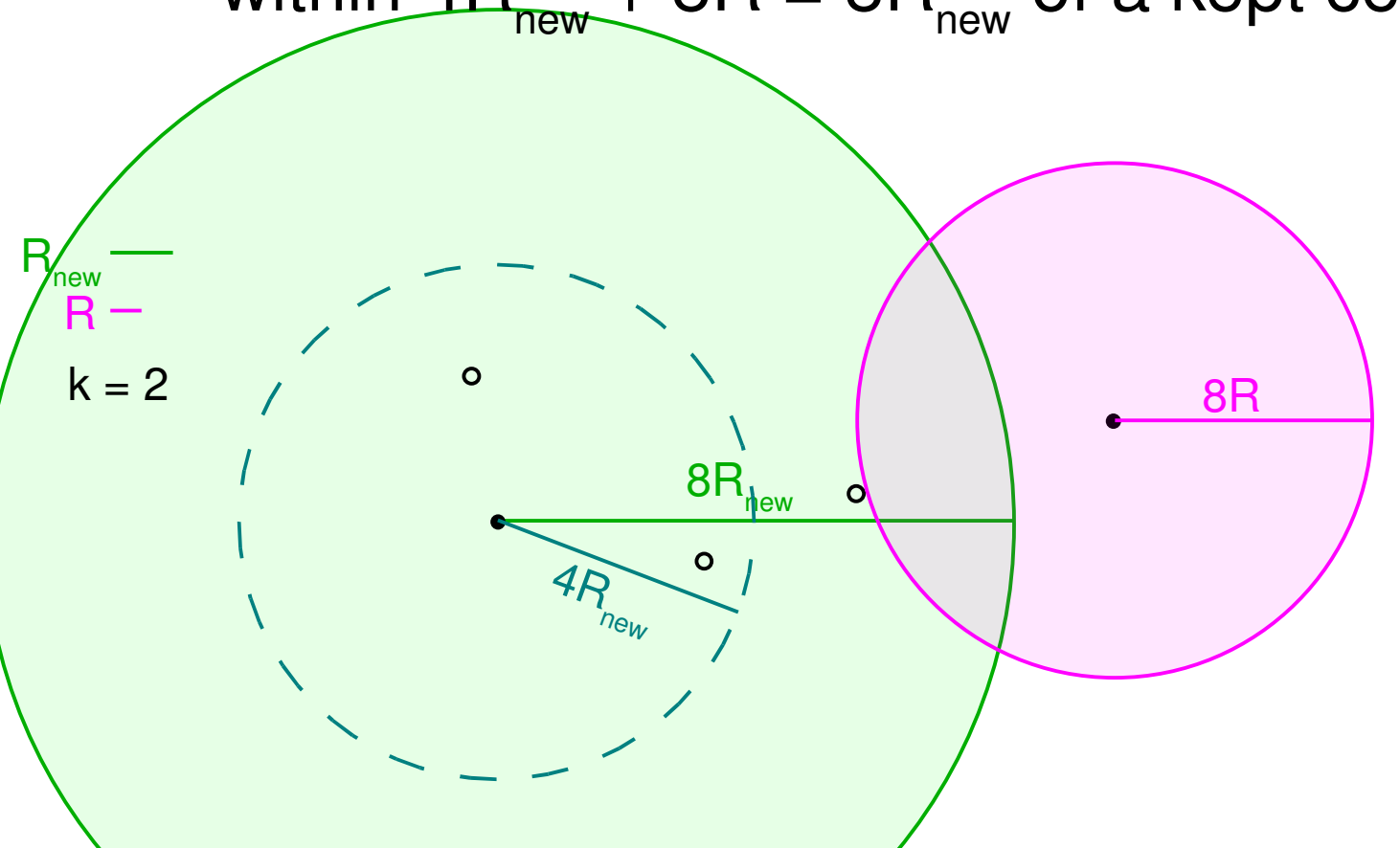
Doubling Algorithm: merging step

- Oops, we have > k stored centers!
  - Restore separation invariant by letting each center greedily subsume others within $4R_{new}$.
  - Every input point belonging to a subsumed center is within $4R_{new} + 8R = 8R_{new}$ of a kept center ⇒ covered.



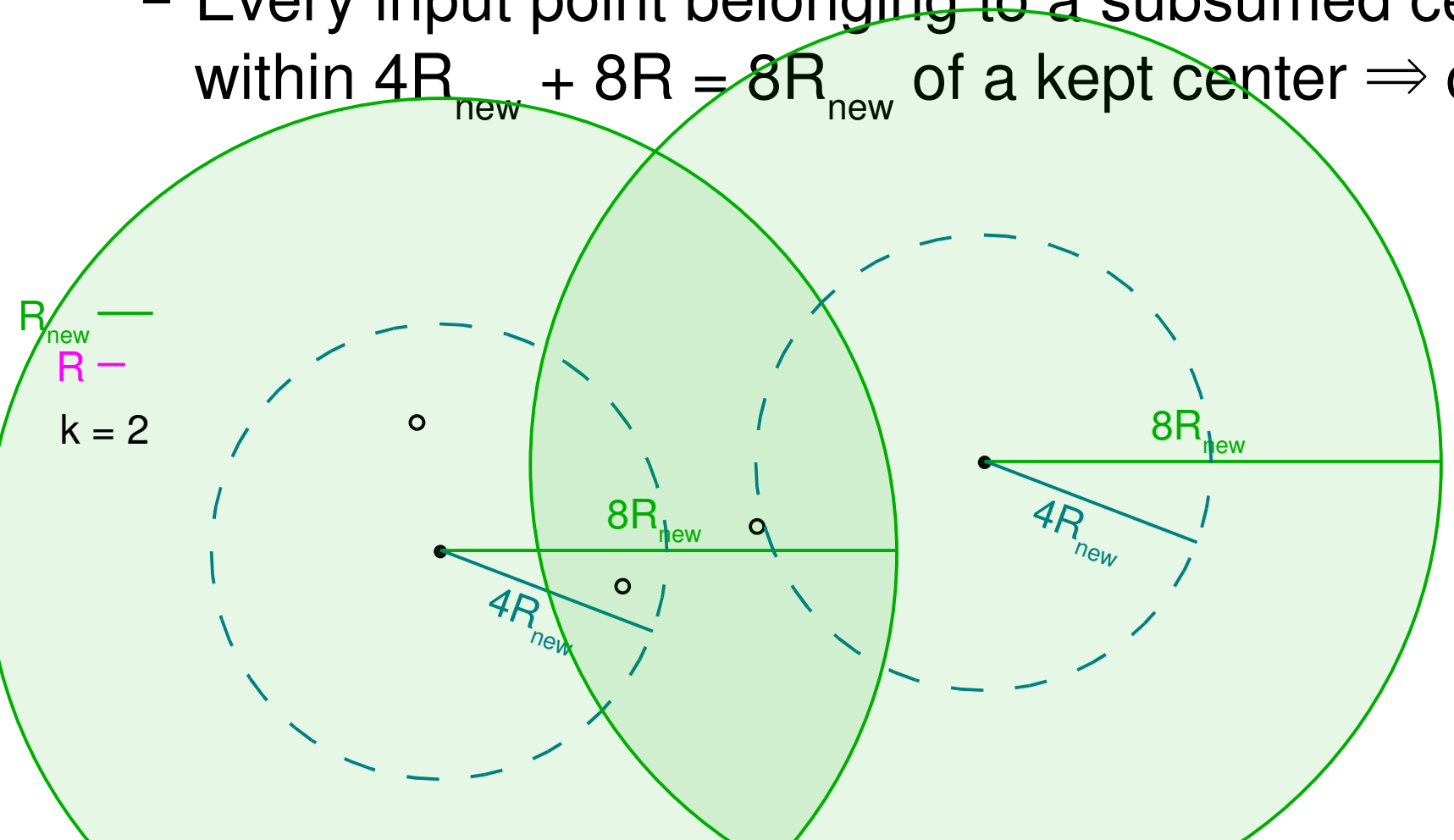
Doubling Algorithm: merging step

- Oops, we have > k stored centers!
  - Restore separation invariant by letting each center greedily subsume others within $4R_{new}$.
  - Every input point belonging to a subsumed center is within $4R_{new} + 8R = 8R_{new}$ of a kept center ⇒ covered.



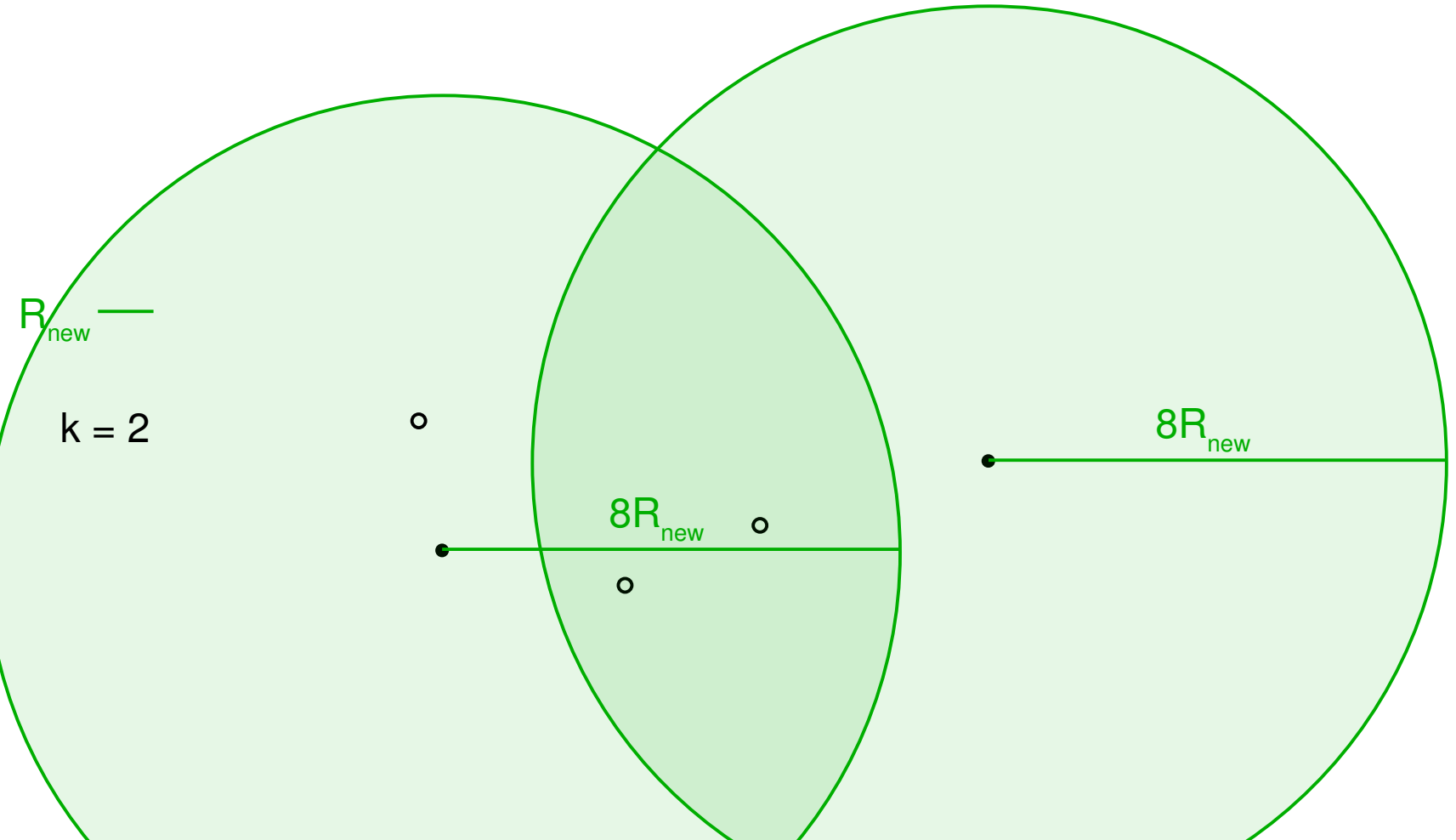
Doubling Algorithm: merging step

- Oops, we have > k stored centers!
  - Restore separation invariant by letting each center greedily subsume others within $4R_{new}$.
  - Every input point belonging to a subsumed center is within $4R_{new} + 8R = 8R_{new}$ of a kept center ⇒ covered.



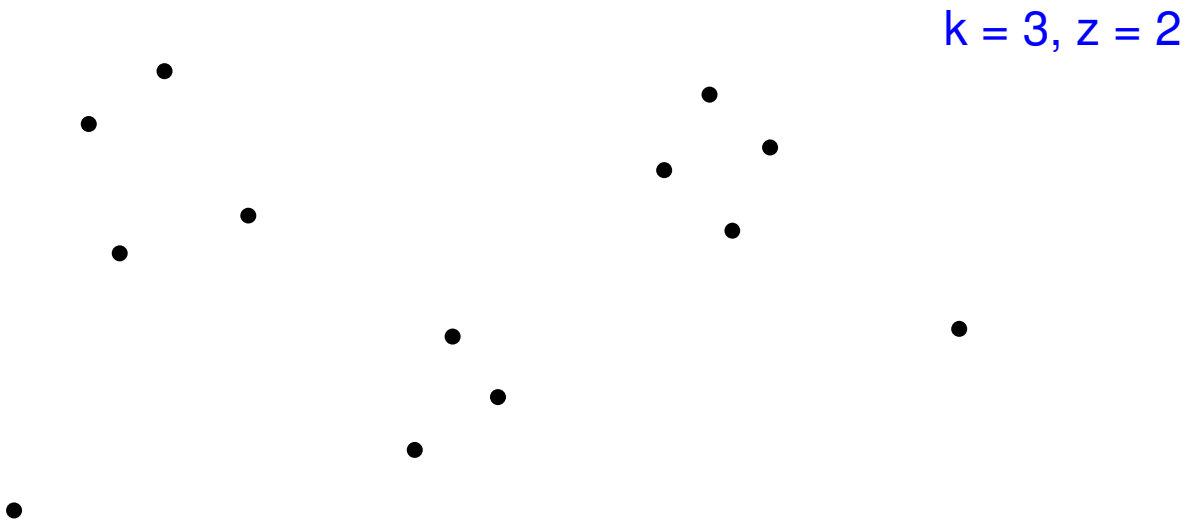
Doubling Algorithm: conclusion

- Proceed...
- When end of input is reached, return clusters of radius 8R at stored centers. An 8-approximation.



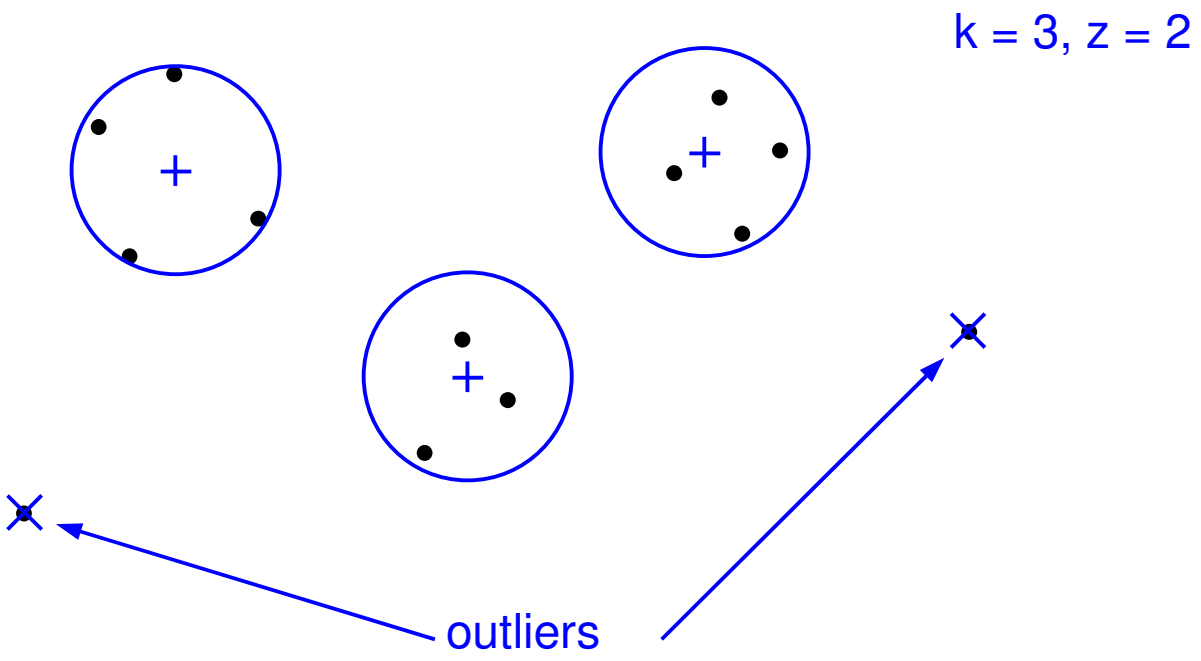
k-center clustering with outliers

- Application: Noisy data
- Clustering can miss up to *z* input points



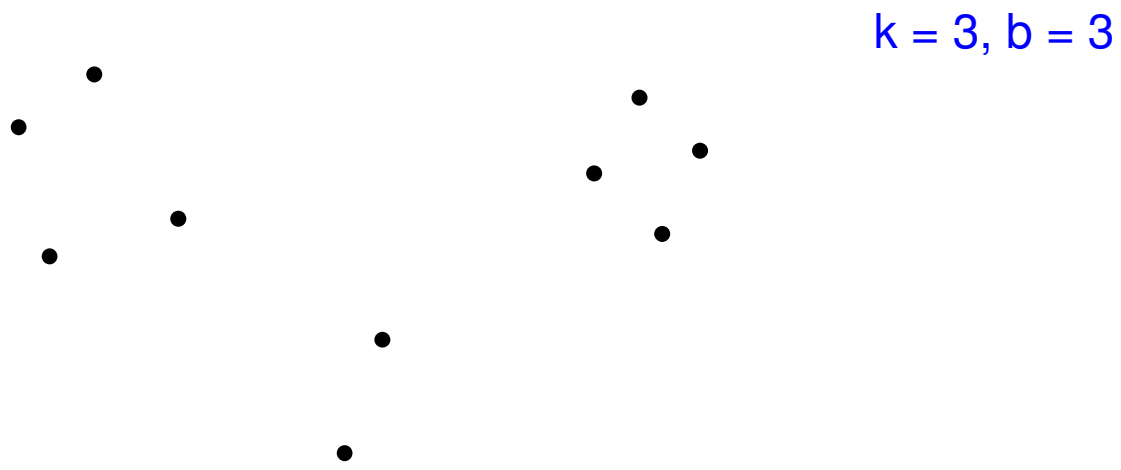
k-center clustering with outliers

- Application: Noisy data
- Clustering can miss up to *z* input points



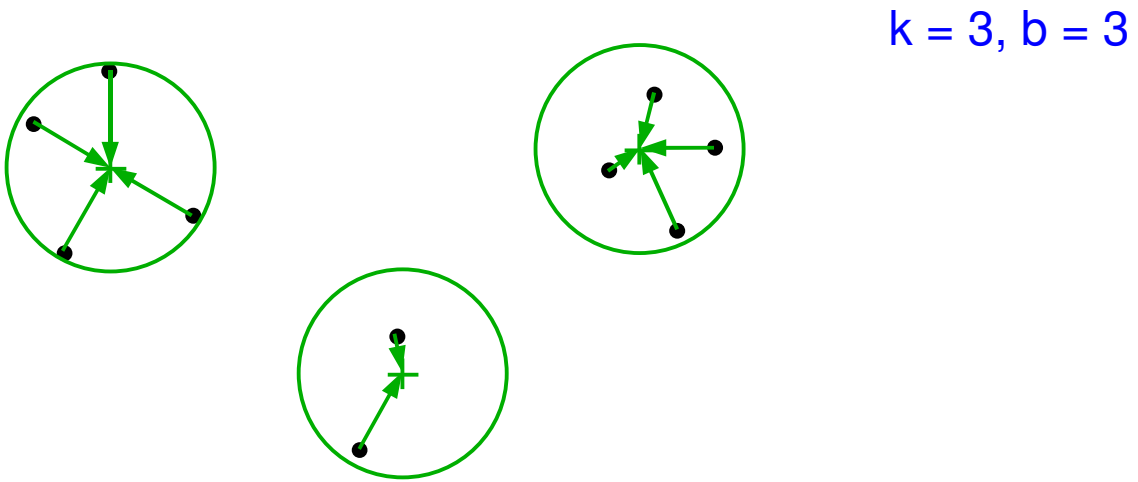
k-center clustering “with anonymity”

- Application: Publish per-cluster statistics without revealing too much about any single input point
- Each cluster gets ≥ *b* points



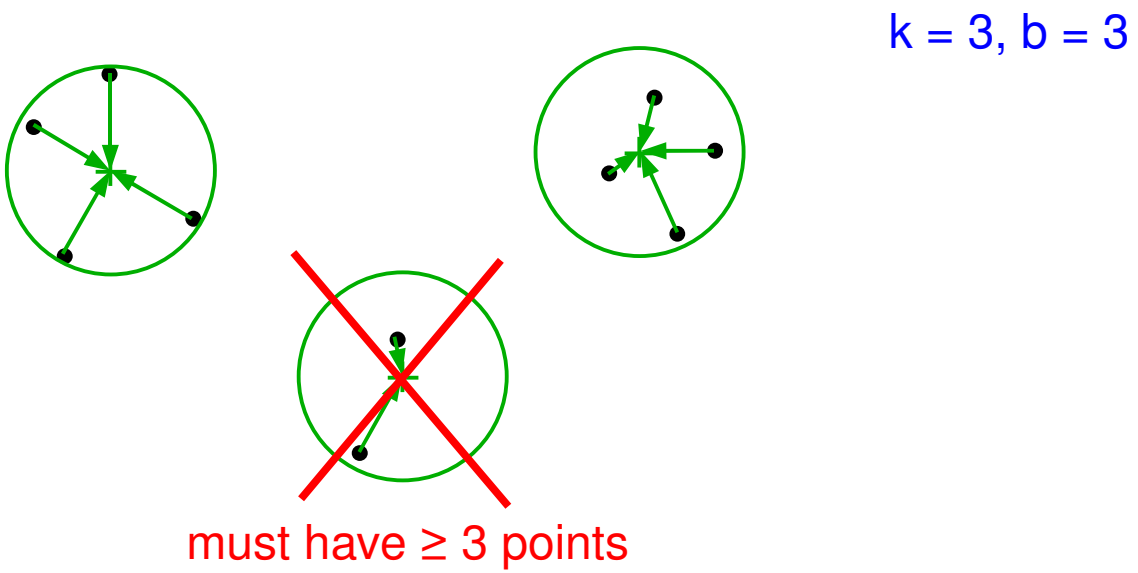
k-center clustering “with anonymity”

- Application: Publish per-cluster statistics without revealing too much about any single input point
- Each cluster gets ≥ *b* points



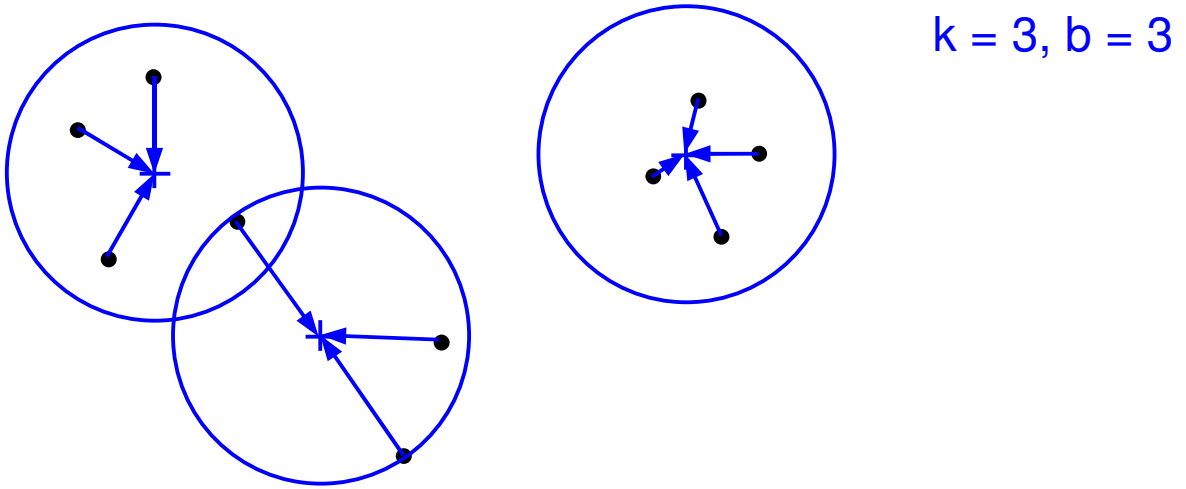
k-center clustering “with anonymity”

- Application: Publish per-cluster statistics without revealing too much about any single input point
- Each cluster gets ≥ *b* points



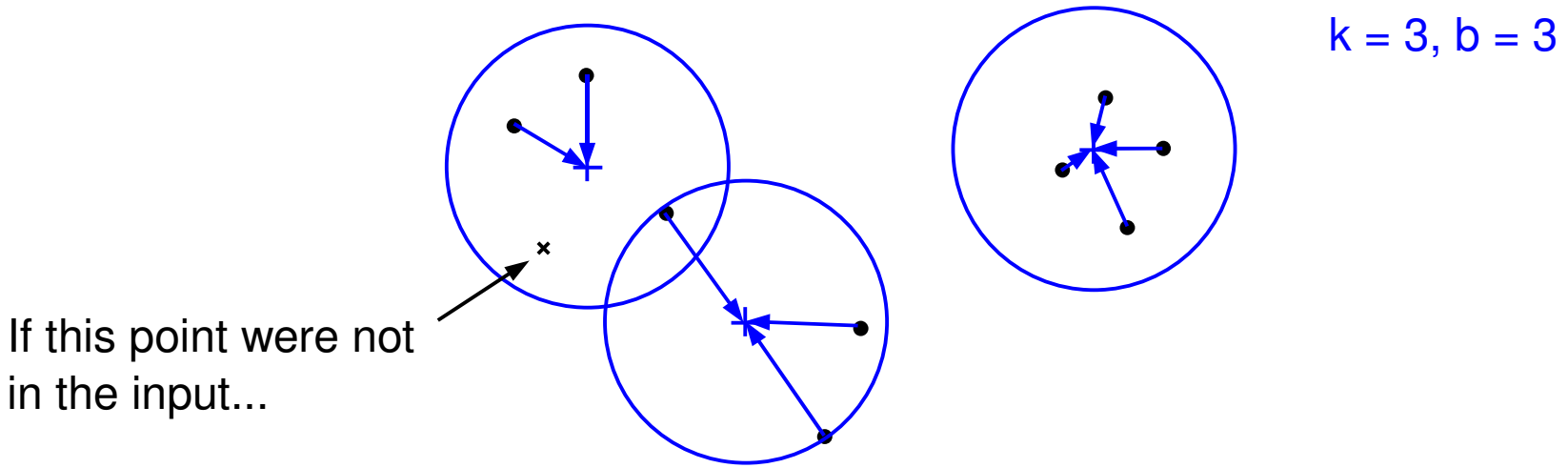
k-center clustering “with anonymity”

- Application: Publish per-cluster statistics without revealing too much about any single input point
- Each cluster gets ≥ *b* points



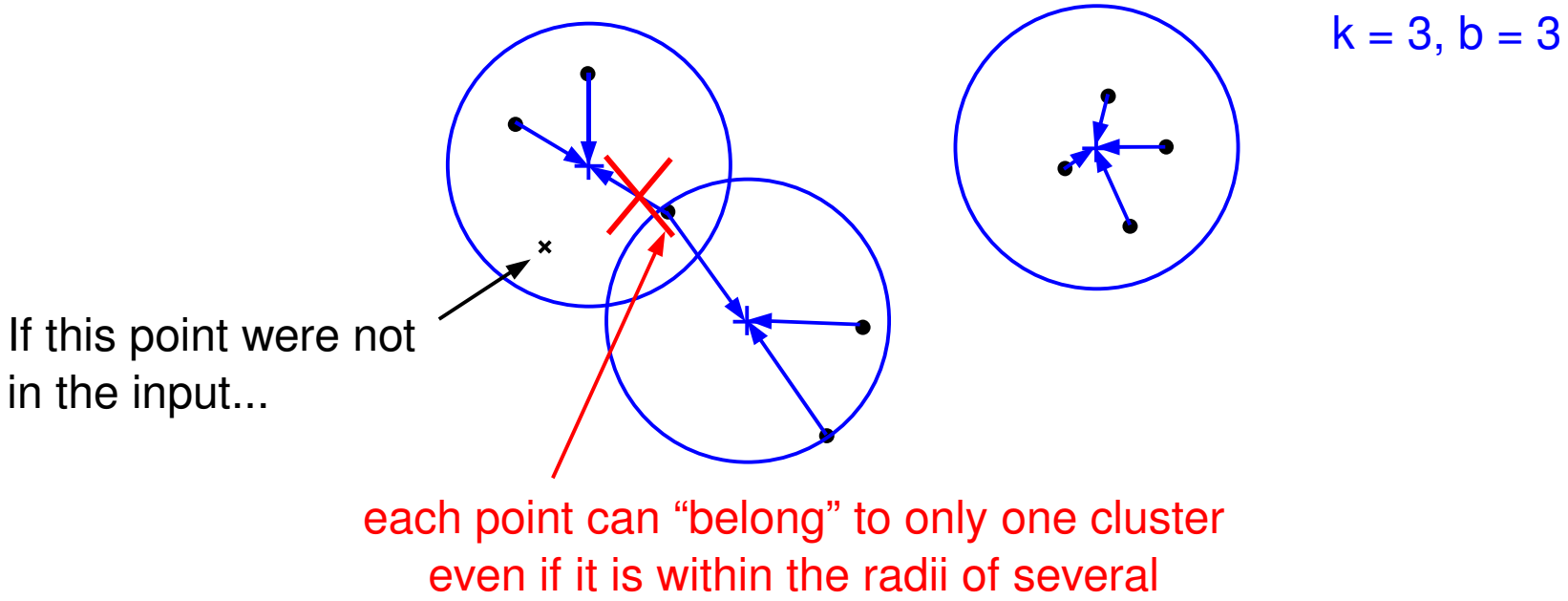
k-center clustering “with anonymity”

- Application: Publish per-cluster statistics without revealing too much about any single input point
- Each cluster gets ≥ *b* points



k-center clustering "with anonymity"

- Application: Publish per-cluster statistics without revealing too much about any single input point
- Each cluster gets ≥ *b* points



Currently known results

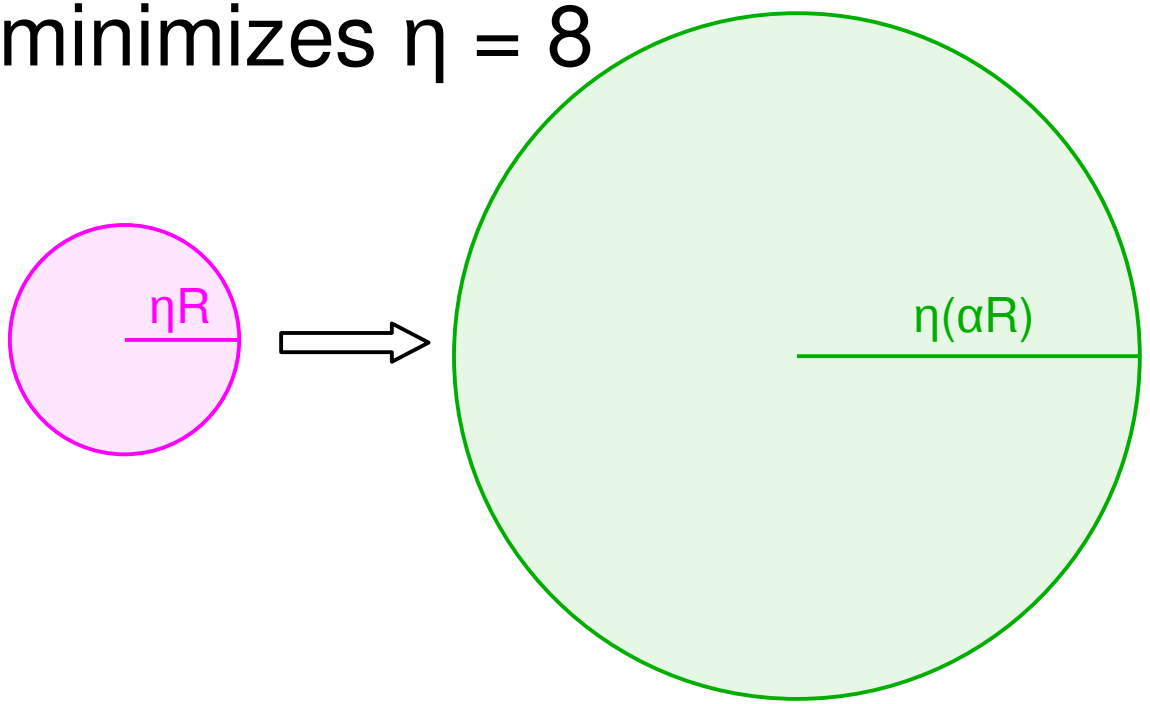
Approximation factor in...
- cluster radius
- number of outliers

Problem	Model	Algorithm	f_r	f_z	Memory
Basic *k*-center	Offline	Greedy (Hochbaum-Shmoys '85)	2		
		Farthest-point (Gonzalez '85)	2		
	Streaming	Doubling (Charikar et al. STOC '97)	8		k
		Parallelized scaling	$2+\varepsilon$		$\varepsilon^{-1}\ln(\varepsilon^{-1})k$
Outliers	Offline	Greedy (Charikar et al. SODA '01)	4	1	
	Streaming	Sampling (Charikar et al. STOC '03)	4	$1+\varepsilon$	$\varepsilon^{-2}k(n/z)$
		Scaling w/ support points	$4+\varepsilon$	1	$\varepsilon^{-1}kz$
	(With enumerable centers, all 4s become 3.)				
Anonymity	Offline	Flow (Aggarwal et al. PODS '06)	2		
	Streaming	Add scaling pass (not in this talk)	$6+\varepsilon$		$\varepsilon^{-1}\ln(\varepsilon^{-1})k + k^2$

(Our contributions)

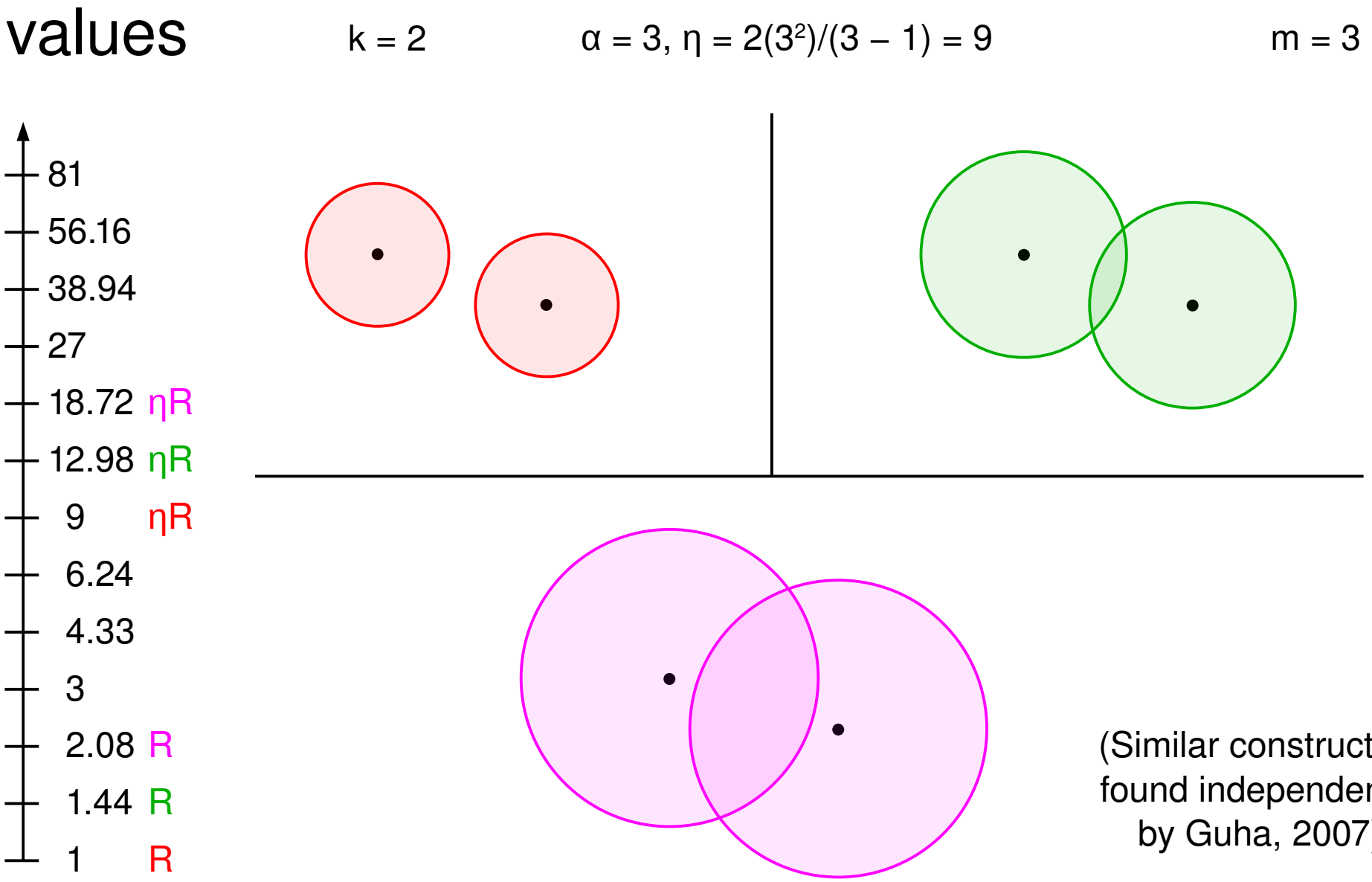
“Scaling” Algorithm

- Doubling Algorithm generalized to an arbitrary scaling factor α:
  - $R_{new} = \alpha R$ when we rule out an optimal solution $< \alpha R$
  - Centers separated by $2\alpha R$
  - Every point within $\eta R = [2\alpha^2/(\alpha - 1)]R$ of a stored center
    - Merging: $2\alpha R_{new} + \eta R = 2\alpha[1 + 1/(\alpha - 1)]R_{new} = \eta R_{new}$
- $\alpha = 2$ minimizes $\eta = 8$



Parallelized Scaling Algorithm

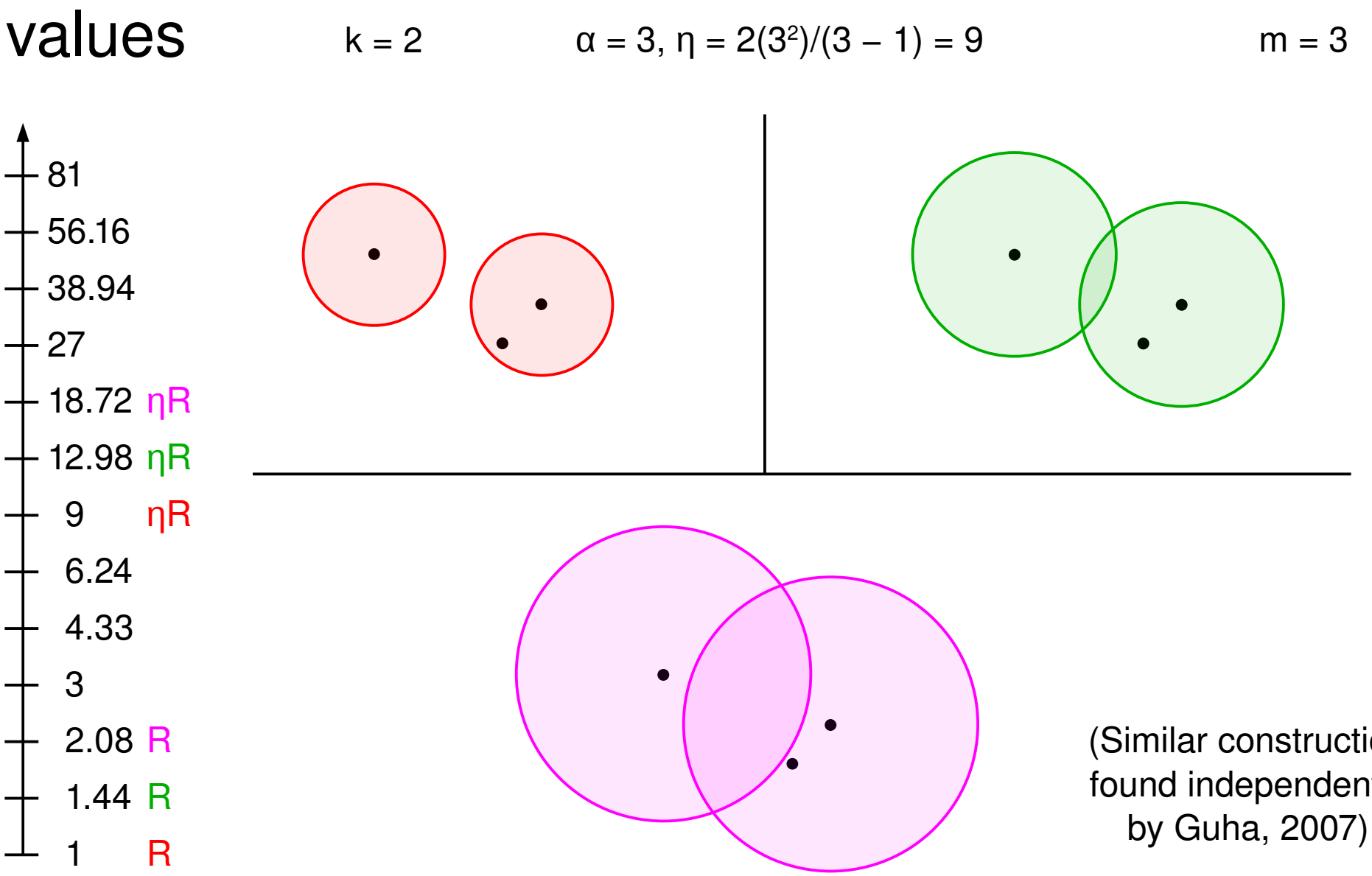
- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

Parallelized Scaling Algorithm

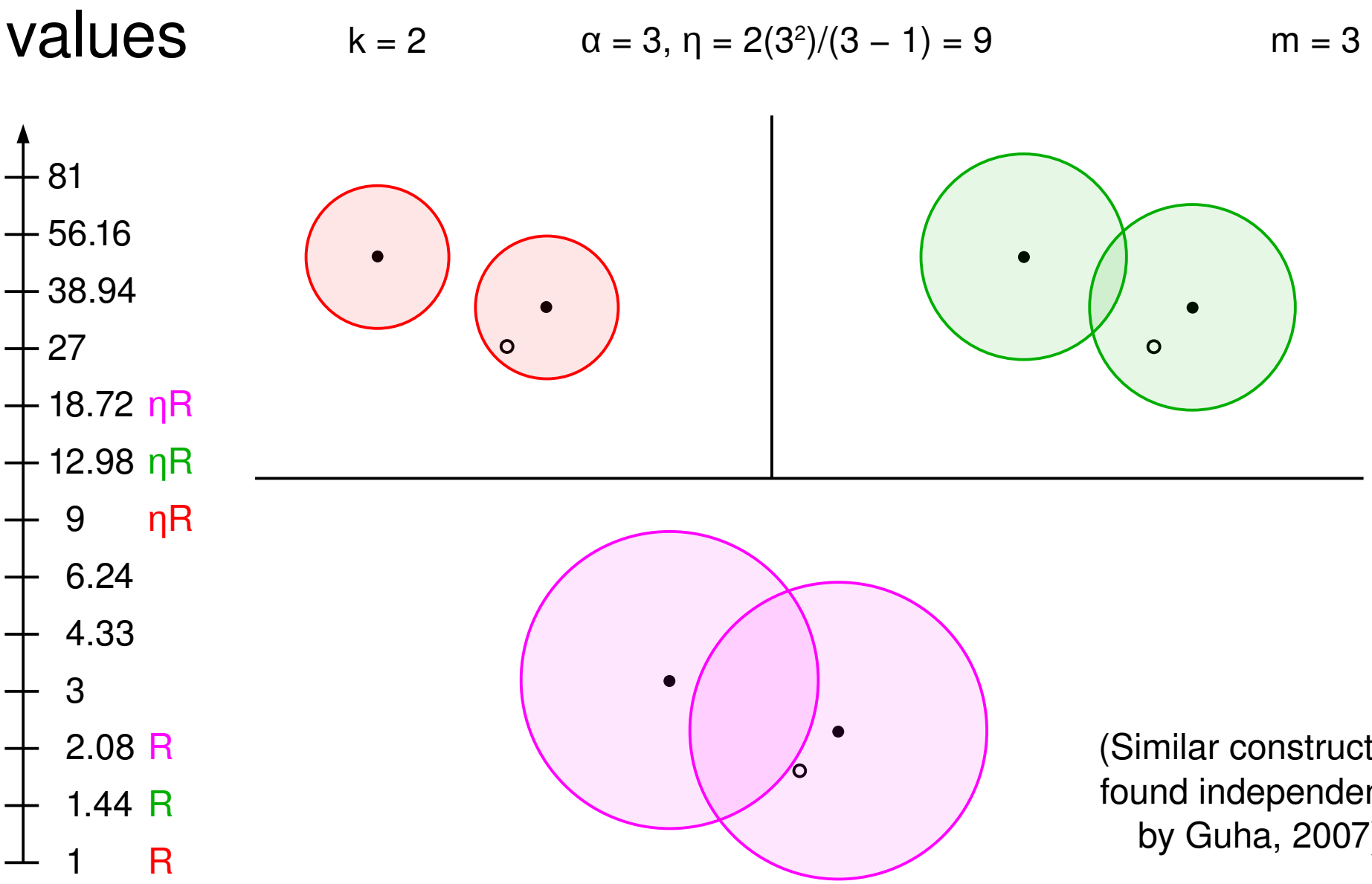
- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

Parallelized Scaling Algorithm

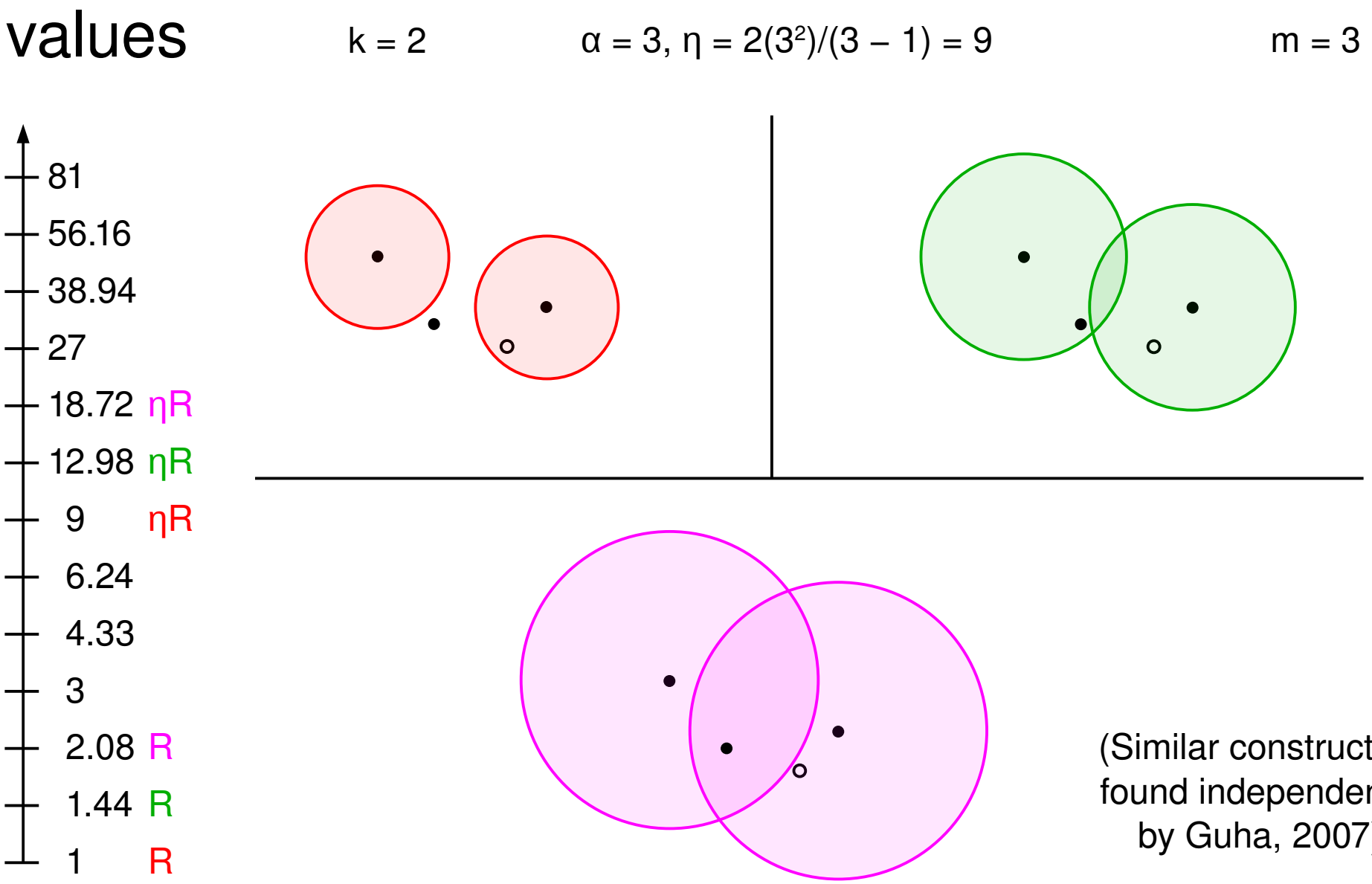
- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

Parallelized Scaling Algorithm

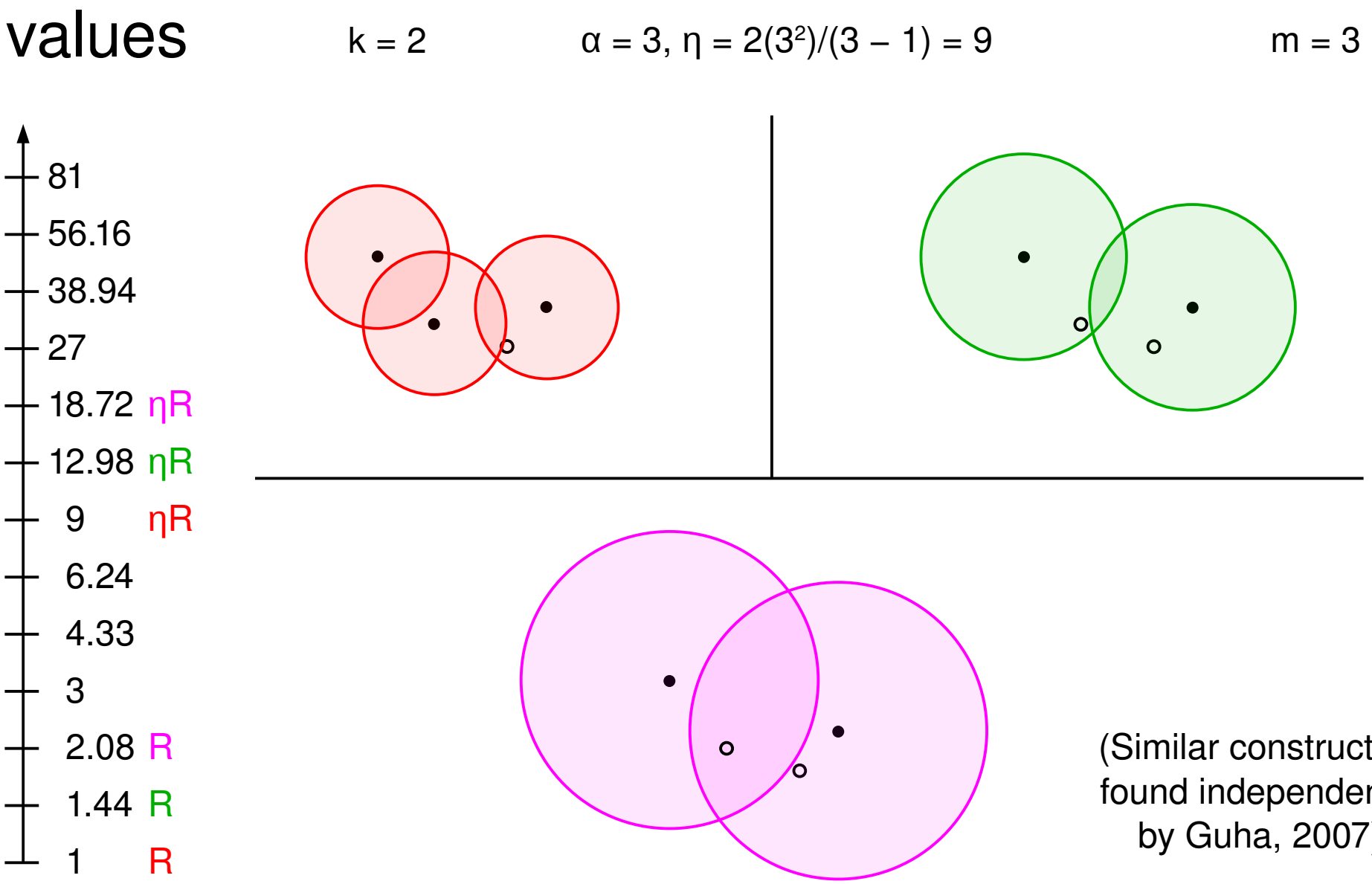
- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

Parallelized Scaling Algorithm

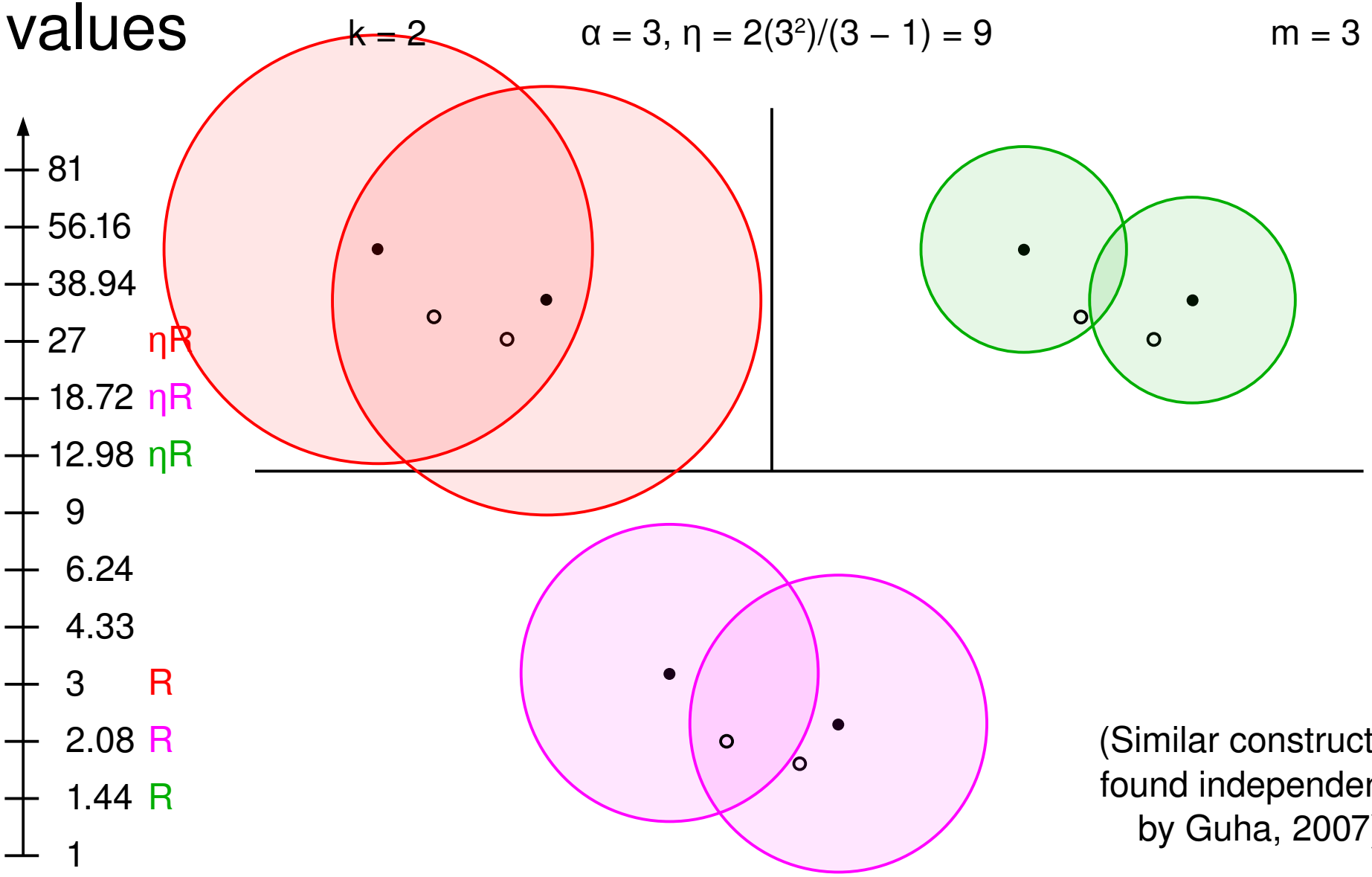
- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

Parallelized Scaling Algorithm

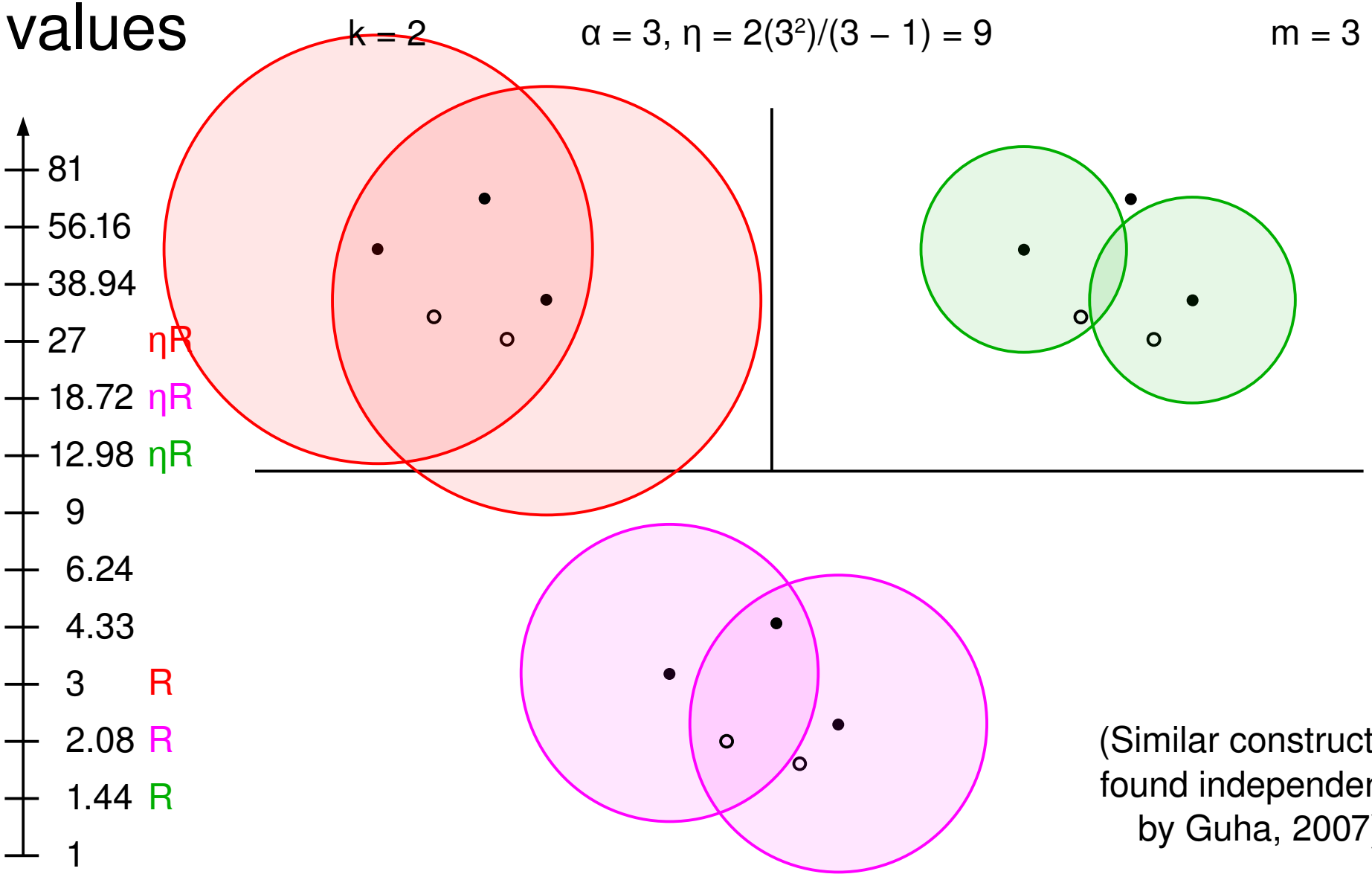
- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

Parallelized Scaling Algorithm

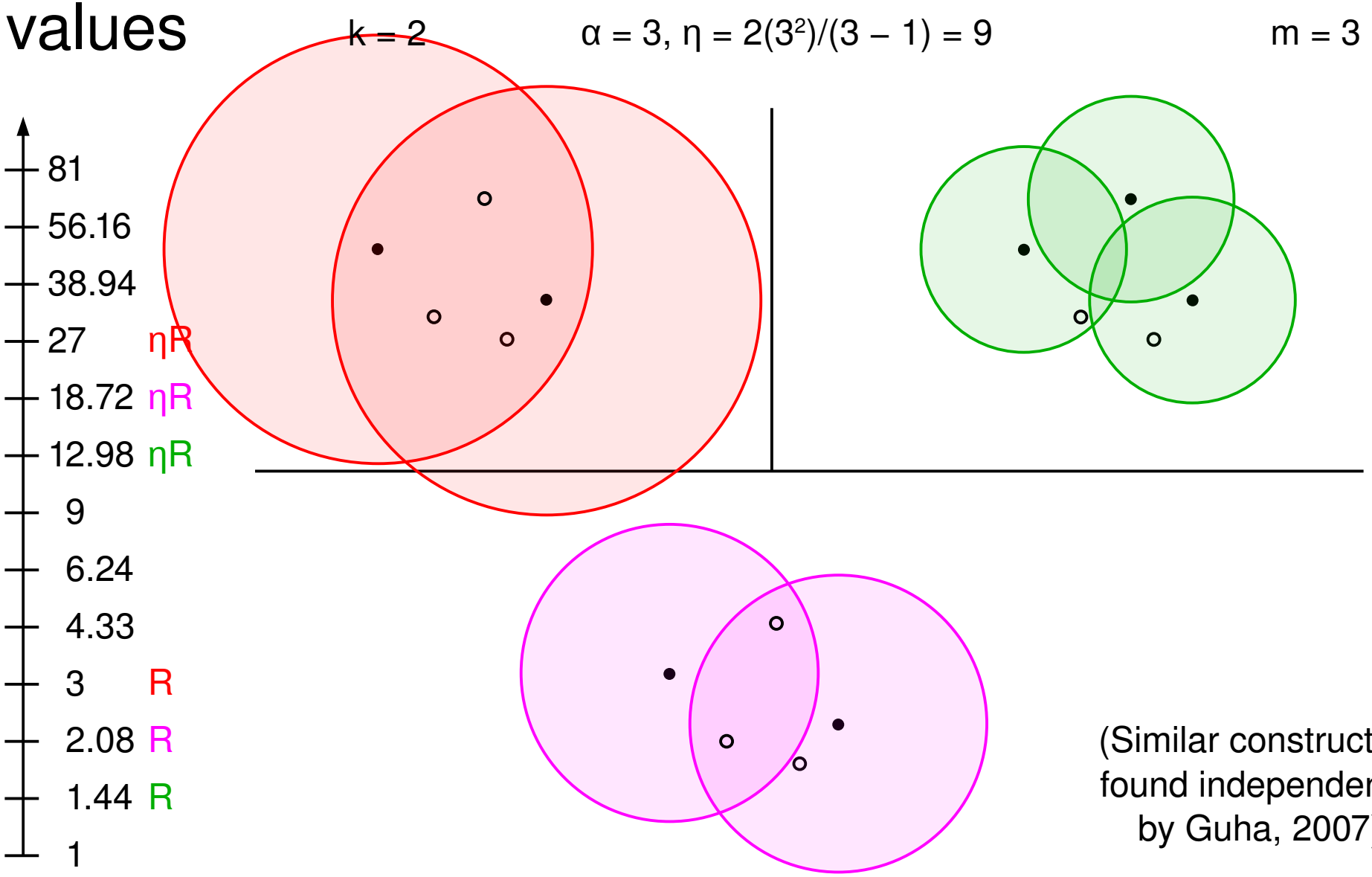
- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

Parallelized Scaling Algorithm

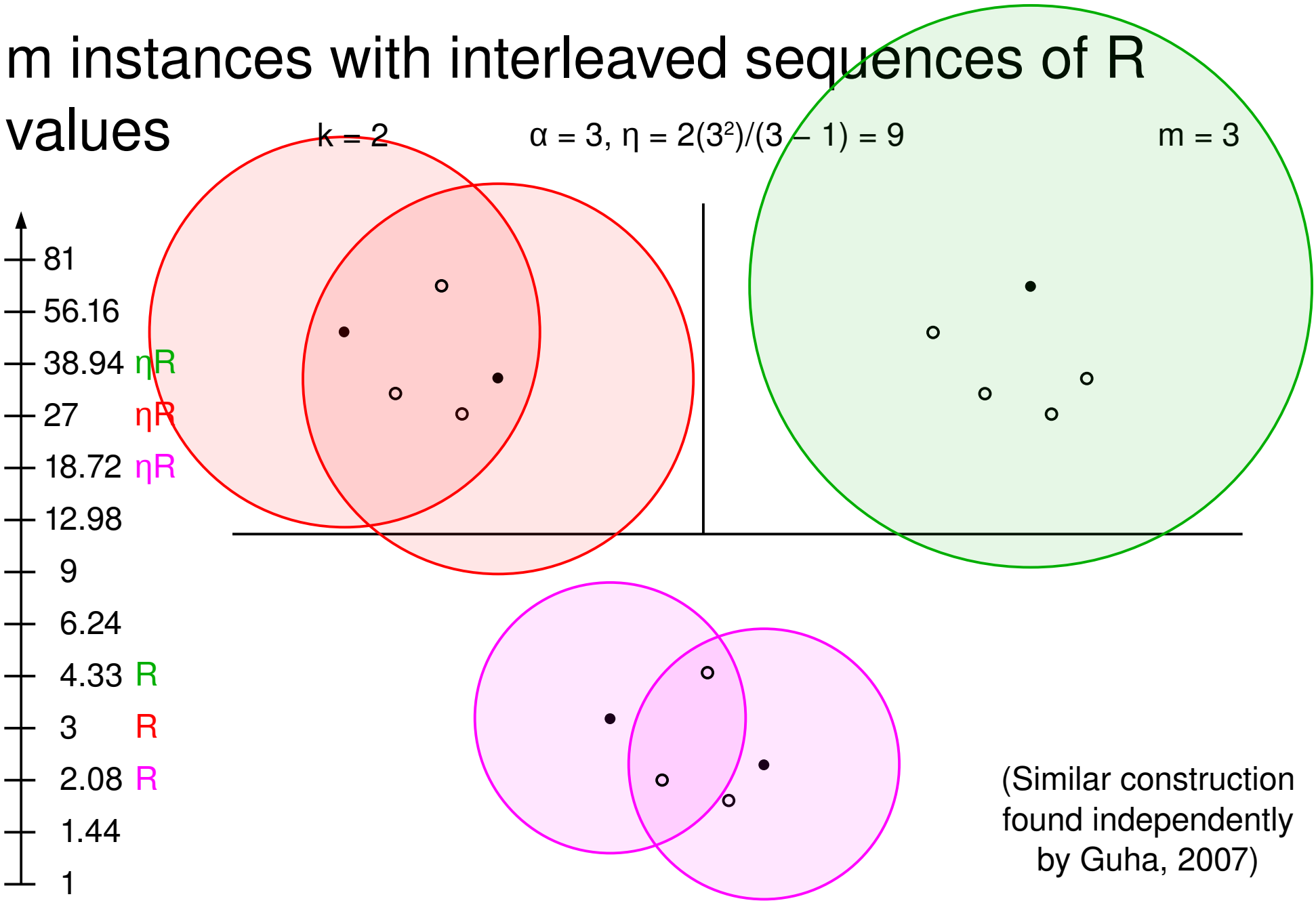
- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

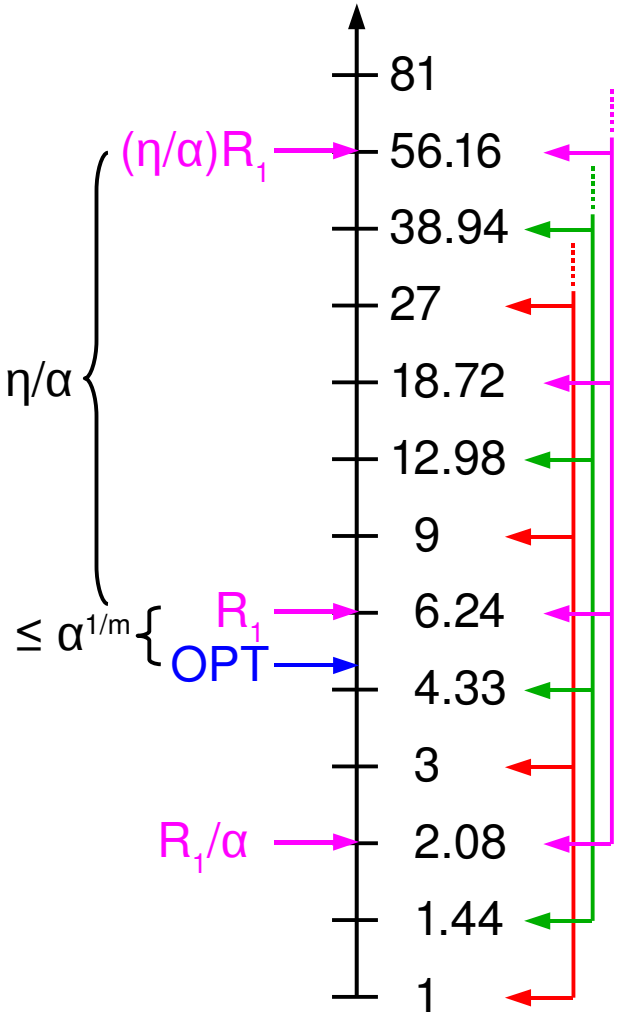
Parallelized Scaling Algorithm

- m instances with interleaved sequences of R values



(Similar construction found independently by Guha, 2007)

Benefit of parallelization



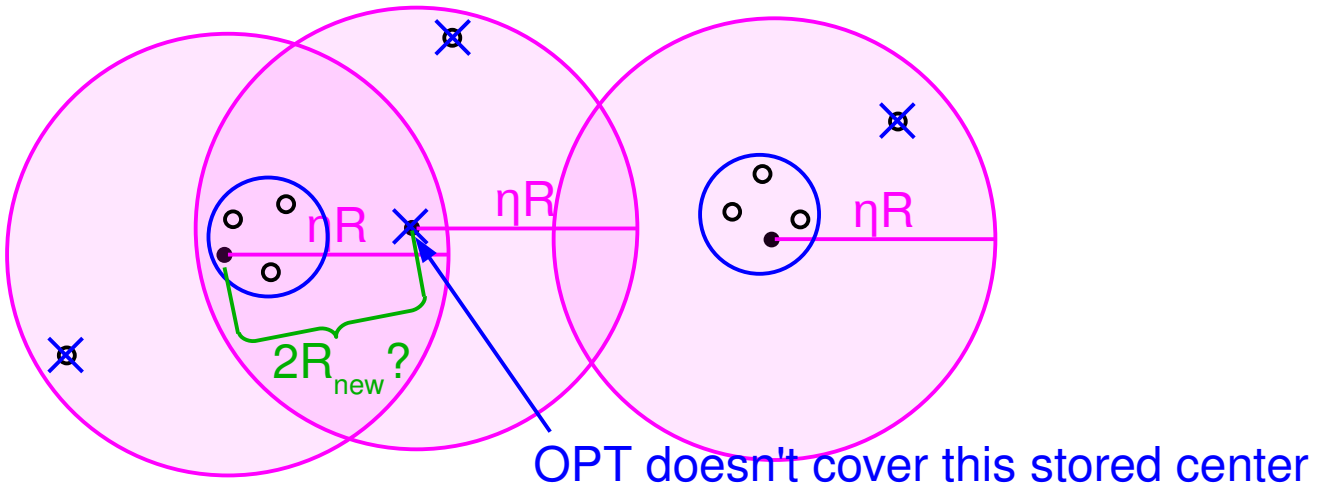
α = 3, m = 3, η = 9:
4.33-approximation

- Take best solution produced by any instance.
- Instance whose R sequence has R_1 will give a $2[1 + 1/(\alpha - 1)]\alpha^{1/m}$-approximation.
- Good approximation: make α large and m even larger!
- Obtain a $(2 + \varepsilon)$-approximation with $\alpha = O(\varepsilon^{-1})$ and $m = O(\varepsilon^{-1}\ln(\varepsilon^{-1}))$

Streaming *k*-center with outliers

- Commonalities with the Scaling Algorithm:
  - Keep $s \le k$ stored centers and a lower bound R on OPT
  - Solution contains clusters of radius ηR at stored centers. Drop input points arriving in those clusters.
  - Set $R_{new} = \alpha R$ when solution becomes infeasible
- Problem: OPT no longer guaranteed to cover all our stored centers ⇒ having > *k* well-separated stored centers no longer means we can raise R.

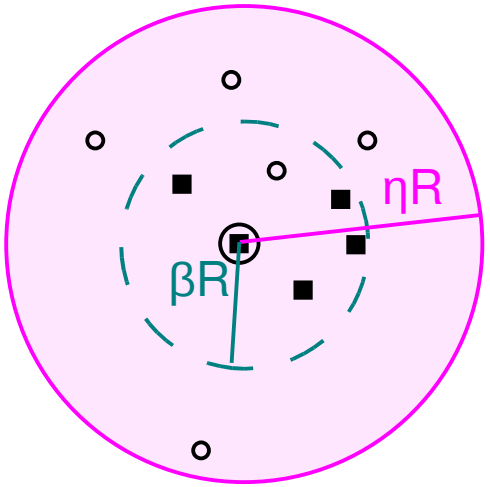
k = 2, z = 4



Coping with outliers

- Solution: Don't store a center until it has $z + 1$ "support points" within βR (for some parameter β). OPT can't mark all the support points as outliers, so it must cover at least one of them.
- Keep each input point as a "free point" until it can be stored as a center or dropped.

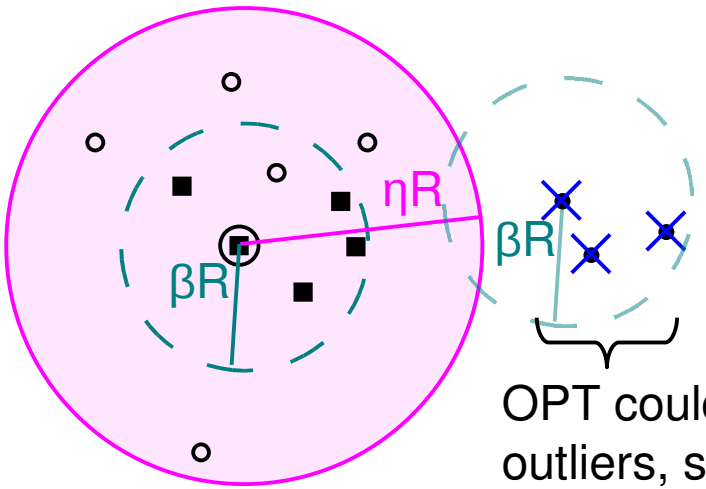
k = 3, z = 4,
s = 1



Coping with outliers

- Solution: Don't store a center until it has $z + 1$ "support points" within βR (for some parameter β). OPT can't mark all the support points as outliers, so it must cover at least one of them.
- Keep each input point as a "free point" until it can be stored as a center or dropped.

k = 3, z = 4,
s = 1

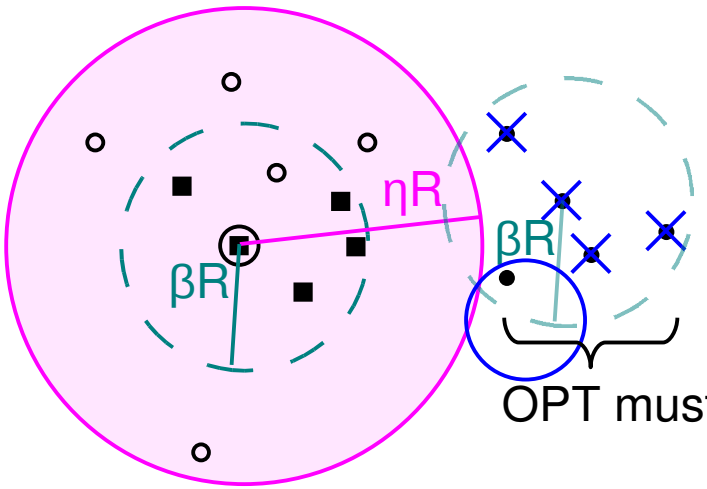


OPT could mark all these as outliers, so don't store yet.

Coping with outliers

- Solution: Don't store a center until it has $z + 1$ "support points" within βR (for some parameter β). OPT can't mark all the support points as outliers, so it must cover at least one of them.
- Keep each input point as a "free point" until it can be stored as a center or dropped.

k = 3, z = 4,
s = 1

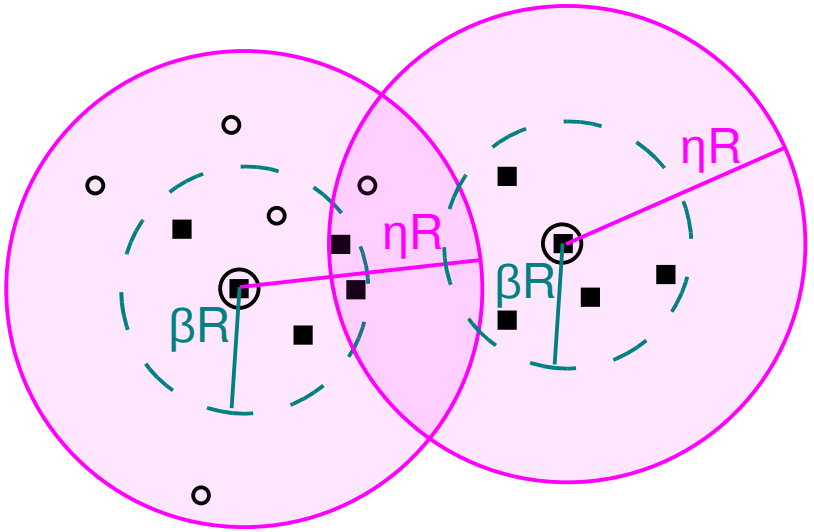


OPT must cover one of these $z + 1$ points.

Coping with outliers

- Solution: Don't store a center until it has $z + 1$ "support points" within βR (for some parameter β). OPT can't mark all the support points as outliers, so it must cover at least one of them.
- Keep each input point as a "free point" until it can be stored as a center or dropped.

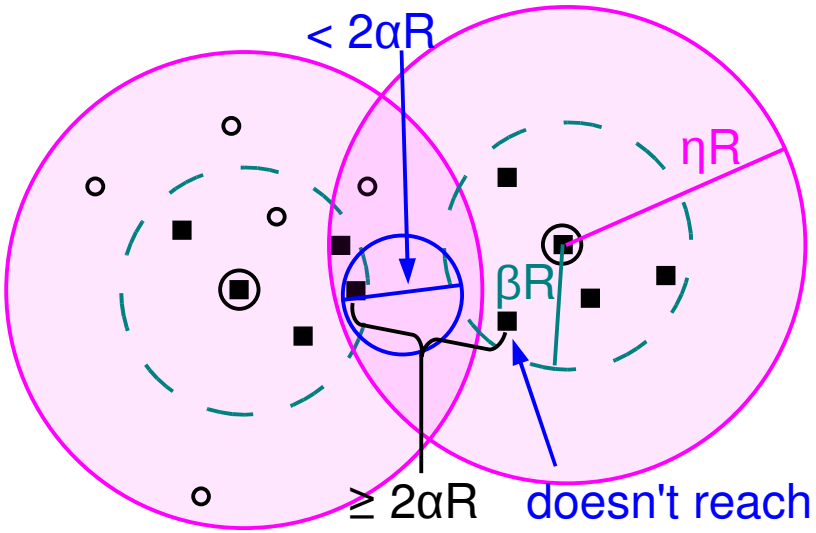
k = 3, z = 4,
s = 2



Separation of support, free points

- Assuming $\eta R \geq \beta R + 2\alpha R$, support points of different centers will be separated by $2\alpha R$, so an optimal cluster of radius $< \alpha R$ can satisfy at most one stored center.

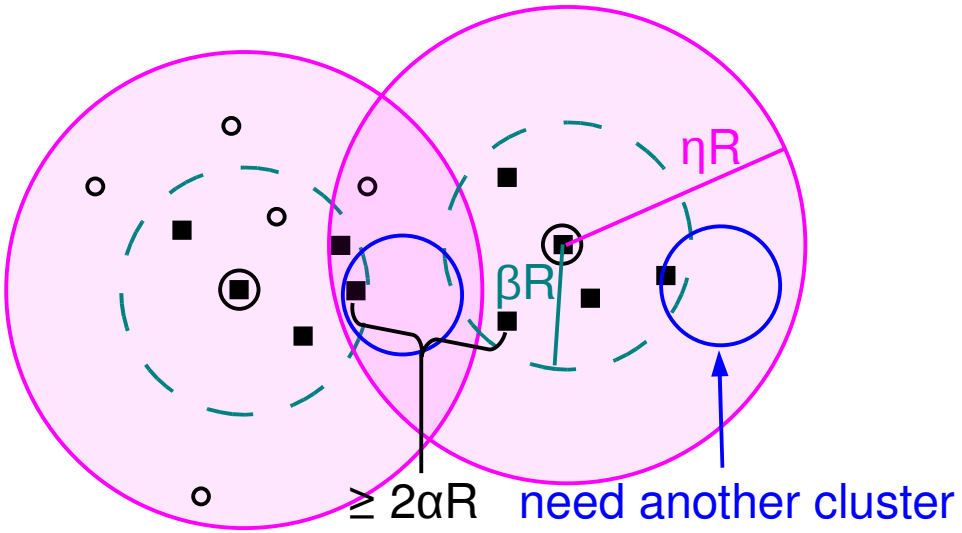
k = 3, z = 4,
s = 2



Separation of support, free points

- Assuming $\eta R \geq \beta R + 2\alpha R$, support points of different centers will be separated by $2\alpha R$, so an optimal cluster of radius $< \alpha R$ can satisfy at most one stored center.
- $\Longrightarrow$ OPT must set aside s clusters to satisfy stored centers. (If $s > k$, raise R.)

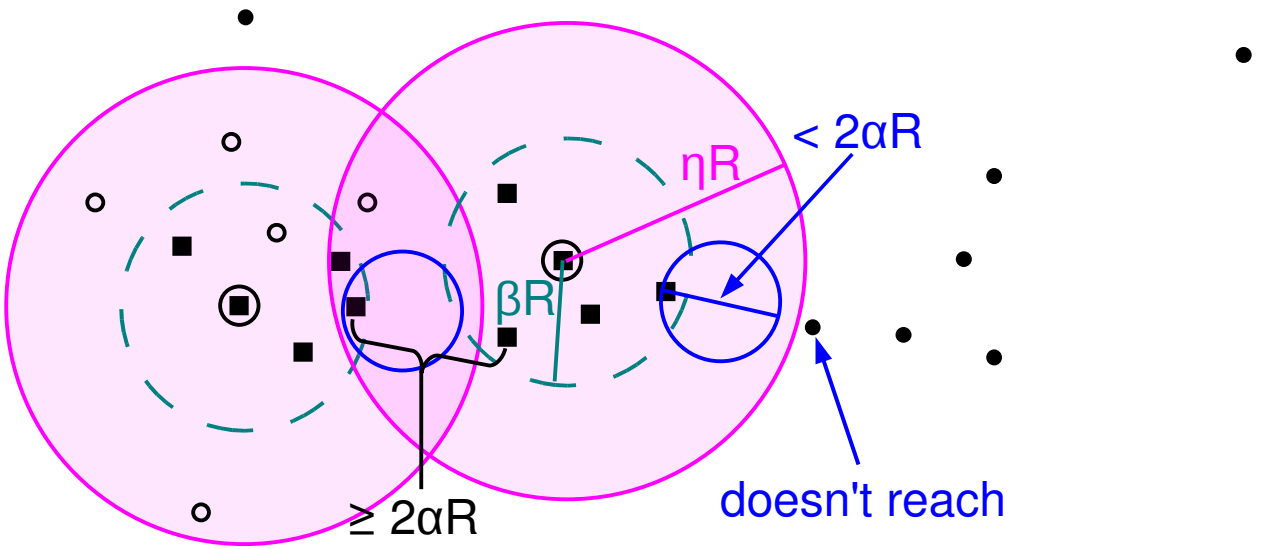
k = 3, z = 4,
s = 2



Separation of support, free points

- Assuming $\eta R \geq \beta R + 2\alpha R$, support points of different centers will be separated by $2\alpha R$, so an optimal cluster of radius $< \alpha R$ can satisfy at most one stored center.
- $\Longrightarrow$ OPT must set aside s clusters to satisfy stored centers. (If $s > k$, raise R.)
  - These clusters can't additionally cover free points.

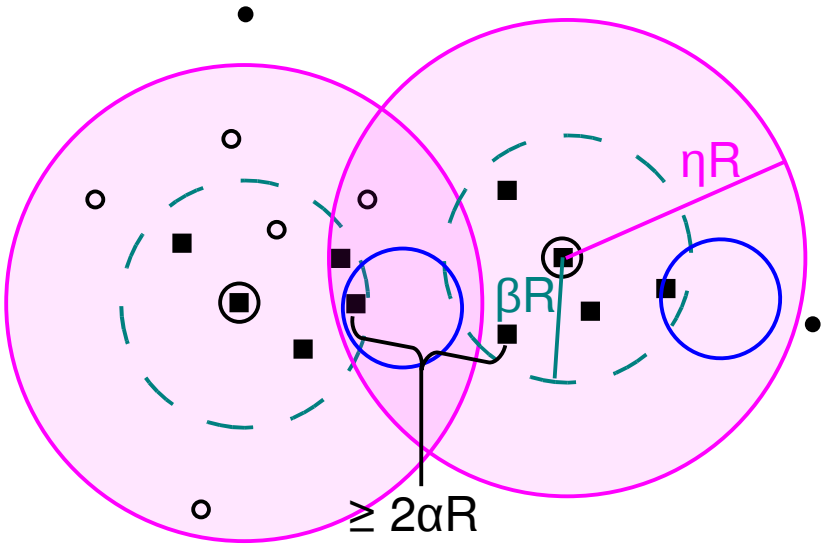
k = 3, z = 4,
s = 2

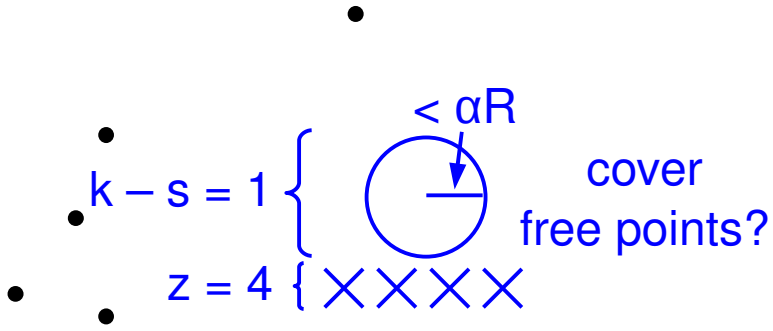


Covering free points

- Can OPT cover the free points with the other k – s clusters of radius < αR and z outliers?

k = 3, z = 4,
s = 2

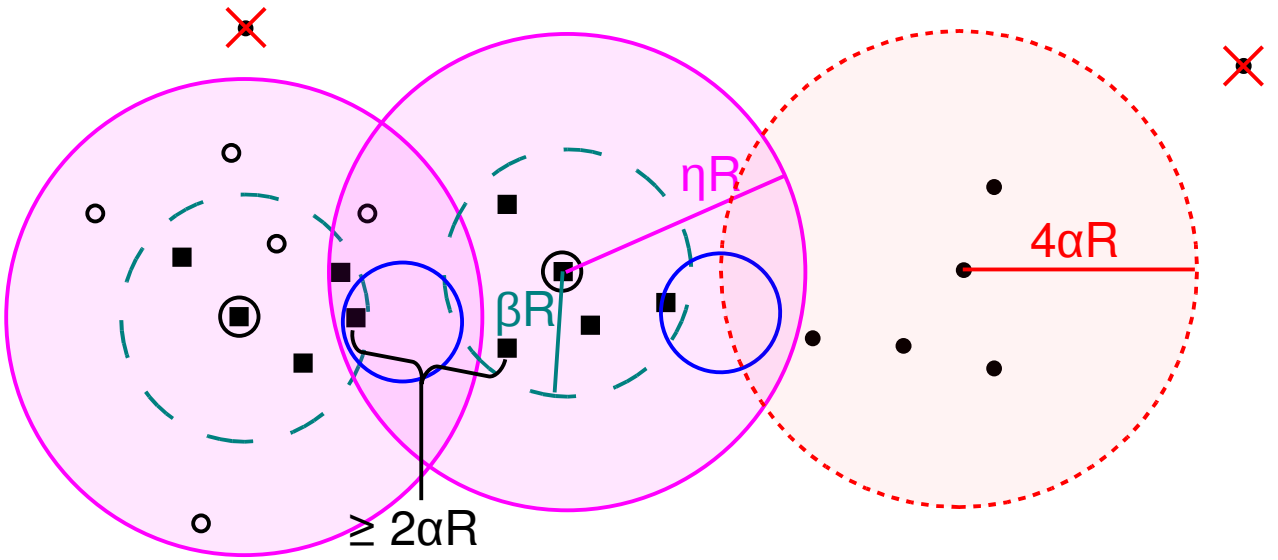




Covering free points

- Can OPT cover the free points with the other k – s clusters of radius < αR and z outliers?
- Try the 4-approximation offline algorithm with guess αR.
  - Success: Have a feasible solution at ηR, assuming ηR ≥ 4αR.

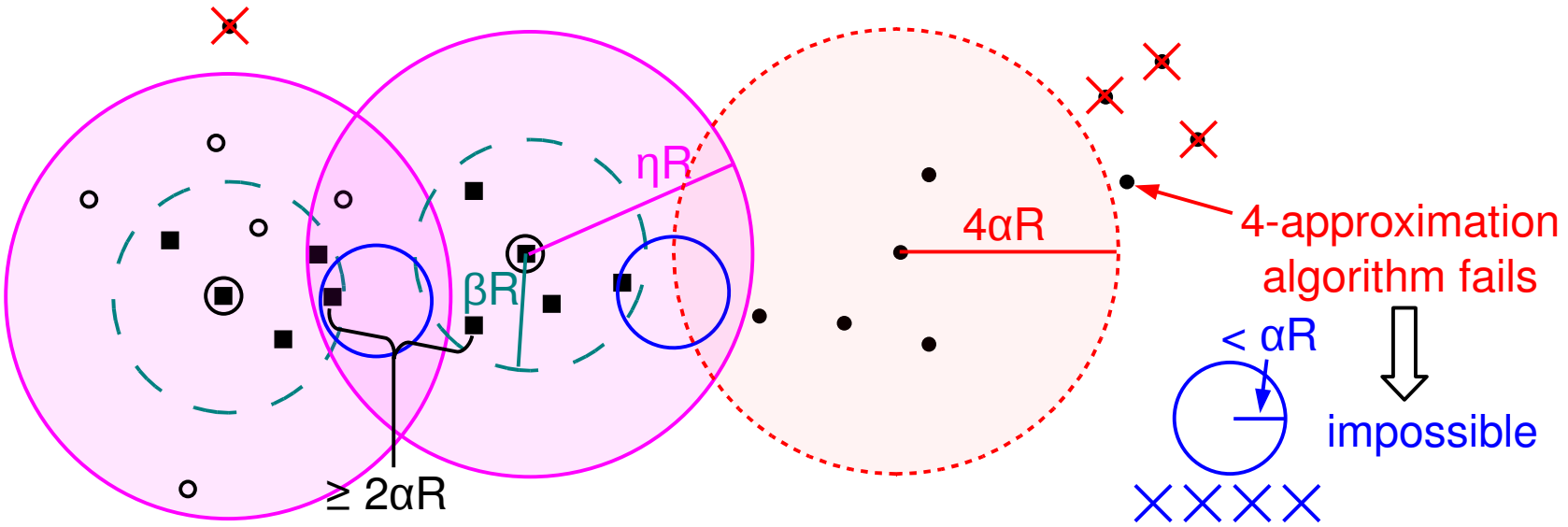
k = 3, z = 4,
s = 2



Covering free points

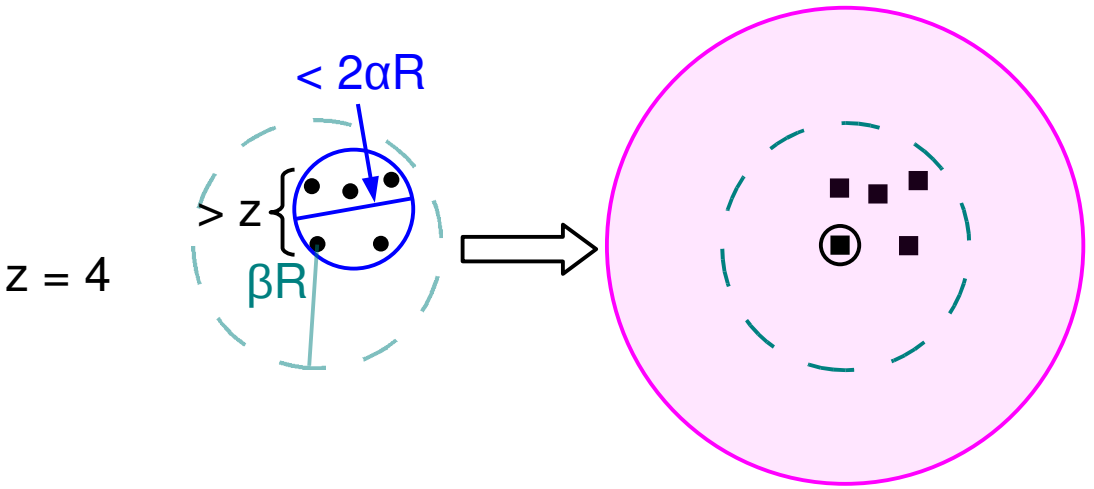
- Can OPT cover the free points with the other k – s clusters of radius < αR and z outliers?
- Try the 4-approximation offline algorithm with guess αR.
  - Success: Have a feasible solution at ηR, assuming ηR ≥ 4αR.
  - Failure: OPT can't do it ⇒ raise R.

k = 3, z = 4,
s = 2



Controlling memory usage

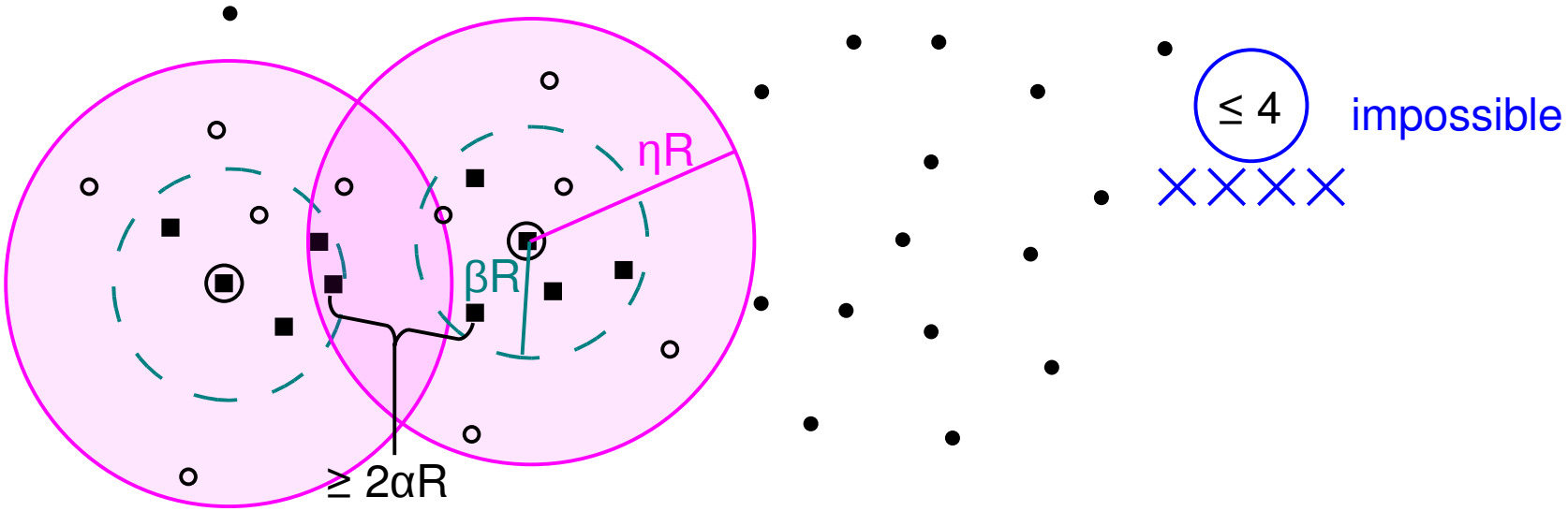
- Must limit number of free points for O(kz) memory bound.
- Assuming $\beta R \geq 2\alpha R$, any optimal cluster of radius $< \alpha R$ that covers more than z free points will have one of its points stored as a center.



Controlling memory usage (2)

- An optimal solution of radius < αR must cover the free points with k – s clusters and z outliers.
- After center-storing, each of those clusters covers ≤ z free points.
- ⟹ If there are > (k – s)z + z free points, it's impossible, so we raise R to maintain the memory bound.

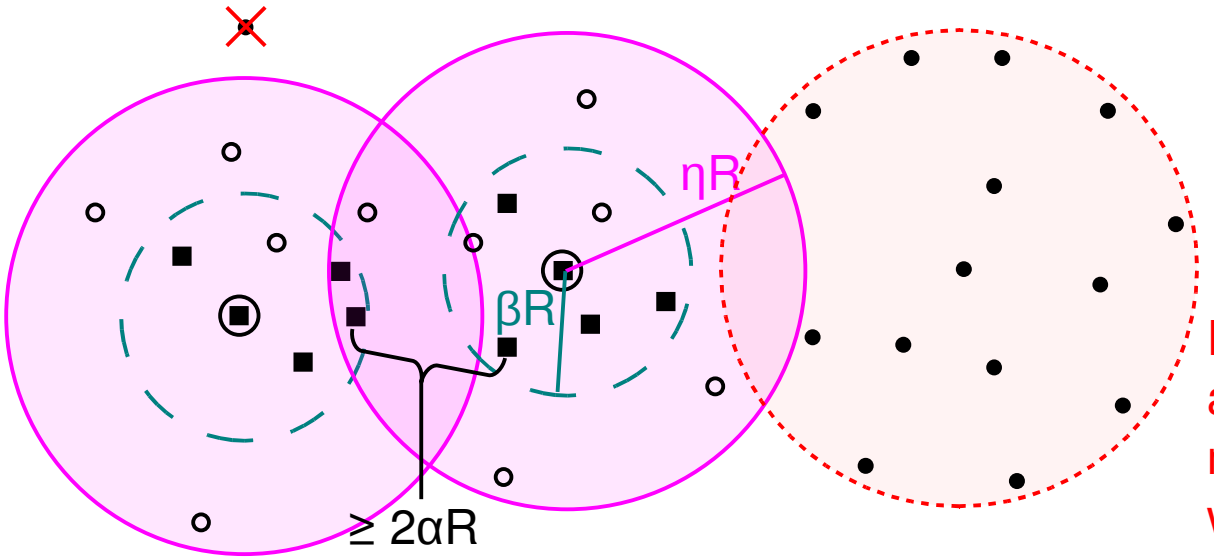
k = 3, z = 4,
s = 2

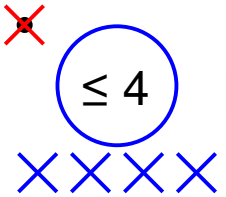


Controlling memory usage (2)

- An optimal solution of radius < αR must cover the free points with k – s clusters and z outliers.
- After center-storing, each of those clusters covers ≤ z free points.
- ⟹ If there are > (k – s)z + z free points, it's impossible, so we raise R to maintain the memory bound.

k = 3, z = 4,
s = 2



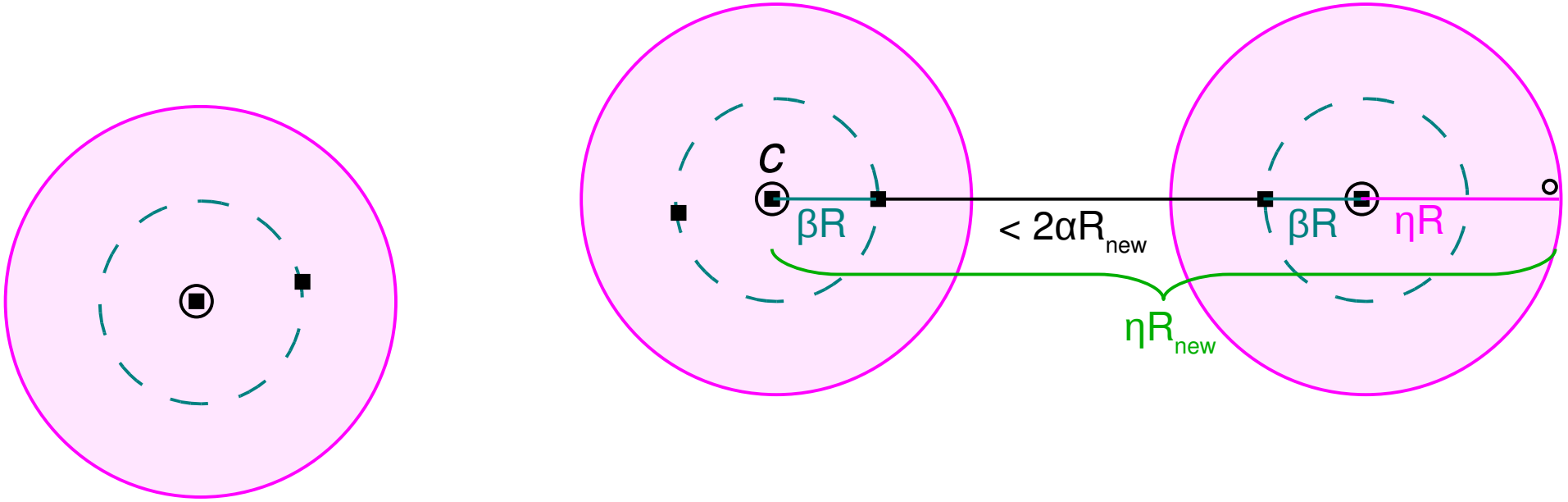


impossible

Even though we have a feasible solution, we must raise R to keep within O(kz) memory.

Merging step

- Must restore separation of 2αR between support points of different centers after raising R.
- Greedy merging pass as in Scaling Algorithm:
  - Each center *c* subsumes any centers with support points closer than $2\alpha R_{new}$ to a support point of *c*.
- Assume $\beta R + 2\alpha(\alpha R) + \beta R + \eta R \leq \eta(\alpha R)$.



Streaming *k*-center w/ outliers: result

$\eta R \geq \beta R + 2\alpha R$
$\eta R \geq 4\alpha R$
$\beta R \geq 2\alpha R$
$\beta R + 2\alpha(\alpha R) + \beta R + \eta R \leq \eta(\alpha R)$

⇩

$\alpha = 4, \beta = 8, \eta = 16$

⇩

m-instance parallelization:
$(4^{1 + 1/m})$-approximation

⇩

$(4 + \varepsilon)$-approximation in radius,
no more than z outliers,
with $O(\varepsilon^{-1}kz)$ memory

Future work

- Outliers (clustering can miss up to z points):
  - Reduce memory usage to $O(\varepsilon^{-1}(k + z))$
    - We think we have a $(14 + \varepsilon)$-approximation (not fully verified)
    - Would match $\Omega(k + z)$ lower bound for deterministic algos
  - Memory requirement when neither z nor n/z is small?

Algorithm	$\mathbf{f_r}$	$\mathbf{f_z}$	**Memory**
Sampling (Charikar et al. STOC '03)	4	1+ε	$\varepsilon^{-2}k(n/z)$
Scaling w/ support points	4+ε	1	$\varepsilon^{-1}kz$

- Anonymity (each cluster gets ≥ b points):
  - Reduce approximation factor from $6 + \varepsilon$ to $2 + \varepsilon$?
- Streaming algorithm for outliers + anonymity
  - Offline 4-approximation is known
- Multi-pass algorithms