*

The Sorting Hat

- or -

Using Clusters to Analyze, Arrange, or Compress Data

ISC1057

Janet Peterson and John Burkardt

Computational Thinking

Fall Semester 2016

At Hogwarts Academy, the sorting hat instantly determines whether a student should assigned to the house of Gryffindor, Hufflepuff, Ravenclaw or Slytherin.

Clustering is the process of assigning individuals to groups.

Sometimes, as at Hogwarts, we already know how many groups there are, and we already know a rule that tells us which student goes to which house (...or at least the sorting hat can tell us this!).

But in other cases, the only thing we have, to begin with, is a large collection of individuals, or data, and it's up to us to decide a sensible procedure for organizing them into groups, and deciding how many groups there should be.

We do simple clustering in language, when we talk about $animal,$ $vegetable,$ or $mineral$ or when we divide creatures according to whether they live in $land,$ $sea,$ or $air.$

In many cases, how we cluster a set of data depends on what we are interested in, and often there are natural ways to do the clustering.

A deck of cards can be clustered by suit, by number, by red/black, by court cards (K,Q,J) versus non-court cards, by odd/even.

A coin collection can be clustered by coin size, value, metallic content, year of issue, country of origin, shape (some coins are square, octagonal, or have a central hole).

Hoofed animals $(ungulates)$ can be classified by the number of toes. Horses and rhinoceroses are clustered in the $odd\text{-}toed$ ungulate group, while camels, pigs, giraffes, deer, antelope, sheep, cows are clustered in the $even\text{-}toed$ ungulate group.

The clusterings we've seen so far come about because of natural human or cultural categories, or, in the case of ungulates, because it was noticed that dividing ungulates into even- and odd-toed clusters created two distinct groups whose individual members had much in common.

Computationally, the clustering problem becomes more difficult. Suppose we have collected a great mass of data:

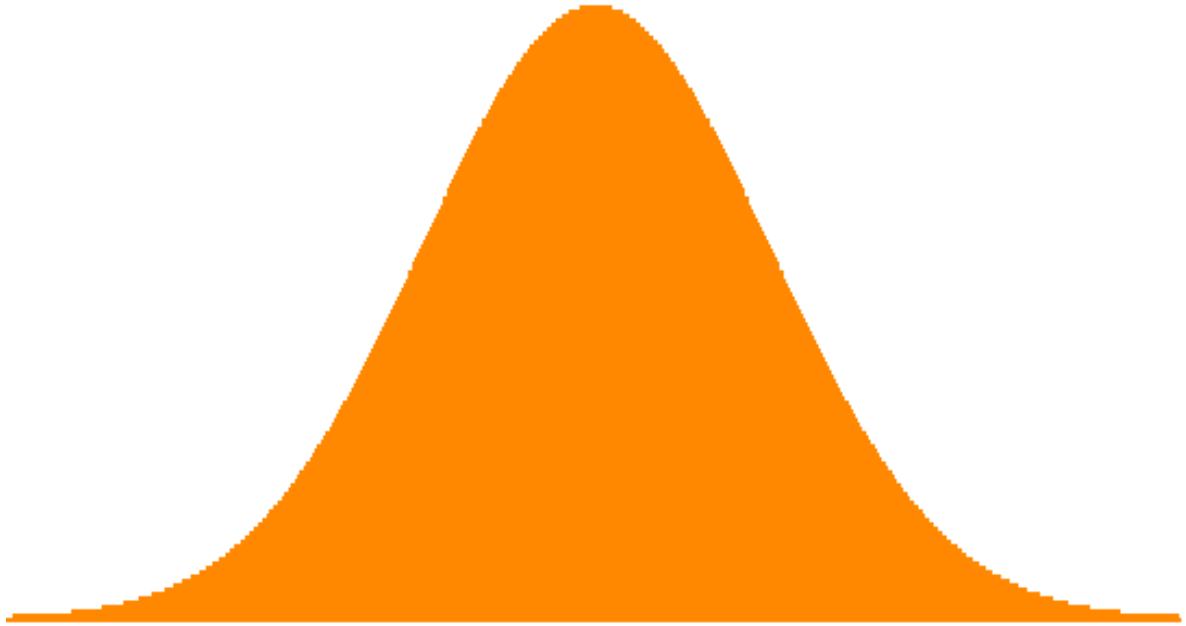
- the location and price of 500 houses for sale,
- the weight, height, and age of 2,000 babies,
- the salary, age, and major of 10,000 former FSU students.

A computer doesn't have any built-in understanding of how these things might be classified, so it is necessary for us to think of how we do this job ourselves.

If the data we're dealing with is numeric, such as weights, salaries, ages or speeds, then we can use the Average Rule:

Numeric data can be represented by its average.

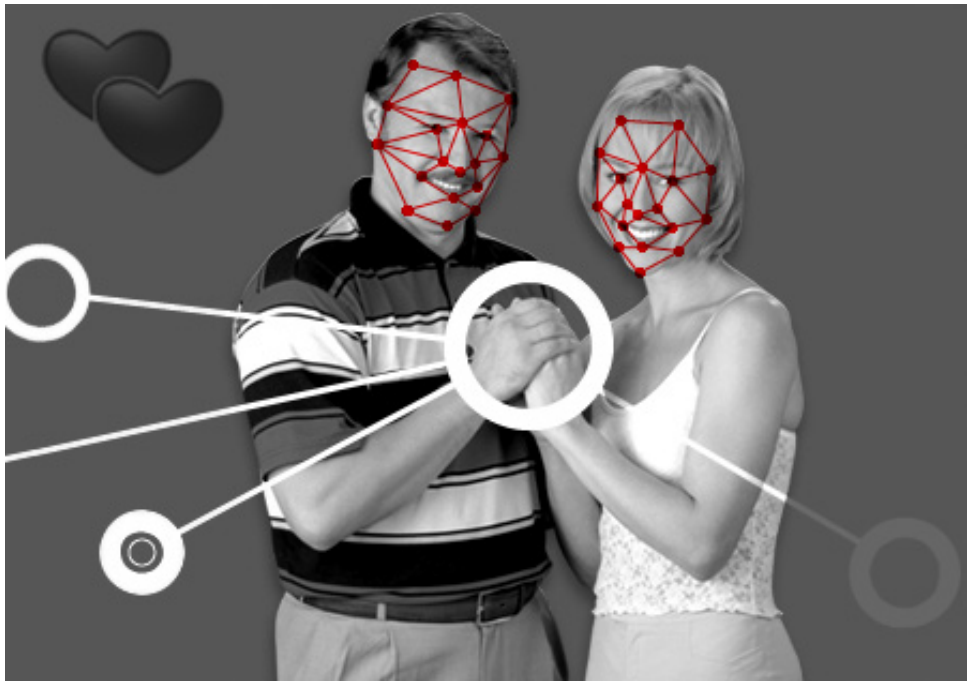
Averages are useful when the quantity has a bell-curve behavior, concentrated near a central value, tapering off at either extreme.



The average is like a single cluster. Sometimes a better model uses brackets. For example, we could cluster numeric test scores into the lower, middle, and upper thirds:

In a famous experiment called Face-to-Facebook, described at http://www.face-to-facebook.net/index.php, the artists Paolo Cirio and Alessandro Ludovico made fun of Facebook.

They collected 1,000,000 profile pictures from FaceBook, in order to create a fake dating service.



Their first task was to determine a gender for each picture. To do this, they first identified a certain number of pictures as "male" or "female" and then let the computer try to guess the gender of new pictures. They corrected the computer when it was wrong, and it adjusted its algorithm, until it was agreeing with the artists. Then they let it assign a gender to all 1,000,000 pictures.

In any dating service, people talk about themselves, and about their ideal partner. Using just the assigned gender, and the profile picture, the artists had to come up with similar characteristics.

They used facial recognition software to analyze the expressions on the profile pictures. Looking at a small sample, they decided that those faces could be divided into one of six categories: *easy-going*, *climber*, *funny*, *mild*, *sly*, *smug*.

They wrote another program that classified new pictures into one of the six categories. Again, they had to "correct" the program when they felt it made a bad classification. Once the program's choice seemed reasonable, they gave it all 1,000,000 pictures to process, and the fake dating service was created.

WHICH DO YOU LIKE? find yourself a lovely face:

Select: Any Nationality | Any Character | Any Gender

Search optional Keyword:

Search your partner

Stephanie Costa Val Dabien Brazilian Easy Going Woman

Click here to contact Stephanie Costa Val Dabien to arrange a date with her.

How do you rate Stephanie Costa Val Dabien?

As Judge her character and how cool is she: Give a vote

Write a comment about this person:

To avoid spamming Stephanie Costa Val Dabien, please confirm her gender: Confirm gender and Post the comment

487 Similar profiles to Stephanie Costa Val Dabien who are Brazilian Easy Going Woman . Some of those:


Tatiana França


Viviane Carvalho Murat


Isabela Pizarro Gualtieri


Marcela Cunha


Lilly Braga


Monica Behring Palhares


Nina Korff Muller


Denizete Ramalho


Marcela Affonso

Best friends of Stephanie Costa Val Dabien:

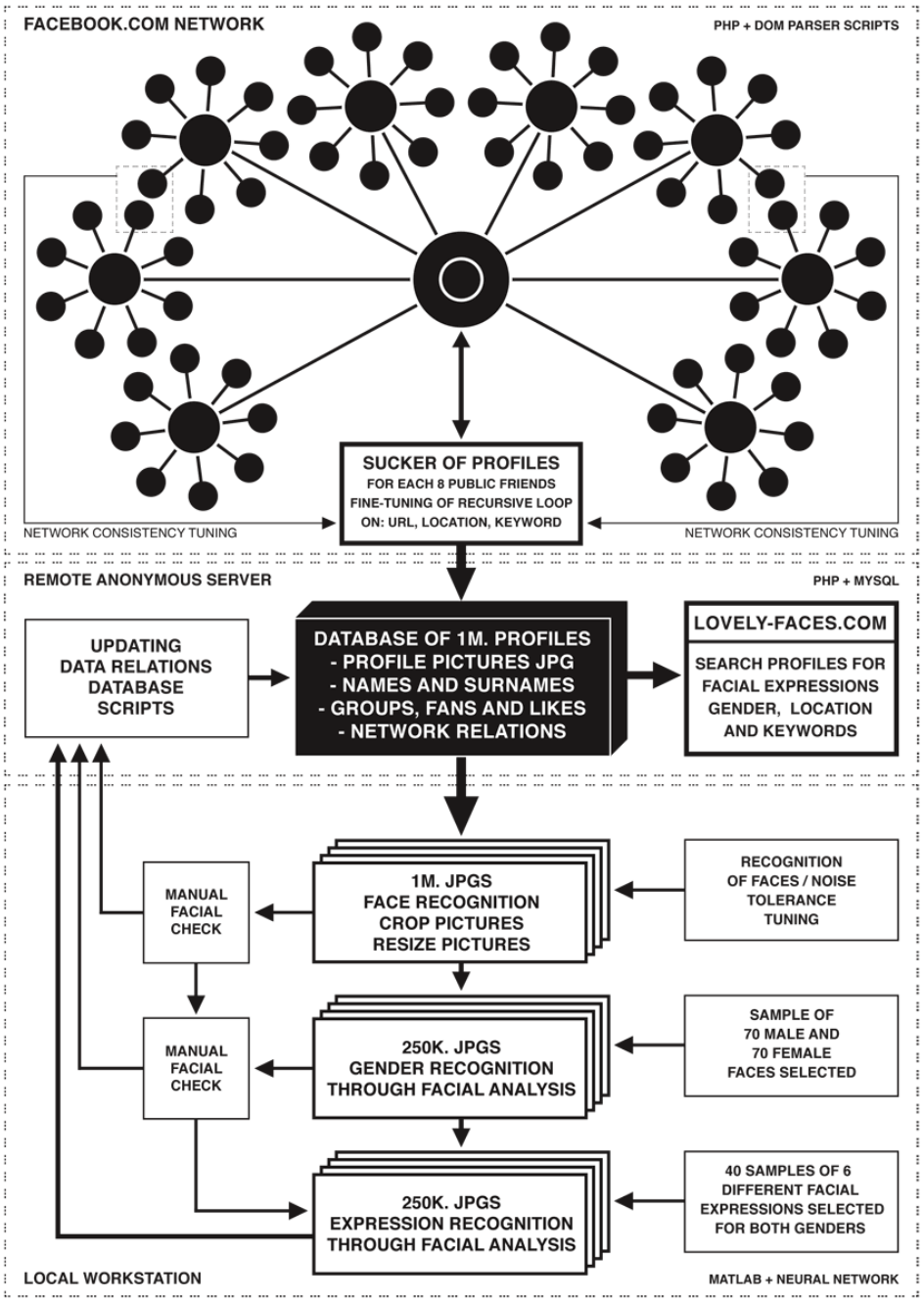

Fellipe Mattos Brazilian Mild Man

Julia Gontijo

Chiara Besenzoni Gottardi Brazilian

Juliana Farah Brazilian

Kika Horta Brazilian


FACEBOOK.COM NETWORK
PHP + DOM PARSER SCRIPTS
SUCKER OF PROFILES
FOR EACH 8 PUBLIC FRIENDS
FINE-TUNING OF RECURSIVE LOOP
ON: URL, LOCATION, KEYWORD
NETWORK CONSISTENCY TUNING
NETWORK CONSISTENCY TUNING
REMOTE ANONYMOUS SERVER
PHP + MYSQL
UPDATING
DATA RELATIONS
DATABASE
SCRIPTS
DATABASE OF 1M. PROFILES
- PROFILE PICTURES JPG
- NAMES AND SURNAMES
- GROUPS, FANS AND LIKES
- NETWORK RELATIONS
LOVELY-FACES.COM
SEARCH PROFILES FOR
FACIAL EXPRESSIONS
GENDER, LOCATION
AND KEYWORDS
MANUAL
FACIAL
CHECK
1M. JPGS
FACE RECOGNITION
CROP PICTURES
RESIZE PICTURES
RECOGNITION
OF FACES / NOISE
TOLERANCE
TUNING
MANUAL
FACIAL
CHECK
250K. JPGS
GENDER RECOGNITION
THROUGH FACIAL ANALYSIS
SAMPLE OF
70 MALE AND
70 FEMALE
FACES SELECTED
250K. JPGS
EXPRESSION RECOGNITION
THROUGH FACIAL ANALYSIS
40 SAMPLES OF 6
DIFFERENT FACIAL
EXPRESSIONS SELECTED
FOR BOTH GENDERS
LOCAL WORKSTATION
MATLAB + NEURAL NETWORK

Here is a short video that summarizes the experiment:

Video: **f2f.mov**

This experiment shows several unusual examples of clustering.

The objects being clustered were Facebook profile photos.

The first clustering sorted the photos into male and female clusters, but this classification was something the computer "learned", by being given some starting examples, and then being corrected when it couldn't do a new classification correctly.

The second clustering involved very vague categories that were observed in a sample of the photos, and again the computer had to learn how to match these categories.

Of course, the whole purpose of the dating service was then to "match" a male and female profile photo of similar types.

It's natural to group together objects that we think are similar in some way.

If a computer is asked do this kind of grouping, then it's necessary to come up with some way of allowing the computer to measure the similarity of any two objects. Researchers came up with the idea of a distance function, which is something like a road atlas.

A road atlas can tell you that Dallas is 966 miles from Chicago, and a distance function can tell a computer that an apple is 3 "units" from an orange, but 7 units from a banana.

Maybe these units of distance are based on the shape of each fruit, for instance, but the actual details don't matter. What's important is that the distance function is a way for the computer to confidently determine how close any two objects are.

A distance function provides several ways to analyze data.

Suppose we are given a set of objects and told that they are all related, and we are asked to come up with a sort of family tree that illustrates which objects are closely or only distantly related.

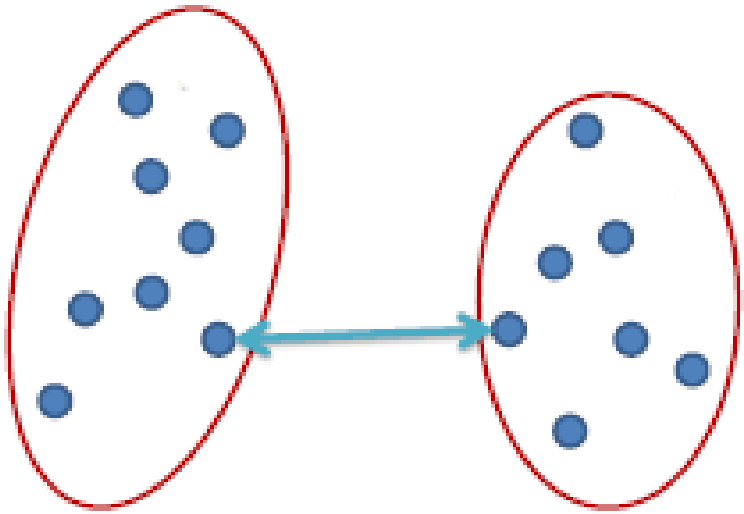
If we have a distance function, then a simple approach is called The Single Linkage Rule:

1. Make every object a separate cluster.
2. Find the two closest clusters and merge them into one.
3. Keep doing this until you only have one cluster.

To find the distance between two clusters, we consider every possible selection of one item from either cluster, and take the closest pair.

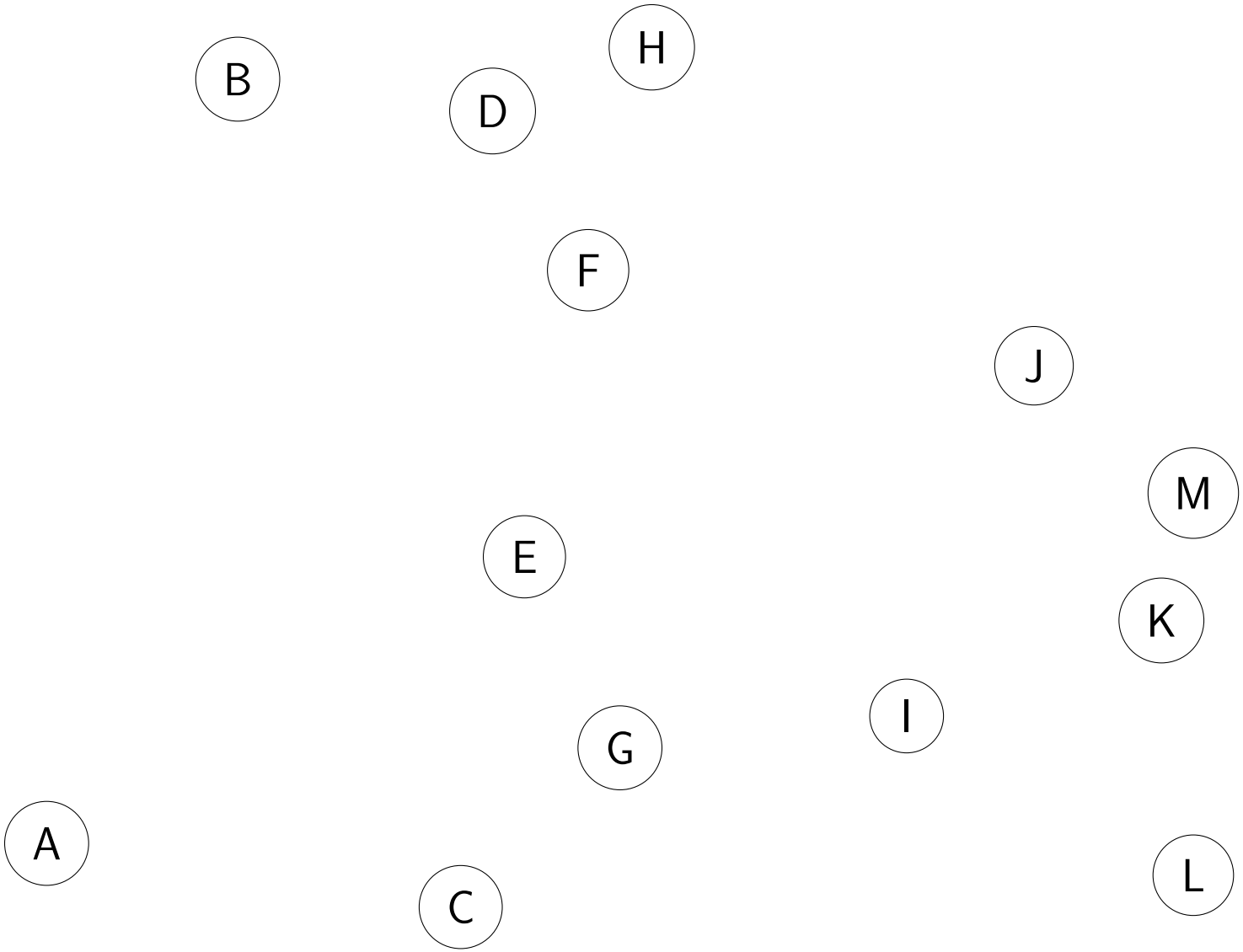
On the first step of the single linkage rule, each cluster is really a single object. So we know how to find the distance between any two clusters: just look up the distance between the two objects.

But after clusters start having multiple objects, how can we measure the distance? For the single linkage rule, we simply find the closest pair of objects and take that as the cluster distance.



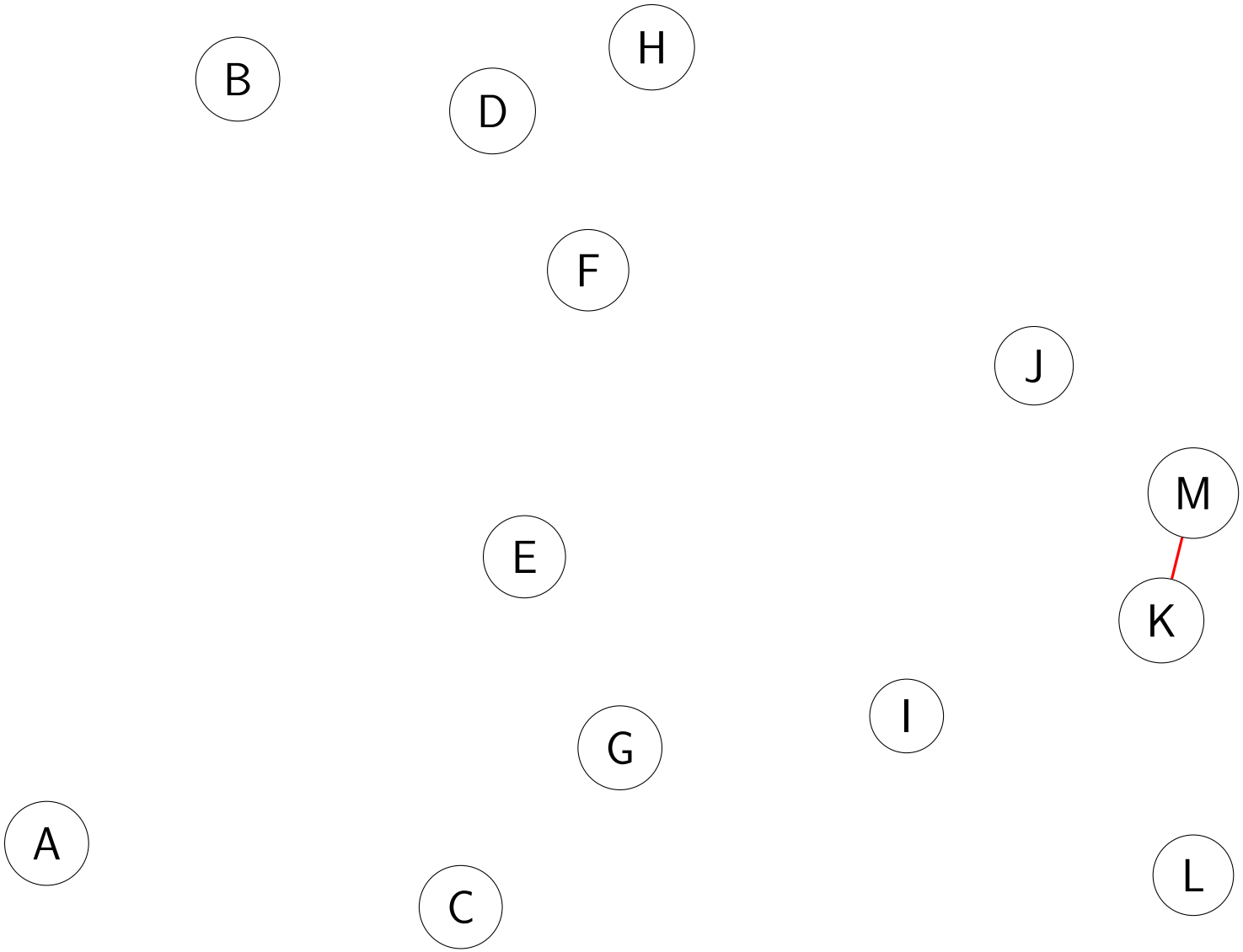
The locations of the islands:

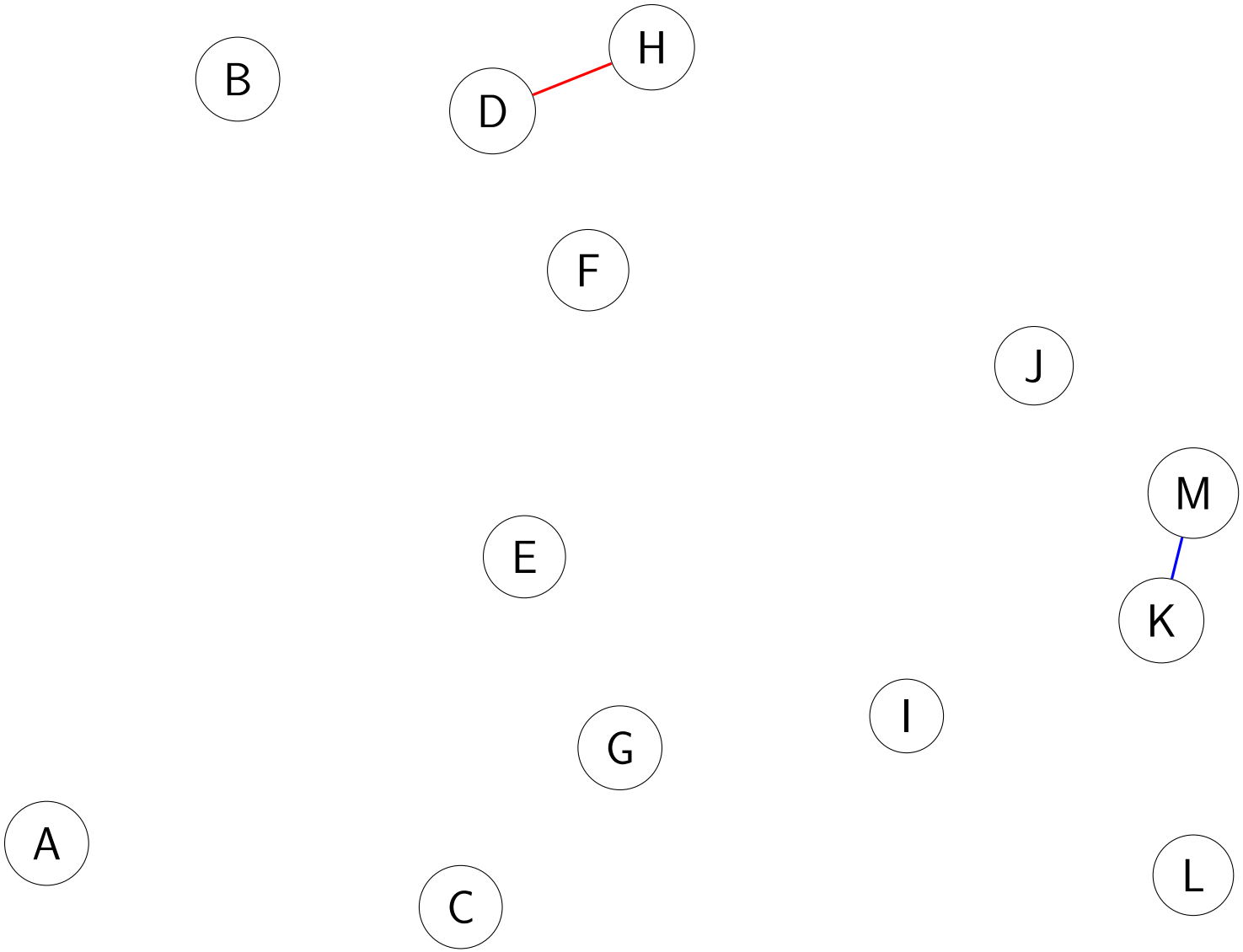
	East	North
A	4	6
B	16	54
C	30	2
D	32	52
E	34	24
F	38	42
G	40	12
H	42	56
I	58	14
J	66	36
K	74	20
L	76	4
M	76	28

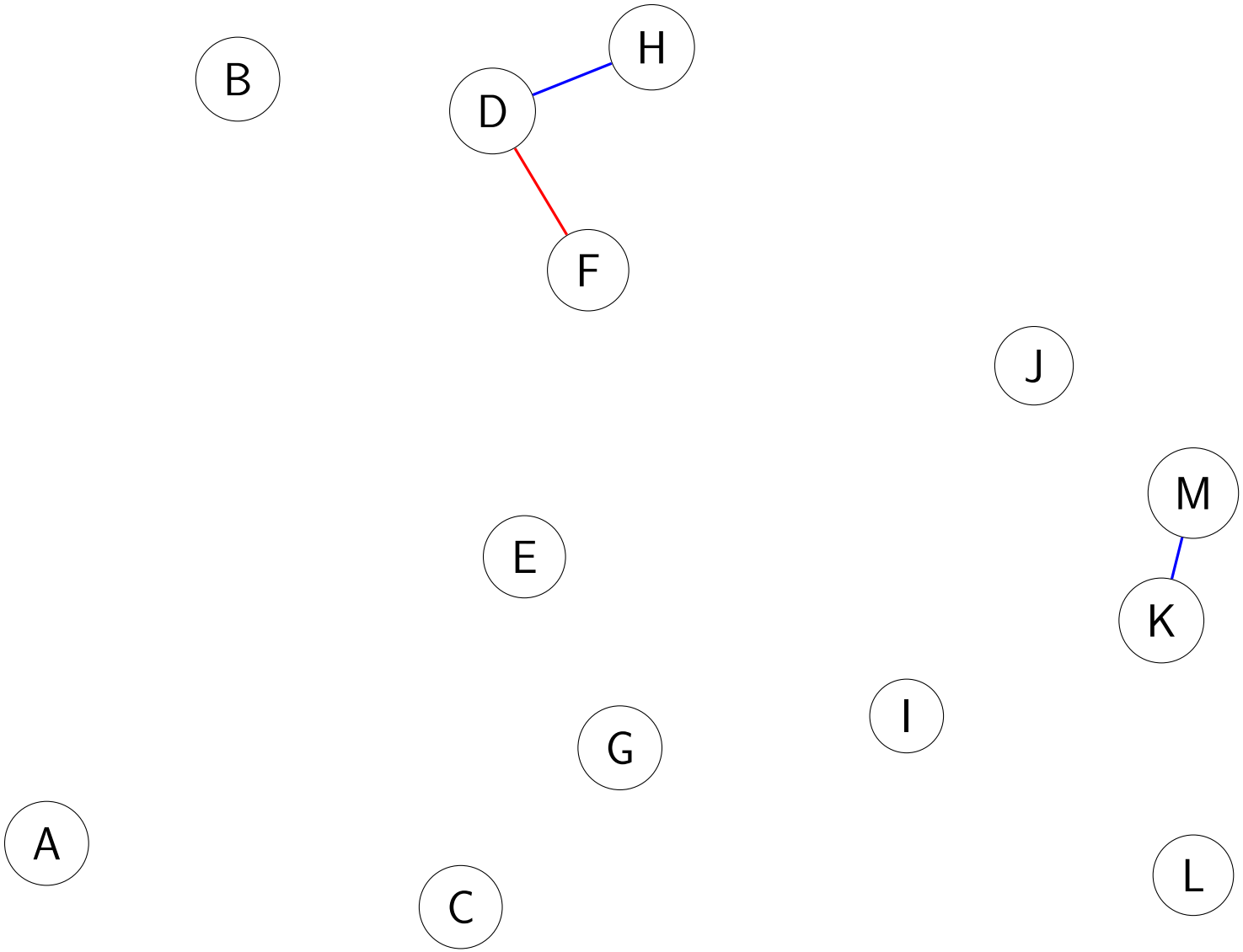

H
B
D
F
J
M
E
K
I
G
A
L
C

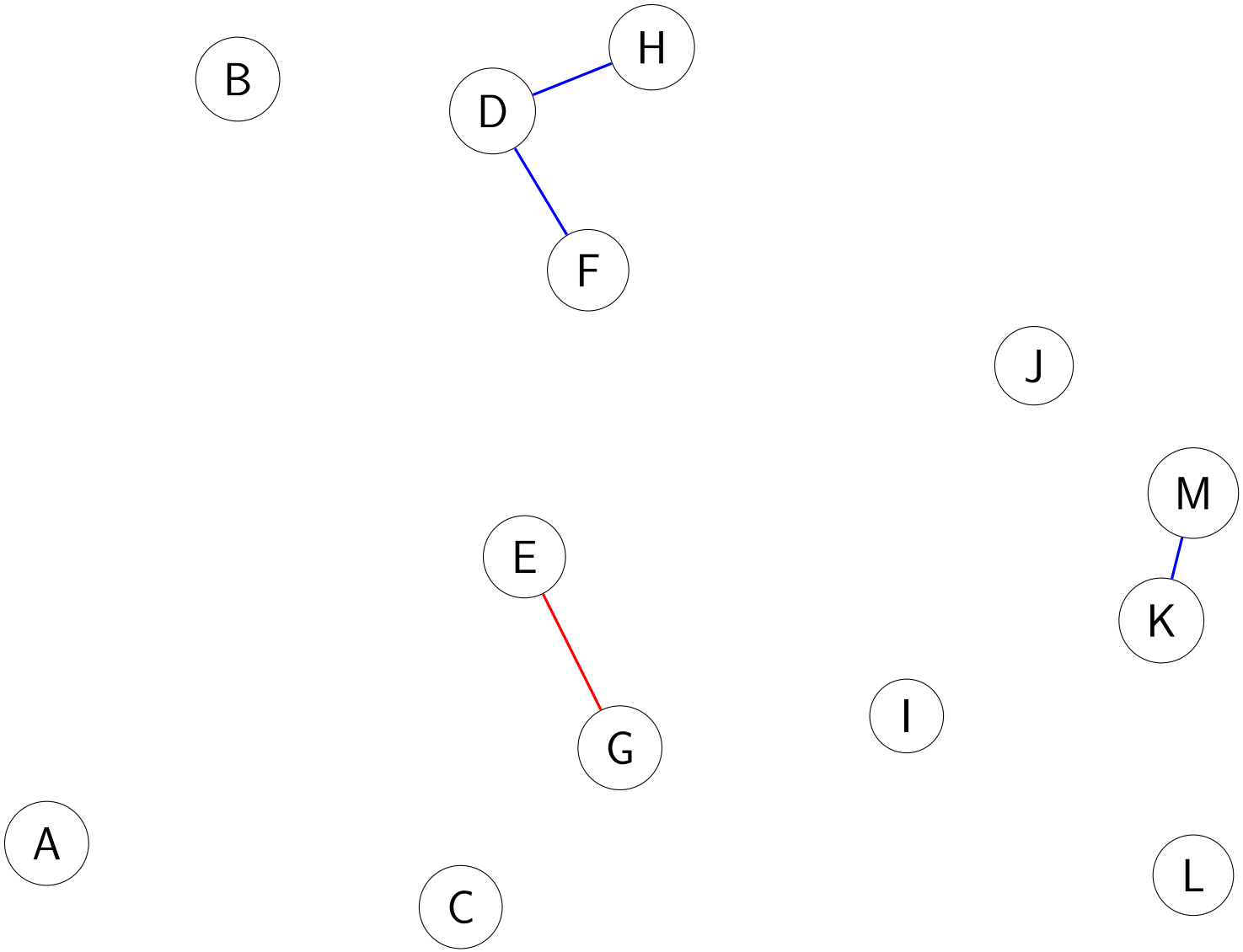
The distance table

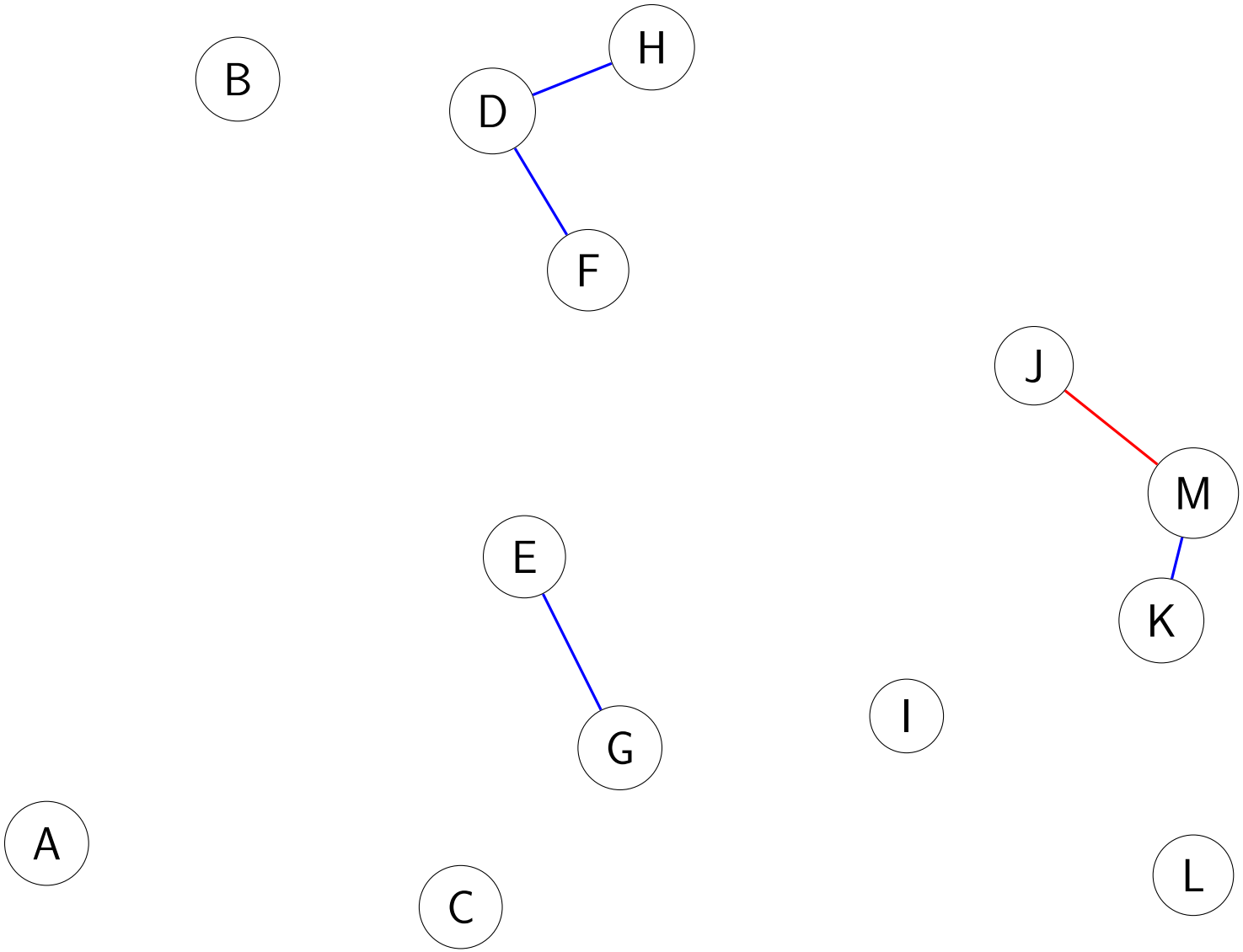
	A	B	C	D	E	F	G	H	I	J	K	L	M
A	0	49	26	54	35	50	36	63	55	69	71	72	75
B	49	0	54	16	35	25	48	26	58	53	67	78	65
C	26	54	0	50	22	41	14	55	30	50	48	46	53
D	54	16	50	0	28	12	41	11	46	38	53	65	50
E	35	35	22	28	0	18	13	33	26	34	40	47	42
F	50	25	41	12	18	0	30	15	34	29	42	54	40
G	36	48	14	41	13	30	0	44	18	35	35	37	39
H	63	26	55	11	33	15	44	0	45	31	48	62	44
I	55	58	30	46	26	34	18	45	0	23	17	21	23
J	69	53	50	38	34	29	35	31	23	0	18	34	13
K	71	67	48	53	40	42	35	48	17	18	0	16	8
L	72	78	46	65	47	54	37	62	21	34	16	0	24
M	75	65	53	50	42	40	39	44	23	13	8	24	0

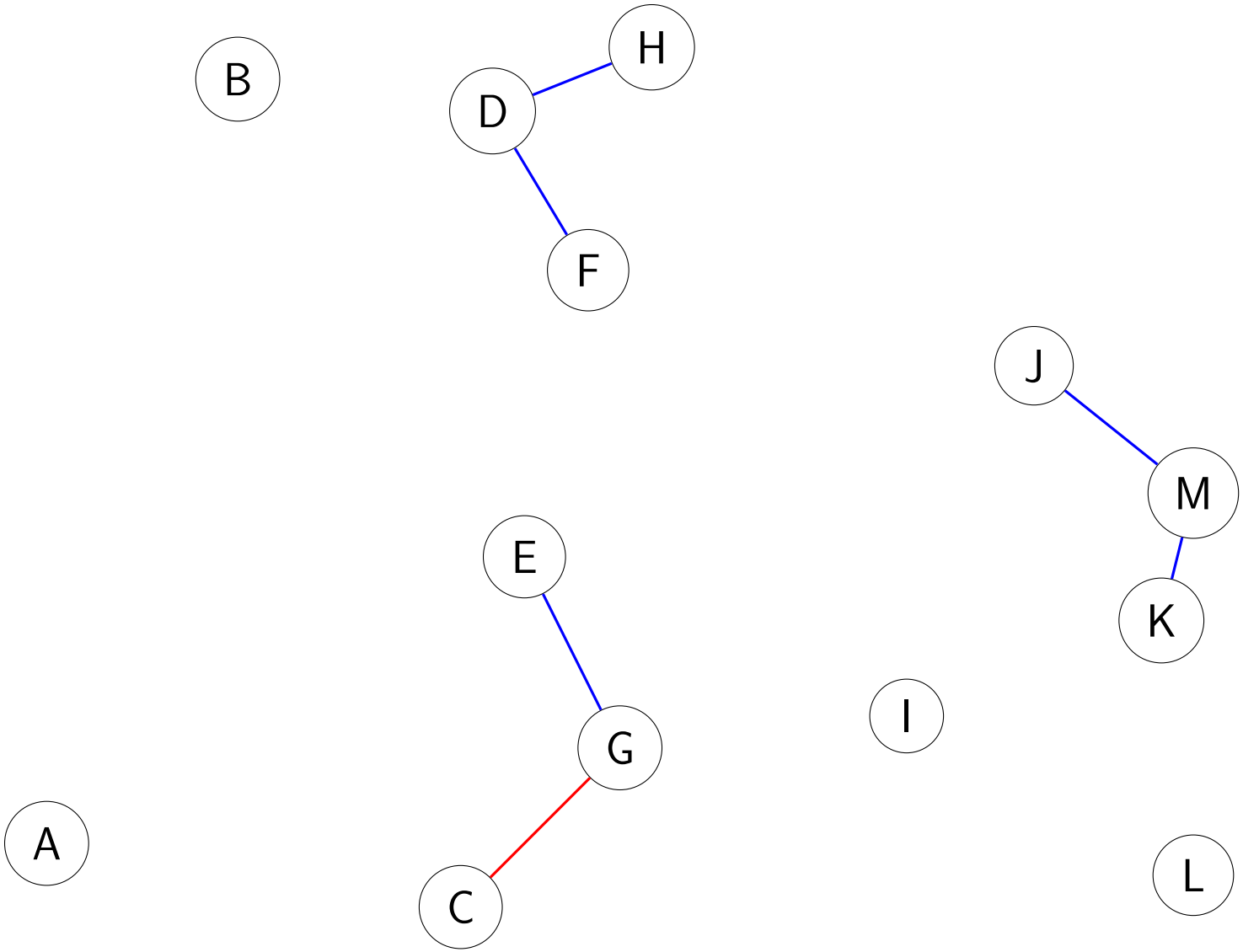

H
B
D
F
J
M
E
K
I
G
A
L
C

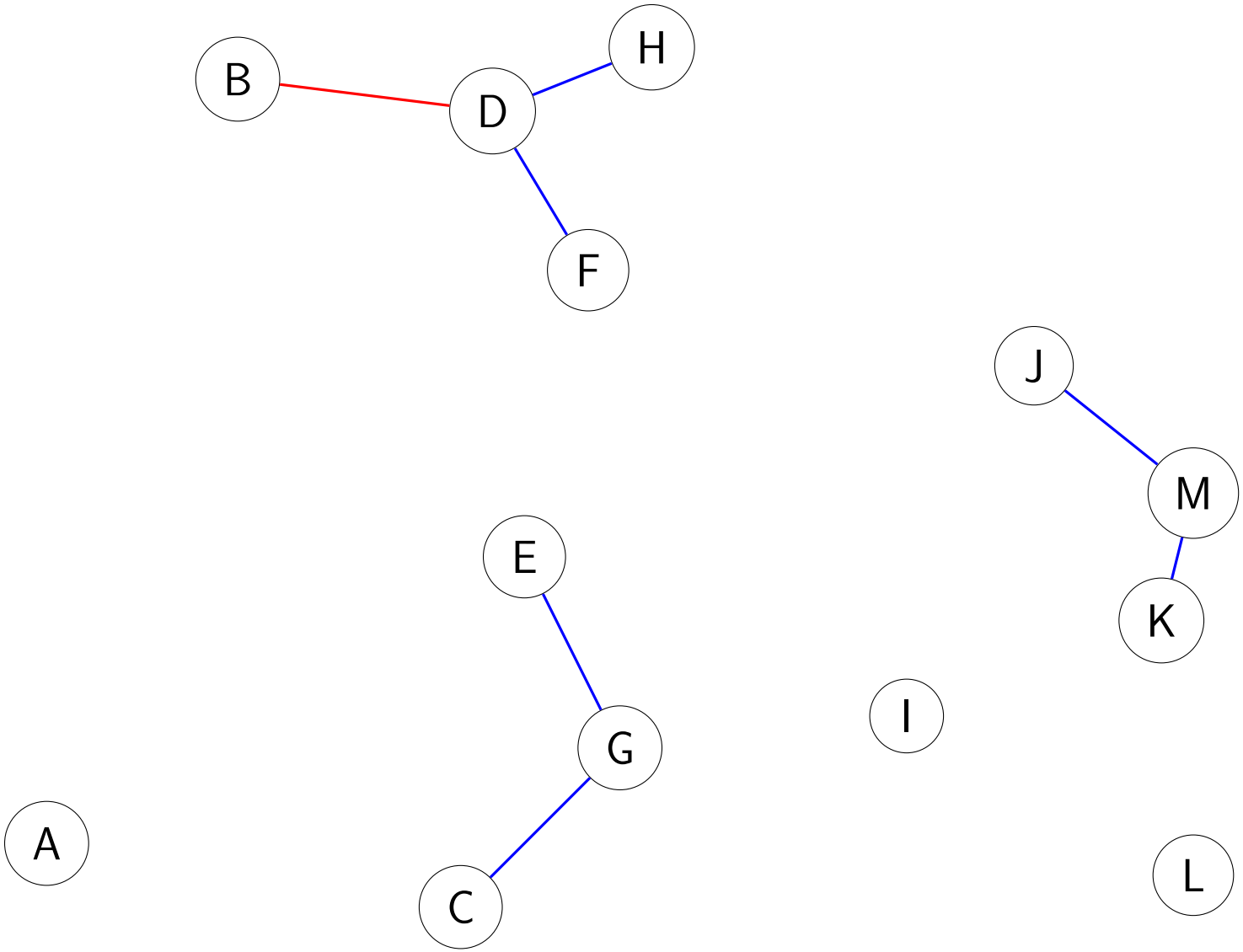

B
H
D
F
J
M
E
K
I
G
A
L
C

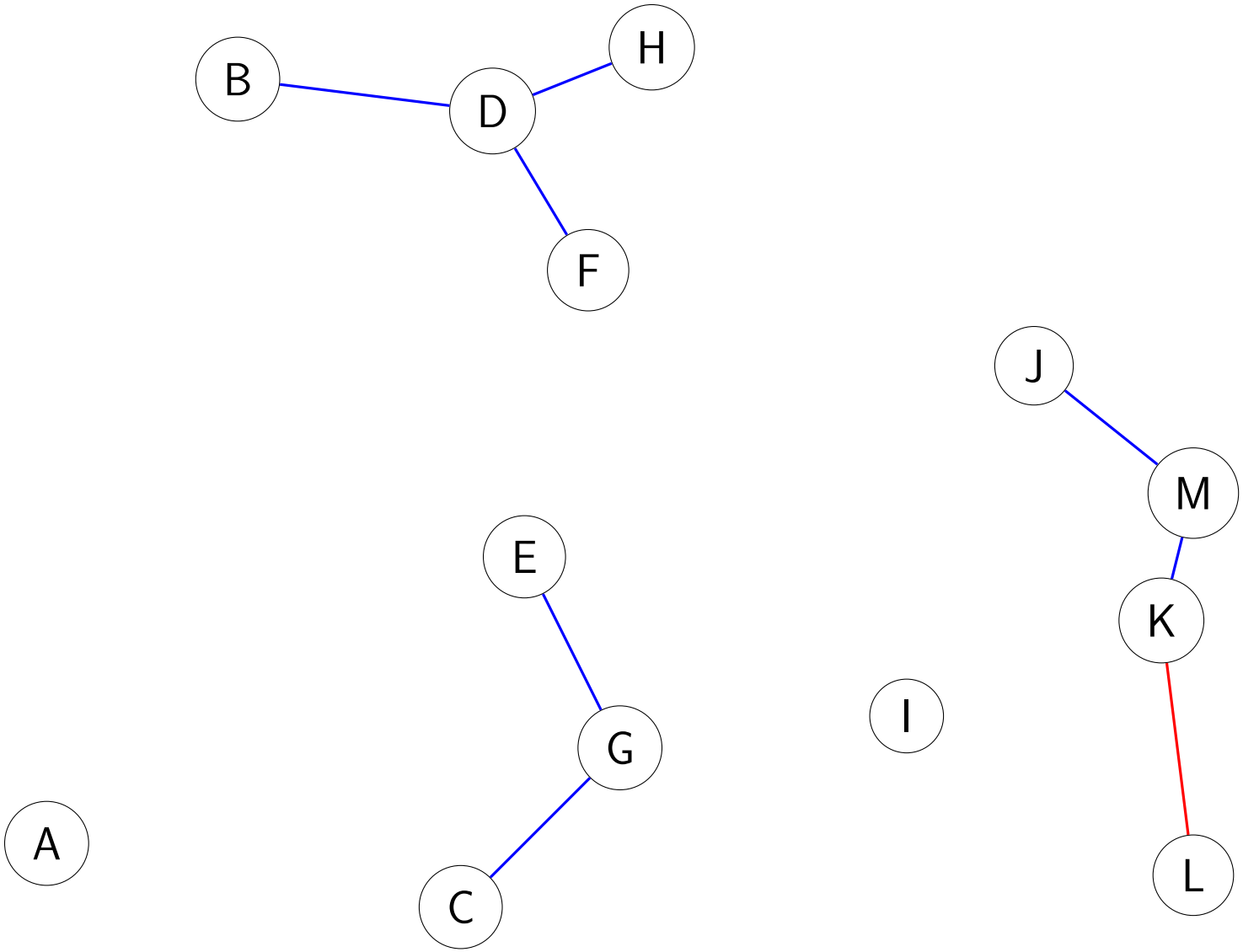

B
H
D
F
J
M
E
K
I
G
A
C
L

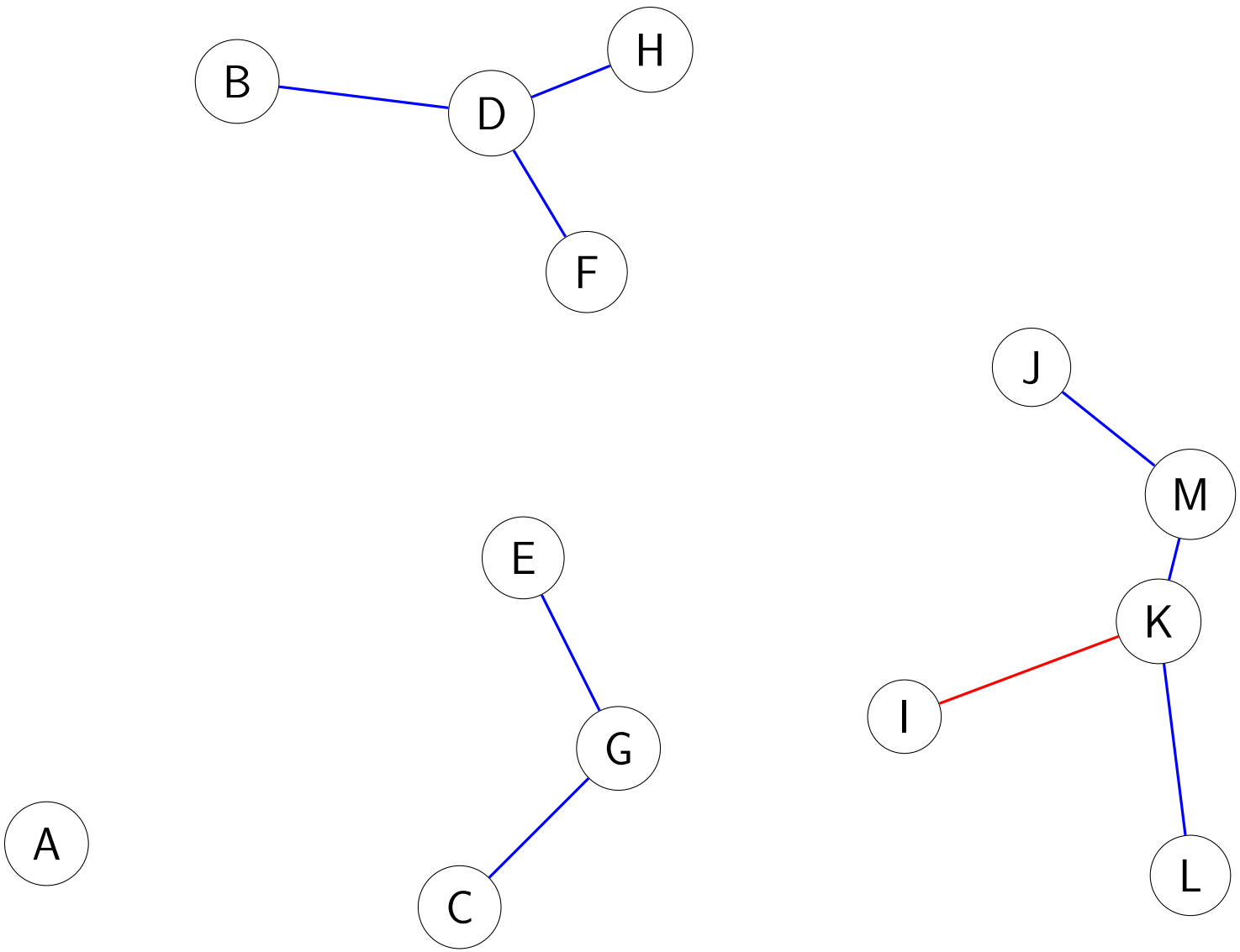

B
H
D
F
J
M
E
K
I
G
A
C
L

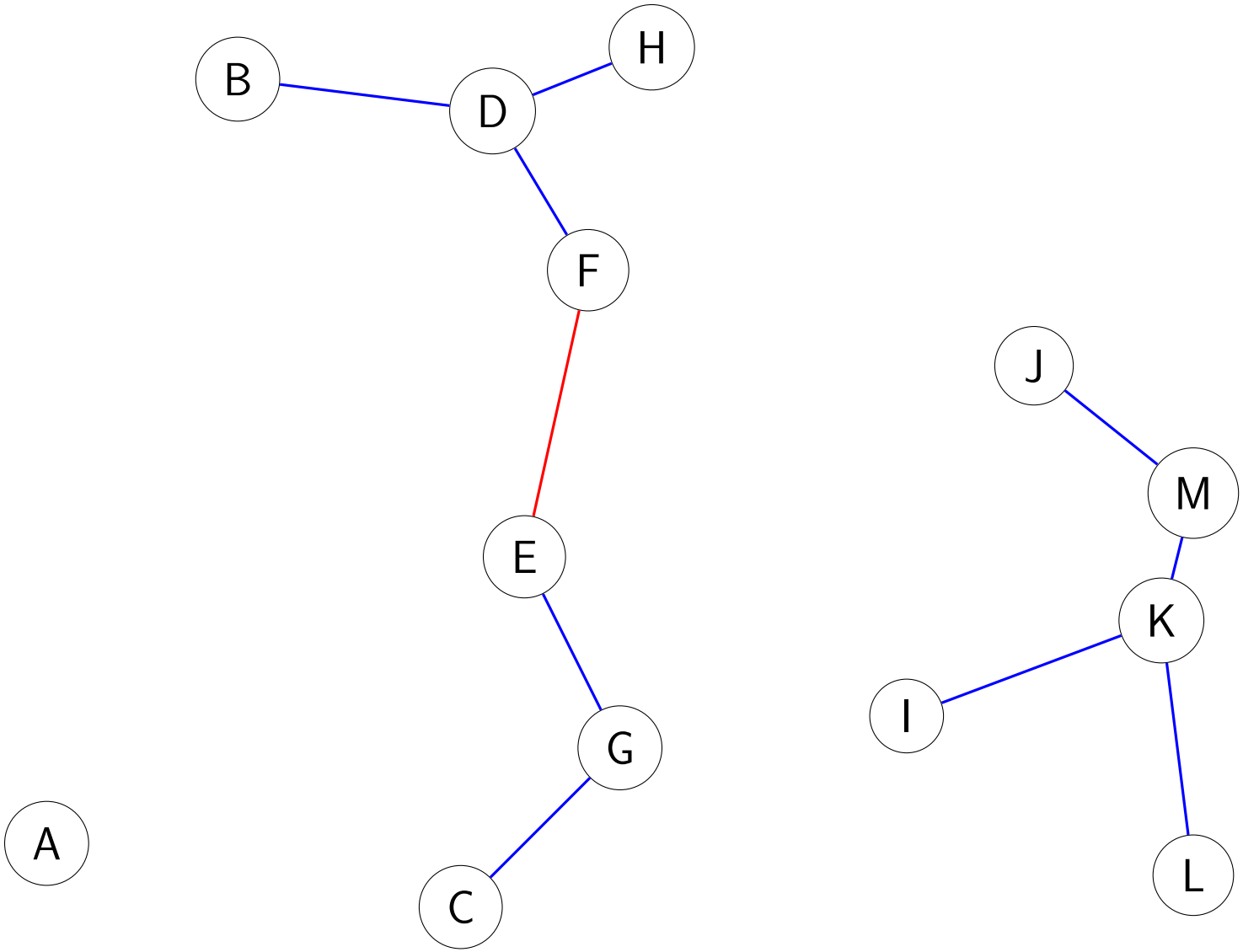

B
H
D
F
J
M
E
K
I
G
A
C
L

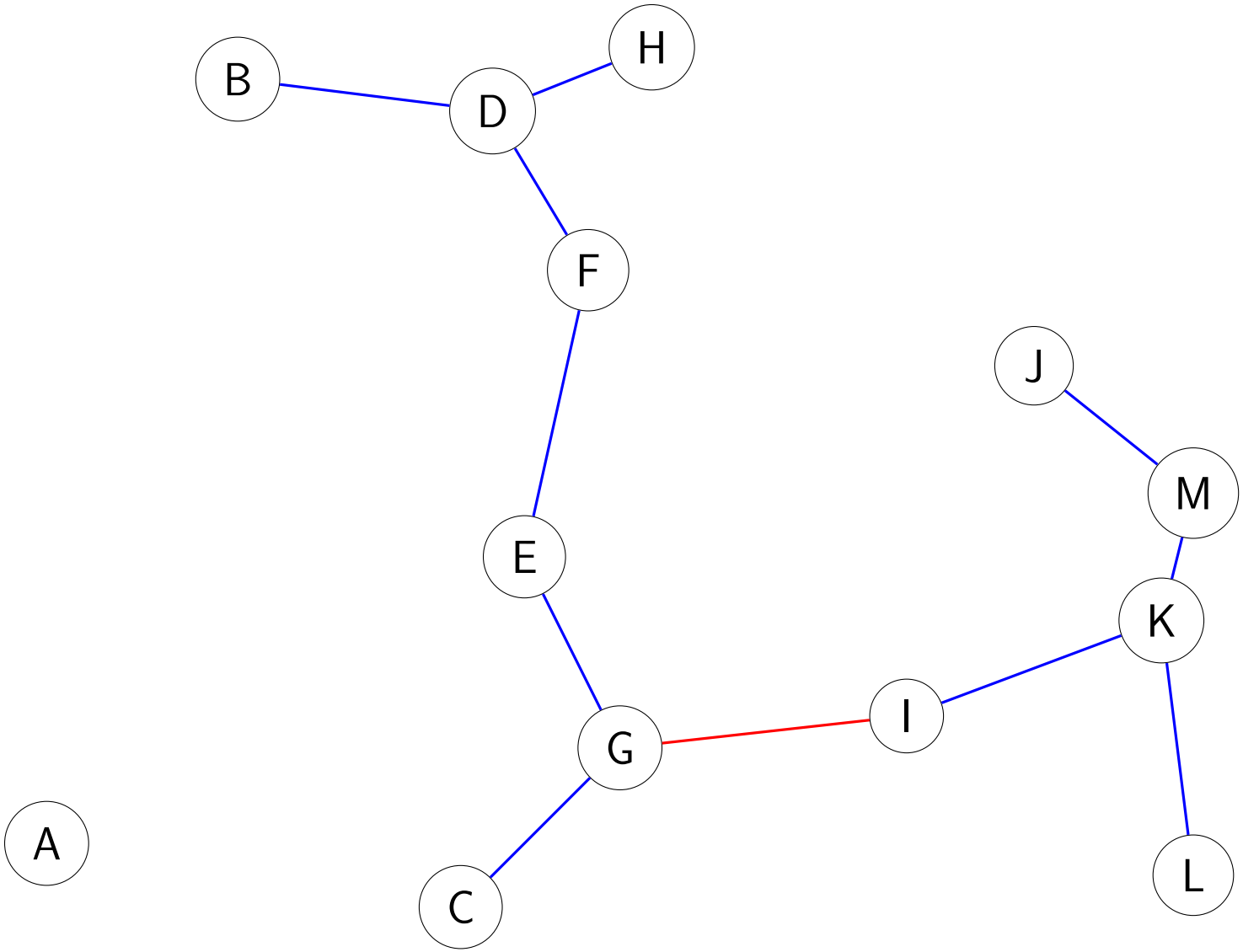

B
H
D
F
J
M
K
E
G
I
A
C
L

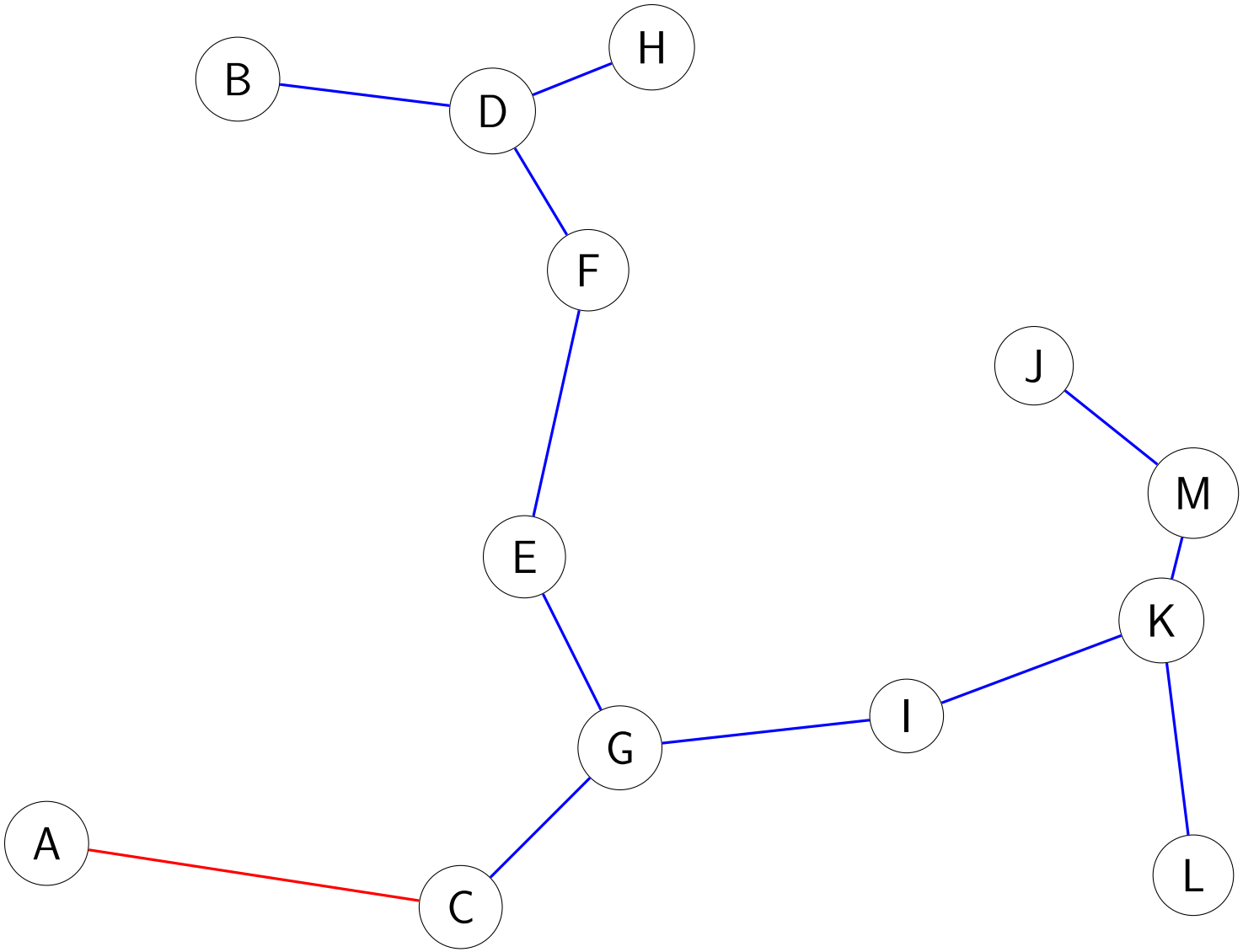

B
H
D
F
J
M
E
K
I
G
A
C
L


B
H
D
F
J
M
E
K
I
G
A
C
L


B
H
D
F
J
M
E
K
G
I
A
C
L


B
H
D
F
J
M
E
K
I
G
A
C
L


B
H
D
F
J
M
E
K
I
G
A
C
L


B
H
D
F
J
M
E
K
I
G
A
C
L

Based on the steps in the single linkage rule, we can watch as the data organizes itself into fewer, "fatter" clusters.

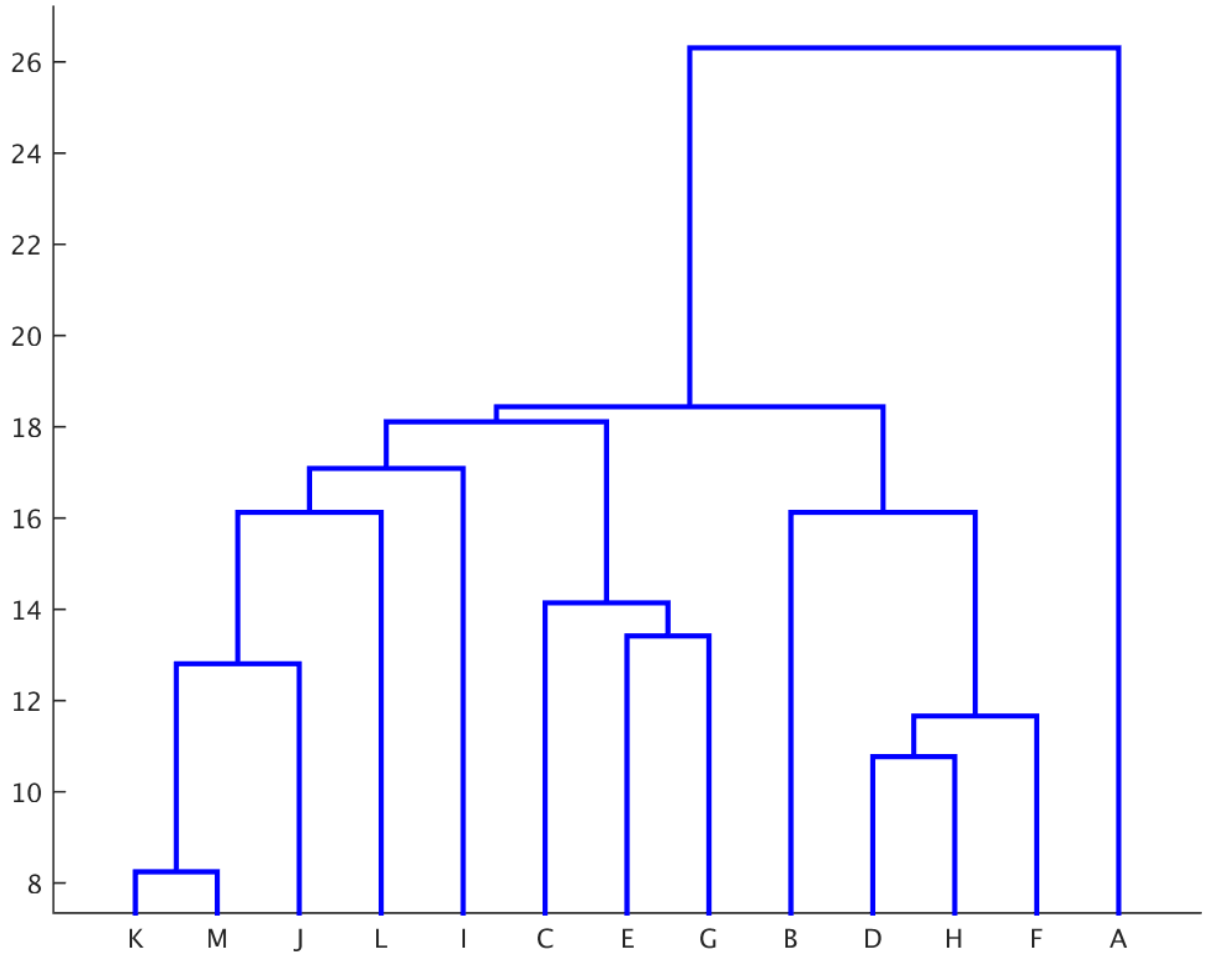
Step													Clusters
1	K	M	J	L	I	C	E	G	B	D	H	F	A	13
2	(K | M)	J	L	I	C	E	G	B	D	H	F	A	12
3	(K	M)	J	L	I	C	E	G	B	(D	H)	F	A	11
4	(K	M)	J	L	I	C	E	G	B	(D	H	F)	A	10
5	(K	M)	J	L	I	C	(E	G)	B	(D	H	F)	A	9
6	(K	M	J)	L	I	C	(E	G)	B	(D	H	F)	A	8
7	(K	M	J)	L	I	(C	E	G)	B	(D	H	F)	A	7
8	(K	M	J)	L	I	(C	E	G)	(B	D	H	F)	A	6
9	(K	M	J	L)	I	(C	E	G)	(B	D	H	F)	A	5
10	(K	M	J	L	I)	(C	E	G)	(B	D	H	F)	A	4
11	(K	M	J	L	I)	(C	E	G	B	D	H	F)	A	3
12	(K	M	J	L	I	C	E	G	B	D	H	F)	A	2
13	(K	M	J	L	I	C	E	G	B	D	H	F	A)	1

One of the really beautiful things about the single linkage rule is that it suggests a way to arrange the data in a tree.

All the data appears at the bottom of the tree.

As we begin to move upwards, we see that items K and M are the first to join, separated by the smallest distance of 8. Shortly after, items D and H, separated by a distance of 11, are joined. As we move further up the tree, the number of clusters is reduced one by one, until the very remote data item A is the last to join up.

Computational biologists especially like using single linkage clustering, because it can arrange plants or animals in a sort of family tree that may suggest a theory of their relationships.


26
24
22
20
18
16
14
12
10
8
K
M
J
L
I
C
E
G
B
D
H
F
A

In 1952, a letter was mailed anonymously from Tennessee:

This Prayer has been sent to you and has been around the world four times. The one who breaks it will have bad luck.

The Prayer. Trust in the Lord with all thy heart and lean not on thy own understandance in all thy ways acknowledge him and he will direct thy path.

Please copy this and see what happens in four days after receiving it. Send this copy and four to someone you wish good luck. It must leave in 24 hours. Don't send any money and don't keep this copy. Gen Patton received $1,600 after receiving it. Gen Allen received $1,600 and lost it because he broke the chain.

You are to have good luck in 4 days. This is not a joke and you will receive by mail.

Copies of this letter were still being received 40 years later. Especially in the beginning (before Xerox copiers, and then email) a person had to write the copies by hand, or typewrite them. A copy received after 40 years had probably been copied 4,000 times!

All copies of the letter can be traced back to one person, the original sender. But any two copies of the letter may come from a single sender much more recently.

The version that was seen in the 1980's was very interesting, because it mentioned several people by name. However, because of carelessness in copying, these names varied from copy to copy. The copying errors make it possible to try to cluster the chain letters. Letters in which the names do not differ much probably came from the same person not too long ago. If letters have many differences, their common ancestor must be further back.

The names in the chain letter are useful because...they are not important! They can change without the letter losing its meaning.

There is a similar situation in DNA. Substantial amounts of DNA are the same for all animals because it contains vital information that can't easily be changed without harming the animal.

But there is much genetic material that doesn't do anything. It's called **junk DNA**. Since it's useless, it can change and the animal will pass on the mutation. Thus, junk DNA can be used to track relatedness of individuals.

The names in the chain letters are like junk DNA. As we watch them change, we can try to guess the pattern of ancestry!

LETTER A

With love all things are possible.

This paper was sent to you for good luck. The original is in **New England** (1). It has been around the world nine times. The luck has been sent to you. You will receive good luck in the mail. Send no money as faith as no price. Do not keep this letter. It must leave your hands within 96 hours.

An R.A.F. officer (2) received $470,000. Joe Elliot receive $40,000 and lost it because he broke the chain. While in the Phillipines, **George Welch** (3) lost his wife 50 days after receiving the letter. He had failed to circulate the letter. However, before her death, he received $7,775,000.

Please send 20 copies and see what happens in four days. The chain comes from Venezuela and was written by **Saul Anthony Degrow** (4), a missionary from South America. Since this copy must tour the world, you must take twenty copies and send them to friends and associates. After a few days, you will get a surprise. This is true,

even if you are not superstitious.

Do note the following: **Constantince** (5) had received the chain in 1933. He asked his secretary to make twenty copies and send them. A few days later he won a lottery of two million dollars. **Carlo Daddit** (6), an office employee, received the letter and forgot it had to leave his hands in 96 hours. He lost his job. Later, after finding the letter again, he got a better job. **Dalen Fairchild** (7) received the letter and not believing, threw the letter away. Nine days later, he died.

In 1987, the letter was received by **a young lady in California** (8). It was very faded and barely readable. She promised herself she would retype the letter and send it out but she put it aside to do it later. She was plagued by various problems, including expensive car repairs. The letter did not leave her hands in 96 hours. She finally retyped the letter as promised and got a new car.

Remember, send no money. Do not ignore it. It works.

Here are some key points in the chain letter:

1. The original is in **New England**.
2. An **R.A.F.** Officer received $470,000.
3. In the Philippines, **Gene Welch** lost his wife...
4. The chain...was written by **Saul Anthony de Groda**...
5. Do note the following: **Constantine Dias**...
6. **Carlos Daddit**, an office employee...
7. **Dalan Fairchild** received the message...
8. In 1987, the message received by **a young woman in California**...

If we compare the key points in two chain letters, we will see some matches and some differences.

To do clustering, we need some way to measure the distance between any two objects. For our chain letters, we can measure distance by looking at differences. The "distance" between two chain letters will be based on the key points that are different.

We will assign a score for agreement or disagreement on each point, and then sum to get the distance.

When we say **New England**, we don't mean the chain letters must both have that particular word. We are instead trying to suggest the position in the letter that must be checked. So we mean, do the chain letters both have the same word there (perhaps **New Zealand!** or not.

1. **New England**, 0 if the same name is used, 1 otherwise
2. **R.A.F.Officer**, 0 if the same rank, 1 otherwise.
3. **Gene Welch**, 0 if the same name, 1 if one name different, 2 if both different.
4. **Saul de Groda**, 0 if the same names, 1, 2 or 3 if 1, 2 or 3 names different or added
5. **Constantine Dias**, 0 if the same name, 1 or 2 if 1 or 2 names different
6. **Carlos Daddit**, 0 if the same name, 1 or 2 if 1 or 2 names different
7. **Dalan Fairchild** 0 if the same name, 1 or 2 if 1 or 2 names different
8. **a young woman in California**, 0 if both letters have or don't have this part, 5 if one does and one doesn't

You will be assigned one of the 11 chain letters, which are labeled $\mathbf{A}$ through $\mathbf{K}$.

Compute the distance of your chain letter to all the other chain letters. Do this by computing the pairwise score of your chain letter against the others.

When you are done, you will have computed 11 numbers, which represent one row of the distance matrix.

Here is a table computed earlier:

	A	B	C	D	E	F	G	H	I	J	K
A	0	9	9	9	7	7	13	10	14	12	9
B	9	0	9	7	10	6	18	11	15	12	8
C	9	9	0	8	9	7	15	11	14	14	9
D	9	7	8	0	8	7	14	9	14	13	0
E	7	10	9	8	0	9	16	11	13	13	8
F	7	6	7	7	9	0	14	11	12	12	8
G	13	18	15	14	16	14	0	16	12	11	13
H	10	11	11	9	11	11	16	0	15	16	9
I	14	15	14	14	13	12	12	15	0	9	12
J	12	12	14	13	13	12	11	16	9	0	11
K	9	8	9	0	8	8	13	9	12	11	0

By chance, the chain letters are grouped in such a way that the distance matrix suggest that G, H, I and J are "far away" from the rest of the letters.

If you are patient, you will also notice that D and K are apparently 0 units apart, and, correspondingly, their rows are almost identical.

However, it's hard to see everything that this distance table is telling us. We can do a single linkage analysis, and get a dendrogram, which may be much easier for us to understand.

The dendrogram gives us a theory that assumes there was a single chain letter which gradually, because of copying mistakes, began to show more and more variations.

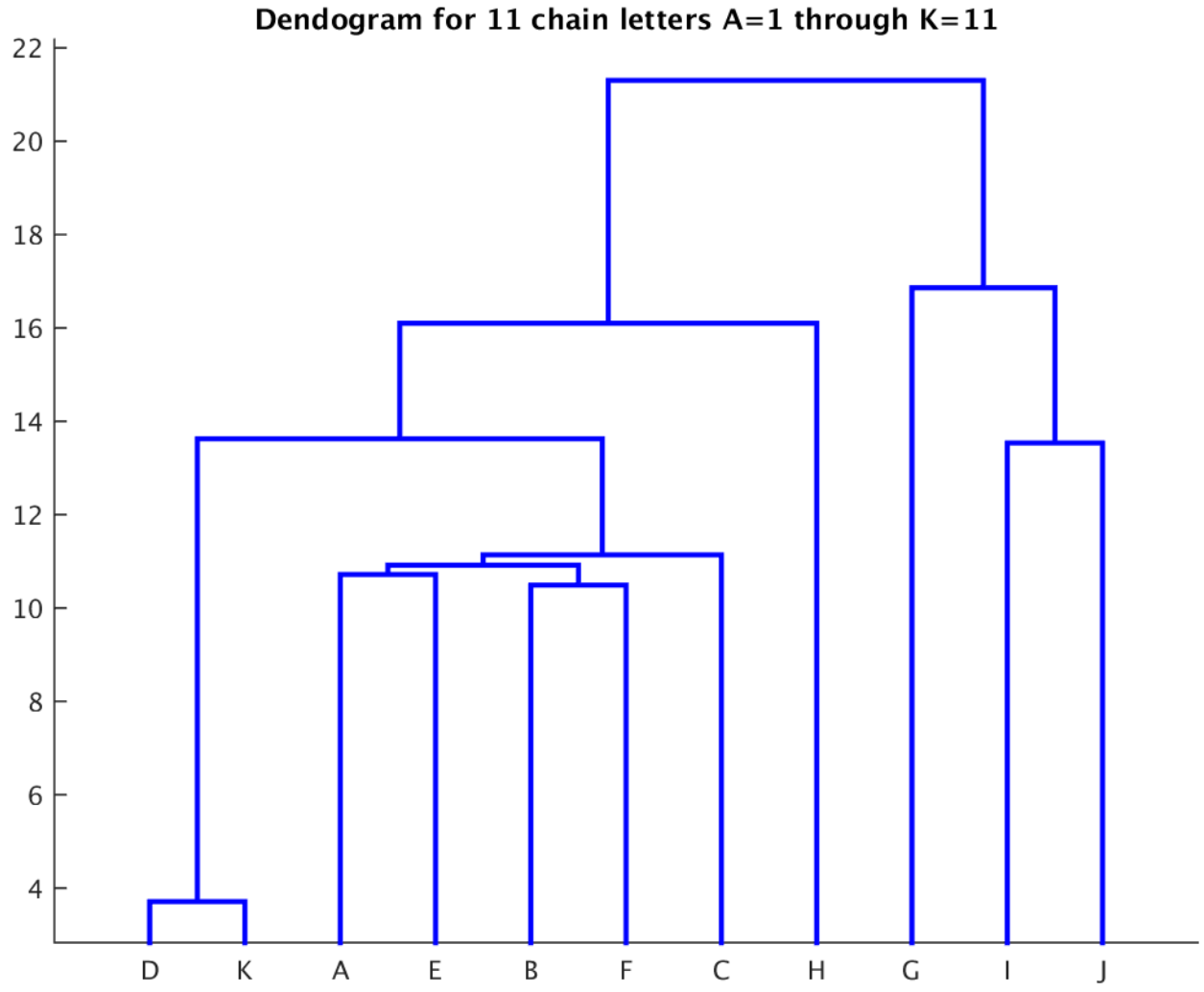
Thus, with our evidence, we can guess that the original chain letter split into two variations, one the parent of ABCDEFHK, the other of GIJ.

GIJ later split into G and IJ, after which I and J split.

ABCDEFHK split into ABCDEFK and H.

ABCDEFK split into DK and ABCEF.

ABCEF split into AE and BF and C.


Dendogram for 11 chain letters A=1 through K=11
22
20
18
16
14
12
10
8
6
4
D
K
A
E
B
F
C
H
G
I
J

Instead of starting at the top, we can pick any single chain letter, and try to trace its heritage backwards. In that case, we have to ask ourselves, as each place where two branches merge, "What features do the two branches have in common, and how do they differ?"

Let's consider chain letter B, whose history reads like this:

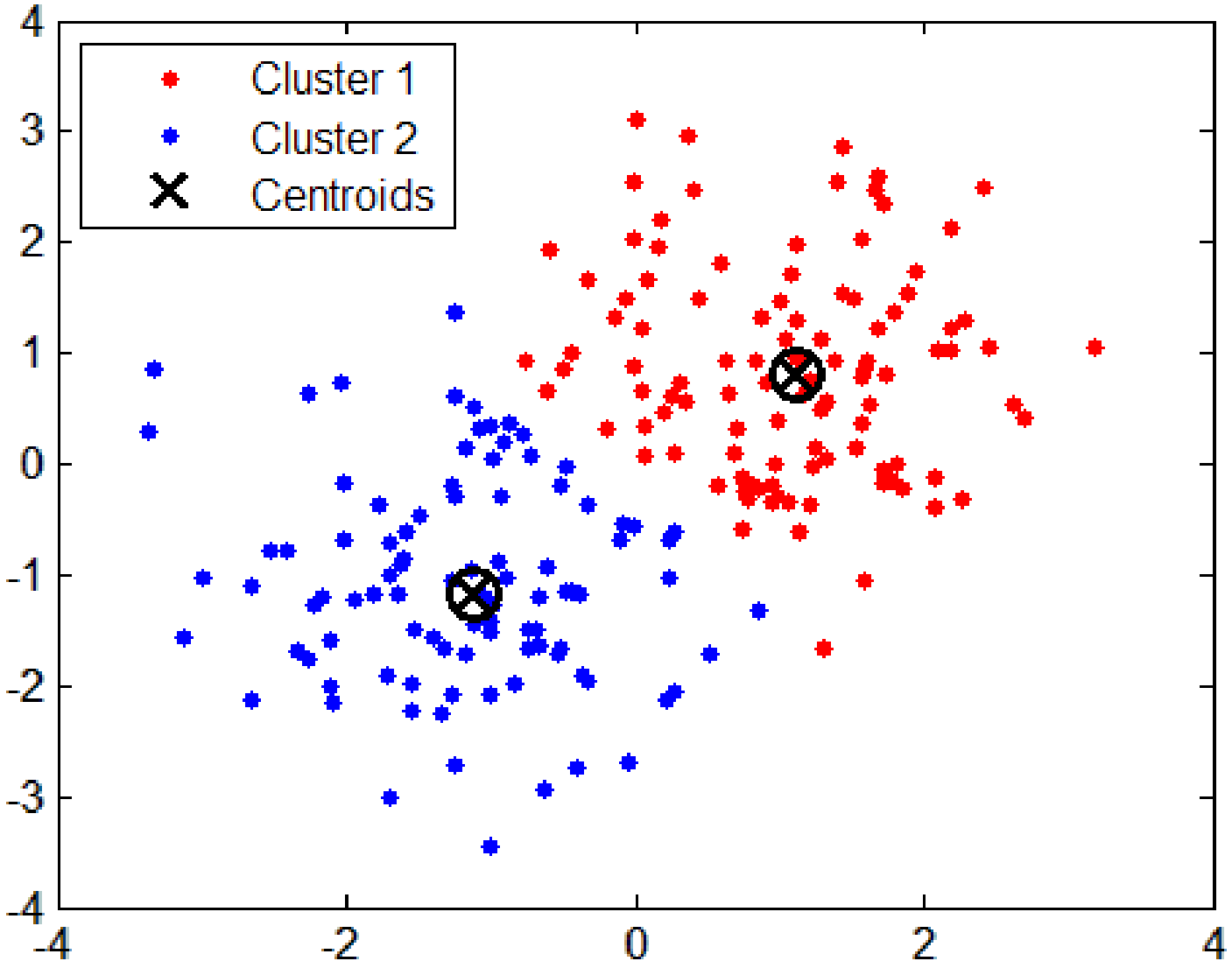
1. B
2. BF
3. ABEF
4. ABCEF
5. ABCDEFK
6. ABCDEFHK
7. ABCDEFGHIJK

Single linkage clustering is especially useful if we have reason to think our data has some kind of family tree relationship.

A more general clustering tool is called **K-Means**. Here the fundamental idea is that we are going to look for $\mathbf{K}$ average or mean values, about which the data can be clustered. Our goal now is perhaps no so much to find a family history as to simply break the data up into $\mathbf{K}$ groups.

We might be doing this in the hope that each group can be "explained" by some common parameter value.

Or we might not be looking for understanding - instead we might simply want to compress the data.


Cluster 1
Cluster 2
Centroids
4
3
2
1
0
-1
-2
-3
-4
-4
-2
0
2
4

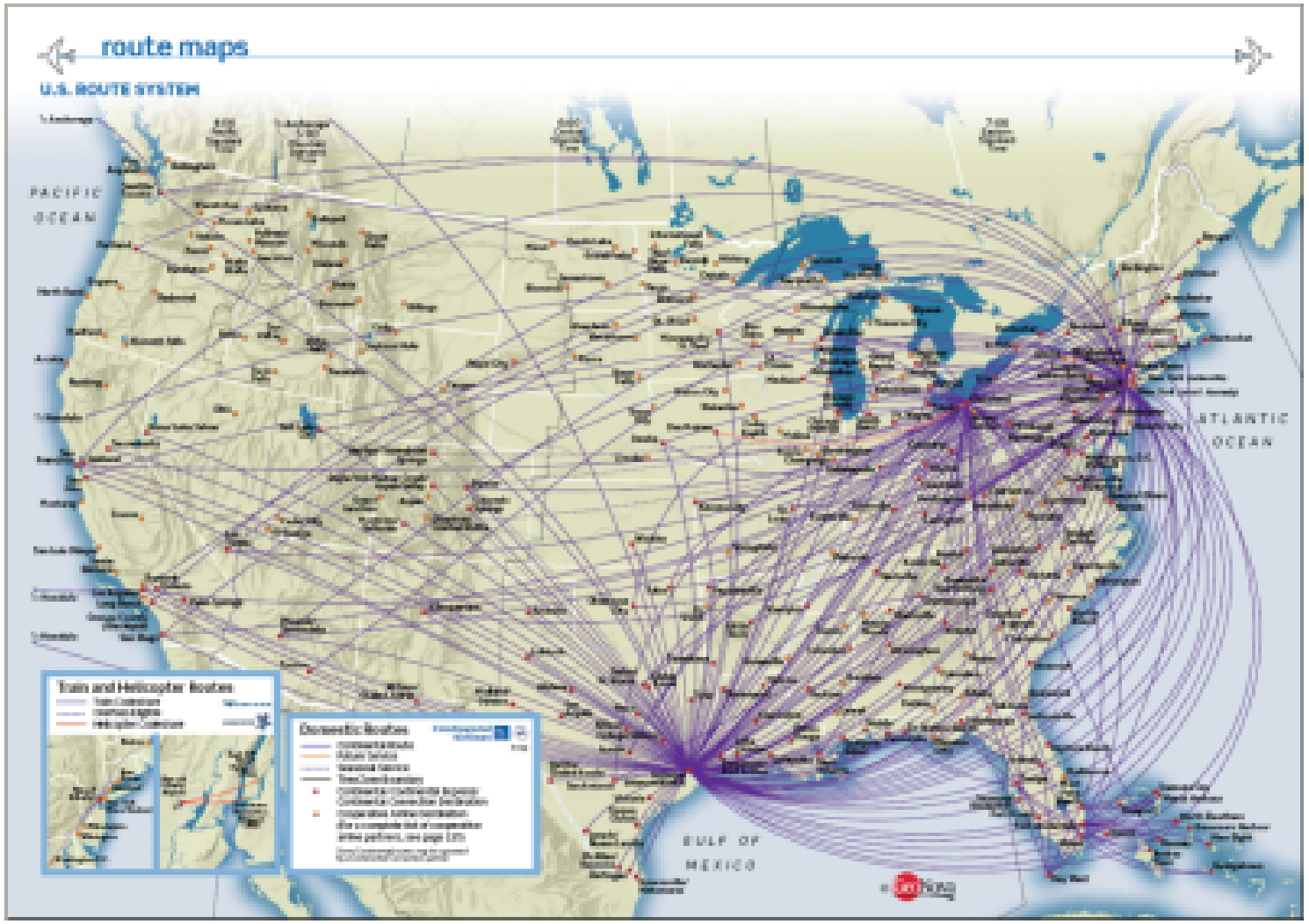
The data that K-Means works with must be numerical. Each data object must be describable in terms of numerical coordinates.

We can imagine that these coordinates represent a spatial position. A surprising number of things can be described this way, including

- the weekly position of one song in the Top 40 ratings;
- the election results, by party, in one particular town;
- the medical test results for one patient in a study;
- the RGB coordinates of one color used in an image

We aren't looking for a theory about how these objects are created or why they are different. We are instead trying to get a representative sample of the data, so that we can see the most typical behaviors and then try to explain patterns that we see:

- do most hit songs vanish after 10 weeks? do songs by new artists tend to be short-lived?
- how do towns differ, based on differences in the election results?
- if some patients in the study got very sick afterwards, are there some test results that would have predicted this?
- if we can only use 8 colors in an image, what is a choice that would be representative?


route maps
U.S. ROUTE SYSTEM
PACIFIC OCEAN
ATLANTIC OCEAN
GULF OF MEXICO
Trains and Helicopter Routes
Domestic Routes

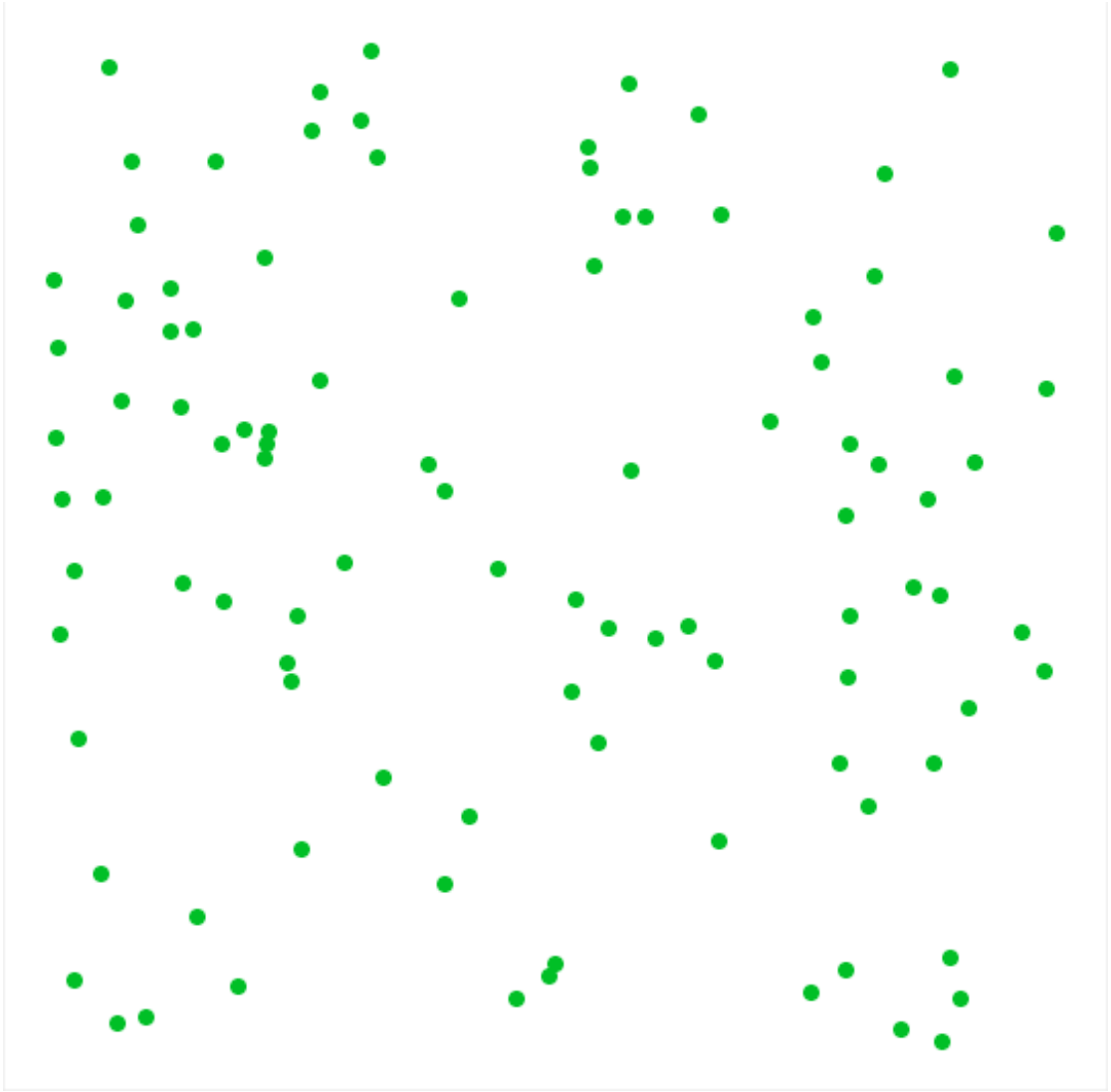
Sometimes the data for K-Means really is spatial, and in that case, we can understand a little better what it is trying to do.

One big advantage is that we can picture the problem on a map, and that the "difference" between two objects is now simply their distance.

For instance, we can imagine that K-means clustering could help us decide where to locate the $\mathbf{K}$ "hubs" of an airline so that they are well spaced around the country, in such a way that most airports are close to their local hub.

A spatial clustering problem might begin with no information, except

- the number of points to be clustered;
- the location of these points;
- a suggested number of clusters to use.

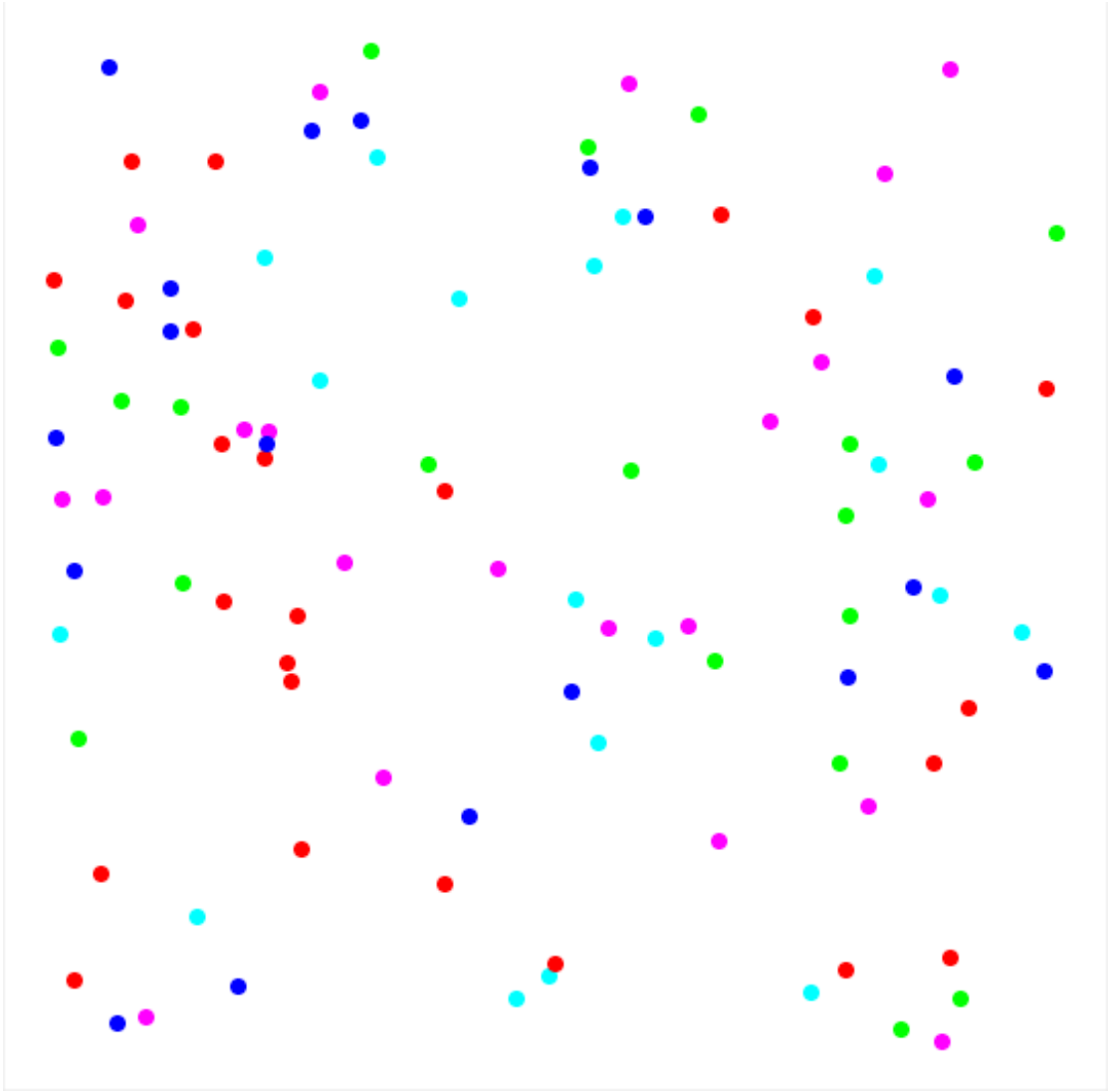


We can look at the picture, and make some reasonable guesses about how the points might be grouped.

But we may ask a computer to work on this problem, so let's assume we can't use our eyes to make a good initial guess for the organization.

In that case, why don't we just try a random clustering, that is, we simply assign every point to one of the clusters, without checking where the points are.

This is a start, not a solution. But sometimes in computational work, all you need is a starting point, and an idea of how to search for small improvements.

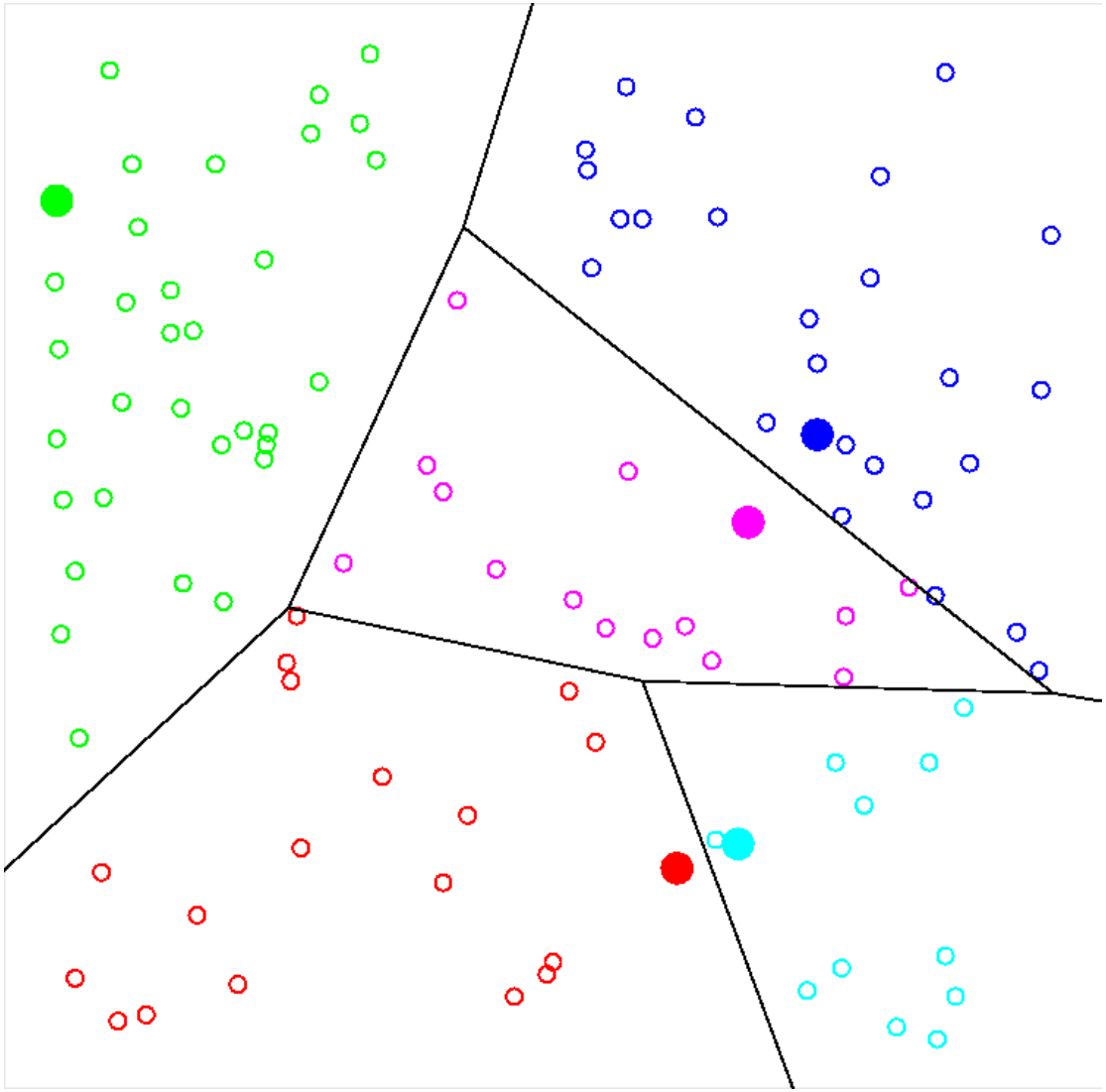


Now we see that our random clustering actually very bad.

We can try an improvement by arbitrarily picking 5 special center pointsin the region, and assigning each point to the nearest center.

This seems to make a big improvement - points really do form clusters now.

However, we don't have any idea how to pick those center points, so again we just chose them randomly.



Even though the clustering induced by our random centers wasn't great, it did divide the data. In fact, the oddest thing about the clusters now is that the centers aren't actually in the center.

So we got our grouping (good) but the cluster centers aren't actually good representatives of the clusters (bad).

Suppose we move our center points to the actual centers of the clusters. Then it's likely that the centers are more evenly spaced throughout the region, and this will also likely reduce the average distance of cluster points to the center.

Of course, if we move the center of the red cluster to the middle of the red cluster, a few points in the red cluster will actually be further away from the new location of the center.

In fact, they might be closer to the new location of the blue cluster. Since all we care about is reducing the distances, it makes sense for such points to switch from the red cluster to the blue cluster.

In fact, we should allow all points to check whether, after the centers moved, it would be to their advantage to switch, and if so, we should switch them.

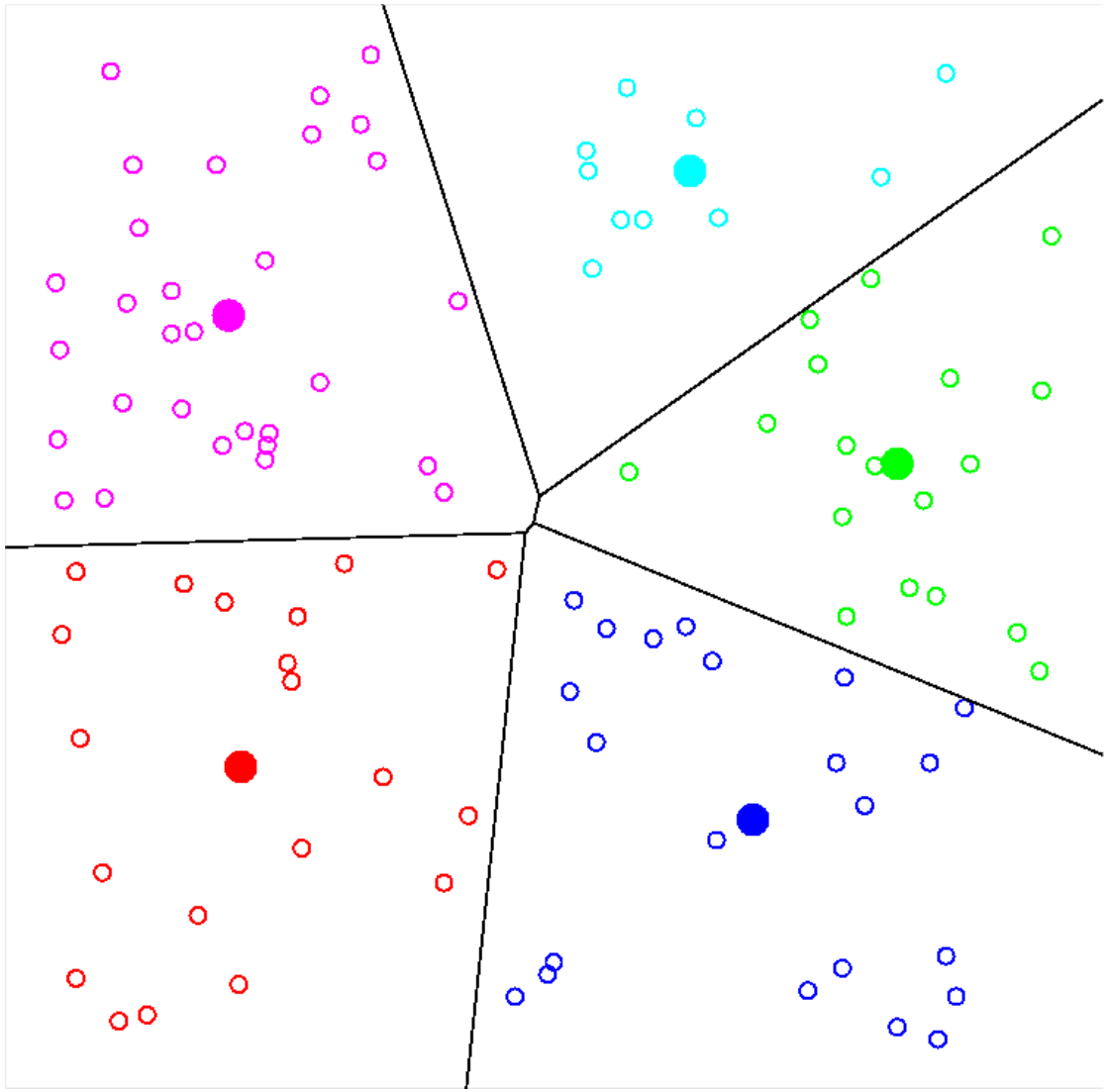
Even if a single point moves to another cluster, this affects the locations of the cluster centers of the cluster it left and the cluster it joined.

That's because we are now defining the cluster centers to be the averages of all the points in the cluster.

So once again we must update the cluster centers, and this in turn may cause some points to want to transfer.

We clearly will need to carry this out as an iteration. It might seem that this process could continue forever. However, ever step of the process reduces our total distance measure (which is good) and the number of points that want to switch typically goes down very quickly and will finally stop.

If we wait til this process has settled down, we get the K-means clustering of the data.



It's easy to see what's going on when the data you are working with is points in a plane.

However, there are many other kinds of data for which clustering can be used.

We will look at an example in which K-means clustering can help us reduce the amount of computer storage required for an image.

Original image

Images are everywhere on the computer, including user profile pictures, party photos on FaceBook, digitized artwork, and movies, which are just a carefully timed sequence of images.

It's now common for a camera to deliver images of several million pixels; a typical image size might be 4,000 x 3,000 = 1,200,000 pixels.

You can think of the image as a checkboard of colors.

You can also think of it as a table of numbers, with the information at row i, column j defining the color of the corresponding pixel.

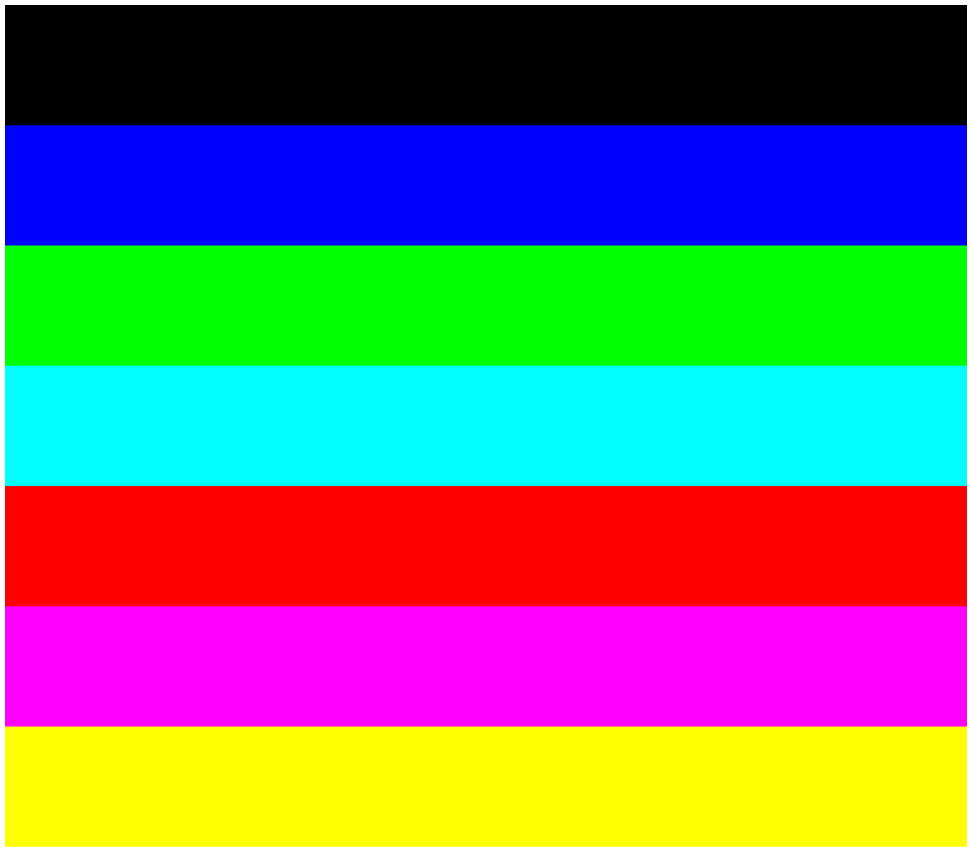
To define the color of a pixel, we need just one number, the gray level, if we're doing a black and white image, or 3 numbers (the red, green and blue levels) for a color image.

Finally, just as crayons come in sets of 8, 16, or 64 colors, computer color levels often have 16 or 256 choices.

To understand how the number of color choices will affect the size of an image, we'll compare two simple cases:

- A true black and white image; each pixel has a choice of black (K) or white (W).
- An 8-color image, with choices of:
  - $\mathbf{K}$ black;
  - $\mathbf{B}$ blue;
  - $\mathbf{G}$ green;
  - $\mathbf{C}$ cyan;
  - $\mathbf{R}$ red;
  - $\mathbf{M}$ magenta;
  - $\mathbf{Y}$ yellow;
  - $\mathbf{W}$ white;

Here are our eight colors, black, blue, green, cyan, red, magenta, yellow, white.



Although the world seems full of millions of colors, our eye is only sensitive to red (R), green (G) and blue (B) light. Color variation depends simply on the strength of these three different components of light.

We can imagine having red, green and blue lamps whose intensities vary from 0 to 1. We can make any color by varying these intensities, and so we can represent a color by its (R,G,B) values in a list.

The color black arises when there is no light at all, so it is (0,0,0). White corresponds to all three lights at full strength, (1,1,1).

We get the primary colors by turning just one light to full strength: red = (1,0,0), green = (0,1,0), and blue = (0,0,1).

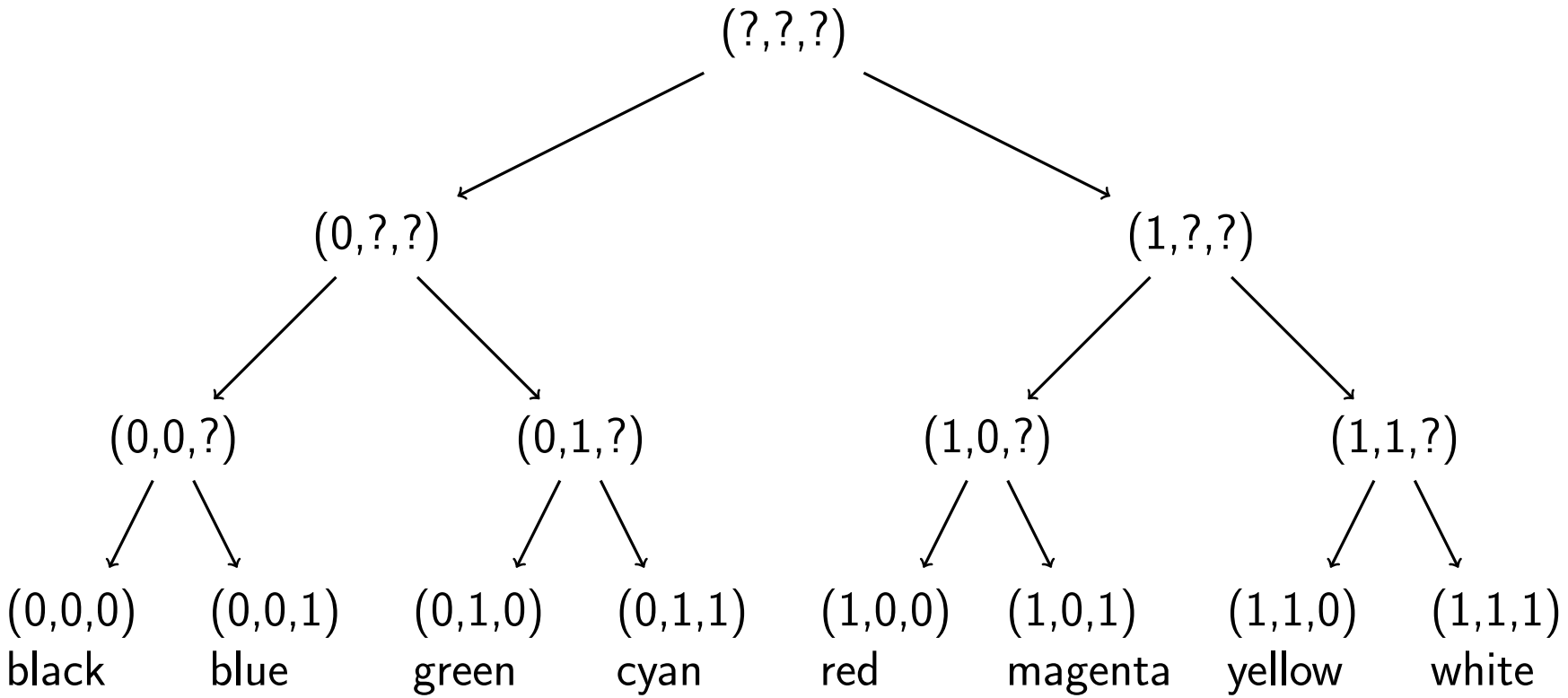
The other three colors in our set of 8 come from turning on two lights to full strength: cyan = (0,1,1), magenta = (1,0,1), yellow = (1,1,0).

Now we are ready to compare the amount of information in a black/white (KW) image and a KBGCRMYW image.

To determine the color of a pixel in a KW image, I only have to answer one question: is this pixel white?

For KBGCRMYW images, it seems like I have to ask 8 questions, is it black? is it blue? is it green?... and so on. But I can actually find the color in just 3 questions; because each color corresponds to the three numbers in the (R,G,B) description, and for our colors, these numbers can only be 0 (fully off) or 1 (fully on).

We can make this clear with a decision tree.



Q1: *Is the red component 1?*
Q2: *Is the green component 1?*
Q3: *Is the blue component 1?*

The color information in a (true) black and white picture requires answering one yes/no question; in an 8-color picture, we need 3 yes/no questions. This means that an 8-color picture is not 4 times as complex as a 2-color picture, but rather 3 times as complex.

Here is a table that suggests how many yes/no questions are needed in order to specify one color from a set of a given size:

Number of colors	Number of questions
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8

The computer stores the answer to a yes/no question as a "1" for yes, and an 0 for "no", and it calls this one bit of information.

For convenience, the computer stores bits in groups of 8, called a byte, and the byte is the basic size unit on the computer.

A character ('A', '2', '@') can be stored in one byte. The Gettysburg Address is about 1,470 bytes, or 1.4 $kilobytes$; An uncompressed color graphics file might be between 500,000 to 2,000,000 bytes, that is, around 1 $megabyte$. A good quality MP3 music file may be 3 to 10 megabytes. The text of a book, stored in PDF format, might run 25 to 200 megabytes. But really large files arise when we deal with movies, which can be on the order of 5 to 25 $gigabytes$, that is, 5 to 25 billion bytes. The computer itself typically has an active memory space of 16 to 32 gigabytes. The hard drive on a modern computer can be as large as 1 $terabyte$, that is, 1 trillion bytes.

A table of sizes:

Measurement	Size in bytes	things that size
1 bit	-	1 or 0
1 byte	1	a single character (= 8 bits)
1 kilobyte	1,000	a text file
1 megabyte	1,000,000	photos, music, PDF documents
1 gigabyte	1,000,000,000	movies, computer memory
1 terabyte	1,000,000,000,000	hard drives

On a computer, if we want to reduce the cost of storing a 256-color image, we won't see any improvement unless we cut the number of colors at least in half, that is, to 128. Further reductions in cost come by repeated halvings.

Surely, if we reduce the number of colors, we can't just throw away the pixels whose colors have been discarded. We have to color them something.

This raises two big questions:

- If a pixel has a discarded color, what color should it become?
- How should we choose the colors to discard?

The amount of information in a typical color computer image can be estimated as the product of the number of pixels times the number of choices for each color level.

The information associated with a 1,200,000 pixel image that is in black and white, using 256 levels of gray, will thus be on the order of 8 x 1,200,000 = 9,600,000.

The information for a color image of the same format might therefore be 3 x 8 x 1,200,000 = 28,800,000.

Because images are so useful, but can be so large, it is necessary to come up with ways of reducing the size of images so that we don't run out of computer storage space.

Original image

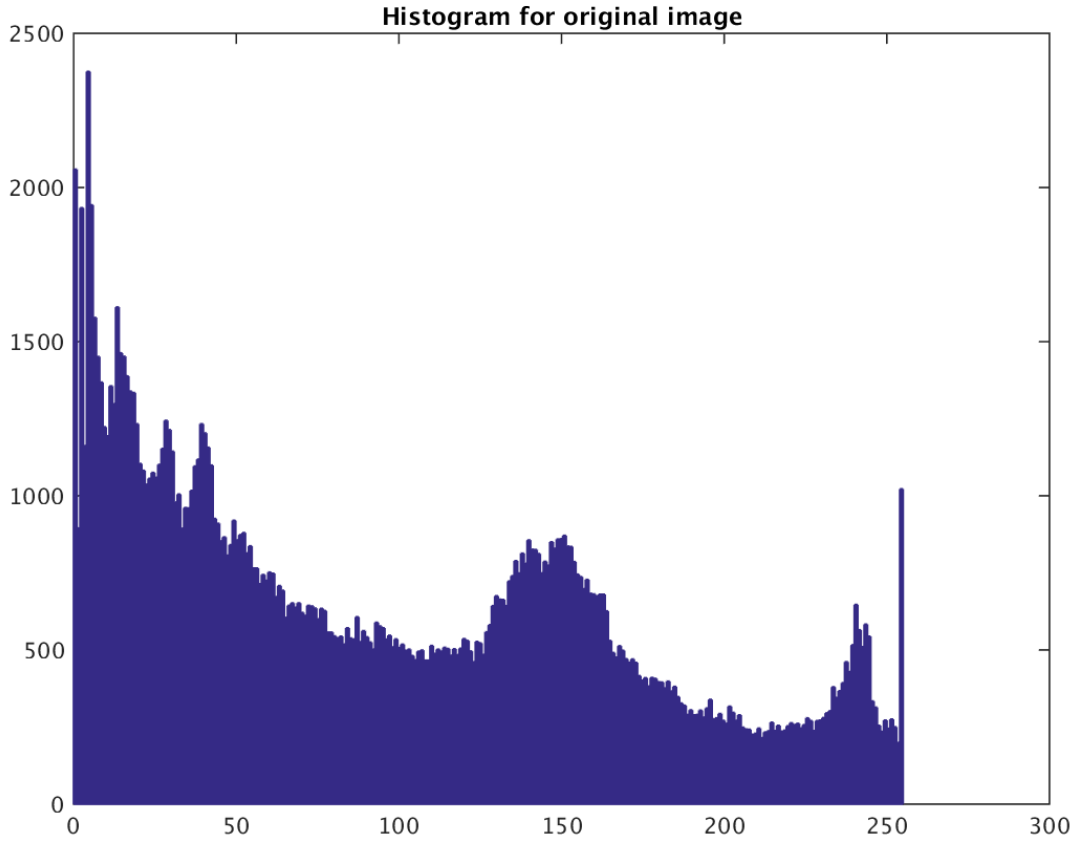
For simplicity, we will look at what we call a black and white photograph, although it's actually a gray scale image. There are 256 levels of grayness, from 0 (black) to 127 (medium gray) to 255 (white). We know that it takes 8 yes/no questions to specify any specify shade of gray, and that means that the computer essentially stores 8 YES or NO's for each pixel of this picture.

If we want to reduce the storage necessary for this picture, we could try throwing away some rows and columns of pixels, or looking for simple patterns in the colors (what JPEG does), or we can try reducing the number of shades of gray.

If we reduce the number of shades of gray, then we have to recolor many pixels. It makes sense to recolor them using the nearest shade of gray that we still have. And this means that we can try the clustering approach.

Clustering approach for image size reduction

1. Start with a lot of data (shades of gray).
2. Organize the shades of gray into groups of about the same size.
3. Pick an average member of each group.
4. Replace every member of the group by the average.


Histogram for original image
2500
2000
1500
1000
500
0
0
50
100
150
200
250
300

The histogram shows how many times each shade of gray, between 0 and 255, occurs in our image. You can see there's a lot of black, a slow decrease as we move towards the right, with the interruption of a noticeable bump around the gray shade 150.

You might guess that we could simply divide the shades of gray by splitting the interval from 0 to 255 into equal ranges of value. However, this would mean that the lower ranges, which correspond to dark grays, would have many more pixels than would the upper ranges. So in fact, the ranges will be adjusted to more evenly spread out the pixels.

In the following steps, we will reduce the number of colors from 256 to 16, then 4, and then 2. In each case, we will also list the values of shades of gray that we use, and you should notice the tendency to favor the darker shades.

Original image

256 Grays: 0, 1, 2, 3, ..., 252, 253, 254, 255

Image using 16 shades

16 Grays: 5,18,32,46,59,72,85,99,115,133,148,165,184,206,228,246

4 Grays: 22, 78, 147, 223

2 Grays: 41, 170


The Plumb-pudding in danger ;— or State Epicures taking an Petit Souper.
"the great Globe itself, and all which it inherit" is too small to satisfy such insatiable appetites
OCEAN
EUROPE

Can we apply clustering to more complicated problems?

The territory of the United States, for example, has been divided into clusters, called states.

The sizes and shapes of these clusters has been determined by history, politics, economics, and geography, but now that the arrangement has been made, almost no one wonders how it got that way, or whether it could be "improved".

But let's explore what would happen if we were allowed to alter the current state system.

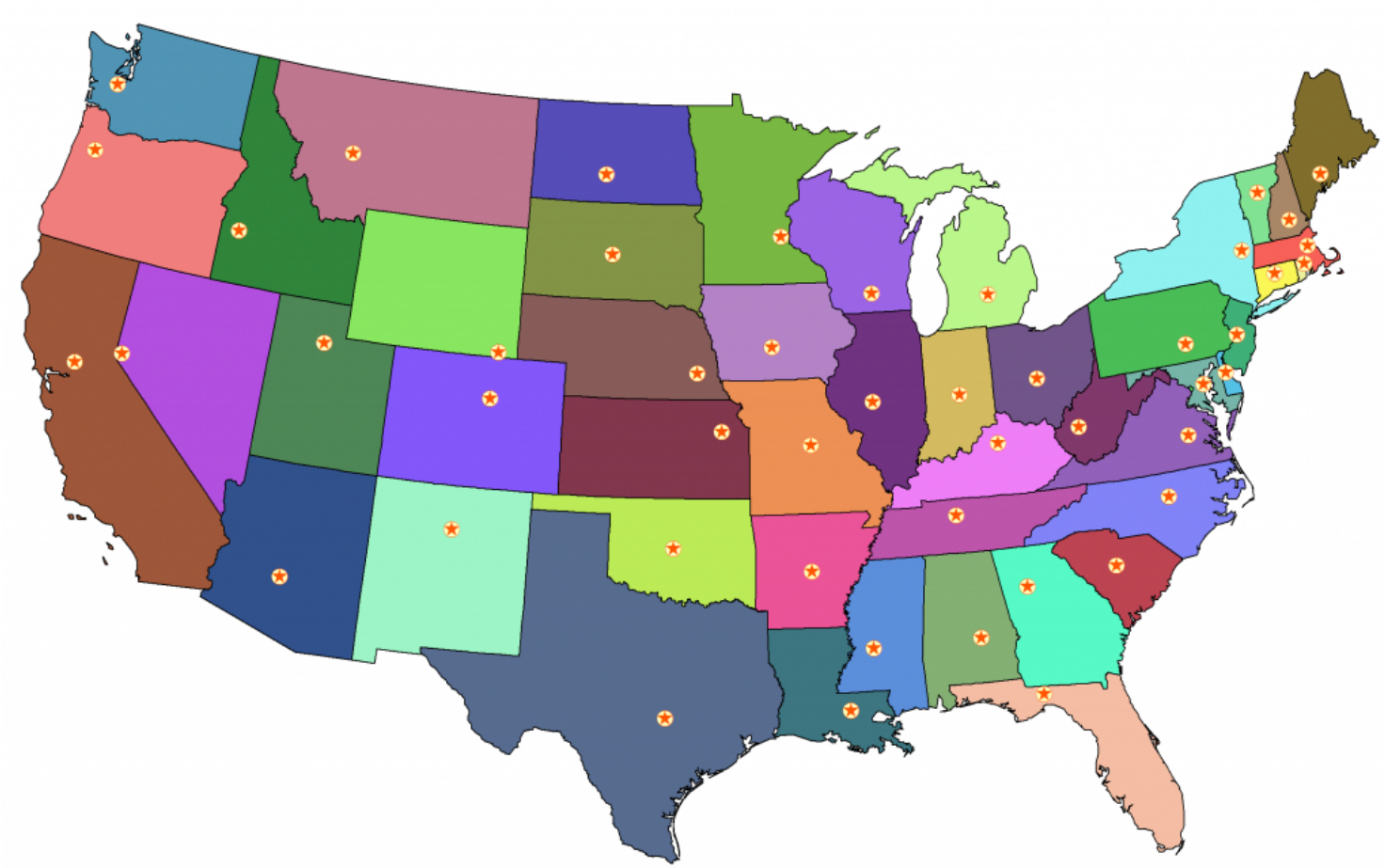
We have to decide what's most important to us, for example, equality of population, or area? Shapes that are compact, rectangular, or following local features like rivers and mountains?

To get an unbiased perspective, we will ask an advisor from another

planet!

So let's suppose a Martian is visiting the United States, and wants to visit every state, and so we take the Martian to each state capital. Then the Martian looks at the map of the US, and says:

> *"This is very strange. The capital of Florida is almost in Georgia. Most of the people in Florida travel very far to reach their capital. And many people in Georgia are closer to Tallahassee than to Atlanta. Isn't this wrong?"*



We can explain to the Martian that the placement of US capital cities is completely accidental. For instance, Florida was originally two pieces, West Florida and East Florida, and it only had two major cities, Pensacola and Jacksonville, so it seemed like a reasonable compromise to put the capital in the "center" of the state! (No one expected people to start moving into the hot swampy lands to the south.)

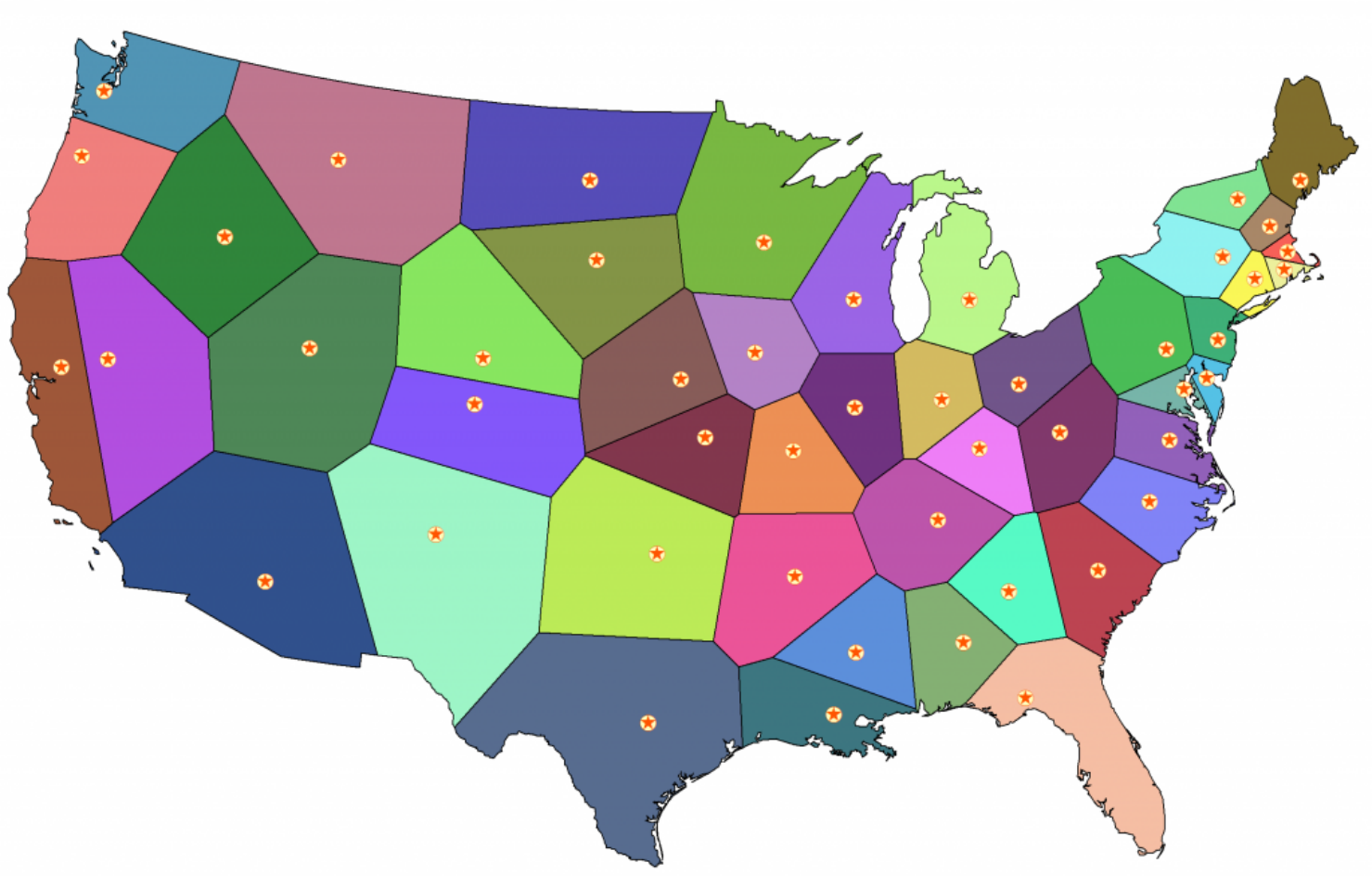
The shape of the US states is also an accident, decided by politicians, and now we just live with the results.

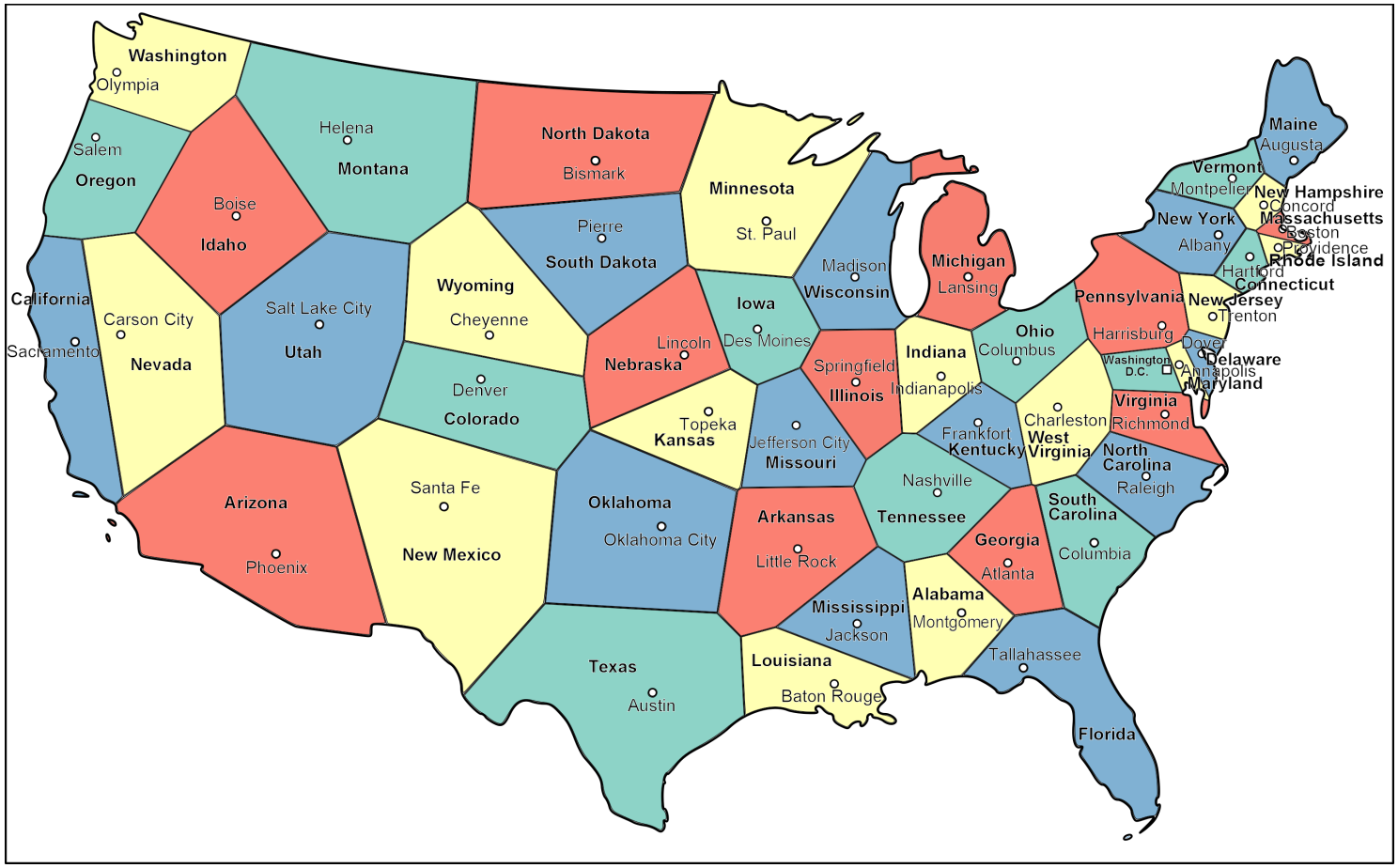
The Martian says:

> *"Well, I can see from visiting your many state capitals that they are not easy to move, with all those huge capitol buildings. But changing the shape of the state looks easy. It's just redrawing a line and moving a lot of signs. Suppose everyone belongs to the state with the nearest capital city? That automatically changes the shapes of the states, right?"*

What the Martian is suggesting is that the new Florida would be all the places in the US for which Tallahassee is the closest state capital. We may lose Pensacola, for instance, but we keep all of South Florida, and we probably can grab a chunk of Georgia as well.

What happens to the rest of the country is probably more complicated!




Washington
Olympia
Salem
Oregon
California
Sacramento
Carson City
Nevada
Boise
Idaho
Helena
Montana
Salt Lake City
Utah
Wyoming
Cheyenne
Arizona
Phoenix
Santa Fe
New Mexico
Denver
Colorado
North Dakota
Bismark
Pierre
South Dakota
Nebraska
Lincoln
Topeka
Kansas
Oklahoma
Oklahoma City
Texas
Austin
Minnesota
St. Paul
Iowa
Des Moines
Missouri
Jefferson City
Arkansas
Little Rock
Louisiana
Baton Rouge
Wisconsin
Madison
Illinois
Springfield
Michigan
Lansing
Indiana
Indianapolis
Ohio
Columbus
Kentucky
Frankfort
Tennessee
Nashville
Mississippi
Jackson
Alabama
Montgomery
Georgia
Atlanta
Florida
Tallahassee
South Carolina
Columbia
North Carolina
Raleigh
Virginia
Richmond
West Virginia
Charleston
Pennsylvania
Harrisburg
New York
Albany
Maine
Augusta
Vermont
Montpelier
New Hampshire
Concord
Massachusetts
Boston
Rhode Island
Providence
Connecticut
Hartford
New Jersey
Trenton
Delaware
Dover
Maryland
Annapolis
Washington D.C.

Our new map has some interesting properties:

- Except on the coastline and international borders, all the states have straight line borders.
- The states are more compact (packed together, like squares or circles)
- No state has a "notch" in it. (We say the states are "convex".) So if you drive from point A to point B (within the state) in a straight line, you never leave the state.
- The states have lost their pandhandles (Idaho, Florida, Oklahoma, Texas)
- States tend to meet at triple corners.
- Washington DC is now much bigger than Maryland.
- If two state capitols are close, then the border is a straight line running halfway between them.

Is this an example of clustering?

Of course! Clustering involves taking a set of objects and assigning them to groups. In this example, each bit of land in the US has been assigned to a "state".

Isn't this different from the examples we did before? Yes! Before, we had a small collection of objects (cities, chain letters, or shades of gray in a picture) and we could imagine clustering as briefly picking up each object and tossing it into the appropriate basket. But if we consider every bit of land in the US, that's a lot more items than we can count or consider one at a time.

So how can we cluster so much data? We have to use some ideas from geometry.

To see how this might go, imagine starting with just two capital cities, A and B.

If they are 100 miles apart, then the break between the two states has to occur midway on the line between them. You can also see that the state border line will extend in the direction perpendicular to this line.

Adding a third capital city C, we have to imagine drawing the lines A-C and then B-C, and correctly trimming the overlap. The slicing we do at each step is the reason that the states will have straight line borders.

No one is going to allow us to redefine the shapes of the states, but this example illustrates that the idea of clustering can be extended from small collections of objects to extensive geometric shapes.

This geometric clustering idea is so common, computationally useful and mathematically interesting that is has its own name, the Voronoi diagram.

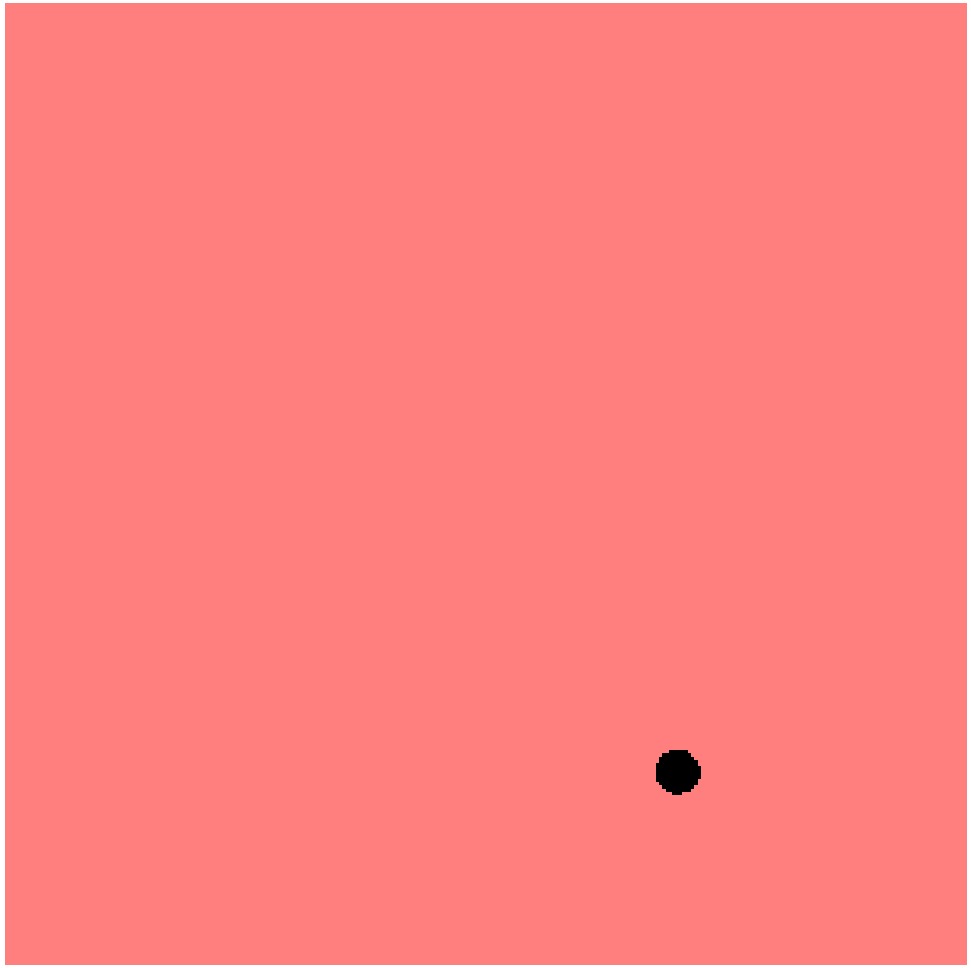
In a typical Voronoi diagram calculation, we are given a geometric region, a set of special locations within the region, and must create subregions of points closest to each special location.

There are a number of practical problems that can be solved with this kind of clustering, and even in Nature, there are examples of this kind of way of organizing space into regions.

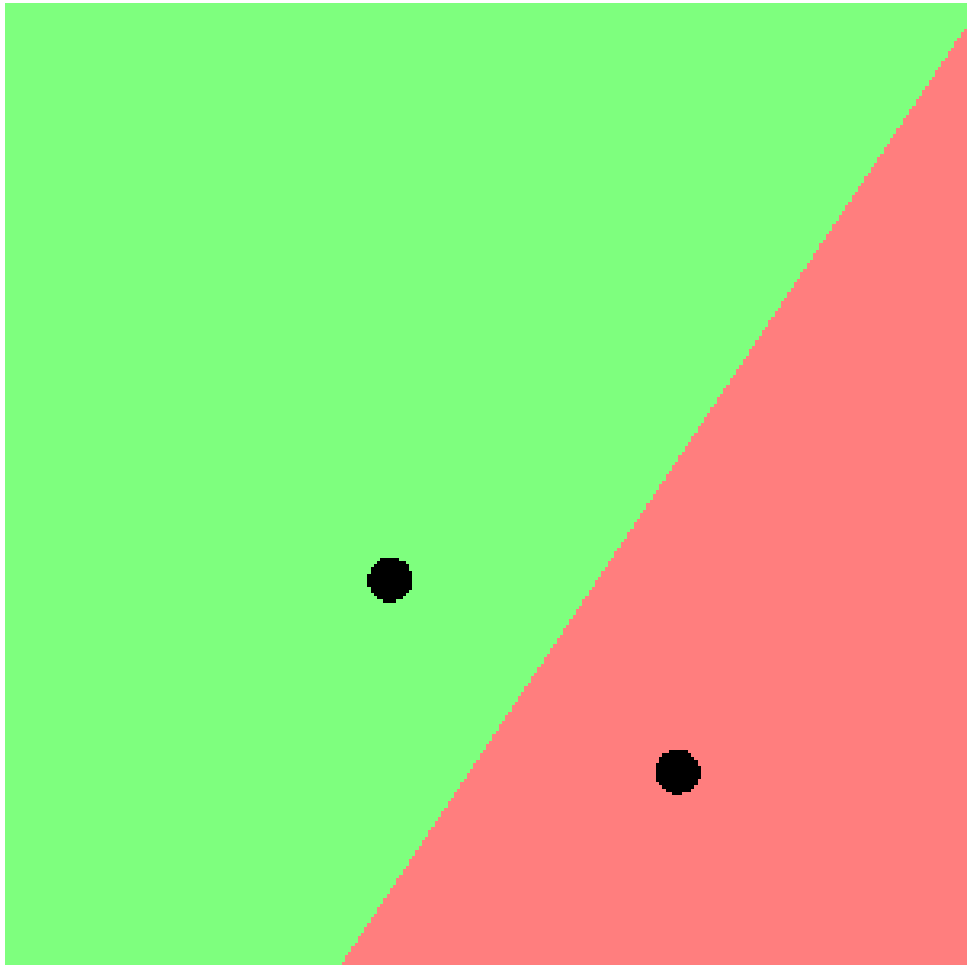
To understand a little better how a Voronoi clustering works, let's simplify our problem. Our region will no longer be the US, but just a square. Instead of 50 state capitals, we'll have no more than 5 special points, and we'll just call them "centers".

It turns out that if we want to understand a clustering involving 5 center points, we can think about this by adding one center at a time.

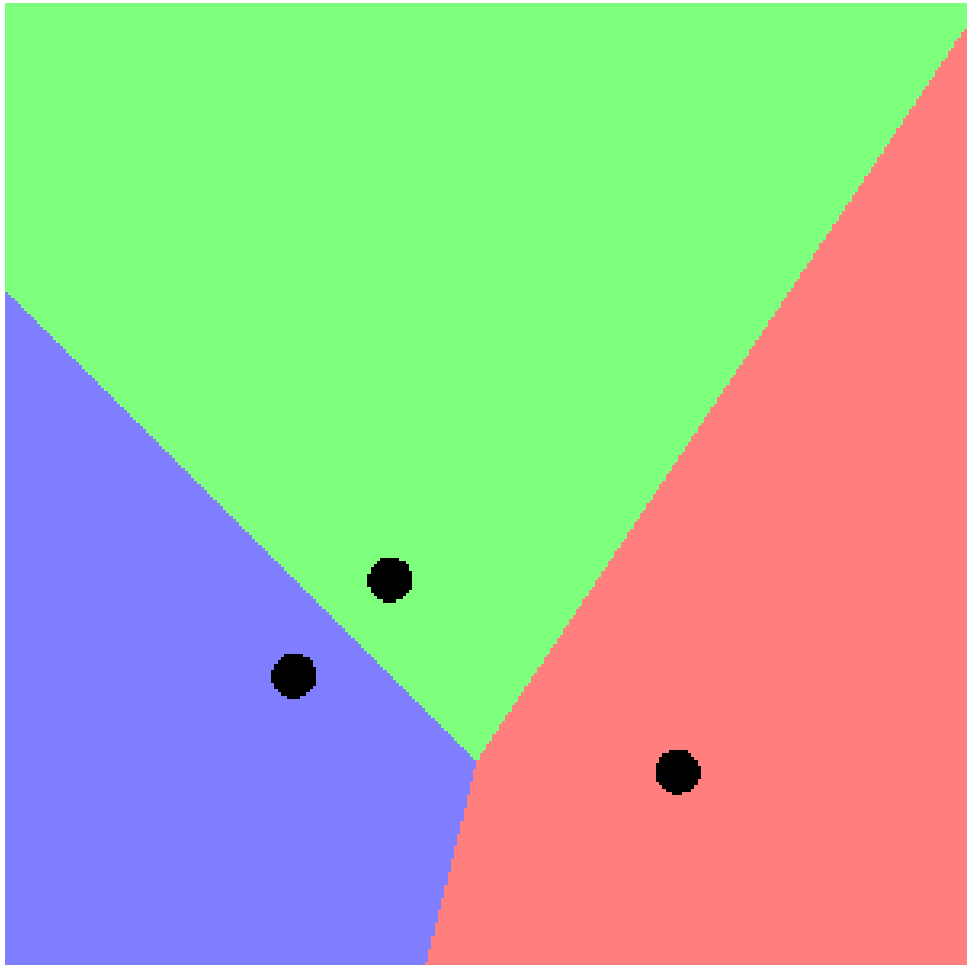
One center, so everything belongs to the “light red” state.



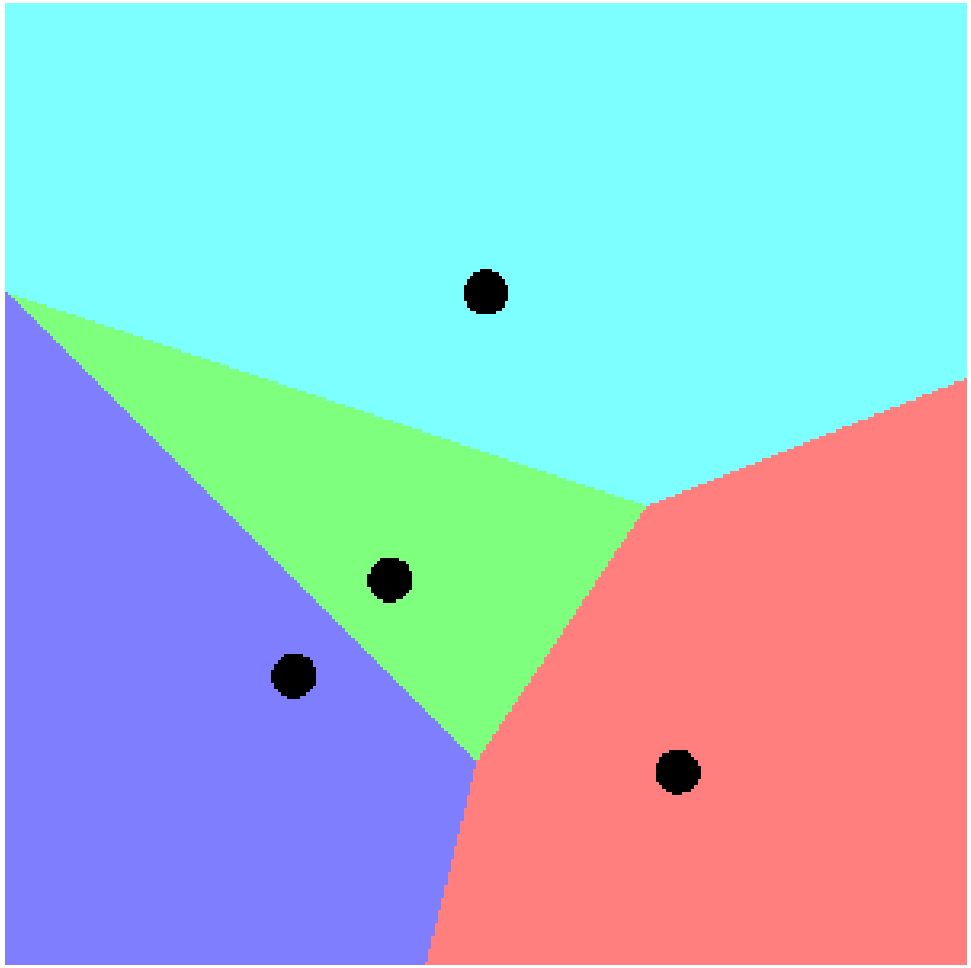
Add a green state. The dividing line runs between the two centers.



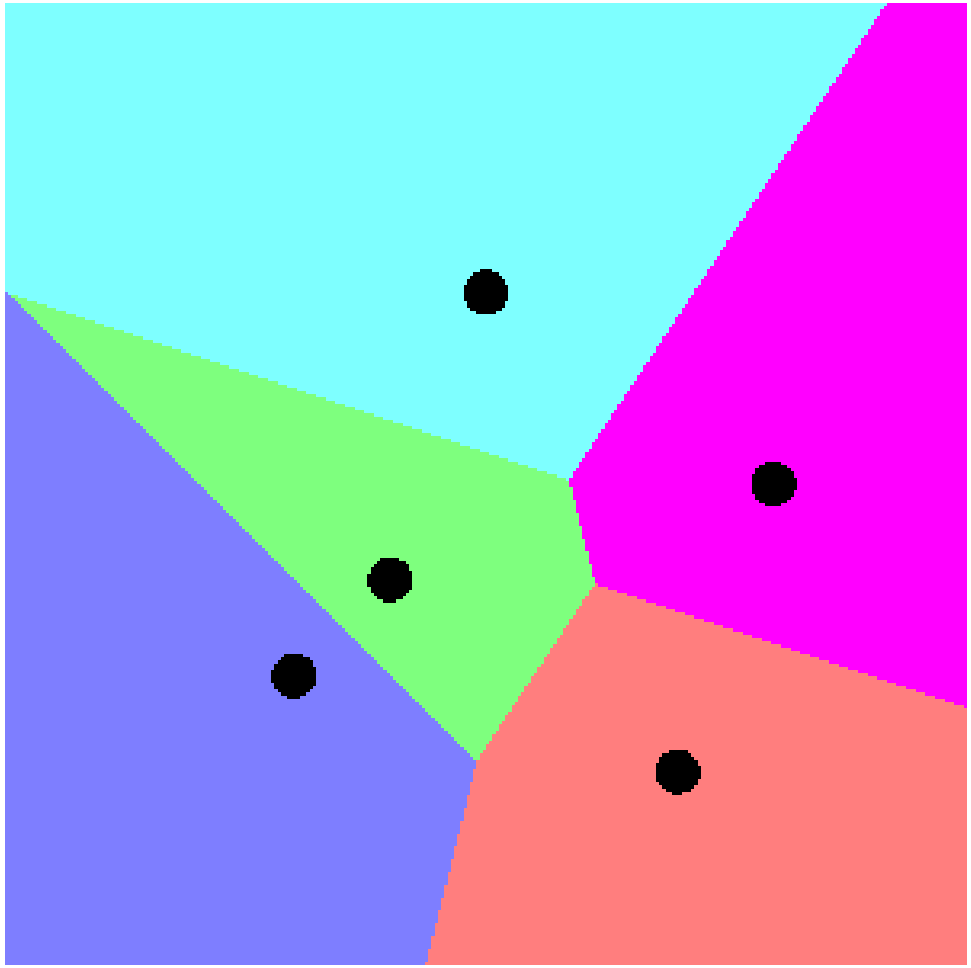
Add a blue state. We slice off green and light red territory.



Add a light blue state. The green state is tiny and surrounded.



The final purple state is added.



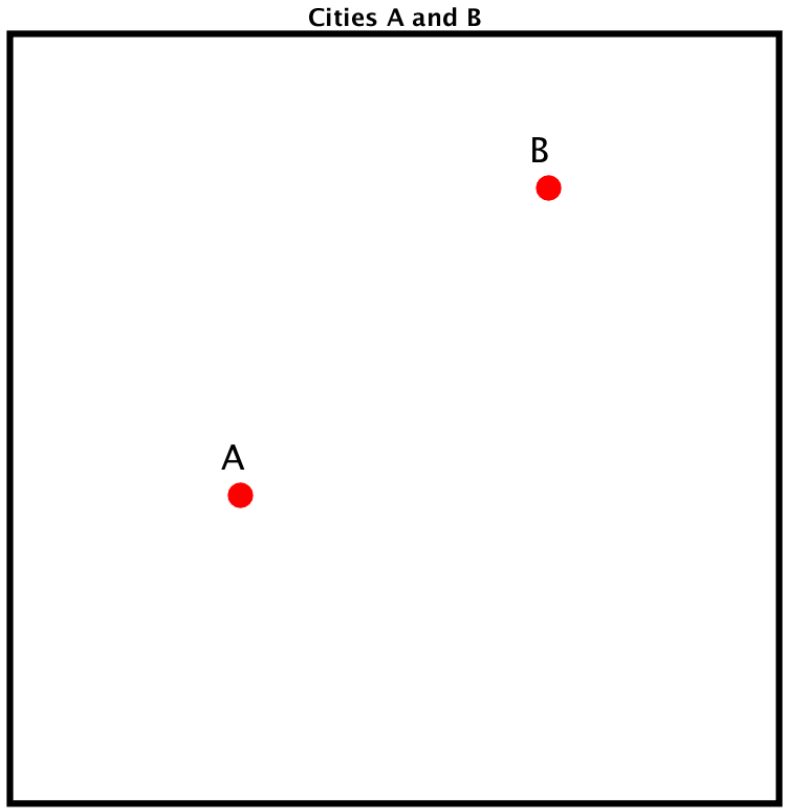
We have watched a map develop as we added one point at at time, without really thinking about why it takes on this shape.

There is actually a very logical process going on, which we can follow.

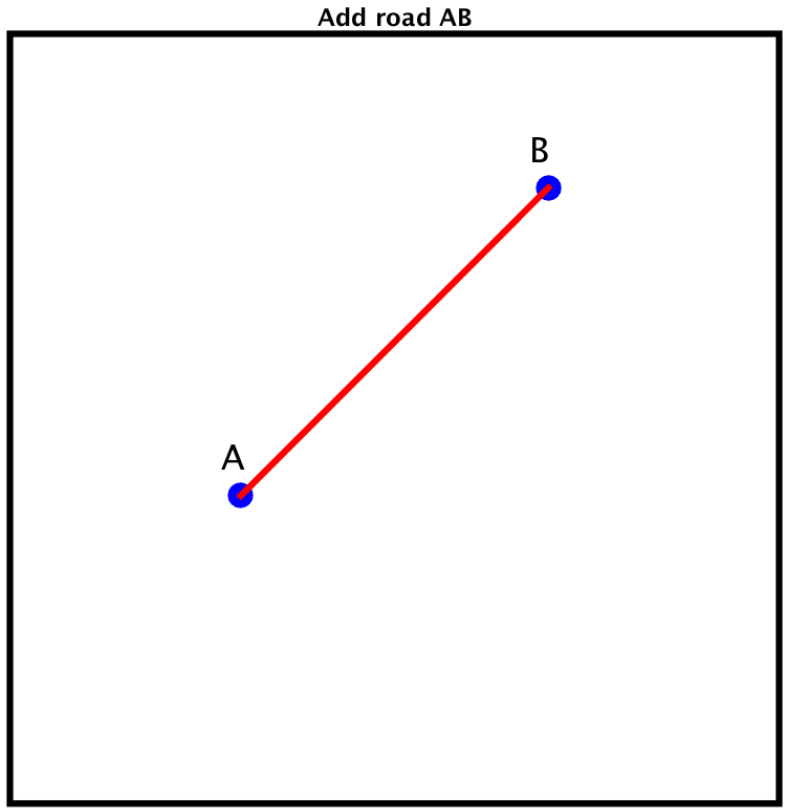
This time, we'll suppose that there are three cities, Atlanta, Baltimore and Charlotte, and we're going to build roads connecting them. Of course, responsibility for maintaining the road between Atlanta and Baltimore will be divided right at the halfway point on the road.

Then it also makes sense for each city to maintain the surrounding countryside that is nearest to it.

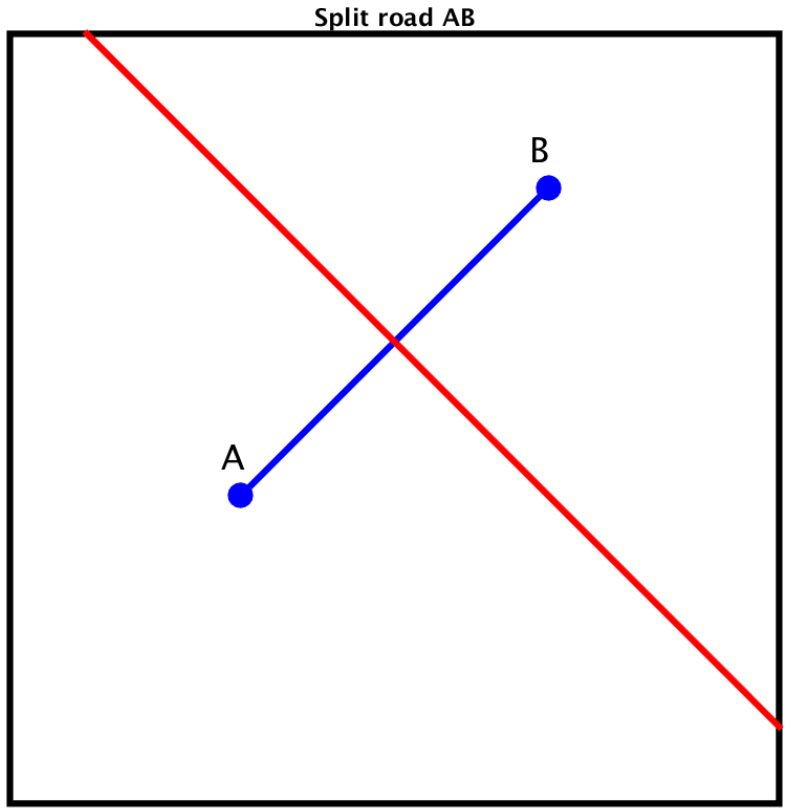
Let's watch step by step as these ideas create a map.



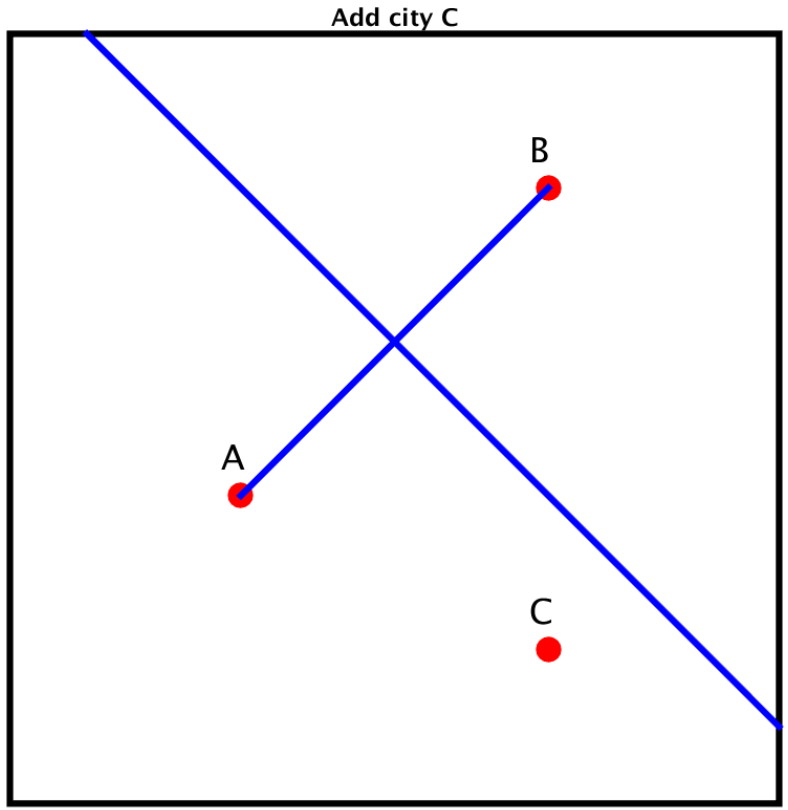
Let's start with just Atlanta and Baltimore.



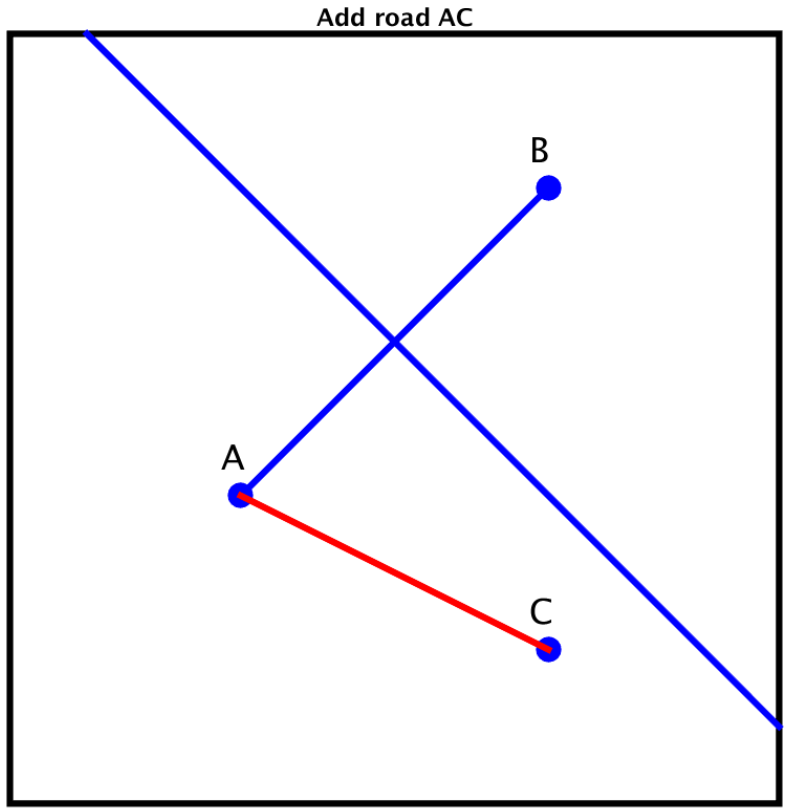
We build the road between them.



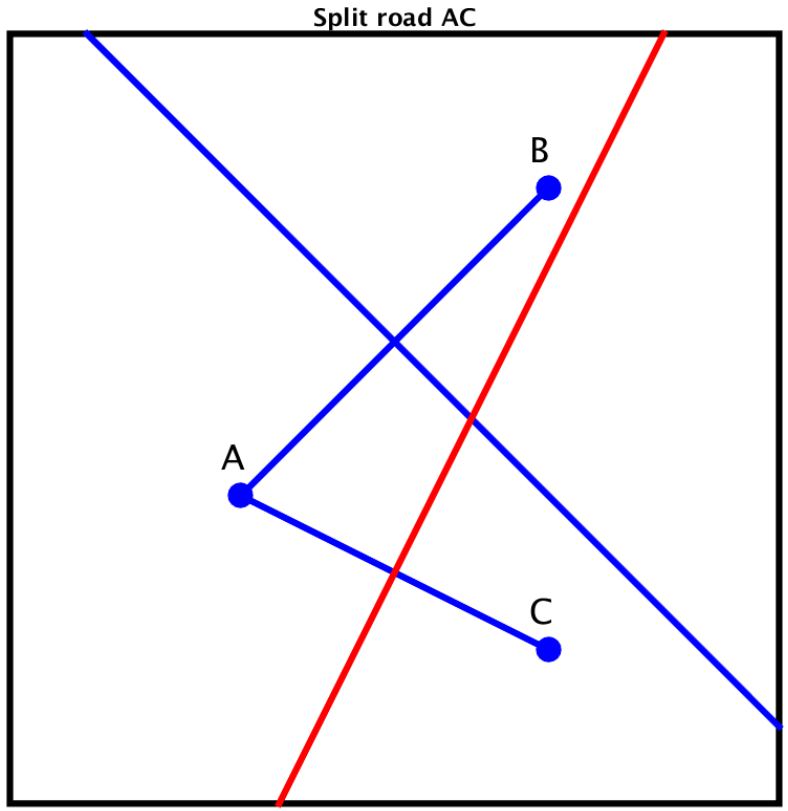
The red line splits the road, and also the countryside.



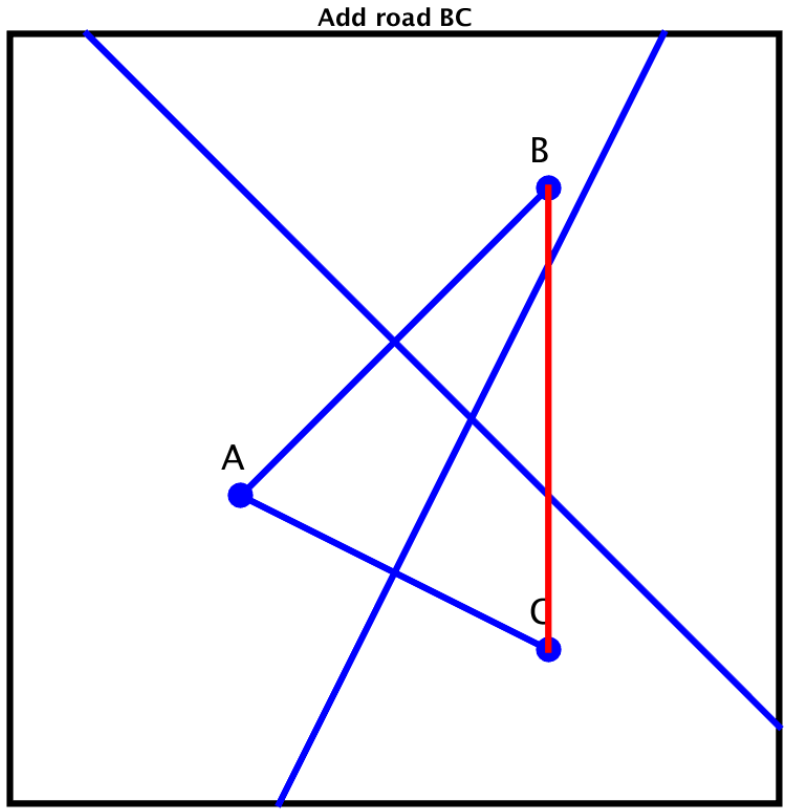
Now we realize that Charlotte is also on the map.



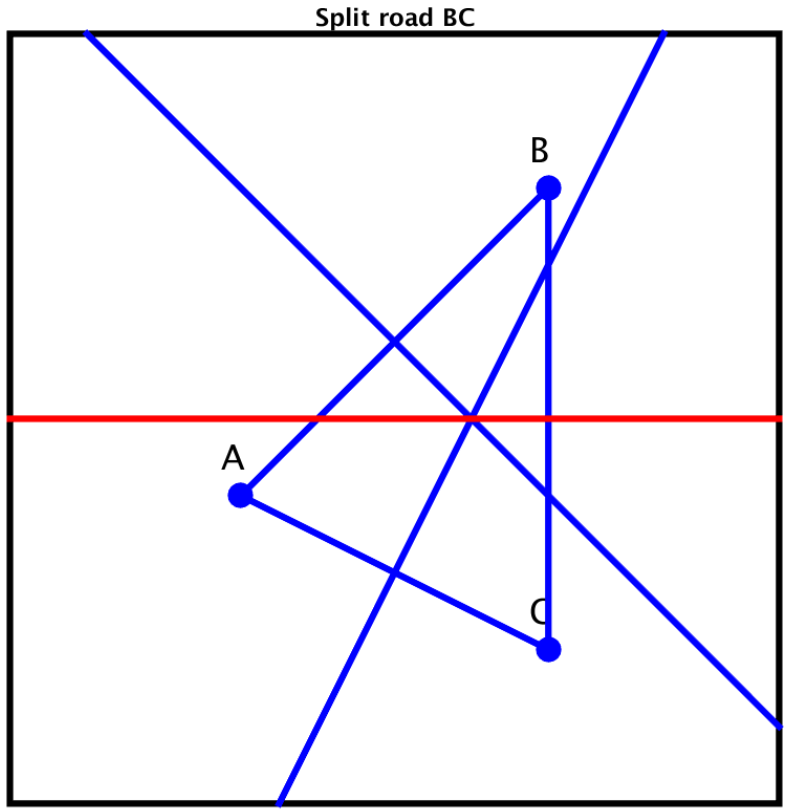
Build the Atlanta-Charlotte road.



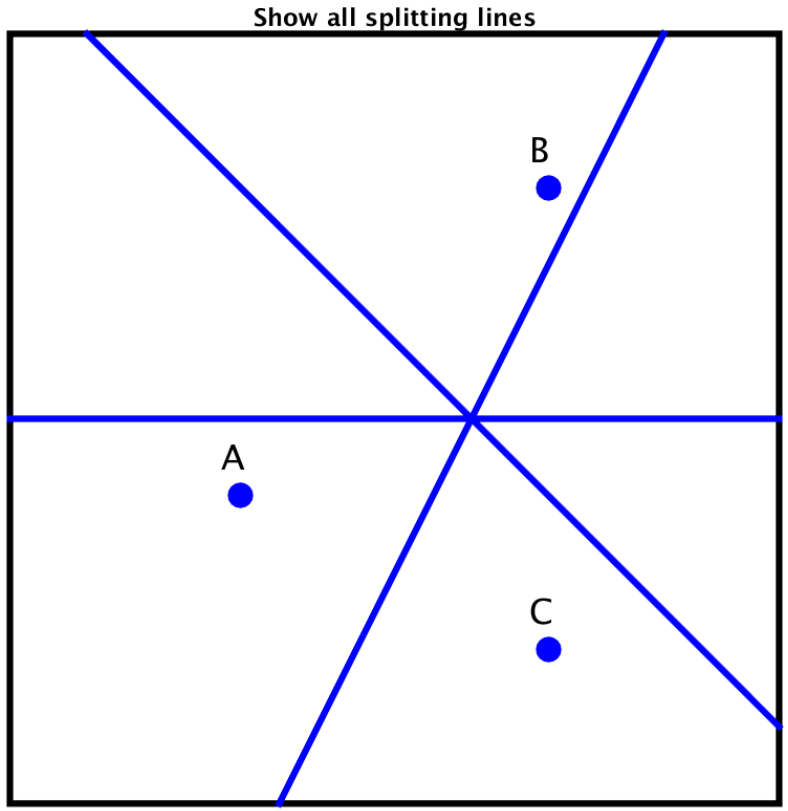
Split the Atlanta-Charlotte road and the countryside.



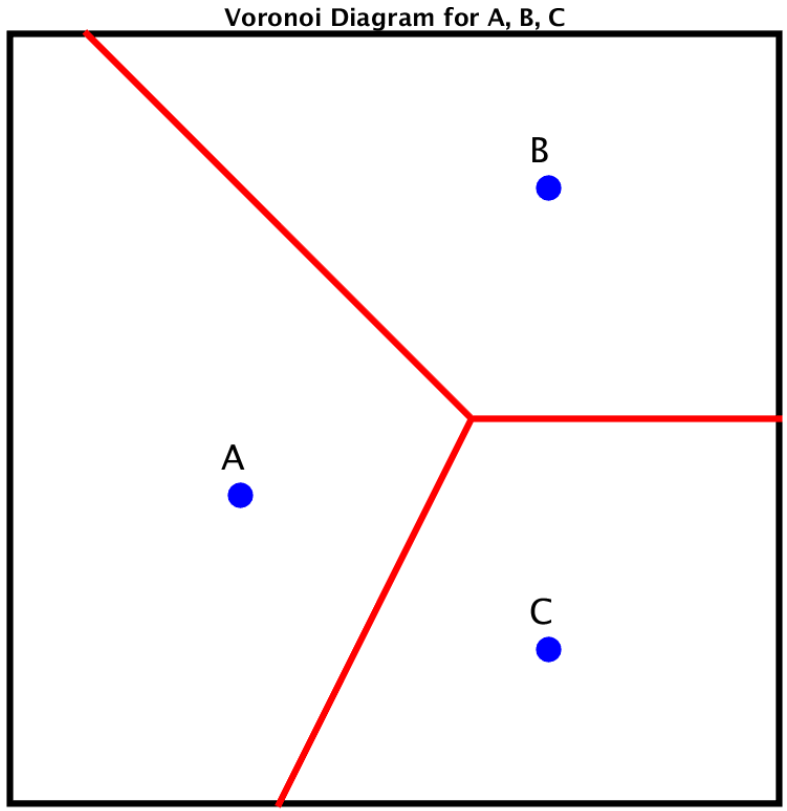
Build the Baltimore-Charlotte road.



Split the Baltimore-Charlotte road.



Splitting lines say which pair (A,B), (B,C) or (C,A) is closer.



We can use closeness pairs to draw the map.

Once we draw the splitting line between A and B, every point knows where A or B is closer.

Once we draw all the splitting lines, a point knows, given any pair of cities, which one is closer.

That's enough for it to know which one is closest, and so which territory it belongs to.

Drawing the splitting line between any two cities is easy. This divides the whole area into many pieces. The territories are put together by assembling the pieces, and so we just have to pick one point in each piece and ask "which city is closest to you?"

The Voronoi diagram provides a way of arranging points in space so that each point is associated with the nearest "center".

This organization by distance automatically clusters the points in a natural way, and the shape of the resulting clusters has some nice properties (straight edges, compact, convex).

We motivated the formation of the clusters by suggesting that in this way, each point has "chosen" its center by picking the one that is the closest, that is, which minimizes the distance.

So, as long as we assume the centers are chosen in advance for us, we've really solved this problem.

But if we are **really, really** concerned about minimizing distances in our map, we might be able to do better.

But this is only possible if the centers are movable. To make this more believable, suppose the centers represent mailboxes in a square town. If the mailbox positions have been decided, then our Voronoi diagram shows how the city is automatically divided up into zones, assuming every person would always choose the nearest mailbox.

So let's go back to our map of the 5 points in the square, and assume the old postmaster is retiring, and proudly hands this map to the new postmaster, who looks at it and says

> *This isn't very good. The dark blue region has its mailbox way off center. I bet by moving the mailbox to the center of its region, it would greatly reduce the walking distance for most customers, (although it would somewhat increase the distance for a few.*

Let's start with our five mailboxes in their somewhat arbitrary positions.

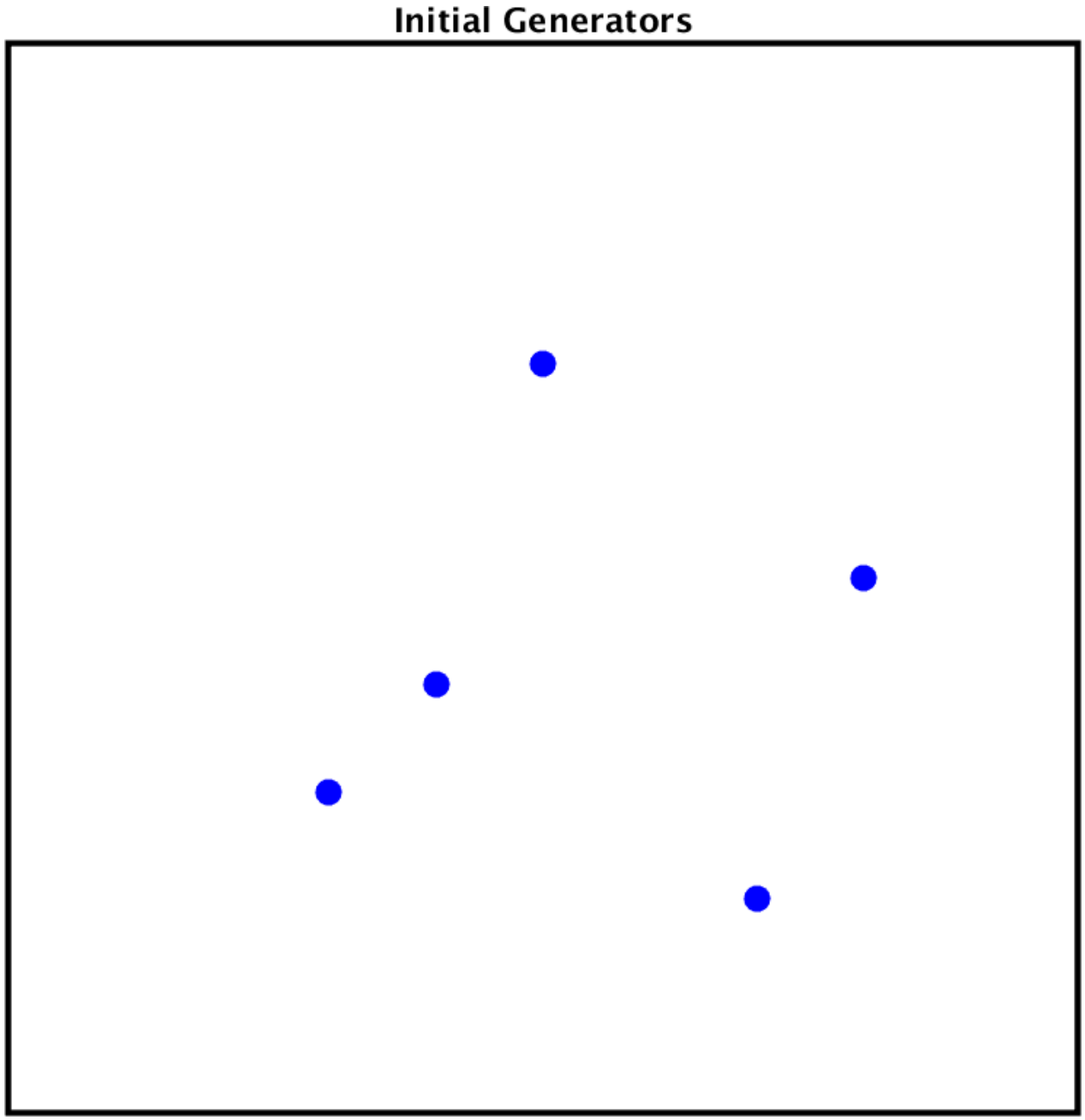
Then we will draw the corresponding "nearest mailbox postal zones".

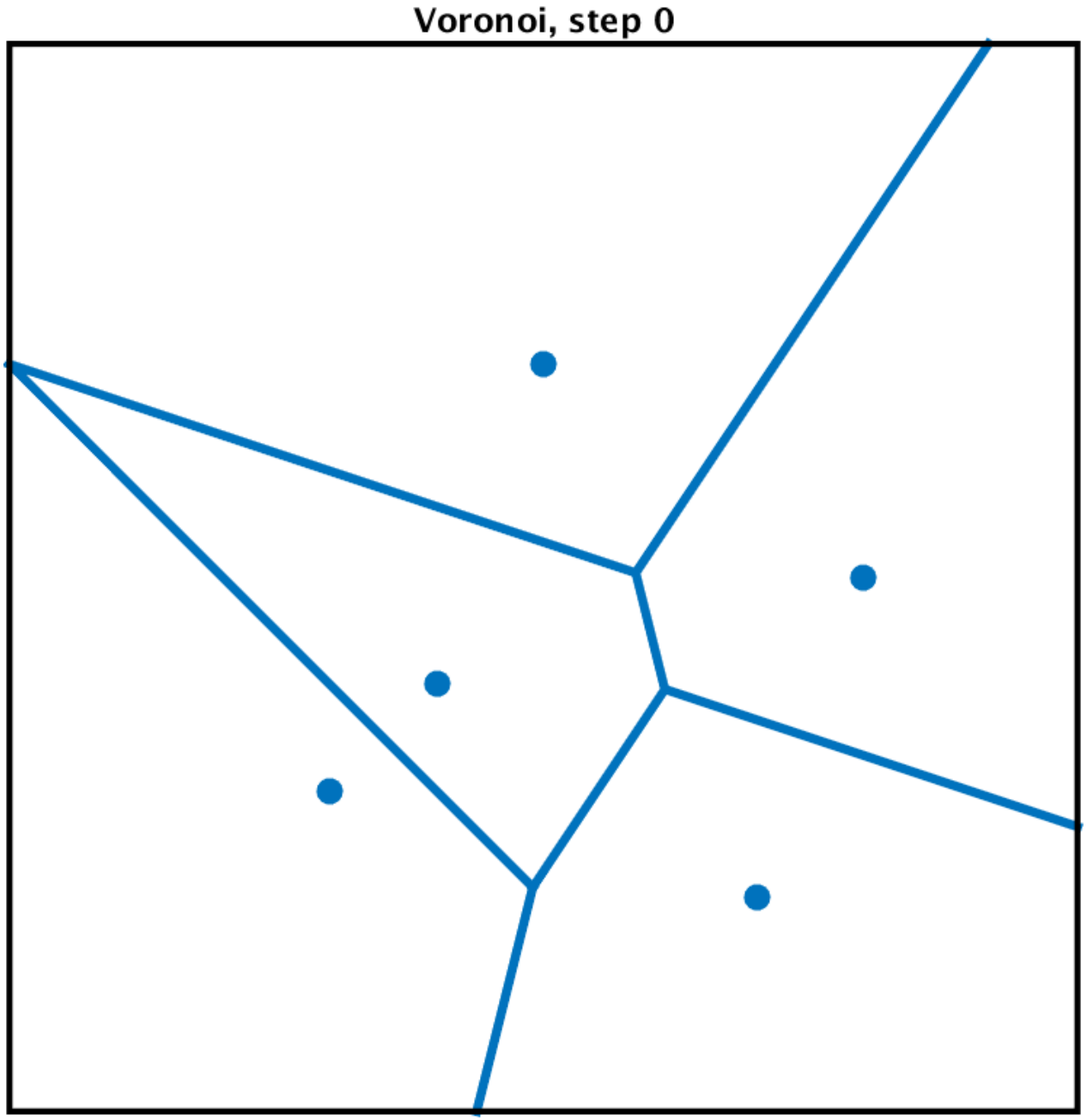
Then we will move the mailboxes to the center of the zones.

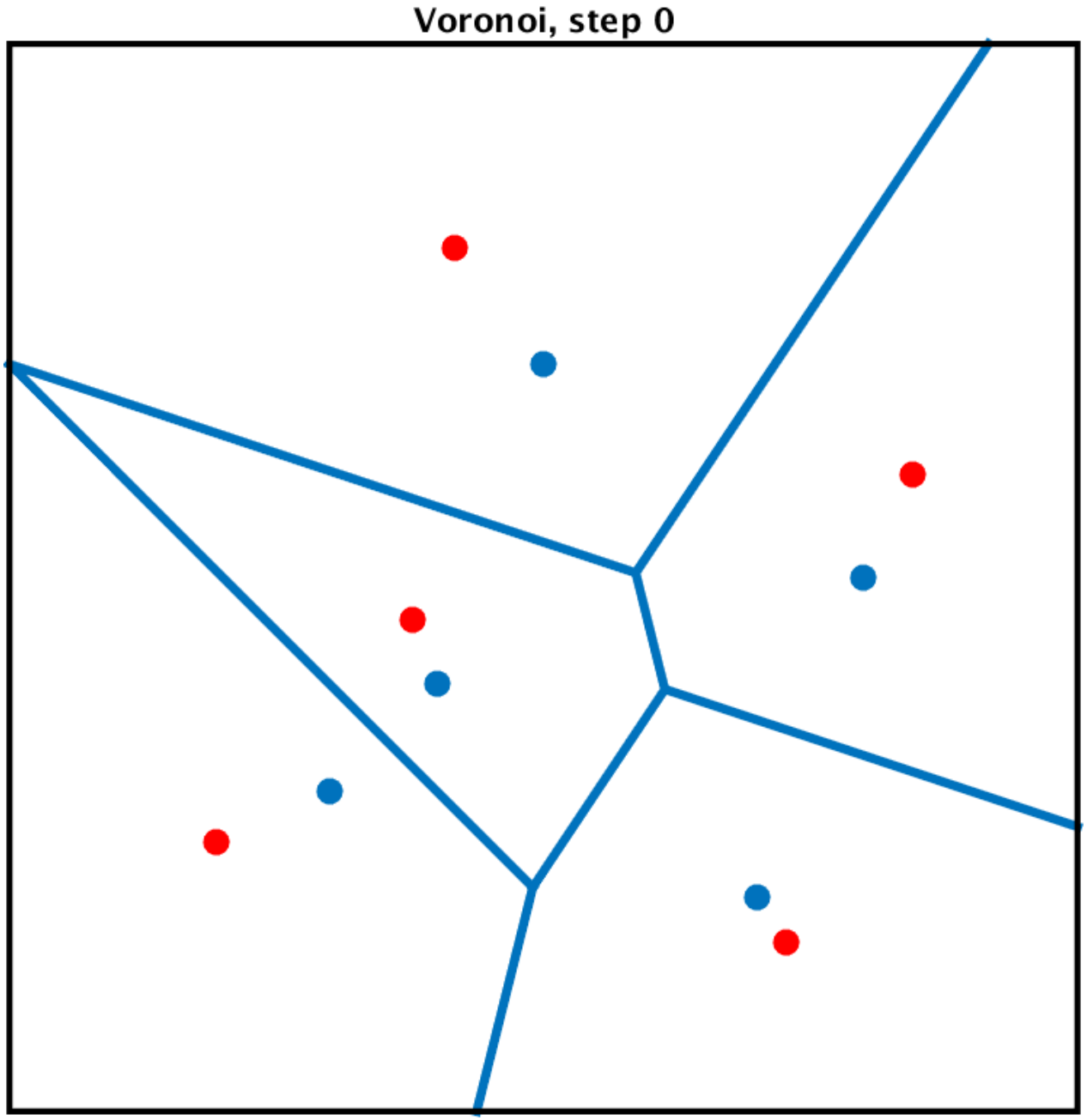
Then we'll redraw the zones.

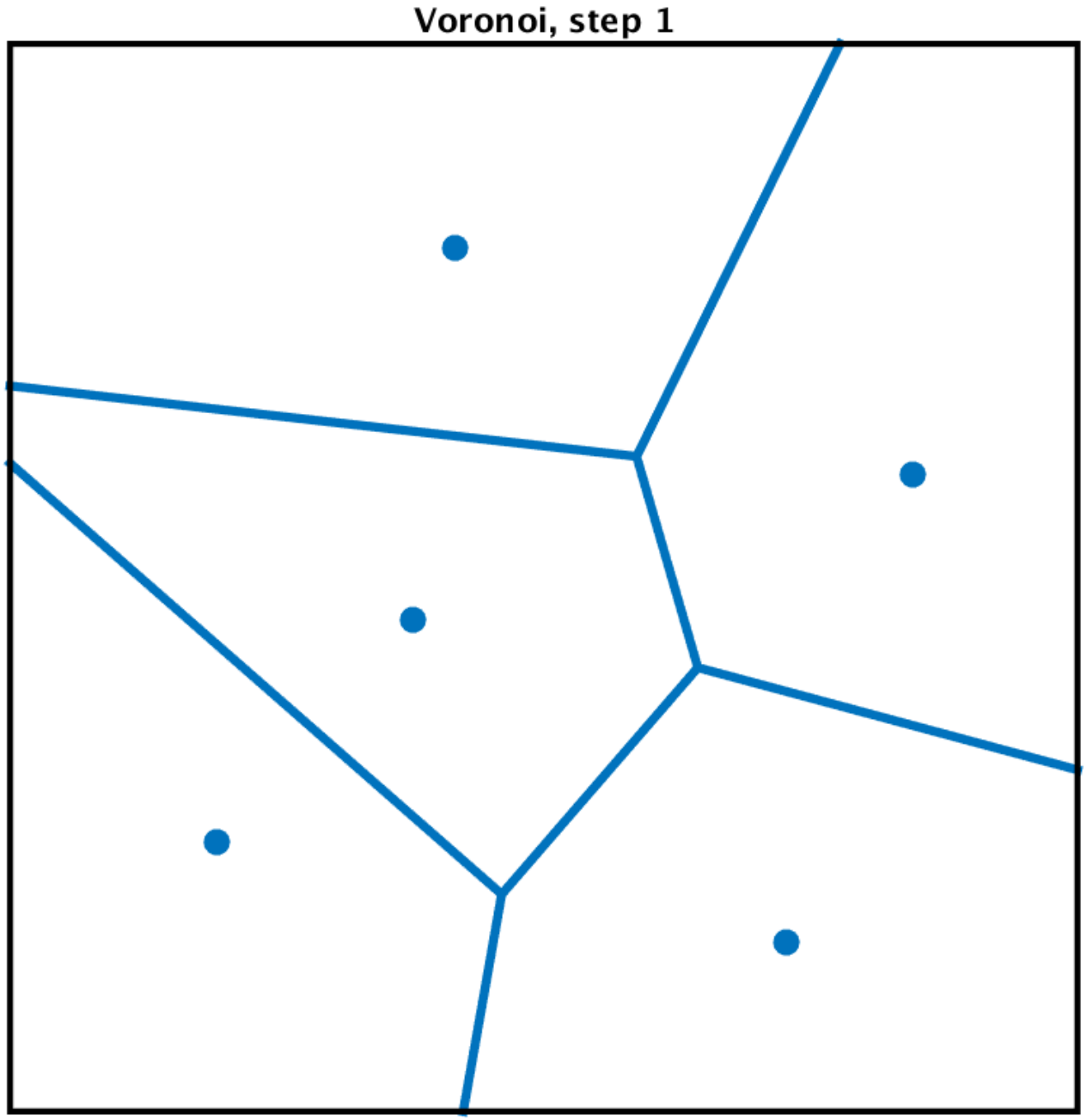
Then move the mailboxes again.

...and keep doing this til things seem to settle down.


Initial Generators


Voronoi, step 0


Voronoi, step 0

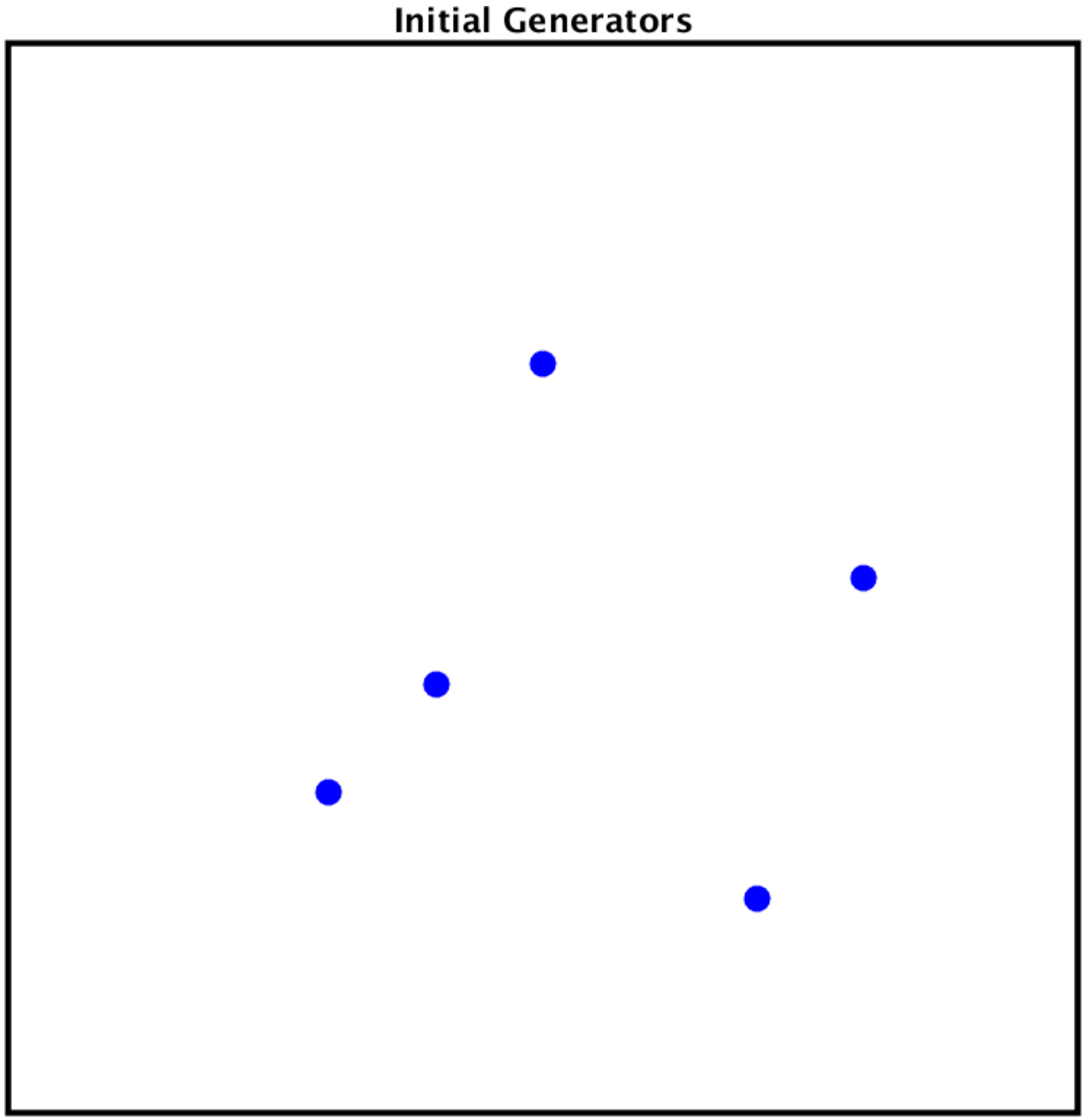

Voronoi, step 1

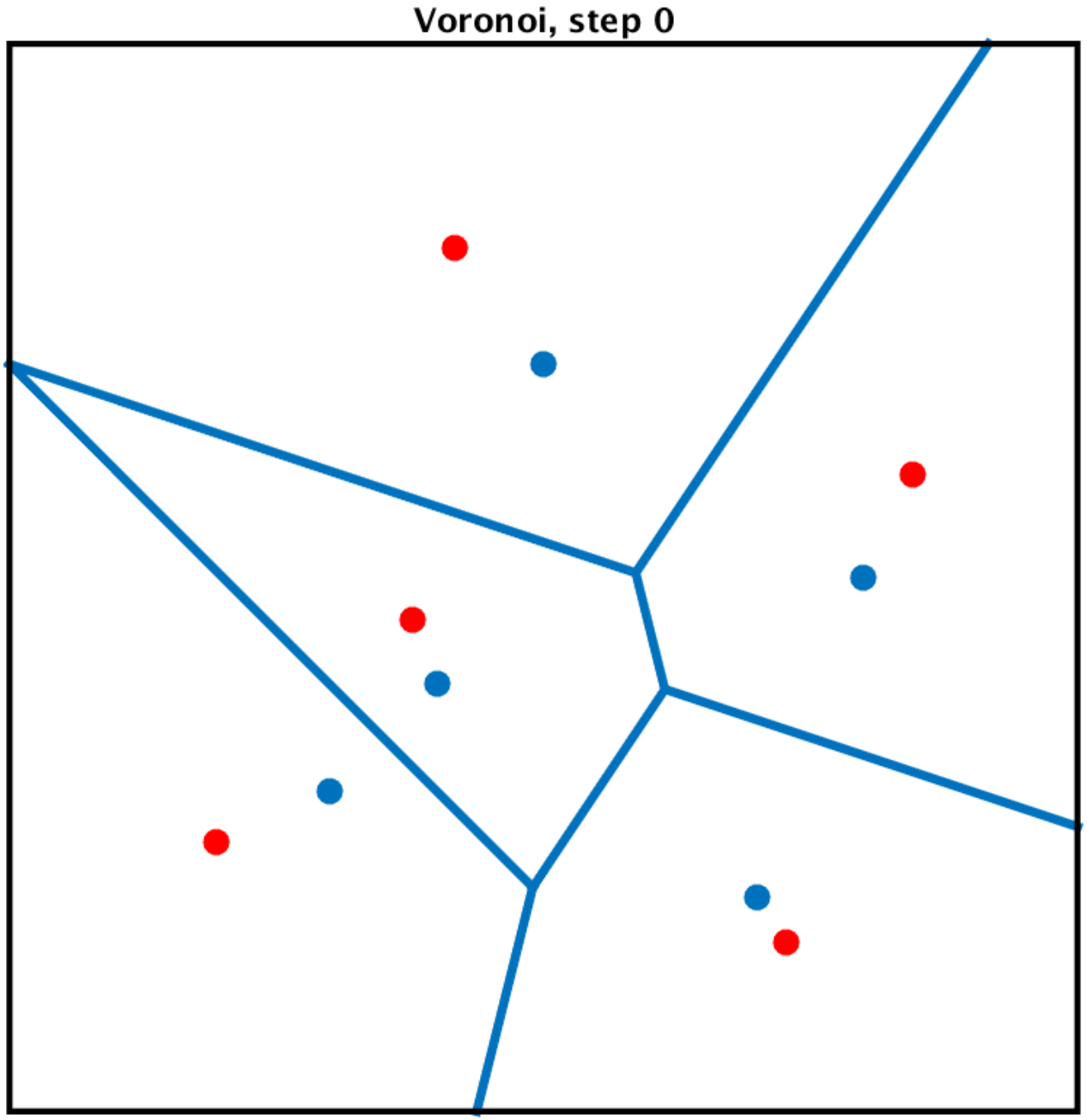
By moving the mailboxes to the centers of their territories, we have, indeed, reduced the **average** distance between any mail box and all its customers.

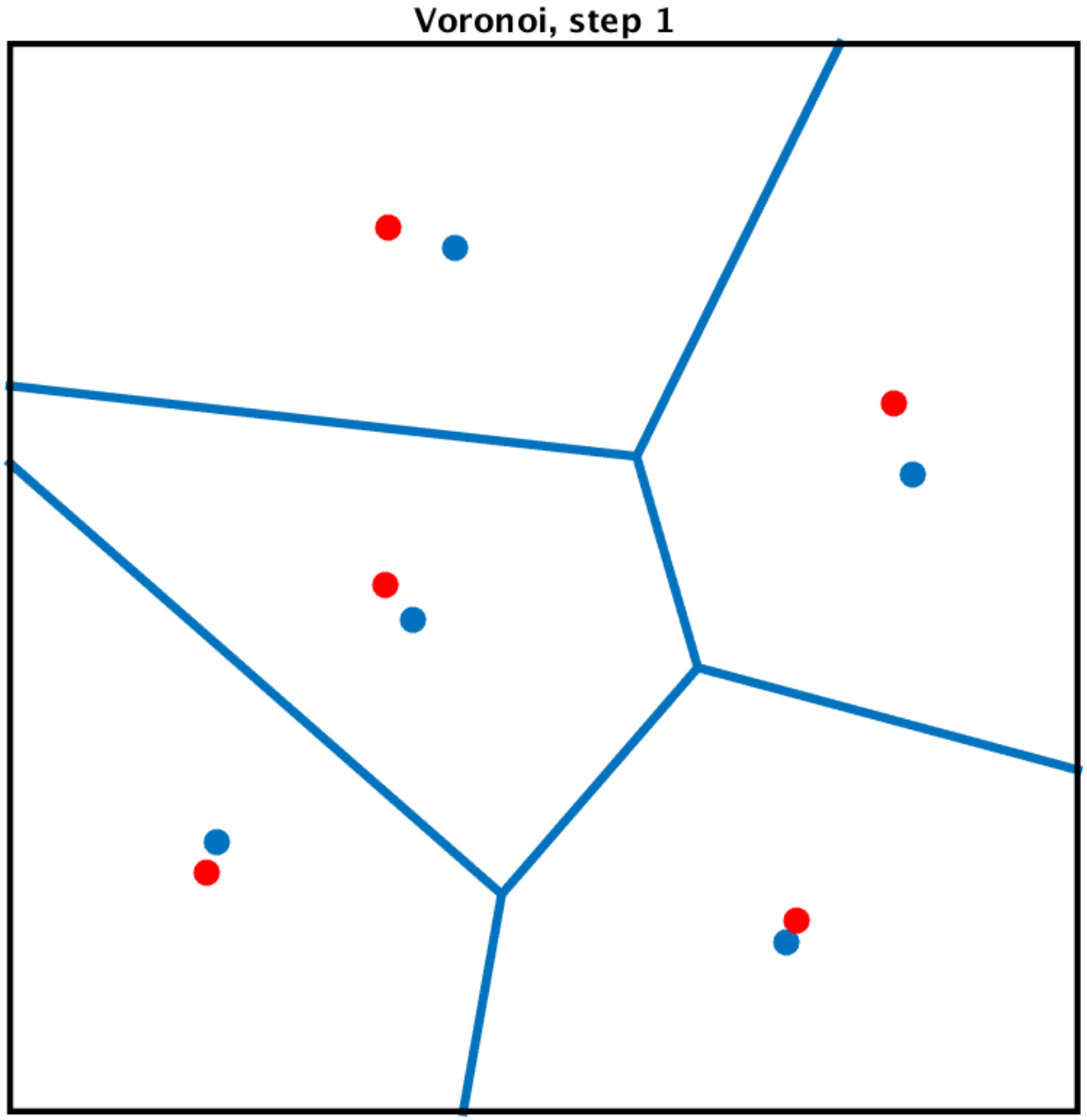
But, if you think about it, and if you look at the adjusted map, something else has happened. Because the mailboxes have moved, there are some customers that now should switch from their old mailbox to a new one, because it has actually come closer to them.

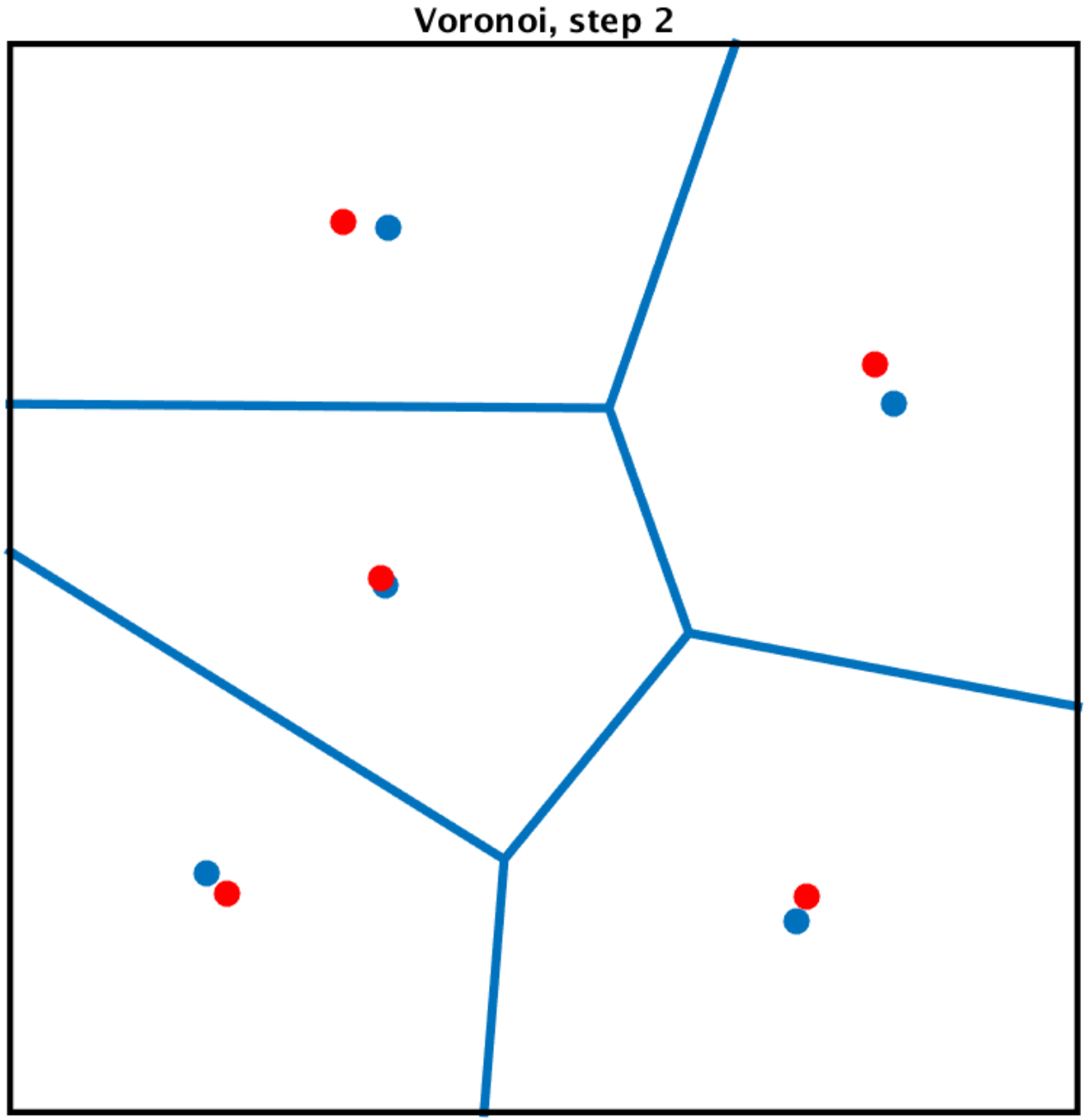
We can verify that this happens by simply doing a new clustering, in which we redraw the territories. You can see that the new territories are in roughly the same shape and size as before, but there has definitely been some movement.

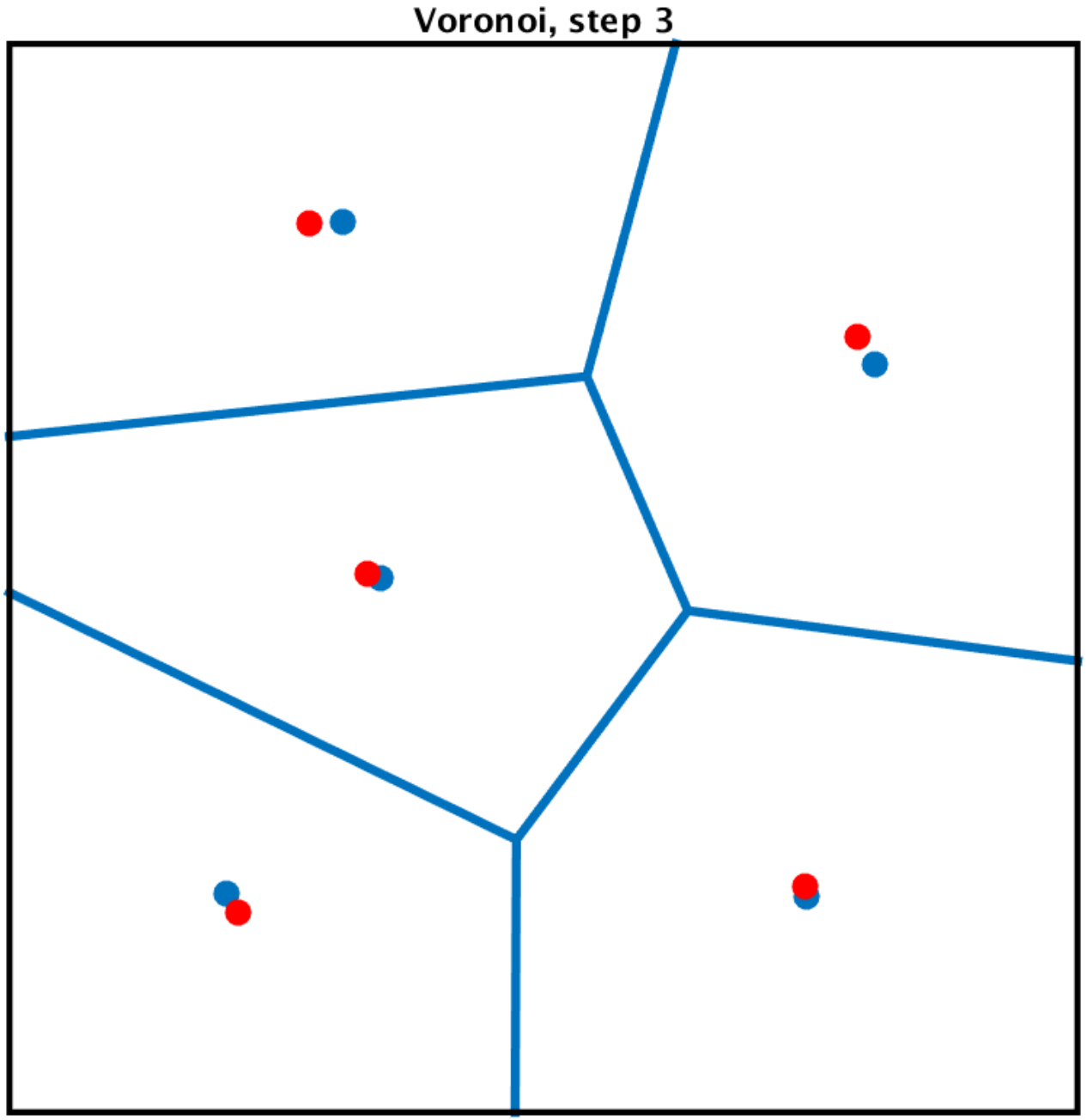
The important thing is that all this movement represents **improvement**, that is, the mailbox and the territories are both adjusting in a way that is reducing the overall distances.

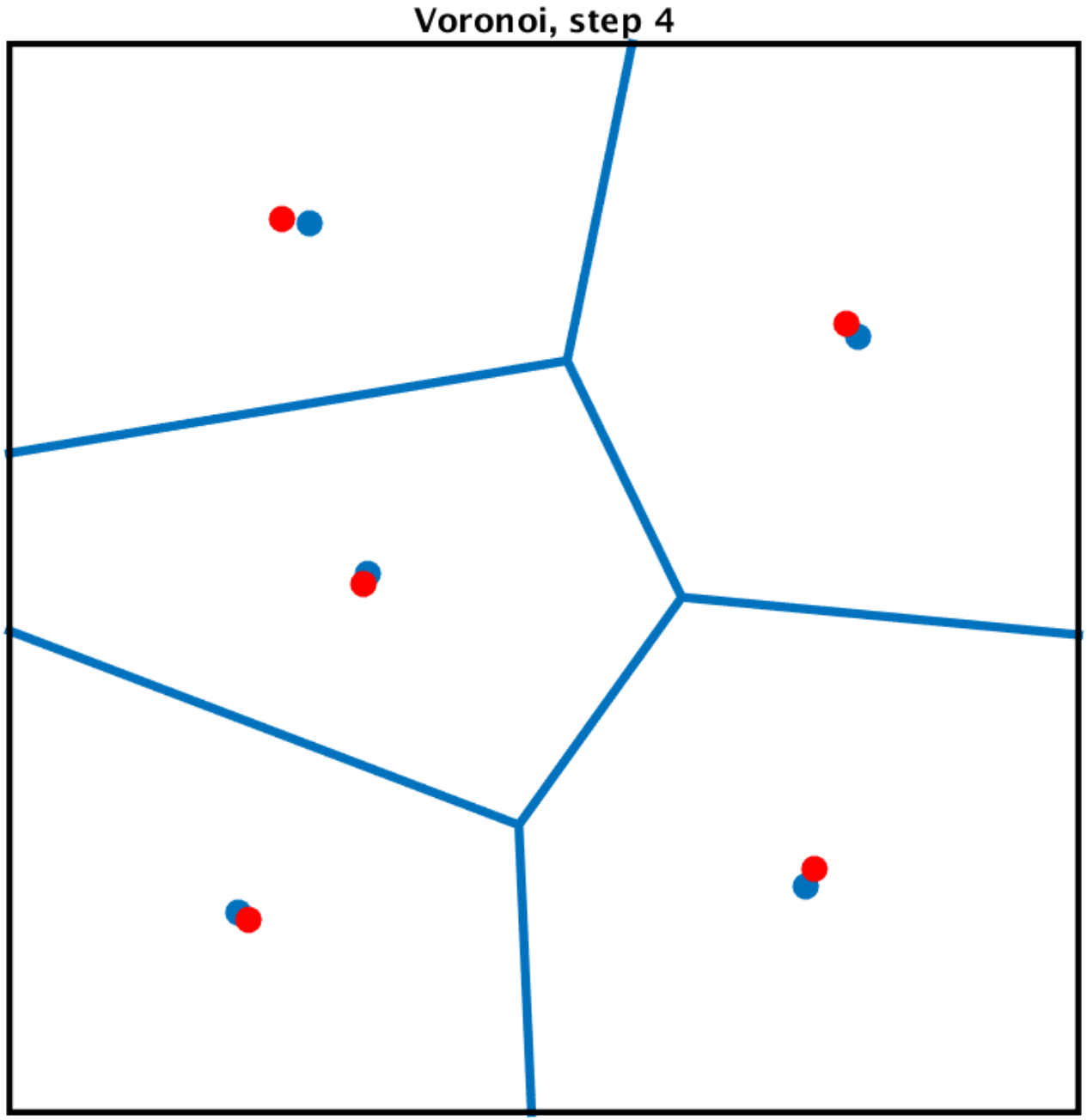

Initial Generators

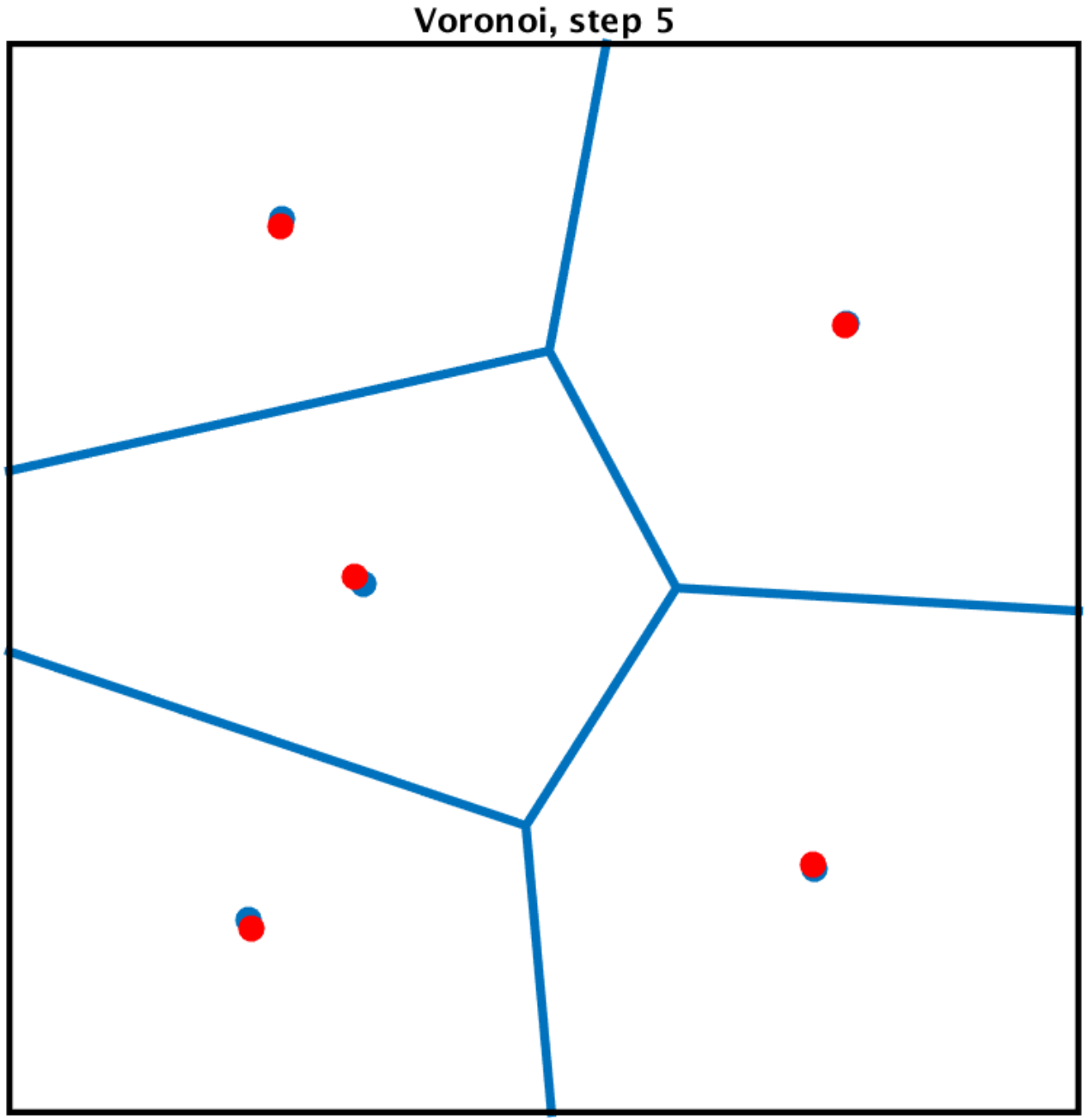

Voronoi, step 0


Voronoi, step 1


Voronoi, step 2


Voronoi, step 3


Voronoi, step 4


Voronoi, step 5

However, it seems like we have been caught in a never-ending trap:

```
Step 1: Move the mailboxes to the centers of the territories.
Step 2: Change the territory assignments to the nearest mailbox.
Step 3: Repeat Step 1 and Step 2.
```

Technically, it's true that we can repeat this cycle forever, but there are three good things to point out:

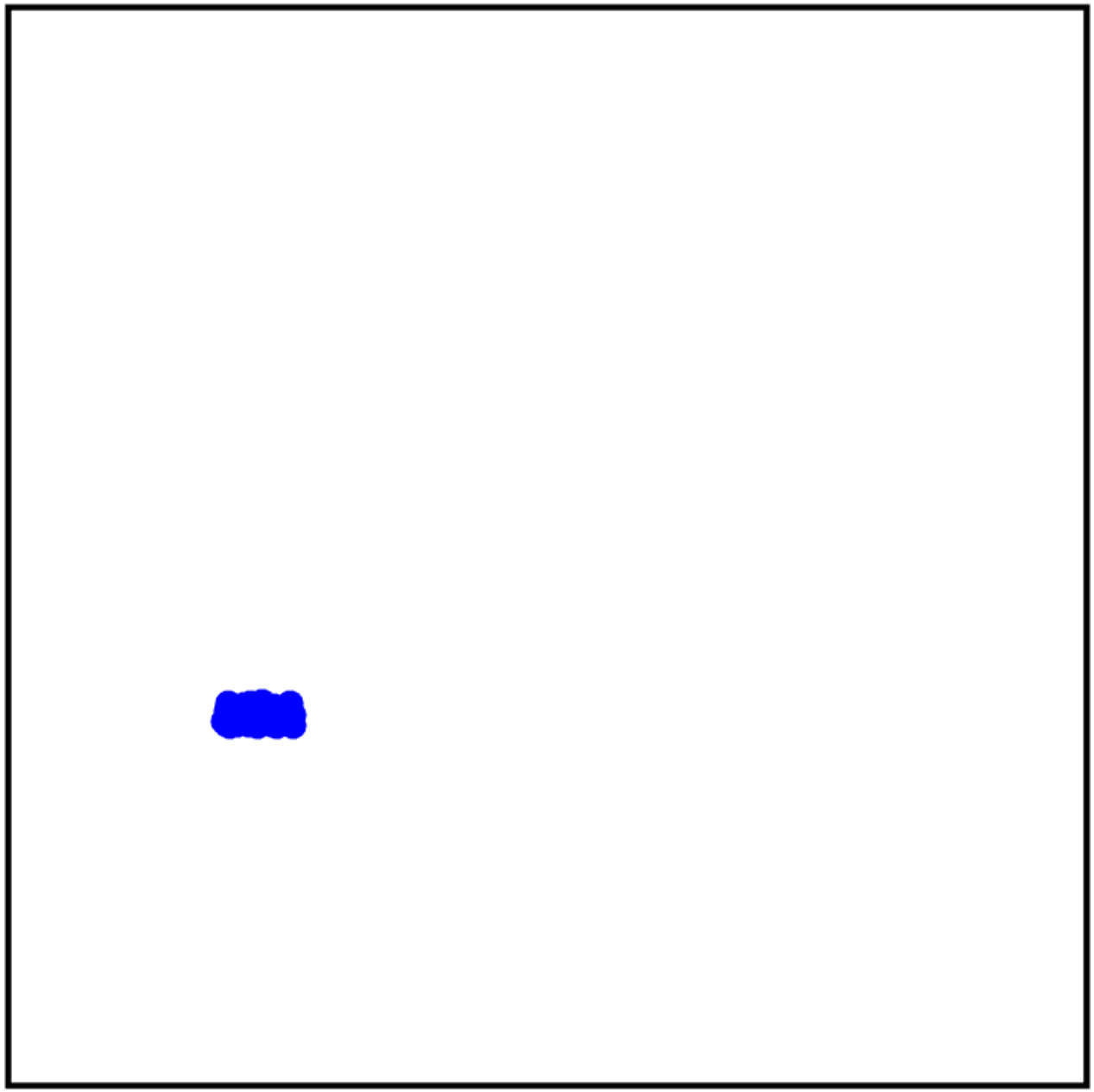
1. Each step we take must reduce the overall distance between each customer and the nearest mailbox;
2. As this overall distance decreases, it can't become zero, so the changes must become less and less;
3. Eventually, the changes will become so small we can assume we have found a nearly perfect arrangement, so we can stop.

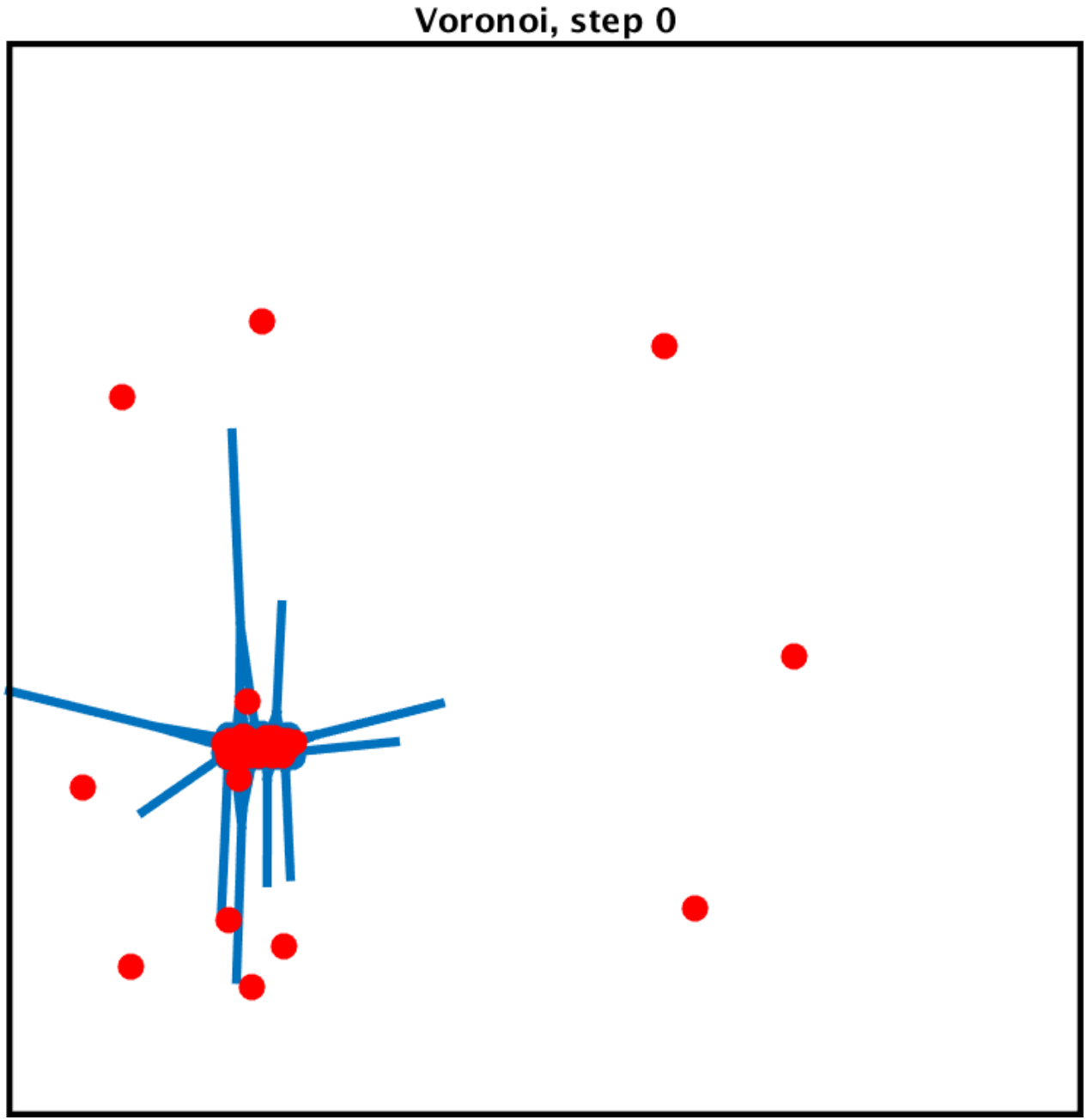
Here is a second example, using 30 mailboxes, in which our initial layout has them all crunched up in one tiny area.

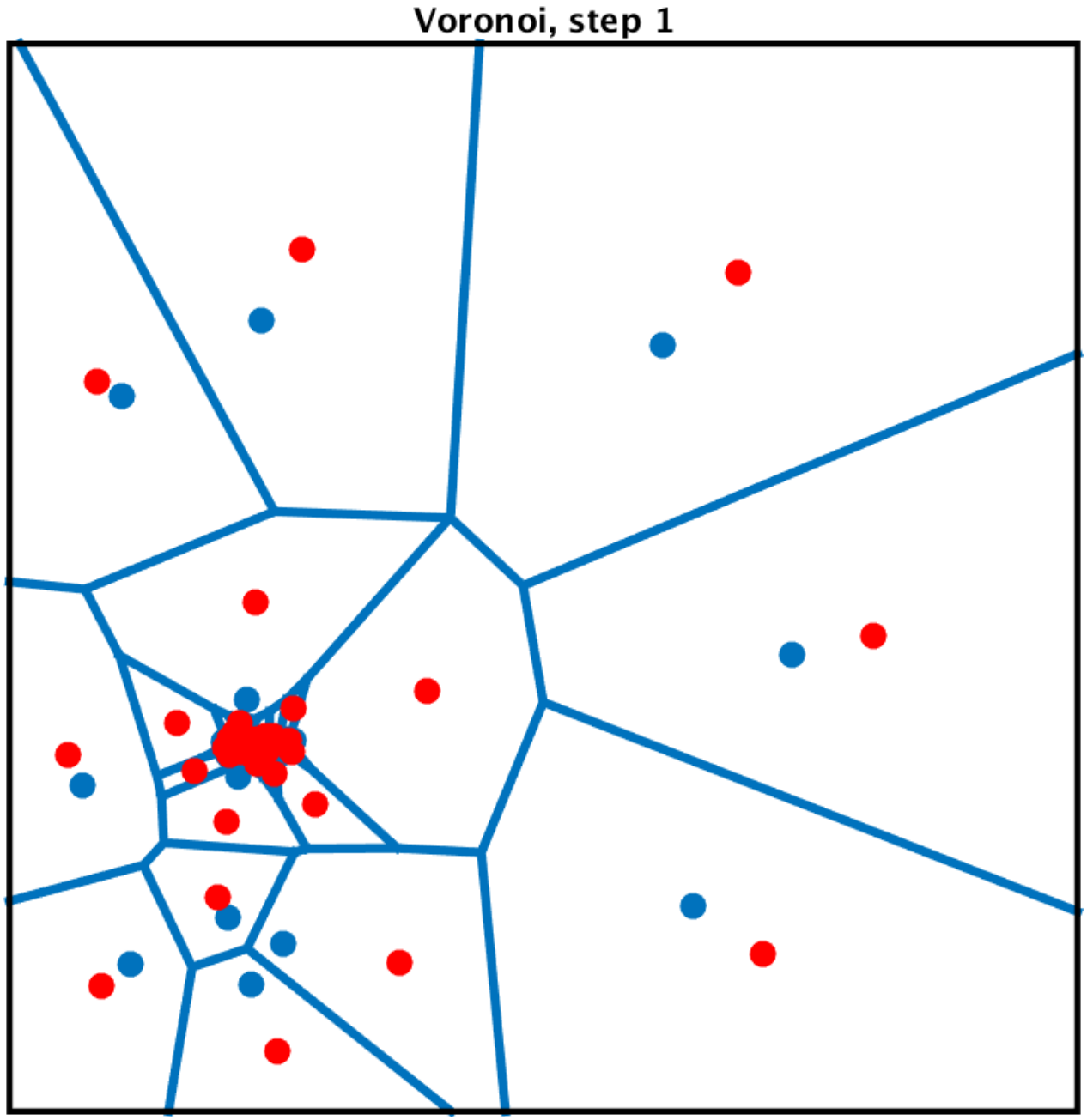
Because of the bad starting values, it takes a while for the points to arrange themselves, so we will have to take many steps.

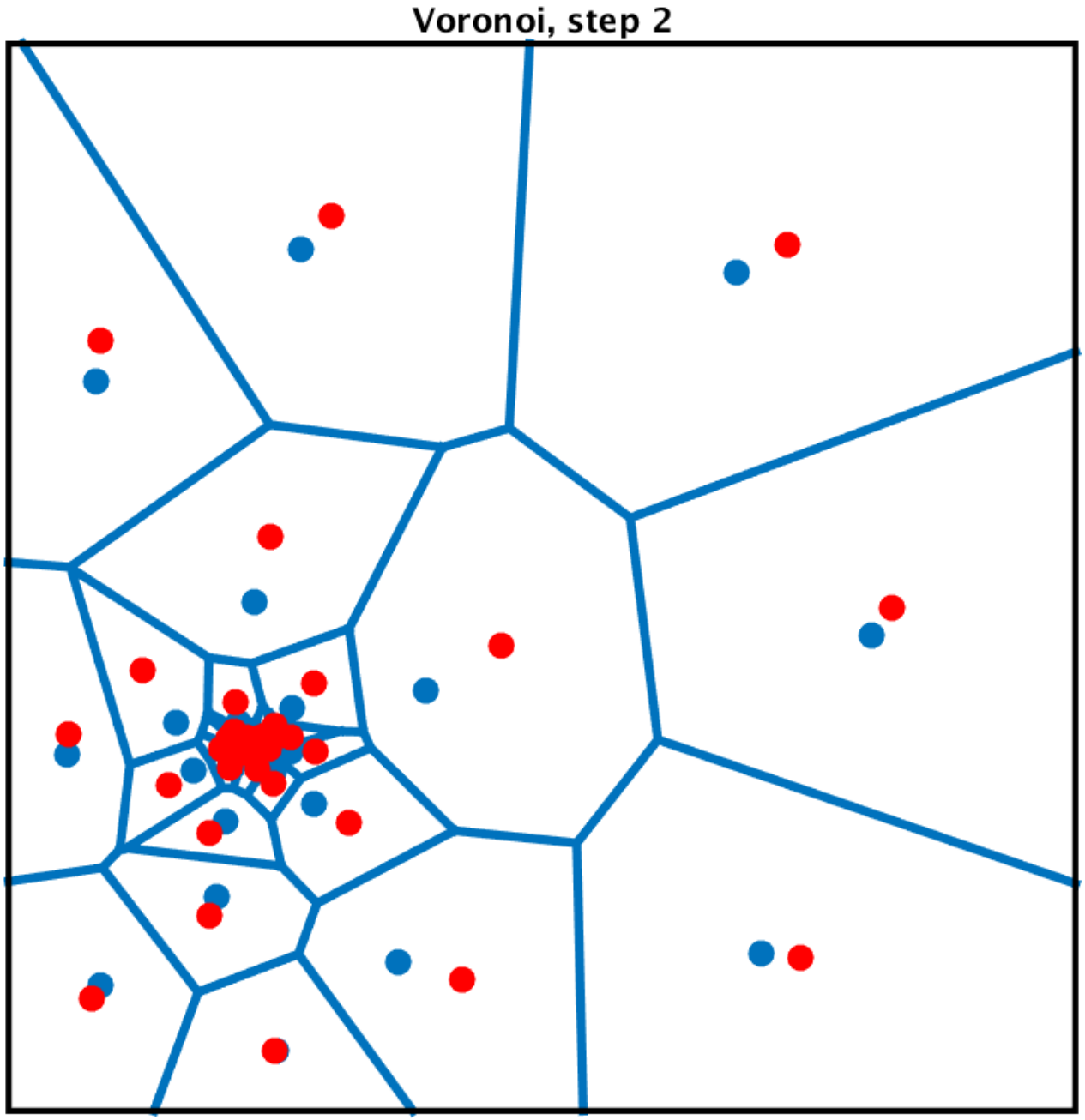
In this presentation, we will show the steps rapidly, like an animation, and skip many of the later steps where things slow down. But you should see that the points seem to move as though they are trying to politely but neatly rearrange themselves.

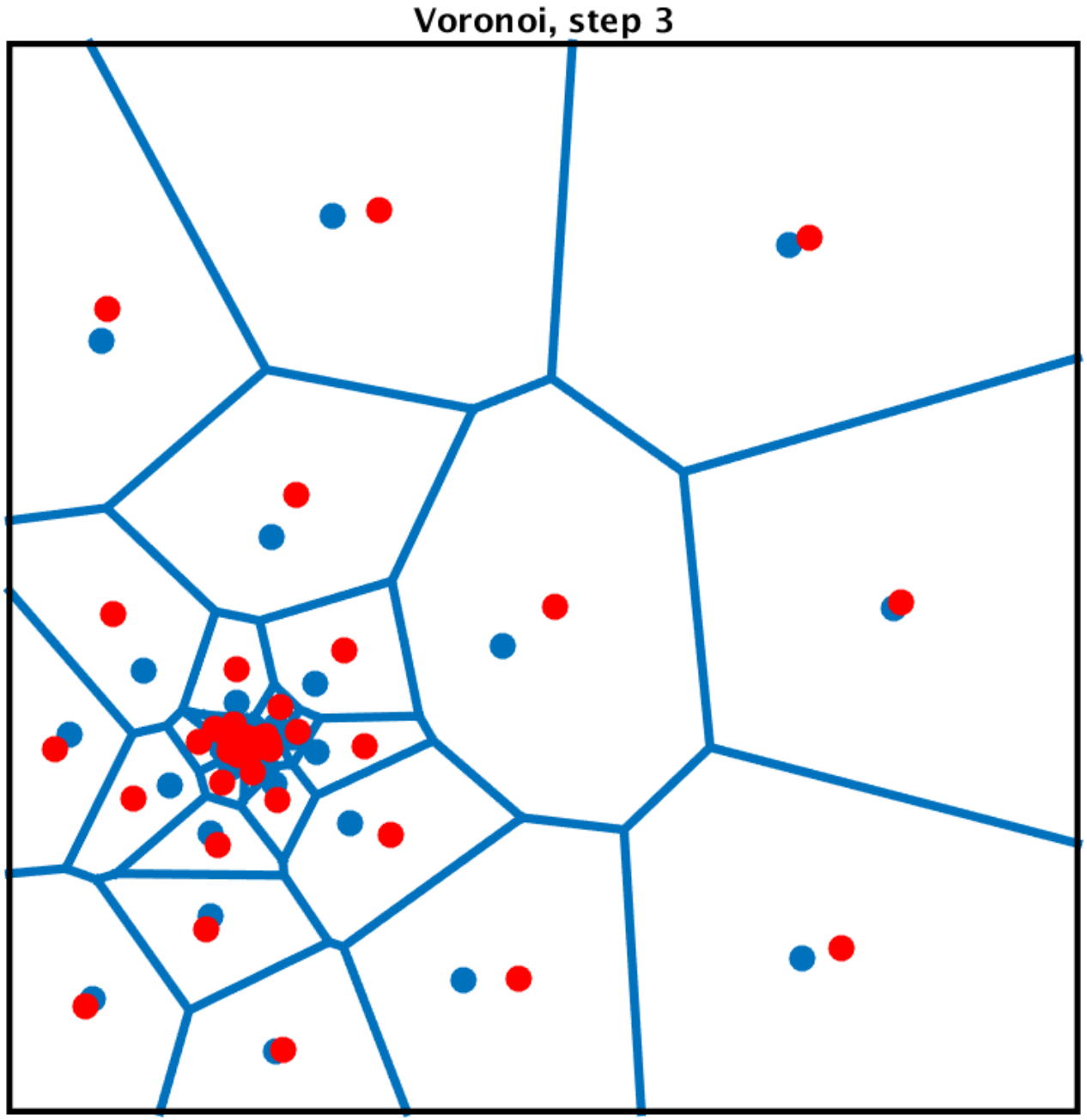
Initial Generators

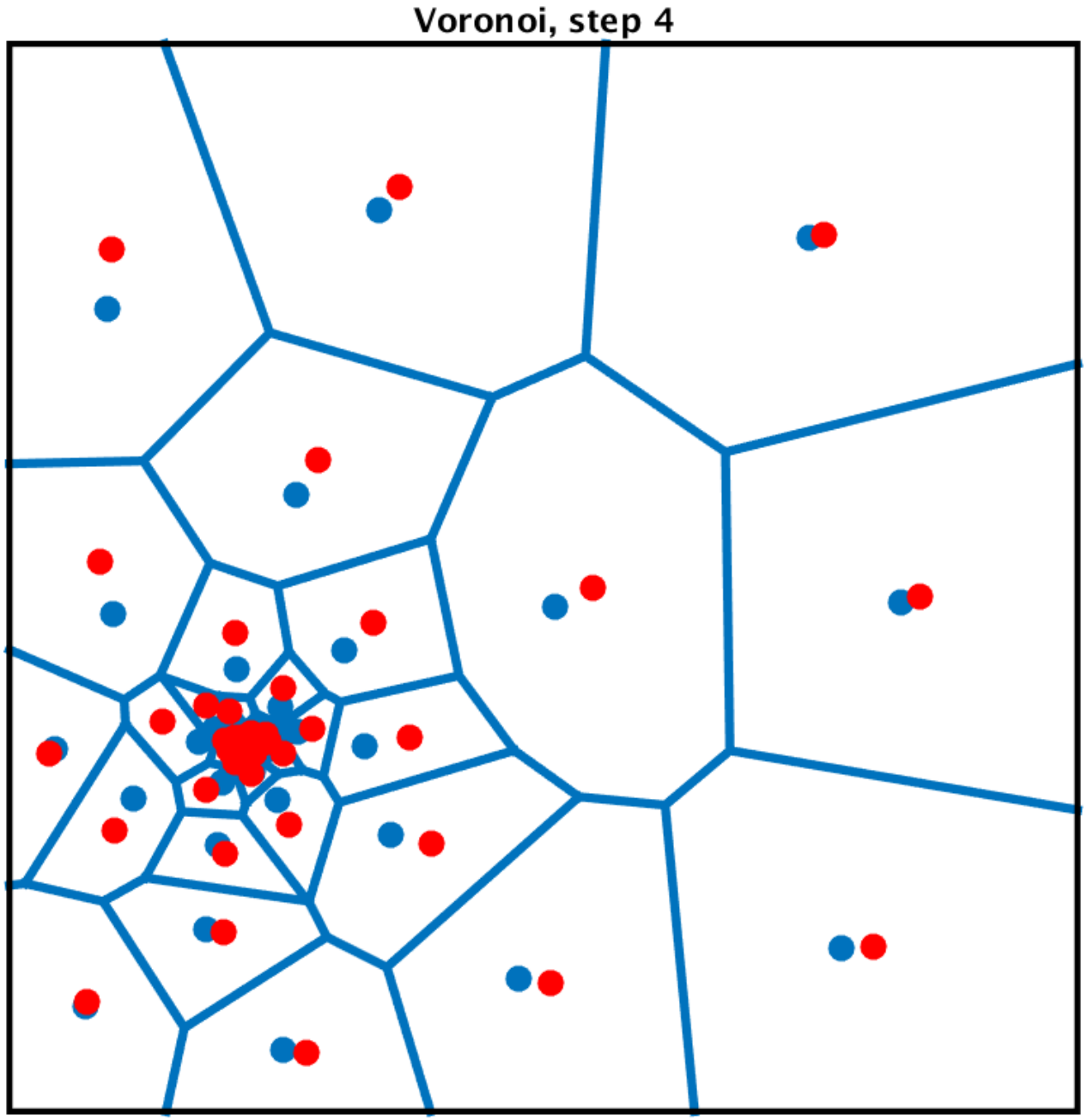


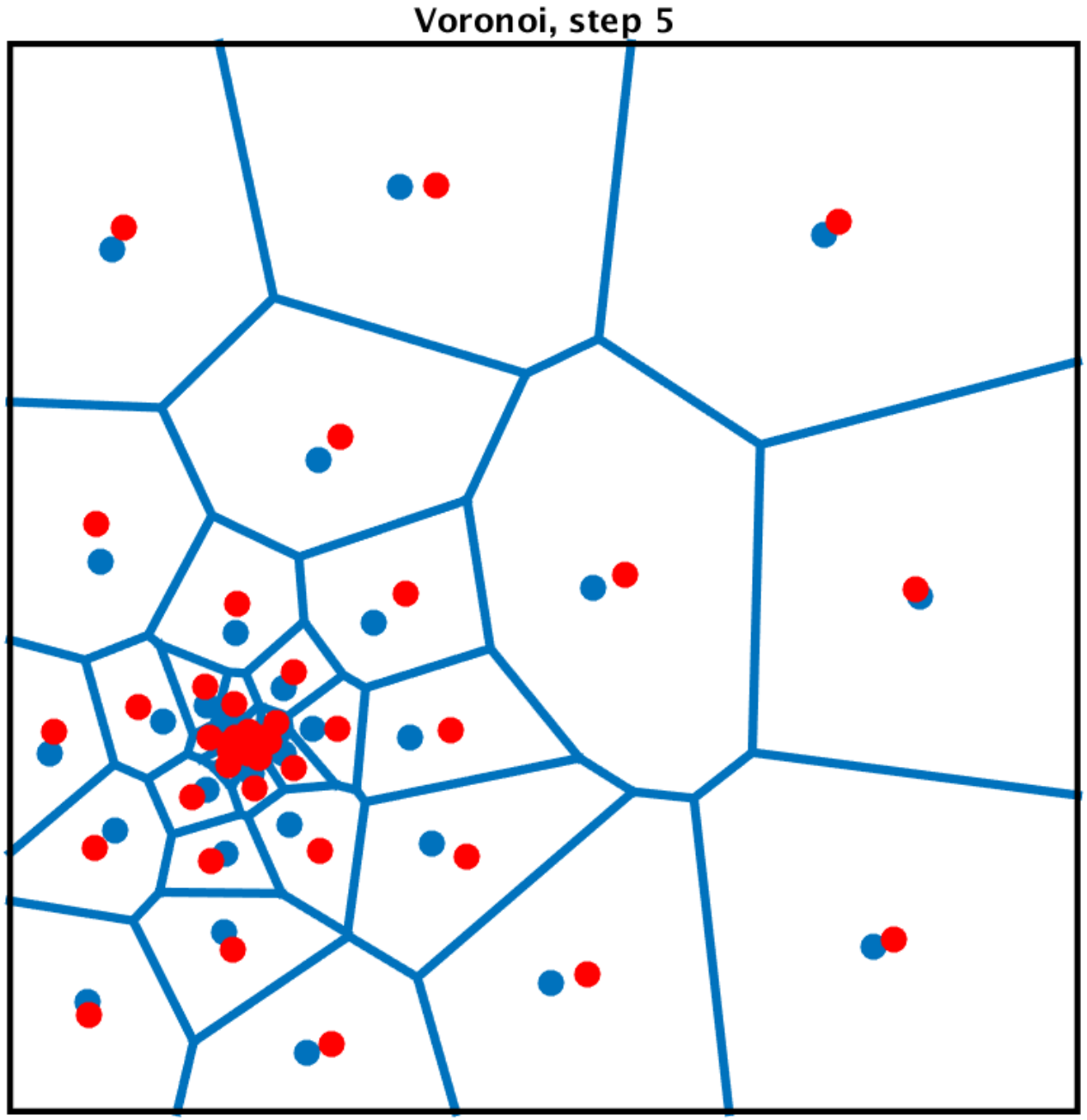

Voronoi, step 0

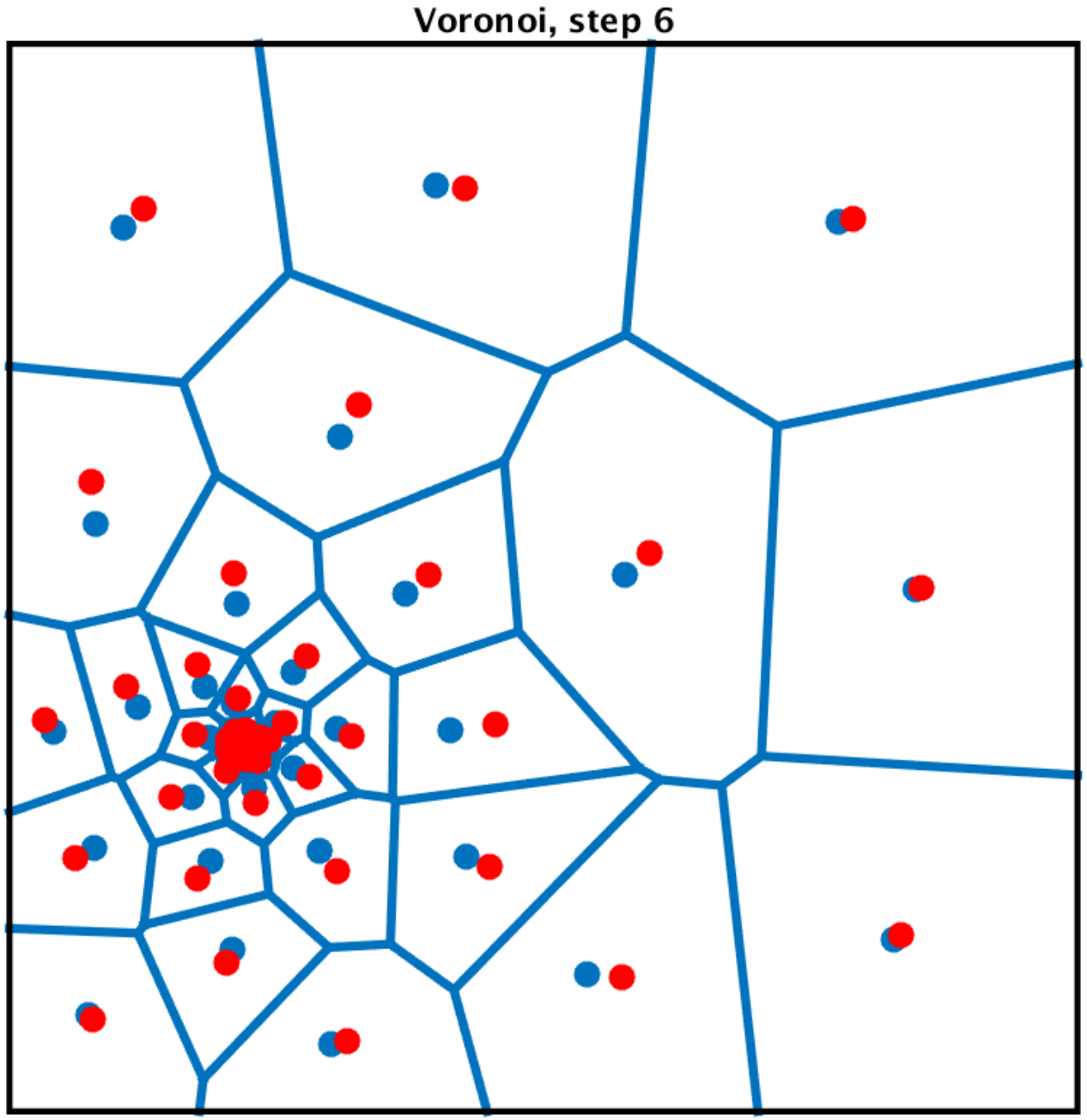

Voronoi, step 1

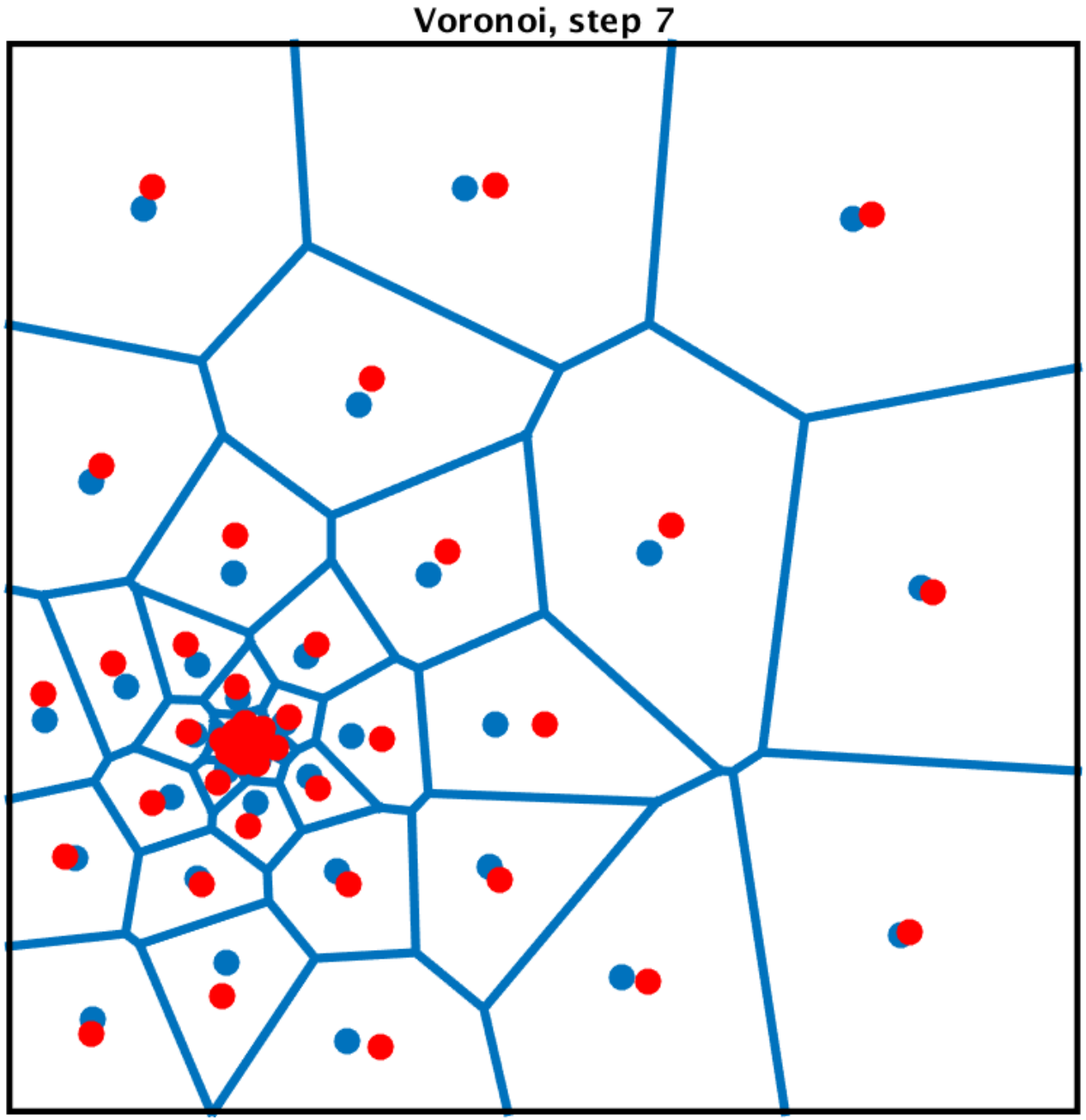

Voronoi, step 2

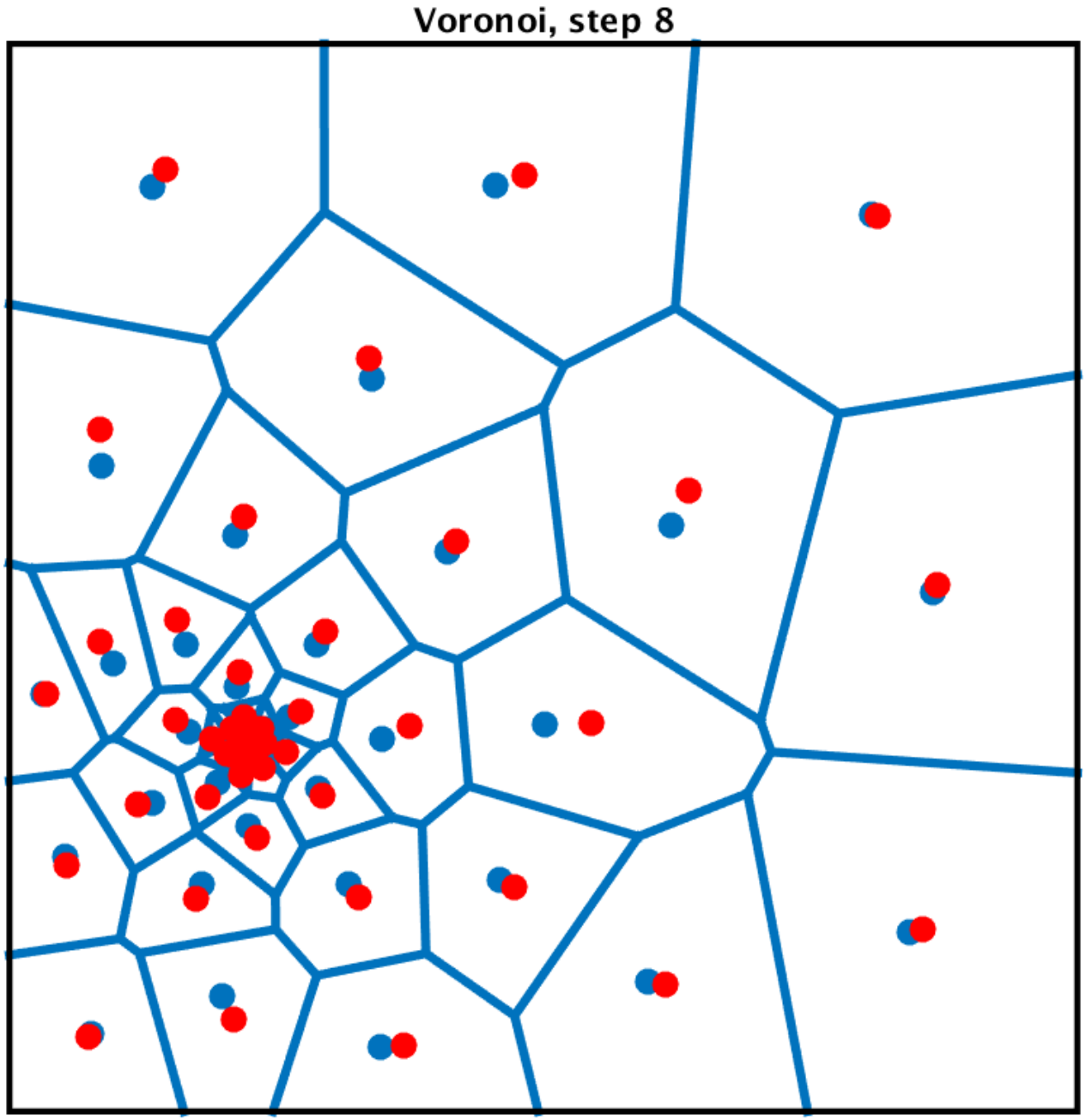

Voronoi, step 3

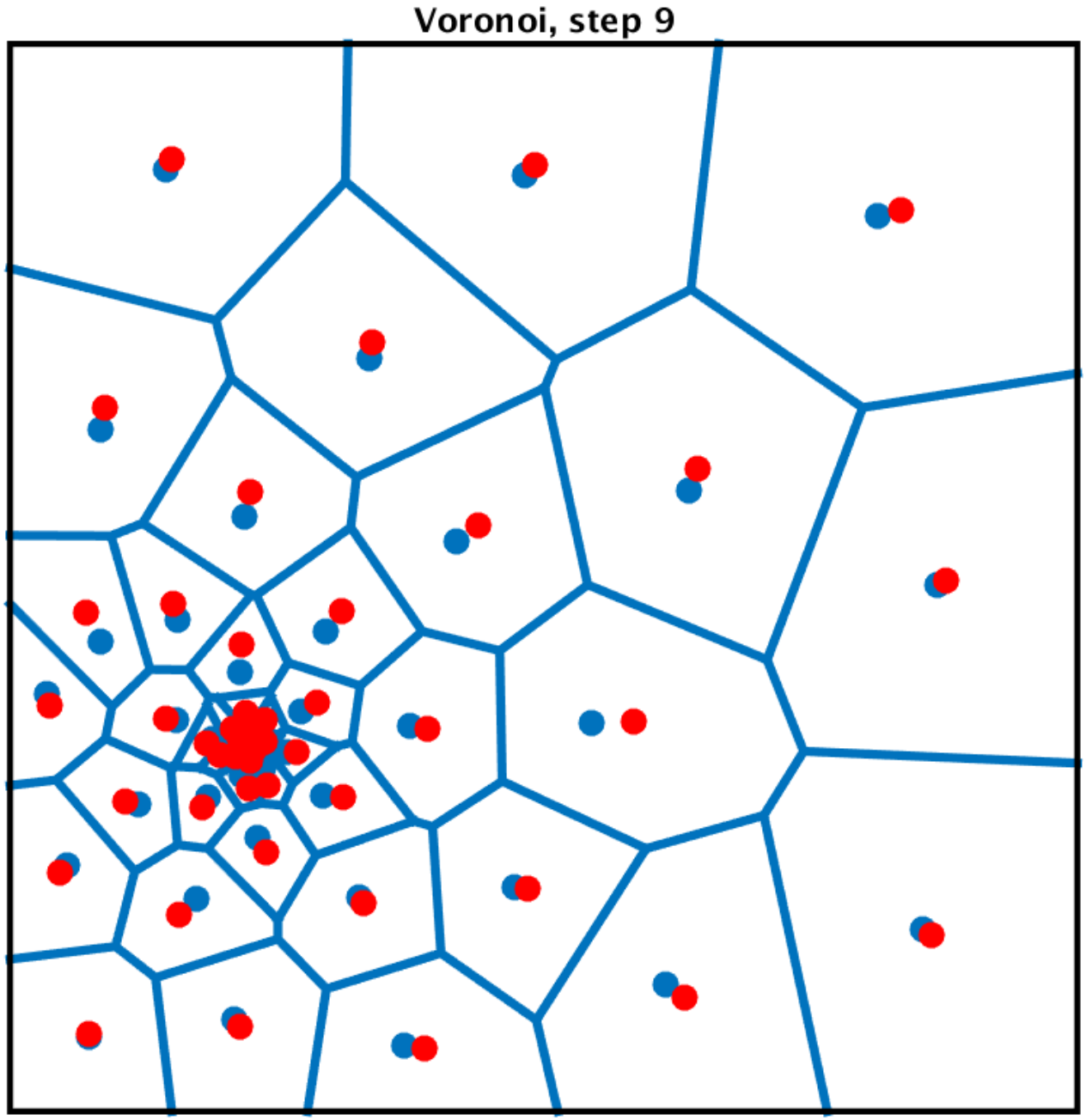

Voronoi, step 4

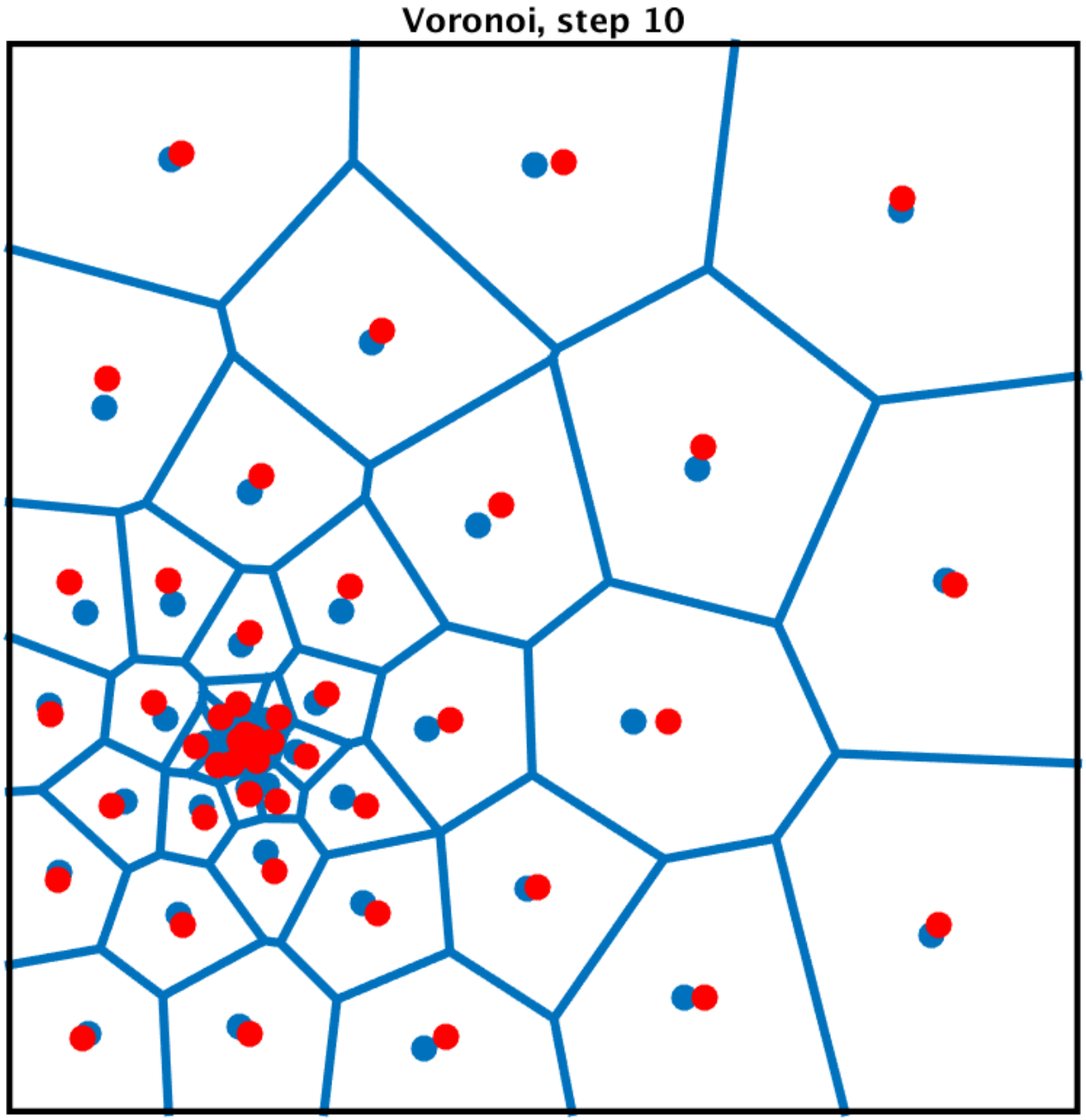

Voronoi, step 5

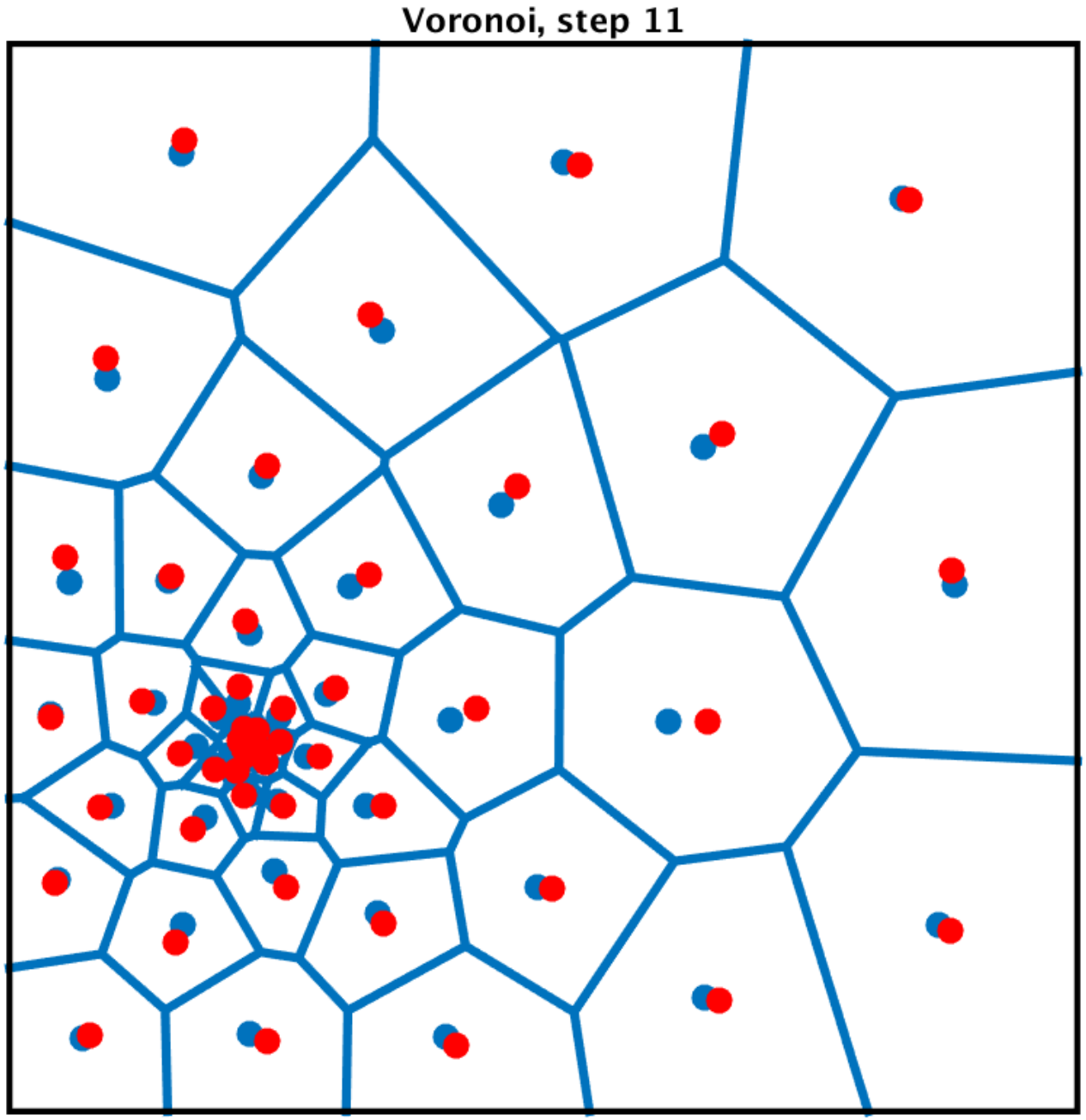

Voronoi, step 6

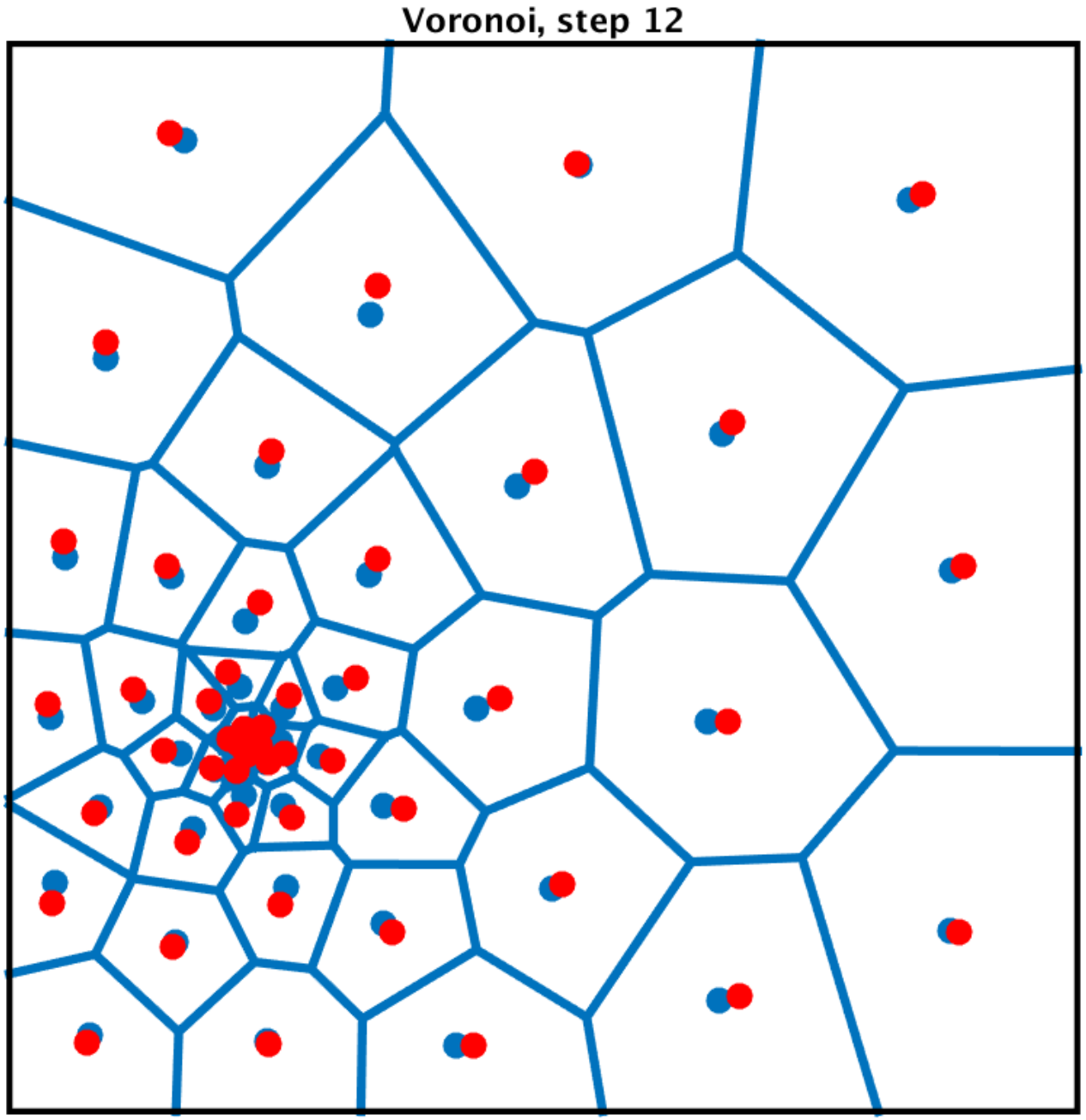

Voronoi, step 7

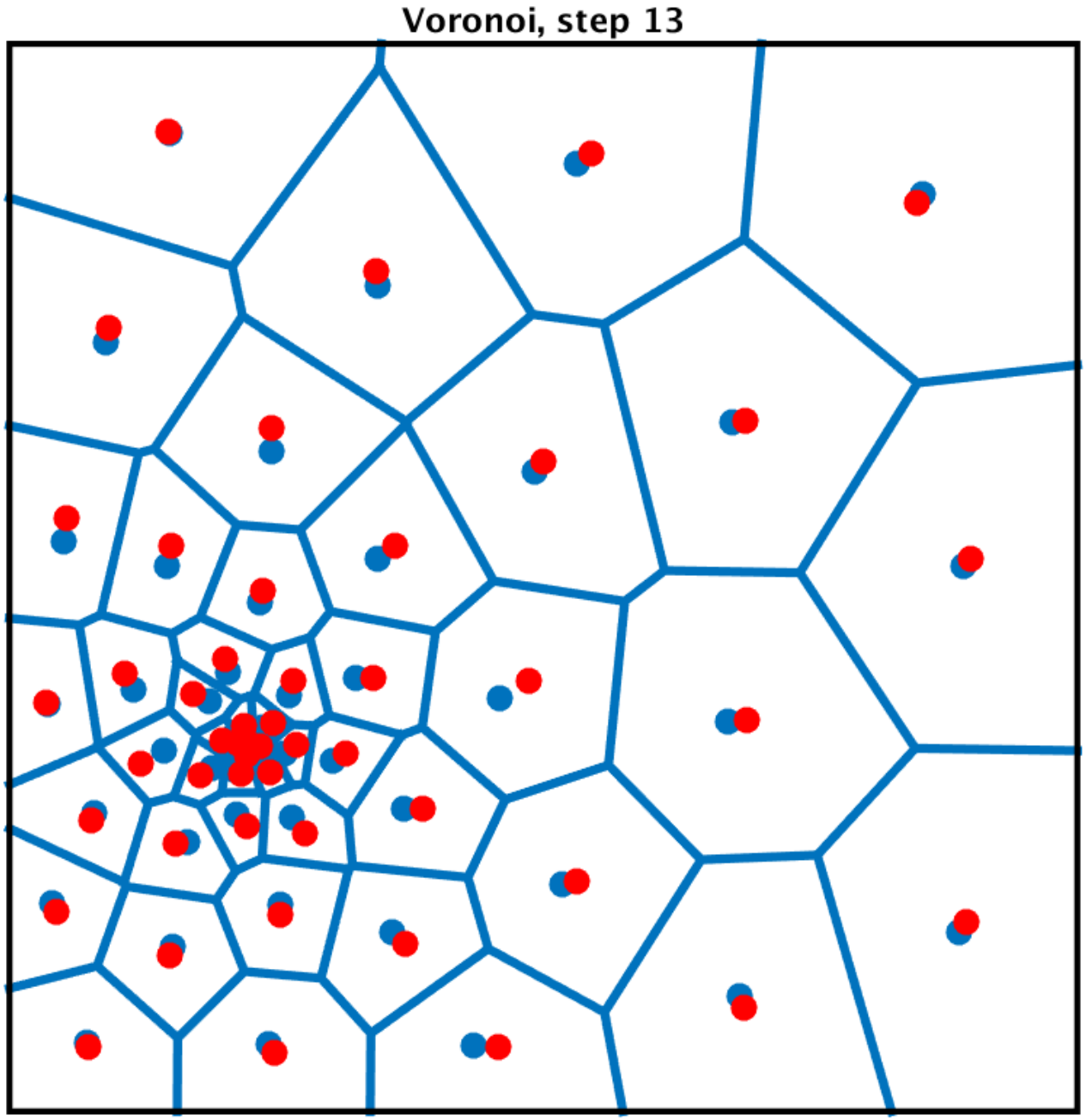

Voronoi, step 8

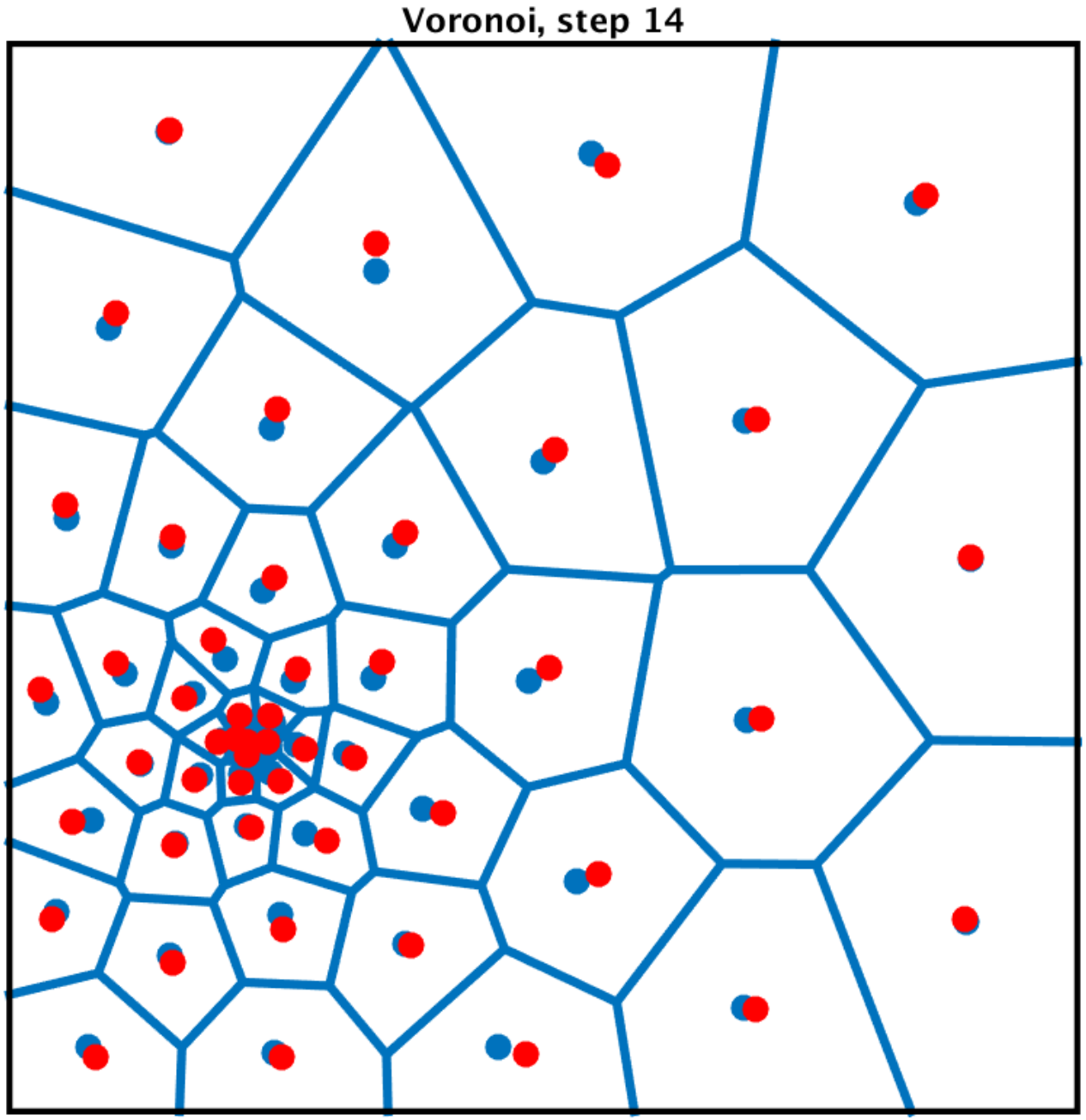

Voronoi, step 9

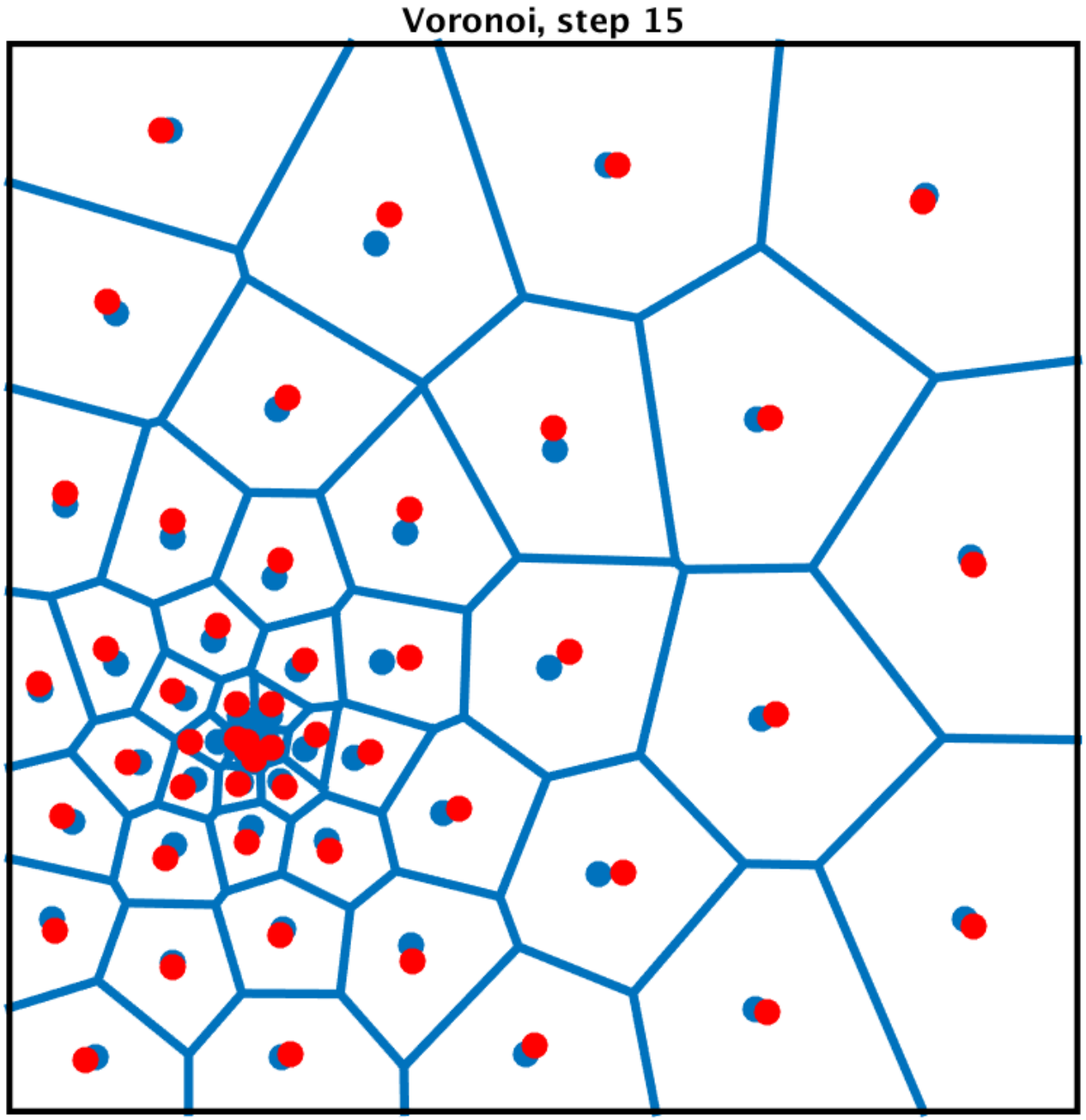

Voronoi, step 10

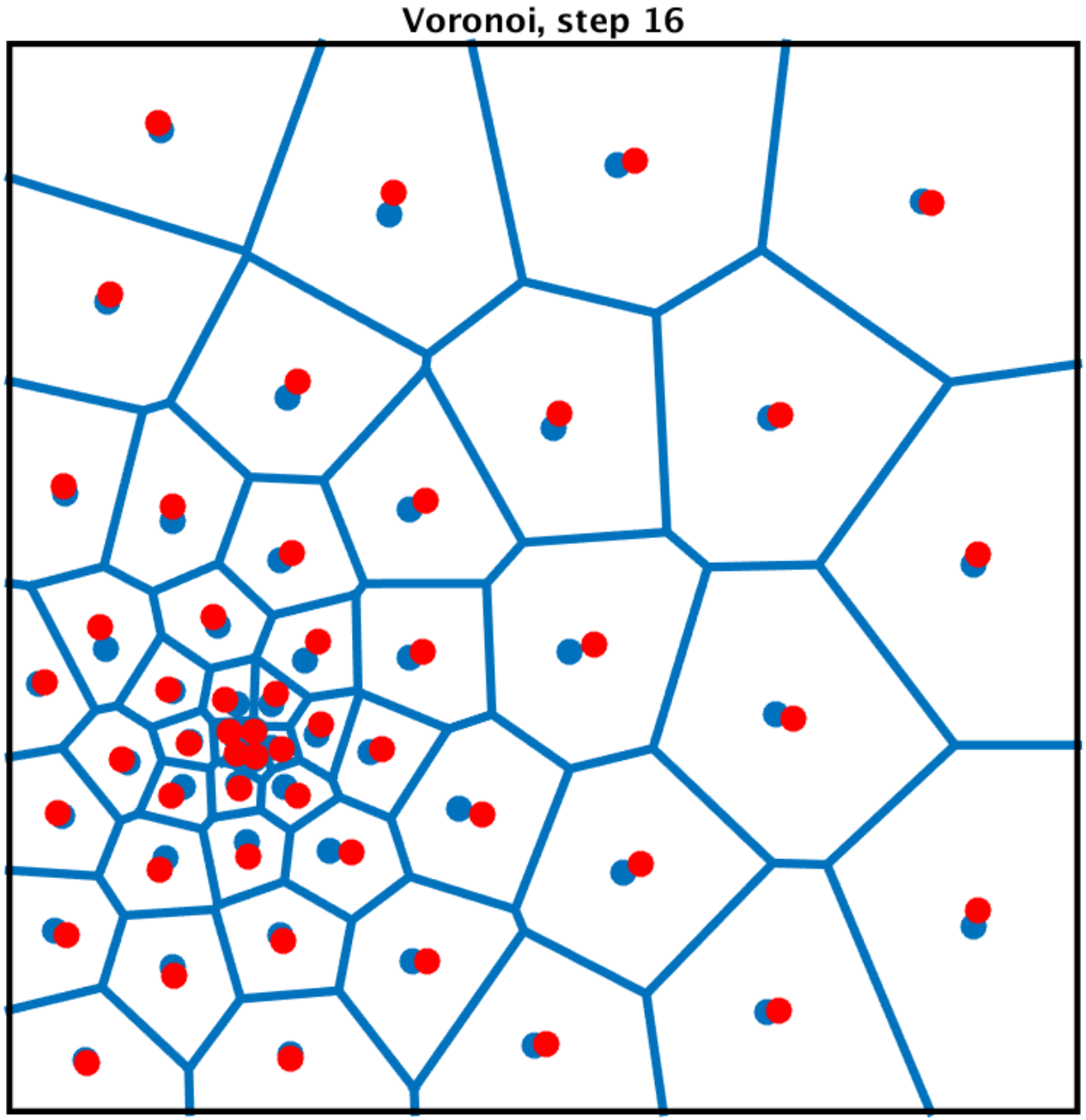

Voronoi, step 11

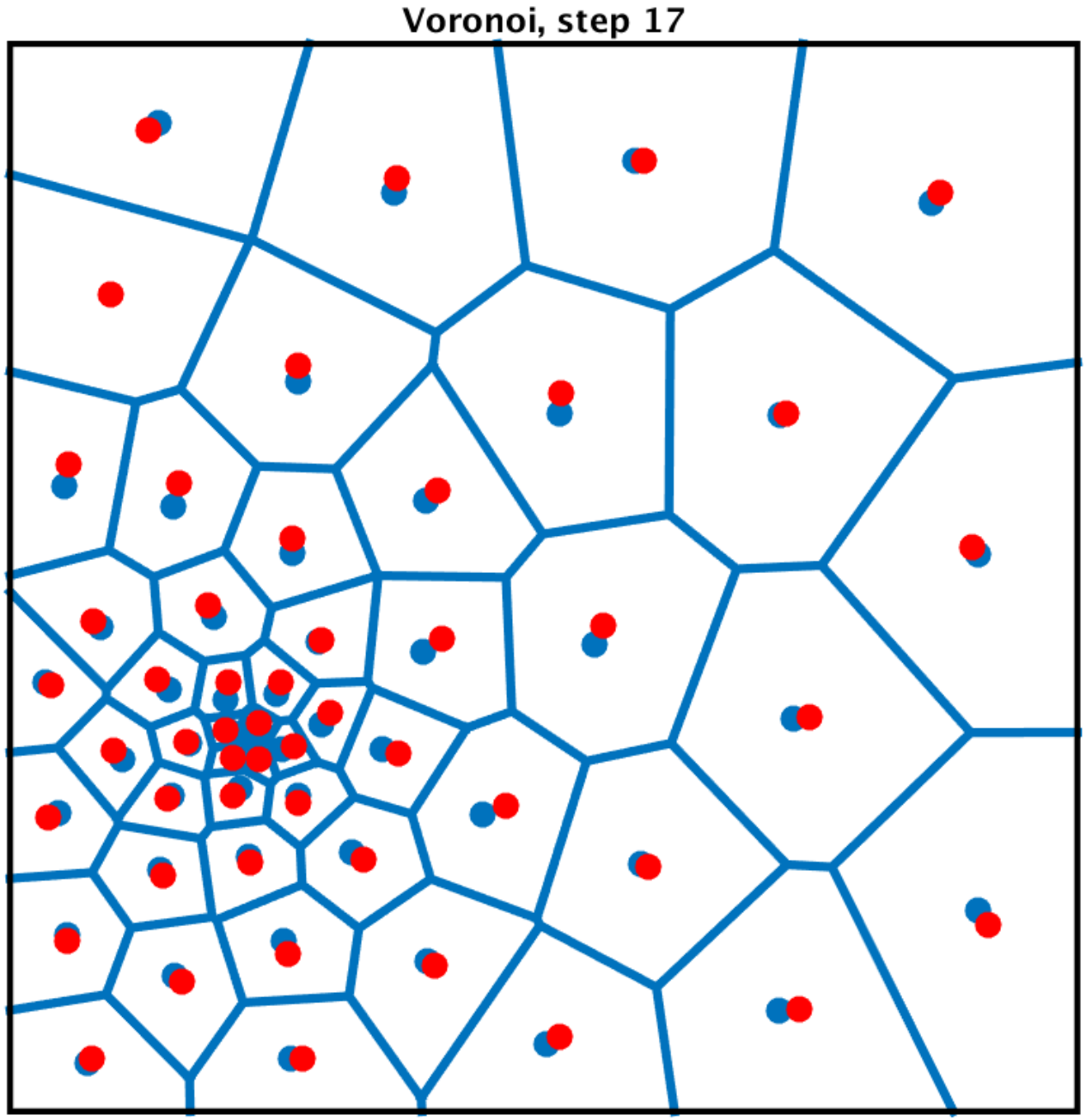

Voronoi, step 12

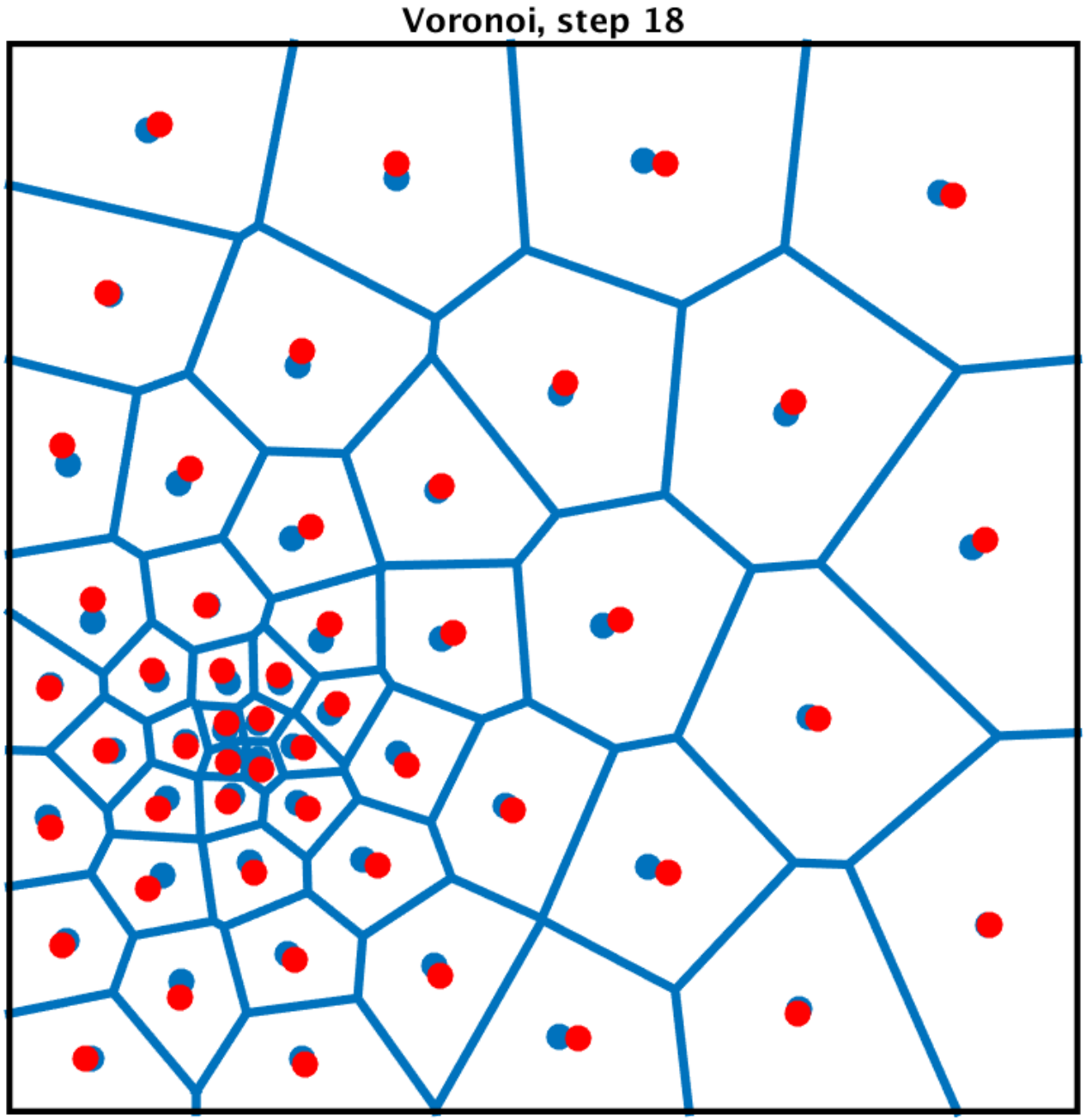

Voronoi, step 13

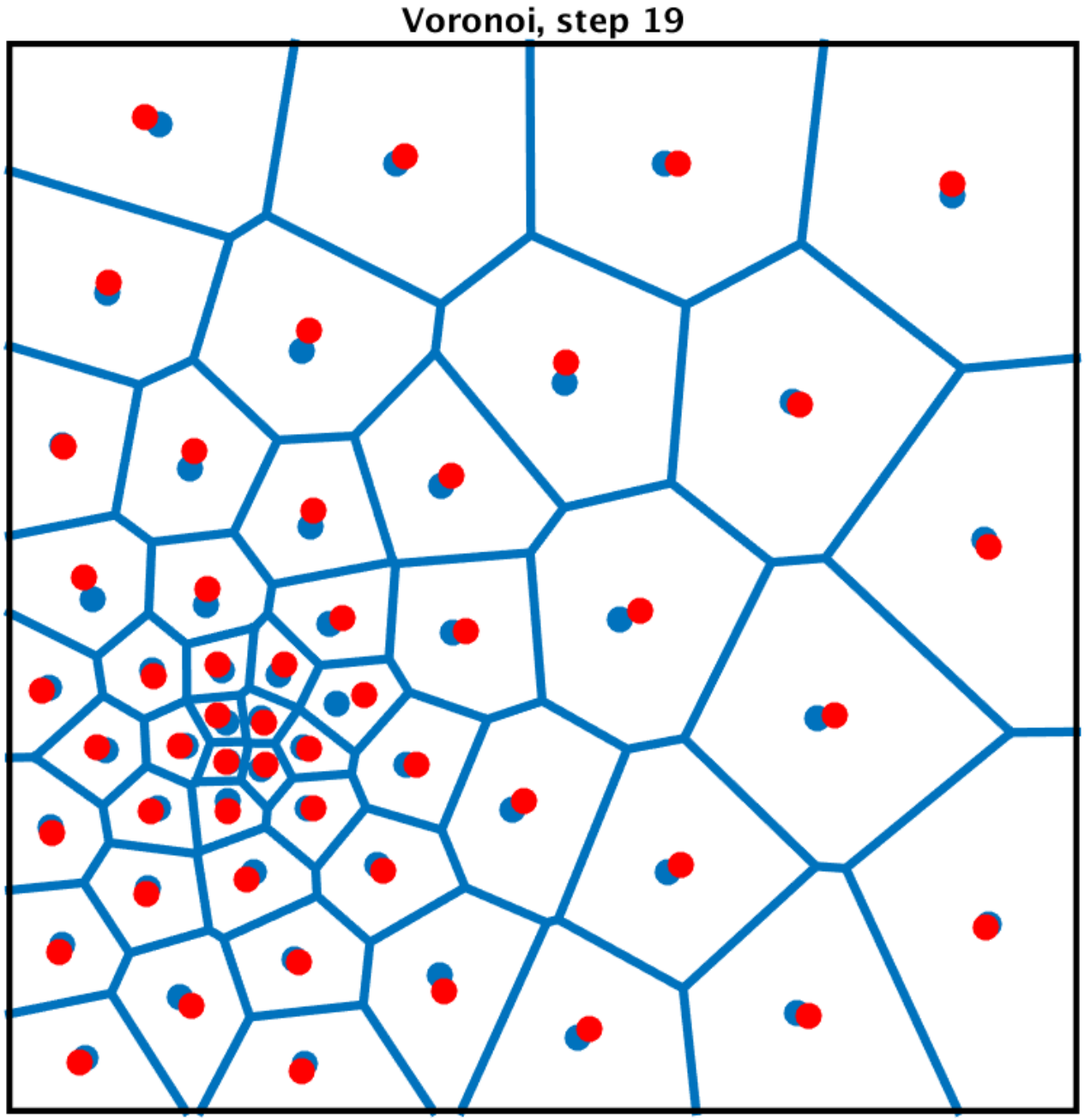

Voronoi, step 14

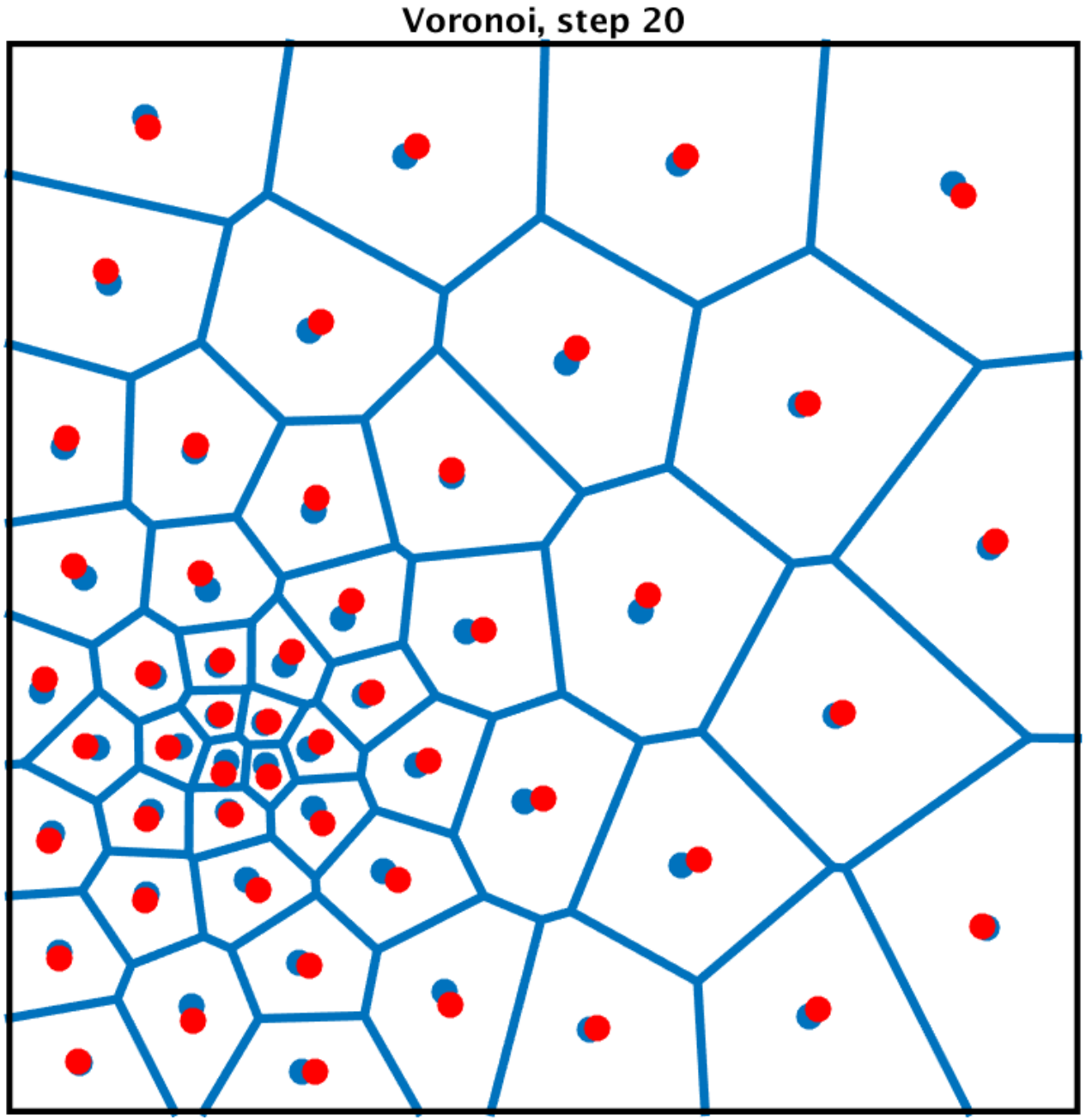

Voronoi, step 15

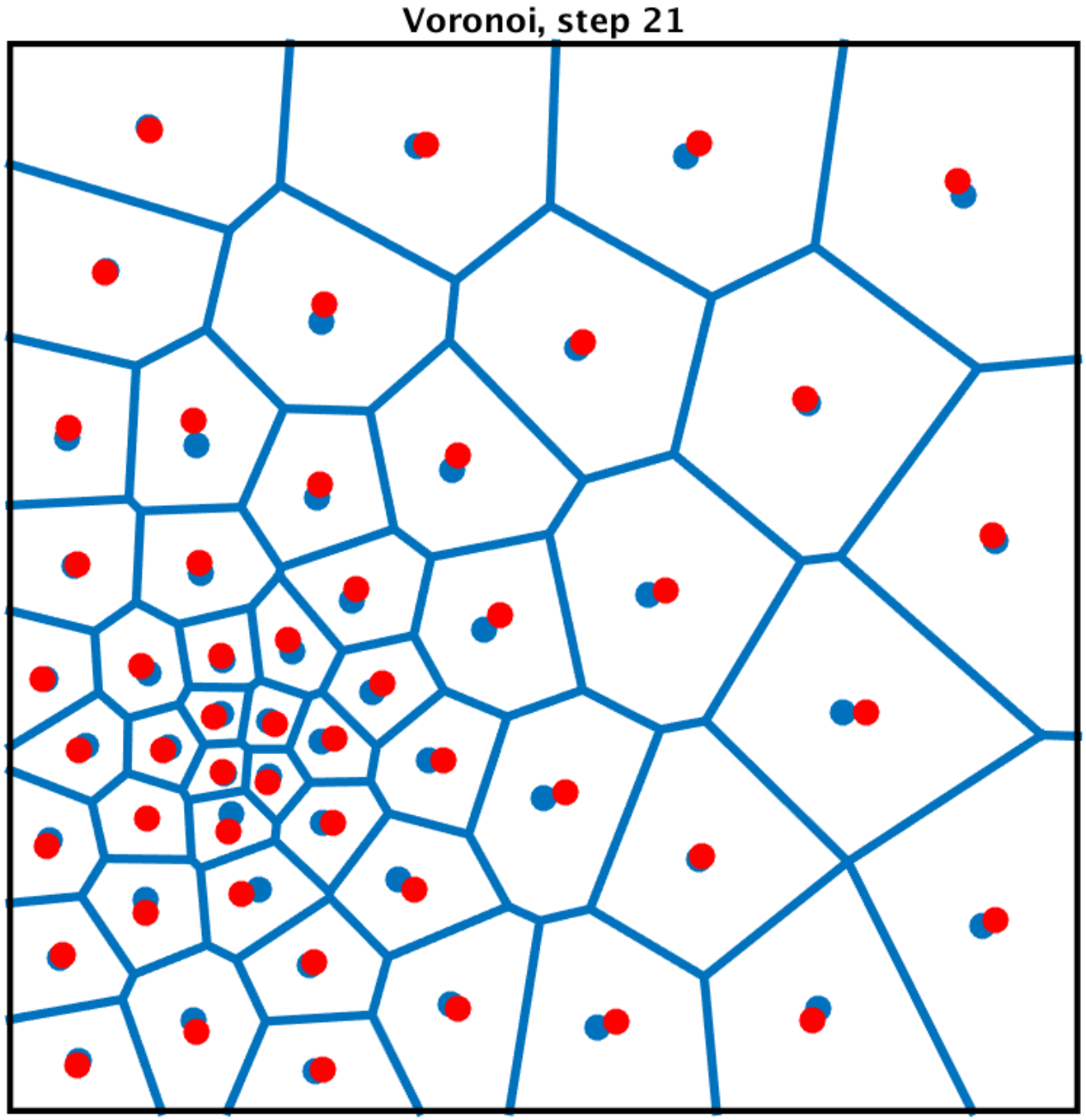

Voronoi, step 16

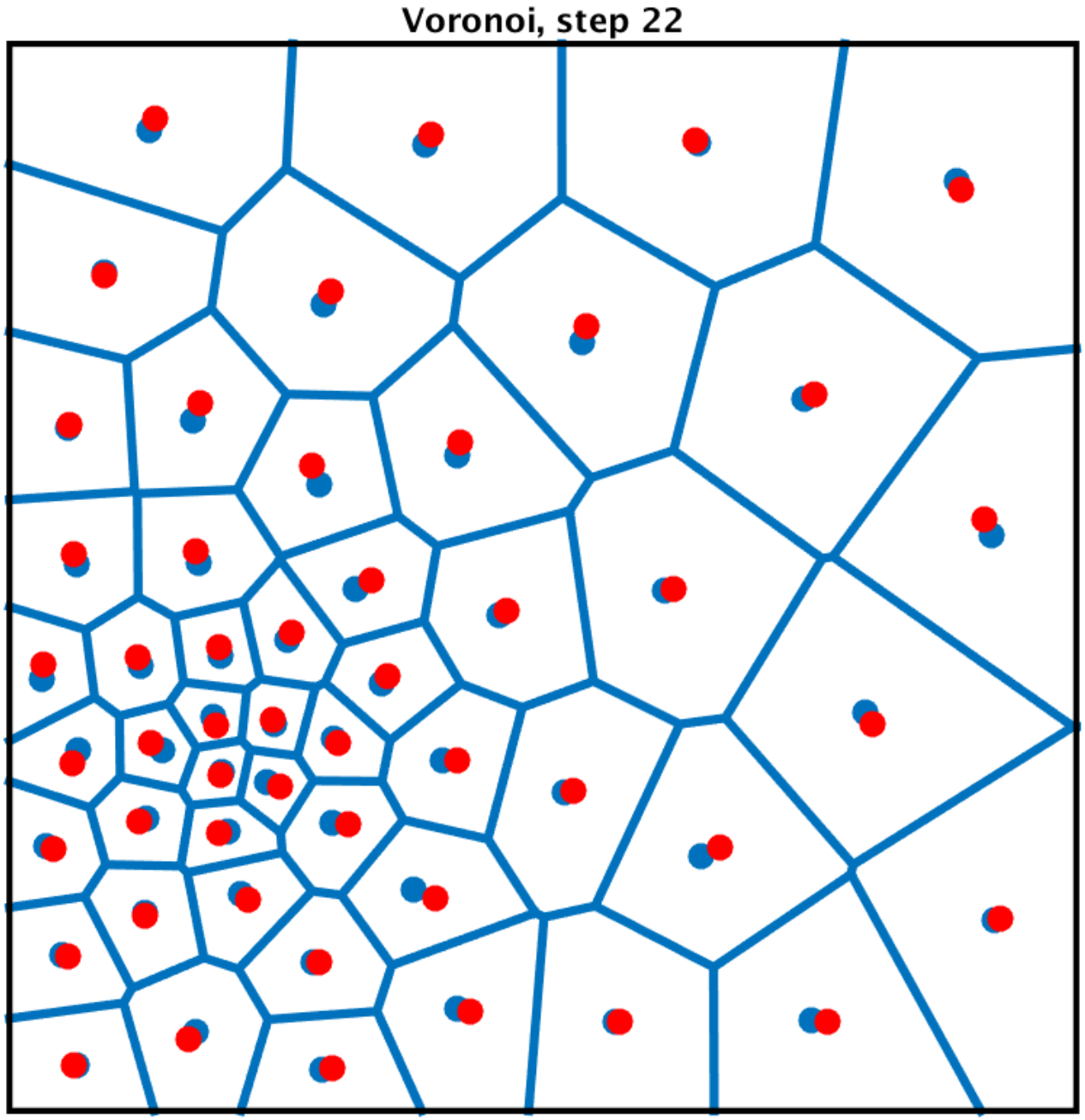

Voronoi, step 17

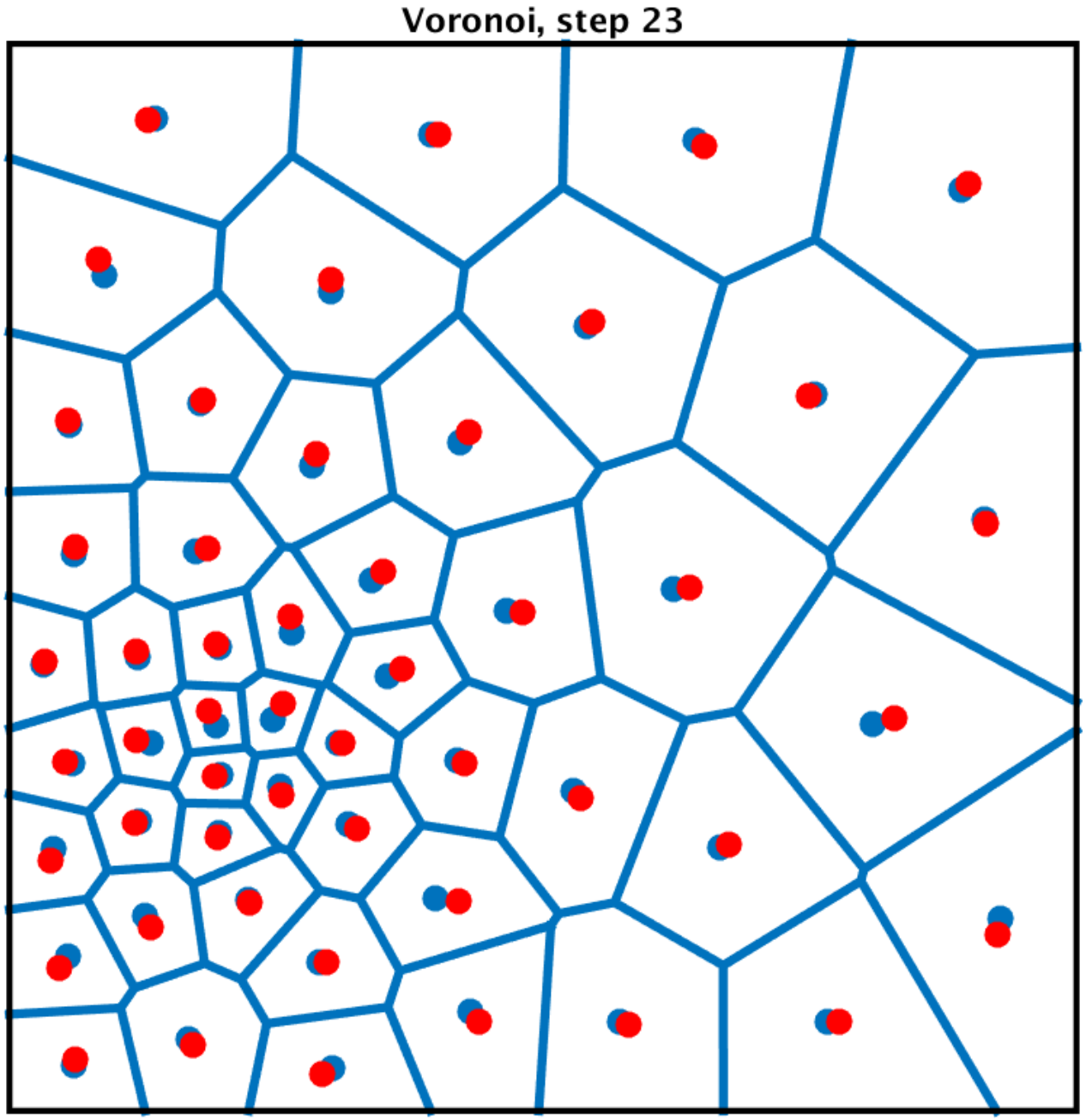

Voronoi, step 18

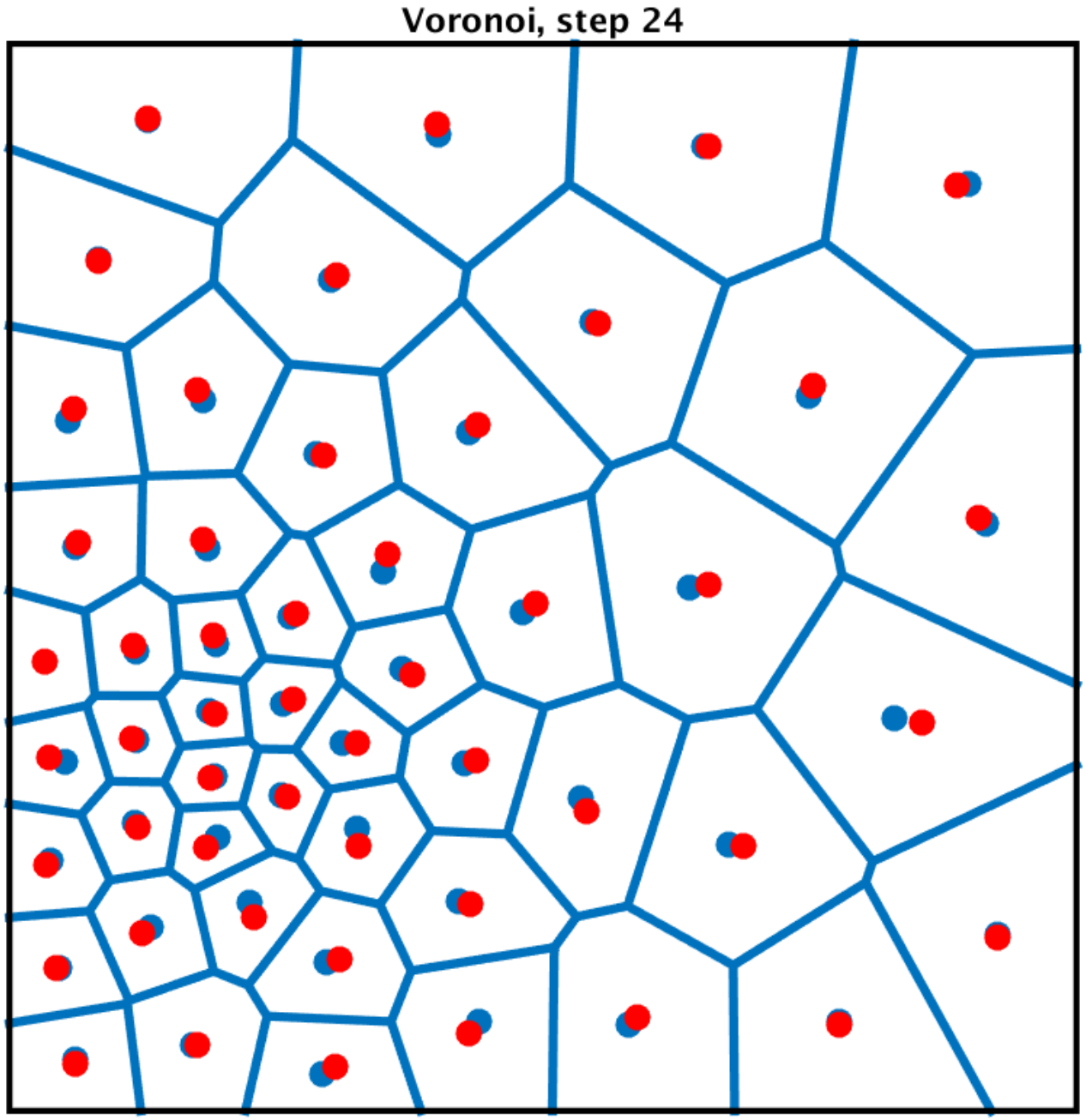

Voronoi, step 19

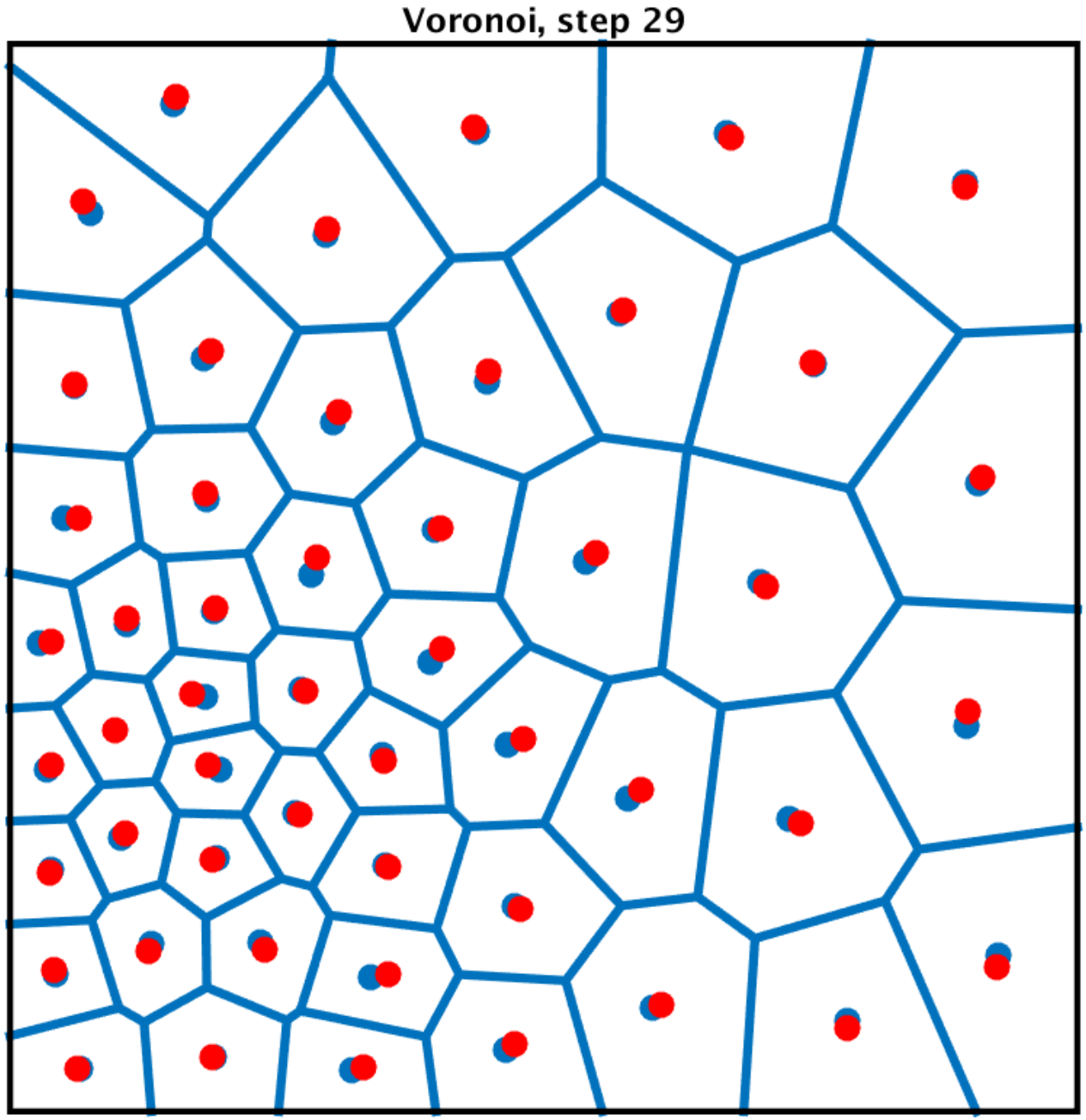

Voronoi, step 20

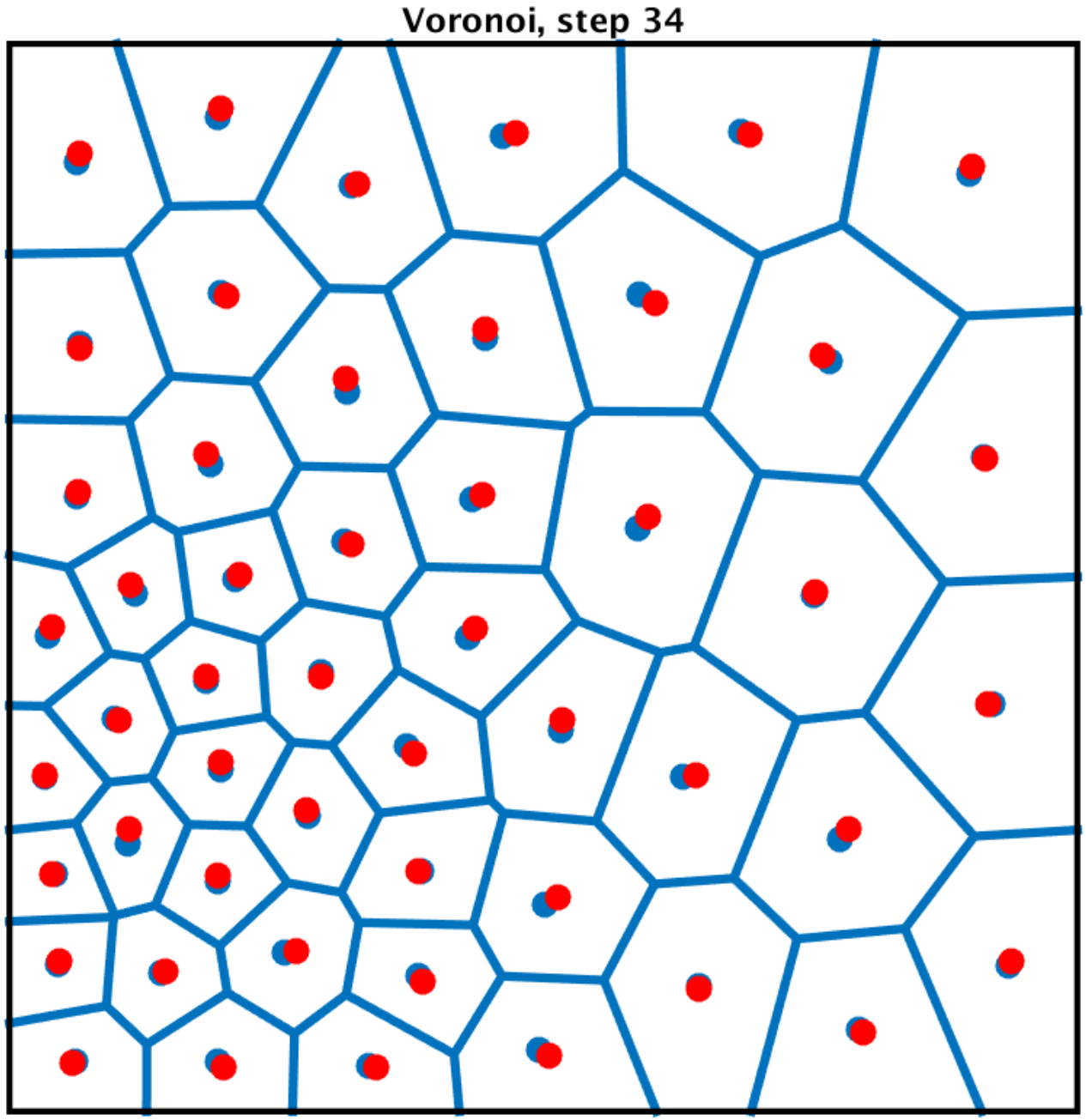

Voronoi, step 21

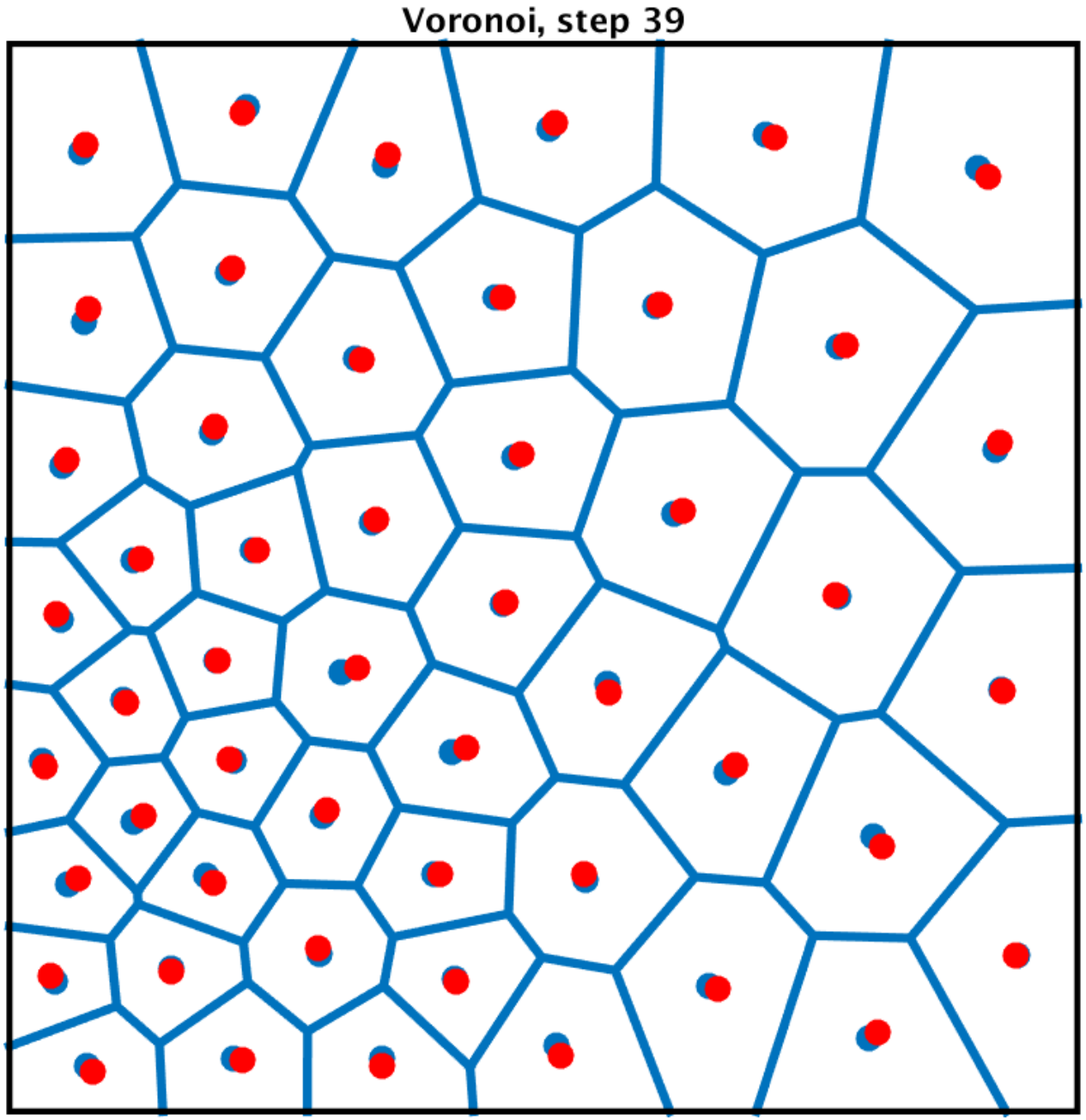

Voronoi, step 22

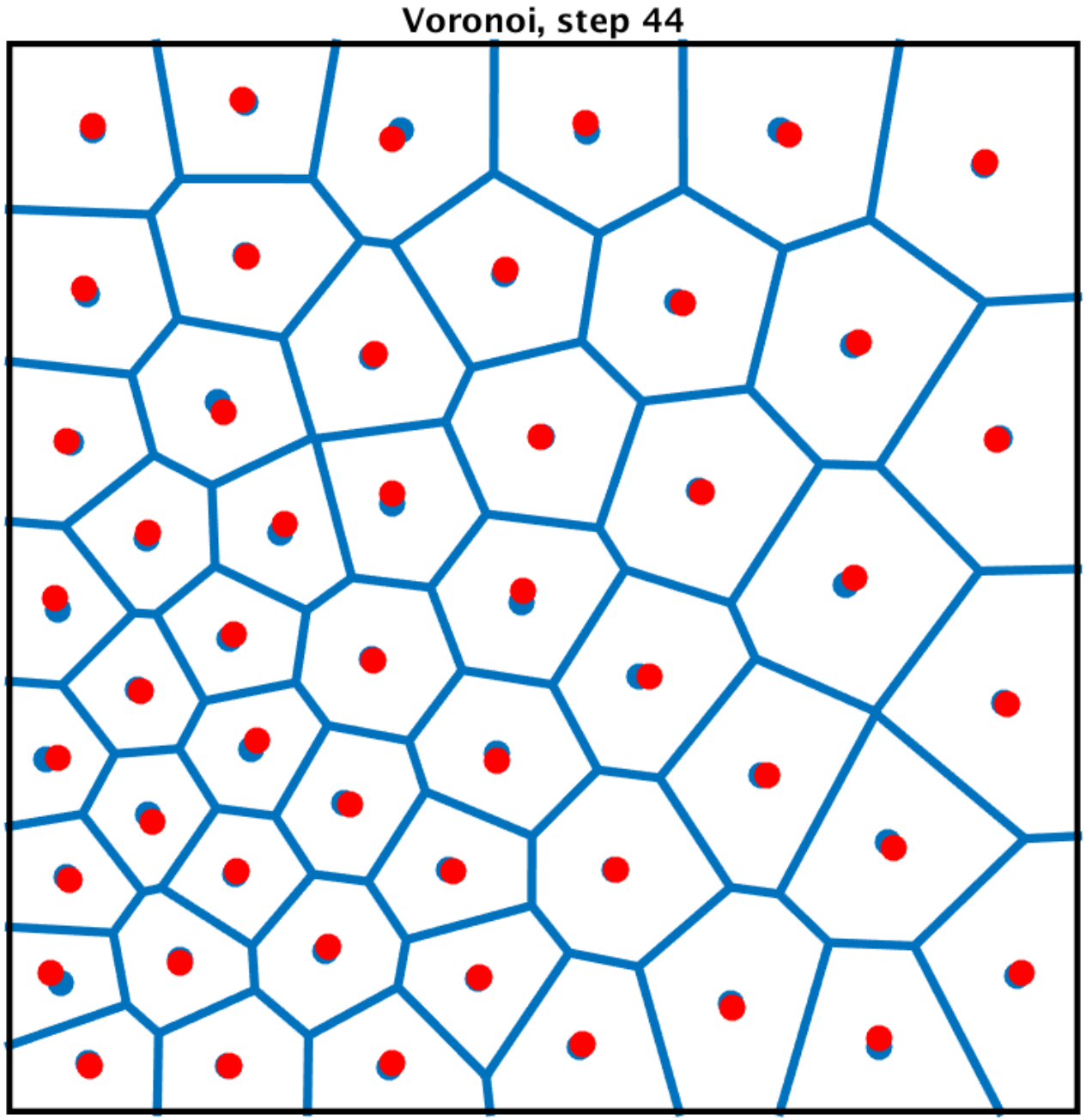

Voronoi, step 23

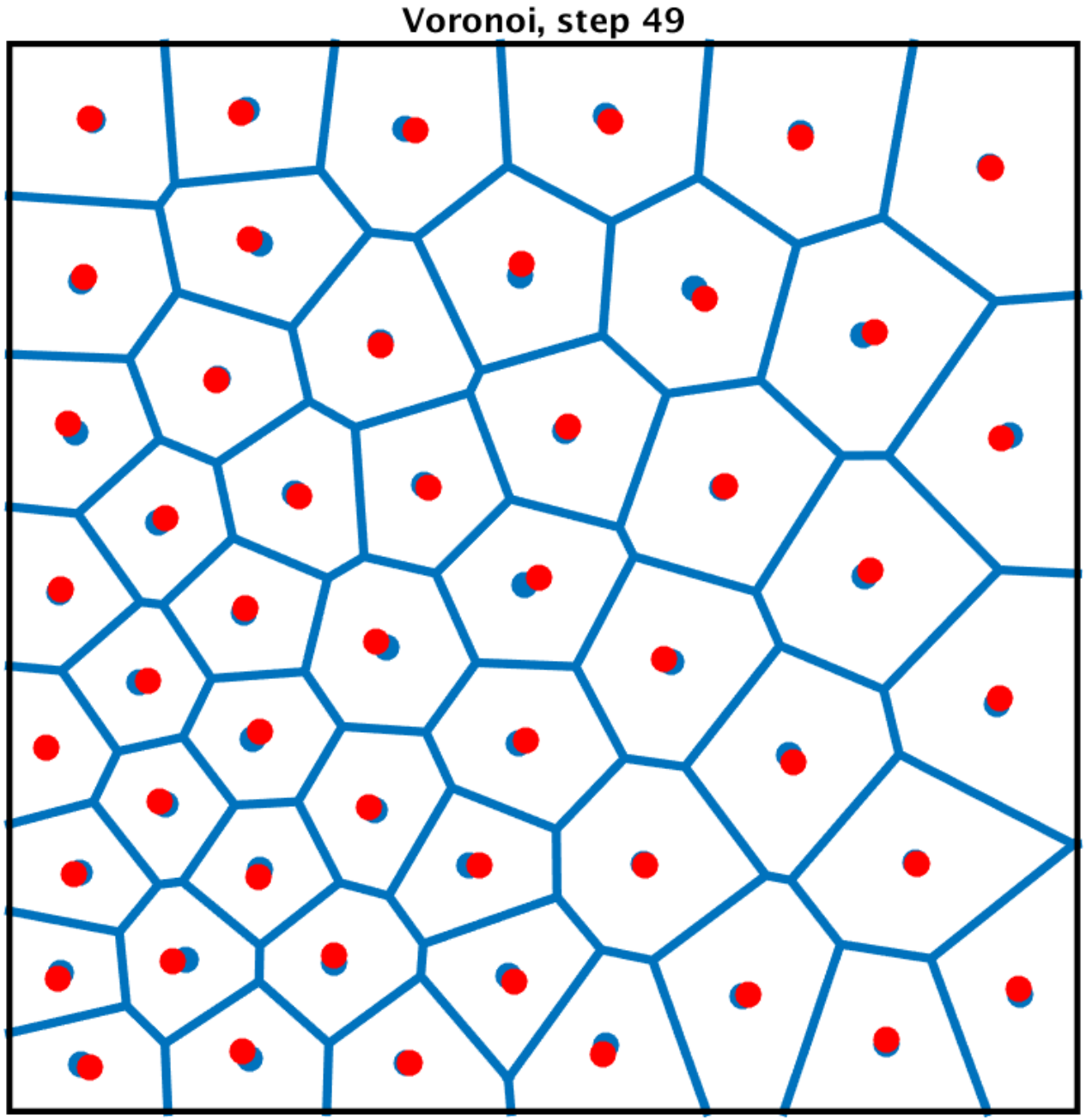

Voronoi, step 24

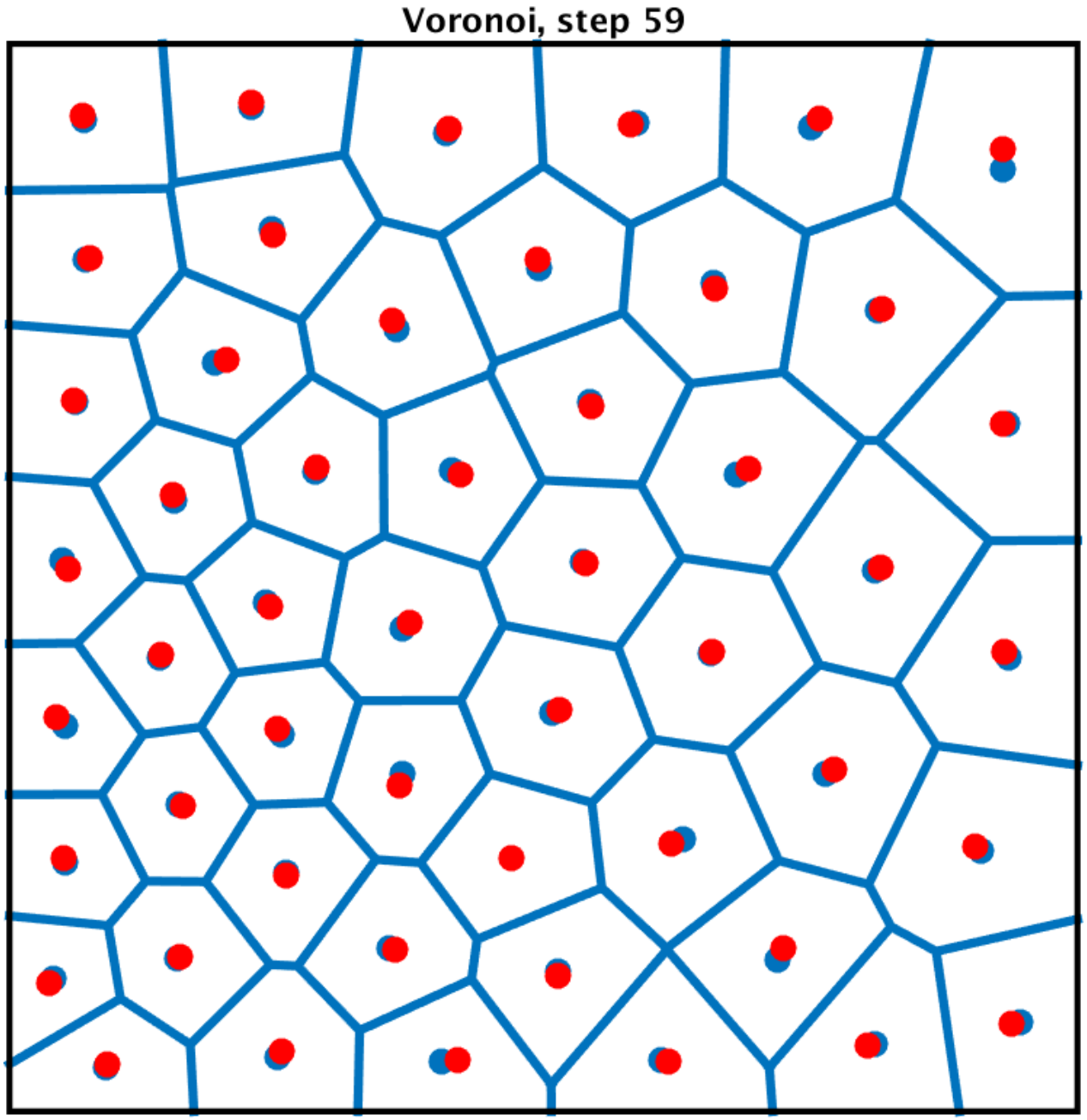

Voronoi, step 29

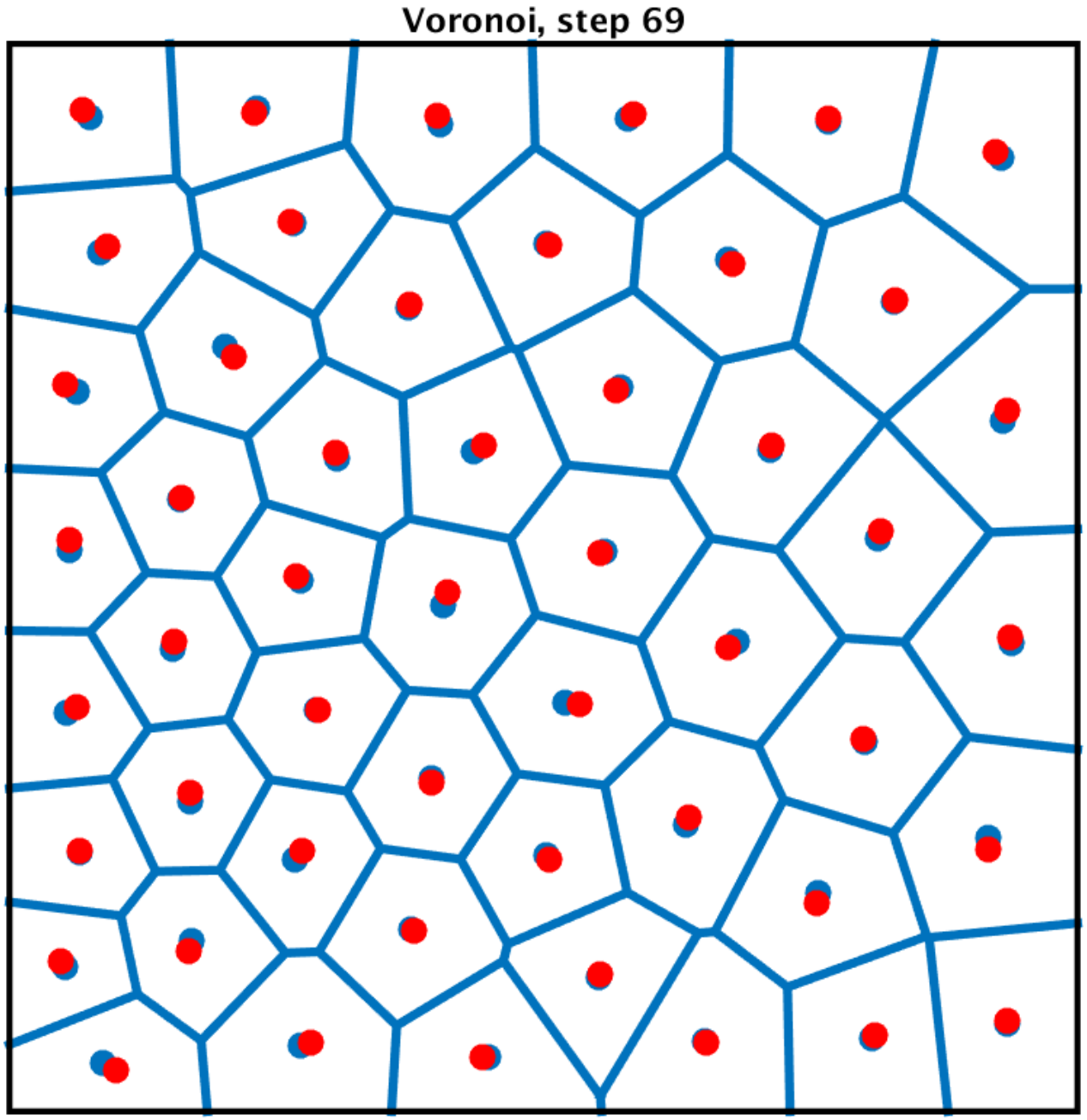

Voronoi, step 34

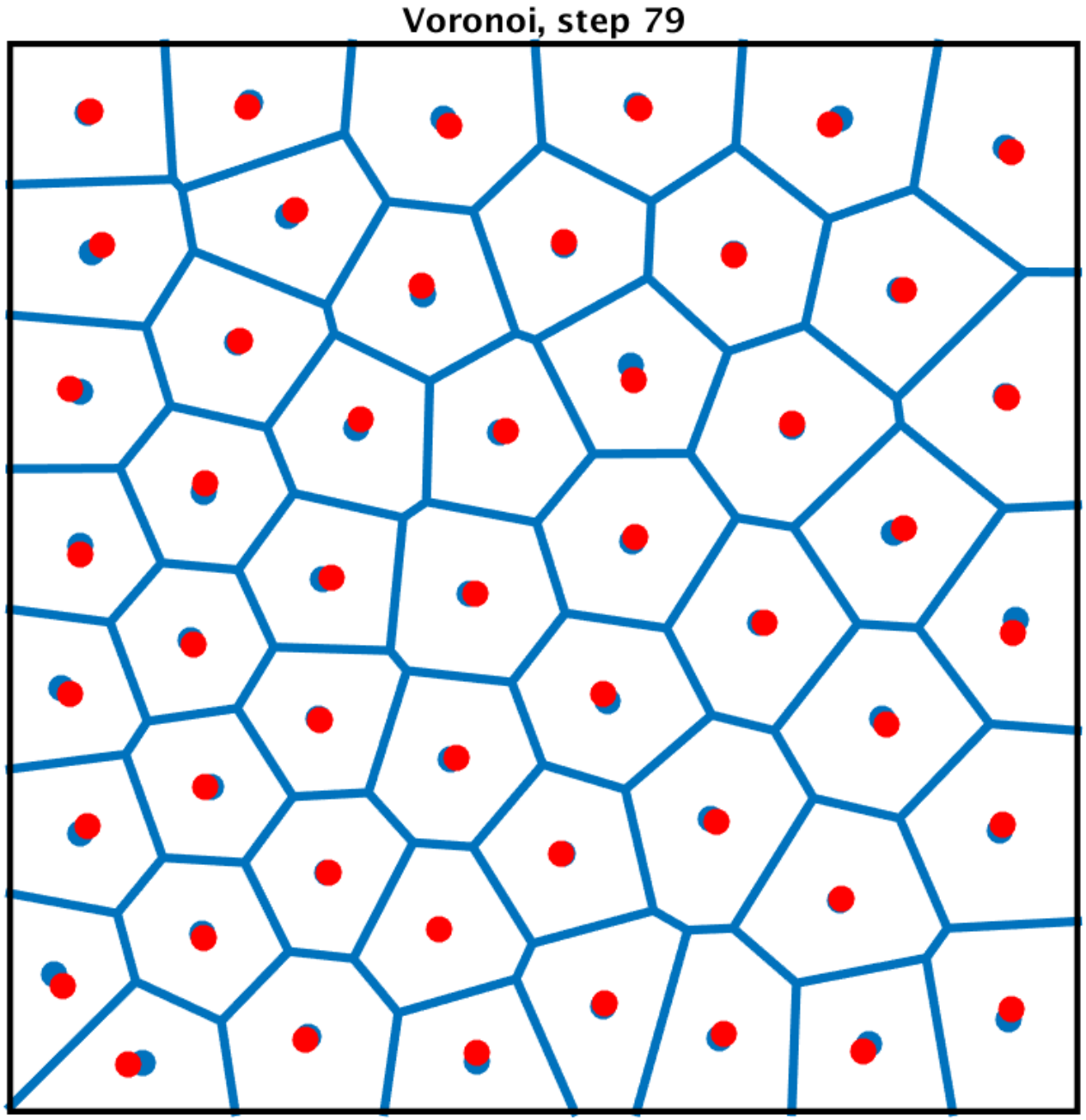

Voronoi, step 39

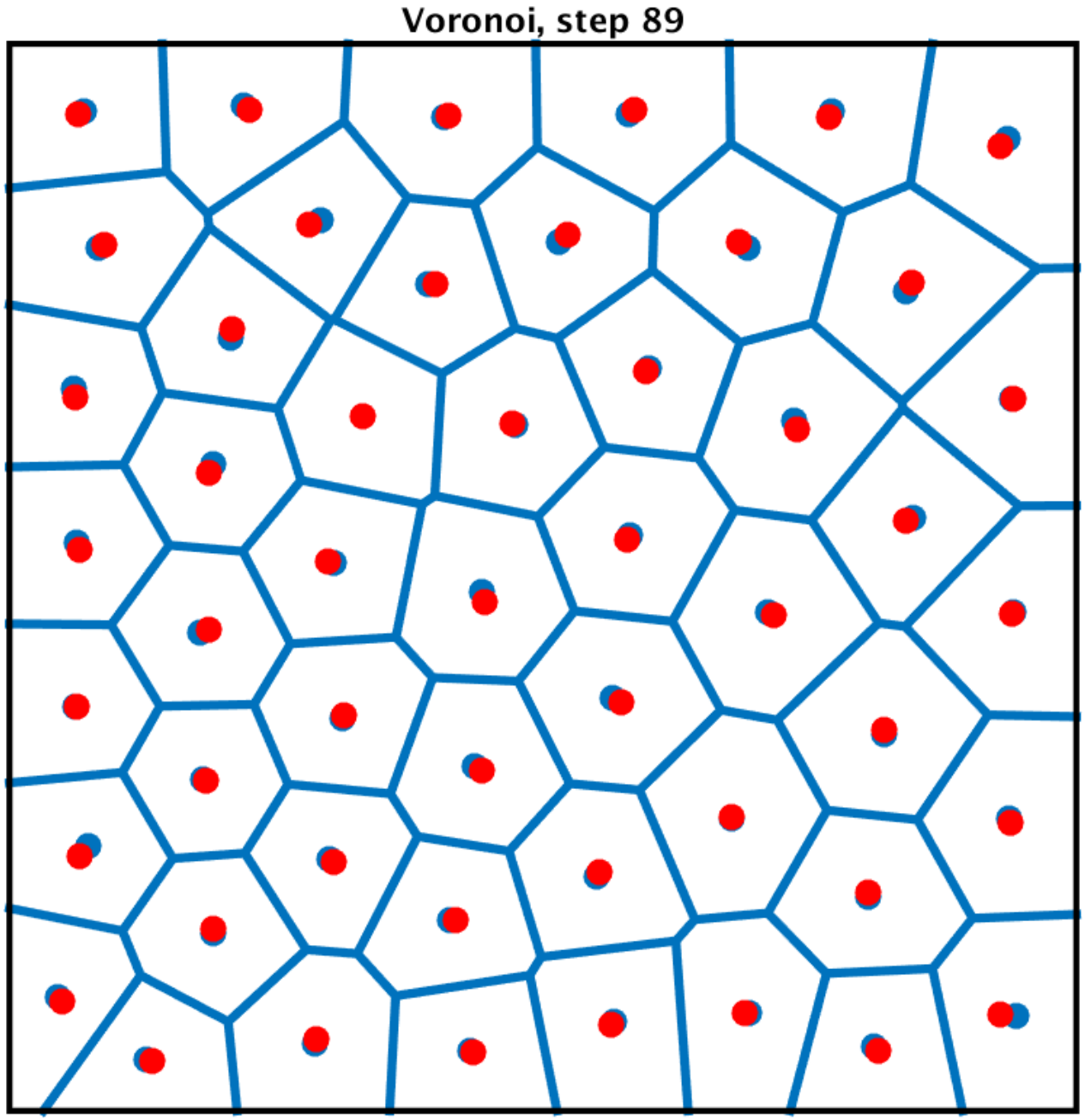

Voronoi, step 44

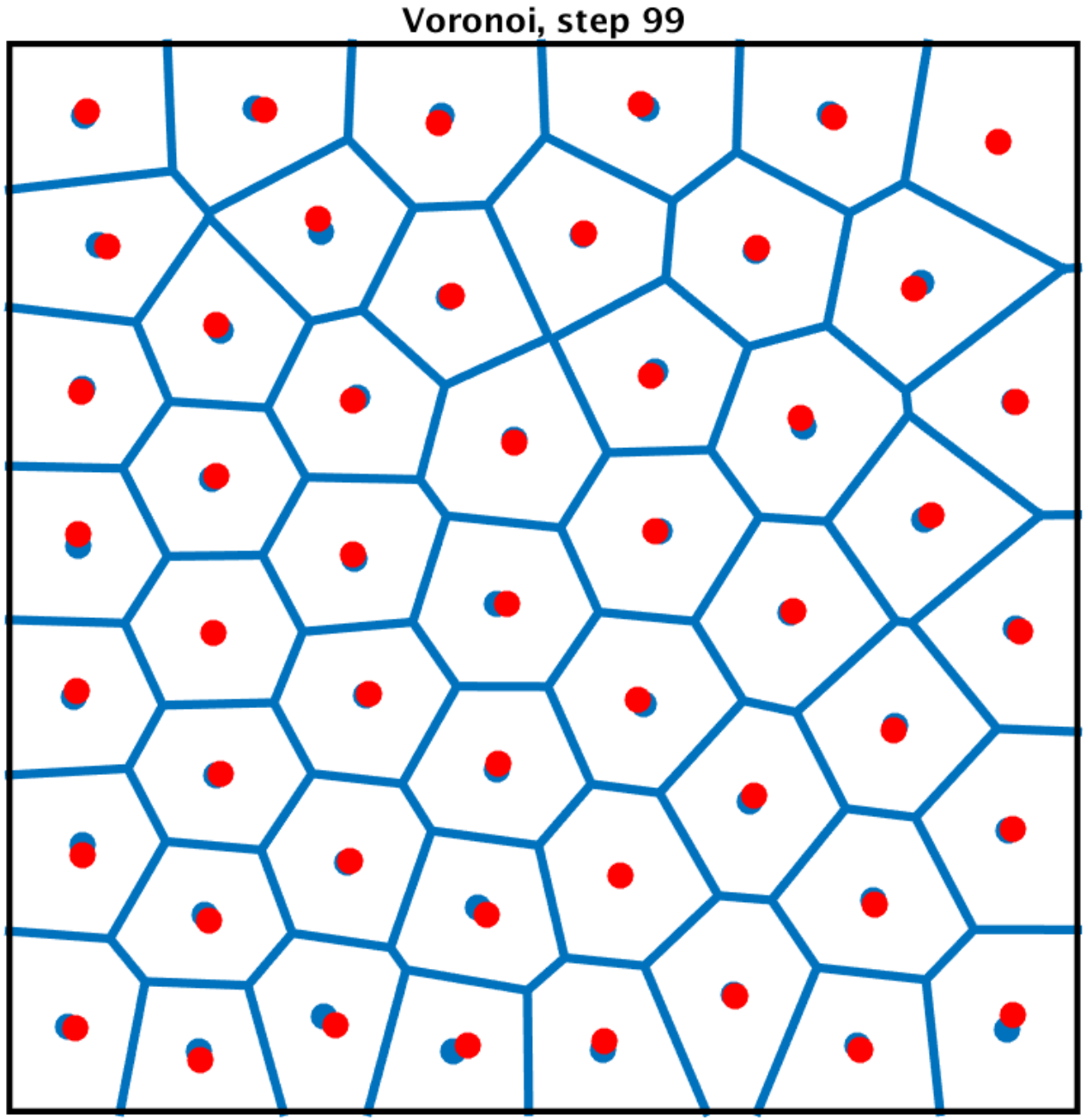

Voronoi, step 49


Voronoi, step 59


Voronoi, step 69


Voronoi, step 79


Voronoi, step 89


Voronoi, step 99

Notice, for the 30 mailbox case, how the final result is very regular:

- Each territory is about the same size;
- Territories along the boundary tend to have sides at right angles to the boundary.
- Territories away from the boundary tend to a hexagonal shape.

We have extended clustering so it can break a large region into small patches, and if we are allowed, we can rearrange the patches so they have nice shapes and roughly even sizes.

This suggests an answer to a serious political problem, namely, the division of each state into congressional districts, done every 10 years, after the national census.

When politicians determine these districts, they draw lines that guarantee they will be reelected, and that their party will get every possible advantage.

Our clustering process suggests ways this process could be made automatic, reducing the political bias.

We'll show later how clustering could change the Florida congressional map.