DIMACS

Center for Discrete Mathematics &
Theoretical Computer Science

DIMACS EDUCATIONAL MODULE SERIES

MODULE 03-7
A Gentle Introduction to Mathematical Cluster Analysis

September, 2003

Harel Barzilai
Department of Mathematics and Computer Science
Salisbury University
Salisbury MD 21801
phone: 410-543-6472
email: hxbarzilai@salisbury.edu
harel@barzilai.org

Alexander Kheyfits
Depatment of Mathematics and Computer Science
Bronx Community College (CUNY)
University Avenue at W. 181st Street
Bronx, NY 10453
phone: 718-289-5616
email: alexander.kheyfits@bcc.cuny.edu

Kathy Andrews
Hillsdale College
Department of Mathematics
33 E. College
Hillsdale, MI 49242
email: kjandrews42@comcast.net

DIMACS Center, CoRE Bldg., Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018
TEL: 732-445-5928 • FAX: 732-445-5932 • EMAIL: `center@dimacs.rutgers.edu`
Web: `http://dimacs.rutgers.edu/`

Founded as a National Science Foundation Science and Technology Center and a Joint Project of Rutgers University, Princeton University, AT&T Labs - Research, Bell Labs, NEC Laboratories America and Telcordia Technologies with affiliated members Avaya Labs, HP Labs, IBM Research, Microsoft Research.

Module Description Information

- **Title:**

  A Gentle Introduction to Mathematical Cluster Analysis

- **Author(s):**

  Harel Barzilai, Department of Mathematics and Computer Science, Salisbury University

  Alexander Kheyfits, Department of Mathematics and Computer Science, Bronx Community College (CUNY)

  Kathy Andrews, Department of Mathematics, Hillsdale College

- **Abstract:**

  Mathematical Clustering or Cluster Analysis is a field which endeavors to "classify". More precisely, given a discrete data set, this set is classified into groupings, or clusters. The criteria for what makes a "good" set of such clusters vary, however in general we want similar data points to be made part of the same cluster, and data points assigned to different clusters should have some important features distinguishing them.

  The basics notions and definitions of clustering are introduced, as well as key algorithms used in the field, including the notion of a "hierarchical" algorithm. The first chapter provides an overview of the field as well as an introduction to "Agglomerative" hierarchical algorithms; the second chapter describes "Divisive" hierarchical algorithms; and the third chapter introduces Sequential clustering algorithms.

  A section of print References is provided for further exploration, as well as as a section of Web References Cited (with full URLs), and additionally an online page, "Online Resources for Further Exploration" is available from the DIMACS page with sections on Tutorials, Sample Applications, Researchers with Expertise, and Software.

- **Informal Description:**

  The types of algorithms surveyed in this module are listed in the Abstract.

  Prospective readers may be interested to note that the wide variety of areas of application of this field include: business planning as well as public, city and regional planning such as choosing the locations for hospitals; taxonomic classification of species; classification of chemical molecules into types; and even web pages classified together as being of similar relevance to a search, or of similar interest to a reader.

  Students who complete this module will have been introduced to the fundamentals of clustering through an informal and intuitive approach which takes the reader gradually into the full formal definitions and algorithms. They will also glimpse some of the applications of this area, and will be provided with ample references for further study of both applications, and the mathematical field of cluster analysis itself.

- **Target Audience:**

  This module is meant to be an "entry level" introduction for freshmen and sophomore students. It can be used in Finite Mathematics, Discrete Mathematics and Applied Mathematics courses, as well as in "Survey of Mathematics" courses for strongly motivated liberal arts students. It can also serve for guided self-study by mathematically inclined high school students. The module may also be of interest in beginning computer science courses, particularly courses with sections which focus on algorithms

- **Prerequisites:**

  The formal prerequisites include only high school mathematics through algebra and functions or precalculus. Graph theory is not assumed, and all but he most basic set-theoretic notation is defined within the module itself.

- **Mathematical Field:**

  Cluster analysis; Classification theory; Graph theory

- **Applications Areas:**

  Classification theory; Biological, Chemical, and Ecological taxonomy and classification; Data Mining; Internet analysis and search engine algorithms.

- **Mathematics Subject Classification:**

  Primary 91C20

  Secondary 05C90, 90B80,92B10

- **Contact Information:**

  Harel Barzilai
  Department of Mathematics and Computer Science
  Salisbury University
  Salisbury MD 21801
  410-543-6472
  hxbarzilai@salisbury.edu, harel@barzilai.org

  Alexander Kheyfits
  Department of Mathematics and Computer Science
  Bronx Community College (CUNY)
  University Avenue at W. 181st Street
  Bronx, NY 10453
  718-289-5616
  alexander.kheyfits@bcc.cuny.edu

  Kathy Andrews
  Hillsdale College
  Department of Mathematics
  33 E. College
  Hillsdale, MI 49242
  301-552-6429
  keandrews@dmci.net

- **Other DIMACS modules related to this module:**

  Module 03-1 by Alex Kheyfits.

  (There may be others of which we are not yet aware)

Contents

ABSTRACT

Mathematical Clustering or Cluster Analysis is a field which endeavors to "classify". More precisely, given a discrete data set, this set is classified into groupings, or clusters. The criteria for what makes a "good" set of such clusters vary, however in general we want similar data points to be made part of the same cluster, and data points assigned to different clusters should have some important features distinguishing them.

The basics notions and definitions of clustering are introduced, as well as key algorithms used in the field, including the notion of a "hierarchical" algorithm. The first chapter provides an overview of the field as well as an introduction to "Agglomerative" hierarchical algorithms; the second chapter describes "Divisive" hierarchical algorithms; and the third chapter introduces Sequential clustering algorithms.

A section of print References is provided for further exploration, as well as as a section of Web References Cited (with full URLs), and additionally an online page, "Online Resources for Further Exploration" is available from the DIMACS page with sections on Tutorials, Sample Applications, Researchers with Expertise, and Software.

0 Preface

Our intended audience for this module are students at the freshman and sophomore level taking a General Education or Liberal Arts mathematics class. Accordingly, only a minimal, high-school background in mathematics is assumed. It is likewise assumed that most of these students will not major in the mathematical sciences, while recognizing that a many will enter fields which utilize mathematical thinking and tools.

Hence, our dual goals are that the minority which may enter a field in the mathematical sciences will be encouraged by this module to explore clustering beyond this initial introduction, and that all students expand their mathematical and analytical thinking skills, and appreciation for the usefulness and relevance of mathematics – particularly mathematical modeling.

Philosophically and pedagogically our aims are to foster through examples, applications, and references an appreciation that mathematics

i is *alive*: that there is active, current research going on

ii is *useful*: that this research concerns questions that people, including non-mathematicians, very much care about

iii is *interesting* (as suggested by the gallery of examples) and

iv is a *creative process* that non-specialists, including non-mathematicians, can engage in, and gain an appreciation for.

The intent is also to motivate and possibly entice the more mathematically inclined minority of Liberal Arts students (after having us whet their appetites with this module) to consider further mathematical explorations, be they in clustering, or in the broader world of mathematics. May you find mathematical adventure for your classroom in the pages that follow!

1 Overview and Agglomerative Clustering

1.1 Introduction

Suppose you are taking a university class which includes a long-term project as an assignment. You are to work in small groups on this project, and most of your group meetings will be held outside of class. Your professor, being kindhearted and thoughtful, wishes to choose group assignments in a fair way which will help you be successful.

Pause for reflection: At a non-residential university, what factors should the professor consider? (before reading on, stop here and take a moment to make up your own list of important factors).

One of the most natural criteria your instructor might use is geography, namely where you and your classmates live, and more precisely, how far apart you live from one another. By "far apart" we mean something more involved than distance, however. For example, it takes more time to travel 5 miles on a small road than on a fast highway. Other complicating issues are that parking costs may vary, and that the time spent waiting to switch from one bus or subway line to another depends on how many such switches are required. For our purposes, we will assume overall *travel costs* have been determined for getting from one student's home to that of a classmate's.

[Comment: some students may be familiar with the Triangle Inequality for distances: that the sum of the lengths of two sides of a triangle is greater than or equal to the length of the third side. It must be cautioned however that in this example, the Triangle Inequality does not hold since you will be using travel costs rather than simply physical distances. The general case in 'clustering problems' discussed in this module is that the Triangle Inequality will not hold; it will hold only in special cases of clustering].

Since your groups will need to meet many times outside of class, it would be a good idea to choose project groups (or 'clusters') in such a way that students who are in the same group live relatively "close" together (or at least not too "far" apart) in terms of the overall travel costs between pairs of students in the same group.

Now let's consider how your instructor might go about selecting project groups according to this criterion. Put yourself in the role of the instructor and think about how you might best accomplish the task of selecting the groups.

To be concrete, let us suppose that five students in class have chosen the same topic, but that project groups can only have two or three members each. Let's say that the five students are named Anna, Brian, Chris, Deborah, and Ethan, to whom we will refer by their first initials: A, B, C, D, and E.

Your task is to divide this set of five students into two project groups. Since groups with more than three or fewer than two members are not allowed, it follows that you need to create, out of the initial set of five students, two project groups: one group with two members, and one with three members.

1.1.1 Matrix Matters: Visualizing our data

Suppose now that the travel costs between the respective dorm rooms or apartments of our five students are as indicated in the following chart (Table 1):

Student					
A	0	1	8	5	4
B	1	0	10	6	7
C	8	10	0	2	9
D	5	6	2	0	3
E	4	7	9	3	0
Travel cost to:	A	B	C	D	E

Table 1: Travel costs between pairs of students

Notice the diagonal line of zeros in our chart: this is because the travel cost from A to A (that is, our "travel cost" if we start at Anna's place and our intended destination is also Anna's place) is zero: clearly, we don't need to travel at all in this case! Similarly, the table has a 0 for the travel cost from B to B, C to C, and so on. Notice that rows are listed in the same order as columns, and that this natural ordering is the reason behind the zeros always lying on a diagonal as we've noticed. In fact, there are other patterns in the above chart.

Pause for reflection: Before reading on, can you describe in writing what other patterns you see?

An important feature in the chart is its symmetry. Notice that there is a mirror-like aspect to the arrangement of numbers in our chart: it has two symmetric halves, separated by the diagonal of zeros we've observed. Do you see it? Furthermore, can you say and then write down in precise terms what this symmetry is, explaining in precise terms *why* this symmetry is present? Before reading on, take a moment or two to carefully describe this symmetry (*Suggestion:* thinking in terms of rows versus columns may help).

What we've found in the chart is more than just a visually pleasing pattern however; just as there was a reason and interpretation behind the diagonal of zeros (the travel cost incurred if your starting point and destination are the same, is zero), likewise there is a reason and a physical interpretation behind the symmetry of two complementary halves in our table of travel costs.

Pause for reflection: What interpretation of the table of travel costs explains specifically this symmetry?

Suppose we use the notation d_{CD} to represent the travel cost from Chris' place to Deborah's. According to our chart, d_{CD}=2. Making the reasonable assumption that in our case travel routes and their associated costs are the same in both directions, the travel costs from Chris' place to Deborah's place is equal to that from Deborah's place to Chris' place. It follows that in our notation, $d_{CD} = d_{DC}$

> [Comment: to avoid unnecessary complications at this point, we assume travel distances are equal in both directions. For towns with one-way streets one would have often had $d_{xy} \neq d_{yx}$ where d_{xy} is the driving distance from x to y].

Similarly, we know that $d_{AB} = d_{BA}$ and so forth. If you look at how the rows and columns are arranged and find the pair d_{AB} and d_{BA} on the chart, and similarly locate the spots where the pair d_{CD} and d_{DC} are located, you will have found the pattern behind the mirror symmetry. What is the general rule for this pattern?

1.1.2 Noticing more general patterns

To see more general patterns, we need to be able to talk about an arbitrary pair of students. To do this, let's suppose our five students are numbered 1, 2, 3, 4, and 5 in some ordering, so that if i and j are any two of these five integers, then we can speak of "student i and student j" without having to choose specifically to refer to "students A and B" or "students D and C" for example. In other words, while symbols such as "A" and "B" were just *abbreviations* for names (Anna, Brian, etc), the *indexes* like i and j are variables: they can vary over ranges of numerical values capable of representing any student. This way we can look for patterns that exist between an arbitrary collection of students. In fact, we can say the following.

Given that d_{ij} is the travel cost of going from the ith student's home to the home of the jth student (here i and j represent the indexes of two students among *any* group of students – not necessarily a group of exactly five students), then the symmetry we have noticed can be expressed by the equation $d_{ij} = d_{ji}$. Furthermore, we have one such equation for each pair of different indexes i and j (we already separated off the case when i = j).

Similarly, we can express the pattern of zeros along the diagonal by the set of equations $d_{ii} = 0$; there is one such equation for each i , that is, one such equation for each student in the group.

Since $d_{ij} = d_{ji}$ for each pair of indexes i and j, and since $d_{ii} = 0$ for each i, we can streamline our chart: no essential information will be missed if we omit the diagonal entries (we know those must be zero), nor if we include only one of the chart's two symmetric halves. This leaves us with the following simplified chart of travel costs between students (we keep the blank A-row and E-column to maintain symmetry and readability):

Student					
A					
B	1				
C	8	10			
D	5	6	2		
E	4	7	9	3	
Travel cost to:	A	B	C	D	E

Table 2: Travel costs between pairs of students

1.1.3 Forming Student Groups: who goes with whom?

Now let's return to our original question: what is the best way for breaking up this collection of five students into two separate project groups, one having two members, and the other with three? Since we're now back to dealing with exactly five students rather than attempting to understand the general case, we can again represent the students by their first initials (A, B, C, D, and E). As you will see below, we will use brackets to denote groupings of students together. We will proceed to group the students together into "clusters" – tentative candidate groupings for what the project groups *might be* – in a series of steps.

Initially, no student is assigned to be "in the same group as" any other student. Therefore, initially we start with five individual students, each grouped (with brackets) as a single-person "cluster": [A], [B], [C], [D], and [E]. This initial set of five students, each belonging to their own one-person cluster is represented in set notation as: {[A], [B], [C], [D], [E]}. Our goal is to end up with two project groups, one with two students, and one with three. For example, we might end up with project groups represented by [A,D] and [B,C,E], which would be the set of two clusters: {[A,D], [B,C,E]}.

Since the only criterion we have to work with at this time is travel cost, we would like to group students together who live *close* to each other in the sense of their being a lower travel cost between such students. If you look at the chart you will notice that Anna and Brian live 1 travel cost unit apart, and also that no other pair of students have a lower mutual travel cost.

So one reasonable first grouping (or clustering) of students would look like this:

First Clustering: { [AB], [C], [D], [E] }
A set of four tentative clusters make up this clustering.

This means that Anna and Brian are to be in the same group, and that the other students are not yet presently assigned to be "together with" anyone else. Until we further examine the chart for more information, it's not clear what the final project groups – that is, what the final clustering – might be. We could end up with {[AB], [CDE]}, but the final clustering might also be {[ABC], [DE]} or {[ABE], [CD]}, or {[ABD], [CE]}

Pause for reflection: Are there any other possibilities? How can you tell? How hard would it be to answer questions of this sort if instead of dealing with five students, we had 10? 30? In other words, given a larger initial set of students, and supposing two students are grouped together, with the rest being as yet ungrouped, can you see that the number of choices for final project groups (even restricting each final project group to 2 or 3 members, say) becomes rather large? Chose a size of initial number of students (e.g. 6, 7, or larger) and try to answer, or estimate, how many possible choices there would be for final project groups.

Now examine the chart again, and notice that the next smallest travel cost between two students is 2, which is the travel cost between C and D. Our eagerness to find pairs of students such that the travel costs between them is as small as possible suggests we may want to adopt the following reasonable *method* : we will "cluster together," at each step, those two students who are closest in travel costs.

Notice that what we are actually combining are the *groups* of these two students, since each student is technically a one-person group. This is an observation we will shortly use when we need to generalize our rule to a method of combining a student with a "cluster" (grouping) of other students. In any case, if we adopt our rule as it stands, then our next clustering will be:

$$\{[\mathrm{AB}],\ [\mathrm{CD}],\ [\mathrm{E}]\}.$$

Since single-member groups are not allowed, we are not done yet. Ethan (student E) needs to be placed into one of the two other groups: either together with Anna and Brian (the [AB] group or *cluster*) or together with Chris and Deborah (the [CD] cluster).

Since the rule defined by our method only tells us which *students* to put together, we need to expand it in order to decide where to place Ethan. Expanding on our observation above, we can expand our rule to be: "find the two students (not in the same cluster) such that the travel cost between them is least, and combine the *groups* to which these students belong".

After all, we started by finding the smallest two numbers in the chart (the "1" and "2"), so it's reasonable to look for the next smallest travel cost, which is 3. This is the value of dDE, which is the travel cost between Deborah and Ethan. Thus, if we put Ethan into the [CD] cluster, that is, into the same group with Chris and Deborah, then at least one of Ethan's groupmates (Deborah), will have travel costs to Ethan as small as possible. Thus, we have now arrived at the following clustering of students into two project groups:

$$\{[\mathrm{AB}],\ [\mathrm{CDE}]\}$$

According to this method we've created, the teacher would have Anna and Brian form one project group, and would declare Chris, Deborah, and Ethan to be a second group. We can now draw the following diagrammatic sequence to represent the stages of the process we used to arrive at the final project groups:

$$\{\ [\mathrm{A}],\ [\mathrm{B}],\ [\mathrm{C}],\ [\mathrm{D}],\ [\mathrm{E}]\} \rightarrow \{[\mathrm{AB}],\ [\mathrm{C}],\ [\mathrm{D}],\ [\mathrm{E}]\ \} \rightarrow \{\ [\mathrm{AB}],\ [\mathrm{CD}],\ [\mathrm{E}]\ \} \rightarrow \{\ [\mathrm{AB}],\ [\mathrm{CDE}]\ \}$$

Is there more to this story? Should we now, pleased with this result, march into our professor's office and declare that we have solved all of their problems of assigning student project groups?

One point to notice is that sequence in the diagram above is not be the only possible order in which the groups can be formed. Had we a tie, say, $d_{AB} = d_{CD} = 1.5$, then the second stage would have been either as above, or {[A], [B], [CD], [E]} (the first stage would still be {[A], [B], [C], [D], [E]}).

Exercise: Had the chart of travel costs been the same as above except that $d_{AB} = d_{CD} = 1.5$, draw a complete diagram for the rest of the process for each of the two possible "second steps". Would they both end with the same final choice for groups?

Note: It turns out that for algorithms different than the one just used, one does not always arrive at the same final choices for groups, when ties create several avenues for the "next step." There are many ways to handle "ties" of this type. One possibility would be to simply arbitrarily choose one of the tied values and treat it as the larger of the two (by flipping a coin, for example). For simplicity, in this module we will, from this point on, always assume that all of the travel costs between pairs of students are different from one another, and thus avoid this complication in our overview of examples and methods. See the list of references for more general treatments of clustering which include such situations. In particular, see section 3.2.6 in [JD].

As we see next, even with this simplification, there are other questions and doubts that arise upon closer consideration of the methods we have used so far for choosing project groups.

1.1.4 Did we choose wisely? A Second Look

Our stepwise *agglomeration* , that is gathering together, of the initial set of five separate students into larger clusters, was a step-by-step process which created partitions of the five-student set with fewer and fewer "pieces". That is, there were fewer clusters at each step (at each successive cluster*ing*) as the size of some of the clusters increased, and this process certainly ended as it was supposed to: our final clustering consisted of exactly two clusters, one having two students, and other having three students.

If this were the *only* requirement, there would be no room for improvement. Recall however that we were also interested in the travel costs between the students, which we wished to minimize in some reasonable sense. Since the meetings of the project groups take place outside of class, at student homes, we wanted to spare our students from having to make "expensive" trips – in terms of time, money, and other factors measured by the travel costs – as they went about having their frequent meetings with fellow project group members.

With regard to this criterion, our method certainly yielded in some sense a *reasonable* answer: Anna and Brian, who have the lowest mutual travel cost, are in the same group, and furthermore, Chris and Deborah, who have lower mutual travel costs than any other pair other than Anna and Brian, are also clustered into the same project group. Finally, we put Ethan in Chris' and Deborah's project group, because the travel costs between Ethan and Deborah is less than for any other pair besides Anna and Brian, and Chris and Deborah. What could be better? But wait! Why does Ethan have a frown on his face? And Chris doesn't look too happy either.

Pause for reflection: Why the frowns? Consider carefully the information we have. What do you think the problem might be?

The travel cost between Chris' place and Ethan's is 9, the second highest pairwise travel cost of *all* the travel costs between pairs of students(*). Thus when meetings are held at Chris' house, Ethan has a difficult trek to make, and similarly when meetings are held at Ethan's, the travel cost for Chris is equally large!

> [Comment: Notice that the travel costs from Chris to Ethan (9) is greater than the sum of the travel costs from Chris to Deborah (2) plus the travel costs from Deborah to Ethan (3). This might happen, for example, if there is a time cost such as switching

between two modes or lines of public transportation when one tries to get from Chris to Ethan via an intermediate stop at Deborah. Thus it is now clear that the Triangle Inequality does not hold for this example since, as you may recall from Euclidean geometry, the *triangle inequality* states that given three points A, B, and C, then dist(A,C) $\leq$ dist(A,B) + dist(B,C). That is, the distance from A to C is less than or equal to the sum of the distance from A to B, plus the distance from B to C. When dissimilarities *do* represent distances, then the set of dissimilarities have this additional, "triangle inequality" property: $d_{i,k} \leq d_{i,j} + d_{j,k}$, that is, the dissimilarity between O_i and O_k is less than or equal to the sum of the dissimilarity between O_i and O_j added to the dissimilarity between O_j and O_k. In general, however, the matrix of dissimilarities need not have this property, because underlying dissimilarities need not represent distances. For example, molecules A and B might be judged by chemists to have dissimilarity 5, and molecules B and C might be judged to have dissimilarity 1, but the dissimilarity between molecules A and C might be 8 – greater than 5+1]

This does not mean that we should automatically throw out all of the preceding work. Nor that we should throw out the informal *algorithm* , or procedure, which we used to guide our stepwise formation of clusterings of students.

But let's think about the problems created for Chris and Ethan which arose from our above algorithm, which we came up with "on the fly."< Do these problems suggest alternative criteria we might use in our clustering procedure? For example, why not try to minimize how "bad" a trip between *any* two members of the candidate project group would be, when we consider lumping students together to form larger project groups? Might this be a goal worth adopting?

[Comment: in general, when solving a problem, we might get different solutions or answers – different final clustings in this case – depending on the goal(s) or objective(s) we adopt]

If we proceed with this modified goal in mind, some interesting differences arise in how our ultimate clusters (project groups) turn out: we get a different sequence of clusterings, and more precisely, we get a different final clustering of the 5 students.

To see this, we follow this new criterion and observe what happens. Initially, we would still cluster Anna and Brian together:

$$\{ [A], [B], [C], [D], [E]\} \rightarrow \{[AB], [C], [D], [E] \}$$

This is due to "1" being the cost of the least costly trip between any two students. Next, we still cluster Chris and Deborah together since "2" is the next least costly trip. So, as before, we have the following sequence of clusterings:

$$\{ [A], [B], [C], [D], [E]\} \rightarrow \{[AB], [C], [D], [E] \} \rightarrow \{ [AB], [CD], [E] \}$$

Now things become more interesting. The next smallest travel cost is "3" and is between Deborah and Ethan. However as we saw, the travel cost between Chris and Ethan is 9 – pretty bad. Can we do better? With so many possible combinations, where do we even begin? How might we proceed?

Our idea is to try to make the "worst" trip be the least costly one we can manage – to see how low a travel cost we can arrange for such a worst trip, while still finding ourselves able to arrive at a final clustering. Under our rules, this means being able to arrive at one cluster (project group) with three students, and another with two.

So far we have Anna and Brian assigned to be in the same group, and Chris and Deborah are assigned to be in the same group. Suppose we wish to require that the "worst" trip have travel cost no more than 3. Would this work? Clearly not, since adding Ethan to Chris and Deborah's group would fail to meet this requirement (Ethan and Chris are 9 travel cost units apart), nor could we add Ethan to Anna and Brian's group (Ethan is 4 and 7 units away from them, respectively). And since four member groups are not allowed, we don't need to consider the possibility of merging [AB] with [CD].

Next, rather than giving up, we relax our requirement a bit. Suppose we try to require that the "worst" trip be no more than 4 travel cost units. We still can't add Ethan to either of the two member groups – can you explain why not? Next, we try 5 for the "worst allowable trip". Why doesn't this work? And why does 6 not work either?

Not giving up, we now try to make 7 the "worst allowable trip". Now, we *can* add Ethan to Anna and Brian's group, since Ethan lives 4 units away from Anna, and 7 units – just barely fitting within our requirement – from Brian. Thus the worst travel cost for any member of this proposed group would be 7.

Notice that under our new criterion, deciding whether to have Ethan join Anna and Brian's group, or to have him join Chris and Deborah's group is not a difficult choice – we would not be tempted to add Ethan to Chris and Deborah's since as we noticed earlier, Ethan and Chris live 9 travel cost units apart, so the "worst trip" for a proposed group of [CDE] would not be 7 units or less.

Thus our new clustering criterion is this: "at each step, minimize how bad [in travel cost] the 'worst possible trip' is within a proposed project group". As we just saw, under this second clustering procedure we get a different sequence of clusterings than the one created by our first method; namely, we get the following sequence of clusterings:

$$\{ [A], [B], [C], [D], [E]\} \rightarrow \{[AB], [C], [D], [E] \} \rightarrow \{ [AB], [CD], [E] \} \rightarrow \{ [ABE], [CD] \}$$

Thus we have ended up with a different final set of clusters (final clustering): [ABE] and [CD]. We are now better able to provide our instructor with algorithms for creating project groups, depending on what goals are considered most important. We have two methods or algorithms, which we created using *ad hoc* methods – in other words, by "following our noses" and intuition rather than by using a systematic approach and analysis. Our two methods seem decent, however we would certainly feel more confident in them if we had a better sense of what other types of clustering outcomes are possible, and if we had a better feel for "the lay of the land" for methods of clustering. Is there are more careful, systematic way to think about it?

In the following sections, we will give more formal, that is more careful and precise definitions of the concepts we've already encountered, and of algorithms for clustering an initial set of elements, such as students, into a collection of groups, called clusters, such as project groups for students. But first, let's slow down for a moment to take a brief first look at the overall enterprise we have embarked on: the field of cluster analysis.

1.1.5 Is there Method in our Madness?

The attentive student might wonder at this point which of the above two methods of clustering the five students is the "right" one. Unfortunately there is no single, simple answer to this question. In fact, you might start to wonder just how many possible methods there are besides the two we stumbled upon. When looking at a clustering problem, do we just try to discover (make up?) new criteria for each new application that arises, with each criterion or method leading to a different solution? And in our opening example, should we search for a third method, a fourth, and so on, leading to a large number of possible clusterings, none of which we can say with any confidence is "the right one"?

These questions are indeed fundamental to understanding the field of clustering. While we cannot answer all of them in full detail here(*), several general remarks can be made at this point.

> [Comment: we encourage interested readers to further investigate these issues. For suggested reading, see the References.]

First, the reader should understand that researchers in the field of cluster analysis do *not* proceed by means of random, *ad hoc* methods invented on the spot. How then are methods chosen? Part of the answer is that, as we shall see, one can approach the question of "what criteria are important?" in a more systematic and precise manner. This more precise notion of "what is important" in terms of criteria can then aid in selecting an appropriate method of clustering. For example, the teacher might decide that, in the interest of fairness, the "average ability" (suitably defined) of any one group should not differ too sharply from that of another project group. Or, a professor might want diversity of student backgrounds within each group. Any criterion, that is, any given and carefully defined choice of desired properties of the clustering, can serve as an aid for choosing among different possible clustering methods, and even for identifying new clustering algorithms.

Different applications and contexts, in which we are clustering things other than students, lead to other natural criteria. For example, suppose that a country, state, or province has enough funds in its budget to build 10 new hospitals in a given region. Where should the hospitals be located? One can obtain data as to where people live, including the population densities in different areas. And one can make an assessment of the degree of need in any one area (perhaps with a greater percentage of older citizens) versus another. These and similar data then form the basis from which more mathematically precise clustering criteria, and ultimately, clustering algorithms, can be created by researchers.

Still other applications arise in industry (where should factories and outlets be built, given consumer preferences and population concentrations?), psychology (given a set of a dozen "scores" from different personality tests, which individuals should be clustered as being of similar "type"? or which sets of personality tests should be clustered as "measuring similar qualities"?), transportation (where should airports be located? How do we "cluster" customer transportation demands into a manageable number of flights and connecting flights?), sociology (who tends to mingle with whom during a social gathering, in a cafeteria, or other location or event?), chemistry (e.g. molecular classification into clusters of "similar" molecules, see references in [1], biology (see [2] and [3] for clustering in the taxonomic classification of species as well as genetics [4], ecology [5], phylogenetic algorithms [6], and even the internet [7]. For example, how can a software program "cluster together" web sites into groupings so that two web sites in the same group (cluster) are likely to be equally interesting and useful – or likely to be equally uninteresting – to the user? (see [CP] in references section). For additional applications, such as image processing, see chapter 5 of [JD].

With this overview in mind, we are now ready to delve more deeply into the world of clustering. As promised, we begin by making our definitions more precise.

1.1.6 Defining our Terms

To be able to sharpen our analysis, we need to be able to put the objects and relations we have been studying into clearer focus, and this is why we need more precise definitions. In the following collection of definitions, modeled after [HJ], each newly defined term is put in *italics* .

First, we need to know what we wish to "cluster". Thus, we need to start with some underlying set; for example, the underlying set in the prelude's opening example was the initial set of 5 students. Such an underlying set in a clustering problem is called the *sample* . Since the elements of the set are quite often objects rather than people, we will usually use the capital letter O for the sample set, and its elements or objects will be denoted with subscripts, for example:

$$\mathrm{O} = \{\mathrm{O}_1, \mathrm{O}_2, \mathrm{O}_3, \mathrm{O}_4, \mathrm{O}_5\}$$

might denote the underlying set (or sample) in our opening example. In general, we may have in the sample O any number of objects (also called *entities*), and if there are N objects, then we write:

$$\mathrm{O} = \{\mathrm{O}_1, \mathrm{O}_2, \mathrm{O}_3, \ldots , \mathrm{O}_{N-1}, \mathrm{O}_N\}.$$

Here O_1 is the "first object," O_2 the "second," and so forth, while O_{N-1} is the "second to last," and O_N is the last of the N objects in the sample being considered. However, the ordering itself is generally unimportant. Its usefulness is simply to allow us to give different names to the different objects, so we can speak unambiguously about which object is clustered together with which: for example, O_2 being clustered together with O_8 rather than O_3 with O_5.

In order for the clustering of the objects in our underling sample to be performed in a way that is not arbitrary, this sample must also come with some way of measuring how "alike" or else, how "dissimilar" any two objects are. When our "objects" were the students in the opening example, the dissimilarity between two objects (students) which we used was the travel cost between their homes.

In other applications, the numerical dissimilarities between objects can have a different meaning. An example is when we are trying to cluster communities into sub-regions or districts, so that one clinic may be built for each district, with each one being considered a cluster. For such an application, dissimilarities might be travel costs in dollars, or in time, between the communities, or actual distances between the communities needing medical care, or some other numerical measure of how different the communities are from one another. Or perhaps a complex measure is created by doctors, statisticians and others, taking into account age distributions, income distributions, the effect of the number of high-speed highways on getting an ambulance to and from a community, and so forth.

One important thing to keep in mind is that dissimilarities need not be distances in the usual sense of distance between geographical places (think of the phrase "distant relatives" for some of your cousins). As we have seen, they come from a wide variety of "measures" which are used when experts in a given field have solid evidence suggesting the measure reasonably captures critical differences between objects in the sample. Other examples of measures of similarity/dissimilarity between elements of a set include the differences in base pairs between strands of DNA , measures of "closeness" between species and their anatomy, and differences between the chemical properties of molecules. In general however, regardless of the application, when we speak of the dissimilarity between two objects, we are speaking of a positive *number* , for dissimilarities are represented as numerical values.

The set of dissimilarities between all possible pairs of objects in a sample is normally given in the form of a matrix. This is just a two-dimensional table whose rows correspond to the elements of O, and whose columns likewise index the elements of O. This table or chart is then filled with numbers: the number in the 2nd row and 3rd column is labeled $d_{2,3}$ and represents the *dissimilarity* between the 2nd and 3rd objects in O. That is, the dissimilarity between O_2 and O_3 is $d_{2,3}$ and generally, the dissimilarity between O_i and O_j is written $d_{i,j}$. This matrix of dissimilarities taken as a whole is called the *dissimilarity matrix* for the sample O.

All of this should have a rather familiar ring to it: the chart in our five-student opening example was nothing but a dissimilarity matrix for our (rather small) sample set of five elements of O, and the "objects" in O were just the students (and as before, we will assume that our dissimilarity matrix contains *no ties*).

Furthermore, from our opening example it's clear that the matrix of dissimilarities will have two crucial properties: $d_{i,i} = 0$ for any i (the dissimilarity when we compare O_i with itself is zero), and for any pair of indexes i and j, $d_{i,j} = d_{j,i}$ (the dissimilarity between O_i and O_j is identical to the dissimilarity between O_j and O_i). Finally, we also require in our definition of a proper dissimilarity matrix that $d_{i,j} \geq 0$. This means that dissimilarities cannot be negative. After all, the dissimilarity, that is the degree of difference between two objects cannot be less than "nothing" –"nothing" meaning that no differences exist– and so cannot be less than zero. Note that if our sample space O has n elements, then our dissimilarity matrix will be an n by n matrix.

The reader should note that all clustering methods depend critically on the matrix of dissimilarities. Such dissimilarities are based either on distances, or on non-distance "proximities," meaning degrees of qualitative closeness or similarity between objects. It is emphasized in [JD] for example that "unless a meaningful measure of..proximity, between pairs of objects has been established, *no meaningful cluster analysis* is possible. The proximity matrix is the *one and only* input to a clustering algorithm" (emphasis added. Note that, whether we think of the entries of the matrix as expressing levels of "dissimilarity" or levels of "similarity"/"proximity," are two sides of the same coin, just as "how big is it?" or "how small it is?" are two ways of looking at a measurement like "size is 4 feet").

The next logical step, with our underlying sample set O and the matrix of dissimilarities now fleshed out, is to give a precise definition of clusters. A *cluster* C is just a subset of O. For example, if O = $\{O_1, O_2, O_3, O_4, O_5\}$, then C = $\{O_3, O_5\}$ might be a cluster.

Thus $C_1 = \{O_1, O_2, O_3\}$ and $C_2 = \{O_4, O_5\}$ can correspond to the two clusters (project groups) into which the five students were divided in our first attempt, each C_i being a cluster. Notice however that when we started with the original set O of five students, what we were after, and what we desired as "the answer" was neither the cluster C_1 nor the cluster C_2. Rather, it was the collection of these two clusters, which together told us how the original group of students was to be divided, which we were after.

A *clustering* (used synonymously with *partition*, see [JD]) is a collection of disjoint subsets (that is, a collection of disjoint clusters) into which the given set O can be divided. More precisely, we have the following definition.

A *partition* of a given set O is a collection of (nonempty) subsets $\{C_1, C_2, ...C_M\}$ such that:

1. If i does not equal j, then C_i and C_j are disjoint; that is, they do not have any elements of O in common. In mathematical notation, we write $C_i \bigcap C_j = \Theta$, where Θ denotes the empty set.

   and such that also:

2. Every element in O belongs to one (and by the preceding property, necessary to only one) of the clusters. In terms of sets, this means that O is equal to the union of the all the clusters in the partition. In mathematical notation this is written:

$$O = C_1 \bigcup C_2 \bigcup ... \bigcup C_M$$

A *clustering method* is then a rule which, given any initial sample set O along with an associated matrix of dissimilarities $\{d_{i,j}\}$, tells us how to produce, perhaps through a sequence of steps, a clustering $\{C_1, C_2, ...C_M\}$ as its final output. Here, the number of clusters in the ultimate clustering, M, depends both on the sample O, and on the dissimilarity matrix we are given in each instance.

Finally, a *clustering algorithm* is a mathematically precise step-by-step procedure for implementing a clustering method. It is written as if in a hypothetical programming language (that is, it's written in what computer scientists call *pseudo-code*), such that were it run on a computer, and given the set O and the matrix $\{d_{i,j}\}$ as inputs, it would produce the clustering determined by the method as its output. In practice, the pseudo-codes given for various clustering algorithms in research articles are used as guides for writing actual computer programs (written in a variety of existing computer languages) which implement the algorithm.

At last we have at set of specific and precise definitions! Still, you may wonder what lies inside the "black box" of such clustering algorithms. In the next section, we will look at the inside workings of two of the most common clustering algorithms. In fact, as we will see, these precise algorithms will put our first two "ad hoc" methods in a new light.

1.2 The Missing Links:

Introducing two common clustering algorithms

It is important to observe that the process of applying our first informal, ad-hoc algorithm to the five-student example produced more than a single, final clustering. Rather, it produced a *sequence* of clusterings, culminating only in the last step with the clustering which determined a possible set of students project groups:

$$\{ [A], [B], [C], [D], [E] \} \rightarrow \{ [AB], [C], [D], [E] \} \rightarrow \{ [AB], [CD], [E] \} \rightarrow \{ [AB], [CDE] \}$$

We can view this process as a sequence of four partitions (four clusterings) of O:

$$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$$

Furthermore, notice that there is a definite relationship among these four successive clusterings; at each clustering, the *clusters* it consists of are "the same as or bigger than" those making up the preceding clustering. More precisely, the clusters in succeeding clusterings (partitions) are gotten by taking the set-theoretical unions of clusters which had made up preceding clusterings. A sequence of clusterings having this property is said to be a sequence of *nested* clusterings.

Clustering algorithms which produce not merely a final clustering from the initial set O, but a sequence of nested clusterings are, in fact, quite natural, since we might be interested in more than just "the last" clustering such an algorithm produces. For example, if the professor in the five-student example later decides that one and two member groups are acceptable, the second to last clustering, { [AB], [CD], [E] }, might be the best choice for assigning student project groups. Therefore, the entire four-step output $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$ is useful, and it's up to the professor to decide which of the four P_i – that is, which of the four clusterings – is the one to be used for assigning project groups.

In general, a clustering algorithm which produces a nested sequence of clusterings is called a *hierarchical* clustering algorithm. This is because it produces a "hierarchy" of possible clusterings: a sequence is produced, starting with clusterings which have a large number of clusters (each containing a relatively small number of objects), and proceeding to clusterings which are made up of a small number of clusters (each of which contains relatively many objects). Hierarchical algorithms which proceed in this order, i.e., which proceed from a large number of 'small' clusters, to a small number of 'large' clusters, as opposed to the opposite order, are called *agglomerative* (In Chapter 2 we will introduce the "other kind" of hierarchical clustering algorithms, called *divisive*).

Both of the algorithms which we'll consider in this section are hierarchical; in fact, they will both be agglomerative. Informally speaking, such agglomerative algorithms proceed by building the clusters "from the ground up" with set-theoretical unions.

More precisely, in such algorithms one first starts with one extreme clustering (partition of O), namely the clustering in which each object forms its own one-object cluster. Then, as the algorithm proceeds, clusters are merged to form larger clusters, until the last clustering (the final partition of O) is reached. This last clustering is the other extreme; it is the single-cluster clustering in which all objects are placed into one cluster. Of course, this cluster is necessarily the cluster which contains all the objects: O itself. Giving a precise definition of such an agglomerative clustering algorithm turns out to be rather straightforward, as we are about to see.

1.3 Single and Complete Linkage: the Inside Scoop

1.3.1 Single Linkage

As we have just seen, an agglomerative hierarchical clustering method begins with the partition (clustering) $P_1 = \{\{O_1\}, \{O_2\}, ..., \{O_N\}\}$. Thus, $P_1 = \{C_1, C_2, ..., C_N\}$, where C_1 is the single object cluster $\{O_1\}$, C_2 is the single object cluster $\{O_2\}$, and in general $C_i = \{O_i\}$.

The single linkage algorithm defined in this section, and the complete linkage method discussed in the next section, are agglomerative hierarchical methods of clustering (henceforth referred to more briefly as *agglomerative algorithms* or *agglomerative methods*). Therefore, under these algorithms we start with $P_1 = \{C_1, C_2, ..., C_N\}$, where $C_i = \{O_i\}$. In successive steps, each of these algorithms merges clusters together, until the last clustering P_q is arrived at, namely $P_q = \{C_1\}$ where $C_1 = O = \{O_1, O_2, ..., O_N\}$, the single cluster containing all objects.

An agglomerative algorithm is determined by its method of joining or agglomerating clusters together. Put differently, the differences between *any* two agglomerative methods, and in particular between the single and complete linkage methods, lie in the different criteria which specify which clusters are to be merged at each step.

Before describing the criteria used for these two algorithms, it is important to note that the contents of the ith cluster Ci are not fixed, but depend on the *stage* of the clustering algorithm. For example, if A, B, C, D, and E correspond respectively to O_1, O_2, O_3, O_4, and O_5, then we may view the now familiar sequence of four clusterings:

$$\{[A], [B], [C], [D], [E]\} \rightarrow \{[AB], [C], [D], [E]\} \rightarrow \{[AB], [CD], [E]\} \rightarrow \{[AB], [CDE]\}$$

As our sequence of clusterings $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$ and we may write them in the notation:

$$\{C_1, C_2, C_3, C_4, C_5\} \rightarrow \{C_1, C_3, C_4, C_5\} \rightarrow \{C_1, C_3, C_5\} \rightarrow \{C_1, C_3\}$$

For example, at the first stage, in P_1, we have C_1=[A] and C_2=[B], but in the second stage the (single member) clusters [A] and [B] are merged together to form the new cluster C_1=[AB] in P_2 (we have chosen an obvious renaming scheme for the C_i to minimize relabeling of indexes).

In the implicit but omitted last step, P_5={[ABCDE]} and we have a single cluster, C_1=[ABCDE]. In the above, it was our knowledge of this particular case – knowing that we seek student project groups having either two or three members – and not a formal "rule for stopping" which made it clear that P4 was the cluster to examine. The general algorithm, as noted, produces a full spectrum of clusterings, one for each stage: from the fine extreme where $C_i = \{O_i\}$ to the coarse extreme with just one $C_1 = \{O_1, ...O_N\}$.

[Comment: A somewhat more complex and more general notation is often used, in which C_{ij} is the jth cluster in the ith clustering, as in [JD]. In such cases commas are added when necessary to avoid ambiguity: since C_{112} could mean the 12th cluster in the first clustering or the 2nd cluster in the 11th, one adds a comma and writes either $C_{1,12}$ or $C_{11,2}$. We will maintain the simpler notation C_i in this exposition, with the understanding that the ith cluster this represents will depend on which clustering, that is, on which step in the clustering algorithm we are].

Now, what criterion does the single linkage method use for merging two clusters together? Actually, both the single linkage and complete linkage methods progress from the clustering P_i to the clustering P_{i+1} by merging together those two clusters which are "closest together"; the difference between the two algorithms is in how the term "closest clusters" is defined. More precisely, the key is how the "dissimilarity" between two *clusters* (not merely between two objects) is defined.

Under the single linkage procedure, the dissimilarity between two clusters C_i and C_j is defined to be the *smallest* dissimilarity between any two objects, one of which must lie in C_i, and the other in C_j (as before, we assume there are no "ties" in the dissimilarity matrix). This measure of dissimilarity between C_i and C_j can be computed as follows:

1. First, consider all pairs of objects, one of which lies in C_i (call it O_k), and the other lies in C_j (call it O_l)
2. Second, look up the dissimilarity between the two objects in *each* such pair (if the two objects are O_k and O_l then this dissimilarity is the number $d_{k,l}$), and lastly
3. Find the *smallest* of all of the dissimilarities found in step 2, and declare this number to be the dissimilarity between C_i and C_j.

Hence we are taking the minimum of all the possible dissimilarities (numbers) $d_{k,l}$ as k and l range over all possible indexes for which it holds that $O_k \in C_i$ and $O_l \in C_j$. Using "SL" to denote "single linkage," in mathematical notation we have the SL-dissimilarity between clusters C_i and C_j, written dissimilarity$_{SL}(C_i, C_j)$ or $d_{SL}(C_i, C_j)$ which is defined by:

$$d_{SL}(C_i, C_j) = minimum\{d_{k,l}\}$$

where the minimum runs over all indexes k and l such that $O_k \in C_i$ and $O_l \in C_j$. In explicit mathematical notation, this is:

$$d_{SL}(C_i, C_j) = \text{minimum}\{d_{k,l}\} \text{ over all k,l such that } O_k \in C_i, O_l \in C_j$$

or more concisely:

$$d_{SL}(C_i, C_j) = \min_{O_k \in C_i, O_l \in C_j} \{d_{k,l}\}$$

Our definition can be rewritten in set-builder notation since the set of values in whose minimum we are interested is $\{d_{k,l} | O_k \in C_i, O_l \in C_j\}$

[Comment: Recall that if A is a set and P is a condition which might be satisfied by some of the elements of A, then $S = \{x \in A|P\}$ stands for a subset of A, namely, for the set of all elements in A to which condition P applies. For example, if $\mathbf{Z}$ represents the integers, then $S = \{n \in \mathbf{Z}|2n + 3 = 5\}$ is the set of all integers which satisfy the given equation, while the set of all integers which are multiples of 3 may be written as $S = \{n \in \mathbf{Z}|n = 3k \text{ for some } k \in \mathbf{Z}\}$ that is, the set of all integers which can be written as three times some other integer].

Thus we have:

$$d_{SL}(C_i, C_j) = \text{Min}\{d_{k,l}|O_k \in C_i, O_l \in C_j\}$$

A pseudo-code algorithm implementing single linkage is therefore:

1. Initialization: begin with $P_1 = \{\{O_1\}, \{O_2\}, ..., \{O_N\}\}$. This defines P_n when n=1.
2. To obtain P_{n+1} from P_n, merge C_i and C_j together for the pair of indexes i and j for which $d_{SL}(C_i, C_j)$ is *smallest*.
3. If the resulting partition (i.e., cluster*ing*) is $P_{n+1} = \{O\}$, then stop. Otherwise, increase n by 1 and then return to step 2.

Exercise: Convince yourself that the single linkage method applied to the opening example of the five students yields precisely the clustering hierarchy we initially obtained,

{ [A], [B], [C], [D], [E]} → {[AB], [C], [D], [E] } → { [AB], [CD], [E] } → { [AB], [CDE] }

At which stage, having the desired groupings, we could have the algorithm halt; if allowed to complete the algorithm would end with { [AB], [CDE] } → { [ABCDE] }. After you've convinced yourself, write a clear explanation showing why this is so. Your exposition should be one which a sceptical but fair fellow student would fully understand and be convinced by, and should include explicit computations with dSL.

1.3.2 Complete Linkage

We proceed exactly as above, the only distinction being how the difference or "dissimilarity" between two clusters C_i and C_j is defined. For the complete linkage (CL) method, this is defined to be not the minimum but the *maximum* of the dissimilarities between pairs of objects, one of which belongs to each of the two clusters under consideration. That is, we define

$$d_{CL}(C_i, C_j) = \text{ maximum}\{d_{k,l}\} \text{ over all k,l such that } O_k \in C_i, O_l \in C_j$$

Or more concisely:

$$d_{CL}(C_i, C_j) = \max_{O_k \in C_i, O_l \in C_j} \{d_{k,l}\}$$

In set-builder notation this becomes just:

$$d_{CL}(C_i, C_j) = \text{Max}\{d_{k,l}|O_k \in C_i, O_l \in C_j\}$$

A pseudo-code algorithm implementing complete linkage is therefore:

1. Initialization: begin with $P_1 = \{\{O_1\}, \{O_2\}, ..., \{O_N\}\}$.This defines P_n when n=1.
2. To obtain P_{n+1} from P_n, merge C_i together with C_j for the pair i and j for which $d_{CL}(C_i, C_j)$ is *smallest* (notice that what we are doing is taking the *minimum* among those "maximum values" which go into the definition of d_{CL}– make sure you understand this!)
3. If the resulting partition (clustering) is $P_{n+1} = \{O\}$, then stop. Otherwise, increase n by 1 and then return to step 2.

Exercise: Convince yourself that the complete linkage method applied to the opening example of the five students yields precisely the clustering hierarchy we obtained in our second attempt (with our second *ad hoc* method), namely, that it would result in the sequence we obtained:

{ [A], [B], [C], [D], [E]} → {[AB], [C], [D], [E] } → { [AB], [CD], [E] } → { [ABE], [CD] }

followed by a final step, { [ABCDE] }. (We had halted before this final step during our second *ad hoc* attempt, since we were initially merely exploring how to obtain 2- or 3- member project groups). As in the preceding exercise, after you've convinced yourself, give a clear and rigorous demonstration (including computations with d_{CL}) which would convince a skeptical but fair fellow student that the complete linkage algorithm would yield precisely this second clustering if applied to the initial group of five students and matrix of dissimilarities. Note the parallel structure between the Single Linkage and Complete Linkage algorithms above. See Algorithm 1 in [M] for a way of informally describing a general agglomerative algorithm.

1.4 Getting Visual: Dendrograms

A *threshold dendrogram* is a way of depicting the clusterings in the order in which they are formed. For example, if we represent

{ [A], [B], [C], [D], [E]} → {[AB], [C], [D], [E] } → { [AB], [CD], [E] } → { [AB], [CDE] } →{[ABCDE]}

by

$$P_1 \to P_2 \to P_3 \to P_4 \to P_5$$

when the associated threshold dendrogram would look as shown in Figure 1:

Here we have included together with the threshold dendrogram a vertical axis depicting each level P_i.

[Comment: Although we shall not use this concept, a *proximity dendrogram* would label the vertical axis according to the dissimilarity level at which the clustering first occurs. For example, at a proximity of "1" only the [A] and [B] clusters merge in the SL algorithm, so the location of P_1 on the vertical axis could be labeled "1" since allowing only mergings of clusters having distance less than or equal to 1 would result in only [A] and [B] being merged. At higher levels, more clusters will have been merged, and at a sufficiently high level, one is at the "top" of the dendrogram. For us, the threshold dendrogram suffices, since in this, our basic example, the threshold levels are just consecutive integers. The student interested in learning more about dendrograms may consult some of the suggested readings in the references provided at the end]

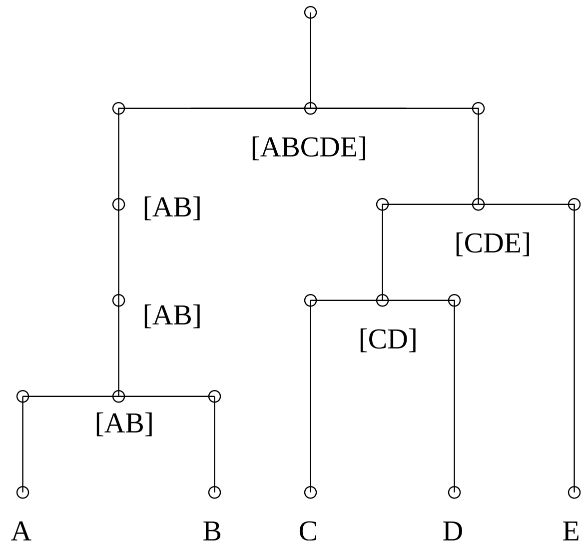

Figure 1:

Exercise: Draw a threshold dendrogram for:

$$\{\ [A], [B], [C], [D], [E]\ \} \to \{\ [AB], [C], [D], [E]\ \} \to \{\ [AB], [CD], [E]\ \} \to \{\ [ABE], [CD]\ \} \to \{\ [ABCDE]\ \}$$

Note: you will want to re-order the students and use an "E, A, B, C, D" ordering instead of "A, B, C, D, E" so that the lines of your diagram will not cross.

Exercise: Create your own dendrogram on a five (or more, if you wish) element set. Then convert it into the equivalent sequence of clusterings.

1.5 Getting Graphical

It turns out that there are useful connections between clustering and a branch of mathematics called *graph theory* . You may recall that in mathematics the term *graph* refers to an object which can be represented as a set of points together with a set of line segments between some of these points.

> [Comment: A graph in this sense is, of course, *not* the same as the notion in algebra of the *graph of a function* which visually represents a relationship of the form y=f(x)]

In a graph, the points are called *vertices* while the line segments are called *edges* . The set of vertices together with the set of edges which connect some of them, jointly constitute the graph. Just as the sample set is often given a label "O" similarly a graph as a whole is often labeled with a capital letter such as "G" (if there are two or more graphs, then "G" and "H" may be used to label them, or else, "G_1" and "G_2" etc). The vertices of the graph are typically depicted as points, or as circular *nodes* . For example, a graph G might look like the image shown in Figure 2:

Note that the edges which are depicted as dashed lines are regular edges, and the dashed-line style is merely used to visually indicate one straight line segment (i.e., edge) being "behind" another edge without the two edges crossing or "touching".

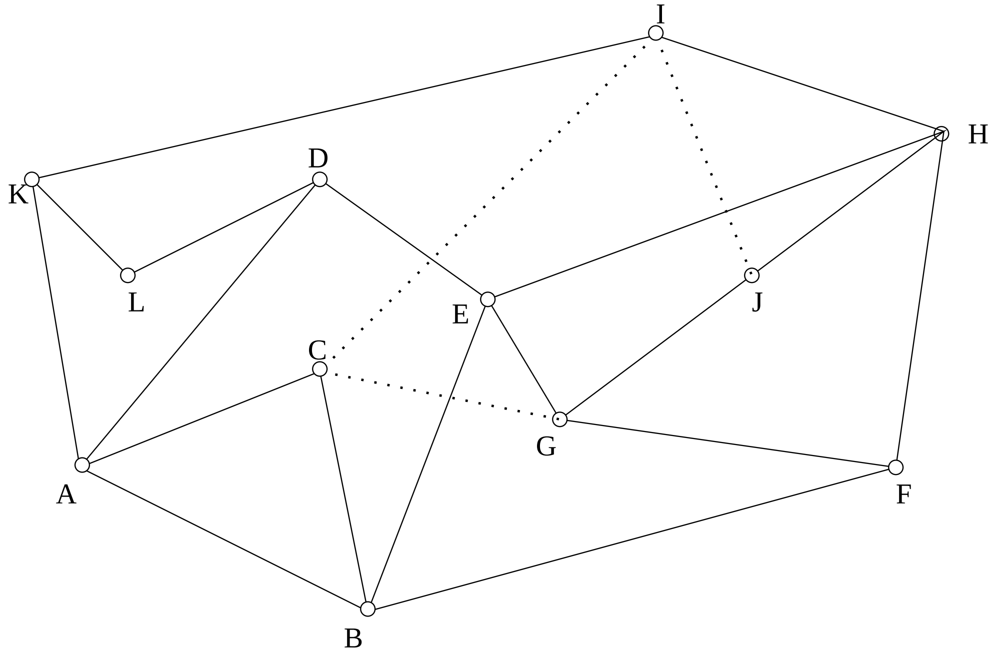

Figure 2:

Often there are names or *labels* which we wish to associate with the vertices, with the edges, or with both. Take our opening example and our five students Anna, Brian, Chris, Deborah, and Ethan, to whom we usually refer by their abbreviations A, B, C, D, and E. We might schematically represent these five students, and the travel costs between any pair of them, by using figure 3:

Note of caution: the edges in this graph are abstract representations of *relationships between vertices* , e.g. of the travel costs between pairs of students, as given by their respective labels. Thus the *visual* length of the edges as they are 'drawn', and whether they are drawn straight or curved (or "crossing"), etc, *do not* represent any meaningful features of the situation we are modeling, any more than does the geometrical arrangement of *where* each vertex in the graph is placed when we draw the graph – we could have depicted vertices B and C on the right instead of on the left, for example. The graph only tells us: "there are five vertices with these labels, and there are the following edges between vertices, coming with their corresponding labels as marked". (In the rare situations in mathematics when a readers needs the lengths of edges to perfectly equal their labels in the drawn graph, a special note to the reader would indicate that this was done).

In the graph above, non-mathematical information could have been used to label edges, e.g., "Bus route #2" or the like. A graph such as this one, in which mathematical quantities are used to label the edges – in our case, these mathematical quantities are the travel costs between two students – are sometimes called *weighted graphs* since each edge's label can then be thought of as a "weight".

1.5.1 At the Threshold of Something New

Now suppose you are considering a clustering problem having elements labeled O_k for k between 1 and N, and with dissimilarity indexes $d_{i,j}$ for the dissimilarity between O_i and O_j. A *threshold graph* associated with a clustering problem is a weighted graph whose vertices are the objects and

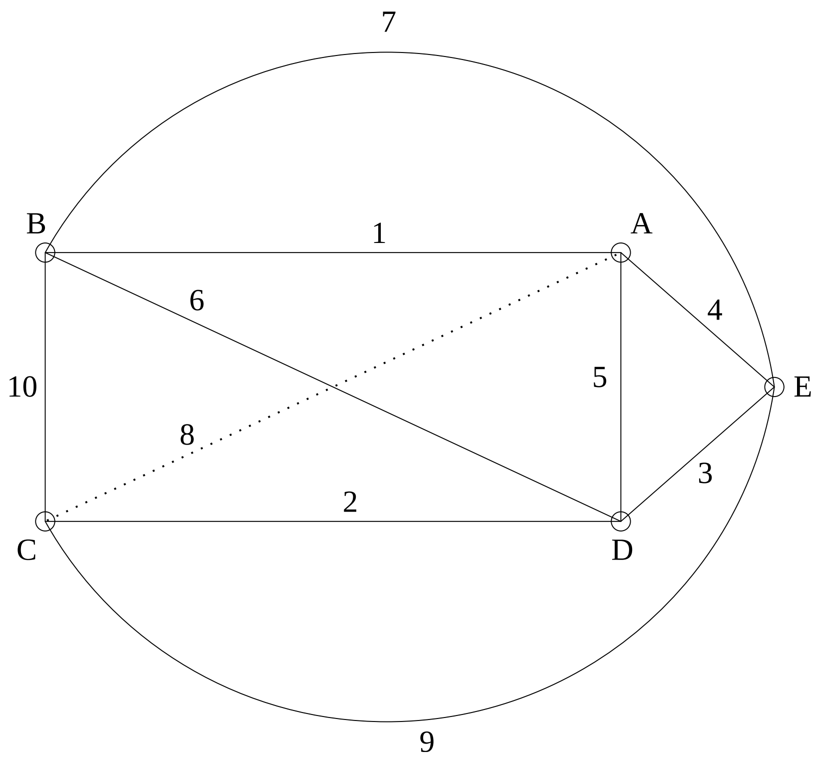

Figure 3:

whose edges represent all the dissimilarities that are less than or equal to some fixed level. For example, a threshold graph with *threshold* 5 would have one vertex for each object, and its edges would correspond exactly to those dissimilarities that are less than or equal to 5. A notation often used for the threshold graph with threshold v is G(v). Figure 4 depicts in six parts the threshold graphs for our opening five student example, as v ranges from 1 to 6:

Exercise: Draw G(v) for v = 7, 8, 9, and 10. What is G(11)? What does G(0) look like? G(-1)? Can G(v+1) differ from G(v) by more than one edge? For example, under what circumstances, if any, would two or more edges to be needed to augment G(1) into the larger G(2)? Can you give an example? (Remember: even though we are assuming "no ties," the values of the dissimilarity matrix, and thus the labels of the threshold graph, need not be integers!)

1.5.2 Some More Graphical Terminology

We are almost ready to put this framework from graph theory into use: we will be able to create algorithms which use graphs and which turn out to correspond to methods of clustering. In order to define these algorithms however, we need to review just a few more definitions from graph theory.

> [Comment: these properties of graphs can describe *arbitrary* graphs without any clustering context; their applicability is not restricted to threshold graphs].

The *connected components* of a graph G may be defined as the "islands" (or "pieces") G is composed of, where two vertices are part of the same island (piece) if one can walk from one to the other using edges. Thus imagining the threshold graphs above to be bird-eye views, we see

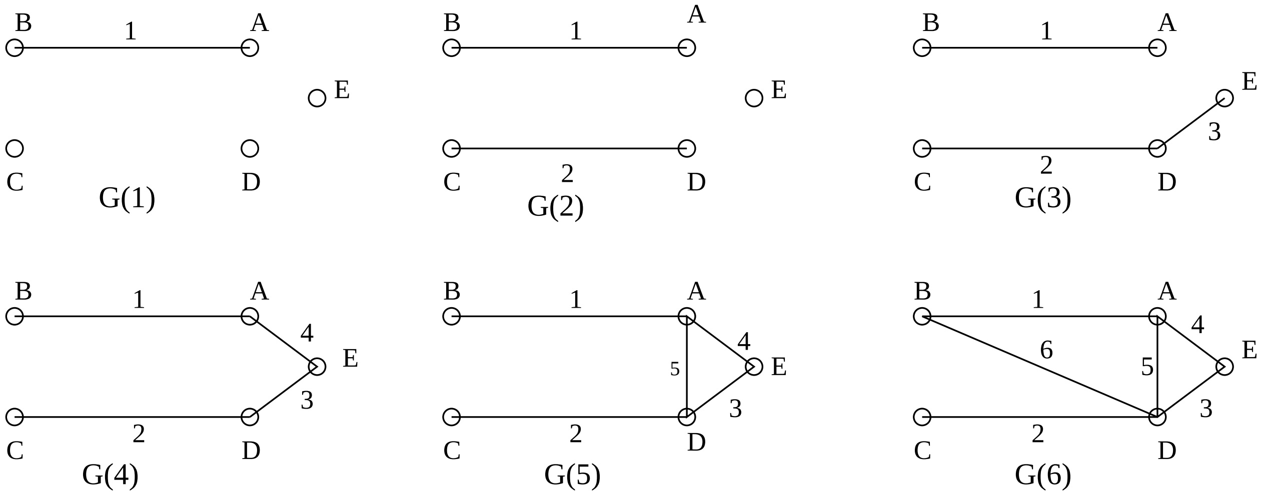

Figure 4:

that G(1) has four connected components: each of C, D, and E are components, while the segment between A and B forms a single, fourth component. G(2) has three components, G(3) has two components, while G(v) for any v greater than or equal to 4 has just one component.

A graph is *complete* if an edge exists between *every* pair of vertices. For example, each of the two graphs in Figure 5 represent a complete graph with four vertices:

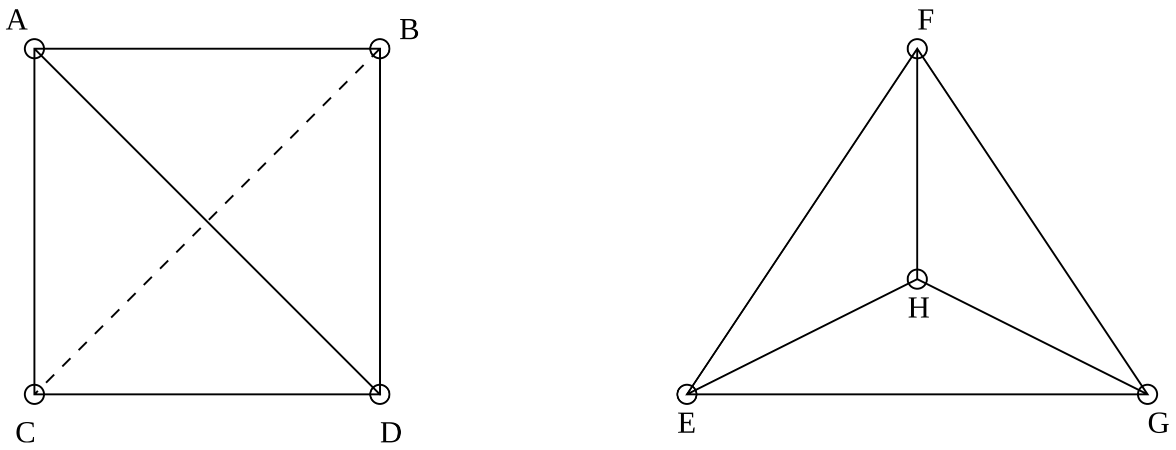

Figure 5:

Here as previously, one of a pair of crossing edges is depicted as a dashed line to aid in telling apart the edges, though this is of course not strictly necessary.

A complete graph with three vertices looks like a “triangle” while a complete graph with two vertices is just an edge between two vertices. A complete graph with five vertices is a “pentagram”, as depicted in Figure 6.

Finally, let H be a *subgraph* of G, that is, a subset of G – a sub-collection of vertices and edges – which itself is a graph, so that if e is any edge of G included in H, then the endpoints of e must be in H as well. Then a *clique* in a graph G is a subgraph H which is *complete*; that is, such that any two vertices in the subgraph are connected by an edge which is included in that subgraph,

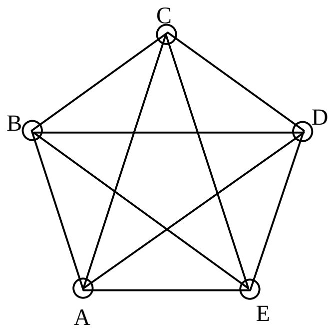

Figure 6:

i.e., by an edge which has been declared as belonging to H in the definition of H. For example, any two vertices joined by an edge define a two-element clique, while in the graph show in Figure 7 the vertices B, C and D along with the three edges connecting them constitute a clique, and similarly with vertices C, D and E as a set.

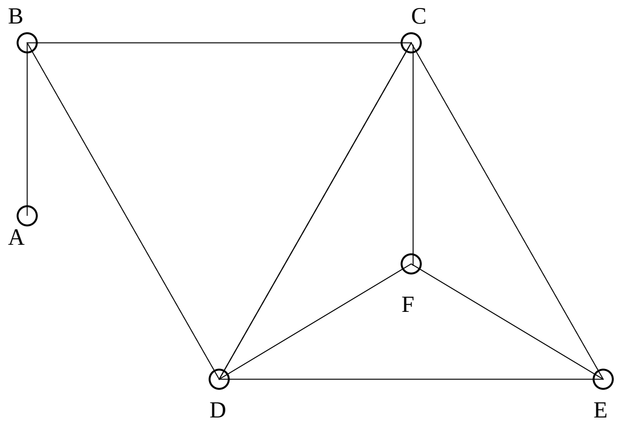

Figure 7:

A clique with four elements can also be found in Figure 7, and its vertices are C, D, E, and F.

[Comment: those interested in a good source of more definitions and examples may want to explore [8] and [9]].

1.6 Clustering Algorithms Revisited: Graphical Versions

With these definitions, it's now surprisingly easy to define the following two algorithms which use graphs, and which turn out – as you will verify – to give us precisely the single linkage and complete linkage clustering algorithms we have discovered and found useful under our original framework for looking at clustering problems.

Graph-theoretic Single Linkage algorithm (GSL)

Step 1: Begin with the disjoint clustering which places every object in its own separate cluster, i.e., for each $O_i \in O$, one forms the cluster $C_i = \{O_i\}$. Define the variable k and initialize its value to zero. [Notice that the clustering defined in step 1 can be arrived at by defining the connected components of G(k) – that is of G(0) since currently k =0 – to be the clusters].

Step 2: Increase the value of k by 1.

Step 3: Using the threshold graph G(k), redefine the current clustering by letting each component of G(k) represent a cluster. If there is only one cluster, then stop. Otherwise, go to step 2.

Graph-theoretic Complete Linkage algorithm (GCL)

Step 1: As before, let k =0 and let the initial clustering be the one implied by the threshold graph G(0).

Step 2: Increase the value of k by 1.

Step 3: Using the threshold graph G(k), redefine the current clustering to correspond to the cliques in G(k). That is, whenever two clusters from the previous stage clustering, form a clique in G(k), they are merged into a single cluster. If there is only one cluster, then stop. Otherwise, go to step 2.

To better understand the workings of Step 3, consider the threshold diagrams G(i) in the subsection, "At the Threshold of Something New" above. Starting at stage G(2) one has the clustering {[AB], [CD], [E]}, and this clustering remains the same through G(3) and G(4).

Even when the algorithm looks at G(5) in Step 3, this same clustering is maintained. This is because although the nodes for [A], [D], and [E] can be seen as a "clique," the algorithm *only* tests whether the set-theoretic unions (combinations) *of existing clusters* are cliques in the new G(k). Yet clearly while one cannot obtain [ADE] by combining existing clusters. Similarly, the algorithm maintains the same clustering {[AB], [CD], [E]} even after examining G(6). However, upon looking at G(7), Step 3 of this algorithm would merge [AB] and [E] to obtain, {[ABE], [CD]}.

[Comment: See [JD] for more details on these graph-theoretical algorithms]

Recall that we are excluding "ties" in the dissimilarity matrix. With the dissimilarities being distinct, nothing is changed by assuming further that the values are all integers. Thus, only a single edge is added each time k is increased by 1.

Exercise: Perform the algorithm GSL on the opening example of our five students and show that the resulting final clustering is the same as that given by the single linkage algorithm. In fact, show more: show that the entire series of nested clusterings is the same as for the single linkage hierarchical clustering algorithm. Do this by including and referring to each of the threshold graphs and indicating at each step (in writing or pictorially) what the connected components are, and what the resulting clusters are for each level of clustering (that is, at each threshold level).

Exercise: You have just shown that GSL gives the same results as the original single linkage method in one particular case, namely for the opening example of five students. Now give a rigorous argument indicating why GSL will give a nested series of clusterings which is *always* identical to that given by the single linkage algorithm. As always, your logical argument should be precise and detailed enough to convince a skeptical (but fair minded) fellow student. *Suggestion:* Think about what the connected components are, and what their role in GSL corresponds to in the original algorithm. Then construct and spell out a detailed argument.

Exercise: Perform GCL, the second of the two graphical algorithms above, on the opening example of our five students and show this time that the resulting final clustering is the same as that given by the complete linkage algorithm. Now prove more: that the entire series of nested clusterings is the same as for the complete linkage hierarchical clustering algorithm. Do this by including and referring to each of the threshold graphs and indicating at each step (in writing or pictorially) what the cliques are, and what the resulting clusters are for each level of clustering.

Exercise: You have just shown that GCL gives the same results as the original complete linkage method in one particular case: for the opening example of five students. Now give a rigorous argument indicating why GCL will give a nested series of clusterings which is *always* identical to that given by the complete linkage algorithm. *Suggestion:* Think about what the cliques are, and what their role in GCL corresponds to in the original algorithm. Then write out your detailed argument which would convince a skeptical but fair-minded fellow student.

Question: Can you suggest a reason why the complete linkage method is appropriately named? (*Suggestion:* review the definition of a clique).

Remark: If one changes the specific numbers in the dissimilarity matrix, one says that the *rank order* is unchanged if the relative sizes of pairs of entries remain unchanged (e.g., if the original number in the third row, second column, is bigger than the original number in the fourth row, first column, this remains true for the new numbers). You may have noticed that only the "rank order" the entries (numbers) in the dissimilarity matrix matters. Keeping the rank order, while modifying the specific numbers will produce the same output by SL and by CL, in other words. You may wish to prove that the output of SL (and likewise of CL) is invariant under an "order preserving bijection" changing the entries of the dissimilarity matrix. Consult with your instructor if you need help in defining an "order preserving bijection".

2 Divisive Clustering

Suppose that a group of 15 boys is to be separated into three groups to work on a certain project. The person in charge of doing this has decided to simply group the boys based on where they live so that their parents can carpool. Following the basic methods examined before we would have to go through many steps in order to get down to only three groups.

Exercise 1 Using an agglomerative method each step combines two previous clusters so that there is now one fewer cluster than before. If you start with 15 boys and want to have three clusters, how many steps will it take using an agglomerative method?

In light of the amount of work needed to use these old methods for this problem it might be easier to use a method called divisive clustering. In agglomerative methods we start with each object in a separate cluster and combine clusters at each step. Divisive clustering will do the opposite – start with all the objects in one cluster and divide a cluster into two smaller clusters at each step. Thus, in this particular example we would start with all 15 boys in one cluster, split this cluster into two sub-clusters, and then split one of those sub-clusters again to form three total sub-clusters. This requires only a two step process.

Let us actually demonstrate using a smaller example, where our goal is to form two clusters.

Example 1 Suppose that five boys have measured how far it is between their houses and the results are listed below:

Student:					
Bob					
Jim	1.3				
John	3.2	2.5			
Matt	2.9	2.7	0.7		
Steve	3.5	4	2.2	1.6	
distance to:	Bob	Jim	John	Matt	Steve

If we would like the maximum distance between any two boys in the same group to be as small as possible, and we don't wish to use an agglomerative method, then we see that we would like Jim and Steve to be in different groups because they live the farthest away from one another. We will follow this type of reasoning to build up the two groups. Once we have finished, we will have divided the entire group into two smaller groups. So, since we wanted Jim and Steve in different groups, we start with

Group 1 {Jim}   Group 2 {Steve}

The next largest distance is between Bob and Steve, so we would also like them to be in different groups. This means that Bob and Jim should be in the same group.

{Jim, Bob}   {Steve}

Continuing on in this way we see that John and Bob are the next farthest apart. So, we should put John in the group with Steve.

{Jim, Bob}   {Steve, John}

Next, Bob and Matt should be in different groups, so we also put Matt in the group with Steve and John.

{Jim, Bob}   {Steve, John, Matt}

Everyone is now in a group so we are done. We have formed two groups with Matt, Steve, and John in one group and Bob and Jim in the other. The largest distance between any two boys in the same group is 2.2 miles between John and Steve.

We can see from this example a method that we might follow to form groups of this kind. However, there are some potential difficulties that did not arise in this example. Let us examine a similar situation. This time we will simply refer to the boys by the letters A to E.

Example 2

Starting as before, C and D are the farthest apart so we should put them in different groups.

$\{C\}$ $\{D\}$

The next farthest apart are D and E. If we also put them in different groups this means that C and E must be in the same group.

A					
B	8				
C	7	1			
D	5	3	10		
E	2	4	6	9	
	A	B	C	D	E

$\{C,\ E\}$ $\qquad\qquad\qquad\qquad$ $\{D\}$

Next, A and B should be in different groups. However, it is not clear how to decide which group to put A in. So, we will wait, and simply keep in mind that we plan to put these two in separate groups.

$\{C,\ E,\ A$ or $B\}$ $\qquad\qquad\qquad\qquad$ $\{D,\ B$ or $A\}$

Here we write A or B in one group and B or A in the other to indicate that we will either choose the first option both times (A in the first group and B in the second) or the second option both times (B in the first group and A in the second).

Next A and C should be separated so this tells us that we should choose the second option listed above. Thus, the two groups are $\{C,\ E,\ B\}$ and $\{D,\ A\}$. So, we see that sometimes we must wait to determine exactly which group to put certain elements in. In a larger example, this type of problem may happen several times and be very difficult to keep track of.

Another potential difficulty arises in the following example, which is quite similar to the last example.

Example 3

A					
B	5				
C	6	1			
D	8	3	10		
E	2	4	7	9	
	A	B	C	D	E

Again we start by placing C and D in separate groups.

$\{C\}$ $\qquad\qquad\qquad\qquad$ $\{D\}$

Also, we need D and E in different groups so that once again C and E are together.

$\{C,\ E\}$ $\qquad\qquad\qquad\qquad$ $\{D\}$

This time the next largest distance is between A and D. So, we put A in the group with C and E.

$\{C,\ E,\ A\}$ $\qquad\qquad\qquad\qquad$ $\{D\}$

Now comes the interesting step. According to the pattern we have followed we should next put C and E in separate groups. This is not possible unless we change the work done so far. In a situation like this, we leave the groups as previously determined. (In fact, we can conclude at this point that the largest difference between members of the same group will be 7, the distance between C and E.) We then continue on as before. We would like A and C to be in different groups. This is again not possible, so we skip to the next pair to be considered. We would like A and B to be in different groups. This works if B is put in the group with D.

$\{C, E, A\}$ $\{D, B\}$

Note that we quit as soon as all five elements are assigned to groups. So, our final groups are A, C, E and B, D.

Sometimes we will want to have more than two groups. If this is the case we can repeat the procedure described above a second time on one of the newly formed groups.

Example 4 Let us consider our original example of 15 boys. Suppose the distances between their houses are given below.

A															
B	.6														
C	.6	4.0													
D	5.5	.1	.5												
E	.8	9.2	8.0	.0											
F	.7	7.4	.2	.3	6.4										
G	5.3	.1	.0	6.2	.5	.4									
H	.5	5.4	5.1	.7	8.3	6.5	.2								
I	.0	8.1	7.5	.8	4.1	5.6	.4	8.4							
J	6.0	.8	.0	4.9	.0	.3	4.2	.3	.9						
K	.7	6.9	7.3	.8	.4	9.5	.2	5.7	.7	.7					
L	6.1	.5	.9	4.3	.8	.2	3.0	.9	.1	5.0	.6				
M	.9	6.3	4.4	.3	7.2	6.6	.9	7.0	9.0	.0	8.6	.4			
N	.2	.0	8.7	.5	4.5	5.8	.7	7.6	4.7	.0	.9	.6	7.1		
O	.6	7.7	6.8	.6	4.8	4.6	.9	6.7	5.2	.7	.8	.5	8.9	5.9	
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

After going through the process described above once, we have the groups $\{A, D, G, J, L\}$ and $\{B, C, E, F, H, I, K, M, N, O\}$. Since we wish to have three groups we will now divide one of the groups again. As we are trying to minimize the distances within groups we look at the largest distance between two boys in the same group. Boys E and K live 14.4 miles apart and are in the same group. So, we will split the second group into two by following the same procedure as before. The result of this process is the groups $\{B, C, H, K, M\}$ and $\{E, F, I, N, O\}$. Putting these two groups with our other group from the first step we see that we have successfully separated the boys into three groups: $\{A, D, G, J, L\}$, $\{B, C, H, K, M\}$ and $\{E, F, I, N, O\}$. The largest distance between any two boys in the same group is now 8.6 miles. (Note: We were very lucky that the groups ended up equal in size. There is nothing about this process that will guarantee this sort of result.)

3 Divisive Clustering - Theory (Optional)

In order to give a specific algorithm for the method used in the previous section we need several facts from graph theory. Hence, we will begin this section with a brief introduction to the necessary concepts from graph theory. Recall from the introductory section on agglomerative methods the definition of a graph, which we formalize here via set theory.

Definition A *graph* is a pair where V is a set and E is a set of pairs of elements of V. The elements of V are called *vertices* and the elements of E (which are pairs of vertices) are called *edges*. If a pair of vertices forms an edge, those two vertices are said to be *adjacent.*

Example 1 If $V = \{A, B, C, D, E\}$ and $E = \{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}, \{D, E\}\}$ then $G = (V, E)$ is a graph that can be represented by dots and line segments, as in Figure 8:

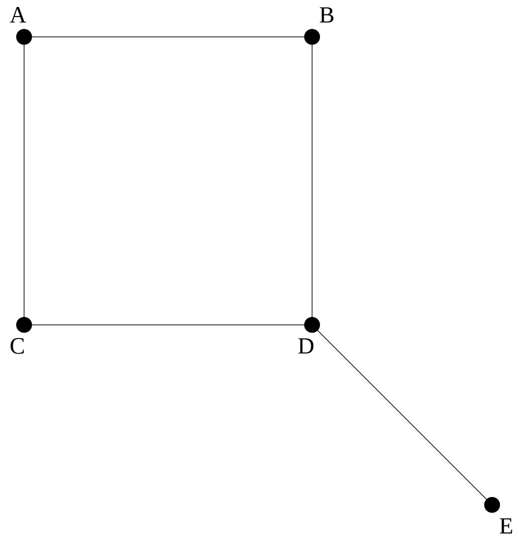

Figure 8:

Definition A graph is said to be *connected* if for any two vertices, v and w, there is a sequence of vertices $v = v_1, v_2, \ldots, v_n = w$ such that each is an edge. In other words, a graph is connected if there is a path through the graph which passes through all its vertices.

Example 2 The graph in example 1 is connected (e.g. E, D, B, A, C is an appropriate path) but the graph in Figure 9 is not.

Definition A *cycle* is a sequence of vertices $\{v_1, v_2, \ldots, v_n\}$ such that each $\{v_i, v_{i+1}\}$ is an edge and also $\{v_1, v_n\}$ is an edge.

Example 3 In the graph in example 1 the vertices A, B, D, C form a cycle.

Definition A *tree* is a connected graph with no cycles.

Example 4 The graph in example 1 is not a tree because it contains a cycle. The graph in example 2 is not a tree because it is not connected. The graph in Figure 10 is a tree.

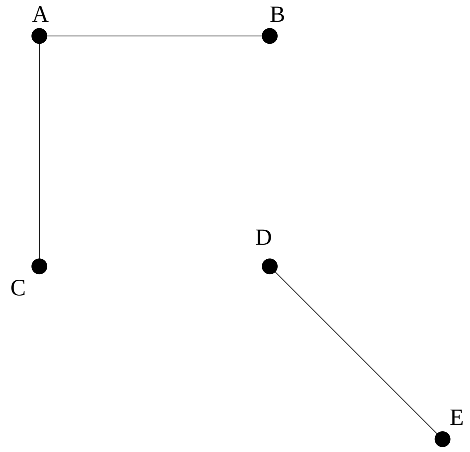

Figure 9:

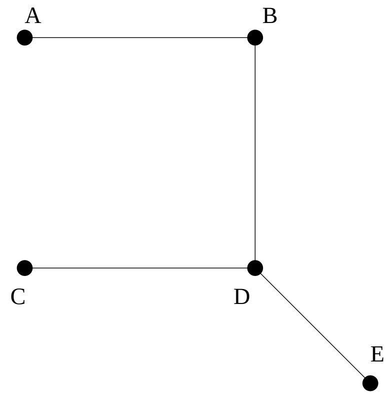

Figure 10:

Definition A *weighted graph* is a graph together with a weight function which associates to each edge a real-valued weight.

Example 5 The graph in Figure 11 is the graph from example 1 with a weight function shown to make it a weighted graph.

Definition Let $G = (V, E)$ be a weighted graph. A *maximal weight spanning tree* of G is a tree on the same set of vertices as G with $E' \subseteq E$ such that the sum of the weights of the edges in E' is as large as possible.

Example 6 In Figure 12 we show the maximal weight spanning tree of the weighted graph in example 5.

Maximal weight spanning trees can be found by a fairly simple algorithm. Let G be the initial graph and H be the graph that is being constructed.

1. Begin H with all vertices of G and no edges.

2. Let$\{u, v\}$ be the edge of G with the largest weight that has not been considered.

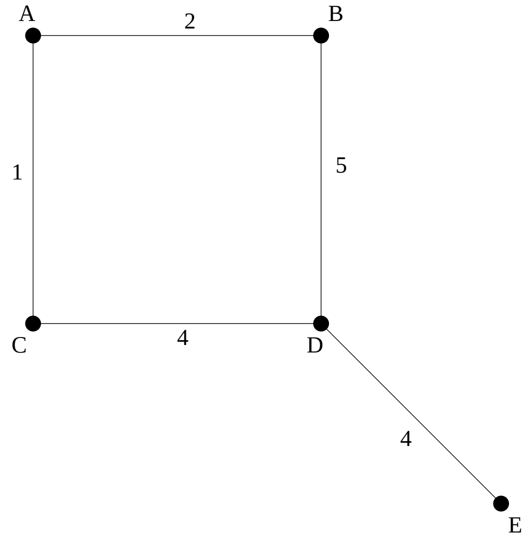

Figure 11:

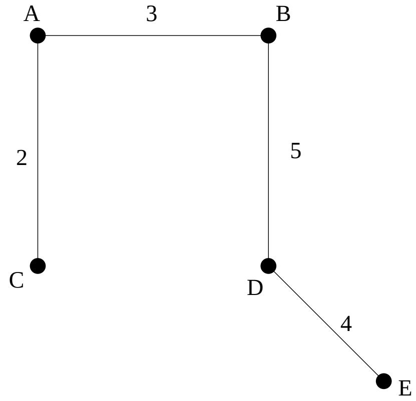

Figure 12:

3. If adding the edge $\{u, v\}$ to H will not complete a cycle, add that edge to H.

4. Repeat steps 2 and 3 until H is connected.

We give now a brief informal proof that this algorithm does what it claims to: We can see that our tree is maximal weight by noting that at each step we add the edge with the largest weight that will not complete a cycle. If a different edge were added, we would have to delete an edge that has a larger weight. This would only reduce the total of the weights. Further, we note that if H is not yet connected it must be possible to add and edge without completing a cycle. This is true because if the H is not connected there must be two vertices, say u and v, that are not connected. Adding edge $\{u, v\}$ would therefore not complete a cycle.

Example 7 Consider the weighted graph G in Figure 13.

The maximal weight spanning tree will be built up as in Figure 14.

Next the side weighted 5 must be skipped because it would cause a cycle through B, C, and E. So the final maximal spanning tree is as shown in Figure 15.

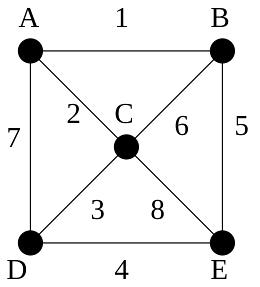

Figure 13:

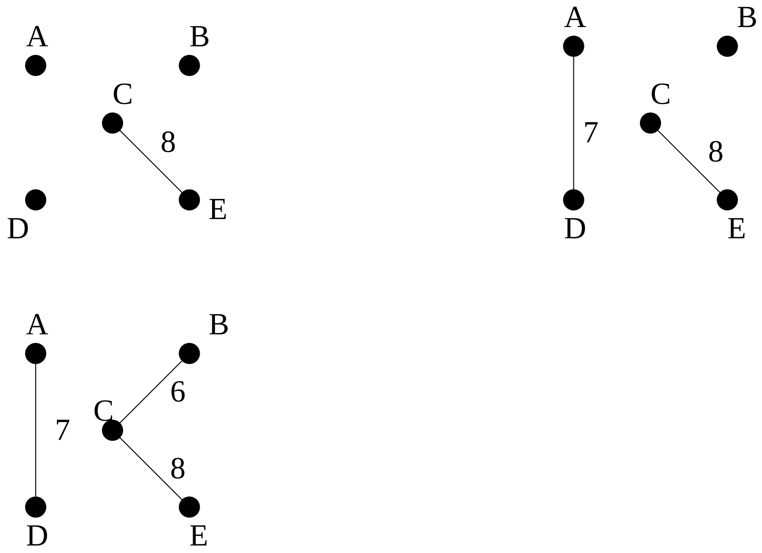

Figure 14:

To wrap up our excursion into graph theory, we also need the concept of a graph coloring. To color a graph we assign a color to each vertex in such a way that if two vertices are adjacent, then they are assigned different colors. It is easy to see that a tree will require only two colors. To color a tree we simply start at any vertex and assign it a color. All vertices adjacent to this vertex are assigned the second color. All vertices adjacent to these are given the first color again. We continue this pattern until all vertices are colored. We can never "come back" to a vertex already colored and require it to be a different color because there are no cycles in a tree.

Example 8 The tree from example 4 can be colored as shown in Figure 16.

Now, recall that we began all of this graph theory as an aid to understanding the algorithm for divisive clustering illustrated in the previous section. Let us now give that algorithm.

Algorithm for divisive clustering:

1. Given a set of elements with a dissimilarity (or "distance") function, form a complete, weighted graph as described below:

   (a) Consider each of the original elements as a vertex

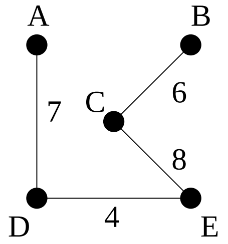

Figure 15:

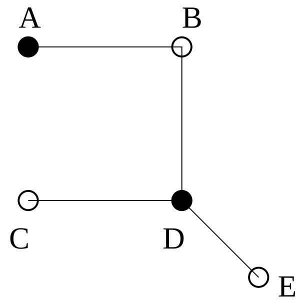

Figure 16:

(b) Include all possible edges (so the graph is complete)

(c) Define the weight of an edge to be the dissimilarity of the vertices

2. Find a maximal spanning tree of the weighted graph.

3. Color the maximal spanning tree.

4. Divide the original set so that all vertices of the same color are in the same part.

Note that with this algorithm we do not have partial, or tentative, groups along the way as we did when following our intuitive idea of divisive clustering in the previous section. At step 4 above the entire group is divided into two groups all in one step.

Example 9 Using the same data as from example 2 in the previous section we have

A					
B	8				
C	7	1			
D	5	3	10		
E	2	4	6	9	
	A	B	C	D	E

Applying the algorithm, we build the maximal spanning tree as show in figure 17. (Note that, in practice, we need not actually draw the original weighted graph, or even use all of the vertices at the start.)

Then coloring the tree gives the final tree as in Figure 18

Hence, the two groups are $\{A,\ D\}$ and $\{B,\ C,\ E\}$ as we saw before.

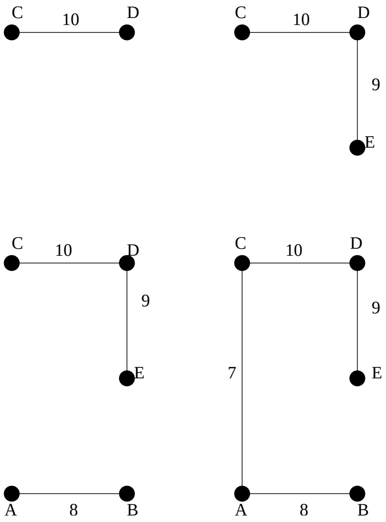

Figure 17:

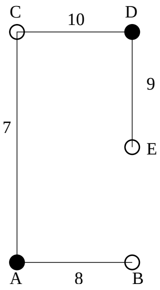

Figure 18:

4 Sequential Clustering

The clustering procedures described above assign all the given entities to clusters. This completeness is necessary in many applications – see, for example, [JD]. However, there are a lot of problems where we do not have to build a complete partition of a given set O. Thus, in problems of image recognition we are to single out only *some* images-clusters off the background. For instance, let a space shuttle rotate around Mars and look for an invisible ("sleeping") landing device. A radar is scanning the surface and a computer program analyses this picture, pixel by pixel, and tries to combine raw elements in meaningful *clusters*, using the algorithm prescribed. After discovering a possibly human-made structure of straight lines and circles, the computer can quit its search over the surrounding area, at least temporarily.

In this section, following P. Hansen, B. Jaumard, and N. Mladenovic [HJM], we consider a corresponding paradigm of cluster analysis, called *sequential clustering.* In sequential clustering we do not initially assign all the objects to single-element clusters as in agglomerative clustering, and we do not initially split O in two clusters as in divisive clustering. Rather, we look for a most naturally separated (in a precisely defined sense) subset of the initially given set O, and designate it as the first cluster C_1. Then we remove C_1 from O and repeat this procedure starting with the *set-difference*[1] $O \backslash C_1$. The algorithm stops when the last set-difference exhibits no relevant internal structure (or it is empty). This approach is useful in many instances, for example, in image recognition [MAM]. Moreover, unlike the hierarchical clustering, this approach does not *a priori* impose a definite structure, such as a hierarchy of partitions, on the clustering being constructed.

First, we derive an algorithm of sequential clustering in a naive way, applying it to the very first example from another module in this series [K].

Example 1. Consider a set of eight entities O $= \{O_1,\ O_2,\ O_3\ O_4,\ O_5,\ O_6,\ O_7,\ O_8\}$ and their dissimilarity table:

	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
O_1	0							
O_2	5	0						
O_3	10	8	0					
O_4	7	12	1	0				
O_5	22	28	9	4	0			
O_6	27	23	19	14	11	0		
O_7	25	17	3	2	16	15	0	
O_8	13	6	26	21	18	20	24	0

Table 3: The dissimilarity table for the model example

The graph in Figure 19 gives a partial representation of this example. The vertices here correspond to the entities and the weights of the edges (not all of them are shown) are the dissimilarities between the entities. Certainly, there are many ways to draw these graphs manually. However, algorithm, presented later on, eliminates this arbitrariness. Here, we have tried to draw the edges shorter if they carry smaller dissimilarities. This allows us to inspect the initial configuration visually:

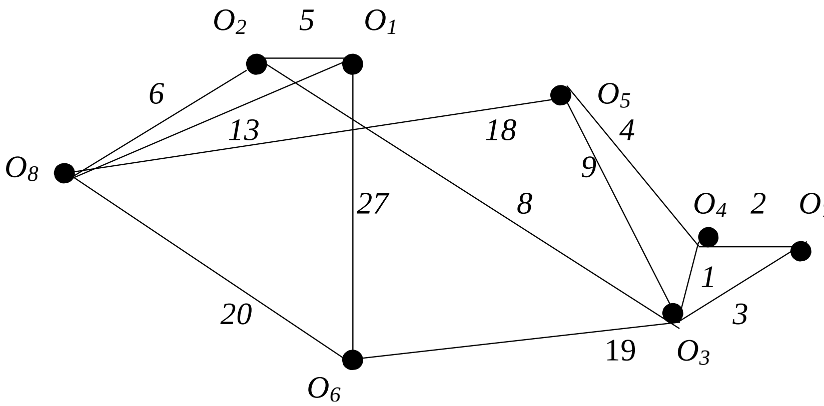

Figure 19: Initial configuration. Some of larger dissimilarities (> 6) are not shown.

We immediately observe three nearby vertices $O_3,$ O_4, and O_7 and separate them in the cluster $C_1 = \{\ O_3,\ O_4,\ O_7\}$. We do not include the vertex O_5 in this cluster, since its distances

[1] Here "\" denotes the set-difference, that is, A\B is the set of all objects which are in A but do not belong to B.

from O_3 and O_7 are considerably larger than those within C_1. We will return to this issue later. Removing C_1 from this graph, we get a configuration in Figure 20, where a few more edges are shown.

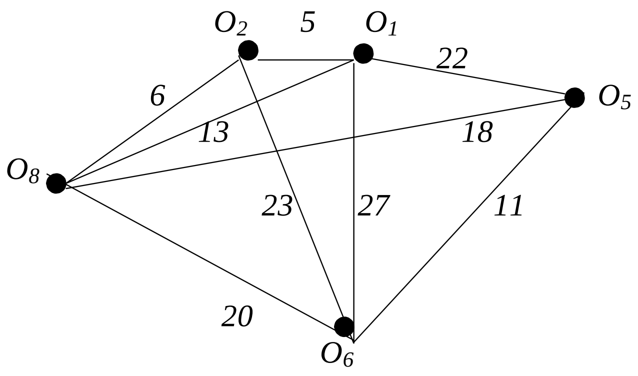

Figure 20: Configuration after removing the cluster C_1

Observing the proximity of O_1, O_2, and O_8 to each other, we may form another cluster C_2 = $\{O_1, O_2, O_8\}$. Removing it, we arrive at Figure 21.

Finally, we can either combine the two remaining entities O_5 and O_6 in a cluster C_3, or we can leave them unassigned. It is interesting to compare our "naive" approach with other clustering algorithms. For example, when the Single Link Algorithm is applied to the same example (see [K]), the cluster C_1 emerges at the second level of clustering, the cluster C_2 appears at the fifth level, and the entities O_5 and O_6 meet one another only at the last, seventh step, in the *conjoint* clustering, when all entities are in one cluster.

Obviously, in this example we have made some "arbitrary" decisions. For example, at the very first step we could have immediately removed a cluster $\hat{C}_1 = \{ O_3, O_4, O_5, O_7\}$ instead of C_1. To avoid this arbitrariness, our decisions must be based on clearly stated criteria regarding the proximity of entities. Such criteria may change from step to step, that is, the criterion we use to choose the first cluster C_1, may differ from the criterion we use to find C_2, etc.; these criteria are called *local algorithms* and are denoted by A_{k}, $k = 1, 2, \ldots$. The following examples will show why we may prefer using different local algorithms at consecutive steps in the same clustering problem.

Even if we separate just a one-element cluster at each step, the maximum number, s, of steps can not exceed the number of objects in O less one, in symbols, $s \leq |O| - 1$. Hereafter, the notation $|X|$ stands for the number of elements in a set X. This number is also called the *cardinal number* or cardinality of X. For example, the cardinal number of the set of all characters in the English alphabet is 26, that is, $|\{$a, b, c,..., x, y, z$\}| = 26$. Hereafter, we denote by N the *cardinal number* of a sample O, $N = |O|$, and by K the number of entities in a cluster C, $K = |C|$.

We also use the following standard notation. The writing a:= b means that a variable a is *assigned* a value b. For example, if $b = 3$, then after executing the command $a := b$ the new value of a is 3, and this value does not depend upon the preceding value of a. Furthermore, we

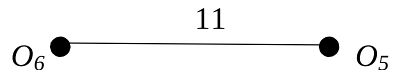

Figure 21: Configuration after removing the clusters C_1 and C_2

recall that the notation $\max\{d_l|l \in L\} \equiv \max_{i\in l}\ d_l$, where a dummy index l runs over the index set L, and similarly $\min\{d_l|l \in L\}$ or $\min_{i\in l}\ d_l$, were introduced in the subsection *Single and Complete Linkage: the Inside Scoop.* Thus, $\max_l d_l$ means that among several given quantities d_l with different subscripts l, we look for (one of) the largest d_l and denote it by $\max\{d_l|l \in L\}$ or just $\max_l\ d_l$. There may be *several equal largest* d_l. If l stands for the indexes of elements $O_l \in C$ of a set C, we denote this maximum by $\max\{d_l|O_l \in C\}$ and read this expression as "*The maximum of all of the quantities* d_l *over all subscripts* l *such that* $O_l \in C$."

Formalizing our considerations, we arrive at the following formal procedure called a *sequential clustering algorithm.* It is written in a so-called *pseudocode* format. In the following algorithm, we gradually remove some entities from the original set O. At every step, S contains unassigned entities, and $C_{\text{k}}$is the cluster we remove from the set of remaining objects. The set R contains the unassigned entities when the algorithm halts.

Sequential Clustering Algorithm. Given a finite sample O, its dissimilarity table, and a set of local algorithms A_{k}, k = 1...$|O|$ - 1.

1. Initialization. Set $k := 0$, $S := O$, and $R := O$.

2. Loop. Set $k := k + 1$.

   Apply a local algorithm $A_{\text{k}}$to the set Sto find C_{k}.

      If A_{k} fails, update$R$:=S *and* End.

      Else: record C_{k} and update S:= $S \setminus C_{\text{k}}$.

      End If

   End Loop.

3. End.

Remark. We may need fewer than $|O| - 1$ local algorithms.

To apply this algorithm, we must specify local algorithms A_{k}. To this end, we have to take into account both the specifics of the problem under consideration and the general philosophy of cluster analysis, that is, getting reasonably homogeneous and wellseparated clusters. Here, a cluster is said to be homogeneous if the dissimilarities between its elements are close to each other — of course, the latter is not a precise definition of the homogeneity. In the same style, two clusters are said to be well separated, if the dissimilarity for each pair of elements within either of these clusters is less than any dissimilarity between an element in one cluster and an element in another cluster. Appropriate local algorithms are often based on the concepts such as the *diameter*, *radius*, and *split* of a set $C \subset O$. The diameter and the radius estimate homogeneity of clusters.

Definition 1. The *diameter*, $d(C)$, of a set $C \subset O$ is the largest dissimilarity between a pair of entities of C, that is,

$$d(c) = \max\{\{\max\ d_{kl}|O_l \in C\}|O_k \in C\}$$

or, which is the same,

$$d(C) = \max\{d_{kl}|O_k \in C, O_l \in C\}.$$

Before considering examples, let us note that the latter expression may look simpler, but the former is simpler in calculations. The first formula defines the diameter as the reiterated maximum of all the dissimilarities between pairs of objects in C. That means that initially we fix ("freeze") the first subscript k and calculate $\max\{d_{kl}|O_l \in C\}$ with this fixed k; here O_l runs over all possible subsets of C. For example, if k=1, we calculate $\max\{d_{1l}|O_l \in C\}$. Then we fix next k and calculate $\max\{d_{kl}|O_l \in C\}$ with this new fixed k. Thus, if k=2, we look for $\max\{d_{2l}|O_l \in C\}$. This way, we find as many numbers $\max\{d_{kl}|O_l \in C\}$ as there are various values of the parameter k and then we select the largest among these numbers $\max\{d_{kl}|O_l \in C\}$. Precisely this largest quantity is denoted by $d(C) = \max\{\{\max d_{kl}|O_l \in C\}|O_k \in C\}$.

Example 2. The graph in Figure 22 represents the very first example in our module — the five students A, B, C, D, and E, denoted here by o_1, o_2, o_3, o_4, o_5, and the distances between their residences:

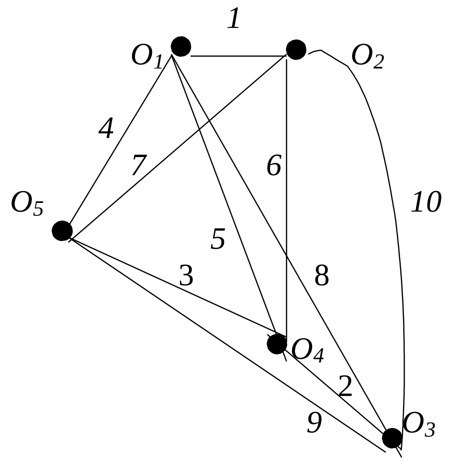

Figure 22: Graph corresponding to the initial example.

Again, we can visually observe in the set $O = \{O_1,\ O_2,\ \ O_3,\ O_4,\ \ O_5\}$ two clusters: C_1= $\{O_1,\ \ O_2\}$ and C_2= $\{O_3,\ O_4,\ \ O_5\}$. It should be noted that this clustering results also from the single linkage clustering algorithm applied to O — see *The Missing Links* Section. But now we want to proceed in a more systematic way using the sequential clustering paradigm. To this end, next we calculate the diameters of these clusters. The set C_1 contains only two elements, therefore, for both $k = 1$ and $k = 2$, $\max\{d_{1l}|O_l \in C\} = \max\{d_{2l}|O_l \in C\} = 1$ and $d(C_1) = \max\{\{\max d_{kl}|O_l \in C_1\}|O_k \in C1\} = \max\{l, l\} = 1$. Further, for C_2 we calculate $\max\{d_{3l}|O_l \in C_2\} = \max\{d_{3,4}, d_{3,5}\} = max\{2, 9\} = 9$, $\max\{d_{4l}|O_l \in C_2\} = \max\{2, 3\} = 3$, $\max\{d_{5l}|O_l \in C_2\} = \max\{3, 9\} = 9$, and finally $d(C_2) = \max\{3, 9, 9\} = 9$. In Section 1, we arrived at the same clustering and calculated the same quantities in a quite different way, by making use of the Single Link Algorithm. To avoid misunderstanding, it should be said that, unlike the definition of dissimilarity between two clusters involving the repeated minimum, the definition of the diameter of a subset involves only *one* set and uses the repeated maximum.

Pause for thought. Try to describe the Single Link Algorithm in terms of the diameters of clusters.

It should be said again that Definition 1 can be written using one maximum: $d(C) = \max\{d_{kl}|O_k \in C, O_l \in C\}$, which makes apparent the choice of the term *diameter*. However, we often prefer the first formula, since we cannot avoid repeating symbols of maximum and minimum in the two following definitions. The radius is the smallest maximum among all the dissimilarities between pairs of objects in C.

Definition 2. The *radius* of a set $C \subset O$ is

$$r(C) = \min\{\max\{d_{kl}|O_l \in C\}|O_k \in C\}.$$

Exercise 1. Prove or disprove that $\min\{\max d_{kl}\} = \max\{\min d_{kl}\}$ for any rectangular table $[d_{kl}]$.

Let us reconsider Example 2 (Figure 22). For the same C_1 we again have $\max\{d_{1l}|O_l \in C_1\} = \max\{d_{2l}|O_l \in C_2\} = 1$. Thus, $r(C_1) = \min\{\max\{d_{kl}|O_l \in C1\}|O_k \in C_l\} = \min\{1, 1\} = 1$. It is worth noting that in our usual plane geometry the diameter of a circle is twice the radius. However, it is not always the case in our current "discrete world" under exploration. It might even happen that the radius and the diameter of a set coincide. Thus, the diameter of C_2 (see above) is three times its radius: $r(C_2) = \min\{3, 9, 9\} = 3$. Comparing this calculation with Figure 4, we get a clearer understanding why this quantity is called the radius of a set. Obviously, the vertex O_4 can be viewed as the center of the set C_2 and all the other vertices in C_2 are apart from O_4 for at most 3 units.

The last definition of this section concerns the separation of a subset from other subsets — the quantity to be defined measures how far a set C is 'split off' from the rest of the sample.

Definition 3. The *split*,$s(C)$, of a set $C \subset O$ is the smallest dissimilarity between a pair of entities, one in C and another in O$\backslash C$, that is,

$$s(C) = \min\{\min\{d_{kl}|O_l \notin C\}|O_k \in C\}.$$

It is worth repeating: the outer minimum is taken over all subsets belonging *to C*, while the inner minimum is taken over all subsets that *do not belong* to C.

We continue studying the same example (Figure 22). For C_1, $\min\{d1l|O_l \notin C\} = \min\{4, 5, 8\} = 4$ and $\min\{d_{2l}|O_l \notin C_l\} = \min\{10, 6, 7\} = 6$. Therefore, $s(C_l) = \min\{\min\{d_{kl}|O_l \notin C_1\}|O_k \in C_1\} = min\{4, 6\} = 4$. For C_2, we have $\min\{d_{3l}|O_l \notin C_2\} = \min\{8, 10\} = 8$, $\min\{d_{4l}|O_l \notin C_2\} = \min\{5, 8\} = 5$, $\min\{d_{5l}|O_l \notin C_2\} = \min\{4, 7\} = 4$, and finally $s(C_2) = \min\{8, 5, 4\} = 4$. In this example, since we have exactly two clusters, $s(C_1) = s(C_2)$.

Exercise 2. Suppose, in a problem we have exactly two clusters, C_1 and C_2. Is it always the case that their splits are equal, $s(C_1) = s(C_2)$?

These three concepts — the diameter, radius, and split give rise to the following commonly used local algorithms. We define three quantities, the *minimum diameter*, the *minimum radius*, and the *maximum split*. In these definitions maximum or minimum is taken over all subsets $C \subset O$ containing exactly K entities, where a number K is to be the size of a cluster we desire to have:

A)	Minimum diameter:	$d_{\text{min}}(O, K) = \min\{d(C)|C \subset O, |C| = K\}$
B)	Minimum radius:	$r_{\text{min}}(O, K) = \min\{r(C)|C \subset O, |C| = K\}$
C)	Maximum split:	$s_{\text{max}}(O, K) = \max\{s(C)|C \subset O, |C| = K\}$.

We repeat that in each of these three definitions the extremum is taken over all K–element subsets $C \subset O$, $|C| = K$, and only the specifics of a problem dictate our choice of K.

To illustrate these local criteria, we apply them to our Example 1. We start with the case A), i.e., we want to minimize the diameters of removed clusters.

A) If we want to separate firstly only *two* entities in a cluster, we set $K = |C| = 2$. Obviously, the smallest diameter among all 2–element subsets of O is 1, that is, $d_{\text{min}}(O, 2) = 1$, and this value is attained if $C = \{O_3,\ O_4\}$. However, if at the very first step we want to remove *three* entities, that is, if we consider initially $K = |C| = 3$, then $d_{\text{min}}(O, 3) = 3$ and this value is attained if C = $C_1 = \{O_3,\ O_4,\ \ O_7\}$. Suppose, to proceed faster, we wish to start with K = 3.

After removing this $C_1 = \{\ O_3,\ O_4,\ \ O_7\}$ from O, if we apply the same algorithm with $K = 3$ to the set $O\backslash C_1$, we obtain a cluster $C_2 = \{O_1,\ \ O_2,\ O_8\}$, that is, we repeat the results of our intuitive approach (see above). Suppose, however, that at the second step we realize that the next three objects are not close enough for our purpose and we decide to change a local criterion. Let us choose now $K = 2$. Then, since the object O_4 was removed in a previous step,the recalculated (with this new K) minimal diameter is $d(C_2) = 5$, where C'$_2$ = $\{O_1,\ O_2\}$. The configuration after removing *this* = $\{O_1,\ O_2\}$ from $O_{\text{new}} = O\backslash C_1$ is shown in Figure 23.

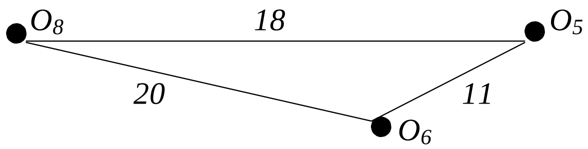

Figure 23: Configuration after removing the cluster C'$_2$.

Now we can either repeat a similar step with $K = 2$, or we can again change a local criterion to $K = 3$, or we can leave the three remaining entities O_5, O_6, O_8 unclustered. The decision, what local algorithm to choose during the run of the sequential clustering algorithm, can be made only after a consultation with an expert in the field where the problem arose. In the end of this section we consider again Example 2, using sequential clustering, and we shall see why it may be useful to change the value of Kas part of solving the same problem.

Exercise 3. Describe all clusters in this example, when we use all possible combinations of local criteria based on the minimal diameter with $K = 2$, $K = 3$, and $K = 4$. That is, give an answer consisting of the entire collection of clusters corresponding to each possible series of choices of K.

B) Let us now work the same example, using the minimum radius as a local criterion. If $K = 2$, then clearly $r_{min}(O, 2) = 1$ and this value is attained at $C = \{O_3,\ O_4\}$. On the other hand, if at the very first step we want to remove three entities, that is, we set $K = |C| = 3$, then, attained at $C_1 = \{O_3,\ O_4,\ \ O_7\}$, since

$$\min\{\max\{d_{3,l}|O_l \in C_1\}, \max\{d_{4,l}|O_l \in C_1\}, \max\{d_{7,l}|O_l \in C_1\}\} = \min\{3; 2; 3\} = 2$$

and no other three-element subset has the radius 2 or less. If we continue to use the criterion with $K = |C| = 3$, then the next three-element cluster to be removed is $C_2 = \{O_1,\ \ O_2,\ O_8\}$ with the minimal radius $r_min(O \setminus C_1, 3) = r(C_2) = 6$. Therefore, we arrive at the same clustering as in A).

Exercise 4. Describe all clusterings in this example, that is, give an answer consisting of the entire collection of clusters, when we use all possible combinations of local criteria based on the minimal radius with $K = 2$, $K = 3$, and $K = 4$.

C) We now apply the maximum split as a local criterion. Unlike the cases A) and B), where we looked for densely tied clusters, here we look for relatively disengaged clusters. If $K = 2$, there are two twoelement subsets with the same maximal split, $s_{max}(O, 2) = 6$. One of these two clusters is $\{O_1, O_2\}$.

Exercise 5. Verify that $s(\{O_1, O_2\}) = 6$ and find another cluster with the same maximal split $s = 6$.

Removal of either such cluster leads to different clusterings and only by knowing the specifics of the problem, can we decide which cluster is to be removed first. For instance, if we remove $C_1 = \{O_1,\ O_2\}$, we get the configuration in Figure 24.

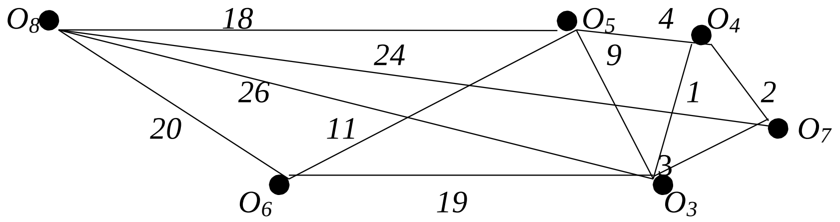

Figure 24: Configuration after removing C_1 =$\{O_1,\ O_2\}$. Not all edges are shown

The next maximum split with $K = 2$ is $s(C_2) = 11$, where again $C_2 = \{O_6,\ \ O_8\}$. Removing C_2, we arrive at the configuration in Figure 25.

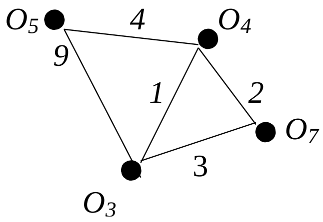

Figure 25: Configuration after removing C_2 = $\{O_6,\ O_8\}$

Here, the maximum split with $K = 2$ is $s_{max}(O‘ \setminus C_2) = s(C_3) = 3$, where $O‘ = O \setminus C_1$ and $C_3 = \{O_3,\ \ O_4\}$. Removing C_3, we arrive at the final configuration (Figure 26) that can be viewed as a cluster $C_4 = \{O_5,\ \ O_7\}$. Otherwise, we can leave these two entities unassigned to any cluster, if these entities are irrelevant for our analysis. We see that using a local criterion based on the maximum split leads to a clustering completely different from the cases A) or B).

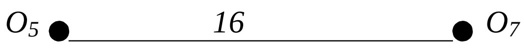

Figure 26: Configuration after removing $C_3 = \{O_3, O_4\}$.

Exercise 6. Describe all clusterings in this example, that is, give an answer consisting of the entire collection of clusters, when we use all possible combinations of local criteria based on the maximum split with $K = 2$, $K = 3$, and $K = 4$.

Example 2 Revisited. We discuss Example 2 again using now the local criterion A), that is, we strive to minimize the diameters of the clusters. Since in this problem only two-element and three-element clusters are acceptable, we start with the value $K = 2$. So the very first cluster is anew $C_1 = \{O_1,\ O_2\}$ with $d(C_1) = 1$. In the set $O \backslash C_1 = \{O_3,\ O_4,\ O_5\}$, a twoelement cluster with the smallest diameter is $C_2 = \{O_3,\ O_4\}$ with. Again, since only two-element and three-element clusters are acceptable, we must assign now the fifth element O_5 to either C_1 or C_2. Thus, we have to reconsider a local algorithm. We apply now the same criterion A) but with the value $K = 3$, that is, we have to calculate the diameters of two possible threeelement subsets: $C`_1 = \{O_1,\ O_2,\ O_5\}$ or $C`_2 = \{O_3,\ O_4,\ O_5\}$. Since $d(C`_1) = 7$ is less than $d(C_2) = 9$, the resulting clustering is $\{\ C`_1,\ C_2\}$, where $C`_1 = \{O_1,\ O_2,\ O_5\}$, $C_2 = \{O_3,\ O_4\}$, and we arrive at the same clustering as before.

In practice, it makes sense to calculate the minimum diameter or radius or the maximum split for $K = 2, 3, \ldots,$ until an increase of K results in an increase of the diameter or radius, or in a decrease of the split.

Answers to Exercises

Exercise 1. Yes, the statement is true.

Exercise 2. Yes, the statement is true.

Exercise 3. We consider only the case when at the first step $K = 4$ and at the second step $K = 2$. Then $d_{min}(O, 4) = 12$, with the corresponding cluster $C_1 = \{O_1, O_2, O_3, O_4\}$. Next, $d_{min}(O \setminus C_1, 2) = 11$, with $C_2 = \{O_5, O_6\}$ and $C_3 = \{O_7, O_8\}$, $d(C_3) = 24$.

Exercise 4. With the same values of $K = 4$ at the first step and $K = 2$ at the second step, we have $r_{min}(O, 4) = 4$ with $C_1 = \{O_3, O_4, O_5, O_7\}$, next, $r_{min}(O \setminus C_1, 2) = 5$ with $C_2 = \{O_1, O_2\}$, and finally, $r(\{O_6, O_8\}) = 20$.

Exercise 5. Another cluster is $\{O_6, O_8\}$, with $s(\{O_6, O_8\}) = 6$.

Exercise 6. Again, with the same parameters $K = 4$ at the first step and $K = 2$ at the second step, we have $s_{min}(O, 4) = 7$ with $C_1 = \{O_1, O_2, O_6, O_8\}$, $s_{min}(O \setminus C_1, 2) = 2$ with $C_2 = \{O_3, O_4\}$ and $C_3 = \{O_5, O_7\}$; or (see Exercise 2) $C_2 = \{O_5, O_7\}$ and $C_3 = \{O_3, O_4\}$.

References

[CP] Hypersearching the Web, by members of the *Clever* Project, *Scientific American*, June 1999, pp. 54-60.

[HJ] P. Hansen, B. Jaumard *Cluster Analysis and mathematical programming.* Mathematical Programming, Vol. 79 (1997) 191-215.

[HJM] P. Hansen, B. Jaumard, N. Mladenovich, *How to choose K entities among N*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 19 (1995), 105-116.

[JD] A. K. Jain, R. C. Dubes, *Algorithms for clustering data.* Prentice Hall, 1988.

[Khey] A. I. Kheyfits, *Introduction to clustering algorithms. DIMACS series of educational modules*, No. 03-01, 2003.

[MA-M] R. López de Màntaras, J. Aguilar-Martín, *Self-learning pattern classification using a sequential clustering technique*, Pattern Recognition, Vol. 18 (1985), No. 3/4, 271-277.

[Murt] F. Murtagh, *A Survey of Recent Advances in Hierarchical Clustering Algorithms.* The Computer Journal, Volume 26 Number 4 (1983).

Web references cited

[1] http://obelia.jde.aca.mmu.ac.uk/multivar/ca.htm

[2] http://grain.jouy.inra.fr/ggpages/DEM/Polymorphism/Waite/Barley_dend.gif

[3] http://taxonomy.zoology.gla.ac.uk/rod/posters/images/dendrogram.gif

[4] http://216.239.57.100/search?q=cache:w91ypUXPCXoC:www.nature.com/cgi-taf/DynaPage.taf

[5] http://www.findarticles.com/cf_dls/m2120/8qy_80/58517866/p1/article.jhtml

[6] http://www.biotech.unl.edu/oldroot/biocomp/gcghelp/growtree.html

[7] http://www.isse.gmu.edu/ snoel/First

[8] http://dmoz.org/Science/Math/Combinatorics/Graph_Theory/References/

[9] http://directory.google.com/Top/Science/Math/Combinatorics/Graph_Theory/References/