DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 36, 1997

# The Role of Applications in Teaching Discrete Mathematics

Fred S. Roberts

## 1. Using Applications Effectively in the Classroom

One of the major reasons for the great increase in interest in discrete mathematics is its importance in solving practical problems. Conversely, practical problems have stimulated the development of discrete mathematics. Applications — discrete or not — should play a major role in the mathematics classroom. They make the subject relevant. They underscore a reason for studying it. They are interesting.

With regard to the role of applications in teaching discrete mathematics, I have developed some rules of thumb over the years, based on my experience with what students respond to and on the philosophy I have developed about the role of applications in mathematics. In my opinion, these rules of thumb are appropriate at all grade levels, though most of my experience with them has been at the college level.

### Rules of Thumb

1. The **Relevance Rule**: Choose applications that are relevant. There are plenty of them.
2. The **Two Are Better Than One Rule**: Never settle for one application when two are available.
3. The **Why Do Things Twice Rule**: Stress the fact that abstract methods developed for dealing with one application are often useful for another.
4. The **Get Real Rule**: Mention real uses of mathematics whenever possible.
5. The **Frontiers Rule**: Show the frontiers of the subject.
6. The **Math Is Alive Rule**: Use applications to show that mathematics is a live subject, done by real people.

1991 *Mathematics Subject Classification.* Primary 00A05, 00A35.

7. The **Motivate Rule**: Let applications motivate theory. Then apply theory to applied problems.
8. The **Don't Be Scared Off Rule**: Don't hesitate to talk about an application because you don't have a background in the subject. Most applications can be explained from general knowledge.
9. The **Modeling Rule**: Choose applications that involve model building. Illustrate the simplifying assumptions in the model and iterate to more complicated (and more realistic) models.

In this paper, I will illustrate these rules of thumb with three examples. In each case, I take one simple mathematical concept and give lots of applications of it. I have used these and similar examples in my college-level courses, but have also used them at all grade levels, including primary grades. The three examples I shall discuss are:

**a:** The traveling salesman problem.
**b:** Graph coloring.
**c:** Eulerian chains and paths.

Almost all of the applications I mention here are discussed in more detail in my book, Roberts [**31**]. For some of them, I will provide additional references, though many of these references are to articles that are more technical in nature.

## 2. The Traveling Salesman Problem

The *traveling salesman problem (TSP)*, in its traditional formulation,[1] is the following: There are $n$ locations. A salesperson must visit all of them, in some order. There is a cost of traveling from location $i$ to location $j$. What is the cheapest route? Most of those who have been exposed to discrete mathematics have seen this problem. They know it is difficult: No one has found a *good TSP algorithm*, that is, a computer algorithm for solving the TSP which is practical for very large $n$, and there is strong evidence that there is none. (The problem belongs to the class of problems that theoretical computer scientists call NP-complete.) Most people who teach discrete mathematics mention the TSP. But you can use it much more effectively by going to the next step: Show how this problem arises in practice in many other forms.

Let me mention some of these other forms.

**The Automated Teller Machine Problem.** Your bank has many ATM machines. Each day, a courier goes from machine to machine to make collections, gather computer information, and so on. In what order should the machines be visited? This problem arises in practice at many banks. One of the earliest banks to use a TSP algorithm to solve it, in the early days of ATM's, was Shawmut Bank in Boston.

[1] Note that the TSP is nowadays frequently referred to as the "traveling salesperson problem". I have chosen to use the historical name.

(This example is from Margaret Cozzens (personal communication), who first developed it as an assignment for her undergraduate operations research class at Northeastern University, and assigned students to study the Shawmut Bank ATM problem, with considerable success.)

**The Phone Booth Problem.** Once a week, each phone booth in a region must be visited, and the coins collected. In what order should that be done?

**The Problem of Robots in an Automated Warehouse.** The warehouse of the future will have orders filled by a robot. Imagine a pharmaceutical warehouse with stacks of goods arranged in rows and columns. An order comes in for ten cases of Tylenol, six cases of shampoo, eight cases of bandaids, etc. Each is located by row, column, and height. In what order should the robot fill the order? The robot needs to be programmed to solve a TSP. In our programs in discrete mathematics for high school and middle school teachers and for high school students at DIMACS (the Center for Discrete Mathematics and Theoretical Computer Science), we sometimes take the students to see a Rutgers University Industrial Engineering robot, which can be used to do exactly this. See [**8, 9**].

**A Problem of X-Ray Crystallography.** In x-ray crystallography, we must move a diffractometer through a sequence of prescribed angles. There is a cost in terms of time and setup for doing one move after another. How do we minimize this cost? See [**4**].

**Manufacturing.** In many factories, there are a number of jobs that must be performed or processes that must be run. After running process $i$, a certain setup cost is inferred before we can run process $j$, a cost in terms of time or money or labor of preparing the machinery for the next process. Sometimes this cost is minimal, for example simply amounting to making minor adjustments, and sometimes it is major, for example requiring complete cleaning of equipment or installation of new equipment. In what order should the processes be run?

These applications illustrate some of my rules of thumb. They all illustrate the **Relevance Rule** (#1) and the **Two Are Better Than One Rule** (#2). They also illustrate the **Don't Be Scared Off Rule** (#8). You don't have to know anything about x-ray crystallography to talk about that application. Yet, I know teachers who are embarrassed to bring in

applications like this because they don't know what some words mean or can't pronounce the words! What is a diffractometer? One of your students might know, or be willing to find out. The entire paper will illustrate these three rules of thumb, so I will usually not explicitly mention those again.

These examples also illustrate the **Get Real Rule** (#4) – it is especially nice to be able to mention real companies (such as Shawmut Bank) that use mathematical methods. I should also note that all of these problems are, in the abstract, the identical problem we have formulated for the TSP. Once we have developed mathematical tools for dealing with the TSP, these same tools can be applied to all of these other practical problems. This illustrates the **Why Do Things Twice Rule** (#3). There are two ways I illustrate this rule. Sometimes, I formulate one version of a problem, translate it into mathematical language (with the students' help), and then develop mathematical methods needed for dealing with the problem. I then formulate another practical problem, show how, in the abstract, it is the same as the first, and then point out that little extra mathematical analysis is needed. At other times, I will formulate a large number of practical problems first, and let the students observe how they are related by formulating them all in the same abstract language, or by guessing why or how they are related.

I should point out that many of these problems in their current formulation involve simplifying assumptions. For example, in the phone booth problem, some telephone booths need to be visited more often than others, since they fill up faster; and in the manufacturing problem, some processes cannot be run before others are completed. In the first round of modeling, these complications are ignored. The next round of modeling should try to handle them. This is an illustration of the **Modeling Rule** (#9). By discussing simplifying assumptions, we teach our students to question assumptions and hypotheses, train them to be more skeptical about technical presentations, and ultimately prepare them to be better decision makers. I always try to involve my students in pinpointing oversimplifications in an initial model for a problem. I also involve them in suggesting how to modify an abstract model to take account of possible complications.

Recently, a group of researchers at four institutions, Rutgers University, AT&T Bell Labs, Bellcore, and Rice University, solved the largest TSP ever solved (up to that time). It had 3038 cities and arose from a practical problem involving the most efficient order in which to drill 3038 holes to make a circuit board (another TSP application). (For information about this, see [**1, 41**].) I like to mention this achievement, and tell my students how a real problem was solved by real people who are at the same institution as I am. This illustrates the **Get Real Rule** (#4), the **Frontiers Rule** (#5), and the **Math Is Alive Rule** (#6): It involves a real application, it is right at the frontiers of modern research, and it was done by real people. Students much prefer to see a real-world application to a make-believe one using "widgets." They get turned on by realizing that they can get to the frontiers of knowledge. They also pay more attention to things that are done

by real people. I know one teacher who believes so strongly in the **Math Is Alive Rule** (#6) that he brings in slides showing pictures of mathematicians whose results he is talking about. Once you have seen a picture of a person, you somehow pay more attention to that person's results, and remember them better by associating them with the picture.

I often use the TSP to introduce the idea of complexity of computation and to motivate an interest in counting and combinatorics. It is a good example to illustrate why one needs to count the number of steps in a computation before implementing it. (Consider the brute force approach of trying all possible orders of the cities in a TSP with, say, 26 cities. Even on a computer that could check one billion orders per second, it would take us almost half a billion years to look at all possible orders.) Once I've introduced the idea of counting the number of steps in a computation, I find it much easier to interest students in methods of counting and combinatorics, which I then relate back to complexity of computation. All of this illustrates the **Motivate Rule** (#7). Students are much more interested in the rules of counting if they see a real application that requires them to be able to count.

## 3. Graph Coloring

A *graph* consists of a set of points or *vertices*, some of which are joined by lines or *edges*. A very old idea is to *color* the vertices of a graph so that if two vertices are joined by an edge, they get different colors. A large number of those who teach discrete mathematics talk about graph coloring. Some mention one application of graph coloring, the historically important application of **map coloring**, where the goal is to color the map with as few colors as possible, so long as countries sharing a border have different colors. We model the countries of a map by vertices of a graph and join two vertices by an edge if their countries have a common boundary. The problem of coloring a map so that countries with a common boundary must get different colors is the same as the problem of coloring the corresponding graph. This is a very important historical example. I like bringing history into my classes, especially when there is a very interesting history of over 100 years that also involves important contributions by non-mathematicians and a historically important use of computers – the first solution to the map-coloring problem used 1200 hours of computer time! That is why I like to use the map coloring example. (For more on its history, see [**2**].) However, there is much more to be said here, because graph coloring has many modern applications which students find both interesting and exciting. I start with some of these examples before going back and giving the historical example. I find that students perk up and take notice from modern and relevant examples. Here are some applications, almost all of which are expanded upon in my papers [**33, 34**].

**Scheduling Meetings of Committees in a State Legislature.** The problem is to assign meeting times so that if two committees have a member in common, they get different meeting times. The solution is to color an appropriate graph. To define a graph, we must say what its vertices and edges are. In this case, the vertices are the committees and there is an edge between two vertices if their corresponding committees have a common member. Then the colors are the meeting times. It should be noted that this problem arises in many places. One particular place of note is the New York State Assembly. (See [5] and [31] for more details.) This illustrates the **Get Real Rule** (#4).

Similar scheduling problems involve assigning final exam times — classes with a common student must get different exam times. Similarly, in an idealized school, students first sign up for classes and then classes are assigned meeting times so that classes with a common student get different meeting times. (This actually happens in some universities, at least for the scheduling of graduate courses in small departments.) Both of these problems are, in the abstract, the identical problem that we have just formulated for the state legislative committees. As with the TSP, once we have formulated the first scheduling problem as an abstract mathematical problem and developed tools for dealing with that problem, we can now "reduce" these new scheduling problems to the old one, in the sense that in the abstract version, they are the same problem and so are amenable to solution using the same tools. This again illustrates the **Why Do Things Twice Rule** (#3).

I usually give simple scheduling problems as examples, have the students translate them into graph problems, and have them try to find graph colorings. We usually end up using a *greedy algorithm* for doing this — color the vertices one at a time, using a new color only if no previously used color can be used. We then ask whether or not we have found a coloring with the fewest number of colors. It is not hard to give examples where such a greedy approach does not work. I point out that graph coloring is again known to be a difficult problem—as with the TSP, there is no known "good" algorithm for finding a graph coloring with the smallest number of colors, and it is unlikely that there will ever be one. This is a place where one can introduce different graph coloring algorithms and use them on practical problems. The abstract methods developed for graph coloring problems that arise from one problem are useful for others. The **Why Do Things Twice Rule** (#3) has again been illustrated. It is no surprise that software designed to solve scheduling problems is sometimes based on graph coloring. It might make a good exercise to have your students explore the software that is used in your school, or to try to write their own programs.

Practical scheduling problems involve many further complications, such as individuals' preferences for when they are to be scheduled, or certain committees being required to meet after certain others. Also, there has

been little mention so far of what makes one schedule (one graph coloring) better than another. This needs to be discussed as well. Is the goal only to use the smallest number of colors? Or is it sometimes good to have a reasonable distribution of colors, i.e., to use each color approximately the same number of times? All of this illustrates the **Modeling Rule** (#9). There is a large literature on scheduling theory: several good references on the subject are the books [**3, 27, 36**].

> **The Channel Assignment Problem.** The problem is to assign channels to radio and television transmitters; transmitters that interfere must get different channels. The solution is to color an appropriate graph. The graph can be defined by letting the vertices be the transmitters and letting an edge correspond to interference. Then, the colors are the channels. Graph coloring methods for solving the channel assignment problem are widely used at such agencies as the Federal Communications Commission, the National Telecommunications and Information Administration, and NATO (the **Get Real Rule** (#4)). See [**12, 6, 34**]

I usually formulate one or two practical problems as graph coloring problems — explaining what to use for vertices and edges and what corresponds to colors. After an example or two, however, I ask the students to help, and they willingly chime in. After hearing about scheduling problems, they can readily translate the channel assignment problem into a graph coloring problem. Indeed, they are eager to think of other problems familiar to them that can be formulated as graph coloring problems.

It is worth mentioning that practical channel assignment problems as well as other applied problems have given rise to a variety of interesting variations of the ordinary concepts of graph coloring. So far, we have not considered what makes one channel assignment better than another. It is not necessarily just that one uses fewer channels than the other; it might be that one has a smaller separation between largest and smallest channel used and thus uses less of the available "spectrum." We have not considered the fact that we might have further restrictions on channels that are closer together than on channels that are further apart but still interfere, or more generally, that we might have different levels of interference. We have not considered the possibility that transmitters might be assigned more than one possible channel over which to transmit, as is the case for mobile radio telephones in cars. The removal of each of these simplifying assumptions leads to an interesting generalization of graph coloring. Some of them are called $T$-colorings, $n$-tuple colorings, and interval colorings [**33**]. Such generalizations are not difficult to explain to students, and many of them are at the forefront of modern research in graph theory. This again illustrates the **Modeling Rule** (#9) and the **Frontiers Rule** (#5).

There is another important point. Real-world channel assignment problems use graphs with thousands of vertices. It is very hard to find the "best" solution under any of a number of definitions of best. Sometimes, we should settle for a solution that can be found in a reasonable amount of time, even if it is not the best. This is a good place to bring in the idea of approximation, and perhaps to mention "heuristic" algorithms that have been developed by real people at real places such as at NATO (the **Math Is Alive Rule** (#6)).

**Garbage Collection Problem.** Garbage trucks follow certain routes in collecting garbage. The problem is to assign each garbage truck route to a day of the week so that if two routes visit a common site, they are scheduled for different days. The solution is to color an appropriate graph. The vertices of that graph are the routes in question and an edge between two routes means that they visit a common site. The colors are the days. This particular problem arose from a more complicated garbage truck routing problem posed by the New York City Department of Sanitation. That problem involves choices of routes as well. This illustrates the **Get Real Rule** (#4) and the **Modeling Rule** (#9). See [**28, 29, 37**].

**Traffic Light Phasing Problem.** We are putting in a new traffic light at a traffic intersection. We need to assign a green light time to each stream of traffic through the intersection so that two streams of traffic that interfere get different green light times. The solution is to color an appropriate graph. The vertices of that graph are the traffic streams, an edge means inteference, and the colors are the green light times. The idea of using graph coloring for phasing new traffic lights was first proposed in an article in a transportation journal, *Transportation Science* [**35**]. (See also [**28**].)

In dealing with the Traffic Light Phasing Problem, we have omitted any discussion of what makes one green light assignment better than another. Also, we are not paying attention to the duration of the green light times, and the fact that one traffic stream might require a longer green light time than another. These complications lead to generalizations of ordinary graph coloring, and in particular the generalization known as interval coloring, which is of current research interest. So, we have again illustrated the **Modeling Rule** (#9) and the **Frontiers Rule** (#5). Stoffers' algorithm for traffic light phasing, and later ones by Opsut and Roberts [**20, 21**] and Raychaudhuri [**24, 25**], are based on linear programming methods to find the best (interval) graph colorings with durations. I like to teach these algorithms to my students, then let them find some local traffic intersections to apply the algorithms to. Margaret Cozzens (personal communication) reports that when her students applied the algorithms to intersections near

the campus of Northeastern University, they found much better traffic light phasings than those actually in use. To complete this really practical experience, they went and convinced the Boston department of transportation to implement their solutions! This is a wonderful example of rule of thumb #4, the **Get Real Rule**.

It should be noted that the same problem arises in scheduling other facilities, such as a classroom, computer, etc. There are different users and some of them interfere. We wish to assign green light times (permission-to-use times) so that interfering users get different green light times. In the abstract, this is the identical problem that we have already analyzed, an illustration of the **Why Do Things Twice Rule** (#3). In addition, the same problem arises in task assignment problems in the workplace. Different tasks need to be assigned times, but some of them interfere because they use the same workers or tools or resources, another illustration of the **Why Do Things Twice Rule** (#3).

> **Fleet Maintenance Problem.** Vehicles (cars, planes, ships) are coming into a facility for regular maintenance according to a fixed schedule. We wish to assign a space to each vehicle. If two vehicles are there at the same time, they must get different spaces. The solution to this problem is to color an appropriate graph. Its vertices are the vehicles and there is an edge between two vehicles if they are in the facility at the same time. The colors are the spaces. (This is the first example I have given where the colors are not times or days or something like that. Students usually see this fairly quickly.) It should be remarked that this problem was first worked on at IBM for ship maintenance (the **Get Real Rule** (#4)). See [**11, 19, 30**].

The Fleet Maintenance Problem again has its complications: What makes one assignment better than another? What if one vehicle requires more space than another? Physically, do the spaces correspond to points or to rectangles or to circles? Each complication leads to a new variation of graph coloring, much as in the channel assignment problem. Here again we illustrate the **Modeling Rule** (#9) and the **Frontiers Rule** (#5).

I have often given a talk to high school audiences that describes the many applications I have given in this section. (It certainly is an illustration of the **Two Are Better Than One Rule** (#2)!) One of these talks led to a very exciting question by one of the students: "Are there careers in graph coloring?" I think I made my point that day! (No, there are not careers in graph coloring. Yes, there are careers in applying mathematics.)

## 4. Eulerian Chains

Given a graph, a *chain* (or walk or path, depending on what terminology you use) arises if we follow the edges from vertex to vertex; an *eulerian*

*chain* is a chain that uses every edge exactly once. An *eulerian closed chain* is an eulerian chain that begins and ends in the same place. Many of those who teach discrete mathematics mention the problems of finding eulerian chains and eulerian closed chains. Some people talk about their history, by describing the famous problem of the **Königsberg bridges**, which was solved by the mathematician Leonhard Euler in 1736 and gave rise to the subject of graph theory. (See [**2**], and see the article by Newman [**18**] in *Scientific American*, and the accompanying translation of the original memoir by Euler [**10**].) Some people even go beyond this, to describe the following problem (though not always connecting it to eulerian chains).

> **The "Chinese Postman Problem".** A mail carrier walking a route must hit every street in the neighborhood and use the smallest amount of time. What route should the carrier take? This problem was first analyzed using graph theoretical methods by a real postman in China, Guan Meigu (the **Get Real Rule** (#4) and the **Math Is Alive Rule** (#6).) It is not exactly the same problem as that of finding an eulerian closed chain, since the mail carrier can walk down a street a second time. However, the eulerian chain problem enters in a critical way into the solution: If there is an eulerian closed chain, this gives the solution. If not, we simply have to find the smallest number of edges to copy so that in the resulting graph there is an eulerian closed chain. See [**15, 17, 31**].

While some people teaching discrete mathematics go as far as mentioning the Chinese Postman Problem, it is so much better to go further, for instance by noting that the exact same problem arises in *street sweeping* and in *snow removal*. Certain streets in a city have to be swept or cleared, and we wish to do this in the least amount of time [**38, 16, 29, 31**]. Again, we have an illustration of the **Why Do Things Twice Rule** (#3). As it turns out, these problems have interesting complications: Only some streets need to be swept every day; there are one-way streets; it takes much longer to go down a street while sweeping it than it does to go down it when one is just passing through. These complications can be handled, and they lead to interesting variations of the Chinese Postman Problem and wonderful exercises for students [**38**]. Here again, we have illustrated the **Modeling Rule** (#9).

Here is another problem that is really the same:

> **Automated Graph Plotting by Computer.** We wish to draw a graph (with pre-specified vertex locations) by computer. When we repeat an edge, we need to pause the computer and raise the plotter pen off the paper. We draw lots of copies of the same graph and so would like to design a way of drawing it which uses as little time as possible. This is again

the Chinese Postman Problem. It has modern practical applications in chip design at IBM, drawing circuit diagrams, electrical and water networks for cities (it has been widely used in Bonn, Germany, for example), control of machines for producing lithographic masks, and so on. See [**13, 26**]. Again, we have illustrated the **Why Do Things Twice Rule** (#3) and the **Get Real Rule** (#4).)

There are other, more subtle applications of eulerian chains. For instance, eulerian chains arise in a telecommunications problem which is concerned with how to tell the position of a rotating roof antenna without going to the roof. The solution involves finding so-called deBruijn diagrams, which also can be connected to the design of computing machines through the theory of shift register sequences. See [**31**] for a discussion.

How many people know that eulerian chains have played a crucial role in the history of molecular biology? They were used in early algorithms for finding an RNA chain given fragments of it that were produced from decomposition by various enzymes. The first RNA chain was determined in 1965 by R.W. Holley and his co-workers at Cornell, using a method that soon was improved using eulerian chains and paths. The specific use of eulerian chains is a bit complicated. However, I can build up to it in several class periods, which involve some rather simple but beautiful applications of the basic counting rules of combinatorics. (See [**31**], Sections 2.13 and 11.4.4.)

After I describe, or at least mention, the use of eulerian chains in molecular biology, I usually lead into a discussion of the many applications of discrete mathematics to modern molecular biology. In particular, I mention the importance of graph theory and combinatorics in the Human Genome Project, the project of mapping and sequencing the entire human genome. For more on this subject, see for example [**7, 14, 23, 32, 39, 40**]. I like to give specific problems here that have come out of recent research in computational biology. Some of these involve eulerian chains and paths (as for example in connection with the "double digest problem" in DNA physical mapping [**22**]). Others involve a variety of questions in graph theory and combinatorics. This illustrates, once again, the **Frontiers Rule** (#5). I also mention the increasing collaboration between the biological sciences community and the mathematical sciences community and the realization on the part of biological scientists that many of their problems are basically problems that are amenable to formulation using discrete mathematics. I give examples of the many collaborations between local mathematicians/computer scientists and local biological scientists that have come about in recent years (the **Math Is Alive Rule** (#6)).

## 5. Concluding Remark

The main message of this paper, and the main reason that we wish to use applications in our courses, can be summed up as follows. There are so

many exciting, relevant applications of discrete mathematics that if you are a good teacher, none of your students should ever again have to ask: *What is mathematics good for?*

## References


[1] Applegate, D., Bixby, R., Chvatal, V., and Cook, B. "Finding Cuts in the TSP," DIMACS Technical Report 95-05, DIMACS Center, RUTGERS University, Piscataway NJ, 1995.

[2] Biggs, N.L., Lloyd, E.K., and Wilson, R.J., *Graph Theory 1736-1936*, Oxford University Press, London, 1976.

[3] Baker, K.R., *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

[4] Bland, R.G., and Shallcross, D.F., "Large Traveling Salesman Problems Arising from Experiments in X-Ray Crystallography: A Preliminary Report on Computation," *Oper. Res. Let.*, 8 (1989), 125-128.

[5] Bodin, L.D., and Friedman, A.J., "Scheduling of Committees for the New York State Assembly," Tech. Report USE No. 71-9, Urban Science and Engineering, State University of New York, Stony Brook, 1971.

[6] Cozzens, M.B., and Roberts, F.S., "T-Colorings of Graphs and the Channel Assignment Problem," *Congr. Numer.*, 35 (1982), 191-208.

[7] DeLisi, C., "Computers in Molecular Biology: Current Applications and Emerging Trends," *Science*, 240 (1988), 47-52.

[8] Elsayed, E.A., "Algorithms for Optimal Material Handling in Automatic Warehousing Systems," *Int. J. Prod. Res.*, 19 (1981), 525-535.

[9] ———, and Stern, R.G., "Computerized Algorithms for Order Processing in Automated Warehousing Systems," *Int. J. Prod. Res.*, 21 (1983), 579-586.

[10] Euler, L., "The Königsberg Bridges," *Sci. Amer.*, 189 (1953), 66-70. (Translation from $18^{th}$ century article.)

[11] Golumbic, M.C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[12] Hale, W.K., "Frequency Assignment: Theory and Applications," *Proc. IEEE*, 68 (1980), 1497-1514.

[13] Korte, B., "Applications of Combinatorial Optimization," in M. Iri and K. Tanabe (eds.), *Mathematical Programming: Recent Developments and Applications*, KTK Scientific Publishing, Tokyo, and Kluwer Academic Publishers, Dordrecht, 1989, pp. 1-55.

[14] Lander, E.S., and Waterman, M.S. (eds.), *Calculating the Secrets of Life: Applications of the Mathematical Sciences in Molecular Biology*, National Academy Press, Washington, DC, 1995.

[15] Lawler, E.L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[16] Liebling, T.M., *Graphentheorie in Planungs-und Tourenproblemen*, Lecture Notes in Operations Research and Mathematical Systems No. 21, Springer-Verlag, New York, 1970.

[17] Minieka, E., *Optimization Algorithms for Networks and Graphs*, Dekker, New York, 1978.

[18] Newman, J.R., "Leonhard Euler and the Königsberg Bridges," *Sci. Amer.*, 189 (1953), 66.

[19] Opsut, R.J., and Roberts, F.S., "On the Fleet Maintenance, Mobile Radio Frequency, Task Assignment, and Traffic Phasing Problems," in G. Chartrand, et al. (eds.), *The Theory and Applications of Graphs*, Wiley, New York, 1981, 479-492.

[20] ———, "I-Colorings, I-Phasings, and I-Intersection Assignments for Graphs, and their Applications," *Networks*, 13 (1983), 327-345.

[21] ———, "Optimal I-Intersection Assignments for Graphs: A Linear Programming Approach," *Networks*, 13 (1983), 317-326.
[22] Pevzner, P.A., "DNA Physical Mapping and Alternating Eulerian Cycles in Colored Graphs," *Algorithmica*, 13 (1995), 77-105.
[23] Pieper, G.W., "Computer Scientists Join Biologists in Genome Project," *SIAM News*, January 1989, 18.
[24] Raychaudhuri, A., "Optimal Scheduling of Subtasks under Compatibility and Precedence Constraints," *Congr. Numer.*, 73 (1990), 223-234.
[25] ———, "Optimal Multiple Interval Assignments in Frequency Assignment and Traffic Phasing," *Discr. Appl. Math.*, 40 (1992), 319-332.
[26] Reingold, E.M., and Tarjan, R.E., "On a Greedy Heuristic for Complete Matching," *SIAM J. Comput.*, 10 (1981), 676-681.
[27] Rinnooy Kan, A.H.G., *Machine Scheduling Problems: Classification, Complexity and Computation*, Nijhof, The Hague, 1976,
[28] Roberts, F.S., *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1976.
[29] ———, *Graph Theory and its Applications to Problems of Society*, NSF-CBMS Monograph No. 29, Society for Industrial and Applied Mathematics, Philadelphia, 1978.
[30] ———, "On the Mobile Radio Frequency Assignment Problem and the Traffic Light Phasing Problem," *Annals NY Acad. Sci*, 319 (1979), 466-483.
[31] ———, *Applied Combinatorics*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
[32] ——— (ed.), *Applications of Combinatorics and Graph Theory to the Biological and Social Sciences*, IMA Volumes in Mathematics and its Applications, Vol. 17, Springer-Verlag, New York, 1989.
[33] ———, "From Garbage to Rainbows: Generalizations of Graph Coloring and their Applications," in Y. Alavi, G. Chartrand, O.R. Oellermann, and A.J. Schwenk (eds.), *Graph Theory, Combinatorics, and Applications*, Vol. 2, Wiley, New York, 1991, pp. 1031-1052.
[34] ———, "T-Colorings of Graphs: Recent Results and Open Problems," *Discr. Math.*, 93 (1991), 229-245.
[35] Stoffers, K.E., "Scheduling of Traffic Lights – A New Approach," *Transportation Res.*, 2 (1968), 199-234.
[36] Slowinski, R., and Weglarz, J. (eds.), *Advances in Project Scheduling*, Elsevier, Amsterdam, 1989.
[37] Tucker, A.C., "Perfect Graphs and an Application to Optimizing Municipal Services," *SIAM Rev.*, 15 (1973), 585-590.
[38] ———, and Bodin, L., "A Model for Municipal Street-Sweeping Operations," in W.F. Lucas, F.S. Roberts, and R.M. Thrall (eds.), *Discrete and System Models*, Vol. 3 of *Modules in Applied Mathematics*, Springer-Verlag, New York, 1983, pp. 76-111.
[39] Waterman, M.S. (ed.), *Mathematical Methods for DNA Sequences*, CRC Press, Boca Raton, FL, 1989.
[40] Waterman, M.S., *Introduction to Computational Biology: Maps, Sequences, and Genomes*, Chapman and Hall, 1995.
[41] Zimmer, C., "And One for the Road," *Discover*, January 1993, 91-92.

Department of Mathematics, Center for Operations Research (RUTCOR), and Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Rutgers University, New Brunswick, NJ 08903
*E-mail address*: froberts@dimacs.rutgers.edu