DIMACS Series in Discrete Mathematics
and Theoretical Computer Science

Greedy Algorithms in Economic Epidemiology

Fred S. Roberts

Abstract. Economic issues are central to the control of disease because of the limited funds available for public health everywhere in the world, even in the wealthiest nations. These economic issues are closely related to issues of individual human behavior, as well as to fundamental disease processes and their relation to the environment. While mathematical formulation of epidemiological processes is an old discipline, combining such formulations with economic, behavioral, and environmental formalisms is relatively new, and has come to define the field of "Economic Epidemiology." Many problems in Economic Epidemiology can be formulated as optimization problems. The simplest approach to solving such a problem is often a greedy algorithm, one that always chooses the best available (cheapest, highest rated, ...) alternative at each step. We review some classical operations research problems arising in Economic Epidemiology for which the greedy algorithm in fact gives an optimal solution, and others for which it can be guaranteed to be reasonably close. We then present two examples from our own work. Examples will be chosen from: assigning workers to health care tasks; choosing medical supplies to maximize value and minimize cost; locating a health care facility so as to minimize the travel times of users; and reopening flooded roads to allow the passage of emergency vehicles. Other examples will include optimal strategies for vaccination given a limited supply; and optimal strategies for sequencing medical tests or public health interventions in order to minimize costs and maximize success.

1. Introduction

Economic issues are central to the control of disease because of the limited funds available for public health everywhere in the world, even in the wealthiest nations. These economic issues are closely related to issues of individual human behavior, as well as to fundamental disease processes and their relation to the environment. While mathematical formulation of epidemiological processes is an old

2010 *Mathematics Subject Classification.* Primary: 92C60 Medical epidemiology, 05C90 Graph theory applications, 05C05 Trees, 90B80 Discrete location and assignment, 91B99 Economics .

The author gratefully acknowledges the support of the National Science Foundation under grant numbers INT-0629714, INT-0629720, EIA-0205116, and DMS-0829652 to Rutgers University.

discipline, combining such formulations with economic, behavioral, and environmental formalisms is relatively new, and has come to define the field of "Economic Epidemiology."

Economic Epidemiology deals with the interplay among economics, individual and group human behavior, and disease ecology, and depends heavily on mathematical formulations of this interplay.[1] Among other things the field seeks optimal strategies and policy to improve our understanding of the spread of infectious agents and of ways to control that spread [**5, 6, 7, 61**]. In recent years, many mathematical models have been developed to compare alternative public health intervention strategies. However, these models commonly disregard varying individual responses to disease outbreaks and the economic contexts in which public health policies and individual responses to them must be evaluated. To evaluate health interventions and potential public policies, models of disease spread must incorporate behaviors of individuals and groups and economic scenarios, and we must develop methods that individuals and public institutions can utilize in response to or preparation for disease events. Some recent surveys of economic epidemiology are [**43, 76**]. Some workshops on Economic Epidemiology contain useful introductions to the field. See [**25, 26**]).

Many problems in Economic Epidemiology can be formulated as optimization problems:

- Find a solution that maximizes or minimizes some value such as lives saved.
- Find the optimal location for a clinic or hospital.
- Find the optimal assignment of health care workers to jobs.
- Optimize investment in health care supplies.
- Minimize the total cost of a series of interventions or medical tests.
- Control an outbreak with as small an investment in vaccines as possible.

Such optimization problems naturally invite inclusion of economic variables. They are also readily modified to include ways of measuring behavioral variables, for example through inclusion of individual or societal utility functions and measures of risk averseness. The challenge here is how to measure utility, risk, and more subtle factors such as peer pressure, motivation, etc. For a discussion of measurement of relevant epidemiological and behavioral variables, see [**66, 67, 68, 69**]. Some ideas of how to measure behavioral variables that can be brought into optimization models are discussed in [**55**], which summarizes a DIMACS workshop aimed at modeling social responses to bioterrorism involving infectious agents.

Often, the simplest approach to an optimization problem is a ***greedy algorithm***: Choose the best (cheapest, highest-rated, ...) available alternative at each step. In general, greedy algorithms will find locally optimal solutions, but not globally optimal ones. We give examples from Economic Epidemiology, some where a greedy solution achieves a global optimum and others where it doesn't, but where we can either make modifications or get a bound on how far from optimal we are. We quickly review some classical operations research problems arising in Economic Epidemiology for which the greedy algorithm in fact gives an optimal solution and others for which it can be guaranteed to be reasonably close. We then present two longer examples from our own work. Examples will be chosen from: assigning

[1]Thanks to Abba Gumel and Ramanan Laxminararayan for introducing me to the topic of Economic Epidemiology and for the ideas in this paragraph.

TABLE 1. Workers and Jobs they Are Qualified For

Worker W_i	W_1	W_2	W_3	W_4	W_5	W_6
Jobs Qualified For	$J_2, J_3, J_5,$	J_2, J_3	J_2, J_3	J_2, J_3	J_1, J_5	J_3
Cost of Using W_i	90	50	80	60	60	100

workers to health care tasks; choosing medical supplies to maximize value and minimize cost; locating a health care facility so as to minimize the travel times of users; and reopening flooded roads to allow the passage of emergency vehicles. The more detailed examples concern optimal strategies for vaccination given a limited supply; and optimal strategies for sequencing medical tests or public health interventions in order to minimize costs and maximize success.

2. Assigning Health Care Workers to Jobs

Suppose we have n health care workers $W_1, W_2, \ldots, W_n$ and m health care jobs $J_1, J_2, \ldots, J_m$ to be filled. We know which workers are qualified to do which jobs and the cost of using each worker. Our goal is to assign workers to jobs they are qualified for, each to at most one job, filling as many jobs as possible, and among all ways of filling as many jobs as possible, find the way to do it with minimum total cost. This is known in Operations Research as the ***Minimum Cost Assignment Problem***. (For a discussion of this problem, see for example [**9**].)

The following is a greedy algorithm for this problem. (See [**9**].) Create a set C of workers chosen. Choose first for C the least expensive worker and assign the worker to any acceptable job. (Choose arbitrarily throughout all the steps if there are ties.) At each stage, add the next least expensive worker to C if there is an acceptable (feasible) assignment using all the workers in the new C obtained by adding this worker. (This might require reassigning jobs from an earlier assignment of workers previously in C.) If a next least expensive worker cannot be added to C, go to the next least expensive one in the list. Continue until you have gone through all the workers. (This is a greedy algorithm because at each step we are choosing the least expensive worker.)

To illustrate this algorithm, consider the data of Table 1. With this data, the algorithm proceeds as follows.

- Step 1: Pick W_2 for C and assign it to, say, J_2.
- Step 2: Pick W_4 for C and use the assignment $W_2 - J_2, W_4 - J_3$. Now, $C = \{W_2, W_4\}$.
- Step 3: Pick W_5 and use the assignment $W_2 - J_2, W_4 - J_3, W_5 - J_5$. Now, $C = \{W_2, W_4, W_5\}$.
- Step 4: Pick W_3. Note that W_2, W_3, W_4 can only be assigned to J_2 and J_3, so there is no way to add W_3 to the set C.
- Step 5: Pick W_1. This can only be assigned to J_2, J_3, or J_5, all of which are already in the assignment. However, we could switch W_5 to J_1 and then assign W_1 to J_5, so we add W_1 to C. C is now $\{W_1, W_2, W_4, W_5\}$.
- Step 6: Pick W_6. Since one of W_2, W_4 has to be assigned to J_3, there is no way to make an assignment for W_6. Thus, we end with $C = \{W_1, W_2, W_4, W_5\}$ and only jobs J_1, J_2, J_3, J_5 are covered.

This solution turns out to be optimal. It is not hard to show that the greedy algorithm always gives an optimal solution to the Minimum Cost Assignment Problem. (See [**9**] for a discussion of why this works.)

3. Investing in Health Care Options

Suppose we are faced with a selection of health care options in which to invest. Option i has an estimated cost c_i and an estimated value v_i. We might be considering alternative health care facilities; alternative supplies for a clinic; alternative research programs; alternative interventions; etc. The problem is to determine which ones to invest in so that the total cost is within budget and the total value is as large as possible.

This is an example of a ***Knapsack Problem***. We have a fixed size knapsack, a number of items we can take, each of a certain size and value, and we want to choose items for the knapsack so as to maximize their value but not exceed the capacity of the knapsack. In general, knapsack problems are difficult to solve computationally: They are NP-complete. The problem we have posed can be stated as follows:

$$\text{Maximize } \sum v_i x_i$$
$$\text{Subject to } \sum c_i x_i \leq B,$$

where x_i = number of items i chosen. There are several variants of the problem. For example, if we allow at most one of each item, then we have the restriction $x_i = 0$ or 1. If the number of items i we can take is bounded for each i, we speak of a ***Bounded Knapsack Problem*** and have the restriction $x_i \in \{0, 1, \ldots, b_i\}$. If we allow an arbitrary number of copies of each item, then we speak of the ***Unbounded Knapsack Problem*** and x_i can be any integer. We shall discuss the latter problem here.

A greedy algorithm for the Unbounded Knapsack Problem is due to George Dantzig [**24**]. Sort items in decreasing order of value per unit cost: v_i/c_i. Pick as many copies of the first item as possible until no more are possible or until one more would violate the budget constraint $\sum c_i x_i \leq B$. Continue in the same way with the second item, then the third, etc. (This is a greedy algorithm since at each step we pick the most valuable item available to us that does not violate the constraint.) To illustrate, consider the data in Table 2. The options in the case of AIDS prevention could be things like: (1) Condom Distribution; (2) Educational Posters; (3) Distribution of Clean Needles; (4) Testing Programs; (5) Funded Research; etc. With this data, the algorithm proceeds as follows.

- Item 1 has the highest ratio v_i/c_i, i.e., 10. We can pick two copies of item 1, with a total cost of 70+70 = 140.
- Item 3 has the next highest ratio v_i/c_i, i.e., 9. We can pick one copy of item 3. The total cost of our knapsak is now 140+35 = 175.
- Next, we pick three copies of item 5, and our knapsack cost is 175+24 = 199.
- Next, we consider item 4, but it is too costly to add to the knapsack.
- Finally, we pick one copy of item 2, and our knapsack cost is 199+5 = 204.

For the Unbounded Knapsack Problem, the greedy algorithm does not always attain the optimal value. For example, consider the data of Table 3. Then the

TABLE 2. Value and Cost of Alternative Health Care Investments

Option i	1	2	3	4	5
Value v_i	700	15	315	35	64
Cost c_i	70	5	35	7	8
Available Budget $B = 205$					

TABLE 3. Value and Cost of Alternative Health Care Investments

Option i	1	2	3
Value v_i	50	52	1
Cost c_i	10	11	1
Available Budget $B = 99$			

greedy algorithm has us pick nine copies of item 1 plus nine copies of item 3, at a total value of 459, while picking nine copies of item 2 gives us a higher value, 468. In fact, the greedy algorithm can lead to solutions that are not close to the optimal value. However, if A is the maximum total value $\sum v_i x_i$ achievable, then the greedy algorithm always achieves a total value of at least $A/2$ [**79**]. Is this a satisfactory outcome? It depends upon the application and the tradeoff between the need to make a speedy decision and the ability to wait and do a complicated calculation. For instance, in a natural disaster such as a hurricane or earthquake, or a rapidly escalating newly-emerging disease outbreak, we might need to make rapid decisions. A case in point is a bioterrorist attack. After it is discovered, e.g., when a plume might still be in the air, we might only have several hours or even minutes to make quick decisions in response. However, if we have the time to do a more time-consuming calculation, we might do better than the relatively speedy greedy algorithm. For the Bounded Knapsack Problem, the algorithm can lead to a solution that is quite a bit further from the optimal. In fact, consider the case where there are two items, with $v_1 = 2, c_1 = 1, v_2 = v, c_2 = v$. If the budget B is v and at most one copy of each item can be chosen, the greedy algorithm chooses item 1 for a total value of 2, whereas choosing item 2 gives a total value of v. Thus, the ratio between the value of the greedy solution and that of the optimal solution can be arbitrarily close to 0.

Representative examples of greedy algorithms for general knapsack problems can be found in [**1, 62, 63, 74**]. Some references on greedy algorithms for the Unbounded Knapsack Problem are [**38, 42, 49, 53, 54**].

4. Locating Health Care Facilities

Suppose that we have a number of users of a planned set of health care facilities. Where do we put the facilities and how do we assign a user to a facility? Suppose we always associate a user to the facility she can get to with minimum cost (choosing arbitrarily in case of ties). There are two costs associated with a choice of locations for the facilities. One is the cost f_i of locating (building, opening, maintaining) a facility at location i. This is either a one-time, fixed cost, or a continuing cost, or a combination of the two. Another cost is the cost c_{ij} of assigning a user at location j to a facility at location i, which is the distance between i and j (or in some applications the distance multiplied by the importance of the user at j

or the demand for facility use at j). We wish to minimize the sum of these two costs, summed over all facilities that are located and over all j corresponding to users. Note that we allow more than one facility and there is a tradeoff between the increased cost if we have more facilities and the corresponding reduced cost of getting users to them.

Given a set S of facilities, let F be the sum of costs f_i over all facilities in S and C be the sum of distances c_{ij} over all users j, where i is the (a) closest facility to j. We seek to find S so that $F + C$ is as small as possible. Of course, if there is a one-time fixed cost for locating facilities and a minimal cost for maintaining them, this could be dominated by the continuing costs of getting users to facilities, and the costs f_i could be disregarded entirely. However, in other cases, the facility location costs are quite high and continuing regularly. There are many variants of the facility location problem. (For reference, see for example [**28**].) This version is called the ***Uncapacitated Facility Location Problem*** - uncapacitated since we have no limit on the number of facilities.

In many cases in the literature, the potential facility locations and the users are located along the edges or at the nodes of a network, with edges having "weights" corresponding to distance between nodes or some other measure of cost of going between the nodes. Charikar and Guha [**18**] consider the case where both facility locations and users are only at nodes of such a network. They use a greedy algorithm to first find a preliminary solution S. To find this, order the nodes of the network in order of increasing cost of locating a facility at the node (choosing arbitrarily in case of ties). Choose p so that if S is the set of the first p facilities, then the cost $F + C$ associated with S is as small as possible. (Note of course that adding facilities would increase F, but decrease (or not increase) C.) (This involves a greedy algorithm because one chooses at each stage the node where the cost of building a facility is as small as possible. This is then greedy in a second sense of choosing how many nodes to include so as to minimize a combination of costs, though this latter choice is not repeated at each step in the way we have defined greedy algorithm.)

To illustrate this part of the algorithm, consider the network of Figure 1. In this, the number on the edge $\{x, y\}$ represents the distance $d(x, y)$ between the nodes x and y if traveling along that edge, while if u and v are not joined by an edge, $d(u, v)$ is the length of a shortest path between u and v in the network. In our example, all distances are taken to be 1. Thus, the distance $d(b, f)$ between b and f is 2 (take the cycle from b to a to f). The costs of locating the facilities are given in the figure next to each node. Suppose that we have three users, located at b, c, f as indicated by circles in the figure. Ordering nodes in order of increasing cost gives the order f, d, b, e, a, c. First take $S = \{f\}$. Then the costs associated with S are $F =$ cost of $f = 0.5$, $C = d(f, f) + d(b, f) + d(c, f) = 0 + 2 + 3 = 5$, so $F + C = 5.5$. Next, take $S = \{f, d\}$. Here, $F =$ cost of $f +$ cost of $d = 0.5 + 1 = 1.5, C = d(f, f) + d(b, f) + d(c, d) = 0 + 2 + 1 = 3, F + C = 4.5$. The third step is to take $S = \{f, d, b\}$. This gives a cost of $F = 4.5$ and $C = 1$, so $F + C = 5.5$. All subsequent sets S have $F > 4.5$, so $F + C > 4.5$. We conclude by the greedy algorithm that the preliminary solution is $S = \{f, d\}$.

The algorithm continues as follows. At each step, we try to improve the cost of the current solution S, starting with the preliminary solution. Choose a node k at random. (It may be one already in S.) Choose a subset T of facilities in S to remove from S. If a user at node j is closer to k than to the facility i in S she is

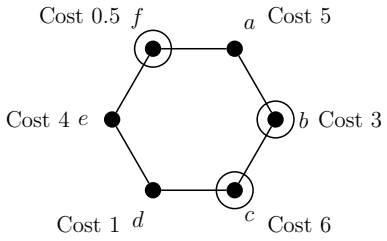

FIGURE 1. Users are located at circled nodes of this network, distances are indicated along edges, and costs of locating facilities are indicated next to nodes where facilities would be located.

assigned to, change her assignment to k and add k to S. If a user at node j was assigned to a facility that was removed from S, i.e., a facility in T, assign her to k and add k to S. Call the resulting set of facilities S^+. For the chosen k, over all possible sets T, find one to remove from S so that S^+ minimizes the total cost $F+C$ of the revised assignment. If the new set of facilities that results has a smaller total cost $F+C$ than that associated with S, use it for the new S. Otherwise, stay with the previous S.

Charikar and Guha [**18**] show that, given ϵ, the algorithm is guaranteed to achieve a cost $F+C$ that is at most $2F^*+3C^*+\epsilon(F^*+C^*)$ in at most $O(nlog(n/\epsilon))$ steps, where F^* and C^* are costs associated with an arbitrary optimal solution.

Note that the way Charikar and Guha define S^+, it is possible that in some steps, node j is not assigned to its closest facility in S^+. We could be tempted to change the definition of S^+ to include a step that would reassign a user at node j to its closest facility i in S^+ and then removing from S^+ all nodes not used for facilities. However, this method does not attain any better bound on the outcome value of $F+C$ and takes longer because we have to find closest facilities.

Representative publications on the Uncapacitated Facility Location Problem, including some with greedy algorithm approaches, are [**16, 34, 39, 40**].

5. Rerouting Emergency Vehicles in Case of Floods

When there is a flood or other natural disaster that results in roads being closed, we need to find fast ways to reroute emergency vehicles such as ambulances. We might want to reroute emergency rescue vehicles to avoid rising flood waters while minimizing delay in provision of medical attention and still getting afflicted people to available hospital facilities. A well-known and widely studied Operations Research problem that arises in this context is the ***Minimum Spanning Tree Problem*** [**70**]. Given a graph or network with positive real numbers as weights on the edges, a ***spanning tree*** is a tree using the edges of the graph and containing all of the nodes. It is ***minimum*** if the sum of the numbers on the edges used is as small as possible. Minimum spanning trees arise in many applications. The emergency vehicle routing problem can be looked at as follows: Given a road network, find usable roads that allow you to go from any node to any other node, minimizing the lengths of the roads used. This is exactly the problem of finding a minimum spanning tree.

A well-known and efficient greedy algorithm for solving the Minimum Spanning Tree problem is Kruskal's algorithm [**47**]: List the edges of the graph in order of

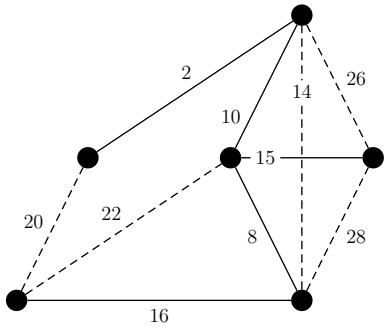

FIGURE 2. The solid edges define a minimum spanning tree.

increasing weight (choosing arbitrarily in case of ties). Going through the edges in this order, for each edge, greedily include it if it does not form a cycle with edges already chosen. Stop when no more edges can be included. (The algorithm is greedy because at each step we choose the least weight alternative that does not violate the constraint about cycles.)

A simple example illustating this algorithm is defined by the network shown in Figure 2. The lowest weight edge is that with weight 2 and the next lowest is that with weight 8. Both are included. The next lowest has weight 10 and it is also included. However, the weight 14 edge, which is next, forms a triangle with the edges of weights 8 and 10, so it is not included. We next include the edge of weight 15 and then the edge of weight 16. The remaining edges all form cycles with edges previously included. The resulting edges form the chosen spanning tree, which is illustrated with the solid lines in the figure. It is well known that Kruskal's algorithm always gives an optimal solution to the Mininum Spanning Tree problem [**22, 47, 70**].

6. Vaccination Strategies for Control of a Highly Infectious Disease Spreading through a Social Network

Many diseases spread through social networks by going from infected people to those with whom they are in contact. We can represent the contacts in a social network in a graph: The nodes are people and an edge between two people indicates contact. In such a social network, nodes are in different states at different times. In one very simple case, suppose there are only two states, susceptible (0) and infected (1). Suppose that $s_i(t)$ gives the state of node i at time t and that times are discrete, i.e., that $t = 0, 1, 2, \ldots$. In the simple case of a highly infectious disease, let us assume that an individual changes state from susceptible to infected at time $t + 1$ if at least one of its neighbors are in the infected state at time t and that an individual never leaves the infected state. (More generally, we can study the variant of this model where an individual changes state from susceptible to infected at time $t + 1$ if at least k of its neighbors are in the infected state at time t. This model has been studied in [**27**], who call it an ***irreversible k-threshold process.*** See also [**11, 46**].)

In this simple situation, we can investigate vaccination strategies. Let us say you have a limited amount of vaccine available each time period, say v doses. Whom should you vaccinate? More precisely: What vaccination strategy minimizes

number of people ultimately infected if a disease breaks out with one infection? Here, we assume that a vaccinated node cannot move into the infected state. This problem in the context of the model above is sometimes known as the ***Firefighter Problem***. The latter problem arises when we have a forest and a fire (epidemic) starts at one tree. At each time period, some trees are burning (infected) and once they start burning, they don't stop. At every time period, we can place v firefighters (think of vaccinating) at unburned nodes. Then the fire spreads to neighboring trees that are not yet burning (susceptible) and not protected by a firefighter (vaccinated node). A rather large literature has grown up about this problem. For some references, see [**27, 36**]. Some of the questions that have been asked in connection with the firefighter problem include:

- Can the fire be contained?
- How many time steps are required before the fire is contained?
- How many firefighters per time step are necessary?
- What fraction of all nodes will be saved (burnt)?
- Does where the fire breaks out matter?
- What if the fire starts at more than one node?
- How does containment work for different graph topologies?
- How can we construct graph topologies to minimize damage?
- What is the complexity of different formalizations of this problem in the computer science sense?

Let $MVS(G, u)$ be the maximum number of nodes that can be saved in graph G if a fire starts at node u and there is one firefighter per time period. MacGillivray and Wang [**50**] asked whether at least p nodes can be saved, i.e., whether there a vaccination strategy such that $MVS(G, u) \geq p$? They showed that this problem is NP-complete.

The firefighter problem has been studied for a variety of particular types of graphs. To give one example, consider the case of an infinite d-dimensional square grid L_d. Here, nodes are located at points $(i, j, \ldots)$ in d-space with integer coordinates $i, j, \ldots$ and two nodes are neighbors if they differ by 1 on one component. Assume that the fire starts at one node. If $d = 1$, then we have nodes on a line and it is trivial to control a fire in two steps with three burned nodes if the number of firefighters v per time step is 1. If $d = 2$, it is impossible to control the fire (epidemic) with $v = 1$. But $v = 2$ suffices and with this case of two firefighters per time step, the fire (epidemic) can be controlled in eight steps with 18 burned trees (infected individuals). To illustrate the latter, consider Figure 3. Each node has an (x, y) coordinate starting with $x = 0$ on the left-most vertical line and with $y = 0$ on the bottom horizontal line. Suppose a fire starts at node (2,2). Here, nodes on fire are shown with black circles, nodes with firefighters with white circles.

- Step 1. Place two firefighters, at nodes (1,2) and (2,1).
- Step 2. The fire spreads to nodes (2,3) and (3,2). We now place firefighters at (1,3) and (2,4).
- Step 3. The fire spreads to (3,1), (3,3), and (4,2) and we place firefighters at (3,0) and (4,1).
- Step 4. The fire now spreads to (3,4), (4,3), and (5,2) and we place firefighters at (5,1) and (6,2).
- Step 5. The fire spreads to (3,5), (4,4), and (5,3) and firefighters are put at (2,5) and (3,6).

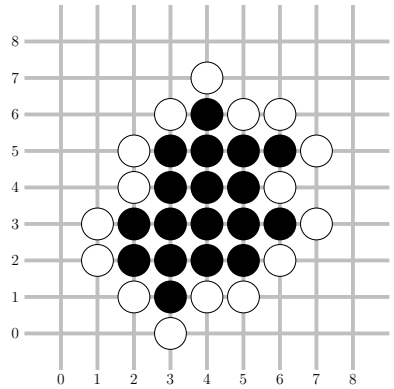

FIGURE 3. Spread of a fire on the grid when we have two firefighters available each time period. Burnt nodes are in black circles, firefighters are placed on nodes with white circles.

- Step 6. The fire spreads to (4,5), (5,4), (6,3) and we place firefighters at (6,4) and (7,3).
- Step 7. The fire spreads to (4,6) and (5,5) and firefighters are placed at (4,7) and (5,6).
- Step 8. The fire spreads to (6,5) and firefighters are placed at (6,6) and (7,5), thus controlling the fire.

We now consider the case $d \geq 3$. A graph is called ***r-regular*** if every node has exactly r neighbors. Wang and Moeller [**77**] observed that in such a graph, $r-1$ firefighters per time step are always sufficient to contain any fire outbreak (at a single node). In L_d, every node has degree $2d$. Thus, when $d \geq 3, 2d-1$ firefighters per time step are sufficient to contain any outbreak starting at a single node. Hartke [**36**] showed that $2d-2$ firefighters per time step are not enough to contain an outbreak of this kind in L_d. Thus, $2d-1$ firefighters per time step is the minimum number required to contain an outbreak in L_d and containment can be attained in two time steps.

If a fire can start at more than one node, then the following are some interesting results about outbreaks in L_d. If $d = 2$, Fogarty [**30**] showed that two firefighters per time step are sufficient to contain any outbreak at a finite number of nodes. If $d \geq 3$, Hartke [**36**] showed that for any positive integer f, f firefighters per time step is not sufficient to contain all finite outbreaks in L_d. In other words, for $d \geq 3$ and any positive integer f, there is an outbreak such that f firefighters per time step cannot contain the outbreak.

Ng and Raff [**59**] have studied L_d and modified the assumption that f is the same for each time period. Let $f(t)$ be the number firefighters available at time t. Ng and Raff assume that $f(t)$ is periodic with period p_f. This could happen for example if firefighters (doses of vaccine) arrive in batches. Then if

$$N_f = f(1) + f(2) + \ + f(p_f),$$

$R_f = N_f/p_f$ is the average number firefighters available per time period. Ng and Raff showed that if $d = 2$ and f is periodic with period $p_f \geq 1$ and $R_f > 1.5$, then an outbreak at any number of nodes can be contained at a finite number of nodes.

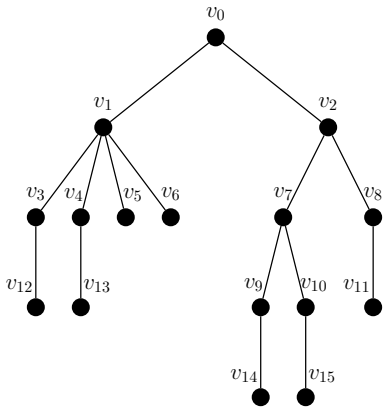

FIGURE 4. The greedy algorithm does not obtain the optimal firefighting solution for this tree.

The firefighter problem has also been studied on graphs that are trees. Consider a rooted tree with a fire starting at the root and suppose that $v = 1$. In a rooted tree, every edge goes down from a node to its immediate "children" one level below it. We define the ***weight*** of node x to be the number of descendants of x in the rooted tree, where a descendant of x is a node obtained from x by following a path that goes from x to a child y of x to a child z of $y \ldots$. In such a tree, once a firefighter is placed at a node, it and all of its descendants are "saved" and cannot be burned. A greedy algorithm for deploying firefighters in a rooted tree is the following: At each time step, place a firefighter at a node x that has not been saved or burned, such that the weight of x is maximized (choosing arbitrarily in case of ties). (This algorithm is greedy since at each step it looks to place a firefighter so as to maximize the number of nodes that can be saved at that stage of the procedure.) Consider for example the tree of Figure 4. Suppose a fire starts at node v_0. Clearly the weight is maximized by using immediate descendants, i.e., children, of v_0, and of these, v_2 has higher weight (7) than v_1 (6). Thus, we put a firefighter at node v_2. In the next time period, the fire spreads to v_1. Note that in the next stage, we can limit ourselves to descendants of v_1 since all descendants of v_2 are saved. We now look at the children of v_1 and put a firefighter on one of them. Two of them maximize weight, i.e., v_3 and v_4, so we put a firefigher on one of them, say v_3. The fire now spreads to the remaining children of v_1, i.e., v_4, v_5, v_6. Finally, we place a firefighter on one of the children of v_4, v_5, v_6, namely v_{13}. In the end, we have saved 11 nodes.

It is not hard to show that the greedy algorithm does not obtain the maximum number of saved nodes, i.e., $M(G, u)$, where u is the root. Indeed, suppose at the first step we put a firefighter at v_1 instead of v_2. The fire now spreads to v_2 and we put the next firefighter at v_7, with the fire spreading to v_8. Finally, we put a firefigher at v_{11}. In this way, we save 13 nodes.

Hartnell and Li [**37**] showed that for any tree G with one fire starting at the root u and one firefighter to be deployed per time step, the greedy algorithm always saves more than one half of $M(G, u)$ nodes, i.e., more than one half of the number of nodes that the best possible algorithm saves. Is this a satisfactory accomplishment?

It could be if time is of the essence, we have a fast-moving epidemic or bioterrorist attack, and we need to develop and implement an efficient algorithm very quickly.

7. Algorithms for Sequential Public Health or Medical Decision Making

Suppose that a patient presents with certain symptoms. Which test do we do first? On the basis of the outcome of the first test, which test do we do next? Tests are expensive. So are false positive and false negative results. In this situation, "cost" is a combination of cost of testing and cost of false results. We might ask: In what order should we do tests in order to minimize total "cost"?

An analogous problem is the following. We have several potential interventions for a public health crisis. Assume funds limit us to one intervention at a time. Which intervention do we invest in first? On the basis of the outcome of the first intervention, which do we launch next? Interventions are expensive. So are false positive and false negative assessments of the outcome of our interventions. "Cost" is a combination of cost of the intervention and cost of false results. In what order should we launch the interventions in order to minimize total "cost"?

Similar "sequential diagnosis" problems arise in many areas: communication networks (testing connectivity, paging cellular customers, sequencing tasks, etc.), manufacturing (testing machines, fault diagnosis, routing customer service calls, etc.), artificial intelligence and computer science (optimal derivation strategies in knowledge bases, best-value satisficing search, coding decision tables, etc.), inspecting containers in ports, etc. An illustrative list of references for such applications includes [**14, 15, 17, 21, 23, 29, 33, 41, 44, 45, 48, 60, 64, 65, 72, 73, 78, 80**]. Sequential diagnosis/decision making is an old subject, but one that has become increasingly important with the need for new models and algorithms as the traditional methods for making decisions sequentially do not scale.

The following approach builds on the approach to sequential diagnosis in inspection of containers at ports, originally presented by Stroud and Saeger [**75**]. In particular, we describe a method that was developed in a series of papers on container inspection: [**2, 10, 51, 52**].

Consider the medical testing problem. A physician is looking to determine if a patient has disease x. The doctor has a variety of tests to choose from (complete blood count, endoscopy, MRI, stress test, etc.). In the end, the patient is to be classified into one of several categories. Simple case: 0 = "doesn't have the disease," 1 = "does have the disease." A testing scheme specifies which test is to be made based on previous observations.

In the case of public health interventions, say we are looking to determine if an epidemic can be controlled. We have a variety of interventions to choose from (close schools if abseenteeism is above 15%, close airports, invest in prophylactics, vaccinate health care workers, etc.). In the end, the epidemic is to be classified into one of several categories, in the simplest case "0 = controllable," "1 = not controllable." An intervention scheme specifies which intervention is to be made based on assessments of previous interventions. We shall present our discussion in the language of the medical testing problem, though an analogous discussion cojuld apply to the public health intervention problem.

The 0's and 1's suggest binary digits (bits). A ***bit string*** is a sequence of bits, e.g., 0001, 1101. A ***Boolean function*** on n variables is a function that assigns

to each bit string of length n a 0 or a 1. Patients have ***attributes*** related to the disease being tested for, each in a number of states. (Sample attributes could be white blood cell count, PSA, creatinin clearance, fever greater than 40 degrees Centigrade, severe cough, severe fatigue). In the simplest case, attributes are in state 0 or 1 (absent or present). Then a patient corresponds to a bit string like 011001 and classification is a decision function F that assigns each bit string to a category. If there are two categories, 0 and 1 ("has disease" or "doesn't have disease"), the decision function F is a Boolean function. This is the case to be discussed here.

There are 2^{2^n} Boolean functions of n variables. This number grows rapidly and it quickly becomes infeasible to look at all possible Boolean functions when considering a testing problem. The problem is then: Given a patient, test her attributes until we know enough to calculate the value of F. A testing scheme tells us in which order to test the attributes to minimize cost. Even this simplified problem is hard computationally. Simplifications include the assumption that F is known. Another is that attributes are independent. Other complications that might arise would be if there were precedence relations in the attributes (e.g., we can't test attribute a_4 before testing attribute a_6). Moreover, the cost of testing for an attribute may depend on attributes tested before, or F may depend on variables that cannot be directly tested or for which tests are too costly. Such problems are hard computationally. There are many possible Boolean functions F. Even if F is fixed, the problem of finding a good (least cost) testing scheme is NP-complete. Several classes of Boolean functions F allow for efficient testing schemes: k-out-of-n systems [**8, 31, 32, 35, 72**], certain series-parallel systems [**4, 13, 57, 71**], read-once systems [**3, 58**], "regular systems" [**12**], and Horn systems [**78**].

We can represent Boolean functions by using binary rooted (directed) trees. All directed edges (arcs) head from a node to either a left or right child. Nodes with no outgoing arcs (no children) are called ***leaves***. When a Boolean function is based on testing for attributes, we can represent it as a binary rooted tree called a ***binary decision tree (BDT)*** where leaf nodes correspond to categories 0 or 1 and other nodes correspond to tests. We take a right arc from a test node if the test indicates that the attribute it is testing for is present, and a left arc otherwise. Consider for example the BDT shown in Figure 5. We reach category 1 from the root by going from a_0 left to a_1, then right to a_2, and then right to 1 or by going from a_0 right to a_2 and then right to 1. Hence, a patient is classified in category 1 iff she has attributes a_1 and a_2 and not a_0 or attributes a_0 and a_2 and possibly a_1. The corresponding Boolean function is given by $F(111) = F(101) = F(011) = 1, F(abc) = 0$ otherwise.

Note that two different BDTs can correspond to the same Boolean function. The BDT of Figure 6 corresponds to the same Boolean function as the BDT of Figure 5. Since the second BDT has fewer test nodes than the first, if all test nodes in a BDT are equally likely to be visited, then the BDT with fewer test nodes would be preferable, at least in the simple case where all tests cost the same and all tests are equally likely to lead to false positives or false negatives. Even in the simple case where the "cost" of a BDT is measured by the number of test nodes, the problem of finding the least cost BDT for a given fixed Boolean function is hard. Doing it by enumerating all possible BDTs is already impractical when there are only four kinds of tests, i.e., we have a Boolean function of $n = 4$ variables. In practice, we don't know what Boolean function to use and seek one that is least

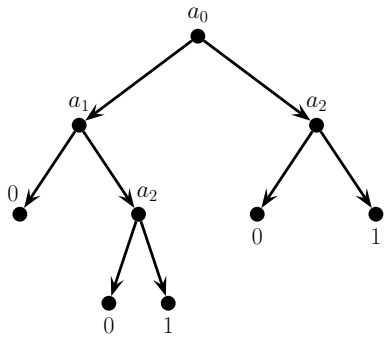

FIGURE 5. A binary decision tree for sequential testing.

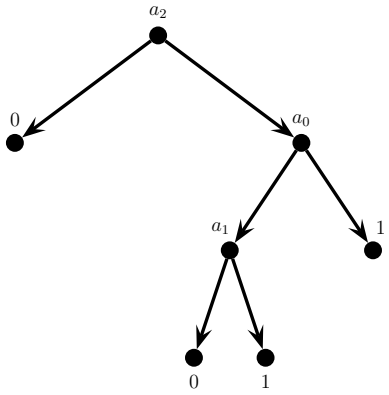

FIGURE 6. A binary decision tree corresponding to the same Boolean function as the BDT of Figure 5.

costly in some sense. Then one approach is to limit the problem to special types of Boolean functions, lest it get totally out of hand.

This is the idea developed by Stroud and Saeger [**75**]. They enumerated all "complete, monotone" Boolean functions and calculated the least expensive corresponding BDTs. Their method is practical for $n \leq 4$ but not for $n = 5$. Given two bit strings $x_1x_2 \ldots x_n, y_1y_2 \ldots y_n$, suppose that $x_i \geq y_i$ for all i implies that $F(x_1x_2 \ldots x_n) \geq F(y_1y_2 \ldots y_n)$. Then we say that F is ***monotone***. Here 111 has highest probability of being in category 1. A Boolean function F of n variables is ***incomplete*** if F can be calculated by finding at most $n-1$ variables and knowing the value of the input string on those variables, and F is ***complete*** otherwise.

Stroud and Saeger found the least cost tree by enumerating all BDTs corresponding to a given complete, monotone Boolean function and repeating this for all complete, monotone Boolean functions. The problem is that there are too many complete, monotone Boolean functions. When $n = 2$, there are 6 monotone Boolean functions, only 2 of them are complete, monotone, and there are 4 BDTs for calculating these 2 complete, monotone Boolean functions. When $n = 3$, there are 9 complete, monotone Boolean functions and 60 distinct BDTs for calculating them. When $n = 4$, there are 114 complete, monotone Boolean functions and 11,808 distinct BDTs for calculating them. (Compare 1,079,779,602 BDTs for all Boolean

functions.) When $n = 5$, there are 6,894 complete, monotone Boolean functions and 263,515,920 corresponding BDTs. (Even worse: compare $5x10^{18}$ BDTs corresponding to all Boolean functions.) We need alternative approaches; enumeration is not feasible!

So far, we have considered only one possibility: that one BDT is cheaper than another if it has fewer nodes. This is oversimplified. There are more complex costs involved than number of tests in a tree. The Stroud-Saeger method can still be applied here. Performing a test has a number of costs. There is a ***unit cost*** of performing the test, a ***fixed cost*** of purchasing the equipment to make the test, and a ***delay cost*** from waiting to take the test and waiting for the results. Much of the work so far has disregarded fixed and delay costs and has been aimed at minimizing unit costs (though Stroud and Saeger have done some work with queueing models to take into account delay costs). Even if we just consider unit costs, the answer may be complicated; it depends on how many (or which) nodes of the decision tree are actually visited during an "average" procedure toward diagnosis. This will depend on the costs of doing a test for a given attribute and on a "distribution" of the disease in the population from which individuals to be tested are chosen. Moreover, it can also depend on the probability of test errors, i.e., the probability that a test for attribute a_i gives a false positive result (saying a_i is present when it is not) or a false negative (saying a_i is not present when it is). If we are given the probability of an error for each kind of test (and assume that this probability is independent of the time that the test is given) and given the a priori probability that a patient has the disease, we can calculate the "expected" cost of utilization of the tree, which we denote by C_{util}. Under these assumptions, we can also calculate the probability P_{FP} that a given BDT leads to a false positive outcome, or the probability P_{FN} that it leads to a false negative outcome in the classification scheme, i.e., the probability that the patient is classified as having the disease when she doesn't or not having it when she does. A challenge is to put costs on these false positive and false negative classifications, C_{FP}, C_{FN} respectively. Both of these costs are hard to measure. In the former case, if it means beginning a series of treatments, it could be expensive, not to mention having a psychological cost to patient. In the latter case, it would mean that the patient would go untreated. In our work [**2, 10, 51, 52**], following Stroud and Saeger, we have used the cost function

$$C_{Tot} = C_{FP} \times P_{FP} + C_{FN} \times P_{FN} + C_{util}$$

.

To repeat: P_{FP}, P_{FN} are calculated from the tree and C_{util} is calculated from the tree, the costs of the tests for the attributes, the a priori probability of disease, and the probabilities of test errors. The values of C_{FP}, C_{FN} are inputs. Note: a model for developing probability of errors in individual tests for attributes, that depends on thresholds, is also developed by Stroud and Saeger [**75**] and by Madigan, Mittal and Roberts [**51, 52**].

Sometimes adding more possibilities results in being able to do more efficient searches. Madigan, Mittal and Roberts expanded the space of trees from those corresponding to Stroud and Saeger's complete, monotonic Boolean Functions to "complete and monotonic" BDTs. There are some advantages to this: Unlike Boolean functions, BDTs may not have to consider all test inputs to give a final

decision. Moreover, we might allow more potentially useful trees to participate in the analysis. In [**51, 52**], a binary tree is called ***monotonic*** if all the left leaves are class "0" and all the right leaves are class "1." A binary tree is called ***complete*** if every type of test occurs at least once in the tree and, at any non-leaf node in the tree, its left and right sub-trees are not identical. We consider the space of ***CM Trees***, trees that are complete and monotonic. This is a much richer space than the space of BDTs corresponding to complete, monotone Boolean functions. If $n = 3$, there 60 BDTs from complete, monotone Boolean functions, but 114 CM trees. If $n = 4$, it is 11,808 for the former, 66,600 for the latter.

Madigan, Mittal, and Roberts define a way of moving in CM Tree Space from one tree to a neighboring one. This follows ideas in the literature from Chipman, George, and McCulloch [**20**] and Miglio and Soffritti [**56**], who provide a comparison of various definitions of neighborhood and proximity between trees, and from Chipman, George, and McCulloch [**19**], who describe methods to traverse the tree space. Specifically, Madigan, Mittal and Roberts define four operations that take a CM tree into a "neighboring" CM tree. We ***split*** a CM tree when we pick a leaf node and replace it with a test node that is not already present in that branch, and then insert arcs from that test node to 0 and to 1. We perform a ***swap*** on a CM tree if we pick a non-leaf node in the tree and swap it with its parent node such that the new tree is still monotonic and complete and no test node occurs more than once in any branch. In the operation ***merge***, we pick a parent node of two leaf nodes and make it a leaf node by collapsing the two leaf nodes below it, or pick a parent node with one leaf node, collapse both the parent node and its one leaf node, and shift the sub-tree up in the tree by one level. Finally, we perform the operation ***replace*** on a CM tree when we pick a test node occurring more than once in the tree and replace it with any other test node such that no test node occurs more than once in any branch. Madigan, Mittal and Roberts showed that any tree in CM tree space can be reached from any other tree by using these neighborhood operations repetitively. These neighborhood operations are the basis of algorithms for searching through CM Tree Space.

The basic idea of Madigan, Mittal, and Roberts is to use a greedy search algorithm. In this algorithm, we randomly start at any tree in CM tree space, find its neighboring trees using the above operations, move to the neighbor with the lowest cost, and iterate until we find a minimum. The problem with this algorithm is that CM Tree Space is highly multi-modal (with more than one local minimum) and it is easy to get stuck at a local minimum. This led Madigan, Mittal, and Roberts to implement a stochastic search algorithm coupled with the method known as simulated annealing to find the best tree. This is a variant of the greedy algorithm. The algorithm is stochastic: It selects a neighboring tree according to a probability distribution over neighboring trees. The simulated annealing aspect involves a so-called "temperature" t, initiated to one and lowered in discrete unequal steps after every h hops so that as the temperature decreases, the probability of moving to the least expensive tree in the neighborhood increases. This greedy search algorithm allowed calculations with n up to 5 types of tests, leading to BDTs with cost very close to (and in many cases equal to) the minimum and finding them significantly faster than existing methods of searching through BDTs obtained from complete, monotonic Boolean functions.

8. Concluding Comments

Allocation of scarce resources, health care or otherwise, has been a topic of mathematical analysis for a long time. In the case of health care, we might try to define "optimality" and seek "optimal" allocation strategies, or near-optimal strategies. In the context of infectious disease prevention, there is the potential to focus our efforts to optimize different, more complex outcomes: the fewest cumulative number of infections, the fewest cumulative number of deaths, the cumulative cost of infection control, the indirect societal impacts caused by an outbreak of disease, the average utility of an outcome to individuals in the population, etc. For many of these, the simplest algorithmic approach is sometimes a greedy one. As we have seen: Sometimes it pays to be greedy.

References

1. Akay, Y., Li, H.,and Xu, S., "Greedy algorithm for the general multidimensional knapsack problem," *Annals of Operations Research*, **150** (2007), 17-29.
2. Anand, S., Madigan, D. Mammone, R., Pathak, S., and Roberts, F., "Experimental analysis of sequential decision making algorithms for port of entry inspection procedures, in S. Mehrotra, D. Zeng, H. Chen, B. Thuraisingham, and F-X Wang (eds.), *Intelligence and Security Informatics, Proceedings of ISI-2006*, Lecture Notes in Computer Science #3975, Springer-Verlag, New York, 2006, 319-330.
3. Angluin, D., Hellerstein, L., and Karpinski, M., "Learning read-once formulas with queries," *Journal of the Association for Computing Machinery*, **40** (1993), 185-210.
4. Arseneau, L., *Optimal Testing Strategies for s,t-Series Parallel Systems*, Master's Thesis, Combinatorics and Optimization, University of Waterloo, 1996.
5. Barrett, S., "Eradication versus control: the economics of global infectious disease policies, *Bull. World Health Organ.*, **82** (2004), 683-688.
6. Bauch, C.T., "Imitation dynamics predict vaccinating behaviour," *Proc. Biol. Sci.*, **272** (2005), 1669-1675.
7. Bauch, C.T., and Earn, D.J., "Vaccination and the theory of games, *Proc. Natl. Acad. Sci. USA*, **101** (2004), 13391-13394.
8. Ben-Dov, Y., "Optimal testing procedures for special structures of coherent systems," *Management Science*, **27** (1981), 1410-1420.
9. Bogart, K.P., *Introductory Combinatorics*, Pitman Publishing, Marshfield, MA, 1983.
10. Boros, E., Elsayed, E., Kantor, P. Roberts, F., and Xie, M., "Optimization problems for port-of-entry detection systems," in H. Chen and C.C. Yang eds.O, *Intelligence and Security Informatics: Techniques and Applications*, Springer, 2008, 319-335.
11. Boros, E., and Gurvich, V., "On complexity of algorithms for modeling disease transmission and optimal vaccination strategy," RUTCOR Research Report 16-2007, Rutgers University, 2007.
12. Boros, E., and Ünlüyurt, T., "Diagnosing double regular systems," *Annals of Mathematics and Artificial Intelligence*, **26** (1999), 171-191.
13. Boros, E., and Ünlüyurt, T., "Sequential testing of series-parallel systems of small depth," in M. Laguna and J. L. Gonzáles Velarde (eds.), *OR Computing Tools for the New Millennium* (Proc. Conference of the INFORMS Computing Society, Cancun, Mexico, January 5-7, 2000), 2000, 39-74.
14. Brule, J.D., Johnson, R.A., and Kletsky, E.J., "Diagnosis of equipment failures," *IRE Transactions on Reliability and Quality Control*, **RQC-9** (1960), 23-34.
15. Butterworth, R., "Some reliability fault testing models," *Operations Research*, **20** (1972), 335-343.
16. Byrka, J., "An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem," in *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2007.
17. Chang, C.L., and Slage, J.R., "An admissible and optimal algorithm for searching and-or graphs," *Artificial Intelligence*, **2** (1971), 117-128.

18. Charikar, M., and Guha, S., "Improved combinatorial algorithms for facility location problems," *SIAM Journal on Computing*, **34** (2005), 803-824.
19. Chipman, H.A., George, E.I., and McCulloch, R.E., "Bayesian CART model search, *Journal of the American Statistical Association*, **93** (1998), 935-960.
20. Chipman, H.A., George, E.I., and McCulloch, R.E., "Extracting representative tree models from a forest," working paper 98-07, Department of Statistics and Actuarial Science, University of Waterloo, 1998.
21. Chiu, S.Y., Cox, L.A., and Sun, X., "Least-cost failure diagnosis in uncertain reliability systems," *Reliability Engineering and System Safety*, **54** (1996), 203–216.
22. Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C., *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, 2001.
23. Cox, L.A., Jr., and Qiu, Y., "Optimal inspection and repair of renewable coherent systems with independent components and constant failure rates," *Naval Research Logistics*, **41** (1994), 771-788.
24. Dantzig, G.B., "Discrete-variable extremum problems," *Operations Research*, **5** (1957), 266-288.
25. DIMACS, DIMACS Workshop on Economic Epidemiology, 2005, http://dimacs.rutgers.edu/Workshops/EconEpid/.
26. DIMACS, DIMACS/MBI/MITACS Workshop on Economic Epidemiology, July-August 2009, http://dimacs.rutgers.edu/Workshops/WSEconEpi/.
27. Dreyer, P.A., and Roberts, F.S., "Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion," *Discrete Applied Mathematics*, **157** (2009), 1615-1627.
28. Drezner, Z, and Hamacher, H.W., *Facility Location: Applications and Theory*, Springer, 2004.
29. Duffuaa, S.O., and Raouf, A., "An optimal sequence in multicharacteristics inspection," *Journal of Optimization Theory and Applications*, **67** (1990), 79-87.
30. Fogarty, P., *Catching the Fire on Grids*, Masters Thesis, University of Vermont, 2003.
31. Garey, M.R., "Simple binary identification problems," *IEEE Transactions on Computers*, **21** (1972), 588-590.
32. Garey, M.R., "Optimal task sequencing with precedence constraints," *Discrete Mathematics*, **4** (1973), 37-56.
33. Geiger, D., and Barnett, J.A., "Optimal satisficing tree searches," in *AAAI91*, Morgan Kaufmann, Anaheim, CA, 1991, 441-445.
34. Guha, S., and Khuller, S., "Greedy strikes back: Improved algorithms for facility location," *Journal of Algorithms,* **31** (1999), 228-248.
35. Halpern, J.Y., "Fault testing for a k-out-of-n system," *Operations Research*, **22** (1974), 1267-1271.
36. Hartke, S.G., *Graph-theoretic Models of Spread and Competition*, Ph.D. Thesis, Rutgers University, 2004.
37. Hartnell, B., and Li, Q., "Firefighting on trees: How bad is the greedy algorithm?", *Congressus Numerantium*, **145** (2000) 1-192.
38. Hu, T.C., and Lenard, M.L., " Optimality of a heuristic solution for a class of knapsack problems." *Oper. Res.*, **24** (1976), 193-196.
39. Jain, K., Mahdian, M., Markakis, E., Saberi, A., and Vazirani, V., "A greedy facility location algorithm analyzed using dual-fitting with factor-revealing LP," *Journal of the ACM*, **50** (2003), 795-824.
40. Jain, K., Mahdian, M, and Saberi, A. , "A new greedy approach for facility location problem," *Proceedings of the Symposium on Theory of Computing*, 2002.
41. Kadane, J.B., "Quiz show problems," *Journal of Mathematical Analysis and Applications*, **27** (1969), 609-623.
42. Kellerer, H., Pferschy, U., and Pisinger, D., *Knapsack Problems,* Springer, New York, 2004.
43. Klein, E., Laxminarayan, R., Smith, D.L., and Gilligan, C.A., "Economic incentives and mathematical models of diseases, *Environment and Development Economics*, **12** (2007), 707-732.
44. Kowalski, R., "Search strategies for theorem proving," in B. Meltzer and D. Mitchie (eds.) *Machine Intelligence*, **5**, Edinburgh University Press. Edinburgh, 1969, 181-201.

45. Kowalski, R., "And-or graphs, theorem proving graphs and bi-directional search," in B. Meltzer and D. Mitchie (eds.) *Machine Intelligence*, **7**, Edinburgh University Press, Edinburgh, 1972, 167-194.
46. Kynčl, J., Lidický, B. and Vyskočil, T., "Irreversible 2-conversion set is NP-complete," preprint, Institute for Theoretical Computer Science, Charles University, 2009.
47. Kruskal, J.B., "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, **7** (1956), 4850.
48. Lauritzen, S.L. and Nilsson, D., "Representing and solving decision problems with limited information," *Management Science*, **47** (2001), 1238-1251.
49. Lueker, G.S., "Two NP-complete problems in nonnegative integer programming," Report No. 178, Computer Science Laboratory, Princeton University, 1975.
50. MacGillivray, G., and Wang, P., "On the firefighter problem," *Journal of Combinatorial Mathematics and Combinatorial Computing*, **47** (2003) 83-96.
51. Madigan, D., Mittal, S., and Roberts, F.S., "Efficient sequential decision-making algorithms for container inspection operations," preprint, DIMACS, Rutgers University, 2009.
52. Madigan, D., Mittal, S., and Roberts, F.S., "Sequential decision making algorithms for port of entry inspection: Overcoming computational challenges, in G. Muresan, T. Altiok, B. Melamed, and D. Zeng (eds.), *Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI-2007)*, IEEE Press, Piscataway, NJ, 2007, 1-7.
53. Magazine, M., Nemhauser, G.L., and Trotter, L.E., Jr., "When the greedy solution solves a class of knapsack problems, " *Oper. Res.*, **23** (1975), 207-217.
54. Martello, S., and Toth, P., *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, England, 1990.
55. McKenzie, E., and Roberts, F.S., "Modeling social responses to bioterrorism involving infectious agents," Technical Report, DIMACS Center, Rutgers University, Piscataway, NJ, July 24, 2003. (Available at http://dimacs.rutgers.edu/Workshops/Modeling/.)
56. Miglio, R., and Soffritti, G., "The comparison between classification trees through proximity measures," *Computational Statistics and Data Analysis*, **45** (2004), 577-593.
57. Monma, C.L., and Sidney, J.B., "Optimal sequencing via modular decomposition: Characterization of sequencing functions," *Mathematics of Operations Research*, **4** (1979), 215-224.
58. Mundici, D., "Functions computed by monotone boolean formulas with no repeated variables," *Theoretical Computer Science*, **66** (1989), 113-114.
59. Ng, K.L., and Raff, P., "A generalization of the firefighter problem on $Z \times Z$," *Discrete Applied Mathematics*, **156** (2008), 730-745.
60. Nilsson, N.J., *Problem-solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.
61. Perry, B.D., and Young, A.S., "The past and future roles of epidemiology and economics in the control of tick-borne diseases of livestock in Africa: The case of theileriosis," *Preventive Veterinary Medicine*, **25** (1995), 107-120.
62. Pisinger, D., "A minimal algorithm for the 0-1 knapsack problem," *Operations Research*, **45** (1997), 758-767.
63. Pisinger, D., "Where are the hard knapsack problems?", *Computers & Operations Research*, **32** (2005), 2271-2284.
64. Pohl, I., "Bi-directional search," in B. Meltzer and D. Mitchie (eds.) *Machine Intelligence*, **6**, Edinburgh University Press, Edinburgh, 1971, 127-140.
65. Reinwald, L.T., and Soland, R.M., "Conversion of limited-entry decision tables to optimal computer programs I: Minimum average processing time," *Journal of the Association for Computing Machinery*, **13** (1966), 339-358.
66. Roberts, F.S., *Measurement Theory, with Applications to Decisionmaking, Utility, and the Social Sciences*, Addison-Wesley, Reading, MA, 1979. Reprinted, Cambridge University Press, Cambridge, UK, 2009.
67. Roberts, F.S., "Meaningfulness of conclusions from combinatorial optimization," *Discr. Appl. Math.* **29** (1990), 221-241.
68. Roberts, F.S., "Limitations on conclusions using scales of measurement," in S.M. Pollock, M.H. Rothkopf, and A. Barnett (eds.), *Operations Research and the Public Sector*, Vol. 6 in *Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, 1994, 621-671.

69. Roberts, F.S., "Meaningful and meaningless statements in epidemiology and public health," preprint, DIMACS, Rutgers University, 2009, submitted for publication.
70. Roberts, F.S., and Tesman, B., *Applied Combinatorics*, 2nd ed., Chapman & Hall/CRC, 2009.
71. Salloum, S., and Breuer, M.A., "An optimum testing algorithm for some symmetric coherent systems," *Journal of Mathematical Analysis and Applications*, **101** (1984), 170–194.
72. Simon, H.A., and Kadane, J.B., "Optimal problem-solving search: All-or-none solutions," *Artificial Intelligence*, **6** (1975), 235-247.
73. Smith, D.E., "Controlling backward inference," *Artificial Intelligence*, **39** (1989), 145-208.
74. Sriram, J., "KS-Solve: Local search for the knapsack problem," http://www.cs.dartmouth.edu/ afra/courses/cs44/winter09/project/report/sriram-report.eps, March 14, 2009,
75. Stroud, P.D, and Saeger, K.J., "Enumeration of increasing Boolean expressions and alternative digraph implementations for diagnostic applications," *Proceedings Volume IV, Computer, Communication and Control Technologies* 2003, 328-333.
76. Tanaka, M.M., Kumm, J., and Feldman, M.W., "Coevolution of pathogens and cultural practices: A new look at behavioral heterogeneity in epidemics," *Theoretical Population Biology*, **62** (2002), 111-119.
77. Wang, P., and Moeller, S.A., "Fire control on graphs," *Journal of Combinatorial Mathematics and Combinatorial Computing*, **41** (2002) 19-34.
78. Wang, J., and Vande Vate, J., "Question asking strategies for Horn clause systems," *AMAI*, **1** (1990), 359-370.
79. http://en.wikipedia.org/wiki/Knapsack_problem#Greedy_approximation_algorithm, September 14, 2009.
80. Yener, A., and Rose, C., "Highly mobile users and paging: Optimal polling strategies," *IEEE Transactions on Vehicular Technology*, **47** (1998), 1251-1257.

DIMACS Center, Rutgers University, Piscataway NJ 08854 USA
E-mail address: froberts@dimacs.rutgers.edu