Finding an Optimal Tooth Color Match

by Cara Cocking

Computer science and dentistry may not be fields you would normally associate with one another, but with the broad applicability of computer science to nearly every other field imaginable, it is not surprising to find the two together in a research project. In this article I describe one such project, in which techniques of combinatorial optimization are used to design a dental shade guide. The problem was brought to my attention by a professor and practicing dentist at my university.

I begin with an overview of the field of combinatorial optimization, including the relevant subfield of facility location. Then I describe the problem and our solution approach, introducing the reader to some of the standard solution techniques for combinatorial optimization problems in the process.

Combinatorial Optimization

You may have heard the aphorism that the more you learn, the less you know. The discipline of computer science is no exception: the more you delve into its depths, the more subfields that open up, each an entire province on its own. Combinatorial optimization is one of these subfields, and it lies at the fuzzy border between computer science and mathematics. Also known as discrete optimization, combinatorial optimization combines techniques of mathematics and computer science to solve optimization problems over discrete structures, e.g., graphs. Cook, et al. [2] present a straightforward introduction to combinatorial optimization.

The Traveling Salesman Problem

You may have already encountered the most famous combinatorial optimization problem: the traveling salesman problem (TSP). Though it has application in various contexts, the TSP gets its name from the problem that a traveling salesman faces when deciding the order in which to visit a set of cities. The salesman seeks to minimize his total distance traveled along a route beginning from his home city, visiting every other city exactly once, and finally returning home.

There are many possible orders in which to visit the cities: if there are n cities, then there are $(n-1)!/2$ different tours through the cities. Each of these tours is called a **feasible solution**. The problem is to find the (or *an*, since there may be more than one) **optimal solution**. However, no simple way of sorting through the $O(n!)$ possibilities has yet been discovered, so this can be a very difficult problem for large n. The TSP, and many of the other problems studied in combinatorial optimization, is NP-hard, a classification that means special solution techniques, possibly producing suboptimal solutions, should be employed for the sake of efficiency. There is an extensive Web site on the TSP [5] for those interested in learning more about this classic problem.

Facility Location

As one would expect, facility location is concerned with optimally locating facilities. The facilities to locate can be, for example, hospitals, fire stations, or warehouses, and the setting may be a town or a region. We can also use facility location to position connectors in a telecommunications network, products in feature space, or, as in our case, tooth colors in color space.

In general, facility location problems can either be continuous or discrete. In the continuous case, facilities may be located anywhere in the plane or space, perhaps with some boundary restrictions. In discrete facility location, the kind we consider here, there is a finite set of locations where facilities may be placed.

There are two main parties involved in any facility location problem: the **facilities** themselves and the **clients** of the facilities. As in the TSP, there are many possibilities for where the facilities can be located, i.e., there are many feasible solutions, and the goal is to find optimal locations. There are different ways to measure optimality, but often we want the facilities to be close to the clients, and thus we use a quantification of "closeness" to measure the goodness of a solution. For example, we may add up the distances from every client to its nearest facility and use this sum as the closeness measure of a feasible solution. Trivially, an optimal solution is one with minimum sum.

Facility location problems are represented using a graph. The nodes of the graph are the union of the clients and the possible facility locations. In some cases a node may represent both a client and a potential facility location. Consider the problem of locating hospitals across a region: the nodes in the graph represent towns, which are the clients of the hospitals as well as the locations where hospitals may be built. In the problems we consider, facilities may be located only at the nodes of the graph.

As an example, consider the graph in Figure 1. Each of the four nodes is a client as well as a potential facility location. The weights on the nodes represent the **demand** of each client (which might be the population in the town), and the weights on the edges represent the **travel cost** between two nodes (which might be the distance in kilometers).

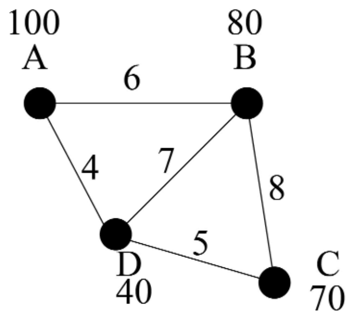

Figure 1: An example facility location problem represented as a graph.

Suppose that we want to select a location for one facility, and our goal is to minimize the total demand-weighted travel cost. To calculate this number, we add up the travel costs, multiplied by demand, of each node's shortest path to the facility node. In this example, if we place a facility at A, the 80 people from B each travel 6 kilometers (80*6=480), the 40 people at D travel 4 kilometers (40*4=160), and the 70 people at C each travel 9 kilometers (70*9=630) to get to A, for a total travel cost of 480+160+630=1270. The 100 people at A do not have to go anywhere, and thus contribute 0 to the total travel cost. Similarly, we can calculate what the total travel cost would be if a facility is located at B (1440), C (1740), or D (1310) and determine that the optimal solution is indeed to build the facility at location A.

If there are n nodes in the graph, and we want to find locations for m facilities, $m < n$, then the problem is to select m nodes from among the n, and there are (n *choose* m) possibilities. While the above n=4, m=1 example appeared trivial, for large values of n, there are far too many feasible solutions to consider. Like the TSP, most facility location problems are NP-hard. For further reading on facility location, I recommend Daskin [3].

Tooth Color and Shade Guides

In dentistry, a crown is placed over a tooth that is badly damaged or decayed. Crowns are typically made of a ceramic material, and it is desirable that this material matches the natural tooth color as closely as possible. For this purpose, companies have manufactured shade guides (see Figure 2), which include a series of shade tabs, each with a bit of ceramic on the tip, representing a range of colors. These tabs are held up next to the tooth to find the best match so that the appropriately colored ceramic can be used for the crown.

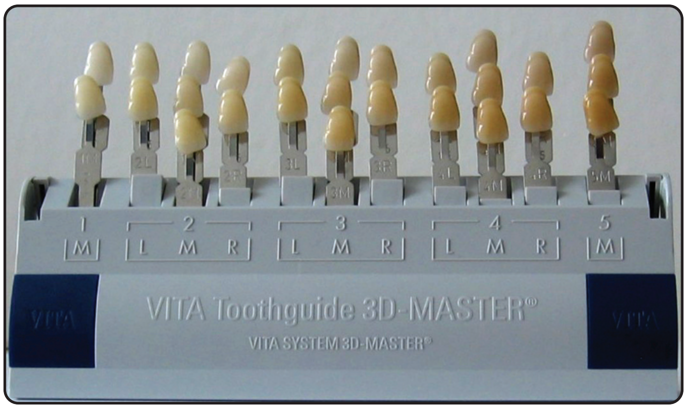

Figure 2: A dental shade guide.

The problem we consider here is the design of an optimal shade guide. Informally, an ideal shade guide would have a range of colors such that for any person who walks into the dentist's office, a match to his or her tooth color can be found.

For the purposes of dental ceramics, color is measured by three dimensions: L (lightness), a (green-red), and b (blue-yellow), with values for typical tooth colors ranging roughly from 50 to 85 for L, -5 to 15 for a, and 10 to 45 for b. Using Euclidean distances between points in this 3-D color space, studies have found that the human eye requires a minimum separation of around 3.7 in order to be able to distinguish between two tooth colors [4]. Consequently, if the color of a crown is fewer than 3.7 units away from the patient's tooth color, it should appear as a match to human eyes.

Designing an optimal shade guide lends itself well to being modeled as a facility location problem: the facilities to locate are the shade tabs, the clients are natural teeth from dental patients, and the setting is *Lab* color space. Our goal was to design a model such that, for any given data set of color points for natural teeth, we can find a set of shade tab colors that optimally represents the color range in the data set.

In this case, the primary criterion for optimality is that every natural tooth (client) has a shade tab (facility) located within 3.7 units. In the terminology of facility location, we say that a client is **covered** if there is a facility located within a given **coverage distance**. In our case, the coverage distance is 3.7. If this condition holds, our shade guide will provide a visually acceptable color match for any tooth in the set.

Achieving complete coverage of data points might require a large number of shade tabs, which is undesirable. The best model for this situation is the maximal covering model, a standard model in facility location. The goal here is to cover as many clients as possible given a fixed number of facilities to locate. In this model, we can specify exactly how many shade tabs we want in the guide.

With a model chosen, the next step is determining a solution strategy, and for that we employ integer programming.

Integer Programming

Integer programming can best be introduced by first considering linear programming. The term "programming" here is not used in the sense that most of us are familiar with today: linear programming was developed in the early to mid 1900s, before computers became prevalent, and it is not a form of computer programming, although the two necessarily have a close relationship.

Linear programming is a way of formulating optimization problems, and the wide availability of good linear program solvers makes it an important piece of nearly any optimization project. A linear program, or LP, is expressed as follows:

$$\begin{array}{lll}\text{Minimize} & \displaystyle\sum_j c_j x_j & \\ \text{Subject to} & \displaystyle\sum_j a_{ij} x_j \geq b_i & \forall i \\ & x_j \geq 0 & \forall j\end{array}$$

The first line is the **objective function**, or goal, and specifies the linear function we want to minimize. The succeeding lines are **constraints** that must be satisfied, and they too must be linear, i.e., no nonlinear terms like x^2. The x_j are variables, and the a's, b's, and c's are given constants. The constraints in an LP specify a **feasible region**, and an optimal solution will be a point in the feasible region that has minimum objective value.

In a linear program, the variables take on real values. In an integer program (IP), also known as a mixed integer program, some or all of the variables must take on integer values. Although nonlinear integer programming is also an important field, the term "integer programming" by itself usually assumes linear programs with integer variables, and that is what we consider here. Integer programs are much more difficult to solve than linear programs because of the additional integrality constraints. However, there are many problems, including many in combinatorial optimization, that require integer variables.

Techniques that are both general-purpose and efficient have been developed for solving LPs, and there are many software packages that include good LP solvers, such as the free GNU linear programming kit, GLPK, and CPLEX, a commercial software package popular in both industry and academics. These packages also contain IP solvers, which use various strategies, including solving a series of LPs, in order to find a solution to the IP. Given enough time and memory, the solution to any IP can theoretically be found (unless the IP is unbounded or infeasible, which we do not consider here), but in practice many large IPs cannot be solved. A standard text on integer programming is Wolsey [7].

Solving the Tooth Color Problem

To solve the tooth color problem, I formulated it as an IP and coded it up using C++ and the CPLEX IP solver.

IP Formulation for an Optimal Shade Guide

As stated previously, the best model for our tooth color problem is a maximal covering model, where we try to cover as many teeth as possible with the shade tabs. The IP formulation and explanation of the terms used are shown in Figure 3. The variables, x_i and z_j, are binary variables so they are limited to taking on the values 0 and 1, and they represent a decision being made. For example, z_j is assigned the value 1 if a facility is sited at node j and 0 otherwise. Each client node i has an associated set N_i that contains all the possible facility sites (nodes) that would cover client i. Thus in our case, we must determine for each tooth color data point we have, which possible facility sites are within 3.7 units.

Term	Meaning
I	set of clients
J	set of posible facility locations
N_i	set of facility locations that would cover i
x_i	1 if i is covered, 0 otherwise
z_j	1 if a facility is positioned at j, 0 otherwise

$$\begin{array}{lll} \text{Maximize} & \sum_{i \in I} x_i & \\ \text{Subject to} & x_i \leq \sum_{j \in N_i} z_j & \forall i \in I \\ & \sum_{j \in J} z_j = p & \\ & x_i \in \{0,1\} & \forall i \in I \\ & z_j \in \{0,1\} & \forall j \in J \end{array}$$

Figure 3: The maximal covering IP formulation.

The objective in Figure 3 states that we want to maximize the client coverage. The first set of constraints, one for each client, states that a client i may only be considered covered (i.e., x_i set to 1) if at least one of the nodes in its N_i set has a facility (i.e., z_j set to 1). The second constraint states that there will be exactly p facilities. The last two lines of the formulation ensure that all x_i and z_j are binary variables.

The last implementation detail to note is the selection of the possible facility sites. We decided to make every point in the color space with integer coordinates a possible facility site. We considered including more points, such as one every 0.2 units, which would have led to more precise results, but there are two reasons not to do this: Mathematically, there would be many more nodes in the graph, possibly rendering the problem unsolvable in a reasonable amount of time. Practically, the process of mixing the ceramics to create a particular color is not exact, and the additional accuracy would not be achievable with current mixing techniques.

Results

As input data we used a set of over 500 tooth colors measured from natural teeth. Our graph had about 20,000 nodes for possible facilities and 500 client nodes. The program I wrote, which included calls to CPLEX to solve the IP, was able to solve the problems in no more than half an hour. I made multiple runs with different values of p, the number of facilities (shade tabs) to locate, and the results are shown in Figure 4.

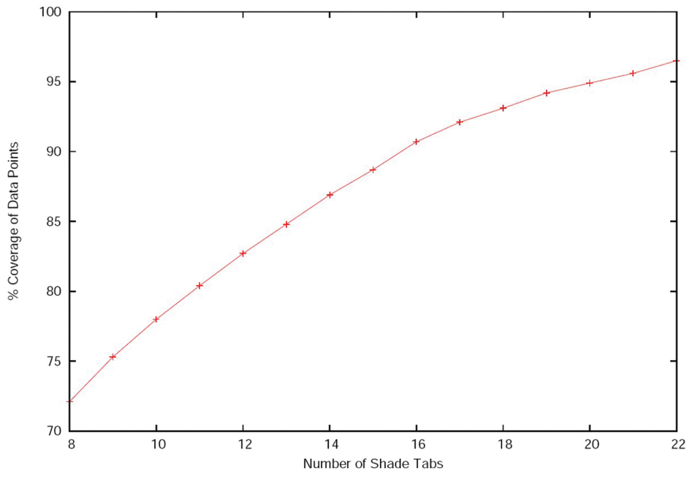

Figure 4: Percent coverage of data points for different numbers of shade tabs.

Ten to twenty shade tabs is a manageable number for a shade guide, so I ran experiments in this range. With 10 shade tabs we covered 78% of the data points, and with 20 shade tabs, we covered 95%. By comparison, one study found that three existing shade guides used by dentists covered 31%, 37%, and 56%, respectively of the teeth in the study's population [1]. Without access to their data, it is impossible to determine what percentage of the teeth could be covered using our model and the same number of shade tabs, but the results on our own data set are comparatively impressive.

The data shown here came from a specific segment of the population, but the model and the computer program can be used with any set of data to determine the optimal colors of shade tabs in a shade guide.

Evaluation

The fact that this problem was so easily solvable using a standard IP solver is not typical in combinatorial optimization. The maximal covering model is one of the simpler models to begin with, and we did not incorporate any additional constraints. In many real-world applications, as well as in many theoretical contexts, additional constraints make the problem much more difficult to solve. In the design of shade guides there are other considerations that we did not attempt to incorporate into this model. For example, it is desirable to have equal spacing (in the color spectrum) between the tabs.

When an IP cannot be solved, there are other ways to approach the problem. For instance, heuristics can be used to find a "good" feasible solution, but one that is probably not optimal, and often with no guarantee as to how close to optimal the produced solution is. Approximation algorithms are heuristics with a guarantee that any solution produced is no worse than a certain factor of the optimal.

In terms of a minimization problem, heuristics produce an upper bound on any optimal solution. One can also approach the problem from the other side and try to find a lower bound on any optimal solution. This may be done by working with the so-called relaxed version of an IP formulation, i.e., allow integer variables to take on real values. Some solution approaches work from both ends at the same time, hoping that the upper and lower bounds converge on an optimal solution.

For further reading on combinatorial optimization, see [2], or for a good book on developing mathematical models for optimization problems, see [6].

Acknowledgments

I would like to acknowledge Alexander Hassel, the professor and dentist who posed the problem; Marcus Oswald, a colleague who helped design the model; and Gerhard Reinelt, my adviser.

References

1. Analoui, M., Papkosta, E., Cochran, M. and Matis, B. 2004. Designing Visually Optimal Shade Guides. *J. Prosth. Dentistry* 92. 371-376.
2. Cook, W. J., Cunningham, W. H., Pulleyblank, W. R. and Schrijver, A. 1998. *Combinatorial Optimization*. Wiley-Interscience.
3. Daskin, M. S. 1995. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley-Interscience.
4. Johnston, W. M. and Kao, E. C. 1989. Assessment of Appearance Match by Visual Observation and Clinical Colorimetry. *J. Dental Resear.* 68, 5. 819-822.
5. Traveling salesman problem. http://www.tsp.gatech.edu/.
6. Williams, H. P. 1999. *Model Building in Mathematical Programming*. John Wiley & Sons.
7. Wolsey, L. A. 1998. *Integer Programming*. Wiley-Interscience.

Biography

Cara Cocking (cara.cocking@gmail.com) is pursuing a doctorate degree in computer science at the University of Heidelberg in Heidelberg, Germany. Her dissertation research is in the area of combinatorial optimization. Her other professional interests include programming languages, algorithms, graph theory, and computer science education. She received a BS and an MS in computer science from Virginia Tech.

Make a difference... Change the world?

Sandia National Laboratories is for people with a hunger and desire to make a difference, a passion to change the world. We're one of the top science and engineering laboratories for national security and technology innovation. Be a part of the team helping America secure a peaceful and free world through technology.

We have exciting opportunities for college graduates at the Bachelor's, Master's, and Ph.D. levels in:

- Nuclear Engineering
- Mechanical Engineering
- Electrical Engineering
- Computing Engineering
- Information Technologies
- Computer Science
- Chemistry
- Business Administration ... and more

We offer internship, co-op, and post-doctoral programs.

Sandia National Laboratories
Operated By
LOCKHEED MARTIN

Sandia is an equal opportunity employer.
We maintain a drug-free workplace.