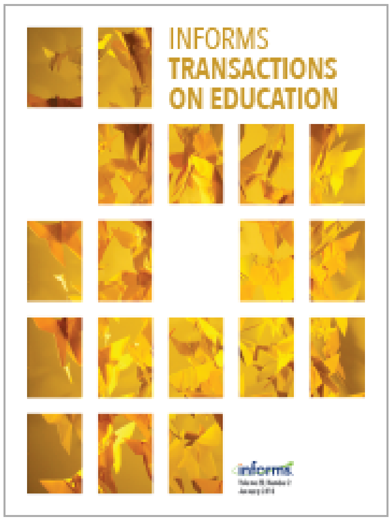


INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Teaching Note—Using the Maximum Clique Problem to Motivate Branch-and-Bound

Dawn M. Strickland,

Vol. 8, No. 2, January 2008, pp. 96–99
ISSN 1532-0545 | 08 | 0802 | 0096

INFORMS Transactions on Education

DOI 10.1287/ited.1070.0005

Teaching Note

Using the Maximum Clique Problem to Motivate Branch-and-Bound

Dawn M. Strickland
Department of Mathematics, Winthrop University, Rock Hill, South Carolina, stricklandd@winthrop.edu

In the typical OR classroom, the method of branch-and-bound is most often introduced when solving integer programs. In this note, we discuss teaching a branch-and-bound procedure motivated by the maximum clique problem rather than by a more traditional integer program. This teaching method is beneficial not only because it provides an alternate perspective on branch-and-bound for students, but also because it requires no optimization software to illustrate. The method is particularly useful in early optimization courses where students have math and engineering backgrounds. There is much literature focused on optimization students with non-technical backgrounds; we present a teaching method specifically intended to enhance the learning process for the more technical students.

Key words: integer programming, branch-and-bound, graph theory, graph color, cliques
History: Received: March 6, 2007; accepted December 18, 2007. This paper was with the author 5 months for 1 revision.

1. Introduction

The introduction of branch-and-bound in an optimization course usually occurs when integer programming (IP) solution methods are discussed. Typically, the integer constraints of the IP are relaxed and the corresponding linear programming (LP) relaxation is solved to generate upper and/or lower bounds. See Winston (2004) for an example of an undergraduate textbook addressing this topic.

It can be difficult for students in such a course to understand branch-and-bound, not because the concept itself is difficult but because the ancillary concepts themselves can be distracting. Specifically,

- Many introductory optimization students do not have a deep understanding of linear programming, so concepts like “LP relaxation” can be hard for them to grasp quickly.
- Solving LP relaxations, even in just two variables, is hard enough for introductory optimization students that they get caught up in that part of the problem, and can’t concentrate enough on the new concept of branch-and-bound.
- If the IP has more than two or three variables, it might even require the use of software to solve quickly in a classroom teaching environment. Such software isn’t always accessible in the classroom, and (as mentioned above) distracts the class from the branch-and-bound concept.

On the other hand, very small IPs with obvious LP relaxation solutions tend to be so far from realistic that students miss the relevance of branch-and-bound and the notion of why IPs are so hard to solve.

Therefore, we suggest a different method for teaching branch-and-bound. Rather than teaching it as an outgrowth of integer programming, we suggest presenting branch-and-bound in the context of the maximum clique problem. Large instances of maximum clique are easily seen to be relevant (see Pardalos 1994 for examples of applications), easy for students to understand intuitively, and easy to run through a simple branch-and-bound procedure. This allows students to focus their learning attention much more specifically on the concept at hand (branch-and-bound), so that they can internalize the ideas more quickly and easily. Also, because most undergraduate optimization students in mathematics are required to take a course in discrete mathematics prior to taking their first course in optimization, students may already be familiar with the concept of cliques and their applications.

The method we introduce is particularly useful for students with technical backgrounds. There is quite a bit of literature devoted to teaching OR to students

Figure 1 Vertices C, E, and F form a Clique of Size 3

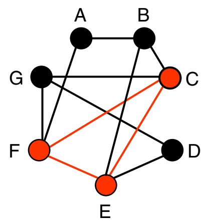
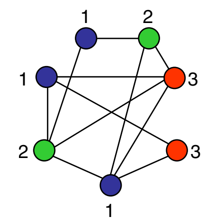


Figure 2 A Coloring of Size 3 Is Shown



in business or management science programs, particularly addressing how to approach students with less mathematical background. This note focuses on the other types of OR students—those from math and engineering backgrounds—who tend to understand the mechanics of certain topics but fail to see the broader perspective.

2. Background

First, we provide background on some basic graph theoretical concepts. For a more extensive survey of graph theory, we refer the reader to Wilson (1996) and Tucker (2002). Because branch-and-bound in the arena of pure integer programming can seem abstract to students, the utilization of simple graph algorithms to illustrate the procedure can make the material more visual and friendlier to learn. For a more traditional approach to branch-and-bound, we refer the reader to Winston (2004).

A graph $G = (V, E)$ is a set of vertices and edges. (Throughout this paper, vertices will be represented by letters and colors assigned to vertices will be represented by integers.) A clique in G is a subset of vertices $C \subseteq V$ in which every pair of vertices is joined by an edge; see Figure 1 for an example. In other words, all the vertices in a clique must be connected to each other. We want to find the largest clique in a graph; this is known as the maximum clique problem. We denote the size of a maximum clique in G by $\omega(G)$.

A nice, simple illustration of this problem is with a dinner party. Let the vertices of the graph represent dinner guests. Let an edge connect the guests if they already know each other. Then the maximum clique will be the largest set of dinner guests in which everyone knows each other.

A vertex coloring is an assignment of colors to the vertices of a graph G such that no two adjacent vertices receive the same color. A minimum coloring is a coloring that uses the fewest possible colors to color the entire graph; see Figure 2 for an example. We denote the size of a minimum coloring in G by $\chi(G)$.

Minimum colorings can also be illustrated using the dinner party example outlined above. Suppose the host would like his friends to meet each other. Then he would only seat people at tables where they do not already know the other guests. If we let different colors represent different tables at the party, then only guests with no edges connecting them (i.e., guests that do not already know each other) will be allowed the same color/table. The minimum coloring corresponds to the minimum number of tables necessary to ensure no guests are seated with people they already know.

Consider a triangle, which is a clique of size 3. Each of the three vertices must have a different color; otherwise, there would be an edge whose endpoints were the same color and the coloring would be invalid. More generally, in any coloring of a graph G, every pair of adjacent vertices must have different colors. Therefore, $\omega(G) \leq \chi(G)$. More importantly, any clique in G is a lower bound on $\omega(G)$, and any coloring of G is an upper bound on $\omega(G)$. In the next section, we give heuristics that find cliques and colorings and, hence, provide bounds to use in a branch-and-bound algorithm.

3. Computing Bounds

To find a lower bound (LB) on the size of the maximum clique, a simple greedy clique-finding heuristic can be implemented as follows on the graph $G = (V, E)$:

1. Let $T = \varnothing$.
2. Let v be the vertex in G but not in T with maximum degree. Add v to T.
3. Delete all vertices not adjacent to v from V.
4. If V is empty, stop. Otherwise, go back to step 2.

Basically, we add to the clique T the vertex with the largest degree that is connected to all the vertices already in the clique. The cardinality of the resulting set T is a clique in the graph G and is, therefore, a lower bound on $\omega(G)$, the size of the maximum clique in G. Figure 3 illustrates this heuristic.

To find an upper bound (UB) on the size of the maximum clique, a simple greedy coloring heuristic can be implemented as follows on the graph $G = (V, E)$:

1. Color vertex v_1 with color c_1.
2. For each successive vertex v_i, choose the lowest-numbered color that does not produce an invalid coloring (i.e. such that no other previously-colored vertex adjacent to v_i shares the same color).

Figure 4 illustrates the coloring heuristic on the same graph as in Figure 3.

Figure 3 An Illustration of the Greedy Clique Heuristic, Where $\omega(G) \geq 4$

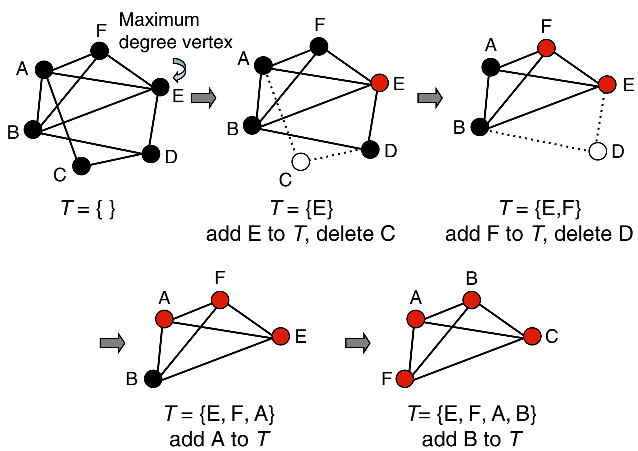
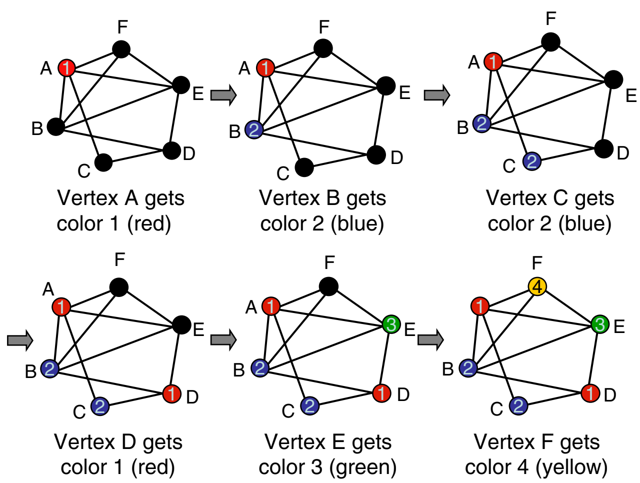


Figure 4 An Illustration of the Greedy Coloring Heuristic, Where $4 \leq \omega(G) \leq \chi(G) = 4$



The result of this heuristic is a valid coloring of G, which is an upper bound on $\omega(G)$, the size of the maximum clique in G.

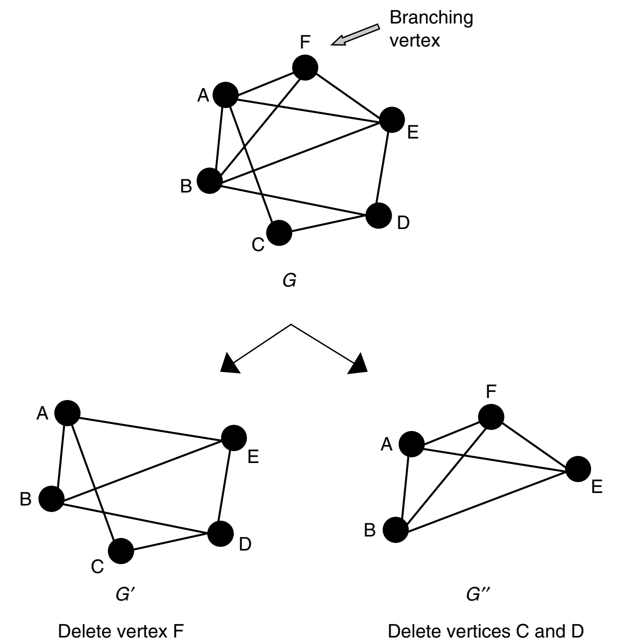
Once the graph theory basics have been covered, we can solve the maximum clique problem on any reasonably-sized graph (i.e. one that can be drawn on a blackboard) by using the method of branch-and-bound. Students can write the entire solution method in their notebooks, often on just one page. This is, of course, not the case when using traditional integer programs where each node in the branch-and-bound tree requires solving a linear program, either manually or on a computer.

4. Branching Procedure

Figure 5 illustrates the following branching procedure:

1. Given graph $G = (V, E)$, choose the branching vertex $v \in V$, where v is any vertex not connected to all the other vertices in G. If no such v exists, then G is a clique. Otherwise, go to 2.

Figure 5 An Illustration of the Branching Procedure

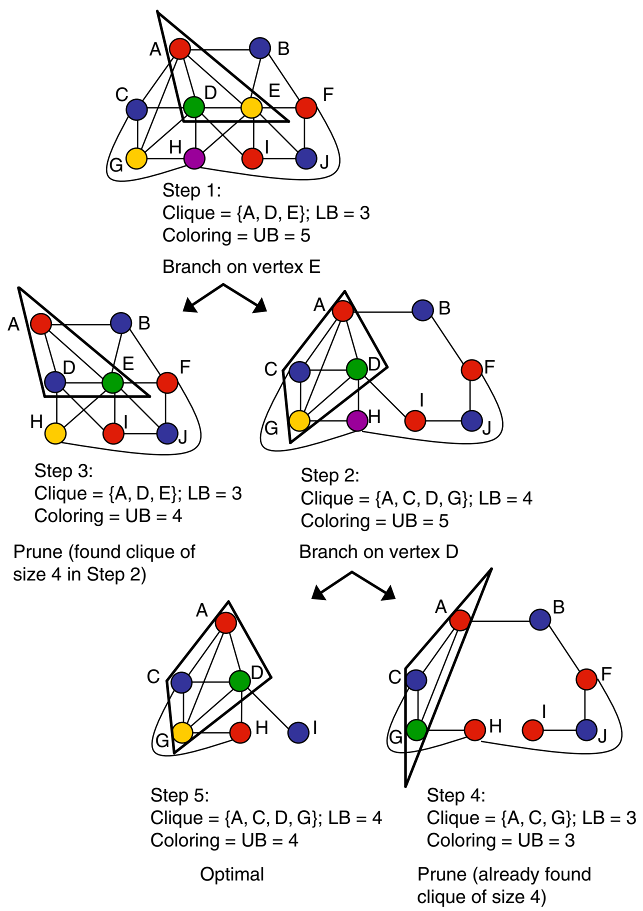


2. Create subgraphs G' and G'' from G as follows:
   - G' is the subgraph of G induced by vertices $V - \{v\}$, i.e. G' is formed by deleting the vertex v (and its adjacent edges) from G.
   - G'' is the subgraph of G induced by vertices v in $N(v)$, where $N(v)$ represents the neighbors of vertex v, i.e. G'' is formed by keeping only the vertex v and all vertices adjacent to it in G.

The maximum degree node either is or is not a member of a maximum clique. Because G' deletes only the maximum degree node from its parent graph and G'' keeps the maximum degree node and all its neighbors, the way in which the subgraphs G' and G'' are created ensures that a maximum clique in G will still exist in one or both of the subgraphs. For more detail, see Strickland (2002). Thus, maximum cliques are not destroyed in this process, but the size of the graphs at each branching vertex decreases with every iteration.

Now that the branching procedure and the computation of the bounds have been described, the entire branch-and-bound algorithm can be illustrated. Figure 6 shows the order of the branching process, the use of upper and lower bounds to prune branches, and the cliques and colorings found using the heuristics outlined above. The cliques are outlined in black, and the colorings are shown having been produced using the given heuristics. Alternate cliques can be found if different choices for the maximum degree vertices are made (in the event of ties) and alternate colorings can be found if the order of the vertices is different from that shown above. (To facilitate the

Figure 6 An Example of the Entire Branch-and-Bound Procedure



creation of photocopies, a black-and-white version of Figure 6 is given in the appendix.)

The maximum clique problem can also be formulated as a binary IP (See Bomze 1999). Instructors can present the IP as a way to relate the branch-and-bound for clique to the more typical branch-and-bound for IP.

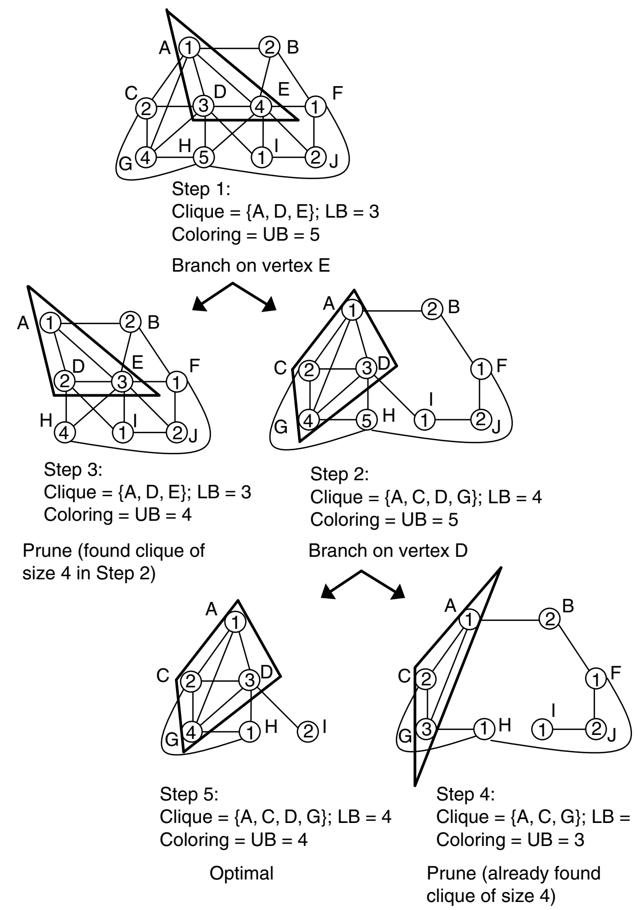
5. Conclusion

We have used this teaching approach several times in small, senior-level optimization courses. Students generally respond well to it; they appreciate the visual aspect of the clique problem as well as its applications. Branch-and-bound is usually presented in its traditional, IP-based form first, and then is supplemented by the maximum clique problem. The use of graphs also further emphasizes the fact that branch-and-bound is a technique used to solve many types of problems and not just integer programs. Because students need no software to calculate bounds at each step (and then track the bounds on a tree created in a separate document, usually by hand), they have a better sense of the problem as a whole, and they tend to understand the concept of branch-and-bound rather than simply memorizing the algorithm.

Acknowledgments

We thank Earl R. Barnes and Joel S. Sokol for giving us many of the ideas for this paper. We also thank the referees for their constructive critiques and suggestions.

Appendix



A black-and-white version of Figure 6 where letters represent vertices and numbers represent colors.

References

Bomze, I. M., M. Budinich, P. M. Pardalos, M. Pelillo. 1999. The maximum clique problem. D.-Z. Du, P. M. Pardalos, eds. *Handbook of Combinatorial Optimization*, Vol. 4. Kluwer, Boston, 1–74.

Pardalos, P. M., J. Xue. 1994. The maximum clique problem. *J. Global Optim.* **4**(3) 301–328.

Strickland, D. M. 2002. Maximum cliques with application to protein structure alignment. Ph.D. thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta.

Tucker, A. 2002. *Applied Combinatorics*. Wiley, New York.

Wilson, R. 1996. *Introduction to Graph Theory*. Pearson Education, Harlow, England.

Winston, W. 2004. *Operations Research*: *Applications and Algorithms*. Duxbury, Belmont, CA.