Slide 1

Outline

- What is an integer program?
- Examples of integer programs
- How do people solve integer programs?
  - Branch and Bound
  - Branch and Cut
  - Branch and Price

A History Lesson

The idea of using branch and bound dates back to the work of Land and Doig [LD60] and Dakin [Dak65]. The term *branch and bound* was coined by Little, Murty, Sweeney, and Karel [LMSK63].

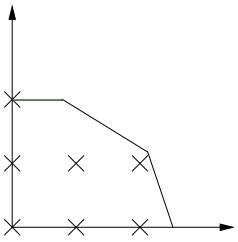
Branch and cut was a term coined by Padberg and Rinaldi in the contect of solving the travelling salesman problem [PR91], but the idea of using cutting planes was the original algorithm for solving integer programs proposed by Gomory [Gom58]. Branch and cut refers to the idea of adding cutting planes at each node of the branch and bound tree, and the term *cut and branch* refers to the addition of cutting planes at the root node of the branch and bound tree only.

Branch and price refers to the technique of combining column generation with branch and bound. I won't have time to cover it here (perhaps another dummies seminar), but for an introduction, I would suggest reading Barnhart, Johnson, Nemhauser, Savelsbergh, and Vance [BJN+98].

Slide 2

An Integer Program

$$z(S) = \max\{c^T x : x \in S\}, \qquad S = \{x \in \mathcal{Z}_+^n : Ax \leq b\}$$



$$\begin{aligned} S &= \{(x_1, x_2) \in \mathcal{Z}_+^2 : 6x_1 + x_2 \leq 15, \\ &\quad 5x_1 + 8x_2 \leq 20, x_2 \leq 2\} \\ &= \{(0,0), (0,1), (0,2), (1,0), \\ &\quad (1,1), (1,2), (2,0)\} \end{aligned}$$

Some points to consider...

- Rounding to the nearest integer *doesn't* work. This is especially true in the case where we have $0-1$ variables, which comprise the large majority of problems encountered in practice.
- $z(S) = z(\text{conv}(S))$

Slide 3

The Knapsack Problem

- A burglar has a knapsack of size b. He breaks into a store that carries a set of items N. Each item has profit c_j and size a_j.

? What items should the burglar select in order to optimize his heist?

$$x_j = \begin{cases} 1 & \text{Item } j \text{ goes in the knapsack} \\ 0 & \text{Otherwise} \end{cases}$$

$$z_{\text{HEIST}} = \max\{\sum_{j \in N} c_j x_j : \sum_{j \in N} a_j x_j \leq b,\ x_j \in \{0,1\}\ \forall j \in N\}.$$

More Knapsack Info

- The knapsack problem is NP-Complete
- It occurs as a substructure in a large number of other IP problems.
- There exist "Dynamic programming" algorithms for solving this problem in "almost" polynomial time
- The fastest algorithms (I think) are some form of "LP based" branch and bound.

Slide 4

Fall into the...

- Given m machines and n jobs, find a least cost assignment of jobs to machines not exceeding the machine capacities

$$\begin{aligned} \min z_1 &\equiv \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \sum_{j=1}^{n} a_{ij}x_{ij} &\leq b_i \quad \forall i \qquad \text{(Machine Capacity)} \\ \sum_{i=1}^{m} x_{ij} &= 1 \quad \forall j \qquad \text{(Assign all jobs)} \\ x_{ij} &\in \{0,1\} \quad \forall i,j \end{aligned}$$

Some Non-Dummy Information (Read it later)

We give this example to provide some intuition as to what might make an integer program hard to solve (or intractable). Consider the case where the machines are identical (i.e. the a_{ij} are the same for all i). Suppose one branches by setting $x_{ij} = 0$ for some fractional (i, j). Since the machines are identical, there exist (for all intents and purposes) many "identical" LP solutions with $x_{ij} = 0$ that are a simple permutation away from the current solution. One of these solutions will be the solution of the child LP. This type of symmetry makes it virtually *impossible* to improve the bound through a branch and bound procedure, and impossible to solve *this* formulation with integer programming.

Sneak preview – A different formulation can "break" this symmetry, and is solved with branch and price.

Slide 5

The Farmer's Daughter

- A travelling salesman must visit all his cities at minimum cost.

$$\min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}$$

$$\begin{aligned}
\sum_{i \in N} x_{ij} &= 1 \quad \forall j \in N \\
\sum_{j \in N} x_{ij} &= 1 \quad \forall i \in N \\
\sum_{i \in S} \sum_{j \notin S} x_{ij} &\geq 1 \quad \forall S \subseteq N, 2 \leq |S| \leq |N| - 2
\end{aligned}$$

- There are exponentially many "subtour elimination" constraints $\Rightarrow$ we *must* use branch and cut to solve this formulation of the problem

Slide 6

TSP Trivia Time!

- Historically, this was the first problem on which "branch and cut" was performed (by Dantzig, Fulkerson, and Johnson) [DFJ54].

  $$\begin{aligned}
  &10185179881672430431342228442046890805257341968329 \\
  &\quad 6812531807022467719064988166835309169868 8.
  \end{aligned}$$

? Is this...

a) The number of advance copies of *Numerical Optimization* [NW99] that were sold?

b) The number of subatomic particles in the universe?

c) The number of subtour elimination constraints when $|N| = 299$.

d) All of the above?

e) None of the above?

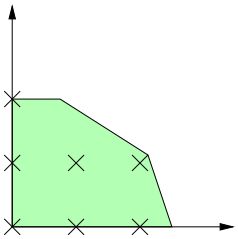
Answer Time!

- The answer is (e). (a)–(c) are all too small (as far as I know) :-). (It is (c), for $|N| = 300$).
- "Exponential" is really big.
- Yet people have solved TSP's with $|N| > 13,000$!
- Branch and cut (and price) *works!*
- (Why "and price"?) $|N| = 13,000 \Rightarrow$ the LP relaxation has more than 100 million variables, so we also need to do some column generation

Slide 7

How to Solve Integer Programs?

- Relaxations!
  - $S' \supseteq S \Rightarrow z(S) \leq z(S')$
  - $\hat{x}$ optimal with S', $\hat{x} \in S \Rightarrow \hat{x}$ optimal with S.
  - People commonly use the linear programming relaxation:



$$z(LP(S)) = \max\{c^T x : x \in LP(S)\},$$
$$LP(S) = \{x \in \mathcal{R}^n_+ : Ax \leq b\}$$

- If $LP(S) = \text{conv}(S)$, we are done.
  - We need only know $\text{conv}(S)$ in the direction of c.
  - The "closer" $LP(S)$ is to $\text{conv}(S)$ the better.

GREAT Formulations

There are a number of integer programs for which $LP(S) = \text{conv}(S)$.

- The Assignment Problem [Bir35]
- Lots of "Network" Problems [AMO93]
- Spanning Tree Problem [Edm70]
- Matching Problem [Edm65]

Slide 8

The 3 Most Important Things

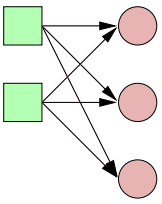
- Formulation
- Formulation
- Formulation

Ex: Uncapacitated facility location.

- ◇ Set of facilities J, Set of customers I. Which facilities should be opened in order to meet customer demand at minimum cost?

Slide 9

UFL



$$\min \sum_{j \in J} f_j x_j + \sum_{i \in I} \sum_{j \in J} f_{ij} y_{ij}$$

$$\sum_{j \in N} y_{ij} = 1 \quad \forall i \in I$$

$$\sum_{i \in I} y_{ij} \leq |I| x_j \quad \forall j \in J \qquad (1)$$

$$\text{OR } y_{ij} \leq x_j \quad \forall i \in I,\ j \in J \qquad (2)$$

? Which formulation is to be preferred?

Ex: $I = J = 40$. Costs random.

- ◇ Formulation 1. 53,121 seconds, optimal solution.
- ◇ Formulation 2. 2 seconds, optimal solution.

But Wait, there's More...

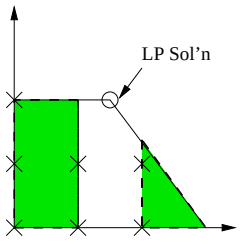
Thus endeth the slides on formulation. There are a number of automatic "preprocessing" techniques that can greatly (and automatically) improve the MIP formulation. (Including finding the "better" formulation above). I will not mention them here, but instead cite Savelsbergh [Sav94].

Slide 10

Feeling Lucky?

? What if we don't get an integer solution to the relaxation?

Ans: Branch and Bound!



- Lots of ways to divide search space. People usually...
  - ◇ Partition the search space into two pieces
  - ◇ Change bounds on the variables to do this. The LP relaxations remain easy to solve.

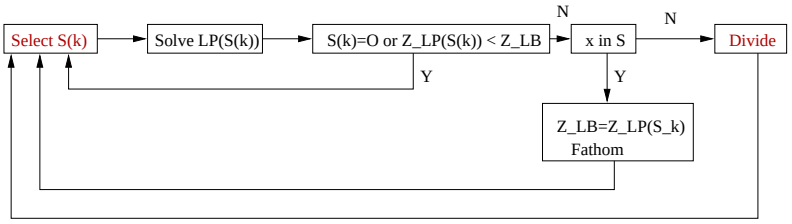
Gentlemen, Warm-Start Your Engines...

Changes made to the LP formulation affect the (primal) feasibility of the current LP solution point. However, this point remains feasible to the *dual* LP. Thus, the dual simplex method is usually used to resolve the LP's after making such small changes, resulting in speedups of up to a couple orders of magnitude in "warm-starting" as opposed to starting from scratch. See Dantzig's book for an explanation of the dual simplex method [Dan63].

Slide 11

Branch and Bound

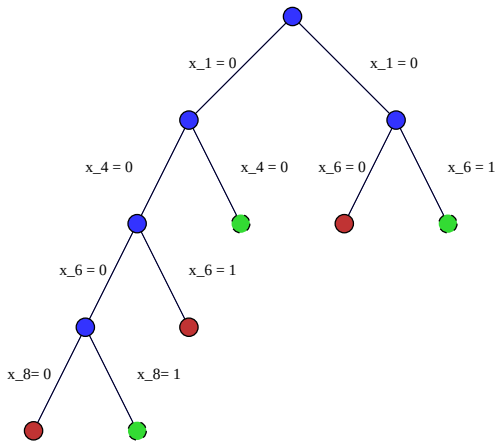
- z_{LB} – A lower bound on the optimal solution value. (The best solution you have found so far)



Slide 12

Branch and Bound Trees

- Conceptually, this selection procedure gives us a tree.



Parallel Computation

Branch and bound is a natural paradigm to map to a multi-processor computer, since exploration of the search space consists of evaluating many independent portions of the space corresponding to different sets of variables that are fixed at their bounds. Eckstein [Eck94a] [Eck94b] and Linderoth [Lin98] are a *very* small subset of the works in this field.

Slide 13

Select – I

- Depth-First – LIFO
  - \+ Relaxed problems change very little ⇒ They solve faster
  - \+ May find feasible solution faster
  - \+ Minimizes memory requirements
  - – Things can go horribly, horribly wrong!
    - ⋆ *bell5* – $> 1M$ nodes with LIFO, 23,000 with BB.
- Best Bound – Choose unsolved problem with max z_{UB}.
  - \+ Minimizes the total number of nodes evaluated
  - – Slower "resolve" times
  - – Larger memory requirements

Slide 14

Select – II

- Best Estimate – Choose unsolved problem with max z_{EST}
  - ? How are we to estimate the best solution obtainable from a node?
- Hybrid methods
  - ◇ Go depth-first "for a while", then jump to a "more promising" node.

Slide 15

Divide

- Select a variable x_j that is fractional in the LP that will change the objective functions of its children "the most".

! This is an *important* decision. Especially at the top of the tree.

- ⋆ *gt2* – $> 100,000$ nodes with "most fractional", 52 nodes with "pseudocosts".

- Pseudocosts – Like a derivative. Computed from observations:
  - ◇ $P_j^- = (z_{LP}^{i-} - z_{LP}^i)/f_j^i, \quad P_j^+ = (z_{LP}^{i+} - z_{LP}^i)/(1 - f_j^i)$
  - ◇ Values can be "conglomerated" through solution procedure

? What if I don't have any values?

- ◇ Compute them (explicitly or partially)
- ◇ Guess and pray

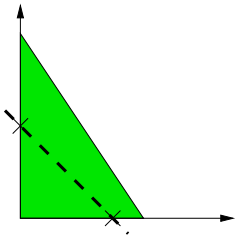
Branching Papers

For a poorly-written, drivel-filled introduction and exposition on branching and selection methods for IP, see Linderoth and Savelsbergh[LS99]. For an earlier (and better) introduction, see Forrest, Hirst, and Tomlin [FHT74].

Slide 16

Cutting Planes

- A way in which to get a closer representation of conv(S).



$$S = \{(x_1, x_2) \in \mathcal{Z}^2_+ : 3x_1 + 2x_2 \leq 4\}$$
$$\Rightarrow \quad x_1 + x_2 \leq 1.$$

- Faces of dimension $n-1$ are called *facets*, and they're good!

- An easy generalization
  - C is a *cover* if $\sum_{j \in C} a_j > b$.
  - Knapsack Cover Inequality: $\sum_{j \in C} x_j \leq |C| - 1$

A Little Facet Discussion

Facets are "good" in the sense that it is sufficient to know all the facets in order to solve the problem. It is necessary to know all the facets if we wish to solve the problem for all objective functions, but we actually need only find the n "right" facets for a given objective function.

A not well understood question is the following: Are all facets are created equal? Often during a cutting plane procedure, we can identify many known facets that cut off the current point. Which one(s) should we add?

Likely, we wish to add the "largest" faces. For the TSP polytope on 5 vertices, the facets $x_{ij} \geq 0$ subtend 80% of the central spherical angle of the polytope!!!! In general, simple facets are the best! An interesting historical perspective on this given by Kuhn [Kuh91].

Slide 17

Big Deal!

? How does knowing inequalities for the (computationally) easy knapsack problem?

Ans: Relaxations to the rescue!

- ◇ If $a^T x \leq b$ is a valid inequality for $S' \supseteq S$, then $a^T x \leq b$ is a valid inequality for S
- ◇ Many practical problems have "knapsack" rows
- ◇ The relaxation is to throw away all rows except the knapsack row.

⋆ *p2756* – 34 seconds with knapsack covers. The lifetime of the universe without them

Slide 18

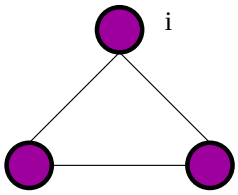
Types of Cutting Planes

- No structure required
  - ◇ Inequalities deduced from the fact that variables are required to be integer
    - ⋆ Gomory cuts
    - ⋆ Lift-and-project cuts
- (Relaxed) structure required
  - ⋆ Knapsack covers
  - ⋆ Knapsack covers with GUBS
  - ⋆ Knapsack covers with a single continuous variable
  - ⋆ Flow covers
  - ⋆ Clique inequalities

Slide 19

GC – CG Inequalities

- $x \in \mathcal{Z}^n_+,\ \sum_{j\in N} a_j x_j \le b \Rightarrow \sum_{j\in N} \lfloor a_j \rfloor x_j \le \lfloor b \rfloor$.



$$
\begin{array}{rcl}
x_i + x_j & \le & 1 \\
x_j + x_k & \le & 1 \\
x_i + x_k & \le & 1 \\
\hline
x_i + x_j + x_k & \le & 3/2 \\
\Rightarrow x_i + x_j + x_k & \le & 1
\end{array}
$$

- This inequality can also be derived from a "combinatorial" implication
- "Real" Gomory cuts are derived from the simplex tableau

Structure Independent Cuts

Chvátal [Chv73] showed that recursively combining inequalities and rounding would yield all the inequalities for S. This result appears implicitly in the earier work of Gomory [Gom58], which is why the inequalities are sometimes called CG (or GC) inequalities.

The (Chvátal)-*rank* of an inequality is the number of times in which you have to do this rounding in order to determine the inequality.

Gomory cuts were dismissed some time ago as being "computationally useless", but recently they have shown to be not so bad [BCCN99].

A more computationally useful "structure independent" class of inequalities are the "lift-and-project" inequalities [BCC93]. We could do a whole dummies seminar just on these!

Slide 20

I wanna' use cuts!

? How do I find these great valid inequalities?

Ans: Solve the *separation problem* for your particular class.

Ex: $S = \{x \in \{0,1\}^5 : 80x_1 + 45x_2 + 79x_3 + 53x_4 + 53x_5 \leq 178\}$, $\hat{x} = (0, 0, 1, 1, 46/53)$

? Does $\exists$ a cover inequality violated by $\hat{x}$

$$z_j = \begin{cases} 1 & x_j \text{ is the cover} \\ 0 & \text{Otherwise} \end{cases}$$

There exists a violated cover if

$$\sum_{j \in N} a_j z_j > b \quad \text{and} \quad \sum_{j \in N} \hat{x}_j z_j > \sum_{j \in N} z_j - 1.$$

Slide 21

Separation – II

- $\gamma = \min_{z \in \{0,1\}^{|N|}} \{\sum_{j \in N} (1 - \hat{x}_j) z_j | \sum_{j \in N} a_j z_j > b\}$
- If (and only if) $\gamma < 1$, z is the incidence vector of a violated cover

$$\begin{aligned} \gamma = \min_{z \in \{0,1\}^5} \; & z_1 + z_2 + 0z_3 + 0z_4 + 7/53 z_5 \\ & 80z_1 + 45z_2 + 79z_3 + 53z_4 + 53z_5 \quad \geq \quad 179 \end{aligned}$$

- $\gamma = 7/53$, $z = (0, 0, 1, 1, 1)$
- $x_3 + x_4 + x_5 \leq 2$ is a violated cover inequality

More on Separation

- For each class of cutting planes, a different "separation" problem must be solved. Many (most) of the separation problems are themselves NP-Complete.

Cut Type	Separation Problem
Gomory Cuts	Simple "formula"
Lift and Project	Solve a (pretty hard) LP
Knapsack Covers	Knapsack Problem
Flow Covers	INLP (knapsack)
Clique	Max weighted clique

Some Humor

"Whose dumb idea was it to solve knapsack problems in order to solve the knapsack problem" – Anton Kleywegt

- Anton uttered this while sitting in a seminar on computational IP also attended by Ellis Johnson, one of the men whose "dumb", award-winning idea this was. [CJP83]

Slide 22

It grows even weirder...

- We can solve even *more* knapsack problems to improve the cover inequality we have found.
- The process is known as "lifting", as the inequality is "lifted" into a higher dimensional space, thus improving the inequality.
- You can also think of it as "tilting".

Ex: For S, Does $\exists$ an inequality of the form $\alpha x_1 + x_3 + x_4 + x_5 \leq 2$?

- ◇ If $x_1 = 0$, α can be anything and inequality remains valid.
- ◇ If $x_1 = 1$, then $\alpha \leq 2 - x_3 - x_4 - x_5$

Slide 23

Lifting – II

- The Lifting Problem!

$$\begin{aligned} \beta_1 = \max_{x \in \{0,1\}^3} & \; x_3 + x_4 + x_5 \\ 79x_3 + 53x_4 + 53x_5 & \leq 178 - 80 \end{aligned}$$

- $\beta_1 = 1 \Rightarrow \alpha \leq 1$

$\Rightarrow$ $x_1 + x_3 + x_4 + x_5 \leq 2$ is a valid inequality for S.

- This process can be repeated for all variables in $N \setminus C$.

(up)Lifting Discussion

The concept of "lifting" has a long and distinguished heritage. Starting with Gomory [Gom69] and Padberg [Pad73]. The main lifting theorem is that if your inequality is a facet for a projection of polyhedron, then each time you lift a variable, the inequality is a facet for the higher dimensional polytope with this variable "unprojected". It is implicit in the other works, but appears explictily in Wolsey [Wol76].

A comprehensive study of strategies for solving the separation and lifting problems for knapsack cover inequalities (and their closely related kin, GUB-cover inequalities) are given in the work of Gu [Gu95] and Gu, Nemhauser, and Savelsbergh [GNS98].

I explained the concept of *sequential* lifting, where the variables are lifted one at a time. Question? In what order should the variables be lifted – different lifting orders produce different facets. This statement is true unless (this is a distinctively "non-dummy" portion of the talk) the "lifting function" is "superadditive". For the truly curious, check out the wonderful papers of Wolsey [Wol77] and Gu, Nemhauser, and Savelsbergh [GNS95].

Slide 24

Conclusions

- Integer programming cannot be explained in one hour
- Integer programming is cool!
- Cool people solve integer programs
- People learn about integer programming by reading [NW88] or [Wol98].

References

[AMO93] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows – Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[BCC93] E. Balas, S. Ceria, and G. Corneujols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[BCCN99] E. Balas, S. Ceria, G. Cornuejols, and N.R. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19, 1999.

[Bir35] G. Birkhoff. Tres observaciones sobre el algebra lineal. *Universidad Nacionale Tucumán Revista*, 5:147–151, 1935.

[BJN+98] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch and Price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[Chv73] V. Chvátal. Edmonds polytopes and a heirarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

[CJP83] H. Crowder, E. L. Johnson, and M. W. Padberg. Solving large scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.

[Dak65] R. J. Dakin. A tree search algorithm for mixed programming problems. *Computer Journal*, 8:250–255, 1965.

[Dan63] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[DFJ54] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large scale traveling salesman problem. *Operations Research*, 2:393–410, 1954.

[Eck94a] J. Eckstein. Parallel branch-and-bound algorithms for general mixed integer programming on the CM-5. *SIAM Journal on Optimization*, 4:794–814, 1994.

[Eck94b] J. Eckstein. Parallel branch-and-bound methods for mixed integer programming. *SIAM News*, 27:12–15, 1994.

[Edm65] J. Edmonds. Maximum matching and a polyhedron with 0-1 vertices. *Journal of Research of the National Bureau of Standards*, 69B:125–130, 1965.

[Edm70] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy *et al.*, editor, *Combinarotial Structures and Their Applications*, pages 69–87. Gordon and Breach, 1970.

[FHT74] J. J. H. Forrest, J. P. H. Hirst, and J. A. Tomlin. Practical solution of large scale mixed integer programming problems with UMPIRE. *Management Science*, 20:736–773, 1974.

[GNS95] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Sequence independent lifting. Technical Report TLI-95-08, Georgia Institute of Technology, 1995.

[GNS98] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Cover inequalities for 0-1 linear programs: Computation. *INFORMS Journal on Computing*, 10:427–437, 1998.

[Gom58] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Monthly*, 64:275–278, 1958.

[Gom69] R. E. Gomory. Some polyhedra related to combinatorial problems. *Linear Algebra and Its Applications*, 2:451–558, 1969.

[Gu95] Z. Gu. *Lifted Cover Inequalities for 0-1 and Mixed 0-1 Integer Programs*. PhD thesis, Georgia Institute of Technology, 1995.

[Kuh91] H. W. Kuhn. On the origin of the Hungarian method. In J. K. Lenstra, A. H. G. Rinnooy Kan, and A. Schrijver, editors, *History of Mathematical Programming: A Collection of Personal Reminiscences*, Amsterdam, 1991. Elsevier.

[LD60] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[Lin98] J. T. Linderoth. *Topics in Parallel Integer Optimization*. PhD thesis, Georgia Institute of Technology, 1998.

[LMSK63] J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Operations Research*, 21:972–989, 1963.

[LS99] J. T. Linderoth and M. W. P. Savelsbergh. A computational study of search strategies in mixed integer programming. *INFORMS Journal on Computing*, 11:173–187, 1999.

[NW88] G. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, 1988.

[NW99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[Pad73] M. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.

[PR91] M. W. Padberg and G. Rinaldi. A branch and cut algorithm for the solution of large scale traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[Sav94] M. W. P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6:445–454, 1994.

[Wol76] L. A. Wolsey. Facets and strong valid inequalities for integer programs. *Operations Research*, 24:367–372, 1976.

[Wol77] L. A. Wolsey. Valid inequalities and superadditivity for 0-1 integer programs. *Mathematics of Operations Research*, 2:66–77, 1977.

[Wol98] L. A. Wolsey. *Integer Programming*. John Wiley and Sons, New York, 1998.