

ROBERT S. GARFINKEL,

Graduate School of Management
University of Rochester

GEORGE L. NEMHAUSER,

Department of Operations Research
Cornell University

Integer Programming

A WILEY-INTERSCIENCE PUBLICATION

JOHN WILEY & SONS

NEW YORK • LONDON • SYDNEY • TORONTO

Copyright © 1972, by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

No part of this book may be reproduced by any means, nor transmitted, nor translated into a machine language without the written permission of the publisher.

Library of Congress Cataloging in Publication Data:

Garfinkel, Robert.

Integer programming.

(Series in decision and control)

Bibliography: p.

1. Integer programming. I. Nemhauser, George L., joint author. II. Title.

T57.7.G37 519.7'7 72-3881

ISBN 0-471-29195-1

Printed in the United States of America

10-9 8 7 6 5 4 3 2 1

Preface

Integer programming deals with the class of optimization problems in which some or all of the variables are required to be integer. This book is intended to be a comprehensive one on the theory, methodology, and applications of integer programming and can be used both as a text and as a reference.

Since the pioneering work of Ralph Gomory in the late 1950s, integer programming has been one of the most exciting and rapidly developing areas of operations research. Very many real-world applications of integer programming models have been developed, and algorithms have been devised for general and specific purpose models. These algorithms vary widely in spirit and mathematical sophistication. One of our objectives is to unify these approaches as far as possible and to classify them into a few generic categories so that similarities and differences may be easily seen.

The book is designed to be a text for a one- or two-semester course, depending on the sophistication of the students. Numerous exercises, ranging from simple numerical calculations to research problems, are provided. Every technique that is developed is illustrated by at least one numerical example. A comprehensive set of notes is included at the end of every chapter. These notes give an historical summary of the development of the material of each chapter. More importantly, they supply a (hopefully complete) set of

references to integer programming, including many works that are not described in the body of the text.

Chapter 1 introduces the reader to some methods of formulating and solving integer programming problems. In addition, a number of applications are presented. Chapter 2 is an introduction to those linear programming methods used in integer programming. A knowledge of some basic linear algebra is needed, and this is actually the only mathematical requirement for reading the entire text. Chapter 3 deals with integer programming and graphs. The first four sections are basic and should be included in any course. Sections 3.5–3.7 are special topics of interest. Chapter 4 on enumeration is one of the fundamental chapters and should be studied in its entirety. Chapter 5 on cutting planes is also essential. The only topics that could reasonably be omitted are the rather complex details of the proof of termination in Section 5.12 and the methods described in Sections 5.14–5.16 and 5.18. The knapsack problem, discussed in Chapter 6 is a specific integer programming model. However, the development of Chapter 7 is, to some extent, dependent on Sections 6.1 and 6.5–6.7. Section 6.12 is of importance in its own right.

Chapter 7 contains some of the most recent approaches to integer programming problems. The first six sections should be included in any course. The set covering and partitioning problems discussed in Chapter 8 are special models with a wide range of applications and relatively efficient algorithms. Theoretical and computational aspects of covering and partitioning are currently an active area for research. Chapter 9 on approximate methods is quite straightforward but very important for anyone interested in solving large problems. We would recommend including this chapter in a course involving the computational aspects of integer programming. Chapter 10 contains a variety of special topics involving nonlinearities. Chapter 11 contains computational experience. It is the only chapter written in the form of the notes; that is, it contains references to authors within the body of the text. Anyone interested in solving real-world problems is strongly advised to read this chapter.

The book, in draft form, has been used as a text for courses in integer programming at Cornell and Rochester, as well as at other universities. Our students and colleagues have been very helpful in finding errors and making suggestions. In particular, we would like to thank Egon Balas, Gerry Bennington, Mark Eisner, Mendu Rao, Don Ratliff, and David Rubin for their comments. Ivan Rosenberg, at the University of Rochester, read a number of the chapters in detail and provided many corrections. We are especially grateful to Les Trotter, at Cornell University, who read the manuscript in its entirety, and whose suggestions and corrections were invaluable. Finally, we are grateful for the efficient typing service provided by Cornell

and Rochester. In particular, special thanks are due Debbie Cagey, Kathy King, and Lorraine Ziegenfuss, all of whom did a marvelous job in typing the bulk of the manuscript.

ROBERT S. GARFINKEL
GEORGE L. NEMHAUSER

*Rochester, New York
Ithaca, New York
February 1972*

Contents

1. Introduction	1
1.1 Background	1
1.2 Defining the Linear Programming Problem	2
1.3 Methods for Solving the LP	2
1.4 Formulating Models as Linear Variables	10
1.5 Applications	13
1.6 Exercises	19
1.7 Notes	20
2. Linear Programming	24
2.1 Introduction	24
2.2 Basic Solutions, Extreme Points, and Optimal Solutions	26
2.3 Improved Basic Feasible Solutions and Optimality Conditions	30
2.4 The Simplex Algorithm	30
2.5 Finding a First Basic Feasible Solution: The Two-Phase Method	34
2.6 Degeneracy and Cycling	38
2.7 The Determinant and Inverse of the $B^{-1}A$ Matrix	41
2.8 A Modified Simplex Algorithm for Bounded Variables	43
2.9 Duality	48
2.10 The Dual Simplex Algorithm	51

Contents

I	Introduction	2
1.1	Background	2
1.2	Defining an Integer Linear Programming Problem	5
1.3	Methods for Solving the ILP	7
1.4	Formulating Models in Binary Variables	10
1.5	Applications	13
1.6	Exercises	19
1.7	Notes	20
2	Linear Programming	24
2.1	Introduction	24
2.2	Basic Solutions, Extreme Points, and Optimal Solutions	26
2.3	Improved Basic Feasible Solutions and Optimality Conditions	28
2.4	The Simplex Algorithm	30
2.5	Finding a First Basic Feasible Solution. The Two-Phase Method	36
2.6	Degeneracy and Cycling	38
2.7	The Determinant and Inverse of the Basis Matrix	41
2.8	A Modified Simplex Algorithm for Bounded Variables	43
2.9	Duality	48
2.10	The Dual Simplex Algorithm	51

2.11	Adding Variables and Constraints	54
2.12	Exercises	58
2.13	Notes	59
3	Integer Programming and Graphs	60
3.1	Introduction	60
3.2	Basic Definitions and Elementary Properties of Graphs	62
3.3	Unimodularity	66
3.4	Some Totally Unimodular LP's on Graphs	70
3.5	An Algorithm for the Shortest Path Problem	74
3.6	Matching and Covering on Graphs	77
3.7	An Algorithm for Maximum Matching	88
3.8	Exercises	101
3.9	Notes	105
4	Enumeration Methods	108
4.1	Enumeration Trees	108
4.2	The Concept of Branch and Bound	111
4.3	Branch and Bound for the ILP	114
4.4	Efficient Branching and Bounding	118
4.5	Implicit Enumeration	122
4.6	Additional Tests for Implicit Enumeration	127
4.7	Surrogate Constraints	130
4.8	Extensions to the MILP	134
4.9	Solving the MILP by Partitioning	135
4.10	Partitioning and Implicit Enumeration	143
4.11	Exercises	148
4.12	Notes	152
5	Cutting Plane Methods	154
5.1	Introduction	155
5.2	A Fundamental Cut	157
5.3	Cuts in the Method of Integer Forms	158
5.4	A Rudimentary Algorithm for the Method of Integer Forms	162
5.5	A Finite Algorithm for the Method of Integer Forms	162
5.6	Cuts with Unit Coefficients	165
5.7	Roundoff Problems and Integrality Tests	167
5.8	Dual All-Integer Cuts	167
5.9	A Finite Dual All-Integer Algorithm	170
5.10	Primal All-Integer Cuts	175
5.11	A Rudimentary Primal All-Integer Algorithm	176
5.12	A Finite Primal All-Integer Algorithm	178
5.13	A Summary of the Basic Algorithms	184
5.14	Composite Cuts in the Method of Integer Forms	185

5.15	Cuts with Different Values of h	193
5.16	Deep Cuts	197
5.17	A Cutting Plane Algorithm for the MILP	200
5.18	Intersection Cuts	203
5.19	Exercises	207
5.20	Notes	211
6	The Knapsack Problem	214
6.1	Problem Formulation	214
6.2	Recursive Equations I	216
6.3	Recursive Equations II	217
6.4	Recursive Equations III	220
6.5	A Periodic Property	223
6.6	Modulo Arithmetic and Abelian Groups	224
6.7	A Group Knapsack Problem	229
6.8	Group Recursion I	230
6.9	Group Recursion II	231
6.10	A Relation between the Knapsack and Group Knapsack Problems	236
6.11	Shortest Path Formulations of Knapsack Problems	239
6.12	Transforming an ILP in Bounded Variables to a Knapsack Problem in Bounded Variables	241
6.13	Exercises	244
6.14	Notes	248
7	Integer Programming Over Cones	250
7.1	Problem Formulation and Basic Analysis	250
7.2	Equivalent ILPC Representations	259
7.3	Group Knapsack Representation of an ILPC	263
7.4	Calculation of Smith Normal Form	267
7.5	A Sufficient Condition for an ILPC to Solve an ILP	273
7.6	Solving an ILP Using an ILPC and Branch and Bound	277
7.7	Properties of Corner Polyhedra	283
7.8	Exercises	292
7.9	Notes	295
8	The Set Covering and Partitioning Problems	298
8.1	Introduction	298
8.2	Some Applications	301
8.3	Reductions	302
8.4	Handling Binary Data	304
8.5	Some Extreme Point Properties	305
8.6	A Cutting Plane Algorithm for the Set Covering Problem	308
8.7	An Enumeration Algorithm for the Set Covering Problem	312

8.8	An Enumeration Algorithm for the Set Partitioning Problem	315
8.9	Exercises	318
8.10	Notes	321
9	Approximate Methods	324
9.1	Introduction	324
9.2	Finding Local Optima by Direct Search	326
9.3	Finding Feasible Solutions	332
9.4	Exercises	336
9.5	Notes	337
10	Integer Nonlinear Programming	340
10.1	Extensions of the Linear Techniques	340
10.2	A Lexicographic Enumeration Algorithm	344
10.3	Pseudo-Boolean Programming	348
10.4	Bottleneck Integer Programming	351
10.5	The Traveling Salesman Problem	354
10.6	Exercises	361
10.7	Notes	365
11	Computational Experience	368
11.1	Introduction	368
11.2	Test Problems	370
11.3	Computational Results. General Methods	376
11.4	Computational Results. Special Purpose Methods	383
11.5	Summary and Synthesis	387
	Bibliography	392
	Author Index	415
-	Subject Index	421

Introduction

INTEGER

PROGRAMMING

1.1. BACKGROUND

1. Introduction	1
2. Linear Programming	2
3. Simplex Method	3
4. Duality	4
5. Network Flow	5
6. Integer Programming	6
7. Branch and Bound	7
8. Cutting Plane	8
9. Heuristics	9
10. Metaheuristics	10
11. Computational Experience	11
12. Conclusions	12
13. Bibliography	13
14. Index	14
15. Appendix	15
16. Glossary	16
17. Notes	17
18. Exercises	18
19. Projects	19
20. Further Reading	20
21. Software	21
22. References	22
23. Acknowledgements	23
24. About the Author	24
25. Copyright	25
26. Disclaimer	26
27. Contact Information	27
28. Revision History	28
29. Change Log	29
30. Errata	30
31. Feedback	31
32. License	32
33. Terms of Use	33
34. Privacy Policy	34
35. Cookies	35
36. Security	36
37. Accessibility	37
38. Sustainability	38
39. Ethics	39
40. Social Responsibility	40
41. Environmental Impact	41
42. Human Rights	42
43. Labor Practices	43
44. Supplier Engagement	44
45. Community Development	45
46. Philanthropy	46
47. Corporate Governance	47
48. Risk Management	48
49. Compliance	49
50. Anti-Corruption	50
51. Anti-Money Laundering	51
52. Anti-Tax Evasion	52
53. Anti-Competitive Practices	53
54. Anti-Fraud	54
55. Anti-Insider Trading	55
56. Anti-Market Manipulation	56
57. Anti-Abuse of Market Power	57
58. Anti-Consumer Protection	58
59. Anti-Data Protection	59
60. Anti-Intellectual Property	60
61. Anti-Trade Secret	61
62. Anti-Confidentiality	62
63. Anti-Privacy	63
64. Anti-Security	64
65. Anti-Resilience	65
66. Anti-Disaster	66
67. Anti-Crisis	67
68. Anti-Recovery	68
69. Anti-Transformation	69
70. Anti-Innovation	70
71. Anti-Research	71
72. Anti-Development	72
73. Anti-Progress	73
74. Anti-Future	74
75. Anti-Past	75
76. Anti-Present	76
77. Anti-World	77
78. Anti-Reality	78
79. Anti-Ideology	79
80. Anti-Philosophy	80
81. Anti-Religion	81
82. Anti-Culture	82
83. Anti-Society	83
84. Anti-Nature	84
85. Anti-Environment	85
86. Anti-Climate	86
87. Anti-Ecosystem	87
88. Anti-Biodiversity	88
89. Anti-Conservation	89
90. Anti-Management	90
91. Anti-Strategy	91
92. Anti-Tactics	92
93. Anti-Operations	93
94. Anti-Logistics	94
95. Anti-Supply Chain	95
96. Anti-Distribution	96
97. Anti-Retail	97
98. Anti-Wholesale	98
99. Anti-Manufacturing	99
100. Anti-Production	100
101. Anti-Service	101
102. Anti-Experience	102
103. Anti-Engagement	103
104. Anti-Relationship	104
105. Anti-Community	105
106. Anti-Network	106
107. Anti-Platform	107
108. Anti-Ecosystem	108
109. Anti-Value Chain	109
110. Anti-Business Model	110
111. Anti-Innovation	111
112. Anti-Research	112
113. Anti-Development	113
114. Anti-Progress	114
115. Anti-Future	115
116. Anti-Past	116
117. Anti-Present	117
118. Anti-World	118
119. Anti-Reality	119
120. Anti-Ideology	120
121. Anti-Philosophy	121
122. Anti-Religion	122
123. Anti-Culture	123
124. Anti-Society	124
125. Anti-Nature	125
126. Anti-Environment	126
127. Anti-Climate	127
128. Anti-Ecosystem	128
129. Anti-Biodiversity	129
130. Anti-Conservation	130
131. Anti-Management	131
132. Anti-Strategy	132
133. Anti-Tactics	133
134. Anti-Operations	134
135. Anti-Logistics	135
136. Anti-Supply Chain	136
137. Anti-Distribution	137
138. Anti-Retail	138
139. Anti-Wholesale	139
140. Anti-Manufacturing	140
141. Anti-Production	141
142. Anti-Service	142
143. Anti-Experience	143
144. Anti-Engagement	144
145. Anti-Relationship	145
146. Anti-Community	146
147. Anti-Network	147
148. Anti-Platform	148
149. Anti-Ecosystem	149
150. Anti-Value Chain	150
151. Anti-Business Model	151
152. Anti-Innovation	152
153. Anti-Research	153
154. Anti-Development	154
155. Anti-Progress	155
156. Anti-Future	156
157. Anti-Past	157
158. Anti-Present	158
159. Anti-World	159
160. Anti-Reality	160
161. Anti-Ideology	161
162. Anti-Philosophy	162
163. Anti-Religion	163
164. Anti-Culture	164
165. Anti-Society	165
166. Anti-Nature	166
167. Anti-Environment	167
168. Anti-Climate	168
169. Anti-Ecosystem	169
170. Anti-Biodiversity	170
171. Anti-Conservation	171
172. Anti-Management	172
173. Anti-Strategy	173
174. Anti-Tactics	174
175. Anti-Operations	175
176. Anti-Logistics	176
177. Anti-Supply Chain	177
178. Anti-Distribution	178
179. Anti-Retail	179
180. Anti-Wholesale	180
181. Anti-Manufacturing	181
182. Anti-Production	182
183. Anti-Service	183
184. Anti-Experience	184
185. Anti-Engagement	185
186. Anti-Relationship	186
187. Anti-Community	187
188. Anti-Network	188
189. Anti-Platform	189
190. Anti-Ecosystem	190
191. Anti-Value Chain	191
192. Anti-Business Model	192
193. Anti-Innovation	193
194. Anti-Research	194
195. Anti-Development	195
196. Anti-Progress	196
197. Anti-Future	197
198. Anti-Past	198
199. Anti-Present	199
200. Anti-World	200
201. Anti-Reality	201
202. Anti-Ideology	202
203. Anti-Philosophy	203
204. Anti-Religion	204
205. Anti-Culture	205
206. Anti-Society	206
207. Anti-Nature	207
208. Anti-Environment	208
209. Anti-Climate	209
210. Anti-Ecosystem	210
211. Anti-Biodiversity	211
212. Anti-Conservation	212
213. Anti-Management	213
214. Anti-Strategy	214
215. Anti-Tactics	215
216. Anti-Operations	216
217. Anti-Logistics	217
218. Anti-Supply Chain	218
219. Anti-Distribution	219
220. Anti-Retail	220
221. Anti-Wholesale	221
222. Anti-Manufacturing	222
223. Anti-Production	223
224. Anti-Service	224
225. Anti-Experience	225
226. Anti-Engagement	226
227. Anti-Relationship	227
228. Anti-Community	228
229. Anti-Network	229
230. Anti-Platform	230
231. Anti-Ecosystem	231
232. Anti-Value Chain	232
233. Anti-Business Model	233
234. Anti-Innovation	234
235. Anti-Research	235
236. Anti-Development	236
237. Anti-Progress	237
238. Anti-Future	238
239. Anti-Past	239
240. Anti-Present	240
241. Anti-World	241
242. Anti-Reality	242
243. Anti-Ideology	243
244. Anti-Philosophy	244
245. Anti-Religion	245
246. Anti-Culture	246
247. Anti-Society	247
248. Anti-Nature	248
249. Anti-Environment	249
250. Anti-Climate	250
251. Anti-Ecosystem	251
252. Anti-Biodiversity	252
253. Anti-Conservation	253
254. Anti-Management	254
255. Anti-Strategy	255
256. Anti-Tactics	256
257. Anti-Operations	257
258. Anti-Logistics	258
259. Anti-Supply Chain	259
260. Anti-Distribution	260
261. Anti-Retail	261
262. Anti-Wholesale	262
263. Anti-Manufacturing	263
264. Anti-Production	264
265. Anti-Service	265
266. Anti-Experience	266
267. Anti-Engagement	267
268. Anti-Relationship	268
269. Anti-Community	269
270. Anti-Network	270
271. Anti-Platform	271
272. Anti-Ecosystem	272
273. Anti-Value Chain	273
274. Anti-Business Model	274
275. Anti-Innovation	275
276. Anti-Research	276
277. Anti-Development	277
278. Anti-Progress	278
279. Anti-Future	279
280. Anti-Past	280
281. Anti-Present	281
282. Anti-World	282
283. Anti-Reality	283
284. Anti-Ideology	284
285. Anti-Philosophy	285
286. Anti-Religion	286
287. Anti-Culture	287
288. Anti-Society	288
289. Anti-Nature	289
290. Anti-Environment	290
291. Anti-Climate	291
292. Anti-Ecosystem	292
293. Anti-Biodiversity	293
294. Anti-Conservation	294
295. Anti-Management	295
296. Anti-Strategy	296
297. Anti-Tactics	297
298. Anti-Operations	298
299. Anti-Logistics	299
300. Anti-Supply Chain	300
301. Anti-Distribution	301
302. Anti-Retail	302
303. Anti-Wholesale	303
304. Anti-Manufacturing	304
305. Anti-Production	305
306. Anti-Service	306
307. Anti-Experience	307
308. Anti-Engagement	308
309. Anti-Relationship	309
310. Anti-Community	310
311. Anti-Network	311
312. Anti-Platform	312
313. Anti-Ecosystem	313
314. Anti-Value Chain	314
315. Anti-Business Model	315
316. Anti-Innovation	316
317. Anti-Research	317
318. Anti-Development	318
319. Anti-Progress	319
320. Anti-Future	320
321. Anti-Past	321
322. Anti-Present	322
323. Anti-World	323
324. Anti-Reality	324
325. Anti-Ideology	325
326. Anti-Philosophy	326
327. Anti-Religion	327
328. Anti-Culture	328
329. Anti-Society	329
330. Anti-Nature	330
331. Anti-Environment	331
332. Anti-Climate	332
333. Anti-Ecosystem	333
334. Anti-Biodiversity	334
335. Anti-Conservation	335
336. Anti-Management	336
337. Anti-Strategy	337
338. Anti-Tactics	338
339. Anti-Operations	339
340. Anti-Logistics	340
341. Anti-Supply Chain	341
342. Anti-Distribution	342
343. Anti-Retail	343
344. Anti-Wholesale	344
345. Anti-Manufacturing	345
346. Anti-Production	346
347. Anti-Service	347
348. Anti-Experience	348
349. Anti-Engagement	349
350. Anti-Relationship	350
351. Anti-Community	351
352. Anti-Network	352
353. Anti-Platform	353
354. Anti-Ecosystem	354
355. Anti-Value Chain	355
356. Anti-Business Model	356
357. Anti-Innovation	357
358. Anti-Research	358
359. Anti-Development	359
360. Anti-Progress	360
361. Anti-Future	361
362. Anti-Past	362
363. Anti-Present	363
364. Anti-World	364
365. Anti-Reality	365
366. Anti-Ideology	366
367. Anti-Philosophy	367
368. Anti-Religion	368
369. Anti-Culture	369
370. Anti-Society	370
371. Anti-Nature	371
372. Anti-Environment	372
373. Anti-Climate	373
374. Anti-Ecosystem	374
375. Anti-Biodiversity	375
376. Anti-Conservation	376
377. Anti-Management	377
378. Anti-Strategy	378
379. Anti-Tactics	379
380. Anti-Operations	380
381. Anti-Logistics	381
382. Anti-Supply Chain	382
383. Anti-Distribution	383
384. Anti-Retail	384
385. Anti-Wholesale	385
386. Anti-Manufacturing	386
387. Anti-Production	387
388. Anti-Service	388
389. Anti-Experience	389
390. Anti-Engagement	390
391. Anti-Relationship	391
392. Anti-Community	392
393. Anti-Network	393
394. Anti-Platform	394
395. Anti-Ecosystem	395
396. Anti-Value Chain	396
397. Anti-Business Model	397
398. Anti-Innovation	398
399. Anti-Research	399
400. Anti-Development	400
401. Anti-Progress	401
402. Anti-Future	402
403. Anti-Past	403
404. Anti-Present	404
405. Anti-World	405
406. Anti-Reality	406
407. Anti-Ideology	407
408. Anti-Philosophy	408
409. Anti-Religion	409
410. Anti-Culture	410
411. Anti-Society	411
412. Anti-Nature	412
413. Anti-Environment	413
414. Anti-Climate	414
415. Anti-Ecosystem	415
416. Anti-Biodiversity	416
417. Anti-Conservation	417
418. Anti-Management	418
419. Anti-Strategy	419
420. Anti-Tactics	420
421. Anti-Operations	421
422. Anti-Logistics	422
423. Anti-Supply Chain	423
424. Anti-Distribution	424
425. Anti-Retail	425
426. Anti-Wholesale	426
427. Anti-Manufacturing	427
428. Anti-Production	428
429. Anti-Service	429
430. Anti-Experience	430
431. Anti-Engagement	431

Every $x \in S$ is called a *feasible solution* to (1). If there is an $x^0 \in S$ satisfying

$$\infty > f(x^0) \geq f(x) \quad \text{for all } x \in S$$

then x^0 is called an *optimal solution* to (1). The objective in a mathematical programming problem is to establish whether an optimal solution exists and then to find one, or perhaps all, optimal solutions.

Although there is no generally agreed-upon definition, an *integer programming problem* is a mathematical programming problem in which

$$S \subseteq Z^n \subseteq R^n$$

where Z^n is the set of all n -dimensional integer vectors. A *mixed integer programming problem* is a mathematical programming problem in which at least one, but not all, of the components of $x \in S$ are required to be integers.

In an applied context, it is convenient to think of (1) as a model of decision making in which S represents the set of all permissible decisions and f assigns a utility or profit to each $x \in S$. Applications of this model abound in the real world and are relevant to various branches of engineering, business, and the physical and social sciences. Some of these applications are discussed in Sections 1.4 and 1.5 and briefly throughout the book.

Much of the work done in modeling and developing solution techniques for mathematical programming problems comes under the names *operations research* and *management science*. Stimulated by scientific applications to military planning problems during World War II, operations research and management science have now become disciplines in their own right.

Linear Programming Problems

Considerable success in analyzing (1) has been achieved for the *linear programming problem* (LP), where

$$f(x) = cx \quad (2)$$

and

$$S = \{x | Ax = b, x \geq 0\} \quad (3)$$

Here, A is an $m \times n$ matrix, b is an m -vector, c is an n -vector, and 0 is an n -vector of zeros. The set S is a *convex set*, since $x, y \in S$ imply $\alpha x + (1 - \alpha)y \in S$ for all $0 \leq \alpha \leq 1$ (see Exercise 1). A convex set defined by linear constraints is called a *polyhedron* or *polytope*.

The Use of Computers

It is very important to keep in mind that the aim of much of the research in mathematical programming is to develop a theory that leads to the construction of algorithms for use on modern, high-speed digital computers. An unsophisticated reader might be attracted to simple algorithms that can be used to solve small problems by hand. Instead, the following important aspects of algorithms should be considered:

1. Is finite termination guaranteed?
2. If so, is there an upper bound on the number of computations?
3. Has computational experience been promising with respect to the speed of the algorithm?
4. Are computer storage requirements reasonable?
5. If the algorithm does not terminate in a specified time, are feasible solutions generated?

Notation

In defining the linear programming problem, we referred to x as an n -vector, without specifying whether it was $n \times 1$ or $1 \times n$. However, from (3) it is clear that x is $n \times 1$, b is $m \times 1$, and 0 is $n \times 1$, since the various matrices must be conformable for matrix multiplication. From (2) it follows

that c is $1 \times n$. In general, we will not use a transpose operation on a matrix, unless the possibility of confusion exists.

In comparing two n -dimensional vectors (n -vectors) a and b , we write $a \geq b$ to mean $a_j \geq b_j$, $j = 1, \dots, n$. If $a \geq b$, and $a_j > b_j$ for some j , we write $a > b$. Similarly, for two sets A and B , $A \subseteq B$ means that A is a subset of B , while $A \subset B$ denotes A a proper subset of B . The number of elements (*cardinality*) of a finite set A is denoted by $|A|$.

1.2. DEFINING AN INTEGER LINEAR PROGRAMMING PROBLEM

Much of this book is devoted to analyzing the *integer linear programming* problem (ILP) in which $f(x)$ is given by (2) and

$$S = \{x | Ax = b, x \geq 0 \text{ integer}\} \quad (4)$$

In more standard form the ILP is written

$$\begin{aligned} \max \quad & cx \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \text{ integer} \end{aligned} \quad (5)$$

In summation notation (5) is

$$\begin{aligned} \max \quad & \sum_{j=1}^n c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^n a_{ij} x_j = b_i, \quad i = 1, \dots, m \\ & x_j \geq 0 \text{ integer}, \quad j = 1, \dots, n \end{aligned} \quad (6)$$

All of the data in c , A , and b will always be assumed integer. This is actually equivalent to assuming the data rational, since multiplication of the objective function by any positive number, or any constraint by any nonzero number, does not change the problem (see Exercise 2).

The ILP can be written in any of a number of forms, and (5) and (6) have been arbitrarily chosen to be the ones generally used. Sometimes we will choose to minimize rather than maximize. This causes no difficulty, since for $x \in S$

$$-(\min -f(x)) = \max f(x)$$

Similarly, we could use the constraint form $\sum_{j=1}^n a_{ij} x_j \leq b_i$, or $\sum_{j=1}^n a_{ij} x_j \geq b_i$. Table 1 gives methods for converting from any of these constraint forms to any other.

Table I

	\leq	\geq	$=$
		$-\sum_j a_{ij}x_j \geq -b_i$	$\sum_j a_{ij}x_j + s_i = b_i$
			$s_i \geq 0$
From \geq	$-\sum_j a_{ij}x_j \leq -b_i$		$\sum_j a_{ij}x_j - t_i = b_i$
			$t_i \geq 0$
$=$	$\sum_j a_{ij}x_j \leq b_i$	$\sum_j a_{ij}x_j \geq b_i$	
	$-\sum_j a_{ij}x_j \leq -b_i$	$-\sum_j a_{ij}x_j \geq -b_i$	

The new variables s_i and t_i introduced in Table I are known as *slack* and *surplus* variables, respectively. Finally, if $x_j \geq 0$ is not required, an appropriate transformation is $x_j = x'_j - x''_j$, $x'_j, x''_j \geq 0$, so that x_j is unrestricted in sign.

The LP obtained by dropping the integrality constraints from the ILP (5) will be referred to as the *corresponding LP*. We also refer to the LP as a *relaxation* of the ILP. In general, the problem

$$P1: \max f(x), \quad x \in S_1$$

is said to be a relaxation of the problem

$$P2: \max f(x), \quad x \in S_2$$

if $S_1 \supseteq S_2$. Similarly, $P2$ is said to be a *restriction* of $P1$. The concepts of relaxation and restriction will be used often throughout the text. Note that if x^0 is an optimal solution to $P1$ and x^* is an optimal solution to $P2$, then $f(x^0) \geq f(x^*)$. Furthermore, if $x^0 \in S_2$, then x^0 is an optimal solution to $P2$ (see Exercise 3).

An important special case of the ILP (6) is the *binary* ILP, where $x_j \geq 0$ and integer is replaced by $x_j = 0, 1$. Although the binary ILP could be written in the form (6) by adding the constraints $x_j \leq 1$, $j = 1, \dots, n$, it will generally be written

$$\begin{aligned} \max \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & x \text{ binary} \end{aligned} \quad (7)$$

There is an important class of binary ILP's in which $a_{ij} = 0, 1$ for all i and j , and $b_i = 1$ for all i . Special methods have been developed for these so-called *covering* and *matching* problems, and they will be discussed in Chapters 3 and 8.

A generalization of the ILP is the *mixed integer linear program* (MILP), where only some of the variables are constrained to be integer. It is written

$$\begin{aligned} \max \quad & c_1x + c_2v \\ & A_1x + A_2v = b \\ & x \geq 0 \text{ integer} \\ & v \geq 0 \end{aligned} \quad (8)$$

where A_1 is $m \times n_1$ and A_2 is $m \times n_2$. If $n_1 = 0$, (8) is an LP, and if $n_2 = 0$, (8) is an ILP.

1.3. METHODS FOR SOLVING THE ILP

In this section, a brief introduction is given to two fundamental approaches for solving ILP's, *enumeration* and *cutting planes*. Since most algorithms for solving ILP's solve subproblems that are LP's, a summary of the basic methodology of linear programming will be presented in Chapter 2. From the relaxation concept introduced in Section 1.2, it follows that if the corresponding LP has an optimal solution x^0 which is an integer vector, then x^0 is a feasible solution, and thus an optimal solution, to the ILP. There is a class of ILP's (see Chapter 3) for which the corresponding LP always has an integer optimal solution. In this class of problems, the relaxation of the ILP to the corresponding LP does not change the essence of the problem.

In general, the corresponding LP does not have an optimal solution that is integer. Such is the case in the following simple example for which two solution techniques are sketched.

Example

$$\begin{aligned} \max \quad & 2x_1 + x_2 \\ & x_1 + x_2 + x_3 = 5 \\ & -x_1 + x_2 + x_4 = 0 \\ & 6x_1 + 2x_2 + x_5 = 21 \\ & x_1, \dots, x_5 \geq 0 \text{ integer} \end{aligned}$$

Since x_3 , x_4 , and x_5 are slack variables, the problem can be written

$$\begin{aligned} \max \quad & 2x_1 + x_2 \\ & x_1 + x_2 \leq 5 \\ & -x_1 + x_2 \leq 0 \\ & 6x_1 + 2x_2 \leq 21 \\ & x_1, x_2 \geq 0 \text{ integer} \end{aligned} \quad (9)$$

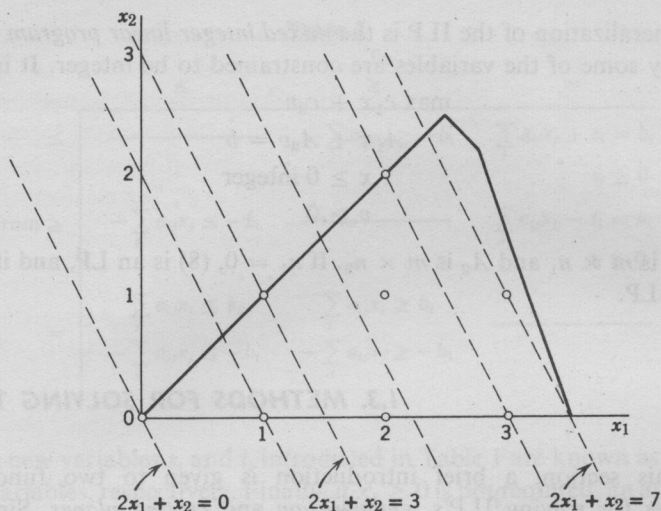


Figure 1

The feasible region of the corresponding LP is shown in Figure 1, along with a sequence of parallel lines representing constant values of the objective function. The circles represent the feasible solutions to (9). From Figure 1 it is clear that there are eight feasible solutions to (9) and that the optimal solution is $x_1 = 3$, $x_2 = 1$.

Enumeration

Without examining Figure 1, it is possible to get an upper bound on the number of feasible points. Note that the first constraint of (9), along with nonnegativity, implies $0 \leq x_1, x_2 \leq 5$. Also, the third constraint implies $0 \leq x_1 \leq 3$. Thus the constraints $0 \leq x_1 \leq 3$ and $0 \leq x_2 \leq 5$, along with the integrality requirements, limit the number of feasible points to not more than 24. For such a small problem, these 24 points could be *totally enumerated* to find that 16 are infeasible, 8 are feasible, and $(x_1, x_2) = (3, 1)$ is optimal.

Using a little more imagination, we might have added the first and second constraints to derive the valid constraint $2x_2 \leq 5$. Along with integrality, this implies $x_2 \leq 2$ and reduces the upper bound on the number of feasible points from 24 to 12.

Note that the feasible point $x_1 = 3$, $x_2 = 0$ yields an objective value of $f(x) = 6$. Thus every optimal solution to (9) must satisfy $2x_1 + x_2 \geq 6$.