シリーズ　2:　シカゴ大学ビジネス･スクール L. Schrage 教授の英知に学ぶ
ビジネスと研究に役立つ問題解決学

LINGO の雛型モデルで簡単に解決できる問題解決（上巻）

何かビジネスや研究で解決したい問題があるとき，Linus Schrage シカゴ大学ビジネス・スクール元教授が，一生かけて整理した種々の産業分野で研究開発された成果を数理計画法のサンプル（雛型）・モデルとしてまとめたものを調べ，該当するものがないか調べて，利用し問題解決を図るべきである．

もし見つかれば，数理計画法ソフトの LINGO や What's Best!でその雛型モデルを流して，概略を理解し，自分の問題に修正して解決するのが最善である．

彼は，シカゴ大学で LINDO Systems Inc.を立ち上げ，早い時期から無償の数理計画法ソフトの評価版を提供してきた．企業や大学教育で，とりあえず評価版を入手し，本書で紹介する種々の雛型（サンプル）モデルの解説を 1 週間程度で理解することを目標にすればよい．

その上で，膨大な無償で公開されている雛型モデルの中から，自分が解決したい問題を探すことを勧める．そして，それを PC で動かしてみて基本を理解すれば，実際の自分の解決したい問題に簡単に拡張し解決することが短期間で実現できる．これほど，「知的生産性」を上げるすべはないであろう．

私は，大学卒業後に「統計的判別分析は現実の問題に適用するのに問題がある」と考えた．彼の知恵に学んで，2016 年末に「新しい判別理論」を完成させ，“New Theory of Discriminant Analysis after R. Fisher (Springer, 2016)”を出版した．その応用として 30 年以上統計的に決着できなかった「癌の遺伝子解析」を解決し，Amazon から“Cancer Gene Analysis to Cancer Gene Diagnosis”を 2017 年 6 月に出版した．また，LINGO の DEA モデルを修正し，「評価の可視化の実践的な手法」を確立し，多くの組織の問題と改善策が分かった．これは既に本シリーズの第 1 巻として『DEA による問題の発見と解決』として 2017 年 10 月 7 日に出版した．読者も私に習って，簡単に解決できるものは解決し，知的生産性を上げることを勧めたい．その上で，未解決の新しいフロンティアに挑戦すべきである．

本シリーズは，Schrage 氏に対する感謝とオマージュで企画した．しかし，2007 年ごろ本書の翻訳本を日本の出版社に打診したが，600 頁弱の原著の翻訳本は，教科書需要を期待した日本の出版市場では高すぎて売れないと数社から断られ，分冊や分か

り易く書き直すなどを試みたが実現しなかった．また，判別理論の研究が忙しくなり出版を中断していた．

今回 Amazon から，学生でも購入しやすい最低価格の 2.99 ドル（337 円）で 200 頁前後の本として出すことにした．このため，原著から「問題解決」にすぐに役立たない「数理計画法」の記述の多くを下線と背景を青にして最初は読み飛ばすことや，訳注を付けるなどの修正を行った．読者は，まず雛型モデルの概略を理解することを目標にしてほしい．すなわち，LINGO のマニュアル的な説明や，数理計画法を詳しく知らないと理解できない部分は読み飛ばしていただきたい．第 2 の目標は，LINGO のスカラー（自然）表記で書かれた雛型モデルを集合表記にしたものを理解してほしい．これは，現実の大きな問題を簡単に解決し，雛型モデルを統計ソフトと同じく汎用的に使いまわしできる．第 3 の目標は，表紙の LINDO Systems Inc.の HP から無償の評価版をダウンロードし，Sample フォルダーに雛型モデルが格納されているので，それで学習の成果を確認してほしい．

LINDO 製品の 顧客は LINDO　Japan の HP に会員登録すれば，英語の最新版の原著，マニュアルなどを無料で Download できる．また各教材を教育目的で使う場合は，格安の 100 台単位のサイト契約を LINDO　Japan と契約すれば，学生にこれら資料を無償か僅かな負担で配布できるよう要望している．

数理計画法の入門としては，『Excel の雛型モデルで学ぶビジネスと教育のための問題解決』を本書の後に出版予定である．実は，こちらの方が入門的で分かりやすいので，本書より先に出版すべきであったと後悔している．初めて数理計画法を学ぶ人でも，分かりやすい雛型モデルでビジネスや教育に役立つ種々の問題を簡単に解決できる．

例えば，ノーベル経済学賞を取った「ポートフォーリオ理論」も，本書の中巻の 13 章を読めば，シカゴ大学で提案された同理論のその後の開発の歴史を，訳者のような専門家でなくても概略が理解できる．すなわち，統計や数理計画法といった学問分野は，専門書を読むよりも使いやすい統計ソフトや数理計画法ソフトで現実の問題の解決を目標にすべきである．それによって多くの人が，この分野で知的生産性を限りなく拡大し，現実の問題を解く力が簡単に手に入る．

序文

本書は，研究，ビジネス，教育などの問題を解決するために，最適化をどのように利用するかを示す．意図する読者は，ビジネスに関係する種々の問題解決を学ぶ学生や，管理者，サラリーマン，技術者などの社会人である．研究者は，Schrage 氏が 147 の文献や書籍を調査しそれを雛型モデルにしている．通常は先行研究を調査し，研究し，成果をモデルにすることが多くの研究者の目標である．しかし，その目標をクリアした雛型モデルを出発点とすれば，研究時間の節約と，より大きな目標を達成できる．

読者に必要となる技術的な能力は，未知量を表すのに記号(変数)を使用することに拒否感を感じないことと，PC を操作することを厭わないことである．

本書を読めば，重要なビジネスおよび産業の最適化問題にどのように対応すればよいかの幅広い事例が理解できる．

本書には，3 種類のタイプの章がある：

1. モデルの導入(1 章，3 章，4 章，および 19 章)．これが本書の初めの方に来るが退屈であるので，最初は読み飛ばした方が良いかもしれない．雛型モデルの理解に必要であれば，該当箇所を後で拾い読みすればよい．特に，雛型モデルを大規模な現実の問題に適用する集合表記のモデルに関する集合の理解は，最初は難しいので概略をつかんだ後で，挑戦したほうがよいと考えます．
2. PC でモデルを解決する方法(2 章，5 章)．
3. LINGO で各種のアプリケーションを解く実例(6 章－18 章)．これらの章で，漠然とした問題が，具体的かつ簡単に解決できることを実感してほしい． 例えば，13 章の「ポートフォーリオ理論」が秀逸である．投資家にならないのに，知的興味で専門書に悪戦苦闘するのは馬鹿げている．僅か 47 頁の解説で，素人の訳者にもノーベル経済学を受賞した幾つかの研究の具体的な内容が理解できた．

最適化になじみの無い読者は，少なくとも最初の 5 章は簡単に目を通す方が良い．直ぐに理解できないところは，最初は読み飛ばせばよい．最適化をよく知っているが，LINGO に不慣れな読者は，少なくとも第 2 章および第 5 章を注意して読むべきである．最適化やモデリング言語の概念をよく知っている読者は，6 章から 18 章の応用事例から初めてもよい．これらの章に強い順序関係があるわけでなく，より分かりやすいトピックが前の章にある．これらの章の中で，第 11 章(整数計画法)および第 12 章(確率的計画法)は特別な価値がある．ただし第 11 章は研究レベルのものが多く難しいが，分枝限定法と呼ばれる整数計画法のアルゴリズムの解説は必読である．これらは，かなり一般的に応用できる可能性がある先端的な技術をカバーしている（実際の問題を抱えた読者向け）．PC がより強力になるにつれ，整数計画法および確率的計画法はさらに重要になる．第 19 章は，最適化モデルの実施のための終章である．本章は

先行する章で図解されているようなモデルの細部に精通していることが要求される．

技術が発展するのに対応し，必要とされる技術は次のように変遷し，必要とされる細かい基礎知識が少なくなってきた．

1） モデルを解く専門的な能力:1950 年代

2） 最適化モデルを定式化する能力:1970 年代

3）やる気があれば，誰もができる雛形モデルを使用する能力： 1990 年代以降

この本には，最適化モデルを解くために必要な数学的な話題が少ない．しかし，新規の最適化アプリケーション開発を考える 顧客のため，最適化モデルの定式化に関して相当の情報がある．現代の即断即決を迫られる管理者のために，本書はすぐに使える雛形モデルが数多くある．

本書の多くのテーマが第 1 世代の LINDO を用いた『実践数理計画法（新村秀一・高森寛訳，朝倉書店）』のテーマを拡張している．主な違いは数理計画法ソフトの機能の拡充に伴い，適用できる問題が増え大規模な問題が容易に扱えるようになったことである．これは，**LINGO の 2 つの非常に重要な機能が貢献**している．

1）非線形モデルを解決する機能と極大値と極小値に代わって最大値と最小値を求める**大域的最適化オプションが開発**された，

2）現実レベルの**整数計画法モデルの計算時間の改善**．

3）大きなモデルを簡潔に表すための集合またはベクトル表示が可能になった点である．すなわち，統計ソフトは，分析データと統計手法が切り離され汎用統計ソフトになった．そして統計学の基盤として多くの人に利用されている．LINGO も集合表記でデータと雛型モデルが切り離され汎用的になった．また本書で紹介していないが，行列演算が開発され，経営科学と統計の基盤になるための準備が整った．

4）Excel のアドイン・ソフトの What’s　Best!は，大規模な問題もセルのコピーで対応できる．また，分析結果をそのままレポートにできる．このため，ビジネスと教育に利用できるハードルの低い数理計画法ソフトである．

注：　青字表記(あるいは章と節が下線表示のもの)は第 1 目標の問題解決に必要がないので，読み飛ばしてもよい部分です．また 5 章までは軽く読み飛ばし，直接具体的な雛型モデルの 6 章以降に時間を割いてください．

謝辞

本書は以下の人のコメントや訂正に負っている．

Egon Balas, Robert Bosch, Angel G. Coca Balta, Sergio Chayet, Richard Darst, Daniel Davidson, Robert Dell, Hamilton Emmons, Saul Gass, Tom Knowles, Milt Gutterman, Changpyo David Hong, Kipp Martin, Syam Menon, Raul Negro, David Phillips, J. Pye, Fritz Raffensperger, Rick Rosenthal,

James Schmidt, Paul Schweitzer, Shuichi Shinmura, Rob Stubbs, David Tulett, Richard Wendell, Mark Wiley, and Gene Woolsey and his students. The outstanding software expertise and sage advice of Kevin Cunningham was crucial. The production of this book (from editing and formatting to printing) was managed by Sarah Snider, Hanzade Izmit, Srinnath Tumu and Jane Rees.

訳者序文

昭和 46 年(1971)に京大の数学科を卒業し社会人になって，さてこれからの人生どうすればいいかお先真っ暗になった．中学校で奈良女出身の国語の先生から，数学ができるから岡潔の随筆を読みなさいといわれた．また，夏目漱石の高等遊民の世界にあこがれていた．現代の高等遊民は，なんとなく大学の先生だと思っていた．しかし，大学に入って吉田キャンパスを歩いていると，中田さんという大男に水泳部の説明会の教室に拉致され，気の弱い私はそのまま水泳部に入ってしまった．決して頭が悪いのでなく，水泳の疲れから数学の理解が進まず，大学院に落ちてやむなく何の準備のないまま就職したわけである（これは単にいいわけです．1 年先輩の松本カルビー会長や，他にも大学院にいき研究者になった人も多い）．

企業に入ってからは，大学生時代に習わなかった統計や OR を本で勉強した．しかし，本による勉強はどこまで行っても際限なく，達成感がなく苦しい 20 代であった．転機が訪れたのは，1977 年に SAS を日本に最初に導入し紹介し普及に貢献したことである．もともと統計も OR も実用の学問である．その頃ようやく汎用の統計ソフトが注目されてきていた．そこで，SPSS を会社に導入してもらおうと思ったが，当時ようやく全世界で 560 顧客の SAS に注目した．高価なソフトの購入を，自分の勉強のために入れてもらうのは気が引ける．そこで代理店になってしまえば，無料で SAS を使い統計が勉強できると考えたわけだ．しかし，日本文化礼賛で有名なビル・トッテン氏に代理店を取られたので，やむなく計算センターに SAS を入れて統計処理の受託計算を業務にした．そして，SAS を先生に統計を独学することにした．その後，1984 年に OR の中でも自分に最も興味のある数理計画法の LINDO をシカゴ大学ビジネス・スクールの Schrage 教授を訪問し総代理店になった．もちろん，代理店になって LINDO を通して数理計画法を勉強しようというわけである．

現在日本の大学でも，統計ソフトを使った実践的なデータ解析の教育が普及してきている．一方，私の力足らずもあるが，数理計画法の実践的な教育は一部の大学に限定されている．未だに限定された専門家の世界である．私はこれを打破し，ビジネスマンや文科系の学生でも，気楽に使ってほしいと願っている．

LINDO はシカゴ大学ビジネス・スクールの Linus Schrage 教授が開発し，その後 LINDO Systems Inc.で開発サポートされている．数理計画法による数学モデルを数式通りに LINDO に入力できる，いわゆる自然表記（スカラー表記）のソフトウエアである．この世界では，長らく IBM の MPSX という数理計画法ソフト（数理計画法の分野ではソルバーと言っている）が世界の定番であった．そして今でも，摩訶不思議な MPSX 形式で記述されたモデルが幅を利かせている．いわゆるレガシー(世界遺産)である．このため世界の大学教育で，分かりやすさから LINDO は教育用のソフトとして受け入れられてきたが，産業界では実用的でないとの誤解も生んだ．以前は，LINDO を産業システムに組み

込むにはCommand形式のBatch処理で行ってきた．私自身の経験でも，某製鉄会社の24時間原料ヤードの操業支援システムで，LINDOのモデルをCで10分間隔で生成し，LINDOで整数計画法を解く，解をCで加工し表示，という繰り返しを行うシステムを開発し納品したことがある．

しかし，その後LINDO社は，

１）What's Best! （WB!）というExcelのアドインソフト，

２）数理計画法モデルを集合とベクトルで生成する最適化言語を持ったLINGO，

３）これらのソルバーの開発に用いられている C ライブラリーを開発用に提供するLINDO API（LINDOの名前は，単に歴史にLINDOの名前を残すための命名）という最適化のCライブラリー，

４）上記3種のソフトをバンドルし，教育用に廉価に提供する目的のSolver Suiteや格安の100台単位のサイト・ライセンス，

５）各製品の評価のための無償のデモ版と，膨大な LINGO と WB!のサンプル・ライブラリーを提供している．

LINDOやMPSXであれば，モデルの規模が大きくなるに連れ，何万行のモデルに関する情報を入力する必要がある．そして係数の値が変わるたびに，煩雑な作業を繰り返す必要がある．LINGOでは，数理計画法作成言語で，モデルのサイズにかかわらず，数10行程度のモデル記述で多くの問題は解決できる．これを使いこなすことで，日本の教育や研究水準が桁違いに向上することを願っている．

一方，アメリカでは初心者に対してWB!による教育が行われ，テキストも多い．これは多くのPCにExcelが導入され，情報基盤になっているからであろう．Excelと線形計画法の概略を知っておれば，確かにWB!は便利である．私も当初は，整数計画法を用いた判別関数の研究でWB!を用いていた．Excelのセルや範囲のコピー機能を使えば，4〜5万個程度の制約式の作成も1分ほどマウスを握りつづければよいだけだ．

さらに統計に比べて出遅れた数理計画法を，できるだけ廉価に提供したいというLinus Schrage教授の信念も貴重である．ビジネス・スクールで教鞭をとりながら，ビジネス的に儲けようという意欲がないわけである．競合製品に比べ価格が格段に安い上に，これらをバンドルしたSolver Suiteやサイト・ライセンスを破格の値段で提供している．さらに禁じられた裏技がある．本来個人の評価用に公開されている無償のデモ版を教育や研究に使い続けている教員も多い（格安の100台単位のサイト契約に切り替えて，学生が自宅で自習用にデモ版を使うよう指導してください）．

統計ソフトが昔に比べ値段が安くなってきているが，それ以上に安い数理計画法による教育が日本ではまだ立ち上がっていない．1980年代から，実務家から「ORが役に立たない」という指摘がされている．それは，理論を現実問題に適用する手段，すなわちソフトウエアがなかったためである．しかし，現実にその手段が破格の値段で目の前にあるのに，なぜ統計に比べ普及していないのであろうか．単にこの分野の教育者が，理

論の蛸壺に安住し，惰眠をむさぼっていなければ良いのだが．私自身，大学教員を退任し障害の研究が完成したので統計ソフトと同じようにこれまで以上に紹介と普及に努めたい．

一方，産業応用を考えると，LINGO や WB!は**線形計画法(LP**:Linear Programming），**整数計画法(IP**: Integer Programming)，**非線形計画法(NLP**: Non Linear Programming)，**確率計画法(SP**: Stochastic Programming)と大域的最適解の探索(Global Option)など，数理計画法のすべてに対応して，ユーザーが各計画法を指定することなく利用できる．しかも，LINGO や WB!で開発したい最適化システムのモデルをプロトタイピングする．そして開発の概略を決めた後，最適化の C ライブラリーの LINDO API で開発すれば，開発工数の短縮を実現し開発の失敗を避けることができる．また，数十年前であるが，ある石油会社では IBM の汎用機上で動かされていたシステムを，PC 上の WB!に移植し，膨大な費用の削減に成功した．費用以上の更なる効果は，難しい技術が，大型コンピュータ室から，事務員の PC 上で日の目を見るようになった点である．

企業で SAS や LINDO 製品を紹介普及するにあたって，SAS による統計システムの開発部隊は作ったが，LINDO 製品は販売主体にしてきた．筆者が大学教員になってしばらくして，元の企業から LINDO 製品は技術対応できないので代理店を探してほしいと依頼があった．筆者と住商情報システム在職中に VAX/SAS を製薬メーカ 32 社に販売した市川さんが，**㈲マイス内**を立ち上げていた．それで彼に LINDO Japan (HP:http://lindo.jp, E-mail:sales@lindo.jp, 電話：03-5748-8730, FAX：03-5748-8732) の業務を行ってくれることを依頼し，快く引き受けてくれたことに感謝したい．

このおかげで，私は研究で多大な協力を Linus Schrage 氏と Kevin Cunningham 氏から受け，自分の一生の研究である「判別分析の新理論」を完成させ，その応用研究として「癌の遺伝子解析と診断」に世界で初めて成功し貢献することができた．

本書出版を試みた　吉祥寺　2007 秋

退官した　柏　2017 年夏

目次

上巻

8.4 プロジェクトのクラッシング（短縮）
8.4.1 クラッシングのための費用と価値
8.4.2 活動のクラッシングの費用
8.4.3 プロジェクトの短縮による価値
8.4.4 短縮問題の定式化
8.5 プロジェクトのリソース制約
8.6 パスによる定式化
8.6.1 例題
8.7 方向性のないネットワーク
8.7.1 例題
8.8 複式簿記：会社のネットワークモデル
8.9 ネットワーク型 LP モデルの拡張
8.9.1 Multicommodity ネットワーク・フロー
8.9.2 Multicommodity 問題のサイズを減らす
8.9.3 Multicommodity フロー例
8.9.4 フリート・ルーティングと割当て
8.9.5 フリートの割当て
8.9.6 レオンチェフ・フローモデル
8.9.7 活動/資源図
8.9.8 スパニングツリー
8.9.9 スタイナー木
8.10 非線形ネットワーク

中巻

第 9 章　多期間計画問題

9.1　はじめに
9.2　動的生産計画問題
9.2.1　定式化
9.2.2　制約と解
9.2.3　絶対値の表現
9.3　多期財務計画モデル
9.3.1　キャシュフロー
9.4　税金を考慮した財務計画モデル
9.4.1　定式化と解
9.4.2　双対価格の解釈
9.5　現在価値対 LP の分析

下巻

12 不確実あるいは確立要素のある多期間決定問題

第1章　数理計画法とは

1.1　はじめに

数理計画法（制約つきの最適化）は，希少資源の最適割り当てを決める数学的方法である．線形計画法（LP）は，**広告**から**生産計画**まで，**ビジネスのほとんど全ての面で実用的な適用**がみられる．**輸送**，**分配**，**多期間の生産計画問題等**は，LP の最も典型的な対象分野である．石油産業は，代表的な LP 顧客である．ある大きな石油会社の情報処理の責任者が，同社における PC 使用時間の 5%から 10%が LP や LP に似たモデルの処理に使われていると発表している．

読者は，LP の「プログラミング」と PC プログラムの「プログラミング」は，内容が異なることを知ることが重要である．前者の場合は「計画」を意味し，後者の場合は「計算」などを遂行するための命令を書くことを意味する．一方のプログラミングの訓練は，他方とは直接には関係がない．

多くの最適問題では，2 つの分類が重要になる．最初は，土地，プラントの能力，販売サイズのように限られた資源であり，2 番目は，「低炭素鋼の生産」，「ステンレス鋼の生産」，「高炭素鋼の生産」のようなアクティビティ（活動）である．各活動は，追加の総資源を消費したり寄与したりする．問題は，実際に利用可能な資源以上を使用しないで，最も良い活動水準の組合せを決定することである．私たちは次の簡単な例を考えることで，LP を最も良く理解できる．

1.2　簡単な製品混合問題

Enginola Television 社 (ET 社) は，2 種類のテレビ Astro と Cosmo を生産している．各製品に対して 1 本の生産ラインがある．Astro の生産ラインの生産能力は，1 日当り 60 台である．一方，Cosmo は一日当り 50 台である．また，Astro は 1 台の生産に 1 人時の労働力を必要とする．一方，Cosmo は 2 人時が必要である．2 種類のテレビの生産に割り当てることのできる 1 日当りの総労働力は，合計で最大 120 人時である．もし利益貢献が Astro と Cosmo で 20 ドルと 30 ドルとすれば，1 日の生産をどの様にすべきだろうか．

これを言葉で記述すれば次のようになる．

最大化　　**（利益）**

制約条件　**（Astro の生産台数）≦（Astro の生産能力）**

（Cosmo の生産台数）≦（Cosmo の生産能力）

（実労働人時）≦（利用可能労働人時）

これを定式化するために，次の簡単な変数（数理計画法では**決定変数**という）A と C を定義すると便利である．

A＝1 日当りに生産すべき Astro の台数

B＝1 日当りに生産すべき Cosmo の台数

標準形と呼ばれている数学表現（LINGO の**スカラー形式**あるいは**自然表記**）で，この問題を定式化すると次のようになる．括弧で示した説明が重要である．「≦や≧」は，等号を省いた「＜や＞」あるいは「<=や>=」で表される．掛け算と割り算は，コンピュータの世界では「*と/」で表す．

```
MAX = 20*A + 30*C ;          (ドル)
      A              <=  60 ; (Astro の生産能力)
C  <=  50 ; (Cosmo の生産能力)
A  +  2*C  <=  120 ; (労働時間)
```

最初の行「MAX = 20*A＋30*C;」は，最大化する「**目的関数**」である．一方，残りの 3 行は「**制約式**」と呼ばれている．最適化プログラムの多くは，全ての変数が非負と仮定している．従って，制約式 A≧ 0，C≧ 0 は必要としない．

資源と活動には，次のような 3 つの資源がある：Astro の生産能力，Cosmo の生産能力，そして労働力である．2 つの活動は，Astro と Cosmo の生産台数である．最適化における各制約式で，幾つかの資源を結合することができる．一方，各決定変数に対して，それに対応する活動がある．各「活動」は，それに対応する「決定係数」で表される．

1.2.1　グラフによる解析

ET 社問題は，**図 1.1 の**グラフで示せる．

実行可能な生産の組合せは，5 本の実線によって囲まれた五角形の範囲内にある点である．これを実行可能解という．数学では，定義域を表す．

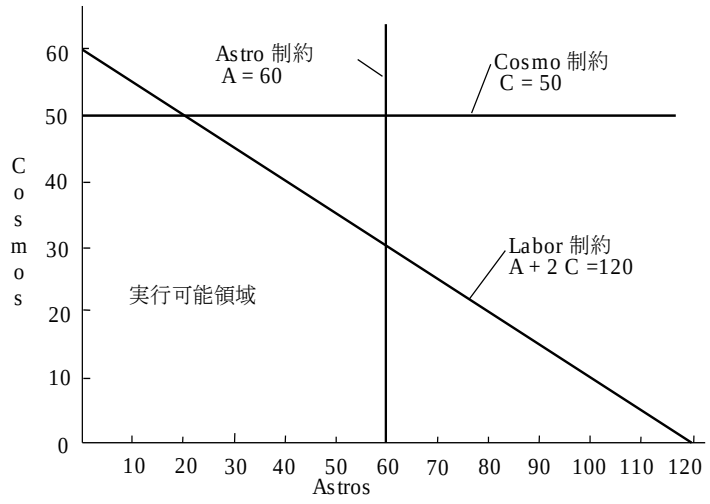


図 1.1　ET 社の実行可能領域

ここで一番高い利益を与える五角形の内点を探したい．どこに最大利益の点があるかを知るため，幾つかの方法がある．A = C = 0 の点は実行可能解だが利益は 0 である．私たちが Cosmo 生産ラインの責任者と話すと，「Cosmo は我が社の利益の大きい製品なので，できるだけ多く（50 台）作って，30*50 = 1500 ドルの利益貢献を達成すべきで

す」と答えるであろう．思慮深い読者は，A =0，C= 50 以外に，1500 ドルの利益を達成する A と C の多くの組合せがあることに気づくであろう．そして，20A + 30C = 1500 の線を図 1.2 のようにグラフに加えると，点線で表される区間上のどの点も 1500 ドルの利益になる．このような一定の利益を表す直線を，等利益直線（または費用最小化問題の場合，等費用直線）という．次に Astro 生産ラインの責任者と話せば，彼はこういうかもしれない：「Cosmo を 50 台生産しても，まだ 20 台の Astro を生産できる労働力が余っている．これは 1900 ドル(= 30*50 + 20*20)の利益を生みます．なぜ，我々は働くのをやめなければいけないのでしょうか？」

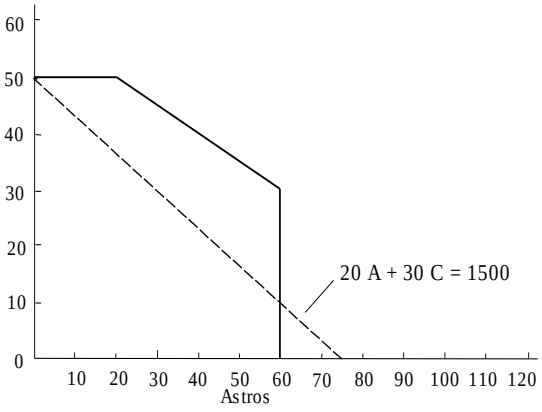


図 1.2　「利益 = 1500」の ET 社問題

注意深い読者は，1900 ドルの利益を得る多くの方法があることに気づく．20A + 30C = 1900 を表す線分を図 1.3 のようにグラフに加えると，この線分上のどの点も 1900 ドルの利益になる．

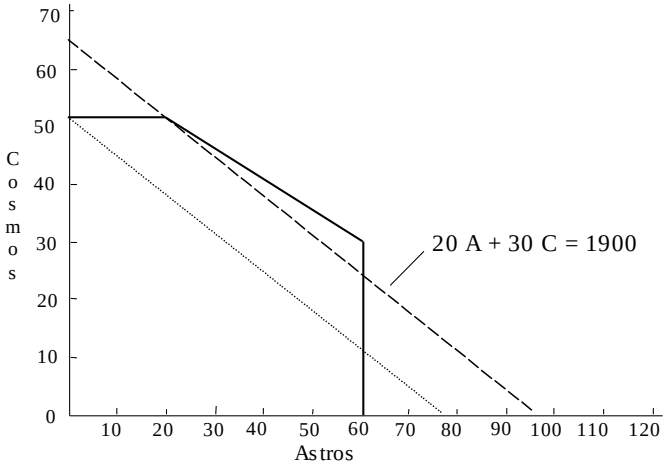


図 1.3 「利益 = 1900」の ET 社問題

注意深い読者は，さらに次のように考えるだろう．私たちが利益を増加させたいのなら，直線上に実行可能解があるぎりぎりまで等利益直線を平行移動させればよい．この最良の実現可能な点は A = 60，C = 30 です．それは**図 1.4** に示す 20A + 30C = 2100 の

線上にある．たとえ Cosmo の 1 単位当たりの利益貢献度が高くても，Cosmo を 50 台作るより 30 台作るほうが最適解になる．この小さな問題の図による解法は，私たちがより大きな問題を分析する場合に何が起こっているか理解するのに役立つ.

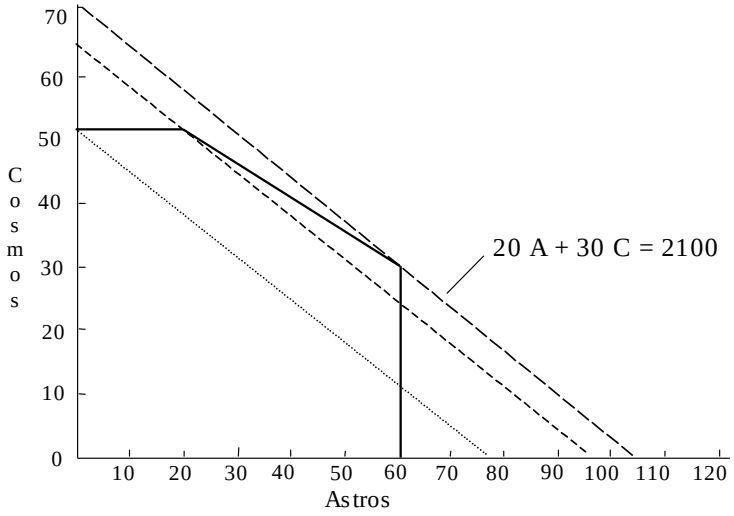


図 1.4　「利益 = 2100」の ET 社問題

1.3　線形性

線形計画法は，私たちが関わることのできる活動の効果が，線形である場合に適用できる．実際問題で，線形要求は次の 3 つで構成されている.

①**比例性**：1 つの変数とそれ自身の活動の効果は，比例的である．例えば，鉄鋼の生産量を 2 倍にすれば，鉄鋼生産の現在の水準に関わらず，販売される鉄鋼の売り上げや鉄鋼を生産するための電気消費量の総計は 2 倍になる.

②**加法性**：変数間の交互作用は加法的でなければならない．例えば，販売高は鉄鋼の販売高，アルミニウムの販売高等の合計である．一方，電気消費量の総計は，鉄鋼，アルミニウム等を生産するために使用された電気量の合計である.

③**連続性**：変数は，連続的でなければならない．すなわち，決定変数に対して 6.38 のような端数の値が許される．これが許されない問題は，整数計画法(IP)で扱う.

「販売価格」と「販売数量」という 2 つの決定変数が含まれているモデルは，おそらく線形ではないだろう.比例要求は満足するが,2 つの変数間に交互作用がある場合は，加法的でなく乗法的である．すなわち，販売高＝販売価格×販売数量であり，販売価格＋販売数量ではない．もし供給者が数量割引を行うと，比例制約を満たさなくなり，発注費用は発注量に比例したものより安くなる.

「建築すべきフロア数」という決定変数が含まれているモデルは，比例，加法要求を満足するかもしれない．しかし，最後の条件は破られる．6.38 階を建てるという決定は，段違い設計に優れた設計者がいなければ実行できない．それにもかかわらず，LP 解は，その様な端数の解を出力するかもしれない.

LP が適用できる問題は，実際にはこの例に示したものより，より一般的である．目的関数は，最大化の場合だけでなく最小化の場合もある．制約式の方向は ≦ の他に ≧ や等しい(=)こともある．幾つかの，または全てのパラメータは，正ではなくて負のこともある．分析できる問題の主な制限は，次で述べる線形制約から生じる．しかし，後で述べるように**整数計画法**や**2 次計画法(QP: Quadratic Programming)**では，これらの線形性は必ずしも満足させる必要がないことが分かる．

図 1.5 は，非線形関数の例を示す．たとえば X*Y は比例的ではあるが加法的ではない．X^2+Y^2 は加法的であっても比例的ではない．これらを含むモデルは，非線形計画法(NLP)で使える．

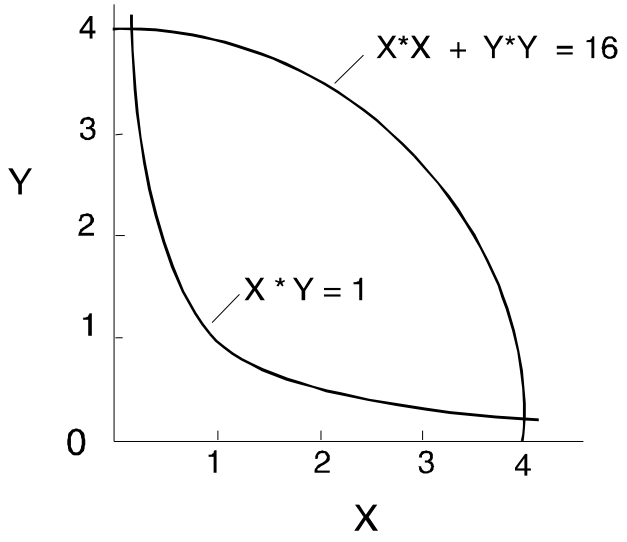


図 1.5　非線形関係

1.4　LP 解の分析

PC で線形計画法を解くと，可能な結果として**図 1.6** のどれかになる．

線形計画法が適切に定式化されていれば，図 1.6 の一番左側の道をたどる．解を求める過程では，最初に 1 つの「可能解」を求めることを試みる．すなわち，「**実行可能解**」とは，全ての制約を同時に満足するが，必ずしも目的関数を最大化していないものをいう．もし，定式化であまりにも厳しい要求をつけた場合，上図の一番右側の「**可能解なし**」の道をたどることがある．その場合には，2 つあるいはそれ以上の制約があって，それらを同時に満たすことが不可能な場合である．例えば，X≦2 を満たし，かつ X≧3 を満たす，というような制約である．従って，「可能解が存在しない」ということは，目的関数で決まることではなく，制約そのもので決まる．実際には，可能解なしという結果は，大きな複雑な問題で起こりうる．例えば，生産活動の利用に上限が設けられ，しかも非現実的なほど高い需要がある場合などである．従って，PC から可能解なしというメッセージがでたときは，言いかえれば「ケーキを持っているという状態と，それ

を食べたという状態は，同時には成立しない」ということを告げているのに等しいと考えるべきである．もし，可能解が見つかれば，次の手続きは最適解を見つけることである．もし，「**非有界**」という事態になれば，その定式化は制限なしの利益が得られることを認めるような，非現実的な定式化を意味している．この場合は，重要な制約式を忘れているか，モデルの入力ミスであることが多い．

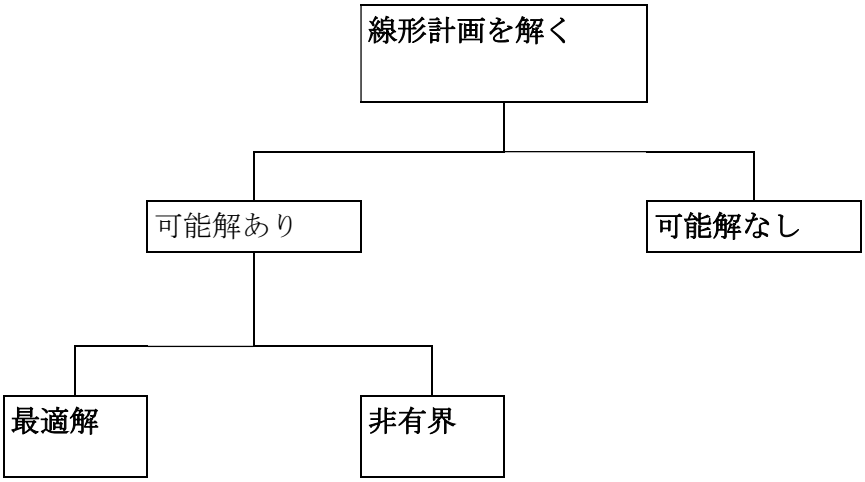


図 1.6　解の結果

ET 社問題を解くと，次のような解が得られる．

```
Feasible solution found at step:  1
Objective value:                2100. 000
Variable           Value        Reduced Cost
       A         60. 00000          0. 0000000
       C         30. 00000          0. 0000000
     Row    Slack or Surplus      Dual Price
       1         2100. 000           1. 00000
       2          0. 0000000         5. 00000
       3         20. 00000           0. 00000
       4          0. 0000000        15. 00000
```

訳注：「Feasible solution found at step: 1」は，単体法で僅か 1step で結果が出たことを示す．複雑で大規模な問題では，100 万 step 以上ということはざらである．昔の数理計画法の授業は,LP 計算の「**単体(Simplex)法**」と呼ばれる計算方法を教えることが大きな比重を占めた．しかし PC で 1 秒もかからないことを教わっても役に立たない．この時代の学生は社会に出て「数理計画法」に拒絶反応を示す人が多く，自分の仕事で役立てることができなかった．また OR 学会の先輩の近畿大学の権藤元教授に無償で LINDO を授業に貸し出した．その見返りに，学生が

テキストの問題を片っ端から LINDO で解いた報告書をいただいた．有名なテキストの問題も答えが間違っているという学生のコメントが思い出される．「Objective value」は目的関数の値で，2100 の利益が得られたことを示す．

上記の出力は，「**変数（VARIABLE）**」と「**行（ROW)**」の 2 つの部分から成り立っている．「変数（VARIABLE)」の部分の最初の 2 行は，決定変数の Astro（A）を 60 単位，Cosmo（C）を 30 単位使用することで，最大利益として 2,100 ドルを達成できる．

訳注:このように決定変数が 0 でない場合は，**減少費用(REDUCED COST)**は 0 になる．もし決定変数が 0 になれば，それを無理やり 1 単位生産すると必ず目的関数の値が悪くなるが，減少費用はその値を示す．もし仮に A = 0 で減少費用が 100 であれば，A を 1 単位生産すると利益は 100 ドル，0.1 単位生産すると 10 ドルという割合で利益を減らすことを示す．

「行（ROW)の部分」の 1 行は，目的関数の情報であるので無視してもよい．この最適解では，第 2 行は最初の A≦60 という制約に対応している．A = 60 であるので代入すると 60≦60 になり，右辺定数項の資源制約の 60 から最適解で求まった A = 60 を引いた値を**スラック変数**という．逆に A≧60 の場合，(A-60)をサープラス変数という．この値が 0 ということは，この制約条件の上限いっぱい資源を使ったことを示す．この場合，制約条件を緩めて 1 単位増やすと双対価格の 5 ドルだけ利益が増える．**図 1.8** の A ≦ 60 を A ≦ 61 に変更して絵を描いて LINGO で確認できる．C ≦ 50 という第 3 行の制約式では，スラック変数が 20 となる．すなわち 20 だけ余裕があるので，C ≦ 50 を C ≦ 51 に増やしても，在庫が増えるだけで利益に貢献しない．また，第 4 行の A + 2C ≦ 120 という制約ではスラック変数はゼロとなっているので，60 + 2*30 = 120 と等式になり，制約条件を A + 2C ≦ 121 に増やしても利益の増大にはならないことに注意しよう．

減少費用(REDUCED COST)と双対価格(DUAL PRICES)は，次でも説明するが，すでに簡略した説明したので最初に読むときは 1.5 を読み飛ばしてもかまわない．

1.5 感度分析：減少費用と双対価格

現実的な線形計画法の問題では，非常に多くのデータが必要とされる．正確なデータを収集するのは非常に高価なため，実際には確信を欠くようなデータを使うことが多い．これに関連して，情報処理の世界で「ゴミを入れれば，ゴミが出てくる(garbage in, garbage out.)」という言葉がある．現実問題を分析する際に重要な事は，入力するデータが変わったとき，そのモデルの特性がどう変わるか，ということである．感度分析は，この種の問題に答えるために利用できる．幸いなことに，線形計画法を解いた結果

の出力には，この感度分析に役立つ補足的な情報が提供されている．すなわち，**減少費用**および**双対価格**という 2 つの標題のもとに示される情報がそれである．

また感度分析は，どの種の情報が注意して推定するかを明らかにしてくれる．例えば，ある種の製品があまり利益をもたらさないことが明らかな場合，その製品の費用を推定するのに時間をかける必要はない．従って，モデル構築で重要な点は，「あるパラメータ推定のエラーが最終的な意思決定にほとんど影響を及ぼさない場合には，そのパラメータの正確な推定値を得るために時間を費やすべきではない」ということである．

1.5.1　減少費用

線形計画法の解の出力は，変数ごとに減少費用が出力される．目的関数の単位がドルであって，ある変数の単位がガロンであれば，その変数の減少費用はドル／ガロンとなる．最適解が得られたとき，その最適解を構成する諸変数の数値は，その値が正のものと，ゼロのものがある．「そして，正の値をとる変数の減少費用は，常にゼロであり，またゼロの値をとる変数の減少費用は正の値（稀に，ゼロのときもあるが）になる．」

正の値になる減少費用（$Z_j - C_j$）の意味は，「その変数（X_j）の利益寄与（目的関数の係数 Cj のこと）が，現在の値よりもあとどれだけ大きいと，その変数 X_j が最適解で正の値をとるかを示す．」減少費用のもう 1 つの解釈は，現在の最適解でゼロの値をとっている変数が，**少しだけその値を増加するように強制されたとき，目的関数の値が悪化する割合**を示している．

例えば，ある変数 X の減少費用が 2 ドル／ガロンとする．第 1 の解釈は，X の利益率（X_{j1} 単位あたりの目的関数の Cj 係数）が現在の値よりも 2 ドル以上大きくならないなら，この変数の値（活動水準）はゼロのままであることが最適解である．また，一方変数 X_j の利益率 C_j を今のままにして X を 1 単位増加すれば，最適解は 2 ドルだけ悪くなる．すなわち，目的関数の値（総利益）は，2 ドルだけ減少してしまうので，この変数 X_j は，ゼロ値であることが最適であるということになる．

1.5.2　双対価格

訳注：ここでの説明は少し理解しにくいかもしれないが，制約式を 1 単位緩めたり厳しくした図を描いて考えたり，LINGO でモデルを実際に修正した解と比較すればよく分かる．

各制約式には，双対価格が出力される．目的関数の単位が円で制約式の単位がキログラムである場合，双対価格の単位は円／kg となる．双対価格の値は，「今，右側の定数項（右辺定数項）の値を少し増加したときに，目的関数の値を改善する割合を表わしている．」数理計画法のプログラム（ソルバーという）によって，双対価格に関する符号の規則が異っている．LINGO では双対価格が正の場合，それは右辺定数項が 1 単位増加したときに，目的関数がどれだけ改善されるかを示す．また負の双対価格の意味は，右

辺定数項が 1 単位増加したとき，目的関数がどれだけ悪化するかを示す．双対価格がゼロの値の時は，右辺定数項を変化させても，目的関数には影響しないことを示す．

従って，この習慣の場合，≦で示された制約式は非負の双対価格をとり，≧で示された制約式は非正の双対価格をとる．そして＝で示された制約式はいずれの値の双対価格をとりうる．なぜであろうか？

そして，「減少費用は符号を逆転した双対価格である」ことに留意されたい．ここでの慣例では，ある変数 X の減少費用は，X が非負という制約で，符号が逆転した双対価格になる．ある変数 X の減少費用が何を測っているかといえば，変数 X の値がゼロの値から増加し始めたときに，目的関数の値が悪化する割合である．一方，X が非負であるという制約に対応する双対価格が測っているものは，右辺定数項がゼロから増加し始めたときに，目的関数値，すなわち解の値が改善する割合である．（制約 Xj≧0 の右辺定数項が増加すれば，制約がいよいよ厳しい方向に向かうので，目的関数値は好ましくない方に変化する．従って双対価格は負になる．）

ET 社問題の解で双対価格を分析すると話が分かりやすい．この問題で，A ≦ 60 という制約に対する双対価格は 1 単位あたり 5 ドルである．一見，この双対価格は 1 単位あたり 20 ドルかと思われる．なぜなら，Astro をあと 1 単位追加生産すれば，その利益は 1 単位あたり 20 ドルである．ところが，Astro をあと 1 単位生産すると，どこか別のところで犠牲が必要となる．今の解では，労働供給はすべて利用されている．そこで Astro をさらにもう 1 台生産するためには，Cosmo の生産を減少しなければいけない．Astro と Cosmo の労働のトレード・オフは 1/2 である．すなわち，Astro をもう 1 台追加生産することは，Cosmo の生産が 1/2 台減少することを意味する．Cosmo は 1 台あたり 30 ドルの利益がある．ここで，正味の利益増分は，＄20 － 1/2×＄30 ＝＄5 となる．次に，労働制約の双対価格が，15 ドル／時間であることについて考えてみよう．もし，もう 1 人日だけ労働を利用できるとしたら，それはすべて Cosmo の生産にふりむけられる．Cosmo1 台の利益は 30 ドル／台である．もう 1 単位余分の 1 人日は，1/2 台の Cosmo の生産に相当するから，1 単位余分の価値は 15 ドル／人日である．

1.6　非有解定式化の例

ここで，労働制約と Cosmo の生産制約を忘れたとしよう．そうすると，Cosmo を限りなく生産でき，無制限の利益が可能となる．線形計画法は次のようになる．

```
MAX = 20 * A + 30 * C;
A <= 60;
```

この結果，次のメッセージウインドウが現れる．

```
UNBOUNDED SOLUTION
```

図 1.7 は実行可能領域で，C が無限に大きくなることを防ぐ制約はない．通常の規模の線形計画法では，無制限に増加できる変数が複数あり，どういう形で無制限な利益拡大ができるか，あるいはできないかということは，簡単には分からない．

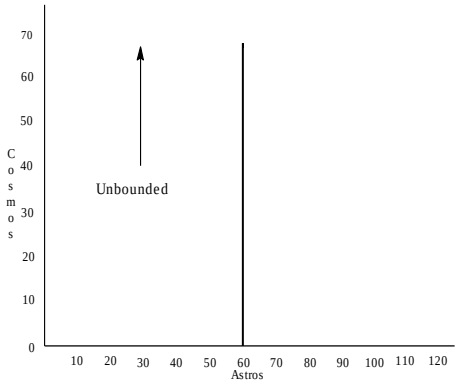


図 1.7 非有界

1.7　不可能定式化の例

実効可能解が存在しない定式化の例として，労働制約の右辺を 190 とし，誤って不等号の向きを逆にした場合を考えてみる．この場合，利用できる労働力は Astro を 60 台，Cosmo を 50 台生産するのに利用される．そして，総労働量は 160(=60＋2×50) 人日である．この場合の定式化と解は次になる．

```
MAX = 20 * A + 30 * C;
A <= 60;
C <= 50;
A + 2 * C >= 190;
```

ウィンドウに次のエラーメッセージが現れる．

```
NO FEASIBLE SOLUTION
```

出力は次の通りである．

Variable	Value	Reduced Cost
A	60. 00000	0. 0000000
C	50. 00000	0. 0000000
Row	Slack or Surplus	Dual Price
1	2700. 000	0. 0000000
2	0. 0000000	1. 000000
3	0. 0000000	2. 000000
4	-30. 00000	-1. 000000

この例では，なぜ不可能な事態が発生したかを，双対価格が説明している．例えば，第 2 行の双対価格の「＋1」は，第 2 行の右辺が 1 だけ増加するとき，不可能性が 1 減ることを示す．また，第 3 行の双対価格の 2 は，Cosmo をもう 1 台余分に生産できるなら，不可能性が 2 減少することを示す．なぜなら，Cosmo を 1 台生産するのには，2 人時必要だからである．そして，第 4 行の双対価格の-1 は，労働制約の右辺が 1 減少し

たとき，不可能性が 1 減少することを表わしている．**図 1.8** はこの制約式を図式化している．

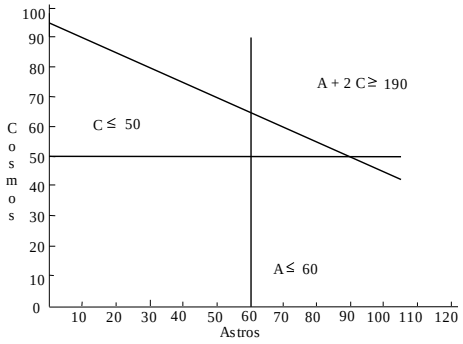


図 1.8　実行可能解のない例

1.8　複数の最適解および退化

有界な最適解が得られるように定式化された線形計画法は，唯一の最適な目的関数の値をとる．しかし，同じ最適な目的関数の値をとる異った解を得ることがある．例えば，A の利益を 5 ドル/台減らすと，解と出力は次のようになる．

```
MAX = 15 * A + 30 * C;
A <= 60;
C <= 50;
A + 2 * C <= 120;
Optimal solution found at step:          1
Objective value:                  1800. 000
Variable           Value        Reduced Cost
       A        20. 00000          0. 0000000
       C        50. 00000          0. 0000000
     Row    Slack or Surplus      Dual Price
       1        1800. 000          1. 000000
       2        40. 00000          0. 0000000
       3       0. 0000000          0. 0000000
       4       0. 0000000          15. 00000
```

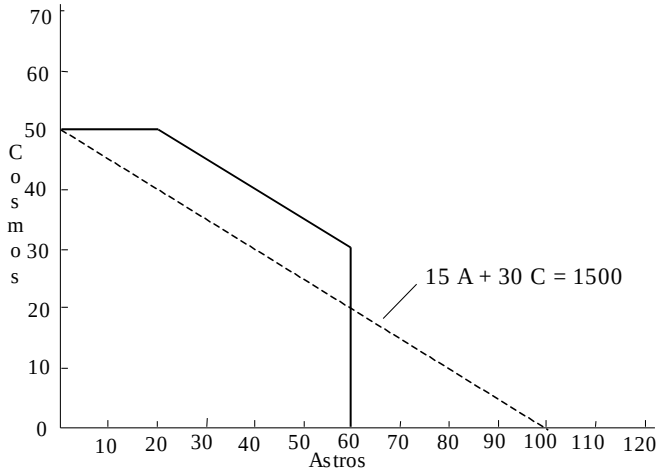


図 1.9 複数の最適解のあるモデル

図 1.9 は，1500 ドルの等利益直線と実行可能領域を表している．A ＋ 2C = 120 と 15A + 30C = 1500 の 2 直線が平行なことに注意してほしい．A ＋ 2C = 120 上の実行可能領域の点，すなわち（A,C）=（20,50）と（A,C）=（60,30）を結ぶ線分上の点全てが最適解になる．

訳注:決定変数の A と C は 0 でないので，減少費用は 0 になっている．減少費用は，決定変数が 0 のものがあれば，それを 1 単位無理に生産すると MAX 問題の場合は利益が減少する値を表す．一方，次の 2 つの制約式のスラックが 0 であるので在庫を増やしても利益に貢献しない．経営に問題のある会社は，過剰在庫を抱えていることが多い．

```
A <= 60;
C <= 50;
```

これに対して，次の労働制約のスラックが 0 であるので 120 を 121 に増やすことを考える．

```
A + 2 * C <= 121;
```

最適解は C = 50 と A + 2 * C = 121 の交点の（A,C）=（21,50）になる．目的関数の値は，15*(20 + 1) + 30*50 = 1800 + 15 で 1800 が 15 だけ増え双対価格は 15 になる．

特に注意深い読者は，次のことに気づくであろう．C≦ 50 という第 3 行の制約では，スラック変数も双対価格も共にゼロである．これは，Cosmo の生産が，総利益に影響なく少量減少できることを示している．もちろんこの場合，Cosmo の生産を減らした分，相殺する意味で，Astro の生産を少し増やす必要がある．そこで，次のように結論づけられる．Cosmo をもうすこし少なく生産し，Astro をもう少し多く生産するという代替的な最適解」が存在する．このことは，次のように Astro の利益率をほんの少し増してみれば分かる．

```
MAX = 15.0001 * A + 30 * C;
```

```
A <= 60;
C <= 50;
A + 2 * C <= 120;
```

出力は次の通りである.

```
    Optimal Solution found at step:         1
    Objective value:                 1800. 006
    Variable             Value        Reduced Cost
           A           60. 00000           0. 0000000
           C           30. 00000           0. 0000000
         Row  Slack or Surplus            Dual Price
           1           1800. 006            1. 00000
           2          0. 0000000       0. 1000000E-03
           3           20. 00000           0. 0000000
           4          0. 0000000            15. 00000
```

予想したように，利益はほぼ 1,800 ドルのままである．しかし，Cosmo の生産は 50 から 30 に減少し，Astro の生産は 20 から 60 に増加し，（A,C）=（20,50）という端点から，（A,C）=（60,30）というもう一つの端点を選んでいる．

訳注：次の 1．8．1 と 1．8．2 は数理計画法の話で，「問題解決学」に重要でないので，最初は読み飛ばすことを勧める．ただし私の判別関数の理論構築に際し，この説明が記憶に残っていて，トラブルの解明に非常に助かった．

1.8.1 蛇の目

一般的にいって複数の最適解が存在するのは，どれかの変数がゼロの値をとり減少費用もゼロの値をとる場合，あるいは，どれかの制約でスラック変数と双対価格もゼロになる場合である．数学的にいうと，そのような解は，「退化」と呼ばれる．従って，代替的な最適解が存在する場合には，PC の解と教科書の解が異なる場合がある．しかし，その場合でも目的関数の値は同じになる．**図 1.9** の例では 2 つの最適解レポートは，いわゆる**基底変数** A と C の値と変数 A，C の制約式のスラック変数でのみ異なっている．双対変数のみが異なる複数の最適解がある場合もある．ET 社問題で Cosmo の生産能力を 30 にした例を考えてみよう．定式化は次の通りである．

訳注：基底変数と双対変数は，LP を解くアルゴリズムの単体法でよく用いられる専門用語である．問題解決に興味がある読者には不用な知識である．

```
MAX = 20 * A + 30 * C;
A < 60;
C < 30;
A + 2 * C < 120;
```

この問題のグラフを図 1.10 に示す．最適解は次の通りである．

```
Optimal Solution found at step:          0
Objective value:                  2100. 000
Variable                Value        Reduced Cost
       A             60. 00000           0. 0000000
       C             30. 00000           0. 0000000
     Row       Slack or Surplus      Dual Price
       1             2100. 000           1. 000000
       2            0. 0000000           5. 000000
       3            0. 0000000           0. 0000000
       4            0. 0000000           15. 00000
```

再び解に「ヘビの眼」が現れている．すなわち，制約式の 3 番目にゼロのペアが現れている．これは Cosmo の生産能力を，目的関数の値を変えないで変更できることを意味する．図 1.10 はこの状況を説明している．3 つの制約式が A=60, C=30 の点で交わる．制約式の 2 つの組もこの点で交わる．A+2C≦120 は冗長であるので，これを省いても実行可能領域は変わらない．

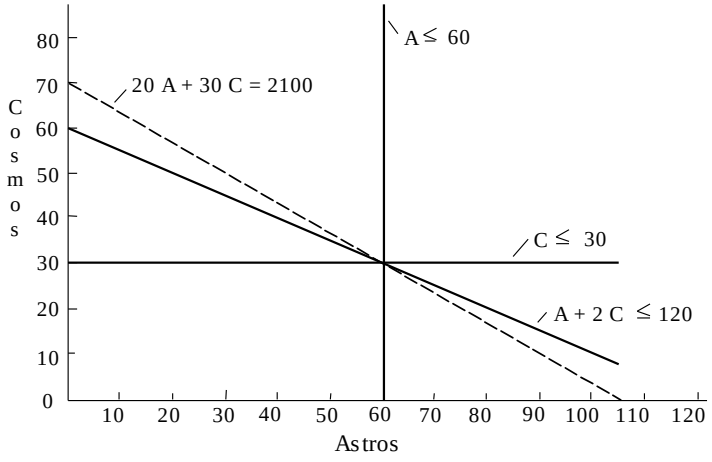


図 1. 10　双対変数の代替解

もし制約式 3 の右辺定数項を少し減らすと次の解が求まる．

```
Optimal Solution found at step:          0
```

```
Objective value:                2100. 000
Variable                Value        Reduced Cost
       A              60. 00000          0. 0000000
       C              30. 00000          0. 0000000
     Row      Slack or Surplus      Dual Price
       1           2100. 000           1. 000000
       2          0. 0000000           20. 00000
       3          0. 0000000           30. 00000
       4          0. 0000000          0. 0000000
```

この解は，前の解と双対変数の値でのみ異なっている．以上をまとめれば，次のルールが成立する．もし解が「ヘビの眼」をもつなら，基底変数か双対変数のいずれかあるいは両方で異なる他の最適解の代替案をもつ．

```
MAX = 20 * A;
A <= 60;
C =  30;
```

解は次の通りである．

```
Optimal Solution found at step:          0
Objective value:                 1200. 000
Variable            Value        Reduced Cost
       A        60. 000000          0. 0000000
       C        30. 000000          0. 0000000
     Row    Slack or Surplus      Dual Price
       1         1200. 000           1. 000000
       2        0. 0000000           20. 00000
       3        0. 0000000          0. 0000000
```

もし解に「ヘビの眼」があれば，自然な問いとして「最適解の代替案が基底変数か双対変数のいずれであるかを，出力結果のみから決めることができるだろうか」という疑問が残る．答は否である．次の２つの関連する問題を考えよう．

問題 D	問題 P
MAX = X + Y; X + Y + Z <= 1; X + 2 * Y <= 1;	MAX = X + Y; X + Y + Z <= 1; X + 2 * Z <= 1;

両モデルとも，次の複数の最適解がある．

解 1

問題 D

OBJECTIVE VALUE

1)	1	
変数	値	減少費用
X	1	0
Y	0	0
Z	0	1
行	スラック/サープラス	双対価格
2)	0	1
3)	0	0

問題 P

OBJECTIVE VALUE

1)	1	
変数	値	減少費用
X	1	0
Y	0	0
Z	0	1
行	スラック/サープラス	双対価格
2)	0	1
3)	0	0

解 2

問題 D

OBJECTIVE VALUE

1)	1	
変数	値	減少費用
X	1	0
Y	0	1
Z	0	0
行	スラック/サープラス	双対価格
2)	0	0
3)	0	1

問題 P

OBJECTIVE VALUE

1)	1	
変数	値	減少費用
X	0	0
Y	1	0
Z	0	1
行	スラック/サープラス	双対価格
2)	0	1
3)	1	0

上の解から次のことが分かる．

・解 1 は，両方の問題とも同じ解になる．

・問題 D は，双対変数が異なる複数の最適解をもつ．

・問題 P は，基底変数が異なる複数の最適解をもつ．

以上のように，出力結果のみから代替案が基底変数あるいは双対変数のいずれで異なるかを決めることはできない．解 1 は，制約式 3 の右辺定数と目的関数 X の係数を 1.001 のように少し大きくすると求まる．解 2 は，逆に 0.9999 のように少し小さくすると求まる．

研究者の中には，基底変数が異なる複数の最適解のことを「dual degenerate」と呼び，双対変数の場合を「primal degenerate」と呼んでいる．他の研究者は，基底変数に複数の最適解をもつ場合のみ，複数の最適解をもつ問題と定義している．

1.8.2 退化と冗長な制約式

2 次元で例を考えることは問題の見通しをよいが，時々この見通しは間違った結論を与えることがある．2 次元において次のことは正しい．

2 次元において，n 個（n＞2）の不等式制約があり，すべてが 1 点で交わる場合，少なくとも（n-2）個の制約式は冗長である．

3 次元へこれを一般化しても正しくはない．実際，3 次元で n 個（n＞3）の不等式制約があり，すべてが 1 点で交わっている場合，それらの全ては必ずしも冗長ではない．すなわち，大きな問題で冗長な制約式があっても，必ずしも退化は起こらない．

これは次の制約式とそれを図化した**図 1.11** を見れば明らかである．

2x－y≦ 1　　2y － x≦ 1　　2z － x≦ 1

2x－z≦ 1　　2y － z≦ 1　　2z － y≦ 1

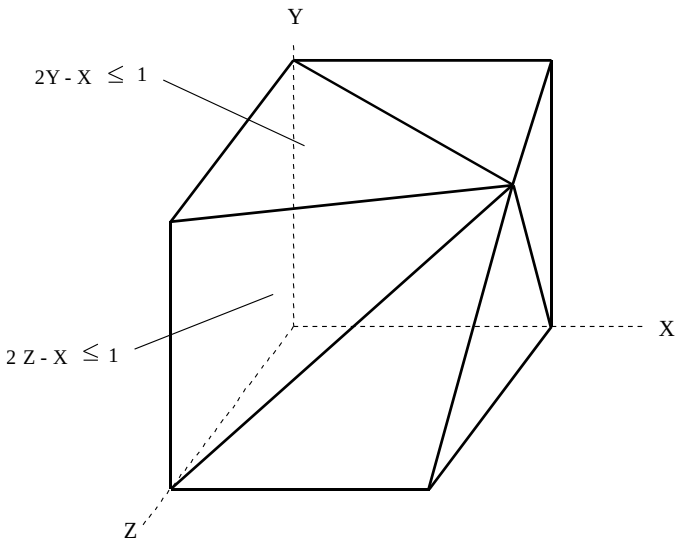


図 1.11 退化しているが冗長ではない例

これらの制約式は，点 x=y=z=1 で 6 面をもつ角錐の頂点になっている．この点は，3 個以上の制約式が交わっているので退化している．しかし，いずれの制約式も冗長ではない．点 x=0．6，y=0，z=0．5 は最初の制約式を満足していないが，他の全てを満足しているので最初の式は冗長ではない．以下同様にして，0．6，0，0．5 の 6 個の全ての組合せを考えると，6 個の制約式のいずれも冗長でないことが分かる．

1.9　非線形モデルと大域的最適化

本書では，LP モデルを重要視している．歴史的に非線形モデルは，次の 2 つの理由でできれば避けたい．a)計算時間がかかる．b)解が求まっても，大域的探索オプションを利用できなければ局所最適解が求まるだけである．

より良い解が近くになければ，解は局所最適解である．しかし，遠く離れてより良い解があるかもしれない．従来の NLP のソルバーは，近視の登山者のようなものである．彼らは最高峰でなく，最も近い峰の頂上に登頂する．決して，山系の最高峰に登頂した保証はない．LINGO8 以降，Global オプションが提供された．このオプションを利用すれば，大域的最適解が保証される．実例を示すため，次の問題を考える．

図 1.12 は，次の式を図示したものである．

```
Min = @sin(x) + 0.5 * @abs(x - 9.5); x <= 12;
```

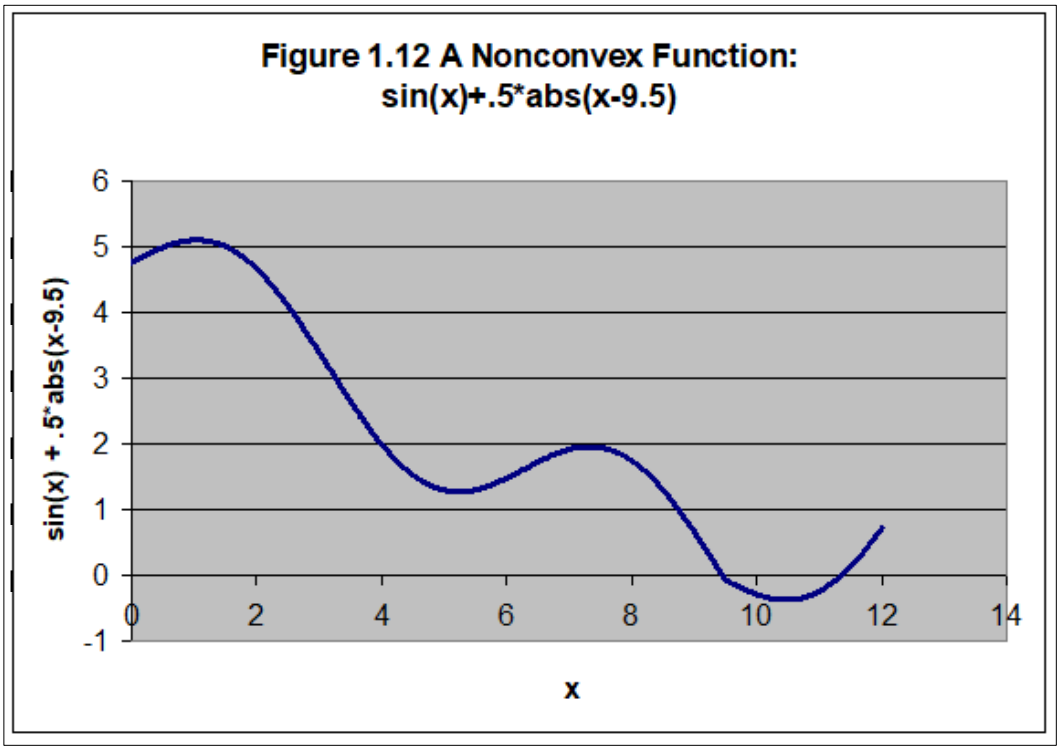


図 1.2　非凸関数（SIN(x)+0.5*ABS(x-9.5)）

もし従来のソルバーでこのモデルを解くと，初期値の違いで x = 0，x = 5.235987，x = 10.47197 の何れかの解を求める．もし Global オプションを指定すれば，解 x = 10.47197 が大域的最適解であることが分かる．しかし，大域的最適解が求まっても，複雑なモデルでは計算時間がかかる場合があるので，欠点(a)は改善できない．

第 2 章　LINGO について

2.1　はじめに

数理計画法は多大な計算量を要するので，PC で計算するのが普通である．本書では，線形計画法，2 次計画法，整数計画法，非線形計画法，確率計画法をサポートするソルバーの LINGO を用いている．LINGO は，モデルを簡単に入力し，それを解き，その解に基づいてモデルが適切かどうかを検討し，そのモデルを簡単に変更することを繰り返すことができる． 顧客は，どの解法を用いるかを指定する必要がなく，LINGO がモデルに適切な解法を選ぶ．また，LINGO には多くのコマンドがあり，それらが正しく使われているかどうか文法チェックする．

LINGO は，次の 2 つの方式で利用できる．

１）　Windows 環境

２）　テキスト形式

テキスト形式は，Unix や MS-DOS でも使用できる．どの版でも，より詳しい情報は，HELP メニューやコマンドで入手できる．Windows のコマンド（命令）を以下で簡単に紹介する．コマンドの紹介の後，簡単なモデルを作成し解いてみよう．

訳注:個人の問題解決は，PC の Windows 環境が最適である．テキスト形式の紹介があるが，産業機械に最適化モデルを組み込むことなどの，大規模な最適化システムをバッチ処理するのに適している．

2.2　Windows 版の LINGO

Windows 版を開始すると，モデル・ウインドウが開く．モデル・ウインドウでモデルを定義する．LINGO の出力は，レポート・ウインドウに表示される．LINGO はモデルに関係する幾つかのレポートを出力する．次は主としてモデル・ウインドウで利用できる命令のリストである．

訳注:　本節の説明は，実際にモデルを PC で操作するようになってから辞書的に読むのが良いかもしれない．

2.2.1 File Menu

NEW F2 

モデル・ウインドウで，新規のモデルを入力するには，ファイル・メニューで NEW（F2）を使用するか，上のアイコンを使用する．

OPEN F3 

既存のファイルを開くには，ファイル・メニューで OPEN(F3)を使用する．

SAVE F4 

モデル，レポート，または命令を含むウィンドウを保管するため，ファイル・メニューで SAVE(F4)を使用する．

SAVE AS F5

新しいモデル，レポート，または命令を含むウィンドウを保管するため，ファイル・メニューで SAVE AS(F5)を使用する．

CLOSE F6

アクティブなウィンドウを閉めるために，ファイル・メニューで CLOSE（F5）を使用する．

PRINT F7 

アクティブなウィンドウを印刷するには，ファイル・メニューで PRINT（F7）使用する．

PRINT SETUP F8

出力プリンターを選ぶために，ファイル・メニューで PRINT SETUP（F8）を使用する．

PRINT PREVIEW Shift +F8

アクティブなウィンドウを印刷のイメージで表示するため，ファイル・メニューで PRINT PREVIEW（Shift +F8）を使用する．

LOG OUTPUT F9

COMMAND ウィンドウのモデル，コマンド，出力をテキスト形式の LOG ファイルに送るために，LOG OUTPUT（F9）を使用する．COMMAND ウィンドウは，LINDO で最初に使われた会話形式のモデル作成と実行形式を，LINGO のウィンドウで行うようにしたものである．

LICENSE

LINGO に新しいパスワードを入れるために LICENSE を使用する．LINGO のアプリケーションを開ける鍵としてもこのパスワードを利用できる．LINGO を更新すれば，新しいパスワードを入れる必要がある．

訳注：無償の評価版ではライセンスの入力は不要である．LINDO Japan から正規版を購入すれば，ライセンスが送られてきて，評価版を立ち上げてライセンスを入れれば商用版になる．また，HP で会員になれば，日本語マニュアルや英語版のマニュアルの PDF は無償でダウンロードできる．

EXIT F10

LINGO を終了するために，ファイル・メニューから EXIT（F10）を使用する．

訳注：以下は，個人の問題解決には不要である．

TAKE COMMANDS　F11

命令およびモデルを含むLINGOのバッチファイルを取り込むために，ファイル・メニューからTAKE（F1）を使用する．

IMPORT LINGO FILE　F12

LINGOのTAKEフォーマットのLINGOモデルを含んでいるファイルを開くために，ファイル・メニューからIMPORT LINGO FILE（F12）を使用する．

MPS FILE

MPSファイル形式を入出力するために，File|MPS Fileから Importか Exportサブコマンドを使用する．MPS ファイル形式は IBM が開発した業界標準のフォーマットで，1つのソルバーから別のソルバーにモデルを渡すために有用である．

2.2.2 Edit Menu

訳注：雛型モデルで問題解決を学ぶ段階では不要である．

UNDO Ctrl+Z

最後の行為を取消すために，編集メニューからUNDO（Control+Z）を使用する．

CUT Ctrl+X 

クリップボードに指定したテキストを切りとったり，他の場所に貼り付けるために，編集メニューからCUT（Control+X）を使用する．

COPY Ctrl+C 

指定したテキストをCOPYしたり，クリップボードから貼り付けるため，編集メニューからCOPY(Control+C)を使用する．

PASTE Ctrl+V 

挿入位置でクリップボードの内容を貼るために，編集メニューからPASTE（Control+V）を使用する．

PASTE SPECIAL

挿入位置でクリップボードの内容を希望の内容で貼るために，編集メニューからPASTE SPECIAL（Control+V）を使用する．

SELECT ALL　　Ctrl +A

アクティブ・ウインドウ全体の内容を選ぶため，編集メニューから SELECT ALL (Ctrl +A) を使用する.

FIND　　Ctrl+F 

アクティブ・ウインドウで希望のテキストを捜すため，FIND (Control+F) を使用する.

FIND NEXT　　Ctrl +N

アクティブ・ウインドウで次の希望のテキストを捜すため，FIND (Control+F) を使用する.

REPLACE Ctrl +H

アクティブ・ウインドウで希望のテキストを他のテキストに置き換えるため，REPLACE (Control+H) を使用する.

GO TO LINE　　Ctrl+T 

アクティブ・ウインドウで行きたい行数に飛ぶため，GO TO LINE (Control+T) を使用する.

MATCH PARENTHESIS Ctrl+P 

選んだ左括弧に対応する右括弧を見つけるために，編集メニューから MATCH PARENTHESIS (Control+P) を使用する.

PASTE FUNCTION

現在の挿入位置に LINGO の組込み関数を貼りつけるため，編集メニューから PASTE FUNCTION を使用する．貼りたいと思う LINGO 関数のカテゴリーを選びメニューから関数を選び，引数を挿入する.

SELECT FONT

現在指定しているテキストの新しいフォントを選ぶため，編集メニューから SELECT FONT 命令を使用する.

INSERT NEW OBJECT

LINGO 文書に OLE 目的関数を埋め込むため，編集メニューから INSERT NEW OBJECT を使用する.

LINKS

あなたの文書に外部の目的関数をリンクするため，編集メニューから LINKS 命令を使用する.

OBJECT PROPERTIES Alt+Enter

指定した埋め込まれた目的関数の特性を指定するため，編集メニューから OBJECT PROPERTIES（Alt+Enter）を使用する．

2.2.3 LINGO Menu

SOLVE Ctrl+S 

LINGO のソルバーにモデルを送るため，SOLVE（Ctrl +S）を使用する．1 つ以上のモデルをウィンドウで開いておれば，送られるのは現在アクティブなものが選ばれる．

SOLUTION… Ctrl+O

LINGO メニューから SOLUTION ダイアログボックスを開いて SOLUTION（Control+O）を使用する．SOLUTION レポートの出力方法を指定できる．OK をクリックすると，LINGO はレポート・ウインドウにレポートを書きこむ．

RANGE Ctrl+R

範囲分析を見るため， RANGE（Control+R）を使用する．

OPTIONS　Ctrl +I 

プログラムの構成に影響を与える幾つかの変数を変えるため，LINGO メニューから OPTIONS（Ctrl+I）を使用する．

GENERATE　Ctrl+G

現在のモデルを LINGO 形式や MPS 形式のモデルを作成するため，GENERATE（Ctrl+G）を使用する．

訳注:MPS 形式とは，かつて IBM のソルバーのファイル形式．他のソルバーがこの形式を提供することで，異なったソルバー間で，モデルを利用できる．

PICTURE Ctrl+K

行列形式でモデルを表示する，LINGO メニューで PICTURE(Ctrl+K)を使用する．行列形式のモデルを見ることで，モデルの特別な構造が識別できる場合もある．

訳注:本書ではテキスト形式の出力を用いている．最新版では，グラフ表示にかえたのでかえって利用しづらくなったと考えている．テキスト形式の出力の復活を提案している．

MODEL STATISTICS Ctrl+E

モデルの要約統計量をリストするため，MODEL STATISTICS （Ctrl+E）を使用する．統計量でモデルが線形か非線形であるかどうかが分かる．

訳注:係数の最大値と最小値が特に重要．最小値を最大値で割ったものの絶対値が，0.0001 以上が望ましい．0.000001 以下になると計算結果に問題が生じることがある．これは他のソフトでも注意すべきことである．

LOOK… Ctrl+L

全てのモデルや指定した行を見るため，LOOK（Control+L）を使用する．

EXPORT TO DATABASE　Ctrl +D

ターゲットのデータベース管理システム(DBMS)の示された列に指定したい名前をつけるため，LINGO メニューから DATABASE を使用する．

EXPORT TO SPREADSHEET

指定した範囲に指定された名前をつけて Excel などに出力するため，EXPORT TO SPREADSHEET を使用する．

2.2.4 Windows Menu

COMMAND WINDOW Ctrl +1

LINGO の COMMAND ウィンドウを開けるため，ウィンドウメニューから COMMAND WINDOW（Ctrl+1）を使用する．COMMAND ウィンドウで，LINGO のコマンドライン・インターフェイスへアクセスできる．**一般に，Windows 顧客には必要はない．**これはプログラムを制御するため，COMMAND WINDOW のスクリプトで LINGO を利用するアプリケーションを作成したい 顧客に提供される．コマンドライン・コマンドは，利用者マニュアルを参照．

訳注：本格的な LINDO API（LINGO や What Best!の開発に用いた C ライブラリー集）を用いたシステム開発の変わりに簡単にアプリケーションを作成できる．

STATUS WINDOW Ctrl +2

Solver Status ウインドウを開くため，STATUS WINDOW（Ctrl +2）を使用する．

SEND TO BACK Ctrl+B 

現ウィンドウを背部のウィンドウにするため，SEND TO BACK（Ctrl+B）を使用する．この命令はモデル・ウインドウとレポートのウィンドウを切り替えるために非常に有用である．

CLOSE ALL Alt+X 

全てのモデルやダイアログボックスを閉めるため，CLOSE ALL（Alt+X）を使用する．

TILE Ctrl +4

ウィンドウ内をタイル状に表示するため TILE (Ctrl +5) を使用する.

CASCADE Ctrl +5

全ての開いているウィンドウで左上から右下の範囲を指定し, カスケイド枠を作るため, CASCADE (Control +5) を使用する.

ARRANGE ICONS Ctrl +6

最小化されたウィンドウを表すアイコンを画面の一番下に整理するため, ARRANGE ICONS (Ctrl+6) を使用する.

2.2.5 Help Menu

HELP TOPICS F1 

HELP Topics を使用して, 分からないことが調べられる.

ABOUT LINGO

現在使用している LINGO の版についての情報(例えば, リリース番号, 制約と変数の数, 記憶容量)を見るため, ABOUT LINGO を使用する.

2.2.6 概要

LINGO の Windows 版で利用できる命令や, より詳細な分析に関しては利用者マニュアルを参照.

2.3 小さな問題への利用

Windows 版の LINGO を起動すると, <untitled>のウィンドウが開く. この<untitled>ウィンドウに, 前の章で見た ET 社モデルを**半角英数字**で入力しよう:

```
MAX = 20 * A + 30 * C;
A < 60;
C < 50;
A + 2 * C < 120;
```

等号なしの「<」は, キーボドに等号ありの「≤」がないので, 実際は「≤」と解釈する. あるいは, 「<=」を使用してもよい. またコメントは感嘆符(!)で始まり, セミコロン(;)で終わる.モデルを解くにはSolve(Ctrl+S)を使用する.計算中はSolver Statusが表示される.

次の解が, Report ウィンドウに表示される.

```
  Optimal solution found at step:          1
  Objective value:                  2100.000
  Variable           Value        Reduced Cost
         A         60.00000           0.0000000
```

```
         C          30. 00000            0. 0000000
       Row    Slack or Surplus      Dual Price
         1          2100. 000            1. 000000
         2         0. 0000000            5. 000000
         3          20. 00000            0. 0000000
         4         0. 0000000            15. 00000
```

モデルの編集は，表示されているモデルを直接「半角英数」で修正すればよい．作成したモデルの保管は，File メニューで「Save As」でモデル名を指定し保管する．

2.4 整数計画法

LINGO は，変数が 0/1 の整数変数の場合**@BIN**(Binary Integer)で，0 以上の整数値の場合は@GIN(General Integer)で一般整数変数を指定できる．より詳しくは，第 11 章を参照．

```
MAX = 4 * TOM + 3 * DICK + 2 * HARRY;
   2. 5 * TOM            + 3. 1 * HARRY <= 5;
    . 2 * TOM + . 7 * DICK + . 4 * HARRY <= 1;
@BIN(TOM);
@BIN(DICK);
@BIN(HARRY);
```

Report ウィンドウに次の解が出力される．

```
Optimal solution found at step:         1
Objective value:                 7. 000000
Branch count:                           0
Variable           Value        Reduced Cost
     TOM        1. 000000           -4. 000000
    DICK        1. 000000           -3. 000000
   HARRY       0. 0000000           -2. 000000
     Row    Slack or Surplus      Dual Price
       1        7. 000000            1. 000000
       2        2. 500000           0. 0000000
       3       0. 1000000           0. 0000000
```

次は，@GIN コマンドの例である．変数 TONIC は 0， 1， 2， 3， …の値をとる．

```
@GIN( TONIC);
```

使用される解法は「**分枝限定法**」である．よりよい解を見つける知的列挙法とでもいう方法である．

```
MAX = 20 * A + 30 * C;
A < 60;
```

```
C < 50;
A + 2 * C < 115;
```

上のモデルを解くと，次の解を得る．

```
    Optimal solution found at step:         1
    Objective value:                 2025. 000
    Variable           Value        Reduced Cost
           A        60. 00000           0. 0000000
           C        27. 50000           0. 0000000
         Row    Slack or Surplus      Dual Price
           1        2025. 000            1. 000000
           2       0. 0000000            5. 000000
           3      22. 50000              0. 0000000
           4       0. 0000000           15. 00000
```

変数 C は実数になる．これが好ましくなければ，A と C を一般整数変数に指定する．

```
MAX = 20 * A + 30 * C;
A < 60;
C < 50;
A + 2 * C < 115;
@GIN( A);
@GIN( C);
```

結果は次の通りである．

```
    Optimal solution found at step:         4
    Objective value:                 2020. 000
    Branch count:                           1
    Variable           Value        Reduced Cost
           A        59. 00000          -20. 00000
           C        28. 00000          -30. 00000
         Row    Slack or Surplus      Dual Price
           1        2020. 000            1. 000000
           2        1. 000000           0. 0000000
           3        22. 00000           0. 0000000
           4       0. 0000000           0. 0000000
```

2．4．1 整数計画法の注意

整数計画法(IP)は非常に強力であるが，効果的に使用するためには技術を必要とする．LP と異なり，IP として問題を定式化すると，短時間で解けるとは限らない．本質的に容易な問題を悪いモデルにしがちである．悪いモデルは膨大な計算時間を要求す

るかもしれない．従って，IP モデルを解く場合，経験ある人に相談すべきである．良い定式化は，第 11 章で更に議論される．

2.5　その他の注意事項

線形あるいは整数計画法の解法は，厳しい数値計算上の問題の解決が求められる．ここでは，解法のアルゴリズムの詳細を示さない．効率的なソルバーの作成には，数年の開発工数が要求される.アルゴリズムの入門書として,Martin(1999)やGreenberg(1978)が進められる．

商用ソフトは頑強性が求められるが， 顧客はモデルの中に極端に大きなあるいは小さな係数を用いないことである．モデルをスケーリングし，極端に大きなあるいは小さな係数を用いないようにしてください．だいたいの目安は非ゼロ係数が 100,000 を越える，あるいは 0.0001 より小さい場合は，その単位を変更すべきである．LINGO は，この係数行列のスケールがよくないと判断をした場合には，警告を印刷する．

スケーリングと数値的な考慮：LINGO は，モデルのデータ行列のスケールを自動変更できない．従って， 顧客は，数値的な問題が生ずるのを避けるために，行あるいは列のスケールを調整しておく必要がある．

第3章　解の分析

3.1　解の経済的分析

解のレポートから，面白い経済的な相当量の情報が収集できる．さらに，範囲分析などのオプションのレポートは,詳細な情報を提供してくれる.これらの情報の使用法は，**What if 分析**をすることである．典型的な What if 分析の質問のポイントは次の点である.

(a)容量や需要を変えることの影響は？

(b) 新しい機会が利用可能になったらどうなるか？ それは，価値があるだろうか？

3.2 双対価格と減少費用の経済的関係

減少費用と双対価格は関係がある．変数 x の減少費用は，制約 x>=0 がある場合の双対価格の符号を逆にしたものに等しい．x の減少費用は，x が 0 から増えた場合に解が悪化する率を測定している．x>=0 の双対価格は右辺定数項が 0 から増えたとき解がよくなる率を測定している.

減少費用と双対価格は，次のように整理できる.

①活動水準がゼロの減少費用：その活動水準を 1 単位強制的に増加したときに，目的関数すなわち利益が減る量を示す.

②制約式の双対価格：制約式に関連する資源の利用可能な量が，1 単位だけ増えたときの利益の増加分.

ある活動水準の減少費用は，次のような解釈が成り立つ．ある活動の活動水準を増加することは，各種資源を喰いつぶすことである．それらを双対価格で価格づけると，減少費用は正味の費用効果になるという経済的な意味がでてくる．もし，ある活動水準を 1 単位だけ増加したとき，それは各資源を消費するので，他の活動のための利用可能性を減少させる効果がでてくる．ところが，これらの資源は双対価格で価値づけされているので，その活動には負担金を科す必要がある．これを，例を用いて示そう.

3.2.1 費用の算定：例

ET 社は，生産ラインにビデオレコーダーの追加を検討しています．同社の市場調査部門と技術部門は，ビデオレコーダーの利益寄与（C_j)は 47 ドル/単位と見ている．この場合，新しいビデオレコーダーは，Astro のラインを使って生産され，また 3 人日の労働が必要である．これを生産すると，Astro と Cosmo の両方の生産が減少する．このトレード・オフは価値がありますか？

ビデオレコーダーは，Astro と Cosmo に比べて 1 単位あたりの利益が多く，Astro に比べて Astro の生産ラインの利用効率はよい．Cosmo と比べると 1 単位当たりの労働時間が多く，Cosmo に比べて労働制約は悪い.

元の解で，Astro の制約式の双対価格が 5 ドルで，労働制約の双対価格が 15 ドルである．従って，追加製造するビデオレコーダーの価格づけは， 1×5 ＋ 3×15 - 47 = 3（ドル）となる．この正味の費用は正である．従って，このビデオレコーダーを生産することは，明らかに価値がない．

制約式	係数	双対価格	チャージ
1	-47	1	-47
2	1	+5	+5
3	0	0	0
4	3	15	**+45**

Total opportunity cost = +3

ここで，手計算による分析は終え，LP を解いてみよう．今，Ｖをビデオレコーダーの生産数とすると，次のようになる．

```
MAX = 20 * A + 30 * C + 47 * V;
          A                + V <=  60;
                     C          <=  50;
          A  + 2 * C  + 3 * V <= 120;
```

解は，次の通りである．

```
  Optimal solution found at step:         1
  Objective value:                 2100.000
  Variable           Value        Reduced Cost
         A        60.000000           0.000000
         C        30.000000           0.000000
         V         0.000000           3.000000
       Row    Slack or Surplus      Dual Price
         1      2100.000000           1.000000
         2         0.000000           5.000000
         3        20.000000           0.000000
         4         0.000000          15.000000
```

この解では，ビデオレコーダーは生産されていない．しかも，Ｖの減少費用が 3 ドルである．この 3 ドルは，我々がＶについて価格づけをした時の値である．この事は，次のような関係を説明する．

ある活動X_jの減少費用（Z_j-C_j）は，その活動（X_j）が利用する諸資源の利用率（a_{ij}）の荷重和から利益寄与（C_j）を差し引いたものである．ただし，荷重として双対価格を使う．

すなわち LP の双対価格は，5×60 ＋ 0×50 ＋ 15×120 ＝ 2100 であるので，総利益が不十分な資源の Astro と労働制約に割り当てられていることに気がつく．

訳注：3.2.2 と 3.3 と 3.4 は，スマホなどの画面で読むには複雑なことと数理計画法の本書で説明していない知識が必要なので，最初は読み飛ばすことを勧める．

3.2.2 双対価格/ラグランジェ定数/KKT 条件と活動費用

LINGO や Excel の Add-in ソルバーの What's Best!で LP 問題を解くと，各制約の双対価格をえる．単純化のため，目的関数が最大化で，制約式は左辺の制約式が右辺定数項より小さいか等しい(<=)と仮定する．制約式の双対価格は，制約の右辺定数項に関する目的関数の価値の変化率である．これは，等式制約に対するラグランジェ乗数の不等式制約の一般化である．この考えは，100 年以上前のアイデアである．実例を示すために，次の非線形問題を考える：

```
[ROW1] MAX = 40*( X+1)^.5 + 30*( Y+1)^.5 + 25*( Z+1)^.5;
[ROW2]          X                        + 15* Z   <= 45;
[ROW3]                          Y             + Z   <= 45;
[ROW4]         X* X      + 3* Y*Y      + 9 * Z*Z <= 3500;
```

X, Y, Z >= 0 を仮定している．これを解くと次の解を得る：

```
  Objective value =  440.7100
  Variable           Value        Reduced Cost
         X         45.00000           0.0000000
         Y         22.17356           0.0000000
         Z         0.0000000          0.1140319
       Row    Slack or Surplus      Dual Price
      ROW2         0.0000000          0.8409353
      ROW3          22.82644          0.0000000
      ROW4         0.0000000          0.02342115
```

例えば,ROW2 の双対価格 0.8409353 は,ROW2 の RHS が少ない量(ε)で増えるならば,目的関数の値がおよそ 0.8409353*ε増加することを意味する．変数 Z が使われていない理由を理解しようとするとき，役に立つ展望は活動の費用を概算することである．我々は変数に目的関数の増分に対する信用を与える．適用される率は，各制約式の双対価格である．この値は，変数に関する LHS の単に偏導関数である．変数 Z の値は，下で示される：

Row	Z で偏微分	双対価格	全変化
ROW1	12.5	-1	-12.5
ROW2	15	.8409353	12.614029
ROW3	1	0	0
ROW4	0	.02342115	0
		Net(減少費用)：.11403	

一方，X は次のようになる：

Row	X で偏微分	双対価格	全変化

ROW1	2.9488391	-1	-2.9488391
ROW2	1	.8409353	.8409353
ROW3	0	0	0
ROW4	90	.02342115	2.107899

Net(減少費用)： 0

これらの 2 つの計算は，最適解における Karush/Kuhn/Tucker（KKT）条件である：

①正の減少費用を持つ変数はゼロ．

②利用される変数(値が正)は，減少費用が 0.

③正の双対価格をもつ「<=」制約は，スラックが 0 になる．

④正のスラックをもつ「<=」制約は，双対価格が 0 になる．

これらの状況は，補完的なスラック条件と呼ばれている．

3.3 減少費用と双対価格の有効な範囲

減少費用と双対価格を説明する時，その変化の幅をきわめて小さい範囲に限定してきた．例えば，ある制約の双対価格が 1 時間あたり 3 ドルであったとすると，利用できる時間数を増加させたとき，その利用可能時間をほんのわずか増加した時，利益が 3 ドル/時間だけ改善することを意味する．この改善の割合は，一般には，いつまでも続くものではない．そこで，この利用できる時間をもっと増やしていくと，これらの時間の価値は，最後には増加しなくなり，減少してしまう．これは，一般的には常に正しいとは言えないけれども，LP に限っていえば，ある制約の右辺定数項を増加することによって，その制約の双対価格を増加させることはできない．従って，双対価格は，同じ値にとどまるか，あるいは減る一方である．LP の右辺定数項を変化させた場合，決定変数の最適解は変わることになるが，最適解の性格が変わらない限り，双対価格も減少費用も変わらない．ただし，ここで 0 でない変数の組み合わせが変わるか，あるいは拘束力をもっている制約の組が変化したとき（基底変数が変わったとき）に，最適解の性格が変わったという．要約すると，右辺定数項を変えていった場合，最適解の性格（基底）が変わらない限り，双対価格は変わらない．

ほとんどの LP は，オプションとして解の出力に加えて，感度分析に関する情報も出力する．この感度分析では，上に述べた最適解の性格，すなわち基底が変化しない範囲を示す．前のモデルを再び利用する．：

```
MAX = 20 * A + 30 * C + 47 * V;
          A              + V <=  60;
                 C          <=  50;
          A  + 2 * C  + 3 * V <= 120;
```

LINGO メニューで Range を選ぶと，次の感度分析が得られる．

Ranges in which the basis is unchanged:

Objective Coefficient Ranges

Variable	Current Coefficient	Allowable Increase	Allowable Decrease
A	20.00000	INFINITY	3.000000
C	30.00000	10.00000	3.000000

V	47.00000	3.000000	INFINITY

Right-hand Side Ranges

Row	Current RHS	Allowable Increase	Allowable Decrease
2	60.00000	60.00000	40.00000
3	50.00000	INFINITY	20.00000
4	120.0000	40.00000	60.00000

ここでも出力は2つある．1つは変数に関するものであり，1つは行（制約式）に関するものである．**A行の3の意味は，Aの利益寄与（Cj)は，単位あたり3ドルまで減少させることができ，しかも生産すべきAとCの最適値が変化しない範囲を示す．**これは，Astro と Cosmo 各1台の利益貢献は，合計で$50 であることを示す．もし，Astro と Cosmo の1組の利益寄与が3ドル減少して，単位あたり47ドルまで下がったならば，V の生産で利益が出てくる．1台の V の生産は，Astro と Cosmo 各1台の生産に使う資源と等しい．

この出力の同じ行に，INFINITY という文字があるが，これはAの利益を正の方にどれだけ増加しても，生産すべきAとCの最適量には変化をもたらさないことを示す．このことは，すでにAをぎりぎりまで生産していることから明らかである．

変数Cの「許容減少「3はAとまったく同じである．Cの行にある10の意味は，Cの利益性（利益寄与）が少なくとも1単位あたり10ドル増加して$40/単位にならなければ，AとCの値の変化は考えられない．Cに対して1単位あたり40ドルであると，労働1時間あたりの利益は，AとCで同じになる．

一般的にいって，ある1つの目的関数の係数が，最適解の変数に関する感度分析に示された範囲内で変化するとき，決定変数の最適値（A と C と V の値）は変化しない．しかし，双対価格，減少費用，および解の利益値は変化する．

制約式に関する感度分析の解釈は，ある制約の右辺定数項が，行に関する感度分析に示された範囲内で変化するとき，双対価格と減少費用の最適解は変化しない．しかし，その場合，決定変数の値および解の利益は変化する．

例えば行に関する感度分析から，第3行の右辺定数項，すなわちC≦50という項の右辺定数項が20以上減少すると，双対価格や減少費用が変化する．このとき，制約はC≦30となり，解の性格は，労働制約がもはや拘束として働かないように変化する．C≦50という制約の右辺定数項を増加させた時，感度分析によると，上の方にいくら増加しても最適な双対価格や減少費用は影響を受けない．すでに Cosmo のラインは容量が余っているので，容量を増やしても影響しないことで理解できる．これらの概念を説明するために，ET 社問題の労働量を 61 時間減らして 59 時間で解いてみよう．定式化は次のようになる．：

```
MAX = 20 * A + 30 * C + 47 * V;
          A              + V <= 60;
                 C          <= 50;
          A  + 2 * C  + 3 * V <= 59;
```

解は次の通りである．：

```
Optimal solution found at step:         1
Objective value:                 1180.000
Variable           Value        Reduced Cost
```

A	59.00000	0.0000000
C	0.0000000	10.00000
V	0.0000000	13.00000

Row	Slack or Surplus	Dual Price
1	1180.000	1.000000
2	1.000000	0.0000000
3	50.00000	0.0000000
4	0.0000000	20.00000

Ranges in which the basis is unchanged:

Objective Coefficient Ranges

Variable	Current Coefficient	Allowable Increase	Allowable Decrease
A	20.00000	INFINITY	4.333333
C	30.00000	10.00000	INFINITY
V	47.00000	13.00000	INFINITY

Right-hand Side Ranges

Row	Current RHS	Allowable Increase	Allowable Decrease
2	60.00000	INFINITY	1.000000
3	50.00000	INFINITY	50.00000
4	59.00000	1.000000	59.00000

まず第 1 に，労働制約の右辺定数項が 60 以上減ると，双対価格と減少費用の多くは変化する．特に，双対価格は$20/時間である．これは，労働があと 1 時間追加されると，20 ドルの Astro をあと 1 個作るのに利用できるからである．このようにして，労働の限界価値は，次のように変化することに注意する．

利用可能労働	双対価格	**理由**
0 から	20 ドル/時	1 時間追加されるごとに，それは
60 時間まで		20 ドルの Astro1 台の生産にあてられる．
60 から	15 ドル/時	追加される各 1 時間は，30 ドル
160 時間まで		の Cosmo 半台の生産にあてられる．
160 から	$13．5/時	Astro の半台をあきらめ，利益(-20 + 47)/2
280 時間まで		をもつ V 半台の生産にあてられる．
280 時間以上	$0	追加される労働資源は利用価値がない．

一般的にいって，どの制約に関する双対価格でも，上のように段階的に減少する．**図 3.1** と **3.2** は，総利益が一つの目的関数の係数か一つの右辺定数項を変えることで，どう影響を受けるかを示す．読者は，最大問題で次の点に注意しよう．

①1 個の目的関数の係数の関数の最適解の総利益は，常にボウル状の形を持つ．数学者はそれを凸関数と言う．

②1 個の右辺定数項の係数の関数の最適解の総利益は，常に逆のボウル状の形を持つ．数学者はそれを凹関数と言う．

この問題では，図 3.1 と 3.2 の場合のように，我々はボウルの半分を見るだけである． 最小化問題では，①と②は単に逆になる．

特定の目的関数の係数または右辺定数項の値でモデルを解くと，これらの曲線のうちの一つの点を得る．範囲報告は，この 1 点がある線分の端点を与える．

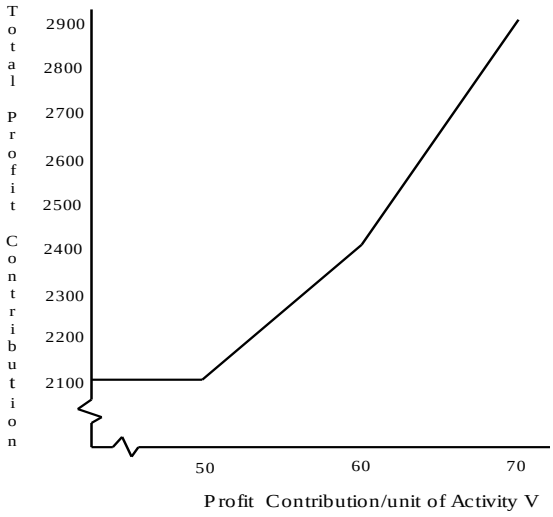


図 3.1 （総利益） vs. （活動 V の利益貢献/単位）

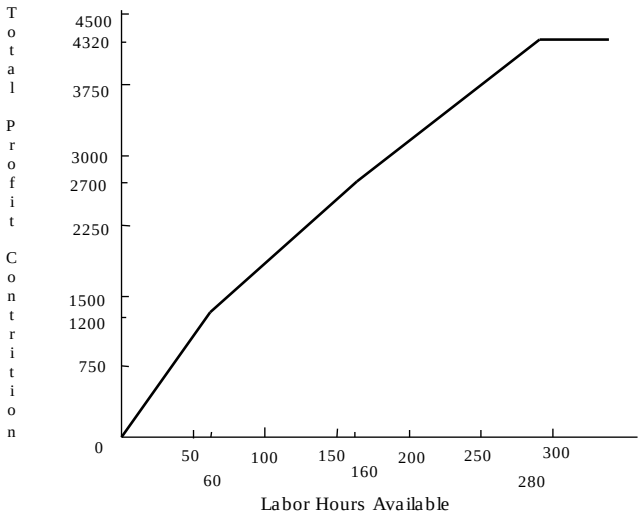


図 3.2 （利益） vs. （利用可能労働量）

3.3.1 パラメータの同時変化の影響の予測-100%ルール

範囲分析の示す情報は，1 つの費用係数（C_j），もしくは資源係数（B_i）が変化したときの効果を表わす．ET 社問題の範囲分析を例にして示すと，次のようである．

Ranges in which the basis is unchanged:

Objective Coefficient Ranges

	Current	Allowable	Allowable
Variable	Coefficient	Increase	Decrease
A	20.00000	INFINITY	5.000000
C	30.00000	10.00000	30.00000

Right-hand Side Ranges

Row	Current	Allowable	Allowable
	RHS	Increase	Decrease
2	60.00000	60.00000	40.00000
3	50.00000	INFINITY	20.00000
4	120.0000	40.00000	60.00000

これによると，Astro の利益寄与（C_j）は，単位あたり 5 ドル以内まで減らしても，基底は変化しない．この場合，最適解は，20 台の Astro と 30 台の Cosmo を生産することを示す．ここで市場競争に勝つため，Astro の価格を 3 ドル/台，Cosmo の価格を 10 ドル/台減少させてみよう．この場合，Astro と Cosmo を同じ混合率で生産するのが有益であろうか？　それぞれの利益寄与を単独に変えるのであれば，　3≦5 であり，10≦30 であるから，これらの変化は解になんら変化をもたらさないであろう．しかし，これらの変化が同時になされると，話は別である．直感的に考えた場合，基底変化をおこさないパラメータの同時変化のルールは，何かないであろうか？

100％ルール：

ここで，許容される変化の範囲はスラックと考えられる．そして，そのスラック分だけ，パラメータが変化するものとする．**もし，スラック変化の比率和が 100 ％以下であれば，どんな変化の組み合わせも基底を変えないというのが事実である．**この場合，同時変化について，次のことがいえる．

(3/5)×100 ＋(10/30)×100 ＝ 60%＋33.3% ＝ 93.3% ＜ 100 %

これは上の条件を満足すので，これらの変化は基底変化を引き起こさない．

Bradley, Hax, Magnanti (1977) は，このルールを 100 %ルールと名付けている．ここで，ＡとＣの最適解における値は変化しないから，これらの利益変化が，最終利益に与える影響を計算すると，

-3 ×60 - 10×30 ＝-480

であり，新しい利益値は，2100 - 480 = 1620 である．

変更した定式化と解と出力は次の通りになる．：

```
MAX = 17 * A + 20 * C;
          A          <= 60;
                  C <= 50;
          A +  2 * C <= 120;
Optimal solution found at step:         1
Objective value:                 1620.000
Variable           Value        Reduced Cost
       A         60.00000           0.0000000
       C         30.00000           0.0000000
```

Row	Slack or Surplus	Dual Price
1	1620.000	1.000000
2	0.0000000	7.000000
3	20.00000	0.0000000
4	0.0000000	10.00000

3.4 制約係数の感度分析

右辺定数項（B_i），および目的関数の係数（C_j）の感度分析は，これらの係数の変化が，適度の範囲内であれば，目的関数の値は線形的に変化するから，比較的理解しやすい．残念ながら，制約式内の係数（A_{ij}）の変化については，目的関数の値は，非線形的に変化する．しかし，これらの制約式内の係数の小さな変動効果を近似する簡単な式がある．ここで，LP で第 i 番目の行の変数 j の係数を少量 e だけ減少した時の効果を調べてみよう．その時の算定式は，次のようである．

目的関数の値の改善分 ≒（変数 j の値）×（第 i 行の双対価格）× e

例)　例えば，次の問題を考えよう．

```
MAX = 20 * A + 30 * C;
A <= 65;
C <= 50;
A + 2 * C <= 115;
```

解は，次のようになる：

```
Optimal solution found at step:         1
Objective value:                 2050.000
```

Variable	Value	Reduced Cost
A	65.00000	0.0000000
C	25.00000	0.0000000

Row	Slack or Surplus	Dual Price
1	2050.000	1.000000
2	0.0000000	5.000000
3	25.00000	0.0000000
4	0.0000000	15.00000

ここで，第 4 行の変数Ｃの係数が，2 ではなくて，2.01 のはずであると発見したとしよう．上の式を使えば，目的関数の値は，次のように近似的に減ることになる．

25×15×0.01=3.75

この変更した問題を解いてみると，実際の目的関数の値は 2046．269 になる．従って，実際の目的関数値の減少分は，3.731 になる．

この制約係数の小さな変化の効果を表わす式は，次のように考えると，意味がよく分かる．もし，係数変化が小さければ，全ての変数や双対価格は，本質的には変化しないはずである．従って，この係数が 2 から 2.01 に変化するということは，実際には労働の必要量が 25×0．1 だけ増えるということになる．そこで，実質的な効果としては，25×0.01 時

間だけ利用できる労働時間が減ったことに相当する．しかし，労働の価値は 15 ドル/時間であるから，利益の変更分は，25×0.01×15 になる．そして，この値は前の簡便な算定式による値と一致する．

この種の感度分析は，どの係数を正確に推定すべきかを発見するための良い指針になる．もし，変数 j の値と i 行の双対価格を掛け算した値が，比較的大きいとき，第 i 行の変数 j の係数を正確に推定しないと，総利益の正確な推定が得られないことになる．

3.5 双対 LP 問題あるいは土地の貸し手と借りて

色々な問題を定式化していて，外見の異なる定式化が同じ問題であることを発見することがある．各々の定式化は正しくて，問題を異なる視点から見たに過ぎない．LP 問題で，数学的に関心を引く事実は，同じ問題に 2 つの定式化が常にあるということである．1 つの定式化は「**主問題**」と呼ばれ，他は「**双対問題**」と呼ばれる．2 つの異なる定式化は，問題の 2 つの異なる視点に起因する．これらの 2 つの視点を「借り手と貸し手」と考えればよい．

以下の状況を考えよう．あるイタリアの織物会社は，自身の製造施設をもたないで，単に必要に応じて適当な器材を所有する会社から借りる．アメリカでは，類似した状況は，製品のリサイクルで存在する．古い電話ケーブルをリサイクルする会社は，単に銅を絶縁物から切り離すために必要な機械を賃借するかもしれない．この賃貸プロセスは，時々「tolling」と呼ばれている．有名ブランドの香水業者は，自身の製造施設を所有しないものが多く，香水を必要に応じて作るため特定の化学会社に使用料を払って製造施設を借りる．この産業の基本的な特徴は，製造設備の所有者は原料や完製品を決して所有しないということである．

今，Astro とCosmo とビデオを製造するET 社から，製造設備を借りて製品を生産することにした． 3 つの資源（Astro とCosmo の生産ライン，労働力）のために，ET 社に支払う 1 時間ごとの料金を決める必要がある．これらの 3 つの 1 時間ごとの料金が決定変数である．実際 3 つの資源の全てを賃借したいと思う．例えば，全資源（60，50，120）の賃借料金を最小にしたい．あなたの申し込みが成功する条件は，あなたの賃貸料金が十分に高いので，ET 社は製品のどれも生産する必要がないことである．賃貸条件の方が自社生産より良いことが，制約になる．

問題を定式化するために，次のような決定変数がいる．

PA = Astro の生産ラインに対する賃貸料（価格/単位）

PC = Cosmo の生産ラインに対する賃貸料（価格/単位）

PL = 労働力に対する対する賃貸料（価格/単位）

双対問題は，次の通りである．

```
MIN = 60 * PA + 50 * PC + 120 * PL;
!ASTRO; PA + PL > 20;
!COSMO; PC + 2*PL > 30;
```

```
!VR;    PA + 3 * PL > 47;
```

3つの制約は賃貸料が十分高いので，ET社は自社生産する必要がない．解は次の通りである：

```
Optimal solution found at step:         2
Objective value:                 2100.000
Variable           Value        Reduced Cost
      PA        5.000000           0.0000000
      PC       0.0000000            20.00000
      PL        15.00000           0.0000000
     Row    Slack or Surplus      Dual Price
       1        2100.000            1.000000
       2       0.0000000           -60.00000
       3       0.0000000           -30.00000
       4        3.000000           0.0000000
```

主問題は，次の通りであった：

```
MAX = 20 * A + 30 * C + 47 * V;
A +             V <=  60;
       C          <=  50;
A + 2 * C + 3 * V <= 120;
```

解は次の通りである．

```
Optimal solution found at step:         1
Objective value:                 2100.000
Variable           Value        Reduced Cost
       A        60.00000           0.0000000
       C        30.00000           0.0000000
       V       0.0000000            3.000000
     Row    Slack or Surplus      Dual Price
       1        2100.000            1.000000
       2       0.0000000            5.000000
       3        20.00000           0.0000000
       4       0.0000000            15.00000
```

価格と決定変数が逆なことを除いて，2つの解が本質的に同じである．特に貸借人が払う価格は，ET 社の最初のモデル(主問題)の利益貢献と同じものである．賃貸料の最小化は，元の利益最大化モデルの双対問題という．ただし，どの資源制約も守られる条件で2つの解の間の同等性が常に保たれる．

なぜ，双対モデルが重要か？ **LP の計算時間は，ほぼ m^2*n と比例す**る．ここで，m = 行数，n = 列数である．双対モデルの行数が主モデルの行数より小さいなら，双対モデルを解くほうがいい．さらに(x ≤ 1)のような単純な制約は，任意の制約より簡単に計算できる．双対モデルが少数の任意の制約だけを含む場合，例えそれが多数の単純な制約を持つとしても，解くことはより簡単である．

双対価格は双対問題における決定変数の減少費用に対応している．

我々は，次のように双対問題についての考えをまとめることができる．元の主問題が「≦」制約で最大化問題であれば，双対問題では「≧」制約で最小化問題になる．双対問題では，主問題の 1 つの制約式は，決定変数になる．そして，主問題の 1 個の決定変数は，1 個の制約式になる．双対問題の右辺定数項が k なら，主問題の目的関数の係数になる．同様に，双対問題の i 行 j 列の係数は，主問題の j 行 i 列の係数になる．全ての制約を同じ種類に変えるため，次の 2 点に注意して，上の方法を適用しなさい．

② 制約式 2x + 3y = 5 は 2x + 3y ≥ 5 と 2x + 3y ≤ 5 に変換する．

② 制約式 2x + 3y ≥ 5 は −2x − 3y ≤ −5 に変換する．

例題：次の問題の双対問題を作成しなさい．

Maximize 4x - 2y

subject to

2x + 6y ≤ 12

3x − 2y = 1

4x + 2y ≥ 5

上の(1)と(2) を用い，次のように書き換える．

Maximize 4x - 2y

subject to

2x + 6y ≤ 12

3x - 2y ≤ l

-3x + 2y ≤ −l

-4x - 2y ≤ -5

4 つの制約式に対応して，次の 4 個の双対変数 r， s， t， u を用い，双対問題を作成する．

Minimize 12r + s − t − 5u

subject to 2r + 3s − 3t − 4u ≥ 4

6r − 2s + 2t − 2u ≥ −2

第 4 章 モデルの定式化

Count what is countable, measure what is measurable, and what is not measurable, make measurable.
Galileo Galilei(1564-1642)

4.1　モデルの定式化の一般論

問題を数理計画法モデルで解決する場合，次の主な 5 ステップがある．

① 現実の問題を理解する．
② 問題を定式化する．
③ モデルのための入力データを収集生成する．例えば使用する 1 単位当りの費用．
④ PC でモデルを解く．
④ 解を現実の世界に組み入れ解釈する．

一般的には，5 つのステップで数回の繰り返しが必要である．最初から，適切なモデルは作成できない．上の中で最も簡単なのは，PC でモデルを解くことである．これが本質的に簡単であるからではなく，数学的に分析方法が確立し LINGO のようなソルバーが準備されているからである．ステップ①，③，⑤は難しいものではないが，少なくとも時間を使う．これらのステップで成功するには，対象の問題に対する深い知識がいる．特に，ステップ②には最も熟練を必要とする．

良いモデルを定式化することは，科学における 1 つの芸術である．なぜなら，これは常に現実世界への接近を含んでいるからである．芸術的能力とは，現実を良く捉えている簡単なモデルを作成することである．私たちは，モデルで現実に良い接近をするために，幾つかの問題の分類があることを知るだろう．

上述のコメントを胸にきざみこんで，これ以降ではモデルの定式化に対する議論に最も多くのページをさきたい．つまり，どのような普遍的な真理がステップ③⑤に適用されているかを述べ，ステップ④のアルゴリズムの導入にしたい．

4.2　モデルの定式化の 2 つの方法

モデルの定式化には，次の 2 つの方法がある．

② コンストラクティブ・アプローチ（**分析的手法**）
②テンプレート（雛型）・アプローチ（**雛形モデルの利用**）

分析的手法は基本的で一般的であるが，経験の少ない初心者は雛形モデルの利用が好ましい．後者は，缶詰めにしたモデルを利用するアプローチである．このアプローチでは，アプリケーションの標準的な例題が示される．もしこれらの**雛形モデルに似た問題であれば，状況にあわせてこれを修正することで必要なモデルを作成できる**．このアプ

ローチの利点は，**実際の状況によく合う雛形モデルさえあれば，技術的な能力をあまり必要としない**ことである．

4.3　雛形モデル

もし，あなたが自分の出会う新たな問題に対して 1 つの分類の基準をもっているなら，あなたのモデル化の能力はより安心と確信を感じるかも知れない．本章では異なった種類の問題についての分類を示す．実際には，大きな現実の問題は一つの分類には属さず，幾つかのモデルの組合せになるであろう．分類は完全なものではないから，どの分類にも属さないモデルに出あったり，モデルを新規に開発する必要があるかもしれない．

4.3.1　製品混合問題

これらは，典型的な LP の導入テキストにある種類の問題である．実際には，この様な簡単な形式の問題に出会うことはめったにない．販売可能な製品の集合とこれらの製品を作るための有限な資源の集合がある．各製品に対して，利益貢献率と資源使用率の集合がある．目的関数は，利用可能な資源量以上に資源を使わずに利益を最大化するような製品（各製品の総計）の配合を見つけることである．条件は，使用可能量以上の資源を使わないことである．これらの問題はつねに，「**ある制約条件以下になるように利益を最大化せよ**」という形式になる．

4.3.2　カバーリング，人員配置，分割問題

これらの問題は，製品混合問題(Product Mix)を補うものである．すなわち，「ある制約条件以上になるように費用を最小化せよ」という形式をとる．この問題における変数は，例えば，その日の様々な交代に対して雇う人数かもしれない．制約は，選ばれた変数の配合がその日の各時間内に必要とされる人数を「カバー」する，という事実から生じてくる．

4.3.3 配合問題 (Blending)

この種の問題は，食品，配合飼料，石油，製鉄，アルミ製造産業等において生じる．この問題は，例えば，異なった種類の肉やシリアルや穀物や原油などの原料の集まりを，ソーセージやドックフードやガソリンなどのような最終製品に仕上げる．最終製品 1 単位当りの費用が，ある質的制約式（例えば，タンパク質の含有率≧15%）を満足しながら最小化するように配合することである．

訳注:　**非常に簡単な LP 問題であり，繰り返し原材料費の最小化を図れるので効果は大きい．しかし，製鉄やアルミ製造産業等では，一部の企業しかこの恩恵を享受していないようである．**

4.3.4 多期計画問題

この問題は，おそらく最も重要な種類の問題の一つである．これらのモデルは，今期なした決定が，将来においてどの決定が妥当であるかを部分的に決定することを考慮している．各期間に用いられるサブモデルは，製品混合問題や他の種類の問題かもしれない．これらのサブモデルは，原料，最終製品，現金，未払い貸付金などの在庫変数で結ばれる．そして，ある期間から次の期間へ繰り越される．

訳注： Excel のアドイン・ソルバーの WB!を使う場合，製品混合，配合問題の個別モデルを Excel の各シートで作成する．次に，これらを結合していけば，大きなシステムになる．LINGO では，個別モデルを作成し，その後でこれらを SUBMODEL 節にして，CALC 節で制御すればよい．

4.3.5 ネットワーク，分配，PERT／CPM モデル

ネットワーク型の LP モデルは，グラフやネットワークで簡単に描けるような形式である．従って，これらは説明や理解が容易である．ネットワーク型の LP は，しばしば生産分配の問題から生じる．幾つかの場所で製品を生産し，それを多くの得意先に分配するようなどんな企業でも，ネットワークに関連した問題を見つけるだろう．特別な注目をネットワーク型の LP に払う主な理由は，これらに対して特殊化した解法が存在する．大きな問題で，これらの特殊な解法は，一般的な解法よりも著しく早いかもしれない．最も簡単なネットワーク問題の 1 つは，ネットワークにおけるある 1 つの点から他の点への最短経路を見つける問題である．この問題の少し違った変形（すなわち，最長経路を見つける）は，プロジェクト管理ツールの PERT（Program Evaluation and Review Technique）や CPM（Critical Path Method）で重要な役割を果すことがある．

訳注： PERT の専用システムは，非常に高価である．しかし，LINGO にある PERT の幾つかのサンプル・モデルを利用すれば費用負担がない．あるいは，それらを基本仕様と考えて，自社で LINDO API を用いて，C 言語でシステム開発すればよい．最適化システムの開発は，通常のシステム開発の方法を踏襲してはいけない．LINGO や WB!で分析したいモデルを検証し，その後でシステム化すべきである．ある電力会社で Fortran で開発された最適化システムの更新に，筆者が在職していた企業の他部門がトラブっていた．そこで LP 部分を全て LINDO に置き換えて，無事更新した．

大企業で，あるプロセスや部門の出力が他のプロセスや部門の入力になるかもしれない．例えば，General Motors 社は，ある工場でエンジンを作っている．これらのエンジンは，他の得意先に販売されるか，自社の車やトラックに使われるかもしれない．これを「垂直統合」と呼ぶ．垂直統合モデルでは，各中間製品に対して 1 つの制約がある．この制約式（数学的に様々なプロセスで使われた中間製品の総計）は，この製品が他の

プロセスで生産した総計を越えることはできない，という基本的な物理法則である．たいてい，利用可能なプロセスの各種類に対して 1 つの決定変数がある．

もし，ある人が自分の見通しを完全経済に広げるならば，考えるべきモデルは，Wassily Leontief(1951)の「**投入／産出モデル**」に似たものになる．各工業は必要とされる投入製品と，生産される産出製品によって記述される．これらの産出は，今度は他の工業に対する投入になるかもしれない．この問題は，各工業が特定の消費物を満たすように操作すべき水準を決定することである．

4.3.6 ランダムな要素を含んだ多期計画問題

最適化モデルの基本的な仮定の 1 つは，全ての入力データが確実に知られているということである．しかし，ある重要なデータがきわめてランダムに発生するような状況がある．例えば，ある石油会社が，来たる冬に対して燃料油の生産計画の決定をするとき，その燃料油に対する需要は非常にランダムな変数である．しかし，全てのランダム変数に対して確率分布が知られているなら，ランダム要素を除いた最適化問題を（おそらく大きくはなるだろうが）それと等価な決定型の最適化問題に変更するモデリング技術がある．ゲーム理論は，競争状態の分析に関係する．これの最も簡単な形式は，ゲームが 2 人のプレイヤーで構成され，各人は自分たちにとって実現可能な決定の集合がある．各プレイヤーは，他のプレイヤーの選択を知らずに意思決定のための戦略を選択する．意思決定の後，各プレイヤーは，どの決定の組合せがなされたかによって左右されるペイオフを受け取る．各プレイヤーの最適戦略を決定する問題は，LP で定式化される．

4.3.7　ポートフォリオ・モデル

近年の最適化の重要な応用は，金融投資における有価証券の設計である．簡単にいえば，高い期待利益と低いリスクを実現するため，株などのリスク商品をどのような割合で投資するかということである．この考えのより複雑な応用は，S&P 500 のような普及したインデックスへの投資である．この分野の多くのサンプル・モデルが開発されている．

4.3.8 ゲーム理論

ゲーム理論は，競争状態の分析を扱う．この最も簡単な形は，ゲームが 2 人のプレイヤーで構成され，各人は自分達にとって実現可能な決定の集合をもっている．各プレイヤーは，他のプレイヤーの選択を知らずに意思決定のための戦略を選択する．意思決定の後，各プレイヤーはどの決定の組み合わせをしたかによって左右される利得を受け取る．各プレイヤーの最適戦略を決定する問題は，LP で定式化される．

出会う全てのモデルが上のカテゴリーの一つに当てはまるわけではない．多くの問題は，上の種類の組み合わせになる．例えば，多期間問題は一つの部分問題が混合や配合問題になるかもしれない．

4.4　分析的手法によるモデルの定式化

分析的手法によるアプローチは，一組のモデルを全く新しく組み立てるための指針である．このアプローチは，少し分析的な技術を必要とするが，モデルの状況に何らかの規則が適用される．現実の状況に正確に合致する雛形モデルを見つける確率は低い．実際，この 2 つのアプローチの組合せは重要である．分析的手法によるアプローチのため，私達は Sam Savage に感謝し，モデル化の ABC ステップと呼ばれモデル組み立ての次の 3 段階のアプローチを提案する．

① 決定変数（WB!では修正可能な決定変数セル）を識別し定義する．決定変数の定義は，測定単位(例えば，トン，時間，等)を含めて行う．決定変数を推論する一つの方法は，次のような質問をすることである．何がこの問題の解になるのだろうか？例えば，各製品の生産量や，農産物の使用量などである．
② 「最適な尺度」をどのようにするかを定義する．すなわち，測定単位を含む決定変数で目的関数を定義する．実行可能な解の空間で，好みか長所(例えば，利益)を測定する．
③ 測定単位を含む決定変数で，制約式を表す．制約式を考える方法は次の通りである．問題の意図した解が与えられたとして，どんな数値点検が解の妥当性を点検するために必要か．

訳注:WB!では，この ABC ステップでモデル作成を行っている．即ち決定変数を表す Adjustable（修正可能）セルを定義する．その後で最適化する目的関数を表す Best セルを決める．最後に，修正可能セルを使って，Constraints（制約）セルを定義する．そして Solve で解を求める手順である．

ほとんどの問題の制約式の大半は，「**SOURCE = USE（搬入量 = 搬出量）**」制約式で考えることができる．もう一つの共通の種類の制約式は，会計制約式である．時々これらの区別は任意である．次の生産の設定を考慮する．ある商品の①初期在庫，②その商品の幾つを生産するか，③商品の販売量，および④期末在庫，である．「搬入量 = 搬出量」制約式は次のように表される．

初期在庫 + 生産量 = 販売量 + 期末在庫

期末在庫の定義は，上の制約式から

期末在庫 =（初期在庫 + 生産量）- 販売量

になる．実際 2 つの式は数学的に等しい．

これらの考え方を説明するために，次の例を考えよう．

4.4.1 例題

Deglo 玩具は，子供用の精密ブロックを製造している．Deglo 社は，暗い中でも光るブロックの生産のため，古い製品の在庫を減らす必要がある．古い製品在庫は，19,900 個の 4 つの窪みブロックおよび 29,700 個の 8 つの窪みブロックである．これらの在庫は，マスタービルダーとエンパイアービルダーという 2 つのキットの形で在庫を一掃できる．目的関数は，この 2 つのキットの販売からの収入を最大にすることである．マスタービルダーは$16.95 で販売し，エンパイアービルダーは$24.95 で販売する．マスタービルダーは 40 個の 8 窪みのブロックと 30 個の 4 窪みのブロックで構成される．エンパイアービルダーは 85 個の 8 窪みのブロックと 40 個の 4 窪みのブロックで構成される．これをモデル化してみよう．

4.4.2 例題の定式化

例題は，次のように ABC ステップで定式化できる．

① 決定変数(Adjustable)は:
M = マスタービルダーの製造個数
E = エンパイアービルダーの製造個数
② 目的関数(Object)は売り上げの最大化(Max = 16.95M + 24.95E).
③ 制約式(Constraints)を決定変数 M と E で記述する.
4 窪みのブロックの使用数 ≤ 19,900 ;
8 窪みのブロックの使用数 ≤ 29,700 ;

記号で表すと次のようになる.

$$30M + 40E \leq 19,900$$
$$40M + 85E \leq 29,700$$

ABC で完成した LINGO モデルは，次のようになる.

```
MAX = 16.95 * M + 24.95 * E;
      30 * M    + 40 * E <= 19900;
      40 * M    + 85 * E <= 29700;
```

解は次の通りである.530 個のマスタービルダーと 100 個のエンタプライズビルダーを作ることになる.

```
Optimal solution found at step:          0
Objective value:                  11478.50
Variable           Value        Reduced Cost
       M        530.0000           0.0000000
       E        100.0000           0.0000000
```

Row	Slack or Surplus	Dual Price
1	11478.50	1.000000
2	0.0000000	0.4660526
3	0.0000000	0.7421052E-01

4.5 費用を正しく選ぶ

目的関数の費用や利益貢献を表す係数を選ぶ際，注意が必要である．多くの会社では，費用データは最適化モデルで必要とされる詳しい水準で利用できないかもしれない．利用できたとしても，モデルの費用係数はこのモデルに不適当かもしれない．基本ルールはかなり簡単である．決定変数の費用係数は，決定変数の変更による総額の変更の率を表すべきである．私達はこの規則に違反する様々な誘惑を議論する．2つの主要な誘惑は，埋没費用と変動費用である．

4.5.1 埋没費用 対 変動費用

埋没費用は，必ずしも支払われていなくても，既に決まっているか約束した費用である．変動費用は，活動水準とともに変わる費用である．埋没費用は，決定変数のどの係数にも用いるべきでない．埋没費用か変動費用かは，決定変数の計画対象期間の長さにも密接に左右される．一般ルールは，次の通りである．短期的な場合は全ての費用は埋没費用で，長期間の場合は全ての費用は変動費用である．次の例で説明する．

埋没費用　対　変動費用の例：

ある会社の製品 X と Y の利益貢献表は次の通りである．

製品:	X	Y
販売価格/単位	$1000	$1000
材料費/単位	$200	$300
労働費/単位	**$495**	**$300**
利益	$305	$400

この 2 製品は，ある共通の組み立て設備を 80 単位使用する．製品 X は 40 単位および Y は 60 単位以下に制限されている．労働者の時給は$15/時間である．モデルは次の通りである．

```
Max = 305 * X + 400 * Y;
      X          <= 40;
                 Y <= 60;
      X    +     Y <= 80;
```

Global optimal solution found.

Objective value: 30100.00

Variable	**Value**	**Reduced Cost**

X	**20.00000**	**0.000000**
Y	**60.00000**	**0.000000**

Row	**Slack or Surplus**	**Dual Price**
1	**30100.00**	**1.000000**
2	**20.00000**	**0.000000**
3	**0.000000**	**95.00000**
4	**0.000000**	**305.0000**

解は 20 個の X と 60 個の Y を作ることである．$15/時間の労働費で，この解で要求される総労働時間は 20*495/15 + 60*300/15 = 1860/時間である．

次に，上記の状態に可能な追加や変更を加える．ある自動車会社では，例えば，1 年間に一定の従業員に仕事を保証する労働契約がある．労働契約に署名する前に，何人従業員を雇うかを決めるために上のモデルを使用すると，$15/時間の労働費は余りに低いかもしれないと考えるようになった．米国では，雇用者は労働手形に 8%に近い社会保障および医療保障税を支払わなければならない．さらに雇用者は従業員の健康保険の費用を支払うので，労働費用は 1 時間あたり$15/時間よりも$20/時間に近くなるかもしれないと考えた．

しかし契約が署名されれば，人件費は埋没費用になる．1 日あたり労働に 1860 時間使用してもいい制約を有する利益貢献表は次の通りである．

製品:	X	Y
販売価格/単位	$1000	$1000
材料費/単位	**$200**	**$300**
総利益	$800	$700

X の方が利益のある製品になった． X を 40 個と Y を 40 個作るという結論に行く前に，労働制約が 1860 時間に固定されていることを思い出すのが重要である．短期間の正しいモデルは，次の通りである．

```
Max = 800 * X + 700 * Y;
          X           <= 40;
                    Y <= 60;
          X       + Y <= 80;
     33 * X +  20 * Y <= 1860;
```

解は次の通りである．

```
Optimal solution found at step:         1
Objective value:   58000.00
Variable            Value        Reduced Cost
       X         20.00000            0.0000000
```

Y	60.00000	0.0000000
Row	Slack or Surplus	Dual Price
1	58000.00	1.000000
2	20.00000	0.0000000
3	0.0000000	215.1515
4	0.0000000	0.0000000
5	0.0000000	24.24242

今，競争があるため x の販売価格が$1000 から$650 に落ちたと仮定する．材料費は$200 であるので，利益は$450 になる．しかし，1 日あたり 1860 時間の労働契約は固定されたままである．これを表す正しいモデルは次の通りである

```
Max = 450 * X + 700 * Y;
          X           <= 40;
                    Y <= 60;
          X         + Y <= 80;
      33 * X    +   20 * Y <= 1860;
```

出力は次の通りである．

```
  Global optimal solution found.
  Objective value:     51000.00
Variable           Value        Reduced Cost
       X        20.00000            0.000000
       Y        60.00000            0.000000
Row    Slack or Surplus      Dual Price
  1        51000.00            1.000000
  2        20.00000            0.000000
  3        0.000000            427.2727
  4        0.000000            0.000000
  5        0.000000            13.63636
```

まだ同じ解（X = 20 と Y = 60)を得るが，モデルを間違って次のようにしたとする．

```
Max = - 45 * X + 400 * Y;
            X           <= 40;
                      Y <= 60;
            X       + Y <= 80;
        33 * X +  20 * Y <= 1860;
```

このモデルは，X を生産しないという解になる．

```
 Global optimal solution found.
Variable           Value        Reduced Cost
```

X	0.000000	45.00000
Y	60.00000	0.000000

Row	Slack or Surplus	Dual Price
1	24000.00	1.000000
2	40.00000	0.000000
3	0.000000	400.0000
4	20.00000	0.000000
5	660.0000	0.000000

上記に類似した多くの計画問題がある．例えば，航空会社かトラック運送会社は，毎日の運行計画モデルの決定と同じモデルを長距間の運行決定のために使用するかもしれない．長期間の運行決定モデルの場合，資本費用は車を持つ費用の毎日の費用に含まれているべきである．一方では，短期間の運行決定モデルの場合，資本費用は車の毎日の費用に含まれているべきでない．但し，使用される車の数は，長期計画で選ばれる運行のサイズ以上であってはいけない．短期モデルを解く場合，車の使用量によって変わる操業費用だけが含まれるべきである．

4.5.2 結合生産

単一プロセスが複数製品を作り出すことを結合生産という（あるいは，一種類の公共財が，異なる種類の複数の便益を産む）．主な特徴はプロセスを動かせば，やむを得ず結合製品のそれぞれの量を得ることである．例は次の通りである．

プロセス	結合製品
原油の蒸留	ガソリン，オイル，燈油，タール
未加工ミルク処理	成分無調整牛乳，上澄みミルク，2%のクリーム，ヨーグルト
肉処理	軽い肉，赤身肉，ステーキ，肩の焼き肉
半導体製造	高速と低速で製造する半導体
貴金属の鉱石	金，銀，銅
訪問販売	種々の製品

税務当局の要請で，費用を出力製品に割振ることを考える．重要な点は，この割振りには明確な目的がないことであり，これは避けるべきである．結合生産をモデル化する適切な方法は，各出力製品毎に別々の決定変数を用いることである．そして結合生産にそれらの決定変数を用いることである．費用および収入は関連する決定変数に加える(蒸留費用は蒸留する原油の量を表す決定変数と関連すべきである）．そしてガソリンを作り出したいと思えば，制約式で蒸留費用を負担する．

例題： 結合費用の例

Chartreuse 社（CC 社)はカボチャを作っている．カボチャ 1 トンを栽培し，収穫し，選別に$800 かかる．CC 社は 150 トンのカボチャを植える．遺伝子工学を用いた CC 社の最善の努力にもかかわらず，収穫したカボチャは「よい，優れた，絶妙」の 3 等級に均等に分類され，市場に出荷するために選別するのに$100/トンを要する．代わりに，3 等級のカボチャを付加的な費用なしで破棄できる．価格は最近落ちてしまったので，3 等級を販売することが有益であるかどうかである．3 等級のトンごとの現在の販売価格は$700， $1100，$2200 である．各等級をどのように扱い販売すべきであるか．適切なモデルは次の通りである．カボチャ 1 トンを栽培し，収穫し，選別に$800 かかるが，これは共通経費と考える．

MAX = (700 - 100)* G + (1100 - 100) * P + (2200 - 100)* E - 800 * R;

R=G+P＋E;

R <= 150;

G <= R/3;

P <= R/3;

E <= R/3;

出力は次の通りである．

Objective value: 65000.0

Variable	Value	Reduced Cost
G	50.00000	0.0000000
P	50.00000	0.0000000
E	50.00000	0.0000000
R	150.0000	0.0000000

Row	Slack or Surplus	Dual Price
1	65000.00	1.000000
2	0.000000	-600.0000
3	0.000000	433.3333
4	0.000000	0.000000
5	0.000000	400.0000
6	0.000000	1500.000

訳注:以下のモデルの説明は少し分りにくい．モデルの簡素化のため，3 等級が 1/3 づつになると仮定しているので，150 トンの生産制約の中で，各等級を 1 トンずつ作る費用を係数として，市場に出す良い，優れた，絶妙の出荷トン数 G， P， E を決めたい．例えば，「優れた」の販売価格は$700 で，市場への選別出荷費用は$100 で，カボチャ 3 トンを栽培し，収穫し，選別に$800 かかるので，$2400 かかる費用を各等級の利益に組み込んでいる．

```
MAX = ( 700-100- 2400/3) * G + (1100-100- 2400/3) * P + (2200-100-2400/3)* E ;
G <= 150/3;
P <= 150/3;
E <= 150/3;
```

出力は次の通りである.

Global optimal solution found.

Objective value: 75000.00

Variable	Value	Reduced Cost
G	0.000000	200.0000
P	50.00000	0.000000
E	50.00000	0.000000

Row	Slack or Surplus	Dual Price
1	75000.00	1.000000
2	50.00000	0.000000
3	0.000000	200.0000
4	0.000000	1300.000

「良い」カボチャは，上記のモデルで負の利益なので，シミュレーションで廃棄することに決める．そこで次に生産費用をＰとＥに割り振ったモデルを考える:

```
MAX = (1100 - 100 - 2400/2) * P + (2200 - 100 - 2400/2) * E;
!G <= 150/3;
P <= 150/2;
E <= 150/2;
```

出力は次の通りである.

```
Objective value:  67500.00
Variable            Value        Reduced Cost
       P         0.000000            200.0000
       E         75.00000            0.000000
Row    Slack or Surplus      Dual Price
  1        67500.00            1.000000
  2        75.00000            0.000000
  3        0.000000            900.0000
```

さて，「優れた」等級は作り出す価値がない．そこで，全費用を絶妙が負担することにすると，これも負になり作る価値がないことになる．従って，私達は有益な事業を始めたのに，共通経費の割振りの盲目的な使用で有益な事業をやめることになった．この物語の教訓は盲目的に原価配分をしないことである.

4.5.3　帳簿価格　対　市場価格

最適化モデルを定式化する際の問題は，費用を在庫製品に付けなければならないことである．典型的な会計システムは，在庫製品に帳簿価格を適用する．在庫製品を使用する費用として，すぐに利用できる数字を使う誘惑にかられる．例えば，ガソリン卸売業者が先月 2．77 ドル/ガロンで 10，000 ガロンの普通ガソリンを買ったとする．そして，今月には，普通ガソリンの在庫が 5，000 ガロンである．現在，普通ガソリンの市価は 2．70 ドル/ガロンに落ちた．そこで，今月あなたは，生産と市場活動を熟考している．今月の普通ガソリンの在庫の使用を幾らにすれば良いだろう？ある人は，購入が埋没費用なので，費用は 0 であると主張するかもしれない．他の人は，帳簿価格（$2．77/ガロン）を請求すると主張するかもしれない．どちらが正しいだろう？単純な速答は，意思決定目的のためであれば，税の計算のために法律で必要な場合を除き，帳簿価格は常に無視するということである．

在庫品を市場で売ることを含み，在庫に関してできる可能な全てのオプションを列挙する．

次の例で詳細を把握し，利用可能な行動のために決定変数を定義することで問題をはっきりさせる．今月あらたに普通ガソリンを PB ガロンだけ買うか，普通ガソリンを$2．70/ガロンで RS だけ販売する．しかし，新規に購入すると，$0．01/ガロンの輸送と業務費用が必要になる．同じように，ガソリンを売るために 0．02 ドル/ガロンの業務費用がかかる．この他の案として，高級ガソリンを同量混合し中級ガソリンを作ることである．あなたは現在，最高 6000 ガロンの中級ガソリンに$2．82/ガロンを払う気がある 1 番目の顧客と，最高 8000 ガロンの中級ガソリンに$2．80/ガロンを払う気がある 2 番目の顧客がいる．高級ガソリンは，無制限に$2．90/ガロンまで RS ガロンを購入する．

次のいずれを選択すべきか．

・何もしない，
・普通ガソリンを市場へ売る，
・高級ガソリンを購入し中級ガソリンに仕立てて顧客 1 あるいは顧客 2 に売る．

最適化の ABC ステップに従い，StepA で次の決定変数を定義する：

・PB=今月市場で購入する普通ガソリンの量，
・RS=今月市場で直接売った普通ガソリンの量，
・RB=購入する高級ガソリン，
・MS1=顧客 1 に売った中級ガソリン，
・MS2=顧客 2 に売った中級ガソリン．

B ステップは，目的関数は（売り上げ － 費用）を最大にすることである．

C ステップは，制約を指定することである．2 つの主な制約は，普通と高級ガソリンの「搬入量 ＝ 搬出量」制約である．公式は，下で示される．1 ガロンの中級ガソリン

は 0．5 ガロンの普通と高級ガソリンを使う．理解がより簡単になるよう，解の報告の制約に[列名]を与える．

```
! （収入- 費用）を最大化;
 MAX = (2.70-.02)*RS+(2.82-.02)*MS1+(2.80-.02)*MS2
     - (2.70 + .01)*PB - (2.90 + .01)*PB;
!レギュラーと高級ガソリンの（購入 = 販売）;
 [SEQUR]     5000 + RB = RS + .5*(MS1 + MS2);
 [SEQUP]     PB =      .5*(MS1 + MS2);
!販売可能な上限;
 [UL1]            MS1 <= 6000;
 [UL2]            MS2 <= 8000;
```

普通ガソリンの在庫には，明確なチャージはない．2．77 ドルの帳簿価格は，公式化のどこにも現れない．在庫は埋没費用または無料の商品とみなされるが，我々は直接それを売るためにオプションに含めた．例えば，中級ガソリンを混合するために普通ガソリンを使うことは，単に直接普通ガソリンを現在の市価で売ることと争わなければならない．解は次の通りである．：

```
 Objective value: 13430.00
   Variable           Value        Reduced Cost
         RS        2000.000            0.000000
        MS1        6000.000            0.000000
        MS2           0.000            0.015000
         RB           0.000            0.030000
         PB        3000.000            0.000000
        Row    Slack or Surplus      Dual Price
          1       13430.000            1.000000
      SEQUR           0.000           -2.680000
      SEQUP           0.000           -2.910000
        UL1           0.000            0.005000
        UL2        8000.000            0.000000
```

市場へ直接売るよりも，普通と高級ガソリンを混ぜ合わせ中級ガソリンとして顧客 1 に売るほうが利益がある．しかし，市場に普通ガソリンを直接売ることは，顧客 2 に中級ガソリンを売るより利益がある．

4.6 一般的な誤り

現実問題を最初に定式化するときは，その定式化には間違いや誤りを含んでいるのが一般的である．これらの誤りは次のように分類される．

① 簡単な印刷上の間違い
② 基本的な定式化の間違い
③ 近似上の間違い

最初の 2 つの間違いは，それを発見すれば修正は簡単である．基本的には，種類 1 の間違いは，本質的には事務的な間違いであるので，発見するのも簡単である．しかし，大規模なモデルの場合には，それを追いかけることはかなり困難になる．種類 2 の間違いは，もう少し基本的なものである．なぜなら，これは現実の問題を間違って理解するか，もしくは LP モデルの性質を間違って理解するかのどちらかである．種類 3 の間違いになるとさらに微妙なものとなる．一般的に，現実水準の LP モデルは，何らかの近似を伴う．例えば，幾つかの製品は，ひとまとめにしてある単一のマクロ製品として扱ったり，一週間の日数を一括したり，費用が生産量に比例しないのにそれを線形として扱う場合が多い．そこでこの種類 3 の間違いを避けるには，どのような近似ができるかを発見する技術が必要になる．

まず第一の種類の間違いを議論しよう．解が得られても，種類 1 のような間違いをすれば途方もない答えがかえってくるので自ずから明らかになる．定式化の誤りは，多くの型があるので議論するのは難しい．初心者がよく犯す間違いは，しばしば次元を分析することで発見できる．物理学や化学の科目をとった人ならば，「単位をチェックする」ことで知られている．例題でこれを説明しよう．

おもちゃ問屋が近づく休暇シーズンに備えて，あるおもちゃを生産する戦略を分析しよう．この生産者は 2 種類のおもちゃ，「Big」と「Tot」を組み立てている．そして，前者は 1 つの製品を作るのに 60 本の棒と 30 個のコネクターが必要であり，後者の製品は 30 本の棒と 20 個のコネクターが必要である．ここで，重要な考察すべき要因として，この季節に備えて生産者は，60,000 個のコネクターと 93,000 本の棒しかもっていない．彼はどちらの製品も，組み立てたならば，全て売却することができる．利益寄与（Cj)は，Big と Tot で 1 単位あたり 5.5 ドルと 3.5 ドルである．彼は利益を最大化するために，それぞれの製品をどれだけ売るべきか？

この定式化で，次の決定変数を考えてみよう．

B＝製品 Big の組み立て個数
T＝製品 Tot の組み立て個数
S＝実際に使われる棒の本数
C＝実際に使われるコネクターの個数

```
MAX = 5.5 * B + 3.5 * T;
B - 30 * C - 60 * S    = 0;
T - 20 * C - 30 * S    = 0;
                    C <= 60000;
                    S <= 93000;
```

最初の２つの制約は次の制約と同じである.

B＝30C＋60S

T＝20C＋30S

あなたがこの定式化に同意するのなら，次の解を分析してみよう.

```
Optimal solution found at step:         0
Objective value:                 0.5455500E+08
Variable           Value        Reduced Cost
       B        7380000.           0.0000000
       T        3990000.           0.0000000
       C        60000.00           0.0000000
       S        93000.00           0.0000000
     Row    Slack or Surplus      Dual Price
       1       0.5455500E+08        1.000000
       2       0.0000000            5.500000
       3       0.0000000            3.500000
       4       0.0000000            235.0000
       5       0.0000000            435.0000
```

ここで定式化に間違いがあることが明白である.なぜならば,この解によれば,93,000個の棒しかないのに製品 Tot を 3,990,000 個も生産できる.これは LP の初心者がよく犯す間違いである．すなわち，ある行動の特徴を制約式で記述しようとする点である.**制約というものは常に，何らかの財の利用量がその財の供給量に等しいか，少ないという記述と考えてよい.**従って，下の２つの制約式はこの性質を持っているけれども，上の２つの制約式はこの性質を持っていない.

そこで，最初の２つの制約式の次元を解析すると，おかしな点が分かる．そこで最初の制約の次元を考えると明らかにこれらは異なる単位になっている.もし２つの項をたし算するときには,それらの単位は同じでなければならない.誰でも知っているように,リンゴの個数とみかんの個数は足してはいけない.ある制約の中におけるそれぞれの項の単位は等しくなければいけない.

語句	単位
B	製品 Big の個数
30c	30（コネクターの個数／Big の個数）×コネクターの個数
60S	60（棒の本数／Big の個数）×棒の本数

そこで，まず問題を言葉で定式化して，それからそれを代数表現に変換すれば，上記のような間違いを避けることができる．言葉でいうならば，我々がやりたいことは,

まず利益寄与を最大化することであり，制約は次のようになる.

コネクターの利用量≦コネクターの供給量

棒の利用量≦棒の供給量

これを代数的に表現すれば，次のようになる．

MAX = 5.5 * B + 3.5 * T;

30 * B + 20 * T <= 60000;

60 * B + 30 * T <= 93000;

この制約　30B+20T ≦ 60000 の項の単位を考えるならば次のようになる．

語句	単位
30B	30（コネクターの個数／Big の個数）× Big の個数＝30 個のコネクター
20T	20（コネクターの個数／Tot の個数）× Tot の個数＝20 個のコネクター
60,000	60,000 個のコネクター

ここで全ての項の単位はコネクターの個数となる．そしてこの問題を解けば，意味のある解が得られる．

```
Optimal solution found at step:         0
Objective value:                 10550.00
Variable           Value        Reduced Cost
       B         200.0000           0.0000000
       T         2700.000           0.0000000
     Row    Slack or Surplus      Dual Price
       1         10550.00           1.000000
       2        0.0000000           0.1500000
       3        0.0000000          0.1666667E-01
```

4.7 非同時による誤り

LP の定式化で全制約が同時に働くことを強調する．従って，全ての制約を同時に満足するような行動水準のうまい組み合わせを見つけなければならない．普通はどちらか一方を満足させるという形で理解しがちであるが，そうではない．ここで，靴下の 1 回分の生産の大きさを B としよう．そうすると，よくある方針としては「もし生産する場合，少なくとも 2 ダース以上は生産したい」というものである．このような場合に B は 0 であるか，あるいは 24 以上の値になるであろう．すると，この方針を次の 2 つの制約で書きたくなる誘惑に陥りがちである．

B ≦ 0

B ≧ 24

ここでは，この 2 つの両方が満たされねばならないので，実行可能解が存在しない．このように，どちらかの制約を満足することが重要な場合には，これは整数計画法に頼らざるをえなくなる．このような定式化については，後で議論する．

第 5 章 集合から見た世界

In Normal form, each attribute/field of an entity/record should depend on the entity key, the whole key, and nothing but the key, so help me Codd.
-anonymous

5.1 はじめに （集合概念は難しいので、初回は読み飛ばす選択肢もある）

LINGO の一番の特色は，巨大なシステムをモデル化する能力である．これを可能にするのが，類似したオブジェクトを集合化するというコンセプトである．現実的な例をモデル化してみると，類似したオブジェクトが複数あることに気付く．例えば，工場，製品，期間，顧客，乗り物，要員などである．**LINGO は，このような類似したオブジェクトを一つのグループにまとめ，集合とする．集合ができると，一つのステートメントで集合に含まれるメンバーそれぞれに適用できる．**

LINGO で作られる大きなシステムは，**①SETS 節，②DATA 節，③モデルの方程式，から成る．SETS 節では，ある種の問題を解く時にデータの構造を説明し，DATA 節でデータを定義する．モデルの方程式部分では，各種データと決定した事項の関係が定義される．**

訳注:複数の最適化モデルを扱ったり，一つのモデルでもループさせたいような最適化モデルの実行には，SUBMODEL 節と CALC 節を用いる．

5.1.1 なぜ集合を使うのか

大規模なモデルでは，非常によく似た計算や制約を幾つも作らなければいけない．LINGO は集合を使い，これらの方程式や制約を効率よく表現できる．例えば，100 ある倉庫の輸送モデルを考える場合，それぞれの制約(例：「倉庫 1 は今ある在庫以上のものを輸送できない，倉庫 2 は今ある在庫以上のものを輸送できない，倉庫 3 は今ある在庫以上のものを輸送できない…」）を書くのは大変面倒な作業である．「各倉庫は今ある在庫以上のものを輸送できない」と一つの式で表す方が便利である．

5.1.2 集合とは？

集合とは，類似したオブジェクトの集まりである．集合は，製品やトラックのリストであったり，要員であったりする．集合に含まれるメンバーには，それぞれ少なくとも一つの関連した特徴（例：重量，価格/単位，収入）がある．これらの特徴を「属性（attribute)」と呼ぶ．同じ集合に含まれるメンバーは，同じ属性(配列で表される)を持つ．LINGO がそのモデルを解くためには，属性の値が前もって分かっていても未知であってもかわない．属性の例として，製品には「価格」があり，トラックには

運搬できる「許容量」がある．そして，雇用する要員には，「給料」や「誕生日」といった属性がある．

5.1.3 集合のタイプ

LINGO には，原始と派生という 2 種類の集合がある．原始集合は，1 次元のオブジェクトから成る 1 次元の集合である．派生集合は，1 次元の集合から作られる **2 次元以上の集合**になる．派生集合は 2 種類あり，a）部分集合の選択，b）デカルト積（クロス乗積，Excel の CROSSPRODUCT）の両方もしくはどちらか一方を用い， **2 次元以上の集合**を作る．派生集合に含まれるメンバーは，既存の他の集合のメンバーでもある．例えば，「WAREHOUSE(倉庫)」と「Customer(顧客)」という 2 つの 1 次元の原始集合がある．そこには，「SHIPLINK(輸送経路)」という派生集合がある．SHIPLINK は，WAREHOUSE と CUSTOMER という 2 つの原始集合のあらゆる組み合わせを表すが，これは他の派生集合から作ることも可能である．

5.2 集合節（SETS Section）

大抵の場合，集合を基盤とした LINGO のモデルでは最初の節が集合節となる．集合節は，「SETS:」（コロンを含む）というキーワードで始まり「ENDSETS」で終わる． しかし，モデルに SETS 節がなくても良い．逆に，複数の SETS 節を持つことも可能である．そして SETS 節は，モデルのどこに置いても問題はないが，重要な規定としてモデル内の制約を参照する前に，必ず集合とそのアトリビュート(属性)を定義する．

5.2.1 原始集合の定義

SETS 節に原始集合を定義するためには：

- 集合の名前
- メンバーの属性（データを格納する配列のこと）

を指定する．シンタックスは次のようになる．

集合名:[属性リスト];

「集合名」とは 顧客が集合につける名前のことである．覚えやすいように説明的な名前のほうが良い．次のように名前をつける．英字（A-Z）で始まり，その後は英数字（1-9）かアンダースコア（ _ ）を使ってもよい，そして 31 字以内でなければならない．LINGO は，大文字と小文字の違いを認識しない．

例：

```
SETS:
WAREHOUSE: CAPACITY;
ENDSETS
```

これは，複数の WAREHOUSE(倉庫)という集合があり，それぞれの倉庫に CAPACITY（許容量）があることを示す．集合内のメンバーには，ゼロから複数の属性があり，それらを「属性リスト」に指定する．属性の名前は LINGO の命名規則に則り，各属性間は，カンマ（,）で区切る．

例えば，WAREHOUSE に場所と船積ドックの数という属性が追加したとする．これらの属性は，次のように加えることができる．

```
WAREHOUSE: CAPACITY, LOCATION,DOCKS;
```

訳注： Fortran などの第 3 世代のプログラミング言語では，配列は脈絡なく定義する．集合 WAREHOUSE が n 個の倉庫を表す場合，各倉庫の容量が CAPACITY という配列で定義し，LINGO の中で配列演算に用いられる．また，CAPACITY， LOCATION， DOCKS という 3 つの配列は，倉庫という集合に関する値を格納するものとしてまとめられる．さらに，集合が n 個の要素を持つ場合，LINGO のループ関数で 3 つの配列を操作計算できる．例えば，$\Sigma_{i=1\cdots n}$ $CAPACITY_i$ などである．これによって大規模な最適化モデルの作成が容易になる．

5.2.2 派生集合の定義

派生集合を定義するには：

- 集合の名前
- 親集合
- メンバーの属性（ある場合）

を指定する．シンタックスは次のようになる．

集合名（親集合のリスト）[メンバーシップ・フィルター] [:属性リスト];

「集合名」は， 顧客が集合につける名前のことである．「メンバーシップ・フィルター」とは，オプションで集合に含まれるメンバーにつける一般条件である．「親集合リスト」はあらかじめ定義した集合で，カンマ（ , ）で区切られている．LINGO は各親集合から全てのメンバーの組み合わせで，派生集合のメンバーを作る．例えば次の Sets 節を考えてみよう．

```
SETS:
PRODUCT ;
MACHINE ;
WEEK;
ALLOWED( PRODUCT,  MACHINE,  WEEK): VOLUME;
ENDSETS
```

PRODUCT, MACHINE, WEEK は 1 次元の原始集合で，ALLOWED は原始集合の PRODUCT, MACHINE, WEEK からの 3 次元の派生集合である．特に定めがない限り，ALLOWED には，上記 3 つの原始集合のメンバー全ての組み合わせが含まれる．属性である 3 次元配列の VOLUME を使うことで，各週どの機械からどの製品を生産するかを特定できる．メンバー全ての組み合わせを含む派生集合を「密な集合（dense set)」と呼ぶ．メンバー

シップ・フィルターかその派生集合のメンバーが明確にDATA節に含まれる場合，その集合は「疎な集合(sparse set)」と呼ぶ.

訳注:3次元の密な集合（dense set）に対して，メンバーシップ・フィルターという条件を付けることで，配列の一部に対して簡単に処理できる．これは，サンプル・モデルなどで実例を見れば，具体的に理解できる.

要は，派生集合のメンバーは下記のどれかから成る.

・DATA節の明確なメンバー・リスト
・メンバーシップ・フィルター
・派生集合のメンバーについて触れない場合は，黙示的に密である

DATA節に含まれる派生集合の明確なメンバーシップリストの仕様は，次の項で説明する．モデルが大きな疎な集合の場合，全てのメンバーをリストするのは困難である．幸いにも多くの疎な集合では，全てのメンバーがメンバーでないものと差別化できるような条件を満たす．この条件を明確にできれば，入力に伴う労力を大幅に減らすことができる．これがまさにメンバーシップ・フィルターの役目である．メンバーシップ・フィルターを使って派生集合のメンバー・リストを定義する方法は，集合それぞれに含まれるメンバーは，それに含まれるために満たさなければならない論理条件がある．いわゆる，この論理条件が基準に達していないメンバーをフィルターにかける役目をする.

例えば，TRUCKSという集合を定義し，トラックそれぞれにはCAPACITYという属性（配列）があるとする．そこで，TRUCKSから許容運搬量が50000以上の大きいものだけを集めて派生集合を作りたい場合，条件に合うトラックを明確なメンバー・リストにしても良いが，メンバーシップ・フィルターを使うと次のような簡単な式になる.

```
HEAVY_DUTY( TRUCKS) | CAPACITY( &1) #GT# 50000;
```

この集合をHEAVY_DUTYとし，親集合のTRUCKSからの派生集合とする．縦棒（|）を使い，メンバーシップ・フィルターの開始ラインとする．このメンバーシップ・フィルターでは，運搬許容量を（CAPACITY(&1)）以上を（#GT#）で表し，運搬許容量が50,000以上のトラックのみをHEAVY_DUTYに入れる．フィルターに含まれる「&1」という記号は，集合インデックスと呼ばれる．メンバーシップ・フィルターを使って派生集合を作る時，LINGOは親集合から全ての組み合わせを作る．これら全ての組み合わせをフィルターにかけ，条件に達するかどうかを確認する．最初の親集合の値が「&1」に入力され，2番目の親集合があれば「&2」に入力され，以下3番，4番も同様に入力される．今回の例では，親集合が一つしかないため，「&2」以降は必要ない.
「#GT#」は論理演算子である．LINGOで認識される論理演算子は以下の通である.

#EQ# or =,　同等
#NE# or ≠,　不等

#GE# or ≧, 以上
#LT# or＜ 未満

5.2.3 まとめ

LINGO は，原始と派生という 2 種類の集合を認識できる．原始集合とは，モデルをそれ以上細かくできない基礎的なオブジェクトを指す．逆に，派生集合は他の集合から取り出して部分集合を構成することもできる．元となる集合を派生集合の親と呼び，原始集合または派生集合から構成されている．派生集合は，疎または密に分かれる．密な集合は親集合の組み合わせが全て含まれる（デカルト積または親集合との交差とも呼ばれる．Excel の SUMPRUDUCT)．疎な集合は交差する親集合の一部を部分集合として含み，明示的なリストまたはメンバーシップ・フィルターのどちらかで定義できる．明示的なリストの場合，疎な集合のメンバーを DATA 節にリストする．メンバーシップ・フィルターは，全てのメンバーが満たさなければならない論理的な条件を定め，メンバーをすっきりとリストする．下記の図は，集合の種類の関係を表すグラフである．

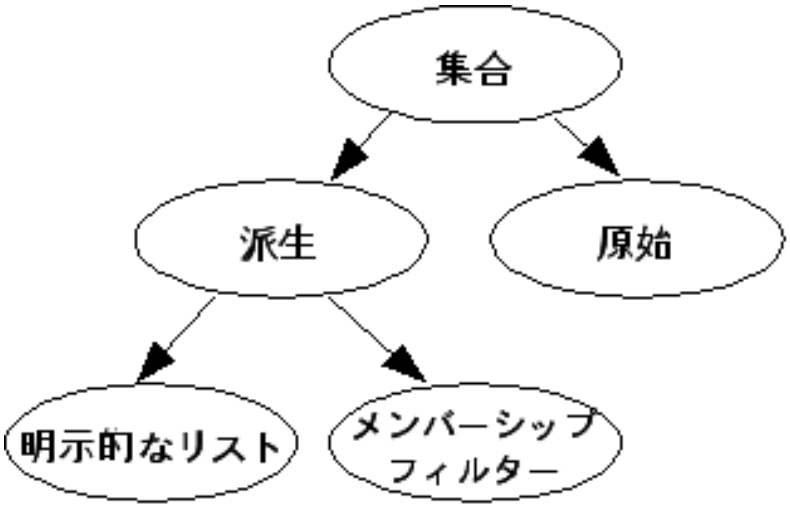


図 5.1 集合の種類

5.3 Data 節

集合節では，ある種類の問題のデータ構造を説明した．DATA 節は，このような問題の特定な事例を作るために，データを提供する．DATA 節は，モデルから変更が生じやすいデータを分離する．データを分離することで，モデルの保守が簡単になり，モデルを拡大・縮小するのが便利になる．

訳注:　汎用統計ソフトは，データと統計手法を分離することで格段に使いやすくなった．第 1 世代のソルバーの代表である IBM の MPSX や LINDO では，数理計画法モデルは具体的な係数値を含んで定義された．このため，同じモデルであっても，週や月次処理をする場合，係数値が異なり，モデルを修正する必要がある．第 2 世代の LINGO では係数が DATA 節で定義し，その配列でモデルが一般的に定義されるので，DATA 節で異なった値を Excel などから読み込むだけで処理ができる．これで，ようやく汎用統計ソフトと同じレベルになった．

大規模なモデルは，a) 集合節，b)Data 節，c)方程式の 3 つの部分に区分するのが良い．モデルを作る人は，この 3 つの節を全て把握する．しかし，開発者がきちんと区分していれば，ユーザーは Data 節で定義された値だけを理解していればよい．

集合節と同様，Data 節は「DATA：」で始まり，「ENDDATA」で終了する．この Data 節に文を入力して定義したい集合の属性やメンバーを初期化することができる．シンタックスは下記の通りである．

属性リスト ＝ 値のリスト；

もしくは以下のようになる．

集合の名前 ＝ メンバー・リスト；

「属性リスト」には，初期化したい属性の名前（カンマで分けてもよい）を入力する．もし，複数の属性の名前が文の左側にある場合，全ての属性は同じ集合に関連していなければいけない．「値のリスト」には，属性リストにある属性に割り当てる値（カンマで分けてもよい）を入力する．次の例を見てみましょう．

```
SETS:
SET1: X,  Y;
ENDSETS
DATA:
SET1 = M1,  M2,  M3;
X = 1 2 3;
Y = 4 5 6;
ENDDATA
```

ここには，2 つの属性(配列)X と Y が 1 次元集合 SET1 で定義されている．DATA 節で要素数は **M1, M2, M3 の 3 個であることが分かる．これは単に「SET1 =1...3;」と指定するのが簡単である．** X の値は，1，2，3，Y の値は，4，5，6 である．

訳注:　以下の説明は，はじめは読まないで，5.4 に行って方が良い．

同じモデルを次のように表現することもできる．

```
SETS:
SET1: X, Y;
```

```
ENDSETS
DATA:
SET1 X Y =
M1 1 4
M2 2 5
M3 3 6;
ENDDATA
```

これを見ると，X が 1，4，2 のように見えるが，LINGO は Data 節の値を「行」でなく「列」形式で読むので，X の値は 1，2，3 なる．Data 節は，派生集合のメンバーを特定するのにも使われる．次の例では，どのように Data 節に集合のメンバーを特定するか，またどのように疎な派生集合を特定するかを説明する．必要はないが，この例では VOLUME 属性の値も指定する．

```
SETS:
PRODUCT ;
MACHINE ;
WEEK ;
ALLOWED( PRODUCT,  MACHINE,  WEEK): VOLUME;
ENDSETS
DATA:
PRODUCT = A B;
MACHINE = M N;
WEEK = 1..2;
ALLOWED,  VOLUME =
A M 1 20.5
A N 2 31.3
B N 1 15.8;
ENDDATA
```

ALLOWED には，8 つのメンバー全ては含まれていない．代わりに，(A，M，1)，(A，N，2)と(B，N，1)の 3 つのみが含まれる**疎な集合**になっている．

LINGO は，多数の標準的な集合を認識する．例えば，Data 節に，

```
PRODUCT = 1..5;
```

と指示すると，PRODUCT のメンバーは，1，2，3，4，5 になる.もしくは，

```
PERIOD = Feb..May;
```

と指定すると，PERIOD のメンバーは，Feb，Mar，Apr，May になる．他の例の集合には，mon..sun や thing1..thing12 がある．

Data 節に属性が指定されていない場合は，デフォルトで決定変数となる．LINGO は，モデルの方程式の節にある文と一致する値を属性とする．ここでは，Data 節の使い方について簡単に紹介したが，ここで挙げた例のようにデータを Data 節に置かなくてもよい．Data 節は，Excel の OLE リンク，データベースの ODBC リンクそしてテキストベースのデータファイルを読み込むことができる．例は，後ほど紹介する．LINGO が派生集合を作る時，一番早く増加するのは一番右側の親集合である．

5.4　集合ループ関数

モデルを表す方程式の節は，色々な属性の関係を示す．集合節や Data 節のない文は，デフォルトで方程式節にある．**集合をベースとしたモデル構築の強みは，一つの文を使って，集合内の全てのメンバーに作業を適用する能力**にある．LINGO で，これを可能にする関数を**集合ループ関数**と呼ぶ．この機能を使わないのは，労力の損失である．集合ループ関数を用いると，ある演算を行うため全メンバーに対して反復処理をする．LINGO には現在 4 つのループ関数がある．

関数	用途
@FOR	集合に含まれる全てのメンバーに制約を課す.
@SUM	全ての集合メンバーに対して合計する.
@MIN	全ての集合メンバー上で最小値を計算する
@MAX	全ての集合メンバー上で最大値を計算する.

集合ループ関数のシンタックスは次の通りである．

@ループ関数(集合名[(集合索引リスト) [|条件文の限定詞]] :
式のリスト);

「@ループ関数」には，上記の表から一つを適切なものを選んで入力する．「集合名」とは，ループさせたい集合を入力する．「集合索引リスト」は，オプションで集合名に指定した親集合である原始集合に対応する索引リストである．LINGO は，「集合名」に指定した集合のメンバー間をループしつつ，作られた索引に「集合名」で指定した集合の現メンバーに対応する値を設定する．「条件文の限定詞」もオプションで使えるフィルターで，集合ループ関数の範囲を限定するのに使える．LINGO が「集合名」に指定した集合のメンバーをループする時，条件文の限定詞をチェックする．結果が true の場合，「式のリスト」がそれぞれのメンバーに対して実行される．そうでなければ，省略される．「式のリスト」とは，「集合名」に指定した集合に含まれるメンバーに適用される方程式のリストを指す．@FOR 関数を使う時には，この式のリストにセミコロン（;）で区切られた複数の式が含まれる．これらの式は，モデルの制

約式として追加される．この他の集合ループ関数（@SUM，@MAX，@MIN）を使う時には，式のリストに含まれる式が一つでなければいけない．もし，集合索引リストを省略した場合，索引リストを参照する全ての属性を「集合名」に指定する集合の中に定義する．

5.4.1 @SUM 集合ループ関数

この例では集合ループ関数の一般的な機能と@SUM 関数の説明をするため，@SUM 関数を使って総和を作る．

次のモデルを見てみましょう：

```
SETS:
SET_A : X;
ENDSETS
DATA:
SET_A = A1 A2 A3 A4 A5;
X = 5 1 3 4 6;
ENDDATA
X_SUM = @SUM( SET_A( J): X( J));
```

LINGO は，先ず内部のアキュミュレーターを 0 に初期化する．次に，SET_A のメンバー間をループする．「索引変数 J」は，先ず SET_A の最初のメンバー（例えば A1）を設定し，次に X(A1)をアキュミュレーターに追加する．次に，J を 2 番目の要素に設定し，この過程をＸの値全てがアキュミュレーターに追加されるまで続ける．合計の値は，X_SUM に保存される．

式のリストにある全ての属性（この例の場合，X が式のリストにある）が索引集合(SET_A)に定義されている．このような場合，総和を次のように書くこともできる．

```
X_SUM = @SUM( SET_A: X);
```

この場合，余計な索引集合リストと X の索引がなくなっている．使われる式がこれだけ簡単明瞭な場合，索引リストは「暗示」されているという．このような索引リストは，式のリストにある属性が，違う親集合から参照されている場合は使えない．

次に，属性 X にある要素の最初の 3 つの和を求めたい場合を想定する．集合索引の条件文の限定詞を使って，次のように遂行する．

```
X3_SUM = @SUM( SET_A( J) | J #LE# 3: X( J));
```

#LE#は論理演算子で，左側の集合索引（J）を右側の 3 と比べ，左側の集合索引が右側より以下か等しい場合は「真」，それ以外の場合は「偽」を返す．そして LINGO が和を求める場合，集合の索引変数 J を条件文の限定詞である J#LE#3 に埋め込む．もし真であれば，X(J)が和に加えられる．結果的に，LINGO は X の最初の 3 つの項を足し，4 番目と 5 番目の項を省略し，答えは 9 となる．

この例を終える前に，ちょっとした注意点がある．それは，最後の集合の索引 J が返した値である．J #LE# 3 を使って，索引変数と 3 を比べた．これが意味をもつためには，J は数値でなくてはいけない．集合の索引は，集合のメンバー間をループするのに使われるため，集合索引はただの現集合メンバーの代替物であると考えられる．ある意味そうではあるが，集合索引が返すのは現集合メンバーの親原始集合での「索引」である．言い換えると，最初のメンバーに対する索引は 1 で，2 番目は 2... という値が返される．このように，集合の索引が数値を返すということを考えれば，他の変数と共に算術式に使うことができる．

5.4.2 @MIN と@MAX ループ関数

@MIN と@MAX 関数は，ある集合のメンバーの最小値と最大値を求める時に使う．次のモデルを見てみよう．

```
SETS:
SET_A : X;
ENDSETS
DATA:
SET_A = A1 A2 A3 A4 A5;
X = 5 1 3 4 6;
ENDDATA
```

X の最小値と最大値を求めるには，次の 2 つの式を加えるだけである．

```
THE_MIN_OF_X = @MIN( SET_A( J): X( J));
THE_MAX_OF_X = @MAX( SET_A( J): X( J));
```

上記の例と同様に，索引集合に属性が設定されているため，暗示的な索引リストを使うことができる．暗示的な索引を使うと，式を再計算できる．

```
THE_MIN_OF_X = @MIN( SET_A: X);
THE_MAX_OF_X = @MAX( SET_A: X);
```

どちらにしても，このモデルを解くために，LINGO は X の最小と最大値を返す．

```
変数                值
THE_MIN_OF_X 1.000000
THE_MAX_OF_X 6.000000
```

説明のため，X の最初の 3 つの要素の最小と最大値を求めたいと仮定しよう．@SUM の例の時のように，条件文の限定詞である J #LE# 3 を加えるだけである．

```
THE_MIN_OF_X_3 = @MIN( SET_A( J) | J #LE# 3: X( J));
THE_MAX_OF_X_3 = @MAX( SET_A( J) | J #LE# 3: X( J));
```

解は次にまとめました．

変数	値
THE_MIN_OF_X_3	1

THE_MAX_OF_X_3	5

5.4.3 @FOR 集合ループ関数

@FOR 関数は，特定の集合のメンバーに制約を付ける場合に使われる．スカラーをベースとしたモデル言語は，各制約をそれぞれ明確に入力するのに対し，@FOR 関数は制約を一度入力し，後は LINGO がメンバーそれぞれに対応する制約を生成する．@FOR は集合をベースとしたモデルの強力なツールである．次の例を見てください．

```
SETS:
TRUCKS : HAUL;
ENDSETS
DATA:
TRUCKS = MAC,  PETERBILT,  FORD,  DODGE;
ENDDATA
```

明確に言うと，HAUL と名づけられた属性と原始集合として 4 台のトラックがある．もし，属性 HAUL がトラックが運べる量を意味するならば，次のように@FOR を使って運べる量を 2,500 ポンドに制約できる．

```
@FOR( TRUCKS( T): HAUL( T) <= 2500);
```

この場合，LINGO が作った制約を見てみるのも有益である．Windows の場合，LINGO|Generate コマンドを使う．他のプラットフォームの場合は GENERATE コマンドを使う．このコマンドで，LINGO が次の 4 つの制約を作るのが分かる．

```
HAUL( MAC) <= 2500;
HAUL( PETERBILT) <= 2500;
HAUL( FORD) <= 2500;
HAUL( DODGE) <= 2500;
```

予想した通り，各トラック毎に一つの制約を作った．

@FOR（太字部分）を使って，属性である GPM にある五つの数字の逆数を求めるモデルは次のようになる．

```
SETS:
OBJECT: GPM,  MPG;
ENDSETS
DATA:
OBJECT = A B C D E;
GPM = .0303 .03571 .04545 .07142 .10;
ENDDATA
@FOR( OBJECT( I):
MPG( I) = 1 / GPM( I));
```

このモデルを解くと，逆数として次の値が得られる．

```
  変数    値
MPG( A) 33.00330
MPG( B) 28.00336
MPG( C) 22.00220
MPG( D) 14.00168
MPG( E) 10.00000
```

ゼロの逆数は定義されていないため，0 に出会ったら，その値をとばすように条件文の限定詞を＠FOR に入力する．そのために＠FOR 文は次のようになる．

```
@FOR( OBJECT( I) | GPM( I) #NE# 0:
MPG( I) = 1 / GPM( I));
```

条件文の限定詞（太字部分）は，GPM が 0 と同等でない（#NE#)ことを確かめる．もし 0 でないならば，計算を続ける．

以上は，＠FOR の簡単な説明であるが，この後の節で色々な例題を挙げる．

5.4.4 ネスト化した集合ループ関数

前節で紹介した簡単なモデルは，＠FOR を一つの集合に対応させていた．大きなモデルの場合，他の集合ループ関数にある集合ループ関数を対応させる必要がある．一つの集合ループ関数が他の集合ループ関数の範囲にある場合，それをネスティングと呼びます．LINGO では，ネスティングすることができる．

次の例は，＠SUM ループが@FOR にネスティングした例である．

```
! The demand constraints;
@FOR( VENDORS( J):
@SUM( WAREHOUSES( I): VOLUME( I,  J)) = DEMAND( J));
```

各ベンダー（VENDERS）の需要量と，倉庫(WAREHOUSE)からベンダーに輸送される量の和を等しくする．＠SUM，@MAX，@MIN は，どの集合ループ関数にもネスティングできる．しかし，＠FOR は@FOR にのみネスティングが可能である．

5.5 集合をベースにしたモデルの例

LINGO で作られる 4 つの集合を思い出して下さい．

・原始集合
・密な派生集合
・疎な派生集合－明示的なリスト
・疎な派生集合－membership filter

この節では，上記 4 種類のモデルの構築と説明を通して，集合ベースのモデリングの向上を目指す．

5.5.1 原始集合の例

次の要員配置モデルでは，原始集合の使い方を説明する．このモデルは，LINGO のメイン・ディレクトリにある，SAMPLES というサブディレクトリに STAFFDEM.LNG というファイル名前で保存されている．

問題

Pluto Dogs という 7 日間毎日営業する人気ホットドッグ店を経営すると仮定する．要員は，5 日間労働し，必ず 2 日間連続で休日を与えるという条件で雇う．各要員の週給は同じである．忙しい曜日とそうでない曜日があり，過去の経験からどの日により多くの労働力が必要かは分かっている．予想では，下記の人数が各曜日に必要である．

曜日	月	火	水	木	金	土	日
必要要員数	20	16	13	16	19	14	12

必要な人数を下回ることなく労働力を最低限に抑えるために，最初に働く各曜日の初めに何人必要かを決めなければいけない．

定式化

集合ベースのモデルを作るために先ず考えなければいけないことは，「どれが関連する集合で，その属性は何か」ということである．このモデルには，「曜日(DAYS)」という一つの原始集合しかない．DAYS には，2 つの属性がある．一つ目は，各曜日に必要な要員の人数（REQUIRED）で，もう一つの決定変数は，各曜日の開始時の要員の人数（START）である．これで集合と Data 節を書き始められる．

```
SETS:
DAYS : REQUIRED,  START;
ENDSETS
DATA:
DAYS = MON TUE WED THU FRI SAT SUN;
REQUIRED = 20 16 13 16 19 14 12;
ENDDATA
```

モデルの数学的な関係（目的と制約）を入力するところまできた．目的関数を書くところから始めよう．ここでは，要員の人数を最小化することが目的となる．これを数学的に表記すると次のようになる．

$$\text{Minimize:} \sum \text{START}_i$$

LINGO の文に置き換えると，数学的な表記に似ていることが分かる．「MIN=」で「Minimize」を置き換え，「Σi」を「@SUM(DAYS(I):」で置き換えるだけである．LINGO の式を書いてみよう．

```
MIN = @SUM( DAYS( I): START( I));
```

これで，残りは制約のみである．このモデルには，一つの制約しかない．それは，毎日必要な要員数を下まわってはいけないということである．書き換えてみると，次のようになる．

各曜日：本日出勤の要員の数≧一日に必要な人数

この式の右側にある必要な要員数は分かっている．それは単に REQUIRED (I)である．左側の「本日の出勤の要員数」を割り出すのは多少困難である．要員は「5 日間働いて 2 日間連続して休日をとる」ため，それを計算する式は次のようになる．

本日の出勤人数＝今日から働き始める人数 + 昨日から働き始めた要員数 + 2 日前に働き始めた要員数 + 3 日前に働き始めた要員数 + 4 日前に働き始めた要員数

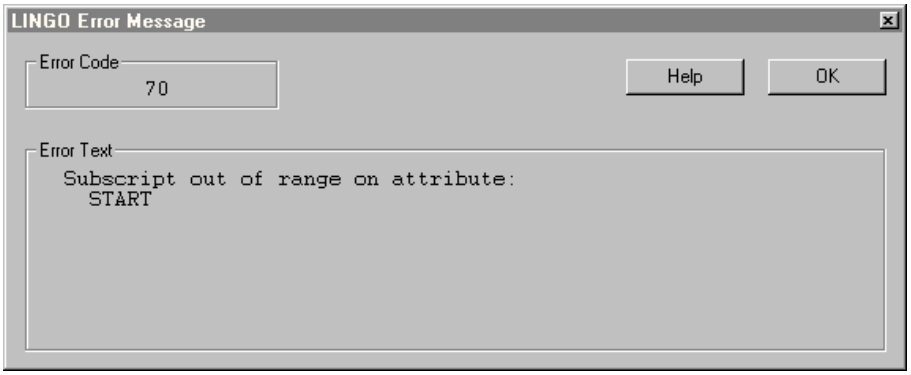
今日働いている人数を計算するには，今日から働き始める人数と 4 日前まで働いていた人数を足した数を計算する．5 日前から働き始めた人数と 6 日前から働き始めた人数は，休日にあたるため数えません．数学的な表現をすると下記のようになる．

$\Sigma_{i=j-4,j}\,START_i$ (REQUIREDj, for j DAYS)

これを LINGO 形式に直すと，次のようになる．

```
@FOR( DAYS( J):
@SUM( DAYS( I) | I #LE# 5: START( J - I + 1))>= REQUIRED( J));
```

これを言葉で言い表すと，各曜日 5 日間に働く人数として，4 日前から今日までに働き始める要員数が必要最低人数と同じ，もしくはそれ以上になるという意味になる．LINGO の文は，4 日前に働き始めた要員から当日までの 5 日間の合計が，当日に要求されている要員数より多いか等しいということなので，一見正しいように見えるが，このモデルをこの制約で解析しようとするとエラーメッセージが出る．



なぜこのエラーメッセージが出るのかを理解するために，木曜日に何が起きているかを考えてみよう．木曜日は，DAYS の索引で「4」とされている．要員を配置する上で木曜日の制約は次のように表される．

```
START( 4 - 1 + 1) + START( 4 - 2 + 1) +
START( 4 - 3 + 1) + START( 4 - 4 + 1) +
```

```
START( 4 - 5 + 1) >= REQUIRED( 4);
```

簡略化すると次のようになる.

```
START( 4) + START( 3) +
START( 2) + START( 1) +
START( 0) >= REQUIRED( 4);
```

START(0)が問題である. START は曜日 1 から 7 なので, START(0)は存在しない. よって, 0 という索引は「範囲外」と認識される.

インデックスが 0 以下の場合, 週の最後に回す. 特に 0 は日曜日(7), -1 は土曜日(6)…となる. LINGO に, これを行う@WRAP 関数がある.

@WRAPS は, INDEX と LIMIT という 2 つの引数を必要とする. 正式には, @WRAP は J = INDEX - K*LIMIT のようになる. ここで, K は J が区間[1, LIMIT] に含まれるような整数である. 略して言えば, @WARP は, INDEX に LIMIT を足す, もしくは引くかを, 解が 1 から LIMIT の間になるまで続ける. よって, それは多重周期計画モデルの索引に対応できる.

@WRAP を取り入れると, 要員の制約は最終的に次のように修正される.

```
@FOR( DAYS( J):
@SUM( DAYS( I) | I #LE# 5:
START( @WRAP( J - I + 1,  7))) >= REQUIRED( J));
```

下記が要員配置モデルの全体像である.

```
SETS:
DAYS : REQUIRED,  START;
ENDSETS
DATA:
DAYS = MON TUE WED THU FRI SAT SUN;
REQUIRED = 20 16 13 16 19 14 12;
ENDDATA
MIN = @SUM( DAYS( I): START( I));
@FOR( DAYS( J):
@SUM( DAYS( I) | I #LE# 5:
START( @WRAP( J - I + 1,  7))) >= REQUIRED( J));
```

このモデルを解くと, 解のレポートが表示される.

```
Optimal solution found at step:  8
Objective value:  22.00000

Variable           Value   Reduced Cost
REQUIRED( MON) 20.00000 0.0000000
REQUIRED( TUE) 16.00000 0.0000000
```

```
REQUIRED( WED) 13.00000 0.0000000
REQUIRED( THU) 16.00000 0.0000000
REQUIRED( FRI) 19.00000 0.0000000
REQUIRED( SAT) 14.00000 0.0000000
REQUIRED( SUN) 12.00000 0.0000000
START( MON)     8.00000 0.0000000
START( TUE)     2.00000 0.0000000
START( WED)     0.00000 0.0000000
START( THU)     6.00000 0.0000000
START( FRI)     3.00000 0.0000000
START( SAT)     3.00000 0.0000000
START( SUN)     0.00000 0.0000000

Row Slack or Surplus Dual Price
1       22.00000     1.000000
2        0.0000000  -0.2000000
3        0.0000000  -0.2000000
4        0.0000000  -0.2000000
5        0.0000000  -0.2000000
6        0.0000000  -0.2000000
7        0.0000000  -0.2000000
8        0.0000000  -0.2000000
```

目的値は 22 で，22 名の要員を雇用することを示す.

スケジュールに沿って要員を配置する.

	月	火	水	木	金	土	日
開始	8	2	0	6	3	3	0

必須要員数を示す行（行 2－7）の Slack or Surplus を見てみると，全ての曜日で 0 となっている．これは，余分な要員は配置されていないということを意味する.

5.5.2 密な派生集合の例

次の例では，混合モデルにおける密な派生集合の使い方を説明する．この例は，LINGO のメイン・ディレクトリにある SAMPLES サブディレクトリにある CHESS.LING というファイル名で保存されている.

問題

Chess Snackfoods(CS)社はPawn, Knight, Bishop, Kingという4種のナッツ製品を取り扱っている．各製品には特定の割合でピーナッツとカシューナッツが混ぜられている．各ナッツの割合と1袋ごとの価格が下記の表にまとめられている．

	Pawn	Knight	Bishop	King
ピーナッツ（oz．）	15	10	6	2
カシューナッツ（oz．）	1	6	10	14
価格（ドル/ポンド）	2	3	4	5

CS社は，業者から一日に750ポンドのピーナッツと250ポンドのカシューナッツを受け取る契約をする．今回の問題は，手持ちのナッツの量を超さずに収益を最大化するには，どの製品を何ポンドずつ生産すれば良いかということになる．

訳注：この表は，各製品に含まれるピーナッツとカシューナッツがオンスであり，価格がドル/ポンドと異なっている点である．原著はポンドに統一したモデルを解いている．オンスで統一したモデルに変更する場合は，下の表を参考にしてモデルを変更してみよう．ちなみに1ポンド16オンスである．

	Pawn	Knight	Bishop	King	在庫（オンス）
ピーナッツ（oz．）	15	10	6	2	750*16=12,000
カシューナッツ（oz．）	1	6	10	14	250*16=4,000
価格（ドル/オンス）	2/16	3/16	4/16	5/16	

定式化

このモデルの原始集合は，ナッツの種類と配合の種類である．NUTS集合には，SUPPLYという一つの属性がある．これはナッツの毎日の供給量をポンドで示したものである．BRANDS集合には，PRICEとPRODUCEという2つの属性がある．PRICEは製品の売値，PRODUCEは一日に何ポンドの製品を作るかを決める決定変数である．製品の式を入力するには，もう一つ集合が必要となる．ナッツの種類と製品を定義する2次元の表が必要となる．これを作るためには，NUTSとBRANDS集合から派生集合を作らなければいけない．この派生集合を付け加えると集合節が完了する．

SETS:

```
NUTS : SUPPLY;
BRANDS : PRICE,  PRODUCE;
FORMULA( NUTS,  BRANDS): OUNCES;
ENDSETS
```

この派生集合を FORMULA と名づけた．FORMULA には OUNCES という属性があり，製品にどのナッツを何オンス使うかを保存する．しかし，この派生集合のメンバーはまだ指定されていないため，LINGO は全てのナッツのペアと製品を含む密な集合を作る．集合の定義ができたので，Data 節に移る．まず，次のように SUPPLY， PRICE，OUNCES を初期化する．

```
DATA:
NUTS = PEANUTS,  CASHEWS;
SUPPLY = 750 250;
BRANDS = PAWN,  KNIGHT,  BISHOP,  KING;
PRICE = 2 3 4 5;
OUNCES = 15 10 6 2 !(Peanuts);
         1 6 10 14; !(Cashews);
ENDDATA
```

データと集合の用意ができたので，目的関数と制約に入る．利益を最大化する目的関数は，比較的簡単である．

```
MAX = @SUM( BRANDS( I): PRICE( I) * PRODUCE( I));
```

制約は，一日に供給されるナッツの量以上は生産できないということのみである．言い変えると，ナッツの種類 i ごとに使用されるナッツ i の量(ポンド)は，ナッツ i の供給量以下でなくてはならない．

LINGO 形式で表現すると，次のようになる．

```
@FOR( NUTS( I):
@SUM( BRANDS( J): OUNCES( I,J) * PRODUCE( J) / 16) <= SUPPLY( I));
```

オンスをポンドに変換するために左辺の総和を 16 で割る．完成したモデルは下記のようになる．

```
SETS:
NUTS : SUPPLY;
BRANDS : PRICE,PRODUCE;
FORMULA( NUTS,BRANDS): OUNCES;
ENDSETS
DATA:
NUTS = PEANUTS, CASHEWS;
SUPPLY = 750 250;
```

```
BRANDS = PAWN,KNIGHT, BISHOP, KING;
PRICE = 2 3 4 5;
OUNCES = 15 10 6 2 !(Peanuts);
                1 6 10 14; !(Cashews);
ENDDATA
MAX = @SUM( BRANDS( I): PRICE( I) * PRODUCE( I));
@FOR( NUTS( I): @SUM( BRANDS( J):
        OUNCES( I, J) * PRODUCE(J)/16) <= SUPPLY(I));
```

訳注:ここで注意すべきは，上のモデルで利益の最大化の式を 16 で割っているのは，オンスをポンドに変換するためである．解のレポートは，以下のように表現される．変数欄で下線を引いた以外の変数は上の表で決められた定数であり，0 がないので減少費用は 0 である．4 製品の生産量は Pawn が 769.2308 ポンド，KING を 230.7692 ポンド製造し，KNIGHT と BISHOP は製造しないことになる．顧客対応で，KNIGHT と BISHOP を 1 単位製造すると KNIGHT は 0.1538462 ドル/オンス，BISHOP は 0.07692308E ドル/オンス利益を悪化させることが減少費用からわかる．

```
Global optimal solution found.
Objective value:    2692.308
Variable                  Value        Reduced Cost
SUPPLY( PEANUTS)          750.0000     0.000000
SUPPLY( CASHEWS)          250.0000     0.000000
PRICE( PAWN)              2.000000     0.000000
PRICE( KNIGHT)            3.000000     0.000000
PRICE( BISHOP)            4.000000     0.000000
PRICE( KING)               5.000000    0.000000
PRODUCE( PAWN)            769.2308     0.000000
PRODUCE( KNIGHT)          0.000000     0.1538462
PRODUCE( BISHOP)          0.00000      0.7692308E-01
PRODUCE( KING)            230.7692     0.000000
OUNCES( PEANUTS,  PAWN)    15.00000     0.000000
OUNCES( PEANUTS,  KNIGHT)  10.00000     0.000000
OUNCES( PEANUTS,  BISHOP)  6.000000     0.000000
OUNCES( PEANUTS,  KING)    2.000000     0.000000
OUNCES( CASHEWS,  PAWN)    1.000000     0.000000
OUNCES( CASHEWS,  KNIGHT)  6.000000     0.000000
OUNCES( CASHEWS,  BISHOP)  10.00000     0.000000
```

```
OUNCES( CASHEWS,  KING)     14.00000     0.000000

Row     Slack or Surplus      Dual Price
  1         2692.308            1.000000
  2         0.000000            1.769231
  3         0.000000            5.461538
```

解のレポートを見てみると，CS 社は＄2692.30 の利益を得るには，769.2 ポンドの Pawn と 230.8 ポンドの King を製造するのがよいことが分かる．さらに，Dual Price(双対価格)列を見てみると，CS 社は 1 ポンドのピーナッツなら＄1.77 まで，カシューナッツなら＄5.46 までなら追加でナッツを購入してもよいという結果が出ている．マーケティング的な理由により，少なくてもある程度の Knight と Bishop を生産する場合，減少費用の列を見てみると，Knight を最初の 1 ポンド生産すると 15.4 セント，Bishop を最初の 1 ポンド生産すると 7.69 セント利益が減少することが分かる．

訳注:SOLVE→Generate->Display model を指定すると，以下のような自然表記のモデルが生成される．最初，SET 集合を用いたモデルに不慣れな場合，確認してみることを勧める．制約式の係数が実数なのはポンドにするため 16 で割っているからである．

```
MODEL:
 [_1] MAX= 2 * PRODUCE_PAWN + 3 * PRODUCE_KNIGHT + 4 * PRODUCE_BISHOP + 5 *
   PRODUCE_KING;
 [_2] 0.9375 * PRODUCE_PAWN + 0.625 * PRODUCE_KNIGHT + 0.375 * PRODUCE_BISHOP +
   0.125 * PRODUCE_KING <= 750;
 [_3] 0.0625 * PRODUCE_PAWN + 0.375 * PRODUCE_KNIGHT + 0.625 * PRODUCE_BISHOP +
   0.875 * PRODUCE_KING <= 250;
 END
```

5.5.3 疎な派生集合－明示的なリストの例

ここでは，疎な派生集合の明示的なリストの使い方を説明する．この方法を使って疎な集合を定義する場合，集合に属する全てのメンバーを直接入力する．それらは大抵，親集合のデカルト積全部から作られる小さくて密な部分集合である．

例として，新製品を展開するプロジェクトに関わるタスクのクリティカルパスを決定する PERT (Project Evaluation and Review Technique) モデルを考えてみよう．PERT とは，1950 年代に開発された大きなプロジェクトの進行具合を管理する責任者達をサポートするための簡単で強力な技法である．公式に適用された初めてのアプリケーションは，「ポラリス・プロジェクト」という弾道ミサイル積載潜水艦プロジェクトでした．Craven (2001)によると，PERT はポラリス・プロジェクトのキーパーソンで

あったAdmiral William F. Rabornという人が名づけた．Rabornには，PERTというニックネームの新妻がいました．その新妻に敬意を表して，ポラリス・プロジェクトをモニターする管理システムをPERTとしたのである．実際，ポラリス・プロジェクトは予定より18ヶ月も早く完了した．PERTは特に遅れたら全体の進行に響いてしまうような重要な仕事の確認に便利である．この全体に響いてしまうようなタスクを**クリティカルパス**と呼ぶ．長期に渡る大プロジェクトの動きを理解することは，プロジェクトが目的から脱線したり，遅延の可能性を大幅に減らすことに繋がる．PERTや**PERTに似たCPM(クリティカルパス法)**は，今後も多様なプロジェクトの成功に役立つでしょう．この例は，LINGOメイン・ディレクトリの**SAMPLESフォルダーにPERT．LNG**というファイル名で保存されている．

訳注:日本ではプロジェクト管理はガントチャートで行われている．PERT専用パッケージは高額であるが，LINGOで簡単に行えるので，ぜひ使ってみてほしい．

問題

Wireless Widgets (WW) 社がSolar Widgetという新製品を売り出そうとしている．発売予定日に遅れがでないよう，WW社はその予定日までのタスクをPERTで分析したいと考えている．その分析により，時間内にしておかなければ遅延が発生する仕事－**クリティカルパス**－を確認することができ，Solar Widgetを予定通りに売り出すことが出来る．それらの仕事は，製品発表の前に完了していなければいけない．それらの予定完了日は，下記の通りである．

仕事	週
デザインの仕上げ	10
需要の予測	14
競合の調査	3
価格設定	3
生産工程の計画	7
費用の見積	4
販売員の訓練	10

特定のタスクは，他のタスクが始まる前に終わらせなければいけない．下記に，優先順位を表す関係図がある．

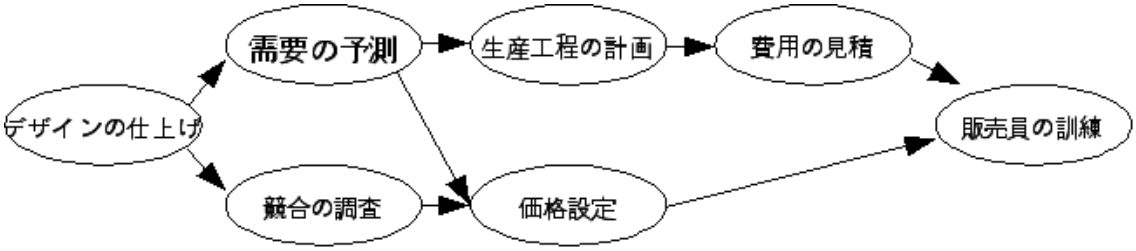


図 5.1 製品発売に向けての優先順位関係

例えば，「需要の予測(Forecast)」から出ている 2 つの矢印は「生産工程の計画(Design)」と「価格設定(Price)」より前に「需要の予測」を終わらせなければならないことを示す．目的は，Solar Widgets の発売に向けて PERT モデルを使い，**タスク（作業）**のクリティカルパスを特定することである．

訳注: 「需要の予測」は先行作業と呼び，「生産工程の計画」と「価格設定」は後行作業という．「需要の予測」と「競合の調査(Survey)」には，先行作業と後行作業の関係はない．そしてこの先行と後行作業の関係と，各作業の作業時間だけが PERT に必要とされる．作業時間は，最短，平均，最長などの 3 点見積もりもあるがここでは平均などの 1 点見積もりで行う．汎用モデルなので，最短や最長のデータで試行すればよい．

定式化

このプロジェクトを表現するには，原始集合が必要である．TASKS という集合に次の 4 つの属性を結び付ける．

TIME ：作業を完了させる時間（既知）
ES ：可能な限り早い作業を開始する時間（計算が必要）
LS ：作業を開始する最遅の時間（計算が必要）
SLACK：その作業の LS と ES の差（計算が必要）

もし，作業の SLACK が 0 ならば，その作業は決められた日時に開始しないとプロジェクト全体が遅れる重要な作業であることを意味する．このように SLACK が 0 の作業がクリティカルパスであり，この作業の進捗管理が重要になる．作業の開始時を計算するには作業間の優先関係を考える必要があるため，モデルに優先順位の関係を入力する．この優先順位関係を作業の順序対のリストとする．例えば，DESIGN が FORECAST の前に終わっていなければならないという関係を表す場合，(DESIGN, FORECAST) となる．TASKS 集合に 2 次元の派生集合を作る事で，優先順位の関係を入力できる．よって，Data 節は次のようになる．

訳注：1 次元集合 TASKS は 7 個の作業からなる要素数 7 の 1 次元の原子集合である．そして，各作業の作業時間が要素数 7 の 1 次元配列に DATA 節で与えられる．ES, LS, SLACK は最適計算で計算された値が格納される要素数 7 の 1 次元配列である．PRED は，TASKS から作られる 2 次元の派生集合であるので，密集合の場合 7*7 の 49 の要素数を持つ 2 次元配列である．しかし，PERT では DATA 節で先行作業と後続作業の 8 個のペアでできた疎な集合として直接定義される．

```
SETS:
TASKS : TIME,  ES,  LS,  SLACK;
PRED( TASKS,  TASKS);
ENDSET
```

PRED には，属性（配列）がないことに留意して下さい．代わって，次の DATA 節で直接定義している．作業の 7 個の要素と時間と順序対を Data 節で入力すると，次のようになる．

```
DATA:
TASKS= DESIGN, FORECAST, SURVEY, PRICE, SCHEDULE, COSTOUT, TRAIN;
TIME = 10, 14, 3, 3, 7, 4, 10;
PRED =
DESIGN,FORECAST,
DESIGN,SURVEY,
FORECAST,PRICE,
FORECAST,SCHEDULE,
SURVEY,PRICE,
SCHEDULE,COSTOUT,
PRICE,TRAIN,
COSTOUT,TRAIN;
ENDDATA
```

PRED の最初のメンバーは，順序対 (DESIGN, FORECAST)であり，DESIGN という一つの作業ではないことを忘れないで下さい．よって，この集合には 49 個ある作業の順序対のうち，合計で 8 個のメンバーが入る．それぞれは，優先順位の関係を示す図のアークに対応する．

この例から学んでおくことは，PRED が明示的なリストを使った**疎な派生集合**であることである．この集合は，7 個の作業の対からできたサブセットである．疎であるのは，49 個あるメンバーの内，8 個のみでできた集合である．明示的なリストであるのは，その集合に入れたいメンバーを直接 DATA 節で定義したからである．明示的に疎な

集合のメンバーのリストを作るのは，数千ものメンバーから選ばなくてはならない場合には便利ではない．しかし，メンバーの集合条件がはっきりと定義されておらず，密なものよりも疎な集合の規模が小さい場合に意味がある．

集合とデータができあがった．次は式を作ってみよう．先ず，一番早い開始時間（ES)，一番遅い開始時（LS)，その時間の差（SLAC)を計算する．これらの時間が分かれば，SLACK はその差を計算するだけであるので，問題は ES と LS の計算である．先ずは，ES を計算するための式から始めてみよう．作業は，それ以前に完了していなければならない作業が全て終わっていなければ始められない．このため，この作業以前の作業全ての一番遅い完了時間が分かれば，この作業の一番早い開始時間になる．簡単に言うと，作業 t の一番早い開始時間は，t 以前の作業全ての一番早い開始時間にその作業時間を足したものの最大値になる．LINGO の表現で表すと次のようになる．

```
@FOR( TASKS( J)| J #GT# 1:
ES( J) = @MAX( PRED( I,J): ES( I) + TIME( I)));
```

最初の作業 Design（TASKS(1)）には，それ以前の作業がないため，「J#GT# 1」を加えて作業 1 の E(1)の計算を省く．E(1)＝0 と考える．

訳注: TASKS(2)は Forcast であるが，ES(2)= @MAX(PRED(I,2): ES(I) + TIME(I)));で ES(2)を計算する．右辺の PRED(I,2)すなわち Forcast を後続作業とする先行作業は，Design（TASKS(1)）しかないので，ES(2)= @MAX(PRED(1, 2): ES(1) + TIME(1)))= ES(1) + TIME(1) = 0 + 10 = 10 になる．すなわち，Forcast は先行作業の Design が 0 から作業を始めるので ES(1)=0 で，作業時間 TIME(1)は 10 時間であり，TASKS(2)の ES(2)は 10 時間後に一番早く作業を始めることができる．同様に TASKS(3)は Survey であるが，ES(3)は同じく 10 時間になる．TASKS(4)は Price であるが，ES(4) = @MAX(PRED(I, 4): ES(I) + TIME(I))) = @MAX[{ES(2) + TIME(2)}, {ES(3) + TIME(3)} = @MAX[{10 + 14}, {10 + 3}] = @MAX[24, 13] = 24; になる．すなわち，TASKS(4)の Price は，先行作業の Forecast が 24 時間後に作業を終え，Survey が 13 時間後に終えるので，両作業が終了する 24 時間後にさも早く作業を始めることができる．最後の TASKS(7)の Training は，先行作業の Costoutt が 35 時間後に作業を終え，Price が 27 時間後に終えるので，両作業が終了する 35 時間後にさも早く作業を始めることができる．すなわち E(7)=35 である．

LS の計算は，手順を逆に考え，MAX の代わりに MIN 演算を用いる以外は ES と似ている．先行作業 I の一番遅い開始時間は，後行作業 j の一番遅い開始時間から I の作業時間を引いた時間の最小値となる．これを LINGO で表現するとこうなる．

```
@FOR( TASKS( I)| I #LT# LTASK:
```

LS(I) = @MIN(PRED(I,J): LS(J) - TIME(I)));

最後の作業は以後に作業はないので，「I #LT# LTASK」で最後の作業の計算を省く．

SLACK は，LS と ES の差なので，次のようなる．

@FOR(TASKS(I): SLACK(I) = LS(I) - ES(I));

作業 1 の開始時間は任意の値に設定できるが，今回は 0 とする．

ES(1) = 0;

最後の作業の一番遅い開始時間を除いて全ての変数の値を計算する式ができた．最後の作業の一番遅い開始時間は，一番早い開始時間と同じと考える．LINGO の式で表すと次のようになる．

LS(7) = ES(7);

これでもいいが，幾つかの作業をモデルに追加するとすると，この式の「7」を作業の数を表す新しい数字と置き換えなければいけない．LINGO の集合ベースのモデリング言語は，式に影響なくデータを変更できることが目的なので，これではデータの独立性を犯すので，次のように書き換えた．

LTASK = @SIZE(TASKS);

LS(LTASK) = ES(LTASK);

@SIZE 関数は集合の大きさを返すので，この問題では 7 になる．作業数を変更した場合でも，@SIZE は新しい正確な値を返す．このように，モデルの構造のデータ独立性を保つことができる．

PERT 全体と解を下記に記載した．

```
SETS:
TASKS : TIME, ES, LS, SLACK;
PRED( TASKS, TASKS);
ENDSETS
DATA:
TASKS= DESIGN, FORECAST, SURVEY, PRICE, SCHEDULE, COSTOUT, TRAIN;
TIME = 10, 14, 3, 3, 7, 4, 10;
PRED =
DESIGN,FORECAST,
DESIGN,SURVEY,
FORECAST,PRICE,
FORECAST,SCHEDULE,
SURVEY,PRICE,
SCHEDULE,COSTOUT,
PRICE,TRAIN,
COSTOUT,TRAIN;
```

```
ENDDATA
@FOR( TASKS( J)| J #GT# 1:
ES( J) = @MAX( PRED( I, J): ES( I) + TIME( I)));
@FOR( TASKS( I)| I #LT# LTASK:
LS( I) = @MIN( PRED( I, J): LS( J) - TIME( I)););
@FOR( TASKS( I): SLACK( I) = LS( I) - ES( I));
ES( 1) = 0;
LTASK = @SIZE( TASKS);
LS( LTASK) = ES( LTASK);
```

以下が出力である.

```
 Variable       Value
 LTASK          7.000000
 ES( DESIGN)    0.000000
 ES( FORECAST) 10.000000
 ES( SURVEY)    10.000000
 ES( PRICE)     24.000000
 ES( SCHEDULE) 24.000000
 ES( COSTOUT)   31.000000
 ES( TRAIN)     35.000000
 LS( DESIGN)    0.000000
 LS( FORECAST) 10.000000
 LS( SURVEY)    29.000000
 LS( PRICE)     32.000000
 LS( SCHEDULE) 24.000000
 LS( COSTOUT)   31.000000
 LS( TRAIN)     35.000000
 SLACK(DESIGN)   0.000000
 SLACK(FORECAST)0.000000
 SLACK(SURVEY) 19.000000
 SLACK(PRICE)    8.000000
 SLACK(SCHEDULE)0.000000
 SLACK(COSTOUT) 0.000000
 SLACK( TRAIN)   0.000000
```

作業のスラックを見てみると, SURVEY と PRICE はそれぞれ 19 と 8 になっている. SURVEY と PRICE の開始時に遅れがでても, このスラック分だけの遅れならばプロジェクト全体の完了予定日を遅らせることはない. DESIGN, FORECAST, SCHEDULE, COSTOUT

と TRAIN は SURVEY と PRICE と違い，スラックが 0 である．これらがプロジェクトのクリティカルパスであり，開始が遅れるとプロジェクトの完了日に遅れが発生する．プロジェクト責任者は，こういったクリティカルパスが，決められた日時に開始し，予定した時間内で完了するよう気を配らなければいけない．最後に，ES(TRAIN)の値が 35 であるので，この Solar Widget プロジェクトの終了は 45 週間かかることを示す．45 週間になるのは，要員のトレーニング開始まで 35 週間かかり，トレーニング自体の作業時間が 10 週間なので，合計 45 週間かかることを示す．

5.5.4 メンバーシップ・フィルターを使った疎な派生集合

ここでは，メンバーシップ・フィルターを使った疎な派生集合を説明する．**メンバーシップ・フィルターとは，派生集合を定義する 3 つ目の方法**である．この方法を使って集合を定義する場合，各メンバーが満たさなければならない論理条件を指定する．この条件は，それを満たさないメンバーを密な集合から除外する時に使う．

マッチング問題で説明しょう．最低限の費用で組み合わせたい N 個のオブジェクトがある．これは，ルームメイト選択問題とも呼ばれている．この問題は，新入生を寮に割り当てるために，大学が年度始めに直面する問題でもある．ペアである（I，J）は，(J，I)と同じであるので， I は J よりも少ないとする．また I と J は順序対でなければいけないので，I が J よりも小さいという条件以外に，不必要な対は作りたくない．この「I は J より小さい」という条件がメンバーシップ・フィルターになる．

この例は，LINGO メイン・ディレクトリの **SAMPLES フォルダーのファイル名** MATCHD.LNG に保存されている．

問題

ある企業の戦略計画部には 8 人のアナリストがおり，部はもうすぐ新しいオフィスに移動する．そのオフィスには合計 4 つの部屋があり，8 人のアナリストを 4 組に分けることにした．観察の結果，誰と誰を組ませると効率が良いかが分っている．部の調和が乱れないよう，相性の悪いアナリストを同室にするのは最小限に抑えたい．これを実現するために不一致度の採点システムを編み出した．採点は 1 から 10 段階あり，組み合わせが 1 の場合は相性がすばらしく良いが，10 の場合はトラブルが考えられるため，オフィスから鋭利なものを片付けなければいけない．下記が点数の一覧表である．

Analysts	1	2	3	4	5	6	7	8
1	-	9	3	4	2	1	5	6
2	-	-	1	7	3	5	2	1

3	-	-	-	4	4	2	9	2
4	-	-	-	-	1	5	5	2
5	-	-	-	-	-	8	7	6
6	-	-	-	-	-	-	2	3
7	-	-	-	-	-	-	-	4

アナリスト「I と J」と「J と I」の組み合わせは同じなので，表の対角線より上部のみを表示した．この問題は，不一致度が最も低いアナリスト同士の組み合わせを探すことである．

定式化

先ずは，8 人のアナリストを集合にする．これは原始集合で，下記のように簡単に書き表せる．

```
ANALYSTS;
```

最後に作りたい集合は,可能な限り全ての組み合わせを含む集合である.これはANALYST同士が交差する派生集合であり密な派生集合を作る.

```
PAIRS( ANALYSTS, ANALYSTS);
```

しかし，この集合にはPAIRS(I, J)とPAIRS(J, I)が含まれている．同じペアは 2 つも要らない．さらに，この集合だと同じアナリスト同士（I,I）も含んでいる．個人で部屋を持つことがベストであるが，これは不可能である．よって，派生集合にメンバーシップ・フィルターをかけ，各（I, J）ペアを最終集合に入れるため，J は I より大きいという条件を課さなければいけない．これを次のように表する．

```
PAIRS( ANALYSTS, ANALYSTS) | &2 #GT# &1;
```

メンバーシップ・フィルターは縦線（|）の右で定義する．Filter 内の「&1 と&2」は行の ANALYST と列の ANALYST の代わりに対応し，メンバーシップ・フィルター内でしか使えないインデックスである．LINGO がそのまま PAIRS 集合を作ると，ANALYSTS の集合同士が重なる密な組み合わせを作り出すので，そこにメンバーシップ・フィルターを付け加える．特に，ANALYSTS 同士の集合に含まれる各ペア（I, J)は，I を&1 と置き換え，J を&2 とし，フィルターに用いる．結果が真のものだけ集合 PAIRS に追加される．表で見てみると，対角線より上部の（I, J）の組み合わせのみが集合の要素になる．

PAIRS の 2 つの属性についても考えなければいけない．先ずは，ペアの不一致度と関連のある属性が必要となる．そして，アナリスト J と I がペアになったことを示す属性が必要である．これらの属性を RATING と MATCH と呼ぶことにする．それらの属性を次のように PAIRS 集合につける．

```
PAIRS( ANALYSTS,ANALYSTS) | &2 #GT# &1: RATING,  MATCH;
```

Data で，上記の不一致度の RATING 属性を与える．

```
DATA:
ANALYSTS = 1..8;
RATING =
9 3 4 2 1 5 6
1 7 3 5 2 1
4 4 2 9 2
1 5 5 2
8 7 6
2 3
4;
ENDDATA
```

アナリストの組み合わせＩとＪが最適化計算で選ばれた場合は MATCH(I, J) = 1, それ以外は 0 にする．これで，MATCH はこのモデルの決定変数となる．目的関数は，最終的な組み合わせの中で不一致度を最小化することである．これは，RATING と MATCH 属の内積(Excel の SUMPRODUCT)で，次のように式に表すことができる．

```
MIN = @SUM( PAIRS( I, J): RATING( I, J) * MATCH( I, J));
```

このモデルには 1 つの制約があり，言葉で表すと「アナリストは，各自別のアナリストとペアを組まなければならない」となる．この制約を LINGO の式で，次のようになる．

```
@FOR( ANALYSTS( I):
@SUM( PAIRS( J, K) | J #EQ# I #OR# K #EQ# I:MATCH( J, K)) = 1 );
```

この制約で興味深いのは，@SUM の条件付き修飾子である J # EQ # I # OR # K # EQ # I である．全アナリスト I に対して，I を含む全ての MATCH 変数を合計して 1 にする．これで，I と組むアナリストが別のアナリストであることを確定できる．この条件付き修飾子は，I を含む MATCH 変数のみを合計するように指定する．

このモデルに必要な機能がもう 1 つある.I と J が組み合わせられた場合は MATCH(I, J)=1 になるように，それ以外は 0 となるように設定する．変数の範囲を制限しないと 0 から無限大の範囲になる．MATCH を 0 か 1 に制限したいので，@BIN という 0 と 1 の整数値しかとらない変数に指定する．変数範囲関数は，制約式にカウントされない．モデルに 2 値変数が含まれる場合，そのモデルは IP モデルになる．IP モデルは LP モデルよりも解くのに時間がかかる．数百以上の整数変数を含む IP モデルは，計算時間がどれだけかかるか予測できない．この理由で，できるだけ IP の使用は避けた方が良い．MATCH に属する全ての変数に@BIN を適用するには，次のように@FOR 関数を用いる．

```
@FOR( PAIRS( I, J): @BIN( MATCH( I, J)));
```

訳注: 誤分類数最小(Minimum Number of Misclassifications, MNM)基準に基づく最適判別関数（改定 IP-OLDF, RIP）を整数計画法で開発した．この RIP だけが，1 万個前後ある遺伝子で，癌症例と健常症例の 2 群判別を 10 秒程度で，MNM=0 で判別した．しかも 10 個前後の遺伝子の判別係数だけが 0 でなく残りが 0 になるので，この 10 個前後の遺伝子が癌遺伝子を含んでいると特定できた．一般に，整数計画法は時間がかかると考えられているが，癌症例と健常症例が MNM=0 で判別できる（線形分離という）場合，IP を用いた判別は容易である．これで，30 年以上統計的に結果の出ていなかった癌の遺伝子解析に成功し，癌の遺伝子診断を提案できた．

モデルと解を下記に記した．

```
SETS:
ANALYSTS;
PAIRS( ANALYSTS, ANALYSTS) | &2 #GT# &1: RATING,  MATCH;
ENDSETS
DATA:
ANALYSTS = 1..8;
RATING =
9 3 4 2 1 5 6
1 7 3 5 2 1
4 4 2 9 2
1 5 5 2
8 7 6
2 3
4;
ENDDATA
MIN = @SUM( PAIRS( I, J): RATING( I, J) * MATCH( I, J));
@FOR( ANALYSTS( I):
@SUM( PAIRS( J, K) | J #EQ# I #OR# K #EQ# I:  MATCH( J, K)) = 1 );
@FOR( PAIRS( I, J): @BIN( MATCH( I, J)));
```

解は次のようになった．

```
 Global optimal solution found.
Objective value:     6.000000
Variable      Value
MATCH( 1, 2) 0.0000000
MATCH( 1, 3) 0.0000000
MATCH( 1, 4) 0.0000000
```

MATCH(1, 5) 0.0000000

MATCH(1, 6) 1.000000

MATCH(1, 7) 0.0000000

MATCH(1, 8) 0.0000000

MATCH(2, 3) 0.0000000

MATCH(2, 4) 0.0000000

MATCH(2, 5) 0.0000000

MATCH(2, 6) 0.0000000

MATCH(2, 7) 1.000000

MATCH(2, 8) 0.0000000

MATCH(3, 4) 0.0000000

MATCH(3, 5) 0.0000000

MATCH(3, 6) 0.0000000

MATCH(3, 7) 0.0000000

MATCH(3, 8) 1.000000

MATCH(4, 5) 1.000000

MATCH(4, 6) 0.0000000

MATCH(4, 7) 0.0000000

MATCH(4, 8) 0.0000000

MATCH(5, 6) 0.0000000

MATCH(5, 7) 0.0000000

MATCH(5, 8) 0.0000000

MATCH(6, 7) 0.0000000

MATCH(6, 8) 0.0000000

MATCH(7, 8) 0.0000000

Row	Slack or Surplus	Dual Price
1	6.000000	-1.000000
2	0.000000	0.000000
3	0.000000	0.000000
4	0.000000	0.000000
5	0.000000	0.000000
6	0.000000	0.000000
7	0.000000	0.000000
8	0.000000	0.000000
9	0.000000	0.000000

目的値を見てみると，最適なペアリングの不一致度の点数の合計が 6 である．Value 列の「1」を見てみると（1,6），(2,7)，(3,8)，(4,5)が最適な組み合わせであることが分かる．

訳注:Generate でモデルを生成すると次のようになる．最初の式は，考えられる全てのペアと重みの積和である．次の 8 個の制約は，各 i と残りの 7 人とのペアの和を 1 に制約している．最後は，28 組のペアを 0/1 の整数変数に指定している．

```
MODEL:
[_1]MIN=9*MATCH_1_2+3*MATCH_1_3+4*MATCH_1_4+2*MATCH_1_5+MATCH_1_6+5*MA
TCH_1_7+6*MATCH_1_8+MATCH_2_3+7*MATCH_2_4+3*MATCH_2_5+5*MATCH_2_6
+2*MATCH_2_7+MATCH_2_8+4*MATCH_3_4+4*MATCH_3_5+2*MATCH_3_6+9*MATCH_3_7
+2*MATCH_3_8+MATCH_4_5+5*MATCH_4_6+5*MATCH_4_7+2*MATCH_4_8+8*MATCH_5_6
+7*MATCH_5_7+6*MATCH_5_8+2*MATCH_6_7+3*MATCH_6_8+4*MATCH_7_8;
[_2]MATCH_1_2+MATCH_1_3+MATCH_1_4+MATCH_1_5+MATCH_1_6+MATCH_1_7+MATCH_
1_8=1;
[_3]MATCH_1_2+MATCH_2_3+MATCH_2_4+MATCH_2_5+MATCH_2_6+MATCH_2_7+MATCH_
2_8=1;
[_4]MATCH_1_3+MATCH_2_3+MATCH_3_4+MATCH_3_5+MATCH_3_6+MATCH_3_7+MATCH_
3_8=1;
[_5]MATCH_1_4+MATCH_2_4+MATCH_3_4+MATCH_4_5+MATCH_4_6+MATCH_4_7+MATCH_
4_8=1;
[_6]MATCH_1_5+MATCH_2_5+MATCH_3_5+MATCH_4_5+MATCH_5_6+MATCH_5_7+MATCH_
5_8=1;
[_7]MATCH_1_6+MATCH_2_6+MATCH_3_6+MATCH_4_6+MATCH_5_6+MATCH_6_7+MATCH_
6_8=1;
[_8]MATCH_1_7+MATCH_2_7+MATCH_3_7+MATCH_4_7+MATCH_5_7+MATCH_6_7+MATCH_
7_8=1;
[_9]MATCH_1_8+MATCH_2_8+MATCH_3_8+MATCH_4_8+MATCH_5_8+MATCH_6_8+MATCH_
7_8=1;
@BIN(MATCH_1_2);@BIN(MATCH_1_3);@BIN(MATCH_1_4);@BIN(MATCH_1_5);@BIN(M
ATCH_1_6);@BIN(MATCH_1_7);@BIN(MATCH_1_8);@BIN(MATCH_2_3);@BIN(MATCH_2
_4);@BIN(MATCH_2_5);@BIN(MATCH_2_6);@BIN(MATCH_2_7);@BIN(MATCH_2_8);@B
IN(MATCH_3_4);@BIN(MATCH_3_5);@BIN(MATCH_3_6);@BIN(MATCH_3_7);@BIN(MAT
CH_3_8);@BIN(MATCH_4_5);@BIN(MATCH_4_6);@BIN(MATCH_4_7);@BIN(MATCH_4_8
);@BIN(MATCH_5_6);@BIN(MATCH_5_7);
```

```
@BIN(MATCH_5_8);@BIN(MATCH_6_7);@BIN(MATCH_6_8);@BIN(MATCH_7_8);
END
```

5.6 変数の定義域設定関数

変数の定義域設定関数は，この章のマッチング問題で@BIN を使った時に簡単に説明した．次の 4 つの定義域設定関数を使うことで，決定変数になりえる値を制約できる．

```
@BIN ( Y);
@GIN ( X);
@BND ( 100, DELIVER, 250);
@FREE ( PROFIT);
```

@BIN(Y)は，変数 Y を 2 値の 0/1 に制限する．これで，変数 Y は，0 か 1 という値になる．

@GIN(X)は，変数 X を非負の整数値に限定する．これで X を，0，1，2，…という非負の整数値に指定できる．

@BND()は，上限と下限値を指定する．例えば，@BND(100, DELIVER, 250)の場合,DELIVER は,[100, 250] の間の数でなければいけない．これには他の表現方法もある．

```
DELIVER >= 100;
DELIVER <= 250;
```

LINGO はデフォルトで，全ての決定変数の下限を 0 に設定する．@FREE(PROFIT)で下限を書き換え，(-∞，∞) になる．定義域設定関数は，他の制約と同様に@FOR ループで使うことができる．

5.7 LINGO とスプレッド・シート

この章では，大規模な問題を解く時に LINGO がどれほど便利かを説明した．モデルを作るのに，一番よく使われるのは，疑うことなくスプレッド・シートである．スプレッド・シートを使ってのモデリングの主な利点を挙げてみました．

・優れた報告書の書式設定機能
・幅広い層の人々に理解される分かりやすさ
・ワードなどの他の良くできたインターフェース能力

一方，LINGO とスプレッド・シートでモデル構築を行う利点を挙げてみた．

- 様々な意味で柔軟である．
- Excel 上にデータを定義してやれば，モデルはデータに依存せずに汎用化できる．
- スプレッド・シートのように列数の上限が 255 だとか，行の上限が 65536 であるという制約がないし，式に使える最大文字数が 900 以下というような制限もない．
- 疎な集合も表現できる．

・ 監査能力と可視性：LINGO では，完全かつ包括的な形で式を見ることができる．複雑なスプレッド・シート上で，モデル式を本当に理解するには，探偵がするような労力がいる．
・ 多次元を簡単に表現できる．スプレッド・シートは，2 次元は扱えるが，3 次元以上は難しい．

LINGO とスプレッド・シートを併用して最大の利益を得るでしょう．LINGO にリンクをつけ，自動的にスプレッド・シート，データベースや普通のファイルからデータを入出力できる． Windows の場合，OLE (Object Linking and Embedding)や ODBC (Open Database Connectivity)インターフェースがすでに備わっている．OLE 機能で Excel のスプレッド・シートを LINGO に繋ぐには 2 つのステップがある．

a) スプレッド・シートでは，出力先，入力元であるデータ域をセル範囲名で指定し，それを LINGO の配列名に用いて，入出力できる．

b)LINGO のモデル内で使われるスプレッド・シートから検索される各属性（ベクトル）（例：工場設備能力）は，LINGO の Data 節に次のように書きます．

```
CAPACITY = @OLE('C:¥MYDATA.XLS');
```

スプレッド・シートに送られるそれぞれの属性（例：輸送量）は，LINGO の Data 節に次のような形で示さなければいけない．

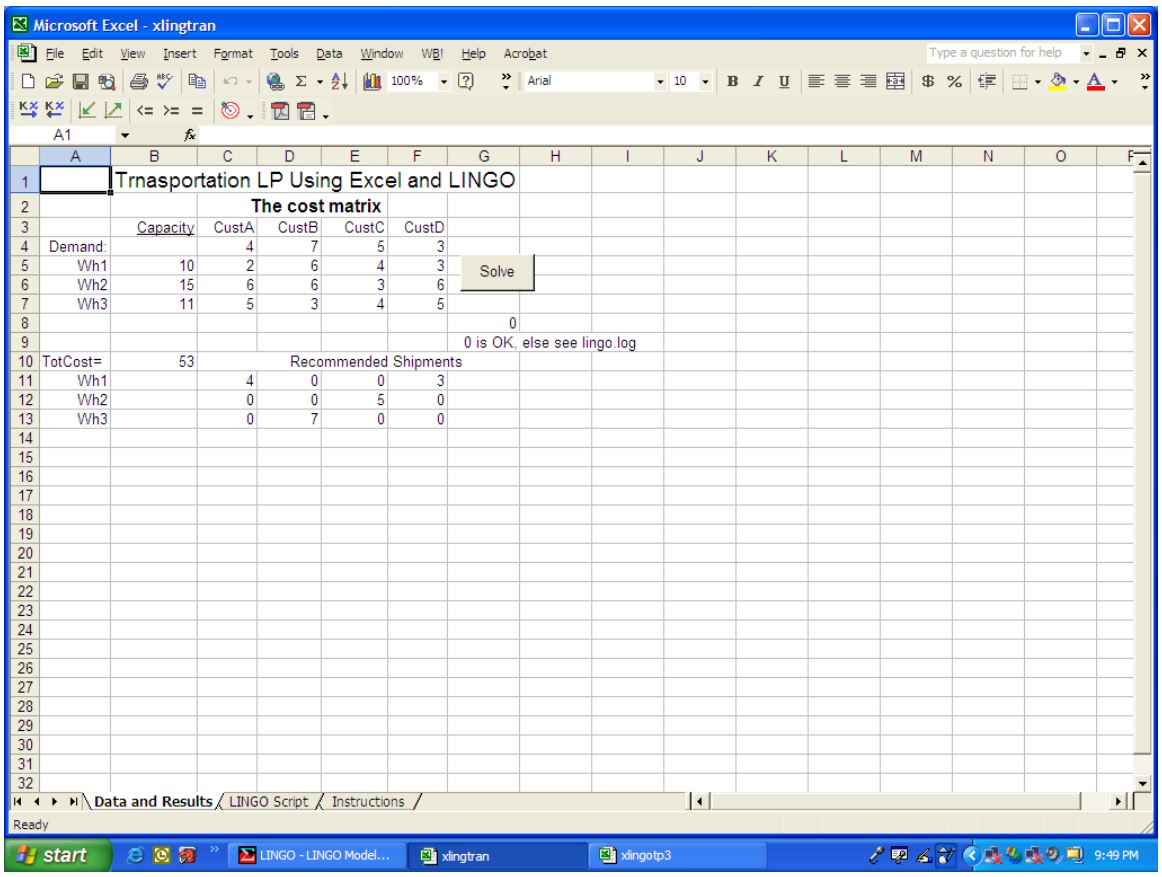
```
@OLE('C:¥MYDATA.XLS') = AMT_SHIPPED;
```

Excel のスプレッド・シートが一つだけしか開いていない場合，次のように簡略できる．

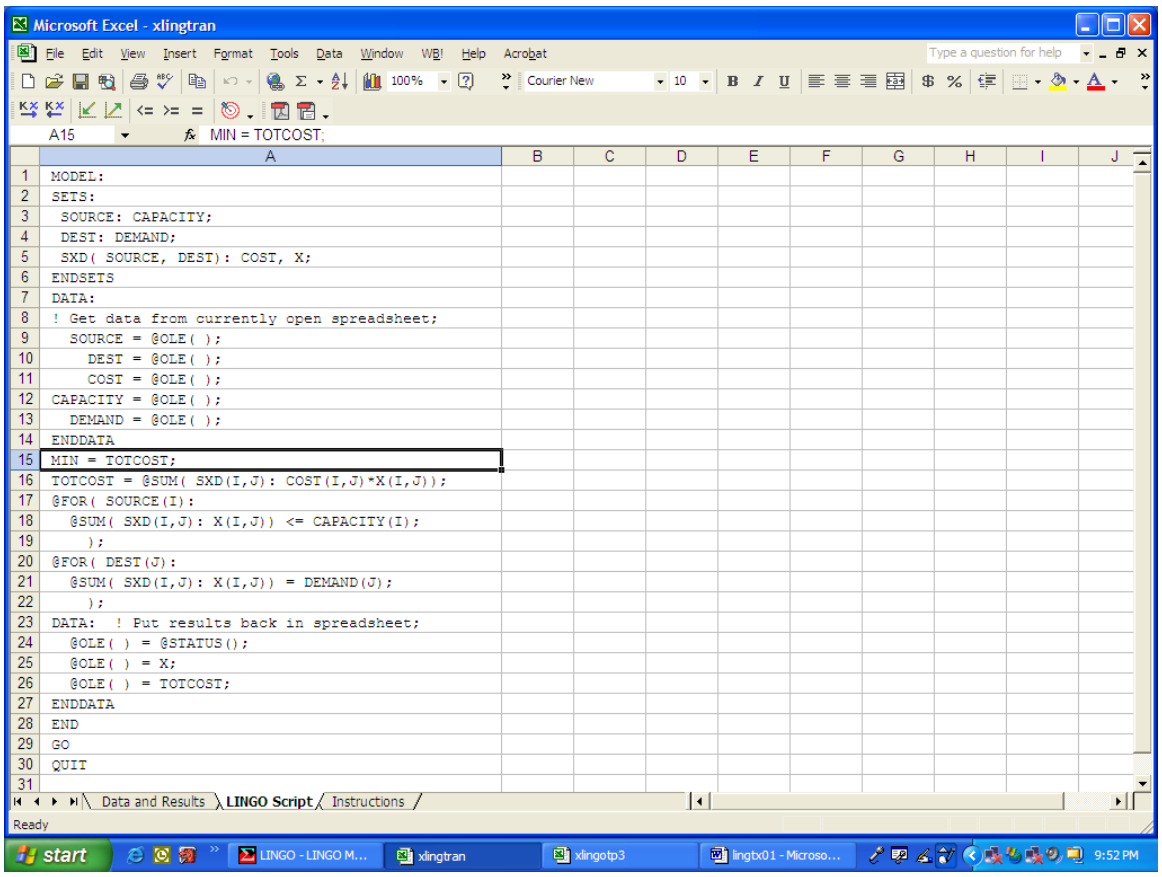
```
CAPACITY = @OLE();
```

LINGO は，開いているスプレッド・シートのみから CAPACITY という範囲を探す．同じモデルを違うスプレッド・シートのデータに用いたい場合，スプレッド・シートが特定されていないのは，とても便利である．

訳注:Excel に与えたデータをセル範囲名 A で指定する．それを LINGO の DATA 節で，「A=@OLE();」と指定することで，セル範囲名 A の値を LINGO の配列 A として使える．最適計算の結果を LINGO の配列 B に格納し，Excel 上のセル範囲名 B に，「@OLE()=B;」で出力できる．他のデータ管理ソフト（DB）でも同じである．



このデータと結果は，スプレッド・シートファイルの最初のタブ/シートに仕分けされている．一見して分からないかもしれないが，LINGO モデルが同じスプレッド・シートの他のタブに保存されている．下記を見て下さい．完全に隠されているのは，VBA のプログラムである．このプログラムが，最初のタブにある解析ボタンが押される度に 2 番目のタブにある LINGO モデルに解析をさせている．この例は，xlingtran.xls というファイル名で保存されている．



@OLE()が LINGO のモデルとスプレッド・シートを結ぶのに使われているように，@ODBC()は LINGO のモデルと SQL インターフェースをサポートするデータベースを繋ぎ，@TEXT()はシンプルなテキスト・ファイルと LINGO のモデルを繋ぎます．例えば，「myfile.out」というファイルに属性 X の値を送るには，次のようにする．

```
DATA:
@TEXT( 'MYFILE.OUT') = X;
ENDDATA
```

次の文で，X の値を説明文と共にスクリーンに出力する．

```
@TEXT() = 'The value of X=', X;
```

もう一つの LINGO を他のアプリケーション合体させる方法は，サブルーチンコールである．一般的な PC プログラミング，例えば C/C++や Visual Basic で，LINGO の DLL (Dynamic Link Library)を呼び出すことができる．モデルは文字列変数として LINGO の DLL に通される．詳しくは，LINGO のマニュアルを参照して下さい．

5.8 LINGO とプログラミング

LINGO10 の注目すべき新機能は，プログラミングかループピング機能である．この機能の主な利点は，

a）モデルで使われるデータの前処理（例：複雑な利益の寄与係数の計算）ができる，
b）LINGO の標準書式ではなく，解をカスタマイズした形で出力する後処理ができる，
c）一度に 2 つもしくはそれ以上のモデルを解析できる．
この複数のモデルを一度のクリックで解析できる機能は，次の場合に便利である．

1. ある臨界パラメータの効用により利益にどのような違いがでるかを見るパラメータ解析を行う．
2. 目標が階層構造をとっている場合，目標計画法問題を解く．
3. 反復しつつ，列生成しながら変数や制約を追加させていく追加的なモデル生成を行う．

5.8.1 プログラミングの部分構成

実行可能ステートメントは CALC 節で扱う．

```
CALC:
! Executable statements;
ENDCALC
```

計算式は，4 つの制御文（＠IFC，@FOR，@WHILE，＠BREAK）に遭遇しない限り，CALC 節の上から下に連続的に計算される．
下記は，「IF 条件」の書式である．

```
@IFC(条件:
! Executable Statements;
@ELSE
! Executable Statements;
@ENDIF
```

ここには，2 つのループの制御文がある．＠FOR が既知のサイズをループするようにしてある．

```
@FOR( 集合 | 条件:! Executable Statements;);
```

＠WHILE で初期値で設定した回数だけループさせる．

```
@WHILE( 条件:! Executable Statements;);
```

次のようにループを中止することもできる．

```
@BREAK
```

出力の文字列を次のような形で書くこともできる．

```
@WRITE( output list);
```

「output list」は明示的な文字列，変数，特定の書式の変数，＠FORMAT( )もしくは行末文字，＠NEWLINE(n)になる．
下記は，＠FORMAT 関数のシンタックスである．

```
@FORMAT( math_expression,  field_description).
```

CALC 節で，参照して解きたい問題は，SUBMODEL として指定する．

例：

```
SUBMODEL mymodel:
! モデル文;
ENDSUBMODEL
```

前の SUMMODEL 節で定義したサブモデルの **mymodel** を，CALC 節の中で@SOLVE を使ってこのサブモデル **mymodel** を解くことができる.

例：

```
@SOLVE( mymodel);
```

Astro-Cosmo 問題の労働時間を 0 から 200 まで，20 刻みで変えて解いた有効フロンティアの計算で説明する.

```
! 有効フロンティア;
SUBMODEL ASTROCOSMO:
MAX = OBJ;
OBJ= 20*A + 30*C;
A <= 60;
C <= 50;
A + 2*C <= LABORAV;
ENDSUBMODEL
DATA:
! 有効フロンティアの計算ポイント数;
NPTS = 11;
! 労働の上限（下限は0）;
UPLIM = 200;
ENDDATA
CALC:
! 出力水準をTERSEOに設定する;
@SET( 'TERSEO', 2);
@WRITE(' Labor Profit', @NEWLINE(1));
! 効率フロンティア上でポイントをループする;
i = 1;
@WHILE( i #LE# NPTS:
LABORAV = UPLIM*(i-1)/(NPTS-1);
! 新しい労働時間制約でモデルを解く;
@SOLVE(ASTROCOSMO);
!OBJの目標値を小数点以下2桁の8文字で表示;
@WRITE(" ", @FORMAT(LABORAV,"8.0f"),
```

```
' ',  @FORMAT(OBJ,"8.2f"),  @NEWLINE(1));
i = i + 1;
); ! @WHILE loopの終了;
ENDCALC
```

これで下記の結果が出力される.

```
Labor Profit
0         0.00
20      400.00
40      800.00
60     1200.00
80     1500.00
100    1800.00
120    2100.00
140    2400.00
160    2700.00
180    2700.00
200    2700.00
```

LINGO のプログラミングの詳細は，オンラインマニュアル（英語版）を参照してください．あるいは，LINDO Japan の HP で会員登録すると，各種の最新版の資料の PDF がダウンロードできる.

第 6 章　製品混合問題　（Product Mix）

6.1 はじめに

製品混合問題は，アストロ・コスモ問題のように概念的には最も理解しやすい．現実では教科書に表われるような簡単な形のものはめったにない．しかし，それは多期間計画問題のような，より大きな問題の重要な構成要素であることが多い．

製品混合問題の特徴は，生産すべき幾つかの製品が，幾つかの有限な資源の利用に関して競合していることである．今 m 個の資源と m 個の製品があるとして，生産計画は，m行ｎ列の技術係数の表で表わすことができる．第ｉ行第ｊ列の係数は，製品ｊを 1 単位生産するのに必要な資源ｉの単位数である．そのような表のある行に並んでいる数値は，単に LP における制約式の係数になる．簡単な製品混合問題では，これらの係数は非負である．さらに，各製品の 1 単位あたりの利益寄与と，各資源の利用可能限度がある．目的関数は，各資源の利用可能限度内で，それらの資源を使いながら利益を最大にするには，各製品をどれだけ生産すべきか，すなわち製品混合を決定することである．

以下に述べる製品混合の例は，LP による定式化を説明するだけではなくて，

①非線形利益関数の記述法と，

②通常の問題は，LP の定式化で幾つかの代替的なやり方がある，

ということを説明している．2 人の人が同じ問題を定式化した場合，明らかに異なる定式化もあるが，両方の解が等しいことが多い．

6.2　製 品 混 合 問 題 の 例

ある工場は，5 つの異なる製品をいかなる組み合わせでも生産できる．各製品は，3 つの機械で次の作業時間で生産できる（単位は分）．

	機械		
製品	1	2	3
A	12	8	5
B	7	9	10
C	8	4	7
D	10	0	3
E	7	11	2

各機械は，週あたり 128 時間利用できる．

製品 A，B，C は市場で，きわめて競争力が強いので，生産した分は 5 ドル，4 ドル，5 ドルで売ることができる．D と E は，週あたり生産された量のうち最初の 20 単位は，各 4 ドルで売ることができる．しかし 20 単位を超えた分は，すべて 1 単位あたり 3 ドルでしか売ることができない．変動労働費は，機械 1 と 2 は 4 ドル/時間であり，機械

3 は 3 ドル/時間である．材料費は製品 A と C は 2 ドル/単位であり，製品 B，D，E は 1 ドル/単位である．そして，会社の利益を最大化したい．

この問題の重要で複雑な部分は，製品 D と E の利益寄与が線形でないことである．この複雑さを取り除くには，今新しい追加的な製品として D2 と E2 を定義するような工夫がいる．そしてこれらは 3 ドル/単位で売れるものとする．そうすると，元の製品 D(=D+D2) と E の売り上げ量に，どんな上限が設定されなければならないか？

変数	定義	単位あたり利益寄与
A	週あたり生産される A の単位数	5-2 = $3
B	週あたり生産される B の単位数	4-1 = $3
C	週あたり生産される C の単位数	5-2 = $3
D	20 単位を超えずに週あたり生産される D の単位数	$3
D_2	20 単位を超えて週あたり生産される D の単位数	$2
E	20 単位を超えずに週あたり生産される E の単位数	$3
E_2	20 単位を超えて週あたり生産される E の単位数	$2
M_1	週あたりの機械 1 の利用時間数	-$4
M_2	週あたりの機械 2 の利用時間数	-$4
M_3	週あたりの機械 3 の利用時間数	-$3

モデルと解は次のようになる．

```
! Maximize revenue minus costs;
[_1]MAX = 3 * A + 3 * B + 3 * C + 3 * D + 2 * D2 + 3 * E
      + 2 * E2 - 4 * M1 - 4 * M2 - 3 * M3;
! Machine time used = machine time made available;
[_2]12*A+7*B+8*C+10*D+10*D2+7*E+7*E2-60*M1 = 0;
[_3]8*A + 9*B + 4*C + 11*E + 11*E2 - 60*M2 = 0;
[_4]5*A+10*B+7*C+3*D+3*D2+2*E +2*E2 - 60*M3=0;
[_5]    D <= 20;  ! Max sellable at high price;
[_6]    E <= 20;
!Machine availability;
[_7]    M1 <= 128;
[_8]    M2 <= 128;
[_9]    M3 <= 128;
END
```

最初の 3 つの制約（[_2]～[_4]）は，機械の使用時間（単位は「分」）を生産した製品の単位数の関数で表している．次の 2 つの制約([_5]～[_6])は，高利益で売る D と E

の生産単位数の上限を与えている．また最後の 3 つの制約は，利用できる機械の時間数についての上限を与えている．

制約 2 はまず最初に次のように書くことができる．

$$\frac{12A+7B+8C+10D+10D2+7E+7E2}{60}=M1$$

これに 60 をかけて M1 を左側にもってくると制約[_2]が得られる．そして，解は次のようになる．

```
 Optimal solution found at step:         4
Objective value:                1777.625
Variable           Value        Reduced Cost
       A        0.0000000            1.358334
       B        0.0000000            0.1854168
       C         942.5000            0.0000000
       D        0.0000000            0.1291668
      D2        0.0000000            1.129167
       E         20.00000            0.0000000
      E2        0.0000000            0.9187501
      M1         128.0000            0.0000000
      M2         66.50000            0.0000000
      M3         110.6250            0.0000000
     Row    Slack or Surplus      Dual Price
       1         1777.625            1.000000
       2        0.0000000            0.2979167
       3        0.0000000            0.6666667E-01
       4        0.0000000            0.5000000E-01
       5         20.00000            0.0000000
       6        0.0000000            0.8125000E-01
       7        0.0000000             13.87500
       8         61.50000            0.0000000
       9         17.37500            0.0000000
```

解の解釈は簡単である．高利益を得ることができる E の単位数が最も利益に寄与するため，制限一杯の 20 単位生産することになる．その後，製品 C が利益率が高いので，C を機械 1 の容量制限一杯作る．この問題は，幾つかの定式化が可能である．これらの代替的な定式化は，すべて正しいが，その制約や変数の数が異なる．例えば，次の制約

```
8A+9B+4C+11E+11E2-60M2=0
```

は次のように書き直すことができる．

M2 = (8A+9B+4C+11E+11E2) /60

そしてM2に代わって右側の式を代入する．右側の式は常に非負であるから，M2 に関する非負制約は自動的に満足される．従って，若干の計算をいとわなければ，この問題からM2とその制約を除去できる．このような議論はM1とM3にも適応できるので，次のモデルが得られる．

```
MAX = 1.416667*A+1.433333*B+1.85*C+2.183334*D+ 1.183333*D2+ 1.7*E + .7*E2;
! Machine time used = machine time made available;
12*A + 7*B + 8*C + 10*D + 10*D2 + 7*E + 7*E2 <= 7680;
8*A + 9*B + 4*C + 11*E + 11*E2 <= 7680;
5*A + 10*B + 7*C + 3*D + 3*D2 + 2*E + 2*E2 <= 7680;
! Product limits;
D < 20;
E < 20;
```

こうして，より標準的な製品混合のモデルが得られた．ここで全ての制約は，何らかの容量制限である．また，このモデルの解が前のモデルの解と同じである．

```
 Optimal solution found at step:         6
 Objective value:                 1777.625
 Variable           Value        Reduced Cost
        A       0.0000000            1.358333
        B       0.0000000           0.1854170
        C        942.5000           0.0000000
        D       0.0000000           0.1291660
       D2       0.0000000            1.129167
        E        20.00000           0.0000000
       E2       0.0000000           0.9187500
      Row    Slack or Surplus      Dual Price
        1         1777.625            1.000000
        2       0.0000000           0.2312500
        3         3690.000           0.0000000
        4         1042.500           0.0000000
        5         20.00000           0.0000000
        6       0.0000000          0.8125000E-01
```

なまけ者の人は最初の定式化を好み，計算をいとわない人は2番目の定式化をすることになる．

6.3 生産工程選択の製品混合問題

製品混合の特徴は，2つあるいはそれ以上の変数が，同じ製品を生産するための代替的な方法に対応づけられている点である．このような場合，LP は 1 つの製品をどれぐらい生産すべきかを発見するだけではなくて，その製品を生産するための最適な生産工程を選択するためにも使われる．

製品混合問題の 2 番目の特徴は，ある製品についてある特定の量までは生産するという条件である．このような条件を追加すると，その問題はもはや単なる製品混合問題とはいえなくなる．ここで，これら 2 つの特徴を備えた問題を考察しよう．

アメリカメタル組み立て会社（AMFC）は，鋼棒から種々の製品を生産している．まず，初期段階の 1 つに成形工程がある．これは 3 種類の回転機械（B3，B4，B5）で遂行される．これらの特徴は次の表で表わされている．

機械	速度（ﾌｨｰﾄ/ 分）	許容原材料厚さ（インチ）	週当たり利用可能時間数	時間あたり運転の為の労働費用
B_3	150	3/16 ～ 3/8	35	$10
B_4	100	5/16 ～ 1/2	35	$15
B_5	75	3/8 ～ 3/4	35	$17

機械の能力がこのような組み合わせで表われるのはめずらしくない．大量な材料を処理する機械は一般的にスピードは遅い．今週生産すべき製品は 3 種類ある．AMFC は，少なくとも 1/4 インチ材料を 218,000 フィート，3/8 インチ材料を 114,000 フィート，そして 1/2 インチ材料を 111,000 フィート生産する．これら 3 種類の製品の利益寄与/フィートは，労働費用を除くと各 0.017， 0.019， 0.02 である．これらの価格は全ての生産に適用される．例えば，必要生産量より多く生産した場合でも，これが適用される．出荷部門はその材料の厚さにかかわらず，週あたり 600,000 フィートの容量しか取り扱うことができない．この問題の制約と決定変数は何であろうか？明らかにこの問題では，AMFC 社の 3 つの限られた機械資源と出荷部門の取り扱い可能容量という 4 つの制約がある．また，3 製品の必要生産量としての制約がさらに 3 つある．しかし，決定変数についてはもう少し考える必要がある．1/4 インチの材料を生産するにはたった 1 つの方法しかない．また，3/8 インチの材料を生産するには 3 つの生産方法があり，1/2 インチ材料については 2 つの方法がある．従って，少なくとも次のような決定変数を考えることができる．

B34=B3 の機械で生産される 1/4 インチ材料の総長（単位 1,000 フィート）
B38=B3 の機械で生産される 3/8 インチ材料の総長（単位 1,000 フィート）
B48=B4 の機械で生産される 3/8 インチ材料の総長（単位 1,000 フィート）
B58=B5 の機械で生産される 3/8 インチ材料の総長（単位 1,000 フィート）
B42=B4 の機械で生産される 1/2 インチ材料の総長（単位 1,000 フィート）
B52=B5 の機械で生産される 1/2 インチ材料の総長（単位 1,000 フィート）

目的関数としては，労働費を含んだ利益寄与を考えなければならない．そしてこの場合は，次のようになる．

変数	1フィートあたりの利益寄与
B_{34}	0.01589
B_{38}	0.01789
B_{48}	0.01650
B_{58}	0.01522
B_{42}	0.01750
B_{52}	0.01622

機械利用の容量制限を求めるには，1,000 フィートを処理するのに必要な時間数が必要である．機械 B3 の場合，この数字は 1000／｛（60 分／時間）×（150 フィート／分）｝＝0.111111（単位は 1，000 フィートあたり時間）である．同様に B4 と B5 については，1,000 フィートあたり各，0.16667 時間，0.22222 時間である．

従って，この定式化は次のようになる．

```
! (収益 - 費用)の最大化;
MAX = 3 * A + 3 * B + 3 * C + 3 * D + 2 * D2 + 3 * E
     + 2 * E2 - 4 * M1 - 4 * M2 - 3 * M3;
! 使用された機械時間=利用可能な機械時間;
12*A + 7*B + 8*C + 10*D + 10*D2 + 7*E +  7*E2 - 60*M1 = 0;
8*A + 9*B + 4*C + 11*E + 11*E2 - 60*M2 = 0;
5*A + 10*B + 7*C + 3*D + 3*D2 + 2*E + 2*E2 - 60*M3=0;
   D <= 20;  ! 販売可能な最高価格;
   E <= 20;
! 利用可能な機械時間;
   M1 <= 128;
   M2 <= 128;
   M3 <= 128;
END
```

出力は次のとおりである．

```
Global optimal solution found.
Objective value:     1777.625
Variable           Value        Reduced Cost
       A        0.000000            1.358333
       B        0.000000           0.1854167
       C        942.5000            0.000000
```

D	0.000000	0.1291667
D2	0.000000	1.129167
E	20.00000	0.000000
E2	0.000000	0.9187500
M1	128.0000	0.000000
M2	66.50000	0.000000
M3	110.6250	0.000000

Row	Slack or Surplus	Dual Price
1	1777.625	1.000000
2	0.000000	0.2979167
3	0.000000	0.6666667E-01
4	0.000000	0.5000000E-01
5	20.00000	0.000000
6	0.000000	0.8125000E-01
7	0.000000	13.87500
8	61.50000	0.000000
9	17.37500	0.000000

最後の 3 つの制約式を取り除くと，これは単なる製品混合の問題である．

ここで，費用という観点から，最適解を演繹的に導き出してみよう．1/4 インチ製品は B3 機械だけでしか生産できないので，B34 は少なくとも 218 であることが分かる．また，3/8 インチ製品は 1/4 インチ製品よりも，B3 機械を使う方が利益率が高いので，B34 は 218 に等しく，残りのスラックとしては，B38 がその値をとることが分かる．1/2 インチと 3/8 インチ製品は，B4 もしくは B5 のどちらでも生産できる．いずれにしても，1/2 インチの方がこの場合，1 フィートあたりの利益率が高いので，B48 と B58 は，絶対的に必要な量よりは多くならない．そこで問題は，「絶対的に必要な量」はどれだけかということである．3/8 インチ材料の場合には，B4 や B5 で処理するよりは，B3 で処理する方が利益率が高い．従って， 3/8 インチ材料の需要を B3 で満たすことが分かる．しかしそれで不十分ならば，その残りが B4 または B5 にあてがわれることが分かる．具体的には，次のようになる．

まず B34 ＝218 にセットする

そうすると，B3 のスラックは，35 － 218 ×0.11111 ＝10.78 時間となる．これは，3/8 インチ材料を 97,000 フィート生産するのに十分である．従って，結論は次のようになる．

B38 ＝97

3/8 インチ材料の需要の残りは，機械 B4 又は B5 で作られなければならない．3/8 インチ材料の利益寄与は，機械 B5 より B4 で作る方が高いので，この 3/8 インチの残り需

要は，機械 B4 で作るべきと思われる．しかし，ここで留意すべきことは，1/2 インチ材料もまた同じ差で，B5 で作るよりも B4 で作る方が利益率が高い．従って，この 3/8 インチ材料の残りは，B4 と B5 のどちらで作ってもよいことになる．そこで恣意的ではあるが,3/8 インチ需要の残りは,B4 機械を利用することにすると,(B48 ＝17)になる．

残った設備容量は，1/2 インチ製品の生産に使われる．ここで，35 - 17×0.16667 ＝32.16667 時間の設備容量が B4 に残っている．この時点で，出荷容量についても配慮する．そこで計算すると，600 - 218 - 97 - 17＝268 （単位は 1，000 フィート）の出荷容量がまだ残っている．B52 よりも B42 の方が利益率が高いので，これをできるだけ大きくする．すなわち，32.16667／0.16667 ＝193 となる（B42 ＝193）．

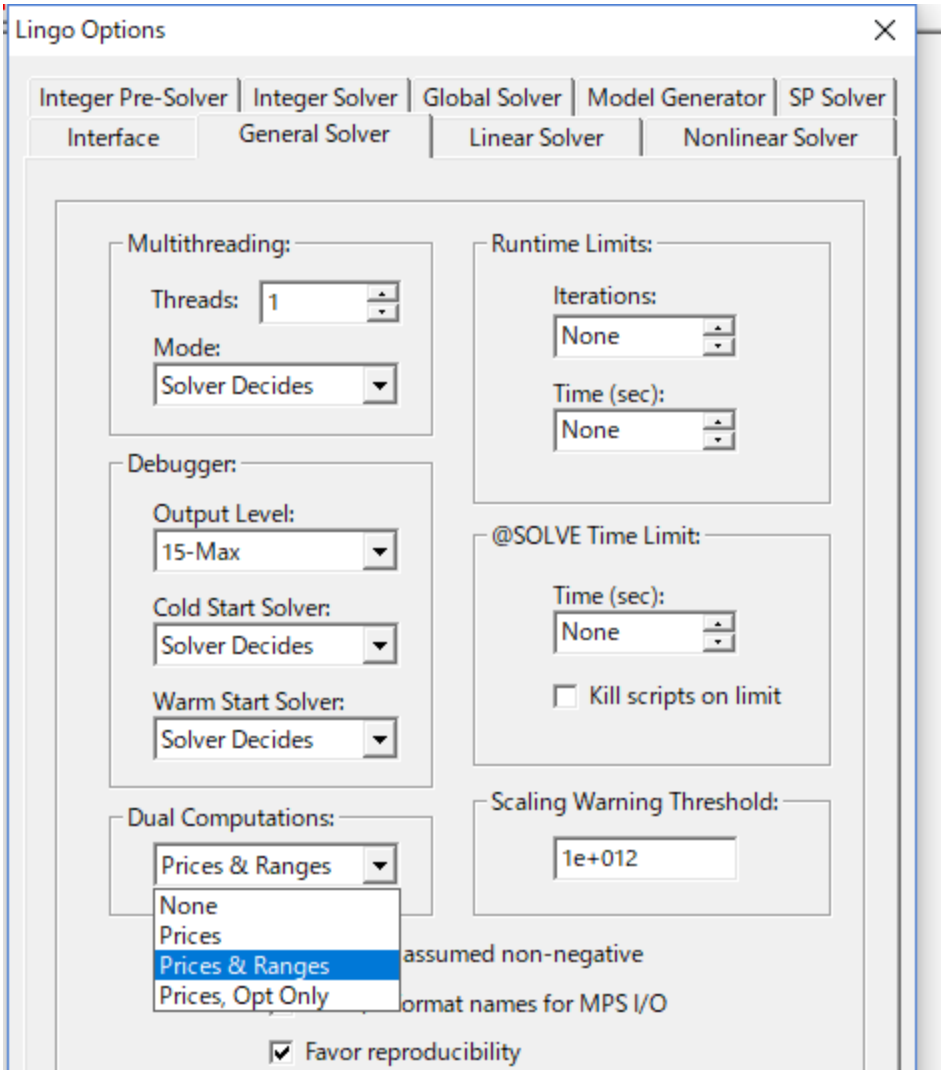
この時点で残っている出荷容量は，268 - 193 ＝75 である（B52 ＝75）．

どんな LP でも，理論的には，上のような経済的な議論で解くことが可能である．しかし，その計算は非常に煩雑であり算術や論理の間違いを犯しやすくなる．

LINGO の解は，上で得られた解と同じであり，以下のようである．

```
Optimal solution found at step:         2
Objective value:                 10073.85
Variable           Value        Reduced Cost
     B34        218.00000          0.000000
     B38         97.00315          0.000000
     B48         16.99685          0.000000
     B58          0.00000          0.000000
     B42        192.99900          0.000000
     B52         75.00105          0.000000
     Row    Slack or Surplus      Dual Price
       1          10073.85          1.000000
       2          0.000000         24.030240
       3          0.000000          7.679846
       4         18.333270          0.000000
       5          0.000000         16.220000
       6          0.000000         -3.000000
       7          0.000000         -1.000000
       8         157.000000          0.000000
```

訳注：以下の範囲分析は，Solver→Options で次の General Solver タブを選ぶ．そして Dual Computation でデフォルトの Prices を Prices & Ranges に変更する．Model の Window で Solve で解を出力し，Solution Report 画面になっているのを，再び Model の Window に戻り，Solver → Range を選ぶことで出力される．



この出力は，決定変数 B34 は現在 15.89 であるが，それに 3 を足した 18.89 を上限とし，下限は無限大まで引いてマイナス無限大になるので，18.89 以下であれば最適解が変わらないことを示す．Righthand Side Ranges は，例えば行 2 の制約式の右辺定数は 35 であるが，それを 1.888520 足したものを上限とし，9.166634 を引いたものを下限とする範囲で変更しても最適解は変わらない．注意すべきは，同時に他の係数を変えてはいけない．

Ranges in which the basis is unchanged:

Objective Coefficient Ranges

Variable	Current Coefficient	Allowable Increase	Allowable Decrease
B34	15.89000	3.000000	INFINITY
B38	17.89000	INFINITY	2.670000
B48	16.50000	1.000000	0.0

B58	15.22000	0.000000	INFINITY
B42	17.50000	0.0	1.000000
B52	16.22000	1.280000	0.0

Righthand Side Ranges

Row	Current RHS	Allowable Increase	Allowable Decrease
2	35.00000	1.888520	9.166634
3	35.00000	12.50043	13.75036
4	35.00000	INFINITY	18.33327
5	600.0000	82.50053	75.00105
6	218.0000	97.00315	16.99685
7	114.0000	157.0000	16.99685
8	111.0000	157.0000	INFINITY

ここでB58は0であるが，その減少費用もまた0であることに留意しよう．その意味は,B58を増加しても減少しても利益に影響を及ぼさない.これはすでに述べたように，3/8インチの需要を満たすにあたって，B48もしくはB58のどちらを使ってもよいというわれわれの観察と一致している．

以下は，集合を用いたモデルである．

```
! これは前の例の集合版です;
MODEL:
SETS:
   MACHINE / B3, B4, B5 / : HPERWK,  TIME;
! これは1日あたりの制約の係数です;
   THICKNESS / FOURTH, EIGH, HALF / : NEED;
! これは生産に必要とされる各厚さの量です;
   METHOD ( MACHINE,  THICKNESS ) : VOLUME, PROFIT,POSSIBLE;
!VOLUMEは変数，PROFITは目的係数，POSSIBLEは与えられた厚さを生成することが可能かどうかを表すブール値です;
ENDSETS
DATA:
! 各マシンで利用可能な時間/週;
    HPERWK =   35, 35,  35;
! 各機械の1000フィートあたりの時間;
      TIME = .11111 .16667 .22222;
! 各製品に必要な金額;
      NEED = 218    114    111;
```

```
! 製品と機械による利益;
    PROFIT = 15.89, 17.89,  0,
             0,     16.5,   17.5,
             0,     15.22, 16.22;
! どの製品がどのマシンで製造できるか;
  POSSIBLE =  1,   1,   0,
              0,   1,   1,
              0,   1,   1;
! 1日あたりの輸送能力;
  SHPERDAY = 600;
ENDDATA
!---------------------------------------;
! 目的関数;
MAX = @SUM( METHOD(I,J): VOLUME(I,J) * PROFIT(I,J));
@SUM( METHOD( K, L): VOLUME( K, L)) <= SHPERDAY;
!This is the max amount that can be made each day;
@FOR( MACHINE( N):
! Maximum time each machine can be used/week. ;
@SUM( THICKNESS( M):
POSSIBLE(N,M) * VOLUME(N,M) * TIME(N))<=HPERWK(N););
@FOR( THICKNESS( Q) :
!Must meet demand for each thickness;
@SUM( MACHINE(P): POSSIBLE(P, Q)*VOLUME(P, Q))>=NEED(Q));
END
```

Solver→Generate→Display model を選ぶと，次の各機械の製造量だけを決定変数とするコンパクトなモデルが表示される．

```
MODEL:
 [_1] MAX= 15.89 * VOLUME_B3_FOURTH + 17.89 * VOLUME_B3_EIGHT + 16.5 *
  VOLUME_B4_EIGHT + 17.5 * VOLUME_B4_HALF + 15.22 * VOLUME_B5_EIGHT + 16.22 *
  VOLUME_B5_HALF;
 [_2] VOLUME_B3_FOURTH + VOLUME_B3_EIGHT + VOLUME_B3_HALF + VOLUME_B4_FOURTH +
  VOLUME_B4_EIGHT + VOLUME_B4_HALF + VOLUME_B5_FOURTH + VOLUME_B5_EIGHT +
  VOLUME_B5_HALF <= 600;
 [_3] 0.11111 * VOLUME_B3_FOURTH + 0.11111 * VOLUME_B3_EIGHT <= 35;
 [_4] 0.16667 * VOLUME_B4_EIGHT + 0.16667 * VOLUME_B4_HALF <= 35;
 [_5] 0.22222 * VOLUME_B5_EIGHT + 0.22222 * VOLUME_B5_HALF <= 35;
```

```
[_6] VOLUME_B3_FOURTH >= 218;
[_7] VOLUME_B3_EIGHT + VOLUME_B4_EIGHT + VOLUME_B5_EIGHT >= 114;
[_8] VOLUME_B4_HALF + VOLUME_B5_HALF >= 111;
END
```

第 7 章　被覆・人員配置・分割問題

7.1　はじめに

被覆問題は,サービス産業でよく利用される.要求される集合がカバー(満たされる)されることに特徴がある．種々の活動があるが，そのうちの幾つかに対応することで要求がみたされる．この問題を言葉で表すと次のようになる．目的関数は，最小の費用で要求を満たす活動水準の組合せを選択することである.種々の問題の活動と要求の例を以下に示す.

問題	要求	活動
要員計画	日あるいは週単位で要求される人数.	労働あるいは交替のパターン．これらのパターンは，幾つかの期間の要求を満たす.
ルーティング	訪問すべき顧客	幾つかの顧客を訪問する，さまざまな道順.
紙，木材，鋼板，服地などの原材料から製品の切断	最終製品のサイズ要求	種々の最終製品のサイズを原材料から切り出す切断のパターン．各パターンは，最終製品の全てではない，幾つかの要求を満たす.

次の節で，これらの問題の幾つかをより詳細に紹介する.

7.1.1 人員配置問題

サービス施設における経営の主要部分は，スケジューリングと人員配置である．すなわち，何人の人間を使うかを決定することである．この問題は，電話会社のオペレータ部門，有料道路，大きな病院などのサービスを供給する施設において存在する．この解のプロセスは 3 つの部分からなる.

② 1 日の時間あたり，または 1 週間の各曜日に必要とされる人数の予測.

② 個人の働ける日と労働協約に基づき働くことが可能なシフト・パターンの識別.

③ 各シフト・パターンで働く人数の決定，すなわち費用が最小かつ各期間で働く人の総人数が，①で決定された要求を満たすこと.

これら 3 ステップはどれも難しいが，LP は③を解決することを助けてくれる.

人員配置問題の最初の論文の一つが Edie(1954) である．彼はニューヨークの港湾当局が管理する料金所の職員の配置方法を開発した．Edie の論文は古いけれども，彼の議論は非常に適切で完全である．概要を紹介する．「試みはリンカーン・トンネルで行なわれた．各料金徴集員のブースの割り当て，交替時間およびスケジュールに厳しく従うように指示されたメモ用紙が渡された．・・・集金の動作やブースの開閉が，通行料の集金が自然に行えるよう指示された．ブース数はわずかに余分で，過度に多すぎない．・・・」

7.1.2　北東有料道路の人員配置問題

シカゴ郊外の北東有料道路の料金所は，24 時間に以下の人員を必要とする．

時間	必要とする人員
0 時から午前 6 時	2
午前 6 時から 10 時	8
午前 10 時から正午	4
正午から午後 4 時	3
午後 4 時から 6 時	6
午後 6 時から 10 時	5
午後 10 時から 24 時	3

料金所の人間は 4 時間働き，1 時間休み，又 4 時間働く．何時から働いてもよい．目的関数は，雇う人数を最小化する LP モデルを定式化する．

(1) 北東有料道路問題の定式化とその解

決定変数を定義する．

x_1 ＝　真夜中から働く人の数

x_2 ＝　午前 1 時から働く人の数

・

・

x_{24}＝　午後 11 時から働く人の数

各時間帯に対し 1 つの制約が存在し，その時間における人員数は要求される数より多い必要がある．目的関数は雇われる人数の最小化である．

$MIN=x_1+ x_2+ x_3+ \cdots + x_{24}$；

ST

$x_1 +x_{24}+ x_{23}+ x_{22}+ x_{20}+ x_{19}+ x_{18}+ x_{17} >= 2$；(夜中から午前 1 時)

$x_2 + x_1 + x_{24}+ x_{23}+ x_{21}+ x_{20}+ x_{19}+ x_{18} >=2$；(午前 1 時から 2 時)

.

$x_7 + x_6 + x_5 + x_4 + x_2 + x_1 + x_{24}+ x_{23} >=8$ ；(午前 6 時から 7 時)

.

$x_{24}+ x_{23}+ x_{22}+ x_{21}+ x_{19}+ x_{18}+ x_{17}+ x_{16} \geqq 3$ ；(午後 11 時から夜中)

PICTURE コマンドから，シフトの間の 1 時間の休みの影響が分かる．

制約	x_1	x_2	x_3	x_4	…	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	RHS
0 時–	1					1	1	1	1		1	1	1	≥2
1 時–	1	1					1	1	1	1		1	1	≥2
2 時–	1	1	1					1	1	1	1		1	≥2
3 時–	1	1	1	1					1	1	1	1		≥2
4 時–		1	1	1						1	1	1	1	≥2
5 時–	1		1	1							1	1	1	≥2
6 時–	1	1		1								1	1	≥8
7 時–	1	1	1										1	≥8
8 時–	1	1	1	1										≥8
9 時–		1	1	1										≥8
10 時			1	1										≥4
11 時				1										≥4
12 時														≥3
13 時														≥3
14 時														≥3

(2) 集合による定式化

LINGO でこの問題を集合で定式化すると，コンパクトになる．2 つの集合がある．最初は 1 日 24 時間であり，もう一つは 9 時間交替である．変数 X の値の剰余を求める @WRAP 関数を使用している．

```
MODEL:! 24 時間シフトスケジューリング;
 SETS:! 各シフトは 4 時間労働，1 時間休憩，4 時間労働です;
  HOUR/1.24/: X, NEED;
 ENDSETS
 DATA:
  NEED=2 2 2 2 2 2 8 8 8 8 4 4 3 3 3 3 6 6 5 5 5 5 3 3;
 ENDDATA
 MIN = @SUM( HOUR(I): X(I));
       @FOR( HOUR( I):
! 1 時に勤務している人は，8 時間前から 1 時までに仕事を始めた人です;
 @SUM(HOUR(J)|(J#LE#9)#AND#(J#NE#5): X(@WRAP((I-J+1), 24)))>= NEED(I));
END
```

非零の解は以下の通りで，目的関数は 15.75 である．

x_2 = 5	x_5 = 0.75	x_{11} = 1	x_{16} = 1
x_3 = 0. 75	x_6 = 0.75	x_{14} = 1	x_{17} = 1
x_4 = 0. 75	x_7 = 0.75	x_{15} = 2	x_{18} = 1

この答えは小数なので役にたたない．15．75 人は，整数解を必要とするから少なくとも 16 人を雇わなければならない．これを解決するため，次の@GIN 関数を END の前に挿入すればよい：@FOR(HOUR: @GIN(X));

これを解くと，次の非零の整数値をもち，目的関数が 16 の解を得る．

x_2 = 4	x_5 = 1	x_{14} = 1	x_{17} = 2
x_3 = 1	x_6 = 1	x_{15} = 1	x_{18} = 1
x_4 = 1	x_7 = 1	x_{16} = 2	

訳注： Generate でモデルを生成すると次の簡単なモデルが生成される．制約式[2]は 1 時に働く人の合計が 2 人以上であることを示す．17 時から 20 時から働く人と，22 時から 1 時まで働く人が 2 人以上であることを示す．21 時から働く人が含まれないのは 1 時に休憩を取るからである．集合モデルの意味が分からなくても，どこを治せば汎用で利用できるかを理解することが重要である．そして，Generate で生成された自然表記で意味を理解すればよい．第 1 世代の LINDO は，この自然表記でモデルを作る必要があった．大きなモデルを作るには時間がかかり，異なったモデルごとに多大な作業がいり，間違いも起こる．しかし汎用の集合モデルは，簡単に異なったモデルに DATA 節の値を変えるだけで対応できる．

```
MODEL:

[_1]MIN=X_1+X_2+X_3+X_4+X_5+X_6+X_7+X_8+X_9+X_10+X_11+X_12+X_13+X_14+X_15+X_16+X_17+
X_18+X_19+X_20+X_21+X_22+X_23+X_24;

[_2]X_1+X_17+X_18+X_19+X_20+X_22+X_23+X_24>=2;

[_3]X_1+X_2+X_18+X_19+X_20+X_21+X_23+X_24>=2;

[_4]X_1+X_2+X_3+X_19+X_20+X_21+X_22+X_24>=2;

[_5]X_1+X_2+X_3+X_4+X_20+X_21+X_22+X_23>=2;

[_6]X_2+X_3+X_4+X_5+X_21+X_22+X_23+X_24>=2;

[_7]X_1+X_3+X_4+X_5+X_6+X_22+X_23+X_24>=2;

[_8]X_1+X_2+X_4+X_5+X_6+X_7+X_23+X_24>=8;

[_9]X_1+X_2+X_3+X_5+X_6+X_7+X_8+X_24>=8;

[_10]X_1+X_2+X_3+X_4+X_6+X_7+X_8+X_9>=8;

[_11]X_2+X_3+X_4+X_5+X_7+X_8+X_9+X_10>=8;

[_12]X_3+X_4+X_5+X_6+X_8+X_9+X_10+X_11>=4;
```

```
[_13]X_4+X_5+X_6+X_7+X_9+X_10+X_11+X_12>=4;

[_14]X_5+X_6+X_7+X_8+X_10+X_11+X_12+X_13>=3;

[_15]X_6+X_7+X_8+X_9+X_11+X_12+X_13+X_14>=3;

[_16]X_7+X_8+X_9+X_10+X_12+X_13+X_14+X_15>=3;

[_17]X_8+X_9+X_10+X_11+X_13+X_14+X_15+X_16>=3;

[_18]X_9+X_10+X_11+X_12+X_14+X_15+X_16+X_17>=6;

[_19]X_10+X_11+X_12+X_13+X_15+X_16+X_17+X_18>=6;

[_20]X_11+X_12+X_13+X_14+X_16+X_17+X_18+X_19>=5;

[_21]X_12+X_13+X_14+X_15+X_17+X_18+X_19+X_20>=5;

[_22]X_13+X_14+X_15+X_16+X_18+X_19+X_20+X_21>=5;

[_23]X_14+X_15+X_16+X_17+X_19+X_20+X_21+X_22>=5;

[_24]X_15+X_16+X_17+X_18+X_20+X_21+X_22+X_23>=3;

[_25]X_16+X_17+X_18+X_19+X_21+X_22+X_23+X_24>=3;

@GIN(X_1);@GIN(X_2);@GIN(X_3);@GIN(X_4);@GIN(X_5);@GIN(X_6);@GIN(X_7);@GIN(X_8);@GIN
(X_9);@GIN(X_10);@GIN(X_11);@GIN(X_12);@GIN(X_13);@GIN(X_14);@GIN(X_15);@GIN(X_16);@
GIN(X_17);@GIN(X_18);@GIN(X_19);@GIN(X_20);@GIN(X_21);@GIN(X_22);@GIN(X_23);@GIN(X_2
4);

 END
```

この種の人員配置問題の最も大きい例の 1 つは，コールセンターである．例えば，クレジットカードのサービスセンター，カタログ通販のための電話受けつけ，などである．オマハやネブラスカの人口のかなりの人が，コールセンターで働いている．コールセンターの典型的な交替パターンは，8 時間労働において，15 分の休憩，30 分の昼休み，およびもう 15 分の休憩から構成されている．

7.1.3　人員配置の付加的機能

人員配置の実現には，先の 3 ステップに加えて，名簿作成，作業パターンの識別，作業パターンの選択が必要である．名簿作成では，特定の個人が特定の作業パターンの仕事をする．例えば航空会社では，個人（例えば，パイロット）は選ばれた仕事のパターンを選択する．人材派遣では，例えば最初の 1 時間に，少なくとも 3 人のスペイン語と 4 人の英語を話すスピーカーが必要かもしれない．派遣従業員のスキルが異なる場合があり，ある人は英語だけ，他の人は英語とスペイン語の両方に通じている．例えばメー

ルの処理では，繁忙期に優先度の低いものの処理を遅らせて，人員不足に対処することもある．

ほとんど全ての状況下で，需要はややランダムなので，要員要件は柔軟に対応すべきです．期間中に少なくとも 10 人が必要なのに，11 人雇うこともある．この場合，予想に反して緊急の仕事が入った場合，余剰人員で対応できる．18 章の待ち行列理論は，余剰人員の限界利益の見積りに頻繁に使用される．

7.2　分割とパターン選択問題

製紙産業では，製品は最初に経済的な生産サイズで作られる．これらのサイズは，最終製品として小さなサイズに分割される．最小費用でより小さなサイズにどのように分割するかの決定が，「分割問題」である．「一次元分割問題」の例として，機械の性能で材料が 72 インチの幅を要求すると仮定する．これをより小さな幅に分割する方法はいろいろあるが，そのうちの 2 つを**図 7.1** に示す．

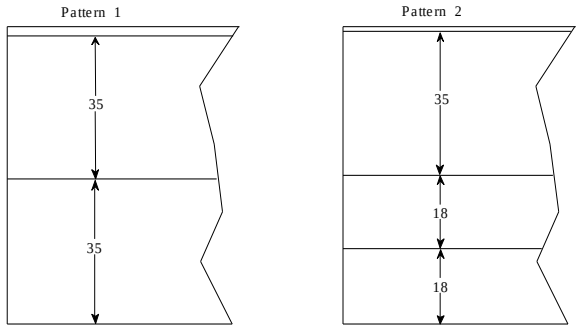


図 7.1　分割パターンの例

パターン 1 の端の無駄は 2 インチ(72 − 2 × 35 = 2)であり，パターン 2 の無駄は 1 インチ(72 − 2 × 18 − 35 = 1)である．しかしパターン 2 は，18 インチの長さは 35 インチの材料の長さに比べて 2 倍でないと有効ではない．よって，端の無駄の妥協点を見つけなければならない．

分割問題の解は，3 ステップに分けることができる．

①最終的に必要とされる幅の推測

②より小さな幅に分割する際の可能なパターンの決定

③ ②の各パターンのどれだけが必要かの決定．すなわち①における要求が最小費用で満たされるかどうか．LP はステップ③の実行に使用できる．

多くの巨大製紙会社は，分割問題の解法をベースにした LP 解を持っている．実際の分割問題は，端の無駄または結果の無駄の妥協に付け加えて様々な費用因子を含んでいる．LP の有効性は，これらの因子の重要性に依存している．次の例は，複雑な費用因子を伴なわない分割問題の基本的な特徴を示している．

7.2.1　クルドット社の分割問題

クルドット社は非常に幅広い家庭用器具, すなわち冷蔵庫やストーブなどを生産している. 原材料費の重要な部分は, 鉄板の購入である. 現在, 鉄板はコイルの形で購入し, 72, 48, 36 インチの 3 種類の幅がある. 製造工程では 8 種類の幅（60, 56, 42, 38, 34, 24, 15, 10 インチ）が要求される. 鉄板は同じ品質と厚さが要求される. 続いて起こる問題は, 無駄を取り除くことである. 例えば 72 インチ幅のコイルを 38 インチ幅のコイルと 15 インチ幅のコイル 2 つに切り離すやり方がある. ここで, 無駄な 4 インチ幅のコイルが出てしまうことになる. 3 つの異なった幅の原材料の 1 フィート当りの価格は, 36 インチ幅が 15 セント, 48 インチ幅が 19 セント, 72 インチ幅が 28 セントである. 単純計算で, 1 インチ×1 フィートあたりの価格は, 36 インチ幅が 15/36＝0. 41667 セント／(インチ×フィート), 48 インチ幅が 0.395833, 72 インチ幅が 0.3888889 である. コイルは適切な方法で切り離される. 3 つの幅の原材料を切り離す効率的なやり方を以下に表にしてある. 例えばパターン C4 は 72 インチ幅のコイルを, 24 インチ幅 1 つと 10 インチ幅 4 つ, 無駄な 8 インチが残るように切り離すことである.

この計画期間で, 要求される様々な幅の長さは, 次の通りである.

幅(inch)	60	56	42	38	34	24	15	10
要求フィート数	500	400	300	450	350	100	800	1000

この期間に使用できる原材料は, 72 インチのコイルで 1600 フィート, 48 インチと 36 インチで 10, 000 フィートである. 様々な幅や要求を満たし, 最小費用になるように分割されるパターンのフィート数を決定するモデルを定式化せよ. あなたは 36 インチ幅の材料の使用量を, 前もって推測できるだろうか?

7.2.2 クルドット分割問題の定式化とその解

分割パターン表に現われる記号 A1, A2,..,E4 を, それに対応するパターンのフィート数を表わすものとする.

原材料の切断パターン

要求される幅の切断数									
パターン名称	60	56	42	38	34	24	15	10	無駄
	72-インチ原材料								
A_1	1	0	0	0	0	0	0	1	2
A_2	0	1	0	0	0	0	1	0	1
A_3	0	1	0	0	0	0	0	1	6
A_4	0	0	1	0	0	1	0	0	6
A_5	0	0	1	0	0	0	2	0	0
A_6	0	0	1	0	0	0	1	1	5
A_7	0	0	1	0	0	0	0	3	0

A_8	0	0	0	1	1	0	0	0	0
A_9	0	0	0	1	0	1	0	1	0
B_0	0	0	0	1	0	0	2	0	4
B_1	0	0	0	1	0	0	1	1	9
B_2	0	0	0	1	0	0	0	3	4
B_3	0	0	0	0	2	0	0	0	4
B_4	0	0	0	0	1	1	0	1	4
B_5	0	0	0	0	1	0	2	0	8
B_6	0	0	0	0	1	0	1	2	3
B_7	0	0	0	0	1	0	0	3	8
B_8	0	0	0	0	0	3	0	0	0
B_9	0	0	0	0	0	2	1	0	9
C_0	0	0	0	0	0	2	0	2	4
C_1	0	0	0	0	0	1	3	0	3
C_2	0	0	0	0	0	1	2	1	8
C_3	0	0	0	0	0	1	1	3	3
C_4	0	0	0	0	0	1	0	4	8
C_5	0	0	0	0	0	0	4	1	2
C_6	0	0	0	0	0	0	3	2	7
C_7	0	0	0	0	0	0	2	4	2
C_8	0	0	0	0	0	0	1	5	7
C_9	0	0	0	0	0	0	0	7	2
48-インチ原材料									
D_0	0	0	1	0	0	0	0	0	6
D_1	0	0	0	1	0	0	0	1	0
D_2	0	0	0	0	1	0	0	1	4
D_3	0	0	0	0	0	2	0	0	0
D_4	0	0	0	0	0	1	1	0	9
D_5	0	0	0	0	0	1	0	2	4
D_6	0	0	0	0	0	0	3	0	3
D_7	0	0	0	0	0	0	2	1	8
D_8	0	0	0	0	0	0	1	3	3
D_9	0	0	0	0	0	0	0	4	8
36-インチ原材料									
E_0	0	0	0	0	1	0	0	0	2
E_1	0	0	0	0	0	1	0	1	2

E_2	0	0	0	0	0	0	2	0	6
E_3	0	0	0	0	0	0	1	2	1
E_4	0	0	0	0	0	0	0	3	6

計算のために，以下のものをつけ加えて定義することは有効である．

T1＝72インチパターンのカットされたフィート数

T2＝48インチパターンのカットされたフィート数

T3＝36インチパターンのカットされたフィート数

W1＝72インチパターンからの無駄のインチ×フィート

W2＝48インチパターンからの無駄のインチ×フィート

W3＝36インチパターンからの無駄のインチ×フィート

X1＝60インチ幅の過剰カットフィート数

X2＝56インチ幅の過剰カットフィート数

・

・

・

X8＝10インチ幅の過剰カットフィート数

目的関数を何にすべきか直接には分からないかも知れない．たとえば，各パターンに対するフィートあたりの無駄の費用を計算することを試み，全ての無駄の費用を最小化したいと思いがちである．すなわち，

```
Min=0.3888891*W1 + 0.395833*W2 + 0.416667*W3;
```

しかし，このような目的関数では，無駄は非常に少ないが費用がかかる解になる．これは特に平方インチあたりの費用が，全ての幅に対して同じでないときに起きる．より適当な目的関数は，次のように全費用を最小にすることである．

```
MIN = 28 * T1 + 19 * T2 + 15 * T3;
```

これをモデルに組み込むと次のモデルになる．

```
MODEL:
 SETS:
!各原材料は， 幅， 使用量，無駄の計，単位費用，無駄の費用， 利用可能量;
 RM: RWDTH,T, W, C, WCOST, S;
! 最終製品は，幅，要求単位，余分の製造などをもつ;
 FG: FWDTH, REQ, X;
 PATTERN: USERM, WASTE, AMT;
 PXF( PATTERN, FG): NUM;
 ENDSETS
 DATA:
! 原材料幅;
```

```
  RM =    R72       R48       R36;
RWDTH=     72        48        36;
   C =    .28       .19       .15;
WCOST= .00388889 .00395833 .00416667;
   S =    1600     10000     10000;
!最終製品幅;
  FG = F60 F56 F42 F38 F34 F24 F15 F10;
FWDTH=  60  56  42  38  34  24  15  10;
  REQ= 500 400 300 450 350 100 800 1000;
!各パターンが使用するインデックスR. M. ;
USERM = 1 1 1 1 1 1 1 1 1 1
        1 1 1 1 1 1 1 1 1 1
        1 1 1 1 1 1 1 1 1
        2 2 2 2 2 2 2 2 2 2
        3 3 3 3 3;
! 各F.G.の幾つがR.M.パターンを持つか;
NUM=   1 0 0 0 0 0 0 1
       0 1 0 0 0 0 1 0
       0 1 0 0 0 0 0 1
       0 0 1 0 0 1 0 0
       0 0 1 0 0 0 2 0
       0 0 1 0 0 0 1 1
       0 0 1 0 0 0 0 3
       0 0 0 1 1 0 0 0
       0 0 0 1 0 1 0 1
       0 0 0 1 0 0 2 0
       0 0 0 1 0 0 1 1
       0 0 0 1 0 0 0 3
       0 0 0 0 2 0 0 0
       0 0 0 0 1 1 0 1
       0 0 0 0 1 0 2 0
       0 0 0 0 1 0 1 2
       0 0 0 0 1 0 0 3
       0 0 0 0 0 3 0 0
       0 0 0 0 0 2 1 0
       0 0 0 0 0 2 0 2
```

```
        0 0 0 0 0 1 3 0
        0 0 0 0 0 1 2 1
        0 0 0 0 0 1 1 3
        0 0 0 0 0 1 0 4
        0 0 0 0 0 0 4 1
        0 0 0 0 0 0 3 2
        0 0 0 0 0 0 2 4
        0 0 0 0 0 0 1 5
        0 0 0 0 0 0 0 7
        0 0 1 0 0 0 0 0
        0 0 0 1 0 0 0 1
        0 0 0 0 1 0 0 1
        0 0 0 0 0 2 0 0
        0 0 0 0 0 1 1 0
        0 0 0 0 0 1 0 2
        0 0 0 0 0 0 3 0
        0 0 0 0 0 0 2 1
        0 0 0 0 0 0 1 3
        0 0 0 0 0 0 0 4
        0 0 0 0 1 0 0 0
        0 0 0 0 0 1 0 1
        0 0 0 0 0 0 2 0
        0 0 0 0 0 0 1 2
        0 0 0 0 0 0 0 3;
ENDDATA
! 使用原材料の費用;
MIN = TCOST;
 TCOST = @SUM(RM(I): C(I)*T(I) );
@FOR( RM( I):
  T( I) = @SUM( PATTERN( K)| USERM(K) #EQ# I: AMT( K));
! 原材料の供給制約;
     T(I) <= S(I););
@FOR( FG(J):
   @SUM(PATTERN(K): NUM(K, J)*AMT(K)) = REQ(J) + X(J););
! 整数解を得るためにこれをオンにする;
!@FOR( PATTERN(K): @GIN(AMT(K)));
```

```
! 廃棄物関連の計算;
! 各パターンに関連付けられた廃棄物を計算する;
 @FOR( PATTERN(K):
   WASTE(K) =RWDTH(USERM(K))-@SUM(FG(J): FWDTH(J)*NUM(K, J)););
! 個の解における各 R. M の廃棄物;
 @FOR( RM( I):
        W(I) = @SUM( PATTERN( K)| USERM(K) #EQ# I: WASTE(K)*AMT( K)););
!無駄な費用の計算;
 TOTWASTE = @SUM( RM(I): WCOST(I)*W(I) );
END
```

2 つの異なった目的関数の異なった解は，以下の表で比較できる．

分割問題の解		
ゼロでないパターン	**無駄を最小にした解(フィート)**	**総費用を最小にした解（フィート）**
A_1	500	500
A_2	400	400
A_5	200	171.4286
A_7	100	128.5714
A_8	350	350
A_9	50	3.571429
B_8	0	32.14286
C_9	0	14.28571
D_1	150	96.42857
D_3	25	0
無駄費用	$5.44	$5.55
総費用	$2348.00	$466.32
X_4	100.000	0
X_6	19650	0
T_1	1600	1600
T_2	10000	96.429
T_3	0	0

この解の鍵となる相違は，「無駄を最小にする解」は 48 インチの材料（T2)をより多く使用し，端の無駄が最小になる方法で分割している．さらに 38 インチ幅と 24 インチ幅の材料を必要以上に作っている．しかしながら，目的関数はそれを無駄としては数え

ていない．どちらの解も，数値が整数でない．実際問題で多くの人は最も近い整数に丸める．ここで@GIN で整数指定すると，費用最小化の解は$466．34 に増える．

7.2.3　分割問題の一般化

巨大な分割問題で，全ての可能性のあるパターンを生成することは非現実的である．最適解で，非常に高い確率で現れるパターンのみを生成する効率的な方法が存在する．この手続きを説明することは，この節の範囲を越える．しかし巨大な問題で，それが重要になる．Dyckhoff(1981)による，この問題の他の定式化を 18 章で紹介している．しかし，制約式は非常に巨大になる．

複雑な分割問題で，さらに以下の費用の考察が重要になる．

①特定のパターンを生成するための固定費用：これは，機械使用時間，労働力，その他からなる．これは，より少ないパターンの解を魅力あるものとする．

②余剰または最終的な無駄が有する価値：例として，当該期間における無駄は，次の期になんらかの需要があるかもしれない．

③機械使用費用：機械を操作する費用は，普通は材料の状態と独立である．これは，幅広の材料を切断する解に有利である．

④規定の量に達しない費用：ある産業で，例えば±5%の供給が許されるかもしれない．最少限の許される量で生産する費用は，既定利益になる．

⑤特別の生産をする材料：もし異なった材料（例えば異なった厚さ，品質，表面の仕上げ，型など）を要求しても，2 つの異なった製品を同じパターンで作ることは不可能かもしれない．

⑥グレードアップの費用：特定需要幅に対し，要求されるものよりグレードの高い材料に変えることで，セットアップ，縁または先端の無駄を減少させることが可能なこともある．

⑦注文による分割費用：もし要求される幅が幾つかのパターンから生成されるならば，出荷に対し，同時に出荷する異なったロットを運ぶための統一費用が存在するかもしれない．

⑧材料幅変更費用：パターンの変更のみを含むセットアップは，普通パターンの変更と材料幅の変更の両方を含むセットアップに比べて少ない時間で済む．これは，少ない材料幅を使用する解を魅力あるものにする．

⑨縁の無駄についての許容値：ある材料は，幅の非常に狭い縁の無駄の扱いが非常にむずかしい．従って，縁の無駄のない，または最小値（例えば 2 センチメートル）を越えるパターンに集中することが好まれる．

⑩納期と順序：ある種の需要は，他が緊急性の少ない場合，すぐに対応が必要である．緊急性が高いか優先度の高いパターンは，すばやく対応する必要がある．緊急性の高い需要が優先度の低いパターンと同じなら，これにすぐに対応するのは難しくなる．

⑪**在庫制約：** 普通顧客の発注は，顧客の全ての需要が出荷可能になるまで出荷されない. このように, 特定顧客の需要が, できるだけ少ないパターンであることを希望する. 顧客が全てのパターンの製品を必要なら, 顧客の注文はあらゆるパターンが動くまで出荷されない．このように，全てのパターンが動くまで，相当な仕掛品がでてくる．

⑫**1 セット・パターンに対する制限：** 製紙のような産業では，パターンを準備することと関連する明示的な費用はないが, パターン変化が起きると対応できる率に制限がある. パターンの変更におよそ 15 分かかるかもしれないが，この仕事の多くが主要機械を止めてされる．1 本のロールの生産に，10 分かかるかもしれない．あまりに多くの 1 セットのパターンが稼動するなら, 主要機械はパターンの変更が完了されるのを待たなければならない.

⑬**パターン規制：** ある応用では，最終製品の幅の合計数やパターンの小さな幅の数に制限があるかもしれない. 限られた数の受け取りリールがスリット状の商品を巻き上げるために用いられる場合，最初の規制にあてはまる．第 2 の規制は，ぎりぎりの製品幅のロールが倒れる傾向を持つ紙業界で起こるかもしれない．一つのパターンで，あまりに多くのものを扱いたくない場合である．一部の顧客は，材料の品質がある場所でより高いと感じるので，製品をその特定の場所（例えばセンター）から切りとることを依頼するかもしれない.

⑭**対になっているパターン：**プラスチックのラップ製造で，生産プロセスの特性から，同時に上と下の出力から 2 つの幅をもつ原料を生じる．例えば，下の出力と同じだけ，上で出力することは，本質的に避けられない．類似した状況は，時々紙製造において偶然に起こる．もし欠陥が生産ラインのベルト上で発生すると，その幅の内部にある小幅な紙は使えない．こうして機械は右と左に 2 つの出力幅のものを生じる．

最も手に負えない複雑さは，高く固定された段取り費用，注文分割費用，材料幅変更費用である．もしこれらが無視できないほど重要であるなら，人は手動による解決を強いられる．LP 解は良い解の洞察を与えるが，他の方法で最終的に実行可能な解を決定する．

7.2.4　2 次元分割問題

1 次元の分割問題は，コイル状の原料を分割することに関係している．基本的な考えは，原料がシート状のものに応用でき，問題はこれをより小さなシートに分割することになる. 例えば, 合板が 48×96 インチの矩形で供給され, 最終的に要求される製品は, 36×50，24×36，20×60，18×30 インチのシートとする．一度あなたが 48×96 のシートを分割する 4 つのより小さなシートの組合せの全ての可能なパターンを数えたなら，問題は前とまったく同じになる.

全ての可能な 2 次元パターンを列挙することは，非常に困難な課題である．実用的な 2 次元の切断問題の 2 つの特徴は，この仕事のサイズを次のように減らすことである：(a)方向性の要求と，(b)「ギロチン切断」要求である．(a)が重要な応用分野は，木や生地の切断である．強度や見た目の理由から，要求の単位は原料で限られた位置

にあるかもしれない．例えば，服の製造業者が方向のある格子縞のスーツを作ることを想像しなさい．よい野球選手は，ヒットするときバットの木目が球に合致していることを知っている．木製品が構造や審美的な目的で使用される場合，木の木目に注意を払われなければならない．ガラスは，方向が重要でない均質な原料の例である．ギロチン切断は，シート材の端から端まで切り取る方法である．

7.3 乗員のスケジューリング問題

航空会社の運営経費の主なものは，乗員の人件費である．大手航空会社の航空機と乗員の管理は複雑な問題である．これらに注目することは価値がある．乗員の年間費用は，典型的な PC の 1 次的費用を簡単に上回るので，より効果的に乗員と飛行機を使用するために PC 資源をつぎこむことは魅力的である．次の小さな問題を以下で議論する．

大手航空会社は，乗員スケジュール問題と呼ばれる要員計画問題に直面する．カバーされる必要条件は，航空会社がある期間（例えば 1 ヵ月）予定しているフライトの乗員の要求を満たすことである．出勤日，特定の乗員は概して同じ航空機に勤務する．問題は，どの飛行機を乗員に割り振るか決定することである．

多くの航空会社でとられる方法は，一般的な要因計画である：①需要要求の確認（すなわちカバーされる飛行）．②1 人の乗員が仕事期間にカバーできる飛行の多数の可能な順序の生成．③ ②で生成された中から，飛行機と乗員も組み合わせを正確に 1 つ選んで，費用を最小にする．ステップ③に IP が使われる．1985 年まで大きな IP を解くのが難しいかったので，大部分の大手航空会社は③を解くため発見的な手続きを使用していた．しかし Marsten，Muller & Killion（1979）は，タイガー航空で非常にうまい IP モデルを記述した．タイガー航空は，乗客が少ないので，結果として IP は経済的に解くことができて，発見的な方法より著しく低い費用の解をえた．この最適化法は，現在大手の航空会社で利用されている．次は，非常に単純化した例である．この例は 10 機の便だけを持つが，主要な航空会社は日に 2000 機以上の便を持つ．

7.3.1 例題

乗員スケジューリング問題の簡単な例として，以下の例を考える．Sayre-priors 航空会社は，以下のフライトを運行している．

フライト			
フライト番号	出発地	目的地	時間
101	Chicago	Los Angeles	午後
410	New York	Chicago	午後
220	New York	Miami	夜
17	Miami	Chicago	朝
7	Los Angeles	Chicago	午後
13	Chicago	New York	夜
11	Miami	New York	朝

19	Chicago	Miami	夜
23	Los Angeles	Miami	夜
3	Miami	Los Angeles	午後

フライトのスケジュールは，次の**図 7.2** に示す.

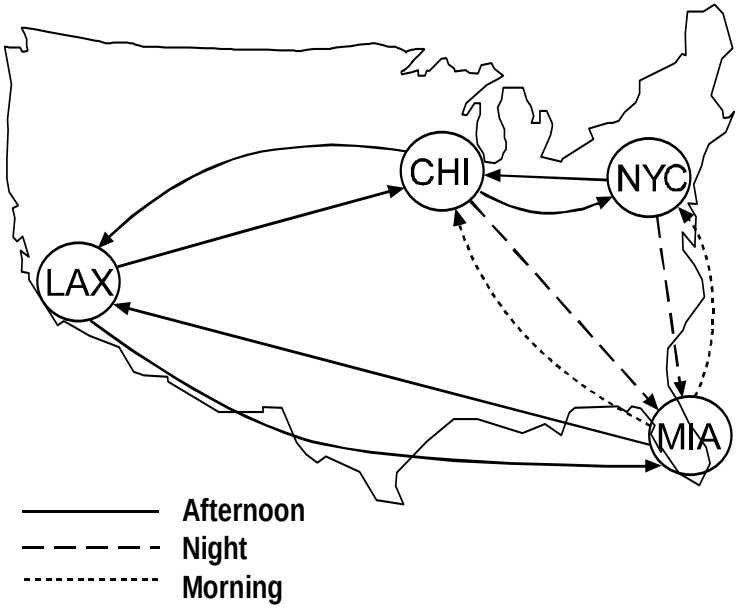


図 7. 2 フライト・スケジュール

フライトを運用する要員は，低費用で乗員を割り当てたいと思っている．基本的な問題は，一人の乗員が一つのフライトを終えた後に，彼が乗り込む次のフライトを決定することである．この問題を理解するのに必要な基本概念は，ツアーの概念である．ツアーの特徴は，次の通りである．

①ツアーは 1 つから 3 つの接続されたフライトからなる．

②ツアーは出発地点で終わると，2000 ドルの費用がかかる．

③ 出発地点以外の場所で終わる行止りツアーは，3000 ドルの費用がかかる．

航空会社の用語では，ツアーはしばしば「ペアリングまたはローテーション」と呼ばれる．以下は，受け入れられるツアーの例である．

ツアー	費用
17, 101, 23	$2,000
220, 17, 101	$3,000
410, 13	$2,000

実際のツアー費用の計算は，上より複雑である．例えば，パイロットの人件費は，飛行中と，フライト間の待機期間と，自宅から離れていることに対する日当などさまざまである．

7.3.2 乗員スケジューリング問題に対する解

この小さな問題で最初に行うことは，全ての可能なツアーを数えあげることである．10 個の 1 回フライト，14 個の 2 回フライトのツアーがある．さらに，出発地点で終わるツアーは，出発地点を区別するかしないかに依存して 37 または 41 個の 3 回フライトのツアーがある．これらのツアーは以下に示される．

表　ツアーのリスト

1 回フライト	費用	2 回フライト	費用	3 回フライト	費用
1. 101	$3,000	11. 101, 23	$3,000	25. 101, 23, 17	$2,000
2. 410	$3,000	12. 410, 13	$2,000	26. 101, 23, 11	$3,000
3. 220	$3,000	13. 410, 19	$3,000	27. 410, 19, 17	$3,000
4. 17	$3,000	14. 220, 17	$3,000	28. 410, 19, 11	$2,000
5. 7	$3,000	15. 220, 11	$2,000	29. 220, 17, 101	$3,000
6. 13	$3,000	16. 17, 101	$3,000	30. 220, 11, 410	$3,000
7. 11	$3,000	17. 7, 13	$3,000	25. 17, 101, 23	$2,000
8. 19	$3,000	18. 7, 19	$3,000	31. 7, 19, 17	$3,000
9. 23	$3,000	19. 11, 410	$3,000	32. 7, 19, 11	$3,000
10. 3	$3,000	20. 19, 17	$2,000	33. 11, 410, 13	$3,000
		21. 19, 11	$3,000	28. 11, 410, 19	$2,000
		22. 23, 17	$3,000	34. 19, 17, 101	$3,000
		23. 23, 11	$3,000	28. 19, 11, 410	$2,000
		24. 3, 23	$2,000	25. 23, 17, 101	$2,000
				35. 23, 11, 410	$3,000
				36. 3, 23, 17	$3,000
				37. 3, 23, 11	$3,000

次の決定変数を定義する．

T_i＝1（ツアー i が採用されたとき），

0（ツアー i が使用されないとき， i ＝1, 2, …, 37）

出発地点で終わる 3 回フライトは，出発地を区別しないことにする．定式化は以下のようになる（費用は 1000 ドル単位）．

目的関数:選択したツアーの費用を最小化する;

制約条件：選んだツアーは正確に 1 つの飛行をカバーしている；

```
MODEL:
[_1]MIN=3*T_1+3*T_2+3*T_3+3*T_4+3*T_5+3*T_6+3*T_7+3*T_8+3*T_9+3*T_10+3*T_11+2*T_12+3*T_13+3*T
_14+2*T_15+3*T_16+3*T_17+3*T_18+3*T_19+2*T_20+3*T_21+3*T_22+3*T_23+2*T_24+2*T_25+3*T_26+3*T_2
7+2*T_28+3*T_29+3*T_30+3*T_31+3*T_32+3*T_33+3*T_34+3*T_35+3*T_36+3*T_37;
[COV_F101]T_1+T_11+T_16+T_25+T_26+T_29+T_34=1;
[COV_F410]T_2+T_12+T_13+T_19+T_27+T_28+T_30+T_33+T_35=1;
```

```
[COV_F220]T_3+T_14+T_15+T_29+T_30=1;
[COV_F17]T_4+T_14+T_16+T_20+T_22+T_25+T_27+T_29+T_31+T_34+T_36=1;
[COV_F7]T_5+T_17+T_18+T_31+T_32=1;
[COV_F13]T_6+T_12+T_17+T_26=1;
[COV_F11]T_7+T_15+T_19+T_21+T_23+T_26+T_28+T_30+T_32+T_33+T_35+T_37=1;
[COV_F19]T_8+T_13+T_18+T_20+T_21+T_27+T_28+T_31+T_32+T_34=1;
[COV_F23]T_9+T_11+T_22+T_23+T_24+T_25+T_35+T_36+T_37=1;
[COV_F3]T_10+T_24+T_36+T_37=1;
@BIN(T_1);@BIN(T_2);@BIN(T_3);@BIN(T_4);@BIN(T_5);@BIN(T_6);@BIN(T_7);@BIN(T_8);@BIN(T_9);@BIN(T_10);@BIN(T_11);@BIN(T_12);@BIN(T_13);@BIN(T_14);@BIN(T_15);@BIN(T_16);@BIN(T_17);@BIN(T_18);@BIN(T_19);@BIN(T_20);@BIN(T_21);@BIN(T_22);@BIN(T_23);@BIN(T_24);@BIN(T_25);@BIN(T_26);@BIN(T_27);@BIN(T_28);@BIN(T_29);@BIN(T_30);@BIN(T_31);@BIN(T_32);@BIN(T_33);@BIN(T_34);@BIN(T_35);@BIN(T_36);@BIN(T_37);
END
```

PICTURE は，問題の構造を明らかにする．例えば，最初の制約（2 行目）は，フライト 101 を含むツアーを選ばなければならないことを示している．

```
                          T T T T T T T T T T T T T T T T T T T T T T T T T T T T T
          T T T T T T T T T 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
          1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7

     1:   3 3 3 3 3 3 3 3 3 3 2 3 3 2 3 3 3 3 2 3 3 3 2 2 3 3 2 3 3 3 3 3 3 3 3 MIN
     2:   1                   1         1                   1 1     1         1       = 1
     3:     1                   1 1           1                 1 1   1     1   1     = 1
     4:       1                     1 1                             1 1             = 1
     5:         1                       1       1   1     1   1   1   1     1   1   = 1
     6:           1                       1 1                         1 1           = 1
     7:             1           1         1                                 1         = 1
     8:               1               1       1   1   1     1   1   1   1 1   1   1 = 1
     9:                 1         1         1   1 1           1 1     1 1   1       = 1
    10:                   1   1                     1 1 1 1 1                 1 1 1 = 1
    11:                     1                         1                       1 1 = 1
```

LP として解いたとき，解は都合の良いことに整数となり，次のものが選ばれた．

ツアー	フライト
T17	7,　13
T24	3,　23
T28	410,　19,　11
T29	220,　17,　101

この解の費用は 10，000 ドルである．T_17 + T_24 + T_28 + T_29 <= 3 という制約を加えると，同じ費用で，他の最適解がある．

ツアー	フライト
T12	410,　13
T24	3,　23

T29	220, 17, 101
T32	7, 11, 19

500 以上の制約を伴う乗員スケジューリング問題の IP は,解くのが驚くほど難しい. この問題の一般的な定式化は，7．4 で述べる．

7.3.3 付加的な事項

実用上，上の定式化に加える点は，乗員の生活場所である．ホームベースは，各旅行と関係している．各ホームベースで生活するパイロット数が分かれば，そのホームベースからの旅行数に上限を付けた制約を加えたいかもしれない.

また実際問題として,パイロットは概して 1 種類の航空機だけに適任である.そこで,機種毎（例えばボーイング 747，エアバス 320，その他）に，乗員配置問題を解くことである．同じように，客室乗務員はパイロットと独立してスケジュールされる．

客室乗務員が選ばれたあと,客室乗務員をパイロットに割り当てる面倒な問題がある.組合との合意によりアメリカでは，主要な航空会社の多くは，パイロットの希望で予定を選ぶのを許す．より小さい航空会社や米国以外の会社では，全社的な立場でパイロットの割り当てを決めているかもしれない.このような中央管理システムは能率の悪さを除去するが，少なくとも高い年功をもつパイロットの希望で決めることは，彼らにより幸せ感を与えるかもしれない.

悪天候と器材故障による不確実性は，定期航空路線にとって残念な事実である．そこで，混乱の影響を受けない乗員予定計画ができればありがたい．予定計画がしばしば飛行機を乗り換えることを乗員に要求し，これらの接続時間が短いならば，この計画は敏感で混乱の影響を受ける傾向がある．飛行機が半時間ほど若干の器材の修繕で遅延し,この飛行機の乗組員が次の空港で飛行機を乗り変える予定であるならば,この空港で次の最高 3 台の便の遅れが発生する：①遅れた飛行機の次の予定，② 遅れた飛行機の乗員を使う予定の次の飛行，そして③この飛行機の乗客が乗り換える飛行機の次の予定.

この小さい例で，代替案があることを知っている．例えば，総乗員費用を最小にすることに加えて,飛行機を乗り換える乗員を含むツアーを避けることで 2 次的な目的関数として遅れの期待値を最小化できるかもしれない．このために．次の方策がある：①短い乗換えを避ける．②多くの乗客が乗り換える便を避ける．③遅れた便の代わりをする乗員を空港に待機させる.

7.4 一般的なカバーリング/分割/梱包モデル

乗員スケジューリング問題のモデルは，非常に特有なものでした．以下は，標準的ないわゆる被覆，分割または梱包の問題で，一般的なものです．このモデルは施設と顧客の視点を取ります．各施設またはパターンが開かれた場合，指定された顧客または要求のセットを提供する．乗員スケジューリングの例に特化すると，ツアーは施設であり，フライトは顧客です．入力される主なデータは，どの施設がどの顧客にサービスを提供しているかを記述する 2 組に関した情報です．このモデルは，顧客が提供されているか

どうかを指定する際にかなりの柔軟性を提供します．パラメータ BUDU が 0 に設定されている場合，これはすべての顧客（またはフライト）が少なくとも 1 つのオープン施設（またはツアー）によって提供されなければならないことを意味します．これは，集合被覆問題とも呼ばれます．また，BUDV が 0 に設定されている場合，各フライトは最大 1 つの選択されたツアーで表示されます．これは，時々集合包装問題と呼ばれます． BUDV と BUDU の両方がゼロに設定されている場合は，各フライトは選択した 1 つのツアーに必ず表示されなければなりません．これは集合分割問題と呼ばれることもあります．実行可能な解決策は，実際には顧客セットをサブセットに分割するため，選択された各オープン施設（またはツアー）の 1 つのサブセットであるため，集合分割と呼ばれます．

```
! 集合被覆/分割/梱包問題 (COVERPAK);
SETS:
! 1組の要求と1組の候補パターンが与えられ，どの要求が各パターンでカバーされているか，
あるいはどのパターンを使用すべきか？;
PATTERN: COST, Y; ! パターンあるいはツアー;
DMND: CU, CV, U, V;
! The "which PATTERN serves which demand" 2-tuples;
PXD( PATTERN, DMND);
ENDSETS
DATA:
! Data for a simple crew scheduling problem;
PATTERN = 1.37;
!   Cost of each PATTERN;
COST = 3 3 3 3 3 3 3 3 3 3 3 2 3 3 2 3 3 3 3 2
       3 3 3 2 2 3 3 2 3 3 3 3 3 3 3 3 3;
! 需要名;
DMND= F101 F410 F220 F17 F7 F13 F11 F19 F23 F3;
! 各需要の下での費用/単位;
CU =     1;
! Cost/unit over at each demand;
CV =     1;
! パターンや施設設備に費やすことができる最大;
BUDGET = 9999;
! 各需要で許容される最大の未成年者数,
0がカバーまたはパーティション化の問題を引き起こす;
BUDU = 0;
! 各需要の最大許容オーバーヘッド，0はそれをパッキング
```

またはパーティショニングの問題にします;

BUDV = 0;

! Both = 0にするとパーティション化の問題になります;

PXD =

1, F101 2,F410 3,F220 4,F17 5,F7 6,F13 7,F11 8,F19 9,F23 10,F3

11, F101 11,F23 12,F410 12,F13 13,F410 13,F19 14,F220 14,F17

15, F220 15,F11 16,F17 16,F101 17,F7 17,F13 18,F7 18,F19

19, F11 19,F410 20,F19 20,F17 21,F19 21,F11 22,F23 22,F17

23, F23 23,F11 24,F3 24,F23 25,F101 25,F23 25,F17

26, F101 26,F13 26,F11 27,F410 27,F19 27,F17

28, F410 28,F19 28,F11 29,F220 29,F17 29,F101 30,F220 30,F11 30,F410

31, F7 31,F19 31,F17 32,F7 32,F19 32,F11 33,F11 33,F410

34, F19 34,F17 34,F101 35,F23 35,F11 35,F410 36,F3 36,F23 36,F17

37, F3 37,F23 37,F11;

ENDDATA

!----------;

! Minimize cost of facilities opened, demands under or over served, ;

MIN = @SUM(PATTERN(I): COST(I) * Y(I))

+ @SUM(DMND(J): CU(J) * U(J) + CV(J) * V(J));

! 各需要に対し, **(**そのサービスパターン総数＋下回る変数―上回る変数＝**1;**

@FOR(DMND(J):

[COV] @SUM(PXD(I, J): Y(I)) + U(J) - V(J) = 1;);

! 予算内に留まる設備費**;**

@SUM(PATTERN: COST * Y) <= BUDGET;

! 必要以上のコストと需要**;**

@SUM(DMND: CU * U) <= BUDU;

@SUM(DMND: CV * V) <= BUDV;

! パターンは開いているか，開いていないか**;**

@FOR(PATTERN(I): @BIN(Y(I)));

Row	Slack or Surplus	Dual Price
1	10.00000	-1.000000
COV(F101)	0.000000	-1.000000
COV(F410)	0.000000	-1.000000
COV(F220)	0.000000	-1.000000
COV(F17)	0.000000	-1.000000
COV(F7)	0.000000	-1.000000

COV(F13)	**0.000000**	**-1.000000**
COV(F11)	**0.000000**	**-1.000000**
COV(F19)	**0.000000**	**-1.000000**
COV(F23)	**0.000000**	**-1.000000**
COV(F3)	**0.000000**	**-1.000000**
12	**9989.000**	**0.000000**
13	**0.000000**	**0.000000**
14	**0.000000**	**0.000000**

訳注:Generateで次のモデルが得られる.

MODEL:

[_1] MIN= U_F101 + V_F101 + U_F410 + V_F410 + U_F220 + V_F220 + U_F17 + V_F17 + U_F7 + V_F7 + U_F13 + V_F13 + U_F11 + V_F11 + U_F19 + V_F19 + U_F23 + V_F23 + U_F3 + V_F3 + 3 * Y_1 + 3 * Y_2 + 3 * Y_3 + 3 * Y_4 + 3 * Y_5 + 3 * Y_6 + 3 * Y_7 + 3 *Y_8 + 3 * Y_9 + 3 * Y_10 + * Y_11 + 2 * Y_12 + 3 * Y_13 + 3 * Y_14 + 2 * Y_15 + 3 * Y_16 + 3 * Y_17 + 3 * Y_18 + 3 * Y_19 + 2 * Y_20 + 3 * Y_21 + 3 * Y_22 + 3 * Y_23 + 2 * Y_24 + 2 * Y_25 + 3 * Y_26 + 3 * Y_27 + 2 * Y_28 + 3 * Y_29 + 3 * Y_30 + 3 * Y_31 + 3 * Y_32 + 3 * Y_33 + 3 * Y_34 + 3 * Y_35 + 3 * Y_36 + 3 *Y_37;

[COV_F101] U_F101 - V_F101 + Y_1 + Y_11 + Y_16 + Y_25 + Y_26 + Y_29 + Y_34 = 1;

[COV_F410] U_F410 - V_F410 + Y_2 + Y_12 + Y_13 + Y_19 + Y_27 + Y_28 + Y_30 + Y_33 + Y_35 = 1;

[COV_F220] U_F220 - V_F220 + Y_3 + Y_14 + Y_15 + Y_29 + Y_30 = 1;

[COV_F17] U_F17 - V_F17 + Y_4 + Y_14 + Y_16 + Y_20 + Y_22 + Y_25 + Y_27 + Y_29 + Y_31 + Y_34 + Y_36 = 1;

[COV_F7] U_F7 - V_F7 + Y_5 + Y_17 + Y_18 + Y_31 + Y_32 = 1;

[COV_F13] U_F13 - V_F13 + Y_6 + Y_12 + Y_17 + Y_26 = 1;

[COV_F11] U_F11 - V_F11 + Y_7 + Y_15 + Y_19 + Y_21 + Y_23 + Y_26 + Y_28 + Y_30 + Y_32 + Y_33 + Y_35 + Y_37 = 1;

[COV_F19] U_F19 - V_F19 + Y_8 + Y_13 + Y_18 + Y_20 + Y_21 + Y_27 + Y_28 + Y_31 + Y_32 + Y_34 = 1;

[COV_F23] U_F23 - V_F23 + Y_9 + Y_11 + Y_22 + Y_23 + Y_24 + Y_25 + Y_35 + Y_36 + Y_37 = 1;

[COV_F3] U_F3 - V_F3 + Y_10 + Y_24 + Y_36 + Y_37 = 1;

[_12] 3 * Y_1 + 3 * Y_2 + 3 * Y_3 + 3 * Y_4 + 3 * Y_5 + 3 * Y_6 + 3 * Y_7 + 3 * Y_8 + 3 * Y_9 + 3 * Y_10 + 3 * Y_11 + 2 * Y_12 + 3 * Y_13 + 3 * Y_14 + 2 * Y_15 + 3 * Y_16 + 3 * Y_17 + 3 * Y_18 + 3 * Y_19 + 2 * Y_20 + 3 * Y_21 + 3 * Y_22 + 3 * Y_23 + 2 * Y_24 + 2 * Y_25 + 3 * Y_26 + 3 * Y_27 + 2 * Y_28 + 3 * Y_29 + 3 * Y_30 + 3 * Y_31 + 3 * Y_32 + 3 * Y_33 + 3 * Y_34 + 3 * Y_35 + 3 * Y_36

```
+ 3 * Y_37 <= 9999;
[_13] U_F101 + U_F410 + U_F220 + U_F17 + U_F7 + U_F13 + U_F11 + U_F19 + U_F23 + U_F3 <=
0;
[_14] V_F101 + V_F410 + V_F220 + V_F17 + V_F7 + V_F13 + V_F11 + V_F19 + V_F23 + V_F3 <=
0;
@BIN( Y_1); @BIN( Y_2); @BIN( Y_3); @BIN( Y_4); @BIN( Y_5); @BIN( Y_6); @BIN( Y_7);
@BIN( Y_8); @BIN( Y_9); @BIN( Y_10); @BIN( Y_11); @BIN( Y_12); @BIN( Y_13);
@BIN( Y_14); @BIN( Y_15); @BIN( Y_16); @BIN( Y_17); @BIN( Y_18); @BIN( Y_19);
@BIN( Y_20); @BIN( Y_21); @BIN( Y_22); @BIN( Y_23); @BIN( Y_24); @BIN( Y_25);
@BIN( Y_26); @BIN( Y_27); @BIN( Y_28); @BIN( Y_29); @BIN( Y_30); @BIN( Y_31);
@BIN( Y_32); @BIN( Y_33); @BIN( Y_34); @BIN( Y_35); @BIN( Y_36); @BIN( Y_37);
 END
```

前の解と違っているが，最適解は同じなので，代替案であることが分かる．

第 8 章 ネットワーク・配送・PERT/CPM

8.1 ネットワーク型モデルの特徴

「**ネットワーク型の LP**」と呼ばれる LP モデルは,特に単純な形であり注目に値する.

② これらの問題は，グラフで簡単に表すことができる.

②通常の条件のもとで整数解をもつ．そしてこのモデルは，種々の輸送戦略を記述し分析するのに役立つ.

③一般的な LP 問題より計算が簡単である.

すぐに思いつく例として，パイプラインや電線のネットワークがある．複数の工場で製品を生産し，それを多くの問屋や客に搬送する企業にとって，ネットワーク型の LP は，種々の輸送戦略を表し，解析するために有用な道具である.

ネットワーク型の LP に注目する 2 つめの理由は，これらの問題を解くための効率的な特殊解法が存在する．これらの解法は，一般の単体法（LP の解放アルゴリズム）と比べて，約 100 倍も速い．これらの解法の中には，単体法が一般の LP 問題に適用されるよりも何年も早く開発されたものもある．Bradley, Brown & Graves(1977)参照

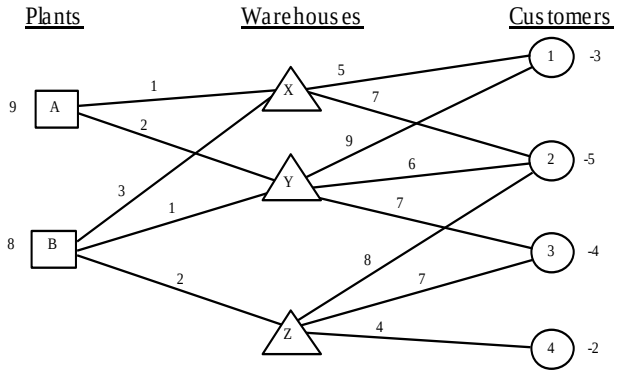


図 8.1 3 段階輸送ネットワーク

図 8.1 は,製品を配送するのに問屋を仲介にしている会社の配送システムを表したネットワークである．この会社は，A，B で表される 2 つの工場と，X，Y，Z で表される 3 つの問屋と,1 から 4 で表される 4 つの販売地域を持っている.各ノードの横の数字は,そのノードにおける製品の利用可能量を表す．例えば工場Ａは，9 単位の製品が搬出可能である．また販売地域 3 では，逆に 4 単位を搬入する必要がある．アーク上の数字は，そのアークを用いて搬出する製品 1 単位当りの費用である．例えば，工場 A の 9 単位のうち，5 単位を問屋 Y に搬送する場合，費用は 5×2＝10 だけかかる．問題は，全ての販売地域の需要を満たし,総費用を最小にするような各アークの搬送量を決定である.

ネットワーク問題であるための LP の主条件は，その問題がネットワークで表現できることである．ネットワーク問題は，①3 段階以上のノードを用いることができる．②

何本のアークでも，またどの２つのノード間でもアークを用いることができる．③アークの流量に制限を与えることができる．

適切に変数を定義することで，この問題を表すLPは，次のようになる．

```
[COST] MIN = AX + 2 * AY + 3 * BX + BY + 2 * BZ + 5 * X1 + 7 * X2 + 9 * Y1 + 6 * Y2 + 7 * Y3 +
8 * Z2 + 7 * Z3 + 4 * Z4;
[A] AX + AY <= 9;
[B] BX + BY + BZ <= 8;
[X] - AX - BX + X1 + X2 = 0;
[Y] - AY - BY + Y1 + Y2 + Y3 = 0;
[Z] - BZ + Z2 + Z3 + Z4 = 0;
[C1] - X1 - Y1 = -3;
[C2] - X2 - Y2 - Z2 = -5;
[C3] - Y3 - Z3 = -4;
[C4] - Z4 = -2;
```

（搬入量）＝（搬出量）となるように各ノードに一つの制約式が存在する．例えば，制約式[Y]は倉庫Ｙに関するもので，搬出量 - 搬入量＝0である．

ネットワーク問題の構造の理解には，「PICTURE」コマンドを用いればよい．

訳注:最新版は,色付きのモザイクで表示され,かえって分かりにくいことも多いので,この元のテキスト形式も選べるように希望を出しました.

	A X 1	A Y 2	B X 3	B Y 1	B Z 2	X 1 5	X 2 7	Y 1 9	Y 2 6	Y 3 7	Z 2 8	Z 3 7	Z 4 4	
COST:														MIN
A:	1	1	,			,			,			,		= 9
B:	,	,	1	1	1	,	,	,	,	,	,	,	,	= 8
X:	-1		-1			1	1		,			,		=
Y:		-1		-1		,		1	1	1		,		=
Z:	,	,	,	,	-1	,	,	,	,	,	1	1	1	=
C1:			,			-1		-1	,			,		= -3
C2:			,			,	-1		-1		-1	,		= -5
C3:	,	,	,	,	,	,	,	,	,	-1	,	-1	,	= -4
C4:			,			,			,			,	-1	= -2

これをみるとネットワーク問題の制約行列の特徴が現れているのが分かる．各変数の上下限制約を無視すると，各列には非ゼロ要素がちょうど２つずつある．一方は＋1であり，他方は-1である．＋1が現れた時はそのアークがこの行のノードから出ていることを表し，-1 が現れた時はそのアークがこの行のノードに入っていることを表す．こ

の程度の問題であれば，図 8.1 より手計算で最適解を求められるであろう．以下に記したLINGOの解と比べてみるとよい．

```
  Global optimal solution found at iteration: 6
  Objective value: 100.0000
Variable Value        Reduced Cost
AX        3.000000    0.000000
AY        3.000000    0.000000
BX        0.000000    3.000000
BY        6.000000    0.000000
BZ        2.000000    0.000000
X1        3.000000    0.000000
X2        0.000000    0.000000
Y1        0.000000    5.000000
Y2        5.000000    0.000000
Y3        4.000000    0.000000
Z2        0.000000    3.000000
Z3        0.000000    1.000000
Z4        2.000000    0.000000
 Row    Slack or Surplus      Dual Price
COST        100.0000           -1.000000
    A       3.000000            0.000000
    B       0.000000            1.000000
    X       0.000000            1.000000
    Y       0.000000            2.000000
    Z       0.000000            3.000000
   C1       0.000000            6.000000
   C2       0.000000            8.000000
   C3       0.000000            9.000000
   C4       0.000000            7.000000
```

この解には，2つの喜ばしい特徴がある．これは，全てのネットワーク問題の解に共通している．

①右辺の係数（上下限値，供給・需要値）が整数ならば，決定変数も整数となる．

②目的関数の係数が整数ならば，双対価格も整数となる．

ネットワーク型のLPは，次のように要約できる．

① 各ノードには，そのノードで利用される供給と需要量が関連づけられている．負の場合は，需要量である．

② 次の数字は，各アークに関連している．
・アーク 1 単位当りの費用（負の場合もある）
・アークの下限（0 の場合が多い）
・アークの上限（この例では無限）
問題は，全ての供給，需要，フロー制約を満たし，総費用を最小にするフローの決定である．

8.1.1 ネットワーク問題の種類

ネットワーク問題には，4 つの変形した種類がある．すなわち，

①輸送問題または分配問題

2 段階のネットワーク問題で，一方の水準の全てのノードが供給者であり，他方の全てのノードが需要者である．そして，供給者から需要者へのアークだけがある場合，この問題を輸送問題または分配問題という．

②最短，最長経路問題

アメリカ合衆国の道路ネットワークが与えられ，Bangor から SanDiego までの最短路を求めたいとしよう．これは，Bangor が 1 単位供給可能な供給地であり，SanDiego が 1 単位必要としている需要地であるような，輸送問題のネットワークの特殊な場合と等価である．アークにかかる費用は，アークの長さである．この問題を解く，単純で速い解法が存在する．最長経路問題は，PERT/CPM プロジェクトの解析において必要である．

③割当問題

輸送問題で，供給者数と需要者数が等しく，更に全ての供給者は 1 単位供給し，全ての需要者は 1 単位必要とする場合，この問題を割り当て問題という．この問題を解く効率的な特殊解法が存在する．

④最大流量問題

各アーク上の流れに上限制約がある方向性のあるネットワークが与えられたとき，ソースから目的地へ輸送できる最大流量を見つけたい．応用として，軍需物資の問題地点への輸送がある．

8.2 PERT／CPM ネットワークと LP

PERT(Program Evaluation and Review Technique)と CPM (Critical Path Method)は，大きなプロジェクトの進行を管理するための手法であり，互いに密接な関係がある．PERT／CPM の主要部分は，**クリティカル･パス**を計算することである．つまり，計画通りプロジェクトが終了するために，必ず時間通り完了する仕事の集合を求めることである．クリティカル･パスの計算は，非常に単純なネットワーク型の LP 問題である．クリティカル･パスは，遅れているプロジェクトの速度を早める「crashing」に役に立つ．次の表は家を立てるプロジェクトにおける仕事のリストである．

活動	略記号	所要時間	先行活動
基礎地盤の掘削	DIG	3	-

基礎コンクリート打ち込み	FOUND	4	DIG
基礎床コンクリート打ち	POURB	2	FOUND
床梁の据付	JOISTS	3	FOUND
壁の据付	WALLS	5	FOUND
垂木の取り付け	RAFTERS	3	WALLS, POURB
床板張り	FLOOR	4	JOISTS
内装下地	ROUGH	6	FLOOR
屋根張り	ROOF	7	RAFTERS
内装仕上げ	FINISH	5	ROUGH, ROOF
造園	SCAPE	2	POURB, WALLS

図 8.2 に，このプロジェクトの AOA 型のネットワークを示す.

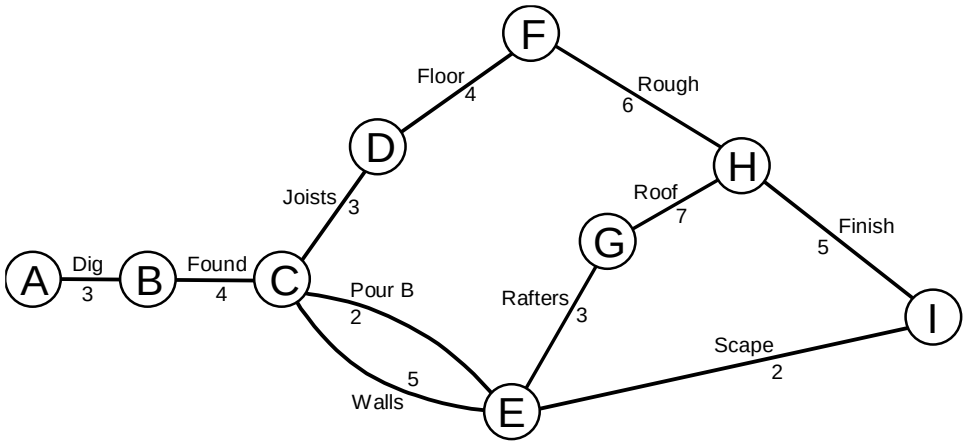


図 8.2 PERT/CPM 型のネットワーク

さて，このプロジェクトを完了するために必要な最短時間を計算したい．この図に関して 1 番興味深いのは図の左から右への最長路である．プロジェクトは，このパス上の仕事にかかる時間の合計より早く完了できない.この場合,クリティカル･パスは DIG, FOUND, WALLS, RAFTERS, ROOF, FINISH という仕事から成り，その長さは 27 である.

この問題は簡単に計算できるが，LP で定式化してみよう．ほとんどの人は 2 つの定式化のうち，最初の 1 つを考えることだろう．最初の定式化は，変数 DIG, FOUND 等はその仕事がクリティカル･パス上にあるかないかにより 1 または 0 の値をとるものとする．目的関数は PERT では最長路を求めることに対応し，次のようになる.

```
MAX = 3 * DIG + 4 * FOUND + 2 * POURB + 3 * JOISTS + 5 * WALLS + 3 * RAFTERS + 4 *
FLOOR + 6 * ROUGH + 7 * ROOF + 5 * FINISH + 2 * SCAPE;
```

この目的関数は一見間違っているように思える.何故ならプロジェクトの長さを最大化したいわけではないからである．しかし，適当な制約式を課すことで，PERT はネットワーク中の最長路を求めることになる．制約式は次の条件を満たすように設定する．

② DIG は必ずクリティカル･パス上にある.

②先行作業の 1 つがクリティカル･パス上にあるときのみ,その仕事がクリティカル･パス上にある可能性がある．また，ある仕事がクリティカル･パス上にあるとき，その後続作業のうちのただ 1 つはクリティカル･パス上にある.

③SCAPE, FINISH のうちどちらかはクリティカル･パス上にある.

次の制約式は上の条件を満足する.

```
- DIG = -1;
- FOUND + DIG = 0;
- JOISTS - POURB - WALLS + FOUND = 0;
- FLOOR + JOISTS = 0;
- RAFTERS - SCAPE + POURB + WALLS = 0;
- ROUGH + FLOOR = 0;
- ROOF + RAFTERS = 0;
- FINISH + ROUGH + ROOF = 0;
+ FINISH + SCAPE = +1;
```

もしネットワーク中のアークの長さをそのアークが表す景色の美しさであると解釈すれば,この定式化は A から I まで行く場合にもっとも景色の良い経路を求めることに相当する．この問題の解は次のようになる.

```
Global optimal solution found at step:         2
Objective value:                     27.00000
Variable           Value        Reduced Cost
     DIG        1.000000            0.0000000
   FOUND        1.000000            0.0000000
   POURB       0.0000000             3.000000
  JOISTS       0.0000000            0.0000000
   WALLS        1.000000            0.0000000
 RAFTERS        1.000000            0.0000000
   FLOOR       0.0000000            0.0000000
   ROUGH       0.0000000             2.000000
    ROOF        1.000000            0.0000000
  FINISH        1.000000            0.0000000
   SCAPE       0.0000000             13.00000
     Row    Slack or Surplus      Dual Price
```

1	27.00000	1.000000
2	0.0000000	6.000000
3	0.0000000	-9.000000
4	0.0000000	-5.000000
5	0.0000000	-2.000000
6	0.0000000	0.0000000
7	0.0000000	2.000000
8	0.0000000	3.000000
9	0.0000000	10.00000
10	0.0000000	15.00000

クリティカル･パス上の作業に相当する変数の値は 1 であることに注目したい．もし最初の制約式 -DIG = -1 を削除した場合，解はどうなるであろうか．

これは，この問題を表現する次の図を見れば分かる．

制約式の各変数は，多くとも 2 つの係数(1 と - 1)しか持たない．これはネットワーク型の LP の顕著な特徴である．

	DIG	FOUND	PROBS	JOWS	WALS	RAFTS	FLOORS	ROUGH	ROOF	FINISH	SCAPE	
1:	3	4	2	3	5	3	4	6	7	5	2	MAX
2:	-1			,			,			,		= -1
3:	1	-1	,	,	,	,	,	,	,	,	,	=
4:		1	-1	-1	-1		,			,		=
5:				1			-1			,		=
6:	,	,	1	,	1	-1	,	,	,	,	-1	=
7:				,			1	-1		,		=
8:				,		1	,		-1	,		=
9:	,	,	,	,	,	,	,	1	1	-1	,	=
10:				,			,			1	1	= 1

2 つめの定式化（AON 型のネットワーク）の目的関数は，プロジェクトにかかる時間を最小にすることである．ここで覚えておくべきことは，AON 型のネットワークでは各ノードは 1 つのイベントを示しているということである．例えば次のようである．

A：地下室を堀り始める時点

C：土台の完成

I：屋根及び内装の完成

変数A, B, C, ・・・, H, Iをこれらのイベントが起こる時点と定義する．すると目的関数は次のようになる．

```
MIN = I - A;
```

イベントが起こるのが予定より遅れれば,それに続くイベントも少なくともその時間だけ遅れる．そこで各々の作業について1つずつ次の制約を得る．

```
B - A >=  3;      ! DIG;
C - B >=  4;      ! FOUND;
E - C >=  2;
D - C >=  3;
E - C >=  5;
F - D >=  4;
G - E >=  3;
H - F >=  6;
H - G >=  7;
I - H >=  5;
I - E >=  2;
```

この問題の解は次のようになる．

```
   Global optimal solution found at step:         13
   Objective value:                 27.00000

   Variable           Value        Reduced Cost
          I        27.00000           0.0000000
          A       0.0000000           0.0000000
          B        3.000000           0.0000000
          C        7.000000           0.0000000
          E        12.00000           0.0000000
          D        10.00000           0.0000000
          F        14.00000           0.0000000
          G        15.00000           0.0000000
          H        22.00000           0.0000000

        Row    Slack or Surplus      Dual Price
          1        27.00000            1.000000
          2       0.0000000           -1.000000
          3       0.0000000           -1.000000
```

4	3.000000	0.0000000
5	0.0000000	0.0000000
6	0.0000000	-1.000000
7	0.0000000	0.0000000
8	0.0000000	-1.000000
9	2.000000	0.0000000
10	0.0000000	-1.000000
11	0.0000000	-1.000000
12	13.00000	0.0000000

目的関数値がクリティカル･パスの長さに等しいことに注目したい．双対価格が非ゼロの制約に注目することで，間接的にクリティカル･パス上の作業を求めることができる．つまりこれらの制約に相当する作業はクリティカル･パス上にある．この一致は偶然ではない．制約の右辺は作業時間である．もしクリティカル･パス上の作業の時間を増やしたならば，プロジェクトの時間も増やすことになる．これは制約の双対価格が非ゼロだからである．この問題の係数行列は下の図のようになる．

```
    A  B  C  D  E  F  G  H  I
1:-1        ,        ,     1 MIN
2:-1  1     ,        ,       > 3
3: , -1 ,1  ,    ,   ,    ,  > 4
4:      -1  ,  1     ,       > 2
5:      -1  1        ,       > 3
6: ,    -1  ,  1 ,   ,    ,  > 5
7:         -1     1  ,       > 4
8:           , -1     1       > 3
9: ,    ,   ,     -1  ,  1 ,  > 6
10:          ,        -1  1    > 7
11:          ,         , -1  1 > 5
12: ,    ,   , -1 ,   ,    , 1 > 2
```

この定式化の図は，1 番目の定式化の図を 90° 回転した図になっている．この 2 つの定式化は一見なんの関係もないように思えたが，実は密接な関係がある．数学者はこの関係を「**双対の関係**」という．

8.3　ネットワーク・ダイヤグラムの表示方法（AOA　対　AON）

ネットワーク・ダイヤグラムの表示方法には 2 通りある．①アークで活動を表す方法（Activity-on-Arc，　AOA），②ノードで活動を表す方法（Activity -on-node，以下で

は AON と略す ）である．図 8.2 では AOA 表現を用いてきた．2 つの方法の特徴は次の通りである．

〔AON の特徴〕

・各活動はネットワークのノードで表される．
・2 つの活動の間の前後関係は，2 つのノードをアークまたはリンクで結びつけることで表される．
・ダミーの活動やアークが必要でないので，間違いが少ない．

〔AOA の特徴〕

・各活動はネットワーク上のアークでもって表される．
・活動 X が Y に先行しているなら，アーク X の後ろにアーク Y が続く．ノードは活動 X の終了を表す事象である．ダミーの活動を表すため長さゼロのアークが，前後関係を正しく表現するために必要になる場合もある．
・スケジューリングに用いられるガントチャートに似ているので，AON に比べ一般的である．

6 個の活動から構成される AON プロジェクトを図 8．3 に示す．各ノードの右側の数字は,各活動に必要な時間を表している.A または B を出発点とし F を終点とする場合，直感的に，最も長いパスは活動 A，C，E，F で，長さは 29 である．

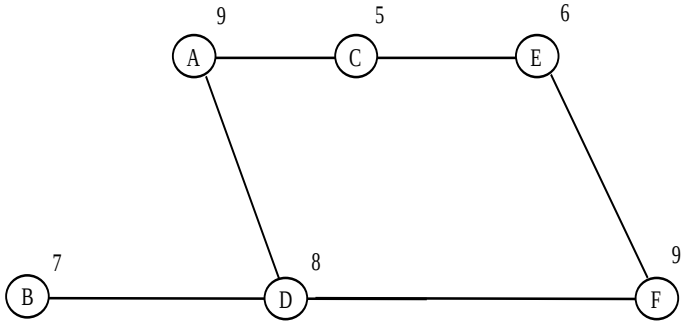


図 8.3 AON 表現

同じプロジェクトを AOA で表すと図 8.4 になる．AOA では，アークの上で円に囲まれた記号の下にある数字が各活動に必要な時間である．数字を 4 角で囲んだものがノードである．ノード 3 と 4 の間にダミーの活動がある．これは，例えば A と B という 2 つの活動が，共通して D を後続の活動として共有し，A だけが C を後続の活動とするような状況で必ず発生する．

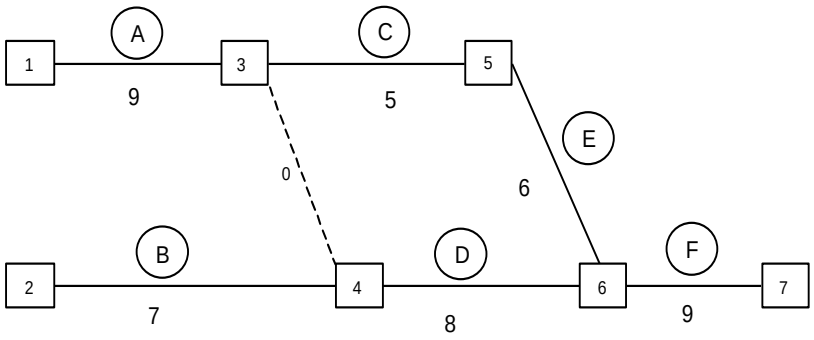


図 8.4 AOA 表現

8.4 プロジェクトのクラッシング（短縮）

クリティカルパスが分かると，次にでる疑問は「このプロジェクトを短縮できるだろうか」ということである．プロジェクトや活動の時間を減少させることを「クラッシング」と呼ぶ．多くのプロジェクトでは， 顧客がお金を十分に支払うことで工期は短縮できる．例えば，ハイウエイ補修プロジェクトは，プロジェクトが目標期日前に完了すれば，1 日に＄5,000 から＄25,000 のインセンティブを払うことは一般的である．

8.4.1 クラッシングのための費用と価値

プロジェクトを短縮することは価値がある．プロジェクトを短縮するには，どの活動を短縮するか否かは，短縮にかかる費用とそれによってもたらされたプロジェクト全体の工期の短縮で得られる経費削減効果との比較で行われる．この決定は，プロジェクトの発注者側と受託者側の間のネゴが必要となり，複雑になることがしばしばある．

8.4.2 活動を短縮化する費用

多くの活動は，残業やシフト制を採用し労働力を投入することで短縮できる．2 交替制を取ると，後者は前者の 1．5 倍の人件費がかかるとしよう．3 交替の場合は，2 倍の費用がかかる．交替制を採用しないと，6 人日（6,000 ドル）で完成される活動を考える．2 交替制をとると 3 日間で完成され，費用は 3×1000＋3×1000×1.5＝7500 ドルと 1.25 倍になる．3 交替制をとると 2 日間で完成され，費用は次のように 1.5 倍になる．

2×1000＋2×1000×1.5＋2×1000×2＝9000 ドル

このようにして，**図 8.5** に示す活動を短縮化するための費用曲線が求まる．

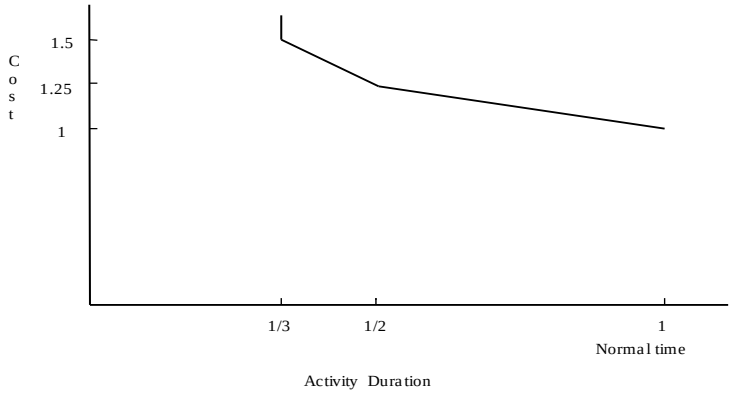


図 8.5 短縮費用曲線

8.4.3 プロジェクトの短縮による価値

どれぐらいプロジェクトを短縮するかを決める 2 通りの方法がある.

② プロジェクトの工期を決め，これを達成するのに必要なだけ短縮する.

② 様々な工期に対し短縮による価値を見積もる.

①の例として，1987 年に Montreal Expos 野球チームの球場建設の例がある．完成目標は，シーズン中に行われる最初の試合の日程に合わせることである.

②の例として，郊外電車の修理を考えよう．早く完成させることで 1 日当たりの価値は幾らであろうか．6,000 人が修理作業の影響を受けると，毎日各人が 10 分の無駄な遅れを被るとすれば，60,000 分（1,000 時間）の遅れが生じる．1 時間当たり 5 ドルの費用とすれば，修理作業を 1 時間短縮する社会的費用は 5,000 ドルになる.

8.4.4 短縮問題の定式化

前の例において各活動の短縮可能性を調査し，次の表の結論を得た.

活動	先行作業	標準作業期間(日)	最大短縮期間(日)	余分な費用/日
A	－	9	5	5000
B	－	7	3	6000
C	A	5	3	4000
D	A, B	8	4	2000
E	C	6	3	3000
F	D, E	9	5	9000

例えば活動 A は，9 日間ではなく 5 日間で完成させることができる．しかし，(9－5)×5,000＝20,000 ドルの余分な費用がかかる．このプロジェクトを，例えば 22 日で完成させなければならないと仮定しよう．どの活動を短縮したらよいだろうか．活動 D は

1日当りの費用が最も安いが，クリティカルパスではないので少し興味をひく程度である．次の変数を定義しよう．

EF_i＝短縮の可能性も考慮して活動ｉが最も早く終了する日数

C_i＝活動ｉを短縮する日数

言葉でLPの定式化を行うと，

最小化（短縮化にかかる費用）

制約条件

各活動ｊと先行活動ｉに対して

ｊの最も早く終了する日数）≧（先行作業ｉの最も早く終了する日数）＋（ｊの実日数）

各活動ｊに対し：（ｊの短縮日数）≦（ｊの実日数）≦（ｊの標準日数）

LINGOで集合による定式化は次のようになる．

```
! 期限付きプロジェクトの最適なクラッシュを見つける;
SETS:
     TASK: NORMAL,  FAST,  COST,  EF,  ACTUAL;
     PRED( TASK,  TASK):;
ENDSETS
DATA:
  TASK,  NORMAL,  FAST,  COST =
   A       9     5     5000
   B       7     3     6000
   C       5     3     4000
   D       8     4     2000
   E       6     3     3000
   F       9     5     9000;
  PRED =
   A, C
   A, D
   B, D
   C, E
   D, F
   E, F;
DUEDATE = 22;
ENDDATA
!-------------------------------------;
! クラッシュ費用を最小限に抑える;
[OBJ]MIN=@SUM(TASK(I):COST(I)*(NORMAL(I)- ACTUAL( I)));
```

```
! 先行作業のないタスクの場合;
 @FOR( TASK( J): EF( J) >= ACTUAL( J));
! 先行作業のあるタスクの場合;
 @FOR( PRED( I,  J):EF( J) >= EF( I) + ACTUAL( J));
! 実働時間の限界;
 @FOR( TASK( I): @BND( FAST(I),  ACTUAL( I),  NORMAL( I)));
! 最後のタスクはプロジェクトの最後とみなす;
  EF( @SIZE( TASK)) <= DUEDATE;
```

解は，次の通りである．

```
    Global optimal solution found at step:            12
     Objective value:                        31000.00

           Variable           Value        Reduced Cost
             EF( A)        7.000000            0.0000000
             EF( B)        7.000000            0.0000000
             EF( C)        10.00000            0.0000000
             EF( D)        13.00000            0.0000000
             EF( E)        13.00000            0.0000000
             EF( F)        22.00000            0.0000000
         ACTUAL( A)        7.000000            0.0000000
         ACTUAL( B)        7.000000           -4000.000
         ACTUAL( C)        3.000000            1000.000
         ACTUAL( D)        6.000000            0.0000000
         ACTUAL( E)        3.000000            2000.000
         ACTUAL( F)        9.000000           -2000.000
```

追加費用は 31,000 ドルで，22 日のデッドラインを達成できた．

訳注:Generat すると次のモデルになる．目的関数は，各作業の実働日数に余分な費用を掛けてマイナスにし，当初の見積もり費用の 222，000 から引いている．当初の見積もり費用はどこにも説明がない？これは，元の目的関数が「COST(I)*(NORMAL(I)-ACTUAL(I))」であり，COST(I)*NORMAL(I)の和の式に決まっているデータを代入し計算できる．「NORMAL(I)- ACTUAL(I)」で，クラッシングで短縮できる日数を計算し，それに余分な費用を掛けて，プロジェクト全体にかかる余分な費用を求め，最小化している．制約式 2 から 7 は，実際の実働時間が「短縮の可能性も考慮して活動 i が最も早く終了する日数」を超えないことを制約している．制約式 8 から 13 は，制約式 8「- EF_A + EF_C - ACTUAL_C >= 0;」を修正して，「EF_C >= EF_A + ACTUAL_C;」に変形すると分かるように，C にとって唯一の先行作業の A がさも早く終了する日数に実働日数を足したものが，「短縮の可能性

も考慮して活動 C が最も早く終了する日数」を超えないことを課している．制約式 22 は，最後の作業 F の 最も早く終了する日数を 22 以下に制限している．このモデルを見れば，単純なようだが，先行と後続の作業関係で間違いなくこの分かりやすいモデルを 1 度で作ることは難しいであろう．上の集合モデルを理解すれば，どんな大きな問題にも対応できる汎用のモデルを得たことになる．つまり，訳者も原著を翻訳していて翻訳に確信があったわけではない．しかし，雛型モデルで間違いなくここで説明できたが，読者も同じ勉強法を取り，時間をかけないことが，知的生産性の向上に役立つ．8 個の @BND で，実際の日数の下限がクラッシングで短縮できる日数とし，上限を現在の見積もり日数である．

```
MODEL:
[OBJ] MIN= - 5000 * ACTUAL_A - 6000 * ACTUAL_B - 4000 * ACTUAL_C - 2000 *ACTUAL_D
- 3000 * ACTUAL_E - 9000 * ACTUAL_F + 222000;
 [_2] EF_A - ACTUAL_A >= 0;
 [_3] EF_B - ACTUAL_B >= 0;
 [_4] EF_C - ACTUAL_C >= 0;
 [_5] EF_D - ACTUAL_D >= 0;
 [_6] EF_E - ACTUAL_E >= 0;
 [_7] EF_F - ACTUAL_F >= 0;
 [_8] - EF_A + EF_C - ACTUAL_C >= 0;
 [_9] - EF_A + EF_D - ACTUAL_D >= 0;
 [_10] - EF_B + EF_D - ACTUAL_D >= 0;
 [_11] - EF_C + EF_E - ACTUAL_E >= 0;
 [_12] - EF_D + EF_F - ACTUAL_F >= 0;
 [_13] - EF_E + EF_F - ACTUAL_F >= 0;
 [_14] EF_F <= 22;
 @BND( 5,  ACTUAL_A,  9); @BND( 3,  ACTUAL_B,  7); @BND( 3,  ACTUAL_C,  5);
 @BND( 4,  ACTUAL_D,  8); @BND( 3,  ACTUAL_E,  6); @BND( 5,  ACTUAL_F,  9);
 END
```

ここで，プロジェクトの工期は厳しくないが，指定した日程よりも早く完成すると 1 日当り 5,000 ドルの割増金が出るものとする．PCRASH は指定の 29 日より早く完成した日数を表す．

```
! プロジェクトの最適なクラッシュを見つけるための
  期限および早期完了のインセンティブ;
SETS:
 TASK: NORMAL,  FAST,  COST,  EF,  ACTUAL;
```

```
 PRED( TASK,  TASK):;
ENDSETS
DATA:
  TASK,  NORMAL,  FAST,  COST =
   A       9     5    5000
   B       7     3    6000
   C       5     3    4000
   D       8     4    2000
   E       6     3    3000
   F       9     5    9000;
  PRED =
   A, C
   A, D
   B, D
   C, E
   D, F
   E, F;
! Incentive for each day we beat the due date;
 INCENT = 5000;
 DUEDATE = 29;
ENDDATA
!---------------------------------------;
! より早期の完了インセンティブ支払い以下にクラッシュさせる費用を最小限に抑える;
[OBJ] MIN=@SUM(TASK(I):COST(I)*(NORMAL(I)- ACTUAL( I)))- INCENT * PCRASH;
! 先行作業がない場合;
 @FOR( TASK( J): EF( J) >= ACTUAL( J));
!  先行作業がある場合;
 @FOR( PRED( I,  J):
    EF( J) >= EF( I) + ACTUAL( J));
! Bound the actual time;
 @FOR( TASK( I): @BND( FAST(I), ACTUAL( I), NORMAL( I)));
! 最後の作業が, プロジェクトの終了を表す;
  EF( @SIZE( TASK)) + PCRASH = DUEDATE;
```

解は次の通りである.

```
   Global optimal solution found at step: 21
    Objective value: -6000.000
```

Variable	Value	Reduced Cost
PCRASH	5.000000	0.0000000
EF(A)	7.000000	0.0000000
EF(B)	7.000000	0.0000000
EF(C)	12.00000	0.0000000
EF(D)	15.00000	0.0000000
EF(E)	15.00000	0.0000000
EF(F)	24.00000	0.0000000
ACTUAL(A)	7.000000	0.0000000
ACTUAL(B)	7.000000	-6000.000
ACTUAL(C)	5.000000	-1000.000
ACTUAL(D)	8.000000	0.0000000
ACTUAL(E)	3.000000	0.0000000
ACTUAL(F)	9.000000	-4000.000

出力結果から 5 日間の短縮を行い，全体の工期が 24 日であることが分かる．短縮にかかる費用よりも割増金の方が 6,000 ドル多くなった．

訳注:Generate したモデルは，次の通りである．目的関数に下線を引いた- 9000 * ACTUAL_F と制約式 14 の「PCRASH + EF_F = 29;」が追加されていることが分かる．

MODEL:

[OBJ]MIN=-5000*PCRASH-5000*ACTUAL_A-6000*ACTUAL_B-4000*ACTUAL_C-2000*ACTUAL_D-3000*ACTUAL_E-9000*ACTUAL_F+222000;

[_2]EF_A-ACTUAL_A>=0;

[_3]EF_B-ACTUAL_B>=0;

[_4]EF_C-ACTUAL_C>=0;

[_5]EF_D-ACTUAL_D>=0;

[_6]EF_E-ACTUAL_E>=0;

[_7]EF_F-ACTUAL_F>=0;

[_8]-EF_A+EF_C-ACTUAL_C>=0;

[_9]-EF_A+EF_D-ACTUAL_D>=0;

[_10]-EF_B+EF_D-ACTUAL_D>=0;

[_11]-EF_C+EF_E-ACTUAL_E>=0;

[_12]-EF_D+EF_F-ACTUAL_F>=0;

[_13]-EF_E+EF_F-ACTUAL_F>=0;

[_14]PCRASH+EF_F=29;

@BND(5, ACTUAL_A, 9);@BND(3, ACTUAL_B, 7);@BND(3, ACTUAL_C, 5);

@BND(4, ACTUAL_D, 8);@BND(3, ACTUAL_E, 6);@BND(5, ACTUAL_F, 9);
END

8.5 プロジェクトのリソース制約

多くのプロジェクトで主要な問題は，資源が限られていることだ．限られた資源制約がある場合は，個別の作業を同時に実行してしまうかもしれない．Pritzker, Watters & Wolfe (1969)は，資源制約を表す定式化とジョブショップ・スケジュール問題の定式化を示した．定式化は，次のようなアイデアに基づいている：①時間は連続でなく日のような離散で扱う，②各活動と各時間に対して，活動がその時間に開始する場合にのみ1になる0/1変数が1つある，③全ての期間と資源に対して，その期間に要求される資源の合計が利用可能量を上回らない制約がある．

```
MODEL:
! リソースの制約を伴うPERT/CPMプロジェクトのスケジューリング;
! リソース/マシンの数には限りがあります.
! アクティビティーは，以下の条件を完了しないとに開始できない.
1) すべての前任者が完了した.
2) 必要な資源/機械が利用可能である．;
SETS:
 ! 一定の期間の作業と
   決定される開始時間の集合があります．;
   TASK: TIME, START, ES;
 ! 優先順位の関係は，先行関係を完了市内で,
     第2の作業を開始できない;
     PRED( TASK. TASK);
  ! 期間の集合;
     PERIOD;
     RESOURCE: CAP;
  ! 一部の業務は，一部の部門の容量が必要です;
     TXR( TASK. RESOURCE): NEED;
  ! 作業Iが期間Tで開始する場合，SX（I，T）= 1 ;
     TXP( TASK. PERIOD): SX;
     RXP( RESOURCE, PERIOD);
    ENDSETS
訳注終了

  DATA:
```

```
! プロジェクトの完了に必要な期間の上限;
   PERIOD = 1.20;
 ! タスク名と期間;
   TASK  TIME =
   FIRST    0
   FCAST    7
   SURVEY   2
   PRICE    1
   SCHED    3
   COSTOUT  2
   FINAL    4;
 ! 先行/後行の組み合わせ;
 PRED=  FIRST, FCAST, FIRST, SURVEY,FCAST,PRICE, FCAST, SCHED, SURVEY, PRICE,
        SCHED, COSTOUT,PRICE,FINAL,COSTOUT, FINAL;
 ! 会計と業務の 2 つの部門がある;
   RESOURCE = ACDEPT, OPNDEPT;
        CAP =   1, 1;
 ! How much each task needs of each resource;
     TXR, NEED =
     FCAST, OPNDEPT, 1
     SURVEY,OPNDEPT,1
     SCHED,OPNDEPT,1
     PRICE,ACDEPT,1
     COSTOUT,ACDEPT,1;
 ENDDATA
 !-----------------------------------------------------;
 ! Minimize start time of last task;
 MIN = START( @SIZE( TASK));
 ! 各タスクの開始時間;
 @FOR( TASK( I):[DEFSTRT] START(I) = @SUM( PERIOD(T): T * SX(I,T)););
 @FOR( TASK( I):
 !  各タスクはある期間内に開始する必要があります;
   [MUSTDO]  @SUM( PERIOD( T): SX( I, T)) = 1;
 !変数 SX はバイナリ,すなわち 0 または 1 である;
   @FOR( PERIOD( T): @BIN( SX( I, T));););
 ! 優先順位制約;
```

```
  @FOR( PRED( I, J):
    [PRECD]  START( J) >= START( I) + TIME( I););
 ! Resource usage, For each resource R and period T;
  @FOR( RXP( R, T):
!Sum over all tasks I that use resource R in period T;
    [RSRUSE] @SUM( TXR( I, R):
   @SUM(PERIOD(S)| S #GE# (T-(TIME(I)-1)) #AND# S
   #LE# T: NEED( I, R) * SX( I, S))) <= CAP( R););
 ! The following makes the formulation tighter;
 ! Compute earliest start disregarding resource constraints;
 @FOR( TASK( J):
  ES(J) = @SMAX(0,@MAX( PRED(I,J):ES(I) + TIME(I)));
 ! Task cannot start earlier than unconstrained early
  start;
 @SUM( PERIOD(T) | T #LE# ES(J): SX(J,T)) = 0; );
 END
```

これを解くとプロジェクトの長さは 14 になった.資源制約がなければ,13 になる.:

```
Global optimal solution found
 Objective value:      14.00000
       Variable          Value
  START( FIRST)        1.000000
  START( FCAST)        1.000000
 START( SURVEY)        11.00000
  START( PRICE)        13.00000
  START( SCHED)        8.000000
START( COSTOUT)        11.00000
  START( FINAL)        14.00000
```

8.6 パスによる定式化

多くのネットワーク問題では,ネットワーク全体でとりうるパスで解を考えることが自然である. 例えば, 図 8.1 では次の 13 個のパスがある.

A → X → 1, A → X → 2, A → Y → 1, A → Y → 2, A → Y → 3, B→ X → 1, B→ X → 2, B → Y → 1, B → Y → 2, B → Y → 3, B → Z → 2, B → Z → 3, B → Z → 4

個々のアークよりも, 完全なパスを決定変数と考える利点は次の通りである.

① 複雑な費用構造をもつモデルを扱える (Geroffrion & Graves (1974)参照) .

② 制限のあるパスを熟考できる．例えば，トラック運転手に 1 日 10 時間までしか労働させられない場合，10 時間を超えるパスを考える必要がなくなる．例えば，サプライチェーンでは，長いパスはリードタイムが長くなり認められない．

③ モデルの制約数は明らかに少なくなる．

④ネットワーク構造をもつ幾つかの問題は IP になるが，パスによる定式化を行うと解法が容易になる．

8.6.1 例題

最初の問題を考えてみよう(**図 8.1**)． A から X への輸送の後，X から 2 への輸送だけを同じ業者に頼んだとしよう．この場合，業者は A から X と X から 2 の費用から輸送費用を$1 /単位割り引くものとする．さらに，製品は壊れやすく長距離輸送に適していないので，B → X と X → 2，あるいは A → と Y → 1 のような輸送は避けたい．

A から X を経由して 1 にいたる輸送を AX1 として表し,PAX1 を輸送数量とすると．定式化は次のようになる．

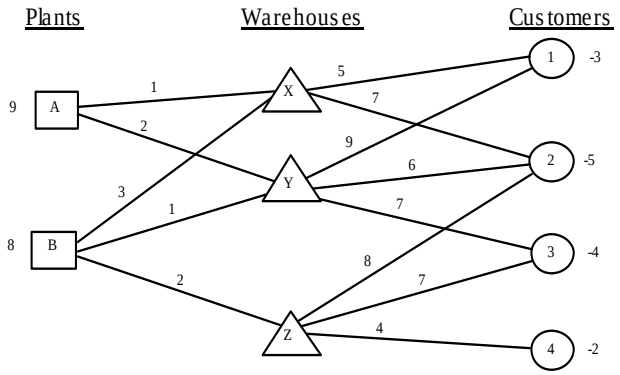


図 8．１ 3 段階輸送ネットワーク(再度掲載)

```
MIN = 6*PAX1 + 7*PAX2 + 8*PAY2 + 9*PAY3 + 8*PBX1 + 10*PBY1 + 7*PBY2 + 8*PBY3 + 10*PBZ2
+ 9*PBZ3 + 6*PBZ4;
[A] PAX1 + PAX2 + PAY2 + PAY3 <= 9;
[B] PBX1+PBY1+PBY2+PBY3 + PBZ2 + PBZ3 + PBZ4 <= 8;
[C1] PAX1 + PBX1 + PBY1 = 3;
[C2] PAX2 + PAY2 + PBY2 + PBZ2 = 5;
[C3] PAY3 + PBY3 + PBZ3 = 4;
[C4] PBZ4 = 2;
```

AX2 の費用は AX2 = 1 + 7 − 1 = 7 になる． またパス BX2 と AY1 は現れない．このモデルは，元のモデルが 9 個の制約式があるのに対して，わずか 6 個になる．このような制約式の減少は，中間ノードの介在が不要になったためである．一般に，パスの定式化で制約式は少なくなるが，決定変数が増える．このモデルを解くと，次の解を得る(非ゼロのみ表示)．

```
Objective value=  97.0000
   Variable           Value
       PAX1        3.000000
       PAX2        3.000000
       PBY2        2.000000
       PBY3        4.000000
       PBZ4        2.000000
```

これは，パス AX2 の費用が $1 安くなるので，前の解よりも安くなる．パスの定式化がネットワーク型の LP に等しいなら，自然な整数解を持つ．パスの定式化は，**長期間の森林管理**によく用いられる．Davis and Johnson (1986)では，「Model I アプローチという方法が紹介されている．標準のネットワーク型の LP は，「Model II アプローチと呼ばれている．Model II による森林管理は，ネットワークのリンクは特定の期間における特定の種類の木を 1 エーカー植林し，いつ伐採するかの意思決定を表している．ノードは，特定の伐採と植林の決定を表す．Model I の決定変数は，特定の土地を管理（伐採と植林)する方法を完全に規定している．Model I のモデルは，数百の制約式で表される半面，100 万以上の決定変数やパスがあるものが多い．

Fourier/Motzkin/Dines 法として知られているパスの定式化が，LP プログラムで一般化できる (Martin(1999)，Dantzig(1963)参照)．パスの定式化をネットワーク LP へ変形するには，ノードに入力アークおよび出力アークのあらゆる組合せの新変数を作ることで，特定のノード(制約式)を除去することである．任意の LP の制約式は，それが最初に右辺定数が 0 の制約式に変更し，それから新変数が制約式で正と負の係数をもつあらゆる組合せに作り変えれば削減できる．このアプローチの不利な点は，制約式の数が 1 つ減るのにたいして，変数の数は元の制約式の数と比べて指数的に増加することがある．

8.7　方向性のないネットワーク

多くの通信ネットワークでは，アークに容量制限があるが方向性はないものとする．例えば，他の市にいる誰かと電話で話す場合，通話に必要な全てのリンクの容量を使用できるが，リンクの方向性は指定できない．長距離通信会社の最大関心事は，通信ネットワークの管理である．これらは，とくに母の日等の特定の休日で重要になる．通話料の増大ばかりでなく，1 年の他の平日のビジネス利用によるパターンとは劇的な違いを見せる．この会社は次の 2 つの問題に直面している．

①設定問題：各リンクにどれだけの容量を設定すべきか.

②運用問題:与えられた容量の範囲内で,需要に対してどのような経路が最善であるか.

このような方向性のないネットワークのモデル化に対して,パスによる定式化を行ってみよう.

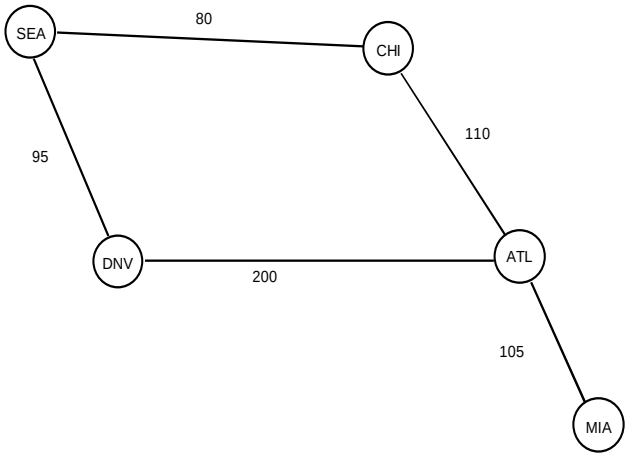


図 8.6 電話会社のネットワーク

図 8.6 に示すネットワークを有する電話会社を考えてみる．各アークにつけられた数字は，そのアーク上を同時に流れる電話の容量である．MIA（マイアミ）の人が SEA（シアトル）に住む母親に電話する場合，電話会社はまず MIA から SEA へのパスの各アークが容量制限内であることを調べなければならない．容量を不経済に使うことは容易である．例えば CHI（シカゴ）と DNV（デンバー）の間に 110 の呼出しがあり，ATL（アトランタ）と SEA の間に 90 の呼出しがあり，これらの呼出しのすべてが ATL と DNV の短いリンクの 200 を用いるとする．次に MIA と SEA の間で呼出しを行いたい．接続は ATL と CHI，そして ATL と DNV の容量を一杯利用しているので不可能である．しかし，CHI と DNV の 110 の呼出しのいくらかを，CHI から SEA を経由して DNV へリンクすれば，MIA と SEA の間で呼出しが可能となる．従来の音声ネットワークでは，呼出しは一度開始されると経路変更できなかった．パケット交換網では，経路変更が可能になった．

8.7.1 例題

ある時点で 2 都市間の電話呼出し需要が，次の表の通りであったとする．さて，どの需要が満たされるだろうか．またどの経路が需要を最大化するのに選ばれるだろうか．

	DNV	CHI	ATL	MIA
SEA	10	20	38	33
DNV		42	48	23
CHI			90	36
ATL				26

もしパスにより定式化すれば，ATL と MIA 間を除いて各都市間のペアには 2 つのパスが存在する．P_{1ij} を都市 i と j の間で最短あるいは北寄りのパスとする．P_{2ij} は，もう一方のパスとする．制約式は次の 2 種類になる．

①各リンクに対する容量の制約

②各都市間の需要に関する制約

以上を定式化すると次のようになる.

```
! Maximize calls carried;
    MAX = P1MIAATL + P1MIADNV + P2MIADNV+ P1MIASEA + P2MIASEA + P1MIACHI+ P2MIACHI +
    P1ATLDNV + P2ATLDNV + P1ATLSEA + P2ATLSEA + P1ATLCHI + P2ATLCHI +
        P1DNVSEA + P2DNVSEA + P1DNVCHI + P2DNVCHI + P1SEACHI + P2SEACHI;
! Capacity constraint for each link;
[KATLMIA]  P1MIAATL + P1MIADNV + P2MIADNV + P1MIASEA + P2MIASEA + P1MIACHI
     + P2MIACHI <= 105;
[KATLDNV]  P1MIADNV + P1MIASEA + P1MIACHI
     + P1ATLDNV + P1ATLSEA + P1ATLCHI
     + P2DNVSEA + P2DNVCHI + P2SEACHI <= 200;
[KDNVSEA]  P2MIADNV + P1MIASEA + P1MIACHI
     + P2ATLDNV + P1ATLSEA + P1ATLCHI
     + P1DNVSEA + P1DNVCHI + P2SEACHI <=  95;
[KSEACHI]  P2MIADNV + P2MIASEA + P1MIACHI
     + P2ATLDNV + P2ATLSEA + P1ATLCHI
     + P2DNVSEA + P1DNVCHI + P1SEACHI <=  80;
[KATLCHI]  P2MIADNV + P2MIASEA + P2MIACHI
     + P2ATLDNV + P2ATLSEA + P2ATLCHI
     + P2DNVSEA + P2DNVCHI + P2SEACHI <= 110;
! Demand constraints for each city pair;
[DMIAATL]             P1MIAATL <= 26;
[DMIADNV]  P1MIADNV + P2MIADNV <= 23;
[DMIASEA]  P1MIASEA + P2MIASEA <= 33;
[DMIACHI]  P1MIACHI + P2MIACHI <= 36;
[DATLDNV]  P1ATLDNV + P2ATLDNV <= 48;
[DATLSEA]  P1ATLSEA + P2ATLSEA <= 38;
[DATLCHI]  P1ATLCHI + P2ATLCHI <= 90;
[DDNVSEA]  P1DNVSEA + P2DNVSEA <= 10;
[DDNVCHI]  P1DNVCHI + P2DNVCHI <= 42;
[DSEACHI]  P1SEACHI + P2SEACHI <= 20;
```

これを解くと 366 の呼び出しの全需要に対して 322 の呼出しを処理できることが分かる.

```
    Optimal solution found at step: 11
    Objective value: 322.0000
```

Variable	Value	Reduced Cost
P1MIAATL	26.00000	0.000000
P1MIADNV	23.00000	0.000000
P2MIADNV	0.00000	2.000000
P1MIASEA	0.00000	0.000000
P2MIASEA	0.00000	0.000000
P1MIACHI	25.00000	0.000000
P2MIACHI	0.00000	0.000000
P1ATLDNV	48.00000	0.000000
P2ATLDNV	0.00000	2.000000
P1ATLSEA	38.00000	0.000000
P2ATLSEA	0.00000	0.000000
P1ATLCHI	23.00000	0.000000
P2ATLCHI	67.00000	0.000000
P1DNVSEA	3.50000	0.000000
P2DNVSEA	6.50000	0.000000
P1DNVCHI	5.50000	0.000000
P2DNVCHI	36.50000	0.000000
P1SEACHI	20.00000	0.000000
P2SEACHI	0.00000	2.000000
Row	Slack or Surplus	Dual Price
1	322.00000	1.000000
KATLMIA	31.00000	0.000000
KATLDNV	0.00000	0.000000
KDNVSEA	0.00000	1.000000
KSEACHI	0.00000	0.000000
KATLCHI	0.00000	1.000000
DMIAATL	0.00000	1.000000
DMIADNV	0.00000	1.000000
DMIASEA	33.00000	0.000000
DMIACHI	11.00000	0.000000
DATLDNV	0.00000	1.000000
DATLSEA	0.00000	0.000000
DATLCHI	0.00000	0.000000
DDNVSEA	0.00000	0.000000
DDNVCHI	0.00000	0.000000

DSEACHI 0.00000 1.000000

実行されない需要は，MIA-CHI 間で 11，MIA-SEA 間で 33 である．明らかに他の最適解があることが分かる．

訳注: この問題は，需要を最大に満たすことを求めているので，表の需要の要求を上限にして≦で解いている．

8.8 複式簿記：会社のネットワークモデル

著者は，しばしば誰がその方法論を最初に使ったか確認するのが好きである．最初のネットワークモデルを明確に述べたのは，Luca Pacioli（L・パチオリ）師である．1594 年にイタリアのフランシスコ会修道院の院長であったとき，彼は複式簿記として知られるようになった方法を発表した．ネットワークの観点から，各々の複式記入はネットワークの弧になる．例えば，あなたが小規模の生地事業を起業したとする．最初の 2 週間に以下の業務が発生する：

CAP	1)事業資金に $50,000 を投資
UR	2) 業者 S から信用で$27,000 の製品を発注
PAY	3)業者 S に$13,000 を支払う
SEL	4) 製品の$5,000 を顧客 C に$8,000 で信用で販売
REC	5) 顧客 C は $2,500 を支払う

我々の表記法では，債務と自己資本は一般的に反対の残高をもっている．例えば，50,000 ドルの流れで，現金の最初の注入は，資産口座（Equity ノード）から現金勘定(Cash)への移動（弧）と一致する．27,000 ドルの流れで，製品の購入は，支払勘定口座ノードから原料在庫口座(Raw material inventory)までの弧と一致する．13,000 ドルの流れは，供給元に 13,000 ドルを払うことは，現金勘定(Cash)から支払勘定口座（Accounts payable）までの弧と一致する．**図 8.7** に実例を示す．

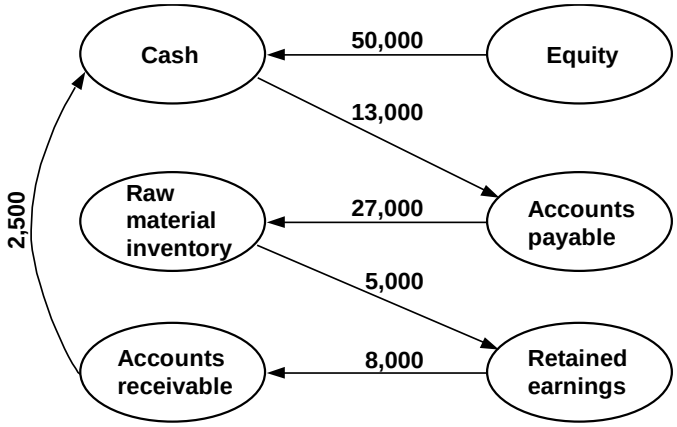


図 8.7 ネットワークモデルとしての複式記入簿記

8.9 ネットワーク型 LP モデルの拡張

実用上重要なネットワークモデルには，幾つかの一般化がある．すなわちこれらの拡張は，真のネットワーク型 LP モデルと同じように 2 つの特徴がある：

・視覚的に表現できる．
・速い解法が，これらの一般化モデルに存在する．

これらの一般化で明らかに分からない特徴は，以下の通りである．

・たとえ入力データが整数であっても，解は一般的にいって整数ではない．

我々が考慮する重要な一般化は，以下の通りである：

① ゲイン(gain)のあるネットワーク：時々一般化されたネットワークと呼ばれているものは，1 つのノードからもう一つのノードまで送られる時，材料の獲得または損失の発生を許す.構造的にこれらの問題は,制約行列の列が高々2 つの非ゼロ要素をもつが，これらの係数が+1 と - 1 という要求は緩められる.これらの問題を解決する特殊な解法（それは単体法より 20 倍速いかもしれません）が存在する．

「ゲインのある輸送」の例として，利子を産む口座への投資，損失の生じる配電，汲み上げ時に天然ガスを燃やす天然ガスパイプラインなどがある．Stroup & Wollmer (1992) は，ゲインモデルによるネットワークが，どこで燃料を購入するべきか，そしてどこからどこに燃料を輸送すべきか決めたい航空機産業に役立つことを示した．Truemper (1976) は，ゲインのあるネットワークが，方向性のないネットワークであれば，適当なスケーリングで純粋なネットワークモデルに変わることを示した．

② 方向性のないネットワーク：通信ネットワークでは，伝送には方向性がない（弧は方向がない）．

④ Multicommodity ネットワーク：多くの配送状況では，複数の必需品がネットワークを流れ競合する．各ソースは必需品を 1 つ生産し，各目的地は 1 つの特定の必需品だけを受けとる．

⑤ Leontief (1951) フロー：いわゆるレオンチェフの投入産出モデルは，各活動は 1 つの必需品だけを生産するが，幾つかの必需品を使う．例えば，1 台の自動車生産は，0.5 トンの鋼材，300 ポンドのプラスチックと 100 ポンドのガラスを使う．材料要求計画(Material Requirements Planning, MRP) モデルは，同じ特徴がある．各出力が 1 つの入力だけを必要とするならば，ゲインのあるネットワークになる．特別な解法が,Leontief フローと MRP モデルを解くために存在する.Jeroslow, Martin, Rardin & Wang (1992)参照.

⑥ 活動/資源図：もし Leontief フローモデルを各活動が幾つかの入力だけでなく出力も持つよう拡張すると，任意の LP モデルになる．これを活動/資源図という．

8.9.1 Multicommodity ネットワーク・フロー

ネットワーク型LPの1つの仮定は，顧客がおそらく価格以外は，その製品がどこからくるかに無関心であるということである．もう一つの仮定は，ネットワークの中を流れているのは，一つの必需品ということである．多くのネットワークでは，複数の異なった必需品がネットワークの中を流れている．各リンクに無限の能力があれば，独立したネットワーク・フローLP として，各必需品別に解くことができる．しかし，全ての必需品の合計に対し有限なリンクであれば， Multicommodity ネットワークになる．

Multicommodity ネットワーク問題の最も分かりやすい例は，船による輸送である．ネットワークが天然ガスパイプラインの場合，必需品はネットワークを流れる異なる燃料であるかもしれない．他の輸送問題（例えば交通割当て，または夜通しのパッケージ搬送）では，各ソースと目的地の一組間に，必需品の受け渡しがある．

重要な特徴は，必需品がネットワークの中で識別されることである．つまり，顧客はどの必需品が届けられるかについて気にする．例は，金属供給会社がトラックを使用して，国中のすべてにアルミニウムバー，ステンレス鋼リング，鋼ビーム，その他を出荷する場合である．このモデルは次のように定式化される．

D_{ik}=ノードiの必需品kの需要量（負の値は供給を意味する）；

C_{ijk} =ノードiからノードjまで必需品kを出荷する費用;

U_{ij} =ノードiからノードjへのリンクの収容力；

我々は，次の値を見つけたい．

X_{ijk} =ノードiからノードjまで出荷される必需品kの量；

$$\min = \Sigma_i \Sigma_j \Sigma_k c_{ijk} x_{ijk}$$

subject to:

各必需品kとノードtに対して:

$$\Sigma_i x_{itk} = D_{tk} + \sum_j x_{tjk}$$

各リンクI，jに対して:

$$\Sigma_k x_{ijk} \le U_{ij}$$

8.9.2 Multicommodity 問題のサイズを減らす

複数の必需品がソースと目的地が同じで，出荷費用が最終的な目的地と独立しているならば，あなたは目的地への必需品をまとめることができる．つまり，ソースと目的地の両方でなく，ソースだけで必需品を特定できる．すなわち，（ソース数）*（目的地数）でなく，ソース数と同じ必需品で考えればよい．例えば，この考察を使用して100-都市間の問題では，1万の必需品でなく100の必需品だけを考えればよい．

存在する Multicommodity ネットワーク問題で最も大きい例の 1 つは，病気である

か負傷した隊員の輸送に関する米国空軍によって開発された Patient Distribution System モデルである.

8.9.3 Multicommodity フロー例

あなたは，フェデラルエクスプレスと競争することを決め，6 都市から小さく始めることにした．下記の行列は，潜在顧客が 1 日に各ソース/目的地の一組の間で動かす必要がある平均数(トン)を表する．例えば，都市 2 の人々は，1 日につき 4 トンを都市 3 の方へ動かす必要がある.

		需要(トン)，D(i, j)ペア	輸送費用/トン，リンク C(i, j)，	容量(トン)，リンク U(i, j)
		1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6
行	1	0 5 9 7 0 4	0 4 5 8 9 9	0 2 3 2 1 20
	2	0 0 4 0 1 0	3 0 3 2 4 6	0 0 2 8 3 9
	3	0 0 0 0 0 0	5 3 0 2 3 5	3 0 0 1 3 9
	4	0 0 0 0 0 0	7 3 3 0 5 6	5 4 6 0 5 9
	5	0 4 0 2 0 8	8 5 3 6 0 3	1 0 2 7 0 9
	6	0 0 0 0 0 0	9 7 4 5 5 0	9 9 9 9 9 0

あなたは，フェデラルエクスプレスのようなハブシステムを使用せず，通常の輸送を考える．ノード i からノード j への輸送費用（トン）を C（i,j）で表す．U（i,j）は 1 日の輸送上限である．この容量規制は，出発地や目的地に関係なく，出荷される全ての商品の総量にあてはまる．C(i,j) と U(i,j) は，ネットワークのリンクにあてはまる．ところが，D（i,j）はソース/目的地のペアを表す．この容量規制は，指定された流れだけにあてはまる．つまり，U（i,j）は U（j,i）に等しい必要はない．i から j へ商品の輸送が，リンク（i,j）を通らないかもしれない．商品はネットワークの中を動くので，商品の識別が重要である．都市 6 はハブのように見えるが，他の全ての都市への高い輸送能力がある.

定式化を小さくするために，3 つの都市（1，2 と 5）だけを考える．我々はネットワークで 3 つのソースに対応した必需品だけを考える.

X_{ijk}=都市 i から都市 j まで必需品 k（トン）を出荷.

定式化は以下の通りである.

```
MODEL:
! キーワード：マルチコモディティ，ネットワーク・フロー，ルーティング;
SETS:
! ネットワーク内のノード;
   NODES/1..6/:;
! 起点となるノードの集合．;
   COMMO(NODES)/1, 2, 5/:;
```

```
    EDGES(NODES,    NODES): D, C, U, V;
    NET(EDGES, COMMO): X;
ENDSETS
DATA:
! 需要量：　原点（行）から目的地まで（col )の出荷量;
D = 0 5 9 7 0 4
     0 0 4 0 1 0
     0 0 0 0 0 0
     0 0 0 0 0 0
     0 4 0 2 0 8
     0 0 0 0 0 0;
! 円弧/リンク上に出荷されるユニットあたりのコスト;
C = 0 4 5 8 9 9
     3 0 3 2 4 6
     5 3 0 2 3 5
     7 3 3 0 5 6
     8 5 3 6 0 3
     9 7 4 5 5 0;
! 各リンクで出荷される金額の上限;
U = 0 2 3 2 1 20
     0 0 2 8 3 9
     3 0 0 1 3 9
     5 4 6 0 5 9
     1 0 2 7 0 9
     9 9 9 9 9 0;
! 円弧/リンクが存在するかどうか;
! V = 0 if U = 0;
! V = 1 otherwise;
V = 0 1 1 1 1 1
     0 0 1 1 1 1
     1 0 0 1 1 1
     1 1 1 0 1 1
     1 0 1 1 0 1
     1 1 1 1 1 0;
ENDDATA
! すべてのリンクで送料を最小限に抑える;
```

```
MIN = @SUM( NET(I, J, K): C(I, J) * X(I, J, K));
! これはバランス制約です。 2 つのケースがあります:バランスをとる必要のあるノードは供給側でなく、
  この場合、（入力量の合計 － 出力の合計）がその都市の需要に等しいかその都市が供給側である。
  （入力量の合計 － 出力の合計）が負であれば、その都市は供給源である;
@FOR(COMMO(K): @FOR(NODES(J)|J #NE# K:
  @SUM(NODES(I): V(I, J) * X(I, J, K) - V(J, I)* X(J, I, K)) = D(K, J); );
  @FOR(NODES(J)|J #EQ# K:
    @SUM(NODES(I): V(I, J)*X(I,J,K)-V(J,I) * X(J,I,K)) = -@SUM( NODES(L): D(K, L))););
! これは容量の制約です;
   @FOR(EDGES(I, J)|I #NE# J: @SUM(COMMO(K): X(I, J, K)) <= U(I, J); );
END
```

3（必需品）*6（都市）= 18 でバランスの取れた制約がある．我々がその代わりにソース/目的地の組合せで商品を特定するならば，9 × 6 = 54 のバランス制約がある．解は次の通りである．

```
   Objective value:                              361.0000
      Variable           Value        Reduced Cost
   X( 1,  2,  1)       2.000000            0.0000000
   X( 1,  3,  1)       3.000000            0.0000000
   X( 1,  4,  1)       2.000000            0.0000000
   X( 1,  5,  1)       1.000000            0.0000000
   X( 1,  6,  1)       17.00000            0.0000000
   X( 2,  3,  2)       2.000000            0.0000000
   X( 2,  4,  2)       2.000000            0.0000000
   X( 2,  5,  2)       1.000000            0.0000000
   X( 3,  4,  5)       1.000000            0.0000000
   X( 4,  2,  5)       4.000000            0.0000000
   X( 4,  3,  2)       2.000000            0.0000000
   X( 5,  3,  1)       1.000000            0.0000000
   X( 5,  3,  5)       1.000000            0.0000000
   X( 5,  4,  5)       5.000000            0.0000000
   X( 5,  6,  5)       8.000000            0.0000000
   X( 6,  2,  1)       3.000000            0.0000000
   X( 6,  3,  1)       5.000000            0.0000000
   X( 6,  4,  1)       5.000000            0.0000000
```

他のリンクの容量制限のため，都市 6 が多くの出荷のために使われている．

8.9.4 フリート・ルーティングと割当て

航空会社やトラック輸送業の重要な問題は,機団や車隊のルーティングと割当てである．決められた一組の輸送または飛行計画があれば，各車両がとる経路と関連した路線パスがある．もし利用できる車両の幾つかの異なる車隊があれば，割り当ては興味深いことである．次の問題は，どの種類の車両が各フライトか輸送に割り当てられるということである． Subramania ら (1994)によってデルタ航空で機団の割り当てに用いられた簡略版を解説する．類似したアプローチが Kontogiorgis & Acharya (1999)によって US 航空で使われた．

次の表は，ユナイテッド航空がかつて典型的な平日に提供していたシカゴ（ORD），デンバー（DEN）とロサンゼルス（LAX）間の飛行スケジュールである．

飛行スケジュール					
		都市		時刻	
	飛行	出発	到着	出発	到着
1	221	ORD	DEN	0800	0934
2	223	ORD	DEN	0900	1039
3	274	LAX	DEN	0800	1116
4	105	ORD	LAX	1100	1314
5	228	DEN	ORD	1100	1423
6	230	DEN	ORD	1200	1521
7	259	ORD	LAX	1400	1609
8	293	DEN	LAX	1400	1510
9	412	LAX	ORD	1400	1959
10	766	LAX	DEN	1600	1912
11	238	DEN	ORD	1800	2121

この予定表は，**図 8.8** のネットワークで表すことができる．図の左上から右下への斜線は，飛行機の到着を意味する．左下から右上への斜めの線は，出発を意味する．図を完成させるために，各飛行機の出発と到着を表す線を加える必要がある．LAX(ロス)からの出発便 274 と DEN(デンバー)への到着をつないでいる細い線は,欠航便を例示する．分かりにくさを避けるために，これらの線は加えていない．予定表が毎日繰り返すならば LAX で例示されるように，ネットワークの各都市に backloop を持たせることは合理的である．ごたごたを避けるために，これらの線は加えていない．

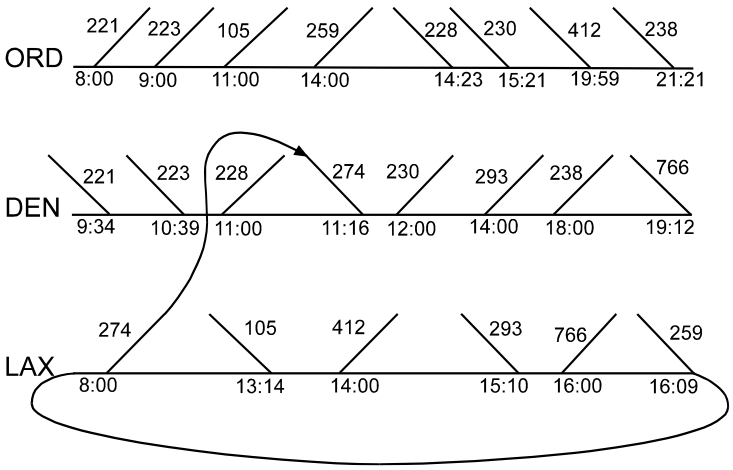


図 8.8 飛行機の運行計画

これをネットワーク問題として解釈する方法は，次の通りである：

①飛行を表す斜めの線（そのパートナーとのつながりで）は変数になる.

②水平線または backloop は，地上で航空機の数を表している決定変数と一致する.

③到着か出発の各々の点はノードになる.

そして，モデルの制約はある時期の各ノードで次の保存則がある：

（この都市にいる航空機の数)+（到達便の数）=

（この都市からの出発便の数）+（この時点以降にいる航空機の数）；

この表記で，22 の制約式（ORD の 8，DEN の 8 と LAX の 6）と 33 の変数（11 の飛行機の変数と 22 の地上の変数）がある．各都市で，解の実現可能性に影響を与えない範囲で次のことを行えば，制約と変数の数は大幅に減らすことができる.

①各到着の時点は，到着後から次の出発時点の前までを表す.

②各出発の時点は，到着の直後から出発までを表す.

このように考えると，必要なノードは，出発便と到着便だけである.

ちょうど 1 種類の航空機の編隊を持っているならば，この予定表で飛ばすために必要な航空機の最小数を知りたい．このモデルを言葉で表すと次になる：

MIN＝夜間地上にいる航空機（決めた予定表でいるべき唯一の場所）；

制約：飛行機のソース＝ネットワークの各ノードでの航空機の使用が各飛行計画をカバーしている；

すべてを考慮すると，以下のネットワーク型 LP になる．変数 G は，指定した瞬間の直後に所定の都市の地上にいる航空機の数を表している.

```
! Fleet routing with a single plane type;
! 地上で一晩過ごす飛行機の数を最小限に抑える;
MIN = GC2400 + GD2400 + GL2400;
! The plane(old) conservation constraints;
! 8 時のシカゴ,  sources - uses = 0;
```

GC2400 - F221 - F223 - F105 - F259 - GC1400 = 0;

! 深夜のシカゴ;

GC1400 + F228 + F230 + F412 + F238 - GC2400 = 0;

! 11 時のデンバー;

GD2400 + F221 + F223 - F228 - GD1100 = 0;

!12 時のデンバー;

GD1100 + F274 - F230 - F293 - F238 - GD1800 = 0;

! 深夜のデンバー;

GD1800 + F766 - GD2400 = 0;

! 8 時の LA;

GL2400 - F274 - GL0800 = 0;

!14 時の LA;

GL0800 + F105 - F412 - GL1400 = 0;

! LA at 1600;

GL1400 + F293 - F766 - GL1600 = 0;

! 深夜の LA;

GL1600 + F259 - GL2400 = 0;

! 飛行制約;

F221 = 1;

F223 = 1;

F274 = 1;

F105 = 1;

F228 = 1;

F230 = 1;

F259 = 1;

F293 = 1;

F412 = 1;

F766 = 1;

F238 = 1;

このモデルは，社員の無賃乗車が行われないと仮定する．つまり，飛行機はその翌日のためにそれを駐機する都市からもう一つの都市まで空で飛ばすことはない．図からおそらく 6 台の航空機が必要であるという単純で直観的な議論ができる．以下は解の詳細である：

```
Optimal solution found at step:         0
Objective value:                 6.000000
Variable           Value        Reduced Cost
```

GC2400	4.000000	0.0000000
GD2400	1.000000	0.0000000
GL2400	1.000000	0.0000000
F221	1.000000	0.0000000
F223	1.000000	0.0000000
F105	1.000000	0.0000000
F259	1.000000	0.0000000
GC1400	0.0000000	1.000000
F228	1.000000	0.0000000
F230	1.000000	0.0000000
F412	1.000000	0.0000000
F238	1.000000	0.0000000
GD1100	2.000000	0.0000000
F274	1.000000	0.0000000
F293	1.000000	0.0000000
GD1800	0.0000000	1.000000
F766	1.000000	0.0000000
GL0800	0.0000000	0.0000000
GL1400	0.0000000	0.0000000
GL1600	0.0000000	1.000000

这样，4 机的航空机是芝加哥，1 机是洛杉矶，1 机是丹佛的地上。

8.9.5 フリートの割当て

航空機が 2 機種以上あるならば，機種を指定する決定変数がいる．現有の機種 A より燃料効率がよい 2 台の機種 B を持つと仮定し，前の例を拡張しよう．しかし，機種 B は機種 A より乗客数は少ない．利益貢献を最大にしたい．機種 i の航空機を**フライト j** に割り当てることからの利益貢献は，以下の通りである：

（フライト j の全ての需要を満たす利益）－（機種 i の限られた収容力のため，フライト j の需要を満たすことができない取りこぼし費用）－（機種 i でフライト j を行う運営経費)+（このフライトで捕えられる前のフライトの取りこぼし需要からの利益）；

取りこぼし経費と回復は，多分，推定するのが最も難しいでしょう．前のモデルは，次の修正で簡単に一般化できる．

① 流れ制約の保存が，機種ごとに必要である．

② 飛行をカバーする 2 つの方法が現在あるので，飛行制約はより柔軟になる．

機種と飛行の各組合せで，慎重に利益貢献を計算した後で以下のモデルを得る．

```
! Fleet routing and assignment with two plane types;
```

```
! 利益を最大化するフライト計画;
MAX = 105 * F221A + 121 * F221B + 109 * F223A + 108 * F223B + 110 * F274A + 115 *
F274B + 130 * F105A + 140 * F105B + 106 * F228A + 122 * F228B + 112 * F230A + 115 *
F230B + 132 * F259A + 129 * F259B + 115 * F293A + 123 * F293B + 133 * F412A + 135 *
F412B + 108 * F766A + 117 * F766B + 116 * F238A + 124 * F238B;
! タイプ A の航空機の流れ制約の保存;
! 8 時の Chicago 空港,  発着 = 出発;
-F221A - F223A - F105A - F259A - GC1400A + GC2400A=0;
! Chicago で夜駐機;
F228A + F230A + F412A + F238A + GC1400A - GC2400A=0;
! 11 時 Denver;
  F221A + F223A - F228A - GD1100A + GD2400A = 0;
! 昼 Denver;
F274A - F230A - F293A - F238A + GD1100A - GD1800A=0;
! 深夜 Denver;
   F766A - GD2400A + GD1800A = 0;
! 8 時 LA;
   - F274A - GL0800A + GL2400A = 0;
! 14 時 LA;
   F105A - F412A + GL0800A - GL1400A = 0;
!16 時 LA;
   F293A - F766A + GL1400A - GL1600A = 0;
! LA で夜駐機;
   F259A - GL2400A + GL1600A = 0;
! Aircraft type B,  conservation of flow;
! Chicago at 8 am;
-F221B - F223B - F105B - F259B - GC1400B +GC2400B=0;
! Chicago で夜駐機;
F228B + F230B + F412B + F238B + GC1400B - GC2400B=0;
! Denver at 11 am;
   F221B + F223B - F228B - GD1100B + GD2400B = 0;
! Denver at high noon;
F274B - F230B - F293B - F238B + GD1100B - GD1800B=0;
! Denver で夜駐機;
   F766B - GD2400B + GD1800B = 0;
! LA at 8 am;
```

```
   - F274B - GL0800B + GL2400B = 0;
! LA at 1400;
   F105B - F412B + GL0800B - GL1400B = 0;
! LA at 1600;
   F293B - F766B + GL1400B - GL1600B = 0;
! LA at midnight;
   F259B - GL2400B + GL1600B = 0;
! Can put at most one plane on each flight;
   F221A + F221B <= 1;
   F223A + F223B <= 1;
   F274A + F274B <= 1;
   F105A + F105B <= 1;
   F228A + F228B <= 1;
   F230A + F230B <= 1;
   F259A + F259B <= 1;
   F293A + F293B <= 1;
   F412A + F412B <= 1;
   F766A + F766B <= 1;
   F238A + F238B <= 1;
! Fleet size of type B;
   GC2400B + GD2400B + GL2400B <= 2;
```

それほど理解が容易でない，次の解が求まる.

```
   Optimal solution found at step:        37
   Objective value:                 1325.000
   Variable           Value        Reduced Cost
      F221B        1.000000           0.0000000
      F223A        1.000000           0.0000000
      F274A        1.000000           0.0000000
      F105A        1.000000           0.0000000
      F228B        1.000000           0.0000000
      F230A        1.000000           0.0000000
      F259A        1.000000           0.0000000
      F293B        1.000000           0.0000000
      F412A        1.000000           0.0000000
      F766B        1.000000           0.0000000
      F238A        1.000000           0.0000000
```

GC2400A	3.000000	0.0000000
GD1100A	1.000000	0.0000000
GL2400A	1.000000	0.0000000
GC2400B	1.000000	0.0000000
GD1100B	1.000000	0.0000000
GD2400B	1.000000	0.0000000

機種 B は，飛行 221，228，293 と 766 をカバーする．2 機種があるので，このモデルは純粋なネットワーク・フローモデルよりむしろ Multicommodity ネットワーク・フローモデルである．当然，このモデルは LP で自然に整数解を見つけることは保証されない．それでも，上の例では整数になった．

上記の方法でモデルを作ることは，退屈である．以下は，集合モデルである．集合で，飛行または機種を加えることは，かなり単純な作業である．

```
MODEL:
SETS:  ！フリートのルーティングと割り当て（FLEETRA）；
    CITY/ ORD DEN LAX/:;  ! The cities involved;
    ACRFT/ A B /: FCOST, FSIZE;  ！航空機の種類;
    FLIGHT/1.11/:;!Each flight gets unique number
    1，2..  ;
    FXCXC(FLIGHT, CITY, CITY)/!Origin-destination
    pairs;
    1 ORD DEN 2 ORD DEN 3 LAX DEN 4 ORD LAX 5 DEN ORD
    6 DEN ORD 7 ORD LAX 8 DEN LAX 9 LAX ORD 10 LAX
    DEN 11 DEN ORD/ : DEPAT,   ARVAT;
    AXC( ACRFT, CITY): OVNITE;
    AXF( ACRFT, FXCXC): X, PC;
ENDSETS
DATA:
    FCOST = 0   0; !Fixed (cost/day)/aircraft of type;
    FSIZE = 7   2;   !Fleet size limits;
    DEPAT =        ! Depart time of each flight;
    800      900      800     1100     1100     1200
   1400     1400     1400     1600     1800;
    ARVAT =        ! Arrival time of each flight;
    934     1039     1116     1314     1423     1521
   1609     1510     1959     1912     2121;
    PC =           ! Profit contribution of each flight;
```

```
    105     109     110      130     106   112!First type A;
    132     115     133      108     116
    121     108     115      140     122   115 !then type B;
    129     123     135      117     124;
ENDDATA
!---------------------- ----------------------;
! フライトからのマイナス寄与をマイナスします
   艦隊における航空機の間接費;
MAX=@SUM(AXF(I,N,J,K):PC(I,N,J,K)*X(I,N,J,K))
    - @SUM( AXC( I, J): FCOST( I) * OVNITE( I, J));
! どんな瞬間でも，特に出発，数
  累積到着数は，> =累積数でなければなりません
  出発.
  各機種のフライトごとに;
 @FOR( ACRFT( I):
  @FOR( FXCXC( N, J, K):
     ! 午前中の地上の航空機+これまでに到着した機種
     >= これから出発する飛行機数;
OVNITE( I, J) +
@SUM( FXCXC( N1, J1, K1)| K1 #EQ# J #AND#
       ARVAT( N1, J1, K1) #LT#    DEPAT( N, J, K):
       X( I, N1, J1, J)) >=
@SUM( FXCXC( N1, J1, K1)| J1 #EQ# J #AND#
       DEPAT( N1, J1, K1) #LE# DEPAT( N, J, K):
       X( I, N1, J, K1));););
  ! このモデルはデッドヘッディングを許さないので,
  到着日の最後の機種は翌日の出発数と同じになる;
 @FOR( ACRFT( I):
    @FOR( CITY( J):
      @SUM( AXF( I, N, J1, J): X( I, N, J1, J)) =
      @SUM( AXF( I, N, J, K): X( I, N, J, K)););
!  各フライトをカバーする必要がある;
 @FOR( FXCXC( N, J, K):
  @SUM( AXF( I, N, J, K): X( I, N, J, K)) = 1;);
! 艦隊のサイズ制限;
 @FOR( ACRFT( I):
```

```
  @SUM( AXC( I, J): OVNITE( I, J)) <= FSIZE( I));
! 実数解は許可されない;
 @FOR( AXF: @GIN( X));
END
```

例えばトラック輸送で，特定の輸送だけをカバーするために賃貸車両を使う選択肢がある．モデルの大きな修正点は，賃貸車両が流れ制約の保存を守る必要がない．他の問題点は，保守に関するものである．航空機では，指定された回数の離着陸の後の保守のために，または飛行時間や特定の経過時間の後で，運転を休止する．そのような詳細を取り入れることは，あまり難しくはないが，モデルはかなり大きくなる．

8.9.6　レオンチェフ・フローモデル

レオンチェフ・フローモデルで，各活動は 1 つの出力を生じる．しかし，それは 0 かそれ以上使うかもしれません．以下に実例を示す．

例：Islandia の投入産出モデル；

Islandia 国には，鋼材，オートバイ，電子機器とプラスチックの 4 つの主な輸出産業がある．経済担当大臣は，（輸出 － 輸入）を最大にしたい．Islandia の通貨はクルツである．鋼材，オートバイ，電子機器とプラスチックの世界市場の価格は 500，1500，300 と 1200 クルツである．鋼材 1 単位の生産は，自動車生産の 0.02 単位，プラスチックの 0.01 単位，世界市場で購入される原料の 250 クルツ，さらに労働力の半年人を必要とする．自動車 1 単位の生産は，鋼材 0.8 単位，電子機器 0.15 単位，プラスチック 0.11 単位，労働力の 1 年人と輸入材料の 300 クルツを必要とする．電子機器の 1 単位の生産は，鋼材 0.01 単位，自動車 0.01 単位，プラスチック 0.05 単位，労働力の半年人と輸入材料の 50 クルツを必要とする．自動車生産は，650,000 単位に制限されている．プラスチックの 1 単位の生産は，自動車生産の 0.03 単位，鋼材の 0.2 単位，電子機器の 0.05 単位，労働の 2 年人さらに輸入材料の 300 クルツを必要とする．プラスチックの上限は，60,000 単位である．Islandia で利用できる全人的資源は 830,000 人/年である．鋼材，オートバイ，電子機器とプラスチックは輸入されません．

どの製品をどれほど生産し，輸出するか？

Islandia 問題の定式化と解

投入産出モデルの定式化は， 2 段階の手続きに従う必要がある．すなわち，①変数を決め，②制約を確認することである．この問題の決定変数を決める鍵は，生産される必需品の量と輸出される量を区別することである．一旦これがされれば，変数は次のように決定できる：

PROD(STEEL)	= 鋼材の生産量
PROD(AUTO)	= 自動車の生産量
PROD(PLASTIC)	= プラスチックの生産量
PROD(ELECT)	= 電子機器の生産量
EXP(STEEL)	= 鋼材の輸出量
EXP(AUTO)	= 自動車の輸出量
EXP(PLASTIC)	= プラスチックの輸出量
EXP(ELECT)	= 電子機器の輸出量

必需品は，鋼，自動車，電子機器，プラスチック，人的資源，自動車生産能力とプラスチック生産能力と，7 つの制約がある.

集合モデルと解は，以下の通りである：

```
MODEL: ! Islandia Input/output model;
SETS:
 COMMO:
   PROD, EXP, REV, COST, MANLAB, CAP;
 CXC(COMMO, COMMO): USERATE;
ENDSETS
DATA:
  COMMO = STEEL,  AUTO,  PLASTIC,  ELECT;
   COST =   250   300    300     50;
   REV  =   500  1500   1200    300;
 MANLAB =   .5    1      2       .5;
      ! 行商品の単位当たりの列商品の使用量;
 USERATE=   -1   .02    .01       0
         .8    -1    .11      .15
         .2   .03     -1      .05
         .01  .01    .05       -1;
  MANPOWER = 830000;
  CAP =   999999 650000  60000 999999;
ENDDATA
[PROFIT] MAX = @SUM( COMMO: REV * EXP - PROD * COST);
 @FOR( COMMO( I):
   [ NETUSE] ! Net use must equal = 0;
     EXP(I) + @SUM(COMMO(J): USERATE(J, I)* PROD(J))
     = 0;
   [CAPLIM] PROD( I) <= CAP( I); );
```

```
[MANLIM] @SUM(COMMO:PROD * MANLAB) < MANPOWER;
END
```

このモデルにはレオンチェフフローの特徴がある．すなわち，各決定変数は，多くても１個の負の制約係数をもつ．解は以下の通りである：

Variable	Value	Reduced Cost
PROD(STEEL)	393958.3	0.0000000
PROD(AUTO)	475833.3	0.0000000
PROD(PLASTIC)	60000.0	0.0000000
PROD(ELECT)	74375.0	0.0000000
EXP(STEEL)	547.9167	0.0000000
EXP(AUTO)	465410.4	0.0000000
EXP(PLASTIC)	0.0	2096.875
EXP(ELECT)	0.0	121.8750
Row	Slack or Surplus	Dual Price
PROFIT	0.4354312E+09	1.000000
NETUSE(STEEL)	0.0000000	500.0000
CAPLIM(STEEL)	606040.7	0.0000000
NETUSE(AUTO)	0.0000000	1500.000
CAPLIM(AUTO)	174166.7	0.0000000
NETUSE(PLASTIC)	0.0000000	3296.875
CAPLIM(PLASTIC)	0.0000000	2082.656
NETUSE(ELECT)	0.0000000	421.8750
CAPLIM(ELECT)	925624.0	0.0000000
MANLIM	0.0000000	374.0625

解は，Islandia の鋼材，自動車，電子機器，プラスチックと人的資源を売る最高の方法が，自動車の生産であることを示す．

この問題は，利益を最大にする代わりに，目標水準が鋼材，自動車，電子機器，プラスチックの輸出（または消費）であれば，レオンチェフの古典的な投入産出モデルになる．問題は，輸出入や消費水準をサポートするのに必要な生産水準を決定することになる．この場合，目的関数は重要でない．

自然な一般化は，種々の必需品を生産する代替技術を見込むことである．これらの種々の技術は，機械化の程度または消費エネルギー（例えば，ガス，石炭，あるいは，水力発電）の形と一致するかもしれない．

8.9.7 活動/資源図

モデルを図で表すアプローチは，任意の LP モデルまで広げることができる．一般的な LP を視覚的に表すために払う費用は，次の要素をネットワークに導入することであ

る．2 つの要素は：①変数に対応し丸で表される活動，②制約に対応し正方形で表される資源．各制約は，各必需品に対応し，「必需品の使用≦必需品の総量」を表す．正方形への矢線は，資源，必需品またはその変数にはインタラクションがある制約と一致する．丸への矢線は，制約がインタラクションがある活動または決定変数と一致しなければなりません．

例：垂直統合した農民

農民は，120 エーカーの農場を持っていて，小麦またはコーンに使用できる．産出高は，年につきエーカー当たり 55 ブッシェルの小麦または 95 ブッシェルのコーンである．労働条件は，小麦 1 ブッシェルにつき 4 時間／エーカー／年とプラス 0.15 時間である．コーン 1 ブッシェルにつき 0.70 時間／エーカーである．種，肥料，その他の原価は，生産される小麦の 1 ブッシェルにつき 20 セントとコーンの 1 ブッシェルにつき 12 セントである．小麦は 0.95 ドル／ブッシェル，コーンは 1.75 ドル／ブッシェルで売ることができた．小麦は 1.50 ドル／ブッシェル，コーンは 2.50 ドル／ブッシェルで買うことができた．

そのうえ，農民はブタや家禽を育てている．彼らが 1 歳に達するとき，農民はブタまたは家禽を売る．ブタは，40 ドルで売れる．家禽はかご単位で測られる．1 つのかごは，40 ドルで販売される．1 匹のブタは，25 ブッシェルの小麦または 20 ブッシェルのコーンを必要とする．さらに，40 時間の労働と 15 平方フィートの床面積を必要とする．家禽の 1 かごは，25 ブッシェルのコーンまたは 10 ブッシェルの小麦，プラス 40 時間の労働と 15 平方フィートの床面積を必要とする．

農民は，10,000 平方フィートの床面積をもつ．彼は，彼の家族と彼自身で各 2,000 時間/年働ける．1.50 ドル/時間で労働者を雇うことができるが，雇われた労働者の各時間の間，農民は 0.15 時間を監督に必要である．どれくらいの土地とコーンと小麦に，そしてどの位のブタや家禽が農民の利益を最大にできるだろうか？この問題は，Hadley（1962）参照．

この問題のために以下の変数を使う：

WH	収穫される小麦(ブッシェル)
CH	収穫されるコーン(ブッシェル)
PH	販売される豚
HS	販売される家禽(かごの数)
LB	労働力(時間)
WS	販売される小麦(ブッシェル)
CS	販売される コーンブッシェル)
CH	家禽の飼育に用いるコーン（ブッシェル)
WH	家禽の飼育に用いる小麦（ブッシェル)
CP	豚の飼育に用いるコーン（ブッシェル)
WP	豚の飼育に用いる小麦（ブッシェル)
CB	購買するコーン(ブッシェル)
WB	購買する小麦（ブッシェル)

この問題の活動-資源図を図 8.9 で示す.

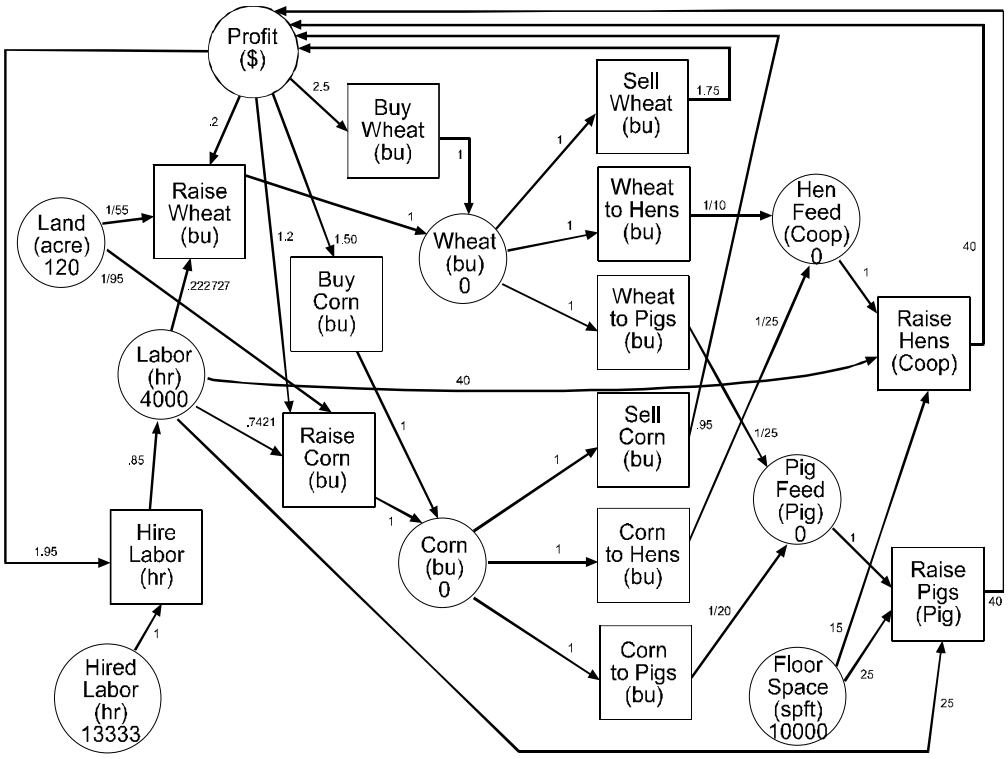


図 8.9 活動-資源図

活動-資源図で注意するものは，以下の通りである：

・図の中の各長方形は，決定変数と一致する.

・図の中の各円は，制約または目的と一致する.

・図の中の各矢線は，係数と一致する.

・各円または長方形は 1 単位（例えば時間またはブッシェル）と関連する.
・各矢線の単位（寸法）は，以下の通りである：長方形の単位／円の単位.

8.9.8 スパニングツリー

もう一つの単純であるが重要なネットワーク関連の問題は，スパニングツリーである. ケーブルテレビを家に提供するため，ケーブルの導入時に起こる. 接続したい一組の家があれば，我々は最小限の費用でそれらの家をネットワークに接続したい. ネットワーク費用は，ネットワークの長さ（弧）とする. 少し熟考すれば最小限の費用は，ネットワークがループ（ネットワークのどんな 2 つのノード（または家）でも，それらをつなぐ唯一 1 本の経路がある）を含まないことが分かる. そのようなネットワークを「スパニングツリー」と呼ぶ.

魅力的で単純なアルゴリズムは，最小費用のスパニングツリーを見つけることである（Kruskal（1956）参照）.

① 集合 Y = {2, 3, 4 … n}（ノード集合は結合されている）.
A = {1}（すでに結合された集合）. ノード 1 を親ノードと定義
② Y が空なら，次を実施
③ i が A で，j が Y にあるような最短のアーク(i, j)を見つけなさい
④ ネットワークにアーク(I, j) を加え，次の設定をする
A = A + j, Y = Y – j.
⑤ ②に戻る.

上記の単純で効率的なアルゴリズムのため，LP は最小スパニングツリー問題を解決する必要はない. 実際，LP で最小スパニングツリー問題を定式化することは，少し退屈である. 以下は，スパニングツリーの LINGO モデルである. このモデルは上記のステップで解かなくて，直接 IP で解決する.

```
MODEL:  !  (MNSPTREE);
! 一組のノードとそれらの間の距離を与え，ネットワーク上のリンクの総距離を最短にしてすべてのノードを接続することは，最小スパニングツリー（MST）と呼ばれる古典的な問題です.
このモデルは，アトランタ，シカゴ，シンシナティ，ヒューストン，LA，およびモントリオールを接続する最小コストのネットワークを見つけ，ネットワークを介してアトランタ（ベース）から他のすべての都市にメッセージを送信できるようにします;
SETS:
CITY / ATL CHI CIN HOU LAX MON/: LVL;
!LVL( I) =都市 I の水準: LVL( 1) = 0;
LINK( CITY,  CITY):  DIST,  ! The distance matrix;
            X;  ! X( I,  J) = 1 if we use link I,  J;
ENDSETS
DATA:  ！距離行列は対象でなくてもよい;
```

```
        ! 都市 1 は樹の基本;
        !to    Atl    Chi    Cin    Hou     LA    Mon ;
DIST =    0    702    454    842 2396 1196 !from Atlanta;
         702      0    324 1093 2136    764 !from Chicago;
         454    324      0 1137 2180    798 !from Cinci;
         842 1093 1137      0 1616 1857 !from Houston;
        2396 2136 2180 1616      0 2900 !from LA;
        1196    764    798 1857 2900      0;!from Montreal;
ENDDATA
!-------------------------------------------------;
! モデルサイズ：警告，N> 8 で遅くなることがあります.
N = @SIZE（CITY）;
!目的関数はリンクされた全ての距離を最小化;
MIN = @SUM( LINK: DIST * X);
! ベース K を除く，ベース以外は..  ;
@FOR( CITY( K)| K #GT# 1: ! It must be entered;
    @SUM( CITY( I)| I #NE# K: X( I,    K)) = 1;
! J-K からリンクがあれば，LVL（K）= LVL（J）+1 となる．  注：これらは大きな問題に対してはあまり強力ではない;
    @FOR( CITY( J)| J #NE# K:
        LVL( K) >= LVL( J) + X( J,    K)
          - ( N - 2) * ( 1 - X( J,    K))
          + ( N - 3) * X( K,    J); ); );
  LVL( 1) = 0;   ! City 1 has level 0;
! 都市 1 の外に弧がある必要があります;
@SUM( CITY( J)| J #GT# 1: X( 1,    J)) >= 1;
! X を 0/1 にする;
@FOR( LINK: @BIN( X); );
! ベース以外の都市の水準は少なくとも 1 で N-1 以下であり，ベースへのリンクがあれば 1 である;
@FOR( CITY( K)| K #GT# 1:
    @BND( 1,    LVL( K),    999999);
    LVL( K) <= N - 1 - ( N - 2) * X( 1,    K);   );
END
```

解は次の通りである.

```
  Optimal solution found at step:         16
```

Objective value:	4000.000	
Variable	Value	Reduced Cost
N	6.000000	0.0000000
LVL(CHI)	2.000000	0.0000000
LVL(CIN)	1.000000	0.0000000
LVL(HOU)	1.000000	0.0000000
LVL(LAX)	2.000000	0.0000000
LVL(MON)	3.000000	0.0000000
X(ATL, CIN)	1.000000	454.0000
X(ATL, HOU)	1.000000	842.0000
X(CHI, MON)	1.000000	764.0000
X(CIN, CHI)	1.000000	324.0000
X(HOU, LAX)	1.000000	1616.000

解は,アトランタがシンシナティとヒューストンにつながらなければならないことを示す．ヒューストンは，次に LA につながる．シンシナティはシカゴにつながり，そしてシカゴはまたモントリオールにつながる．

8.9.9 スタイナー木

スタイナー木は，最小スパニングツリーの一般化である．違いは与えられたネットワークで，指定されたノードの部分集合だけがスタイナー木につながれる．

例

チェスのゲームでグランドマスター（ゲイリーカスパロフ）を破った最初の PC（IBM RS6000)は,スタイナー木のような最適化法の助けを借りて設計された電子回路である．この PC の典型的な VLSI（Very Large Scale Integrated）回路は，線幅が 2 ミリメートル未満である．それでも，回路上の装置をつなぐために 120 メートル以上のパスがある．回路の速度を上げる重要な点は，回路上で経路の長さを減らすことである．**図 8.10** は，5 台の装置が一般木としてつながれた回路を示す．いろいろな装置が前もって設置されているため，唯一利用できる経路は，図で示したものである．4 角いノード A，B，C，D と E は，つながれなければいけない．丸いノード F，G などは，使われるかもしれないが，必ずしもつなぐ必要がない．リンクの総距離を最小にするために，リンクのどの集合を使う必要があるか？

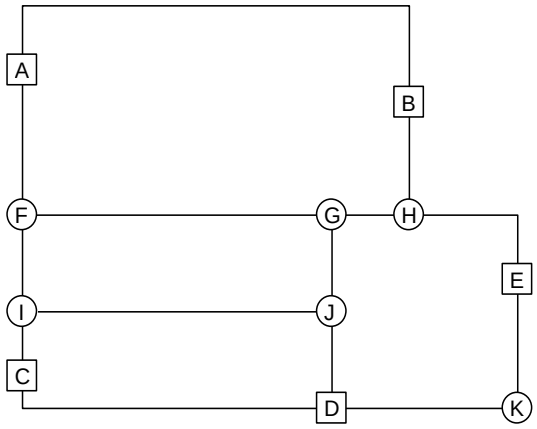


図 8.10 スタイナー木問題

最小の長さのスタイナー木を見つけることは，最小の長さスパニングツリーを見つけることよりかなり難しい．以下の LINGO モデルは，最小の長さのスタイナー木を捜しだす．データは，図 8.10 のネットワークと一致する．

```
MODEL:      !  (STEINERT);
!与えられたノードの集合、それらの間の距離、およびノードの指定されたサブセットは、総距離が最小になるようにリンクのセットを見出し、指定されたサブセット内のノードの各対の間に（ユニークな）パスがある。 これはスタイナー木問題と呼ばれています;
SETS:
 ALLNODE : U;
  ! U( I) = level of node I in the tree;
  ! U( 1) = 0;
 MUSTNOD( ALLNODE); ! The subset of nodes that must be connected;
 LINK( ALLNODE,   ALLNODE):
            DIST,     ! The distance matrix;
             X;       ! X( I, J) = 1 if we use link I,    J;
ENDSETS
DATA:
 ALLNODE=    ! Distance matrix need not be symmetric;
          A     B     C     D     E     F     G     H     I     J     K;
 DIST =0    14 999 999 999     4 999 999 999 999 999
        14     0 999 999 999 999 999     3 999 999 999
       999 999     0     9 999 999 999 999     2 999 999
       999 999     9     0 999 999 999 999 999     3     6
       999 999 999 999     0 999 999     5 999 999     3
         4 999 999 999 999     0 999 999     3 999 999       999 999 999 999 999 999     0     2 999     3
999
```

```
        999     3 999 999     5 999     2     0 999 999 999
        999 999     2 999 999     3 999 999     0     8 999
        999 999 999     3 999 999     3 999     8     0 999
        999 999 999     6     3 999 999 999 999 999     0;
! 接続する必要があるノードのサブセット.
  最初のノードは必須ノードでなければなりません;
MUSTNOD =   A B C D E;
ENDDATA
!-----------------------------------------------;
! モデルサイズ：警告，N> 8 で遅くなる;
N = @SIZE( ALLNODE);
! 目標はリンクの総距離を最小にすることです;
MIN = @SUM( LINK: DIST * X);
! ベースを除く各必須ノード K に対して，..  ;
@FOR( MUSTNOD( K)| K #GT# 1:
! 入力する必要があります;
  @SUM( ALLNODE( I)| I #NE# K: X( I, K)) = 1;
! U（J）=ノード J とノード 1 との間のアーク数を強制する.
注：これは，大きな問題で強くない.；
@FOR( ALLNODE( J)| J #GT# 1 #AND# J #NE# K:
    U( J) >= U( K) + X ( K, J) - ( N - 2) * ( 1 - X( K, J)) +( N - 3) * X( J, K); ););
! ノード 1 の外に弧が存在する必要があります;
@SUM( ALLNODE( J)| J #GT# 1: X( 1, J)) >= 1;
! ノード J からの弧がある場合は，;
@FOR( ALLNODE( J)| J #GT# 1:
 @FOR( ALLNODE( K)| K #NE# J:
  @SUM( ALLNODE( I)| I #NE# K #AND# I #NE# J:X( I, J)) >= X( J, K);); );
! X の 0 /1;
@FOR( LINK: @BIN( X); );
! ベースを除くノードの水準は，少なくとも 1 で N-1 以下であり，ベースへのリンクがあれば 1
である;
@FOR( ALLNODE( K)| K #GT# 1:
 @BND( 1, U( K), 999999);
 U( K) < N - 1 - ( N - 2) * X( 1, K);    );
END
```

解は費用が 33 である．用いられているリンクは次の通りである.

```
Optimal solution found at step:        30
Objective value:                    33.00000
Branch count:                              0
Variable           Value        Reduced Cost
X( A,  F)       1.000000            4.000000
X( F,  I)       1.000000            3.000000
X( G,  H)       1.000000            2.000000
X( H,  B)       1.000000            3.000000
X( H,  E)       1.000000            5.000000
X( I,  C)       1.000000            2.000000
X( I,  J)       1.000000            8.000000
X( J,  D)       1.000000            3.000000
X( J,  G)       1.000000            3.000000
```

対応する最小ステイナー木は**図 8.11** である.

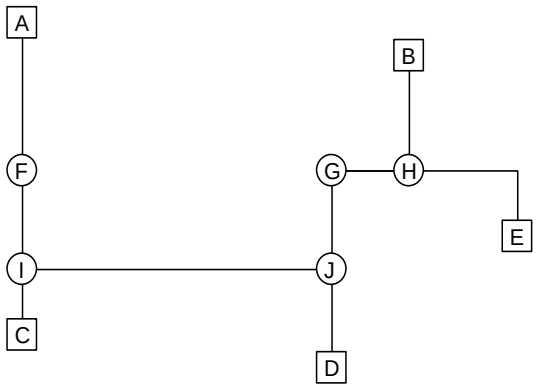


図 8.11 最短ステイナー木

ノードK(図8.11) は木にない. この例は半導体回路の重要な2つの特徴を欠いている:

① 図8.11は平面的なパスだけ示す. 実際は, 層を重ねることで3次元のパスが可能である.

② 図8.11は1本の木だけを示す. 実際は, 多くの木があるかもしれない(例えば, ある装置は電気的な「地面に」接続, 他はクロックに接続する別々の回路がある).

8.10 非線形ネットワーク

目的関数が非線形であるか, ネットワークに非線形な条件が付加的される重要な問題がある. 最初の例は, 目的地への船積みの量や項目を割り当てる値が, 次の条件に左右される輸送問題である.

(a)何項目かその行先に既に出荷され，(b)項目のタイプとタイプの行先が分かっている．軍隊では，この種の問題は「武器の割当」問題として知られている：

X(i，j)=仕事 i に割り当てられるタイプ j の単位数；
p(i，j)=Prob{タイプ j が首尾よく仕事 i を達成しない単位}；

そして独立性を仮定して，仕事 i が首尾よく達成できる確率は次の通りである．

$$1 - \pi_j \, p(i, j)^{x(i, j)}$$

適切な目的関数は，首尾よく完了する仕事の確率を最大にすることである． 次のモデルは Bracken ＆ McCormick(1968)からのデータを使用してこの考えを説明する．

```
MODEL:
! (TARGET)  Bracken and McCormick;
SETS:
DESTN/1.20/: VALUE,  DEM,  LFAILP;
SOURCE/1.5/: AVAIL;
DXS( DESTN,  SOURCE): PROB,  VOL;
ENDSETS
DATA:
! ソース J のユニットが目的地 I で仕事をしない確率;
PROB=
1.00      .84      .96      1.00      .92
 .95      .83      .95      1.00      .94
1.00      .85      .96      1.00      .92
1.00      .84      .96      1.00      .95
1.00      .85      .96      1.00      .95
 .85      .81      .90      1.00      .98
 .90      .81      .92      1.00      .98
 .85      .82      .91      1.00     1.00
 .80      .80      .92      1.00     1.00
1.00      .86      .95       .96      .90
1.00     1.00      .99       .91      .95
1.00      .98      .98       .92      .96
1.00     1.00      .99       .91      .91
1.00      .88      .98       .92      .98
1.00      .87      .97       .98      .99
1.00      .88      .98       .93      .99
1.00      .85      .95      1.00     1.00
```

```
 .95          .84          .92          1.00          1.00
1.00          .85          .93          1.00          1.00
1.00          .85          .92          1.00          1.00;
! 各ソースで利用可能なユニット;
AVAIL= 200          100          300          150          250;
! 各目的地に必要な最小単位;
DEM=
 30     0     0     0     0 100     0     0     0    40
  0     0     0    50    70    35     0     0     0    10;
! 目的地 J の値;
VALUE=
 60    50    50    75    40    60    35    30    25 150
 30    45 125 200 200 130 100 100 100 150;
ENDDATA
! I 以上の最大和：（destn I の値）* Prob {I の成功};
      MAX = @SUM( DESTN( I): VALUE( I) *
                  ( 1 - @EXP( LFAILP( I))));
! 供給制約;
@FOR( SOURCE( J):
    @SUM( DESTN( I): VOL( I,    J)) <= AVAIL( J));
@FOR( DESTN( I):
! 需要の制約;
@SUM( SOURCE( J): VOL( I,    J)) > DEM( I);
! 宛先 I の失敗確率のログを計算;
@FREE( LFAILP( I));
 LFAILP( I) = @SUM(SOURCE(J): @LOG(PROB(I，J)) * VOL(I，J)););
END
```

目的関数で，LFAILP（i)を計算する式を使用することで簡単になる．しかし，必要な計算時間はおそらくこれで大幅に増加する(？訳注:1 秒未満で求まる)．理由は目的関数で非線形を表す変数の数が劇的に増えるからである．非線形プログラムの一般ルールは次の通りである:

「目的関数や制約式で，非線形にある変数の数を，中間変数を定義して線形制約に置き換えることができれば，多分計算時間を改良できる．」

LFAILP（i)を定義する制約が，LFAILP（i)および Vol（i，j)の両方で線形であることを確認しなさい.

解は実数を含む．モデルの簡単な一般化は，Vol()変数が整数であることを要求することである．

```
Optimal solution found at step:       152
Objective value:                1735.570
   Variable           Value        Reduced Cost
 VOL( 1, 5)        50.81594           0.0000000
 VOL( 2, 1)        13.51739           0.0000000
 VOL( 2, 2)        1.360311           0.2176940
 VOL( 2, 5)        45.40872           0.0000000
 VOL( 3, 5)        48.62891           0.0000000
 VOL( 4, 2)        23.48955           0.0000000
 VOL( 5, 2)        20.89957           0.0000000
 VOL( 6, 1)        100.0000           0.0000000
 VOL( 7, 1)        39.10010           0.0000000
 VOL( 8, 1)        27.06643           0.0000000
 VOL( 8, 5)      0.4547474E-12        0.0000000
 VOL( 9, 1)        20.31608           0.0000000
VOL( 10, 5)        51.13144           0.0000000
VOL( 11, 4)        33.19754           0.0000000
VOL( 12, 4)        40.93452           0.0000000
VOL( 13, 5)        54.01499           0.0000000
VOL( 14, 4)        58.82350           0.0000000
VOL( 14, 5)      0.5684342E-13        0.0000000
VOL( 15, 2)        26.21095           0.0000000
VOL( 15, 3)        43.78905           0.0000000
VOL( 16, 2)        24.23657           0.2176940
VOL( 16, 4)        17.04444           0.0000000
VOL( 17, 2)        3.803054           0.0000000
VOL( 17, 3)        72.03255          -0.4182476E-05
VOL( 18, 2)      0.8881784E-15        0.1489908
VOL( 18, 3)        57.55117           0.0000000
VOL( 19, 3)        64.21183           0.0000000
VOL( 20, 3)        62.41540           0.0000000
```

規則はネットワークを流れる物質(例えば，水，ガス，または電気)のタイプに左右される．ネットワークの平衡は２組の変数で記述されている：

a)各アークを通る流れ;
b)各ノードの圧力(例えば，電気ネットワークの電圧)
　平衡で，(a)および(b)の値はネットワークの平衡を定める規則を満たさなければならない．一般的に，これらの規則は次の通りである:
①各ノードで，フロー制約の標準的な conservatio は流れの制約の値に適用する;
②各アークで，2 つのエンドポイントの間の圧力相違は，アーク上の流れおよびアークの抵抗と関連している．電気ネットワークで，条件(ii)は電圧差 V である．ワイヤーによって接続される 2 点間で抵抗 R（オーム）と I アンペアの流れは次の関係を満たさなければならない:v=i＊R
　制約式(ii)が非線形になりがちである．次のモデルは，都市のための簡単な水配電網の平衡計算によって説明する．ポンプは 2 つのノードで指定圧力を適用する．他のノードで，水は指定率で取除かれる．　私達は各ノードで各アークの意味された流動度および圧力を定めたいと思う．

```
MODEL:
! ネットワーク平衡 NETEQ1：Hansen らの計算に基づく．;
SETS:
NODE/A, B, C, D, E, F, G, H/: P;!Pressure at this node;
ARC( NODE, NODE)/ B A, C A, C B, D C, E D, F D, G D,
 F E, H E, G F, H F/: R,!Resistance on this arc;
FLO;        ! Flow on this arc;
 SRC(NODE)/G, H/: PFIXED; ! ソースノードでの固定または所定の圧力;
 DEST(NODE) | #noT# @IN(SRC,&1): DEMAND;
  ! 宛先ノードにおける要求;
ENDSETS
DATA:
PFIXED =   240, 240;
DEMAND =   1, 2, 4, 6, 8, 7;
R =   1, 25, 1, 3, 18, 45, 1, 12, 1, 30, 1;
PPAM = 1; ! Compressibility parameter;
! 非圧縮性流体と電気の場合：PPAM = 1,ガスの場合：PPAM = 2;
FPAM = 1.852;   !Resistance due to flow parameter;
! 電気ネットワーク ：FPAM = 1;
! 他の流体：1.8≦FOAM≦2;
! 最適化ネットワークの場合： flow> = 0 のアークに対して FPAM = 0;
ENDDATA
```

```
! ソース/リザーバノードの圧力を設定する;
@FOR( SRC( I): P( I) = PFIXED( I); );
! 非ソースノードでのフローの保存;
@FOR( DEST( J):
  @SUM( ARC( I, J): FLO( I, J)) = DEMAND( J) +
  @SUM( ARC( J, K): FLO( J, K)););
! 各円弧の 2 つの端の圧力を関連付ける;
@FOR( ARC( I, J):
P(I)^ PPAM - P(J)^ PPAM = R(I,J)* FLO(I,J)^ FPAM;);
END
```

次の解が各ノードの流れの保存を満たし，各アーク上の圧力降下がモデルの抵抗方程式を満たすことを確認しなさい.

```
     Feasible solution found at step:       22
       Variable           Value
           PPAM        1.000000
           FPAM        1.852000
          P( A)        42.29544
          P( B)        42.61468
          P( C)        48.23412
          P( D)        158.4497
          P( E)        188.0738
          P( F)        197.3609
          P( G)        240.0000
          P( H)        240.0000
       R( B, A)        1.000000
       R( C, A)        25.00000
       R( C, B)        1.000000
       R( D, C)        3.000000
       R( E, D)        18.00000
       R( F, D)        45.00000
       R( F, E)        12.00000
       R( G, D)        1.000000
       R( G, F)        30.00000
       R( H, E)        1.000000
       R( H, F)        1.000000
     FLO( B, A)        0.5398153
```

FLO(C, A)	0.4601847
FLO(C, B)	2.539815
FLO(D, C)	7.000000
FLO(E, D)	1.308675
FLO(F, D)	0.9245077
FLO(F, E)	0.8707683
FLO(G, D)	10.76682
FLO(G, F)	1.209051
FLO(H, E)	8.437907
FLO(H, F)	7.586225
PFIXED(G)	240.0000
PFIXED(H)	240.0000
DEMAND(A)	1.000000
DEMAND(B)	2.000000
DEMAND(C)	4.000000
DEMAND(D)	6.000000
DEMAND(E)	8.000000
DEMAND(F)	7.000000

References

[1]. Adams, J.L. (1986), Conceptual Blockbusting, Addison-Wesley, Reading, MA.

[2]. Adams, W. and H. Sherali. (2005), "A Hierarchy of Relaxations Leading to the Convex Hull Representation for General Discrete Optimization Problems", Annals of Operations Research, vol. 140, pp.21-47.

[3]. Ahuja, R. K., T. L. Magnanti, J. B. Orlin (1993), Network Flows, Theory, Algorithms, and Applications, Prentice-Hall, Englewood Cliffs, NJ.

[4]. Andrews, B. and H. Parsons. (1993), "Establishing Telephone-Agent Staffing Levels through Economic Optimization", Interfaces, Vol. 23, No. 2, pp. 14-20.

[5]. Arnold, L., and D. Botkin. "Portfolios to Satisfy Damage Judgement: A Linear Programming Approach", Interfaces, Vol. 8, No. 2 (Feb. 1978).

[6]. Aykin, T. (1996), "Optimal Shift Scheduling with Multiple Break Windows", Management Science, Vol . 42 No. 4 (April), pp. 591-602.

[7]. Baker, E. K. and M. L. Fisher (1981), "Computational Results for Very Large Air Crew Scheduling Problems", Omega, Vol. 9, pp. 613-618.

[8]. Balas, E. (1979), "Disjunctive Programming", Annals of Discrete Mathematics, Vol. 5, pp. 3-51.

[9]. Banks, J. and R. Gibson. (1997), "10 Rules for Determining When Simulation is Not Appropriate", IIE Solutions, Vol. 29, No. 9 (September), pp. 30-32.

[10]. Barnett, A. (1994), "How Numbers Can Trick You", Technology Review, MIT, Vol. 97, No. 7(October), pp. 38-45.

[11]. Belobaba, P.P. (1989), "Application of a Probabilistic Decision Model to Airline Seat Inventory Control", Operations Research, vol. 37, no. 2, (March-April) pp. 183-197.

[12]. Bessent, A., W. Bessent, J. Kennington, and B. Reagan (1982), "An Application of Mathematical Programming to Assess Productivity in the Houston Independent School District", Management Science, Vol. 28, No. 12 (December), pp. 1355-1367.

[13]. Birge, J. R. (1997), "Stochastic Programming Computation and Applications", INFORMS Journal on Computing, Vol. 9, No. 2, pp.111-133.

[14]. Birge, J. and F. Louveaux. (1997), Introduction to Stochastic Programming, Springer-Verlag, New York, NY.

[15]. Black, F. (1989), "How We Came Up with the Option Formula", The Journal of Portfolio Management, Winter, pp. 4-8.

[16]. Black, F., E. Derman, and W. Toy. (1990), “A One-Factor Model of Interest Rates and Its Application to Treasury Bond Options”, Financial Analyst Journal, Vol. 46, pp. 33-39.

[17]. Black, F., and M. Scholes (1973), “The Pricing of Options and Corporate Liabilities”, Journal of Political Economy, Vol. 81, pp. 637-654.

[18]. Bland, R. G. and D. F. Shallcross (1989), “Large Traveling Salesman Problems Arising in X-ray Crystallography: A Preliminary Report on Computation”, O.R. Letters, Vol. 8, No. 3, pp. 125-128.

[19]. Bosch, R.A. (1993), "Big Mac Attack, The Diet Problem revisited: Eating at McDonald's", OR/MS Today, (August), pp. 30-31.

[20]. Bracken, J. and G.P. McCormick. (1968), Selected Applications of Nonlinear Programming, John Wiley & Sons, Inc., New York, NY.

[21]. Bradley, G. H., G. G. Brown and G. W. Graves. (1977), “Design and Implementation of Large Scale Primal Transshipment Algorithms”, Management Science, Vol. 24, pp. 1-34.

[22]. Bradley, S.P., A. C. Hax and T. L. Magnanti. (1977), Applied Mathematical Programming, Addison-Wesley Publishing Company, Reading, Mass.

[23]. Braess, D. (1968), “Uber ein Paradoxon aus der Verkehplanung”, Unternehmensforschung, Vol. 12, pp. 258-268.

[24]. Brearley, A. L., G. Mitra and H. P. Williams. (1975), “An Analysis of Mathematical Programming Problems Prior to Applying the Simplex Algorithm”, Mathematical Programming, Vol. 8, pp. 54-83.

[25]. Brown, G.G., R.F. Dell, and R.K. Wood. (1997), “Optimization and Persistence”, Interfaces, Vol. 27, No. 5, (Sept-Oct), pp. 15-37.

[26]. Brown, G.G., C.J. Ellis, G.W. Graves, and D. Ronen (1987), “Real-Time, Wide Areas Dispatch of Mobil Tank Trucks”, Interfaces, Vol. 17, No. 1, pp. 107-120.

[27]. Brown, G. G. and D. S. Thomen (1980), “Automatic Identification of Generalized Upper Bounds in Large-Scale Optimization Models”, Management Science, Vol. 26, No. 11, pp. 1166-1184.

[28]. Carino, D.R., T. Kent, D.H. Myers, C. Stacy, M. Sylvanus, A.L. Turner, K. Watanabe, and W.T. Ziemba (1994), “The Russell-Yasuda Kasai Model: An Asset/Liability Model for a Japanese Insurance Company Using Multistage Stochastic Programming”, Interfaces, Vol. 24, No. 1, pp. 29-49.

[29]. Charnes, A., W.W. Cooper and E. Rhodes (1978), “Measuring the Efficiency of Decision Making Units”, European Journal of Operational Research, Vol. 2 (1978) pp. 429-444.

[30]. Ciriani, T.A. and R. C. Leachman (1993), Optimization in Industry, John Wiley & Sons, Chichester.

[31]. Clarke, G. and J. W. Wright (1964), “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points”, Operations Research, Vol. 12, No. 4 (July-Aug.), pp. 568-581.

[32]. Claus, A. (1984), “A New Formulation for the Travelling Salesman Problem”, SIAM Journal on Algebraic and Discrete Methods, vol. 5, no. 1, pp. 21-25.

[33]. Clyman, D.R. (1995), “Unreasonable Rationality?”, Management Science, Vol 41, No. 9 (Sept.), pp. 1538-1548.

[34]. Craven, J. P. (2001), The Silent War: the Cold War Battle Beneath the Sea, Simon & Schuster, New York.

[35]. Dantzig, G. (1963), Linear Programming and Extensions, Princeton University Press, Princeton.

[36]. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson(1954), “Solution of a Large-Scale Traveling-Salesman Problem”, Operations Research, vol. 2, no. 4, pp. 393-410.

[37]. Dantzig, G. and N. N. Thapa (1997), Linear Programming, Vol. 1, Springer, New York.

[38]. Dantzig, G. and B. Wolfe (1960), “Decomposition Principle for Linear Programs”, Operations Research, Vol. 8, pp. 101-111.

[39]. Danusaputro, S., C. Lee, and L. Martin-Vega (1990), “An Efficient Algorithm for Drilling Printed Circuit Boards”, Computers and Industrial Engineering, Vol. 18, pp. 145-151.

[40]. Dauch, R.E. (1993), Passion for Manufacturing, Society of Manufacturing Engineers, Dearborn, MI.

[41]. Davis, L. S. and K. N. Johnson (1987), Forest Management, 3rd ed., McGraw-Hill Company.

[42]. Dembo, R.S., A. Chiarri, J.G. Martin, and L. Paradinas (1990), “Managing Hidroeléctrica Española's Hydroelectric Power System”, Interfaces, Vol. 20, No. 1 (Jan.-Feb.), pp. 115-135.

[43]. d'Epenoux, F. (1963), “A Probabilistic Production and Inventory Problem”, Management Science, Vol. 10, No. 1 (Oct), pp. 98-108.

[44]. DeRosa, D. (1992), Options on Foreign Exchange, Irwin Professional Publishing, New York.

[45]. DeWitt, C. W., L. Lasdon, A. Waren, D. Brenner and S. Melhem (1989), “OMEGA: An Improved Gasoline Blending System for Texaco,” Interfaces, Vol. 19, No. 1 (Jan.-Feb.), pp. 85-101.

[46]. Dikin, I. I. (1967), “Iterative Solution of Problems of Linear and Quadratic Programming”, Soviet Mathematics Doklady, Vol. 8, pp. 674-675.

[47]. Dial, R.B. (1994), “Minimizing Trailer-on-Flat-Car Costs: A Network Optimization Model”, Transportation Science, Vol. 28, pp. 24-35.

[48]. Ding, X. and M. Puterman(2002), “The Censored Newsvendor and the Optimal Acquisition of Information”, Operations Research, vol. 50, no. 3, pp. 517-527.

[49]. Dutton, R., G. Hinman and C. B. Millham (1974), "The Optimal Location of Nuclear-Power Facilities in the Pacific Northwest", Operations Research, Vol. 22, No. 3 (May-June), pp. 478-487.

[50]. Dyckhoff, H. (1981), "A New Linear Programming Approach to the Cutting Stock Problem", Operations Research, Vol. 29, No. 6 (Nov.-Dec.), pp. 1092-1104.

[51]. Edie, L. C. (1954), "Traffic Delays at Toll Booths", Operations Research, Vol. 2, No. 2 (May), pp. 107-138.

[52]. Elshafei, A. (1977), "Hospital Lay-out as a Quadratic Assignment Problem", Operational Research Quarterly, Vol. 28, pp. 167-169.

[53]. Emmelhainz, L. W., M. A. Emmelhainz, and J. R. Stock (1991), "Logistics Implications of Retail Stockouts", Journal of Business Logistics, Vol. 12, No. 2, pp. 129-142.

[54]. Eppen, G., K. Martin, and L. Schrage (1988), "A Scenario Approach to Capacity Planning", Operations Research, Vol. 37, No. 4 (July-August), pp. 517-530.

[55]. Eppen, G. D. and R. K. Martin (1987), "Solving Multi-Item Capacitated Lot-Sizing Problems Using Variable Redefinition." Operations Research, Vol. 35, No. 6 (Nov.-Dec.), pp. 832-848.

[56]. Farley, A. A. (1990), "A Note on Bounding a Class of Linear Programming Problems, Including Cutting Stock Problems", Operations Research, Vol. 38, No. 5 (Sept.-Oct.), pp. 922-923.

[57]. Fields, C., J. F. Hourican and E. A. McGee (1978), "Developing a Minimum Cost Feed Blending System for Intensive Use", Joint National TIMS/ORSA Meeting, New York, NY.

[58]. Fieldhouse, M. (1993), "The Pooling Problem", Optimization in Industry, (Eds.) T. A. Ciriani and R. C. Leachman, John Wiley & Sons.

[59]. Fillon, M. (1996), "Taming the Yangtze", Popular Mechanics, Vol. 173, No. 7 (July), pp. 52-56.

[60]. Fisher, M. and A. Raman (1996), "Reducing the Cost of Demand Uncertainty Through Accurate Response to Early Sales", Operations Research, Vol. 44, No. 1 (Jan.-Feb.), pp. 87-99.

[61]. Florian, M (1977), "An Improved Linear Approximation Algorithm for the Network Equilibrium (Packet Switching) Problem", Proceedings 1977 IEEE Conference Decision and Control.

[62]. Fudenberg, D. and J. Tirole (1993) Game Theory, The MIT Press, Cambridge, MA.

[63]. Gaballa, A. and W. Pearce. (1979), "Telephone Sales Manpower Planning at Qantas", Interfaces, Vol. 9, No. 3,(May), pp. 1-9.

[64]. Geoffrion, A. (1976), "The Purpose of Mathematical Programming is Insight, Not Numbers", Interfaces, Vol. 7, No. 1 (November), pp. 81-92.

[65]. Geoffrion, A. and G. W. Graves. (1974), “Multicommodity Distribution System Design by Benders Decomposition”, Management Science, Vol. 20, No. 5 (January), pp. 822-844.

[66]. Glover, F. and D. Klingman (1977), “Network Applications in Industry and Government”, AIIE Transactions, Vol. 9, pp. 363-376.

[67]. Golabi, K., R.B. Kulkarni, and G.B. Way (1982), “A Statewide Pavement Management System”, Interfaces, Vol. 12, No. 6 (Nov.-Dec.), pp. 5-21.

[68]. Gomory, R. E. (1958), “Outline of an Algorithm for Integer Solutions to Linear Programs”, Bulletin of the American Mathematical Society, Vol. 64, pp. 275-278.

[69]. Grandine, T.A. (1998), "Assigning Season Tickets Fairly", Interfaces, Vol. 28, No. 4(July-August), pp. 15-20.

[70]. Graves, R., J. Sankaran, and L. Schrage (1993), “An Auction Method for Course Registration", Interfaces, Vol. 23, No. 5 (1993), pp. 81-92.

[71]. Greenberg, H.J. (1978), Design and Implementation of Optimization Software, Sijthoff & Noordhoff.

[72]. Grinold, R.C. (1983), “Model Building Techniques for the Correction of End Effects in Multistage Convex Programs”, Operations Research, Vol. 31, No. 3, pp. 407-431.

[73]. Gross, D. and C. Harris. (1998), Fundamentals of Queueing Theory, 3rd ed., Wiley Interscience, New York.

[74]. Grötschel, M., M. Jünger and G. Reinelt (1985), “Facets of the Linear Ordering Polytope”, Mathematical Programming, Vol. 33, pp. 43-60.

[75]. Gunawardane, G., S. Hoff and L. Schrage (1981), “Identification of Special Structure Constraints in Linear Programs”, Mathematical Programming, Vol. 21, pp. 90-97.

[76]. Hadley, G. (1962), Linear Programming, Addison-Wesley.

[77]. Hane, C.A., C. Barnhart, E.L. Johnson, R.E. Marsten, G.L. Nemhauser, G. Sigismondi (1995), “The Fleet Assignment Problem: Solving a Large Scale Integer Program”, Mathematical Programming, Vol. 70, pp. 211-232.

[78]. Hansen, C.T., K. Madsen, and H.B. Nielsen (1991), “Optimization of Pipe Networks”, Mathematical Programming, Vol. 52, pp. 45-58.

[79]. Hanson, W. and R. K. Martin (1990), “Optimal Bundle Pricing”, Management Science, Vol. 36, No. 2 (February), pp. 155-174.

[80]. Haverly, C. A. (1978), “Studies of the Behavior of Recursion for the Pooling Problem”, SIGMAP Bulletin, Association for Computing Machinery.

[81]. Heath, D., R. Jarrow, and A. Morton (1992), “Bond Pricing and the Term Structure of Interest Rates: A New Methodology for Contingent Claims Valuation”, Econometrica, Vol. 60, No. 1, pp. 77-105.

[82]. Held, M. and R. Karp. (1962), “A Dynamic Programming Approach to Sequencing Problems”, SIAM Journal of Applied Math, vol. 10, no. 1, pp. 196-210.

[83]. Infanger, G. (1994), Planning Under Uncertainty: Solving Large-Scale Stochastic Linear Programs, Boyd & Fraser, Danvers, MA.

[84]. Jackson, J.R. (1963), “Jobshop-Like Queueing Systems”, Management Science, Vol. 10, No. 1, pp. 131-142.

[85]. Jenkins, L. (1982), “Parametric Mixed Integer Programming: An Application to Solid Waste Management”, Management Science, Vol 28, No. 11 (Nov.), pp. 1270-1284.

[86]. Jeroslow, R.G., K. Martin, R.L. Rardin, J. Wang (1992),“Gainfree Leontief Substitution Flow Problems”, Mathematical Programming, Vol. 57, pp. 375-414.

[87]. Jorion, P. (2001), Value at Risk, 2nd ed., McGraw-Hill.

[88]. Kall, P. and S.W. Wallace (1994), Stochastic Programming, John Wiley & Sons, New York, NY.

[89]. Karmarkar, N. K. (1985), “A New Polynomial Time Algorithm for Linear Programming”, Combinatorica, Vol. 4, pp. 373-395.

[90]. Kehoe, T.J. (1985), “A Numerical Investigation of Multiplicity of Equilibria”, Mathematical Programming Study 23, pp. 240-258.

[91]. Khachian, L. G. (1979), “A Polynomial Algorithm in Linear Programming”, Soviet Mathematics Doklady, Vol. 20, No. 1, pp. 191-194.

[92]. King, R. H. and Love, R. R. (1980), “Coordinating Decisions for Increased Profits”, Interfaces, Vol. 10, No. 6 (December), pp. 4-19.

[93]. Konno, H. and H. Yamazaki (1991), “Mean-Absolute Deviation Portfolio Optimization Model and Its Applications to Tokyo Stock Market”, Management Science, Vol. 37, No. 5 (May), pp. 519-531.

[94]. Kontogiorgis, S. and S. Acharya (1999), "US Airways Automates Its Weekend Fleet Assignment", Interfaces, Vol. 29, No. 3(May-June), pp. 52-62).

[95]. Koopmans, T. and M. Beckmann (1957), “Assignment Problems and the Location of Economic Activities”, Econometrica, Vol. 25, pp. 53-76.

[96]. Kruskal, Jr., J. B. (1956), “On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem”, Proc. Amer. Math. Soc., Vol. 7, pp. 48-50.

[97]. Lasdon, L. S., and Terjung, R. C. (1971), “An Efficient Algorithm for Multi-Item Scheduling”, Operations Research, Vol. 19, No. 4, pp. 946-69.

[98]. Lawler, E. L. (1963), “The Quadratic Assignment Problem”, Management Science, Vol. 19, pp. 586-599.

[99]. Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (1985), “The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization”, John Wiley & Sons.

[100]. Leontief, W. (1951), The Structure of American Economy, 1919-1931, Oxford University Press, New York, NY.

[101]. Levy, F. K. (1978), “Portfolios to Satisfy Damage Judgements: A Simple Approach”, Interfaces, Vol. 9, No. 1 (Nov.), pp. 106-107.

[102]. Lin, S. and B. Kernighan (1973), “An Effective Heuristic Algorithm for the Traveling Salesman Problem”, Operations Research, Vol. 21, pp. 498-516.

[103]. Little, J. D. C. (1961), "A Proof of the Queuing Formula L = λW", Operations Research, Vol. 9, No. 3 (May-June), pp. 383-387.

[104]. Madansky, A. (1962), “Methods of Solution of Linear Programs Under Uncertainty”, Operations Research, Vol. 10, pp. 463-471.

[105]. Mangasarian, O.L. (1993), “Mathematical Programming in Neural Networks”, ORSA Journal on Computing, Vol. 5, No. 4, pp. 349-360.

[106]. Manne, A. (1960), “Linear Programming and Sequential Decisions”, Management Science, Vol. 6, No. 3 (April), pp. 259-267.

[107]. Markowitz, H. M. (1959), Portfolio Selection, Efficient Diversification of Investments, John Wiley & Sons, Inc.

[108]. Markowitz, H. and A. Perold (1981), “Portfolio Analysis with Scenarios and Factors”, Journal of Finance, Vol. 36, pp. 871-877.

[109]. Marsten, R. E., M. P. Muller and C. L. Killion (1979), “Crew Planning at Flying Tiger: A Successful Application of Integer Programming”, Management Science, Vol. 25, No. 12 (Dec.), pp. 1175-1183.

[110]. Marsten, R. E., M. J. Saltzman, D. F. Shanno, G. S. Pierce, and J. F. Ballintijn (1989), “Implementation of a Dual Affine Interior Point Algorithm for Linear Programming”, ORSA J. on Computing, Vol. 1, No. 4, pp. 287-297.

[111]. Martin, R. K. (1999) Large Scale Linear and Integer Optimization: A Unified Approach, Kluwer Academic Publishers, Boston.

[112]. Maschler, M., B. Peleg, and L. S. Shapley (1979), “Geometric Properties of the Kernel, Nucleolus, and Related Solution Concepts”, Mathematics of Operations Research, Vol. 4, No. 4 (Nov.), pp. 303-338.

[113]. Mehrabian, S. G. Jahanshahloo, M. Alirezaee, and G. Amin (2000) “An Assurance Interval for the non_Archimedean Epsilon in DEA Models”, Operations Research, Vol. 48, No. 2, pp. 344-347.

[114]. Miller, H. E., W. P. Pierskalla and G. J. Rath (1976), “Nurse Scheduling using Mathematical Programming”, Operations Research, Vol. 24, pp. 857-870.

[115]. Miller, C. E., A. W. Tucker, and R. A. Zemlin. (1960), ”Integer Programming Formulations and Travelng Salesman Problems”, Journal of ACM, pp. 326-329.

[116]. Moldovanu, B. and M. Tietzel (1998) "Goethe's Second-Price Auction", Journal of Political Economy, Vol. 106, No. 4, pp. 854-858

[117]. Murchland, J. D. (1970), “Braess's Paradox of Traffic Flow”, Transportation Research, Vol. 4, pp. 391-394.

[118]. Nahmias, S. (1997) Production and Operations Analysis, 3rd ed., Irwin Publishing, Homewood, IL.

[119]. Nauss, R. M. (1986), “True Interest Cost in Municipal Bond Bidding: An Integer Programming Approach”, Management Science, Vol. 32, No. 7, pp. 870-877.

[120]. Nauss, R. M. and B. R. Keeler (1981), “Minimizing Net Interest Cost in Municipal Bond Bidding”, Management Science, Vol. 27, No. 4 (April), pp. 365-376.

[121]. Nauss, R. M. And R. Markland (1981), “Theory and Application of an Optimization Procedure for Lock Box Location Analysis”, Management Science, Vol. 27, No. 8 (August), pp. 855-865.

[122]. Neebe, A. W. (1987), “An Improved, Multiplier Adjustment Procedure for the Segregated Storage Problem”, Journal of the Operational Research Society, Vol. 38, No. 9, pp. 1-11.

[123]. Orlin, J.B. (1982), “Minimizing the Number of Vehicles to Meet a fixed Periodic Schedule: An Application of Periodic Posets”, Operations Research, Vol. 30, No. 4, pp. 760-776.

[124]. Nemhauser, G. L. and L. A. Wolsey (1988), Integer and Combinatorial Optimization, John Wiley & Sons, Inc.

[125]. Padberg, M. and G. Rinaldi (1987), “Optimization of a 532-City Symmetric Traveling Salesman Problem by Branch and Cut”, Operations Research Letters, Vol. 6, No. 1.

[126]. Palmquist, J., Uryasev, S., and Krokhmal, P.(2002), “Portfolio Optimization with Conditional Value-at-Risk Objective and Constraints”, The Journal of Risk, vol. 4, pp. 11-27.

[127]. Parker, R.G. and R.L. Rardin (1988), Discrete Optimization, Academic Press, San Diego.

[128]. Peiser, R.B. and S.G. Andrus (1983), “Phasing of Income-Producing Real Estate”, Interfaces, Vol. 13, No. 5 (Oct), pp. 1-9.

[129]. Perold, A. F. (1984), “Large Scale Portfolio Optimization”, Management Science, Vol. 30, pp. 1143-1160.

[130]. Plane, D. R. and T. E. Hendrick (1977), “Mathematical Programming and the Location of Fire Companies for the Denver Fire Department”, Operations Research, Vol. 25, No. 4 (July-August), pp. 563-578.

[131]. Pritzker, A., L. Watters, and P. Wolfe (1969), “Multiproject Scheduling with Limited Resources: a Zero-One Programming Approach”, Management Science, Vol. 16, No. 1 (Sept.), pp. 93-108.

[132]. Puterman, M. L. (1994), Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley.

[133]. Quinn, P., B. Andrews, and H. Parsons(1991) “Allocating Telecommunications Resources at L.L. Bean, Inc., Interfaces, vol. 21, no. 1, pp. 75-91.

[134]. Rardin, R. L. (1998), Optimization in Operations Research, Prentice Hall, New Jersey.

[135]. Rigby, B., L. Lasdon, and A. Waren (1995),“The Evolution of Texaco’s Blending Systems: From Omega to StarBlend”, Interfaces, Vol. 25, No. 5, pp. 64-83.

[136]. Roache, P. J.(1998), Verification and Validation in Computational Science and Engineering, Hermosa, NM. ISBN 0-913478-08-3.

[137]. Rogers, D.F., R.D. Plante, R.T. Wong, and J.R. Evans (1991), “Aggregation and Disaggregation Techniques and Methodology in Optimization”, Operations Research, Vol. 39, No. 4 (July-August), pp. 553-582.

[138]. Ross, G. T. and R. M. Soland (1975), “Modeling Facility Location Problems as Generalized Assignment Problems”, Management Science, Vol. 24, pp. 345-357.

[139]. Rosenthal, R. and R. Riefel (1994), "Optimal Order-Picking", Bulletin for the ORSA/TIMS Detroit Meeting, INFORMS, Baltimore, MD.

[140]. Rothstein, M. (1985), “OR and the Airline Overbooking Problem”, Operations Research, Vol. 33, No. 2 (March-April), pp. 237-248.

[141]. Roy, A. D. (1952), “Safety First and the Holding of Assets”, Econometrica, Vol. 20 (July), pp. 431-439.

[142]. Samuelson, D. (1999), “Predictive Dialing for Outbound Telephone Call Centers”, Interfaces, vol. 29, no. 5,(Sept-Oct), pp. 66-81.

[143]. Sankaran, J. (1989), Bidding Systems for Certain Nonmarket Allocations of Indivisible Items, Ph.D. dissertation, University of Chicago.

[144]. Schrage, L. (1975), “Implicit Representation of Variable Upper Bounds in Linear Programming”, Mathematical Programming, Study 4, pp. 118-132.

[145]. Schrage, L. (1978), “Implicit Representation of Generalized Variable Upper Bounds in Linear Programming”, Mathematical Programming, Vol. 14, No. 1, pp. 11-20.

[146]. Schrage, L. and L. Wolsey (1985), “Sensitivity Analysis for Branch-and-bound Integer Programming”, Operations Research, Vol. 33, No. 5 (Sept., Oct.), pp. 1008-1023.

[147]. Schrijver, A. (1986), Theory of Linear and Integer Programming, John Wiley & Sons, Ltd.

[148]. Schuster, E.W. and S.J. Allen (1998), “Raw Material Management at Welch's, Inc.” Interfaces, vol. 28, no. 5, pp. 13-24.

[149]. Serafini, P. (1996), “Scheduling Jobs on Several Machines with the Job Splitting Property”, Operations Research, Vol. 44, No. 4, (July-August), pp. 617-628.

[150]. Sexton, T.R., S. Sleeper, and R. E. Taggart, Jr. (1994), “Improving Pupil Transportation in North Carolina”, Interfaces, Vol. 24, No. 1 (Jan.-Feb.), pp. 87-103.

[151]. Sharpe, W. F. (1963), “A Simplified Model for Portfolio Analysis”, Management Science, Vol. 9 (Jan.), pp. 277-293.

[152]. Sherali, H., and W. Adams. (1999), A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, Kluwer Academic Publishers, Dordrecht, The Netherlands.

[153]. Sherbrooke, C.C. (1992), Optimal Inventory Modeling of Systems, Multi-echelon Techniques, John Wiley & Sons, New York, NY.

[154]. Sherman, H. D., and G. Ladino (1995), “Managing Bank Productivity Using Data Envelopment Analysis (DEA)”, Interfaces, Vol. 25, No. 2 (March-April), pp. 60-73.

[155]. Shlifer, E. and Y. Vardi (1975), "An Airline Overbooking Policy," Transportation Science, Vol. 9, No. 2 (May), pp. 101-114.

[156]. Srinivasan, V. (1976), “Linear Programming Computational Procedures for Ordinal Regression”, Journal of ACM, Vol. 23, No. 3 (July), pp. 475-487.

[157]. Steinberg, L. (1961), “The Backboard Wiring Problem: A Placement Algorithm”, SIAM Review, Vol. 3, pp. 37-50.

[158]. Stern, G. and R. Blumenstein (1996), “GM Expands Plan to Speed Cars to Buyers”, Wall Street Journal, 21 October, p. A3.

[159]. Stigler, G. (1963), “United States vs. Loew's, Inc: A Note on Block Booking”, Supreme Court Review, p. 152.

[160]. Stigler, G. J (1945), “The Cost of Subsistence”, Journal of Farm Economics, Vol. 27, No. 2 (May), pp. 303-314.

[161]. Stone, J.C. (1988), “Formulation and Solution of Economic Equilibrium Problems”, Tech. Report. SOL. 88-7. Stanford University.

[162]. Strevell, M. and P. Chong (1985), “Gambling on Vacation”, Interfaces, Vol. 15, No. 2 (March-April), pp. 63-67.

[163]. Stroup, J.S., and R.D. Wollmer (1992), “A Fuel Management Model for the Airline Industry”, Operations Research, Vol. 40, No. 2 (March-April), pp. 229-237.

[164]. Subramanian, R.A., R.P. Scheff, J.D. Quillinan, D.S. Wiper, and R.E. Marsten (1994), “Coldstart: Fleet Assignment at Delta Air Lines”, Interfaces, Vol. 24, No. 1 (Jan.-Feb.), pp. 104-120.

[165]. Sze, D. Y. (1984), "A Queueing Model for Telephone Operator Staffing," Operations Research, Vol. 32, No. 2 (March-April), pp. 229-249.

[166]. Thompson, R. G., F. D. Singleton, R. M. Thrall and B. A. Smith (1986), “Comparative Site Evaluations for Locating a High-Energy Physics Lab in Texas”, Interfaces, Vol. 16, pp. 35-49.

[167]. Tomlin, J. and J. S. Welch (1985), “Integration of a Primal Simplex Algorithm with a Large Scale Mathematical Programming System”, ACM Trans. Math. Software, Vol. 11, pp. 1-11.

[168]. Troutt, M.D (1985), “Spying on the Cost Structure of Naive Bidding Competitors via Linear Programming Models”, Operations Research Letters, Vol. 4, No. 4, pp. 181-184.

[169]. Truemper, K. (1976), “An Efficient Scaling Procedure for Gains Networks”, Networks, Vol. 6, pp. 151-160.

[170]. Vickrey, W. (1961), “Counterspeculation, Auctions, and Competitive Sealed Tenders”, Journal of Finance, Vol. 16, No. 1 (March), pp. 8-37.

[171]. Wagner, H. M. and T. M. Whitin (1958), “Dynamic Version of the Economic Lot-Size Model”, Management Science, Vol. 5, No. 1, pp. 89-96.

[172]. Wall Street Journal, “UAL's United Alters Schedule, Cuts Costs, Boosts Flights in Face of Discount Fares”, (19 June, 1978), p. 8.

[173]. Wang, K.C.P. and J.P. Zaniewski (1996), “20/30 Hindsight: The New Pavement Optimization in the Arizona State Highway Network”, Interfaces, Vol. 26, No. 3 (May-June), pp. 77-89.

[174]. Warner, D. M. (1976), “Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Programming Approach”, Operations Research, Vol. 24, No. 5 (September-October), pp. 842-856.

[175]. Weingartner, H. M. (1972), “Municipal Bond Coupon Schedules with Limitations on the Number of Coupons”, Management Science, Vol. 19, No. 4 (Dec.), pp. 369-378.

[176]. What's Best User Manual. LINDO Systems, Chicago, (1998).

[177]. Whitt, W. (1993), “Approximations for the GI/G/m Queue”, Production and Operations Management, vol. 2. no. 2, pp. 114-161.

[178]. Wolsey, L. (1998), “Integer Programming”, Wiley Interscience, New York.