研究ノート

「ダイナミック・プログラミング」の方法について(1)

井　原　健　雄

I はじめに

本稿では,「線形計画」のもつ限界を克服する有力な数理計画法としての「ダイナミック・プログラミング」(Dynamic Programming) を考察する。すなわち，列挙的な最適化の手法を吟味することが，本稿の目的である。

ダイナミック・プログラミングは，線形計画の場合と同様に，電子計算機の発達にともなって完成された，大型コンピュータに基礎をおく計算手続である。線形計画にまさる，その相対的利点は，２つある。

i）最適値の発見に際して，実行可能領域の凸性を必ずしも前提としない。

ii）最適値の発見に際して，変数の連続性を必ずしも前提としない。

これら２つの利点は，選択変数の数が多くなるのに応じてその計算手続が一層複雑になるというダイナミック・プログラミングの欠点を補なってあまりあり，その結果，数多くの潜在的な問題に広く適用することが可能となる。(2)

ダイナミック・プログラミングは，線形計画のようなアルゴリズムであるというよりは，むしろ最適化へのアプローチを意味するものである。したがって，システムの設計者や企業の経営者は，過去に作成したダイナミック・プログラミングの手続を，新しい問題に対して利用できないのが普通である。すなわち，新しい問題に対しては，つねに新しい手続の作成が必要である。以下において，われわれは，ダイナミック・プログラミングに

---

(1) 本稿は，R. de Neufville & J. H. Stafford, "Systems Analysis for Engineers and Managers", McGraw-Hill, 1971の第５章 Dynamic Programming を教材として，本学経済学部学生の尾崎功基，竹中淳二，湯尾信弘の３名と講読・討議した研究成果である。

(2) Wagner, H. M., "Principles of Operations Research," Prentice-Hall, 1969, および White, D.J., "Dynamic Programming," Oliver & Boyd, 1969, 参照。

よる定式化の理解と，その解釈について，とくに注意を払うことにする。

II ダイナミック・プログラミングの概要

まず，最初に，当該変数の連続性を仮定する方法[3]は，とくに大規模システムの最適化にとって不適当であるという理由を確認することが重要である。その意味で，伝統的な微分等による分析方式は，必ずしも有効なものとはなりえない。換言すれば，伝統的な分析方式のもつ有効範囲を見定めることが，ダイナミック・プログラミングによる改善の評価を可能とする。

（伝統的な分析方式の限界）

伝統的な分析方式が有効となりうる問題の数は，きわめて限られている。事実，その適用の有効範囲は，重力や磁力，圧力のような，もっぱら力の場を含む問題に限定されている。その理由として，伝統的な分析方式は，変数が連続であり，そのすべての点で連続な導関数をもつという，強い仮定を前提としているからに他ならない。いうまでもなく，この仮定は，現実の問題においては，通常みたされていない。たとえば，物質の輸送やパイプラインのようなネットワーク・システムの問題，さらにまた，独立した大型プロジェクトの決定問題等に対しては，変数の連続性を仮定する伝統的な分析方式を用いて，その問題を有効に定式化することはできないのである。

また，伝統的な分析方式による最適化の手法は，きわめて限られた単純な問題状況に適用できるだけである。すなわち，その場合，まず，最適条件を定義し，さらに，その方程式を導出し，そして最後に，その解を求めることが必要である。しかしながら，最適条件を定義する微分方程式が，必ずしも明示的な解をもちえないことに留意すべきである。

さらに，伝統的な分析方式による最適化の手法は，いわゆる2階の条件がみたされなければ，極大値と極小値とを区別できない，ということにも留意すべきである。また，その手法では，もしも与えられた問題の最適値が，つぎの図1で示す変域の境界で与えられるような場合には，不都合が生ずる。すなわち，伝統的な微分による最適条件は，局所的な極大値を区別し得ないばかりでなく，また実行可能領域の境界で生ずる最適値を，必ずしも識別し得ないのである。

つぎに，伝統的な分析方式では，最適化の条件を定式化することすらできない場合もあ

（3） たとえば，線形計画法（Linear Programming）は，これにあたる。

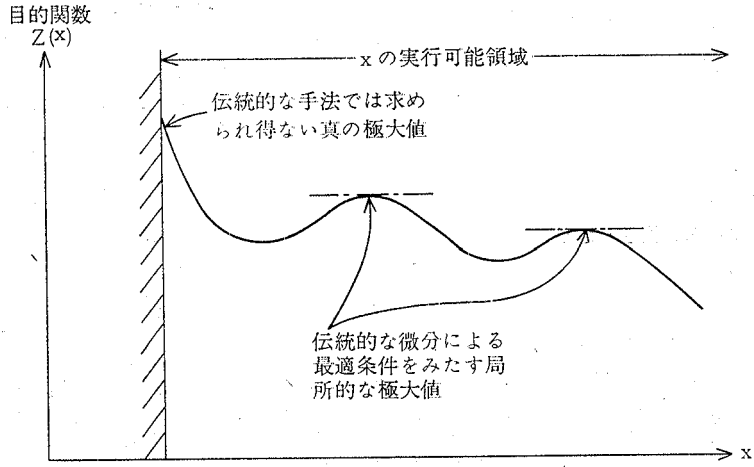

図1　伝統的な分析方式の弱点を示す仮設例

る。たとえば，変数の連続性がみたされず，したがってその導関数が少なくとも部分的に定義できない場合には，与えられた問題の関数が，十分に微分可能であるというわけにはいかなくなる。このような場合の例が，つぎの図2に示されている。

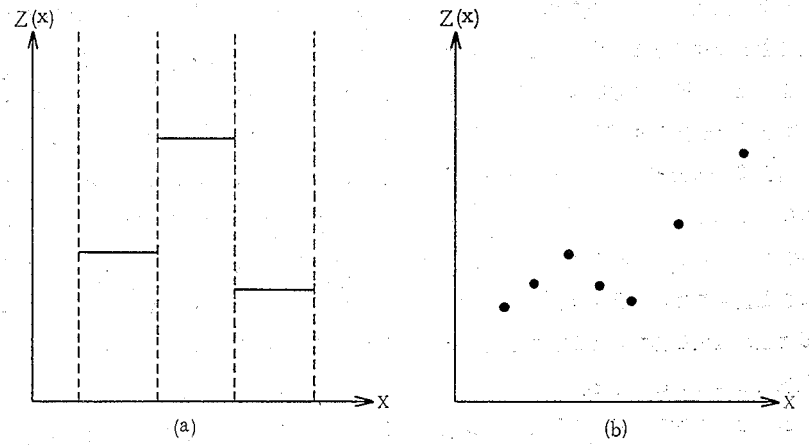

図2　最適化の条件が定式化され得ない場合
（a）微分不可能な関数　（b）離散的な集合

微分による最適化の基準は，目的関係が区分的に線形であるような場合には，その最適値を定義することさえ不可能となる。(4) さらにまた，微分方程式が，差分方程式として書き改められるような場合(5)には，その最適基準にもとづく解が，きわめて不安定なものとなりうるかもしれない。(6)

要約すれば，伝統的な分析方式による最適化の手法は，一般のシステム分析に対して，きわめて限定された有効性しかもち得ないということである。なぜなら，その手法に従がうかぎり，変数値の変化に伴なう解の感応度を定義する，実践的な手段を得ることができないからである。(7)

(列挙による最適化の手法)

叙上の問題は，計算上の煩雑さを伴なうとはいえ，理論上，その解を逐一列挙し，かつ吟味することによって解明することができる。その場合，目的関数に含まれる変数について，そのすべての可能な組み合わせの解を規則的に吟味するためには，もとより最適値を定義しておく必要がある。かかる理論上のアイデアを礎石として，しかも能率よく，最適解を見い出す要請が，列挙による最適化の手法，とくにダイナミック・プログラミングの開発を促がしたわけである。

しかし，すべての可能な解の組み合わせを逐一列挙し，かつ吟味する費用は，通常，厖大なものとなる。たとえば，10個の変数が，それぞれ10個の異なる値をとるものと想定した場合の問題について，考察してみよう。この場合，可能な解の組み合わせの数は，全部で10^{10}通りとなる。そこで，このすべてを逐一列挙し，かつ吟味するとすれば，たとえ大型コンピュータを利用するとしても，かなり多くの時間と費用がかかることになろう。(8)

そこで，この煩雑さを回避し，しかも一般的な問題を解明するのに役立つ列挙の仕方

---

(4) たとえば，線形計画においては，関数が連続でなく，したがってその導関数が定義できない結果，最適値は，つねにその関数の境界線上にあることを想起せよ。

(5) 電子計算機を利用する場合，この操作は必要不可欠なものとなる。

(6) この場合，任意の選択変数X_iに対する目的関数Zの変化率（すなわち$\triangle Z/\triangle X_i$の値）が，最適値の近傍できわめて小さい値をとる結果，この微分は，Zの評価の際に生ずる「丸めの誤差」(Round-off Errors) の影響を著しく受けるものとなる。

(7) これに対比して，線形計画，および，ダイナミック・プログラミングでは，その解の1部として，感応度分析を与えることになっている。

(8) システム分析のもつ現実的な意味を考えれば，10^{10}通りという組み合わせの数自体は，けっして異常なものではない。むしろ問題は，列挙によってその組み合わせの内容を求めることに，多くの時間がかかるということにある。

が，有効な列挙技術の開発を促がす結果となったのである。すなわち，その場合，与えられた問題の構造について，ある仮定[9]が導入され，その仮定にもとづいて，支配的（すなわち，より有効）な解の組み合わせを要素とする解の部分集合が，通常，定義される。したがって，われわれは，かかる解の部分集合の各要素についてのみ，考察を限定しさえすればよいこと[10]になり，その結果，時間と費用の大幅な節約が可能となるわけである。[11]

III　ダイナミック・プログラミングによる定式化

ダイナミック・プログラミングは，コンピュータにもとづく列挙技術であり，それは，目的関数と制約式が非線形で，しかもその制約式によって定義される実行可能領域が凸状でないような問題に対しても，適用可能である。これまでのところ，ダイナミック・プログラミングは，異時間にわたる幾つかの段階によって示されるような問題の多くに適用されてきたが，もとよりその適用領域は，かかる連続的な問題だけにとどまるものではない。すなわち，非連続的な問題に対しても，十分に適用可能である。とくに注意すべきこととして，ダイナミック・プログラミングは，本来，時間と関係する必要が必ずしもない，という点が指摘される。[12]しかしながら，時間的な意味をもつ通常の言葉でダイナミック・プログラミングの内容に言及することは，文献を読む者にとっては，好都合となるであろう。

（数学的表現）

ダイナミック・プログラミングは，目的関数

$$Z(X)=\max\ G(X_1,\ X_2,\ \cdots,\ X_N)$$

を，少数の変数だけに依存する幾つかの段階に分解して，その段階ごとに最適化を試みる

（9）　たとえば，加法性の仮定が指摘される。

（10）　この部分集合の要素についての検討が，ダイナミック・プログラミングでは，余す所なく試みられるのに対比して，“Branch-and-bound”の方法では，部分的に試みられる。

（11）　すべての数理計画法は，段階ごとにその最適解へ接近していくという意味で，列挙による最適化の手法である，ともいえる。しかし，本稿では，それを文字通りの意味（すなわち，狭義）に解釈して用いていることに留意すべきである。

（12）　時間（より正確には，時刻）は，1つの代数的な変数として，特別な性質をもつものではない。

ものである。[13] そこで，この目的関数の分解を記述しようとすれば，収益関数 $g_i(X_i)$ という概念の導入が必要となる。収益関数とは，個々の変数 X_i によって引き起こされる $G(X_1, X_2, \cdots, X_N)$ の変化を表わしている。その場合，分解とは，目的関数について，すべての変数を同時に変化させてしまうのではなく，1度にはごく少数の変数だけを変化させることを意味している。したがって，もしも1つの変数だけで各段階が定義できるような場合には，もっとも有効な解法手順を得ることになる。いま，変数 X_1 だけに注目し，その値をかえた場合の目的関数を，つぎのように書くことができる。

$$Z(X)=\max_{X_1}\left[g_1(X_1), \max_{X_2\to X_N} G'(X_2, X_3, \cdots, X_N)\right] \qquad (1)$$

また，X_1 を除く他の変数についても，その各々によって，上式右辺の $G'(X_2, X_3, \cdots, X_N)$ を分解する必要が生じよう。ただし，上式(1)の表示法，すなわち，$\max_{X_i}[A, B]$ は，A と B のすべての可能な組み合わせに対して，X_i の関数として，関数 $[A, B]$ の最大値をもとめることを表わしている。したがって，目的関数 $Z(X)$ を，(1)式の形式に分解していくことは，最適化問題を，同時に全変数を考慮する1つの大きな組み合わせの問題から，各段階ごとにごく少数の変数だけを考慮する多数の小さな問題に変換することを意味する。その場合，組み合わせ問題の規模は，同時に考慮する変数の数が多くなれば，一層大きくなる結果，上述した目的関数の分解ができる場合には，計算上の労力を著しく減少させることになるのである。では，どこまで問題を分解することができるか，ということは，$Z(X)$ の数学的構造と，変数 X_i にかかる制約式の数とに依存する。たとえば，ある特定の段階において，変数にかかる制約が非負である，すなわち，$X_i \geqq 0$ であるとか，あるいはまた，変数 X_i のとる値が，ある所与の範囲に限られている，すなわち $R \leqq X_i \leqq S$ で表わされるような場合，その制約は，他の段階には影響を及ぼさない。しかし，一般に，各段階を表わそうとすれば，各変数にかかる制約の数と同じだけ多くの変数が必要となる。その結果，変数にかかる制約の存在は，ダイナミック・プログラミングの次数を著しく増大させ，その計算能力を低下させるのである。

目的関数が分解できる条件は，2つある。その条件とは，分離可能性（Separability）と単調性（Monotonicity）である。いま，各変数 X_i に対する収益関数 $g_i(X_i)$ が，他の

(13) ここで言う段階とは，連続的な問題（たとえば，月ロケットの飛行計画）については，時間，空間，あるいは，その他の座標系における特定の点などを表わし，非連続的な問題（たとえば，静学的な資源配分問題）については，当該資源の配分を可能とする特定の活動などを表わすことになる。

変数 X_j（ただし，$j \neq i$）の値とは独立であるとき，その目的関数は，分離可能であるという。また，$g_i(X_i') \geqq g_i(X_i'')$ をみたすすべての場合について，つぎの不等式

$$[g_i(X_i'),\ G'(X_2,\ X_3, \cdots,\ X_N)] \geqq [g_i(X''),\ G'(X_2,\ X_3, \cdots,\ X_N)]$$

が成立するとき，その目的関数は，単調であるという。したがって，目的関数が分離可能であり，しかも，単調であれば，その目的関数は，（1）式のように分解可能である。

なかでも，分離可能性の条件は，実際的な意味において，とくに重要である。なぜなら，それは，多数の可能解を消去して最適解を見い出すのに役立つからである。したがって，どのような仮定を設けるかが，ダイナミック・プログラミングにとって，殊の外，重要となる。(14)

加法的な目的関数，および乗法的な目的関数は，いずれも分解可能である。すなわち，加法的な目的関数，

$$Z(X) = \sum_{i=1}^{N} g_i(X_i) \qquad (2)$$

は，変数 X_i の実数値に対して，つねに分解可能である。しかし，乗法的な目的関数，

$$Z(X) = [g_1(X_1)][g_2(X_2)] \cdots [g_N(X_N)] \qquad (3)$$

は，変数 X_i が実数で，しかも非負（すなわち，$X_i \geqq 0$）であるときに限り，分離可能となるのである。(15)

（解法戦略）

最適化の過程とは，変数 X_i について種々の特定値 x_i をあてはめることにより，その最良な集合を決定することである。このとき，x_i の値を要素とする特定の集合を，ひとつの「政策」とよぶこともできよう。その意味で，ダイナミック・プログラミングの目的は，最適政策（$x_1^*,\ x_2^*, \cdots,\ x_N^*$）をもとめることである，ということができる。

その解法戦略は，各変数が，他のいかなる変数の影響からも独立している，という仮定のもとで，導出される。この独立性の仮定が意味することは，最適政策（$x_1^*,\ x_2^*, \cdots,\ x_N^*$）のいかなる部分集合（$x_1^*,\ x_2^*, \cdots,\ x_M^*$），$M \leqq N$ といえども，それ自体が，また

(14)　しかしながら，各段階やプロジェクトの独立性を仮定することは，多くの興味ある問題に対してダイナミック・プログラミングの適用を排除することになる，という犠牲を払っていることにも注意すべきである。

(15)　Nemhauser, G., "Introduction to Dynamic Programming," John Wiley & Sons, 1966, 参照。

K 単位の資源（$K=\sum_{i}^{M} x_i^*$）を最初の M 段階に配分するのにとって，最適でなければならない，ということである。これは，R. E. Bellman の最適性の原理[16]として知られているように，それ以後に続く各変数の最適値，すなわち，x_{M+1}^*, x_{M+2}^*, $\cdots x_N^*$ も，また同様な理由で，それ以前のすべての段階について最適でなければならない。[17] この分離可能性の条件は，その論理的結果として，ダイナミック・プログラミングによる解法の，いわば核心となる「再帰式」（Recurrence Formula）を直接もたらすことになるのである。[18]

ダイナミック・プログラミングの再帰式は，ある段階までの最適値が与えられたものとして，それ以降の各段階での最適値を与えるものである。この点を明らかにするため，M 個の段階の集合に対する累積的な収益関数を，つぎのように定義しよう。

$$f_M(K)=\max Z(x_1,\ x_2,\cdots,\ x_M),\quad M\leqq N$$

ただし，制約式は加法的で，しかも，その上限が $\bar{X}$ に等しいという条件のもとで，$K=\sum_{1}^{M} x_i \leqq \bar{X}$ をみたすものとする。つまり，収益関数 $f_M(K)$ は，K 単位の資源を，最初の M 個の段階に配分することによって得られうる最大値を与えるものであることがわかる。それゆえ，再帰式は，K のありとあらゆるすべての値について考慮された $f_M(K)$ にもとづいて，つぎの $f_{M+1}(K)$ をもとめる手順を提供するものである。その結果，ダイナミック・プログラミングでは，この再帰式を各段階ごとに N 回適用することによって，その最適値を容易にもとめることができるのである。

つぎに，叙上の再帰式が，ダイナミック・プログラミングのなかで，どのように使われるのかを明らかにしておこう。そのために，ただ1つの制約式のもとで，加法的な目的関数をもつ問題を想定する。いま，変数 X_i の特定値を x_i で表わせば，この問題を，つぎのように書き表わすことができる。

$$\left.\begin{array}{l}\text{制約条件 } \Sigma x_i \leqq \bar{X} \text{ のもとで，目的関数}\\ Z(X)=\Sigma g_i(x_i) \text{ を最大化せよ。}\end{array}\right\} \qquad (4)$$

ただし，$\bar{X}$ は，利用可能な資源の総量を表わすものとする。

---

(16) Bellman, R., "Dynamic Programming," Princeton University Press, 1957, および Bellman, R., and S. Dreyfus, "Applied Dynamic Programming," Princeton University Press, 1962, 参照。

(17) Bellman の最適性の原理とは，要するに「最適政策の部分系列も，また最適政策になっている」ということである。

(18) ある所与の目的関数についての再帰式は，一連の増分による最適配分をもたらし，それが，結果として，全体的な最適政策をもたらすことになるのである。

この場合，再帰式は，つぎのようになる。(19)

$$f_M(K)=\max\{g_M(x_M)+f_{M-1}(K-x_M)\} \qquad (5)$$

ただし，$0\leqq x_M\leqq K$，$K\leqq\overline{X}$ をみたすものとする。

すなわち，K 単位の資源を M 個の段階に配分する最良な方法は，K のある部分，すなわち，x_M，を M 番目の段階に配分し，その残りの部分，すなわち，$(K-x_M)$ をそれ以前の（$M-1$）個の段階に配分する，すべての可能な方法を吟味することによってもとめられる。ここで，L を，われわれが考察する K についての水準の数とするとき，(20) これは，L^2 個の可能性をもつ単純な組み合わせの問題に帰着する。そして，この個数は，コンピュータの能力から判断して，さほど大きい数ではなく，したがって，$f_M(K)$ の値を，K についてのすべての L 個の水準に対して，容易にもとめることができるのである。

ダイナミック・プログラミングによる最適化の手順は，叙上の（5）式を用いることから始められる。まず最初に，配分される資源が存在しない場合には，累積的な収益関数の値をゼロと仮定することが，つねに可能である。すなわち，M 個のすべての段階に対して，

$$f_M(0)=0$$

と表わされる。ここで留意すべきは，基準線と，各変数に対する収益関数 $g_i(X_i)$ を，新たにゼロと定義しているということである。(21) これと同様に，利用可能な K 単位の資源がいかなる段階にも配分されていない場合には，その累積的な収益関数の値が，ゼロとなる。すなわち，

$$f_0(K)=0$$

と表わされる。したがって，われわれは，第1段階の累積的な収益関数 $f_1(K)$ を，1番目の変数に対する収益関数，$g_1(X_1)$，から直接的にもとめることができる。これを具体的に示せば，

$$f_1(K)=g_1(K) \qquad (6)$$

---

(19)　目的関数は，（2）式と同様に，加法的であることに注意せよ。

(20)　各段階に配分されるべき K 単位の資源が，いかなる単位の部分にまで分割可能であるかを示している。

(21)　すなわち，（5）式において，$K=0$ とおけば，その再帰式は，$f_M(0)=\max\{g_M(0)+f_{M-1}(0)\}$，$f_{M-1}(0)=\max\{g_{M-1}(0)+f_{M-2}(0)\}$，…，$f_1(0)=\max\{g_1(0)+f_0(0)\}$ となっており，したがって，$f_0(0)=0$，および，$g_i(X_i)=g_i(0)=0$（ただし，$i=1$，2，…，M）を，前以って与えておく必要があるということである。

となる。ただし，$g_1(K)$ は，単調であるものとする。[22] なお，各変数に対する収益関数，$g_i(X_i)$，は，相互に独立である結果，そのうち，いずれの変数を最初に選ぶかは，任意であることに，留意すべきである。その意味で，叙上の（6）式は，利用可能な K 単位の資源を第1段階に配分することによって得られる最大収益が，定義によって，第1段階の変数に対する収益関数と同じであることを示しているのに過ぎない。

ダイナミック・プログラミングの再帰式は，ひとたび用いられると，当該問題の最適値を決定するべく，N 個の段階の各々について，それが直接適用される。この方法による解法は，直接的な列挙による方法と比較して，最適値をもとめるのに要する計算を著しく削減することができる。[23] すなわち，直接的な列挙による方法に従がえば，考慮する必要のある組み合わせの総数は，L^N であるのに対して，ダイナミック・プログラミングによれば，その各段階がただ1つの変数によって記述されるかぎり，その数は，$L^2\times N$ となるのである。

ダイナミック・プログラミングによる解法が有するこの計算上の効率は，すでに指摘したように，変数にかかる制約式の数が増加するのに応じて急速に低下する。各段階での収益関数を記述するためには，制約式の数と同じだけ多くの変数，（これを V としよう）が存在しなければならない。そこで，ダイナミック・プログラミングによって吟味される組み合わせの数は，近似的に，$L^V\times N$ となるのである。これより明らかなように，通常，L と N は，ほぼ同じ大きさであると考えられるから，たとえ用いられる制約式の数が2つであったとしても，それに要する計算上の手数は，著しく増大することになる。したがって，実際に，ダイナミック・プログラミングを，2つ以上の制約式がある問題に適用している実例は，きわめて少ないのである。

最後に，ダイナミック・プログラミングに固有の感応度分析は，その方法の有するもっとも魅力的な特徴である点に，注意を払う必要がある。最適化の過程は，それ自体，相対的に少ない資源と，相対的に少ない活動，[24] すなわち，$f_M(K)$，$K\leqq\overline{X}$，と $f_{M-1}(K)$，との計算を必要としている。しかしながら，それは，線形計画の場合のように，変数の変化

(22) 前節で与えた，目的関数が単調であるための条件を想起せよ。
(23) 実際に，計算時間がどれほど削減できるかについては，Bellman, R., "Adaptive Control Processes : A Guided Tour," Princeton University Press, 1961, および Bellman, R., and S. Dreyfus, *op. cit.*, 1962, 参照。
(24) この活動（Activities）は，段階（Stages）に対応するものである。

を，1度に1つずつ分析するものではない。むしろ，ここでは，利用可能な投入量を増加させた場合の効果を，再帰式の簡単な適用によって，容易にもとめることができるのである。

（例題による解法）

つぎに，ダイナミック・プログラミングがどのように用いられるかを，簡単な例題に即して考察してみよう。その結果をみることにより，最適でない解がどのようにして初期の段階で認識され，れそ以後の考察の対象から取り除かれるかを明らかにすることができる。

いま，3つの異なる川の流域にダムを建設することによって，利用可能となる電力 $Z(X)$ を最大にすることを想定しよう。いうまでもなく，川の流域は，それぞれ独立と考えられるから，この問題を，各々の流域に1つずつ，合計3つの段階に分解できる。したがって，目的関数は，つぎのようになる。

$$Z(X)=g_1(x_1)+g_2(x_2)+g_3(x_3)$$

また，利用可能な総予算が3単位であるものと仮定すれば，当該問題の予算制約式は，つぎのようになる。

$$x_1+x_2+x_3\leqq 3=\overline{X}$$

さらに，簡単化のため，投資の実行可能な水準 L は，0，1，2，または3単位のうちのいずれかであるものと仮定しよう。[25] また，このときの収益関数（すなわち，ダムの建設によって利用可能となる電力は，表1，および図3によって与えられているものとしよう。いうまでもなく，これらの関数は，ダムを建設する谷の地形が，通常，異なっているため，一様に凸状，あるいは凹状である，ということにはならないのである。

(25)　この仮定は，ダムを建設しようとすれば，ある程度のまとまった資金が必要であるという経験的事実にもとづいている。なお，より厳密に述べれば，L は，実行可能な投資水準の数を表わすことから，0，1，2，または3単位をその内容とする4通りである，と言うことになる。

収益関数	段　階（流域）		
	$i=1$	$i=2$	$i=3$
$g_i(0)$	0	0	0
$g_i(1)$	2	1	3
$g_i(2)$	4	5	5
$g_i(3)$	6	6	6

表1　各流域を段階とみなした場合の異なる投資水準（0，1，2または3）に対する独立した収益関数

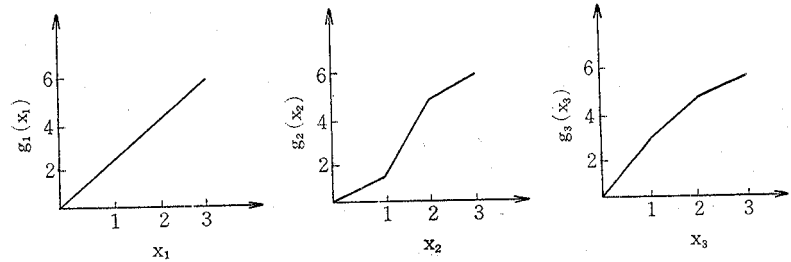

図3　各流域におけるダム建設のため，異なった水準の投資を行なった場合の収益関数 $g_i(x_i)$

さて，前節で明らかにしたように，いかなる活動[26]に対しても資源を配分しないかぎり収益は生じ得ないから，すべての X_i に対して $f_i(0)=0$ となる。また，利用可能な資源についての異なる水準を最初の段階（すなわち，第1ダムの建設）に配分すれば，

$$f_1(K)=g_1(K)$$

となり，したがって，これは，

$$f_1(0)=0,\ f_1(1)=2,\ f_1(2)=4,\ f_1(3)=6$$

を意味することになる。

つぎに，資源の異なる水準を，2つの段階（すなわち，第1ダムの建設と第2ダムの建

(26)　すなわち，各流域におけるダム建設を意味する。

設の双方）に対してどのように配分できるかを考えてみよう。この場合の再帰式は，つぎのようになる。

$$f_2(K)=\max_{0\leq x_2\leq K}\{g_2(x_2)+f_1(K-x_2)\}$$

これを具体的に表わせば，

$$f_2(0)=\max\{g_2(0)+f_1(0)\}=0$$

および，

$$f_2(1)=\max\{\{g_2(0)+f_1(1)\}\text{ または }\{g_2(1)+f_1(0)\}$$
$$=\max\{(0+2)\text{ または }(1+0)\}=2$$

となる。なぜなら，2つの段階に対して1単位の資源を配分する方法は，2つあるからである。さらに，表1によって容易に確かめられるように，以下同様の計算にもとづいて，

$$f_2(2)=\max\{\{g_2(0)+f_1(2)\}\text{ または }\{g_2(1)+f_1(1)\}$$
$$\text{または }\{g_2(2)+f_1(0)\}\}$$
$$=\max\{(0+4)\text{ または }(1+2)\text{ または }(5+0)\}$$
$$=5$$
$$f_2(3)=\max\{\{g_2(0)+f_1(3)\}\text{ または }\{g_2(1)+f_1(2)\}$$
$$\text{または }\{g_2(2)+f_1(1)\}\text{ または }\{g_2(3)+f_1(0)\}\}$$
$$=\max\{(0+6)\text{ または }(1+4)\text{ または }(5+2)$$
$$\text{または }(6+0)\}$$
$$=7$$

を得る。

また，$f_3(K)$ についても，同様な仕方で，容易にもとめることができる。[27] その結果のみをまとめて記せば，つぎのようになる。

$$f_3(0)=\max(0+0)=0$$
$$f_3(1)=\max\{(0+2)\text{ または }(3+0)\}=3$$
$$f_3(2)=\max\{(0+5)\text{ または }(3+2)\text{ または }(5+0)\}$$
$$=5$$
$$f_3(3)=\max\{(0+7)\text{ または }(3+5)\text{ または }(5+2)$$
$$\text{または }(6+0)\}$$

---

(27)　この場合，再帰式は，$f_3(K)=\max_{0\leq x_3\leq K}\{g_3(x_3)+f_2(K-x_3)\}$ として表わされる。

$= 8$

以上の結果より，利用可能な3単位の資源を3つの段階に対して，配分することによって得られる最大収益は，8であることが判明する。

つぎに，資源のどのような配分方法が，この最大収益8をもたらしているかをみるために，与えられた問題の解法手順をふりかえってみることにしよう。(28)

まず，最初に，$f_3(3)$ は，つぎのようにしてもとめられている。

$$f_3(3) = 8 = g_3(1) + f_2(2)$$

これより，1単位の投入が，X_3 に配分され，また，残りの2単位が，最初の2つの段階に配分されていることが明らかとなる。つぎに，

$$f_2(2) = 5 = g_2(2) + f_1(0)$$

であったことから，残りの2単位の資源は，すべて第2の段階に対して配分されていることが判明する。表2は，以上の結果を要約したものである。

ここで注目すべき点は，ダイナミック・プログラミングが，実行可能領域の凸性を必ずしもその前提条件とはしていない，ということである。(29) したがって，通常の限界分析，あるいは，線形計画では，つねに実行可能領域の凸性を前提としているので，それをここでの例題に適用したとすれば，当然間違った答えを導くことになるであろう。たとえば，図3の収益関数 $g_i(x_i)$，$(i = 1, 2, 3)$ に従がって，積み上げ方式による最高便益が得られるように，利用可能な資源を配分してみよう。(30) このとき，まず，最初の1単位の資源は，第3の段階に配分されることになる。さらに，つぎの1単位は，第1の段階，または第3の段階に配分される。(31) そして，最後の1単位は，もしも2番目の資源が第1の段階に配分された場合には，第1の段階，または第3の段階のいずれかその一方に配分され，もしも2番目の資源が第3の段階に配分された場合には，第1の段階に配分されることになる。したがって，この最終的な配分形態は，(1，0，2)，または (2，0，1) となる

---

(28) この作業を "Work back" という。したがって，ここでの課題は，最大収益をもたらした具体的径路を見つけ出すことにある。

(29) この点については，表1，および，図3によって確認される。

(30) これは，最小単位の投資水準から出発して，収益関数の勾配がもっとも大きいものを，すべての段階のなかから選び出し，逐次，その段階に資源を配分していく方式である。

(31) この場合，第1の段階への資源配分と第3の段階への資源配分は，同じ勾配（すなわち，2）をもっている結果，無差別となる。

が，これより得られる最大収益は，いずれも7となっている。[32] すなわち，この限界分析による計算結果は，ダイナミック・プログラミングによってわれわれがもとめた最大収益8とは異なっている点に留意すべきである。

項　　目	段　階（ダムの建設予定流域）			
	1	2	3	合　　計
各段階に対する最適投入 x_i^*	0	2	1	$\sum x_i^* = \overline{X} = 3$
最適投入にもとづく収益 $g_i(x_i^*)$	0	5	3	$Z(X^*) = 8$

表2　例題における最適資源配分

（解法の図解）

ダイナミック・プログラミングは，すべての可能な段階に対して，すべての可能な資源配分の量を考慮することにより，当該問題の最適な収益をもとめるものではあるが，しかしすべての可能な組み合わせの効果を計算しているわけではない。すなわち，各段階ごとに累積的な収益関数，$f_M(K)$，の値を求めることは，その配分について1つの径路を選び出し，その他の径路を放棄するという意味で，事実上，部分的な最適化を行なっているわけである。つぎの図4～図7は，各段階での部分的な最適化を例示したものである。

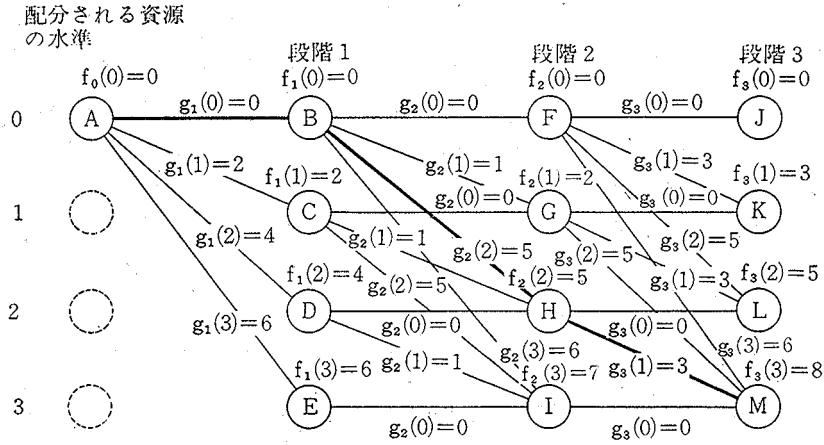

図4　資源配分のノードによる図式化

(32)　ここで，（1，0，2）は，1単位の資源を第1ダムの建設に充当し，2単位の資

まず，図4は，利用可能な0，1，2，および3単位の資源を，ダム建設という活動に配分する，すべての可能な仕方を，「ノード」表示[33]したものである。たとえば，その第2行は，1単位の投入がどのように使われているかを示している。また，いかなる段階に対しても，一切投入をしていないことを示すノードAから出発して，1単位の資源を2つの段階のうち，そのいずれか一方に配分することを示すノードGに至るまでの径路は，1単位の資源を2つの段階について配分する，そのすべての可能な組み合わせの仕方を表わしている。この場合，その径路として，ABGとACGの2つが描かれており，これらは，1単位の資源を第2段階と第1段階に，それぞれ配分することに対応している。このうち，より有効な配分の仕方は，$x_1=1$，$x_2=0$のACGであり，その累積的な収益関数の値は，$f_2(1)=2$として与えられた。[34] そこで，つぎに，2単位の資源を3つの活動にもっとも有効に配分する仕方を考えてみることにしよう。すなわち，これは，どのようにして図4のノードLに到達するかを考えてみることである。このとき，Gを通る径路は，ABGLとACGLの2通り存在する。しかしながら，そのうちCを通る径路は，Bを通る径路よりも望ましいことを，われわれは，すでに明らかにしている。[35] したがって，ABGLの径路は，第2の段階において考慮の対象から取り除かれ，それ以降において，再び取り上げられることはない。このようにして，吟味されるべき径路（すなわち，組み合わせ）の総数を減少させることができるのである。

つぎに，図5～図7は，吟味されるべき組み合わせの数が，どれほど減少されるかを示したものである。そのうち，図5は，各段階ごとに，吟味されるべき組み合わせの数が，どれほど大きくなっていくかを示している。すなわち，ある段階での組み合わせの数は，つぎの段階でさらに一層大きくなり，その結果，図5のような“Tree Diagram”を得るのである。したがって，ダイナミック・プログラミングの過程は，各段階で吟味されるべき余分の枝（すなわち，有効ではない組み合わせ）を切り落すことに他ならない。そこで，図6を用いて，1単位の資源を2つの段階に配分する場合の数を考えてみることにし

源を第3ダムの建設に充当することを意味する。また，（2，0，1）についても，同様な意味づけが可能である。

(33) これは，図4に示しているように，結節点（これをNodeという）による表示方式を意味する。

(34) $f_2(1)=\max\{[g_2(0)+f_1(1)]$ または $[g_2(1)+f_1(0)]\}$ としてもとめられたことを想起せよ。

(35) $[g_2(0)+f_1(1)]>[g_2(1)+f_1(0)]$ を意味する。

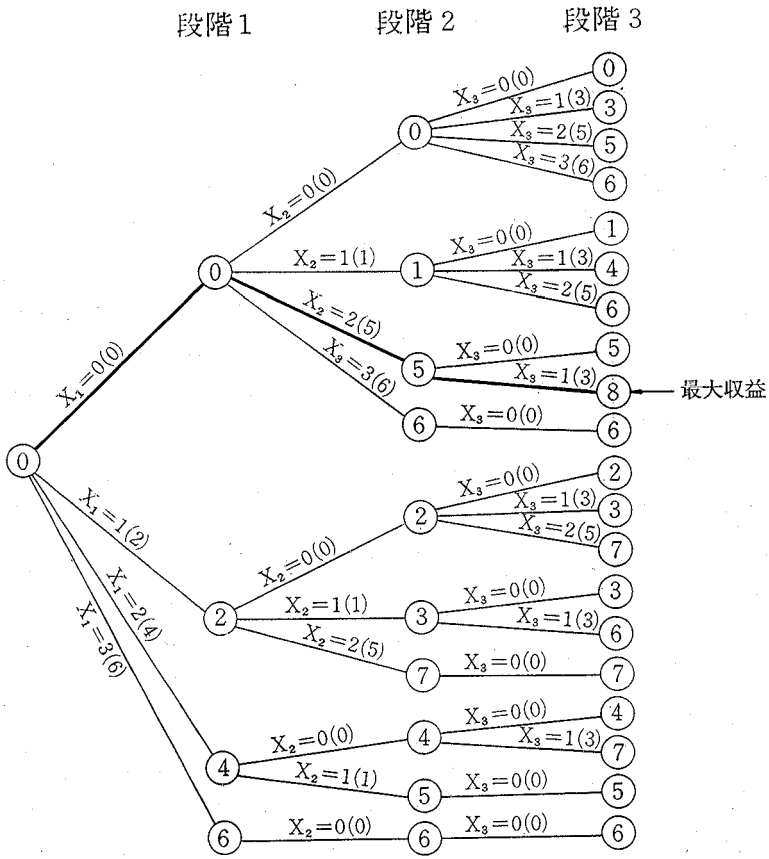

図5　資源配分の "Tree Diagram"

ただし，$X_1=1(2)$は，第1の段階に対する変数 X_1 の値を1単位としたとき，2の収益が得られることを意味する。また，⑤は，累積された収益が，5であることを意味する。

よう。この例題で，すでに示したように，利用可能な1単位の資源を第2の段階にのみ配分する仕方は，最適なものではなく，したがって，それにもとづく組み合わせの仕方をそれ以降の段階において考慮する必要はないのである。[36] さらに，2単位の資源を最初の2つ

(36)　図6の(a)，参照。

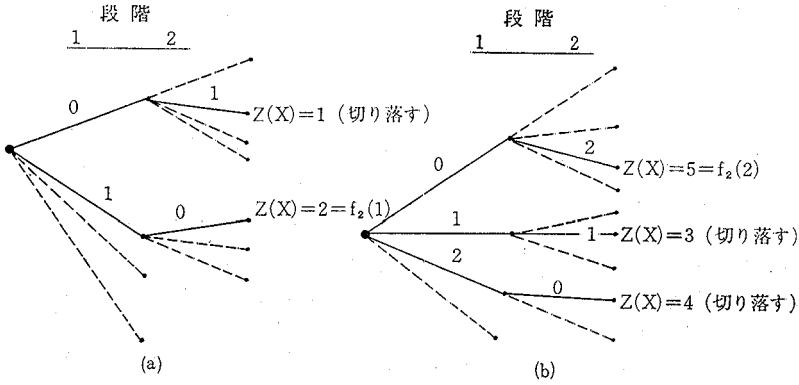

図6 最適でない枝（組み合わせ）の剪定
ただし，2つの段階に対して，（a）は，1単位の資源を，また（b）は，2単位の資源を，それぞれ配分する場合を表わしている。

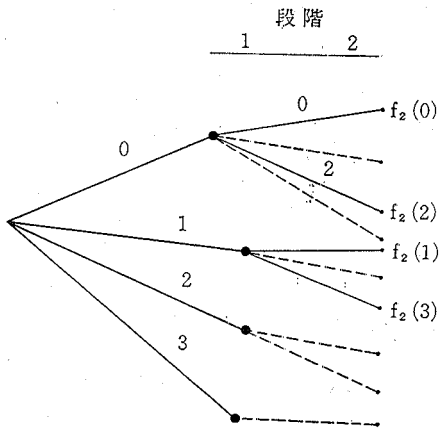

図7 各水準における資源の最適な組み合わせ
ただし，4つの組み合わせを除く他のすべてが，各段階ごとに剪定を受けている。

の段階に対して配分する仕方についても，同様に吟味され，余分の枝が切り落されることになるのである。[37] 最後に，図7は，各段階において，残されるべき枝（すなわち，組み合わせ）の数が，利用可能な資源の水準と同じ数になることを示している。そして，これは，各段階において，累積的な収益関数 $f_M(K)$ の値が，L 個存在するのと対応している。

IV　ダイナミック・プログラミングの応用

システム計画についてのすべての制約を，1つか2つの制約条件にまで減少させることは，一般には難しいので，どのような計画問題に対してもダイナミック・プログラミングを適用できるわけではない。したがって，それは，予算に対するプロジェクトの選択問題や，また，所与のシステムに対する操作と管理の決定問題を取り扱かうのに，もっぱら利用されてきた。そこで，つぎに，ダイナミック・プログラミングを適用することがふさわしいと思われる主要な問題群を，要約的に述べておくことにしよう。[38]

（不連続な問題群）

ダイナミック・プログラミングは，割り当てられるべき項目の数が少ない場合にかぎって，その配分問題を解くことができる。ただし，この場合，ある特定の連続性を意味しているわけではない。すでに，例題でみてきたように，そこでの段階は，単に当該項目が割り当てられるべきプロジェクト，ないしスロットを表わしているのにすぎない。そこで，このような配分問題に対して，ダイナミック・プログラミングを適用している典型的な事例として，われわれは，資本の予算化問題[39]や人員配置問題，[40] さらに，また各市場に対する

(37)　図6の(b)，参照。

(38)　広範囲にわたる適用例については，Aris, R., "The Optimal Design of Chemical Reactors," Academic Press, 1961, Bellman, R., *op. cit.*, 1961, Bellman, R., *op. cit.*, 1957, Bellman, R., and S. Dreyfus, *op. cit.*, 1962, Wagner, H. M., *op. cit.*, 1969, および White, D. J., *op. cit.*, 1969, 等を参照せよ。

(39)　R. de Neufville, and Y. Mori, "Optimal Highway Staging Through Dynamic Programming," ASCE Journal of Transportation Engineering, Vol. 2, 1970, および Gulbrandsen, O., "Optimal Priority Rating of Resource Allocation by Dynamic Programming—Application on Road Investments," Transportation Science, Vol. 1, 1967, 参照。

(40)　Wagner, H. M., *op. cit*, 1969, 参照。

物資の輸送問題[41]等をあげることができる。

配分問題は，加法的であることもあれば，また，乗法的であることもある。そのうち，乗法的な配分問題に対する再帰式は，

$$f_M(K)=[g_M(x_M)][f_{M-1}(K-x_M)]$$

となる。ただし，x_M は，$0 \leqq x_M \leqq K$，をみたすものとする。このような再帰式は，たとえば，われわれが成功の確率や，システムの信頼性を最大化しようとする場合には，つねに妥当するものとなるであろう。この場合，われわれは，すべての要素が同時に失敗するものとして定義される，失敗の機会を最小化するように努めているのである。したがって，上式における $g_i(x_i)$ の項は，投資水準 x_i が与えられたものとして，i 番目の要素が失敗する確率を表わしていることになる。

これらの配分問題は，すべて前方探究的（Forward-seeking）であるといえる。なぜなら，この場合，分析者は，まずもって彼がなにを持ち合わせているかということを知ることから手始め，そして，彼が達成し得る最良の状態を定義しようと望むからである。もとより，これとは逆の問題を提起することもできる。すなわち，この場合，分析者は，まずもって彼が達成しなければならない状態を知ることから手始め，そして，その目的を達成するべき最良の手段を知ろうと望むであろう。これらの後方探究的(Backward-seeking)な問題は，一見したところ，きわめて不自然に思われるかもしれないが，しかし，これらの問題は，叙上の前方探究的な問題と，数学的にはまったく同じものであり，しかもダイナミック・プログラミングを実際問題に適用したケースの大半が，この後方探究的な問題であることは，注目するに値いしよう。

われわれは，後方探究的な問題で，しかも不連続な問題の典型として，いわゆる「仕分け問題」[42]をあげることができる。この場合，分析者に対しては，最小の費用で，ある与えられた需要をみたすように，各種用役の施設を振り分けることが要請される。このような問題の実例として，たとえば，どのようなサイズの鋼材をつくって各工場に送るべきであるかといった問題や，あるいはまた，薄板に対する需要を最小の費用でみたすためには，どのような厚さの薄板を生産したらよいかといった問題が指摘される。

---

(41) Hillier, F. S., and G. J. Lieberman, "Introduction to Operations Research," Holden-Day, 1967, および White, D. J., *op. cit.*, 1969, 参照。

(42) "Assortment Problem" とよばれるものである。White, D. J., *op. cit.*, 1969, 参照。

（変数の再解釈）

ダイナミック・プログラミングの発展において，変数 X_i は，ある資源 $\overline{X}$ の与えられた数量 x_i が投資されるようなプロジェクトを表わすものとして解釈されてきた。このような変数の解釈は，当該プロセスを説明するのには便利であるが，しかし，ダイナミック・プログラミングが適用できるもっと一般的な状況の説明としては，著しく限定的なものである，といわざるを得ない。事実，配分されるべき資源が，一切存在しないような状況も多く，そのような場合，分析者の関心は，たとえば，ある物理的な位置といったような，特定の状態にとどまる費用や，また飛行機のスピードといったような，ある特定の属性をもつことの費用や，さらに，また N 単位のストックを持つ状態といったような，ある特定化された指標を持つことの費用に，もっぱら向けられている。

したがって，これまでにわれわれが用いてきた記号に対して，それがもっと一般的な意味をもつように，再解釈をすることが必要となる。とりわけ，$f_M(K)$ を，再帰過程の始まりから数えて，M 段階に至った時点で状態 K にあるときの最適値を表わすものとして，再解釈をほどこすことは有益である。ただし，再帰過程の始まりとは，当該問題が，前方探究的な問題として定式化されている場合には，その問題における最初の段階を意味するものとなり，また，それが，後方探究的な問題として定式化されている場合には，その問題における最後の段階を意味するものとなる。各々の段階における状態は，K によって特徴づけられ，そして，それは，一般に，資源配分問題でみたような，総合的な制約によって限定を受けるのではなく，むしろ，そのプロセスにおける制約によって限定を受けるのである。したがって，たとえば，いかなる段階においても，在庫の状態は，在庫量が「現在入庫されている数量」プラス「実際に生産された数量」を上廻ることはない，という事実によって限定を受けることになるのである。

（連続な問題群）

ある種の問題は，たとえば，ある目的を達成させるべく，最短距離のルート，あるいはまた，最小費用のルートを見つける問題のように，空間あるいは時間という局面のなかで，自然にその様相を呈するものである。このような問題の場合，ダイナミック・プログラミングは，すでに示した理由によって，後方探究的なものとなるであろうが，そこでの段階は，当該空間あるいは当該時間における自然の停止点を意味するものとなる。

これに関連して，ダイナミック・プログラミングのもっとも重要な適用例を，われわれ

は，将来の需要に対する各種生産能力の計画や，在庫調整などにもとめることができる。[43] そのとき，段階は，在庫量や計画期間を表わすことになり，また，その再帰式は，

$$f_M(K)=\min\{C(P+I)+f_{M+1}(K)\}$$

として表わされる。ただし，I は，現在入庫されている数量，P は，段階 M において生産された数量を表わし，また，$C(P+I)$は，在庫量を維持する費用と，その生産のための費用を表わすものとする。したがって，K は，生産量と入庫量から需要量を差し引いた残高であることが判明する。

同様に，生産過程の制御問題を定式化することも可能である。この場合，いかなる段階においても，その状態は，物理的ないしは化学的な過程によって規定されるものとなる。たとえば，R. E. Bellmanは，この方法で解くことのできる広範囲の問題群を指摘しているし，[44] また，R. Aris は，化学的な過程の最適化について，一冊の本を著わしている。[45] 最後に，十分満足のいくものではないとはいえ，所与の目的のために，機械や用役の段取りを決める問題について，ダイナミック・プログラミングを用いることができるように，そのような問題[46]を定式化することが可能であることを指摘しておく。[47]

V "Branch-and-bound" の手法

いわゆる "Branch-and-bound" の手法とは，図6，および図7によって明らかにしたように，最適でない選択枝を剪定するという考え方にもとづいた，1つの最適化の手順を意味するものである。したがって，それを概念的に述べれば，"Branch-and-bound" の手法は，ダイナミック・プログラミングの拡張であると考えることもできよう。

"Branch-and-bound" の手法が有する最大の利点は，当該問題の構造について，最小の仮定しか行なう必要がないという点にもとめられる。これを具体的に述べれば，ある与えられた変数についての収益関数が，他の変数についての収益関数とは独立である，という仮定を必要とはしないし，また，変数の連続性という仮定も必要とはしていない，ということである。その結果，"Branch-and-bound" の手法は，とくに整数計画問題に対し

(43) Wagner, H. M., *op. cit.*, 1969, および White, D. J., *op. cit.*, 1969, 参照。
(44) Bellman, R., *op. cit.*, 1961, 参照。
(45) Aris, R., *op. cit.*, 1961, 参照。
(46) これを "Job Sequencing Problem" という。
(47) Bellman, R., and S. Dreyfus, *op. cit.*, 1962, 参照。

て，首尾よく適用されてきたのである。[48] この手法が有するもう1つの実際的な利点は，たとえ，その最適化の手順を分析者の手もとにある問題に合うように加工する必要があるとはいえ，それは，すぐれて容易に，コンピュータのプログラムとして組み込むことができる点にもとめられる。

"Branch-and-bound" の手法は，もしも要素の結合の仕方に制約が課せられていないならば得られたであろう最適値と比較することによって，有効でない解の集合を切り捨てるものである。したがって，すべての制約をみたすという意味で，実行可能な解である変数の集合について，その上限[49]が，他の集合の上限よりも優位を占めるとき，われわれは，この後者の集合を，もはやこれ以上，考慮の対象とする必要がないわけである。すなわち，一旦，劣位におかれた変数の集合は，たとえそれが実行可能であったとしても，けっして最適解を与えるものではない，ということである。換言すれば，"Branch-and-bound" の手法にあっては，特定の解について，その実行可能性を吟味する必要がない，ということである。その意味で，"Branch-and-bound" の手法が有する効率性は，剪定の過程で可能解として考慮される，有意な変数の集合を規定する能力と，さらに，各集合の最適値についての稠密(tight)な限界を確立する能力に依存するものである。

つぎに，"Branch-and-bound" の手法が，どのように使われるのかを示すために，つぎのような人員配置問題を考えてみることにしよう。[50] この問題の目的は，与えられた人々を4つの仕事にそれぞれ割り当てることによって，そのすべての仕事を完成させるのに要する費用を最小にすることである。ただし，各人に異なった仕事を1つずつ割り当てた場合に要する費用は，表3に示されているものと想定しよう。

また，ここでの制約条件として，1つの仕事に対して割り当てられる人数は，ただ1人に限られているものと想定しよう。このとき，考えられる実行可能な組み合わせの総数は，4!=24となり，これらすべての可能解を逐一列挙し，その各々を吟味することによって，当該問題の最適解を得ることは，もとより可能である。しかし，"Branch-and-bound" の手法を用いれば，もっと少ない手順によってその最適解を得ることができるのである。

---

(48)　Wagner, H. M., *op. cit.*, 1969, 参照。

(49)　いうまでもなく，ここでは，最大化問題を前提としている。したがって，もしも，最小化問題を前提とすれば，「上限」という言葉を，「下限」という言葉に，置き換えなければならない。

(50)　この人員配置問題は，Hiller, F. S., and G. J. Lieberman, *op. cit.*, 1967, に負っている。

人	仕事			
	1	2	3	4
A	26	15	26	20
B	30	25	26	14
C	10	20	9	25
D	20	5	20	15

表3　A，B，C，Dの4人に対して，異なった4つの仕事を1つずつ割り当てた場合に要する費用

まず，最初のステップは，通常，最適値についての限界（これを“Bound”という）を決めることである。(51) 各々の仕事を完成させるのに要する最小費用の総和は，表3を吟味することによって，つぎのようにもとめられる。

総合的な下限（すなわち，費用）＝10＋5＋9＋14＝38

与えられた人々をどのように割り当てなおしたとしても，これより費用が安くなることはないのである。ところが，この下限は，C氏が，仕事1と仕事3のいずれにも割り当てられている結果，当該問題の制約条件をみたしておらず，したがって，実行可能な解ではないことが判明する。しかしながら，この総合的な下限が有する有効性は，たとえば，大規模な問題において，そのすべての可能性を探索することができないような場合に，“Branch-and-bound”の手法をいつ止めるべきかを決定する指針を，分析者やコンピュータのプログラマーに提供し得る点にもとめられる。すなわち，当該問題の解が，この下限にもっとも近くなったとき，われわれは，計算を打ち切ることができるのである。

つぎのステップは，比較のために，可能解の集合を定義することである。この問題では，仕事1を担当し得る4人の異なった人々によって，4つの自然なグループを構成することができる。さらに，また，この各々のグループに対して，費用についての稠密な下限を確立することもできるのである。たとえば，A氏が，仕事1を担当しているグループについて，その費用の下限をもとめれば，つぎのようになる。

A氏を含むグループの下限＝26＋5＋9＋14＝54

(51)　この問題については，下限を決めることを意味する。

なお，これは，実行可能解となっていることに注意されたい。また，これと同じように，2，3，4の仕事を割り当てるとき，B，C，Dの各氏を逐次取り除いて構成される，残りの3つの可能性について，その下限をもとめることも可能である。(52)

これらのステップによる検討結果は，つぎの図8に示されており，そこでは，4つの最初の解の集合が，最初のステップからの分枝（これを“Branch”という）として，表わされている。

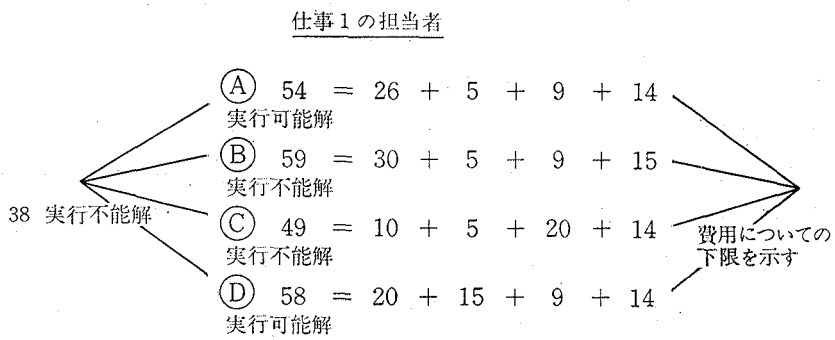

図8　A，B，C，Dの4人のうち，だれが仕事1を担当するかによって識別された，4つのグループの費用についての下限

このうち，A氏とD氏が，仕事1を担当したときの下限は，いずれも実行可能である。また，B氏とD氏のいずれか一方が，仕事1を担当したときの集合についての下限は，A氏が，その仕事を担当したときの実行可能解よりも高くなっているので，これら2つの解の集合を，われわれは，それ以降の考慮の対象から取り除くことができるのである。つぎに，C氏が，仕事1を担当したときのグループは，これまでのところ，最小費用の実行可能解とみなされていたもの（すなわち，A氏が，仕事1を担当することによって，54の費用をもたらすもの）と比較して，相対的に費用は少ないが，しかし実行不能な限界をもっていることが判明する。そこで，仕事1をC氏に担当させたときのグループが，はたして，この54という費用よりも相対的に少ない費用で，しかも実行可能な解をそのうちに含んで

(52)　たとえば，B氏を仕事1に割り当てるとき，残りの3つの仕事2，3，4に対して，B氏は必然的に排除される結果，B氏を除く最小費用の割り当てを表3によってもとめれば，仕事2をD氏，仕事3をC氏，仕事4をD氏に，それぞれ割り当てることになる。なお，この組み合わせが，図8の第2分枝に対応している。

いるかどうかを判定するために，このグループの部分集合を，さらに吟味しなければならなくなる。

かくして，第3のステップは，C氏が，仕事1を担当するグループの部分集合について，さらに検討を加えることである。ところで，この問題におけるその部分集合を，論理的に考察すれば，それは，A，B，Dの各氏に対して仕事2を割り当てる組み合わせを意味するものとなる。(53) これらの検討結果は，図9によって示されている。

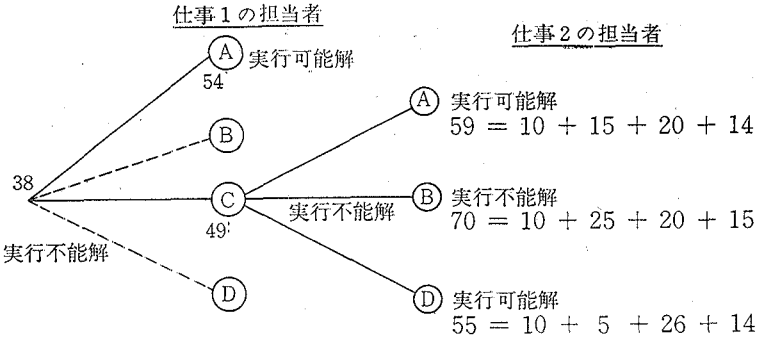

図9 第3のステップにおける，費用についての下限
ただし，第2のステップにおけるBおよびDを含むグループは，最適でない結果，除外されている。

この図より明らかなように，これらの部分集合について，その下限をもとめれば，55となっており，これは，第2のステップで得た実行可能な下限（すなわち，54）よりも大きな値となっている。したがって，結局，実行可能であるこの最下限（すなわち，54）が，当該問題の最適値となるのである。

要するに，われわれは，合計8通りの組み合わせ（すなわち，これは，総数24のうちのちょうど⅓にあたる）を吟味することによって，当該問題の最適解(54)を得ることができたのである。

最後に，“Branch-and-bound”の手法によって考慮されるべき組み合わせの総数が，

(53) なぜなら，C氏は，すでに仕事1を担当するべく，割り当てられているからである。

(54) すなわち，仕事1をA氏，仕事2をD氏，仕事3をC氏，仕事4をB氏に，それぞれ割り当てることを意味する。

一体いくらになるかを，われわれは，前以って決めることができない，という点を指摘しておこう。いうまでもなく，剪定を受ける可能解のグループは，著しい数にのぼるのではあるが，それでも，なお多くのグループが，検討の対象として残ることになれば，当該問題の最適解を，コンピュータによる計算の許容時間内で決定することは，かなり難しいものとなるであろう。そのような場合，分析者は，“Branch-and-bound”の手法を用いることによって，当初，彼が出発したときの解よりも，相対的に優る解を探索することを望む筈である。そのとき，彼が，絶対的な下限からどれほど乖離しているかを知ることは，当該問題の解明にとって，すぐれた指針を提供するものと考えられる。