解　説

難しい組合せ論的問題と近似アルゴリズム†

谷　口　健　一††

1. まえがき

アルゴリズムの設計と解析は計算機科学の重要な一分野を占めており，過去約20年間にわたって精力的な研究がなされてきた．問題固有の性質を有効に利用し，データ構造の細部まで工夫して能率のよいアルゴリズムが開発されてきた問題もいくつかある一方，能率のよいアルゴリズムが存在しそうにない難しい問題のあることもわかってきた．

本稿では，能率のよい解法が見つかっていない一連の問題について，能率のよいアルゴリズムはもはや存在しそうにないという強い予想の根拠や，近似解を能率よく求める近似アルゴリズム等について紹介する．

最適化問題の例として，次の問題を考えてみる．

●ナップザック問題：大きさがそれぞれ $s_1, s_2, \cdots, s_n$ であり，価値がそれぞれ $p_1, p_2, \cdots, p_n$ である n 個の物体と，容量が L の一つのナップザック（背負い袋）が与えられたとき，大きさの和が L 以下で，しかも価値の和を最大にするには，どれどれを選んでナップザックに詰めればよいかを求める問題（形式的に言えば，$\sum_{i=1}^{n} x_i s_i \leq L$ の制限の下で $\sum_{i=1}^{n} x_i p_i$ を最大にするような 0/1 ベクトル $(x_1, x_2, \cdots, x_n)$ を求める問題）．

このような最適化問題に対応して，ごく自然に，つぎのような判定問題（決定問題ともいわれる）が考えられる．判定問題における個々の個別問題に対する答は “Yes”（はい），または “No”（いいえ）である．

●ナップザック判定問題：大きさ $s_1, s_2, \cdots, s_n$，価値 $p_1, p_2, \cdots, p_n$ および容量の上限 L，価値の下限 k が与えられたとき，

$$\sum_{i=1}^{n} x_i s_i \leq L \text{ かつ } \sum_{i=1}^{n} x_i p_i \geq k$$

であるような 0/1 ベクトル $(x_1, x_2, \cdots, x_n)$ があるかどうかを判定する問題．

† Difficult Combinatorial Problems and Approximation Algorithms by Kenichi TANIGUCHI (Faculty of Engineering Science, Osaka University).

†† 大阪大学基礎工学部情報工学科

表-1　入力サイズと計算時間

計算時間（マイクロ秒）	入力サイズ n 10	20	30	50	100	200
$1{,}000n$	0.01秒	0.02秒	0.03秒	0.05秒	0.1秒	0.2秒
$100n^2$	0.01秒	0.04秒	0.09秒	0.25秒	1.0秒	4.0秒
$10n^3$	1.0秒	32.0秒	4.0分	52.1分	27.8時間	37日
2^n	0.001秒	1.0秒	17.9分	35.7年	—	—
$n!$	3.6秒	7.7×10^4年	—	—	—	—

最適化問題に対するアルゴリズムを用いて，その解（答）から，対応する判定問題に対する答がただちにわかるという意味で，判定問題の方がより易しい（難しいことはない）と考えられる．

このような問題も，能率のことを考えなければ，およそ可能な組合せをすべて列挙し，条件に合うものを選び出す（あるいはそれがあるかを調べる）という方法で解くことができる．しかし，ナップザック問題における可能な組合せの数（部分集合の個数）は 2^n であり，もしそのような方法を使うとすれば，与えられる個別問題のサイズが大きくなったとき，計算に要する時間は途方もなく大きくなって現実には不可能である（表-1）．

また，計算時間が 2^n や $n!$ に比例するアルゴリズムでは，計算機が速くなっても，解ける問題のサイズ n の増加はほとんど期待できない．

すべての可能な組合せをなるべく無駄を省いて調べる手法として，バックトラック法（分枝限定法），動的計画法などがある．現実に起こりうるような具体例に対しては，これらの方法が十分役立つ場合が多いが，最悪の場合，あるいは，平均として，いずれも，与えられる個別問題のサイズ（記述長）に対し，指数的あるいはそれ以上に増加するのが普通である．

2. アルゴリズムの計算複雑度

2.1 多項式時間アルゴリズム

問題 Q に属する各個別問題（その問題の各具体例）I は，固定した有限個の基本記号の系列として記述される．このような記述の系列全体の集合を $L(Q)$ とす

る．ある正数 c，関数 g があって，$L(Q)$ に属する任意の長さ n の系列（個別問題 I の記述とし，その長さ n を個別問題 I のサイズあるいは入力サイズともいう）が与えられたとき，アルゴリズム A により $cg(n)$ 時間以下の計算時間で I に対する正解を求められるとき，A の時間複雑度（時間計算量ともいわれる）は，$O(g(n))$（オーダ $g(n)$ と読む）であるといい，また，問題 Q は $O(g(n))$ 時間で計算可能な問題という．

この定義は，Q のみでなく，基本記号および符号化の方法にも依存しているが，“標準的”な方法（たとえば，m 個の頂点をもつグラフは，各頂点を1から m までの10進数で表わし，各辺をその端点を示す10進数 i, j の対 (i, j) で表わすなど）を用いる限り，議論はそれ程変らない．

多項式時間アルゴリズムとは，ある多項式 $p(n)$ が存在して，そのアルゴリズムの時間複雑度が $O(p(n))$ であるようなアルゴリズムをいう．言いかえれば，各 n に対し，サイズが n のどんな個別問題に対しても，計算時間が cn^d 以内であると言えるような定数 c, d が存在するときをいう．

以下において多項式時間アルゴリズムの“クラス”を考える主な理由は，

- 多項式時間アルゴリズムは“能率のよい”方のアルゴリズムと考えられること．すなわち，少なくとも多項式時間のアルゴリズムが無ければ，実際上ほとんど解けないし(表-1)，また，多項式時間で解けるとき，現実の問題では $O(n^d)$ の d は比較的小さいことが多い．
- 多項式のクラスはかけ算や合成で閉じているので，いくつかの多項式時間アルゴリズムからつくった一つのアルゴリズムは多項式時間アルゴリズムとなる．
- 計算機械（あるいはアルゴリズム）のモデルとしていろいろあるが，多項式時間というクラスで議論する限り，差はない*．

多項式時間アルゴリズムで解ける判定問題のクラスを $\boldsymbol{P}$ で表わす．

2.2 非決定性アルゴリズム

理論的な考察においては，いわゆる“推測”の機能をもった“非決定性アルゴリズム”を考えることができる**．非決定性アルゴリズムは，通常の演算・命令のほかに，(i)可能性の中から任意のものを選び出すという **choice** 文，および，(ii)2種類の停止文 **failure**, **success** を持つ．**choice**(S) は集合Sから任意の一つを(何の判断基準もなく)選ぶという操作である．たとえば，代入文

x←**choice** (1 : n)

は，整数の集合 $\{1, 2, \cdots, n\}$ の中の任意の一つが変数 x に代入されるという意味である．**failure** はその非決定性アルゴリズムによる計算が途中の **choice** での選択がまずく失敗したということを表わして停止する命令である．**success** に至ったときの出力がそのアルゴリズムによる計算結果であると考える．したがって，上手な選択のしかたがあって，そのうちの一つでも，**success** へ導くような計算過程が存在するものなら，その選び方で計算が行われるものと考えてよい．

図-1 に非決定性ソーティングアルゴリズムの例を示す[1]．2～6行目で，$a_1, \cdots, a_n$ を任意に置換したものをつくり，7～9行目で，それが“≦”の順になっているかをチェックしている．

非決定性アルゴリズム A は入力 I に対して **success** へ至るような計算の道があるとする．A の I に対する計算時間 $t_A(I)$ とは，**success** へ至るまでの計算時間であって，もし **success** へ至るような計算の道がいくつかあるときは，そのうちの最小の計算時間をいう(**choice** は単位時間で実行されるとする)．ある正数 c，関数 g があって，サイズが n で，かつ，それに対して **success** へ到達するというような任意の個別問題 I に対して，$t_A(I) \leq cg(n)$ がなりたつとき，A の時間複雑度は $O(g(n))$ であるという．前述の図-1 は時間複雑度が $O(n)$ の非決定性アルゴリズムである．

非決定性アルゴリズム A が判定問題 Q に対する非決定性判定アルゴリズムであるというのは次の条件を満

```
入力  n 個の正整数 a1, a2, ..., an
出力  入力の正整数を非減小の順に並べ換えたもの．答は長さ
      n の配列 S に入れる．
手続き
 1  各 i について，S(i) を 0 にセットする．
 2  for i←1 to n do
 3    j←choice (1 : n)
 4    if S(j)≠0 then failure
 5                 else S(j)←ai
 6    end
 7  for i←1 to n−1 do
 8    if S(i)>S(i+1) then failure
 9    end
10  S(1), S(2), ..., S(n) を出力
11  success
```

図-1 非決定性ソーティングアルゴリズム

* ただし，(計算機の一語に入らない)いくらでも大きい数の乗除算が一定時間で実行できるとは考えない．本稿では，計算機械のモデルとして，常識的に，記憶容量の十分大きい固定語長の計算機を考えればよい．

** 非決定性チューリング機械に対応する．

たすときをいう.

条件:「Qの個別問題全体の集合を $L(Q)$ とする. $L(Q)$のうち,答がYesであるような個別問題Iが与えられたときは,必ずある選択のしかたがあって**success**へ至る.答がNoであるような個別問題Iに対しては,**success**へ至るような選択のしかたが存在しない,すなわち,どのように選択しても,必ず,**failure**に到達するか,あるいは,無限ループに陥る.」

このとき,非決定性アルゴリズムAは判定問題Qを"解く"ということにする.**success**へ至ったとき,答Yesを出力するようにしておけば,その答は正しい.しかし,ある選択のしかたで**failure**へ至ったからといって,答はNoとは限らない.決して**success**へ至らないとき,答はNoであると見なすのである.

非決定性判定アルゴリズムにより,多項式時間で解ける判定問題のクラスをNPと書く.すなわち,答がYesであるような個別問題が与えられたとき,**success**へ至る(判定結果のYesを出力する)までの最短の計算時間が入力サイズのある多項式でおさえられているような判定問題のクラスである.前述のナップザック判定問題はNPに属する.「……を満たすような解があるかどうか」といったタイプの判定問題に対しては,(i)**choice**による非決定性を利用し,任意に"解"(ナップザック判定問題の場合は0/1ベクトル$(x_1,\cdots,x_n)$)を一つ推測し,次いで(ii)決定性アルゴリズムにより,その推測が正しいか(今の場合は,その$(x_1,\cdots,x_n)$が条件$\sum_{i=1}^{n} x_i s_i \leq L$, $\sum_{i=1}^{n} x_i p_i \geq k$を満たしているか)どうかを調べる(もしそうなら**success**で停止),という形の非決定性アルゴリズムを考えるのが普通である.NPに入るためには,上の(i),(ii)が入力サイズの多項式時間で実行できなければならない.

注意1:決定性の多項式時間判定アルゴリズムは答がYesのときもNoのときも,入力サイズのある多項式時間以内に停止して,その答を出力する.

注意2:判定問題Qを解く非決定性アルゴリズムAにたとえ**choice**が含まれていなくても,それはQを解く決定性アルゴリズムとは必ずしも見なせない.この定義によれば,答がNoであるような個別問題に対しては,前者は必ずしも計算が停止しなくてよく,後者は停止してNoを出力しなければならない.

2.3 P と NP の関係

決定性アルゴリズムにより多項式時間で解ける判定問題は,非決定性アルゴリズムにより多項式時間で解ける.したがって$P \subseteq NP$であることは明らかである.ところで,NPはPより真に広いのであろうか.そのことは十分予想されるが,まだ証明されていない.

「$P=NP$か,あるいは,$P \neq NP$か」

のどちらであるかを示すことは,現在,計算機科学における代表的な難問の一つである.

$P \neq NP$の予想のよりどころとして,

- NPではあるが,多項式時間の決定性判定アルゴリズムが見つかっていない問題が非常にたくさんある.
- そのような問題に対し,たくさんの人が長い間考えたにもかかわらず,誰一人として多項式時間のアルゴリズムを見つけることができなかった.
- **choice**による推測の力がそれほど小さいとは考えられない.

などがあげられるが,$P=NP$か$P \neq NP$かという問題に関連した議論を次の3で行い,$P \neq NP$という予想のより強力な"状況証拠"を与える.

3. NP 困難と NP 完全

3.1 定 義

問題Q_1をQ_2に(多項式時間)帰着できるとは,「問題Q_2を解く多項式時間アルゴリズムがもしあるなら,それをサブルーチンとして(必要なら何回でも)利用して,Q_1を解く多項式時間アルゴリズムが構成できる」ときをいう.そのことを$Q_1 \propto Q_2$と書く.この意味では,ナップザック最適化問題はナップザック判定問題に帰着できる*(最適解の値$\sum x_i p_i$を求めるのに2分探索法を用いよ).

"帰着"の特別な場合として,問題Q_1からQ_2への"(多項式時間)変換"がある.Q_1, Q_2が判定問題のとき,Q_1からQ_2へ(多項式時間)変換できるとは,「Q_1における任意の個別問題$I \in L_1(Q_1)$を,Q_2におけるある個別問題$f(I) \in L_2(Q_2)$に,Q_1におけるIに対する答がYesのとき,かつそのときに限り,Q_2における$f(I)$に対する答がYesであるように変換する(fを計算する)(決定性)多項式時間アルゴリズムがある」ときをいう.

問題Qに対し,NPに属する任意の判定問題Q_iを,それぞれ,そのQに帰着させることができるとき,QはNP困難であるという.QがNP困難であるというのは,多項式時間程度の差は同じと見るなら,Qは

* ナップザック判定問題がナップザック最適化問題に帰着できるのはもちろんである.

NPに属するどの問題よりも易しくなく，少なくとも同程度あるいはそれ以上に難しい問題であることを意味する．定義より

〔性質〕 Qを**NP**困難な問題とする．もしQが多項式時間アルゴリズムで解けるならば，$P=NP$がなりたつ．

判定問題Qは，(i) $Q\in NP$，かつ (ii) Qは**NP**困難，であるとき，**NP**完全であるという*．定義より，

〔性質〕 Qを**NP**完全な問題とする．$Q\in P$のとき，かつそのときのみ，$P=NP$である．

このような性質をもつある特定の判定問題Qは実際に存在するのであろうか．S. A. Cookは，このような疑問について考察し，1971年に次の画期的な結果を証明した．

〔Cookの定理〕 和積形論理式の充足可能性問題SATは**NP**完全である[4]．

和積形論理式の充足可能性問題とは，与えられた和積形論理式(たとえば$(x_1\vee\bar{x}_2\vee x_4)\wedge(\bar{x}_1\vee x_3)\wedge(x_2\vee\bar{x}_3\vee x_4)$)に対し，その式の値を真理値'1'にするような，各変数への'0'，'1'の割当て方があるかどうか，を判定する問題である．この定理の証明は省くが，SAT$\in NP$であり，また，多項式時間非決定性判定アルゴリズムAとそれへの入力Iから，「AのIに対する計算で，**success**へ至るような計算の道が存在するとき，かつそのときのみ，$E(A,I)$は充足可能である」という性質をもつ和積形論理式$E(A,I)$が，(Iの記述長に関する)多項式時間の決定性アルゴリズムによりつくれるのである．

Qを**NP**完全な問題とする．$Q\propto Q'$かつ$Q'\in NP$ならば，Q'も**NP**完全な問題である．この方法により，SATから出発して，それ以外のいくつかの問題も**NP**完全であることが示せる．今では何百という数の**NP**完全な問題が知られている．文献3)の後半にはそのリストが載っている．

何百という数の**NP**完全問題のどれに対しても多項式時間決定性アルゴリズムが見つかっていないという事実や，もし一つでも多項式時間アルゴリズムが見つかればほかのすべての**NP**完全な問題(および**NP**に属するすべての問題)に対するそれぞれの多項式時間決定性アルゴリズムが自動的に出てくるという信じ難い結果より，

「**NP**完全な(あるいは**NP**困難な)問題に対しては，決定性の多項式時間アルゴリズムは存在しない．したがって，$P\neq NP$．」

と予想される．信じるに足る十分な理由はあるが，証明されているものではない．このように，**NP**完全(あるいは**NP**困難)であることが示された問題は，最悪の場合の計算時間が入力サイズの多項式でおさえられるような能率のよいアルゴリズムは存在しそうにない"難しい問題"である．

3.2 NP困難問題の例

(1) 前述のナップザック問題は**NP**困難であり，ナップザック判定問題は**NP**完全である．

(2) グラフの彩色問題

グラフGが与えられて，辺で結ばれた頂点は互いに異なる色を塗るという条件の下で，最少数の色を用いてGの各頂点を彩色する問題をグラフの彩色問題という．この問題は**NP**困難である．正整数kも与えて，「k色でGを彩色できるか」という判定問題は**NP**完全である．特に，$k=3$と固定し，かつ，Gが頂点次数高々4の平面グラフであっても**NP**完全である．

(3) 巡回セールスマン問題

辺に重み(>0)の付いた完全グラフが与えられたとき，すべての頂点をちょうど1度ずつ通過して元の頂点へ戻ってくるような閉路のうち，それに含まれる辺の重みの和が最小であるような閉路を見い出す問題(巡回セールスマン問題)は**NP**困難であり，$k(>0)$も与えて，「重みの和がk以下のそのような閉路があるか」という判定問題は**NP**完全である．

(4) スケジューリング問題[5]

m台の同一プロセッサからなる処理系において，処理されるべきn個のタスクの集合$\{T_1,T_2,\cdots,T_n\}$および各T_iについて，その処理に要する時間$\tau_i(>0)$が与えられているとする．各タスクはどの時点においても一台のプロセッサでしか処理できないし，また，各プロセッサはどの時点においても一つのタスクしか処理できない．スケジュールとは各タスクをいつからいつまでどのプロセッサで処理すべきかを定めたものである．あるタスクの処理が開始されたなら，そのタスクの処理が終了するまでそのプロセッサを専有するという制約の下でのスケジュール(横取り権の無いスケジュール，non-preemptive schedule)を考える．スケジュールSにおけるT_iの終了時刻を$f_i(S)$と書くと，$\mathrm{FT}(S)=\max_{1\le i\le n}\{f_i(S)\}$を$S$のスケジュール長といい，これを最小にするような$S$を求める問題を，

* 通常，条件(ii)における"帰着"は"変換"の方を用いる．その方が見かけ上条件が強い．

最短スケジュール問題という．$m=2$ と固定した場合でさえ，最短スケジュール問題は ***NP*** 困難であり，期限Dも与えて，$FT(S)\leq D$ なるスケジュール S があるかという判定問題は ***NP*** 完全である．

タスク間に処理の前後関係を示す半順序関係‘<’も指定されている場合を考える．$T_i<T_j$ は，T_i が終了した時点以降でしか T_j は開始できないということを示す．‘<’も指定する場合，各タスクの処理時間を同一ですべて1である（$\tau_i=1, 1\leq i\leq n$）としても，最短スケジュール問題は ***NP*** 困難であり[6]，$D=3$ として，$FT(S)\leq D$ なる S があるかという判定問題も ***NP*** 完全である[7]．なお，特定の固定した $m(\geq 3)$ について，最短スケジュールを求める多項式時間のアルゴリズムがあるかどうかは未解決である．処理時間として1と2の二種類を許せば，$m=2$ と固定して，最短スケジュール問題は ***NP*** 困難，対応する判定問題は ***NP*** 完全になる[6]．

（5） コード生成問題

計算過程を表現した（出次数が2以下の，すなわち，高々2項の演算子しか含まない）閉路のない有向グラフ $G=(V, E)$ が与えられたとき，G のすべての葉頂点（出次数0の頂点）の値はメモリ内にあるとし，LOAD 命令（指定したメモリの内容をレジスタへ移す），各 OP 命令（演算要素の一方（あるいは指定された方）がレジスタにあるとき，指定された演算を行って結果をレジスタに残す），STORE 命令（レジスタの内容を指定したメモリ内へ移す）のみを用いて，Gのすべての根頂点（入次数0の頂点）の値を計算する命令数最小のプログラムを生成する問題（1レジスタ機械に対する最適コード生成問題）は ***NP*** 困難である[8]．正整数 k も与えて，「命令数 k 以下のプログラムがあるか」という判定問題は ***NP*** 完全である．

4. 近似アルゴリズム

NP 困難な問題，***NP*** 完全な問題に対しては，現在知られているアルゴリズムは，最悪の場合，入力サイズに対して指数的に手数が増大する．そこで，***NP*** 困難であることが示されている最適化問題に対して，

「求まる答は必ずしも最適解でなくてもよく，最適解に近いものであればよい．しかし，答を出すまでの時間は入力サイズに対して少なくとも多項式，できるならば $O(n), O(n^2)$ 程度にしたい．」

ということが考えられる．実用的にはこれで十分なことが多い．そのように，最適解に近い解を近似解といい，それを求める（決定性）アルゴリズムを近似アルゴリズムという．

4.1 近似の程度

Qを最適化問題とし，Qの一つの個別問題を I とする．I に対する許容解（必ずしも最適解ではないが，解の候補となるもの．たとえば，ナップザック問題では選んだ物体番号の集合（0/1 ベクトル $(x_1, \cdots, x_n)$ で表わされる）で大きさの和 $\sum x_i s_i$ が上限 L を越えないもの）のそれぞれに“値”（ナップザック問題では価値の和 $\sum x_i p_i$）が対応づけられる．I に対する最適解の“値”を $OPT(I)$ で表わす．

Qに対する近似アルゴリズム Aは与えられた各個別問題 I に対し，I に対する許容解の一つを出力する．その許容解の“値”を $v_A(I)$ で表わす．

問題Qに対する近似アルゴリズム Aは，Qのすべての個別問題 I に対し $|OPT(I)-v_A(I)|\leq k$ であるような定数 k があるとき，絶対近似アルゴリズムという．

問題Qに対する近似アルゴリズム Aは，Qのサイズ n のすべての個別問題 I に対し

$$|OPT(I)-v_A(I)|/OPT(I)\leq f(n)$$

であるとき，$f(n)$-近似アルゴリズムという．（$OPT(I)>0$ を仮定している．）とくに，ある定数 ε があって $f(n)\leq\varepsilon$ であるとき，ε-近似アルゴリズムという．

任意に小さい $\varepsilon(\varepsilon>0)$ が指定されても，すべての個別問題 I に対し

$$|OPT(I)-v_{A_\varepsilon}(I)|/OPT(I)\leq\varepsilon$$

が成り立つような能率の良い近似アルゴリズム A_ε があることが望ましい．ε もパラメータとするそのような一般的な ε-近似アルゴリズム（のクラス）を近似スキームという．とくに，計算時間が個別問題のサイズ n と $1/\varepsilon$ の両方の多項式で押えられるようなものを両完全多項式時間近似スキームという．答の精度，計算時間の点から望ましいものである．

4.2 絶対近似アルゴリズム

多項式時間の絶対近似アルゴリズムがあるような ***NP*** 困難な最適化問題は少ない．一つの例は，平面グラフの彩色数を求める問題である．与えられた平面グラフが3色で彩色できるかという判定問題は ***NP*** 完全である．しかし，すべての平面グラフは4色で彩色可能であることが知られているので，無条件に答4を出力するアルゴリズムは絶対近似アルゴリズムといえる．

絶対近似アルゴリズムは望ましい近似アルゴリズムといえるが，多くの ***NP*** 困難な最適化問題に対して，多項式時間の絶対近似アルゴリズムはほとんど期待で

きない．たとえば，$P \neq NP$ ならば，ナップザック問題に対する多項式時間の絶対近似アルゴリズムは存在しない．これは，価値 p_i の互いの差が絶対誤差 k より大きければ，絶対近似アルゴリズムは最適解を出力しなければならないということから示せる．

4.3 ε-近似アルゴリズム

非常に単純な方法でも，かなり良い ε-近似アルゴリズムとなることがある．例を示す．

（1） 箱詰め問題

大きさが $l_i(1 \leq i \leq n)$ である n 個の物体と容量が L の箱がいくつか与えられたとき，n 個の物体をどのように詰めれば，必要な箱の個数が最小になるか，という箱詰め問題を考える．ただし，各物体は小さく分けて異なる箱に入れることはできず，また，各箱に詰められた物体の大きさの和が L を越えてはいけない．

この問題は NP 困難であることがわかっている．この問題に対する単純な経験的法則として次の4つがあげられる．

a） 最初に合った箱へ (First Fit, FF)：物体を 1, 2, …, n とし，その順に詰めていくことを考える．物体 i を可能な限り番号の小さい箱へ詰める（$L-l_i$ 以上空いているような最初の箱を見つけ，それへ詰める）．

b） 最も良く合う箱へ (Best Fit, BF)：物体を 1, 2, …, n の順に詰めるが，物体 i はそれを詰めると箱が最も一杯になる（$L-l_i$ 以上空きがあるが，それが最も小さい）もので，番号が小さい箱へ詰める．

c） 大きいものから，最初に合った箱へ (First Fit Decreasing, FFD)：物体を $l_i \geq l_{i+1}(1 \leq i < n)$ になるように並べ換えたのち，FF の方法を使う．

d） 大きいものから，最も良く合う箱へ (Best Fit Decreasing, BFD)：物体を $l_i \geq l_{i+1}(1 \leq i < n)$ になるように並べ換えたのち，BF の方法を使う．

I を箱詰め問題の任意の個別問題とし，必要な箱の最小個数を $\mathrm{OPT}(I)$ とする．FF あるいは BF で得られる箱の個数は $(17/10)\,\mathrm{OPT}(I)+2$ を越えず，また，FFD あるいは BFD で得られる箱の個数は $(11/9)\,\mathrm{OPT}(I)+4$ を越えないということが示されている[9]．

（2） スケジューリング問題

前述の m プロセッサ・n タスクの（半順序関係 '>' が指定されていない場合の）最短スケジュールを求める問題を考える．LPT (Longest Processing Time-first) スケジュールとは，プロセッサが空いたとき，残っている未割当てのタスクのうち，処理時間最大のものをそのプロセッサに割当てるという方法で得られるスケジュールをいう．任意の個別問題 I に対し，最適スケジュール，および LPT スケジュールのスケジュール長をそれぞれ $\mathrm{OPT}(I)$, $v_{\mathrm{LPT}}(I)$ とすると

$$(v_{\mathrm{LPT}}(I)-\mathrm{OPT}(I))/\mathrm{OPT}(I) \leq 1/3-1/(3m) \leq 1.33\cdots$$

が成立する[10]．LPT スケジュールは $O(n \log n)$ 時間で求められる．また，相対誤差 $(v_A(I)-\mathrm{OPT}(I))/\mathrm{OPT}(I)$ が 1.22… 以下であるような近似解を求めるほとんど $O(n \log n)$ 時間のアルゴリズムも知られている[11]．

4.4 ε-近似スキーム

前述の m プロセッサ・n タスクの（順序関係の指定のない）最短スケジュール問題に対する以下の近似アルゴリズムを考える．

（i） k をある固定した正整数とする．

（ii） n 個のタスクのうち，処理時間が長いものから k 個を選び，その k タスクに対する最適スケジュールを求めよ．

（iii） 残りの $n-k$ タスクは，プロセッサが空いたとき勝手に割当てよ（もちろん，たとえば LPT の方法を用いた方がよい）．

最短スケジュール問題の任意の個別問題 I に対し，最適スケジュールおよび上記の方法で得られるスケジュールのスケジュール長をそれぞれ $\mathrm{OPT}(I)$, $v_{k\text{-OPT}}(I)$ とすると

$$\frac{v_{k\text{-OPT}}(I)-\mathrm{OPT}(I)}{\mathrm{OPT}(I)} \leq \frac{1-1/m}{1+\lfloor k/m \rfloor}$$

が成立する[10]．任意の誤差の精度 $\varepsilon>0$ が指定されたときは $(1-1/m)/(1+\lfloor k/m \rfloor) \leq \varepsilon$ を満たす k を選んで使用すれば，上記の方法は ε-近似スキームとなる．

計算時間は，n タスクを処理時間の長いもの順に並べ換える時間 $O(n \log n)$ と，m プロセッサ・k タスクの最適スケジュールを求める時間，たとえば分枝根定法で $O(m^k)$ 時間，の和である．後者が支配的であり，プロセッサ数 m，および，ε あるいは k を固定すれば計算時間はタスク数 n に対し多項式的であるが，そうでなければ $1/\varepsilon$ に対しては指数的に増大する．

入力サイズ n に対して多項式時間の ε-近似スキームの存在が期待できない最適化問題も多い．たとえば，巡回セールスマン問題に対しては，どんなに大きな定数 K をもってきても，$v_A(I) \leq K \cdot \mathrm{OPT}(I)$ を保証する多項式時間アルゴリズムは，もし $P \neq NP$ ならば存在しないということが示される．

4.5 両完全多項式時間近似スキーム

問題のサイズを n，許される相対誤差を ε とすると，計算時間が n と $1/\varepsilon$ の両方の多項式であるような近似

スキームを得るテクニックが知られている[1),12)]．ナップザック問題を例にとって，それを紹介しよう．

（1） 丸め法：個別問題 I：$(s_1, s_2, \cdots, s_n)$；$(p_1, p_2, \cdots, p_n)$；L および ε が与えられたとする（どの s_i も L より大きくないとする）．$\mathrm{OPT}(I) \geq \max\{p_i\}$ ゆえ，各 p_i において $\lfloor \max\{p_i\}\varepsilon/n \rfloor$ 以下の端数を無視しても誤差の比率 $|\mathrm{OPT}(\cdot) - v_A(\cdot)|/\mathrm{OPT}(\cdot)$ は ε 以内におさまるので，各 p_i を $p_i' = p_i - (p_i \text{ を } \max\{p_i\}\varepsilon/n \text{ で割った余り})$ に丸める．さらに見やすいように，$\max\{p_i\}\varepsilon/n$ を '1' と見て各 p_i' を $p_i'' = p_i'/(\max\{p_i\}\varepsilon/n)$ でおきなおす．p_i'' は $p_i'' \leq \lfloor n/\varepsilon \rfloor$ なる整数である．その個別問題 I''：$(s_1, \cdots, s_n)$；$(p_1'', \cdots, p_n'')$：L に対する最適解 $(x_1, \cdots, x_n)$ を求め，それを I に対する近似解とする．I'' に対する最適解は分枝限定法あるいは動的計画法と呼ばれる次の方法で求める．

$x_1, x_2, \cdots, x_i$ まで 0, 1 を割当てたとき，

$$s = \sum_{j=1}^{i} x_j s_j,\ p = \sum_{j=1}^{i} x_j p_j''$$

とおいた対 $<s, p>$ のすべての集合のうち $s \leq L$ である（すなわち，許容解になりうる）ものの全体を $S^{(i)}$ と表わす．$S^{(0)} = \{<0, 0>\}$ から順次 $S^{(1)}, S^{(2)}, \cdots, S^{(n)}$ を求める．ただし，ナップザック問題では

$$s \leq s' \text{ かつ } p \geq p'$$

であるような $<s, p>, <s', p'>$ がともに $S^{(i)}$ に入ってくるときは $<s, p>$ を残し，$<s', p'>$ は除去してもかまわない．$S^{(i)}$ にそのような $<s', p'>$ は無いとすると，各 p_i'' は整数 $\lfloor n/\varepsilon \rfloor$ であるので

$$|S^{(i)}| \leq 1 + \sum_{j=1}^{i} p_j'' \leq 1 + i\lfloor n/\varepsilon \rfloor.$$

少し工夫すれば，除去ルールを適用しつつ，$S^{(i)}$ から $S^{(i+1)}$ を，$|S^{(i)}|$ に比例する手間で求められるので，$S^{(n)}$ を求める全計算時間は $\sum_{i=1}^{n} |S^{(i)}| \leq O(n^3/\varepsilon)$ である．$S^{(n)}$ から逆にさかのぼって，最適解を見い出すのは容易である．ナップザック問題に対しては，この手法をさらに精密化して，$O(n/\varepsilon^2 + n \log n)$ 時間の近似スキームが得られており[13)]，さらに，それより速いものも知られている．

（2） 区間分割法：与えられた個別問題 I に対し，$x_1, \cdots, x_i$ まで 0, 1 を割当てた許容解の集合を $S^{(i)}$ とし，$S^{(0)}$ から順次 $S^{(1)}, \cdots, S^{(n)}$ を求めていく．$S^{(i)}$ に属する許容解のうちで $\sum_{j=1}^{i} x_j p_j$ の最大値を P_i とし，区間 $[0, P_i]$ を幅 $P_i\varepsilon/(n-1)$ の $\lceil (n-1)/\varepsilon \rceil$ 個の部分区間に分割し，$\sum_{j=1}^{i} x_j p_j$ が同じ部分区間に入るものは同一視し，一つだけ代表を選んで $S^{(i)}$ の要素を減らす．そうすると，$S^{(i)}$ の要素数は高々 $\lceil (n-1)/\varepsilon \rceil$ であり，$S^{(n)}$ を求める全計算時間は $\sum_{i=1}^{n} |S^{(i)}| = O(n^2/\varepsilon)$ でよい．また，誤差は加法的であり，$|\mathrm{OPT}(I) - v_A(I)| \leq \sum_{i=1}^{n-1} P_i \varepsilon/(n-1)$．$P_i \leq \mathrm{OPT}(I)$ ゆえ，$|\mathrm{OPT}(I) - v_A(I)|/\mathrm{OPT}(I) \leq \varepsilon$ である．

（3） 分離法：区間分割法とほぼ同様であるが，$S^{(i)}$ 中で $\sum_{j=1}^{i} x_j p_j$ の値が互いに $P_i\varepsilon/(n-1)$ 以内の要素は選ばないようにするやり方である．計算時間や誤差に関しては区間分割法と同じ議論が成り立つ．

両完全多項式時間近似スキームは，どのような性質をもつ最適化問題に対して存在するのか？ これに関連して，以下のことが知られている．

Q を最適化問題とし，I をその個別問題とする．I は何個かの整数の並びで表現されているとし，その全記述長（ビット長）を $\mathrm{Length}(I)$，I 中の最大数を $\mathrm{Max}(I)$ とする．多項式 p に対し，$\mathrm{Max}(I) \leq p(\mathrm{Length}(I))$ であるような Q の個別問題 I の集合を Q_p と書く．ある多項式 p があって Q_p が ***NP*** 困難であるとき，Q は強 ***NP*** 困難であるといわれる．前述のグラフの彩色問題，巡回セールスマン問題，箱詰め問題，m プロセッサ・n タスク・最短スケジュール問題等は強 ***NP*** 困難である．ナップザック問題は強 ***NP*** 困難ではないと思われる．

Q の最適解を求めるアルゴリズムは，時間複雑度が $\mathrm{Length}(I)$ と $\mathrm{Max}(I)$ に関する多項式であるとき，擬多項式時間アルゴリズムと呼ばれる．ナップザック問題は分枝限定法あるいは動的計画法により O（対象物の個数 $n \times \sum_{i=1}^{n} p_i$）*時間アルゴリズムで，したがって，$O(\mathrm{Length}(I)^2 \times \mathrm{Max}(I))$ の擬多項式時間アルゴリズムで解ける[1)]．また，最短スケジュール問題は任意の固定した m に対し，擬多項式時間アルゴリズムで解ける．

上述の両完全多項式時間スキームはこのような擬多項式時間アルゴリズムを，許される精度 ε を考慮し，計算の手間を省くように修正して得られたものである．

Q を最適化問題とし，その個別問題 I に対する許容解は正整数の値をもち，かつ，最適解の値 $\mathrm{OPT}(I)$ はある多項式 p によって $\mathrm{OPT}(I) < p(\mathrm{Length}(I), \mathrm{Max}(I))$ のように押えられているとする．もし Q が両完全多項式時間スキームを持つなら，Q は擬多項式時間アルゴリズムで解ける，ということがわかっている[14)]．

---

* 各 p_i は整数としている．

強 *NP* 困難な問題は，定義より，*P=NP* でなければ擬多項式時間アルゴリズムを持たない．したがって，上述の OPT (I) に関する条件を満たす（すなわち，それほど大きくない正整数値の解がある）強 *NP* 困難問題，たとえば，彩色問題，箱詰め問題等は，*P=NP* でなければ，両完全多項式時間スキームを持ち得ない．彩色問題については，もっと強い結果が知られており，もし *P≠NP* ならば，任意の定数 $\gamma<2$, d に対し

$$v_A(I)\leq\gamma\cdot\mathrm{OPT}(I)+d$$

を保証する多項式時間の近似アルゴリズムはない[15]．

以上より，*NP* 困難な問題は（対応する判定問題はいずれも *NP* 完全であり，同じ程度の難しさであるが）能率のよい近似アルゴリズムがあるかどうかに関しては問題によって非常に異なることがわかる．

5. あとがき

NP 完全問題に関連した諸結果については，文献3)に詳述されている．

ここで紹介した近似アルゴリズムは，最悪の場合でも誤差をこれこれに押えるという立場をとっている．これに対し，問題の具体例のある確率分布のもとにランダムに個別問題を選ぶならば，ほとんどの場合正解あるいは非常によい近似解を与えるというような“確率的に良いアルゴリズム”や，あるいは，アルゴリズム自身に乱数発生というようなランダム性を導入した“確率的アルゴリズム”などの考え方も提案されている．これらについては文献16)に概説されている．

NP 完全問題やここで扱った *NP* 困難問題はいずれも，それぞれ適当な多項式 p があって $O(2^{p(n)})$ 時間の決定性アルゴリズムで解けるものであった．それを解くには，たとえば $O(2^{\cdot^{\cdot^{\cdot^{2^n}}}})$ 時間は絶対に必要であると証明されているような真に難しい問題もあるが[2]，それらについてはふれなかった．また，クラス *NP* の中で難しい問題である *NP* 完全問題についてだけ述べた．多項式領域を用いて解ける判定問題のクラス，指数的時間を用いて解ける判定問題のクラス，等の中で難しい（そのクラスにおいて完全である）問題についてもいくつかの結果が知られているが，省略した．

謝辞 ご教示いただいた嵩忠雄教授，藤井護助教授に感謝する．

参考文献

1) Horowitz, E. and Sahni, S.: Fundamentals of computer algorithms, Computer Science Press (1978).
2) Aho, A. V., Hopcroft, J. E. and Ullman, J. D.: The design and analysis of computer algorithms, Addison-Wesley (1974)（野崎昭弘他訳：アルゴリズムの設計と解析 I, II, サイエンス社）.
3) Garey, M. R. and Johnson, D. S.: Computers and intractability-a guide to the theory of NP-completeness, W. H. Freeman and Company (1979).
4) Cook, S. A.: The complexity of theorem-proving procedures, Proc. 3rd Ann. ACM Sympo. on Theory of Computing, pp. 151-158 (1971).
5) Coffman, E. G., Jr. (ed.): Computer and job-shop scheduling theory, John Wiley & Sons (1976).
6) Ullman, J. D.: NP-complete scheduling problems, JCSS, Vol. 10, No. 3, pp. 384-393 (June 1975).
7) Lenstra, J. K. and Rinnooy Kan, A. H. G.: Complexity of scheduling under precedence constraints, Operations Research, Vol. 26, No. 1, pp. 22-35 (Jan.-Feb. 1978).
8) Bruno, J. and Sethi, R.: Code generation for a one-register machine, JACM, Vol. 23, No. 3, pp. 502-510 (July 1976).
9) Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R. and Graham, R. L.: Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. on Computing, Vol. 3, No. 4, pp. 299-325 (Dec. 1974).
10) Graham, R. L.: Bounds on multiprocessing timing anomalies, SIAM J. on Applied Mathematics, Vol. 17, No. 2, pp. 416-429 (Mar. 1969).
11) Coffman, E. G., Jr., Garey, M. R. and Johnson, D. S.: An application of bin-packing to multiprocessor scheduling, SIAM J. on Computing, Vol. 7, No. 1, pp. 1-17 (Feb. 1978).
12) Sahni, S.: General techniques for combinational approximation, Operations Research, Vol. 25, No. 6, pp. 920-936 (Nov.-Dec. 1977).
13) Ibarra, O. H. and Kim, C. E.: Fast approximation algorithms for the knapsack and sum of subset problems, JACM, Vol. 22, No. 4, pp. 463-468 (Oct. 1975).
14) Garey, M. R. and Johnson, D. S.: Strong NP-completeness results: motivation, examples, and implications, JACM, Vol. 25, No. 3, pp. 499-508 (July 1978).
15) Garey, M. R. and Johnson, D. S.: The complexity of near-optimal graph coloring, JACM, Vol. 23, No. 1, pp. 43-49 (Jan. 1976).
16) 五十嵐善英：確率的アルゴリズムの概観，情報処理，Vol. 21, No. 1, pp. 13-18 (1, 1980).

（昭和55年8月28日受付）