資 料 3

第５世代の電子計算機に関する
調査研究報告書
—— ソフトウェア工学 ——

昭 和 55 年 3 月

JIPDEC

財団法人 日本情報処理開発協会

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和５４年度に実施した「第５世代の電子計算機に関する調査研究」の成果をとりまとめたものであります。

ソフトウェア工学・執筆者

（順不同）

氏　　名	所　　属
古　川　康　一	電子技術総合研究所ソフトウェア部 情報システム研究室主任研究官
真　野　芳　久	電子技術総合研究所ソフトウェア部 言語処理研究室研究員
大谷木　重　夫	電子技術総合研究所制御部 論理システム研究室研究員
実　近　憲　昭	電子技術総合研究所制御部 論理システム研究室主任研究官
宮　川　正　弘	電子技術総合研究所パターン情報部 数理基礎研究室主任研究官
中　安　富士大	早稲田大学理工学部数学科

ソフトウェア工学

目　　次

1．はじめに

1. はじめに

ハードウェアの進歩に比べて、ソフトウェア開発技術が大きく遅れをとっていることは周知の事実である。このことが、今後、超ＬＳＩ（ＶＬＳＩ）の出現を迎えるにあたり、一層深刻な問題となることは、目に見えている。この大きなボトル・ネックをとり除くことが、今後の情報処理研究の最大の課題であることは、論を待たない。現在のソフトウェア技術のかかえている最も大きな問題は、ソフトウェアのハード化である。すなわち、ＯＳ、言語処理プロセッサ、データベースのような大規模ソフトウェアは、その維持、管理が大変なうえ、仕様の些細な変更に伴うシステムの改訂や、虫取り、あるいは他の計算機システムへの移行などが非常に困難である。この問題は、データベースのように、情報の蓄積が本質的であるシステムにとっては、まさに死命を制する問題である。

この問題を解決するには、つぎの２つの面からの追求が必要である。

(1)　システムを、必要以上に複雑にしないための技術

(2)　より複雑な問題を解決し得るソフトウェア技術

この２つの要求は、一見矛盾しているように見えるが、(1)は、問題が有している複雑さ以上にシステムが複雑になることを防ぐことを、(2)は、現在のソフトウェア・システムによって実現し得る機能を凌ぐ、より高度な機能が実現されるべきことを、夫々要求している。この２つの目標を達成するための、新しいソフトウェア工学の追求が必要である。その中心的な役割を果す可能性のある技術の第１は、データ抽象化に基づく階層プログラミング技法であり、第２は、ルールに基づくプログラムの変換技法である。この第２の方法は、記号レベルでのプログラムの実行方式（記号実行）と言い換えることもできる。

これらの２つの技術は、知識ベースを基にした知識情報処理システムにより、容易に実現される。その場合、ここでの知識ベースを構成する知識は問題領域に関する知識およびプログラミング言語に関する知識である。問題領域に関する知識は、実世界のモデルとなっており、プログラミング言語に関する知識は処理の

のモデルを与えていると言える。このようなシステムでは、ソフトウェアの開発を、それらのモデルの開発の問題に置き替えることが可能である。それらのモデルの記述言語は、人間系に近い程、非手続き性が要求され、処理系に近い程、手続き性が要求される。この2つの側面を満足する言語として、関数型言語、論理プログラミング言語などが、最近話題となっている。関数型言語は、表現能力には劣るが、モデル記述に適し、モジュラリティも良い。このため、複雑なプログラムを、各モジュールの単純な結合によって、容易に開発し得る。先に述べたデータ抽象化に基づく階層プログラミング法を採り入れることにより、プログラムの構造が非常に単純になり、開発が一層容易になる。このアプローチでの最大の問題は、関数型あるいは論理型言語が、履歴に依存する計算の記述に向いていない点である。この問題を避けるには、プログラミング言語を表現力豊かな、Algol流の言語にすることが必要である。その場合に、そのプログラムを、人間が書く方式と、システムが作り出す方式とが考えられる。前者のアプローチでは、仕様とプログラム間の検証問題が重要となり、多くの方法が開発されている。これらの各方法については、第2章で、詳しく論ずる。後者の、自動プログラミング方式は、前者に比べて、より一層困難な課題である。本報告書では、この分野の研究のいくつかの流れを手短かに紹介する。

知識ベースに基づくシステムにおいて、プログラミング言語に関する知識が必要であるが、この知識は、プログラミング言語の数学的な意味の扱いが必要となる。この点については、第3章ソフトウェア基礎論で述べる。さらに、そこではソフトウエア工学のより本質的な研究テーマであるアルゴリズムについて、複雑性、並列性などの観点から論ずる。

2．知的プログラミング環境

2. 知的プログラミング環境

2.1 プログラミングの一部としての仕様記述とその処理システム

2.1.1 プログラミングと仕様

仕様（specification）の持つ役割は大きく2つに分けられる。1つは、プログラム開発時に仕様作成者とそれをより現実化する者との間（両者は一致していてもよい）に厳密かつ明確なインタフェースを保つことである。他の1つはインプリメントされたものの正しさを調べるための基準としてである。

現在ソフトウェアの生産現場で使われている仕様は、ある程度標準化されてはいても、あいまいさを含む自然語や図表を利用したものが中心であり、上記の役割のいずれをも十分に果たしているとは言えない。

上記役割を果たすように仕様を記述することにより、プログラミングの各段階でまぎれ込み易い不完全さやあいまいさを取り除く効果があることはもちろん、下降型（top－down）プログラミングや段階的詳細化（stepwise refinement）プログラミングの考え方と組み合わせて仕様を記述しプログラミングを進めていくことは、信頼性あるソフトウェアのすみやかな開発に大いに役立つと言われる。このように、厳密な仕様の記述は、プログラミングの中の大きな要因として位置付けることができる。

ソフトウェアのライフサイクルとして図2－1のように表わされるモデルがある〔Ram 77〕。設計の各段階において上記役割を果たすべく仕様が作成されねばならない。一般に良い仕様を書くことは正しいプログラムを書く以上に難しい。プログラムレベルで様々なツールが開発されてきたように、仕様記述を支援してくれるシステムの必要性も大きい。ここでは仕様記述法とその支援システムという立場から、図2－1 bにおける仕様（仮にソフトウェア設計仕様と呼ぶ）に

焦点をあてて述べる。

a．データ処理システムの定義

　　ソフトウェアに対して期待されている
　　目標を明確に記述する

b．ソフトウェア方式の設計

　　システムの持つべき機能を明示し、部分システムの
　　機能、相互の関係を明確に記述する。

c．ソフトウェアの詳細設計

　　特定の計算機環境の下で最適と考えられる
　　アルゴリズム・データ構造を構成する

d・インプリメント

　　計算機による処理の可能な形でコーディングする

e．運用・保守

　　要求や環境の変化に応じた変更を行なう

（設計 ／ 開発）

図２－１　ソフトウェアのライフサイクル

2.1.2　形式的仕様の重要性

仕様記述がその役割を十分に果たすためには次の性質を持っていなければならない〔Lis 75〕。

(a)　形式性（ formality ）：数学的に厳密に定義された記法を用いること。

(b)　記述性（ constructibility ）：書き易いこと。

(c)　理解性（ comprehensibility ）：記述されている内容を読み取り易いこと。

(d)　最小性（ minimality ）：必要最小限の情報だけを含むこと。特に、操作手順ではなく機能を述べる。

(e)　広適用性（ wide　range　of　applicability ）：特定の領域だけで

なく広い範囲に適用できること。

(f) 拡張性（ extensibility ）：記述すべき内容の変更に伴う仕様の変更が容易であること。

仕様記述の困難さを考えると、これに

(g) テスト可能性（ testability ）：意図に合致した仕様になっているか否かをテストできること。

を加えることも重要と思われる〔Bal 79〕。

これらの性質の中でも、形式性は高信頼性を保証したいプログラム作成において最も重要な要素であると言うことができる。形式性を持たせて記述することにより、次にあげる利点が期待できる。

(a) 非形式的な仕様に比べてあいまい性が少ないので意志疎通用の媒体として有用である。

(b) プログラムの検証を厳密に行なう基礎となり得る。

(c) インプリメント前に仕様のレベルで無矛盾性、完全性、意図を満足しているか、などをチエックすることができ、設計段階での誤りをなくせる。

(d) 形式的（機械的）な取扱いが可能であり、上記(a)、(b)、(c)の計算機による実用的処理、支援が期待できる。

このような利点を持つ形式的仕様は、その記述法の提案、支援システムの開発という形で発展してきた。特に図２－１　bのプログラム仕様においてその発展は著しい。これは一つに抽象データ型というプログラム構造の重要な要素が仕様記述の単位として等識され研究されてきたからである。

2.1.3　形式的ソフトウエア設計仕様の記述法

仕様を記述する対象は、十分に抽象化されているものが望ましい。そのような単位として、機能抽象を与える手続き、データ抽象を与える抽象データ型の２つが明確に認識されるようになった。

手続き、抽象データ型という仕様記述の単位に対する代表的な仕様記述法は表

2－1のようにまとめることができる〔Lis 76〕。

それぞれについて以下で概観し、現在研究活動の盛んな抽象データ型の仕様記述法については次節で支援システムの代表例を選んでその特徴を述べる。

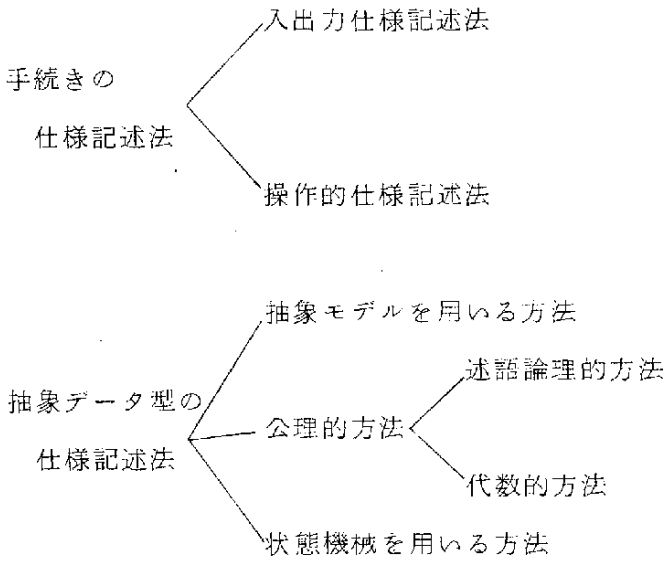


表２－１　仕様記述法の分類

a、手続きの入出力仕様記述法

この方法は、手続きの入力と出力が満たすべき性質を表明（assertion）の対として表現するものである。表明は、Hoareの記法〔Hoa 69〕を使うと

$$P\{M\}Q$$

のように表現される。P，Qがそれぞれ入力、出力の表明であり、通常プログラム変数やプログラム中の手続き名を含む一階述語論理で表わされる。Mが仕様記述の対象となった手続きである。

この仕様は、命題Pが成立していて手続きMが実行されたとき、もしMの実行が停止するのなら（又は、Mの実行が停止してかつ）Qが成立する。として解釈される。

この仕様記述法は、プログラムの正しさを含む諸性質を証明するために最

も初期に開発された方法であり、現在もプログラムの正当性証明システムや自動プログラミングを目指すシステム等で広く用いられている。例えば〔Man 78〕〔Els 79〕参照。

2つの正整数の最大公約数を求める手続きを例として入出力仕様を書くと次の次のようになる。

インタフェース

gcd(integer, integer) → integer

入出力仕様

in(x,y) {gcd(x,y)} out(x,y)

ただし

in(x,y) ≡ x>o ∧ y>o

out(x,y) ≡ gcd(x,y) | x ∧ gcd(x,y) | y

$\wedge^{\forall} i\ [i|x \wedge i|y \Rightarrow i \leq \gcd(x,y)]$

b、手続きの操作的仕様記述法

この方法は、記述すべき手続きの機能をプログラムとして示す。しかしそれを記述する言語はインプリメント用言語とは異なり、可能な限り単純かつ手続きの機能の本質面を表現できるものでなければならない。

そのような言語としてMcCarthy による再帰方程式が利用できる。再帰方程式は、静的な関係式とも、動的な再帰プログラムとも見ることができる単純な構造を持った言語である。

GCDを求める手続きの再帰方程式を用いた仕様記述例は以下のようになる。

インタフェース

gcd (integer ; integer) → integer

方程式

gcd(x,y) = if $x \leq o \vee y \leq o$ then error

else f(x,y,min(x,y))

$$f(x,y,z) \leftarrow \text{if } rem(x,z) = 0 \ \wedge \ rem(y,z) = 0$$

$$\text{then } z$$

$$\text{else } f(x,y,z-1)$$

c、抽象モデルを用いる抽象データ型の仕様記述法

この方法は、すでに形式的仕様が与えられ、性質もよく知られた他の抽象モデルを用いて、仕様記述すべき抽象データ型を表現し意味付けるものである。抽象データ型を特徴付ける各演算は1つの手続きとして、抽象モデル上の演算を用いて仕様が記述される。

d、抽象データ型の公理的仕様記述法

この方法は、仕様記述すべき抽象データ型を特徴付ける演算の間に成立する関係を公理として述べることにより、そのデータ型の振舞いを記述する。公理を述べる言語の違いにより、述語論理的方法と代数的方法の2つの代表的方法がある。

与えられた演算によって定められる表現（式又は項と呼ばれる）の集合が得られるが、この集合を公理によって定められる同値関係によって分類した商集合が、抽象データ型の値の集合とみなされる。

代数的方法では、抽象データ型の任意の値は演算の有限・適用によって構成されることを前提としており、一般的な記述を行なう述語論理的方法に比べ、表現が簡潔化され理解性を増している。現在研究活動の盛んな分野である。

e、状態機械を用いる抽象データ型の仕様記述法

この方法は、抽象データ型の要素を状態機械の状態に対応させ、演算の実行前後の間に成立する状態値の関係を公理として記述するものである。その意味で公理的方法に近い。

2.1.4 形式的プログラム仕様の具体例とその処理システム

手続きの仕様・検証の方法、システムについては、以前から研究されよく知ら

れている（例えば〔Man 78〕参照）ので、ここでは比較的新しく仕様記述単位として認識された抽象データ型に関して述べていく。

a、Alghard －抽象モデルの利用－〔Wul 76〕

Alphard はプログラミング言語であるが、その抽象化機構である form は仕様部と実現部に分かれ、それらの対応を調べることによる検証を１つの目標としている。

仕様部では、記述すべき対象に対応する抽象モデルを記述し、抽象データ型を特徴付ける各演算に対してはこの抽象モデル上での入出力仕様が書かれる。

実現部では、記述すべき対象の具体表現、具体表現から抽象モデルへの写像（rep と書く）、具体表現を用いた各演算の入出力仕様が書かれる。

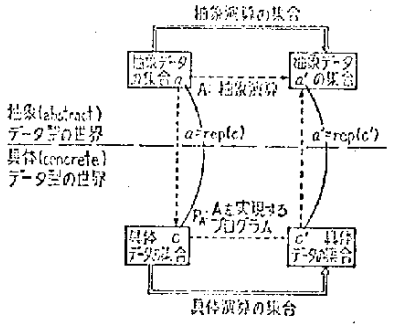


図２－２　抽象データ型の検証の概念図

```
form  istack (n : integer) =
  beginform
  specifications
    reguires  n > 0 ;
    let  istack = <·····xi·····> where  xi  is  integer ;
                          ↑
                          └── 抽象モデル
    invariant  0 ≦ length ( istack )≦n ;
    initially  istack = nulleq ;
```

```
function
    push ( s:istack, x:integer )
        pre   0 ≦ length(s) < n      ← 抽象モデルを用いた
        post  s = s'~x                 演算の入出力仕様
    ⋮
representation
  uniqne v : vector ( integer,1,n ), sp:integer  init
                sp← 0;
  rep ( v,sp ) = seq ( v,1,sp)
  invariant  0 ≦ sp ≦ n        ← 具体表現から抽象モデルへの写像
    ⋮
implementation
  body  push  out (s.sp = s.sp'+1  ∧          ← 具体表現に
                   s.v = α ( s.v' ,s.sp,x) )    よる演算の入
        = mt,normal ::                          出力仕様
              ( 実現コード )
          otherwise  :: FAIL
    ⋮
endform
```

図２－３　スタックを定義するAlphardプログラム

仕様部と実現部との関係は、抽象データ型の検証に関する基本的な概念図２－２を見ると理解し易い。図中の破線で示される変換が可換であることを示すことが、抽象データ型の検証に対応する。即ち、具体表現を用いた抽象データ型の正しさが証明されることになる。

Alphard によるスタックの記述例を図２－３に示す。そこに示される具体

表現から抽象モデルへの写像、各演算の抽象モデル上、具体表現上での入出力仕様等を用いて、図２－２の破線部分が可換となるための条件を作ることができる。それは一階述語論理における命題であり、この命題を証明することが、抽象データ型の実現の正しさを証明することになる。

Alphard のコンパイラ、及びヴェリファイアは現在開発中である。

b．ｔ－述語論理による公理的方法－　〔Nak 77〕

階層的で証明可能なプログラム作成の道具としてｔ言語が設計され、現在その支援システムが開発されている。

ｔ言語での基本的な単位はモジュールと呼ばれ、type module，procedure module，theory module，sype module の４種類がある。各モジュールは仕様を与える抽象部分（ theory ）とその実現部分とが明確に区別されている。あるモジュールが他のモジュールの上に作成される場合には抽象部分のみに依存することになる。

ｔ言語の特徴として、

- データ抽象、手続き抽象に共通な統一的証明法を与えている。
- 共通の性質のみに注目したsype theory を作り、その上に type module や procedure module を作ることができる。

があげられる。

(a)

```
interface sype FIELD;
  thru fn ZERO:→ @ as 0
          ADD:(@,@)→ @as @+@;
            ⋮
          MULT:(@,@)→@as @*@;
end interface
specification sype FIELD;
  var X,Y,Z:@;
```

```
  axiom 1 : 0+X = X;
      :
end  specification

interface  type  RATIONAL;
  is  FIELD
  and  thru  fn  ORDER : (@,@)→ bool  as @ ≦ @;
end  interface

realization  type  RATIONAL;
  rep = record 1,2: int  end;
(b) fn  ADD(X, Y:  rep) return (Z: rep)
  <= var I: int;
      Z . 1 := X . 1*Y . 2+X . 2*Y . 1 ;
      Z . 2 := X . 2*Y . 2 ;
      I :=  abs( gcd (Z . 1,Z . 2));
      Z . 1 := div( Z.1,I);
      Z . 2 := div( Z.2,I)
  end  fn
     :
end  realization

pre-spec type RATIONAL;
  rep = record 1,2: int  end;
  var  X,Y: rep
  axiom 1: realize(X) ⊃ X.2 ≒ 0 ∧ gcd(X.1, X.2) = 1;
        2: equal(X,Y)  X.1 ≡ Y.1 ∧ X.2 = Y.2;
     :
end  pre-spec
```

図２－４　ｉ言語によるプログラム例

図２－４(a)は、数学でいう体（field）の性質を抽出したsype modnle FIELDである。このsype上の関数がinterface部分で提示され、それらの公理系がspecification部分に与えられている。図２－４(b)はFIELDを部分theoryとして持つtype RATIONALを定めており、これはFIELDの持つ諸性質にORDERという関数を追加したものであり、２つの整数の対という表現で実現されている。

sype上で証明できる命題は、それを部分theoryとして含むあらゆるtype上でも成立する。この性質により、sypeを用いてプログラムを積み上げ方式で作成していくことが保証される。

interface部分で公理として書かれている内容を実現部から直接証明することは一般に困難である。ιでは公理と実現部との中間にpre-specと呼ぶ部分を置き、証明を２段階に分けることで機械実現の可能な形にしている。即ち、pre-spec部分には実現部の入出力仕様が公理として書かれ、これはHoareの証明システム〔Hoa 69〕により証明される。又、interface部の各公理は代数的変換により論理式に変換され、この論理式はpre-spec部分の公理を用いて証明される。

ιシステムは、仕様、実現の記述及び検証を並行して行なうことの利点を生かすように設計されている。又、対話的使用法の有効さを考慮して、エディタ機能を組み込んだ対話型のプログラム作成／変更サブシステムを持っている。このサブシステムはモジュール単位に取り扱うことができ、完成したモジュールはモジュール間関係とともに内部形式の形でデータベースに登録される。

ιシステムはまた、モジュール仕様を実現部が満たしていることを対話的に検証するVerifierサブシステム、ι論理用対話的定理証明サブシステムProver等からなる。現在Lispにより開発中ということである。

pre-spec部分はι言語の特徴であるが、定理証明システムの能力不足を人間側で補っているものでもある。記述する側は、ιの検証法と記述しようとしているものに対する深い理解が必要である。対話型システムはこの難点を補

ってくれるであろうが、システムの完成が待たれる。

c．ＯＢＪ－代数的記述による公理的方法－〔Gog 79〕

代数的仕様記述では、演算の間に成立する関係を公理として述べることにより、データ型の振舞いが記述される。従って、その検証は、具体データ型を使って抽象データ型を実現するプログラムが、具体データ型はそれ自身の仕様を満たすことを仮定して、確かにその抽象データ型の公理を満たすことを示すことになる。その手段として、例えば、具体データ型の仕様として書かれた公理及び再帰方程式の形で書かれた実現部とを、書換え規則とみて変形していく方法が考えられる。

代数的仕様はこのように検証法が比較的簡単であることが１つの特徴である。

公理を書換え規則として捉える見方を更に推し進めると、代数的方法で記述された仕様に対して計算したい関数及びその引数を与えれば、公理を入力データに適用し書換え規則として簡単化の方向に変形していくことにより、最終値が得られる。即ち実行されたことになる。

ＯＢＪはこの考え方に基づき、仕様言語かつプログラミング言語として設計された先駆的言語であり、システムも初期の版ＯＢＪ－０が既にインプリメントされている。現在は、モジュール化機能等を強化したＯＢＪ－Ｔがインプリメント中である。

ＯＢＪの開発においては、仕様を正しく書くことの困難さが十分に認識され、仕様を正しく作るのに役立つ機能の実現が重視されている。上記の意味で仕様をテスト的に実行させること、エラーが発生した時にエラーの源をユーザに知らせるエラー式等のメカニズム、型チェック機能、等がそのような機能である。

図２－５はＯＢＪで記述された整数を要素とするリストの仕様記述　及びテスト実行例である。

代数的仕様の公理を書換え規則として捉えることはプログラム作成の自動化の有力な方法でもある〔杉山 80〕。その１つの欠点としては、変換の適用順序等に戦略を組み込みにくいことがあげられる。大きな問題に対してどの程度

の実用性を持たすことができるかが興味深い。

```
OBJ      LIST-OF-INT
SORTS    LIST / INT
OK-OPS
    --  : LIST  LIST → LIST (ASSOCIATIVE)
    -   : INT → LIST
JBO
RUN  1  2  3  NUR
AS LIST : (1 2 3)
OBJ    LIST2
SORTS / LIST INT
OK-OPS
   HEAD - : LIST → INT
   TAIL - : LIST → LIST
ERR-OPS
   NO-TAIL! : → LIST
VARS
   I : INT
   L : LIST
OK-EQNS
   AS INT (HEAD(I,L)=I)
   (TAIL(I,L)=L)
   AS INT (HEAD(I)=I)
ERR-EQNS
   (TAIL(I)= NO-TAIL!)
JBO
RUN HEAD (2,3,4+HEAD(6)) NUR
```

```
Which sort would you like it ?
> AS INT
AS INT : 2
AS INT RUN HEAD (2,3,4+ HEAD TAIL 6) NUR
AS INT : >> ERROR << (HEAD (2,3,4+(HEAD NO-TAIL!)))
```

図２－５　ＯＢＪでの記述例とその実行例

d、ＳＰＥＣＩＡＬ　－状態機械モデルを用いる方法－　〔Rob 77a〕

Parnasは状態機械モデルを用いた仕様記述法を提案した〔Par 72〕が、そこで用いた言語は形式的に定義されたものではなかった。ＳＰＥＣＩＡＬは、ＳＲＩ階層的設計法〔Rob 77b〕に基づきParnasの記述法を形式的に定義した言語である。

ＳＰＥＣＩＡＬで記述される仕様は、基本的には、状態値を表わすＶ関数、状態を変化させるＯ関数、状態を変化させかつ状態値を持つＯＶ関数の記述から成る。これら関数の記述は、Ｖ関数の場合には返すべき値と例外条件、Ｏ関数の場合にはその効果を実行前後の状態値の関係及び例外条件とによりなされる。

Ｏ関数の効果を記すための様々な演算子が用意されており、また、モジュールの外からは参照できないhidden関数を使うことを許している。

ＳＰＥＣＩＡＬは形式的記述言語であるけれど、モジュール単位で見た場合、その書き易さや理解し易さの点ですぐれていると言えよう。一方、形式的ではあるが機械実現可能な形で検証法を構成することは困難なようである。検証を行なうこともＳＰＥＣＩＡＬの目的の１つであるが、未だ十分強力な検証システムは作成されていないようである。いくつかの支援ツールによる設計支援の道具として使われてきている。

2.1.5 まとめ

プログラミングの一部として仕様記述を行なうことは信頼性あるソフトウェア作成にとって極めて重要であるが、仕様記述の困難さを考えると計算機支援は不可欠である。形式的なソフトウェア設計仕様の記述法及び計算機支援についてはいくつかの例を見てきた。それらをより充実させるとともに、要求仕様等の前段との統一的な結び付きが期待される。

2.2 自動プログラミング

2.2.1 自動プログラミング方式

仕様からプログラムへの変換は、What から How への変換であり、その質的な差は大きい。その変換は、一般には、高度な知的思考活動を必要とし、その自動化は困難な課題である。その中でも、比較的簡単なアプローチは、すでに解法が知られている問題の解法を辞書化し、問題が与えられたときは、その辞書を参照して解法を知る方法で、このような辞書はアルゴリズム・バンクと呼ばれている。その場合でも、実際に必要とするプログラムと辞書中にあるアルゴリズムの記述には、大きな隔りがある。さらに、すべての問題の解法を辞書化するわけでなく、プログラムの部品の単位での辞書化が必要である。そのとき、各部品の結合の問題が生じてくる。これらの問題に対する適切な解を求めることが、この方法の成否を決定づけるであろう。

第2の、より困難な課題は、解法自身を発見することである。これは、人工知能の分野で、問題解決の1つとして研究されてきた領域である。問題が単純で、基本演算を高々十数個組み合わせればプログラムが作れる場合には、すべての組合せを調べるような強引な方法によっても解を得ることが可能である。また、通常は、発見的な方法によって、場合の数を大巾に減らすことも可能である。しかしながら、問題が多少複雑になると、そのような方法では手に負えない。問題の

複雑さを本質的に減少させる工夫が必要である。その基本的な考えは、問題をより単純ないくつかの問題（サブ・ゴール）に還元する方法で、問題解決の分野ではProblem Reduction Methodとして知られている。先に述べたデータ抽象化による階層化は、このProblem Reductionを行う一つの方法であると考えられる。ここで、2つのアプローチが考えられる。その1つは、抽象データ型を人間が与えてやるアプローチである。問題解決を人間と機械の協調により行うという考え方に立てば、この方式は有望である。他のアプローチは、サブ・ゴールの設定自身を機械に行わせる方法である。このような流れを追求しているいくつかの研究もみられる。たとえば、〔間野 1979〕のデータ・フロー解析に基づくプログラム・スキーマ（プログラムの大まかな構造）の選定によるProblem Reductionなどである。

自動プログラミング方式の問題点の最大のものは、検証方式の第1の問題点と同様、仕様記述自身の困難性である。アルゴリズム・バンク方式の問題点は、出来たプログラムの質が保証されない点と、プログラムの理解が困難な点である。プログラムの質を保証するには、効率に関する仕様も考えなければならないが、現状ではそこまで行っていない。解法生成方式の問題点は、解法が未知の問題を解くのに、ほんとうに有効かどうかが不明である点である。さらに、自動プログラミング方式が、次節で述べる仕様実行方式あるいはプログラム変換方式に比して、どれほど有効であるかも、十分に吟味してみる必要がある。

2.2.2 仕様実行方式

プログラミング言語自身を高水準化することによって、仕様言語の水準にまで持ち上げるのが本方式である。現在、2つの流れがある。第1の流れは、非決定性手続きの直接実行を行う言語で、PLANNER, PROLOGなどがこれに属する。その基本的な考えは、問題解決の基本的機構であるProblem Reductionの自然な記述を可能にすることである。サブ・ゴールの設定がいくつか考えられるとき、非決定的な計算機構により、適当なサブ・ゴールの選択と、そ

れが誤りであったときの制御が自動的になされ、ユーザのプログラムでそれらの制御を行う必要がない。この方式の利点は、検証方式で必要であった仕様とプログラム間の検証が不要な点である。また、計算と検索が同一表現で行われるという長所もある。欠点は、問題解決過程がシステムによって制御されているので、きめの細かい能率のよいプログラムを作れない点である。たとえば、サブゴールの選択を誤ったときには、自動的に後戻り制御が行われるが、その戻り方は常に最近の選択点とするのが通常である（この点に関しては、最近、Truth Maintenance System の研究から、もっと効率良い方法が提案されている）。制御構造の問題は、本質的には、並行プログラムの枠組の中で論ずるべきであり、そこでの理論の整備が望まれる。

第2の流れは、関数型プログラミング、あるいは、代数に基づくものである。この方法は、抽象データ型による階層プログラミングと密接に結びついている。

参 考 文 献

〔Ram 77〕 C.V.Ramamoorthy,H.H.So

Software requirements and specification : atatus and perspectives,

Tutovial : Software Methodology IEEE 1978（邦訳あり）。

〔Lis 75〕 B.H.Liskov,S.N.Zilles

Specification techniques for data abstractions,

IEEE SE−1,1 1975

〔Bal 79〕 R.Balzer , N.Goldman

Principles of good software specification and their inplications for specification langnage

Proc. of Specifications of Reliable Software 1979

〔Lis 76〕 B.H.Liskov, V.Berzine,

An appraisal of program specifications,

MIT Lab. for Comp. Sci. CSG-Memo 141 1976

〔Hoa 69〕 C.A.R.Hoare

An axiomatic basis for computer programming

CACM 12,10 1969

〔Man 78〕 Z.Manna, R.Waldinger

The logic of computer programming

IEEE SE−4,3 1978, 邦訳あり、

〔Els 79〕 R.Elschlager, J.Phillips

Automatic programming

Stanford Univ. HPP−79−24（STAN−CS−79−758）

〔Wul 76〕 W.A.Wulf, R.L.London, M.Shaw

An introduction to the construction and verifica−

tion of Alphard programs
IEEE SE-2.4 1976

〔Nak 77〕 R.Nakajima, M.Honda, H.Nakahara
Describing and verifying programs with abstract data types
Proc. IFIP Working Conf. on Formal Description of Programming Concepts 1977

〔Gog 79〕 J.A.Goguen. J.J.Tardo
An introduction to OBJ
Proc. Specifications of Reliable Software 1979

〔杉山 80〕 杉山、谷口、嵩
代数的仕様記述言語とその部分言語としての関数的プログラミング言語
信学技報 AL 79-99

〔Rob 77a〕 L.Robinson, O.Roubine
SPECIAL - A specification and assertion language
SRI Tech. Report CSL-46 1977

〔Par 72〕 D.L.Parnas
A technique for software module specification with examples
CACM 15, 5 1972

〔Rob 77b〕 L.Robinson. K.L.Levitt
Proof techniques for hierarchically structured programs
CACM 20, 4 1977

〔間野 79〕 間野暢興
データフローの概念に基づくプログラム合成方式
信学技報 AL 79-70

3．ソフトウェア基礎編

3. ソフトウェア基礎論

3.1 プログラムの意味論

この節では、ソフトウェア工業の基礎として今後ますます重要性の増すと思われるプログラムの意味論の中から、次の3つの話題に焦点をしぼって解説する。

- Floyd-Howre 流の意味論と
  Dynamic Longic
- General Recursion Theory と
  Denotational Semantios
- 代数的仕様とその意味

この内、代数的仕様とその意味に関しては抽象データ型の代数的仕様記述法から発展しつつある"代数的プログラミング"との関連から、マシン理論報告書、第5章2節で解説しているので、そちらを参照されたい。

3.1.1 Floyd-Hoare 流の意味論と Dynamic logic

(1) はじめに

プログラミング言語の意味をいかなるものと規定すべきかという点については様々な考え方がある。

主なものとして

① Operational

② Axiomatic

③ Denotational

④ Algebraic

の4つの方法があげられる。

①の Operational な方法というのは抽象的な machine を定義し、その上への interpreter を構成することによりプログラム言語の semantics を定

義しようとするものである。

③のDenotational な方法というのはプログラムに対し、その計算過程全体のなす集合を割当てることによりsemantics を定義するもので λ－Calculus のような関数型言語を媒介することによりプログラムの構文に対しても直接 semantics を与えることができる。

④のAlgebraic な方法とはAbstract Data type を代表とするものであるが、これはプログラムをData type とその間の関数関係ととらえるもので、プログラムのsemantics は各Data type の semantics を規定することで定まる。各Data type の semantics はそのData tyqe への関数全体として定義される。この点は、はなはだ抽象的に見えるが、そうむづかしいことではない。

ここで取上げようとする②のAxiomatic な方法というのはプログラムをInput world で成立するlogic をOutput world で成立するlogic に変換する変換器とみなすものである。従って厳密にsemantics を導入するためにはPratt の Dynamic logic で行なわれたようにmodal logic のsemantics にたよるのが正解であろう。

本方法の利点は

① プログラムの正当性の証明が厳密に行なえる公理系をそなえている。

② Dynamic logic で行なわれたように並列処理系のsemantics が規定できる。

の2点である。

以下に簡単にFloyd Hoare logic 及びDynamic logic について解説したい。

(2) Floyd Hoare Method

Floyd Hoare 流の考え方ではプログラムの正当性の証明を

1. 入力データの満たす条件をP
2. プログラムSequence をS

3. 出力データの満たす条件をQ

とするとき

プログラムSequence Sが確かにPを満たすものをQを満たすものに変換しているか調べることであると見なしている。

この条件を

$$P\{S\}Q$$

と書くものとして、プログラムSの構文それぞれについて公理を設けると、この公理系での証明としてプログラムの正当性の証明が位置づけられる。

次にHoare による公理系の一部を紹介しよう。

R1 assignment

$$P\{X:=T\}P$$

R2 consequence

$$P \Rightarrow Q,\ Q\{A\}R \vdash P\{A\}R$$

$$P\{A\}Q,\ Q \Rightarrow R \vdash P\{A\}R$$

R3 conposition

$$P\{A\}Q,\ Q\{B\}R \vdash P\{A;B\}R$$

R4 conditional

$$P \wedge C\{S_1\}Q,\ P \wedge \neg C\{S_2\}Q$$

$$P\{\text{if } C \text{ then } S_1 \text{ else } S_2\}Q$$

R5 iteration

$$P \wedge C\{S\}P \vdash P\{\text{while } C \text{ do } S\}P \wedge \neg C$$

R6 substitution

$$P(X,V)\{p(X;V)\}Q(X,V)$$

$$\vdash P(A,E)\{p(A;E)\}Q(A,E)$$

R7 procedure declaration

$$\text{procedure } p(X,V);B,\ P\{p(X,V)\}Q \Rightarrow P\{B\}Q$$

$$\vdash P\{p(X,V)\}Q$$

R 8　adaptation

$$I(X,V)\{p(X,V)\}O(X,V)$$
$$\vdash {}^{\exists}K'\ I(X,V)\wedge{}^{\forall}X'\ (O(X',V)\Rightarrow Q(X',V))$$
$$\{p(X,V)\}Q(X,V)$$

R 9　function declaration

$$\text{function } f(X,V);B,I\{p(X,V)\}O|^{f}_{f(X,V)}\Rightarrow I\{B\}O$$
$$\vdash I\{p(X,V)\}\ \ O|^{f}_{f(x,v)}$$

R 1 0　function call (in assignment)

$$P=I_1\wedge(O_1\Rightarrow\cdots\cdots\Rightarrow Q|^{X}_{E})$$
$$\vdash P\{X\leftarrow E\}Q$$

但し、I_1 、O_1 ……ＩＮ、ＯＮはＥに於ける関数呼出しの activation の順序に応じその関数呼出しの際のＩＯＯ条件を述べたもの

R 1 1　function call (in if statement)

$$P\Rightarrow I_1\wedge(O_1\Rightarrow\cdots\cdots(B\Rightarrow L)\cdots),L\{S_1\}Q$$
$$P\Rightarrow I_1\wedge(O_1\Rightarrow\cdots\cdots(\neg B\Rightarrow M)\cdots),M\{S_2\}Q$$
$$\vdash P\{\text{if } B \text{ then } S_1 \text{ else } S_2\}Q$$

R 1 2　function call (in while statement)

$$P\Rightarrow I_1\wedge(O_1\Rightarrow\cdots\cdots(B\Rightarrow L)\cdots),L\{S\}P$$
$$P\Rightarrow I_1\wedge(O_1\Rightarrow\cdots\cdots(\neg B\Rightarrow Q)\cdots)$$
$$\vdash P\{\text{while } B \text{ do } s\}Q$$

R 1 3　procedure declaration

$$X=X_0\wedge I(X,V)\{p(X,V)\}O(X,X_0,V)\Rightarrow P\{S\}Q,$$
$$X=X_0\wedge I(X,V)\{p(X,V)\}O(X,X_0,V)$$
$$\Rightarrow X=X_0\wedge I(X,V)\{V_0:=V;B(X,V);V:=V_0\}O(X,X_0,V)$$
$$\vdash P\{\text{procedure } P(X,V);B(X,V);S\}Q$$

R 1 4　procedure call

$X = X_0 \wedge I(X_0, V)\{p(X, V)\}O(X, X_0, V)$

$\vdash {}^{\exists}K(I(A,E) \vee {}^{\forall}X'(O(X', A, E) \Rightarrow Q(X', E)))\{p(A,E)\}$

$Q(A,E)$

但しKはX，A，E，Qでしばられている変数のリスト

このlogic で実際にProgram の正当性を証明してみよう。

例1.　Procedurc F (var X : interger) ;

initial $X = X_0$;

entry $X \geq 0$;

exit $X = X_0$ / ;

begin

if $X = X_0$　then　X := 1　else

begin

Y := X − 1 ;

F(Y) ;

X := Y * X

end

end .

なるプログラムが与ふられたとして

$X = X_0 \wedge X \geq 0\{F(X)\}X = X_0$ /

を示すこととする。（ /は階乗を表わす。）

i)　$1 = X_0$ /{ X := 1 } $X = X_0$ /　（代入公理）

ii)　$X = X_0 \wedge X \geq 0 \wedge X = 0 \Rightarrow 1 = X_0$ /　（階乗計算の性質）

iii)　$X = X_0 \wedge X \geq 0 \wedge X = 0$ { X := 1 } $X = X_0$ /　（ i と ii の結果）

iv)　$Y * X = X_0$ /{ X := Y * X } $X = X_0$ /　（代入公理）

v)　$Y \geq 0 \wedge {}^{\forall}Z(Z = Y/ \Rightarrow Z * X = X_0/)\{F(Y)\}Y * X = X_0/$　（手続き呼出しの公理）

vi) $X-1 \geq 0 \wedge^{\forall} Z\ (Z=(X-1)\ /\Rightarrow X*Z=X_0\ /)$

$\{Y:=X-1\}\ Y \geq 0 \wedge^{\forall} Z\ (Z=Y\ /\Rightarrow Z*X=X_0\ /)$

（代入公理）

vii) $X=X_0 \wedge X \geq 0 \vee X \neq 0$

$\Rightarrow X-1 \geq 0 \wedge^{\forall} Z\ (Z=(X-1)\ /\Rightarrow X*Z=X_0\ /)$

（階乗計算の性質）

viii) $X=X_0 \wedge X \geq 0 \wedge X \neq 0$

$\{Y:=X-1\}\ Y \geq 0 \wedge^{\forall} Z\ (Z=Y\ /=Z*X=X_0\ /)$

（viとviiより）

ix) $X=X_0 \wedge X \geq 0 \wedge X \neq 0$

$\{Y:=X-1\ ;\ F(Y)\ ;\ X:=Y*X\}\ X=X_0\ /$

（iv、v、viiiより）

x) $X=X_0 \wedge X \geq 0$ { if $X=0$ then $X:=1$ else

begin

$Y:=X-1$;

$F(Y)$; $X:=Y*X$

end } $X=X_0$ /

（条件文の公理とiii、ixより）

以上より

$X=X_0 \wedge X \geq 0\ \{F(X)\}\ X=X_0$ /

が示された。

さてここでプログラム言語の各構文に対して与えられた公理はプログラム言語のその構文の意味を与えていると見ることができる。しかし、通常のlogicに於けるようなモデル解釈が与えられないのが難点であり、従って公理系の健全性（完全性等々）について述べることができない。Pratt 等の開発したDynamic logic はその点で大きな前進といえる。

(3) Dynamic logic

ここでは簡単のためRegular First order Dynamic logicの解説をする。（以下ＲＧと略）ＲＧのＷＦＦは次のように定義される。

可算個の関数記号 f_0 、f_1 ……、述語記号 p^0 、p^1 ……、及び変数記号 x_0 、x_1 ……が与えられているとき

項とは

i) 変数 xi は項

ii) f がn変数関数記号で α_0 …… α_n が項のとき f（α_0 …… α_n）は項

によって定義されるものとする。

このとき regular program の集合ＲＧとDynamic logic の整式ＷＦＦを次のように定義する。

i) 変数 x と項 e に対し　$x \leftarrow e \in RG$

ii) atomic formula　$P \in WFF$

iii) α , $\beta \in RG \Rightarrow \alpha ; \beta$, $\alpha \cup \beta$, $\alpha^* \in RG$

iv) program 修飾即ち $<\alpha>$ 部分を含まない任意のＷＦＦ　Ｐに対し

$P? \in RG$

v) Ｐ , Ｑ∈ＷＦＦ、$\alpha \in RG$のとき

$$P , (P \vee Q) , \exists x P , <\alpha> P \in WFF$$

Regular first order Dynamic logic　ではプログラムを正則式とif文で書かれるものに限定している。

これはプログラムをDo 文とＩｆ文の constract に限定している反面 $\alpha \cup \beta$ なる構文を許すことによって非決定的乃至は並列な実行を許容している点でプログラム言語としては、かなり広いものである。（再帰的プログラムも書けるようなcontext free dynamic logic についても研究が進んでいる。）

また $<\alpha>$Ｐ（或るプログラム α の実行列があってと読む）という述語を導入することによってプログラムの実行系列に対するquantification を許容

している。これは並列実行や非決定的実行に対しても実行の正当性を表明する文を書けるということを意味している。

Dynamie logic のＷＦＦに対して semantics を導入することができる。集合Ｄ上の関数と述語をＤＬの関数記号と述語記号に assign するものをＩとし、状態と呼ぶことにする。またこのようなＩの全体Γを possible states と呼ぶ。

このときDynamic logic のプログラムαに対してはΓ上の２項関係$m(\alpha) \subset \Gamma \times \Gamma$をまたＷＦＦ Ｐに対しては$\Gamma$の部分集合を対応させるものとするわけである。即ちモデル解釈を次のように定める。

Γがいま集合Ｄ上の或り得べき状態全体としたとき

i) $m(x \leftarrow e) = \{(I, J) \mid J = [e_I / x]I,\ I, J \in \Gamma\}$

但し$[e_I / x]I$は項ｘに対しe_Iを対応させる他はＩと同じ

ii) $m(P?) = \{(I, I) \mid I \models P\}$

但し$I \models P$はＩでＰが真であることをあらわしている。

iii) $\alpha, \beta \in RG$のとき

$m(\alpha ; \beta) = m(\alpha) \circ m(\beta)$

$m(\alpha \cup \beta) = m(\alpha) \cup m(\beta)$

$m(\alpha^*) = m(\alpha)^*$

iv) atomic formula $P(e_1 \cdots\cdots e_K)$に対しては

$I \models P(e_1, \cdots\cdots, e_K) \equiv_{fD}$ $P_I(e_{1_I}, \cdots\cdots, e_{K_I})$が真

v) $\alpha \in RG$，Ｐ，Ｑがｗ ＦＦ，ｘが変数のとき

$I \models \neg P \equiv_{fD}$ $I \models P$でない

$I \models (P \vee Q) \equiv_{fD}$ $I \models P$ か $I \models Q$

$I \models \exists x P \equiv_{fD}$ $\exists d \in D$ $[d / x] I \models P$

$I \models \langle\alpha\rangle P \equiv_{fD}$ $\exists J \in \Gamma$ $(I, J) \in m(\alpha) \wedge J \models P$

と定めＤＬの任意の整式ＰがΓの任意の状態Ｉに対し$I \models P$のときΓはＤＬのモ

モデルであるという。

さて〔α〕P$=_{fD}\neg<\alpha>\neg$PとするとHoareのP｛α｝Qなる述語はDLでは（P$\Rightarrow$〔α〕Q）と書けることになる。

Dynamic logicはモデル解釈が定義されるので公理系の完全性等についても明確に述べることが可能となった。

DLの一つの公理系として次のものが知られている。

公理

a）命題論理の公理

b）〔x$\leftarrow$e〕P$\equiv$P$^{e}_{x}$

c）〔Q ?〕P$=$Q$\Rightarrow$P

d）〔α ; β〕P$=$〔α〕〔β〕P

e）〔$\alpha\cup\beta$〕P$=$〔α〕P$\wedge$〔β〕P

推論規則

a）P , P$\Rightarrow$Q $\vdash$ Q

b）P$\Rightarrow$Q $\vdash$ 〔α〕P$\Rightarrow$〔α〕Q

P$\Rightarrow$Q $\vdash$ $\exists_{x}$ P$\Rightarrow\exists_{x}$ Q

c）P $\Rightarrow$〔α〕P $\vdash$ P$\Rightarrow$〔α^{*}〕P

d）P（n+1）$\Rightarrow<\alpha>$P(n) $\vdash$ P(n)$\Rightarrow<\alpha^{*}>$P(0)

領域Dとしてarithmatic universeと呼ばかる集合Aをとるものとする。Aでは

ⅰ）＋，・が通常の意味で定義されている。

ⅱ）0，1は0変数の関数として定義されている。

ⅲ）nat_{1}(d)$\equiv$“dは自然数”なる述語natが定義されている。

ⅳ）或るR（x，i，y）なる述語が定義されており

$\forall x_{1}\cdots x_{n}\exists y\forall x\forall i$（（nat(i)$\wedge$n$\geq$i）$=$(R(x,i,y)$=x=$xi））

が成立（R（x，i，y）はxがyのi番目の数と読む）

が成り立っている。

このような領域A上では上記の公理系が完全であることが証明されている。

ＤＬではプログラムの意味を関数mで与えている。従って一般のプログラミング言語にＤＬで意味を与えるにはそれぞれの言語の構文をＤＬのプログラムの構文に翻訳する規則を作ればよいことになる。

Dynamic logic で Program の正当性の証明を行なう例をあげてみる。

例2　$f(Z)=$ if $Z\geq 100$ then $Z-10$ else $f(f(Z+11))$

という recursive に定義された関数が

$$f(Z)=\begin{cases} Z-10 & Z>100 \\ 91 & 100\geq Z \end{cases}$$

という関数を実現していることを示そう。

プログラムを正則文で書かねばならないので

α：$100\geq Z$？；$Z\leftarrow Z+11$；$y\leftarrow y+1$

β：$Z\leftarrow Z-10$；$y\leftarrow y-1$

γ：α^{*}；$(100<Z\wedge y\neq 1)$？；β

とおくと$<\gamma^{*}>$が recursive definition にそったＤＬのプログラムである。

すると証明すべきことは

$$(101\geq Z\wedge y=1)\Rightarrow<\gamma^{*}>(Z=101\wedge y=1)$$

である。$P(n)=y>0\wedge Z>0\wedge 111\geq Z\wedge n=90-Z+11y$

とおくとき

ⅰ）　$P(n)=<\gamma^{n}>(Z=101\wedge y=1)$

ⅱ）　$(101\geq Z\wedge y=1)\Rightarrow\exists_{n}P(n)$

が示されれば十分

$P_1(n)\equiv 100<Z\wedge P(n)$

$P_2(n)\equiv 100\geq Z\wedge Z\geq 90\wedge P(n)$

$P_3(n)\equiv Z<90\wedge P(n)$

とおくと$P(n)\equiv P_1(n)\vee P_2(n)\vee P_3(n)$　　P_1、P_2 についてはⅰ）ⅱ）は明ら

かであるからP_3 について ｉ) を示せばよい。

$$P_3\ (n+1) = <\alpha> P_3(n)$$

$$\equiv P_3(n+1) = <\alpha^*>(1<y \wedge 100<Z \wedge 121 \geq Z \wedge n = 89-Z+11y)$$

である。

$$P_3(n+1) \Rightarrow <\alpha^*;\alpha;\alpha>(1<y \quad 100<Z \quad 121 \geq Z \quad n=89-Z+11y)$$

について示せば十分であろう。

これは

$$P_3(n+1) \Rightarrow <\alpha^*>(Z>78 \quad 89 \geq Z \quad n=89-Z+11y)$$

$$Q(m) \equiv y>0 \wedge Z<90 \wedge n=89-Z+11y \wedge Z>0$$

$$m = floor((100-Z)/11)-1$$

$$m = Floor(a/b) \equiv (a \geq m \cdot b \quad (m+1) \cdot b > a)$$

とおくとき

$$P_3(n+1) \Rightarrow \ni m\,Q(m)$$

$$Q(n+1) \Rightarrow <\alpha> Q(m)$$

$$Q(0) \Rightarrow (Z>78 \wedge 89 \geq Z \wedge n=89-Z+11y)$$

が成立するから証明終り。

3.1.2 General Recursion Teory and Denotational Semantics

(1)ではGeneral Recursion Theor についてのsurvey を行なう。この理論は、一般にAbstract な Domain に対するRecursive なプロセスや関数について研究するもので、いくつかの異なったアプローチが可能である。

ここではDenotational Semantics との関連から、R. A. Platek や Y.N. Moschovakis らのInductive なアプローチについて述べる。

(2)では(1)のapplication としてパスカルのサブセットについての数学的Semantics について紹介する。

(1) General Recursion

(1.1) Type Stvucture

Definition：Type Symbols TS を次のように定義する。

1) $0 \in TS$

2) if $\sigma \in TS$, $\tau \in TS \Rightarrow \sigma \to \tau \in TS$

Cor ：TSの定義よりType Symbol σ ($\neq 0$)は次の様な形でunique に表現される。

$$\sigma = \sigma_1 \to (\sigma_2 \to (\cdots(\sigma_K \to 0)\cdots))$$

以下では上式の右辺を$\sigma_1 \to \sigma_2 \to \cdots \to \sigma_K \to 0$と略記することにする。

Definition：Type Symbol σのレベル l を次の様に定義する。

1) $l(0) = 0$

2) $l(\sigma) = \max\{ l(\sigma i) + 1 \}$ if $\sigma = \sigma_1 \to \sigma_2 \to \cdots \to \sigma_K \to O$

Obを任意の集合とする、このとき各$\sigma \in TS$に対して

$T(\sigma)$ (= Type σ の total objects)

$PT(\sigma)$ (= Type σ の Partial objects)

$HC(\sigma)$ (= Type σ の hereditarily consistent objects)

を次のように定義する。

$T(0) = Ob$

$T(\sigma) = \{ f \mid f : T(\sigma_1) \times \cdots \times T(\sigma_K) \to Ob \}$

($= T(\sigma_1) \times \cdots \times T(\sigma_K)$ 上Obへの total function の全体) if $\sigma = \sigma_1 \to \sigma_2 \to \cdots \to \sigma_K \to O$

$PT(0) = Ob$

$PT(\sigma) = \{ f \mid f : T(\sigma_1) \times \cdots \times T(\sigma_K) \xrightarrow{p} Ob \}$

($= T(\sigma_1) \times \cdots \times T(\sigma_K)$ 上Obへの partial function の全体) if $\sigma = \sigma_1 \to \sigma_2 \to \cdots \to \sigma_K \to O$

$HC(0) = Ob$

$\sigma \neq 0$のとき$HC(\sigma)$は、$HC(\sigma_1) \times HC(\sigma_2) \times \cdots \times HC(\sigma_K)$ 上

の partial monotone function の全体をいうが、fがmonotone であるとは、

$f(g^{\sigma_1}, g^{\sigma_2}, \cdots, g^{\sigma_K}) \simeq \times$ でしかも

$g^{\sigma_1} \leq h^{\sigma_1}, \cdots, g^{\sigma_K} \leq h^{\sigma_K}$ ならば

$f(h^{\sigma_1}, \cdots h^{\sigma_K}) \simeq \times$ となることをいう。

また$g \leq h$とはhが partial fiunction として、gの extension となっていることを示している。

(1.2) Platek's System

次にPlatek の理論について述べることにしよう。これはHC上の基本的な関数からFixed point operatorを用いて次々とRecursive function を構成するシステムであり、このシステムにおけるFixed point operator の果す役割を明確にするReduction 定理と呼ばれるものが成立する。

HCを1.1で定義した hereditarily consistent objectsの全体とするとき

Definition : $\leq$HCとなる$\mathcal{B}$について$R\omega(\mathcal{B})$を次に定義する。

1) $R\omega(\mathcal{B}) \ni \uparrow$ (= totaly undefined object)

2) $\mathcal{B} \leq R\omega(\mathcal{B})$

3) DC, I, K, S, FP $\in R\omega(\mathcal{B})$

4) $f^{\sigma\to\tau}, g^{\sigma} \in R\omega(\mathcal{B}) \Rightarrow f(g) \in R\omega(\mathcal{B})$

このようにして定義された$R\omega(\mathcal{B})$がRecursive なプロシジャー全体を構成しているのだが、次にDC, I, K, S, FPについて説明していくことにする。

$$DC(x, y, f^1, g^1) = \begin{cases} f & \text{if } x = y \\ g & \text{if } x \neq y \end{cases}$$

と定義され、Definition by Cases と呼ばれる。

$I^{\sigma\to\sigma}$, $K^{\sigma\to\tau\to\sigma}$, $S^{(\sigma\to\tau\to\lambda)\to(\sigma\to\tau)\to\sigma-\lambda}$ は

$I(f^{\sigma}) = f^{\sigma}$

$K(f^{\sigma}, g^{\tau}) = f^{\sigma}$

$S(f^{\sigma\to\tau\to\lambda}, g^{\sigma\to\tau}, h^{\sigma}) = f(h)(g(h))$

と定義され、Combinatory Logic においてよく知られているようにI，K，Sの導入によって、λ－notation を除去することが可能となる。

Fixed point operator FP について述べる前に次のPropositionについて説明する。

Prop ：

$f \in HC(\tau\to\tau)$，$\tau \neq o$とするとき$f(g) = g$

を満足するような、Relation $\leq$ に関する最小のobject $g \in H.C(\tau)$が存在する。

これは

$g_0 = \uparrow^{\tau} \leq g_1 \leq \cdots \leq g_{\partial} \leq g_{\partial H} \cdots \leq g_{\lambda} \leq \cdots$

（ただし ∂, λはordinal number ）

を次の様に構成することによって容易に示せる。

$g_0 = \uparrow^{\tau}$（＝type τのtotaly undefined object ）

$g_{\partial+1} = g(g_{\partial})$

$g_{\lambda} = \bigcup_{\partial<\lambda} g_{\partial}$　　λがlimit ordinal のとき

このPropositionによって定まるfのLeast Fixed pointによって

Definition ：

$FP(f^{\tau\to\tau}) =$ Least Fixed point of f

すなわちFPはfのLeast Fixed point を構成するoperator として定義される。

Reduction 定理：

$\mathscr{B}$が自然数のcharacteristic function，$\varphi(n) = n+1$や $\varphi(n) = n\dot{-}1$などの基本関数を含んでいるとき

$R_{\omega(\mathscr{B})^{\ell}+3} = R_{\ell+1(\mathscr{B})^{\ell}+3}$

ここで　$R\omega(\mathcal{B})_{\ell+3}$ とは　$R\omega(\mathcal{B})$ のうちレベルが $\ell+3$ 以下のもの全体を示し、$R_{\ell+1}(\mathcal{B})$ とは Fixed point operator $FP(\tau\to\tau)\to\tau$ の使用を $\ell(\tau)\leq\ell+1$ にかぎって許していることを示している。

つまり　$f^{\tau\to\tau}\in HC(\tau\to\tau)$　$\ell(\tau)\leq\ell+1$ となる f に対してのみ FP が使用できるという意味である。

したがってこの定理から、もしも我々がレベルが $\ell+3$ 以下の object にしか興味を持っていない場合には、Fixed point operator としてレベル $\ell+1$ 以下のもののみに着目すればよいことを示している。

(1.3)　Moschovakis のアプローチ

Platek の理論は Inductive な構造についての理論としては、十分に満足のできるものであるが他への応用という観点から見るといくつかの技術的に困難な点を持っている。

以下では、Y. N. Moschovakis らのアプローチについて述べよう。

A を自然数　$N=\{0,1,2,\cdots\cdots\}$ を含む集合として A から N への partial function の全体を

Definition :

$Pf_n(A)=\{f : f : A^n \xrightarrow{p} N\}$

f は n 変数の partial function と定義する。

A 上の partial functional Φ とは $A^n\times Pf_{n_1}(A)\times\cdots\times Pf_{n_K}(A)$ 上の N への partial fuction のことで

$\Phi : A^n\times Pf_{nk}(A)\times\cdots\times Pf_{nk}(A)\xrightarrow{p} N$

が monotone であるとは、

$f_1\leq g_1, \cdots, f_m\leq g_m$ つまり g_K が A 上の partial function として f_K の拡張となっているとき、

$\Phi(\vec{\times} f_1\cdots f_m)=W \Rightarrow \Phi(\vec{\times} g_1\cdots g_m)=W$

となることをいう。

monotone な functional $\Phi(\vec{\times} f\ g)$ が与えられたとき

$$Pf_n(A) \ni f \quad \vdash \quad \lambda \vec{x} \Phi (\vec{x} f g) \in Pf_n(A)$$

は$Pf_n(A)$から$Pf_n(A)$へのmonotoneなoperatorとなっているので、1.2で述べたのと同様の議論によって、そのLeast Fixed pointが存在するがこれをΦ^∞と書くことにしよう。

monotoneなfunctionalのクラス$\mathscr{F}$が与えられているとき、$\mathscr{F} \ni \Phi$について

$\lambda \vec{x} g \Psi (\vec{x} g) = \lambda \vec{x} g \Phi^\infty (\vec{n} \vec{x} , g)$, $\vec{n} \in N^k$ となるようなΨを$\mathscr{F}$－recursiveといい、このような$\mathscr{F}$－recursiveなfunctionalの全体、つまりmonotone functionalのクラス$\mathscr{F}$のFixed point全体を$Ind(\mathscr{F})$と書き$\mathscr{F}$－recursiveなfunctionalという。

さて$Ind(\mathscr{F})$がRecursion Theoryの Abstract な拡張となるためには、クラス$\mathscr{F}$はNの原始帰納的functionやEvaluation $\Phi (f \vec{x}) = f (\vec{x})$を含み、合成や関数のおき換えつまり

$$\Phi (\vec{x} g) = \Psi_0 (\vec{x} \lambda \vec{x} \Psi_1 (\vec{x} g) g)$$

$$\Psi_0 , \Psi_1 \in \mathscr{F}$$

さらにはDefinition by Casesについて閉じているなどの条件が必要となるがこのような$Ind(\mathscr{F})$についてはS.C. Kleeneの First Recursion定理の拡張として次の定理が成立する。

Induction Completeness 定理

$\Phi (\vec{x} g f) \in Ind(\mathscr{F})$とするとき$\Phi (\vec{x} g f)$のFixed point

$\Phi^\infty (\vec{x} f)$ もまた $\Phi^\infty (\vec{x} f) \in Ind(\mathscr{F})$

（Y.N. Moschovakis 1976）

(2) Denotational Semantics

以下ではパスカルのサブセットのSemanticsについてのJ.E. Donahue〔1〕の仕事について紹介する。プログラムのSemanticsを定めるFunctionalについてはその連続性について言及すべきであるが、ここでは1との関

連から連続性については他に譲り、monotoneなFunctionalのFixed point Theoryの応用としてプログラムのspecificationを見てみることにする。またこのサブセットについてのinformalな説明は、Donahue〔1〕を参考されたい。

(2.1) Domain

(2.1.1) Syntactic Domain

Id	identifiers
T＝{ true, false }	truth value names
N＝…－1，0，1…	numerals
Var＝Id＋〔Id×Exp〕	変数
Exp＝T	
＋N	
＋eof	end of file indicator
＋〔Id×Varg*〕	関数
＋〔Uop × Varg〕	unary opevators
＋〔Bop × Exp×Exp〕	binary operators
Varg ＝ Id＋Exp	value arguments
Stmt ＝ null	null statement
＋〔goto × Id〕	
＋〔Var × Exp〕	代入
＋〔write × Exp〕	
＋〔Id×Id*×Varg*〕	procedure designators
＋〔Exp×Stmt×Stmt〕	conditional
＋〔Exp × Stmt〕	while
＋〔Id×Exp×Exp×Stmt〕	for statements
＋〔Stmt × Stmt〕	sequenees
＋〔begin×Stmt×end〕	compound statements

Vblock = 〔Id*×Id*×Lblock〕

変数宣言あるいはarray 宣言にはじまるブロックでラベル宣言ブロックに続く。

Lblock = 〔〔Id×Stmt〕×Lblock〕 + Pblock

ラベルブロック

Pblock = 〔〔Id×Id*×Id*×Vblock〕×Pblock〕

+ 〔〔Id×Id*×Vblock〕×Pblock〕

+ 〔begin stmt end〕

プロシジャー宣言、関数宣言あるいは

compound statement

Prog = Id×Vblpck

プログラム

Remark

$D^{*} = D^{0} + D^{1} + D^{2} + \cdots + D^{n} + \cdots$

D^{0} = { nil }

(2.1.2) Semantic Domain

Int	整数
Bool	Booleans
Undef = { T }	undefmed obiect
Val = Int + Bool + Undef	値
Func = Arg* → Val	
Arg = Val + 〔Int → Val〕	arguments
Proc = Id* → 〔Arg* → 〔C → 〔S → Val*〕〕〕	

プロジジャーは変数とvalue argumentsを与えるとcontinuation C, state Sに依存してoutput fileを決定する。

Lab = C

ラベル

$C = S \rightarrow Val^{*}$ continuations

continuation は computation の残りを示し、state S に依存して output file の最終値を定める。

$S = [Id \rightarrow [Val + [Int \rightarrow Val]]] \times Val^{*} \times Val^{*}$

states はプログラムの変数、input, output file の値を保持する。なおこの3つのコンポーネントをベクトルとして S = (sv, si, so) と書くことにする。

$Env = Id \rightarrow [Func + Proc + Lab]$ environments

environment はアクセスできる関数、プロシジャーおよびラベルの値を保持する。

(2.1.3) Meaning functions

$Me : Exp \rightarrow [Env \rightarrow [S \rightarrow Val]]$

$Ma : Varg \rightarrow [Env \rightarrow [S \rightarrow Arg]]$

$Ms : [Pblock + Vblock + Lblock] \rightarrow [Env \rightarrow [C \rightarrow [S \rightarrow Val^{*}]]]$

$Mp : Prog \rightarrow [Val^{*} \rightarrow Val^{*}]$

Me, Ma は通常の方法で帰納的に定義され、Ms は次に定義されるような Functional $\mathcal{F}$ の Least Fixed point として定められる。

$\mathcal{F} : [Pblock + Vblock + Lblock] \rightarrow [Env \rightarrow [C \rightarrow S]] \rightarrow Ms \rightarrow Val$

(2.1.4) $\mathcal{F}$ の定義

Statements の場合分けによって $\mathcal{F}$ を定義する

Notation : substitution を

$S[x \leftarrow v] = (sv[x \leftarrow v], si, so)$ すなわち state の変数 x の値を v に置換えることを上のように略記する。

$\mathcal{F}$ の Definition :

$\mathcal{F}(\langle\!| id := exp |\!\rangle, e, c, s, Ms) =$

if　　sv⟦id⟧∈Val then C〔S〔id←v〕〕

else　undef

ただし　v＝Me⟦exp⟧(e，s)

$\mathcal{I}$(⟦id〔exp_1〕：＝exp_2⟧，e，c，s，Ms)＝

if　　sv⟦id⟧∈Int → Val then C(S〔id(v_1)←v_2〕)

else　undef

ただし　v_1＝Me⟦exp_1⟧(e，s)∈Int

v_2＝Me⟦exp_2⟧(e，s)

$\mathcal{I}$(⟦null⟧，e，c，s，Ms)＝C(S)

$\mathcal{I}$(⟦read id⟧，e，c，s，Ms)＝

if　si＝Val^0 then undef

else

if　sv⟦id⟧∈Val

then C((sv〔id←hd(si)，tl(si)，so))

Remark

hd：D^*∋$<x_1 \cdots x_K>$ ↦x_1∈D

tl：D^*∋$<x_1 \cdots x_\ell>$ ↦$<x_2 \cdots x_\ell>$∈D^*

$\mathcal{I}$(⟦write exp⟧，e，c，s，Ms)＝

if　so∈Val^* then C(csv，si，append

(so，v))

else undef

Remark

append：$D^* \times D$∋($<x_1 \cdots x_\ell>$，a>　↦$<x_1 \cdots x_l a>$∈D^*

$\mathcal{I}$(⟦goto id⟧，e，c，s，Ms)＝

if　e⟦id⟧∈Lab then e⟦id⟧(S) else undef

$\mathcal{I}$(⟦id(id*：varg*)⟧，e，c，s，Ms)＝

if　e⟦id⟧∈Proc then e⟦id⟧(id*，a，c，s)

ただし　　$a = Ma[\![varg^*]\!]$

プロシジャーコール

$([\![\text{if exp then } stmt_1 \text{ else } stmt_2 \text{ fi}]\!], e, c, s, Ms) =$

if $Me[\![exp]\!] = true$

then $Ms[\![stmt_1]\!](e, c, s)$

else $Ms[\![stmt_2]\!](e, c, s)$

$([\![\text{while exp do stmt od}]\!], e, c, s, Ms) =$

if $Me[\![exp]\!] = true$

then

$Ms[\![stmt]\!](e, \lambda sMs[\![\text{while exp do stmt od}]\!](e, c, s), s)$

else $C(S)$

$([\![stmt_1 ; stmt_2]\!], e, c, s, Ms) =$

$Ms[\![stmt_1]\!](e, \lambda sMs[\![stmt_2]\!](e, c, s), s)$

$([\![\text{begin stmt end}]\!], e, c, s, Ms) = Ms[\![stmt]\!](e, c, s)$

$([\![\text{procedure id } (id_1^*: id_2^*) \text{ vblock:pblock}]\!], e, c, s, Ms) =$

$Ms[\![pblock]\!](e[id \leftarrow p], c, s[id \leftarrow T])$

ここで　procedure p は Least Fixed point of

$P([\![vblock]\!], x, y, cp, sp, p) =$

$Ms[\![vblock]\!](e'[id \leftarrow p], cp', sp')$

ただし

$e'[\![i]\!] = \text{if } e[\![i]\!] \in Lab \text{ then undef else } e[\![i]\!]$

$cp'(s_1) = cp(\langle\!\langle spvr_X \leftarrow s_1 v[\![id_1^*]\!], s_1 i, s_1 o \rangle\!\rangle)$

$sp' = (T[id_1^* \leftarrow sp[\![x]\!], id_2^* \leftarrow y], spi, spo)$

$x : Id^*$　　$y : Arg^*$　　$cp : C$,　　$sp : S$

Remark

◦ continuation はプロシジャーコールのときにパラメータの受渡しの役割を果す。

○ Tはグローバル変数の参照を禁じている。

○ プロシジャーボディより受継いだenvironment のうちLab を undef にする。

$\mathcal{F}$(〖 label id : stmt : lblock〗, e, c, s, Ms) =

Ms〖lblock〗(e〔id←c'〕, c, s〔id←T〕)

ただし

$c' = Ms〖stmt〗(e, c)$

(2.1.5) プログラムの定義

(2.1.4)によって$\mathcal{F}$を定義し、Ms を$\mathcal{F}$のLeast Fixed point とする。このMs を用いて

Mp〖program id : vblock〗(v^*) =

Ms〖procedune id : vblock ; begin id end〗

(e − init, λs(so), (T, V^*, nil in Val^*))

すなわちプログラムはinput file v^*が与えられたとき initial environment e − init と state S よりoutput file をセレクトするcontinuation λs(so) に依存して決定される。

References

(1) J.E Donahue, Complementary Definitions of Programming Language Semantics Lecture Notes in C. S. (42)

(2) Kechris, Moschovakis / Recursion in Higher Types Handbook of Mathematical Logic (North − Holland)

(3) R. A. Platek, Foundations of recursion Theory, Stanford University, January 1966

3.2 計算の複雑さ

3.2.1 はじめに

計算機の使用が浸透していった過去３０年間に、理論的側面からは"計算"ということに関するいくつかの本質的で画期的な事柄が明らかになった。最初に明らかになったことは、計算とは何かということである。すなわち、現在の計算機の数学モデルとして広く認知されているチューリング機械（Turing machine）をはじめとして、いくつかの抽象的モデルが作られ、その上で計算が厳密に定義され、これらのモデルの能力が互いに一致することが証明された。こうして計算あるいはアルゴリズムという概念が安定なものとして確立した。アルゴリズムとは、機械的に物事の成否を確定する手順であり、それは任意の入力に対して働く。この概念の誕生と同時に、皮肉にも、多くの有用な事柄について、その成否を一般的に確定する手順が存在しないことが証明された。このような事柄は決定不能問題と呼ばれる。たとえば、入力プログラムと計算機が与えられたとき、その計算が終了するかどうか（いいかえれば、計算機は無限ループに突入しないかどうか）の判定は決定不能問題の例である。これらの成果は計算不可能性の理論と呼ぶことができよう。

一方、計算可能な問題についても、それを現実の計算機を用いて実用的な時間内に解くという観点に立って考えたとき、実用上有用な多くの問題がそれを実用的な時間内で解くことは原理的に不可能な類の中に含まれることが明らかになった（Cook ７０，Karp ７０）。たとえば、論理回路の簡単化の問題などは、問題の寸法がすこし大きくなれば、それを解くのに要する時間が途方もなく増大して、現実には解き得ないのである。これらの問題はＮＰ完全問題（ＮＰ－ complte）と呼ばれる。この概念のくわしい説明は、§3.2.3で行う。ＮＰ完全問題は計算機科学に対する自然の挑戦である。新たな突破口を求めて努力が積み重ねられている。厳密解と近似解の意味づけが問い直されている。計算機の本質とさ

れる決定性の計算の意義が問い直されている。与えられたデータの中から適当な個数を　数に従って抜き取り、そのデータを使って処理を行なえば格段に早くなるいくつかの問題が明らかになった（Rabin 7 6 ）。近似解法および発見的解法（厳密解を与えるとは証明できないが、よさそうな答えを与える手法）の重要性が計算の複雑さとの兼ね合いで認識され、活発に研究されている。この分野の現状と展望は§ 3.2.4で取上げた。§ 3.2.5では乱数を発生する能力を持ち、乱数に従って計算の制御が変る確率的チューリング機械が通常の機械では到達できない高速度計算の能力を持っていること、並列機械でやれば単一機械でやるより1台当りの問題処理能力が増える例、逆にk台でやっても能率が高々0.7K倍しか増え得ない例などについて述べる。

アルゴリズム論の主な研究分野は、

(i) 作成（創造活動）

(ii) 表現（プログラム言語、データ構造）

(iii) 検証（正しさの証明）

(iv) 解析（有効の検証）

等であるが、計算の複雑さ理論は主に一番最後の項目を主題にしている。その役割りは、

(1) アルゴリズムの有効性を測る目安となる測度の選択

(2) 測度に基づく評価値の算出

(3) 問題と測度を与えたときの評価値の理論的限界（下限値問題）

(4) 問題固有の複雑度の概念の導入及び複雑度に基づく問題の分類

等である。

計算の複雑さの理論は、問題の最善の解法を追求するものである。これにより、従来より格段に速い解法が発見されたこと、新らしい型の解法が生みだされたことなど、その現実世界へ与える利益は大きい。

本稿は最近の主な成果を中心にまとめたものである。今後の研究の展望に役立つものと信じている。

次の§3.2.2では導入として、組合せ問題を解く代表的な手法である動的計画法と解法の複雑さ（計算量）を例に基づいて解説する。

付録1に組み合せ問題の別の有力な一般解法である分枝限定法について解説した。付録2に"確率チューリング機械による高速計算"（R.フレイヴァルト、リガ1975年）の原論文の訳を掲げた。最後の文献表は、約20の内外の論文誌から1975年以降のものを中心に作成した。

3.2.2 組合せ問題と計算量

ここでは具体的な組み合せ問題を取りあげ、それを解く一般的な手法とその計算量について考察してみよう。

最小決定木の構成問題

まず、プログラミングやLSIその他の論理設計に使われる決定表（decision table）の最小表現の問題を例としてとりあげる。決定表はいくつかの条件の組合せとそれが満たされたときの実行すべきことがら（行動と呼びここでは文字で表わす）を列記したものである。記号"－"の導入により入りくんだ状況を簡潔に表現することができる。図3－1の決定表はx_1、x_2、x_3で表わされる3つの条件とaとbという2つの行動からなっている。

数字0で条件が満されないこと、1で条件が満されること、記号－で条件の成否にかかわらず対応する行動を実行することを示す。たとえば、この表の第1行は条件x_1とx_2が共に満されなければ、x_3の成否に関係なく行動aを実行することを示している。すなわち、第1行は000(a)と001(a)を合せたものである。決定表は手続きを表現してはいないが、1つのプログラムを表わしている。決定表を実行するときは具体的に逐次に変数の値を調べて、その結果実行すべき行動が決まる。この手順は、図3－2の決定木によって表わすことができる。決定木の各内節でその中に書かれた変数の値が判定され、その後その変数の値が0(1)ならば左(右)の枝がたどられ、端節につくまで処理が続けられる。今、$x_1 = 0$、

$x_2 = 0$, $x_3 = 0$ と仮定すれば、図３－２の(イ)の決定木では、x_2 で左の枝がたどられ、x_1 で左の枝がたどられて、行動ａが実行される（これは、x_1 と x_2 の値だけで、x_3 の値に依らず行動ａを実行することを意味した、表の第１行に対応する。）同じ決定表は別の決定木、たとえば図３－２の(ロ)でも表現できる。決定木の内節はプログラムで表現すれば、１つの判定 if xi = 0 then go to LEFT else go to RIGHT に対応し、内節の個数が少ない方が実行時間の点からもプログラムの寸法の点からも望ましいことがわかる。そこで、与えられた決定表に対して、内節の個数が最小である決定木を求める問題を考える（最小決定木の構成問題）。決定木の内節の個数を木の値と呼び z で表わす（図３－２のイでは z = 6 , ロでは z = 5 ）。この問題はまず、全ての決定木を数えあげることにより解くことができる。

条　　件

x_1	x_2	x_3	行　動
0	0	−	a
0	1	−	b
1	0	0	a
1	0	1	b
1	1	−	a

図３－１　決　定　表

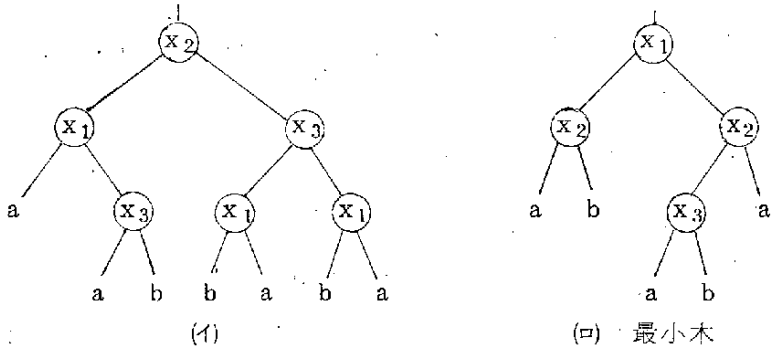


図３－２　決　定　木

数え上げによる解法

与えられたｎ変数の決定表から決定木をつくる操作は次のように述べることができる。まず最初に判定する変数ｘｉを選び、それを決定木の根に郡く。

次に、与えられた表Ｔとｘｉからｎ－１変数を持った２つの表Ｔ（ｉ′），Ｔ(i)をつくる。ここにＴ（i′）（Ｔ(i)）はＴにおいてｘｉ＝０（ｘｉ＝１）の行を集めたものである（この操作をＴをｘｉで分けると呼ぶ。図３－３参照）。これで第１レベルができた。以下同様のことをＴ（i′）とＴ(i)のそれぞれについて操返せばよい。すなわち、Ｔ（i′）から変数ｘｊを選び、Ｔ(i)から変数ｘｋを選び、Ｔ（i′），Ｔ(i)をｘｊとｘｋでそれぞれ分けて、２つの節を第２レベルに作る。このようにして得られた表が、１つの行動、たとえば記号ａしか含まなくなったら、端節にその記号を書いてその枝を終了する。この過程は図３－４に示されている。この図の端節に１つの決定木が対応する。端節の個数は$C_i = i(C_{i-1})^2$，$C_2 = 2$ の数列によって定まるCn で与えられる。最小決定木を求める問題は、このぼう大な問題空間を探索することになる。

決定木の数はｎ＝５のとき1.6×10^6，ｎ＝６のとき1.6×10^{13}と急激に増える。今ｎ＝５の時を例として、決定木を計算機を用いて作成する時間を計算してみよう。５変数の決定表は$2^5 = 32$語に収められているとする。表Ｔをｘｉで分けるためには表の全ての行について

・ｘｉに対応した１ｂｉｔを抜き出す

・それが０か１か判定する

・判定の結果に従ってＴ（i′）（ｘｉ＝０のとき）とＴ(i)（ｘｉ＝１のとき）に表の残りのビットを転送する。

の３つの操作は必要である。１つの操作に10^{-6}秒かかるとする（実際の計算機では上の操作をいくつかの基本操作に分けて行うのでもっと時間が必要である）。１つの木は５レベルから成っているとすれば、１つの木を作るのに$5 \times (3 \times 32) \times 10^{-6} = 5 \times 10^{-4}$（秒）かかる。したがって、５変数のときの全ての木を作る時間は、

$1.6 \times 10^{6} \times 5 \times 10^{-4} = 800$秒＝13分

となる。同じ評価で6変数の全ての木を作る時間を計算すると、800×10^{7}秒＝250年となる。明らかにこの方法は現実的でない。問題空間の探索領域を制限する必要がある。

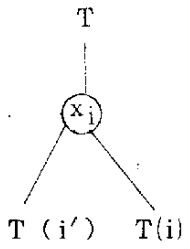


図3－3　Tをx_iで分けてT(i′)とT(i)をつくる

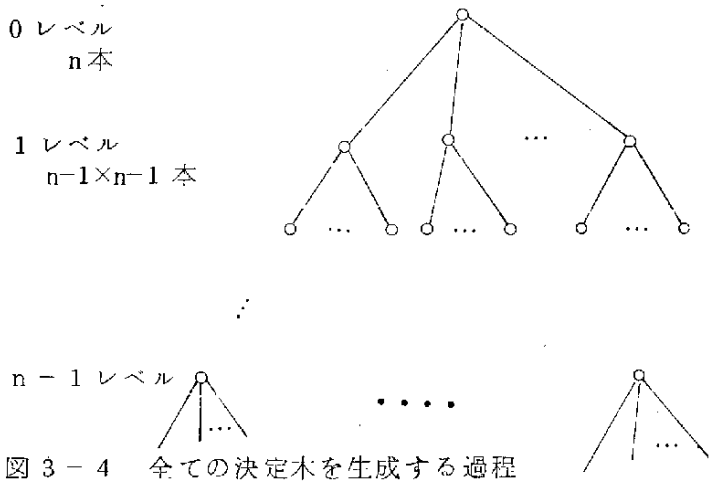


図3－4　全ての決定木を生成する過程

動的計画法による解法

決定木の任意の部分木には部分決定表が対応しており、最小決定木の任意の部分木は最小部分木でなければならない（この性質を一般に動的計画法の原理と呼んでいる）。

表Tに対応する決定木の根には、x_1，…，x_nのいずれかが置かれ、このとき図3－3のT(i)とT(i)に対応する木は最小木でなければならない。いいかえれば、Tに対応する最小木を求めるには、変数x_1，……，x_iでTを分けて得られる大

きさ n − 1 のそれぞれの表 T(i′) と T(i)に対応する最小木（とその値）を計算しておけば良いことがわかる。このことを式で書けば、

$$\min T = \min (\min T(i') + \min T(i) + 1) \qquad (1)$$
$$1 \leq i \leq n$$

となる。T(i′) と T(i)のそれぞれに対応する全ての木を作るかわりに、最小木だけを作ればよいことが要点である。

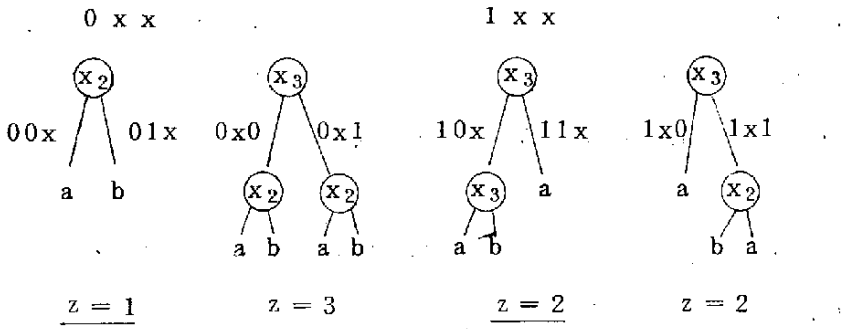
x_1	x_2	x_3	行　動
0	0	0	a
0	0	1	a
0	1	0	b
0	1	1	b
1	0	0	a
1	0	1	b
1	1	0	a
1	1	1	a

図 3 − 5　完全な決定表

段階 1

x 0 0	x 0 1	x 1 0	x 1 1
a			
z = 0	z = 1	z = 1	z = 1
0 x 0	0 x 1	1 x 0	1 x 1
		a	
z = 1	z = 1	z = 0	z = 1
0 0 x	0 1 x	1 0 x	1 1 x
a	b		a
z = 0	z = 0	z = 1	z = 0

段階 2


0 x x
1 x x
x 0 x
x 1 x
x x 0
x x 1
00x
01x
0x0
0x1
10x
11x
1x0
1x1
x00
x01
x10
x11
a
b
z = 1
z = 3
z = 2
z = 2
z = 2
z = 2
z = 2
z = 2
z = 2
z = 2
z = 3
z = 3

段階 3

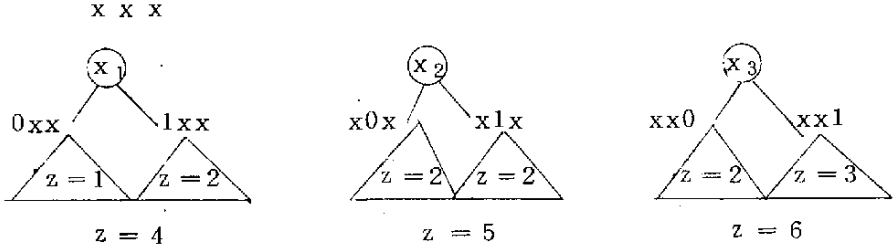

x x x
0xx
1xx
x0x
x1x
xx0
xx1
z = 1
z = 2
z = 2
z = 2
z = 2
z = 3
z = 4
z = 5
z = 6

図 3 − 6

このように、動的計画法の原理を別用して問題を解く手法は動的計画法（dynamic programming；ＤＰ）と呼ばれる。図３－１の決定表に対する最小木を動的計画法を用いて実際に解いてみよう。（図３－６参照）まず、もとの決定表を、記号－をほぐした形の表（図３－５：このような表を完全な決定表と呼ぶ）に書きなす。手順は次の３つの段階からなる。

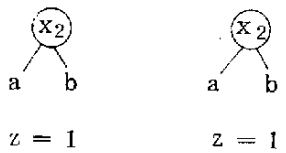
段階１：大きさ１の表に対する最小木を求める。

段階２：大きさ２の表に対する最小木を求める。

段階３：大きさ３の表（これは目的の表Ｔである）に対する最小木を求める。

段階１では、大きさ１の表に対する木は１つしかないからただちに最小木である。ただし、たとえば$x_2=0$，$x_3=0$によって定まる変数x_1だけからなる決定表（この表を$x\,0\,0$と表す）はx_1の値によらず行動ａになっているから、単に行動ａとだけ書いた空な木とする（$z=0$）。これは、境界での特殊事情である。

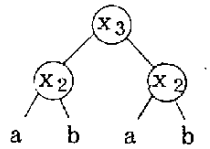
段階２では、式(1)と段階１の結果を使う。たとえば、大きさ２の表０ｘｘ（もとの表で$x_1=0$として得られた部分表）に対応する最小木は、根に置く変数をx_2とした場合とx_3とした場合をそれぞれ求めて、そのうちの小さい方である（式(1)）。根の変数をx_2とした場合は、その左右の部分表は００ｘと０１ｘで表わされ、これらに対する最小木（それぞれａとｂ）とその値ｚは段階１ですでに求まっているから（今の場合いずれも０）、この場合に得られた木の値は$z=1$である（式(1)）。一方、根にx_3を置く場合には、その左右の部分表は、０ｘ０と０ｘ１で、これらに対応する最小木とその値は段階１で



（一致している）

$z=1$　　$z=1$

とすでに求まっているので、得られた木は



z ＝ 3 である。従って最小木は根に x_2 を置いた木である。大きさが3の表についても同様にして、最小木としては、図3－2(ロ)(z ＝ 1) が得られる。

以上の手順に必要な時間は何を単位として測るのが適当であろうか。根の節に置く変数を選び、それに対応した木を作る操作がこの手順の基本操作であるから、全体の手順の時間はこの基本操作の個数に比例すると考えるのが妥当である。全体の基本操作の個数は

$$\sum_{k=1}^{n} 2^{n-k}\ {}_nC_{n-k} \cdot k = n\,3^{n-1}$$

と計算できる。入力として用いたデータはn変数の完全な決定表だったから、その大きさは、$N = 2^n$ と考えることができる （入力データの符号化は最も能率よいものを採用する）。上の手数をNを用いて書けば、$1/3 \cdot \log N \cdot N^{\log 3}$ これはNが十分大きいとき、N^2 より小さい（このことを、アルゴリズムの計算量は $O(N^2)$ より小さいと呼ぶ）。この例にみるように計算量の評価は入力データの表現に依存する。今の場合入力データを完全な表としたので、表の大きさは 2^n が妥当であり、それによって手数が $O(N^2)$ より小さいという評価が得られた。問題を、"任意に与えられた決定表から最小決定木を求めよ"と変えたとき、能率の良い手順（たとえば $O(N^2)$） が存在するかわかっていない。一般の決定表を完全な決定表に書きなおす手間は、入力表の寸法の多項式では押えられない。ここでは節数が最小の木を求める問題を考えたが、根から端節までの節の和を最小にする問題（これは決定表による処理に要する時間の平均を最小にする問題と考えることができる）は、ＮＰ完全と呼ばれる複雑さであることが証明されている。

リテラル数最小決定表の構成問題

先の例は完全な決定表を入力としたとき、$O(N^2)$ 以下の手順であった。記号"－"を利用して表の行数だけでなく表中の０と１（この２つの文字をリテラルと呼ぶことにする）を減らすことができる。たとえば、

x_1	x_2	x_3	
0	0	−	a
1	0	0	a

⇒

x_1	x_2	x_3	
0	0	−	a
−	0	0	a

はリテラルの個数がひとつ減少する例である。与えられた完全な決定表からリテラル数最小の決定表を作る問題を考える（これは"－"の個数を最大にすることと同値である）。"－"を増やす操作は行動が異る２つの行には作用しない。したがって、"－"の個数を増やすには行動の欄にある記号が等しい行をまとめて、そのなかで"－"の数を最大にすればよい。ところで、同一行動を持った行をまとめたものはその行動が実行される条件を項和形式と呼ばれる論理式で書いたものとみなすことができる。たとえば図３－１における行動ａが実行される条件は、

$$\bar{x}_1\ \bar{x}_2\ \bar{x}_3 \vee \bar{x}_1\ \bar{x}_2\ x_3 \vee x_1\ \bar{x}_2\ \bar{x}_3 \vee x_1\ x_2\ \bar{x}_3 \vee x_1\ x_2\ x_3$$

と書ける。表中のリテラルの個数と項和形式における変数の出現回数は等しいから、リテラル数最小の決定表の構成問題はただちに有名な論理関数の最簡形式の構成問題に帰着されることがわかる。論理関数の最簡化手順で入力の多項式で手数がおさえられるようなものは見つかっていず、この問題もＮＰ完全（次節でくわしく説明する）であることが示されている。図３－７にリテラル数最小の決定表を示す。いま一つ考えられる別の問題として与えられた完全な決定表から"－"を用いて行数最小の決定表を構成する問題がある。この問題は最小決定木の構成問題とは独立な問題であるが、同じように動的計画法を用いてＯ（N^2）以下で解ける。図３－１の表はその１つの解になっている。リテラル数最小化の問題と行数最小化の問題は非常に似ているが、その手数は桁違いに違う。この節の内容は主に〔Miyakawa，Sabelfeld ７７〕による。

x_1	x_2	x_3	行　動
0	0	−	a
−	0	0	a
1	1	−	a
0	1	−	b
−	0	1	b

図３－７　リテラル数最小の決定表

3.2.3　複雑さ理論の成果

前節において、アルゴリズムの複雑度に関する種々の問題点を決定表の簡易化という具体的な例に基づいて説明した。本節ではこのテーマについて既に定式化されている事柄を、やや形式的に整理してみよう。

諸　定　義

我々の扱っている問題とは、ある制約条件の下で、与えられた目的関数を最大或は最小にする解を求める最適化問題、又はある与えられた条件を満す解が存在するか否かを判定するいわゆる判定問題をいう。又最適化問題はこれと等価な判定問題に変換することが可能であるので、以下では特に判定問題のみに限定することにする。今Lを、いくつかのパラメタをかえて得られるある型の問題全てから成る問題空間とし、α（L）をLを解くアルゴリズムの全体とする。あるアルゴリズム$A \in \alpha$（L）および問題例$I \in L$が与えられたとき、ある計算機上でAをプログラム化してIを解くものとして、解を得るまでの時間（或は必要記憶容量）はAの有効性の測度として妥当なものである。ここでは時間的要素のみに話を限定する。現実にはどの様な計算機を用いるか、或は問題例Iをどの様に表現するかによって、この時間は大幅に変動する。しかし時間をプログラムの実行ステップ数で表現し、同じ問題表現を（I）から出発するものとすれば、どの様な逐次型計算機を用いても、その大きさに本質的な差異が生じないことが示されている〔Aho, Hopcrsft, Ullman 75〕。

話を具体的にする為に用いる計算機は、単一の無限長テープを持つチューリング機械とし問題表現ε（I）を有限長のビット系列で表現する。この系列の長さn（正整数）を入力の大きさと呼ぶ。問題空間Lは0と1から成る有限長ビット系列の全体$\{0, 1\}^*$の一つの部分集合で表わせる。すなわちチューリング機械上に組込まれたアルゴリズムAは、$I \in L$のとき1（肯定）を出力し、$I \notin L$のとき0（否定）を出力して停止する。アルゴリズムAが問題例Iを解く手数を$C_A(n)$で表わす。ここでnはIの大きさとする。$C_A(n)$なる関数の典型的な型と

て、単純なものから順に列挙すると定数，$\log n$，n，$n\log n$，n^k（$k \geq 2$），2^n，$2^{n\log n}$，$n!$，… 等がある。ここで$n \to \infty$とすると$C_A(n)$が高々多項式（n^k以下）のときと指数関数（2^n以上）の間には、その増大の仕方に本質的な差異があり、特に区別する。次にＮＰ問題の概念を導入しよう。正確には非決定チューリング機械で高々多項式時間内で決定可能なアルゴリズムの存在する問題をＮＰ問題と呼ぶが、ここでは非決定チューリング機械の考えを用いないで、これと等価な定義を与えることにする。

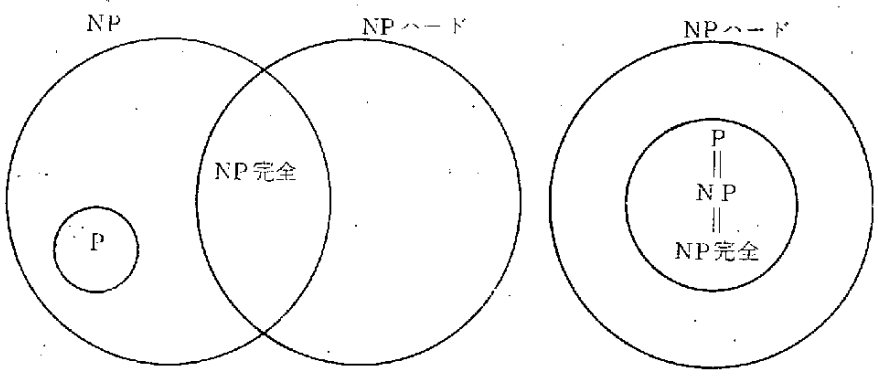
定義：問題Ｌに対して、あるアルゴリズムＡが存在して、大きさｎの入力に対する複雑度$C_A(n)$が$2^{f(n)}$の規模でありかつ$f(n)$が高々ｎの多項式であるとき、Ｌはクラスｎｐに属す、或は単にＮＰであると呼ぶ。特にある多項式$g(n)$に対して、$f(n) = \log_2 g(n)$なるとき$C_A(n) = g(n)$である。この場合、Ｌはクラスｐに属すと云う。

定義から明らかにＰ⊆ＮＰである。

定義：２つの問題空間L_1とL_2に関してL_1からL_2への写像φが存在して、多項式時間で計算可能であり、$I \in L_1$のときかつそのときに限り$\varphi(I) \in L_2$となる場合、L_1はL_2に多項式時間で帰着可能であると呼び$L_1 \propto L_2$と記す。

定義：あるL_0に関し、全てのＬ∈ＮＰに対して$L \propto L_0$なるときL_0はＮＰハードであるという。特にL_0がＮＰハードでありかつ$L_0 \in$ＮＰなるときL_0はＮＰ完全であるという。

ＮＰ完全な問題のクラスはＮＰハードな問題の真の部分集合であることが知られている。あるＮＰハードな問題の内一つでもＰに属するものが存在すれば、定義よりＮＰ＝Ｐとなってしまう。しかし現在に至るまでその様な例は発見されていないし、種々の状況証拠から考えてＮＰ$\supsetneq$Ｐであることが予想されているので、ＮＰハードな問題で多項式時間で解けるものは存在しないであろうという仮説が強く支持されている。図3.2.8の(a)及び(b)にＮＰ$\supsetneq$Ｐなる仮説がそれぞれ真なるときと偽なるときのＰ，ＮＰ，ＮＰハード、ＮＰ完全の各クラスの包含関係を示す。



(a) NP $\supsetneq$ P のとき　　(b) NP＝P のとき

図3－8

P問題

クラスPに属する問題として知られている多くの例は、多項式時間で解けること自体は自明場合が多い。むしろここでは

$\min_{A\in\alpha(L)} \max_{I\in L} C_A(I)$ を求めること（下限値問題）、或はこの値になるべく近い複雑度を有するアルゴリズムAを具体的に求めることが問題となる。P問題はNP問題に比べればわずかであるとはいえその数にかなりなものである。トピック風に例を挙げよう。

整列と探索〔Knuth 72〕

- n個のデータの整列化　　O(n log n)
- 2つの整列化されたデータ配列の融合　　$\log\left(\frac{m+n}{n}\right) \leq MERGE(m,n) \leq m+n-1$
- 整列化された配列の探索　　$\lceil \log_2 n \rceil$
- 最大（小）値（ランダム配列）　　n
- 最大値とその次に大きい値（ランダム配列）　　$n-2+\lceil \log n \rceil$

・中間値（ランダム配列）　　5.43n－16.3

グラフ　　n；節点数　　e；枝数

・平面グラフの判定　　O（n）

〔Hopcroft, Tarjan 74〕

・グラフの連結成分への分解　　O（n＋e）

〔Tarjan 72〕

・グラフの非可分成分への分解　　O（n＋e）

〔Tarjan 72〕

・三角化グラフの判定　　O（n＋e）

〔Rose, Tarjan, Lueker 76〕

・区間グラフの判定　　O（n＋e）

〔Booth, Lueker 76〕

・2組グラフ$G=(V_1, V_2, E)$の対応づけ〔Lawler 76〕

$|V_1|=m$,　　$|V_2|=n$

(a) 最大マッチング　　$O(m^2 n)$, $m \leq n$

〔Hopcroft, Karp 73〕　　$O(n^{2.5})$, $m=n$

(b) 重み和を最大にするマッチング　　$O(m^2 n)$

(c) Maxmin 割当（ボトルネック問題）　　$O(m^2 n)$

ネットワーク　　n：節点数　　m：弧数

・最大流量　　$O(m^2 n)$

〔Edmonds, Karp 72〕

・最小経費流量〔Edmonds, Karp 72〕　$O(m^2 v)$

一定流量vで経費最小化

〔Lawler 7 6　出典〕

・最短経路

(a)　各弧が正の値を持つ場合　　O (n^2)

〔Dijkstra　5 9〕

(b)　一般の場合（負閉路なし）　　O (n^3)

〔Yen　7 0〕

・最小生成木〔Whitney　7 0〕　　O (n^2)

・全ての点対間の最短経路　　O ($n^{2.5}$)

（各弧が正の値を有する場合）

〔Fredman　7 6〕

・負閉路の検出〔Lawler　7 6出典〕　　O (n^3)

・経過時間を持つネットワークの最短経路　　O ($n^3 T$)

（全所要時間がT以下）

〔Lawler　7.6出典〕

・M番目までの最短経路　　O (Mn log n)

〔Dreyfus　6 9〕

・繰返し節点なしのM番目までの最短経路　　O (Mn^3)

〔Lawler　7 6　出典〕

・最小経費対時間比を持つ閉路　　O (n^3 log n)

〔Lawler　7 6　出典〕

数値演算　〔Aho，Hopcroft，Ullman　7 4　出典〕

・行列積　　O ($n^{2.81}$)

・行列式　　O ($n^{2.81}$)

・逆行列　　O ($n^{2.81}$)

・n次多項式の評価　　O (n)

ＮＰ完全問題

ある問題ＬがＮＰ完全であることを定義から直接示すことはかなり難かしい。しかしＬ∈ＮＰを示すと共に、別のあるＮＰ完全問題L_0を持ってきて$L_0 \propto L$を示せば十分であることが容易に証明出来る。従ってＮＰ完全な問題を一つでも発見出来ればこれからいもずる式に新しいＮＰ完全な問題を見つけてゆくことが可能になってくる。Cookは充足可能性問題（後述のリスト参照）がＮＰ完全であることを示した。〔Cook　７１〕

現在知られているＮＰ完全な問題は既に相当な数に達しており、現在尚急速に増加しつつある。以下その内の幾かのものを項目別に分類しリストアップしてみよう。尚〔Garey, Johnson　７９〕中に約３００種程度のＮＰ完全問題のリストが掲載されている

ＮＰ完全問題のリスト

〔１〕　論理関数

〔1.1〕　充足可能性

入力：ｎ変数x_1 …，x_nから成るｐ個の和項C_1，C_2，…，Ｃｐ

判定：論理積$C_1\ C_2$ …Ｃｐ　が恒等的に０（偽）か？

〔1.2〕　３－充足可能性

1.1の充足可能性問題に於いて各和項に含まれるリテラルの個数を３以下に制限した問題

〔1.3〕　２－充足可能性

入力：ｎ変数x_1，…，x_nから成るｐ個の和項C_1，…Ｃｐ．及び正整数ｋ（≦ｐ）

判定：少なくともｋ個の和項から成る論理積が恒等的に０（偽）か

〔２〕　分　　割

〔2.1〕　等　分　割

入力：ｎ個の正整数　a_1，…，a_n

判定：２つに分割して、それぞれの和を等しく出来るか

〔2.2〕　グラフの等分割

入力：2 p個の節点を持つグラフGと正整数k

判定：Gをp個の節点を持った2つの集合V_1 とV_2 に分割し、V_1 とV_2 をつなぐ枝の数がk以下に出来るか？

〔2.3〕　2点 s，tを指定したグラフの等分割

2.2のグラフの等分割に於いて、$s \in V_1$ ，$t \in V_2$ なる条件を付加した問題。

〔2.4〕　3コ組分解

入力：正整数の集合｛ a_1 a_2 …，a_n ｝及び正整数bとd

判定：与えられた集合を要素の数が3以下のp個の集合S_1 ，…，S_p に分割しp≦d，かつ各Si の和をb以下にできるか

〔2.5〕　1進法入力分割

入力：正整数n，a_1 ，…，a_{3n} 及び以下の条件を満すb

$$\sum_{1 \leq i \leq 3n} a_i = nb, \quad \frac{b}{4} < a_i < \frac{b}{2} \quad (1 \leq i \leq 3n)$$

但し、入力は1進法表示とする。

判定：集合｛ a_1 ，…，a_{3n} ｝をA_1，…，Anに分割して、各Aiは3個の要素を含み、その和が丁度bとなる様に出来るか？

〔2.6〕　グラフの三角形分割

入力：3 n個の節点を持ったグラフG

判定：Gの全ての節点をn個の3角形で被えるか？

〔2.7〕　彩　色　数

入力：グラフG、正整数k

判定：Gの節点をk以下の色で塗り分けて隣接節点が同色にならないように出来るか？

〔2.8〕　三色問題

入力：次数が高々4である平面グラフG

判定：Gの節点を3色以下の色で塗り分けて隣接節点が同色にならないように出来るか？

〔2.9〕 単純最大カット

入力：グラフG、正整数k

判定：Gの節点集合を2つの部分V_1 とV_2 に分割して、S_1 とS_2 の間を結ぶ枝の数をk以上に出来るか？

〔3〕 被　覆

〔3.1〕 厳密被覆

入力：有限集合Sとその部分集合の族｛S_1，S_2，…，Sn｝

判定：与えられた集合族の中から互に素ないくつかの集合を選んでSを被覆出来るか？

〔3.2〕 集合被覆

入力：有限集合Sとその部分集合族｛S_1，S_2，…Sn｝と正整数k

判定：与えられた部分集合族の内k個を選んで、Sを被覆出来るか？

〔3.3〕 節点被覆

入力：グラフG、正整数l

判定：l個の節点を選んで全ての枝が、いずれかの節点に接合しているように出来るか

〔3.4〕 次数3のグラフの節点被覆

3.3の節点被覆に於いて、入力グラフGの節点の次数を3以下に制限した問題。

〔3.5〕 次数6の平面グラフの節点被覆

3.3の節点被覆に於いて、入力グラフGを平面グラフで各節点の次数を6以下に制限した問題。

〔3.6〕 クリーク被覆

入力：グラフG、正整数k

判定：k個以下のクリーク（極大な完全部分グラフ）でGの節点を全て

覆えるか

〔3.7〕 節点による閉路被覆

入力：有向グラフG、正整数k

判定：k個の節点を除去して、全ての有向閉路が除去可能か？

〔3.8〕 弧による閉路被覆

入力：有向グラフG、正整数k

判定：k個の弧を除去して、全ての有向閉路を除去可能か

〔3.9〕 グラフの核

入力：有向グラフG

判定：互に弧で結ばれていない節点集合Kを選んで、Kに含まれないどの節点からもKのある節点に向う弧が存在する様に出来るか？

〔3.10〕 当り集合

入力：ある有限集合S上の部分集合族 $\{S_1, S_2, \cdots, Sn\}$

判定：Sの部分集合Tを選んで、全てのiについて $|T \cap Si| = 1$ と出来るか

〔3.11〕 被覆道集合

入力：グラフG、正整数k

判定：Gの全ての節点を被覆する互に素なk個の非サイクル道が存在するか？

〔4〕 グラフ

〔4.1〕 ハミルトン閉路

入力：グラフG

判定：各節点を一度だけ通る閉路（ハミルトン閉路）が存在するか？

〔4.2〕 ハミルトン順路

入力：有向グラフG

判定：各節点を一度だけ通る順路（有向閉路）が存在するか？

〔4.3〕 平面グラフ上のハミルトン閉路

4.1のハミルトン閉路で入力グラフを平面グラフに制限した問題

〔4.4〕　立体グラフ上のハミルトン閉路

4.1のハミルトン閉路で入力グラフを立体グラフ（各頂点の次数3）に制限した問題

〔4.5〕　平面有向ハミルトン順路

4.2のハミルトン順路で入力グラフを入次数が3以下、出次数4以下の有向グラフに制限した問題

〔4.6〕　最大2組グラフ（節点除去）

入力：グラフG、正整数k

判定：高々k個の節点を除去して2組グラフに変換可能か？

〔4.7〕　最大2組グラフ（枝除去）

4.6の問題で、節点でなく枝をk以下除去[注)]して2組グラフに変える問題。

〔4.8〕　クリーク

入力：グラフG、正整数k

判定：Gはk次クリーク（k次完全グラフ）を持っているか？

〔4.9〕　制約木

入力：グラフG、一つの節点v_0，正整数k

判定：v_0 に隣接した各部分木の等点数をk以下にする様な生成木が存在するか？

〔5〕　対　応

〔5.1〕　3次元対応

入力：有限集合T、集合$U \subset T \times T \times T$

判定：Uのある部分集合Wを選んで$|W| = |T|$かつW中のどの2つの要素も、3つのどの座標に於ても一致しないように出来るか？

注）　枝の除去の仕方に開放除去と短絡除去と2通りあるがいづれによってもNP完全となる。

〔6〕　整数計画

〔6.1〕　0－1整数計画

入力：整数行列A（m×n次）及び整数ベクトルB（m次）

判定：各要素が0と1のいづれしかとらないn次ベクトルXでAX＝B

なるものが存在するか

〔6.2〕　ナップサック

入力：n＋1個の正整数a_1，a_2，…，a_n，b、

判定：a_1，…a_n から適当にいくつか選んでその和をbに一致させる

ことが出来るか

〔7〕　ネットワーク

〔7.1〕　ネットワーク設計

入力：枝に重みのついたグラフG、定数b、c

判定：枝の重み和がb以下である様なGの部分グラフG′を選んで、G′

内の全ての節点対を結ぶ最短経路の長さの和がc以下にすること

が出来るか？

〔7.2〕　スタイナ木

入力：枝に重みのついたグラフG、部分節点集合U、正整数k

判定：Uの全ての節点を結びかつ重み和がk以下である様な木が存在す

るか？

〔7.3〕　経路制約木

入力：枝に重みのついたグラフG、あるGの節点v_0，正整数r、k

判定：枝の重み和がr以下である様なGの生成木を選んで、v_0 から他

の任意の節点に至る木の上の経路に含まれる枝の数がk以下で

ある様に出来るか？

〔7.4〕絶対m中心

入力：枝に長さの定義されたグラフG＝（V，E）、正整数$m \leq |V|$，ℓ，

判定：m個の節点集合Vmを選んで

$$\max_{y \in V} \min_{x \in Vm} d(x, y) \leq \ell$$

なる様に出来るか？

但しd（x，y）はxとyを結ぶ最短経路の長さとする。

〔7.5〕 互に交わらない経路

入力：グラフG、n個の節点対（Si，ti）、$1 \leq i \leq n$

判定：各節点対（Si，ti）を結ぶ経路を作り、互に節点を共有しないように出来るか？

〔8〕 スケジューリング

〔8.1〕 スケジューリング問題

入力：処理さるべき仕事集合｛T_1，…，Tn｝及びTiの処理に要する時間τ_i、仕事集合の上で定義された、処理の順序関係を示す半順序グラフG、正整数n、m、ℓ

判定：n個の仕事Ti（$1 \leq i \leq n$）をm個の同種のプロセッサで処理して、全所要時間をℓ以下にするスケジュリングが存在するか？

注）この問題について以下の部分問題も又ＮＰ完全である。(イ)各τ_iが等しい場合、(ロ)m＝2で処理時間が1又は2の場合。

〔8.2〕 独立な仕事のスケジュリング

〔8.1〕の問題で、各仕事間の前後関係がない問題。更にm＝2に限定した場合もＮＰ完全である。

〔8.3〕 仕事の処理順序

入力：仕事集合｛T_1，…，Tn｝及び各Tiについて、処理時間τ_i、期限Di、罰金Pi、正整数k

判定：n個の仕事をある順序で実行し、仕事Tiが期限Djに間に合わぬときは罰金piを課せるものとして、罰金の総額をk以下に押えることが出来るか？

〔8.4〕 節点集合の最適順序づけ

入力：グラフG＝（V，E）、正整数k

判定：|V|＝nとして、各節点に1からn迄の番号を振り

$$\sum_{(u,v)\in E} |f(u)-f(v)| \leq k$$

を満すことが出来るか？

但し、f(u)は節点uの番号とする。

〔8.5〕 根付木のスケジュリング

入力：各節点に重み付の根付木、正整数n、k

判定：各節点に1からnまでのラベルを割当て、どのラベルも2回以下しか出現しないようにする。又根から葉に至る任意の経路の上でそのラベル系列が増大し、かつ任意のラベルjに対して、同じラベルを持つ節点の重み和がkを越えないように出来るか？

〔8.6〕 デッドロック回避〔Sugiyama，Araki，Okui，Kasami 77〕

入力：m個の資源、及びn個のプロセスについて、プロセスj（$1 \leq j \leq n$）が現在専有している資源のリストLj、及び今後どの様に資源を要求、あるいは専有解除するかを記述したプロセス流れ図Pj（有向非閉路グラフ）が与えられる。

判定：全てのプロセスを完了させるような資源割当のスケジュールが存在するか？

〔8.7〕 フローショップスケジューリング

入力：m個のプロセッサとn個の仕事について、各仕事iはm個の作業T_{1i}，T_{2i}，…T_{mi}から構成されている。作業Tjiはプロセッサjによって行なわれるものとし、その処理時間tjiが与えられている。正整数kが与えられる。

判定：各仕事iについて、作業Tjiは$T_{j-1,i}$が終了後に開始されなければいけない。又一つのプロセッサがある作業にとりかかった

場合、それを終了するまで続けるものとする。以上の制約の下で、所要時間がｋ以下で全ての仕事を終えるスケジュリングが存在するか？

〔９〕　詰合せ

〔9.1〕　集合詰合せ

入力：有限集合とその部分集合の族Ｆ、正整数ｋ

判定：Ｆからｋ個の互に素な部分族を選べるか？

〔9.2〕　箱詰問題

入力：ｎ個の品物とそれぞれの容積Ｓｉ（１≦ｉ≦ｎ）、ｋ個の容量ｂの箱

判定：ｎ個の品物を全てｋ個の箱に収容可能か？

〔１０〕　その他

〔１０.１〕　プログラムの最適化

開始点および終了点が１つずつ存在し、開始点から任意の節点へ道があり、任意の節点から終了点への道がある有向グラフを流れ図という。道の集合が、(a)各々の道は互いに共通な節点をもたない、(b)Ｖのすべての節点はいずれかの道に含まれる、の２つの条件を満たすとき、被覆道集合と呼ぶ。流れ図の節点はプログラムの文に対応すると考える。流れ図の枝であって被覆道の枝でない枝にＧＯＴＯ文を対応させれば、最小被覆道に対して、ＧＯＴＯ文の出現の数最小のプログラムが決まることになる（ＩＦ　ＴＨＥＮ　ＥＬＳＥ文は使えないものとする）。次の判定はＮＰ完全である。

入力：各節点の入次数、出次数ともに３以下の流れ図Ｇと整数ｋ

判定：Ｇに個数ｋの被覆道集合があるか？

（西関、高見沢、斉藤　７７）

入力：各節点の入次数・出次数がともに２以下で、全てのサイクルの長さが２以下である流れ図Ｇと整数ｋ

判定：Gに個数kの被覆道があるか？

（永松、本田　７７）

入力：上の制限に加えて、各サイクルについて、その外部からサイクルに入る枝の数が1で、サイクルから外部に出ていく枝の数が2である流れ図Gと整数k

判定：Gに個数kの被覆道集合があるか？

（山下、本多　７８）

〔10.2〕　ペトリ網（Araki, Taniguchi, Kasami　７６）

ペトリ網は有限の節点集合Πと遷移点の集合Σと、それらを結ぶ弧の集合Eと各節点に置かれた石の個数で表わされる初期配置M_0の組

$$N = < \Pi , \Sigma , E , M_0 >$$

として定義される。

遷移点は全ての入力節点に石が1つ以上ある時に発火して、石を1個出力節点に送る。初期配置から到達可能な配置を到達可能配置と言う。任意の到達可能な配置における各節点の石の個数が1以下のとき、Nは安全と言われる。またある配置M_fが与えられたとき、NがN_fに関して正しく停止すると言われるのは、任意の到達可能配置からM_fに到達可能であり、M_fでは発火しないときである。各遷移点はただ1度だけしか発火しないペトリ網において、次の決定はNP完全である（Araki, Taniguchi, Kasami　７６）

i）入力：Nと配置M

判定：MはNから到達可能か？

ii）入力：N

判定：Nは安全か？

iii）入力：NとM_f

判定：N はM_fに関して正しく停止するか

〔10.3〕 探　索

いくつかの最適探索木問題がＮＰ完全であることがわかっている〔弓場、星80〕。特に次に述べる端節路長和最小（これは平均処理時間最小と同じ）の決定木を構成する問題はＮＰ完全である（Hyafil，Rivest 76）

入力：行動を表わす記号が全て異っているような決定表と整数ｋ

判定：全ての端節のレベルの和がｋ以下である決定木が存在するか？

3.2.4　近似アルゴリズム

何故、近似アルゴリズムが必要か

ほとんど大部分の興味ある問題が、ＮＰハードであることが明らかにされている。これの打開策として、第一に考えるべきことは、Ｐ≒ＮＰなる仮設命題の真偽を確かめることである。しかしこの命題が提起されて以来、費された、数多くの研究者達の精力的な努力にもかかわらず未解決のままである。この命題が否定されれば勿論大変なことになるが、又予想通り肯定的に証明されても、恐らくこの為にある重要な方法論が提起され、アルゴリズムの研究の分野に一大転期をもたらすことになろう。しかしその時まで何もせず待つ訳にも行かない。次に考えられることは、ＮＰ－アルゴリズムの改善である。例えばα^nの手数を要したものが$\alpha^{n/\beta}$（$\beta>1$）で出来れば、これは明らかに一つの改善である。動的計画法、整数計画法、分枝限定法等の計算速度向上の為になされた努力の大半はこれに相当する。しかしこれだけでは問題の規模が少し大きくなれば、現在の最先端を行く計算機を用いてもたちまち行きづまってしまうことは既に述べた。

そこで第三に登場したのが以下に述べる近似アルゴリズムである。

近似アルゴリズムＬとは何か

一言で云えば、問題の枠組を変更して得られるＰアルゴリズムのことである。問題変更の度合は当然出来るだけ小さい方が望ましいが、その仕方には大きく分けて次の二つがある。

(a)　解の精度に関する緩和

(b)　問題空間の縮少

(a)は、最適化問題に於いて、近似解で妥協し、その代りアルゴリズムの簡易化を得ようとする。(b)は、問題空間の内、真に困難な問題例を削除してしまおうとするアプローチである。勿論、この２つの複合した場合も存在する。方法論的には、発見的手法の導入とその評価、及び確率的手法の導入が特徴的である。

第四には、並列アルゴリズムに基づいた並列機械及び確率機械の導入であるが、これについては3.2.5節で述べる。

以下、近似アルゴリズムに焦点を絞る。

1.　近似アルゴリズムで用いられる方法論

1.1　発見的手法

近似アルゴリズムは、しばしば、発見的手法に基づいて構成される。発見的手法が、これに対するいわゆる厳密アルゴリズムと比べて一段と価値の低いものに見なされているのは事実である。しかし厳密なアルゴリズムではＮＰハードの複雑さのものしか得られない以上何らかの直観と経験に基づく妥協的手法ー発見的手法ーに依らざるを得ない。Weinerよれば〔Weiner 75〕

与えられた仕事に対する発見的手法の設計に関して、かなり多くの標準的な手法がある。

しかし現時点ではあくまで実験的な道具であり、正確な科学でない。

というコメントがある。アルゴリズムの複雑さの研究は従来最悪ケースに於ける所要手数、或は必要なメモリ空間の大きさを求めることに重点が置かれてきた。しかし現実問題としては、正確な最適値の限界を知ることはそれ程切実な要求でない場合が多く、むしろ理論的な興味に根差している。すなわち、近似解であっても、それを求めるコストが、厳密解のそれより安ければ、前者の方が好ましい場合がはるかに多い。又現実の問題が、理論的な研究レベルで例として用いられている問題に比して、著しく大規模であり、莫大な入力変数の全てに対して正確

なデータを集めることさえ困難であるのが普通である。この場合、入力データとして、統計に基づく推測値が用いられ、出力として、近似的、平均的な解の方がより大切な意味を持つ。以上のことから発見的手法の重要性を再認識し、その標準的な手法の形態を列挙してみよう。〔Weiner　75〕

〔大附　79〕

(a)　逐次構成法：部分解を逐次拡大して完全解まで構成して行く手法。各拡大の段階に於いて、局所的な判断に基づいて、拡大の仕方に関して、決定を下す。greedy法（貪欲法）〔Horowitz, Sahni 78〕もこれの一種である。

(b)　逐次改善法　与えられた問題の制限条件を満す一つの解（実行可能解）から出発し、目的関数の値を逐次改善しながら、最終解に至る手法。逐次改善の仕方は、何らかの意味で、現段階の解の近傍にある実行可能解の内に、移行行の範囲を絞る。又初期解の与え方により、最終解の近似度が大きく変動する場合もあり、ランダム処理が有効となる場合もある。

(c)　分割統治法（devide and conquer）；　原問題をいくつかの、より扱い部分問題に分割して、それぞれを解き、各部分解を統合して、最終解を得る。今大きさnの問題をk等分して分割統治法を用いた場合のコストは

$$C' = k\,C\left(\frac{n}{k}\right) + C_0$$

となる。ここでC_0は統合の為のコストとする。今$C(n) = n^{\alpha}$とすれば

$C' = \dfrac{n}{K^{\alpha - 1}} + C_0$ となり、

$\alpha > 1$ならば、この手法によって、手法を改善出来る可能性が残されたことになる。

(d)　探索法；木探索等に於いて、その深さ、広がり等が著しく大きい場合、適切な、中間節点の評価、或は、探索打切条件を導入して、ある時点で探索を打切り、これを近似解として出力する手法である。人工知能研究分野に於け

る、問題解決プログラム等に見られる一つの典型的な手法である。

(e) 特徴抽出法：ある問題を、いくつかの異った解法で解いてみて、それらの解の間に共通に出現する特徴を抽出する。その後で、新たにその特徴を予め含む様な実行可能解に限定することにより、問題空間の縮少をはかる方法。

1.2 確率的手法

以上の様な発見的手法も、唯これだけでは単なる丼勘定にすぎない。その能力に対する何らかの評価がなされて、はじめて科学となり得る。しかし従来、この種の発見的手法の評価法として行なわれたことは、これをプログラム化して、いくつかの問題例を実際に解いて、何らかの方法で求めた厳密解と比較したり、或は別の発見的手法による解との比較をして、その差異に関する統計より、有効性の有無を判断する、いわゆる事後テストによるのが普通であった。この評価法は重要ではあるが、時として、標本として選んだ問題に偏りがあったり、又厳密解を求める困難さの為、標本例が少なすぎたりしてしばしば説得力に欠ける場合が多い。本節では、これらの近似アルゴリズムの誤差の限界を解析的に定める評価法の他、上述の標本問題抽出を偏りなく行う為に、予め問題空間に確率分布を導入し、平均的な動作の解析を可能にした状態で評価することについて述べる。

又問題空間ではなく、アルゴリズム自体の中に積極的にランダム化の要素を組込んで、改善をはかった、いわゆるランダムアルゴリズムについても触れる。

2. 解の精度に関する緩和

近似アルゴリズムの誤差の限界を、確率的手法を用いないで解析的に定める評価法について述べる。

2.1 記号と諸定義

L：ある問題空間

A：Lを解く一つのアルゴリズム

I ；問題例　　I ∈ L

$f_A(I)$； Lが最適化問題であるとき、問題例 I ∈ L をアルゴリズムAで解いたときの解の値

$f^*(I)$； 問題例Iに対する最適解

| I | ；問題例Iの大きさ

アルゴリズムAの精度は、次式で表現される。

$$| f_A(I) - f^*(I) | \leq \varphi (f^*(I))$$

ここで $\varphi (f^*(I))$ なる関数が、求めるべき精度であり当然、全ての I ∈ L に対して、その値が出来るだけ小さい方が秀れたアルゴリズムとなる。

$\varphi (f^*(I))$ のとる具体的な形によって、近似形式に関するいくつかの用語が導入されている。〔 Horowitz, Sahni 78 〕

a) 絶対近似；$\varphi (f^*(I)) = K$（定数）

b) C(n) -近似；$\varphi (f^*(I)) = C(n) f^*(I)$

　　但し　　$n = | I |$

c) ε -近似；$\varphi (f^*(I)) = \varepsilon f^*(I)$

　　但し　　ε は定数

この様な諸定義が任意のNPハードな問題の解決に対して有効という訳ではなく、それどころか、各近似形式で有効となる問題は極く限定された数にすぎない。

以下特にa）及びb）の場合について述べる。

2.2 絶対近似

この形式は誤差の限界が問題例に依存しないで、一定値で限られる点で、他の相対的近似より厳しいが、それだけにこの形式での成功例はわずかである。

例1. 平面グラフの彩色問題

アルゴリズムA

1. 頂点数が1なら彩色数＝1

2. グラフが2組グラフならば彩色数＝2そうでなければ彩色数＝4。

全ての平面グラフの彩色数は4色で十分なことが証明されたが、3色問題はNP完全である。従って $|f_A(I)-f^*(I)| \leq 1$ であり、Aは1－絶対近似アルゴリズムである。Aの複雑度は$O(|V|+|E|)$である。但しV、Eはそれぞれ与えられた平面グラフの節点及び枝集合である。

例2. k箱への最大詰合せ

アルゴリズムA

1. n個の品物を小さい順に並べ換える。
2. 一つの空箱に、品物を小さい順につめる。

途中品物がなくなれが終了。

Aはgreedy法に基づいている。又（k－1）－絶対近似である。これは、k個の箱をつなぎ合せた大きな箱（容積kL）に品物を小さい順に詰め込み、丁度箱の継目に来た品物を除外することを考えれば容易に理解出来る。Aの複雑度は、O（n log n）である。

絶対近似形式はかなり厳しい要求で、この形式ではいかなるアルゴリズムでもNPハードの複雑度から脱し切れない様な問題が大部分である。ある問題空間Lに関するいかなるK－絶対近似アルゴリズムもNPハードであることを示す一つの手法として以下のようなものがある。

1. 任意の問題例$I \in L$から$f^*(I')=(k+1)f^*(I)$を満たす様な問題例$I' \in L$を作る
2. Iの最適解を求めることとI′のそれを求めることが等価である様にI′を選ぶ。
3. I′に対する任意のk－絶対近似な解は、実際にはその最適解と一致することを示す。

以上のことが示せれば、Lを解くk－絶対アルゴリズムAはI′の厳密解を与え(3)、従ってAはIを解く厳密解でもある(2)ことが云える。

例1. 整数計画問題

但し目的関数　$Z = \sum_{i=1}^{n} p_i x_i$ に於いて係数 p_i は整数であるとする。

この問題Lに対するいかなるk－絶対近似アルゴリズムもNPハードである。

$I \in L$に対して、$I' \in L$を、制約条件はそのままとし、目的関数を

$Z = \sum_{i=1}^{n} (k+1) p_i x_i$ とすればよい。

例2.　最大クリーク

この問題に対するk－絶対近似アルゴリズムはNPハードである。入力グラフGに対してG'は、k＋1個のGのコピー、及びこれに互いに異なるコピー間の全ての頂点対に辺を付加したグラフとすればよい。

2.3　ε－近似

k－絶対近似に対応して、ε－相対近似と呼んでも良い性質のものである。$0 < \varepsilon \ll 1$でないと良い解とは云えない。絶対近似よりやや制限がゆるく、多くのスケジューリング問題、詰め合せ問題において良い成果が得られている。絶対近似の場合と異なり、εを任意を与えて、その代りにアルゴリズムの複雑度が入力サイズnのみでなく$1/\varepsilon$の関数である様な問題も存在する。更に、いかなるε－近似アルゴリズムもNPハードな複雑度を有する問題も数多く存在する。以上の各場合を例をあげて説明することにしよう。

例1.　独立タスクのスケジューリング問題（その1）

処理時間の既知なn個の独立なタスクをm個の互に等しいプロセッサで並列処理し、その終了時刻を最小にする問題である。

アルゴリズム　LPT（Longest Processing Time）

あるプロセッサが、手空きになったら、その時点で残されているタスクの内最も処理時間の長いものにとりかかる。

GrahamはLPTが$(\frac{1}{3} - \frac{1}{3m})$－近似であることを示した。〔Graham 69〕これは逐次構成法に基づいており、O（n log n）の複雑度を有する。

例 2.　独立タスクのスケジューリング問題（その２）

アルゴリズムＡ

1.　あるＫ（＜ｎ）を定める。

2.　タスクを処理時間の長い順に並べ、最初のＫ個のタスクについて最適解を求める。

3.　残りのｎ－Ｋ個のタスクについてＬＰＴ法を適用する。

これはε－近似アルゴリズム（$\varepsilon = (1 - 1/m)/(1 + \lceil k/m \rceil)$）である。〔Graham　６９〕このアルゴリズムは分割法と逐次構成法（ＬＰＴ）を組合せたものである。最初のＫ個のタスクに関する最適値を例えばＤＰ法で求めると$O(m^k)$の複雑度であり、残りのｎ－ｋ個に関しては$O(n \log n)$である。

全体として$O(n \log n + m^{(\frac{m-1}{\varepsilon} - m)})$となる。精度εを任意に小さく設定出来るが、アルゴリズムの複雑度は、$m^{\frac{1}{\varepsilon}}$で増大するのが特徴である。

例 3.　箱詰問題

ｎ個の品物（大きさa_i，$1 \leq i \leq n$）が与えられ、これらを全て収容するのに最低何個の箱（容量Ｌ）が必要かという問題である。

アルゴリズムＡ

空箱に番号をつける。ある時点で残っている最大の品物を、番号の順に箱調べ、収容可能な最初の箱に入れる。このとき

$$f_A(I) \leq (11/9) f^*(I) + 4$$

なることが示されている。〔Johnson, Demers, Ullman, Garey, Graham 74〕

3.　問題空間の縮少

3.1　確率モデルの導入と定義

ＬＰにおける単体法は、縮退化する一部の特異な場合を除けば、良好なアルゴリズムであり、現実にも広く用いられている。これは問題空間をわずかに縮少して成功した典型的な例である。この節では Karp に従って、与えら

れた問題空間　のほとんど全てに対して良好な振舞を示すアルゴリズムについて述べる。〔Karp　76〕

大きさnの問題例から成る標本空間をLn（Ln⊂L，n＝1, 2……）で表わす。これらLnの上である確率分布が与えられているものとする。

定義：アルゴリズムAが問題Lをほとんど至る所で（almost everywhere，以後　a．eと略記する）解くとは、各Lnから標標抽出した問題系列I_1，I_2，……に対してAが解くことの出来ないIn（n＝1，2……）の個数が有限である確率が1であることを意味する。

現実問題として、問題空間上にある確率分布を知ることは困難な場合が多く、無理に導入すれば実際と遊離してしまう恐れも生ずる。しかし確率的な解析の基礎を固める意味で、これも一つのアプローチである。

3.2　諸　　例

例1.　ユークリッド巡回セールスマン

空間内に与えられたn個の点の全てを少なくとも一度は通る経路の内、長さ最小なものを求める問題である。拡張は容易であるので平面で考える。確率分布を導入する為単位正方形内にn点をランダムにばらまく。

アルゴリズムA

1. 単位正方形に、一辺の長さ$\sqrt{\frac{t(n)}{n}}$の格子状分割をほどこす。ここでt(n)はt(n)～$\log_2 \log_2 n$でかつn／t(n)が平方数となる様に選ぶ。
2. 各格子内で、最短経路を求める。（DP法）
3. 各格子内の部分経路Piを点とみなして、これら全てを結ぶ、最小生成木を作る。このときPiとPj（i≒j）間の距離はPiとPj間の最短距離とする。
4. 各Piをそれぞれ一度、生成木の各枝を2度づつ通る閉路Wを作り、これを解とする。

アルゴリズムAは分割統治法に基づいている。KarpはAがO（n log n）（a．e）の複雑度を有し、かつ任意のεに対してε－近似（a・e）であ

ることを示した。

例2. ランダムグラフ上のハミルトン順路

n頂点ランダムグラフとは各点

対の間に、ある等しい確率P(n)に従って、辺を付加して得られる。

アルゴリズムA 〔Posa 76〕

1. 始点をs、終点t←s、パスP＝φとする。

2. 端点tに接続した頂点の1つを選び、これをvとする。

a) v＝sかつパスPがグラフ上の全ての点を含むとき、ハミルトン巡路は達成され終了。

全ての点を含まぬときは失敗

b) v≠sかつv∈Pのとき、P上でvの次に位置する点v′(≠t)が存在する。Pより(v, v′)を除去し、その代りに辺(v, t)を加える。新しいPは、v′を端点とする。(t←v′)長さは不変である。

ステップ2を繰えす。

アルゴリズムAではc)の段階で、同じ長さ、同じ端点を有するパスは作らない様にする。Aは$O(n^2)$の複雑度を持つ。又明らかにいつも正解に達するとは限らない。

定理 〔Posa〕 $P(n)\sim\alpha\frac{\ln n}{n}$のとき

$\alpha>1$ならば、Aはほとんど至る所でLを解く。

$\alpha<1$ならば、グラフは非連結(a.e)である。

〔Erdos and Renie 59〕従ってハミルトン順路も存在しない。

例3. ランダムグラフ上の最大独立集合

グラフGの頂点のある部分集合が独立であるとは、それに属すどの2点も隣接していないことである。最大独立集合を求める問題はNP hardである。

アルゴリズムA

1. N←φ

2. Nに属さない点で、Nのどの点とも接続していない点が存在すればこ

れをNに加え、2.を繰返す。

存在しなければ終了。

この単純なアルゴリズムは明らかに0(n)である。

定理〔Karp〕　P(n)＝C（定数）ならば

任意の$\varepsilon > 0$に対して

$$f^{*}(I) - f(I) \leq (0.5 + \varepsilon) f^{*}(I) \quad (a.e)$$

4. ランダムアルゴリズム

前節において、問題空間に確率分布を導入し、アルゴリズムの正解率を解析する確率モデルについて述べた。Rabin は、これと異った仕方で、アルゴリズムの中にランダム化の過程を積極的に持ち込み、複雑度の改善をはかることに成功した。〔Rabin　76〕いわば前節のアプローチが消極的確率アルゴリズムであるのに比べ、本節のそれは積極的確率アルゴリズムと呼ばれるべきものであり、その意味でこれをランダムアルゴリズムと呼び区別することにする。Rabin は2つの問題に対するランダムアルゴリズムを与えたが、その内の一つ（最近隣接点対）に対しては、常に厳密解を与えるが、少ない確率で元と同程度の複雑度を残す。他の素数判定問題では、複雑度は改善するが、小さい確率で誤判定の可能性を残すと云う点でややニュアンスが異なる。一方ランダム化を積極的にとり入れたアルゴリズムは、Rabin のものが始めてという訳でもない。探索技法として良く知られているHashing 法或は、統計的決定における標本抽出法等がこれにあたる。以下Rabin のアルゴリズムの概略と共に既存の手法との類似性を求め、ランダム化の意味を明らかにして見よう。尚ランダムアルゴリズムについて五十嵐による秀れた解説〔五十嵐80〕がある。

例1.　最近隣接点対

空間に、与えられたn点の内、最も隣接した点対を求める問題である。この問題は普通に解いてもn（n－1）／2回の比較で求まる。

アルゴリズムA

1. m ($= n^{2/3}$) 個の点をランダムに選ぶ
2. 1.で選んだm個の点の間の最近点対を求め、その距離を δ_1 とする。
3. 任意の点を原点にとり、4点 (0 , 0) (0 , δ_i) (δ_i , 0) (δ_1 , δ_1) をそれぞれ格子点とする。各点を通る4つの格子状分割Li($1 \leq i \leq 4$) を考える。ここで各Li の格子の大きさは一辺が $2\delta_1$ とする。
4. 各Li について、それぞれの格子内の最近隣接点対を求め、それらの間の最小値を求め更に4つのLi の間での最小値 δ を求め出力する。

このアルゴリズムの要点は、2の手続きがO(n)であること。3.で求めた4つの格子分割のいづれか一つの格子内に、必ず、真の最近隣接点対が含まれること。及び4.での手続きがO(n)の期待値を持つことである。

次にこのアルゴリズムAとHashing 法との類似性を対比させてみよう。

	最近隣接点対	ハッシング
目的	最近隣接点対の位置及びその値を求める。	あるデータが表 (HT) 中に存在するか否かを判定し、あればその場所を求める。
原理	探索空間を分割し求める点対が一つの分割領域に含まれる様に選び、分割統治の考えで手数を縮少する。	表HTを同義語空間 (見出ハッシング番地が同一となる語の集合) に分割し、その中だけの探索でよいようにする。
ランダム化の意図	m個の標本点の内の最近点対間の距離の期待値は橋本点をランダムに選んだときが最も小さくなる。この距離は、格子分割の網のサイズを決めこれが小さい程、アルゴリズムAの4の手続きは短くなる。	ハッシング番地をランダムに割付けることにより各同義語空間中のデータ数を一様に分布させる効果がある。それにより平均探索時間が短縮される。

パラメタ決定のトレードオフ	m（1≦m≦n）を大きくとると、4の平均手数は減少するが、1の手数が増大する。	ハッシング関数のランダム様構を複雑化すれば、探索時間は短縮出来るが、計算に手間取る。

例2. 素数判定

与えられた数nが素数か否かの判定は高々$\sqrt{n}$回の除算で判定可能である。

アルゴリズムA

1. あるmを定め、m個の数bi（1＜bi＜n）をランダムに選ぶ。
2. 全てのbi（1≦i≦m）について、命題W（bi）が成立しないとき、nを素数と判定する。

   但し W(b)は次の通りである。

   (a) b^{n-1}≠1（mod n）又は

   (b) あるiに対して（n－1）／2^{i}が整数Kとなり、b^{k-1}とnの最大公約数が1とnの間にある。

アルゴリズムAの正当性は次の事実に基づいている。

定理〔Millor〕 nが合成数ならばW(b)が成立する様なbの個数は1とnの間に半分以上存在する。

アルゴリズムAは、高々m回のW(b)のテストで終了し、合成数の判定は常に正しく、素数と判定した場合でも、それが誤りである確率は、1／2^{m}である。mの選び方は任意であるので、任意に精度を上げることが出来る。Karpは、この問題を解く鍵となった命題W(b)を満たすbを証人（witness）と呼んでいるが、この証人が大勢（少なくとも入力サイズの定数パーセント）いることがポイントである。又このアルゴリズムAの場合ランダム化の意味は、W(b)を満たすbの分布がランダムであることに起因している。この分布に偏りがあるならば、それに沿って抽出した方が当然効率は向上する等である。

3.2.5　新しい機械

(1)　確率機械

我々の良く使っている機械はすべて決定性のものである。つまり入力と機械の内部状態により次の処理は一意的に決ってしまう。この制限をゆるめて、計算の過程が一定の確率で生起する機械は確率オートマトンと呼ばれ、すでに研究の対象とされた。しかし、言語を受理する能力は決定性機械と同等であり〔De Leewe, Rabin 56〕その意味では新らしい可能性は生まれなかった。しかし、その処理速度が決定性機械よりも本当に格段に秀れている問題があることが最近証明されるに及んで、にわかに新らしい発展が注目されている。

ここではその動作原理と、主要な結果の一端を紹介する。

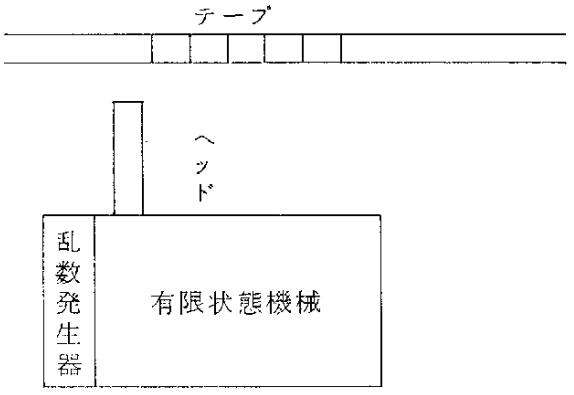


図３－９.　確率機械

通常の機械に０か１を確率的に発生する機構を備えたものを確率機械と呼ぶ（図３－９）。通常は０と１を等確率で乱数発生するものを考える。この機械では発生された乱数によって計算の過程が変るわけだから、計算過程は分布をもった確率事象としてとらえられる。従って計算の手間も確率変量である。

機械Ｍが言語Ｌを確率ｐで時間ｔ(x)で認識するとは、Ｍが任意の入力ｘに働くとき、次の事象の生起する確率がｐより大きいことであると定義する：機械

Mは t(x)時間以内に、もし $x \in L$ ならば1を出力して停止し、$x \in \overline{L}$ ならば0を出力して停止する。

このような機械の計算が意味を持つためには、上の確率 p は少くとも 1／2 より大きくなければならないし、1に近いことが勿論理想である。

例　集合 A_1 と A_2 がそれぞれ決定性機械 M_1 と M_2 で認識されるとき、集合 $A_1 \cap A_2$ を意味のある確率で認識する確率機械Mを作ることができる。Mの動作は次のように決める。任意の入力 x がきたとき、Mは乱数を発生させ、確率 α で M_1 の計算を真似し、残りの $1-\alpha$ の確率で M_2 の計算を真似る。M_i（$i=1$ と 2）が x に対して0を出力すれば（つまり $x \in A_i$ のとき）、Mは0を出力して停止する。M_i が1を出力したときは（つまり $x \in A_i$ のとき）、Mは確率 β で1を、残りの $1-\beta$ の確率で0を出力して停止する。α と β をうまく選ぶことにより、上の確率機械Mが集合 $A_1 \cap A_2$ を認識する能力（確率）を 2／3 より大きくできることが次のような考察よりわかる。今 $x \in A_1 \cap A_2$ の確率を q_1，$x \in A_2 \cap \overline{A}_1$ のそれを q_2，$x \in A_1 \cap A_2$ のそれを q_{12}，$x \in \bar{A}_1 \cap \bar{A}_2$（xが A_1 にも A_2 にも属さない）それを q_0 とすれば、上のMが集合 $A_1 \cap A_2$ を正しく認識する確率は、

$$p = 1 - (\alpha\beta q_1 + (1-\alpha)\beta q_2 + (1-\beta) q_{12})$$

となり、$q_0 + q_1 + q_2 + q_{12} = 1$ と合せて、$\alpha = 1/2$、$\beta = 2/3$ とすれば、q_1，q_2，q_{12} の値によらず $p > \frac{2}{3}$ を得る。この $\alpha = 1/2$、$\beta = 2/3$、$p > \frac{2}{3}$ はMの能力が一番悪い場合の判別能力が最大になるようにして決った値であることに注意しよう（これは相手の出方が全くわからないときは、こちらは最悪の場合の被害を最小にするように戦術 α と β を決定するのが最も良いという、ゲームの理論での min max 原理に対応する）。q_1，q_2，q_{12} があらかじめわかっていれば、解は上の場合とは異る。たとえば、q_2 が最小であれば、$\beta = 1$、$\alpha = 0$ のとき、つまり乱数を使わないで M_2 のみを真似る機械が最高の能力を発揮する。この例で見たように、確率機械は、計算過程の入力データに対する分布が不明のとき（現実問題ではたいていこれがあてはまる）

手間の平均値を減少させる。また、確率論の手法が解析に使えるので、理論的な見通しが良くなる。

R. Freivald は次の定理により、確率チューリング機械が決定性チューリング機械より計算速度が桁違いに速い場合があることを示した。

定理、任意の $\varepsilon > 0$ について、確率 $1-\varepsilon$ で、時間 $const \cdot |x| \cdot \log_2 |x|$ 以内に x が対称2進語であるかを判定する確率チューリング機械が存在する。

決定性のチューリング機械では x の長さ（$|x|$）がある程度大きくなると x が対称2進語であるのを判定するのに $const \cdot |x|^2$ の時間がどうしてもかかることが証明されている。

R. Freivald の確率チューリング機械は次のように動作する。左欄に動作を右欄にその動作に要する時間を記す。このアルゴリズムは、与えられた ε に対して定まる定数（自然数）c を使う。対称2進語の集合を D で表わす。

$x \in D$ を $const \cdot |x| \cdot \log_2 |x|$ 時間で判定する確率機械

1. 〔x の中心を求める〕 x の中心 Q（Q＝0，1 または λ）と左右の文字列 P', P'', $|P'| = |P''| = n$ を求める（$x = P'QP''$ とする）。	$const \cdot |x| \log_2 |x|$
2. 〔乱数の長さの計算〕 $\log_2 cn$ より大きい最小の整数 k を求める。	
3. 〔乱数発生〕 長さ k の2進乱数を発生させ、それを自然数 m の2進表現とみなす（$1 \leq m \leq 2^k$）。	
4. 〔素数判定〕 m が素数か判定する。素数でなければ段階3に行く	$cnost \cdot \sqrt{n} \cdot (\log_2 n)^2$

5. 〔N′≡N″（mod m）の判定〕 P′とP″ を2進数とみなした数をそれぞれN′とN″とする（ただし、P″では右の桁を上位とみる） N′≡N″（mod m）を決定性チューリング機械により判定する。もしこれが成立していないならx∈$\overline{D}$、もし成立していればx∈Dと判定する。	cnost·n·log n

説　明

素数定理により、乱数をconst·$(\log_2 n)^2$　回発生させれば、そうして得られたmの中に素数が含まれる確率は1－O(1)であることが示される。したがって素数mの生成と判定の手間const·$\sqrt{n}$·$(\log_2 n)^4$ はO(n)で押えられる。また、最後の判定でx∈Dが確率1－ε以上で保証されることは次の事実に基づいている。すなわち、任意のε＞0に対してある定数c（これがアルゴリズムの入力で使われたcである）が存在して、$\log_2 c$ n桁で表わされる2進数のうち素数であるものの個数全体に対して、このような素数のうち｜N′－N″｜、N′, N″≦2^n を整除するものの占める割合がεより小さくなることが証明されるからである。このアルゴリズムによって判定が加速された理由は、N′≡N″（長さn）の判定の代りに、長さが$\log_2 c$ nの2進列について、つまりN′≡N″（mod m）を判定することで代用できることによる。

ここでは〔Freivald　77〕の一部を紹介した。最初に得られに結果は手間の評価が｜x｜$(\log |x|)^2$ になっている（付録2参照）。

同じ著者により、確率アルゴリズムは整数や行列や多項式の乗算が正しく行なわれたかどうかを判定する手続きについても、決定性アルゴリズムより速いことが証明されている〔Freivald　79〕。

(2)　並列機械と並列アルゴリズム

一般にある仕事をk人で分担すれば1人でやったときに比べ能率がk倍より以上になることはない。仕事が逐次片づけていかねばならない性質のものであれば、一人でやろうとk人でやろうとかかる時間は同じである。一般の並列処

理では効率は仕事量が同じならば１／ｋと１の間のいずれかに落着くと思われる。しかし、仕事を与える代わりに、ある課題を与えるとすれば、ｋ人でやった方が一人でやった場合よりも、複数人という利点を生かしたやり方をすることにより、能率がｋ倍以上にあがる場合がありうる。ここでは、ｋ人並列処理が１人でやったときに比べｋ倍以上の能率をあげる場合、およびどんなに理想的に処理しても、$\frac{2}{3}$ ｋ以上にはできない場合の２つの例を示す。前者は分枝限定法（付録１参照）において、深さ優先の方針を採用したときに見られ〔今井、吉山、福村　７９〕、後者はｎ次の多項式を評価するときに証明されている〔Hyafil と Kung ７７〕 最後に整列や探索を専用に実行する機械についても最近の成果を簡単に述べる。

例１.　能率がｋ倍以上になる例

分枝限定法においては各部分問題を独立に解いてよいので、各部分問題に図3.－１１のような並列処理装置のいずれか一つを割当てて問題を解くというモデルを考える。例題として用いる探索木を図3.－１２に示す。問題は節０から出発して、コスト最小の節（図の節７）にできるだけはやくたどりつくことである。分枝限定法では、各節において、その節を木とする部分木の中のコストの下限を与える関数ｚが与えられている。一般に、探索は活性な節の集合のなかでｚの値が最小な節を展開する（つまり、その問題を部分問題に分割し、各部分問題に対応する子節をすべて生成し、それらを活性な節に合わせた集合を新たに活性な節の集合とする）という操作を操返して進行する。端節にはコストが置かれているとする。ここでは、活性な節のなかでレベルが最も深い節を次に展開するという方針を採用することにする。同じ深さのレベルの節が２つ以上あればｚの値の小さい方を採用する。このような探索は、深さ優先の探索と呼ばれる選択関数をｓで表わす。今最小化問題 P_0 が与えられたとき、上の方針に従った並列形分枝限定アルゴリズムを書き下すと次のようになる（このアルゴリズムの出力は、P_0 のすべての最適解の集合 X_0 とそのコスト Copt である）。このアルゴリズムは次の４つの変数を各処理装置が共通に参照する

（その競合は排他制御されるものとする）。

I　：未解決な部分問題に対応する節の集合

Xopt：最適解の候補の節の集合

Copt：Xoptにある節のコスト

act　：部分問題の処理を実行中の処理装置のリスト

初期設定：

$I \leftarrow \{P_0\}$，Xopt$\leftarrow\phi$，Copt$\leftarrow\infty$，act$\leftarrow 0$

p台の処理装置で並列に次の手続きを実行する；全ての処理装置が停止したとき、アルゴリズムは終了する。

手続き

1. 〔停止条件〕

(i) $I=\phi$かつact$=\phi$なら停止

(ii) $I=\phi$かつact>1なら1へ

(iii) $I\neq\phi$なら次へ

2. 〔部分問題の選択〕

act$\leftarrow$act$+1$，$P_i \leftarrow S(I)$，$I \leftarrow I-\{P_i\}$

（actに加えられた処理装置で選択関数Sで選んだP_iを処理する）

3. 〔部分問題の評価〕

(i) P_iが端節ならば6へ

(ii) P_iが端節でないならば次へ

4. 〔下界zの計算と部分問題の除去〕

(i) $z(P_i) > Copt$なら7へ

(ii) $z(P_i) \leq Copt$なら，Copt$\leftarrow z(P_i)$，次へ

5. 〔部分問題の分割〕

P_iを展開して得られるk個の部分問題をIに加える。7へ行く。

6. 〔最適解の候補の更新〕

(i) $$Xopt \leftarrow \begin{cases} Xopt, & \text{if } cost(P_i) \geq Copt \\ \{P_i\}, & \text{if } cost(P_i) < Copt \end{cases}$$

(ii)　Copt ← min (Copt , cost (P_i)) ;

(iii)　z (P_i) ＞ Copt を満たす P_i を全てＩから除く。

7.　〔部分問題 P_i の終了〕

act ← act － 1 とし、1へ行く

（手続き１終）

上の手順で処理装置１台を用いて例題を解いた場合と２台で解いた場合を以下に示す。

１台での探索過程

時間	1	2	3	4	5	6	7	8	9	10	11
処理装置１	0	2	6	13	5	11	1	4	9	3	⑦

２台での探索過程

時間	1	2	3	4
処理装置１	0	2	4	⑦
処理装置２	－	1	3	8

２台で処理した時の能率（速度）は１台の時より２倍以上よい。

活性な節のなかでｚの値最小の節を選択関数Ｓで選んだとしたら次のようになる

時間	1	2	3	4	5	6
処理装置１	0	2	1	4	3	⑦

能率が上がった理由

上の例題において、時間２が終了した段階で、２台並列に実行しているときは活性な節Ｉとして｛３（ｚ＝２），４（ｚ＝１），５（ｚ＝６），６（ｚ＝５）｝が得られ、これに対し１台だけではＩ＝｛５（ｚ＝６），６（ｚ＝５）｝である。つまり、２台処理装置を用いたことにより、コストの小さい部分問題が早い段階で探索され、これにより、手続きの段階４で部分問題が除去される可能性が増える。

最小被覆問題を上述の方法で解く場合のシミュレーション結果では、処理装

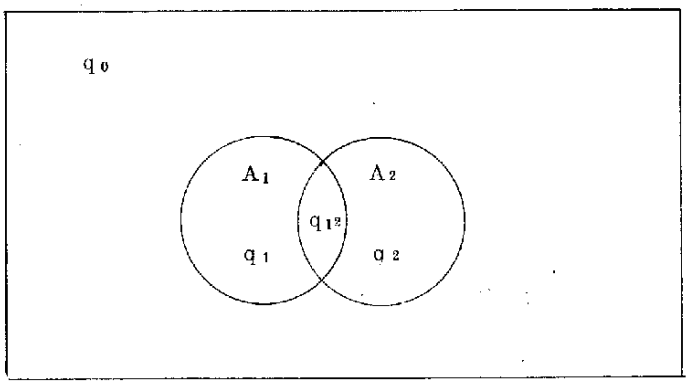


図３－１０

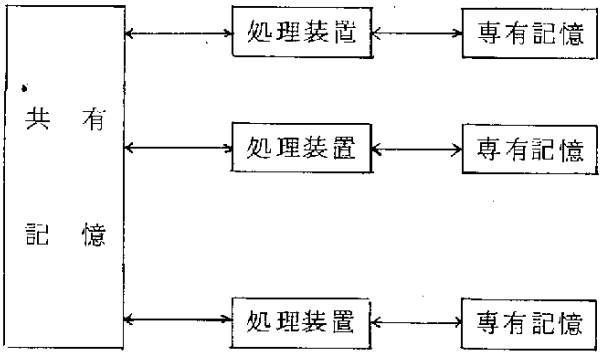


図３－１１　並列処理機械

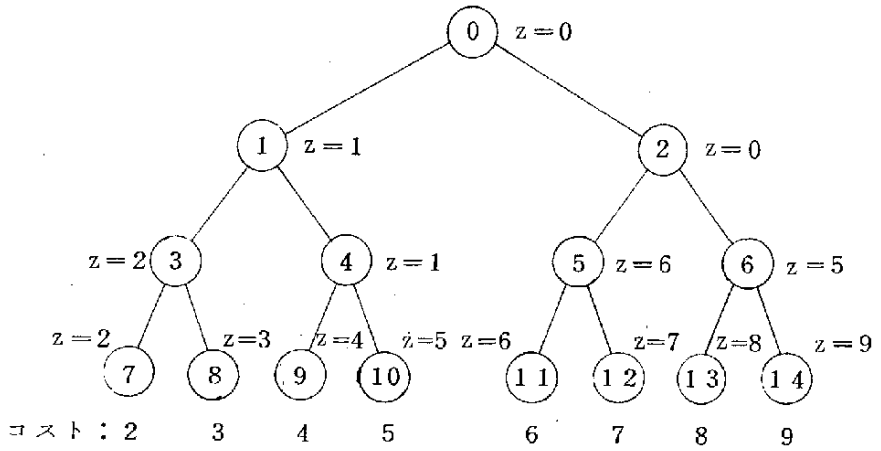


図３－１２　分枝限定法の木

置台数が２台から６４台ぐらいまでは、１台で解いた場合に比べ展開された節の全体の個数が約半分であると同じ著者らは報告している。この加速効果は深さ優先の探索においてのみみられることに注意しなければならない（この例題は〔今井、吉田、福村　７９）による）。

例 2.　能率が高々2／3k倍にしかならない例

$$H_7 = ((a_1 b_1 + a_2) b_2 + a_3) b_3 + a_4$$

の式を評価する場合を考える（このような ax＋b の形の式の結合したものは Horner式と呼ばれる）。単１処理装置で上の式を評価すれば６回の単位演算が必要になり、これ以下では評価することはできないことは証明されている（Horner式は最適な評価法である）。一方複数台の処理装置を使えば、最大限どれだけ速度を上げることができるだろうか。このような複数処理装置による評価は次のグラフにより表現することができる。

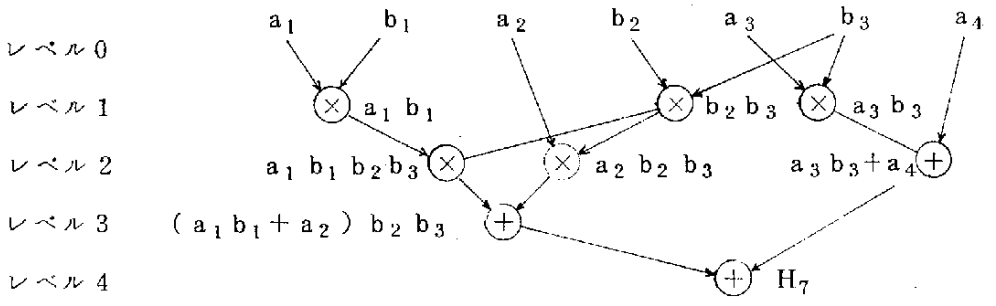


図 3－13

すなわち、レベル１では３台の処理装置により、$a_1 b_1$, $b_2 b_3$, $a_3 b_3$ が、レベル２では同じく３台の処理装置により、$a_1 b_1 b_2 b_3$, $a_2 b_2 b_3$, $a_3 b_3 + a_4$ が同時に評価される。この図では４単位の時間（３台の処理装置による並列演算）がH_7 を評価するのに必要である。この図では、同じデータが異った処理装置に入力されているし（通常これはできない）、また処理装置間のデータ転送時間も加味されていないことに注意しよう。

一般に$H_{2n+1} = (((a_1 b_1 + a_2) b_2 + a_3) + a_n) b_n + a_n + 1$ を k台の処理装置を用いて評価するとき、その処理速度は、高々$\frac{2}{3} k + \frac{1}{3}$ にしか増加しえないことをHyafil と Kung は証明した。

何故k倍でなく原理的に0.7 k倍にしか速度を上げ得ないかは、与えられた式において変数の依存関係が存在するからである。〔この例題は〔Hyafil, Kung 77〕による)。

(3) そ の 他

最後に整列や探索を行う特殊目的の機械を設計解析する試みにふれる。2次元配列状に、次の動作をする簡単な処理装置を並べて、それにより、n^2 個のデータをO(n)の手間で整列する〔Thompsn と Kung 77, Nassimi と Sahni 79〕:

〔Thompso と Kung 77, Nassimi と Sahni 79〕:

i) レジスタの中味を一斉に一定方向に隣の処理装置に送る

ii) 各処理装置内で2つのレジスタの内容の条件付き交換を行う。

付録１　　合枝限定法による解法

分枝限定法では解の構造を利用して、全ての決定木の集合を分割し、そのなかのどの部分集合に解が含まれているかをある評価関数に基づいて予想するという過程を繰返す。部分集合が細分され、最後にたった１つの要素を含んだ部分集合に至れば、これがすなわち解である。部分集合が小さくなるにつれて求める解の様子がくわしくわかるようになり、解にはなり得ない部分集合を探索の領域からはずすことができる。これが分枝限定法が有効に働く理由である。しかし、一体どれくらい有効であるかという評価を定量的にすることはむずかしい。

最小決定木を求める例題（図３－５）を分枝限定法により解く。決定表Ｔにおいて、たとえば $x_1 = 0$, $x_2 = 1$ として得られる部分決定表をＴ（１′，２）と表わす。表Ｔにおいて、変数 x_k が本質本数のとき $\delta_k(T) = 1$、それ以外を $\delta_k(T) = 0$ と書く。明らかに、表Ｔの決定木の内節の個数はＴの本質変数の個数より必らず多い。すなわち、図３－３に示す部分構造（これを未完の木と呼ぶ）から生成される決定木の値の１つの下限は、

Ｚ＝Ｔ（ｉ′）の本質変数の個数＋Ｔ(i)の本質変数の個数＋１

により与えられる。今未完の木のｚの値を z_0 とし、その木のどれか１つの部分表Ｔを図3.　3に示すように x_i で分けたとき、その結果の未完の木に対するｚは次の式により与えられる：

$$z = z_0 + v_i$$

ここに　$$v_i = \sum_{k \neq i} \delta_k(i') \, \delta_k(i) + 1 - \delta_i$$

δ_i はＴにおいて変数 x_i が本質変数ならば１、それ以外のときは０とする（Reinwald，Soland　６７）。この下限を用いて、最小決定木を求める分枝限定法の手順は次のように書くことができる。

Ｖを現在の未完の木を納めるための変数とする。

初期設定　Ｖ←λ（λにはＴが対応する）。

1. 〔終了か〕

Vが完全な決定木ならVが求める最小木である（手順は終了する）。

2. 〔Vを展開する〕

未完の木Vの寸法最大の各決定表から1つづつ変数を選び、それによってその表を分け未完の木を成長させる。次にその未完の木を納めた節Wj をVの子節として作る。変数の全ての組み合せについて、このような節Wj をつくり、その下限zを評価する。

3. 〔Vの更新〕

分枝限定により作成された木の現在の端節の集合のなかで、zの値が最小のものをVに代入し、1へ行く。

（手順終り）

この手順では、Vが一度完成された決定木に至れば、その木のコストとそこでのzの値は一致するから、そのコストより大きいコストを持った未完の木を持った節は、以後この手順では展開されることはない（これが"限定（bound）"の意味である）。このようにして探索の領域が制限される。

図3－14に上の手順に従って図3－5の表Tに対する最小木が求められる過程を示す。手順の段階3でzの値が同一のものが複数個あるときは、深さの深いものを優先して展開する。　の本質変数の個数は3個である。手順は、段階2で、Tの3つの変数x_1，x_2，x_3 で分けた未完の木をそれぞれ作成し、そのzの値を、それぞれz＝4，4，5と評価する（図3－15参照）。次に段階3でN2がVとして選ばれ、次の段階2でN2の子節として、N5，N6，N7，N8の節がつくられ、zがそれぞれ評価される。次の段階3におけるVの更新は、現在の木の端節｛N5，N6，N7，N8，N3，N4｝の中から、z最小でかつレベルが最も深く、かつ左側を優先して、N5がVとして選ばれる。N5の未完の木では、変数の個数が1以上の部分表はT（1，2′）のみであり、それをx_3で展開して、z＝コスト＝4の最小木を得る。

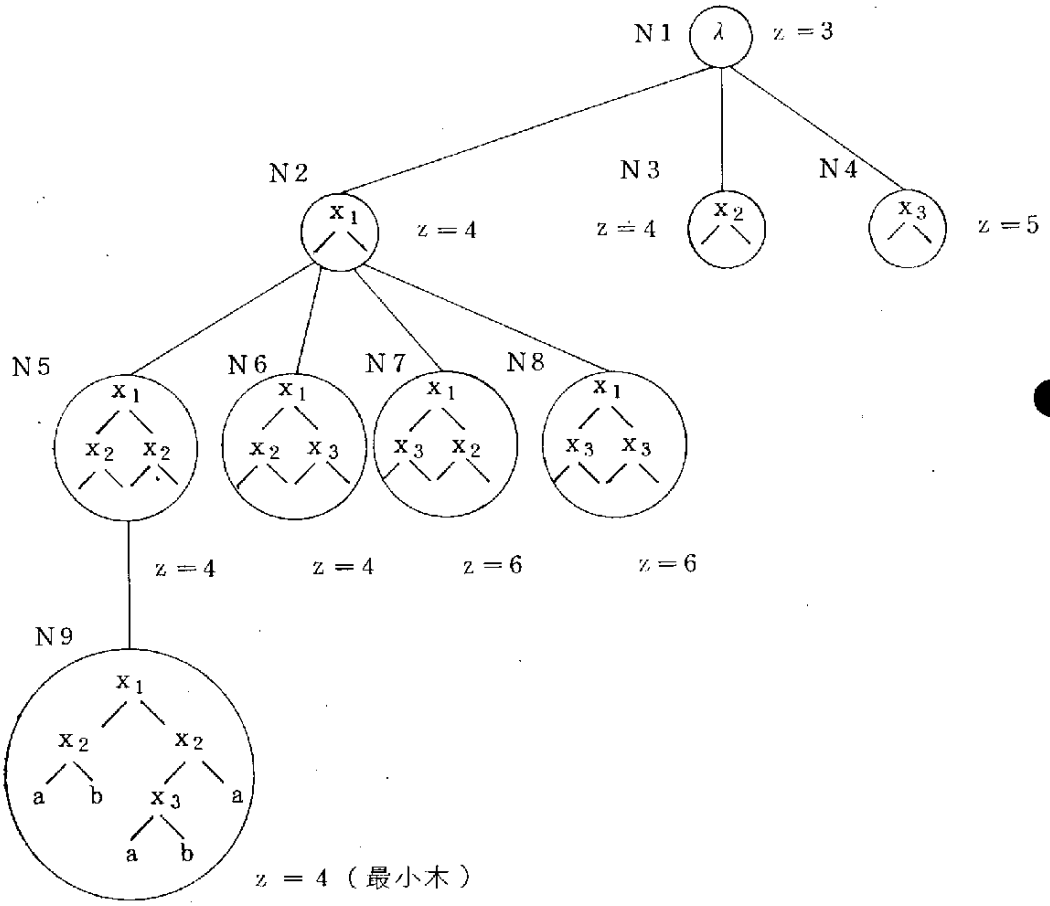


図３－１４　　分枝限定法の動作

T(1′) $x_1 = 0$

x_2	x_3	
0	0	a
0	1	a
1	0	b
1	1	b
δi 1	0	

T(1) $x_1 = 1$

x_2	x_3	
0	0	a
0	1	b
1	0	a
1	1	a
δi 1	1	

$v_1 = 1$

T(2′) $x_2 = 0$

x_1	x_3	
0	0	a
0	1	a
1	0	a
1	1	b
δi 1	1	

T(2) $x_2 = 1$

x_1	x_3	
0	0	b
0	1	b
1	0	a
1	1	a
δi 1	0	

$v_2 = 1$

T(3′) $x_3 = 0$

x_1	x_2	
0	0	a
0	1	b
1	0	a
1	1	a
δi 1	1	

T(3) $x_3 = 1$

x_1	x_2	
0	0	a
0	1	b
1	0	b
1	1	a
δi 1	1	

$v_3 = 2$

図 3 − 1 5　$v_i = \sum_{k \neq i} \delta_k(i') \, \delta_k(i) + 1 - \delta_i$

付録 2

Р.В.Фрейвалд :

"Быстрые вычисления на вероятностных машинах тьюринга,"

Теория алгоритмов и програм, выпуск 2, Ученые записи том 233,

Латвийский государственный университет, Рига, 1975, 201-205 .

R. V. フレイヴァルト："確率チューリング機械による高速計算"

アルゴリズムとプログラムの理論 第2巻

ラトビア国立大学学術報告２３３号，リガ，１９７５年

頁２０１－２０５

通常の１テープ・チューリング機械に、等確率１／２で文字０と１をベルヌーイ的に、つまり互いに独立に、発生する乱数発生器を備えたものを考える。このような機械を以下では確率チューリング機械と呼ぶ。

確率チューリング機械$\mathcal{M}$が集合Ａを、ｐ（$\frac{1}{2} \leq p < 1$）より大きい確率でｔ(x)時間以内に認識するとは、任意の語ｘの入力に対して$\mathcal{M}$が働くとき次の事象がｐより大きい確率で起ることである：$\mathcal{M}$がｔ(n)時間*以内に次の結果を出力して停止する

$$C_A(x) = \begin{cases} 1 , & x \in A \text{のとき} \\ 0 , & x \in \bar{A} \text{のとき} \end{cases}$$

確率チューリング機械$\mathcal{M}$が集合Ａを確率１－Ｏ(1)でｔ(x)時間以内に認識するとは、任意の$\varepsilon > 0$に対して、ある数ｎが存在して、長さｎ以上の任意の入力語ｘに対して$\mathcal{M}$が働くとき、１－εより大きい確率で次の事象が起ることである：$\mathcal{M}$はｔ(x)時間以内に結果$C_A(x)$を出力して停止する。

論文〔１〕の中でＪ．Ｍ．バーズディンは次のことを証明している：対称２進語**を認識する任意の決定性チューリング機械$\mathcal{M}$に対して、ある定数ｃ＞０が存在し、

有限個を除いた全てのnに対して、$\mathcal{M}$の働く時間が$c \cdot n^2$を越える長さnの入力語が存在する。

〔訳注〕*) チューリング機械にxを入力した始りの構成q_0xから停止した構成$x'q_f$に至る構成の列（これを計算と呼ぶ）の長さが$t(x)$である。

**) $\{0,1\}^*$の上のparlindrome

定理1. （1）任意の$\varepsilon > 0$に対して、$1-\varepsilon$より大きい確率で、時間$c|x| \cdot \log_2^2|x|$以内に対称2進語を認識する確率チューリング機械が存在する、ここに$|x|$は語xの長さを表わし、$c > 0$はxに依存しない定数である。

（2）確率$1-O(1)$で時間$c \cdot |x| \cdot \log_2^2|x| \cdot \log_2\log_2|x|$以内に対称2進語を認識する確率チューリング機械が存在する、ここに$|x|$は語xの長さを表わし、$c > 0$はxに依存しない定数である。

Bardzdinの定理と定理1を対比してみると、結果の正しさを高い確率で要求する場合でさえも、最も簡単な乱数発生器を使用することが計算速度を大幅に増大する可能性を与えることがわかる。この結果はB. A.トラフテンブロット〔2〕の問いのひとつに対する肯定的な答えである。

定理1の証明は次の2つの補題に依る。

補題1. $p_1, p_2, \cdots\cdots, p_k$を互いに異った素数として、

$$N' \equiv N'' \pmod{p_1}$$

$$N' \equiv N'' \pmod{p_2}$$

$$N' \equiv N'' \pmod{p_k},$$

が成立すれば、

$$N' \equiv N'' \pmod{p_1 \cdot p_2 \cdots\cdots p_k}$$

補題2. $N' \leq 2^n$, $N'' \leq 2^n$ かつ $N' \neq N''$ とする。このとき、定数 $c_1 > 0$ $c_2 > 0$ が存在して、最初の n^2 個の自然数の中に $\left(c_1 \cdot \frac{n^2}{\log_2 n} - c_2 \frac{n}{\log_2 n} \right)$ 個以上の $| N' - N'' |$ を剰余0で割り切る素数が存在する。

定理2. 確率チューリング機械がある $p > \frac{1}{2}$ より大きい確率で時間 $t(x)$ 以内に対称2進語を認識するならば、ある定数 $c > 0$ が存在して、無限個の語 x に対して、$t(x) > c \cdot |x| \cdot \log_2 |x|$。

B. A. トラフテンブロト〔3〕は、決定性チューリング機械により時間 $O(|x| \cdot \log_2 |x|)$ 以内で認識される集合は有限オートマトンで認識されることを証明した。

B_1 を $0^n 1 0^n$ ($n = 0, 1, 2 \cdots\cdots$) の型の語の集合とする。*

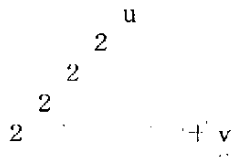
定理3. 確率 $1 - O(1)$ で時間 $c \cdot |x| \cdot \log_2 \log_2 |x|$ 以内で集合 B_1 を認識する確率チューリング機械が存在する、ここに $|x|$ は認識される語の長さを表わし、$c > 0$ は x に依存しない定数である。

定理4. ある $p > \frac{1}{2}$ を越える確率で時間 $t(x)$ 以内に確率チューリング機械が集合 B_1 を認識すれば、ある定数 $c > 0$ が存在して、無限個の語 x について

$$t(x) > c \cdot |x| \cdot \log_2 \log_2 |x|$$

定理3と4は確率チューリング機械に対してはトラフテンブロトの定理が成立しないことを意味している。J. Gill 〔4〕は次の予想を発表した：再帰的関数 f がある $p > \frac{1}{2}$ よりも大きい確率で $t(x)$ 時間以内に確率チューリング機械で計算可能ならば、ある決定性チューリング機械が存在して、ある $C > 0$ のもとで無限個の x について、$\tilde{t}(x) \leq c \cdot t(x)$ となる動作時間 $\tilde{t}(x)$ で f を計算する。

B_2 を次のような自然数 u と v が存在するような単一アルファベットからなる全ての語 x の集合とする：1) $0 \leq v < u$, 2) 語 x の長さは、



に等しい。

定理5. （1）集合 B_2 を確率 $1-Q(1)$ で時間 $c \cdot |x| \cdot \log_2 \log_2 |x|$ 以内に認識する確率チューリング機械が存在する、ここに $|x|$ は認識されている語の長さであり、$c>0$ は x に依存しない定数である。

（2）B_2 を認識する任意の決定性チューリング機械に対して、ある定数 $c>0$ が存在し、有限の個数を除いて、全ての語 x について、入力 x に対する機械の動作時間が $c \cdot |x| \cdot \log_2 |x|$ よりも大きい。

〔訳注〕 ＊） B_1 は有限オートマトンでは認識できない。

このようにして、定理5はJ. Gill の予想を覆している。

このノートは他の論文よりおくれて本巻に収められることになったものである。紙数の制限からここにあげた定理の証明を収めることができなかった。これらの証明は別の場所で公表するつもりである。

1. Барздинь Я.М. Сложность расповнавания симметрии на машинах Тьюринга.- В кн.: Проблемы кибернетики , 1965, вып. 15.

2. Трахтенброт Б.А. Замечания о сложности вычислений на вероятностных машинах. - В кн.: Исследования по теории алгоритмов и математической логике. М., 1974.

3. Трахтенброт Б.А. Сложность алгоритмов и вычислений. Новосибирск, 1967.

4. Gill J.T. III Computational complexity of probabilistic Turing machines. - Proc. 6-th Annual symposium ACM on theory of computing. 1974.

[Araki 76]

T. Araki, K. Taniguchi and T. Kasami, "Some NP-complete problems for Bounded Petri Nets", the transactions of the IECE of Japan, Vol. E 59, No. 7, July 1976.

〔荒木、都倉　７９〕

荒木俊郎、都倉信樹「フロー表現の等価問題の決定不能性」電子通信学会論文誌 '７９／６ Vol. J６２-D №6、PP４２７～

〔有沢　７９〕

有沢誠「チューリング機械による計算の極限収束について」電子通信学会論文誌 '７９／２ Vol. J６２-D №2、PP１６０～

[Adleman, Booth, Preparata, Ruzzo 78]

L. Adleman, K. S. Booth, F. P. Preparata, and W. L. Ruzzo,

"Improved time and space bounds for Boolean Matrix Multiplication", Acta Informatica 11, pp.61-75 (1978).

[Aho, Hopcroft, Ullman 75]
A. V. Aho, J. E. Hopcroft, J. D. Ullman, "The design and analysis of computer algorithms", Addison-Welley 1975.

[Ajtai, Komlós, Szemerédi 78]
M. Ajtai, J. Komlós and E. Szemerédi, "There is no fast single hashing algorithm", Information processing letter, Vol. 7, No. 6, October 1978, pp.270-

[Baker, Gill, Solovay 75]
T. Baker, J. Gill and R. Solovay, "Relativizations of the P = ?NP question", SIAM J. Comput. Vol. 4, No. 4, December 1975.

[Bauer, Blankinship 77]
W. R. Bauer, W. A. Blankinship, "A fast algorithm for certain dynamic programming minimizations", SIAM J. Appl. Math. Vol. 33, No. 2, September 1977, pp.323-

[Berman, Hartmanis 77]
L. Berman and J. Hartmanis, "On isomorphisms and density of NP and other complete sets", SIAM J. Comput. Vol. 6, No. 2, June 1977, pp.305-

[Bini, Capovani, Romani, Lotti 79]

D. Bini, M. Capovani, F. Romani, G. Lotti, "$O(n^{2.7799})$ complexity for n X n approximate matrix multiplication", Information processing letter, Vol. 8, No. 5, June 1979, pp.148-

[Blazewicz]

J. Blazewicz, "Simple algorithms for multiprocessor scheduling to meet deadlines", Information processing letters, Vol. 6, No. 5, pp.162-

[Brown, Tarjan 79]

M. R. Brown, R. E. Tarjan, "A fast merging algorithm", Journal of the Association for Computing Machinery, Vol. 26, No. 2, April 1979, pp.211-226

[Brélaz 79]

D. Brélaz, "New methods to color the vertices of a graph", Communication of the ACM, April 1979, Vol. 22, No. 4, pp.251-

[Brucker, Garey, Johnson 77]

P. Brucker, M. R. Garey and D. S. Johnson, "Scheduling equal-length tasks under treelike precedence constraints to minimize maximum lateness", Mathematics of operations research Vol. 2, No. 3, August 1977, Printed in U.S.A., pp.275-

[Bruno, Coffman Jr., Sethi 74]

J. Bruno, E. G. Coffman, Jr., and R. Sethi, "Scheduling independent tasks to reduce mean finishing time", Comm. ACM 17 (1974), pp.382-387.

[Chandra, Hirschberg, Wong 75]

A. K. Chandra, D. S. Hirschberg, C. K. Wong, "Approximate algorithms for the knapsack problem and its generalization", IBM research report RC 5616 (1975).

[Chandra, Wong 75]

A. K. Chandra and C. K. Wong, "Worst-case analysis of a placement algorithm related to storage allocation", SIAM J. Comput., Vol. 4, No. 3, September 1975, pp.249-263.

[Chang, Roberts 79]

E. Chang, R. Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processes", Communications of the ACM, May 1979, Vol. 22, No. 5, pp.281-

[Chin 78]

F. Y. Chin, "An O(n) algorithm for determining a near-optimal computation order of matrix chain products", Communication of the ACM, July 1978, Vol. 21, No. 7, pp.544-

[Cody, Coffman, Jr. 76]

R. A. Cody and E. G. Coffman, Jr., "Record allocation for minimizing expected retrieval costs on drum-like storage devices", J. ACM 23 (1976), pp.103-115.

[Cohen, Roth 76]

J. Cohen and M. Roth, "On the implementation of strassen's fast multiplication algorithm", Acta Informatica 6, pp.341-355 (1976).

[Christofides, Mingozzi, Toth, Sandi 79]

N. Christofides, A. Mingozzi, P. Toth, C. Sandi (ed.), "Combinatorial optimization", Wiley, 1979.

[Colbourn 79]

C. J. Colbourn, "The complexity of symmetrizing matrices", Information processing letter, Vol. 9, No. 3, 5 October 1979, pp.108.

[Cornvejols, Fisher, Nemhauser 75]

G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, "An analysis of heuristics and relaxations for the uncapacitated location problem", Cornell Univ. Dept. of Oper. Res. Tech. Rep. pp.271 (1975).

[Danaraj, Klee 77]

G. Danaraj, Victor Klee, "The connectedness game and the c-complexity of certain graphs", SIAM J. Appl. Math., Vol. 32, No. 2, March 1977, pp.431-

[De Leeuw K. 56]

K. De Leeuw, E. F. Moore, C. E. Shannon and N. Shapiro, "Computability by probabilistic machines", Automata Studies, Annals of Mathematics Studies No. 34, Princeton University Press, Princeton, NJ, 1956, pp.183-212.

[Floyd, Rivest 75]

Robert W. Floyd and Ronald L. Rivest, "Expected time bounds for selection", Communications of the ACM, March 1975, Vol. 18, No. 3, pp.165-

[Frederickson 79]

Greg N. Frederickson, "Approximation algorithms for some postman problems", Journal of the Association for Computing Machinery, Vol. 26, No. 3, July 1979, pp.538-554.

[Fredman 76]

Michael L. Fredman, "New bounds on the complexity of the shortest path problem", SIAM J. Comput. Vol. 5, No. 1, March 1976.

[Freivald 75]

R. V. Freivald, "Fast computation by probabilistic Turing machines", theory of algorithms, No. 2, Latvian State University, Riga, pp.201-205 (in Russian) (1975).

[Freivald 77]

Rúsins Freivalds, "Probabilistic machines can use less running time", Information processing 77, B. Gilchrist, Editor IFIP, North-Holland publishing company (1977).

[Freivald 79]

Rúsins Freivalds, "Fast probabilistic algorithms", Proceedings of math. theory of computation at Czechoslovakia 79, Springer lecture note.

[Fussenegger, Gabow 79]

Frank Fussenegger, Harold N. Gabow, "A counting approach to lower bounds for selection problems", Journal of the Association for Computing Machinery, Vol. 26, No. 2, April 1979, pp.227-238.

[Gabow 78]

H. N. Gabow, "A good algorithm for smallest spanning trees with a degree constraint", Networks, Vol. 8 (1978), pp.201-208.

[Galil 76]

Zvi Galil, "Hierarchies of complete problems", Acta Informatica 6, pp.77-88 (1976).

[Galil 79]

Zvi Galil, "On improving the worst case running time of the Boyer-Moore string matching algorithm", Communications of the ACM, September 1979, Vol. 22, No. 9, pp.505-

[Galil, Megiddo 79]

Avi Galil, Nimrod Megiddo, "A fast selection algorithm and the problem of optimum distribution of effort", Journal of the Association for Computing Machinery, Vol. 26, No. 1, January 1979, pp.58-64.

[Garfinkel, Gilbert 78]

R. S. Garfinkel, K. C. Gilbert, "The bottleneck traveling salesman problem: Algorithms and Probabilistic analysis", Journal of the Association for Computing Machinery, Vol. 25, No. 3, July 1978, pp.435-448.

[Garey, Graham 74]

M. R. Garey and R. L. Graham, "Performance bounds on the splitting algorithm for binary testing", Acta. Informatica, 3 (1974), pp.347-356.

[Garey, Graham 75]
M. R. Garey and R. L. Graham, "Bounds for multi-processor scheduling with resource constraints", SIAM J. on Computing, 4 (1975), pp.187-200.

[Garey, Graham, Johnson 76]
M. R. Garey, R. L. Graham and D. S. Johnson, "Some *NP*-complete geometric problems", Proc. 8th ACM Symposium on Theory of Computing, pp.10-22 (1976).

[Garey, Graham, Johnson 77]
M. R. Garey, R. L. Graham and D. S. Johnson, "The complexity of computing steiner minimal trees", SIAM J. Appl. Math., Vol. 32, No. 4, June 1977, pp.835-

[Garey, Graham, Johnson, Knuth 78]
M. R. Garey, R. L. Graham, D. S. Johnson and D. E. Knuth, "Complexity results for bandwidth minimization", SIAM J. Appl. Math., Vol. 34, No. 3, May 1978, pp.477-

[Garey, Johnson 76]
M. R. Garey, D. S. Johnson, "Scheduling tasks with nonuniform deadlines on two processors", Journal of the Association for Computing Machinery, Vol. 23, No. 3, July 1978, pp.461-467.

[Garey, Johnson 76]

M. R. Garey and D. S. Johnson, "The complexity of near-optimal graph coloring", J. ACM, 23, pp.43 (1976).

[Garey, Johnson 77]

M. R. Garey, D. S. Johnson, "The rectilinear steiner tree problem is *NP*-complete", SIAM J. Appl. Math. Vol. 32, No. 4, June 1977, pp.826-

[De Millo, Vairavan, Sycara-Cyranski 77]

R. A. Demillo, K. Vairavan, E. Sycara-Cyranski, "A study of schedules as models of synchronous parallel computation", Journal of the Association for Computing Machinery, Vol. 24, No. 4, October 1977, pp.544-565.

[Dionne, Florian 79]

R. Dionne, M. Florian, "Exact and approximate algorithms for optimal network design", Networks, Vol. 9 (1979), pp.37-59, 1979 John Wiley & Sons, Inc.

[Dixon, Goodman 76]

E. T. Dixon, S. E. Goodman, "An algorithm for the longest cycle problem", Networks, 6: pp.139-149 1976 by John Wiley & Sons, Inc.

[Dobkin, Leeuwen 76]

David Dobkin, Jan Van Leeuwen, "The complexity of vector-products", Information processing letters, Vol. 4, No. 6, March 1976, pp149-

[Dobkin, Lipton, Reiss 79]

David Dobkin, Richard J. Lipton and Steven Reiss, "Linear programming is log-space hard for P", Vol. 8, No. 2, Information processing letters, February 1979, pp.96-

[Dreyfus 69]

S. E. Dreyfus, "An appraisal of some shortest-path algorithms", Operations Research, 17 (1969), pp.395-412.

[Easton, Wong 75]

M. C. Easton and C. K. Wong, "The effect of a capacity constraint on the minimal cost of a partition", J. ACM 22 (1975), pp.441-499.

[Ellis 75]

Clarence A. Ellis, "Optimal scheduling of homogeneous job systems", Information Sciences 9, pp.323-358 (1975).

[Even, Tarjan 76]

S. Even, R. E. Tarjan, "A combinatorial problem which is complete in polynomial space", Journal of the Association for Computing Machinery, Vol. 23, No. 4, October 1976, pp.710-719.

[Garey, Johnson 79]

M. R. Garey and D. S. Johnson, "Computers and intractability - a guide to the theory of *NP*-completeness", Freeman 1979.

[Garey, Johnson, Stockmeyer 74]

M. R. Garey, D. S. Johnson and L. Stockmeyer, "Some simplified *NP*-complete problems", Proc. 6th ACM Symp. on Theory of Computing, pp.47 (1974).

[Garey, Johnson, Tarjan 76]

M. R. Garey, D. S. Johnson and R. Endre Tarjan, "The planer hamiltonian circuit problem is *NP*-complete", SIAM J. Comput, Vol. 5, No. 4, December 1976, pp.704-

[Gavril 78]

F. Gavril, "A recognition algorithm for the total graphs", Networks, Vol. 8 (1978) pp.121-133

[Gentleman 78]

W. Morven Gentleman, "Some complexity results for matrix computations on parallel processor", Journal of the Association for Computing Machinery, Vol 25, No. 1, January 1978, pp.112-115.

[Gill 74]

J. T. Gill III, "Computational complexity of probabilistic Turing machines", Proc. 6th ACM symposium on theory of computing, pp.91-95 (1974).

[Gill 77]

John Gill, "Computational complexity of probabilistic Turing machines", SIAM J. Comput, Vol. 6, No. 4, December 1977, pp.675-695.

[Gonzalez, Ibarra, Sahni 75]

T. Gonzalez, O. H. Ibarra, and S. Sahni, "Bounds for LPT schedules on uniform processors", Univ. of Minn. Comp. Sci. Tech. Rep. 75-1 (1975).

[Gonzalez, Sahni 75]

T. Gonzalez and S. Sahni, "Flow shop and job shop schedules", Univ. of Minn. Comp. Sci. Tech. Rep. 75-14 (1975).

[Gonzalez, Sahni 78]

T. Gonzalez and S. Sahni, "Flowshop and jobshop schedule: Complexity and approximation", OPERATIONS RESEARCH, Vol. 26, No. 1, January-February 1978, pp.36-

[Goto, Vincentelli 78]

S. Goto, A. Sangiovanni-Vincentelli, "A new shortest path updating algorithm", Networks, Vol. 8 (1978) pp.341-372.

[Graham 66]

R. L. Graham, "Bounds for certain multiprocessing anomalies", Bell Sys. Tech. Jour., 45 (1966), pp.1563-1581.

[Graham 69]

R. Graham, "Bounds on multiprocessor timing anomalies", SIAM Jr. on Appl. Math., 17(2), pp.416-429, 1969.

[Graham 72]

R. L. Graham, "Bounds on multiprocessing anomalies and related packing problems", Proceedings of the Spring Joint Computer Conference (1972), pp.205-217. A survey of bounds for scheduling and packing algorithms, as of 1971.

[Graham 74]

R. L. Graham, "Bounds on the performance of scheduling algorithms", in E. G. Coffman (ed.), Computer and Job-Shop Scheduling, John Wiley and Songs, 1975. A survey of bounds for scheduling and packing algorithms, as of 1974.

[Gurari, Ibarra 79]

Eitan M. Gurari and Oscar H. Ibarra, "An *NP*-complete number-theoretic problem", Journal of the Association for Computing Machinery, Vol. 26, No. 3, July 1979, pp.567-581.

[Hadlock 77]

F. O. Hadlock, "A shortest path algorithm for grid graphs", Networks, 7, pp.323-334.

[Hausmann, Korte 78]

Dirk Hausmann and Bernhard Korte, "Lower bounds on the worst-case complexity of some oracle algorithms", Discrete Mathematics 24 (1978) pp.261-276.

[Hikita, Goto 77]

Teruo Hikita and Eiichi Goto, "An O(*N*) algorithm for finding periodicity of a sequence using hash coding", Information Processing Letters, Vol. 6, No. 2, April 1977, pp.69-

[Hirschberg 75]

D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences", Communications of the ACM, June 1975, Vol 18, No. 6, pp.341-

[Hirschberg 77]

Daniel S. Hirschberg, "Algorithms for the longest common subsequence problem", Journal of the Association for Computing Machinery, Vol. 24, No. 4, October 1977, pp.664-675.

[Hirshberg 78]

D. S. Hirschberg, "An information-theoretic lower bounds for the longest common subsequence problem", Information Processing Letters, Vol. 7, No. 1, January 1978, pp.140-

[Hirschberg, Chandra, Sarwate 79]

D. S. Hirschberg, A. K. Chandra, D. V. Sarwate, "Computing connected components on parallel computers", Communications of the ACM, August 1979, Vol. 22, No. 8, pp.461-

[Hopcroft, Karp 73]

J. E. Hopcroft and R. M. Karp, "A $N^{5/2}$ algorithm for maximum matchings in bipartite graphs", SIAM J. Comput., 2 (1973) pp.225-231.

[Hopcroft, Paul, Valiant 77]

John Hopcroft, Wolfgang Paul and Leslie Valiant, "On time versus space", Journal of the Association for Computing Machinery, Vol. 24, No. 2, April 1977; pp.332-337.

[Hopcroft, Tarjan 74]

John Hopcroft and Robert Tarjan, "Efficient planarity testing", JACM, Vol. 21, No. 4, October 1974, pp.549-568.

[Horibe 77]

Yasuichi Horibe, "An improved bound for weight-balanced tree", INFORMATION AND CONTROL 34, pp.148-151 (1977).

[Horowitz, Sahni 76]

Ellis Horowitz and Sartaj Sahni, "Exact and approximate algorithms for scheduling nonidentical processors", Journal of the Association for Computing Machinery, Vol. 23, No. 2, April 1978, pp.317-327.

[Horowitz, Sahni 78]

E. Horowitz and S. Sahni, "Fundamentals of computer algorithms", Computer Science Press. 1978.

[Horvath 78]

E. Horvath, "Stable sorting in asymptotically optimal time and extra space", Journal of the Association for Computing Machinery, Vol. 25, No. 2, April 1978, pp.177-199.

[Hunt, Szymanski 77]

James W. Hunt and Thomas G. Szymanski, "A fast algorithm for computing longest common subsequences", Communication of the ACM, May 1977, Vol. 20, No. 5, pp.350-

[Hwang 79]

F. K. Hwang, "An O(n log n) algorithm for rectilinear minimal spanning trees", Journal of the Association for Computing Machinery, Vol. 26, No. 2, April 1979, pp.177-182.

[Hyafil, Rivest 76]

L. Hyafil, R. L. Rivest, "Constructing optimal binary tree is *NP*-complete", Information Processing Letters, Vol. 5, No. 1, May 1976, pp.15-17.

[Hyafil, Kung 77]

L. Hyafil and H. T. Kung, "The complexity of parallel evaluation of linear recurrences", Journal of the Association for Computing Machinery, Vol. 24, No. 3, July 1977, pp.513-521.

[Hwang 76]

F. K. Hwang, "On steiner minimal trees with rectilinear distance", SIAM J. Appl. Math., 30 (1976), pp.104-114.

[Ibarra, Kim 75]

O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors", Univ. of Minn. Comp. Sci. Tech. Rep. 75-7 (1975).

[Ibarra, Kim 75]

O. H. Ibarra and C. E. Kim, "Fast approximation algorithms for the knapsack and sum of subset problems", JACM, Vol. 22, No. 4, October 1975, pp.463-468.

〔茨木、亀田、樋田　７９〕

茨木俊秀、亀田恒彦、樋田俊一" ＮＰ－ complete diagnosis problems on system graphs" The transactions of The Institute of Electronics and Communication Engineers of Japan (Section E) Vol. E６２、№2、Feb、1979.

〔衣斐、森田、菅田　７９〕

衣斐寛之、森田憲一、菅田一博「テープ限定チューリング変換機の計算能力」電子通信学会論文誌 '７９／２ Vol. Ｊ６２-Ｄ №2、ＰＰ１１１～

[Igarashi 77]

Y. Igarashi, "General properties of derivational complexity", Acta Informatica 8, pp.267-283 (1977).

〔五十嵐　８０〕

五十嵐善英「確率的アルゴリズムの概観」情報処理 Vol.２１、№1、１９８０／１、ＰＰ１３～１８

〔今井　７９〕

今井正治、吉田雄二、福村晃夫「分枝限定アルゴリズムの並列化とその評価」電子通信学会論文誌 '７９／６ Vol. J６２－D №6、PP４０３～４１０。

[Itai, Rodeh, Tanimoto 78]

A. Itai, M. Rodeh and S. L. Tanimoto, "Some matching problems for bipartite graphs", Journal of the Association for Computing Machinery, Vol. 25, No. 4, October 1978, pp.517-525.

[Jackowski, Kubiak, Sokolowski 77]

B. L. Jackowski, R. Kubiak and S. Sokolowski, "Complexity of sorting by distributive partitioning", Information Processing Letters, Vol. 9, No. 2, August 1977, pp.80.

[Johnson 73]

D. S. Johnson, "Near-optimal bin packing algorithms", Doctoral Thesis, Mathematics Department, Mass. Inst. of Tech., Cambridge, Mass., 1973.

[Johnson 74]

D. S. Johnson, "Approximation algorithms for combinatorial problems", J. Comput. Systems Sci., 9 (1974), pp.256-278.

[Johnson 74]

D. S. Johnson, "Worst case behavior of graph coloring algorithms", Proceedings of the Fifth Southeastern Conference on Combinatorics, Graph Theory, and Computing, Utilitas Mathematica Publishing, Winnipeg, 1974, pp.513-527.

[Johnson 74]

D. S. Johnson, "Fast algorithms for bin packing", J. Comput. Systems Sci., 8 (1974), pp.272-314.

[Johnson, Demers, Ullman, Garey, Graham 74]

D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey and R. L. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms", SIAM J. on Computing, 3 (1974), pp.299-325.

[Johnson, Kashdan 78]

D. B. Johnson and S. D. Kashdan, "Lower bounds for selection in $X + Y$ and other multisets", Journal of the Association for Computing Machinery, Vol. 25, No. 4, October 1978, pp.556-570.

[Kafura 74]

D. G. Kafura, "Analysis of scheduling algorithms for a model of a multiprocessing computer system", Doctoral Thesis, Computer Science Department, Purdue University 1974.

[Kafura, Shen 74]

D. G. Kafura and V. Y. Shen, "Scheduling independent processors with different storage capacities", Proceedings ACM National Conference (1974).

〔金田 78〕

金田悠紀夫「並列処理システムによる線形計画計算と実対称行列の三重対角比計算」情報処理 Vol. 19 No.1、1978／1、PP39～

[Kariv, Hakimi 79]

O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. I: The p-centers", SIAM J. Appl. Math., Vol. 37, No. 3, December 1979, pp.513-

[Kariv, Hakimi 79]

O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. II: The p-medians", SIAM J. Appl. Math., Vol. 37, No. 3, December 1979, pp.539-

[Karp 72]

R. M. Karp, "Reducibility among combinatorial problems" in "Complexity of Computer Computations (R.E. Miller and J. W. Thatcher, eds.)", Plenum Press, pp.85 (1972).

[Karp 75]

R. M. Karp, "The fast approximation solution of hard combinatorial problems", Proc. 6th Southeastern Conf. Combinatorics, Graph Theory and Computing, pp.15 (1975).

[Karp 76]

R. M. Karp, "The probabilistic analysis of some combinatorial search algorithms", in Algorithms and Complexity : New Directions and Recent Results, J. F. Traub ed., Academic Press, New York, pp.1-19 (1976).

[Karp 77]

R. M. Karp, "Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane", Mathematics of Operations Research, Vol. 2, No. 3, August 1977, Printed in U.S.A., pp.209-

[Karp 78]

R. M. Karp, "An algorithm to solve the m x n assignment problem in expected time $O(mn \log n)$", Memorandum No. UCB/ERL M 78/67, Electronics Research Laboratory, University of California, Berkley (1978).

[Karp 79]

R. M. Karp, "Probabilistic analysis of canonocal numbering algorithm for graphs", Proc. AMS Symposia in Applied Mathematics, Vol. 34 (1979).

[Karp 79]

R. M. Karp, "A patching algorithm for the nonsymmetric traveling-salesman problem", to appear in SIAM J. Computing (1979).

[Karp 79]

R. M. Karp, "Recent advances in the probabilistic analysis of graph-theoretic algorithms", Proc. 6th Colloquium on Automata, Languages and Programming, pp.338-339 (1979).

[Karp, McKellar, Wong 75]

R. M. Karp, A. C. McKellar, C. K. Wong, "Near-optimal solutions to a 2-dimensional placement problem", SIAM J. on Computing, 4 (1975), pp.271-286.

[Kasai, Adachi, Iwata 79]

T. Kasai, A. Adachi and S. Iwata, "Classes of pebble games and complete problems", SIAM J. Comput., Vol. 8, No. 4, November 1979, pp.574-

〔嵩　７７〕

嵩忠雄「ソフトウエアの進歩とその基礎理論の発展」電子通信学会誌8／'７７、Vol.868、創立６０周年記念特集、ＰＰ８６８～

〔葛山、谷口、嵩　７６〕

葛山善基、谷口健一、嵩忠雄「サブルーチン化によるプログラム形の簡易化」電子通信学会論文誌 '７６／５、Vol. Ｊ５９－Ｄ №5、ＰＰ３３９～

〔葛山、谷口、嵩　７６〕

葛山善基、谷口健一、嵩忠雄「プログラムモジュール群の再構成」電子通信学会論文誌 '７６／１１、Vol. Ｊ５９－Ｄ №11、ＰＰ７７０～７７７

[Kaufman 74]

M. T. Kaufman, "An almost-optimal algorithm for the assembly line scheduling problem", IEEE Trans. on Comp., C-23 (1974) pp.1169-1174.

[Kelton, Law 78]

W. D. Kelton and A. M. Law, "A mean-time comparison of algorithms for the all-pairs shortest-path problem with arbitrary arc lengths", Networks, Vol. 8 (1978) pp.97-106.

〔木村、原尾、野口　７９〕

木村春彦、原尾政輝、野口正一「１レジスタ機械における直列形プログラムのコード最適化」電子通信学会論文誌 '７９／４、Vol. Ｊ６２－Ｄ №4、Ｐ２８１～

〔木村、原尾、野口　７９〕

木村春彦、原尾政輝、野口正一「直列形プログラムのコード最適化 ―1レジスタ機械におけるアルゴリズムの複雑さ ―」電子通信学会論文誌 '７９／８、Vol. J６２－D №8、PP５１５～

[Knuth 72]

D. E. Knuth, "The art of computer programming", Volume 1/ Fundamental algorithms, Volume 2/ Semi numerical algorithms, Volume 3/ Sorting and searching, Addisson Wesley, 1972.

[Kou, Stockmeyer, Wong 78]

L. T. Kou, L. J. Stockmeyer and C. K. Wong, "Covering edges by cliques with regard to keyword conflicts and intersection graphs", Communications of the ACM, February 1978, Vol. 21, No. 2, pp.135-

[Krause 73]

K. L. Krause, "Analysis of Computer Scheduling with Memory Constraints", Doctoral Thesis, Computer Science Dept., Purdue University, 1973.

[Krause, Shen, Schwetman 75]

K. L. Krause, V. Y. Shen, and H. D. Schwetman, "Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems", J. ACM 22 (1975), pp.522-550.

[Krishnamoorthy, Deo 79]

M. S. Krishnamoorthy and N. Deo, "Complexity of the minimum-dummy-activities problem in a Pert Network", Networks, Vol. 9 (1979) pp.189-194.

[Kronsjo 79]

L. I. Kronsjo, "Algorithms - Their complexity and efficiency", John Wiley & Sons, Inc. 1979.

[Kuck, Maruyama 75]

D. J. Kuck and K. Maruyama, "Time bounds on the parallel evaluation of arithmetic expressions", SIAM J. Comput., Vol. 4, No. 2, June 1975, pp.147-

[Kung, Luccio, Preparata 75]

H. T. Kung, F. Luccio and F. P. Preparata, "On finding the maxima of a set of vectors", Journal of the Association for Computing Machinery, Vol. 22, No. 4, October 1975, pp.469-476.

[Lawler 75]

E. L. Lawler, "Algorithms, graphs, and complexity", Networks, 5: pp.89-92.

[Lawler 76]

E. Lawler, "Combinatorial optimization - networks and matroids", Holt. Rinehart Winston, 1976.

[Lawler, Labetoulle 78]

E. L. Lawler, J. Labetoulle, "On preemptive scheduling of unrelated parallel processors by Linear Programming", Journal of the Association for Computing Machinery, Vol. 25, No. 4, October 1978, pp.612-619.

[Lenstra, Kan 76]

J. K. Lenstra, A. H. G. Rinnooy Kan, "On general routing problems", Networks, 6: pp.273-280.

[Lipski 77]

W. Lipski, Jr., "One more polynomial complete consecutive retrieval problem", Information Processing Letters, Vol. 6, No. 3, June 1977, pp.91-

[Lipton, Tarjan 79]

Richard J. Lipton, Robert Endre Tarjan, "A separator theorem for planar graphs", SIAM J. Appl. Math., Vol. 36, No. 2, April 1979, pp.177-

[Liu 75]

C. L. Liu, "Approximation algorithms for discrete optimization problems", Proceedings of the Fourth Texas Conference on Computing Systems (1975).

[Liu, Liu 74]

J. W. S. Liu and C. L. Liu, "Bounds on scheduling algorithms for heterogeneous computing systems", Proc. of the 1974 IFIP Cong., North Holland Publishing Company (1974), pp.349-353.

[Lueker, Booth 79]

George S. Lueker, Kellogg S. Booth, "A linear time algorithm for deciding interval graph isomorphism", Journal of the Association for Computing Machinery, Vol. 26, No. 2, April 1979, pp.183-195.

[Lynch 75]

Nancy Lynch, "On reducibility to complex or sparse sets", Journal of the Association for Computing Machinery, Vol. 22, No. 3, July 1975, pp.311-315.

[Manacher 75]

Glenn Nanacher, "A new linear-time 'On-Line' algorithm for finding the smallest initial palindrome of a string", Journal of the Association for Computing Machinery, Vol. 22, No. 3, July 1975, pp.345-351.

[Manacher 79]

Glenn K. Manacher, "The Ford-Johnson sorting algorithm is not optimal", Journal of the Association for Computing Machinery, Vol. 26, No. 3, July 1979, pp.441-456.

[Miyakawa, Sabelfeld 77]

M. Miyakawa, V. K. Sabelfeld, "Some results on the complexity of program schemes (Russian)", Computing Center, Siberian Div. of Academy of Sciences of USSR, 1977.

[Misra, Gries 78]

Jayadev Misra, David Gries, "A linear sieve algorithm for finding prime numbers", Communications of the ACM, December 19, Vol. 21, No. 12, pp.999-

[Moulin 76]

Hervé Moulin, "Cooperation in mixed equilibrium", MATHEMATICS OF OPERATIONS RESEARCH, Vol. 1, No. 3, August, 1976, Printed in U.S.A., pp.273-

[H. Mori, H. Aiso]

H. Mori, H. Aiso, "Parallel processing of the FFT by an array processor", The transactions of the Institute of Electronics and Communication Engineers of Japan, Vol. E61, No. 2, Feb., 1978.

〔森田、梅尾、衣斐、菅田　７８〕

森田憲一、梅尾博司、衣斐寛之、菅田一博「２次元チューリング機械におけるテープ計算量の下界について」電子通信学会論文誌 '７８／６、vol.Ｊ６１－Ｄ №6、Ｐ３８１～

〔森田、梅尾、菅田　７７〕

森田憲一、梅尾博司、菅田一博「各種の２次元テープオートマトンの言語受理能力とテープ計算量の関係」電子通信学会論文誌 '７７／１２、vol.Ｊ６０－Ｄ №12、Ｐ１０７７～

[Muller, Preparate 75]

David E. Muller and Franco P. Preparata, "Bounds to complexities of networks for sorting and for switching", Journal of the Association for Computing Machinery, Vol. 22, No. 2, April 1975, pp.195-201.

[Murphy, Paull 79]

Paul E. Murphy and Marvin C. Paull, "Minimum comparison merging of sets of approximately equal size", Information and Control 42, pp.87-96 (1979).

〔永松、本多　７７〕

永松正博、本多波雄「有限個のモジュールにより生成される流れ図の G_0 T_0 文最小のプログラム記述について」電子通信学会論文誌 '７７／９、Vol. Ｊ６０－Ｄ №9、ＰＰ６９４～７０１。

[Nassimi, Sahni 79]

Bitonic Sort on a Mesh-Connected Parallel Computer, David Nassimi and Sartaj Sahni, IEEE Transactions on Computers, Volume C-27 No. 1, January 1979, pp.2-7

〔永松、本多　７７〕

永松正博、本多波雄「ＢＪ流れ図のG$\bar{o}$ T$\bar{o}$文最小のプログラム記述とあるＮＰ完全性問題」電子通信学会論文誌 '７７／９、Vol. Ｊ６０－Ｄ №9、ＰＰ７０２～７０９

[Nemhauser, Yu 72]

G. L. Nemhauser and P. L. Yu, "A problem in bulk service scheduling", Oper. Res., 20 (1972), pp.813-819.

〔祢津　７７〕

祢津孔二「デッドロックデテクタ」電子通信学会論文誌 '７７／１０、Vol. Ｊ６０－Ｄ №10、Ｐ８０９～

〔祢津　７８〕

祢津孔二「簡易化したデッドロックデテタク」電子通信学会論文誌 '７８／９、Vol. Ｊ６１－Ｄ №9、Ｐ７１９～

[Nigmatullion 69]

R. G. Nigmatullin, "The fastest descent method for covering problems" (in Russian), Proceedings of a Symposium on Questions of Precision and Efficiency of Computer Algorithms, Book 5, Kiev (1969), pp.116-126.

〔西岡、高見沢、斉藤　７７〕

西岡隆夫、高見沢一彦、斉藤伸自「Gō Tō文最小プログラム記述の計算手数について」電子通信学会論文誌 '７７／３、 Vol. J６０－D №3、P２０１～

[Nozaki 79]

Akihiro Nozaki, "A note on the complexity of approximative evaluation of polynomials", Information Processing Letters, Vol. 9, No. 2, 17 August 1979, pp.73-

[Orloff 76]

C. S. Orloff, "On general routing problems: Comments", Networks, 6: pp.281-284.

〔大附　７９〕

大附辰夫「アルゴリズムの複雑度の理論 ― グラフ理論の応用小特集６」

Vol. ６２ №7、P７８９～

[Paradimitriou 76]

Christos H. Papadimitriou, "On the complexity of edge traversing", Journal of the Association for Computing Machinery, Vol. 23, No. 3, July 1976, pp.544-554.

[Pierce 75]

A. R. Pierce, "Bibliography on algorithms for shortest path, shortest spanning tree, and related circuit routing problems (1956-1974)", Networks, 5: pp.129-149.

[Pippenger, Fischer 79]

Nicholas Pippenger, Michael J. Fischer, "Relations among complexity measures", Journal of the Association for Computing Machinery, Vol. 26, No. 2, April 1979, pp.361-381.

[Perl, Garey, Even 75]

Y. Perl, M. R. Garey, S. Even, "Efficient generation of optimal prefix code: Equiprobable words using unequal cost letters", Journal of the Association for Computing Machinery, Vol. 22, No. 2, April 1975, pp.202-214.

[Perl, Itai, Avni 78]

Yehoshua Perl, Alon Itai, Haim Avni, "Interpolation search - A LogLogN search", Communications of the ACM, July 1978, Vol. 21, No. 7, pp.550-

[Plesnik 79]

J. Plesńik, "The NP-completeness of the Hamiltonian cycle problem in palnar digraphs with degree bound two", Information Processing Letters, Vol. 8, No. 4, 30 April 1979, pp.199-

[Papadimitriou 78]

C. H. Papadimitriou, "The complexity of the capacitated tree problem", Networks, Vol. 8 (1978), pp.217-230

[Papadimitriou 79]

Christos H. Papadimitriou, "Optimality of the Fast Fourier Transform", Journal of the Association for Computing Machinery, Vol. 26, No. 1, January 1979, pp.95-102.

[Plaisted 78]

David Plaisted, "Some Polynomial and integer divisibility problems are NP-HARD", SIAM J. Comput., Vol. 7, No. 4, November 1978, pp.458-

[Posa 76]

"Hamilton circuits in random graphs", Discrete Math., 14, pp.359-364.

[Preparata 79]

F. P. Preparata, "An optimal real-time algorithm for planar convex hulls", Communications of the ACM, July 1979, Vol. 22, No. 7, pp.402-

[Preparata, Sarwate 78]

F. P. Preparata and D. V. Sarwate, "An improved parallel processor bound in fast matrix inversion", Information Processing Letters, Vol. 7, No. 3, April 1978.

[Rabin M. O. 63]

M. O. Rabin, "Probabilistic automata", Information and Control, 6 (1963), pp.230-245; also in "Sequential Machines", E. F. Moore, ed., Addison-Wesley, Reading, MA, 1964, pp.98-114.

M. O. Rabin, "Real-time computation", Israel J. Math., 1 (1963), pp.203-211.

M. O. Rabin, "Probabilistic algorithms, Algorithms and Complexity: New Directions and Recent Results", J. F. Traub, ed., Academic Press, New York, 1976, pp.21-39.

[Rabin 76]

M. O. Rabin, "Probabilistic Algorithms", Algorithms and complexity (Traub. ed.), Academic Press 1976, pp.21-39.

[Räihä, Zweben 79]

Kari-Jouko Räihä, Stuart H. Zweben, "An optimal insertion algorithm for one-sided height-balanced binary search trees", Communications of the ACM, September 1979, Vol. 22, No. 9.

[Read, Tarjan 75]

R. C. Read, R. E. Tarjan, "Bounds on backtrack algorithms for listing cycles, paths, and spanning trees", Networks, 5: pp.237-252.

[Reinwald, Soland 67]

Reinwald, L. T., Soland, R. M. "Conversion of Limited-Entry Decision Tables to Optimal Computer Programs II: Minimum storage Requirement", JACM Vol. 14, No. 4, October 1967, pp.742-755.

[Rohlf 78]

F. James Rohlf, "A probabilistic minimum spanning tree algorithm", Information Processing Letters, Vol. 7, No. 1, January 1978, pp.44-

[Rowicki 75]

Andrzej Rowicki, "A note on optimal scheduling for two-processor systems", Information Processing Letters, Vol. 4, No. 2, November 1975, pp.27-

[Rowicki 77]

Andrzej Rowicki, "A note on optimal preemptive scheduling for two-processor systems", Information Processing Letters, Vol. 6, No. 1, February 1977, pp.25-

[Rose, Tarjan 78]

Donald J. Rose, Robert Endre Tarjan, "Algorithmic aspects of vertex elimination on directed graphs", SIAM J. Appl. Math., Vol. 34, No. 1, January 1978, pp.176-

[Rosenkrantz, Stearns, Lewis 74]

D. J. Rosenkrantz, R. E. Stearns, P. M. Lewis, "Approximate algorithms for the traveling salesperson problem", Proceedings of the 15th Annual IEEE Symposium on Switching and Automata Theory (1974), pp.33-42.

[Russell, Igo 79]

R. Russell, W. Igo, "An assignment routing problem", Networks, Vol. 9 (1979), pp 1-17.

〔西条、丸岡、木村　７９〕

西条経治、丸岡章、木村正行「ファイルの最適分割」電子通信学会論文誌 '７９／８、 Vol. J６２-D №8、P５４３～

[Sahni 75]

Sartaj Sahni, "Approximate algorithms for the 0/1 knapsack prblem", Journal of the Association for Computing Machinery, Vol. 22, No. 1, January 1975, pp.115-124.

[Sahni 76]

S. Sahni, "Algorithms for scheduling independent tasks", J. ACM 23 (1976), pp.116-127.

[Sahni 77]

Sartaj Sahni, "General techniques for combinatorial approximation", Operations Research, Vol. 25, No. 6, November-December 1977, p.920-

[Sahni, Gonzalez 76]

Sartaj Sahni and Teofilo Gonzalez, "P-complete approximation problems", Journal of the Association for Computing Machinery, Vol. 23, No. 3, July 1976, pp.555-565.

[Sahni, Horowitz 78]

Sartaj Sahni, Ellis Horowitz, "Combinatorial problems: Reducibility and approximation", Operations Research, Vol. 26, No. 5, September-October 1978, p.718-

〔白井、茨木、長谷川 76〕

白井康好、茨木俊秀、長谷川利治「組合せ最適化問題の有限状態正単調モデルによる最小表現」電子通信学会論文誌 '76／2、 Vol. J59-D №2、P93～

[Rozenberg, Ehrenfreucht 79]

Grzegorz Rozenberg, Andrzej Ehrenfreucht, "Finding a homomorphism between two words is NP-complete", Information Processing Letters, Vol. 9, No. 2, 17 August 1979, pp.86-

[Schnorr 76]

C. P. Schnorr, "The network complexity and the Turing Machine complexity of finite functions*", Acta Informatica 7, pp 95-107 (1976)

[Simovici, Grigoras 79]

Dan A. Simovici, Gh. Grigoras, "Even initial feedback vertex set problem is NP-complete", Information Processing Letters, Vol. 8, No. 2, February 1979, p.64-

[Slater, Goodman, Hedetniemi 76]

P. J. Slater, S. E. Goodman, S. T. Hedetniemi, "On the optional Hamiltonian completion problem", Networks, 6: 35-51, 1976

[Stein 78]

David M. Stein, "An asymptotic, probabilistic analysis of a routing problem*, Mathematics of Operations Research, Vol. 3, No. 2, May 1978, Printed in U.S.A., p.89-

[Stone, Fuller 73]

H. S. STone and S. H. Fuller, "On the near-optimality of the shortest-latency-time-first drum scheduling discipline", Comm. ACM, 16 (1973), pp.352-353.

[Sudborough 78]

I. H. Sudborough, "On the tape complexity of deterministic context-free languages", Journal of the Association for Computing Machinery, Vol. 25, No. 3, July 1978, pp.405-414.

〔高岡　７７〕

高岡忠雄「最小総和法に対する２つの$\overline{O}$（ｎ log ｎ）アルゴリズム」情報処理　Vol.１８　№9、Sep.１９７７、P９２１～

〔高崎、玄、奥野　７７〕

高崎茂、玄光男、奥野治雄「０－１計画法による切換えが不完全なシステム信頼性の最適化」電子通信学会論文誌　'７７／７、　Vol.Ｊ６０－Ｄ　№7、Ｐ５１５～

〔武野、柿倉、向殿、増田　７８〕

武野純一、柿倉正義、向殿政男、増田英次郎「平面グラフの単純閉路による最小被覆を求めるアルゴリズム」情報処理　Vol.19　№11、Nov.1978、P1058～

〔田中、野坂、増山　７９〕

田中譲、野坂雄幸、増山顕成「Stream型Data･Base Machine №1（暫定版）― データベースマシン構成用の木構成処理モジュール」Aug.１９７９

〔田中、野坂、増山　７９〕

田中譲、野坂雄幸、増山顕成「Stream型Data Base Machine №2（暫定版）― 木構成サーチエンジンとソートエンジンを用いたデータベースマシンの構成」Aug.１９７９

〔谷、翁長、角所　７８〕
谷勝二、翁長健治、角所収「容量制限のある計算グラフのスケジューリング」電子通信学会論文誌 '７８／７、vol.J６１－D №7

[Tarjan 75]
Robert Endre Tarjan, "Efficiency of a good but not linear set union algorithm", Journal of the Association for Computing Machinery, Vol. 22, No. 2, April 1975, pp.215-225.

[Tarjan 76]
Robert Endre Tarjan, "Edge-disjoint spanning trees and depth-first search*",

[Tarjan 77]
R. E. Tarjan, "Finding optimum branchings", Networks, 7: pp.25-35.

[Tarjan, Yao 79]
Robert Endre Tarjan and Andrew Chi-Chih Yao, "Storing a sparse table", Communications of the ACM, November 1979, Vol. 22, No. 11, pp.606-

[Thomas 76]
R. E. Thomas, "On optimum path problems", Networks, 6: pp.287-305.

〔富田　７８〕

富田悦次「分岐アルゴリズムによる決定性プッシュダウンオートマトン（クラス Do：Ro ）の等価性判定」電子通信学会論文誌 '７８／１０、Vol. J６１－D №１０、P７５９～

[Thompson, Kung 77]

C.D. Thompson and H.T. Kung, Sorting on a Mesh-Connected Parallel Computer, Communications of the ACM, April 1977, Volume 20 Number 4, pp.263-271

[Traub 73]

J. F. Traub, "Complexity of sequential and parallel numerical algorithms", Academic Press, 1973

[Traub 76]

J. F. Traub (ed.), "Algorithms and complexity", Academic Press, 1976

[Ullmann 76]

J. R. Ullmann, "An algorithm for subgraph isomorphism", Journal of the Association for Computing Machinery, Vol. 23, No. 1, January 1976, pp.31-42.

[van Emde Boas 70]

P. van Emde Boas, "Preserving order in a forest in less than logarithmic time and linear space", Information Processing Letters, Vol. 6, No. 3, June 1970, p.80-

[Weide, Fredman 78]

Bruce Weide, Michael L. Fedman, "On the complexity of computing the measure of U[ai bi]", Communications of the ACM, July 1978, Vol. 21, No. 7, pp.540-

[Weiner 75]

P. Weiner, "Heuristics", Networks, 5: pp.101-103, 1975

[Winograd 75]

S. Winograd, "On the parallel evaluation of certain arithmetic expressions", Journal of the Association for Computing Machinery, Vol. 22, No. 4, October 1975, pp.477-492.

[Wong, Changra 76]

C. K. Wong and Ashok K. Chandra, "Bounds for the string editing problem",

[Wong, Coppersmith 74]

C. K. Wong and D. Coppersmith, "A combinatorial problem related to multimodule memory organizations", J. ACM 21 (1974), pp.392-402.

[Wong, Yao 75]

C. K. Wong and A. C.-C. Yao, "A combinatorial optimization problem related to data set allocation", IBM Research Report RC 5392 (1975).

〔山下、本多　７７〕

山下晶、本多波雄「可約なフローグラフの最小帰還点集合」電子通信学会論文誌 '７７／１０、 Vol.Ｊ６０－Ｄ №１０、Ｐ７９３～

〔山下、本多　７８〕

山下晶、本多波雄「フローグラフの最小被覆道集合問題の時間複雑さ」電子通信学会論文誌 '７８／１０、 Vol. Ｊ６１－Ｄ №１０、ＰＰ７２７～７３４。

[Yao 75]

Andrew Chi-chih Yao, "An $O(|E|\log\log|V|)$ algorithm for finding minimum spanning trees", Information Processing Letters, Vol. 4, No. 1, September 1975, pp.21-

[Yao 77]

A. C. Yao, "Probabilistic computations toward a unified measure of complexity", Proc. 18th IEEE Symposium on Foundations of Computer Science, pp.222-227 (1977).

[Yao 77]

A. C. Yao, "A lower bound to palindrome recognition by probabilistic turing machines", Computer Science Report, STAN-CS-77-647, Stanford University, Stanford, California (1977).

[Yao 79]

F. Frances Yao, "Graph 2-isomorphism is NP-complete", Information Processing Letters, Vol. 9, No. 2, 17 August 1979, pp.68-

[Yao, Yao 76]

Andrew Chi-Chih Yao and Foong Frances Yao, "Lower bounds on merging networks", Journal of the Association for Computing Machinery, Vol. 23, No. 3, July 1976, pp.566-571.

[Yap 76]

Chee K. Yap, "New upper bounds for selection", Communications of the ACM, September 1976, Vol. 19, No. 9, pp.501-

〔弓場、星〕

弓場敏嗣、星守「木構造を用いた見出し探索の技法」、情報処理２１巻１号、昭和５５年１月、ＰＰ．２８－４９

[Yuval 76]

G. Yuval, "An algorithm for finding all shortest paths using $N^{2.81}$ infinite-precision multiplications", Information Processing Letters, Vol. 4, No. 6, March 1976, pp.155-

[Yuval 76]

G. Yuval, "Finding nearest neighbors", information processing letters, Vol. 5, pp.63-65 (1976).

昭和55年3月発行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園3丁目5番8号
機械振興会館内
TEL（434）8211（代表）

印刷所　株式会社　昌文社
東京都港区芝5丁目26番30号
（全専売ビル）
TEL（452）4931