成長能力をもつ解探索システムの研究†
——開放システム論的アプローチ——

渡辺 俊典††

解探索技法はすでにさまざまのものが提案されており，多方面で利用されているが，問題に適した解法の選択とか，解法に付随するパラメータ値の設定に関してかなりの利用知識が要請される．この問題への対策として，これらの知識を，状況部とそのもとで実行すべき解手続き部とからなるコードで表現し，これを蓄積して利用する方式を提案した．未経験状況に対する新コードの生成を支援する母コードを導入し，生成されたコードの間に競争による淘汰作用を加えることによって，システムに成長能力，すなわちインクリメンタルな知識の獲得や改良の能力をもたせることが可能となることを示した．

1. まえがき

周知のように，解探索技法はシステム技法の中核の一つである．最近ではシミュレータと接合して，自動シミュレーションを目ざす研究も盛んになってきている[1)~5)]．最適化すべき問題の性質に応じて過去多くの解法が提案されてきたが[6)]，利用者側からみると，問題の性質に応じた解法の選択やパラメータの設定に関し，かなりの利用知識が要求される．

万能の解法があればこの難点は解消されるが，それは困難なようである．そこで伊理[7)]は，種々の問題と解法を取り揃え，両者の間の相性の良さを経験的に知識化するという接近法を示唆している．この示唆に刺激され，先に各種の解法を集積しておき，問題の性質の診断結果や解探索途中の状態等によって適当な解手続きを選択する解探索システムを提案した[8)]．ただし，解手続きとそれを呼び出す場面や状況との対応を事前に定義しておく必要があった．

しかしながら，経験の蓄積と活用を目的とするシステムは，試行錯誤の過程を通じて徐々に完成されるものでなくてはならない．これを可能化するには，外部環境からの情報流入によって自己の構造や形相を常時変化させうる系，すなわち解放系[9),10)]としてシステムを構成する必要がある．本論文では，この観点に立った解探索システムを提案する．

最近，人工知能学分野で知識工学の提案がなされた[11)]．これは，専門家の経験を蓄積した知識ベースによって高度な問題解決を目ざすものであるが，知識ベースの形成問題は今後の大きな課題となっている．本論文では，この課題への一接近法を提案する．

2. 基本構想

2.1 問題分析

解探索システムの代表例として会話型非線形計画システム(**図1**)を念頭におき，問題点を考察する．使用時には，まず未知変数 x から目標や制約値 $f(x)$ を計算する手続きを定義し(10)，プログラム化する(20)．解探索をおこなうには，問題の性質，解探索経過，解の状態などを分析し(50)，解法集合(30)より適当なものを選び，スーパバイザ(40)に指示する(60)．解探索過程で注目すべき場面や状況，それらの場合に使用する手続きやパラメータ値は，利用者の経験的知識(70)をもとに選ばれる．解探索能力の向上には，解法の開発(80)と手続き集合(30)の充実が必要であり，従来から数多くの提案がなされた．しかし，利用者側で荷なわれてきた(50)～(70)の機能を向上させることも同時に大切である．この部分を以下仮に知識管理部(KM)とよぶ．KMの機能は，モニタ機能からの情報により適切な手続きを呼び出すことであり，数学的には次の写像 Ψ にほかならない．

$$\Psi : K(s) \rightarrow A(P)$$

ここに，$K(s)$ は場面や状況 s に依存する観測情報の集合，$A(P)$ は(30)の要素の連鎖の集合であり，P は手続きのパラメータ依存性を示す．以下，解探索システムの機能向上のための Ψ の自動形成とその利用について考える．このときの課題を考察する．

(1) Ψ の漸次形成：当初から登録できる知識以外は，気付いた都度逐次追加せざるをえない．

(2) 環境変化等への適応：問題(10)，モニタ機能

† Design of a Knowledge Based Heuristic Optimizer which Evolves: An Open System Approach by TOSHINORI WATANABE (Systems Development Laboratory of Hitachi).

†† (株)日立製作所システム開発研究所

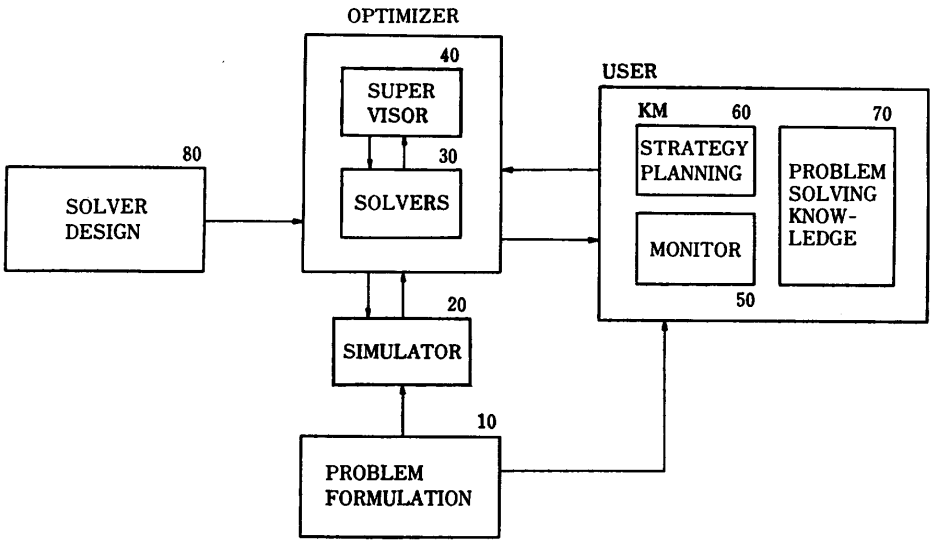


図 1 最適化システム使用時の作業環境
Fig. 1 Conventional optimization task environment.

(50)，解手続き集合(30)は知識ベース(70)の外部環境であるが，これらはしだいに変化する．Ψ はこのような動的環境の中で漸次完成される必要がある．類似状況への対応力を向上させる専門化能力と同時に，未経験状況に対処する可塑性も必要である．一般に環境からの情報には外乱が加わることが多く，統計的な意味での適応能力が要請される．

(3) 応用動作：知識ベースの容量制約や，知識獲得時間の節約のために，蓄積した経験のうちで現在の場面や状況に類似のものを活用できる必要がある．

2.2 課題の解決方法

(1) コードによる知識表現：Ψ の表現にデシジョンテーブル[12]の利用が考えられるが，前節(2)の点で問題がある．他の方法としてパーセプトロン[13]に代表される学習機械の系統が考えられる．これは，$A(P)$ の元 $\boldsymbol{A}(\boldsymbol{P})$ に対し，集合 $K(s)$ 上の判別関数 $\{f(\boldsymbol{A}(\boldsymbol{P}), \boldsymbol{K}(s), \boldsymbol{W}) | \boldsymbol{A}(\boldsymbol{P}) \in A(P), \ \boldsymbol{K}(s) \in K(s), \ \boldsymbol{W} \in R^m, \ m = \dim(\boldsymbol{K}(s)) + 1\}$ を対応させ，入力 $\boldsymbol{K}(s)$ のもとで最大値をとる判別関数に対応する $\boldsymbol{A}(\boldsymbol{P})$ を行動として選択する．$A(P)$ が小規模な有限集合であれば，変数 $\boldsymbol{W}$ の調節によって前節(2)をある程度実現できるが，$A(P)$ がパラメータ $\boldsymbol{P}$ の摂動によって一般には無限集合となるので問題が生じる．そこで，原始的ではあるが，場面 s を特徴づけるパラメータベクトル $\boldsymbol{K}(s)$ と対応する解手続き連鎖 $\boldsymbol{A}(\boldsymbol{P})$ からなる次のコードを利用する．

$$(\boldsymbol{K}(s), \boldsymbol{A}(\boldsymbol{P})) = (K_1(s), K_2(s), \cdots, K_n(s), \boldsymbol{A}_1(\boldsymbol{P}_1), \boldsymbol{A}_2(\boldsymbol{P}_2), \cdots, \boldsymbol{A}_m(\boldsymbol{P}_m))$$

このコードを必要に応じて蓄積すれば，前節(1)が実現される．また，解探索途上のさまざまの場面の状況をパラメータ化してキーを作成し，これによって上記コードの集合を検索すれば，適当な手続きを呼び出すことができる．

(2) 解手続き等のモジュール化と一括プール：前節(2)でふれた場面状況 $K(s)$ をモニタする手続きと解手続き $A(P)$ とを一括プールする．これを以下では新たに $A(P)$ と表現する．$A(P)$ の要素はできるだけモジュール化する．情報連絡の複雑化をさけるために，手続き間の交信は，共通記憶域上に各手続きが記入する情報を各手続きが参照する方式とする．手続きの変更や追加は，プール $A(P)$ の要素の変更によって実施する．

(3) コードの増殖と淘汰：任意のキーで呼び出し可能で，呼び出された場合には利用者入力によって任意の手続き連鎖をもつコードを生成できる母コードを用意する．さらに，使用される都度，自己のコピーを生成し，コピー上のパラメータを利用者入力によって変更できる能力をコードにもたせる．次に，使用したコードに利用者側から特性評点を与えることを可能化する．さらに，各コードに，生成時点を0歳とし，以降，コード群へのアクセスが発生する都度1歳加齢さ

れる年齢を与える．蓄積可能なコードの数に制限を設けておき，コード総数が制限に達した後は，上記評点と年齢によって，コード間に競争による淘汰が働くようにする．

母コードは未経験場面に対する行動の登録を可能化し，自己増殖と評点による淘汰は，専門化すなわち類似場面に対する Ψ の特性向上を可能化する．次に年齢による淘汰は，使用頻度が低く，したがって増殖機会の少ないコードを追放し，現在の環境に対して有効なコードの増加すなわち Ψ の可塑性を保障する．コード呼出し時に複数個の類似コードのうち，特性評点の良いものをいくつか残し，それらのうちからランダムにひとつ選ぶという方式を用いる．これにより，コード特性やその評点に種々の外乱が入る場合でも，良好な特性を示す確率の高いコードがしだいに支配的な地位を占めることが可能となる．以上のように，増殖・変異形成・淘汰によって前節(2)の課題に対処できる．

(4) 集合 $K(s)$ の分割：前節(3)の課題に対処するため，場面状況ベクトルの定義域 $K(s)$ をいくつかのカテゴリに分割し同一カテゴリのものは類似コードとみなす．ただし，$K(s)$ の分割自身が経験を通じて形成されるべきであり，事前に分割を定義するのは問題である．そこで，本研究では事前に暫定的分割を与えておき，上記の適応機能によって暫定分割のおのおののカテゴリを適応度の高いコードで徐々に埋めさせることにする．

3. システム設計

3.1 基本構造

図2(1)にシステムの基本構造を示す．各部の説明を下記におこなう．

(1) KB：コードを蓄積したもの．図2(2)はコードの構造を示す．ここで，コードの特性評点と年齢もキーの一部に加えた．コード番号と状況パラメータからなる部分を，以下 $\boldsymbol{K}_m(s)$ と表す．

(2) KBM：KB 内のコードの呼出し，実行およびコードの KB への収納をおこなう．

(3) KBH：読書き機能 (R & W) により，RAD エリアの内容が示す KB の番地のコードを KBBUF 1 に読み込み，SAD エリアの内容が示す KB の番地に KBBUF 2 上のコードを格納する．WKEY はワーク，SKEY はコードの KB への格納先検索用キー記憶域，RKEY は KBBUF 1 へ呼び出すコードの KB 内番地検索キー記憶域である．フォーマットは図2(2)のキー部と同じである．SKEY や RKEY 上のキーを用いて RAD や SAD に記入する KB 内の番地を求める作業は KBBUF 1 上のコードに記入された図2(1)内の SOLV の要素によって実施される．

(4) KSM：KBBUF 1 は呼び出されたコードの，KBBUF 2 は実行後のコードの記憶域．おのおののフォーマットはコードと同一である．KBCUR は実行手続き記憶域で，フォーマットは図2(2)の解手続き部と等しい．EXEC は KBBUF 1 上のコードを解釈し，SOLV 内の所定の手続きを実行させる．詳細は後述する．

(5) PSF：集合 $A(P)$ の個々の要素のプログラムの集合である SOLV(詳細後述) と，SOLV の要素が解探索に使用するアレイ SA および定数 $\boldsymbol{c}$ の記憶域 C よりなる．

(6) SA, C：フォーマットを図2(3)に示す．KSM で呼び出された手続きが解探索手続きであれば，それらは固有の方法でベクトル $\boldsymbol{x}$ を発生させ，シミュレータを駆動して $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$ を求め，$\boldsymbol{x}$ を改善してゆく．これらは図中の $\boldsymbol{x}, \boldsymbol{f}$ 部に記憶される．これ以外にも，図中の種々の要素，たとえば探索域定義行の内容を変更する手続きも SOLV 内に存在する．C は，シミュレータに与える定数の記憶域である．

3.2 KBM の動作

図3に KBM の動作を示す．KBM は KB からのコード呼び出し，実行，実行後のコードの KB への収納のサイクルを繰り返す．

(1) PRE PROCESSING：外部記憶装置上に退避してある KB の内容を取り込み，SA 上の変数名，関数名等のディフォルト値を設定する．

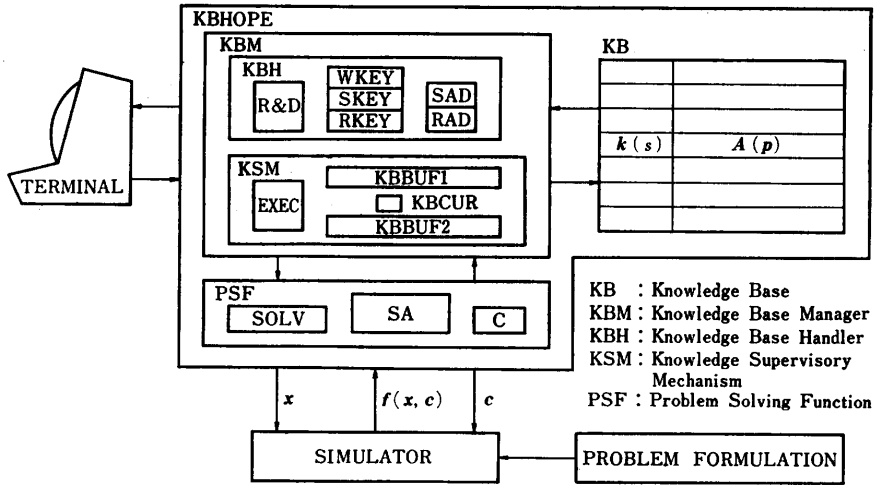
(2) SAD, RAD の初期設定：最初に実行させるコードの KB 上の番地を RAD に，KB 上のギャベッジ・コレクト用コードの番地を SAD に書き込む．

(3) コードの呼出しと収納：RAD の示す番地のコードを KBBUF 1 に呼び出し，SAD の示す番地に KBBUF 2 の内容を収納する．

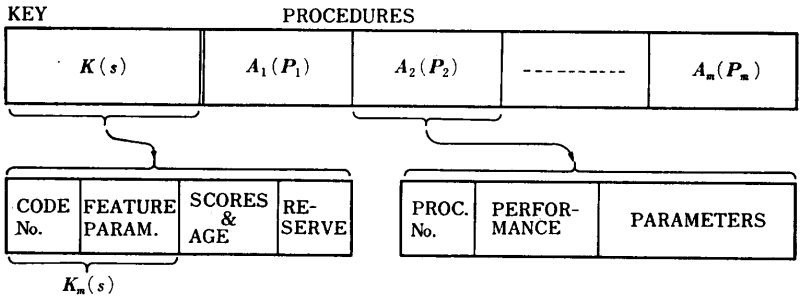
(4) 判定：KBBUF 1 上のコードの実行を完了したら(8)へ，未完なら(5)へ．

(5) $\boldsymbol{A}_i(\boldsymbol{P}_i)$ コピー：EXEC により KBBUF 1 上の手続き $\boldsymbol{A}_i(\boldsymbol{P}_i)$ を KBCUR にコピーする．

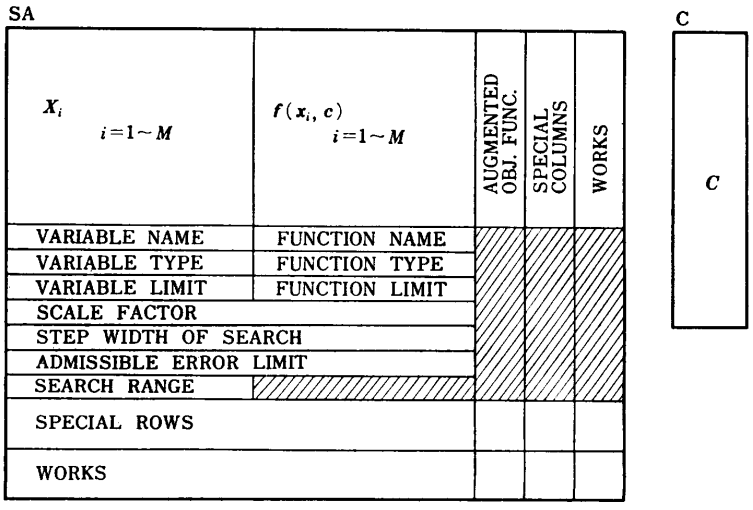
(6) 実行：EXEC により，KBCUR 上の情報を解釈し SOLV 中の該当手続きを呼び出して実行させる．呼び出された手続きは KBCUR 上のパラメータ $\boldsymbol{P}_i$ を端末に表示し，利用者入力によって KBCUR 上



(1) システム構造 System structure



(2) コードの形式 Code format



(3) アレイの形式 Array format

図 2 システム構造の説明

Fig. 2 Illustration of system structure.

の P_i を P_i' に変更し，P_i' を用いて実行に入り，実行経過や結果を端末に表示する．コード上には継続コード検索キーを RKEY に作成するもの，これを用いて KB を検索し結果を RAD に記入するもの，実行後にコードの特性評価をするもの，SKEY の内容を用いて KB を検索し結果を SAD に記入するもの等も含まれている．

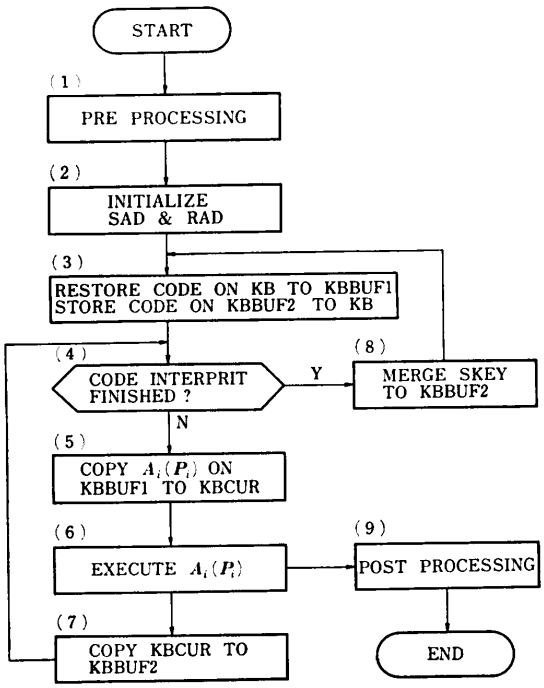


図 3 KBM の動作
Fig. 3 Behaviour of KBM.

(7) KBCUR の KBBUF 2 へのコピー：EXEC によって実施する．

(8) キーのマージ：SKEY の内容をKBBUF 2 上のコードのキー部にマージする．後述するように，当処理以前に KBBUF 1 へのコード呼出しに使用した RKEY の内容は WKEY に転送され，評点等を記入された後，SKEY に転送されている．よって(6)～(8)により，KBBUF 2 上には自己の呼出し条件パラメータと評点をキー部にもち，自己の変異コピー $A(P')$ を手続き部にもつコードが生成されている．

(9) 終了操作：システム終了に際し KB の内容を外部記憶装置に保存する．

4. 解手続き集合とコード基本形

4.1 解手続き集合

現段階での集合 $A(P)$ すなわち SOLV の要素を図 4 に示す．図中 1 は SA やCの外部記憶装置への保存等，2 は SA やCの内容変更，3 は f や Δf の計算，4 は x や f の許容誤差等の修正，5 は SA への x

```
1. PRESERVATION OF STATUS
   1. STORE CURRENT STATUS
   2. RESTORE PREVIOUS STATUS
   3. PROCEDURE LIST OUTPUT
2. MODIFY PROBLEM
   1. MODIFY VARIABLES ATTRIBUTES
   2. MODIFY FUNCTIONS ATTRIBUTES
   3. MODIFY CONSTANTS ATTRIBUTES
3. COMPUTE VALUES OF PROBLEM FUNCTIONS
   1. COMPUTE FUNCTIONS VALUES
   2. COMPUTE FUNCTIONS DERIVATIVES
   3. OUTPUT BY USERS SUBROUTINE
4. MODIFY CONTROL INFORMATIONS
   1. MODIFY CONTROL PARAMETERS
   2. MODIFY PRINT LEVEL & LIMITATIONS
   3. MODIFY VARIABLES SEARCH RANGES
5. OPERATE VECTORS
   1. OUTPUT VECTORS
   2. INPUT VECTORS
   3. DELETE VECTORS
   4. MAKE VECTORS RANDOMLY
6. OUTPUT SOLUTIONS
   1. OUTPUT BEST VECTORS
   2. OUTPUT VARIABLES EXISTING RANGES
7. MAKE PROBLEM UNCONSTRAINED
   1. MODIFY PENALTY FUNCTIONS
   2. OUTPUT BEST VECTORS OF PROBLEM
   3. CLUSTER ANALYSIS TYPE 1
   4. CLUSTER OUTPUT
   5. CLUSTER ANALYSIS TYPE 2
   6. OPTIMALITY ANALYSIS
8. PROBLEM ANALYSIS
   1. CONVEXITY DIAGNOSIS
11. LINE SEARCH
   1. QUADRATIC INTERPOLATION METHOD
   2. GOLDEN SEARCH METHOD
12. UNCONSTRAINED MINIMIZATION
   1. RANDOM SEARCH
   2. GRID SEARCH
   3. SIMPLEX METHOD
   4. DAVIES-CAMPEY-SWANN METHOD
   5. REFLECTION METHOD
13. CONSTRAINED MINIMIZATION
   1. MULTIPLIER METHOD
   2. COMPLEX METHOD
21. KNOWLEDGE BASE DATA EDITOR
   1. MAKE CODE
   2. REVISE CODE
   3. MAKE PROCEDURE
   4. DELETE CODE OR PROCEDURE
22. CONTROL KNOWLEDGE BASE
   1. STORE PREPARE
   2. RESTORE PREPARE
   3. NEXT CODE OR MAKE CODE
23. EVALUTION
   1. EVALUATE CODE
90. KNOWLEDGE BASE MANAGEMENT SYSTEM  END
```

図 4 解手続き一覧
Fig. 4 Procedure list.

の登録や削除，6，7 は解の出力やその性質の分析，8 は問題の性質の診断，11～13 は解探索，21 は KB 内コードのエディタ，22 は KB からのコードの呼出し，収納制御，23 はコードの特性評価，90 は終了処理を実施する．以下に，一部を詳述する．

(5, 3)：SA 内のベクトル $\boldsymbol{x}$ を削除する．

(7, 3)：パラメータ P_1, P_2, P_3 をもつ．SA 内の P_1 番目の点から他の点に至る P_2 分点上での拡張目的数値の増加具合を調べ，前者と同一クラスタに属する点にクラスタ番号 P_3 を与える．別クラスタの点があればその番号を，なければ 0 を出力表示する．

(7, 4)：$P_1 \neq 0$ のとき，各クラスタから P_2 組の $(\boldsymbol{x}, \boldsymbol{f})$ をクラスタ番号とともに出力する．$P_1=0$ のときは出力しない．

(8, 1)：継続コード番号と問題の性質診断結果とで継続コード検索キーを作り RKEY に記入する．継続コード番号は三つのパラメータ P_1, P_2, P_3 にデフォルトしてあるものを用いる．利用者が変更することもできる．$P_1=0$ のときは利用者入力を用いる．問題の性質は $\boldsymbol{x}$ の次元数，下式 M で定義される拡張目的関数の多峰度，$\boldsymbol{f}$ の計算に要する時間で表現する．これらは，シミュレータを駆動して求める．

$$M \equiv E_{\boldsymbol{x}_i, \boldsymbol{x}_j}\{U[L(\boldsymbol{\mu}, \boldsymbol{f}, (\boldsymbol{x}_i+\boldsymbol{x}_j)/2) - (L(\boldsymbol{\mu}, \boldsymbol{f}, \boldsymbol{x}_i)+L(\boldsymbol{\mu}, \boldsymbol{f}, \boldsymbol{x}_j))/2]\}$$

ここに，

E は期待値

$$U(t)=\begin{cases}1 & (t>0)\\ 0 & (t\leqq 0)\end{cases}$$

$L(\boldsymbol{\mu}, \boldsymbol{f}, \boldsymbol{x})$ は点 $\boldsymbol{x}$ での $\boldsymbol{f}$ の拡張目的関数値である．

(12, 3)：Nelder-Mead のシンプレクス法[14]．P_1 は初期シンプレクスサイズ決定，P_2 は収束判定，P_3～P_5 は鏡像，縮退，拡大射影係数，P_6 は，出力制御パラメータである．拡張目的関数を最小化する．

(12, 5)：W. L. Price の射影法．SA 内に $P_4 \times \dim(\boldsymbol{x})$ 個のランダムベクトルを作り，それらのうちから $1+\dim(\boldsymbol{x})$ 個をランダムに選んでシンプレクス法と類似の方法で解を改善する操作を繰り返す[15]．拡張目的関数を最小化する．多峰関数の最適解探索を目的としているが，所要計算回数は多い．P_1～P_{11} のパラメータをもつが，説明は省略する．

(21, 3)：利用者の希望する解手続きとそのパラメータを KBCUR に書き込む．KBCUR の内容は KBBUF 2 に転送されるので，KBBUF 2 上に希望するコードが生成される．パラメータはもたない．

(22, 1)：SKEY の内容を用いて KBBUF 2 上のコードの KB への格納先を求め SAD に記入する．$P_1=0$ のとき，KB 内の未使用先頭番地を，未使用分がなければ最悪コードの番地を記入する．最悪コードの判定は次のようにする．まず WKEY の内容とマッチング(後注)する KB 内コード数 N を求める．$N<P_5$ なら高齢コードのうち，第 2 評点$>P_4$ となるコードの番地を用いる．そのようなコードがなければギャベッジ・コレクトコードの番地を用いる．$N \geqq P_5$ のときは，N個のコードのうち第 2 評点が最大のものの番地を用いる．$P_1=1$ のときは P_3 の値を，$P_1=2$ のときはギャベッジ・コレクトコードの番地を，$P_1=3$ のときは利用者入力番地を使用する．$P_2=1$ のときは，KBCUR をクリアし，$P_2 \neq 1$ のときはクリアしない．以上の処理の後，WKEY 上の年齢記入部を 0 とし，収納コードを 0 歳とする．最後に，WKEY の内容を SKEY に移し，KB 内のコードをすべて 1 歳加齢する．

(注) 二つのキーは，そのコード番号と状況パラメータすなわち $\boldsymbol{K}_m(s)$ の各要素がマッチするときのみマッチすると定める．各要素は，少なくとも一方が "*" のとき，およびあらかじめ定められた同一カテゴリに含まれるときのみマッチする．カテゴリの例として，たとえば問題の性質を特徴づけるパラメータのうち，$\boldsymbol{x}$ の次元が 1～5 ならカテゴリ名は 1，6～10 は 6，11～20 は 11，21 以上は 21 と定める．多峰度については $M \leqq 0$ なら 0，$M>0$ なら 1 と定める．

(22, 2)：KBBUF 1 に呼び出すコードの KB 内番地を求めて RAD に記入する．パラメータ P_1～P_4 をもつ．P_1～P_3 の役割は (22, 1) と類似している．まず RKEY の内容を WKEY に移したあと，$P_1=0$ ならば，WKEY とマッチする KB 内のコードのうち，第 1 評点の小さいほうからたかだか P_4 個を残し，そのうちからランダムにひとつ選んだコードの番号を RAD に記入する．$P_1 \neq 0$ の場合は，(22, 1) 項参照のこと．

(22, 3)：利用者入力等をもとにして，継続コード検索用キーを RKEY に作る．パラメータ P_1～P_3 をもっている．RKEY を "*" でうめた後，(8, 1) の前半部で説明した手続きをおこなう．

(23, 1)：パラメータはもたない．コードの実行後に，シミュレータ駆動回数と計算機使用時間とを利用者に表示し，0～100 を定義域とする二つの評点を入力

CODE No.			FEATURE PARAM.				SCORES & AGE					PROC. No.																							
LEVEL1	LEVEL2	LEVEL3	DIM(X)	CONVERITY	UNIT TIME	SPARE	SCORE1	SCORE2	AGE	NO. OF SIM.	CPU TIME	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3
												P_4	P_5	P_6	P_4	P_5	P_6	P_4	P_5	P_6	P_4	P_5	P_6	P_4	P_5	P_6	P_4	P_5	P_6	P_4	P_5	P_6	P_4	P_5	P_6

(1) MOTHER CODE

												(21,3)			---			(21,3)			(22,3)			(23,1)			(22,1)			(22,2)		
*	*	*	*	*	*	*	100	0	0	0	0	A	A	A	A	A	A	A	A	A	0	A	A	A	A	A	0	1	A	0	1	A
												A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	C	C	A	C	A	A

(2) NON REPRODUCTIVE CODE

												C			---			C			(22,3)			(23,1)			(22,1)			(22,2)		
C	C	C	C	C	C	*	C	C	C	C	C	C	C	C	C	C	C	C	C	C	0	A	A	A	A	A	2	A	A	0	A	A
												C	C	C	C	C	C	C	C	C	A	A	A	A	A	A	A	A	A	C	A	A

(3) REPRODUCTIVE CODE

												C			---			C			(22,3)			(23,1)			(22,1)			(22,2)		
C	C	C	C	C	C	*	C	C	C	C	C	C	C	C	C	C	C	C	C	C	0	A	A	A	A	A	0	0	A	0	0	A
												C	C	C	C	C	C	C	C	C	A	A	A	A	A	A	C	C	A	C	A	A

(4) START CODE

												(1,3)			(22,3)			(23,1)			(22,1)			(22,2)			0			0		
B	B	B	*	*	*	*	100	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
												0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

(5) END CODE

												(90,1)			0			0			0			0			0			0		
90	1	*	*	*	*	*	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(6) PROBLEM ANALYSIS CODE

												(8,1)			(23,1)			(22,1)			(22,2)			0			0			0		
1000	1	*	*	*	*	*	90	18	0	0	0	1000	2	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
												0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0

(7) SIMPLEX OPTIMIZER CODE

												(12,3)			(22,3)			(23,1)			(22,1)			(22,2)			0			0		
1000	2	1	*	*	*	*	90	18	0	0	0	100	B	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
												0.5	2	100	0	0	0	0	0	0	20	4	0	2	0	0	0	0	0	0	0	0

(8) REFLEX OPTIMIZER CODE

												(12,5)			(22,3)			(23,1)			(22,1)			(22,2)			0			0		
1000	2	2	*	*	*	*	90	18	0	0	0	B	B	B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
												B	B	B	0	0	0	0	0	0	20	4	0	2	0	0	0	0	0	0	0	0

(9) HEURISTIC GROBAL OPTIMIZER CODE

												(5,3)			(12,5)			(12,3)			(7,3)			(12,3)			(7,3)			(7,4)		
1000	2	3	*	*	*	*	90	18	0	0	0	0	0	0	B	B	B	100	B	1	1	4	1	100	B	1	C	4	2	1	0	0
												0	0	0	B	B	B	0.5	2	100	0	0	0	0.5	2	100	0	0	0	0	0	0

												(22,3)			(23,1)			(22,1)			(22,2)			0			0		
												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
												0	0	0	0	0	0	20	4	0	2	0	0	0	0	0	0	0	0	0	0	0

NOTE
- A → ARBITRARY
- B → BEYOND DESCRIPTION
- C → CASE DEPENDENT

図 5 基本コードとそのバリエーション
Fig. 5 Code varieties.

させ，WKEY に書き込む．第1評点はコード呼出し時の，第2評点はコード追放時の判断規準である．評点は少ないほど呼び出されやすく，追放されにくい．

4.2 コードの例

ここでは，上記の手続きを用いてさまざまの特性をもつコードを定義できることを示す．後の説明の都合上，5章と関係するコード(6)～(9)も説明する．

(1) 母コード：**図5**(1)のコードは，すべてのキーとマッチするが，最も呼び出されにくく（第1評点100），最も追放されにくい(第2評点0)．あるキー $K^*(s)$ で呼び出された場合，(21,3)により利用者の希望コードを KBBUF 2 に作成する．その後，利用者から継続コード番号を入力させて RKEY に書き込み(22,3)，WKEY 上の $K^*(s)$ の評点欄に生成したコードの評点を与え(23,1)，これを KBBUF 2 上のコードにマージした後，KB への収納先の計算(22,1)と，継続コードの呼出し先の計算(22,2)をおこなう．以上の説明でわかるように，本コードは呼び出された条件をキー部にもち，その条件のもとでの解手続きを行動部にもつコードを生成する．

(2) 非増殖性コード：図中(2)のコードは，(22,1)の $P_1=2$ であるので，KBBUF 2 上の自己のコピーを KB 内のギャベッジ・コレクトコードに捨てる．すなわち増殖能をもたない．特定のコードをカタログして保存したいときに使用する．

(3) 増殖性コード：図中(3)は，(22,1)の $P_1=0$ であるので，KBBUF 2 上の自己のコピーは KB 内の劣性のコードを淘汰して KB に入り込む．よって，コード上のパラメータを徐々に改良し，コードの特性を向上させたい場合に使用する．

(4) 処理開始コード：システム起動時に図中(4)のコードを呼び出すと，手続きリスト(Fig. 4)を出力(1,3)した後，利用者に継続コードの番号を入力させる．

(5) 終了コード：図中(5)のコードは，システム終了操作(90,1)を実行する．

(6) 問題診断コード：図中(6)のコードは，問題診断ののち，コード番号(1000, 2, *)と問題診断結果のパラメータから成るキーを作成し，利用者に表示して必要な修正を受けた後，継続コードの KB 内番地を求める．非増殖性である．

(7) シンプレクス解探索：図中(7)のコードはシンプレクス法(12,3)による解探索をおこなう．増殖性コードである．

(8) 射影法による解探索：図中(8)のコードは，W. L. Price による射影法を用いた解探索をおこなう．増殖性コードである．

(9) ヒューリスティック・グローバル探索：図中(9)は，SA の x, f 部をクリア(5,3)した後，射影法によって暫定的探索をおこない，途中からシンプレクス法に切り換えて局所最適解を求め，解の点を中心としてクラスタ分析をおこなう．後2者を2回繰り返したのち，結果を出力する．増殖性である．

5. システム試作と機能実験

5.1 システム試作

前節までに述べた解探索システムを試作した．言語は FORTRAN を使用した．

5.2 機能実験の方法

2.1 節で考察した写像 Ψ のインクリメンタルな形成が可能か否かを実験する．**図6**に示すように，システムに次々に異なる問題を提示して解探索をおこなわせ，システムの動作を観測する．

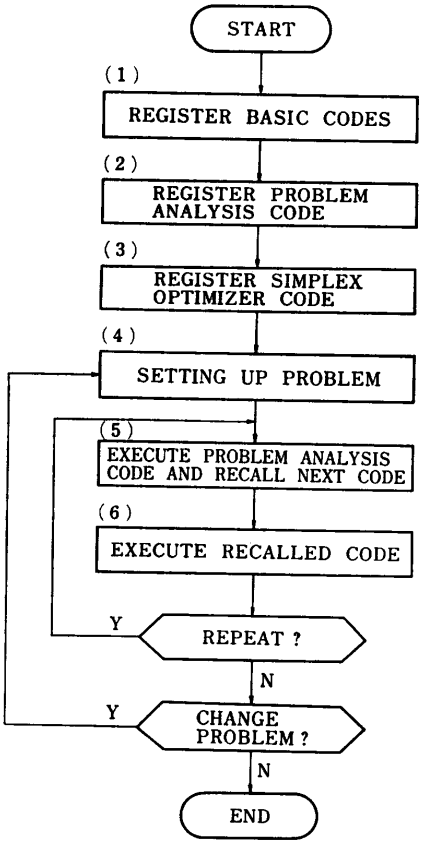


図6 実験の過程
Fig. 6 Process of experiment.

5.3 実験結果

(1) 初期コードの設定：図5(1),(4),(5)のコードをKBに登録し，起動と同時に(4)を呼び出すようにした．

(2) 問題診断コードの登録：起動と同時にコード(4)が実行され，図4のリストを表示した後，継続コード番号を質問した．そこで(1000, 1) と入力した．母コードが呼び出されたので，図5(6)のコードを登録した．継続コードを質問してきたので(1000, 2, 1)を入力した．

(3) シンプレクス解探索コードの登録：母コードが呼び出された．図5(7)のコードを登録した．

(4) 問題設定：図2(1)のシミュレータ内には，次の目的関数を設定した．これを最小化させる．

$$f_0(\boldsymbol{x})=\{A_i\|\boldsymbol{x}-\boldsymbol{x}_i\|^{B_i}+C_i\mid$$
$$i=\mathrm{Arg}\cdot\mathrm{Min}_{i\in I}\|\boldsymbol{x}-\boldsymbol{x}_i\|\}$$

ここに，$(A_i, B_i, C_i \mid i\in I,\ |I|=2)$ は定数である．$f_0(\boldsymbol{x})$ は，あらかじめ設定された点 $\boldsymbol{x}_i(i\in I)$ のうち $\boldsymbol{x}$ に最も近いものと $\boldsymbol{x}$ との距離の関数であり，$\boldsymbol{x}_i$ で局所最小値 C_i をもつ．最初の問題として単峰($|I|=1$) 4次元で，$\boldsymbol{x}_1=(0.5, 0.5, 0.5, 0.5)$ が解となるものを設定した．

(5) 問題診断コードの呼出しと実行：コード番号(1000, 1) を入力し，図5(6)のコードを呼び出した．実行結果，単峰性が検出され，次のキーが生成された(1000, 2, ＊, 4, 0, 0.316E-2, ＊)．

(6) 継続コード呼出しと実行：上記キーにマッチする図5(1),(7)のうち，(1)が呼び出されたので，(3)と同様のコードを登録した．

(7) 既登録コードの繰返し使用：図5(6)を再度実行した．その結果，図5(7)が呼び出された．シンプレクス法のパラメータを変更しつつ上記(5),(6)を繰り返した．**図7**は2回目の繰返しの際の画面表示である．経過のサマリを**図8**に示す．この経過の間に

```
**********
** START SIMPLEX METHOD **
**********
 * INPUT * STARTING VEC. NO. ?
?
2
 ------- PARAMETERS OF SIMPLEX METHOD -------
  1. COEF. OF INITIAL SYMPLEX SIZE          500.0
  2. STOPPING STANDARD ERROR OF FUNC.     0.1000000E-01
  3. REFLECTION COEF.                       1.0
  4. CONTRACTION COEF.                      0.5
  5. EXPANSION COEF.                        2.0
  6. PRINT LEVEL                            100
 -------------------------------------------
 * INPUT * PARAM NO. , VALUE TO CHANGE ?
?
1,1000

 * INPUT * PARAM NO. , VALUE TO CHANGE ?
?
0
ITER NO. =          0   OBJ. FUNC =  0.2146698E+01
ITER NO. =          0   OBJ. FUNC =  0.2146698E+01
ITER NO. =        100   OBJ. FUNC =  0.1723969E+01
ITER NO. =        200   OBJ. FUNC =  0.4921884E+00
---------- SOLUTION OF SIMPLEX METHOD ----------
     ITER NO. =       294
                  SOLUTION        FINAL SIMPLEX'S EXISTING RANGE
NO. VAR.NAME TYPE (VEC.NO.= 2)     MINIMUM          MAXIMUM          WIDTH
  1 VAR10001 N    0.5163437E+00   0.3236722E+00   0.5163437E+00   0.193E+00
  2 VAR10002 N    0.5347165E+00   0.5065165E+00   0.6244991E+00   0.118E+00
  3 VAR10003 N    0.5312824E+00   0.4325309E+00   0.6878704E+00   0.255E+00
  4 VAR10004 N    0.4230405E+00   0.3104395E+00   0.6240879E+00   0.314E+00
NO.FUNC.NAME TYPE                  MINIMUM          MAXIMUM          WIDTH
  1 FUNC0001 O    0.8373708E-02   0.8373708E-02   0.5571105E-01   0.473E-01
AUGMENTED OBJ.FUNC 0.8373708E-02   0.8373708E-02   0.5571105E-01   0.473E-01
-------------------------------------------------
**********
** END   SIMPLEX METHOD **
**********
```

図7 シンプレクス法の動作
Fig. 7 Behaviour of simplex procedure.

	REPETITION NUMBER	1	2	3	4	5	6
PARAMETERS ON RECALLED CODE	SIMPLEX SIZE COEF. P_1	100	500	500	2000	2000	2000
	STOPPING COEF. P_2	0.0001	0.01	0.01	0.01	0.002	0.002
	REFLECTION COEF. P_3	1.0	1.0	1.0	1.0	1.0	1.0
	CONTRACTION COEF. P_4	0.5	0.5	0.5	0.5	0.5	0.5
	EXPANSION COEF. P_5	2.0	2.0	2.0	2.0	2.0	2.0
	PRINT LEVEL P_6	100	100	100	100	100	100
PARAMETERS REVISED BY THE USER	SIMPLEX SIZE COEF. P_1	500	1000	2000	NC	NC	5000
	STOPPING COEF. P_2	0.01	NC	NC	0.002	NC	NC
	REFLECTION COEF. P_3	NO CHANGE	NC	NC	NC	NC	NC
	CONTRACTION COEF. P_4	NC	NC	NC	NC	NC	NC
	EXPANSION COEF. P_5	NC	NC	NC	NC	NC	NC
	PRINT LEVEL P_6	NC	NC	NC	NC	NC	NC
SOLUTION	x_1	0.472	0.516	0.549	0.472	0.529	0.537
	x_2	0.550	0.535	0.488	0.491	0.487	0.506
	x_3	0.540	0.531	0.514	0.498	0.492	0.491
	x_4	0.446	0.423	0.509	0.492	0.490	0.510
	NO. OF FUNC. EVAL.	313	294	238	209	216	215
	CODE SCORE 1	88	87	85	84	83	82
	CODE SCORE 2	88	87	85	84	83	82

図 8 シンプレクス・コードの解探索性能の向上
Fig. 8 Specialization process of simplex procedure.

KB の空エリアはなくなった．図より，解を得るまでの関数計算回数などがしだいに改善される様子がわかる．これは，コード間の淘汰に起因する．

(8) 問題の変更：8次元単峰で，$x_1=(0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)$ を解とする問題を設定した．問題診断コードを実行した結果，未経験状況なので母コードが呼び出された．そこで図5(7)と同様のコードを登録した．その後，上記(7)と同様の繰り返しをおこなった．解探索効率はしだいに向上した．

(9) 問題の変更：$x_1=(0.5, 0.5, 0.5, 0.5)$ と $x_2=(-0.5, -0.5, -0.5, -0.5)$ で値1.0をとる4次元2峰性関数を設定した．問題診断の結果，非凸性が検出され次のキーが生成された．(1000, 2, ＊, 4, 0.3, 0.273 E-2, ＊)．母コードが呼び出されたので，図5(8)のコードを登録した．問題診断コードを実行した結果，このキーとマッチする図5(1), (7), (8)のうち(7)が呼び出された．これは単峰関数用コードなので，悪い評点を与えた．(8)が呼び出されたときは良い評点を与えるようにして繰り返した結果，(8)が確実に呼び出されるようになった．

(10) 問題の変更：上記(9)と類似の8次元問題を設定した．問題診断の結果，母コードが呼び出された．そこで図5(9)のコードを登録した．繰返し使用によってコードの特性は向上した．

(11) 類似問題の提示：6次元単峰問題を設定した．問題診断の結果，上記(8)で獲得したコードが呼び出された．コードのパラメータを変更せずに実行させた結果，1度の実行で良好な解が求められた．

(12) 類似問題の提示：3次元2峰問題を設定した．上記(9)で獲得したコードが呼び出され，パラメータ変更なしで二つの解が求められた．

(13) 類似問題の提示：10次元2峰問題を設定した．上記(10)で獲得したコードが呼び出され，パラメータの修正なしで1度の実行で二つの解が求まった．総計算回数は1,431回であった．比較のために，図5(8)のコードで求解させた結果，2解を得るには約6,000回の計算が必要であった．さらに初期点を変えて図5(7)を繰り返し実行させた結果，2解を得るのに約4,000回の計算を要した．

(14) 終了操作：図5(5)のコードを呼び出すことにより実験を終了した．

5.4 結果の考察

以上によって，2.1節(1), (2), (3)の課題を一応解決できたことが示された．なお，前節(9)の過程は，コード呼出し用パラメータに，コード番号のみが存在する状況しか経験していなかったKBが，問題の特徴パラメータも観測できるという新事態，すなわち集合 $K(s)$ の変化に適応したことを示す．

5.5 システムの機能拡張に関する考察

試作システムにおいては，事前に $K(s)$ の分割をおこなったが，システムの使用を通じて事後的に分割させるほうが自然である．一方式として各コードに，そ

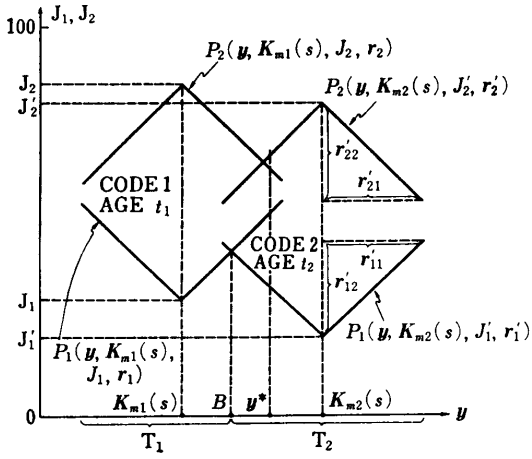


図 9　ポテンシャル関数による $K_m(s)$ の自動分割法
Fig. 9　Automatic $K_m(s)$ categorization by potential functions.

のテリトリ（縄張り）を導入すればよい．具体的には，各コードのキー部のベクトル $\boldsymbol{K}_m(s)$ と，呼出し評点 J_1，追い出し評点 J_2，年齢 t，パラメータ $\boldsymbol{r}_1, \boldsymbol{r}_2$ とを用いて各コードに次の関数組を随伴させる．

$$\{P_1(\boldsymbol{y}, \boldsymbol{K}_m(s), J_1, \boldsymbol{r}_1),\ P_2(\boldsymbol{y}, \boldsymbol{K}_m(s), J_2, \boldsymbol{r}_2),\ t \mid \boldsymbol{y} \in \boldsymbol{K}_m(s)\}$$

ここに，P_1 は $\boldsymbol{y}=\boldsymbol{K}_m(s)$ で唯一の最小値 J_1 を，P_2 は唯一の最大値 J_2 をとる．$\boldsymbol{r}_1, \boldsymbol{r}_2$ は関数の開き具合や定義範囲を規定する（**図 9**）．キー $\boldsymbol{y}^*$ でコードを呼び出す場合には $P_1(\boldsymbol{y}^*, \boldsymbol{K}_m(s), J_1, \boldsymbol{r}_1)$ 値の小さいコードを対象とし，KB から追い出す最悪コードを選ぶ場合には $P_2(\boldsymbol{y}^*, \boldsymbol{K}_m(s), J_2, \boldsymbol{r}_2)$ 値が，キー $\boldsymbol{y}^*$ をもつコードの第 2 評点よりも大きいコードを対象とする．この方式のもとでは，キー $\boldsymbol{K}_{m1}(s), \boldsymbol{K}_{m2}(s)$ をもつ二つのコードのテリトリの $K_m(s)$ 上の分岐集合は次のようになる．

$$\{\boldsymbol{y} \mid P_1(\boldsymbol{y}, \boldsymbol{K}_{m1}(s), J_1, \boldsymbol{r}_1) = P_1(\boldsymbol{y}, \boldsymbol{K}_{m2}(s), J_1', \boldsymbol{r}_1'), \boldsymbol{y} \in K_m(s)\}$$

図の例では $\boldsymbol{y}$ 軸上の点 B が分岐集合であり，二つのテリトリ T_1, T_2 を分離している．

各コードへの評点の与え方によって，各コードのテリトリや分岐集合は事後的に形成される．

6. むすび

6.1 結　論

（1）　解探索システムの機能向上のために，システム利用知識獲得機能をもたせることが重要であることを指摘し，実現上の課題を考察した．

（2）　利用知識を条件部と解手続き部からなるコードで表現し，これを蓄積して知識ベース化することを提案した．

（3）　コードに，自己の変異コピー生成能力をもたせ，生成されたコードの間に，評点や年齢による淘汰作用を印加することにより，知識ベースに適応能力や知識の漸次獲得能力をもたせることを提案した．

（4）　試作システムにより提案の有効性を示した．

6.2 今後の課題

解探索アルゴリズムやヒューリスティクスの蓄積，問題を特徴づけるパラメータや，その他の解探索途中の重要場面を特徴づけるパラメータの抽出，実用を通じての利用知識の蓄積等が今後の課題である．

謝辞　日頃，数理計画法等に関してご指導いただいている伊理正夫博士に，また研究の機会を与えていただいた日立製作所システム開発研究所，三浦武雄所長，川崎淳副所長，井原広一部長に深謝します．研究を激励してくださった佐々木浩二主任研究員，プログラム開発を支援していただいた安信千津子氏に深謝します．

参考文献

1) Smith, D. E.: Automatic Optimum-Seeking Program for Digital Simulation, *Simulation*, Vol. 27, No. 1, pp. 27-31 (Jul. 1976).
2) Polak, E.: Algorithms for a Class of Computer-aided Design Problems—A Review, *Automatica*, Vol. 15, No. 5, pp. 531-538 (Sep. 1979).
3) Langrana, N. A. and Lee, T. W.: Analysis of Dynamic Systems Using Heuristic Optimization, *J. of Dynamic Systems, M&C, ASME*, Vol. 102, No. 1, pp. 35-40 (Mar. 1980).
4) Kumar, B. and Davidson, E. S.: Computer System Design Using a Hierarchical Approach to Performance Evaluation, *Comm. ACM*, Vol. 23, No. 9, pp. 511-521 (Sep. 1980).
5) Roach, J. R. and Chow, E. P.: An Interactive Digital Simulation and Optimization: FORTRAN Program MINISIM and OPTISIM, *Simulation*, Vol. 25, No. 4, pp. 115-120 (Oct. 1975).
6) Dixon, L. C. W.: *Nonlinear Optimization*, The English University Press (1972).
7) 伊理編：組合せ理論と OR，組合せ理論部会終了報告書，日本オペレーションズ・リサーチ学会 組合せ理論部会，pp. 6-13 (1973. 3).
8) 渡辺，増田：集積回路のパラメータ自動最適化法に関する一提案，信学論，Vol. J64-D, No. 9, pp. 885-892 (1981. 9).
9) Bertalanffy, L. V.: *General System Theory*,

George Braziller, New York (1968).
10) Glansdorff, P. and Prigogine, I.: *Thermodynamic Theory of Structure, Stability and Fluctuations*, Wiley-Interscience, London (1971).
11) Winston, P. H.: *Artificial Intelligence*, Addison-Wesley (1977).
12) Hughes, M. L. et al.: *Decision Tables*, MDI Publications (1968).
13) Rosenblatt, F.: *Principles of Neurodynamics*, Spartan, New York (1962).
14) Nelder, J. A. and Mead, R.: A Simplex Method for Function Minimization, *Comput. J.*, Vol. 7, No. 4, pp. 308-313 (1965).
15) Price, W. L.: Controlled Random Search Procedure for Global Optimization, *Comput. J.*, Vol. 20, No. 4, pp. 367-370 (1977).

(昭和57年4月7日受付)
(昭和57年9月6日採録)