バージョン空間法を応用した類似事例の検索と索引の更新方式

秋藤俊介† 辻 洋†

蓄積した事例に対する類似例の検索では，索引付けが課題である．実用化システムでは，事例を追加した場合に逐次的に索引を更新することが望ましい．本論文では体系的な領域知識の獲得が困難な分野であるFORTRANプログラムチューニング問題を対象とし，問題解決時に次第に索引を精緻化するバージョン空間法を応用した検索手法を提案する．すなわち(1)互いに独立な属性により定義した事例に対して属性の論理積からなる以下の2レベルの条件を設定する．2レベルの条件とは，与えられた問題が合致したとき事例の持つ解決策をそのまま適用できると判定する第1種条件と問題解決の可能性を判定する第2種条件である．(2)検索では，各事例ごとに第1種条件と第2種条件の半順序関係が定めるバージョン空間に対して問題の属性値が占める相対位置を求める．これを類似度とし，類似度が高い順に事例を出力する．(3)索引の更新では，検索された事例が問題解決に利用できた場合は，その事例の類似度を大きくするように第1種条件を変更し，利用できなかった場合は同じ問題に対して検索されないように第2種条件を変更する．問題解決時に(3)を繰り返すことにより逐次索引を精緻化する．提案方式をビット列操作で実現したシステムを開発した．FORTRANチューニングのノウハウ検索問題へ適用・実験した結果，検索事例数を絞り込むと共に利用できる割合が向上しており，本方式の有効性を確認できた．

Application of Version-Space Method for Case Retrieval and Indexing

Shunsuke Akifuji† and Hiroshi Tsuji†

This paper proposes a method for case indexing in similarity case retrieval system, increasing applicable rate using version-space approach. Case is a set of problem part and solving plan part. For each case, we define two conditions indicating applicability. When the given problem satisfies the first condition, the plan part of the case is always applicable to the problem. The plan may be applicable to the problem, when the problem satisfies the second condition. The retriever derives a set of cases relating to a given problem in the similarity order from a case memory. The similarity is the relative position that the attribute values of the problem occupy in a version-space, a hierarchy of all possible features between the two boundary conditions. Two conditions are refined as follows: The first condition is updated to enlarge the similarity when a retrieved case is applicable to the problem. The second condition is updated not to retrieve for the same problem unless the retrieved case is applicable. The presented method is implemented using a bit operation. An experimental simulation, FORTRAN program tuning expert system, shows that the proposed method decreased the number of cases retrieved and increased the applicable rate among them.

1. はじめに

これまで多くのエキスパートシステムが開発，実用化されてきたが，その過程で知識獲得が課題となっている．この解決策として過去の問題の解，事例を検索し，検索された事例を現在の問題に適合するように修正し，解を導く事例ベース推論が注目されている[1)~3)]．事例ベース推論では適切な事例を検索できる索引付けが大きな課題である．特に実用化システムでは，事例を追加した場合にユーザの手を煩わせずに索引を更新することが望ましい．また，事例を修正する処理は，問題解決に適用できる事例から開始しないと解に至らないのが現実であるため，検索された事例の中で問題解決に適用できる事例が多いことが重要である．従って本論文では事例の索引付けに焦点を絞って議論する．

これまで研究された事例ベースシステムの検索処理では，事例を属性値の組で索引付け，合致した属性数

† (株)日立製作所 システム開発研究所関西システムラボラトリ
Kansai Systems Laboratory, Systems Development Laboratory, Hitachi, Ltd.

を基準に検索を行うものが多い．例えば，奥田[2]では検索に用いる属性に事前に優先度を与え，問題と事例で各属性が完全に一致しないときは優先度の高い属性が一致するものを検索する．このシステムではどの属性を事例の索引とするかは事前に決めておく必要があり，事例を追加した場合，新たに人手で索引付けする必要がある．

Hammond の CHEF[3] では，各事例を特徴付ける属性と特徴付けない属性を設け，それらを弁別ネットの構造でインプリメントしている．各属性には静的な優先度を与え，与えられた問題と同数の属性が一致する事例が複数存在する場合は，優先度の高い属性が一致するものを検索する．検索され一部修正を施された事例が問題解決に失敗したときは，そのときの状況，原因，属性を記憶し，次回以降の問題解決時に利用するようにしている．しかし，推論の失敗から直接索引を変更するようにはしていない．

Barletta は故障診断を対象として EBL（説明に基づく学習）によって不完全な領域知識からの索引作成を試みている[4]．事例は故障から回復までのオペレータの系列からなる．故障の症状と原因を記述した領域知識を使って事例の正当性を証明し，その結果から各種属性を事例に関連するもの，関連しないもの，関連する可能性があるものの３種に分離する．関連するものを事例を検索するための一次索引とし，可能性があるものを一次索引で検索された事例の中から一つを選択するための二次索引とする．事例の検定（validation）ではなく正当性（justification）の証明を行うだけなので領域知識が不完全でもよいと主張しているが，知識が容易に体系化できる領域でしか有効でない．

本論文では体系的な知識や領域知識の獲得が困難な分野である FORTRAN チューニングを対象とし，問題解決時に索引を更新し，次第に索引を精緻化するバージョン空間法[5),6)]を応用した検索手法を提案する．バージョン空間法を応用したのは，実用的な観点からすでに分かっている条件がある場合，それを設定できることと，検索が行われる度に逐次的な学習が可能であるからである．これに対して例えばニューラルネットワークによる学習では問題解決に適用可能な条件が分かっている場合でも，直接重み付けに反映することはできず，事例により学習させなければならない[7]．

本論文で述べる手法を簡単に説明すると以下のようになる．（1）互いに独立な属性により定義した事例に対して属性の論理積からなる以下の２レベルの条件を設定する．２レベルの条件とは，与えられた問題が合致したとき事例の持つ解決策をそのまま適用できると判定する第１種条件と問題解決の可能性を判定する第２種条件である．（2）検索では，まず問題に対して事例と同様に属性値を求める．次に各事例ごとに第１種条件と第２種条件の半順序関係が定めるバージョン空間に対して問題の属性値が占める相対位置を求め，これを類似度とする．最後に類似度が高い順に事例を出力する．（3）索引の更新では，検索された事例が推論に適用できた場合は，その事例の類似度を大きくするように第１種条件を変更し，適用できなかった場合は同じ問題に対して検索されないように第２種条件を変更する．提案方式は問題解決時に（3）を繰り返すことにより逐次索引を精緻化していく．提案方式をビット列操作で実現したシステムを開発し，FORTRAN チューニングノウハウの検索問題へ適用した．

以下，２章でバージョン空間法を応用した検索と索引の更新について述べ，第３章で提案手法をインプリメントしたシステムについて述べる．４章では開発したシステムの FORTRAN チューニングシステムの中での適用を述べ，提案手法の有効性や課題について検討する．

2. バージョン空間法を応用した検索と索引の更新

2.1 事例の表現と２レベルの索引付け

事例はあらかじめ定義した属性と属性値を用いて記述する．属性名を述語名，属性値を述語の引き数に対応させると属性は変数が１つの一階述語論理とみることができ，事例は属性を表す述語の論理演算で表現される．例えば「ドアが赤く，排気量 2000 cc の車」は色と排気量を属性，それぞれの属性値を赤と 2000 cc と表現でき，これを「色（赤）かつ排気量（2000 cc）が真」と表現する．

属性を区別するために各属性は番号付けされているとし，j 番目の属性を a_j と表す．属性値は一般化と特殊化の半順序によって上位/下位の階層構造をもたせることが可能である．ここで，互いに異なる記述 x と y があり，x が成立することが可能なすべての事例が，y が成立することが可能なすべての事例を含んでいるとき，x は y より一般化方向にあり，y は x より特殊化方向にあるという．属性値の同じ階層では

互いに排他的である．

便宜上，属性 a_j の最上位の属性値は属性の種類によらず，それぞれの属性に1個ずつ存在し v_j と表す．同じ階層で互いに排他な属性値の個数はその親の属性値に依存する．本論文では上位属性値が容易に分かるように下位の属性値は，上位の属性値の添字のすぐ右に同じ階層でのユニークな番号を付けて

$$v_{j,1_1 1_2 \cdots 1_k} \qquad (1)$$

と表す（**図1**）．ここで，

$$1_1 1_2 \cdots 1_k \qquad (2)$$

の桁数 k は最上位からの段数を表す．つまり，ある属性値が与えられたとき，最上位から数えたその属性値の段数を求める関数 K は，

$$K(v_{j,1_1 1_2 \cdots 1_k}) = \text{添字 } 1_1 1_2 \cdots 1_k \text{ の文字数} \qquad (3)$$

である．例えば，図1において $v_{j,01}$ は $v_{j,0}$ のすぐ下位，すなわち一段だけ特殊化方向にある属性値であり，添字の文字数が2であるから段数も2である．また，ある属性値のすぐ上位，すなわち一段だけ一般化方向にある属性値を求める関数 P は

$$P(v_{j,1_1 1_2 \cdots 1_k}) = v_{j,1_1 1_2 \cdots 1_{k-1}} \qquad (4)$$

である．例えば，$v_{i,10}$ のすぐ上位属性値は添字の右の1文字 '0' を除いて $v_{i,1}$ である．

属性値は一般化と特殊化の順序によって順序づけられているので下位属性値を代入した属性が真であるとき，それより上位の属性値でも真である．つまり

$$\begin{aligned} a_j(v_{j,1_1 1_2 \cdots 1_k}) = \text{true} &\Rightarrow a_j(v_{j,1_1 1_2 \cdots 1_{k-1}}) = \text{true} \\ &\Rightarrow a_j(v_{j,1_1 1_2 \cdots 1_{k-2}}) = \text{true} \\ &\cdots \\ &\Rightarrow a_j(v_j) = \text{true} \end{aligned} \qquad (5)$$

である．また，最上位の $a_j(v_j)$ は常に真である．

蓄えた i 番目の事例を C_i で表し，各属性の属性値から構成する2種類の条件で表す．2種類の条件とは，与えられた問題が合致したとき，事例の持つ解決策をそのまま適用できると判定する第1種条件と問題解決の可能性を判定する第2種条件である．本論文では事例 i の前者の条件を S_i とし，後者を G_i とする．両条件ともに全属性に値を代入した論理積の論理和で表す．これを数式で表すと式(6)から(10)のように記述できる．式(7)と(9)が示すように各論理積の少なくとも1個が真であれば S_i，または G_i は真であり，式(8)と(10)の各論理積は "＜＞" 内の属性値を代入した属性がすべて成立すれば，真である．

$$C_i \equiv (S_i, G_i) \qquad (6)$$

$$\begin{aligned} S_i &= \vee_{ms(i)} S_{i,ms(i)} \ (ms(i) = 0 \sim Ms(i)) \\ &\equiv \{S_{i,0}, S_{i,1}, \cdots S_{i,Ms(i)}\} \end{aligned}$$

ただし，$Ms(i) = S_i$ を構成する論理積の個数 -1 (7)

$$\begin{aligned} S_{i,ms(i)} &= \wedge_j a_j(v_{j,1_1 1_2 \cdots 1_{k(i,j,ms(i))}}) \ (j = 0 \sim J) \\ &\equiv \langle v_{0,1_1 1_2 \cdots 1_{k(i,0,ms(i))}}, v_{1,1_1 1_2 \cdots 1_{k(i,1,ms(i))}}, \\ &\quad \cdots v_{J,1_1 1_2 \cdots 1_{k(i,J,ms(i))}} \rangle \end{aligned}$$

ただし，J ＝属性数 -1 (8)

$$\begin{aligned} G_i &= \vee_{mg(i)} G_{i,mg(i)} \ (mg(i) = 0 \sim Mg(i)) \\ &\equiv \{G_{i,0}, G_{i,1}, \cdots G_{i,Mg(i)}\} \end{aligned}$$

ただし，$Mg(i) = G_i$ を構成する論理積の個数 -1 (9)

$$\begin{aligned} G_{i,mg(i)} &= \wedge_j a_j(v_{j,1_1 1_2 \cdots 1_{k(i,j,mg(i))}}) \ (j = 0 \sim J) \\ &\equiv \langle v_{0,1_1 1_2 \cdots 1_{k(i,0,mg(i))}}, v_{1,1_1 1_2 \cdots 1_{k(i,1,mg(i))}}, \\ &\quad \cdots v_{J,1_1,1_2 \cdots 1_{k(i,J,mg(i))}} \rangle \end{aligned}$$

ただし，J ＝属性数 -1 (10)

与えられた問題が合致したとき事例の持つ解決策をそのまま適用できる場合は，問題解決の可能性がある場合に含まれるので常に以下の式が成立する必要がある．

$$S_i = \text{true} \Rightarrow G_1 = \text{true} \qquad (11)$$

式(11)から G_i のすべての論理積 $G_{i,mg(i)}$ は S_i のすべての論理積 $S_{i,ms(i)}$ よりも一般化方向にある．

$$\begin{aligned} &a_j(V_{j,1_1 1_2 \cdots 1_{k(i,j,ms(i))}}) = \text{true} \\ &\Rightarrow a_j(V_{j,1_1 1_2 \cdots 1_{k(i,j,mg(i))}}) = \text{true} \end{aligned} \qquad (12)$$

また，後述するように1個の事例に対して G_i は複数個の論理積から構成されるが S_i は唯1個の論理積からなる．

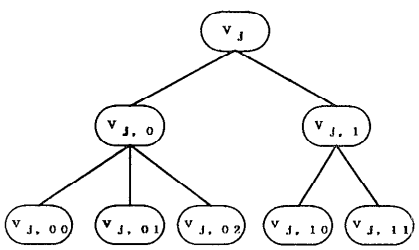


図1 属性値の階層の例
Fig. 1 An example of attribute hierarchy.

2.2 類似度算出による検索処理方式

入力された問題の属性値を求め，蓄えられた事例と入力された問題との間で類似度を計算し，指定された値以上の類似度をもつ事例を検索する．類似度は以下の手順で計算する．直観的に表現すると事例の2つの論理積 G_i と S_i の間で異なる属性値の中で問題が満足する割合が類似度である．属性値は階層構造があるので，その段数も含める．各属性値の添字を省略し，

式(8)と(10)の各属性値を次のように表す．

$$v_{j,1_1 1_2 \cdots 1_{k(i,j,ms(i))}} = s_{ij} \quad (13)$$

$$v_{j,1_1 1_2 \cdots 1_{k(i,j,mg(i))}} = g_{ij} \quad (14)$$

まず，問題において成立するすべての属性の属性値を求め，これを T とする．T は式(8)や(10)と同様な形式で記述できるので，各属性値を t_j と表す．次に S_i と G_i を構成する論理積 $S_{i,ms(i)}$ と $G_{i,mg(i)}$ のすべての組合せを選択し，

①すべての属性について，$G_{i,mg(i)}$ を構成する属性値 g_{ij} と T を構成する属性値 t_j において

$$\forall_j (g_{ij} \leftarrow t_j \text{ OR } g_{ij} = t_j) \quad (15)$$

ならば②へ進む．そうでなければ類似度＝0 とする．$x \leftarrow y$ は x が y より一般化方向にあることを表す．

②すべての属性について

$$\forall_j (s_{ij} \leftarrow t_j \text{ OR } s_{ij} = t_j) \quad (16)$$

が成立するならば，類似度＝1 とし，成立しないならば③へ進む．

③すべての属性について

$$\forall_j (t_j' \leftarrow t_j, (t_j' \leftarrow s_{ij} \text{ OR } t_j' = s_{ij})) \quad (17)$$

が成立する t_j を最小限に特殊化，つまり特殊化する属性値の段数が最小となるように変更した t_j' を求め，これを式(18)に代入して類似度を計算する．この計算によって求めた t_j' から構成される論理積 T' を最小共通論理積という．

$$\text{類似度 } (T, C_i) = \frac{\sum_{j=0}^{J-1} \text{diff}(t_j', g_{ij}) + 1}{\sum_{j=0}^{J-1} \text{diff}(s_{ij}, g_{ij}) + 1} \quad (18)$$

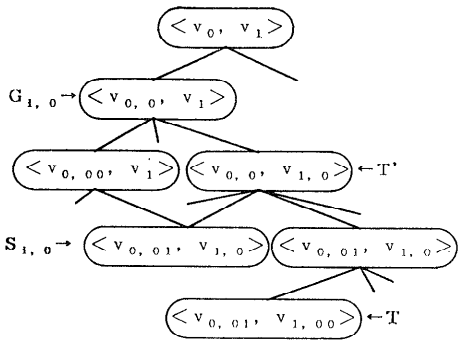
式(18)で，$\text{diff}(x, y)$ は，属性値 x と y の属性値の段差を計算する関数であり，式(3)を用いると

$$\text{diff}(x, y) = K(x) - K(y) \quad (19)$$

である．分子と分母に1を加えるのは，T が G_i と一致したとき，つまりすべての t_j と g_{ij} が等しい場合も類似する範囲であると考え，類似度が0にならないようにするためである．

S_i と G_i を構成する論理積 $S_{i,ms(i)}$ と $G_{i,mg(i)}$ のすべての組合せについて①②③を行った後，最も大きな数値をその事例の類似度とする．

例えば，**図2**のような $S_i = \{S_{i,0}\}$ と $G_i = \{G_{i,0}\}$ を持つ事例と問題 T の類似度は，T と G_i の関係が①に該当せず，②で T を最小限に一般化方向へ移動すると2個の属性値を1段ずつ一般化し T' になる．S_i と T' の差が1であり，S_i と G_i の差が2であるから式(18)より類似度は 2/3 となる．



$$\text{類似度} = \frac{\text{diff}(v_{0,0}, v_{0,0}) + \text{diff}(v_{1,0}, v_1) + 1}{\text{diff}(v_{0,00}, v_{0,0}) + \text{diff}(v_{1,0}, v_1) + 1} = \frac{0+1+1}{1+1+1} = \frac{2}{3}$$

図2 類似度計算の例
Fig. 2 An example of similarity calculation.

2.3 バージョン空間法を応用した索引付更新方式

バージョン空間とは概念を記述する述語の論理積を一般化/特殊化の関係に基づいて関係付けた論理積の集合であり，帰納推論で用いられた[5),6)]．任意の論理積 G (general) と G より特殊化方向にある任意の論理積 S (specific) を与えれば，G と S に囲まれるバージョン空間内のすべての概念を一意に決定することができる．概念を満たす正例と満たさない負例を任意の順序で与えたとき，正例をすべて含み，負例をすべて排除するように S と G を一致するまで更新し，概念を導く．

提案する索引付けの更新手法では，S と G を索引として用い，2.2 節で述べたように事例の類似度を算出する．従って，上記の原典と異なり S と G を一致させることを目的としない．ある問題 T に対して検索した事例が正しいときは，バージョン空間内で S と T の一般化方向に存在するように S を変更することによって類似度を大きくする．また，誤っていたときは，G を T の一般化方向に存在しないように特殊化することによって類似度を0にする．この処理には Mitchell の候補消去 (candidate-elimination) アルゴリズムを応用した．以下の①②を検索されたすべての事例について行う．

①問題 T に対して検索された事例が正しいとき，つまり問題解決に有効であった場合，G_i から T の一般化方向にない論理積 $G_{i,mg(i)}$ を除いた論理積の集合を新たな G_i とし，$S_{i,ms(i)}$ と T の一般方向に存在する最も特殊な論理積 $S'_{i,ms(i)}$ の集合を新たな S_i と

する．

②問題 T に対して検索された事例が誤りであったとき，つまり問題解決に有効でなかった場合，S_i から T の一般化方向にある論理積を除いた論理積 $S_{i,ms(i)}$ の集合を新たな S_i とし，T の一般化方向に存在しないように特殊化した論理積 $G'_{i,mg(i)}$ の集合を新たな G_i とする．

一般に特殊化，一般化する属性の選択の組合せにより $S_{i,ms(i)}$ と $G_{i,mg(i)}$ は複数の可能性がある．しかし，②で $G_{i,mg(i)}$ は複数になるが，①で一般化する $S_{i,ms(i)}$ は，属性が1変数なので初期値が1個の論理積だけであるとすると，これと T の一般化方向に存在する論理積は両者で共通の部分からなる論理積1個だけである．

新しく追加する事例で，判定条件が分からない場合は G_i を常に成立する条件，すなわちすべての属性値が最も一般化された値とする．予め部分的な判定条件が分かっている場合は，その条件を設定する．いずれの場合でも S_i は最も特殊な属性値に設定しておく．

$$G_i=\{G_{i,0}\} \tag{20}$$

$$G_{i,0}=\langle v_0,\cdots v_J\rangle \text{ または}$$
$$G_{i,0}=\langle v_{0,l_1\cdots l_{k(i,0,mg(i))}},\cdots v_{J,l_1\cdots l_{k(i,j,mg(i))}}\rangle \tag{21}$$

$$S_i=\{S_{i,0}\} \tag{22}$$

$$S_{i,0}=\langle v_{0,l_1\cdots l_{\max k(i,0,ms(i))}},\cdots v_{J,l_1\cdots l_{\max k(i,J,ms(i))}}\rangle \tag{23}$$

上述のアルゴリズムによって S_i と G_i は事例の適否が入力されるたびに変更され，不適の場合は，その事例と同じ属性値が成立する事例を検索しないようになり，適当の場合は，S_i が一般化方向に進む，つまり G_i の方向に進むので類似度が大きくなる．従って検索結果の適否を外部から入力することにより領域知識が十分でない場合でも索引付けが可能となる．

3. 類似事例検索システム

3.1 システム構成

システムの構成を**図3**に示す．事例検索や索引更新を行う以前に属性辞書と事例索引を作成する．属性と属性値の階層構造は以下の形式で記述し，後述する方法で数値化したものを属性辞書に蓄える．

属性名：属性値1＞属性値2：属性値1＞属性値3

ここで，不等号“＞”は，不等号より左の属性値が右の属性値より一段だけ一般化方向に存在することを意味する．

事例の検索は，2.2節で述べたように問題の記述から問題の属性値を決定し，事例ベースの各事例について式(15)から(18)を用いて類似度を計算し，類似度が大きい順に事例を出力する．索引の更新では，外部から入力された事例の適否と検索された事例の S と G，問題の属性値を用い2.3節で述べたアルゴリズムに従って索引を変更する．

3.2 ビット列操作による処理の高速化

検索処理と索引更新処理は，属性値間の順序関係の判定，式(19)の段数差計算，属性値の一般化/特殊化からなる．これらの処理量は事例数とそれぞれの事例の $G_{i,mg(i)}$ の個数の積に比例する．従って，検索や索引更新を高速に行うためには，これらの各処理を高速化する必要がある．段数計算は一般化の回数を数える処理に帰着するので，(1)2属性値間の順序関係判定，(2)属性値の一般化，(3)属性値の特殊化，の3種を高速化すればよい．開発したシステムでは階層構造を持った属性値をある数を基数とした数値，コードに変換し，3種類の処理をシフト演算を用いて高速に行う．

属性値をコード化する準備として，属性値のそれぞれの階層で互いに独立な属性値の個数の最大値 Md と最上位から最下位までの段数の最大値 Mh を求め，以下の条件を満足する基数 B と桁数 D を求める．

$$Md\leqq B \text{ かつ } B=2^n \quad n:\text{自然数} \tag{24}$$

属性値のコード化では各属性ごとに最上位の属性に

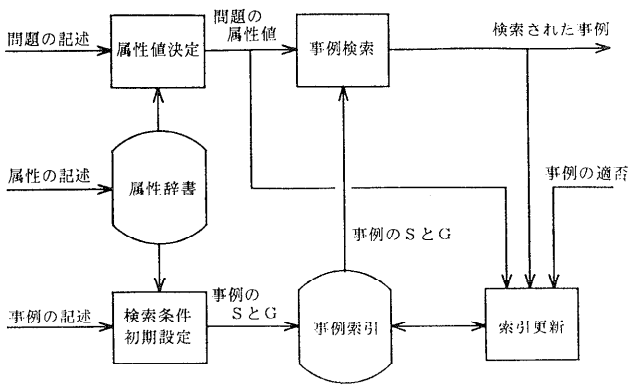


図3 システム構成
Fig. 3 System configuration.

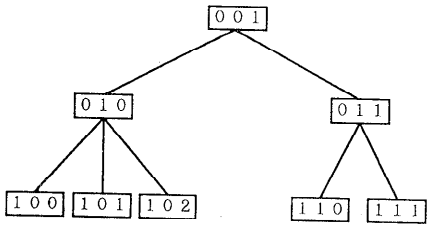


図 4 コード化した属性値の階層
Fig. 4 An example of coded attribute hierarchy.

対応するコードを1とし，属性値階層の最上位から最下位まで式(26)を用いてコードを求める．

$$Mh \leqq D \tag{25}$$

$$\mathrm{code}(q_i)=\mathrm{code}(p)\times B+i \quad i=0\sim M-1 \tag{26}$$

ここで，p は任意の属性値，q_i は p を1段だけ特殊化した属性値，M は q_i の属する階層で互いに独立である属性値の個数である．

例えば，図1では最大3段，各段でとりえる独立な属性値の種類の最大値は3であるから，基数 B を4とし，桁数 D を3桁とすれば各属性値のコードは**図4**のようになる．

このように属性値を数値化すると属性値の一般化は定数による割算，特殊化は定数を乗じ属性値の個数だけ1ずつ数値を加算する処理に帰着できる．つまり，順序関係の判定は上位のビットパタンが一致するように割算を繰り返したときの結果となる．基数を適当に設定することにより掛け算と割算はシフト演算に変更できる．以下に，各処理を具体的に記述する．

(1) 属性値 X と Y の順序関係 order (X, Y)

```
begin
  if X−Y
    X と Y は等しい;
  else
    if X の桁数>Y の桁数 begin
      X を下位方向に桁数の差だけシフト;
      if X=Y
        X は Y より特殊化方向にある;
      else
          順序関係はない;
      end;
      else begin
        Y を下位方向に桁数の差だけシフト;
        if X=Y
          X は Y より一般化方向にある;
      else
          順序関係はない;
        end;
  end;
```

(2) i 番目の属性値の一般化 general(i)

```
begin
  X←論理積の i 番目の属性値
  Y←X を下位方向に一桁分だけシフト;
  return Y;
end;
```

(3) i 番目の属性値の特殊化 specific(i)

```
begin
  X←論理積の i 番目の属性値
  Y←X を上位方向に一桁分だけシフト;
  X の下位属性の個数回以下を繰り返す
  begin
    if X≠Y
    begin
      TMP←論理積の複製;
      TMP[i]←Y;
      if TMP が既に存在しない
        TMP を新たな論理積として登録;
    end;
    Y←Y+1;
  end;
end;
```

これらの処理方法は計算量のオーダを減少できないが，計算機メモリ上での探索を行わないので高速な処理が期待できる．

4. プログラムチューニングへの適用

4.1 プログラムチューニングの特徴

対象とするのは HITAC S-810, S-820 シリーズのように行列計算をパイプライン処理により高速に処理するベクトル計算機である．現在のベクトル計算機用の FORTRAN コンパイラは，DO ループ内のデータの依存関係（代入と参照の関係）を解析し，複数のデータに対して同時に同じ演算を行うベクトル命令を自動生成する．しかし，ハードウェアやコンパイラの特性を考慮しないでコーティングしたプログラムでは適当なベクトル命令が生成されず，実行速度を上げることができない．そこで，等価性を保証しながらプログラムを部分的に書き換えて処理効率を改善するチューニングが必要になる．

チューニングの基本的な方針は，DO ループ内の演

算量の増加，ループ長の増加，メモリアクセスの連続化である．具体的には，データの依存関係の無視を指示するベクトルコンパイラ用オプションの挿入から，多次元配列の一次元化などデータ構造の変更，アルゴリズムの変更まで様々な種類の手法が存在する．これら多種類のチューニング手法から適切なものを選択するためにはプログラマのレベル，計算の目的と性質，ハードウェアの特性など多種類の条件を考慮しなければならない．

一方，実際にチューニングの熟練者から初心者への伝承では具体的なプログラムの形をとる機会が多い[8),9)]．これは，チューニングを一般的な形式として定式化することが困難であることと，複雑なものを説明する場合に具体的な形を持った特定のものを説明した方がよく理解できるからである．

チューニングは一般的な方針として記述されるが，定式化は困難であり，実際にどのように書き換えるかは様々な条件を反映した具体例で呈示されるという特徴を持つ．

4.2 チューニング事例

4.1 節に述べたプログラムチューニングの特徴を考慮し，具体的なプログラムを含むチューニング事例を検索し，アドバイスするエキスパートシステム PROTEUS (Programming Technique Adviser) を開発した．このシステムの対象利用者はチューニングの初心者であり，熟練者の持つノウハウを伝える．

チューニング事例はチューニング前後のソースプログラム，満たすべき条件，手法の目的や効果を文章や図で表したものである（**図5**）．ソースプログラムは書籍や社内の熟練者から収集し，チューニングの効果はループ長を変えて計算機で実行した実測値から求めた．各チューニング事例には 2.1 節で述べた 2 レベルの索引を付随させている．

これまでに収集した 125 個のチューニング事例にはデータ構造やアルゴリズムの変換を含むもの，ある変数の値が極めて 0 に近いなど利用者に質問しなければわからない条件を含むものもあった．従って，属性を記述する属性辞書にはプログラム中の演算の種類や変数間の依存関係の記述のほか，最大値の探索などのような数ステップで記述された部分に対する目的の記述まで含めた．

事例コード：H07
プログラム：
チューニング前：h07

```
      DO 10 I=NS, NE
        C(I)=A(I)*B(I)
        D(I)=C(I+IA)+A(I)
10  CONTINUE
```

チューニング後：

```
*VOPTION INDEP(C)
      DO 10 I=NS, NE
        C(I)=A(I)*B(I)
        D(I)=C(I+IA)+A(I)
10  CONTINUE
```

説明：
特徴：
最内側DOループ内の配列の定義が参照範囲に重なりがないことが、ユーザに分かっていてもコンパイラには不明な場合がある。このような時、最内側DOループの直前に強制ベクトル化オプション（*VOPTION INDEP（配列名））を指定するとベクトル化する。
*VOPTION VECよりもコンパイラにとってよい命令を出しやすい。

条件：
IAが正のとき、定義参照関係がない。もし、関係があるときにこのオプションを利用すると計算結果が不正になるので注意が必要である。

効果：効果の測定は、NS＝1，NE＝ループ実行回数　で行っています。

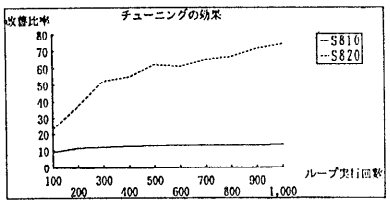


図5 PROTEUS の事例
Fig. 5 A case in PROTEUS.

4.3 実験システムの構成

PROTEUS のインプリメンテーションには日立クリエイティブワークステーション 2050/32 E 上にエキスパートシステム構築ツール ES/KERNEL/W，ワードプロセッサソフトウェア OFIS/REPORT 2，表計算ソフトウェア OFIS/POL 2 を用いた．

属性辞書にはあらかじめベクトル計算機に関連するループの構造や文の種類など 95 個の属性と各属性について属性値を登録した．チューニング事例はシステムの内部構造を知らない利用者が容易に作成できるようにワードプロセッサソフトウェアで作成し，そのまま事例ベースに登録する．チューニング事例の索引である 2 レベルの条件は表形式で表現し，表計算ソフトウェアを用いて登録する（**図6**）．ここで，縦 1 例は式(8)と(10)に対応し，各属性値を登録したものである．

利用者は，チューニングする問題プログラムの属性値を対話形式で入力する．属性値の入力が完了するとシステムは問題プログラムと各事例間の類似度を計算

図 6 表計算ソフトウェアを用いた索引登録
Fig. 6 Index entry using spread sheet.

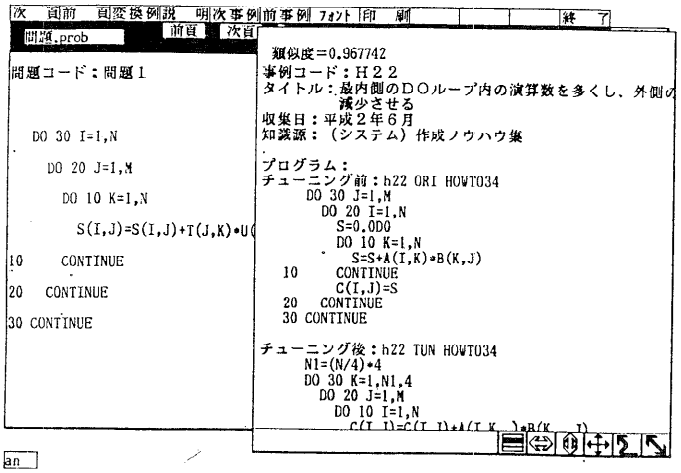


図 7 検索された事例の一例
Fig. 7 A display example (problem and relevant case).

し，ユーザから指定された値以上の類似度をもつ事例を検索する．このときの画面の例を**図 7** に示す．利用者はこれらの事例を見ながら，問題プログラムをチューニングする．

4.4 実験評価

開発した実験システムを実際のプログラムチューニング作業に適用し，評価した．検索事例の問題解決の適用性について人間による判断とシステムによる判断を分離すると**表 1** になる．評価指標として表 1 に基づき，次の 3 種を設定した．

(1) 正答率：全事例の中で本来適用できる事例が検索され，適用できない事例が検索されない割合．

$$\text{正答率}=(A+D)/(A+B+C+D) \quad (27)$$

(2) 適合率：検索された事例に占める適用できる事例の割合．

$$\text{適合率}=A/(A+B) \quad (28)$$

(3) 再現率：本来適用できる事例に占める検索される事例の割合．

$$\text{再現率}=A/(A+C) \quad (29)$$

実験に用いた問題は任意に 20 個選び，検索された事例の問題解決への適用可能/不可能は人間が判断した．属性値の入力は問題，事例ともに人手によった．属性値の個数と階層の段数は，2 個，2 段のものが 73 属性，3 個，2 段のものが 16 属性，3 段で属性値が各段で 2 から 4 個のものが 6 属性であり，属性値の平均個数は最上段の 1 個を除き，2.45 個であった．

実験は，人間が判断する最小限の属性を設定した条件に対して 10 題連続して索引の更新を行い，他の 10 題で検索することによって行った．類似度 0 以上での事例を検索し，検索された事例の適否は必ず入力する．索引更新の前後における検索事例数の変化と評価値の増減を**表 2** と**表 3** に示す．

索引更新に用いた 10 題では適否入力時に「適用できない」場合が約 90% を占めた．従って主に G の更新が行われた．初めに設定した G は 3 または 4 個であったが，索引の更新後，約 200 から 400 個に増加した．

正答率はすべての問題で増加，適合率は 6 題で増加，再現率は全体で減少した．式 (27) から (29) で B が減少し，C が増加した．$A+C$ と $B+D$ はそれぞれ一定であるから，A は減少し，D は増加したことに

表 1 検索事例の分類
Table 1 Classification of retrieved cases.

		人間の判断	
		適用可能	適用不可能
検索	類似	A	B
	類似でない	C	D

表 2　索引更新前後の検索事例数の変化
Table 2　Number of retrieved cases by the indexing method.

	索引更新前		索引更新後	
	適用可能	適用不可能	適用可能	適用不可能
問題1	2	4	0	0
問題2	4	5	3	0
問題3	1	3	1	1
問題4	8	25	4	4
問題5	3	30	0	8
問題6	2	6	0	1
問題7	1	2	0	0
問題8	11	24	5	0
問題9	10	21	1	1
問題10	9	7	7	3

表 3　索引更新の効果
Table 3　Performance improvement of the indexing method.

	増加した	変化しない	減少した
正答率	10	0	0
適合率	6	0	4
再現率	0	1	9

なる．この結果，正当率が増加した．再現率が減少した4題（問題 1, 5, 6, 7）では，索引更新後全く事例が検索されなくなった．検索対象の問題も事例の1つとして登録されているので，適否の誤りがあったことを示す．これらの適否の入力誤りの場合を除くと，提案手法は全般的に検索事例数を減少させるが，検索された事例のなかで利用可能な事例を増加させることを示す．

1個の事例の検索に要する時間 T は

$$T=1.7\times(G_{i,max}\text{の個数})\ [\text{ms}] \qquad (30)$$

であった．実験に用いたワークステーションの CPU はモトローラ社 MC 68030, 25 MHz である．ビット処理により約 2 ms で 95 個すべての属性の判定を行うことができた．現在入手可能なワークステーションは約 10 倍の処理性能があり，事例が 100 個，各事例について平均 100 個程度の条件が存在する場合は約2秒で検索できる．

4.5 考　　察

行った実験結果から，提案手法の特徴を考察する．

（1）　2レベルの条件による更新手法

提案手法は，全般的に検索事例数を減少させるが，検索された事例のなかで利用可能な事例を増加させることができる．候補事例をしぼり込むことができ，同時に適用可能な場合が増加するので検索効率の向上が期待できる．

（2）　ビット列操作による処理の高速化手法

ビット演算による処理により数秒で事例を検索する見込みを得た．検索を効率よく行うために索引の特別な構造を作成する必要がなく，索引の更新と検索が連続して行える．

（3）　検索事例のしぼり込みに要する適否入力回数

表2から分かるように索引変更前の3から4個の条件設定で検索された事例の個数は3から35個である．10回の適否入力後は，0から10個まで減少した．FORTRAN チューニングでは，ユーザが参考とする事例は 10 個程度と考えられるので，10 回程度の適否入力でほぼ必要な事例数までしぼり込まれる．

（4）　バージョン空間法と事例ベースの混合

提案手法は，元となったバージョン空間法の欠点である教示入力の誤りに弱い点を引き継いでいる[10]．従って，検索された事例の適不適が判断できないときは，適否による索引更新を行わず，疑わしい事例は残した方がよい．

（5）　条件の急激な増大

索引の更新によって判定すべき条件が急激に増大する可能性がある．これは 2.3 節で述べた G_i の特殊化が連続して行われ，S_i の一般化が行われないとき，すなわち検索された事例が問題解決に有効でない場合が連続するときに最も可能性が高い[10]．この対策として検索後すぐに索引更新をせず，可能な限り教示の順序を‘適用不可能’の後に‘適用可能’がくるように入れ替え，G_i の特殊化が連続しないようにする必要がある．

5. おわりに

本論文では，蓄積した事例に対する類似事例検索の索引として属性の論理積からなる2レベルの条件を設定し，これを検索時に外部からの教示によって逐次更新する手法を提案した．

（1）　互いに独立な属性により定義した事例に対して与えられた問題が合致したとき事例の持つ解決策をそのまま適用できると判定する第1種条件と，問題解決の可能性を判定する第2種条件を設定した．

（2）　検索では類似度を求め，高い順に事例を出力する．類似度は，問題の属性値が各事例の第1種条件と第2種条件の半順序関係が定めるバージョン空間に対

して占める相対位置である．

（3） 索引の更新では，検索された事例が問題解決に利用できた場合は，その事例の類似度を大きくするように第1種条件を変更し，利用できなかった場合は同じ問題に対して検索されないように第2種条件を変更する．

提案する検索方式と索引更新方式をビット列操作で実現し，FORTRAN プログラムのチューニング支援に適用した．実験結果は，誤った教示を与えたもの以外はすべて検索される事例数をしぼり込むと共に，検索された事例の中で利用できる割合が向上しており，提案手法の有効性を示した．今後の課題としては，判定条件の急激な増大に対する対策が挙げられる．

謝辞 本研究の機会と数多くの助言を頂いた，（株）日立製作所システム開発研究所 絹川博之博士，吉原郁夫博士，（株）日立製作所ソフトウエア開発本部 松尾洋氏，日立東北ソフトウェア（株） 磧谷利夫氏，ならびに研究の遂行に当たって協力して頂いた日立東北ソフトウェア（株） 高橋広氏，阿部郁男氏に深謝いたします．

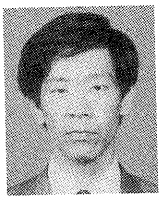
参 考 文 献

1) Kolodner, J., Simpson, R. L. and Sycava-Cyranski, K.: A Process Model of Case-based Reasoning in Problem Solving, *Proc. 9th IJCAI*, pp. 284-290 (1985).
2) 奥田健三，山崎勝弘：電力系統事故時復旧支援における事例ベースの構築法と洗練化，情報処理学会研究報告，91-AI-75, pp. 49-58 (1991).
3) Hammond, K. J.: *Case-Based Planning: Viewing Planning as a Memory Task*, Academic Press (1989).
4) Barletta, R. and Mark, W.: Explanation-Based Indexing of Cases, *Proc. DARPA CBR WS*, pp. 50-58 (1988).
5) Mitchell, T. M.: Version Space: A Candidate Elimination Approach to Rule Learning, *Proc. 5th IJCAI*, pp. 305-310 (1977).
6) Mitchell, T. M.: An Analysis of Generalization as a Search Problem, *Proc. 6th IJCAI*, pp. 577-582 (1979).
7) 安西祐一郎：認識と学習，pp. 329-371，岩波書店 (1989).
8) 島崎眞昭：スーパーコンピュータとプログラミング，共立出版 (1989).
9) Levesque, J. M. and Williamson, J. W.: *A Guidebook to Fortran on Supercomputers*, Academic Press (1988).
10) Cohen, P. R., Feigenbaum, E. A.（編），田中幸吉，淵 一博（監訳）：人工知能ハンドブック第3巻，pp. 504-522，共立出版 (1984).

（平成5年7月19日受付）
（平成6年11月17日採録）

秋藤 俊介（正会員）

1986年東京大学工学部船舶工学科卒業．1988年同大大学院修士課程修了．同年，（株）日立製作所システム開発研究所に入所．エキスパートシステムの開発を経て，ヒューマンインタフェースの研究／開発に従事．現在，関西システムラボラトリに所属．人工知能学会，日本造船学会各会員．



辻 洋（正会員）

1976年京都大学工学部数理工学科卒業．1978年同大大学院修士課程修了．同年，（株）日立製作所システム開発研究所に入所．意志決定支援システム，オフィスオートメーション，エキスパートシステムの開発を経て，ヒューマンインタフェースの研究／開発に従事．1987年9月より1988年6月まで，Carnegie-Mellon 大学客員研究員．現在，関西システムラボラトリ主任研究員．工学博士．情報処理部門技術士．人工知能学会，IEEE-CS, ACM-SIGCHI などの会員．