システム/制御/情報, Vol. 55, No. 11, pp. 480–485, 2011

解　説

遺伝的ネットワークプログラミングのアーキテクチャについて

間普　真吾*・平澤宏太郎*

1. はじめに

生物界における進化現象を規範とした進化論的計算手法の代表的なものに遺伝的アルゴリズム (Genetic Algorithm: GA)[1], 遺伝的プログラミング (Genetic Programming: GP)[2] などがあり, これらは解をおもにストリング構造や木構造によって表現している. 筆者らは2000年頃からGA, GPの拡張として有向グラフ構造を進化させる遺伝的ネットワークプログラミング (Genetic Network Programming: GNP)[3] の研究を行ってきた. また当初より, GA, GPの単なる拡張という点のみならず, 生物の脳の構造がネットワーク状になっていることから, グラフ構造による解の表現方法には多くの特長が内包されていると考え, さまざまなプログラムを柔軟に表現可能であり, さまざまな分野への適用が可能ではないかという着想をもっていた.

GNPは以下の特徴をもっている. 第一に, 判定と処理の基本単位をそれぞれ判定ノードおよび処理ノードとよばれるノードとして用意し, 複数のノードをグラフ構造状に接続することによって複雑な判定・処理のルールを生成することができる. 第二に, 有向グラフ構造をもつことで特定の処理を反復して行うことが可能となり, これがGPの分野における自動定義関数 (Automatic Defined Function: ADF)[4], つまりサブルーチン的な働きを可能にする. これにより, 少ないノード数で効率的なプログラムを生成することが可能になるため, 構造が非常にコンパクトになる. したがって, メモリの使用量と計算時間を節約することができる. また, GNPのグラフ構造はノード数を固定して進化させることができるため, ブロート（世代を経るにつれてプログラムサイズが増大する現象）を起こす恐れがない. 最後に, GNPのノード遷移は開始ノードから始まるが終端ノードは存在せず, ノード間の接続関係に基づいて判定と処理が順次実行されていく. したがって, ノード遷移自体を過去の判定と処理の履歴とみなすことができ, これが暗黙的なメモリ機能として働く. これにより動的な環境において, 現在の情報のみならず過去の情報をも考慮しながら問題を解決することができる. グラフ構造を進化させるシステムとしては, 進化論的プログラミング (Evolutionary Programming: EP)[5] による有限オートマトンの自動生成手法が提案されている. EPではすべての状態において, すべての入力の組合せに対する出力と状態遷移のルールを定義する必要があるため, 状態数と入力の数が多くなると構造が極端に大きくなってしまう. 一方, GNPは状況に応じて必要な判定ノードと処理ノードのみを用いるようにノード間の接続を進化させていくため, すべての入力に対するノード遷移を定義する必要がなく構造がコンパクトになる. このような特徴をもつGNPは動的な問題に対して特に有効である. たとえば, エレベータ群管理システムへの応用 [6] では, 高層ビル内に存在する複数のエレベータの運行ルールを自動生成し, 乗客の待ち時間などの短縮を可能にしている.

GNPはデータマイニングにも応用されている [7–10]. まず, 一般的な進化論的計算手法では, 最終世代に得られた個体が最終的な解となるが, 本手法では初期世代から最終世代までの各世代において, 全個体の実行により抽出されたルールを蓄積していく世代継続的なルール抽出を行う. したがって, 従来手法 [11] では抽出条件を満たす少数のルールを獲得することが目的であるが, 本手法では知識発見に十分な数の興味深いルールを蓄積していくことが可能である. 本データマイニング手法は都市交通流の予測 [12] やネットワーク侵入検知システム [13] などに応用されている.

2. GNPの基本アルゴリズム

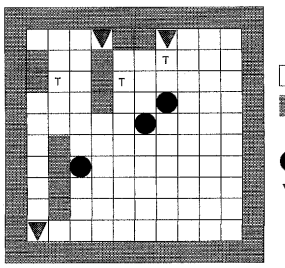
本節ではGNPの構成要素, 構造, 進化の方法などについて述べる. [14]

2.1 例題：タイルワールド

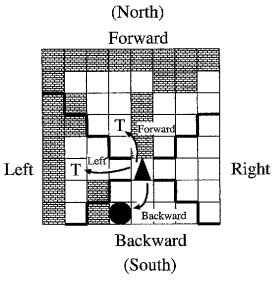
GNPの各種機能の説明を具体的にするため, マルチエージェントシステムのベンチマーク問題として知られるタイルワールドをまず紹介する. 第1図はタイルワールドの例であり, 複数のエージェント, 障害物, タイル, 穴, 床からなる2次元格子状の仮想世界である. エージェントは複数のセンサ機能と行動機能を持ち, それらを駆使してできるだけ多くのタイルを穴に落とすことを目的とする. しかし, それらはそれぞれ単独ではタスク達成のための直接的な機能として成り立たないため, 状況に応じた組合せが求められる. 第2図はエージェン

* 早稲田大学 理工学術院 情報生産システム研究科

Key Words: evolutionary computation, directed graph, data mining.



第 1 図　タイルワールド



第 2 図　エージェントが認識できる 4 方向

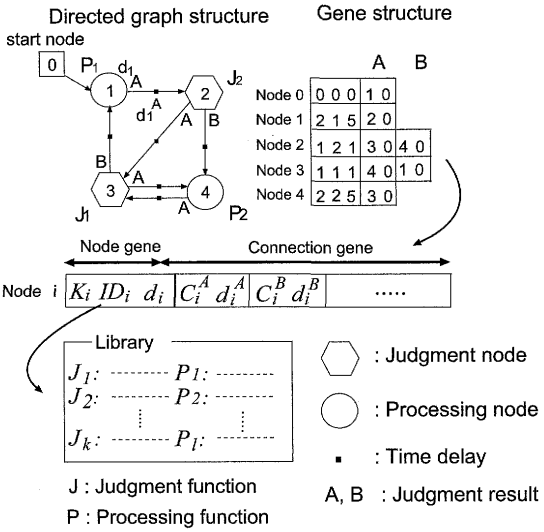
トが北側（図の上方向）を向いているときに認識できる 4 方向（forward, backward, right, left）を表している．エージェントは前進する，右を向く，左を向く，その場にとどまる，の四つの行動が可能であり，センサとして，前後左右のセルがそれぞれ何であるか，最も近いタイルの方向，穴の方向，最も近いタイルから見た最も近い穴の方向，二番目に近いタイルの方向を知ることができる．

2.2　GNP の構成要素

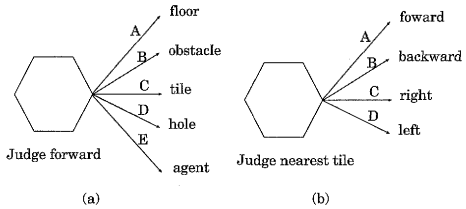
第 3 図は GNP の基本構造を表している．図の有向グラフ構造は簡単のため，1 個の開始ノード，2 個の判定ノード，および 2 個の処理ノードで構成された例を表している．開始ノードははじめに実行する処理/判定ノードに接続する機能のみをもち，判定ノードは *if-then* 型の条件分岐を行って次のノードへ遷移する．たとえば，エージェントの前方に何があるかの判定を行い，判定結果にしたがって次のノードが選択される．第 3 図の判定ノードでは，判定結果 A または B を得たあと，対応する分岐を選択し次のノードへ進む．タイルワールドに応用する場合の判定ノードの例として，第 4 図 (a) は前方に何があるかを判定し，その結果に応じて次のノード遷移先を決定するもので，(b) は一番近いタイルの方向を判定するものである．判定内容はライブラリに保存されているが，前節で説明した 8 種類のセンサに対してそれぞれ判定ノードを用意することができ，ライブラリの $J_1,\ldots,J_8$ に対応する．

処理ノードは具体的な行動を決定する．タイルワールドの場合，前進，右折，左折，とどまるの 4 種類の処理ノードが設定可能であり，それぞれ第 3 図のライブラリの $P_1,\ldots,P_4$ に対応する．ただし，処理ノードは判定ノードと異なり条件分岐を行わない．つまり，判定結果（分岐）A のみを利用する．これら判定ノードと処理ノードの接続関係を適切に進化させることによって，有効な行動ルールを生成することができる．

GNP は，判定/処理ノードの実行に要する時間，およびノード遷移に要する時間を遅れ時間 (time delay) として設定できる．実世界では，状況を判断し，次の行動に移行し，実際に行動を行う際に時間を費やすため，遅れ時間を考えた枠組みは有効である．たとえば，ある人が



第 3 図　GNP の基本構造



第 4 図　判定ノードの例

歩いていて目の前に水たまりがあったとする．このとき，水たまりを認識し（判定ノードの遅れ時間），次の行動に移行し（判定ノードから処理ノードへの遷移時間），水たまりを避ける（処理ノードの遅れ時間）ことになる．遅れ時間はそれぞれのノードの遺伝子に書き込まれており，そのノードの特徴を表す情報の一つとなっているため，GNP は遅れ時間も考慮した柔軟なプログラムを生成できる．

しかし，遅れ時間にはさらに重要な機能がある．実際の GNP の遷移では，判定ノードのループが発生し処理ノードが実行できない問題が発生することがあり，対策として以下の 1 ステップの定義が有効である．まず，タイルワールドでは，判定ノードの遅れ時間を 1，処理ノードの遅れ時間を 5，ノード遷移の遅れ時間を 0 と設定する．つぎに，エージェントの 1 回の意思決定を 1 ステップと定義する．1 ステップは，あらかじめ決められた自然数の閾値（たとえば 5）に達するか超えたら終了することにする．このとき，1 ステップで可能なノード遷移は (1) 4 回以内の連続する判定のあと処理ノードを 1 回実行，(2) 5 回の判定ノードの実行，(3) 処理ノードのみ一回実行，に限られる．したがって，判定ノードのループが発生する場合でも，5 回の判定で 1 ステップが終了し，その後もループが続くようであれば何もできない（処理ノードが実行されない）ままタスクが制限時間などにより終了する．そのような個体は適合度が低くな

るため自然淘汰されることになり，プログラムのデッドロックが起こらない．また，判定ノードの遅れ時間を1，処理ノードの遅れ時間を閾値と同じに設定すれば，1ステップの間に使用することのできる判定ノードの最大数を設定していることにもなる．

2.3 GNPの遺伝子構造

前節のGNPの構造は，以下で説明する遺伝子の集合で決定される．第3図にはノード$i(0 \leq i \leq n-1)$[1]の遺伝子構造も示されている．まずノード遺伝子(node gene)について，K_iはノードの種類を表し，$K_i=0$は開始ノード，$K_i=1$は判定ノード，$K_i=2$は処理ノードを表す．ID_iはノードiの判定/処理内容の番号であり，ライブラリの中にその内容が記載されている．d_iは判定/処理に関する遅れ時間である．つぎに接続遺伝子(connection gene)について説明する．$C_i^A, C_i^B, \ldots$は次ノードの番号を示しており，$d_i^A, d_i^B, \ldots$は次ノードに遷移するときの遅れ時間を示す．ここで，判定ノードは，参照する接続遺伝子の上側の添え字A, Bなどを判定結果にしたがって決定する．たとえば，ある判定ノードに接続ブランチAとBが存在したとする．このとき，もし判定結果が"B"であればGNPはC_i^Bとd_i^Bを参照し次のノードへ遷移する．ただし，処理ノードは判定による分岐がないので常にC_i^Aとd_i^Aを参照する．

2.4 初期集団の生成

以下の(1)–(4)は初期個体生成の手順を示しており，これを繰り返すことで，あらかじめ決められた数の個体を生成する．

(1) 開始ノード1個と，判定ノードおよび処理ノードを設計者が決めた数だけ用意する．(2) 各ノードの関数を設定する（各関数の数は設計者があらかじめ設定）．(3) 各ノードから延びる接続ブランチを，ランダムに選ばれた他のノード（開始ノード除く）へ接続する．(4) 遅れ時間を設定する（値は設計者があらかじめ設定）．

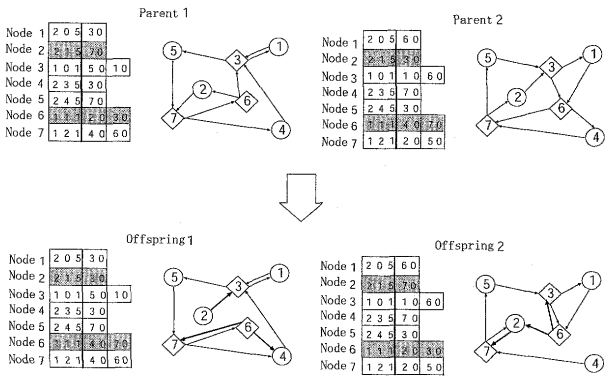
2.5 進化方法

GNPは交叉と突然変異を用いて進化を行うが，まず全個体中からよい適合度を示したエリート個体を保存し，その他の個体を，交叉と突然変異で生成した新しい個体と入れ替える．

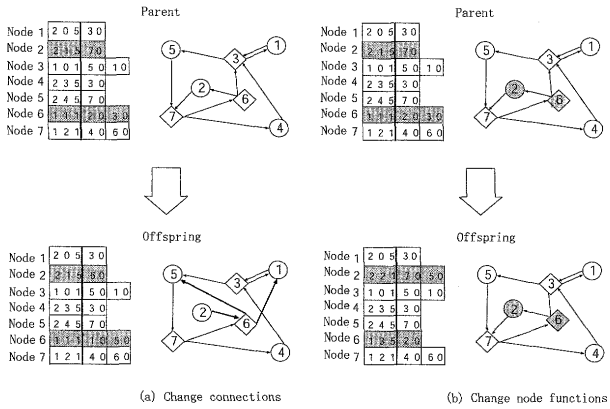
交叉は2個の親個体間で行われ，2個の子個体を生成する（第5図）．このとき，選択された交叉ノードの遺伝子がすべて入れ替わる．手順は以下の通りである．

(1) 任意の選択法（トーナメント選択など）を用いて2個の親個体を選択する．
(2) 親個体中の各ノードを確率P_cで交叉ノードとして選択する．
(3) 親個体間で，同じノード番号の交叉ノードの遺伝子をすべて交換する．

[1]ノード数がn個のとき，それぞれのノードには0から$n-1$の番号がついている．



第5図 交叉



第6図 突然変異

(4) 生成された2個の子個体が次世代の個体となる．

突然変異は1個体で行われ，新しい1個体が生成される（第6図）．手順は以下のとおりである．

(1) 任意の選択法を用いて1個体を選択する．
(2) 突然変異操作
  (a) 接続の変更：各接続ブランチについてそれぞれ確率P_mで選択し，選択された接続をランダムに選ばれた他のノードへの接続に変更する．
  (b) ノード関数の変更：各ノードについてそれぞれ確率P_mで選択し，選択されたノードの関数をランダムに選ばれた他の関数に変更する．ただし，処理ノードは他の処理ノード，判定ノードは他の判定ノードに変更される．
(3) 生成された個体が次世代の個体となる．

GNPの交叉では，各ノードが接続ブランチをもったまま交換されるため，交叉後に実行不可能解になることがない．また，突然変異後も各ノードから次のノードへの接続が必ず存在するため実行不可能になることがない．

2.6 タイルワールドのシミュレーション結果

以下のように個体の適合度を設定し，第1表のような条件で進化を行った結果を示す．

$$\text{適合度} = 100 \times \text{落としたタイルの数} + 20 \times \text{タイルを穴へ近づけたマスの数} + \text{残りステップ数} \quad (1)$$

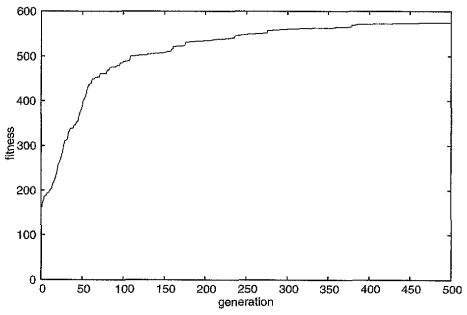
第1表 シミュレーション条件

the number of individuals	301 (crossover: 120 mutation: 180, elite: 1)
the number of nodes	61 (各関数5個+開始ノード)
Crossover rate P_c	0.1
Mutation rate P_m	0.01
generations	500
steps	60

適合度は，落としたタイルの数を最も評価し，3個のタイルを落とすことができれば300点となる．3個のタイルの初期位置は穴から合計で10マス離れたところにあるため，すべて落とすことができれば200点，落とすことができなくても近付けることができれば加点となる．残りステップ数は，すべてのタイルを落とせたときに，より早く落とすことができた個体を高く評価するためのものである．したがって，すべてのタイルを落とすことができれば500点以上を獲得できることになる．

個体数は301個であり，初期世代ではランダムに生成されるが，次世代からは前世代の最良個体1個，交叉によって生成された個体120個，突然変異によって生成された個体180個によって構成される．ノード数は，開始ノード1個と各関数（P_1,...,P_4,J_1,...,J_8）ごとに5個ずつの計61個である．これらの値は実験的に設定したものであるが，他の問題に対してもGNPの初期設定としてこれらの値からシミュレーションを開始することが多い．

第7図は最良個体の適合度を30回の独立したシミュレーションで平均化したものであり，世代を経るにつれて適合度が向上していることがわかる．また，適合度が500を超えていることからすべてのタイルを落とすことができていることがわかる．さらに詳細なシミュレーション結果，GNPの拡張アルゴリズムである学習進化型GNP，GP，EPとの比較については[3]を参照されたい．



第7図 適合度曲線

3. GNPのデータマイニングへの応用

本節では，GNPの応用例としてデータベースからの興味深い相関ルール抽出方法[14,7–10]を紹介する．

3.1 世代継続的なルール抽出

前節までに紹介したGNPでは，最終世代の最良個体が解となるが，本手法では初期世代から最終世代までの間に進化を行ったGNPの全個体を用いてルール抽出を実行する．支持度(support)，確信度(confidence)など，ルールの興味深さを示す指標の閾値をユーザがあらかじめ設定しておき，それらをみたすルールを各世代で抽出しルールプールに蓄積していく．

GAを用いたルール抽出手法[11]などでは，ルールをGAの個体（解）として扱うため，抽出条件をみたす少数のルールの獲得が目的となっている．一方，本手法では各世代のGNPの個体が抽出した興味深いルールをルールプールに蓄積する方式であるため，短時間の抽出でユーザにとって十分な数のルールを得ることが可能であり，抽出ルールを用いた予測など他の問題に利用することができる．

3.2 相関ルールおよびGNPと相関ルールの対応

相関ルール[15]は，関係データベースにおいて，「あるレコードがXを満たせばYも満たす」という事実を表現するもので，$X \Rightarrow Y$と記述される．Xを前提部，Yを結論部という．X，Yの部分に属性（アイテム）に関する条件が入る．XとYの両方を満たすレコードの割合を相関ルールの支持度，Xを含むレコードのうちYを含む割合を確信度とよび，ルール抽出の条件として，たとえば支持度が0.1以上かつ確信度が0.8以上などとする．本節では，データベースの属性をA_iとし，属性値は1または0とする．また，結論部Yをクラスラベル$k \in \{class\ 0, class\ 1, class\ 2,...\}$とすればクラス相関ルールを表すことになり，本節ではクラス相関ルールの抽出法について述べる．したがって，$X \Rightarrow class\ 1$は，ある条件Xをみたすときクラス1であるというルールを表す．

3.3 ノード遷移によるクラス相関ルール抽出

以下，GNPとクラス相関ルールの対応について説明する．各判定ノードはあらかじめ割り当てられた属性を満たすかどうかを判定し，Yes/Noに2分岐する．処理ノードはルール抽出の開始点を表し，最初に実行する判定ノードへ接続されている．本手法では，まず処理ノードP_1からルール抽出を開始し，終了したら処理ノードP_2から再度ルール抽出を開始するという手順で行うが，詳細は後述する．

第8図は，クラスが1と0のクラス相関ルールとGNPの対応例を示したものである．図中の$A_1=1$，$A_2=1$，$A_3=1$は判定ノードの内容を表している．処理ノードP_1からルール抽出を開始するので，このノード遷移

からは，以下が候補ルールとして抽出される．

$(A_1=1)\Rightarrow Class\ 1$,
$(A_1=1)\wedge(A_2=1)\Rightarrow Class\ 1$,
$(A_1=1)\wedge(A_2=1)\wedge(A_3=1)\Rightarrow Class\ 1$,
$(A_1=1)\Rightarrow Class\ 0$,
$(A_1=1)\wedge(A_2=1)\Rightarrow Class\ 0$,
$(A_1=1)\wedge(A_2=1)\wedge(A_3=1)\Rightarrow Class\ 0$

具体的な GNP の実行手順を第 8, 9 図，第 2 表を例に説明する．

まず，最初のレコード（第 2 表の Record 1）を読み込み，ノード遷移を処理ノード P_1 から開始する．処理ノードは判定ノード J_1 に接続されており，$A_1=1$ をみたすかどうかチェックする．Record 1 はこれをみたすので，Yes 側の接続先に遷移する．このときカウンタ a,$a(1)$（第 8 図）をそれぞれ 1 増やす．a は判定ノード J_1 で Yes 側の接続に遷移した回数，$a(1)$ は a のうちクラスが 1 であった回数を表す（同様に他のクラスのカウンタについても $a(0)$ などと表現する）．判定ノードの Yes 側は必ず他の判定ノードに接続されており，判定ノード J_2 を実行し，Record 1 は $A_2=1$ もみたすので，カウンタ b,$b(1)$ を 1 増やし，Yes 側の接続先に遷移する．判定ノード J_3 の内容 $A_3=1$ を Record 1 はみたさないので，No 側の接続ブランチが選択され，これは次の処理ノード P_2 へ接続される．以上の手順を最後の処理ノードまで繰り返した後，次のレコード（Record 2）を読み込み，同様の手順を処理ノード P_1 から開始する．

判定ノードの判定結果が連続して Yes の場合，長い前提部のルールが抽出されるが，処理ノード以降に実行された判定ノード数が閾値に達したら判定ノードの判定結果によらず次の処理ノードに遷移する．これにより，前提部を構成する属性の最大数を設定することができる．

カウンタは各処理ノードから開始されるノード遷移別に用意されており，異なる処理ノードから始まるノード遷移によって同じ判定ノードが実行される場合は，同じカウンタを利用せず，各ノード遷移固有のカウンタを利用する．

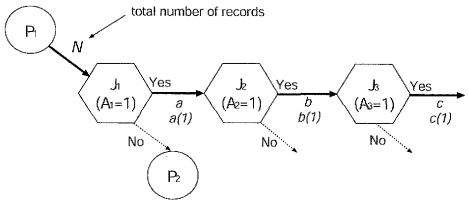
3.3.1 指標の算出とルールプールへの蓄積

ノード遷移が終了したら，カウンタ a,b,c,..., $a(1)$,$b(1)$, $c(1)$,..., $a(0)$,$b(0)$,$c(0)$,... を用いて支持度，確信度の指標を第 3 表のように計算することができる．ただし，N は総レコード数である．また，$X\Rightarrow Y$ の χ^2 値は，X の支持度 x，Y の支持度 y，$X\cup Y$ の支持度 z を用いて

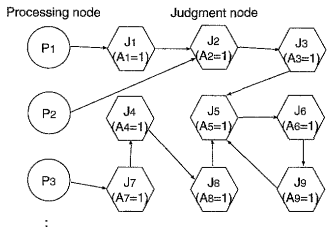
$$\chi^2=\frac{N(z-xy)^2}{xy(1-x)(1-y)} \tag{2}$$

と計算できる．

たとえば，$A_1=1$ の支持度は $x=a/N$，$class\ 1$ の支持度 y は総レコード数に対するクラス 1 レコードの割合，$(A_1=1)\Rightarrow class\ 1$ の支持度 z は $a(1)/N$ であるから，(2) 式を利用して χ^2 値を計算できる．



第 8 図 相関ルールと GNP の対応



第 9 図 相関ルール抽出のための GNP 個体

第 2 表 データベースの例

Record	A_1	A_2	A_3	class
1	1	1	0	1
2	1	0	0	1
3	0	0	1	0
4	1	1	0	1

第 3 表 相関ルールと指標

Association rule	support	confidence
$(A_1=1)\Rightarrow class\ 1$	a(1)/N	a(1)/a
$(A_1=1)\wedge(A_2=1)\Rightarrow class\ 1$	b(1)/N	b(1)/b
$(A_1=1)\wedge(A_2=1)\wedge(A_3=1)\Rightarrow class\ 1$	c(1)/N	c(1)/c
$(A_1=1)\Rightarrow class\ 0$	a(0)/N	a(0)/a
$(A_1=1)\wedge(A_2=1)\Rightarrow class\ 0$	b(0)/N	b(0)/b
$(A_1=1)\wedge(A_2=1)\wedge(A_3=1)\Rightarrow class\ 0$	c(0)/N	c(0)/c

抽出されたルールの支持度，確信度および χ^2 値があらかじめ設定した条件（たとえば，支持度 0.1 以上，確信度 0.8 以上，χ^2 値 6.64 以上[1]）をみたし，これまでに抽出されていないルールの場合，ルールをルールプールに蓄積する．

3.3.2 GNP の進化

各個体の適合度は，未抽出の興味深い相関ルールを次世代において抽出することが期待できる個体ほど大きな値をもつように設定する．

$$\text{適合度}=\sum_{r\in R}\left[\chi^2(r)+10(n_{ante}(r)-1)+\alpha_{new}(r)\right]$$

ただし，R は当該個体が抽出したルールの集合，$\chi^2(r)$ はルール r の χ^2 値，$n_{ante}(r)$ はルール r の前提部に含まれる属性数であり，$\alpha_{new}(r)$ は以下のように定義される．

[1] 6.64 は 99%有意水準，3.84 と設定すれば 95%有意水準となる．

$$\alpha_{new}(r)=\begin{cases}\alpha_{new}, & \text{ルール}r\text{がこれまでに抽出}\\ & \text{されていなかったとき}\\ 0, & \text{それ以外のとき}\end{cases}$$

α_{new}は適合度関数内の各評価項目の重要性を考慮して決定するが，たとえば150などと設定する．

各世代で適合度の高い個体を選択し，突然変異，交叉を行って次世代を構成する．突然変異や交叉によってノード接続先の変更や，判定ノードで判定する属性が変更されるが，これらは前世代の個体内に存在する興味深いルールの一部を組みかえて新しいルールの候補を作り出すことに対応する．

4. おわりに

本解説では，基本的なGNPの構造，個体の実行方法，進化方法ついて説明し，データマイニングへの応用例について述べた．これらに基づくさまざまなアルゴリズムは，エレベータ群管理システム，株式売買モデル[16]，交通流予測モデル[12]，ネットワーク侵入検知システム[13]などへ応用されており，今後は実世界の不確実性に対応できるロバストなシステムづくりに取り組んでいく予定である．

(2011年8月17日受付)

参考文献

[1] J. H. Holland: *Adaptation in Natural and Artifical Systems*, University of Michigan Press (1975)

[2] J. R. Koza: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press (1992)

[3] S. Mabu, K. Hirasawa and J. Hu: A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning; *Evolutionary Computation*, Vo. 15, No. 3, pp. 369–398 (2007)

[4] J. R. Koza: *Genetic Programming II: Automatic Discovery of Reusable Programms*, MIT Press (1994)

[5] D. B. Fogel: An introduction to simulated evolutionary optimization; *IEEE Trans. on Neural Networks*, Vol. 5, No. 1, pp. 3–14 (1994)

[6] K. Hirasawa, T. Eguchi, J. Zhou, L. Yu and S. Markon: A double-deck elevator group supervisory control system using genetic network programming; *IEEE Trans. on Systems, Man and Cybernetics, Part C*, Vol. 38, No. 4, pp. 535–550 (2008)

[7] K. Shimada, K. Hirasawa and J. Hu: Genetic network programming with acquisition mechanisms of association rules; *Journal of Advanced Computational Intelligence and Intelligent Informatic*, Vol. 10, No. 1, pp. 102–111 (2006)

[8] K. Shimada, K. Hirasawa and J. Hu: Class association rule mining with chi-squared test using genetic network programming; *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 5338–5344 (2006)

[9] 嶋田，平澤，古月：獲得した情報を用いる遺伝的ネットワークプログラミングによるデータマイニング；情報処理学会論文誌，第46巻，第10号，pp. 2576–2586 (2005)

[10] 嶋田，間普，森川，平澤，古月：遺伝的ネットワークプログラミングによる不完全データベースからのクラス相関ルール抽出；電気学会論文誌C，第128巻，第5号，pp. 795–803 (2008)

[11] A. A. Freitas: *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer (2002)

[12] H. Zhou, S. Mabu, W. Wei, K. Shimada and K. Hirasawa: Time related class association rule mining and its application to traffic prediction; 電気学会論文誌C，Vol. 130, No. 2, pp. 289–301 (2010)

[13] S. Mabu, C. Chen, N. Lu, K. Shimada and K. Hirasawa: An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming; *IEEE Trans. on Systems, Man and Cybernetics, Part C*, Vol. 41, No. 1, pp. 130–139 (2011)

[14] 電気学会 進化技術応用調査専門委員会（編）：進化技術ハンドブック，近代科学社 (2010)

[15] 福田，森本，徳山：データマイニング，共立出版 (2001)

[16] 間普，平澤，古月：強化学習と重要度指標を用いた遺伝的ネットワークプログラミングによる株式売買モデル；電気学会論文誌C，第127巻，第7号，pp. 1061–1067 (2007)

著者略歴

間普 真吾

1979年1月11日生．2003年3月早稲田大学大学院情報生産システム研究科情報生産システム工学専攻博士後期課程修了．同年4月同大学理工学総合研究センター客員講師，2004年4月理工学術院助教となり現在に至る．進化論的計算手法，強化学習，システムセキュリティなどの研究に従事．IEEE，電気学会，計測自動制御学会の会員．

平澤宏太郎

1942年1月26日生．1966年3月九州大学大学院工学研究科電気工学専攻修士課程修了．同年4月（株）日立製作所入社．日立研究所勤務，1989年同研究所副所長，1991年同大みか工場主管技師長．1992年九州大学工学部教授，1996年同大学大学院システム情報科学研究科教授，2000年同大学院システム情報科学研究院教授，2002年早稲田大学大学院情報生産システム研究科教授となり現在に至る．進化論的計算手法，データマイニング，渋滞予測システムなどの研究に従事．IEEE，電気学会，計測自動制御学会の会員．