制約充足問題解決のための確率的リペア操作を内包した 共進化型遺伝的アルゴリズム†

半田久志*・片井修**・渡部克之***
小西忠孝*・馬場充*

Coevolutionary GA with a Stochastic Genetic Repair Operator for
Solving Constraint Satisfaction Problems†

Hisashi HANDA*, Osamu KATAI**, Katsuyuki WATANABE***,
Tadataka KONISHI* and Mitsuru BABA*

In this paper, we discuss on new Coevolutionary Genetic Algorithm for Constraint Satisfaction. Our basic idea is to explore effective genetic information in the population, i.e., schemata, and to exploit the genetic information in order to guide the population to better solutions. Our Coevolutionary Genetic Algorithm (CGA) consists of two GA populations; the first GA, called "H-GA", searches for the solutions in a given environment (problem), and the second GA, called "P-GA", searches for effective genetic information involved in the H-GA, namely, good schemata. Two kinds of the fitness evaluation methods for the P-GA are introduced. We then applied our CGA to Constraint Satisfaction Problems (CSPs) incorporating a stochastic "repair" operator for the P-GA to raise the consistency of schemata with the (local) constraint conditions in CSPs. Various computer simulations on general CSPs, dynamic CSPs and cluster-structured CSPs elucidate the effectiveness of our approach.

Key Words: genetic algorithms, constraint satisfaction, co-evolution, genetic repair operation

1. はじめに

様々な問題に課せられた制約（拘束）条件に注目して制約充足という観点から問題解決を行う制約指向問題解決は人工知能分野に限らず広く適用される問題解決の枠組である^{1), 2)}。ここでは、まず、対象としている問題に内在する要素を洗い出し、それら要素間の関係性を“制約”という形を通して具体的にとらえ、制約群すべてを充足する解を求める。この部分は通常、制約ソルバという機構を用いて処理する。すなわち、制約指向問題解決では、(1) 対象問題を制約充足問題 (CSP: Constraint Satisfaction Problem) として定式化、(2) 制約ソルバによる問題解決、という順序で処理が行なわれる。制約ソルバは対象問題の構造や性質に影響を受けにくく、汎用的に用いることが可能である。このような背景のもと、効率

的な制約ソルバの研究がこれまでも広くなされている。制約ソルバとして用いられるアルゴリズムは、解構成型アルゴリズムと状態空間アルゴリズムの2つに大別される。前者は、制約に矛盾しないラベルの割り当てを、順次、より多くの変数に拡張して行くアルゴリズムであり、厳密解発見を保証するものであるが、制約充足問題のクラスが有するNPの性質をまともに受けてしまう。解構成型は、さらに、弛緩法、木探索法、併合法に分類される。一方、後者は、近似解法とも呼ばれ、探索状況をもとに状態空間の構成を鑑みつつ、実時間で十分な精度を有する解を探索するためのアルゴリズムであり、リスタート法や大域的探索法がある。リスタート法に分類される代表的なアルゴリズムとして山登り法があり、これまでに数多く研究がなされているが、近年、計算機の発達に伴い、大域的探索法のひとつである進化型計算のCSPへの適用が注目を浴びている³⁾。

本論文では、筆者らが提案してきた新しい遺伝的アルゴリズムである共進化型遺伝的アルゴリズム^{4)~6)}に制約充足問題解決のための確率的リペア演算子を導入した方法を提案し、その有用性について検討する。この遺伝的操作はCSPに内包される局所的制約構造をもとにスキーマの制約整合性を高めるものである。制約充足問題では、問題を要素間の関係（制約）として問題を定式化しているため、問題の局所的

† システム／情報部門シンポジウム 99'で発表 (1999・11)

* 岡山大学工学部 岡山市津島中 3-1-1

** 京都大学大学院情報学研究科 京都市左京区吉田本町

*** 日本IBM(株) 東京都中央区日本橋箱崎町 19-21

* Faculty of Engineering, Okayama University, Okayama

** Graduate School of Informatics, Kyoto University, Kyoto

*** Japan IBM Ltd., Tokyo

(Received September 6, 2000)

(Revised February 9, 2001)

Fig. 1 An example of CSP: graph coloring problem

評価が可能であり、その評価をもとにリペア操作として局所探索を行ない、GAの大域的探索と結びつけ探索の効率化を図ることが多い^{9)~12)}。本論文では、共進化型GAを構成するGA集団のひとつであるスキーマを探索するGA (P-GA) に対して、この遺伝演算子を導入することにより、P-GAがより整合性の高いスキーマを保持し、その結果、効果的な探索が実現できると考える。つまり、スキーマという短いオーダをもつ遺伝情報に局所探索 (リペア操作) を施すことにより、少ない計算量で制約を充足する (整合性の高い) 部分解を見出し、それらに交叉などのGA演算を施すことによってさらなる探索を行ない、解候補集団であるH-GAの探索能力改善を試みる。さらに、本論文では、制約密度と制約拘束度という指標によって分類される一般制約充足問題やその一般制約充足問題の列として定義される動的制約充足問題について実験を行なうとともに、構造化された制約充足問題についても考察を加える。

関連する研究を以下に述べる：制約充足問題へのGA適用に関する研究がこれまでになされている。制約充足問題解決のための適応度関数について、制約違反時の重みを動的に変更したり⁷⁾、変数に張られた制約関係の個数を考慮した方法が提案されている⁸⁾。また、局所探索法と組み合わせることにより、効率的な探索を指向した研究もいくつか存在している^{9)~12)}。共進化という機構をGAに適用したものでは、Hillsによる先駆的な研究がある¹³⁾。ParedisはこのHillsの競合関係にある共進化のメカニズムを制約充足問題に適用した¹⁴⁾。すなわち、彼の手法では、解候補集団と制約集団との2集団からなっており、解候補集団は制約集団の中で充足が難しい制約を充足すればするほど高い適応度が与えられる。一方、制約集団は解候補集団が充足することが難しい制約ほど高い適応度が与えられ、この結果、現在の遺伝情報では充足が困難であるような制約を充足するように解候補集団は進化を行なうようになる。近年、競合関係とは異なった、片利共生に分類される進化型計算として、ウイルス進化論に基づくGAが提案されている^{15), 16)}。これらの手法では、解候補集団とスキーマ集団が与えられた問題を解決するために導入されている。ここでの、スキーマ集団はHillsらの共進化アルゴリズムとは異なり解候補集団の改善度に基づいて適応度

計算がなされる。本論文で用いる共進化型アルゴリズムは、解候補集団とスキーマ集団が片利共生の関係を保ちつつ探索を行なうという点でこのウイルス進化に基づくGAと類似しているが、直接、GAを用いることによってよりよいスキーマ集団を探索しているところが異なる。つまり、本論文では、スキーマ集団であるP-GAは制約充足問題に特化したリペア演算を用いて制約違反の少ないスキーマを生成し、さらに、交叉によってより長い定義長となるスキーマを探索する。さらに、ウイルス進化論に基づくGAと同様に、その探索結果を解候補集団であるH-GAに伝播し、遺伝的アルゴリズムの探索能力改善を図る。

2. 制約充足問題の定式化

制約充足問題 (CSP) は変数と変数間に張られた制約からなる問題のクラスである。とくに、CSPで2変数間のみについて制約が張られている問題のクラスはバイナリーCSPと呼ばれ、これまで盛んに研究がなされており、本論文でもこのクラスについて考察を加える。また、本論文では、変数の取り得るラベル (値) が離散値である問題を取り扱う。以下では、バイナリーCSPのひとつであり、ベンチマーク問題としても良く用いられているグラフ彩色問題を例にして、CSPを説明する。Fig. 1に描かれているように、CSPは (U, L, T, R) の4つ組を用いて定義される。ここで、 U, L, T, R はそれぞれ、変数の集合、ラベルの集合、変数間の制約関係、制約関係に内在するラベル間の制約構造を表す。この3色グラフ彩色問題では、変数の集合 U は、与えられた問題をグラフ表現したとき、ノードで表現される。ラベルの集合 L は彩色を行なう3色の色を表す3つのラベル (r, g, b) で定義され、変数間の制約関係 T はノード間に隣接関係が存在するとき規定され、問題をグラフ表現したときエッジで示される。制約関係に内在するラベル間の制約構造 R は、隣接するノード間で異なった色の彩色を示すラベルの組み合わせで表現される。CSPはすべての制約関係を充足するようなラベルの組み合わせを見出すことによって解かれる。

制約充足問題を解く際、問題の難易度を表現する指標として、制約拘束度と制約密度という二つの指標を導入する¹⁾。エッジ α に関する制約拘束度 T_α は次式で表現される：

$$T_{\alpha} = \frac{\text{エッジ}\alpha\text{に存在する制約の個数}}{\text{エッジ}\alpha\text{に規定されるラベル組の総数}} \times 100.$$

問題 P_r の制約拘束度 T_{Pr} は問題に内在するすべてのエッジについてこのエッジに関する制約拘束度を求め、それらの平均を取ることににより計算する。一方、問題の制約密度 D_{Pr} は

$$D_{Pr} = \frac{\text{問題に内在する制約関係の個数}}{\text{問題における2変数の組合せ総数}} \times 100.$$

で表現される。たとえば、Fig. 1 では、エッジ XY に関する制約拘束度は次のように計算される：エッジ XY に規定された制約は $\{(r, g), (r, b), (g, r), (g, b), (b, r), (b, g)\}$ の6個である。そして、エッジ XY で取り得るラベル組の総数は 3×3 の9個であるから、エッジ XY に関する制約拘束度は $6/9 \times 100$ となる。このグラフ彩色問題では、すべてのエッジについて同じ制約構造を内包しているので、3色グラフ彩色問題の制約拘束度は $66.6\cdots$ となる。また、この例題の制約密度は、問題に内在する制約関係の個数が5、問題に内在する変数の2変数の組合せ総数が ${}_4C_2 = 6$ であるから、 $5/6 \times 100 = 83.3\cdots$ となる。

3. 制約充足問題のための共進化型 GA

本論文では、制約充足問題解決のために、これまで我々が提案を行ってきた共進化型 GA^{4)~6)} を適用する。Fig. 2 に描かれているように、共進化型 GA は H-GA (Host GA) と P-GA (Parasite GA) という2つの GA 集団からなる。H-GA は従来型 GA であり、与えられて問題に対して進化を行ない、より良い解の探索を行なう。一方、P-GA は H-GA 集団の遺伝情報に内在する有用スキーマの探索を行なう。この P-GA では、個体 (P-個体) は H-GA の対立遺伝子に “*” を加えることで表現される (H-GA におけるスキーマ表現となる)。この図のように、共進化型 GA では、2つの GA による探索情報をそれぞれに伝播させるために superposition と

transcription という遺伝演算子が導入されている。以下の小節では、P-個体の適応度評価、superposition、transcription 遺伝演算子、さらに、本論文で導入した確率的リペア演算子の詳細について述べる。

3.1 Superposition 演算子

P-GA 集団中の1つの個体 (P-個体) の情報を H-GA の中の1つの個体 (H-個体) に反映させる操作を、Fig. 3 にあるような superposition 演算子として導入する。この図では12変数でそれぞれの遺伝子座に5種類の遺伝子が入りうるような問題を例としてあげており、つまり、H-個体の遺伝情報が “32104...” で、P-個体が “** *14...” で表されている。これらの個体から superposition 演算子を用いて生成された個体は1番目の遺伝子座から3番目の遺伝子座は “321” と H-個体と同一となり、4、5番目の遺伝子座は “14” と P-個体と同じである。すなわち、この遺伝演算子は “*” を除いた P-個体の遺伝情報を、H-個体に上書きし新たな遺伝情報を生成する。P-個体は H-GA でのスキーマを表現しており、この操作はスキーマ情報を個体 (H-個体) に取り込むものであると考えられる。Fig. 4 に示すように、スキーマは解空間の部分空間を表現している。この部分空間はドントケアシンボルとなっている変数 (遺伝子座) に何かしらの値 (遺伝子) を代入することによってできる候補補から構成されている。また、superposition 演算子は P-個体の中でドントケアシンボルとなっている変数に、H-個体の値を埋め込むものであるから、この演算子はある H-個体を P-個体が表現する部分空間に射影することにより新たな解 (H-個体) を生成する演算を行なうと解釈することができる。この演算によってできた新たな遺伝情報 (H-個体のもつ情報) は P-個体の適応度評価に用いられ、この詳細を次小節にて述べる。

3.2 P-GA の適応度評価

P-GA は H-GA に内在する有用なスキーマを探索している。ここでは、H-GA にとって有用であるようなスキーマには次のようなものがあると考えられる：(1) H-GA によっては未だ発見されていなかった新しい有効スキーマ、もしくは、未発見か否かにかかわらず、(2) 平均適応度の高い部分空間を表現しているスキーマ。もし、(1) で示しているようなスキーマを発見することができたなら、その遺伝情報を H-GA に伝播してやることにより、H-GA の進化を促進させ、よりすばやく充足解へと導くことができると考える。また、制約充足という観点より、(2) の意味での有用スキーマは、特定されている遺伝子座間の整合性評価を用いることを考える：制約を充足している部分解 (スキーマ) が必ず完全充足解のいずれかに包含されるとは限らないが、部分解レベルですでに制約を違反しているようなスキーマは、必ず完全充足解のいずれにも包含されない。つまり、部分解の整合性評価を用いて、スキーマの有用度が近似できると考える。

以下では、P-個体の適応度評価について、これまで提案を行ってきた評価法である (a) 差分法⁶⁾に加えて、制約充足問題に特化した評価法である (b) 制約充足数法について説明

Fig. 2 Process of coevolutionary genetic algorithm

Fig. 3 Mechanism of superposition operator

Fig. 4 A Depiction of the relationship between the genetic space of H-GA and a P-Indiv

する：

(a) 差分法 (DF) 仮に、P-GA によって発見されたスキーマ情報がすでに H-GA によって探索されている場合、H-GA はこの P-GA による「発見」から有益な情報を得ることができないと考えられる。そこで、P-GA に H-GA がまだ探索していない有用なスキーマを探索するようにするために、つぎのような適応度評価を P-GA に対して行なう：それぞれの P-個体について、(i) H-個体を η 回ランダムにサンプルし、(ii) これらのサンプルされた H-個体を $i_1, \dots, i_n$ で表し、新たに生成された個体を $\tilde{i}_1, \dots, \tilde{i}_n$ で表現する。さて、H-個体 i の適応度と P-個体 j の適応度をそれぞれ、 $f(i)$ と F_{DF}^j で表現するとき、P-個体 j の適応度を次式で表現されるような superposition 演算の「貢献」を基にして定義する：

$$F_{DF}^j = \sum_{k=1}^n \max(0, f(\tilde{i}_k) - f(i_k)), \quad (k = 1, \dots, n).$$

すなわち、この適応度関数は superposition 演算の「貢献度」を表現しており、また、式中の差分が負の値を取るときは、この貢献度は 0 とした。

(b) 制約充足数法 (CS) GA の CSP 解決への適用に際して、GA におけるスキーマは CSP における部分解とみなすことができ、部分解の整合性、すなわち、特定されている遺伝子間の整合性を調べることができる。ここではこの特定されている遺伝子間の制約充足度を P-個体の適応度とする。つまり、この適応度評価法における P-個体の適応度 F_{CS}^j は

$$F_{CS}^j = \sum_{i,k \in sp(j)} I_{C(i,k)}(v_j(i), v_j(k))$$

となる。ここで、 $sp(j)$ は P-個体 j で指定されている遺伝子座の集合を表し、 $C(i, k)$ は、エッジ i, k に規定された制約の集合、 $v_j(i)$ は個体 j の遺伝子座 i における値を示しており、さらに $I_{C(i,k)}(v_j(i), v_j(k))$ は次式で表される関数である。

Fig. 5 The differences among the two fitness evaluations for P-GA introduced in this paper

$$I_{C(i,k)}(v_j(i), v_j(k)) = \begin{cases} 1 & \text{if } (v_j(i), v_j(k)) \in C(i, k) \\ 0 & \text{otherwise,} \end{cases}$$

さて、ここに挙げた P-GA のための 2 つの適応度評価方法の違いについて述べる。Fig. 5 に示すように、差分法 (図中 (a)) では、適応度評価を行なう際に、P-個体が特定している遺伝子座間に張られている制約関係のほかに、P-個体とその他の変数との間に張られた制約関係について適応度評価を行なっている。つまり、この評価方法では、これらの制約関係のうち、制約充足・制約違反といった評価が H-個体と異なる制約関係についてのみ、P-個体の適応度評価に用いている。一方、図中 (b) の制約充足数法では、P-個体が特定している変数 (遺伝子座) 間に張られている制約関係の整合性について着目し、それを適応度に反映している。つまり、この図では P-個体が白丸の 3 変数についてドントケアシンボルでない値が指定していたときに、3 変数間の制約チェックを行ない、充足制約数をその適応度としている。

3.3 Transcription 演算子

superpose された個体は P-個体の適応度計算に用いられるが、この時点では、H-GA の集団は superposition 演算子による影響を全く受けない。そこで、P-GA によって得られた有用スキーマ情報を H-GA に転写することを考える。すなわち、もともとの H-個体と superpose された個体がある確率で入れ換えるような演算を考える。もともとの H-個体を i_h 、superpose された個体 $\tilde{i}_h$ としたとき、transcription 演算が適用される確率 P_{tr} を次のように定義する：

$$P_{tr} = \begin{cases} 1, & D(i_h, \tilde{i}_h) > D(\max(t), \min(t)) \\ \frac{D(i_h, \tilde{i}_h)}{D(\max(t), \min(t))}, & \text{otherwise.} \end{cases}$$

ここで、 $\max(t)$ と $\min(t)$ はそれぞれ t 世代目における H-GA の最大適応度と最小適応度をもつ個体を表しており、 $D(A, B)$ は 2 個体 A, B の適応度 f_A, f_B を利用してつぎのように計算する：

$$D(A, B) = \max(0, f_A - f_B)$$

これらの式からわかるように、もともとの H-個体と super-

Fig. 6 Process of our Genetic Repair Operation for a P-indiv

poseされた個体の適応度の差が大きいほど高い確率で H-GA に情報が転写される。また、 $\max(t)$ と $\min(t)$ の差が少ないとき、より高い確率で情報転写がなされる。このとき、H-GA の探索が高原状態にあると考えられ、積極的に P-GA による探索結果を伝播することによって H-GA の高原状態からの脱却を試みる。

3.4 確率的リペア演算子

制約充足問題解決では、P-個体が表現する部分解が整合性を保っているとき、P-GA は効果的に有用な解を組み立てることができると考えられる。そこで、部分解（スキーマ）の整合性を保つために、P-個体に確率的リペア操作（演算子）を適用する^(注1)。

確率的リペア操作は Fig. 6 に描かれているような手順で行なわれる：図の右上に書かれた “*1*32*” が今からリペア操作を行なう P-個体とする。まず、値が指定されている遺伝子座に対して、リペア操作を行なう順序をランダムに指定する。つぎに、この順序にしたがって整合性判定に基づくリペア操作を行なう。たとえば、図中右側 2 行目の “1” という遺伝子がはいっている遺伝子座では、順序のなかで前方にある “3” という遺伝子についてのみ制約チェックを行なう。そして、制約違反をしていない対立遺伝子をランダムに選択する（ここでは例として 4 が選択されている）。その後順序付けられている “2” という遺伝子がはいっている遺伝子座でも同様に前方にある遺伝子（ここでは “3” と “4”）に対して制約チェックを行ない、制約違反数が最小となる対立遺伝子を選択する。このとき、複数の対立遺伝子が制約違反最小となり得る場合、それらの中からランダムに選択する。最後に、もとの並びに遺伝子座を戻すことでこの演算が終了する。

(注1) ここでリペア操作とは制約違反を減らすための局所的操作であり、以下の手順で行なわれる：1) ある遺伝子座に着目し、その遺伝子座に入りうる対立遺伝子のそれぞれについて、個体の他の変数（遺伝子）に割り当てられている値との間に制約違反があるかどうかをチェックし、あればカウントする。2) すべての対立遺伝子について制約違反数をカウントした後、その中から制約違反が最小になる対立遺伝子を選ぶ。

本論文で導入したリペア操作は、各遺伝子座について、高々 1 回ずつしかリペア操作を行なわない。さらに、リペア操作を行なう順番は、ランダムに決定している。この理由は、P-個体のそれぞれが同一の局所解に陥ることを防ぐためである。したがって、この演算子により生成される新たな解はランダムに指定された順序に大きく依存して変わる。本論文では、この確率的リペア操作を行なう前に、あらかじめ決められた突然変異確率にしたがって突然変異操作を適用し、P-個体集団における多様性維持を図る。

4. 実験結果

4.1 パラメータ設定

本論文では、制約充足問題の変数を個体の文字列長とし、それぞれの変数に定義されているラベルの集合を対立遺伝子とするようなコーディング法を用いた。制約充足問題を解くための適応度関数として、すなわち、従来型 GA として用いた SSGA、Eiben らの Heuristic GA や共進化型 GA における H-GA の適応度関数として、

$$1 - \text{制約違反数} / \text{最大制約違反数}$$

という式を用いた。また、実験では 10 種類の乱数シードを用いて実験し、その平均値を実験の結果とした。

リスタート法は制約違反最小化戦略を山登り法として用い、1000 回、制約違反数を減少することができなければ、初期値を生成し直して探索を再開した³⁾。Heuristic GA は文献¹¹⁾で提案されている 3 つのタイプの中で一番良い結果を与えているものを用い、さらにそこに記載されているパラメータ値をそのまま用いた：世代交代モデルとして Steady-State Selection を用い、集団サイズを 10、遺伝演算子は Asexual Heuristic Operator と Random Mutation とした。Asexual Heuristic Operator では、文字列長の 1/4 個の遺伝子座について、それぞれの遺伝子座における制約違反数が最小となる対立遺伝子を選ぶ。Random Mutation の突然変異確率 P_m は $P_m = 1/(\text{文字列長})$ とした。また、Steady-State GA は、世代交代モデルとして Steady-State Selection を、交叉演算子、突然変異演算子として SGA¹⁹⁾ で用いられている 2 点交叉、突然変異を実装した。集団サイズ、交叉確率 P_c 、突然変異確率 P_m は、それぞれ、10、 $P_c = 0.8$ 、 $P_m = 1/(\text{文字列長})$ とした。一方、提案手法の GA パラメータとして以下のものを用いた：H-GA と P-GA の個体数は 10 とし、世代交代モデルは Heuristic GA と同じく、Steady-State Selection を用いた。ただし、P-GA の適応度評価法として差分法を用いる場合、世代交代モデルとしてルーレット選択を用いた。交叉確率 P_c 、突然変異確率 P_m をそれぞれ、 $P_c = 0.8$ 、 $P_m = 1/(\text{文字列長})$ とした。また、P-GA の初期個体生成時に、ドントケアシンボルの発生確率が高くなるよう（ドントケアでない遺伝子座が文字列長の 1/10 程度）に設定した。

4.2 一般制約充足問題への適用

本節では、変数の個数が 70 でそれぞれの変数に関連付け

Fig. 7 success ratio of finding satisfiable solutions (UPPER) and the average number of constraint checks until finding satisfiable solutions (LOWER) against variety of couples (*density*, *tightness*) over 10 examinations for each couple

られたドメインの大きさは10であるような一般制約充足問題に基づいて、提案手法、リスタート法、Eibenらによって提案されているリペア操作を併用した Heuristic GA¹¹⁾、さらにリペア操作を用いていない従来型進化型計算手法である Steady-State GA¹⁸⁾との比較を行なう。一般制約充足問題では2節で述べた問題の難易度を表現する制約拘束度と制約密度に依拠した二つのパラメータをもとに問題をランダムに生成する。一般制約充足問題はつぎのようにして生成される：まず、制約関係の存在割合とそれぞれの制約関係間に内在する制約の存在割合を示す *density* と *tightness* を指定する。つぎに、2変数のすべての組み合わせについて、*density* で指定された割合をもとに制約関係が存在するか否か決定する。さらに、ここで生成された制約関係すべてについて、*tightness* で指定された割合に応じて、制約関係に内在する制約を規定する。

Fig. 7 に示されているようにさまざまな *density* と *tightness* を用いて生成した一般制約充足問題を用いて実験を行なった。*density* と *tightness* は、それぞれ、10 から 90 まで 10 刻みで変化させて実験した。本実験では制約チェック数が 20 億回となるとところで計算を打ち切り、それまでに充足解を発見できなかった試行を失敗とした。図中、上側のグラフが、充足解発見率を示している。ここで充足解発見率とは、100 試行中の成功試行の割合を表している。白色の領域が全試行（発見率 1）で充足解を発見したことを表しており、逆に、グレーの領域が 0.2~0 の発見率でしか充足解を探索できなかったことを示している。一方、下側のグラフは充足解発見に至るまでの平均制約チェック数を表わしてい

る。ただし、失敗試行の場合、その試行に関する制約チェック数は 20 億回とした。このグラフでは対数スケールで色を変えており、横線の領域は制約チェック数が $10^5 \sim 10^6$ 回の間で充足解が発見されたことを、メッシュ状の領域では $10^8 \sim 10^9$ でようやく充足解が発見できたか、試行中に充足解が発見できなかったことを示している。

これらのグラフより、制約充足型の共進化型 GA (CS) は、完全に充足解を発見した領域、すなわち、Fig. 7 の上側のグラフで白い領域が一番広く、良く充足解を発見していたことが分かる。しかしながら、グレーの領域、低割合でしか解くことができなかった領域は、Heuristic GA の方が良い結果を示したといえる。また、Heuristic GA は解が見つかるときは、提案手法よりすばやく探索しているが、グラフ左上（制約関係が密に存在して、満たすべき制約がたくさんあるようなところ）で白い領域が少ないことより、比較的容易な問題に対しても、時々、充足解ではなく、局所解に収束してしまっていることがわかる。いずれの手法についても、問題の性質が激変する、いわゆる“Phase transition”がグラフの右上から左下にかけて起きている²⁰⁾。このような領域では“発見率が低く、かつ、解が見つかったとしても、それに手間がかかっている”。このような領域と比較的容易にとける領域の境界領域において、CS は、比較的速くそしてよく解を発見しているので、良い性能を示していると考えられる。この実験よりリペア操作を一切用いていない従来手法である SSGA や共進化型 GA の実験結果がリペア操作を用いた Heuristic GA や CS と比べて充足解発見割合、Heuristic GA 充足解発見制約チェック数の点で劣っていることがわかる。これは、

Fig. 8 An instance of Dynamic Constraint Satisfaction Problems: At every interval ivl , the nature of instance is randomly changed

Fig. 9 Success ratio of tracking new satisfiable solutions in changed environment against variety of intervals ivl and variety of couples ($density$, $tightness$) over 10 examinations for each couple

CSP 適用時におけるリペア操作の有用性を表しているのではないかと考えられる。また、リスタート法は問題規模が大きいためいずれの GA よりも悪い結果となった。

4.3 動的制約充足問題への適用

つぎに、動的な一般制約充足問題に対していくつかの計算機シミュレーションを行なった。本論文では、動的な一般制約充足問題を一般制約充足問題の列で表現し、時定数 ivl で問題の性質が変わるものとする²¹⁾。本論文ではこの時定数における1時刻を1回の適応度評価とした。環境変化が行われるたびに前時刻の一般制約充足問題に対して、制約を追加したり、取り除くことで環境変化後の問題を構成する。本論文では、ランダムに環境変化を行なう変数を決定し、その変数と他の変数について、問題生成時に指定された $density$ と $tightness$ を用いて、再度、制約を張り直す。このとき、環境変化を行なう変数の割合をあらかじめパラメータ $frac$ として指定することにする。

ここで、環境への追従度とは環境変化が起きた時から、次の環境変化まで、すなわち、環境変化後から ivl 時刻の間に充足解が見つかれば1、見つからなければ0とし、試行中のすべての環境変化について平均を取ったものである。なお、

Fig. 10 Success ratio of tracking new satisfiable solutions in changed environment against variety of changing-fraction $frac$ and variety of couples ($density$, $tightness$) over 10 examinations for each couple

グラフでは、さらに、乱数のシードを替え10試行を計算し、平均を取っている。この実験では、問題生成とアルゴリズムについて乱数系列を分けているので、試行を行なったすべてのアルゴリズムについて、同じ環境変化すなわち、CSとHeuristic GAに対して同じ一般制約充足問題からなる問題系列を与えて実験を行なった。変数の個数が20でそれぞれの変数に関連付けられたドメインの大きさは10であるような問題に対して、前節の実験で成績が良かったHeuristic GAとCSについて比較する。Fig. 9では、 $fraction$ を0.3とし、環境変化の起こるインターバルを表す ivl を1000, 2000, 5000回と変えて実験を行ない、一方、Fig. 10は、 ivl を5000、とし、環境変化時における問題変化の度合いを表す $fraction$ を0.3, 0.5, 0.7と変えて実験を行なった。図中のグラフはFig. 7の上側の制約充足解発見割合と同じ形式をしているが、色塗りをされた領域が表しているのは、充足解発見割合ではなく、環境変化への追従割合である。前節のFig. 7の結果では充足解発見までの制約チェック数はHeuristic GAの方が結果が良かったが、環境変化への追従性では、提案手法の方がよい結果を示したといえる。提案手法は解候補集合とスキーマ集合との2つの集団を用いて探索しており、さらに、スキーマレベルにリペア操作を加えている。解候補よりもスキーマの方がオーダが短い分、解候補にリペア操作を加えるよりも容易に環境変化へ追従できたのではないかと考えられる。

4.4 構造化された制約充足問題への適用

最後に、「構造化された」制約充足問題について実験を行なう。ここでは、与えた制約充足問題がいくつかの密に結合したクラスタ構造をしており、それらが疎に結合した問題を用いて比較・検討を行なう。Fig. 11に示すように、本論文で用いたクラスタ構造のCSPは一つのクラスタが5変数からなり、さらにそのクラスタ4つで構成される、計20変数か

Fig. 11 A depiction of "latent" structured CSPs

Fig. 12 Success ratio of finding satisfiable solutions (UPPER) and the average number of fitness evaluations until finding satisfiable solutions (LOWER) versus various weights

らなる問題を用いた。ここでは、図中で表されているように、クラスタ間を結合する制約関係のことをクラスタ間制約関係 (Interconnecting Cluster Relations) と呼ぶ。この構造化 CSP は 2 つのパラメータ p, q を用いて構成される。ここで、 p は相互結合制約関係における制約拘束度を表わしており、 q はクラスタに内在する制約関係における制約拘束度を表わしている。ここでは、この p と q に 30 という値を設定した。さらに、本実験では、クラスタ間の epistasis を変えるために、相互結合制約を違反した際に、その制約違反に $weight$ だけ重み付けを行なった。この $weight$ を、それぞれ、1, 2, 3, 5, 10 と設定し、先ほどの一般制約充足問題の時と同じように 100 試行に対する充足解発見率、充足解発見までの平均適応度回数を調べた。

Fig. 12 に描かれているように、共進化型 GA はこのような構造化された問題についても良い結果を得ることができた。確率的リペア演算子が、潜在的に埋め込まれたクラスタ構造を検出していると考えられる。このことを確認するために、ある試行における各クラスタ内の制約とクラスタ間制約をどれだけの P-個体が探索しているかについて調べた：まず、クラスタ c の P-個体 j (スキーマ) による占有率 O_c を次のように定める。

$$O_c = \frac{\text{スキーマの中で } c \text{ に属する変数の個数}}{|c|}$$

さらに、クラスタ間制約関係全体を s で表わし、クラスタ間制約関係に対するスキーマの占有率 O_s も同様に定める。

$$O_s = \frac{\text{スキーマの中で } s \text{ に含まれる制約関係の個数}}{|s|}$$

ただし、これらの占有率は制約が充足もしくは違反しているかについては考慮していないことに留意する必要がある。これは P-個体がどのクラスタを探索しているかについて調べるためにこの指標を導入したからである。各世代で P-GA に内在する個体についてこれらの占有率を調べ平均を取ったものを Fig. 13 に示す。このグラフは平均値ではなく、典型的な実行結果を選んだ。この図からわかるように DF では、各クラスタに分散して探索している様子が見える。一方、CS では $weight$ が 10 になると、先にクラスタ間に張られた制約関係を探索するようになることがわかる。すなわち、 $weight$ が 1 の時は、クラスタ間の制約関係に対する占有率が 1 にならないことがしばしばあるのに対し、 $weight$ が 10 になると、まずクラスタ間の制約関係を探索している様子が伺える。このように、 $weight$ の違い、つまり問題構造の違いに応じて探索状況が異なっており、CS における P-GA は問題の構造を捉えていると考えられる。

5. む す び

制約充足問題の特徴づけるパラメータである $density$ と $tightness$ の様々な値の組合せについて一般制約充足問題を生成し、充足解発見率、充足解に至るまでの適応度評価回数という 2 つの指標に着目した計算機実験より共進化型 GA の有効性が確認された。そこでは、提案手法はリペア操作を用いている従来型手法である Heuristic GA よりもより確実に充足解を見出していることが確認された。比較的容易な問題を解く際、Heuristic GA の方が少ない適応度評価回数で問題解決していることが多かった。しかしながら、動的制約充足問題では、問題変化の割合が大きくなったり、問題変化の頻度が高くなればなるほど、提案手法は環境変化への追従性という点で Heuristic GA より良いが得られた。さらにリペア演算子をまったく用いていない Steady-State GA との比較により、進化型計算手法の CSP 適用時におけるリペア演算子の重要性も確認された。

また、クラスタ構造を内包した制約充足問題についても実験を行なった。ここでは、スキーマによるクラスタの占有率

Fig. 13 the changes of occupancies of P-Indiv.'s in each cluster and in constraint relations interconnecting clusters

という指標を導入することにより、P-GAの探索状況を確認することができた。本論文で提案した制約充足数法の適応度評価法をP-GAに導入することにより、P-GAは、問題の構造を反映しつつ、短いスキーマからより長いスキーマへと構成しながら探索していることがわかった。このことが、提案手法(CS)が、従来型手法よりよい性能を示すことができた主要因ではないかと考えられる。

参考文献

- 1) E. Tsang: Foundation of Constraint Satisfaction, Academic Press (1993)
- 2) 西原清一: 制約充足問題の基礎と展望, 人工知能学会誌, **12**-3, 351/358 (1997)
- 3) 狩野均: 制約充足問題の近似解法, 人工知能学会誌, **12**-3, 359/365 (1997)
- 4) H. Handa, N. Baba *et al.*: Genetic Algorithm involving Coevolution Mechanism to Search for Effective Genetic Information, Proc. of the 4th ICEC, 709/714 (1997)
- 5) H. Handa, O. Katai *et al.*: Coevolutionary Genetic Algorithm for Constraint Satisfaction with a Genetic Repair Operator for Effective Schemata Formation, Proc. of the SMC99, **III**, 617/621 (1999)
- 6) 半田, 片井, 馬場剛夫, 榎木, 小西, 馬場充: 共進化の機構を用いた遺伝的アルゴリズムの提案, 計測自動制御学会論文集, **35**-11, 1438/1446 (1999)
- 7) A.E. Eiben and Zs. Ruttkay: Self-asaptivity for Constraint Satisfaction Problems: Learning Penalty Functions, Proc. of 3rd ICEC, 258/261 (1996)
- 8) M.C. Riff Rojas: Using the Knowledge of the Constraints Network to Design an Evolutionary Algorithm that solves CSP, Proc. of 3rd ICEC, 279/284 (1996)
- 9) A.E. Eiben, P.E. Raue *et al.*: Solving Constraint Satisfaction Problems Using Genetic Algorithms, Proc. of 1st ICEC, 542/545 (1994)
- 10) E. Marchiori: Combining Constraint Processing and Genetic Algorithms for Constraint Satisfaction Problems, Proc. of 7th ICGA, 330/337 (1997)
- 11) B.G.W. Craenen, A.E. Eiben *et al.*: Solving Constraint Satisfaction Problems with Heuristic-based Evolutionary Algorithms, Proc. of the 2000 CEC, 1571/1577 (2000)
- 12) G. Dozier, J. Bowen *et al.*: Solving Small and Large Scale Constraint Satisfaction Problems Using a Heuristic-Based Microgenetic Algorithm, Proc. of 1st ICEC, 306/311, (1994)
- 13) W. D. Hills: Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure, Artificial Life II, 313/324 (1992)
- 14) J. Paredis: Co-evolutionary Constraint Satisfaction, Proc. of PPSN III, 46/55 (1994)
- 15) 久保田, 下島, 福田: ウイルス進化論に基づく遺伝的アルゴリズム, 第8回自律分散システム・シンポジウム資料, 167/170 (1996)
- 16) 狩野, 長谷川, 松本, 西原: ウイルス進化論に基づく制約充足問題の解法, 第23回知能システムシンポジウム資料, 75/80 (1996)
- 17) S. Minton *et al.*: Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method, Proc. of the 8th AAAI, 17/24 (1990)
- 18) G. Syswerda: A Study of Reproduction in Generational and Steady-State Genetic Algorithms, Foundations of Genetic Algorithms, G.J.E Rawlins Eds., Morgan Kaufmann, 94/101 (1990)
- 19) D.E. Goldberg: Genetic Algorithm in Search, Optimization, and Machine Learning, Addison-Wesley (1989)
- 20) I.P. Gent, E. MacIntyre *et al.*: Scaling Effects in the CSP Phase Transition, Proc. of CP95, 70/87 (1995)
- 21) 横尾, 平山: CSPの新しい展開: 分散/動的/不完全 CSP, 人工知能学会誌, 人工知能学会誌, **12**-3, 381/389 (1997)

[著者紹介]

半田 久志 (正会員)

1996年大阪教育大学大学院修士課程教育学研究科総合基礎科学専攻修了, 同年京都大学大学院博士後期課程精密工学専攻入学, 98年同大学院退学, 同年より岡山大学工学部情報工学科助手, 現在に至る。遺伝的アルゴリズムの研究に従事。IEEE, ISGEC, JSAIの会員。修士(学術)

片井 修 (正会員)

1969年京都大学工学部機械工学科卒業。同大学院博士課程機械工学第二専攻修了。74年同大学助手。83年同助教授。94年同教授。96年同大学院工学研究科教授。98年同大学院情報学研究科教授, 現在に至る。その間, 80~81年フランス国INRIA客員研究員。主としてシステムの知能化に関する研究に従事。(工学博士)

渡部克之

1997 年京都大学工学部精密工学科卒業。99 年同大学院博士前期課程精密工学専攻修了，同年日本 IBM 入社，現在に至る。修士（工学）

小 西 忠 孝 (正会員)

1960年京都大学工学部機械工学科卒業。65年同大学院博士課程機械工学専攻修了，同年同大学工学部助教授，82年岡山大学工学部機械工学科教授，89年同情報工学科教授，現在に至る。設計制御，計測工学などの研究に従事。日本機械学会，日本トライボロジー学会などの会員（工学博士）。

馬場 充（正会員）

1979年京都大学大学院工学研究科修士課程修了，同年NKKに入社，82年岡山大学工学部助手，90年岡山大学工学部情報工学科助教授，現在に至る。おもに，三次元画像計測，センサ信号処理，知能化計測などの研究に従事，電子情報通信学会，IEEE，SPIEなどの会員，(工学博士)

|||||