人　工　知　能 93－1
（1994．3．8）

スキーマ処理に基づく集団型探索アルゴリズム

相澤 彰子
学術情報センター

本稿では，確率的スキーマ貪欲法（Stochastic Schemata Exploiter, SSE）と呼ぶ新しい集団型探索アルゴリズムを提案する．SSE は遺伝的アルゴリズム（Genetic Algorithms, GA）と同様に，スキーマと呼ばれる超平面表現の処理により解空間の探索を進めるが，GA と比較して局所的探索処理を重視している点が特徴である．従来より GA に関しては，その適応的な大域的探索能力が長所として強調されてきた．これに対して SSE では，現実の最適化問題への適用では GA の大域的探索能力が必ずしも有効な形で反映されないという観点から，GA の大域的探索処理を，集団型探索の特徴を生かしつつ簡単化して，制御パラメタの数が少なく単純な探索法を実現している．論文中では，GA におけるスキーマ処理を概観した後にスキーマ貪欲と呼ぶ性質を定義し，これに基づきスキーマ貪欲な探索アルゴリズムを構成，簡単な GA 容易/困難テスト問題を用いて単純 GA と比較・評価を行っている．

Evolving a New Population-based Search Scheme based on Schemata Processing

Akiko N. AIZAWA
National Center for Science Information Systems
3-29-1 Otsuka, Bunkyo-ku, Tokyo 112, Japan
e-mail : akiko@nacsis.ac.jp

This paper proposes Stochastic Schemata Exploiter (SSE), a new population-oriented search scheme which employs a schemata processing mechanism similar to the one used in Genetic Algorithms (GAs). Compared with the basic GA, SSE has the following features : first, SSE exploits more the ability of local search of schemata processing while the GA puts more emphasis on the ability of adaptive global search. Second, SSE reduces the number of control parameters which are in many cases problem dependent and should be determined heuristically. In the paper, the advantage of SSE is demonstrated using GA-easy and GA-hard test functions.

1 はじめに

遺伝的アルゴリズム (Genetic Algorithms, 以下 GA) は，生物の進化における適応化のプロセスを模倣することにより，各種の最適化問題を解こうとする手法である．GA の基本的な枠組は 1970 年代に Holland らにより定式化されたが，特に 1980 年代の後半になり計算機の高性能化や並列化が進むと，ニューラルネットワークや焼きなまし法 (simulated annealing) などに続く新しい最適化手法として一般に注目されるようになった．

探索手法としての GA の工学的な側面が強調されるのに伴い，その有効性を示すために，アルゴリズムの頑強性や効率性を改善する様々な努力が行われるようになった．頑強性に関しては，GA が苦手な問題を「だまし問題」(deceptive problems) として定義し [8] [18]，これを克服するための数々の工夫が提案されている [10] [16]．また効率性に関しては，ヒューリスティクスとの結合による問題領域ごとの効率化に関する研究が盛んである [14]．このような動きの中で，GA を単独で適用するのではなく，異なる設定のもとで並列動作させたり，焼きなまし法などの他の探索アルゴリズムと併用したりするハイブリッド手法の有効性が注目されるようになった．ハイブリッド手法の例として，たとえば，[13] [17] [15] などがあげられる．

本稿では，GA のこのようなハイブリッド手法への適用においては，必ずしも汎用的な探索能力は必要とされないことに注目する．その理由は，多くのハイブリッド手法が相補的な役割を果たす探索法の組み合わせで実現されているためである．多数の制御パラメタが GA の全体的なふるまいに影響を与えてしまうとハイブリッド手法の設計が困難になることから，このような場合には汎用性よりはむしろ，GA の持つ特徴を生かしたより単純なアルゴリズムを用いることが望ましいと考えられる．

以上を考慮して本稿では，GA の特徴である集団型の探索処理に注目した新しい探索アルゴリズムの構成を試みる．本稿で提案する手法はテスト問題を用いた評価によると，GA よりも優れた局所的探索能力を持ち，かつ GA とほぼ同様の大域的探索能力を実現している．加えて GA よりも制御パラメタの数が少ないため，ハイブリッド手法の構成要素としても適したものである．

以下，**2.** で GA における探索処理の概要および特徴について述べる．次に，**3.** で GA において特徴的なスキーマと呼ばれる部分解の処理モデルを検討する．また，**4.** で GA の局所的な探索能力に注目してスキーマ貪欲と呼ぶ性質を定義する．そして，**5.** で実際にスキーマ貪欲な探索アルゴリズムの構成を試みる．**6.** ではテスト問題を用いた動作比較を行う．

なお本稿は，いわゆる単純 GA で扱える問題を対象としている．すべての個体は等しい長さの 2 進数ストリングで表現されているものとし，コード化や遺伝的操作子の適用において，問題領域に固有の特別なヒューリスティクスは考慮していない．

2 GA における探索処理

2.1 GA における探索処理の概要

GA の探索問題への適用では一般に，解空間として各座標軸が '0' または '1' の 2 値であるような多次元空間を扱う．解空間上の個々の点は個体と呼ばれる．各個体は，その座標軸上での値を並べた 2 進数ストリングにより表現されている．GA の実行においては，各個体に評価値を与える関数，すなわち最適化の目的関数，は予め準備されていると考える．以下，各個体を c_i で，対応する 2 進数ストリングを $\boldsymbol{x}_i$ で表記する．評価関数を f で表すとき，個体 c_i の評価値は $f(\boldsymbol{x}_i)$ である．

GA における探索処理は，世代と呼ばれる個体集団の逐次生成により進行する．以下，第 t 世代における個体集団を P_t と表記する．新たな個体集団 P_{t+1} の生成では，まず P_t に含まれるすべての個体に f を適用して各々の評価値を求める．次に，集団中の相対評価値に応じて確率的に優れた個体を選択する．これを淘汰という．そして最後に，交叉や突然変異と呼ばれる遺伝操作子を適用して，新しい世代 P_{t+1} を生成する．初期集団は通常の場合ランダムに生成される．

GA ではこのように，より優れた個体がより大きな確率で次世代に生き残ったり，類似の個体を新たに生成したりしながら，適応的に有望な領域の探索を実現している．

2.2 GA における探索処理の特徴

GA が他の探索手法と比較して特徴的なのは，解空間上で複数の探索点を同時に保持しつつ処理を進めることである．

このような探索処理の特質を調べるため，GA に代表される集団型探索法 (population-based search) を，焼きなまし法などに代表される点型探索法 (point-based search) と対比させながら以下に検討を行う．ここで点型探索法とは，探索の過程において 1 個の解 (incumbent) を保持し，これに対して解空間上で定義される近傍解を調べることにより処理が進行するものを指す．また集団型探索法とは，探索の過程において複数個の解集団を保持し，これらの間で共有される解空間の中の超平面 (次節で述べる，いわゆるスキーマと呼ばれる部分解) に基づいて処理が進行するものを示すこととする．

(1) 探索処理の双対性

一般に確率的な探索プロセスは，2 つの双対的な処理がバランスをとりながら進行するものと考えられている．その 1 つは，限定した領域の中で優れた解を追求する局所的な処理であり，もう 1 つは，未知の領域の中から有望な新しい解を探索する大域的な処理である．GA に関する文献において，前者は exploitation, convergence, profiting などの言葉で参照され，これに対応して後者は，exploration, divergence, acquiring などの言葉で呼ばれる．

対極に位置するこれら 2 つの処理の間にはトレードオフ関係が存在する．一般にこの関係は与えられた問題に依存し，よいトレードオフを決定する有力な方法は知られていない．したがって，優れた可能性を持つ探索アルゴリズムでも，実際の問題に適用しようとする場合には，両者の適当なバランスを実現する制御パラメタの調整がアルゴリズムの性能を決める重要な要因となる．

(2) 点型探索法の探索処理

近傍処理に基づく点型探索法において局所的処理に相当するのは，いわゆる山登り法 (hill climbing) である．ここで山登り法とは，解空間上の近傍点の中から解を改善するものを選び，その方向に探索を進める広い意味で貪欲的な方法である．山登り法による探索は，限定され

た解空間中での局所的な最適解を目指す．これに対して大域的処理を実現するのが，山登りに対する山下り，すなわち近傍解の中から悪い解を受け入れる動作である．

焼きなまし法などの確率的山登り法（Stochastic Hill Climbing, SHC）では上記の 2 つの処理を同時に実現する [3]．すなわち，局所探索を行う山登り法に対して，確率的に悪い解を受け入れることにより探索範囲を広げ，局所最適解から脱出することを目指すものである．

確率的山登り法においても，局所的および大域的処理のバランスをとるために，悪い解の受け入れ確率をうまく制御することが必要である．たとえば焼きなまし法において，焼きなまし温度のスケジューリングは問題依存であり，これをいかに調節するかが個々の問題への適用において重要になっている．なお，GA でも操作子として突然変異のみを用いる場合には，確率的山登り法の一つであるとみなすのが一般的である．

(3) 集団型探索法の探索処理

GA においても，様々な制御パラメタが局所的処理と大域的処理のバランスに影響を与えることは，従来より多くの研究者によって指摘されてきた．たとえば GA の制御パラメタとして，文献 [7] では，集団の大きさ，交叉率，突然変異率，世代交代率，最適解保存戦略の有無，評価値のスケーリングのためのウィンドウサイズ，の 6 つをあげている．このうちの集団の大きさが小さければ探索は局所的であり，大きければ大域的である．また，突然変異率が小さければ局所的，大きければ大域的処理が重視であるといえる．しかし，GA などの集団型探索法においてはアルゴリズム自体が複雑なため，局所的および大域的処理が具体的にどのように実現されているかは自明ではなく，パラメタ値の調整についてヒューリスティクな傾向がより強い．

ここで，GA においては従来より，大域的な探索能力が重視され，局所的な探索能力に関して議論されることが少なかった．その理由の 1 つは，GA の利点として，交叉による適応的な大域的探索能力が強調されてきたことであると考えられる．しかし交叉の有効性はいまだ議論の的であり，近年では交叉が有効な問題は比較的限定されているとの報告もある [4]．

そこで本稿では，GA に特徴的な処理として，GA が与えられた超平面上で行う局所的な探索処理に注目する．そして，GA における局所的探索は近傍処理に基づく山登り法とは異なるという視点から考察を行い，新たな集団型探索法の構成を試みる．このためにまず，次節でスキーマと呼ばれる超平面の表現についてまとめる．

3 スキーマ処理モデル

3.1 スキーマとは

前節で述べたように，GA において個体 c_i は，アルファベット $\{0,1\}$ の L 個の並びにより表記される，ここで，$\{0,1,*\}$ の 3 アルファベットの L 個の並びにより表現される文字列をスキーマ（schema）と呼ぶ．

スキーマは個体間で共有する 2 進数ストリングのパターンを表している．記号 '*' は，対応する文字列の位置に '0' と '1' のどちらが来てもよいことを示す．以下，任意のスキーマを H として，そのスキーマを含むストリングの集合を $s(H)$，要素の個数を $|s(H)|$ で表す．たとえば，$H = '01{*}{*}11'$ のとき，$s(H) = \{ '010011', '010111', '011011', '011111' \}$，$|s(H)| = 4$ である．

スキーマ H の中の '0' または '1' の数をスキーマの次数（order）と呼び，$o(H)$ で表す．スキーマ中の '*' の数は $(L - o(H))$ 個であることから，$|s(H)| = 2^{(L-o(H))}$ である．またスキーマ H の評価値 $f(H)$ を，H を含むすべての個体の評価値の平均として，以下のように定義する．

$$f(H) = \frac{1}{|s(H)|} \sum_{\boldsymbol{x} \in s(H)} f(\boldsymbol{x}) . \qquad (1)$$

解空間上ではスキーマは，部分空間あるいは超平面（hyperplane）に対応している．

3.2 スキーマの統計的なモデル

スキーマの評価値は上記の式 (1) で定義されるが，現実的には，スキーマ H を含む個体は多数存在する場合が多く，$f(H)$ を厳密に求めることは困難である．このため，スキーマ H を統計的にモデル化し，H を含む個体の評価値を H のサンプル（標本点）とみなして評価値を推定することが行われる．

時点 t における個体集団 P_t の中で H を含むものの集合を $s(H,t)$ で表すとき，H の評価値の統計的推定は，$s(H,t)$ の平均値により次のように表せる．

$$\widehat{f}(H,t) = \frac{1}{|s(H,t)|} \sum_{\boldsymbol{x} \in s(H,t)} f(\boldsymbol{x}) . \qquad (2)$$

逆に，与えられたスキーマ H からランダムに標本点をとるためには，H の長さ L の文字列のうち $*$ にあたる文字について，ランダムに 0 または 1 を割り当てればよい．これにより $\boldsymbol{x} \in s(H)$ なる個体 $\boldsymbol{x}$ を発生することができる．

スキーマの統計的なモデルとして一般的なのは，その確率密度関数を正規分布とみなすことである [12]．このとき $f(H)$ は，与えられた正規分布の平均値で表される．異なるスキーマどうしは異なる平均値や分散を持つ．このような正規分布の仮定は，解析を行う上で非常に都合のよいものであり，たとえば最適な個体集団の大きさを決めたり，雑音の影響を考慮したりするための道具として用いられている [5] [9] [2]．

3.3 スキーマ定理

GA は暗黙のうちにスキーマの高並列な処理を行うアルゴリズムだと言われ，この性質はしばしば GA の暗黙的並列処理（implicit parallelism）として参照される．GA のスキーマ処理を数学的に表現したものが，以下に示すスキーマ定理（schema theorem）[12] である．

[定理] スキーマ定理

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H,t)}{\overline{f}(t)} \left[1 - p_d(H)\right] . \qquad (3)$$

ここで $m(H,t)$ は，世代 t においてスキーマ H を含む個体の数（$= |s(H,t)|$），$f(H,t)/\overline{f}(t)$ は世代 t における H の相対評価値，そして $p_d(H)$ は交叉や突然変異などにより世代 $(t+1)$ までに H が破壊される確率である．要約すれば，この定理は，個体集団の中で「平均よりも

よい」スキーマを持つ個体の数が，時間とともに指数関数的に増加することを述べている [11].

GA の実行においてスキーマ定理は，局所的および大域的な処理のバランスを定めていると解釈される．前節で述べた GA の制御パラメタはすべて，スキーマ定理の式 (3) のいずれかの項に影響を与えており，GA におけるスキーマ処理の性質に深く関係している．

3.4 スキーマ定理が意味するもの

では，スキーマ定理において,「よい」スキーマの指数関数的な増加を保証する理由は何故だろうか？ その数学的な根拠は，70 年代に発表された Holland の啓発的論文 [12] における，2 腕スロットマシン問題 (2-armed bandit problem) のアナロジーにさかのぼる．

2 腕スロットマシン問題は，統計学の逐次決定理論の分野で有名な決定問題の 1 つである [6]：左右に 2 本の腕を持つスロットマシンがあり，それぞれの腕を選んだ場合の利益が正規分布 $N(\mu_1, \sigma_1)$, $N(\mu_2, \sigma_2)$ で表されるとする．ただし，$\mu_1 > \mu_2$ とし，左右どちらの腕が μ_1 を持つかは未知である．全体で N 回の試行を行う場合に，利益の総計を最大にするためには左右の腕をそれぞれ何回ずつ選べばよいだろうか？

詳細は省略するが Holland の解析によると，この問題に対する最適戦略は，サンプル平均の大きな腕に対する試行を，ほぼ N の指数に相当するオーダーで増やすことで近似できる．Holland はまた，この結果を $k > 2$ 本の腕を持つ場合に拡張できるとしている．

ここで，k 本の腕をそれぞれ k 個のスキーマに対応すると考え，上記の指数関数的な戦略を $m(H,t)$ の配分則に適用すると，式 (3) のスキーマ定理が説明される．言い換えれば，スキーマ定理は，真に平均値が高いスキーマを漸近的にほぼ最適に識別するサンプル配分を定めていると解釈されるのである．

4 スキーマ貪欲法

4.1 スキーマ貪欲法の定義

集団型探索法はスキーマ処理に基づいており，その探索領域は解空間上の近傍点に限定されていない．したがって局所的探索法としても，点型探索法における山登り法とは異なる定義が必要であると考えられる [19]．本稿では集団型探索法において，解空間上での山登り法に対応する概念をスキーマ貪欲法 (schema exploiting) と呼ぶ．

さて，前節で述べたスキーマ定理の目的は，サンプル平均に基づく動的なサンプルの配分により，適応的な大域的探索を実現することであった．しかし，現実の問題での GA のふるまいにおいて，スキーマ定理が主張する最適な指数関数的戦略が厳密に守られているとは考えにくい．その理由として，経験的に設定される GA の制御パラメタの値がサンプルの配分に大きな影響を持つことがあげられる．これより，交叉により実現される適応的な大域的探索が，突然変異により導入されるランダムな多様性より効率がよいことは，必ずしも保証されるものではない．たとえば [4] では，交叉が有効な問題の性質を述べ，現実的にはこのような条件を満足する問題は限られていると議論している．

以上に基づき本稿では，スキーマ貪欲法を以下のように定義する [1].

[定義] スキーマ貪欲法

既存のスキーマの中で，サンプル平均のよいスキーマをさらにサンプルすること

ここで，スキーマの選択は真の平均値 (式 (1)) ではなくサンプル平均 (式 (2)) に依存することに注意．

言い換えるとスキーマ貪欲法では，観測されたよい個体集合 (個体ではない) に含まれるスキーマを選び，さらに新しい個体を生成する．スキーマ定理では，真に平均値が高いスキーマを識別するため，観測されるサンプル平均が劣るスキーマにも少数のサンプルを配分することを主張していた．これに対してスキーマ貪欲な方法では，サンプル平均がよいものだけが選ばれるのである．

4.2 スキーマ貪欲法の性質

ここで，山登り法における局所最適解に対応して，スキーマ貪欲法により選ばれやすいスキーマを誘引的 (attractive) であると表現することにする．あるスキーマが誘引的であるかどうかはスキーマの真の平均値だけでは決まらない．観測されるサンプル平均の真の平均値からのずれは分散より決まるため，スキーマの平均と分散の両者を考慮することが必要である．

スキーマ処理の複雑さから，多くのスキーマの中でどのスキーマが誘引的であるかを正確に知る現実的な方法は存在しないと予想される．正規分布モデル上で行ったシミュレーションによると，

- 平均値が大きく，かつ
- 分散が小さい

スキーマが，多くの場合誘引的であることが観察されている．これは直観的には，探索法が平均値が高いなだらかな (分散が小さい) 領域に陥りやすいことに対応しており，従来のだまし問題における観察とも一致している．

4.3 スキーマ統計データと GA 容易/困難問題

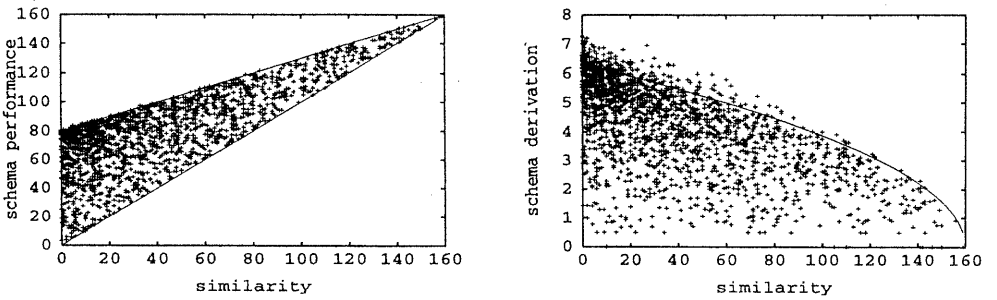
GA 容易 (GA-easy) な問題では，誘引的なスキーマは最適解と多くのビットを共有すると考えられる．一方，いわゆる GA のだまし問題では，誘引的なスキーマは最適解との共有ビットが少なく，結果として探索の方向を最適解とは逆の方向に惑わせると考えられる．そこで，GA のテスト問題の例として，評価関数がストリング $\boldsymbol{x}$ 中の'1' (または '0') の数で与えられる問題 (unitary problem) をとりあげ，スキーマの統計データに基づき GA 容易/困難性の分析を試みる．

まず，ストリング長 L として，ストリング $\boldsymbol{x}$ の中に含まれる '1' の数を $count(\boldsymbol{x})$ で表す．典型的な GA 容易問題である '1' 数え上げ問題では，$\boldsymbol{x}$ に対する評価値は次式で与えられる．

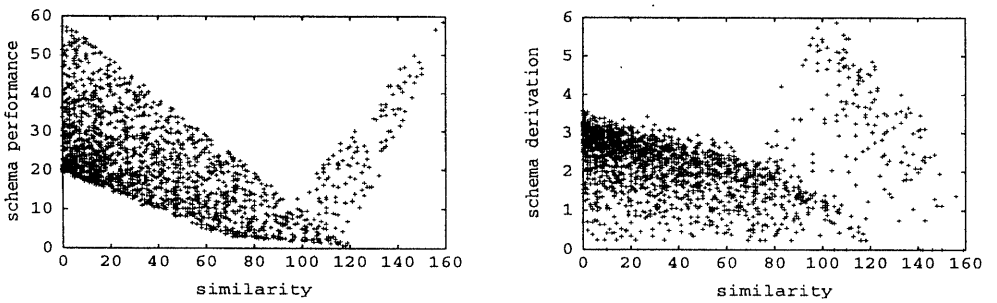
$$f(\boldsymbol{x}) = count(\boldsymbol{x}). \tag{4}$$

一方 GA の困難問題として，$a < b, 0 < c < L$ なる整数 a, b, c を考え，ストリング $\boldsymbol{x}$ に対する評価値が以下の式で定義される場合を考える．

$$f(\boldsymbol{x}) = \begin{cases} \dfrac{a(c - count(\boldsymbol{x}))}{c} & (count(\boldsymbol{x}) < c) \\ \dfrac{b(count(\boldsymbol{x}) - c)}{(L - c)} & (count(\boldsymbol{x}) \geq c) \end{cases}. \tag{5}$$



(a) GA 容易問題の例



(b) GA 部分困難問題の例

図 1 スキーマ統計と GA 容易/困難問題

ここで $c=(L-1)$ のとき上式は完全困難 (fully deceptive) であり，それ以外の場合は部分困難 (partially deceptive) である．

式 (4) と式 (5) のテスト関数について，次数および '1' の数をランダムに選びながら $K(=2000)$ 個のスキーマを生成して，各々 $k(=100)$ 個のサンプル（そのスキーマを含む個体の評価値）から平均および標準偏差を調べた．$L=160$ とした場合の結果を図 1 に示す．グラフ中の各点は個々のスキーマに対応し，横軸に最適解（この場合はすべて '1'）からのハミング距離 ($\leq o(s)$) を，縦軸に平均または標準偏差を示している．

図 1-(a) は GA 容易問題 (式 (4)) の統計データを示す．図中では，横軸の値の大きいスキーマほど平均値が大きく分散が小さい傾向がみられる．すなわち最適解に近いほど誘引的であり，GA にとり好ましい問題であるということができる．なお理論値は，スキーマ H (次数は $o(H)$) について，H の中に含まれる '1' の数を $count(H)$ で表すとき，平均値は $(count(H)+\frac{L-o(H)}{2})$，分散は $(\frac{L-o(H)}{4})$ である．図中の実線は理論値による境界を示す．観測値が計算値とほぼ一致していることから観測の精度は十分であるといえる．

図 1-(b) は GA 部分困難問題 (式 (5) で $a=59$, $b=60$, $c=120$ とした場合) の統計データを示す．スキーマの最適解からの距離と平均，分散の間に逆転関係が見られることがわかる．図中には示していないが，$c=(L-1)$ の完全困難問題では，平均と分散の関係は図 1-(a) の横軸に関してほぼ完全に逆転する．

本稿では，後出の動作比較において，これらの分析により容易/困難性を調べたテスト問題を用いることにする．なお，この方法によるテスト問題の解析はあくまで最適解（あるいは目標解）が与えられている場合のみ有効であり，未知の問題の容易/困難性を判定するためのものではない．

5 スキーマ貪欲な探索アルゴリズムの構成

5.1 探索法の概要

実際にスキーマ貪欲な確率的探索法の構成を試みる．

まず手順 (1) として，個体集団の優れたものの中から共通ビットを取り出し，有用なスキーマを抽出する．このために各個体の評価値に基づき，個体部分集合を平均評価値のよいもの順に並べ，上位の部分集合より共通のスキーマを取り出す．

次に手順 (2) として，得られた有用なスキーマに基づき確率的な方法で新たな個体を生成する．このために手順 (1) で得られたスキーマからランダムに個体を発生するとともに，突然変異操作を用いて集団中に多様性を取り込む．突然変異率はアルゴリズムの制御パラメタとする．

以下，(1)(2) のそれぞれの手順について，具体的な実現方法を述べる．

5.2 手順 (1)：有用なスキーマの抽出手順

一般に，長さ L の 2 進数コードを持つ大きさ M の集団中には，$O(M^3)$ 個ものスキーマが存在するとされる．すべてのスキーマを並べ立てることは現実的ではないことから，ここでは，最良のスキーマから出発して，必要

な数だけのスキーマを上位から順に書き出すことにする．

まず，任意の時点 t において，個体集団 P_t 中の M 個の個体をその評価値の順に並べ，$c_1, c_2, \cdots, c_M$ のようにインデクスをつける．一般性を失うことなく最大化問題を仮定し，$f(x_1) > f(x_2) > f(x_3) > \cdots$ とする．c_1 が最良の個体である．

スキーマ H の評価値は，そのスキーマを含む個体部分集合 $s(H,t)$ の平均評価値より推定される（式(2)）ことに注意すると，P_t に含まれるスキーマのうちで最良の推定評価値を持つのは，$s(H,t) = \{c_1\}$，すなわち c_1 だけに含まれるスキーマである．この中で一番次数が高いスキーマは，c_1 のストリング x_1 そのものである．これを $\Lambda(\{c_1\})$ と表記する．Λ は，P_t の部分集合から，その要素に共有される最大次数のスキーマを取り出すオペレータである（この部分集合により共有される他のスキーマはすべて，最大次数のスキーマにより表されるとみなし，以後特に言及しない）．同様にして，$\frac{f(x_1)+f(x_2)}{2} > f(x_2)$ に注意すると，P_t に含まれるスキーマのうちで2番目に位置づけられるのは，c_1 と c_2 に共通に含まれるスキーマ，すなわち $\Lambda(\{c_1, c_2\})$ となる（図2）．このようにして，スキーマの順位付け問題は，M 個の個体が作るベキ集合の要素 $\{c_1\}, \{c_1, c_2\}, \cdots$ を，その平均値にしたがって並べる問題に置き換えることができる．ただし厳密には P_t の部分集合 S_1, S_2 について，$\Lambda(S_1) = \Lambda(S_1 \vee S_2)$ なる場合もあり得るが，その影響は比較的少ないと考え本稿では考慮しない．

さて，P_t の任意の部分集合 S $(\neq \emptyset)$ について，その中に含まれるインデクス最大の個体を $c_{\,last(S)}$ $(last(S) = 1, \cdots, M)$ と表記する．$last(S) < M$ のとき，P_t の部分集合の平均値の間に以下の半順序関係が存在する．

- $\Lambda(S)$ は $\Lambda(S \vee c_{\,(last(S)+1)})$ よりもよい．
- $\Lambda(S)$ は $\Lambda((S - c_{\,(last(S))}) \vee c_{\,(last(S)+1)})$ よりもよい．

ただし $(S - c_k)$ は，集合 S から要素 c_k を除いた集合とする．図3は，$M = 4$ の場合に，最良の集合 $\{c_1\}$ をルートとして，この半順序で定まる集合の関係を示したものである．簡単のため図中ではインデクスのみ示してある．

以上の半順序関係を利用して，上位から M 個の部分集合を取り出す順位付け手順は次のとおりである．

(1) まず，長さ M の処理用リスト **active_list** を作る．リストの第1要素に $\{c_1\}$ を置く．$i = 1$ とする（ステップ1）．

(2) ステップ i においてリストの第 i 要素 S_i を取り出し，$last(S_i) < M$ ならば，部分順序関係にしたがう2つの部分集合 $S_i \vee c_{\,(last(S_i)+1)}$ および $(S_i - c_{\,last(S_i)}) \vee c_{\,(last(S_i)+1)}$ を生成する．（これらは図3では，親ノードと2つの子ノードの関係で表現される）

(3) 新しい部分集合を **active_list** の $(i+1)$ 番目以降の要素と比較し，よいもの順にリストに挿入する．リストからあふれた要素は捨てる．

(4) $i < (M-1)$ ならば $i = i+1$ として，(2) から (4) を繰り返す．

ステップ $(M-1)$ 終了後，**active_list** 上の部分集合に対してオペレータ Λ をほどこせば，上位より M 個のスキーマが求まる．

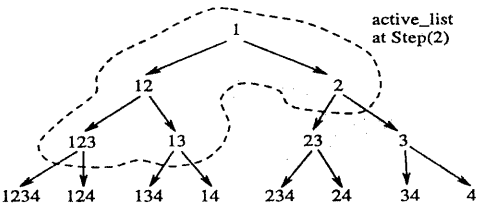


図3 インデクス集合の部分順序関係．

例として，図3の点線で囲まれた部分は，ステップ2で部分集合 $\{c_1\}$ と $\{c_1, c_2\}$ が調べられた時点でのノードの展開状況を示す．この時点で，**active_list** は第1要素 $\{1\}$，第2要素 $\{1,2\}$ であり，第3~5要素には，$\{1,2,3\}$，$\{1,3\}$，$\{2\}$ の3つがソートされて格納される．

各ステップごとに2個の部分集合が生成されることから，吟味される部分集合の数は $2(M-1)$ 個であり，ステップ i において，長さ $(M-i)$ のリストに対してバイナリ挿入が行われる．また最初にM個の個体を評価値のよいもの順にソートしておく必要がある．これより以上の手順による計算時間は $M\,log(M)$ のオーダーである．この方法では単純さを重視し特に最適を意識していないが，現在の計算機の処理能力を考えると，探索アルゴリズムの動作において問題になる程度ではない．

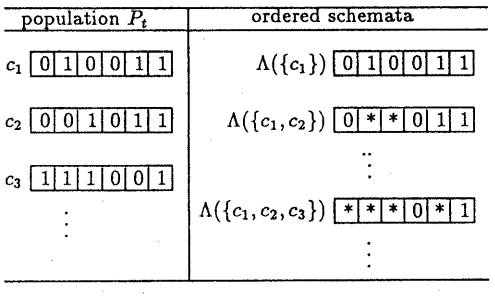


図2 P_t の部分集合とスキーマの対応．

5.3 手順(2)：新しい個体の生成手順

新しい個体の生成手順は以下のとおりである．

(1) 手順(1)で選んだ M 個のスキーマをそれぞれ含む M 個の個体をランダムに生成する．

(2) 新たに生成した個体に対して突然変異を適用する．

以上により生成した個体は，対応する個体部分集合の共有スキーマを含むことから，この操作は，親の数を2個から任意個に拡張した均一交叉（uniform crossover）となっている．

ここで(2)の突然変異を適応的に行う方法として，たとえば，焼きなまし的なスケジューリングを行う，スキーマの順位に応じて非均一な確率を用いる，多様性の尺度を導入する，等，様々な方法が考えられる．しかし，これ

らの方法を詳細に検討することは本稿の目的ではないので，以下では単純に一様な確率による突然変異を用いる．突然変異率はアルゴリズムの制御パラメタとして，適当な値を設定するものとする．

5.4 GA との比較

手順 (1) について，GA におけるスキーマの選択は原則として，単一の個体の評価値を用いて確率的に行われる．一方，ここで検討するアルゴリズムにおいては，スキーマの選択は確率的ではなく，すぐれた個体部分集合で共有される要素として一意に決定される．このため GA と比較して，局所解に引き付けられる「貪欲的な」傾向が強くなると予想される．

手順 (2) について，GA における個体の生成では，交叉により多くの有用なスキーマを効率よく集めることにより，ランダムな探索より高い性能を実現することが特徴であった．一方，ここで検討するアルゴリズムにおいては，スキーマのサンプルはランダムに行われ，組み合わせによる効果は考慮していない．このため GA が効率よく動作している場合には，大域的な探索は不利になる可能性がある．

次節では，**4.3** で検討したテスト問題を用いて，両者の比較を行った結果を示す．

6 テスト問題による動作比較

6.1 実験の条件

実験では，(1) 世代ごとに個体を入れ換える単純 GA (basic GA, 以下 BGA)，(2) GA において，操作子として突然変異のみを用いる確率的山登り法 (Stochastic Hill Climber, 以下 SHC)，および本稿で述べた (3) 確率的スキーマ貪欲法 (Stochastic Schemata Exploiter, 以下 SSE) の 3 者について動作比較を行った．設定したパラメタの値は，BGA, SHC および SSE について，個体集団の大きさ 100，突然変異率 0.001 である．また，最良解保存戦略を用いた．さらに，BGA について交叉率 0.6 とし，BGA と SHC については，スケーリングのウィンドウサイズを文献 [7] にしたがい 7 に設定した．

テスト問題としては，式 (4) および式 (5) で定義した GA 容易/困難問題を用いた．これは，DeJong のテスト問題等を **4.2** の手法により解析した結果，コード化技法により問題の性質が大きく変わることが観察されたため，コード化の影響が少ない問題として上記の 2 つを選んだものである．また，以下の結果中では，各世代 t における最良の個体の評価値を探索アルゴリズムの性能値としている．

6.2 動作比較例 (1)：GA 容易問題への適用

まず，局所的な探索処理能力を比較するため，GAB, SHC, SSE の 3 つを式 (4) で与えられるテスト問題に適用した．この問題は GA 容易であり，GA などスキーマ処理に基づく探索法は容易に最適解に到達することが予想される．

実験では，各探索法が最適解に到達した最初の世代数を記録し，平均値を調べた．図 4 に，ストリング長 L を $(80, 160, \cdots, 480)$ と変化させ，各々についてシミュレーションを 100 回繰り返した結果を示す．いずれの場合についても，SSE は，GAB や SHC と比較して，速い収束性を示していることがわかる．

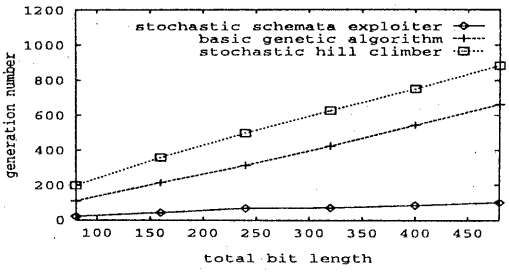


図 4 GA 容易問題への適用による動作比較

6.3 動作比較例 (2)：GA 困難問題への適用

次に，大域的な探索処理能力を比較するため，GAB, SHC, SSE の 3 つを式 (5) で与えられるテスト問題に適用した．この問題は GA 部分困難問題であり，**4.** の手法による分析から，その困難性はテスト問題のパラメタ c に依存すると予想される．

実験では各探索法を，最高 10,000 世代まで実行し，最終的に最適解に到達するかどうかを判定した．10,000 世代目までに最適解を見つけることができればシミュレーションは成功，そうでなければ失敗であると判断した．

図 5 に，ストリング長を $L = 160$ とし，$a = 59$, $c = 60$ に設定して c を変化させた場合の結果を示す．図中，横軸は c の値，縦軸は試行 50 回中の成功率である．これより BGA, SHC, SSE は，ほぼ等しいふるまいを示していることがわかる．すなわち，変数 c の値が $L/2$ $(= 80)$ よりも小さい場合には，これらの探索アルゴリズムの成功率は 1.0 である．一方，c の値が $L/2$ より大きくなると，最適解の探索は例外なく失敗する．これは，これらの探索法がすべて，だまし問題に対して頑強ではないことを示すと考えられる．

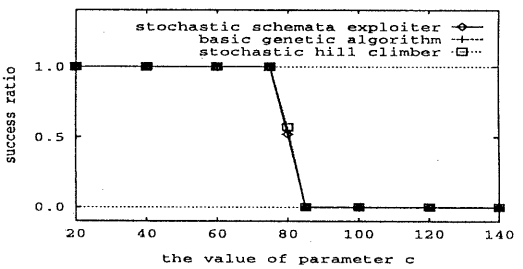


図 5 GA 困難問題への適用による収束率の比較

6.4 動作比較例 (3)：探索法の拡張

最後に，ハイブリット化における BGA と SSE の適性を比較するため，両者に局所解を脱出するための簡単な処理ルーチンを加えて性能を調べた．

この拡張した探索法では，集団が局所最適解に陥った場合に，これを判定して，以下の手順を起動する．まず，対象となる局所解を集団から取り除き，異なるメモリ領域に保存する．次に，その局所解からのハミング距離がランダムに与えられる M 個の個体を生成して，新たな集

団とする．これにより集団中に多様性が導入され，新しい領域の探索が始められる．ただし，この処理は集団が収束した場合にのみ起動されるため，通常の局所的探索動作には影響を与えない．

BGA においては，集団の中の α % 以上の個体が等しい場合に，探索が収束したとみなす．ここでは $\alpha = 80$ % とする．一方，SSE においては，順位づけされた M 個のスキーマの次数の総数が ML，すなわち，どのスキーマも '*' を持たない場合に，探索が収束したと判断する．

図6 に，このようにして拡張した BGA と SSE を 式 (5) で与えられるテスト問題に適用した結果を示す．シミュレーションの条件は，図5の場合と同じである．図中の横軸は c の値，縦軸は 50 回のシミュレーションにおける平均の終了世代数を示している．拡張した探索法では両者とも例外なく，10,000 世代以内に最適解を発見することができた．そして，拡張した SSE は拡張した BGA よりもよい性能を示していることがわかる．

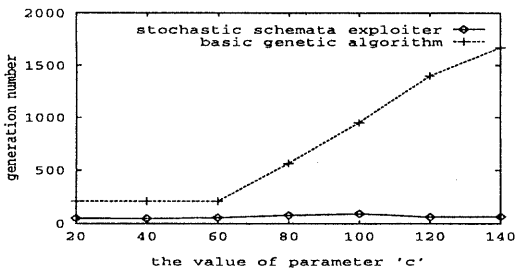


図 6 GA 困難問題への適用による動作比較

7 おわりに

本稿では，SSE (Stochastic Schemata Exploiter) と呼ぶ新しい集団型探索法を提案した．SSE は GA と同様に，スキーマに基づく局所的探索処理を行うが，一方で，簡単化した大域的探索処理を用いているため，設定すべきパラメタの数が少ないという利点がある．論文中では，簡単な GA 容易/困難問題を用いて評価を行い，SSE が単純 GA よりも局所的探索処理に優れ，かつ適用したテスト問題の範囲で，ほぼ同等の大域的探索処理の能力を持つことを示した．

SSE は，はやい収束特性を持ち，また制御パラメタの数も少ないことから，ハイブリッド探索法への適用においても有効であると考えられる．本稿では，ハイブリッド手法の簡単な例として，SSE と単純 GA に対して局所解を脱出するための特別な手順を加え，拡張した 2 つの探索法の性能を比較した．その結果，拡張した SSE について，拡張した GA よりも優れた性能が得られ，ハイブリッド手法において SSE の持つ速い収束特性が有利となる可能性を示した．

今後の課題として，より複雑なだまし問題を用いた評価や，現実的な応用問題への適用などがあげられる．

謝辞

末筆ながら，本研究の一部は文部省科学研究費（奨励研究 (A) 05750385）の補助によることを記し，謝意を表します．また，本研究に関連する話題について活発な議論をして頂きましたイリノイ大学アーバナシャンペイン校の Benjamin W.Wah 教授およびグループのメンバーの方々，および，本研究の機会を与えて下さいました学術情報センター猪瀬博所長，浅野正一郎教授ならび所内の方々に深く感謝いたします．

参考文献

[1] 相澤彰子: "スキーマ定理に関する一考察", 情報処理学会第 47 回全国大会, 7A-5.

[2] Aizawa, N.A. and Wah, B.W. : "Dynamic Control of Genetic Algorithms in a Noisy Environment," *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp.48-55 (1993).

[3] Bolc, L. and Cytowski, J.: "Search Methods for Artificial Intelligence," Academic Press, 1992.

[4] Eshelman, L. J. and Schaffer, J. D.: "Crossover's Niche," *proceedings of the 5th International Conference on Gentic Algorithms*, pp.9-14 (1993).

[5] Fitzpatrick, J. M. and Grefenstette, J. J. : "Genetic Algorithms in Noisy Environments," *Machine Learning*, 3 (2/3):101-120 (1988).

[6] Ghosh, B. K. and Sen, P. K.: "Handbook of Sequential Analysis," Marcel Dekker, Inc., 1991.

[7] Grefenstette, J. J.: "Optimization of Control Parameters for Genetic Algorithms," *IEEE transactions on Systems, Man, and Cybernetics*, Vol.SMC-16, No.1, pp.122-128 (1986).

[8] Grefenstette, J. J.: "Deception Considered Harmful," *Foundations of Genetic Algorithms 2*, pp.75-91 (1993).

[9] Goldberg, D. E., Deb, K. and Clark, J. H.: "Genetic Algorithms, Noise, and the Sizing of Populations," *IlliGAL* Technical Report, 91010, Univ.of Illinois at Urbana-Champaign (1991).

[10] Goldberg, D. E., Korb, B. and Deb, K. : "Messy Genetic Algorithms: Motivation, Analysis, and First Results," Complex Systems, vol.3, pp.493-530 (1989).

[11] Goldberg, D. E. : "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley (1989).

[12] Holland, J. H.: "Adaptation in Natural and Artificial Systems," the University of Michigan Press (1975).

[13] Kido, T., Kitano, H. and Nakanishi, M.: "A Hybrid Search for Genetic Algorithms: Combining Genetic Algorithms, TABU Search, and Simulated Annealing," *Proceedings of the Fifth International Conference on Genetic Algorithms*, p.641 (1993).

[14] 北野宏明編 : "遺伝的アルゴリズム", 産業図書 (1993).

[15] Manfoud, S. W. and Goldberg, D. E. : "A Genetic Algorithm for Parallel Simulated Annealing," *Proceedings of Parallel Problem Solving from Nature* (1992).

[16] 坂無英徳, 鈴木恵二, 嘉数侑昇: "遺伝的アルゴリズムにおける探索戦略の制御", 情報処理学会論文誌, Vol.34, No.4 (1993).

[17] Tsutsui, S. and Fujimoto, Y. : "Forking Genetic Algorithm with Blocking and Scrinking Modes (fGA)," *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp.206-213 (1993).

[18] Whitley, L. D.: "Fundamental Principles of Deception in Genetic Search," *Foundations of Genetic Algorithms*, pp.221-241 (1991).

[19] Wilson, S. W.: "GA-Easy Does Not Imply Steepest-Ascent Optimizable," *proceedings of the 4th International Conference on Gentic Algorithms*, pp.85-89 (1991).