特集 進化的アルゴリズムとファジィ理論

進化的アルゴリズムの方法論[†]

玉置 久*

1. はじめに

ここ数年の間に、進化的アルゴリズム (Evolutionary Algorithms: EA) あるいは進化的計算 (Evolutionary Computation: EC) という言葉をよく耳にするようになった。これらは、遺伝的アルゴリズム (Genetic Algorithms: GA)^{1),2)}, 進化的戦略 (Evolution Strategies: ES)³⁾ および進化的プログラミング (Evolutionary Programming: EP)⁴⁾ などに代表されるような、生物の集団遺伝・進化の過程を模倣して最適値を探索するための計算モデルあるいはアルゴリズムを総称するものである。

進化的アルゴリズムについては、これまでにても関連論文のサーベイを含めて総括的な解説がなされているが⁵⁻¹¹⁾, 本稿では、これらの解説を参考にしながらそれぞれのアルゴリズムの特徴を整理するとともに、最適化問題への応用を中心に進化的アルゴリズムの方法論を紹介しよう^{12,13)}。

2. 進化的アルゴリズム

進化的アルゴリズム (EA) では、問題解決のためのキーとして、生物の進化過程を模倣した計算モデルが用いられる。以下で紹介するように、これまでに種々の EA が提案されているが、これらのアルゴリズムに共通するのは、

- (1) 問題の解を個体 (individual) として表し、その集合 (個体群: population) を保持すること、

世代カウンタの初期化: $t = 0$

個体群 $\mathcal{P}(0)$ の初期化

while (終了条件 $\neq$ true) **do**

 組み換え: $\mathcal{P}'(t) = R(\mathcal{P}(t))$

 突然変異: $\mathcal{P}''(t) = M(\mathcal{P}'(t))$

 個体群 $\mathcal{P}''(t)$ の評価

 選択/淘汰/複製: $\mathcal{P}(t+1) = S(\mathcal{P}''(t) \cup \mathcal{P}(t))$

 世代カウンタの更新: $t = t + 1$

end

図1 進化的アルゴリズムの概要

$\mathcal{P}(t)$ は世代 t の個体群を表し、 R, M および S は、それぞれ組み換え、突然変異 および 選択/淘汰/複製の演算子を表す。

(2) 選択/淘汰 (selection), 複製 (reproduction), 組み換え (recombination), 突然変異 (mutation) などの遺伝演算子 (genetic operators) を適用して個体群を進化 (evolution) させること、

(3) 各個体には、環境 (environment) に対する適応度 (fitness) が与えられること、

(4) 選択/淘汰/複製によって、より高い適応度を持つ個体がより多く残され (exploitation), 一方、組み換えや突然変異により、個体に摂動を加えて多様な個体を生みだす (exploration) こと、

である。図1にEAの概略を示しておく。

以下では、代表的なEAについて、そのアルゴリズムを記述する。

3. 遺伝的アルゴリズム

遺伝的アルゴリズム (GA) は、EAの中で最もポピュラーなアルゴリズムであり、その起源は1960年代に遡る¹⁴⁾。当初は、Hollandによって、適応システム (adaptive system) という一般的な枠組みが構築されたが、その後、最適値探索アルゴリズム

[†] Evolutionary Computation and Optimization

* Hisashi TAMAKI

神戸大学工学部電気電子工学科

Dept. of Electrical and Electronics Engineering, Faculty of Engineering, Kobe University

ムとして DeJong によって計算機実験が行われ¹⁵⁾, 1989 年に Goldberg により体系的にまとめられた²⁾. これまでに, 種々の拡張・工夫が施された GA が数多く報告されているが, ここでは, 標準的 GA (canonical GA)⁵⁾を対象とする.

GA の特徴は,

- (1) 適当なコーディング方法により, 問題の解候補を記号列(遺伝子列: 個々の記号を遺伝子(gene)に対応させる)を用いて表現し, 解候補およびその遺伝子表現を, それぞれ表現型(phenotype)および遺伝子型(genotype)に対応させること,
- (2) 組み換えの一種として交叉(crossover)を用いること,
- (3) 突然変異は一様にランダムに行われること,
- (4) 適応度に基づいて確率的に選択が行われること,

などである. また, 後述の進化的戦略や進化的プログラミングと比較すると,

- (5) 離散空間で探索することを基本としているため, 組合せ最適化問題に多く適用されていること,

- (6) 交叉演算を重視すること,

といった特徴が挙げられる.

GA に関する理論的成果については, 多くの試みがなされているにもかかわらず決め手となるものはないというのが現状であろう. ここでは, GA の基本定理(fundamental theorem)あるいはスキーマ定理(schema theorem)と呼ばれているものを取り上げ²⁾, 簡単に紹介しておこう.

まず, 遺伝子列の集合(遺伝子空間の部分空間)を定めるための概念として, 部分的な遺伝子列をパターン化したものを定義し, これをスキーマ(schema または similarity template)と呼ぶ. 具体的には, 遺伝子を 2 進数で表現した場合の部分的なビット・パターンとして定義され, 記号{0, 1, *}を要素として表される(*はドントケア記号). 例えば, スキーマ

$$H_1 : 1 \quad * \quad 0 \quad * \quad 0 \quad 1 \quad *$$

は,

$$I_1 : 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0$$

$$I_2 : 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1$$

などの 8 つの遺伝子型(3 つの '*' に対するビットのすべての組合せ)に含まれる, すなわちこれらに共通するパターンである.

さらに, スキーマの次元(order)および定義長(defining length)を次のように定義する.

- ・次元: スキーマ H に含まれるドントケア記号 * 以外の記号の数 $o(H)$; 上の例では, $o(H_1) = 4$.
- ・定義長: スキーマ H に含まれるドントケア記号 * 以外の記号間の距離の最大値 $\delta(H)$; 上の例では, $\delta(H_1) = 5$.

このとき, 遺伝演算によるスキーマの増減を定量的に表したのが, 次の定理である.

基本定理(スキーマ定理):

$$\begin{aligned} m(H, t+1) &\geq m(H, t) \frac{f(H)}{\bar{f}(t)} \left\{ 1 - p_c \frac{\delta(H)}{l-1} \right\} (1 - p_m)^{o(H)} \\ &\approx m(H, t) \frac{f(H)}{\bar{f}(t)} \left\{ 1 - p_c \frac{\delta(H)}{l-1} - p_m o(H) \right\} \quad (1) \end{aligned}$$

ただし,

$m(H, t)$: 世代 t でスキーマ H を含む個体の数,

$f(H)$: スキーマ H を含む個体の適応度の平均値,

$\bar{f}(t)$: 世代 t での個体群の平均適応度,

である.

式(1)では, 交叉および突然変異によって破壊されるスキーマのみが考慮されており, これらの演算によって新たに生成されるスキーマに関する項が含まれていないことに注意しよう. したがって, この定理はあるスキーマに注目した場合に, その個数が個体群中でどのように変化するかについての下限値を与えているに過ぎない.

2.2 進化的戦略

進化的戦略(ES)は, 当初より多峰性の非線形最適化問題を解くことを目的に, Rechenberg と Schwefel によって 1960 年代に開発されたアルゴリズムである^{3,16,17)}. ES の特徴は,

(1) 扱う問題として n 次元実数空間上での最適化問題を想定していること、

(2) 正規分布に従った摂動を加える形で突然変異を導入しており、これを主な探索手段としていること、

(3) 選択は確定的に行われ、

- ・ (μ, λ) -ES: λ 個の子個体のうちで適応度の高いものから(目的関数値の良いものから)順に μ 個を選ぶ方法、および

- ・ $(\mu + \lambda)$ -ES: μ 個の親個体と λ 個の子個体を合わせたものの中から μ 個を選ぶ方法、があること、

である。

ES は、その選択の方法によって種々のバリエーションが考えられる⁵⁾。これまでに、

- ・ (1+1)-ES: 山登り法、
- ・ (1,1)-ES: ランダム探索法、
- ・ (1+ λ)-ES および (1, λ)-ES: 近傍探索法、
- ・ $(\mu+1)$ -ES: 連続世代型の多点探索法、
- ・ $(\mu+\lambda)$ -ES: エリート保存による多点探索法、
- ・ (μ, λ) -ES: 多点探索法、

などの ES が提案・応用されている。歴史的には、親個体数 μ および子個体数 λ をともに 1 とする (1+1)-ES から研究が始まり、複数の親個体を用いて組み換え演算を導入できる $(\mu+1)$ -ES を経て、 $(\mu+\lambda)$ -ES や (μ, λ) -ES へと発展してきた。

実数空間上での非線形関数の最適化問題に対して ES のような手法を用いることの利点は、

- ・ 最適化すべき目的関数が多数の局所的最適点を持つ多峰性の関数である、
- ・ 目的関数が実験やシミュレーションなどを通じて評価されるため、雑音などが重畳されたり、微係数を求めることができなかつたりする、

といった場合にも適用することが可能であるという適用範囲の広さにある。一方、最適化すべき目的関数が局所的最適解を持たない場合や、微係数を容易に計算できる場合には勾配法や Newton 法など、より効率の良いアルゴリズムが知られており^{9,18)}、これらの利用を検討すべきであろう。

以下、ES の最も基本的な構成法である (1+1)-ES を紹介しておこう。いま、 n 次元実数空間 $\mathbf{R}^n$ 上での目的関数 $f(x)$ の最小化問題

$$\min_x f(x) \quad (2)$$

を考える。制約条件については、ペナルティ関数などの形で目的関数 $f(x)$ に反映されているものとし、陽には考えない(後の 3.3 参照)。

(1+1)-ES では、現在の探索点 x に正規分布に従う摂動を突然変異として加えて新しい探索点 x' を生成することによって探索が進められる。以下では、決定変数ベクトル x の各成分 x_i ($i=1, \dots, n$) に対して独立に、平均 0、分散 σ^2 の正規分布 $N(0, \sigma^2)$ に従う乱数を摂動として加えることを考える。この場合、(1+1)-ES のアルゴリズムは以下のように記述される。

0° $\mathbf{P}(0) = \{(x, \sigma)\}$ を適当な方法で初期化し、
 $t=0$ とする。

1° $f(x)$ を評価する。

2° 1/5 規則に従い σ を更新して σ' を得る。

3° x の各成分に正規分布 $N(0, \sigma'^2)$ に従う乱数を加えて x' を得る。

4° $f(x')$ を評価する。

5° $f(x') \leq f(x)$ ならば $\mathbf{P}(t+1) = \{(x', \sigma')\}$ とし、そうでなければ $\mathbf{P}(t+1) = \mathbf{P}(t)$ とする。

6° $t=t+1$ とする。終了条件が満たされていれば終了。満たされていなければ 2° に戻る。

ここで、上のアルゴリズムの 2° における“1/5 規則”について触れておく。最適解 x^* への収束の速さ ψ を、

$$\psi = \mathbf{E}\{\|x^* - x(t)\| - \|x^* - x(t+1)\|\} \quad (3)$$

と定義し、この ψ を最大にする、すなわち

$$\left. \frac{d\psi}{d\sigma} \right|_{\sigma=\sigma^*} = 0 \quad (4)$$

を満たすような標準偏差 σ^* (“最適ステップサイズ”と呼ばれる)を考える。これまでにも、いくつかの問題例に対して最適なステップサイズ σ^* が求められており、例えば問題：

$$\min_x f(x) = \sum_{i=1}^n x_i^2 \quad (5)$$

に対しては、

$$\sigma^* \approx 1.224 \frac{\|x^* - x\|}{n} \quad (6)$$

となることが知られている^{8,9)}。

さらに、突然変異によって良い個体が生じる(“突然変異の成功(successful mutation)”と言う)確率:

$$p = \text{Prob}\{f(M(x)) \leq f(x)\} \quad (7)$$

を、式(6)の最適ステップサイズ σ^* に対して求めると、問題(5)に対しては、

$$p^* \approx 0.270 \quad (8)$$

となる。他の問題例についても同様に p^* を求めると、いずれも $1/5$ に近い値になることが確認されている。そこで、 σ を調整する経験則として、以下の規則が Schwefel により提案されている⁹⁾。

5分の1規則(1/5-success rule): 突然変異が成功する割合は $1/5$ とすべきである。もし、その割合が $1/5$ 以上であればステップサイズを上げ、 $1/5$ 以下であれば下げるべきである。□

この規則に従い、上の $(1+1)$ -ES における σ の具体的な更新手続きとして、以下のような手法が提案されている:

σ の更新手続き: n 回(すなわち、ベクトル x の次元数に等しい回数)の突然変異ごとに、過去 $10n$ 回の突然変異における成功した回数を調べ、これが $2n$ を下回れば係数 $c=0.85$ を σ に乗じ、上回れば σ を c で割る。□

2.3 進化的プログラミング

進化的プログラミング(EP)は、予測問題を解くための有限状態機械を進化させる枠組みとして、1960年代に L. Fogel らによって開発された^{4,19)}。その後 D. Fogel によって、2.2で紹介した ES の研究とは独立に実数関数の最適化問題への適用が行われ、ES に類似した手法として発展してきた。

EP は、ES とほとんど同じ手続きによって構成されるが、

- (1) 選択は確率的に行われること^注
- (2) 各個体には、問題の解候補(決定変数値) $x \in \mathbf{R}^n$ のみを持たせること、

という点にその違いがある。さらに、

- (3) ES は“個体の進化”の過程を模したものであるが、EP では“種の進化”をその手本としているので、組み換えのための演算は適用されない⁸⁾、

という特徴がある。

(1)に関しては、各個体 I を、個体群中からランダムに選ばれた q 個の個体との間で比較し、個体 I が優っている数をその個体 I の適応度とする。そして、この適応度の高いものから順に μ 個を選択することになる。このとき、 q を大きく選ぶと、ES における確定的な選択に似たものとなる⁵⁾。

また、(2)に関して、L. Fogel らの枠組みでは、突然変異は一様にランダムに行われることになっていたが、1990年代になって D. Fogel は、実数変数最適化問題を解くために EP を拡張し、正規分布に従う突然変異を導入するとともに、その分散 $v \in \mathbf{R}_+^n$ (すべての成分が正の n 次元の実数ベクトル)を個体表現の一部に加えた(“meta-EP”と呼ばれている)^{5,8)}。

ここで、3種類の EA(GA, ES および EP)の主な特徴を表1にまとめておく⁵⁾。なお、これら3種の方法は、生物の進化を手本として少しずつ異なる背景のもとに独立に開発されたものであるが、近年、相互の交流が深まり区別は意味を失いつつある。

また、EAの適用を考える場合、主たる探索方法として組み替え(交叉)演算を用いるか、あるいは突然変異演算を用いるかという選択肢があるが、これについては個々の適用対象に対する依存性が強く、一般的にどちらが良いとは言えない。適用対象に応じて効率的な演算子を工夫することが重要であろう。

2.4 その他のアルゴリズム

2.1~2.3では、EAの代表例として3種類のアルゴリズムを整理・紹介した。これら以外にも、EAの範疇に入れられるアルゴリズムとして、以

注: 2.2の表記法に従えば、“確率的な $(\mu + \mu)$ -ES”に相当する。

表 1 進化的アルゴリズムの特徴

	GA	ES	EP
個体表現	離散値表現	連続値表現	連続値表現
適応度	目的関数値をスケーリング	目的関数値をそのまま使用	目的関数値をスケーリング
突然変異	補助的探索法	主要な探索法	唯一の探索法
組み換え (交叉)	主要な探索法	補助的探索法	使わない
選択/淘汰/複製	確率的, 保存的	確定的, 消滅的	確率的, 消滅的

下のものが挙げられている⁷⁾.

分類システム(classifier system) : GA を応用したルール学習システムで, Holland らによって確立された^{2,20)}. ガス・パイプラインを制御するためのルール獲得などの応用がある. なお, この分類システムは強化学習(reinforcement learning)²¹⁾という枠組みに入るものであり, この分類システム以外にも Q 学習(Q -learning)²²⁾などいくつかの方法が提案されている^{21,23)}. また, GA でルールを学習(発見)する方法としては, 他にピッツバーク・アプローチ(Pitts approach)^{2,24)}と呼ばれるものもある(これに対して, 分類システムはミシガン・アプローチ(Michigan approach)とも呼ばれる).

遺伝的プログラミング(genetic programming) : 木構造の個体表現法を持つ GA で, Koza によって体系化された²⁵⁾. Lisp などのプログラミング言語の進化と親和性が高く, 関数合成や知識獲得などへの応用がある²⁵⁻²⁷⁾.

また, 進化的アルゴリズムに関連する最近の話題として, 共進化モデル^{28,29)}, 人為選択によるシミュレーション^{30,31)}, 多目的性を考慮した探索モデル³²⁻³⁴⁾などが注目されている.

3. 進化的アルゴリズムと最適化

進化的アルゴリズム(EA)を最適化問題に適用する場合について考える. なお, 以下では EA の中でも最も広範な応用が報告されている遺伝的アルゴリズム(GA)を取り上げ, この GA を組合せ最適化問題に適用する場合の枠組みを整理しておく¹³⁾.

3.1 組合せ最適化問題

組合せ最適化問題とは, 組合せ的・離散的な制約条件のもとである目的関数を最小化(あるいは最大化)する数理計画問題であり, 一般に次のように記述される.

$$\min_x f(x) \quad (9)$$

$$\text{subject to } x \in F \quad (10)$$

$$F \subseteq X \quad (11)$$

ここで, 基本空間 X およびその部分集合である可能領域(解空間) F は組合せ的・離散的なものである¹⁸⁾.

組合せ最適化では, 実数集合を対象とした連続性や微分の概念に基づく古典的な最適化手法を直接利用することはできない. したがって, 組合せ最適化問題の解法は連続変数の最適化手法と本質的に異なるものとなり, 一般に解を調べ上げるというアプローチとならざるを得ない. しかし, 解空間 F の要素(組合せ)の総数は有限ではあっても膨大な数にのぼることが多く, 実際に列挙・探索する範囲をいかに限定するかが重要となる.

3.2 遺伝的アルゴリズムによる組合せ最適化

2.1 でも記したように, GA では, まず問題の解候補を記号列(遺伝子列)からなる個体として表し, 実際の解候補を表現型, 記号列の個体を遺伝子型とする. そして, 個体に対して選択/複製, 交叉および突然変異といった遺伝演算子を繰り返し適用することにより, 適応度の高い個体(すなわち質の良い解候補)を逐次生成していくことになる.

ここで注意しなければならないのは, (少なくとも

も現時点では)形式的に GA を適用すればそれだけで効率的な最適値探索が実現されるような方法論が確立されておらず、対象とする最適化問題ごとに独自の工夫が必要であるという点である。これまでも種々の問題例に対してさまざまな工夫の例が報告されているが、現実的な最適化問題においては、特にその制約条件の取り扱いに工夫を要する場合が多い。そこで、以下では GA を構成する手順について、制約条件の取り扱い方法を中心に整理しておこう。

一般に、最適化問題に対して GA を構成する場合の手順は、以下のように記述できる。まず、対象となる問題のモデル化・定式化、すなわち、

- 1° 決定変数および制約条件の選定：問題の解空間 F の設定、
 - 2° 評価値の定量化：最小化あるいは最大化すべき目的関数 f の設定、
- をしなければならない。次に、GA の適用方法について、
- 3° 問題の解候補を遺伝子表現(コーディング)する方法の決定：GA による探索の対象となる空間 S の設定、
 - 4° 表現型への変換(デコーディング)方法の決定、
 - 5° 適応度の定量化：目的関数 f と適応度関数 g との対応規則を定める、
 - 6° 遺伝演算子の具体化：交叉、突然変異および選択演算の方法を定める、

を行わなければならない。図 2 に、この標準的な構成手順による探索の手続きを示す。さらに、図 3 に、そのときの探索空間 S と解空間 F の関係を示しておく。

GA による探索空間について見ると、問題の制約条件(式(10)および式(11))を満たす解空間全体を効率的に探索することが要求される。実際の工学的最適化においては、この(一般に複雑な)制約条件を満足させることが極めて重要であり、GA

図 3 探索空間と解空間

探索空間内の各個体は、一旦表現型(解表現)にデコーディングされ、その目的関数値に基づいて適応度が算出される。

図 2 遺伝的アルゴリズムによる探索手続き

図中の数字は、構成手順に対応する。

を構成する場合においても、まず制約条件をどのような形で取り扱うのかを決めることが肝要である。そこで、GA 構成における制約条件の取り扱い方法を分類・整理しておこう。

3.3 制約条件の取り扱い方法

制約条件を満たすための最も単純な方法としては、図 2 にも示すように、

(a) GA による探索空間 S と問題の解空間 F との対応が“1 対 1”になるように探索空間 S を定める、

というものが考えられる。この際、交叉や突然変異演算によって生成される新しい候補解も制約条件を満たす実行可能解となるようにしなければならない。これを実現する方法としては、

(a') 常に解空間 F に含まれる候補が生成されるように交叉演算子および突然変異演算子を定める、

(a'') 解空間 F に含まれる候補が生成されるまで交叉や突然変異を繰り返す、

などが考えられる。前者は、場合によって交叉・突然変異演算子の構成にかなりの工夫を要し、また、後者は計算効率の低下を覚悟しなければならない。

方法(a) (あるいは(a'))の実現が難しい場合の次の方法として、解空間 F を含むような領域(例えば基本空間 X)に対応するように探索空間 S を設定しておき、

(b) 探索空間 S 内の各点を、解空間 F 内の点(実行可能解)に“変換”する手続きを組み込む、という方法が考えられる(図 4)。すなわち、探索空間と解空間の関係を多対 1 の関係に緩めるのである。この場合、交叉演算子および突然変異演算子の設計にかなりの自由度が得られる。一方、探索空間が解空間に比べて広がるので、一般には探索効率の低下が予想される。

次に、制約条件を満足させる解の生成手続きを与えることも困難な場合には、

(c) 解空間 F に含まれない候補にはペナルティを課す、

という方法がある¹⁶。この場合、GA によって得られる解が実行可能解となることは一般には保証されず、ペナルティが少ない場合には、探索が制約を侵す解に収束することも起こり得る。このため、ペナルティの量を調整するなどの作業が必要になってくる。

ペナルティを与える方法の極端な場合として、制約条件を満たさない解候補は目的関数値を無限大(最大化問題の場合は無限小)とし、受理されることを禁止する方法もある。この方法は、制約条件を侵す解を生成しないという利点を持つ反面、制約条件が厳しい場合には、交叉や突然変異によって新しい探索空間内の点を生成することが困難になりやすい。

4. まとめ

本稿では、進化的アルゴリズム(EA)について、遺伝的アルゴリズム(GA)、進化戦略(ES)および進化プログラミング(EP)を中心にそのアルゴリズムを紹介するとともに、GA を最適化問題に適用する場合について、制約条件の取り扱い方法に留意しながらその標準的方法論を整理・解説した。本稿で解説した内容からもわかるように、現実的な問題に対して EA を適用する場合には、アルゴ

図 4 変換手続きを組み込んだGAの構成

注：ペナルティを課すことについては、問題の制約条件を満たさない度合いを定量化し、これを目的関数に含めるという形で定式化が行われているとの見方もある。

リズムを具体化する前段階として問題の構造・性質を十分に把握することが肝要である。言い換えると、EAを構成するための体系化された標準かつ万能な方法論が確立されていて、誰でもいつでもそれを機械的に適用すれば問題が解けるわけではない。この意味でも、本稿がEAを現実問題に応用する際の一助になれば幸いである。

最後に、紙面の都合上(と言うより著者の残学さゆえ)、EAとファジィ理論との関連等に触れられなかったことをお詫び申し上げたい。

参 考 文 献

- [1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan, 1975.
- [2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
- [3] H.-P. Schwefel, *Numerical Optimization of Computer Models*, Birkhäuser, 1977.
- [4] L. J. Fogel, A. J. Owens and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons, 1966.
- [5] T. Bäck and H.-P. Schwefel, An Overview of Evolutionary Algorithms for Parameter Optimization, *Evolutionary Computation*, 1, 1, 1/23, 1993.
- [6] W. M. Spears, K. A. DeJong, T. Bäck, D. B. Fogel, H. de Garis, An Overview of Evolutionary Computation, *Proc. of European Conf. on Machine Learning*, 442/459, 1993.
- [7] J. Heitkötter and D. Beasley (Eds.), *The Hitch-Hiker's Guide to Evolutionary Computation* (FAQ in 'comp.ai.genetic'), 1994.
- [8] D. B. Fogel, *Evolutionary Computation*, IEEE Press, 1995.
- [9] H.-P. Schwefel, *Evolution and Optimum Seeking*, John Wiley & Sons, 1995.
- [10] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford Univ. Press, 1996.
- [11] T. Bäck, D. B. Fogel and Z. Michalewicz, *Handbook of Evolutionary Computation*, Oxford Univ. Press, 1997.
- [12] 玉置・喜多・西川, 進化アルゴリズム研究の動向, 第6回自律分散システム・シンポジウム, 175/180, 1995.
- [13] 三宮・喜多・玉置・岩本, 遺伝アルゴリズムと最適化, 朝倉書店, 1998.
- [14] J. H. Holland, Outline for a Logical Theory of Adaptive Systems, *Journal of the Association for Computing Machinery*, 3, 297/314, 1962.
- [15] K. A. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral dissertation, Univ. of Michigan, 1975.
- [16] I. Rechenberg, Cybernetic Solution Path of an Experimental Problem, *Roy. Aircr. Establ., Libr. transl.*, 1122, 1965.
- [17] H.-P. Schwefel, *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik*, Diploma thesis, Technical Univ. of Berlin, 1965.
- [18] 西川・三宮・英木, 最適化, 岩波書店, 1982.
- [19] D. B. Fogel, *Evolving Artificial Intelligence*, Doctoral dissertation, Univ. of California, 1992.
- [20] J. H. Holland and J. S. Reitman, Cognitive Systems Based on Adaptive Algorithms, *Pattern Directed Inference Systems*, 313/329, Academic Press, 1978.
- [21] 小林編, 特集「強化学習」, 人工知能学会誌, 12, 6, 1997.
- [22] C. J. C. H. Watkins, *Learning from Delayed Rewards*, Ph.D thesis, Univ. of Cambridge, 1989.
- [23] L. P. Kaelbling, *Learning in Embedded Systems*, MIT Press, 1993.
- [24] S. E. Smith, *A Learning System based on Genetic Adaptive Algorithms*, Doctoral dissertation, Univ. of Pittsburgh, 1980.
- [25] J. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, MIT Press, 1992.
- [26] J. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.
- [27] 伊庭, 遺伝的プログラミング, 東京電機大学出版局, 1996.
- [28] W. D. Hills, Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure, *Artificial Life II*, 313/324, Addison Wesley, 1992.
- [29] K. Sims, Evolving 3D Morphology and Behavior by Competition, *Artificial Life IV*, 28/39, MIT Press, 1994.
- [30] R. Dawkins, *The Blind Watchmaker*, Longman, 1986.
- [31] K. Sims, Interactive Evolution for Computer Graphics, *Computer Graphics*, 25, 4, 319/328, 1991.
- [32] H. Tamaki, H. Kita and S. Kobayashi, Multi-Objective Optimization by Genetic Algorithms: A Review, *Proc. of 1996 IEEE Int. Conf. on Evolutionary Computation*, 517/522, 1996.
- [33] C. M. Fonseca and P. J. Fleming, Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms—Part I: A Unified Formulation, *IEEE Trans. on*

- Systems, Man and Cybernetics (Part A)*, **28**, 1, 26/37, 1998.
- [34] C. M. Fonseca and P. J. Fleming, Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms—Part II: Application Example, *IEEE Trans. on Systems, Man and Cybernetics (Part A)*, **28**, 1, 38/47, 1998.

[問い合わせ先]

〒657-8501

神戸市灘区六甲台町1-1

神戸大学工学部電気電子工学科

玉置 久

TEL : 078-803-1063

FAX : 078-803-1063

E-mail : tamaki@eedept.kobe-u.ac.jp

(1998年5月7日 受付)

著者紹介

玉置 久 (たまき ひさし)

神戸大学工学部電気電子工学科

1985年 京都大学工学部電気工学科卒業, 1990年 京都大学大学院工学研究科博士課程研究指導認定退学(電気工学専攻), 同年 京都大学工学部電気工学第二学科助手, 1995年 神戸大学工学部電気電子工学科講師, 現在に至る。博士(工学), スケジューリング問題のモデル化と解法, 進化型計算法の理論と応用に関する研究に従事。