自己組織化マップを再生に用いた
遺伝的アルゴリズムに関する研究

久保田　良輔

本稿で使用する記号の一覧表

A	タンクの断面積 $[m^2]$
$\mathcal{B}$	バイナリ空間
b_i	バイナリ個体における i 番目の要素
BMU	勝者ユニット（Best Matching Unit）の番号
$\boldsymbol{C}^R$	実数値遺伝的アルゴリズムにおける子個体
c_{ji}^R	実数値遺伝的アルゴリズムにおける j 番目の子個体の i 番目の要素
d_j	j 番目の競合層ユニットと勝者ユニット間の距離
e^{DO}	タンク系における立ち下がりオーバーシュートの行き過ぎ量
e^{UO}	タンク系における立ち上がりオーバーシュートの行き過ぎ量
e^{DE}	タンク系における立ち下がりステップ応答の定常誤差
e^{UE}	タンク系における立ち上がりステップ応答の定常誤差
e	タンク系における目標水位との誤差
$f_{\boldsymbol{I}}$	個体 $\boldsymbol{I}$ の適合度
g	重力加速度 $[m/s^2]$
H	自己組織化マップの近傍関数
h，h^t	水位の高さと目標値 $[m]$
$\boldsymbol{I}$	遺伝的アルゴリズムにおける個体
$\boldsymbol{I}^B$	バイナリ遺伝的アルゴリズムにおける個体
$\boldsymbol{I}^R$	実数値遺伝的アルゴリズムにおける個体
i	個体，入力ベクトルの要素の番号
j	個体，競合層ユニットの番号
K，k	入力ベクトル数と番号
K_P	PID コントローラにおける比例要素のゲイン
k_1〜k_4	ルンゲ・クッタ法のパラメータ
L	ナップサックの重量の制限
M	個体，入力ベクトルおよび結合重みベクトルの要素数
N	集団の個体数，競合層ユニット数
$NC_{BMU,SOM}$	自己組織化マップにおける近傍係数
n	ルンゲ・クッタ法における刻み幅
ND	正規分布
$\boldsymbol{P}_j^R$	実数値遺伝的アルゴリズムにおける親個体
P_c	交叉率
P_m	突然変異率
p_j	ルーレット選択における j 番目の個体の選択確率
p_{ji}^R	実数値遺伝的アルゴリズムにおける j 番目の子個体の i 番目の要素
q，q_{max}	タンクに流れ込む 1 秒あたりの水量とその最大流量 $[m^3/s]$
r_i	実数値個体における i 番目の要素
$\mathcal{R}$	実数値空間

S	タンクにおける放出部の断面積 $[m^2]$
s_i	入力ベクトルの i 番目の要素の重要度
T_I	PID コントローラにおける積分要素のゲイン
T_D	PID コントローラにおける微分要素のゲイン
T	PID コントローラの状態を算出するステップ
$t,\ t_{max}$	学習およびタンク系の状態を算出するステップとその最終値
UD	一様分布
u_k	0-1 ナップサック問題における k 番目の荷物の重量
v_k	0-1 ナップサック問題における k 番目の荷物の価値
$\boldsymbol{w}_j^B$	j 番目の競合層ユニットと入力層ユニット間のバイナリ結合重みベクトル
w_{ji}^B	j 番目のバイナリ結合重みベクトルにおける i 番目の要素
$\boldsymbol{w}_j^R$	j 番目の競合層ユニットと入力層ユニット間の実数値結合重みベクトル
w_{ji}^R	j 番目の実数値結合重みベクトルにおける i 番目の要素
$\boldsymbol{x}^B$	バイナリ入力ベクトル
x_{ki}^B	k 番目のバイナリ入力ベクトルにおける i 番目の要素
$\boldsymbol{x}^R$	実数値入力ベクトル
x_{ki}^R	k 番目の実数値入力ベクトルにおける i 番目の要素
α	BLX-α 交叉における子個体生成領域に関するパラメータ
α_S	バイナリ自己組織化マップを用いた再生手法における学習に関するパラメータ
α_{BSOM}	バイナリ自己組織化マップの学習係数
α_{RSOM}	実数値自己組織化マップの学習係数
δ_i	BLX-α 交叉における親個体間の対応する遺伝子の距離

目 次

第1章　序論

本論文は，自己組織化マップの分布近似能力を用いた，遺伝的アルゴリズムの新しい再生手法について述べたものである．

遺伝的アルゴリズム [1] は，生物の進化と選択淘汰の過程を模擬した確率的探索アルゴリズムであり，J. Holland により 1962 年に提案された．遺伝的アルゴリズムは，進化的計算法 [2]-[4] のひとつである．進化的計算法とは，生物の進化過程を利用した計算法の総称であり，これらの計算法には，遺伝的アルゴリズムの他に，遺伝的プログラミング [5][6] や，進化戦略 [7][8] 等がある．これらの計算法の中で，遺伝的アルゴリズムは，そのアルゴリズムの簡便さと，応用の幅の広さから，多くの分野で用いられおり，その有効性が示されている．遺伝的アルゴリズムでは，問題の解を個体として表し，これらの集団に種々の遺伝的操作を世代ごとに繰り返し適用することにより，与えられた問題に対する最適な解を探索していく．各個体は，探索すべき解の表現方法に応じて，バイナリベクトルと実数値ベクトルによって表される．また，遺伝的アルゴリズムは，個体の表現方法の違いに応じて，バイナリ遺伝的アルゴリズム [9]-[12] と実数値遺伝的アルゴリズム [13] に大別される．一般的に，バイナリ遺伝的アルゴリズムは組み合わせ最適化問題に [14]-[17]，実数値遺伝的アルゴリズムは連続変数最適化問題に適用される [18]-[21].

各遺伝的アルゴリズムの遺伝的操作には，交叉・突然変異・再生があり，それぞれ，局所的な探索・大域的な探索・適合度に基づく選択淘汰という役割をもつ．これらの操作の中で，特に再生は次世代の探索の基点を決定するうえで非常に重要である．しかしながら，ルーレット選択に代表される従来の再生手法では，現世代の個体を次世代にコピーすることにより次世代の探索基点を決定するので，個体集団の多様性が減少する．多様性の減少は，局所解への収束や交叉効率の低下を引き起こすので，探索効率悪化の原因となる [22][23]．突然変異の割合を高くすることで，多様性の減少を抑えることは可能である．しかしながら，探索のランダム性が高くなってしまうので，良質な個体が破壊されてしまう可能性も高くなり，良質な個体を次世代に継承するにはエリート保存戦略 [11] に基づく手法が必要である [24][25]．また，適合度の分布に基づく交叉を行うことで，多様性を維持する手法が提案されている [23]．しかしながら，再生手法そのものによる多様性の減少は議論されていない．したがって，効率のよい探索を実現するためには，ランダム性を抑え，かつ高い多様性を維持することが可能な再生手法が必要である．

一方，自己組織化マップ [26]-[29] は，教師信号なしで学習を行うニューラルネットワークのひとつであり，T. Kohonen により 1982 年に提案された．自己組織化マップは，自身がもつ結合重みベクトルを徐々に更新していくことにより，提示される入力ベクトル集合の分布を近似するという特長をもち，この特長を利用して，種々のパターン認識・識別や量子化アルゴリズム等に応用されている [30]-[37]．また，一般的な自己組織化マップでは，入力ベクトルとして実数値ベクトルが扱われるのに対して，バイナリベクトルを扱うバイナリ自己組織化マップ [38][39] も 2002 年に提案されており，バイナリ入力ベクトル集合の分布を近似することも可能である．本論文では，バイナリ遺伝的アルゴリズム，実数値遺伝的アルゴリズムとの対応関係を明確にするために，実数値入力ベクトルを扱う自己組織化マップを実数値自己組織化マップと明記し，バイナリ自己組織化マップと実数値自己組織化マップを総称して自己組織化マップと呼ぶことにする．

本研究では，遺伝的アルゴリズムにおける個体集団の多様性減少によって引き起こされる探索効率悪化という問題点を解決するために，自己組織化マップの分布近似能力を用いた再生手法を提案する．提案する再生手法では，入力ベクトル集合を現世代の個体集団，結合重みベクトル集合を次世代の個体集団とする．すなわち，次世代の個体集団は，自己組織化マップの学習によって決定される．提案する再生手法に用いる自己組織化マップの学習では，結合重みベクトルの更新則が従来の入力ベクトルの分布を近似するものとは異なる．本研究では，適合度に基づく新しい更新則を導入することにより，高い適合度をもつ個体に類似した性質をもつ個体を生成することが可能となる．また，結合重みベクトルを段階的に更新していくことにより，完全に同一の性質をもつ個体の生成を抑え，個体集団の多様性を維持することが可能である．さらに，適合度に基づく結合重みベクトル更新則は，入力ベクトルがバイナリベクトル，実数値ベクトルにかかわらず適用可能である．したがって，提案する再生手法は，バイナリ自己組織化マップと実数値自己組織化マップを使い分けることによって，バイナリ遺伝的アルゴリズム，実数値遺伝的アルゴリズム双方に用いることが可能である．さらに，提案する再生手法は，これまでに提案されているエリート保存戦略や適合度の分布に基づく交叉法と併用することによって，さらに効率のよい探索が可能になる．この考え方は，様々な自己組織化マップを再生操作に用いることで，遺伝的アルゴリズムのみならず，遺伝的プログラミングや，進化戦略等へ応用することも可能である．

本論文は，図 1.1 に示すように 4 つの章からなる．

第 1 章は序論である．

第 2 章では，一般的なバイナリ遺伝的アルゴリズムと実数値遺伝的アルゴリズムの特長と問題点について述べる．また，自己組織化マップの構造と特長について述べる．

第 3 章では，自己組織化マップを用いた再生手法を提案し，遺伝的アルゴリズムに適用する．本章では，まず，実数値自己組織化マップを用いた再生手法を提案し，実数値遺伝的アルゴリズム

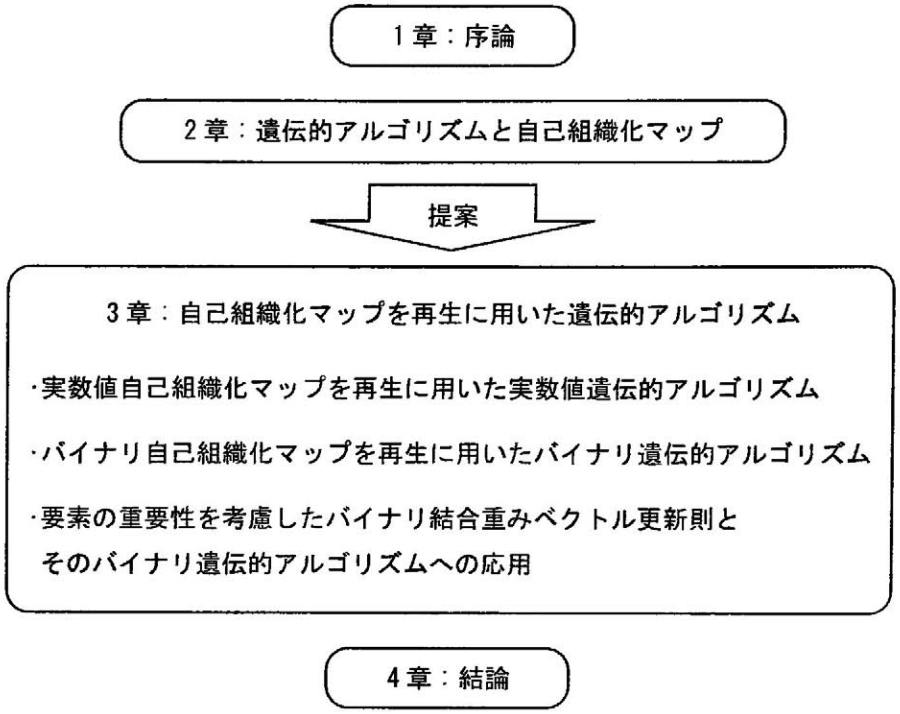


図 1.1: 本論文の構成.

に適用する．提案する再生手法では，入力ベクトルが実数値ベクトルである場合，適合度に基づく結合重みベクトル更新則を導入する．適合度に基づく段階的な更新は，各世代において完全に同一な性質をもつ複数の個体が生成されることを抑制する．これにより，個体集団の多様性を維持し，高い適合度をもつ個体と類似した性質をもつ多くの個体を生成することが可能である．提案手法を実数値遺伝的アルゴリズムに適用し，3 つの連続変数最適化問題を解く．これらの連続変数最適化問題は，DeJong によって提案されたベンチマーク問題である．具体的には，単峰性関数への局所的探索能力，ノイズを含む単峰性関数への局所的探索能力，多峰関数への大域的探索能力について従来の再生手法と比較することによって，提案する再生手法の連続変数最適化問題に対する有効性を示す．また，実問題への応用として，提案する再生手法を用いて，タンク系の水位制御器を設計する．

次に，バイナリ自己組織化マップを用いた再生手法を提案し，バイナリ遺伝的アルゴリズムに適用する．バイナリ自己組織化マップを用いた再生手法では，扱うベクトルがバイナリで表現されているので，それに応じて結合重みベクトルの更新則も異なる．しかしながら，更新則の概念は，実数値の場合と同様である．提案する再生手法を用いたバイナリ遺伝的アルゴリズムにより，組み合わせ最適化のベンチマーク問題である 0-1 ナップサック問題を解く．具体的には，バイナリ

探索空間での探索能力について，従来の再生手法と比較することによって，提案する再生手法の組み合わせ最適化問題に対する有効性を示す．

さらに，ベクトルの要素の重要性に着目したバイナリ結合重みベクトル更新則について述べる．従来のバイナリ結合重みベクトル更新則 [43][44] では，更新する要素の順序が考慮されていないので，その結果，探索が非効率になるという問題がある [45]．この問題点を解決するために，提案するバイナリ結合重みベクトル更新則では，個体集団がもつ適合度に応じて各要素の更新順序を決定する．組み合わせ最適化のベンチマーク問題である 0-1 ナップサック問題を解き，探索性能について，従来の結合重みベクトル更新則を用いたバイナリ自己組織化マップを再生に用いる場合と比較することで，提案手法の有効性を示す．

第 4 章は結論である．

第2章　遺伝的アルゴリズムと自己組織化マップ

2.1　はじめに

遺伝的アルゴリズムは，生物の進化と選択淘汰の過程を模擬した確率的探索アルゴリズムであり，J. Holland により 1962 年に提案された [1]，[9]．遺伝的アルゴリズムでは，問題の解を個体として表し，これらの集団に種々の遺伝的操作を繰り返すことで，世代ごとに集団が進化していく．遺伝的アルゴリズムは，以下の特長をもつ．

(1) 遺伝的アルゴリズムは，集団による多点同時探索を行う．

(2) 遺伝的アルゴリズムでは，問題に対する充足度（適合度）のみを用い，先見的な補助知識を必要としない．

(3) 遺伝的アルゴリズムにおける探索点の遷移は，確率論のみに基づき，決定論的なルールを必要としない．

遺伝的アルゴリズムの遺伝的操作には，交叉，突然変異，再生がある．これらの操作において，再生は次世代の探索基点を決定するうえで非常に重要である．しかしながら，従来の再生手法では，個体集団の多様性が損なわれるので，探索の効率が悪くなるという問題がある [22][23]．この問題は，次世代の個体集団を生成する際に，適合度に応じて現世代の個体を次世代へ“コピー”することに起因する．

一方，自己組織化マップは，教師信号なしで学習を行うニューラルネットワークのひとつであり，T. Kohonen により 1982 年に提案された [26]-[28]．自己組織化マップは，自身がもつ参照ベクトルを徐々に更新していくことにより，提示される入力ベクトル集合の分布を近似することが可能である．

本章では，まず，一般的な遺伝的アルゴリズムの概要について説明し，従来の再生手法の問題点を挙げる．次に，自己組織化マップの概要を説明する．

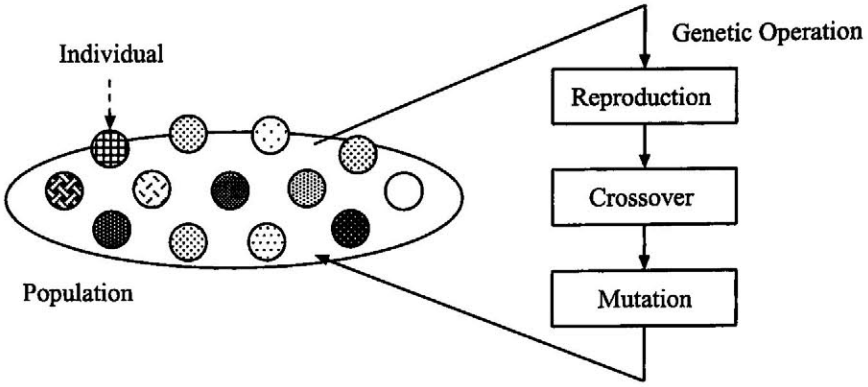


図 2.1: 遺伝的アルゴリズムにおける探索の概要.

2.2 遺伝的アルゴリズム

遺伝的アルゴリズムの概要を図 2.1 に示す．遺伝的アルゴリズムでは，問題の解を個体（Individual）として表現し，これらの集団（Population）に対して，各世代において種々の遺伝的操作（Genetic Operation）を繰り返し適用することによって探索が進行する．遺伝的操作には，再生（Reproduction），交叉（Crossover），突然変異（Mutation）がある．これらの遺伝的操作の特長を以下に示す．

再生： 現世代の個体集団と各個体がもつ適合度をもとに，次世代に残す個体集団を選択する．高い適合度をもつ個体は，高い確率で選択されるので増殖していくのに対して，低い適合度をもつ個体は，ほとんど選択されないので淘汰されていく．再生は，次世代の探索の基点を決定するという役目をもつ．

交叉： 複数の個体内の遺伝子列の一部を組み替えて新たな個体を生成する操作である．一般的な交叉法では，交叉する 2 つの親個体はランダムに個体集団から選ばれ，2 つの子個体が生成される．子個体は，親個体の性質を継承するので，局所的な探索に効果を発揮する．また，交叉は交叉率（Crossover rate）P_c と呼ばれる確率にしたがって行われる．

突然変異： 個体集団の中からランダムに個体を選択し，個体がもつ情報をランダムに置き換える．突然変異は，突然変異率（Mutation rate）P_m と呼ばれる確率にしたがって行われ，交叉では生成不可能な個体を生成することが可能であるので，大域的な探索に効果を発揮する．しかしながら，突然変異率が高い場合，良質な個体を破壊してしまう可能性が高くなり，探索が収束しない場合がある．

これらの特長を組み合わせることで，遺伝的アルゴリズムは，大域的かつ局所的な探索を行うこ

とが可能となる．遺伝的アルゴリズムの探索は，以下のアルゴリズムにしたがって行われる．

Step G-0: 乱数を用いて個体集団の初期化を行い，各個体の適合度を算出する．

Step G-1: 集団から，各個体の適合度をもとに，再生によって次世代に残す個体を決定する．

Step G-2: 集団から親個体を選出し，交叉によって子個体を生成する．

Step G-3: 集団から個体を選出し，突然変異によって個体を生成する．

Step G-4: 集団内の各個体の適合度を算出し，終了条件を満たしていれば終了する．終了条件を満たしていない場合は，Step G-1 に戻る．

終了条件は，以下に示すいずれかの条件によって決定される．

- 個体集団内の最高適合度が規定値を超えた場合．
- 個体集団の平均適合度が規定値を超えた場合．
- 世代数が規定値を超えた場合．

遺伝的アルゴリズムでは，各個体は数値ベクトルによって表される．数値ベクトルは，解く最適化問題の種類に応じて，バイナリベクトルもしくは実数値ベクトルによって表現される．遺伝的アルゴリズムでは，アルゴリズムの概念は同じであるものの，扱うベクトルの種類に応じて，異なるアルゴリズムが用いられる．バイナリベクトル，実数値ベクトルを個体として扱う遺伝的アルゴリズムをそれぞれ，バイナリ遺伝的アルゴリズム，実数値遺伝的アルゴリズムと呼ぶ．一般的に，バイナリ遺伝的アルゴリズムは，0-1 ナップサック問題や巡回セールスマン問題等に代表される組み合わせ最適化問題に用いられ [14]-[17]，実数値遺伝的アルゴリズムは，コントローラやモデルのパラメータチューニング等に代表される連続変数最適化問題に用いられる [18]-[21]．本論文では，バイナリ遺伝的アルゴリズムと実数値遺伝的アルゴリズム双方を指す場合，これらを総称して単に遺伝的アルゴリズムと呼ぶ．

各遺伝的アルゴリズムにおける遺伝的操作は，ベクトルの表現方法によりその実現方法が異なる．しかしながら，各操作がもつ特長は，ベクトルの表現方法によらず同じである．本節では，バイナリおよび実数値遺伝的アルゴリズムについて，各遺伝的操作の方法を説明する．

2.2.1 バイナリ遺伝的アルゴリズムでの個体表現と遺伝的操作

バイナリ遺伝的アルゴリズムで扱われる個体を図 2.2 に示す．個体（Individual）は，複数の遺伝子（Gene）の列で構成されており，実世界での染色体（Chromosome）に相当する．バイナリ

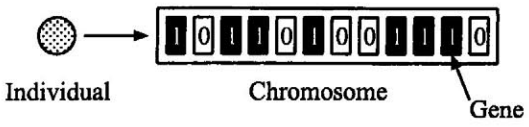


図 2.2: バイナリ遺伝的アルゴリズムにおける個体.

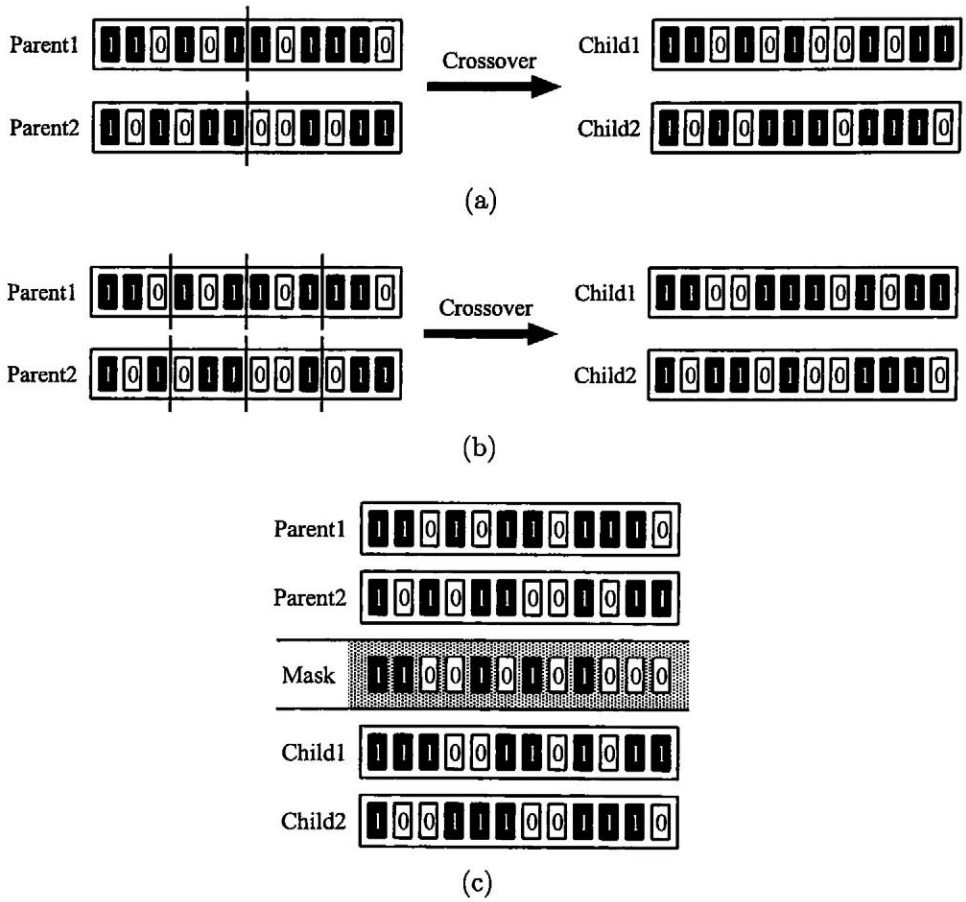


図 2.3: バイナリ遺伝的アルゴリズムの交叉. (a) 一点交叉. (b) 多点交叉. (c) 一様交叉.

遺伝的アルゴリズムでは，図 2.2 に示すように，各遺伝子が ‘0’ と ‘1’ のバイナリ情報で表現されている．バイナリ遺伝的アルゴリズムで用いる個体 $\boldsymbol{I}^B$ は，次式によって表される．

$$\boldsymbol{I}^B = (b_1, \cdots, b_i, \cdots, b_M), \quad b_i \in \mathcal{B} \tag{2.1}$$

ここで，b_i は，個体における i 番目の遺伝子を表し，バイナリ値をとる．

バイナリ遺伝的アルゴリズムにおいて一般的に用いられる交叉法を，図 2.3 に示す．交叉法に

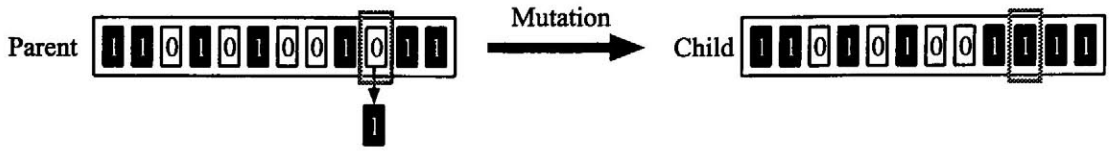


図 2.4: バイナリ遺伝的アルゴリズムにおける突然変異.

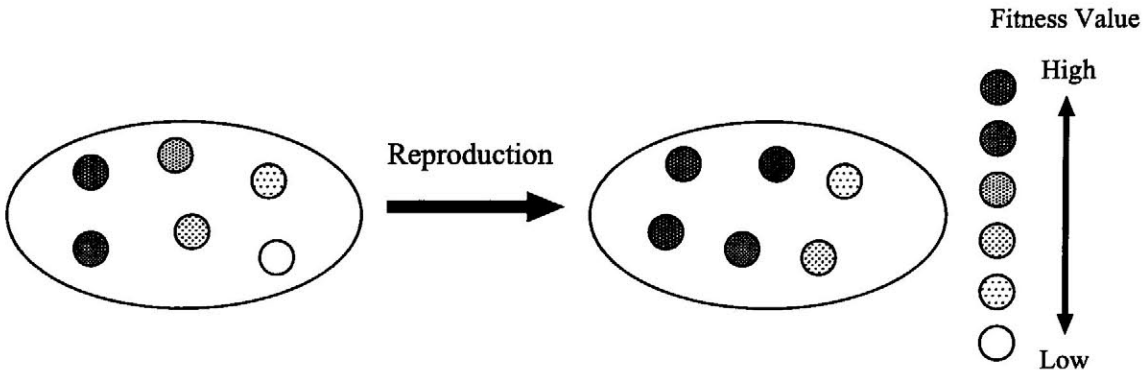


図 2.5: 遺伝的アルゴリズムにおける再生.

は，一点交叉（Onepoint crossover）や多点交叉（Multi-points crossover），一様交叉（Uniform crossover）があり，交叉点の数や位置が異なる．一点交叉や多点交叉では，まず，親個体（Parent）を選択する．次に，選択した親個体同士の部分的な遺伝子列を交叉点を基準にして入れ替えることによって子個体（Child）を生成する．一様交叉では，他の交叉方法と同様の処理で選出した親個体に対して，あらかじめ用意しておいたマスク（Mask）によって，入れ替える遺伝子を決定し，交叉を実現している．

突然変異では，選択された個体に対して，図 2.4 に示すように，一つ，もしくは複数の遺伝子を対立する遺伝子に置き換えることによって新しい個体を生成する．

再生では，算出された各個体の適合度をもとに，個体集団内から次世代に残す個体を選択する．図 2.5 に再生の概要を示す．再生操作によって，適合度が高い個体は増殖され，低い個体は淘汰されていく．再生手法には，ルーレット選択 (Roulette Wheel Selection) やトーナメント選択 (Tounament Selection) 等が一般的によく用いられる．ルーレット選択では，全ての個体に対して，適合度から算出した選択確率に基づいて，次世代に残す個体を決定する．一方，トーナメント選択では，複数の個体を個体集団からランダムに選び，これらのうちで最大適合度をもつ個体を次世代に残す．これらの再生手法は，その実現方法は異なるものの，概念は同じである．ルーレット選択のアル

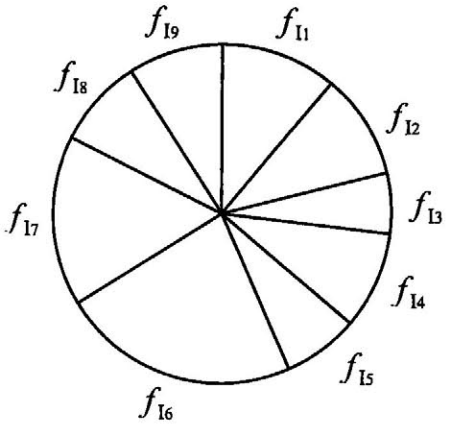


図 2.6: ルーレット選択.

ゴリズムを以下に示す.

Step R-0: 個体集団内の j 番目の個体 $\boldsymbol{I}_j$ † が選択される確率 p_j を次式により算出する.

$$p_j = \frac{f_{\boldsymbol{I}_j}}{\sum_{k=1}^{N} f_{\boldsymbol{I}_k}} \tag{2.2}$$

ここで, $f_{\boldsymbol{I}_j}$ は, 個体 $\boldsymbol{I}_j$ の適合度である. (2.2) 式は, 図 2.6 のように, 各個体の適合度に基づいて選択確率を決定することを意味する.

Step R-1: Step R-0 で算出した確率をもとに, 個体集団から 1 つの個体を選択する.

Step R-2: Step R-1 を規定回数繰り返す. 一般的に, 規定回数には集団内の全個体数を用いる.

また, 現世代の個体集団内で最大適合度をもつ個体を無条件で次世代に残すというエリート保存選択も提案されており, ルーレット選択等と併用することで, 良質な個体が破壊されるのを防ぐことが可能である.

2.2.2 実数値遺伝的アルゴリズムでの個体表現と遺伝的操作

実数値遺伝的アルゴリズムで扱われる個体の例を図 2.7 に示す. 実数値遺伝的アルゴリズムでは, 各遺伝子が実数値で表現されている. 実数値遺伝的アルゴリズムで用いる個体 $\boldsymbol{I}^R$ は, 次式に

† 本論文では, バイナリで表現される個体を $\boldsymbol{I}^B$, 実数値で表現される個体を $\boldsymbol{I}^R$ と表記するが, 各遺伝的アルゴリズムにおいて個体の表現方法に関係なく, 同じ演算を行う場合は添字 B と R を省略する.

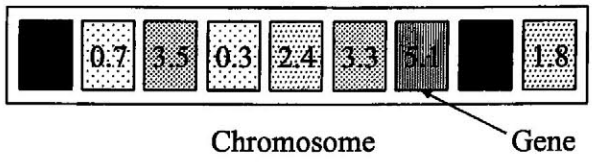


図 2.7: 実数値遺伝的アルゴリズムにおける個体.

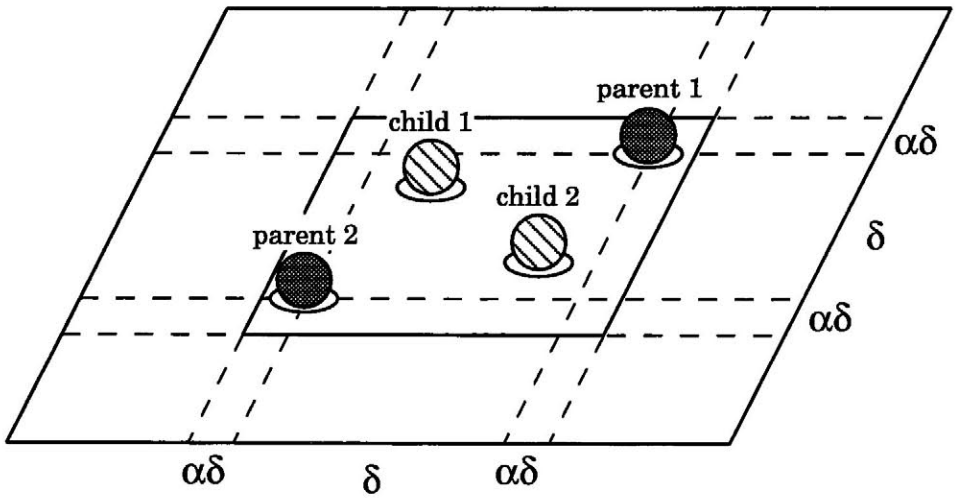


図 2.8: BLX-α 交叉.

よって表される.

$$\boldsymbol{I}^R = (r_1, \cdots, r_i, \cdots, r_M), \quad r_i \in \mathcal{R} \tag{2.3}$$

ここで，r_i は，個体における i 番目の遺伝子を表し，実数値をとる.

実数値遺伝的アルゴリズムでは，交叉と突然変異の方法がバイナリ遺伝的アルゴリズムの場合と異なる．実数値遺伝的アルゴリズムの交叉では，選択された親個体によって決定された領域から子個体を生成する．交叉法には，一般的によく用いられる BLX-α 交叉や単峰性正規分布交叉（Unimodal normal distribution crossover）がある．BLX-α 交叉における子個体生成の例を図 2.8 に示す．BLX-α 交叉では，選択された親個体に対して，親個体を表す実数ベクトルにおける各変数の区間 δ_i を両側に $\alpha\delta_i$ だけ拡張した区間から一様乱数に従ってランダムに子個体を生成する．すなわち，親個体の周辺の各辺が軸に平行な超直方体の領域が子個体の生成領域となる．子個体 $\boldsymbol{C}^R = (c_1^R, \cdots, c_i^R, \cdots, c_M^R)$ は，2 つの親個体 $\boldsymbol{P}_1^R = (p_{11}^R, \cdots, p_{1i}^R, \cdots, p_{1M}^R)$,

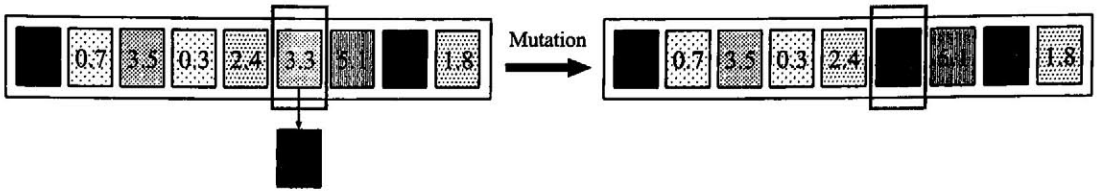


図 2.9: 一様突然変異.

$\boldsymbol{P}_2^R = (p_{21}^R, \cdots, p_{2i}^R, \cdots, p_{2M}^R)$ から次式により生成される.

$$c_i^R = U(\min(p_{1i}^R, p_{2i}^R) - \alpha\delta_i, \max(p_{1i}^R, p_{2i}^R) + \alpha\delta_i) \tag{2.4}$$

$$\delta_i = |p_{1i}^R - p_{2i}^R| \tag{2.5}$$

ここで，$\max(x, y)$，$\min(x, y)$，$U(x, y)$ は，それぞれ，x，y の最大値，最小値，区間 $[x, y]$ の一様乱数を意味する.

突然変異では，選択された個体に対して，一つ，もしくは複数の遺伝子をランダムな遺伝子に置き換える．突然変異の方法には，置き換える際に，実行可能領域内に一様乱数によって新しい実数値を発生させる一様突然変異と，実行可能領域の境界上に一様乱数によって新しい実数値を発生させる境界突然変異がある.

再生では，個体の表現方法によらず，適合度のみによって選択確率が算出されるので，バイナリ遺伝的アルゴリズムと同様にルーレット選択やトーナメント選択が用いられる.

2.3 従来の再生手法の問題点

ルーレット選択に代表される従来の再生手法では，現世代の個体をコピーして次世代に残すので，個体の多様性が喪失する．多様性の喪失は，適合度が高い個体ほど，同じ遺伝子列をもつ個体が複数回選択されることに起因し，世代が進むごとに探索点が著しく減少していく．探索点が減少すると，同じ遺伝子列を持つ親個体同士で交叉が行われる確率が高くなるので，交叉の性能にも悪影響を与える．また，バイナリ遺伝的アルゴリズム，実数値遺伝的アルゴリズム双方において同じ再生手法を用いているので，この問題は各遺伝的アルゴリズムに起こる．図 2.10 に，実数値遺伝的アルゴリズムにおける再生前後の個体配置を示す†．各軸は，遺伝子の値を表す．図 2.10(a) は，再生前の個体集団の配置である．各点は，探索点であり，一つの個体を表す．図 2.10(b) は，従来手法による再生後の個体集団の配置である．図 2.10(b) では，探索点が減少しているので，個

†図 2.10 では，個体が実数値ベクトルで表現されている場合について示しているが，このような状況は個体がバイナリベクトルで表現されている場合にも起こる.

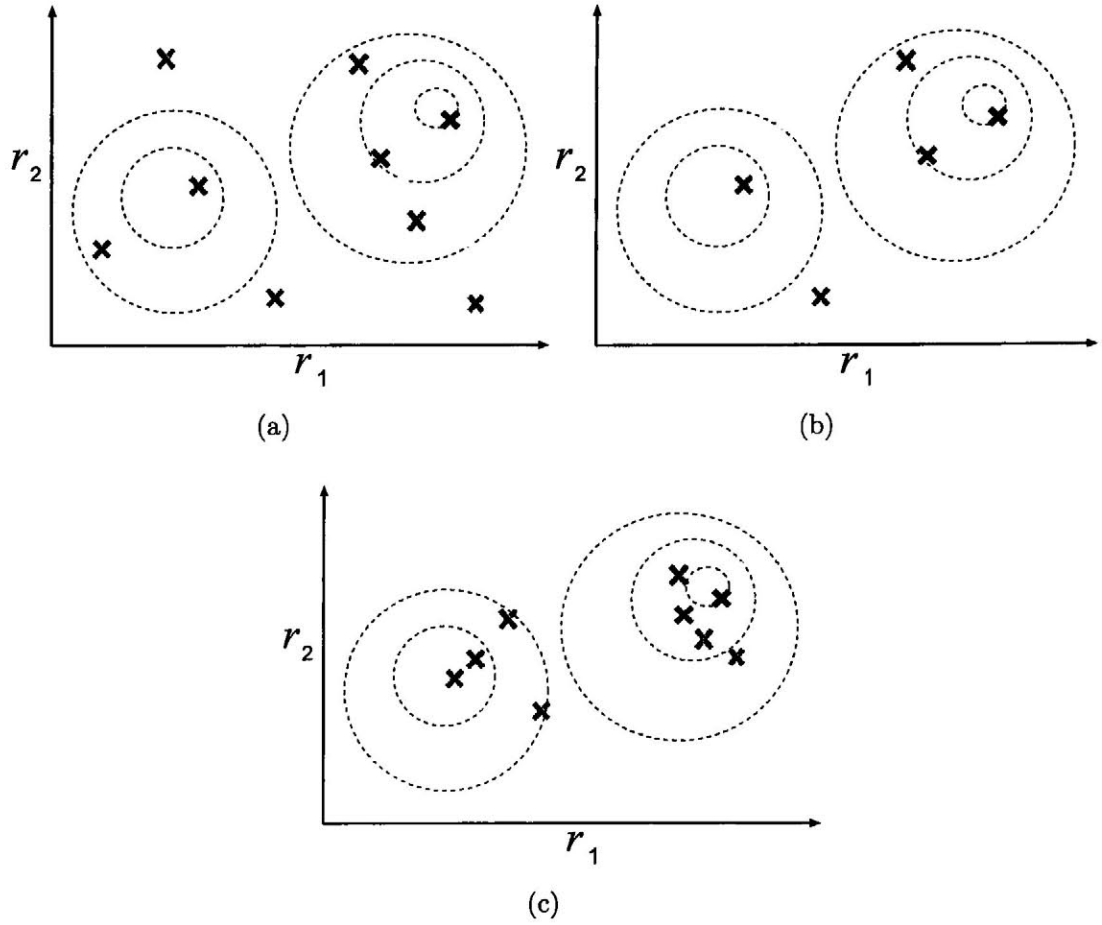


図 2.10: 従来の再生と好ましい再生. (a) 再生前の個体集団. (b) 従来手法による再生後の個体集団. (c) 好ましい再生後の個体集団. 各軸と円は，それぞれ個体の要素と適合度の等高線を表す. 円の半径が小さいほど，その領域の適合度は高い.

体数が減少しているように見える. しかし実際は，複数の個体が同じ探索点に配置されているにすぎず，個体数は減少していない. 突然変異を起こす確率を上げることで，多様性を維持することは可能である. しかしながら，ランダム性が高い多様性は，良質な個体を破壊してしまう恐れがあり，収束性に悪影響を与えるので，効率がよい探索とはいえない. したがって，図 2.10(c) に示すような，低いランダム性で，かつ適合度が高く多様な個体集団を生成する再生手法が好ましい.

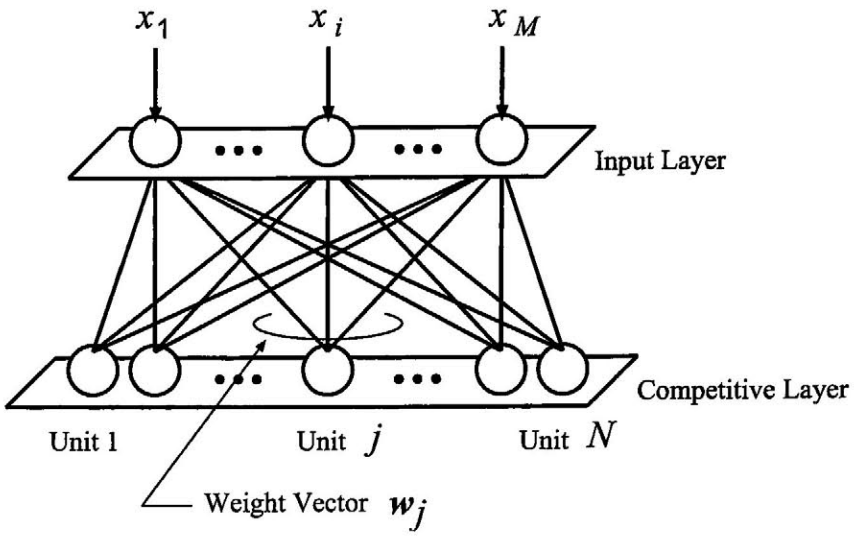


図 2.11: 競合層のユニットを1次元に配置した場合の自己組織化マップの構成.

2.4 自己組織化マップ

自己組織化マップの構成を図2.11に示す. 自己組織化マップはそれぞれ M 個, N 個のユニットをもつ入力層と競合層から構成される. 競合層上の j 番目のユニット (以下では, 競合層ユニット j と表す) は結合重みベクトルを介して入力層ユニットと結合している. 自己組織化マップでは, ベクトル集合を入力層に提示し, 選択された競合層ユニットの結合重みベクトルを更新する. この操作を繰り返し行なうことで, 競合層ユニットの結合重みベクトルはベクトル集合の分布を近似する. この過程を学習と呼ぶ. 自己組織化マップの学習は, 結合重みベクトルを更新する際, ユニットを競合させるので, 特別に競合学習と呼ばれることもある. これは, 教師なし学習の1つである.

一般的な自己組織化マップ [28] では, 入力ベクトルと結合重みベクトルは実数値ベクトルで表現される. しかしながら, バイナリ入力ベクトル集合に対して, バイナリ結合重みベクトルを用いて学習する自己組織化マップのアルゴリズムが提案されており, その有効性が示されている [38]. 本論文では, バイナリ遺伝的アルゴリズム, 実数値遺伝的アルゴリズムとの対応関係を明確にするために, バイナリ入力ベクトルを扱う自己組織化マップをバイナリ自己組織化マップ, 実数値入力ベクトルを扱う自己組織化マップを実数値自己組織化マップと呼び, この二つを総称して自己組織化マップと呼ぶ. 本節では, 各自己組織化マップの学習アルゴリズムと, 特長について説明する.

2.4.1 実数値自己組織化マップ

学習に用いる実数値入力ベクトル集合として，次の K 個のベクトルを考える．

$$\{\boldsymbol{x}_k^R = (x_{k1}^R, \cdots, x_{ki}^R, \cdots, x_{kM}^R) \mid k = 1, \cdots, K\}, \quad x_i^R \in \mathcal{R} \tag{2.6}$$

実数値自己組織化マップの学習は，以下のアルゴリズムに従って行なわれる．

Step S-0: 競合層ユニットの実数値結合重みベクトル $\boldsymbol{w}_j^R = (w_{j1}^R, \cdots, w_{ji}^R, \cdots, w_{jM}^R)$ の初期化を行なう．通常，乱数を用いて全てのベクトル w_{ji}^R の値を決定する．

Step S-1: ベクトル集合の中から 1 つのベクトル $\boldsymbol{x}_k^R$ を選択し入力層へ提示する．以後，特別な場合を除いて入力ベクトル番号 k を省略する．

Step S-2: 入力ベクトルとのユークリッド距離が最小となる結合重みベクトルをもつ競合層ユニット BMU を次式で決定し，勝者ユニットとする．

$$BMU = \arg\min_j \sqrt{\sum_{i=1}^{M}(x_i^R - w_{ji}^R)^2} \tag{2.7}$$

Step S-3: 提示された入力ベクトルに対して，全ての結合重みベクトルを次式によって更新する．

$$\boldsymbol{w}_j^R(t+1) = \boldsymbol{w}_j^R(t) + \alpha_{RSOM}(t)H(d_j, NC_{BMU,SOM}(t))(\boldsymbol{x}^R - \boldsymbol{w}_j^R(t)) \tag{2.8}$$

ここで，t は学習のステップを表す．$\boldsymbol{w}_j^R(t+1)$ と $\boldsymbol{w}_j^R(t)$ はそれぞれ更新後と更新前の結合重みベクトルを表す．$\alpha_{RSOM}(t)$ は実数値自己組織化マップの学習ステップ t における学習係数である．$H(d_j, NC_{BMU,SOM}(t))$ は，近傍関数と呼ばれ，次式で定義される．

$$H(d_j, h_{BMU,SOM}(t)) = \exp(-\frac{d_j^2}{h_{BMU,SOM}(t)^2}) \tag{2.9}$$

d_j は，j 番目の競合層ユニットと勝者ユニット間のユークリッド距離である．$NC_{BMU,SOM}(t)$ は学習ステップ t における近傍係数である．

Step S-4: ベクトル集合の中に一度も選択していないベクトルがある場合は，Step S-1 に戻り，これまで選択していないベクトルを新たな入力ベクトルとする．ベクトル集合内の全てのベクトルを選択した場合は，Step S-5 に進む．

Step S-5: Step S-1〜Step S-4 の操作を規定回数繰り返す．繰り返す過程で，学習係数 $\alpha_{RSOM}(t)$ と近傍係数 $NC_{BMU,SOM}(t)$ を小さくしていく．

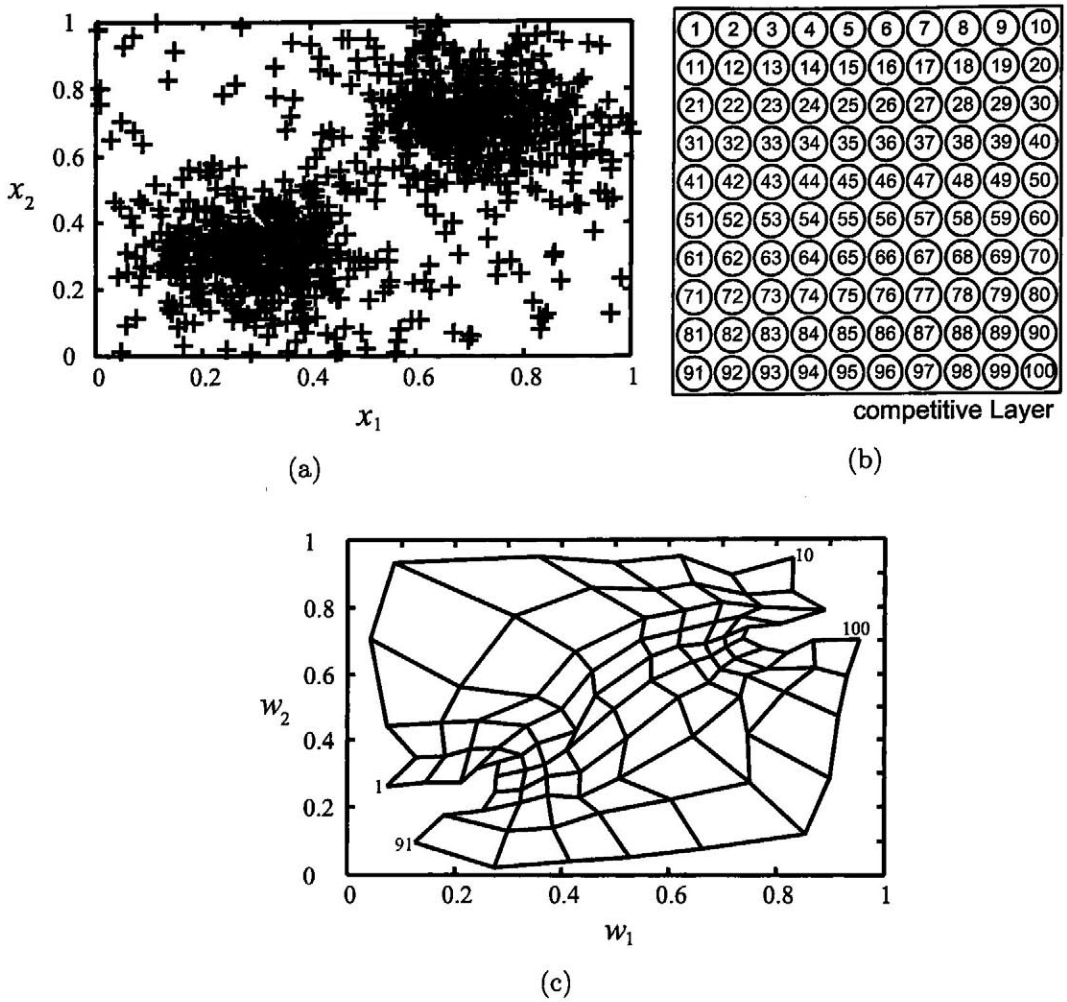


図 2.12: 実数値自己組織化マップの学習の例. (a) 学習に用いた 1000 個の実数値入力ベクトル. (b) 競合層ユニットの配置. (c) 学習後の結合重みベクトルの分布. ×は結合重みベクトルを表し，隣り合うユニットの結合重みベクトルは線で結ばれている. 結合重みベクトルに示された番号は，それをもつユニットの番号を表す.

例として，図 2.12(a) に示す 1000 個のベクトル集合を用いて行なった実数値自己組織化マップの学習結果を示す. このベクトル集合は，正規分布 $ND_{x_1}(0.3, 0.1)$，$ND_{x_2}(0.3, 0.1)$ から得られたベクトル 400 個，正規分布 $N_{x_1}(0.7, 0.1)$，$ND_{x_2}(0.7, 0.1)$ から得られたベクトル 400 個，および一様分布 $UD_{x_1}[0:1]$，$U_{x_2}[0:1]$ から得られたベクトル 200 個から成る. 競合層ユニットの数は，図 2.12(b) に示すように 10×10 の 100 個とした. 図 2.12(b) 中の数字はユニットの番号を表す. 結合重みベクトルの初期値は，一様分布 $UD[0.45:0.55]$ から得られた値を用いて決定した. 学習回数

は，1000 個の入力ベクトル全てを 1 度ずつ提示することを学習の単位とし，1000 回とした．学習係数と近傍範囲の初期値を $\alpha_{RSOM}(0) = 0.5$，$h_{BMU,SOM}(0) = 10$ として，学習終了時に 0 になるように線形に減少させた．学習後の結合重みベクトルの分布を図 2.12(c) に示す．×は結合重みベクトルを示し，隣り合うユニットの結合重みベクトルは線で結ばれている．結合重みベクトルの 1，10，91，100 の数字は，その結合重みベクトルをもつユニット番号を示している．図 2.12(b) と (c) から，結合重みベクトルの位置関係は，競合層ユニットの位置関係を保持していることがわかる．これは，各競合層ユニットの結合重みベクトルが更新される際に，勝者ユニットとの距離が近いほど，その更新量が多いことに起因する．学習後の実数値自己組織化マップに入力ベクトルを提示すると，距離の近い入力ベクトルは，競合層上で近くに位置するユニットを勝者ユニットとする．これは，相対的な位置関係を保持したまま入力ベクトルを競合層へ写像することを意味し，トポロジカルマッピングと呼ばれている．この特徴を利用して，実数値自己組織化マップをパターン分類へ応用した研究が数多く報告されている [30]-[32],[37]．図 2.12(a) と (c) から，入力ベクトルの出現頻度が高い領域に多くの結合重みベクトルが分布していることがわかる．これは，各競合層ユニットの結合重みベクトルが，入力ベクトルに近付く方向に更新されることに起因する．学習に用いた入力ベクトル集合を学習後の実数値自己組織化マップに提示すると，全ての競合層ユニットは等確率で勝者ユニットに選択される．この特徴を利用して，実数値自己組織化マップはベクトル量子化やベクトル集合の確率密度近似，データ解析等へ応用されている [33]-[35]．

以上をまとめると，実数値自己組織化マップは以下の特徴をもつ．

(1) 学習後の結合重みベクトルの配置は，競合層ユニットの位置関係を保持している．

(2) 学習後の結合重みベクトルの分布は，入力ベクトル集合の分布を近似する．

実数値自己組織化マップは，結合重みベクトルの競合層上での配置や，結合重みベクトル空間内での分布により実数値入力ベクトル集合のもつ情報を表現することが可能である．

2.4.2 バイナリ自己組織化マップ

学習に用いるバイナリ入力ベクトル集合として，次の K 個のベクトルを考える．

$$\{x_k^B = (x_{k1}^B, \cdots, x_{ki}^B, \cdots, x_{kn}^B) \mid k = 1, \cdots, K\}, \quad x_i^B \in \mathcal{B} \tag{2.10}$$

バイナリ自己組織化マップの学習は，実数値自己組織化マップの学習と同様の流れに従って行なわれる．しかしながら，ベクトルの表現方法の違いから，結合重みベクトルの初期化，ベクトル間の距離尺度，結合重みベクトルの更新方法が実数値自己組織化マップの場合と異なる．バイナ

リ自己組織化マップの学習アルゴリズムにおいて，実数値自己組織化マップの学習アルゴリズムと異なる部分を以下に示す．

Step S-0: 競合層ユニットのバイナリ結合重みベクトル $\boldsymbol{w}_j^B = (w_{j1}^B, \cdots, w_{ji}^B, \cdots, B_{jM}^R)$ の初期化を行なう．通常，乱数を用いて全てのベクトル w_{ji}^B の値を ‘0’ か ‘1’ に決定する．

Step S-2: 入力ベクトルとのハミング距離が最小となる結合重みベクトルをもつ競合層ユニットを次式で決定する．

$$BMU = \arg\min_j \{\sum_{j=1}^{M} \{(x_i \oplus w_{ji})\} \tag{2.11}$$

ここで，$\oplus$ は，排他的論理和 (XOR) 演算を表し，XOR の値が 1 の場合，x_i と w_{ji} が互いに異なることを意味する．したがって，(2.11) 式は，入力ベクトルと j 番目の重みベクトルの各要素を比較し，互いに異なる要素の数が最小のユニットを BMU として定義することを意味する．

Step S-3: 提示された入力ベクトルに対して，全ての結合重みベクトルを更新する．近傍関数は，実数値自己組織化マップと同様に (2.9) 式によって定義される．更新は，入力ベクトルと結合重みベクトルの対応する要素に対して XOR をとり，XOR が 1 の要素を $\alpha_{BSOM}(t)$ 個，入力ベクトルに揃えることで実現される．ここで，$\alpha_{BSOM}(t)$ は，ステップ t における学習係数であり，正の整数で表される．

これらの演算を導入することにより，バイナリ自己組織化マップは，入力ベクトルがバイナリベクトルで表現されている場合においても，その確率密度関数を近似することが可能となる．

例として，図 2.13(a) に示す 4 個のバイナリ入力ベクトル集合を用いて行なったバイナリ自己組織化マップの学習結果を示す．このベクトル集合に含まれる各入力ベクトルは，64 個のバイナリ値から構成されており，8×8 ピクセルのバイナリ画像を表現している．競合層ユニットの数は，16 個として，2 次元に配置した．結合重みベクトルの初期値は，ランダムに決定した．学習回数は，4 個の入力ベクトル全てを 1 度ずつ提示することを学習の単位とし，1000 回とした．学習係数の初期値を $\alpha_{BSOM}(0) = 16$，近傍範囲の初期値は，$h_{BMU,SOM}(0) = 2$ として，それぞれ，学習終了時に 0 になるように線形に減少させた．学習後の結合重みベクトルを図 2.13(b) に示す．図 2.12(b) から，バイナリ自己組織化マップは，結合重みベクトルによって，バイナリ入力ベクトル集合を近似できていることがわかる．また，競合層の中心付近に位置するユニットの結合重みベクトルは，2 つの異なる入力ベクトルの特徴を近似していることから，学習後の結合重みベクトルは，4 つの入力ベクトル集合を近似し，かつ，その集合を内挿していることがわかる．これらのことから，バイナリ自己組織化マップは，実数値遺伝的アルゴリズムと同様の特長をもつといえる．

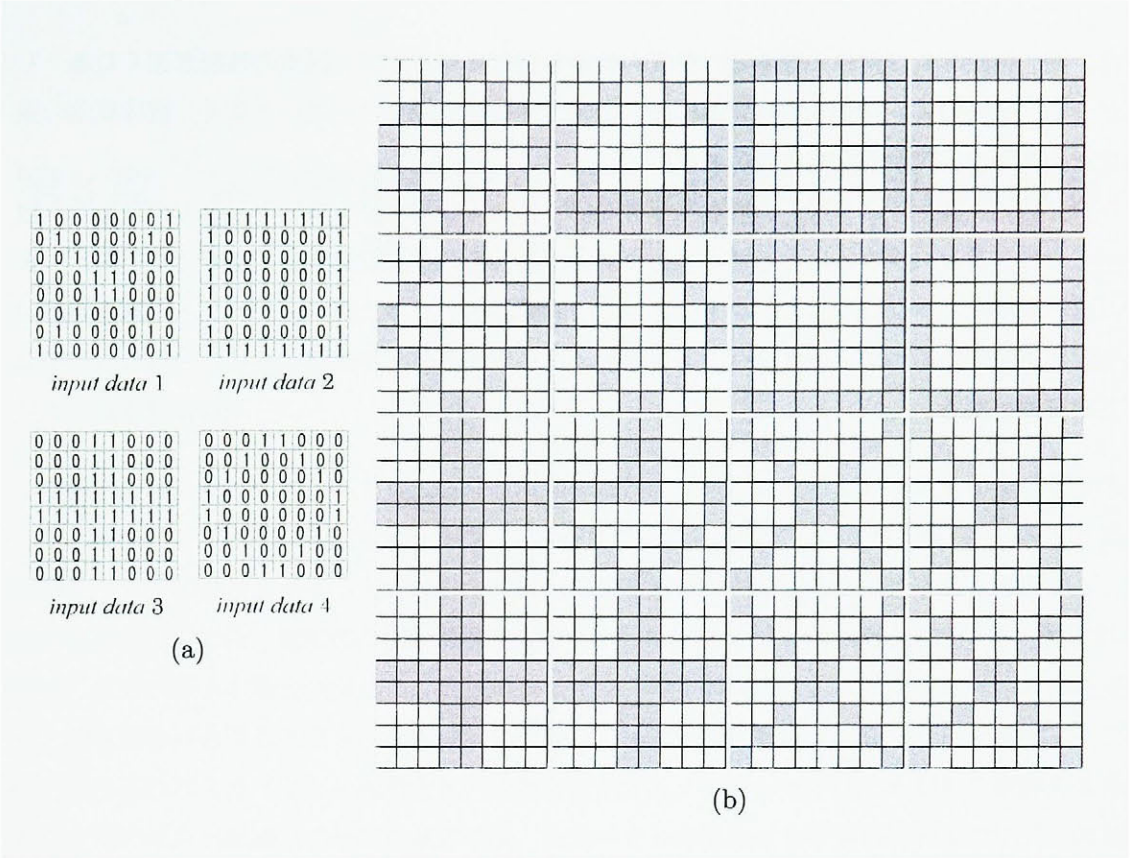


図 2.13: バイナリ自己組織化マップの学習の例．(a) 学習に用いた 4 個のバイナリ入力ベクトル．(b) 学習後の結合重みベクトル．(b) において，各結合重みベクトルの表示位置は競合層ユニットの配置と対応している．

2.5 おわりに

本章では，まず，遺伝的アルゴリズムの概要と探索アルゴリズムについて説明し，従来の再生手法の問題点について述べた．次に自己組織化マップの構造と学習アルゴリズムについて述べた．

従来の遺伝的アルゴリズムでは，個体の表現方法がバイナリベクトル，実数値ベクトルにかかわらず，ルーレット選択等の再生手法が用いられる．これらの再生手法は，現世代の個体集団から高い適合度をもつ個体を次世代にコピーすることによって実現される．再生操作により，高い適合度を持つ個体を高確率で次世代に残すことが可能である．したがって，個体集団全体の適合度の平均値を上昇させることが可能となる．しかしながら，従来の再生操作では，個体集団内から適合度が高い個体を選択する際に，同じ個体が複数回選択される．これは，個体集団の多様性の減少を引き起こし，適合度が低い個体が淘汰され，高い個体が増殖されることに起因する．個体集団の多様性減少は，探索点の減少を意味し，探索初期における局所解への収束を引き起こす．

また，同一性質をもつ個体の増加は，交叉の効率も悪化させるので，探索の性能が悪くなる．したがって，効率のよい探索を実現するには，個体の多様性を減少させることなく，個体集団の適合度を上昇させる再生手法が必要がある．

一方，自己組織化マップは，教師なし学習を行うニューラルネットワークのひとつであり，自身がもつ結合重みベクトルによって，入力ベクトル集合の分布を近似することが可能である．また，自己組織化マップは，入力ベクトルとして，実数値ベクトルとバイナリベクトル双方を用いることが可能であるので，実数値ベクトル集合，バイナリベクトル集合どちらの分布も近似することが可能である．

第3章　自己組織化マップを再生に用いた遺伝的アルゴリズム

3.1　はじめに

本研究では，遺伝的アルゴリズムにおける個体集団の多様性喪失による探索効率悪化という問題点を解決するために，遺伝的アルゴリズムの特長である“集団による多点探索”をふまえ，自己組織化マップの特長である“入力ベクトル集合の分布近似”を利用した再生手法を提案する．提案する再生手法では，現世代の個体集団を入力ベクトル集合として自己組織化マップへ提示する．次に，入力ベクトル集合に対して，自己組織化マップの学習過程により結合重みベクトルを更新し，学習後の結合重みベクトルを再生後の個体として次世代に残す．従来の自己組織化マップでは，提示された入力ベクトル集合全体の分布を近似する．しかしながら，本研究で提案する再生手法に用いた自己組織化マップの学習では，入力ベクトル集合として遺伝的アルゴリズムの個体集団を扱うので，個体の適合度に基づいて結合重みベクトルの更新を行う．これによって，提案する再生手法では，提示された入力ベクトル集合内の高い適合度をもつ入力ベクトル集合の分布を近似することが可能となる．また，自己組織化マップは，バイナリ，実数値双方の入力ベクトルを扱うことが可能であるので，バイナリ自己組織化マップと実数値自己組織化マップを再生に用いることにより，バイナリ遺伝的アルゴリズムと実数値遺伝的アルゴリズムへ適用可能である．

本章では，本研究で提案する再生手法について説明する．

3.1 節は，本章の概要である．

3.2 節では，提案する再生手法の概要と特長について説明する．

3.3 節では，まず，実数値自己組織化マップを再生に用いた実数値遺伝的アルゴリズムについて説明する．次に，提案する再生手法の基本的な特性を示すために，連続変数最適化問題を解く．本節では，DeJong により提案されたベンチマーク関数のひとつである“Rosenbrock's Saddle”関数を連続変数最適化問題として用いた．この関数は，探索アルゴリズムの収束性を調べることが可能である．さらに，提案する再生手法を DeJong により提案された別のベンチマーク関数である“Quadratic With Noise”関数と“Shekels' Foxfoles”関数に適用し，探索のロバスト性と大域的探索能力について従来の再生手法と比較し，提案手法の有効性を示す．さらに，提案する再生手法の応用として，実数値遺伝的アルゴリズムを用いて，タンク系の水位を調節する PID コント

ローラを設計する.

3.4節では，まず，バイナリ自己組織化マップを再生に用いたバイナリ遺伝的アルゴリズムについて説明する．次に，提案する再生手法を用いて，組み合わせ最適化問題のひとつである0-1ナップサック問題を解く．従来の再生手法と比較することによって，提案する再生手法の有効性を示す.

3.5節では，要素の重要性を考慮したバイナリ結合重みベクトル更新則を提案し，バイナリ自己組織化マップに適用する．従来のバイナリ結合重みベクトル更新則では，入力ベクトルの各要素の重要度が等しく扱い，結合重みベクトル内の各要素を入力ベクトルに揃える操作を行う．その際，ベクトル内の要素の更新順序が探索の性能に影響を与えるので，探索効率が悪化する場合がある[43][44]．提案する結合重みベクトル更新則では，適合度に関して入力ベクトル集合の各要素の重み付き平均値を求め，これをもとに要素の更新順序を決定する．提案するバイナリ結合重みベクトル更新則を採用したバイナリ自己組織化マップを再生に適用し，3.4節と同様に0-1ナップサック問題を解くことにより，その有効性を示す.

3.6節は，本章のまとめである.

3.2 自己組織化マップを再生に用いた遺伝的アルゴリズム

本研究で提案する自己組織化マップを再生に用いた遺伝的アルゴリズムの概要を図3.1に示す．提案する自己組織化マップによる再生手法では，現世代の個体集団を自己組織化マップへの入力ベクトル集合，次世代の個体集団を自己組織化マップの結合重みベクトル集合として取り扱う．すなわち，次世代の個体集団は，自己組織化マップの学習によって生成される．本研究では，望ましい再生を実現するために，表3.1に示すような新しい学習機能をもつ自己組織化マップを導入する．表3.1に示すように，通常の自己組織化マップでは，学習後の結合重みベクトルは，提示された入力ベクトル集合内のすべての入力ベクトルの分布を近似する．しかしながら，本研究では，各入力ベクトルを個体として扱うので，入力ベクトルに適合度が付随した場合の学習を導入する．本研究で導入する自己組織化マップの学習では，結合重みベクトルは，入力ベクトル集合内の高い適合度をもつ入力ベクトルのみの分布を近似する．また，一般的に，自己組織化マップでは，少数の結合重みベクトル集合を用いて，大量の入力データ集合の分布を近似することにより，ベクトル量子化やパターン認識等へ応用されてきた．しかしながら，本研究では，自己組織化マップへ提示する入力ベクトル集合と結合重みベクトル集合は，それぞれ，遺伝的アルゴリズムにおける個体集団を表す．したがって，各ベクトル集合内のベクトル数は，遺伝的アルゴリズムの個体数と等しい．これらの点を導入することによって，自己組織化マップの学習によって，次世代の個体集団は，現世代における各個体の適合度に基づいて生成することが可能となる.

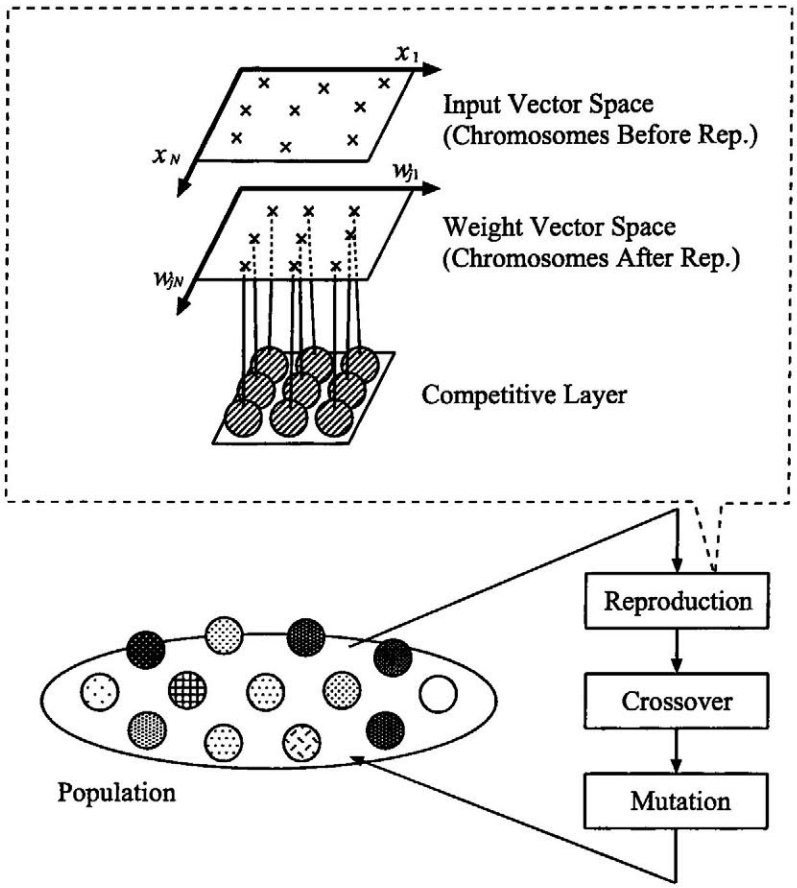


図 3.1: 提案する再生手法の概要.

表 3.1: 従来の自己組織化マップと本研究で導入する自己組織化マップの学習の違い

	入力ベクトルと結合重みベクトルの数	近似する分布の対象
従来の学習	入力ベクトル数 >> 結合重みベクトル数	全入力ベクトル
導入する学習	入力ベクトル数 = 結合重みベクトル数	適合度の高い入力ベクトル

3.3 実数値自己組織化マップを再生に用いた実数値遺伝的アルゴリズム

3.3.1 探索アルゴリズム

実数値自己組織化マップを再生として用いた遺伝的アルゴリズムにおいて，探索（個体）空間，入力ベクトル空間，結合重みベクトル空間の関係を図 3.2 に示す．図 3.2 において，各空間は個別に表記しているが，すべて同じ次元をもち，適合度を算出することが可能である．適合度に基づ

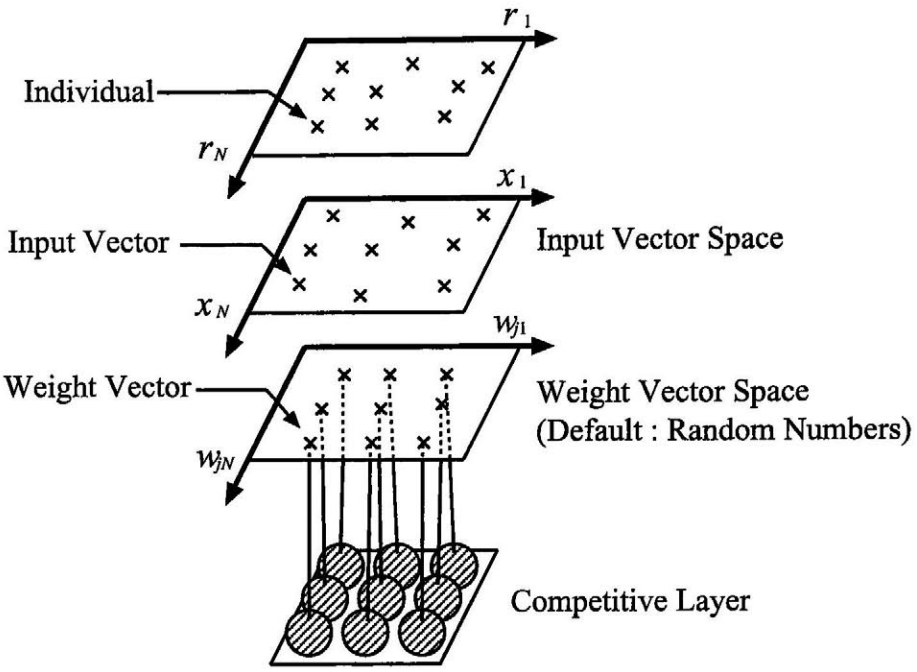


図 3.2: 自己組織化マップを再生として用いた遺伝的アルゴリズムにおける探索空間，入力ベクトル空間，結合重みベクトル空間の関係.

いて結合重みベクトルを更新する実数値自己組織化マップを再生として用いた実数値遺伝的アルゴリズムでは，以下のアルゴリズムにしたがって探索が行われる.

Step G-0: 乱数を用いて個体集団 I^R の初期化を行い，各個体の適合度を算出する. 次に，従来の実数値自己組織化マップに，入力ベクトル空間からランダムに選んだ実数値入力ベクトルを提示し，結合重みベクトルの更新を行う. この操作を繰り返すことにより，結合重みベクトルは，実数値結合重みベクトル空間内において，一様に配置される.

Step G-1: 個体集団を実数値入力ベクトル集合として ($I^R = x^R$)，各入力ベクトルの適合度をもとに，以下の再生アルゴリズムによって次世代に残す個体を決定する.

Step S-1: 入力ベクトル集合の中から1つのベクトル x_k^R を選択し入力層へ提示する.

Step S-2: 入力ベクトルとのユークリッド距離が最小となる結合重みベクトルをもつ競合層ユニット BMU を次式で決定し，勝者ユニットとする.

$$BMU = \arg\min_j \sqrt{\sum_{i=1}^{M}(x_i^R - w_{ji}^R)^2} \tag{3.1}$$

Step S-3: 全ての結合重みベクトルを次式で更新する．

$$\boldsymbol{w}_j^R(t+1) = \boldsymbol{w}_j^R(t) + f_{\boldsymbol{x}^R} \cdot H(d_j, f_{\boldsymbol{x}^R}) \cdot (1 - f_{\boldsymbol{w}_j^R(t)})(x^R - \boldsymbol{w}_j^R(t)) \tag{3.2}$$

ここで，$f_{\boldsymbol{x}^R}$ と $f_{\boldsymbol{w}_j^R}$ は，それぞれ入力ベクトルと更新前の結合重みベクトルの適合度である．また，$H(d_j, f_{\boldsymbol{x}^R})$ は近傍関数であり，次式で定義される．

$$H(d_j, f_{\boldsymbol{x}^R}) = \exp\left(-\frac{{d_j}^2}{2{f_{\boldsymbol{x}^R}}^2}\right) \tag{3.3}$$

Step S-4: ベクトル集合の中に一度も選択していないベクトルがある場合は，Step S-1 に戻り，これまで選択していないベクトルを新たな入力ベクトルとする．ベクトル集合内の全てのベクトルを選択した場合は，Step S-5 に進む．

Step S-5: 学習後の結合重みベクトル集合を再生後の個体集団とする．
$(\boldsymbol{w}_j^R(t+1) = \boldsymbol{I}^R)$

Step G-2: 個体集団から親個体を選出し，交叉によって子個体を生成する．

Step G-3: 個体集団から個体を選出し，突然変異によって個体を生成する．

Step G-4: 集団内の各個体の適合度を算出し，終了条件を満たしていれば終了する．
終了条件を満たしていない場合は，Step G-1 に戻る．

提案する再生手法に用いる実数値自己組織化マップでは，Step S-3 の結合重みベクトル更新方法が，従来の実数値自己組織化マップの更新方法とは異なる．(3.2) 式では，$f_{\boldsymbol{x}^R} \cdot H(d_j, f_{\boldsymbol{x}^R}) \cdot (1 - f_{\boldsymbol{w}_j^R(t)})$ が，従来の実数値自己組織化マップの結合重みベクトルの更新式 (2.8) 式の学習係数 $\alpha_{RSOM}(t)$ に対応している．学習係数は，結合重みベクトルを入力ベクトルに近づける割合（更新量）を決定するパラメータである．(3.2) 式では，以下の 3 つの観点に基づいて結合重みベクトルの更新量を決定する．

(1) $f_{\boldsymbol{x}^R}$:入力ベクトルの適合度が高い場合，一つの結合重みベクトルの更新量を大きくする．

(2) $H(d_j, f_{\boldsymbol{x}^R})$:入力ベクトルの適合度が高い場合，多くの結合重みベクトルの更新量を大きくする．

(3) $(1 - f_{\boldsymbol{w}_j^R(t)})$:更新前の結合重みベクトルの適合度が高い場合，結合重みベクトルの更新量を小さくする．

(1) は，低い適合度をもつ結合重みベクトルは，高い適合度をもつ入力ベクトルに引き寄せられることを意味する．(2) において，$H(d_j, f_{\boldsymbol{x}^R})$ は，(3.3) 式で定義される．(3.3) 式では，図 3.3 に

示すように，入力ベクトルの適合度をガウス分布の分散として用いることにより，入力ベクトルの適合度に応じて，各結合重みベクトルの更新量を決定することを意味する．(3) は，既に最適解に近づいた解を，ある程度保持することを意味する．(3) において，“1”は，最適，もしくは準最適解の適合度を表す．

例として，従来の実数値自己組織化マップと提案する再生手法に用いる実数値自己組織化マップの学習データと学習過程を図 3.4，3.5 に示す．図 3.4(a) は，2 章で示した図 2.12(a) と同じ方法で作成した 1000 個のベクトル集合である．結合重みベクトルの初期値は，一様分布 $UD[0.45:0.55]$ から得られた値を用いて決定した．学習回数は，1000 個の入力ベクトル全てを 1 度ずつ提示することを学習の単位とし，1000 回とした．学習係数と近傍範囲の初期値を $\alpha_{RSOM}(0)=0.8$，$h_{BMU,SOM}(0)=14$ として，学習終了時に 0 になるように線形に減少させた．200 ステップごとの学習後の結合重みベクトルの分布を図 3.4(b)〜(f) に示す．図 3.4(b)〜(f) より，従来の自己組織化マップでは，学習が進行していくごとに，結合重みベクトルが (2.8) 式によって入力ベクトル集合全体の分布を近似していくことがわかる．一方，提案する再生手法に用いる自己組織化マップでは，図 3.5(a) に示す 100 個の各入力ベクトルに適合度が付随したベクトル集合を考える．このベクトル集合は，正規分布 $ND_{x_1}(0.3,0.1)$，$ND_{x_2}(0.3,0.1)$ から得られたベクトル 50 個，正規分布 $N_{x_1}(0.7,0.1)$，$ND_{x_2}(0.7,0.1)$ から得られたベクトル 50 個から成る．また，図中の＋と＊は，それぞれ適合度が 0.8 と 0 の入力ベクトルを表す．学習回数は，100 個の入力ベクトル全てを 1 度ずつ提示することを学習の単位とし，30 回とした．6 ステップごとの学習後の結合重みベクトルの分布を図 3.5(b)〜(f) に示す．図 3.5(b)〜(f) より，提案する再生手法に用いる自己組織化マップでは，(3.2) 式が意味する 3 つの観点を総合的に考慮して結合重みベクトルを更新することによって，個体集団の高い多様性を維持し，かつ，高い適合度をもつ個体集団のみの分布を近似することが可能となる．その結果，交叉の効果を最大限に発揮させることが可能であり，効率のよい探索を実現することができる．また，従来の学習係数に相当する箇所に入力ベクトルの適合度を当てはめることにより，学習の収束が速くなり，繰り返し学習を必要としなくなることがわかる．

3.3.2 Rosenbrock's Saddle 関数への適用

本小節では，本研究で提案する再生手法の有効性を検証するために，以下に示すベンチマーク関数を連続変数最適化問題として解く．

Problem 1 Minimize $100(r_1^2-r_2)^2+(1-r_1)^2$, $-5.11 \le r_k \le 5.11$

この関数は，DeJong が提案したベンチマーク関数 [11] のひとつで，Rosenbrock's Saddle 関数と呼ばれる単峰性関数であり，局所的な収束性を検証する連続変数最適化問題である．この問題の

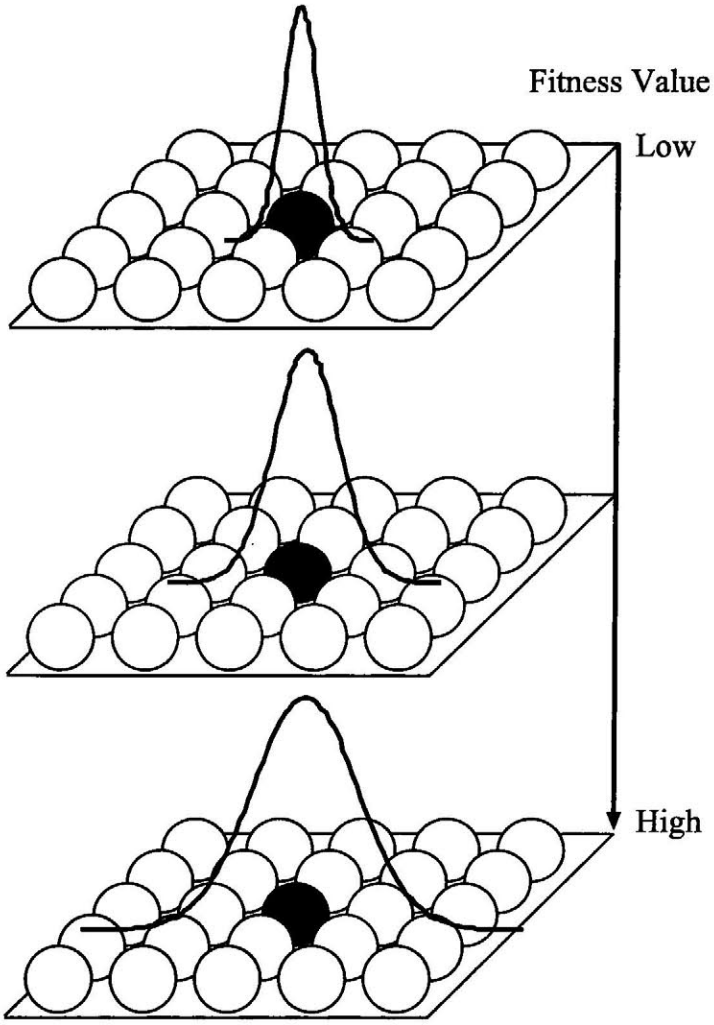


図 3.3: 提案する近傍関数の概形.

最適解は，$(r_1, r_2) = (1, 1)$ である．この関数の概形とその最適解付近の拡大図を図 3.6 に示す．

本小節のシミュレーションでは，提案する再生手法と従来のルーレット選択（RWS）により，個体集団の多様性と適合度が，世代ごとにどのように変化するかを示し，その結果について考察する．表 3.2 に，シミュレーションに用いた各実数値遺伝的アルゴリズムのパラメータを示す．このシミュレーションでは，各実数値遺伝的アルゴリズムの交叉と突然変異にそれぞれ，BLX-α 交叉と一様突然変異を用いた．RWS1 と RWS2 では，使用する遺伝的操作は同じであるが，交叉と突然変異が起こる確率が異なる．

図 3.7，3.8，3.9(a) に，各遺伝的アルゴリズムで探索した際の個体集団の多様性を示す．また，図 3.7，3.8，3.9(b) に，各遺伝的アルゴリズムで探索した際の集団内の個体の適合度の最大値と平均値を示す．図 3.7(a) は，RWS1 において個体集団の多様性の喪失が起こっていることを示し

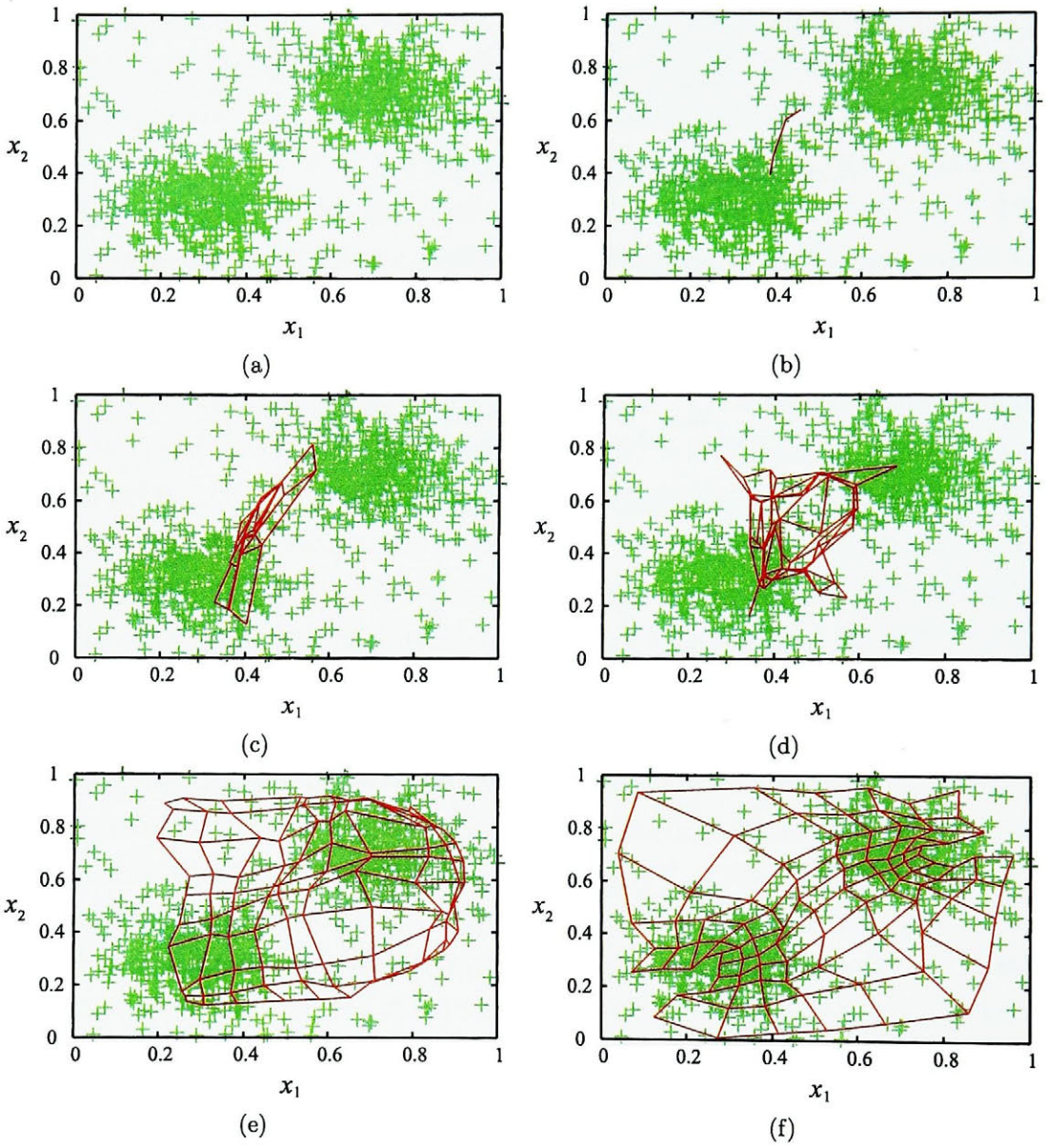


図 3.4: 従来の自己組織化マップにおける学習過程の結合重みベクトル. (a) 学習データ. (b)200 回学習後. (c)400 回学習後. (d)600 回学習後. (e)800 回学習後. (f)1000 回学習後.

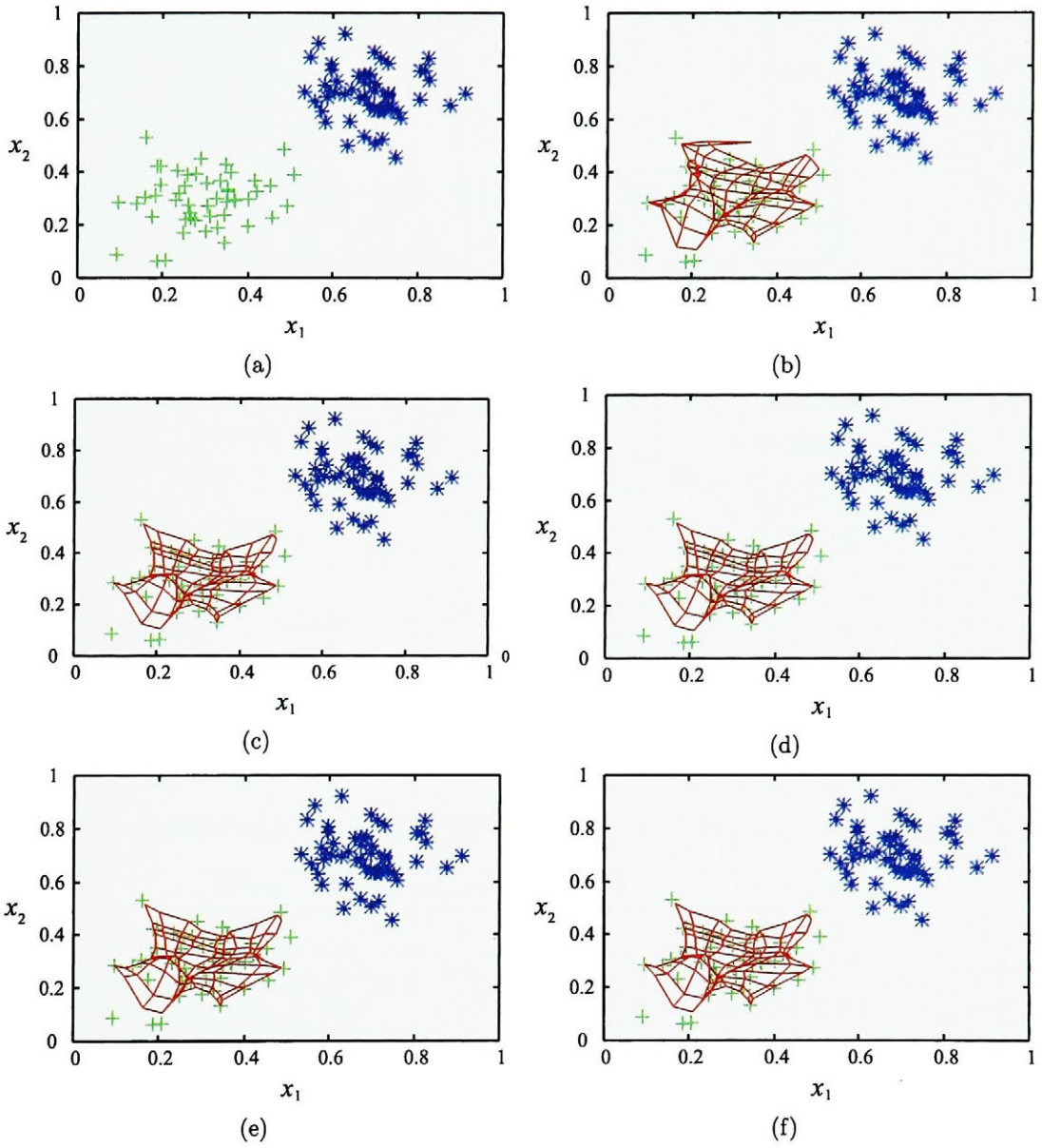


図 3.5: 提案する自己組織化マップにおける学習過程の結合重みベクトル．(a) 学習データ．図中の“＋”は適合度が高い入力ベクトル，“＊”は適合度が低い入力ベクトルを表す．(b)6 回学習後．(c)12 回学習後．(d)18 回学習後．(e)24 回学習後．(f)30 回学習後．

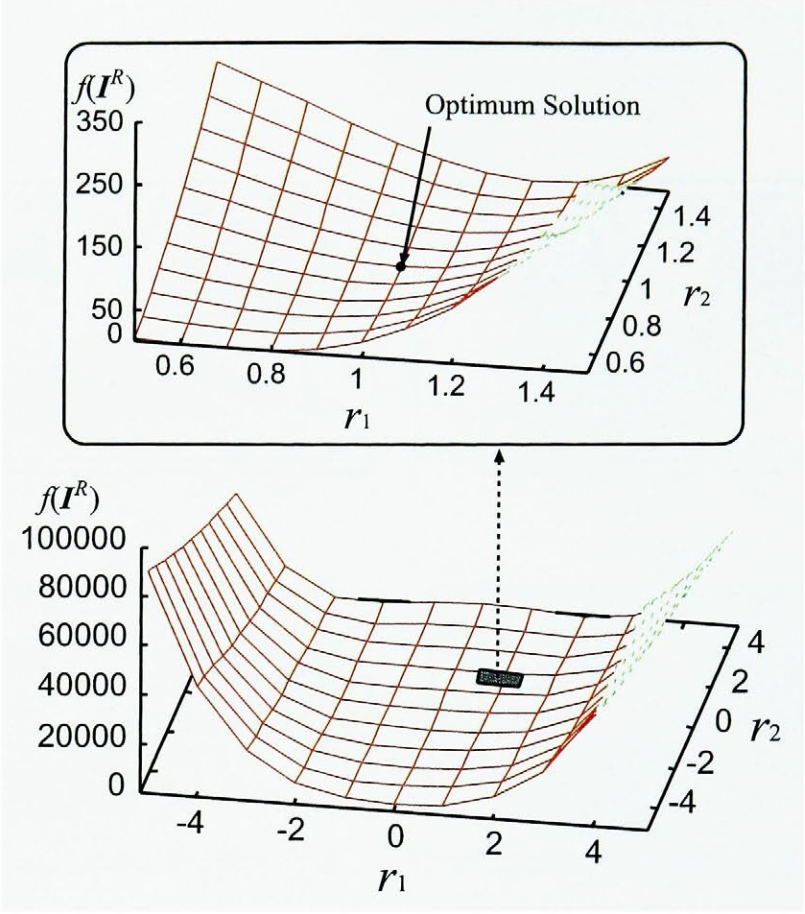


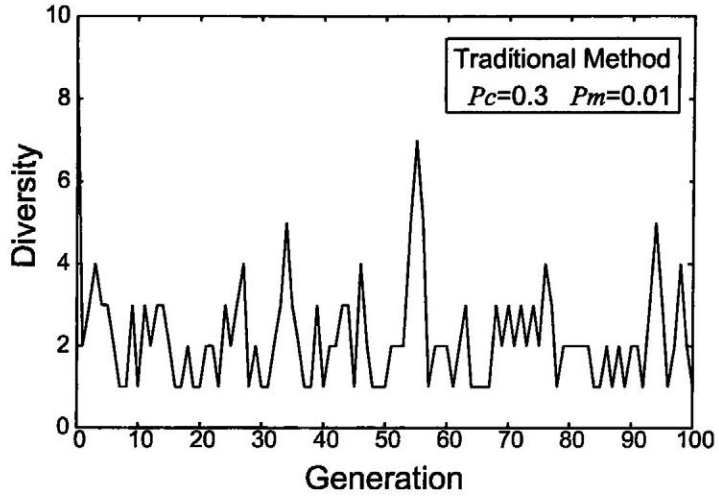
図 3.6: DeJong の Rosenbrock's Saddle 関数.

表 3.2: Problem 1 を解く際に用いた実数値遺伝的アルゴリズムのパラメータ

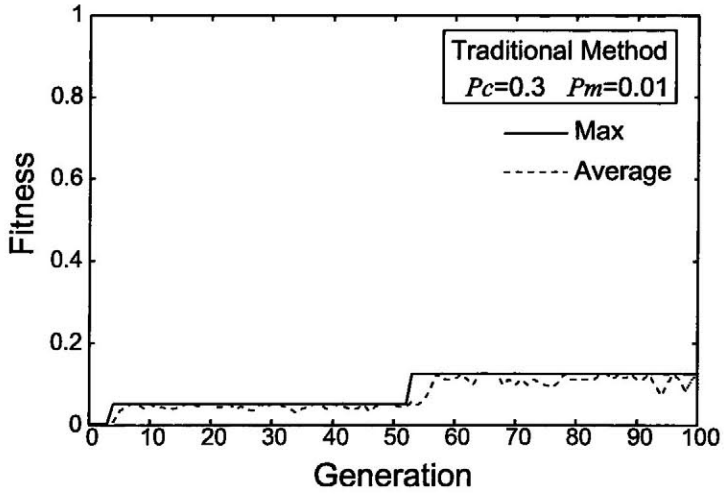
再生手法	Proposed	RWS1	RWS2
集団のサイズ N	9	9	9
交叉を起こす確率 Pc	0.3	0.3	0.8
突然変異を起こす確率 Pm	0.01	0.01	0.8

ている．これは，個体集団内に完全に同一の性質をもつ個体が複数個存在することを意味しており，交叉の効果が最大限に発揮されていないということを示している．したがって，図 3.7(b) のように，個体集団内の適合度の上昇に多くの世代を要する．つまり，従来の再生手法を用いた遺伝的アルゴリズムの探索は，非効率である．

一方，突然変異の確率を上げることによって，個体の多様性を維持することは可能である．図 3.8(a) において，RWS2 を用いた場合は，個体の多様性が維持されている．しかしながら，図 3.8(b)



(a)

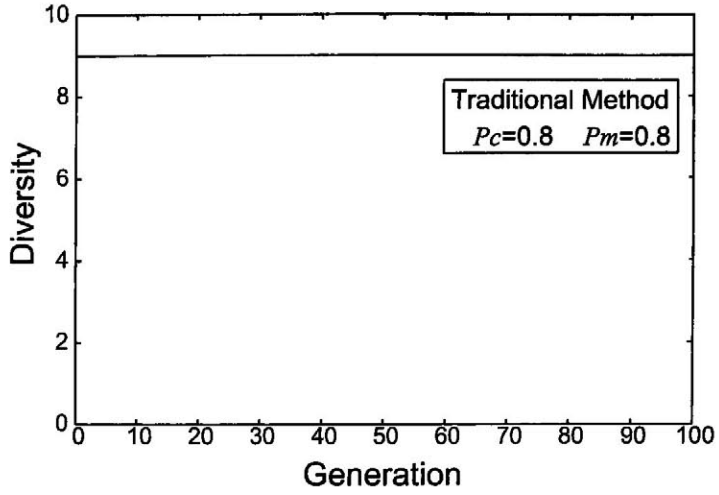


(b)

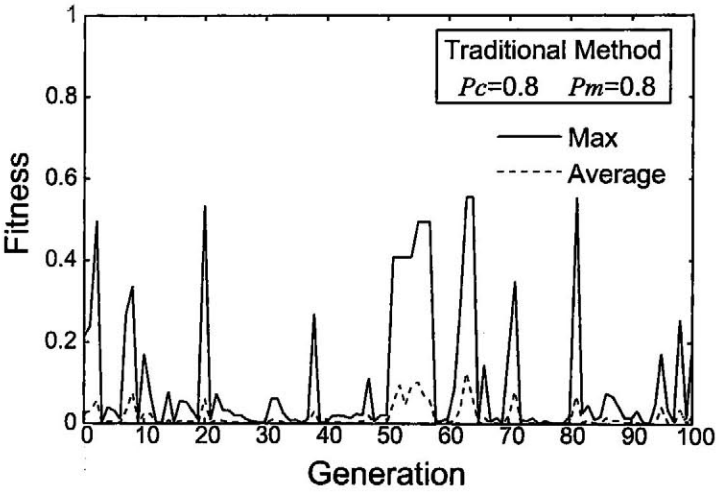
図 3.7: ルーレット選択（RWS1）を用いた実数値遺伝的アルゴリズムを用いて Problem 1 を解いた際の個体集団の多様性と適合度 ($P_c = 0.3, P_m = 0.01$). (a) 個体集団の多様性. (b) 個体集団の適合度.

では，個体集団の適合度の最大値と平均値が世代ごとに上昇しているとはいえない．これは，高い確率の突然変異が，再生によって次世代に残された良質な個体を破壊してしまうことに起因する．したがって，図 3.7 図 3.8 から，効率のよい探索を実現するためには，ランダム性を抑えた高い多様性の維持が必要である．

図 3.9(a) において，提案する再生手法では，図 3.8(a) と同様に個体集団の高い多様性を維持することができている．また，図 3.9(b) に示すように，個体集団の適合度の平均値が上昇している



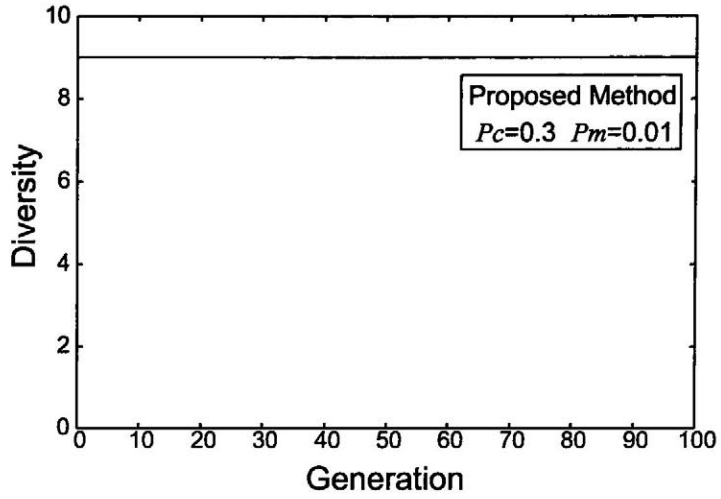
(a)



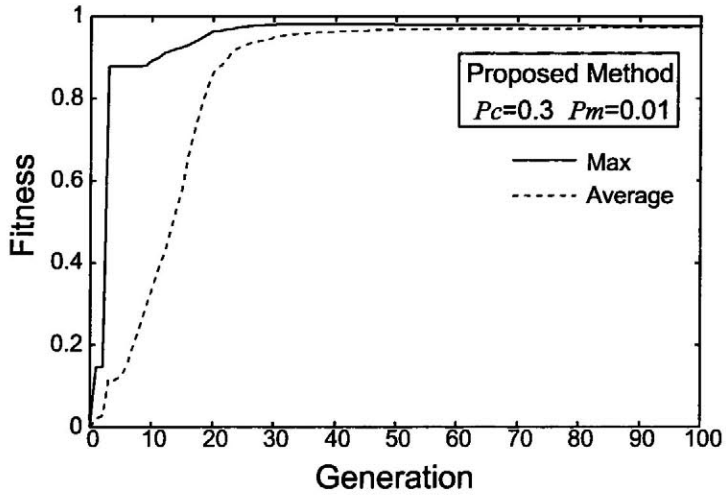
(b)

図 3.8: ルーレット選択（RWS2）を用いた実数値遺伝的アルゴリズムを用いて Problem 1 を解いた際の個体集団の多様性と適合度 ($P_c = 0.8, P_m = 0.8$). (a) 個体集団の多様性. (b) 個体集団の適合度.

ことから，提案する再生手法では，個体集団の高い多様性を維持し，かつ，高い適合度をもつ個体集団を生成することができていることがわかる．これは，本研究で導入した実数値自己組織化マップが，高い適合度をもつ個体集団のみの分布を近似できていることを示している．したがって，提案する再生手法は，高い多様性と適合度をもつ個体集団を生成することによって，交叉の効果を最大限に発揮させており，その結果，図 3.9(b) に示すように，個体集団の適合度の最大値を急速に上昇させることができる．



(a)



(b)

図 3.9: 提案する再生手法を用いた実数値遺伝的アルゴリズムを用いて Problem 1 を解いた際の個体集団の多様性と適合度 ($P_c = 0.3, P_m = 0.01$). (a) 個体集団の多様性. (b) 個体集団の適合度.

探索の例として，ルーレット選択を用いた実数値遺伝的アルゴリズムにおける個体集団の分布を図 3.10 に示す．図 3.10 の (a)〜(f) において，×と＋は，それぞれ個体（探索点）と最適解を表す．この例では，実数値遺伝的アルゴリズムの個体数，交叉，突然変異を起こす確率を，それぞれ 25 個，0.3，0.05 として，100 世代まで探索を行った．図 3.10 より，ルーレット選択を用いた実数値遺伝的アルゴリズムでは，探索初期から個体集団の多様性が著しく減少していくことがわかる．また，図 3.10(f) に示す 100 世代後における探索空間の拡大図から，複数の個体が，完全に同一の

性質をもっていることがわかる．したがって，この状態は，局所的に収束しているので，交叉の性能が発揮できない．つまり，このような状態に陥ってしまうと，突然変異によって良質な解を発見する以外に，探索を進めることはできない．一方，同じ例に対して，提案する再生手法を用いた実数値遺伝的アルゴリズムにおける個体集団の分布を図 3.11 に示す．図 3.11 より，提案する再生手法を用いた実数値遺伝的アルゴリズムでは，世代が進行しても個体集団の多様性が維持されており，最終的に最適解付近へ収束していくことがわかる．また，3.11(f) に示す 100 世代後における探索空間の拡大図から，100 世代後の個体集団は，図 3.6 に示した探索空間の拡大図における高い適合度の領域に分布していることがわかる．したがって，提案する自己組織化マップの学習が，高い多様性の維持と高い適合度をもつ個体集団のみの分布の近似を可能にしているといえる．これらの結果から，実数値自己組織化マップを用いた再生手法は，単峰性関数における局所的な収束性という点について，従来のルーレット選択と比較して，効率のよい探索を実現できている．

また，ルーレット選択において多様性を維持する方法として，選択後の個体集団に乱数を加えることを考える．具体的には，各世代において，ひとつの個体を選ぶ際に，その個体がすでに一度選択されていた場合，一様乱数を加算した個体を次世代に残すという処理を行う．この処理をルーレット選択に加えることによって，多様な個体集団を次世代に残すことが可能となる．図 3.12 に，一様乱数によって多様性を維持したルーレット選択を用いた実数値遺伝的アルゴリズムにおける Problem 1 の探索過程を示す．図 3.12(a) は，重複した個体に，一様分布 $UD[-0.5 : 0.5]$ を加算した際の 100 世代後の個体集団を表す．また，図 3.12(b)(c) は，重複した個体に一様分布 $UD[-0.05 : 0.05]$ を加算した際の 100 世代後の個体集団とその拡大図を表す．図 3.12(a)(b) より，ルーレット選択に乱数を加えた場合は，図 3.10(e)(f) と比較して，個体集団の多様性を維持することができているといえる．ここで，ほぼ同じ範囲の領域に個体集団が分布している図 3.11(f) と図 3.12(c) を比較することにより，提案する再生手法と乱数によって多様性を維持したルーレット選択の比較する．図 3.11(f) では，個体集団の多様性は維持できているものの，図 3.6 に示した探索空間の拡大図における高い適合度の領域での分布を近似するまでには至っていないことがわかる．この場合，ルーレット選択に基づいて個体集団の多様性を維持できたとしても，準最適解には至らない程度の適合度の領域において個体集団の収束が起こることが考えられる．したがって，このような状況に陥ってしまうと，交叉の効果がなくなり，高確率の突然変異が必要になる．それに対して，図 3.11(a)〜(f) では，初期化によって探索空間に一様に分布した個体集団が，図 3.6 に示した高い適合度の領域を包括するように進化していくことがわかる．このことは，提案する再生手法に用いた実数値自己組織化マップが，適合度の高い個体集団の分布を詳細に近似することができていることを示している．したがって，探索の終盤においても，提案する再生手法は，個体集団が局所的に収束することなく，高い適合度の領域内で交叉の性能を発揮することにより，高速な探索が

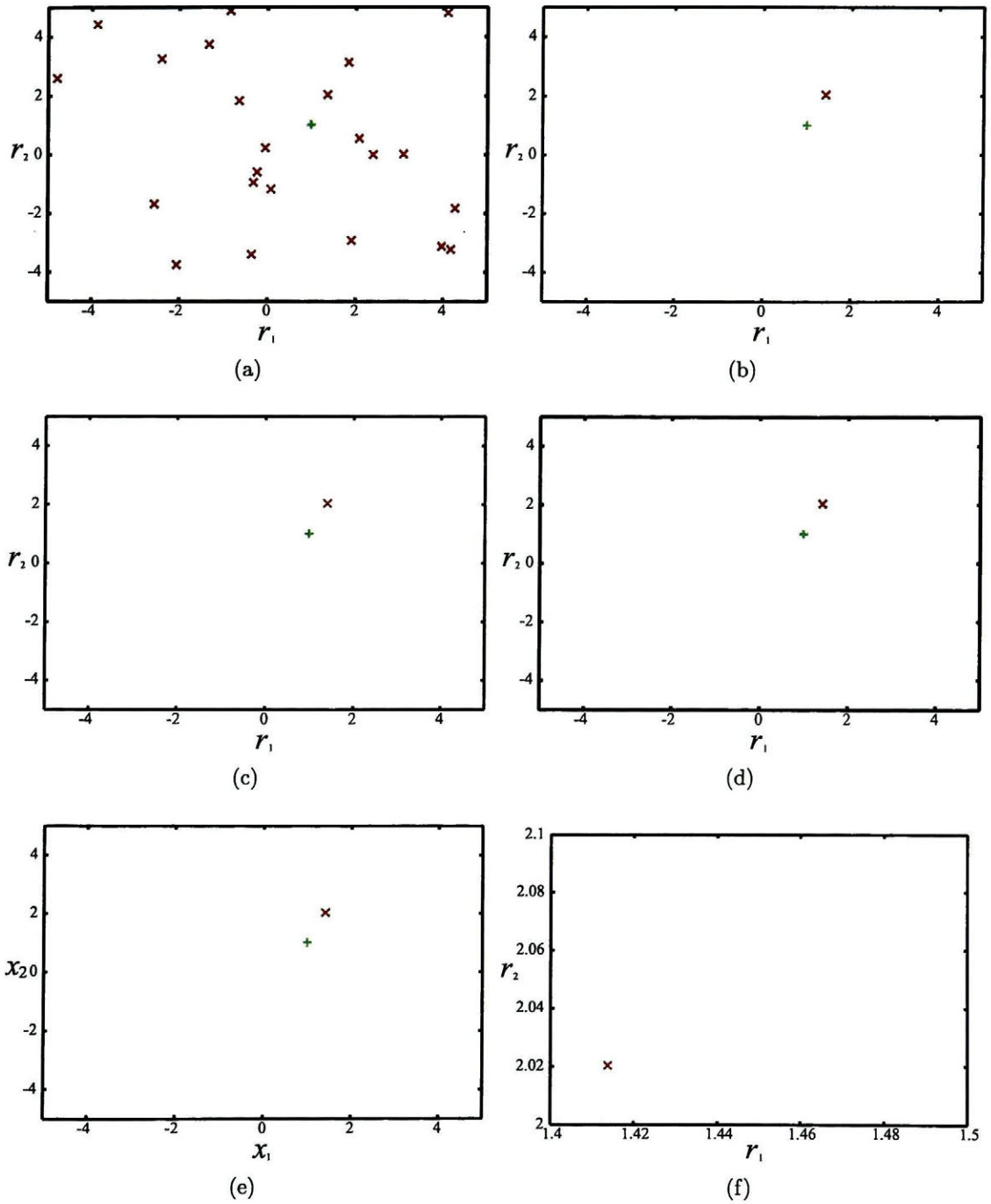


図 3.10: ルーレット選択を用いた実数値遺伝的アルゴリズムにおける探索過程. “ × ” と “ + ” は, それぞれ個体と最適解を表す. (a) 初期状態. (b)30 世代後. (c)60 世代後. (d)90 世代後. (e)100 世代後. (f)100 世代後の拡大図.

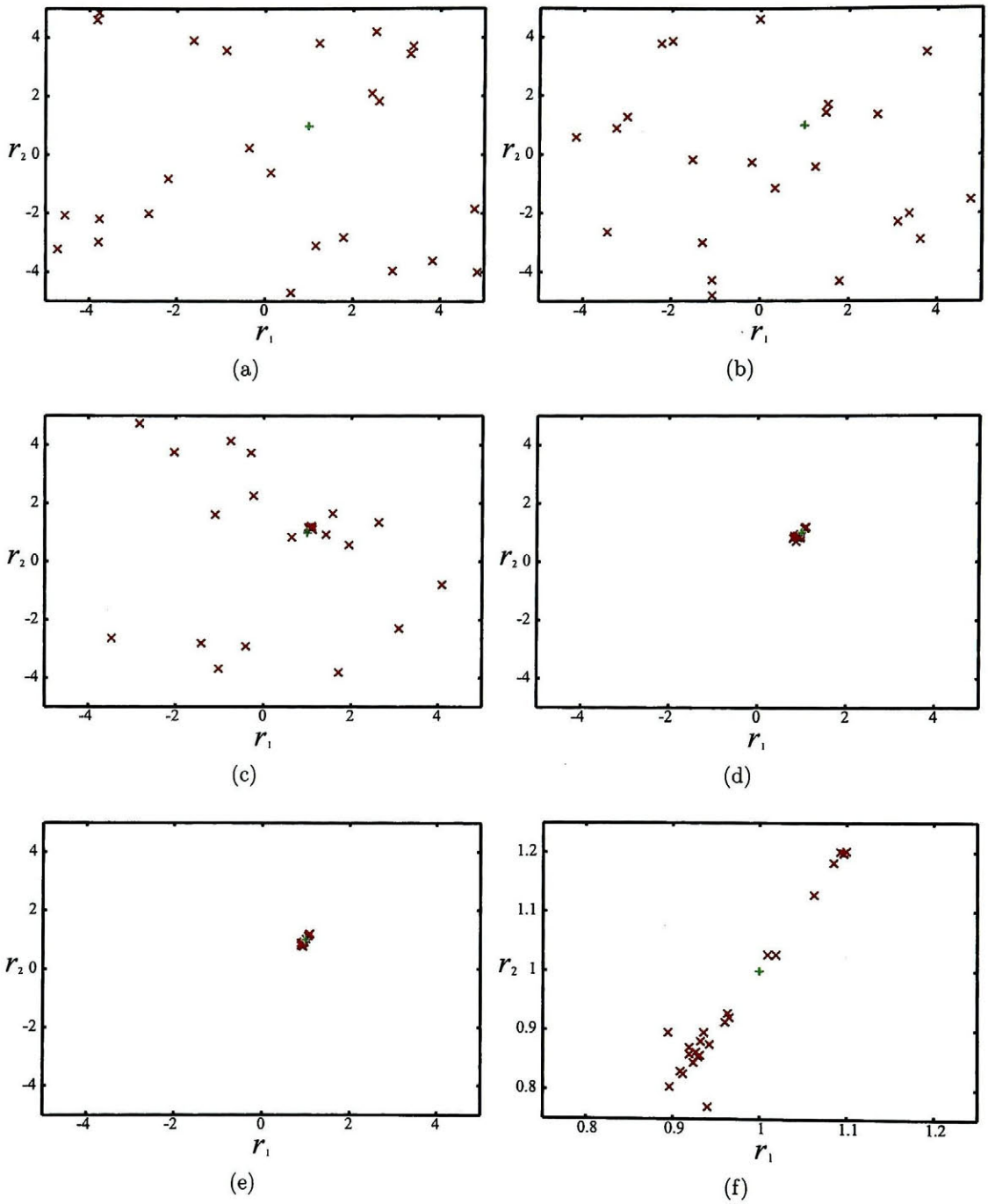


図 3.11: 提案する再生手法を用いた実数値遺伝的アルゴリズムにおける探索過程. “×”と“＋”は, それぞれ個体と最適解を表す. (a) 初期状態. (b)30 世代後. (c)60 世代後. (d)90 世代後. (e)100 世代後. (f)100 世代後の拡大図.

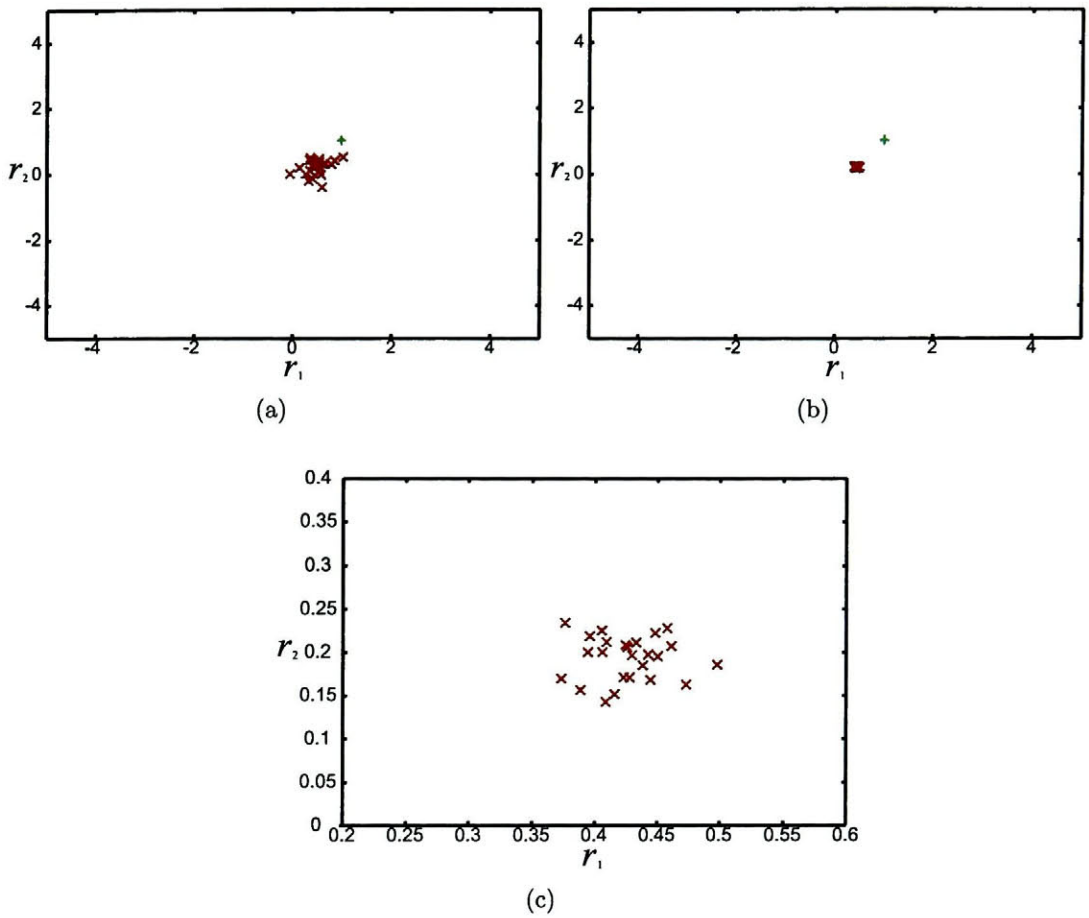


図 3.12: 乱数によって多様性を維持するルーレット選択を用いた実数値遺伝的アルゴリズムにおける Problem 1 の探索過程. (a) 一様分布 $UD[-0.5:0.5]$ を加えた場合における 100 世代後. (b) 一様分布 $UD[-0.05:0.05]$ を加えた場合における 100 世代後. (c) 一様分布 $UD[-0.05:0.05]$ を加えた場合における 100 世代後の拡大図.

可能となる.

3.3.3 Quadratic With Noise 関数への適用

本小節のシミュレーションでは，以下の連続変数最適化問題を解く.

Problem 2 Maximize $-\left\{\sum_{k=1}^{30} k{r_k}^4 + GAUSS(0,1)\right\}$, $-1.27 \leq r_k \leq 1.28$

この関数は，Problem 1 と同様に，DeJong が提案したベンチマーク関数のひとつで，Quadratic With Noise 関数と呼ばれる高次元の単峰性関数であり，局所的な収束性に関して，ノイズに対する探索のロバスト性を検証する連続変数最適化問題である．Quadratic With Noise 関数の概形を

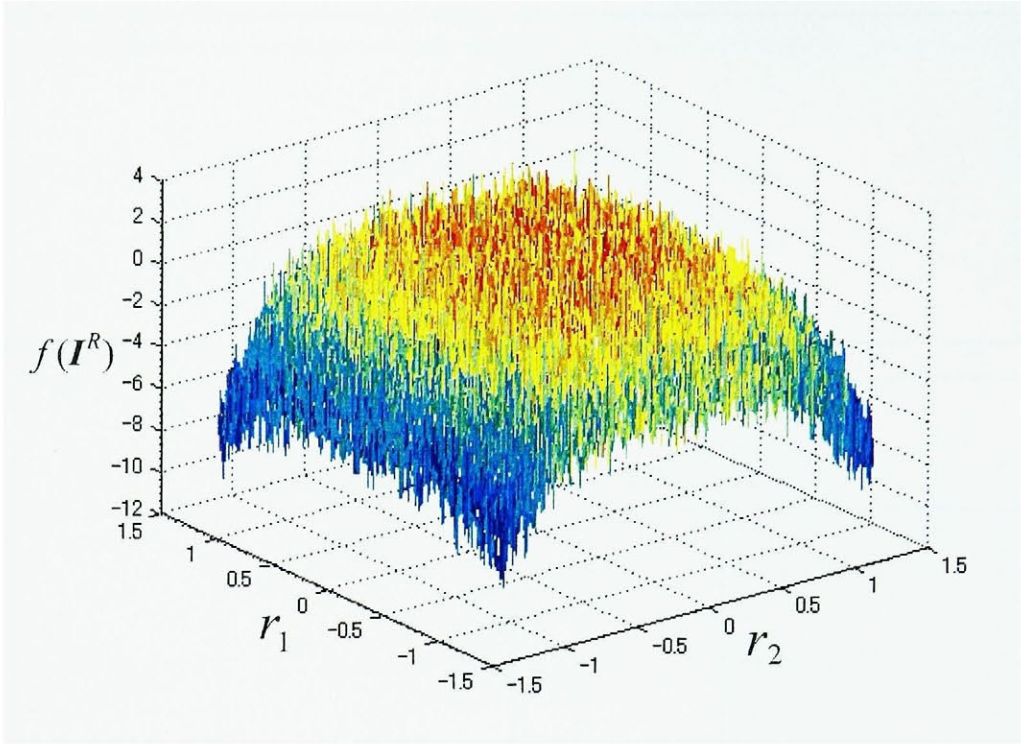


図 3.13: Problem 2: Quadratic With Noise 関数の概形.

図 3.13 に示す．この概形は，30 個の変数のうち，2 個の変数に関して描画したものである．

このシミュレーションに用いた各実数値遺伝的アルゴリズムのパラメータを，表 3.3 に示す．また，各実数値遺伝的アルゴリズムの交叉と突然変異には，それぞれ，BLX-α 交叉と一様突然変異を用いた．

前小節では，提案する再生手法とルーレット選択を用いた各実数値遺伝的アルゴリズムにおいて，世代経過における個体集団のふるまいについて比較した．しかしながら，提案する再生手法とルーレット選択は，1 世代に要する計算時間が異なる．したがって，本小節のシミュレーションでは，提案する再生手法とルーレット選択を用いた各実数値遺伝的アルゴリズムについて，探索に要する計算時間を比較する．

表 3.4 に，提案する再生手法とルーレット選択を用いた各実数値遺伝的アルゴリズムによって，Problem 2 を解くのに要した世代数と計算時間を示す．確率的要因を抑えるために，表の世代数と計算時間には，10 回試行した際の平均値を示した．表 3.4 から，提案する再生手法を用いた実数値遺伝的アルゴリズムは，ルーレット選択を用いた実数値遺伝的アルゴリズムと比較して，世代数，計算時間それぞれに対して，高速な探索が実現できていることがわかる．また，表 3.4 に示す改善比は，ルーレット選択を用いた従来の実数値遺伝的アルゴリズムによる探索時間を 1 と

表 3.3: Problem 2 を解く際に用いた実数値遺伝的アルゴリズムのパラメータ

集団のサイズ N	100
交叉を起こす確率 Pc	0.3
突然変異を起こす確率 Pm	0.05

表 3.4: Problem 2 において探索に要した世代数と計算時間

	世代数	計算時間	改善比
RWS	195796	646.1 *sec.*	1.0
Proposed	1640	289.6 *sec.*	5.1

した場合の，提案する再生手法を用いた実数値遺伝的アルゴリズムによる探索時間の倍率を表す．改善比から，提案する再生手法を用いた実数値遺伝的アルゴリズムは，ルーレット選択を用いた従来の実数値遺伝的アルゴリズムと比較して，約 5 倍の効率で探索を実現できていることがわかる．この結果から，提案する再生手法では，実数値自己組織化マップを用いることによって，個体集団の多様性を維持し，かつ，最適解に類似した高い適合度を持つ個体を多く生成できていることがいえる．これは，本研究で導入した実数値自己組織化マップによって，適合度が高い個体集団の分布を近似することができていることに起因する．したがって，提案する再生手法は、ノイズによるロバスト性を含む最適化問題に対して，ルーレット選択と比較して有効である．

3.3.4 Shekel's Foxfoles 関数への適用

本小節のシミュレーションでは，以下の連続変数最適化問題を解く．

Problem 3 Maximize $-\left\{0.02+\sum_{l=1}^{25}\frac{1}{\sum_{k=1}^{2}(r_k-a_{kl})^6}\right\}$, $-65.535 \le r_k \le 65.536$

この関数もまた，Problem 1 や Problem 2 と同様に，DeJong が提案したベンチマーク関数であり，Shekel's Foxfoles 関数と呼ばれる．Shekel's Foxfoles 関数は，2 次元の多峰性関数であり，大域的な探索能力を検証する連続変数最適化問題である．Shekel's Foxfoles 関数の概形を図 3.14 に示す．

本小節のシミュレーションに用いた各実数値遺伝的アルゴリズムのパラメータは，表 3.3 と同様である．a_{kl} は，ランダムに決定した．また，各実数値遺伝的アルゴリズムの交叉と突然変異には，それぞれ，BLX-α 交叉と一様突然変異を用いた．本小節でも前小節と同様に，提案する再生手法

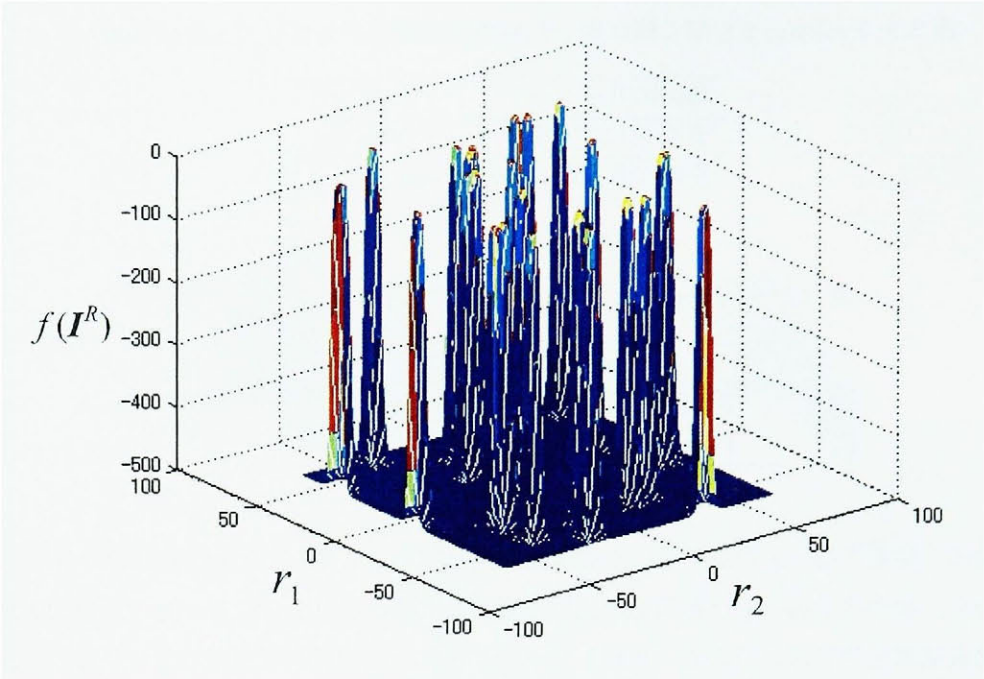


図 3.14: Problem 3: Shekel's Foxfoles 関数の概形.

とルーレット選択を用いた各実数値遺伝的アルゴリズムについて，探索に要する計算時間を比較する.

表 3.5 に，提案する再生手法とルーレット選択を用いた各実数値遺伝的アルゴリズムによって，Problem 3 を解くのに要した世代数と計算時間を示す．確率的要因を抑えるために，表の世代数と計算時間には，10 回試行した際の平均値を示した．また，10 回の試行において，a_{kl} は，同様のパラメータを用いた．表 3.5 から，提案する再生手法を用いた実数値遺伝的アルゴリズムは，ルーレット選択を用いた実数値遺伝的アルゴリズムと比較して，世代数，計算時間それぞれに対して，高速な探索が実現できていることがわかる．また，表 3.5 に示す改善比から，提案する再生手法を用いた実数値遺伝的アルゴリズムは，ルーレット選択を用いた従来の実数値遺伝的アルゴリズムと比較して，約 2 倍の効率で探索を実現できていることがわかる．この結果から，ルーレット選択を用いた従来の再生手法を用いた実数値遺伝的アルゴリズムでは，最適ではない解に収束することによって，個体の多様性を失う可能性が高いので，探索が非効率になることがわかる．しかしながら，提案する再生手法では，本研究で導入した実数値自己組織化マップを用いることによって，最適解以外への収束を避け，大域的な探索空間において個体集団の多様性を維持することが可能である．これは，本研究で導入した実数値自己組織化マップの学習則では，入力ベクトルの適合度が低い場合，結合重みベクトルがほとんど更新されないことに起因する．したがって，

表 3.5: Problem 3 において探索に要した世代数と計算時間

	世代数	計算時間	改善比
RWS	16391	73.8 *sec.*	1.0
Proposed	166	25.5 *sec.*	2.2

提案する再生手法は、大域的な探索能力を必要とする多峰性の最適化問題に対して，ルーレット選択と比較して有効である．

3.3.5 タンクの水位制御への応用

本小節では，実数値遺伝的アルゴリズムを用いて，タンク内の水位を制御する PID コントローラのパラメータを設計する．本シミュレーションでは，図 3.15 に示すように，底に放出部をもつタンクを考える．放出部から放出される 1 秒あたりの水量 $q^o[m^3/s]$ は，タンク内の水位 $h[m]$ によって決まり，放出部の断面積を $S[m^2]$，重力加速度を $g[m/s^2]$ とすると，放出部からの水速がベルヌーイの定理により $\sqrt{2gh}$ で与えられることから，次式で表される．

$$q^o = S\sqrt{2gh} \tag{3.4}$$

したがって，タンクの断面積を $A[m^2]$，タンクへ流れ込む 1 秒あたりの水量を $q[m^3/s]$ とすると，タンク内の水位の時間変化は次式により表される．

$$\frac{dh}{dt} = \frac{1}{A}(q - S\sqrt{2gh}) \tag{3.5}$$

タンク系では，流れ込む水量 q を調節することによって，タンク内の水位 h を目標値 h^t に近づけることを目的とする．水量の調節は，PID コントローラによって行われる．PID コントローラによるタンクの水位制御系は，図 3.16 に示すように，現在の水位と目標水位の誤差とその時間変化，積分値をもとに，次時刻の流入量 q^* を操作する．PID コントローラによる操作量 $\dot{q^*}$ は，次式で与えられる．

$$\dot{q^*} = K_P\left(\dot{e} + \frac{1}{T_I}e + T_D\ddot{e}\right) \tag{3.6}$$

ここで，K_P，T_I，T_D は，それぞれ PID コントローラの比例，積分，微分ゲインである．また，e は目標水位との誤差であり，次式で表される．

$$e = h^t - h \tag{3.7}$$

また，(3.5) 式はルンゲ・クッタ法により

$$h(t+1) = h(t) + n\frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \tag{3.8}$$

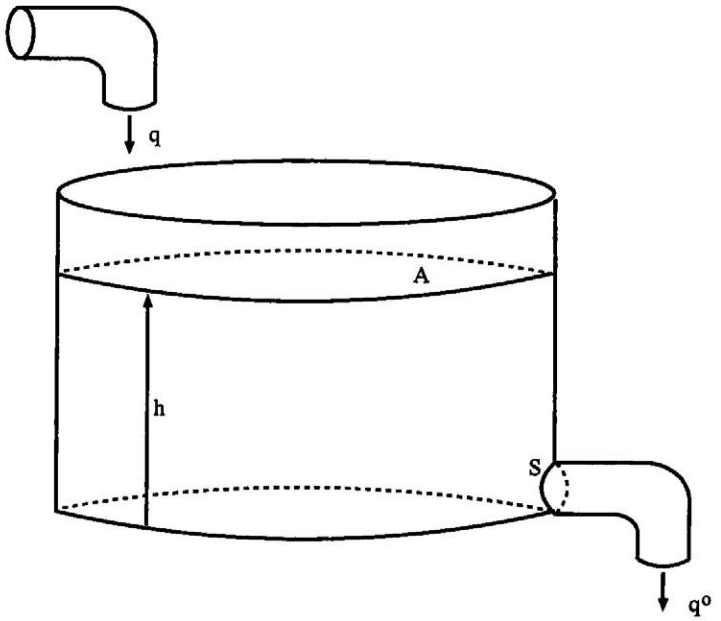


図 3.15: タンクの概形.

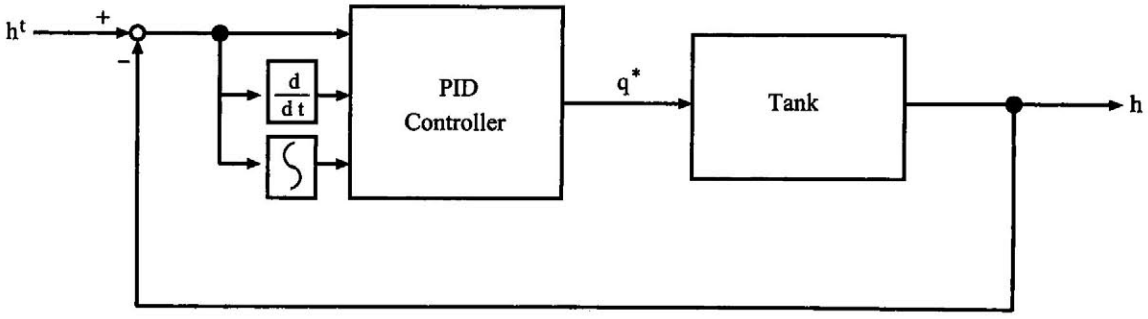


図 3.16: タンクの水位制御系の概要.

と表される [40]. ここで,

$$\begin{aligned} k_1 &= \frac{1}{A}(q - S\sqrt{2gh(t)}) \\ k_2 &= \frac{1}{A}(q - S\sqrt{2g\{h(t) + nk_1/2\}}) \\ k_3 &= \frac{1}{A}(q - S\sqrt{2g\{h(t) + nk_2/2\}}) \\ k_4 &= \frac{1}{A}(q - S\sqrt{2g\{h(t) + nk_3\}}) \end{aligned} \tag{3.9}$$

であり, n は刻み幅である. また, (3.6) 式にオイラー法を適用することにより, PID コントローラの操作量 $q^*(t+1)$ は,

$$q^*(t+1) = q^*(t) + K_P\left[\left(1 + \frac{T}{T_I} + \frac{T_D}{T}\right)e(t) - \left(1 + 2\frac{T_D}{T}\right)e(t-1) + \frac{T_D}{T}e(t-2)\right] \tag{3.10}$$

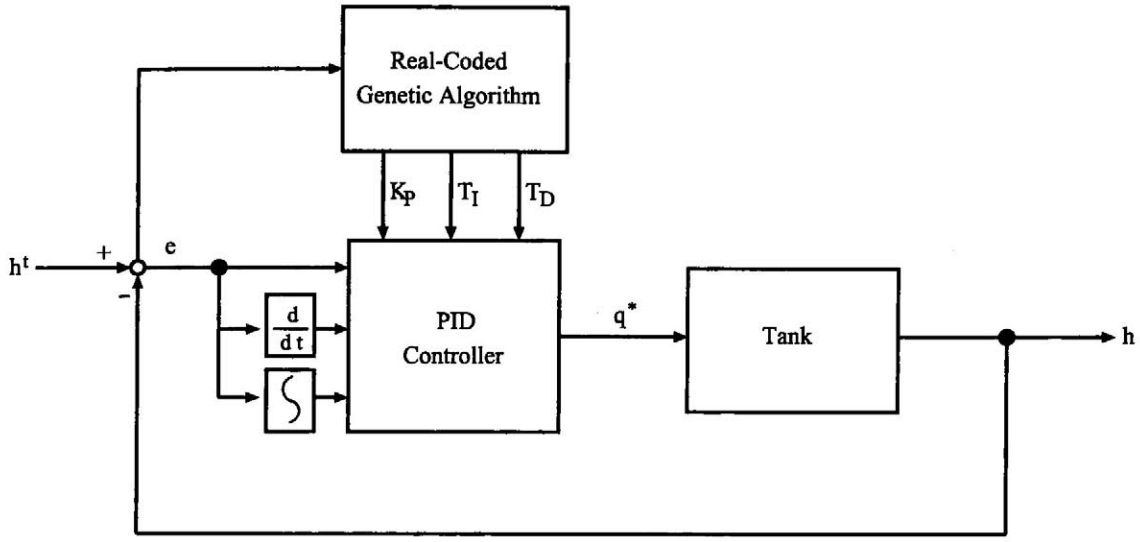


図 3.17: 実数値遺伝的アルゴリズムを用いた，PID コントローラのゲイン調整システム.

と表される [41]. ここで，T は PID コントローラの状態を算出するステップである.

一般的に，K_P，T_I，T_D は，限界感度法などを用いて熟練者が設計を行う [42]. しかしながら，従来の設計法は，制御対象やコントローラに関する高度な知識を必要とし，人的負担も大きい. この問題を解決するために，本研究では，実数値遺伝的アルゴリズムを用いて PID コントローラの K_P，T_I，T_D を調整する. 実数値遺伝的アルゴリズムを用いた PID コントローラのパラメータ調整システムの概要を図 3.17 に示す. 本シミュレーションでは，実数値遺伝的アルゴリズムの個体として，K_P，T_I，T_D を要素とする実数値ベクトル $\boldsymbol{I}^R$ を用いる. 実数値遺伝的アルゴリズムでの評価関数 $f_{\boldsymbol{I}^R}$ は，個体 $\boldsymbol{I}^R$ の要素を採用した PID コントローラを用いてタンク系の制御を行い，立ち上がりと立ち下がりオーバーシュートの行き過ぎ量 $e^{UO}_{\boldsymbol{I}^R}$，$e^{DO}_{\boldsymbol{I}^R}$ とステップ応答の定常誤差 $e^{UE}_{\boldsymbol{I}^R}$，$e^{DE}_{\boldsymbol{I}^R}$ の絶対値により設計する. 評価関数 $f_{\boldsymbol{I}^R}$ は，以下のように決定した.

$$f_{\boldsymbol{I}^R} = \frac{1}{1 + \left(|e^{UO}_{\boldsymbol{I}^R}| + |e^{DO}_{\boldsymbol{I}^R}| + |e^{UE}_{\boldsymbol{I}^R}| + |e^{DE}_{\boldsymbol{I}^R}| \right)} \tag{3.11}$$

すなわち，(3.11) 式は，立ち上がりと立ち下がりのステップ応答に関して，オーバーシュートの行き過ぎ量と定常誤差が少ないほど，適合度が高くなるということを示している.

本シミュレーションで用いたタンク系と実数値遺伝的アルゴリズムのパラメータをそれぞれ，表 3.6，表 3.7 に示す. また，探索を行う際に，個体の各要素を正規化した. 図 3.18 に熟練者と提案する再生手法を採用した実数値遺伝的アルゴリズムで設計した各 PID コントローラで制御した結果を示す. 図 3.18(a)，(b) から，実数値遺伝的アルゴリズムによって設計した PID コントローラは，熟練者が設計したコンロトーラとほぼ同じ性能を示していることがわかる. このことから，実

表 3.6: 本シミュレーションに用いたタンク系のパラメータ

タンクの断面積 A $[m^2]$	20
タンク底部の放出部の断面積 S $[m^2]$	0.5
タンクへの流入水量の初期値 $q(0)$ $[m^3/s]$	0
タンクへの流入水量の最大値 q_{max} $[m^3/s]$	0.5
タンク内の水位の初期値 $h(0)$ $[m]$	1
刻み幅 n $[s]$	0.05
ステップ数の最終値 t_{max} $[s]$	500

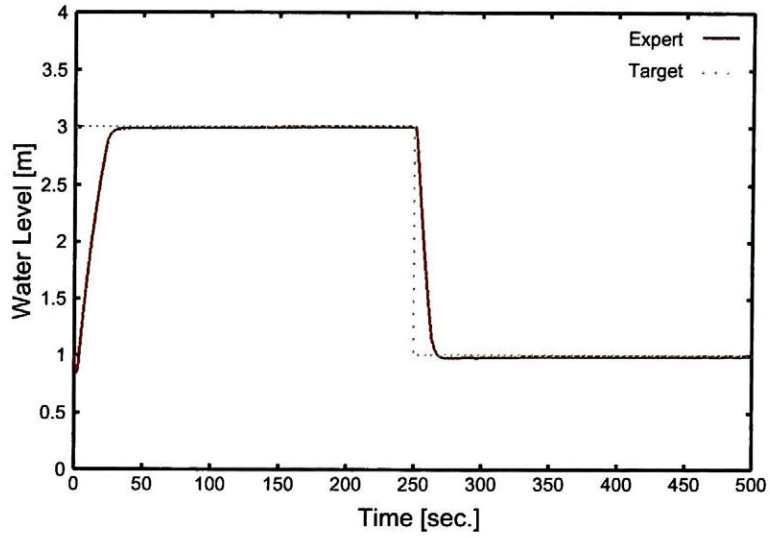
表 3.7: 本シミュレーションに用いた実数値遺伝的アルゴリズムのパラメータ

集団のサイズ N	4
交叉を起こす確率 Pc	0.3
突然変異を起こす確率 Pm	0.1

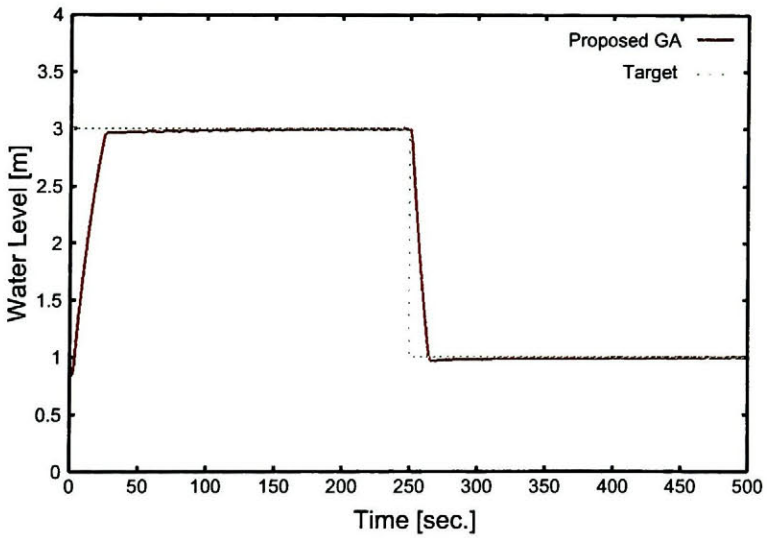
数値遺伝的アルゴリズムでは，オーバーシュートの行き過ぎ量と定常誤差のみを用いた評価によって，熟練者とほぼ同等の PID コントローラの設計ができることがわかる．表 3.8 に，双方で求めた各パラメータを示す．

さらに図 3.18 に示すステップ応答をもとに設計した表 3.8 のパラメータを用いて，設計に用いていない目標値に対する制御を行った．図 3.19 に設計に用いていない目標値への制御結果を示す．図 3.19 から，提案する再生手法を採用した実数値遺伝的アルゴリズムで求めたパラメータは，設計に用いていない目標値に関しても熟練者が決定したものと同等の性能を発揮できていることがわかる．

また，従来の実数値遺伝的アルゴリズムにおいても，同等の性能は発揮できる．そこで，探索に要する世代数と計算時間に関して，本研究で提案する再生手法を用いた実数値遺伝的アルゴリズムとルーレット選択を用いた実数値遺伝的アルゴリズムによる比較を行う．双方の実数値遺伝的アルゴリズムにおいて探索に要した世代数と計算時間を表 3.9 に示す．表 3.9 から，提案する再生手法を用いた実数値遺伝的アルゴリズムは，ルーレット選択を用いた従来の実数値遺伝的アルゴリズムと比較して，少ない世代数，かつ，計算時間で妥当解を見つけることができていることがわかる．このことから，提案する再生手法は，従来の実数値遺伝的アルゴリズムと比較して効率のよい探索が実現できているといえる．しかしながら，実数値遺伝的アルゴリズムで PID コントローラのパラメータ設計を行う場合，評価関数の設計に関してある程度の知識を必要とする．また，計算機シミュレーションを行うので，制御対象の物理モデルが既知である必要があるという

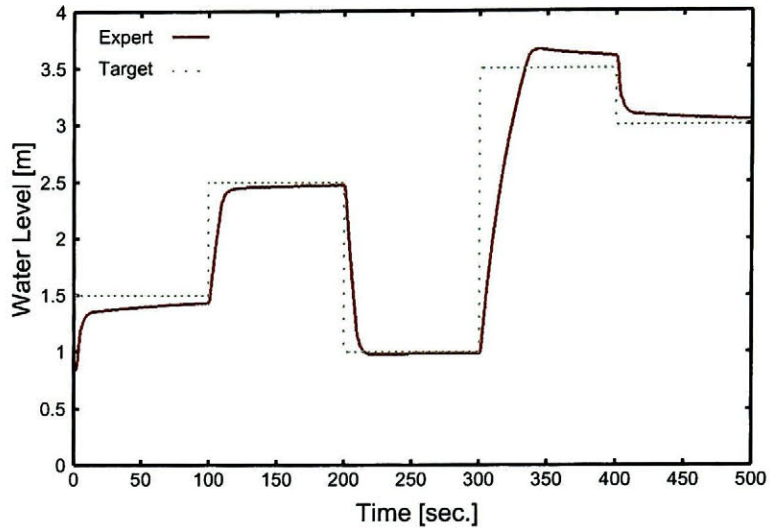


(a)

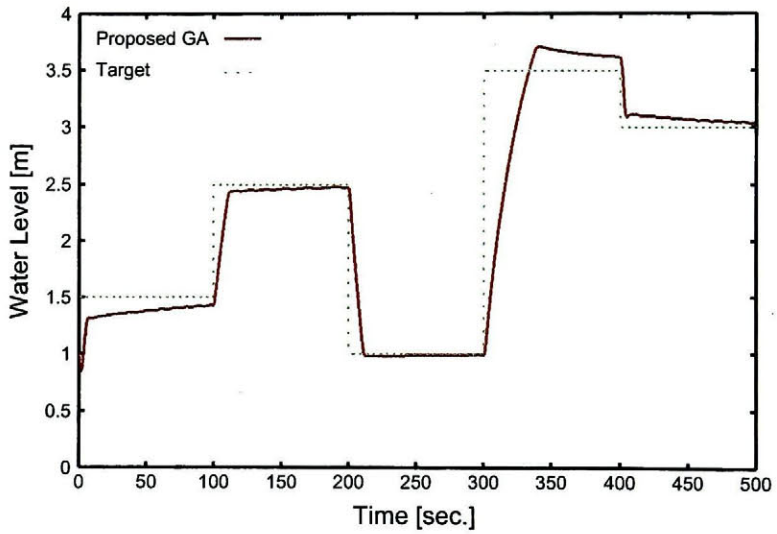


(b)

図 3.18: 熟練者と提案する再生手法を採用した実数値遺伝的アルゴリズムを用いて設計した PID コンロトーラを用いた制御結果. (a) 熟練者が設計した PID コントローラ. (b) 提案する再生手法を採用した実数値遺伝的アルゴリズムを用いて設計した PID コントローラ.



(a)



(b)

図 3.19: 熟練者と提案する再生手法を採用した実数値遺伝的アルゴリズムを用いて設計した PID コンロトーラを用いた制御結果（テスト）. (a) 熟練者が設計した PID コントローラ. (b) 提案する再生手法を採用した実数値遺伝的アルゴリズムを用いて設計した PID コントローラ.

表 3.8: 設計した PID コントローラの各パラメータ

	K_P	T_D	T_I
熟練者	14.5	1.4	108.0
実数値遺伝的アルゴリズム	11.46	0.0	88.91

表 3.9: 探索に要した世代数と計算時間の比較

	世代数	計算時間	改善比
RWS	3168	14949.8 *sec.*	1.0
Prposed	109	2168.6 *sec.*	6.9

ことに留意すべきである.

3.4 バイナリ自己組織化マップを再生に用いたバイナリ遺伝的アルゴリズム

3.4.1 探索アルゴリズム

適合度に基づいて結合重みベクトルを更新するバイナリ自己組織化マップを再生として用いたバイナリ遺伝的アルゴリズムの探索アルゴリズムは，実数値自己組織化マップを再生に用いた実数値遺伝的アルゴリズムの場合と同様の流れに従って行なわれる．しかしながら，ベクトルの表現方法の違いから，結合重みベクトルの初期化，ベクトル間の距離尺度，結合重みベクトルの更新方法が実数値自己組織化マップの場合と異なる．提案する再生手法を用いたバイナリ遺伝的アルゴリズムの探索アルゴリズムにおいて，実数値遺伝的アルゴリズムの探索アルゴリズムと異なる部分を以下に示す.

Step G-0: 乱数を用いて個体集団 $\boldsymbol{I}^R$ の初期化を行い，各個体の適合度を算出する．次に，従来のバイナリ自己組織化マップに，入力ベクトル空間からランダムに選んだバイナリ入力ベクトルを提示し，結合重みベクトルの更新を行う．この操作を繰り返すことにより，結合重みベクトルは，バイナリ結合重みベクトル空間内において，一様に配置される.

Step G-1: 個体集団をバイナリ入力ベクトル集合として ($\boldsymbol{I}^B = \boldsymbol{x}^B$)，各入力ベクトルの適合度をもとに，再生アルゴリズムによって次世代に残す個体を決定する.

Step S-2: 入力ベクトルとのユークリッド距離が最小となる結合重みベクトルをもつ競合層ユニット BMU を次式で決定し，勝者ユニットとする．

$$BMU = \arg\min_j \{\sum_{j=1}^{M} \{(x_i \oplus w_{ji})\} \tag{3.12}$$

Step S-3: 全ての結合重みベクトルを更新する．更新は，入力ベクトルと結合重みベクトルの対応する要素に対して XOR をとり，XOR が 1 の要素を $\alpha_{BSOM}(t)$ 個，入力ベクトルに揃えることで実現される．ここで，$\alpha_{BSOM}(t)$ は，次式によって表される．

$$\alpha_{BSOM} = \lfloor \alpha_S \cdot f_{\boldsymbol{x}^B} \cdot h(f_{\boldsymbol{x}^B}, d_j) \cdot (1 - f_{\boldsymbol{w}_j^B(t)}) \rfloor \tag{3.13}$$

ここで，$\lfloor\ \rfloor$ は，ガウスの記号を表す．α_S は，入力ベクトルの要素数によって決定されるパラメータであり，更新する要素数の上限をもとに決定される．また，$H(d_j, f_{\boldsymbol{x}^B})$ は，次式で定義される．

$$H(d_j, f_{\boldsymbol{x}^B}) = \exp\left(-\frac{{d_j}^2}{2{f_{\boldsymbol{x}^B}}^2}\right) \tag{3.14}$$

Step S-5: 学習後の結合重みベクトル集合を再生後の個体集団とする．
$(\boldsymbol{w}_j^B(t+1) = \boldsymbol{I}^B)$

(3.14) 式は，(3.3) 式と同様に，勝者ユニットに近い競合層ユニットほど，その結合重みベクトルの更新要素数が多く，入力ベクトルの適合度が高いほど，多くの競合層ユニットの結合重みベクトルを更新することを意味する．また，(3.13) 式は，(3.14) 式に加えて，入力ベクトルの適合度が高いほど結合重みベクトルの更新要素数を多くし，更新前の結合重みベクトルの適合度が高いほど更新要素数を少なくすることを意味する．(3.13) 式により，適合度が低い結合重みベクトルは，適合度が高い入力ベクトルに近づくように更新され，適合度が高い結合重みベクトルはほとんど更新されない．したがって，学習終了後の結合重みベクトルは，入力ベクトル集合内の適合度が高い入力ベクトルの周辺に密に配置される．つまり，本節で提案するバイナリ自己組織化マップは，高い適合度をもつバイナリ入力ベクトル集合について，その分布を近似することが可能になり，前節で導入した実数値自己組織化マップと同様の機能をもつ．

3.4.2 0-1 ナップサック問題への適用

提案手法の有効性を示すために，組み合わせ最適化問題の一つである 0-1 ナップサック問題に適用する．一般的なナップサック問題では，ナップサックに詰めこむことができる荷物の最大合計重量を制約として，各荷物固有の価値の合計を最大化することが目的である．本節のシミュレー

表 3.10: 本シミュレーションに用いたバイナリ遺伝的アルゴリズムのパラメータ

集団のサイズ N	9
交叉を起こす確率 Pc	0.3
突然変異を起こす確率 Pm	0.01

表 3.11: 0-1 ナップサック問題に用いた各荷物の価値と質量（荷物数 20 個）

荷物番号	1	2	3	4	5	6	7	8	9	10
荷物の価値	8	3	9	2	3	8	2	9	2	2
荷物の質量	5	8	2	3	6	1	3	9	5	7
荷物番号	11	12	13	14	15	16	17	18	19	20
荷物の価値	3	4	5	2	5	7	2	3	4	8
荷物の質量	4	6	4	3	8	9	2	4	1	12

ションでは，荷物の個数が 20 個と 50 個の場合について，提案するバイナリ自己組織化マップによる再生手法とルーレット選択を用いた双方のバイナリ遺伝的アルゴリズムについて，集団の適合度の最大値が一定以上になるまでの世代数と計算時間を比較する．20 個のナップサック問題は，以下のように示される．

Problem 4: Maximize $\sum_{k=1}^{20} v_k b_k$, where $b_k \in \{0,1\}$, subject to $\sum_{k=1}^{20} u_k b_k \leq L$

ここで，b_k は ‘1’ か ‘0’ をとり，それぞれ，k 番目の荷物をナップサックに詰めるか詰めないかという行動を表す．また，v_k，u_k は，それぞれ，k 番目の荷物の価値と重量を表す．L は，ナップサックに詰めこむことができる最大の合計重量を表す．

シミュレーションに用いたバイナリ遺伝的アルゴリズムのパラメータを，表 3.10 に示す．交叉と突然変異には，双方のバイナリ遺伝的アルゴリズムにおいて，それぞれ，一点交叉と一様突然変異を用いた．また，提案する再生オペレータの学習係数 α_s は，荷物の個数が 20 の場合は 10，50 の場合は 40 とした．各荷物の価値と重量は，それぞれ表 3.11，3.12 に示すように決定した．

表 3.13 に，荷物の個数が 20 個の場合の，探索に要した世代数と計算時間を示す．様々な確率的影響を抑えるために，10 回試行した際の平均値を示した．なお，10 回の試行において，各荷物の価値と重量は同じものを用いた．表 3.13 から，提案手法を用いたバイナリ遺伝的アルゴリズムは，従来の再生手法を用いたバイナリ遺伝的アルゴリズムと比較して，少ない世代数，かつ短時間で解を求めることができていることがわかる．また，表 3.13 に示す改善比から，提案する再生手法を用いたバイナリ遺伝的アルゴリズムは，ルーレット選択を用いた従来のバイナリ遺伝的アルゴ

表 3.12: 0-1 ナップサック問題に用いた各荷物の価値と質量（荷物数 50 個）

荷物番号	1	2	3	4	5	6	7	8	9	10
荷物の価値	20	21	35	33	40	28	4	14	40	7
荷物の質量	9	7	7	9	3	7	5	7	5	8
荷物番号	11	12	13	14	15	16	17	18	19	20
荷物の価値	2	31	36	22	4	28	28	25	2	21
荷物の質量	8	7	9	10	3	2	5	7	8	4
荷物番号	21	22	23	24	25	26	27	28	29	30
荷物の価値	15	24	12	18	21	28	36	14	22	28
荷物の質量	10	9	4	10	2	7	9	7	10	2
荷物番号	31	32	33	34	35	36	37	38	39	40
荷物の価値	21	33	40	14	40	7	2	36	22	4
荷物の質量	7	9	3	7	5	8	8	9	10	3
荷物番号	41	42	43	44	45	46	47	48	49	50
荷物の価値	28	25	3	15	24	12	21	28	22	21
荷物の質量	5	8	7	10	9	4	2	7	10	2

表 3.13: 探索に要した世代数と計算時間の比較 (荷物 20 個)

	世代数	計算時間	改善比
RWS	5444	0.544 *sec.*	1.0
Proposed	673	0.269 *sec.*	2.0

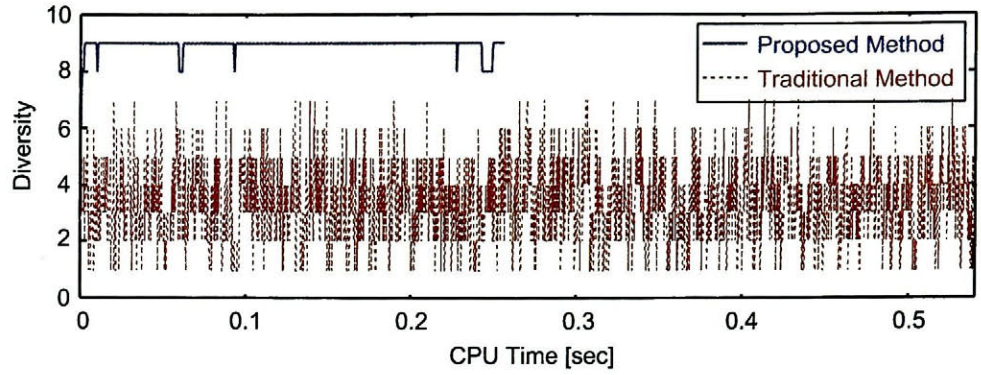
リズムと比較して，約 2 倍の効率で探索を実現できていることがわかる．

図 3.20 に，荷物の個数が 20 個の場合の，提案する再生手法とルーレット選択による従来の再生手法を採用した場合の多様性と個体集団の適合度の平均値，最大値を示す．図 3.20(a) は，従来手法では，世代が進行すると急激に多様性が減少するのに対して，提案手法を用いた再生では，世代が減少しても多様性が減少しないことを示している．これは，図 3.20(b) に示すように，双方の個体集団の適合度の平均値に差がないことから，提案手法を用いた再生では，適合度が高く，かつ多様な個体を作りだすことができていることを意味する．その結果，交叉の性能を充分に引き出すことができており，局所的な探索の効果が上がったことで，図 3.20(c) に示すように，個体集団の最大適合度が上昇する．したがって，提案手法を用いた再生は，従来の選択方法と比較して，効率的な探索を行っているといえる．

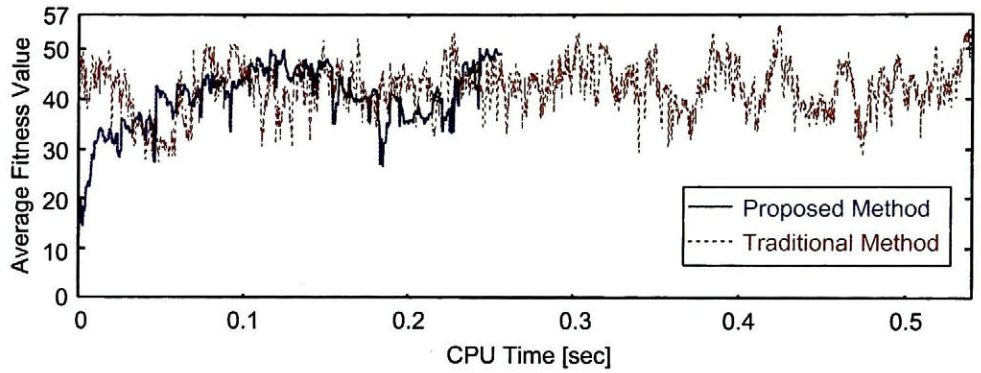
表 3.14: 探索に要した世代数と計算時間の比較 (荷物 50 個)

	世代数	計算時間	改善比
RWS	10384	3.115 *sec.*	1.0
Proposed	506	0.404 *sec.*	7.1

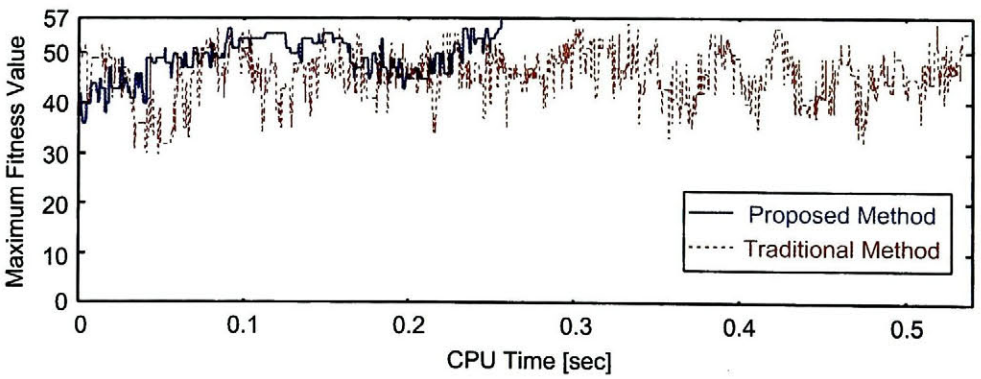
表 3.14 に，荷物の個数が 50 個の場合の，探索に要した世代数と計算時間を示す．荷物が 20 個の場合と同様に，10 回試行した際の平均値を示した．表 3.14 から，提案手法を用いたバイナリ遺伝的アルゴリズムは，従来の再生手法を用いたバイナリ遺伝的アルゴリズムと比較して，少ない世代数，かつ短時間で解を求めることができている．さらに，提案手法を用いたバイナリ遺伝的アルゴリズムでは，荷物の個数が増えた場合，探索に必要とした時間がさほど増加しないことがわかる．これは，表 3.14 に示す改善比からも明らかであり，提案する再生手法を用いた実数値遺伝的アルゴリズムは，ルーレット選択を用いた従来の実数値遺伝的アルゴリズムと比較して，約 7 倍の効率で探索を実現できていることがわかる．図 3.21 に，荷物の個数が 50 個の場合の，提案手法を用いた再生と従来手法を採用した場合の多様性と個体集団の適合度の平均値，最大値を示す．図 3.21 では，図 3.20 よりも各手法の差が顕著になっており，提案手法を用いた再生は，従来手法と比較して，高い多様性を維持し，かつ高い適合度をもつ個体集団を生成できている．これらの結果から，提案する再生手法を用いたバイナリ遺伝的アルゴリズムでは，交叉の性能が十分に発揮されていることがわかる．



(a)



(b)



(c)

図 3.20: 提案手法とルーレット選択を用いた場合の，多様性と適合度の時間的推移の比較 (荷物 20 個)．(a) 多様性．(b) 個体集団の適合度の平均値．(c) 個体集団内の適合度の最大値．

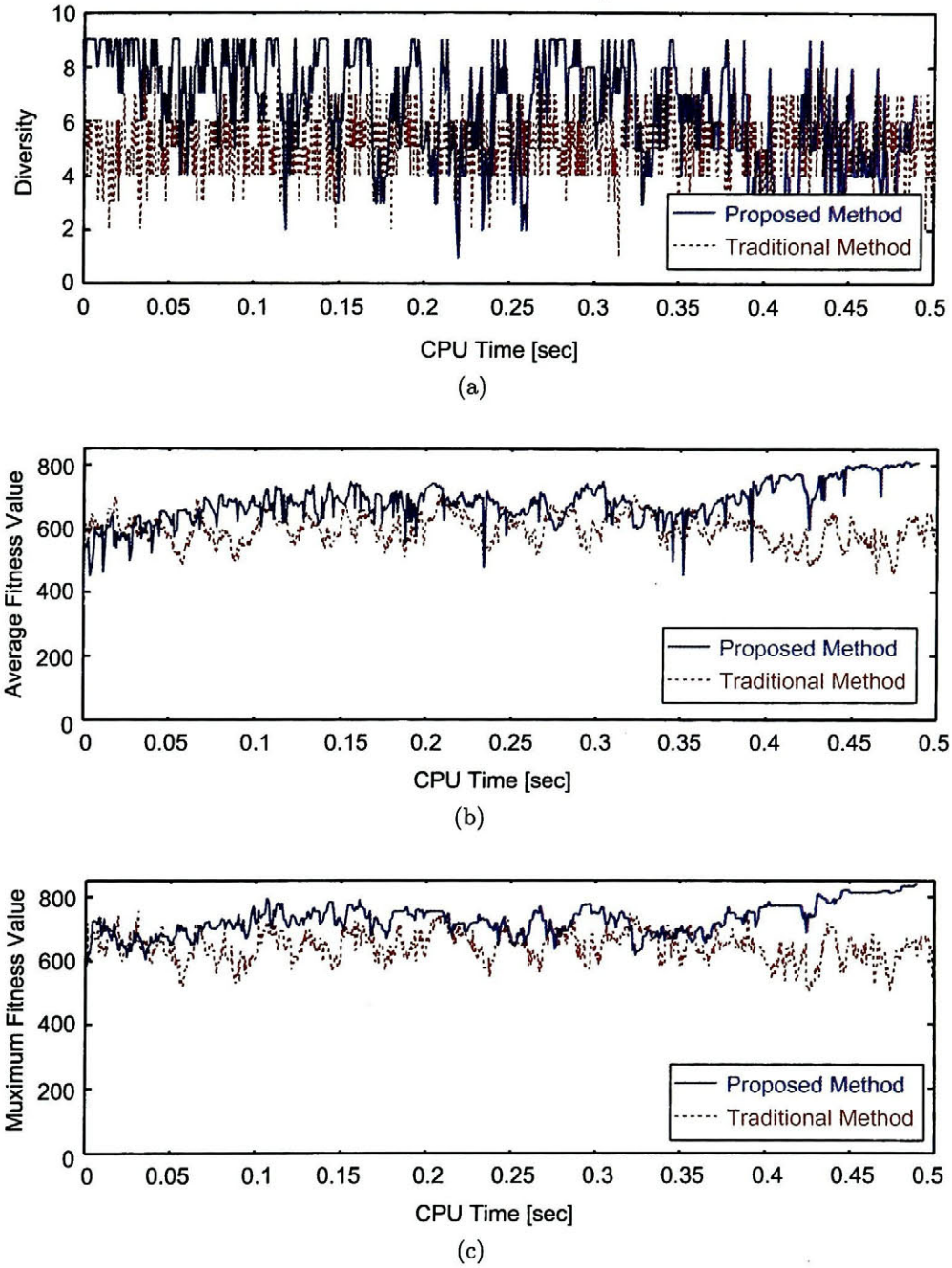


図 3.21: 提案手法とルーレット選択を用いた場合の，多様性と適合度の時間的推移の比較 (荷物 50 個). (a) 多様性. (b) 個体集団の適合度の平均値 (c) 個体集団内の適合度の最大値.

3.5 入力ベクトルの重要性に基づいたバイナリ結合重みベクトルの更新則

3.5.1 従来のバイナリ自己組織化マップの結合重みベクトル更新則の問題点

従来のバイナリ自己組織化マップにおける結合重みベクトル更新則では，結合重みベクトルの各要素を入力ベクトルに揃える際に，更新する要素の順序がランダムに決定される．しかしながら，更新する順序をランダムに決定した場合，どの要素を更新するかによって，更新後の個体における適合度の上昇の度合いが異なる．したがって，更新する要素の選び方によっては，更新後の結合重みベクトルの適合度が効率よく上昇しないので，探索が非効率になるという問題がある[43][44]．効率のよい探索を実現するには，結合重みベクトル内の各要素の重要度を考慮した更新則が望ましい．

3.5.2 要素の重要性を考慮したバイナリ結合重みベクトル更新則

本節では，バイナリ結合重みベクトルの各要素の更新順序に起因する探索効率の改善を目的として，各要素の重要性に着目した新しいバイナリ結合重みベクトル更新則を提案する．本節で提案するバイナリ結合重みベクトル更新則では，更新後の結合重みベクトルの適合度を効率よく上昇させる可能性が高い要素から順に，入力ベクトルに対して要素の値を揃えていく．各要素を更新する順序は，次式に基づいて算出される．

$$s_i = \frac{\sum_{k=1}^{K} x_{ki}^{B} f_{\boldsymbol{x}_k^B}}{\sum_{k=1}^{K} f_{\boldsymbol{x}_k^B}} \tag{3.15}$$

ここで，i，K はそれぞれ，各ベクトルの要素番号，入力ベクトルの数である．(3.15) 式は，ベクトルの各要素に対して，入力ベクトルの適合度 $f_{\boldsymbol{x}_k^B}$ に関する重み付き平均を求めることを意味し，遺伝的アルゴリズムにおけるスキーマという概念に基づく．スキーマとは，個体をベクトルとして表現した際の確定された要素を部分集合として含むテンプレートである [11]．確定したビットは ‘0’，もしくは ‘1’ で表され，確定していないビットは ‘*’ (Don't Care) で表され，スキーマ内の確定した要素の数をスキーマの次数と呼ぶ．例として，2つの個体 $(0,1,0)$，$(0,1,1)$ は，スキーマ $(0,1,*)$ に合致し，その次数は2である．各スキーマは，個体と同様に適合度を定義することが可能である．遺伝的アルゴリズムでは，次数が低く，評価値の高いスキーマが組み合わされることで，最適解に近い解が生成されるという積み木仮説 (Building Block Hypothesis) が提唱されている [11]．本研究で用いる (3.15) 式は，各要素を次数1のスキーマと捉えた場合の各スキーマの適合度に関する重心とみることができる．つまり，各要素において，高い適合度をもつ個体

表 3.15: 探索に要した世代数と計算時間

荷物数	手法	世代数	計算時間 (*sec.*)	改善比
20	RWS	5444	0.298	1.0
	Previous	673	0.123	2.4
	Proposed	570	0.112	2.7

集団に‘0’が多ければS_iは‘0’に，‘1’が多ければS_iは‘1’に近づく．したがって，S_iが‘0’または‘1’に近いということは，高い適合度をもつ個体を生成するために，その要素が‘0’または‘1’である確定性 (重要性) が高いということを示している．また，S_iが‘0.5’に近いほど，その要素は適合度の変動に関して，さほど確定的でないことを意味する．提案する結合重みベクトル更新則では，S_iが‘0’もしくは‘1’に近い要素から順に，結合重みベクトルを更新していく．高い適合度をもつ個体を生成するために確定的な要素から更新していくことで，適合度が高い個体を効率よく生成することが可能となる．

3.5.3 0-1 ナップサック問題への適用

提案手法の有効性を示すために，組み合わせ最適化問題の一つである 0-1 ナップサック問題に適用する．本節のシミュレーションでは，前節と同様に荷物数が 20 個と 50 個の各場合に関して，提案する結合重みベクトル更新則を用いたバイナリ自己組織化マップによる再生手法とルーレット選択，従来の結合重みベクトル更新則を用いたバイナリ自己組織化マップによる再生手法を適用した各バイナリ遺伝的アルゴリズムで探索し，個体集団の適合度の最大値が一定以上になるまでの世代数と計算時間を比較する．

シミュレーションに用いたバイナリ遺伝的アルゴリズムのパラメータと，各荷物の価値と重要は，前節と同様のものを用いた．

表 3.15 に，探索に要した世代数と計算時間を示す．様々な確率的影響を抑えるために，10 回試行した際の平均値を示した．なお，10 回の試行において，各荷物の価値と重量は同じものを用いた．表 3.15 から，提案手法を用いたバイナリ遺伝的アルゴリズムは，従来の結合重みベクトル更新則による再生手法を用いたバイナリ遺伝的アルゴリズムやルーレット選択を採用したバイナリ遺伝的アルゴリズムと比較して，少ない世代数，かつ若干ではあるが，短時間で解を求めることができていることがわかる．

さらに，荷物数を 50，100，150 の場合について，探索に要する世代数と計算時間に関する比較を行う．100 個，150 個の荷物の価値と重量に関しては，50 個の荷物の重量と価値を重複するこ

表 3.16: 探索に要した世代数と計算時間

荷物数	手法	世代数	計算時間 (*sec.*)
50	RWS	1594	0.130
	Previous	135	0.031
	Proposed	41	0.011
100	RWS	148525	18.11
	Previous	5368	1.718
	Proposed	2504	1.177
150	RWS	10080858	1622.01
	Previous	486172	173.95
	Proposed	46706	25.61

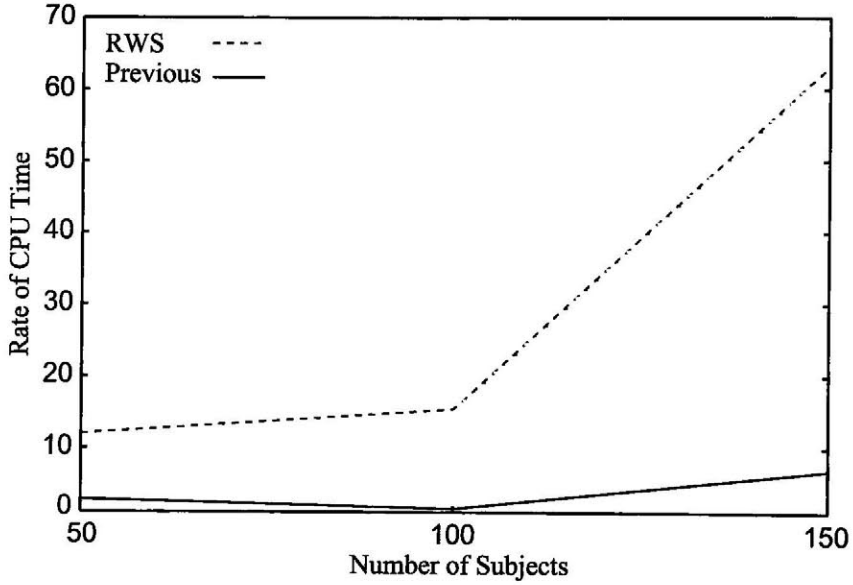


図 3.22: 提案する再生手法を用いたバイナリ遺伝的アルゴリズムでの計算時間を 1 とした場合の従来の再生手法とルーレット選択を用いたバイナリ遺伝的アルゴリズムの計算時間の比率.

とによって作成した．また，終了条件に関しても，各荷物数の場合において，ほぼ一定となるように設定した．表 3.16 に，探索に要した世代数と計算時間を示す．表 3.16 から，各荷物数において，提案手法を用いたバイナリ遺伝的アルゴリズムは，従来の結合重みベクトル更新則による再生手法やルーレット選択を用いたバイナリ遺伝的アルゴリズムと比較して，少ない世代数，かつ，計算時間で妥当解を求めることができていることがわかる．また，図 3.22 に，各荷物数での探索において，提案するバイナリ遺伝的アルゴリズムによる計算時間を 1 とした場合の，従来のバイナリ結合重みベクトル更新則による再生手法とルーレット選択を用いた双方のバイナリ遺伝的ア

ルゴリズムの計算時間の比率を示す．図 3.22 から，ルーレット選択を用いたバイナリ遺伝的アルゴリズムでは，荷物数の増加に伴って計算時間も著しく増加することがわかる．また，従来の結合重みベクトル更新則を用いたバイナリ遺伝的アルゴリズムにおいても，ルーレット選択を用いたバイナリ遺伝的アルゴリズムによる探索と比較すると，さほど計算時間の増大は見られないものの，提案するバイナリ遺伝的アルゴリズムと比較して，数倍程度の増加が見られる．これらのことから，荷物数が多くなるほど，提案するバイナリ遺伝的アルゴリズムは効率よく探索できている．つまり，提案手法は，探索空間が大きくなるほどその有効性を示すことができるといえる．

3.6 おわりに

本章では，遺伝的アルゴリズムにおける従来の再生手法における，個体集団の多様性減少による探索効率の悪化という問題点を解決するために，自己組織化マップを用いた再生手法を提案した．提案する再生手法では，再生後の望ましい個体集団を生成するために自己組織化マップの特長のひとつである，入力ベクトル集合の分布近似能力に注目し，従来とは異なる自己組織化マップを採用している．本研究で採用した自己組織化マップでは，以下の点が従来の自己組織化マップと異なる．

(1) 入力ベクトル数と競合層ユニット数が等しい．

(2) 入力ベクトル集合内の適合度が高い入力ベクトルの分布を近似する．

(1) は，入力ベクトルを再生前の個体集団，学習後の結合重みベクトルを再生後の個体集団として学習を行うことに起因し，(2) は，適合度が高い個体集団の分布を近似することに起因する．また，(2) を実現するために，従来の結合重みベクトルでは，単調減少する学習係数に基づいて，入力ベクトルに一様に近づけるだけであったのに対して，提案する再生手法に用いる結合重みベクトル更新則を以下のように改良した．

- 入力ベクトルの適合度が高い場合は，結合重みベクトルの更新量を大きくする．
- 入力ベクトルの適合度が高い場合は，多くの結合重みベクトルを更新する．
- 更新前の結合重みベクトルの適合度が既に高い場合は，結合重みベクトルの更新量を小さくする．

これらの観点から，実数値結合重みベクトルの更新量とバイナリ結合重みベクトルの更新ビット数が総合的に決定される．この結合重みベクトル更新則を採用した自己組織化マップを再生手法

として用いることにより，高い多様性を維持し，かつ高い適合度をもつ個体集団を生成することが可能となる．この結合重みベクトル更新則を適用した実数値自己組織化マップとバイナリ自己組織化マップを再生手法に用いて，実数値遺伝的アルゴリズムとバイナリ遺伝的アルゴリズムへ適用した．

実数値遺伝的アルゴリズムでは，3つの連続変数最適化問題を解くことによって，提案する再生手法を用いた際の，局所的な収束性，収束のロバスト性，大域的な探索性能について検証した．探索に要した世代数と計算時間に関してルーレット選択を用いた従来の遺伝的アルゴリズムと比較することにより，提案する再生手法の有効性を示した．また，タンクの水位を制御するPIDコントローラのパラメータ設計へ応用し，熟練者が設計したPIDコントローラとほぼ同等の性能をもつコントローラを設計することができた．実数値遺伝的アルゴリズムを用いたコントローラ設計は，設計者の人的負担を軽減することが可能となるので，工学的に非常に有効な最適化手法である．しかしながら，以下の点を考慮する必要がある．

- 遺伝的アルゴリズムにおける評価関数を設計する際，制御対象に関するある程度の知識を必要とする．
- 計算機シミュレーションを行なうので，制御対象やコントローラの物理モデルが既知でなければならない．

バイナリ遺伝的アルゴリズムでは，0-1ナップサック問題を解くことによって，提案する再生手法を用いた際のバイナリ空間における探索性能について検証した．探索に要した世代数と計算時間に関してルーレット選択を用いた従来の遺伝的アルゴリズムと比較することにより，提案する再生手法の有効性を示した．しかしながら，従来のバイナリ結合重みベクトル更新則では，ベクトルの各要素を更新する順序が考慮されていないので，更新する要素によっては，更新後の結合重みベクトルの適合度が適切に上昇しない場合があり，探索が非効率になることがある．

この問題点を解決するために，要素の重要性を考慮したバイナリ結合重みベクトル更新則を提案した．提案するバイナリ結合重みベクトル更新則では，入力ベクトルの各要素に対して，適合度に関する重み付き平均値を算出する．この算出した平均値をもとに，更新する要素の順序を決定する．提案する結合重みベクトル更新則を採用した再生手法と，従来の結合重みベクトル更新則を採用した再生手法，ルーレット選択を用いた各バイナリ遺伝的アルゴリズムを0-1ナップサック問題に適用した結果，提案する結合重みベクトル更新則を採用した再生手法は，従来の再生手法と比較して，世代数，計算時間ともに効率のよい探索を実現できた．

第4章　結論

本論文では，自己組織化マップの分布近似能力を用いた，遺伝的アルゴリズムの新しい再生手法を開発した．以下，各章で得られた成果について述べる．

2章では，まず，バイナリおよび実数値遺伝的アルゴリズムの個体表現方法，遺伝的操作，探索方法および特徴を述べた．次に，バイナリおよび実数値自己組織化マップの構成，学習方法および特徴を述べ，最後に遺伝的アルゴリズムの従来の再生手法の問題点を挙げた．

遺伝的アルゴリズムは，以下に述べる特長をもつ．

(1) 遺伝的アルゴリズムは，集団による多点探索を行う．

(2) 遺伝的アルゴリズムでは，問題に対する充足度（適合度）のみを用い，先見的な補助知識を必要としない．

(3) 遺伝的アルゴリズムにおける探索点の遷移は，確率論のみに基づき，決定論的なルールを必要としない．

従来のバイナリおよび実数値遺伝的アルゴリズムでは，交叉・突然変異・再生処理を世代ごとに繰り返すことによって，探索が進行していく．再生手法は，これらの処理の中で，次世代に残す個体を生成するという特に重要な役割をもつ．従来のバイナリおよび実数値遺伝的アルゴリズムでは，ルーレット選択やトーナメント選択などの再生手法が用いられる．これらの再生手法では，次世代の個体集団は，現世代の個体集団から，高い適合度の個体を増殖させ，低い適合度の個体を淘汰することによって生成される．具体的には，高い適合度の個体を“コピー”することで実現される．しかしながら，コピーにより実現される次世代個体集団の生成法は，個体集団の多様性を損なうので，探索初期における局所解への収束や，交叉効率の低下による探索効率の悪化を引き起こす．

一方，自己組織化マップは，提示されたバイナリおよび実数値の入力ベクトルに対して，結合重みベクトルを近づける操作（学習）を行うことにより，以下に述べる特長をもつ．

(1) 学習後の結合重みベクトルは，競合層上で，入力ベクトル集合の位相的な関係を保持している．

(2) 学習後の結合重みベクトル集合の分布は，入力ベクトル集合の分布を近似している．

本研究では，遺伝的アルゴリズムの“(1) 集団による多点探索”という特長をふまえ，自己組織化マップの“(2) 結合重みベクトル集合の分布は入力ベクトル集合の分布を近似する”という特長を利用して，遺伝的アルゴリズムの問題点を解決する．

3 章では，自己組織化マップを再生手法として用いた遺伝的アルゴリズムを提案した．本章で導入した自己組織化マップは，以下に述べる点が従来の自己組織化マップと異なる．

- 入力ベクトル数と競合層ユニット数（結合重みベクトル数）が等しい．
- 入力ベクトルの適合度を考慮し，適合度に基づく結合重みベクトル更新を行うことにより，入力ベクトル集合内の高い適合度をもつ入力ベクトルの分布を近似することができる．

提案する再生手法では，自己組織化マップへの入力ベクトル集合を現世代の個体集団，結合重みベクトル集合を次世代に残す個体集団として扱い，上記の点を考慮した自己組織化マップの学習によって，再生後の個体集団を生成する．自己組織化マップを用いることによって，高い多様性，かつ高い適合度をもつ個体集団を次世代に残すことが可能となる．また，本研究で導入した適合度に基づく結合重みベクトル更新則は，実数値自己組織化マップ，バイナリ自己組織化マップ双方に用いることが可能である．

特に，バイナリ自己組織化マップの結合重みベクトル更新則に関しては，更新する要素の順序が探索の効率に影響を及ぼすという問題があった．本研究では，個体集団内の各要素に対して，適合度に関する重み付き平均値をもとに，更新する要素の順序を決定する手法を提案した．提案するバイナリ結合重みベクトル更新則を用いた再生手法では，更新する要素の順序を考慮することにより，従来の再生手法と比較して，より効率のよい探索が可能である．

提案する再生手法を，実数値遺伝的アルゴリズムに適用し，DeJong がベンチマークとして提案した連続変数最適化問題と 0-1 ナップサック問題に適用した．適用した結果，以下のような特長をもつ探索空間において，提案する再生手法の有効性を示した．

- 実数値探索空間での局所的探索能力
- 実数値探索空間での局所的探索能力に関するロバスト性
- 実数値探索空間での大域的探索能力
- バイナリ探索空間での探索能力

また，実問題への応用として，タンクの水位を制御する PID コントローラのパラメータ設計を行った．設計した結果，評価関数と制御対象のモデル化に対する考慮は必要であるが，熟練者とほぼ同様の性能を示す PID コントローラを設計することができた．

以上のように，本研究では，入力ベクトルの適合度に基づいて学習を行う自己組織化マップ用いた再生手法を提案し，遺伝的アルゴリズムに適用した．提案する再生手法を採用した実数値およびバイナリ遺伝的アルゴリズムを用いて，連続変数最適化問題と組み合わせ最適化問題を解いた．提案する再生手法と従来の再生手法を用いた遺伝的アルゴリズムにおいて，探索に要した世代数と計算時間を比較することによって，提案手法の有効性を示した．また，提案する再生手法を用いた遺伝的アルゴリズムは，これまでに提案されている高性能な交叉法やエリート保存選択等と併用することにより，さらにその探索の効率を上げることができると考えられる．さらに，自己組織化マップに入力するデータをベクトルのみでなく，木構造等にすることにより，提案する再生手法は，遺伝的アルゴリズムのみならず，遺伝的プログラミングや進化戦略等への応用も期待できると考えられる．

謝辞

本研究を遂行するにあたり，終始懇切丁寧なご指導を賜わりました九州工業大学大学院生命体工学研究科 山川 烈 教授に心から感謝致します．また，本論文をまとめるにあたり，有意義なご助言とご討論を頂いた九州工業大学情報工学部システム創成情報工学科 岡崎悦明 教授，廣瀬英雄 教授，同大学情報工学部電子情報工学科 古川昌司 教授に謝意を申し上げます．さらに，本研究及び本論文作成にあたり数々の丁寧なご支援を賜りました九州工業大学大学院生命体工学研究科 堀尾恵一 助手に心から感謝致します．また，終始ご助言とご激励を頂いた九州工業大学大学院生命体工学研究科山川研究室の方々に厚く感謝致します．

なお，この学位論文の研究の一部は，21 世紀 COE プログラム「生物とロボットが織りなす脳情報工学の世界」（拠点番号 J19）の推進事業として実施いたしました． 関係 各位ならびに関係部署に深く感謝いたします．

最後に，博士課程での研究を遂行するにあたり，多くのご支援とご激励を頂いた福岡県嘉穂郡筑穂町 永芳達夫 町長を始めとする関係 各位ならびに関係部署と，影ながら支えて頂いた両親に心から感謝の意を表します．

参考文献

[1] J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," Journal of the Association for Computing Machinery, Vol. 3, pp. 297-314, 1962.

[2] D. B. Fogrl, "Evolutionaly Computation: A New Transactions", IEEE Transaction on Evolutionary Computation, Vol. 1, No. 1, p.1 1998.

[3] 玉置 久, "進化的アルゴリズムの方法論," 日本ファジィ学会誌, Vol. 10, No. 4, pp.593-601, 1998.

[4] Y. Jin and H. Branke, "Evolutionary Optimization in unsertain environments - A survey", *IEEE Transaction on Evolutionary Computation*, vol. 9, no. 3, pp. 303-317, 2005.

[5] J. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, MIT Press 1992.

[6] J. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press 1994.

[7] T. Back, F. Hoffmeister and H. -P. Schwefel, "A Survey of Evolution Strategies," Proceedings of 4th International Conference on Genetic Algorithms, pp. 2-9, 1991.

[8] H. -P. Schwefel, *Numerical Optimization of Computer Models*, Birkhauser, 1977.

[9] J. H. Holland, *Adaptation in Natural and artificial systems*, University of Michigan Press, 1975.

[10] K. A. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral dissertation, University of Michigan Press, 1975.

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company Inc., 1989.

[12] L. Davis, *The Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1990.

[13] L. J. Eshleman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," Foundations of Genetic Algorithms 2, pp. 187-202, 1993.

[14] 北野 宏明, 遺伝的アルゴリズム 1, 産業図書, 1993.

[15] 伊庭 斉志, 遺伝的アルゴリズムの基礎ー GA の謎を解く, オーム社, 1994.

[16] 坂和 正敏, 田中 雅博, 遺伝的アルゴリズム, 朝倉書店, 1995.

[17] 北野 宏明, 遺伝的アルゴリズム 3, 産業図書, 1997.

[18] C. Z. Jonikow and Z. Michalewicz, "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms," Proceedings of 4th International Conference on Genetic Algorithms, pp. 31-36, 1993.

[19] I. Ono, S. Kobayashi and K. Yoshida, "Global and multi-objective optimization for lens design by real-coded genetic algorithms," SPIE Proc. of International Optical Design Conference, Vol. 3482, pp. 110-121, 1998.

[20] 小野 功, 山村 雅幸, 喜多 一, "実数値 GA とその応用", 人工知能学会誌, Vol. 15, No. 2, pp. 259-266, 2000.

[21] 北野 宏明, 遺伝的アルゴリズム 4, 産業図書, 2000.

[22] D. T. Pham and G. Jin, "Genetic algorithm using gradient-like reproduction operator," Electronics Letters, Vol. 31, No. 10, pp. 1558-1559, 1995.

[23] I. Ono and S. Kobayashi, "A real-coded genetic algorithm for function optimization using unimodal normal distribution Crossover," Proc. of 7th International Conference on Genetic Algorithms, pp. 246-253, 1997.

[24] 小嶋 和徳, 石亀 昌明, 松尾 広, "遺伝的アルゴリズムにおける SRG 選択法の提案," 情報処理学会論文誌, Vol. 40, No. 12, pp. 4248-4257, 1999.

[25] 小嶋 和徳, 石亀 昌明, "遺伝的アルゴリズムにおける大域探索と局所探索両面を考慮した適応型 HRG 選択法の提案," 画像電子学会誌, Vol. 29, No. 2, pp. 140-146, 2000.

[26] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biological Cybernetics, Vol. 43, pp. 59-69, 1982.

[27] T. Kohonen, *Self-Organization and Associative memory*, Springer-Verlag, 1984.

[28] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, 1995.

[29] M. M. Van Hulle, *Faithful Representations and topographic maps: From distortion-to information-based self-organization*, Jhon Wiley & Sons, Inc., 2000.

[30] S. Lu, "Pattern Classification Using Self-Organizing Maps," Proceedings of International Joint Conference on Neural Networks, Vol.3, pp. 471-476, 1990.

[31] F. Blayo and P. Demaritines, "Data Analysis: How to Compare Kohonen Neural Networks to Other Techniques?," Proceedings of International Workshop on Artificial Neural Networks, pp. 469-476, 1991.

[32] X. Zhang and Y. Li, "Self-Organizing Map as a New Method for Clustering and Data Analysis," Proceedings of International Joint Conference on Neural Networks, Vol.3, pp.2448-2451, 1993.

[33] S. Nakagawa, Y. Ono and K. Hur "Estimation of Probability Density Function and Evaluation by Vowel Recognition," Proceedings of International Joint Conference on Neural Networks, Vol.3, pp.2223-2226, 1993.

[34] C. Diamantini and A. Spalvieri, "Vector Quantization for Minimum Error Probability," Proceedings of International Conference on Artificial Neural Networks, Vol.2, pp.1091-1094, 1994.

[35] E. Hakkinen and P. Koikkalainen, "The Neural Data Analysis Environment," Proceedings of Workshop on Self-Organizing Maps, pp.69-74, 1997.

[36] S. Kaski, J. Kangas and T. Kohonen, "Bibliography of Self-Organizing Map (SOM) Papers:1981-1997," Neural Computing Surveys 1, pp. 102-350, 1998.

[37] H. Tokutaka, K. Yoshihara, K. Fujimura, K. Iwamoto, T. Watanabe and S. Kishida, "Application of Self-Organizing Maps to the Composition Determination of Chemical Products," Proceedings of International Joint Conference on Neural Networks, Vol.1, pp.301-305 1998.

[38] T. Yamakawa, K. Horio and T. Hiratsuka, "Advanced Self-Organizing Maps using binary weight vector and its digital hardware design," Proc. of the 9th International Conference on Neural Information Processing (ICONIP'02), Vol. 3, pp. 1330-1335, 2002.

[39] 山川 烈, 堀尾 恵一, 平塚 智一, "バイナリ重みベクトルの自己組織化マップとそのディジタルハードウェア設計, " 第18回ファジィシステムシンポジウム講演論文集, pp. 551-554, 2002.

[40] 佐藤 次男, 中村 理一郎, 伊藤 惇, Cによる理工学問題の解法, 日刊工業新聞社, 1981.

[41] G. F. Franklin, J. D. Powell and M. Workman, *Digital Control of Dynamic Systems*, Addison-Wesley, 1998.

[42] 得丸 英勝, 自動制御, 森北出版, 1981.

[43] R. Kubota, K. Horio and T. Yamakawa, "Reproduction Strategy Based on Self-Organizing Map for Genetic Algorithms," *International Journal of Innovative Computing, Information and Control* , Vol. 1, No. 4, pp. 595-607, 2005.

[44] R. Kubota, K. Horio and T. Yamakawa, "Binary SOM Based on Significance of Inputs and Its Application to Reproduction of GA," Proceedings of Workshop on Self-Organizing Maps 2005 (WSOM'05), pp.211-218, 2005.

[45] 久保田 良輔, 堀尾 恵一, 山川 烈, "スキーマを利用した自己組織化マップの重み更新則とその遺伝的アルゴリズムへの応用、" 第18回バイオメディカル・ファジィ・システム学会年次大会講演論文集, pp.39-42, 2005.