競合型進化的計算法に基づく階層的最適化による複数の探索空間を持つ問題の解決に関する研究

著者	石川 秀大
発行年	2016
その他のタイトル	The Study of Solving the Problem Having Multiple Search Spaces by Hierarchical Optimize Method Based on the Competitive
学位授与年度	平成27年度
学位授与番号	17104甲生工第266号
URL	http://hdl.handle.net/10228/5717

競合型進化的計算法に基づく階層的最適化による複数の探索空間を持つ問題の解決に関する研究

石川　秀大

目 次

第1章　序論

本論文は，複数の探索空間を持つ問題に対して有効な解決アルゴリズムの開発を目的とし，遺伝的アルゴリズムをベースとした，階層構造を持つ最適化手法について述べたものである．

1.1　背景と目的

遺伝的アルゴリズム（Genetic Algorithm：GA）[1] は，生物の進化の過程を工学的に模倣した最適化手法として，1962 年に J. Holland によって提案された．GA は，生物の進化過程を利用した計算法の総称である，進化的計算法 [2, 3] のひとつであり，これらの計算法には，GA 以外に遺伝的プログラミング [4] や進化戦略 [5] などがある．この中でも，GA はアルゴリズムの簡便さや応用の幅の広さから，多くの分野で用いられ，その有効性が示されている．GA では，与えられた問題の設計変数を個体として表現し，個体の集合を母集団と呼ぶ．母集団に対して，選択・交叉・突然変異と呼ばれる遺伝的操作を毎世代適用することによって，母集団は進化を繰り返し，与えられた問題に対して最適な解を探索する．

GA は，扱う問題の特徴に応じて，より効率的な解探索を実現するために，これまで様々な拡張手法が開発されてきた．階層型 GA[6, 7, 8] は，GA の拡張手法の一つであり，システムの構成とパラメータを同時に最適化する必要がある問題に対して有効性が示されている．システムの構成とパラメータを同時に最適化する必要がある問題とは，言い換えれば，全体の最適化問題の中に部分問題として最適化問題があるような問題のことを指す．例えば，ニューラルネットワークにおいて最適な中間層の数，各ニューロンにおける重み係数の決定 [7]，ファジィ推論におけるメンバーシップ関数とそれらのルールの決定 [8]，多項式曲線フィッティングにおける適切な次数とパラメータの決定などが該当する．つまり，これらの問題は，各システムの構成におけるパラメータを決定するので，探索する空間はシステムの構成の組合せ数通り存在する．本研究では，このような性質を持つ問題を複数の探索空間を持つ問題と定義する．階層型 GA には，1) 個体そのものを階層化し，コントロール遺伝子とパラメトリック遺伝子の 2 つの遺伝子を用いる方法 [9]，2) 問題を階層的に分割し，上位の層ではシステムの構成，下位の層ではそれらのパラメータをそれぞれ GA によって最適化する方法 [10] の 2 つの方法がある．一般的に階層型 GA とは，個体を階層化する 1) の方法のことを指すが，これらの方法は，本質的な特徴はほぼ同じである．多項式曲線

フィッティングのように，システムの構成が異なる個体間の交叉において，それぞれのパラメータが利用できる問題には 1) の方法は非常に有効だが，2) の方法では，基本的にシステムの構成が異なる個体間において，それぞれ独立して探索が行われ，パラメータの情報のやりとりは行われないので，このような問題を解決可能ではあるが，有効であるとはいえない．一方で，下位層の問題（パラメータの決定）が組み合わせ最適化となっている問題では，システムの構成が変わると，それまで保持していたパラメータ情報を有効に活用できなくなるので，1) の方法と比べてシステムの構成に応じて独立に GA を行う 2) の方法が有効である．また，それぞれの遺伝的オペレータについて，1) の方法では，システムの構成によっては個体の次元数が異なる場合がある．この場合，単純な交叉の方法は適用できず，問題に応じて特殊な交叉方法を検討する必要がある．一方，2) の方法では，上位層，下位層では単純な GA の処理が行われるため，これまでの交叉方法が適用可能であり，特殊な交叉方法を検討する必要がない．2) の方法の他の利点として，上位層，下位層において，GA を用いた方法のほかに計算コストの小さい，粒子群最適化（Particle Swarm Optimization : PSO）[11] や蟻コロニー最適化（Ant Colony Optimization : ACO）[12]，タブー探索法 [13] など問題に応じて利用する方法を選択可能である点，比較的実装が容易な点が挙げられる．本研究では，手法の汎用性，実装の容易さ，拡張性の高さを考慮し，2) の方法を階層型 GA として採用する．

複数の探索空間を持つ問題では，考慮する探索空間の数によって問題解決のアプローチは異なる．例えば，与えられたデータから，モデルの次数と設計変数を決定する多項式曲線フィッティングは，探索空間の数は考慮する次数の数によって決まる．つまり，多項式曲線フィッティングでは，考慮する探索空間の数はそれほど多くないので，すべての探索空間をもれなく探索することが可能であり，並列処理によって容易に解決できる．しかしながら，考慮する探索空間の数が非常に多い場合は，現実的な時間内にすべての空間を探索することは困難であり，できるだけ最適解の存在する可能性の高い空間を絞って探索する必要がある．例えば，複数車両配送計画問題（Multiple Vehicle Routing Problem : mVRP）のように，複数の車両によって各顧客を巡回するといった問題では，探索空間は各車両が担当する顧客の割り当てによって決められ，このとき探索空間の数は顧客数によって爆発的に増加する．この場合は階層型 GA の概念を用いて，探索空間も最適化手法によって最適化することで解決可能である．複数の探索空間を持つ問題では，多くの空間を探索する必要があるため，GA を適用した場合，非常に多くの計算時間を必要とする．また最適解の存在する探索空間以外の探索は不要であるので，最適解の存在する探索空間を早期に決定し，重点的に探索を行うことができれば，少ない計算時間で安定的に最適解を発見することが可能になる．

本論文では，“ 探索空間どうしの競合 ” という概念を用いて，少ない計算コストで安定的に複数

の探索空間から最適解を発見できる 2 つの新しい最適化手法を提案する．

1.2 探索空間どうしの競合に基づく階層的最適化手法

本研究では，複数の探索空間を持つ問題に着目し，考慮する探索空間の数が少ない問題において，複数解空間競合型分散 GA（Multi-space Competitive Distributed GA：mcDGA）と，膨大な数の探索空間を持つ問題において，階層的複数解空間競合型分散 GA（Hierarchical mcDGA：HmcDGA）を提案する．mcDGA は，Tanese によって提案された分散 GA（Distributed Genetic Algorithm：DGA）[14] の概念に着想を得て開発したものである．分散 GA は，複数の母集団を持つ GA であり，個体群（母集団）はいくつかの小規模な個体群（サブ母集団）に分割され，各サブ母集団は独立に探索を行う．DGA では，探索の途中に個体の移住という処理を行う．移住によって，サブ母集団間において個体のやり取りが行われ，異なる形質の個体が集団に追加されるため，各サブ母集団における多様性が高まり，従来の GA と比較してより安定して解を求めることができる．mcDGA では，DGA におけるサブ母集団を単一の探索空間ではなく，異なる探索空間を探索する．しかしながら，各探索空間の次元数は異なるため，従来の分散 GA のような個体の移住は不可能である．したがって，mcDGA では，競合によって各サブ母集団の個体数のみを調節する．競合によって，解の存在する可能性の高い探索空間に個体が集まり，早期に，良質な解を発見することが期待できる．

HmcDGA は，前述の階層型 GA と mcDGA を着想として，考慮する探索空間の数が非常に多い問題に対応するために開発した．HmcDGA は，探索空間の最適化を行う上位層と，各探索空間の最適解を探索する下位層から構成される．例えば，前述の mVRP を HmcDGA によって解決する場合，各車両が配送する顧客の割り当て（探索空間）は上位層，それらの経路（各探索空間における解）は下位層に配置される．HmcDGA では，個体の評価は下位層においてのみ行われ，下位層における最高評価値が各探索空間の評価値となる．上位層の最適化は，下位層の評価値に基づいて行われ，この操作を設定世代数繰り返す．HmcDGA では，mcDGA と同様に“探索空間どうしの競合”を行うことで，最適解が存在する可能性が高い探索空間の個体数が増加し，それ以外の探索空間の個体数は減少する．競合によって，計算コストを大幅に抑えることができ，解を発見する可能性は高くなる．また，最適化手法として GA を利用しているため，変数の種類に関係なく，多くの最適化問題に適用可能である．さらに，階層化によって上位層，下位層ともに単純な最適化問題となっているため，従来の階層型 GA と同様，複雑な設計変数や更新方法を必要とせず，一般的な GA に適用されている処理を用いることができる．このことより，HmcDGA は，汎用的な最適化ツールとして期待することができる．

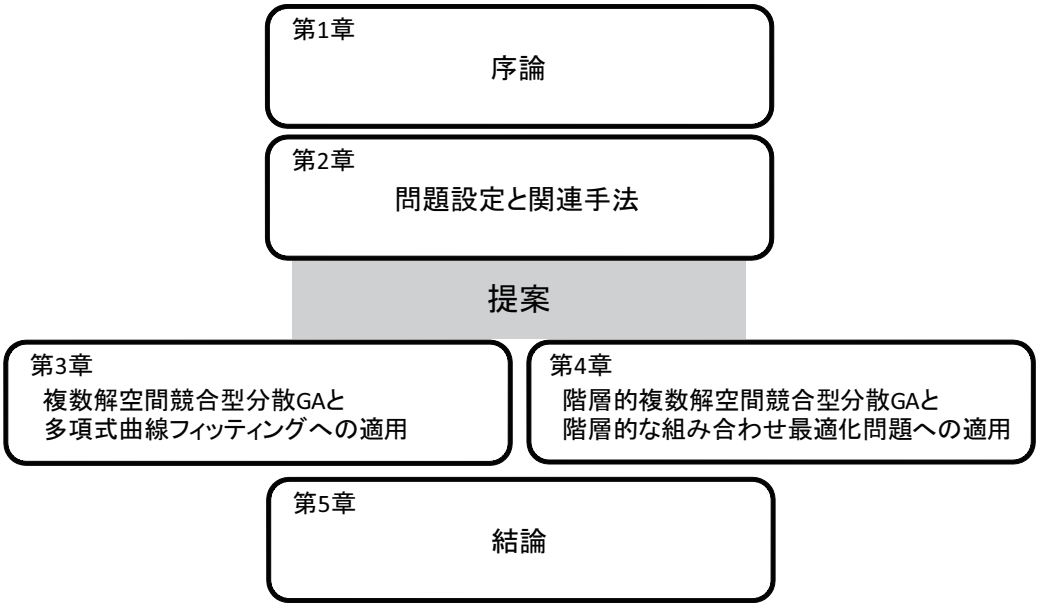


図 1.1: 本論文の構成.

1.3 本論文の構成

本論文は図 1.1 に示すように，5 つの章から構成される

第 1 章は序論である.

第 2 章では，本研究で扱う問題を提起し，従来の GA と拡張手法である階層型 GA と分散 GA の概要とそれらの特長について述べる.

第 3 章では，複数の探索空間から最適解を発見する手法として，mcDGA を提案する．mcDGA を多項式曲線フィッティングに適用し，競合の基本的な動作と有効性について説明する．シミュレーションでは，8 つの探索空間 (1 次から 15 次の奇関数) を考慮し，次元数の異なる 2 つのテストデータを用いて，競合を行なわない並列 GA[15] と解を発見するまでの世代数について比較する．本章では，多項式曲線フィッティングを用いて，2 つの競合の方法やタイミングを検討し，その結果について示す.

第 4 章では，mcDGA の拡張であり，考慮する探索空間の数が多い場合に有効な最適化手法として HmcDGA を提案し，HmcDGA のアルゴリズムと特長について述べ，複数車両配送計画問題（ Multiple Vehicle Routing Problem : mVRP ）[16, 17] と Flexible Job-shop Scheduling 問題（Flexible Job-shop Scheduling Problem : FJSP）[18] に適用する．mVRP は異なる複数の車両を用いて，顧客に商品を配送する問題であり，この問題を HmcDGA によって解決する場合，各車両が配送する顧客の割り当て（探索空間）は上位層，それらの経路（各探索空間における解）は

下位層に配置される．FJSP は，mVRP 同様，非常に多くの探索空間を考慮しなければならない問題であり，FJSP において，探索空間の数は各ジョブの処理を行うマシンの組み合わせによって決まる．FJSP のベンチマーク問題として，Brandimarte[19] を用い，他の文献の手法の結果と比較する．HmcDGA が GA をベースとし，問題に特化した他の手法に近い性能を持つことを示すことで，FJSP における HmcDGA の解探索能力および様々な問題への汎用性について考察する．

第 5 章は結論である．

第2章　問題設定と関連手法

2.1　はじめに

本論文では，複数の部分最適化問題が階層的に構成された階層的最適化問題を複数の探索空間を持つ問題と定義し，遺伝的アルゴリズムを主とした階層的最適化手法によって解決するというものである．本章では，本論文で扱う問題の位置づけを明らかにし，関連する手法の基本的な処理について説明する．2.2 節では，本研究で扱う問題について説明する．2.3 節では，本研究においてメインの手法である遺伝的アルゴリズムと，階層型遺伝的アルゴリズム，分散遺伝的アルゴリズムについて説明する．2.4 節では，本章を統括する．

2.2　複数の探索空間を持つ最適化問題

本研究では，複数の部分最適化問題が階層的に構成された階層的最適化問題を扱う．階層的最適化問題には，組み合わせ最適化問題や線形・非線形計画問題，スケジューリング問題のような代表的な最適化問題だけでなく，多目的最適化問題や人の主観に基づく最適化問題のような特殊な最適化問題も部分最適化問題として構成される場合があり，様々な性質を持つ問題が存在する．したがって，単一の方法で解決することは困難である．階層的最適化問題は，より複雑な処理を機械によって自動化する場合に，解決しなければならない，もしくは問題解決のための 1 つのアプローチとして位置づけることができる．つまり，今後も進歩し続ける科学技術によって複雑化された社会環境において，階層的最適化問題を効率的に解決する方法を示すことは，学術的な寄与が非常に大きい．

階層的最適化問題は，各部分問題どうしに対応関係がある場合が多い．例えば，ニューラルネットワークにおいて最適な中間層の数，各ニューロンにおける重み係数の決定 [7] では，上位層において中間の層の数が決定され，下位層において各ニューロンの重み係数が決定される．ファジィ推論におけるメンバーシップ関数とそれらのルールの決定 [8] や多項式曲線フィッティングにおける適切な関数とパラメータの決定なども同様に階層化することができる．つまり，階層的最適化問題は，複数の探索空間を持つ問題であると捉えることができる．本研究では，階層的最適化問題における上位層の部分問題の解を問題全体の探索空間，下位層の部分問題の解を各探索空間の

解と定義する．階層的最適化問題を複数の探索空間を持つ問題とするとき，探索空間の数は上位層の部分問題のスケールによって決まる．本論文では，多項式曲線フィッティング問題，複数車両配送問題，Flexible Job-shop Scheduling 問題の探索空間の数の異なる 3 種類の階層的最適化問題を扱う．多項式曲線フィッティング問題は，探索空間の数が少ない問題の例として扱い，この問題では，探索する関数を一つの探索空間とし，それらの係数を探索空間の解とする．探索空間の数が少ない場合，考慮するすべての探索空間を探索することが可能であるが，解の存在する可能性の低い探索空間も探索を行うことになり，余計な計算時間を要することになる．一方，複数車両配送問題と Flexible Job-shop Scheduling 問題は，探索空間の数が非常に多い問題の例として扱う．複数車両配送問題では，各車両における顧客の割当を上位層において決定し，各割り当てにおける最短ルートを下位層において決定する．Flexible Job-shop Scheduling 問題では，使用するマシンの組合せを上位層において決定し，それらの使用順序を下位層において決定する．この場合，現実的な時間内にすべての探索空間を探索することは不可能であり，探索空間も最適化によって決定する必要がある．つまり，探索空間の数が少ない問題と比較して，最適解の存在しない探索空間を多く探すことになり，多大な計算時間を必要とする．

2.3 関連手法

2.3.1 遺伝的アルゴリズム

遺伝的アルゴリズム（Genetic Algorithm：GA）は，生物の進化と選択淘汰の過程を模擬した最適化アルゴリズムであり，J.Holland により 1962 年に提案された．

自然界では，環境に適応できない生物は死滅し，子孫を残すことができないが，環境に適応した生物は生き残り，子孫を残していくことができる．そのような世代交代を幾度も繰り返すことによって，集団の中に優れた遺伝子が広がり，集団は繁栄する．このメカニズムをモデル化し，環境により適応した生物，すなわちある目的関数に対して最適な値を与えるような解（設計変数値）を計算機上で生成しようというのが GA である．GA では，問題の解を個体として表し，これらの集団に選択，交叉，突然変異などの遺伝的操作を繰り返すことで世代を重ねるに毎に集団が進化していく．GA は以下のような特長を持つ．

1. 集団による多点同時探索を行うので，大域的な探索が可能である．
2. 問題に対する適合度のみを用いるので，先験的な補助知識を必要としない．
3. GA における探索点の遷移は，確率論にのみ基づき，決定的なルールを必要としない．
4. 個体の設計を問題に応じて変えることで，幅広い分野の問題に適用できる．

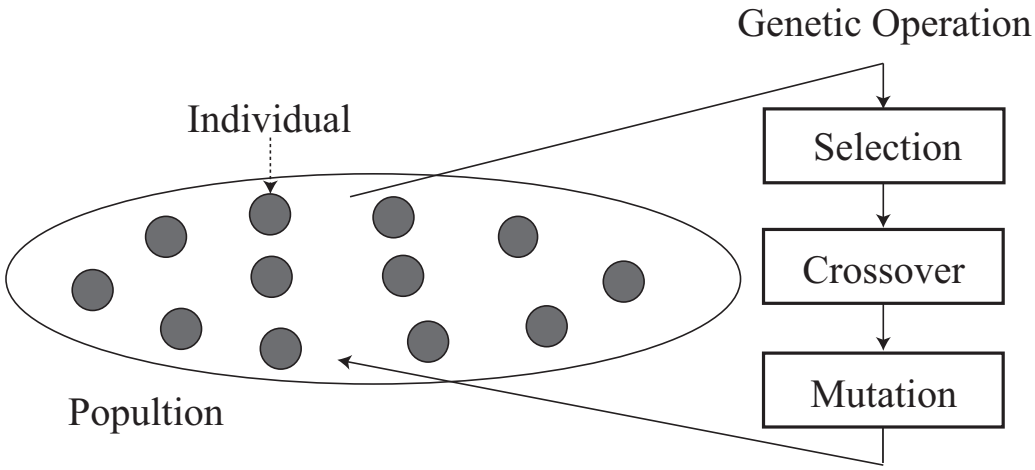


図 2.1: GA の探索の概念.

本節では，まず一般的な GA の概要，操作について説明し，次に GA の拡張手法である階層型 GA，分散 GA について説明する．図 2.1 に GA の概念図を示す.

GA では，問題の解を個体（Individual）として表現し，各個体は設計変数値がコーディングされた染色体（Chromosome）によって構成されている．この染色体をデコーディングすることにより設計変数に変換し，目的関数の値を計算する．このとき，染色体で表現されたものを遺伝子型（GenoType），これによって定まる個体の性質や特性を表現型（Pheno Type）と呼ぶ．また，個体の集合を個体集団（Population）と呼び，ある世代（Generation）を形成している個体のうち，環境（目的関数）への適合度（Fitness）が高い個体ほど次の世代に高い確率で生き残るように選択（Selection）する．さらに，それぞれの個体に対して，交叉（Crossover），突然変異（Mutation）などの遺伝的操作（Genetic Operater）によって次世代の個体集団を形成する．これらの一連の操作を繰り返し行うことによって解探索を行い，探索が進むごとに，高い適合度を持つ個体（最適解に近い個体）が増加し，やがて最適解が得られると期待できる．これが GA の基本的な概念である．以下に GA の流れを示し，それぞれの操作について説明する．

選択（Selection）

選択は次世代の個体集団を形成するための個体を選ぶ操作のことであり，適合度の高い個体ほど選択されやすい．したがって，適合度の低い個体は淘汰され，適合度の高い個体が増殖するので，世代が進むごとに個体集団全体の適合度が向上する．選択には様々な手法があり，いずれも適合度の高い個体が次の世代に選ばれやすくなっている．また，選択の方式を拡張したものとして，個体集団の中で最も適合度の高い個体は無条件に次の世代に選択されるというエリート保存戦略がある．この操作によって，親個体よりも劣る個体が多数生成された

場合も，もとの個体群で選ばれた最も適合度の高い個体が保存されているため，母集団全体の適合度は低下しない．

交叉（Crossover）

交叉は複数の個体内の染色体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的である．交叉は交叉率（Crossover rate）と呼ばれる確率にしたがって行われる．

突然変異（Mutation）

選択と交叉だけでは，初期個体集団内に存在する遺伝子の組み合わせでしか探索が行われない．よって，個体集団の多様性が乏しくなり，ある限られた性質の個体が増加するので，局所解に収束することが多い．この傾向は，バイナリ GA において特に顕著である．したがって，突然変異によって，交叉では生成できない子個体を生成し，探索領域を広げることで，局所解への収束を抑制する．突然変異は，突然変異率（Mutation rate）と呼ばれる確率に従い，染色体のある遺伝子座を対立遺伝子と置き換えることを示す．対立遺伝子とはその遺伝子座に入り得る遺伝子のことである．また，突然変異率が高すぎると良質な解を破壊してしまう可能性が高くなり，探索が収束しない可能性がある．

これらの特徴を組み合わせることで，GA は大域的かつ局所的な探索が可能となる．遺伝的アルゴル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 1：乱数を用いて個体集団を初期化し，各個体の適合度を算出する．

Step 2：集団から，各個体の適合度に基づいて，交叉によって次世代に子孫を残す個体を選択する．

Step 3：集団から選択された親個体どうしの交叉によって個体を生成する．

Step 4：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 5：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 2 に戻る．

終了条件は以下に示すいずれかによって決定される．

(1) 個体集団内の最高適合度が規定値を超えた場合．

(2) 個体集団の平均適合度が規定値を超えた場合．

(3) 世代交代の数が規定回数を超えた場合．

GA では，各個体は数値ベクトルによって表現される．数値ベクトルは，解決する最適化問題の種類によって，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．GA では，アルゴリズムの概念は同じものの，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．バイナリベクトルを個体として扱う GA をバイナリ GA[20, 21, 22, 23]，実数値ベクトルを扱う GA を実数値 GA[24] とよぶ．一般的にバイナリ GA は，組み合わせ最適化問題 [25, 3] に用いられ，代表例として 0-1 ナップサック問題 [26, 27] や巡回セールスマン問題 [28, 29] 等がある．実数値 GA は，連続最適化問題に用いられ [30, 31, 32]，代表例としてコントローラーやモデルのパラメータチューニング等がある．本論文では，バイナリ GA と実数値 GA の双方を指す場合，単に GA と表現する．また，GA の拡張手法として，階層型 GA がある．階層型 GA は，本研究で扱う，複数の探索空間を持つ問題に対して有効であり，1) 階層的な個体を持つ方法と 2) 問題を階層的に分割して解決する方法の 2 通りがある．この 2 つの方法は，本質的な特徴は似ている点が多いが，扱う問題によって使用する最適化手法を選択できるという点で，1) の方法と比較して 2) の方法は拡張性が高く，実装も容易である．その他の拡張手法として，分散 GA がある．分散 GA は，母集団全体をサブ母集団に分割し，サブ母集団間での個体の交換を行いながら探索を行う．それによって，多様性を保ちながら探索を行うことができるので，局所解に陥りにくく，短い世代での解の発見が期待できる．分散 GA はネットワークコーディング [33] や規則抽出 [34] といった問題に用いられている．

次小節では，バイナリおよび実数値 GA における各遺伝的操作の方法，および階層型 GA と分散 GA の概要について説明する．

バイナリ遺伝的アルゴリズム

バイナリ GA において扱われる個体を図 2.2 に示す．個体（Individual）は複数の遺伝子（Gene）によって構成されており，実世界における染色体（Chromosome）に相当する．バイナリ GA おける遺伝子は，図 2.2 に示すように ’ 0 ’ と ’ 1 ’ のバイナリ情報によって表現される．バイナリ GA で用いる個体 $\boldsymbol{I}^B$ は次式によって表される．

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

式 2.1 において，b_i は，個体における i 番目の遺伝子を表し，それぞれの遺伝子はバイナリ値をとる．4 章で扱う，巡回セールスマン問題や配送計画問題のような経路を求める問題では，各遺伝子に都市の番号や巡回する順番などが使われることもある．この場合は，後述する従来の交叉方法が適用できないため，Partially Matched Crossover (PMX)[35] や Cycle Crossover (CX) [36] , Order Crossover (OX) [37] など経路を最適化する問題に特化した交叉方法が用いられる．選択で

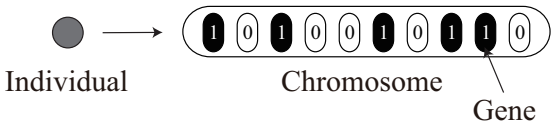


図 2.2: バイナリ GA における個体.

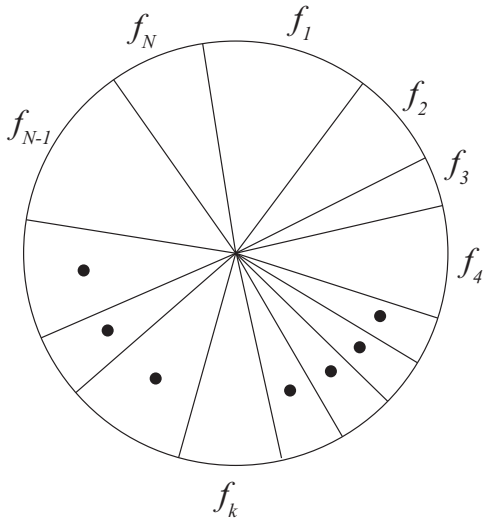


図 2.3: ルーレット選択.

は，算出された適合度をもとに，個体集団から交叉を行う個体を選択する．一般的にな選択手法としてルーレット選択（Roulette Wheel Selection）やトーナメント選択（Tournament Selection）などがよく用いられる．ルーレット選択では，すべ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交叉を行う個体を決定する．一方，トーナメント方式では，複数の個体を個体集団から無作為に選び，これらのうちで適合度が最も高い個体を交叉に用いる個体とする．これらの選択手法は，実現方法は異なるが概念はいずれも同じである．ルーレット選択の概要を以下に示す.

1. 個体群に含まれる N 個体の適合度 f_i の総和 F を計算する．

$$F = \sum_{i=1}^{N} f_i \tag{2.2}$$

2. 乱数 $rand \in [0, 1]$ を生成し，閾値 θ を計算する．

$$\theta = rand \times F \tag{2.3}$$

3. 次式のように個体の適合度を順に累積し，累積値が閾値 θ を越えるような n を求め，k 番目の個体を選択する．

$$\sum_{i=1}^{n} f_i \geq \theta \tag{2.4}$$

式 2.4 は図 2.3 のように，各個体の適合度に基づいて個体を選択することを意味する．また，選択は個体の表現に関係なく，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 GA 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．

一般的にバイナリ GA に用いられている交叉法を図 2.2 に示す．
交叉には，一点交叉（One-point Crossover）や多点交叉（Multi-points Crossover），一様交叉（Uniform Crossover）があり，それぞれ交叉点の数や位置が異なる．一点交叉や多点交叉では，まず選択によって親個体（Parent）を選択する．次に，選択した親個体どうしの遺伝子の一部を，交差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，親個体に対してあらかじめ用意しておいたマスク（Mask）をかけ，入れ替える遺伝子を決めることによって交叉を実現している．それぞれの交叉の様子を図 2.4 に示す．
突然変異では，選択された個体に対して，図 2.5 に示すように 1 つもしくは複数の遺伝子を対立する遺伝子に置換することによって新しい個体を生成する．

実数値遺伝的アルゴリズム

実数値 GA において扱われる個体の例を図 2.6 に示す．実数値 GA では，個体の遺伝子部分が実数値によって表現されている．実数値 GA で用いる個体 $\boldsymbol{I}^R$ は次式によって表される．

$$\boldsymbol{I}^R = (r_1, \ldots, r_i, \ldots, r_H), \quad r_i \in \mathcal{R}, \tag{2.5}$$

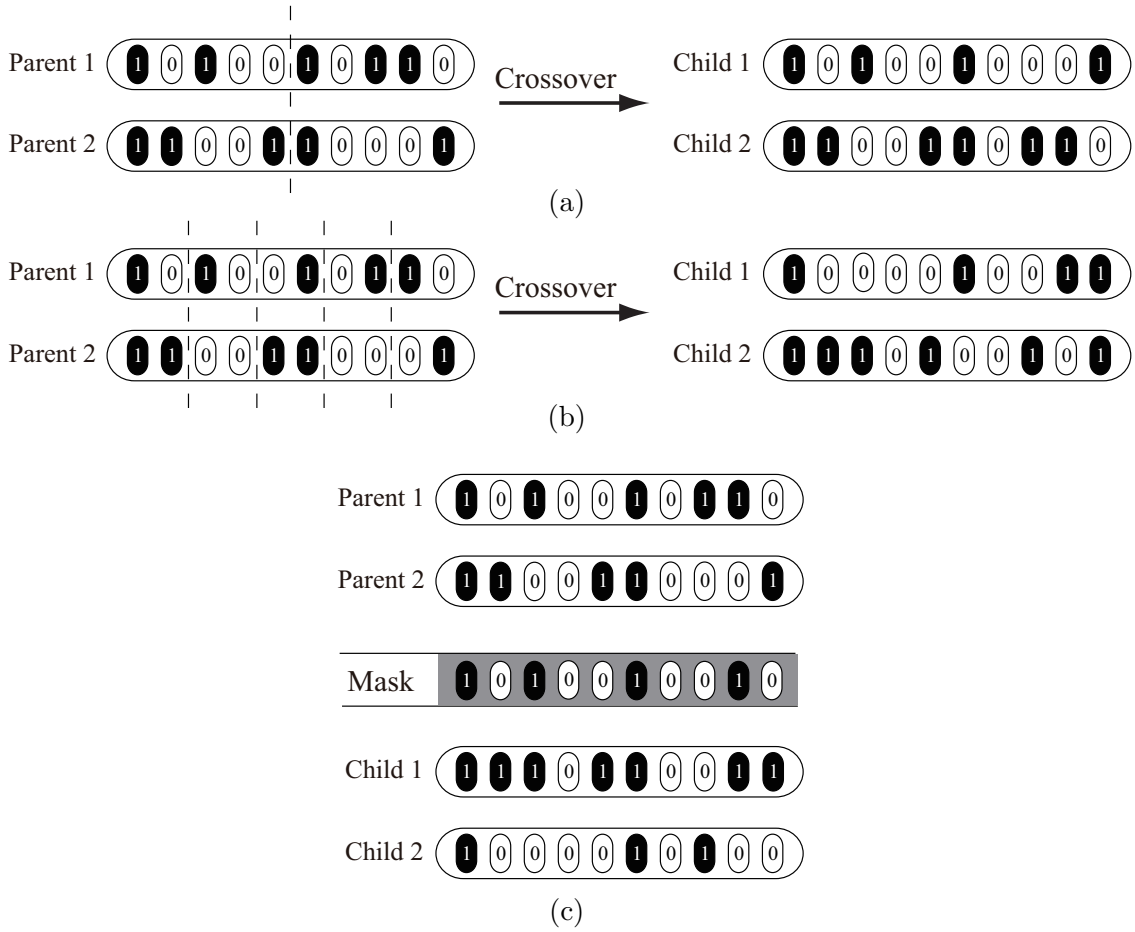


図 2.4: バイナリ GA における交叉．(a) 一点交叉．(b) 多点交叉．(c) 一様交叉．

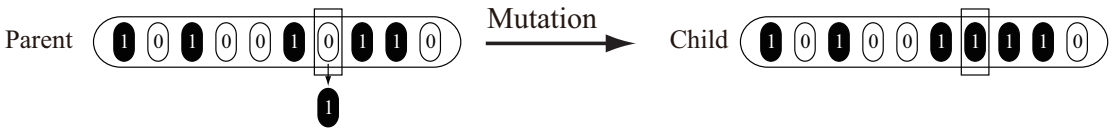


図 2.5: バイナリ GA における突然変異．

式 2.5 において，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 GA における選択方法は，バイナリ GA における選択方法と同様のものが用いられるが，交叉と突然変異の方法は，バイナリ GA の場合とは異なる．実数値 GA における交叉では，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．一般的に，BLX-α 交叉 [24]，単峰性正規分布交叉（Unimodal normal distribution crossover）[38]，シンプレックス交叉（Simplex crossover）[39] などが用いられている．BLX-α 交叉における子個体の生成の例を図 2.7 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 d_i を両側に αd_i だけ拡張し

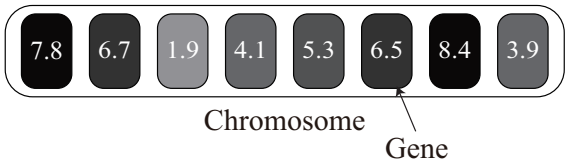


図 2.6: 実数値 GA における個体

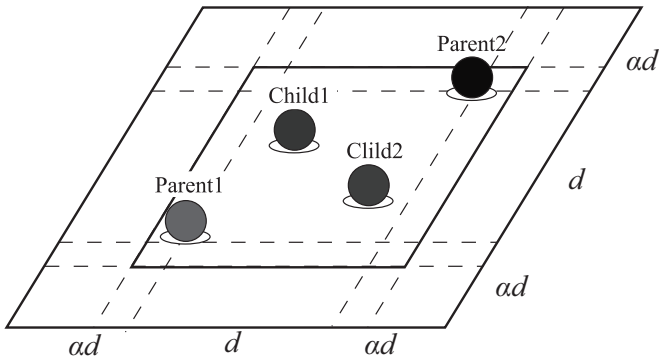


図 2.7: BLX-α 交叉

た区間に，一様乱数にしたが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 $\boldsymbol{C}^R = (c_1^R, \ldots, c_i^R, \ldots, c_H^R)$ は，2 つの親個体 $\boldsymbol{P}_1^R = (p_{11}^R, \ldots, p_{1i}^R, \ldots, p_{1H}^R)$,$\boldsymbol{P}_2^R = (p_{21}^R, \ldots, p_{2i}^R, \ldots, p_{2H}^R)$ から次式によって生成される．

$$c_i^R = U(min(p_{1i}^R, p_{2i}^R) - \alpha d_i,\, max(p_{1i}^R, p_{2i}^R) + \alpha d_i), \tag{2.6}$$

$$d_i = |p_{1i}^R - p_{2i}^R|), \tag{2.7}$$

式 2.6 において，$min(x, y)$, $max(x, y)$, $U(x, y)$ はそれぞれ，x，y の最小値，最大値，区間 $[x, y]$ の一様乱数を示す．

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換する．突然変異の方法には，置換する際に，実行可能領域内に一様乱数によって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実数値を発生させる境界突然変異がある．図 2.8 に一様突然変異の例を示す．

GA では，多点探索という特徴から，個体数が増えるほど，解の発見確率は高くなるが，選択，

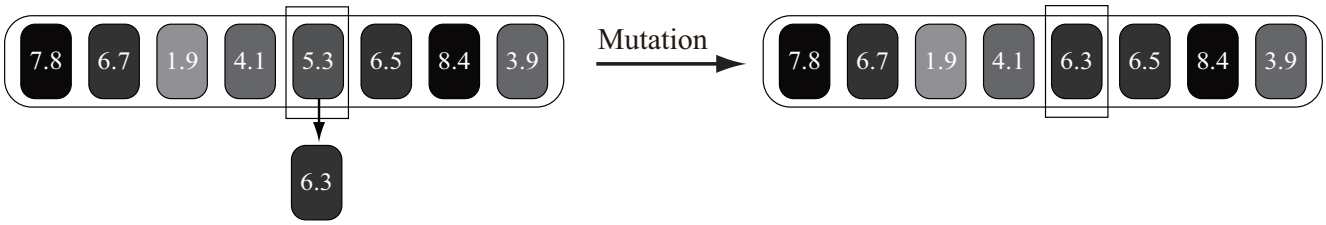


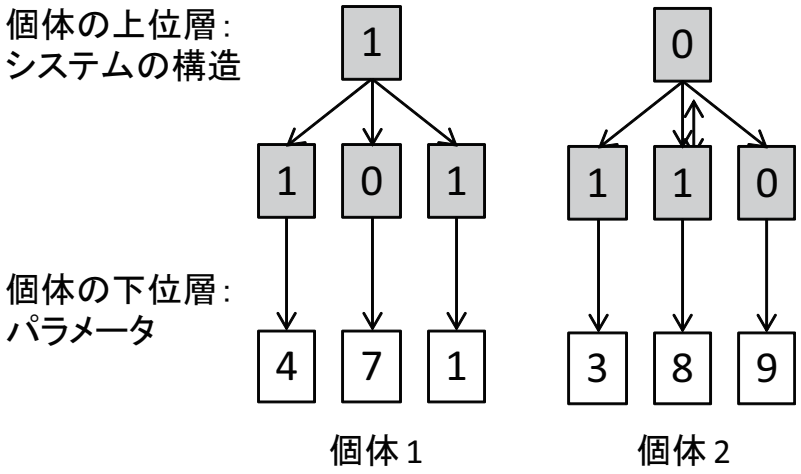
図 2.8: 一様突然変異.

交叉，突然変異といった遺伝的操作と適合度計算を各個体に対して何世代も行うので，多くの計算量を必要とする．したがって，効率的に最適解を得るためには，問題に応じて適切な個体数を設定しなければならない.

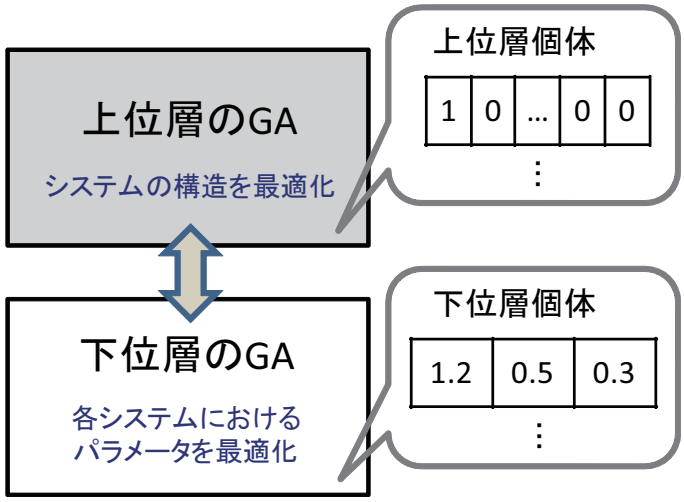
2.3.2 階層型遺伝的アルゴリズム

階層型遺伝的アルゴリズム（Hierarchical Genetic Algorithm : HGA）には，1) 個体そのものを階層化し，コントロール遺伝子とパラメトリック遺伝子の 2 つの遺伝子を用いる方法 [9]，2) 問題を階層的に分割し，上位の層ではシステムの構造，下位の層ではそれらのパラメータをそれぞれ GA によって最適化する方法 [10] の 2 種類がある．それぞれの階層型 GA のイメージを図 2.9(a), 図 2.9(b) に示す．一般的に階層型 GA とは，個体を階層化する 1) の方法のことを指し，この方法はニューラルネットワークにおいて最適な中間層の数，各ニューロンにおける結合重みの決定 [7], ファジィ推論におけるメンバーシップ関数とそれらのルールの決定 [8] や多目的最適化に応用されている [40]．図 2.10(a) と図 2.10(b) に，ニューラルネットワークのトポロジー最適化に 1) の階層型 GA を適用した場合の個体の表現と処理全体のブロック図をそれぞれ示す [8]．図 2.10(a) において，最上位の層は中間層の活性と不活性をバイナリで表現しており，中位の層では，各ニューロンの活性と不活性をバイナリで表現している．最下層では，上位 2 つの層において決められた活性ニューロンへの結合重みと閾値が実数値で表現されている.

上述の 2 つの階層型 GA は，本質的な特徴はほぼ同じである．多項式曲線フィッティングのように，システムの構成が異なる個体間の交叉においても，それぞれのパラメータが利用できる問題には 1) の方法は非常に有効だが，2) の方法では，基本的にシステムの構成が異なる個体間において，それぞれ独立して探索が行われ，パラメータの情報のやりとりは行われないので，このような問題を解決可能ではあるが，有効であるとは言えない．一方で，下位層の問題（パラメータの決定）が組み合わせ最適化となっている問題では，システムの構成が変わると，それまで保持していたパラメータ情報を有効に活用できなくなるので，1) の方法と比べてシステムの構成に応じて独立に GA を行う 2) の方法が有効である．また，それぞれの遺伝的オペレータについて，1) の



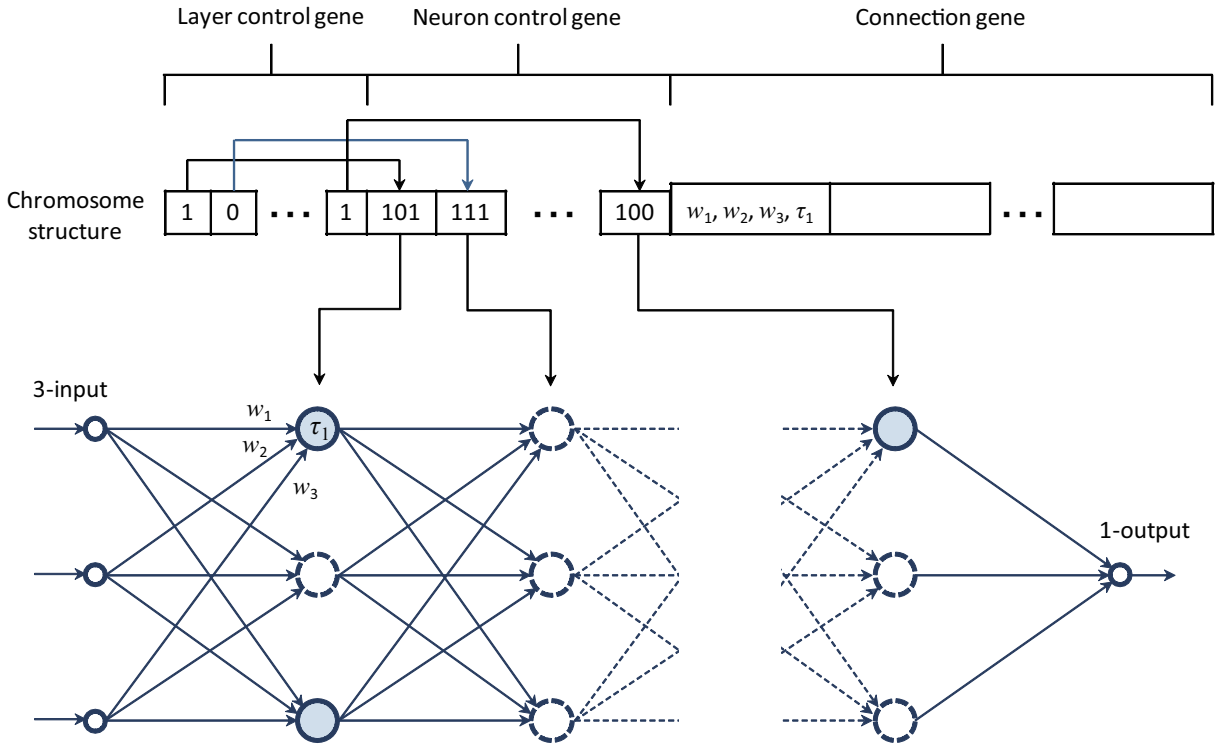
(a) 個体の階層化に基づく階層型 GA.



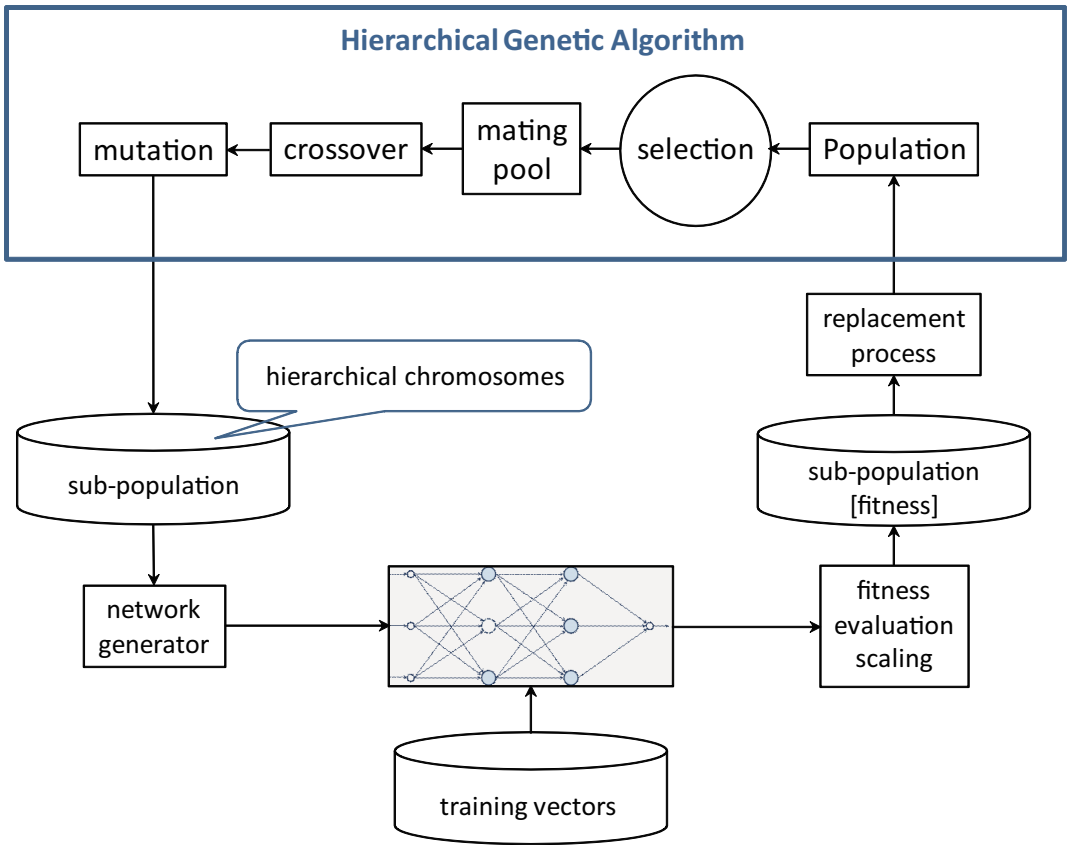
(b) 問題の階層化に基づく階層型 GA.

図 2.9: 階層型 GA の種類.

方法では，システムの構成によっては個体の次元数が異なる場合がある．この場合，単純な交叉の方法は適用できず，問題に応じて特殊な交叉方法を検討する必要がある．一方，2) の方法では，上位層，下位層では非常に単純な処理が行われるため，これまでの交叉方法が適用可能であり，特殊な交叉方法を検討する必要がない．2) の方法の他の利点として，上位層，もしくは下位層において，GA のほかに計算コストの小さい，粒子群最適化（Particle Swarm Optimization : PSO）[11] や蟻コロニー最適化（Ant Colony Optimization : ACO）[41, 12]，タブー探索法 [13] など問題に応じて利用する方法を選択可能である点が挙げられる．2) 階層型 GA では，各層はそれぞれ独立に処理を行っているため，各層間において必要な情報さえ交換できれば，その方法は問わな



(a) ニューラルネットワークのトポロジー最適化における階層型 GA の個体の表現.



(b) ニューラルネットワークのトポロジー最適化における階層型 GA のブロック図.

図 2.10: ニューラルネットワークのトポロジー最適化の階層化の例.

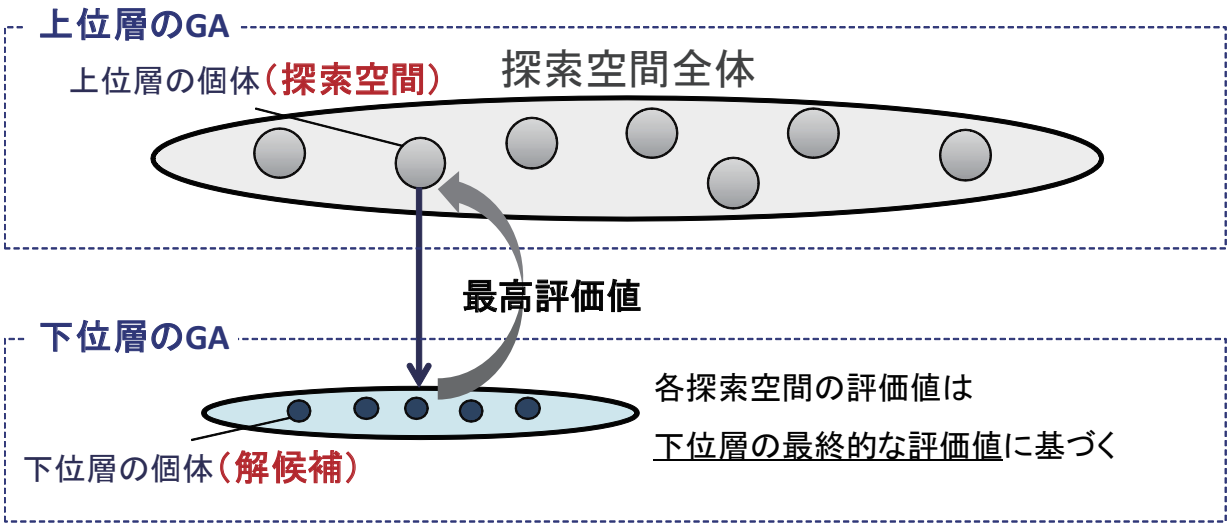


図 2.11: 本研究で扱う問題設定と階層型 GA のイメージ.

い．つまり，扱う問題に応じて，その問題を得意な方法を選択できるという点で 2) の方法は拡張性が高い．また，実装においても，2 つの GA を階層的に並べるだけなので，比較的実装は容易である．本研究では，手法の汎用性，拡張性の高さ，実装の容易さを考慮し，2) の方法を階層型 GA として採用する．図 2.11 に本研究で扱う階層的最適化問題と階層型 GA のイメージ図を示す.

2.3.3 分散遺伝的アルゴリズム

分散 GA は，1989 年に Reiko Tanese によって提案された．分散 GA では，全個体を含む母集団（Population）を複数のサブ母集団（Sub Population）に分割し，各サブ母集団内では従来の GA の処理を行う．このサブ母集団をプロセッサに割り当てることによって並列処理を行う．各サブ母集団では従来の GA と同様の処理を行うが，一定期間に到達するとサブ母集団間において個体の交換を行う．この操作を移住（Migration）と呼び，個体の移住を行わないものを並列 GA(Parallel Genetic Algorithm:PGA) と呼ぶ．移住にはさまざまな方法が提案されており，その代表的な例として「島モデル」,「踏み石モデル」,「近傍モデル」の 3 つがある．以下にそれぞれのモデルについて説明する.

島モデル (Island Model)

島モデルでは母集団をサブ母集団に分割し，各サブ母集団ではそれぞれ GA の処理を行う．移住世代になると，複数存在するサブ母集団から，移住相手となるサブ母集団をランダムに選び，一定量の個体を移動させる．これを最大指定世代数繰り返す．図 2.12(a) にその様子を示す.

踏み石モデル (Stepping Stone Model)

踏み石モデルは島モデルと同様に，移住を行うまで各サブ母集団内ではそれぞれ GA の処理を行う．移住方法は島モデルと異なり，移住相手はサブ母集団ごとにあらかじめ決まっており，その多くは，サブ母集団の近傍から選ばれる．図 2.12(b) にその様子を示す.

近傍モデル (Neighborhood Model)

近傍モデルでは，プロセッサは一つまたは少数の遺伝子を管理して GA の処理を行い，移住はプロセッサの近傍のみで毎世代行う．図 2.12(c) にその様子を示す.

移住は，移住間隔（Migration Interval）と移住率（Migration Rate）というパラメータによって制御され，移住間隔は移住を行う世代間隔，移住率はサブ母集団に属する全個体のうち，移住する個体の割合を示す値である．分散 GA は並列処理に加え，移住という操作を加えることによって，以下のような特長を持つ.

計算時間の短縮

分割された母集団は各プロセッサにより GA の処理が行われる．プロセッサ間通信は移住の世代のみで行われるので，オーバーヘッドも少なく，分散処理による計算時間の短縮が望める.

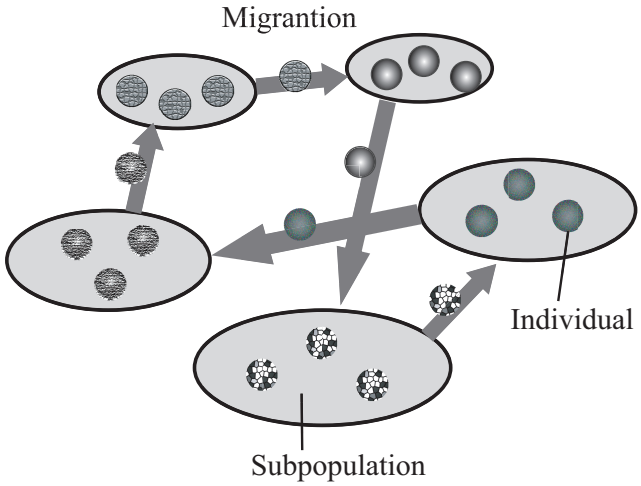
多様性の保持と早熟収束の回避

分散 GA では GA と比較して，収束までに多くの世代が必要となる．これは母集団を分割することによって，各サブ母集団上で固有の染色体が育ち，集団の多様性が保持されるためである．それによって探索空間内の十分な探索が行われた後に，解への収束が始まるので，GA の問題の一つである早熟収束問題が回避しやすくなる．したがって，より複雑な問題への GA の適用が可能となり，アルゴリズムのロバスト性も向上する.

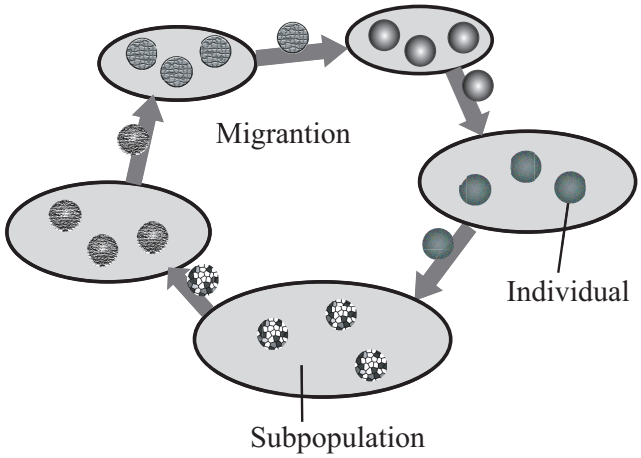
パラメータ設定の容易さ

島モデルでは，各島ごとに異なるパラメータ (交叉率，突然変異率) を設定することが可能である．Tanese が行った実験では，この方法を使用したことにより，パラメータ設定が GA ほど困難でなくなったと報告されている.

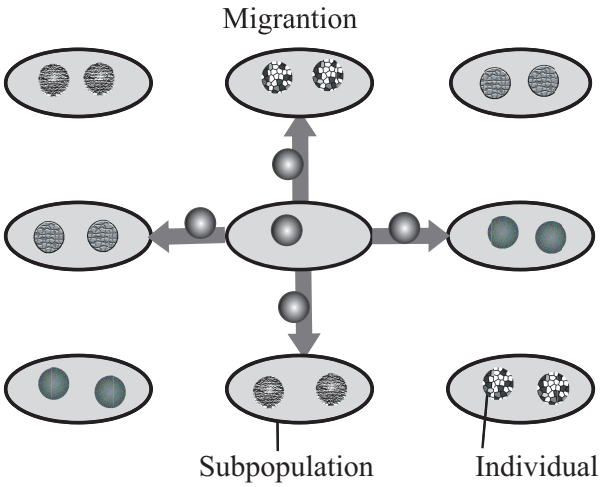
しかしながら，移住間隔や移住率は適用する問題によって最適な値が異なるため，ユーザー自身の手によって設定しなければならない．この問題を解決するために移住間隔を乱数を用いて決定する方法 [42] やエリート更新時に移住を行う方法が考案されている.



(a) 島モデルの移住の様子.



(b) 踏み石モデルの移住の様子.



(c) 近傍モデルの移住の様子.

図 2.12: 分散 GA における移住モデル.

2.4 おわりに

本章では，まずはじめに本研究で扱う階層的最適化問題と，複数の探索空間を持つ問題の定義を行った．次に，GA の概要と探索アルゴリズムについて説明し，GA の拡張である階層型 GA と分散 GA のアルゴリズムについて説明した．GA の拡張はこのほかにも多く存在する．例えば，評価部分に人間の主観的要素を取り入れることによって，人間の感性という複雑な構造を解析する対話型 GA[43] や，実行形式の異なる複数の個体群によって探索を行う個体群探索分岐型 GA[44]，複数の目的関数を持つ多目的最適化問題に有効な多目的 GA[45, 46] などがある．このように，GA は非常に多くの拡張手法が存在し，様々な分野で利用されている．しかしながら，GA は多点探索を行う手法の為，いずれの手法についても，個体数と計算時間，解の発見精度の間にトレードオフの関係が生じる．特に，階層型 GA のように，GA の処理を階層的に行う場合は，従来の GA と比較して非常に多くの計算時間を必要とする．多点探索の欠点を軽減するためには，最適解が存在しそうな場所もしくは探索空間を早期に推定し，その部分の探索を強化することである．つまり，不要な探索を避け，ある領域に個体を集めることができるので，最適解を早期に発見することが可能になる．

第3章 複数解空間競合型分散GAの提案と多項式曲線フィッティングによる検証

3.1 はじめに

本研究では，探索空間が少ない場合において，最適解の存在する探索空間とそのパラメータを同時に推定を行うことができる，複数解空間競合型分散遺伝的アルゴリズム (Multi-space Competitive Distributed Genetic Algorithm: mcDGA) を提案する．mcDGA では，探索の途中に “探索空間どうしの競合”という操作を行うことによって，複数の探索空間から解の存在する可能性が高い探索空間を推定し，その中から最適な解を発見することができる．競合の結果に基づいた個体の移住を行うことによって，従来の GA と比較して少ない世代で解を発見し，計算時間を短縮することが期待できる．本章では，多項式曲線フィッティング [47] を用いて mcDGA の処理を説明し，競合の影響と規則について述べる．このとき，解発見までの世代数を並列 GA と比較し，mcDGA の有効性を示す.

本章では，本研究において提案する mcDGA について説明する．3.2 節では，mcDGA の概要について説明し，多項式曲線フィッティングに mcDGA を適用した場合の mcDGA の性能について評価する．3.3 節では，本章のまとめである．

3.2 複数解空間競合型分散 GA

3.2.1 概要

提案手法の概念図とフローチャートをそれぞれ図 3.1，図 3.2 に示す．提案手法では，分散 GA と同様，GA の母集団をいくつかのサブ母集団に分割し，図 3.1 と図 3.2 より，サブ母集団は考慮する次元数と同様に S 個用意する．図 3.2 における g は競合を行う間隔を表す．各サブ母集団はそれぞれ独立して GA の操作を行い，g が設定した条件を満たした場合に競合を行う．この場合の競合とは，各サブ母集団どうしにおいて最大評価値や平均評価値などの値を比較することを示す．競合に勝利したサブ母集団には個体を追加し，敗北したサブ母集団からは個体を削除する．このとき追加する個体数は，設定世代数や問題の種類に応じてユーザが任意に決定することができる．追加された個体はサブ母集団内で最も優秀な個体の周辺に配置される．また，削除される個体は，

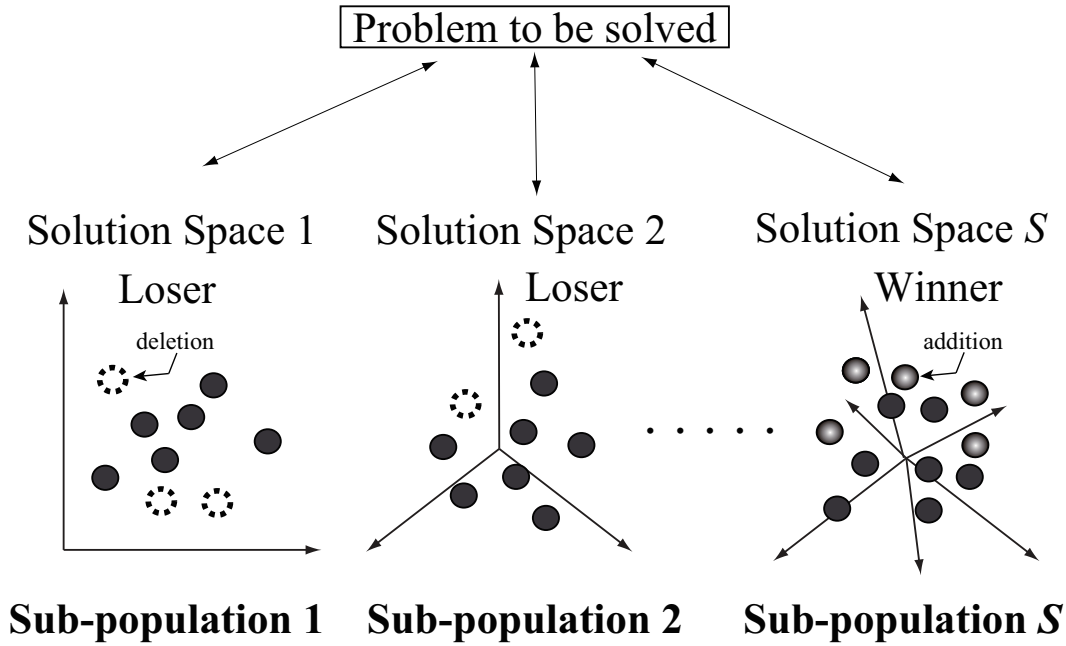


図 3.1: 複数解空間競合型分散遺伝的アルゴリズムの概念図.

比較する値が低いサブ母集団ほど選ばれやすい．したがって，競合に勝ち続けるほど個体数は増加し，解を発見する可能性が高くなる．2 章で述べたように，GA は多点探索という特長から，個体数が多いほど解を発見する可能性は高くなるが，計算時間が増加するという問題がある．つまり，計算時間を削減するためには，できるだけ少ない個体数で探索を行う必要がある．提案手法では競合により全体の個体数を変えることなく，ひとつのサブ母集団に個体を集中させることができるので，個体が集中したサブ母集団では，早期に解を発見し，計算時間を短縮することが期待される．

3.2.2 多項式曲線フィッティングによる検証

本小節では，多項式曲線フィッティング [47] を用いて，mcDGA の動作と有効性，競合の影響について評価する．

多項式曲線フィッティングは，実験的に得られたデータや制約条件に最もよく当てはまる近似曲線を求める問題である．多項式曲線フィッティングはモデル選択の代表的な例の一つであり，できるだけ少ない次元数でデータを表現するモデルを選ぶ必要がある．つまり，多項式曲線フィッティングは，関数の次数と各係数を求める問題であり，M 次関数の係数を求める場合，探索空間は $M+1$ 次元空間となる．一般的に，多項式曲線フィッティングは最小二乗法などを用いて解析的に解くことができる．

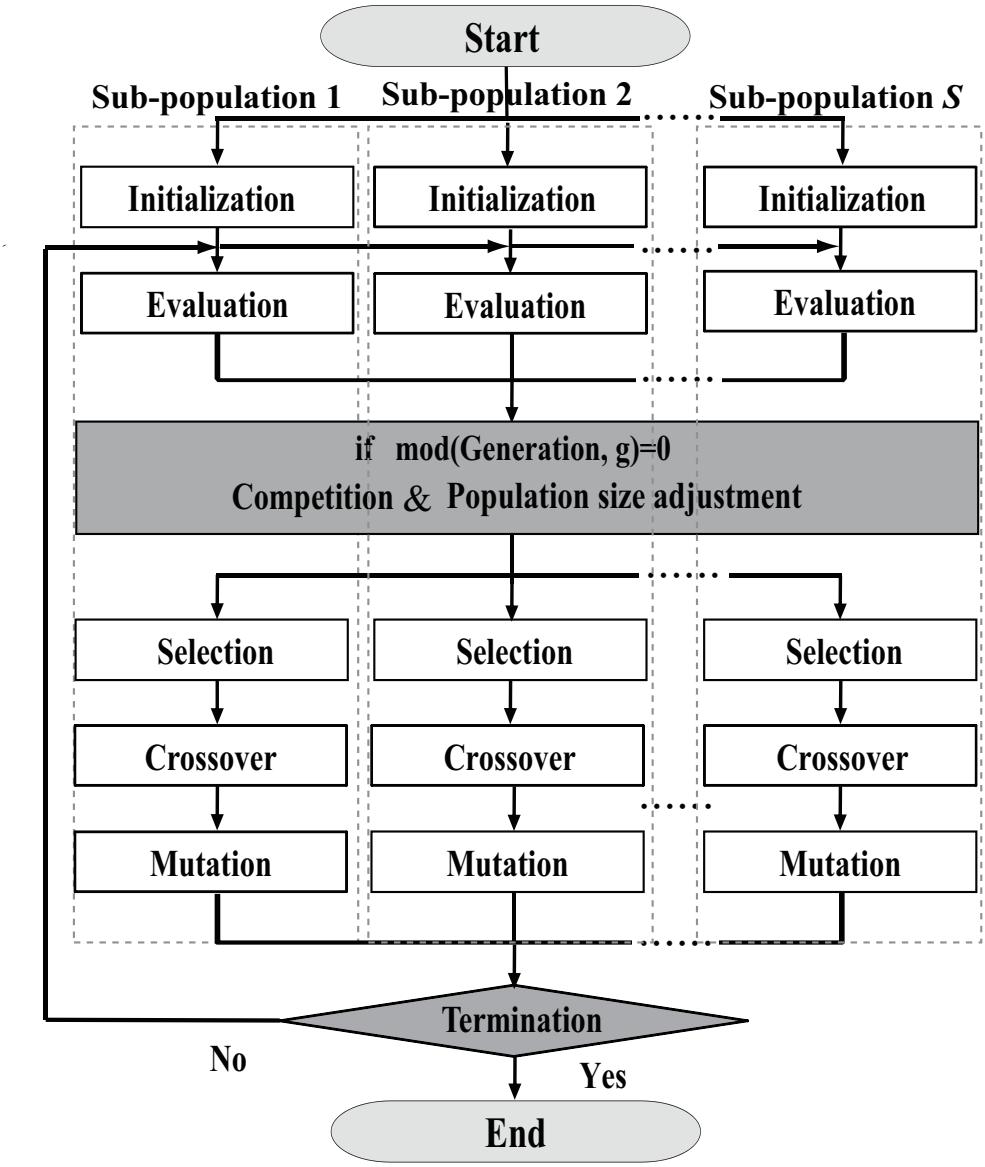


図 3.2: 複数解空間競合型分散 GA のフローチャート.

シミュレーション条件

シミュレーションでは，正解の関数として 5 次関数 $t = -0.055x^5 - 0.026x^4 + 0.66x^3 - 0.12x^2 - 1.83x + 0.21$ と 7 次関数 $t = -0.03x^7 - 0.01x^6 + 0.39x^5 + 0.16x^4 - 1.29x^3 - 0.55x^2 + 0.21x + 0.40$ を用いた．トレーニングデータ $\{(x_i, t_i) \mid i = 1, \ldots, N\}$ はそれぞれの関数にガウスノイズ N(0, 0.3) を加えたものである．図 3.4(a), 3.4(b) にシミュレーションに用いた関数とトレーニングデータを示す．この問題におけるモデルは次の式を用いた．

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M, \tag{3.1}$$

$$\boldsymbol{w} = (w_0, \ldots, w_M)^T, \tag{3.2}$$

GA における個体は関数の係数ベクトル $\boldsymbol{w}$ とし，係数の数はモデルの次数 M によって決まる．今回のシミュレーションでは，用意するサブ母集団（探索空間）を 8 個とし，各サブ母集団における関数の次数は，1 次，3 次，5 次，7 次，9 次，11 次，13 次，15 次とした．個体の評価には，赤池情報量基準 (AIC) [48] を適用し，個体の評価 E は次式によって決まる．

$$E = \exp(-e/\gamma), \tag{3.3}$$

$$e = N(\log(2\pi \frac{Se(\boldsymbol{w})}{N} + 1) + 2(M + 1 + 2)), \tag{3.4}$$

$$Se(\boldsymbol{w}) = \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2, \tag{3.5}$$

この評価関数は，トレーニングデータ t と推定値 $y(x_n, \boldsymbol{w})$ の二乗誤差の和 $Se(\boldsymbol{w})$ が小さくなる個体ほど高い評価を与えるものである．また，γ は評価の厳しさを決めるパラメータであり，ここでは（データ点数/5）とした．

並列 GA は，分散 GA と同様に，複数のサブ母集団によって探索を行うが，競合を行わないものを指す．シミュレーションでは，mcDGA と並列 GA を多項式曲線フィッティングに適用し，それぞれの評価値の上昇の程度について比較する．個体は初期化によってランダムに配置し，選択にはルーレット法，交叉には BLX-α 交叉，突然変異には一様突然変異をそれぞれ用いた．交叉率，突然変異率はそれぞれ，0.8，0.1 とし，各サブ母集団における初期個体数は，それぞれ 50 個体，探索終了世代は 100000 世代とした．

個体数の影響

図 3.3 に個体数の影響を示す．GA の探索は多点探索によって行われるので，個体数が多いほど解の発見までに要する世代数は短くなる．つまり，早期に評価値が上昇することを意味する．図 3.3 より，個体数の多いほど評価の上昇が早いことが確認できた．

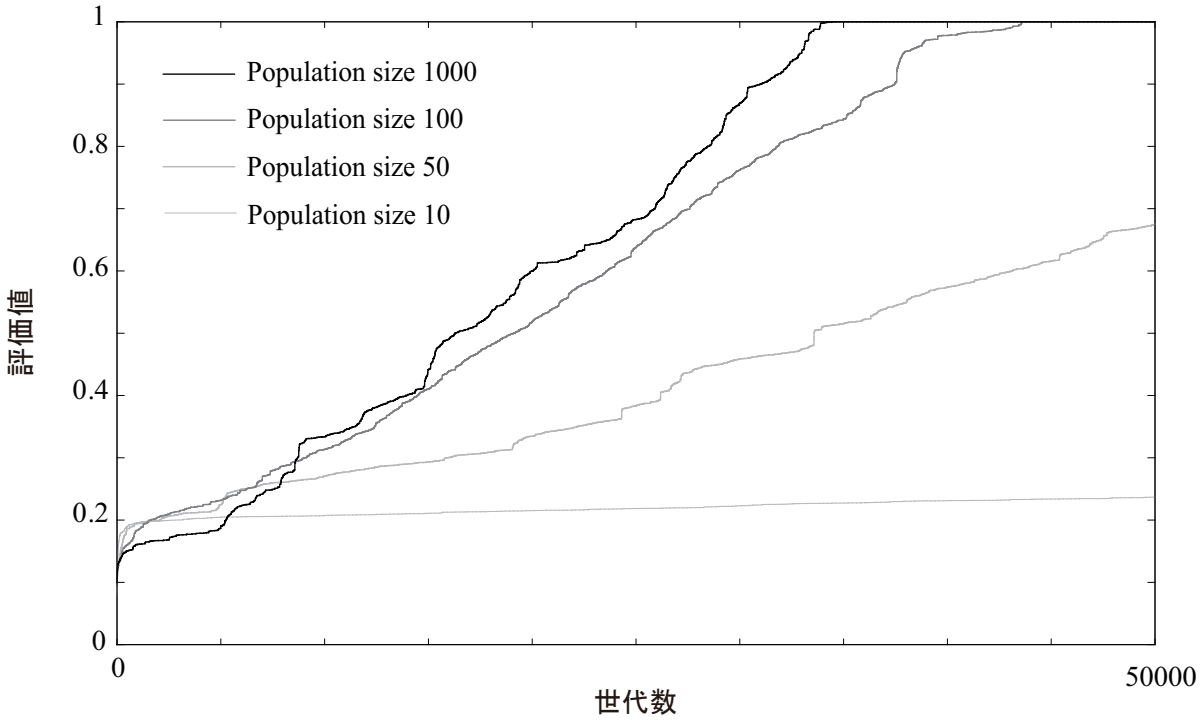
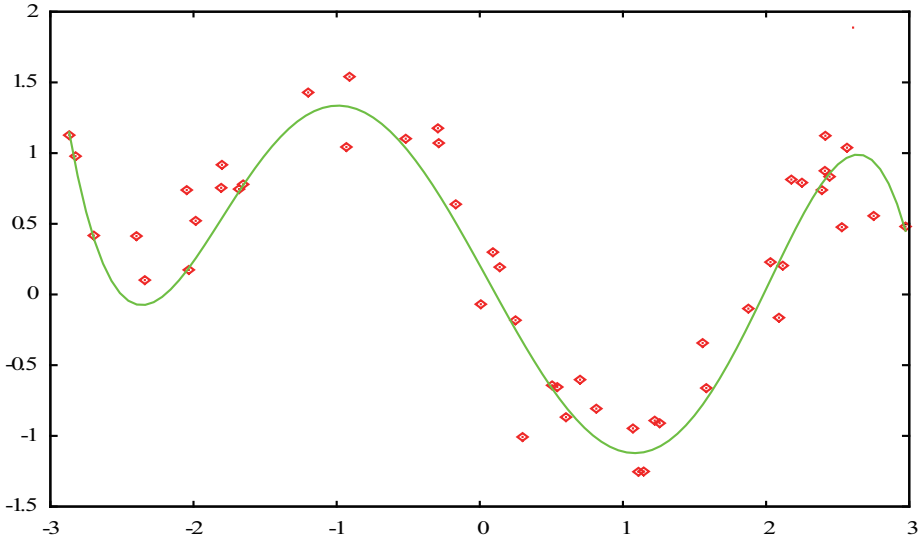


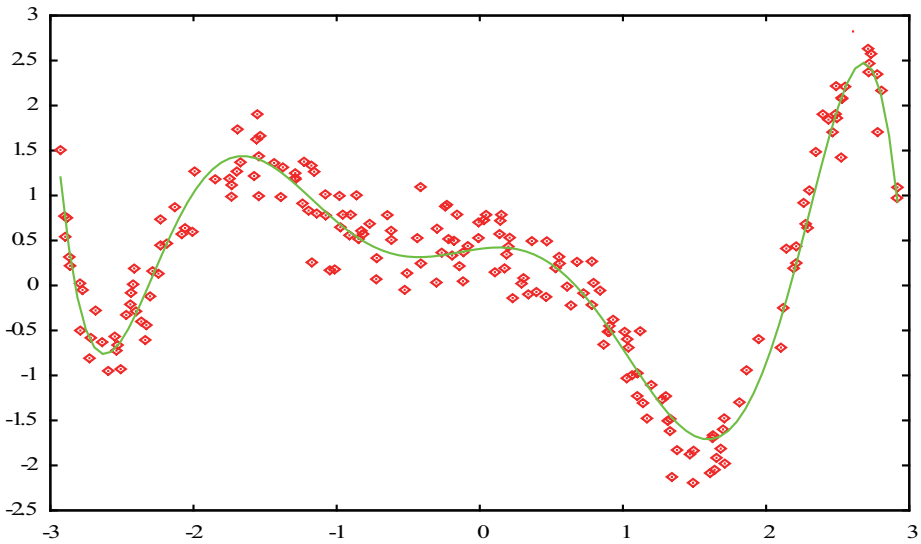
図 3.3: 個体数の影響.

競合の規則

本手法では，競合の間隔が重要なパラメータとなる．競合の間隔は，短く設定すると個体数の増加が早く，競合による影響は大きくなる．しかしながら，その場合，解の発見が困難な高次元空間を探索するサブ母集団では，早期に個体が絶滅してしまい，解を発見できない可能性がある．競合間隔が短い場合の探索結果を図 3.5 に示す．図 3.5 は 5 回試行した際の結果である．図 3.5 より，競合の間隔が短い場合，個体は低次元の探索空間を探索するモデルに集中する．しかしながら，間隔が長すぎると，競合における個体の増加の影響が少なくなる．したがって，競合の間隔は適用する問題や，初期個体数に応じて適切な値を設定しなければならない．今回のシミュレーションでは，高い次元のモデルにおける早期の個体の枯渇を避けるため，以下の 2 つの競合方法を検討した．1) 探索初期は競合を行わず，ある世代以降から競合を開始する，2) 探索が進むにつれて探索空間を徐々にひとつに絞る．図 3.6(a) に 1) の競合方法，図 3.6(b) に 2) の競合方法のイメージ図を示す．今回の競合では，比較する値として各モデルの最大評価値とする．また，競合を行うタイミングとして，いずれかのモデルの最大評価値の更新した場合に競合を行うこととした．



(a) 5 次関数.



(b) 7 次関数.

図 3.4: シミュレーションに使用した関数.

5 次関数におけるシミュレーション結果

まず，競合を開始する世代を 1000 世代以降としてシミュレーションを行った．結果として，5 回試行中正解のモデルが競合に勝利したのは 1 回のみであり，内訳としては 3 次のモデルが 4 回である．また，競合を開始する世代数を増やすほど，高次のモデルが競合に勝利するという傾向が見られた．これは，高次のモデルが解を探索するために必要な世代数に到達していることを意味する．しかしながら，5 次関数を探索するモデルにおいて，1000 世代は解を得るために不十分で

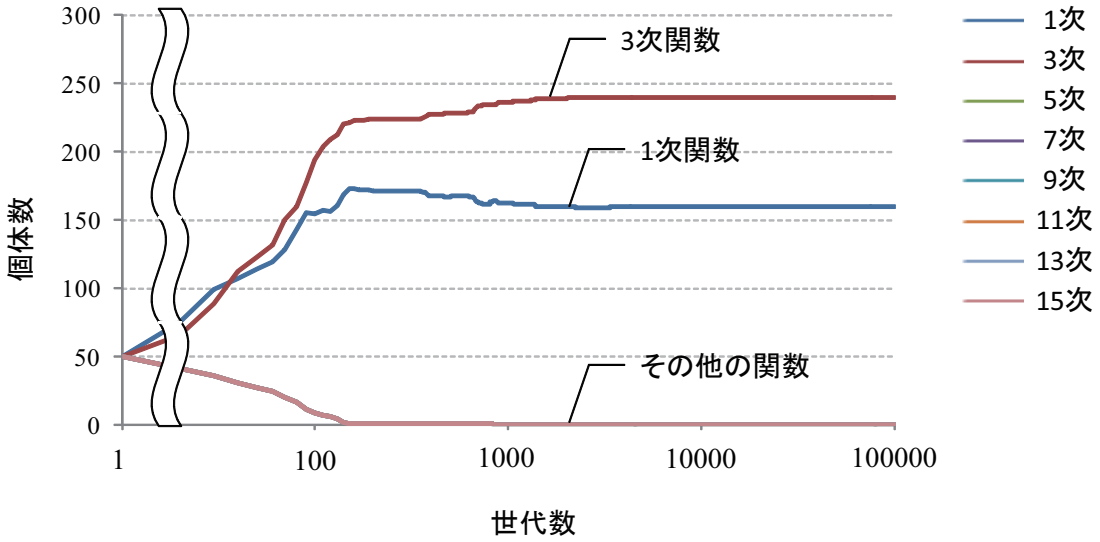


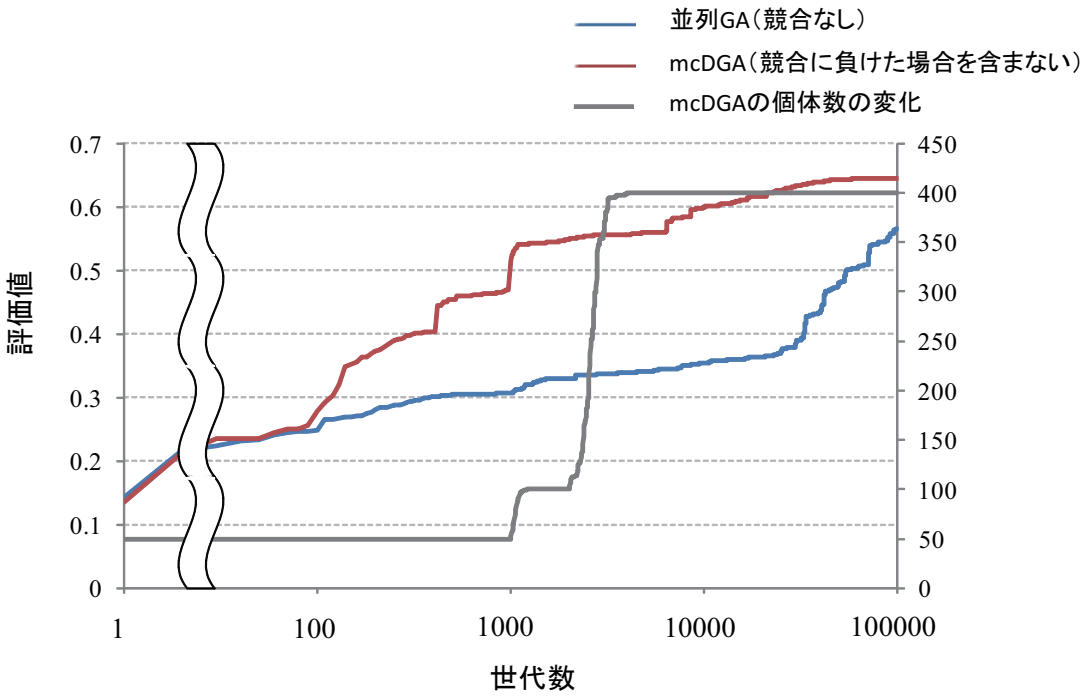
図 3.5: 競合の間隔が短い場合の個体数の変化.

あったと考えられる．また，この方法では，規定の世代後に個体が一気に減るため，その時々の評価値に大きく影響し，安定性に欠ける．次に，上述の方法同様，1000 世代までは競合を行わず，1000 世代後に評価の高い上位半分サブ母集団を競合によって決定する方法を検討した．この方法では，1000 世代以降は，評価値の高い上位半分のサブ母集団には個体を追加し，それ以外のサブ母集団の個体数を減少させる．これによって，1000 世代時に 1 番ではないがある程度評価の高いサブ母集団が生き残ることができ，さらに個体数も増加するので，正解のサブ母集団が含まれていれば，解を発見する確率は高くなる．1000 世代以降に上位半分を決定する操作は 2000 世代まで続ける．つまり，上位半分のサブ母集団は，最大 100 個の個体で 1000 世代探索を行うことができる．2000 世代以降は，上位半分のサブ母集団から，もっとも評価値の高いサブ母集団に個体を集め，最終的に生き残った者を勝者とする．この方法を用いて，5 次関数を推定した場合，5 回中 4 回は競合に勝利することができた．本シミュレーションでは，試行ごとに異なるのトレーニングデータを用いているため，その分布によっては 5 次関数での推定が有利でない場合もある．1 回の敗北は，データの分布における影響が少なからずあると推測する．図 3.7(a)，3.7(b) に，mcDGA と並列 GA の評価値の比較，推定された曲線を示す．図 3.7(a) より，競合に勝利した試行の評価値だけをみれば，並列 GA よりも良い解を早期に得られていることが確認できる．つまり，mcDGA では，競合によって個体数を増加させることで，並列 GA と比較して短い世代数で解を発見できることを示した．さらに 3.7(b) より，mcDGA は最小二乗法によって求められた曲線とほとんど一致していることが確認できる．よって，シミュレーションによって，mcDGA は並列 GA と比べて早期に良質な解を発見できることが示された.

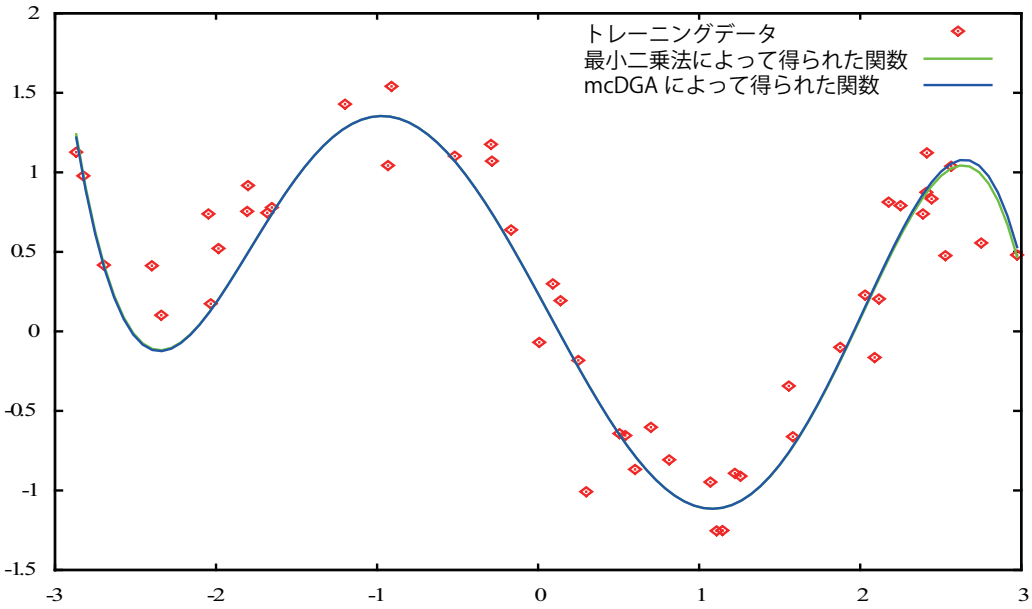
(a) ある世代以降から競合を開始する競合方法.

(b) 探索空間を徐々にひとつに絞る競合方法.

図 3.6: 検討した競合方法のイメージ.



(a) mcDGA と並列 GA の評価値の比較.



(b) 最終的に得られた曲線.

図 3.7: mcDGA による推定結果.

7 次関数におけるシミュレーション結果

正解を 7 次関数とした場合のシミュレーションでも，5 次関数で用いた方法と同じ方法を適用した．結果として，1000 世代以降に競合を行う方法では，7 次の関数を探索するサブ母集団は一度も競合に勝利することなく，9 次関数が 3 回，5 次関数が 1 回，3 次関数が 1 回だった．これは今回使用した関数がやや中途半端な形状をしているということも影響しているのではないかと推測できる．今回の関数にノイズを付加した場合，5 次関数とも 9 次関数ともとれるはっきりしない形状になる．5 次関数の推定においてうまくいったといえる， 1000 世代後に評価の高い上位半分サブ母集団を競合によって決定する方法を適用した．結果として，1 から 9 次までのサブ母集団がそれぞれ 1 回ずつ競合に勝利した．これは，上位半分に絞ることで，探索に時間のかかる高次のサブ母集団は除くことができたが，正解のサブ母集団にとっても十分な世代数ではなかったのではないかと考察できる．7 次関数においても，ノイズは試行ごとに異なるので，分布の仕方によっては正解のサブ母集団の探索が困難になる場合もある．

3.3 おわりに

本章では，提案手法である mcDGA を多項式曲線フィッティングに適用し，その性能について評価した．正解の関数として，5 次関数と 7 次関数を用い，シミュレーションによって，競合における個体数と競合を行うタイミングについて検討した．競合のルールとして，1) 探索初期は競合を行わず，ある世代から競合を始める方法，2) 探索空間を徐々に一つに絞る方法について検証を行った．正解の関数が 5 次関数の場合においては，2) の競合の方法が有効であり，1000 世代から競合を開始し，2000 世代までは上位半分，2000 世代以降はひとつに絞る方法において，適切な探索空間と解を求めることができ，競合を行わない並列 GA と比較して早期に解を発見できることが確認された．これは競合によって，不要な探索を避け，解の存在する可能性の高い探索空間に個体を集中させたことが非常に影響している．しかしながら，正解が 7 次関数の場合は，同様の競合の方法では，正解でない探索空間に個体が集中してしまい，安定して解を求めることができなかった．これは競合の影響に関係なく，単に今回のシミュレーションで用いた個体数や世代数が適切でなかったことも原因の一つとして考えられる．7 次関数は 5 次関数と比較して，探索空間の大きさが極端に大きく，そもそもの探索自体が容易でない．つまり，それぞれの探索空間の大きさが同程度の問題には，今回のシミュレーションで用いた競合方法は有効だが，探索空間の大きさや複雑さが大きく異なる問題では，今回の競合方法は有効ではないということが推測できる．探索空間の大きさや複雑さが大きく異なる問題において，競合の有効性を示すためには，各探索空間において必要な計算リソースを事前に検証する必要がある．そして，検証よって求められたリ

ソースに基づき，初期個体数や世代数，競合のタイミングなどを決定しなければならない．今後は，探索空間の大きさや複雑さに応じて，必要な個体数や世代数を明らかにし，mcDGA をツールとして使用する際の指標を示す必要がある．

第4章　階層的複数解空間競合型分散GAの提案と階層的な組み合わせ最適化問題への適用

4.1　はじめに

本章では，本研究において提案した mcDGA を階層的に改良した階層的 mcDGA をそれぞれ異なる組合せ問題に適用し，HmcDGA の有効性を検証する．本研究において開発した mcDGA では，探索空間の数を事前に決めておく必要があり，探索空間の数が少数の場合しか対応できない．したがって，膨大な数の探索空間を考慮しなければならない問題では，探索空間の最適化を行う必要があり，このような問題に対して mcDGA は有効ではない．本研究では，階層型 GA の概念に基づき，mcDGA を拡張した階層的 mcDGA(Hierarchical mcDGA：HmcDGA) を提案する．HmcDGA は階層型 GA と同様，上位層と下位層の 2 層から構成され，上位層では探索空間の最適化，下位層では各探索空間における解の探索を行う．探索の途中，mcDGA と同様に “探索空間どうしの競合”による個体数の増減によって，解の存在する可能性の高い探索空間に個体を集める．また，HmcDGA では，問題そのものを階層的に分割するため，各解層において解決する問題は簡単な問題に置き換えられる．つまり，複雑な問題を簡単な問題に置き換えることができるので，問題の難易度に関わらず，様々な問題に対して有効である汎用的な最適化手法であると期待される．本章では，HmcDGA を複数車両配送計画問題に適用し，競合を行わない階層型 GA と比較して，短時間かつ高精度で解を発見できることを示す．次に，Flexible Job-shop Scheduling 問題に適用し，GA をベースとした他手法と同等の解を得られることを示し，HmcDGA の解探索能力と汎用性について考察する．

4.2 節では，HmcDGA の概要について説明し，mVRP を用いて HmcDGA の性能を評価する．

4.3 節では，HmcDGA を Flexible Job-shop Scheduling 問題に適用し，得られた結果について考察を行う．

4.4 節は，本章のまとめである．

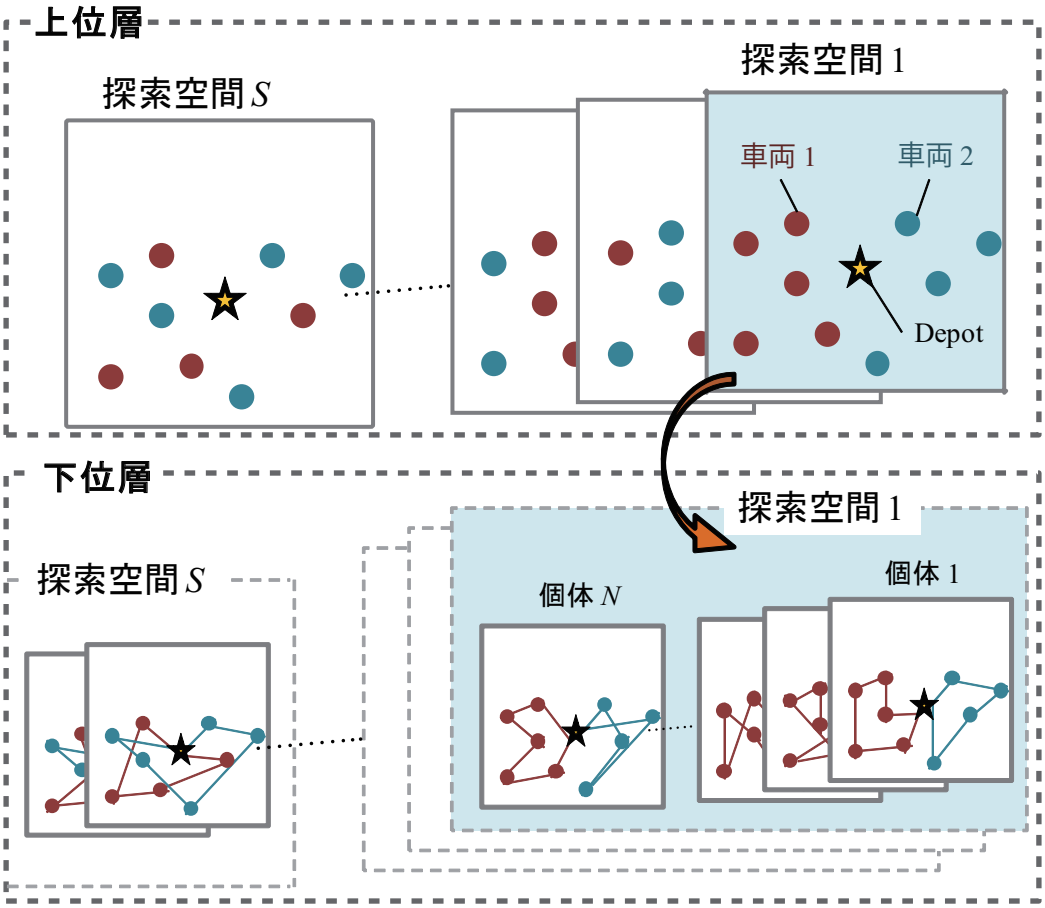


図 4.1: HmcDGA を mVRP に適用した場合の概念図.

4.2　階層的複数解空間競合型分散 GA

4.2.1　概要

HmcDGA は上位層と下位層の 2 層によって構成され，上位層では探索空間の最適化，下位層では各探索空間における最適解を探索する．mcDGA と同様に，探索の途中に探索空間どうしの競合を行い，評価値に基づいて各探索空間の個体数を調節する．HmcDGA の概念図とフローチャートをそれぞれ図 4.1，図 4.2 に示す．図 4.1 は，mVRP に適用した場合の概念図であり，図中の赤丸は車両 1，青丸は車両 2，星印は各車両の出発地点（Depot）である．上位層において，破線によって囲まれた領域は一つの探索空間を示し，下位層において，実線によって囲まれた領域は一つの個体を示す．一つの探索空間では N 個の個体によって探索が行われる．したがって，図 4.1 において，解空間全体の個体数は $S \times N$ 個となる．図 4.2 において，左側の破線で囲まれた部分が上位層，右側の破線で囲まれた部分が下位層となる．図 4.2 より，個体の評価は下位層のみにお

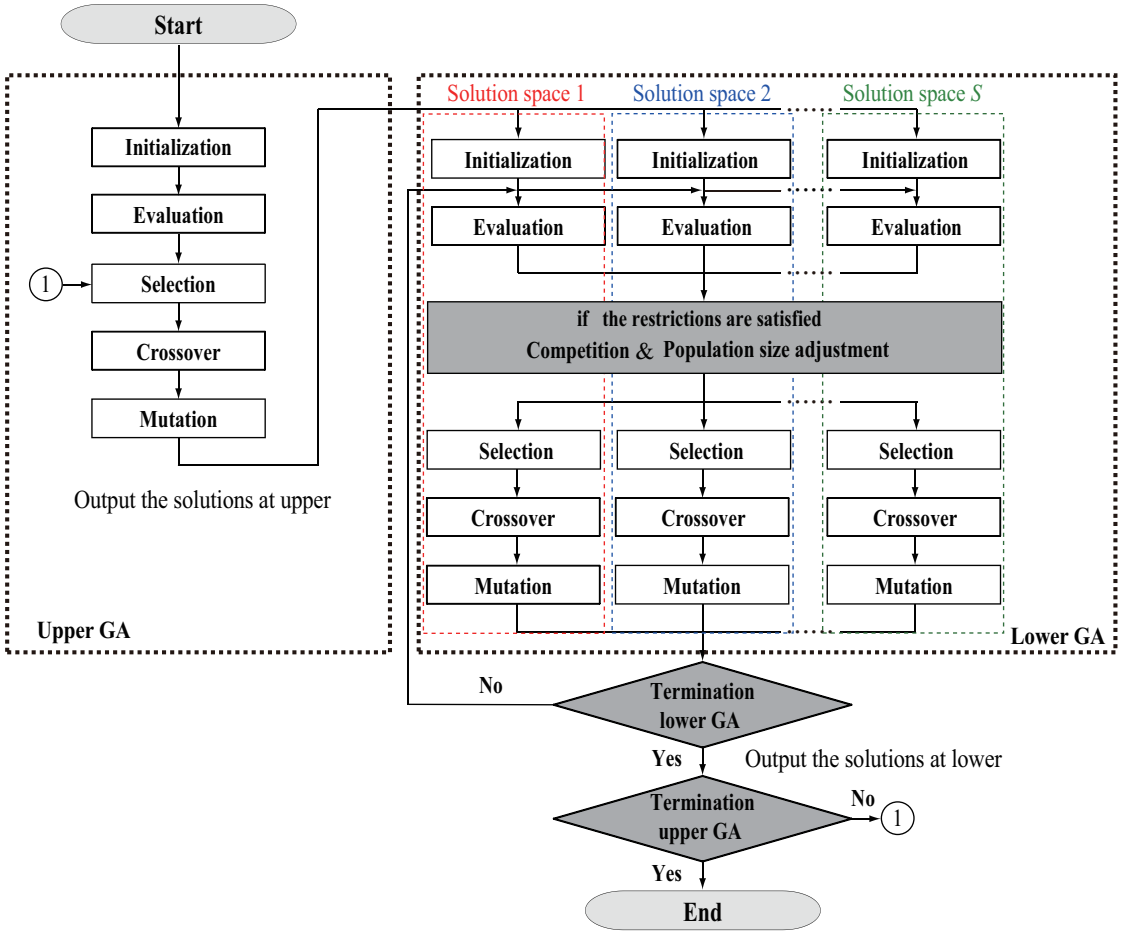


図 4.2: HmcDGA のフローチャート.

いて行われる（初期個体の評価を除く），評価は各探索空間において独立に行われる．競合は下位層において行われ，予め設定された競合の規則に基づいて各探索空間の最大評価値を比較し，各探索空間の個体数を調整する．競合の規則とは，競合を行うタイミングや競合後の個体数の調整方法のことを指す．競合の規則は，適用する問題や上位層における個体数，下位層における最大規定世代などに応じて，ユーザーが適宜設定する必要がある．例えば，競合を行うタイミングを，いずれかの探索空間において最適解が更新された世代としたとき，毎世代競合が行われる可能性がある．その場合，競合の結果は各探索空間の初期値に大きく影響され，各探索空間において良質な解が発見される前に探索が中止されてしまうことがあるので，最適な解空間の推定は難しくなる．したがって，ある世代までは競合を行わないなどの規則を設定する必要がある．上位層における GA の処理は，下位層において探索が終了した際の各探索空間の最大評価値に基づいて行われる．つまり，下位層における評価値が高い探索空間ほど最適解の存在する可能性の高い探索空間となる．

4.2.2 複数車両配送問題への適用

本小節では，複数車両配送計画問題（Multiple Vehicle Routing Problem : mVRP）を用いて，HmcDGA の動作と有効性を評価する．

mVRP は，一般的に知られている配送計画問題（Vehicle Routing Problem : VRP）[16] の拡張であり，複数の車両を用いて全ての顧客をちょうど一回ずつ訪問するような経路集合から，コストが最小の経路を求める問題である．この問題は，代表的な最適化問題の一つであり，非常に実用性のある問題である．例えば，この問題の応用として，郵便や新聞の配達，スクールバスのスケジューリング，コンビニの商品配達などが挙げられる．近年では，荷物の積み下ろしに要する時間や，配送スタッフの勤務日数や給与などが制約条件に含まれる問題も検討されており，非常に活発に研究されている [49, 50]．mVRP の必要条件は以下のようになっている．

- 車両は顧客を訪問した後，出発地点に戻る．
- 顧客の位置座標は既知である．
- いずれかの車両によって，すべての顧客を一度だけ訪問する．
- 車両の総移動距離を最小化し，顧客間の距離はユークリッド距離によって算出される．

この問題を解決するために，これまで様々な方法が検討されてきた．例えば，各車両が巡回する顧客をクラスタリング手法によって決定する方法 [51] では，探索空間が大幅に縮小されるので，容易に解を発見することができるが，積載量や時間窓などの制約によっては適切にクラスタリングすることができない．したがって，解の発見は困難になる．遺伝的アルゴリズムを用いた解決方法も多く提案されており，その中の一つに階層型 GA を用いた解決方法がある [52]．この手法は，HmcDGA と同様に，上位層と下位層の 2 層から成り，各層で GA の処理を行う．しかしながら，GA は多点探索という特徴をもつため，解の発見には多くの計算時間を必要とする．したがって，GA を上位層，下位層ともに GA を用いる手法は従来の GA よりも多くの計算時間を必要とする．また，GA は広大な解空間を探索するので，mVRP のような最短経路を求める問題を早期に解決することを得意としない．HmcDGA では，競合の結果に基づく個体数の調整によって，計算時間を大幅に削減することができる．本小節では，HmcDGA と階層型 GA の計算時間と解の発見精度について比較する．

顧客	C1	C2	C3	C4	C5	C6	C7	C8	C9
車両1	1	0	1	1	0	0	0	0	1
車両2	0	1	0	0	1	1	1	1	0

(a) 上位層における個体の表現.

巡回順序	1	2	3	4	5
車両1	C1	C4	C9	C3	
車両2	C8	C5	C2	C7	C6

(b) 下位層における個体の表現.

図 4.3: HmcDGA を mVRP に適用した場合の個体の表現.

表 4.1: シミュレーションに用いた遺伝的操作.

	Upper level	Lower level
Selection	Roulette selection	
Crossover	Uniform	PMX
Mutation	Translocation	

個体の表現

HmcDGA をこの問題に適用した場合，個体の表現は上位層と下位層で異なる．図 4.3 (a)，(b) に上位層および下位層における個体の表現を示す．上位層では，個体はバイナリで表現され，配列の番号が顧客番号に相当する．配列の値が 1 のとき，その配列の番号の顧客を巡回する．つまり，図 4.3 (a) において，車両 1 が巡回する顧客は C1，C3，C4，C9，車両 2 が巡回する顧客は C2，C5，C6，C7，C8 となる．また，下位層では，個体は顧客番号によって表現される．図 4.3 (b) において，左側から順に顧客を巡回する．したがって，図 4.3 (b) の場合，車両 1 は C1，C4，C9，C3 の順に巡回し，車両 2 は C8，C5，C2，C7，C6 の順に巡回する．

遺伝的操作

HmcDGA を mVRP に適用した場合，上位層と下位層の個体の表現はそれぞれ異なるので，各層において個体の表現に対応した遺伝的操作を適用する必要がある．表 4.1 にシミュレーションで

用いた遺伝的操作を示す．巡回セールスマン問題（Traveling Salesman Problem : TSP）や VRP のような最短経路問題では，個体を巡回する都市もしくは顧客の番号を用いることがあり，その場合特殊な交叉方法が用いられる．本シミュレーションでは，最短経路問題における交叉方法として最も一般的な Partially Matched Crossover (PMX)[35] を用いた．その他の交叉の方法として，Cycle Crossover (CX) [36] や Order Crossover (OX) [37] などがある．また，個体の評価 F は以下の式によって求められる．

$$F = \frac{1}{d_{max}}, \tag{4.1}$$

ここで， d_{max} はすべての車両における総移動距離であり， d_{max} が小さい個体ほど高い評価が与えられる．つまり，この場合，各車両の走行距離が近くなるように収束していく．

シミュレーション条件

本シミュレーションにおいて扱う問題は，Depot 数 1，顧客数 20，総車両数 2 をとした．表 4.2 において使用したパラメータを示す．上位層において，階層 GA と HmcDGA の個体数は 10 とし，探索終了世代は 30000 世代とした．下位層における個体数や，HmcDGA における競合のタイミングは図 4.4 の結果に基づいて決めた．図 4.4 は個体数と世代数及び最適解の発見確率について表したものであり，灰色のグラフは 10 回試行で得られた最適解の平均を示しており，黒線は最適解の発見確率を示している．競合の影響をよりわかりやすく示すためには，できるだけ個体数は少なくした方がよい．図 4.4 より，個体数が 10 の場合は，いずれの世代数においても解の発見率が低く，推定された経路も大きい．しかしながら，個体数が 20 以上になると，最適解の発見確率は大幅の向上する．また，世代数に注目すると，世代数が増えるにつれて解の発見精度が向上するのは当然のことであるが，100 世代ではほとんど解が発見できていないにも関わらず，200 世代以降はそれぞれの個体数において解の発見精度が大幅に向上している．特に，個体数が 20 の場合においては，300 世代以降から得られた解の経路の距離にそれほど大きな変化がなくなっている．したがって，図 4.4 の結果より，HmcDGA における初期個体数は 20 とし，競合開始の世代を 300 世代とした．図 4.5 に，シミュレーションに用いた競合の規則のイメージを示す．今回のシミュレーションでは，多項式曲線フィッティングの場合に有効であった，探索空間を徐々にひとつに絞る方法を採用した．今回用いた競合では，300 世代から 500 世代までの間に探索空間の数を半分に絞り，500 世代以降は，探索空間を 1 つに絞る．図 4.4 の結果から，300 世代から 500 世代までの間における各探索空間の最大個体数は 30 個体とし，500 世代以降の探索空間の最大個体数は 50 個体とした．階層型 GA の下位層における個体数は HmcDGA と比較するために 20 および 50 とし，探索終了

表 4.2: シミュレーションに用いた階層 GA と HmcDGA のパラメータ.

		階層 GA	HmcDGA
Upper level	Population size	10	
	Generation	30000	
	Crossover rate	0.8	
	Mutation rate	0.3	
Lower level	Population size	20, 50	Variable
	Generation	500, 1000	1000
	Crossover rate	0.8	
	Mutation rate	0.1	

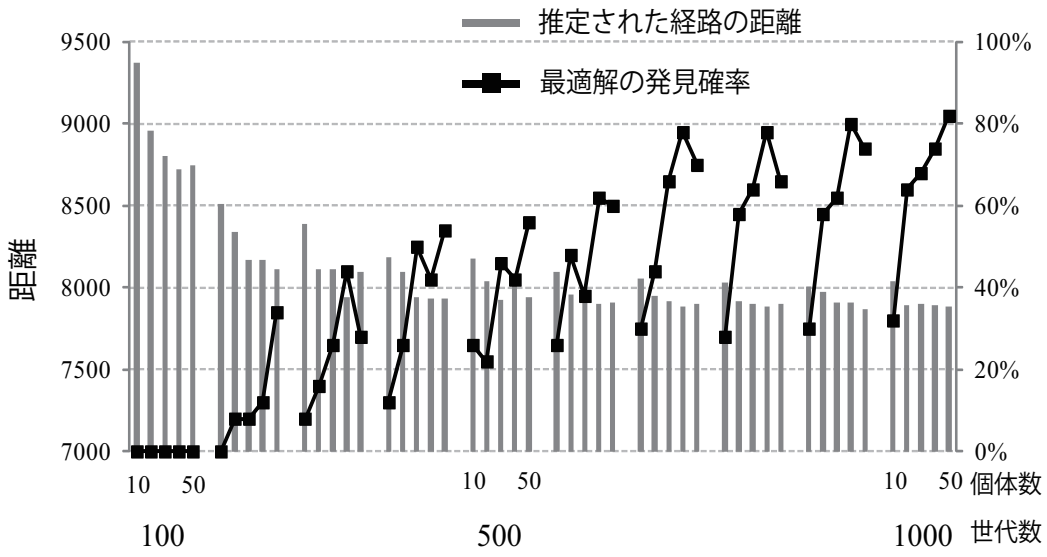


図 4.4: 個体数と世代数および最適解の発見精度の関係図．灰色のグラフは 10 回試行で得られた最適解の平均を示しており，黒線は最適解の発見確率を示している．

世代は 500 および 1000 とした．つまり，(i) 個体数 20，世代数 500，(ii) 個体数 20，世代数 1000，(iii) 個体数 50，世代数 500，(iv) 個体数 50，世代数 1000 の 4 通りの組み合わせでシミュレーションを行った．HmcDGA の下位層の個体数は競合の規則に基づいて適応的に調節され，探索終了世代を 1000 とした．

結果と考察

図 4.6 にシミュレーション結果を示す．試行は 10 回行い，その平均値を比較する．図 4.6 において，左軸は最終的に得られた解，つまり 2 台の車両の移動距離を示し，右軸はシステム全体にお

図 4.5: シミュレーションに用いた競合の規則のイメージ図.

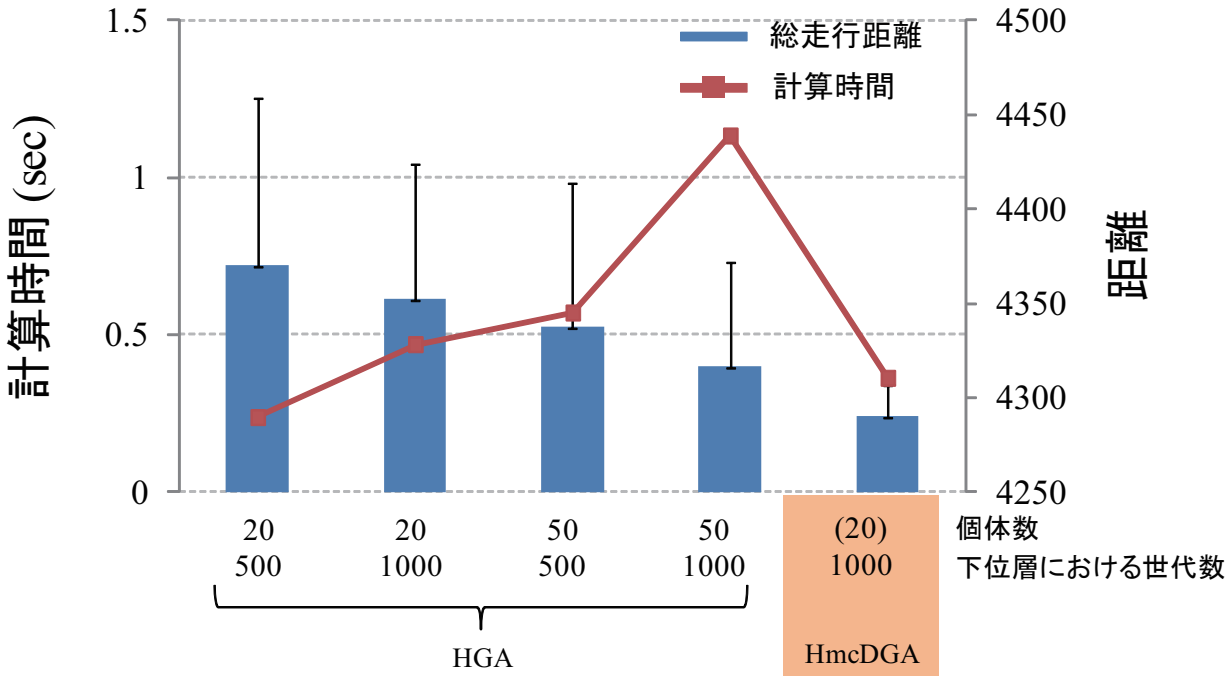


図 4.6: シミュレーション結果.

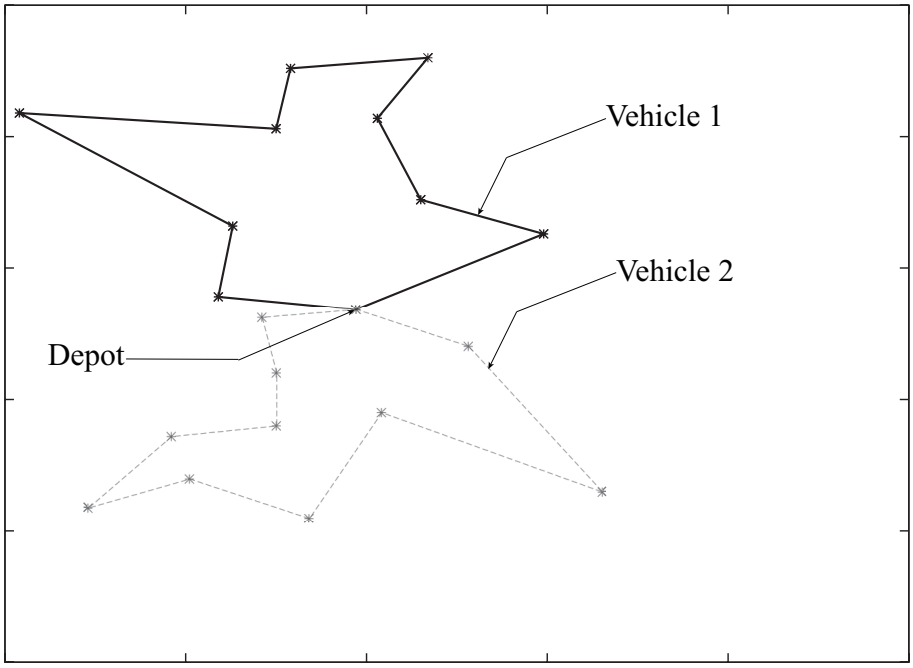


図 4.7: 得られた巡回マップ.

ける 1 世代の処理に必要な計算時間を示す．横軸において，左から 4 つのグラフは階層型 GA において，個体数と世代数をそれぞれ変えた場合の結果であり，最も右のグラフが HmcDGA の結果である．図 4.6 より，階層型 GA では，下位層における個体数が少ないほど，計算時間は短くなるが，解の発見精度は悪くなることが分かる．反対に，個体数が多い個体ほど，解の発見精度は高くなるが，多くの計算時間を必要とすることが分かる．階層型 GA では，下位層における個体数や世代数は計算時間だけでなく解の発見精度に大きく影響するので，適切な値を設定する必要がある．しかしながら，計算時間と解の発見精度はトレードオフの関係があり，適切な値を設定するのは容易ではなく，どちらかを犠牲にしなければならない．一方，HmcDGA では，短い計算時間で精度よく解を発見できていることが分かる．特に，計算時間については，階層 GA において最も少ない計算時間の組み合わせとほぼ同じ値になった．これは，今回用いた競合規則では，図 4.5 に示すように解空間全体の個体数が減少していることが大きく影響している．階層型 GA の個体数を 20，世代数を 1000 とした場合，システム全体の 1 世代における計算時間 T_u は，以下の式によって求められる．

$$T_u = 20 \times 10 \times 1000 \times t, \tag{4.2}$$

$$= 2.0 \times 10^5 \times t. \tag{4.3}$$

ここで，t は各探索空間における GA の処理の 1 世代に必要な時間である．一方，今回のシミュレーションにおける HmcDGA システム全体の 1 世代における計算時間 T_H は以下の式によって求められる．

$$T_H = (20 \times 10 \times 300 + 30 \times 5 \times 200 + 50 \times 1 \times 500)t, \tag{4.4}$$

$$= 1.15 \times 10^5 \times t. \tag{4.5}$$

これらの式より，HmcDGA では，階層型 GA の個体数を 20，世代数を 1000 とした場合と比較して，計算時間をおよそ 30% 削減でき，図 4.6 からもこの関係性を読み取ることができる．

4.2.3　Flexible Job-shop Scheduling 問題への適用

ジョブショップスケジューリング問題（Job-shop Scheduling Problem：JSP）[53] は，複数の機械を用いて複数の仕事（ジョブ）を処理するとき，すべての仕事を完了するまでの総作業時間（メイクスパン）を最小にする各機械上における各仕事の処理順序を決める問題である．言いかえると，最も作業時間のかからないような機械のスケジュールを決める問題である．JSP は身近な実問題例であり，製品の生産や設計の段階において，大規模かつ複雑な組み合わせ問題として扱われる．Flexible Job-shop Scheduling 問題（Flexible Job-shop Scheduling Problem：FJSP）[18] は，JSP の拡張であり，仕事の各作業を処理することができる機械が複数存在する．したがって，作業を処理できる機械が増えるので，JSP と比べて総作業時間を短縮することができるが，スケジューリングは複雑になり，最適解の発見は難しくなる．これまで，FJSP を解決するために蟻コロニー最適化 [54] や PSO[55] など様々な最適化手法が適用されており，GA を用いた手法も多く適用されている．例えば，[56] では，初期個体の生成の際，全体の総作業時間が短くなるようにスケジューリングする Global Selection と，各仕事の総処理時間が短くなるようにスケジューリングする Local Selection をランダムに用いて個体を生成する方法を提案した．文献 [18] では，初期個体の精製や選択などの遺伝的操作を FJSP に特化したものに改良している．また，文献 [57] では，新しい遺伝的操作の提案だけでなく，1 つの機械の最大作業量に着目した，個体の生成法についても提案している．その他，GA と局所最適化手法のハイブリッド手法 [58] や特殊な選択方法を適用した手法 [59] など，多くの解決手法が提案されている．しかしながら，これまで良い結果を示している解決手法は，FJSP に特化したものであり，同様の遺伝的操作や概念は他の問題に対して有効であるとはいえない．例えば，GA と局所最適化手法のハイブリッド手法を FJSP に適用し

Job	J_1			J_2			J_3	
Operation	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}
Machine	M_2	M_3	M_4	M_1	M_5	M_1	M_1	M_5

(a) 上位層における個体の表現.

Sequence	1	2	3	4	5	6	7	8
Operation	O_{11}	O_{21}	O_{12}	O_{13}	O_{22}	O_{31}	O_{32}	O_{23}

(b) 下位層における個体の表現.

図 4.8: HmcDGA を FJSP に適用した場合の個体の表現.

た場合，局所最適化によって処理の順序を推定しているが，これは局所最適化手法が組み合わせ問題に適しているからであり，広大な実数値空間における最適化問題には有効ではない．また，各遺伝的操作を特殊化している問題においても，同様のことがいえる．本章では，FJSP において，これまで提案されている GA を用いた解決手法と HmcDGA の結果を比較し，HmcDGA が複雑な組み合わせ問題において，問題に特化した手法と同等の結果が得られることを示し，HmcDGA が十分な解探索能力を持つことを示す.

個体の表現

図 4.8 (a)，(b) に HmcDGA を FJSP に適用した場合の上位層および下位層における個体の表現を示す．説明を容易にするために仕事数を 2 つとする．上位の個体は，各仕事の各処理を遂行する機械の組み合わせから形成される．図 4.8 (a) の場合，仕事 1 と仕事 2 はそれぞれ 5 つずつ必要な処理があり，図の 1 行目の O_{11} は仕事 1 の処理 1，O_{21} は仕事 2 の処理 1 を意味する．つまり，このとき仕事 1 の処理 1 は機械 2，仕事 2 の処理 1 は機械 1 が適用される．コストとは，ある機械がある仕事の処理を行うために必要な時間を意味する．例えば，図 4.8 (a) において，機械 2 が O_{11} を処理するのに必要な時間は 3 であり，機械 1 が O_{21} を処理するのに必要な時間は 2 である．次に，下位層の個体は遂行する処理の順序から形成される．図 4.8 (b) の場合，一番最初に行われる処理は仕事 1 の処理 1（O_{11}），次に行われる処理は仕事 2 の処理 1（O_{21}）となる．

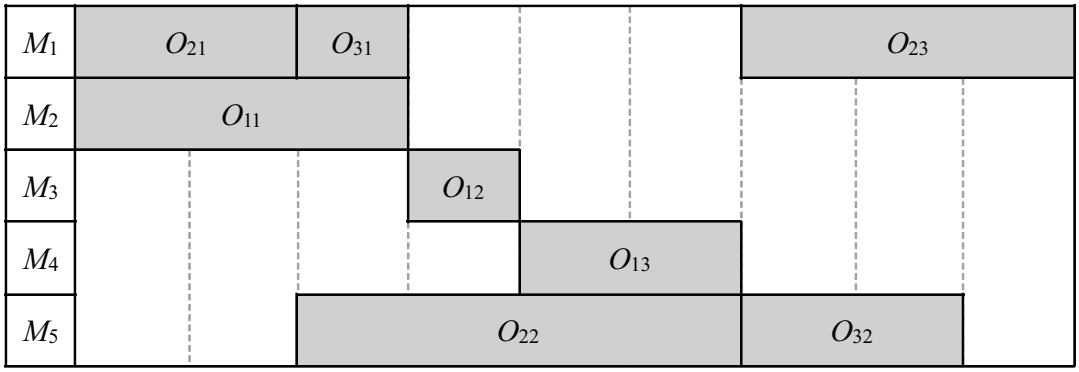


図 4.9: 図 4.8 の個体におけるガントチャート.

個体の評価

FJSP では，個体を評価するためのガントチャートを用いる．ガントチャートとは，横軸を時間，縦軸を機械とし，作業に必要な時間経過を各機械ごとに表したグラフである．図 4.9 に図 4.8 の個体を用いた時のガントチャートを示す．図中の M_1 から M_5 は使用した機械を意味し，色のついている部分は処理を行っていることを意味する．色のついていない部分は機械が稼働していないことを示す．O_{23} の処理が終了した時点の時間を総作業時間とし，この場合は 9 となる．ガントチャート作成の規則（FJSP における制約）は以下のようになる．

- ある機械が一度に複数の処理を行うことはできない.
- 前の処理が終了していなければ次の処理を行うことができない.
- すべての機械を用いる必要はないが，必ずすべての処理を行う.
- ある機械がある仕事の処理を行うために必要な時間は事前に与えられている.

これらの制約を満たすようにガントチャートを作成し，総作業時間が短い個体ほど高い評価を与える.

遺伝的操作

HmcDGA を FJSP に適用した場合にも，個体の表現に適した遺伝的操作を行う必要がある．表 4.3 に，FJSP のシミュレーションに用いた遺伝的操作を示す．交叉方法について，この問題も mVRP 同様，下位層の個体は特殊な表現をしているため，Partially Matched Crossover (PMX)[35] や Cycle Crossover (CX) [36]，Order Crossover (OX) [37] といった交叉方法が検討されているが，

表 4.3: FJSP に用いた遺伝的操作

	Upper level	Lower level
Selection	Roulette selection	
Crossover	Uniform	POX[60]
Mutation	Normal mutation	Translocation

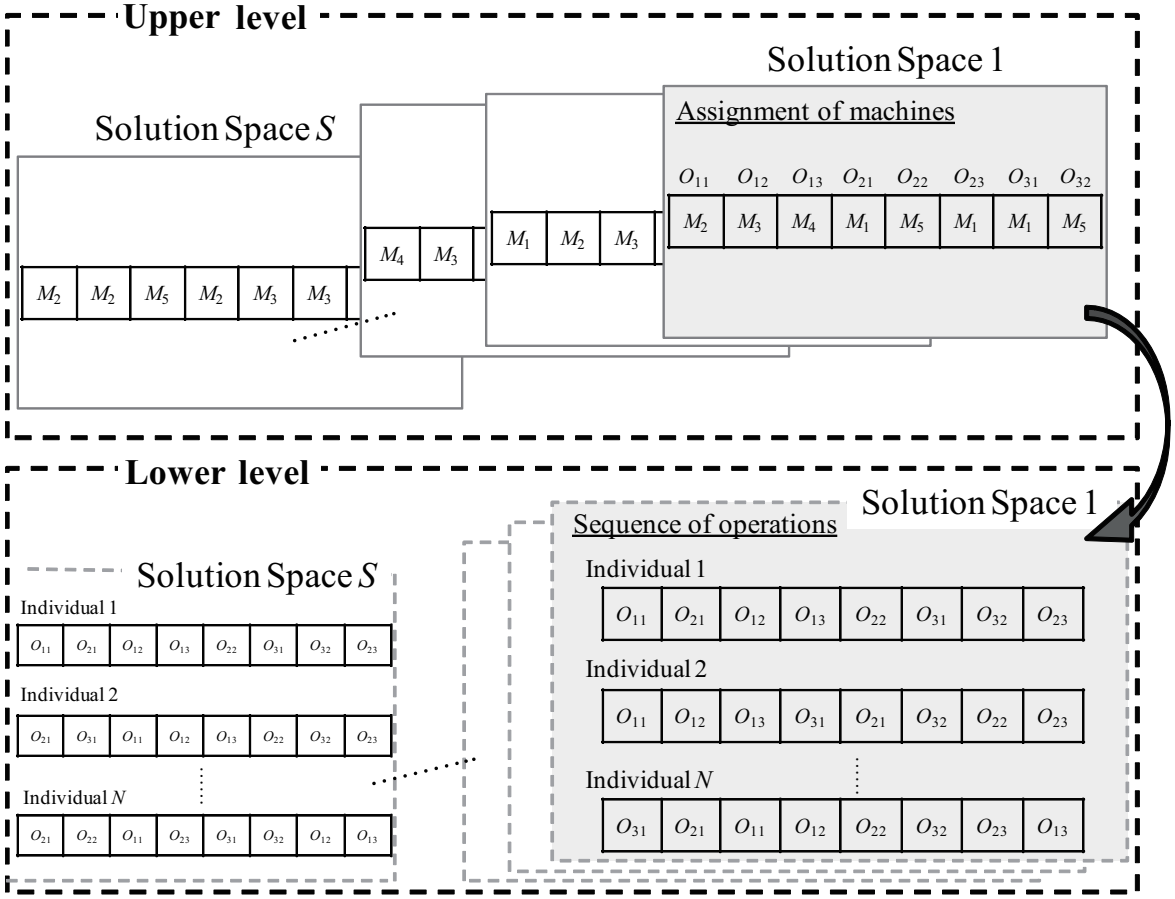


図 4.10: FJSP を階層化した場合のイメージ図.

本研究では FJSP において有効であると言われている Preserving order-based crossover (POX)[60] を採用する.

図 4.10 に FJSP を階層化した場合のイメージ図を示す. この問題において, 上位層では各処理を行う機械の組合せの最適化, 下位層ではそれらの順序を最適化する.

表 4.4: シミュレーションに用いた HmcDGA のパラメータ.

	Upper level	Lower level
Population size	100	Variable
Generation	30000	100
Crossover rate	0.8	
Mutation rate	0.3	0.1

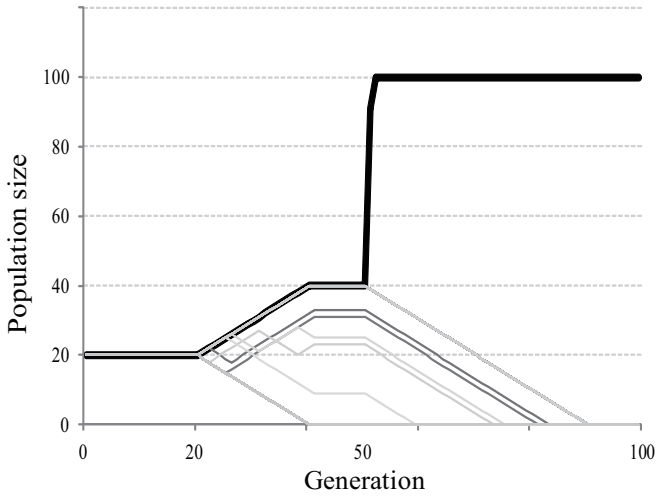
シミュレーション条件

本シミュレーションでは，ベンチマーク問題として Brandimarte を用いた．Brandimarte は仕事数や機械の数がそれぞれ異なる 10 種類の問題であり，FJSP のベンチマーク問題の中で，最も多くの文献で用いられている．表 4.4 に本シミュレーションで用いた HmcDGA のパラメータを示す．今回用いた規則では，上位層における，個体数は 100 とし，探索終了世代は 30000 世代とした．下位層における個体数は初期個体数を 20 とし，探索終了世代は 100 世代とした．競合は 20 世代から開始され，50 世代に至るまで，一度の競合につき，評価値が高い上位半分の探索空間に 1 つ個体を追加し，下位半分の探索空間からランダムに個体を 1 つ削除する．このとき削除される個体にエリートは選ばれない．50 世代以降は，1 度の競合につき，最も評価値の高い探索空間に個体を 1 つ追加し，それ以外の探索空間からはランダムに 1 つ削除する．つまり，50 世代以降は，探索空間は 1 つに絞られる．このとき，探索空間の最大個体数は 100 個体とした．10 種類の問題にすべて対して，同様の条件でシミュレーションを行った．

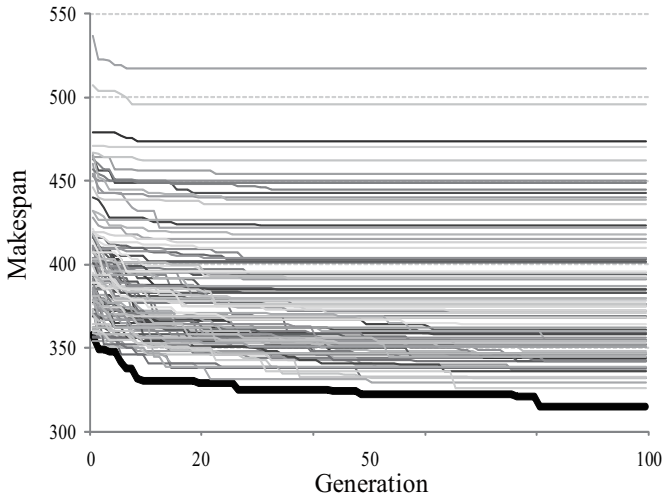
結果と考察

図 4.11(a) にシミュレーションで用いた競合の規則を示す．図 4.11(b) は，ある世代におけるすべての探索空間の個体数をグラフ化したものであり，太い実線は競合に勝利した探索空間の個体数，それ以外の点線は競合に敗れた探索空間の個体数を示している．
図 4.11(c) は，図 4.5(b) の世代における個体の評価値をグラフ化したものである．これらの図は，今回の競合において最も理想的な状況を示しており，世代によっては，最終的に一つの探索空間に絞ることができない場合もある．表 4.5 に他の文献の結果と本シミュレーションの結果を示し，図 4.12 に Mk04 の最適解のガントチャートを示す．表 4.5 において，1 列目は問題の種類，2 列目はこの問題の下界値である．各文献について，Best はその文献における手法のベストの値であり，RE は LB との相対誤差を示す．本シミュレーションにおいて，HmcDGA の試行は 3 回行い，表における値はそのベストである．一般的に，FJSP の結果の比較は LB との相対誤差によって

(a) FJSP において採用した競合の規則のイメージ.



(b) 個体数の増減の様子.



(c) (b) の世代におけるメイクスパンの変化.

図 4.11: HmcDGA における競合.

表 4.5: Brandimarte ベンチマーク問題における HmcDGA と既存手法の結果

		Chen		Pezzella		Ida		Zhang		Teekeng		HmcDGA	
	LB	Best	RE	Best	RE	Best	RE	Best	RE	Best	RE	Best	RE
Mk01	36	40	11.11	40	11.11	40	11.11	40	11.11	40	11.11	40	11.11
Mk02	24	29	20.83	26	8.33	26	8.33	26	8.33	27	12.50	26	8.33
Mk03	204	204	0.00	204	0.00	204	0.00	204	0.00	204	0.00	204	0.00
Mk04	48	63	31.25	60	25.00	60	25.00	60	25.00	64	33.33	60	25.00
Mk05	168	181	7.74	173	2.98	172	2.38	173	2.98	175	4.17	175	4.17
Mk06	33	60	81.82	63	90.91	58	75.76	58	75.76	65	96.97	60	81.82
Mk07	133	148	11.28	139	4.51	139	4.51	144	8.27	144	8.27	144	8.27
Mk08	523	523	0.00	523	0.00	523	0.00	523	0.00	523	0.00	523	0.00
Mk09	299	308	3.01	311	4.01	307	2.68	307	2.68	309	3.34	311	4.01
Mk10	165	212	28.48	212	28.48	197	19.39	198	20.00	234	41.82	217	31.52
Ave			19.55		17.53		14.92		15.41		21.15		17.42

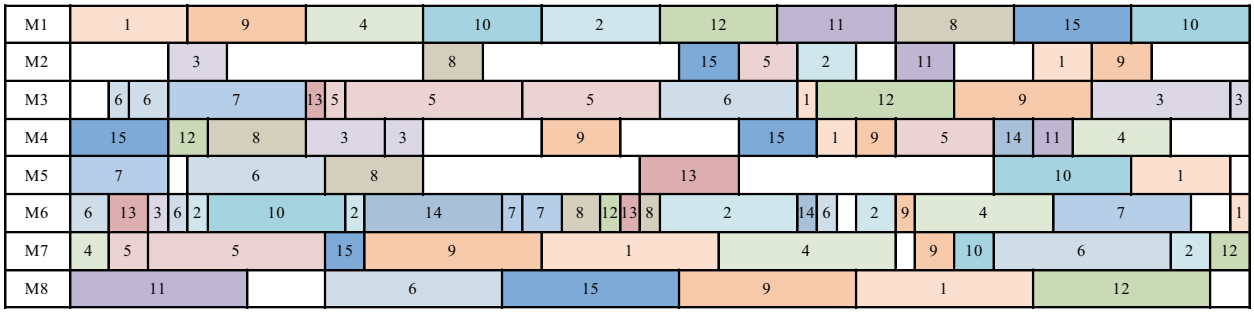


図 4.12: HmcDGA によって得られた Mk04 の解.

行われ，この値が小さいほど良い結果であるといえる．表 4.5 における文献の結果は，いずれも FJSP を解くために特化した手法であり，基盤となる手法として GA が用いられている．表 4.5 において，HmcDGA では，他の手法と比較してほぼ同程度の結果が得られているといえる．つまり，HmcDGA は，FJSP のような複雑な組み合わせ最適化問題に対して有効であるということが示せた．しかしながら，Mk10 の結果においては，他の手法と比べてやや総作業時間が大きい．Mk10 は用いる機械と行う仕事の数がこの中で最も大きい問題であり，上位層だけでなく，下位層における組み合わせ数も非常に多い．したがって，今回のシミュレーションにおける個体数や世代数では，下位層において適切に解が発見されていない可能性がある．しかしながら，FJSP では，評価の度にガントチャートを作成しなければならず，扱う問題によっては非常に多くの計算時間が必要になり，必ず解が見つかる保証もないので，安易に個体数や世代数を増やすことは危険である．つまり，扱う問題の性質をよく理解し，それに応じて競合の規則や初期個体数などを検証する必要がある．これはすべての最適化手法において共通の問題であるが，HmcDGA を FJSP に適

用する場合はより慎重な検証が必要である．

4.3 おわりに

本章では，提案手法である HmcDGA を実問題例として mVRP と Flexible Job-shop Scheduling 問題に適用した．HmcDGA を mVRP に適用した場合は，競合を行わない階層型 GA と比較して，少ない計算時間で精度よく，安定して解を発見できることを示した．今回用いた mVRP も上位層の組み合わせ次第では，下位層の探索空間の次元数に大きく差が出る．したがって，競合を行う際に上位層の評価が上がりきらず，多項式曲線フィッティングで示されたように，本来高い評価値を持つサブ母集団が競合に敗北することも懸念された．mVRP の場合は，巡回する顧客の数が増えるにつれて経路を求めるのが難しくなるため，片方の車両に顧客の割り当てが偏ってしまった場合は探索空間が広くなり，解の発見が困難になる．しかしながら，今回のシミュレーションで用いた評価関数は，各車両が同程度の移動距離になるほど評価が高くなるものに設定されており，一方の車両に偏るようなサブ母集団には比較的低い評価が与えられていたので，今回の競合のルールが対応できた．今回，本論文には記載しなかったが，車両の総走行距離が最小になるような評価関数を用いた場合でも，適切に解を求めることができた．これは，上位層において，できるだけ一つの車両で巡回しようとする個体ほど高い評価が与えられるため，上位層における探索空間の形状が非常に簡単な形状になっているからだと考えられる．したがって，競合のルールは上位層における探索空間の形状も考慮して検討する必要がある．次に，HmcDGA を FJSP に適用し，GA をベースとし，FJSP を解決するために特化した手法の結果と比較を行った．本シミュレーションは，HmcDGA の解探索能力を検証する目的で行った．シミュレーションでは，組み合わせ数の最も多い Mk10 において，他手法とやや総作業時間に差が生じたが，その他の問題においてはほぼ同程度の解を発見することができた．このことから，HmcDGA は FJSP のような複雑な最適化問題にも高い探索能力を発揮することが示された．しかしながら，FJSP では，非常に多くの計算時間を必要としたため，競合のタイミングや個体数について十分な検証を行えていない．よって，他の手法より劣っていたベンチマーク問題では，競合において本来高い評価値を与えられるはずのサブ母集団が選ばれにくくなっていたことも考えられる．つまり，適切に競合を行うことができれば，従来の手法よりも良い結果を示すことも可能かもしれない．FJSP の場合は，上位層における探索空間の形状が把握しにくいので，より慎重な検証が必要である．

第5章　結論

本論文では，複数の部分的最適化問題を階層的に構成した階層的最適化問題を，複数の探索空間を持つ最適化問題と定義し，探索空間どうしの競合という操作によって効率的に解を探索する mcDGA と HmcDGA の 2 つの手法をについて述べた．以下，各章で得られた成果について述べる．

第 2 章では，本研究で扱う問題である階層的最適化問題について述べ，上位層と下位層の部分問題に関係性のある場合，階層的最適化問題は複数の探索空間を持つ問題に置き換えることができると説明した．次に，提案手法の基盤である，遺伝的アルゴリズムについて述べた．まず，バイナリおよび実数値 GA の表現方法，遺伝的操作および特徴について述べた．次に，遺伝的アルゴリズムの拡張である解層型遺伝的アルゴリズムについて 2 種類の方法を紹介し，それぞれの特徴と，個体を階層化した場合のイメージ，本研究で採用した階層型 GA について示した．また，複数の探索空間を持つ問題を階層型 GA によって解決する場合のイメージを示し，上位層の GA の個体を探索空間，下位層の GA の個体を各探索空間の解であると定義した．最後に，提案手法を着想したきっかけとなった分散 GA について説明し，その特徴について述べた．

3 章では，mcDGA における探索空間どうしの競合の影響について検証するために，mcDGA を探索空間の数が少ない問題の例である多項式曲線フィッティング適用し，競合を行わない並列 GA と解発見までの世代数について比較した．また，競合のルールについて 2 つの方法を検証した．結果として，競合を開始する世代数を適切に設定すること，探索空間を一気にひとつに絞るよりは，まず半分にするなど，段階的に探索する空間を減らしていく方が有効であることを示した．正解が 5 次関数の場合においては，正解の探索空間と最適解を発見することができ，並列 GA と比較して短い世代数でより良い解を発見できることを示し，競合の有効性について示した．しかしながら，7 次関数においては，今回用いた競合の規則では，適切にモデルを選択できなかった．原因として，データセットが適切でない場合も考えられるが，7 次関数を適切に探索するための計算リソースが十分でなかったことが挙げられる．つまり，今回用いた競合の規則で mcDGA が有効性を示すことができるのは，各探索空間の大きさや複雑さにそれほど差がない場合に限られ，各探索空間の大きさや複雑さが大きく異なる問題に対しては，個体数や世代数の計算リソースに重みを付ける，競合のタイミングを遅らせるなど工夫する必要があり，さらなる検証が必要である．

4 章では，探索空間の数が非常に多い問題の例として，複数車両配送計画問題（ Multiple Vehicle Routing Problem : mVRP ）と Flexible Job-shop Scheduling 問題（Flexible Job-shop Scheduling Problem : FJSP）を挙げ，mcDGA を階層化した HmcDGA を適用した．両問題において，競合の規則は 3 章で述べた方法と同様の方法を採用した．HmcDGA は，探索空間どうしの競合によって少ない計算時間で解を発見でき，複雑な問題を階層化することで，簡単な問題に置き換えることができるという 2 つの特長を持つと期待できる．まず，HmcDGA が従来の階層型 GA と比較して少ない計算時間で安定して解を発見できることを示すために，HmcDGA を mVRP に適用した．mVRP では，各車両が巡回する顧客の割り当てを探索空間とし，それらは上位層によって最適化される．割り当てられた顧客の配置における最短経路を下位層によって決定する．下位層において得られた最大評価値が各探索空間の評価値となる．結果として，HmcDGA では従来の階層型 GA と比較して，少ない計算時間で精度よく解を発見できることを示した．次に，HmcDGA の解探索能力と汎用性について考察するために，HmcDGA を FJSP に適用し，ベンチマーク問題を用いて FJSP に特化した手法と結果を比較した．FJSP では，使用する機械の組合せによって探索空間が決まり，mVRP と比較して探索空間の組み合わせ数が多い問題である．結果として，HmcDGA は他の手法と同等の結果を得ることができた．つまり，HmcDGA は FJSP 問題に対して有効であり，問題に特化した手法と同程度の性能があることを示し，汎用性についても示唆された．しかしながら，ベンチマーク問題において，最も組み合わせ数の多い問題では，他の手法と比較してやや劣った結果となった．これは，下位層において適切に解が発見されていないことを示唆しており，FJSP のような組み合わせ数が多い問題では，初期個体数および競合において調節する個体数を慎重に設定する必要がある．各問題において，必要な個体数や世代数を調査し，有効なアプリケーションとして利用するためには，問題に対して有効な値の範囲を示す必要がある．

以上のように，本論文では探索空間が複数ある問題に対して有効な mcDGA とその拡張である HmcDGA を提案した．問題例として，mcDGA を多項式曲線フィッティングに，HmcDGA を mVRP と FJSP に適用することで提案手法の有効性を示した．今後の課題として，扱う問題のスケールに応じた競合の規則の明確な指標（有効であるかそうでないか）の決定を挙げる．

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りました九州工業大学大学院生命体工学研究科人間知能システム工学専攻 堀尾 恵一 准教授に心から感謝致します．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大学院生命体工学研究科人間知能システム工学専攻，古川 徹生 教授，吉田 香 准教授に謝意を申し上げます．また，これまでの学生生活において，多くのご支援を頂き，いつも一緒に遊んでくれた九州工業大学大学院生命体工学研究科人間知能システム工学専攻堀尾研究室，山川研究室をはじめとした多くの研究室の方々に厚く御礼申し上げます．最後に，29 年間いつも私のことを応援し，支えてくれた両親や兄妹に心から感謝いたします．

参考文献

[1] J.H. Holland. Outline for a Logical Theory of Adaptive Systems. *Journal of the Association for Computing*, Vol. 9, pp. 297–314, 1962.

[2] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE Transactions on*, Vol. 9, No. 3, pp. 303–317, 2005.

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

[4] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992.

[5] Thomas Back, David B Fogel, and Zbigniew Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.

[6] Edwin D. de Jong, Richard a. Watson, and Dirk Thierens. A generator for hierarchical problems. *Proceedings of the 2005 workshops on Genetic and evolutionary computation - GECCO '05*, p. 321, 2005.

[7] K.S. Tang, C.Y. Chan, K.F. Man, and S Kwong. Genetic structure for NN topology and weights optimization. In *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*, 1995.

[8] Kim-Fung Man, Kit Sang TANG, and Sam Kwong. *Genetic algorithms: Concepts and designs*. Springer Science & Business Media, 2012.

[9] K.S. Tang, K.F. Man, and R.S.H Istepanian. Teleoperation controller design using hierarchical genetic algorithm. In *IEEE International Conference on Industrial Technology*, 2000.

[10] Mehmet Gulsen and Alice E Smith. A hierarchical genetic algorithm for system identification and curve fitting with a supercomputer implementation. In *Evolutionary Algorithms*, pp. 111–137. Springer, 1999.

[11] James Kennedy. Particle Swarm Optimization. *Encyclopedia of Machine Learning*, pp. 760–766, 2010.

[12] Marco Dorigo and Mauro Birattari. Ant colony optimization. In *Encyclopedia of machine learning*, pp. 36–39. Springer, 2010.

[13] F Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, Vol. 13, pp. 533–549, 1986.

[14] Reiko Tanese. *Distributed Genetic Algorithm for Function Optimization.* PhD thesis, Department of Electrical Engineering and Computer Science. University of Michigan, 1989.

[15] Reiko Tanese. Parallel Genetic Algorithm for a Hypercube. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*, pp. 177–183, 1987.

[16] John Hubert Dantzig, George Bernard; Ramser. The Truck Dispatching Problem. *Management Science*, Vol. 1, No. 6, pp. 80–91, 1959.

[17] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, Vol. 59, pp. 345–358, 1992.

[18] F Pezzella, G Morganti, and G Ciaschetti. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, Vol. 35, No. 10, pp. 3202–3212, 2008.

[19] Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, Vol. 41, pp. 157–183, 1993.

[20] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[21] K. A. DeJong. *An Analysis of the Behavior of a class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbour, Department of Computer and Communication Sciences, 1975.

[22] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, Vol. 3, No. 2, pp. 95–99, 1988.

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

[24] LJ Eshelman. Real-Coded Genetic Algorithms and Interval-Schemata. *Foundations of Genetic Algorithms*, Vol. 2, pp. 187–202, 1993.

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

[26] Paul C Chu and John E Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of heuristics*, Vol. 4, No. 1, pp. 63–86, 1998.

[27] Robert S Garfinkel and George L Nemhauser. *Integer programming*, Vol. 4. Wiley New York, 1972.

[28] Jun Wang, L Huang, CG Zhou, W Pang, et al. Traveling salesman problem. In *Conference on Machine Learning and Cybernejcs*, Vol. 3, pp. 1583–1585, 2003.

[29] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, Vol. 21, No. 2, pp. 498–516, 1973.

[30] Cezary Z Janikow and Zbigniew Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In *ICGA*, pp. 31–36, 1991.

[31] 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.

[32] Isao Ono, Shigenobu Kobayashi, and Koichi Yoshida. Global and Multi- objective Optimization for Lens Design by Real-coded Genetic Algorithms. In *International Optical Design Conference*, pp. 110–121, 1998.

[33] Una-May O'Reilly Minkyu Kim, Varun Aggarwal and Muriel Medard. A Doubly Distributed Genetic Algorithm for Network Coding. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007.

[34] M. A. Rodriguez A. Peregrin. Efficient Distributed Genetic Algorithm for Rule Extraction. *Eighth International Conference on Hybrid Intelligent Systems*, pp. 531–536, 2008.

[35] D.E. Goldberg. Alleles, loci, and the traveling salesman problem. In D.E.Goldberg, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Application, Lawrence Erolaum, New Jersy*, pp. 154–159, 1985.

[36] J.R.C Holland I.M.Oliver, D.J.Smith. tudy of permutation crossover operators on the traveling salesman problem. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*, 1987.

[37] L. Davis. Applying adaptive Algorithms to Epistatic Domains. In *IJCAI*, pp. 162–164, 1985.

[38] 小野功, 佐藤浩, 小林重信. 単峰性正規分布交叉 UNDX を用いた実数値 GA による関数最適化. 人工知能学会誌, Vol. 14, No. 6, pp. 1146–1155, 1999.

[39] 樋口隆英, 筒井茂義, 山村雅幸. 実数値 GA におけるシンプレクス交叉の提案. 人工知能学会論文誌, Vol. 16, pp. 144–155, 2001.

[40] Ranjan Kumar, Kazuhiro Izui, Masataka Yoshimura, and Shinji Nishiwaki. Multi-objective hierarchical genetic algorithms for multilevel redundancy allocation optimization. *Reliability Engineering & System Safety*, Vol. 94, No. 4, pp. 891 – 904, 2009.

[41] Marco Dorigo, Mauro Birattari, Christian Blum, Maurice Clerc, Thomas Stützle, and Alan Winfield. *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings*, Vol. 5217. Springer, 2008.

[42] Tomoyuki Hiroyasu, Mitsunori Miki, and Masami Negami. Distributed genetic algorithms with randomized migration rate. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, 第1巻, pp. 689–694, 1999.

[43] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, Vol. 89, No. 9, pp. 1275–1296, 2001.

[44] Y. Fujimoto S. Tsutsui. The fGA : Forking Genetic Algorithm with Blocking and Shrinking Modes. In *Proceedings of 5th International Conference on Genetic Algorithms*, 1993.

[45] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Lecture notes in computer science*, Vol. 1917, pp. 849–858, 2000.

[46] Kalyanmoy Deb and Santosh Tiwari. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, Vol. 185, No. 3, pp. 1062–1087, 2008.

[47] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[48] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pp. 199–213. Springer, 1998.

[49] Günther Zäpfel and Michael Bögl. Multi-period vehicle routing and crew scheduling with outsourcing options. *International Journal of Production Economics*, Vol. 113, pp. 980–996, 2008.

[50] Giselher Pankratz. A Grouping Genetic Algorithm for the Pickup and Delivery Problem with Time Windows. *OR Spectrum*, Vol. 27, No. 1, pp. 21–41, 2005.

[51] He Miao Ding Chao, Cheng Ye. Two-Level Genetic Algorithm for Clustered Traveling Salesman Problem with Application in Large-Scale TSPs. *TSINGHUA SCIENCE AND TECHNOLOGY*, Vol. 12, No. 4, pp. 459–495, 2007.

[52] Timothy Potter and Terry Bossomaier. Solving Vehicle Routing Problems with Genetic Algorithms. In *Evolutionary Computation, 1995., IEEE International Conference on*, 1995.

[53] M. R. Garey, D. S. Johnson, and Ravi Sethi. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operation Research*, Vol. 1, pp. 117–129, 1976.

[54] BS Girish and N Jawahar. Scheduling job shop associated with multiple routings with genetic and ant colony heuristics. *International journal of production research*, Vol. 47, No. 14, pp. 3891–3917, 2009.

[55] BS Girish and Natarajan Jawahar. A particle swarm optimization algorithm for flexible job shop scheduling problem. In *Automation Science and Engineering, 2009. CASE 2009. IEEE International Conference on*, pp. 298–303, 2009.

[56] G.H. Zhang, L. Gao, and Y. Shi. An effective genetic algorithm for the flexible job-shop scheduling problem. *Flexible Services and Manufacturing Journal*, Vol. 23, pp. 64–85, 2011.

[57] Kenichi Ida and Kensaku Oka. Flexible Job-shop Scheduling Problem by Genetic Algorithm. *Electrical Engineering in Japan*, Vol. 177, No. 3, pp. 505–511, 2009.

[58] José Fernando Gonçalves, Jorge José de Magalhães Mendes, and Maurıcio GC Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, Vol. 167, No. 1, pp. 77–95, 2005.

[59] Wannaporn Teekeng and Arit Thammano. Modified genetic algorithm for flexible job-shop scheduling problems. *Procedia Computer Science*, Vol. 12, pp. 122–128, 2012.

[60] KM. Lee K. Lee, T. Yamakawa. A genetic algorithm for general machine scheduling problems. *International Journal of Knowledge-Based Electronic*, Vol. 2, pp. 60–66, 1998.