博士（工学）論文

ＧＡによる大域的最適化とその実用に向けた研究

Global Optimization using Genetic Algorithms for Practical Uses

平成15年３月

指導教官　　小林重信教授

学籍番号　　00D53042

氏　　名　　池田 心

目 次

第1章　はじめに

1.1　研究の背景

近年のコンピュータ環境の発達に伴い、領域のエキスパートに頼るしかなかった設計や制御などの困難な最適化問題においても、直接探索法を用いた自動化による作業の効率化が図られるようになってきている。分枝限定法（Branch And Bound; BAB）[Brucker 94] をはじめとする厳密解法が最適解を必ず発見できる代償に非常に長い時間を要するのに対し、現実的な時間で効率良く精度の高い解を求めることを目的とした焼きなまし法（Simulated Annealing; SA）[Kirkpatrick 83] やタブサーチ（Tabu Search; TS）[Glover 89] などの確率的最適化手法が注目されている。

遺伝的アルゴリズム（Genetic Algorithm; GA）[Holland 75] は、生物の進化過程に着想を得た確率的最適化手法の一つである。GA では解候補は一つの個体と解釈され、その質に応じて次世代に生き残るか淘汰されるかが決定される。GA は、関数型や勾配に関する情報を用いず目的関数値のみを用いる直接探索法であること、確率的最適化手法であることの他に、複数の解候補を保持する多点探索法であるという大きな特徴を持つ。進化の過程を経て洗練された複数の解候補の情報を利用することで、より良い解候補を生成することを目的とした交叉オペレータは、GA にとって最も本質的な操作である。また近年重要性を増してきている多目的最適化問題においては合理的解（例えば Pareto 最適解）が複数要素の集合をなすため、一試行で複数の解候補を得られる GA との親和性は高い [Fonseca 93]。これらのことから、GA は最適化のための枠組みとして非常に高い潜在能力を持つと考えられる。

GA 研究の初期段階では、生物の進化モデルに沿うために、解候補をビットストリングで表現し、1 点交叉や 2 点交叉といった組換え操作を用い、親集団と子集団を無条件で入れ替える世代交代モデルを用いていた。この 10 年ほどの間に、最適化手法として GA を

見たときの潜在能力を十分発揮させるために、1）実数値表現 [Ono 97] など問題に適した解表現、2）問題に特化した交叉の設計 [Nagata 97]、3）質の高い解を残しかつ多様性を維持する世代交代モデルの開発 [Takahashi 99]、など様々な工夫が加えられてきた。その結果 GA は多くのベンチマーク問題で他の手法を上回る性能を示し、最も有望な最適化手法の一つとして期待されている。

しかしながら、実用の世界ではそれぞれの問題に最も適した GA の設計を行うことは非常に困難であり、必ずしもファーストチョイスとされていない、あるいは他手法を凌駕する性能を挙げていないというのが現状である。十分な性能が引き出せない原因は主に、実問題には研究室レベルで想定しているより、さらに複雑な構造や制約条件、さらに多数の変数や目的関数が存在していることであると考えられる。また、GA の専門家でない一般のユーザにとっては高度な GA を設計するための指針が不足していることも一因となっており、ツールとして使いやすい GA の実用化が望まれている。

1.2 研究の動機と目的

GA は枠組みとして高い潜在能力を持ち、また適切な設計を行うことで実際に多くの問題で非常に優秀な探索性能を示している。しかし学術的研究の多くは対象を狭いクラスに限定した議論を行い、実問題の持つ複雑な構造や制約、多数の評価規範まで考慮に入れた GA の設計論はまだまだ発展段階にあると考えられる。また GA は多くの構成要素を持ち、オペレータやパラメータを適切に設計するのは SA 等の他の最適化手法に比べても困難であるため、実用に向けては一般ユーザのための簡明な設計指針が必要である。

GA を実問題に適用することを考慮に入れるならば、a）今まで想定していたよりも扱うクラスを広範化すること、b）最適化の本来の目的に即してユーザが真に得たい解と GA が提示する解の乖離を防ぐこと、c）各問題に対する GA の設計に要する開発コストを削減するため一般的で簡明な設計指針を示すこと、が求められる。これらが達成されれば、最適化ツールとしての GA がより広く実問題に適用され、設計や制御などの分野を中心に多大な恩恵がもたらされると考えられる。

本研究の目的は、問題の定式化から GA を用いて最適化するまでの流れの中で要点とな

る 4 つの点を挙げ、それぞれ実用化のために求められる課題を解決するための方法論を提示することにある。具体的には、1）複雑な制約を持つ最適化問題に適切に対処できる戦略の提示、2）探索空間中に複数の有力な局所解が存在するときに真の最適解を見つけるための世代交代モデルの設計、3）ユーザの許容するトレードオフ比を考慮して探索を進める多目的最適化戦略の提示、4）近傍と距離から容易に構成される汎用かつ強力な交叉の設計、およびこれら 4 点に関する GA の設計戦略を明確にすることを目標とする。

1.3 論文の構成

本論文は「GA による大域的最適化とその実用に向けた研究」と題し、7 章より構成されている。

第 2 章「GA を用いた大域的最適化」では、まず与えられた問題を定式化し GA を用いて最適化するまでの流れを示し、鍵となる 4 つのテーマを取り上げ、これら全体を俯瞰した“ 大域的な ”最適化の必要性を論じる。

第 3 章から第 6 章までは各論である。第 3 章「近傍と距離から容易に構成できる汎用的な交叉の設計」では、開発に大きなコストを要する発見的交叉および性能の不十分な従来の汎用的交叉に代わり、両親の良い要素を“ いいとこどり ”する、設計の容易な交叉を提案する。

第 4 章「大域的多峰性を考慮した世代交代モデルの設計」では、複数の有力な局所解が遠く離れて存在するという従来想定していなかった景観構造に着目し、従来の GA が探索に失敗する原因を解析した上で、このような問題でも真の最適解を発見できる世代交代モデルを提案する。

第 5 章「α-domination 戦略に基づく多目的最適化」では、Pareto 最適解をまんべんなく求めるという従来の多目的最適化の目標を見直し、ユーザが真に必要とする有益な解だけを高速かつ安定に提示する最適化戦略を提案する。

第 6 章「独立制約充足戦略に基づく制約条件付最適化」では、複雑な制約条件を持った問題に対して頻繁に用いられるペナルティ加算法が制約だけを満たした質の悪い解を安易に発見するに留まることを指摘し、解の質を悪化させずに制約を満たすための戦略を提案

する。

これらの4つの章は概ね、背景と目的の提示、従来法とその問題点の考察、接近法の提案、比較実験、まとめという形式を持つ。

第7章では本研究の成果と意義、今後の展望をまとめる。

第2章　GAを用いた大域的最適化

2.1　問題解決過程における最適化

本節では、問題が定式化されGAを用いて実際に最適化されるまでのおおまかな流れを記し、性能を左右する鍵として本論文で扱う4つのテーマがどこにあたるのかを述べる。問題解決の流れは必ずしもトップダウンの不可逆的なものではないが、概ね以下のような手順を踏む。

1. ＜目的の設定＞「高速な飛行機を設計したい」「二足歩行ロボットを制御したい」「強いコンピュータ囲碁プレイヤを作りたい」といった自然言語的な問題を設定する。

2. ＜問題の定式化＞ 最適化に帰着する前に、漠然とした問題を整理・限定・具体化する必要がある。このフェイズには以下の手順が含まれる。

   - 何を最適化対象とするのか、すなわち**解表現と探索空間**を定める。目標が同じであっても解表現には任意性があり、その選択は性能だけでなく以後の最適化手順にも大きな影響を及ぼす。例えば関節のトルクを出力とするロボット制御を行う場合、「時刻を入力とした時系列データ」を解とするのと「センサー入力を用いた関数器のパラメータ」を解とするのでは同じ問題を解いているとは言えないほどの差がある。
   - どうなると嬉しいのか、すなわち**目的関数**あるいは解に対する評価値を定める。問題によっては評価値は必ずしも静的には定まらず [Sano 2002]、また複数の目的関数が存在することも多い [Knowles 99]。
   - 何が許され何が禁止されているのか、すなわち**制約条件**を定める。探索空間中の制約を満たす解を特に実行可能（feasible）解と呼ぶ。
   - 問題に対する領域知識（domain knowledge）や見本例（exemplar）など、補助的

に利用できるものを整理する。GA について言えば、交叉・突然変異の設計の際に領域知識を利用することが有効な場合が多く、また見本例は初期集団の生成に利用することで序盤の探索を加速させるなどの効果が期待できる。

- シミュレータの有無、計算機資源、許容コストといった外部環境を整理する。GA を含む全ての最適化アルゴリズムにはそれぞれ特徴があり、その中から一つを選択する際、またアルゴリズムの各構成要素を設計する際に外部環境との親和性は非常に重要になる。

3. ＜最適化問題＞ 問題が以上のステップを踏んで定式化・客観化されたとき、最適化問題に帰着されたという。最適化問題は少なくとも以下の形式・要素を持つ。

   - 解の探索空間 X. $X \subset \mathbb{R}^{n\in}$ のとき特に実数値最適化問題と呼ぶ。
   - 実行可能解の集合 $X_{Feasible} \subset X$.
   - 解 $x \in X$ に対する目的関数 $f_1(x), \ldots, f_m(x)$; $m \in \mathbb{N}$. $m = 1$ のとき単目的最適化問題、$m \geqq 2$ のとき多目的最適化問題と呼ぶ。

4. ＜最適化の戦略決定＞ 3. のように定式化された問題には、まだこのままでは最適化アルゴリズムを適用することはできず、以下の 2 つの重要な点について戦略を決定しなければならない。

   - **制約条件の扱い**：制約を全て満たした実行可能解の集合 $X_{Feasible}$ に探索を限れば、原理的には任意の最適化アルゴリズムを適用することができる。しかし通常、特に実問題においては制約をただちに満たすことは非常に困難であり [安達 2001]、悪くすると一つも制約充足解を見つけることなく探索が終了してしまう可能性すらある。そこで制約違反の度合いに応じたペナルティを評価値に付与し、徐々に違反を減らしていくといくなどの工夫がなされている [Francisco 2000]。
   - **複数の目的関数の扱い** ：単目的最適化においては、全ての解は互いに優劣を定めることができ、唯一の最適評価値が存在する。しかし一般には実問題は複数の評価基準を持つことが多く、このような多目的最適化では二つの解の間には明確な優劣が存在しない場合がある。従って、求める解は単一とは限らず、Pareto 最

適性など何らかの合理性を満たす複数の解を提示する必要がある。

5. ＜アルゴリズムの選択＞ 問題の特性や環境に応じて様々な最適化アルゴリズムの中から一つを選択する。本研究では、連続関数最適化および組み合わせ最適化両方に用いることができ、また多峰性を考慮した大域的探索能力を持ち、さらに多目的最適化とも親和性のある GA に対象を絞り議論を行う。

6. ＜アルゴリズムの設計＞ GA を含む全ての最適化アルゴリズムはいくつかのオペレータとパラメータを必要とし、各々について盛んに研究が行われている。本研究では、多点探索法であることが GA の最も特徴的な点であると考え、それを踏まえて特に以下の 2 点に着目する。

   - **交叉の設計**：交叉こそは、GA に固有かつ最も性能に影響を与えるオペレータである。しかし、良い交叉を独自に設計するには対象問題および GA 双方への深い理解が必要であり、一方、一様交叉などの汎用的な交叉では十分な性能が引き出せないという問題がある。
   - **世代交代モデルの設計**：GA は多点探索法であるために、SA や TS といった他の確率的探索手法に比べ関数の多峰性に強いと言われる [小野 97]。この性質を十分に生かすためには、集団の多様さを維持するための世代交代モデルの適切な設計が必要である。

7. ＜アルゴリズムの適用＞ 実際にアルゴリズムを適用し、得られた解を本来の目的に照らして評価し、必要に応じて前のフェイズに戻るフィードバック作業を行う。

2.2 広い意味での大域的最適化

本論文では、(1.) 目的が設定されてから (3.) 最適化問題に定式化するまでの段階、さらには (7.) フィードバック作業についてはシステム工学の方法論に委ねることとし、(4.) 最適化の戦略決定および (6.)GA を用いた場合の設計論に議論の対象を絞る。すなわち、「制約条件をどう扱うか」「複数の目的関数をどう扱うか」「世代交代モデルはどう設計するか」「交叉はどう設計するか」の 4 つのテーマに着目し、最適化の流れの中で各々が果た

す役割に留意しつつ解決すべき点を明らかにする。

これらのテーマがGAを用いた最適化にとって大切であるということを説いたのは本論文が初めてではなく、むしろ頻繁に議論されてきた話題でもある。しかしながら、各論を問題解決過程から切り離して議論しようとすると、A）狭い問題クラスに対象を限定したり、B）問題を解くという本来の目的とは離れた、便宜的・形式的な目標を定める、といった陥穽におちいる場合が多い。このために、a）ある問題クラスについては議論から得られた知見が有効に働くが、実問題ではしばしばそのクラスから外れた問題に遭遇するために十分な性能が発揮できない、b1）ユーザが真に得たい解とアルゴリズムが提供する“ 良好な ”解が乖離する、b2）形式的な目標にとらわれることでユーザにとって真に有益な手法が盲点となる、といった困難が生じる。

本研究では、最適化の本来の目的を達成するために、A）従来の研究が想定してこなかったクラスにまで対象領域を拡げ（これを本論文では**広範化**と呼ぶ）、B）従来の研究が定めてきた形式的な目標にとらわれずに、「何が求められているのか」を考慮して目標を**柔軟に**見直す。このように、全体の流れを俯瞰し考慮した最適化を本論文では**広い意味での大域的最適化**と呼ぶことにする。一般に用いられる狭い意味での大域的最適化とは局所的最適化の上位語として定義され、局所解にとらわれずに探索空間全体から真の最適解を発見できる最適化のことを指す。本論文ではこれをさらに拡張した上位語として定義し直し、限定的なクラスや目標にとらわれずにユーザにとって有益な解を発見できる広範で柔軟な最適化のことを指すものとする。

2.3 実用化に向けてのアプローチ

本節では、広い意味での大域的最適化を達成し実用化するために、扱う4つのテーマについて何を問題意識とし、それに対してどのようなアプローチをとるのかを論じる。

- **交叉の設計**

  巡回セールスマン問題（Traveling Salesman Problem; TSP）における枝組み立て交叉（Edge Assembly Crossover; EAX）は問題に特化した優秀な交叉である。しかし問題に特化したこのような交叉の設計には大きな開発コストが必要となるため、問

題への依存度が低い汎用的な交叉の設計が必須である。しかしながら、一様交叉などの従来の汎用的交叉は親の要素を混ぜ合わせるという形式的な目標にとらわれており、親の良い所だけを子に引き継ぎたいという本来の目的を達成できるものは少なかった。本研究ではこの目標を**柔軟化**し、親の“ いいとこどり ”を行うオペレータを交叉として提案する。

- **世代交代モデルの設計**

  多点探索法である GA は、SA や TS 等に比べ多峰性関数における局所解にとらわれにくいとされる。大谷構造 [Boese 94,95] とは、景観構造が多くの局所解を持っていても全体としては一つの大きな谷を成すというものであり、このクラスの下では GA が効率的に働くことが知られている。しかしながら、困難な問題の多くには必ずしもこのような GA にとって都合の良い構造が存在しているとは限らず、複数の有力な局所解が離れて存在していることが知られている。本研究では GA の適用可能クラスを**広範化**するために、このような場合でも大域的最適解を得ることを可能にする世代交代モデルを提案する。

- **複数の目的関数の扱い**

  多目的最適化では、何らかの合理性を持った複数の解を提示する必要がある。この分野の殆どの研究では、Pareto 最適解すなわち他のどの解にも dominate されない解をまんべんなく求めることを目標としている。しかしながら、Pareto 最適解の中には一つの目的関数については優れているが他の目的関数では非常に悪い、ユーザにとって全く必要のない解も含まれる。本研究では、多目的最適化は Pareto 最適解を求めなければいけないという目標を**柔軟化**し、ユーザにとって選択しうる有益な解だけを提示する最適化戦略を提案する。

- **制約条件の扱い**

  制約違反解を直ちに修正するのが困難な問題に対しては、違反の度合いに応じたペナルティを本来の評価値に線形和する手法（ペナルティ加算法; Additive Penalty Method; APM）[Francisco 2000] が頻繁に採用される。しかしながら、ペナルティ加

算法では制約を充足することと本来の評価値を最適化することが対立的となる場合が多く、このような場合制約は充足されるが質は劣った解を安易に発見するにとどまる可能性が高い。本研究では GA の適用可能クラスを**広範化**するために、このような困難な問題でも解の質を犠牲にすることなく制約を充足するための戦略を提案する。

GA が有用なツールとして広く活用されるためには、性能が優秀であり、広いクラスを扱え、ユーザにとって真に必要な解を得られるということだけでなく、設計と実装が容易であるということが求められる。GA による最適化を全自動化する、パラメータを自動調整するといった試みもなされているが、実際にはこれらは性能を犠牲にすることが多いのが現状である。

本研究では、それぞれのテーマにおいて大域的な最適化を達成するための簡潔な方法論を提示し、広いクラスを扱える優秀かつ合理的な GA の設計を可能にする。さらに、実際に問題に臨むときの設計および実装を容易にするために、各テーマの周辺についてどのような場合にどのようにすべきか具体的な指針を提示する。

第3章　近傍と距離から容易に構成できる汎用的な交叉の設計

本章では、性能は良いが設計に大きな開発コストを要する発見的交叉、一様交叉などの性能の不十分な従来の汎用交叉にかわり、近傍と距離から容易に構成できる高性能な交叉の提案を目標とする。このために本論文では、交叉は親の要素を混ぜ合わせるものであるという形式的な目標を柔軟化し、交叉の本来の目的である親の“ いいとこどり ”を行う操作が必要との立場をとる。

3.1　はじめに

GA は数ある確率的最適化手法のうちでも最も期待されているもののひとつであり、巡回セールスマン問題（Traveling Salesman Problem; TSP）における永田らの接近 [Nagata 97] をはじめとして多くの問題への適用がなされている。SA や TS と比較した場合の GA の大きな特徴の一つは、多点探索法であり、解同士の情報交換を探索に有効に利用している点である。子個体と呼ばれる次世代の解候補は親と呼ばれる現世代の解群から交叉オペレータによって生成される。このとき複数（通常 2 つ）の親の良いところを受け継いだ子孫を生成することが交叉オペレータの目的である。

しかしながら、GA を与えられた問題に適用するにあたっては、その問題にとって十分適切な交叉を設計することはしばしば非常に困難である。特に離散変数を含む最適化問題、組み合わせ最適化問題においては交叉を設計することが GA にとっての最も大きな課題となっている。例えば TSP は単純な定義と軽い制約条件を持つベンチマーク問題であるが、制約を満たしつつ親の良い点を十分に受け継ぐことができる交叉が開発されるまでには、十年以上の時間と何十もの交叉を積み重ねる必要があった [Reinelt 94]。

多くの場合、ある問題に対し適切な交叉を設計するためには、問題そのものへの十分な

理解と、GA というアルゴリズムおよびその中で交叉が果たす役割についての十分な理解双方が必要となる。そのため、GA に必ずしも精通していない一般ユーザにとっては、一様交叉等の汎用的な交叉をファーストチョイスとしていることが多いのが現状である。しかしこれら汎用的な交叉を用いても、きつい制約条件を持つ問題では実行可能解を生成するのが困難であったり、必ずしも親の良いところを受け継ぐ子個体を生成することができないために、GA の性能を十分に引き出すことができない場合が多い。

このような GA の課題に対して、問題ごとに近傍構造と距離尺度を導入することだけで交叉を容易に構成する試みが山田らによってなされている [Yamada 95],[Yamada 96,97]。一方永田らは親の要素を 1/2 の確率で受け継ぐ枝組立て交叉 EAX を、選択的に良い要素のみを受け継ぐよう変更することで性能を改善させている [Nagata 2000]。この 2 つのアイデアを用いることにより、本章では近傍と距離から容易に構成できる高性能な交叉の設計を可能にすることを目標とする。

3.2 従来法と問題点

本節では、従来の交叉設計法を概観し、その長所と短所を明らかにする。

3.2.1 発見的な交叉

離散変数最適化・組み合わせ最適化に適用され成功している EAX, JOX [Ono 96] などの交叉の殆どは、問題の性質と GA における交叉の役割を十分に考慮して設計された発見的交叉である。これらの交叉は問題ごとに個別に設計を行わねばならないために、問題そのものと GA への十分な理解を必要とし、ベンチマーク問題レベルであっても開発に大きなコストを要する。一般のユーザが現実の問題で十分な性能を持った発見的交叉を短期間に設計することは不可能に近く、GA が最適化アルゴリズムとしてファーストチョイスとされない一因となっている。

3.2.2 汎用的な交叉

発見的交叉に対し、問題への依存度が低く一般的な枠組みを持ち適用範囲の広い交叉を汎用的交叉と呼ぶ。最適化すべき変数が連続の実数値である実数値最適化問題の分野では、全ての問題が探索空間 $X \subset \mathbb{R}^{r\in}$ を持つため、SPX[樋口 2001], UNDX[Ono 97] などの実数値空間上での汎用交叉が定義でき、優秀な性能をあげると同時にパッケージ化も可能となっている。すなわち、実数値最適化問題においては古典的な GA の考え方から脱皮した交叉設計の方法論が完成しつつあるといえる。

一方離散変数最適化・組み合わせ最適化における汎用交叉としては、解をビットストリングなどで表現した上での一様交叉・多点交叉などが一般的である。しかしながらこのような操作は解の変数間の依存関係を崩してしまうことが多く、制約充足解が得られない、親の良い要素を十分に受け継ぐことができないなどの問題がある。これは、従来の汎用的交叉が親の要素を混ぜ合わせるという形式的な目標にとらわれており、両親の優れた要素を組み合わせるという本来の目標を達成できていないことを意味する。

3.2.3 MSX と MSXF

山田らは、片方の親から他方の親に向かって近傍を辿って解を生成していくことで、親の要素を混合させた子個体を生成することに成功した [Yamada 95]。以下に交叉 MSX（Multi Step Crossover）の一般的な手順を示す。

Multi Step Crossover

1. 両親を p_1, p_2 とするとき、初期探索点 $x_1 = p_1$ を定める。
2. ＜ステップ k＞ 探索点 x_k の近傍集合, $\mathcal{N}(x_k)$ を用意する。近傍の数が多すぎて列挙できないあるいは非効率的な場合は、ランダムに μ つの近傍のみを選択する。
3. 近傍の各要素 $y_i \in \mathcal{N}(x_k)$ を、他方の親 p_2 との距離 $d(y_i, p_2)$ に応じて昇順に並べ替える。
4. ランクに反比例した確率である近傍 $y_i \in \mathcal{N}(x_k)$ を選択する。
5. y_i を次ステップの探索点 x_{k+1} に定める。探索点が p_2 に到達するかステップ数が k_{max} に達したら 6. へ、そうでなければステップ数 k に 1 を加算して 2. へ戻る。
6. 全探索点 $x_1 ... x_{k_{max}}$ の中から最も適応度の高い解を採用し、なんらかのルールに基づいて集団に戻す。

MSX は、近傍構造 $\mathcal{N}(x) \subset X$ および距離尺度 $d(x,y) \in \mathbb{R}$ を定義することでただちに実装できる、汎用的交叉である。MSX のもう一つのメリットは、制約を充足した解を生成するのが容易であることである。これは、単純に親の要素を混合させる一様交叉のような操作に対し、制約を充足した解の周辺にこれを維持した解を近傍として生成することは比較的容易なためである。

一方で、MSX は解の質を鑑みることなく親 p_1, p_2 の間に解を生成するため、必ずしも効率的に良い解を得ることができないという欠点があった。MSXF（Multi Step Crossover Fusion）[Yamada 96,97] は解候補の受諾にメトロポリス規範を導入することで、一方の親 p_1 からできるだけ良い解を辿って他方の親 p_2 に向かう解を得ることを目的として提案された。以下の手順のうち 5 および 6 が MSXF に特有のものであり、温度パラメータ T を必要とする。

Multi Step Crossover Fusion

1. 両親を p_1, p_2 とするとき、初期探索点 $x_1 = p_1$ を定める。
2. ＜ステップ k＞ 探索点 x_k の近傍集合, $\mathcal{N}(x_k)$ を用意する。近傍の数が多すぎて列挙できないあるいは非効率的な場合は、ランダムに μ つの近傍のみを選択する。
3. 近傍の各要素 $y_i \in \mathcal{N}(x_k)$ を、他方の親 p_2 との距離 $d(y_i, p_2)$ に応じて昇順に並べ替える。
4. ランクに反比例した確率である近傍 $y_i \in \mathcal{N}(x_k)$ を選択する。
5. ＜メトロポリス規範＞ y_i が x_k より優れていた場合遷移を受諾、劣っていた場合は確率 $\exp(-\Delta V/T)$ で受諾する。ただし ΔV は y_i と x_k の評価値の差を表す。
6. y_i への遷移が棄却された場合、4. に戻る。
7. y_i を次ステップの探索点 x_{k+1} に定める。探索点が p_2 に到達するかステップ数が k_{max} に達したら 8. へ、そうでなければ 2. へ戻る。
8. 全探索点 $x_1 ... x_{k_{max}}$ の中から最も適応度の高い解を採用し、なんらかのルールに基づいて集団に戻す。オリジナルの論文では集団中の最悪個体の代わりに集団に戻す Steady State モデルを用いている。

もし手順 3 および 4 において p_2 への距離が考慮されずに近傍が選択されれば、MSXF は SA と同様の挙動を示す。一方手順 5 および 6 において解の質が考慮されずに近傍が受諾されれば、MSXF は MSX と同様に両親間に解を生成するにとどまる。すなわち、MSXF は SA と交叉の持つ好ましい性質両方を持つように設計されていると言える。図 3.1 に MSXF の動作の様子を簡単に示す。

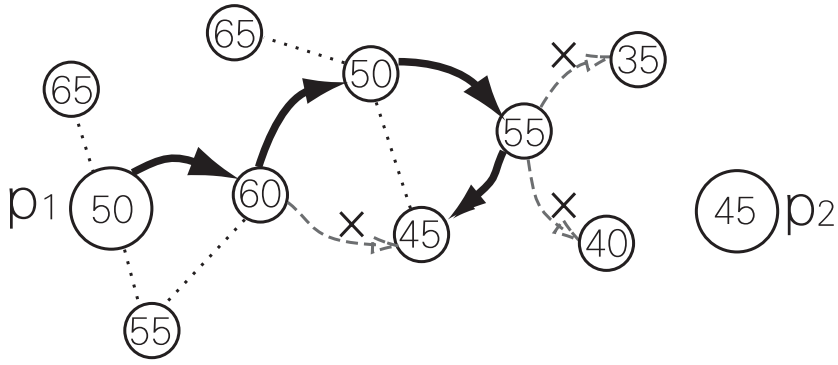


図 3.1: MSXF の概念図： 各円が一つの解を、円内の数値が解の適応度を表す。点線は解同士の近傍関係、太い矢印は受諾された遷移、破線の矢印は棄却された遷移を表す。この例では、適応度 55 を持った 4 番目の探索点から p_2 に近づく遷移は全て棄却され p_1 の方に戻る遷移が行われている。

温度パラメータ T が適切に設定されれば、MSXF は質の高い解を生成することが可能であり、例えばこれを用いた GA はジョブショップスケジューリング問題で従来法を凌駕する性能をあげている [Yamada 96,97]。しかしながら、もし T が高すぎた場合、ほぼ全ての遷移が受諾されるために、MSXF は MSX と同等の手順をとることになり、両親の間に質を考慮せずに単純に解を生成するに留まる。一方もし T が低すぎた場合、ほぼ全ての改悪する遷移が棄却されるために、景観構造に多峰性を持つ問題においては MSXF の一回の走査は容易に局所解に陥ってしまう。適切な温度 T を定めることは、特に探索の序盤と終盤で評価値のスケールに違いが生じる問題では非常に困難となる。

3.2.4 EAX-1AB

枝組立て交叉 EAX は、TSP における制約を満たしつつ親の要素を効率よく混合することができる発見的交叉である。EAX においては、両親のもつ枝は AB サイクルと呼ばれる複数の要素に分割される（詳細は 3.5 節に記述し、ここでは概念を述べるにとどめる）。オリジナルの EAX ではこの AB サイクルを各々1/2 の確率で選択することで、両親の特徴を半々に受け継いだ子を生成している。これは、TSP におけるいわゆるビルディングブロックを壊さずに子に受け継ぐ一様交叉と捉えることができる。

オリジナルの EAX には、生成された子個体が親の良い部分も悪い部分もランダムに受

け継ぎ、良い部分をより多く含んだ子個体が生成されにくいという一様交叉の持つ欠点がある。EAX-1AB においてはただ一つの AB サイクルがランダムに選択され、この結果生成される子個体は親 A に非常に近く一部分親 B の特徴を受け継いだものになる。このような子個体群は**親 B に近づく親 A の近傍**の一種であり、親 A 直系子個体、親 A を含めると親 A 直系家族と呼ぶ。これらの中から最も優れたものを選んで親 A と置き換えることで、親 A は親 B の良い要素１つだけを取り込むことができ、全体としても多様性を失わずに良い解を探索することができる。

EAX-1AB は TSP に特化した交叉法であり、親 A 直系子個体の中に親 A よりも優れるものが見つかる可能性は高い。しかしながら一般の問題においては、親の近傍に親よりも優れるものがすぐに見つかる可能性は低く、何度か近傍を辿って両親の間にある解を探索する必要がある。

3.3 接近法：いいとこどり交叉

優れた発見的交叉の開発には大きなコストが必要となり、従来の汎用的交叉は十分な性能を発揮できない場合が多い。MSXF は近傍構造と距離尺度から交叉を構成する一般的な手順を提供したが、温度パラメータ T の設定によっては期待される挙動を示さないことも多い。EAX-1AB は相手親の良い要素のみを取り込むことを目標とした交叉であるが、一般の問題にこのアイデアを取り入れるためには、局所解を抜け出せるよう何度か近傍を辿ることができるようにすべきである。

そこで、多段階を経て近傍を辿り相手親に近づくという MSXF のアイデアと、相手親の良い要素１つだけを取り込むという EAX-1AB のアイデアを用いた新しい交叉を提案する。これを、片方の親の良い要素だけを取り込んでいくという意味で、**いいとこどり交叉**と呼ぶ。

いいとこどり交叉

1. 両親を p_1, p_2 とするとき、初期探索点 $x_1 = p_1$ を定める。
2. ＜ステップ k ＞ 探索点 x_k の p_2 に近づく近傍集合, $\mathcal{N}_{\rightarrow p_2}(x_k)$ を用意する。要素数は最大 μ 個とし、全ての $y_i \in \mathcal{N}_{\rightarrow p_2}(x_k)$ は $d(y_i, p_2) < d(x_k, p_2)$ を満たすものとする。
3. 近傍集合 $\mathcal{N}_{\rightarrow p_2}(x_k)$ の中で最も優れた解を次探索点 x_{k+1} とする。探索点が p_2 に到達するかステップ数が k_{max} に達したら 4. へ、そうでなければ 2. へ戻る。
4. 全探索点 $x_1 ... x_{k_{max}}$ の中から最も適応度の高い解を採用し、親 p_1 の代わりに集団に戻す。

この交叉は近傍構造と距離尺度から構成され、温度パラメータ T を必要とせず、毎ステップ必ず相手親に近づくために走査中に MSXF のように局所解に留まることがない。いいとこどり交叉の概念図を図 3.2 に示す。いいとこどり交叉が一般的な枠組みを持ち、容易に実装できることは明らかであるが、温度パラメータ T を取り除いたことによって SA の持つ好ましい性質を失う可能性もある。次節以降では、いいとこどり交叉を 1max 問題と TSP に適用することにより、一様交叉と比較しても十分高い性能を持っていることを示す。

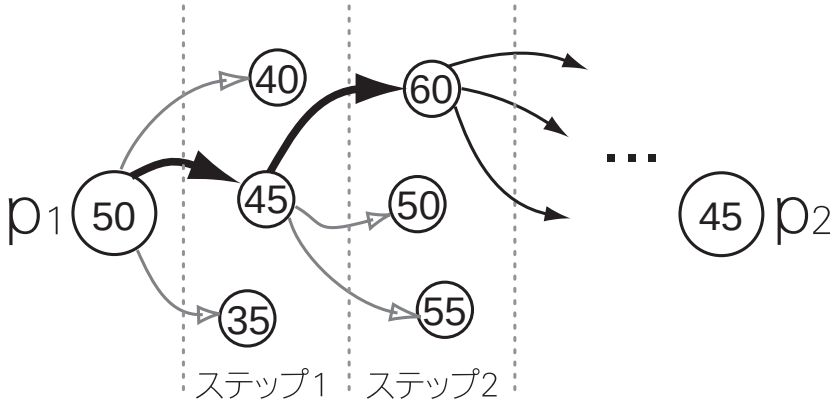


図 3.2: いいとこどり交叉の概念図： 各円が一つの解を、円内の数値が解の適応度を、太い矢印が親 p_2 に近づく近傍の中で最も優れた解への遷移を表す。この例では、親 p_1 の周辺には改善する個体を発見できなかったが、その中でも最も優れた解を次の探索点とすることで次ステップにつなげ、適応度 60 を持った解を発見している。

3.4 1max 問題での比較実験

1max 問題は、ビットストリングで解が表現される最も原始的な最適化問題の一つである。各変数（遺伝子座、0 か 1）同士には依存関係が存在せず、解の評価値は解の持つ 1 の数で定義される。本節では、この単純なベンチマーク関数を用いて、従来法と提案手法の比較解析を行うとともに、いいとこどり交叉の挙動をわかりやすく示すことを目的とする。

3.4.1 従来の汎用的交叉の性能評価

変数同士が依存関係を持たない 1max 問題においては、複数の変数をまとめて交叉する必要がないために、一点交叉、多点交叉（Multi-Point Crossover; MPX）よりも一様交叉（Uniform Crossover; UX）が高い性能を持つことが予想される。本節ではまずこれを確かめるための予備実験として、遺伝子長 L_{bit}=1000 とした 1max 問題に対して標準的な GA を用いて従来の交叉の性能を比較する。世代交代モデルとしては MGG モデル [Satoh 96] の修正版を用いる。この世代交代モデルを用いた場合、一世代あたり $C_{cross}{\cdot}N_{pop}/2$ 個の個体が生成され、評価される。

修正版 MGG-best2

1. 現集団から非復元抽出により $N_{pop}/2$ つの両親の組を生成する。ただし N_{pop} は集団サイズである。
2. 各組に対して、C_{cross} 個の子個体を交叉によって生成する。
3. 各組の親 2 個体および子個体あわせて $(C_{cross}+2)$ 個体のうちから最も優れた 2 つの個体を選択し、親 2 個体の代わりに集団に戻す。

実験 1：UX と MPX の比較

まず、UX と 20 点交叉、10 点交叉、5 点交叉の性能を比較する。実験設定は以下のとおりである。

- 集団サイズ $N_{pop}=20$, 生成子個体数 $C_{cross}=200$ とする。
- 初期個体はランダムに生成される。

- 交叉の性能の違いを明確にするために、突然変異オペレータは用いない。
- 20 世代にわたり集団内の最善評価値に改善がみられなかった場合に収束とみなして探索を打ち切る。
- GA は異なる乱数の種を用いて 50 試行行われ、発見された最善評価値の平均と標準偏差、および探索打ち切り時すなわち収束時の世代数で評価される。最善評価値の平均値が高く、収束が早いほど良い性能であると言える。

表 3.1 に実験の結果を示す。この結果、UX が最も質の高い解を得ることができ、また収束も早いことがわかる。これは、UX が全ての遺伝子座を分離して扱うのに対し、MPX は隣り合う遺伝子座をまとめて扱うために十分要素を混合できないためである。

表 3.1: （実験 1）1max 問題における UX と MPX の性能の比較

交叉	最善評価値の平均	標準偏差	収束世代数
一様交叉 UX	992.16	2.09	47.66
20 点交叉 MPX-20	924.90	7.65	55.14
10 点交叉 MPX-10	858.45	11.07	55.54
5 点交叉 MPX-5	803.85	10.62	56.09

実験 2：生成子個体数の与える影響

一つの両親のペアに対して生成する子個体の数が多いほど、その中により良い解が生成される確率は高まる。このとき、多数の子の中で最も優れるこのような解は、親の良い要素をより多く、悪い要素はより少なく受け継いだ解であるといえる。そのため、生成子個体数 C_{cross} を大きくするほどに、GA そのものが最終的に得る解の質も向上することが予想される。この実験では、交叉オペレータを UX に固定し、$C_{cross} = 10, 20, 50, 200$ としてその影響を比較する。他の設定は実験 1 と同様である。

表 3.2 に結果をまとめる。この結果は、総評価回数のオーダが異なるために、「C_{cross} はできるだけ大きくすべきである」ということを主張するものではない。この実験で確かめられたのは、交叉を行う場合、親の代わりに導入される解が親より優れていればいるほど最終的に得られる解が良くなるということである。次節では子個体が親と比べどの程度優れているかという指標を用いることで交叉の性能を考察する。

表 3.2: （実験 2）生成子個体数 C_{cross} の与える影響

交叉	C_{cross}	最善評価値の平均	標準偏差	収束世代数	総評価回数
一様交叉	200	992.16	2.09	47.66	95520
	50	983.64	3.79	55.24	27670
	20	970.50	4.52	63.95	12810
	10	946.30	8.54	74.40	7450

実験 3：PR 値を用いた UX の特徴に関する考察

PR 値（Progress Rate）とは、主に ES（Evolutionary Strategy）の領域で使われる指標であり [Markon 2001]、1 ステップあたりの評価値の改善量の期待値として定義される。本論文では、これを GA の領域に導入するために、両親および子集団の中の最も優れた解が、親の良い方に比べてどれだけ優れているかという量として定義しなおす。すなわち関数 PR は親 p_1, p_2 および交叉法、生成子個体数 C_{cross} を引数とし、$\mathrm{PR}(p_1, p_2, \text{交叉}, C_{cross}) = E(\max_{x \in X_{family}} f(x) - \max_{x \in \{p_1, p_2\}} f(x))$ ただし $X_{family} = \{p_1, p_2, c_1, ..., c_{C_{cross}}\}$ と定義される。例えば、親 1 の評価値が 70、親 2 の評価値が 80、子個体の評価値が 65,75,90 だった場合 PR 値は 90-80=10 となる。子個体数を固定したとき、PR 値が高いほど良い交叉であると言うことができる。ここでは、両親の質が同等の場合と偏りがある場合を考慮にいれ、ビット数 $L_{bit} = 100$ とした上で、パラメータ m を用いて両親を $p_1 = \overbrace{000..000}^{1..m}\overbrace{111..111}^{m+1..100}$, $p_2 = \overbrace{111..111}^{1..m}\overbrace{000..000}^{m+1..100}$ と設定する。パラメータ m は両親の質の偏りを制御するパラメータであり、$m = 50$ のとき両者は質において同等である。

図 3.3 実線に $1 \leq m \leq 99$ とした場合の UX の PR 値を示す。この結果、$m \leq 30$ および $70 \leq m$ の領域、すなわち両親の質に大きな偏りがある場合は、UX はより良い子個体を生成できていないことがわかる。これは、一般的に用いられる UX が各要素を 1/2 の確率で選択するために、両親からほぼ等距離の領域にのみ子個体を生成するためである。例えば $m = 80$、すなわち親 2 が親 1 よりも非常に優れるような場合、親 2 を改善することができる解は、親 2 の近くでありながら親 1 の良い部分のみを導入した解であるはずであり、UX がこういった解を生成できる確率は非常に低くなる。

親 1 や親 2 の近くにも解を生成するために、UX における各遺伝子座の選択確率 p を 1/2 に固定せず、子個体生成のたびにランダムに決定する交叉を、仮に UXr と呼ぶことにす

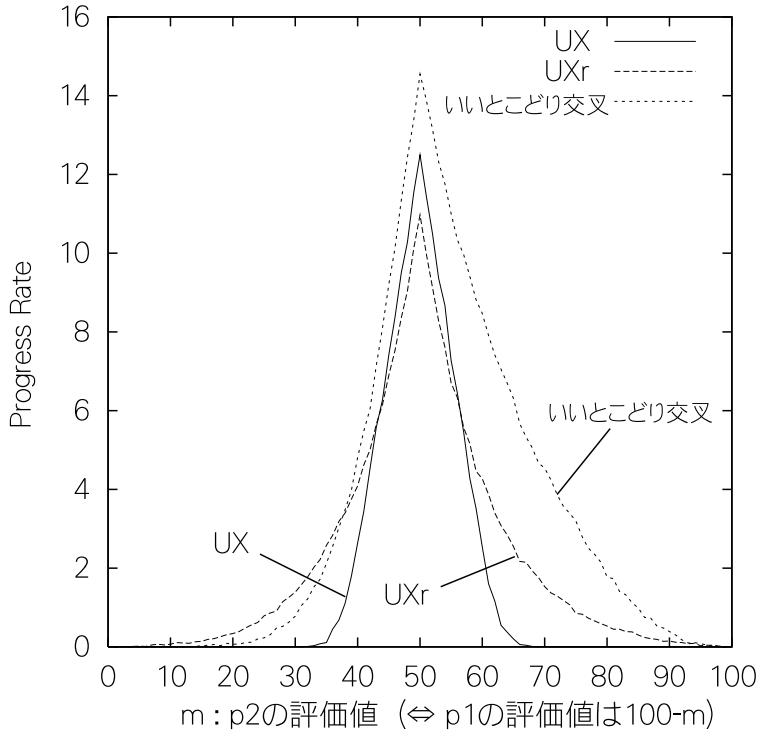


図 3.3: （実験 3, 実験 4a）両親の質の偏りを横軸に取った場合の PR 値：UX（実線）, UXr（破線）, いいとこどり交叉（点線）

る。UXr の PR 値は図 3.3 破線に示される。UXr は $m = 20, 80$ といった領域でも両親を改善する子個体を生成することに成功しているが、一方 $m = 50$ 付近での改善量は UX に劣っている。

3.4.2 いいとこどり交叉の性能評価

いいとこどり交叉を 1max 問題に適用するためには、親 2 に近づく近傍集合を定義しなければならない。親 1 と親 2 が n つの遺伝子座で異なる場合、探索解 x_k から親 2 に近づく解 $y_j \in \mathcal{N}_{\rightarrow p_2}(x_k)$ は、x_k に p_2 と異なる遺伝子座を約 n/k_{max} 個導入することで生成される。このとき、y_i は x_k に比べ Hamming 距離において p_2 に近づくことになり、k_{max} ステップ後には最終探索点 $x_{k_{max}}$ はほぼ p_2 と等しくなる。1max 問題におけるいいとこどり交叉の挙動例を表 3.3 に示す。

いいとこどり交叉を用いた場合の世代交代モデルとしては以下のような CCM モデル [Ono 98] の修正版（便宜上 CCM リレーと呼ぶ）を用いる。この世代交代モデルを用いた

表 3.3: 1max 問題における いいとこどり交叉の挙動例 ： パラメータ $k_{max} = 3,\ \mu = 3.$ ビット数 $L_{bit} = 10,\ p_1$ =0000111111, p_2 =1111110000, このとき n=8。 x_k のビット上のバーは x_k が p_2 と異なる部分を表し、候補解 y_i のビット上のハットはその部分が p_2 から導入されたことを表す。解の後ろの数値はその解の評価値である。

	step 1	step 2	step 3
(探索解 x_k)	$\bar{0}\bar{0}\bar{0}\bar{0}11\bar{1}\bar{1}\bar{1}\bar{1}$ (6)	$11\bar{0}\bar{0}11\bar{1}\bar{1}0\bar{1}$ (7)	$111111\bar{1}00\bar{1}$ (8)
(解候補 $y_i \in \mathcal{N}_{\rightarrow p_2}(x_k)$)	$\hat{1}\hat{1}00111\hat{1}\hat{0}1$ (**7**) $\hookrightarrow$ $000\hat{1}11\hat{0}111$ (6) $00001\hat{1}\hat{0}\hat{0}\hat{0}1$ (3)	$1100\hat{1}1\hat{0}\hat{0}\hat{0}\hat{0}$ (4) $11\hat{1}011\hat{0}101$ (7) $11\hat{1}\hat{1}111\hat{0}01$ (**8**) $\hookrightarrow$	p_2 1111110000 (6)

場合、一世代あたり $k_{max}{\cdot}\mu{\cdot}N_{pop}$ 個の個体が生成され、評価される。

CCM リレーモデル

1. 現集団をランダムな順序に並べ替え、$p_1, ..., p_{N_{pop}}$ とする。
2. p_1 を $p_{N_{pop}+1}$ に複製しておく。
3. （ステップ 1）p_1 から p_2 に向かっていいとこどり交叉を用いて子個体を生成し、p_1 および子個体群の中から最も優れたものを p_1 のかわりに集団に戻す。
4. （ステップ i）以下同様に p_i から p_{i+1} に向かって同様の操作を行う。

実験 4：PR 値における UX との比較

本節ではまず、いいとこどり交叉を PR 値の観点から UX および UXr と比較する。1 世代あたりに生成評価される子個体の数を UX および UXr と同数にするためには、$k_{max}{\cdot}\mu = C_{cross}/2$ としなければならない。そこで、ここでは $k_{max} = 10,\ \mu = 5$ と設定し、実験 3 と同様の比較を行う。図 3.3 の点線がいいとこどり交叉の PR 値であり、全ての m において UX を上回っていることがわかる。これは、親の質が偏っている場合にも優秀な親の近くに改善解を生成でき、同程度の場合にも両親の良い要素を受け継ぐことができていることを示している。

次に、生成子個体数を変化させた場合の UX,UXr, いいとこどり交叉の PR 値の変化を調べる。両親は同等の質すなわち $m = 50$ とし、パラメータ (k_{max}, μ) を $(4, 2), (6, 3), ..., (20, 10),$ と変化させる。UX,UXr に対しては、1 世代あたりに生成評価される個体数を合わせるため、$C_{cross} = 2{\cdot}k_{max}{\cdot}\mu$ とする。図 3.4 に示すとおり、生成子個体数が増えるほどにいいとこどり交叉の UX,UXr に対する優位性は大きくなる。

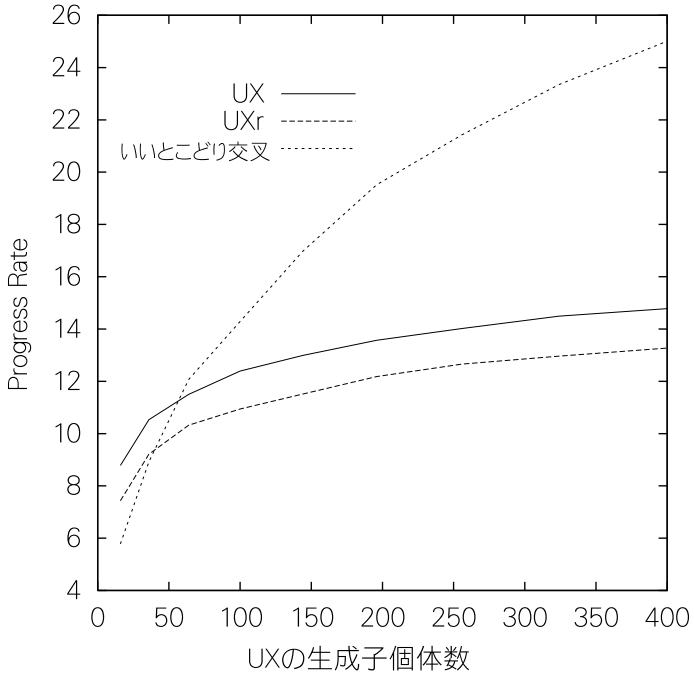


図 3.4: （実験 4b）生成子個体数を変化させた場合の PR 値：UX（実線）, UXr（破線）, いいとこどり交叉交叉（点線）

実験 5：従来の交叉といいとこどり交叉を用いた GA の性能比較

PR 値という観点ではいいとこどり交叉の性能が UX 等に比べ優れるということが明らかになった。本節では、実際にいくつかの交叉を用いて実験 1 と同じ、同等の条件下で GA を試行し、その結果を比較する。比較するのは、UX, UXr, 6 つのパラメータセット (k_{max}, μ) を用いたいいとこどり交叉, および好ましい (k_{max}, μ) に加え温度パラメータ $T = 1, 2$ を用いた MSXF である。

表 3.4 に結果をまとめる。これらの結果から以下のようなことがわかる。

- いいとこどり交叉による最も良い結果は最適解からのエラー率が 0.16 %であり、これは UX の 1/5 程度である。収束も UX に比べ早い。
- いいとこどり交叉では非常に広い範囲のパラメータを用いているにもかかわらず、性能の劣化は非常にゆるやかである。これは、いいとこどり交叉を設計する際にはさほどパラメータ設定に敏感にならずともよいということを示している。
- MSXF の温度 $T = 1$ は UX よりも良い性能を示すが、パラメータを $T = 2$ に変動さ

表 3.4: （実験 5）UX, UXr, いいとこどり交叉 , MSXF の性能比較

交叉	パラメータ ,(世代あたりの生成子個体数)	最善評価値の平均	標準偏差	収束世代数
UX	$C_{cross} = 200$ (2000)	992.16	2.09	47.66
UXr	$C_{cross} = 200$ (2000)	986.50	3.52	51.30
いいとこどり交叉	$k_{max} = 5,\ \mu = 20$ (2000)	996.13	2.27	41.36
	$k_{max} = 7,\ \mu = 14$ (1960)	997.23	1.58	39.50
	$k_{max} = 10,\ \mu = 10$ (2000)	997.89	1.22	37.75
	$k_{max} = 14,\ \mu = 7$ (1960)	**998.40**	1.36	36.66
	$k_{max} = 20,\ \mu = 5$ (2000)	998.25	1.08	36.10
	$k_{max} = 50,\ \mu = 2$ (2000)	997.25	1.75	39.55
MSXF	$k_{max}=14,\ \mu=7,\ T=1$ (1960)	996.39	2.97	48.69
	$k_{max}=14,\ \mu=7,\ T=2$ (1960)	985.79	4.70	55.60

せるとエラー率は 3 倍も悪くなる。

1max 問題における本節の実験・解析により、変数間依存関係のない問題であってもいいとこどり交叉が UX などの従来手法に比べて高性能、かつ十分に頑健であることが示せた。

3.5 巡回セールスマン問題への適用

巡回セールスマン問題 TSP は、各都市間に距離が定められているグラフにおいて、全ての都市を一度ずつ訪問して一周する Hamilton cycle（ツアーと呼ばれる）の中で最も経路長が短いものを求める問題である。TSP は古典的な NP 困難問題であり、非常に多くの最適化手法が解法として提案されている [Reinelt 94]。

枝組立て交叉 EAX は芸術の域とも言える TSP 特有のすばらしい発見的交叉であり、EAX を用いた GA は他の手法の中でも非常に優秀な性能をあげている [Shimodaira 99]。本節では、EAX を「両親のツアーを AB サイクルと呼ばれる独立した要素に分けて一様交叉する操作」と捉えた上で、一様交叉する代わりにいいとこどり交叉の枠組みを導入することにより、さらなる性能の向上を目指す。

3.5.1 枝組立て交叉 EAX

ツアー A とツアー B が親として与えられたとき、EAX は以下の手順で子個体を生成する。

EAX による子の生成

1. G' 上の枝（グラフのエッジ）を複数の AB サイクルに分割する。ここで G' はツアー A とツアー B を重ねたものである。一つの AB サイクルは、接続するツアー A のエッジとツアー B のエッジを交互に選択していくことで得られる。（図 3.5c）
2. AB サイクルの集合から、何らかの基準に基づいて幾つかを選択し、*E-set* と呼ぶ。
3. *E-set* をツアー A に排他的論理加算することで緩和個体を生成する。すなわち、ツアー A と *E-set* が共に持つ枝を取り除き、*E-set* だけが持つ枝を加える。（図 3.5d）
4. 緩和個体を実行可能解にするための修正を行う。この修正操作は決定論的な手続きによってなされる。（図 3.5e）

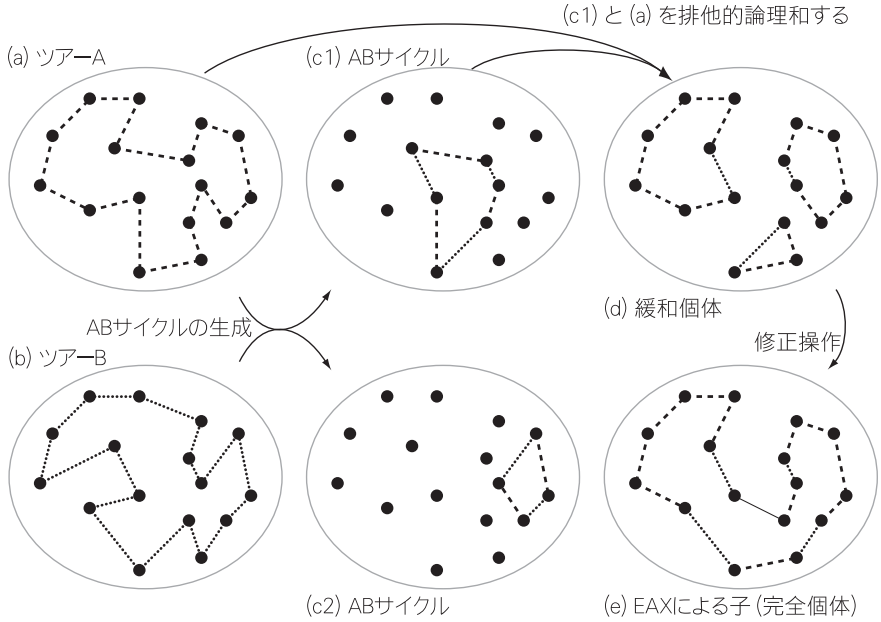


図 3.5: EAX の実行例: (a) ツアー A および (b) ツアー B が所与のとき、AB サイクル (c1)(c2) などが生成され、(d) ツアー A といくつかの AB サイクルを排他的論理和することで緩和個体を生成し、(e) 緩和個体を修正することで実行可能解が得られる

手順 2 において AB サイクルを選択する基準はいくつか提案されている。[Nagata 97] においてはランダム選択基準と、問題の性質を考慮した込み入った選択基準が提案されている。ランダム選択基準による EAX（ここでは EAX-rand と呼ぶ； 図 3.6a）は全ての AB サイクルを 1/2 の確率で選択するものであり、これは AB サイクルという独立した要素を

一様交叉している操作であると捉えることができる。

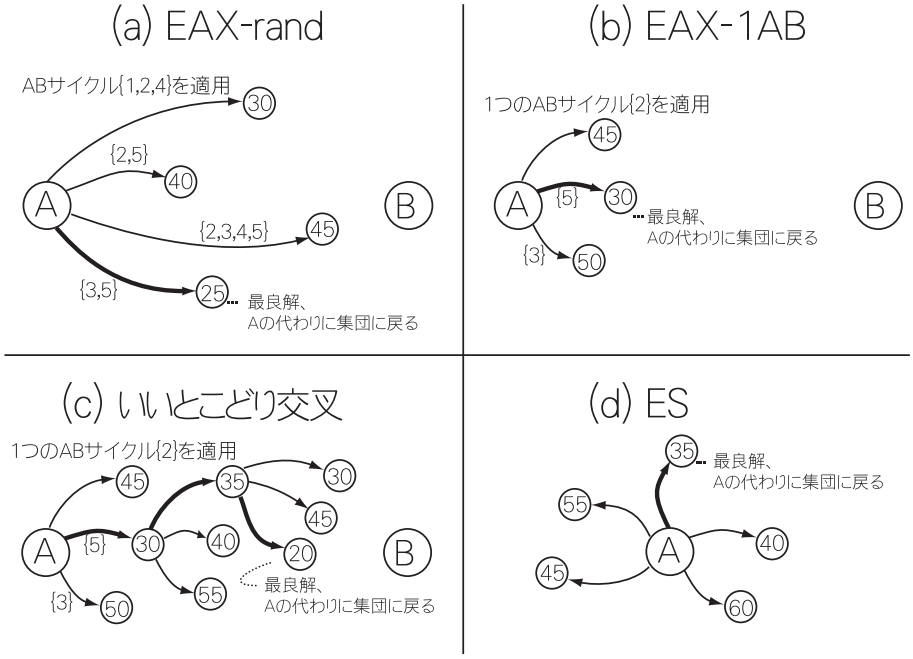


図 3.6: 探索の模式図: (a)EAX-rand, (b)EAX-1AB; いいとこどり交叉の 1 ステップ版と捉えられる。 (c) いいとこどり交叉, (d)ES; EAX-1AB と比較すると、探索が等方的である。

3.5.2 EAX-1AB を用いたいいとこどり交叉

EAX-rand は両親の枝を独立した要素に分解して一様交叉を行うため、良い要素も悪い要素も同時に子に受け継がれることになる。一方 EAX-1AB では、一つの AB サイクルのみを $E-set$ として選択し、親 A に親 B の要素の一部分のみを移植した子個体を生成する。これらの子個体は「親 A の、親 B へ近づく近傍」と捉えることができる（図 3.6b)。実際、ツアー X とツアー Y の距離 $d(X,Y)$ を“ X と Y が共有しない枝の本数 ”で定義するとき、任意の子個体 y_j に対して $d(y_j,\text{ tour-B}) \leq d(\text{tour-A, tour-B})$ が成立している。

EAX-1AB のアルゴリズムを用いることで「ある解から、親 B へ近づく近傍」を生成することができることがわかった。よって、ここからただちに、この近傍構造を用いた TSP におけるいいとこどり交叉を構成することができる（図3.6c)。本節では、EAX-rand, EAX-1AB, EAX-1AB を用いたいいとこどり交叉を 7 つのインスタンスで比較する。パラメータと実験設定は以下の通りである。

- EAX-rand, EAX-1AB における生成子個体数 $C_{cross} = 10$ とする。生成子個体数を大きくすると若干の性能の向上が見られるがその分計算時間が大きくなるため、10 程度が良いとされている。
- いいとこどり交叉におけるパラメータ (k_{max}, μ) を、1000 都市未満では (4,6)、以上では（5,8）とする。
- 集団サイズは 300 または 1000 とした。
- 初期解は 2opt ローカルサーチを用いて生成した。
- 突然変異は用いない。
- 30 世代にわたり集団内の最善解の評価値が改善しなかった場合探索を打ち切る。
- 全ての設定において 30 試行を行い、最適解発見回数、平均エラー率、平均 CPU 時間を比較する。なお CPU には 1GHz の Pentium3 を用いた。

表 3.5: EAX-rand, EAX-1AB, いいとこどり交叉 の性能比較：opt は最適解発見回数/30 試行, error は平均エラー率（%）、time は平均 CPU 時間（秒）を表す。

		EAX-rand			EAX-1AB			いいとこどり交叉		
問題	集団サイズ	opt.	error	time	opt.	error	time	opt.	error	time
att532	300	10	0.025	29	23	0.006	20	27	0.002	24
rat575	300	3	0.020	47	8	0.012	27	17	0.006	30
rat783	300	20	0.006	72	28	0.001	42	30	0	49
pcb1173	300	2	0.023	172	10	0.017	71	12	0.010	79
pr2392	300	0	0.034	657	18	0.048	365	23	0.026	473
fl3795	300	2	0.028	2716	11	0.009	1768	17	0.006	2301
fnl4461	1000	0	0.031	10876	3	0.002	6952	8	0.001	9784

この結果、$EAX - rand$ に比べ、この近傍構造を用いたいいとこどり交叉は非常に良い解を得られていることがわかる。本節では一様交叉といいとこどり交叉を比較するために、AB サイクルという TSP に固有の構造を利用した交叉のみを比較したが、3-opt などこれ以外の近傍構造を用いることでもいいとこどり交叉は構成することができる。なお、これらの結果は他のアルゴリズムを TSP に適用した結果と比較しても非常に優れている。例えば、att532 問題に対して分枝限定法を適用した場合、IBM3090/600 において

5時間を要し [Padberg 91]、TSP 専用の GA である Asparagos96[Schleuter 97] は 170MHz の CPU で 95 分を要している。

3.6 まとめ

本章では、問題ごとに近傍構造と距離尺度を導入することだけで交叉を容易に構成する MSX のアイデアと、選択的に相手親の持つ良い要素のみを受け継ぐ EAX-1AB のアイデアを用いて、GA の最も本質的なオペレータである交叉の設計を行う方法を論じた。最適化の流れを考慮にいれると、交叉に求められるのは親の要素を混合するということではなく、親の良い部分だけをえりすぐることである。

いいとこどり交叉は、相手親に近づく近傍構造を導入することでただちに構成される、汎用的交叉である。いいとこどり交叉の持つ特徴は、その構成が非常に容易であるというだけでなく、一様交叉等に比べて優れた探索性能を示し、その上パラメータに対して頑健であるということである。離散の組み合わせ最適化問題や複雑な制約条件を持つ問題では特に高性能な汎用交叉の需要は高く、いいとこどり交叉の持つ意義は大きい。

なおいいとこどり交叉は完全に内挿的な交叉であり、単独で用いる場合は初期集団の覆う領域が最適解を含む、あるいは最適解に必要な要素を初期集団が含むことを前提にしている。このようなことが期待できない場合、たとえば実数値最適化問題に用いる場合には、突然変異オペレータを用いるか、あるいは EDX[Sakuma 2000] のような外挿的な交叉を用いる必要がある。いいとこどり交叉の適用可能クラスおよび設計指針、パラメータの設定法については付録 A に付す。

第4章　大域的多峰性を考慮した世代交代モデルの設計

本章では、従来 GA が想定していた大谷構造が困難な問題の多くには存在しておらず、複数の有力な局所解が離れて存在している可能性を考慮した上で、GA の適用可能クラスを広範化するために、このような場合でも大域的最適解を得ることができる世代交代モデルを提案することを目標とする。

4.1　はじめに

Boese が提唱する大谷構造仮説 (big valley structure hypothesis)[Boese 94,95] は、微視的に見れば多数の小さな谷が存在し多数の局所解が存在するが、巨視的に見れば大きな谷が一つだけ存在しこの谷が最適解になっている景観、すなわち大谷構造のもとでは GA が理想的な挙動を示すと説明している。

しかし、例えば Fletcher and Powell 関数（以下 FP 関数）[FP63] やジョブショップスケジューリング問題（JSP）のように景観が複雑な問題に GA を適用したとき、最適解の発見にしばしば失敗する事実は、大谷構造仮説が成立しない問題クラスが存在することを示している。

探索空間に大きな谷が複数存在し、かつ各谷に有力な局所解が存在するとき、このような景観を**大域的多峰** (globally multimodal) と呼ぶ [Ikeda 2000]。大域的多峰性は、GA にとって難しい実数値最適化問題や組み合せ最適化問題に共通する特徴である。大域的多峰な問題では、GA の実行過程において、探索領域の絞り込みを適切に制御することが難しいために、最適解を含む谷を外して、その他の谷へ探索が重点化されてしまう恐れがある。

本章では、大域的多峰性問題における GA に特徴的な探索失敗現象を **UV 現象**と名付

け、これを引き起こす景観構造を説明する仮説として**UV 構造仮説**を提唱する。UV 構造仮説の妥当性を検証するために、UV 構造を持つことが自明なテスト関数を設計し、実験により UV 現象との因果関係を明らかにする。ついで、困難な実数値最適化問題として知られる FP 関数を対象に UV 構造仮説に基づく UV 現象の解析を行い、UV 構造仮説の観点から多様性維持を考慮した従来の世代交代モデルの特徴と限界を考察する。さらに、GA の新しいモデルを提案した上で、最も困難な組み合わせ最適化問題の一つとされる JSP にこれを適用し、その有効性を示す。

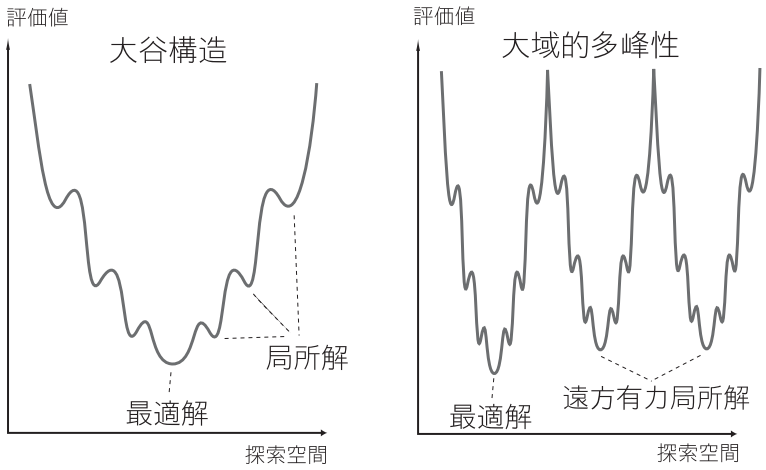


図 4.1: 大谷構造と大域的多峰

大谷構造

図 4.1 左に示すように、探索空間に局所解が多数存在するにもかかわらず、巨視的には大きな谷が一つだけあって、その中心近くに最適解が存在する景観を**大谷構造**という。Rastrigin 関数 [EP5] の景観は大谷構造の典型例である。実数値 GA により Rastrigin 関数の最適解を高確率で発見できる事実 [Ono 97] は、大谷構造仮説の妥当性を示している。

しかし、本論文で取り上げる FP 関数、あるいは JSP や二次割り当て問題 (Quadratic Assignment Problem : QAP)、二重円 TSP のように大谷構造を想定できない問題は多数存在し、また最適化が困難な実問題も同様である。

大域的多峰性

図 4.1 右に示すように、探索空間に複数の大きな谷が存在し、各谷が有力な局所解を持つ場合を考える。ただし、各谷の景観をそれぞれ単独で見れば大谷構造仮説に従うものとする。このような景観を**大域的多峰**であると言い、各谷の底にある有力な局所解を**遠方有力局所解**という。

大域的多峰性は、FP 関数、JSP、QAP、二重円 TSP などに共通する特徴である。大域的多峰性のもとでは、従来の GA は実行過程において、探索領域の絞り込みを適切に制御することが難しいために、最適解を含む谷を外してその他の谷へ探索が重点化されてしまう恐れがある。このような現象は騙し (deception) と呼ばれるが [高橋 2001]、その詳細なメカニズムは今まで明らかにされてこなかった。

したがって、大域的多峰性のもとで従来の GA が探索に失敗する現象とその原因をより的確に説明する仮説、ならびにそれに基づく新しい GA の枠組みが求められる。

4.2 問題意識：UV 現象と UV 構造仮説

4.2.1 UV 構造仮説

GA が困難な問題では必ずしも最適解を発見できず、局所解に収束してしまう可能性があることはよく知られたことである。この中でも本論文では、大域的多峰性問題において、集団が最適解から遠く離れた局所解に収束してしまう現象に着目する。これは詳しく見ると、最初には探索空間に一様に分布していた個体群が、世代を経るに従い最適解付近よりも遠方の局所解付近に集まり始め、ついには最適解付近には存在しなくなり遠方の局所解に収束する、という過程を経る。これを、後述の景観構造のイメージに基づき **UV 現象**と名付ける。UV 現象の特徴として、問題ごとに収束しやすい有力局所解が存在するということが挙げられる。例えば、小野らによる JSP への GA の適用 [小野 98] においては、いくつかのテスト問題で、10 試行中全試行で全く同じ遠方の局所解に収束している。

我々は、UV 現象は次のようなプロセスで生じているとする、**UV 構造仮説**を提唱する。

まず、2 つの特徴的な谷が存在する景観を考える。一つは遠方有力局所解を含み他方より広く良好な解を発見することが容易である U 谷、もう一つは最適解を含むものの狭く

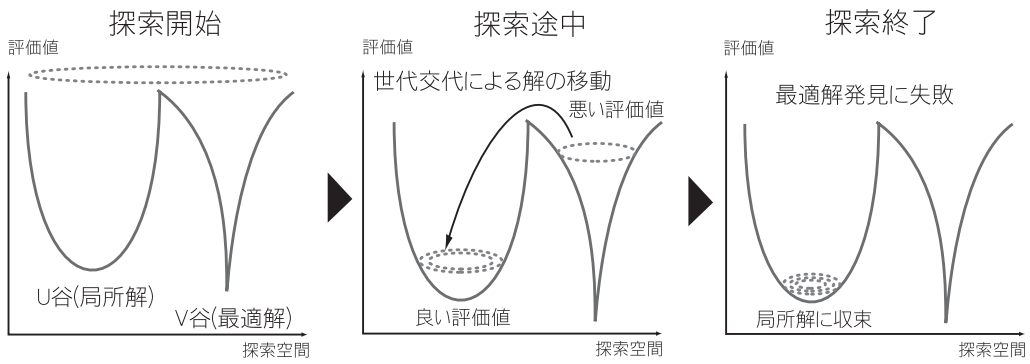


図 4.2: UV 現象

なかなか良い解が見つからない V 谷であるとし、これを典型的 UV 構造と呼ぶ。

図 4.2 を用いて、典型的 UV 構造における従来の GA の挙動を説明する。探索序盤において U 谷と V 谷の両方に同数の個体が存在したとしても、両者の構造の違いから、V 谷に比べて U 谷に存在する個体の評価値のほうが良くなる。そのため、評価値に基づく生存選択を行えば、世代交代を重ねるにつれて、V 谷に存在する個体が淘汰される割合が高く、探索は U 谷に重点化されることになり、最適解の発見に失敗する結果に終わる可能性が大きい。一旦探索の誤った重点化が起きると、最適解を含む谷の再発見が極めて困難になるのが大域的多峰性の特徴である。それは、高性能な GA は通常、現集団が有望な領域を探索していると考え、その周辺に子個体を生成するからである。古典的なバイナリ GA においては、遠く離れた場所に子個体が生成される場合も考えられるが、これはランダムサンプリングに近く、UV 現象が生じるかどうか以前に優秀な性能を期待できない。

典型的 UV 構造を持たなくても、なんらかの景観構造によって、探索途中において最適解の谷の個体の評価値が局所解側の谷の個体よりも劣れば、上記のプロセスを経て UV 現象が生ずる。このような景観構造を、**UV 構造**と呼ぶ。U 谷 V 谷というのはあくまでイメージであり、このような現象を生じさせる谷が幾何的に見て U 型 V 型をしているということを主張するものではない。また、U 谷 V 谷の定義は相対的なものであり、ある一つの谷だけをもってそれが U 谷であるか V 谷であるかを定めることはできない。これらの議論をまとめる。

UV 構造仮説

UV 現象が頻繁に生じる場合、各谷の景観構造の違いから、最適解の谷の個体の評価値が局所解の谷の個体の評価値よりも劣る可能性が大きくなっていると考えられる。このような構造を **UV 構造**と呼ぶ。この場合、世代交代を重ねるにつれ最適解の谷の個体が淘汰され、遠方の局所解周辺のみを探索するようになり、最適解の発見に失敗していると考えられる。

4.2.2 UV 構造のクラス分け

UV 構造は最適解の谷の個体の評価値を相対的に劣らせる構造としてしか定義されていない。本節では、UV 構造が原因となって UV 現象を引き起こす過程の違いに着目して、UV 構造を 3 つのクラスに分類する。用いる手法やそれらが想定する近傍系によって、同じ関数であっても GA の挙動は大きく異なるため、UV 現象の原因となっている UV 構造を、現象とは切り離した形で明確に列挙することは不可能である。また、本章の主旨は、いくつかの明解な但し限られた問題クラスで GA が探索に失敗するという例を挙げることにあるのではなく、従来の理論では説明できなかった GA の挙動を解析してその問題点を明らかにすることにある。そこで本論文では、各クラスごとに代表的な構造を定性的に記述し、ある関数がその構造を持つかどうかを判定する指標を導入することで一般的な議論を行う。また、実際の問題においては各クラスは排他的ではなく、例えばクラス 1 とクラス 2 の特徴を同時に持つような構造もありうることに注意されたい。

クラス 1：探索序盤での有望さに影響する構造

代表的構造とその確認:　最適解を含む谷が全体として浅い構造。前述の典型的 UV 構造はこの一例である。この構造は、その谷から個体を一様にサンプルしたときの平均評価値が他の谷のそれよりも悪いことで確認される。

UV 現象が起こる過程:　上記のような構造が存在すると、個体の分布が一様に近い探索序盤では最適解を含む谷の評価値が悪く、進化がある程度進まないと優れた評価値を持つ個体の発見が難しくなる。このとき、図 4.3 に示すように、他の谷の個体との比較にお

いて最適解を含む谷が劣ると判断される。この場合、谷の真の有望さが明らかになる前に、最適解を含む谷が探索範囲から除外されてしまう可能性が大きい。

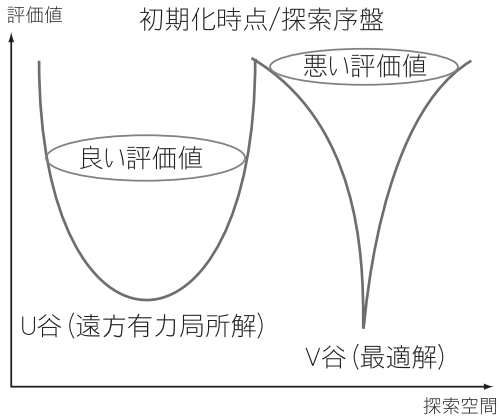


図 4.3: UV 構造 クラス 1

クラス 2：探索個体数の多寡に影響する構造

代表的構造とその確認: 1. 最適解を含む谷が他の谷に比べて相対的に小さい構造。探索空間中にランダムに解を多数生成し、局所的探索法で谷の底に到達させたとき、各谷への到達割合で谷の大きさが定義できる。 2. 最適解を含む谷が探索空間の辺境付近に位置する構造。ある谷を見たとき、その谷底とランダムな点との平均距離が大きいほど、辺境に位置すると定義できる。

UV 現象が起こる過程: 上記のような構造が存在すると、最適解を含む谷を探索する個体の数が相対的に少なくなる。このとき、交叉を行う際に最適解を含む谷から両親が選ばれる確率は 2 乗のオーダで低下し、他の谷に比べてより優秀な個体が生成される可能性は大きく低下する。このことは、最適解を含む谷を探索する個体の評価値が他の個体に比べて劣ることにつながり、誤った探索の重点化により最適解を含む谷が探索範囲から除外されてしまう可能性が大きい。

クラス 3： 同条件下での進化速度に影響する構造

代表的構造: 1. 最適解を含む谷が他の谷に比べて、Rosenbrock 関数 [EP5] のように非線型性が強い構造。 2. 最適解を含む谷が他の谷に比べて、より多くの局所解を含む

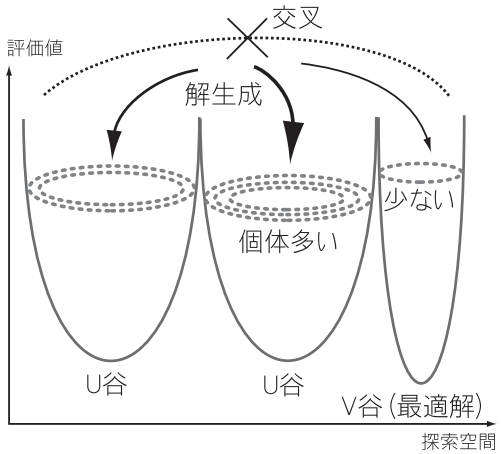


図 4.4: UV 構造 クラス 2

構造。

UV 現象が起こる過程:　上記のような構造が存在すると、仮に最適解を含む谷の探索が他の谷と同じ個体数で行われたとしても、評価値の改善速度が他の谷よりも劣ることになる。このとき、図 4.5 に示すように世代を経るにつれ評価値が劣り、UV 現象が生じる。

構造確認のための指標:　谷の非線型性や局所解の数は現象と切り離して定義することも可能であろうが、その実際の影響度合は用いる GA のオペレータにより大きく異なる。そこで、各谷内に一様に初期個体を生成し、その谷だけを独立で GA で探索させてその挙動を観察する。各世代での平均評価値及び谷底発見までにかかる世代数を比較することで、どの谷は進化速度が遅くどの谷が速いのかを知ることができる。これは手法依存の定義になるが、クラス 3 に属する UV 構造を抜き出すには十分である。

4.2.3　テスト関数による仮説の検証

本節では、まず UV 構造仮説の 3 つのクラスを自明な構造として持つテスト関数を提案し、UV 現象との因果関係を明らかにすることにより、UV 構造仮説の妥当性を検証する。

テスト関数 f_9 は、図 4.6 に示すように、9 つの谷を持つ関数であり、1 つの谷だけが沈んだ景観を持つ。f_9 は最小化問題であり、最小値は 0 である。また f_9 はパラメータ $a, b, c_{opt}, c_{loc}, d_{opt}, d_{loc}$ を変えることによりそれぞれ特徴のある景観をつくることができる。

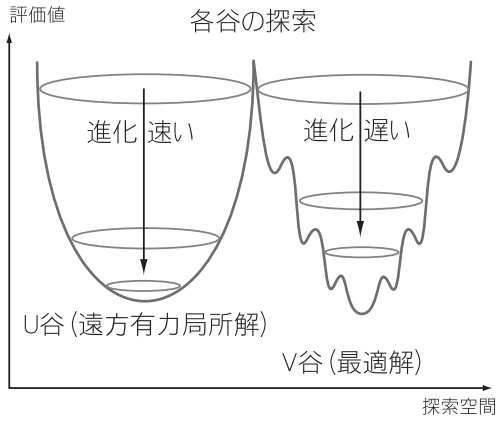


図 4.5: UV 構造 クラス 3

f_9 関数

$$f_9(x_1, x_2) := \sum_{i=1}^{2} \{4(x_i - \lfloor x_i \rfloor - 0.5)^2 + bB(x_i)\} + A_{x_1x_2} \quad x_1, x_2 \in (0, 3)$$

$$B(x_i) := 1 - \left\{ \frac{1 + cos(2\pi C_{x_1x_2} x_i)}{2} \right\}^{D_{x_1x_2}}$$

$$(A_{x_1x_2}, C_{x_1x_2}, D_{x_1x_2}) := \begin{cases} (0, c_{opt}, d_{opt}), & \text{if } \lfloor x_1 \rfloor = r_1, \lfloor x_2 \rfloor = r_2 \\ (a, c_{loc}, d_{loc}), & \text{otherwise} \end{cases}$$

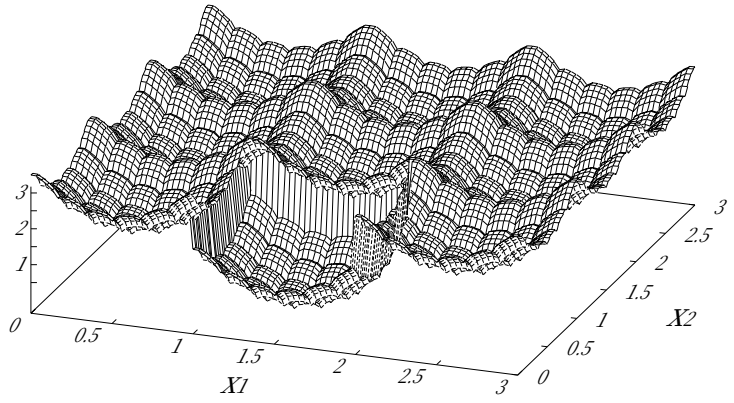


図 4.6: f_9 関数の例 : a=1.5, b=0.1, c_{opt}=c_{loc}=6, d_{opt}=d_{loc}=4 , r_1=1, r_2=0

以下の実験では、a=0.005, b=1, c_{opt}=c_{loc}=16, d_{opt}=d_{loc}=4 をデフォルトとする。f_{9c} を最適解が中央の谷にある関数 (r_1=1,r_2=1)、f_{9s} を最適解が辺にある関数 (r_1=1,r_2=0)、 f_{9e}

を最適解が隅にある関数 (r_1=0,r_2=0) としてそれぞれ定義する。f_{9e} の持つ特徴は UV 構造クラス 2 に属する。

f_{9s+} は f_{9s} において d_{opt}=8 とした関数であり、最適解の谷だけ溝が狭く進化序盤では評価値が改善されにくい。ちなみに、f_{9s+} では最適解の谷の平均評価値は 2.274, 他の谷は 2.124 であり、f_{9s+} 以外の関数では最適解の谷の平均評価値は 2.119 である。f_{9s+} の持つ特徴は UV 構造クラス 1 に属する。

f_{9s*} は、f_{9s} において c_{loc}=4 とした関数である。遠方有力局所解の谷は最適解の谷に比べ凸凹が少なく、進化が早く進む。各谷ごとに従来の GA を適用したところ（設定は後述)、最適解の谷は収束に 44 世代を要したのに対し、他の谷は 21 世代で収束した。f_{9s*} の持つ特徴は UV 構造クラス 3 に属する。

以上 5 つのテスト関数の特徴をまとめたものを表 4.1 に示す。

表 4.1: 各テスト関数の特徴

関数	f_{9s}	f_{9s+}	f_{9c}	f_{9e}	f_{9s*}
最適解の谷の評価値	良い	**悪い**	良い	良い	良い
最適解の位置	辺	辺	**中央**	**隅**	辺
最適解の谷の収束	同じ	同じ	同じ	同じ	**遅い**
UV 構造クラス	-	1	-	2	3

各テスト関数の困難さを調べるために、従来の GA を適用した。実験には、実数値最適化に現在最も適していると思われる、交叉 UNDX[Ono 97](生成子個体数 100) と世代交代モデル DDA [Takahashi 99] を用いた。高性能な GA は、現集団が探索空間中の有望な領域にあると考え、その周辺に子個体を生成することで効率的な探索を行う。本論文で採用した UNDX と DDA はそのような GA の一例にすぎず、UV 現象に関しては他の優秀な GA にも同様の傾向が見られるであろうことは容易に想像できる。

実験は集団サイズを変えて各 60 試行行った。また、打ち切り世代数は 1000、最良評価値が 10^{-7} 以下になったときを最適解発見とみなした。

結果を表 4.2 に示す。いずれの場合も、最適解を発見するか、最適解を含む谷から個体がなくなり別の谷の局所解に収束するという 2 つのケースだけが確認された。後者は UV 現象が生じて探索に失敗したことを意味する。

表 4.2: f_9 関数を GA で解いたときの最適解発見頻度

集団サイズ	f_{9s}	f_{9s+}	f_{9c}	f_{9e}	f_{9s*}
20	11/60	2/60	46/60	4/60	5/60
100	34/60	13/60	60/60	8/60	16/60
500	60/60	28/60	60/60	27/60	54/60
UV 構造クラス	-	1	-	2	3

この結果からまず、クラス 1 の UV 構造を持つ f_{9s+} は f_{9s} よりも非常に難しい、すなわち UV 現象が生じやいことがわかる。

次に、f_{9c},f_{9s},f_{9e} の 3 つの中では f_{9c} が最も簡単で、f_{9e} が最も難しいことがわかる。これは、クラス 2 の最適解が隅に位置するという理由により UV 現象が生じていることを示す。同じ大きさの谷でも、位置によって困難さが変わるのには、GA の交叉オペレータの特性が関係している。本論文で採用した交叉 UNDX は、親集団の平均・分散・共分散を保存する [喜多 99] ものの、集団分布の様子を変えてしまう可能性は否定できない。実際、一様な初期集団は各谷に 1/9 ずつ個体を持つが、この親集団から生成される子個体群の分布は、中央の谷 20 %、辺 12 %、端の谷 8 %と大きな偏りを持つ。

最後に、クラス 3 の UV 構造を持つ f_{9s*} は f_{9s} よりも難しくなっていることがわかる。

以上のことから、UV 構造仮説の各クラスがテスト関数 f_9 において UV 現象を起こすことが確認され、UV 構造仮説の妥当性が検証されたといえる。

4.2.4 UV 構造仮説に基づく FP 関数の解析

本節では、困難な実数値最適化問題として知られる FP 関数を取り上げ、UV 構造仮説に基づいて UV 現象に関する知見を得るための解析を行う。FP 関数は複雑な変数間依存関係と強い悪スケール性を持った多峰性関数であり、GA 以外の最適化手法にとっても最適解を得ることが非常に困難なことが知られている。

FP 関数

FP 関数は、次式で定義される N 次元の実数値多峰性関数であり、その最小化問題は定義により自明な最適値 0 を持つ。

FP 関数

$$F(x) := \sum_{i=1}^{n}(A_i - B_i)^2$$

$$A_i := \sum_{j=1}^{n}(a_{ij}sin\ \alpha_j + b_{ij}cos\ \alpha_j) \quad B_i := \sum_{j=1}^{n}(a_{ij}sinx_j + b_{ij}cosx_j)$$

$$-\pi \leq x_i \leq \pi,$$

定数 a_{ij}, b_{ij}, α_j は [Bäck 96] に与えられている。問題の景観は次元数 N によって異なるが、どの次元数であっても他に数種の最適解と多数の局所解が十分離れて存在する大域的多峰性関数であることがわかっている。

本論文では、事前の実験により UV 現象が最も顕著に観察される 12 次元問題をとりあげ、仮説に基づく解析を行った。FP 関数では 9 次元,13 次元,14 次元,15 次元等にも UV 現象が観察され、12 次元はその典型にすぎない。なお、FP 関数は各軸において周期関数になっているが、本論文で用いる距離や交叉はこの周期性を考慮したものになっている。

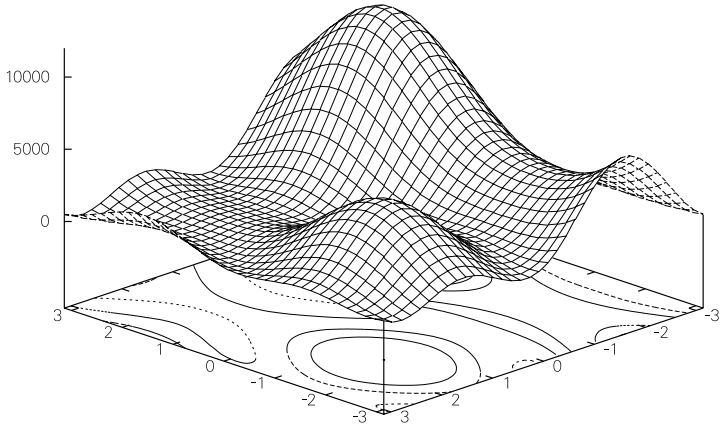


図 4.7: 2 次元 FP 関数

大域的多峰性の確認

12 次元 FP 関数では 4 個の最適解と 11 個の局所解が確認されているが、これらを区別するために便宜上名前をつける。最適解については 1 次元目の値の小さい順に A,B,C,D とし、局所解については評価値の良い順に a,b,c,...,k と記号を割り当てる。

まず、12 次元 FP 関数に大域的多峰性があることを確かめるために 15 個の最適解・局所解同士の距離を測った。その結果、A と C は非常に近く 0.3827 しか離れていないが、次に近い 2 つは a と c で 0.9663、平均は 3.8069 であった。これは定義域が一辺 2π の超立方体であることを考えると十分大きい。12 次元 FP 関数は有力な局所解がそれぞれ離れて存在する関数であることがわかる。

UV 現象の確認

UV 現象を確認するために、12 次元 FP 関数を交叉 UNDX および世代交代モデル DDA を用いた GA で解いた。集団内の評価値の差が 10^{-7} 以下になったとき収束とみなし、集団サイズ 10, 20, 50, 200 について各 500 試行行い、それぞれ収束した解を数え上げた結果を表 4.3 に示す。

表 4.3: 12 次元 FP 関数を GA で解いたときの発見解; 試行数 500 回

解	評価値	集団サイズ			
		10	20	50	200
A	0	21	17	8	2
B	0	36	45	25	6
C	0	16	5	0	0
D	0	9	4	0	0
a	0.813	45	50	22	9
b	10.62	26	14	3	0
c	13.28	176	291	431	483
d	249.6	12	8	1	0
e	334.5	27	9	1	0
f	817.9	32	15	1	0
g	1170	65	31	6	0
h	1426	19	8	2	0
i	2790	11	2	0	0
j	3554	2	1	0	0
k	7064	3	0	0	0

集団サイズ 10, 20 の試行において、4 種ある最適解をすべて発見しているが、多くの試行で局所解にとらわれている。GA の性能に関する常識として、局所解に多くとらわれる場合は集団サイズを増やせばよいという考え方がある。しかし、集団サイズを 50, 200 と大きくするにしたがって、最適解の発見率は減少し、主に局所解 c に収束している。これ

は、c を含む谷が非常に有力な谷であり、UV 現象が強く生じているためと考えられる。

集団サイズが比較的小さいときは、c を含む谷に解が生成されず、結果としてこの谷の有力さにだまされることもなかった場合に最適解を発見できていると考えられる。最適解の中ではB が最も出やすく、以下 A, C, D と出にくくなり、c の他に a, g も有力な局所解であることがわかる。

UV 構造の解析

12 次元 FP 関数において UV 現象が生じている原因について、仮説の 3 つのクラスの観点から考察を加える。

表 4.4 は、最適解 A..D, 代表的な局所解 a..c の各解の谷のクラス 1, 2 に関する特徴をまとめたものである。まずクラス 1 に関して、各解の周り半径 r=0.01, 0.1, 1 の超球面上にランダムな点を 3 万点サンプルしたときの評価値の平均を示す。これは探索序盤での谷の有望さに関する情報を与える。さらにクラス 2 に関して、探索空間上にランダムに 2 万点の初期解を生成し、関数型を既知として微分情報を用いた最急降下法で局所探索した場合に各解に到達した回数を数え上げた。この回数が大きいほど、それぞれの谷の間口が広いと考えることができる。

表 4.4: 各解の、ランダムサンプリングによる平均評価値、および局所探索による到達回数

解	評価値	r=0.01	r=0.1	r=1	到達回数
A	0	6.3053	627.87	58124	481
B	0	5.6183	561.17	53690	586
C	0	6.4132	640.62	59092	608
D	0	6.9172	680.69	61341	568
a	0.813	6.7484	582.44	54606	2142
b	10.62	16.987	642.88	58340	1363
c	13.28	18.978	574.09	53697	4333

UV 構造クラス 1 についての検証

表 4.4 より、解の周辺の平均評価値をみると、半径 0.01 ほどにまで集団が小さくなれば最適解の周りの解はもはや局所解よりもよい質を持つようになるが、半径 0.1, 1 すなわち

まだ谷に解が生成された初期に近い段階では、解 B, a, c などが良い平均評価値を持ち、解 A, C, D, b 等は比較的悪い平均評価値を持っていることがわかる。有力な局所解 c, a のほか B は最適解の中でも比較的出やすい解であり、C, D, b は出にくい解である。クラス 1 がこれらの GA の挙動に大きな影響を与えていることがわかる。

UV 構造クラス 2 についての検証

到達回数について見ると、最適解 4 つはいずれも谷が小さく、局所解 a, c 等は比較的大きい。特に c は探索空間の約 22%を覆っており、クラス 2 によって c が最も有力な局所解になっていることが支持される。

UV 構造クラス 3 についての検証

二つの谷についての進化速度の違いを調べるために、各局所解の周り半径 r=0.5 の超球面上に 100 個体ずつの初期集団を生成し、集団間の相互作用なしに独立に進化させる。その結果、両者はそれぞれ谷の局所解に収束する。

UV 構造仮説クラス 3 が問題にしているのは、進化速度の異なる谷があった場合に、進化の遅い谷に存在する個体が淘汰されてしまうことである。実験は、各世代ごとに両方の谷からサンプルを 1 つずつ取ってきたときに、どちらの評価値が優れているかという確率の世代変化を調べた。図 4.8 は、谷 A の解が谷 C の解よりも優れている確率（左)、谷 c の解が谷 a の解よりも優れている確率（右）の世代変化を示す。

A, C はともに最適解であり、クラス 1 に関してはやや A が劣り、クラス 2 に関してはほぼ同等の構造を持っている。しかし、最初の世代で淘汰割合がほぼ互角であった両者は、進化が進むにつれ谷 A の集団のほうが谷 C の集団より優位になってゆく。これはクラス 3 の構造により A が C よりも出やすいことを説明している。

a, c はともに有力な局所解であり、a もかなり大きな間口を持っていてクラス 2 に関してもやや劣る程度である。しかし c に比較して a はかなり出にくい。図を見ると、やはり谷 a の進化速度が谷 c に比較してかなり劣ることがわかる。最後に c の方が負けているの

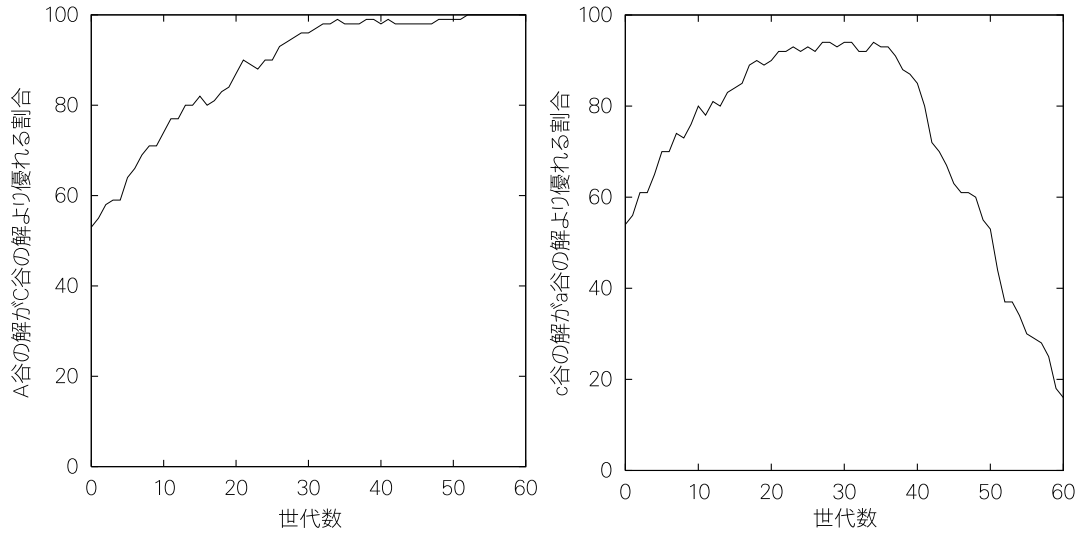


図 4.8: 一方の谷の解が他方より優れている確率

は、谷 a の個体が独立に進化を続けた結果、c の持つ評価値 13.28 を下回ったためである。

各谷の進化の速さの違いを見るために、先の実験と同様の初期集団を用い、その集団の上位半分（50 個体）の平均評価値を調べた。表 4.5 に、各解につき 50 試行を行った平均を示す。世代交代モデルに採用した DDA モデルは、まれに悪い解を比較的長期にわたり残してしまう傾向があるため、有意な検証を行うために上位半分の平均評価値を用いた。

表 4.5: 各解の周辺を初期集団とする GA の進化

解	評価値	世代 10	20	30	40
A	0	50.676	1.4382	0.0999	0.0262
B	0	199.73	8.0932	0.3671	0.0331
C	0	137.01	4.8792	0.2716	0.0426
D	0	571.75	59.097	4.7445	0.5500
a	0.813	681.21	159.14	45.558	13.960
b	10.62	57.871	11.865	10.728	10.646
c	13.28	182.64	21.679	13.766	13.348

この結果、A や b の進化が非常に速く、a の進化は非常に遅いことがわかる。またこれらは、先の実験で行った A と C の比較、a と c の比較の結果とも合致している。

まとめ

以上の解析結果を表 4.6 に要約する。

表 4.6: UV 構造と従来の GA による出やすさとの関係

解 (評価値)	周辺評価値	占有率	進化速度	発見数
A (0)	悪い	2.4 %	非常に速い	17
B (0)	良い	2.9 %	速い	45
C (0)	悪い	3.0 %	速い	5
D (0)	非常に悪い	2.8 %	遅い	4
a (0.813)	良い	10.7%	非常に遅い	50
b (10.62)	悪い	6.8 %	非常に速い	14
c (13.28)	良い	21.6%	速い	291

12 次元 FP 関数は UV 現象を起こす関数であり、それは集団サイズを増やしても防げずにむしろ悪化する。従来の GA を実行した際のそれぞれの解の発見頻度には大きな偏りがあり、それらは UV 構造の各クラスで説明される。

これまでの 9 谷関数および 12 次元 FP 関数に関する議論から、UV 現象が GA にとって致命的な性能低下をもたらすことがわかった。大域的多峰性を持つ問題の多くは必ずしも最適解を含む谷が最も GA にとって有力であるとは限らず、このような場合従来の世代交代モデルでは UV 現象が生じる可能性がある。大域的多峰性は困難な問題に共通の特徴であると考えられ、UV 現象を起こさずに探索を進める世代交代モデルを開発することが求められる。

4.3 接近法：生得分離モデル ISM

UV 構造仮説によれば、UV 現象は最適解を含む谷を探索する個体の評価値が、遠方有力局所解を含む谷を探索する個体の評価値よりも劣るために、その両者が比較されて最適解側の個体が淘汰されることにより起こる。最も原始的な GA(Simple GA, SGA) では集団全体から評価値の劣るものを淘汰してゆくため必然的に UV 現象が生じる。また、本論文の実験で採用した CCM, DDA のように多様性維持の点で優秀であるといわれる家族選択モデルであっても、両親が別の谷から選択されれば結果として生き残る二つの個体がどちらかに偏ることはありうる。困難なクラスの問題では、一つの有望な領域だけを探すのでは不十分であり、いくつもの領域を同時に探索する必要があると考えられる。

4.3.1 従来手法の考察

困難な問題を解くためには、大谷構造を効率良く探索すると同時に、UV 構造に騙されずに探索を進めることが必要である。本節では通常の GA の他に島モデル GA[Mühlenbein 91] [Smith 93] および並列局所探索を取りあげ、それらの特徴と性能に関する考察を行う。島モデルの他にも Split Ring モデル [Harvey 99] など局所的な選択・交叉を行う手法はあるが本論文では取り扱わない。また、明示的に集団の多様性を一定以上に維持する手法としてシェアリング手法 [Goldberg 87] や熱力学的遺伝アルゴリズム [森 96] が提案されているが、前者は距離に大きく依存する手法のため悪スケール（ill-scaled)[喜多 98] な問題では性能を発揮できず、後者は計算量の大きさが問題となっている。

通常の GA では、探索は一つの大きな解集団により行われる。各解の初期化は領域全体で一様に行われ、交叉の親選択および解淘汰も集団全体で行われる。大谷構造下では通常の GA は理想的な挙動を示すことが知られているが、UV 構造下では UV 現象を引き起こし、最適解を発見できなくなる [Ikeda 2000]。

島モデル GA は元来、交叉や淘汰を制限することで解の多様性を維持し、局所解で初期収束を起こすことを防ぐために開発された手法である。島モデル GA では探索は複数の小集団により行われる。各小集団の初期化は領域全体で一様に行われるが、交叉の親選択および解淘汰は小集団ごと独立に行われる。また、多くの島モデル GA では各小集団同士で移民を行うことで解の質の向上を図っている。

島モデル GA では、各小集団のみを見た場合は個体数が小さくなるために大谷構造下での性能はやや劣る場合もある。一方大域的多峰性の問題では、各谷をそれぞれ探索することが期待できるため、UV 構造がきつくなければ通常の GA よりも最適解を発見できる可能性は高まる。しかし初期化を領域全体で一様に行うためにすべての小集団が最初から最適解の谷をはずしてしまうことも考えられ、UV 構造への対応としては不十分である。クラスタリングにより探索の局所化を図る手法 [Jelasity 98], [高橋 2001] にも同様のことが言える。また多くの島モデルでは移民を用いており、最適解の谷を探索している集団が別の谷の見かけ上の良さに騙されてしまうこともありうる。より明示的に探索空間を分離するという意味での研究としては [Tsutsui 97] などがある。

解同士が影響しあう問題を回避するために、交叉により情報を交換することなく、各解が独立に局所探索を行うことが考えられる。ここではこれを並列局所探索と呼ぶ。並列局所探索では、各解は最初からいずれかの谷に含まれているため、UV 構造が存在したとしても他の谷の良さに騙されることなく探索を進めることができる。しかし各谷内の局所解に陥る確率は GA を用いた場合よりも高く、それは局所探索の代わりに SA 等を用いても同じである。

4.3.2 ISM の提案

島モデルの欠点は初期化を領域全体で一様に行うことおよび移民による騙されにある。そこで並列局所探索のように最初から各小集団が一つの谷に入るように初期化して、その後は小集団ごと独立に探索を進めることにより、GA の持つ強力な探索能力を維持したまま UV 現象を回避する戦略が考えられる。これを最初から分離されているという意味で生得分離モデル (Innately Split Model : ISM) と呼ぶ。

生得分離モデル ISM

1. 探索は複数の小集団で行う。
2. 各小集団は、小さい領域で初期化する。
3. 交叉や淘汰は小集団ごと独立に行う。
4. 同じ谷を探索している複数の小集団があった場合、一方を排除して、再初期化する。
5. 悪い評価値のまま長く停滞している小集団は、排除して再初期化する。

ここで、4 および 5 は探索効率化のためのオプションである。

ISM は複数の小集団を各々小さい領域で初期化して、独立に探索を進める手法である。従来手法と提案手法 ISM の特徴を表 4.7 にまとめる。

ISM を実装するためには、従来の GA が必要とする交叉等の設定の他に小集団の数および各小集団の個体数、各小集団の探索打ちきり条件、初期化領域の設定が必要であるが、これら設定の指針とその詳細は付録 B1 に記す。

表 4.7: 従来手法と提案手法 ISM の比較

探索の枠組みと性能		通常の GA	島モデル GA	並列局所探索	ISM
枠組み	探索主体 各集団の初期化 解の生成と更新	1 つの大集団 領域全体で一様 集団全体	複数の小集団 領域全体で一様 小集団ごと+移民	複数の個 領域内の 1 点 個ごと	複数の小集団 小さい領域で一様 小集団ごと
性能	大谷構造への対応	◎	○	△	○
	UV 構造への対応	×	△	○	○
	両者への対応	×	△	△	○

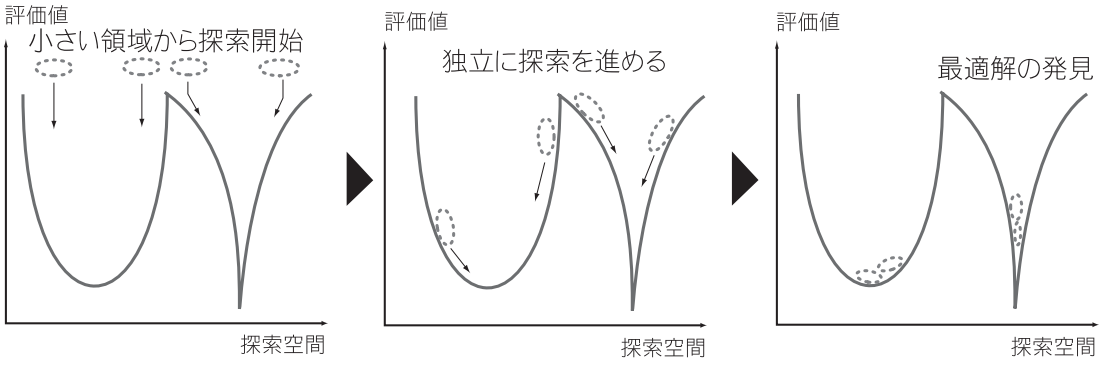


図 4.9: Innately Split Model の探索イメージ

4.3.3 ISM の特徴

図 4.9 に ISM の探索イメージを示す。本節では、ISM の特徴について考察する。

- **UV 現象の回避**

  もし、最適解を含む谷が探索空間全体に占める割合が十分大きく、また各谷が単独で谷底を発見できる程度に簡単であるならば、ISM は UV 構造が存在して従来の GA に UV 現象が生じる問題を非常によく解く。これは、有望な領域と一見有望でない領域を比較することなく探索を進めることで UV 構造に騙されないためである。

我々は解くべき問題として大域的多峰性の問題を想定しているが、ISM が最適解を発見するためには最適解を含む谷の内部またはそれにまたがって ISM の小集団が生成されなければならない。例えば Fletcher and Powell 関数 12 次元問題では、全体の 2.5 から 3%ほどを占める谷が各 4 つの最適解の周辺に形成されており、数十の集団を生成することでいくつかは最適解に到達できると考えられる。

他に考えられるISMの有効な点として以下が挙げられる。

- **終盤での効率的な探索**

  ISMは近い解同士だけで交叉を行う。一般に進化の中盤以降は、遠すぎる解同士の交叉は良い子供を生成しない。従来のGAが十分多様性を確保していたとしても、大域的多峰性問題を解く際にすべての領域から両親を選び交叉を行うことは無駄な探索につながる。

- **ニッチングへの応用**

  評価関数が動的であったり信頼できない場合に複数の局所解に対する実際的な要請があり、さまざまなニッチング手法が提案されている[Watson 99]。ISMは評価値の悪い局所解をも保持することができるため、従来提案されているGAベースのニッチング手法よりも多数のしかも精度の高い局所解を保持できる。

- **並列計算との親和性**

  通常のGAを並列計算機を用いて実装する場合、解情報などの通信を頻繁に行わねばならないことは実行速度に大きな悪影響を及ぼす。一方ISMは複数の小集団を独立に進化させる。相互作用を考えねばならないのは手順のごく一部であり、並列計算機への実装は容易でかつ実用的である。

ISMは、従来のGAがうまく解けない問題を効率よく解くと思われるが、一方従来のGAが解ける問題をうまく解けないケースも考えられる。

- **大域的多峰性が存在しない場合**

  従来のGAが探索空間全体の情報を利用して探索を進めるのに対し、ISMは各小集団はその周辺を徐々に探索していく。そのため、一つの大谷を解くのであれば従来のGAの方が効率よく最適解を発見できるであろう。

- **最適解を含む大谷が非常に小さい場合**

  最適解を含む谷が全体の0.001%しか占めていないようなケースでは、初期集団が最適解を含む谷の内部に生成される可能性は期待できない。このような場合、十分広

い初期化領域が最適解を含む谷を含み、その中でうまく最適解を発見してくれることを期待するしかなくなる。

- **一つ一つの谷が難しい場合**

  ISM は、大谷の内部で谷底の解を発見すること自体は可能であることを前提に設計されている。もし大谷の一部分に初期化された小集団が谷底を発見できないほどに難しいならば、ISM では良好な解を得られない。このような場合、全体を探索できる分従来の GA の方が効率良く質の高い解を発見できることもありうる。

- **オンラインパフォーマンスが重要となる場合**

  通常の GA が探索を有望な領域に急激に絞り込むのに対して、ISM は劣る領域も候補として保持しながら探索を進める。よって、探索の序盤中盤においては従来の GA のほうが平均的には良い解を保持することが考えられる。ISM は最終的にできるだけ良い解を得ることを目的に開発されたものであり、オンラインパフォーマンスが特に重要となる場合は適用に際して何らかの工夫が必要になると思われる。

以上の議論から、ISM は大域的多峰性を持つ多くの問題に対して非常に有望な手法であることが期待できる。また、これらの議論を通して、ISM の適用が好ましい問題領域とそうでない領域が明らかになった。

4.4 ジョブショップスケジューリング問題への適用

JSP は複数の独立な仕事を行うために、仕事を処理する機械の時間的な割り当てを決定する問題である。各仕事は定められた順序で機械に処理されねばならず、また一つの機械は同時に二つ以上の仕事を処理することはできない。JSP を最適化するとは、すべての仕事が完成するまでの総作業時間 (makespan) を最小化するように各機械上での仕事の処理順序を決定することである。

JSP は NP-困難な組合せ最適化問題の一つであり、ごく小さな規模の問題であってもその最適解を発見することは非常に困難である。分枝限定法（Branch and Bound, BAB）による厳密解法は活発に研究されてきた [Brucker 94] が、1984 年に提案されたたかだか

20 仕事 10 機械の問題の最適解がこの方法により発見されたのは 1996 年のことである。一方、GA をはじめとする確率的最適化手法による近似解法も多数提案されており、厳密解法に比べ現実的な時間で良好な解を得ている。なかでも、タブサーチ (TS) を用いたもの [Nowicki 93]、シミュレーテッドアニーリング (SA) を用いたもの [Yamada 95,96] [Aarts 94]、GA を用いたもの [Yamada 96,97] [Sakuma 2000] は特に良好な結果を残している。

小野らの GA による JSP への解法 [小野 98] に着目すると、大きな集団サイズを用いているにもかかわらず困難な問題においては最適解を発見できていない。さらに、いくつかの問題では、全 10 試行において最終的に得られた解がすべて同じ局所解であるという極めて特徴的な現象がみられる。これはまさに、JSP において我々が主張する UV 現象が起こっていることの証左と考えられる。

本節では、JSP の困難さが大域的多峰性と UV 現象にあると仮定し、実験によりそれを確認する。我々の主張する UV 構造仮説によれば、UV 現象は 3 つの UV 構造のいずれかまたは複数が景観に存在することにより生じる。JSP における UV 現象と UV 構造の因果関係が確かめられれば、提案手法 ISM を JSP に適用することで GA の性能向上が期待できる。

4.4.1 UV 構造仮説に基づく JSP の解析

大域的多峰性

本節では、JSP が大域的多峰性をもつことを確かめる。図 4.10 は比較的易しい代表的インスタンス ft10 の最適解からの距離と評価値の差の関係をプロットしたものである。いくつもの局所解のまわりに谷が形成されており、また最適解から十分遠い場所に有力な局所解が存在することがわかる。距離の尺度としては、個体の仕事列要素の不一致率および I_2 距離 [Sakuma 2000](付録 B2 参照) を用いた。

ft10 における最適解 930 とある有力な局所解 938 の I_2 距離は 170、不一致率は 74 %である。これは、ランダム個体間の平均 I_2 距離 231、平均不一致率 83 %に比べ十分大きい。つまり、JSP は大域的に多峰な問題だといえる。

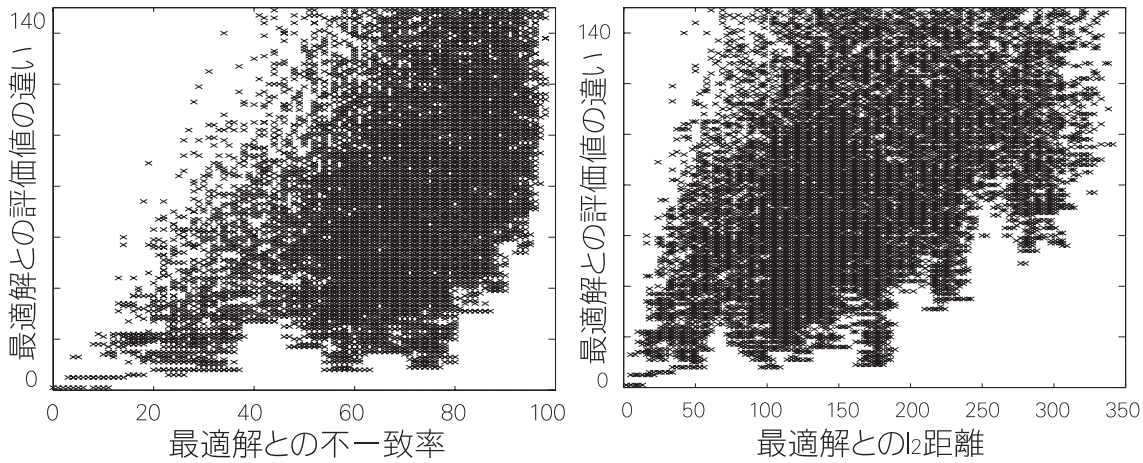


図 4.10: ft10 の評価値と不一致率 (左), I_2 距離 (右) の関係：ISM を十分な時間適用し、評価した全ての解のうち最適解からの評価値のずれが 150 以下のものをプロット

UV 現象

本節では、上記 ft10 問題と、従来の GA にとって困難である abz5 問題を対象に、UV 現象が生じるかどうかを調べ、また UV 構造との関係を考察する。

実数値最適化問題と異なり、組み合わせ問題では「最適解に向かう谷に含まれる集団」を定義するのは困難である。そこで、どちらの問題にも有力な局所解が一つずつ存在することを利用する。ここでは最適解 (又はある有力な局所解) に向かう集団を、パラメータ d を用いて「最適解（又は局所解）との距離を d_1、局所解（又は最適解）との距離を d_2 としたとき $d_1 + d \leq d_2$ 」となるランダム個体の集団で定義する。図 4.11 には、d=50 のときに最適解に向かう集団を表す領域を示す。例えば白丸で表されている解は最適解から 60、局所解から 100 離れており、最適解に近くはあるがこの場合 d=40 となるため指定の領域には含まれない。

表 4.8 は、各集団を初期集団として独立に各 100 回 GA を行った場合に、期待される解（最適解に向かう集団であれば最適解、ある局所解に向かう集団であればその局所解）に達した割合を示す。さらに、両者を混ぜて進化させた場合の最適解の発見回数を示す。集団サイズは 30、生成子個体数は 20、打ち切り世代数は 200 とし、世代交代モデルは CCM[Ono 98] を用いた。

JSP は各谷を探索するだけでも困難な問題であるが、d=30, 50 としたときに、最適解

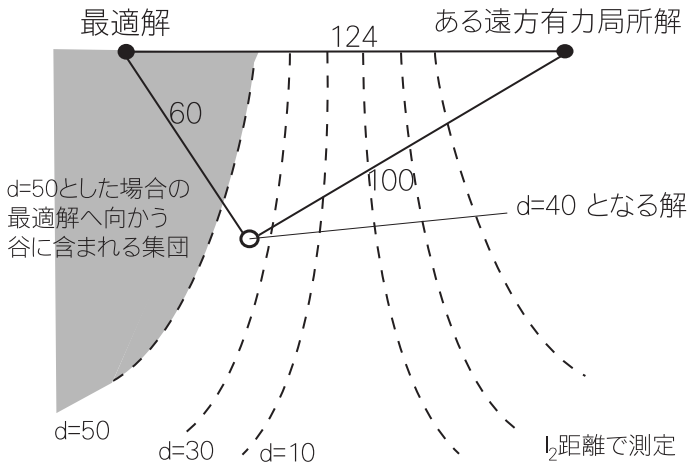


図 4.11: 谷に含まれる集団の選択：ランダムに生成した解について最適解との距離が 60、遠方有力局所解との距離が 100 だった場合、パラメータ d=50 ならば「最適解に向かう谷に含まれる解」とはされない

表 4.8: 特殊な初期解群による収束実験：最適解側に偏らせて生成された初期集団を用いて GA を実行した場合の最適解発見回数（/100 試行）、ある局所解側に偏らせた場合のその局所解発見回数、両者を混ぜて GA を実行した場合の最適解発見回数

d	最適解側から	局所解側から	混ぜた場合
10(abz5)	21/100	72/100	8 /100
30(abz5)	43/100	74/100	15/100
50(abz5)	80/100	81/100	33/100
30(ft10)	64/100	78/100	53/100

側に生成した集団は最適解を半分以上の試行で発見し、局所解側に生成した集団はほぼ局所解を発見している。これから、本実験における「最適解に向かう谷に含まれる集団」の生成法には妥当性があると言える。

さて、ft10 問題では、最適解側に作った集団は 64 回最適解を発見している。局所解側に作った集団と混ぜることでも、最適解の発見回数は 53 回にしか低下しない。これは、局所解を含む谷が ft10 問題ではあまり有力でないことを示している。一方 abz5 問題では、単独では最適解を発見できていても、局所解側に作った集団と混ぜることでその最適解発見率は激減する。これは、abz5 問題では UV 現象が生じており、またそれゆえに困難な問題になっているということを示している。

UV 構造

ft10 問題と abz5 問題を UV 構造の観点から比較する。表 4.9 は、前節の実験で各々100 試行 ×30 個体を生成したときに、各領域に含まれる解の質の平均とランダム個体が各領域に生成される確率、さらに各集団が目的とする解を発見した平均世代数を示したものである。

表 4.9: 前節で生成した初期解群の UV 構造に関する特徴

距離 d (問題)	クラス 1：探索序盤の有望さ 初期解の質（平均 ± 標準偏差）		クラス 2：探索個体数の多寡 谷への解の生成率（%）		クラス 3：進化の速度 目的解発見世代数（平均）	
	最適解側	局所解側	最適解側	局所解側	最適解側	局所解側
10(abz5)	1532 ± 94.0	**1501** ± 87.7	19.1	**59.8**	89	**67**
30(abz5)	1535 ± 92.1	**1490** ± 80.2	9.0	**37.7**	66	**44**
50(abz5)	1520 ± 94.5	**1486** ± 82.0	2.9	**17.4**	54	**40**
30(ft10)	1197 ± 59.6	1187 ± 61.4	**28.1**	22.5	78	**63**

- **クラス 1：探索序盤の有望さについて**

  表 4.9 の初期解の質の欄にあるとおり、ft10 問題では最適解側/局所解側にさほど差がないが、abz5 問題では局所解側の平均評価値のほうが優れている。これは探索序盤の有望さによる UV 現象が abz5 問題で生じていることを強く示唆する。

- **クラス 2：探索個体数の多寡について**

  各領域への生成率を比べると、ft10 問題では最適解側に多く解が生成されるのに対し、abz5 問題では局所解側に多く解が生成される。またこれとは別に、最適解と局所解がどういう場所に位置しているかを知るために、各々が I_2 距離でランダム解とどれだけ離れているかを調べた。その結果、最適解とは平均で 145、有力局所解とは平均で 129 離れていた。これらは、探索空間内で最適解がこの局所解よりも端方面に位置していることを示しており、クラス 2 による UV 現象が abz5 問題で生じていることを強く示唆する。一方 ft10 問題では最適解側のほうが中心方面にあり、これにより最適解が発見されやすくなっていると考えられる。

- **クラス 3：進化の速度について**

表に示すとおり、abz5 問題では最適解を発見するのに要した世代数が局所解を発見するのに要した世代数より大きくなっている。これは最適解を含む谷の周辺に解が生成されたとしても、進化を進め最適解を発見するのにはより長い時間がかかっていることを示している。すなわち、進化のスピードによる UV 現象が abz5 問題に生じていることが示唆される。

まとめ

abz5 問題は UV 現象を起こし、またそれゆえに難しい。UV 現象の原因として UV 構造を調べた結果、クラス 1, 2, 3 が確認された。一方 UV 現象を生じない ft10 問題では、むしろクラス 2 において最適解を発見しやすい構造が確認された。これらから、JSP において UV 現象が生じるかどうかと、難しさおよび UV 構造が存在するかということの間に明確な因果関係が存在することが明らかになった。

4.4.2 ISM の JSP への適用

これまでの議論から JSP の困難なインスタンスには UV 構造が存在し、UV 現象が起こることで GA が最適解の発見に失敗していることがわかった。提案手法 ISM は UV 現象を回避することができる新しい GA のモデルであり、JSP はその適用の格好の対象と考えられる。本節では、提案手法 ISM を用いた GA を JSP の代表的ベンチマークである 10 難問 (10 tough problems, [Applegate 91]) に適用し、従来の GA 及び他の確率的最適化手法と比較する。

設定

JSP の実験で用いた設定を述べる。世代交代を CCM で行っているほかは、4.4 節の実験も同様である。

- 修正操作として GT 法 [Giffler 60] と、より強いバイアスをかけた GTb 法 (付録 B3 参照) を用いた。さらに、LR 法 [Yamada 96,97] を用いることで、評価値の改善確率の向上と、バイアスの打ち消しを図った。

- 突然変異は、交叉の際に同じ評価値のものが選ばれた場合のみ交叉の代わりに用いた。用いたオペレータは Job-order based mutation (shift change, swap)[Ono 98] である。
- 交叉は JOX[Ono 96] を用いた。さらに優秀な交叉として MSXF[Yamada 96,97], EDX[Sakuma 2000] が提案されているが、これらは JSP 固有の CB 近傍による SA 的手法を用いている。本論文の興味の対象は交叉ではなく世代交代モデルであるから、シンプルな交叉として JOX を選択した。
- ISM の集団数は 40、一つの集団の個体数は 30 とした。
- 集団内の世代交代モデルは CCM を採用した。
- 初期化領域はある解から一回突然変異を行った範囲とした。
- 2000 世代にわたって集団内の最良評価値が変化しなかったとき、その集団はある解に収束したと判断した。

これらの設定のもと、打ち切り時間 20 時間で実験を行った。計算環境は WindowsMe (CPU 1GHz)、使用言語は Delphi4 (Object Pascal) である。ISM は本質的に多数の試行を内包しているため、一試行あたりの計算を長く行うのが好ましい。総時間としては、現在最良の手法と思われる YN96(SA)[Yamada 95,96] が 3 時間 $\times$20 試行を行っているのでこれとの比較は妥当であろう。なお、各小集団は 2000 世代にわたって最良評価値が変化しなかったときに収束とみなされその小集団のみが新しく再初期化される。この初期化は 20 時間の制約のもとでは、*la21* 等小さい問題では 200 回ほど、*abz7* 等大きい問題では 70 回ほど行われる。

結果

表 4.10 に、得られた最良解と、最適解を得た場合はその回数、および同等の条件で世代交代モデルを CCM にしたもの、同じ交叉 JOX, 世代交代モデル CCM を用いた [小野 98] での結果を示す。[小野 98] では集団サイズ 1200 で 10 試行しているが、これでは時間がかかりすぎると思われるので比較のための CCM では集団サイズ 200 とした。また、このモ

デルでは一回集団が収束するとただちに新しい初期集団を生成し、GA を繰り返す。一回の収束に *la21* 等規模の小さい問題では 30 分程、*abz7* 等大きい問題では 80 分程を要するので、20 時間の制限のもとではそれぞれ 40 回、15 回ほどの試行を行っていることになる。

さらに、集団数 40 および各集団の個体数 30、オペレータ等を変化させずに、島モデルを行い比較した。島モデルでは各集団は探索空間全体にランダムに生成され、50 世代に 1 回ランダムな個体を小集団同士で移民させることにする。*opt/UB の欄に、分枝限定法により確認された最適解もしくは既知最良解 (Upper Bound, UB) を示す。ISM は la21,24,25,27,38 で最適解を発見しており、世代交代モデルだけが異なり他は同じ設定の CCM よりも一意に良い結果である。また [小野 98] を CCM が上回っていることから他の設定の妥当性も示された。また、島モデルは CCM よりは概ね性能が向上し、UV 現象を多少なり抑制していると思われるが、提案手法には及ばない。

表 4.10: 提案手法による 10Tough 問題への適用結果

inst.	*opt/UB	Best(opts)	CCM	小野 98	島モデル
abz7	*656	664	671	680	668
abz8	665	670	671	685	674
abz9	679	682	688	702	688
la21	*1046	***1046**(9)	1052	1050	1052
la24	*935	***935**(21)	938	944	939
la25	*977	***977**(12)	984	984	*977
la27	*1235	***1235**(4)	1249	1258	1240
la29	*1152	1157	1167	1189	1162
la38	*1196	***1196**(5)	1208	1202	1202
la40	*1222	1224	1228	1235	1228

さらに, 表 4.11 で高性能な近似解法との比較を行う。Nowi は [Nowicki 93] において提案されたタブサーチによる解法である。Aarts は [Aarts 94] において提案された SA による解法である。Appl は [Applegate 91] において提案された部分的分枝限定法 (シャッフルアルゴリズム) による解法である。YN96 は [Yamada 95,96] において提案された適応的 SA による解法であり、従来最も優れた解法であった。YN97 は [Yamada 96,97] における応用範囲の広い交叉 MSXF を用いた GA による解法である。

表 4.11 より、他手法に対する ISM による GA のほぼ完全な優位性が示されたといえる。

表 4.11: 高性能な近似解法との比較

inst.	ISM	Nowi	Aarts	Appl	YN96	YN97
abz7	**664**	-	668	668	665	678
abz8	**670**	-	670	687	675	686
abz9	**682**	-	691	707	686	697
la21	***1046**	1047	1053	1053	*1046	*1046
la24	***935**	939	*935	*935	*935	*935
la25	***977**	*977	983	*977	*977	*977
la27	***1235**	1236	1249	1269	*1235	*1235
la29	1157	1160	1185	1195	**1154**	1166
la38	***1196**	*1196	1208	1209	1198	*1196
la40	1224	1229	1225	***1222**	1228	1224

なおISMでは、さらに時間をかけることでabz7で658, abz8で669, abz9で679(UB), la29で1153と非常に良好な解を得ている。

4.4.3 実験のまとめ

我々は、JSPの探索においてUV現象があることで従来のGAが最適解発見に失敗していることを確認した上で、提案手法ISMをJSPの代表的インスタンス10難問に適用することにより、仮説の妥当性と提案手法の有用性を示した。

まず、提案手法は従来多様性維持に有効とされていた世代交代モデルCCMと比較した。この結果、計算時間や使用オペレータは全く同じであるにもかかわらず、ISMは一意に良い性能を示した。これは、CCMでは最も良く見える谷だけを探索してしまい、仮にその谷を完全に探索していても局所解しか発見できなかったのに対し、ISMは探索空間中の谷の多くを独立に探索することでより良い解を得ることができたことを示している。

また、島モデルを用いることで、CCMよりは概ね良い結果が出たが、これも提案手法には及ばなかった。島モデルは移民を用いることで良い解への収束を図るが、これにより相対的に評価値の悪い最適解の谷を探索する解群が淘汰されてしまっていると考えられる。また移民を用いなかったとしても、初期に広い領域を探索するために有力局所解の谷への偏りが生じてしまうであろう。

本論文ではCCMおよび島モデルとの比較しか行っていないが、多様性を全く考慮しな

いSimpleGAはもとより、ニッチングを用いることでもUV現象は完全には回避されない。ニッチング手法は探索空間全体をくまなく探索するには適しているが、探索を集中させないために谷の中のごく良い解を発見するのには向かない。JSPの10難問は一つの谷を探索するのさえ非常に困難であり、これら古典的な手法で良好な性能を示すことは不可能と思われる。

UV構造のもとでSAを用いる場合、初期温度を高くしすぎると探索空間全体を動いて良さそうな場所に落ち着いていくSAの性質から、有力な局所解の谷につかまり最適解が発見できないということも予想される。[Yamada 95,96]のように温度に関する十分な工夫を行えばこの現象は回避されると考えられるが、SAに関する一知見として我々の研究が役立つと期待される。

ISMは、探索空間に離れて存在する谷をくまなく探索しながら、かつ各谷はGAの強力な探索性能を用いて探索することができる手法であり、非常に困難な問題に対して有効な選択であるということがこれらの実験から示された。問題をJSPに限れば、さらに[Applegate 91]で用いられているシャッフルアルゴリズムを局所探索法として補助的に用いたり、[Yamada 96,97]の交叉MSXF、[Sakuma 2000]の交叉EDX等を用いることでさらなる性能向上が期待できる。

4.5 まとめ

今日までGAは、個体を探索空間全体に生成し、最適解を含む形で探索を重点化していくのが理想的であるとされてきた。実際、Rastrigin関数のように微視的には多峰であっても、全体としては1つの大きな谷と考えることができるような大谷構造をもった景観であれば、従来のGAはその期待通りの挙動を示す。しかしながらこれは、有望そうな領域が確かに有望であるような場合にしか期待できないことである。実際、Rastrigin関数と並び古くから用いられる実数値最適化問題であるRosenbrock関数は、単峰であるにもかかわらず、GAを用いると一旦集団全体が最適解から離れた原点付近に集まるという性質がある。Rosenbrock関数の場合は原点と最適解が緩い坂で繋がっているために探索を進めることで最適解を発見できるが、もしこれらが別の谷の底であるなら、一旦小さく収束

した解集団は決して最適解まで辿りつくことはできないであろう。

Rastrigin 関数、Rosenbrock 関数などの人工的なベンチマーク問題と異なり、JSP や FP 関数のような実問題から派生した問題には多くの有力な局所解が存在している。従来の GA がそれらを一斉に探索しようとすると、探索の重点化が最適解を含む谷に対して行われるという保証はない。それどころか、UV 構造が存在してある強力な局所解を含む谷のほうが有望に見える場合には、逆にその谷への探索の重点化がより高い確率で起こってしまうことになる。GA にとっての困難さが UV 構造による場合があることは、FP 関数と JSP の解析によって確かめられた。

数個、数十個の谷が存在する場合、最適解を含む谷が最も有望に見えるということを仮定するのはむしろ不自然であり、困難な問題では各谷を独立に探索する必要がある。本論文で我々は、生得分離モデル ISM を大域的多峰性問題を解くための新しい GA のモデルとして提案した。ISM は、集団を複数の小集団に分けて、各小集団が小さい領域を独立に探索することにより、UV 構造のもとでも UV 現象を回避することができる。

ISM の優秀さは JSP への適用により示された。他手法との比較ばかりでなく、GA のモデルのみを従来手法 CCM および島モデルにした GA との比較によりそれは一層明らかである。また、本論文には詳しく述べられていないが、FP 関数への適用によっても ISM が UV 現象を回避して容易に最適解を得ていることを付す。さらに、JSP と同様大域的多峰性を持つと思われる二次割り当て問題でも ISM は従来の GA を上回る結果を出している。一方 TSP や Rastrigin 関数等、大谷構造を持つ問題では ISM は従来の GA よりも低い性能しか発揮できず、適用可能クラスを考慮した選択が必要となる。

これらの結果は、ISM が多様な景観構造を持つ実問題への GA の適用可能性を大幅に拡げるという主張を強く支持するものであると言える。ISM の設計指針、パラメータの設定法については付録 B1 に付す。

第5章　α-domination戦略に基づく多目的最適化

本章では、Pareto最適解をまんべんなく求めるという従来の多目的最適化の目標を柔軟化し、ユーザが真に必要とする有益な解だけを求めるための最適化戦略を提案することを目標とする。

5.1　はじめに

近年多目的最適化問題への関心は高まるばかりであり、さまざまな分野の研究者がこれに取り組んでいる。多目的最適化においては唯一の最適評価値というものが存在しないために、何らかの合理性を満たした複数の解を提示することがアルゴリズムには求められる。そのために、複数の解を保持することのできるGAは多目的最適化と親和性が高く、PESA [Corne *et al.* 2000], NSGA-II [Deb *et al.* 2000] といった優秀なアルゴリズムがテスト関数において他を上回る性能をあげている。多目的最適化のための進化計算アルゴリズムをMOEAs(Multi Objective Evolutionary Algorithms)と総称する。

これらのMOEAsでは他の解にdominateされない解、すなわち現集団内のPareto最適解は良い解として次世代に保存され、もし集団全体が解で溢れてしまえば最も混みあっている部分から排除される。これらは一見合理的に思われるが、もしも明らかに質が悪いにもかかわらずdominateされにくい解が集団に多数存在してしまうとすると、これらPareto生存戦略を用いた手法はより質の高い解を探索できなくなってしまうという問題点がある。

さらに、現実の問題では、Pareto最適解をまんべんなく求めるという目標自体に意味がない場合が多い。例えば、自動車のエンジンを開発する際に、燃費と最高速度という二つの目的関数が与えられたとする。この二つの値の間には明示的なトレードオフ比が存在

しないため、A(10km/l, 200km/h) というエンジンと B(20km/l, 150km/h) というエンジンには優劣は定義できず、どちらも解候補として意味を持つであろう。一方、A'(1km/l, 210km/h) や B'(25km/l, 50km/h) といったエンジンがユーザにとって有益かとなると非常に疑わしい。しかしながら、通常の dominance の条件に照らせば、A' も B' も A や B に劣っているとは言えず、従来の MOEAs は計算時間やメモリの多くを無益な解を求めるために費やすことになる。

すなわち、従来の多目的最適化には、1）質が悪いにもかかわらず dominate されにくい解を効率的に排除できない、2）ユーザにとって無益な解を提示してしまう、という問題点がある。本論文ではこのような現象が生じる単純な例を挙げ、いかにこれに対処すべきかを述べる。

5.2 従来法と問題点

5.2.1 Pareto 生存戦略

Pareto 生存戦略を用いた MOEAs の主たるアイデアは、dominate されない解、すなわち現集団内の Pareto 最適解を優先的に次世代に残すということである。このため、domination や Pareto 最適の概念を用いない手法と比べた場合、Pareto 最適解を求めるという観点に立てば Pareto 生存戦略を用いるほうが好ましいと思われる。本節ではまず手法を具体化した上でこれを確認する。

Pareto 生存戦略の手順

本論文では、実験にあたり PbEA（Pareto-based Evolutionary Algorithm）と呼ぶ Pareto 生存戦略を用いた単純なアルゴリズムを用いる。PbEA は PESA を単純化したものであり、少ないパラメータ数で制御できるため Pareto 生存戦略の特徴を考察するのに適している。PbEA の具体的な手順を示す。なお、多目的最小化問題において解 x が解 y を dominate するとは、$\forall i\ f_i(x) \leq f_i(y)$ かつ $\exists i\ f_i(x) < f_i(y)$ となることである。

PbEA の手順

1. N 個体の初期解をランダムに生成する。
2. 集団の中から親個体がランダムに選ばれ、交叉又は突然変異により 1 つの子個体が生成される。
3. 子個体が集団の中のある個体を dominate するなら、その個体と子個体を交代し Step 2 に戻る。
4. 子個体が集団の中のある個体に dominate されるなら、Step 2 に戻る。
5. 集団内である個体が別の個体に dominate されるなら、子個体を交代し Step 2 に戻る。
6. N+1 個体が Pareto 解として存在するため、最も混み合っている場所の個体を排除する。停止条件を満たすまで Step 2 から繰り返す。

一方 domination の観念を用いない手法も様々開発されており、ここでは [Laumanns *et al.* 99] による Predator-Prey モデル (PPM) をとりあげる。PPM では、N 匹の獲物が解を表し、M 匹の捕食者によって淘汰されていく。各捕食者は各々目的関数の 1 つのみによって獲物を選択する。すなわち、各捕食者の視界の中で最もその目的関数が劣るものが排除される。よって、PPM による多目的最適化においては、Pareto 解を明示的に保存するということはなされない。

PPM との比較実験

PbEA と PPM の挙動を確認するために、[Laumanns *et al.* 99] で用いられているのと同じ設定および対象問題を用いて比較を行う。この論文では以下の 2 変数 2 目的のテスト関数を用いている。

Test-1 : $\text{minimize}_{x,y} \quad g_1, g_2$

$$g_1(x,y) = -10\exp(-0.2\sqrt{x^2+y^2}\,)$$
$$g_2(x,y) = |x|^{4/5} + |y|^{4/5} + 5\,(\sin^3 x + \sin^3 y)$$

$$-50 \le x \le 50, \qquad -50 \le y \le 50.$$

対等の比較を行うために、PbEA では PPM と同じ突然変異を用い、交叉は用いない。また、集団サイズは PPM よりも小さい 100 とした。PPM には突然変異時のステップサイズを減衰させるタイプもありこれも同時に調べた。

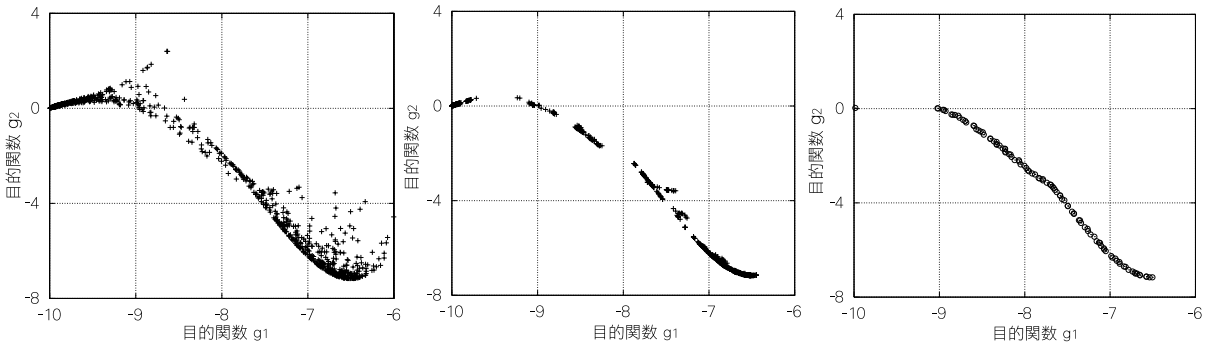


図 5.1: Test-1 における (a)stepsize 固定の PPM, (b)stepsize を減衰させる PPM, (c)PbEA の挙動：目的関数空間で表示、すなわち左下にあるほど良い

図 5.1 に各 MOEAs で得られた解の様子を示す。ステップサイズ固定の PPM(図 5.1(a)) は、良い解を多く持つものの、非 Pareto な解も多く残してしまっており、これは探索をこれ以上続けても同じである。 一方ステップサイズを減衰させる PPM(図 5.1(b)) では、より Pareto 最適に近い解が得られているものの、いくつもの部分にちぎれてしまっていることがわかる。これは PPM が Pareto 解を安定して保つことができないためであり、世代を重ねれば最後にはいくつかの点になってしまう。

一方 PbEA(図 5.1(c)) では、1) 非 Pareto な解は殆どない 2)Pareto 面を安定に維持できる、という利点が見て取れる。この例でもわかるように、Pareto 生存戦略を用いた MOEA は Pareto 解を得るという目的に関しては、良い性能を示す。

5.2.2 Dominance 抵抗解

多目的最適化の研究者はしばしば、解とは 1）他の解によって遠からず dominate される解、 2）なかなか dominate されない Pareto 最適に近い解 の 2 つに分けられる、と考えがちである。dominate されにくい解が良い解であると考えるからこそ、Pareto 生存戦略を用いた MOEAs を用いるわけである。

しかしながら、Pareto 解を求めるという目的そのものがユーザにとって無益な解を提示してしまうということは前述のとおりである。

また、喜多らが指摘したように [Kita *et al.* 1996]、ある種の問題ではいくつかの解は明らかに質が劣るにもかかわらず、長い世代淘汰されずに生き残ってしまう。それは、その

ような解を dominate する解が見つかりにくい構造になっているからである。本節ではまずこのような解に着目し、これらが生じる原因を明らかにする。本論文では、このような少なくとも 1 つの目的関数で大幅に劣り Pareto 最適面とはかけはなれているにもかかわらず、殆ど dominate されない解を dominance 抵抗解 (Dominance Resistant Solutions :DRSs) と呼ぶことにする。

もし DRSs が存在し、その探索空間内での割合が Pareto 最適面よりも大きいならば、Pareto 生存戦略を用いた MOEAs は探索に失敗する。なぜなら、本来非 Pareto な質の悪い解が大量に集団内に残ってしまうことで、真に探索すべき部分を重点的に探索することができなくなってしまうからである。

可優越測度

DRSs の“ 他の解に殆ど dominate されない ”特徴を定量化するために、可優越測度という指標を導入する。探索空間 X と解 x、集合 $Y = \{y \in X \mid y \text{ は } x \text{ を } dominate \text{ する}\} \subset X$ とするとき、Y の X に対する測度を x の可優越測度と定義する。これは解 x を dominate できる解の、探索空間全体に占める割合を示す指標であり、小さいほど x が dominate されにくいことを表す。また、探索空間全体ではなく解 x の周辺における x を dominate できる解の割合を、局所的可優越測度と呼ぶ。

Dominance 抵抗解の例

DRSs は、非常に単純な以下の 2 変数 3 目的関数ですら観察される。

Test-2 : $\text{minimize}_{x,y} \quad f_1, f_2, f_3$

$$f_1(x, y) = x^2$$
$$f_2(x, y) = (x - 20)^2$$
$$f_3(x, y) = y^2$$

$$-50 \leq x \leq 50, \qquad -50 \leq y \leq 50.$$

真の Pareto 最適解は線分 $(x, y) = (x, 0), \quad 0 \leq x \leq 20.$ になる。Test-2 では、例えば $(x_A, y_A) = (8, 40), \ (f_1, f_2, f_3) = (64, 144, 1600)$ は明らかに Pareto 最適解からは離れて

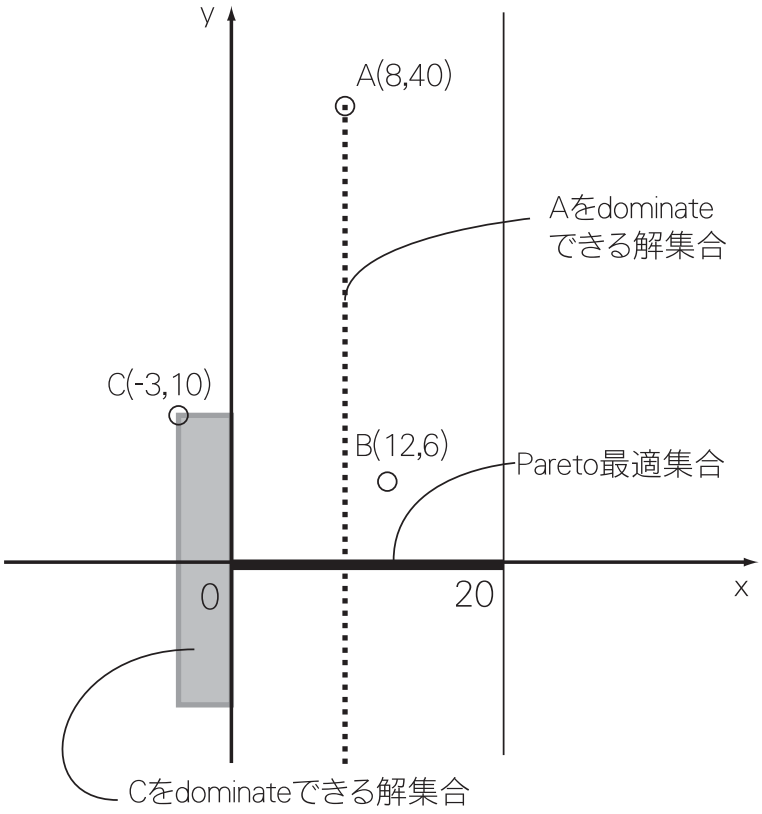


図 5.2: Test-2 の変数空間: 解 C の可優越測度は 0.006, 局所可優越測度は 0.25； 解 A の可優越測度、局所可優越測度は 0。

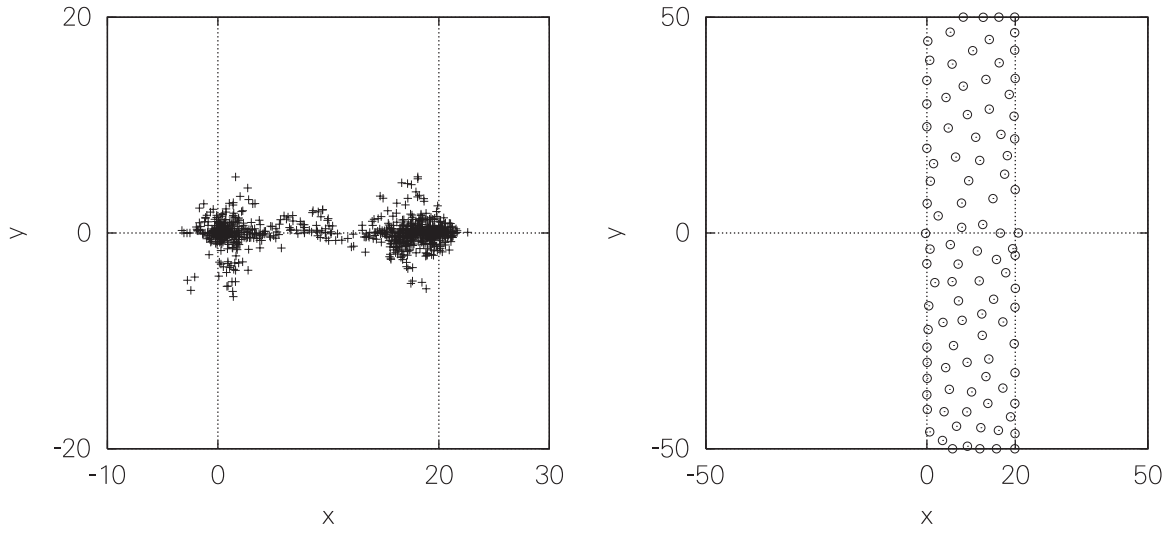
おり評価関数 f_3 は極めて悪い。しかしながら、例えば $(x_B, y_B) = (12, 6),\ (f_1, f_2, f_3) = (144, 64, 36)$ は A よりもずっと Pareto 最適解に近く、良好な解であるが A を dominate することはできない。

実際のところ、$A(8, 40)$ を dominate するような解は線分 $(x, y) = (8, y), -40 < y < 40$ 上にしかない (図 5.2)。すなわち、A の可優越測度は 0 である。通常の突然変異を用いた場合、この線分上の解は決して見つからない。もしバイナリでコード化してビット反転などの突然変異を用いれば、この場合は一つの変数だけが変化することでこの線分上の解は見つかる可能性がある。しかしこの問題を回転させた、本質的には変わらない問題を考えれば、そうした手法も同様に DRS を排除できないと考えられる。

Test-2 では, $0 < x < 20,\ -50 \leq y \leq 50$ の範囲にある解は全て同じ理由で決して dominate されない。つまり、この問題では探索空間の 20%を DRSs が占めていることになる。

Test-2 での PbEA のふるまい

実際に Pareto 生存戦略を用いた MOEA が先ほどの問題で破綻をきたすことを実験で確かめる。PbEA と PPM の実験設定は Test-1 のものと同じである。



(a)stepsize 固定 PPM　　(b)PbEA

図 5.3: Test-2 での MOEAs の挙動：変数空間で表示

図 5.3(a) を見ると、PPM の解は粗さや分断はあるものの Pareto 最適な線分 $(x, y) = (x, 0),\ 0 < x < 20$ の周辺に集まっていることがわかる。対照的に図 5.3(b) を見ると、予想されたように $0 \leq x \leq 20,\ -50 \leq y \leq 50$ の領域に解が分散しており、良い解は全く得られていない。

5.2.3 困難な問題クラス

本節では、多目的最適化問題の中で、容易に DRSs を生じる問題クラスを一つとりあげる。

P_1 を K 目的最適化問題とし、これは n 次元ユークリッド空間 $X = \mathbb{R}^n$ で定義され、$f_1, ..., f_K$ を最小化するものだとする。また X^* を P_1 の Pareto 最適集合とする。同様に、L 目的最適化問題 P_2 を考え、これは m 次元ユークリッド空間 $Y = \mathbb{R}^m$ で定義され、$g_1, ..., g_L$ を最小化するものだとする。また同様に Y^* を P_2 の Pareto 最適集合とする。

さて、P_1 と P_2 を組み合わせることで、$(n+m)$ 変数 $(K+L)$ 目的問題 P_3 を考えることができる。

$$P_3: \text{minimize}_{x_1,...,x_n,y_1,...,y_m} \quad o_1, ..., o_{K+L}$$
$$o_i := \begin{cases} f_i(x_1, ..., x_n) & \text{if } 1 \le i \le K, \\ g_{i-K}(y_1, ..., y_m) & \text{if } K+1 \le i \le K+L. \end{cases}$$

このように、いくつかの部分に分解できる多目的問題クラスをここで *block-separable* 問題と呼ぶ。 Test-2 は明らかにこのクラスに属する。さて、$V = X \times Y$ とし、V^* を P_3 の Pareto 最適集合とすると、以下の関係が成り立つ。本章の命題、定理、系の証明は付録 C2 に付す。

命題： $V^* = X^* \times Y^*$.

ところで $X^* \times Y$ という領域を考えると、これは他の解にめったに dominate されない解である。$x^* \in X^*$, $y^- \in Y \setminus Y^*$ とし、$X^0 \subset X$ および $Y^+ \subset Y$ という集合を考える。$X^0 := \{x \in X \mid \forall i,\ f_i(x) = f_i(x^*)\}$ $Y^+ := \{y \in Y \mid y \prec y^-\}$. ここで、'$\prec$' は domination を表す。すなわち $a \prec b$ とは a が b を dominate するということである。このとき、以下の関係が成り立つ。

定理： $(x, y) \prec (x^*, y^-) \Leftrightarrow (x, y) \in X^0 \times Y^+$.

通常 X^0 の測度は X で 0 である。そのため、$X^0 \times Y^+$ の測度も V で 0 である。よって、$(x^*, y) \in X^* \times Y$ の可優越測度は 0、つまり全く発見されないということをこの定理は主張している。一方 $X^* \times Y$ の領域は V のより広い部分を占める。このため、block-separable な問題では Test-2 で見たように多くの解が DRSs になり (図 5.4)、探索は破綻する。block-separable な問題はごくありふれており、DRSs を回避することは Pareto 生存戦略を用いた MOEAs にとって極めて重要である。

また、目的関数数が増えれば、単純な構造を持つ問題であっても解は dominate され

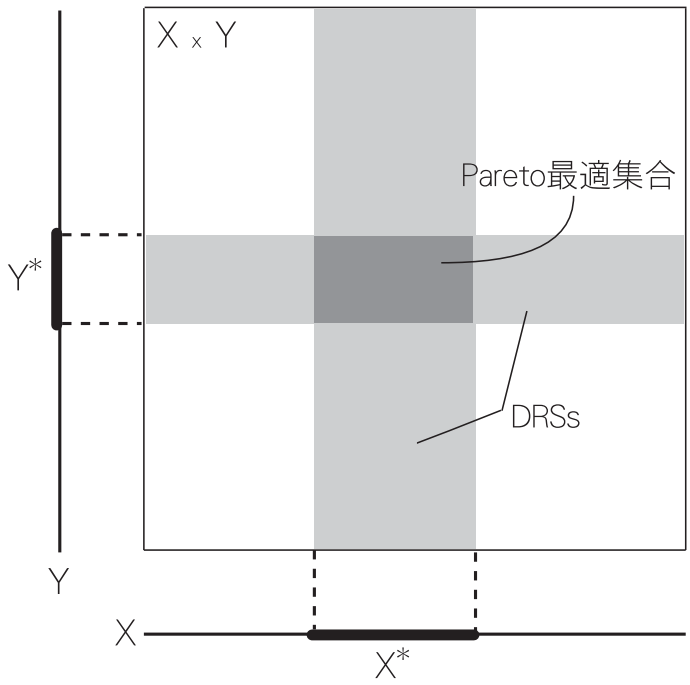


図 5.4: block-separable 問題での DRSs

にくくなる。例えば、m 目的最小化問題 $f_i(\mathbf{x}) = x_i,\ 0 \leq x_i \leq 100$ を考えると、解 $\mathbf{x} = (10, 10, ..., 10)$ の可優越測度は $1/10^m$, 局所可優越測度でも $1/2^m$ しかない。実用的な最適化のためには、多数の目的関数がある場合でも効率よく劣る解を排除する仕組みが必要となる。

5.2.4 dominance 条件の緩和

DRSs が生じること、またユーザにとって無益な解を残してしまうことは、共に dominance の条件が厳しすぎることによっている。Pareto 最適解を求めるという目的から離れ、dominate はされないが総合的に見て劣る解を排除しようとするならば、dominance の条件を緩和することが必要となる。本節では、dominance の条件を緩和した二つの既存手法を取り上げる。

トレードオフ比の導入による単目的化

各目的関数同士の価値の相対比、すなわちトレードオフ比を定めることができれば、多目的最適化問題は単目的最適化問題に置き換えることができる。この場合解は非常に広い

領域の解を自分より劣るとして排除できる（図 5.5）ため、DRSs は生じない上に、もしそのトレードオフ比がユーザによって与えられたものなのであればそのユーザにとって必要とする解だけを得ることができる。

しかしながら通常このようなトレードオフ比を唯一に定めることは非常に困難であるし、またそれゆえに多目的最適化が必要になっているとも言える。基本的に解を一つしか提示できない単目的化は、多目的最適化のそもそもの要求を無視したものであると考えられる。

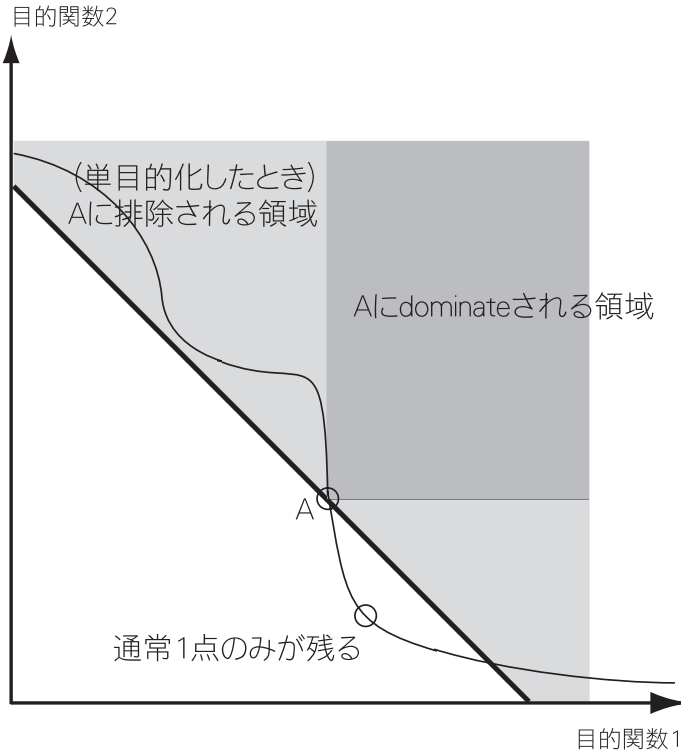


図 5.5: トレードオフ比を用いた dominance 条件の緩和： 濃い領域は A が dominate できる領域を、薄い領域は単目的化された場合に A が排除できる A より劣る領域を表す。

ε -domination

ε -domination 戦略は、解 x が解 y に対して幾つかの目的関数で多少劣っているとしても y を排除することで、探索を停滞させないことを目指した戦略である [Zitzler 2002]。実際、解 x が解 y を ε -dominate するとは、$\forall i\ f_i(x) \leq \varepsilon\ f_i(y)$ となることである（図 5.6）。

しかしながら ε -domination 戦略には、x が y を ε -dominate すると同時に y が x を ε -

dominate してしまうこともあるという致命的な欠点がある。これはすなわち通常の domination に対して求めたい Pareto 最適集合が定まるのに比べ、ε-domination 戦略では“ 安定して他に ε-dominate されない集合 ”が定まらないことを意味する。言い換えれば、ε-domination 戦略を用いて得られる解には何の合理的な意味づけも行うことができない。

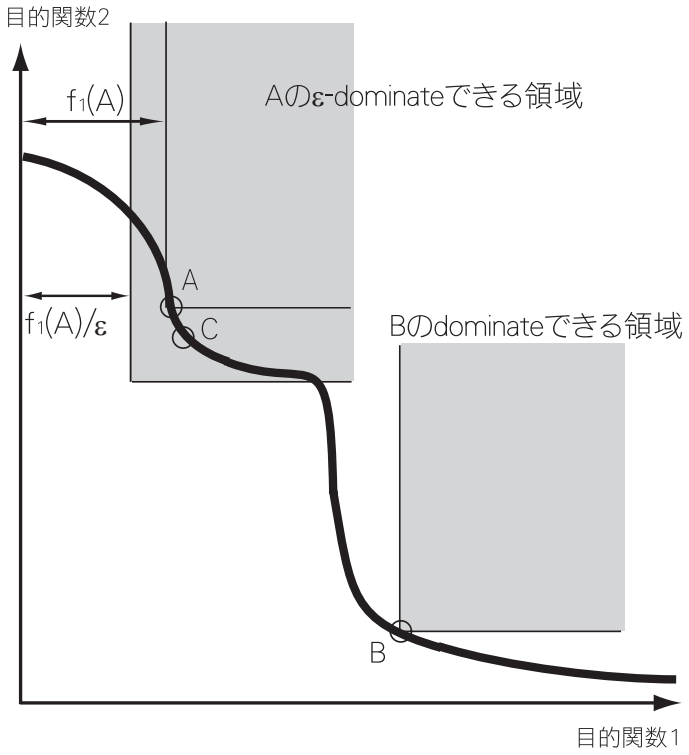


図 5.6: ε-dominate による dominance 条件の緩和：各目的関数が ε (> 1) 倍劣っていても良いため、少し広い領域の解を排除することができる; B が dominate できる領域と、A が ε-dominate できる領域を示す；A は C を、C は A を ε-dominate する。

5.2.5 探索範囲の限定

Pareto 最適解を全て求めることに意味がないことが分かっている場合、各目的関数値 f_i に上限値 τ_i を定めることで探索範囲を限定することも可能である（図 5.7）。これは、Pareto 最適解のうちユーザが許容する範囲だけを探索することを目的とするが、実際には適用が困難な場合が多い。なぜなら、最終的に必要とするような領域には初期解が生成されることは一般的には珍しく、悪い解から徐々にその範囲に入るような解が見つかるからである。

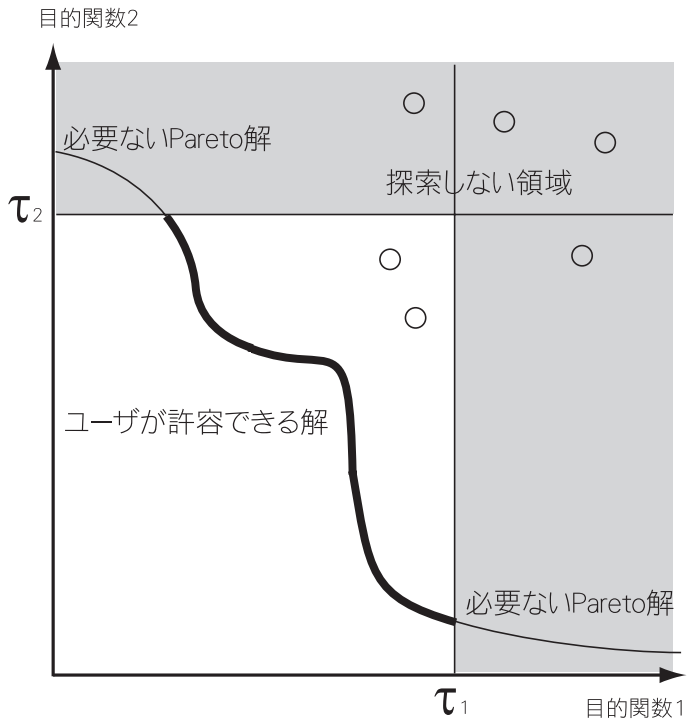


図 5.7: 目的関数の上限値 τ_i を定めることによる探索範囲の限定：灰色部分は探索せず、極端な Pareto 最適解（細い曲線）を提示しないが、限定された範囲内に初期解が見つかる可能性は低い

5.3 接近法： α-domination 戦略

DRSs が生じたりユーザにとって無益な解を提示してしまうのは dominance の条件が厳しすぎるためであるが、従来の dominance 条件の緩和法にはさまざまな問題があり、解決には不十分である。本研究では、ユーザが許容するトレードオフ比の範囲の中だけを探索するように dominance の条件を緩和し、これに適合しない解を排除することで探索の高速化、合理化、さらには DRSs の発生の防止を行う。

5.3.1 トレードオフ比の許容範囲 α_{ij}

トレードオフ比そのものを一つ決定することは非常に困難であるが、その上下界を設定すること、すなわち“ f_j を 1 単位得ることは最低でも f_i を α_{ij} 単位得ること以上の価値がある ”というような $\alpha_{ij} \geq 0$ を決定することはさほど困難ではない。たとえば、US ドルとユーロの厳密な両替レートを決めることは困難であるが、1 ドルには最低 0.1 ユーロの価値があり、 逆に 1 ユーロには最低 0.1 ドルの価値がある、ということは納得できる。

このとき、α_{ij} および α_{ji} はそれぞれ f_i と f_j のトレードオフ比の上下界になっている。すなわち、

$$\alpha_{ji} \leq \frac{\Delta f_i}{\Delta f_j} \leq \frac{1}{\alpha_{ij}},$$

ここで Δf_i と Δf_j は f_i と f_j の変化量である。

5.3.2 α-domination と α-Pareto 最適

K 目的最小化問題 $f_1(x), ..., f_n(x) \quad x \in X$ を考える。ここで

$$g_i(x,y) := f_i(x) - f_i(y) + \sum_{j \neq i}^{1..K} \alpha_{ij}(f_j(x) - f_j(y)) \;\; x, y \in X$$
$$(\alpha_{ii} = 1 \text{ のときは } g_i(x,y) := \sum_{j=1}^{K} \alpha_{ij}(f_j(x) - f_j(y)) \text{ とできる})$$

としたとき、

Def.1. 解 x が解yをα-dominateする ($x \overset{\alpha}{\prec} y$ と書く)$\Leftrightarrow \forall i \; g_i(x,y) \leq 0$, かつ $\exists i \; g_i(x,y) < 0$.

$\alpha_{ij} \equiv 0 \; \forall (i \neq j)$, のとき、$\alpha$-domination は従来の domination と等しい。α-domination は x が y にある目的関数で少し劣り、ある目的関数では大きく優れている場合に dominate することを許す。劣る目的関数数が一つの場合、α-domination により残る解はユーザの求めたい解と必要十分である。一方劣る目的関数数が $n \geqq 2$ 個ある場合は、ユーザの求めたい解と求まる解を近づけるには $\alpha_{i \neq j}$ を n で割るなどの工夫を加える必要があり、最も妥当な実装法については今後の課題である。本論文では、$f_j(x) > f_j(y)$ のとき、係数に α_{ij} の代わりに $1/\alpha_{ji}$ を用いる。すなわち、劣る関数を過大評価することで指定されたトレードオフ比と実際に求まる解を近づけている。

Def.2. $x \in X$ が α-Pareto 最適である $\quad \Leftrightarrow \{y \in X \mid y \overset{\alpha}{\prec} x\} = \emptyset$.

これらの概念に対し、次の系が成立する。

系 1： $x \prec y \;\Rightarrow\; x \overset{\alpha}{\prec} y$.

系 2： α-Pareto 最適集合 $X^\alpha = \{ \, x \in X \mid x$ は α-Pareto 最適 $\}$ は、Pareto 最適集合 X^*

の部分集合である。

既存手法のε-domination戦略が求める解に合理性を持たなかったのに対し、α-domination戦略は“ Pareto最適集合の部分集合であり、かつユーザの許容するトレードオフ比内にある解集合 ”を探索するという合理的な意味を持つ。

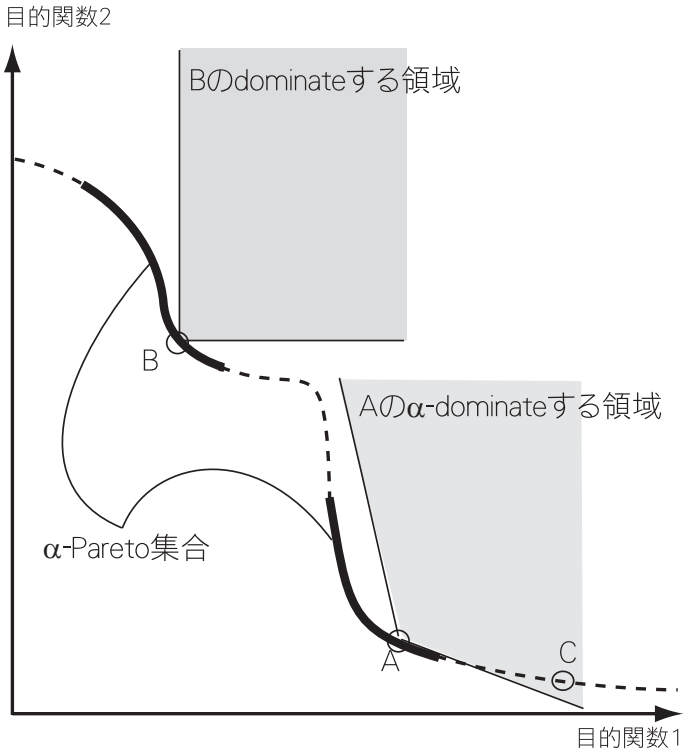


図 5.8: 最小化問題の α-domination と厳密な domination： 解 C は Pareto 最適ではあるが A に α-dominate されるため α-Pareto 最適ではない。

5.3.3 意義

α-domination 戦略を用いるにあたっては、トレードオフ比の許容範囲を定めた行列 α_{ij} を定めなければならない。これは一見従来の domination を用いて Pareto 最適集合を求める戦略に比べて複雑かつ面倒に思えるが、ユーザにとって真に有益な解のみを提示しようとすればむしろ最初から α-Pareto 最適集合のみを求めるのが近道である。さらに、α-domination は domination の自然な拡張になっており、もし α_{ij} を定めるのが面倒であれば、単位行列 $\mathbf{I}$ を用いればこれは従来の domination と等しく Pareto 最適集合を求める

ものになる。

さらに、Pareto 生存戦略を用いた従来の優秀な MOEAs のランキング選択やシェアリングなどの枠組みを残したまま、厳密な domination を α-domination に置きかえることは容易である。つまり、α-domination 戦略を導入するにあたっては、α_{ij} を与えてやり、$g(i,j)$ を各々計算して従来のアルゴリズムの dominance 判定部分と置き換えてやるだけでよい。

α-domination 戦略を導入することにより、DRSs を生じさせないで探索を進めることができ、またユーザが必要とする解だけを探索し、不必要な解を探索するために時間を費やす必要がなくなるということである。

5.4 比較実験

α-domination が DRS に対処していることを示すために、Test-2 に α-domination を用いた PbEA を適用した。α_{ij} の値は全ての $i \neq j$ で一定値 c に固定し、$c = \frac{1}{3}, \frac{1}{9}, \frac{1}{99}$ として変化の様子を調べた。それぞれの場合に対して、線分 $(x, 0)$ $5 \leq x \leq 15, 2 \leq x \leq 18, \frac{1}{5} \leq x \leq \frac{99}{5}$ が α-Pareto 最適集合となる。

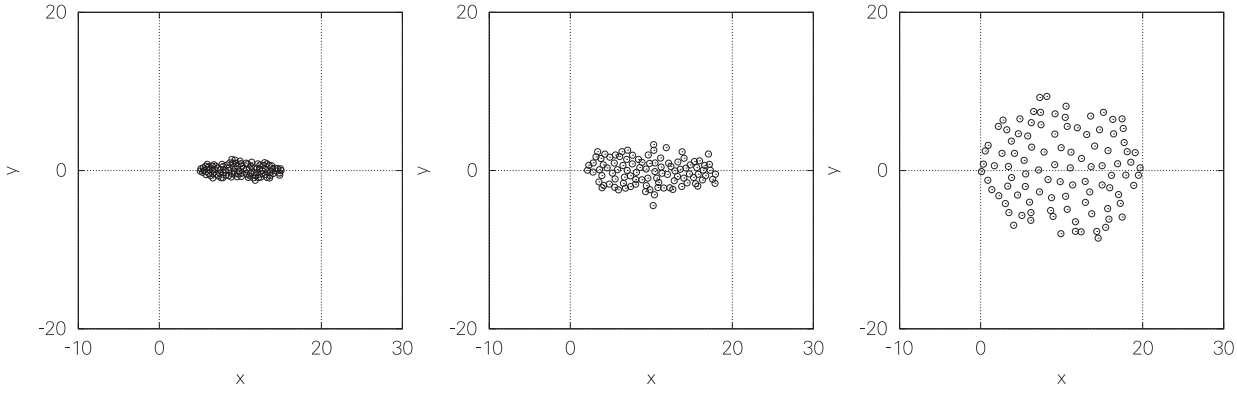


図 5.9: Test-2 での、α-domination を用いた PbEA; $\alpha_{i \neq j}$=c; (a)$c = 1/3$, (b)$c = 1/9$, (c)$c = 1/99$

図 5.3 との比較で良好な探索をしている様子が図 5.9 からわかる。もし α_{ij} の値を大きくすれば domination の条件は緩和され探索は精密化される一方、α-Pareto 集合は小さくなる (a)。逆に値を小さくすればするほど、厳密な domination を用いた PbEA の結果に近づく (c)。

次に、局所可優越測度の観点から α-domination の効果を確認する。局所可優越測度は即ち、ある Pareto 最適でない解の微小な近傍のうち、その解を排除できる解の割合を示す。つまり局所可優越測度が大きいほど、探索が高速に進むことを意味する。m 目的最小化問題 $f_i(\mathbf{x}) = x_i, 0 \leq x_i \leq 100$ および解 $\mathbf{x} = (50, 50, ..., 50)$、また各次元標準偏差 1 の近傍を考える。この場合通常の domination においては局所可優越測度は $1/2^m$ となる。表 5.1 に目的関数数 $m = 2, 3, 4, 5$ とし、$\alpha_{i \neq j} = c = 0, \frac{1}{9}, \frac{1}{3}$ とした場合の α -domination における局所可優越測度を示す。$\alpha_{i \neq j} = \frac{1}{3}$ とはつまり「1 ドル＜3 ユーロ＜9 ドル」といった非常にゆるい許容範囲であるが、これを与えただけでも目的関数数が 5 になれば、通常の domination を用いるのに比べ 6 倍の確率で解 $\mathbf{x}$ を排除できることになる。

表 5.1: 目的関数数と α $_{i \neq j}$=c を変化させた場合の、解 $\mathbf{x}$ の局所可優越測度の変化; 大きいほど探索が進みやすい。通常の domination では目的関数数が上がると急激に探索は遅くなるが、α -domination では十分高速な探索が可能

目的関数数	$c = 0$; 通常の domination	$c = \frac{1}{9}$	$c = \frac{1}{3}$
2	0.250	0.285	0.353
3	0.125	0.179	0.284
4	0.062	0.119	0.240
5	0.031	0.083	0.206

5.5 まとめ

従来の多目的最適化は、Pareto 最適解をまんべんなく求めることを最大の目標としていた。このために、1）ユーザにとって真に有益な解だけを提示することができず、2）質の悪い解を効率良く排除することができず問題によっては DRSs が生じ探索が停滞する、といった問題があった。これらは共に domination の条件が厳しすぎることに由来しており、その緩和が必要であるが従来の ε -domination 戦略などの緩和法には致命的な問題があった。

本研究ではユーザが許容するトレードオフの範囲を行列 α_{ij} で表し、これを用いて domination の条件を緩和することで質が悪いが dominate されにくい DRSs やユーザにとって不要な解を効率的に排除する α -domination 戦略を提示した。α -domination 戦略は Pareto 最適集合の中でユーザが許容するトレードオフ範囲内のものだけを探索する能力を持ち、

また特に目的関数数が多くなるほど通常の domination に比べて高速に探索することができるといった利点を持つ。

これらから、Pareto 最適解をまんべんなく求めるという従来の多目的最適化の目標を柔軟化し、ユーザが真に必要とする有益な解だけを求めるための最適化戦略を提案することができたと言える。α-domination 戦略の適用可能クラスおよび設計指針、パラメータの設定法については付録 C1 に示す。

第6章　独立制約充足戦略に基づく制約条件付最適化

本章では、制約条件を効率的に処理し、最適化の妨げにならないようにする戦略を提案することを目標とする。これにより、従来法では制約を満たしただけの安易な解しか発見できなかったような、複雑かつ目的関数とは対立的な制約条件を持つ困難な問題にまでGAの適用クラスを広範化することができると考えられる。

6.1　はじめに

GAをはじめとする確率的最適化手法を実世界の問題に適用する際に、解に要請される制約条件をどのように扱うかは非常に重要な問題となる。最も単純なものとしては、制約条件を満たさない解は直ちに放棄する方法や、問題に特有な発見的ロジックにより制約を満たすように強制操作を加える方法が用いられる [坂本 2000] [Giffler 60]。しかしこのような手段は簡単な操作で比較的容易に制約充足解が得られるような場合に限られ、複雑・多様な制約条件を持つ実問題で有効であるとは言い難い。

そこで、最近のGAの実問題への適用においては、制約違反の度合いに応じて個体の適応度にペナルティを加える方法がとられることが多い [安達 2001]。これにより、進化が進むにつれてより深刻な制約違反を犯す解が淘汰され、最終的には制約条件を全て満たしつつ本来の目的関数を最適にする解を得ることが期待される。しかしながら、制約を満たすことと目的関数を改善することは多くの場合対立的である。すなわち、制約条件を満たす為には解の質が犠牲になり、逆に解の質を向上させようとすると制約を違反する可能性が高い。このような場合にGAなどで最適化を行うと、制約を満たしただけの安易な解に収束してしまい、真に質の高い解を得ることは困難である。本章ではこの現象のメカニズムを明らかにするとともに、制約条件を取り扱うための、より適切な方法について論じる。

制約充足と本来の目的が対立する実問題として、本章では下水道送水系の制御プランニング問題を取り上げる。適切な制約条件の処理に加え、不確実な予測への対処を行うことにより、実際の運用に十分適用可能な制御が獲得できることを示す。

6.2 従来法と問題点

6.2.1 従来法の分類

最適化問題を GA などの手法で解く場合、制約条件をどう扱うかは極めて重要な問題である。特に実問題には非常に多数のかつ複雑な制約条件を持つものが少なくなく、制約充足を含めて制約条件の取り扱いが性能に決定的な影響を及ぼす。通常用いられる手段は以下の 4 通りに分類できる。なお、これ以降簡単のため目的関数の**最大化**問題に議論を限定するが、一般性は失われない。

- **解の放棄**

  制約を満たさない解が生成された場合その解を放棄し、制約を満たす解が見つかるまで再初期化を繰り返すか、記録しておいた制約充足解を再び用いる。GA においては個体を致死（適応度 $-\infty$）とすることなどがこれにあたる。最も簡単に実装できるが、制約を満たす解が簡単に得られる場合に限られる。

- **強制操作**

  制約を違反した解を、問題に依存した何らかの発見的ロジックによって制約を充足するように強制操作を加える。この際解の質を犠牲にすることが多く、また複数の制約条件がお互いに複雑に関連しているような場合にはこのような操作の発見は非常に困難になり、実用的ではない。

- **ペナルティ加算法**

  制約を違反した度合いに応じて、本来の目的関数に負のペナルティを加える。全ての制約を即座に満たすことが困難であるような場合に、探索が進むにつれ徐々に違反の度合いを減少させることが期待できる。これには多くの適用例があるが、次節で詳述するように“ 優れているが制約をわずかに満たさない解 ”よりも“ 制約を満た

す安易な解 ”を優先して選択していくために、真に質の高い解を得るのが困難になる場合もある。

なお本論文では、ペナルティを別の方法で用いる手法と区別するために、本来の評価関数に加算する本法を**ペナルティ加算法**（Additive Penalty Method, APM [Francisco 2000]）と呼ぶ。

- **多目的アプローチ**

  本来の目的関数の最適化と、制約違反のペナルティの最小化を別の軸で捉え、多目的最適化手法 [Knowles 99] を用いて解く。“ 制約を満たさない優れた解 ”も保持され続け探索に貢献できるため、ペナルティ加算法よりも質の高い解を得られる可能性が高い。しかしながら、多様な解を保持するため相対的に長い計算時間を要することや、高度な多目的最適化手法の実装の困難さなどの課題を持つ。

より好ましいペナルティ関数の設計については [Deb 2000] 等の研究があるが、制約の充足と目的関数の改善が本質的に対立的である以上、ペナルティ加算法は十分な性能を発揮できない。次節では、一つのトイプロブレムを例に挙げてこの理由を明らかにし、多目的アプローチの一種を用いたより適切な手法を示す。

6.2.2 加算的ペナルティ法の問題点

数直線区間 $[0, L]$ 上に、長さ 1 の線分を重ならずに何本置けるかという問題を考える。これは、重なりがないという制約条件のもとで本数を最大化するフロアプラン問題の一種と捉えられる。

線分配置問題

決定変数： $n \in \mathbb{N}$, 及び $(x_1, x_2, ..., x_n) \in [0, L-1]^n$
目的関数： n （最大化）
制約条件： $\forall (i, j), \quad |x_i - x_j| \geq 1$

線分配置問題の解は n によって可変長となる。本数 $n = i$ のときの探索空間は $S_i = [0, L-1]^i \subset \mathbb{R}^i$ であり、問題全体としての探索空間は $S = \cup_i S_i$ である。この問題は自明な最適解 $n = \lfloor L \rfloor$ を持つが、問題に対する事前知識を用いない場合は十分難しいベンチ

マークになりうる。これをペナルティ加算法による局所探索で解く場合、制約を充足させるために次のようなペナルティ関数が考えられる。

$$P(n;\ x_1, x_2, ..., x_n) := \sum_{i,j} p(\ |x_i - x_j|\ ),\ \text{ただし}\ p(d) := 1 + 100(1-d)^2,\ if\ d < 1.$$

探索オペレータには、

- 【↔】半径 $r \leq 1$ を決め、各 x_i に $\delta_i \in [-r, r]$ を加える
- 【＋】ランダムな位置に一つ線分を加える
- 【－】線分の一つを取り除く

を組み合わせて使うとする。このときペナルティ加算法の探索は以下のような過程を辿ると考えられる（図 6.1）。

1. 探索初期：ランダムに配置された線分はお互いに重なり、初期の解 (a) は多くの制約違反を持つ。
2. 探索序盤：オペレータ【－】により重なっている線分が取り除かれ (b)、制約を満たした解 (c) が生成されはじめる。
3. 探索中盤：オペレータ【↔】と【＋】がうまく協調した場合、評価値を 1 上げる解 (d) が生成される。
4. 探索終盤：制約を満たすように新たな線分が加わった解 (e) が発見される確率は極めて小さくなる。

このような現象は、各探索空間 S_n の評価値景観の違いによって生じる。制約を満たす解を比較すれば本数 n の大きいほうが評価値も高いが、容易に想像がつくように本数が増えるほど制約を満たすのは困難になり、平均的な解の評価値は劣ることになる（表 6.1 に例を示す）。このように、真に優れた解を含むが全体に評価値の低い部分空間（この場合 S_4）と、全体に評価値が高いが最適解を含まない部分空間（この場合 S_3, S_2）がある景観構造を UV 構造と言う [池田 2002]（図 6.1 下）。UV 構造下で GA などの確率的最適化を行うと、真に優れた解を発見する前に一見有望である局所解しか含まない側の部分空間を重点的に探索してしまい、最適解を発見できないという現象（UV 現象）が生じる。

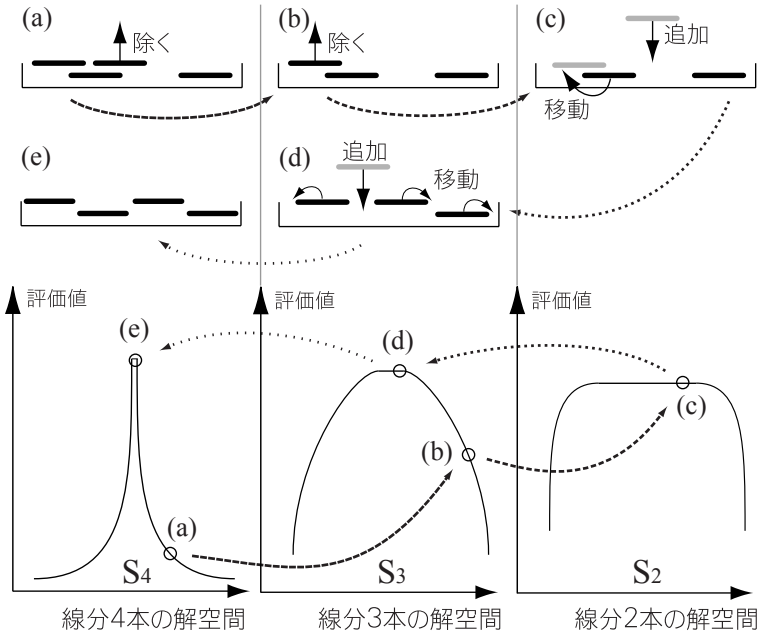


図 6.1: 線分配置問題におけるペナルティ加算法の典型的な探索過程：（上）代表的な解の様子,（下）評価値景観における解の位置; (a) 初期解, (b) 違反が一つ減った解, (c) 制約を満たしただけの解, (d) 制約を満たしつつ改善された解, (e) 最適解

表 6.1: $L = 4.5$ とした場合の、部分探索空間 $S_1, ..., S_5$ における解の最大評価値と平均評価値、および制約充足解の割合

探索部分空間	最大評価値	平均評価値	充足解の割合
S_1	1	1	100%
S_2	2	-16.18	50.95%
S_3	3	-51.6	7.79%
S_4	4（最適）	-104.9	0.039%
S_5	-5.25	-176.9	（存在しない）

線分配置問題に限らず、目的関数と制約条件が対立的な場合このような現象は頻繁に生じる。このとき、もし最適解を含む部分空間だけを独立に探索できるならば、他の一見有望な部分空間に騙されることはなくなる。

6.3 接近法：独立制約充足

ペナルティ加算法が探索に失敗する原因は、制約違反を減らすために同時に解の質を悪化させてしまう妥協にある。目的関数値の等しい部分空間（前節の例ならば $S_1,\ S_2,\ S_3, ...$）

だけを独立に探索すれば、このような現象を防ぐことができると考えられる。これを**独立制約充足**（Seperated Constraint Satisfaction, SCS) と呼ぶ。

まず単純に考えると、探索空間を目的関数値のほぼ等しい部分空間に分け、それぞれを独立に探索し、制約を満たした中で最も良い解を選択すれば良い。

独立制約充足 ： 並列型

1. 探索空間 S を、N 個の部分空間 $S_{B_i}^{B_{i+1}}, \quad B_i \in \mathbb{R}$ に分割する。このとき解 $x \in S_{B_i}^{B_{i+1}}$ は $B_i \leq f(x) < B_{i+1}$ を満たすとする。すなわち、この部分空間群は一定範囲内の目的関数値を持つ。
2. 全ての部分空間を、$B_i \leq f(x) < B_{i+1}$ という条件を満たしたまま、ペナルティを 0 にするように一定期間最適化を行う。
3. 制約を満たした解の中で、最も目的関数値が高いものを採用する。

この手法ならば、部分空間 $S_{B_i}^{B_{i+1}}$ を探索している間は、違反を減らすために妥協して目的関数値を B_i 未満にすることはない。すなわち B_i はこの部分空間の探索における保証値と言うことができる。次に、この並列型の改良を考える。

まず、部分空間の中だけを探索するという意味では手順 (2) の条件 $f(x) < B_{i+1}$ は必要であるが、より良い解になる可能性を考えればこの部分は必要ない。次に、一度ある $f(x) = B$ なる制約充足解 x を発見したならば、これより劣る $f(y) < B$ なる解を探索することには意味がない。これらのことから、達成が困難な目標値から制約充足が可能かを試し、制約充足できるまで徐々に目標値を落としていくタイプが考えられる。これは一般の制約付き最適化における外点法にあたる。

独立制約充足 : 降下型

1. 目的関数の目標値 LB_f を**高め**に定める。
2. 制約違反のペナルティを 0 にするように一定期間最適化を行う。ただし、このとき $f(x) < LB_f$ となる解 x は許容しない。
3. 制約充足に**失敗**したら、目標値 LB_f を**下げて** step2 に戻る。
4. 充足に成功するまで以上を繰り返す。

降下型は並列型に比べて楽観的な手法であり、初期の目標値を高く設定しすぎると、一度も制約に成功することなく計算時間が終了してしまう恐れがある。逆に、確実に制約充足しながら徐々に目標値を上げていくタイプも考えられる。これは一般の制約付き最適化

における内点法にあたる。

独立制約充足 : 上昇型

1. 目的関数の目標値 LB_f を**低め**に定める。
2. 制約違反のペナルティを 0 にするように一定期間最適化を行う。ただし、このとき $f(x) < LB_f$ となる解 x は許容しない。
3. 制約充足に**成功**したら、目標値 LB_f を**上げて** step2 に戻る。
4. 充足に失敗するまで以上を繰り返す。

上昇型は降下型に比べて悲観的な手法であり、初期の目標値を低く設定しすぎると、延々と制約充足できて当たり前の部分空間を探索し、真に得たい辺りの解を十分に探索できないという恐れがある。我々が求めたいのは、“ 制約が充足できるぎりぎりの良い解 ”なのであるから、もしこれがある程度分かっているならばより適切な初期値を用い、適応的に探索を行うべきである。

独立制約充足 ： 適応型

1. 目的関数の目標値 LB_f を定める。
2. 制約違反のペナルティを 0 にするように一定期間最適化を行う。ただし、このとき $f(x) < LB_f$ となる解 x は許容しない。
3. もし制約充足解が見つかれば、目標値 LB_f を上げて step2 に戻る。ただし、以前制約充足に失敗した目標値以上の値にはしない。
4. 制約充足に失敗したら、目標値 LB_f を下げて step2 に戻る。ただし、以前制約充足に成功した目標値以下の値にはしない。
5. 許された計算期間、以上の手順を繰り返す。

実際に独立制約充足を問題に適用しようとする場合は、それぞれの特性に合わせて以上の 4 つのタイプから適したものを選択することができる。どの場合も、目標値 LB_f の初期値と変量は性能に大きな影響を与えると考えられる。

6.3.1 既存研究との関連

独立制約充足は既存の 2 つの手法から着想を得ており、制約付き最適化問題に適用する場合のこれら手法の応用または新しい解釈と捉えることができる。

評価値の景観が大域的に多峰である場合、そのそれぞれが異なった特徴を持っており、最適解を含む領域が他の領域に比べ平均的な評価値が低いケースはしばしば生じ、制約を

ペナルティ加算法で解いた場合はまさにこれにあたる。**生得分離モデル** [池田 2002] はこのような場合にそれぞれの領域だけを独立に探索するために開発された、GA におけるモデルである。生得分離モデルでは GA の個体群は複数の小集団に分けられ、それぞれが小さな領域で初期化されその中でのみ交叉・世代交代を行う。これは即ち、線分配置問題で言えば部分空間 $S_1,\ S_2,\ S_3, ...$ だけをそれぞれ探索することを狙ったものである。そうすれば平均的な評価値が高い部分空間に騙されることなくそれぞれの中で探索が進み、いずれかは制約違反のない真に質の高い解を得るだろう、という期待は独立制約充足と全く同じである。

一方 6.2.1 節でも述べたとおり、制約付き最大化問題は目的関数最大化と制約違反最小化という 2 目的問題として扱うことができる。このとき、独立制約充足は、多目的最適化アルゴリズムの ϵ **制約法** [Bhaskar 2000] の新しい解釈とも捉えることができる。ϵ 制約法は、単一目的として一つの目的関数 $f_i(x)$ を採用し、それ以外の目的関数 $f_{j\neq i}(x)$ には下界値 ϵ_j を与えて $f_j(x) < \epsilon_j$ なる解 x は破棄するものである。ϵ_j の値をいろいろに変化させることで、その下界値のもとでの最も良い $f_i(x)$ を得ることが期待できる。

下界値 ϵ_j 以上でなければならない $f_j(x)$ を本来の目的関数、$f_i(x)$ が 0 に最小化したい違反であると考えれば、これは目標値 $LB_f = \epsilon_j$ としたときの独立制約充足に他ならない (図 6.2)。多目的最適化は本来は多様なトレードオフを持つ Pareto 解を得るために用いられるが、我々が対象としている問題はそのうち $f_i(x) = 0$ となりかつ最大の $f_j(x)$ を持つ解だけを求めたいわけであり、数ある多目的最適化手法の中でも特に ϵ 制約法が応用として適している。

6.3.2 線分配置問題おける実験

本節では、線分配置問題にペナルティ加算法と独立制約充足を用いた局所探索を適用してその性能を比較する。独立制約充足では上昇型を用い、まず $LB_f = 2$ として本数を変更しないようにオペレータ【↔】だけを用いて制約充足を行う（注：オペレータ【＋】を用いればより良い性能を得られるがここでは用いない)。これに成功した場合 $LB_f = 3$ とし、以降制約充足に失敗するまで 1 ずつ目標値を増加させてゆく。これにより、オペレー

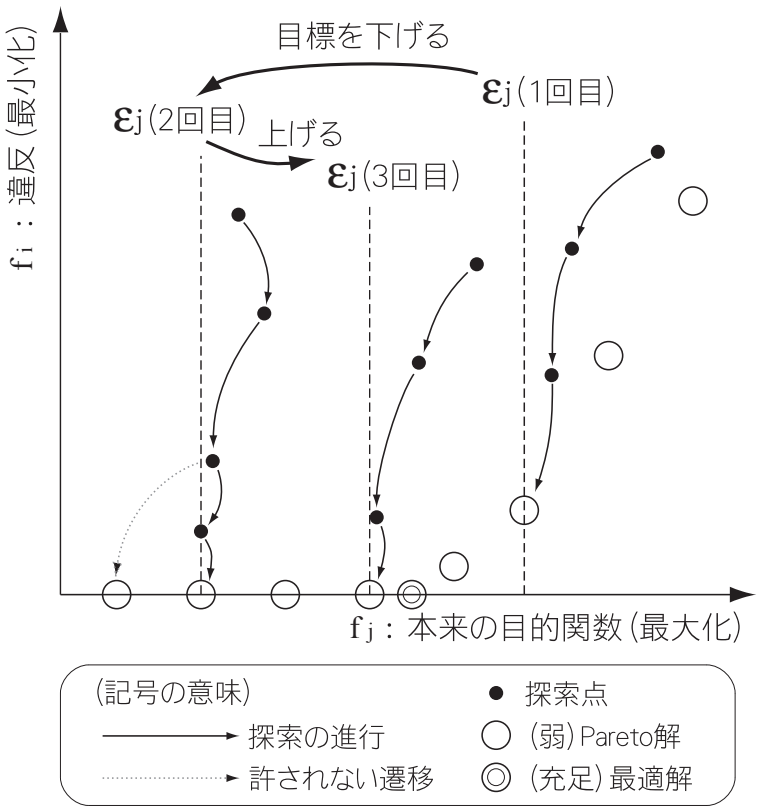


図 6.2: ϵ 制約法による最適化：横軸は本来の目的関数の最大化, 縦軸は違反の最小化を行う 2 目的問題; (1) まず $LB_f = \epsilon_j$ を高めに設定するが制約充足解に達しない。(2) そこで ϵ_j を低めると制約充足解を発見する。途中、質を妥協して制約を満たすような変異は許容しない。(3) さらに ϵ_j を高めるとより質の高い制約充足解を発見する。

タ【−】によって安易に制約充足することなく探索を進めることができると考えられる。

問題サイズ $L = 4.5,\ 5.5,\ ...29.5$、打ち切り総評価回数を 10 万回とし、それぞれ 20 試行分のエラー平均を計測したものが図 6.3 である。両者ともペナルティ関数は同じものを用いていることに留意されたい。

この結果、ペナルティ加算法は L=10.5 程度でも最適解を発見できなくなるが、独立制約充足では L=17.5 まで全試行において最適解を発見できていることがわかる。また打ち切り総評価回数に対して問題サイズが大きくなりすぎると、n=2 から全ての本数について独立に制約充足していくのは効率が悪くなるという上昇型の欠点が顕著になり、両者の性能に大きな差はなくなる。

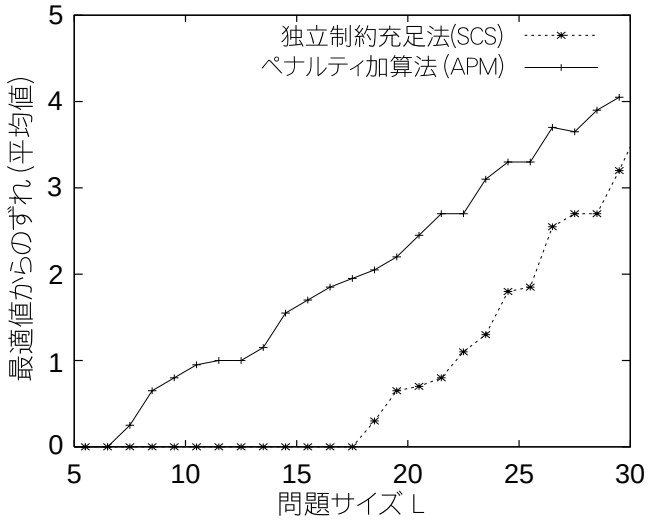


図 6.3: 線分配置問題におけるペナルティ加算法と独立制約充足の性能比較：問題サイズ L に対する, 最適値 $\lfloor L \rfloor$ からのずれ

6.4 下水道送水系制御問題への適用

下水道は近代生活において非常に重要な役割を担うライフラインのひとつであり、いまや都市部では必要不可欠なものとなりつつある。近年では下水道施設の広範な普及やコスト削減の必要性により、送水系の効率的な運用および自動化、あるいは実運用に対する補助的支援などが望まれている。現実の下水道制御の多くはいまだに熟練者の経験に基づいて運転されており、これを最新の最適化手法により自動化することができれば多くの恩恵がある。対立する制約条件を伴う実問題は資源割当問題、物流・運送に関する問題など枚挙に暇が無いが、特に上水道配水系はフローが逆である問題であり本論文のアプローチがただちに適用できる。

6.4.1 問題設定

下水道送水系は一つの処理場と複数の中継ポンプ場からなり、それらが配管によって連結している（図 6.4）。各施設は貯水池を持ち、各タイムステップでその時間間隔に揚水する量を決定し、下流の施設に送り出す。これを最適化問題として捉えると以下のようになる。

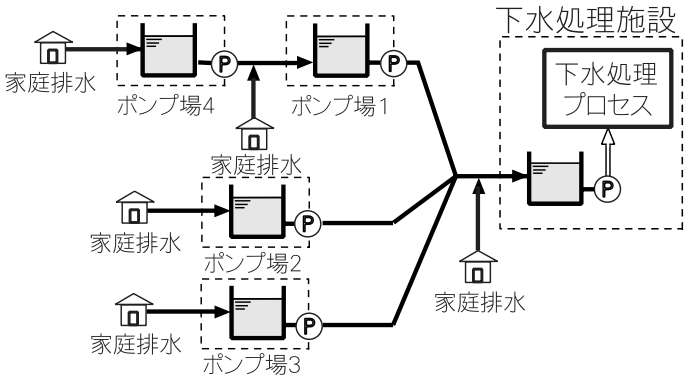


図 6.4: 下水道送水系の例 ： 5 施設

- **主目的 ： 処理量平滑化**

  下水処理施設では排水を浄化する過程で沈殿や生物化学処理を施す必要がある。この処理には通常長い時間を要し、薬剤や有機物の設定変更には大きなコストがかかる。そのため、設定の変更を少なくするために処理量（＝処理場の揚水量）をできるだけ一定に保ちたいという要請があり、これが第一の目的とされる。

- **制約条件 : 水位の上下限**

  各施設が持つ貯水池の水位は一定の範囲内で運用されなければならない。貯水池が大きければそれだけ柔軟な制御ができ処理量をより平滑化しやすくなるが、立地・維持コストも高くなるため現実には厳しい制約が課せられる。

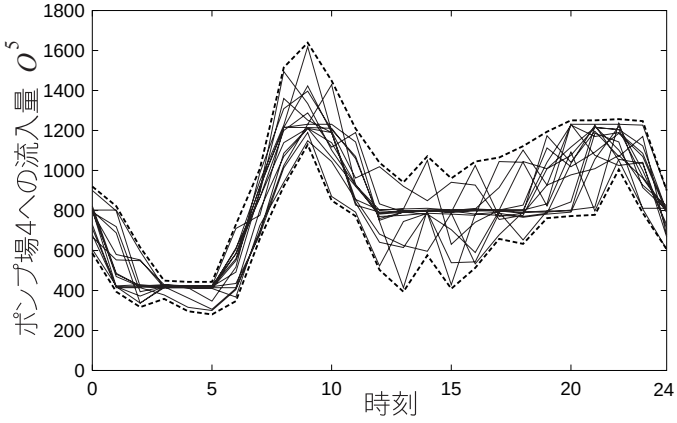


図 6.5: ポンプ場 4 に流入する家庭排水量の時間系列データ： 実線は 1 日ごとの排水量、点線はその最大・最小値

- **アルゴリズムへの状態入力**

  一つの下水道系は集中管理され、現在の水位に関する情報に遅れはないものとする。管理者は、現在の日時・水位・揚水量に加え、将来の家庭排水の流入量予測に基づいて運用計画を作成する。各家庭は任意に排水を行うために流入量を事前に正確に把握することはできないが、例えば深夜に少なく昼や夜に多いといったパターンは持っており（図 6.5)、ある程度の予測は可能である。

- **アルゴリズムの出力：各施設の揚水量**

  各施設には複数のポンプが設置されており、その離散的な組み合わせで揚水量を調整する。管理者は、各タイムステップごとに、各施設がどれだけの揚水を行うかを決定する。あるいは、ただ単に揚水量を決定するだけでなく、どのポンプを稼動させるかを決定し、コスト的に好ましい組み合わせを選択する必要もある。

- **副目的：運用コスト**

  現実の問題には多くの副目的が伴い、それは下水道制御においても同様である。例えば、夜間の安価な電力を使用することでコストを削減したい、中継ポンプ場の揚水量もできるだけ平滑化したい、などである。本論文においては、処理場の揚水量平滑化が最も優先度が高く、これが同等の場合のみ副目的による優劣をつけるものとする。

6.4.2 既存研究

これまで上下水道制御問題に対しては、運用計画最適化を整数計画問題とみなした上で線型計画法、動的計画法、分枝限定法などを用いる方法が研究されてきた。しかしこの整数計画問題はいわゆる NP 困難な問題であるため、大規模な問題を十分きめ細かく最適化しようとすると現実的な計算時間では解けなくなる恐れがある。そのため、近似的最適解を高速に得るために多段線型計画法とヒューリスティックをあわせた手法 [栗栖 94] や確率的最適化手法でプランニングする研究が行われている。

また、分散強化学習はネットワーク状に連接した施設を不確実性のもとでも制御できる手法として注目されており、下水道送水系にも適用され実運用に耐えうる政策を獲得して

いる [青木]。強化学習では過去の汚水流入量を学習データとして用い、それに対する適切な制御を事前に獲得しておくことで、同様のパターンを持つテストデータに対して再計算することなく制御を出力することができる。その一方、ポンプの故障や特異日など環境が大幅に変更された場合、再学習を行わなければ良好な制御ができないという欠点も持つ。

6.4.3 GA による制御

遺伝的アルゴリズムによる送配水の制御としては、上水道に対するアプローチがなされている [坂本 2000]。GA の各個体は一日分の各施設の送水スケジュールを表し、それぞれについて浄水需要予測に基づきシミュレーションが行われ、適応度が計算される。水位制約を満たさない解は致死個体となるが、このままでは制約を満たす解が殆ど見つからないために、問題に固有のバックトラックロジックを用いた強制操作が行われる。強制操作は実行可能解を発見するためには貢献するが、水位制約違反する時刻の配水量を近視眼的に変更するために平滑化という主目的に対しては対立するものである。

また、この GA によるアプローチを下水道送水系に適用しようとする場合、制約条件の扱いに加えて予測精度の悪化が大きな問題となる。上水道配水系においては需要に対する供給不足は絶対に避けなければならないために非常に高精度な需要予測が行われているが、下水道送水系における流入量予測の精度は雨水の流入などにより比較的低いのが現状であり、実用上は予測からの誤差は現場の技師の経験と勘で補正されることになる。運用計画を GA によって最適化しても、一日の後半では誤差が蓄積されて予定していた運用では水位制約を満たせなくなる事態が発生し、結果として期待される処理場揚水量の平滑化は十分には行われないと考えられる。

6.4.4 アプローチ

本論文では下水道制御問題の特徴と現状の GA の不具合を踏まえた上で、以下の 2 点を特に重視してより良い制御を可能にする手法を提案する。

- **毎タイムステップでの再スケジューリング**

  従来の運用計画法では、早朝に一日分のスケジュールを作成して特に問題のない限

り終日それに従う。しかし前述の通り下水道系においては予測は十分正確ではなく、しばしば予定された水位と実際の水位がかけ離れることで近視眼的な運用変更が必要となる。そこで提案手法では、実際に観測された水位に基づき運用計画を毎タイムステップ見なおすことにする。すなわち、スケジュール作成は1ステップ後（例えば現在から20分後まで）の揚水量決定のために行われ、将来にわたって従うことを前提にしたものとはしない。これにより、最新の情報に基づいた意思決定が可能となり、精度の低い予測のもとでも適切な制御が可能になると考えられる。

- **揚水量平滑化と制約充足の分離**

  水位制約を満たすために強制操作を用いたりペナルティ加算法を用いることは、制約を満たすが質の劣った安易な解を多く生み出す原因となると考えられる。提案手法では独立制約充足型GAを用い、揚水量の切り替え回数を固定した上でそのそれぞれについて制約充足だけを行い、制約を充足した中から最も切り替え回数が少ない解を最終的に選択する。制約充足の段階では切り替え回数は変更しないため、制約違反の解消が平滑化に悪影響を与えないということが期待できる。

6.4.5 実験設定

本論文では5施設からなる下水道送水系のモデル[青木]を用いる。各施設iは池底面積$B^i[m^2]$, 上限運用水位$h^i_{MAX}[m]$, 下限$h^i_{MIN}[m]$の貯水池とn^i台の揚水ポンプ$P^i_{1..n^i}[m^3/h]$を装備し、それぞれ下流施設に接続している（図6.4）。本論文で用いたこれら施設の緒元を表6.2に示す。

表 6.2: 5施設下水道送水系の緒元

施設	接続先	池底面積 $B_i[m^2]$	運用上下限 h^i_{MAX}〜$h^i_{MIN}[m]$	揚水ポンプ $P^i[m^3/h]\times$ 台数
処理場	-	2000	0.8〜−2.5	$980\times 2, 2000\times 2$
ポンプ場1	処理場	1000	0.8〜−2.5	$500\times 2, 1000\times 2$
ポンプ場2	処理場	1000	0.8〜−2.5	$300\times 2, 500\times 2$
ポンプ場3	処理場	300	0.8〜−2.5	$150\times 2, 400\times 2$
ポンプ場4	ポンプ場1	600	0.8〜−2.5	$300\times 2, 500\times 2$

本論文で用いる下水道送水系のシミュレータの動作は以下の通りである。シミュレータ

は、(a)GA が個体の適応度を評価する際、(b) アルゴリズムの性能を比較評価する際の両方に用いられる。シミュレータは 1 タイムステップ（$T_{step}[min]$）ごとに計算を行う。

シミュレータの動作

1. 時刻・水位・ポンプ稼動状況を初期化
2. 時刻 T_t, 各施設 i の水位 $h^i(t)$, 1 ステップ前の揚水量 $a^i(t-1)$ を観測
3. 1 ステップ分の（実際の, あるいは予測の）家庭排水量 $o^i(t)$ を T_t から算出
4. アルゴリズムに従って揚水量 $a^i(t)$ を決定
5. 貯水池水位を更新 ： 上流施設からの施設 i への揚水量を $f^i(t)$ と書くと、$h^i(t+1) := h^i(t) + \frac{o^i(t)+f^i(t)-a^i(t)}{B^i}$.
6. $t < t_{MAX}$（シミュレータ終了ステップ数）なら、ステップを $t+1$ に更新して（2）に戻る
7. 処理場揚水量の変更回数と水位制約違反度合いを出力

各アルゴリズムの概要

本章で比較する手法は全て、現在の水位・ポンプ稼動状況および将来の水位予測に従って H_{SC} 時間分の運行計画を立てる。この計画は、予測の範囲内で水位が運用上下限を越えることなく、処理場揚水量の変化量を最小化するよう GA によって最適化される。

GA^{lazy}_{APM} はまずペナルティ加算型 GA により一度運行計画を作成し、その期限である H_{SC} 時間後になるかあるいは予測の誤差により次ステップで運用上下限を満たさないとわかった場合のみ再スケジュールを行うものとし、現在の運用に最も近いと言える。

GA_{APM} はペナルティ加算型 GA により H_{SC} 時間分運行計画を作成し、その最初の 1 ステップ分だけに従う。1 ステップ後は最新の水位データに基づき再スケジュールを行う。

GA_{SCS} は独立制約充足型 GA により運行計画を作成し、1 ステップごと再スケジュールを行う。ここでは降下型を用い、まず $LB_f = 0$ すなわち揚水量を一度も変化させないとして制約充足を試みる。これに失敗した場合 $LB_f = -1$（一回だけ揚水量を変化させる）とし、以降制約充足に成功するまで 1 ずつ目標値を減少させてゆく。

どのアルゴリズムも、計算時間内に制約を満たす解が発見できなかった場合は最も違反の少ないものを採用することとする。これら 3 つのアルゴリズムを比較することにより、本論文の提案である (1) 毎タイムステップでの再スケジューリング, (2) 揚水量平滑化と制約充足の分離が有効であることを示す。

オペレータ

実験に用いたGAの設定は以下の通りである。

- **コード化：** 集団サイズを N_{pop} とする。各個体 S は各施設の H_{SC} 時間後までの運行計画を表す。すなわち、個体のスケジュール S=$[s_1, ..., s_{H_{SC}}]$, 各 $s_t = (a_t^1, ..., a_t^5)$ は t 時間後の各施設の揚水量である。なお、計画の段階では1時間は同じ運用を続け、これはタイムステップ T_{step} には依存しない。よって探索次元数は $5H_{SC}$ となる。

- **目的関数：** 個体に対する目的関数（最大化）は $f(S)$=$-\sum_{t=1}^{H_{SC}}\{1-\delta(a_t^1 - a_{t-1}^1)\}$ すなわち処理場の揚水量変化回数である。なお、$\mid a_t^1 - a_{t-1}^1 \mid \geq 1500\ [m^3/h]$ となった場合は2回分の変化とカウントする。

- **制約違反：** 各個体に対して H_{SC} 時間後までのシミュレーションを行う。施設 i に t 時間後の水位制約違反がシミュレータ上で観測された場合、ペナルティ $p(S)$ に -2^t を加算する。

- **副目的関数：** 制約を満たし（$p = 0$) かつ目的関数 f の等しい二つの個体が生じた場合は、その運用の余裕によって優劣を決定する。施設 i の t 時間後の予測水位を h_t^i と書き、局所的余裕 $m_t^i = \min(h_{MAX}^i - h_t^i, h_t^i - h_{MIN}^i)$ と定義した上で、個体の運用の余裕 $m(S) = \min_{i,t}(m_t^i)$ とする。$m(S)$ が大きいほど、予測値の外れに対する適応が容易であると考えられる。

- **交叉：** 親1を S^*=$[s_1^*, ..., s_{H_{SC}}^*]$, 親2を S^+=$[s_1^+, ..., s_{H_{SC}}^+]$, とするとき、これに対して2点交叉を行う。即ち、$1 \leq t_1 \leq t_2 \leq H_{SC}$ を定め、
  子 S^c=$[s_1^*, ..., s_{t_1-1}^*, s_{t_1}^+, ..., s_{t_2}^+, s_{t_2+1}^*, ..., s_{H_{SC}}^*]$ とする。

- **突然変異：** $1 \leq t' \leq H_{SC}$ を定め、$t' < t$ なる全ての a_t^i について、ポンプ一台を停止または稼動させる変異を確率 p_{mute} で行う。

- **世代交代：** 親 S^1, S^2 のペアに対して、N_{child} の子個体が生成され、それぞれに突然変異が適用される。全ての子個体および親 S^1 のうちで最もよい適応度を持つ個体が親 S^1 の代わりに集団に戻される。

マージンの導入

GA は、流入量の予測に基づいて各個体を水位違反のないように最適化する。しかしながら予測には誤差があるため、運用上下限ぎりぎりの水位を計画すると実際には制約を満たさないことがしばしばある。そこで、計画・解評価の段階では、真の運用上下限よりもさらに $r_{mgn}\%$ 厳しい制約を設ける。すなわち真の水位上限 h^i_{MAX}, 下限 h^i_{MIN} の代わりに上限 $h^i_{MAX} - h_r$, 下限 $h^i_{MIN} + h_r$ を用いることとする。ここで h_r=$(h^i_{MAX} - h^i_{MIN}) \times \frac{r_{mgn}}{100}$ である。

制約違反が大きな損害を与えるほど、また予測が不正確なほどマージン r_{mgn} は大きく取らねばならない。マージンを大きく取るということは、それだけ不自由な水位制約の中で運用を行わねばならないということであり、平滑化を行うことは難しくなる。

再スケジュール頻度

運用計画を与えたのち長い時間それに従い続ければ、予測値と実際の流入量との誤差は蓄積される。逆に頻繁に水位を確認して再スケジュールを行えば予測誤差は最低限で押さえられ、溢れに対するマージンも小さく取ることができる。GA_{APM} および GA_{SCS} は T_{step} 分ごとに再スケジュールを行うが、このタイムステップが短いほど精密な制御ができるということになる。

6.4.6 実験

本アプローチの正当性を確かめる実験

本節では (A) 予測が完全な場合 (B) 過去のデータの平均値を予測とする場合、について GA^{lazy}_{APM}, GA_{APM}, GA_{SCS} の 3 手法を比較する。運行スケジュール範囲 $H_{SC} = 12[h]$, 再スケジュール頻度 $T_{step} = 20[min]$, 突然変異率 p_{mute}=0.03, 集団サイズ $N_{pop} = 50$, 生成子個体数 $N_{child} = 2$ とし、一回のスケジューリングの打ち切り評価回数を 3000, 10000, 30000 とした。これは 1Ghz の CPU でそれぞれ約 1, 3, 10 秒に相当するが、実際の運用上は 1 分程度の計算時間は許容されている。溢れを防ぐためのマージン r_{mgn} は、(A) では 0%（必要ない）, (B) では 4% とした。

実験には家庭排水量の実データ 14 日分を用い、シミュレータで運用した上で処理場の一日あたりの揚水量変化回数、および水位制約違反回数を用いて比較を行う。10 試行の結果の平均値を表 6.3 にまとめる。

表 6.3: 3 つの GA による運用成績の比較：処理場揚水量切り替え回数および各施設での水位制約違反回数、それぞれ一日あたりの平均値

手法	計算時間	(A) 予測が完全		(B) 平均値を用いる	
	(sec)	切替回数	違反	切替回数	違反
GA_{APM}^{lazy}	1	3.08	-	3.85	0.17
	3	3.24	-	4.25	0.085
	10	2.99	-	4.07	0.14
GA_{APM}	1	2.20	-	2.84	0
	3	1.83	-	2.69	0.007
	10	2.00	-	2.57	0
GA_{SCS}	1	1.45	-	1.92	0.028
	3	1.37	-	1.71	0.050
	10	**1.25**	-	**1.67**	0.021

この結果から、以下のような知見を得ることができた。

- マージンを導入することにより、予測が不正確な場合でも 1 日あたりの溢れ率を許容範囲内に押さえることができる。マージンの大小が与える影響については次節で詳述する。
- GA_{APM}^{lazy} と GA_{APM} は全く同じ最適化アルゴリズムを用いているにもかかわらず 1.5 倍ほどの性能差が出ており、毎ステップ再スケジュールを行うことが性能に貢献している。
- GA_{SCS} は GA_{APM} よりも 3〜5 割良い性能を出しており、この問題では独立制約充足がペナルティ加算法よりも有効であることを示している。また、GA_{SCS} は他手法に比べ、計算時間を長く与えることがより明確に性能に貢献している。
- (A) の予測が完全に正確な場合というのは非現実的な設定であるが、予測が正確になるほど性能が向上するであろうということは期待できる。

マージンと再スケジュール頻度の関係を調べる実験

前節の実験 (B) では、再スケジュール期間 $T_{step} = 20\ [min]$ に対して、経験的に最も適当と思われるマージン $r_{mgn} = 4\%$ を所与とした。本節では、この二つのパラメータの性能に与える影響について検証する。なお、本節以降では用いるアルゴリズムを GA_{SCS}, 計算時間を 10 秒 (30000 評価回数で打ち切り) に固定する。表 6.4 は r_{mgn} を $1\%,\ 2\%,\ 4\%,\ 8\%$ とし、T_{step} を $60,\ 30,\ 20,\ 10[min]$ とした場合の揚水量切替回数と制約違反回数をまとめたものである。

表 6.4: 再スケジュール期間 T_{step} とマージン r_{mgn} を変化させた場合の運用成績の比較

T_{step}	$r_{mgn} = 1\%$		$r_{mgn} = 2\%$		$r_{mgn} = 4\%$		$r_{mgn} = 8\%$	
[min]	切替数	違反	切替数	違反	切替数	違反	切替数	違反
60	1.84	2.21	1.87	1.60	2.09	0.92	2.34	0.20
30	1.74	0.98	1.62	0.55	1.68	0.15	1.82	0.007
20	1.55	0.65	1.55	0.23	1.67	0.021	1.77	0
10	1.48	0.09	**1.52**	**0.007**	1.64	0	1.80	0

この結果、概ねマージンを小さくするほどに性能は良くなり、一方溢れ率は大きくなることがわかる。再スケジュールを頻繁にやれば小さなマージンでも溢れ率は十分小さくなり、1 日あたり 1.52 回（あふれ率 0.7%) の切り替え回数は、同じ条件下での先行研究 [青木] よりも良好な制御である。

予測の精度による影響を調べる実験

上水道配水系においては、浄水需要に対する供給不足は絶対に避けねばならないために GMDH やニューロなどの手法を用いて高精度の予測が行われている。一方家庭排水に対する予測の研究は従来あまり盛んには行われてこなかった。本論文の実験環境 (B) でも高精度な予測は行わず、過去のデータの時刻ごとの平均値を用いるにとどめている。しかしながら予測を完全に正確なものと仮定した環境 (A) に比べるとその性能差は歴然としており、予測をより正確にすることがコスト削減に大きく貢献することが予想される。

本節では、下水道送水系の家庭排水流入量予測の研究を促す願いも込めて、予測の精度が性能にどの程度影響を与えるかを調べる。実際の流入量 o^{real}, データの平均値 o^{av}, 予測値

o^{prd} に対して、予測の精度 r_{acc} を $\frac{o^{prd}-o^{av}}{o^{real}-o^{av}}$ と定義する。$r_{acc}=1$ のとき予測値は正確に実際の値であり (A)、$r_{acc}=0$ のとき平均値を用いる場合と同じ (B) になる。r_{acc} の値をさまざまに変化させた場合の結果は表 6.5 に示す通りである。再スケジュール期間 $T_{step}=20[min]$, また予測が正確なほどマージンは小さくしてよいため $r_{mgn}=4\times(1-r_{acc})\%$ とした。

表 6.5: 予測の正確さ r_{acc} による運用成績の影響

精度 r_{acc}	マージン r_{mgn}	切替回数	溢れ率
0	4%	1.67	0.021
0.25	3%	1.60	0.007
0.50	2%	1.44	0
0.75	1%	1.37	0
1	0	1.25	-

この結果、予測の精度があがるに従いマージンを小さくしても溢れ率は上昇せず、また処理場の揚水量切り替え回数を少なく抑えることができていることがわかる。この事実は、下水道送水系に対しても予測の精度を上げる研究を進めることがコスト削減に大きく貢献することを示唆している。

6.4.7 実験のまとめ

前述の 3 つの実験から、下水道の制御に関して以下の結論を得ることができた。

- 毎タイムステップごとに再スケジュールするというアプローチが性能を向上させた。
- 揚水量平滑化と制約充足を分離するというアプローチが性能を向上させた。
- 再スケジュール間隔を短くすればマージンを小さくすることができ、揚水量平滑化に貢献できる。
- 予測は正確なほど性能が向上するが、精度の低い予測であっても実用に耐える性能が出せる。

6.5 まとめ

現実の最適化問題には複雑かつ多様な制約条件が存在することが多く、これを即座に充足させるオペレータを設計することは非常に困難である。そのため制約違反の度合いをペナルティ関数として数値化し、これを最小化することで徐々に制約を満たす解を得ようとするのが一般的である。負のペナルティを評価値に加算し、目的関数の最適化と制約の充足を同時に行う**ペナルティ加算法**には多くの適用例がある。

本章では、目的関数最適化と制約充足とが対立的な場合にペナルティ加算法を用いると、制約を満たしただけの安易な解に陥る恐れが大きいことを指摘した。これは、目的関数が良い部分空間では制約を満たすことが難しいためにペナルティが大きくなり、評価値としては悪くなってしまう可能性が高いためである。GA などの確率的最適化手法では、真に質の高い解を含む領域よりも平均的な評価値が良い領域に探索が重点化されてしまい、制約を満たしかつ真に良好な解を得ることは困難になるのである。

ペナルティ加算法の問題点は、目的関数の最適化と制約の充足という対立する二つの目的を同時に最適化しようとする点にあった。本章ではこの二つの目的を別の軸で捉え、目的関数の下界値を定めた上で制約充足を行う**独立制約充足**を採用した。これは、部分空間を独立に探索するという意味では生得分離モデル [池田 2002] の応用と捉えられ、多目的アプローチという意味では ϵ 制約法 [Bhaskar 2000] の解釈と捉えることができる。

下水道送水系の制御は、処理場揚水量平滑化と水位制約充足という対立する目的を持つ実問題である。本章では不確実な流入量予測に対する適切な対策を取ると同時に、最適化アルゴリズムに独立制約充足型 GA を用いることで従来の運用成績を上回る制御法を確立することができた。

これらの結果は、独立制約充足が取り扱いの困難な制約条件を持つ実問題への GA の適用可能性を大幅に拡げるという主張を強く支持するものであると言える。独立制約充足戦略の適用可能クラスおよび設計指針、パラメータの設定法については付録 D に示す。

第7章　おわりに

研究成果のまとめ

本研究では、実用に向けた最適化を達成するために、従来の研究が想定してこなかったクラスにまで対象領域を拡げ、従来の研究が定めてきた形式的な目標を柔軟に見直してきた。「制約条件をどう扱うか」「複数の目的関数をどう扱うか」「世代交代モデルはどう設計するか」「交叉はどう設計するか」という 4 つのテーマに着目し、GA を用いた最適化の流れの中で各々に何が求められているのかを考慮した上で従来法が達成できていない課題に取り組んだ結果、それぞれに簡潔で有力な手法を提案することができた。これらの成果と意義を総括する。

- **交叉はどう設計するか**

  GA には TSP における EAX など問題に特化した優秀な交叉があるが、このような交叉の設計には大きな開発コストが必要となる。一方、問題への依存度が低い汎用的な交叉は、親の要素を混ぜ合わせるという形式的な目標にとらわれており、親の良い所だけを子に引き継ぐという交叉本来の目的を達成できていなかった。

  本研究では、この形式的な目標を柔軟化し、親の良い要素を少しずつ取っていく**いいとこどり交叉**を提案した。いいとこどり交叉は多段階を踏んで近傍を辿り相手親に近づくという MSXF のアイデアと相手親の良い要素 1 つだけを取り込むという EAX-1AB のアイデアを用いた交叉である。いいとこどり交叉は相手親に近づく近傍を定義することで容易に構成できるだけでなく、一様交叉よりも高性能でかつパラメータに頑健であることが確認された。

  EAX は TSP における要素の一様交叉であると捉えることができ優秀な性能をあげているが、これをいいとこどり交叉に置き換えることでさらにその潜在的な性能を引き出すことに成功した。これらの結果から、GA の本質的なオペレータである交叉

を低コストで高性能に設計する方法論の一つが築けたと言える。

- **世代交代モデルはどう設計するか**

  GA は他の最適化アルゴリズムに比べて多峰性関数における局所解にとらわれにくいとされ、大谷構造の下では効率的に働くことが知られている。しかしながら、実問題の多くには大域的な多峰性が存在しており、場合によっては最適解よりも有力な局所解の方が見つかりやすい UV 現象が生じる。

  本研究ではこの原因となる UV 構造を分類し、複数の問題で UV 現象を生じることと UV 構造の有無の相関を確認した。さらに UV 構造を克服して最適解を得るために、集団の初期化や交叉・選択を局所化した**生得分離モデル**を提案し、GA の適用可能クラスを広範化することに成功した。

  生得分離モデルは非常に困難なベンチマークとして知られる JSP に適用され、他手法を凌駕する性能をあげた。生得分離モデルは大域的多峰性における UV 現象を回避することができるだけでなく並列計算との親和性も高く、実問題のための世代交代モデルとして非常に魅力的であると言える。

- **複数の目的関数をどう扱うか**

  多目的最適化では、何らかの合理性を持った複数の解を提示する必要があるが、従来は主に Pareto 最適解をまんべんなく求めることを目標とされてきた。しかしながら、Pareto 最適解の中にはユーザにとって全く必要のない解も含まれ、これを探索することは時間の無駄であるだけでなく時には DRSs を生じて探索を頓挫させる原因となることがわかった。

  本研究では、ユーザの許容するトレードオフ比の上下限を定める行列 α_{ij} を用い、その範囲内だけで探索を行う **α-domination 戦略**を提案した。これにより、多目的最適化は Pareto 最適解を求めなければいけないという目標を柔軟化し、ユーザにとって選択しうる有益な解だけを高速かつ安定に求める最適化戦略を提案することができた。

- **制約条件をどう扱うか**

制約違反解を直ちに修正するのが困難な問題に対しては、ペナルティ加算法が頻繁に採用されてきた。しかしながら、ペナルティ加算法では制約を充足することと本来の評価値を最適化することが対立的となる場合が多く、制約は充足されるが質は劣った解を安易に発見するにとどまる可能性が高い。

本研究ではこれがUV現象によるものであり、ペナルティを加算することで景観にUV構造が生じていることを突き止めた。その上で、目的関数の最適化と制約の充足という対立する二つの目的を同時に最適化するのではなく別の軸で捉え、目的関数の下界値を定めた上で制約充足を行う**独立制約充足**を提案した。これにより、GAの適用可能クラスを広範化し、困難な問題でも解の質を犠牲にすることなく制約を充足するための戦略が提案できた。

独立制約充足は下水道送水系の制御に適用され、ペナルティ加算法によるGAを含めた他手法を上回る性能をあげた。独立制約充足は並列計算との親和性も高く、複雑な制約条件を持つことの多い実問題に対する非常に有力な選択となりうると言える。

本研究は各々のテーマだけに視野を限定せず、流れ全体を俯瞰して大域的な最適化を目指したものであり、これらの成果はGAを用いた問題解決の実用化に大きく寄与するものと考えられる。提案された4つの手法・戦略は簡潔なものであり、またその適用可能クラスと設計指針・パラメータ設定法を具体的に提供することで、一般ユーザにとってGAを採用しやすいものにすることができたと考えている。

今後の展望

GAが扱う最適化問題は静的な評価値を持つものが想定されることが多く、本研究も例外ではない。しかしながら現実の問題には観測ノイズが入ったり、環境の変化によって同じ解であっても毎回違った評価値が得られてしまうといった事態は容易に想定できる。今後GAを実問題に適用するにあたっては、このような静的でない評価値に対する適切な処置が求められる。

GAはSAなどの他の最適化アルゴリズムに比べて多数の要素を持ち、設計が困難であると同時にプログラムの実装の困難さもまたハードルを高くしている一因となっている。

そのため、GA の研究者には、一般ユーザが利用しやすいようにパッケージやソースを提供することが求められる。

マルコフ決定過程（Markov Decision Process; MDP）下での政策の最適化は非常に広い応用範囲を持つ研究領域であり、強化学習と呼ばれる接近法が一般的である。しかし、一つの政策を一つの解とし、一定期間の平均報酬を評価値とすることでこれを最適化することも可能であり、いくつかの適用がなされている [武市 2000]。GA の交叉オペレータと親和性の高い解表現として政策を事例ベースで表現するアプローチは非常に有望であり、本研究で得られた知見を用いて困難な MDP 下での政策獲得可能にすることは、新しい領域を開拓する上で非常に有用と考えられる。

謝辞

本論文をまとめるにあたり多大なるご指導をいただきました小林重信教授、特に第 5 章多目的最適化に関して助けていただきました喜多一教授に深く感謝致します。また、研究・生活全般にわたりお世話になりました佐久間淳さん、佐野泰仁さん、中村一尊さん、佐塚直也さん、小林研の皆さんに感謝の意を表します。私を育ててくれた両親に、良き友人井上浩二君、野口基樹君、親愛なる弟に、感謝します。最後に、一番大切なひと、内藤有花さんに感謝と愛情を込めてこの論文を贈ります。

公表論文

＜学術論文＞

1. 池田, 小林： GA の探索における UV 現象と UV 構造仮説, 人工知能学会誌 Vol.17 No.3G (2002)
2. 池田, 小林： 生得分離モデルを用いた GA と JSP への適用, 人工知能学会誌 Vol.17 No.5A (2002)
3. 池田, 青木, 長岩, 小林 : 独立制約充足による最適化と送水制御への適用, 人工知能学会誌（投稿中）

＜国際会議＞

1. Ikeda, K. and Kobayashi, S: GA based on the UV-structure Hypothesis and Its Application to JSP, 6th Parallel Problem Solving from Nature (PPSN　), pp.273-282 (2000)
2. K.Ikeda, H.Kita, and S.Kobayashi : Failure of Pareto-Based MOEAS: Does Non-Dominated Really Mean Near to Optimal?, Congress on Evolutionary Computation (CEC2001) pp. 957-962 (2001)
3. K.Ikeda, S.Kobayashi : Deterministic Multi-Step Crossover Fusion : A Handy Crossover Composition for GAs, 7th Parallel Problem Solving from Nature (PPSN　), pp.162-171 (2002)

＜国内の学会＞

1. 池田, 小林： UV 構造仮説に基づく GA の設計と JSP への適用, 第 27 回 知能システムシンポジウム (SICE) pp. 43-48 (2000)

2. 池田, 喜多, 小林： Pareto 生存戦略を用いた多目的進化計算の破綻するとき, 第 11 回 インテリジェント・システム・シンポジウム Fuzzy, Artificial Intelligence, Neural Networks and Computational Intelligence(FAN01) pp.263-266 (2001)
3. 池田, 小林 : 履歴を用いた状態評価関数の序列推定 , 第 29 回 知能システムシンポジウム (SICE) pp. 109-114 (2002)

関連図書

[Aarts 94] Aarts, E. , van Laarhoven, P. , Lenstra, J. and Ulder, N. : A Computational Study of Local Search Algorithms for Job-shop Scheduling, *ORSA J. on Comput, Vol. 6, No. 2*, pp. 118-125 (1994).

[Applegate 91] Appleegate, D. and Cook, W. : A Computational Study of the Job-shop Scheduling Problem, *ORSA J. on Comput. , Vol. 3, No. 2* , pp. 149-156 (1991).

[Bäck 96] Thomas Bäck. Evolutionary Algorithms in Theory and Practice, *Oxford university press* 1996

[Bhaskar 2000] Bhaskar V, Gupta SK, Ray AK. Applications of multiobjective optimization in chemical engineering. *Reviews in Chem Eng 2000*; 16(1): pp. 1-54 (2000)

[Boese 94,95] K. Boese, "Cost versus Distance in the Traveling Salesman Problem," *Tech.Rep. TR-950018, UCLA CS Department*, (1995)

[Brucker 94] Brucker, P., Jurisch, B. and Sievers, B.: A Branch & Bound Algorithm for the Job-shop Scheduling Problem, *Discrete Applied Mathematics*, Vol.49, pp.107-127 (1994).

[Corne *et al.* 2000] D. W. Corne, J. Knowles, M. J. Oates : The Pareto Envelope-Based Selection Algorithm for Multiobjective Optimization, *PPSN6*, pp. 839-848 (2000)

[Deb 2000] Deb, K. : An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering* (2000)

[Deb *et al.* 2000] K. Deb, S. Arawal, A. Pratap, and T. Meyarivan : A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization : NSGA-II, *Parallel Problem Solving from Nature(PPSN)6, pp. 849-858 (2000)*

[EP5] Ralf Salomon, Performance Degradation of Genetic Algorithms under Coorinate Rotation, *Evolutionary Programming 5*, pp.157

[FP63] R. Fletcher and M.J.D. Powell. A rapidly convergent descent method for minimization. *Computer Journal, 6*:163-168, 1963

[Fonseca 93] C.M. Fonseca and P.J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. Proceedings of the Fifth International Conference on Genetic Algorithms, pp 416-423 (1993)

[Francisco 2000] Francisco O. Jr. , James R.S. : A Genetic Algorithms with a modified Desirability Function Approach to Multiple Response Optimization, *Electronic Proceedings for IIE 2002 Annual Conference* , No.2125 (2000)

[Giffler 60] Giffler, B. and Thompson, G. : Algorithms for Solving Production Schedulng Problems, *Oper. Res. , Vol. 8*, pp. 487-503 (1960).

[Glover 89] Glover, F.: Tabu search I, ORSA Journal on Computing, Vol. 1, pp.190-206 (1989), Glover, F.: Tabu search II, ORSA Journal on Computing, Vol. 2, pp.4-32 (1989)

[Goldberg 87] Genetic Algorithms and Evolutionary Strategies in Engineering and Computer Science, chapter Adaptive Niching Via Coevolutionary Sharing. (1987)

[Harvey 99] Harvey K.B., Pettey C.C.: The outlaw method for solving multimodal functions with split ring parallel algorithms, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp.274-280 (1999)

[Holland 75] Holland, J.H.: Adaptation in natural and artificial systems, The University of Michigan Press, Ann Arbor (1975)

[Ikeda 2000] Ikeda, K. and Kobayashi, S. : GA based on the UV-structure Hypothesis and Its Application to JSP, *6th Parallel Problem Solving from Nature*, pp.273-282(2000)

[Ikeda *et al.* 2001] K. Ikeda, H. Kita, and S. Kobayashi : Failure of Pareto-Based MOEAs: Does Non-Dominated Really Mean Near to Optimal? *Congress on Evolutionary Computation* pp. 957-962 (2001)

[Jelasity 98] Jelasity, M. and Dombi, J.: "GAS, a concept on modeling species in genetic algorithm", *Artificial Intelligence 99*, pp. 1-19 (1998).

[Kirkpatrick 83] Kirkpatrick, S., Gelatt, C. D., Vecchi, M.P. : Optimization by simulated annealing, Science, Vol. 220, No. 4598, pp.671-680 (1983)

[Kita *et al.* 1996] H. Kita, Y. Yabumoto, N. Mori and Y. Nishikawa: Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm, *PPSN4*, pp. 504-512 (1996).

[Knowles 99] J.Knowles, D.W. Corne : The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation, *Congress on Evolutionary Computation*, pp. 98-105 (1999)

[Laumanns *et al.* 99] M. Laumanns, G. Rudolph, and H.P. Schwefel : A Spatial Predator-Prey Approach to Multi-objective Optimization : A Preliminary Study, *PPSN5*, pp. 241-249 (1999)

[Mühlenbein 91] Mühlenbein, H., Schomisch, M. and Born, J.: The parallel genetic algorithm as function optimizer, *Parallel Computing*,17, pp.619-632(1991)

[Markon 2001] Sandor Markon, Dirk V.Arnold, Thomas Bäck, Thomas Beielstein, and Hans-Georg Beyer : Thresholding - a Selection Operator for Noisy ES, *Congress on Evolutionary Computation*, pp. 465-472 (2001)

[Nagata 2000] Nagata, Y. : Genetic Algorithm for Traveling Salesman Problem using Edge Assembly Crossover: its Proposal and Analysis, *Doctoral thesis* (2000)

[Nagata 97] Nagata, Y. and Kobayashi, S. : Edge Assembly Crossover: A High-power Genetic Algorithm fot the Traveling Salesman Problem, *Proceedings of the 7th International Conference on Genetic Algorithms*, pp. 450-457 (1997)

[Nowicki 93] Nowicki, E. and Smutnicki, C. : A Fast Taboo Search Algorithm for the Job Shop Problem, Institute of Engineering Cybernetics, Technical University of Wroclaw, *Poland. , Vol. Preprinty nr 8/93* (1993).

[Ono 96] Ono, I. , Yamamura, M. and Kobayashi, S. : A Genetic Algorithm for Job-shop Scheduling Problems Using Job-based Order Crossover, *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pp. 547-552 (1996).

[Ono 97] Ono, I. "A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover",*Proc. of 7th Intr. Conf. on Genetic Algorithms* , pp 246-253, 1997

[Ono 98] Ono, I. and Kobayashi, S. : A Genetic Algorithm Taking Account of Characteristics Preservation for Job-shop Scheduling Problems,*Proceedings of the 5th conference on Intelligent Autonomous Systems,* pp. 711-718 (1998).

[Padberg 91] Padberg, M. and G.Rinaldi: A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems, *SIAM Review, Vol.33, No.1* pp.60-100 (1991)

[Reinelt 94] G.Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications. Vol.840 of Lecture Notes in Computer Science, Springer-Verlag (1994)

[Sakuma 2000] J. Sakuma, S. Kobayashi : Extrapolation-Directed Crossover for Job-shop Scheduling Problems ; complementary combination with Job-based Order Crossover, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp 973-980 (2000).

[Sano 2002] Y. Sano, H. Kita, H. Kaji and M.Yamaguchi: Optimization of Dynamic Fitness Function By Means Of Genetic Algorithm Using Sub-Populations, *Proc. of SEAL 2002*

[Satoh 96] H.Satoh, M.Yamamura, and S.Kobayashi : Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation, *Proc. of IIZUKA*, pp.494-497 (1996)

[Schleuter 97] Gorges-Schleuter, M. : Asparagos96 and the Traveling Salesman Problem, *Proc. of the 1997 IEEE International Conference of Evolutionary Computation,* pp.171-174 (1997)

[Shimodaira 99] Hisashi Shimodaira: A Diversity Control Oriented Genetic Algorithm (DCGA): Development and Experimental Results, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 603-611 (1999)

[Smith 93] Smith, R.R., Forrest, S., Peterlson, A.S. : Searching for diverse cooperative populations with genetic algorithms, *Evolutionary Computation*, Vol.1, No.2, pp. 127-149 (1993).

[Takahashi 99] Takahashi, O. "A Distance Dependent Alternation Model on Real-coded Genetic Algorithms" *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 619-624, 1999

[Tsutsui 97] S. Tsutsui, Y. Fujimoto and A. Ghosh: Forking GAs: GAs with Search Space Division Schemes, *Evolutionary Computation, MIT Press, Vol. 5, No. 1*, pp.61-80(1997)

[Watson 99] Jean-Paul Watson, "A performance assessment of modern niching methods for parameter optimization problems" *GECCO 99*, pp 702-709

[Yamada 95,96] Yamada, T. and Ryohei, N. : Job-shop Scheduling by Simulated Annealing Combined with Deterministic Local Search, *Kluwer Academic Publishers, MA, USA* (1995); 情報処理学会誌 Vol.37 No.4 , pp.597-604(1996).

[Yamada 96,97] Yamada, T. and Ryohei, N. : Scheduling by Generic Local Search with Multi-Step Crossover, *4th Parallel Problem Solving fron Nature* , pp. 960-969 (1996); 情報処理学会誌 Vol 38 No.6, pp.1126-1138(1997).

[Yamada 95] Yamada, T. and Nakano, R.: A GA with multi-step crossover for job-shop scheduling problems, *Proc. of Int. Conf. on GAs in Engneering Systems: Innovations and Applications (GALESIA) '95*, pp.146-151 (1995)

[Zitzler 2002] A Tutorial on Evolutionary Multiobjective Optimization; Workshop on Multiple-Objective Metaheuristics (2002)

[安達 2001] 安達, 佐藤, 小林 : 航空機スケジューリング問題への遺伝アルゴリズムの応用, 電子情報通信学会論文誌 D-I vol. J84-D-I, no.6, pp888-895 (2001)

[喜多 98] 喜多 : 実数値 GA のための正規分布交叉の多数の親を用いた拡張法の提案, システム/情報合同シンポジウム *98*, pp. 157-162 (1998)

[喜多 99] 喜多, 小野, 小林 : 実数値 GA のための正規分布交叉に関する理論的考察, 計測自動制御学会, *Vol.35, No.11*, pp. 1333-1339 (1999)

[栗栖 94] 栗栖, 西谷, 舘, 安達 : 数理計画法とヒューリスティック法を組み合わせた動的配分計画法の上水道運用計画問題への適用, 計測自動制御学会論文集, Vol.30, No.2, pp. 198-207 (1994)

[高橋 2001] 高橋, 木村, 小林 : 交叉的突然変異による適応的近傍探索 人工知能学会誌, *Vol.16, No.2C*, pp.175-184. (2001)

[坂本 2000] 坂本, 黒川, 佐野, 山田, 芦木, 結城 : GA による送水計画の近似的最適化手法, 電学論 D, 120 巻 8/9 号 (2000)

[小野 97] 小野: 形質遺伝を重視した遺伝的アルゴリズムによる最適化; 博士論文 (1997)

[小野 98] 小野, 小林: Inter-machine JOX に基づく JSP の進化的解法 人工知能学会誌 Vol.13 No.5 pp.780-790(1998).

[森 96] 森, 吉田, 喜多, 西川 :遺伝アルゴリズムにおける熱力学的選択ルールの提案; システム制御情報学会誌, *Vol.9, No.2*, pp.82-90.

[青木] 青木，木村，長岩，小林 : 分散強化学習による下水道送水系の制御, 電気学会論文誌 （掲載予定）

[池田 2002] 池田, 小林: GA の探索における UV 現象と UV 構造仮説, 人工知能学会誌 Vol.17 No.3 (2002)

[樋口 2001] 樋口, 筒井，山村: 実数値 GA におけるシンプレクス交叉の提案， 人工知能学会誌， Vol. 16, No. 1, pp. 147-155 (2001).

[武市 2000] 武市, 小林: GA による並列二重倒立振子の振り上げ安定化制御; 第 27 回知能システムシンポジウム（計測自動制御学会）, pp.133-138 (2000).

A. いいとこどり交叉の適用可能クラス、設計指針とパラメータ設定

適用可能クラス

実数値最適化問題に対しては、SPX や LUNDX+EDX 等の優秀な汎用交叉が存在するため、いいとこどり交叉を敢えて用いる必要はない。一方離散の組み合わせ最適化においては、対象がビットストリングのように単純なものから、JSP や下水道スケジュールなどの複雑なものまで、広く第一選択となりうる。解が実数値パラメータと離散パラメータを共に含む場合はそれぞれの重みと特性により何が適当かは異なるが、一般に実数値パラメータ間に大きな依存関係がない場合はいいとこどり交叉が適用可能である。

いいとこどり交叉が一様交叉などと比べて効果を発揮するのは比較的生成子個体数が多く取れる場合であることは本文 3.4.2 節の実験からもわかる。生成子個体数が 10 程度であれば、1 ステップのいいとこどり交叉、即ち通常の CCM モデルを用いるのが適切であろう。

設計指針とパラメータ設定

いいとこどり交叉を構成するにあたっては、解 x から解 p_2 に近づく近傍集合 $\mathcal{N}_{\rightarrow p_2}(x)$ が μ 個生成される必要があり、またパラメータ μ および k_{max} を定める必要がある。

いいとこどり交叉の性能を最も左右するのは $\mathcal{N}_{\rightarrow p_2}(x)$ の構成の仕方であり、これは SA や TS などの探索が近傍集合 $\mathcal{N}(x)$ の構成法によって大きく性能を変えるのと同様である。一方、パラメータ μ および k_{max} については、$5 \leq \mu$ および $3 \leq k_{max}$ 程度を満たしていればさして敏感になる必要はない。k_{max} は近傍の設定に影響されて変更する必要が生じる。例えば、相手親に対して 1/3 程度近づくような近傍を用意したのならば、$4 \leq k_{max}$ とすることには意味がない。一方 μ については計算資源に影響されて変更する必要が生じる。これは μ を大きくするほど計算量は大きく、解の精度は上がることが予想されるからである。

もし、対象とする問題にすでに通常の近傍 $\mathcal{N}(x)$ および距離 $d(x, y)$ が導入されている場合は $\mathcal{N}_{\rightarrow p_2}(x)$ はただちに構成できる。これらが導入されていない場合は、一旦これらを導入したのちに $\mathcal{N}_{\rightarrow p_2}(x)$ を構成するアプローチと、ただちに $\mathcal{N}_{\rightarrow p_2}(x)$ 自体を構成するア

プローチが考えられる。TSP や QAP のように単純な制約を持つ問題では後者のアプローチが手っ取り早いが、JSP やフロアプラン問題などただちに $\mathcal{N}_{\rightarrow p_2}(x)$ を構成することが困難な場合は前者のアプローチを取る方が結局開発コストは小さくなる。

B-1. ISM の設計指針とパラメータ設定

ISM を実装する際の設定に関する指針を、ISM の各要素ごとに記す。ISM の適用可能クラスに関しては本文中に記してある。

設定 1．小集団の数、各集団の個体数

各集団の個体数に関しては、少なすぎると GA の集団探索の利点を生かせないし、多すぎると進化に時間を要する。ISM は小集団に分けることで多様性を維持するので、各集団の個体数を大きくすることで多様性を維持しなければいけない場合はむしろ ISM よりも従来の GA がうまくいくと考えられる。

集団数に関しては、多すぎるといくつもの集団が同じ谷を探索することになり効率が悪い。逆に集団数が少なくても再初期化を繰り返すことで最適解を発見してくれることが期待できる。

集団の数と各集団の個体数は本質的には問題に依存するが、どちらも数十にすることを推奨する。

設定 2．集団内での交叉・淘汰

交叉・突然変異については従来と同じでよいが、小さな領域から開始される GA であるため、景観に応じて探索領域を広げられるオペレータの方が望ましい。

集団内の世代交代モデルは従来と同じでよいが、頑健かつ多様な探索を行うという観点から、実数値問題では DDA, 組み合わせ問題では CCM モデルを推奨する。

設定 3. 小さい領域での初期化

ISM にとって最も本質的な設定は、初期化領域の決定である。初期化領域が大きすぎると、従来の GA と変わらずにいくつもの谷にまたがって解が生成されてしまうため、UV 現象が生じるリスクを十分回避できない。一方初期化領域が小さすぎると、各谷の中でその谷底に到達することすらできず、取るに足らない局所解にとらわれてしまう可能性が生じる。事前に問題の景観について、谷のおおまかな数や谷の内部の局所解の有力さがわ

かっている場合はこれらを利用して適当な初期化領域を決定することができる。これらが未知な場合は、以下の指針に基づきパラメータの調整を行うことを推奨する。

1. 実数値関数最適化問題の場合、半径 r の超球または一辺 r の超立方体の中の一様分布により初期化領域を決定する。r は定義域の半分、1/10、1/100 等を用いる。または、全探索空間の 1/10、1/100 等を占めるように決める。
2. 組み合わせ最適化問題の場合、問題に固有な距離を導入して、ある点から半径 r の距離内の分布により初期化領域を決定する。または一つの点から突然変異オペレータを n 回かけた領域を用いる。
3. ISM の試行により、ほとんど全ての集団が異なる解に収束している場合、谷の数が多いか、谷の中に有力な局所解が多数存在することが考えられる。この場合、初期化領域パラメータ r, n を大きくすることで、より広い領域をカバーする探索を行う。
4. ISM の試行により、多くの集団が同じ解に収束している場合、探索の局所化が不十分であることが考えられる。この場合、初期化領域パラメータ r, n を小さくすることで、より小さい領域から探索を開始する。

設定 4. 同じ場所を探索しない

このオプションは、有力な局所解が存在して多くの小集団がそこだけを探索しているときに探索効率化の為に用いられる。この場合は問題に適当な距離を導入し、2 つの集団内のベスト個体間の距離が十分小さいとき排除を行う。この閾値は、最適解・有力局所解同士の距離よりも十分小さくとらないと別の谷を探索している集団を排除してしまうことになる。

設定 5. 収束条件

各集団の探索打ちきり条件は従来の GA と同様でよい。全集団の中で相対的に評価値が劣っているのに、進化が長期間滞っている集団はその解に収束したとみなす。または、小集団全体の多様性が十分失われたとき収束したとみなす。

ISM の打ちきり

ISM は小集団が独立に探索し、それぞれが収束・重複して取り除かれるたびに新しい小集団が生成されるため、ISM の一試行自体が小さな GA の多試行を内包していると考えることができる。ゆえに、各小集団の再生成設定が適当になされていれば ISM 全体をリセットする必要はなく、一試行を時間の許す限り続けることで十分無駄のない探索ができる。

B-2. I_2 距離

JSP の解空間の計量として Inter Index 距離 (I_2 距離) を採用する。I_2 距離は遷移オペレータであるスワップによって定義される距離で、距離空間の定義を満足する計量である。以下にその定義を示す。

s_a, s_b をスケジュールとする。ここで $|M|$ は s の機械数, $|J|$ は s の仕事数である。ここで $o(p,q)$ を仕事名 p 機械名 q の作業とする。j_i を同じ仕事名を持つ作業の集合とする. i は仕事名である。つまり $j_i = \{o(i,k)|k = 1, ..., M\}, s = \{j_k|k = 1, ..., J\}$ である。

さらに、l を作業から、その作業が存在する遺伝子座名 (実行される順番) を返す関数とする。ここで、j_i^a, j_i^b の I_{2j} 距離 $I_{2i}(s^a, s^b)$ を以下のように定義する。

$$I_{2i}(s^a, s^b) = \sum_{k=1}^{M} |l(o^a(i,k)) - l(o^b(i,k))|$$

さらにスケジュール s_a, s_b 間の I_2 距離を以下のように定義する。

$$I_2(s^a, s^b) = \sum_{k=1}^{J} I_{2k}(s_a, s_b)$$

B-3. GTb 法

GTb 法は、現スケジュールに対してよりよいスケジュールを逐次的に作り直す手法である。決定途中の状態に対して

C : 実行可能作業集合 (C$\neq \emptyset$)

m_c：c∈C に対し、c が行われる機械

g_m：機械 m で、予定スケジュール上は次に最も早く処理すべきだとされている作業 (g_m∈C とは限らない)

St(c)：作業 c∈C を開始できる最早時間

Et(c)：作業 c∈C を終了できる最早時間

t_m：機械 m が最後に処理を終了した時刻

としたとき、

Phase1.

存在するなら任意の g_m ∈C s.t. t_m=St(g_m) を実行

↩Phase 1.

Phase2.

C_2:={c∈C|Et(c)≤St(g_{m_c})} として

もし $C_2 = \emptyset$ ならば ↪Phase 3.

else 任意の c_0 s.t. St(c_0)=$\min_{c \in C_2}$St(c) を実行して

↩Phase 1.

Phase3.

C_3:={$g_m \in C$} として

もし $C_3 = \emptyset$ ならば ↪Phase 4.

else 任意の c_0 s.t. Et(c_0)=$\min_{c \in C_3}$Et(c) を実行して

↩Phase 1.

Phase4.

任意の c_0 s.t. Et(c_0)=$\min_{c \in C}$Et(c) を実行して

↩Phase 1.

GTb 法の計算量のオーダは GT 法に等しい。また Phase2, 3, 4 で実行する作業を選択する際に条件をゆるめることで生成解の多様性を操作できる。

C-1. α-domination 戦略の適用可能クラス、設計指針とパラメータ設定

適用可能クラス

α-domination 戦略を用いる動機および利点には主に以下の 3 つがある。

1. **必要な範囲だけが欲しい**

   実問題では、Pareto 最適解が全て必要な局面は殆どない。概ね一定の範囲内、あるいは一定のトレードオフ比内の解だけが必要となる。一定の範囲内の解だけを求めることを直接的に実装するのは本文中に記述したように困難が伴うため、α-domination を用いて一定のトレードオフ比内だけを求めるのが好ましい。

2. **探索を停滞させたくない**

   多くの目的関数がある場合、単純な問題であっても一つの解を dominate できる解を発見する可能性は小さくなる。このような場合探索の速度は極端に遅くなり、DRSs が生じれば探索は完全に破綻する。こういった事態を避けるために、比較的 $\alpha_{i\neq j}$ の小さい α-domination を適用することは効果的である。

3. **探索を並列化したい**

   実問題で多目的最適化を行う場合、計算に大きなコストがかかることが少なくない。多目的最適化では島モデル等の分散法が適切ではないため、並列化が困難なクラスの一つとなっている。α-domination を用いれば、Pareto 集合のうち $0 \leq \frac{\Delta f_1}{\Delta f_2} < 0.5$ となる部分、$0.5 \leq \frac{\Delta f_1}{\Delta f_2} < 2$ となる部分、$2 \leq \frac{\Delta f_1}{\Delta f_2}$ となる部分 の 3 つに分けての並列探索などが容易かつ自然に行える。

設計指針とパラメータ設定

α-domination 戦略を用いる動機が (1.) による場合、パラメータ α_{ij} の設定は容易であり、ユーザの必要とするトレードオフ上下限を指定すればよい。ユーザにとってトレードオフ比が不明な場合なペア $f_i,\ f_j$ に対しては $\alpha_{ij} = \alpha_{ji} = 0$ とすれば支障は生じない。

動機 (2.) による場合、非常に小さな $\alpha_{i\neq j}$ を用いるだけでも DRSs を効果的に回避し、十分探索を高速化させていることが本文 5.4 節の実験からわかる。あるペアに対してトレードオフ比の上下限を限定してしまうことが現実の目標に即さない可能性がある場合は、$\alpha_{i\neq j}$

を 0 に近い値、あるいは 0 にしてもよい。

動機 (3.) による場合、目的関数が m つあり、各目的関数のペアを n 分割すると、n^m 回の試行が必要になる。並列台数にあわせて n を決定する必要がある。分割の仕方については適宜定めるが、概ね各領域が同程度の重要性を持つように設定する。

C-2. 第 5 章の定理等の証明

命題： P_3 のパレート最適集合 $V^* = X^* \times Y^*$.

証明：

$x^- \in X \setminus X^*$ および $y^0 \in Y.$ とする。

$x^- \notin X^*$ であるから、 $\exists x^+ \in X$ s.t. $x^+ \prec x^-$.

すなわち、 $\forall i,\ f_i(x^+) \leq f_i(x^-),$ かつ $\exists i,\ f_i(x^+) < f_i(x^-), 1 \leq i \leq K.$

ゆえに $(x^-, y^0) \notin V^*,$ であり $(x^+, y^0) \in V$ に dominate される。なぜなら、

$$1 \leq \forall i \leq K,\ o_i(x^+, y^0) \leq o_i(x^-, y^0)$$

$$1 \leq \exists i \leq K,\ o_i(x^+, y^0) < o_i(x^-, y^0)$$

$$K+1 \leq \forall i \leq K+L,\ o_i(x^+, y^0) = o_i(x^-, y^0).$$

これから、 $((X \setminus X^*) \times Y) \cap V^* = \emptyset.$　　同様に $(X \times (Y \setminus Y^*)) \cap V^* = \emptyset.$

次に $(x^*, y^*) \in V^*$ ($x^* \in X^*,\ y^* \in Y^*$) を示す。

もし $(x^*, y^*) \notin V^*$ であると仮定すると、

$$\exists (x^+, y^+) \in V,\ (x^+, y^+) \prec (x^*, y^*).$$

このとき $1 \leq \exists i \leq K+L,\ o_i(x^+, y^+) < o_i(x^*, y^*).$ しかし, $1 \leq i \leq K$ であればこれは $f_i(x^+) < f_i(x^*)$ を意味し、 $x^* \in X^*$ であることに反する。 同様に $K+1 \leq i \leq K+L$ であれば $y^* \in Y^*$ であることに反する。よって仮定は棄却される。

これらの議論から、 $V^* = X^* \times Y^*$ であることが示された。■

定理 $(x,y) \prec (x^*, y^-) \Leftrightarrow (x,y) \in X^0 \times Y^+$.

証明：

$\Leftarrow$) 定義から自明。

$\Rightarrow$)$\forall i,\ o_i(x,y) \leq o_i(x^*, y^-)$ であるから、 $\forall i,\ f_i(x) \leq f_i(x^*)$.

一方、 $\forall i,\ f_i(x^*) \leq f_i(x)$ なぜなら $x^* \in X^*$.

ゆえに $\forall i,\ f_i(x^*) = f_i(x)$ であり、つまり $x \in X^0$ となる。

これらから以下の関係が成立する。

$$1 \leq \forall i \leq K,\ o_i(x,y) = o_i(x^*, y^-) \tag{1}$$

$$1 \leq \forall i \leq K+L,\ o_i(x,y) \leq o_i(x^*, y^-) \tag{2}$$

$$1 \leq \exists i \leq K+L,\ o_i(x,y) < o_i(x^*, y^-). \tag{3}$$

(1) および (3) から、

$$K+1 \leq \exists i \leq K+L,\ o_i(x,y) < o_i(x^*, y^-). \tag{4}$$

が成立する。さらに (2) と (4) から、 $y \prec y^-$ because $o_i(x,y) = g_{i-K}(y), K+1 \leq i \leq K+L$ よって $y \in Y^+$。これらから $(x,y) \in X^0 \times Y^+$. ■

系 1 ： $x \prec y \ \Rightarrow\ x \overset{\alpha}{\prec} y$.

証明：

domination の定義から、 $\forall i\ f_i(x) - f_i(y) \leq 0,\ \exists i\ f_i(x) - f_i(y) < 0$.

ここで $\forall i,j\ \ 0 \leq \alpha_{ij}$ かつ $f_j(x) - f_j(y) \leq 0$ であるから、

$$\forall i \quad \sum_{j \neq i}^{1..K} \alpha_{ij}(f_j(x) - f_j(y)) \leq 0.$$

ゆえに、$\forall i\ g_i(x,y) \leq 0$ かつ、$f_j(x) - f_j(y) < 0$ なる j において $g_j(x,y) < 0$。よって $x \overset{\alpha}{\prec} y$. ■

系 2： α-Pareto 最適集合 $X^\alpha = \{\ x \in X \mid x \text{ is } \alpha\text{-Pareto optimal}\ \}$ は Pareto 最適集合 X^*

の部分集合である。

証明：

x を α-Pareto 最適集合の要素とする。

このとき $\{\ y \in X \mid y \overset{\alpha}{\prec} x\} = \emptyset$.

もし $w(\in X) \prec x$ とすると $w \overset{\alpha}{\prec} x$ (系 1).

ゆえに $\{\ w \in X \mid w \prec x\} \subset \{w \in X \mid w \overset{\alpha}{\prec} x\} = \emptyset$, すなわち x は Pareto 最適. ■

D. 独立制約充足戦略の適用可能クラス、設計指針とパラメータ設定

適用可能クラス

独立制約充足は本質的に多数の試行を必要とするため、数試行程度しか行えないような問題・環境においては適用が不可能である。また、制約条件が非常に緩い、あるいは強制操作による制約充足が容易で解を悪化させない問題に対しては適用することにあまり意味がない。制約充足のために違反をペナルティ化する必要があり、これを減少させることが解の質を悪化させやすい場合に、かつ多数の試行が（並列計算か逐次計算かは問わず）可能ならば、独立制約充足戦略を適用することが好ましいと考えられる。

設計指針とパラメータ設定

ペナルティそのものの設計に関しては通常の方法論に従うものとする。ただし独立制約充足においては、本来の目的関数とペナルティの線型和重みを考慮する必要がないため、適切なペナルティを設計することは比較的容易になると考えられる。

独立制約充足には大きく分けて並列型、上昇型、下降型、適応型の 4 つのタイプが存在する。計算資源が十分な場合はどの方法をとっても問題ないが、通常はそれぞれ好ましいものを選択すべきである。以下にそれぞれの特徴を示し、必要となるパラメータの設定法を記す。

並列型 ： 逐次的な計算が必要ないため、並列計算環境では並列型を用いるのが最も実装が容易である。並列型では、部分空間 $S_{B_i}^{B_{i+1}}, \quad B_i \in \mathbb{R}$ を最初に定める必要がある。各試行では、仮にその部分空間に制約充足解があったとしてもそれを発見できるとは限らないため、同じ部分空間を 2 度以上探索したり、部分空間が重なっていたりしても問題はない。並列型の難点は、問題を解く前からある程度どのくらいの目的関数値が得られるのか分かっていないと、その周辺だけを細分化するといった工夫がしにくいことである。目的関数値が連続値の場合は、降下型や上昇型に比べ並列型や適応型が適している。

降下型 ： 降下型は、初期の目標値を高めに設定し、制約充足が困難な状況から探索を開始する楽観的な手法である。下水道制御の場合は、目標値が高めとは処理場での揚水量変化が少ないことを意味し、これは探索空間が小さいことを意味する（何回も変化させる

ほうがパターンとしては多い)。このように、目標値が高くても探索が容易な場合は適するが、線分配置問題のように目標値すなわち配置する本数が多いほど探索が困難になるような問題には適さない。目的関数値が離散値の場合、降下型と上昇型はパラメータの設計が容易であり特に適する。

上昇型 ： 上昇型は、線分配置問題のように目標値が低いほど探索が容易な問題に適する。上昇型の欠点は、目標値上昇中に制約充足に失敗したらそこで探索を終了してしまう点にある。特殊な目標値では制約充足できないような問題では効率的ではなく、明らかに目標値が小さいほど制約充足が容易な問題において第一選択となる。

降下型/上昇型では、初期目標値 LB_f^{init} をそれぞれ「多分制約充足できない/できる」値に設定する。n 試行が可能であり、予想される目標値が LB_f^{expect} 程度である場合、目標値の変更分 δ_{LB_f} は $(LB_f^{init} - LB_f^{expect}) \times \frac{3}{2n}$ 程度とする。逆に、目的関数が離散の問題では δ_{LB_f} が 1 などに定まるため、初期目標値 LB_f^{init} を $LB_f^{expect} - \frac{2n}{3}\delta_{LB_f}$ 程度とする。

適応型 ： 適応型は、他のタイプよりも精密な探索が可能であるため、計算資源が比較的豊富で解の質を求める場合は第一選択となる。また、目的関数値によって探索の難度がさして変わらないと思われる場合、すなわち上昇型・降下型への親和性が明らかでない場合に特に適切である。目的関数値が連続値の場合、もしどの程度で制約充足が可能になるかが分からない場合は並列型よりも適応型の方がパラメータ設計が容易である。

予想される目標値が LB_f^{expect}、これ以上良い/悪いことはないと思われる値を LB_f^{best}、LB_f^{worst} とする場合、初期値 $LB_f^{init} = LB_f^{expect}$、初期変化量 δ_{LB_f}=$(LB_f^{best} - LB_f^{worst})/5$ 程度とする。制約充足に引き続き成功あるいは引き続き失敗した場合は変化量 δ_{LB_f} は変えず、成功後失敗あるいは失敗後成功した場合は δ_{LB_f} は $\frac{1}{3}$ 程度にする。