Vol.2016-EC-41 No.12
Vol.2016-EC-41 No.12
2016/8/5

カーレーシングゲームにおける多目的最適化に基づくコントローラの設計

金澤 直人[1,a)]　池田 心[1,b)]

概要：シューティングゲームなど，リアルタイム性をもつコンピュータゲームにおけるコンピュータプレイヤーは多くの場合チェス等のボードゲームにおける強さを獲得するには至っていない．これは深度の大きい木探索を行うには，与えられる時間が短すぎるためである．このうちいわゆるカーレーシングゲームにおいては入出力モデルを操作決定のために設定し，“ あるコースでのラップタイム ” を用いたオフラインでのパラメータ最適化を行う手法が主流となっている．しかしこの手法では，必ずしも初めて走行させるコースで満足な性能を発揮できないことが分かっている．本稿ではラップタイムと “ 走行時のマージン ” を目的関数とした 2 目的最適化を実行し，“ 高速だが危険 ” から “ 低速だが安全 ” まで多様なパラメータの候補を保持し，未知のコース毎に切り替えを行うというアプローチを取る．提案した目的関数に基づいて実際に最適化を行い，その後未知のコースで走行し候補から選択させる実験を行い，実際に多様かつ優れた解が獲得され，与えられた未知のコース毎に適切な候補を選択できることを示した．さらに，本稿での最適化での評価 1 回当たりの所要時間の長さを鑑みて，評価関数を推定しながらの最適化によって評価回数を大幅に減少させるアルゴリズムを導入した．

KANAZAWA NAOTO[1,a)]　IKEDA KOKOLO[1,b)]

1. はじめに

最適化はコンピュータ科学において最も重要な概念のひとつと言える．勾配法や進化計算など，多くの数値最適化アルゴリズムが開発され，様々な領域に応用されてきた．ゲームは最適化手法の Testbed として用いられる一方，ゲームに対する様々な目的の研究で最適化が有効であることも理解されており，応用先の典型例であるといえる．

これまでゲーム，特にチェスや碁などのボードゲームにおける最適化の主要な目的はコンピュータプレイヤーの強さを追求することにあった．しかし現在，チェスについてはコンピュータプレイヤーが人間を明白に上回っており，これまでの技術では困難な課題とされてきた碁でも今年に入り Google の AlphaGo[1] がトッププロであるイ・セドルに対し 4 勝 1 敗で勝ち越すなど，複数のボードゲームで ” 強さ ” を求める上では充分な能力が獲得された．したがって現在のゲームに関する研究は，対象や目的が多様化している．この流れの中で，シューティングなどのリアルタイム性をもつビデオゲームを対象とした研究も活発に行われている．

このうち本稿では，カーレーシングゲーム等と呼ばれるジャンルのゲームを対象とする．オープンソースの研究環境 [2] がよく整備されており，高速な走行を実現させるプレイヤー（以下コントローラ）と実現のための技術が次々と提案され，コンペティションも複数回 [3][4] にわたって行われてきた．カーレーシングにおいてはボードゲームと比較してはるかに短時間（高々 30ms 程度）でアクセル・ブレーキ・ステアリング等の操作を決定できることが重要である．そのためこれまでボードゲームのコンピュータプレイヤーの研究で用いられた木探索等による先読みはうまく機能しない．したがって，現在のところはニューラルネットワーク（ANN）をはじめとしたパラメータ化された入出力モデルをコントローラのモデルとして，ラップタイムに

[1] 北陸先端科学技術大学院大学
JAIST, Nomi, Ishikawa 923–1211, Japan
[a)] s1620005@jaist.ac.jp
[b)] kokolo@jaist.ac.jp

よる評価を通したオフラインのパラメータ最適化（例えば，ANN における重み等が対象となる）を実行するという手法が主流である．これらの手法は応答時間が短くなり，これまでのカーレーシングゲームについての研究でも効果的であると示されてきた．

一方でオフラインで最適化を実行した時には，得られたパラメータを広く一般のコースに適用してよいとは限らない．コントローラ（およびパラメータ）は最適化を実行したコース X の特徴に対して最適化されたものとなり，別のコース Y で走行させると遅すぎたり，コースアウトしてしまうことも多い．すなわち，特定のコースへの過適合は数秒のロスにとどまらず，致命的な結果をもたらしうるということである．一方で，コンペティションでは専用のコースが新たに用意された上で，ごく短い調整時間（例えば，[4] ではコース 5 周）が与えられることが多い．この時間を利用してレースまでに自動で再調整するアプローチがしばしば用いられているが，短時間では完了しない場合も示されている [5].

本稿ではまず，少回数の試行で未知のコースへ適応することを前提とした我々のアプローチ [6] について説明する．ここでは「速いが危険」「安全だが遅い」など多様な特徴のコントローラ群をオフラインの最適化で準備したうえで，少回数の実走行から選択する方法を試みた．また，準備のためにラップタイムだけではなく，走行時に「どの程度余裕を持っていたか」を表す目的関数を含めての多目的最適化を実行することとし，まず走行中の余裕を表現する目的関数の設計を試みた．そして実際に最適化を実行し，得られた解から最もよい性能を与えるコントローラを選択する実験を行い妥当性を検証した．

これを踏まえて本稿では，新たにオフライン最適化部分の時間短縮を図った工夫を試みた．ゲームにおける最適化においてはシミュレーション等実験による評価がしばしば伴うが，本稿では評価 1 回当たりの時間等のコストが大きいことを考慮し，目的関数の大域的景観を推定しながら最適化を行う Efficient Global Optimization(EGO) アルゴリズムを導入して，実際に評価回数及び実行時間を削減して，[6] と同等な結果が得られるかどうかを確認する．

2. 関連研究

2.1 多目的最適化

探索空間 X と目的関数 $f(\mathbf{x} \in \mathbb{R})$ が与えられているとき，単目的の最適化（最短距離のルートや最も強固な構造物の探索）のゴールは $f(\mathbf{x})$ を最大（最小）化する最適解 $\mathbf{x}^*$ を求めることにある．本稿で対象とするカーレーシングにおいては，タイム最速のコントローラを探索することに対応する．目的関数が 1 個の場合は，ただ 1 つの $\mathbf{x}^*$ を求められれば充分である．

しかし実問題においては新たな評価関数を考慮することが必要になる状況がしばしば現れる．先の例で言えば，距離の短い道ほど混雑している，強固な構造物はほど金銭的コストも高い，高速なコントローラほど高いコースアウトのリスクを伴う，といったものである. このような問題は多目的最適化として定式化される. ここで，$f_1(\mathbf{x})$,$f_2(\mathbf{x})$ といった複数の目的関数が与えられた時，求めるのは非劣解の集合（パレート最適解）$P \subset X$ である（x が非劣解であるというのは，端的には全ての $\mathbf{y} \in X$ に対してある i が存在して，$f_i(\mathbf{x})$ が $f_i(\mathbf{y})$ より優れていることを指す）．パレート最適解を得るために多くの手法が提案されているが，その中でも進化計算アルゴリズムは，シミュレーテッドアニーリング等単目的最適化で用いられる技術と比較すると複数個の解が自然に得られる性質があることから，多目的最適化の解法としてしばしば用いられる．

多目的最適化のための進化計算アルゴリズム（遺伝的アルゴリズム）としてはじめに VEGA が開発された後，現在では NSGA-2[7] 等がゲームの研究で用いられることがある．

2.2 Efficient Global Optimization

実世界における最適化問題の中には，例えば [8] のように評価値は単なる関数ではなくシミュレーションや実験の結果として付与されることがしばしばある．このときには，評価 1 回当たりの所要時間等のコストも考慮していく必要がある．

ゲームの研究における最適化でも，コンピュータ等が実際にプレーした結果を評価値とすることは多い．このとき，1 評価にかかる時間が，計算上のボトルネックにならない程度であれば無視することも可能だが，そうでない場合には結果の質と所要評価回数や時間的リソースとの兼ね合いは考慮しなければならない．カーレーシングゲームでは後述する標準的な研究プラットフォームの場合最小でも 1 回当たり 10 秒程度必要で，用いるコントローラによっては数分かかることもあるため，要求される評価回数によっては最適化そのものを検討し直す必要が生じうる．

以上で述べてきた実問題からの要請に対応する，実際の評価回数を大きく抑制する手法の一つとして EGO(Efficient Global Optimzation)[9] が存在する．これは目的関数の代替として Kriging モデルと呼ばれる近似式を与え，評価値の改善量を示す指標の組合せによって次に評価すべき点を決定しながら最適化する技術であり，[8] 等の主に大規模なシミュレーションによる評価を行う最適化問題に採用されている．しかし後述するように，モデルの推定や評価すべき点の決定においては多数の行列演算と数値積分が要求され，これに伴い通常の最適化と比較すると評価以外の処理には高い計算負荷がかかる上に，長時間の計算が必要となる．よって，適用にあたっては問題設定をよく確認しておく必要がある．

2.3 カーレーシング

現在行われている 3D の，物理演算を考慮したカーレーシングゲームに対するプレイヤー（以下，コントローラ）については，オープンソースの 3D カーシミュレータ TORCS（The Open source Racing Car Simulator）をベースとした研究環境が登場し，これに基づいて手法の比較が行われるようになって 10 年ほどである．TORCS[2] をベースとした研究用プラットフォームはよく整備され，容易に利用できるようになっており，現在のカーレーシングゲームに関する研究環境としてはデファクト・スタンダードとなっている．このプラットフォームを用いたコンペティションには多様なアプローチのコントローラが参加してきたが，有力な手法ではしばしば入出力モデルに進化計算アルゴリズムを適用したものが見られた．これらはおおまかに分類すればルールベースモデルとニューラルネットワークをはじめとしたその他のモデルをベースとしたものに類別できる．

“ Mr.Racer ”[10] はルールベースモデルと進化計算アルゴリズムの組み合わせによるコントローラの中で有力かつ著名なものであり，2011 年から 2013 年にかけて複数のコンペティションで優勝した．このコントローラは 28 個のパラメータに基づく if-then ルールによって制御を行うモデルをベースに，パラメータについてはオフラインの最適化を行う．また，オンラインでコースのレイアウトを計測・学習する手法を導入し，操作プランを適応的に構築するようになっている．

一方 “ GRNDriver ”[11] も遺伝制御ネットワークと呼ばれるモデルと進化計算アルゴリズムの組み合わせで構成された有力な手法で，2015 年に開催されたコンペティションでは Mr.Racer を上回って優勝を果たした．走行距離を目的関数として，3 段階に分けたオフラインでの最適化によって重みの調整を行う仕様になっている．

以上で述べた手法をはじめ，よいデザインの進化計算によるアプローチが複数存在するが，単一のコースで最適化されている場合には，未知のコースとの組合せによっては期待される性能を大きく下回るケースがあることも知られている．[12] で導入されたオンラインでの適用手法が用いられることもあるが，この手法については適応に長時間かかる場合があることも知られており [5]，コンペティションから短時間での適応（例えば，[4] における 5 周）が要求された場合対応できない可能性がある．

我々が取るアプローチに近い研究としては [13] で行われた，多目的最適化によるアプローチが挙げられる．[13] ではまず，3 種類のコース A，B，C で走行させた時のタイムをそれぞれ目的関数とした 3 目的の最適化を実行した．そのパレート解上に存在するパラメータを代表して，各コースでの最速となるもの，A〜C でのタイムの合計が最小となるもの，合計 4 種類のコントローラを選出した．その後，4 種のコントローラで組をつくり，選択を行いながら未知のコースを走行させた．このとき，A〜C でのタイムの合計が最小となるコントローラ “ Good Compromise ”は，A〜C でのタイムの線形和を目的関数とした単目的最適化によるコントローラよりも未知のコースでの高い性能を獲得していたことが，この研究の主要な成果である．

一方で “ Good Compromise ”自体の性能は [13] 中の他の比較対象から劣っており，この結果に基づき，論文中で用いられなかったパレート解上のコントローラにも着目すべきであり，同時にタイム以外の評価関数を用いた最適化を試みるべきであるとするのが本稿のアプローチの根本である．

3. 走行時のマージンを考慮した多目的最適化

これまで述べてきたように，過去の研究では目的関数を何個用意するにせよ，全ての目的関数がラップタイム（または走行距離）を表したものとなっていた．この場合単にあるコースで最速となるような調整ではよい結果を得られるが，複数の，未知のコースに対しては有効でない．したがって, 未知のコースにおける性能を推測するための新たな目的関数が必要となる．本稿では得られたパラメータを未知のコースにも適用することを念頭に置き，ラップタイムと，走行中の “ 安全度 ”を表す目的関数を置いた 2 目的最適化を実行する．これにより “ やや遅いが安全 ”, “ 高速だが危険 ”といった様々な特徴のあるコントローラ群を獲得することができる．

3.1 目的関数の定式化

従来の研究ではラップタイムや走行距離を目的関数と置いてパラメータの評価を行っていた．これらの目的関数のみで評価を行う場合，未知のコースでよい性能を得るのが困難であることは既に述べた通りで，従って我々はコントローラの操作の危険度を評価する新たな目的関数を導入する．本稿では [4] から配布されているサンプルコントローラを，4 個のパラメータを付与した上で使用する．このコントローラには車を常に中央に維持しようとする性質があり，我々は危険度 f_1 をコース中央からのどの程度離れたかとして以下の通り定義する．なお，$trackPos()$ は TORCS が提供する情報 [14] である．

$$\begin{aligned} f_1(\mathbf{x}) &= trackPos(\mathbf{x}) \\ trackPos(\mathbf{x}) &\in [0,1] \end{aligned} \tag{1}$$

先述の通り，危険度はコース中央から離れるほど上昇し，これはすなわち，コースアウトのリスクが高まるということである．

一方，これまでの研究と同様にスタート地点から 1 周したラップタイムも評価関数 f_2 として以下の通り導入する．

$$\begin{aligned} f_2(\mathbf{x}) &= Time \\ Time &\geq 0 \end{aligned} \tag{2}$$

3.2 コントローラ群の獲得

我々の手法は，本節で述べるオフラインでの「多様で優れたコントローラ群の多目的最適化」と，４章で述べるオンラインでの「適したコントローラの選択」の２つのステップからなる．

多様で優れたコントローラ群の多目的最適化では，前節で述べた「危険度」および「ラップタイム」を最小化するように，Pareto 解と呼ばれる複数の解を求めることを目的とする．前論文で用いたものは NSGA-2 と呼ばれる標準的な多目的最適化アルゴリズム [2] である．その基本的な流れは以下の通りである．

(1) 50 個の解（パラメータセット．コントローラ.）をランダムに初期化し，（コースを走らせることで）２つの目的関数値を評価する．
(2) 交叉オペレータ BLX-$\alpha(\alpha = 0.3)$ を用いて，新しく 50 個の解を生成し，２つの目的関数値を評価する．
(3) 100 個の解の *rank* を決定する．すなわち，他のどの解にも優越（dominate．どちらの目的関数においても優れること）されない解を $rank = 1$ とし，$rank = 1$ 以外のどの解にも優越されない解を $rank = 2$ とする．
(4) 100 個の解から，rank の優れる順に 50 個の解を選ぶ．同じ rank の解同士を比較しなければいけない場合には，混雑度という指標を用いて，「他とあまり似ていない解」を優先して残す．
(5) 総評価回数が 3000 回になるまで繰り返し，得られた 50 個の解を出力する．

コース X (track-X) を用いて最適化した１試行分の結果を図１に再掲する．[6] 横軸はラップタイム，縦軸は危険度を表し，左下側にあるほど良い解ということになる．

黒丸は $rank = 1$ の解，×は $rank = 2$ の解である．左上にある解はこのコース X では速いが他のコースでは危険すぎると思われる解，右下にある解は危険度は低いものの余りにも遅すぎる少し価値の低い解である．別のコース Y に対する最適化の結果も似たようなものとなっている．

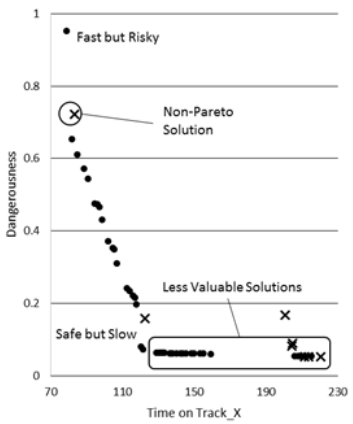

図 1 NSGA-2 によるパレート解

次章で説明する選択のフェイズの前に，得られた 50 個の解から有用なものだけを取り出すフィルタリングを行った．すなわち，$rank = 1$ 以外の解を削除し，また危険度の改善がほんの少しなのに速度の悪化が大きい，価値の低い解を α-domination という手法 [15] を参考にして削除した．本来はこのような削除は探索中に行うべきであり，それならば 3000 回よりも少ない評価回数で同質の解が得られるかもしれない．このフィルタリングを行った結果，残った解の数は Track-X で 20，Track-Y で 9 となった [6]．これがオフライン多目的最適化で得られた「多様で優れたコントローラ群」ということになる．

4. パラメータ選択

本章では，前章で得たコントローラ群から，コース毎に適切なコントローラを選択する方法 [6] について述べる．競技会においては，新しいマップが毎回提示され，短い練習時間（５周程度）が与えられる．この短い時間で，オンラインのコントローラ選択を行いたいというのが我々の研究の趣旨である．前章で得たコントローラについては，危険度が小さいほどより困難な（急カーブ等が多い）コースへも対応しやすいとみなしている．したがって，得られたコントローラは「高速だが危険」「安全だが低速」と多様な特徴を持っているということである．

4.1 スイッチング方法

我々は，コース X やコース Y でのオフライン最適化で得られたコントローラ間にあるラップタイムおよび危険度の大小関係が，未知のコースでも保たれていると仮定している．つまり，コース X でコントローラ 1 がコントローラ 2 よりも高速で危険な走行ならば，それはコース Z でも同じであろうということである．どの程度の危険さを許容するかはコースによるので，以下の手続きによる二分法風のアルゴリズムによって，完走できるコントローラのうち最もラップタイムが速いものを選択することを狙う．これは簡単には，平均的なコントローラを試して，完走できたならもっと危険なものを試し，完走できなかったならもっと安全なものを試す，ということである．

(1) 最適化で得た N 個の解を危険度が小さい順にソート
(2) 区間 $[a,b]$ を $a = 1$，$b = N$ として初期化
(3) ベストタイム t^* とそれを与えるインデックス i^* を $t^* = +\infty$，$i^* = 0$ で初期化
(4) $i = Floor(\frac{a+b}{2})$ としてコントローラ C_i を走行し，コースアウトの有無とラップタイム t の測定を行う
(5) コースアウトなしのとき $a = i+1$，特に $t < t^*$ ならば $t^* = t$，$i^* = i$
コースアウトした場合 $b = i-1$
(6) $b - a < 2$ ならば終了してコントローラ C_{i^*} を出力
そうでなければ 4 へ戻る

表 1 提案手法及び比較用コントローラのラップタイム（秒）

コントローラ	コース a	コース b	コース c	コース d	コース e	合計
(A) X で最適化	完走せず	完走せず	完走せず	完走せず	完走せず	-
(A) Y で最適化	完走せず	完走せず	完走せず	完走せず	完走せず	-
(B) 3 コースの総和	完走せず	完走せず	完走せず	224	135	-
(C) X 最速を選択	完走せず	完走せず	完走せず	完走せず	136	-
(C) Y 最速を選択	完走せず	完走せず	完走せず	完走せず	136	-
(C) 総和最小を選択	完走せず	完走せず	完走せず	225	136	-
(D) w=50, X で最適化	177	215	172	318	161	1043
(D) w=100, X で最適化	185	218	完走せず	322	163	-
(D) w=150, X で最適化	188	229	182	338	171	1108
(D) w=50, Y で最適化	141	169	135	243	134	822
(D) w=100, Y で最適化	153	194	152	291	147	937
(D) w=150, Y で最適化	152	193	152	291	148	935
(E) a-e で最適化	129	163	128	215	127	762
提案手法 X で最適化	248	170	134	221	140	914
提案手法 Y で最適化	137	170	133	229	138	808

計算量は通常の二分法と同等であり，例えば候補の数が 31 個以下であれば，[4] で定める 5 周という調整時間の範囲内で調整が行える．

5. 実験

比較対象としたのは以下の (A)-(E) の 5 つのコントローラである．

(A) ラップタイムのみ，かつコース X または Y のうち 1 つで単目的最適化した場合．

(B) コース X,Y,Z の 3 つのラップタイムを合計して，単目的最適化した場合．1 つのコースの場合に比べて頑健なコントローラが得られる可能性がある．

(C) コース X,Y,Z の 3 つのラップタイムそれぞれを，3 目的最適化した場合．得られた Pareto 解の中から，コース X について最速のもの・コース Y について最速のもの・コース X,Y の合計が最速のものをそれぞれ比較した．

(D) ラップタイムに加えて危険度も計算するが，それを（多目的最適化ではなく）$f = f_1 \times w + f_2$ と重みづけ和して単目的最適化した場合．

(E) 評価対象の“新しい”コース (a-e) について，ラップタイムを用いて単目的最適化した場合．これはもちろん競技会では行えないことであるが，そのコースにおける限界性能と我々の手法の性能を比較するために行ったものである．

5.1 結果・考察

我々はまず，4.1 節で説明したスイッチング方法が妥当なのか検証した．我々は 20 または 9 個のコントローラ群を持っているにもかかわらず，新しいコースが与えられるたびに二分法で最大でも 5 個しか調べずにコントローラを選択している．これは場合によっては危険で，より良いコントローラを見逃している可能性もある．そこで，20 または 9 のコントローラを全て調べたところ，二分法を用いた場合と最善のコントローラは同じだった．これは幸運である可能性もあるが，少なくとも今回の場合は問題がなかったということである．

5.1.1 (E) との比較

表 1 に，(A)-(E) の手法および我々の手法の結果を示す．まず (E) との比較を行う．(E) は本来未知のコースに対してそれを既知のものとして最適化を行った，限界性能を示すラップタイムである．

結果を比較すると，コース X で学習したものをコース a で用いた場合は非常に性能が悪化しているが，その他の 9 つの場合では性能悪化は 3% – 10% 程度にとどまっている．問題設定の難しさを考えれば，概ね許容範囲といえると考える．

5.1.2 (A)，(B)，(C) との比較

続いて，危険度を考慮しない (A)(B)(C) との比較を行う．表 3 上部を見れば分かるように，これらのコントローラは多くのコースで完走することができなかった．これは概ねコース X,Y よりも a-d は困難な（慎重を要する）コースであったためだと考えている．コース e では我々の手法よりも早くゴールできているので「常に我々の手法が良い」とは言えないが，全般的には我々の手法のほうが優れていると考える．

5.1.3 (D) との比較

最後に，危険度を用いたうえで単目的最適化する (D) との比較を行う．(D) は明らかに (A)(B)(C) よりは完走能力に優れており，危険度という我々の指標は仮に多目的最適化を用いない場合でも有益であることが分かる．

一方で，重みパラメータ w を大きくすると全体的にラッ

プタイムが遅くなる，つまり安全に走りすぎるということも分かる．逆に w を小さくしすぎると完走できない場合が出てくることも分かっている．つまり，(D) の手法では，与えられたコースごとに，（短いオンライン適応時間で）w を調整しなければいけないということである．

5.1.4 ここまでのまとめと課題

ここまで，既発表の実験結果 [6] について述べてきた．実験結果は概ね満足できるもので，「危険度の算出」「多目的最適化」「二分法の選択」という我々のアプローチは有効であると考える．しかしながら，課題も多く残っている．まずオフライン学習はどのコースで行うべきなのか，複数を用いるべきなのかが不明な点．学習用コース 2 つと評価用コース 5 つの計 10 通りしか行っておらず，また確率的最適化を用いているのに 1 試行しかしていない点などである．

これらは実は，本論文で新しく述べるもう一つの課題にも影響されている．すなわち，この問題のように「シミュレータ上でコースを走らせて 1 つの解を評価する」ような最適化問題では，その評価にかかる時間が実験のボトルネックになるということである．今回の実験では 3000 評価を行い，1 評価に標準的な PC で 10 秒ほど要するので，評価部分だけでも 1 試行あたり 9 時間ほど要する．これがより複雑なコントローラ（パラメータ数）を用いるとすれば 1 評価あたりの評価時間も増えるし，十分な精度の解を得るための最適化にかかる評価回数も増えるため，さまざまな実験を行うことはより困難になる．そこで本論文では，次章で述べる評価回数削減方法をとることにする．

6. Efficient Global Optimization

EGO は，最適化問題の中でも 1 回の評価コストが大きい問題において，過去の解の評価値を用いてできるだけ評価回数を抑えることを目的とした最適化手法である．そこでは，6.1 節で述べる Kriging モデルという最尤推定法で評価値の予測景観とその不確かさを計算し，6.2 節で述べる「この点を調べたらどの程度評価値の改善が期待できるか」という EI 値を用いて仮想の最適化を行い，有望な点のみを実際に評価する，というサイクルが行われる．最尤推定，EI 値の計算を伴う仮想の最適化にはそれなりに大きなコストを要するため，実評価コストが高い場合でなければかえって効率が悪いこともある．

本研究の対象であるレーシングゲームでは 1 回の評価コストが 10 秒〜数分にも及ぶため，（例えば巡回セールスマン問題のような古典的な最適化に比べ）EGO を用いるのに適した問題であると言える．

6.1 Kriging モデル

Kriging モデルにおいては目的関数 f を以下のように表す．

$$f(\mathbf{x}) = \mu(\mathbf{x}) + \varepsilon(\mathbf{x}) \tag{3}$$

ここで $\mathbf{x}$ は m 次元ベクトルである．実際の目的関数で評価を行ったサンプル点 x^i とサンプル点 x^i が与えられた時，$\varepsilon(\mathbf{x}^i)$ と $\varepsilon(\mathbf{x}^j)$ の相関は

$$d(\mathbf{x}^i, \mathbf{x}^j) = \sum_{k=0}^{m} \theta_k (x_l^i - x_l^j)^2 \tag{4}$$

により

$$Corr(\mathbf{x}^i, \mathbf{x}^j) = \exp[-d(\mathbf{x}^i, \mathbf{x}^j)] \tag{5}$$

と与えられる．実評価を行った全てのサンプル点間での $Corr$ を (i, j) 成分に持つ行列を R として与えたとき，新たなサンプル点の候補 x での Kriging モデルによる推定値は

$$\hat{f}(\mathbf{x}) = \hat{\mu}(\mathbf{x}) + \mathbf{r}R^{-1}(\mathbf{f} - \mu) \tag{6}$$

となる．ここで，$\hat{\mu}$ は μ の推定値で，$\mathbf{f}$ は各サンプル点の実評価値である．6 からはその対数尤度

$$Ln(\hat{\mu}, \hat{\sigma}^2, \theta_k) = -\frac{n}{2}\log(\hat{\sigma}^2) - \frac{1}{2}\ln(|R|) \tag{7}$$

が導かれ，このとき $\hat{\mu}$,$\hat{\sigma}^2$ が対数尤度から導かれる．7 の最大化，すなわち最尤推定を通し，m 個の $\theta_k (0 \leq \theta_k < \infty)$ を求めることで，Kriging モデルの形状を決定する．

6.1.1 Kriging による推定の例

図 2 は，2 次元の単純な関数（sin 関数の和）に対して Kriging で推定を行った例である．左端図は本来の評価関数値により色付けしたもので，赤が正値，青が負値を表す．ここから 50 点のランダムな箇所をサンプリングし，その評価値を用いて推定を行ったのが左から 2 番目の図である．全体として傾向は正しく推定できているが，一部ずれは生じている．その推定誤差を同じく赤と青で表現したのが右端の図であり，サンプル点の近くでは正しく評価できているが，サンプル点が疎な部分で大きな誤差が生じていることが分かる．Kriging モデルでは，推定値のみならずその誤差（0 以上）も出力される．それが右から 2 番目の図である．サンプルが疎な部分では大きな誤差が予測されており，これは実際の誤差と比べても適切な予測になっている．このように「不確かな部分」が分かるために，次節で述べる EI という指標を用いることができる．簡単に言うと，「良い値が予測されるがほぼ確実な部分」よりも「それよりは少し悪い値が予測されるが，不確かさの多い部分」が探索の際には優先されることになる．

6.2 Expected Improvement

ある点 $\mathbf{x}$ における Kriging モデルの推定精度は，平均二乗誤差

$$S^2(\mathbf{x}) = \hat{\sigma}^2\left(1 - \mathbf{r}^T R^{-1}\mathbf{r} + \frac{(1 - \mathbf{1}^T R^{-1}\mathbf{r})^2}{\mathbf{1}^T R^{-1}\mathbf{1}}\right) \tag{8}$$

によって表され，Kriging モデルによる目的関数 s の推定

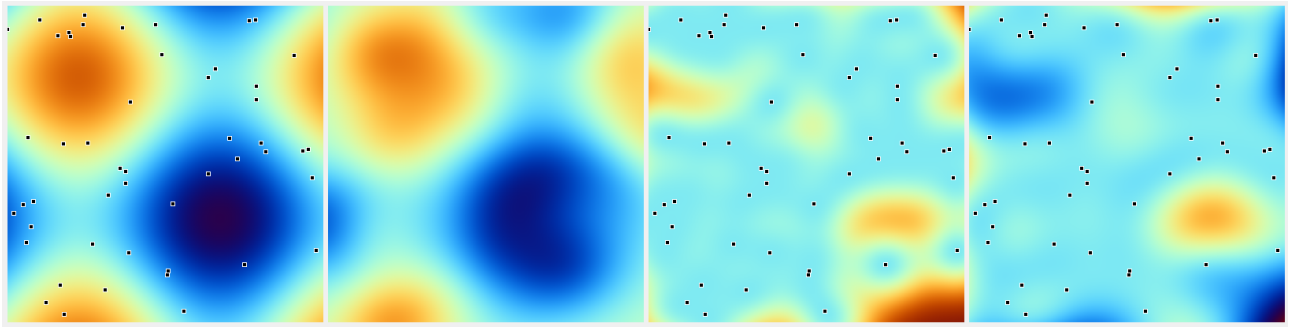

図 2　Kriging モデルによる推定

値は $N(\hat{f}(\mathbf{x}),S^2(\mathbf{x}))$ なる正規分布に従う不確定なものとなっている．

このとき，ある $\mathbf{x}$ で f がどの程度改善されうるか，その期待値を Expected Improvement（EI）といい，

$$E(I(\mathbf{x})) = \int_{-\infty}^{f_{ref}} (f_{ref} - f(\mathbf{x}))\varphi(f(\mathbf{x}))df \qquad (9)$$

と定義する．$\varphi(f(\mathbf{x}))$ は $N(\hat{f}(\mathbf{x}),S^2(\mathbf{x}))$ の確率密度関数である．多目的最適化では各目的関数に対して EI を求め，そのパレート最適解を得て次に実評価を行う点を決定する．なお，f_{ref} に用いる値については，多目的最適化の場合各目的関数に対する Kriging モデル上での $\hat{f}$ の最悪値を用いる．

6.3　全体の流れと結果

以上の Kriging モデル及び EI にもとづき，本稿で用いる EGO アルゴリズムは以下のフローで実行される．

(1) 初期解を 21 点ランダムに生成して実際の目的関数で評価する．
(2) サンプル点にもとづき，ラップタイムと危険度の 2 つの Kriging モデルを構成する．Kriging モデルに用いる評価回数は 2 万とした．
(3) そこで得られた 2 つの EI 値関数を用いて，仮想の多目的最適化を実行する．
  (a) 初期解を 100 点ランダムに生成して 2 つの EI 値を計算する．
  (b) 交叉オペレータ $BLX-\alpha$ によって 100 個の子個体を生成し，2 つの EI 値を計算する．
  (c) NSGA-2 と同じ方法で次世代の解 100 個を決定する．
  (d) 2 万評価に達するまで，交叉と世代交代を行い，最終的に 100 個の Pareto 解を出力する．
(4) パレート解から新しいサンプルとして 5 個を均等に抜き出す．
(5) 新しいサンプルを実際の目的関数で実評価する．
(6) 実評価回数が 100 になれば終了．そうでなければ Kriging モデルを更新する，すなわち (2) に戻る．

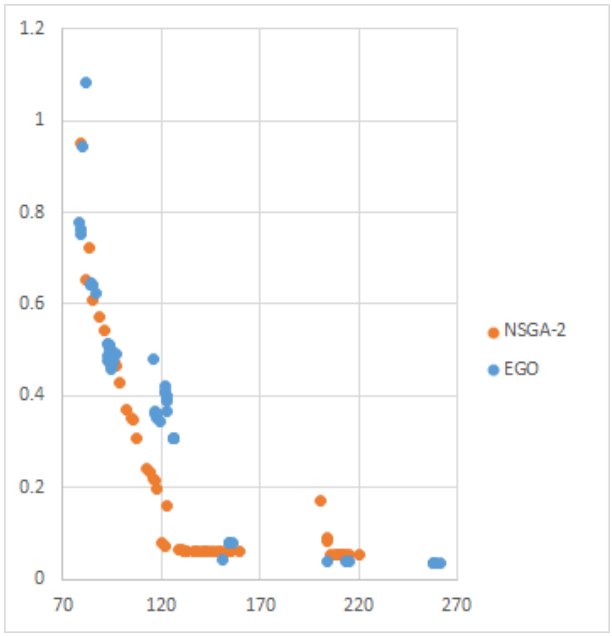

図 3　NSGA-2（3000 評価後）と EGO（100 評価後）の比較

この結果得られた解集合を図 3 に示す．黄色い点は NSGA-2 の 3000 評価回数で得られたものであり，これには実時間で 9 時間ほど要している．一方青い点は EGO の 100 評価回数で得られたものであり，これには実時間で 90 分ほど要している．EGO の場合，最尤推定や仮想の最適化にも時間を要するので，評価回数の差がそのまま実計算時間に反映されるわけではないが，それでも大きな時間短縮になっている．これはより複雑なコントローラを用いて 1 回あたりの評価時間（今は 10 秒）がより長くなったときにはさらに大きな差になる．

得られた解の質を詳しく見てみると，両端に近い点，即ち速度最優先の点や安全度最優先の点については EGO のほうが優れており，一方で中庸の点においては NSGA-2 のほうが優れている．この原因はさまざま考えられるが，上記のアルゴリズムの (4) において新しいサンプルを均等に 5 つ抜き出すのではなく，中庸付近に厚くなるように抜き出すことで性能が改善されるかもしれない．

7. まとめ

本稿では，レーシングゲームの AI 制御を行う際に，オフライン最適化を行うと未知のコースでは性能が悪くなる問題に取り組んだ．ラップタイムのみで最適化せず，危険度という指標を加えて多目的最適化することで「速いが危険」「安全だが遅い」といった多様なコントローラの集合を準備するアプローチをとる．競技会等で新しいコースが与えられた場合，短い準備期間の中でもそれらの集合から二分法により比較的良い解が選べることが実験から分かっている．本論文ではさらに，１回の実評価に大きなコストがかかることを鑑み，EGO という評価値景観推定を行いながら有望そうな点のみを実際に評価するというアプローチを実装し，既存手法と比較した．

既発表論文および本論文では，数個の学習コース・評価コースしか用いておらず，またコントローラも比較的単純なものに留まり，最適化の試行回数も少ないという問題点がある．今後はこれらを改善し，我々のアプローチが真に有望であることを示していきたい．

参考文献

[1] David Silver et.al.: Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, No. 7585, pp. 484–489 (2016).
[2] `http://torcs.sourceforge.net/`.
[3] `http://www.slideshare.net/dloiacono/gecco13scr/`.
[4] `http://cs.adelaide.edu.au/~optlog/SCR2015/`.
[5] Quadflieg, J., Preuss, M. and Rudolph, G.: Driving Faster Than a Human Player, *Applications of Evolutionary Computation*, No. LNCS 6624, pp. 143–152 (online), DOI: 10.1007/978-3-642-20525-5_15 (2011).
[6] Naoto, K. and Kokolo, I.: Multi-objective Optimization for Balancing Speed and Safeness in Car Racing Game, *Proceedings on 2016 International Workshop on Nonlinear Circuits and Signal Prosessing(NCSP'16)* (2016).
[7] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197 (online), DOI: 10.1109/4235.996017 (2002).
[8] Nestor V. Queipo, et al.: Surrogate-based analysis and optimization, *Progress in Aerospace SciencesProgress in Aerospace Sciences*, Vol. 41, No. 1, pp. 1–28 (2005).
[9] et.al., J. Q.: Donald R. Jones, Matthias Schonlau, William J. Welch, *Journal of Global Optimization*, pp. 455 – 491 (1998).
[10] Quadflieg, J., Preuss, M. and Rudolph, G.: Driving as a human: a track learning based adaptable architecture for a car racing controller, *Genetic Programming and Evolvable Machines*, Vol. 15, No. 4, pp. 433–476 (online), DOI: Doi 10.1007/S10710-014-9227-Z (2014).
[11] Sanchez, S. and Cussat-Blanc, S.: Gene regulated car driving: using a gene regulatory network to drive a virtual car, *Genetic Programming and Evolvable Machines*, Vol. 15, pp. 477–511 (online), DOI: 10.1007/s10710-014-9228-y (2014).
[12] et.al., J. Q.: Learning the track and planning ahead in a car racing controller, *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pp. 395 – 402 (2010).
[13] Quadflieg, J., Rudolph, G. and Preuss, M.: How Costly IS a Good Compromise : Multi-Objective TORCS Controller Parameter Optimization, *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 454–460 (2015).
[14] Loiacono, D., Cardamone, L. and Lanzi, P. L.: Simulated Car Racing Championship: Competition Software Manual, No. April (online), available from ⟨http://arxiv.org/abs/1304.1672⟩ (2013).
[15] Ikeda, K., Kita, H. and Kobayashi, S.: Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal?, *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 2, pp. 957–962 (online), DOI: 10.1109/CEC.2001.934293 (2001).