IDE/gravity とIDE/moving vector の相補効果

船木, 亮平
九州大学大学院芸術工学研究院

高木, 英行
九州大学大学院芸術工学研究院

出版情報：第2回進化計算学会研究会第8回進化計算フロンティア研究会合同研究会, pp.189-195, 2012-03. 進化計算学会
バージョン：
権利関係：

IDE/gravity と IDE/moving vector の相補効果

Complementary Effect of IDE/gravity and IDE/moving vector

船木亮平 [1*]　　高木英行 [2]
Ryohei Funaki[1]　Hideyuki Takagi[2]

[1] 九州大学大学院芸術工学府
[1] Graduate School of Design, Kyushu University
[2] 九州大学大学院芸術工学研究院
[2] Faculty of Design, Kyushu University

Abstract: ユーザとの対話によって探索を進める対話型進化計算では，ユーザ疲労が問題としてあげられる．対話型差分進化では対比較によって探索を進めることができ，遺伝的アルゴリズムなどの全個体比較に比べ評価におけるユーザ疲労を削減することができる．本発表ではさらにユーザの負担を軽減するために対話型差分進化の高速化手法を提案し，シミュレーション実験を行なうことで，対話型進化計算の条件下において提案手法が従来手法よりもユーザ疲労を軽減できることを示す．提案手法では世代間移動平均ベクトル（moving vector）と個体群重心点を用い，個体群全体を移動しながら個体分布の重心点周辺を探索する．

1　はじめに

人間が評価を行う対話型進化計算（IEC）ではユーザの疲労軽減が課題となっている．遺伝的アルゴリズム（GA）のように全個体の適応度を基に選択演算を行う進化計算を対話型に用いる場合，IEC ユーザは提示された全個体を比較し評価点を与えることを繰り返して最適化を行う．ユーザの反復評価を通して最適化を行う IEC では，探索時のユーザの疲労が結果に影響することもあり，評価によるユーザの疲労問題が問題としてあげられる．

これまでの色々な IEC 疲労対策が提案されてきた．例えば，IEC インターフェースの改善，より高性能な進化計算の発案，解への個体群の収束を早める高速化手法の導入などがある [13].

IEC インターフェースの改善の 1 つにトーナメント方式 GA[2] を IEC に応用したトーナメント方式対話型 GA（IGA）がある．全個体のトーナメントを行い，個体間順位を評価値とする手法である．ユーザは対比較で評価を行うため，全個体比較を行う通常の IGA よりも疲労軽減が可能であるが，本来全個体比較に基づいて選択演算のための評価値を決定する GA 演算ではなくトーナメント方式の部分比較しか行わないため評価値に含まれる情報量が少なくなり，選択演算に影響を与えて収束速度低下につながる可能性がある．

*連絡先：九州大学大学院芸術工学府
815-8540 福岡市南区塩原 4 丁目 9 番 1 号
Email: r.funaki620802@kyudai.jp

対比較ベースの対話型差分進化（IDE）[15, 16] も同様な対策の 1 つである．差分進化（DE）を IEC に応用した IDE [1, 7, 8] に DE 演算中の対比較特性を積極的に用いた手法で，前述のトーナメント IGA と異なり，基本となる DE 演算を簡略化することなく対比較を行うため，ユーザ疲労軽減としながら従来の IGA に比べて良い探索性能が示されている [15, 16].

また，疲労軽減のための様々な高速化手法も提案されている．例えば，これまでよく使われてきた GA 以外の進化計算を取り入れる対話型 PSO[9, 10] や上述の IDE もその取組の一環である．単峰性関数で適応度景観を近似したり適応度景観の周波数成分をフィルタリングすることで近似したりして，得られた近似曲面からエリートを求めて IEC の高速化に用いる手法もその 1 つである [11, 12, 14].

筆者らも 2 つの IDE 高速化手法（IDE/gravity と IDE/moving vector）を提案した [4, 5]．本来 IDE/best であれば IDE/rand よりもより良い収束性能が期待できるものの，対比較比較ベースの IDE で best 個体を見つけるためには対比較以外に全個体比較が必要でありユーザ疲労が増える．これらの手法の目的は，IDE/rand のユーザ疲労で IDE/best 並の収束性能を実現することであった．DE/gravity[4, 5] は大局的最適解周辺に探索個体が散らばり個体分布の重心は大局的最適解に近いであろうことを前提としている．世代を重ねる毎に中心に収束して来るほどこの前提条件に合ってくるであろうことが期待できる．しかし，大局的最適解が探索空間の境界に偏るほど大局的最適解から見た境界側

の個体数が少なくなるためこの前提条件が成り立たず，DE/gravity の性能が劣化する [6]．一方，DE/moving vector は，この前提条件が成り立つほど平均移動ベクトルが 0 に近づくため効果が少なくなるが，大局的最適解が探索空間の境界に偏るほど大局的最適解方向への平均移動ベクトルが大きくなり効果が期待できる．

本論文での目的は，これら相補的に効果が異なる両提案手法を組み合わせることで，大局的最適解がどこにあっても収束効果が発揮できることを実験的に確認することである．評価実験には大局的最適解を探索空間の中心から徐々に探索境界へ移動した評価関数を用いて，提案手法 DE/(gravity + moving vector) の高速化性能を評価する．参照手法には従来手法の DE/rand, DE/best，さらに DE/gravity を用いる．また，提案手法 DE/(gravity + moving vector) は IEC でのユーザ疲労軽減を目的としているため，実験は少個体数など IEC を想定した条件で行う．

2 比較手法

2.1 通常の DE と対話型 DE

DE では，個体間差分を用いて探索を行う．進化の対象となる個体である target vector と次世代の候補となる trial vector との比較を行い，良い方を次の世代に引き継ぐ．この部分が対比較である．この操作を全個体に行うことで 1 世代となる．trial vector の生成方法は，その基となる base vector の選択方法によって 2 つの種類がある．

DE/rand では，全個体からランダムに選択した 1 個体を base vector とする．ランダムに選択した 2 個体の差分 vector をこの base vector に加えて mutant vector を生成し，この mutant vector と target vector を交叉させて trial vector を生成する．DE/best では，全個体の中の最優良個体を base vector とする．その後は DE/rand と同様に trial vector を作成する．

DE は base vector の周辺を探索する手法であり，base vector がランダムに選択される DE/rand は広い範囲を探索し，DE/best では最優良個体の周辺を探索する．DE/best は DE/rand よりも収束性能が高く，筆者らの予備シミュレーション実験でも同様の結果が得られている [3, 4, 5, 6].

IDE ではユーザが個体の比較を行う．この時，ユーザは提示 2 個体のうち良い個体を選択するだけで探索を進めることができるため，全個体比較を行いそれぞれに点数を与える IGA よりも疲労軽減効果があり，特に，評価対象が音声や動画などの時系列タスクの場合に疲労軽減効果が顕著である．

2.2 DE/gravity

対比較ベースの IDE は対比較のみを行うため全個体の評価の相対関係は判らない．IDE/best は IDE/rand よりも収束性能が高いのだが，全個体を比較して最優良個体を決定する必要があるため，IDE/rand よりも IDE ユーザ負荷が増えてしまう．特に，音声や動画のような時系列タスクの場合，全個体比較は困難である．

IDE/gravity は探索空間に散らばった個体群の重心点を base vector とする手法である（図 1）．探索が進むと個体群は大局的最適解の周辺に収束していく．この時，重心点は大局的最適解に近づくので，その重心点を base vector として周辺探索すれば解探索が高速化できる．しかし，大局的最適解から見て個体がどちらかに極端に偏った場合，重心点は大局的最適解から離れるため性能は劣化するという欠点がある [6]．特に，大局的最適解が探索範囲の境界にある場合に，探索が進み個体群が収束しても大局的最適解の周辺に一様に散らばることはない．

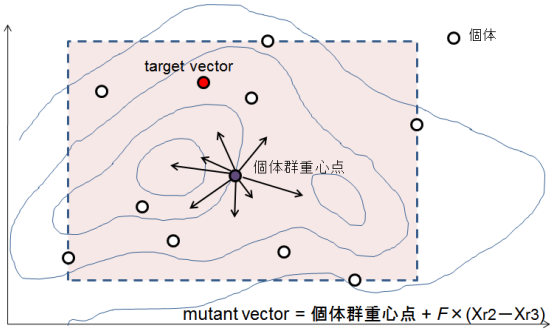

図 1: 個体群の重心を base vector とする IDE/gravity

2.3 DE/moving vector

DE では，target vector と trial vector の対比較を行うため，世代交代前後での個体の対応関係が明確であり，個体の移動の軌跡を求めることができる．この個体移動の軌跡によって個体群の収束方向を求め，探索に利用することで収束の高速化を目指す．本提案では，世代交代後に全個体の移動軌跡の平均移動ベクトル（moving vector）を求め，base vector に加えることで収束方向への収束を早める．以下がその手順である．

1. moving vector の算出

   target vector が trial vector に書き換わった時には両者の差分ベクトルを，書き換わらなかった時には差分ベクトルの向きを反転させたものを累積保存する．全個体比較後のこの累積ベクトルの平均が moving vector である．

2. 新しい候補の追加

trial vector 作成時，base vector に moving vector を加える．target vector は，base vector + moving vector + ($F\times$ 差分ベクトル) によって作成された mutant vector との交叉を行なう．

最適解が探索範囲の中心にありその周辺に個体群が分布している場合，moving vecotr は打ち消し合い効果的ではない．逆に最適解が探索範囲の偏った位置にある場合，個体全体がその方向に移動するため moving vecotr が大きくなり高速化が期待できる（図 2)．

DE/moving vector はそれぞれの base vector 選択方式に適用する手法であり，今後はそれぞれ DE/(rand + moving)，DE/(best + moving)，DE/(gravity + moving) と表記する．

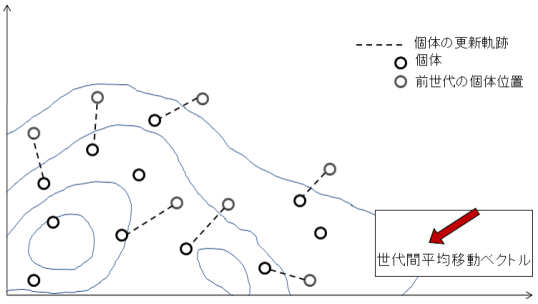

図 2: moving vector によって収束方向を推定

3 実験条件

DE/gravity，DE/rand，DE/best と，DE/(gravity + moving)，DE/(rand + moving)，DE/(best + moving) の合計 6 手法で性能比較を行う．実験条件は，16 個体数，10 次元パラメータ，交差率 0.8 の一様交差，差分ベクトルのスケーリングファクタを 0.8，とする．また，100 世代までの探索を千回試行繰り返し，危険率 5%と 1%で Kruskal-Wallis 検定を行う．検定を行なう世代数は，IDE でユーザが評価を行えるであろう 10 世代から 15 世代付近で行い，収束が早く 10 世代で各手法が収束し終わってしまう場合は 5 世代で，収束が遅い場合は 20 世代で検定を行なう．

シミュレーション実験のテスト関数として，ユーザ評価特性モデルである混合ガウス関数，検証用としてベル関数を用いる．また，大局的最適解を探索空間の中心，または端になるように探索範囲を調整する．

混合ガウス関数

人間の評価特性を取り入れた評価関数である．多峰性ではあるが大局的には大谷構造で，騙し問題のような悪問題特性を持たない評価特性が，人間の評価モデルに近いであろうとの仮定の下に作成された関数である (図 3 参照)．具体的には次式に示す混合ガウス関数を評価関数とする．

$$f(x_1,...,x_n)=\sum_{i=1}^{k} a_i \exp\left(\sum_{j=1}^{n}\frac{(x_{ij}-\mu_{ij})^2}{2\sigma_{ij}^2}\right) \qquad (1)$$

k は組み合わせるガウス関数の最大個数で本実験では k=4，n は次元数であり 10 として実験を行う．σ_{ij}，a_{ij}，μ_{ij} は以下に示す定数である．

$$\sigma=\begin{pmatrix}1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5\\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2\end{pmatrix}$$

$$a=\begin{pmatrix}3.1\\ 3.4\\ 4.1\\ 3\end{pmatrix}$$

$$\mu=\begin{pmatrix}-1 & 1.5 & -2 & -2.5 & -1 & 1.5 & -2 & -2.5 & -1 & 1.5\\ 0 & -2 & 3 & 1 & 0 & -2 & 3 & 1 & 0 & -2\\ -2.5 & -2 & 1.5 & 3.5 & -2.5 & -2 & 1.5 & 3.5 & -2.5 & -2\\ -2 & 1 & -1 & 3 & -2 & 1 & -1 & 3 & -2 & 1\end{pmatrix}$$

今回は，図 4 のように，この関数の探索領域をシフトすることで大局的最適解の位置を探索空間の中心，端の 2 つのパターンで実験を行う．探索空間の大きさは各次元とも 10.24 とする．

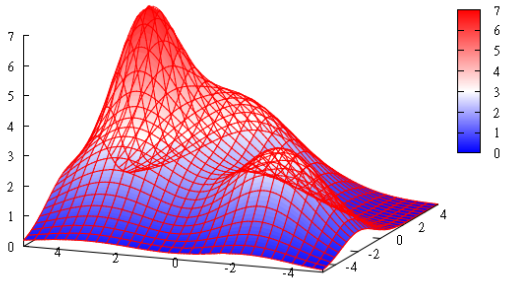

図 3: 2 次元表示の混合ガウス関数

ベル関数

DE/gravity では大局的最適解が探索空間の端にあり，かつ評価値が急激に変化するような特性をもつ評価関数で性能が低下することが分かっている [6]．傾斜特性

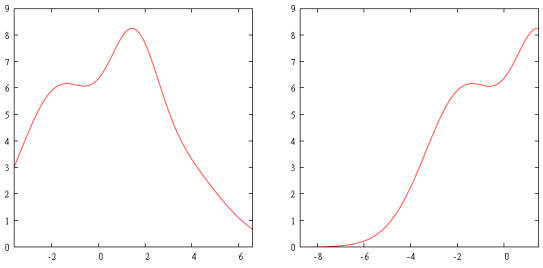

図 4: 探索範囲をシフトすることで混合ガウス関数の大局的最適解を位置を調整する．大局的最適解が探索範囲の中心にある場合（左）と端にある場合（右）．

をパラメトリックに変えられる式（2）の鐘形状関数を用い，改善手法との比較解析を行う．

$$f(x_1, ..., x_n) = \frac{1}{1 + \sum_{i=1}^{n} \left(\frac{x_i - a}{c}\right)^{2b}} \qquad (2)$$

ここで b と c は定数で，b は 3，c は 1, 3, の 2 種類とする．c の値が大きいと全体的に緩やかに，小さい場合は最適解周辺で大きく評価値が変化する．最適解は [0, ⋯ ,0] で最大値 1.0 をとる．本実験では最適解が探索範囲の中心になる ($-5.12 \leq x \leq 5.12$) と最適解が探索範囲の端になる ($0.0 \leq x \leq 10.24$) での探索を行う．

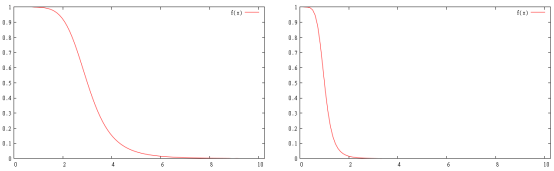

図 5: 左から，c 値が 3，1 のベル関数形状．

4 実験結果

4.1 混合ガウス関数

収束結果を図 6 に，15 世代目での検定結果を表 1 に示す．

最適解が中心にある場合，DE/gravity は 25 世代付近まで DE/best と同じような曲線を描いた．DE/moving vector と組み合わせることで，DE/gravity，DE/best では性能に変化は見られなかったが，DE/rand では性能が低下した．

最適解位置が端になると，DE/gravity は DE/best よりも大きく性能が低下し，DE/rand と DE/best の中間程度の曲線となった．また，DE/moving vector と組み合わせることで 3 手法とも性能が向上し，DE/(gravity + moving) は DE/best に近い曲線になった．

表 1: 世代数 15 での有意差．≈ は有意差があるとは言えない，＞と ≫ はノンパラメトリック検定（Kruskal-Wallis 法）で各々危険率 5%と 1%で有意差があることを表す．

中心	best + moving ≈ gravity ≈ gravity + moving ≈ best ≫ rand ≈ rand + moving
端	best + moving ≫ best ≫ gravity + moving ≫ rand + moving ≈ gravity ≫ rand + moving

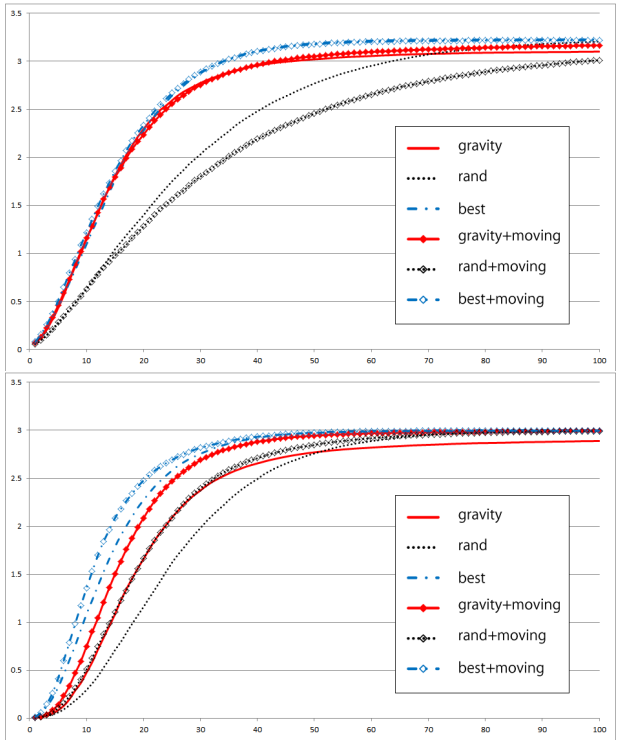

図 6: 混合ガウス関数での収束特性．横軸は世代数，縦軸は適応度．(上) 最適解が中心にある場合，(下) 最適解が端にある場合

4.2 ベル関数 $c = 3$

収束結果を図 7 に，15 世代目での検定結果を表 2 に示す．

表 2: 世代数 15 での有意差．記号は表 1 を参照のこと．

中心	gravity ≫ gravity + moving ≫ best + moving ≫ best ≫ rand ≫ rand + moving
端	best + moving ≫ best ≫ rand + moving ≫ gravity + moving ≫ rand ≈ gravity

最適解が中心にある場合，DE/gravity は DE/rand,

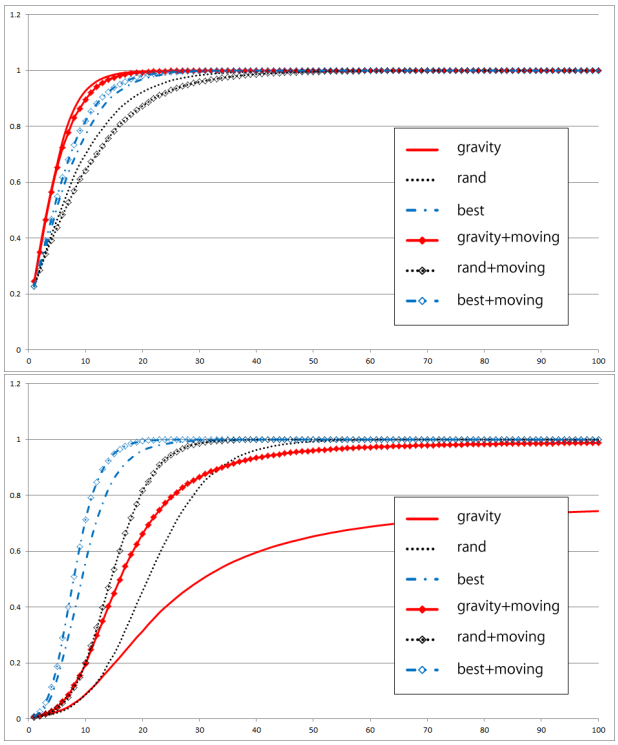

図 7: ベル関数（$c = 3$）での収束特性．横軸は世代数，縦軸は適応度．（上）最適解が中心にある場合，（下）最適解が端にある場合

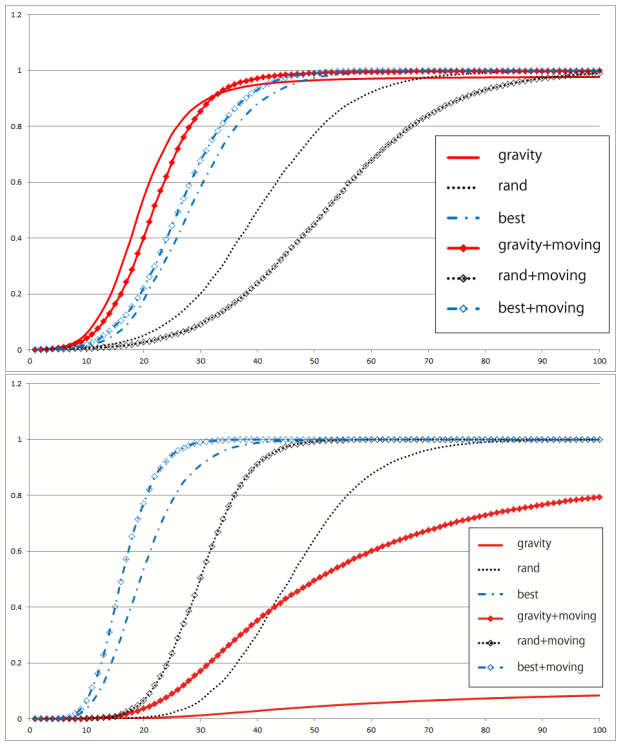

図 8: ベル関数（$c = 1$）での収束特性．横軸は世代数，縦軸は適応度．（上）最適解が中心にある場合，（下）最適解が端にある場合

DE/best よりも早く収束した．DE/moving vector と組み合わせることで DE/best では性能が向上したが，DE/gravity，DE/rand では性能が低下した．

最適解が端にある場合，DE/gravity は 10 世代付近から DE/rand よりも大きく性能が低くなった．また，DE/moving vector と組み合わせることで 3 手法とも性能向上がみられ，DE/(gravity + moving) は DE/rand よりも高性能となった．

4.3 ベル関数 $c = 1$

収束結果を図 8 に，15 世代目での検定結果を表 3 に示す．

表 3: 世代数 20 での有意差．記号は表 1 を参照のこと．

中心	gravity ≫ gravity + moving ≫ best + moving ≫ best ≫ rand ≫ rand + moving
端	best + moving ≫ best ≫ rand + moving ≫ gravity + moving ≫ rand ≫ gravity

最適解が中心にある場合，DE/gravity は DE/rand, DE/best 方式よりも早く収束した．DE/moving vector と組み合わせることで DE/best では性能が向上したが，DE/gravity，DE/rand では性能が低下した．

最適解が端にある場合，DE/gravity は DE/rand よりも大きく性能が低くなった．また，DE/moving vector と組み合わせることで，DE/(gravity + moving) は DE/rand よりも 40 世代付近まで高性能となった．

5 考察

第 1 節に述べた各 DE 手法の性質に今回の実験結果を加えると，各 IDE 手法の性質は表 4 のように表される．

大局的最適解が探索範囲の中心にある場合はすべての評価関数で提案手法 DE/gravity と DE/(gravity + moving) の両方で DE/rand よりも高性能となった．また，人間の評価特性をモデル化した混合ガウス関数では大局的最適解が探索範囲の端の場合でも DE/gravity は DE/rand よりも高性能であり，DE/(gravity + moving) はさらに性能が上がり，DE/best に匹敵する性能となった．

ベル関数（$c = 3$）では最適解が端にある場合に DE/gravity は DE/rand 以下の性能となっていたが，DE/moving

表 4: 各種 IDE 手法の性質.「中心」「端」は大局的最適解が探索空間の中心部にある場合と端にあることを表す.

各種 IDE 手法	収束性		ユーザ疲労
	中心	端	
IDE/best	○	○	×
IDE/rand	×	△	○
IDE/gravity	○	×	○
IDE/(best + moving)	○	○	×
IDE/(rand + moving)	×	△	○
IDE/(gravity + moving)	○	△	○

vector を組み合わせることで DE/rand よりも高性能となった.

ベル関数 ($c = 1$) でも同様に DE/gravity と DE/moving vector を組み合わせることで改善することができ，40 世代まで提案手法は DE/rand よりも高性能となった. しかし，それ以降は DE/rand の方が高性能となった. IEC 探索問題ではユーザは 40 世代も探索を行うことができないため，序盤での収束性能の向上は十分によい傾向である．また，この評価関数では従来手法である DE/rand での探索で 20 世代でも全く最適解へ収束が始まっておらず，IEC 探索問題としては難易度が高すぎる問題といえる.

6 結論

大局的最適解が探索範囲の端にあると性能の低下がみられる DE/gravity に DE/moving vector を導入することで改善することができた．この DE/(gravity + moving) は IEC 探索問題の条件下では大局的最適解が端にあっても十分に高速化を期待できる.

謝辞

本研究は科学研究費（課題番号 23500279）の助成を受けたものである.

参考文献

[1] T. Akbal, G. N. Demir, A. E. Kanlikilicer, M. C. Kus, and F. H. Ulu, “Interactive Nature-Inspired Heuristics for Automatic Facial Composite Generation”, Genetic and Evolutionary Computation Conference (GECCO2006), Undergraduate Student Workshop, Seattle, WA, USA (July, 2006).

[2] P. J. Angeline and J. B. Pollack, “Competitive Environments Evolve Better Solutions for Complex Tasks,” 5th Int. Conf. on Genetic Algorithms (ICGA1993), Urbana/Champaign IL, USA, pp. 264–270 (July, 1993).

[3] 船木亮平，高木英行「対話型差分進化の高速化手法の提案」進化計算シンポジウム 2010，福岡，pp.117–122, (2010 年 12 月).

[4] 船木亮平，高木英行「個体群重心と個体群移動ベクトルを用いた差分進化と対話型差分進化の高速化」第 6 回進化計算フロンティア研究会 (SIG-ECF)，名古屋，pp.122–127，(2011 年 3 月).

[5] R. Funaki and H. Takagi, “Acceleration Methods with a Gravity Vector and a Moving Vector for both Differential Evolution and Interactive Differential Evolution,” 5th Int. Conf. on Genetic and Evolutionary Computing, Kinmen/Taiwan and Xiamen/China, pp.287-290 (Aug./Sept., 2011).

[6] 船木亮平，高木英行「対話型差分進化高速化手法 DE/gravity の大局的最適解位置と収束特性との関係解析」進化計算シンポジウム 2011，岩沼，pp.103–110 (2011 年 12 月）.

[7] A. E. Kanlikilicer, “Interactive Differential Evolution for Facial Composite Generation”, Genetic and Evolutionary Computation (GECCO2006), Seatle, WA, USA (July, 2006).

[8] B. Kurt, A. S. Etaner-Uyar, T. Akbal, N. Demir, A. E. Kanlikilicer, M. C. Kus, and F. H. Ulu, “Active Appearance Model-Based Facial Composite Generation with Interactive Nature-Inspired Heuristics,” Multimedia Content Representation, Classification and Security, Lecture Notes in Computer Science. vol. 4105, pp.183–190 (July, 2006).

[9] Y. Nakano and H. Takagi, “Influence of Quantization Noise in Fitness on the Performance of Interactive PSO,” Congress on Evolutionary Computation (CEC2009), Trondheim, Norway, pp.2416–2422 (May, 2009).

[10] 中野雄, 高木英行「対話型 PSO」第 19 回インテリジェントシステムシンポジウム，会津若松, pp.228–233 (2009 年 9 月).

[11] 裴岩，高木英行「次元削減によって得られたエリートを用いた進化計算の高速化」第 1 回進化計算研究会・第 7 回進化計算フロンティア研究会合同研究会，東京，pp.25–31 (2011 年 9 月).

[12] 裴岩，高木英行「適応度景観のフーリエ解析による進化的探索高速化の試み」進化計算シンポジウム 2011，岩沼 (2011 年 12 月).

[13] H. Takagi, “Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation,” Proceedings of the IEEE, vol.89, no.9, pp.1275–1296 (2001).

[14] 高木英行, 印具毅雄, 大西圭「単峰性関数当てはめによる GA 収束高速化」知能と情報（日本知能情報ファジィ学会誌）, vol.15, no.2, pp.219–229 (2003).

[15] H. Takagi and D. Pallez, “Paired Comparison-based Interactive Differential Evolution,” 1st World Congress on Nature and Biologically Inspired Computing (NaBIC2009), Coimbatore, India, pp.375–480 (Dec., 2009).

[16] 高木英行, D. Pallez 「対比較ベース対話型差分進化」第 3 回進化計算シンポジウム，那覇，pp.245–251 (2009 年 12 月).