ニューラルネットワークを用いた多目的最適化に関する研究

須田 瑛斗

東京大学大学院情報理工学系研究科電子情報学専攻

伊庭 斉志

東京大学大学院情報理工学系研究科電子情報学専攻

1 序論

最適化問題とは, ある目的関数について最小値または最大値を示す変数の組みを求める問題である. 特に, 目的関数が与えられておらず, 入力に対する出力のみが与られるような場合の最適化問題をブラックボックス最適化 (BBO : Black Box Optimization) と呼ぶ. ブラックボックス最適化は実世界でも多数の応用が存在するが, 実世界の関数は悪条件であることが多く, 様々なアルゴリズムが研究されている.

目的関数が 1 つのものを単目的最適化と言うことに対して, 目的関数が 2 つ以上あるような問題を多目的最適化と言う. 多目的最適化はロケット発射台に設置される空力音響火炎デフレクタの設計 [1] や洋上風力発電所の設計 [2] に応用されている. 図 1 は二つの目的関数に対して最小化しようとする問題を想定した図であるが, 一般に F_1 および F_2 がともに最小になるような理想的な解は実行可能解に含まれていない. そこで, 多目的最適化の目標は二つの目的関数がトレードオフな関係になるような領域であるパレート界を探索することである. NSGAII[3] や MOEA/D[4] のように多目的最適化のアルゴリズムは多数研究されており, この論文も新たな多目的最適化のアルゴリズムを提案する論文である. この論文では単目的最適化のアルゴリズムとして提案されていた Deep-Opt[5] を多目的最適化に拡張することでニューラルネットワークを用いた最適化が多目的最適化にも応用可能であることを示すことを目標としている.

この論文の構成としては, 2 章では関連研究について説明した後に 3 章で提案手法について説明する. そして 4 章で実験設定について説明し 5 章でその結果, 6 章で考察を述べて 7 章で結論および今後の課題を述べる.

2 関連研究

2.1 Next-Ascent Stochastic Hillclimbing(NASH)

next-ascent stochastic hillclimbing(NASH) とは非常に単純な局所探索アルゴリズムである. その疑似コードは Algorithm1 にある. 各世代において, いくつか突然変異を起こすパラメータを選択して 0.75*p〜1.25*p の間の数で置き換えるということを行う. ただし, Algorithm1 における m は 0.02 程度の値である.

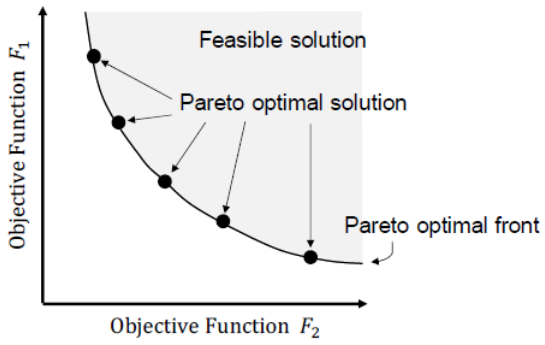

Figure 1: パレート界と実行可能解の様子 ([6] より引用)

置き換えた後の解が基の解よりも適合度がよくなっていれば次の世代では置き換えた解を基にして突然変異を起こす.

2.2 Deep-Opt

Deep-Opt[5] はディープニューラルネットワーク (DNN) を用いて NASH の効率を上昇させた研究である. 疑似コードは Algorithm2 にある. ディープニューラルネットワークを用いて BBO の解とその適合値のマッピングを学習し, back-drive によりそのネットワーク上で適合度が良いとされる箇所を見つけてその解周辺を NASH により探索している.

Algorithm 2 の 1 行目では最初に DNN がマッピングを学習するための解を生成するが, この方法はいくつかある. 1 つ目は単に全くランダムに解を生成する方法である. 2 つ目は NASH を行いその際に評価された解を全て保存しておいてそれらを初期解とする方法である. 3 つ目はある程度は全くランダムに生成するが, 残りはその生成された解の周辺を探索することで生成し, それらを初期解とする方法である. 元の論文 [5] では 3 つ目の方法が実験をして良い結果が得られたと書かれているため, 本研究でも 3 つ目の初期化の方法を用いることにする. 疑似コード 5 行目の DNN の学習について, 論文内ではテストセットをランダムに生成して学習の度にテストを行い, テストセットにおける真の適合値と DNN が出力する適合値の間の相関係数が負になると一度 DNN の重みを初期化するということを行っ

Algorithm 1 Next-Ascent Stochastic Hillclimbing(NASH) for Maximization

```
1: Initialize a random solution c
2: Best_Evaluation = Evaluate(c)
3: while termination condition is not met do
4:     c′ = c
5:     Number_Perturbations = Select an Integer from [1,m*|c|]
6:     for i = 1 .. Number_Perturbations do
7:         Select a parameter p from c′ randomly
8:         c′ = Modify p to a random value between [(0.75*p), (1.25*p)]
9:     end for
10:    Candidate_Evaluation = Evaluate(c′)
11:    if Candidate_Evaluation < Best_Evaluation then
12:        c = c′
13:        Best_Evaluation = Candidate_Evaluation
14:    else
15:        Discard c′
16:    end if
17: end while
```

ている. また相関係数が下がったタイミングで学習を終了させる.

17 から 19 行目について, back-drive により DNN が高い適合値を示す場所を探し当てる. ここで back-drive とは, 一般に DNN の重みを学習する際に行われる誤差逆伝播と同じことを入力について行うことを指している. すなわち DNN の重みを固定した上で, 誤差関数についてその誤差の値が小さくなる方向に最急降下法にて入力を更新していく. 今回出力値は 0.0 から 1.0 に正規化されているので最大値の探索の場合には 1.0 を目標値としてそれに近づくように入力を変化させている.

23,24 行目では back-drive により作られた解のうち最も良い解について NASH を行うことでその解周辺を探索している.NASH による探索を行う際に, 作り出された解を全て残しておく. それは次世代における DNN の学習のためである. ただし, その残された解の全てを次世代に受け継ぐわけではなく適合度の高い一部の解のみを次世代に受け渡す (26 から 28 行目). その理由として, 最初期はランダムな広範囲のマッピングを学習するが, 徐々に世代を経るにつれて適合度の高い解がある周辺のマッピングを DNN が学習することで back-drive による探索を効率的に行い, 全体における探索範囲を絞っていくためである.

Algorithm 2 Deep-Opt

```
1: Create a set of solutions S
2: Evaluate all solutions in S
3: while termination condition is not met do
4:     Scale all solutions in S to [0.0,1.0]
5:     Train a DNN to map S → Evaluation(S)
6:     Freeze the weights of DNN
7:
8:     Initialize C to be empty
9:     while |C| < Number-To-Generate-From-Model do
10:        Create a new solution by perturbing the best solution
11:        Append the new solution to C
12:    end while
13:    Select a fraction of C → C′
14:
15:    Clamp the network's output to 1.0
16:    for each d ∈ C′ do
17:        Initialize the DNN's input with d
18:        Change the input to d' by back-drive
19:        Replace d in C with d'
20:    end for
21:    Evaluate all solutions in C
22:
23:    Initialize NASH with a solution which have the highest evaluation in C
24:    Run NASH and record all of the unique solutions (into set Y ) evaluated by the NASH run.
25:
26:    Select a subset of the best solutions from Y → Y'
27:    S ← S + Y'
28:    prune S if necessary
29: end while
```

2.3 MOEA/D

MOEA/D とは多目的最適化問題を多数の単目的最適化問題へと分割して最適化する手法である. 単目的最適化問題へと分割する際には各解に割り当てられた重みベクトルによって計算される集約関数で定義される. 集約関数にはいくつか種類があり, [4] では Weighted Sum, Tchebycheff, PBI が提案されている. MOEA/D の流れは下記の通りである.

STEP1 初期化のステップ

STEP1.1 外部個体群を初期化

STEP1.2 一様な重みベクトル $\lambda^1, ..., \lambda^N$ を生成して各重みベクトルに対して T 個の近傍 B(i) = {$i_1, ... , i_T$}を定義する.

STEP1.3 各重みベクトルに対して初期個体 $x^1, ..., x^N$ を生み出し評価する.

STEP1.4 参照点を z=(z_1, ..., z_m) と設定 (ただし, z_i は全個体における目的関数値の i 番目の要素のうち最も小さい値)

STEP2 次世代の解を生成 (i = 1, ..., N)

STEP2.1 B(i) から二つの個体 (j,k) を選択して x_j, x_k を交叉させ, その後突然変異を施し子個体 y を生成する.

STEP2.2 必要に応じて参照点を更新する.

STEP2.3 子個体 y と B(i) 内の個体を集約関数を用いて比較して優越していたら更新.

STEP2.4 外部個体群に含まれる各個体に y が優越されなかったら外部個体群に y を追加し, 一方 y によって優越される個体が外部個体群に存在したらその個体を外部個体群から削除する.

3 提案手法

3.1 提案手法の概要

提案手法では, Deep-Opt を多目的最適化に拡張する. その疑似コードは Algorithm3 にある. 基本的な考え方は MOEA/D と同じで, 初期解をランダムに生み出した後に各解に重みベクトルを割り当てる. その後, 初期解を突然変異させることで DNN を学習するためのデータセットを作成する (4 行目). データセットの大きさは一世代の個体数の n 倍である. 本論文では n は 10 で固定している. その後, 終了条件を満たすまで探索を行う. 提案手法でも元の Deep-Opt と同様に解を [0.0, 1.0] にスケーリングする. その後, 先ほど作り出したデータセットを用いて DNN を学習する. この際, DNN が学習するのは設計変数から目的関数値のマッピングである. 学習の終了条件はそれぞれの目的関数において DNN の出力と実際の値の相関係数が一つでも下がったらである. また, その相関係数が一つでも 0 を下回った場合には DNN の重みを一度初期化してもう一度最初から学習する. これらは基となる Deep-Opt を参考にしている. 学習が終了したら back-driving を行うために DNN の重みを固定する (8 行目). DNN の出力を 1.0 に固定して目的関数の値が 1.0 に近づくように入力を更新していく. ただしこのときの入力は S に含まれる解で, S は各重みベクトルについてその重みベクトルが割り当てられている解の中で最も適合度の高い解の集合である. すなわち, S の要素数は重みベクトルの数に等しい. back-driving で解を改善したら突然変異を行って新たな解を作成する. このように作成された解と基となる解を重みベクトルを用いて適合度を比較して新たに作成した方が良ければ基となる解は S から取り除いて新たに作成した解を S に追加する. 一方, 良くならなければ新たに作成された解は基の解に置き換わらずに DNN を学習するデータセットに追加されるのみである. データセットに追加する理由としては, データセットには過去数世代に生み出された解が入っており, 一番古くに生み出された解よりは今世代の解の方がより現在探索している空間に近いためと考えられるからである. そうすることで DNN が現在探索している空間周辺をより精密に学習することでより高精度な解の生成が行える. 最後に, データセットは一定の大きさになるように必要に応じて最も古いものから順番に削除していく.

Algorithm 3 Deep-Opt for Multi-Objecive

```
1:  Create a set of solutions S
2:  Evaluate all solutions in S
3:  Assign a weight vector to each solution in S
4:  Make dataset to train DNN
5:  while termination condition is not met do
6:      Scale all solutions to [0.0,1.0]
7:      Train a DNN to map solutions → Evaluation(solutions) with dataset
8:      Freeze the weights of DNN
9:
10:     Clamp the network's outputs to [1.0]*number_of_objectives
11:     for each s ∈ S do
12:         Initialize the DNN's input with s
13:         Change the input to s' by back-drive
14:         Mutate s' to s"
15:         if evaluate(s", weight_vector_for_s) is better than evaluate(s, weight_vector_for_s) then
16:             replace s in S with s"
17:         end if
18:         Add s" to dataset to train DNN
19:     end for
20:     prune dataset if necessary
21: end while
```

3.2 Surrogate Model

提案手法では DNN を設計変数から目的関数値を推測するためのものとして学習させた. そのため, この DNN を Surrogate Model として用いることも可能である. 具体的には, Algorithm3 の 14 行目において突然変異する際に, 突然変異で生み出された解を DNN に通すことでその解の目的関数値の推定値を得ることができる. 本論文での Surrogate Model の使い方としては, Algorithm3 の 14 行目において突然変異にて新しい個体を 100 個体生み出し, その個体すべてで目的関数値の推定値を得る. そうした後に, その推定値を用いてルーレット選択を行うことで今世代の解を決定するという流れになっている. ここで最大値を取らない理由としては, DNN の推定精度が高い水準になるためには学習点が少ないと考えられるためルーレット選択にすることで多少の誤差は許容して適合度が低くなるような解を選ばなくするということに主眼を置いているからである.

4 実験設定

4.1 ベンチマーク問題

提案手法の性能を評価するために 2 種類のベンチマーク問題を用いた. 一つ目は LZ[7] であり, 設計変数においてパレート界が複雑になるように設定された関数である. LZ には 9 つの関数が存在してその目的関数および設計変数の次元数は表 1 の通りである. この実験では jMetalPy[8] による実装を用いた.

二つ目の関数は RE[9] である. この関数は実問題を関数に落とし込んだベンチマーク問題となっている. それぞれの問題における目的関数および設計変数の次元数は表 2 の通りである.

関数	目的関数の次元数	設計変数の次元数
F1	2	10
F2	2	30
F3	2	30
F4	2	30
F5	2	30
F6	3	10
F7	2	10
F8	2	10
F9	2	30

Table 1: LZ における目的関数および設計変数も次元数

関数	目的関数の次元数	設計変数の次元数
RE21	2	4
RE22	2	3
RE23	2	4
RE24	2	2
RE25	2	3
RE31	3	3
RE32	3	4
RE33	3	4
RE34	3	5
RE35	3	7
RE36	3	4
RE37	3	4

Table 2: RE における目的関数および設計変数も次元数

4.2 パラメータ設定

提案手法におけるパラメータは一世代の個体数・突然変異率・DNN の学習に用いるデータセットの数である. 突然変異率は 1/(設計変数の数) としていて, 一度の突然変異で変化する遺伝子の数の期待値が一つとなるようにしている. DNN の学習に用いるデータセットの数は提案手法の章でも書いた通り本論文では一世代の個体数の 10 倍としている.

目的関数の次元数	H	N(一世代の個体数)
2	29	30
	49	50
	99	100
	199	200
	299	300
3	6	28
	12	91
	18	190
	23	300
	33	595
	43	990

Table 3: 本実験における H の値

一世代の個体数について説明する前に提案手法における重みベクトルの生成方法を説明する. 提案手法における重みベクトルは H をハイパーパラメータとして{i/H | i=0,1,...H-1,H}から合計が 1 となるように重複を許して目的関数の数と同じ数だけ並べることで作り出す. すなわち, 重みベクトルの i 番目の成分を λ_i と置くと式 (1) のようになる. また, 重みベクトルは網羅的に生成するためその総数は式 (2) の N のようになる. ただし, ここで m は目的関数の数である.

$$\sum_i \lambda_i = 1, \lambda_i \in \{i/H | i = 0, 1, ...H - 1, H\} \tag{1}$$

$$N = {}_{H+m-1}C_{m-1} \tag{2}$$

したがって, 一世代の個体数は N となり H によって決定される数となる. H は性能に大きく影響を及ぼすパラメータであるためいくつか変えて実験を行っている. 具体的には表 3 の通りである. 目的関数が 2 つの際には N=H+1 となるので N は自由に設定できるが目的関数が 3 つ以上になると N は自由に設定できないため, N が 30, 100, 200, 300, 600, 1000 に最も近くなるような H を採用した. また, MOEA/D において目的関数値と重みベクトルを用いた適合度の計算法はいくつかあるが今回は単純に重みベクトルと目的関数値の内積を用いた (Weighted Sum).

提案手法と比較する手法は NSGAII[3], MOEA/D[4], MOEA/D-DRA[10] である. これらの手法を比較する際には jMetalPy[8] の実装を用いた.

4.3 評価手法

評価手法は LZ では IGD(Inverted Generational Distance) と HV(HyperVolume) を用いて, RE では HV のみを用いた. IGD とは理想的なパレート界における各解からアルゴリズムによって求められた解群のうち最も近い点の距離の平均である. すなわちアルゴリズムによって求められた解がどれほど理想的なパレート界に近いのかということを測る指標である. IGD は式 (3) で計算することができる. ただし, A はアルゴリズムによって求められた解群であり, R は理想的なパレート界である. また本実

験では p=2 としているので $d(r,a)^p$ は r と a のユークリッド距離である.

$$IGD = \left(\frac{1}{|R|} \sum_{r \in R} \min_{a \in A} d(r,a)^p \right)^{1/p} \quad (3)$$

HV は参照点を一つ設定した際にアルゴリズムによって求められた解とその参照点によって作り出される超平面の大きさである. 例えば目的関数の次元数が 2 のときを考えてみると図 2 のようになる. ここで $p^{(i)}$(i=1,2,3) はアルゴリズムによって求められた解であり, r は参照点である. HV は黒色の部分の面積となり, 大きければ大きいほど求めた解群がカバーできる面積が大きくなるため良い. 目的関数の次元数が 3 次元以上になっても同様に考えることができる. また, HV は参照点があれば計算ができるため理想的なパレート界が必要ない評価指標である. RE で IGD を用いない理由としては, RE で現在報告されているパレート界は存在するが実問題において理想的なパレート界というのはわからないため IGD を使うことが適切ではないと考えられるからである. 一方, LZ は人手で設計された問題であるため理想的なパレート界が既知であるから HV と IGD の両方を評価指標として用いている.

ただし, HV の結果については HV そのままの値を用いずに式 (4) によって計算される値を用いている.

$$\text{value\_in\_table} = 1 - \text{HV}/(\text{maximize\_of\_HV}) \quad (4)$$

分母にある HV の最大値はパレート最適解による HV である. すなわち, この表の値はパレート最適解と比較してどれだけの割合の領域をカバーできていないかという値となる. そのため, この値は小さい方がより広い領域をカバーできているということとなるため, 小さい方が性能が良いということとなる. HV の値をそのまま用いない理由としては, 特に RE で問題ごとに HV の値が大きく変わるためどれだけの性能なのかということをわかりやすくするためであり, 同時に HV の最大値で割った値を 1 から引く理由としては, アルゴリズムごとの差をより顕著にするためである. 例えば, HV の最大値で割った値をそのまま採用すると結果の表の上では同じ値であるのにも関わらずその両者で有意差が生じる可能性があるからである.

また, 実験結果の表に記載されている値は 20 回アルゴリズムを実行した際の平均値と標準偏差である.

5 実験結果

5.1 LZ の実験結果

4 章で述べた通り LZ の評価には HV と IGD の両者を用いる. HV についての結果は表 4 にあり, IGD についての結果は表 5 にある. IGD の結果の表には IGD の値をそのまま記載しているが, HV の結果の表に記載している値は 4.3 節でも述べた通り式 (4) で計算される値である. 表 4, 5 共に有意水準 0.01 で全ての手法の中で最もよかったものを太字にしている. LZ における HV の結果では全ての関数において提案手法は他手法と比較して統計的に有意に悪くなっていた. 一方, IGD では F8 のみ提案手法が他手法に比べて良くなっていたがその他の関数では他手法に比べて統計的に有意に悪くなっていた. ただ, HV の結果を見るとわかる通り全ての関数において提案手法でも理想的なパレート界と比較して 95%以上の領域をカバーすることができている. そのため, 他手法に比べて悪くなっているとはいえ提案手法はパレート界を充分求めることができているといえる.

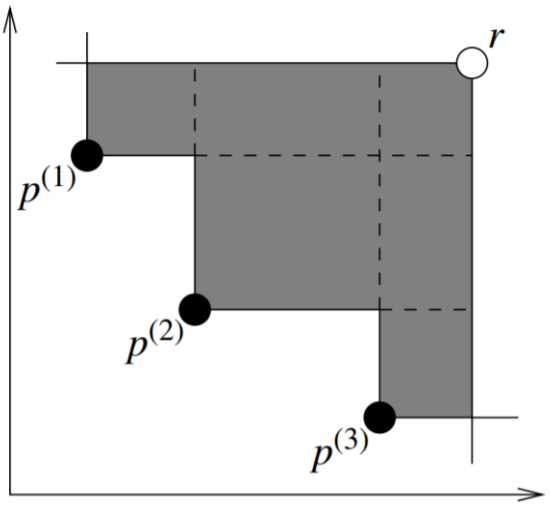

Figure 2: 二次元における HV の様子 ([11] より引用)

5.2 RE の実験結果

RE の実験結果は表 6 にある. 4.3 節で説明した通り, ここでも表に記載している値は式 (4) によって計算される値である. また, 記載されている値は 20 回アルゴリズムを実行した際の平均値と標準偏差である. ここでも同様に有意水準を 0.01 としてすべての手法の中で最も良くなっていたものを太字にしている. LZ では全ての関数で HV は他手法に比べて提案手法は統計的に有意に悪くなっていたが, RE では一部の関数で統計的に有意に良くなっていることがわかる. また, ほかの関数でも他手法よりも悪くなっているとはいえ, 理想的なパレート界と比較して 95%以上の領域をカバーすることができているため提案手法は充分パレート界を発見することができているということがわかる.

特に RE33 に注目するとこの関数での表の値は負の値となっている. これは現在報告されているパレート界よりも HV が大きいパレート界が発見されたことを意味している. 他の手法では全ての関数で正の値を取っているため報告されているパレート最適界よりも HV が大きいパレート界は見つからなかった. そのため問題によっては提案手法が他のアルゴリズムでは発見できない新たな解を発見できる可能性を RE33 の結果は示唆している.

5.3 Surrogate Model の実験結果

また、Surrogate Model についての実験結果も表 4, 5, 6 にある. この表を見てもわかる通り Surrogate Model として利用した際に他手法および提案手法の Surrogate Model 無しの場合と比較して統計的に有意に最も良くなっているような関数は無かった. また, Surrogate Model を用いることで用いないときに比べて良くなっているというようなことも無かった. ただ, 図 3 にあるように Surrogate Model 有りの方が Surrogate Model 無しの場合に比べて少ない評価回数においては性能が良くなっているような関数も存在した. 図 3 では LZ の F5 について, 横軸を評価回数として縦軸を式 (4) によって計算される値としたグラフである. 全ての関数でこのように Surrogate Model 有りの方が早く良いパレート界を発見できていたわけではないが, NN を Surrogate Model として使う際に収束したときの評価値は変わらないが評価回数が少ない場合には良い結果が得られるような可能性がある.

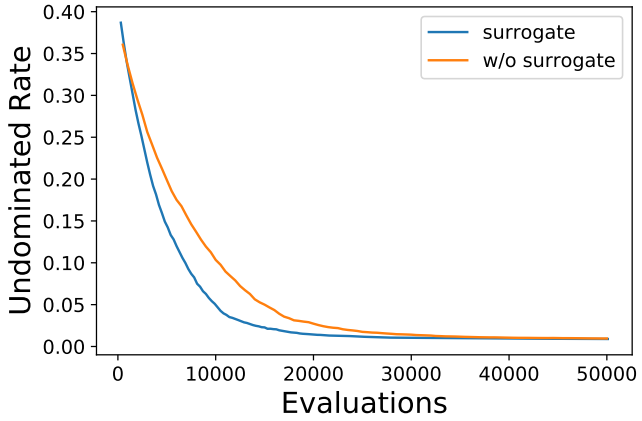

Figure 3: LZ F5 における評価回数に対する提案手法による HV の改善の様子 (縦軸の値は式 (4) によって計算される値である)

6 考察

LZ においては F8 の IGD 以外では提案手法が比較手法に比べて劣っていた. 一方, RE では問題によっては NSGAII 等の比較手法を統計的に有意に上回ることができていた. その理由として考えられるのは設計変数の次元数の違いである. LZ では設計変数の次元数は少なくとも 10 次元であるのに対して RE では設計変数の次元数は 7 であり, 比較手法を上回った関数では 4 次元以下であった. 設計変数の次元数が増えると NN において学習しなければならない重みの数が大幅に増えるため少ない学習点だと充分に学習することができず back-driving による解の生成の精度が落ちてしまうという状況が考えられる. LZ においてこのときの学習点の数は F6 以外で多くとも 3000 点であり, 入力次元が 30 次元で中間層が 3 層ある NN の学習をするのには不十分であると思われる. そのため, 設計変数の次元数が多い問題で提案手法を適用する場合には効率よく学習する必要があるか, 世代ごとに学習点の数を一定にするのではなくて徐々に増やしていき最終世代に向かうにつれて NN の精度が上がるようにするといった解決策がある.

次に Surrogate Model について, Surrogate Model を用いることで最終的な成績が良くなることは無かったが用いない際と比べて早い段階で良いパレート界を発見することができていた. 突然変異によって生成される解には当然ランダム性があるため良くなることもあれば悪くなることもある. そこで Surrogate Model を用いることで悪くなる解を評価することを防ぐことができていた問題も存在することがわかる. ただ, 最終的な成績が向上していないことから Surrogate Model を扱うことでできることはあくまで悪い解を選ぶ確率を減らすということだけであると考えられる. 最終的な成績が向上しない理由としてはパレート最適解に近づいてきたら突然変異の改善だけではより良い解を発見することが難しいためということが考えられる. 探索初期ではある程度良い解を高い確率で選べばそれだけ早く良いパレート界の発見に繋がるが, 探索が進むにつれて確実により良い解を選ぶ必要がある. ただし, ニューラルネットワークの精度が高くないことが考えられるので良い解を確率的に選択するということが難しく最終的な成績は Surrogate Model の有無に関わらず同じような値収束してしまったということが考えられる. また, 実験データでは示していないが Surrogate Model によって推測された解から評価される解の選択をルーレット選択ではなくて推測値が最も大きいものとして実験も行ったが, それはルーレット選択のときよりも性能が遥かに悪くなっていた. その理由は先ほどと同様でニューラルネットワークの精度が高くないため適合度の推測値の信頼度が低く, 推測値が最も高くとも実際の適合度はそこまで高くなかったということが考えられる.

7 結論

本研究ではニューラルネットワークの入力を変化させることで最適化を行う back-driving という手法を多目的最適化へと応用した. 結果として LZ では NSGAII, MOEA/D, MOEA/D-DRA と比較して良くなっているような問題は無かったが, 実問題をベンチマーク関数に落とし込んだ RE ではいくつかの問題で他手法を統計的に有意に上回る結果が得られた. 特に RE33 では今まで報告されているパレート界よりも高い HV を持つパレート界を発見することができていた. 一方, ニューラルネットワークを Surrogate Model として用いても最終的には用いなかった場合と比較してより良いパレート界を得ることはできなかった. ただ, Surrogate Model 無しの場合と比較すると Surrogate Model 有りの場合の方が少ない評価回数で高い HV を持つパレート界を発見できているような問題も存在した. このことから現状の Surrogate Model では最終的に高い適合度の解を得るというよりは少ない評価回数でより良い解を発見することの手助けにな

関数	提案手法 (w/o surrogate)	提案手法 (with surrogate)	NSGAII	MOEA/D	MOEA/D-DRA
F1	$2.35\times10^{-4}\pm4.92\times10^{-5}$	$2.49\times10^{-4}\pm6.79\times10^{-5}$	$8.98\times10^{-5}\pm2.09\times10^{-5}$	$\mathbf{3.12\times10^{-5}\pm1.34\times10^{-5}}$	$1.42\times10^{-4}\pm1.05\times10^{-4}$
F2	$2.00\times10^{-2}\pm1.44\times10^{-3}$	$2.04\times10^{-2}\pm1.27\times10^{-3}$	$1.52\times10^{-2}\pm8.04\times10^{-3}$	$\mathbf{3.59\times10^{-3}\pm3.05\times10^{-3}}$	$\mathbf{7.30\times10^{-3}\pm8.35\times10^{-3}}$
F3	$1.06\times10^{-2}\pm9.82\times10^{-4}$	$9.47\times10^{-3}\pm2.86\times10^{-3}$	$9.51\times10^{-3}\pm4.99\times10^{-3}$	$6.92\times10^{-3}\pm8.48\times10^{-4}$	$\mathbf{3.94\times10^{-3}\pm6.27\times10^{-3}}$
F4	$1.65\times10^{-2}\pm4.77\times10^{-3}$	$1.43\times10^{-2}\pm4.63\times10^{-3}$	$1.47\times10^{-2}\pm1.44\times10^{-3}$	$4.16\times10^{-3}\pm3.35\times10^{-3}$	$\mathbf{9.59\times10^{-4}\pm3.32\times10^{-4}}$
F5	$9.57\times10^{-3}\pm2.58\times10^{-3}$	$8.95\times10^{-3}\pm2.65\times10^{-3}$	$\mathbf{5.39\times10^{-3}\pm3.08\times10^{-3}}$	$\mathbf{3.93\times10^{-3}\pm4.73\times10^{-3}}$	$\mathbf{3.05\times10^{-3}\pm2.54\times10^{-3}}$
F6	$4.30\times10^{-3}\pm1.39\times10^{-3}$	$4.00\times10^{-3}\pm9.83\times10^{-4}$	$1.28\times10^{-3}\pm7.40\times10^{-4}$	$2.76\times10^{-4}\pm9.07\times10^{-5}$	$\mathbf{1.88\times10^{-4}\pm4.34\times10^{-5}}$
F7	$1.77\times10^{-2}\pm5.71\times10^{-3}$	$1.64\times10^{-2}\pm4.85\times10^{-3}$	$3.69\times10^{-2}\pm1.29\times10^{-3}$	$\mathbf{1.13\times10^{-2}\pm7.01\times10^{-3}}$	$2.17\times10^{-2}\pm8.46\times10^{-3}$
F8	$3.01\times10^{-2}\pm5.95\times10^{-3}$	$3.08\times10^{-2}\pm8.16\times10^{-3}$	$3.63\times10^{-2}\pm1.23\times10^{-2}$	$\mathbf{1.43\times10^{-2}\pm1.04\times10^{-2}}$	$2.29\times10^{-2}\pm1.32\times10^{-2}$
F9	$3.56\times10^{-2}\pm3.72\times10^{-3}$	$3.36\times10^{-2}\pm3.10\times10^{-3}$	$4.16\times10^{-2}\pm1.76\times10^{-3}$	$\mathbf{2.96\times10^{-3}\pm1.43\times10^{-3}}$	$\mathbf{5.48\times10^{-3}\pm4.65\times10^{-3}}$

Table 4: LZ における HV の結果

関数	提案手法 (w/o surrogate)	提案手法 (with surrogate)	NSGAII	MOEA/D	MOEA/D-DRA
F1	$1.01\times10^{-2}\pm1.15\times10^{-3}$	$9.88\times10^{-3}\pm8.19\times10^{-4}$	$4.24\times10^{-3}\pm4.44\times10^{-4}$	$\mathbf{1.32\times10^{-3}\pm2.66\times10^{-5}}$	$1.56\times10^{-3}\pm1.26\times10^{-4}$
F2	$5.75\times10^{-2}\pm5.18\times10^{-3}$	$5.81\times10^{-2}\pm4.10\times10^{-3}$	$1.02\times10^{-1}\pm1.80\times10^{-2}$	$\mathbf{3.84\times10^{-2}\pm1.36\times10^{-2}}$	$4.56\times10^{-2}\pm2.49\times10^{-2}$
F3	$4.17\times10^{-2}\pm1.18\times10^{-2}$	$3.97\times10^{-2}\pm4.42\times10^{-3}$	$5.13\times10^{-2}\pm1.27\times10^{-2}$	$\mathbf{2.81\times10^{-2}\pm2.18\times10^{-2}}$	$\mathbf{2.07\times10^{-2}\pm1.37\times10^{-2}}$
F4	$5.18\times10^{-2}\pm8.40\times10^{-3}$	$4.86\times10^{-2}\pm9.96\times10^{-3}$	$4.93\times10^{-2}\pm5.15\times10^{-3}$	$1.85\times10^{-2}\pm6.74\times10^{-3}$	$\mathbf{7.77\times10^{-3}\pm1.46\times10^{-3}}$
F5	$4.02\times10^{-2}\pm4.07\times10^{-3}$	$3.72\times10^{-2}\pm4.70\times10^{-3}$	$3.89\times10^{-2}\pm8.29\times10^{-3}$	$\mathbf{2.22\times10^{-2}\pm6.65\times10^{-3}}$	$\mathbf{2.03\times10^{-2}\pm9.37\times10^{-3}}$
F6	$1.26\times10^{-1}\pm8.59\times10^{-3}$	$1.26\times10^{-1}\pm1.09\times10^{-2}$	$1.11\times10^{-1}\pm1.54\times10^{-2}$	$\mathbf{4.53\times10^{-2}\pm4.05\times10^{-3}}$	$\mathbf{4.48\times10^{-2}\pm3.38\times10^{-3}}$
F7	$6.52\times10^{-2}\pm1.94\times10^{-2}$	$6.28\times10^{-2}\pm1.27\times10^{-2}$	$1.50\times10^{-1}\pm7.79\times10^{-2}$	$\mathbf{2.42\times10^{-2}\pm2.22\times10^{-2}}$	$4.75\times10^{-2}\pm2.48\times10^{-2}$
F8	$\mathbf{1.44\times10^{-1}\pm2.23\times10^{-2}}$	$1.54\times10^{-1}\pm2.84\times10^{-2}$	$1.53\times10^{-1}\pm3.04\times10^{-2}$	$1.75\times10^{-1}\pm3.34\times10^{-2}$	$2.09\times10^{-1}\pm8.11\times10^{-2}$
F9	$6.83\times10^{-2}\pm6.35\times10^{-3}$	$6.91\times10^{-2}\pm4.66\times10^{-3}$	$1.09\times10^{-1}\pm2.88\times10^{-2}$	$\mathbf{4.14\times10^{-3}\pm2.54\times10^{-2}}$	$5.15\times10^{-2}\pm3.16\times10^{-2}$

Table 5: LZ における IGD の結果

関数	提案手法 (w/o surrogate)	提案手法 (with surrogate)	NSGAII	MOEA/D	MOEA/D-DRA
RE21	$\mathbf{2.57\times10^{-3}\pm1.92\times10^{-4}}$	$\mathbf{2.46\times10^{-3}\pm2.17\times10^{-4}}$	$2.97\times10^{-3}\pm1.06\times10^{-4}$	$6.03\times10^{-1}\pm1.06\times10^{-2}$	$5.90\times10^{-1}\pm1.04\times10^{-2}$
RE22	$3.30\times10^{-2}\pm6.44\times10^{-3}$	$3.29\times10^{-2}\pm9.55\times10^{-3}$	$4.08\times10^{-3}\pm3.08\times10^{-4}$	$\mathbf{2.74\times10^{-3}\pm8.21\times10^{-5}}$	$\mathbf{2.71\times10^{-3}\pm7.43\times10^{-4}}$
RE23	$2.44\times10^{-3}\pm7.23\times10^{-4}$	$2.55\times10^{-3}\pm6.76\times10^{-4}$	$\mathbf{2.41\times10^{-4}\pm1.26\times10^{-5}}$	$1.98\times10^{-2}\pm2.51\times10^{-4}$	$2.01\times10^{-2}\pm4.38\times10^{-4}$
RE24	$\mathbf{5.11\times10^{-4}\pm6.49\times10^{-5}}$	$\mathbf{4.90\times10^{-4}\pm6.80\times10^{-5}}$	$6.72\times10^{-4}\pm5.34\times10^{-5}$	$5.52\times10^{-3}\pm3.53\times10^{-5}$	$5.50\times10^{-3}\pm3.22\times10^{-5}$
RE25	$\mathbf{2.28\times10^{-5}\pm5.30\times10^{-5}}$	$\mathbf{2.59\times10^{-5}\pm6.74\times10^{-5}}$	$\mathbf{3.01\times10^{-10}\pm1.05\times10^{-11}}$	$5.63\times10^{-1}\pm3.00\times10^{-12}$	$5.63\times10^{-1}\pm7.89\times10^{-12}$
RE31	$5.67\times10^{-5}\pm3.66\times10^{-5}$	$5.35\times10^{-5}\pm2.17\times10^{-5}$	$\mathbf{4.01\times10^{-6}\pm2.61\times10^{-6}}$	$2.42\times10^{-4}\pm9.79\times10^{-6}$	$2.43\times10^{-4}\pm9.34\times10^{-6}$
RE32	$2.36\times10^{-4}\pm9.47\times10^{-5}$	$2.21\times10^{-4}\pm7.80\times10^{-5}$	$\mathbf{4.58\times10^{-5}\pm1.05\times10^{-5}}$	$3.58\times10^{-2}\pm7.51\times10^{-4}$	$3.54\times10^{-2}\pm6.05\times10^{-4}$
RE33	$\mathbf{-1.23\times10^{-4}\pm2.89\times10^{-4}}$	$\mathbf{-3.47\times10^{-4}\pm2.33\times10^{-4}}$	$9.53\times10^{-4}\pm6.61\times10^{-4}$	$2.90\times10^{-4}\pm4.14\times10^{-4}$	$6.76\times10^{-5}\pm9.17\times10^{-4}$
RE34	$2.55\times10^{-2}\pm2.12\times10^{-3}$	$2.54\times10^{-2}\pm2.23\times10^{-3}$	$\mathbf{4.21\times10^{-3}\pm5.69\times10^{-4}}$	$6.07\times10^{-2}\pm8.39\times10^{-3}$	$6.23\times10^{-2}\pm9.10\times10^{-3}$
RE35	$1.16\times10^{-3}\pm1.15\times10^{-4}$	$1.14\times10^{-3}\pm1.54\times10^{-4}$	$1.42\times10^{-3}\pm4.22\times10^{-4}$	$\mathbf{1.04\times10^{-3}\pm1.70\times10^{-4}}$	$\mathbf{1.07\times10^{-3}\pm2.03\times10^{-3}}$
RE36	$2.51\times10^{-4}\pm1.67\times10^{-4}$	$3.75\times10^{-4}\pm1.96\times10^{-4}$	$\mathbf{1.08\times10^{-4}\pm7.81\times10^{-5}}$	$5.18\times10^{-4}\pm4.20\times10^{-4}$	$3.25\times10^{-4}\pm1.21\times10^{-4}$
RE37	$1.71\times10^{-2}\pm1.73\times10^{-3}$	$1.66\times10^{-2}\pm2.00\times10^{-3}$	$1.08\times10^{-2}\pm1.11\times10^{-3}$	$\mathbf{6.33\times10^{-3}\pm3.96\times10^{-4}}$	$\mathbf{6.16\times10^{-3}\pm3.18\times10^{-4}}$

Table 6: RE における HV の結果

る可能性があることが示唆されている.

今後の課題としては, モデル化してから最適化をするメリットとして別の問題を最適化した際の知識を他の問題に適用することでより素早い探索を可能にするということが考えられる. 学習されたニューラルネットワークには最適化した問題の知識が溜まっているためその知識を他の問題 (特に似たような問題) へと転用することが可能であると考えられる.

References

[1] Tomoaki Tatsukawa, Taku Nonomura, Akira Oyama, and Kozo Fujii. Multi-objective aeroacoustic design exploration of launch-pad flame deflector using large-eddy simulation. *Journal of Spacecraft and Rockets*, 53(4):751–758, 2016.

[2] S Rodrigues, Pavol Bauer, and Peter AN Bosman. Multi-objective optimization of wind farm layouts–complexity, constraint handling and scalability. *Renewable and sustainable energy reviews*, 65:587–609, 2016.

[3] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[4] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[5] Shumeet Baluja. Deep learning for explicitly modeling optimization landscapes. *arXiv preprint arXiv:1703.07394*, 2017.

[6] 池上 貴志. 用語解説（第 87 回テーマ：多目的最適化問題）／電力・エネルギー部門誌 2018 年 6 月号目次. **電気学会論文誌 B（電力・エネルギー部門誌）**, 138(6):NL6_6–NL6_6, 2018.

[7] Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE transactions on evolutionary computation*, 13(2):284–302, 2008.

[8] Antonio Benítez-Hidalgo, Antonio J Nebro, José García-Nieto, Izaskun Oregi, and Javier Del Ser. jmetalpy: A python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, 51:100598, 2019.

[9] Ryoji Tanabe and Hisao Ishibuchi. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing*, 89:106078, 2020.

[10] Qingfu Zhang, Wudong Liu, and Hui Li. The performance of a new version of moea/d on cec09 unconstrained mop test instances. In *2009 IEEE congress on evolutionary computation*, pages 203–208. IEEE, 2009.

[11] C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1157–1163, 2006.