No Free Lunch Theorem
—理想の＊＊の探し方—

東京大学大学院新領域創成科学研究科基盤情報学専攻
矢吹太朗　伊庭斉志
{yabuki,iba}@miv.t.u-tokyo.ac.jp
http://www.miv.t.u-tokyo.ac.jp/∼yabuki/

見つかったら、その人に捧げる

There ain't no such thing as a free lunch.
Robert Heinlein, *"The Moon is a Harsh Mistress"*

1　FAQ（よくある質問）

我々は進化論的計算、特に遺伝的アルゴリズムや遺伝的プログラミングをさまざまな問題に応用する研究をしているが、次のような質問をよく受ける。

1. なぜその問題に遺伝的アルゴリズムを適用したのか。
2. そもそも遺伝的アルゴリズムというのはよい探索手法なのか。

最初のものは適切な質問だが、満足に答えることは今のところできない。他のさまざまな探索アルゴリズムと比較すれば、ある程度答えられるだろうが、応用の場合にそのような比較をいつもしているのは無駄である。応用において重視されるべきは、結果であって手法ではない。

後の質問については、実はそれが適切な問いかどうかをまず考えなければならない。そもそも、どんな問題にも適用できるよい探索アルゴリズムというのはあるのだろうか。ここではこの疑問について考察する。理解を助けるためにたとえ話を用いているが、政治的に正しくあるために、＊＊という記号を使っている。この部分は何か適当なもの（女や男、食材、ワイン、家具、家など）で置き換えて読んでほしい（いずれにしても" 好み "などと言っている時点で政治的に正しくないのかもしれない）。" ＊＊に入れる政治的に正しい例 "で置き換えるのもいいかもしれない。

理想の＊＊の探し方を研究したいんです。でも、自分にしか役立たない方法だと論文になりませんし、誰も聞いてくれないじゃないですか。だから、誰にでも使える方法を見つけたいんですよ。題して、" 理想の＊＊の探し方–好みの違いを超えて–"

どんな問題も効率よく解けるようなアルゴリズム（聖杯）の存在を暗に仮定しているような研究は過去に多く行われてきた [6]（キリストが最後の晩餐で用い、後に十字架の下で彼の血を受けたと言われる聖杯の名がなぜ万能のアルゴリズムを指すのに使われるのかは不明）。「問題 A に関して、探索アルゴリズム B は他のいくつかのものより効率がよい」という研究は多くあるが、発表されるアルゴリズムは問題ごとに違っていて、聖杯と呼べるものは見つからなかった。そして 1995 年に Wolpert と Macready が no free lunch theorem (NFL) を発表し、これが聖杯の存在へのアンチテーゼとされたのである。

ここではこの NFL について考察する。以下では、ふつうは多項式時間で処理できるなどのように絶対的な意味で使われる" 効率がよい "という言葉を、相対的つまり" 他の方法より効率がよい "という意味で用いる。さらに言えば、計算資源に関する効率は考察しない。そのため、扱う計算モデルを厳密にする必要はないかもしれないが、一応チューリング・マシンのもののみと考えておこう。量子計算のような計算モデルはここでの考察の対象にはならないだろう。

2　NFL の簡単な例

世の中にはいろんな好みを持った人がいるじゃないですか。僕がかわいいと思う＊＊をなんとも思わない人がいると思えば、ちょっとどうなんでしょうっていう＊＊を熱烈に愛しちゃったりする人もいるわけです。それでも、考えられるかぎりすべての好み

に対応できる“ 理想の＊＊の探し方 ”はありませんかねえ。

次のような問題 Ω を考えよう。

> 問題 Ω
>
> 0 から 7 の整数に対して 1 から 5 の整数のいずれかを返すような関数 f がある。f を 4 回呼んでその最大値を推測する最も効率のよい方法はどのようなものだろうか。

直観的にはそのような方法は無さそうに見える。実際にそのことを数学的に示したのが次の定理である。

> 定理 (No Free Lunch)
>
> すべての評価関数に適用できる効率のよいアルゴリズムは存在しない。

“ すべての評価関数 ”というのは上の例で言えば“ すべての f ”ということである。この定理を証明する前に、まずよく知られた探索アルゴリズム [5] を挙げて、探索とはそもそもどのようなものなのかを説明しよう。

3　探索アルゴリズム

“ 探索 ”というのは問題の解の候補の中からよいものを探し出すことである（同語反復という感じだが)。ここでは次のように、評価関数が定義された問題を解く過程のことを探索と呼ぶことにする。

解の候補の有限集合を X、その要素のひとつを x とする。評価関数 f は X から有限集合 Y への写像である（Y の要素のひとつを y で表す)。Y の最大値を与えるような x が問題の解で、それを見つけたいのである。

このような探索問題を解くためのアルゴリズムには大きく分けて次の 2 つがある。

- A^* アルゴリズムのように知識を用いて解を構成するもの（適切なヒューリスティックスが必要）
- ある解の候補 x を与え、その評価・改良を繰り返しならが解を見つけるもの（候補の評価値のみを考慮して候補を改良する）

ここで扱うのは後者である。その中の代表的なものを次に挙げる。

3.1　山登り探索

山登り探索は次のような探索アルゴリズムである。

1. 適当な候補 x をランダムに生成する。
2. x の近傍でもっとも評価が高い点に移動することを繰り返す。
3. 極大値に到達したら 1 に戻る。

十分な回数繰り返せば、いつかは最適解を見つけることができるだろう。探索空間に局所的最大や高原、峰が多くあるとうまく探索できない。

問題 Ω において図 1 のような f が与えられたとしよう。我々はこの答えを知っているが、探索アルゴリズムは何も知らないところから探索を始める。適当な解の候補を $x = 2$ とする。$f(2) = 2$ である。$x = 2$ の近傍を調べる。$f(1) = 2, f(3) = 1$ であるから、山を上るために $x = 1$ に移動する。$f(0) = 3$ であるから $x = 0$ に移動するが、これで 4 回 f を呼び出したため探索は終わりである。見つかったのは最大値（$f(4) = 4$）ではなく極大値でだった。

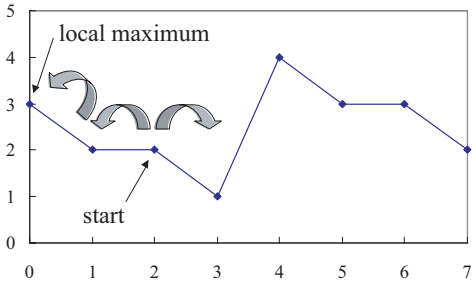

図 1: 山登り法の例

たまたま好みに合う＊＊が見つかったとします。その周辺で最も好みに合う＊＊を探します。そういうことを繰り返せばいつか理想の＊＊にめぐり合えるように思えます。（例えば＊＊が“ 女性 ”で周辺が“ 合コン友達 ”の場合には、“ 幹事 MAX ”(合コンでは幹事が一番いい）という経験則があって、僕は山を登れません。僕と正反対の好みを持つ人にとってはいい話のように思うかもしれませんが … 。絶対的な女性の基準の有無については何も言っていません。あくまで僕の好みの話です。）

3.2 焼きなまし法

焼きなまし法は液体が凍る過程にヒントを得た探索アルゴリズムである。

1. 適当な候補 x をランダムに生成し、$T = 0$ とする。
2. 現在の評価と近傍の評価の差を ΔE とする。
3. $\Delta E > 0$ ならば近傍に移動、そうでなくても $\exp(\Delta E/T)$ の確率で近傍に移動する。
4. T をある決まった方法で減らし（冷却）、2 に戻る。

T が大きいうちは探索空間を動き回るが、T が小さくなるにつれて、だんだん動かなくなっていく。T の減らし方には自由度があって、T を十分ゆっくり下げるなら、最適解を見つけることができる。

3.3 遺伝的アルゴリズム

遺伝的アルゴリズムは生物の進化の過程にヒントを得た探索アルゴリズムである。

1. 適当な候補 x をランダムに複数生成する。
2. 評価の高い候補を選択する。
3. 選択された候補をもとに次の候補を生成し、2 に戻る。

個体の数や候補の選択方法、次の候補の生成方法（交叉や突然変異）に自由度がある。
みんな結局これに似たことをやっているわけですよね。

3.4 ランダム探索

ランダム探索は文字通り、X からランダムに候補 x を選んで $f(x)$ を評価することを繰り返す方法である（探索アルゴリズムとは言えないかもしれない）。先の問題 Ω ではどんな f についても同じような探索効率になるのは明らかだろう。
「これから 5 番目にここを通る女の子をナンパしろ」みたいなのがこれです。試したことはありません。

3.5 考察の対象になること

何か探索したい問題があるとしよう（たとえば東京での巡回セールスマン問題 (TSP)）。こういうものはもっと一般的な問題（たとえば TSP）の一例である場合が多い。そこで以下では、一般的な問題を “問題”、その一例を “インスタンス” と呼ぶことにする。ふつう我々は個々のインスタンスを解決するものではなく、問題つまりすべてのインスタンスを効率よく解決する探索アルゴリズムを考えようとする（そうでない場合もある）。

探索においては、まずインスタンスの解の候補を空間（候補空間）の点に対応付けるようなコーディング方法を決める。次に解の候補つまり点の評価値を返すような評価関数を定義する。これによって探索空間が定まる。そして探索アルゴリズムによって実際に探索する。以上をまとめると図 2 のようになる。NFL が直接かかわるのは点線の内側である。塗りつぶされた部分は人間が工夫できる［しなければならない］ところである。

探索について記述するために、以下のような記号を導入する。NFL では評価関数を評価した回数 m に対する効率を考えるため、探索履歴 d_m^x には重複がないようにしなければならない。

X: 候補空間
Y: 評価値空間
$X \otimes Y$: 探索空間
a: 探索アルゴリズム
f: 評価関数 $(X \mapsto Y)$
F: すべての評価関数の集合
m: 評価関数を呼んだ回数
$d_m = \{d_m^x, d_m^y\}$: 探索履歴
$d_m^x = \langle x_1, x_2, \cdots, x_m \rangle$ （if $i \neq j$, $x_i \neq y_j$）
$d_m^y = \langle y_1, y_2, \cdots, y_m \rangle$

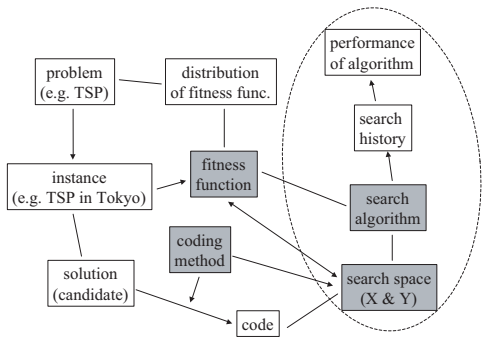

図 2: 考察の対象

そもそも＊＊を評価できない場合はどうするんだろうなどいうのは考えすぎです。あくまでたとえ話で

すから、一目見たら評価できるんだと思ってください（評価できない＊＊の評価値を低くするのも哀しいですし）。

4 証明のアイディア

どういう＊＊と知り合ってきたかでその人の人生を評価しましょう。いい＊＊にめぐり合えたら勝ち、めぐり合えなかったら負けという感じに。単純すぎると思う方は他の評価方法を採ることもできます（看護婦ははずせないとか、よくわかりませんが）。

アルゴリズム a の効率 $Perf$ は探索履歴 d_m^y で決まると考える。図 3 のように d_m^y は探索アルゴリズム a と評価関数 f、それを呼び出す回数 m を与えれば決まる。つまり、

$$\begin{aligned} Perf &= Perf(d_m^y) \\ &= Perf(a, f, m) \end{aligned}$$

である。

＊＊を探す場合は、探索アルゴリズムと好みを与えるだけではその人がどんな人生を送るかは決まりそうにありません。出会いはもっと確率的ですし（僕にはそう見えます。もっと運命と思ったほうがいいのかもしれませんが）、自分の行動が世界を変えることもありえますから。しかし、ここではあまり悩まないで話を単純なままにしておきましょう。

平均効率を求めるために、すべての f についての和をとる。図 4 からわかるように d_m^y と a を与えれば d_m^x が決まるため、f もある程度決めることができる。つまり、

$$\begin{aligned} \sum_f Perf(a, f, m) &\propto \sum_{d_m^y} Perf(d_m^y, a) \\ &= \sum_{d_m^y} Perf(d_m^y) \end{aligned}$$

となって、和はアルゴリズム a によらなくなる。

すべての好みを考えているので、勝手に持ってきた＊＊遍歴でも、そういう人生を歩むと思われる好みの持ち主がいるのです。ですから、すべての＊＊遍歴を考慮することは、すべての好みを考慮することに近いのです。その人がどんな＊＊に出会ったかを見れば、その人がどんな好みなのかは見当がつくでしょう。

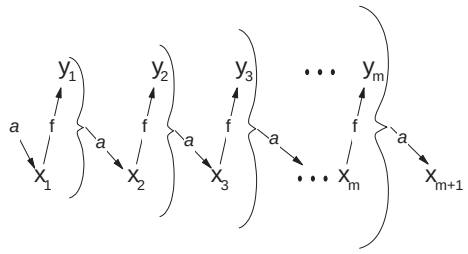

図 3: a と f を与えれば探索履歴が決まる [6]。

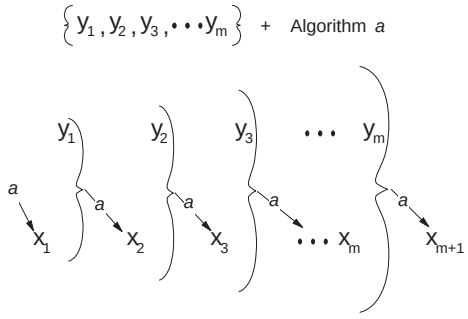

図 4: a と d_m^y を与えれば d_m^x は決まり、f も制限できる [6]。

5 簡単な証明

NFL を簡単に証明しよう [6]。a, f, m を指定すると、図 3 のように d_m^y が決まる。つまり、$d_m^y = d_m^y(a, f, m)$ である。また、図 4 のようにして、任意の d_m^y と a から d_m^x を決めることができる。$d_m = \{d_m^x, d_m^y\}$ には m 組の (x_i, y_i) があるが、d_m^x の中に同じ点がないため、それらをすべて満たすような関数は、F の中に $|Y|^{|X|-m}$ 個ある (ただし $m \leq |X|$)。

求めたいのは、アルゴリズムの効率 $Perf(d_m^y)$ のすべての f に関する平均である。そのために $\sum_{f \in F} Perf(d_m^y(a, f, m))$ を計算する。この f に関する和は d_m^y に関する和に置き換えることができる。なぜなら、考えられる d_m^y は全部で $|Y|^m$ 個あって、そのひとつひとつに対して、d_m^y と a を与えたときに得られるような d_m を満たす f は $|Y|^{|X|-m}$ 個あることがわかるからである。確かに、$|Y|^m \cdot |Y|^{|X|-m} = |Y|^{|X|}$ となって、すべての f を尽くしていることがわかる。

よって、

$$\sum_{f \in F} Perf(d_m^y(a, f, m))$$

$$= \sum_{d_m^y \in Y^m} \sum_{\substack{d_m^y \text{と } a \text{ から決まる} \\ d_m \text{を満たす } f}} Perf(d_m^y(a,f,m))$$
$$= \sum_{d_m^y \in Y^m} |Y|^{|X|-m} Perf(d_m^y)$$

となるが、これは a によらない。結局、次の定理が示されたことになる。

定理 ((NFL1))

$$\forall a, a', m \ \frac{1}{|Y|^{|X|}} \sum_{f \in F} Perf(d_m^y(a,f,m)) = \frac{1}{|Y|^{|X|}} \sum_{f \in F} Perf(d_m^y(a',f,m)) \quad (1)$$

どうも納得できないんですが、要するにすべての好みに対応できる理想の＊＊の探し方はないということになってしまったみたいです。

6 確率理論を用いた証明

Wolpert らによる確率理論を用いた証明を紹介しよう。ここでは扱わないが、この証明が探索アルゴリズムを扱うフレームワークを与えるかもしれない [9]。

証明には確率理論の次の公式を用いる。

1. $P(A,B) = P(A|B)P(B)$（条件付確率の定義）
2. $P(A,B|C) = P(A|B,C)P(B|C)$
3. A と C が独立なら $P(A|B,C) = P(A|B)$
4. $A_i \cap A_j = \phi$ かつ $\sum_i P(A_i) = 1$ ならば、
   (a) $P(E) = \sum_i P(E|A_i)P(A_i)$
   (b) $P(E|F) = \sum_i P(E|F,A_i)P(A_i|F)$

f, m, a を指定したときに探索の結果が（勝手に与えた）d_m^y になる確率 $P(d_m^y|f,m,a)$ のすべての f についての平均

$$\frac{1}{|Y|^{|X|}} P(d_m^y|f,m,a) \quad (2)$$

が a によらないことを、m に関する数学的帰納法で示す。

$m = 1$ のとき、

$$P(y_1|f, m=1, a) = \delta(y_1, f(x_1))$$

である。ただし、x_1 は a で与えられる。$f(x_1) = y_1$ となるような f は F の中に $|Y|^{|X|-1}$ 個あるため、

$$\sum_f P(y_1|f, m=1, a) = |Y|^{|X|-1}$$

となって、(2) は a によらないことがわかる。

次に m のときに (2) が a によらないと仮定する。公式 2 を使うと、

$$\sum_f P(d_{m+1}^y|f, m+1, a)$$
$$= \sum_f P(y_{m+1}|d_m^y, f, m+1, a) P(d_m^y|f, m+1, a)$$

（公式 4b を使う）

$$= \sum_f \sum_{x_{m+1}} P(y_{m+1}|d_m^y, f, m+1, a, x_{m+1}) \times P(x_{m+1}|d_m^y, f, m+1, a) P(d_m^y|f, m+1, a)$$
$$= \sum_f \sum_{x_{m+1}} \delta(y_{m+1}, f(x_{m+1})) \times P(x_{m+1}|d_m^y, f, m+1, a) P(d_m^y|f, m+1, a)$$

（公式 4b と 3 を使う）

$$= \sum_f \sum_{x_{m+1}} \sum_{d_m^x} \delta(y_{m+1}, f(x_{m+1})) P(x_{m+1}|d_m^y, a, d_m^x) \times P(d_m^x|d_m^y, f, m+1, a) P(d_m^y|f, m+1, a)$$

（$P(x_{m+1}|d_m^y, a, d_m^x) = \delta(x_{m+1}, a(d_m))$ と公式 2、$P(d_m|f, m+1, a) = P(d_m|f, m, a)$ を使う）

$$= \sum_f \sum_{d_m^x} \delta(y_{m+1}, f(a(d_m))) P(d_m|f, m, a) \quad (3)$$

f についての和を先にとる。そのために、f を $x \in d_m^x$ の部分 g と $x \notin d_m^x$ の部分 h に分けて考える。そうすると (3) は、

$$= \sum_{d_m^x} \sum_h \sum_g \delta(y_{m+1}, h(a(d_m))) P(d_m|g, m, a)$$
$$= \sum_{d_m^x} \sum_g |Y|^{|X|-m-1} P(d_m|g, m, a)$$

（g についての和を f についての和に直す。ひとつの g に対して $|Y|^{|X|-m}$ 個の f がある。公式 2 と 4b を使う）

$$= \sum_{d_m^x} \frac{1}{|Y|} \sum_f P(d_m|f, m, a)$$
$$= \frac{1}{|Y|} \sum_f P(d_m^y|f, m, a)$$

となるが、仮定によりこれは a によらない。以上によって、数学的帰納法により (2) が a によらないことが示された。まとめると、

定理 (NFL2)

$\forall a, a', m$

$$\frac{1}{|Y|^{|X|}} \sum_f P(d_m^y|f, m, a) = \frac{1}{|Y|^{|X|}} \sum_f P(d_m^y|f, m, a') \quad (4)$$

ということになる。

7 考察

7.1 効率の定義

どんな＊＊に出会ったかの履歴は重要ではなくて、最後にいい＊＊にめぐりあえればいいと思う人もいるでしょう。そういう場合でも NFL は成り立っています。

NFL において、探索アルゴリズムの効率 $Perf(d_m^y)$ の詳細については何も仮定していない。実際、何によって効率を測るかには柔軟性がある。しかしながら、ここでいう効率は、異なる x に対して f を評価した回数 m に関するものだという点で一般に言われる効率とはかなり異なっている。

ふつうアルゴリズムの効率として考えられるのは、必要とする計算時間や記憶容量に対してどの程度よい解が得られるかということである。たとえば先に挙げたランダム探索、山登り探索、焼きなまし法、遺伝的アルゴリズムなどは、同じ点を複数回評価する可能性がある。NFL においてはそのようなことは無視している。計算時間と m の関係がたとえ線型だとしても探索アルゴリズムごとに違っているだろうからふつうに使われる効率に簡単に変換することはできない。

探索アルゴリズムを修正して同じ点を評価しないようにすることはできるが、そのためには一度評価した点をすべて記憶しておかなければならないうえに、そうしたところでふつうに使われる効率がわかるようになるわけではない。

いずれにしても NFL は計算資源に関しての効率については何も主張していない。また、そのような効率をすべての f について平均しようとしても、得られる結果は意味のないものになるだろう。そのような場合にはランダム探索の効率がよくなる。なぜなら、ランダム探索は重複せずにたくさんの点を評価できるからである。さらに言えば、$x = 0, 1, 2, \cdots$ と順に評価していくアルゴリズムのほうが乱数を計算する必要がないぶん効率がよいことになるだろう。

ここでは紹介しないが、すべての評価関数についての平均ではなく分散など他の統計量を用いて効率を定義しても、その平均はアルゴリズムによらない [4]。

7.2 メタ・レベルの探索

NFL の主張は、すべての評価関数を効率よく最適化する探索アルゴリズムはないということであった。このことから、すべての評価関数 f について、効率のよいアルゴリズムを効率よく見つけるようなアルゴリズムも無いように見える。もしそのような効率のよい探索アルゴリズム b があれば、その b によって効率のよい探索アルゴリズム a を求め、a によって f を最適化することができることになって NFL に反することになるからである。しかしながら、NFL はそこまでの主張はできない。NFL の対象になるのはあくまで評価値だけをみて探索するアルゴリズムのみだからである。

7.3 確率的なアルゴリズムの場合

先に示した証明では、探索アルゴリズムは決定的だと仮定されていた。それに対して、ランダム探索、山登り探索、焼きなまし法、遺伝的アルゴリズムはいずれも確率的なアルゴリズムである。これらの探索アルゴリズムにも NFL を適用できるのかと疑問に思うかもしれない。しかし、現在のコンピュータ上にアルゴリズムを実装する場合には、乱数生成器もアルゴリズムの一部だと思えばアルゴリズムは決定的になるため問題ないだろう（実は、本当に確率的な場合にも定理は簡単に拡張できる [8]）。

7.4 探索空間と評価関数を作り方

問題が与えられたときにそのインスタンスを解決するための探索空間と評価関数、コーディング方法（以下、3 要素と呼ぶ）の作り方はさまざまである。

例えば問題 Ω のための探索空間の場合、X を 10 進数（0 ～ 7）で表して、それを 1 次元の空間（図 5）に並べることもできれば、2 進数で表して 3 次元空間（図 6）に配置することもできる。$x = 0$ に局所的最大値を持つ図 5 と比べれば、そのような点を持たない図 6 のほうが山登り探索にとっては扱いやすいことが予想される（もちろん 1 次元の候補空間のほうが探索しやすいような問題 Ω のインスタンスもあるだろう。これは NFL に似た話になっている）。つまり、問題に対して 3 要素を固定してしまうと、インスタンスによっては探索が困難になるということが起こりうる。

それならば、インスタンスが与えられたときに、図 7 の左側のような（以下、富士山型と呼ぶ）探索

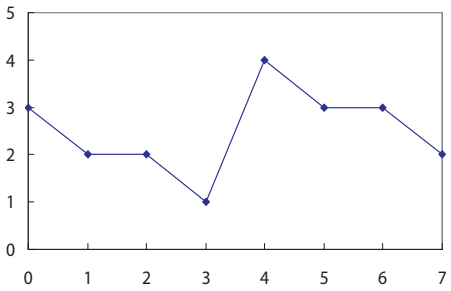

図 5: 問題 Ω の探索空間 (10 進表記)

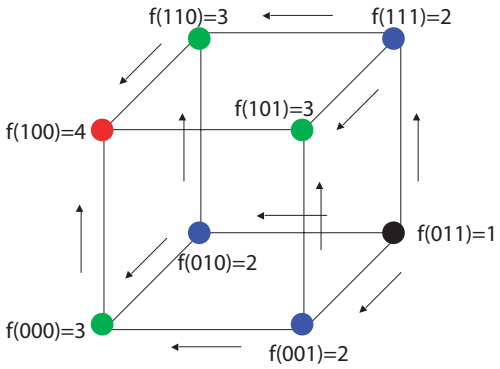

図 6: 問題 Ω の探索空間 (2 進表記)

空間を簡単に（たとえば多項式時間で）作る方法があればよいように思える。しかしながら、そのような方法はおそらく存在しない。

もしそういうことが可能だとすると、NP 完全問題にそれを適用して富士山型の評価関数を作ることができる（NP 完全問題とは候補が解かどうかを多項式時間で決定できる問題の中でもっとも難しいと思われている問題である [5]）。皮肉なことに、富士山型の評価関数はここで扱っているような探索アルゴリズムによってではなく統計的な方法によって多項式時間で最適化できることがわかっている [3]。そうすると、NP 完全問題を多項式時間で解けることになり、一般に正しいと信じられている仮説“ P≠NP 完全 ”に反することになってしまう。

ただし、このことが富士山型の評価関数を作ることができないことを意味しているわけではない。実際それは可能であることが示されている [7]。ただ、それが多項式時間ではおそらく不可能だということである。

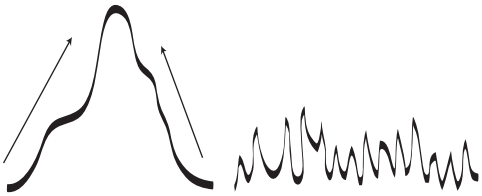

図 7: どちらが探索しやすいかは明らかだろう。

8 聖杯はどこへ消えた？

すべての好みを考慮するっていうのはやっぱりやりすぎなんじゃないでしょうか（これがこの記事の中で最も政治的に正しくない発言でしょう）。世界の人口は有限ですし。たとえ人類が永遠に続いたとしても …

NFL の主張は、すべての $f \in F$ を効率よく最適化できるようなうまい話は存在しないということであった。そして確かに問題 Ω はその一例になっている。実は F は探索空間で定義できる関数すべてよりも小さく取ることができることがわかっているが、その場合も F が組替えについて閉じているかぎり NFL は成立する [2]。

神性は多少損なわれてしまったわけだが、聖杯を求める円卓の騎士はまず次の 2 つの可能性（独立ではない）を考えなければならない。

- ある問題のインスタンスの中で実際に解きたいものに対応する評価関数の集合が F を覆い尽くさない可能性
- 問題のすべてのインスタンスに対応する評価関数の集合が F を覆い尽くさないような探索空間を作る可能性

おそらくこのどちらもふつうは成り立っているだろうと思われる。そのような問題（の一部）について最適な探索アルゴリズムを与えるためには何が必要だろうか。図 2 に示したように、探索アルゴリズム以外で人間が工夫しなければならないもの（探索空間と評価関数、コーディング方法）は互いに関連しており、最適なものを独立に作ることはできないだろう。先にみたようにインスタンスごとに探索空間を変えたほうがよい可能性もある（自己言及的に変化するアルゴリズムはうまく探索できるだろうか）。

そこでまず、探索アルゴリズムと探索空間、評価関数を与えたときに、探索の効率をある程度推定する指標を考えたい。もしそのような指標があれば、

それを用いて先にあげた 3 つの要素を同時に考慮しながらよりよい組み合わせを探索することができるだろう。

探索の過程で探索空間や評価関数が変化しないなどの NFL における仮定を再検討することでも、何らかのヒントが得られるかもしれない。

9 まとめ

効率のよい探索アルゴリズムを求めることへのアンチテーゼとしての NFL とその証明、その意味する［しない］ことを見てきた。NFL は、
「万能の探索アルゴリズムは存在しない」
と表現されることが多いが、このような言い方には誤解があり、正しく表現するには多くの仮定を明示する必要があることもみてきた。最後にそれらをまとめておこう。

- 候補空間、評価値空間は有限である。
- 探索履歴のみを見るような探索アルゴリズムに限る。
- アルゴリズムの効率は評価関数の評価回数に関するものである（同じ点を何度も評価する可能性を考慮しない）。
- すべての評価関数に対して平均した場合の話である（すべての問題に対して平均したという話ではない）。
- アルゴリズムが探索空間や評価関数を変化させることはないとしている。

見つかったけど [1]**、どうしていいかわからないって?**

TANSTAAFL.

参考文献

[1] In *MORE*, No. 287, pp. 113–120. SHUEISHA, 5 2001.

[2] Schumacher C., Vose M.D., and Whitley L.D. The no free lunch and problem description length. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 565–570, 2001.

[3] R. Das and D. Whitley. The only challenging problems are deceptive. In *Proceedings of the Fourth International Conference of Genetic Algorithms*, pp. 166–173, 1991.

[4] N. J. Radcliffe and P. D. Surry. Fundamental limitations on search algorithms: Evolutionary computing in perspective. *Lecture Notes in Computer Science*, Vol. 1000, p. 275, 1995.

[5] Stuart J. Russell and Peter Norvig. *Artificial Intelligence—A Modern Approach*. Prentice–Hall, Inc., 1995. N. G[WFgAv[‘lHm˙o, 1997.

[6] Oliver John Sharpe. *Towards a Rational Methodology for Using Evolutionary Search Algorithms*. PhD thesis, University of Sussex, 2000. http://www.cogs.susx.ac.uk/users/olivers/.

[7] Michael D. Vose and Gunar E. Liepins. Schema disruptions. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 237–243, 1991.

[8] David H. Wolpert and William G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, 1995.

[9] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 67–82, 1997.