Generative Design における最適化手法

DX 事業統括　コンサルティングサービスグループ
柳澤　幸信

1. はじめに

近年、生成 AI(=Generative AI)が話題となっている。人間に代わって AI が文書を書き、絵を描き、作曲を行うようになり、日進月歩の勢いで進化している。この調子でいくと、近い将来、土木や建築のデザイン業務も代替してくれるかもしれないと思う反面、現時点ではなかなかそのレベルには達していない。この分野は、理論や正確性が要求される。学習データから確率論的に生成するだけでは、きっちりしたデザインを創造することは、なかなか難しそうである。

似た言葉に、ジェネレーティブデザイン(=Generative Design)というものがある。これは、デザインを生み出すものであるが、デザインの生成方法、つまりアルゴリズム自体は人が設計し、コントロールするものである。

本稿では、このジェネレーティブデザインの手法の1つとして「遺伝的アルゴリズム」を使う例と、そのポイントを紹介する。

2. ジェネレーティブデザインの流れ

簡単にジェネレーティブデザインの処理の流れを説明する。

設計するものに対して、その要因をいくつかの要素に分解し、パラメータとして定義する。このパラメータのことを**説明変数**と呼ぶ。例えば、簡単なビル状の建物設計を考えた場合、説明変数として間口、奥行き、高さの3つを定義するようなイメージである。

次に、そのパラメータから設計を行うアルゴリズムを書く。パラメータを細かくたくさん定義すれば、その分ロジックは複雑になるが、多種多様なデザインを生み出せるものとなる。先ほどの例では、間口・奥行き・高さを持った直方体を生成するようなイメージである。

最後に、出来上がった設計に対して良し悪しを判断する指標を算出する**アルゴリズム**を作成する。良し悪しは、本来、人が1つ1つ主観的な基準も含めて判断するものだが、これを客観的にプログラムが判断できるよう指標化するのである。先ほどの例では、床面積や容積、収容人数、この直方体が作る影や大きさなどである。容量や数量などはルールに従いきっちり計算できるが、快適性や好感度などの主観に依存するものは指標化しにくい。そのようなものが設計上で重要であれば、客観的な指標をいろいろ組み合わせて「快適度」を代弁するような近似指標を捻出する必要がある。

なお、この指標のことを**目的関数**または**フィットネス関数**と呼ぶ。

ここまで出来ると、あとは最適化アルゴリズムを用いて、良い指標が出るようなパラメータセットを探索すればよい。

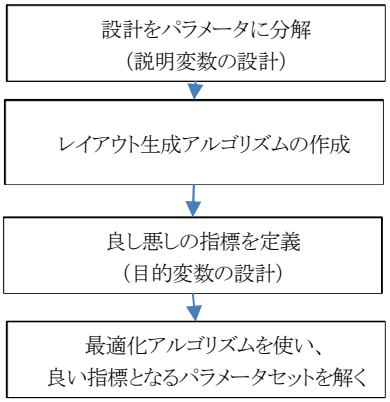

図 1　Generative Design の考え方

3. 遺伝的アルゴリズムによる最適化

3. 1　遺伝的アルゴリズムの狙い

説明変数からアルゴリズムを用いてデザインを設計するといった場合、説明変数の数と各説明変数の取り得る値の組み合わせの数だけデザインが存在する。

例えば、先ほどの簡単なビルの例に当てはめれば、説明変数として間口、奥行き、高さの3つがあり、それぞれ 3m,6m,9m の3つの値をとると仮定すれば、3×3×3 で 27 通りのデザインが生まれる。

27 通り程度のデザインであれば、全数を設計してそのデザインや指標の良し悪しを人が判断すればよい。

しかし、実際のデザインケースでは、設計に影響する説明変数はもっと大量にあり、またその説明変数が取り得る値は実数のような無段階のレンジであることが多い。こうなると、パラメータ（＝説明変数の値）の組み合わせは、数兆通りを超えるような無限とも言える数となり、全通りを試すことは現実的な時間内では困難である。

この課題解決の助けになる手法の1つとして、遺伝的アルゴリズムによる最適化がある。

遺伝的アルゴリズムは、ヒューリスティック(発見的)な手法であり、得られる解が最も期待されるものであるという「正確性」を保証しない代わりに、その近似解を導くための時間が短くて済むという特徴がある。ここで「正確性を保証しない」という表現から実用性を疑問視する読者もおられると思うが、実際に使ってみると、実用範囲で解が得られることが多く、ほぼ最適解と言える解になることもある。もちろん、正真正銘、厳密な正解が欲しいというケースにおいては、他の手法を使うべきである。

3. 2　遺伝的アルゴリズムの仕組み

遺伝的アルゴリズムは、大まかに以下のような処理を経て近似解を得る。

(1)　説明変数のパラメータの組み合わせをランダムで複数組生成する。（このパラメータセットの集団を Population と呼び、特に最初のセットを初期 Population と呼ぶ）

(2)　初期 Population それぞれのパラメータでデザインを作成し、良し悪しの指標を計算する。

(3)　指標の良いパラメータセット同士の一部を交換（＝交配と呼ぶ）したり、一部のパラメータを改変（＝変異と呼ぶ）したりを複数実施し、新たな設計が生まれるパラメータセットを生成する。

(4)　元のパラメータセットと新たに生成したパラメータセットの全体から、指標が上位のパラメータセットを残す（＝この残す行為を選択と呼び、選択によって新たに生成された集団を子世代の Population と呼ぶ）。

(5)　(4)の子世代のパラメータセットに対して(3)(4)

を何度も繰り返し(=世代を進める)、そして次第に、より良い指標を持つデザインの集合が残る。

考え方の基本にあるのは、良いデザインは、良いデザインを生む「良い素質」のパラメータ値を持つはずで、良いデザインが持つパラメータの一部を交換や合成すれば、さらに良いデザインが生まれる可能性がある、ということである。

このとき、設計パラメータである説明変数に対して、その説明変数から生成されるデザイン結果の指標、つまり目的変数に有意な変化が生じることが重要となる。

4. ケーススタディ

ここからは、簡単な具体例を用いてジェネレーティブデザインの流れや考え方を説明する。

4.1 試行環境

今回はレイアウトデザインを扱う事とし、CAD 的な図形処理がビジュアルプログラミングとして簡単に扱え、遺伝的アルゴリズムをアドインで利用できる環境を選んだ。よく知られているものとして、Autodesk 社製 BIM ソフト Revit とそれに付属する Dynamo や、日本国内ではアプリクラフト社が販売している CAD ソフト Rhinoceros3D とそれに付属するプログラミングツール Grasshopper 等がある。今回は後者の Grasshopper を利用した。また、最適化処理のために、Wallacei というアドインを利用した。

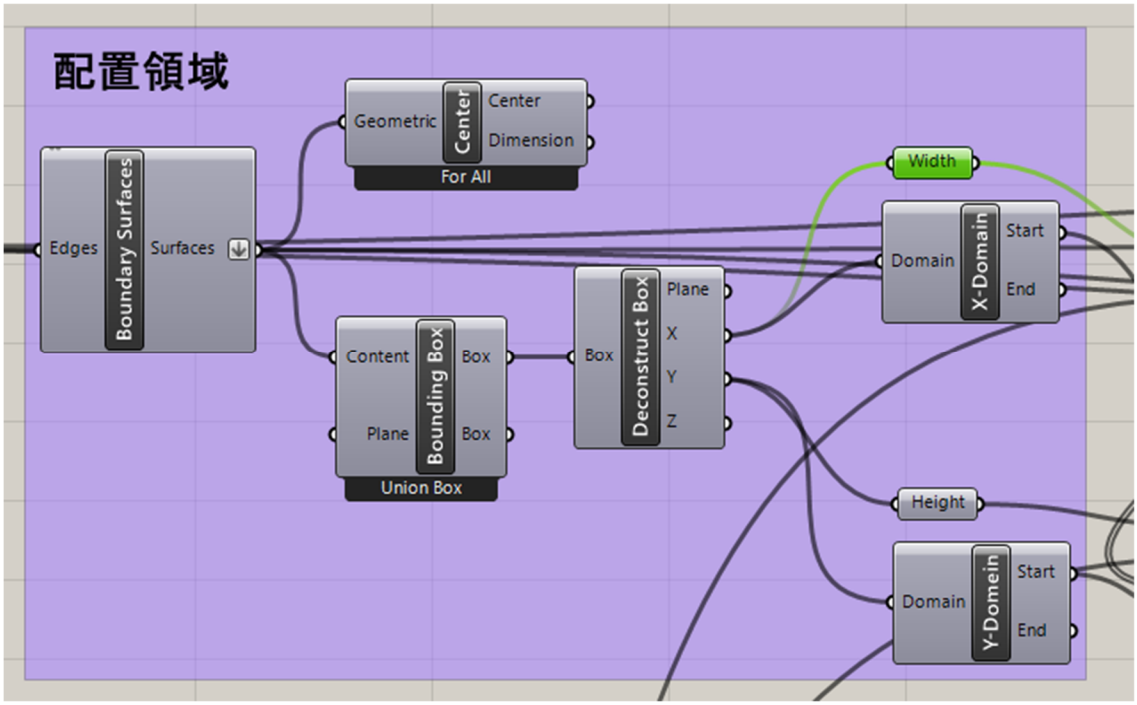

図 2 Grasshopper のプログラムの様子

4．2　レイアウトデザインの題材

ある領域の内側に5つの長方形をなるべく距離を離しつつ、長方形間の距離はなるべく等間隔に配置する場合を考える。

表 1　題材の具体的な条件設定

No	条件
1	長方形は領域からはみ出さないこと。
2	長方形同士は重ならないこと。
3	5つの長方形はお互いになるべく遠ざける。
4	5つの長方形間はなるべく等距離にする。

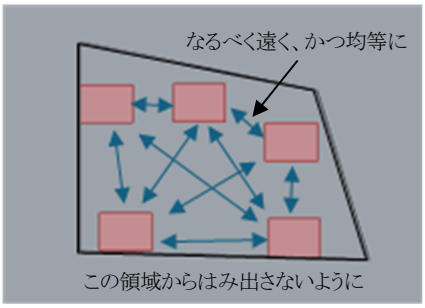

図 3 題材の要件イメージ

5．試行1

4.2 のような題材に対して、最適化の要素を具体的に設計してみる。

5．1　パラメータ(説明変数)の設計

長方形を配置するということで、その(X,Y)の座標をパラメータとする。(X,Y)が5組あるので、合計 10 個となる。

なお、座標の範囲は、X は領域の幅、Y は領域の高さとする。領域は矩形ではないので、この範囲条件だけでは領域からはみ出してしまう可能性がある。はみだし有無は別途判断させることにする。

5．2　配置設計アルゴリズムの設計

パラメータを(X,Y)としたので、それぞれを座標に合わせて配置するだけである。

5．3　良し悪しを決める指標(目的関数)の設計

4つの条件から4つの指標として考えられるものを選出してみた。

表 2　目的関数(試行1)

No	目的関数とその判断基準
1	＜条件 1 より＞領域からのはみ出し有無。はみ出す配置より、はみ出さない配置を良しとする。
2	＜条件 2 より＞長方形同士の重なり有無。重なる配置より、重ならない配置を良しとする。
3	＜条件3より＞長方形同士の距離の総和。より大きいものを良しとする。
4	＜条件4より＞長方形同士の距離の標準偏差。より小さいものを良しとする。

座標パラメータを決め、それをもとに配置し、各種指標を計算するようにしたプログラムが図 4 である。

（紙面が限られているため縮小して見づらくなっているが、プログラム規模感の参考程度に確認して頂きたい。）

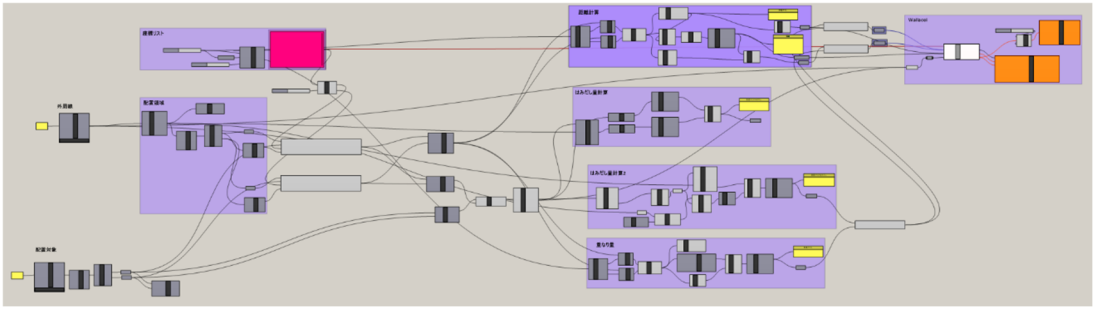

図 4 最適化のビジュアルプログラム

これを Wallacei アドイン用いて最適化を実行させてみる。実行に際して Population を 50，世代数を 100 と設定してみた。この場合、50×100=5000 のデザイン生成を試行することを意味する。実行に要する時間は数分であった。

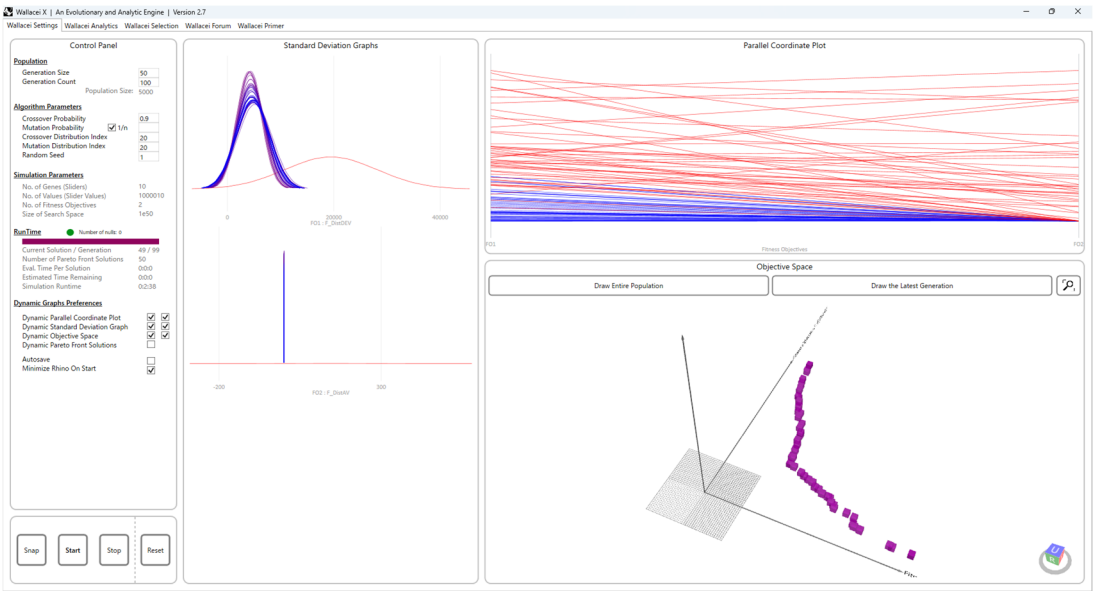

図 5 Wallacei の動作中の画面

この試行では、目的関数を4つ設定した。このように複数の目的(目標)を設定して同時に最適化することを**多目的最適化**と呼ぶ。この4つの目的それぞれに対して良いものや、いくつかの目的についてバランスをとったものなどを解として得ることができる。

実際に試した結果として、以下(図 6)のような解が出た。(最終結果では50個のパレート解が出るが、紙面の関係で特徴的な3つを抜粋)

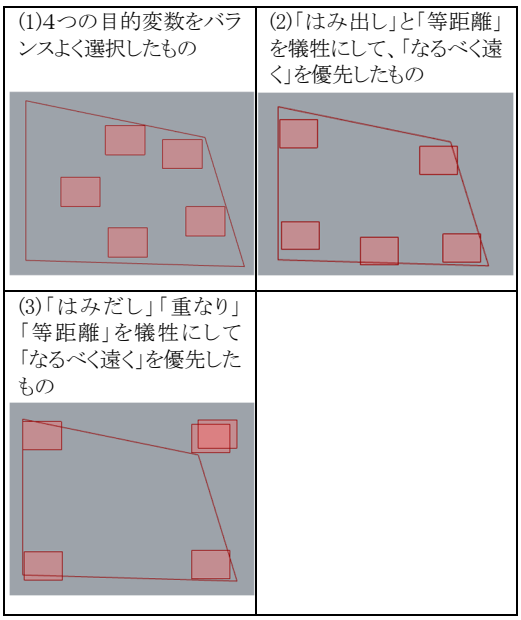

図 6 試行1の結果例

(1)については、期待される配置に近くなっているが、左側に余白が多く残っており、拡大の余地がある。(2)(3)については、目的関数に「はみ出さない」や「重ならない」を設定しているが、「多少はみ出していても遠くに配置」する案なども多数出ている。

なるべく遠くかつ同距離でという目的だけで座標を調整させ、(1)のような正五角形状に配置される解が出てきたことはまずまずの成果ではあるが、さらに最適化が必要と思われる。特に、はみだしや重なりを含む解は検討に値しないので、これらを省く方策を考える。

6. 試行2 (改善)

試行1の結果を踏まえ、以下の改善点を検討した。

(1) はみ出し有無を目的関数としたことで、はみ出しを許容した上でその他の条件が良い解が得られたが、検討対象にはならないので排除する。

(2) 重なり具合も同様である。

(3) はみ出しと重なりの指標は、「有無」の2値より、度合い化した方が良い。例えば大きくはみ出している場合と少しはみ出している場合で優劣をつけることで、遺伝的アルゴリズムの進化の効率が向上することが期待できる。

上記3点を考慮し、はみだし有無と重なり有無を目的関数からはずし、その変わりに計算した「はみ出し量」と「重なり量」を、それを目的関数の「ペナルティ」として調整することとした。

表 3 改善した目的関数

No	目的関数とその判断基準
①	長方形同士の距離の総和。より大きいものを良しとする。 ただし、ペナルティを減算することで、はみ出したり重なったりしているものは、そうでないものより悪いものとする。
②	長方形同士の距離の標準偏差。より小さいものを良しとする。 ただし、ペナルティを加算することで、はみ出したり重なったりしているものは、そうでないものより悪いものとする。

このような改良を加えて、最適化した結果が以下(図 7)である。(このケースも Population を 50，世代数を 100 とし、計算時間は数分であった)

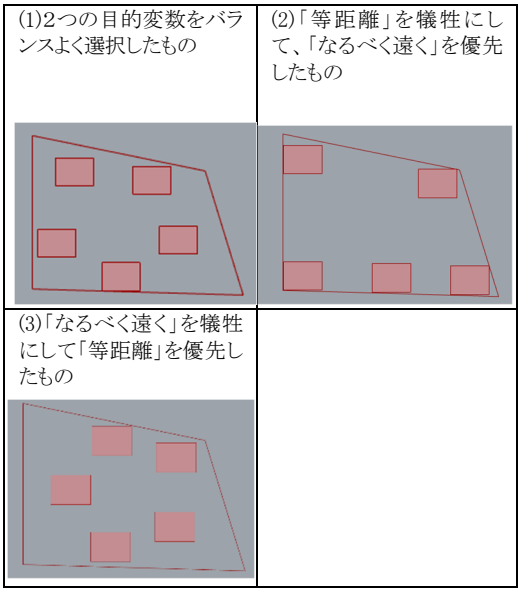

図 7　試行2の結果(抜粋)

試行2では、領域からはみ出したり、長方形が重なったりするような案は1つも出現しなかった。試行1と比べて、理想に近い案がでていると思われる。

なお、wallacei では、目的関数値をそれぞれ座標空間に表現したグラフ(図 8)を表示できる。

理想とする解は、このグラフでなるべく原点に近いところである。しかしながら、距離を遠くしようとすれば等距離性が失われ、等距離にしようとすれば距離の遠さを妥協しなければならないトレードオフの関係にある双曲線状になっていることがわかる。この曲線をパレートフロントと呼び、この曲線上の解をパレート解と呼ぶ。

図 7 で示す(1)～(3)の解は、それぞれ図 8 のグラフの(1)～(3)のプロットに該当し、(1)は原点に近い中庸状態のものである。

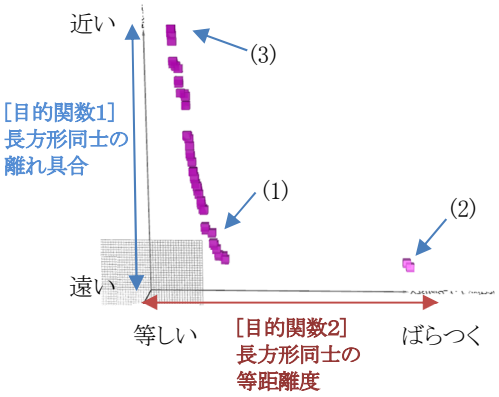

図 8　目的関数のプロット

7. 試行3 （別案）

これまでの試行から、正五角形の頂点辺りに配置すれば等間隔になりそうである。

正五角形状に配置することを前提に、五角形の大きさ(半径)と中心位置をどこに決めれば、お互い最も遠い距離に配置できるのかを最適化を使って求めてみる。

7. 1　パラメータ(説明変数)設計

配置を円周上に行うことを考えて、まず円の中心と半径を想定する。

長方形5つを 72 度ずつ回転した位置に配置すれば等間隔なるが、最初の1つをどの位置に配置するか、その角度もパラメータすることで、微妙な配置調整ができるようにする。

以上のことを整理すると、説明変数は中心位置(X)と(Y)、半径(R)、1つ目の角度(θ)の4つとなる。

7. 2　配置設計アルゴリズムの設計

パラメータで説明した通り、(X,Y)を中心とした半径 R の円周上に、長方形を5つ等間隔に配置する。

7．3　目的関数の設計

試行2と同じだが、正五角形上に配置するため、距離の標準偏差は実質意味を持たない。従って、目的関数から外した。つまり、目的関数が「距離の平均」の1つだけの「単一目的最適化」とした。

7．4　最適化の結果

このようにして最適化した結果が以下である。

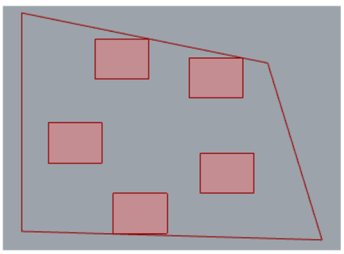

図 9　試行3の結果

計算時間は試行2と比べ格段に短くて済み、数秒でこの解を得ることができた。

領域の端ギリギリまで利用したレイアウトとなっている。正五角形に配置するというアルゴリズムであるため、並びは綺麗である。ここで注視したいポイントは、今回の結果が試行2の結果(1)とかなり近しいことである。説明変数と配置アルゴリズムは違えども、目的を満たす配置は同じだったということがわかる。

並びを五角形に決めてしまったことで、試行2-(1)のように、少し配置形状をゆがめても良いから、領域をなるべく広く活用したい、というような案は出てこない。

配置の自由度やバリエーションの数をとるか、配置を限定して短時間で解を見つけるか、それは設計者の目的や、要件次第ということになる。

8．まとめ

本稿では、遺伝的アルゴリズムを利用した配置最適化の簡単な実例を紹介した。人の思考になぞらえて説明変数や配置アルゴリズム、目的関数を設計してやれば、最適化アルゴリズムが比較的短時間でもっともらしい解をはじき出してくれることがわかる。

実際の設計は、もっと複雑であるから、より深い考察とアルゴリズム設計、最適化時間が必要である。その場合であっても、今回のような考え方を発展・応用させていくことで、最適解に近いデザインを得ることができると考える。

本稿により、ジェネレーティブデザインが読者の課題解決の手法の1つとなれば幸いである。

＜参考文献＞

1)　「Grasshopper コンポーネント Index」（Applicraft, https://www.applicraft.com/ghcp_index/）

2)　「Wallacei: Evolutionary Engine for Grasshopper3D」（Wallacei, https://www.wallacei.com/）