強制約条件付き最適化問題の２段階探索

余 俊

九州大学大学院
芸術工学府

李 宇豪

九州大学大学院
芸術工学府

高木 英行

九州大学大学院
芸術工学研究院

月着陸探査ミッションの最適化問題

- 設計変数と範囲

  経度∈ [0,1]　　緯度∈ [0,1]

- 制約条件

  連続日陰日数　＜ 0.05

  着陸地点傾斜角 ＜ 0.3

  氷の存在確率は考慮しない

- 単目的の場合は，通算通信時間の最大化問題
- 多目的の場合は，連続日陰日数の最小化，
  通算通信時間の最大化，傾斜角の最小化，の3目的

2段階探索

$$\text{制約度} = \frac{\text{制約条件を満たした制約数}}{\text{総制約数}}$$

- Step 1: 制約を満たす解のEC探索

  提案する制約度でEC探索，fitness計算なし.
  (explorationに相当).

- Step 2: Step 1の有効解周辺の局所探索

  提案する制約度とfitness値でEC探索.
  (exploitationに相当).

Fig. 1 強い制約条件付きの最適化問題の例．灰色部分は致死領域．

アルゴリズム自体の提案は
余俊，高木英行「強制約条件付き最適化問題の2段階探索」
進化計算シンポジウム 2017, 北海道茅部郡森町，pp.38-41 (2017年12月9-10日).

2段階探索

目的:

- Step 1: 個体の質(fitness)に関係なく制約条件を満たす解領域方向に進化を進め，できるだけ多くの制約充足解を見つけ出す．(制約度だけ)

- Step 2: より良い実行可能解に進化．(制約度+ fitness値)

（全制約が満たされ，かつ，子個体fitnessが親個体を上回る時のみ親個体が子個体に書き換えられる．）

Algorithm 1 任意の進化計算と組み合わせた本提案の 2 段階探索法.

```
1:  個体群の乱数初期化
2:  各個体の fitness と制約度に基づく評価
3:  while 終了条件が満たされ間 do
4:      for i = 1 to 個体数 do
5:          if 第 i 番目個体が制約をすべて満たしていない
            then
6:              （第 1 段階探索）制約充足解領域に向かうよ
                う fitness 代わりに制約度を用いる最適化.
7:          else
8:              （第 2 段階探索）fitness と制約度が高いより
                良い解を選び最適化.
9:          end if
10:     end for
11: end while
12: 終了
```

評価実験

- 30,000評価回数 × 21試行
- 乱数初期化
- ２変数の探査範囲は共に[0,1]

Table 1. ２段階探索 + 差分進化

	探索法	評価
Step 1	DE	制約度
Step 2	DE	制約度 + fitness値

Table 2. 差分進化パラメータの実験条件

個体数	40
交差率	0.7
scale factor F	0.3
DE 演算	DE/rand/1/bin

実験結果

21試行数	f1(=−通算通信日数)	c1(=連続日陰日数)	c2(=傾斜角)	d1(=経度)	d2(=緯度)
第1回試行	-0.9	0.016667	0.279794	0.831973	0.088969
第2回試行	-0.69	0.003333	0.168903	0.924894	0.064471
第3回試行	-0.9	0.016667	0.281835	0.832005	0.088964
第4回試行	-0.9	0.016667	0.276884	0.831998	0.088977
第5回試行	-0.9	0.016667	0.254431	0.832042	0.088995
第6回試行	-0.9	0.016667	0.260004	0.832033	0.088988
第7回試行	-0.803333	0.01	0.247826	0.823701	0.089547
第8回試行	-0.793333	0.006667	0.273661	0.841639	0.083697
第9回試行	-0.793333	0.006667	0.273661	0.841639	0.083697
第10回試行	-0.9	0.016667	0.268486	0.832065	0.088946
第11回試行	-0.9	0.016667	0.2719	0.831979	0.088997
第12回試行	-0.71	0.003333	0.264126	0.910415	0.004392
第13回試行	-0.9	0.016667	0.279946	0.831971	0.088967
第14回試行	-0.116667	0.006667	0.232276	0.665125	0.962102
第15回試行	-0.9	0.016667	0.269794	0.832038	0.088966
第16回試行	-0.793333	0.006667	0.273661	0.841639	0.083697
第17回試行	-0.9	0.016667	0.266101	0.832068	0.088949
第18回試行	-0.806667	0.01	0.279225	0.82363	0.089641
第19回試行	-0.9	0.016667	0.272617	0.832011	0.088977
第20回試行	-0.313333	0.007385	0.272813	0.701521	0.755923
第21回試行	-0.9	0.016667	0.271674	0.832044	0.088954

考察

- Step 1 では，fitnessを無視し，全制約を満たすまで解を進化.
- Step 2 では，制約充足解のfitnessを向上させるよう進化.
- （Step 1 ＋ Step 2 ＋既存ECアルゴリズム）は組合せ容易.
  汎用性ある解法.
- さらに改善余地あり.
  （多様性向上のための，2段階探索の反復実行など）

IDEにおける個体分散に基づく突然変異

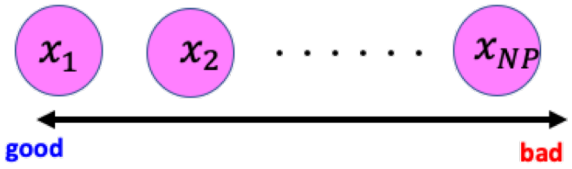
P2-02　龍谷大学
○古川雄大　小野景子　川畑忠宏

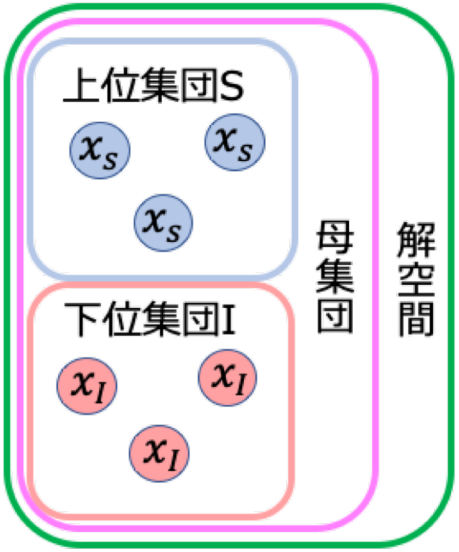
龍谷大学
RYUKOKU UNIVERSITY

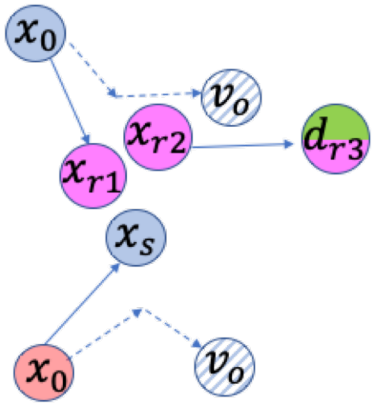
概要

IDE
Differential EvolutionWith an Individual-Dependent Mechanism

◆ Lixin Tangらが 提案した差分進化法の一種
◆個体の評価値に基づいてパラメーターを設定するIDP settingと探索方向、範囲を設定するIDM戦略により、効率の良い探索が可能



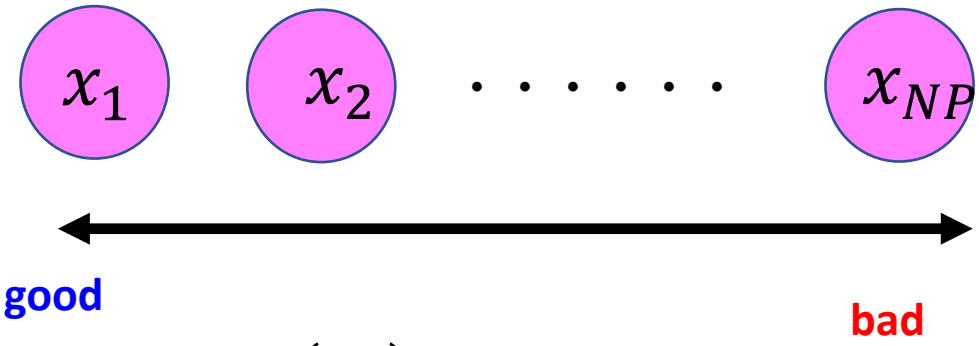




龍谷大学
RYUKOKU UNIVERSITY

Mutation:IDP Setting

個体の適応度が高い順にソート



Fの生成式:$\mathrm{F}_o = (randn\left(\frac{o}{NP}\right), 0.1) \quad (o = 1,2,\ldots,NP)$

- ◆ 適応度が高い個体はFが小さくなり探索範囲が縮小
- ◆ 適応度が低い個体はFが大きくなり探索範囲が拡大

CRの生成式:$\mathrm{CR}_i = (randn\left(\frac{\mathrm{i}}{NP}\right), 0.1) \quad (\mathrm{i} = 1,2,\ldots,NP)$

- ◆適応度が高い個体はCRが小さくなり親個体の情報を多く継承
- ◆適応度が低い個体はCRが大きくなり生成個体の情報を多く継承

龍谷大学
RYUKOKU UNIVERSITY

Mutation:IDM Strategy

解空間

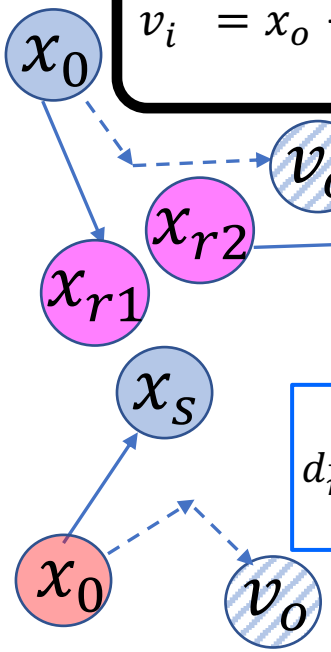
母集団 NP

上位集団S $\frac{ps}{NP}$個

x_s x_s x_s

下位集団I $\frac{1-ps}{NP}$個

x_I x_I x_I



$$v_i = x_o + \begin{cases} F_o * (x_{r1} - x_o) + F_o * (x_{r2} - d_{r3}) o \in S \\ F_o * (x_{better} - x_o) + F_o * (x_{r2} - d_{r3}) o \in I \end{cases}$$

S:上位集団
I:下位集団
x_{better}:上位集団からランダムに選択した個体

$$d_{r3}^j \begin{cases} L^j + rand_i^j(0,1) \cdot (U^j - L^j), \text{if}(rand_i^j(0,1) < P_d) \\ x_{r3}^j, otherwise \end{cases}$$

解空間から完全に新しい個体を生成
または、現在の個体からランダムに選択

ps:dr3の構成と上位集団と下位集団の個体の割合を決定

龍谷大学
RYUKOKU UNIVERSITY

Proposed Method

Point

psを個体の分散から求める

psを探索回数から求める場合、個体の多様性や収束の早さなどを考慮していない

提案手法は、個体の分散からpsの値を求める

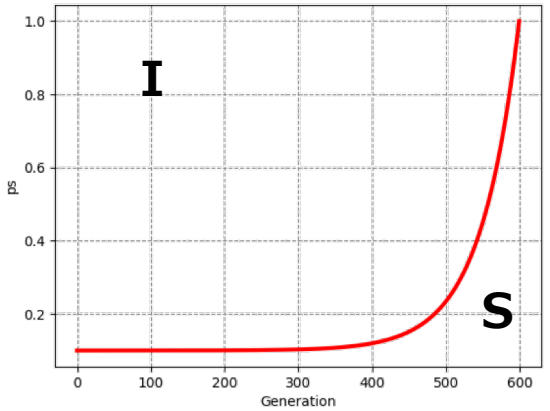
ps→各個体と解空間の重心とのユークリッド距離をとり、その分散を用いて求める

$$r(x_i,m) = \sqrt{(x_{i,1}-m_1)^2 + \cdots + (x_{i,D}-m_D)^2}$$

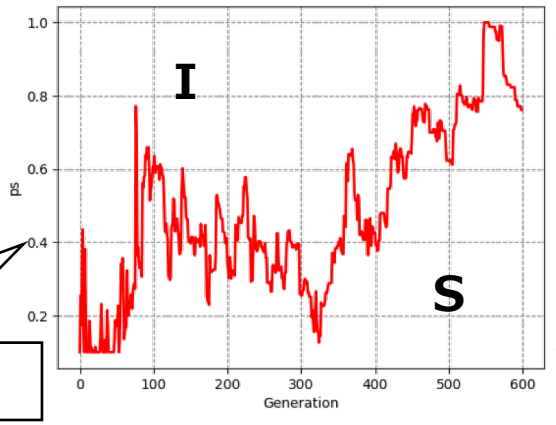
$$v(r) = \frac{1}{NP}\sum_{i=1}^{NP}(r-\bar{r})^2$$

$$ps = 0.1 + 0.9 \cdot (1 - \frac{v_n}{v_1})$$

探索回数からpsを求めた場合



個体の分散からpsを求めた場合



収束

拡大

探索状況に応じたpsの生成

Crossover

$FOR\ j = 1\ to\ D$

$$u_{i,g}^{j} = \begin{cases} v_{i,g}^{j} & \text{if}(rand_i^j(0,1) \le CR_i\ or\ j = j_{rand}) \\ x_{i,g}^{j}, otherwise \end{cases}$$

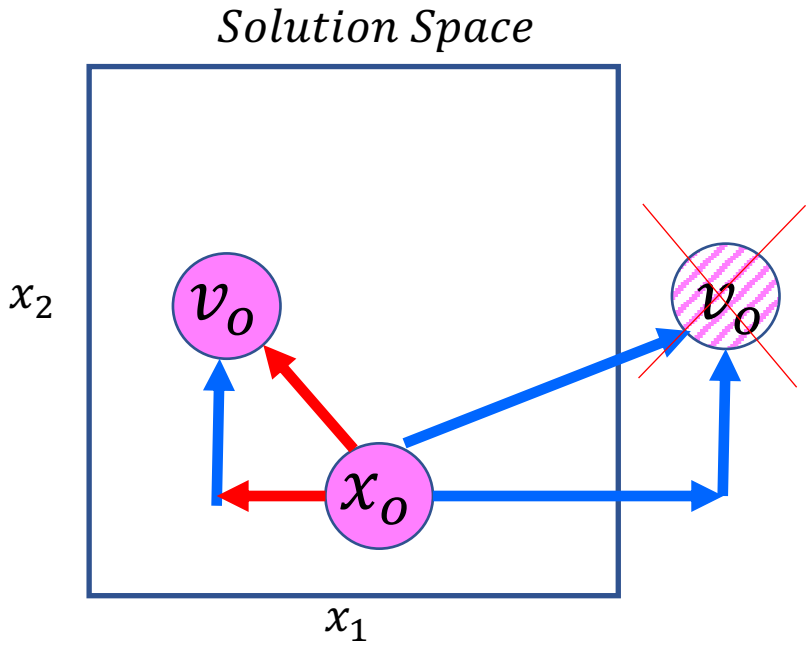
g世代i番目の個体のj次元目について交叉するかどうかを決定
(ランダムに交叉点を選択)

$IF(u_{i,g}^{j} < L \quad or\ u_{i,g}^{j} > U \quad)$

$u_{i,g}^{j} = L \quad + rand_i^j(0,1) \cdot (U \quad - L \quad)$

解空間からj次元目の値が外れた場合
解空間に収まるように値を再設定

→解空間に引き戻すのではなく
解空間の範囲からランダムに値を
決定



パラメータ設定

◆パラメータ
・個体数: 50

・世代数: 600

・初期集団:設計空間全体に乱数で生成

s03

月着陸最適候補地の選定問題における探索領域の限定による効率的な探索

室蘭工業大学大学院

開発　拓也，渡邉真也

Computational Intelligence Laboratory

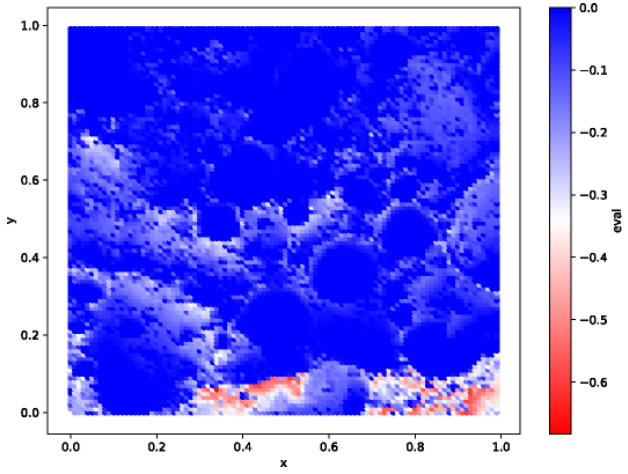
背景

月着陸ミッションの最適着陸地点の選定問題

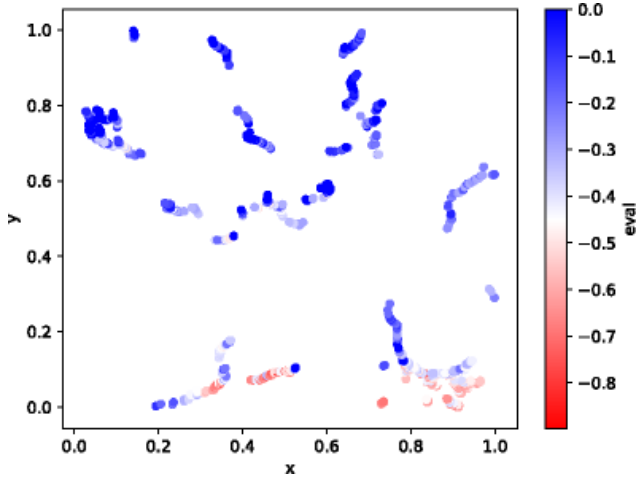
- 緯度，経度の2変数による最適化問題
- 探索領域のランドスケープが複雑
- 非常に厳しい制約条件
- 評価回数の少なさから効率性を求められる

制約充足解の発見の困難性

探索領域を絞り込み，ある限定された探索領域での最適化を行う事で効率よく良質な解を発見する

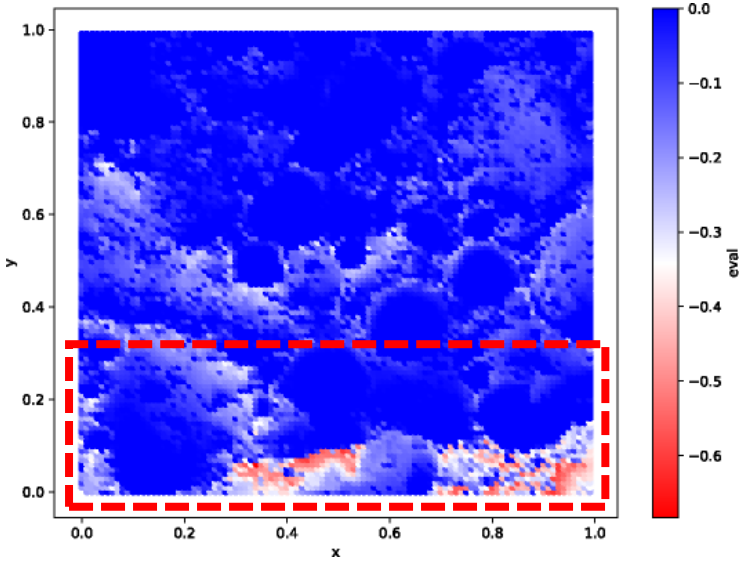


探索領域全体のランドスケープ(f_2)

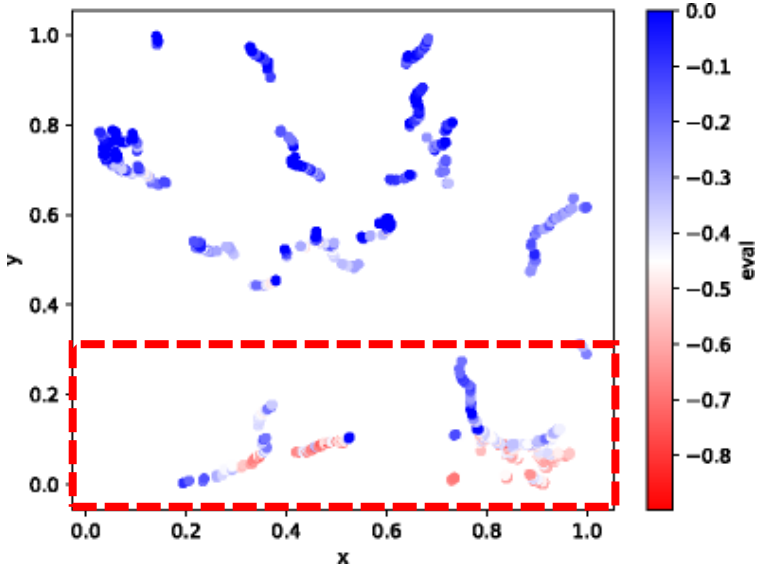


探索領域における制約充足解(f_2)

アプローチ



探索領域全体のランドスケープ(f_2)



探索領域における制約充足解(f_2)

探索における有望領域は探索空間のごく一部に限定

探索領域を限定し，有望領域に絞った探索

従来の交叉に基づくアプローチでは効率よく探索することが困難

探索領域全体を荒くサンプリング，その結果から探索範囲を限定し分布による最適化

提案手法：RSL-GS

Random Search in limited search area by Grid Search

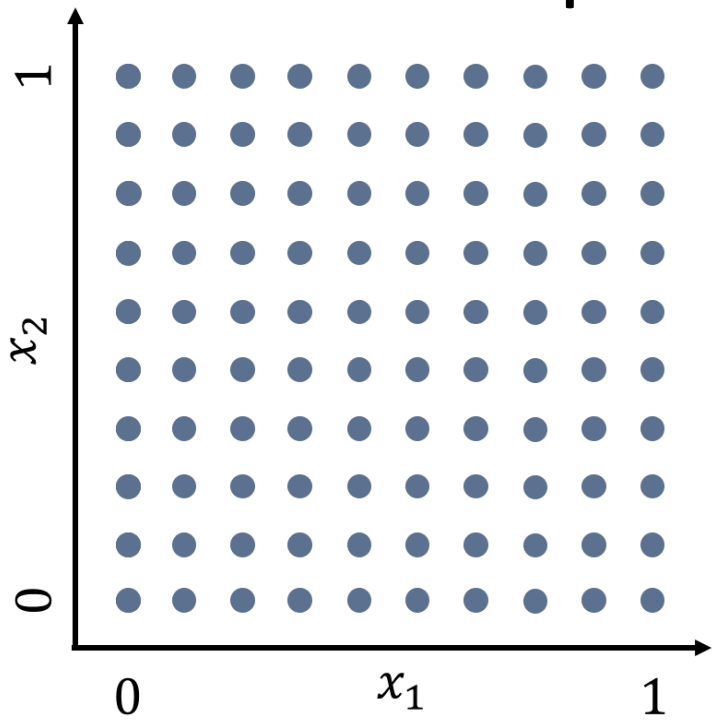
Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

Step1-1.探索領域全体に対してGrid Searchを適用

探索空間全体に対して
n 点の評価点を生成

制約条件を考慮せず
目的関数値のみ

depth = 1



提案手法：RSL-GS

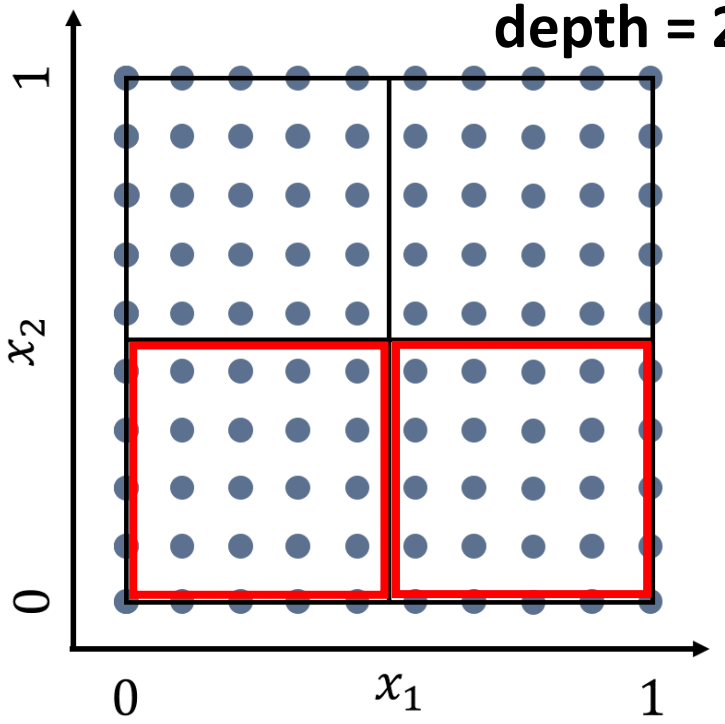
Random Search in limited search area by Grid Search

Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

Step1-2.探索領域の分割，限定

探索空間を4分割し，
各領域での最良解を比較

上位2つの領域を次の
探索領域とし，depth = +1する



提案手法：RSL-GS

Random Search in limited search area by Grid Search

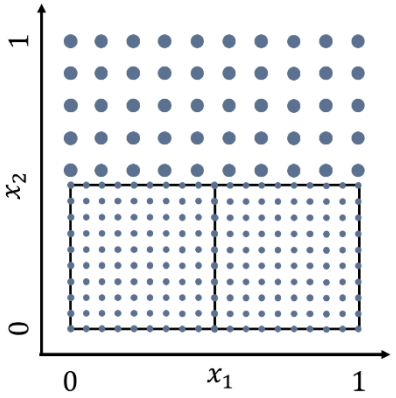
Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

Step1-3.終了条件

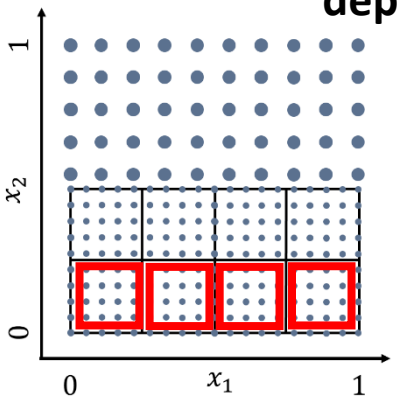
以下，終了条件を満たすまで
Step1.Step2.を繰り返す

本問題ではdepth=4を
終了条件に設定した

depth = 2



depth = 3



提案手法：RSL-GS

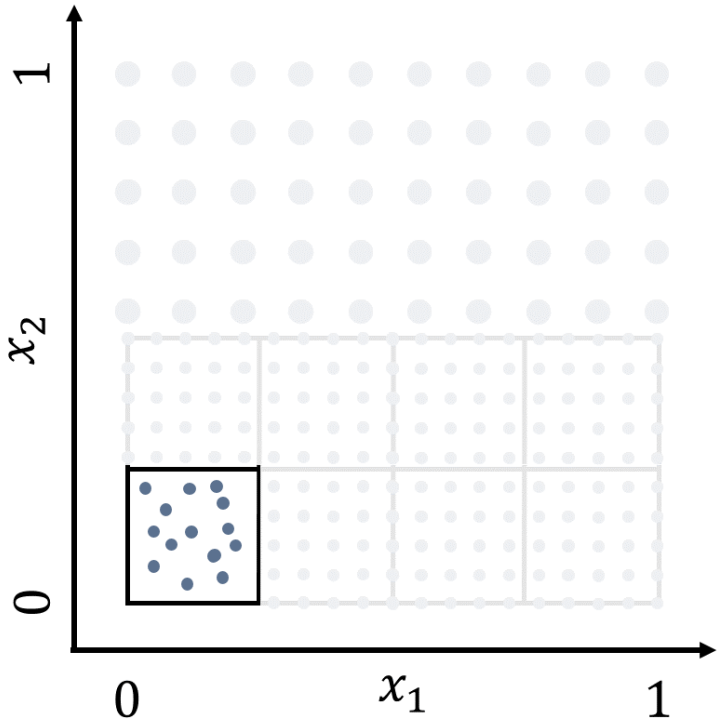
Random Search in limited search area by Grid Search

Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

Step2.限定された領域におけるRandom Search

限定された領域内において
ランダムに母集団を生成

本問題ではdepth=4を
終了条件に設定した



提案手法：RSL-GS

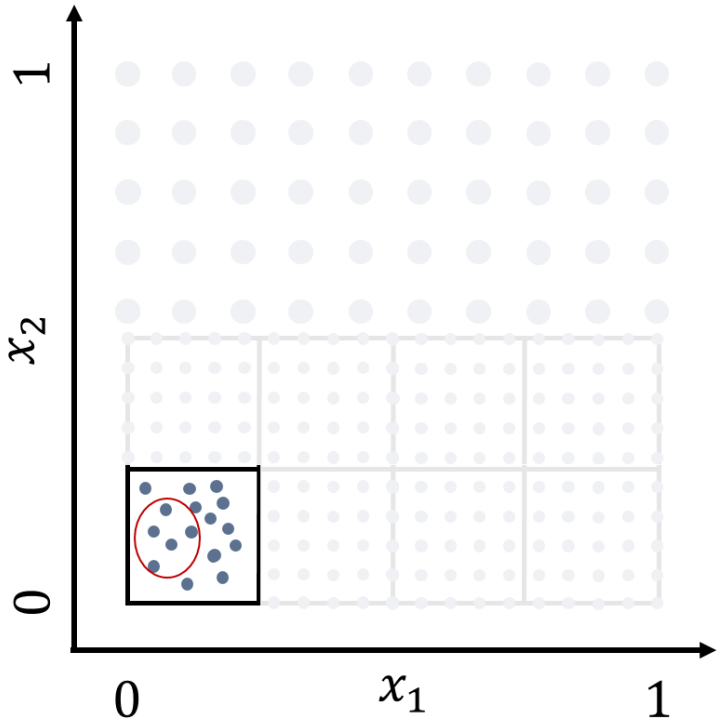
Random Search in limited search area by Grid Search

Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

Step2.限定された領域におけるRandom Search

母集団を評価し，
評価値の良い上位 R 個から
平均，標準偏差を計算

評価値は
・目的関数 f_2
・制約条件によるペナルティ
の合算を用いる



提案手法：RSL-GS

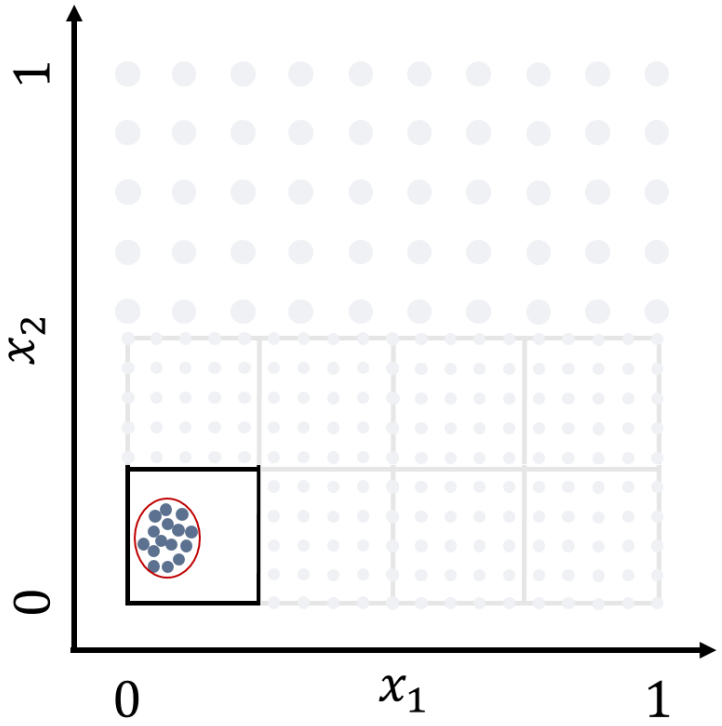
Random Search in limited search area by Grid Search

Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

Step2.限定された領域におけるRandom Search

平均，標準偏差を元に正規分布によって次世代集団を生成

生成した集団を評価，
平均，標準偏差を計算



提案手法：RSL-GS

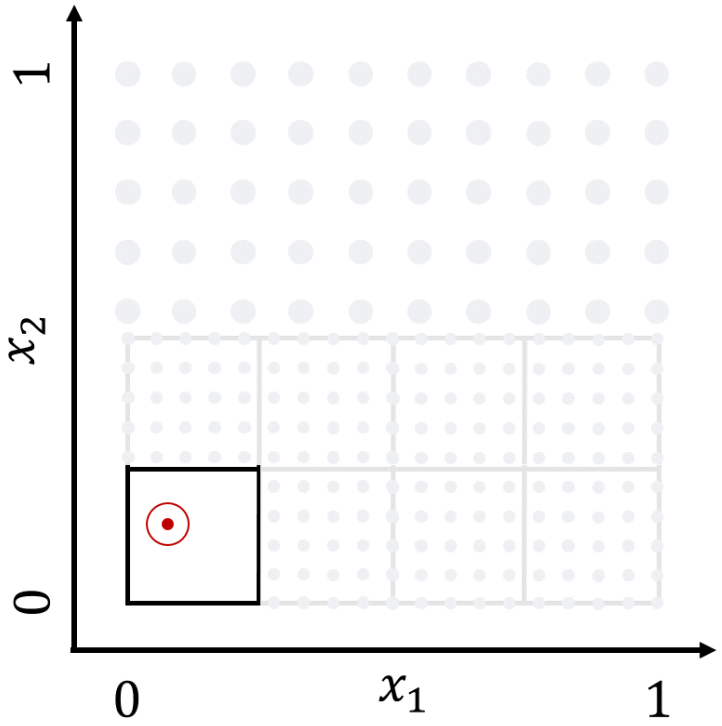
Random Search in limited search area by Grid Search

Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

Step2.限定された領域におけるRandom Search

次世代集団がベスト解を更新した場合，ベスト解の近傍を探索

平均をベスト解の設計変数
標準偏差を ϵ とする



提案手法：RSL-GS

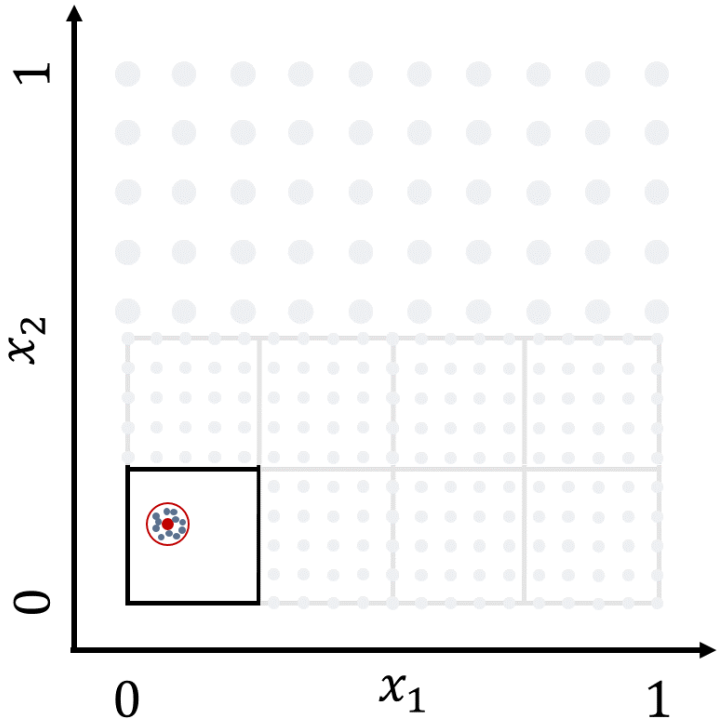
Random Search in limited search area by Grid Search

Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

Step2.限定された領域におけるRandom Search

ベスト解の近傍を探索することで制約を満たした解を複数生成する

平均をベスト解の設計変数
標準偏差を ϵ とする



提案手法：RSL-GS

Random Search in limited search area by Grid Search

Grid Searchによる探索領域の限定とRandam Searchによる最適化を組み合わせた最適化

チューニングパラメータ

Grid Search
- 分割深度 depth = 4
- グリッド点数 n = 100

Random Search
- 選択個体数 R = 20
- 近傍距離 ϵ = 0.001

アルゴリズム全体
- 母集団数 100

探索領域の限定
＋
大域的，局所的探索の併用
‖
質の良い解を発見する

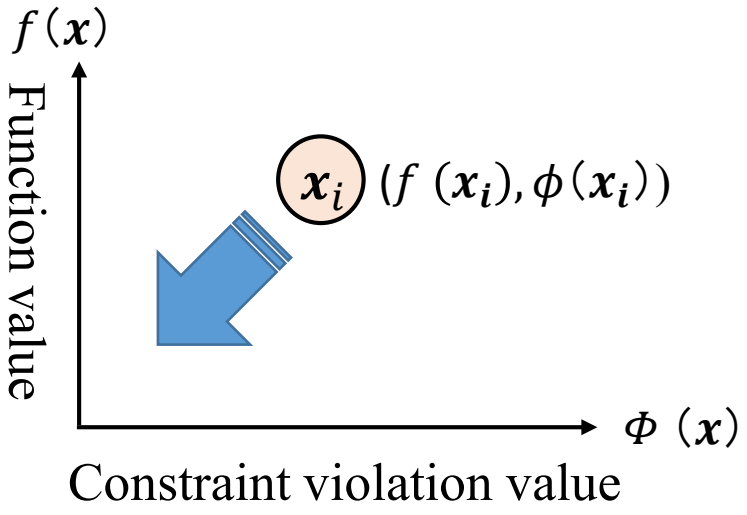
中央値：-0.87066
d_1：0.475812　d_2：0.091834

最適値：-0.97878
d_1：0.475979　d_2：0.091614

進化計算コンペティション2018
月着陸ミッションの最適着陸地点の選定問題
(単目的最適化部門)

串田淳一 （広島市立大学）

- **手法**：ε 制約Differential Evolution (εDE)
- εDEでは制約逸脱度$\Phi(\boldsymbol{x})$と目的関数$f(\boldsymbol{x})$を別々に最適化
- 制約逸脱度: 制約違反量の総和



Algorithm of εDE

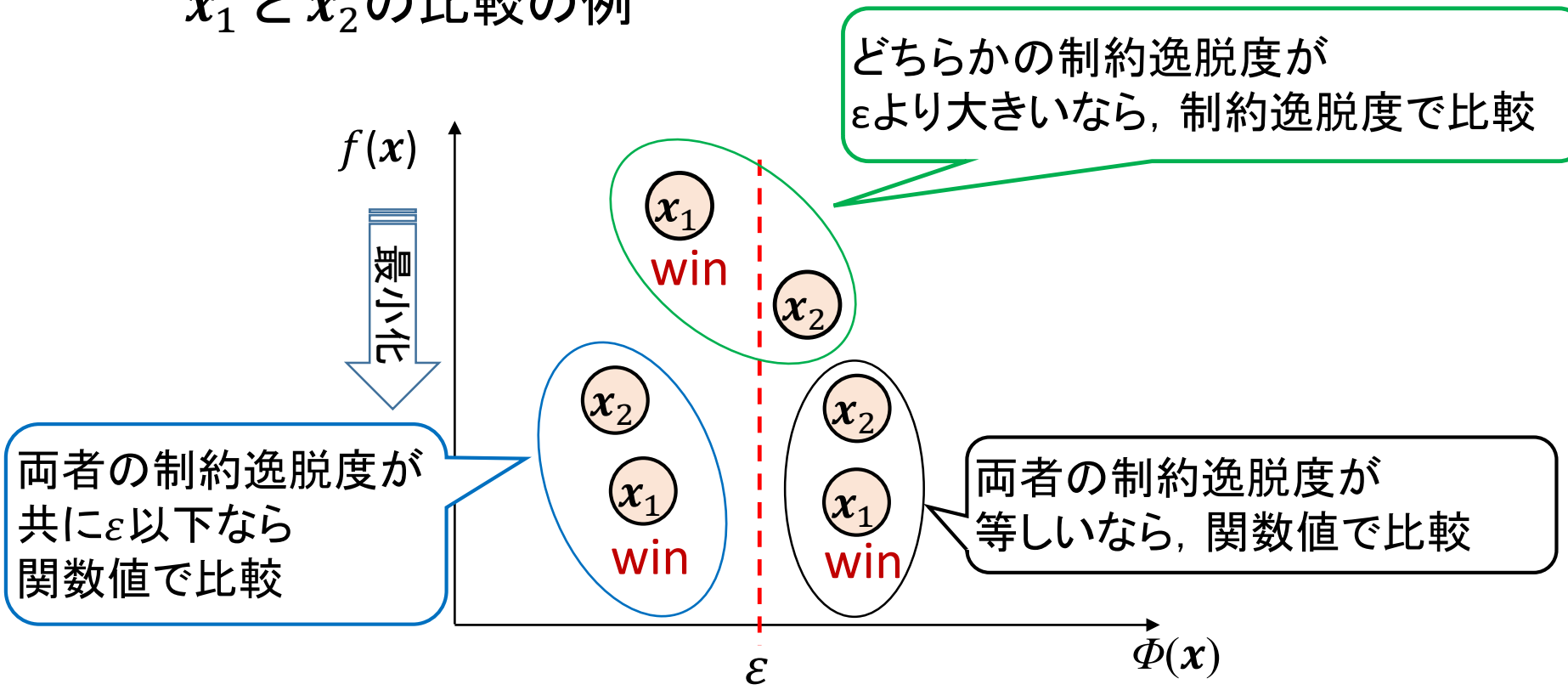
Generate NP individuals
Set initial ϵ value
for $t = 1$ **to** T_{max} **do**
- Generate trial vector
- Evaluation
- Selection by ϵ level comparison
- Control ϵ level

end for

ϵ レベル比較

制約逸脱度を目的関数より優先する辞書式比較
（制約条件を緩和して比較）

x_1 と x_2 の比較の例



ϵ レベルの制御

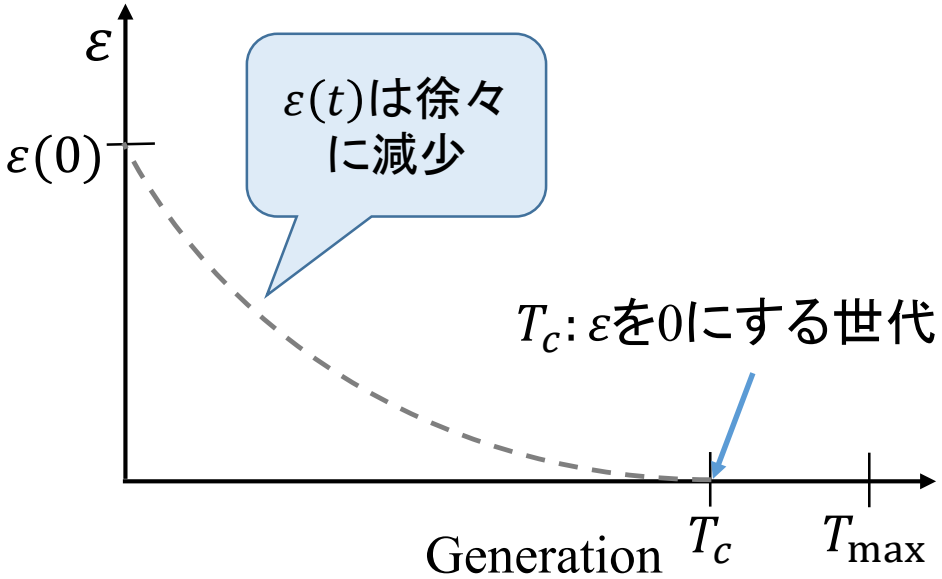
世代tにおけるεの値：$\varepsilon(t)$

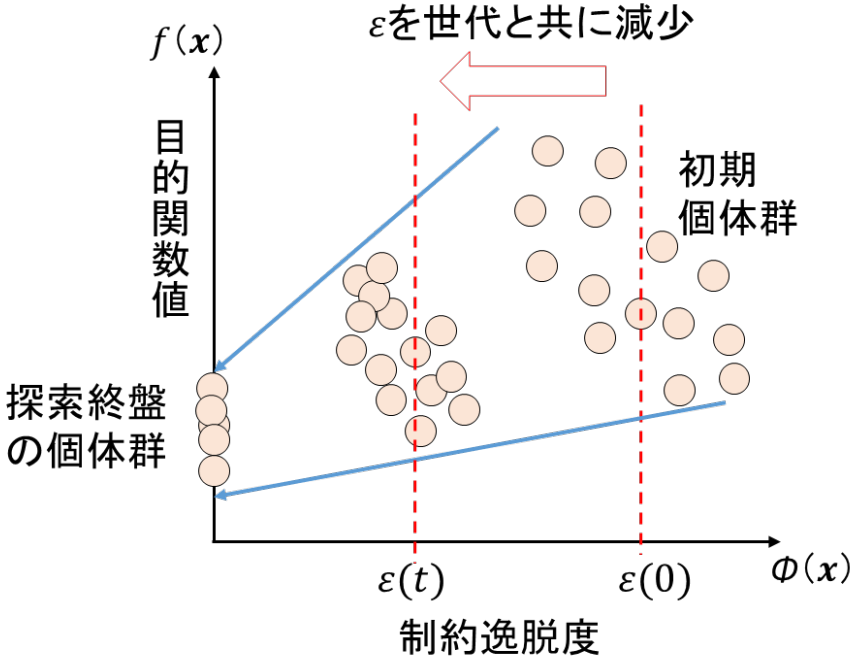
cp: ε制御のためのパラメータ

$$\epsilon(t) = \begin{cases} \epsilon(0)(1 - \frac{t}{T_c})^{cp}, & 0 < t < T_c \\ 0, & t \geq T_c \end{cases}$$

探索開始のεの値：$\varepsilon(0) = \Phi(\boldsymbol{x}_\theta)$

$\boldsymbol{x}_\theta$は，初期個体群を制約逸脱度で昇順にソートし
上位θ番目の個体 ($\theta = r \times NP$)





実験設定

設定したパラメータ

パラメータ	設定した値
個体数NP	100
最大世代数$T_{\max}$	300 (最大評価回数/NP)
εを0にする世代T_c	240 ($T_{\max}$*0.8)
$\boldsymbol{x}_\theta$決定のためのパラメータr	0.3
べき乗パラメータcp	2

- ε DEの戦略: ε DE/rand/1

  変異ベクトル$\boldsymbol{v} = \boldsymbol{x}_{r1} + F_t(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3})$

  世代tのFの値$F_t = 0.5 + 0.2 * \dfrac{\varepsilon(t)}{\varepsilon(0)}$

カスタマイズした点

0.7から0.5へと
世代の経過と共に減少

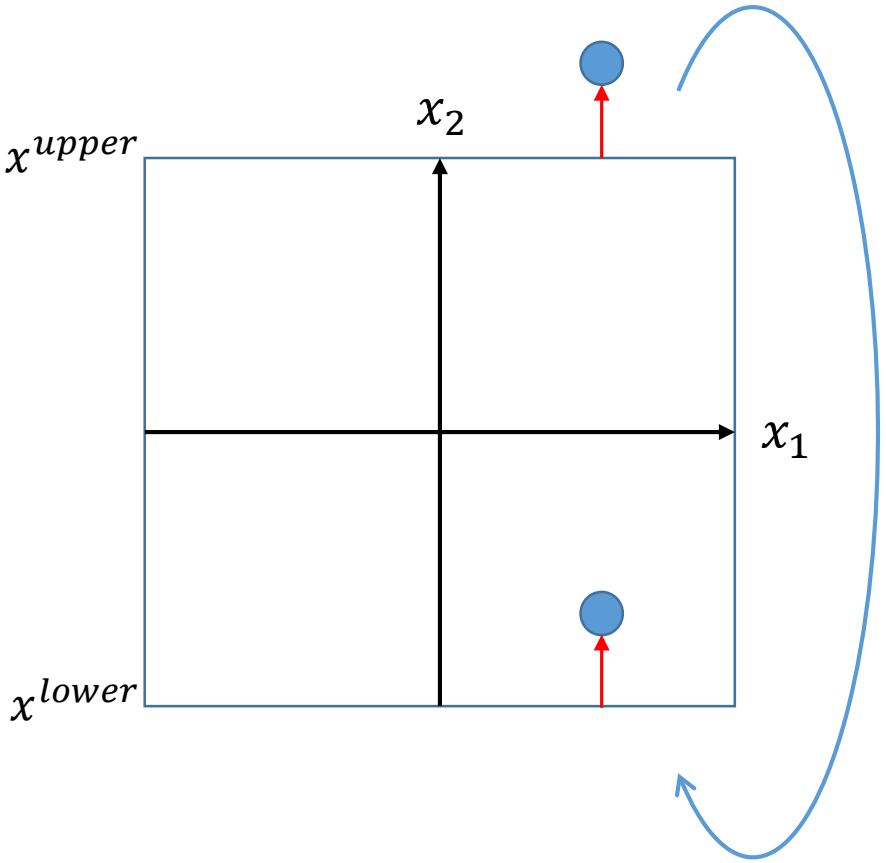
$F_{\max} = 0.7, F_{\min} = 0.5$

- 交叉率$CR = 1$
  - 変異ベクトル$\boldsymbol{v}$の全ての遺伝子を子$\boldsymbol{u}$へ (子個体$\boldsymbol{u} = \boldsymbol{v}$)
  - 回転不変性を持つ

上下限制約外に生成された解の修正

カスタマイズした点

探索空間をトーラス構造とみなす



$\boldsymbol{x} = (x_1, x_2)$

x_2が上限を超えた場合

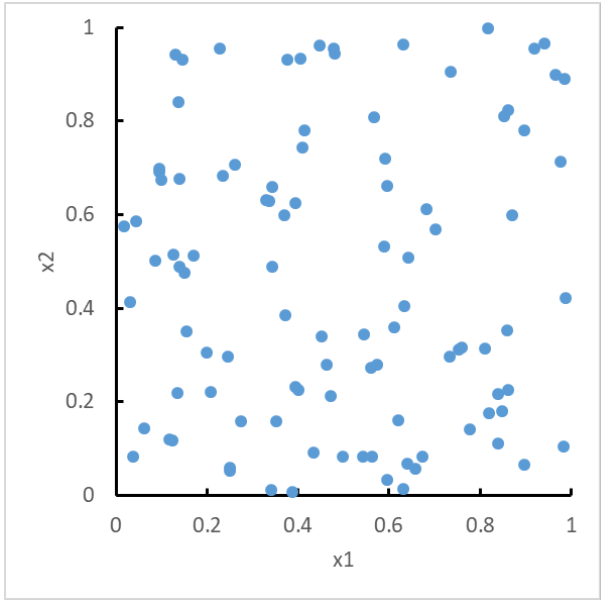
$(x_2 > x^{upper})$

$$x_2 = x^{lower} - (x_2 - x^{upper})$$

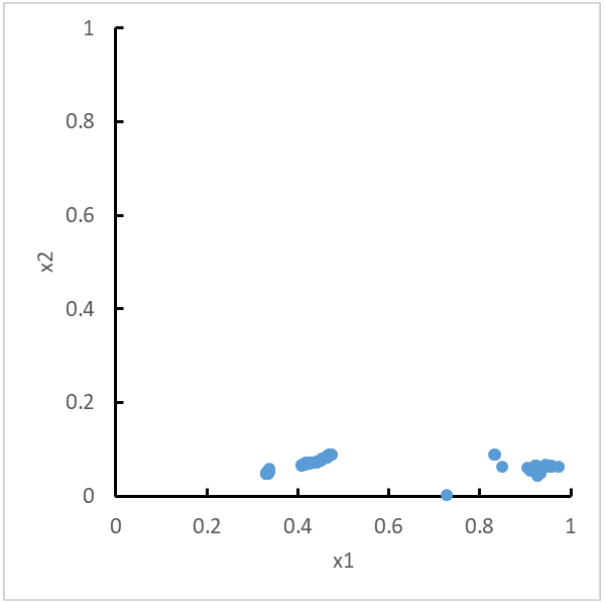
として，x_2を修正

x_1についても同様

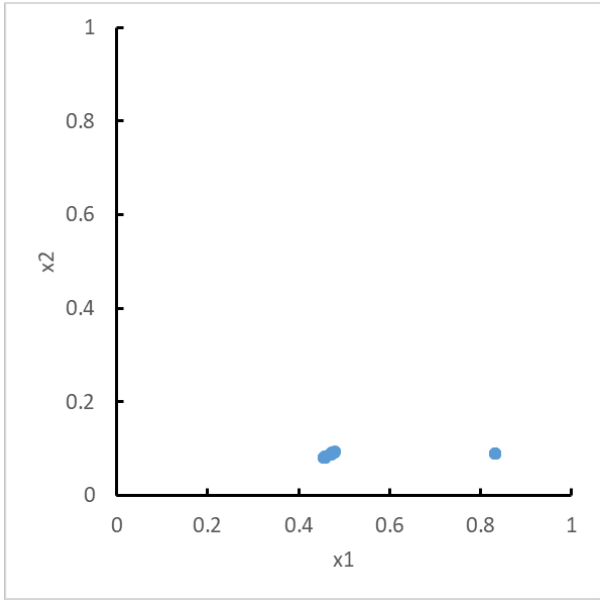
変数空間(x_1, x_2)における収束の様子



初期化時



探索中盤



探索終盤

進化計算コンペティション2018
アルゴリズム紹介

原田 智広 (立命館大学)

進化計算シンポジウム2018，2018年12月8-9日@福岡県福岡市休暇村志賀島

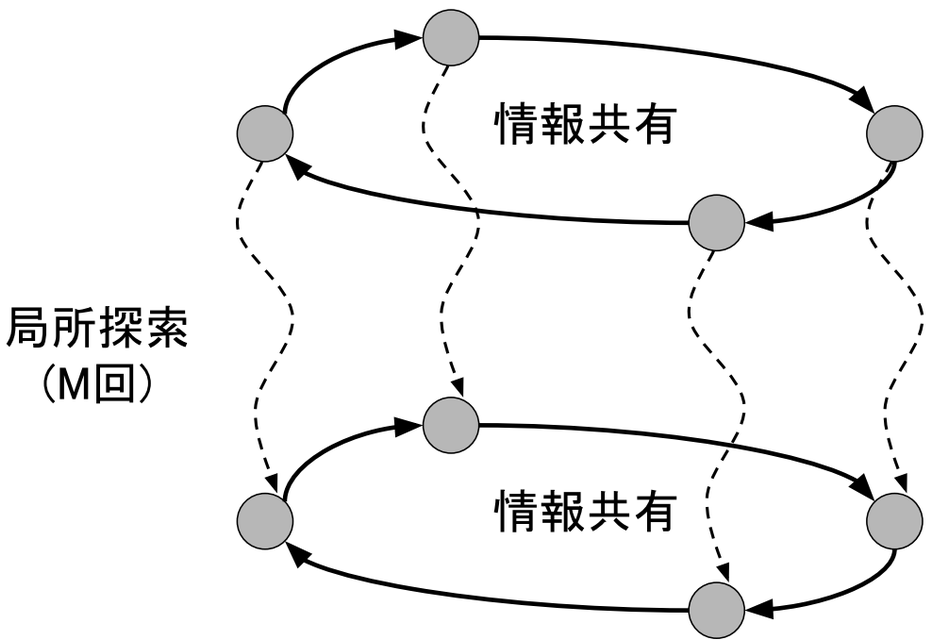
使用アルゴリズム

分散型局所探索法

- 局所探索法を複数の探索点から同時に開始
- 一定探索回数ごとに情報共有
- 局所探索法として以下の2つのハイブリッドを使用
  - Simulated Annealing (SA)
    - 温度パラメータTに従って遷移確率を決定

$$P(x, x') = \begin{cases} 1 & f(x) > f(x') \\ \exp\left(\frac{f(x)-f(x')}{T}\right) & otherwise \end{cases}$$

    - 温度パラメータを徐々に下げる



  - Variable Neighborhood Search (VNS)
    - 近傍関数を徐々に拡大しながら局所探索
    - 一定回数更新がなければ近傍関数を拡大

問題に特化した改良①

ペナルティ法を用いた適応度評価

- 実行不可能解であっても周囲に有望な解が存在する可能性
  →ペナルティ法により実行不可能解も探索候補点とする

$$f'(x) = f(x) + \alpha \left(\frac{\max(0, -g_1(x))}{0.05} + \frac{\max(0, -g_2(x))}{0.3} \right)$$

- $f(x)$：目的関数値 (通算通信時間)
- $g_1(x)$：連続日陰日数制約
- $g_2(x)$：傾斜角制約

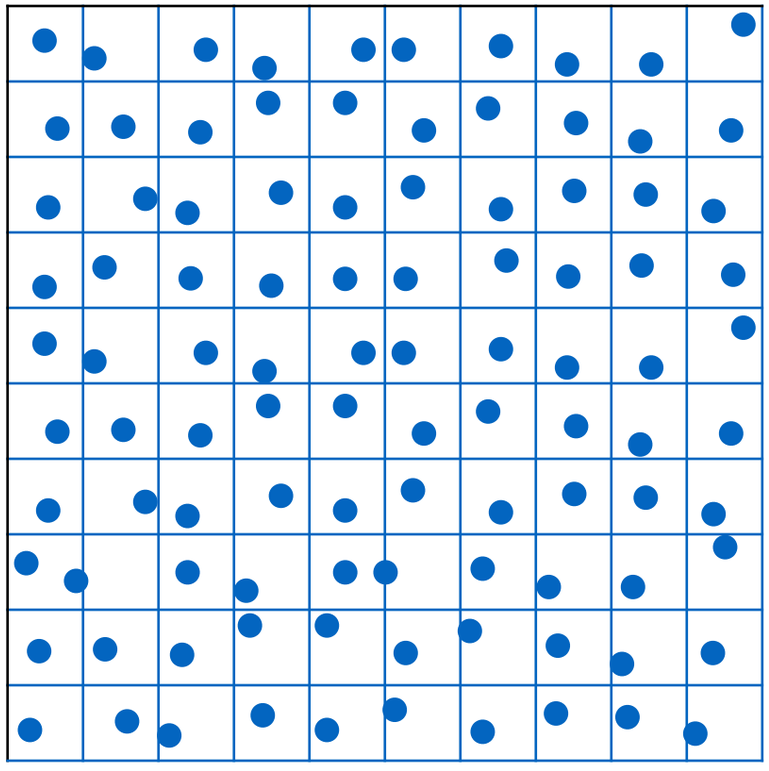
- 実行可能解同士，実行不可能解同士はペナルティ関数を用いて比較

問題に特化した改良②

<u>グリッドサーチによる初期探索点決定</u>

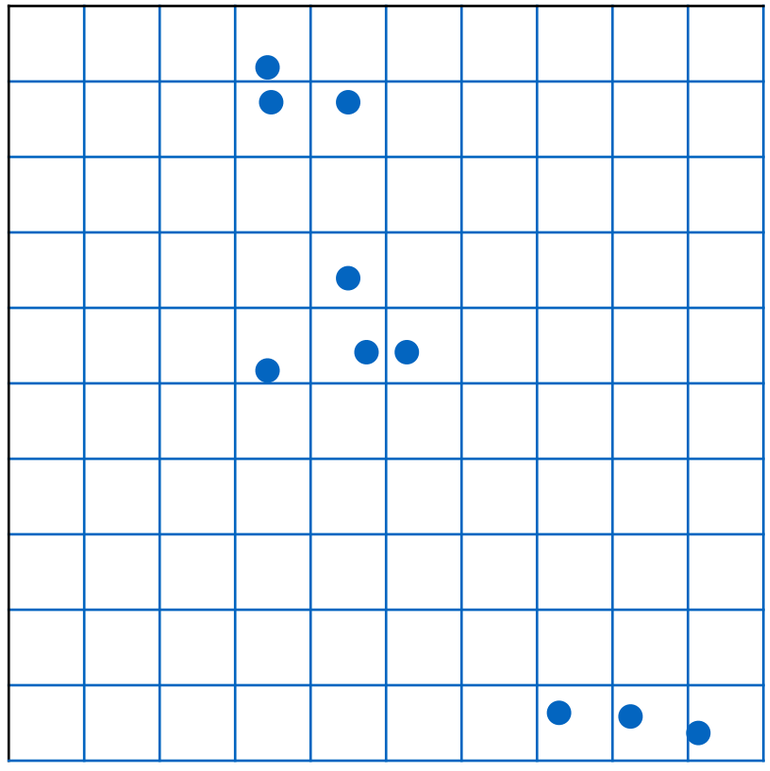
- 最適解候補点に偏りがある
  →開始点を適切に決定する必要
- 探索開始前にグリッドサーチを実行し，優良解を局所探索の開始点に設定

探索開始前



$f'(x)$に基づいて選択

初期探索開始点



全体アルゴリズム

2,500評価

グリッドサーチ

1.探索領域を50×50のグリッドに分割
2.各領域にランダムに個体を生成
3.上位100個体を分散SAの初期探索点に設定

17,500評価

分散SA

1.SAによる局所探索を100個体×5試行実行
2.探索点同士で情報共有し，探索点を更新
3.一定評価回数まで1. 2.を繰り返す
4.全探索点の上位100個体を分散VNSの初期探索点に設定

10,000評価

分散VNS

1.VNSによる局所探索を100個体×5試行実行
2.探索点同士で情報共有し，探索点を更新
3.一定評価回数まで1. 2.を繰り返す
4.最良解を出力

ハイパーパラメータ

個体数	100
グリッドサーチ幅	50
分散SA評価回数	17,500
分散VNS評価回数	10,000
ペナルティ重みα	0.1
情報共有間隔	5

近傍関数（SA）	$\mathcal{N}(x, 0.002)$
温度パラメータ	$T_i \leftarrow T_{i-1} \times 0.9$
初期温度T_0	0.1
温度更新頻度	1/5
近傍関数（VNS）	$\mathcal{N}(x, 0.001 + 0.002k)$
k更新頻度	1/5

詳細はポスターP3-01

自然進化戦略CR-FM-NESによる月着陸最適候補地の選定

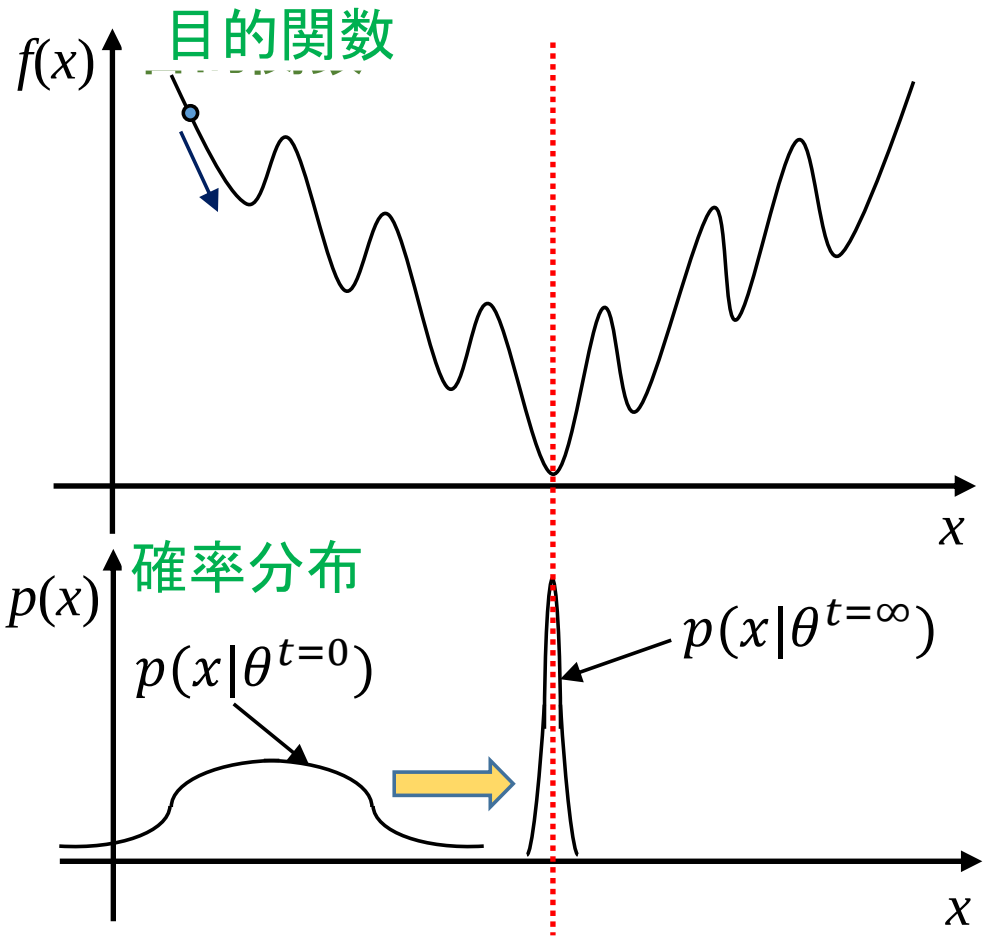
グループ s06 加藤 拓也, 小野 功
東京工業大学

はじめに

- 自然進化戦略 (Natural Evolution Strategies, NESs) [Wierstra 08]
  - ➢有力なブラックボックス関数最適化手法の1つ
  - ➢確率分布から生成される個体の期待評価値を最小化するように，確率分布のパラメータを逐次更新
- CR-FM-NES [野村 17, 加藤 18]
  - ➢Complexity-Reduction Fast-Moving NES
  - ➢NESsの中でも収束性能が優れる
    - ◆10,000次元規模の問題においても良好な性能
  - ➢進化計算シンポジウム2017でも良好な成績
    - ◆単目的部門4位，多目的部門1位（チェビシェフノルム法と併用）
- 月着陸最適候補地の選定問題への適用
  - ➢制約条件が厳しく，実行可能領域が非常に狭い → 初期収束しやすい
  - ➢探索途中で以下の条件を満たした場合に，確率分布の大きさを適当に拡大
    - ◆確率分布が実行可能領域をある程度覆った場合
    - ◆確率分布の大きさが小さくなってしまった場合

自然進化戦略(NESs) [Wierstra 08]

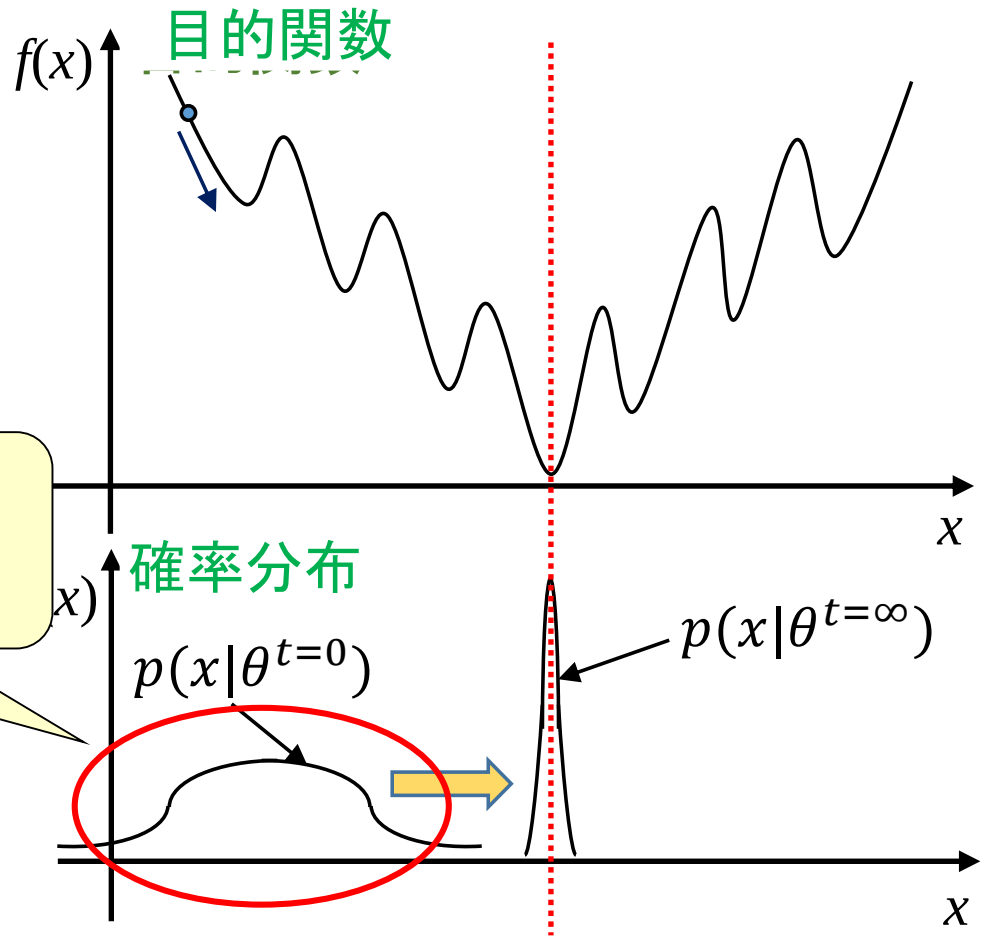
- 期待評価値 $J(\boldsymbol{\theta})$ の最小化
  - ➢$J(\boldsymbol{\theta}) = \int f(\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x}$.
    - ◆$f(\boldsymbol{x})$: 目的関数
    - ◆$p(\boldsymbol{x}|\boldsymbol{\theta})$:確率分布
      - 例) 正規分布
        - $\boldsymbol{\theta}=(\boldsymbol{m}, \boldsymbol{C})$: 平均と共分散行列
  - ➢$p(\boldsymbol{x}|\boldsymbol{\theta})$を適当に選べば$J(\boldsymbol{\theta})$ は $\boldsymbol{\theta}$ で微分可能
- 自然勾配法[Amari 85]:
  - ➢$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{F}^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.
    - ◆η: 学習率
    - ◆$\boldsymbol{F}(\boldsymbol{\theta})$:フィッシャー情報行列
      - $\boldsymbol{F}(\boldsymbol{\theta}) = E_{\boldsymbol{x}}[\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) (\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}))^{\mathrm{T}}]$



自然進化戦略(NESs) [Wierstra 08]

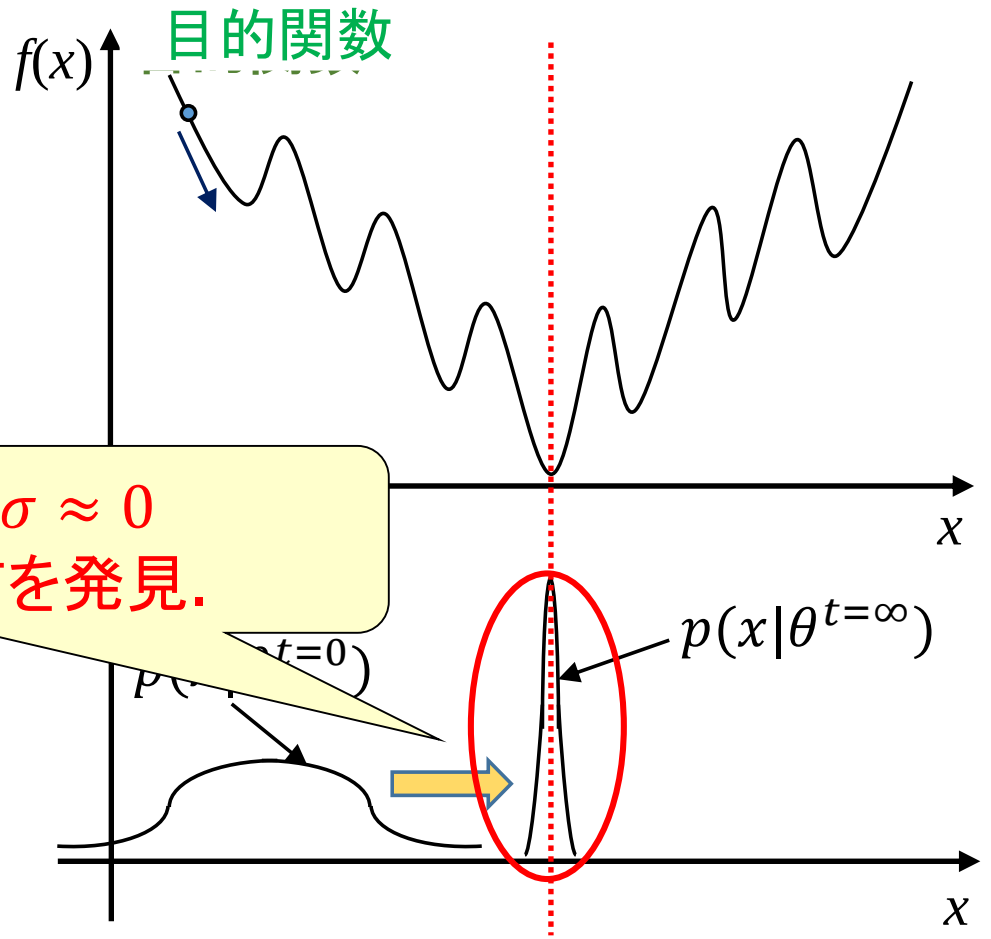
- 期待評価値 $J(\boldsymbol{\theta})$ の最小化
  - ➢$J(\boldsymbol{\theta}) = \int f(\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x}$.
    - ◆$f(\boldsymbol{x})$: 目的関数
    - ◆$p(\boldsymbol{x}|\boldsymbol{\theta})$:確率分布
      - 例) 正規分布
        - $\boldsymbol{\theta}=(\boldsymbol{m}, \boldsymbol{C})$: 平均と共分散行列
  - ➢$p(\boldsymbol{x}|\boldsymbol{\theta})$を適当に選べば
- 自然勾配法[Amari 85]:
  - ➢$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{F}^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.
    - ◆η: 学習率
    - ◆$\boldsymbol{F}(\boldsymbol{\theta})$:フィッシャー情報行列
      - $\boldsymbol{F}(\boldsymbol{\theta}) = E_{\boldsymbol{x}}[\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) (\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}))^{\mathrm{T}}]$

確率分布をランダムに
初期化.



自然進化戦略(NESs) [Wierstra 08]

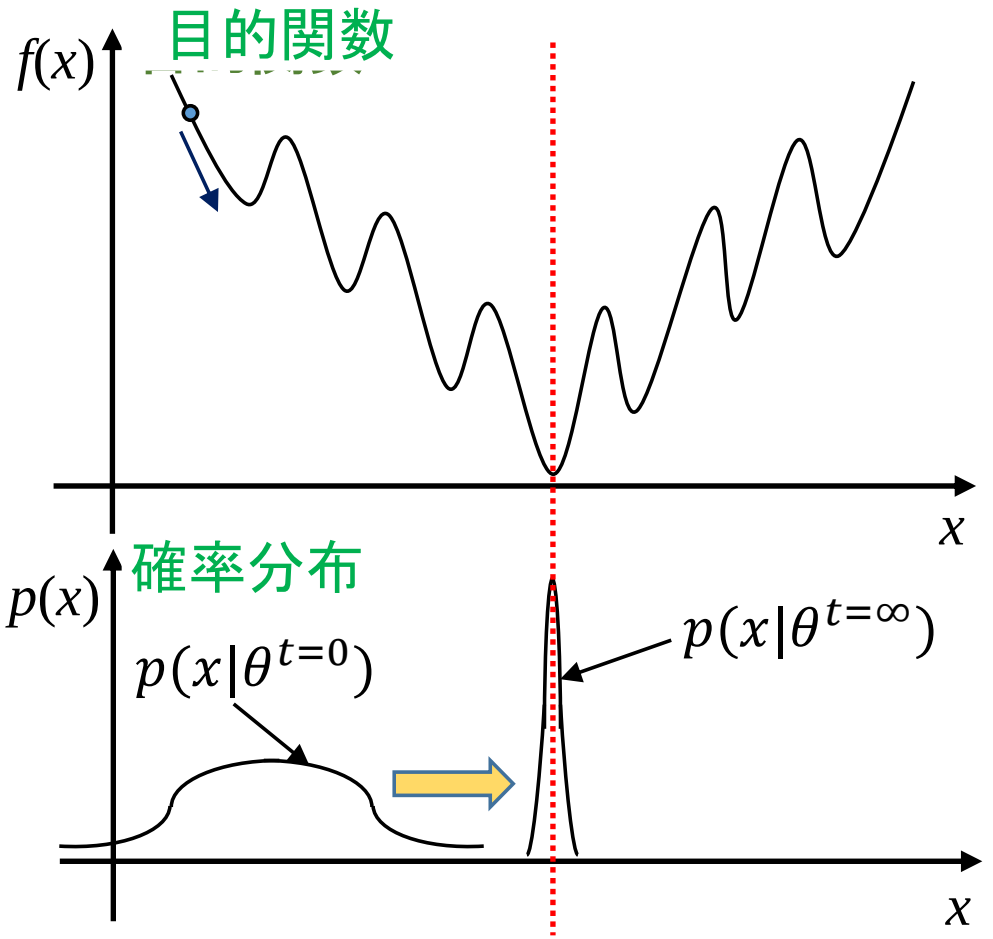
- 期待評価値 $J(\boldsymbol{\theta})$ の最小化
  - ➢$J(\boldsymbol{\theta}) = \int f(\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x}$.
    - ◆$f(\boldsymbol{x})$: 目的関数
    - ◆$p(\boldsymbol{x}|\boldsymbol{\theta})$:確率分布
      - 例) 正規分布
        - $\boldsymbol{\theta}=(\boldsymbol{m}, \boldsymbol{C})$: 平均と共分散行列
  - ➢$p(\boldsymbol{x}|\boldsymbol{\theta})$を適当に選べば$J(\boldsymbol{\theta})$
- 自然勾配法[Amari 85]:
  - ➢$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{F}^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.
    - ◆η: 学習率
    - ◆$\boldsymbol{F}(\boldsymbol{\theta})$:フィッシャー情報行列
      - $\boldsymbol{F}(\boldsymbol{\theta}) = E_{\boldsymbol{x}}[\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) (\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}))^{\mathrm{T}}]$



$m = x^{\mathrm{opt}}$, $\sigma \approx 0$
となる確率分布を発見.

自然進化戦略(NESs) [Wierstra 08]

- 期待評価値 $J(\boldsymbol{\theta})$ の最小化
  - ➢$J(\boldsymbol{\theta}) = \int f(\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x}$.
    - ◆$f(\boldsymbol{x})$: 目的関数
    - ◆$p(\boldsymbol{x}|\boldsymbol{\theta})$:確率分布
      - 例) 正規分布
        - $\boldsymbol{\theta}=(\boldsymbol{m}, \boldsymbol{C})$: 平均と共分散行列
  - ➢$p(\boldsymbol{x}|\boldsymbol{\theta})$を適当に選べば$J(\boldsymbol{\theta})$ は $\boldsymbol{\theta}$ で微分可能
- 自然勾配法[Amari 85]:
  - ➢$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{F}^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.
    - ◆η: 学習率
    - ◆$\boldsymbol{F}(\boldsymbol{\theta})$:フィッシャー情報行列
      - $\boldsymbol{F}(\boldsymbol{\theta}) = E_{\boldsymbol{x}}[\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) (\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}))^{\mathrm{T}}]$



CR-FM-NES [野村 17, 加藤 18]

- 探索のロバスト化に関する工夫
  - Fitness shaping [Wierstra 08]
  - 対称変量法
  - 非明示制約を考慮した個体の選好順序 [福島 13]
- 探索の高速化に関する工夫
  - 多変量正規分布の共分散行列表現を制限 [Akimoto 14]
    - $\mathbf{C} = \sigma^2 \mathbf{D}(\mathbf{I} + \mathbf{v}\mathbf{v}^{\mathrm{T}})\mathbf{D}$
  - 探索状況による学習率切換 [福島 13]
    - 移動期，収束期・・・学習率：大
    - 停滞期・・・学習率：小
    - 探索状況の判定・・・進化パス [Hansen 06]の大きさを利用
  - 移動期におけるDistance weightの利用 [福島 13]
  - 移動期におけるランク1更新 [野村 17]

最適着陸地点の選定問題への適用(1)

- ベンチマーク問題

  Decision variables:

  $x = (\text{経度},\ \text{緯度})^T \in \mathbb{R}^2.$

  Minimize:

  $f_1(x) = -\text{通算通信日数}.$

  Subject to:

  $c_1(x) = 0.05 - \text{連続日陰日数} \geq 0$

  $c_2(x) = 0.3 - \text{着陸地点傾斜角} \geq 0.$

- CR-FM-NESで最適化する問題

  Decision Variables:

  $x = (\text{経度},\ \text{緯度})^T \in \mathbb{R}^2.$

  Minimize:

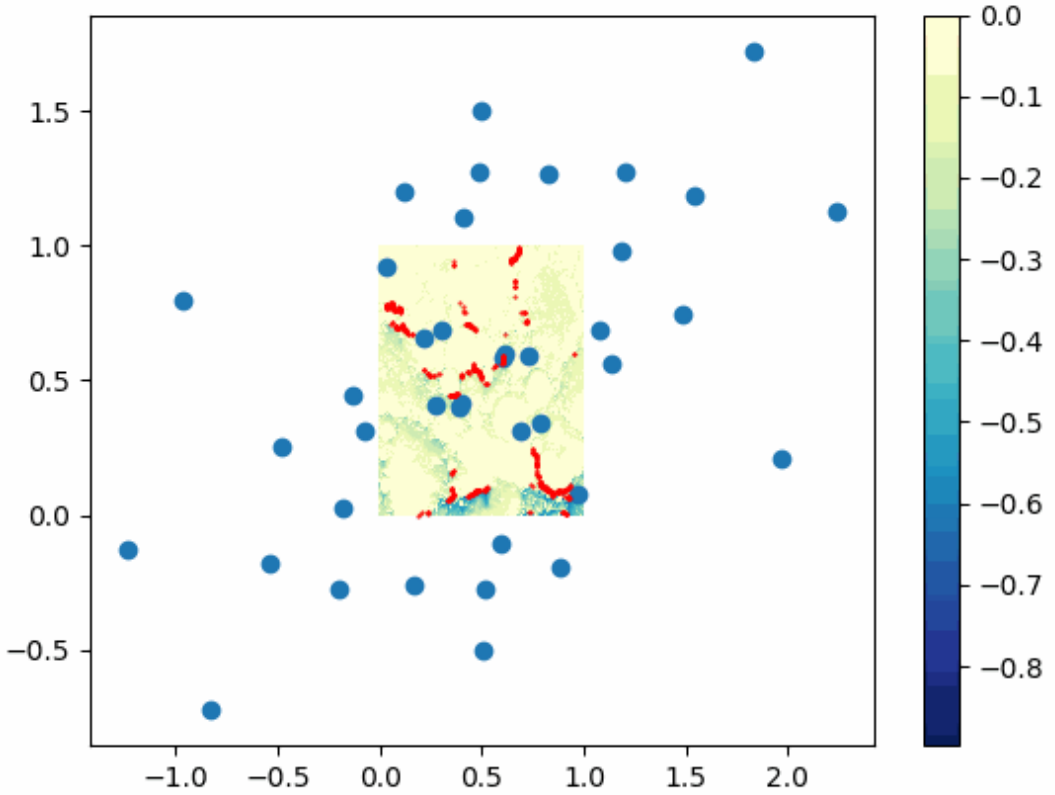
  $f(x) = f_1(x) +$

  $+k[\max(0, -c_1(x)) + \max(0, -c_2(x))]$

  k:ペナルティ係数

最適着陸地点の選定問題への適用(2)

- 初期化方法
  - ➢中心ベクトル: (0.5,0.5)
  - ➢ステップサイズ: 0.5
  - ➢個体数: 40
  - ➢定義域を外れたものは評価値を0にする



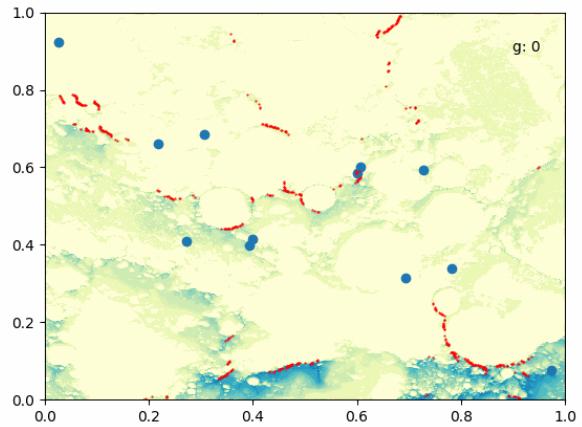
最適着陸地点の選定問題への適用(3)

- 確率分布が実行可能領域をある程度覆った場合に確率分布を拡大
  - 制約条件を満たす解の割合rが$r_{\mathrm{threshold}}(0 < r_{\mathrm{threshold}} < 1)$より大きくなったら，$\sigma$を$\sigma_{\mathrm{expand1}}$にして確率分布を広げる
- 確率分布が小さくなったら確率分布を拡大
  - σが$\sigma_{\mathrm{minimum}}$より小さくなったら，$\sigma$を$\sigma_{\mathrm{expand2}}$にして確率分布を広げる

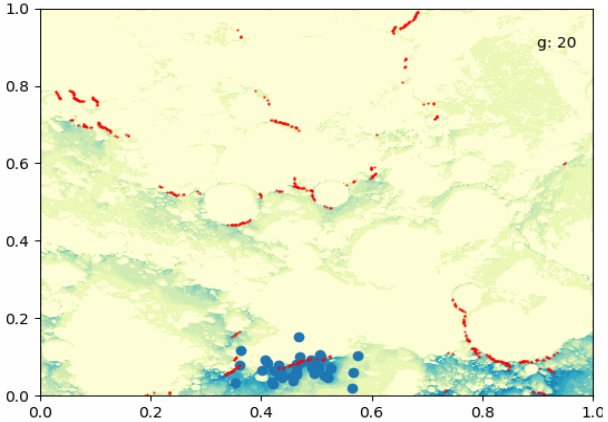
最適着陸地点の選定問題への適用(4)

- チューニングを行ったパラメータ
  - 目的関数のペナルティ係数
    - $k = 0$
  - 確率分布を広げる際のパラメータ
    - $r_{\mathrm{threshold}} = 0.6$
    - $\sigma_{\mathrm{expand1}} = 0.15$
    - $\sigma_{\mathrm{minimum}} = 0.00003$
    - $\sigma_{\mathrm{expand2}} = 0.01$

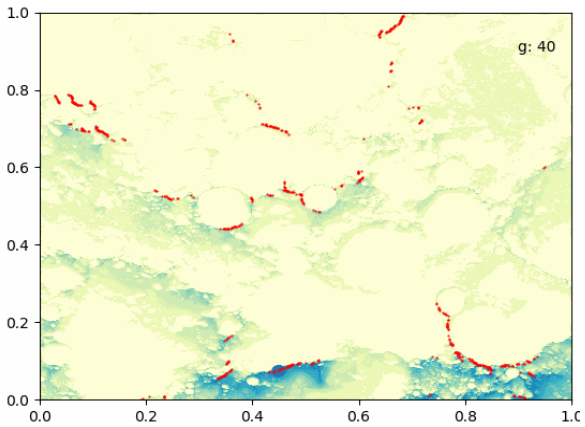
探索の様子



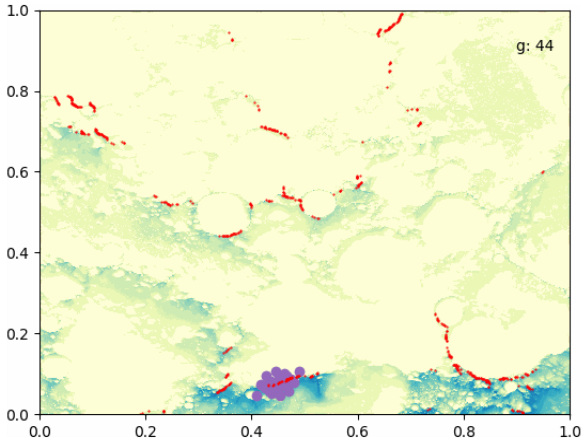
世代g=0



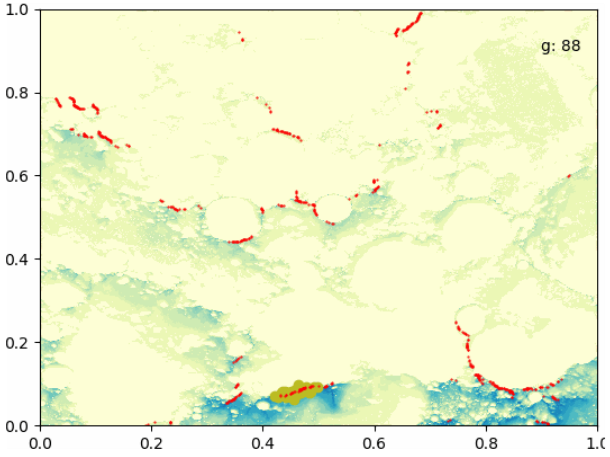
世代g=20



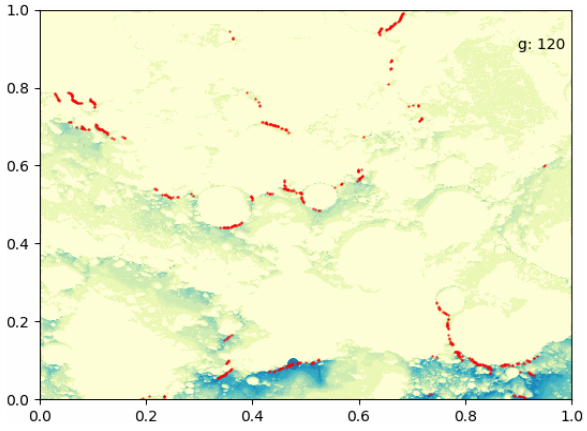
世代g=40



世代g=44

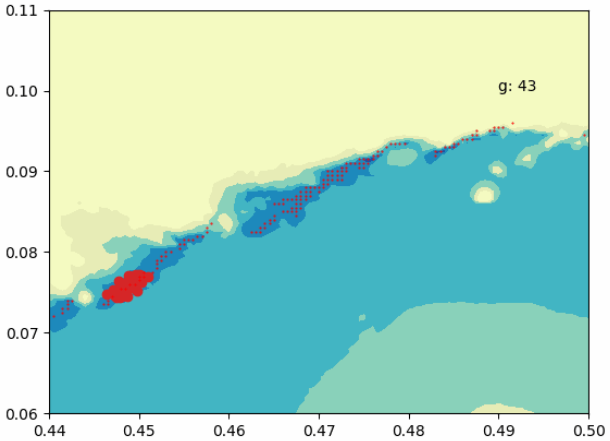


世代g=88

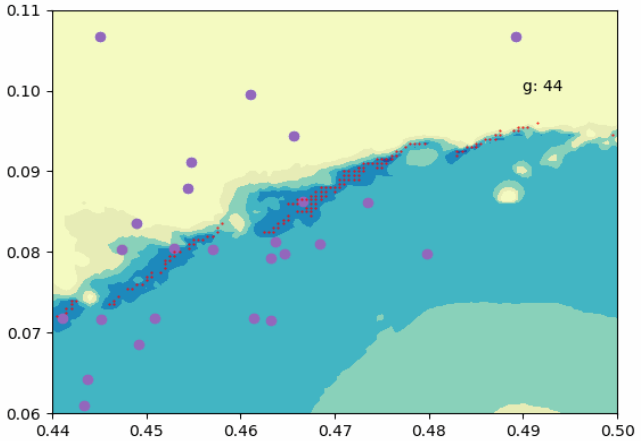


世代g=120

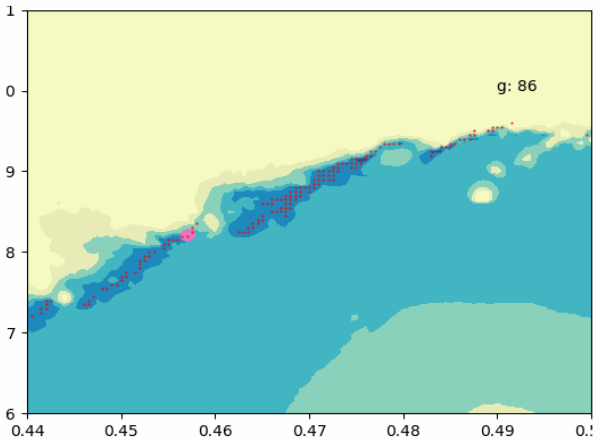
探索の様子



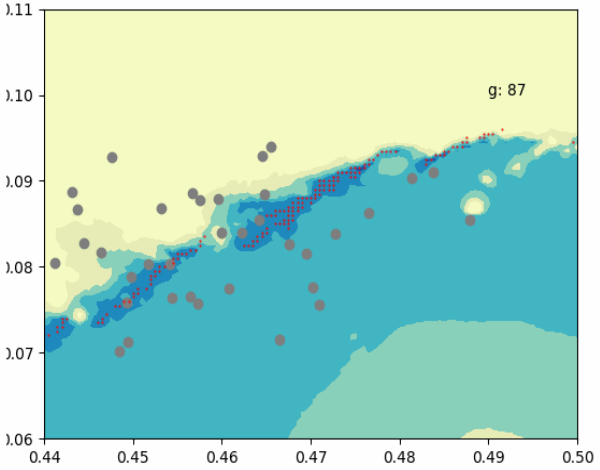
世代g=43



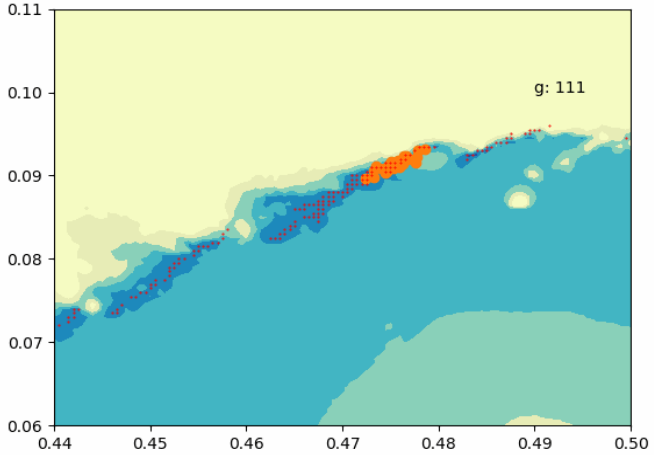
世代g=44



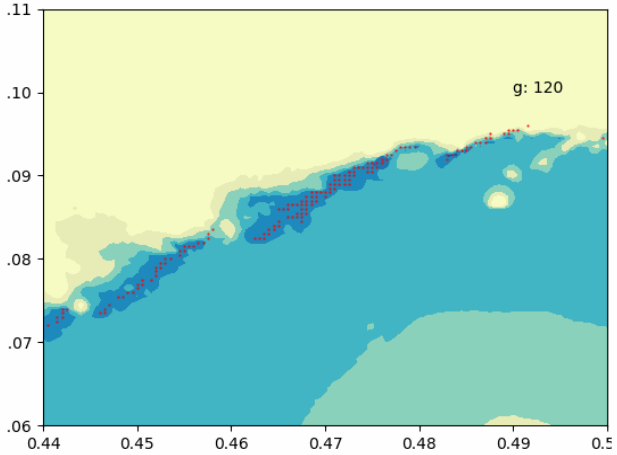
世代g=86



世代g=44



世代g=111



世代g=120

参考文献

- [Wierstra 08] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on, pp. 3381-3387. IEEE, 2008.
- [野村 17] 野村将寛, 金子研一郎, 小野功. ブラックボックス関数最適化のための自然進化戦略の高速化.第12回進化計算学会研究会，P1-4．2017.
- [加藤 18] 加藤遊馬，青木勇輔，小野功．自然進化戦略 CR-FM-NES の探索性能向上に関する研究第14回進化計算学会研究会，P3-8．2018.
- [Amari 85] Amari, S. (1985). Differential-geometrical methods in statistics. Lecture Notes in Statistics 28. New York: Springer-Verlag.
- [福島 13]福島信純, 永田裕一, 小野功. 非明示制約付きブラックボックス関数最適化のための自然進化戦略の提案. 第48 回システム工学部会研究会, 2013.
- [Akimoto 14] Youhei Akimoto, Anne Auger, and Nikolaus Hansen. Comparison-based natural gradient optimization in high dimension. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 373-380. ACM,2014.
- [Hansen 06] Nikolaus Hansen. The cma evolution strategy: a comparing review. Towards a new evolutionary computation, pp. 75-102, 2006.

Evolutionary Computation Competition 2018

December 03, 2018

Jair Pereira Junior, Yuri Lavinas, Claus Aranha

Department of Computer Science
Graduate School of Systems and Information Engineering
University of Tsukuba

1 Introduction

The Evolutionary Computation Competition 2018 address the problem of selecting the optimal landing site for the lunar landing exploration mission[1]. For the Single Objective Optimization, the variables longitude and latitude should maximize the total communication day while handling two constraints, the number of days of continuous shade and the inclination angle of the landing point. The Particle Swarm Optimization (PSO) method was chosen to approach this problem.

1.1 Moon Landing Problem

The goal for single objective optimization category is to maximize the total communication day[1]. Since the communication day is represented with a minus sign, the goal is actually to minimize it. There are two constraints, the number of days of continuous shade has to be lower than 0.05 and the inclination angle of the landing point has to be lower than 0.3. The input data is a set of latitude and longitude normalized between 0 and 1.

2 Method

A slightly modified version of the PSO that can be found in the github repository SwarmPackagePy[2] was used. The SwarmPackagePy original code would call for the external evaluation model[1] for each candidate solution, thus resulting in a long run-time. A modification was made in order to call the external evaluation model for a entire generation of candidate solutions at once. This modification does not affect the behavior of the original code.

The PSO starts with a set of random candidate solutions. In every generation, each solution is updated based on the global best and on the particle best[3]. This method takes 5 parameters[2]: number of agents, number of iterations, two acceleration coefficients and a weight for the velocity. The two acceleration coefficients are the balance between the importance of the global best and the particle best when updating a candidate solution[3].

2.1 Constraints Handling and Parameters settings

A penalty method was chosen to handle the constraints. To evolve the PSO, the following fitness function F was used:

$$F = -f + (c1+c2)w$$

where **f** is fitness computed by the external evaluation model, **c1** and **c2** are the constrains values, and **w** represents the importance of constraints in guiding the search. The value of **w** was arbitrarily set at 0.7.

In order to achieve 30,000 evaluations, the number of iterations and agents were determined by the following equation:

$$30000 = (\text{n\_iterations}+1)\text{n\_agents}$$

Lower values of agents resulted in early convergence, thus, in order to increase the diversity of candidate solutions, its value was set at 2000 and the number of iterations at 14. The two acceleration coefficients were both set at 1, and its weight at 0.5.

3 Result

Four experiments were performed, and each experiment was run 21 times. Two experiments was run with number of agents set at 600 and the number of iterations at 49, and two other experiments was run with number of agents set at 2000 and the number of iterations at 19. The parameters for the data submitted to the competition is in the subsection *Constraints Handling and Parameters settings*.

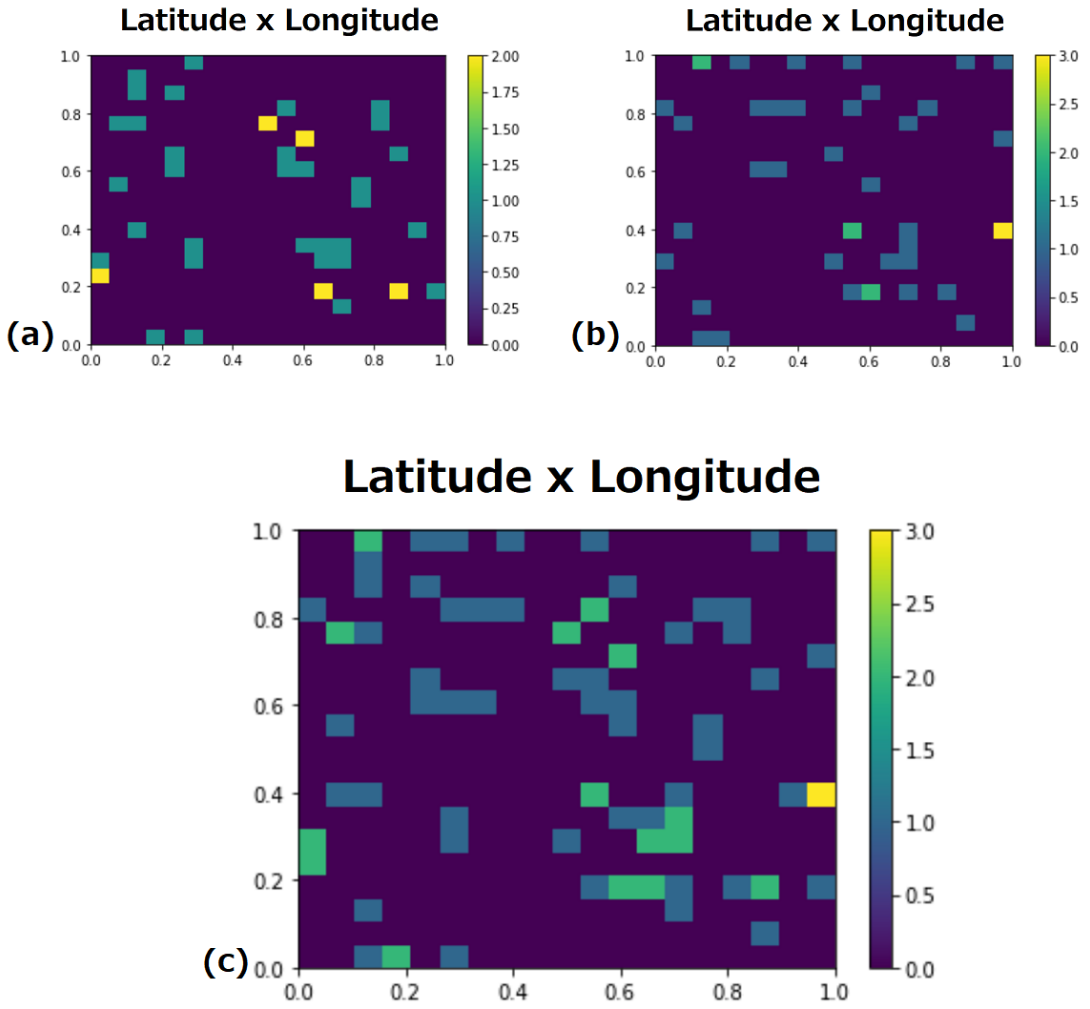


Figure 1: ***(a)*** *Heatmap for agents = 600, iterations = 49, and total results = 42;* ***(b)*** *Heatmap for agents = 2000, iterations = 14, and total results = 42;* ***(c)*** *Merged results from (a) and (b).*

	Median	**Best**	**Worst**	**Mean**	**Stdev**
Communication time	-0.60835892	-0.7323314	-0.13424031	-0.5462231319	0.1575111594

Table 1: *The best results found with the parameter settings at agents=2000 and iterations=19*

References

[1] Jpnsec.org. (2018). EC Symposium 2018 Competition. [online] Available at: http://www.jpnsec.org/files/competition2018/EC-Symposium-2018-Competition.html.

[2] SISDevelop, Library of swarm optimization algorithms, (2017), GitHub repository Available at: https://github.com/SISDevelop/SwarmPackagePy

[3] Engelbrecht, Andries P. (2002), Computational Intelligence: An Introduction, Wiley, Chichester, UK.

m05

Dynamic Niche Radiusに基づく個体間距離を考慮したBat Algorithm

○岩瀬拓哉　高野諒　上野史　高玉圭樹（電気通信大学）

Dynamic Niche Radiusに基づく個体間距離を考慮した Bat Algorithm

○岩瀬拓哉　高野諒　上野史　高玉圭樹（電気通信大学）

Bat Algorithm [Yang. X.S, 2010] …

コウモリの発するラウドネスAとその反射波rにより大域探索と局所探索を自動で切り替えることが可能

STEP1: 最良個体方向へ探索①

$$v_i^{t+1} = v_i^t + (x_*^t - x_i^t) * rand$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

STEP2: グローバルベスト近辺を局所探索②

if $rand > r_i$

$$x_{loc} = x_* + A_i^t * rand$$

endif

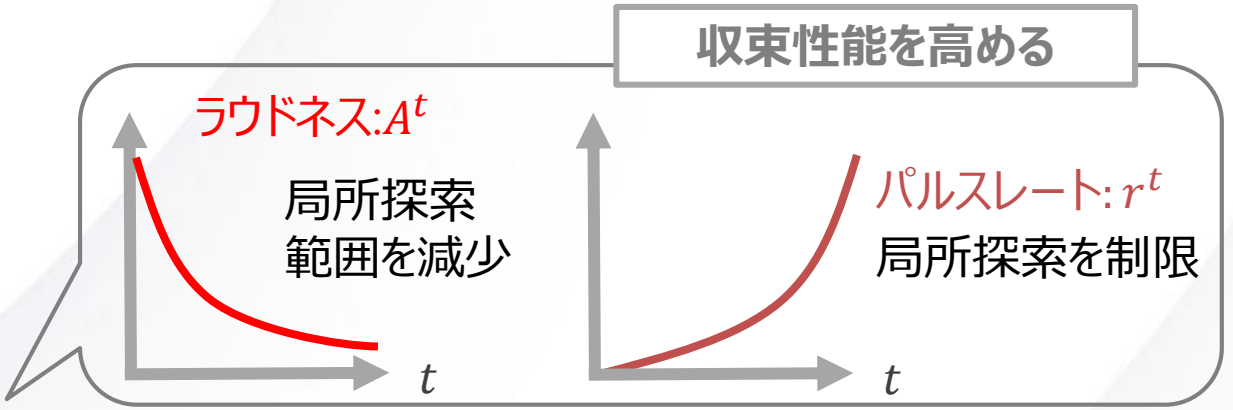
STEP3:ランダムによる大域探索③

$$x_{rnd} = x_{lb} + (x_{ub} - x_{lb}) * rand$$

STEP4: ①,②,③の解候補と現在の解の評価

x_{i*}を更新

$$A_i^{t+1} = \alpha A_i^t \qquad r_i^{t+1} = r_i^t[1 - exp(-\gamma t)]$$



Dynamic Niche Radiusに基づく個体間距離を考慮した Bat Algorithm

○岩瀬拓哉　高野諒　上野史　高玉圭樹（電気通信大学）

Dynamic Niche Radius [Miller,1996]

$$\lambda = \frac{1}{2}\sqrt{(x_{ub} - x_{lb})^2}$$

探索範囲の上限と下限: x_{ub}, x_{lb}

$$\text{Niche Radius} : \sigma = \frac{\lambda}{\sqrt[D]{N}}$$

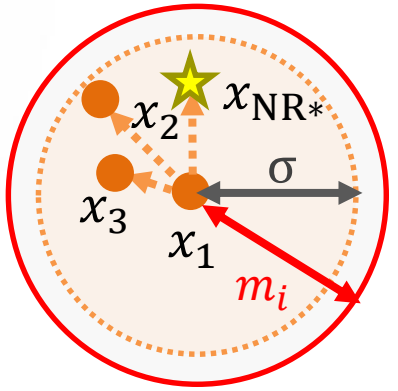
次元数: D　個体数: N

$$\text{Sharing function} : sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma}) & (\text{if } d_{ij} < \sigma) \\ 0 & (\text{otherwise}) \end{cases}$$

$$\text{Niche count} : m_i = \sum_{j=1}^{N} sh(d_{ij})$$

値が大きいほど周辺に個体が密集

$$\text{Dynamic Niche Radius} : m_i^{dyn} = \begin{cases} \sigma & (\text{if } m_i < \sigma) \\ m_i & (\text{otherwise}) \end{cases}$$



★ : $x_{\mathrm{NR}*}$

● : x_i $(i = 1,2,\ldots,N)$

Dynamic Niche Radiusに基づく個体間距離を考慮した Bat Algorithm

○岩瀬拓哉　高野諒　上野史　高玉圭樹（電気通信大学）

Dynamic Niche Radius-based Bat Algorithm (DNRBA)

STEP1: Dynamic Niche Radiusの算出

STEP2: 最良個体から離れる方向へ探索①

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + (\boldsymbol{x}_i^t - \boldsymbol{x}_{NR*}) * rand$$

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} & (\text{if } d_i < m_i^{dyn}) \\ \boldsymbol{x}_i^t & (\text{otherwise}) \end{cases}$$

d_i: 個体間距離

STEP3: Niche Radius内で局所探索②

if $rand > r_i$

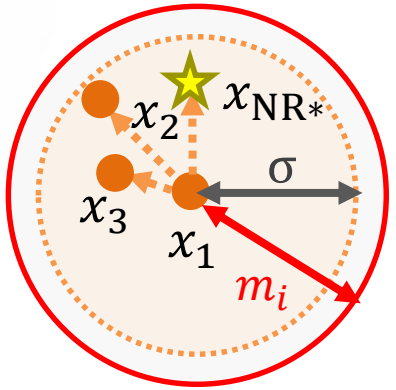
$$\boldsymbol{x}_{loc} = \boldsymbol{x}_{NR*} + A_i^t * rand$$

endif

STEP4: Niche Radius内で大域探索③

$$\boldsymbol{x}_{rnd} = \boldsymbol{x}_{NR_{i*}} + rand(1, D, [-m, m])$$

STEP5: ①,②,③の解候補と現在の解の評価



★: $x_{\mathrm{NR}*}$

●: x_i $(i = 1,2,\ldots,N)$

月着陸探査ミッションの最適着陸地点選定問題に合わせた離散化と交叉操作

深瀬 貴史[1], 橋本 龍一[1], 能島　裕介[1],
増山 直輝[1], 石渕 久生[2]

[1]大阪府立大学
[2]南方科技大学

アルゴリズムの概要

月着陸探査ミッションの最適着陸地点選定問題の特徴

1. 10m精度のデータを基に作成
   ⇨ 決定変数空間の離散化

2. 帯状の実行可能領域
   ⇨ 実行可能領域の形状に合わせた交叉

アルゴリズムの概要

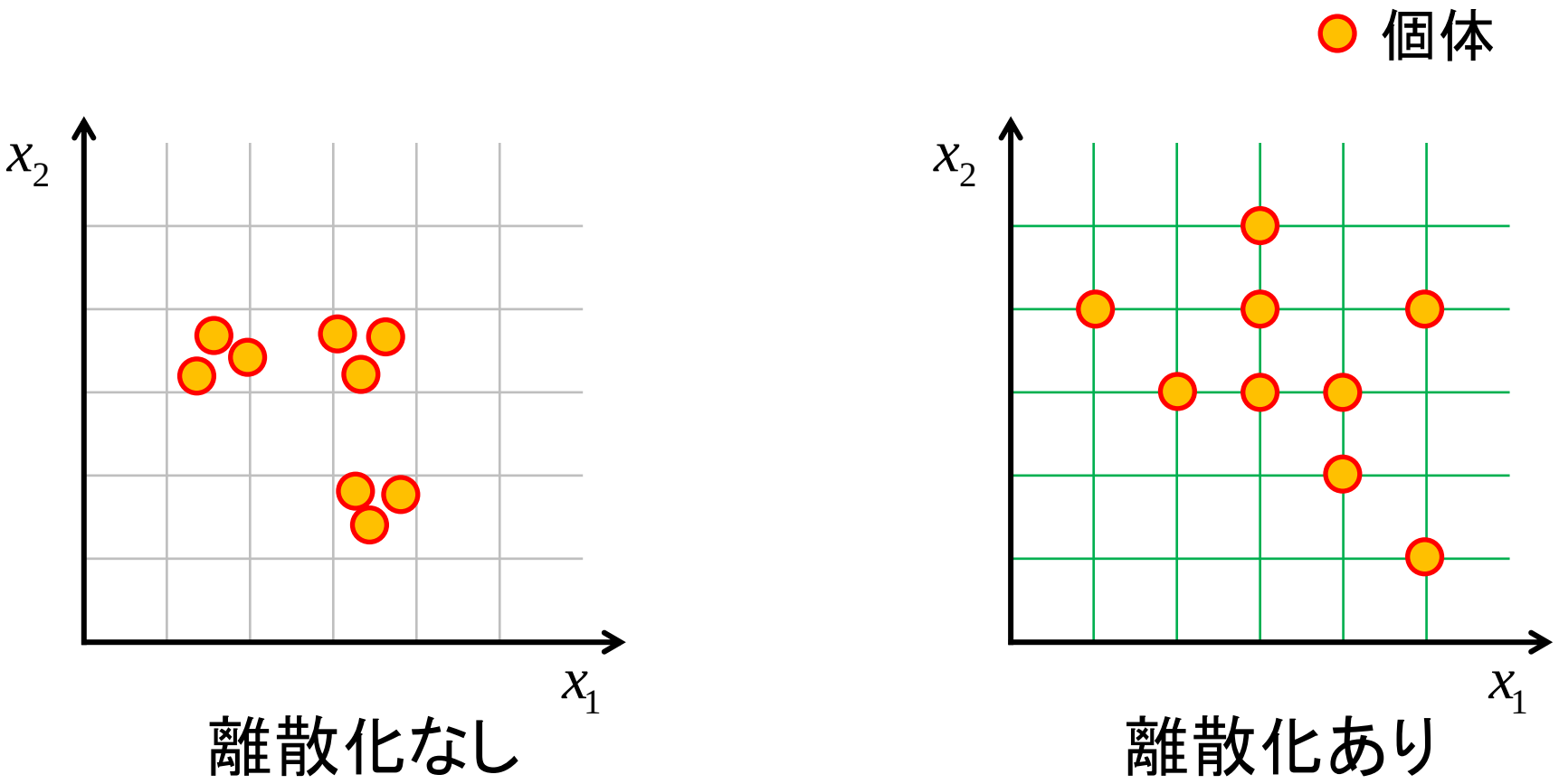
月着陸探査ミッションの最適着陸地点選定問題の特徴

1. 10m精度のデータを基に作成
   ⇨ 決定変数空間の離散化

2. 帯状の実行不可能領域
   ⇨ 実行不可能領域の形状に合わせた交叉

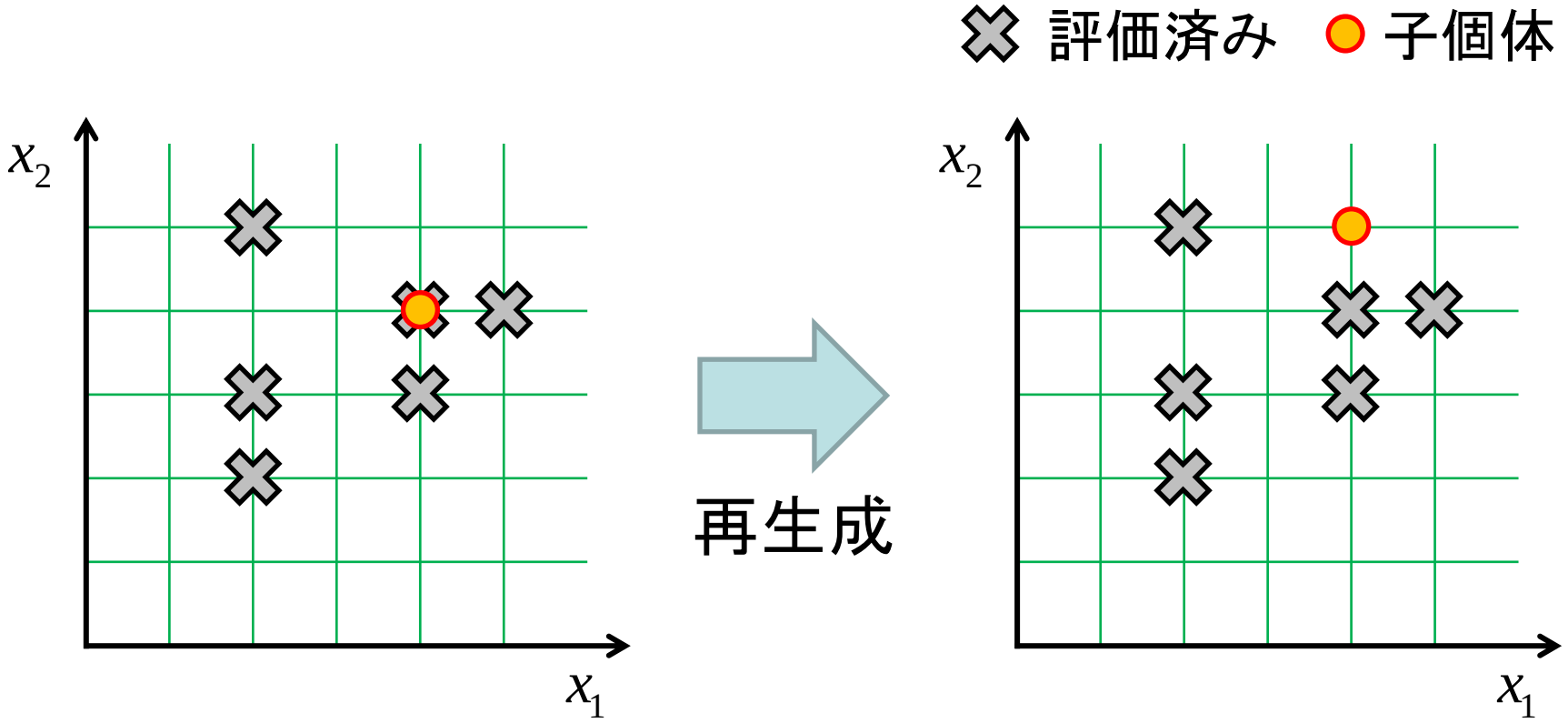
決定変数空間の離散化

問題の精度(1/30,000)以上に細かく探索を行わないため，
決定変数空間の離散化を行う．



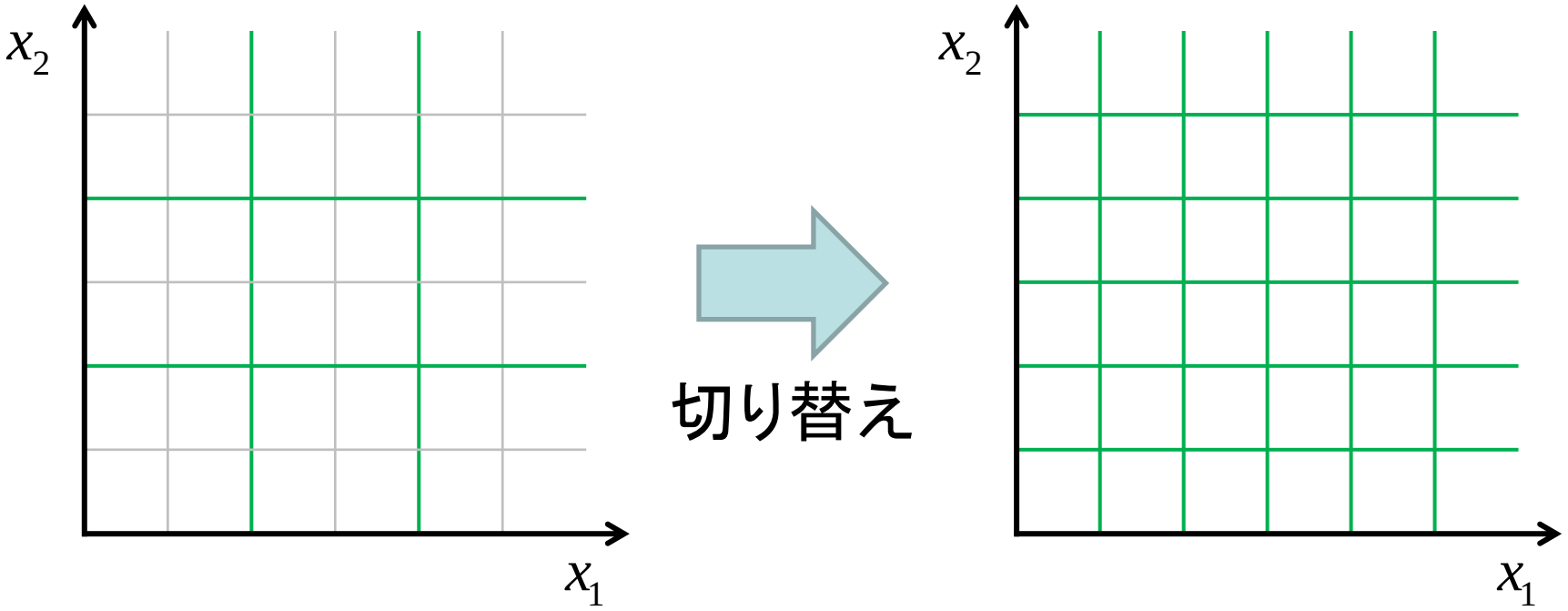
決定変数空間の離散化
無駄な個体評価の削減

評価済みの格子点に子個体が生成された場合，
もう一度生成し直す．



決定変数空間の離散化
離散幅の調整

探索前半は粗い離散幅で大域探索を行い,
探索後半は細かい離散幅で局所探索を行う.



離散幅: 1/10,000

離散幅: 1/30,000

アルゴリズムの概要

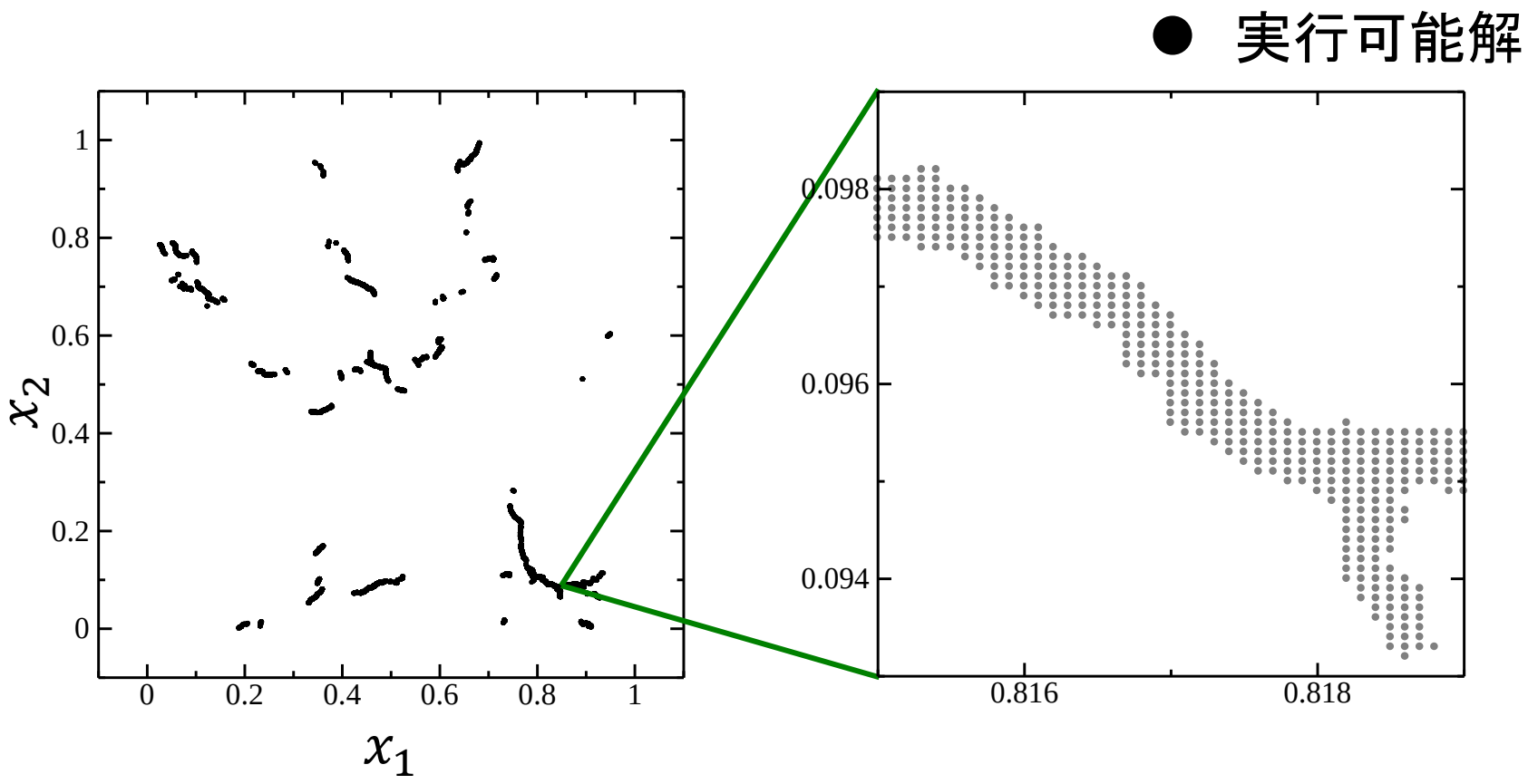
月着陸探査ミッションの最適着陸地点選定問題の特徴

1. 10m精度のデータを基に作成
   ⇨ 決定変数空間の離散化

2. 帯状の実行可能領域
   ⇨ 実行可能領域の形状に合わせた交叉

実行可能領域の形状に合わせた交叉

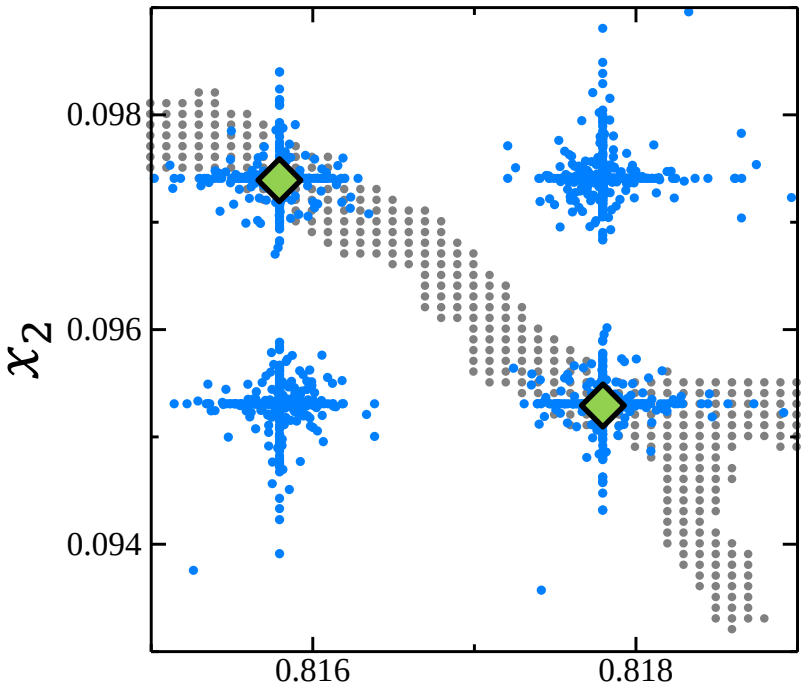
10,000×10,000のグリッドサーチで得られた
実行可能解の分布



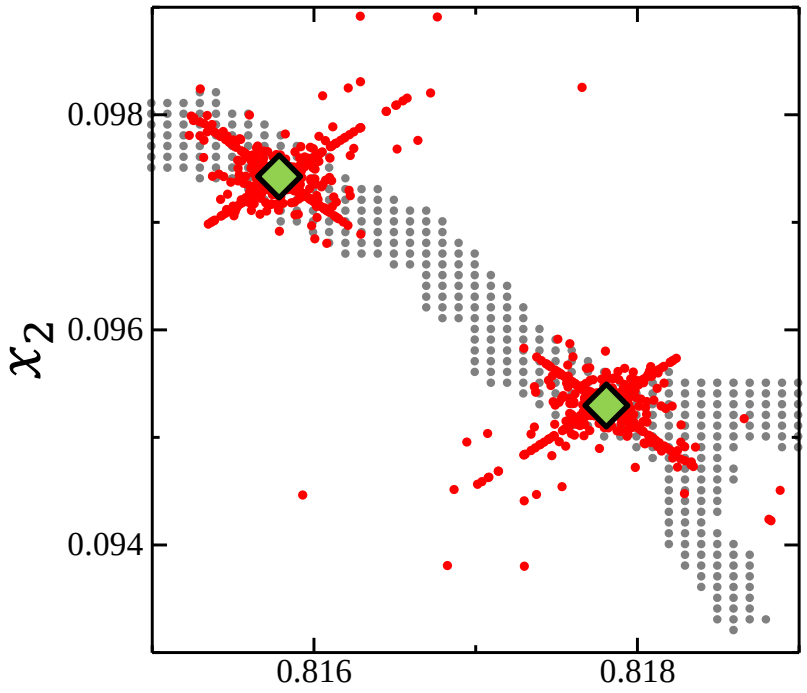
実行可能領域の形状に合わせた交叉

親個体の周りにのみ解を生成する.
親個体を結んだ直線の傾きに合わせて回転処理を加える.

◆親個体 ●● 子個体 ● 実行可能解



従来SBX



改良SBX

チューニングしたパラメータ

突然変異

- 突然変異確率: 0.1
- Distribution index :100

離散幅調整

- 離散幅(前半): 1/10,000
- 離散幅 (後半): 1/30,000
- 切り替え世代: 60世代

多目的クラスタリングベース有望個体囲い込み法mCPIEによる月着陸最適候補地の選定

多目的部門 グループm02

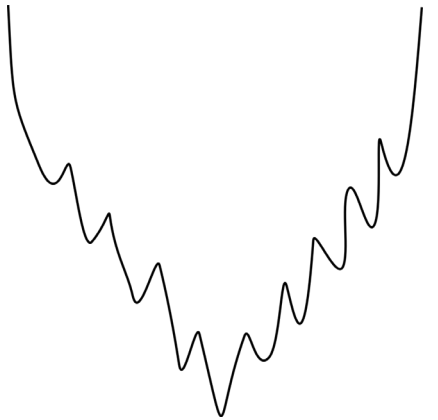
鎌田 一樹 青木勇輔 小野功

東京工業大学

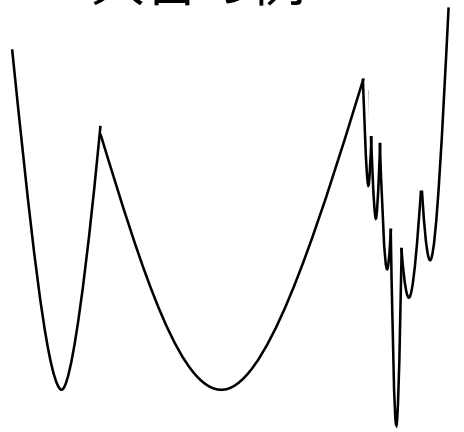
はじめに(1/2)

- 多目的クラスタリングベース有望個体囲い込み法：**mCPIE**
  (Multi-objective Clustering-based Promising Individual Enclosure)
  - 単目的最適化手法**CPIE**を多目的に拡張した手法

- CPIE[戸田 17]
  - 非明示制約付き大域的多峰性ブラックボックス関数最適化において，
    複数の大谷に存在する有力局所解を一度の探索で求めるための探索手法
    - **非明示制約付き問題**
      解の実行可能性しか与えられない問題
      制約違反量を用いることが不可能
    - **大谷**[Boese 95]
      微視的には多くの局所解を持つが，巨視的には１つの大きな谷
    - **大域的多峰性**[池田 02]
      探索空間に複数の大谷が存在する性質
  - 大域的多峰性を有する問題は既存の進化計算手法では探索が困難
    - 特に，探索序盤で最適解の大谷の期待値が他の大谷より悪い場合に困難
  - CPIEではこの問題を克服し，かつ複数の有力局所解の探索が可能



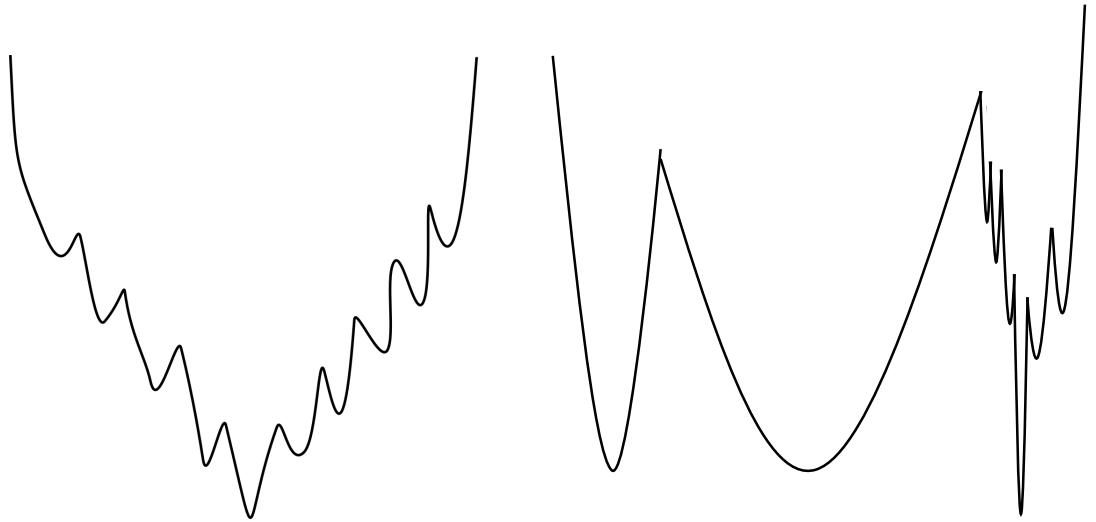
大谷の例



大域的多峰性の例

はじめに(2/2)

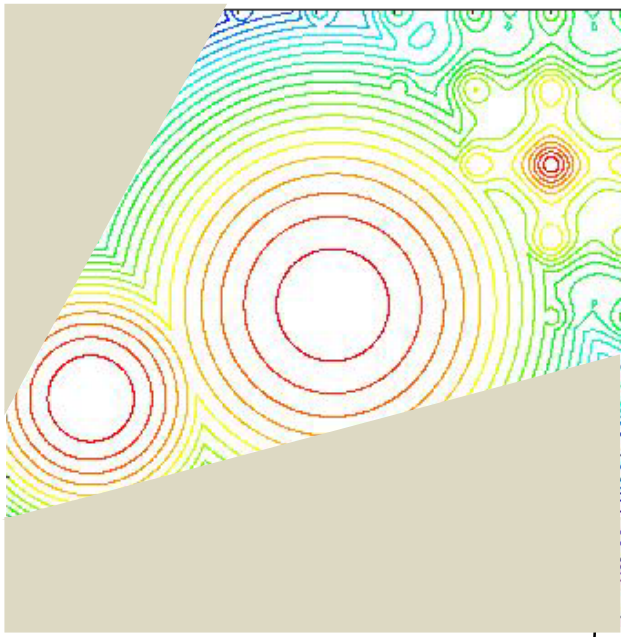
- CPIE[戸田 17]
  - 大域的多峰性を有する問題において，複数の有力局所解を同時に探索可能
- mCPIE (multi-objective CPIE)
  - CPIEを多目的最適化へと拡張
  - Indicatorベースな多目的最適化
  - HV値(Hypervolume [Auger 09])を最大化
  - 月着陸最適候補地の選定問題を
    非明示制約付き大域的多峰性問題とみなす

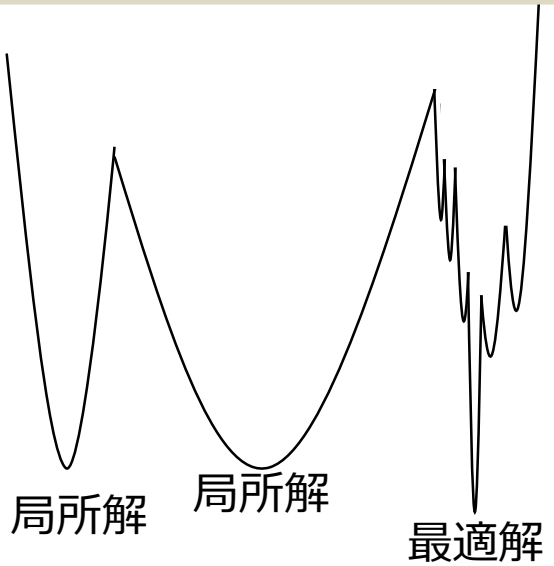


CPIEのアルゴリズム(1/6)

●解を囲う楕円体を更新・分割して探索を進める

1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る

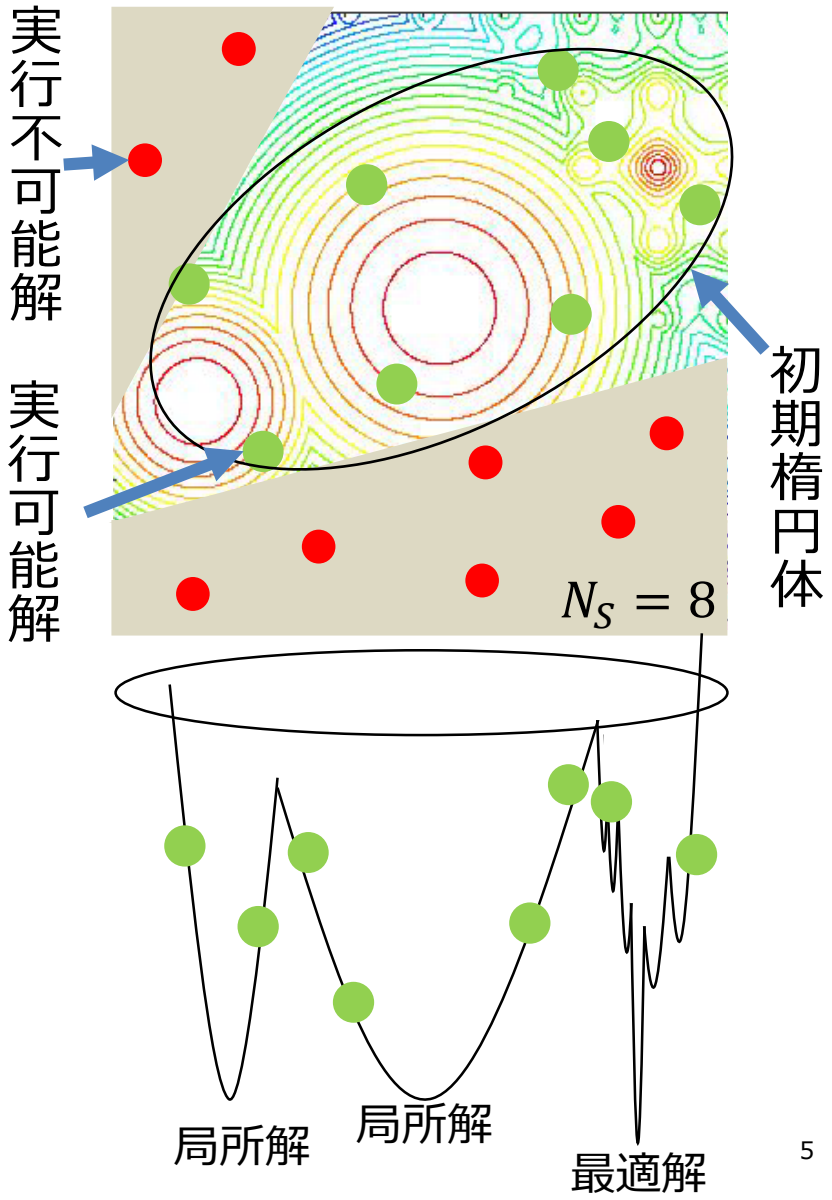




CPIEのアルゴリズム(2/6)

●解を囲う楕円体を更新・分割して探索を進める

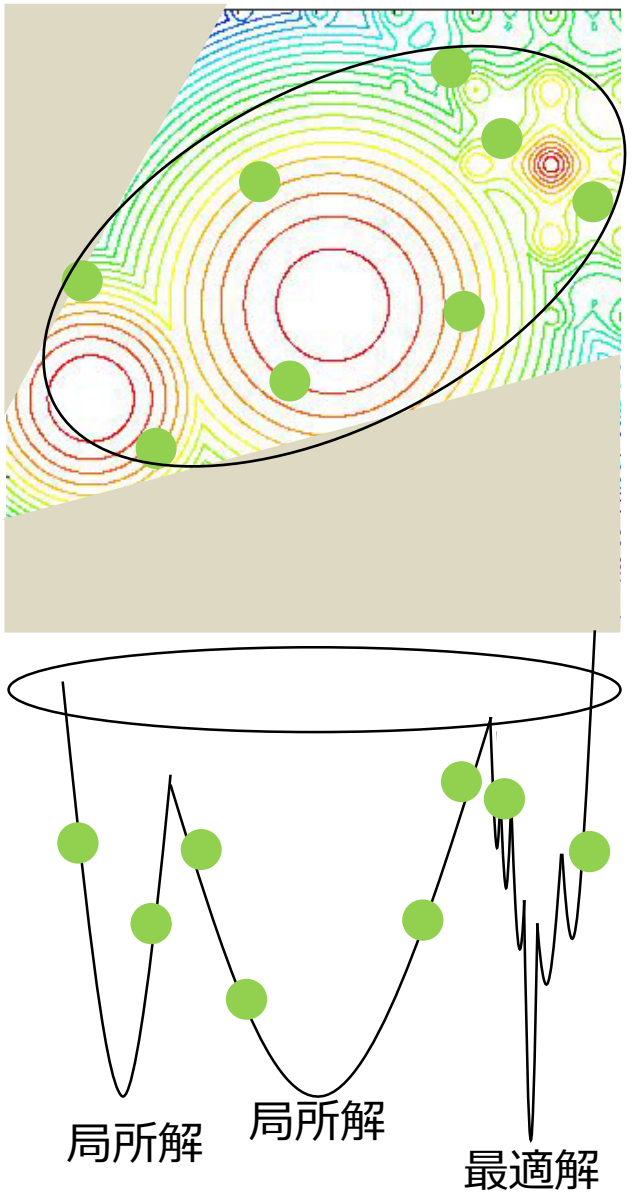
1. 初期化
   - ➢ 一様ランダムに解を生成
   - ➢ 実行可能解がN_S個生成されたら，初期楕円体で囲う
2. 解生成と解集合の更新
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る



CPIEのアルゴリズム(3/6)

●解を囲う楕円体を更新・分割して探索を進める

1. 初期化
2. 解生成と解集合の更新
   - 探索を進める集合を選択
   - 楕円体の表面上に解を生成
   - 解集合内の最悪個体よりも良い評価値ならば，
     生成解を解集合に追加し，最悪個体を除去
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る



CPIEのアルゴリズム(3/6)

●解を囲う楕円体を更新・分割して探索を進める

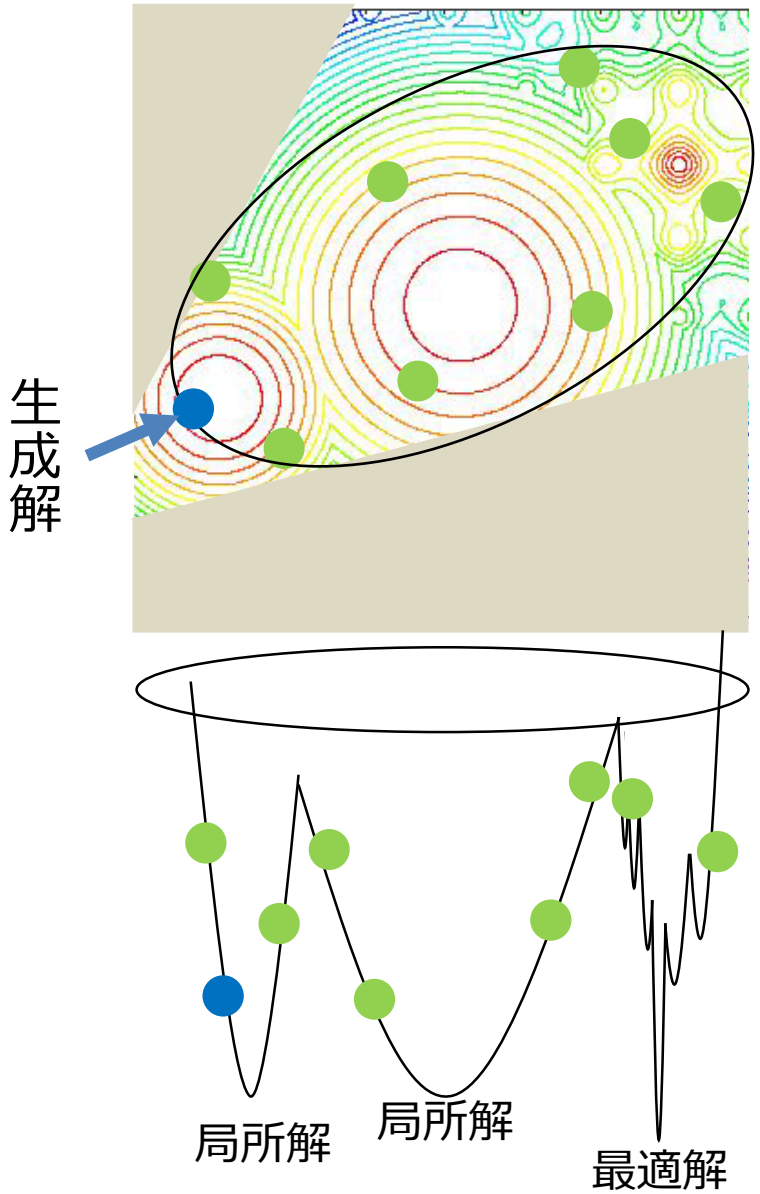
1. 初期化

2. 解生成と解集合の更新
   - ➢ 探索を進める集合を選択
   - ➢ 楕円体の表面上に解を生成
   - ➢ 解集合内の最悪個体よりも良い評価値ならば，生成解を解集合に追加し，最悪個体を除去

3. 楕円体の更新

4. 解集合と楕円体の分割

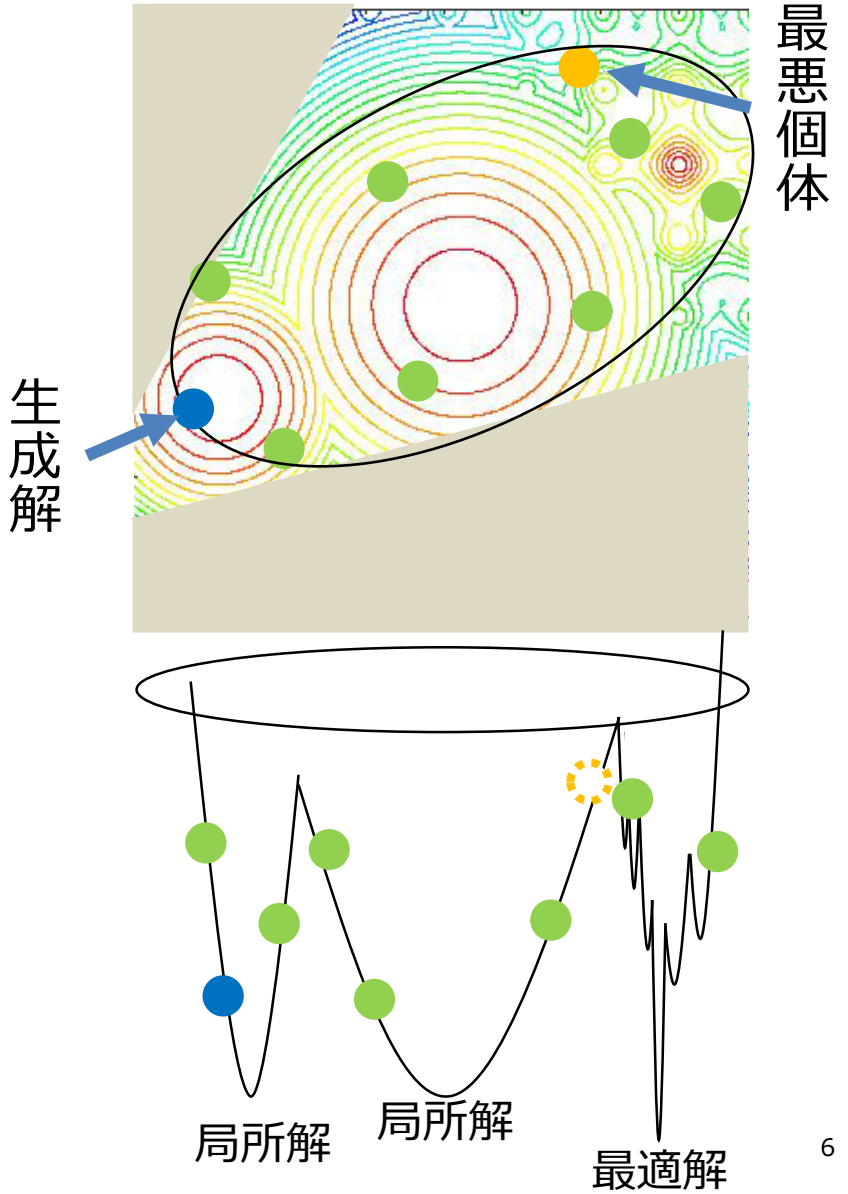
5. 2. へ戻る



CPIEのアルゴリズム(3/6)

●解を囲う楕円体を更新・分割して探索を進める

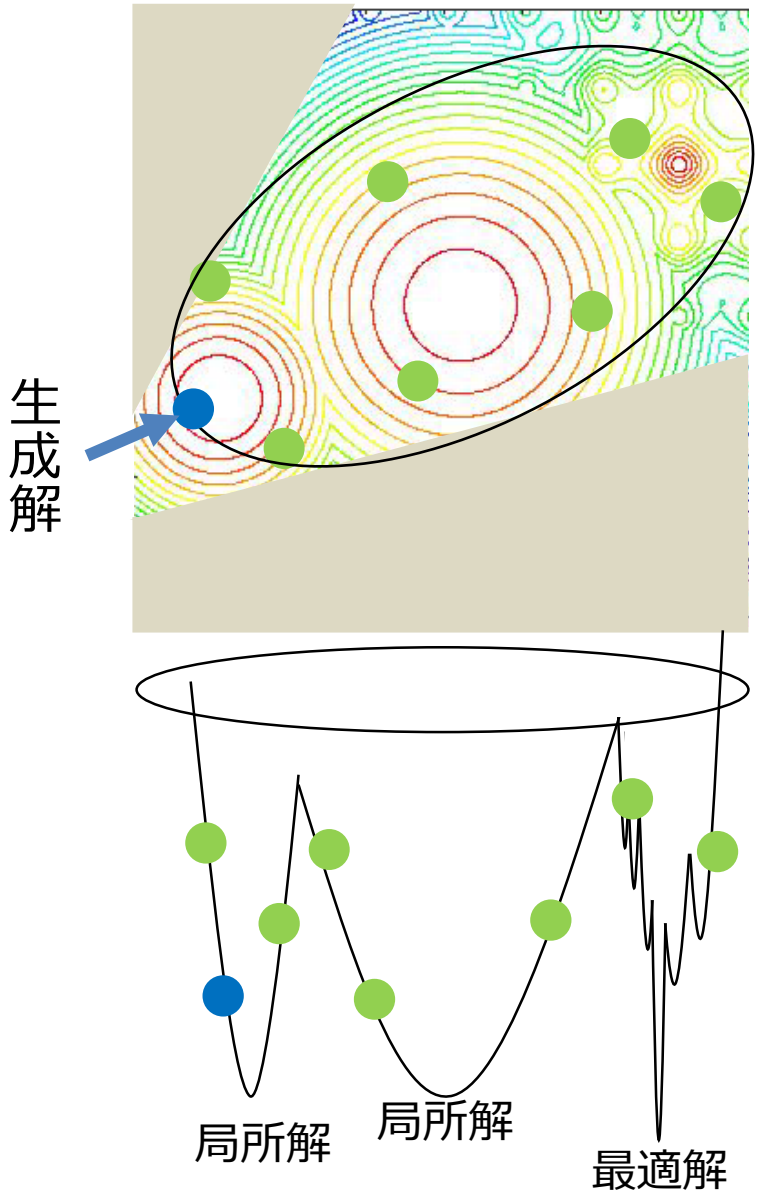
1. 初期化
2. 解生成と解集合の更新
   - ➢ 探索を進める集合を選択
   - ➢ 楕円体の表面上に解を生成
   - ➢ 解集合内の最悪個体よりも良い評価値ならば，生成解を解集合に追加し，最悪個体を除去
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る



CPIEのアルゴリズム(4/6)

●解を囲う楕円体を更新・分割して探索を進める

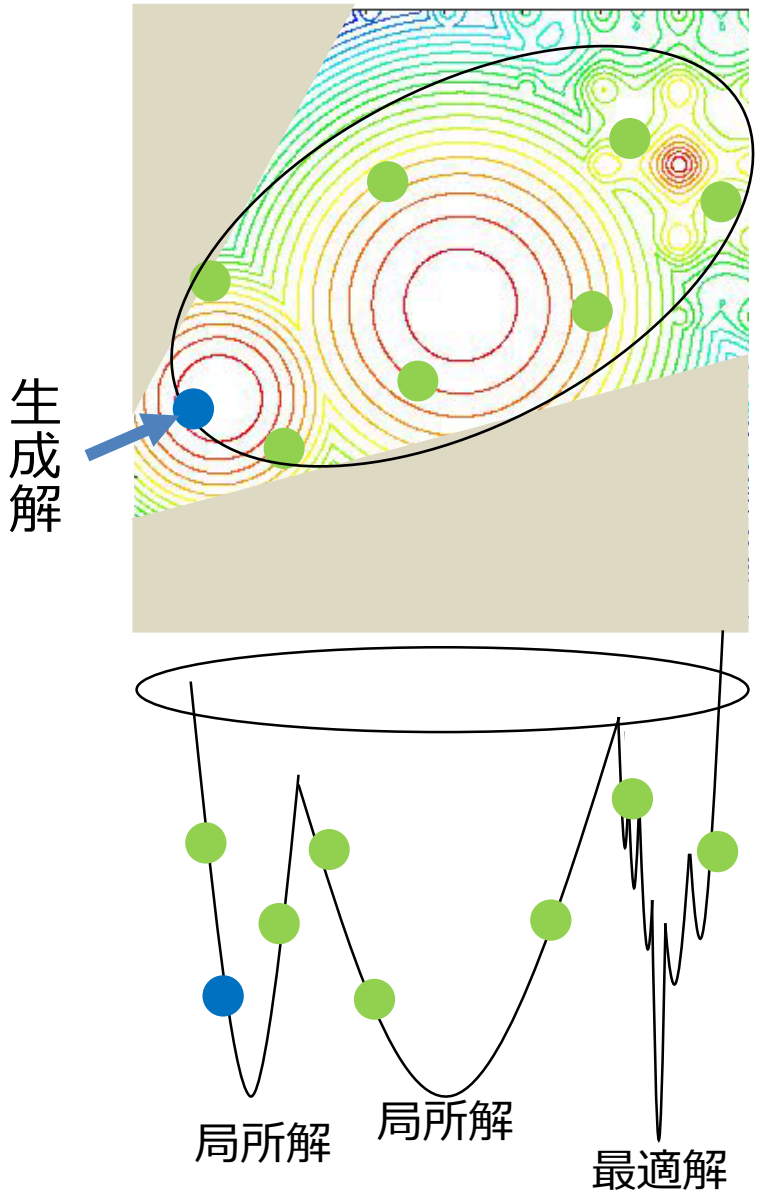
1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
   - ➢ 解集合が更新された場合
     - • 生成された解を囲うように楕円体を拡大
   - ➢ 更新されなかった場合
     - • 生成された解を囲わないように楕円体を縮小
4. 解集合と楕円体の分割
5. 2. へ戻る



CPIEのアルゴリズム(4/6)

●解を囲う楕円体を更新・分割して探索を進める

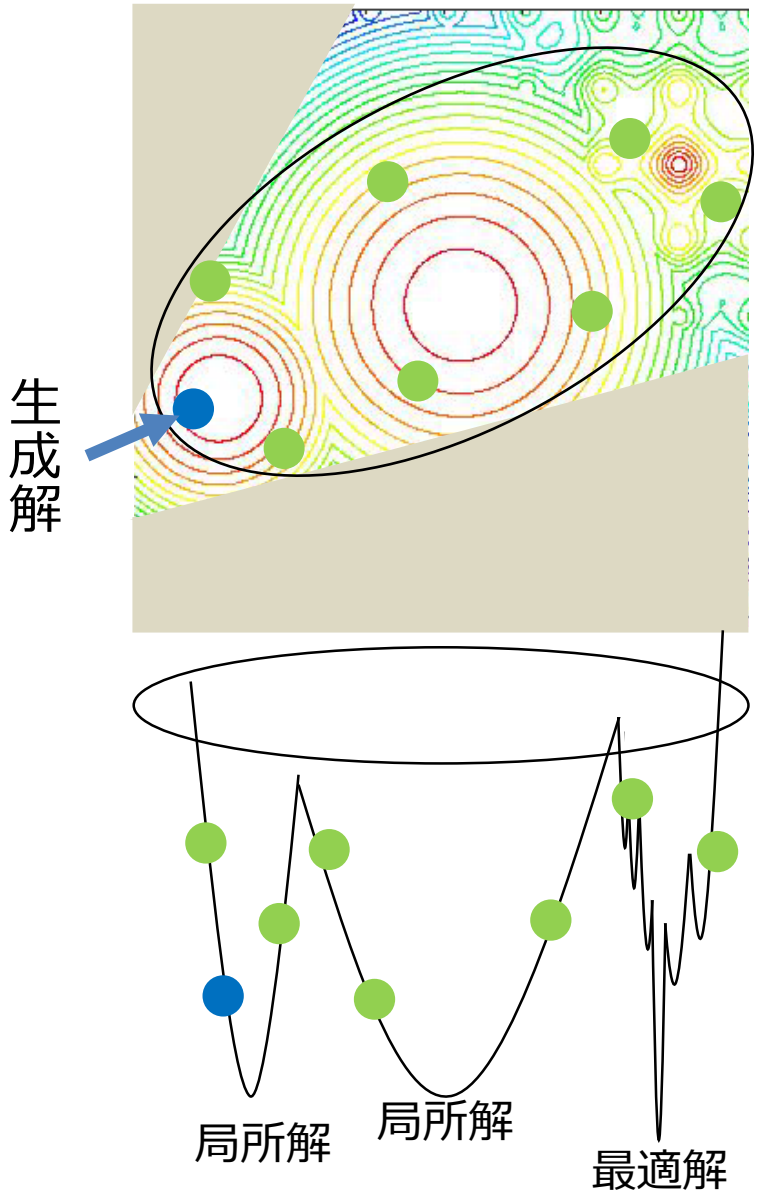
1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
   - ➢ 解集合が更新された場合
     - • 生成された解を囲うように楕円体を拡大
   - ➢ 更新されなかった場合
     - • 生成された解を囲わないように楕円体を縮小
4. 解集合と楕円体の分割
5. 2. へ戻る



CPIEのアルゴリズム(4/6)

●解を囲う楕円体を更新・分割して探索を進める

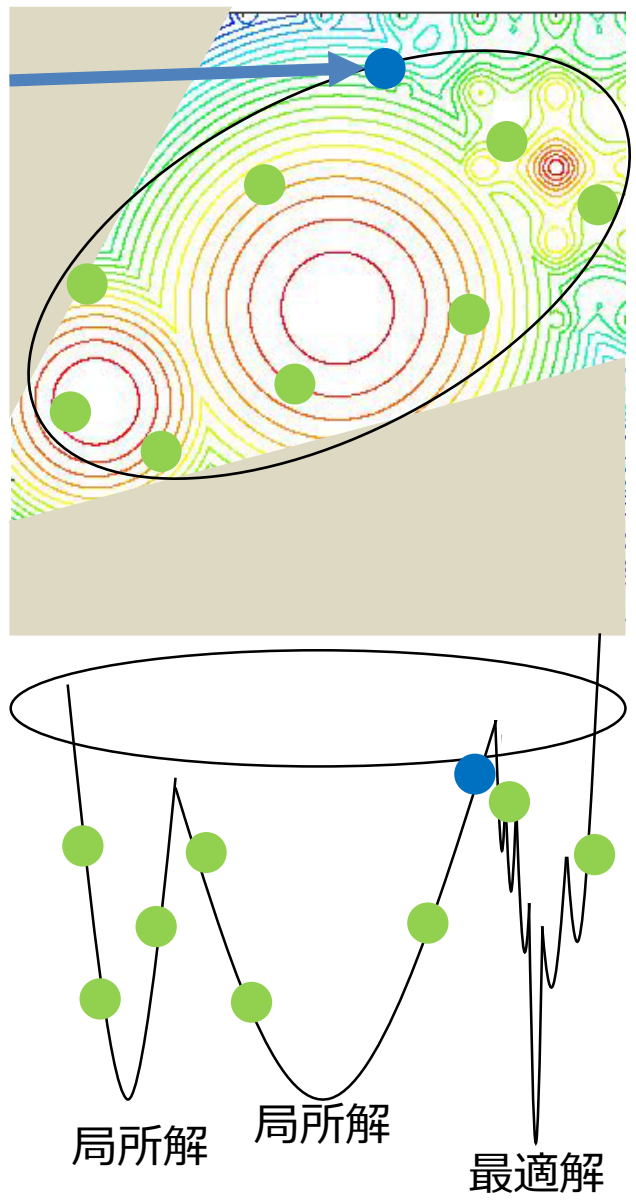
1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
   - ➢ 解集合が更新された場合
     - 生成された解を囲うように楕円体を拡大
   - ➢ 更新されなかった場合
     - 生成された解を囲わないように楕円体を縮小
4. 解集合と楕円体の分割
5. 2. へ戻る



CPIEのアルゴリズム(4/6)

●解を囲う楕円体を更新・分割して探索を進める

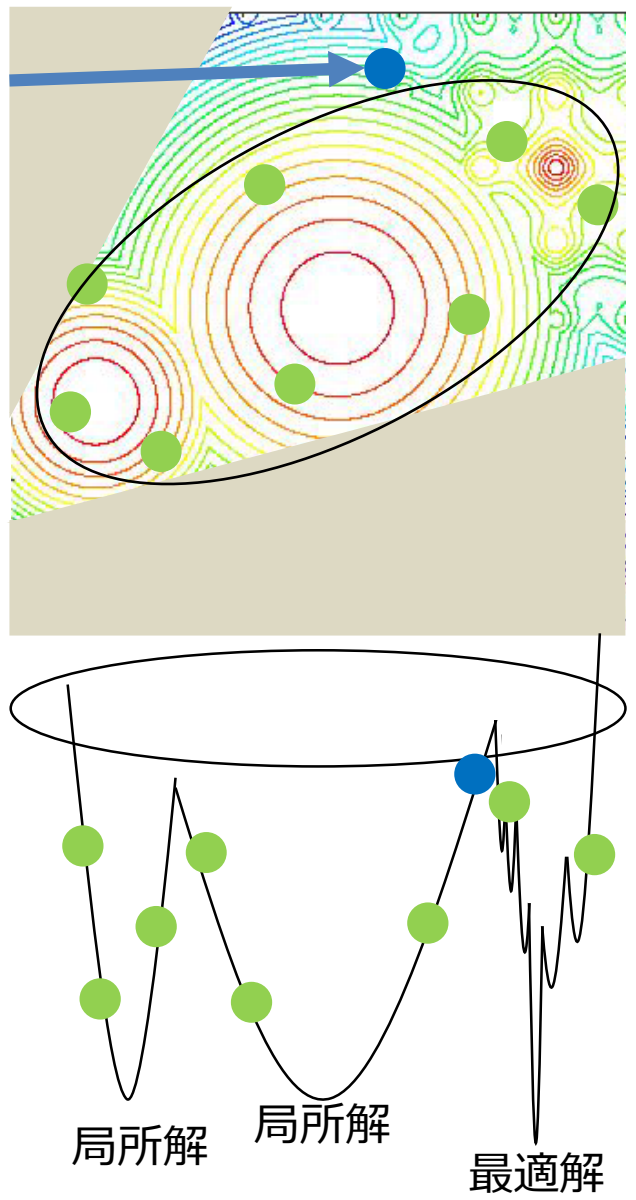
1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
   - ➢ 解集合が更新された場合
     - • 生成された解を囲うように楕円体を拡大
   - ➢ 更新されなかった場合
     - • 生成された解を囲わないように楕円体を縮小
4. 解集合と楕円体の分割
5. 2. へ戻る



CPIEのアルゴリズム(4/6)

●解を囲う楕円体を更新・分割して探索を進める

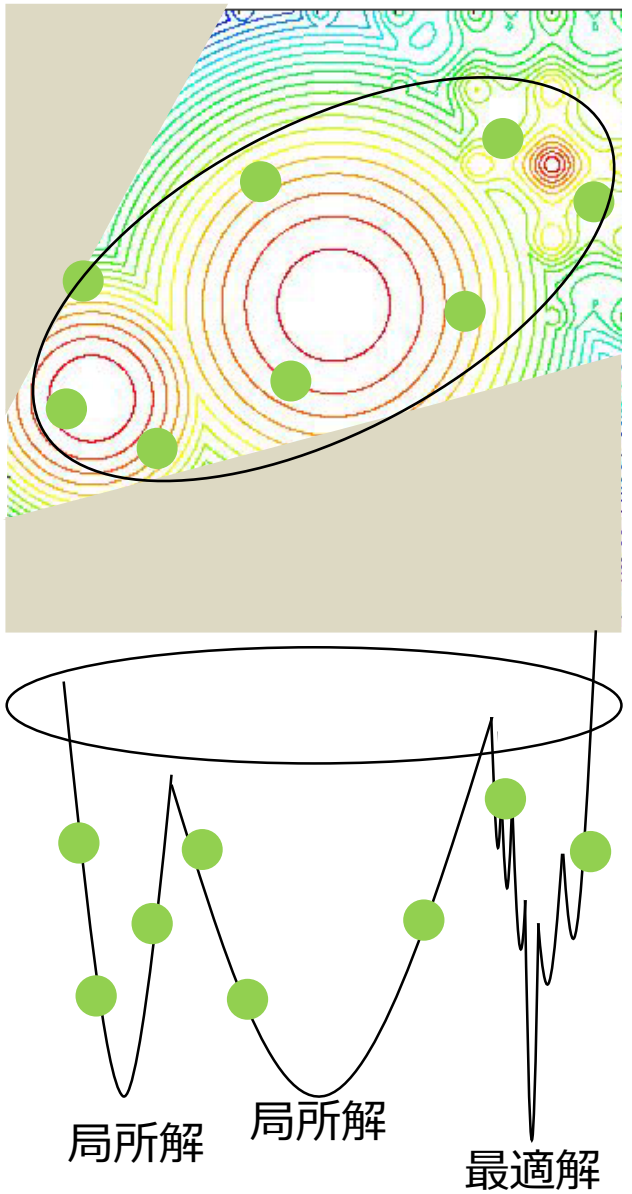
1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
   - ➢ 解集合が更新された場合
     - • 生成された解を囲うように楕円体を拡大
   - ➢ 更新されなかった場合
     - • 生成された解を囲わないように楕円体を縮小
4. 解集合と楕円体の分割
5. 2. へ戻る



CPIEのアルゴリズム(5/6)

●解を囲う楕円体を更新・分割して探索を進める

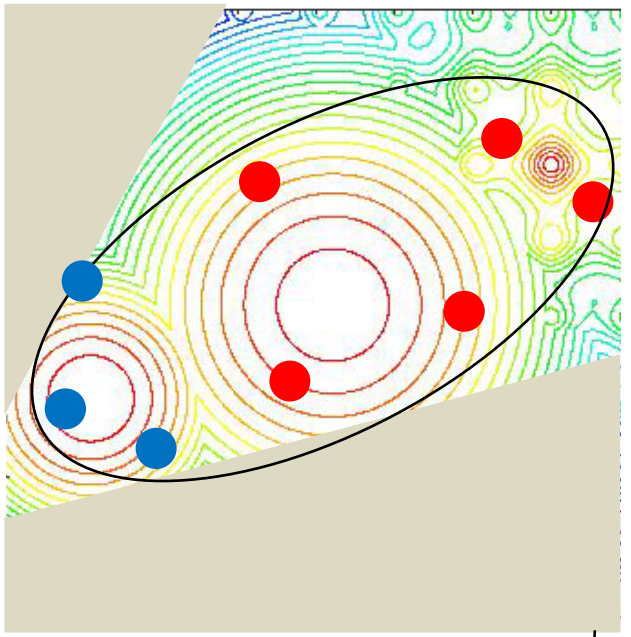
1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
4. 解集合と楕円体の分割
   - ➢ 探索が停滞し，解集合内の個体の分布が非連結と判断された場合は解集合と楕円体を分割
   - ➢ 解集合の分割にはk-means法を用いる
5. 2. へ戻る

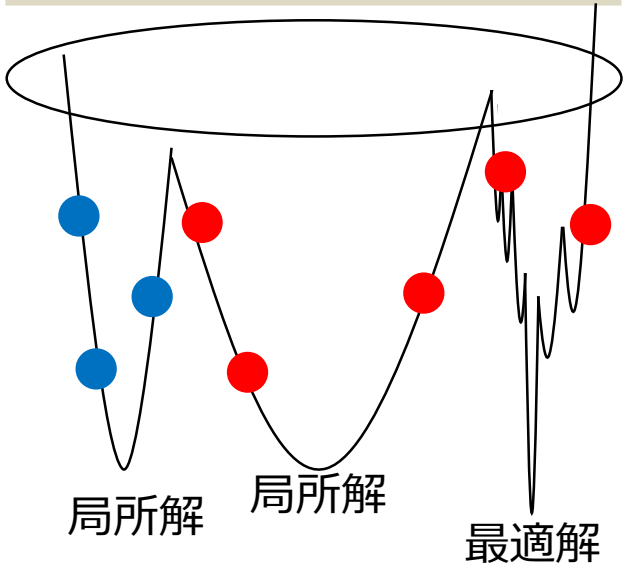


CPIEのアルゴリズム(5/6)

●解を囲う楕円体を更新・分割して探索を進める

1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
4. 解集合と楕円体の分割
   - ➢ 探索が停滞し，解集合内の個体の分布が非連結と判断された場合は解集合と楕円体を分割
   - ➢ 解集合の分割にはk-means法を用いる
5. 2. へ戻る

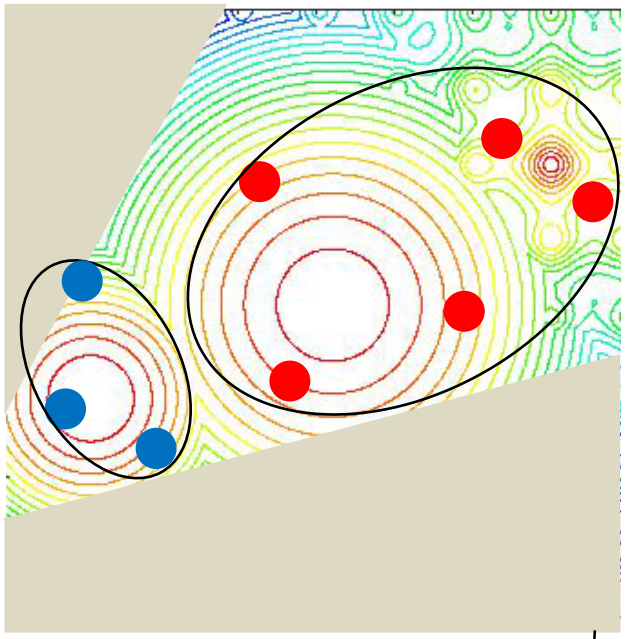


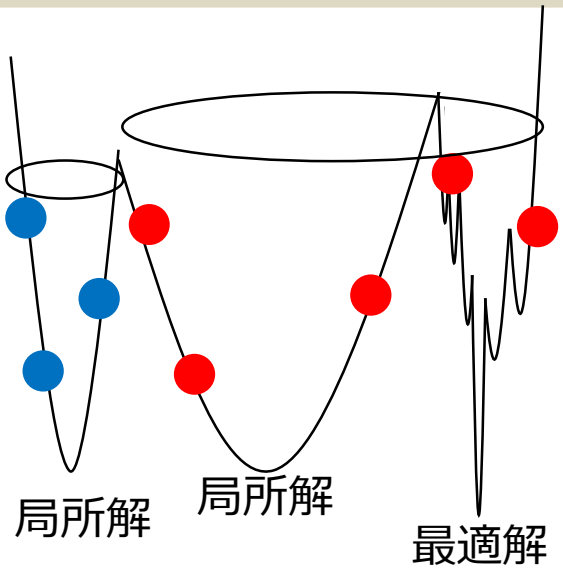


CPIEのアルゴリズム(5/6)

●解を囲う楕円体を更新・分割して探索を進める

1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
4. 解集合と楕円体の分割
   - ➢ 探索が停滞し，解集合内の個体の分布が非連結と判断された場合は解集合と楕円体を分割
   - ➢ 解集合の分割にはk-means法を用いる
5. 2. へ戻る

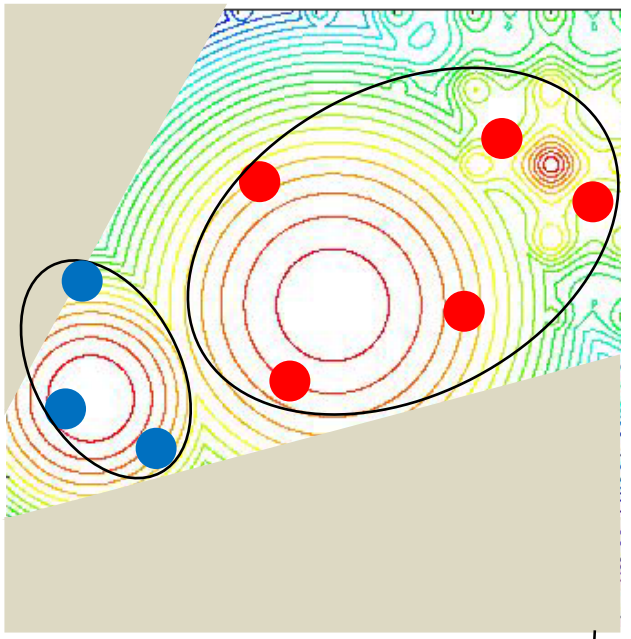


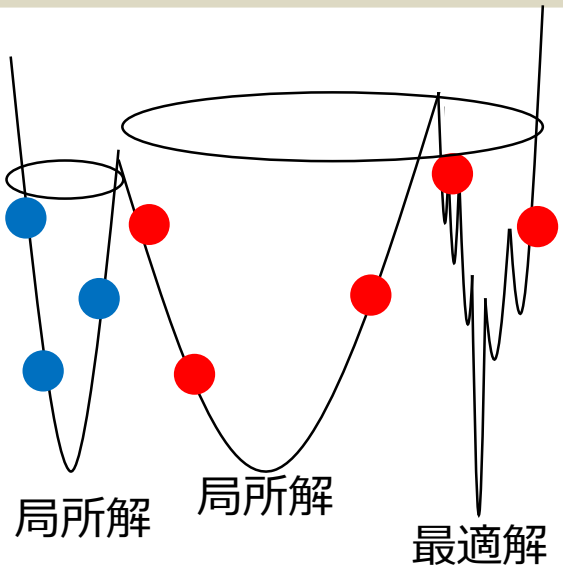


CPIEのアルゴリズム(5/6)

●解を囲う楕円体を更新・分割して探索を進める

1. 初期化
2. 解生成と解集合の更新
3. 楕円体の更新
4. 解集合と楕円体の分割
   - ➢ 探索が停滞し，解集合内の個体の分布が非連結と判断された場合は解集合と楕円体を分割
   - ➢ 解集合の分割にはk-means法を用いる
5. 2. へ戻る

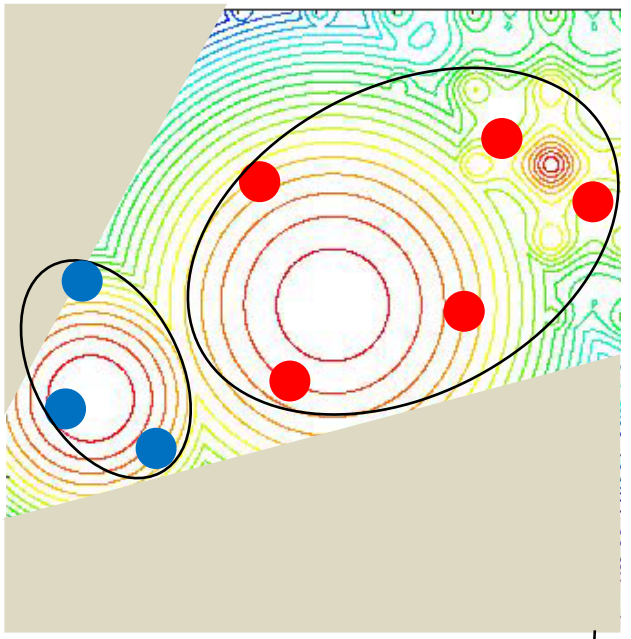


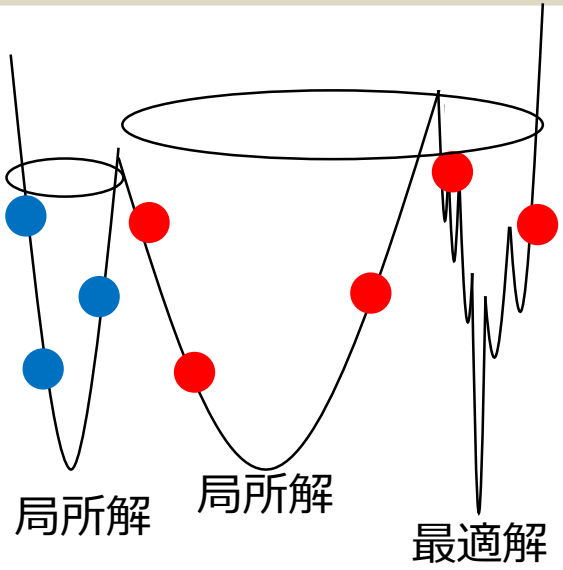


CPIEのアルゴリズム(6/6)

●解を囲う楕円体を更新・分割して探索を進める

1. 初期化
2. 解生成と解集合の更新
   - 探索を進める集合を選択
     - 個体数N_S個未満の集合があるならば個体を追加
     - 改善される見込みの高い集合を選択
   - 楕円体の表面上に解を生成
   - 解集合内の最悪個体よりも良い評価値ならば，生成解を解集合に追加し，最悪個体を除去
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る

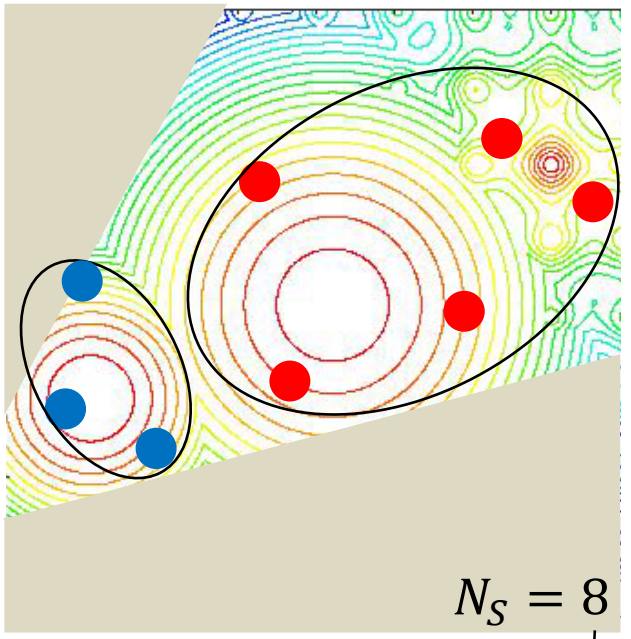


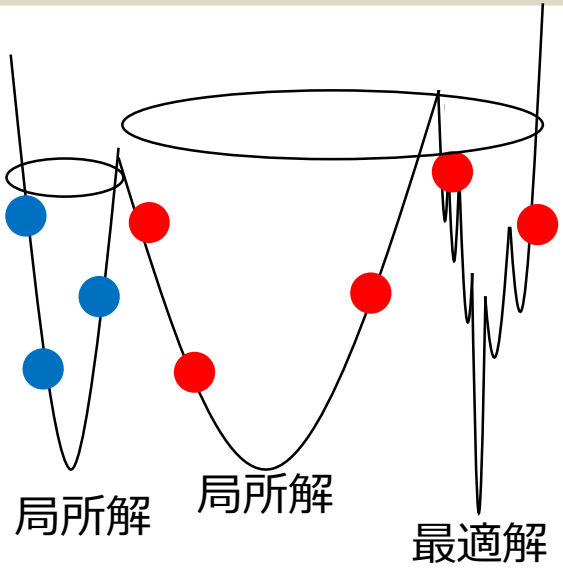


CPIEのアルゴリズム(6/6)

●解を囲う楕円体を更新・分割して探索を進める

1. 初期化
2. 解生成と解集合の更新
   - ➢ 探索を進める集合を選択
     - 個体数N_S個未満の集合があるならば個体を追加
     - 改善される見込みの高い集合を選択
   - ➢ 楕円体の表面上に解を生成
   - ➢ 解集合内の最悪個体よりも良い評価値ならば，生成解を解集合に追加し，最悪個体を除去
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る

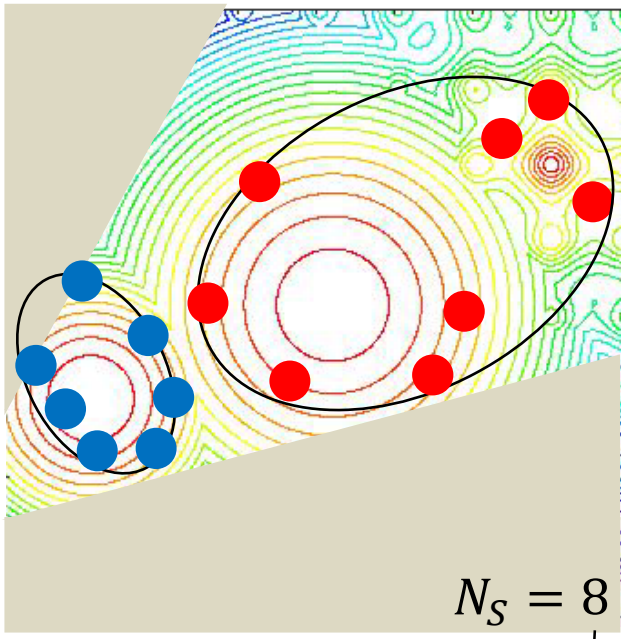


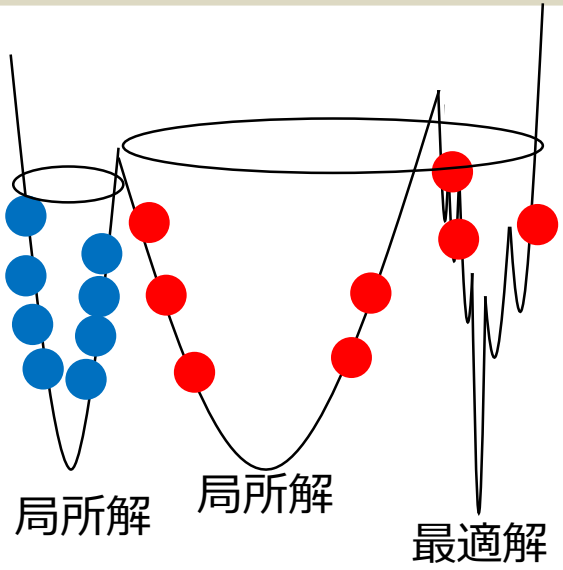


CPIEのアルゴリズム(6/6)

●解を囲う楕円体を更新・分割して探索を進める

1. 初期化
2. 解生成と解集合の更新
   - ➢ 探索を進める集合を選択
     - 個体数N_S個未満の集合があるならば個体を追加
     - 改善される見込みの高い集合を選択
   - ➢ 楕円体の表面上に解を生成
   - ➢ 解集合内の最悪個体よりも良い評価値ならば，生成解を解集合に追加し，最悪個体を除去
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る

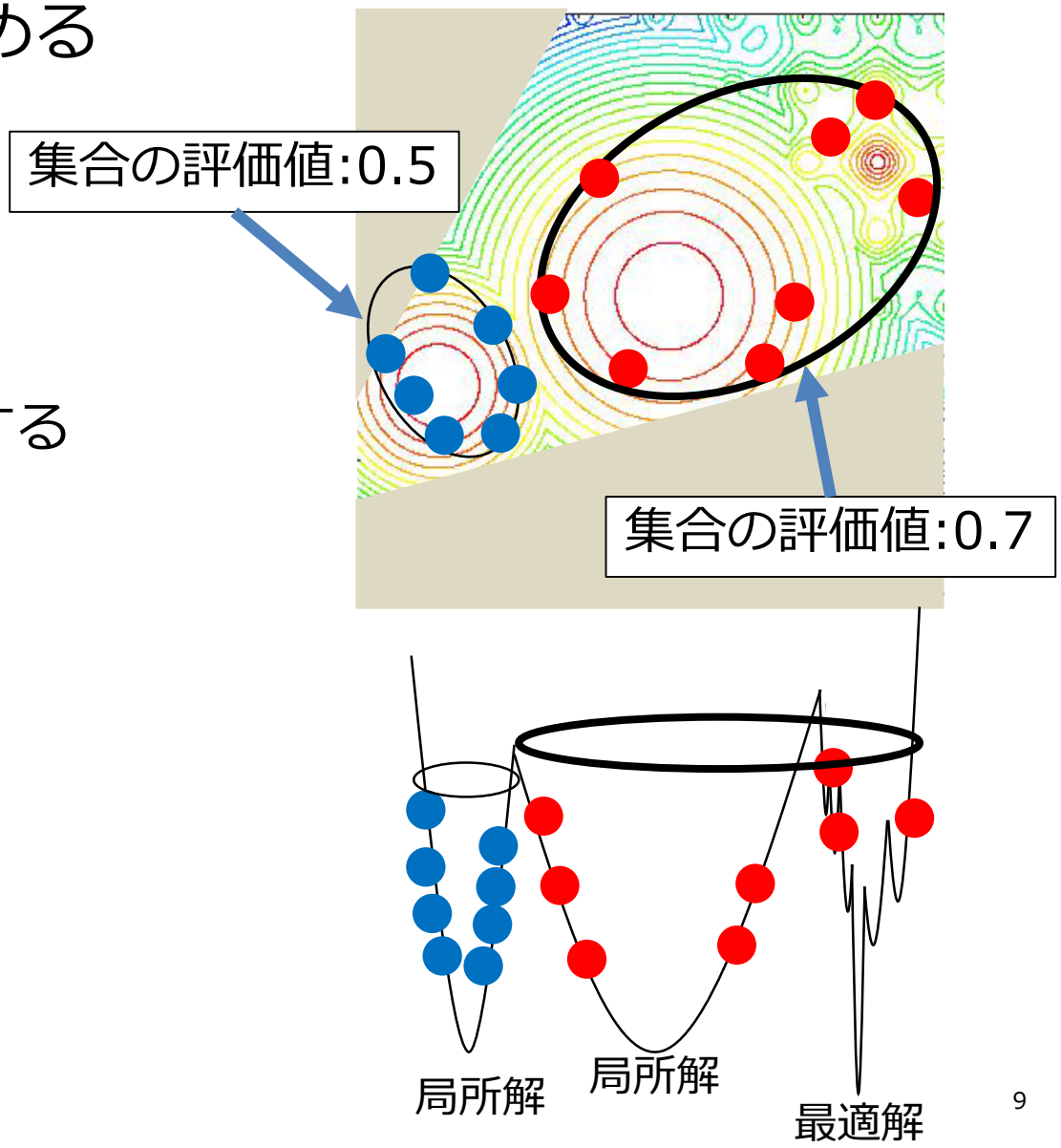




CPIEのアルゴリズム(6/6)

●解を囲う楕円体を更新・分割して探索を進める

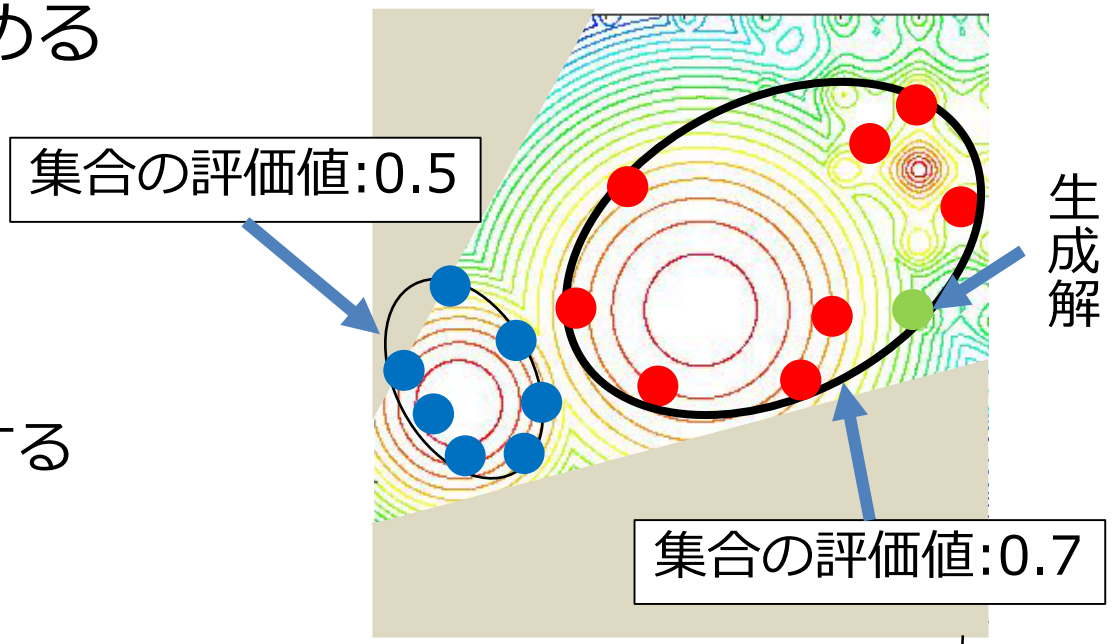
1. 初期化
2. 解生成と解集合の更新
   - ➢ 探索を進める集合を選択
     - 個体数N_S個未満の集合があるならば個体を追加する
     - 改善される見込みの高い集合を選択
   - ➢ 楕円体の表面上に解を生成
   - ➢ 解集合内の最悪個体よりも良い評価値ならば，生成解を解集合に追加し，最悪個体を除去
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る

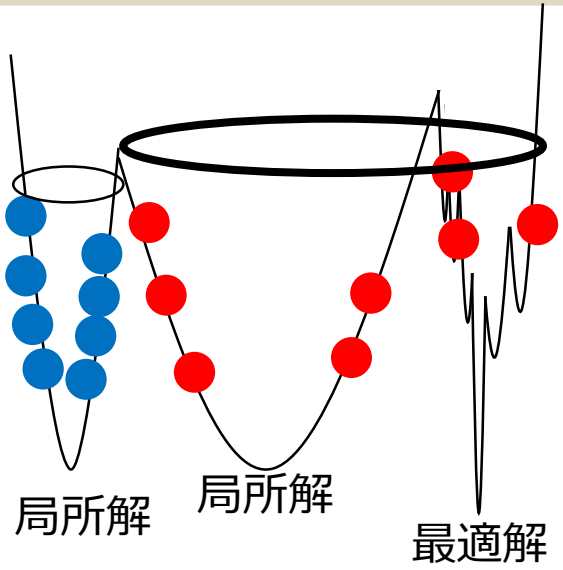


CPIEのアルゴリズム(6/6)

●解を囲う楕円体を更新・分割して探索を進める

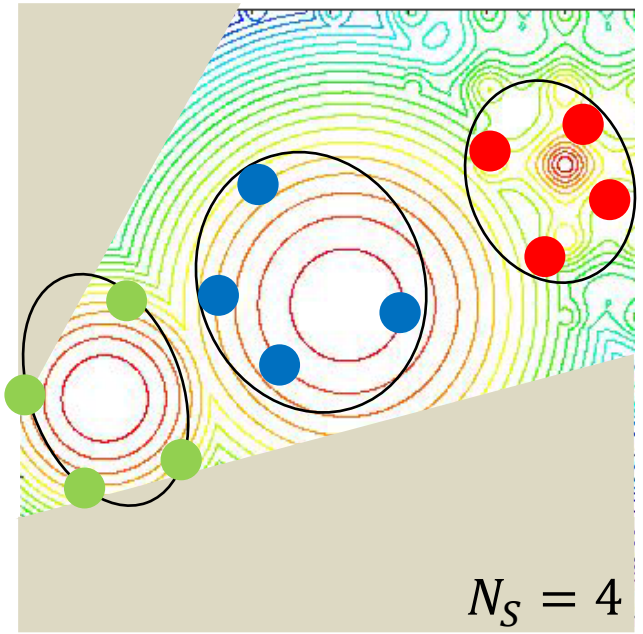
1. 初期化
2. 解生成と解集合の更新
   - 探索を進める集合を選択
     - 個体数N_S個未満の集合があるならば個体を追加する
     - 改善される見込みの高い集合を選択
   - 楕円体の表面上に解を生成
   - 解集合内の最悪個体よりも良い評価値ならば，生成解を解集合に追加し，最悪個体を除去
3. 楕円体の更新
4. 解集合と楕円体の分割
5. 2. へ戻る





CPIEの多目的最適化への拡張：mCPIE

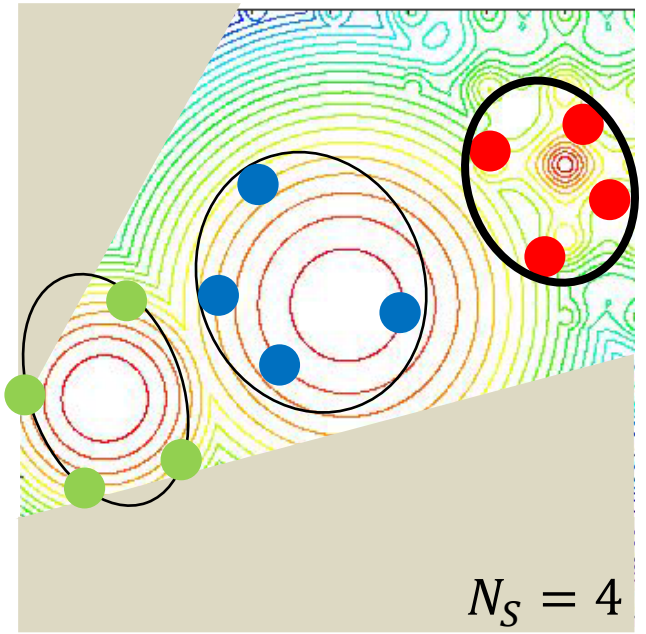
1. 集合の評価値として集合が持つHV値を利用
   - 確率$(1-\varepsilon)$で最もHV値の良い集合を解生成時に選択
   - 確率εでランダムに集合を選択，　$\varepsilon = 0.2$と設定(ε-greedy)
2. 個体の交替指標としてHV値の改善量を利用
   - 生成個体$\boldsymbol{x}$を集合内の個体と交替したときのHV値の改善量を全て計算
   - 最も改善量の大きい個体と生成個体を交代
     - 改善がない場合は交替しない
     - 集団が保有する各評価関数の最小値を有する個体とは交替しない



CPIEの多目的最適化への拡張：mCPIE

1. 集合の評価値として集合が持つHV値を利用
   - 確率$(1-\varepsilon)$で最もHV値の良い集合を解生成時に選択
   - 確率εでランダムに集合を選択，　$\varepsilon = 0.2$と設定(ε-greedy)

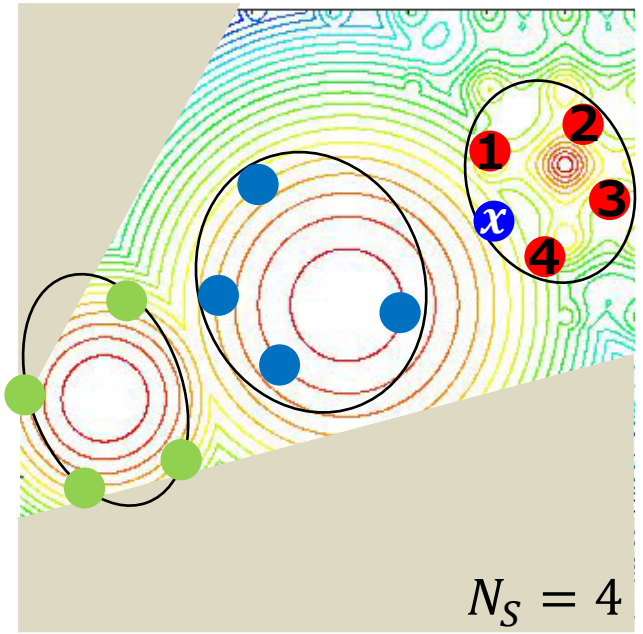
2. 個体の交替指標としてHV値の改善量を利用
   - 生成個体$\boldsymbol{x}$を集合内の個体と交替したときのHV値の改善量を全て計算
   - 最も改善量の大きい個体と生成個体を交代
     - 改善がない場合は交替しない
     - 集団が保有する各評価関数の最小値を有する個体とは交替しない



CPIEの多目的最適化への拡張：mCPIE

1. 集合の評価値として集合が持つHV値を利用
   - 確率$(1-\varepsilon)$で最もHV値の良い集合を解生成時に選択
   - 確率εでランダムに集合を選択，　$\varepsilon = 0.2$と設定(ε-greedy)

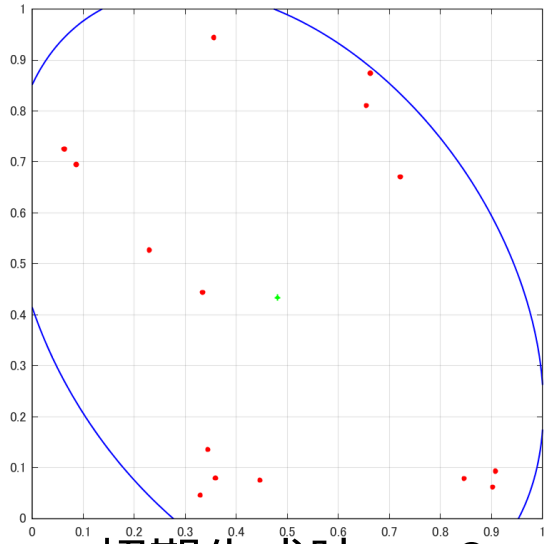
2. 個体の交替指標としてHV値の改善量を利用
   - 生成個体$\boldsymbol{x}$を集合内の個体と交替したときのHV値の改善量を全て計算
   - 最も改善量の大きい個体と生成個体を交代
     - 改善がない場合は交替しない
     - 集団が保有する各評価関数の最小値を有する個体とは交替しない

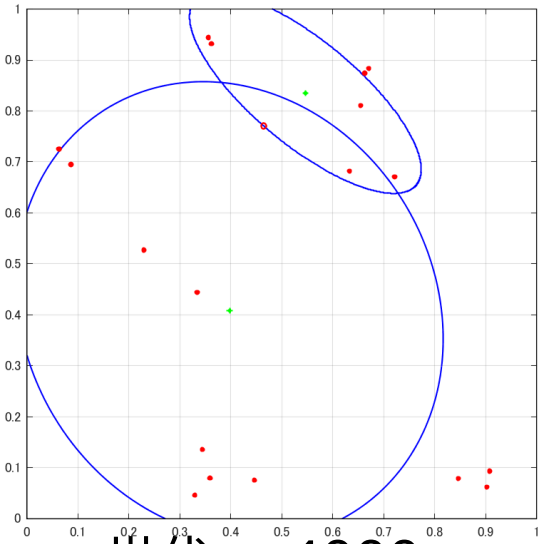


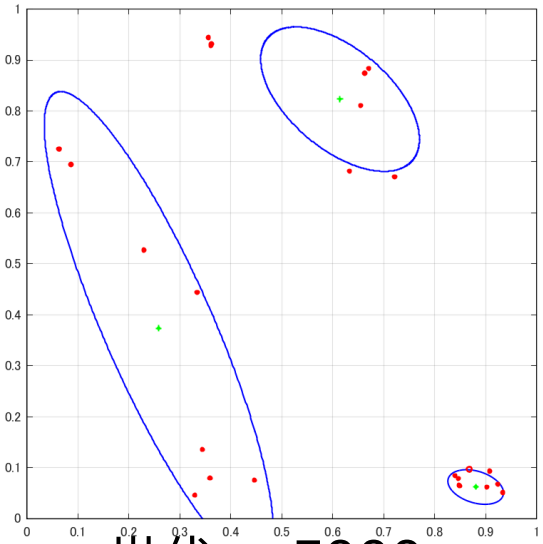
1 2 3 4 のHV値：0.70

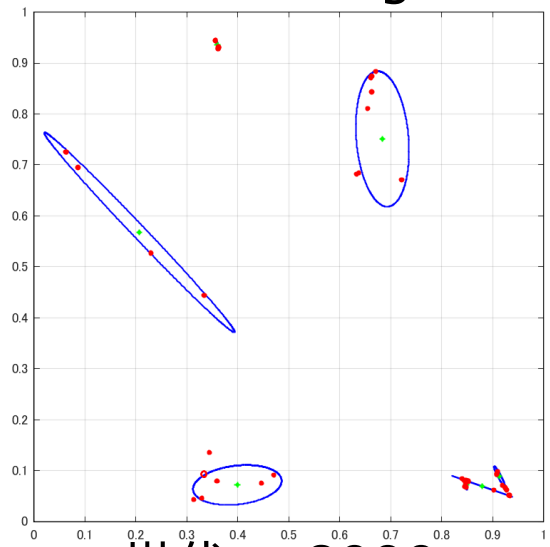
x 2 3 4 のHV値：**0.72**
1 x 3 4 のHV値：0.60
1 2 x 4 のHV値：0.65
1 2 3 x のHV値：**0.75**

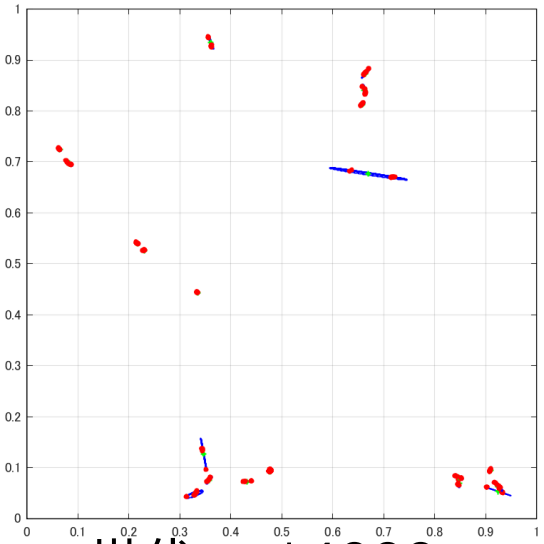
月着陸最適候補地の選定問題に対するmCPIEの探索の様子

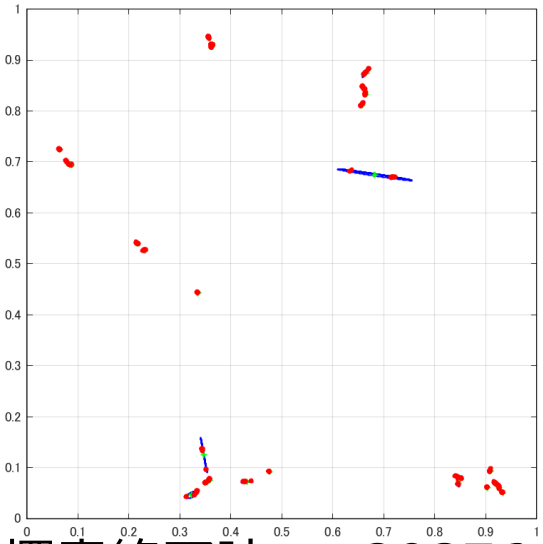

初期生成時 g=0


世代g=4000


世代g=5000


世代g=8000


世代g=14000

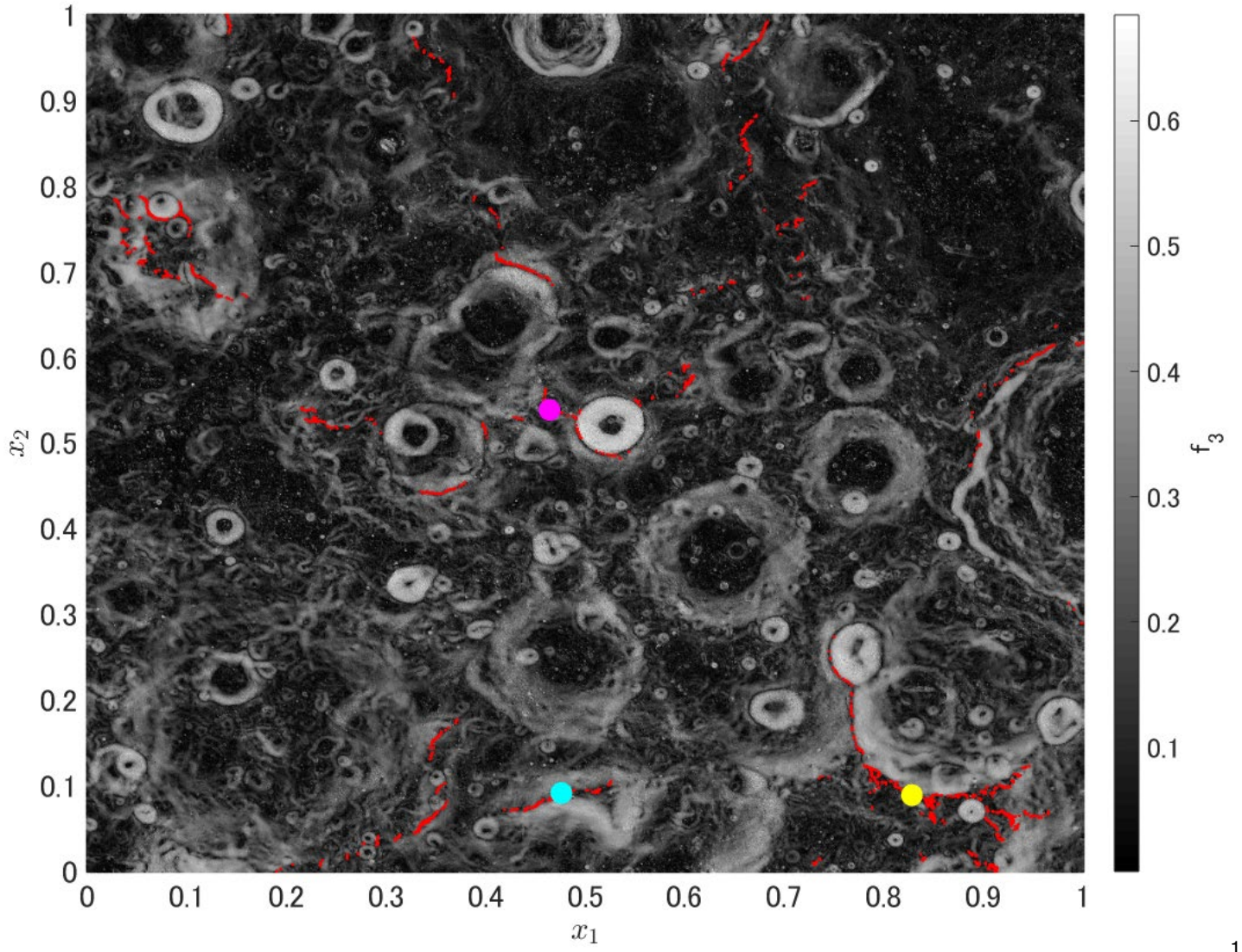

探索終了時 g=20376

月着陸最適候補地の選定問題における既知最良解

- f_1の最小値： 0.01333333
- f_2の最小値：-0.98000002
- f_3の最小値： 0.00047304

赤点はグリッドサーチにより
判明した実行可能解

各目的関数の最小値はmCPIEの
別々の試行により求まった値



参考文献

[Boese 95] K. D. Boese, “Cost Versus Distance In the Traveling Salesman Problem”, UCLA Computer Science Department, 1995.

[池田 02] 池田心, 小林重信, “GAの探索におけるUV現象とUV構造仮説”, 人工知能学会論文誌, vol. 17, no. 3, pp. 239-246, 2002.

[Auger 09] A. Auger, J. Bader, D. Brockhoff, E. Zitzler, “Theory of the Hypervolume Indicator: Optimal μ-Distributions and the Choice Of the Reference Point”, Foundations of Genetic Algorithms (FOGA 2009), 2009.

[戸田 17] 戸田淳, 小野功, “有望領域の非連結性に着目した有望個体囲い込み法による大域的多峰性関数最適化”, 第12回進化計算学会研究会, pp. 89-96, 2017.

m-03

月着陸最適候補地の選定問題に対する最適化アルゴリズム

室蘭工業大学
宮本 将英, 中田 涼介, 渡邉 真也

背景

今回のベンチマーク問題の特性

- 制約条件が厳しい

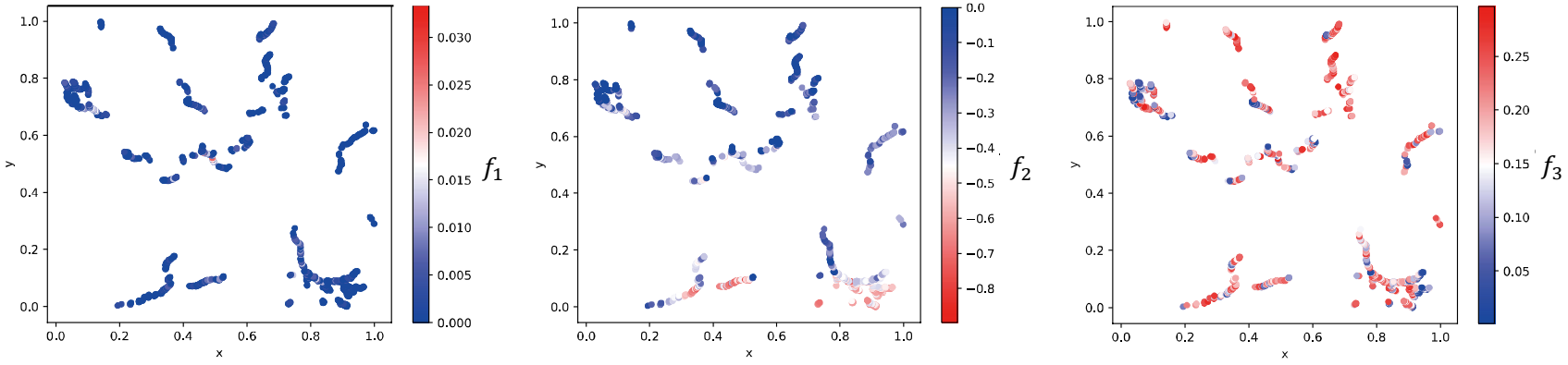
  f_1(連続日陰日数)における制約条件を満たす個体を生成しにくい

  - **重みベクトルによる分割探索を行わない**
  - 制約条件を無視して探索し，探索結果から制約条件を満たす個体を取り出す

- 非常に複雑な各目的関数の構造

  多峰性が非常に強く，実行可能領域が細かく分かれている

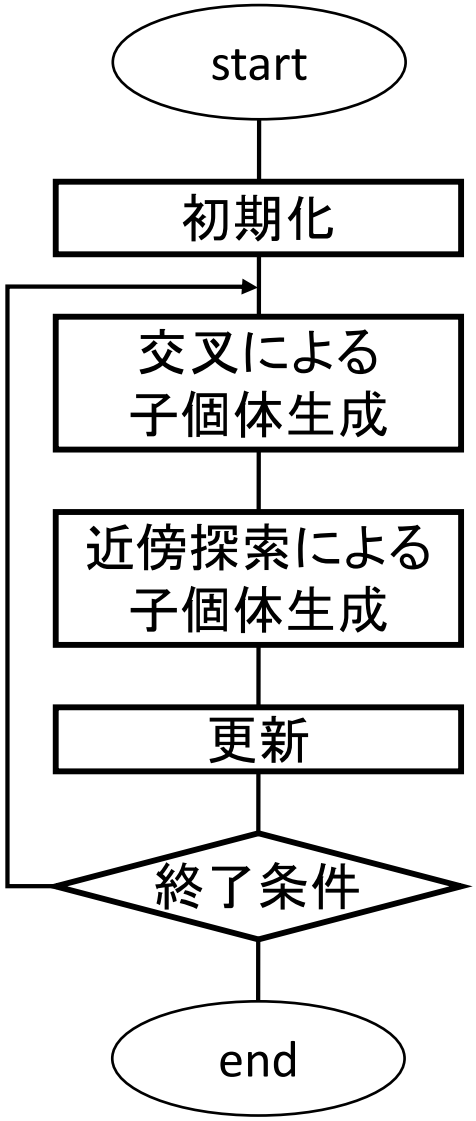
  - 比較的多峰性の弱いf_2(通算通信時間)に着目した初期解生成
  - **初期解生成により有望領域を見つけ、近傍探索を中心とした探索を行う**



アルゴリズム

アルゴリズムのフロー

1. Grid Searchによる初期化
2. 交叉による子個体生成
3. 近傍探索による子個体生成
4. 更新



アルゴリズム

1. Grid Searchによる初期化

分割付きGridSearchにより有望領域を推定し，初期解を生成する

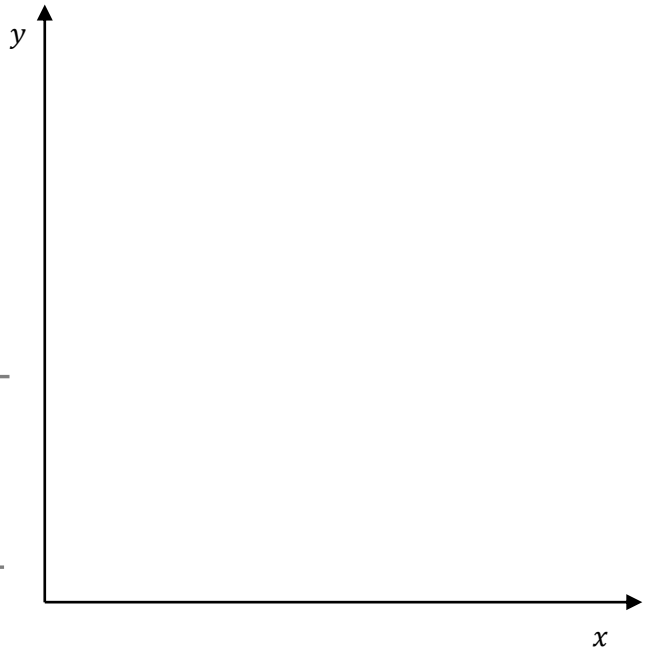
1-1. GridSearchを行う

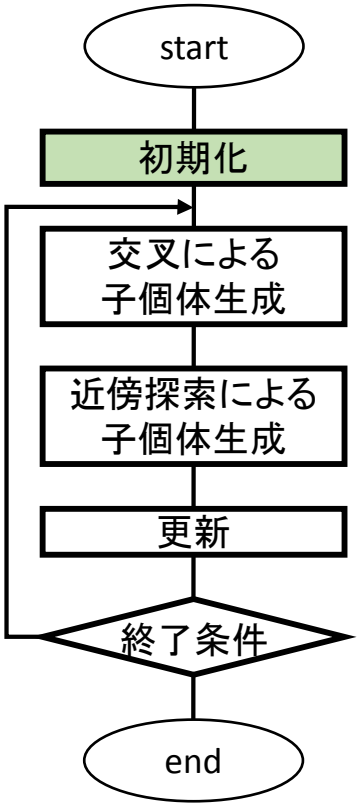
1-2. 4分割する

1-3. 上位2つの領域を選択する

1-4. 1-1から1-3を深さDまで繰り返す

1-5. 生成された全ての解から
非劣解ランクの低い解を取り出す





アルゴリズム

1. Grid Searchによる初期化

分割付きGridSearchにより有望領域を推定し，初期解を生成する

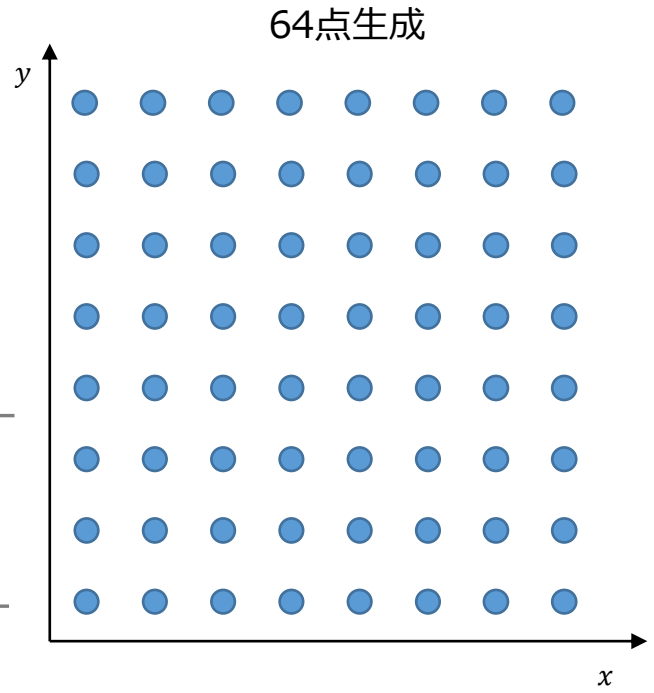
1-1. GridSearchを行う

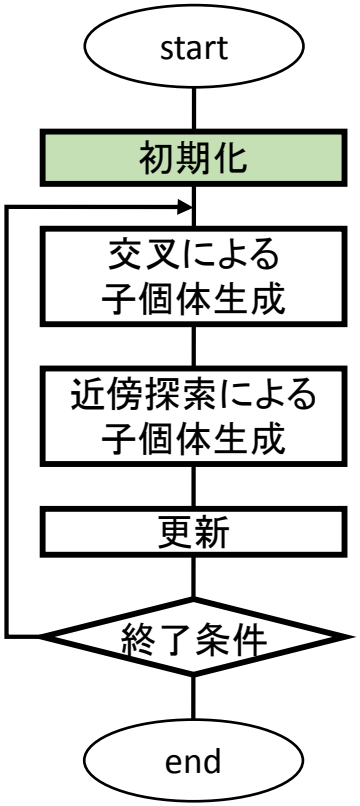
1-2. 4分割する

1-3. 上位2つの領域を選択する

1-4. 1-1から1-3を深さDまで繰り返す

1-5. 生成された全ての解から
非劣解ランクの低い解を取り出す





アルゴリズム

1. Grid Searchによる初期化

分割付きGridSearchにより有望領域を推定し，初期解を生成する

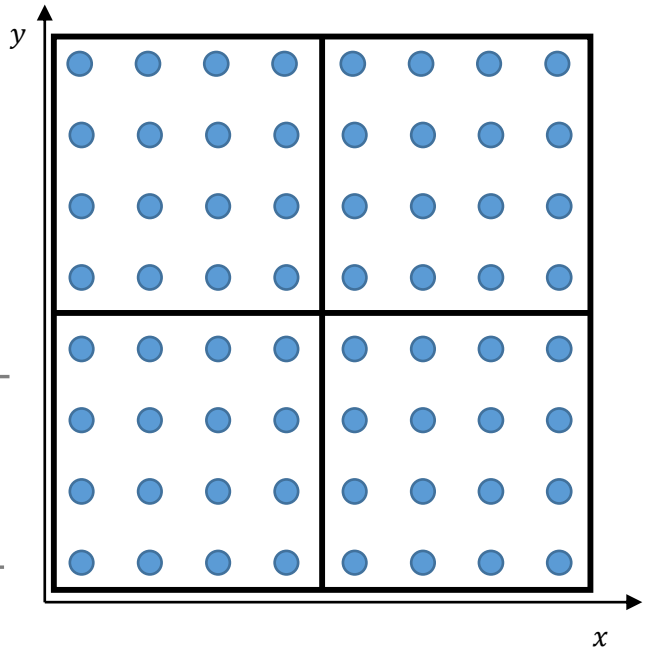
1-1. GridSearchを行う

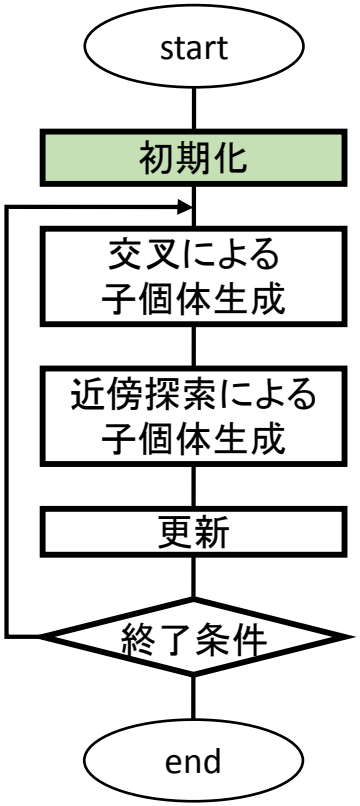
1-2. 4分割する

1-3. 上位2つの領域を選択する

1-4. 1-1から1-3を深さDまで繰り返す

1-5. 生成された全ての解から
非劣解ランクの低い解を取り出す





アルゴリズム

1. Grid Searchによる初期化

分割付きGridSearchにより有望領域を推定し，初期解を生成する

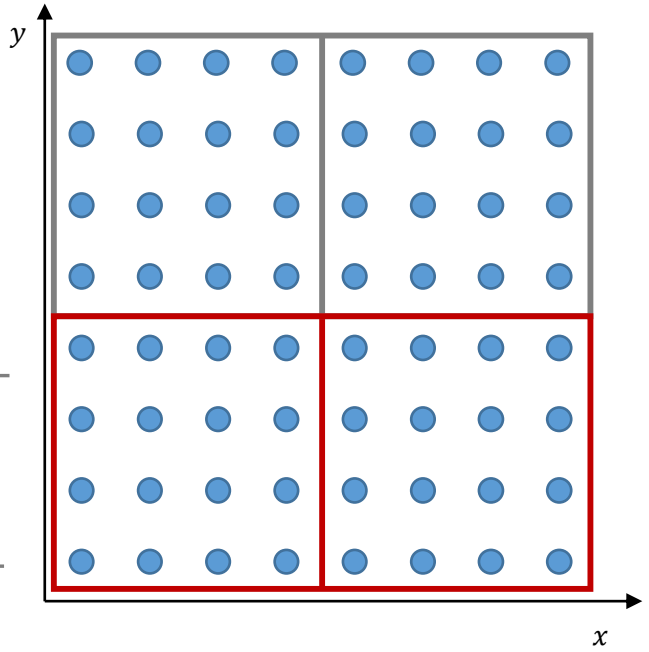
1-1. GridSearchを行う

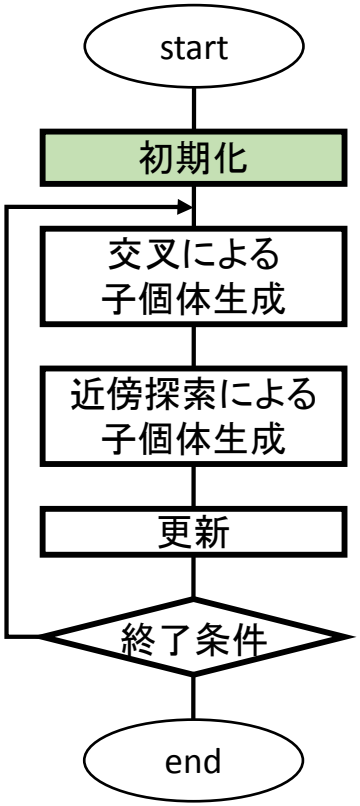
1-2. 4分割する

1-3. 上位2つの領域を選択する

1-4. 1-1から1-3を深さDまで繰り返す

1-5. 生成された全ての解から
非劣解ランクの低い解を取り出す





アルゴリズム

1. Grid Searchによる初期化

分割付きGridSearchにより有望領域を推定し，初期解を生成する

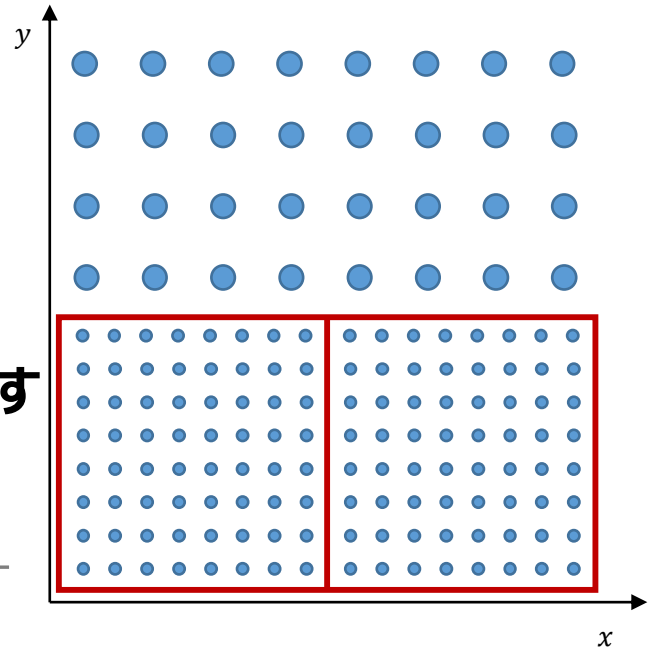
1-1. GridSearchを行う

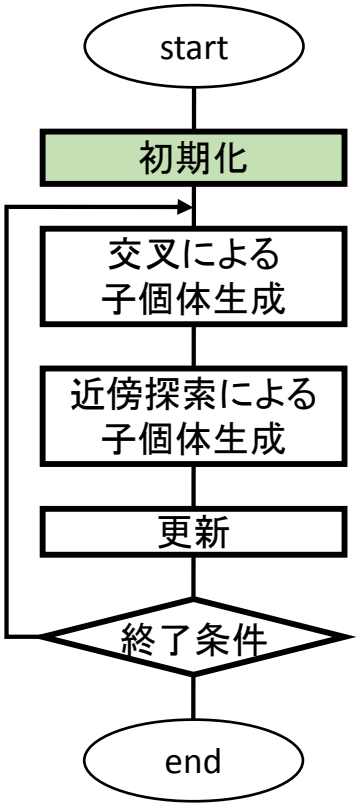
1-2. 4分割する

1-3. 上位2つの領域を選択する

1-4. 1-1から1-3を深さDまで繰り返す

1-5. 生成された全ての解から
非劣解ランクの低い解を取り出す





アルゴリズム

1. Grid Searchによる初期化

分割付きGridSearchにより有望領域を推定し，初期解を生成する

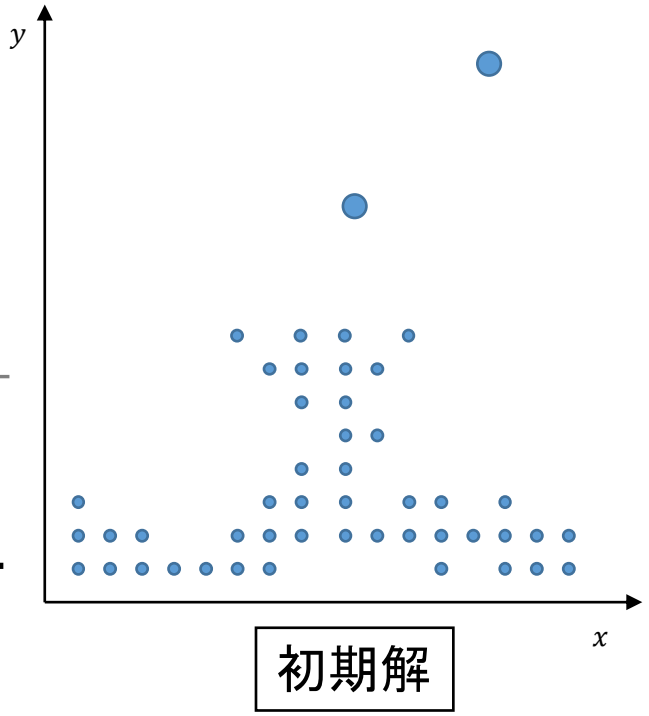
1-1. GridSearchを行う

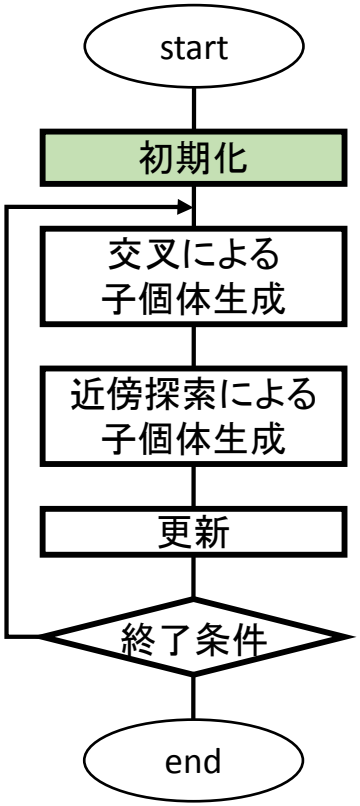
1-2. 4分割する

1-3. 上位2つの領域を選択する

1-4. 1-1から1-3を深さDまで繰り返す

1-5. 生成された全ての解から
非劣解ランクの低い解を取り出す





アルゴリズム

1. Grid Searchによる初期化

分割付きGridSearchにより有望領域を推定し，初期解を生成する

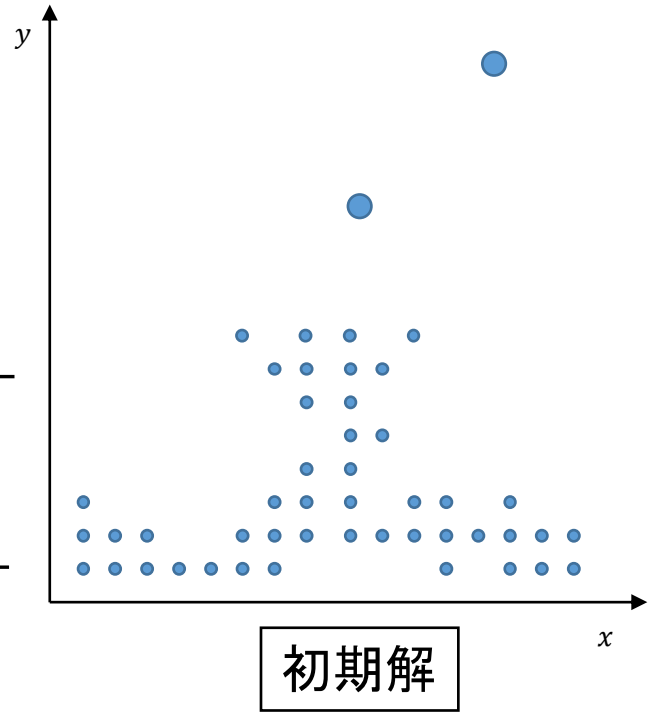
1-1. GridSearchを行う

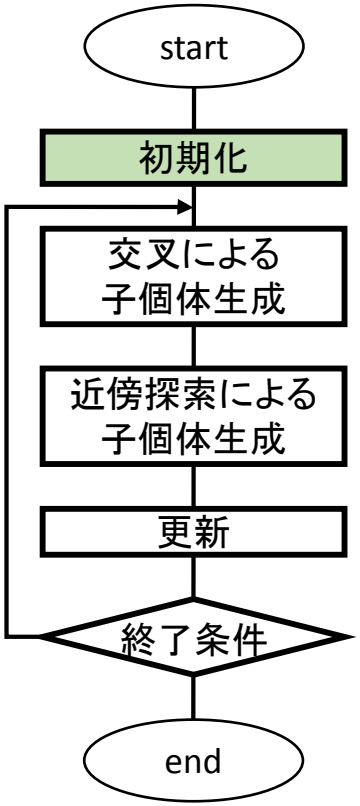
1-2. 4分割する

1-3. 上位2つの領域を選択する

1-4. 1-1から1-3を深さDまで繰り返す

1-5. 生成された全ての解から
非劣解ランクの低い解を取り出す





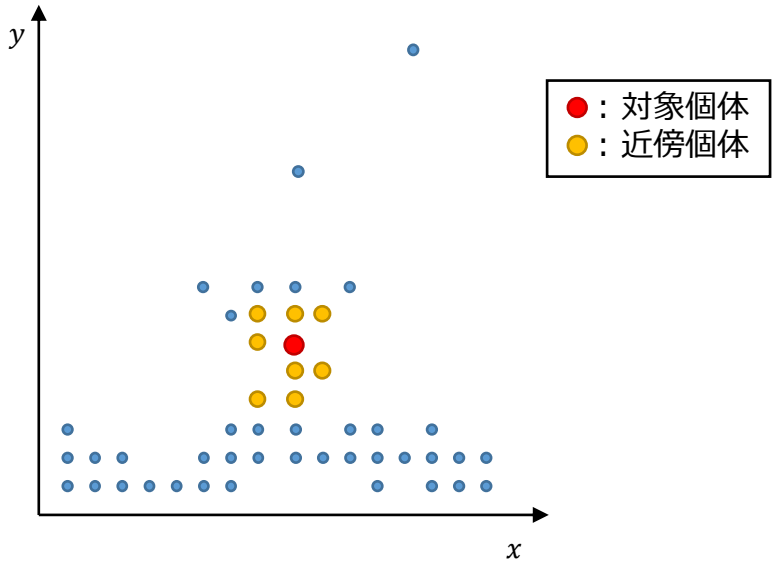
有望領域に分布した非劣解を初期解として用いる

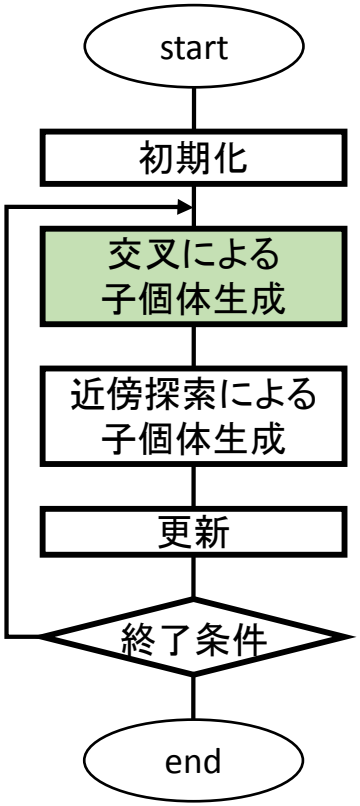
アルゴリズム

2. 交叉による子個体生成

全ての目的関数において強い多峰性がある
重みベクトルを用いた場合，制約条件を満たした個体を生成しにくい

**近傍個体を「設計空間の距離の近い個体」と定義し，
DEによる交叉とPMによる突然変異を行う**





交叉，突然変異によって解の近傍を探索

アルゴリズム

3. 近傍探索による子個体生成

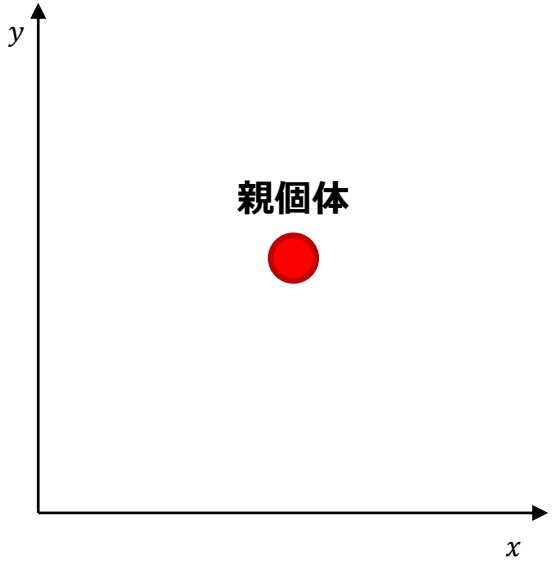
制約条件を満たす個体を生成しにくい
実行可能領域において実行可能解が連続して存在

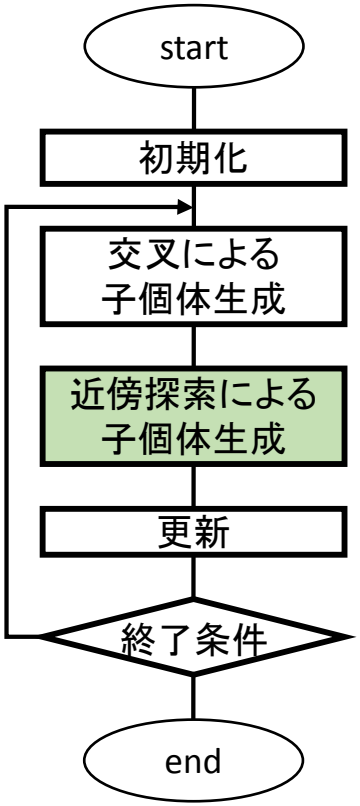
大域的探索である交叉に加え，
局所的探索として各個体の近傍に子個体を4点生成する

3-1. 親個体を中心とする円の半径rを
[0.001, 0.01]からの乱数で決定

3-2. 子個体を生成する角度θを
[0, 90]からの乱数で決定

3-3. 子個体を十字に生成





アルゴリズム

3. 近傍探索による子個体生成

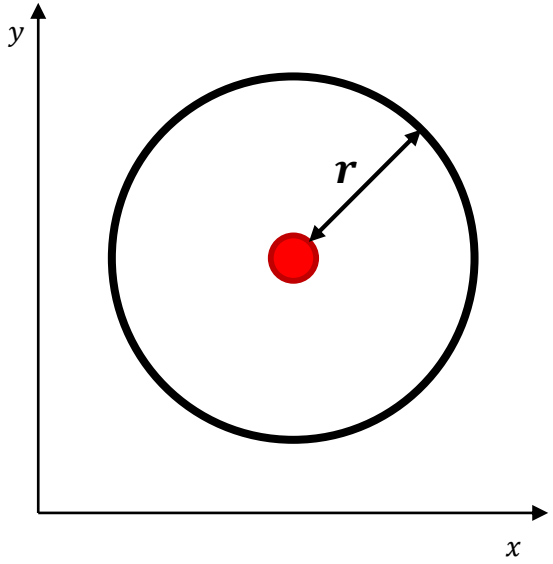
制約条件を満たす個体を生成しにくい
実行可能領域において実行可能解が連続して存在

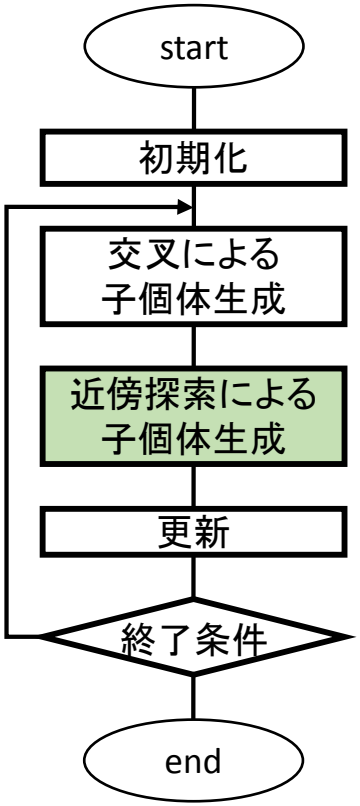
大域的探索である交叉に加え，
局所的探索として各個体の近傍に子個体を4点生成する

3-1. 親個体を中心とする円の半径rを
[0.001, 0.01]からの乱数で決定

3-2. 子個体を生成する角度θを
[0, 90]からの乱数で決定

3-3. 子個体を十字に生成





アルゴリズム

3. 近傍探索による子個体生成

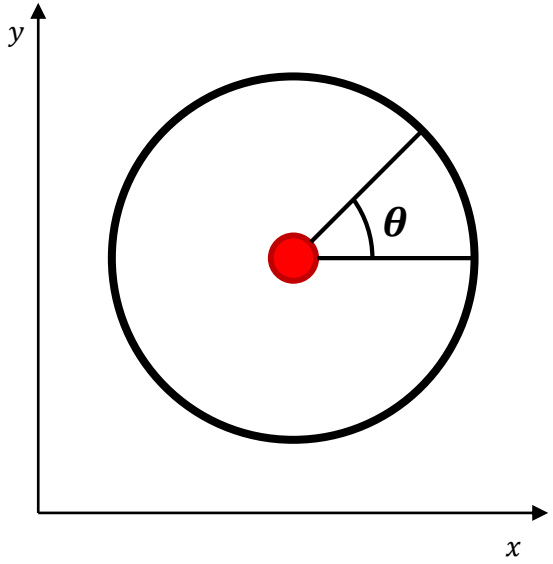
制約条件を満たす個体を生成しにくい
実行可能領域において実行可能解が連続して存在

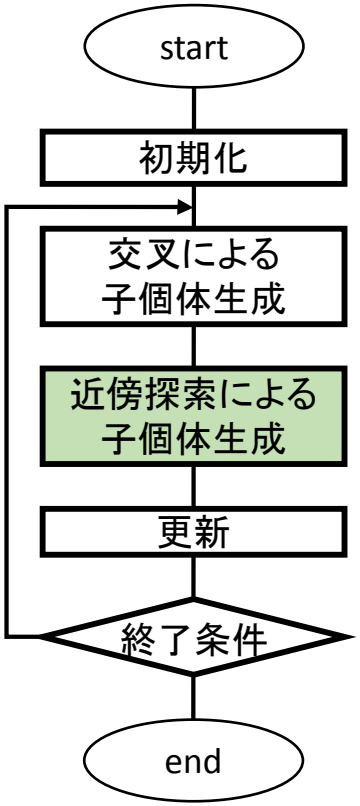
大域的探索である交叉に加え，
局所的探索として各個体の近傍に子個体を4点生成する

3-1. 親個体を中心とする円の半径rを
[0.001, 0.01]からの乱数で決定

3-2. 子個体を生成する角度θを
[0, 90]からの乱数で決定

3-3. 子個体を十字に生成





アルゴリズム

3. 近傍探索による子個体生成

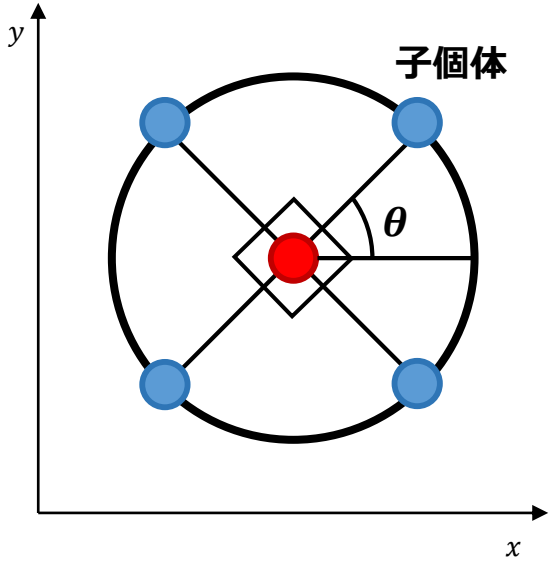
制約条件を満たす個体を生成しにくい
実行可能領域において実行可能解が連続して存在

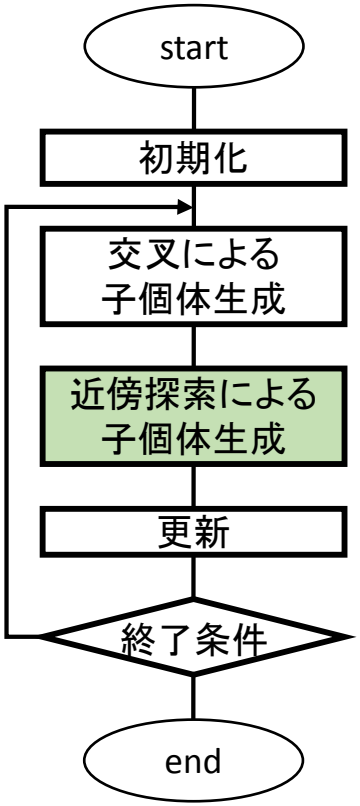
**大域的探索である交叉に加え，
局所的探索として各個体の近傍に子個体を4点生成する**

3-1. 親個体を中心とする円の半径rを
[0.001, 0.01]からの乱数で決定

3-2. 子個体を生成する角度θを
[0, 90]からの乱数で決定

3-3. 子個体を十字に生成





アルゴリズム

3. 近傍探索による子個体生成

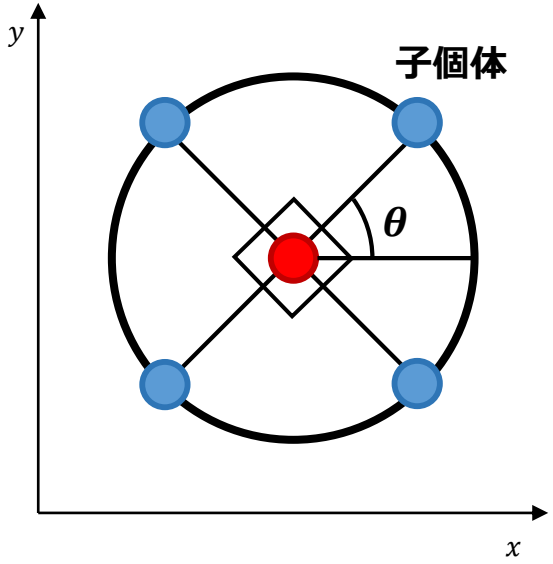
制約条件を満たす個体を生成しにくい
実行可能領域において実行可能解が連続して存在

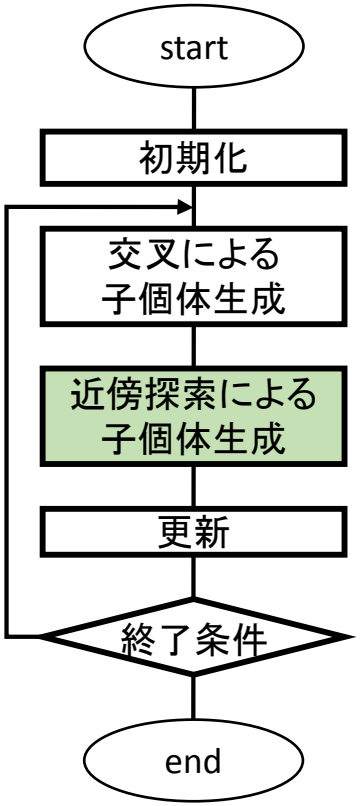
大域的探索である交叉に加え，
局所的探索として各個体の近傍に子個体を4点生成する

3-1. 親個体を中心とする円の半径rを
[0.001, 0.01]からの乱数で決定

3-2. 子個体を生成する角度θを
[0, 90]からの乱数で決定

3-3. 子個体を十字に生成





近傍探索により，制約条件を満たした多様な解を生成

アルゴリズム

4. 更新

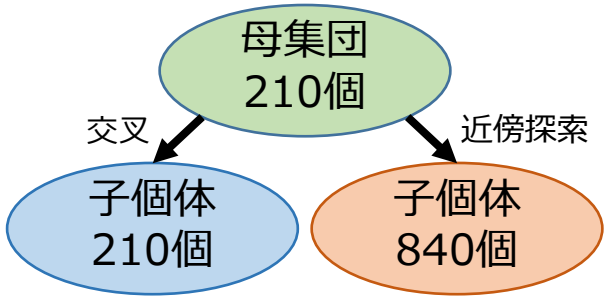
制約条件を満たす個体を生成しにくい
有望領域で探索を行うために多様性を失いやすい

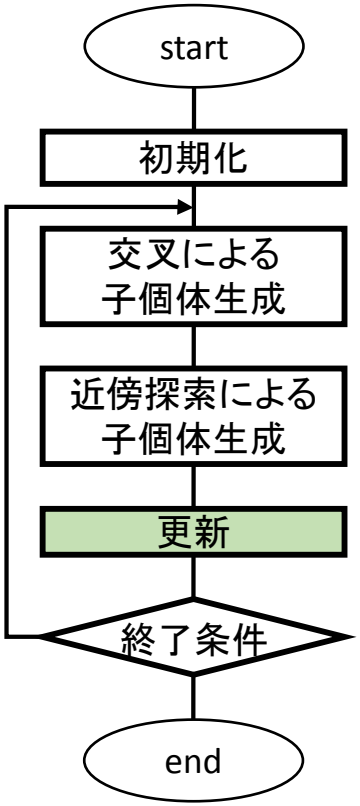
非劣解をベースにした更新を行い，母集団の多様性を維持する

4-1. 母集団と生成された全ての子個体を
一つの解集団とする

4-2. 解集団を非劣解ランク順にソート

4-3. 非劣解ランクの低いものから
210個取り出し，次世代の母集団とする





アルゴリズム

4. 更新

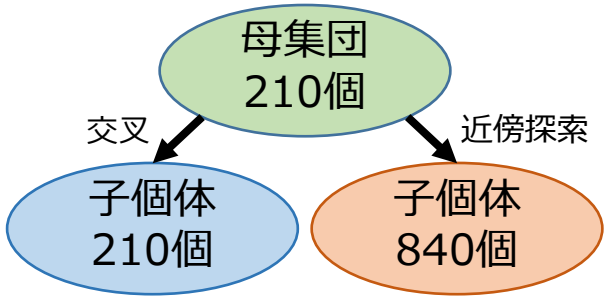
制約条件を満たす個体を生成しにくい
有望領域で探索を行うために多様性を失いやすい

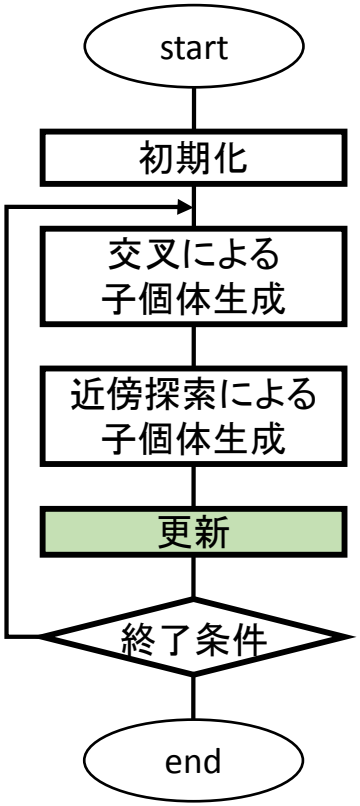
非劣解をベースにした更新を行い，母集団の多様性を維持する

4-1. 母集団と生成された全ての子個体を一つの解集団とする

4-2. 解集団を非劣解ランク順にソート

4-3. 非劣解ランクの低いものから210個取り出し，次世代の母集団とする





アルゴリズム

4. 更新

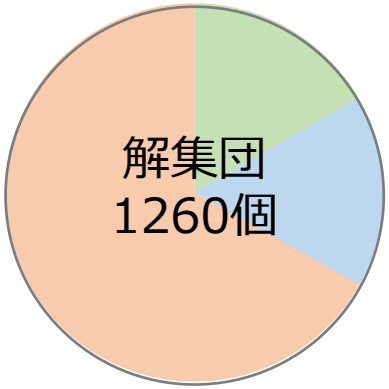
制約条件を満たす個体を生成しにくい
有望領域で探索を行うために多様性を失いやすい

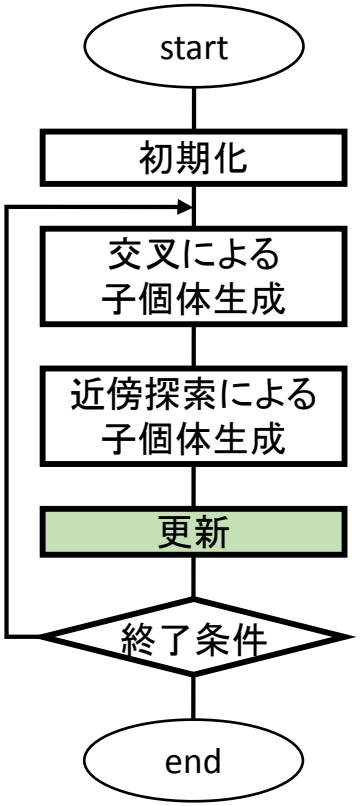
非劣解をベースにした更新を行い，母集団の多様性を維持する

4-1. 母集団と生成された全ての子個体を一つの解集団とする

4-2. 解集団を非劣解ランク順にソート

4-3. 非劣解ランクの低いものから210個取り出し，次世代の母集団とする





アルゴリズム

4. 更新

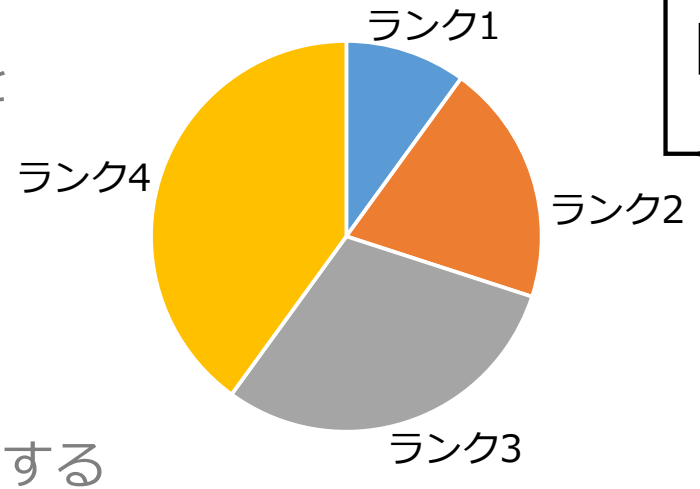
制約条件を満たす個体を生成しにくい
有望領域で探索を行うために多様性を失いやすい

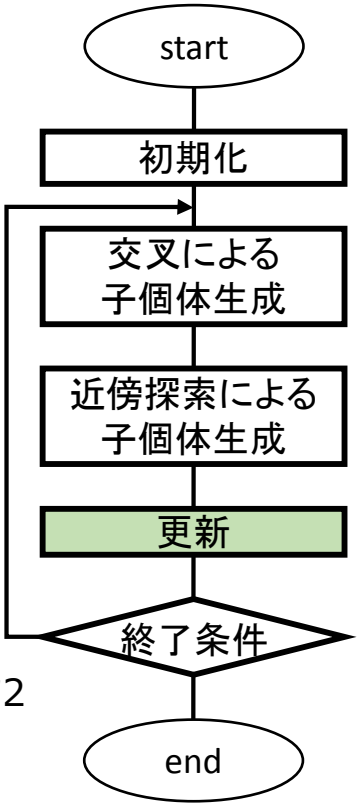
非劣解をベースにした更新を行い，母集団の多様性を維持する

4-1. 母集団と生成された全ての子個体を
一つの解集団とする

4-2. 解集団を非劣解ランク順にソート

4-3. 非劣解ランクの低いものから
210個取り出し，次世代の母集団とする





アルゴリズム

4. 更新

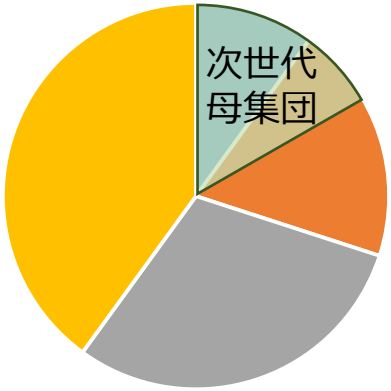
制約条件を満たす個体を生成しにくい
有望領域で探索を行うために多様性を失いやすい

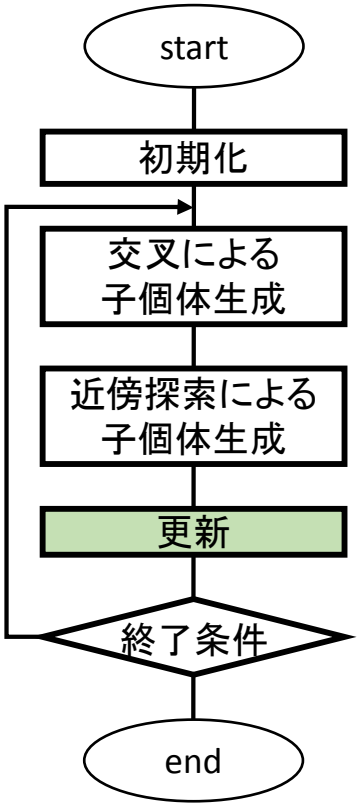
非劣解をベースにした更新を行い，母集団の多様性を維持する

4-1. 母集団と生成された全ての子個体を
一つの解集団とする

4-2. 解集団を非劣解ランク順にソート

**4-3. 非劣解ランクの低いものから
210個取り出し，次世代の母集団とする**





アルゴリズム

4. 更新

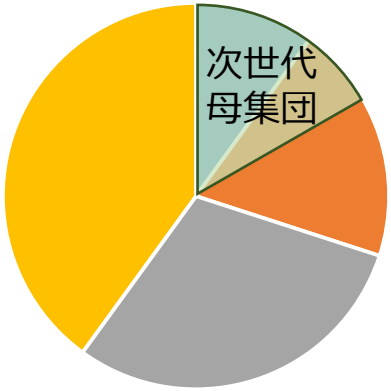
制約条件を満たす個体を生成しにくい
有望領域で探索を行うために多様性を失いやすい

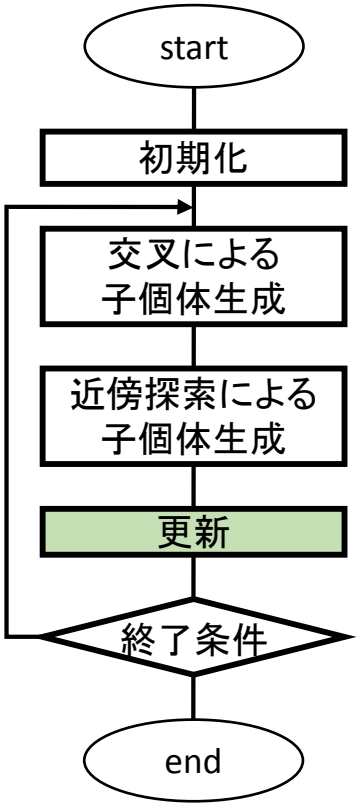
非劣解をベースにした更新を行い，母集団の多様性を維持する

4-1. 母集団と生成された全ての子個体を
一つの解集団とする

4-2. 解集団を非劣解ランク順にソート

4-3. 非劣解ランクの低いものから
210個取り出し，次世代の母集団とする





常に多様性を維持しつつ，近傍主体の探索を行う

パラメータ・結果

パラメータ

母集団サイズ	210
近傍サイズ	20
交叉方法	DE(Differential Evolution)
交叉率(DE)	1.0
スケールパラメータ(DE)	0.5
突然変異手法	PM(Polynomial Mutation)
突然変異率	0.1
分布指数(PM)	20
次元分割数(GridSearch)	8
分割深度(GridSearch)	4
近傍生成数	4

結果

パラメータ・結果

パラメータ

母集団サイズ	210
近傍サイズ	20
交叉方法	DE(Differential Evolution)
交叉率(DE)	1.0
スケールパラメータ(DE)	0.5
突然変異手法	PM(Polynomial Mutation)
突然変異率	0.1
分布指数(PM)	20
次元分割数(GridSearch)	8
分割深度(GridSearch)	4
近傍生成数	4

結果

HVの中央値：0.889648　(非劣解数：18個)
HVの最良値：0.934281

コンペティション アルゴリズム解説

立命館大学
知能エンターテインメント研究室
磯林　知志、大伴　周也、原田　智広、Ruck Thawonmas

提案手法

問題の特徴

- 本問題では、扱う個体の変数が座標(x, y)の2値であることから、一般的な交叉の効果が小さいと考察
- 既存研究[1]より、探索空間内の一部に実行可能解が密集

提案手法

- NSGA-IIに局所探索手法Variable neighborhood search (VNS)を導入したアルゴリズムを提案
- VNSの効率を高めるために初期個体を探索範囲に均等に生成

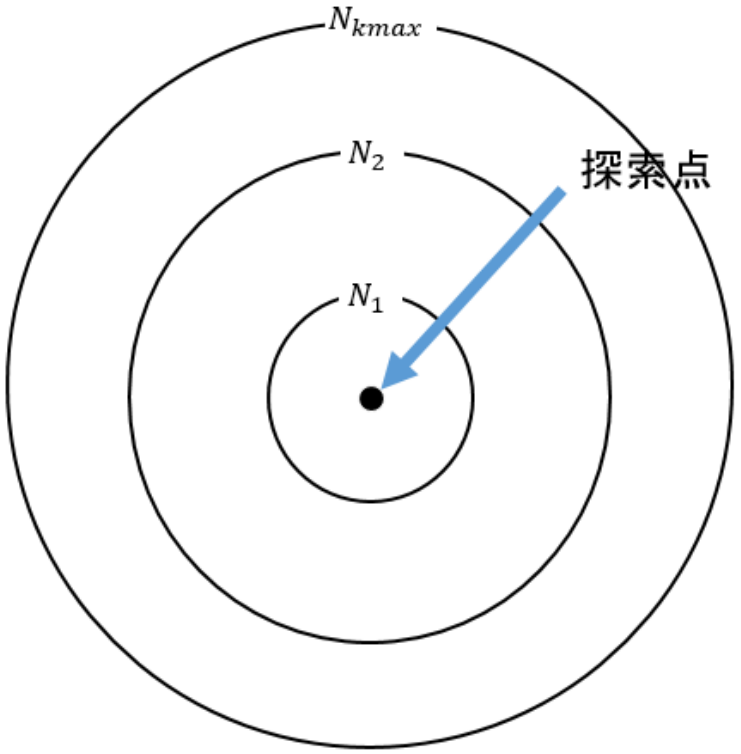
[1]西山万里, 大嶽久志, 星野健, 橋本樹明, 渡辺毅, 立川智章, 大山聖. 月周回衛星「かぐや」のデータを用いた多目的最適化による月着陸最適候補地の選定. 宇宙科学情報解析論文誌, Vol. 5, pp. 51-57, 2015.

Variable Neighborhood Search (VNS)

- VNSは探索領域の異なる複数の近傍関数N_kを利用し、局所探索の領域を徐々に拡大しながら探索
  例）$N_1 < N_2 < \cdots < N_{k_{max}}$
- 探索領域の小さい近傍関数から探索を開始
- 探索点が更新されない場合
  →探索範囲が大きい近傍関数を利用
- 探索点が更新される場合
  →探索範囲の小さい近傍関数を利用

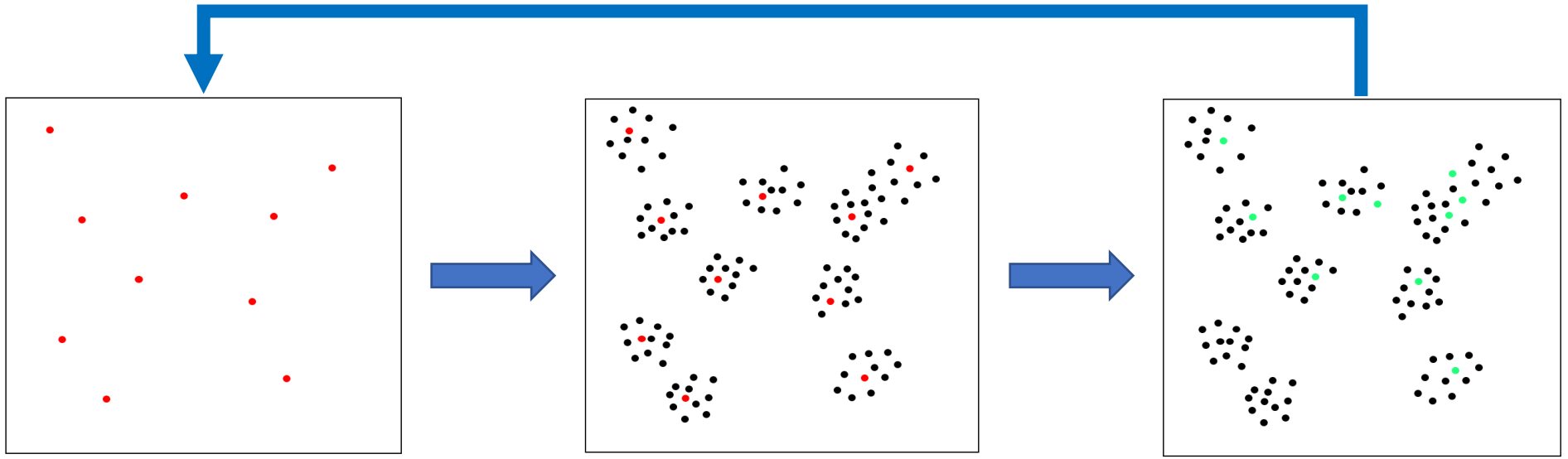
- 近傍関数は探索点を平均値、標準偏差を$0.001 + 0.002 \times k$とする正規乱数で定義

VNS概念図



提案手法の狙い

- VNSによる局所探索と非支配ソートにより、世代数が進むにつれ優良解付近の探索を重点的に行える



探索点を選択

VNSにより探索点の周りに新規解を複数生成

非支配ソートにより次の探索点候補を決定

探索の流れ

パラメータ: $s = 50, n = 100, \sigma = 0.001, \Delta\sigma = 0.002, kmax = 5, M = 50$

1. 初期個体集団($s \times s$)個を探索領域から均等に生成
2. 初期個体集団を非支配ソートし、上位n個体を選択、個体集団($x_1 \sim x_n$)を生成
3. 各個体x_iと標準偏差σの正規乱数でx'_iを生成
4. x'_iがx_iを支配していればx_iを更新、σを初期値へ更新
5. x_iがk_{max}回数更新されなければσに$\Delta\sigma$を加算
6. M世代分3~5を繰り返し、その中で生成された全個体を非支配ソートし上位n個体を選択
7. 規定評価回数まで3~6を繰り返す

Evolutionary Computation Competition 2018

Yuri Lavinas[1] and Claus Aranha[2]

[1]Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba

December 4, 2018

1 Introduction

The Evolutionary Computation Competition 2018 address the problem of selecting the optimal landing site for the lunar landing exploration mission.

1.1 Moon Landing Problem

For the multi-objective objective optimization, the variables longitude and latitude should minimize 3 objectives. The first is the number of shade days, the second is the communication time and the third is the inclination angle. In reality, we want to consider the maximum communication time. for consistence, the value for the communication time is negated, leading to a mimization problem.

Also, this optimization considers two constraints, the number of days of continuous shade and the inclination angle of the landing point. The number of days of continuous shade has to be lower than 0.05 while the inclination angle of the landing point has to be lower than 0.3. The input data is a set of latitude and longitude normalized between 0 and 1.

2 Method

MOEA/D-DE variant of MOEA/D framework, was chosen to address this problem. uses DE mutation operator instead of SBX crossover operator to produce offspring solution.

In this study, we are interested in addressing the moon lading problem by using the Multi-objective Evolutionary Algorithm based on Decomposition framework, MOEA/D [1].MOEA/D represents a class of population-based metaheuristics for solving Multi Objective Problems. In this framework, each individual has a specific weight vector which is used to decompose the original multi-objective problem into simpler, single-objective subproblems by means of scalarizations. Each subproblem is then evaluated and its utility value is calculated by an aggregation function given the related weight vector.

The work in this paper is based on MOEA/D-DE [2]. A slight modified version of the MOEA/D-DE was implemented using the code available in the R package MOEADr [3].

Here, the configuration and parameters used in this MOEA/D-DE variation are introduced.

The decomposition method used was Simplex-Lattice Design (SLD), with the number of weights as 14, leading to a population size of size $N = 120$.

The scalar aggregation function used was the adjusted Weighted Tchebycheff (AWT).

The neighborhood assignment function is based on the distances between weight vectors in the space of objectives, with the neighborhood size $T = 20$, and the neighborhood search probability $\delta_p = 0.9$.

The reproduction procedure consists of applying Differential Mutation, followed by a polynomial mutation, and then truncation. After truncating, a local search is performed followed by truncation. The three individuals participating in the Differential mutation are chosen at random, and the Φ parameter is randomly chosen between 0 and 1 (independently sampled for each operation). The Polynomial mutation operator values are $\eta_m 20$ and $p_m = 0.445$ [4]. The local search three-point quadratic approximation with $gamma = 0.95$.

The replacement strategy used was the restricted neighborhood replacement, with size $nr = 2$.

Constraint handling method used was to use a penalty function with the penalization constant $\beta = 0.95$.

Following many recent works, as the one from Cai et. al [5], we used an external archive which is updated using nondominated sorting and crowding distance principle of NSGA-II [6]. The size of the population on this external archive is the same as the working population, 120.

The termination criteria used was number of evaluations, being set to 30000.

References

[1] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[2] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE Transactions on evolutionary computation*, vol. 13, no. 2, pp. 284–302, 2009.

[3] F. Campelo, L. Batista, and C. Aranha, "the moeadr package - a component-based framework for multiobjective evolutionary algorithms based on decomposition," 2017.

[4] F. Campelo, L. S. Batista, and C. Aranha, "The moeadr package-a component-based framework for multiobjective evolutionary algorithms based on decomposition," *arXiv preprint arXiv:1807.06731*, 2018.

[5] X. Cai, Y. Li, Z. Fan, and Q. Zhang, "An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 508–523, 2015.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.