進捗報告

1 今週やったこと

DEAP 周りの実装を追ってました．特に GA のオペレーション関連を重点的に．今週はその紹介です．[1]

2 初期化

2.1 initRepeat

deap.tools.initRepeat(container, func, n) 予め決められた関数を n 回回して初期個体を生成する

2.2 initIterate

deap.tools.initIterate(container, generator) generator から初期個体を生成する．

2.3 initCycle

deap.tools.initCycle(container, seq_func, n=1) 関数のリストを受取り，それを繰り返すように初期個体を生成する．

3 交差

3.1 cxOnePoint

deap.tools.cxOnePoint(ind1, ind2) 普通の 1 点交差．切る場所はランダムで決まる．

3.2 cxTwoPoint

deap.tools.cxTwoPoint(ind1, ind2) 2 点交差．真ん中を入れ替える．

3.3 cxUniform

deap.tools.cxUniform(ind1, ind2, indpb) 一様交差．n-1 点交差に同じ．ランダムに各インデックスの値を入れ替える．

3.4 cxPartialyMatched

deap.tools.cxPartialyMatched(ind1, ind2) 部分一致交差法．同じ個体間で数字を入れ替えて解を構成するので数字に重複を作らない．

3.5 cxUniformPartialyMatched

deap.tools.cxUniformPartialyMatched(ind1, ind2, indpb) 部分一致交差法と一様交差を合わせたもの．

3.6 cxOrdered

deap.tools.cxOrdered(ind1, ind2) 順序交差．片方の親からは一部をそのまま引き継ぎ，残りの部分については相対的な順番は保存しつつ受け継ぐ方法．[2] がわかりやすい．

3.7 cxBlend

deap.tools.cxBlend(ind1, ind2, alpha) 2 つの親子体のベクトルから子供の発生しうる座標の最大値と最小値を決める．その際にどこまで空間を広げるかを alpha で指定する．

3.8 cxSimulatedBinary

deap.tools.cxSimulatedBinary(ind1, ind2, eta) SimulatedBinaryCrossover(SBC)．バイナリで遺伝子を持ってる場合にできる賢い 1 点交差？多項式確率分布に従って 2 この親から 2 この子を生成するらしいが，調べてもよくわからなかった．[3] にアルゴリズムがのってます．

3.9 cxSimulatedBinaryBounded

deap.tools.cxSimulatedBinaryBounded(ind1, ind2, eta, low, up) SBC の境界付きのもの．

3.10 cxMessyOnePoint

deap.tools.cxMessyOnePoint(ind1, ind2) 1 点交差だが，それぞれ別のところで切ってつなげるため，遺伝子の長さが変わる．

4 選択

4.1 selRandom

deap.tools.selRandom(individuals, k) ランダムに次の個体を選ぶ．内部のサブモジュール的な感じで使われる．

4.2 selBest

deap.tools.selBest(individuals, k, fit_attr='fitness') 適応度最良の個体を上から選ぶ．内部のサブモジュール的な感じで使われる．

4.3 selWorst

deap.tools.selWorst(individuals, k, fit_attr='fitness') 適応度最悪の個体を上から選ぶ．内部のサブモジュール的な感じで使われる．

4.4 selTournament

deap.tools.selTournament(individuals, k, tournsize, fit_attr='fitness') 普通のトーナメント方式．個体群から一定数の個体をランダムにとり，その中で最も適応度の高いものを次世代に残す．

4.5 selRoulette

deap.tools.selRoulette(individuals, k, fit_attr='fitness') ルーレット選択．適応度に比例する確率で次世代の個体をサンプリングする．

4.6 selDoubleTournament

deap.tools.selDoubleTournament(individuals, k, fitness_size, parsimony_size, fitness_first, GP で主に用いられる，個体のサイズを考慮したトーナメント方式．普通のトーナメント選択に加えてサイズに基づく別のサンプリング関数を用いたトーナメントをする．

4.7 selStochasticUniversalSampling

deap.tools.selStochasticUniversalSampling(individuals, k, fit_attr='fitness') 確率的ユニバーサルサンプリング．適応度の総和の平均を距離として等間隔に点を配置し，個体の適応度を足していきながら各点をサンプリングするかを決定する．

4.8 selLexicase

deap.tools.selLexicase(individuals, k) ランダムな訓練ケースを生成して，そのデータに対する誤差の集合を各個体が持ち，その中で誤差の最も小さい個体を残すアルゴリズム．許容量を ϵ とした ϵ-Lexicase Selection という拡張版も．詳しくは [4] を参照．

4.9 selEpsilonLexicase

deap.tools.selEpsilonLexicase(individuals, k, epsilon) 上で書いた Laxicase の拡張版．許容量 ϵ で選ばれた個体の中からランダムで次世代の個体に加える．

4.10 selAutomaticEpsilonLexicase

deap.tools.selAutomaticEpsilonLexicase(individuals, k) 上の EpsilonLexicase の変数 ϵ を自動で決める．具体的には ϵ を絶対偏差の中央値 (Median absolute deviation:MAD) で置き換える．

5 突然変異

5.1 mutGaussian

deap.tools.mutGaussian(individual, mu, sigma, indpb) 一定確率でガウス分布からサンプリングした値を各遺伝子に足す．

5.2 mutShuffleIndexes

deap.tools.mutShuffleIndexes(individual, indpb) 一定確率で他の遺伝子と位置を入れ替える．

5.3 mutFlipBit

deap.tools.mutFlipBit(individual, indpb) 一定確率で各ビットを反転する．

5.4 mutUniformInt

deap.tools.mutUniformInt(individual, low, up, indpb) 一定確率で一様分布 [low, up] からサンプリングした値に置き換わる．

5.5 mutPolynomialBounded

deap.tools.mutPolynomialBounded(individual, eta, low, up, indpb) 多目的最適化アルゴリズムを解くために考案されたアルゴリズム．NSGA-II をベースにしている．詳細は [5] に．

6 来週の予定

未定．相談しつつ GA 実装 or 4 コマ．

参考文献

[1] deap-documentation. https://deap.readthedocs.io/en/master/api/tools.html#operators.

[2] Applying GA to ordering problems. http://ono-t.d.dooo.jp/GA/GA-order.html#OX.

[3] Real-Coded Genetic Algorithms. https://engineering.purdue.edu/~sudhoff/ee630/Lecture04.pdf.

[4] T. Helmuth, L. Spector, and J. Matheson. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643, 2015.

[5] NSGA-II. https://qiita.com/Taichi-Furukawa/items/a7a0982cc20a401133c0.