情報生体工学特論　レポート　2720460191　中西宏介

複数同時車種設計の 2 目的最適化のコーディング課題の報告書

① 開発環境：Anaconda

https://www.anaconda.com/products/individual　よりダウンロードし、パッケージをインストールする．私は D ドライブ直下に[anaconda]というファイル名でインストールしました．

② 仮想環境の作成

Python のバージョン、パッケージなどをまとめる仮想環境を作成する．
Anaconda Prompt を実行し、[cd D:/anaconda/envs]と入力しディレクトリを移動したのち、「conda create -n 環境名」と入力し実行すると、入力した環境名のフォルダが作成され、それが仮想環境となる．その後、必要とするライブラリなどを仮想環境にインストールし、アクティベートさせると、その仮想環境のライブラリや設定を使用することができる．

③ 仮想環境をカーネルとして選択できるようにする

②で生成した仮想環境を jupyter notebook で使用するために以下のコードを Anaconda Prompt で仮想環境フォルダまでディレクトリを移動する。そして[activate 仮想環境名]で仮想環境をアクティベートします．その後、以下のコードを anaconda prompyt 上で実行します．[ipython kernel install --user --name=先ほど生成した仮想環境名]

④ 多目的最適化ライブラリ「platypus」のインストール

今回は,多目的最適化ライブラリである[platypus]を使用するため、先ほど生成した仮想環境にインストールする．まずは anaconda prompt を立ち上げ、仮想環境のフォルダまでディレクトリを移動します（ex. [cd D:/anaconda3/envs/環境名]）．　次にその仮想環境を使用するために、[activate 仮想環境名]で仮想環境をアクティベートします．続いて platypus をインストールするために[conda install platypus-opt]と入力し実行することで、仮想環境のフォルダの中にインストールされます（仮想環境名 -> Lib -> site-packages -> platypus）．今回は platypus のファイル名を platypus2 と変更する．

⑤ Algorithm.py の置き換え

④でインストールし、ファイル名を変更した platypus2 の中の algorithm.py を、別途添付した algorithm.py に置き換える．

⑥ 複数同時車種設計のデータファイルをダウンロード

https://ladse.eng.isas.jaxa.jp/benchmark/jpn/index.html　のサイトの「ダウンロード」という欄から、「ここ」をクリックし、データファイルを仮想環境の直下にダウンロードする.

⑦　Mazda_mop.ipynb と run.bat を仮想環境ファイルに置く

別途に添付した Mazda_mop.ipynb と run.bat を仮想環境のファイル内に置く

⑧ Jupyter notebook を起動

Anaconda Prompt を実行し、仮想環境のフォルダまでチェンジディレクトリし、仮想環境をアクティベートしたのち、[jupyter notebook]と入力し、実行することで、ブラウザ上で jupyter notebook という開発環境が起動する. すると仮想環境フォルダが開かれた状態となっているため、Mazda_mop.ipynb をクリックし開く.

⑨ プログラムの説明

Platypus のファイル名はここでは[platypus2]に変更しています.

Def belegundu：最適化問題を定義

（1）　platypus で個体を設定した数生成

（2）　1 個体情報を pop_vars_eval.txt に記入

（3）　Run.bat を実行し、f1（総重量）と f2（共通部品数）と制約 54 条件の値を返す

（4）　Objs_f[0] = f1, objs_f[1] = f2, cons = [制約 54 条件の値]

　共通部品数は最大化が目的なので,objs_f[1] = 74 – objs_f[1]とし、最小化問題として定義して、objs_f[1]が小さくなる場合は共通部品数が増えるように定義した.

（5）　制約 54 条件の内、負の値である制約違反の数を penalty に代入

（6）　Objs_f[0],objs_f[1],penalty を返す　return [objs_f[0], objs_f[1]], [penalty]

Pop_size=個体数（300~1000）

gen_size =世代数（100~300）

problem = Problem(222,2,1) : [一個体の設計変数，目的関数の数，制約の数]

problem.types[:] =Real[0:3]　：各設計変数は 0~3 の実数をとる

problem.directions[:] =[Problem.MINIMIZE,Problem.MINIMIZE]で最小化問題と定義

problem.constraints[:] = “<=0” で制約の値が 0 以下の時に解として出力される

problem.function = belegundu　で最適化問題を problem.py に投げる

Jupyter notebook の Kernel ボタンを押し、Restart & Run all ボタンを押すと、複数同時車種設計の問題を platypus 内の algorithm.py の MOEAD という進化型多目的最適化を用いて最適化を始めます. 実行すると、毎世代の最適化が終わり次第、横軸が 3 車種の総重量、縦軸が 74 – 共通部品数を表すグラフを仮想環境フォルダ -> data -> mazda_pop に保存します. 設定した世代数だけ個体集団の更新が終了するとプログラムは終了しますが、プログラムを強制終了するには Run ボタンの横にある黒い■ボタンを押します.

Jupyter notebook は、セルごとにコードを実行させることができます。Mazda_mop.ipynb では 2 つのセルにコードを分けており、一つ目のセルではプログラムの実行と、各世代の個体集団のグラフの保存がされます。2 つ目のセルでは、最後に更新された世代の個体集団の全個体の設計変数、目的関数の値、制約違反した数を mazda_mop.csv ファイルに保存する形となっており、例えば 1 個体目の情報は、A1~A222 に設計変数、A223 に 3 車総重量、A224 に共通部品数、A225 に制約違反数が保存されています.