Laboratorium: Python i wizualizacja

Wojciech Jaśkowski

termin: +1 tydzień

1 Motywacja

Istotnym elementem sprawnej komunikacji człowiek-komputer jest wizualizacja danych. Python i narzędzia zbudowane wokół dostarczają wygodnej platformy do obróbki, **interaktywnej** analizy i wizualizacji danych.

2 Zadanie

Dokonaj wizualizacji danych pochodzących z eksperymentów ewolucyjnych. Napisz w Pythonie program, który:

1. Wczytaj dane z plików. Zawierają one wyniki działania pięciu różnych algorytmów ewolucyjnych[1]. Każdy algorytm został uruchomiony 32 razy. Wyniki (im więcej, tym lepiej) dla kolejnych uruchomień algorytmu znajdują się w kolumnach. Każdy wiersz reprezentuje "czas", tj. numer pokolenia w algorytmie (max 200) lub, inaczej patrząc, liczba rozegranych gier (max ok. 500.000). Jedno pokolenie to ileś rozwiązań, dlatego możemy przyjąć dwie różne skale osi OX.

2. Wygeneruje wykresy wizualizujące (**patrz strona 3**):

   (a) **Na ocenę 3.0:** przebiegi w funkcji 'czasu' algorytmów. Oś OX to liczba rozegranych gier, tj. wiersz w tablicy. Oś OY to średnia skuteczność algorytmu – uśrednione po 32 uruchomieniach, czyli liczymy średnią dla wiersza (OX) po 32 kolumnach.

   (b) **Na ocenę 5.0:** Wyniki końcowe algorytmów, czyli statystyki dla wyników uzyskanych w ostatniej generacji. Jako, że zadanie na 3.0 jest podzadaniem zadania na 5.0, nie trzeba robić tego pierwszego, gdy robi się to drugie).

   Oczekiwane wyniki wizualizacji (w zależności od poziomu aspiracji) przedstawione są poniżej. Do wizualizacji użyj biblioteki matplotlib.

2.1 Uwagi

- **Ważne:** Wygenerowane wykresy powinny być (w granicach rozsądku) dokładnym odwzorowaniem tych pokazanych w tym pliku na stronie 3 (pokazanie umiejętności ustawienia kolorów, skali, siatki na wykresie, itp). Dodatkowo w wersji na 5.0 stosujemy Latexową czcionkę lub Times New Roman.

[1] Algorytmy te szukały strategii gry dla uogólnionego problemu iterowanego dylematu więźnia. Były to warianty algorytmów ewolucyjnych i koewolucyjnych.

- Graficzne wyjaśnienie co widzimy na wykresie pudełkowym: http://en.wikipedia.org/wiki/Interquartile_range. Czerwona kreska oznacza medianę, a kropka — wartość średnią.

2.2 Od czego zacząć?

Listing 1: Some simple plot

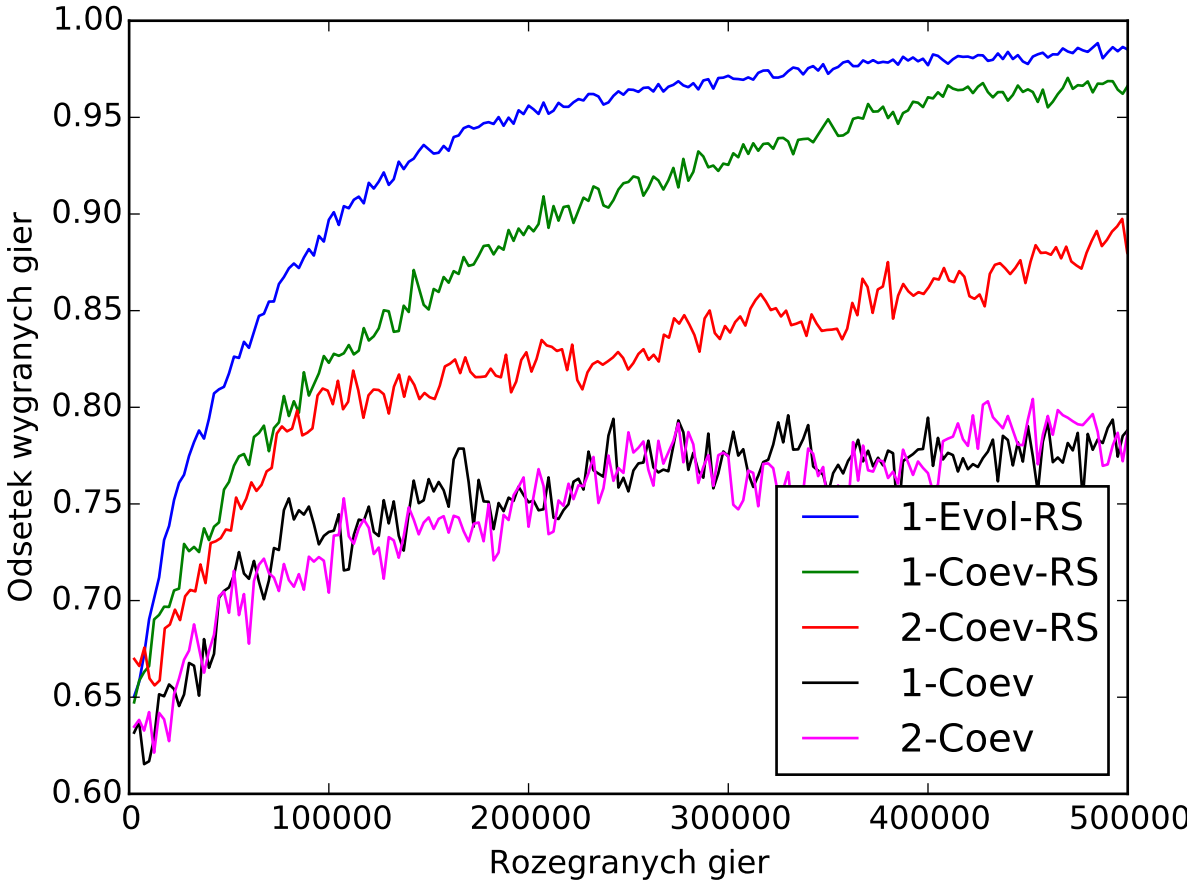
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt

def main():
    plt.figure(figsize=(3, 3))

    plt.plot([100,200,300,400],[0.1,0.2,0.8,0.9])
    plt.savefig('myplot.pdf')
    plt.close()

if __name__ == '__main__':
    main()
```

Sugerowane jest jednak zrobienie zadania w notebooku z włączoną flaga *%matplotlib inline*. Przykładowy plik znajduje się tutaj.

2.3 Podpowiedzi

- Matplotlib.pyplot:
  http://matplotlib.org/api/pyplot_summary.html
- Szerokość rysunku to ok. 480 punktów, czyli ok. 6.7 cala.
- Wykres liniowy:
  http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot
- Formatowanie osi:
  http://matplotlib.org/examples/pylab_examples/custom_ticker1.html
- Wykres pudełkowy:
  http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.boxplot
- Dwa wykresy na jednym obrazku:
  http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.subplot
- Dodanie kropek najłatwiej osiągnąć za pomocą nałożenia wykresu punktowego:
  http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.scatter
- Markery co N punktów:
  http://matplotlib.org/api/artist_api.html#matplotlib.lines.Line2D.set_markevery

2.4 Wizualizacja na 3.0



2.5 Wizualizacja na 5.0 (proszę zwrócić uwagę na czcionkę)