Rafał Mironko, 331405
Zadanie: zoptymalizuj włączanie i wyłączanie pieca, maksymalizując funkcję.

Naprawdę spodobała mi się implementacja algorytmu genetycznego. Może i algorytm ma 55 lat, ale jego implementacja i badanie było jak cofnięcie się w czasie i odrzucenie wszystkich aktualnych standardów, oraz odkrycie, że zalecenia ludzi takich jak John Holland co do parametrów startowych miały wtedy sporo sensu.

Dlaczego o tym mówię? Ponieważ, kiedy napisałem algorytm i chciałem uruchomić go z wysoką mutacją i niskim krzyżowaniem, ten był praktycznie bezużyteczny.

Założenia uruchamiania algorytmu:
- losowa populacja startowa
- 25 osobników
- 1000 iteracji

Postanowiłem zacząć od krzyżowania ustawionego na 0.1 i mutacji na 0.6. Byłem bardzo zaskoczony, kiedy zobaczyłem następujący wykres. Każdy punkt jest jedną z 1000 iteracji.

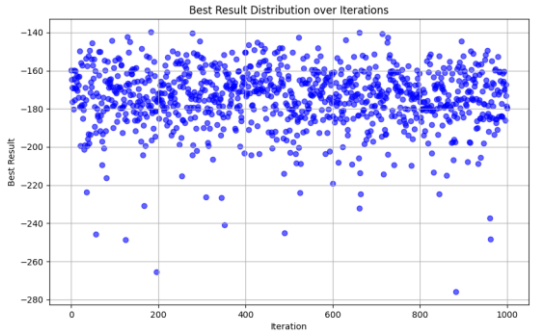


Fig. 1 Siła mutacji - 0.6, krzyżowanie - 0.1

Punkty wyglądają, jakby znalazły się tam losowo, do tego nawet nie zbliżyły się do zalecanego wyniku -135.

Postanowiłem przyjrzeć się założeniom i konfiguracji, jakie zalecali twórcy oryginalnego algorytmu genetycznego. Okazało się, że mutacja 0.001 i krzyżowanie 0.9 zapewniały znacznie lepsze wyniki:

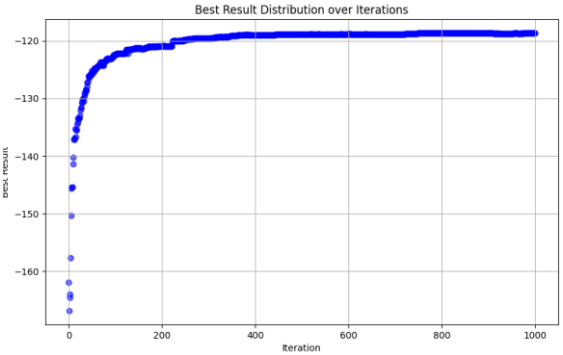


Fig. 2 Siła mutacji - 0.001, Krzyżowanie - 0.9

Ten wynik wygląda, jakby po prostu wszedł do jakiegoś optimum lokalnego – które jednak daje nam całkiem satysfakcjonujące wyniki – mimo że algorytm nie potrafi z niego uciec. Zobaczmy, co się stanie, kiedy uruchomi go dla tych danych 25 razy.

Algorytm wykonuje się na jednym wątku, więc nawet dla dość mocnego Ryzena 5 5600X zajęło to dość sporo czasu (125 sekund). Uznałem jednak, że nie będę go refaktoryzował na wykorzystanie wszystkich 12 wątków – nie ma to sensu na potrzeby pracy domowej, szczególnie, że musiałbym najpierw się tego nauczyć.

Po 125 sekundach obliczeń widzimy, że wszystkie wyniki znalazły się gdzieś w okolicach tego samego optimum lokalnego:

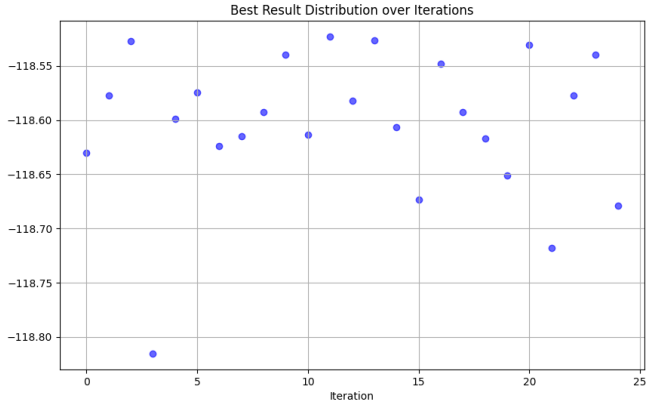


Fig. 3 Powtórzenie 0.001;0.9 25 razy

Wygląda na to, że są dwie opcje: funkcja ma optimum globalne w okolicach 118.5, albo parametry są źle ustawione. Oczywiście od razu przyjmuję tę drugą opcję i zaczynam testy. Mam wrażenie, że rozmiar populacji nie daje zbyt wiele. Podobnie z ilością iteracji, a przynajmniej w momencie, gdy wpadamy do jakiegoś optimum i mamy niewielką mutację.

Udało mi się za to odnotować widoczne na wykresie przeskoki pomiędzy optimami lokalnymi, gdy ustwiłem mutację na 0.001 i krzyżowanie na 0.1. Przeskok widoczny w iteracji 2 i 825:

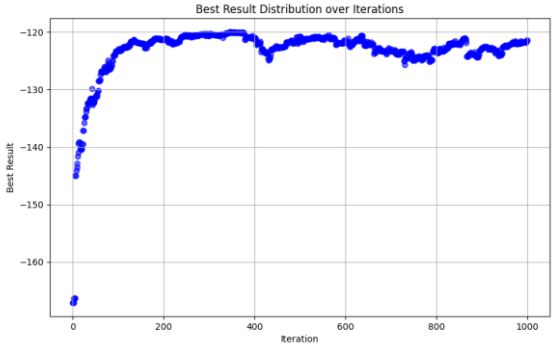


Fig. 4 0.001;0.1

Hipoteza: twórcy algorytmu mieli rację, ustawiając niewielką siłę mutacji. Mylili się jednak co do wartości krzyżowania i lepiej jest ustawić niższą, zgodnie z współczesnymi badaniami.

W związku z tym, zbadam wpływ siły mutacji na wychodzenie z optimów lokalnych i potencjalne wyszukiwanie optimów globalnych.

Parametry początkowe:
- losowa populacja startowa
- 25 osobników
- krzyżowanie 0.1
- 1000 iteracji, czasem sprawdzę też 10000, kiedy osiągnę obiecujący wynik. Będzie to jednak tylko alternatywa dla powtórzenia testu 25 razy.

Wróćmy do naszego wykresu. Widać tam, że przeskok pomiędzy optimami jest możliwy. Z wcześniejszego badania funkcji wiemy, że badanie tego obszernego optimum, do którego często trafiamy, skutkuje zbliżaniem się do -118.5. Uznajmy to za wynik, którego przebicie będzie sukcesem badań tej funkcji. Jeśli go nie przekroczymy, trzeba będzie zastanowić się , czy jednak utknięcie w optimum lokalnym może czasem mieć więcej sensu.

Test 1: uruchomienie algorytmu z siłą mutacji 0.001 25 razy:

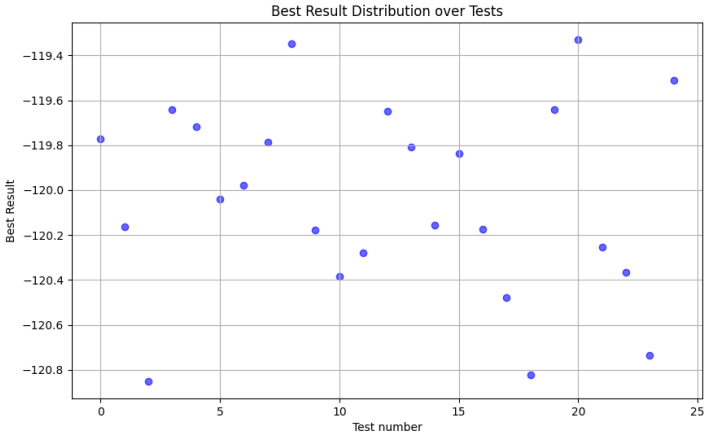


Fig. 5 0.001;0.1 25 razy

Jak widać, mimo możliwości przeskoków pomiędzy optimami, cały czas trafiamy do okolic optimum -118.5. Algorytm najwyraźniej jest w stanie na chwilę je opuścić, ale odbija się i ciągle mniej więcej tam wraca. Hipoteza: siła mutacji może być zbyt niewielka. Podwajam siłę mutacji.

Dla siły mutacji 0.002 łatwo zaobserwować, że występuje więcej chwilowych wyjść z optimum. Ciężko jednak stwierdzić, czy są to różne optima, czy ciągle jesteśmy w tym samym:

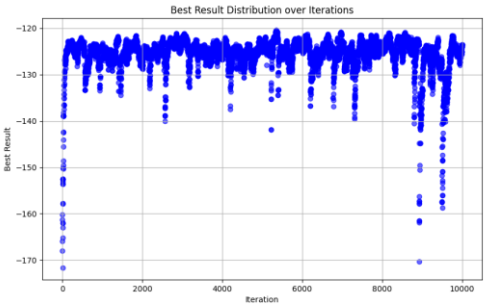


Fig. 6 0.002

Uruchomię algorytm 25 razy po 1000 iteracji i zobaczymy, co się stanie.

Fig. 7 0.002 25 razy

Jak można było przewidzieć, mutacja najwyraźniej jest zbyt duża lub zbyt ciężko jest wydostać się z optimum. Przetestujmy jeszcze większą mutację – 0.003.

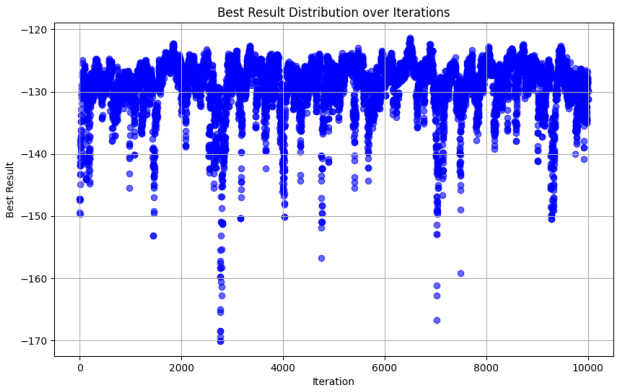


Fig. 8 0.003 10000 iteracji

Tak wysoka mutacja jest bez sensu. Nawet jeśli wydostaniemy się z optimum, algorytm nie będzie miał czasu ustabilizować się w nowym, bo od razu będzie przeskakiwał dalej. Nawet, jeśli znajdziemy nowe, lepsze optimum, nie dowiemy się o tym – bo algorytm przejdzie dalej.

Sprawdzę mutację 0.0015.

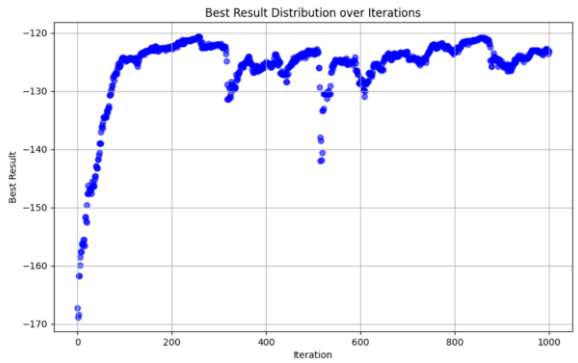
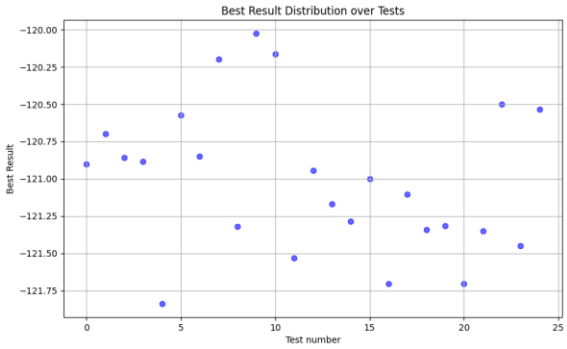


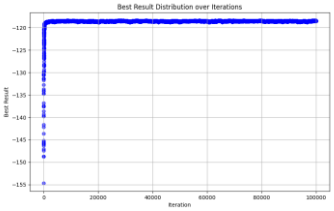
Fig. 9 0.0015 1000 iteracji

Wygląda naprawdę obiecująco. Mamy wciąż dużo wyjść z optimów, a algorytm na szansę ustabilizować się w nowym.

Kiedy jednak sprawdziłem go dla 10000 iteracji lub 25 wywołań, okazuje się, że po prostu nie możemy opuścić tego jednego optimum.



Przed wyciągnięciem wniosków postanowiłem wyznaczyć wartość tego optimum lokalnego, do którego zbliżaliśmy się na początku, uruchamiając go z ilością 100 000 iteracji. Czysta ciekawość, czy trafi powyżej -118.5.



Jak widać, tak – osiągnąłem -118.4325. Ciekawe. A oto znaleziona przez niego tablica:

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 1 0 1
 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0 1 1
 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Algorytm oszczędza energię na początku i na końcu, postanawia na początku mocno nagrzać w piecu, a potem mniej więcej utrzymuje tę samą temperaturę, używając powtarzalnego patternu. Pewnie, gdyby zależało mi na faktycznym rozwiązaniu tego problemu, przestałbym badać początkowe i końcowe zera, prawdopodobnie też początkowe jedynki – i skupiłbym się na

środkowych stanach. To jednak nie miałoby zbyt wiele wspólnego z ogólną implementacją algorytmów genetycznych, tylko z czymś bardzo szczególnym.

WNIOSKI

- Ten algorytm nie nadaje się do szukania optimum globalnego zadanej funkcji. Nic dziwnego, że powstały jego różne implementacje i poprawy – szczególnie obiecująca byłaby według mnie strategia ES(mu + lambda).

- Twórcy algorytmu genetycznego mieli rację, dobierając niską siłę mutacji i wysokie krzyżowanie. To jest optymalne wykorzystanie tego algorytmu. Jeśli chcemy mieć lepsze wyniki, zamiast zmieniać konfigurację, musimy zmienić algorytm.

- Algorytm genetyczny potrafi dobrze eksploatować minimum lokalne, ale nie radzi sobie najlepiej z wychodzeniem poza nie. Albo po prostu nie jest w stanie tego zrobić, albo uda mu się to chwilowo, jednak wtedy siła mutacji jest tak wielka, że nie da rady wystarczająco dobrze eksploatować nowego minimum – czyli traci swoją największą zaletę. Po prostu jest to algorytm eksploatacyjny, nie eksploracyjny. Do eksploatacji zostały jednak wymyślone lepsze algorytmy, więc genetyczny powinien po prostu trafić na zasłużoną półkę w muzeum.