

**Wydział Elektrotechniki Automatyki Informatyki i Inżynierii Biomedycznej**

***Algorytm Ewolucyjny w zastosowaniu w optymalizacji na szeregach czasowych.***

Autorzy: Dawid Bugajny

*Wojciech Woszczek*

*Igor Ratajczyk*

Kierunek studiów: Automatyka i Robotyka

Przedmiot: *Badania Operacyjne 2*

Kraków, 2021/2022

**Spis treści**

**Ogólne wytyczne:**

**Tytuł rozdziału: Calibri Light 14, Bold**

**Tekst: Calibri tekst podstawowy 12**

**Ctrl+shift+enter na nową stronę**

1. **Zarys Problematyki**

Na podstawie historycznych danych zachorowalności na COVID-19 można stwierdzić znaczną sezonowość tego procesu. Aby skutecznie walczyć z zarazą należy skutecznie pobierać wymazy osób podejrzanych o chorobę, aby móc skutecznie przeciwdziałać rozwojowi epidemii. Jako że żadna służba zdrowia na świecie nie jest przygotowana na tego typu wydarzenia, ale sytuacja wymaga znacznie większej liczby laborantów pobierających wymazy. Praca i szkolenie laborantów kosztują.

Problem optymalizacyjny polega na takim sterowaniu liczbą przyjmowanych laborantów, aby zminimalizować koszty szkolenia i pracy personelu medycznego oraz koszty śmierci nieprzebadanego pacjenta.

Dzięki ubiegło sezonowym danym i modelowi SARIMA będziemy prognozować przyszłą liczbę dziennych zachorowań .

Zakładamy, że każdy laborant pobierający wymazy jest w stanie pobrać wymazów dziennie. Szkolenie nowego laboranta trwa jeden dzień i kosztuje a dzień jego pracy jest warty . Pacjent chory, który nie zostanie przebadany kosztuje państwo .

Z racji ograniczonej pojemności sal i mając na celu wydajność szkolenia w danym dniu nie można szkolić więcej niż laborantów.

Zakładamy, że jako dyrekcja zakładu interesuje nas ograniczone okno czasowe 1 rok

1. **Model Matematyczny Problemu.**

2.0. Przyjęte oznaczenia.

– operator różnicowy

– stała do schematu kosztu oczekujących pacjentów

predykcja dziennej liczby zachorowań (liczba naturalna)

– backshift operator – operator opóźnienia

górne ograniczenie dziennej liczby szkolonych laborantów

– funkcja Heavyside’a

– operator sumy kumulowanej

– warunek początkowy liczby laborantów

– liczba wymazów, jaką dziennie pobiera laborant

– liczba dni, po których niewymazany pacjent nie zostanie wymazany

– szereg czasowy pacjentów, którzy nie otrzymali wymazu

Koszt szkolenia jednego laboranta

2.1. Funkcja celu.

Implementacja algorytmu pozwala na optymalizację problemów, gdzie koszt może być liczony na dwa sposoby.

Funkcja celu (Time Series Covid Problem Naive):

Można to interpretować jako sumę kosztów po wszystkich dniach, gdzie uwzględniamy koszt pracy wszystkich laborantów, oczekiwany koszt życia niewymazanych pacjentów oraz koszt szkolenia nowych laborantów.

Ten model funkcji kosztu jest znacznie prostszy w zrozumieniu, stanowi sensowny sposób badania zachowania algorytmu, gdzie wymagana jest znajomość rozwiązania oraz może być traktowana jako punkt wyjściowy do zrozumienia bardziej zaawansowanej funkcji kosztu z punktu drugiego.

Funkcja celu (Time Series Covid Problem):

gdzie:

takie, że:

Ogólny zamysł drugiej funkcji kosztu implementuje scenariusz, gdzie pacjent niewymazany może zostać przebadany jeszcze przez dni od pierwszego dnia zjawienia się w przychodni. Jeżeli liczba pacjentów przekracza możliwości badawcze zakładu ustawia się on w kolejce na oczekujących a kolejka oczekujących „opóźnia się” o jeden dzień. Jeżeli liczba pacjentów nie przekracza możliwości przetwórczych zakładu to wymazywani są pacjenci oczekujący, ale najpierw wymazywani są pacjenci oczekujący najdłużej ( odpowiada za zmniejszanie liczby oczekujących pacjentów).

Koszt naliczany jest względem oczekującej kolejki pacjentów: każdy kolejny dzień oczekiwania kosztuje wykładniczo (domyślnie: mniej) niż poprzedni, co odzwierciedla rzeczywistość, gdzie pacjent zaraz po zakażeniu czuje się na tyle dobrze, że jest w stanie dalej rozprzestrzeniać wirusa.

2.2. Ograniczenia oraz postać rozwiązania

Rozwiązanie jest postaci:

Gdzie każdemu dniu roku jest przypisywana liczba.

Ograniczenia są postaci:

Oznacza to, że:

* Liczba laborantów na początku roku jest zadania
* Przyrost laborantów jest ograniczony
* Liczba laborantów jest nieujemna

1. **Algorytm Ewolucyjny.**

Pseudokod algorytmu:

Time: = 0

Population = new Population ()

Fitness = FitnessFunction ()

While (not Stop):

Do

T++

Return result

1. **Opis elementów opracowanych.**

Należy zwrócić szczególną uwagę, że z racji wielowymiarowości problemu, klasyczne podejście nakazuje położenie szczególnego nacisku na zachowywanie dopuszczalności rozwiązania. Tak też zostało poczynione, dlatego zaprojektowane operatory genetyczne, pseudogenetyczne oraz inne elementy rozwiązania skupiają się na zachowywaniu dopuszczalności rozwiązania. Dokładniejsze omówienie zastosowanych elementów znajduje się odpowiadających im podrozdziałach.

4.0. Kodowanie rozwiązania.

Genotyp jest kodowany jako wektor liczb całkowitych, liczby zatrudnionych laborantów każdego dnia:

Fenotyp jest kodowany jako wektor liczb całkowitych, liczby laborantów na każdy dzień:

4.1. Konstrukcja rozwiązań początkowych.

Domyślna metoda konstrukcji rozwiązań początkowych obejmuje konstruowanie rozwiązania dopuszczalnego. Można ją zapisać jako:

Co oznacza, że rozwiązania są konstruowane w przód, próbkowane z rozkładu jednostajnego w dopuszczalnym zakresie (tj. tak aby przyrost laborantów był ograniczony liczba laborantów była nieujemna).

Jest możliwość konstrukcji rozwiązania dopuszczalnego z wykorzystaniem innych rozkładów prawdopodobieństwa.

Konstrukcja rozwiązania niedopuszczalnego obejmuje próbkowanie z rozkładu nieograniczonego.

4.2. Operatory Krzyżowania.

Zasadniczym operatorem krzyżowania jest operator kombinacji wypukłej z parametrem dany wzorem:

Wypada rozważyć, czy zachodzi:

(4.2.3)

tj. czy z dopuszczalności rozwiązań rodzicielskich wynika dopuszczalność rozwiązań potomnych.

Z uwagi na liniowe ograniczenia genotypu i liniowe ograniczenia liniowych przekształceń genotypu (suma kumulowana i operator różnicowy) ograniczenia 2 i 3 są również zachowane.

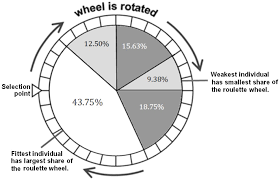
4.3. Operatory Mutacji.

Zaimplementowane operatory mutacji obejmują:

4.4. Selekcja.

Domyślną selekcją w naszym programie jest selekcja ruletkowa. W każdej iteracji algorytmu losowana jest określona liczba osobników ze zbioru otrzymanego po ew. krzyżówce oraz mutacji. Prawdopodobieństwo wybrania osobnika jest proporcjonalne do jego przystosowania (wyliczonego według zadanej funkcji celu) i wyraża się wzorem:

Zakładając rozkład jednostajny, sytuacja ta jest analogiczna do kręcenia tarczą, gdzie pole każdej opcji jest proporcjonalne do jego wartości przystosowania:



Zaimplementowaliśmy również selekcję turniejową. Polega ona na przeprowadzeniu paru „turniejów” pośród losowo wybranych osobników z populacji. Zwycięzca każdego turnieju jest wybierany do następnej populacji (w kolejnej iteracji). Metoda turniejowa może być opisana przez poniższy pseudokod:

Dopóki nie osiągniemy pożądanej liczności populacji:

Wybierz losowo k osobników z poprzedniej populacji,

Najlepszy (funkcja kosztu) osobnik jest wybrany z prawd. p

Drugi najlepszy osobnik jest wybrany z prawdopodobieństwem p\*(1-p)

Drugi najlepszy osobnik jest wybrany z prawdopodobieństwem p\*((1-p)^2)

itd.

4.5. Operator rzutowania na rozwiązania dopuszczalne.

Jako, że operatory mutacji nie zawsze zachowują dopuszczalność rozwiązania opracowany został operator rzutowania na rozwiązania dopuszczalne. Jest on postaci:

Podobnie jak operator konstrukcji rozwiązania, buduje on rozwiązania w przód.

Łatwo sprawdzić, że dla rozwiązania dopuszczalnego nie zmienia on postaci rozwiązania:

ale:

co nie zachodzi dla rozwiązań dopuszczalnych.

4.6. Hybrydowy algorytm ewolucyjny.

Z uwagi na wielowymiarowość problemu i dynamikę algorymu ewolucyjnego zostały zaimplementowane dodatkowe metody optymalizacji najlepszego rozwiązania.

Obydwie metody optymalizacji można zakfalifikować do metod zachłannych. Optymalizuje ona przechodząc po rozwiązaniu w przód przeszukując wszystkie dopuszczalne wartości i wstawia w jej miejsce wartość minimalizującą funkcję celu.

For t = 1:end

Solution = min(Solution(t) = d for d in feasible)

Druga metoda zachłannej optymalizacji iteruje wiele razy po tym samym roziwiązaniu dopóki następują jakieś zmiany.

While Solution has changed

For t = 1:end

Solution = min(Solution(t) = d for d in feasible)

1. **Parametry Algorytmu**

5.1. Parametry funkcji celu.

* ***name\_of\_fitness\_function****: str –* parameter określający pożądaną funkcję celu. Możliwe wartości: “TimeSeriesCovidProblemNaive”, “TimeSeriesCovidProblem” (obie funkcje opisane w rozdziale 2.1);
* worker\_cost – ***worker\_cost*** *: float–* koszt każdego pracownika w jednostce czasu;
* death\_probability***death\_probablity****: float –* prawodpodobieństwo śmierci pacjenta, który nie został zbadany wymazem;
* ***cost\_of\_death***: float – oczekiwany koszt śmierci pacjenta;
* ***training\_cost****: float –* koszt trenowania pracownika;
* ***swabs\_per\_day****: int –* ilość wymazów wykonywana przez pracownika w jednostce czasu;
* ***timer\_series****: np.ndarray –* szereg czasowy określający przewidywana ilość zakażeń na następny rok – otrzymany przy pomocy danych historycznych oraz modelu ARIMA;

Parametry używane wyłącznie w przypadku używania funkcji celu “TimeSeriesCovidProblem”:

* ***delay****: int –*
* ***cost\_of\_non\_immediate\_swab****: float –* oczekiwany koszt nieprzeprowadzenia natychmiastowego wymazu;
* ***days\_for\_swab****: int –* jednostki czasu (dni, tygodnie) w których można przeprowadzić wymaz;
* ***delayed\_cost****: bool –* oczekiwany koszt opóźnienia wykonania wymazu;
* ***learning\_type****: str –*
* ***learning\_parameter****: float –*

5.2. Parametry algorytmu ewolucyjnego.

* ***allow\_eps\_ff\_stop****: bool* –
* ***esp\_ff****: float* –
* ***eps\_ff\_type****: str* –
* ***allow\_no\_iter\_stop****: bool –* czy ograniczenie liczby iteracji to kryterium stopu;
* ***no\_iter****: int –* liczba iteracji (kryterium stopu);
* ***allow\_indifferent\_population\_stop****: bool –*
* ***population\_diversity\_measure****: str –*
* ***pop\_div\_eps:*** *float* ***–***
* ***population\_number****: int – ­*liczność zbioru populacji w każdej iteracji (liczba rozwiązań).

5.3. Parametry rozwiązania.

* ***problem\_name****: str –* rodzaj rozwiązywanego problemu. Możliwe wartości: „TSOCOV19D”;
* ***create\_feasible****: bool –* Wartość logiczna określająca, czy początkowe rozwiązanie ma być dopuszczalne;
* ***solution\_size****: int* – rozmiar rozwiązania (długość wektora);
* ***L\_limit****: int –* dolny limit zatrudnionych pracowników, wartość >=0;
* ***DL\_limit****: int –* górny limit dodatniego przyrostu liczby pracowników;
* ***L0****: int –* liczba zatrudnionych pracowników w ostatni dzień poprzedniego roku.

5.4. Parametry mutacji.

* *type\_of\_mutation: str –* typ mutacji uzywanej w algorytmie. Możliwe wartości: „Cauchy”, „Gaussian”;
* *mutation\_probability: float –* prawdoopodobnieństwo przeprowadzenia przez algorytm mutacji w każdej iteracji;

Parametry używane tylko w przypadku używania mutacji Cauchy’ego:

* *mu –* parametrśredniej rozkładu Cauchy’ego;
* *gamma –* parametrgamma (określający rozrzut) rozkładu Cauchy’ego;

Parametry używane tylko w przypadku używania mutacji gaussowskiej:

* *mean –* parametrśredniej rozkładu Gaussa;
* *std –* odchylenie standardowe rozkładu Gaussa;

5.5. Parametry krzyżowania

* ***type\_of\_crossover****: str –* typ pożadanej krzyżówki. Możliwe wartości: "One Point", "Two Points", "Average", "Convex Combination", "Uniform";
* ***crossover\_probability****: float –* prawdopodobieństwo wykonania krzyżówki przez algorytm w każdej iteracji;

Parametr używany wyłącznie w przypadku używania krzyżówek *One Point* oraz *Two Points*:

* ***distribution\_of\_cut****: str –* typ rozkładu, według którego przeprowadzane jest cięcie. Możliwe wartości: „Uniform”;

Parametr używany wyłącznie w przypadku używania krzyżówki *Convex Combination*:

* ***alpha\_distribution****: str* – typ rozkładu, według któregow każdej iteracji wybierany jest parametr alfa. Możliwe wartości: „Uniform”, „Arcsine”.

5.6. Parametry selekcji

* ***type\_of\_selection****: str–* typ przeprowadzanej w każdej iteracji algorytmu selekcji. Możliwe wartości: „Roulette”, „Tournament”;
* ***elite****: bool –* wartość logiczna określająca, czy chcemy użyć selekcji elitarnej;
* ***truncation****: bool –* wartość logiczna określająca, czy chcemy użyć mechanizmu ucinania;
* ***elite\_count****: int –* liczność zbioru elitarnego;
* ***proportion****: float –* proporcja najlepszych osobników używany w mechanizmie cięcia;

Parametry używane tylko w przypadku używania selekcji turniejowej:

* ***k****: int –* wielkość turnieju (liczba uczestników);
* ***p****: float –* prawdopodobieństwo wygrania najlepszego osobnika w turnieju.

1. **Aplikacja Algorytmu**
2. **Testy algorytmu**

**Kolejne testy to pewnie 7.1. 7.2 itd.**

**Najpierw niech nam zadziała kod**

1. **Podsumowanie**