

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>1</b>
<b>2</b>	<b>Model matematyczny</b>	<b>1</b>
2.1	Opis problemu: . . . . .	1
2.1.1	Stałe: . . . . .	2
2.1.2	Zmienne: . . . . .	2
2.1.3	Postać rozwiązania: . . . . .	2
2.1.4	Postać funkcji celu: . . . . .	2
2.1.5	Ograniczenia: . . . . .	2
<b>3</b>	<b>Implementacja</b>	<b>3</b>
3.1	Implementacja klasy jako modelu rozwiązania . . . . .	3
3.2	symulowane wyżarzanie . . . . .	3
3.2.1	metody algorytmu . . . . .	3
3.3	algorytm genetyczny . . . . .	4
3.4	wyniki . . . . .	4
<b>4</b>	<b>Problemy</b>	<b>4</b>

## 1 Wstęp

W ramach projektu każdy z naszej trójki miał wymysleć pomysł optymalizacji który można by zaimplementować w ramach zajęć oraz rozwiązać implementując jeden z algorytmów optymalizacyjnych.

- Pomysł Dawida zakładał optymalizację wydatków związanych z zakupem opału w sezonie grzewczym
- Pomysł Piotrka opierał się na optymalizacji zysków z hodowli roślinnej w średnim gospodarstwie rolnym.
- Pomysł Bartka bazował na maksymalizacji jakości komponentów w składanym komputerze przy minimalizacji kosztów

Wspólną decyzją był pomysł Piotrka optymalizacji gospodarstwa rolnego.

## 2 Model matematyczny

### 2.1 Opis problemu:

Problem polega na stworzeniu kilkuletniego planu upraw dla niewielkiego gospodarstwa rolnego w zależności od zmiennej kategorii) jakości gleby (w postaci cyfry w zakresie od 0 - 100) i odległości uprawy od gospodarstwa. Celem będzie maksymalizacja zysków . Zakładamy przy tym że co roku nabywamy nowy materiał siewny.

### 2.1.1 Stałe:

- $N$  - Liczba dostępnych pól uprawnych.
- $Y$  - liczba lat planowania upraw.
- $T$  - stały koszt dojazdu na kilometr
- $P$  - powierzchnia pola uprawnego w hektarach (każde pole ma identyczną powierzchnię)
- $D_i$  - Odległość  $i$ -tego pola od gospodarstwa, gdzie  $i = 1, \dots, N$
- $C_x$  - koszt produkcji danej rośliny na jeden hektar (koszt materiału siewnego, koszt pracy ludzkiej, itp.), gdzie  $x$  - nazwa rośliny
- $W_x$  - wpływ uprawy na glebę (zależne od uprawianej rośliny)
- $S_x$  - zsumowana ilość dopłat i wszelkich dodatków (w zależności od uprawianej rośliny)
- $G = [g_{qx}]$  - macierz zysków z pola gdzie komórka  $g_{qx}$  zawiera zysk z danej rośliny w zależności od jakości gleby  $q$  i uprawianej rośliny  $x$ .

### 2.1.2 Zmienne:

- $y$  - Obecny rok,  $y = 1, \dots, Y$
- $Q = [q_{yi}]_{Y \times N}$  - Macierz klas jakości gleby gdzie komórka  $q_{yi}$  zawiera jakość ziemi którą na  $i$ -tym polu w roku  $y$ .

### 2.1.3 Postać rozwiązania:

- $X = [x_{yi}]_{Y \times N}$  - macierz decyzyjna o wymiarach  $Y \times N$ , gdzie komórka  $x_{yi}$  zawiera indeks rośliny którą siejemy na  $i$ -tym polu w roku  $y$ .

### 2.1.4 Postać funkcji celu:

$$f(X) = \sum_{y=1}^Y \sum_{i=1}^N G_{q_{yi}x_{yi}} + S_{x_{yi}} - (C_{x_{yi}} * P + D_i * T) \quad (1)$$

$$q_{yi} = q_{(y-1)i} + W_{x_{(y-1)i}} \quad (2)$$

### 2.1.5 Ograniczenia:

- $0 \leq q_{yi} \leq 100$  Jakość gleby może zmieniać się w zakresie od 0 do 100
- $x_{i-1} \neq x_i$ , gdzie  $x_k$  nie jest stanem pustym pola

## 3 Implementacja

Naszą implementację zaczęliśmy od zaimplementowania modelu matematycznego w formie funkcji pythonowej.

Postać rozwiązania jest przedstawiana w postaci macierzowej (listy list w pythonie), gdzie wiersze przedstawiają lata symulacji zaś numery kolumn odpowiadają odpowiedniemu polu.

Funkcja celu matematycznie jest zapisana w formie podwójnej sumy, zaś zaprogramowana jako podwójna pętla for. W programie nie jest to oczywiste ponieważ pętla po latach uprawy wywołuje w sobie funkcję pomocniczą w której jest kolejna pętla już idąca po polach.

### 3.1 Implementacja klasy jako modelu rozwiązania

Początkowo zakładaliśmy że cały nasz projekt będzie w postaci jednej klasy Pythona jednakże w trakcie rozwiązywania problemu doszliśmy do wniosków że chcielibyśmy porównać dwa sposoby rozwiązania problemu dlatego też w pierwotnej klasie ... !TODO

### 3.2 symulowane wyżarzanie

Nasz problem, na podstawie sugestii pani Profesor postanowiliśmy rozwiązać algorytmem symulowanego wyżarzania (z ang. simulated annealing). Jest to nasz pierwszy pomysł na rozwiązanie problemu, w dalszej części opisujemy drugi, który jest zaimplementowany na podstawie metaheurystyki algorytmu genetycznego (z ang. genetic alg.).

#### 3.2.1 metody algorytmu

Aby zaimplementować nasze rozwiązanie metaheurystyką symulowanego wyżarzania potrzebowaliśmy zdefiniować następujące funkcje:

- `simulated_annealing`
- `__annealing_temp`
- `__annealing_neig`
- `__annealing_P`

**annealing temperature** Funkcja wspomagająca algorytm symulowanego wyżarzania, zwraca temperaturę według wzoru  $1 - (k + 1)/k_{max}$  (jeśli wyrażenie jest pozytywne) lub  $1/k_{max}$  jeżeli wzór zwraca wartości negatywne. Ten sposób wyliczania temperatury wydawał się najprostszy i najpraktyczniejszy do zaimplementowania.

**annealing Probability** Funkcja pomocnicza akceptująca następujące parametry: wartość  $f$ . celu dla obecnego rozwiązania, wartość  $f$ . celu dla nowego rozwiązania, obecną temperaturę. Funkcja wylicza prawdopodobieństwo przejścia do wybranego, nowego rozwiązania. Jako że maksymalizujemy to funkcja zwraca 1 jeżeli nowe rozwiązanie ma wyższą wartość niż obecne, zaś w przeciwnym wypadku jest wyliczane według wzoru:  $\exp((-1) * (f - f_{new})/temp)$ .

**annealing neighbour** Funkcja pomocnicza wybierająca kandydata na nowe rozwiązanie. Kandydat jest macierzą (wyjaśnienie tutaj) w której zmieniono w randomowy sposób jedną roślinę (randomowo dobieramy rok i pole, czyli pozycję w macierzy). Roślina na jaką zamieniamy to pole w macierzy wybieramy randomowo z listy roślin z wykluczeniem poprzedniej na tej pozycji. Funkcja wykorzystuje model brute-force do sprawdzenia czy nowe rozwiązanie nadaje się do symulacji, tzn. czy nie wyskoczy żaden błąd przy próbie odpalenia simulate farm, zaś jeżeli wyskoczy rekurencyjne powtórzenie funkcji.

**simulated annealing** Główna funkcja zaimplementowanego algorytmu na podstawie pseudo-kodu:

- Let  $s = s_0$
- For  $k = 0$  through  $k_{\max}$  (exclusive):
  - $T \leftarrow \text{temperature}(1 - (k+1)/k_{\max})$
  - Pick a random neighbour,  $s_{\text{new}} \leftarrow \text{neighbour}(s)$
  - If  $P(E(s), E(s_{\text{new}}), T) \geq \text{random}(0, 1)$ :
    - $s \leftarrow s_{\text{new}}$
- Output: the final state  $s$

Rysunek 1: Pseudo-kod głównej części algorytmu simulated annealing

Algorytm działa  $k_{\max}$  razy w pętli for, w pierwszym kroku każdej iteracji jest przypisywana nowa temperatura zwracana z funkcji annealing temp

### 3.3 algorytm genetyczny

### 3.4 wyniki

## 4 Problemy

Indeksacja

wybór następnego kandydata na rozwiązanie