AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Projekt dyplomowy

Modelowanie i optymalizacja produkcji rolnej
Modeling and Optimization of Agricultural Production

Autor: Piotr Paweł Hudaszek
Kierunek studiów: Automatyka i Robotyka
Opiekun pracy: Dr inż. Piotr Kadłuczka

Kraków, 2022

Spis treści

Wstęp

Rolnictwo jest ważną dziedziną gospodarki, więc naturalne jest się staranie do jak najlepszej optymalizacji tego zagadnienia. Dzięki rozpowszechnieniu technologii takich jak między innymi GPS, zdjęcia lotnicze i czujniki wilgotności gleby, można uzyskać dokładne dane do różnego rodzaju modelów rolniczych. Rozważany w tej pracy problem dotyczy planowania procesu uprawy.

Celem pracy jest stworzenie programu który ma za zadanie zoptymalizować terminarz upraw na określonej liczbie pól tak aby jak najlepiej wykorzystać dostępne zasoby. Został on napisany w Pythonie jako że jest to jeden z bardziej popularnych języków programowania który wciąż się rozwija i zyskuje popularność [PYPL]. Jako algorytm optymalizacji wybrano algorytm genetyczny.

W pierwszym rozdziale został opisany model matematyczny wraz z rozważaniem na temat użytych uproszczeń i przydatności modelu w realnych zastosowaniach. Rozdział drugi jest poświęcony przystosowaniu algorytmu genetycznego do podanego problemu w tym opisanie użytych operatorów genetycznych. W dalszej części pracy zostały zaprezentowane rozwiązania implementacyjne oraz przeprowadzone testy.

1 Opis zagadnienia

1.1 Model opisowy

Opisywane zagadnienie polega na optymalizacji zysku z uprawy w przedsiębiorstwie rolnym. Jest możliwość uprawy różnych rodzajów produktów. Każdy rodzaj potrzebuje określone zasoby na danym etapie uprawy. Przedsiębiorca ma dostęp do pewnej ilości każdego rodzaju zasobów. Zasoby dzielą się na dwie klasy: zasoby dzienne które są przydzielone na dany dzień oraz zasoby całkowite przydzielone na cały rozpatrywany okres uprawy. Jeśli uprawa zostanie przeprowadzona i zasobów nie zabraknie to przedsiębiorca uzyska określony bazowy zysk z jednostki pola, który zależny jest od rodzaju produktu. Rozpatrywane przedsiębiorstwo ma określoną liczbę pól. Każde pole można opisać jego obszarem oraz współczynnikami które określają jak dobrze dane pole pasuje do uprawy danego rodzaju produktu. Aby uzyskać rzeczywisty zysk należy przemnożyć zysk bazowy z współczynnikiem dopasowania oraz wielkością pola.

Optymalizacja polega na wyborze jaki produkt jest uprawiany na danym polu oraz kiedy rozstanie rozpoczęta uprawa. Na polu można uprawiać kilka produktów po sobie. Data rozpoczęcia uprawy jest ograniczona przez ramy czasowe danego produktu.

1.2 Model matematyczny

Dane:

$t \in T$ - dzień

$p \in P$ - pole dostępne do uprawy

S_p - powierzchnia pola p

y_{pn} - rodzaj n-tej rozpoczętej uprawy na polu p

x_{pn} - dzień rozpoczęcia n-tej uprawy na polu p

d_y - dochód z jednostki pola na którym był uprawiany produkt y

w_{py} - współczynnik dopasowania pola p do produktu y

$z_d \in Z_d$ - dostępny zasób dzienny (odnawialny po każdym dniu)

$z_c \in Z_c$ - dostępny zasób całkowity (przydzielony na cały rozpatrywany okres uprawy)

z_{dyt} - zasoby dzienne potrzebne do uprawy produktu y w t dniu uprawy

z_{cyt} - zasoby całkowite potrzebne do uprawy produktu y w t dniu uprawy

t_{y0} - pierwszy dzień w którym można zacząć uprawę produktu y

t_{yN} - ostatni dzień w którym można zacząć uprawę produktu y

o_y - czas trwania uprawy produktu y

Postać rozwiązania:

Lista dwu elementowych wektorów dla każdego pola gdzie pierwszy element x oznacza dzień rozpoczęcia a drugi y rodzaj uprawy.

$$\begin{Bmatrix} (x_{11}, y_{11}) & (x_{12}, y_{12}) & \cdots & (x_{1N_1}, y_{1N_1}) \\ (x_{21}, y_{21}) & (x_{22}, y_{22}) & \cdots & (x_{2N_2}, y_{2N_2}) \\ \vdots & \vdots & \ddots & \vdots \\ (x_{P1}, y_{P1}) & (x_{P2}, y_{P2}) & \cdots & (x_{PN_P}, y_{PN_P}) \end{Bmatrix}$$

Funkcja celu:

Zysk ze sprzedaży uzyskanych produktów .

$$f(x_{pn}, y_{pn}) = \sum_{p \in P} \sum_{n \in N_p} S_p d_{y_{pn}} w_{py_{pn}} \rightarrow max$$

Ograniczenia:

Dla każdego dnia t każdego rodzaju zasobu dziennego ilość użyta nie może przekroczyć dostępnej.

$$\forall z_d \in Z_d \; \forall t \in T \quad \sum_{p \in P} z_{dtp} \leq z_d$$

Dla każdego rodzaju zasobu całkowitego ilość użyta przez cały okres uprawy nie może przekroczyć dostępnej.

$$\forall z_c \in Z_c \quad \sum_{t \in T} \sum_{p \in P} z_{ctp} \leq z_c$$

Uprawę produktu typu y można zacząć w wyznaczonym oknie czasowym.

$$t_{y0} \leq o_y \leq t_{yN}$$

W danym dniu na jednym polu można uprawiać tylko jeden rodzaj produktu.

$$\forall p \in P \; \forall n \in N_p \quad x_{pn} + d_y < x_{pn+1}$$

1.3 Model a rzeczywisty problem

Tak sformułowany model pozwala oddać typowe zależności podczas planowania uprawy tak jak ograniczenie terminu rozpoczęcia uprawy i konieczne zasoby. Ilość zasobów nie jest zdefiniowana na sztywno pozwala to użytkownikowi programu wprowadzić zasoby specyficzne dla danego typu uprawy i sprawić że program jest bardziej uniwersalny. Model można też zastosować do innych zagadnień niż rolnicze na przykład do bilansowania zasobów w zakładzie produkcyjnym.

Model nie bierze pod uwagę wpływu nieprzewidywalnych zmian cen oraz pogody. Natomiast użytkownik może wprowadzić odpowiednio większe potrzebne ilości zasobów lub dłuższy czas uprawy aby mieć pewien margines bezpieczeństwa.

Mimo tych ograniczeń uważam że model może być przydatny. Szczególnie w uprawie szklarniowej gdzie wpływ pogody jest ograniczony oraz w przypadku gdy optymalne wykorzystanie któregoś z zasobów jest kluczowe. Tak jest na przykład w przypadku rejonów z ograniczoną ilością wody[Optimal Use of Irrigation Water].

1.4 Podobne modele w literaturze

Przedstawiony model nie jest oderwany od rozważanych problemów w literaturze. Poniżej opisano kilka interesujących przykładów.

Jednym z pierwszych opisanych przykładów zastosowań badań operacyjnych w rolnictwie jest praca Doktora C.W. Thornthwaite w latach 1946-50. Dotyczyła ona uprawy grochu na 7000 arowej farmie. Problemy, z jakimi borykała się farma to:

1. nie było naukowej metody na znalezienie odpowiedniego czasu na zbiory,
2. cała uprawa dojrzewała na raz, w związku z czym stało się niemożliwe nadążanie za dojrzewającym grochem nawet przy użyciu wszystkich maszyn i siły roboczej.

Pierwszy problem został rozwiązany za pomocą badania wpływów czynników pogodowych na proces dojrzewania groszku. Natomiast drugi został rozwiązany za pomocą terminarza upraw, dzięki któremu cały groch nie dojrzewał w tym samym czasie[Urja Thakkar]. W mojej pracy nie poruszono problemu wyznaczania parametrów różnych rodzajów upraw i procesu ich uprawy, ale umożliwiono ich wygodne wprowadzenie do programu.

Bardziej skomplikowany problem przedstawia praca mająca na celu optymalizację planu upraw i hodowli w regionie Chang Qing w Chinach. Ważnym założeniem w pracy było nie wywieranie zbyt negatywnego efektu na środowisko naturalne oraz odporność planu na

ekstremalne warunki pogodowe (susza lub powódź). Wykorzystano do tego celu teorię gier oraz programowanie liniowe. Uzyskane wyniki zastosowano w latach 1983-1985 i na ich skutek zysk z upraw wzrósł o 12,33%, a z hodowli o 53,77% [Zhao Qingzhen].

Praca [Ivano heterogeneous] dotyczy rejonu Irkutska w Rosji, w niej również zwrócono uwagę na wpływ ekstremalnych warunków pogodowych. Porównano w niej metodę programowania liniowego oraz stochastycznego programowania liniowego. Metoda stochastyczna biorąca pod uwagę ryzyko wystąpienia niepożądanych warunków pogodowych pozwoliła lepiej zaalokować dostępne zasoby. Zwrócono też uwagę na wykorzystanie technologii rolnictwa precyzyjnego [Stanisław Samborski Rolnictwo precyzyjne] które pozwala gromadzić dane o przestrzennym zróżnicowaniu pola i stanu uprawy. Dzięki tym danym można między innymi wybiórczo stosować zabiegi nawożenia, co sprawia że zasoby są lepiej wykorzystywane.

Ciekawym problemem opisanym w pracy [wild food resources] jest połączenie uprawy rolnej ze zbiorem dziko rosnących dóbr. Takie podejście według autorów może mieć zastosowanie w wielu rzadziej zaludnionych rejonach.

Nie można zapomnieć o tym aby prowadzone zabiegi optymalizacyjne nie odbywały się kosztem środowiska naturalnego ani niszczenia gleby. Jednak ciężko jest te warunki uwzględnić w modelu nie posiadając bogatej wiedzy na temat rolnictwa i ekologii. Dlatego w przedstawionym programie udostępniono użytkownikowi narzędzie pozwalające w pewnym stopniu ograniczyć negatywny wpływ na środowisko. Tym narzędziem jest wprowadzanie ograniczeń. Można na przykład wprowadzić limit dzienny na zużycie nawozów lub środków ochrony roślin. Można też wprowadzić jak bardzo każdy rodzaj uprawy wyjaławia glebę i ograniczyć dopuszczalne wyjałowienie gleby.

2 Opracowanie algorytmu ewolucyjnego

2.1 Zasada działania algorytmów ewolucyjnych

Algorytmy ewolucyjne są bazowane na biologicznej ewolucji sformułowanej po raz pierwszy przez Charles'a Darwina. Jego teoria tłumaczy adaptacyjne zmiany w osobnikach poprzez naturalną selekcję, która sprzyja przetrwaniu najlepiej przystosowanych do warunków środowiskowych osobników. Poza selekcją kolejnym ważnym czynnikiem rozpoznanym przez Darwina jest występowanie małych, losowych i nie ukierunkowanych zmian pomiędzy fenotypami (zespół cech organizmu). Te mutacje przeżywają w populacji jeżeli okazują się przydatne w aktualnym środowisku, w przeciwnym wypadku wymierają[Bäck, Evol. algo.].

Siłą napędową naturalnej selekcji jest produkcja potomków, co w efekcie sprawia że wielkość populacji rośnie. Jest ona jednak ograniczona poprzez skończone zasoby dostępne dla populacji. To sprawia że osobniki lepiej wykorzystujące zasoby mają przewagę.

Jeśli zamiast osobników użyjemy rozwiązań problemu który chcemy zoptymalizować, oraz odpowiednio zdefiniujemy reguły mutacji, selekcji oraz tworzenia potomków to można naśladując naturę zbudować algorytm optymalizacji. Warto jednak zwrócić uwagę na to że proces ewolucji zachodzącej w naturze nie ma na celu optymalizacji. Jest to automatyczny proces który się cały czas toczy, ale nie dąży do niczego[The Blind Watchmaker]. Optymalizacja jest efektem pobocznym ewolucji naturalnej. Dlatego natura jest inspiracją, a nie ścisłym zbiorem reguł dla algorytmów ewolucyjnych.

2.2 Schemat algorytmu

Programy ewolucyjne są zbudowane według poniższego ogólnego schematu [Michalewicz Algorytmy genetyczne …] gdzie P(t) oznacza populację w iteracji t.

```
procedure program ewolucyjny
begin
  t←0
  ustal początkowe P(t)
  oceń P(t)
  while (not warunek zakończenia) do
  begin
    t←t+1
    wybierz P(t) z P(t-1)  //Selekcja
    zmień P(t)             //Mutacja oraz krzyżowanie
    oceń P(t)
  end
end
```

W tej pracy przedstawię jak przystosowałem powyższy schemat do przedstawionego problemu.

2.3 Populacja początkowa

Pierwszym krokiem w algorytmach populacyjnych jest ustalenie populacji początkowej. Wybór tej populacji zależy od problemu, ale zazwyczaj dąży się aby wszystkie rozwiązania w populacji były dopuszczalne oraz aby rozwiązania nie były skupione w jednym punkcie.

Populację początkową można wygenerować jako populację rozwiązań zerowych czyli takich bez jakiejkolwiek rozpoczętej uprawy w rozwiązaniu. Jest to jednak nie najlepszy sposób z powodu tego że część iteracji algorytmu na początku będzie musiała być poświęcona na wstępne zapełnienie rozwiązań. Można ten czas oszczędzić generując od razu populację w której każde rozwiązanie ma już kilka zaczętych upraw. Przykład takiej metody jest przedstawiony poniżej

Dla każdego rozwiązania w populacji losujemy kolejność pól zgodnie z którą będzie uzupełniane rozwiązanie. Następnie dla każdego pola losujemy uprawę oraz dzień rozpoczęcia z przedziału dopuszczalnego dla danej uprawy. Jeśli po sprawdzeniu ograniczeń zasobowych rozwiązanie z nową uprawą jest dopuszczalne to dodajemy tą uprawę, jeśli nie to przechodzimy do następnego pola w kolejności.

Plusy tej metody są następujące żadne pole ani rodzaj uprawy nie będzie faworyzowany oraz otrzymamy zawsze rozwiązanie dopuszczalne. Minusem jest to że metoda jest bardziej skomplikowana od generacji rozwiązań zerowych oraz to że metoda stworzy co najwyżej jedną uprawę na polu.

2.4 Selekcja

Selekcja polega na wyborze rozwiązań które przechodzą do następnej iteracji algorytmu. Celem tego etapu jest wprowadzenie presji ewolucyjnej. Sprawia ona że rozwiązania dążą do poprawy funkcji celu. Bez selekcji populacja rozwiązań losowo błądziłaby po przestrzeni rozwiązań. W metodach selekcji opisanych poniżej lepsze rozwiązania mają większe prawdopodobieństwo przejścia do następnej iteracji.

W metodzie koła ruletki prawdopodobieństwo wyboru danego osobnika jest wprost proporcjonalne wartości jego funkcji celu. Aby wykorzystać tą metodę należy najpierw zsumować wartości funkcji celu wszystkich osobników w populacji, a następnie dla każdego osobnika podzielić wartość funkcji celu przez tą sumę. Otrzymamy w ten sposób prawdopodobieństwa wyboru dla każdego osobnika sumujące się do 1. Prawdopodobieństwo wyboru osobnika opisuje wzór:

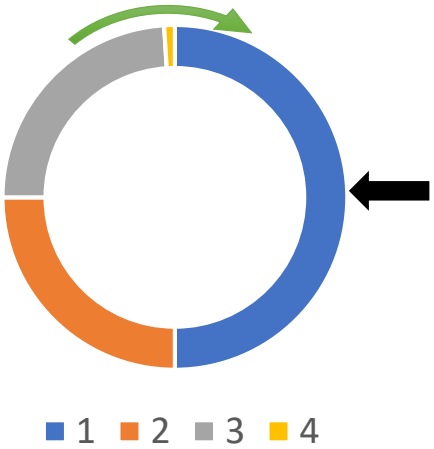
$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

gdzie:

f_i - wartość funkcji celu osobnika i,

N - liczba osobników w populacji.

Następnie należy przypisać każdemu rozwianiu część koła zgodnie z prawdopodobieństwem wyboru oraz określić punkt wyboru rozwiązania. Po zakręceniu kołem osobnik na którym znajdzie się ten punkt jest wybierany. Należy zakręcić kołem tylekrotnie ile chcemy wybrać osobników.



Osobnik	Wartość funkcji celu	Prawdopodobieństwo wyboru
1	25	0,5
2	12,5	0,25
3	12	0,24
4	0,5	0,01

Rys. 1 Metoda ruletki, gdzie czarna strzałka oznacza punkt wyboru rozwiązania, a tabela po prawej przedstawia obliczenia prawdopodobieństwa.

Standardowa wersja tej metody nie pozwala na zastosowanie w przypadku ujemnych wartości funkcji celu. Można jednak zdefiniować podobną metodę która różni się tylko tym że na początku dodaje do wszystkich wartości funkcji celu najbardziej ujemną wartość funkcji celu w populacji. Niestety tracimy wtedy własność że prawdopodobieństwo wyboru jest wprost proporcjonalne wartości funkcji celu. Jeśli nie występują wartości ujemne to ta metoda działa tak samo jak standardowa. Minus tej metody uwidacznia się kiedy wszystkie osobniki mają bardzo bliską wartość funkcji celu a przez to bliskie prawdopodobieństwo wyboru. Wtedy presja ewolucyjna jest efektywnie tracona.

W selekcji rankingowej osobniki w populacji są sortowane rosnąco ze względu na wartość funkcji celu. Selekcja polega na wyborze określonej liczby osobników z końca posortowanej listy. Plusem tej metody jest zachowanie presji ewolucyjnej również wtedy gdy osobniki mają bliskie wartości funkcji celu. Metoda działa zarówna dla ujemnych i dodatnich

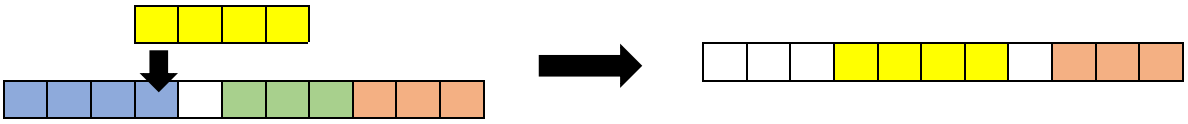
wartości funkcji celu. Minusem tej metody jest brak losowości w wyborze, przez co gorsze rozwiązania nie maja możliwości bycia wybranym. Może to negatywnie wpływać na różnorodność w populacji.

W selekcji turniejowej przeprowadzamy tyle turniejów ile osobników potrzebujemy wybrać. Turniej polega na wyborze najlepszego rozwiązania z grupy losowych rozwiązań. Wielkość tej grupy wynosi dwa lub więcej osobników. Metoda działa zarówna dla ujemnych i dodatnich wartości funkcji celu.

2.5 Mutacja

Mutacja w algorytmach ewolucyjnych ma na celu zróżnicowanie rozwiązań i przeciwdziała utknięciu populacji w lokalnym ekstremum. Polega ona na małej zmianie losowej części rozwiązania.

Proponowana metoda mutacji polega na losowym wyborze typu uprawy, następnie dnia rozpoczęcia z możliwych dla danego typu. Należy też wybrać pole na którym zostanie rozpoczęta ta uprawa. Można wybrać pole losowo gdzie każde pola ma równe prawdopodobieństwo, lub uwzględnić współczynnik dopasowania pola do typu produktu. Dobrze nadaje się do tego do tego metoda ruletki w której prawdopodobieństwo wyboru jest wprost proporcjonalne do współczynnika dopasowania. Jeśli na danym polu i terminie są już uprawy to należy je usunąć z rozwiązania.



Rys. 2 Zasada wstawiania nowej uprawy (kolor żółty) do harmonogramu upraw na polu.

Mutacji poddawana jest w każdej iteracji część osobników. Osobniki do mutacji są wybierane losowo zgodnie z pewnym prawdopodobieństwem. Jest ono takie samo dla każdego osobnika i zazwyczaj takie samo dla wszystkich iteracji algorytmu. Właściwy dobór prawdopodobieństwa mutacji jest ważnym zagadnieniem i istotnie wpływa na skuteczność algorytmu.

2.6 Krzyżowanie

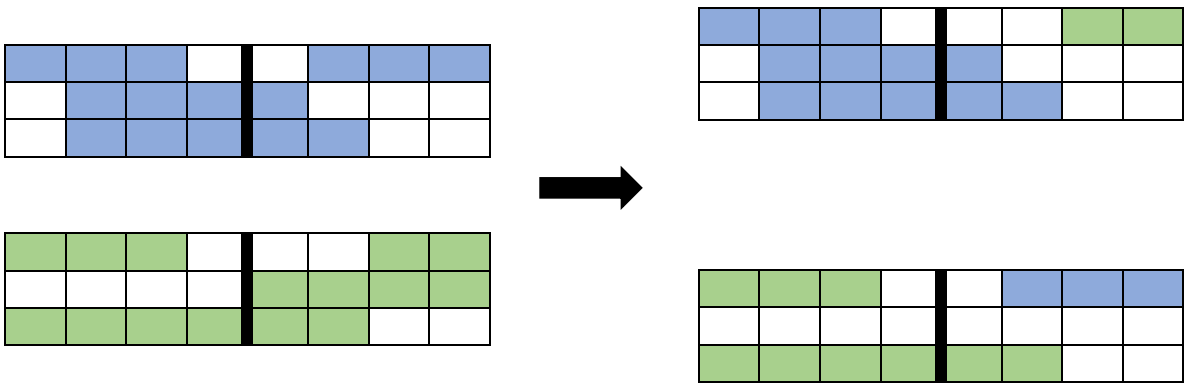
Krzyżowanie ma za zadanie stworzenie nowego rozwiązania wykorzystując do jego budowy dwa inne rozwiązania. Dzięki krzyżowaniu uzyskujemy rozwiązania o kombinacji cech rodziców. Jeśli cechy te są dobre pod względem funkcji celu to otrzymamy osobnika lepszego od każdego z rodziców. Co prawda możliwe jest również otrzymanie osobnika gorszego od obu rodziców, ale takie rozwiązanie zostanie prawdopodobnie odrzucone na etapie selekcji.

W przedstawionym problemie nasuwają się dwa główne sposoby na przeprowadzenie krzyżowania. Krzyżowanie ze względu na:

- dni,
- pola.

Krzyżowanie ze względu na dni polega na wyborze części dni z jednego rozwiązania i reszty z drugiego. Problemem który pojawi się podczas przeprowadzania tego typu krzyżowania są uprawy będące w trakcie na granicy łączenia. Aby zminimalizować ten problem zastosowano krzyżowanie jednopunktowe. Jednak nie eliminuje to całkowicie tego problemu.

Proponowane rozwiązanie polega na wzięciu wszystkich upraw do punktu podziału z pierwszego rodzica, a po tym punkcie z drugiego rodzica. W ten sposób otrzymamy część upraw z pierwszego rodzica nie skończonych, a część z drugiego nie zaczętych. Uprawy rozpoczęte kończymy, a jeśli koliduje to z innymi uprawami to je usuwamy. Uprawy nie zaczęte usuwamy. Powtarzamy to zamieniając kolejnością rodziców aby otrzymać drugie rozwiązanie. Proces jest przedstawiony na rysunku poniżej.



Rys. 3 Zasada krzyżowania rozwiązań. Z rozwiązań po lewej otrzymano nowe po prawej.

Krzyżowanie ze względu na pola jest prostsze ponieważ kolejność pól nie ma znaczenia. Wybrano metodę krzyżowania biorącą parzyste pola z pierwszego rozwiązania oraz nieparzyste z drugiego i tworzące w ten sposób nowe rozwiązanie. Aby utrzymać symetrię należy powtórzyć proces zamieniając najpierw kolejnością rozwiązania, wtedy uzyskamy drugie nowe rozwiązanie.

2.7 Ocena rozwiązania i funkcja kary

Na skutek zdefiniowanych powyżej metod krzyżowania lub mutacji mogą się pojawić rozwiązanie nie spełniające ograniczeń zasobowych. Inne ograniczenia będą zawsze spełnione.

Można ten problem rozwiązać stosując między innymi metodę funkcji kary lub funkcji poprawy.

„Ich [metody funkcji kary] istota polega na modyfikacji funkcji celu przez włączenie do niej ograniczeń za pomocą tak zwanej funkcji kary. Wprowadzenie funkcji kary przekształca zagadnienie optymalizacji z ograniczeniami w zadanie optymalizacji bez ograniczeń. Dla zadania minimalizacji funkcja kary ma wartość dodatnią, a wartość zmodyfikowanej funkcji celu rośnie proporcjonalnie do wielkości o jaką naruszone zostały ograniczenia. […] Tak więc funkcja [kary] przypisuje pewną karę liczbową każdemu punktowi spoza zbioru rozwiązań dopuszczalnych”[Kusiak optymalizacja].

Jako że mamy do czynienia z maksymalizacją w przedstawionym problemie to funkcja kary musi mieć wartość ujemną. Należy najpierw znaleźć zbiór zasobów które zostały przekroczone K. Zasoby dzienne danego rodzaju są liczone osobno dla każdego dnia i traktowane jako osobny zasób. Jest wiele możliwości konstrukcji tej funkcji. W rozpatrywanym problemie przyjęto że każdy rodzaj zasobu będzie miał taki sam wpływ na funkcję kary oraz że wartość kary będzie zależna od procent przekroczenia zasobu. Te założenia spełnia funkcja kary liczona według poniższego wzoru:

$$f = \sum_{k \in K} \frac{Z_k - z_k}{Z_k} f_{max}$$

gdzie:

k - rodzaj przekroczonego zasobu,

Z_k - dostępna ilość zasobu k,

z_k - zużyta ilość zasobu k,

f_{max} - maksymalna uzyskana dotychczas wartość funkcji celu.

Funkcja kary może być zależna od numeru iteracji. Ma to na celu sprawienie aby na początku działania algorytmu funkcja kary nie wpływała znacząco na wartość funkcji celu, ale wraz z czasem coraz bardziej. Pozwala to na lepsze przeszukanie zbioru rozwiązań w początkowych iteracjach, a jednocześnie sprawia że prawdopodobieństwo otrzymania rozwiązania niedopuszczalnego pod koniec działania algorytmu jest mniejsze.

W programie jest możliwość uwzględnienia zależności funkcji kary od iteracji poprzez podanie dwóch liczb. Pierwsza określa przez jaką liczbę należy przemnożyć funkcję kary w pierwszej iteracji, a druga przez jaką należy przemnożyć w ostatniej. Dla pozostałych iteracji liczba przez którą należy przemnożyć jest uzyskana za pomocą interpolacji liniowej.

Metoda funkcji poprawy ma na celu zmianę rozwiązania niedopuszczalnego na dopuszczalne. W przedstawionym problemie polega to na usunięciu upraw które zużywają zasoby których brakuje. Należy wybrać uprawy do usunięcia w losowy sposób oraz przerwać usuwanie kiedy rozwiązanie stanie się dopuszczalne.

Dla przekroczonych zasobów dziennych w danym dniu można to uzyskać w następujący sposób. Należy ustawić pola w losowej kolejności a następnie po kolei sprawdzać czy na polu w danym dniu jest uprawa która zużywa dany zasób. Jeśli tak to usunąć tą uprawę. Następnie sprawdzić czy zasób dalej jest przekroczony, jeśli już nie to przerwać sprawdzanie. W przypadku zasobów całkowitych czyli przydzielonych na wszystkie dni, można postępować podobnie tylko należy dodatkowo losowo wybrać kolejność dni.

2.8 Parametry algorytmu oraz warunek zakończenia

W opracowanym algorytmie jest możliwość wyboru:

- rodzaju i wielkości populacji początkowej,
- rodzaju krzyżowania,
- rodzaju oraz prawdopodobieństwa mutacji,
- funkcji kary lub poprawy,
- rodzaju selekcji.

Algorytm zakończy pracę kiedy wystąpi jeden z poniższych warunków:

- maksymalna liczba iteracji zostanie przekroczona,
- maksymalna liczba iteracji bez poprawy wartości funkcji celu zostanie przekroczona,
- nastąpi manualne zatrzymanie algorytmu.

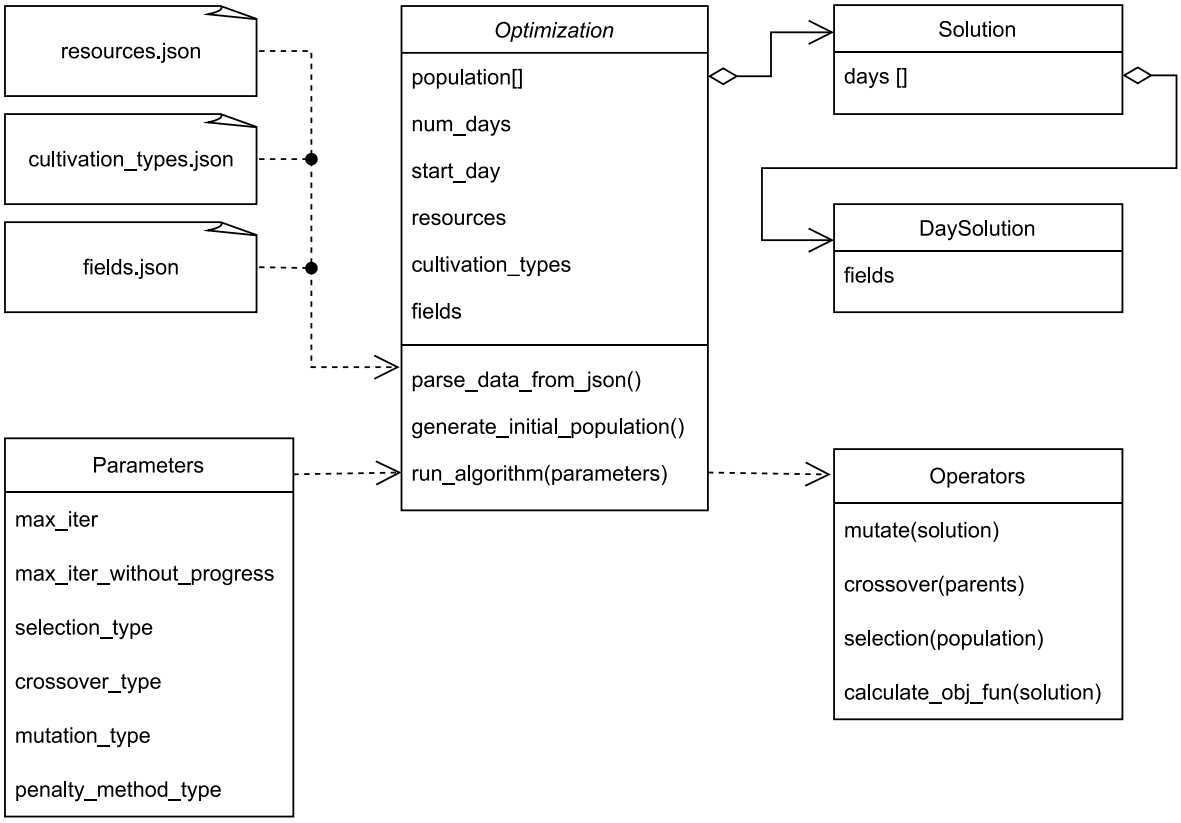
Są to standardowe warunki zakończenia pracy algorytmów optymalizacyjnych.

3 Implementacja programu

3.1 Struktura programu

Program został podzielony na trzy zasadnicze części:

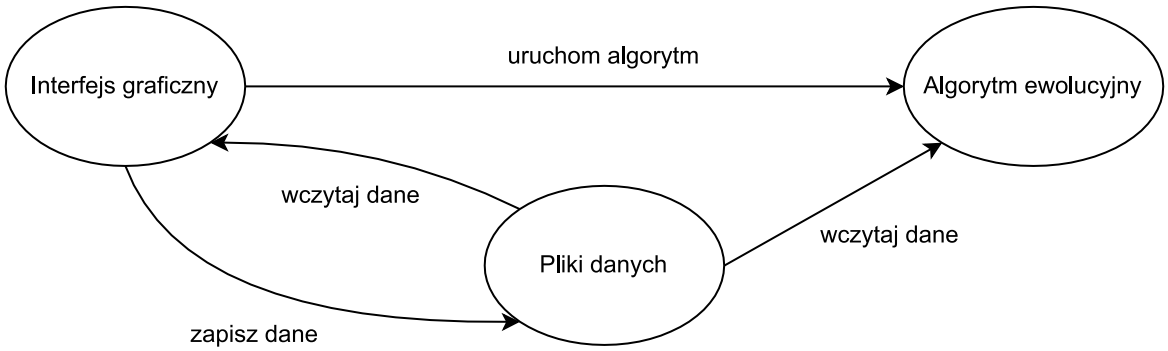
- interfejs graficzny,
- algorytm genetyczny,
- pliki danych



Rys. 4 Struktura programu

Do zapisu danych użyto plików typu JSON z powodu jego popularności oraz dobrej współpracy z Pythonem. Są one przekształcane w strukturę zagnieżdżonych list i słowników zaraz przed uruchomieniem algorytmu oraz zapisywane jako pola głównej klasy (Optimization). Przekształcenie pliku na strukturę danych jest konieczne ponieważ korzystnie bezpośrednio z pliku podczas działania algorytmu jest bardzo czasochłonne.

Dzięki podanej strukturze programu udało się wyraźnie oddzielić interfejs graficzny od algorytmu. Schemat współpracy między tymi komponentami przedstawiony jest poniżej. Każda strzałka odpowiada wywołaniu jednej metody.



Rys. 5 Schemat współpracy pomiędzy komponentami systemu

3.2 Reprezentacja rozwiązania

Rozwiązanie jest obiektem klasy …

3.3 Implementacja interfejsu graficznego

Zdecydowałem się na użycie Pythona również do utworzenia interfejsu graficznego. Nie jest to typowo język wykorzystywany do tego celu, ale taki wybór pozwala wykorzystać jego biblioteki jak między innymi matplotlib i pandas do tworzenia wykresów. Dodatkową zaletą wykorzystania jednego języka jest uproszczenie procesu budowania i instalacji programu.

Dwa najpopularniejsze narzędzia do tworzenia interfejsu graficznego w Pythonie to Tkinter [tkinter] oraz Qt [QT].

Tab 1 Porównanie popularnych narzędzi do tworzenia interfejsu graficznego w Pythonie

	Tkinter	Qt
Instalacja	Nie jest konieczna, jest to standardowa biblioteka.	Konieczna.
Prosty do zrozumienia	Tak, mała liczba funkcjonalności.	Nie, zrozumienie całej biblioteki zajmuje dużo czasu
Dostępne widżety	Tylko podstawowe widżety	Szeroki wybór

Graficzny konstruktor interfejsu	Brak	Dostępne narzędzie Qt Designer pozwalające dodawać widżety metodą przeciągnij i upuść (ang. drag and drop)
Licencja	Darmowa	Płatna dla użytku komercyjnego

4 Testy programu

4.1 Testy poprawności programu w tym testy jednostkowe

Przeprowadzono testy jednostkowe wszystkich metod w programie. Większość tych metod ma sobie element losowy, co jest problemem podczas przeprowadzania testów. Aby zniwelować ten problem w argumentach metody przekazywałem zmockwaną metodę która była później wykorzystywana do generacji liczb losowych.

Testy jednostkowe nie są wystarczające aby stwierdzić poprawność dlatego przeprowadzono również …

Sprawdzono również czy program znajdzie rozwiązanie optymalne dla bardzo prostych danych. Rozwiązanie optymalne uzyskano metodą …

Program testowano głównie na systemie windows 10 64 bit, ale odpalono program również na systemach … i przeprowadzono podstawowe testy. Testy na innych systemach nie wykazały różnic w stosunku do windowsa 10.

4.2 Testy interfejsu

Przeprowadzono manualne testy poniższych podstawowych scenariuszy end-to-end czyli przedstawiających cały proces użycia programu.

- Uruchomienie programu, wczytanie przykładowych danych, odpalenie algorytmu
- Uruchomienie programu, wczytanie przykładowych danych, zmiana danych, odpalenie algorytmu, zapisanie zmienionych danych do plików
- Uruchomienie, wczytanie zapisanych wcześniej danych, odpalenie
- Uruchomienie wpisanie danych od zera, odpalenie

Sprawdzono również działanie programu w przypadku jeśli zostaną wprowadzone niepoprawne dane. Sprawdzono poniższe scenariusze:

…

4.3 Wpływ parametrów mutacji oraz krzyżowania na wyniki

Najważniejszym parametrem mutacji jest prawdopodobieństwo jej wystąpienia. Przeprowadzono testy dla prawdopodobieństw mutacji przedstawionych w tabeli. Aby zapewnić wiarygodność testów scenariusze miały te same parametry: …. Z powodu losowości w algorytmie nie wystarczy przeprowadzić pojedynczego testu dla danego zestawu

parametrów. Przeprowadzono … testów dla każdej wartości prawdopodobieństwa oraz wyliczono z tych danych średnią.

Tabela

Sprawdzono jak zachowują się dwa zaimplementowane metody krzyżowania

4.4 Wpływ parametrów funkcji kary i poprawy na wyniki

Przetestowano funkcję kary z o poniższych parametrach…

Według [Michalewicz] skuteczność funkcji kary zależy w dużym stopniu od tego w jak dużym stopniu przestrzeń rozwiązań jest niedopuszczalna. Dlatego dla każdego zestawu parametrów przeprowadzono testy na trzech przypadkach dostępności zasobów:

- Duża ilość pól w stosunku do zasobów. Możliwość uprawy tylko na części pól z powodu braku zasobów
- Średnia ilość zasobów
- Duża ilość zasobów

Przeprowadzono podobny eksperyment dla funkcji poprawy.

…

Jak widać w tabeli …(porównanie kary z poprawą)

4.5 Wpływ selekcji oraz populacji początkowej na wyniki

Zbadano poniższe metody selekcji

- metoda ruletki
- Selekcja turniejowa
  - wielkość turnieju 2
  - wielkość turnieju 5
  - 10
- Selekcja rankingowa

4.6 Badanie działania dla rozbudowanych danych testowych

W tym rozdziale zbadano jak program poradzi sobie z bardzo rozbudowanymi danymi. Dane będą zawierały … pól, … rodzajów upraw i … rodzajów zasobów.

Jakie warunki zakończenia

5 Podsumowanie

6 Literatura

[1] Michalewicz Zbigniew, Algorytmy genetyczne + struktury danych = programy ewolucyjne z angielskiego przełożył Zbigniew Nahorski.Wydanie trzecie. Warszawa : Wydawnictwa Naukowo-Techniczne, https://katalogagh.cyfronet.pl/lib/item?id=chamo:53931&fromLocationLink=false&theme=bgagh

[2] Ibrahim M. Al-Harkan, et al, A Decision Support System for Optimal Use of Irrigation Water and Crop Selection, Journal of King Saud University - Engineering Sciences, Volume 21, Issue 2, 2009, Pages 77-84, ISSN 1018-3639, https://doi.org/10.1016/S1018-3639(18)30511-7

[3] Ya Ivanyo et al 2021 J. Phys.: Conf. Ser. 1989 012041 Optimization models of agricultural production with heterogeneous land resources https://doi.org/10.1088/1742-6596/1989/1/012041

[4] Ya Ivanyo, et al, Models of optimization of combination of production of agrarian products and harvesting of wild food resources, E3S Web of Conf. Volume 222, 2020 https://doi.org/10.1051/e3sconf/202022201016

[5] Urja Thakkar, et al, Application of Operations Research in Agriculture, International Journal of Scientific & Engineering Research Volume 10, Issue 10, October-2019 https://www.ijser.org/researchpaper/Application-of-Operations-Research-in-Agriculture.pdf

[6] Badanie popularności języków programowania PYPL PopularitY of Programming Language Index: https://pypl.github.io/PYPL.html (28.11.2022)

[7] Dokumentacja biblioteki Tkinker: https://docs.python.org/3/library/tkinter.html (28.11.2022)

[8] Dokumentacja narzędzia QT Designer: https://doc.qt.io/qt-6/qtdesigner-manual.html (28.11.2022).

[9] Maffezzoli, et al, (2022). Agriculture 4.0: A systematic literature review on the paradigm, technologies and benefits. Futures. 142. 102998. 10.1016/j.futures.2022.102998. https://www.researchgate.net/publication/362191241_Agriculture_40_a_systematic_literature_review_on_the_paradigm_technologies_and_benefits

[10] Bäck, Thomas, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms (New York, 1996; online edn, Oxford

Academic, 12 Nov. 2020),
https://doi.org/10.1093/oso/9780195099713.002.0001

[11] Stanisław Samborski, Rolnictwo precyzyjne, Wydawnictwo Naukowe PWN, ISBN: 9788301198985

[12] Anna Danielewska-Tułecka, Jan Kusiak, Piotr Oprocha, Optymalizacja Wybrane metody z przykładami zastosowań, Wydawnictwo Naukowe PWN 2009

[13] Richard Dawkins, The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design, W. W. Norton & Company, 28.09.2015

[14]Zhao Qingzhen, et al, (1991) The Application of Operations Research in the Optimization of Agricultural Production. Operations Research 39(2):194-205.
https://doi.org/10.1287/opre.39.2.194