



**Politechnika Łódzka**  
Wydział Fizyki Technicznej, Informatyki  
i Matematyki Stosowanej

POLITECHNIKA ŁÓDZKA  
WYDZIAŁ FIZYKI TECHNICZNEJ, INFORMATYKI I  
MATEMATYKI STOSOWANEJ  
KIERUNEK FIZYKA TECHNICZNA

**Praca dyplomowa inżynierska**

Modelowanie dyspersyjnych własności optycznych zwierciadeł DBR za  
pomocą algorytmów rojowych

Autor: Ramzi Hadrich

Kierujący pracą: dr hab. inż. Maciej Dems

Łódź, luty 2019



## **Streszczenie**

W niniejszej pracy zastosowano algorytmy rojowe do kontroli dyspersji w zwierciadłach DBR w celu otrzymania ultrakrótkich impulsów. Na początku została zaprezentowana zasada działania algorytmów rojowych i zostało sprawdzone czy algorytmy te faktycznie działają. W dalszej części została przedstawiona metoda macierzy przejścia w celu określenia odbijalności i dyspersji grupowego opóźnienia danej struktury. Dodatkowo pokazano metodę działania funkcji celu. Następnie zaprezentowano trzy podejścia przedstawienia danych na których operuje algorytm rojowy, jak i wyniki obliczeń na bazie tych podejść. Najlepsze wyniki otrzymano na podstawie pierwszego, najprostszego podejścia zakładającego wykorzystanie grubości warstw, lecz wymaga ono znajomość dobrej struktury do rozpoczęcia obliczeń. Obserwując resztę wyników można zauważyć, że metoda faktycznie działa i program dąży do uzyskania struktur lepszych.

## **Abstract**

Swarm algorithms were used to control the dispersion in DBR mirrors in order to obtain ultrashort pulses. At the beginning, swarm algorithms were presented and tested if they really work. After that, the transition matrix method for determining the reflectivity and group delay dispersion for the given structure was shown, as well as the goal function. Then, 3 different approaches for representing the working variable of the algorithm for resolving the problem of finding DBR mirrors with possibly smallest value of group delay dispersion were presented, along with the corresponding results. The best results were observed in the first and easiest approach, that consisted in using layers' thickness, however it requires to start the calculations from a known, well optimised, structure. Observing the remaining results, it is clear that this method is working and the algorithm tends to generate better structures.



# Spis treści

<b>Symbole przyjete w pracy</b>	<b>9</b>
<b>1 Wprowadzenie</b>	<b>11</b>
<b>2 Algorytmy rojowe</b>	<b>14</b>
2.1 Algorytmy rojowe w literaturze . . . . .	14
2.2 Algorytm zastosowany w programie . . . . .	16
2.3 Przykład zastosowania na funkcji prostej . . . . .	20
<b>3 Implementacja</b>	<b>23</b>
3.1 Metoda macierzy przejścia . . . . .	23
3.2 Funkcja celu . . . . .	26
3.3 Przedstawienie danych wejściowych i wyniki . . . . .	28
3.3.1 Losowanie grubości warstw . . . . .	29
3.3.2 Losowanie parametru wykorzystującego warunek Bragg'a . . . . .	31
3.3.3 Losowanie 2 parametrów korzystając z warunku Bragg'a i zależności między grubościami warstw . . . . .	34
3.4 Zastosowane technologie informatyczne . . . . .	37
<b>4 Podsumowanie</b>	<b>39</b>
<b>Bibliografia</b>	<b>41</b>

# Spis rysunków

1.1	Współczynnik odbicia i GDD w przypadku rozwiązania analitycznego . . . . .	13
2.1	Ilustrowany schemat działania algorytmu rojowego . . . . .	17
2.2	Przebieg algorytmu rojowego zastosowanego w programie . . . . .	19
2.3	Funkcja Ackleya . . . . .	20
3.1	Schemat zwierciadła DBR . . . . .	24
3.2	Warunki w funkcji celu . . . . .	27
3.3	Dispersyjne własności struktury 1 (gorszej) . . . . .	27
3.4	Dispersyjne własności struktury 2 (lepszej) . . . . .	28
3.5	Wyniki obliczeń w przypadku losowania grubości warstw przy rozpoczęciu obliczeń od losowo wygenerowanej struktury . . . . .	29
3.6	Wyniki obliczeń w przypadku losowania grubości warstw przy rozpoczęciu obliczeń od struktury 1 (gorszej) . . . . .	30
3.7	Wyniki obliczeń w przypadku losowania grubości warstw przy rozpoczęciu obliczeń od struktury 2 (lepszej) . . . . .	30
3.8	Wyniki obliczeń w przypadku losowania grubości warstw przy rozpoczęciu obliczeń od struktury 2 (lepszej) i $N = 100000$ pszczół . . . . .	31
3.9	Wyniki obliczeń w przypadku losowania parametru wykorzystującego warunek Bragg'a przy wykorzystaniu współczynników załamania ze struktury 1 . . . . .	33

3.10 Wyniki obliczeń w przypadku losowania parametru wykorzystującego warunek Bragg'a przy wykorzystaniu współczynników załamania ze struktury 2 . . . . .	33
3.11 Wyniki obliczeń w przypadku losowania parametru wykorzystującego warunek Bragg'a przy wykorzystaniu współczynników załamania ze struktury 2 . . . . .	34
3.12 Wyniki obliczeń w przypadku losowania 2 parametrów przy wykorzystaniu współczynników załamania ze struktury 1 . . . . .	36
3.13 Wyniki obliczeń w przypadku losowania 2 parametrów przy wykorzystaniu współczynników załamania ze struktury 2 . . . . .	36
3.14 Wyniki obliczeń w przypadku losowania 2 parametrów przy wykorzystaniu współczynników załamania ze struktury 2 . . . . .	37

# **Spis tabel**

2.1 Wyniki dla funkcji Ackleya . . . . .	21
3.1 Weryfikacja działania funkcji celu . . . . .	28

# Symbole przyjęte w pracy

Jeśli w tekście nie wykazano inaczej, stosowane symbole należy rozumieć jako:

$N$  - liczba pszczół zwiadowców

$n$  - liczba rozwiązań

$n_{best}$  - liczba najlepszych rozwiązań

$n_{qual}$  - liczba rozwiązań dobrych

$n_{left}$  - liczba rozwiązań pozostałych (nie zdefiniowanych jako dobre lub najlepsze)

$w_{best}$  - współczynnik określający procent rozwiązań uważanych za najlepsze względem wszystkich rozwiązań

$w_{qual}$  - współczynnik określający procent rozwiązań uważanych za dobre względem wszystkich rozwiązań

$w_{better}$  - współczynnik określający ile więcej pszczół będzie przypisana do przeszukiwania sąsiedztwa rozwiązań najlepszych

$d_{near}$  - wartość określająca odległość przeszukiwania od rozwiązania pierwotnego (sąsiedztwo)

$d_{far}$  - wartość określająca odległość przeszukiwania od danych początkowych (globalny obszar przeszukiwania)

GDD - dyspersja grupowego opóźnienia (ang. Group Delay Dispersion)

R - współczynnik odbicia

$c_R$  - waga funkcji celu związana z wartością R

---

$c_{avGDD}$  - waga funkcji celu związana ze średnią wartością GDD

$c_{devGDD}$  - waga funkcji celu związana z odchyleniem standardowym od średniej GDD

$c_{ptpGDD}$  - waga funkcji celu związana z różnicą między ekstremami GDD

$n_{warstw}$  - liczba warstw w zwierciadle

# Rozdział 1

## Wprowadzenie

Ultraszybkie lasery znajdują wiele zastosowań w przeróżnych dziedzinach - informatyka, fizyka, medycyna... Dla przykładu, takie lasery są wykorzystywane w kryptografii kwantowej czy przy systemach telekomunikacji o szybkości 10 Gb/s lub wyższej [1]. W przypadku kryptografii kwantowej wiązka z ultraszybkiego lasera jest "wstrzykiwana" do wiązki innego lasera, dzięki temu ograniczone są odchylenia sygnału od jego ustalonych charakterystyk (ang. time jitter) i pojawia się możliwość szybkiego przekazywania zaszyfrowanej informacji (z prędkością 1 Mb/s) [2]. Natomiast w przypadku systemów telekomunikacyjnych, ultraszybkie lasery są używane ze względu na małą ilość zakłóceń w emitowanej wiązce w porównaniu do laserów ciągłych [1].

Pierwsze ultrakrótkie impulsy, o długości rzędu pikosekund, w laserach na ciele stałym zostały wyemitowane dzięki pasywnie synchronizującym mody laserowi Nd:szkło, zaprojektowanemu przez De Marię i współpracowników w roku 1966, 6 lat po zaprezentowaniu pierwszego lasera w historii [3]. Przełom w tworzeniu ultrakrótkich impulsów w laserach na ciele stałym nastąpił w 1991 roku razem z odkryciem samosynchronizacji modów w laserze Ti:szafir przez grupę Sibbett [4]. To zjawisko jest dziś znane w literaturze jako synchronizacja metodą samoogniskowania Kerra (ang. Kerr lens mode locking - KLM) i polega ono na samonasyceniu się absorbenta, którym jest kryształ ośrodku czynnego. Dzięki KLM, praktycznie w każdym laserze na ciele stałym zachodzi samorzutna synchronizacja modów, a co za tym idzie generacja impulsów piko- i femtosekundowych bez potrzeby stosowanie urządzeń modulujących, gdyż modulatorem jest sam ośrodek czynny. Jednak, w celu osiągnięcia impulsów stabilnych, powtarzalnych i ze ścisłe zdefiniowanym kształtem należy zastosować urządzenie kontrolujące dyspersję grupowego opóźnienia (ang. Group Delay Dispersion - GDD) [5], która to jest zde-

finowana jako druga pochodna zmiany fazy  $\phi$  fali przechodzącej przez strukturę po częstotliwości  $\omega$ :

$$GDD = \frac{d^2\phi}{d\omega^2}, \quad (1.1)$$

i charakteryzuje wpływ zjawiska dyspersji na czas przejścia przez strukturę. Im wartość GDD jest większa tym ten wpływ jest większy i fala jest opóźniana.

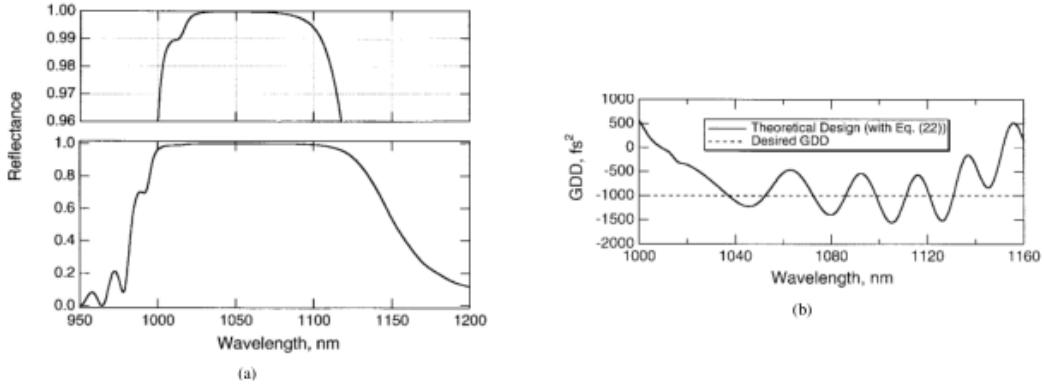
Dalszy rozwój laserów o ultrakrótkich impulsach opierał się na zaprojektowaniu tego urządzenia. W taki oto sposób pojawiły się zwierciadła ze zmiennymi grubościami warstw (ang. chirped mirrors) po raz pierwszy zaprezentowane przez grupę TU-Vienna, aby chwilę później być rozbudowane w ramach koncepcji zwierciadeł ze zmieniątą grubością pary warstw jak i zmienną względną grubością warstw w każdej parze (ang. double chirped mirror) przez ETH-Zurich [4].

Dalsze prace związane ze zwierciadłami DBR dotyczyły prób uzyskania jak najmniejszej i najstabilniejszej wartości dyspersji grupowego opóźnienia przy zachowaniu jak najwyższej odbijalności lustra. Pojawiły się prace próbujące rozwiązać sposób analitycznie, jak np. praca [6], gdzie wyprowadzono następujące równanie pozwalające określić grubości warstw na podstawie żądanej wartości GDD:

$$k_B(m) = k_B^{max} \cdot \sqrt{1 - \frac{4\pi^2}{c^2 D_0 (\pi - |\kappa_0|) (k_B^{max})^2} |m|}, \quad (1.2)$$

gdzie:  $k_B$  — lokalny numer fali Bragg'a, zdefiniowany jako  $k_B = \frac{\pi}{n_1 d_{1,m} + n_2 d_{2,m}}$ ,  $k_B^{max}$  — największy lokalny numer fali Bragg'a pozwalający otrzymać ujemne GDD,  $m$  — numer pary warstw (założono, że jest to wartość ciągła),  $c$  — szybkość światła w próżni,  $D_0$  — wartość bezwzględna żądanej wartości GDD oraz  $\kappa_0$  — współczynnik sprzężenia między warstwami w parze.

Z tak wyprowadzonego wzoru skorzystano w celu znalezienia struktury złożonej z pół-przewodników o współczynnikach załamania kolejno  $n_1 = 3,0$  i  $n_2 = 3,5$ . Ustalonej wartości bezwzględnej żądanej wartości GDD aby wynosiła  $D_0 = 1000 \text{ fs}^2$ .  $k_B^{max}$  ustalone natomiast na wartość  $2\pi/980 \text{ nm}$ . Tak zaprojektowane zwierciadło charakteryzuje się, faktycznie, wartością GDD na poziomie  $-1000 \text{ fs}^2$  przy dużej odbijalności, tak jak przedstawiono to na rysunku 1.1.



Rysunek 1.1: Współczynnik odbicia i GDD w przypadku rozwiązyania analitycznego [6]

Otrzymana wartość GDD jest niska, lecz nie jest do końca stabilna. Dlatego próbowało uzyskać jeszcze lepsze wartości tego parametru korzystając z innych podejść do problemu oraz z potęgi komputerów w celu zoptymalizowania obliczeń. W taki też sposób pojawiła się między innymi praca [7] wykorzystująca metodę macierzy przejścia, która jest szczegółowo omówiona w 3.1, w celu wyznaczenia wzorów pozwalających obliczyć współczynnik odbicia i GDD dla danego zwierciadła. Ponadto tę metodę zoptymalizowano korzystając z algorytmów genetycznych aby uzyskać jak najlepsze wyniki. Udało się wyznaczyć strukturę z wartością GDD wynoszącą około  $-3600 \text{ fs}^2$  przy zakresie długości fali  $1030 - 1050 \text{ nm}$ . Metoda ta okazała się skuteczna, lecz wyniki wciąż można próbować polepszyć. Jedną z rzeczy, która jeszcze nie została zrobiona jest sprawdzenie efektu zastosowania innego rodzaju algorytmów, m.in. algorytmów rojowych, należących do grupy algorytmów stadnych.

Dlatego też niniejsza praca przedstawia próbę otrzymania jeszcze lepszych wyników wykorzystując optymalizację obliczeń przy wykorzystaniu algorytmów rojowych. Dodatkowym celem pracy było poznanie sposobu działania tych właśnie algorytmów i sprawdzenie ich potencjału przy rozwiązywaniu podobnych problemów.

# Rozdział 2

## Algorytmy rojowe

Po ogólnym wprowadzeniu do tematu w rozdziale 1, czas przejść do omówienia zastosowanej metody. Jednym z bardziej rzucających się w oczy wyrażeń w tytule pracy jest "algorytmy rojowe" i to właśnie nimi zajmiemy się w tym rozdziale. Zostanie omówiona istota algorytmów rojowych, zarówno w literaturze jak i ostateczna wersja zastosowana w programie, oraz, na koniec, zostanie przedstawiony przykład implementacji na funkcji dwóch zmiennych.

### 2.1 Algorytmy rojowe w literaturze

Algorytmy rojowe należą do rodziny algorytmów stadnych. Ich główną ideą jest wykorzystanie sposobu zachowywania się pszczół zwiadowców podczas zbierania nektaru. Pszczoły, zgodnie z [8], wyruszają na łąkę w poszukiwaniu kwiatów z dużą ilością nektaru. Następnie wracają do ula i wykonują tak zwany taniec pszczeli podczas którego dochodzi do wymiany informacji między pszczołami na temat lokalizacji kwiatów o dużej zawartości nektaru. To pozwala na wysłanie pozostałych pszczół (czyli tych, które nie znalazły dobrych źródeł nektaru) do najlepszych lokalizacji. Im źródło zawiera więcej nektaru, tym więcej pszczół kieruje się do niego.

W literaturze ([8], [9], [10]) można znaleźć różne opisy funkcjonowania tych algorytmów. Podejście różni się od źródła do źródła, lecz ogólny przebieg algorytmu rojowego jest taki sam i przedstawia się następująco:

0. inicjalizacja parametrów przekazanych programowi przy uruchomieniu:

$w_{best}$  - współczynnik określający procent rozwiązań uważanych za najlepsze względem wszystkich rozwiązań,

$w_{qual}$  - współczynnik określający procent rozwiązań uważanych za dobre względem wszystkich rozwiązań,

$w_{better}$  - współczynnik określający w jaki sposób pszczoły podzielą się na rozwiązania dobre i najlepsze. Jest to wartość z zakresu  $(0, 1)$  i im jest większa tym więcej pszczół zostanie przydzielonych do przeszukiwania obszaru wokół rozwiązań najlepszych,

$d_{near}$  - wartość określająca odległość przeszukiwania od rozwiązań pierwotnego (sąsiedztwo),

$N$  - liczba pszczół zwiadowców biorąca udział w poszukiwaniu rozwiązania,

oraz zmiennych:

$n$  - liczba wszystkich rozwiązań,

$n_{best}$  - liczba najlepszych rozwiązań (w chwili początkowej= 0) określona współczynnikiem  $w_{best}$ ,

$n_{qual}$  - liczba rozwiązań dobrych (w chwili początkowej= 0) określona współczynnikiem  $w_{qual}$ ,

$n_{left}$  - liczba rozwiązań pozostałych (nie zdefiniowanych jako dobre lub najlepsze) (w chwili początkowej= 0),

1. wygenerowanie początkowych  $N$  rozwiązań,
2. ocena znalezionych rozwiązań przy użyciu funkcji celu, podział ich na kategorie (najlepsze, dobre i pozostałe lub dobre i pozostałe),
3. przeszukiwanie sąsiedztwa najlepszych znalezionych rozwiązań w promieniu  $d_{near}$  od danego rozwiązania,
4. wybranie najlepszego rozwiązania z każdej przeszukiwanej lokalizacji,
5. zapisanie najlepszego rozwiązania,
6. generacja nowej populacji rozwiązań w zadanym obszarze, uwzględniając te znalezione w etapie 3, tak aby ich liczba osiągnęła  $N$ ,

7. powrót do etapu 2 lub zakończenie działania programu w przypadku spełnienia warunku stopu.

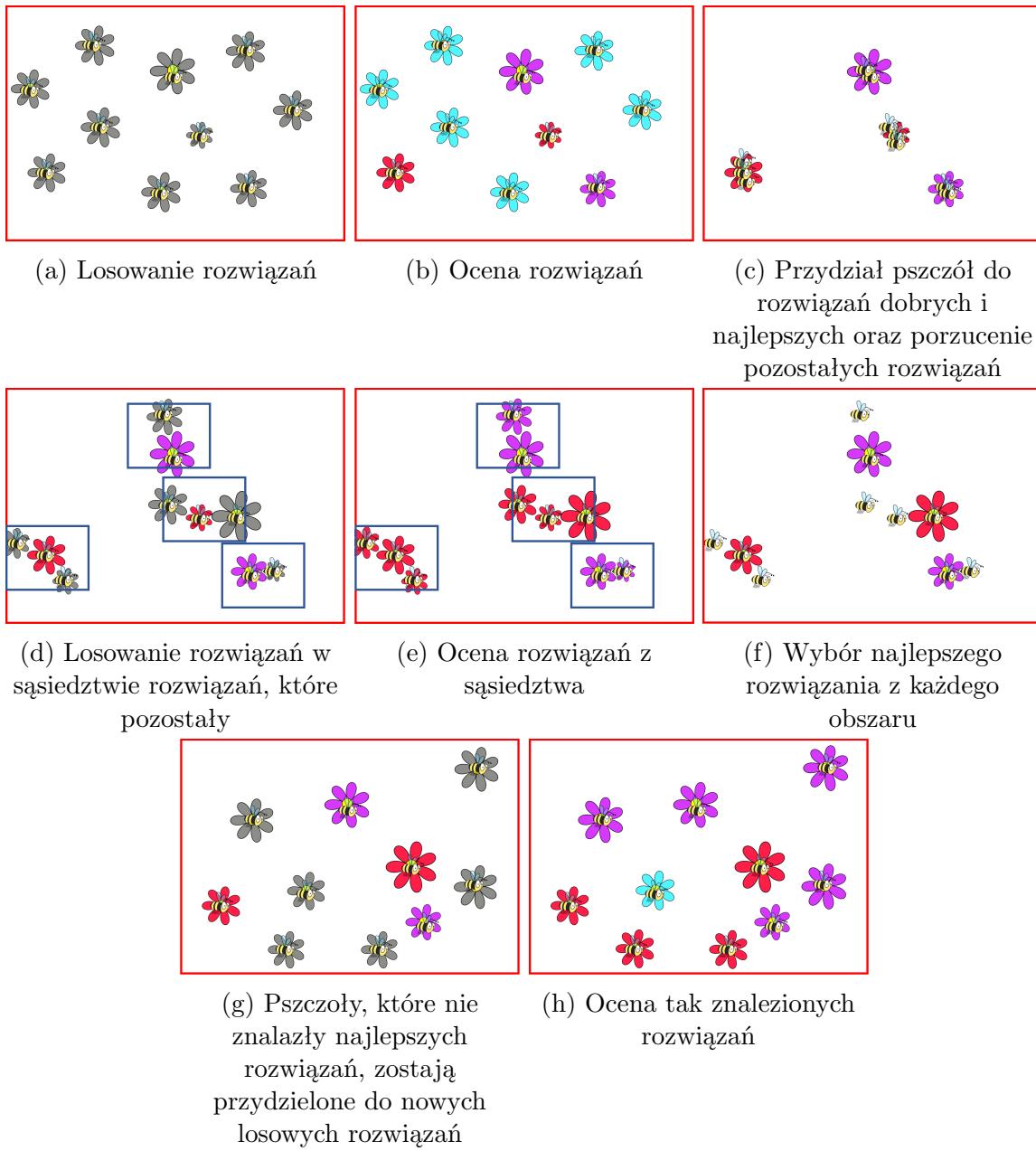
W celu lepszego zilustrowanie działania algorytmu zostały wykonane schematy na rysunku 2.1. Przebieg przedstawiony tam odpowiada dokładnie temu opisanemu w [8].

Wszelkie różnice pojawiają się w szczegółowym przebiegu poszczególnych etapów. Pierwsza taka zauważalna różnica pojawia się w etapie 2, gdzie niektóre źródła ([8] i [10]) proponują podział ocenionych rozwiązań na dobre i najlepsze na podstawie parametrów  $w_{best}$  i  $w_{qual}$ . W [8] te parametry oznaczają wartość funkcji celu od jakiej dane rozwiązanie jest rozumiane jako dobre lub elitarne, natomiast w [10] są to parametry które określają procent rozwiązań z danej kategorii względem wszystkich rozwiązań. Inne źródła ([9]) proponują posortowanie rozwiązań od najlepszego do najgorszego bez rozróżniania najlepszych i elitarnych. Następnie, w etapie 3, przy przeszukiwaniu rozwiązań, w [8] i [10] przydziela się odpowiednią ilość pszczół zwiadowców do przeszukiwania sąsiedztwa danego rozwiązania w zależności od rangi rozwiązania (elitarne bądź dobre) zgodnie z parametrem  $w_{better}$ . Ten parametr ilustruje stosunek ilości pszczół przypisanych do przeszukiwania sąsiedztwa rozwiązań dobrych i elitarnych. Natomiast w [9], biorąc pod uwagę brak podziału na dobre i elitarne, pszczoły są przydzielane w równych proporcjach do rozwiązań, które zostały wyznaczone do przeszukania ich sąsiedztwa. Ponadto, etap 5 jest pominięty w [8], ale nie odgrywa on szczególnie ważnej roli, gdyż najlepszy wynik zawsze przechodzi do następnej iteracji i zawsze istnieje możliwość jego odczytania.

Z dodatkowych różnic można wspomnieć wprowadzenie długości życia rozwiązania w [10]. Ta długość życia miałaby pozwolić na wyeliminowanie rozwiązań, które będą się powtarzać przez określoną wcześniej liczbę iteracji.

## 2.2 Algorytm zastosowany w programie

Algorytm zastosowany w tejże pracy przeszedł parę delikatnych transformacji przez cały czas sporządzania finalnego modelu wartości dyspersyjnych zwierciadła DBR. Jako algorytm wejściowy został zastosowany ten opisany w [8]. Na początku funkcja celu została zaprojektowana tak, aby zwracała wartości z zakresu (0,1). Rozwiązania elitarne i dobre były ustalane na podstawie porównania wartości zwróconej przez funkcję celu z dwoma parametrami algorytmu, określającymi próg, od którego rozwiązania



Rysunek 2.1: Ilustrowany schemat działania algorytmu rojowego zgodnie z [8].

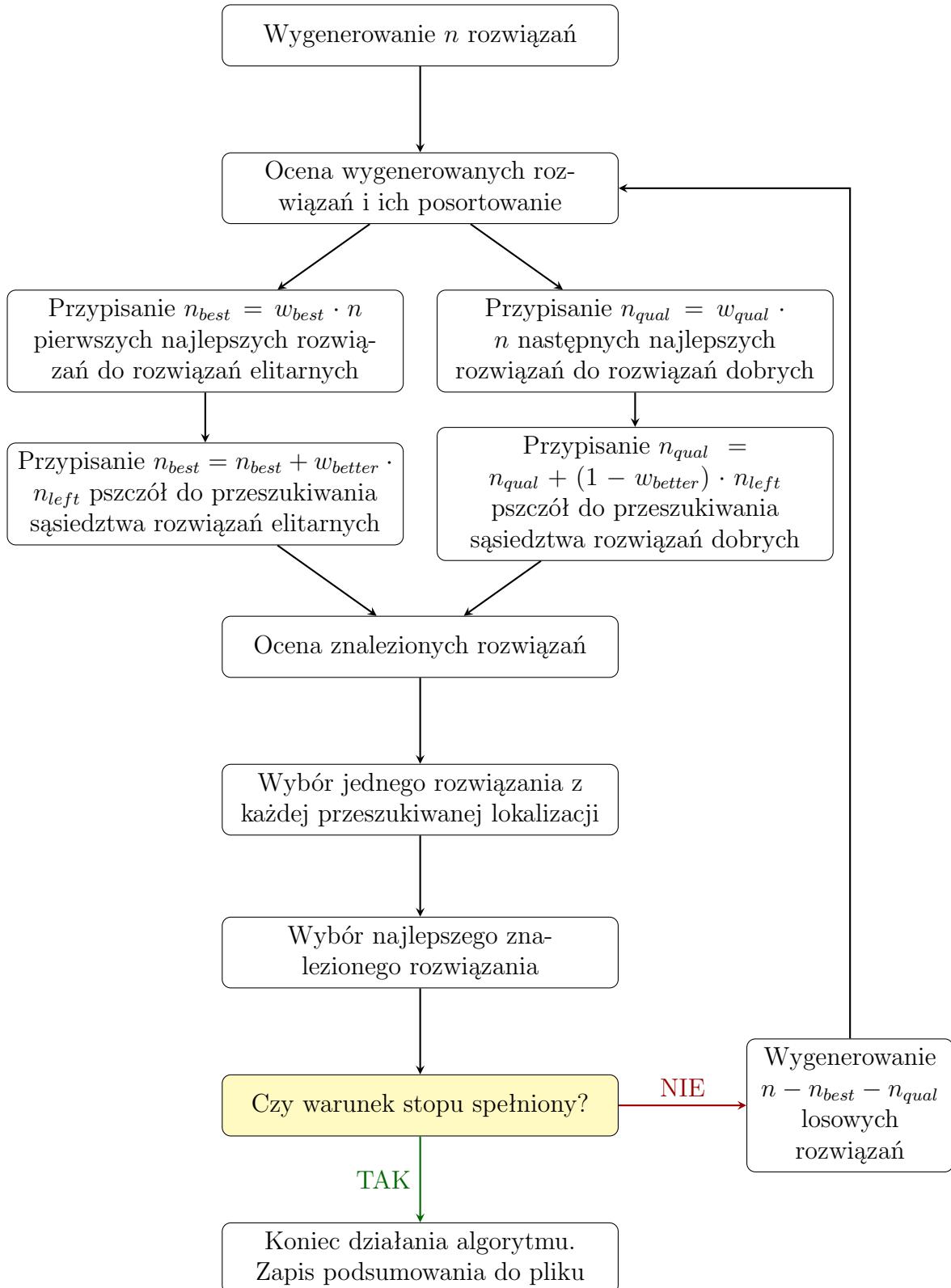
Oznaczenia: kwiatek czerwony — rozwiązanie najlepsze, kwiatek fioletowy — rozwiązanie dobre, kwiatek błękitny — rozwiązanie z grupy pozostałych, czerwony prostokąt — obszar losowania rozwiązań, niebieski prostokąt — obszar sąsiedztwa danego rozwiązania, rozmiar kwiatka świadczy o jakości rozwiązania w swojej kategorii, tj. im kwiatek większy tym rozwiązanie lepsze.

jest uznawane za dobre i próg, od którego rozwiązanie jest uznawane za elitarne. Dla tak dobranych rozwiązań, w sposób losowy, były generowane rozwiązania z sąsiedztwa danego rozwiązania. Po etapie 5 polegającym na zapisaniu najlepszego rozwiązania, algorytm sprawdzał czy suma rozwiązań dobrych i elitarnych nie przekroczyła danego progu będącego parametrem algorytmu, jeśli tak, wówczas próg minimalnej wartości funkcji celu dla rozwiązań dobrych i elitarnych był zwiększany oraz rozmiar sąsiedztwa rozwiązań był zwiększany. Dodanie tego etapu wynikało z chęci przyspieszenia dążenia algorytmu do rozwiązania i nie występuje on w żadnym z wymienionych źródeł.

Po wielu testach zauważono, iż algorytm napotyka problemy przy generowaniu rozwiązań spełniających próg rozwiązań dobrych i elitarnych. Większość rozwiązań otrzymywała ocenę równą零. Problem polegał na funkcji celu i trudności w otrzymaniu oceny umożliwiającej rozróżnienie jakości wszystkich wygenerowanych rozwiązań, przy zachowaniu skali oceny w zakresie (0,1). Ten fakt wynikał z tego, iż cel, który program miałby osiągnąć nie mógł zostać określony w sposób jednoznaczny, aby móc ustalić maksymalną wartość rozwiązań jakie zostaną wygenerowane. Dlatego też funkcja celu została zmodyfikowana w taki sposób aby wystawiać ocenę rozwiązania bez korzystania z jakiekolwiek skali. Tak zbudowana funkcja celu jest w stanie rozróżnić między jakością każdego wygenerowanego rozwiązania, lecz bez możliwości jednoznacznego określenia maksymalnej i minimalnej oceny jaką może wystawić. Ten fakt spowodował konieczność modyfikacji algorytmu. Zamiast wyznaczania rozwiązań elitarnych i dobrych na podstawie progów, rozwiązania są teraz sortowane na podstawie wartości funkcji celu od najlepszego do najgorszego, tak jak w [10]. Następnie, na podstawie parametrów  $w_{best}$  i  $w_{qual}$  określającymi procent rozwiązań elitarnych i dobrych względem wszystkich rozwiązań, rozwiązania są kategoryzowane. Dla przykładu, jeżeli bierzemy pod uwagę 100 rozwiązań i parametry  $w_{best}$  i  $w_{qual}$  wynoszą kolejno 0,1 i 0,2, po posortowaniu od najlepszego do najgorszego pierwsze 10 rozwiązań jest uznawane za najlepsze, natomiast 20 następnych rozwiązań jako dobre. Dalej przebieg wygląda tak samo jak w źródle [8].

W algorytmie dodatkowo dopuszczono generowanie rozwiązań początkowych w odległości  $d_{far}$  od rozwiązania początkowego załadowanego z zewnątrz programu. Warto zauważać, że  $d_{far}$  jest dużo większe od  $d_{near}$ .

Finalny algorytm zastosowany w programie został zaprezentowany na rysunku 2.2.



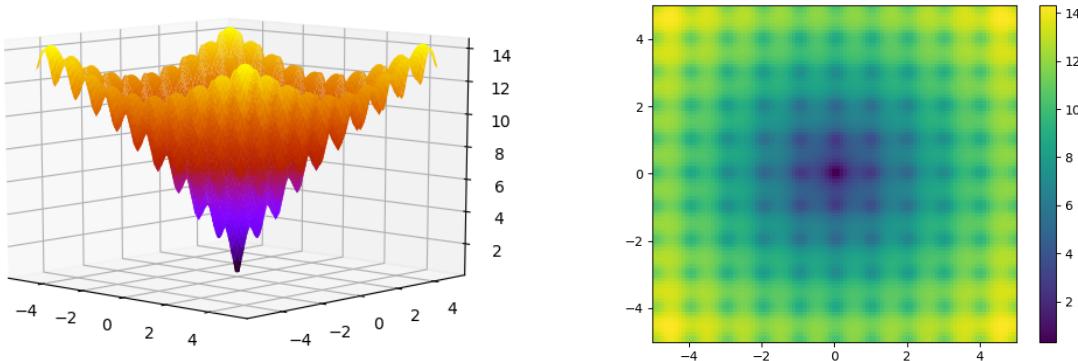
Rysunek 2.2: Przebieg algorytmu rojowego zastosowanego w programie

## 2.3 Przykład zastosowania na funkcji prostej

W celu sprawdzenia efektywności algorytmu zostały wykonane testy na prostej funkcji dwóch zmiennych. Została wybrana funkcja Ackleya o wzorze:

$$f(x,y) = -20 \exp \left[ -0.2 \sqrt{0.5 (x^2 + y^2)} \right] - \exp [0.5 (\cos 2\pi x + \cos 2\pi y)] + e + 20, \quad (2.1)$$

ze względu na to iż charakteryzuje się falistym wykresem z dużą ilością minimów lokalnych o wartościach malejących przy oddalaniu się od punktu  $(0,0)$ , w którym to też znajduje się minimum globalne funkcji o wartości 0. Dla lepszego zobrazowania tej charakterystycznej struktury, wykres funkcji Ackleya został zaprezentowany na rysunku 2.3.

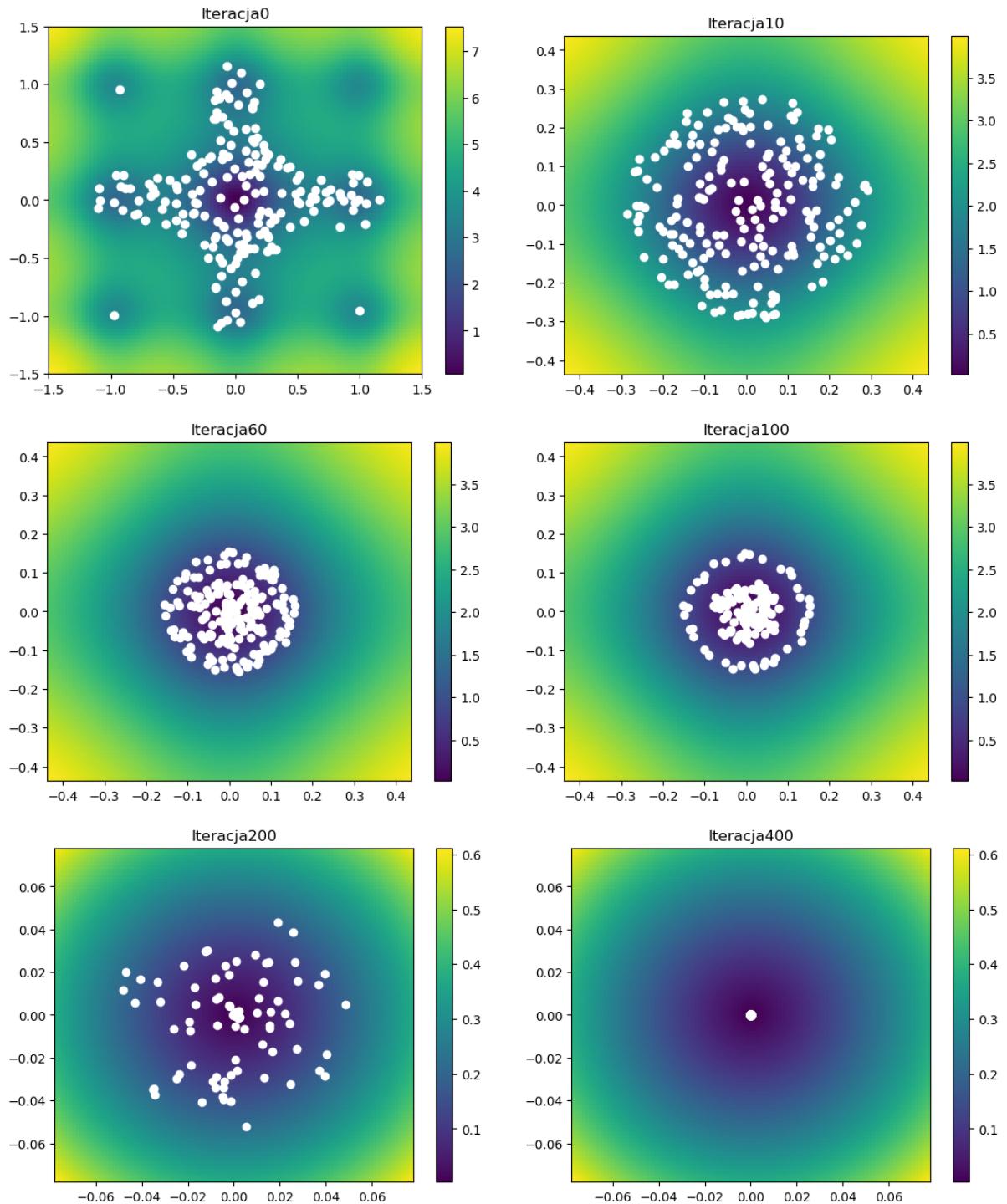


Rysunek 2.3: Funkcja Ackleya

Oprócz doboru badanej funkcji, bardzo istotnym elementem było określenie funkcji celu. W tym wypadku funkcja celu została zaprogramowana tak, że zwraca wartości z przedziału  $\langle 0, 1 \rangle$  na podstawie modułu różnicy wartości funkcji od wartości 0 w punkcie  $(0,0)$ . Otrzymana liczba jest dzielona przez największą wartość jaką może przyjąć funkcja w danych przedziale (w tej symulacji dla  $x$  i  $y$  w przedziale  $\langle -1,5, 1,5 \rangle$  maksymalna wartość funkcji wynosi 8) i tak otrzymana liczba jest odejmowana od 1 aby otrzymać skalę ocen w taki sposób, że im punkt bliższy rozwiążaniu tym ocena wyższa.

Symulacja została wykonana dla  $N = 1000$  zwiadowców, przy następujących wartościach parametrów:  $d_{near} = 0,0005$ ,  $w_{qual} = 0,20$ ,  $w_{best} = 0,10$ ,  $w_{better} = 0,6$ . Wyniki obliczeń zostały przedstawione w tabeli 2.1.

Tabela 2.1: Wyniki dla funkcji Ackleya



Można zauważać na podstawie wyników z tabeli 2.1, że z iteracji na iterację obszar poszukiwań zawsze się do coraz to bliższego sąsiedztwa poszukiwanego rozwiązania,

dokładnie tak jak to było zamierzone. Dodatkowo otrzymany wynik po 400 iteracjach wynosi  $6,32 \cdot 10^{-6}$  w punkcie  $(-0,36 \cdot 10^{-6}, 2,21 \cdot 10^{-6})$  co jest dobrym przybliżeniem szukanego rozwiązania.

Podsumowując algorytmy rojowe wydają się być dobrym narzędziem do rozwiązywania problemów w których celem jest odnalezienie minimum bądź maksimum danej funkcji. Algorytm wyznaczania struktury z najlepszymi właściwościami dyspersyjnymi jest właśnie poniekąd taką funkcją, której minimum będziemy próbować znaleźć w następnym rozdziale.

# Rozdział 3

## Implementacja

Po zrozumieniu istoty algorytmów rojowych przyszedł czas na zastosowanie tej wiedzy w praktyce. W tym rozdziale zostanie zaprezentowane w jaki sposób jest przedstawiona funkcja obliczająca własności dyspersyjne zwierciadła, jak została na jej podstawie zbudowana funkcja celu oraz jakie zostały zastosowane biblioteki i techniki programistyczne w opracowaniu tego algorytmu.

### 3.1 Metoda macierzy przejścia

Pierwszym etapem jest wyznaczenie funkcji pozwalającej na określenie wartości współczynnika odbicia  $R$  jak i GDD dla dowolnego zwierciadła DBR. Posłużymy się metodą macierzy przejścia, która polega na wyznaczeniu macierzy przejścia wektora natężenia pola elektrycznego z jednej warstwy do drugiej, a następnie, na tej podstawie, macierzy przejścia dla całej struktury.

Wiązka światłą padającą na zwierciadło DBR jest, zgodnie z istotą korpuskularnofalową światła, falą elektromagnetyczną. Pole elektryczne w każdej z warstw zwierciadła DBR ma postać:

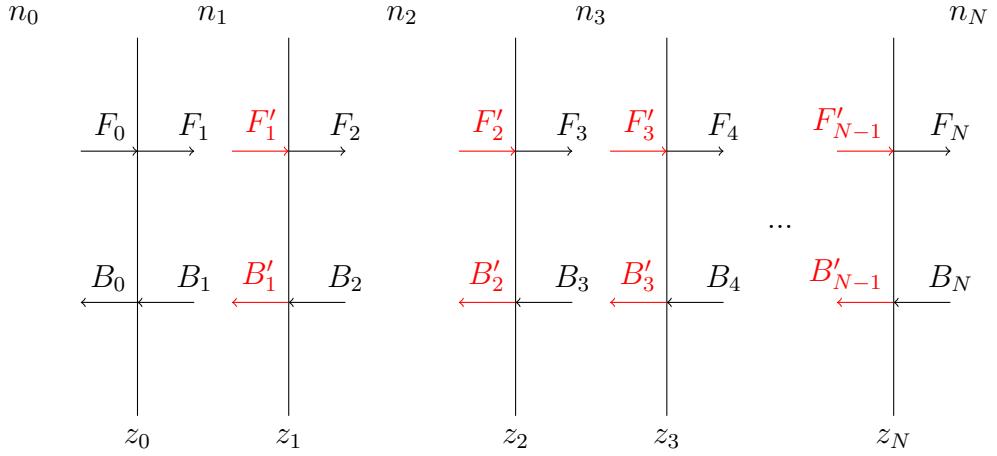
$$E_m(z) = F_m e^{-in_m \frac{\omega}{c} z} + B_m e^{in_m \frac{\omega}{c} z}, \quad (3.1)$$

gdzie  $F_m$  i  $B_m$  odpowiadają fali poruszającej się w kierunku  $z$  z kolejno zwoitem dodatnim i ujemnym.

Pochodna powyższej wartości ma postać:

$$\frac{\partial E_m}{\partial z} = -in_m \frac{\omega}{c} (F_m e^{-in_m \frac{\omega}{c} z} - B_m e^{in_m \frac{\omega}{c} z}). \quad (3.2)$$

Na rysunku 3.1 przedstawiono schemat ilustrujący rozkład sił pola elektrycznego w zwierciadle. Przyjmijmy  $F'_m = F_m e^{-i\phi_m}$  i  $B'_m = B_m e^{i\phi_m}$ , gdzie  $\phi = n_m \frac{\omega}{c} d_m$ .



Rysunek 3.1: Schemat zwierciadła DBR

Warunek ciągłości dla poszczególnych warstw można zapisać następująco:

$$\begin{cases} F'_m + B'_m = F_{m+1} + B_{m+1}, \\ n_m(F'_m - B'_m) = n_{m+1}(F_{m+1} - B_{m+1}). \end{cases} \quad (3.3)$$

Rozwiązujeając powyższy układ równań względem  $F'_m$  i  $B'_m$  otrzymujemy:

$$\begin{bmatrix} F_{m+1} \\ B_{m+1} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} (1 + \frac{n_m}{n_{m+1}}) & (1 - \frac{n_m}{n_{m+1}}) \\ (1 - \frac{n_m}{n_{m+1}}) & (1 + \frac{n_m}{n_{m+1}}) \end{bmatrix} \begin{bmatrix} F'_m \\ B'_m \end{bmatrix}. \quad (3.4)$$

Korzystając z definicji  $F'_m$  i  $B'_m$  możemy zapisać:

$$\begin{bmatrix} F_{m+1} \\ B_{m+1} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} (1 + \frac{n_m}{n_{m+1}}) & (1 - \frac{n_m}{n_{m+1}}) \\ (1 - \frac{n_m}{n_{m+1}}) & (1 + \frac{n_m}{n_{m+1}}) \end{bmatrix} \begin{bmatrix} e^{-i\phi_m} \\ e^{i\phi_m} \end{bmatrix} \begin{bmatrix} F_m \\ B_m \end{bmatrix}, \quad (3.5)$$

co ostatecznie daje:

$$\begin{bmatrix} F_{m+1} \\ B_{m+1} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} (1 + \frac{n_m}{n_{m+1}})e^{-i\phi_m} & (1 - \frac{n_m}{n_{m+1}})e^{i\phi_m} \\ (1 - \frac{n_m}{n_{m+1}})e^{-i\phi_m} & (1 + \frac{n_m}{n_{m+1}})e^{i\phi_m} \end{bmatrix} \begin{bmatrix} F_m \\ B_m \end{bmatrix}. \quad (3.6)$$

Macierz  $\begin{bmatrix} (1 + \frac{n_m}{n_{m+1}})e^{-i\phi_m} & (1 - \frac{n_m}{n_{m+1}})e^{i\phi_m} \\ (1 - \frac{n_m}{n_{m+1}})e^{-i\phi_m} & (1 + \frac{n_m}{n_{m+1}})e^{i\phi_m} \end{bmatrix}$  nazywamy macierzą przejścia i notujemy  $T_m$ .

Dla lepszej czytelności zapiszmy  $E_m = \begin{bmatrix} F_m \\ B_m \end{bmatrix}$ . Można zauważyc, że wartość natężenia pola elektrycznego dla danej warstwy zależy od wartości natężenia dla warstwy poprzedniej:

$$\begin{aligned} E_1 &= T_0 E_0, \\ E_2 &= T_1 T_0 E_0, \\ &\dots \\ E_N &= (T_N \dots T_1 T_0) E_0. \end{aligned}$$

Iloczyn poszczególnych macierzy przejścia, tj. macierz  $T = T_N \dots T_1 T_0$ , nazywamy macierzą przejścia całej struktury. Zapiszmy macierz przejścia całej struktury w sposób następujący:  $T = \begin{bmatrix} t_1 & t_2 \\ t_3 & t_4 \end{bmatrix}$ .

Korzystając z tak obliczonej macierzy przejścia można obliczyć amplitudowy współczynnik odbicia, który definiuje stosunek amplitudy fali odbitej do fali padającej. Jako, że  $t_3$  odpowiada fali odbitej, natomiast  $t_4$  fali padającej, można otrzymać wzór  $r = -\frac{t_3}{t_4}$ , oraz, na tej podstawie, współczynnik odbicia  $R = rr^*$ .

Grupowe opóźnienie GD (Group Delay) można zdefiniować jako pierwsza pochodna zmiany fazy  $\phi$  po częstotliwości  $\omega$ . Wiemy, że zmianę fazy można zapisać w postaci:

$$\phi = \arctan \left( \frac{\text{Im}(r)}{\text{Re}(r)} \right), \quad (3.7)$$

stąd GD obliczamy korzystając ze wzoru:

$$GD = \frac{d\phi}{d\omega} = \frac{\operatorname{Im}(r) \frac{d}{d\omega} \operatorname{Re}(r) - \operatorname{Re}(r) \frac{d}{d\omega} \operatorname{Im}(r)}{rr^*}, \quad (3.8)$$

natomiast GDD jest pochodną GD po częstotliwości  $\omega$ , tak jak to przedstawiono we wzorze poniżej:

$$GDD = \frac{d}{d\omega} GD. \quad (3.9)$$

Tak otrzymane wzory posłużą do sporządzenia modelu zwierciadła DBR (w postaci klasy) umożliwiającego obliczenie GDD i R dla danego zwierciadła.

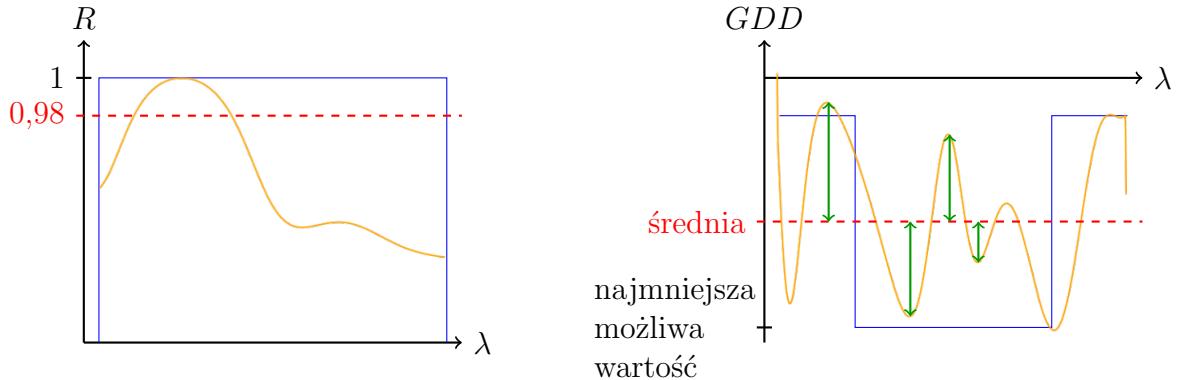
## 3.2 Funkcja celu

Mając już do dyspozycji wartości GDD i R, należy być w stanie je ocenić przy pomocy funkcji celu, która zostanie użyta w algorytmie rojowym. Celem pracy jest osiągnięcie wartości współczynnika odbicia jak najbliższej jedności, oraz możliwie najmniejszej wartości GDD przy zachowaniu minimalnej zmiany tej wartości w zadany zakresie długości fali. Dlatego funkcja celu powinna brać pod uwagę następujące aspekty:

1. czy wartość R jest bliska jedności poprzez zliczenie ilości wystąpień wartości większych niż 0,98 na danym przedziale długości fali,
2. średnia wartości GDD,
3. wahanie wartości GDD poprzez obliczenie odchylenia standardowego od średniej oraz różnicy między największą i najmniejszą wartością jaką przyjmuje GDD w zadanym przedziale,

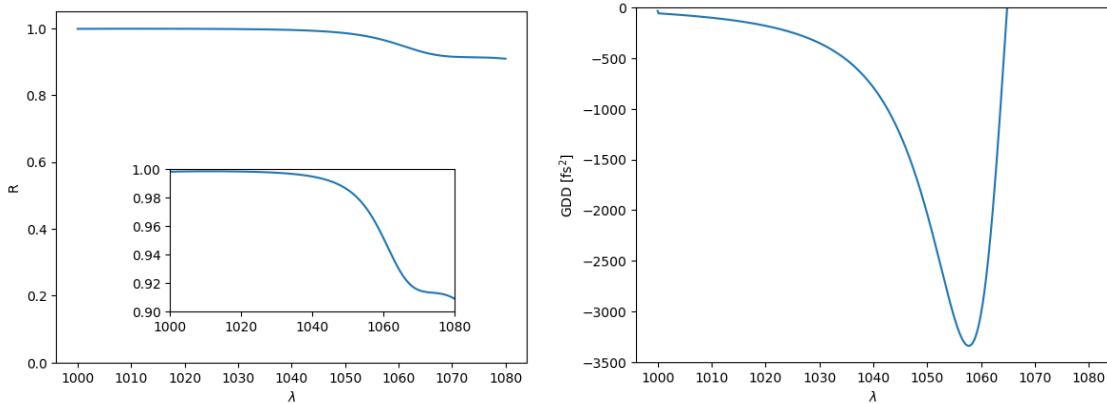
warunki te pokazano na rysunku 3.2.

Do każdego tak ustalonego parametru jest przypisywana waga, która określa w jakim stopniu dany parametr wpływa na wynik końcowy. Im dany parametr jest bliżej żądanej wartości tym bardziej funkcja celu nagradza aktualne rozwiązanie. Otrzymujemy więc 4 wagi, kolejno:  $c_R$ ,  $c_{avGDD}$ ,  $c_{devGDD}$ ,  $c_{ptpGDD}$ .



Rysunek 3.2: Warunki w funkcji celu

Legenda: niebieska krzywa — funkcja oczekiwana, pomarańczowa krzywa — funkcja rzeczywista, zielone strzałki — odchylenie punktu od wartości średniej.



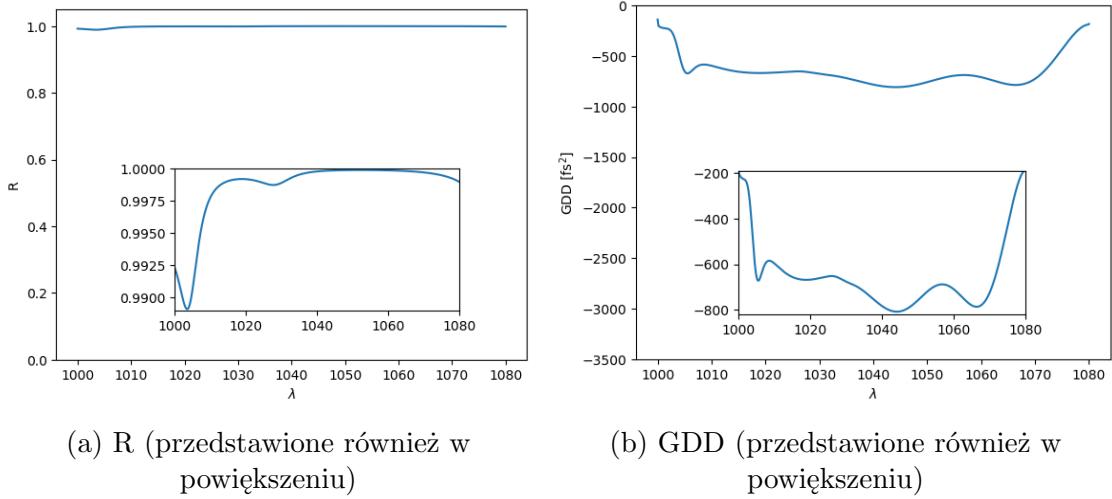
(a) R (przedstawione również w powiększeniu)

(b) GDD

Rysunek 3.3: Dyspersyjne własności struktury 1 z [11] (gorszej)

W celu sprawdzenia poprawności tak zdefiniowanej funkcji celu, wzięto pod uwagę dwie struktury z wyraźnie różniącym się R i GDD. Dyspersyjne własności pierwszej struktury (pochodzącej z [11]) przedstawiono na rysunku 3.3, natomiast drugiej (pochodzącej z [7]) na 3.4. Można łatwo zauważyć, że w przypadku obu struktur odbijalność jest na prawidłowym poziomie, natomiast GDD w przypadku struktury 1 charakteryzuje się wyraźnym pikiem w granicach 1055 nm. Struktura 2 natomiast charakteryzuje się GDD prawie, że stałym na poziomie  $800 \text{ fs}^2$ . Z pełną pewnością można stwierdzić, że struktura 2 jest lepsza od struktury 1.

Wykonano obliczenia dla dwóch różnych wartości wag i otrzymano wyniki przedstawione w tabeli 3.1.



Rysunek 3.4: Dyspersyjne własności struktury 2 z [7] (lepszej)

Tabela 3.1: Weryfikacja działania funkcji celu

$c_R$	$c_{avGDD}$	$c_{devGDD}$	$c_{ptpGDD}$	Wartość funkcji celu	
				Struktura 1 (gorsza)	Struktura 2 (lepsza)
0,5	2	10	1	69,596	459,668
100000	70	1	0	4583.742	50419.277

Obserwując wyniki z tabeli 3.1 widać wyraźnie, że struktura 2 dostała znacznie lepszą ocenę niż struktura 1, która została uznana za gorszą na podstawie obserwacji wykresów z rysunków 3.3 i 3.4. Więc funkcja celu działa w sposób prawidłowy i można jej użyć w algorytmie rojowym.

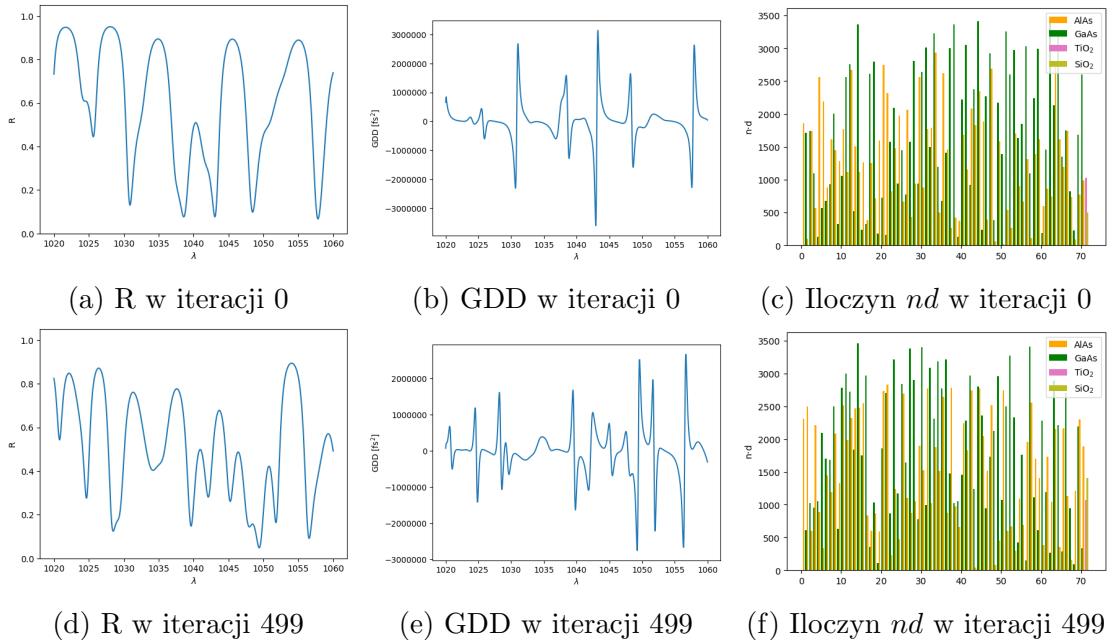
### 3.3 Przedstawienie danych wejściowych i wyniki

W celu uproszczenia problemu, założono, że zwierciadło jest zbudowane ze stałej liczby naprzemiennych warstw dwóch materiałów półprzewodnikowych: arsenku glinu AlAs oraz arsenku galu GaAs o współczynnikach załamania światła kolejno  $n_1 = 2,956$  i  $n_2 = 3,489$ , które są również stałe podczas działania algorytmu. Dlatego też wystarczające będzie przedstawienie danych wejściowych w postaci grubości warstw do zastosowania algorytmu rojowego. Wprowadza to dużą losowość uzyskanych wyników, dlatego wykonano również obliczenia przedstawiając dane w postaci zmiennej wynikającej z warunku Bragg'a. Ponadto, sprawdzono wpływ przedstawienia danych w postaci dwóch parametrów: zmiennej wynikającej z warunku Bragg'a oraz zależności między

grubościami warstw w parze. W dalszej części tekstu zostały przedstawione dokładne zależności między warstwami jakie zostały wzięte pod uwagę oraz wyniki jakie uzyskano dla każdego omawianego przypadku.

### 3.3.1 Losowanie grubości warstw

W pierwszej kolejności zastosowano bezpośrednio grubości warstw struktury złożonej ze 122 powłok kolejno GaAs i AlAs. Pierwsze  $N$  rozwiązań jest generowane losowo w przedziale  $\langle 5 \text{ nm}, 1000 \text{ nm} \rangle$ . Otrzymane tak wyniki przedstawiono na rysunku 3.5.

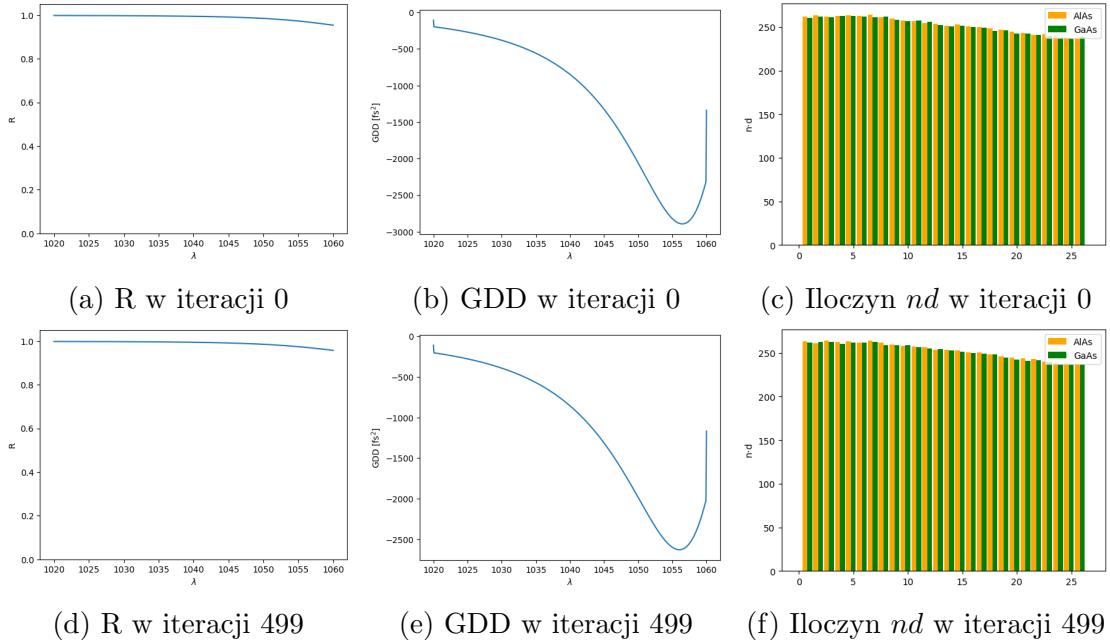


Rysunek 3.5: Wyniki obliczeń w przypadku losowania grubości warstw przy rozpoczęciu obliczeń od losowo wygenerowanej struktury

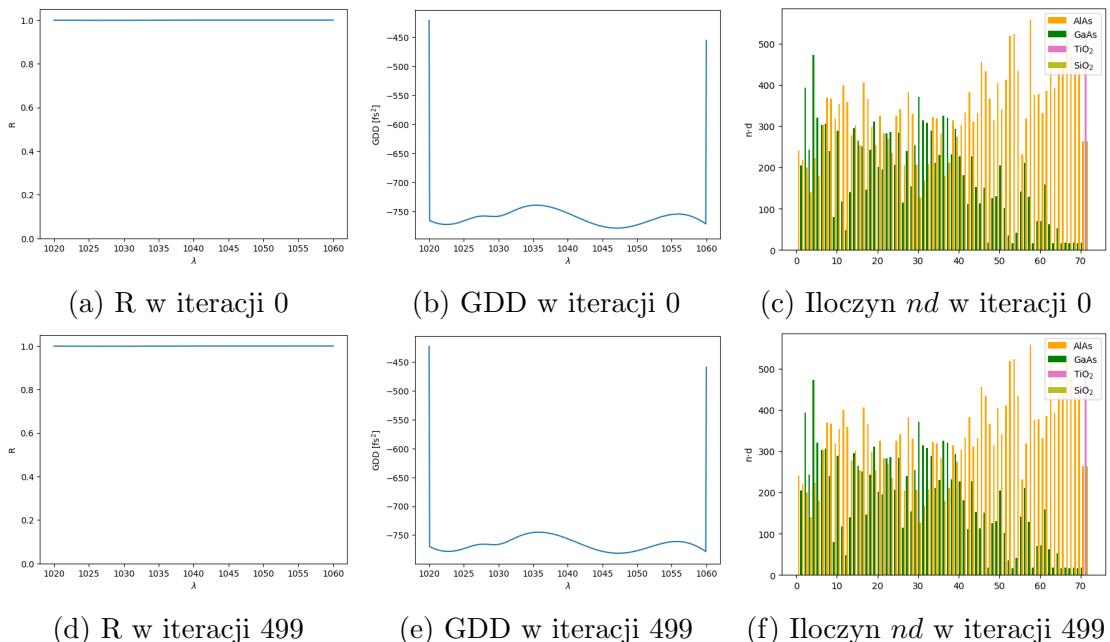
Wartości parametrów:  $N = 1000$ ,  $d_{near} = 0,005$ ,  $w_{qual} = 0,40$ ,  $w_{best} = 0,20$ ,  
 $w_{better} = 0,8$ ,  $p_1 = 0,5$ ,  $p_2 = 2$ ,  $p_3 = 10$ ,  $p_4 = 1$

Można zauważyc, że wyniki wyglądają bardzo źle, ale widać jednocześnie zmianę wynikowej struktury co świadczy o działaniu programu. Można wywnioskować, że rozpoczętając obliczenia w sposób zupełnie losowy jest bardzo mało prawdopodobne znalezienie rozwiązań dobrych. Dlatego postanowiono generować pierwsze  $N$  rozwiązań generując rozwiązania z dalekiego sąsiedztwa  $d_{far}$  struktury już wcześniej znanej. W tym celu zostały wykorzystane struktury 1 i 2 omawiane w podrozdziale 3.2. Wyniki zostały przedstawione na rysunkach 3.6 i 3.7. Parametry zostały dobrane w identyczny

sposób jak w przypadku rysunku 3.5, natomiast wartość parametru  $d_{far}$  wynosi 50.



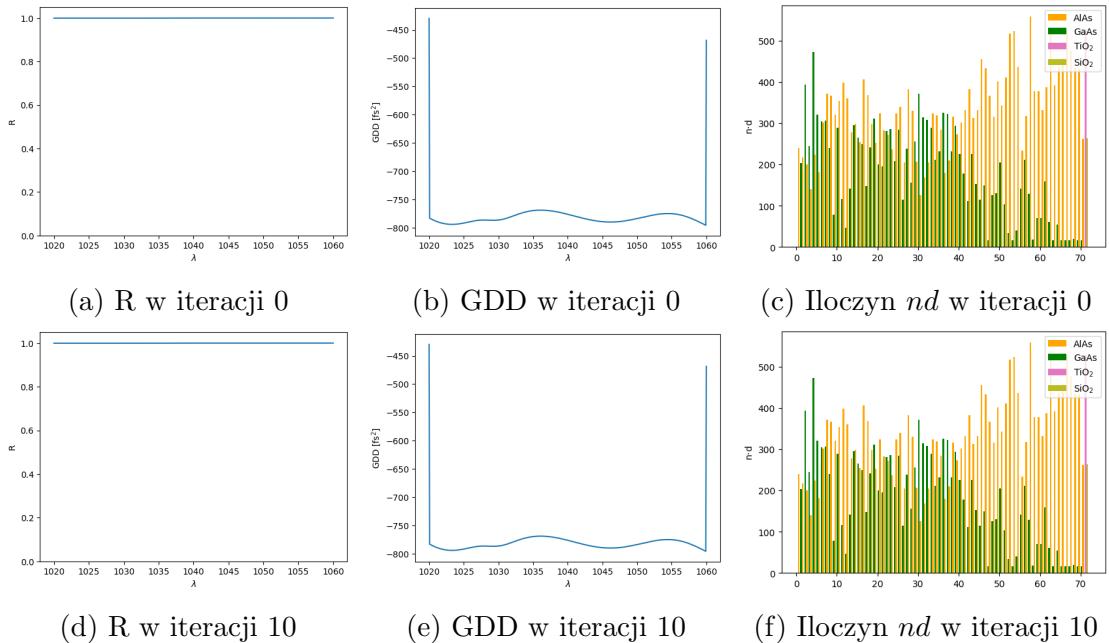
Rysunek 3.6: Wyniki obliczeń w przypadku losowania grubości warstw przy rozpoczęciu obliczeń od struktury 1 (gorszej)



Rysunek 3.7: Wyniki obliczeń w przypadku losowania grubości warstw przy rozpoczęciu obliczeń od struktury 2 (lepszej)

Można zauważyć, że dla lepszej struktury otrzymano wyniki zdecydowanie lepsze, co dowodzi, że to jest dobry sposób na lekkie polepszenie już istniejącej struktury, jednocześnie nie umożliwiając stworzenia nowej struktury zupełnie od zera. Dodatkowo można zauważyć, że w przypadku 3.7 najlepsza struktura nie zmieniła się po 499 iteracjach, co ilustruje trudność otrzymania w ten sposób rozwiązań naprawdę dobrych.

Dodatkowo powtórzono obliczenia z rysunku 3.7, lecz wykorzystując tym razem 100 razy większą ilość pszczół zwiadowców przy mniejszej ilości iteracji (ze względu na ich znaczącą długość). Otrzymane wyniki przedstawiono na rysunku 3.8. Można zauważyć, że różnice między otrzymanymi wykresami nie są znaczące i zmiana tego parametru nie ma wpływu na wynik w tym wypadku.



Rysunek 3.8: Wyniki obliczeń w przypadku losowania grubości warstw przy rozpoczęciu obliczeń od struktury 2 (lepszej) i  $N = 100000$  pszczół

### 3.3.2 Losowanie parametru wykorzystującego warunek Bragg'a

Następnie, aby polepszyć uzyskiwanie wyniki i zminimalizować losowość, skorzystano z warunku Bragg'a:

$$n_1 d_1 + n_2 d_2 = \frac{\lambda_B}{2}, \quad (3.10)$$

gdzie  $n_1, n_2$  — współczynniki załamania warstw 1 i 2,  $d_1, d_2$  — grubości warstw 1 i 2,  $\lambda_B$  — długość fali Bragg'a czyli długość fali dla której zwierciadło osiąga maksimum

odbijalności. Warunek ten posłużył do wprowadzenia zmiennej  $c$ , zdefiniowanej według wzoru:

$$\lambda_B = \lambda c. \quad (3.11)$$

Wartość  $\lambda$  oznacza środek właśnie badanego przedziału długości fali. Tak dobrana zmienna  $c$  oscyluje w przedziale  $\langle 0,5, 1,5 \rangle$ , a co za tym idzie  $\lambda_B$  przyjmuje wartości od 500 nm do 1500 nm przy  $\lambda = 1000$  nm. Dodatkowo wprowadzono warunek na zachowanie równości iloczynu  $nd$  dla każdej z warstw w danej parze, zgodnie ze wzorem  $n_1 d_1 = n_2 d_2$ . Na tej podstawie otrzymano następujące wzory na grubości warstw w parze:

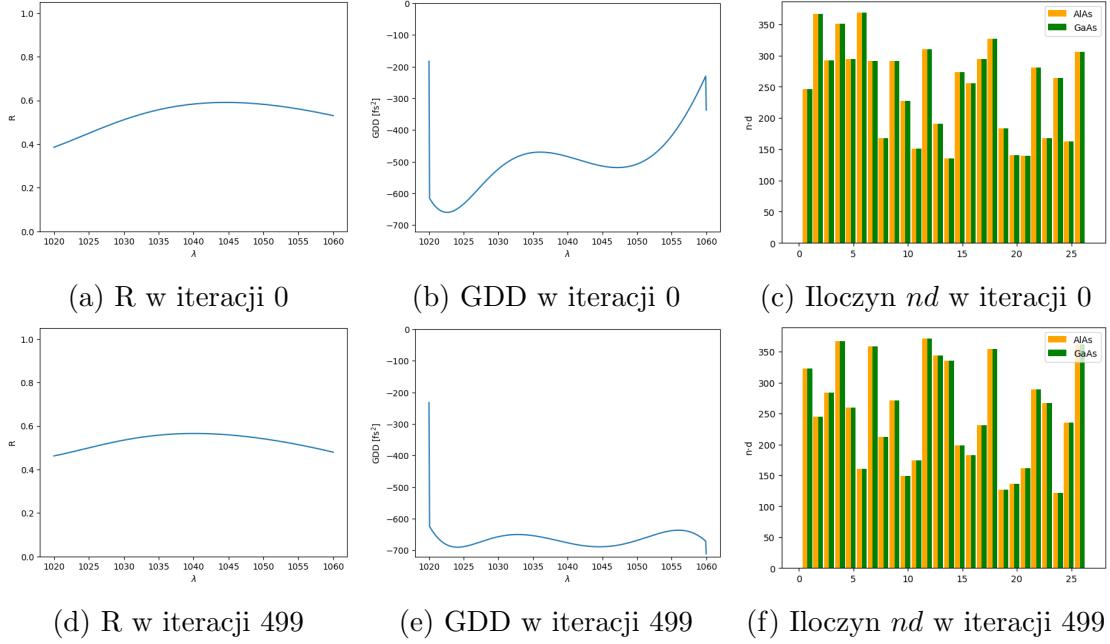
$$\begin{cases} d_1 = \frac{c\lambda}{4n_1}, \\ d_2 = \frac{c\lambda}{4n_2}, \end{cases} \quad (3.12)$$

natomiast wartość  $c$  można obliczyć podstawiając 3.11 do 3.10, co daje następujące równanie:

$$c = \frac{2}{\lambda} \cdot (n_1 d_1 + n_2 d_2). \quad (3.13)$$

Tak otrzymane równania posłużyły do stworzenia funkcji konwersji z grubości warstw do zmiennej  $c$  i w odwrotnym kierunku. W tym momencie możliwe było już zastosowanie zmiennej  $c$  w algorytmie rojowym. Tak też zostało zrobione i po licznych próbach korzystając z różnych wartości wag funkcji celu, udało się uzyskać wyniki przedstawione na rysunku 3.9 korzystając ze współczynników załamania struktury 1 (gorszej). Dodatkowo wykonano dla tych samych parametrów następne obliczenia na rysunku 3.10 tym razem korzystając z zestawu współczynników odbicia ze struktury 2 (lepszej). Na koniec wykonano jeszcze obliczenia zwiększąc liczbę pszczół zwiadowców 100 krotnie względem 3.10 i przedstawiono wyniki na rysunku 3.11.

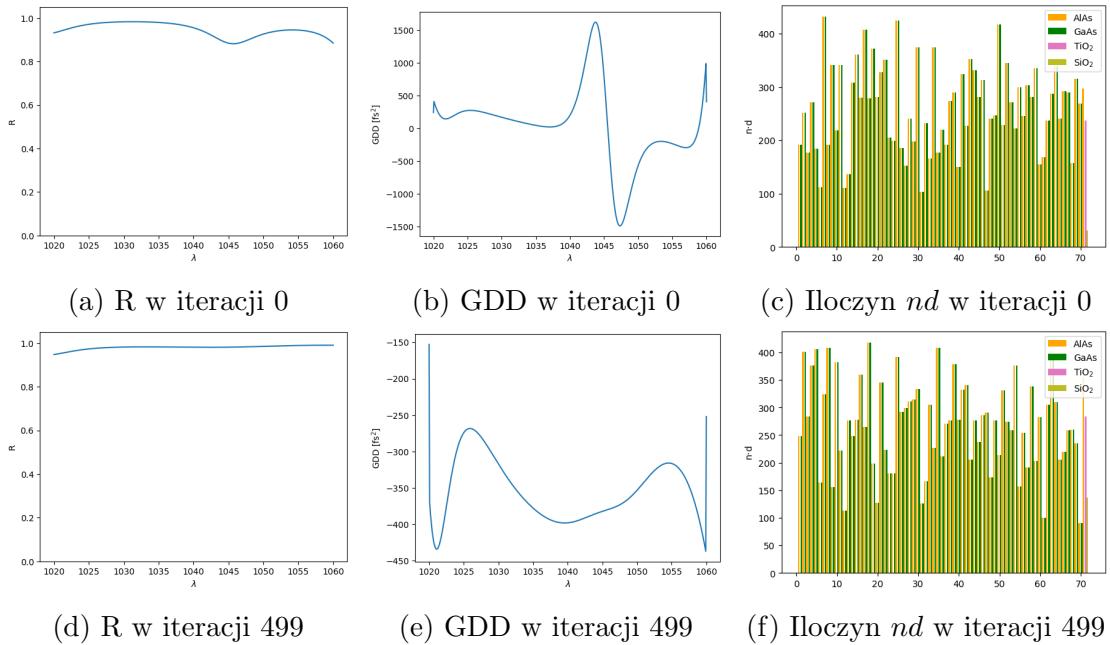
Można zauważyć na podstawie przedstawionych wyników, że otrzymane wyniki są gorsze od uzyskanego na rysunku 3.7. Dodatkowo iloczyny  $nd$  dla każdej warstwy z danej pary są równe, co świadczy o działaniu programu w sposób pożądany. Dodatkowo można zaobserwować ciekawy rezultat — wyniki korzystające ze struktury teoretycznej gorszej dały zauważalnie lepszy wynik niż te korzystające ze struktury teoretycznie lepszej. Oprócz tego widać, że obliczenia przy większej ilości pszczół zwiadowców (rysunek 3.11) dają gorsze rezultaty niż te przy mniejszej ich ilości (rysunek 3.10), w odróżnieniu od podejścia korzystającego z samych grubości warstw.



Rysunek 3.9: Wyniki obliczeń w przypadku losowania parametru wykorzystującego warunek Bragg'a przy wykorzystaniu współczynników załamania ze struktury 1

Wartości parametrów:  $N = 1000$ ,  $d_{near} = 0,005$ ,  $w_{qual} = 0,40$ ,  $w_{best} = 0,20$ ,

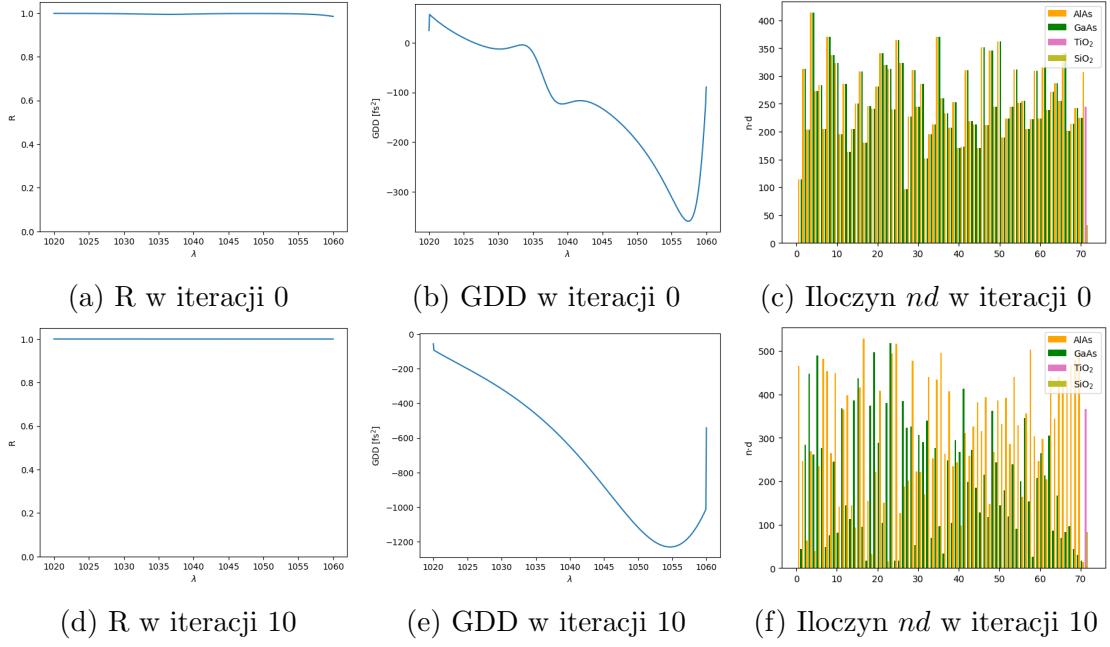
$$w_{better} = 0,8, p_1 = 0,5, p_2 = 2, p_3 = 10, p_4 = 1$$



Rysunek 3.10: Wyniki obliczeń w przypadku losowania parametru wykorzystującego warunek Bragg'a przy wykorzystaniu współczynników załamania ze struktury 2

Wartości parametrów:  $N = 1000$ ,  $d_{near} = 0,005$ ,  $w_{qual} = 0,40$ ,  $w_{best} = 0,20$ ,

$$w_{better} = 0,8, p_1 = 0,5, p_2 = 2, p_3 = 10, p_4 = 1$$



Rysunek 3.11: Wyniki obliczeń w przypadku losowania parametru wykorzystującego warunek Bragg'a przy wykorzystaniu współczynników załamania ze struktury 2

Wartości parametrów:  $N = 100000$ ,  $d_{near} = 0,005$ ,  $w_{qual} = 0,40$ ,  $w_{best} = 0,20$ ,  
 $w_{better} = 0,8$ ,  $p_1 = 0,5$ ,  $p_2 = 2$ ,  $p_3 = 10$ ,  $p_4 = 1$

### 3.3.3 Losowanie 2 parametrów korzystając z warunku Bragg'a i zależności między grubościami warstw

Na koniec, inspirowając się pracą [7], gdzie z sukcesem udało się znaleźć dobre wyniki zarówno dla GDD i R korzystając z algorytmów genetycznych i prezentując dane w postaci dwóch parametrów: jeden zależny od warunku Bragg'a jak w przypadku poprzednim a drugi wprowadzający zależność między iloczynami  $nd$  dla warstw w każdej parze, postanowiono sprawdzić czy również w przypadku algorytmów rojowych ta reprezentacja da równe dobre wyniki. W tym celu zastosowano następujący układ równań:

$$\begin{cases} n_1 d_1 + n_2 d_2 = \frac{\lambda_B}{2}, \\ n_1 d_1 = h(n_1 d_1 + n_2 d_2) = h \cdot \frac{\lambda_B}{2}, \\ n_2 d_2 = (1 - h)(n_1 d_1 + n_2 d_2) = (1 - h) \cdot \frac{\lambda_B}{2}, \\ \lambda_B = \lambda \left[ 1 + s \left( c - \frac{1}{2} \right) \right], \end{cases} \quad (3.14)$$

gdzie:  $c$ ,  $h$  — zmienne przyjmujące wartości z zakresu  $\langle 0, 1 \rangle$ , które będą używane w algorytmie rojowym jako argument badanej funkcji (jedna para tych wartości odpowi-

wiada parze w strukturze DBR),  $s$  — parametr pozwalający określić zakres wartości jakie przyjmuje  $\lambda_B$ .

Korzystając z tak przygotowanego układu równań, można wyznaczyć grubości warstw w danej parze:

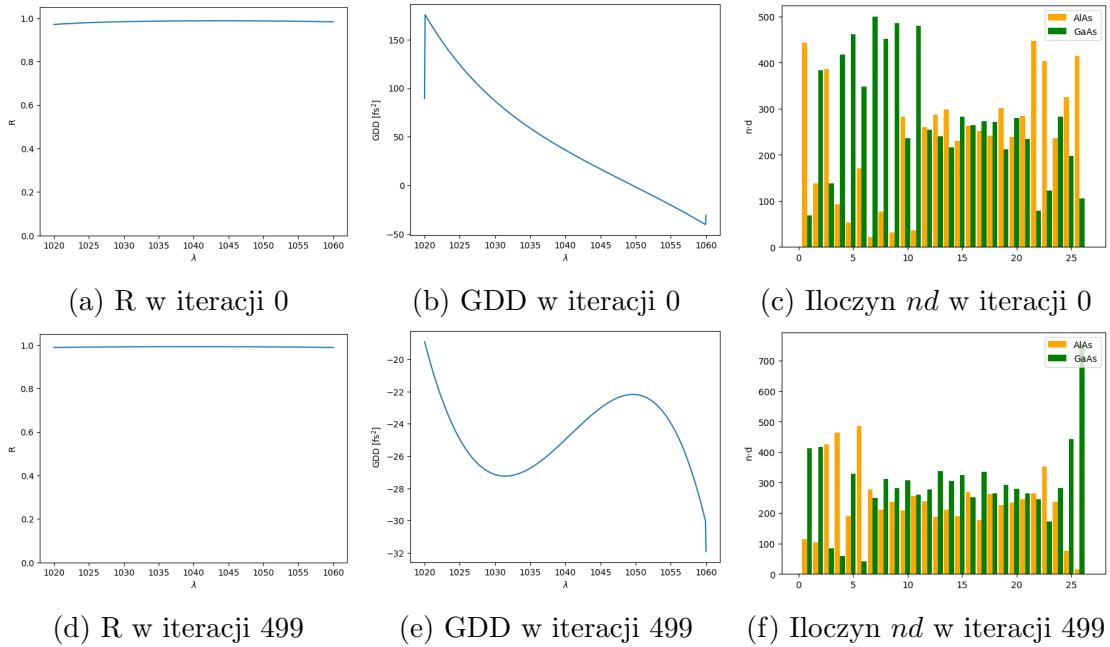
$$\begin{cases} d_1 = \frac{h\lambda_B}{2n_1}, \\ d_2 = \frac{(1-h)\lambda_B}{2n_2}. \end{cases} \quad (3.15)$$

Na podstawie tych wzorów sporządzono funkcję umożliwiającą transformację tablicy zmiennych  $c$  i  $h$  w tablicę grubości warstw w zwierciadle. Dodatkowo można było sporządzić funkcję odwrotną poprzez wyznaczenie wartości  $c$  i  $h$  zgodnie z:

$$\begin{cases} c = \frac{2}{\lambda s}(n_1 d_1 + n_2 d_2) - \frac{1}{s} + \frac{1}{2}, \\ h = 1 + \frac{n_2 d_2}{n_1 d_1}. \end{cases} \quad (3.16)$$

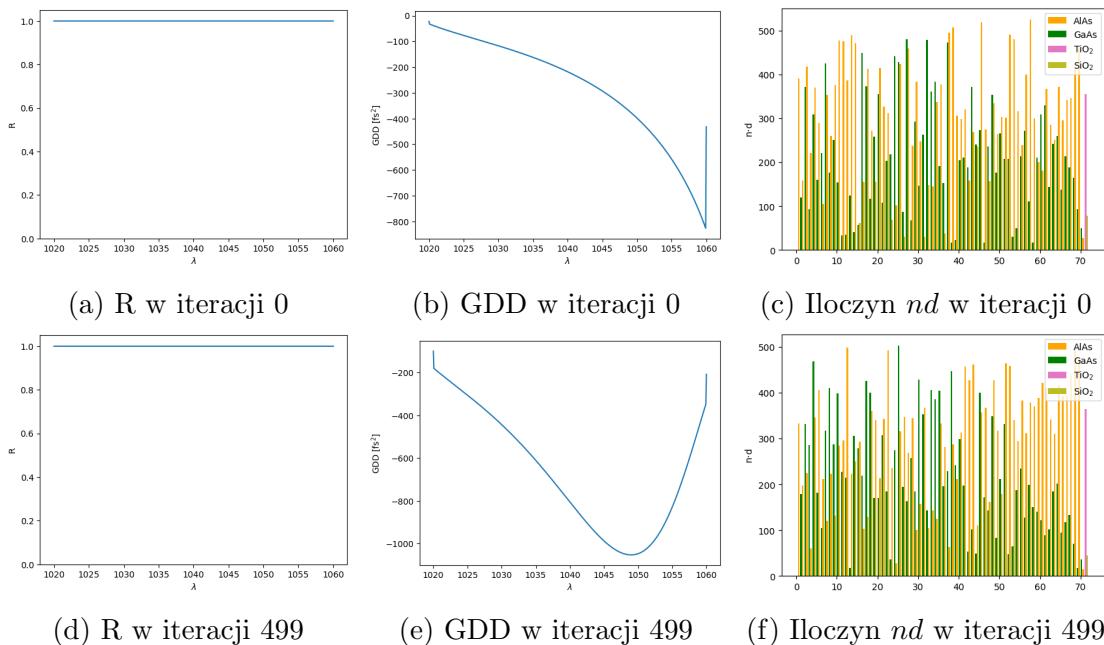
Na podstawie tak wyznaczonych zmiennych i funkcji dokonano obliczeń dla wielu zestawów parametrów funkcji celu. Najlepsze wyniki jakie udało się uzyskać przy użyciu współczynników załamania ze struktury 1 omawianej w 3.2 przedstawiono na rysunku 3.12, natomiast te przy użyciu współczynników załamania ze struktury 2 przedstawiono na rysunku 3.13. Dodatkowo postanowiono sprawdzić, również tutaj, wpływ zmiany ilości zwiadowców na wynik. W tym celu przyjęto  $N = 100000$  i wykonano obliczenia, których rezultat został przedstawiony na rysunku 3.14.

Można zauważyć, że wyniki przedstawione na rysunkach 3.12, 3.13 i 3.14 są znacznie gorsze od wyników otrzymanych w poprzednich podejściach. Natomiast porównując je bezpośrednio do siebie, można zauważyć, że zmiana ilości par zwierciadeł DBR i wprowadzenie dodatkowych warstw z innych materiałów półprzewodnikowych znacząco polepszyło rozwiązanie na rysunku 3.13 względem 3.12. Ponadto widać, że w tym wypadku, zwiększenie liczby pszczół zwiadowców nie wpłynęło szczególnie na jakość otrzymanego rozwiązania, natomiast zauważalnie przyspieszyło proces wyznaczenia rozwiązania finalnego, gdyż na rysunku 3.14 widać, że wykresy dla iteracji 0 jak i 10 są identyczne. Podsumowując, wyniki w przypadku algorytmu rojowego nie zyskują na jakości po wprowadzeniu tego sposobu reprezentacji danych, w odróżnieniu do algorytmów genetycznych. Możliwym sposobem na polepszenie wyników jest dopuszczenie zmiany ilości par zwierciadeł jak i dopuszczenie dodania dodatkowych warstw na koniec z innych materiałów. Takie rozwiązanie niestety nie zostało sprawdzone.



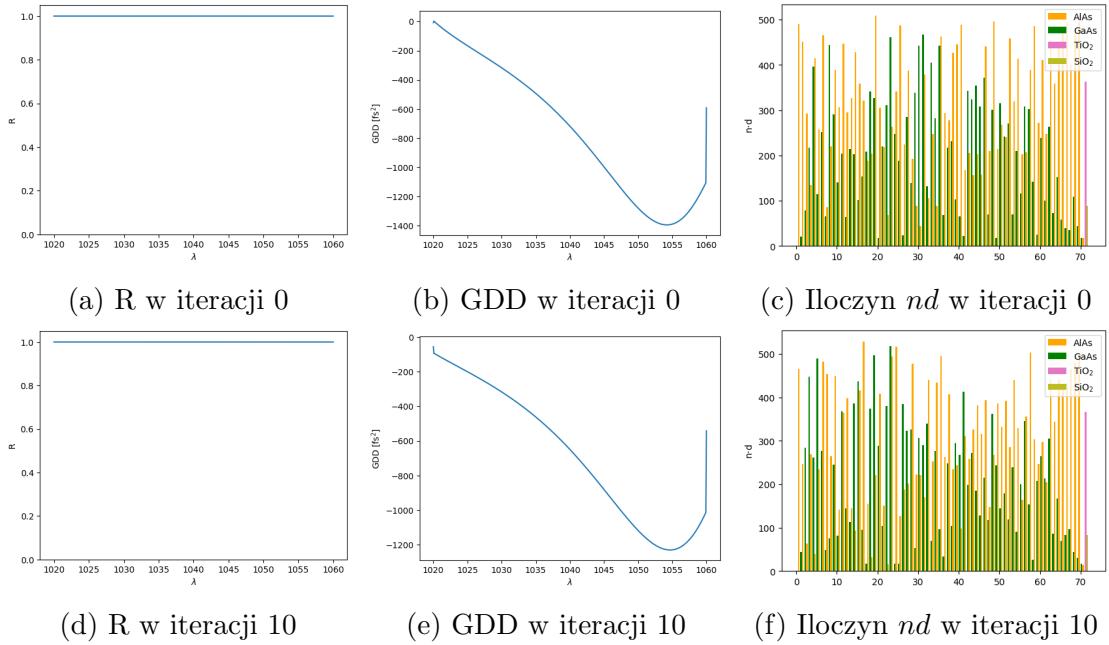
Rysunek 3.12: Wyniki obliczeń w przypadku losowania 2 parametrów przy wykorzystaniu współczynników załamania ze struktury 1

Wartości parametrów:  $N = 1000$ ,  $d_{near} = 0,05$ ,  $w_{qual} = 0,2$ ,  $w_{best} = 0,2$ ,  $w_{better} = 0,8$ ,  $p_1 = 0,5$ ,  $p_2 = 2$ ,  $p_3 = 10$ ,  $p_4 = 1$ ,  $s = 0,038$



Rysunek 3.13: Wyniki obliczeń w przypadku losowania 2 parametrów przy wykorzystaniu współczynników załamania ze struktury 2

Wartości parametrów:  $N = 1000$ ,  $d_{near} = 0,005$ ,  $w_{qual} = 0,4$ ,  $w_{best} = 0,2$ ,  $w_{better} = 0,8$ ,  $p_1 = 10$ ,  $p_2 = 20$ ,  $p_3 = 10$ ,  $p_4 = 5$ ,  $s = 0,038$



Rysunek 3.14: Wyniki obliczeń w przypadku losowania 2 parametrów przy wykorzystaniu współczynników załamania ze struktury 2

Wartości parametrów:  $N = 100000$ ,  $d_{near} = 0,005$ ,  $w_{qual} = 0,4$ ,  $w_{best} = 0,2$ ,  $w_{better} = 0,8$ ,  $p_1 = 10$ ,  $p_2 = 20$ ,  $p_3 = 10$ ,  $p_4 = 5$ ,  $s = 0,038$

## 3.4 Zastosowane technologie informatyczne

Pod koniec tego rozdziału należy jeszcze odpowiedzieć na pytanie jakie technologie zostały wykorzystane przy pisaniu programu jak i przy samym liczeniu. Program został napisany w języku *Python 3* z wykorzystaniem biblioteki *numpy* do wykonywania obliczeń na macierzach oraz *matplotlib* do wykonywania wykresów.

Cały algorytm podczas jednego cyklu wykonywał ok. miliona obliczeń trwających ok. 0,40 s każde, co skutkowało trudnością w uruchomienia go na komputerze osobistym ze względu na małą moc obliczeniową i co za tym idzie bardzo bardzo długim czasie wykonywania obliczeń (ponad 111 godzin). Dlatego też skorzystano z klastrów komputerowych pod nazwą *Dragon* i *Hydra* należących do Zespołu Fotoniki Instytutu Fizyki Politechniki Łódzkiej, co umożliwiło znaczne skrócenie obliczeń, do około 3 godzin przy wykorzystaniu 48 wątków na klastrze.

W celu wykorzystania dobrodziejstw klastra zastosowano bibliotekę *mpi4py*, która pozwoliła na podział danych na poszczególne wątki i wykonywania obliczeń niezależnie.

Później pod koniec każdej iteracji dane są zbierane do jednego wątku w celu podsumowania otrzymanych wyników.

Podsumowując udało się, z sukcesem, zastosować algorytmy rojowe w celu znalezienia zwierciadła DBR oferującego jak najmniejszą możliwą dyspersję opróżnienia grupowego. Niestety otrzymane wyniki są zauważalnie gorsze od tych znajdujących się w literaturze (np. w [7]), lecz metoda ta pokazuje zauważalny potencjał.

# Rozdział 4

## Podsumowanie

W niniejszej pracy przedstawiono sposób modelowania zwierciadeł DBR w celu uzyskania najmniejszej możliwej dyspersji opóźnienia grupowego przy pomocy algorytmów rojowych. Przedstawiony został, w sposób klarowny, schemat działania algorytmów rojowych oraz udowodniono, że faktycznie ta metoda działa. Przedstawiono również sposób wykorzystania algorytmów rojowych w celu znalezienia pożądanych zwierciadeł DBR, jak i wyniki, które z tego sposobu wynikały.

Uzyskane wyniki wykazują w sposób jasny, że wykorzystana metoda działa i jest możliwe wygenerowanie zwierciadeł ze względu dobrą dyspersją opóźnienia grupowego. Można zauważyć, że najlepsze wyniki udało się uzyskać w przypadku pierwszego podejścia polegającego na losowaniu grubości warstw. W tym wypadku otrzymano GDD na poziomie  $-800 \text{ fs}^2$  i współczynnik odbicia równy 1 na całym badanym przedziale. Te wyniki, mimo, że są dobre, polegają na znanych już strukturach zwierciadeł, które trudno pozyskać. Jedynymi sposobami na uzyskanie takich struktur jest korzystanie z wcześniej wykonanych prac, bądź też wygenerowanie struktury korzystając z jednego z dwóch pozostałych podejść.

Najgorsze wyniki otrzymano zdecydowanie korzystając z podejścia polegającego na losowaniu dwóch parametrów. Tutaj, w odróżnieniu od wyników uzyskanych przy pomocy algorytmów genetycznych w [7], algorytmy rojowe nie sprawdziły się i otrzymane wyniki są najgorsze z otrzymanych, szczególnie pod kątem wariancji GDD na danych zakresie długości fal.

Podsumowując, otrzymane wyniki odbiegają znacznie od wyników uzyskanych w innych pracach, jak chociażby praca [7]. Świadczy to o tym, że temat algorytmów

#### *Rozdział 4: Podsumowanie*

---

rojowych do modelowania dyspersyjnych własności zwierciadeł DBR pozostaje otwarty i należy znaleźć sposób na znaczne poprawienia wyników. Być może problem polega na wykorzystaniu generatora liczb pseudo-losowych w celu znalezienia nowych rozwiązań z sąsiedztwa i należy opracować nową metodę ich generowania.

# Bibliografia

- [1] U. Keller, A. C. Tropper. Passively modelocked surface-emitting semiconductor lasers. *Physics Reports*, 429:67–120, 2006.
- [2] L. C. Comandar, M. Lucamarini, B. Fröhlich, J. F. Dynes, A. W. Sharpe, S. W.-B. Tam, Z. L. Yuan, R. V. Penty, A. J. Shields. Quantum key distribution without detector vulnerabilities using optically seeded lasers. *Nature Photonics*, 10:312–315, 2016.
- [3] U. Keller. Recent developments in compact ultrafast lasers. *NATURE*, 424:831–838, sierpień 2003.
- [4] Franz X. Kärtner, Erich P. Ippen, Steven T. Cundiff. *Femtosecond Laser Development*, strony 54–77. Springer US, Boston, MA, 2005.
- [5] LABORATORIUM LASEROWEJ SPEKTROSKOPII MOLEKULARNEJ. Międzyresortowy Instytut Techniki Radiacyjnej. Politechnika Łódzka. Opis i zastosowanie laserów femtosekundowych. <http://www.mit.r.p.lodz.pl/raman/femtosekundy.pdf>.
- [6] N. Matuschek, F. X. Kärtner, U. Keller. Analytical design of double-chirped mirrors with custom-tailored dispersion characteristics. *IEEE JOURNAL OF QUANTUM ELECTRONICS*, 35(2), luty 1999.
- [7] M. Dems, P. Wnuk, P. Wasylczyk, Ł. Zinkiewicz, A. Wójcik-Jedlińska, K. Regiński, K. Hejduk, A. Jasik. Optimization of broadband semiconductor chirped mirrors with genetic algorithm. *Applied Physics B Laser and Optics*, 122(10):266, październik 2016.

- [8] Joanna Kwiecień. *Algorytmy stadne w rozwiązywaniu wybranych zagadnień optymalizacji dyskretnej i kombinatorycznej*, wolumen 315 serii *Rozprawy Monografie*. Wydawnictwa AGH, 2015.
- [9] Dariusz Baczyński. *Metody inteligencji obliczeniowej w elektroenergetyce*, wolumen 145 serii *Prace naukowe - Elektryka*. Oficyna Wydawnicza Politechniki Warszawskiej, 2013.
- [10] W Chmiel, P. Kadłuczka, G. Packanik. Zastosowanie algorytmów rojowych w rozwiązywaniu zagadnień permutacyjnych. *Automatyka*, 2(15):117–126, 2011.
- [11] A Jasik, M. Dems, P. Wnuk, P. Wasylczyk, A. Wójcik-Jedlińska, K. Regiński, Ł. Zinkiewicz, K. Hejduk. Design and fabrication of highly dispersive semiconductor double-chirped mirrors. *Applied Physics B Laser and Optics*, październik 2013.