|  |  |  |
| --- | --- | --- |
| **Teoria i metody optymalizacji - projekt** | | |
| *Imię i nazwisko*  **Sebastian Kułaga**  **209281**  **Paweł Kowalski**  **209436** | *Grupa:*  **Wtorek (parzysty)**  **godz. 15:15–16:55** | *Data:*  **30.05.2017r.** |
| *Prowadzący:* **Dr inż. Ewa Szlachcic** |
| **Algorytm VEGA dla dwukryterialnego zadania optymalizacji nieliniowej z ograniczeniami** | | |

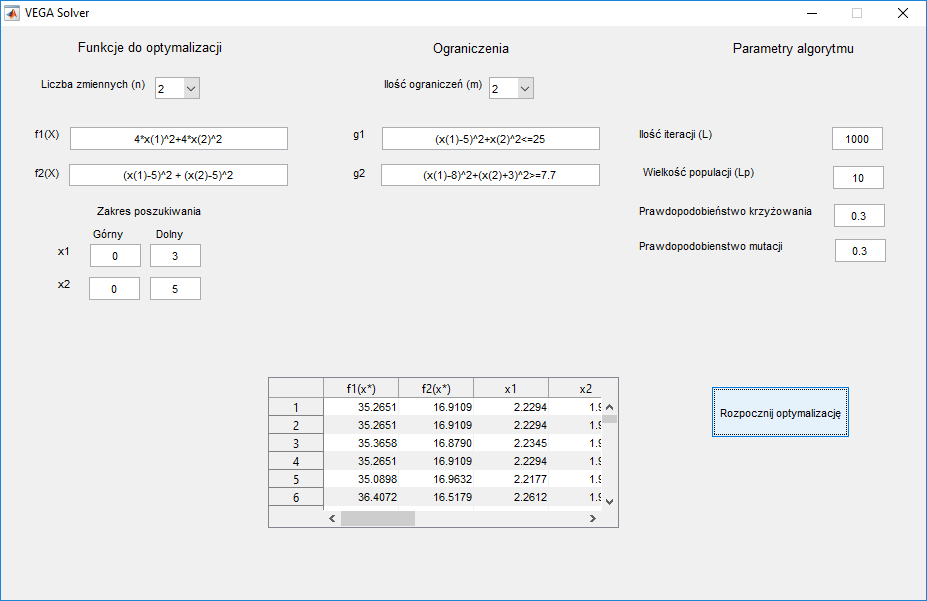
1. **Opis projektu:**
   1. **Celem projektu** było wykonanie programu wyszukującego minimum dwuwymiarowej funkcji celu *F(x)*, przy ograniczeniach nieliniowych, korzystając z algorytmu VEGA.
   2. **Algorytm VEGA** opracowany przez Davida Schaeffera w 1985 r. (Vector Evaluated Genetic Algorithm) to heurystyczny, ewolucyjny algorytm optymalizacji globalnej przeszukujący przestrzeń rozwiązań, za pomocą mechanizmów ewolucji i doboru naturalnego, czyli krzyżowania i mutacji. Kryterium stopu to liczba iteracji L.

Algorytm posiada nieliniowe ograniczenia dla odnalezienia minimum**.** Typowe oraz wykorzystane operatory heurystyczne dla algorytmów genetycznych to krzyżowanie i mutacja:

* Krzyżowanie polega na połączeniu wybieranych losowo genotypów w jeden. Efektem krzyżowania jest posiadanie przez potomka kombinacji zespołu cech osobników rodzicielskich. Potomkowie następnie zastępują rodziców. Parametr *pc* określa prawdopodobieństwo z jakim każde rozwiązanie może wziąć udział w krzyżowaniu. W projekcie wykorzystano metodę obliczania wartości średniej genów.
* Mutacja wprowadza do genotypu losowe zmiany, zapewniając różnorodność w populacji i zabezpieczająca algorytm przed przedwczesną zbieżnością. W projekcie wykorzystano metodę losowej zmiany wartości z prawdopodobieństwem *pm.*
  1. **Środowisko programistyczne** w którym wykonano projekt to MatLAB R2015a. Do algorytmu stworzono również graficzny interfejs użytkownika (GUI). Umożliwia on dobór parametrów i funkcji bez wpisywania ich w skrypt programu.

Program zbudowany jest z 4 modułów:

* **Form1.m** – funkcje odpowiadające za komunikację z GUI
* **VEGA.m** – algorytm VEGA, wywoływany z odpowiednimi argumentami
* **ograniczenia.m** - funkcja odpowiadająca za sprawdzanie ograniczeń
* **form1.fig** – projekt graficznego interfejsu użytkownika



Rysunek 1 Widok graficznego interfejsu użytkownika z wprowadzonymi danymi

Aby skorzystać z aplikacji należy uruchomić plik Form1.m w MatLAB-ie, a następnie wprowadzić żądane funkcje celu, ograniczenia i parametry dodatkowe. Następnie należy kliknąć przycisk *Rozpocznij optymalizację* w celu znalezienia optymalnego rozwiązania. Program wyświetli w dolnej części tabeli wartości optymalne funkcji celu wraz z argumentami. Dodatkowo w nowym oknie pojawią się wykresy znalezionych rozwiązań (zdominowane i niezdominowane) jak i front Paretto (rozwiązania niezdominowane).

* 1. **Zasady danych**

W projekcie przyjęto następujące oznaczenia:

* *Lp* – Ilość osobników w danej populacji.
* *L* – Ilość iteracji algorytmu.
* *Pc*- Prawdopodobieństwo krzyżowania osobników.
* *Pm*-Prawdopodobieństwo mutacji osobników.
* *n* – liczba zmiennych wektora wejściowego.
* *m* – liczba zmiennych funkcji ograniczających.
  1. **Funkcje wykorzystane przy testach algorytmu:**

|  |
| --- |
| **Funkcja Binh’a i Korna** |
| Rysunek 2 Wykres funkcji Binh'a i Korna (wzorzec) Rysunek 3 Wykres funkcji Binh'a Korna (projekt) |
| Rysunek 2 Funkcje celu |
| Rysunek 3 Ograniczenia |
| Rysunek 4 Zakres poszukiwań |

|  |
| --- |
| **Funkcja Chankong’a i Haimes’a** |
| **C:\Users\Darth\AppData\Local\Microsoft\Windows\INetCacheContent.Word\fp2l1000lp10.jpg**  Rysunek 5 Wykres funkcji Chakong'a i Haimes'a (wzorzec) Rysunek 6 Wykres funkcji Chankong'a i Haimes'a (projekt) |
| Rysunek 7 Funkcje celu |
| Rysunek 8 Ograniczenia |
| Rysunek 9 Zakres poszukiwań |

|  |
| --- |
| **Funkcja testowa Binh’a nr 4** |
| C:\Users\Darth\AppData\Local\Microsoft\Windows\INetCacheContent.Word\fp3l1000lp10.jpg  Rysunek 10 Wykres funkcji testowej Binh'a nr 4 (wzorzec) Rysunek 11 Wykres funkcji testowej Binh'a nr 4 (projekt) |
| Rysunek 12 Funkcje celu |
| Rysunek 13 Ograniczenia |
| Rysunek 14 Zakres poszukiwań |

|  |
| --- |
| **Funkcja Osyczki i Kundu** |
| C:\Users\Darth\AppData\Local\Microsoft\Windows\INetCacheContent.Word\n6l1000lp10.jpg  Rysunek 15 Wykres funkcji Osyczki i Kundu (wzorzec) Rysunek 16 Wykres funkcji Osyczki i Kundu (projekt) |
| |  | | --- | | Rysunek 17 Funkcje celu | |
| Rysunek 18 Ograniczenia |
| Rysunek 19 Zakres poszukiwań |

1. **Testy algorytmu:**

Wykonano dla czterech funkcji opisanych w powyższym punkcie. Trzy z nich składały się z dwóch zmiennych (*n* = 2). Funkcja Osyczki i Kudu składa się z sześciu zmiennych (*n*=6). Testy wykonywano kolejno dla liczby iteracji *L*=1000, *L* = 2000, *L*=3000 oraz liczby osobników w populacji *Lp*=10, *Lp*=15, *Lp*=20.

* 1. **Przebieg testów dla funkcji Chakong’a i Haimes’a:**

|  |  |
| --- | --- |
| **Liczba iteracji (L)** | **Liczba osobników w populacji (Lp)** |
| Rysunek 20 Liczba iteracji L=1000 | Rysunek 25 Liczba osobników w populacji Lp=10 |
| Rysunek 22 Liczba iteracji L=2000 | Rysunek 26 Liczba osobników w populacji Lp=15 |
| Rysunek 24 Liczba iteracji L=3000 | Rysunek 27 Liczba osobników w populacji Lp=20 |

1. **Wnioski:**

Zaimplementowany algorytm genetyczny VEGA rozwiązuje zadanie optymalizacji dwukryterialnej dla zadanych przykładów.

Dla testowanych funkcji celu zauważono, że największy wpływ na dokładność rozwiązania ma ilość iteracji. Jednakże czas rozwiązywania zdania dla ilości powtórzeń większego od 3000 jest dosyć długi (powyżej minuty czasu dla funkcji kryterialnych z dwiema zmiennymi). Najwięcej czasu zajmuje sortowanie rozwiązań przy tworzeniu wykresu Paretto. Podobnie zwiększając ilość osobników w populacji można zwiększyć dokładność rozwiązań. Parametry krzyżowania i mutacji należy dobrać doświadczalnie, aczkolwiek zauważono, że dla przykładu z n=6 większa mutacja jest korzystniejsza, ponieważ umożliwia szersze przeszukiwanie zbioru rozwiązań. Wykresy generowane przez projekt różnią się od wykresów wzorcowych, co do liczby punktów w pewnych zakresach wykresu, gdyż algorytm VEGA ma skłonność do znajdowania skrajnych wartości funkcji celu. Na wynik końcowy wpływają wylosowane na początku rozwiązania.

Główną zaletą algorytmu jest to, że używa prostych założeń i jest łatwy w implementacji. Jednakże jego wada to sprawdzanie rozwiązania dla jednej funkcji celu, co nie jest korzystne przy optymalizacji wielokryterialnej. Bardzo prawdopodobne jest iż, zostaną wyszukane rozwiązania optymalne dla jednej funkcji celu przez operator selekcji w danej populacji, co wpływa na wspomnianą wcześniej skłonność do znajdowania skrajnych rozwiązań.

1. **Bibliografia:**

* *VEGA and MOGA an Approach to Multi-Objective Optimization*, Autor: Indresh Kumar Gupta, Jeetendra Kumar (stan na dzień 29.05.2017)
* Funkcje testowe dla zadań optymalizacji: https://en.wikipedia.org/wiki/Test\_functions\_for\_optimization (stan na dzień 29.05.2017)
* http://staff.iiar.pwr.wroc.pl/ewa.szlachcic/materialy%20dydaktyczne/air\_studia\_2\_stopnia/metody\_optymalizacji\_wielokryterialnej\_Mlynek.pdf (stan na dzień 29.05.2017)