

Data oddania: \_\_\_\_\_

Ocena: \_\_\_\_\_

Paweł Musiał      178726  
Łukasz Michalski    178724

## Zadanie 5 - Optymalizacja wielowymiarowa z ograniczeniami\*

### 1. Cel

Napisać program rozwiązujący problem optymalizacji z ograniczeniami za pomocą jednej z poniższych metod:

- zewnętrzna lub wewnętrzna funkcja kary,
- SLP,
- optymalizacja wielokryterialna - metoda sumy ważonej,
- Box.

### 2. Rozwiązanie zadania

#### 2.1. Optymalizacja wielokryterialna - metoda sumy ważonej

Problemy wielokryterialne, to problemy w których optymalizowanym wyrażeniem, jest parę funkcji celu. Jednym z proponowanych rozwiązań w literaturze [?] do tego typu problemów jest metoda sumy ważonej. Metoda ta polega na zastąpieniu wielu funkcji jedną funkcją będącą liniową kombinacją składowych.

Dla przykładu, mając problem postaci:

$$\min F(f_1, f_2, f_3, \dots, f_n) \tag{1}$$

---

\* SVN: [https://serce.ics.p.lodz.pl/svn/labs/moo/lc\\_cz1600/lmpm](https://serce.ics.p.lodz.pl/svn/labs/moo/lc_cz1600/lmpm)

przy ograniczeniach :

$$g(x) \leq 0, h(x) = 0$$

zastępujemy grupę funkcji celu za pomocą :

$$U = \sum_{i=1}^n w_i \cdot f_i \quad (2)$$

gdzie :

$$\sum_{i=1}^n w_i = 1$$

oraz  $w > 0$ .

Rozwiązanie takiego problemu będzie leżało na tak zwanym froncie pareto. Jest to powierzchnia na której występują rozwiązania niedominujących, czyli rozwiązań w sensie pareto, co oznacza, że każde rozwiązanie w zbiorze jest optymalne i nie lepsze ani gorsze od pozostałych.

Metoda ta jedynie opisuje sposób w jaki można rozwiązać problem z wieloma funkcjami celu, nie precyzuje jednak sposobu dalszych rozwiązań wyznaczonego problemu zastępczego  $U$ . Wartości wag w tej metodzie, można dobierać według potrzeb, oczywiście pamiętając o nakreślonych restrykcjach. Z drugiej strony zostały zaproponowanie rozwiązania adaptacyjnego doboru wag.

Jako, że metoda ta jedynie mówi o podejściu jakie można zastosować do problemów wielokryterialnych, polegającym na złożeniu wielu funkcji bazowych w jedną funkcję będącą superpozycją składowych przemnożonych przez jakieś wartości wag, tak na prawdę każdy bardziej złożony problem możemy traktować jako takie właśnie rozwiązanie. Dlatego też w naszym programie skupiliśmy się na jednej z metod rozwiązywania problemu „zastępczego” czyli problemu wielowymiarowej optymalizacji z ograniczeniami.

## 2.2. Zewnętrzna funkcja kary

Jest to metoda rozwiązywania problemów wielokryterialnych z ograniczeniami. Metoda ta w swojej prostocie polega na rozwiązywaniu problemu zastępczego, w którym rozwiązujemy problem podstawowy z dodaną funkcją kary będącą wartością - karą, za niespełnienie ograniczeń. Niestety nie przedstawimy tutaj dowodu poprawności tej metody. Jednak w wielu przypadkach metoda ta pozwala na dobre wyniki. Przedstawiony poniżej opis jest zaczerpnięty z [?].

Minimalizacja  $f(x)$ , przy ograniczeniach:

$$\begin{aligned} [g(x)]_m &\leq 0 \\ [h(x)]_l &= 0 \end{aligned}$$

Wprowadzamy problem zastępczy, postaci:

$$F(x, r_h, r_g) = f(x) + P(x, r_h, r_g) \quad (3)$$

gdzie :

$$P(x, r_h, r_g) = r_h \left[ \sum_{j=1}^l (h_j(x))^2 \right] + r_g \left[ \sum_{j=1}^m (\max(0, g_j(x)))^2 \right] \quad (4)$$

Gdzie parametry  $r_h$  i  $r_g$  mówią, jak „mocno” karać za naruszenie ograniczeń. Podstawiając wszystko do jednego równania otrzymujemy :

$$F(x, r_h, r_g) = f(x) + r_h \left[ \sum_{j=1}^l (h_j(x))^2 \right] + r_g \left[ \sum_{j=1}^m (\max(0, g_j(x)))^2 \right] \quad (5)$$

Jak widać niejako pozbyliśmy się ograniczeń włączając je do minimalizowanego wyrażenia. W sposób ten otrzymaliśmy problem wielowymiarowej optymalizacji bez ograniczeń który możemy rozwiązywać znanymi nam metodami czyli na przykład algorytmem pełzającego sympleksu, bądź jedną w metod quasi-newtonowskich.

Algorytm w opisie krokowym:

1. Wybierz punkt początkowy  $x = x_0$  znajdujący się w obszarze dopuszczalnym, współczynniki kary  $r_h$ ,  $r_g$  i odpowiadające im współczynniki skalujące  $C_{r_h}$  i  $C_{r_g}$ , oraz kryterium stopu, tutaj ilość iteracji  $k_{max}$ , zainicjuj licznik iteracji  $k = 1$ .
2. rozwiąż problem zastępczy  $x_q = \min(F(x, r_g, r_h))$
3. sprawdź czy ograniczenia zostały naruszone  $g_i(x_q) \leq 0$ ,  $h_j(x_q) = 0$ ,  $i \in (1 \dots m)$ ,  $j \in (1 \dots l)$ , jeśli ograniczenia nie zostały naruszone, sprawdź kryterium stopu.
4. skaluj współczynniki kary :  $r_h = r_h \cdot C_{r_h}$ ,  $r_g = r_g \cdot C_{r_g}$
5. zwiększ licznik iteracji  $k = k + 1$ , przypisz  $x = x_q$  przejdź do 2.

### 3. Opis programu

```

1 function fstr=equalConSum(f)
2 fstr='( ';
3 for i=1:length(f)-1
4     fstr=strcat(fstr, ',', f(i), ')^2 + ');
5 end
6 fstr=strcat(fstr, '(', f(length(f)), ')^2 )';

```

ograniczenia równościowe

```

1 function fstr=inEqualConSum(f)
2 fstr='( ';
3 for i=1:length(f)-1
4     fstr=strcat(fstr, '(max(0, ', f(i), '))^2 + ');
5 end
6 fstr=strcat(fstr, '(max(0, ', f(length(f)), '))^2 )';

```

Ograniczenia nierównościowe

```

1 function fout = penaltyFun( f, h, g, rh, rg )
2 hfun=0;
3 gfun=0;
4 if ~isempty(h)
5     hfun=@(x,y) subs(h,{ 'x' , 'y' },{x,y});
6 end
7 if ~isempty(g)
8     gfun=@(x,y) subs(g,{ 'x' , 'y' },{x,y});
9 end
10 ffun=@(x,y) subs(f,{ 'x' , 'y' },{x,y});
12 fout = @(x) ffun(x(1),x(2)) + rh*hfun(x(1),x(2)) +
14     rg*gfun(x(1),x(2));
16 end

```

Funkcja problemu zastępczego

```

1 function [xi, fval] = epf(f, h, g, rh, rg, x0, Crh, Crg, kMax )
2 hpenalty='';
3 gpenalty='';
4 if ~isempty(h)
5     hpenalty=equalConSum(h);
6 end
7 if ~isempty(g)
8     gpenalty=inEqualConSum(g);
9 end
10 xi=x0
11 for i=1:kMax
12
13     Fp=penaltyFun(f, hpenalty, gpenalty, rh, rg);
14
15     [xi, fval]=fminunc(Fp, xi)
16     unsatisfied=0;
17     for j=1:length(h)
18         hc=subs(h(j),{ 'x' , 'y' },{ xi(1), xi(2) });
19         if (hc~=0)
20             unsatisfied=unsatisfied+1;
21         end
22     end
23     for j=1:length(g)
24         gc=subs(char(g(j)),{ 'x' , 'y' },{ xi(1), xi(2) });
25         if (gc>0)
26             unsatisfied=unsatisfied+1;
27         end
28     end
29     if(unsatisfied==0 || i>=kMax)
30         break;
31     end
32     rh=rh*Crh;
33     rg=rg*Crg;
34 end
35 end

```

Algorytm zewnętrznej funkcji kary

## 4. Wyniki

Rozwiązanie problemu testowano na funkcji przykładowej zawartej w książce [?].

$$f(x_1, x_2) = x_1^4 - 2x_1^2x_2 + x_1^2 + x_1x_2^2 - 2x_1 + 4 \quad (6)$$

przy ograniczeniach:

$$h(x_1, x_2) : x_1^2 + x_2^2 - 2 = 0$$

$$g(x_1, x_2) : 0.25x_1^2 + 0.75x_2^2 - 1 \leq 0$$

przy dobranych parametrach:

1.  $x_0 = [1, 2]$
2.  $k_{max} = 3$
3.  $r_h = 5, C_{r_h} = 0.95$
4.  $r_g = 5, C_{r_g} = 0.95$

iter	x	f(x)
1	0.9464, 1.0364	2.9740
2	0.9454, 1.0369	2.9736
3	0.9445, 1.0373	2.9731

Wynik w [?] dla porównania wynosił  $x=[0.9775, 1.0165]$ ,  $f(x)=2.9810$ . Jak widać uzyskany wynik jest bardzo bliski wartości otrzymanej przez obliczonej przez naszą implementację, te same metody już po 3 iteracjach.

## 5. Wnioski

Przedstawiona powyżej implementacja optymalizacji wielowymiarowej z ograniczeniami przy pomocy zewnętrznej funkcji kary jest jedną z metod optymalizacji znajdujących rozwiązanie rozwiązując przybliżone problemy do oryginalnego. Podejście takie skutkuje dobrymi wynikami i jest łatwe w implementacji ponieważ opierają się one z reguły na algorytmach rozwiązywania standardowych problemów optymalizacji, mamy tutaj na uwadze takie algorytmy jak SLP, SQP, metody oparte o funkcję kary. Jedynym minusem ich działania jest nie działanie na prawdziwej funkcji celu a jedynie na jej przybliżeniu, co w konsekwencji niejako implikuje słabą jakość przy zadaniach bardziej złożonych. Zaimplementowana przez nas metoda testowana była na prostym przykładzie dlatego też szybko uzyskała zbieżność.

## Literatura

- [1] „An Introduction to Optimization” , Edwin Kah Pin Chong and Stanislaw H. Zak, Hoboken, EUA : Wiley-Interscience 2008
- [2] „Introduction to Optimum Design”, Arora, Jasbir S., Elsevier Academic Press 2004
- [3] „Applied Optimization with MATLAB Programming”, Venkatamaran P., Wiley 2001