ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: AUTOMATYKA z.117 Nr kol. 1337

Wiesław PORĘBSKI
Politechnika Gdańska

KONCEPCJE OBIEKTOWE W OPTYMALIZACJI WIELOKRYTERIALNEJ

Streszczenie. Niniejsza praca prezentuje problem optymalizacji wielokryterialnej i optymalne w sensie Pareto własności rozwiązań nietrywialnego, nieliniowego zadania z ograniczeniami nierównościowymi. Pokazano zastosowanie do tego celu nowych koncepcji obiektowych, implementowanych w języku C++. W szczególności pokazano konstrukcję *obiektów funkcyjnych* (funktorów) i ich zastosowanie do znajdowania ekstremów utopijnych, a także przesyłanie komunikatów dla uzyskania rozwiązania kompromisowego.

OBJECT-ORIENTED APPROACH IN MULTICRITERIA DECISION MAKING

Summary. This paper presents a nontrivial, nonlinear multicriteria problem with given inequality constraints. The solution to the problem is given in terms of object-oriented approach and implemented in C++ language. In this context, the construct of function objects, or *functors*, is given and used in seeking the utopia points. Since the results form a Pareto-optimal compromise set of solutions, the final decision making requires interaction with the user. For this, a message-passing method between interacting objects is used.

1. Sformułowanie problemu

W projektowaniu systemów jakość rozwiązań wyrażamy często za pomocą zbioru k kwantyfikowalnych celów $C = \{ c_1, c_2, ..., c_k \}$. Prawidłowe postawienie zadania wymaga, aby cele te były niesprzeczne i konkurencyjne. Kwantyfikacja polega zwykle na zapisie celów w postaci k kryteriów $g_1, g_2, ..., g_k$, z których każde zależy od n zmiennych decyzyjnych $\underline{x} = [x_1, x_2, ..., x_n]$. Ze względów technicznych, technologicznych lub ekonomicznych zmienne decyzyjne podlegają pewnym ograniczeniom. Realizacja dowolnego celu c_i ze zbioru C polega zazwyczaj na znalezieniu wartości ekstremalnej odpowiadającego mu kryterium g_i.

Załóżmy, że zmienne decyzyjne x_i są wielkościami skalarnymi, a na zbiór decyzji dopuszczalnych $\underline{x} = [x_1, x_2, ..., x_n]$ nałożono ograniczenia: $\underline{x} \in D$, gdzie $D \subset R^n$. Zależne od $\underline{x}$ kryteria możemy traktować jako składowe $g_i(\underline{x})$ funkcji wektorowej:

$$\underline{g}(\underline{x}) = \left[g_1(\underline{x}), ..., g_k(\underline{x}) \right] \tag{1}$$

która odwzorowuje zbiór D w przestrzeni decyzji na zbiór G w przestrzeni kryterialnej:

$$\underline{g}: D \mapsto G \subset R^k \tag{2}$$

gdzie: G – zbiór *wartości kryterium wektorowego* $\underline{g}(\underline{x})$, $\underline{x} \in D$, R^k – *przestrzeń kryterialna.*

Jeżeli każda ze składowych kryterium wektorowego (1) osiąga na zbiorze D swoją wartość ekstremalną $\overset{*}{g}_i$, wówczas wektor:

$$\overset{*}{\underline{g}} = \left[\overset{*}{g}_1, \ldots, \overset{*}{g}_k \right] \tag{3}$$

nazywamy *punktem utopijnym* lub *punktem idealnym* w przestrzeni kryterialnej R^k [5,7].

Z doświadczenia wynika, że tylko w niezwykle rzadkich przypadkach kryteria składowe osiągają swoje ekstrema w tym samym punkcie przestrzeni decyzyjnej (tj. w tym samym punkcie $\overset{*}{\underline{x}} \in D$). Z reguły otrzymujemy k różnych punktów $\overset{*}{\underline{x}}$, w których składowe $g_i(\underline{x})$ osiągają swoje wartości utopijne.

Jest też oczywiste, że nietrywialny problem optymalizacji wielocelowej to taki, w którym występuje zjawisko konkurencyjności w zbiorze celów. Oznacza to, że np. przejście od punktu $\left[\overset{*}{g}_1, g_2, \ldots, g_k \right]$ do punktu $\left[g_1, \overset{*}{g}_2, \ldots, g_k \right]$ w przestrzeni R^k zwiększa stopień realizacji celu c_2, lecz jednocześnie zmniejsza stopień realizacji celu c_1. Powstaje zatem pytanie: jak znaleźć zadowalające rozwiązanie?

1.1. Optymalizacja w sensie Pareto

Jedną z możliwych metod jest wykorzystanie koncepcji, sformułowanej przez włoskiego socjologa i ekonomistę Vilfredo Pareto w roku 1896. Koncepcja ta nazywana jest obecnie warunkiem optymalności w sensie Pareto. Dla przypadku jednoczesnej maksymalizacji k składowych kryterium wektorowego (1) warunek Pareto ma postać [3,4,6,7]:

wektor dopuszczalny $\underline{x}^p \in D$ jest polioptymalny, jeżeli nie istnieje żaden inny wektor dopuszczalny $\underline{x}$, dla którego:

$$\underline{g}(\underline{x}) \geq \underline{g}(\underline{x}^p) \tag{4}$$

Nierówność (4) należy traktować jako formalny [2.5,7] zapis k nierówności:

$$g_i(\underline{x}) \geq g_i(\underline{x}^p); \quad i = 1,2, \ldots, k \tag{5}$$

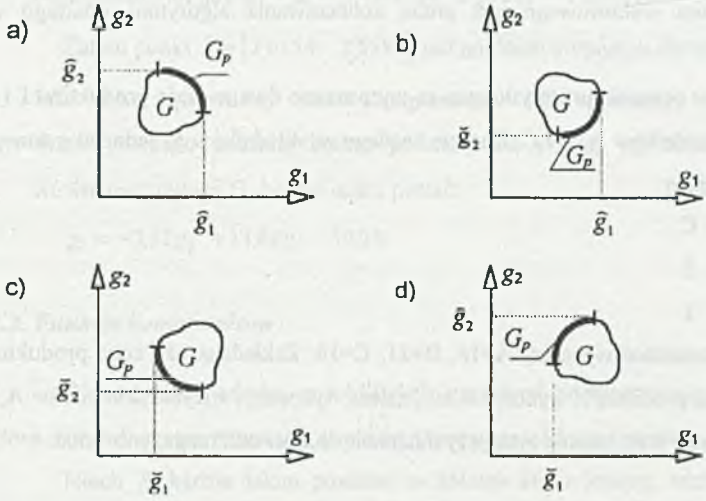
Uwaga. Wektor polioptymalny ma w literaturze przedmiotu następujące synonimy: rozwiązanie efektywne, rozwiązanie niezdominowane.

1.2. Rozwiązania efektywne i kompromisy

Koncepcja Pareto daje warunek konieczny optymalności; z własności punktów utopijnych wynika, że tylko w przypadku istnienia jednego rozwiązania (przypadek raczej odosobniony) utopijnego $\dot{\underline{x}} \in D$ warunek Pareto spełnia jeden punkt (wektor) $\underline{x}^p \in D$. W ogólności mamy zbiór tzw. *rozwiązań efektywnych* $P \subset D \subset R^n$:

$$P = \{ \underline{x}^p \in D \mid \nexists\, \underline{x} \in D, \text{ dla którego } \underline{g}(\underline{x}) \geq \underline{g}(\underline{x}^p) \} \tag{6}$$

Rysunek 1 ilustruje odwzorowania (według przepisu (2)) zbioru rozwiązań efektywnych P (według (6)) na zbiór G^p, który jest podzbiorem zbioru G w przestrzeni kryterialnej R^2.



Rys. 1. Zbiór wartości kryterium wektorowego i zbiór kompromisów
Fig. 1. A set of criteria values and a compromise set

Zbiór G^p, który „godzi" konkurencyjne cele, nazywany jest *zbiorem* kompromisów. Na rysunku 1 pokazano cztery warianty rozwiązań zadania optymalizacji dwukryterialnej, przy czym wartości utopijne $\dot{g}_i$ oznaczono: $\dot{g}_i = \hat{g}_i$ lub $\dot{g}_i = \check{g}_i$, $\hat{g}_i = g_{i\max}$, $\check{g}_i = g_{i\min}$.

Wariant a) odpowiada jednoczesnej maksymalizacji g_1 i g_2, wariant b) – jednoczesnej maksymalizacji g_1 i minimalizacji g_2, wariant c) – jednoczesnej minimalizacji g_1 i g_2, wariant d) – jednoczesnej minimalizacji g_1 i maksymalizacji g_2.

Zauważmy, że wraz z wyznaczeniem zbioru kompromisów G^p, a stąd zbioru rozwiązań efektywnych P, kończy się ścisłe postępowanie optymalizacyjne. Pozostaje nierozstrzygnięta kwestia wyboru jednego rozwiązania efektywnego ze zbioru P. Tymczasem w praktyce trzeba

ostatecznie wybrać jedno rozwiązanie lub, co najwyżej, kilka alternatywnych. Jest to możliwe w sytuacji, gdy dysponujemy pewną informacją egzogeniczną o problemie. Można wtedy na zbiorze rozwiązań określić funkcję, która pozwala na jego zawężenie. Jest to zazwyczaj pewna funkcja odchyleń względnych od granic zbioru G^P (rys. 1), nazywana *funkcją kompromisów*.

2. Nieliniowe zadanie dwukryterialne

W większości prac poświęconych rozwiązaniom problemu optymalizacji wielokryterialnej przyjmuje się liniowe funkcje składowe g_i i liniowe ograniczenia oraz z reguły pomija aspekty obliczeniowe. Prezentowane niżej zadanie z nieliniową funkcją składową g_i kryterium wektorowego jest próbą zobrazowania algorytmu opartego na koncepcji Pareto.

Załóżmy, że w procesie przemysłowym są wytwarzane dwa rodzaje produktów I i II przy użyciu trzech składników A, B i C. Zużycie każdego ze składników na jednostkę danego produktu jest następujące:

	A	B	C
I	1	3	2
II	4	4	1

Dostępne zasoby składników wynoszą: A=16, B=21, C=12. Zakładając, że cena produktu I wynosi 1zł/szt., a cena produktu II wynosi 3zł/szt., należy wyznaczyć zużycie składników A, B oraz C, tak aby osiągnąć maksymalny zysk przy minimalnych kosztach magazynowania.

2.1. Rozwiązanie

Oznaczmy przez x_1 i x_2 liczbę jednostek produktów I i II. Wówczas ograniczenia, wynikające z podanych wyżej zasobów składników A, B i C będą miały postać:

$$\left.\begin{array}{ll} h_1: & x_1 + 4x_2 \le 16 \\ h_2: & 3x_1 + 4x_2 \le 21 \\ h_3: & 2x_1 + x_2 \le 12 \\ & x_1 \ge 0, x_2 \ge 0 \end{array}\right\} D \tag{7}$$

Załóżmy, że koszty utrzymania zasobów zadane są funkcją: $g_0(\underline{x}) = \frac{(x_1-5)^2}{4} + (x_2-3)^2 - 16$,

a przychód ze sprzedaży jest funkcją liniową zależną od podanych wyżej cen jednostkowych: $g_2(\underline{x}) = 1x_1 + 3x_2$.

Jeżeli w miejsce $g_0(\underline{x})$ weźmiemy $g_1(\underline{x}) = -g_0(\underline{x})$, to zadanie sprowadzi się do maksymalizacji kryterium wektorowego

$$\underline{g}(\underline{x}) = \left[\underline{g}_1(\underline{x}) \quad g_2(\underline{x})\right] \to \max, \quad \underline{x} \in D \tag{8}$$

przy zadanych ograniczeniach (7).

Dla określenia zbioru rozwiązań efektywnych P wyznaczamy najpierw maksima utopijne funkcji $g_1(\underline{x})$ i $g_2(\underline{x})$ oraz zbiór kompromisów.

1. Maksimum utopijne $g_2 = \hat{g}_2 = g_2(2{,}5 \quad 3{,}375) = 12{,}625$. (9)

   Zatem punkt $\hat{\underline{x}} = [2{,}5 \quad 3{,}375]$ jest punktem utopijnym dla składowej $g_2(\underline{x})$.

2. Maksimum utopijne $g_1 = \hat{g}_1 = g_1(3{,}6154 \quad 2{,}5385) = 15{,}3077$. (10)

   Zatem punkt $\hat{\underline{x}} = [3{,}6154 \quad 2{,}5385]$ jest punktem utopijnym dla składowej $g_1(\underline{x})$.

3. Jak łatwo sprawdzić, zbiór kompromisów G_p, spełniający warunek Pareto dla rozpatrywanego zadania, będzie położony na prostej ograniczającej $3x_1 + 4x_2 = 21$. Równanie krzywej G_p będzie miało postać:

$$g_1 = -0{,}52g_2^2 + 11{,}68g_2 - 50{,}28 \tag{11}$$

2.2. Funkcje kompromisów

Dla ustalenia jednego z możliwych rozwiązań posłużymy się funkcją kompromisu, która daje jednakową minimalną stratę względem wartości utopijnych.

Niech $\tilde{\underline{x}}$ będzie takim punktem w zbiorze D, w którym odchylenia względne obu kryteriów od ich utopijnych wartości maksymalnych:

$$w_i = \frac{\hat{g}_i - g_i}{\hat{g}_i} \quad i = 1,2 \tag{12}$$

będą jednakowe:

$$\frac{\hat{g}_1 - g_1(\tilde{\underline{x}})}{\hat{g}_1} = \frac{\hat{g}_2 - g_2(\tilde{\underline{x}})}{\hat{g}_2} \tag{13}$$

Ponieważ $$\frac{\hat{g}_i - g_i(\underline{x})}{\hat{g}_i} = 1 - \frac{g_i(\underline{x})}{\hat{g}_i} \tag{14}$$

to z równania: $$1 - \frac{g_1}{\hat{g}_1} = 1 - \frac{g_2}{\hat{g}_2} \tag{15}$$

otrzymamy równanie prostej $g_2 = 0{,}8247 \times g_1$ w przestrzeni kryterialnej.

Prosta ta przecina krzywą G_p (por. wzór (11) i rys.1a) w przestrzeni kryterialnej w punkcie o współrzędnych: $g_1(\underline{\tilde{x}}) = 14{,}8049$ oraz $g_2(\underline{\tilde{x}}) = 12{,}2103$.

Przechodząc do przestrzeni decyzyjnej otrzymujemy:

$$\underline{\tilde{x}} = [\tilde{x}_1 \quad \tilde{x}_2] = [2{,}8318 \quad 3{,}1262]$$

Odchyłki względne obu kryteriów od ich wartości maksymalnych (utopijnych) są w punkcie $\underline{\tilde{x}}$ równe i wynoszą: $w_1 = w_2 = 3{,}28\%$.

3. Implementacja programowa

Opisany w p.2 algorytm rozwiązania dwukryterialnego zadania optymalizacji zakodowano w języku C++, wykorzystując zaprojektowane w tym celu idiomy językowe.

3.1. Obiekty funkcyjne

W programowaniu obiektowym wykorzystującym język C++ *obiekt funkcyjny* lub *funktor* jest idiomem językowym [9]. Logicznie, funktory są to takie wystąpienia klas (obiekty), które zachowują się jak funkcje. Obiekt taki spełnia rolę funkcji, ale może być tworzony, przekazywany jako parametr i manipulowany w podobny sposób, jak zwykłe wystąpienie klasy. Skoro funktory są obiektami określonych klas, to ich elementami mogą być zarówno funkcje składowe, jak i składowe struktury danych (zmienne składowe). Tak więc, w odróżnieniu od „zwykłej" funkcji (składowej lub globalnej) funktor może posiadać „pamięć" (składowe struktury danych). Dzięki temu obiekt funkcyjny może przechowywać pośrednie i końcowe wyniki obliczeń [11].

Konstrukcja klasy generującej funktory jest prosta [9,11,12]. Obiekty takiej klasy będą się zachowywać jak funkcje, jeżeli w deklaracji klasy umieścimy definicję przeciążonego operatora wywołania funkcji, tj. "()". Postać ogólna takiej składowej funkcji operatorowej jest następująca:

```
typ klasa::operator() (argumenty) { Blok funkcji }
```

np.

```
int MyClass::operator()() { /*...*/ }
```

lub

```
double MyClass::operator()(double x) { return x; }
```

Jak stąd wynika, nazwą tego operatora jest operator( ) ( ).

Używanie funktorów jest naturalną konsekwencją podejścia obiektowego: zamiast operować funkcjami i pisać programy proceduralne operujemy obiektami, które mogą być albo modelami obiektów fizycznych, albo koncepcjami abstrakcyjnymi.

3.2. Funktory i optymalizacja

Dla dyskutowanego w tej pracy problemu wygodne okazało się zaprojektowanie funktorów zgodnie z poniższymi deklaracjami klas:

```
class Kryterium1 {
public:
     double operator() (double b1, double b2)
     { return fu1(b1,b2); }
     double fu1(double d1, double d2)
     { /* Oblicza ekstremum utopijne g1 */ }

private:
     double x11,x12,g1u;
     friend class Kompromisy;
};
class Kryterium2 {
public:
     double operator() (double b1, double b2)
     { return fu2(b1,b2); }
     double fu2(double d1, double d2)
     { /* Oblicza ekstremum utopijne g2 */ }
private:
     double x21,x22,g2u;
     friend class Kompromisy;
};
class Kompromisy {
public:
     double stratal(Kryterium1& kr1, Kryterium2& kr2)
        {
            kr1(0.5,0.5);
            kr2(0.5,0.5);
            // Obliczenie rozwiązania kompromisowego
            //i odchylenia względnego w
            return w;
        }
private:
     double x1k,x1k,g1k,g2k,w;

};
```

Funkcje składowe fu1() i fu2() obliczają odpowiednio ekstrema utopijne $\hat{g}_1$ i $\hat{g}_2$ oraz punkty utopijne $\hat{\underline{x}}$. Wartości te są przechowywane w zmiennych składowych x11, x12, g1u, x21, x22 i g2u. Do obliczania ekstremów utopijnych zastosowano zmodyfikowaną metodę gradientową.

Modyfikacja polega na tym, że wewnątrz ograniczeń poruszamy się w kierunku gradientu funkcji kryterialnych; po dojściu do ograniczenia ruch odbywa się po liniach ograniczających aż do osiągnięcia lokalnego ekstremum.

Przykładowe wywołania mogą mieć postać:

```
Kryterium1 g1;
Kryterium2 g2;
Kompromisy g;
g.strata1(g1,g2);
```

Pierwsze trzy instrukcje służą do utworzenia, za pomocą konstruktora domyślnego, trzech obiektów: `g1`, `g2` i `g`. Ostatnia instrukcja: `g.strata1(g1,g2);` jest typowym wywołaniem funkcji składowej obiektu. Można ją traktować jako komunikat o nazwie `strata1`, przesłany przez obiekt `g` do siebie samego. Treścią tego komunikatu są parametry aktualne funkcji `Kompromisy::strata1()`. Parametry te są przekazywane przez referencję, dzięki czemu unikamy kopiowania całych obiektów [14]. Wykonanie bloku funkcji `strata1()` przebiega następująco:

Najpierw zostaje wywołany funktor `kr1(0.5,0.5)`. Parametry `(0.5,0.5)` zostają przekazane do funkcji operatorowej: `operator()(double,double)` klasy `Kryterium1`, która wywołuje (w instrukcji `return`) funkcję `fu1(double,double)`, a ta wykonuje opisane wyżej obliczenia. Identycznie przebiega wywołanie funktora `kr2(0.5,0.5)`. Ostatnią operacją wykonywaną przez funkcję `strata1()` jest wyznaczenie wektora $\underline{\tilde{x}} = [\tilde{x}_1 \quad \tilde{x}_2]$ i obliczenie odchylenia względnego dla wybranej funkcji kompromisu.

Wprowadzone tutaj funktory można, w razie potrzeby, wołać bezpośrednio, np.

```
double d = g1(0.5,0.5);
```

LITERATURA

1. Zadeh L.A.: Optimality and Non-Scalar-Valued Performance Criteria. IEEE Trans. on Automatic Control, January 1963, s. 59-60.
2. Opial Z.: Algebra wyższa. PWN, Warszawa 1972.
3. Liu P.L., Leitmann G.: Compromise Solutions, Domination Structures and Salukwadze's Solution. Journal of Optimization Theory and Applications, vol. 13, No. 3, 1974.
4. Peschel M., Riedel C.: Polioptymalizacja. Metody podejmowania decyzji kompromisowych w zagadnieniach inżynieryjno-technicznych. WNT, Warszawa 1979.
5. Konarzewska-Gubała E.: Programowanie przy wielorakości celów. PWN, Warszawa 1980.

6. Ameliańczyk A.: Optymalizacja wielokryterialna w problemach sterowania i zarządzania. Ossolineum, Warszawa 1984.

7. Porębski W.: Dwukryterialne zadanie optymalizacji. Zesz. Nauk. Pol. Gd., Elektronika LXI, 1985, Nr. 393, s. 29-40.

8. Galas Z., Nykowski I., Żółkiewski Z.: Programowanie wielokryterialne. PWE, 1987.

9. James O. Coplien: Advanced C++ Programming Styles and Idioms. Addison-Wesley, 1992.

10. Vincke P.: Multicriteria Decision-Aid. J. Wiley, 1992.

11. Barton J.J., Nackman L.R.: Scientific and Engineering C++. Addison-Wesley, 1994.

12. Horstmann C.S.: Mastering Object-Oriented Design in C++. Wiley, 1995.

13. Koenig A.: Function objects, templates, and inheritance. Journal of Object-Oriented Programming 8(5):65-84, 1995.

14. Porębski W.: Programowanie w języku C++. Komputerowa Oficyna Wydawnicza HELP, Warszawa 1995.

15. Alger J.: Secrets of the C++ Masters. AP Professional, Cambridge, MA 1995.

Recenzent: Dr hab.inż. Andrzej Świerniak, prof.Pol.Śl.

Wpłynęło do Redakcji do 30.06.1996 r.

Abstract

In the paper a multicriteria decision making problem is considered. Necessary conditions for Pareto optimality and the main features of the set of Pareto solutions are discussed. As an application of general conditions, a nontrivial, nonlinear bicriterion problem with given inequality constraints is solved. The solution process has been subdivided into two parts:

The first part includes the problem formulation and the problem solving algorithm. Its outcome are utopia points and the compromise set. Then the utopia points are used in a selected compromise function to obtain the compromise solution both in the criteria space and in the decision space.

The second part is concerned with new computational ideas applied to the problem under consideration. The solution to the problem is shown in terms of object-oriented approach and implemented in C++ language. In this context, the construct of function objects, or *functors*, is given and used in searching the utopia points. Another object that sends messages has been used to find the compromise solution.