Kierunek Inżynieria Obliczeniowa	**Temat laboratoriów** Ćwiczenie 5- Optymalizacja wielokryterialna	**Data ćwiczenia** 11.12.2024, 18.12.2024
Przedmiot Optymalizacja	Karolina Kurowska Szymon Majdak Amelia Nalborczyk	**Grupa** 2

Cel

Celem ćwiczenia jest zapoznanie się z problematyką optymalizacji wielokryterialnej i wyznaczenie rozwiązań minimalnych w sensie Pareto.

Przebieg ćwiczenia

Zadanie 1 funkcja testowa celu

Na ćwiczeniach zaimplementowano algorytmy metod optymalizacji wielokryterialnej:

1. Algorytm Powella

```
solution Powell(matrix(*ff)(matrix, matrix, matrix), matrix x0, double epsilon, int Nmax, matrix ud1, matrix ud2)
{
    try
    {
        solution::clear_calls();

        int n = get_len(x0);
        matrix d = ident_mat(n);
        solution Xi, h;
        solution p;
        Xi.x = x0;
        matrix aw = ud1;
        matrix con(n, 2);
        double* exp_result;

        while (true) {
            p = Xi;
            for (int j = 0; j < n; j++) {
                con.set_col(p.x, 0);
                con.set_col(d[j], 1);
                exp_result = expansion(ff, 0, 1, 1.2, Nmax, aw, con);
                h = golden(ff, exp_result[0], exp_result[1], epsilon, Nmax, aw, con);
                p.x = p.x + h.x * d[j];
            }
            if (norm(p.x - Xi.x) < epsilon) {
                Xi.fit_fun(ff, aw, ud2);
                Xi.flag = 1;
                return Xi;
            }
            for (int j = 0; j < n - 1; ++j) {
                d.set_col(d[j + 1], j);
            }
            d.set_col(p.x - Xi.x, n - 1);
            con.set_col(p.x, 0);
            con.set_col(d[n - 1], 1);
            exp_result = expansion(ff, 0, 1, 1.2, Nmax, aw, con);
            h = golden(ff, exp_result[0], exp_result[1], epsilon, Nmax, aw, con);
            Xi.x = p.x + h.x * d[n - 1];


            if (solution::f_calls > Nmax) {
                Xi.flag = -1;
                cout << "Error: f_calls > Nmax" << endl;
                break;
            }
        }


    }
    catch (string ex_info)
    {
        throw ("solution Powell(...):\n" + ex_info);
    }
}
```

2. Funkcja testowa

```
matrix ff5T_f1(matrix x, matrix ud1, matrix ud2) {
    matrix f1 = ud1(0) * (pow(x(0) - 2, 2) + pow(x(1) - 2, 2));
    return f1;
}

matrix ff5T_f2(matrix x, matrix ud1, matrix ud2) {
    matrix f2 = (1.0 / ud1(0)) * (pow(x(0) + 2, 2) + pow(x(1) + 2, 2));
    return f2;
}
```

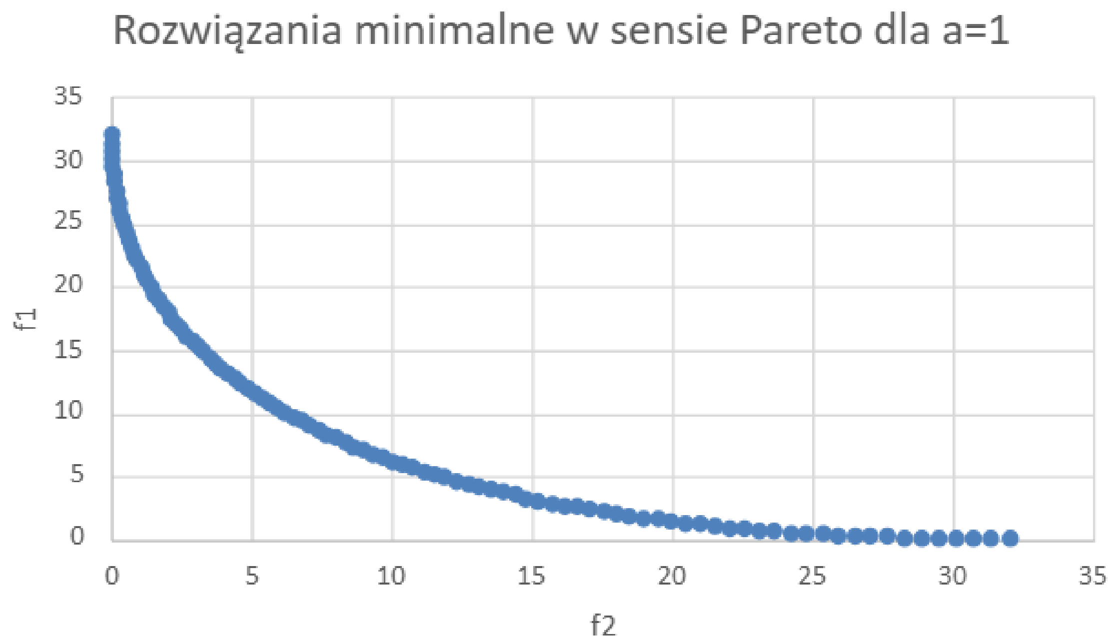
3. Fragment funkcji main zapewniającej zapis wszystkich potrzebnych danych z problemu testowego do jednego pliku csv.

```cpp
void lab5()
{
    //SEKCJA TOSTOWA
    ofstream file("lab_05_tostowe.csv");
    if (!file.is_open())throw string("Nie da sie otworzyc pliku csv");
    file << "x0_1, x0_2, x1_a1, x2_a1, f1_a1, f2_a1, f_calls_a1, "
        << "x1_a2, x2_a2, f1_a2, f2_a2, f_calls_a2, "
        << "x1_a3, x2_a3, f1_a3, f2_a3, f_calls_a3, \n";

    double a1 = 1.0;
    double a2 = 10.0;
    double a3 = 100.0;

    double epsilon = 1e-3;
    int Nmax = 10000;

    matrix aw = matrix(2, 1);
    matrix x0(2, 1);

    solution powell;
    matrix f1, f2;

    for (double w = 0.00; w <= 1.01; w += 0.01) {
        aw(1) = w;

        x0 = 20 * rand_mat(2, 1) - 10;
        file
            << x0(0) << ","
            << x0(1) << ",";

        aw(0) = a1;
        //cout << x0 << endl << endl;
        powell = Powell(ff5T, x0, epsilon, Nmax, aw);
        //cout << powell << endl << endl;
        f1 = ff5T_f1(powell.x, aw);
        f2 = ff5T_f2(powell.x, aw);
        //cout << f1 << endl;
        //cout << f2 << endl;

        file
            << powell.x(0) << ","
            << powell.x(1) << ","
            << f1(0) << ","
            << f2(0) << ","
            << solution::f_calls << ",";
```

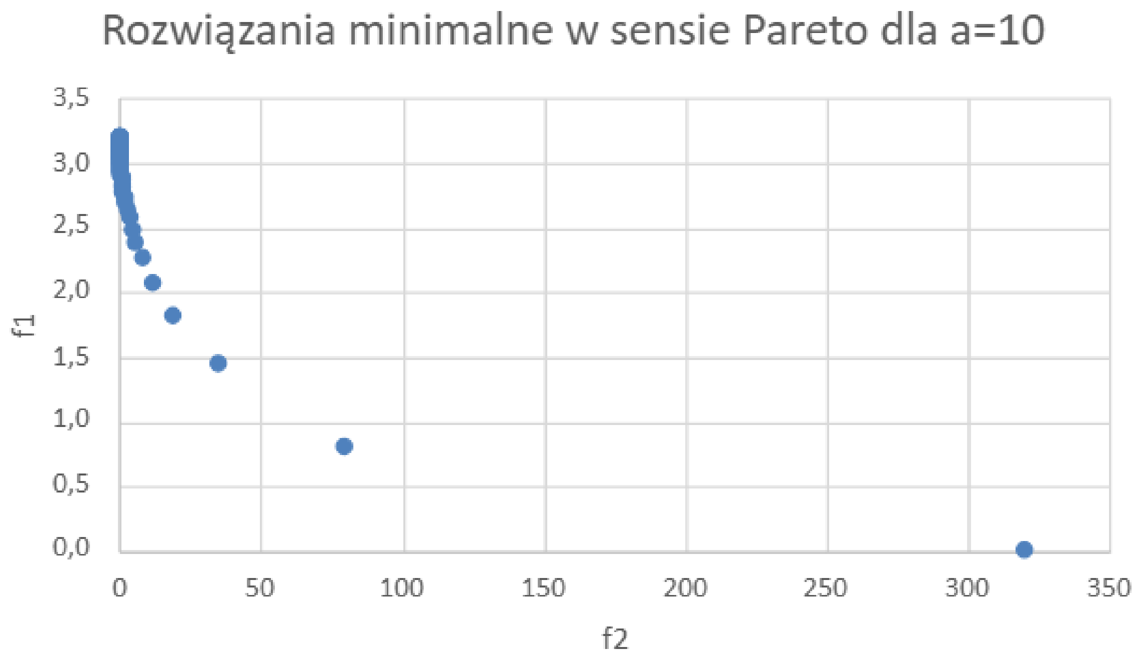
```
        aw(0) = a2;
        //cout << x0 << endl << endl;
        powell = Powell(ff5T, x0, epsilon, Nmax, aw);
        //cout << powell << endl << endl;
        f1 = ff5T_f1(powell.x, aw);
        f2 = ff5T_f2(powell.x, aw);
        //cout << f1 << endl;
        //cout << f2 << endl;

        file
            << powell.x(0) << ","
            << powell.x(1) << ","
            << f1(0) << ","
            << f2(0) << ","
            << solution::f_calls << ",";

        aw(0) = a3;
        //cout << x0 << endl << endl;
        powell = Powell(ff5T, x0, epsilon, Nmax, aw);
        //cout << powell << endl << endl;
        f1 = ff5T_f1(powell.x, aw);
        f2 = ff5T_f2(powell.x, aw);
        //cout << f1 << endl;
        //cout << f2 << endl;

        file
            << powell.x(0) << ","
            << powell.x(1) << ","
            << f1(0) << ","
            << f2(0) << ","
            << solution::f_calls << ","
            << "\n";

    }

    file.close();
}
```

Wykonujemy zadanie polegające na przeprowadzeniu 101 optymalizacji (dla w = {0, 0.01, 0.02, ... ,1}) dla każdej wartości parametru a = {1,10,100} startując z losowego punktu początkowego. Optymalizacja była przeprowadzana metodą Powella, w której minimalizację na kierunku realizowano metodą złotego podziału, a początkowy przedział wyszukiwania ustalano metodą ekspansji.

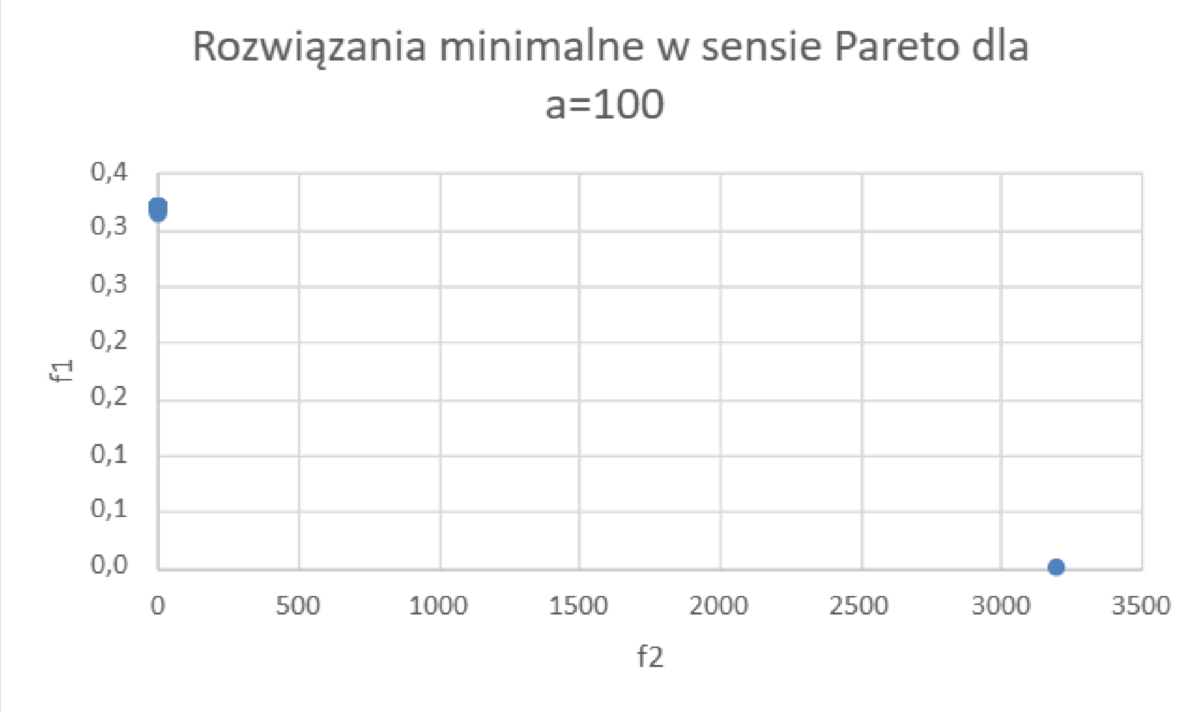
Wyniki optymalizacji zebrano w arkuszu "Tabela 1" w pliku Excel, a rozwiązania minimalne w sensie Pareto dla każdej wartości przedstawiono na wykresach, które zamieszczamy poniżej.



Przy a=1 różnice w skalach funkcji celu są niewielkie, co prowadzi do bardziej równomiernego rozłożenia rozwiązań na wykresie. Rozwiązania Pareto układają się w gładką krzywą, co oznacza płynny kompromis między wartościami funkcji f1, a f2. Gdy f1 rośnie, f2 maleje i odwrotnie.



Wzrost wartości a zwiększa wpływ funkcji f1 na wyniki optymalizacji, co powoduje większą dysproporcję w rozkładzie rozwiązań. Rozwiązania Pareto zaczynają koncentrować się w obszarze niskich wartości f2 i wysokich f1. Krzywa staje się mniej równomierna, a kompromis między kryteriami jest bardziej złożony.



Przy a=100 różnica skali między funkcjami celu sprawia, że optymalizacja koncentruje się głównie na minimalizacji f1, marginalizując wpływ f2. Rozwiązania Pareto przesuwają się znacząco w kierunku minimalizacji funkcji f1. Obszar niskich wartości f2 jest praktycznie pomijany, co oznacza, że wysoka wartość a silnie faworyzuje f1.

Zadanie 2 problem rzeczywisty

Następnym zadaniem jest przeprowadzenie symulacji. Problem rzeczywisty dotyczy optymalizacji belki obciążonej siłą, mającej na celu minimalizację jej masy i ugięcia przy określonych ograniczeniach. Belka ma zmienną długość l i średnicę d, a materiał, z którego jest wykonana, ma określoną gęstość. Optymalizacja polega na znalezieniu najlepszych wartości l i d, które spełniają ograniczenia dotyczące maksymalnego ugięcia i naprężenia.

Ugięcie belki pod wpływem działania siły wynosi:

$$u = \frac{64 \cdot P \cdot l^3}{3 \cdot E \cdot \pi \cdot d^4}$$

Występujące naprężenie wynosi:

$$\sigma = \frac{32 \cdot P \cdot l}{\pi \cdot d^3}$$

W celu rozwiązania tego problemu stworzyliśmy funkcję odwzorowujące dane równania, które zostały zapisane w pliku user_funs.cpp. Funkcję tą zamieszczamy poniżej:

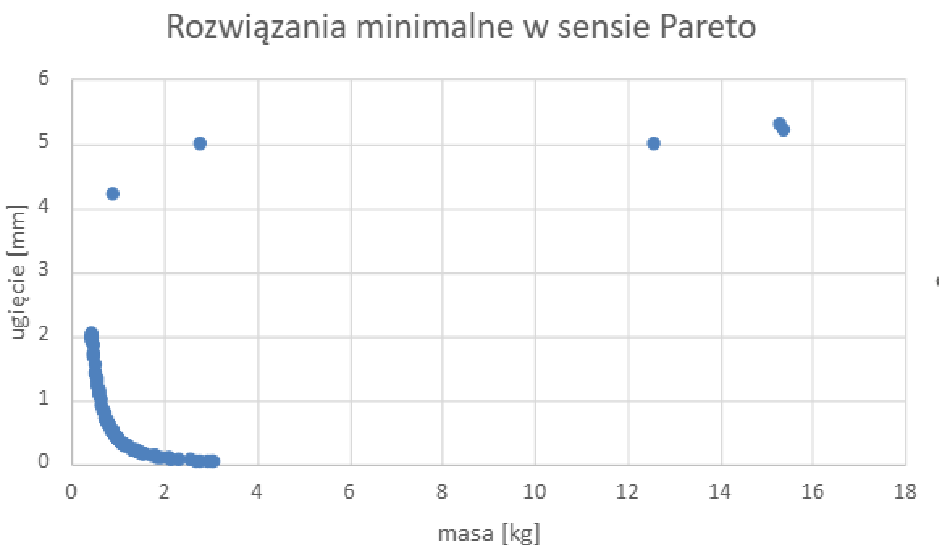
```cpp
matrix ff5R (matrix x, matrix ud1, matrix ud2)
{
    matrix Y;
    double l = x(0);         // Długość [m]
    double d = x(1);         // Średnica [m]

    if (isnan(ud2 (0, 0))) {
        double P = 1000.0;                  // Działająca siła [N]
        double E = 207e9;                   // Moduł Younga [Pa]
        double ro = 7800.0;                 // Gęstość belki [kg/m3]

        Y = matrix (3, 1);

        Y (0) = (ro * l * M_PI * pow (d, 2) / 4);                                       //Masa [kg]
        Y (1) = ((64 * P * pow (l, 3)) / (3 * E * M_PI * pow (d, 4)));                  //Ugiecie [m]
        Y (2) = (32 * P * l) / (M_PI * pow (d, 3));                                     //Naprezenie [Pa]
    }
    else {
        matrix y_pos;
        matrix x_pos = ud2[0] + x * ud2[1];
        y_pos = ff5R (x_pos, ud1);

        //Normalizacja wyników (wybane przez nas wartosci)
        Y = ud1 * (y_pos (0) - 0.1) / (5.0 - 0.1) + (1 - ud1) * (y_pos (1) - 0.00005) / (0.005 - 0.00005);
        double curry = 1e10;

        //ograniczenia dla długości belki <200, 1000 mm> -> m
        if (x_pos (0) < 0.2) {
            Y = Y + curry * pow (0.2 - x_pos (0), 2);
        }
        if (x_pos (0) > 1) {
            Y = Y + curry * pow (x_pos (0) - 1, 2);
        }

        //ograniczenia dla średnicy belki <10, 50 mm> -> m
        if (x_pos (1) < 0.01) {
            Y = Y + curry * pow (0.01 - x_pos (1), 2);
        }
        if (x_pos (1) > 0.05) {
            Y = Y + curry * pow (x_pos (1) - 0.05, 2);
        }

        //ograniczenie dla ugięcia belki max 5 mm -> m
        if (y_pos (1) > 0.005) {
            Y = Y + curry * pow (y_pos (1) - 0.005, 2);
        }

        //Ograniczenie dla naprężenia max 300MPa
        if (y_pos (2) > 300e6) {
            Y = Y + curry * pow (y_pos (2) - 300e6, 2);
        }
    }
    return Y;
}
```

Całość rozwiązania w funkcji main wygląda następująco:

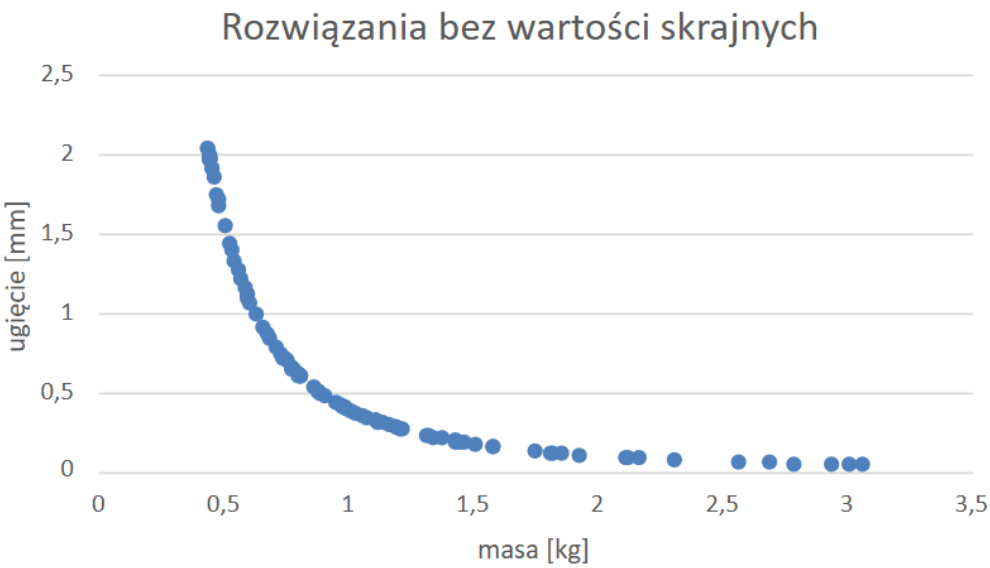
```cpp
//FUNKCJA RZECZYWISTA
matrix x_0(2, 1);        // Macierz startowa
double epsilon = 1e-3;   // Dokładność
int N_max = 10000;       // Maksymalna liczba iteracji

std::ofstream Sout ("symulacja_lab5.csv");

Sout << "w,l(0) [mm],d(0) [mm],l* [mm],d* [mm],masa* [kg],ugięcie* [mm],Liczba wywołań funkcji celu\n";

for (double waga = 0.0; waga <= 1.01; waga += 0.01) {
    x_0(0) = 800 * m2d (rand_mat ()) + 200;  // Długość l <200, 1000> [mm]
    x_0(1) = 40 * m2d (rand_mat ()) + 10;    // Średnica d <10, 50> [mm]

    solution wyn_opt = Powell (ff5R, x_0, epsilon, N_max, waga);

    // Zapis wyników do pliku
    Sout << waga << ","
        << x_0 (0) << ","              // l(0) [mm]
        << x_0 (1) << ","              // d(0) [mm]
        << wyn_opt.x (0) << ","        // l2   [mm]
        << wyn_opt.x (1) << ","        // d2   [mm]
        << wyn_opt.y (0) << ","        // masa [kg]
        << wyn_opt.y (1) << ","        // ugięcie [mm]
        << solution::f_calls << "\n"; // Liczba wywołań funkcji celu

    solution::clear_calls ();
}
Sout.close ();
```

W celu rozwiązania zadania przeprowadziliśmy 101 optymalizacji dla *w* w przedziale [0,1] startując z losowego punktu początkowego. Optymalizacja jest przeprowadzana za pomocą metody Powella, z uwzględnieniem zewnętrznych funkcji kary dla ograniczeń.

Wyniki tej optymalizacji zamieszczamy w arkuszu "Tabela 2" w pliku Excel. Na podstawie danych tworzymy również wykres przedstawiający rozwiązania minimalne w sensie Pareto, który zamieszczony zostaje poniżej.



Wykres przedstawia zbiór rozwiązań minimalnych w sensie Pareto dla dwóch kryteriów: masy [kg] oraz ugięcia [mm]. Na osi poziomej (X) znajduje się masa konstrukcji, a na osi pionowej (Y) – ugięcie. Punkty na wykresie pokazują, że zmniejszenie masy konstrukcji prowadzi do zwiększenia jej ugięcia. Rozwiązania znajdujące się na krzywej Pareto są rozwiązaniami optymalnymi, ponieważ dla każdego z nich nie można poprawić jednego z kryteriów bez pogorszenia drugiego.



Stworzyliśmy również drugi wykres, na podstawie poprzedniego. W porównaniu do poprzedniego wykresu, usunięto punkty, które znacząco odbiegały od pozostałych (o wysokiej masie lub dużym ugięciu), aby lepiej zobrazować obszar, w którym znajduje się większość optymalnych rozwiązań. Eliminacja wartości skrajnych pozwala ograniczyć wpływ nietypowych rozwiązań, które mogą być mało użyteczne lub nieodpowiednie w kontekście założonych kryteriów projektowych.

Wnioski

Przeprowadzona optymalizacja wielokryterialna pozwoliła wyznaczyć rozwiązania minimalne w sensie Pareto, które obrazują kompromis między masą konstrukcji a jej ugięciem. Wyniki wskazują, że wzrost parametru a w funkcji testowej prowadzi do dominacji jednego z kryteriów, co skutkuje nierównomiernym rozkładem rozwiązań Pareto i bardziej złożonym kompromisem między celami. W przypadku rzeczywistego problemu optymalizacji belki usunięcie wartości skrajnych umożliwiło lepszą analizę kluczowego obszaru rozwiązań. Zastosowana metoda Powella, wsparta złotym podziałem i ekspansją, okazała się skuteczna w wyznaczaniu rozwiązań optymalnych. Uzyskane wyniki podkreślają znaczenie kompromisów w projektowaniu konstrukcji, a krzywe Pareto są cennym narzędziem wspierającym wybór optymalnych parametrów przy różnych priorytetach projektowych.