1 Metody optymalizacji wielokryterialnej.

1.1 Ogólna charakterystyka problemu.

Optymalizację wielokryterialną najprościej przedstawić w odniesieniu do optymalizacji jednokryterialnej, w której to funkcja celu zwraca jedną wartość. Wiele problemów podejmowania decyzji, projektowania itp. trudno jest jednak sformułować jako takie zdanie, gdyż zamiast jednego liczbowego kryterium oceny, musimy niekiedy uwzględnić cały ich zbiór.
Co za tym idzie rozwiązaniem takiego problemu wielokryterialnego nie jest jeden punkt, tylko zbiór punktów (rysunek 1).
W celu lepszego uświadomienia różnicy pomiędzy optymalizacją wielokryterialna a jednokryterialną wprowadźmy parę prostych przykładów:

Przykład 1. Kupno optymalnego samochodu (pod uwagę bierzemy cenę, wyposażenie, moc silnika, zużycie paliwa…)

Przykład 2. Konstruktor musi zaprojektować urządzenie, tak aby było tanie w produkcji oraz wysokiej jakości

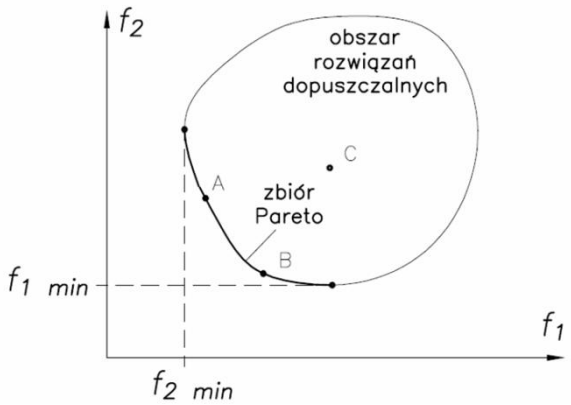
Przykład 3. Projektowanie pręta wspornikowego o jak najmniejszej masie i jak najmniejszym ugięciu końca.

W przedstawionych wyżej przykładach z reguły nie będzie możliwe znalezienie takiego rozwiązana dla którego wszystkie kryteria będą optymalne. Dzieje się tak ponieważ w większości przypadków kryteria nie są ze sobą zgodne (minimalizacja jednego z nich może powodować wzrost innych), o takiej sytuacji mówimy że jedno kryterium dominuje nad innym co możemy zapisać symbolicznie w następujący sposób:

$$x \succ y \quad \Leftrightarrow \exists i \quad f_i(x) \leq f_i(y),$$

słownie :
rozwiązanie x dominuje nad rozwiązaniem y wtedy i tylko wtedy gdy wartość funkcji celu dla x nie jest większa niż dla rozwiązania y w przypadku minimalizacji funkcji celu.

*Naturalny staje się zatem podział możliwych rozwiązań zadania optymalizacji wielokryterialnej na **zdominowane** i **niezdominowane**.*
Rozwiązanie jest niezdominowane (paretooptymalne) wówczas gdy, nie jest możliwe znalezienie rozwiązania lepszego z uwagi na co najmniej jedno kryterium bez pogorszenia z uwagi na pozostałe. Graficznie zasadę tą przedstawiono na rysunku 1.



Rysunek 1 Zbiór Pareto dla zadania minimalizacji

Rozwiązanie C może zostać polepszone zarówno wobec kryterium f_1, jak i f_2. Dla rozwiązań A i B taka możliwość nie istnieje – poprawa względem jednego kryterium powoduje pogorszenie z uwagi na drugie – należą one zatem do zbioru rozwiązań optymalnych w sensie Pareto.

Rozwiązania zdominowane i niezdominowane możemy również zdefiniować w następujący sposób:
*Dla zadania **minimalizacji** zestawu k funkcji celu:*

$$f(x)=(f_1(x), f_2(x),...,f_i(x))$$

*Rozwiązanie x jest **zdominowane**, jeśli istnieje dopuszczalne rozwiązanie y nie gorsze od x, tzn. dla każdej funkcji celu f_i:*

$$f_i(x) \geq f_i(y); \quad (\text{dla } i=1,... k)$$

*W przeciwnym wypadku x jest rozwiązaniem **niezdominowanym** (paretooptymalnym).*

*Dla zadania **maksymalizacji** zestawu k funkcji celu:*

$$f(x)=(f_1(x), f_2(x),...,f_i(x))$$

Rozwiązanie x jest zdominowane, jeśli istnieje dopuszczalne rozwiązanie y nie gorsze od x, tzn. dla każdej funkcji celu f_i:

$$f_i(x) \leq f_i(y); \quad (\text{dla } i=1,... k)$$

W przeciwnym wypadku x jest rozwiązaniem niezdominowanym.

1.2 Tradycyjne metody optymalizacji wielokryterialnej.

1.2.1 Metoda ważonych kryteriów.

Weighted Objectives Method

Polega ona na sprowadzeniu optymalizacji wielokryterialnej do jednokryterialnej przez wprowadzenie kryterium zastępczego, będącego sumą ważoną kryteriów:

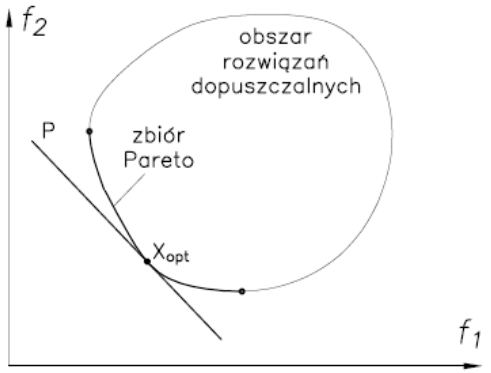
$$F(x) = \sum_{i=1}^{k} w_i f_i(x)$$

gdzie:

k – ilość funkcji celu;
x – wektor rozwiązań;
w_i – wagi takie, że:

$$\boldsymbol{w \in [0,1] \ oraz \ \sum_{i=1}^{k} w_i = 1}$$

W wyniku takiego przekształcenia uzyskujemy pojedynczą funkcję celu F(x), którą optymalizujemy przy użyciu metod typowych dla zadania jednokryterialnego. Graficznie rozwiązanie można przedstawić jako punkt przecięcia obszaru rozwiązań dopuszczalnych z hiperprostą P, zależną od wartości ważności kryteriów w_i (rysunek 2). Problematyczny w tej metodzie jest wybór wartości wag kryteriów, co w oczywisty sposób może prowadzić do różnych rozwiązań.
Niewątpliwą zaletą tej chyba najczęściej stosowanej metody jest łatwy sposób implementacji.



Rysunek 2. Rozwiązanie optymalne w metodzie ważonych kryteriów

1.2.2 Metoda optymalizacji hierarchiczne.

Hierarchical Optimization Metod

Polega ona, podobnie jak metoda przedstawiona poprzednim punkcie, na sprowadzeniu optymalizacji wielokryterialnej do optymalizacji jednokryterialnej kolejno wykonywanej względem wszystkich kryteriów. W tym celu należy wykonać następujące procedury:

Uszeregować kryteria od najważniejszego (f_1) do najmniej ważnego (f_m).
Znaleść rozwiązanie optymalne $\boldsymbol{X_1}$ względem kryterium f_1
i pierwotnych ograniczeniach.
Poszukiwać rozwiązań optymalnych $\boldsymbol{X}_i$, i = 2, 3, ...,M względem pozostałych kryteriów przy wprowadzaniu dodatkowych ograniczeń:

$$\boldsymbol{f}_{i-1}(\boldsymbol{X}) \leq (\mathbf{1} \pm \boldsymbol{\varepsilon}_{i-1}) \cdot \boldsymbol{f}_{i-1}(\boldsymbol{x}_{i-1})$$

Gdzie:
ε *- jest to procentową wartością wariancji dozwoloną dla funkcji kryterialnej f_i. Wartość ta jest swoistą ważnością obliczonego w poprzednim kroku postępowania optimum. Wartość wariancji może również przyjmować wartości równe zero, wtedy taką metodę wielokryterialną nazywa się* ***metodą leksykograficzną***
(ang. Lexicographic Method).

1.2.3 Metoda ograniczonych kryteriów.

Trade-Off Metod

W tej metodzie z kolei są ustalane poziomy wartości, jakie mogą przyjmować poszczególne kryteria co prowadzi do ograniczenia przestrzeni rozwiązań dopuszczalnych. Problem optymalizacji wielokryterialnej jest sprowadzany do problemu optymalizacji względem wybranego kryterium f_r przy zwiększonej o (M-1) liczbie ograniczeń wynikających z pozostałych kryteriów, co matematycznie można zapisać następująco:

$$\begin{aligned} & f_r(X) \to MIN \\ & f_i(X) \leq \varepsilon_i; i = 1,...,M; i \neq r \\ & g_k(X) \leq 0; k = 1,...,K \\ & h_j(X) = 0; \; j = 1,...,J \end{aligned}$$

Gdzie:
ε_i *- wartość poziomu ograniczającego dane kryterium,*
$g_k(X)$ *– pierwotne ograniczenia nierównościowe,*
$h_j(X)$ *– pierwotne ograniczenia równościowe.*

1.2.4 Metoda kryterium globalnego.

Global Criterion Method

*W metodzie tej poszukuje się rozwiązania przybliżonego **F(X*)** (może nim być rozwiązanie stanowiące ekstremum dla poszczególnych kryteriów rozpatrywanych osobno) dla sformułowania kryterium dla optymalizacji jednokryterialnej o postaci:*

$$\sum_{i=1}^{M}\left(\frac{f_i(X*)-f_i(X)}{f_i(X*)}\right)^P \rightarrow MIN$$

przy zachowaniu ograniczeń równościowych i nierównościowych. Wartość wykładnika P jest przyjmowana najczęściej z przedziału (1, 2).

1.2.5 Metody funkcji odległości i Mini-Maxu.

Method of Distance Functions, Min-Max Method

Metody te są zbliżone do metody kryterium globalnego, gdzie również na początku postępowania poszukuje się pewnego rozwiązana przybliżonego (lub idealnego), minimalizując w drugiej fazie funkcję o postaci:

$$\sum_{i=1}^{M}\left[\left(\frac{f_i(X*)-f_i(X)}{f_i(X*)}\right)^P\right]^{\frac{1}{P}} \rightarrow MIN$$

*Jeżeli P = 2 wtedy minimalizujemy odległość między rozwiązaniem przybliżonym a optymalnym (**Metoda funkcji odległości**).*
*Wartość P = ∞ prowadzi do **metody Min-Max –** minimalizacji maksymalnych odchyleń rozwiązania optymalnego od przybliżonego:*

$$\underset{i=1,\ldots,M}{MAX}\left|\frac{f_i(X*)-f_i(X)}{f_i(X*)}\right| \rightarrow MIN$$

Stosuje się również metodą Min-Max z wagami, przypisanymi do poszczególnych kryteriów:

$$\underset{i=1,\ldots,M}{MAX}\left|w_j\,\frac{f_i(X*)-f_i(X)}{f_i(X*)}\right| \rightarrow MIN$$

1.2.6 Metoda programowania celów.

Goal Programming Method

jest pewną ogólną techniką optymalizacji wielokryterialnej. W tym podejściu kryteria są traktowane jako cele które należy osiągnąć lub jako wartości progowe których wartości kryteriów nie mogą przekroczyć. Na wartości kryteriów mogą zostać zatem narzucone warunki: większy lub równy, mniejszy lub równy, równy.Rozważmy problem optymalizacji wielokryterialnej z uwagi na dwie funkcje f_1 i f_2. Przyjmijmy pierwszy cel –kryterium f_1 będzie mniejsze lub równe z_1, oraz drugi – niech f_2 będzie równe z_2. Wtedy zapis metody programowania celów będzie następujący:

$$(\boldsymbol{w_1^+ d_1^+ + w_2^+ d_2^+ + w_2^- d_2^-}) \rightarrow \boldsymbol{MIN}$$
$$\boldsymbol{f_1(X) - d_1^+ \leq z_1}$$
$$\boldsymbol{f_1(X) - d_2^+ + d_2^- = z_2}$$

Z uwzględnieniem ograniczeń równościowych i nierównościowych.
W powyższym wzorze w_i są współczynnikami kary odpowiadającymi każdej odchyłce wartości kryterium d_i które wyznaczają niepożądane odchylenia osiągniętego celu.

Podsumowując tradycyjne metody optymalizacji wielokryterialnej są ciągle bardzo popularne. Jest to spowodowane tym że, dają one dobre rezultaty w znajdywaniu potencjalnych rozwiązań (dla nie wielkiej liczby rozwiązań w sensie Paret) , a także ponieważ wiedza na ich temat i dostępność materiałów jest dzisiaj szeroka. Nie mniej jednak metody te nie pozostają bez wad. O ile dla nie wielkiej ilości zbioru Pareto-optymalnego spisują się całkiem nie źle, to już większy zbiór powoduje w znacznym stopniu wzrost kosztu obliczeń. Dzieje się tak bowiem zazwyczaj metody tradycyjne wymagają kilkukrotnego uruchomienia w celu wyznaczenia zbioru Pareto-optymalnego. Ponadto niektóre techniki takie jak na przykład metoda ważonych kryteriów, są wrażliwe na kształt frontu Pareto-optymalnego.

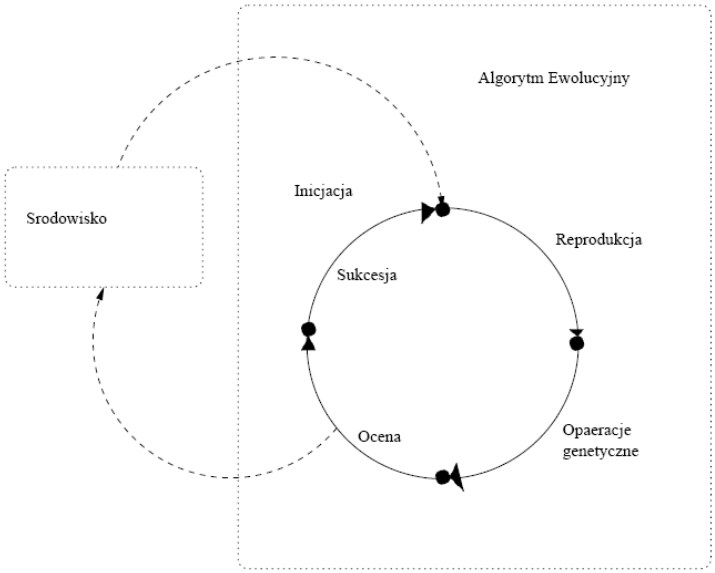
1.3 Ewolucyjne metody optymalizacji wielokryterialnej.

Pomimo ciągłej popularności przedstawionych w poprzednim punkcie tradycyjnych metod optymalizacji wielokryterialnej, coraz częściej ich alternatywą w wielu problemach stają się algorytmy ewolucyjne. Jest to spowodowane tym iż lepiej one radzą sobie w przypadku gdy mamy odczynienia z potencjalnie duża liczbą rozwiązań w sensie Pareto.

1.3.1 Wstępne informacje o algorytmach ewolucyjnych

Definicja algorytmu ewolucyjnego
Algorytm ewolucyjny przeszukuje przestrzeń alternatywnych rozwiązań problemu w celu odnalezienia rozwiązań najlepszych lub potencjalnie najlepszych. Zalicza się go do klasy algorytmów heurystycznych. Przeszukiwanie odbywa się za pomocą mechanizmów ewolucji oraz doboru naturalnego. W praktyce te słowa oznaczają, że wykorzystujemy ewolucję, aby poprzez krzyżowanie i mutacje stworzyć z grupy losowych zestawów danych to, co nas będzie satysfakcjonować. Pierwszy algorytm ewolucyjny został zaproponowany przez Johna Hollanda w 1975 roku, który bardzo interesował się biologią i często czerpał z niej inspirację w swych pracach informatycznych.

Zasada działania:
Algorytm ewolucyjny przetwarza populacje osobników, z których każdy jest propozycją rozwiązania postawionego problemu. Działa on w środowisku, które można zdefiniować na podstawie problemu rozwiązywanego prze z algorytm. W środowisku każdemu osobnikowi jest przyporządkowana określona wartość która cechuje jakość reprezentującego przez niego rozwiązania; wartość ta jest nazwana przystosowaniem osobnika. Każdy osobnik jest wyposażony w informację stanowiącą jego ***genotyp****, na podstawie którego tworzony jest* ***fenotyp*** *– czyli zestaw cech, który podlega ocenie środowiska. Właśnie ta ocena jest przystosowaniem osobnika opisywanym wcześniej. Proces tworzenia fenotypu z genotypu nazywamy* ***kodowaniem*** *. Można powiedzieć, że fenotyp należą do przestrzeni rozwiązań problemu, natomiast genotyp do przestrzeni kodów. Sposób oceniania fenotypu opisujemy za pomocą* ***funkcji przystosowania****, która w sposób całkowity opisuje środowisko. Funkcja ta może być stacjonarna, zmienna w czasie bądź podlegać randomizacji. Genotyp osobnika jest budowany przez* ***chromosomy****. Natomiast każdy chromosom składa się z zestawu genów, których wartościami są* ***allele****. Działanie algorytmu ewolucyjnego sprowadza się do wykonywania pętli pokazanej na rysunku 3. , w której następuji po sobie:* ***reprodukcja, operacje genetyczne, ocena, sukcesja****.*



Rysunek 3 Schemat algorytmu ewolucyjnego.

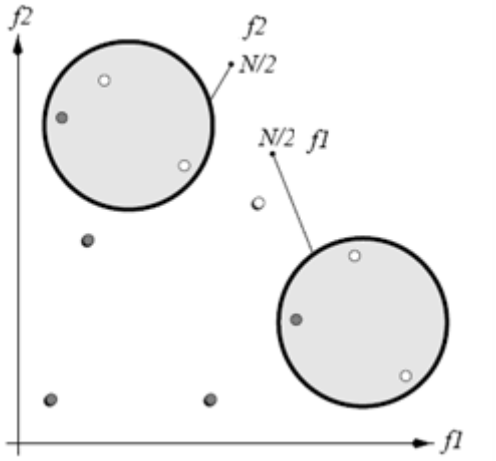
Reprodukcja W połączeniu z operatorami genetycznymi modeluje rozmnażanie, podczas którego materiał genetyczny rodziców jest przekazywany potomkom. Powstałe w wyniku reprodukcji kopie, zwane są osobnikami rodzicielskimi, poddawane są operacją genetycznym (mutacji i krzyżowaniu). Mutacja polega na perturbacji genotypu jednego osobnika rodzicielskiego. Krzyżowanie z kolei działa na wielu osobnikach rodzicielskich i prowadzi do wygenerowania jednego lub wielu osobników potomnych, których chromosomy powstają w wyniku wymieszania odpowiednich chromosomów pochodzących z rożnych osobników rodzicielskich. Osobniki utworzone w wyniku działania operatorów genetycznych stanowią ***populacje potomną****.*

Populacja potomna jest poddawana ocenie środowiska, po czym następuje sukcesja – tworzy się nową populację bazową, mogącą zawierać osobniki zarówno z populacji potomnej jak i ze starej populacji bazowej.

1.3.2 Algorytm VEGA.

Schaffer's Vector Evaluated Genetic Algorithm

Podejście do zagadnienia wielokryterialnego w tym algorytmie przedstawił w 1985 r. Schaffer. Istotą tego podejścia jest nieco inny sposób przeprowadzania selekcji osobników, reprezentujących pojedyncze rozwiązanie danego problemu, w odniesieniu do prostego algorytmu genetycznego. Mianowicie populacja bazowa P_t *dzielona jest na k-podzbiorów , gdzie* k *oznacza liczbę kryteriów, następnie z każdego podzbioru dokonywana jest selekcja takiej samej liczby* (N/k) *najlepszych osobników względem jednego z kryteriów. Wybrane w procesie selekcji osobniki są przenoszone do populacji tymczasowej* P' *gdzie są mieszane i podawane operacją genetycznym: mutacji i krzyżowaniu. Obrazowo proces selekcji dla k=2 przedstawiona na rysunku poniżej.*



Rysunek 4 Selekcja w metodzie VEGA.

Schemat krokowy algorytmu VEGA

Dane podawane na wejście:
N – rozmiar populacji,
T – maksymalna liczba pokoleń,
p_c – prawdopodobieństwo krzyżowania,
p_m – prawdopodobieństwo mutacji
Wyjście algorytmu:
A – zbiór rozwiązań niezdominowanych.

Krok 1: Inicjalizacji:
Niech $P_0=\emptyset$ oraz $t=0$. Następnie dla $i=1,\ldots,N$ wykonaj
a) wybierz osobnika i,
b) dodaj osobnika i do populacji P_0.

Krok 2: Przystosowanie i selekcja:
Ustaw $P'=\emptyset$ oraz $i=1$ i wykonaj:

a) dla każdego osobnika $i \in P_t$ wyznacz jego przystosowanie według wzoru: $F(i) = f_i(m(i))$,
b) dla $j=1,\ldots,N/k$ dokonaj selekcji osobnika i skopiuj go do populacji tymczasowej: $P' = P' + \{i\}$,
c) przypisz $i=i+1$,
d) jeżeli $i \leq k$ przejdź do a) w przeciwnym razie przejdź do Kroku 3.

Krok 3: Rekombinacja:
Ustaw $P''=\emptyset$ i dla każdego $i=1,\ldots,N/2$ wykonaj:
a) wybierz dwa osobniki $i, j \in P'$ a następnie usuń je z P',
b) skrzyżuj osobniki rodzicielskie **i** oraz **j.** Wynikiem tego otrzymujesz osobniki potomne **k, l**,
c) dodaj **k, l** do **P''** z prawdopodobieństwem p_c w przeciwnym razie przenieś osobniki rodzicielskie i, j do P''.

Krok 4: Mutacja:
Ustaw $P'''=\emptyset$ dla każdego $i \in P''$ wykonaj:
a) podaj mutacji osobnika i z prawdopodobieństwem p_m. wyniku mutacji powstaje osobnik j ,
b) dodaj osobnika j do P'''.

Krok 5: Kończenie algorytmu:
Ustaw nowe pokolenie bazowe $P_{t+1}=P'''$ oraz $t=t+1$.
Sprawdź jeżeli $t \geq T$ lub dodatkowe kryterium stopu jest spełnione, umieść rozwiązania niezdominowane z populacji P_t w zbiorze **A** i zakończ działanie algorytmu.
W przeciwnym razie przejdź do Krok 2.

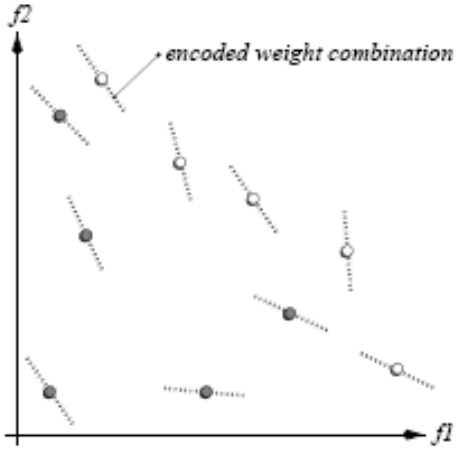
Niewątpliwą zaletą przedstawionego wyżej algorytmu jest jego prosta implementacja. Niestety algorytm ten ma tendencje do poszukiwania „ekstremalnych" rozwiązań niezdominowanych tzw. takich, które znajdują się blisko krańców zbioru rozwiązań niezdominowanych (pomijanie rozwiązań pośrednich). Można zapobiegać temu zjawisku, stosując techniki podziału przystosowania, jednakże istnieje wówczas ryzyko wystąpienia trudności z osiągnięciem rozwiązań niezdominowanych. Innym rozwiązaniem jest ograniczenie wymiany osobników między populacjami – zarówno jej części (nie w każdej populacji), jak i intensywności (nie wszystkie osobniki tylko np. najlepsze).

1.3.3 Algorytm HLGA.

Hajela and Lin's Weighting-based Genetic Algorithm

Algorytm ten opracowali Hajela and Lin w 1992. Podejście to bazuje na przestawionej w punkcie 1.2.1 metodzie ważonych kryteriów. Tutaj jednak wagi nie są przydzielane bezpośrednio do każdego kryterium tylko koduje się je wraz z danym osobnikiem. Dzięki temu każdy osobnik ewoluuje wobec różnej kombinacji wag, inaczej mówiąc każdy osobnik ewoluuje

przy innej funkcji kryterialnej. Powoduje to że optymalizacja jest wykonywana w tym przypadku w wielu kierunkach równocześnie (rysunek 5). Niemniej jednak wady metody przestawionej w punkcie 1.2.1 mogą w znaczący sposób obniżyć efektywność tej ewolucyjnej metody.



Rysunek 5 Selekcja w metodzie HLGA.

Schemat krokowy algorytmu HLGA

Dane podawane na wejście:
N – rozmiar populacji,
T – maksymalna liczba pokoleń,
p_c – prawdopodobieństwo krzyżowania,
p_m – prawdopodobieństwo mutacji
Wyjście algorytmu:
A – zbiór rozwiązań niezdominowanych.

Krok 1: Inicjalizacji:
Niech $P_0=\emptyset$ oraz t=0. Następnie dla i=1,…,N wykonaj
a) wybierz osobnika i,
b) dodaj osobnika i do populacji P_0.

Krok 2: Przystosowanie:
Dla każdego osobnika $\boldsymbol{i} \in \boldsymbol{P_t}$ wykonaj:
a) wydostań wagę $\boldsymbol{w_j\ (j=1,\ldots,k)}$ z osobnika $\boldsymbol{i}$,
b) oblicz przystosowanie osobnika według wzoru:

$$\boldsymbol{F(i) = w_1 \cdot f_1(m(i)) + , \ldots , w_k \cdot f_k(m(i))}.$$

Krok 3: Selekcja:
Ustaw $\boldsymbol{P' = \emptyset}$. Następnie dla każdego i=1,…,N wykonaj:
a) wybierz jednego osobnika $\boldsymbol{i \in P_t}$ pod kontem najlepszej wartości przystosowania F(i).
b) Dadaj do populacji tymczasowej danego osobnika $\boldsymbol{P' = P' + \{i\}}$.

Krok 4: *Rekombinacja:*
Ustaw $P''=\emptyset$ *i dla każdego* $i=1,\ldots,N/2$ *wykonaj:*

a) *wybierz dwa osobniki* $i, j \in P'$ *a następnie usuń je z* ***P'***,
b) *skrzyżuj osobniki rodzicielskie i oraz j. Wynikiem tego otrzymujesz osobniki potomne* ***k, l,***
c) *dodaj* ***k, l*** *do* ***P''*** *z prawdopodobieństwem* p_c *w przeciwnym razie przenieś osobniki rodzicielskie i, j do P''.*

Krok 5: *Mutacja:*
Ustaw $P'''=\emptyset$ *dla każdego* $i \in P''$ *wykonaj:*

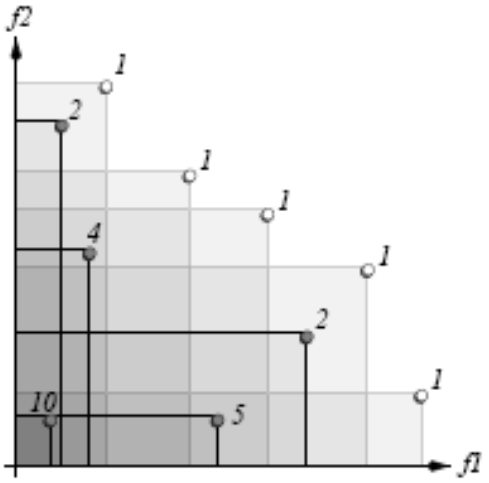
a) *poddaj mutacji osobnika i z prawdopodobieństwem* p_m*. wyniku mutacji powstaje osobnik* ***j*** *,*
b) *dodaj osobnika* ***j*** *do* ***P''***.

Krok 6: *Kończenie algorytmu:*
Ustaw nowe pokolenie bazowe $P_{t+1}=P'''$ *oraz* $t=t+1$.
Sprawdź jeżeli $t \geq T$ *lub dodatkowe kryterium stopu jest spełnione, umieść rozwiązania niezdominowane z populacji* P_t *w zbiorze A i zakończ działanie algorytmu. W przeciwnym razie przejdź do Krok 2.*

1.3.4 Algorytm FFGA.

Fonseca and Fleming's Multiobjective Genetic Algorithm

Algorytm ten został opracowany w 1993 r. przez Fonseca i Fleming, nie mniej koncepcja takiego podejścia została już wcześniej bowiem w 1989 r. przedstawiona przez Goldberga. Bazuje ona na nadawaniu rang osobnikom , które później determinują wartość ich przystosowania. Rangi są nadawane w taki sposób że z populacji bazowej wybierane są osobniki niezdominowane i przypisywana jest im ranga jeden a następnie są one tymczasowo usuwane z populacji. Z pozostałych osobników wybieramy znowu te które są niezdominowane i nadajemy im rangę równą dwa. Postępujemy tak do póki nie nadamy rang wszystkim osobnikom. Należy jednak pamiętać iż nie każda wartość rangi musi zostać nadana.
Na poniższym rysunku została przedstawiona przykładowa populacja z przypisanymi rangami dla poszczególnych osobników. Osobniki które nie są zdominowane mają rangę równą 1, podczas gdy najgorszym przypisano rangę równą 10. Na podstawie tych rang jest wypełniana populacja tymczasowa. Ponad to widzimy że nie ma kilku pośrednich wartości rang, gdyż nie jest to wymagane.



Rysunek 6 Nadanie rang w dwu-kryterialnym zadaniu (Algorytm FFGA).

Schemat krokowy algorytmu FFGA.

Dane podawane na wejście:
N – rozmiar populacji,
T – maksymalna liczba pokoleń,
p_c – prawdopodobieństwo krzyżowania,
p_m – prawdopodobieństwo mutacji
σ_{share} - promień niszy.
Wyjście algorytmu:
A – zbiór rozwiązań niezdominowanych.

Krok 1: Inicjalizacji:
Niech $P_0=\emptyset$ oraz t=0. Następnie dla i=1,…,N wykonaj
wybierz osobnika i,
dodaj osobnika i do populacji P_0.

Krok 2: Przystosowanie:

a) *Dla każdego i ∈ P_t oblicz rangę: $\boldsymbol{r(i,t) = 1 + p_i^{(t)}}$*
*gdzie $\boldsymbol{p_i^{(t)}}$ – liczba osobników dominujących osobnika **i** w danym pokoleniu,*
b) *posortuj populacje według nadanej rangi (od najlepszej do najgorszej),*
c) *przypisz każdemu osobnikowi $\boldsymbol{i \in P_t}$ wartość wstępnego przystosowania $\boldsymbol{F'(i)}$ uzyskaną poprzez liniową interpolację od najlepszej rangi do najgorszej,*
d) *Wyznacz wartość przystosowania $\boldsymbol{F(i)}$ uśredniając wartość funkcji $\boldsymbol{F'(i)}$ dla takich $\boldsymbol{i \in P_t}$ dla których wartości $\boldsymbol{r(i,t)}$ są identyczne.*

Krok 3:
Krok 4:
Krok 5:
Krok 6:

} *Takie same jak w algorytmie HLGA (p. 1.3.2).*

Rozpatrując tego typu metody, bazującej na dominacji należy zwrócić szczególna uwagę na mogący się pojawić problem nierównomiernego pokryciu zbioru rozwiązań niezdominowanych. Problem ten ujawnia się zwykle przy niewielkich populacjach bazowych, gdzie większość osobników, reprezentujących poszczególne rozwiązania, jest zdominowana. Schemat selekcji preferujący niezdominowane osobniki prowadzi do intensywniejszego przeszukania ich sąsiedztwa. W efekcie pojawiają się skupiska osobników wokół uprzedni nie zdominowanych. W skutek tego rozmieszczenie osobników w zbiorze punktów niezdominowaych jest bardzo nierównomierne.

Aby zapowiedz temu zjawisku w powyższej metodzie zastosowano technikę optymalizacji wielomodalnej, podziału przystosowania (fitness sharing). Metoda ta została opracowana przez Goldberg i Richardson w 1987 r. Głowna idea tej metody polega na podzieleniu populacji na stabilne podpopulacje zwane niszami. Funkcja podziału określa zmniejszenie dopasowania osobnika w stosunku do sąsiada oddalonego o pewną miarę odległości $\boldsymbol{d(i,j)}$ *i uzależniona jest promienia niszy* $\boldsymbol{\sigma_{share}}$.

Matematycznie funkcję podziału można to zdefiniować w następujący sposób:

$$s\big(d(i,j)\big) = \begin{cases} 1 - \left(\dfrac{d(i,j)}{\sigma_{share}}\right)^{\propto} & \textbf{jeżeli } d(i,j) < \sigma_{share} \\ 0 & \textbf{w przeciwnym wypadku} \end{cases}$$

gdzie:

$s\big(d(i,j)\big)$ *– funkcja podziału,*

α *- pewna stała wartość.*

Znając powyższą zależność możemy w końcu zdefiniować matematycznie funkcję przystosowań ***F(i):***

$$F(i) = \frac{F'(i)}{\sum_{j \epsilon P} s(d(i,j))}$$

gdzie:

$F'(i)$ **-** *wartość wstępnego przystosowania*

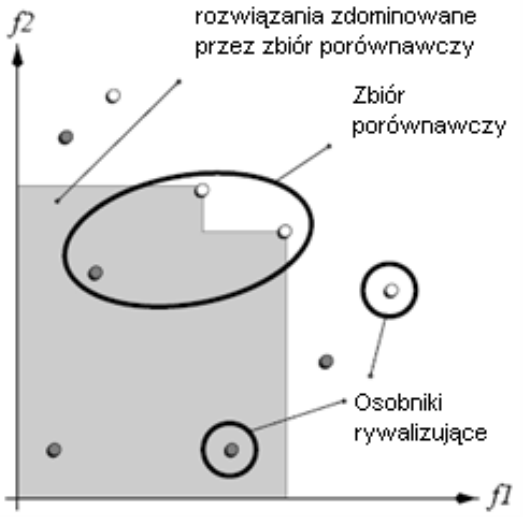
Analizując powyżej wzory z łatwością możemy zauważyć że funkcja przystosowania $F(i)$ *zmniejsza się proporcjonalnie do liczby i bliskości sąsiednich punktów.*

Oznacza to że kiedy sąsiadują ze sobą wielu osobników, to wpływają one na liczebność niszy, w ten sposób obniżając własne wartości dopasowań. Dziki temu ogranicza się właśnie niekontrolowany wzrost osobników danej niszy i zapobiega nierównomiernemu pokryciu zbioru rozwiązań.

1.3.5 Algorytm NPGA.

Niched Pareto Gentic Algorithm

Podejście do rozwiązania zadań wielokryterialnych zawarte w tej metodzie zostało przedstawione przez Horn i Nafpliotis w 1993 r. Algorytm NPGA podobnie jak przestawiony w p. 1.3.3 algorytm FFGA bazuje na technice opartej na dominacji, z jednocześnie wprowadzoną selekcją turniejową. W celu przeprowadzenia selekcji w odpowiedni sposób ,tak aby zachować odpowiedni nacisk selekcji oraz kontrole nad ilością rozwiązaniami niezdominowanych tworzy się losowy zbiór osobników porównawczych $\boldsymbol{P_{dom}}$ *o liczebności określonej przez wartość nacisku dominacji* $\boldsymbol{t_{dom}}$ *. Ten osobnik który zdominuje losowy zbiór* $\boldsymbol{P_{dom}}$ *wygrywa selekcję i zostaje przenoszony do populacji tymczasowej* $\boldsymbol{P'}$*. Proces selekcji graficznie przedstawiono na rysunku poniżej. Na biało zaznaczono zwycięzców turnieju selekcji.*



Rysunek 7 Selekcja w algorytmie NPGA.

Schemat krokowy algorytmu NPGA.

Dane podawane na wejście:
N – rozmiar populacji,
T – maksymalna liczba pokoleń,
p_c – prawdopodobieństwo krzyżowania,
p_m – prawdopodobieństwo mutacji
σ_{share} - promień niszy.
Wyjście algorytmu:
A – zbiór rozwiązań niezdominowanych.

Krok 1: *Inicjalizacji:*
Niech $P_0=\emptyset$ oraz $t=0$. Następnie dla $i=1,\ldots,N$ wykonaj
a) wybierz osobnika i,
b) dodaj osobnika i do populacji P_0.

Krok 2: *Przystosowanie i Selekcja:*
Ustaw $i=1$ oraz $\boldsymbol{P}' = \emptyset$.

a) *Wylosuj dwa osobniki $i, j \in P_t$ do rywalizacji oraz zapełnij losowymi osobnikami zbiór porównawczy $P_{dom} \subseteq P_t$ zgodnie z wartością t_{dom} ,*
b) *Jeżeli osobnik **i** dojmuje nad zbiorem P_{dom} oraz osobnik **j** nie dominuje nad P_{dom} wówczas **i** jest zwycięzcą turnieju i zostaje przeniesiony do populacji tymczasowej $\boldsymbol{P}' = \boldsymbol{P}' + \{\boldsymbol{i}\}$,*
c) *jeżeli zaś osobnik **j** dominuje nad zbiorem P_{dom} oraz osobnik **i** nie dominuje nad P_{dom} to $\boldsymbol{P}' = \boldsymbol{P}' + \{\boldsymbol{j}\}$,*
d) *jeżeli warunki w punkacie b) oraz c) nie są spełnione wówczas:*
    a. *oblicz liczbę osobników w populacji tymczasowej których odległość od osobnika **i** jest mniejsza od promienia niszy, posłuż się wzorem:*

$$\boldsymbol{n}(\boldsymbol{i}) = \left|\left\{\boldsymbol{k} | \boldsymbol{k} \in \boldsymbol{P}' \bigwedge \boldsymbol{d}(\boldsymbol{i},\boldsymbol{k}) < \boldsymbol{\sigma}_{\boldsymbol{share}}\right\}\right|$$

    b. *postępując jak w p. a. oblicz n(j),*
    c. *jeśli n(i)<n(j) wówczas $\boldsymbol{P}' = \boldsymbol{P}' + \{\boldsymbol{i}\}$ w przeciwnym wypadku $\boldsymbol{P}' = \boldsymbol{P}' + \{\boldsymbol{j}\}$.*

e) *Ustaw i=i+1, jeżeli i<N idź do p. a), w przeciwnym razie idź do Krok 3.*

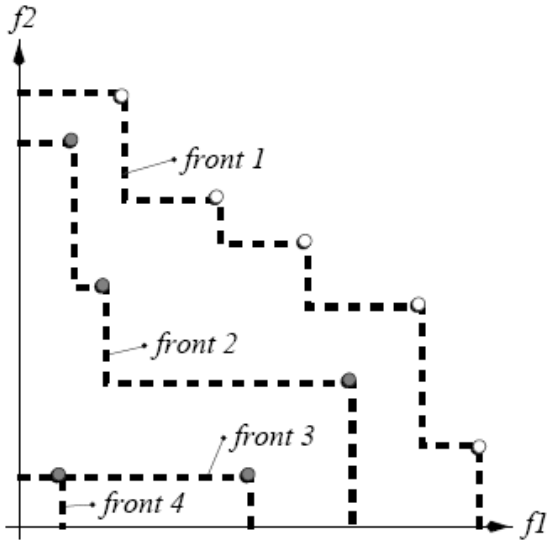
Krok 3:
Krok 4: } *Takie same jak w algorytmie VEGA (p. 1.3.1).*
Krok 5:

1.3.6 Algorytm NSGA.

Nondominated Sorting Genetic Algorithm

Algorytm NSGA po raz pierwszy został zaimplementowany w 1994 r. przez Srinivas i Deb. Metoda ta podobnie jak algorytm przedstawiony w p. 1.3.3 bazuje na koncepcji Goldberga z 1989 r. Idea takiego podejścia skupia się na podzieleniu osobników w podpopulacje, ze względu na nada im rangi, następnie korzystając z metody niszowej zapewnia równomierne pokrycie zbioru rozwiązań. Dokładnie podejście to zostało wyjaśnione w
p. 1.3.3 . Podstawowym celem takiego rozwiązania było wyeliminowanie wady algorytmu VEGA polegającej na pomijaniu pewnych pośrednich rozwiązań a skupianiu się na poszukiwaniu rozwiązań znajdujących się blisko krańców zbioru rozwiązań niezdominowanych.

Takich samych założeń jak w przypadku wcześniej wymienionych metod, jak również podobnego skutku w działaniu metoda ta różni się między innymi nieco zmieniona postacią funkcji przystosowań a co za tym idzie również selekcją (rysunek 8.).



Rysunek 8 Selekcja w algorytmie NSGA.

Jeszcze przed selekcją populacja jest porządkowana pod względem dominacji. Wszystkie osobniki niezdominowane zalicza się do pierwszego frontu (przydzielając im sztuczną wartość funkcji przystosowań F_d *proporcjonalna do liczebności populacji, aby zapewnić tym osobnikom równy potencjał reprodukcyjny). Dla utrzymania różnorodności populacji stosuje się dzielenie tej sztucznej funkcji przystosowań (patrz p. 1.3.3). Następnie tą grupę osobników niezdominowanych ignoruje się tymczasowo, resztę populacji podaje się temu samemu mechanizmowi i wyznacza się drugą warstwę rozwiązań niezdominowanych. Nowemu zbiorowi osobników przypisuje się nową wartość sztucznego*

przystosowania, która jest trochę mniejsza od poprzedniej. Ten proces jest powtarzany dopóki cała populacja nie zostanie sklasyfikowana.

***Schemat krokowy algorytmu NSGA**.*

Dane podawane na wejście:
N – rozmiar populacji,
T – maksymalna liczba pokoleń,
p_c – prawdopodobieństwo krzyżowania,
p_m – prawdopodobieństwo mutacji
σ_{share} - promień niszy.
Wyjście algorytmu:
A – zbiór rozwiązań niezdominowanych.

Krok 1: *Inicjalizacji:*
Niech $P_0=\emptyset$ oraz $t=0$. Następnie dla $i=1,\ldots,N$ wykonaj
a) *wybierz osobnika **i**,*
b) *dodaj osobnika i do populacji P_0.*

Krok 2: *Przystosowanie:*
Ustaw $P_{remain}=P_t$ i zainicjalizuj wartość sztucznego dopasowania F_d liczbą N.
a) *Ustal zbiór rozwiązań niezdominowanych P_{nondom} wybierając ze zbioru P_{remain} osobniki niezdominowane. Następnie usuń je przyszłego procesu klasyfikacji $P_{remain} = P_{remain} - P_{nondom}$,*
b) *przydziel osobnikom z P_{nondom} wartość przystosowania F_d, następnie wykonaj podział przystosowania (sparing fitness).*
c) *Zmniejsz wartość funkcji sztucznego przystosowania tak aby była ona mniejsza od najmniejszej wartości ze zbioru P_{nondom}: $0 < F_d < min\{F(i) \mid i \in P_{momdom}\}$,*
d) *jeżeli $P_{remain} \neq \emptyset$ wówczas idź do punktu a) w przeciwnym wypadku idź do Krok 3.*

Krok 3:
Krok 4:
Krok 5:
Krok 6:
} *Takie same jak w algorytmie HLGA (p. 1.3.2).*

Mimo lepszych wyników w stosunku do algorytmu VEGA przedstawiony wyżej algorytm nie uniknął także pewnych wad. Jedną z nich jest duża złożoność obliczeniowa która dla NSGA wynosi $O(kN^3)$. Do wad również można zaliczyć brak tutaj selekcji elitarnej która za zwyczaj przyspiesza

działanie algorytmu ewolucyjnego, ale także pomaga zapobiegać w pomijaniu dobrych rozwiązań problemu. Powyższe wady jaki i niewątpliwe zalety algorytmu NSGA skłoniły naukowców do opracowania jego szybszej wersji o nazwie NSGA-II , której w danej pracy nie zaprezentowano.

2 Literatura:

[1] Eckart Zitzler: *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications,* Zürich 1999.
[2] Jarosław Arabas: *Wykłady z algorytmów ewolucyjnych,* WNT 2001.
[3] Michalewicz Z.: *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, WNT 1999; s. 202–206.
[3] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler: *A Tutorial on Evolutionary Multiobjective Optimization, Swiss Federal Institute of Technology (ETH) Zurich.*
[4] Jeffrey Horn, Nicholas Nafpliotis, David E. Goldberg: *A Niche Pareto Genetic Algorithm for Multiobjective Optimalization,* IEEE 1994 , Volume 1, pp. 82-87.
[5] N. Srinivas, Kalyanmoy Deb: *Multiobjective Optimization Using Nondominated Sorting In Genetic Algorithm, Department of Mechanical Engineering, Indiana Institute of Technology.*

Źródła internetowe:

[6] http://www.iitk.ac.in/kangal/codes.shtml
[7] http://delta.cs.cinvestav.mx/~ccoello/EMOO/
[8] http://www.mlahanas.de/MOEA/HDRMOGA/main.htm