Wrocław, 11 maja 2021

Zaawansowane **T**echniki **O**ptymalizacji

Laboratorium 6

Wybrane problemy populacyjnego poszukiwania rozwiązań

prowadzący: *dr inż. Jarosław Rudy*

1 Cel laboratorium

Celem laboratorium jest zapoznanie się z specyfiką i problemami rozwiązywania wybranych zadań optymalizacji dyskretnej i ciągłej z wykorzystaniem metaheurystyk populacyjnych. Zagadnienie obejmuje zdefiniowanie reprezentacji rozwiązania w zależności od problemu, wybór implementacji algorytmu oraz kalibrację (dobór parametrów).

2 Przebieg zajęć

Laboratorium obejmuje zajęcia nr 11 i 12 (4 godziny zajęć). Praca odbywa się w ramach grup dwuosobowych. Każda grupa otrzymuje do zrealizowania jeden problem optymalizacji dyskretnej z poniższej listy:

1. Kwadratowe zagadnienie przydziału (problem 2.4).
2. Dyskretny problem plecakowy (problem 2.5).
3. Problem komiwojażera (problem 2.6).
4. Szeregowanie zadań na jednej maszynie z ważoną sumą opóźnień $w_i T_i$ (problem 2.7).
5. Problem przepływowy dla ustalonej liczby maszyn np. $m = 3$ (problem 2.8).

Dla powyższego problemu należy zrealizować **jedną** z poniższych opcji:

1. Implementacja algorytmu genetycznego (GA).
2. Implementacja metody sztucznej kolonii pszczół (ABC).

Dodatkowo na wyższą ocenę grupa może poprosić o dodatkowe zadanie optymalizacji ciągłej, dla której należy wykonać implementację metody optymalizacji rojem cząstek (PSO). Poszczególne problemy opisane są w osobnym pliku PDF na stronie kursu. Metody rozwiązania opisane są w dalszej części instrukcji. Przydziału problemów i metod do grup dokonuje prowadzący podczas zajęć.

Naliczanie oceny zaczynamy od 0. Za poprawną implementację GA lub ABC dla zadanego problemu można otrzymać maksymalnie +4.0 do oceny. Za implementację algorytmu PSO dla dodatkowego (wskazanego przez prowadzącego) zadania optymalizacji ciągłej n zmiennych można otrzymać dodatkowe +1.0 do oceny.

3 Metaheurystyki poszukiwania populacyjnego

Metaheurystyki poszukiwania populacyjnego stanowią dużą grupę metaheurystyk. Wiele z nich czerpie inspiracje ze zjawisk biologicznych lub zachowań społecznych owadów. Do najbardziej znanych należy rodzina metod ewolucyjnych (algorytm ewolucyjny, algorytm genetyczny, programowanie ewolucyjne, algorytm memetyczny), optymalizacja kolonią mrówek, sztuczna kolonia pszczół, sztuczny system immunologiczny, algorytm nietoperza czy optymalizacja z użyciem roju cząstek.

Charakterystycznym elementem odróżniającym te metody od przedstawionych w poprzednim laboratorium jest obecność pewnej liczby obiektów (mrówek, pszczół, osobników, cząstek), które całościowo zwykle nazywane są populacją. Najczęściej z każdym obiektem tego typu skojarzone jest rozwiązanie problemu. Istnieje więc jednocześnie wiele rozwiązań. Najczęściej rozwiązania istnieją przez cały czas (są tylko modyfikowane lub zastępowane innymi), stanowiąc rozbudowaną pamięć algorytmu. W niektórych metodach rozwiązania nie są zapamiętywane bezpośrednio lecz pod inną postacią (w kolonii mrówek jest to macierz feromonu).

Meteheurystyki populacyjne eksplorują często wiele ścieżek (trajektorii) poszukiwań równocześnie. Metody te są też bardziej podatne na skrócenie czasu działania z wykorzystaniem technik obliczeń równoległych. Ponadto wiele z nich w dużej mierze bazuje na znanym z poprzednich zajęć pojęciu sąsiedztwa, choć w części z nich stosuje się też inne mechanizmy. Jednakże, jak w każdej metaheurystyce, mechanizmy te odpowiadają za intensyfikację i dywersyfikację poszukiwań – kluczowy jest odpowiedni balans pomiędzy tymi procesami. W przypadku zbyt dużej intensyfikacji poszczególne osobniki w populacji staną się do siebie podobne, a algorytm straci najważniejszą siłę którą daje populacja: różnorodność. Istotny jest również odpowiedni dobór parametrów algorytmu.

3.1 Sztuczna kolonia pszczół

Sztuczna kolonia pszczół (Artificial Bee Colony, ABC) jest metodą bazującą na zachowaniu społecznych pszczół miodnych. Robotnice-zbieraczki po wyszukaniu źródła nektaru przekazują informacje o nim (kąt/azymut, odległość od ula, jakość/wielkość) innym pszczołom z wykorzystaniem serii akrobacji (tańca). Inne pszczoły skupiają się na „zareklamowanych" miejscach. Znajdywane są również nowe źródła nektaru. Całość pozwala na zmaksymalizowanie zbiorów nektaru.

W podstawowej formie ABC jest względnie prostą metaheurystyką, bazującą w znacznej mierze na pojęciu sąsiedztwa (przynajmniej dla problemów optymalizacji dyskretnej, implementacja dla problemów ciągłych jest trochę inna). Pszczoły formalnie dzieli się na 3 grupy: zbieraczy, obserwatorów i zwiadowców, choć podział ten na poziomie kodu programu nie jest widoczny. W każdej iteracji istnieje pewna liczba k rozwiązań (gdzie k to rozmiar populacji). Początkowo są to rozwiązania losowe. Każda z k zbieraczy generuje sąsiada swojego rozwiązania (czyli i-ty zbieracz zajmuje się i-tym rozwiązaniem). Jeśli sąsiad jest lepszy, to zastępuje stare rozwiązanie. Następnie k obserwatorów powtarza ten proces. Różnica polega na tym, że każdy obserwator losuje rozwiązanie, którym się będzie zajmował. Może się więc okazać, że częścią rozwiązań nie zajmie się żaden obserwator, a niektórymi zajmie się wielu. Prawdopodobieństwo wyboru zależy od jakości rozwiązania. Za każdym razem jeśli sąsiad (stworzony przez zbieracza lub obserwatora) nie poprawi rozwiązania, zwiększa się jego licznik. Po przekroczeniu limitu zmian bez poprawy wysyłana jest pszczoła zwiadowca w celu zastąpienia rozwiązania innym, zwykle losowym.

Poniżej przedstawiono ogólny schemat dla problemu *maksymalizacji*:

1. Wylosuj rozwiązanie x_1 i wykonaj $x_{\text{best}} \leftarrow x_1$ oraz $c_1 = 0$.
2. Dla i od 2 do k:
   2.1. Wylosuj rozwiązanie x_i i wykonaj $c_i = 0$.
   2.2. Jeśli $f(x_i) > f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow x_i$.

3. Dopóki warunek końca nie jest spełniony:

   3.1. Dla i od 1 do k:

      3.1.1. Wygeneruj x'_i jako losowego sąsiada x_i.

      3.1.2. Jeśli $f(x'_i) > f(x_i)$ to:

         3.1.2.1. Wykonaj $c_i \leftarrow 0$.

         3.1.2.2. Wykonaj $x_i \leftarrow x'_i$.

         3.1.2.3. Jeśli $f(x_i) > f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow x_i$.

      3.1.3. W przeciwnym razie wykonaj $c_i \leftarrow c_i + 1$.

   3.2. Oblicz prawdopodobieństwa koła ruletki.

   3.3. Dla i od 1 do k:

      3.3.1. Wylosuj j na podstawie koła ruletki.

      3.3.2. Wygeneruj x'_j jako losowego sąsiada x_j.

      3.3.3. Jeśli $f(x'_j) > f(x_j)$ to:

         3.3.3.1. Wykonaj $c_j \leftarrow 0$.

         3.3.3.2. Wykonaj $x_j \leftarrow x'_j$.

         3.3.3.3. Jeśli $f(x_j) > f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow x_j$.

      3.3.4. W przeciwnym razie wykonaj $c_j \leftarrow c_j + 1$.

   3.4. Dla i od 1 do k:

      3.4.1. Jeśli $c_i > L$ to:

         3.4.1.1. Wylosuj rozwiązanie x_i.

         3.4.1.2. Wykonaj $c_i \leftarrow 0$.

         3.4.1.3. Jeśli $f(x_i) > f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow x_i$.

4. Wypisz rozwiązanie x_{best}.

W powyższym algorytmie L jest limitem zmian bez poprawy dla rozwiązania, k jest rozmiarem populacji, zaś $f(x)$ standardowo jest wartością funkcji celu rozwiązania x.

Prawdopodobieństwo koła ruletki z punktu 3.2. polega na przypisaniu każdemu rozwiązaniu wycinku koła. Prawdopodobieństwo P_i wynosi:

$$P_i = \frac{f(x_i)}{\sum_{j=1}^{k} f(x_j)}. \tag{1}$$

Przykładowo, jeśli mamy osobników x_1, x_2 oraz x_3 tak że $f(x_1) = 7$, $f(x_2) = 10$ oraz $f(x_3) = 8$, to $P_1 = \frac{7}{25} = 0.28$, $P_2 = \frac{10}{25} = 0.4$ oraz $P_3 = \frac{8}{25} = 0.32$[1]. Wylosowanie rozwiązania (punkt 3.3.1) najłatwiej zrealizować losując jedną liczbę z zakresu od 1 do $\sum_{j=1}^{k} f(x_j)$. W naszym przykładzie jeśli będzie mniejsza lub równa 7 to wylosowano rozwiązanie 1, jeśli będzie od 8 do 17 to wylosowano rozwiązanie 2, zaś jeśli będzie od 18 do 25, to wylosowano rozwiązanie 3. Ten prosty system działa jednak tylko jeśli wartości $f(x)$ są całkowitoliczbowe. Uwaga! Powyższa implementacja koła ruletki dotyczy problemów maksymalizacji! Dla problemów minimalizacji wzór (1) przyjmuje inną formę.

Uwaga! Otrzymanie maksymalnej oceny za implementację metody ABC wymaga więcej niż pokazano powyżej. Można rozważyć (porównać) różne rodzaje sąsiedztw. Można też zastąpić punkt 3.1.1 w ten sposób, że x'_i powstaje nie jako sąsiad x_i, ale jako potomek powstały z krzyżowania (jak w algorytmie genetycznym poniżej) x_i oraz x_j, gdzie x_j jest wybrany losowo (ale tak by $i \neq j$).

[1] Zauważmy że $P_1 + P_2 + P_3 = 1$.

3.2 Algorytm genetyczny

Algorytm genetyczny (Genetic Algorithm, GA) jest przedstawicielem szerszego kręgu metod ewolucyjnych. Metody te są metodami probabilistycznymi (losowymi) i działają poprzez naśladowanie procesu ewolucji biologicznej. Słynna teoria Darwina zakłada bowiem że:

Najlepiej przystosowane osobniki mają największą szansę na przekazanie genów

Teoria ta zakłada, każdy osobnik ma zestaw genów (genotyp), które określają jego cechy zewnętrzne (fenotyp), które następnie wpływają na jego funkcję przystosowania. Osobniki dobierają się w pary, przy czym im większa wartość funkcji przystosowania tym większa szansa wyboru danego osobnika. Wybrane osobniki krzyżują (rozmnażają) się, tworząc osobniki potomne. Osobniki potomne mogą być poddane mutacji, która powoduje niewielkie zmiany genów. Powyższy proces prowadzi do ustalenia nowej populacji osobników, co kończy daną generację (pokolenie, iterację). Proces ten jest następnie powtarzany cyklicznie.

Metoda GA wykorzystuje podobny (uproszczony) mechanizm, zastępując pojęcia biologiczne pojęciami z domeny danego problemu. Odpowiednikiem genotypu i fenotypu (ten drugi nie zawsze jest wyróżniany) są reprezentacje rozwiązania. Przykładowo w wielu problemach szeregowania zadań genotypem byłaby kolejność wykonywania zadań, a fenotypem konkretny harmonogram. Z kolei w przykładzie z testem jednokrotnego wyboru n pytań z 4 wariantami odpowiedzi genotypem (i jednocześnie fenotypem) byłby wektor binarny $4n$ elementów. Alternatywnie może być to wektor n elementów, gdzie każdy jest wybranym wariantem danego pytania[2]. Funkcją przystosowania jest po prostu funkcja celu (przynajmniej dla problemów maksymalizacji).

Następnie należy zaimplementować mechanizmy takie jak operatory genetyczne i selekcję, a następnie odpowiednio je ze sobą powiązać. Istnieje wiele szczegółowych podejść. Poniżej opiszemy jedno z bardziej standardowych. Zacznijmy od operatora krzyżowania. Jest on jednym z najważniejszych elementów. Najczęściej operator przyjmuje na wejściu dwóch rodziców, tworząc dwójkę potomków (dzieci) na wyjściu, choć spotykane są inne konfiguracje (np. jeden potomek lub troje rodziców). W dalszej części instrukcji zakładamy dwoje rodziców i dwoje potomków. W przykładowym problemie testów jednokrotnego wyboru załóżmy, że mamy skróconą reprezentację dla $n = 12$ pytań. Wtedy rodzice r_1 i r_2 mogą wyglądać następująco:

$$r_1 = \texttt{3 2 3 1|2 4 3 1 3 2|4 2}$$
$$r_2 = \texttt{1 4 2 3|2 3 4 1 2 4|1 2}$$

Pionowe linie reprezentują tzw. punkty przecięć. Ich liczba jest zależna od operatora krzyżowania, ale najczęściej są dwa. Przy każdym krzyżowaniu punkty przecięć są wybierane losowo, symulując wymianę materiału genetycznego między osobnikami. Nasz przykładowy operator może stworzyć potomka c_1 w taki sposób, że fragment pomiędzy punktami przecięć pochodzi z r_1 a reszta z r_2. Potomka c_2 tworzymy analogicznie, zamieniając r_1 oraz r_2 miejscami:

$$c_1 = \texttt{1 4 2 3|2 4 3 1 3 2|1 2}$$
$$c_2 = \texttt{3 2 3 1|2 3 4 1 2 4|4 2}$$

Jak łatwo zauważyć, dla tego problemu operator krzyżowania jest trywialny, jednakże dla wielu problemów operatory muszą zapewniać, by powstałe rozwiązania były dopuszczalne (zakładając dopuszczalność rodziców). Często istnieje wiele operatorów krzyżowania dla danego problemu.

Aby jednak w ogóle doszło do krzyżowania potrzebna jest selekcja rodzicielska, która powoduje wybranie par rodziców. W najprostszej postaci dla populacji rozmiaru p tworzy się $\frac{p}{2}$ par rodziców, tworząc tym samym p potomków. Zakłada to, że każda para dokonuje krzyżowania. Możliwe jest

[2]Puryści nazwali by taką implementację Algorytmem Ewolucyjnym (EA), rezerwując nazwę GA tylko dla tych metod, gdzie genotyp jest zapisany w postaci binarnej.

jednak określenie prawdopodobieństwa krzyżowania (wynoszącego zwykle od 70% do 100%). W takim jednak przypadku liczba powstałych potomków będzie zmienna, co trzeba wziąć pod uwagę (najczęściej dobiera się rodziców para po parze aż do uzyskania pożądanej liczby potomków).

Pozostaje jeszcze kwestia doboru rodziców. Typowym (i często koniecznym) założeniem jest to, że dany rodzic może być w więcej niż jednej parze rodzicielskiej. Prawdopodobieństwo wyboru rodzica powinno być zależne od jego funkcji przystosowania. Zwykle chcemy bowiem krzyżować dobrego rodzica z dobrym, ewentualnie dobrego ze słabym (w nadziei że słaby posiada te nieliczne geny, których dobremu brakuje). Krzyżowanie słaby-słaby jest rzadkością (choć nie eliminuje się tego zjawiska całkowicie). W praktyce najczęściej spotyka się dwa podejścia:

1. Metoda ruletki – analogicznie do tej opisanej dla metody ABC powyżej. Należy pamiętać, że podstawowa wersja przeznaczona jest dla problemu maksymalizacji.

2. Metoda turniejowa – w celu wybrania rodzica przeprowadza się turniej. Polega to na wylosowaniu (z równym prawdopodobieństwem) t różnych osobników, gdzie t jest rozmiarem turnieju. Zwycięzcą turnieju jest najlepszy z wylosowanych osobników. Czasami z turnieju wybiera się więcej niż jednego osobnika (np. od razu parę, czyli dwóch najlepszych). Rozmiar turnieju trzeba odpowiednio dobrać. Zbyt mały spowoduje, że wybór rodzica będzie niezależny od jego jakości, zaś zbyt duży sprawi, że słabsze osobniki w ogóle nie będą miały szansy na rozmnażanie.

Przejdźmy teraz do mutacji potomków. Mutacja jest mechanizmem dywersyfikującym, wprowadzającym różnorodność genetyczną. Z tego powodu mechanizm ten powinien mieć dużo mniejsze znaczenie niż krzyżowanie. Dlatego w najprostszym podejściu ustala się prawdopodobieństwo mutacji (zwykle pomiędzy 0.5% a 5%). Mutacja najczęściej jest rozumiana jako wykonanie jednego ruchu tj. zastąpienie potomka przez jego losowego sąsiada. Efekt jest binarny: albo mutujemy osobnika nieznacznie, albo nie mutujemy go w ogóle. Podejście bliższe mutacji biologicznej zakłada, że każdy gen ma osobną szansę na mutację. W przykładzie testów jednokrotnego wyboru dla n pytań polega to na teście prawdopodobieństwa dla każdego z n genów osobno, jednakże prawdopodobieństwo mutacji musi być n razy mniejsze. W średnim przypadku zmieni się jeden gen (jak w zwykłej mutacji), ale możliwa jest też zmiana wszystkich genów.

Należy też zwrócić uwagę, że operator krzyżowania tworzy nowe osobniki. Następnie niektóre osobniki są mutowane. Zauważmy że obie czynności mogą stworzyć nowego osobnika lub go zmienić, co oznacza że mogą też doprowadzić do uzyskania najlepszego do tej pory znanego osobnika. Dlatego też formalnie dla potomka, który uległ mutacji funkcję celu należy obliczyć dwukrotnie: po krzyżowania i po mutacji, by upewnić się, że nie przeoczyliśmy poprawy.

Bardzo ważnym krokiem jest selekcja nowej populacji. W ogólności chcemy wybrać nową populację p osobników na podstawie (1) starej (aktualnej) populacji p osobników, (2) potomków (zwykle w liczbie p, ale niekoniecznie) oraz ewentualnie (3) populacji rodzicielskiej (w której są kopie czy raczej wskaźniki/referencje na osobniki z populacji aktualnej). Najprostszym (i w większości zgodnym z procesem biologicznym) sposobem jest zastąpienie całej starej populacji potomkami. Przy małym rozmiarze populacji może to jednak powodować utratę obiecujących genów. Innym sposobem jest połączenie populacji starej z potomkami (najczęściej oznacza to zbiór $2p$ osobników), po czym wybranie z nich p osobników. Wybór odbywa się jak przy selekcji rodziców: z pomocą ruletki lub turnieju. Często spotyka się również element elitaryzmu: część (np. 5–10%) nowej populacji wypełnia się najlepszymi dostępnymi osobnikami. Nie należy jednak przesadzać. Przykładowo zawsze wybieranie p najlepszych osobników spowoduje przedwczesną zbieżność algorytmu.

Ostatnim elementem jest populacja początkowa. Najczęściej tworzą ją osobnicy losowi. Aby podnieść jakość populacji początkowej można dodać część osobników dostarczonych przez inne algorytmy (np. zachłanne), ewentualnie częściowo pogorszonych (przez wykonanie pewnej liczby mutacji). Inną strategią, jeśli potrzebujemy 100 osobników, może być najpierw wylosowanie 2000 osobników i wybranie spośród nich 20 najlepszych. Pozostałych 80 wybieramy losowo.

Poniżej znajduje się (bardzo ogólny) schemat metody GA dla problemu **minimalizacji**:

1. Ustal osobnika x_1, oblicz dla niego $f(x_1)$, wykonaj $x_{\text{best}} \leftarrow x_1$ oraz dodaj x_1 do X.
2. Dla j od 2 do p:
   2.1. Ustal osobnika x_j oraz oblicz dla niego $f(x_j)$.
   2.2. Jeśli $f(x_j) < f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow x_j$.
   2.3. Dodaj x_j do X.
3. Dopóki nie jest spełniony warunek stopu:
   3.1. Na podstawie X ustal zbiór R zawierający pary rodziców do krzyżowania (selekcja rodzicielska).
   3.2. Dla każdej pary rodzicielskiej $(r_1, r_2) \in R$:
      3.2.1. Wykonaj operator krzyżowania dla (r_1, r_2) i dodaj uzyskanych potomków do C.
   3.3. Dla każdego potomka $c \in C$:
      3.3.1. Oblicz $f(c)$.
      3.3.2. Jeśli $f(c) < f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow c$.
      3.3.3. Wykonaj operator mutacji dla c (z odpowiednim prawdopodobieństwem).
      3.3.4. Jeśli mutacja wystąpiła (spowodowała zmiany) w c to:
         3.3.4.1. Oblicz $f(c)$.
         3.3.4.2. Jeśli $f(c) < f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow c$.
   3.4. Na postawie zbiorów X oraz C (ewentualnie C oraz R) ustaw nową zawartość zbioru X (selekcja nowej populacji).
   3.5. Wyczyść zbiory R oraz C.
4. Wypisz rozwiązanie x_{best}.

3.3 Optymalizacja rojem cząstek

Metoda optymalizacji rojem cząstek (Particle Swarm Optimization, PSO) bazowana jest częściowo na zachowaniach stadnym zwierząt (ławice ryb, stada antylop, klucze ptaków). Każdy element populacji (osobnik) dokonuje własnej (lokalnej) optymalizacji na podstawie swojej pozycji (rozwiązania), ale bierze też pod uwagę najlepszą pozycję w całym stadzie (lidera).

Rozwiązania w metodzie PSO są traktowane jako punkty (wektory, pozycje) w przestrzeni. Działanie algorytmu polega na iteracyjnym przesuwaniu tych punktów za pomocą wektora prędkości. Z kolei wektor prędkości obliczany jest w oparciu o (1) różnicę między aktualną pozycją i najlepszą zapamiętaną przez cząstkę, (2) różnicę pomiędzy aktualną pozycją i najlepszą zapamiętaną przez cały rój, (3) pewną wartość losową, (4) parametry algorytmu. Ze względu na to matematyczne podejście i operacje arytmetyczne (mnożenie i dodawanie wektorów) metodę PSO najłatwiej zastosować dla problemów ciągłych.

Załóżmy, że rozwiązania są w postaci wektorów $x = (x_1, x_2, \ldots, x_n)$. Odpowiada to problemowi optymalizacji funkcji n zmiennych. Niech $L = (L_1, L_2, \ldots, L_n)$ oraz $U = (U_1, U_2, \ldots, U_n)$ będą wektorami określającymi zakres przestrzeni (np. x_2 zmienia się w zakresie od L_2 do U_2). Ponadto, ω, φ_l, φ_g oraz c (ten ostatni, zwany tempem uczenia, jest w przedziale od 0 do 1) są parametrami algorytmu, które należy dobrać. Podobnie należy dobrać k będące liczbą cząstek. Cząstki będziemy numerować z użyciem indeksów górnych. Pozycja cząstki j oznaczona jest przez $x^j = (x_1^j, x_2^j, \ldots, x_n^j)$, zaś jej prędkość przez $v^j = (v_1^j, v_2^j, \ldots, v_n^j)$. Przez l^j będziemy oznaczać najlepsze znane rozwiązanie przez cząstkę j (lokalne). Z kolei przez g będziemy oznaczać najlepsze rozwiązanie znane przez cały rój (globalne). Wreszcie przez $\mathcal{U}$ oznaczamy jednostajny rozkład prawdopodobieństwa. Przykładowo

$\mathcal{U}(L_i, U_i)$ oznacza wylosowanie liczby z przedziału $[L_i, U_i]$ zaś $\mathcal{U}(L, U)$ oznacza wylosowanie całego n-elementowego wektora z zadanego wycinka przestrzeni n-wymiarowej.

Przy tej notacji szablon metody PSO dla problemu *minimalizacji* wygląda następująco:

1. Wykonaj $x^1 \leftarrow \mathcal{U}(L, U)$, $l^1 \leftarrow x^1$ oraz $g \leftarrow x^1$.
2. Wykonaj $v^1 \leftarrow \mathcal{U}(L - U, U - L)$.
3. Dla j od 2 do k:
   - 3.1. Wykonaj $x^j \leftarrow \mathcal{U}(L, U)$ oraz $l^j \leftarrow x^j$.
   - 3.2. Jeśli $f(x^j) < f(g)$ to wykonaj $g \leftarrow x^j$.
   - 3.3. Wykonaj $v^j \leftarrow \mathcal{U}(L - U, U - L)$.
4. Dopóki nie jest spełniony warunek końca:
   - 4.1. Dla j od 1 do k:
     - 4.1.1. Dla i od 1 do n:
       - 4.1.1.1. Wykonaj $r_l \leftarrow \mathcal{U}(0, 1)$ oraz $r_g \leftarrow \mathcal{U}(0, 1)$.
       - 4.1.1.2. Wykonaj $v_i^j \leftarrow \omega\, v_i^j + \varphi_l\, r_l\, (l_i^j - x_i^j) + \varphi_g\, r_g\, (g_i - x_i^j)$.
     - 4.1.2. Wykonaj $x_i^j \leftarrow x_i^j + c\, v_i^j$.
     - 4.1.3. Jeśli $f(x^j) < f(l^j)$:
       - 4.1.3.1. Wykonaj $l^j \leftarrow x^j$.
       - 4.1.3.2. Jeśli $f(l^j) < f(g)$ to wykonaj $g \leftarrow l^j$.
5. Wypisz rozwiązanie g.