Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki
Instytut Informatyki

Daniel Kostrzewa

PRZESZUKIWANIE PRZESTRZENI ROZWIĄZAŃ W OPTYMALIZACJI PLANÓW ZAPYTAŃ DO BAZ DANYCH Z WYKORZYSTANIEM HEURYSTYCZNEGO ALGORYTMU IWO

Rozprawa doktorska napisana
pod kierunkiem
Prof. dr hab. inż. Stanisława Kozielskiego

Gliwice, 2015

Pragnę podziękować mojemu Promotorowi
prof. dr hab. inż. Stanisławowi Kozielskiemu
za opiekę naukową, życzliwość, czas poświęcony
na dyskusje oraz cenne wskazówki.

Składam serdeczne podziękowania mojemu
Przyjacielowi dr inż. Henrykowi Josińskiemu
za pomoc naukową, cierpliwość oraz wiele cennych uwag,
bez których ta praca nie mogłaby powstać.

Dziękuję Rodzicom za pomoc, wsparcie i zachętę.

Szczególne podziękowania składam mojej Żonie Marcie
za cierpliwość, wyrozumiałość i wsparcie w chwilach zwątpienia.

Spis treści

1. Wprowadzenie

Intensywny rozwój systemów informatycznych wprowadził znaczące zmiany w zakresie gromadzenia i przetwarzania danych. Systemy oparte na bazach danych obecne są w niemal każdej gałęzi przemysłu i usług. Z tego względu kluczowym zagadnieniem stało się efektywne wyszukiwanie danych. Ma ono zasadnicze znaczenie dla szybkości dostępu do informacji w każdym systemie informatycznym.

Ta sytuacja spowodowała, że głównym przedmiotem rozważań niniejszej rozprawy stało się skuteczne wyszukiwanie danych.

Problem wydajnej realizacji zapytań bazodanowych jest niezwykle trudnym zagadnieniem badawczym. Złożoność tego zadania powoduje, że prowadzone prace koncentrują się wokół wielu problemów składowych, do których między innymi należy *optymalizacja zapytania*.

Optymalizacja zapytania sprowadza się do określenia takiej kolejności wykonania poszczególnych operacji składowych, aby czas oczekiwania użytkownika na dane będące wynikiem końcowym był jak najkrótszy [45, 38, 28, 39]. Do operacji składowych zalicza się [45]:

- skanowanie (relacji lub indeksu),
- selekcję,
- projekcję,
- złączenie,
- grupowanie,
- agregację.

Wśród wymienionych operacji niezwykle ważną rolę pełnią *złączenia*, które służą do łączenia zbiorów danych pochodzących z różnych relacji. W większości zapytań bazodanowych angażujących wiele relacji proces łączenia danych jest najbardziej złożonym i czasochłonnym elementem realizacji zapytania. W związku z tym minimalizacja czasu trwania tego procesu ma kluczowe znaczenie dla całego zadania optymalizacji zapytania kierowanego do bazy danych.

Opisana problematyka od wielu lat jest przedmiotem intensywnych prac badawczych [19, 61, 124, 64], ale tematyka tych badań jest ciągle aktualna [33, 142, 147].

Głównymi tematami prowadzonych badań są:

- dobór modelu kosztów [117, 124, 64] oraz
- wyznaczenie kolejności realizacji operacji złączenia [92, 62, 19].

Metody rozwiązujące zadanie określenia kolejności złączeń opierają się m.in. na: idei programowania dynamicznego [105], przeszukiwania lokalnego [133] bądź wykorzystują różne metaheurystyki kombinatoryczne [108, 129, 99, 34]. Mimo, że powstało bardzo wiele technik rozwiązywania wspomnianego problemu, to nadal poszukuje się nowych, skutecznych strategii.

W 2006 roku po raz pierwszy zaprezentowano algorytm heurystyczny o nazwie *Invasive Weed Optimization* (IWO). Metoda ta często jest wykorzystywana do rozwiązywania różnych problemów optymalizacyjnych [101, 98, 125, 122, 48, 49, 112, 146]. Przekonujące rezultaty otrzymywane przy użyciu wspomnianej techniki stanowiły dla autora rozprawy sugestię, aby zmodyfikować algorytm IWO, a następnie zbadać możliwość zastosowania go do problemu wyznaczenia optymalnej kolejności realizacji operacji złączenia.

W związku z tym **celem rozprawy stało się opracowanie zmodyfikowanej metody *Invasive Weed Optimization* dla potrzeb zadań optymalizacji, w szczególności wyznaczania kolejności złączeń w zadaniach optymalizacji zapytań do baz danych**.

Osiągnięcie tak postawionego celu wymagało zarówno dokładnego zapoznania się ze znanymi rozwiązaniami, jak i przeprowadzenia szczegółowej analizy algorytmu IWO. Czynniki te dały podstawę do opracowania metody autorskiej. Tezę rozprawy sformułowano w następującej postaci:

Zaproponowany w pracy zmodyfikowany algorytm *Invasive Weed Optimization* może zostać skutecznie wykorzystany w zadaniach optymalizacji, a w szczególności do wyznaczania kolejności złączeń w procesie realizacji zapytań w bazach danych.

Niniejsza rozprawa składa się z siedmiu rozdziałów, spisów: literatury, rysunków i tabel oraz dwóch dodatków.

Rozdział drugi stanowi wprowadzenie teoretyczne w tematykę optymalizacji zapytania kierowanego do scentralizowanej bazy danych. Przedstawia miejsce optymalizacji w procesie wykonywania zapytania oraz opisuje jej etapy.

Dokładnemu opisowi problemu określania kolejności złączeń w realizacji zapytań poświęcono rozdział trzeci. W tej części rozprawy zaprezentowano także dwa klasyczne problemy optymalizacyjne: znajdowanie minimum funkcji wielowymiarowej i komiwojażera.

Przegląd strategii optymalizacyjnych wraz ze szczegółowym opisem oryginalnej wersji algorytmu *Invasive Weed Optimization* zawarto w rozdziale czwartym.

W rozdziale piątym zaprezentowano autorską wersję algorytmu IWO. W dalszej części rozdziału przedstawiono modyfikacje wprowadzone na potrzeby przystosowania algorytmu do wybranych problemów optymalizacyjnych oraz podjęto próbę wyznaczenia złożoności obliczeniowej opracowanej metody.

Wyniki badań eksperymentalnych przeprowadzonych dla wybranych problemów optymalizacyjnych zgromadzono w rozdziale szóstym. Zawarto w nim także analizę otrzymanych rezultatów.

Rozdział siódmy zawiera wnioski końcowe podsumowujące pracę oraz kierunki dalszych badań i możliwości rozwoju algorytmu IWO.

W dodatkach do pracy zamieszczono szczegółowe wyniki wykonanych eksperymentów oraz opisano programy komputerowe stworzone w celu przeprowadzenia badań.

2. Podstawowe pojęcia związane z optymalizacją zapytania

Bazy danych są w centrum uwagi niniejszej pracy. Rozważania w niej zawarte dotyczą modelu relacyjnego, którego twórcą jest Edgar Frank Codd [22, 23]. Rozdział ten jest poświęcony wyjaśnieniu podstawowych zagadnień związanych z optymalizacją zapytań kierowanych do systemu zarządzania bazą danych (SZBD).

Zapytania poddawane optymalizacji formułowane są przez użytkownika w języku SQL (ang. *Structured Query Language*) [17, 13]. Język ten należy do grupy języków czwartej generacji, języków *deklaratywnych* (niestrukturalnych) [14, 96]. Termin ten oznacza klasę języków, w których określa się cel, jaki użytkownik zamierza osiągnąć, a nie kroki do niego prowadzące [132]. Wykonanie zapytania SQL wymaga od systemu zarządzania bazą danych określenia zestawu operacji oraz kolejności ich realizacji. Pozwala to na wydobycie z bazy danych oczekiwanych informacji. Większość zapytań może być przetworzonych na wiele sposobów. Kluczowym zagadnieniem optymalizacyjnym jest wyznaczenie jak najbardziej efektywnej strategii (*planu*) wykonania zapytania [140]. Rozmiar przestrzeni poszukiwań, w której poszukuje się planu optymalnego, zależy od czynników, takich jak: liczba wyświetlanych atrybutów, liczba tabel biorących udział w zapytaniu czy liczba i rodzaj dodatkowych warunków nakładanych na atrybuty. Jako plan „optymalny” w tym przypadku rozumie się najlepsze z aktualnie osiągalnych, nie zaś najlepsze ze wszystkich możliwych rozwiązań.

Proces optymalizacji może się okazać czynnością czasochłonną. Z praktycznego punktu widzenia nie do zaakceptowania jest sytuacja, w której czas poszukiwania rozwiązania optymalnego znacznie przekracza czas realizacji zapytania według wyznaczonego planu. Istotniejsze jest zatem odnalezienie planu o zadowalającej jakości (i uniknięcie nieefektywnego, długotrwałego planu realizacji) niż dążenie do rozwiązania optymalnego globalnie. Programy optymalizujące zapytania zawężają przestrzeń poszukiwań, często korzystając z metod heurystycznych. Cechą wyróżniającą podejścia heurystyczne jest wykorzystanie zarówno wiedzy dziedzinowej, jak i intuicyjnych sposobów otrzymywania możliwie najlepszych rozwiązań. Heurystyki nie zawsze prowadzą do rozwiązań optymalnych globalnie, lecz ich skuteczność połączona z szybkością działania decyduje o ich praktycznym znaczeniu.

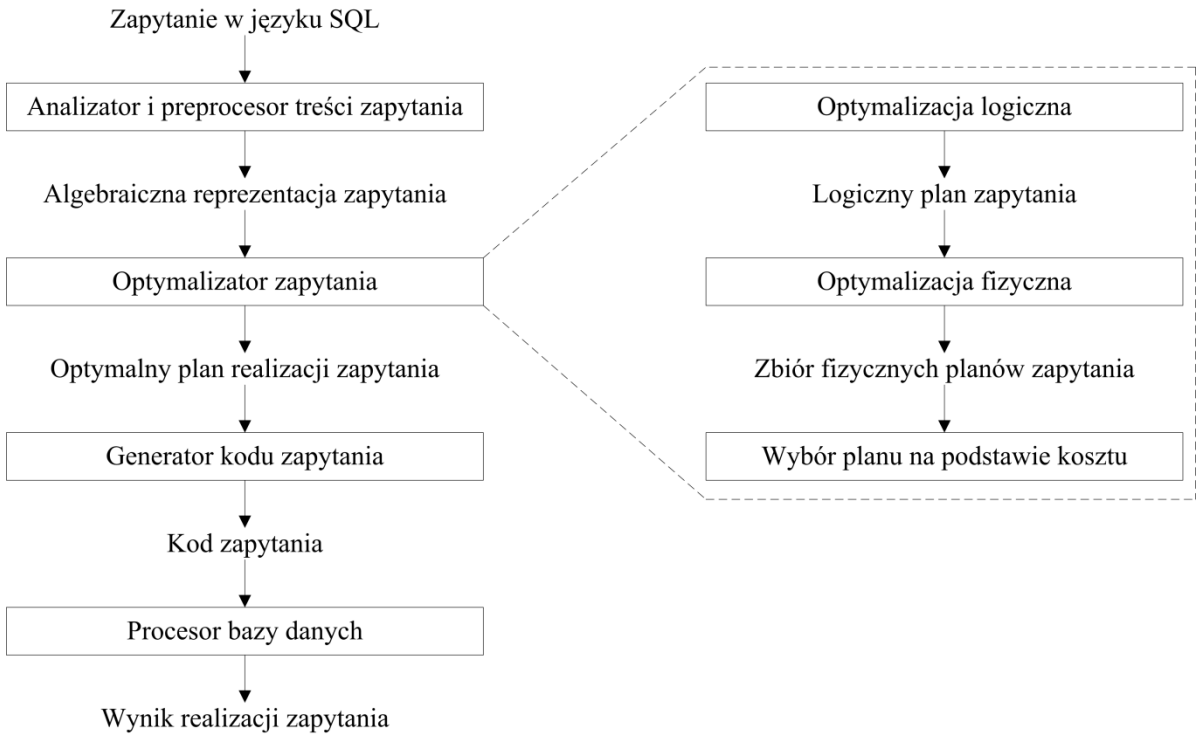
W pracy [43] wyszczególniono następujące klasy planu realizacji zapytania:

- planem *dobrym* nazywa się rozwiązanie, które nie przekracza dwukrotnej wartości kosztu planu optymalnego globalnie;
- koszt planu *akceptowalnego* jest ponad dwukrotnie wyższy od planu optymalnego, a zarazem nie przekracza jego dziesięciokrotnej wartości;
- plany *nie do zaakceptowania* przekraczają dziesięciokrotną wartość kosztu planu optymalnego.

Nadrzędnym celem optymalizacji planu wykonania zapytania jest uniknięcie rozwiązania nieefektywnego. Warunkiem wystarczającym zakończenie poszukiwań planu realizacji jest więc otrzymanie planu o „dobrej" jakości.

2.1. Optymalizacja jako jeden z etapów procesu przetwarzania treści zapytania

Etapy przetwarzania zapytania, którego jednym z kluczowych elementów jest optymalizacja, przedstawiono na rys. 1 (na podstawie [45]).



Rys. 1. Etapy przetwarzania zapytania w systemie zarządzania bazą danych.

Zadaniem analizatora jest przekształcenie zapytania sformułowanego w języku SQL w tzw. *drzewo składniowe* [44]. Realizacja transformacji polega na: wyodrębnieniu słów kluczowych języka SQL, nazw relacji i atrybutów; sprawdzeniu poprawności syntaktycznej analizowanego zapytania oraz zbudowaniu drzewa. Słowa kluczowe wraz z nazwami relacji i atrybutów stają się liśćmi drzewa składniowego, natomiast węzły wewnętrzne stanowią tzw. *kategorie składniowe*, czyli nazwy typów elementów zapytania. Przykładową kategorią składniową może być *warunek* będący wyrażeniem występującym we frazie WHERE zapytania i określający wymagania nakładane na zestaw wierszy wynikowych [45].

Preprocesor spełnia kilka różnych ról. Jeśli relacja biorąca udział w zapytaniu jest perspektywą, to każde jej wystąpienie jest zastępowane stosownym drzewem składniowym (drzewo to otrzymywane jest z definicji perspektywy). Preprocesor zajmuje się również analizą semantyczną zapytania. Do najważniejszych zadań należą m.in.: sprawdzenie użycia

relacji (czy używane relacje są dostępne w schemacie, którego dotyczy zapytanie), weryfikacja atrybutów (każdy atrybut musi być elementem relacji określonej we frazie FROM zapytania SQL oraz musi być jednoznaczny) oraz sprawdzenie typów (sposób użycia atrybutu musi być zgodny z jego typem, aby istniała możliwość zastosowania niektórych operatorów, np. LIKE) [44].

Po utworzeniu i analizie semantycznej drzewo zapytania należy przekształcić w dogodny logiczny plan zapytania. Zadanie to wykonuje się w dwóch fazach. W pierwszej fazie drzewo składniowe zamieniane jest za pomocą pewnych reguł na reprezentację algebry relacyjnej. Natomiast druga faza, zwana *optymalizacją logiczną* (*algebraiczną*), polega na przekształceniu otrzymanego w pierwszej fazie wyrażenia na wyrażenie mu równoważne. W tym celu optymalizator posługuje się regułami heurystycznymi wynikającymi z *praw algebry relacji* [31, 144]. Uzyskany w ten sposób *logiczny plan zapytania* [28] staje się podstawą do wygenerowania efektywnego *fizycznego planu zapytania*, otrzymywanego na etapie *optymalizacji fizycznej* [44].

Podczas optymalizacji fizycznej każdemu działaniu z planu logicznego przypisuje się realizującą go metodę (algorytm). Określa się kolejność wykonywania oraz grupuje się poszczególne działania (tj. złączenie, suma, przecięcie itp.). Na etapie tym bierze się pod uwagę wiele czynników, np. organizację fizyczną tabel, istnienie indeksów czy konieczność sortowania wierszy. W trakcie wykonywania optymalizacji fizycznej zwykle rozważa się wiele równoważnych fizycznych planów zapytania. Wybór planu fizycznego, który zostanie skierowany do realizacji, następuje na podstawie oszacowanego kosztu jego wykonania, metody heurystycznej, lub, najczęściej, techniki stanowiącej kombinację obu podejść [38].

Wyrażenie zawierające wszystkie dane mające wpływ na koszt realizacji zapytania nazywa się funkcją kosztu (funkcją celu, ang. *cost function).* Jej miarą są zwykle: liczba dostępów do dysku, czas procesora i czas komunikacji (w przypadku środowiska wieloprocesorowego lub rozproszonego). Wartości funkcji są wyrażone w jednostkach czasu lub są bezwymiarowe.

W sytuacji, gdy wybór planu realizacji zapytania jest dokonywany na podstawie kosztu, przydatne są pewne dodatkowe informacje – metadane bazy danych. Obejmują one m.in.: rozmiar poszczególnych relacji, przybliżone liczby oraz częstości wystąpień wartości atrybutów relacji, organizację struktur tabel i sposoby dostępu do danych [45]. Istotnym zagadnieniem jest także oszacowanie liczebności relacji pośrednich stanowiących wyniki cząstkowe, które powstałyby w trakcje wykonania analizowanego planu realizacji zapytania.

Dokładne oszacowanie kosztu wykonania planu jest zadaniem bardzo trudnym pomimo dysponowania wspomnianymi informacjami. Należy pamiętać, że wykonaniu podlega tylko jeden wybrany plan, zatem szacowanie kosztów różnych planów odbywa się bez ich realizacji.

Optymalizacja logiczna, fizyczna i wybór planu realizacji na podstawie kosztu są ze sobą ściśle powiązane [53]. Z tego względu operuje się ogólnym terminem „optymalizacja", który obejmuje wszystkie trzy wymienione zagadnienia.

Generacja kodu ma na celu przekształcenie wybranego planu fizycznego w kod wykonywalny. Rozwiązanie alternatywne polega na zastosowaniu interpretera bezpośrednio przetwarzającego plan realizacji.

3. Problemy optymalizacyjne

Jedno z praw algebry relacji, prawo łączności operacji złączenia ($R \rhd\lhd (S \rhd\lhd T) = (R \rhd\lhd S) \rhd\lhd T$, gdzie R, S, T oznaczają relacje, a symbol $\rhd\lhd$ oznacza operację złączenia) [144, 142], pozwala na rozważenie wielu wariantów określania kolejności złączeń relacji biorących udział w zapytaniu. Poszukiwanie optymalnej kolejności złączeń jest kluczowym elementem optymalizacji zapytania i zalicza się je do klasy problemów NP-trudnych [60]. Porządkowanie ciągów innych operacji dwuargumentowych (suma, przecięcia) ma mniejsze znaczenie, ponieważ rzadziej występują one w grupach, a ich wykonanie zajmuje zwykle mniej czasu niż realizacja złączenia [45].

Określanie kolejności złączeń ma szczególne znaczenie dla niniejszej rozprawy, dlatego zostanie ono szczegółowo omówione w podrozdziale 3.1.

3.1. Problem określenia kolejności złączeń w realizacji zapytań bazodanowych

Zakłada się, że przed przystąpieniem do wyznaczania kolejności złączeń dane są wstępnie przetworzone. Oznacza to, że na tabelach bazy danych, biorących udział w zapytaniu, przeprowadzono, zgodnie z własnościami wyrażeń algebry relacji, operacje projekcji i selekcji. Dla określenia takich wstępnie przetworzonych danych, w niniejszej pracy stosuje się termin *zbiór rekordów*.

Proces łączenia danych w zbiór rekordów stanowiący wynik końcowy zapytania adresowanego do scentralizowanej bazy danych przebiega wieloetapowo. Polega on na wykonaniu wielu operacji złączenia. Celem każdej z nich jest złączenie dwóch zbiorów rekordów, z których każdy może być zbiorem pobranym bezpośrednio z bazy scentralizowanej (zbiór pierwotny) lub może stanowić wynik wcześniej przeprowadzonej operacji złączenia (zbiór pośredni, wynik pośredni).

W prowadzonych pracach skupiono uwagę na operacji złączenia [138]. Celem niniejszego zadania optymalizacji jest wyznaczenie takiego porządku realizacji złączeń, aby wynik końcowy zapytania został przedstawiony użytkownikowi w jak najkrótszym czasie.

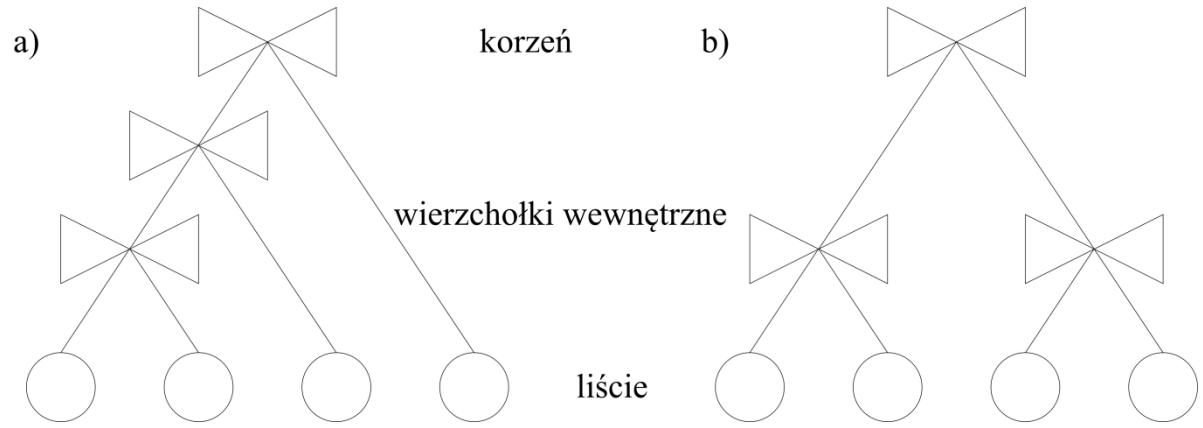
Przestrzenią poszukiwań dla rozpatrywanego zagadnienia jest zbiór planów realizacji zadanego ciągu złączeń. Pojedynczy plan realizacji może być reprezentowany przez drzewo realizacji złączeń (ang. *join processing tree*). W drzewie realizacji złączeń wyróżnia się następujące elementy (rys. 2):

- liście przedstawiające zbiory pierwotne,
- wierzchołki wewnętrzne, które symbolizują operacje złączenia; wierzchołek nazywany korzeniem jest elementem, któremu przypisano złączenie realizowane jako ostatnie;

drzewo o n liściach posiada n-1 wierzchołków wewnętrznych (a więc realizowanych jest n-1 złączeń),

- krawędzie łączące każdy wierzchołek wewnętrzny z dwoma niżej położonymi elementami drzewa (wierzchołkami wewnętrznymi lub liśćmi); każda taka trójka elementów wyraża pojedynczą operację złączenia i jej argumenty. Krawędzie ilustrują przepływ danych od poziomu liści aż do korzenia.

Układ krawędzi, wierzchołków i liści decyduje o kształcie drzewa. Jeśli do każdego wierzchołka wewnętrznego prowadzi co najmniej jedna krawędź wychodząca z liścia, to drzewo charakteryzuje się sekwencyjnym układem wierzchołków (rys. 2a). Wystąpienie co najmniej jednego wierzchołka, do którego prowadzą krawędzie wychodzące z wierzchołków wewnętrznych, a nie z liści, oznacza potencjalną możliwość równoczesnej realizacji złączeń reprezentowanych przez te dwa wierzchołki. Drzewo o takiej własności nosi nazwę drzewa o równoległym układzie wierzchołków (rys. 2b).



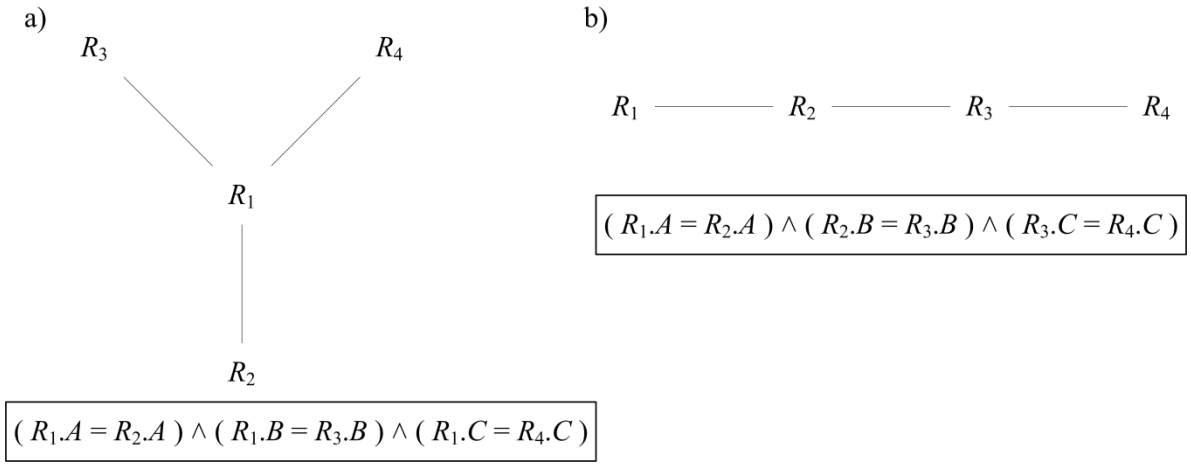
Rys. 2. Kształty drzew realizacji złączeń o układzie wierzchołków: a) sekwencyjnym, b) równoległym.

Jedną z form opisu zapytania stanowi graf zapytania (ang. *query graph*). Graf $G = (\{R_1,\ldots,R_n\}, E)$, który reprezentuje wyłącznie ciąg operacji złączenia, nazywany jest grafem złączeń (ang. *join graph*). Stanowi on strukturę złożoną z wierzchołków połączonych nieskierowanymi krawędziami. Wierzchołki grafu są odpowiednikami zbiorów rekordów $R_1,\ldots,R_n$. Każda z krawędzi, reprezentowanych przez zbiór E, który jest zbiorem par (R_i, R_j), $1 \le i, j \le n$, $i \ne j$, symbolizuje fakt istnienia wyrażenia łączącego między jednostkami tworzącymi pojedynczą parę [123].

Na szczególną uwagę zasługują charakterystyczne kształty grafu złączeń różniące się postacią wyrażeń łączących [62]:

- graf gwiazdy (ang. *star query graph*), w którym jeden ze zbiorów rekordów występuje w każdym wyrażeniu łączącym (jest centralnym wierzchołkiem grafu przedstawionego na rys. 3a),

- graf łańcuchowy (ang. *chain query graph*), gdzie dwa zbiory rekordów (stanowiące skrajne wierzchołki grafu przedstawionego na rys. 3b) występują tylko jednokrotnie w wyrażeniach łączących, natomiast pozostałe zbiory – dwukrotnie.



Rys. 3. Graf złączeń i odpowiadające mu złożone wyrażenie łączące: a) graf gwiazdy, b) graf łańcuchowy.

Literatura związana z tematem wyznaczania kolejności złączeń w realizacji zapytań bazodanowych jest bardzo bogata. Zadanie to jest rozwiązywane m.in. przy użyciu następujących metod: Minimum Selectivity, Top-Down, KBZ, Relational Difference Calculus, AB [130, 129, 134], a także za pomocą programowania dynamicznego [105], przeszukiwania lokalnego [133] oraz metod probabilistycznych [139]. Wykorzystywano w tym celu również metaheurystyki kombinatoryczne: algorytm symulowanego wyżarzania [63, 130, 129], algorytm iteracyjnego poprawiania [63, 130, 129], algorytm z listą tabu [108, 100], a także algorytmy genetyczne [34, 131] i metody hybrydowe [63, 92, 99].

Opisane zadanie jest zagadnieniem bardzo trudnym do rozwiązania. Wydaje się więc zasadne w pierwszej kolejności obszerne przetestowanie rozważanego w niniejszej pracy algorytmu heurystycznego IWO, rozwiązując klasyczne zadania optymalizacji, a dopiero potem przystąpienie do eksperymentów dotyczących problemu wyznaczania kolejności złączeń.

Zadania optymalizacyjne można podzielić na dwie kategorie: o charakterze ciągłym oraz dyskretnym, do której należy również określanie kolejności złączeń w realizacji zapytania kierowanego do bazy danych. Najpopularniejszymi i najczęściej poruszanymi w literaturze zagadnieniami są [26, 8, 113, 46]:

- znajdowanie ekstremum funkcji matematycznej,
- problem komiwojażera,
- określanie cyklu Hamiltona,
- znajdowanie maksymalnej kliki w grafie,
- problem trójkolorowalności,
- znajdowanie pokrycia wierzchołkowego,

- kolorowanie grafu,
- problem plecakowy,
- szukanie największego wspólnego podgrafu,
- określanie drzewa rozpinającego o minimalnym stopniu,
- znajdowanie najdłuższego wspólnego podciągu,
- problem podziału zbioru na podzbiory i wiele innych.

W niniejszej pracy postanowiono skupić się na problemach znajdowania minimum funkcji matematycznej oraz komiwojażera. Wybór padł na te zadania ze względu na ich ogólność oraz bogatą literaturę.

3.2. Problem znajdowania minimum funkcji wielowymiarowych

Problem znajdowania minimum funkcji wielowymiarowych (funkcji wielu zmiennych) jest podstawowym zadaniem optymalizacji o charakterze ciągłym i sprowadza się do wyznaczenia wartości argumentów funkcji, dla których funkcja ta posiada ekstremum globalne, w tym przypadku minimum. Minimum globalne funkcji f ma miejsce w punkcie x_0 wtedy, gdy dla dowolnego x należącego do dziedziny funkcji f zachodzi zależność $f(x_0) \leq f(x)$ dla $x \neq x_0$ [56]. Należy zauważyć, że problem ten ma nieskończenie dużą liczbę potencjalnych rozwiązań. W związku z tym odnalezienie wartości optymalnych bez uwzględnienia dodatkowych założeń jest praktycznie niemożliwe.

Zgodnie z pracą [106] wyróżnia się cztery rodzaje funkcji matematycznych stosowanych do testowania algorytmów optymalizacyjnych:

1. Jednomodalne, wypukłe, wielowymiarowe.
2. Wielomodalne, dwuwymiarowe z małą liczbą ekstremów lokalnych.
3. Wielomodalne, dwuwymiarowe z bardzo dużą liczbą ekstremów lokalnych.
4. Wielomodalne, wielowymiarowe z bardzo dużą liczbą ekstremów lokalnych.

Grupa pierwsza to zwykle funkcje proste pod względem optymalizacyjnym. W niektórych przypadkach ich charakterystyka powoduje bardzo powolne zbliżanie się do jednego globalnego minimum. Do testowania standardowych procedur optymalizacyjnych stosuje się funkcje klasy drugiej, zawierające kilka lokalnych minimów i jedno, dość wyraźne minimum globalne. Dwie ostatnie grupy zawierają funkcje, które są uważane przez autorów pracy [106] za bardzo trudne pod względem optymalizacyjnym. Znajdują one zastosowanie w testowaniu jakości najlepszych, najodporniejszych na trudne warunki metod optymalizacyjnych.

Najbardziej znanymi i najczęściej stosowanymi algorytmami, które pozwalają na optymalizację problemu znajdowania minimum funkcji wielowymiarowych są: algorytmy genetyczne [109], przeszukiwanie z listą tabu [20, 58], symulowane wyżarzanie [41], algorytm kolonii pszczół [73], algorytm mrówkowy [127] oraz algorytm IWO [101].

3.3. Problem komiwojażera

Problem komiwojażera został po raz pierwszy sformułowany w latach trzydziestych ubiegłego wieku i od tego czasu jest jednym z najbardziej znanych i rozpowszechnionych zadań optymalizacyjnych o charakterze dyskretnym [5]. Ma on bardzo wiele odmian o praktycznym zastosowaniu (przy wytwarzaniu elektronicznych płytek drukowanych, dla zmniejszenia kosztów transportu czy przy trasowaniu pojazdów z oknami czasowymi [27]).

Nazwa niniejszego zagadnienia pochodzi od wędrującego od miasta do miasta sprzedawcy (komiwojażera). W najprostszej wersji tego zadania dana jest pewna liczba miejsc, które mają być odwiedzone przez sprzedawcę dokładnie jeden raz, oraz koszt podróży między poszczególnymi lokalizacjami. Koszt ten wyrażany jest najczęściej za pomocą odległości, czasu podróży lub jego ceny. Celem zadania optymalizacji jest wyszukanie takiej trasy, prowadzącej przez wszystkie miasta, która ma najniższy koszt przejścia. Zwykle nakłada się dodatkowe założenie określające punkt startowy, będący zarazem miejscem docelowym komiwojażera.

Rozróżnia się dwie podstawowe wersje zagadnienia: symetryczną i asymetryczną. Symetryczny problem komiwojażera polega na tym, że koszt podróży między dwoma miastami w obydwu kierunkach jest jednakowy. Natomiast w odmianie asymetrycznej koszt ten może być różny. Symetryczna wersja jest zatem szczególnym przypadkiem asymetrycznej [57].

Liczba wszystkich możliwych tras dla komiwojażera asymetrycznego (zakładając, że każde miasto jest połączone z każdym innym) wynosi $(n-1)!$, gdzie n oznacza liczbę miast, z kolei dla problemu symetrycznego liczba ta jest dwukrotnie mniejsza [54]. Zakładając chęć odwiedzenia wszystkich 908 miast w Polsce [52], należałoby przeanalizować około $1{,}67 \cdot 10^{2290}$ tras, by znaleźć drogę optymalną. Z praktycznego punktu widzenia nie da się rozpatrzeć wszystkich możliwości. Zagadnienie komiwojażera jest zaliczane do problemów NP-trudnych.

Ze względu na taką złożoność problem komiwojażera rozwiązuje się zwykle przez zastosowanie metod heurystycznych [1]. Najbardziej znanymi i najczęściej stosowanymi algorytmami służącymi do optymalizacji niniejszego problemu są: 2-opt i 3-opt [94] wraz z ich wariantami, przeszukiwanie z listą tabu [47], algorytm Lina-Kernighana [59], symulowane wyżarzanie [2], algorytm mrówkowy [35] i genetyczny [93].

Jedną z najbardziej efektywnych metod rozwiązujących niniejszy problem jest algorytm Concorde. Za jego pomocą w marcu 2005 roku znaleziono trasę optymalną dla zadania zawierającego 33 810 miast, a obliczenia zajęły 15,7 procesoro-lat. Natomiast w kwietniu 2006 rozwiązano zadanie z 85 900 miejsc (czas wykonania wyniósł 136 procesoro-lat) [5]. Użycie metod heurystycznych pozwala na otrzymanie wyniku trochę gorszego w stosunku do optimum w rozsądnym czasie [32, 34, 73].

4. Przegląd algorytmów optymalizacyjnych

Algorytmy służące do rozwiązywania problemów optymalizacyjnych polegają zazwyczaj na podejmowaniu ciągu decyzji, opartych na zestawie przyjętych reguł. Algorytmy takie korzystają z operacji ekspansji lub transformacji (zwanymi także poszukiwaniem konstruktywnym lub zaburzeniowym [137]). Ekspansja polega na budowaniu rozwiązania znajdując kolejne jego elementy, natomiast transformacja przekształca pełne rozwiązanie w inne. Gdy rozmiar przestrzeni poszukiwań powoduje, że analiza wszystkich rozwiązań staje się zbyt czasochłonna, stosuje się inne metody pozwalające na ograniczenie się do analizy określonego podzbioru rozwiązań.

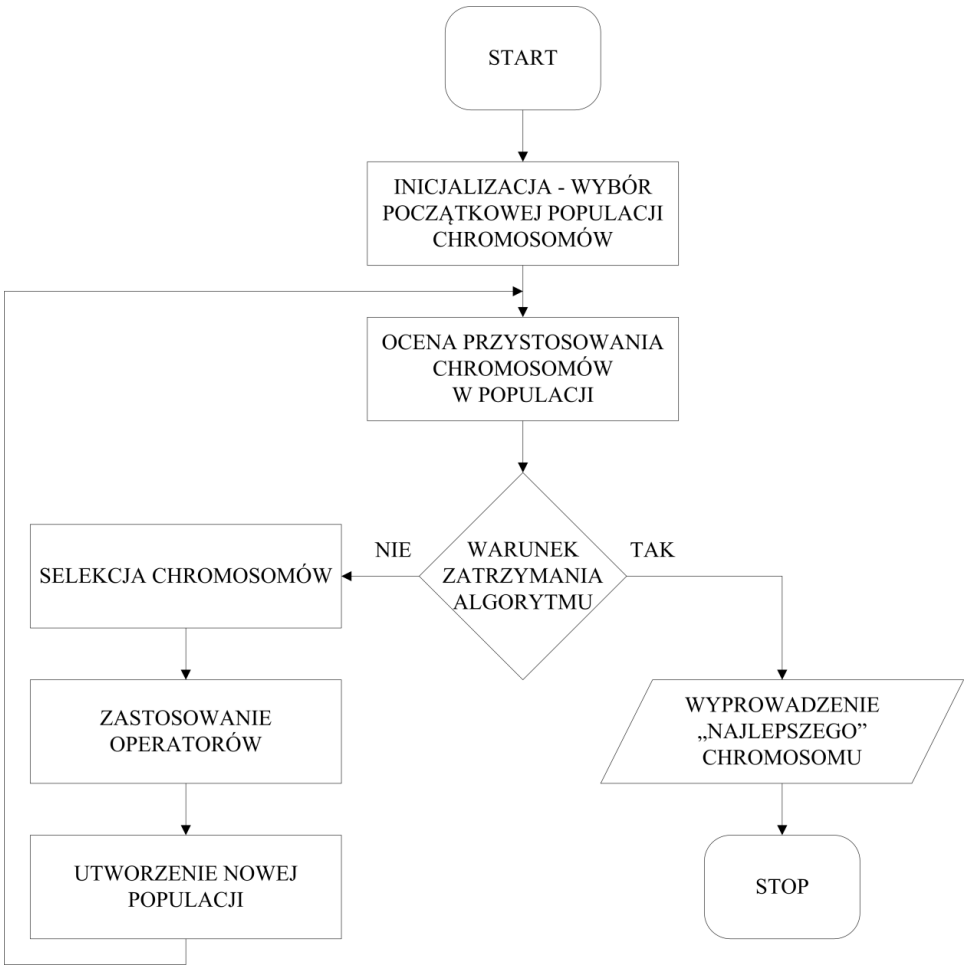
Niniejszy rozdział zawiera opis najważniejszych strategii optymalizacyjnych oraz miejsce algorytmu IWO w świetle aktualnej literatury.

IWO należy do grupy algorytmów ewolucyjnych. Z tego względu na wstępnie zostanie objaśniona istota takich algorytmów.

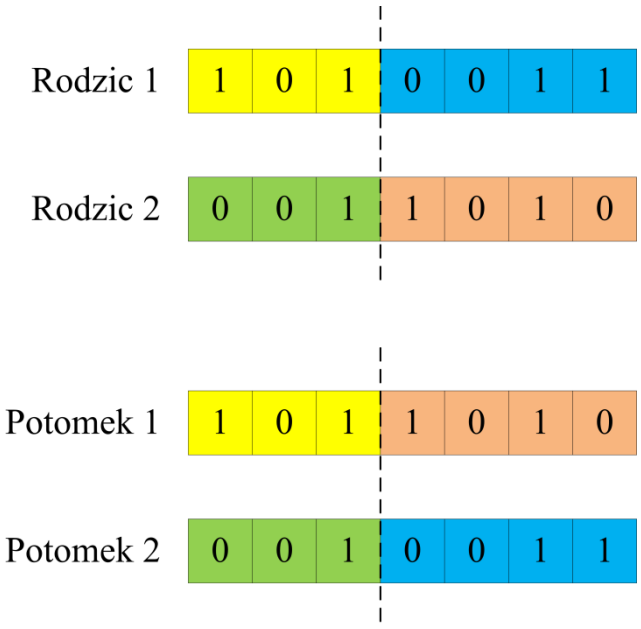
4.1. Algorytm ewolucyjny

Terminem „algorytmy ewolucyjne” określa się komputerowe systemy rozwiązywania problemów optymalizacji, działające na zasadach, jakie można zaobserwować w ewolucji organizmów żywych [114]. Podstawy działania niniejszego algorytmu wzorowane są na zjawisku ewolucji i doboru naturalnego. Ideę tę przedstawił na przełomie lat sześćdziesiątych i siedemdziesiątych XX wieku John Henry Holland. Rysunek 4 prezentuje schemat blokowy algorytmu ewolucyjnego.

W algorytmie ewolucyjnym początkowo losuje się pewną liczbę rozwiązań, zwanych chromosomami lub osobnikami, a następnie określa się ich jakość. Jest ona wyznaczana przez obliczenie wartości pewnej funkcji, ściśle zależnej od optymalizowanego problemu, zwanej funkcją przystosowania. Zazwyczaj osobnikom lepiej przystosowanym daje się większe szanse tworzenia chromosomów potomnych, wchodzących w skład kolejnej populacji. Proces tworzenia nowych rozwiązań odbywa się drogą krzyżowania chromosomów lub mutacji. Wspomniane metody tworzenia nowych osobników zwane są operatorami genetycznymi i stanowią centralny punkt algorytmu. Krzyżowanie polega na skojarzeniu dwóch osobników w parę i wybraniu miejsca „cięcia”. Następnie potomkowie takiej pary otrzymują część danych od pierwszego, a część od drugiego rodzica [18]. Krzyżowanie jednopunktowe w najprostszej postaci przedstawia rys. 5.



Rys. 4. Schemat blokowy algorytmu ewolucyjnego.



Rys. 5. Przykładowy operator krzyżowania.

Na operator często nakłada się dodatkową własność, która mówi, że skrzyżowanie dwóch identycznych osobników ze sobą powinno dać w rezultacie takiego samego osobnika [141]. Operator mutacji jest stosowany znacznie rzadziej niż operator krzyżowania i polega na zmianie wartości losowo wybranego genu (stanowiącego część składową osobnika). Często nakłada się na tworzony operator mutacji dodatkową własność, która mówi, że każdy osobnik, po skończonej liczbie wykonanych mutacji, powinien dać w efekcie dowolnego innego osobnika [141].

Mimo, że z biologicznego punktu widzenia przedstawione podejście jest znacznym uproszczeniem procesów ewolucyjnych, to algorytmy te są wystarczająco mocnym narzędziem gwarantującym bardzo szybkie znajdowanie dobrych rozwiązań [114]. Są one w dalszym ciągu obiektem badań i modyfikacji, co znajduje odbicie w wielu publikacjach [24, 9, 148].

4.2. Algorytm IWO

Algorytm IWO (ang. *Invasive Weed Optimization*) jest metodą optymalizacyjną, która została zainspirowana charakterystycznym dla chwastów gwałtownym rozprzestrzenianiem się oraz szybkim przystosowywaniem się do zmiennych warunków otoczenia. Algorytm ten po raz pierwszy został przedstawiony w 2006 roku przez naukowców Uniwersytetu w Teheranie, A. R. Mehrabiana i C. Lucasa [101].

IWO należy do rodziny algorytmów ewolucyjnych. Wprowadzono w nim nazewnictwo luźno odnoszące się do słownictwa związanego z chwastami (tabela 1).

Tabela 1
Zestawienie nazewnictwa algorytmu ewolucyjnego i algorytmu IWO

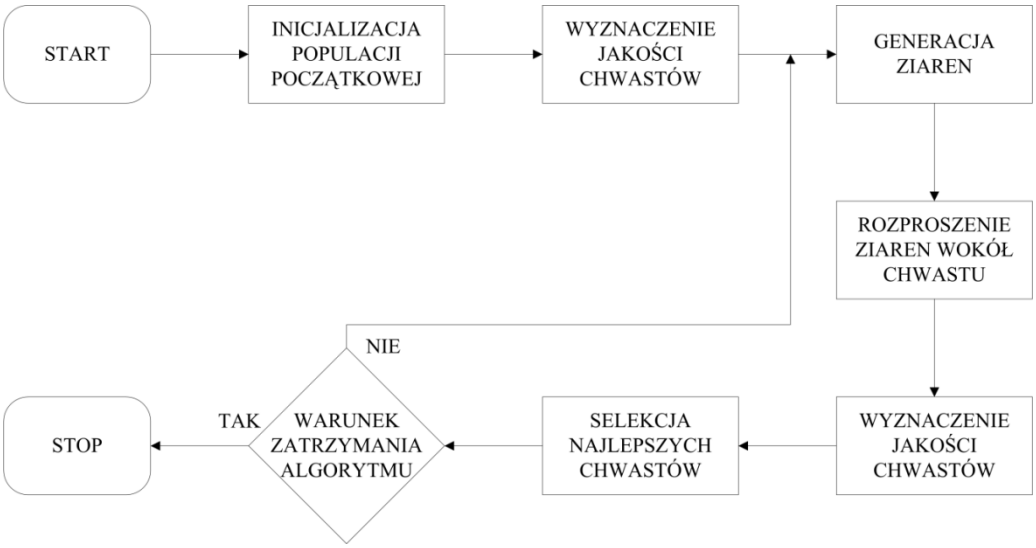
Nazwa ogólna	**Algorytm ewolucyjny**	**Algorytm IWO**
iteracja	pokolenie / populacja	pokolenie / populacja
rozwiązanie	osobnik / chromosom	Chwast
rozwiązanie pochodne	potomek	Ziarno
transformacja	krzyżowanie / mutacja	rozproszenie

4.2.1. Idea algorytmu IWO

Na rys. 6 przedstawiono schemat blokowy algorytmu IWO.

Inicjalizacja populacji początkowej

Pierwszy krok polega na swobodnym rozproszeniu ziaren chwastów po całej przestrzeni poszukiwań. Odpowiada to losowemu wyborowi określonej liczby propozycji rozwiązań rozpatrywanego problemu. Liczba chwastów biorących udział w populacji jest jednym z parametrów algorytmu.



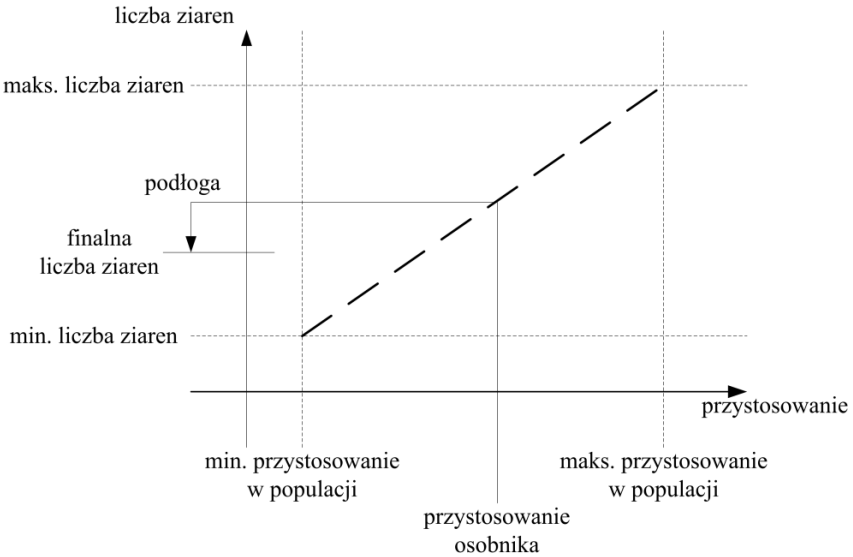
Rys. 6. Schemat blokowy algorytmu IWO.

Wyznaczenie jakości chwastów

Realizacja tego etapu jest równoznaczna z oceną rozwiązania reprezentowanego przez analizowany chwast. Sprowadza się to do obliczenia wartości pewnej funkcji związanej z kryterium optymalizacji, zwanej funkcją przystosowania. Jest ona budowana na podstawie modelu obliczeniowego, ściśle zależnego od rozwiązywanego zadania optymalizacyjnego.

Generacja ziaren

Każdy chwast tworzy pewną liczbę ziaren. Liczba ta jest powiązana z przystosowaniem do otoczenia generującego je osobnika. Im lepsze jest dane rozwiązanie, tym większą liczbę ziaren może wytworzyć. Zależność ta ma charakter liniowy i można ją przedstawić jak na rys. 7.



Rys. 7. Metoda wyznaczania liczby ziaren.

Liczbę generowanych przez chwast ziaren można zatem zapisać następującym wzorem [101]:

$$S_{chw} = S_{min} + \left\lfloor (f_{chw} - f_{min}) \cdot \frac{S_{max} - S_{min}}{f_{max} - f_{min}} \right\rfloor \tag{1}$$

gdzie: S_{chw} – liczba ziaren wygenerowanych przez analizowany chwast; S_{min}, S_{max} – minimalna i maksymalna liczba ziaren, jaka może być wygenerowana przez chwast; f_{chw} – wartość funkcji przystosowania analizowanego chwastu; f_{min}, f_{max} – minimalna i maksymalna wartość funkcji przystosowania osobników w aktualnym pokoleniu chwastów.

Rozproszenie ziaren wokół chwastu

Krok ten polega na umieszczeniu nowopowstałych ziaren *Z* w pobliżu osobnika macierzystego *M*. Odbywa się to zgodnie z pseudokodem opracowanym na podstawie opisu algorytmu w pracy [101] (rys. 8).

```
Dla aktualnego ziarna Z wykonaj:
   Wylosuj odległość upadku ziarna Z od rośliny macierzystej M
      zgodnie z rozkładem normalnym N(0,σ_iter), gdzie σ_iter
      jest wyznaczane zgodnie ze wzorem 2.
   Wylosuj kierunek przesunięcia ziarna Z względem rośliny
      macierzystej M.
   Z = M
   Dokonaj transformacji Z zgodnie z wyznaczonymi odległością
      i kierunkiem.
   Zwróć Z.
```

Rys. 8. Rozpraszanie ziaren wokół chwastu w algorytmie IWO.

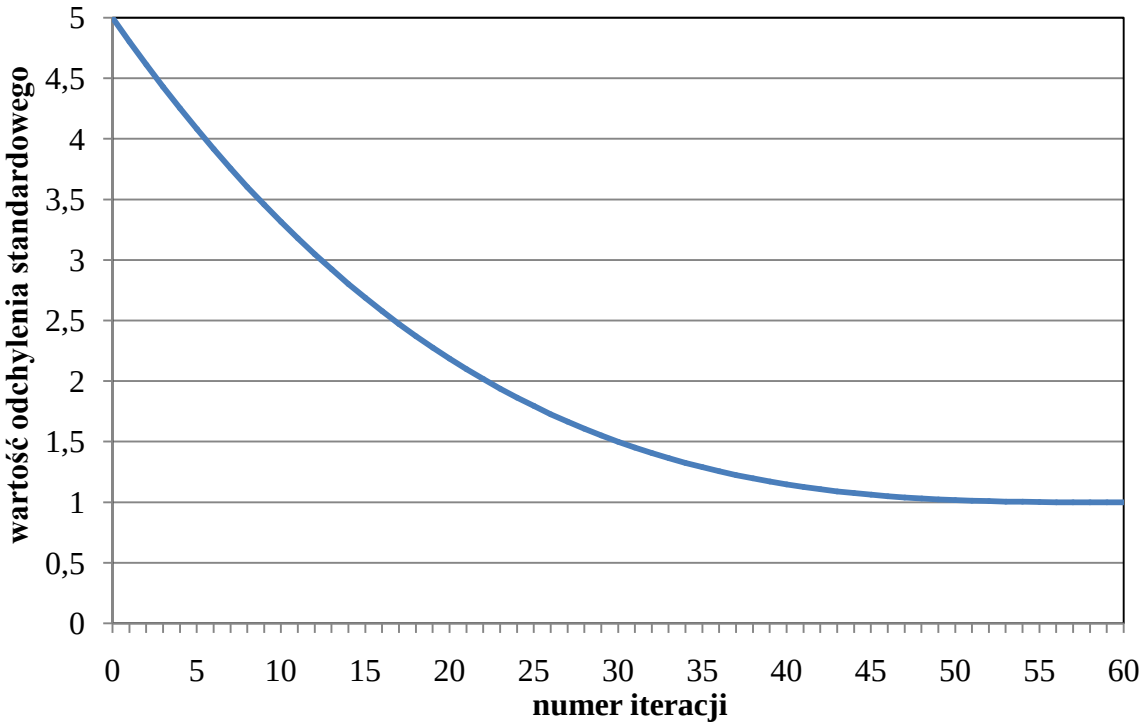
Rozproszenie na powierzchni wymaga określenia dwóch współczynników: odległości i kierunku. Odległość między chwastem a ziarnem potomnym jest opisana za pomocą rozkładu normalnego o wartości oczekiwanej 0 i zmiennym odchyleniu standardowym. Odchylenie zmniejsza się wraz ze wzrostem numeru iteracji, co prezentuje następujące równanie [101]:

$$\sigma_{iter} = \left(\frac{iter_{max} - iter}{iter_{max}} \right)^m (\sigma_{init} - \sigma_{fin}) + \sigma_{fin} \tag{2}$$

gdzie: *iter* – aktualny numer iteracji algorytmu; $iter_{max}$ – łączna liczba iteracji algorytmu; σ_{iter} – odchylenie standardowe dla iteracji o numerze *iter*; σ_{init} – odchylenie standardowe początkowe; σ_{fin} – odchylenie standardowe końcowe; *m* – współczynnik modulacji nieliniowej. Rola współczynnika modulacji nieliniowej sprowadza się do określenia kształtu krzywej, według której zmienia się wartość odchylenia standardowego. Rys. 9 ilustruje przykładową zmianę wartości odchylenia standardowego dla następujących danych: $iter_{max} = 60$; $\sigma_{init} = 5{,}0$; $\sigma_{fin} = 1{,}0$; $m = 3{,}0$.

W przypadku, gdy algorytm rozwiązuje problem optymalizacyjny o charakterze ciągłym, konstrukcja nowego osobnika polega na wykonaniu na kopii chwastu macierzystego transformacji, oddalającej potomka o odległość obliczoną w poprzednim punkcie algorytmu, a kierunek jest dobierany losowo (zakłada się rozkład równomierny).

Dla problemów dyskretnych wykonuje się transformacje, w liczbie równej zdyskretyzowanej wartości wyznaczonej przez rozkład normalny. Określenie losowego kierunku przesunięcia ziarna sprowadza się do zapewnienia równego prawdopodobieństwa każdej z generujących poprawne rozwiązanie transformacji.



Rys. 9. Przykładowy kształt krzywej, według której zmienia się wartość odchylenia standardowego.

Selekcja najlepszych chwastów

W związku z tworzeniem ziaren, których liczebność może kilkukrotnie przewyższać początkową liczbę chwastów, zachodzi konieczność selekcji osobników. Z grupy $(\mu + \lambda)$ złożonej ze wszystkich osobników macierzystych μ i potomnych λ wybierane są rozwiązania najlepsze jakościowo, zachowując stałą liczebność tworzonej w ten sposób populacji.

Warunek zatrzymania algorytmu

Ostatnim etapem jest sprawdzenie warunku zatrzymania algorytmu, którym jest zazwyczaj założona z góry liczba iteracji.

W tabeli 2 przedstawiono zestawienie wszystkich parametrów algorytmu.

Tabela 2

Zestawienie parametrów algorytmu IWO

Nazwa parametru	**Typ**
liczba pokoleń	liczba naturalna
liczba chwastów w pokoleniu	liczba naturalna
maksymalna liczba ziaren	liczba naturalna
minimalna liczba ziaren	liczba naturalna
początkowe odchylenie standardowe odległości pomiędzy rośliną macierzystą a potomną	liczba rzeczywista dodatnia
końcowe odchylenie standardowe odległości pomiędzy rośliną macierzystą a potomną	liczba rzeczywista dodatnia mniejsza od początkowego odchylenia standardowego
współczynnik modulacji nieliniowej	liczba rzeczywista

4.2.2. Analiza złożoności obliczeniowej

Do pełnego opisu algorytmu konieczne jest wyznaczenie jego złożoności obliczeniowej. Autorowi nie jest znana żadna próba oceny złożoności IWO opisana w literaturze. Należy mieć na uwadze fakt, iż charakter i rozmiar zadania optymalizacji w różnym stopniu wpływają na czas wykonania algorytmu. W związku z tym rozważania zawarte w niniejszym podrozdziale dotyczą jedynie ogólnej idei IWO i nie są związane z żadnym konkretnym problemem optymalizacyjnym. W podrozdziale 5.3.4 zostanie podjęta próba oszacowania złożoności obliczeniowej dla problemu wyznaczania kolejności złączeń w realizacji zapytań relacyjnej bazy danych.

Złożoność zostanie wyznaczona na podstawie kolejno wykonywanych elementów algorytmu. Jako operacje dominujące przyjęto przebieg pętli realizowanych w algorytmie.

Faza wstępna, czyli przed rozpoczęciem ewolucji osobników, składa się z inicjalizacji pierwszego pokolenia chwastów oraz oceny ich jakości. Złożoność tych czynności jest liniowo zależna od liczby stworzonych osobników. Liczba ta jest parametrem IWO.

$$T_{init} = O[\text{liczba\_chwastów (ze względu na inicjalizację)} + \text{liczba\_chwastów (ocena jakości)}] = O[\text{liczba\_chwastów}] \quad (3)$$

Kolejne etapy algorytmu są wykonywane iteracyjnie. Liczba iteracji, będąca zarazem liczbą pokoleń, jest również parametrem określanym przed uruchomieniem metody.

Pierwszym krokiem należącym do fazy ewolucji osobników jest wyznaczenie liczby ziaren. Rozpatrując przypadek pesymistyczny można założyć, że każdy chwast wygeneruje maksymalną ustaloną ich liczbę. Dla każdego ziarna wyznaczane jest miejsce upadku, a następnie obliczana jest jakość nowopowstałego chwastu. Opisane etapy można odwzorować za pomocą następującej zależności:

$$T_{ziaren} = O[\text{liczba\_chwastów (wyznaczenie liczby ziaren)} + \text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren (rozproszenie + ocena jakości)}] = O[\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren}] \quad (4)$$

Ostatnim elementem algorytmu jest selekcja. Do przeprowadzenia tej operacji przydatne jest posortowanie chwastów ze względu na ich przystosowanie. Można w tym celu wykorzystać strukturę drzewiastą zapewniającą, że elementy będą dodawane do struktury ze złożonością logarytmiczną. Następnie z drzewa pobierana jest wymagana liczba najlepiej przystosowanych osobników, zapewniając tym samym stworzenie nowego pokolenia (równanie 5).

$$T_{selekcji} = O[\log(\text{liczba\_chwastów} + \text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren}) + \text{liczba\_chwastów (pobranie ze struktury)}] = O[\log(\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren}) + \text{liczba\_chwastów}] \quad (5)$$

Z zależności 3, 4 i 5 wynika następująca całkowita złożoność algorytmu IWO:

$$T_{IWO} = O[T_{init} + \text{liczba\_iteracji} \cdot (T_{ziaren} + T_{selekcji})] \quad (6)$$

Po podstawieniu:

$$T_{IWO} = O[\text{liczba\_chwastów} + \text{liczba\_iteracji} \cdot (\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} + \log(\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren}) + \text{liczba\_chwastów})] \quad (7)$$

Zakładając, że liczba chwastów jest duża, można pominąć wyrażenie z logarytmem, którego wartość jest znacznie mniejsza. W ten sposób otrzymuje się końcową złożoność obliczeniową rzędu:

$$T_{IWO} = O[\text{liczba\_chwastów} \cdot \text{liczba\_iteracji} \cdot \text{maks\_liczba\_ziaren}] \quad (8)$$

4.2.3. Zastosowania algorytmu IWO w problemach o charakterze ciągłym

Algorytm IWO od samych początków swojego istnienia zyskuje na popularności. Znajduje zastosowanie w wielu praktycznych problemach optymalizacyjnych zarówno o charakterze ciągłym, jak i dyskretnym.

Autorzy algorytmu w swojej pierwszej pracy zajęli się określaniem minimum funkcji wielowymiarowych [101]. Wykorzystali oni takie funkcje, jak: pierwsza funkcja De Jonga, funkcje Griewanka, Rastrigina i inne, a uzyskane wyniki porównali z wynikami algorytmów genetycznego (ang. *genetic algorithm*, GA), memetycznego (ang. *memetic algorithm*, MA), roju cząstek (ang. *particie swarm optimization*, PSO) oraz skaczących żab (ang. *shuffled frog leaping*, SFL).

W wielu pracach [98, 97, 74] algorytm IWO stosowany jest do określenia współczynników definiujących wymiary i kształty anten, w tym anten wieloelementowych. Anteny takie są wykorzystywane m.in. w sieciach bezprzewodowych do osiągania przepustowości powyżej 100Mb/s.

W artykule [122] autorzy przedstawili implementację IWO do systemu wspomagającego handel na rynku energetycznym. W przypadku takiego rynku każdy dostawca energii elek-

trycznej składa swoją ofertę w formie krzywej opisującej cenę względem ilości produkowanej energii. Następnie niezależni operatorzy systemu energetycznego pozwalają na dokonanie transakcji, biorąc pod uwagę ograniczenia systemu, takie jak maksymalne napięcie i możliwości linii przesyłowych. Zastosowanie algorytmu IWO pozwala na licytację zarówno ceny, jak i ilości sprzedawanej energii elektrycznej.

Praca [49] zawiera opis zmodyfikowanego algorytmu IWO służącego do dostosowania w fazie uczenia wag i stałych jednokierunkowej sztucznej sieci neuronowej. Wspomniana modyfikacja algorytmu polega na zmianie funkcji służącej do wyznaczania odchylenia standardowego odległości pomiędzy rośliną macierzystą a potomną w zależności od numeru aktualnego pokolenia. Uzyskane wyniki porównano z kilkoma innymi algorytmami uczenia, m.in. metodą wstecznej propagacji błędów, ewolucji różnicowej czy jednokrokową metodą siecznych.

4.2.4. Zastosowania IWO w problemach o charakterze dyskretnym

H. Sepehri wraz z jednym z autorów IWO, C. Lucasem, zastosowali algorytm IWO w systemie rekomendacji na stronach WWW [125]. Systemy te nabrały dużego znaczenia komercyjnego, ponieważ pomagają potencjalnym klientom korzystającym z serwisów internetowych odnaleźć się w gąszczu produktów.

Dyskretną odmianę algorytmu IWO z powodzeniem wykorzystano także do przydzielania zadań cywilnym samolotom bezzałogowym [48]. Jakość rozwiązań otrzymanych przy jego użyciu porównano z wynikami uzyskanymi za pomocą algorytmów: genetycznego, memetycznego, roju cząstek, mrówkowego i skaczących żab. Funkcję kosztu oparto na dwóch czynnikach: czasie realizacji zadań oraz bezpośredniej cenie ich wykonania.

W kolejnym przypadku zastosowano algorytm IWO do wielokryterialnej optymalizacji pokonywanej przez kierowcę samochodu drogi w środowisku miejskim [112]. Zmodyfikowano algorytm w taki sposób, by mógł być wykorzystany do problemu dyskretnego. Przyjęto następujące miary oceny uzyskiwanych rozwiązań: czas podróży, dystans oraz trudność przebytej drogi. Trudność trasy mierzono przez zliczanie zakrętów wykonywanych przez kierowcę pojazdu. W pracy rozważano przejazd przez skrzyżowanie w prawostronnym ruchu ulicznym. Zauważono, że zdecydowanie łatwiej i szybciej skręca się w prawo niż w lewo.

Chińscy naukowcy z kolei przystosowali algorytm IWO do kodowania informacji w sekwencjach DNA [146]. Kryteriami oceny stały się ograniczenia termodynamiczne. Skorzystano z dwóch najbardziej istotnych parametrów, którymi są wolna energia oraz temperatura topnienia.

4.3. Inne algorytmy optymalizacyjne

W niniejszym podrozdziale krótko opisano inne algorytmy optymalizacyjne, które mogą stanowić przeciwwagę dla opisywanego wcześniej algorytmu IWO. Zaprezentowane metody należą do dwóch grup: ekspansywnych i opierających się na transformacji.

Algorytmy zachłanne [110] należą do grupy algorytmów ekspansywnych. Wykonują one zawsze działania, które wydają się w danej chwili najkorzystniejsze. Wybierają lokalnie optymalną możliwość, w nadziei, że doprowadzi ona do globalnie optymalnego rozwiązania. Algorytmy te praktycznie nie sprawdzają, czy otrzymany wynik może być rozwiązaniem optymalnym. Często nawet zdarza się, że rezultat działania jest niepoprawny, dlatego ważnym elementem jest analiza poprawności działania algorytmu zachłannego [26].

Wiele algorytmów opiera się na transformacji, polegającej na przekształceniu pełnego rozwiązania w inne, jako metodzie przeszukiwania przestrzeni możliwych rozwiązań [111]. W dalszej części niniejszego podrozdziału kolejno omówionych zostanie kilka najbardziej popularnych algorytmów tego typu.

Działanie algorytmu największego wzrostu [104] polega na poszukiwaniu maksimum lokalnego, dla losowo wybranego rozwiązania początkowego. Z punktu reprezentującego aktualnie analizowane rozwiązanie przechodzi się do takiego punktu sąsiedniego, który jest najlepszy. Punktem sąsiednim jest rozwiązanie, które może być otrzymane w wyniku pojedynczej transformacji rozwiązania aktualnego. Po dotarciu do maksimum lokalnego algorytm jest zatrzymywany. W przypadku stosowania tego algorytmu konieczne jest zastosowanie tzw. multistartu, czyli wielokrotnego uruchomienia algorytmu. Dzięki takiemu rozwiązaniu znalezionych zostaje więcej maksimów lokalnych, a następnie spośród nich wybiera się najlepsze rozwiązanie.

Algorytm z listą tabu [50, 51] jest algorytmem opartym na przeszukiwaniu lokalnym. Idea polega na przeglądaniu rozwiązań w taki sposób, aby uniknąć powrotu do tych rozwiązań, które już były analizowane. W tym celu tworzy się strukturę nazywaną listą tabu, do której zapisuje się wszystkie zbadane rozwiązania. Algorytm sprawdza sąsiedztwo (lub jego fragment) aktualnie przeglądanego rozwiązania w poszukiwaniu sąsiada nieobecnego na liście tabu o najniższym koszcie, niezależnie od tego, czy jest on lepszy, czy gorszy od aktualnego. Dzięki temu uzyskuje się szybkie „wchodzenie" do minimum lokalnego i powolne „wychodzenie" [6].

Algorytm symulowanego wyżarzania [77, 16, 6] jest rozszerzeniem algorytmu poszukiwania lokalnego, a jego działanie opiera się na transformacjach. Dla optymalizowanego problemu jest wyznaczana wartość funkcji kosztu rozwiązania. Algorytm ten pozwala na wykonanie ruchu pogarszającego jakość rozwiązania, a prawdopodobieństwo przejścia do takiego stanu jest zależne od parametru zwanego temperaturą. Temperatura wraz z działaniem algo-

rytmu jest stopniowo obniżana (według przyjętego schematu schładzania), a, co za tym idzie, prawdopodobieństwo przejścia do rozwiązania gorszego maleje.

Idea algorytmu mrówkowego została zaczerpnięta z zachowania kolonii mrówek w przyrodzie [36, 25]. Każdy z owadów porusza się w sposób losowy szukając pożywienia i pozostawiając równocześnie za sobą ślad feromonowy. Intensywność tego śladu z czasem maleje (feromon paruje). Zatem, im dłuższa trasa dzieli mrowisko od pożywienia, tym dłużej feromon będzie parował, a siła jego oddziaływania będzie malała. Kolejne mrówki wybierają ścieżkę z prawdopodobieństwem wprost proporcjonalnym do wyczuwanego śladu feromonowego. Taka metoda doboru drogi zwiększa szansę wybrania najlepszej (najkrótszej) ścieżki, natomiast parowanie śladu wyeliminuje złe trasy.

Algorytm roju cząstek opiera się na analizie przestrzeni poszukiwań z wykorzystaniem prostych elementów – cząstek [75, 115]. Dla każdej z nich oblicza się wartość funkcji przystosowania w lokalizacji, w której aktualnie się znajduje. Następnie określa swój ruch uwzględniając aktualną i najlepszą z odwiedzonych lokalizacji, czyli taką, w której wartość funkcji przystosowania była najwyższa. Pod uwagę są również brane pozycje pewnej liczby członków roju. Kolejna iteracja ma miejsce, gdy wszystkie cząstki znajdą się w nowych miejscach. Rój jako całość zachowuje się jak stado ptaków szukających pożywienia. Celem cząstek roju jest znalezienie się możliwie jak najbliżej wartości optymalnej funkcji dopasowania.

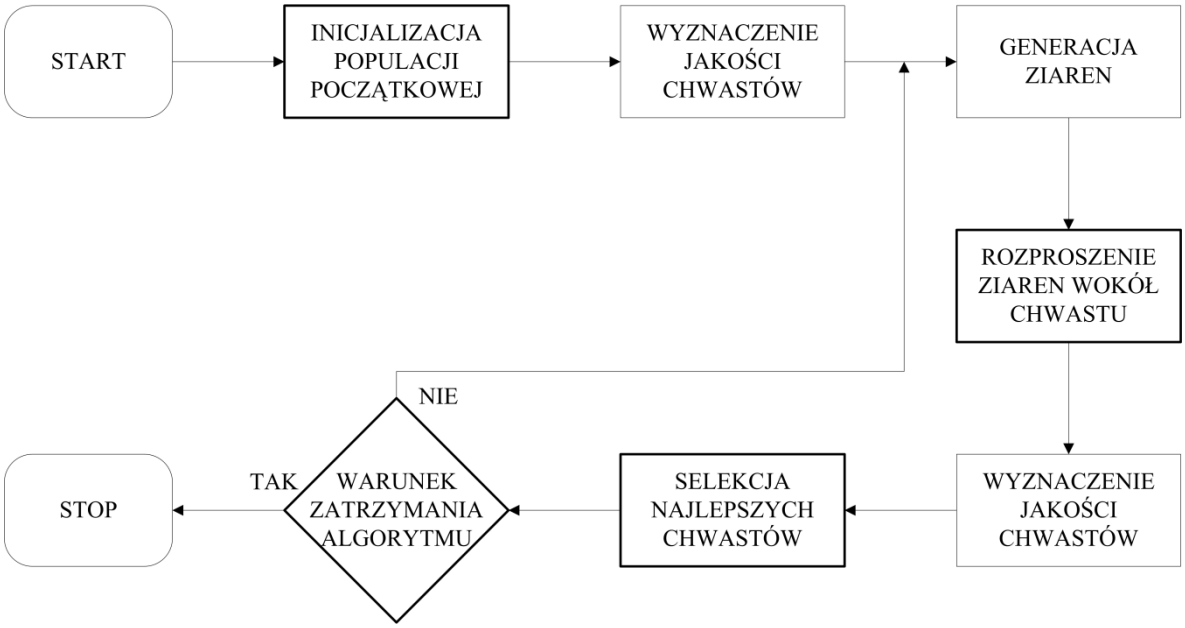
Wszystkie opisane algorytmy mogą być zastosowane do rozwiązywania większości zadań optymalizacji. Każdy z nich pozwala na znalezienie lepszych rozwiązań niż pozostałe metody w bardzo wąskich zastosowaniach. Jednak żaden z omówionych algorytmów nie wykazuje charakteru dominującego.

5. Zmodyfikowany algorytm IWO (expanded IWO, exIWO)

W rozdziale tym przedstawiono modyfikacje algorytmu IWO wraz z jego przystosowaniem do poszczególnych problemów optymalizacyjnych, co stanowi wkład własny autora. W pierwszej części rozdziału zawarto opis tych zmian algorytmu IWO, które mogą być zastosowane we wszystkich przypadkach. Dalej opisano szczegółowe modyfikacje zależne od zastosowania algorytmu.

5.1. Idea zmodyfikowanego algorytmu IWO

Na rys. 10 przedstawiono schemat blokowy algorytmu IWO z zaznaczeniem tych części, które zostały zmodyfikowane przez autora.



Rys. 10. Schemat blokowy algorytmu IWO z zaznaczeniem zmodyfikowanych jego części.

Inicjalizacja populacji początkowej

Podczas przeprowadzania wstępnych badań zauważono, iż nie warto losowo wybierać osobników do populacji początkowej. Wstępna selekcja rozwiązań pozwala na osiągnięcie znacznie lepszych wyników lub rezultatów o podobnej jakości w zdecydowanie krótszym czasie.

Przeprowadzenie wstępnego doboru osobników jest ściśle zależne od problemu optymalizacyjnego, który ma być rozwiązywany. Do tego celu może być wykorzystany np. algorytm zachłanny.

Zastosowanie algorytmu do inicjalizacji populacji początkowej zwiększa złożoność obliczeniową metody, co jednak jest rekompensowane przez uzyskiwanie lepszych rezultatów.

Rozproszenie ziaren wokół chwastu

Etap ten jest centralną częścią IWO. Dlatego autor upatruje największych możliwości w modyfikacji tej części algorytmu.

Opisywana faza składa się z dwóch kroków: wylosowania metody rozpraszania oraz jej przeprowadzenia. Metoda rozpraszania jest losowana dla każdego z ziaren z osobna. Wartości prawdopodobieństwa przypisane poszczególnym metodom: rozwiewaniu ($p_{rozwiew.}$), rozsiewaniu ($p_{rozsiew.}$) i staczaniu ($p_{stacz.}$) stanowią parametry algorytmu oraz sumują się do jedności:

$$p_{rozwiew.} + p_{rozsiew.} + p_{stacz.} = 1 \tag{9}$$

Wybór metody rozpraszania polega na wylosowaniu pewnej liczby r z rozkładem równomiernym w zakresie $\langle 0;1)$, a następnie na podstawie wylosowanej wartości i zależności:

$$metoda\ rozpraszania = \begin{cases} rozwiewanie & dla\ r < p_{rozwiew.} \\ rozsiewanie & dla\ p_{rozwiew.} \le r < p_{rozwiew.} + p_{rozsiew.} \\ staczanie & dla\ p_{rozwiew.} + p_{rozsiew.} \le r \end{cases} \tag{10}$$

dokonuje się przyporządkowania metody rozpraszania dla aktualnie rozpatrywanego ziarna.

Rozwiewanie jest metodą zbliżoną do znanej z oryginalnej wersji algorytmu IWO. Polega ona na wyznaczeniu kierunku i odległości miejsca upadku ziarna Z od chwastu macierzystego M. Odległość jest opisana rozkładem normalnym lub rozkładem t-Studenta. Rozkład ten charakteryzuje się większym prawdopodobieństwem wylosowania wartości bardziej oddalonej od wartości średniej niż w przypadku rozkładu normalnego. Cecha ta może zapobiegać utknięciu w minimum lokalnym. Typ rozkładu (normalny lub t-Studenta) jest parametrem metody. W przypadku rozkładu normalnego wartość oczekiwana wynosi 0, odchylenie standardowe zmienia się zgodnie ze wzorem (2), a cała metoda jest zgodna z pseudokodem przedstawionym na rys. 8.

Rozkład t-Studenta jest scharakteryzowany przez określoną liczbę stopni swobody p [42]. Wartość otrzymywana z generatora liczb pseudolosowych jest przemnażana przez współczynnik skalujący γ_{iter}. Współczynnik skalujący zmienia się wraz z każdą kolejną iteracją algorytmu, a formuła opisująca tę zmianę jest podobna do wzoru (2), określającego zmianę odchylenia standardowego w rozkładzie normalnym:

$$\gamma_{iter} = \left(\frac{iter_{max} - iter}{iter_{max}} \right)^m \left(\gamma_{init} - \gamma_{fin} \right) + \gamma_{fin} \tag{11}$$

gdzie: γ_{init} i γ_{fin} – początkowa i końcowa wartość współczynnika skalującego; $iter$ – aktualny numer iteracji algorytmu; $iter_{max}$ – łączna liczba iteracji algorytmu; m – współczynnik modulacji nieliniowej. Na rys. 11 przedstawiono pseudokod ilustrujący rozwiewanie z użyciem rozkładu t-Studenta.

```
Dla aktualnego ziarna Z wykonaj:
   Wylosuj odległość upadku ziarna Z od rośliny macierzystej M
      zgodnie z rozkładem t-Studenta t(p,γ_iter), gdzie γ_iter
      jest wyznaczana zgodnie ze wzorem 11.
   Wylosuj kierunek przesunięcia ziarna Z względem rośliny
      macierzystej M.
   Z = M
   Dokonaj transformacji Z zgodnie z wyznaczonymi odległością
      i kierunkiem.
   Zwróć Z.
```

Rys. 11. Rozwiewanie ziaren wokół chwastu z użyciem rozkładu *t*-Studenta.

Podobnie jak w przypadku IWO w wersji oryginalnej, gdy algorytm rozwiązuje problem optymalizacyjny o charakterze ciągłym, konstrukcja nowego osobnika (będąca odzwierciedleniem rozproszenia ziarna) polega na wykonaniu na kopii chwastu macierzystego transformacji, oddalającej potomka, w dowolnym losowym kierunku. Prawdopodobieństwo wyboru każdego z kierunków jest takie samo.

Dla problemów dyskretnych wykonuje się transformacje, w liczbie równej zdyskretyzowanej wartości wyznaczonej przez wybrany rozkład (normalny lub *t*-Studenta). Określenie losowego kierunku przesunięcia ziarna sprowadza się do zapewnienia równego prawdopodobieństwa każdej z transformacji generujących poprawne rozwiązanie.

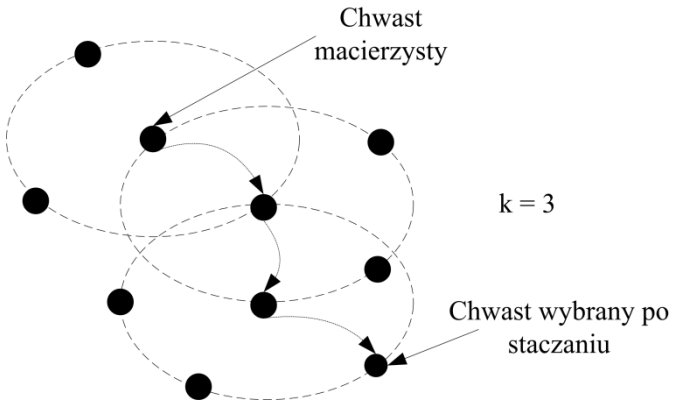
Rozsiewanie to losowe rozrzucanie ziaren w całej przestrzeni poszukiwań. Zakłada się rozkład równomierny we wszystkich wymiarach przestrzeni. Działanie to sprowadza się zatem do utworzenia w losowego osobnika. Metoda ta ma na celu zapobieganie utknięciom w minimach lokalnych.

Staczanie (rys. 12) opiera się na badaniu sąsiedztwa chwastu macierzystego *M*.

```
Dla aktualnego ziarna Z wykonaj:
   Wylosuj odległość upadku ziarna od rośliny macierzystej M
      zgodnie z rozkładem normalnym N(0,σ_iter) (σ_iter wyznacza
      się zgodnie ze wzorem 2) lub t-Studenta t(p,γ_iter) (γ_iter
      wyznacza się zgodnie ze wzorem 11).
      Z = M
   Dla m liczby ruchów wykonaj:
      Dla k liczby sąsiadów wykonaj:
         Wylosuj kierunek przesunięcia względem Z.
         Stwórz osobnika sąsiedniego Z' zgodnie z wyznaczonymi
            odległością i kierunkiem.
         Wyznacz jakość osobnika Z'.
      Z = najlepiej przystosowany osobnik sąsiedni Z'.
   Zwróć Z.
```

Rys. 12. Staczanie ziaren.

Przez osobniki sąsiednie rozumie się takie rozwiązania, które różnią się od rozwiązania pierwotnego dokładnie o jedną transformację. Spośród tak zdefiniowanego zbioru osobników sąsiednich w losowy sposób wybiera się określoną liczbę *k* chwastów *Z'*. W kolejnym kroku wyznaczana jest jakość każdego osobnika *Z'* należącego do tego podzbioru oraz wybiera się najlepsze z nich. Następnie przeprowadza się analizę jego sąsiedztwa, ponownie wybierając rozwiązanie o najlepszej jakości. Procedurę tę wykonuje się *m*-krotnie. Dla uproszczenia przyjęto, że liczba analizowanych sąsiadów *k* oraz liczba przejść do najlepszego z nich *m*, stanowiąca liczbę iteracji staczania, są sobie równe i stanowią parametr metody. Przykład staczania dla liczby sąsiadów i przejść *k* = 3 zobrazowano na rys. 13.



Rys. 13. Procedura wykonania metody staczania dla liczby sąsiadów i liczby przejść *k* = 3.

Kompletny krok rozpraszania ziaren wokół chwastu prezentuje w formie schematu blokowego rys. 14.

Selekcja najlepszych chwastów

Etap selekcji zawiera trzy metody do wyboru: wybór z całej populacji, wybór z osobników potomnych oraz wybór w ramach jednej rodziny. Wszystkie sposoby mają charakter deterministyczny.

Ogólną definicję selekcji można przedstawić w następujący sposób:

Niech:

$\mathbf{Z}_p$ *oznacza zbiór osobników w pokoleniu p o liczebności* $n_p = |\,\mathbf{Z}_p\,|$

$\mathbf{Z}_{p+1}$ *oznacza zbiór osobników w pokoleniu* $p+1$ *o liczebności* $n_{p+1} = |\,\mathbf{Z}_{p+1}\,|$

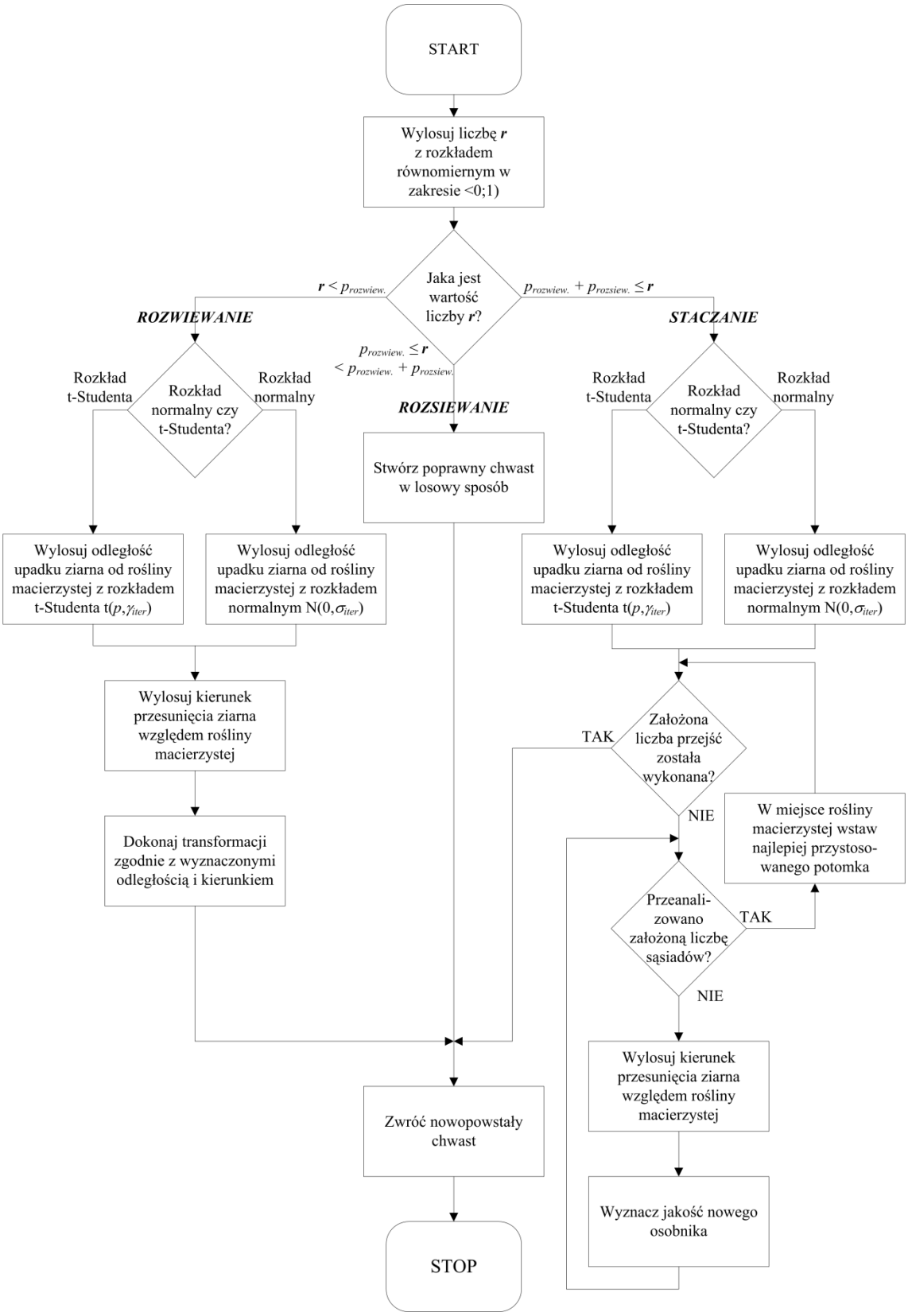
Założenie:

$n_p \geq n_{p+1}$

Definicja:

Selekcja jest funkcją f przekształcającą zbiór $\mathbf{Z}_p$ *w zbiór* $\mathbf{Z}_{p+1}$ *taką, że* $\mathbf{Z}_{p+1}$ *zawiera się w* $\mathbf{Z}_p$:

$$f : \mathbf{Z}_p \rightarrow \mathbf{Z}_{p+1} \Leftrightarrow \mathbf{Z}_p \supseteq \mathbf{Z}_{p+1} \tag{12}$$



Rys. 14. Schemat blokowy rozpraszania ziaren wokół chwastu w zmodyfikowanym algorytmie IWO.

Dla uproszczenia opisu prezentowanych w dalszej części niniejszego podrozdziału strategii selekcji wprowadzono następujące oznaczenia: $\mathbf{M}_p$ – zbiór osobników macierzystych w pokoleniu p oraz $\mathbf{P}_p$ – zbiór osobników potomnych stworzonych na podstawie osobników macierzystych $\mathbf{M}_p$ w pokoleniu p [103].

Selekcja z całej populacji jest techniką znaną z oryginalnej wersji algorytmu IWO i polega na wyborze najlepiej przystosowanych osobników ze zbioru złożonego ze wszystkich chwastów macierzystych i potomnych [103]. Opisaną metodę można zdefiniować następująco:

$$f : \mathbf{Z}_p \rightarrow \mathbf{Z}_{p+1} \Leftrightarrow \max_{n_{p+1}}\left(\mathbf{M}_p \cup \mathbf{P}_p\right) = \mathbf{M}_{p+1} \tag{13}$$

gdzie $\max_{n_{p+1}}\left(\mathbf{M}_p \cup \mathbf{P}_p\right)$ oznacza funkcję wybierającą n_{p+1} najlepiej przystosowanych osobników ze zbioru $\mathbf{M}_p \cup \mathbf{P}_p$.

Selekcja z osobników potomnych sprowadza się do wyboru rozwiązań ze zbioru złożonego tylko z chwastów potomnych:

$$f : \mathbf{Z}_p \rightarrow \mathbf{Z}_{p+1} \Leftrightarrow \max_{n_{p+1}}\left(\mathbf{P}_p\right) = \mathbf{M}_{p+1} \tag{14}$$

Nie ma niebezpieczeństwa redukcji liczności nowego pokolenia, ponieważ łączna liczba ziaren przewyższa liczbę chwastów macierzystych. Metoda ta nie gwarantuje, iż nowe pokolenie nie będzie gorsze od aktualnego. Można jednak założyć, że część chwastów potomnych przystosuje się lepiej niż odpowiadające im osobniki macierzyste. Efektem tego założenia jest fakt, że zastosowanie wyboru z osobników potomnych jest oszczędnością czasu, gdyż nie ma potrzeby przeglądania zbioru wszystkich ($\mathbf{M}_p$ + $\mathbf{P}_p$) rozwiązań, a jedynie jego części. Co więcej, takie podejście zmniejsza szanse na stagnację przy nieoptymalnym rozwiązaniu [103].

Obie metody, selekcji z całej populacji oraz selekcji z osobników potomnych, są technikami wyboru o charakterze globalnym. Metoda selekcji w ramach jednej rodziny koncentruje się na wyborze lokalnym osobników [135]. Strategia ta wybiera do kolejnego pokolenia jeden chwast ze zbioru złożonego z jednej rośliny macierzystej i jej bezpośrednich potomków. Opisywana selekcja przeprowadzana jest dla każdej rodziny oddzielnie, zapewniając tym samym stałą liczebność nowej populacji chwastów:

$$f : \mathbf{Z}_p \rightarrow \mathbf{Z}_{p+1} \Leftrightarrow \underset{C \in \mathbf{M}_p}{\forall} \max_{1}\left(C + \mathbf{P}_p^C\right) = \mathbf{M}_{p+1} \tag{15}$$

gdzie: $\left(C + \mathbf{P}_p^C\right)$ jest zbiorem osobników złożonym z rośliny macierzystej C oraz jej bezpośrednich potomków w pokoleniu p ($\mathbf{P}_p^C$), natomiast $\max_{1}\left(C + \mathbf{P}_p^C\right)$ oznacza funkcję wybierającą jednego najlepiej przystosowanego osobnika ze zbioru $C + \mathbf{P}_p^C$.

Warunek zatrzymania algorytmu

Tradycyjnym warunkiem zatrzymania algorytmu jest z góry założona liczba iteracji. Element ten został wzbogacony o dodatkowe kryterium, jakim jest czas optymalizacji. Wprowadzono go w celu lepszej możliwości porównania różnych metod optymalizacyjnych, gdyż w związku z różnym czasem realizacji algorytmów bardziej miarodajnym kryterium zatrzymania algorytmu wydaje się być z góry założony interwał czasowy (zakładając wykonanie wszystkich eksperymentów na tej samej maszynie obliczeniowej).

Czas, jaki upłynął od początku działania algorytmu IWO, jest sprawdzany tylko na koniec iteracji. Pozwala to na dokończenie obliczeń związanych z danym pokoleniem chwastów.

Tabela 3

Zestawienie dodatkowych parametrów algorytmu IWO

Nazwa parametru	**Typ**
typ rozkładu	normalny lub *t*-Studenta
typ selekcji	wybór z całej populacji, wybór z osobników potomnych lub wybór w ramach jednej rodziny
warunek zatrzymania algorytmu	liczba iteracji lub czas
czas przeznaczony na wykonanie algorytmu (gdy warunkiem zatrzymania jest czas)	liczba rzeczywista dodatnia określająca czas w sekundach
liczba stopni swobody (dla rozkładu *t*-Studenta)	liczba naturalna
początkowa wartość współczynnika skalującego γ_{init} (dla rozkładu *t*-Studenta)	liczba rzeczywista dodatnia
końcowa wartość współczynnika skalującego γ_{fin} (dla rozkładu *t*-Studenta)	liczba rzeczywista dodatnia mniejsza od początkowej wartości współczynnika skalującego
liczba przejść k (dla staczania)	liczba naturalna
prawdopodobieństwo wylosowania metody rozwiewania	liczba rzeczywista z przedziału <0; 1>, prawdopodobieństwa rozwiewania, rozsiewania i staczania sumują się do 1
prawdopodobieństwo wylosowania metody rozsiewania	liczba rzeczywista z przedziału <0; 1>, prawdopodobieństwa rozwiewania, rozsiewania i staczania sumują się do 1
prawdopodobieństwo wylosowania metody staczania	liczba rzeczywista z przedziału <0; 1>, prawdopodobieństwa rozwiewania, rozsiewania i staczania sumują się do 1

Przy wyborze czasu optymalizacji jako warunku zatrzymania niezbędne jest szacowanie maksymalnej liczby iteracji na potrzeby obliczania opisanych wcześniej: odchylenia standardowego odległości pomiędzy chwastem macierzystym a potomnym (rozkład normalny) lub współczynnika skalującego (rozkład *t*-Studenta). Naturalną metodą jest sprawdzenie liczby iteracji wykonanych od początku działania algorytmu, a następnie przeskalowanie tej liczby na cały zadany czas.

W tabeli 3 zawarto zestawienie dodatkowych parametrów algorytmu IWO.

5.2. Analiza złożoności obliczeniowej

Złożoność obliczeniowa exIWO zostanie wyznaczona w podobny sposób jak w przypadku wersji oryginalnej i dotyczy jedynie jego idei (nie jest związana z żadnym zadaniem optymalizacji), natomiast w rozdziale 5.3.4 zawarto próbę oszacowania złożoności dla problemu wyznaczania kolejności złączeń w realizacji zapytań w relacyjnej bazie danych.

Modyfikacje algorytmu zostały przeprowadzone przede wszystkim w zakresie rozpraszania ziaren wokół rośliny macierzystej oraz selekcji chwastów. Faza wstępna, obejmująca stworzenie roślin uczestniczących w pierwszym pokoleniu oraz ocenę ich jakości, będzie miała złożoność identyczną jak w przypadku oryginalnego IWO (wzór 3). Do inicjalizacji osobników można wykorzystać również algorytm, który pozwoli na preselekcję rozwiązań. Jednak ta metoda jest ściśle zależna od rozwiązywanego problemu i nie będzie rozpatrywana.

Kolejny etap algorytmu, generacja ziaren, jest identyczny z oryginalnym. Zmiany są zauważalne w fazie rozpraszania ziaren i zależą od przyjętej metody. Każda ze strategii rozpraszania charakteryzuje się inną złożonością. Rozwiewanie jest techniką przedstawioną przez autorów oryginalnego algorytmu IWO, a jej analiza złożoności została przedstawiona w rozdziale 4.2.2. Rozsiewanie jest równoznaczne ze stworzeniem nowego osobnika, zatem koszt realizacji tej metody jest liniowo zależny od liczby stworzonych w ten sposób chwastów.

Największą złożonością charakteryzuje się staczanie, gdyż opiera się na wielokrotnym generowaniu osobników oraz na ocenie ich jakości. Zgodnie z pseudokodem przedstawionym na rys. 12, złożoność omawianej strategii jest iloczynem liczby analizowanych roślin sąsiednich w każdym kroku metody oraz liczby przejść do najlepszego z badanych sąsiadów. Mając na uwadze uproszczenie poczynione w podrozdziale 5.1, mówiące o tym, że liczba chwastów sąsiednich oraz liczba przejść są sobie równe, złożoność obliczeniowa opisywanej techniki ma charakter kwadratowy O[liczba_przejść^2]. Z tej też przyczyny w realnych zastosowaniach liczba ta jest bardzo mała i nie przekracza wartości 5.

Przyjmując wariant pesymistyczny, czyli sytuację, w której wszystkie nowe rośliny są tworzone przy użyciu strategii staczania, otrzymuje się następującą zależność:

$$T_{ziaren} = O[\text{liczba\_chwastów (wyznaczenie liczby ziaren)} + \text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_przejść}^2 \text{ (rozproszenie)} + \text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_przejść}^2 \text{ (ocena jakości)}] = O[\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_przejść}^2] \quad (16)$$

Złożoność obliczeniowa fazy selekcji również zależy od przyjętej metody. Wybór z całej populacji jest techniką znaną z oryginalnej wersji IWO, a jej złożoność jest opisana zależnością (5). Wybór z osobników potomnych działa na bardzo podobnej zasadzie jak wybór z całej populacji. Różnica polega na tym, że do struktury drzewiastej nie dodaje się osobników macierzystych, co pozwala na częściowe obniżenie złożoności obliczeniowej:

$$T_{\text{selekcji z potomków}} = O[\log(\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren})] \quad (17)$$

Metoda wyboru w ramach jednej rodziny skupia się na przeszukiwaniu lokalnym i polega na dodawaniu do struktury drzewiastej osobnika macierzystego wraz z jego wszystkimi bezpośrednimi chwastami potomnymi, a następnie wyborze jednego z osobników. Strategia ta jest powtarzana dla każdej rośliny (wzór 18).

$$T_{\text{selekcji z rodziny}} = O[\text{liczba\_chwastów} \cdot \log(1 + \text{maks\_liczba\_ziaren}) + \text{liczba\_chwastów (pobranie ze struktury)}] = O[\text{liczba\_chwastów} \cdot (\log(1 + \text{maks\_liczba\_ziaren}) + 1)] \quad (18)$$

Korzystając z zależności 6 i podstawiając wzory 3, 16 i 18 otrzymuje się:

$$T_{exIWO} = O[\text{liczba\_chwastów} + \text{liczba\_iteracji} \cdot (\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_ przejść}^2 + \text{liczba\_chwastów} \cdot (\log(1 + \text{maks\_liczba\_ziaren}) + 1))] \quad (19)$$

Zakładając, że w praktycznych zastosowaniach maksymalna liczba ziaren jest mała, można pominąć wyrażenie z logarytmem. W ten sposób otrzymuje się końcową złożoność obliczeniową zmodyfikowanego algorytmu IWO rzędu:

$$T_{IWO} = O[\text{liczba\_chwastów} \cdot \text{liczba\_iteracji} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_ przejść}^2] \quad (20)$$

Wyznaczona dla exIWO złożoność obliczeniowa jest większa niż złożoność algorytmu w wersji podstawowej. Mimo to zmodyfikowany algorytm IWO (podobnie jak klasyczne IWO) należy do metod o złożoności wielomianowej, co nie dyskwalifikuje go z dalszych rozważań.

5.3. Modyfikacje IWO opracowane na potrzeby problemów optymalizacyjnych

Do pełnego opisu algorytmu niezbędne jest zdefiniowanie jego części ściśle zależnych od rozwiązywanego problemu optymalizacyjnego. Konieczne jest przedstawienie reprezentacji pojedynczego rozwiązania zadania optymalizacji. Wymagane jest także zdefiniowanie metod inicjalizacji pierwszej populacji oraz transformacji pojedynczego osobnika. Ostatnim waż-

nym elementem jest zaprojektowanie modelu obliczeniowego oraz opartej na tym modelu funkcji kosztu, służącej do porównywania jakości otrzymywanych rozwiązań.

5.3.1. Znajdowanie minimum funkcji wielowymiarowych

Potencjalnymi rozwiązaniami zagadnienia znajdowania minimum funkcji *n*-wymiarowej (funkcji *n* zmiennych) są punkty w przestrzeni, dla których ta funkcja jest określona. Do jednoznacznego zdefiniowania takiego punktu należy znać jego współrzędne w każdym z wymiarów. Zatem chwast reprezentujący pojedyncze rozwiązanie jest *n*-elementową tablicą zawierającą współrzędne punktu w *n*-wymiarach.

Najprostszą formą wyboru osobników do populacji początkowej jest wybór losowy. Jedynym ograniczeniem staje się zakres możliwych wartości początkowych, w jakich ma się mieścić generowany chwast.

Transformacja osobnika, wykorzystywana w operatorach rozwiewania i staczania (rozsiewanie nie wymaga stosowania transformacji), polega na umiejscowieniu nowego chwastu w pewnej odległości od rośliny macierzystej. Odległość ta obliczana jest zgodnie z wybranym rozkładem (rozkładem normalnym lub *t*-Studenta). Ostatnim elementem jest wyznaczenie kierunku upadku ziarna. Odbywa się to w trzech etapach:

1. Dla każdego wymiaru *i* losuje się liczbę x_i o rozkładzie jednostajnym z zakresu <0; 1).
2. Wyznacza się składową dla każdego z wymiarów według następującego wzoru:

$$y_j = x_j \cdot dist \cdot \sqrt{\frac{1}{\sum_{i=1}^{n} x_i^2}}, \tag{21}$$

   gdzie: y_j – składowa dla wymiaru *j*; x_i (x_j) – liczba wylosowana dla *i*-tego (*j*-tego) wymiaru; *n* – liczba wymiarów; *dist* – odległość pomiędzy roślinami wyznaczona przy użyciu jednego z rozkładów. Formuła (21) powstała w wyniku przekształceń podstawowych reguł geometrii analitycznej.
3. Zwrot składowych każdego z wymiarów jest losowany z prawdopodobieństwem 0,5.

Model obliczeniowy w przypadku znajdowania minimum funkcji wielowymiarowej jest bardzo prosty i sprowadza się do wyznaczenia wartości tej funkcji w określonym punkcie.

5.3.2. Problem komiwojażera

Użycie algorytmu IWO do problemu komiwojażera wymaga dokonania wyboru postaci osobnika (chwastu) jako trasy stanowiącej pojedyncze rozwiązanie problemu. W literaturze proponuje się reprezentacje wektorowe: ścieżkową, porządkową i w postaci listy sąsiedztwa (przyległościową) oraz reprezentacje macierzowe (m.in. binarną macierz pierwszeństwa [103]). Reprezentacja ścieżkowa, zgodnie z którą kolejne pozycje na liście oznaczają kolejno

odwiedzane miasta (np. dla 9 miast jedna z możliwych tras to 2 3 5 7 1 9 4 8 6), wydaje się być najbardziej naturalną. Z tego też względu zdecydowano się na wybór tej reprezentacji.

Inicjalizacja pierwszej populacji chwastów odbywa się przy użyciu algorytmu zachłannego. Poszczególne rozwiązania konstruuje się w następujący sposób:

1. Losuje się miasto, które staje się miastem startowym.
2. Z pozostałych miast wybiera się to, które leży najbliżej ostatnio wybranego.
3. Miejsce wybrane w punkcie 2 jest usuwane ze zbioru miast nieodwiedzonych.
4. Punkty 2 i 3 powtarzane są do chwili, w której zbiór miast nieodwiedzonych jest pusty.

W pracy [135] opisano rozwiązanie problemu komiwojażera za pomocą algorytmu ewolucyjnego przy użyciu operatora o nazwie *inver-over*. Operator ten łączy w sobie cechy mutacji i krzyżowania. Rezultaty tych badań pozwoliły określić zaproponowany algorytm jako „prawdopodobnie najszybszy dotychczas opracowany algorytm ewolucyjny dla problemu komiwojażera" [103]. Z tego względu zdecydowano się na wykorzystanie operatora *inver-over* w algorytmie IWO. Prócz tego zastosowano dwa tradycyjne operatory, którymi są zamiana oraz inwersja [72, 145, 40].

Zamiana polega na prostej wymianie miejscami dwóch wylosowanych miast.

W przypadku inwersji losuje się dwa miasta wyznaczające fragment trasy. Następnie trasa jest rozcinana w punktach umiejscowionych bezpośrednio za wybranymi miastami (rys. 5a). Kolejność miast w wyciętym fragmencie jest odwracana, a odwrócony fragment wstawia się w to samo miejsce (rys. 5b).

a) 2 3 | 5 7 1 9 4 | 8 6 b) 2 3 4 9 1 7 5 8 6

Rys. 15. Operator inwersji – przykład: a) osobnik z zaznaczonymi miejscami przecięcia; b) wynik inwersji – odwrócenie kolejności miast.

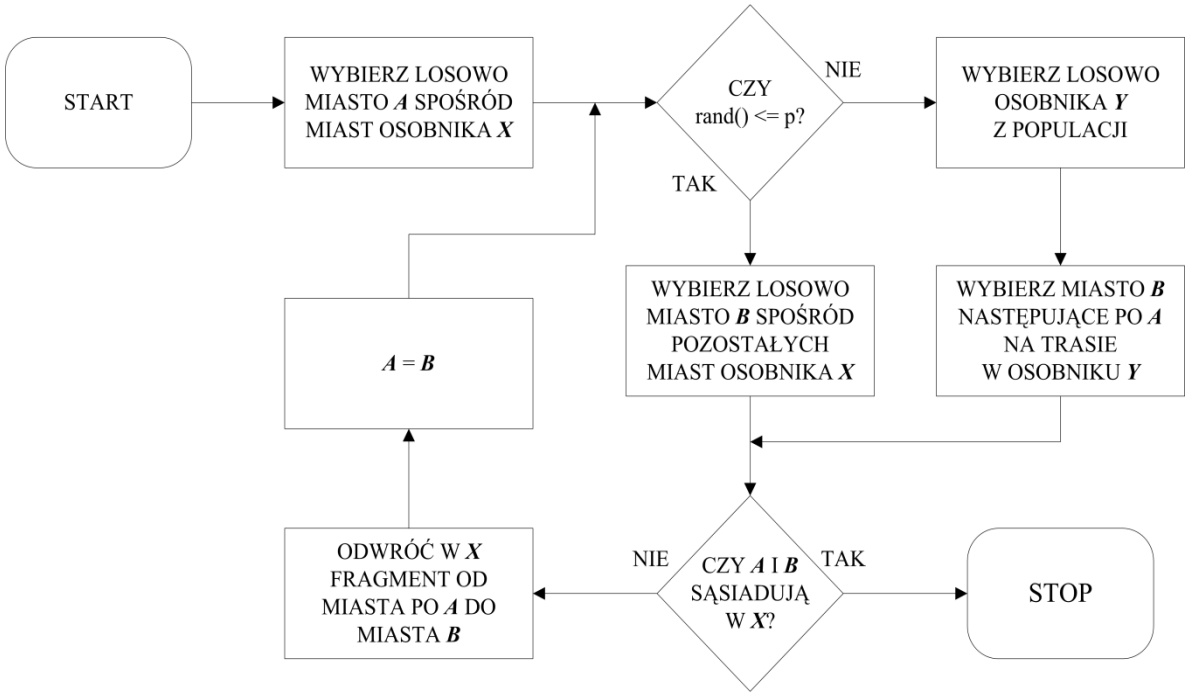
Operator *inver-over* [103, 135, 104] stanowi rozbudowany wariant operatora inwersji. Miasto wyznaczające pierwszy koniec odwracanego fragmentu trasy jest losowane z osobnika poddawanego działaniu operatora. Drugi koniec fragmentu może zostać wybrany zarówno z tego samego osobnika (co odpowiada inwersji), jak i z innego wylosowanego osobnika populacji. Prawdopodobieństwo pierwszej z wymienionych sytuacji jest jednak małe. Dzięki temu operator ma po części charakter krzyżowania, gdyż w przypadku użycia innego osobnika co najmniej dwa miasta reprezentowanej przez niego trasy trafią do osobnika potomnego. Wyjaśnienie tego stwierdzenia przynosi następujący przykład zaczerpnięty z pracy jednego z autorów operatora *inver-over* [103].

Niech rozpatrywany osobnik ma postać (2 3 9 4 1 5 8 6 7), a miastem wskazującym pierwszy koniec jest 3. W przypadku klasycznej inwersji miasto wskazujące drugi koniec fragmentu wybierane jest z tego samego osobnika (niech będzie to 8). Wobec czego osobnik z punktami rozcięcia ma postać (2 3 | 9 4 1 5 8 | 6 7), a nowy osobnik wygląda następująco (2 3 8 5 1 4 9 6 7) (odwrócony fragment podkreślono).

Natomiast, jeśli sięgnie się do innego osobnika, np. o postaci (1 6 4 3 5 7 9 2 8), wybierane jest miasto znajdujące się bezpośrednio za miastem 3 (w tym wypadku będzie to 5). Działanie to powoduje, że osobnik poddawany działaniu operatora będzie rozcinany w następujący sposób: (2 3 | 9 4 1 5 | 8 6 7). Doprowadzi to do powstania osobnika potomnego o postaci (2 3 5 1 4 9 8 6 7). Od innego osobnika pochodzi więc fragment „3 5”.

Należy zwrócić uwagę na fakt, że operator *inver-over* jest na ogół stosowany wielokrotnie dla tego samego osobnika.

Rysunek 16 przedstawia schemat blokowy operatora *inver-over* (rand() – funkcja generująca wartości pseudolosowe o rozkładzie jednostajnym z przedziału <0; 1); *p* – prawdopodobieństwo zdarzenia, że obydwa końce fragmentu będą pochodziły z tego samego osobnika).



Rys. 16. Schemat blokowy operatora *inver-over*.

Ostatni element algorytmu, który wymaga sprecyzowania, to postać funkcji przystosowania osobnika. Dla rozważanego problemu optymalizacyjnego jest ona łączną długością trasy wyznaczonej przez kolejne numery miast zapisanych na kolejnych pozycjach chwastu.

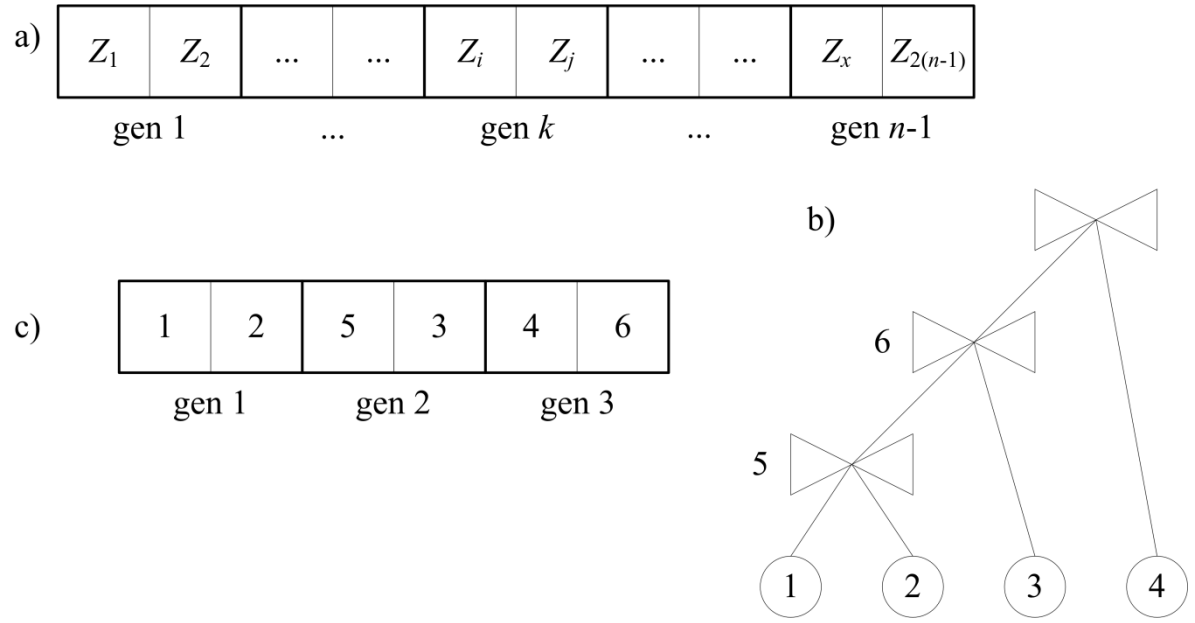
5.3.3. Określanie kolejności złączeń w realizacji zapytań w scentralizowanych bazach danych

Podrozdział ten opisuje modyfikacje algorytmu IWO wprowadzone na potrzeby rozwiązywania problemu określania kolejności złączeń w realizacji zapytań w scentralizowanych bazach danych. Należy mieć na uwadze fakt, iż przy niewielkich zmianach jest możliwe przeprowadzenie optymalizacji dla baz danych rozproszonych. Tematyka ta jest jednym z potencjalnych kierunków rozwoju niniejszej pracy.

We wszystkich rozważaniach dotyczących problemu określania kolejności złączeń w realizacji zapytań kierowanych do baz danych zakłada się, że dane, które podlegają analizie, są danymi wstępnie przetworzonymi. Oznacza to, że dysponuje się zbiorami rekordów, powstałymi w wyniku zastosowania na tabelach bazodanowych operacji selekcji i projekcji.

Pojedynczy osobnik populacji chwastów stanowi odpowiednik pojedynczego rozwiązania problemu, którym jest określony porządek realizacji złączeń zbiorów rekordów. Jego reprezentację przedstawia rys. 17a.

Symbole Z_i, Z_j oznaczają numery zbiorów rekordów, które zostaną złączone w k-tym kroku. Dwójka symboli (Z_i, Z_j) będzie nazywana genem. Liczba genów w osobniku jest równa n–1, gdzie n jest liczbą zbiorów pierwotnych wymienionych w zapytaniu. Przez zbiór pierwotny rozumie się taki zbiór rekordów, który powstał w wyniku zastosowania operacji selekcji i projekcji na tabelach bazy danych. Zbiory pierwotne otrzymują kolejne numery od 1 do n. Układ genów odpowiada kolejności wykonywania poszczególnych złączeń. Wynik złączenia opisanego pierwszym genem będzie zbiorem rekordów o numerze n+1, a kolejne zbiory (wyniki pośrednie, zbiory pośrednie) otrzymają numery będące kolejnymi liczbami naturalnymi (n+2, n+3, …). Ponieważ wyniki pośrednie są wykorzystywane w dalszych złączeniach, będą występować w kolejnych genach. Jednakże musi być spełniony następujący warunek: gen zawierający numer zbioru stanowiącego wynik pośredni może w osobniku wystąpić dopiero po genie, który reprezentuje operację złączenia generującą ten wynik pośredni.



Rys. 17. Reprezentacja pojedynczego rozwiązania: a) symboliczna reprezentacja osobnika; b) przykładowe drzewo realizacji złączeń; c) osobnik odpowiadający drzewu z punktu b).

Przykład przedstawiony na rysunku 17b i 17c można opisać w następująco:

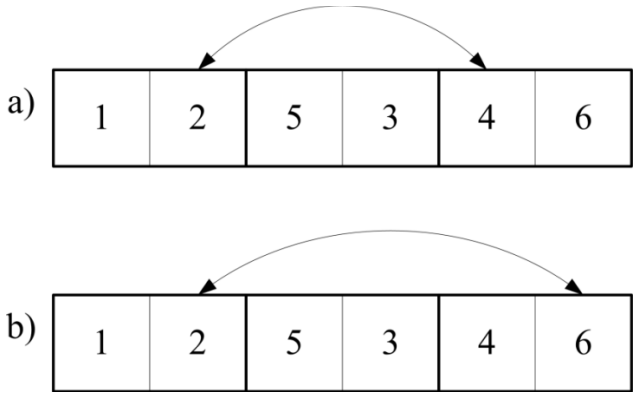
- Zapytanie jest oparte na czterech zbiorach pierwotnych (pochodzących ze wstępnego przetworzenia tabel bazy danych) o numerach 1, 2, 3 i 4.

- W pierwszym kroku, reprezentowanym przez gen 1, zostaną złączone zbiory pierwotne o numerach 1 i 2. Wynikiem tego złączenia jest zbiór pośredni o numerze 5 (jest to kolejna liczba naturalna większa od największego numeru zbioru pierwotnego).
- Kolejnym etapem jest wykonanie złączenia zbioru pośredniego o numerze 5 ze zbiorem pierwotnym o numerze 3 (gen 2). W wyniku tej operacji powstanie zbiór pośredni o numerze 6.
- Trzecie złączenie (gen 3) zostanie przeprowadzone na zbiorze pierwotnym o numerze 4 oraz zbiorze pośrednim numer 6. Zbiór powstały w wyniku tego złączenia jest zarazem zbiorem końcowym zapytania skierowanego do bazy danych.

Konstruowanie opisanych osobników do populacji początkowej polega na określeniu w każdym z genów pary zbiorów rekordów, jakie mają być złączone. Należy pamiętać, że numery zbiorów pośrednich mogą wystąpić dopiero po genie reprezentującym złączenie, w którym ten zbiór powstaje. Zbiory rekordów dobiera się zatem do poszczególnych genów w taki sposób, aby nie wystąpiła sytuacja nieprawidłowa.

Transformacja rośliny macierzystej, prowadząca do powstania chwastu potomnego polega na zmianie jednego z pary łączonych ze sobą zbiorów. Mechanizm wykorzystany w algorytmie IWO odpowiada operatorowi genetycznemu mutacji, zastosowanemu w algorytmie ewolucyjnym opisanym w pracach [104, 93, 6].

Transformacja numerów zbiorów jest realizowana przez zamianę dwóch wylosowanych zbiorów należących do różnych genów. W ten sposób nie dojdzie do niedopuszczalnego wielokrotnego wystąpienia tego samego numeru zbioru w pojedynczym osobniku. Ponadto sposób przeprowadzenia transformacji zapewnia, że do genu trafia taki numer zbioru, dla którego gen, reprezentujący operację złączenia tworzącą ten zbiór, zajmuje w osobniku wcześniejszą pozycję niż gen poddawany transformacji. Na rys. 18 przedstawiono przykładowe realizacje transformacji poprawnej i niepoprawnej (zbiór o numerze 6 jest wynikiem drugiego złączenia, więc nie może znaleźć się przed genem reprezentującym to złączenie).



Rys. 18. Przykład transformacji: a) poprawnej, b) niepoprawnej.

Algorytm IWO wymaga do oceny jakości rozwiązań zdefiniowania funkcji celu, którą w rozważanym problemie jest szacunkowy koszt (czas) realizacji ciągu złączeń. W prowadzonych badaniach poddano modyfikacji model zastosowany w pracach [130, 129, 123, 71]. Wymaga on zdefiniowania formuł umożliwiających oszacowanie liczebności zbioru wynikowego operacji złączenia i rozmiaru rekordu w zbiorze wynikowym oraz kosztu operacji złączenia.

Rozmiar rekordu w zbiorze wynikowym złączenia wyznaczany jest zgodnie z następującą formułą:

$$dr(X \triangleright\triangleleft Y) = \eta \cdot (dr(X) + dr(Y)) \tag{22}$$

gdzie: $dr(X)$, $dr(Y)$ – średnie długości rekordów składowych; η – współczynnik redukcji długości wiersza. Jeśli dla operacji złączenia dane jest wyrażenie łączące, to wartość współczynnika η pochodzi z przedziału (0, 1>, przy czym wartość 1 oznacza brak redukcji długości wiersza. W przeciwnym przypadku złączenie realizowane jest jako iloczyn kartezjański obu zbiorów, a współczynnik redukcji przyjmuje wartość 1. Długość rekordu jest ważna ze względu na konieczność odczytania danych z dysku twardego podczas wykonania złączenia. W niniejszych rozważaniach zakłada się, że całkowity rozmiar zbioru wynikowego znacznie przekracza dostępną pamięć operacyjną. Stąd konieczność przechowywania danych na dysku.

Sposób wyznaczenia liczebności zbioru wynikowego operacji złączenia wymaga rozważenia następujących przypadków:

- jeśli nie podano wyrażenia łączącego, to liczebność zbioru wynikowego jako iloczynu kartezjańskiego obu zbiorów jest równa iloczynowi liczebności łączonych zbiorów $lr(X)$ oraz $lr(Y)$:

$$lr(X \triangleright\triangleleft Y) = lr(X) \cdot lr(Y) \tag{23}$$

- jeśli między wartościami atrybutów obu zbiorów występujących w wyrażeniu łączącym występuje zależność polegająca na tym, że pojedynczej wartości atrybutu zbioru o niewiększej liczebności odpowiada co najwyżej jedna wartość atrybutu drugiego zbioru, to poprawnym oszacowaniem liczebności zbioru wynikowego jest mniejsza spośród liczebności obu zbiorów (złączenie typu 1:1, przykładem ilustrującym tę zależność może być wyrażenie łączące, w którym występują klucze główne obu zbiorów):

$$lr(X \triangleright\triangleleft Y) = \min(lr(X), lr(Y)) \tag{24}$$

- jeśli między wartościami atrybutów obu zbiorów występujących w wyrażeniu łączącym występuje zależność polegająca na tym, że pojedynczej wartości atrybutu zbioru X odpowiada wiele wartości atrybutu zbioru Y i równocześnie pojedynczej wartości atrybutu zbioru Y odpowiada co najwyżej jedna wartość atrybutu zbioru X, to poprawnym oszacowaniem liczebności zbioru wynikowego jest liczebność zbioru Y (złączenie typu 1:n,

przykładem ilustrującym tę zależność może być wyrażenie łączące, w którym występuje klucz główny zbioru *X* i odpowiadający mu klucz obcy zbioru *Y*):

$$lr(X \rhd\lhd Y) = lr(Y) \tag{25}$$

- w pozostałych przypadkach stosowany jest następujący wzór:

$$lr(X \rhd\lhd Y) = \frac{1}{sel} \cdot \min(lr(X), lr(Y)) \tag{26}$$

gdzie symbol *sel* reprezentuje współczynnik unikalności wartości atrybutów występujących w wyrażeniu łączącym. Definiowany jest jako iloraz liczby wartości unikalnych atrybutu przez liczebność zbioru. Dla uproszczenia przyjęto jednakową wartość współczynnika *sel* dla wszystkich zbiorów.

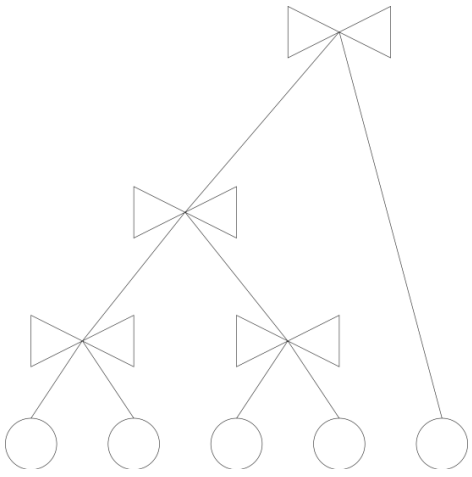
Koszt wykonania operacji złączenia K_Z zostanie oszacowany według formuły opartej na pracach [21, 130]. Wzór ten został wzbogacony przez autora o człon określający narzut czasowy związany z odczytem danych z dysku twardego komputera wykonującego złączenie:

$$K_Z = \frac{lr(X) + lr(Y) + lr(X \rhd\lhd Y)}{v} + \frac{lr(X) \cdot dr(X) + lr(Y) \cdot dr(Y)}{d} \tag{27}$$

gdzie: $lr(X)$, $lr(Y)$ – liczba rekordów w zbiorach łączonych; $lr(X \rhd\lhd Y)$ – liczba rekordów zbioru wynikowego; v – moc obliczeniowa węzła (wyrażona w liczbie przetwarzanych rekordów w jednostce czasu); $dr(X)$, $dr(Y)$ – średnie długości rekordów zbiorów łączonych; d – szybkość odczytu danych z dysku (wyrażona w bajtach odczytanych w jednostce czasu).

Wzór (27) składa się z dwóch członów. Pierwszy charakteryzuje szacowany koszt realizacji złączenia przez procesor węzła, natomiast drugi określa wspomniany narzut czasowy związany z odczytem danych z dysku (dotyczy to tylko zbiorów danych o bardzo dużym rozmiarze).

Do obliczenia wartości funkcji celu dla pojedynczego rozwiązania, czyli określenia kolejności złączeń w realizacji zapytania skierowanego do bazy danych, wykorzystano strukturę drzewiastą (rys. 19).



Rys. 19. Przykładowa struktura drzewiasta służąca do wyznaczenia wartości funkcji celu.

Liście struktury zawierają charakterystykę zbiorów pierwotnych, natomiast w wierzchołkach zapisane są informacje o poszczególnych operacjach złączenia i ich rezultatach. Pojedyncza krawędź łączy liść lub wierzchołek opisujący powstanie danego zbioru z wierzchołkiem przyporządkowanym złączeniu z udziałem tego zbioru. Każdy wierzchołek jest zwieńczeniem poddrzewa obejmującego określony fragment rozwiązania i zawiera informacje pozwalające na obliczenie wartości funkcji celu dla tego fragmentu. Przez stopniowe kumulowanie wartości funkcji celu kolejne elementy struktury są wypełniane w jednym przebiegu – od liści w kierunku korzenia.

Wartość funkcji przystosowania, odgrywająca w przypadku algorytmu IWO istotną rolę w określeniu liczby nasion rozsiewanych przez daną roślinę, obliczana jest jako odwrotność funkcji celu.

Jak wspomniano w podrozdziale 3.1, jedną z form reprezentacji zapytania jest graf złączeń. Jego wierzchołki odpowiadają zbiorom pierwotnym, a każda z krawędzi wyraża fakt istnienia między nimi wyrażenia łączącego. Jest to kryterium dopasowania rekordów tych zbiorów do siebie. Do charakterystycznych kształtów grafu złączeń zalicza się (rys. 3):

- graf gwiazdy, w którym jeden ze zbiorów rekordów występuje w każdym wyrażeniu łączącym,
- graf łańcuchowy, gdzie dwa zbiory rekordów występują tylko jednokrotnie w wyrażeniach łączących, natomiast pozostałe zbiory – dwukrotnie.

Podana charakterystyka stanowi podstawę do zaproponowania formuły pozwalającej na identyfikację kształtu grafu liczącego n wierzchołków, gdzie z każdym wierzchołkiem i związanych jest k_i krawędzi. Formuła ma następującą postać:

$$L = \frac{1}{n} \cdot \sum_{i=1}^{n} k_i \tag{28}$$

Dla grafu o charakterze gwiazdy liczba L wynosi $L = \frac{n^2 - 2n + 2}{n^2 - n}$, zaś dla grafu łańcuchowego $L = \frac{n+2}{2n}$. Pozostałe kształty określa się mianem grafów nieregularnych. Dzięki łatwej identyfikacji kształtu grafu złączeń istnieje możliwość lepszego doboru współczynników algorytmu.

5.3.4. Analiza złożoności obliczeniowej dla problemu określania kolejności złączeń w realizacji zapytań w bazach danych

Niniejszy podrozdział zawiera próbę oszacowania złożoności obliczeniowych dla dwóch wersji algorytmu IWO, oryginalnej i zmodyfikowanej, dla problemu wyznaczania kolejności złączeń w realizacji zapytań relacyjnej bazy danych.

Złożoności zostaną wyznaczone, podobnie jak w podrozdziałach 4.2.2 i 5.2, na podstawie kolejno wykonywanych kroków algorytmu. Jako operacje dominujące przyjęto działania podejmowane w poszczególnych pętlach metod.

Złożoności algorytmu IWO, zarówno w wersji oryginalnej, jak i zmodyfikowanej, można przedstawić za pomocą ogólnej formuły (6):

$$T_{IWO} = O[T_{init} + \text{liczba\_iteracji} \cdot (T_{ziaren} + T_{selekcji})].$$

Faza wstępna (T_{init}) składa się z inicjalizacji pierwszego pokolenia chwastów oraz oceny ich jakości. Jak nadmieniono w podrozdziale 4.2.2, złożoność tych czynności jest liniowo zależna od liczby tworzonych osobników. Oznaczając złożoność wygenerowania pojedynczej rośliny jako $T_{generacji}$ oraz złożoność oceny jakości chwastu jako T_{oceny} otrzymuje się następującą zależność:

$$T_{init} = O[\text{liczba\_chwastów} \cdot T_{generacji} + \text{liczba\_chwastów} \cdot T_{oceny}] \quad (29)$$

Stworzenie osobnika wymaga przeanalizowania wszystkich zbiorów pierwotnych biorących udział w zapytaniu wraz z możliwymi do wykonania złączeniami pomiędzy zbiorami rekordów. Następnie wybiera się do poszczególnych złączeń takie pary zbiorów rekordów, aby nie wystąpiła sytuacja niemożliwa lub iloczyn kartezjański, którego wykonanie wiązałoby się z bardzo dużym narzutem czasowym. Ostatnim krokiem jest wpisanie wylosowanych numerów zbiorów do struktury reprezentowanej przez uporządkowany zestaw genów, których liczba jest równa liczbie wykonywanych złączeń. Wszystkie te elementy tworzą wzór o postaci:

$$T_{generacji} = O[\text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń} \cdot \text{liczba\_realizowanych\_złączeń}] \quad (30)$$

Jak wspomniano w podrozdziale 3.1, w każdym zapytaniu liczba realizowanych złączeń jest o jeden mniejsza od liczby zbiorów pierwotnych. W ten sposób otrzymuje się uproszczenie formuły (30):

$$T_{generacji} = O[\text{liczba\_zbiorów\_pierwotnych}^2 \cdot \text{liczba\_możliwych\_złączeń}] \quad (31)$$

W trakcie wyznaczania jakości chwastu konieczne jest zbudowanie drzewa wspomagającego obliczanie wartości funkcji celu (rys. 19). Złożoność obliczeniowa tego etapu jest zatem równoznaczna ze stworzeniem wspomnianej struktury. Drzewo budowane jest od dołu, to znaczy od liści do korzenia. W związku z tym w pierwszym kroku uzupełniane są liście drzewa, których liczba jest równa liczbie zbiorów pierwotnych (podrozdział 3.1). Wraz z danymi definiującymi zbiór pierwotny przechowywane są informacje dotyczące możliwych złączeń. Daje to złożoność rzędu:

$$T_{liści} = O[\text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń}] \quad (32)$$

Następnie odczytuje się dane z genów analizowanego osobnika (przechowujących informacje o zbiorach pośrednich) i wpisuje się je do węzłów oraz krawędzi drzewa:

$$T_{węzłów} = O[\text{liczba\_genów} + \text{liczba\_zbiorów\_pośrednich} + \text{liczba\_krawędzi\_drzewa}] \quad (33)$$

Ostatnia faza uzupełniania struktury jest związana z koniecznością wykonania obliczeń zależnych od liczby zbiorów pośrednich i dopasowania możliwych złączeń w trakcie realizacji złączenia dwóch zbiorów pierwotnych:

$$T_{obliczeń} = O[\text{liczba\_zbiorów\_pośrednich} \cdot \text{liczba\_możliwych\_złączeń}^2] \quad (34)$$

Ostateczna złożoność obliczeniowa oceny jakości pojedynczego chwastu jest sumą wcześniej opisanych elementów:

$$T_{oceny} = T_{liści} + T_{węzłów} + T_{obliczeń} = O[\text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń} + \text{liczba\_genów} + \text{liczba\_zbiorów\_pośrednich} + \text{liczba\_krawędzi\_drzewa} + \text{liczba\_zbiorów\_pośrednich} \cdot \text{liczba\_możliwych\_złączeń}^2] \quad (35)$$

Warto nadmienić, że liczba zbiorów pośrednich jest równa liczbie genów, a zarazem jest o jeden mniejsza od liczby zbiorów pierwotnych, a liczba krawędzi drzewa jest o jeden mniejsza od sumy liczby zbiorów pierwotnych i pośrednich (uzasadnieniem tych stwierdzeń jest charakter drzewa realizacji złączeń zaprezentowany na rys. 17b). Można zatem założyć, że liczba zbiorów pośrednich i liczba genów są równe liczbie zbiorów pierwotnych, natomiast liczba krawędzi drzewa jest dwa razy większa od liczby zbiorów pierwotnych. Uwzględniając te założenia, otrzymuje się formułę:

$$T_{oceny} = O[\text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń} + 4 \cdot \text{liczba\_zbiorów\_pierwotnych} + \text{liczba\_zbiorów\_ pierwotnych} \cdot \text{liczba\_możliwych\_złączeń}^2] = O[\text{liczba\_zbiorów\_pierwotnych} \cdot (\text{liczba\_możliwych\_złączeń}^2 + \text{liczba\_możliwych\_złączeń} + 4)] = O[\text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń}^2] \quad (36)$$

Podstawiając zależności (31) i (36) do (29) otrzymuje się:

$$T_{init} = O[\text{liczba\_chwastów} \cdot \text{liczba\_zbiorów\_pierwotnych}^2 \cdot \text{liczba\_możliwych\_złączeń} + \text{liczba\_chwastów} \cdot \text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń}^2] = O[\text{liczba\_chwastów} \cdot \text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń} \cdot (\text{liczba\_zbiorów\_pierwotnych} + \text{liczba\_możliwych\_złączeń})] \quad (37)$$

Kolejne etapy algorytmu, których łączna złożoność jest oznaczona przez T_{ziaren}, są wykonywane iteracyjnie. Pierwszym krokiem należącym do tej fazy jest wyznaczenie, według wzoru (1), liczby ziaren dla każdego chwastu:

$$T_{wyzn.\ ziaren} = O[\text{liczba\_chwastów}] \quad (38)$$

Rozpatrując przypadek pesymistyczny można założyć, że każdy chwast wygeneruje maksymalną ustaloną ich liczbę. Dla każdego ziarna wyznaczane jest miejsce upadku:

$$T_{rozprosz.\ wszystkich} = O[\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot T_{rozproszenie\ pojedyncze}] \quad (39)$$

Szerszego wyjaśnienia wymaga krok rozproszenia pojedynczego ziarna $T_{\text{rozproszenie pojedyncze}}$. Etap ten jest zależny od wersji algorytmu. W przypadku oryginalnej wersji IWO jest on równoznaczny z metodą rozwiewania ($T_{\text{rozproszenie pojedyncze}} = T_{\text{rozwiewania}}$), natomiast w przypadku wersji zmodyfikowanej można przyjąć, że złożoność rozproszenia pojedynczego ziarna jest maksymalną wartością złożoności wszystkich trzech metod (rozwiewania, rozsiewania i staczania):

$$T_{\text{rozproszenie pojedyncze}} = \max\{T_{\text{rozwiewania}}, T_{\text{rozsiewania}}, T_{\text{staczania}}\} \tag{40}$$

W związku z tym wymagane jest oszacowanie złożoności wszystkich wspomnianych metod.

Rozwiewanie w przypadku problemu dyskretnego polega na stworzeniu rozwiązań sąsiednich (różniących się jedną transformacją) w liczbie wygenerowanej za pomocą odpowiedniego rozkładu (rozkładu normalnego o średniej 0 i odchyleniu standardowym opisanym zależnością (2) lub rozkładu *t*-Studenta o ustalonej liczbie stopni swobody przemnożonego przez wartość współczynnika skalującego opisanego wzorem (11)). Złożoność wygenerowania nowego sąsiada jest liniowo zależna od liczby genów tworzonego osobnika, a zatem jest o jeden mniejsza od liczby zbiorów pierwotnych biorących udział w zapytaniu:

$$T_{\text{generacji sąsiada}} = O[\text{liczba\_genów}] = O[\text{liczba\_zbiorów\_pierwotnych}] \tag{41}$$

Otrzymany osobnik jest poddawany ocenie jakości. Wszystkie te elementy tworzą następującą zależność:

$$\begin{aligned} T_{\text{rozwiewania}} &= O[\text{liczba\_generowanych\_sąsiadów} \cdot T_{\text{generacji sąsiada}} + T_{\text{oceny}}] = \\ &O[\text{liczba\_generowanych\_sąsiadów} \cdot \text{liczba\_zbiorów\_pierwotnych} + \\ &\quad \text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń}^2] = \\ &O[\text{liczba\_zbiorów\_pierwotnych} \cdot (\text{liczba\_generowanych\_sąsiadów} + \\ &\quad \text{liczba\_możliwych\_złączeń}^2)] \end{aligned} \tag{42}$$

Rozsiewanie polega na stworzeniu losowego osobnika, więc jego złożoność jest równoważna sumie złożoności generacji pojedynczego osobnika i złożoności jego oceny:

$$\begin{aligned} T_{\text{rozsiewania}} &= T_{\text{generacji}} + T_{\text{oceny}} = \\ &O[\text{liczba\_zbiorów\_pierwotnych}^2 \cdot \text{liczba\_możliwych\_złączeń} + \\ &\quad \text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń}^2] \end{aligned} \tag{43}$$

Ostatnia z technik rozpraszania ziaren, staczanie (podrozdział 5.1), opiera się na przeszukiwaniu lokalnym. Jak zaprezentowano na rys. 12, staczanie polega na iteracyjnym (liczba przejść) generowaniu pewnej z góry założonej liczby sąsiadów (liczba sąsiadów), ocenie ich jakości i przejściu do najlepiej przystosowanego. Całość sprowadza się do następującej formuły:

$$T_{staczania} = O[liczba\_przejść \cdot liczba\_sąsiadów \cdot T_{generacji\ sąsiada} \cdot T_{oceny}] = \quad (44)$$
$$O[liczba\_przejść \cdot liczba\_sąsiadów \cdot liczba\_zbiorów\_pierwotnych^2 \cdot liczba\_możliwych\_złączeń \cdot liczba\_zbiorów\_pierwotnych \cdot liczba\_możliwych\_złączeń^2] =$$
$$O[liczba\_przejść \cdot liczba\_sąsiadów \cdot liczba\_zbiorów\_pierwotnych^3 \cdot liczba\_możliwych\_złączeń^3]$$

Zgodnie z założeniem poczynionym w podrozdziale 5.1, dla uproszczenia przyjmuje się, że liczba przejść jest równa liczbie analizowanych sąsiadów, a zatem:

$$T_{staczania} = O[liczba\_przejść^2 \cdot liczba\_zbiorów\_pierwotnych^3 \cdot liczba\_możliwych\_złączeń^3] \quad (45)$$

Z zestawienia zależności (42), (43) i (45) wynika, że najwyższą złożonością charakteryzuje się technika staczania, zatem można przyjąć, że dla algorytmu exIWO:

$$T_{rozproszenie\ pojedyncze} = \max\{T_{rozwiewania}, T_{rozsiewania}, T_{staczania}\} = T_{staczania} \quad (46)$$

Łączną złożoność wykonywanych iteracyjnie w algorytmie etapów, zarówno dla wersji oryginalnej, jak i zmodyfikowanej, można zapisać następująco:

$$T_{ziaren} = T_{wyzn\_ziaren} + T_{rozprosz.\ wszystkich} \quad (47)$$

Warto jednak zauważyć, że złożoność obliczeniowa kroku wyznaczania liczby generowanych ziaren jest pomijalnie mała w stosunku do złożoności rozproszenia wszystkich ziaren. Oznacza to, że w zależności od wersji algorytmu otrzymuje się złożoności rzędu:

$$T_{ziaren\text{-}IWO\ oryginalne} = O[liczba\_chwastów \cdot maks\_liczba\_ziaren \cdot T_{rozwiewania}] = \quad (48)$$
$$O[liczba\_chwastów \cdot maks\_liczba\_ziaren \cdot liczba\_zbiorów\_pierwotnych \cdot (liczba\_generowanych\_sąsiadów + liczba\_możliwych\_złączeń^2)]$$

$$T_{ziaren\text{-}exIWO} = O[liczba\_chwastów \cdot maks\_liczba\_ziaren \cdot T_{staczania}] = \quad (49)$$
$$O[liczba\_chwastów \cdot maks\_liczba\_ziaren \cdot liczba\_przejść^2 \cdot liczba\_zbiorów\_pierwotnych^3 \cdot liczba\_możliwych\_złączeń^3]$$

Ostatnim nieoszacowanym elementem jest selekcja najlepszych chwastów ($T_{selekcji}$), której wynik stanie się podstawą kolejnego pokolenia. Złożoność tej fazy zależy od przyjętej metody. Wybór z całej populacji jest techniką znaną z oryginalnej wersji IWO, a zatem:

$$T_{selekcji\text{-}IWO\ oryginalne} = T_{selekcji\ z\ całej\ popul.} \quad (50)$$

Dla algorytmu w wersji zmodyfikowanej można założyć, że złożoność selekcji jest równoważna maksymalnej wartości złożoności wszystkich metod (wybór z całej populacji, wybór z osobników potomnych oraz wybór w ramach jednej rodziny):

$$T_{selekcji\text{-}exIWO} = \max\{T_{selekcji\ z\ całej\ popul.}, T_{selekcji\ z\ potomków}, T_{selekcji\ z\ rodziny}\} \quad (51)$$

Aby dokonać wyboru metody selekcji o maksymalnej złożoności, wymagane jest oszacowanie złożoności każdej z tych metod. Jednak złożoność tych metod nie zależy od rozpatrywanego problemu. W związku z tym aktualne są zależności (5), (17) i (18):

$$T_{\text{selekcji z całej popul.}} = O[\log(\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren}) + \text{liczba\_chwastów}] \quad (5)$$

$$T_{\text{selekcji z potomków}} = O[\log(\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren})] \quad (17)$$

$$T_{\text{selekcji z rodziny}} = O[\text{liczba\_chwastów} \cdot (\log(1 + \text{maks\_liczba\_ziaren}) + 1)] \quad (18)$$

Ze wszystkich wymienionych metod selekcji, wybór w ramach jednej rodziny charakteryzuje się najwyższą złożonością obliczeniową, więc wynikiem formuły (51) jest:

$$T_{\text{selekcji-exIWO}} = O[\text{liczba\_chwastów} \cdot (\log(1 + \text{maks\_liczba\_ziaren}) + 1)] \quad (52)$$

Dysponując wszystkimi wymaganymi składowymi, zarówno dla algorytmu w wersji oryginalnej, jak i zmodyfikowanej, można podstawić je do zależności (6) otrzymując:

$$T_{\text{IWO oryginalne}} = O[T_{\text{init}} + \text{liczba\_iteracji} \cdot (T_{\text{ziaren-IWO oryginalne}} + T_{\text{selekcji-IWO oryginalne}})] = \quad (53)$$
$$O[\text{liczba\_chwastów} \cdot \text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń} \cdot (\text{liczba\_zbiorów\_pierwotnych} + \text{liczba\_możliwych\_złączeń}) + \text{liczba\_iteracji} \cdot (\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_zbiorów\_pierwotnych} \cdot (\text{liczba\_generowanych\_sąsiadów} + \text{liczba\_możliwych\_złączeń}^2) + \log(\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren}) + \text{liczba\_chwastów})]$$

oraz

$$T_{\text{exIWO}} = O[T_{\text{init}} + \text{liczba\_iteracji} \cdot (T_{\text{ziaren-exIWO}} + T_{\text{selekcji-exIWO}})] = \quad (54)$$
$$O[\text{liczba\_chwastów} \cdot \text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń} \cdot (\text{liczba\_zbiorów\_pierwotnych} + \text{liczba\_możliwych\_złączeń}) + \text{liczba\_iteracji} \cdot (\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_przejść}^2 \cdot \text{liczba\_zbiorów\_pierwotnych}^3 \cdot \text{liczba\_możliwych\_złączeń}^3 + \text{liczba\_chwastów} \cdot (\log(1 + \text{maks\_liczba\_ziaren}) + 1))]$$

Zakładając, że liczba chwastów jest duża, we wzorze (53) można pominąć wyrażenie z logarytmem, którego wartość jest znacznie mniejsza. Podobnie w zależności (54) można pominąć wyrażenie z logarytmem, gdyż w praktycznych zastosowaniach maksymalna liczba ziaren jest mała. W ten sposób otrzymuje się złożoności:

$$T_{\text{IWO oryginalne}} = O[\text{liczba\_chwastów} \cdot \text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń} \cdot (\text{liczba\_zbiorów\_pierwotnych} + \text{liczba\_możliwych\_złączeń}) + \text{liczba\_iteracji} \cdot (\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_zbiorów\_pierwotnych} \cdot (\text{liczba\_generowanych\_sąsiadów} + \text{liczba\_możliwych\_złączeń}^2) + \text{liczba\_chwastów})] \quad (55)$$

$$T_{\text{exIWO}} = O[\text{liczba\_chwastów} \cdot \text{liczba\_zbiorów\_pierwotnych} \cdot \text{liczba\_możliwych\_złączeń} \cdot (\text{liczba\_zbiorów\_pierwotnych} + \text{liczba\_możliwych\_złączeń}) + \text{liczba\_iteracji} \cdot (\text{liczba\_chwastów} \cdot \text{maks\_liczba\_ziaren} \cdot \text{liczba\_przejść}^2 \cdot \text{liczba\_zbiorów\_pierwotnych}^3 \cdot \text{liczba\_możliwych\_złączeń}^3 + \text{liczba\_chwastów})] \quad (56)$$

Dokonując uproszczeń formuł (55) i (56) przyjęto następujące założenia:

- liczba generowanych sąsiadów jest znacznie mniejsza od kwadratu liczby możliwych złączeń oraz
- liczba zbiorów pierwotnych i liczba możliwych złączeń są znacznie mniejsze od iloczynu liczby iteracji algorytmu, maksymalnej liczby ziaren i liczby możliwych złączeń.

Po zastosowaniu opisanych reguł uzyskuje się następujące końcowe złożoności obliczeniowe:

$$T_{IWO\ oryginalne} = O[liczba\_chwastów \cdot liczba\_zbiorów\_pierwotnych \cdot liczba\_iteracji \cdot maks\_liczba\_ziaren \cdot liczba\_możliwych\_złączeń^2] \quad (57)$$

$$T_{exIWO} = O[liczba\_chwastów \cdot liczba\_zbiorów\_pierwotnych^3 \cdot liczba\_iteracji \cdot maks\_liczba\_ziaren \cdot liczba\_możliwych\_złączeń^3 \cdot liczba\_przejść^2)] \quad (58)$$

Z formuł (57) i (58) wynika, że IWO w wersji zmodyfikowanej charakteryzuje się większą złożonością. Na uwagę zasługuje fakt, iż obydwie zależności mają charakter wielomianowy.

Złożoności obliczeniowe dla oryginalnej i zmodyfikowanej wersji algorytmu IWO dla problemu określania kolejności złączeń w realizacji zapytań w bazach danych można powiązać w następujący sposób:

$$T_{exIWO} = O[T_{IWO\ oryginalne} \cdot liczba\_zbiorów\_pierwotnych^2 \cdot liczba\_możliwych\_złączeń \cdot liczba\_przejść^2)] \quad (59)$$

6. Wyniki badań i ich analiza

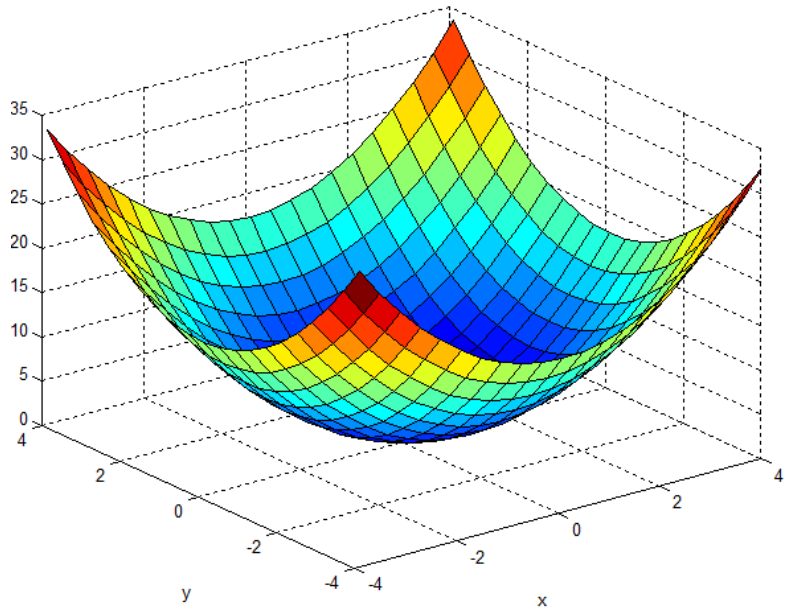
Niniejszy rozdział jest poświęcony doborowi danych wybranych do realizacji testów. Przedstawiono również wyniki eksperymentów oraz ich analizę. Rozdział podzielono na trzy części zawierające zagadnienia dotyczące poszczególnych problemów optymalizacyjnych.

6.1. Problem znajdowania minimum funkcji wielowymiarowej

Problem znajdowania minimum globalnego funkcji wielowymiarowej (funkcji wielu zmiennych) jest jednym z klasycznych zagadnień optymalizacyjnych o charakterze ciągłym. Realne zestawy danych optymalizacyjnych często zawierają dziesiątki, setki, a czasami nawet tysiące wymiarów. Z tego względu ważnym wydaje się być wszechstronne przetestowanie algorytmu optymalizacyjnego.

6.1.1. Dobór funkcji testowych

Z całego wachlarza funkcji do testów postanowiono wybrać pięć: pierwszą funkcję De Jonga [29] oraz funkcje Rosenbrocka [121], Rastrigina [118], Griewanka [55] i Ackleya [3]. Minimum globalnym dla wszystkich wymienionych funkcji jest wartość równa 0 [116, 95].



Rys. 20. Dwuwymiarowa pierwsza funkcja De Jonga.

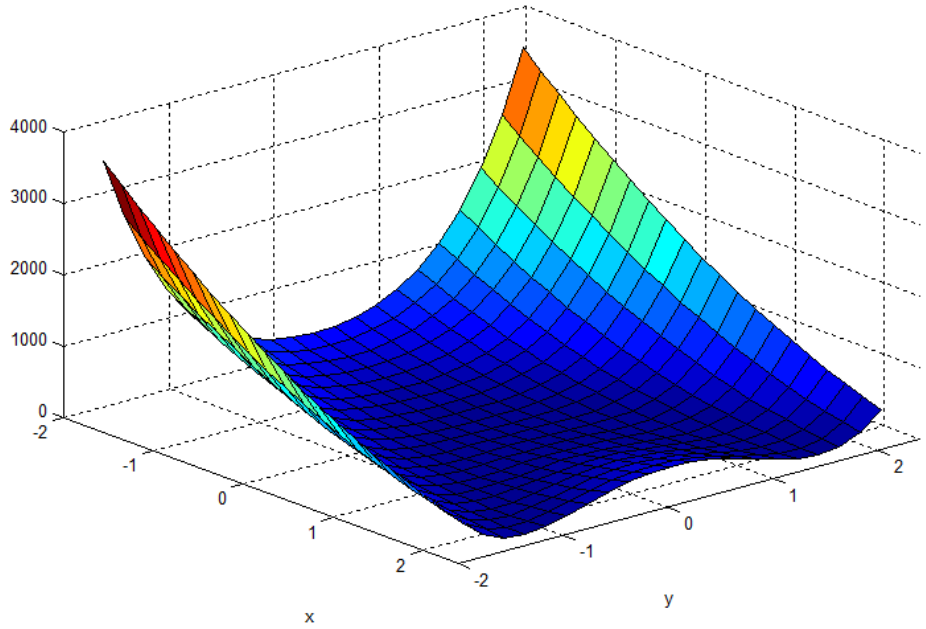
Pierwsza funkcja De Jonga [29] (rys. 20), reprezentująca pierwszą klasę funkcji (klasyfikację funkcji zamieszczono w podrozdziale 3.2), jest jedną z najprostszych funkcji testowych. Wybrano ją ze względu na jej popularność i prostotę. Gdyby analizowana metoda optymalizacyjna nie pozwalała na uzyskanie wyników zbliżonych do globalnego minimum funkcji, bezcelowa byłaby kontynuacja badań dla funkcji bardziej złożonych.

Pierwszą funkcję De Jonga można przedstawić za pomocą następującego wzoru [76]:

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{60}$$

Obszar testowy zwykle zawężany jest do hiperkostki o wymiarach $-5{,}12 \le x_i \le 5{,}12$, dla $i = 1, \ldots, n$, gdzie n jest liczbą wymiarów [106].

Funkcja Rosenbrocka [121] (rys. 21), zwana także: doliną Rosenbrocka, funkcją bananową lub drugą funkcją De Jonga, jest klasycznym problemem optymalizacyjnym [137]. Optimum globalne leży wewnątrz długiej, wąskiej doliny o płaskim dnie. Odnalezienie wspomnianej doliny jest zadaniem trywialnym. Jednak dotarcie do minimum globalnego jest trudne z powodu powolnego spadku dna doliny do minimum. Z tego względu funkcję Rosenbrocka często wykorzystuje się do testowania algorytmów optymalizacyjnych. Klasyczna funkcja Rosenbrocka jest dwuwymiarowa i jednomodalna. Jednak jej wielowymiarowy odpowiednik posiada dwa minima [126].



Rys. 21. Dwuwymiarowa funkcja Rosenbrocka.

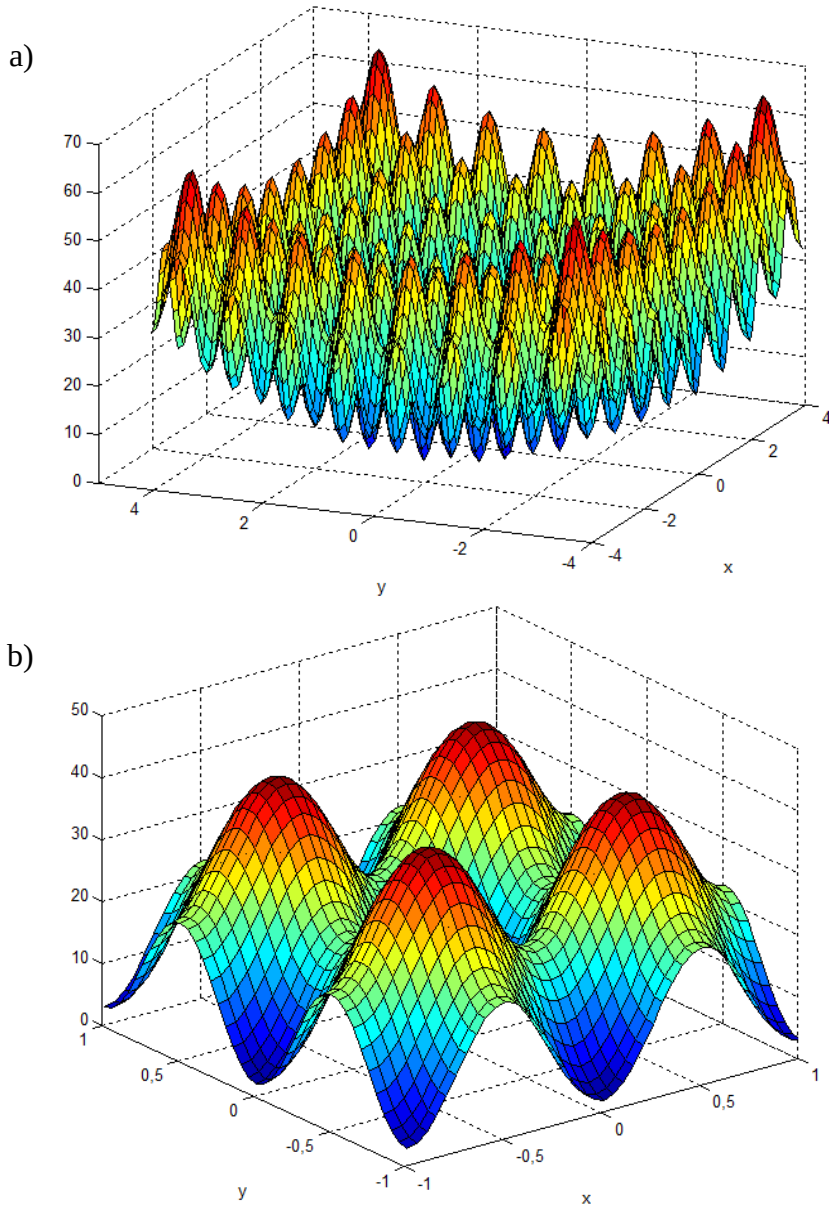
Obszar optymalizacji zawiera się zwykle w przedziale $-2{,}048 \le x_i \le 2{,}048$, dla $i = 1, \ldots, n$, gdzie n jest liczbą wymiarów [11]. Dla liczby wymiarów $n > 1$ funkcja ta opisana jest następującą formułą [37]:

$$f(x) = \sum_{i=1}^{n-1} \left[100 \cdot \left(x_{i+1} - x_i^2 \right)^2 + \left(1 - x_i \right)^2 \right] \tag{61}$$

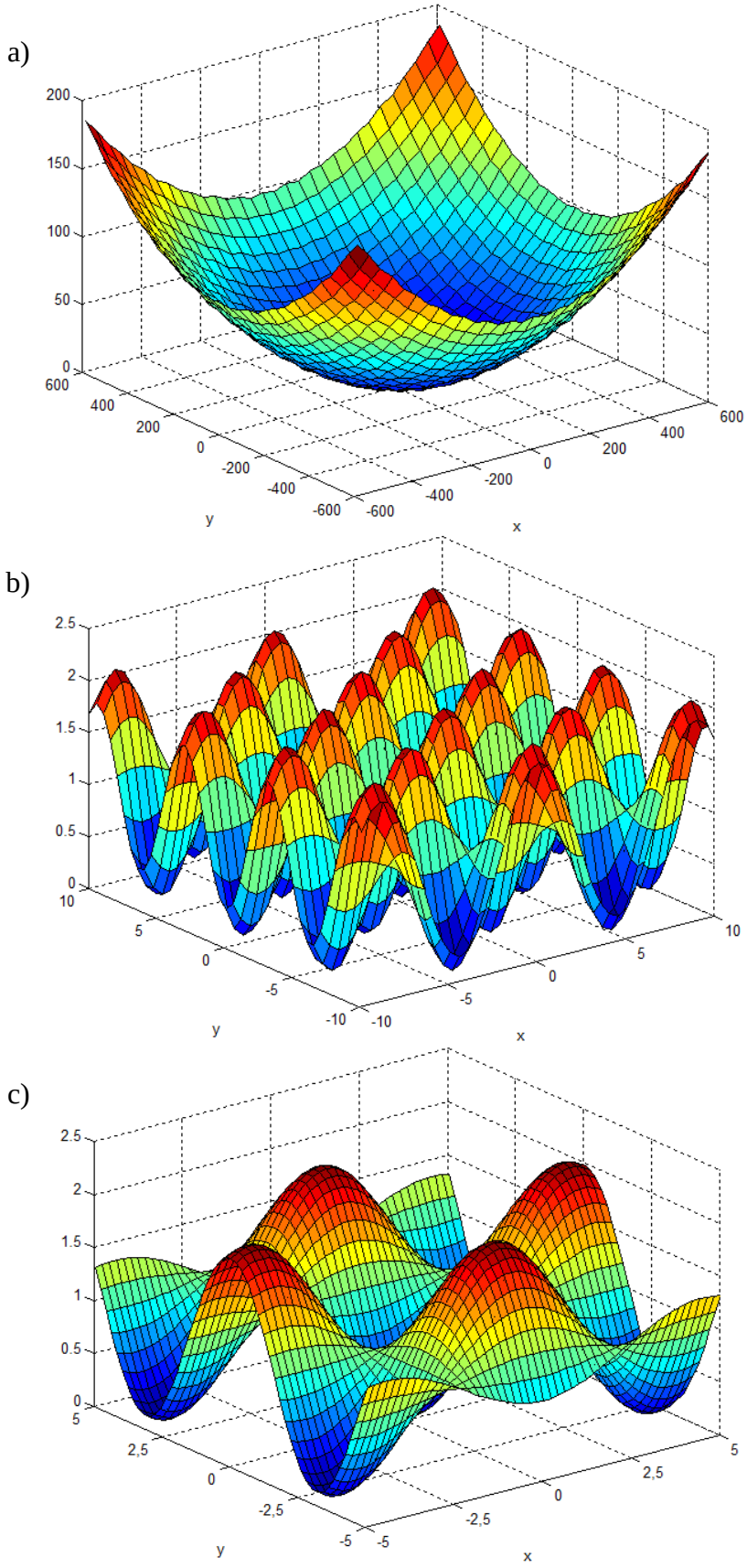
Funkcja Rastrigina [118] (rys. 22), należąca do czwartej klasy funkcji testowych, jest odmianą pierwszej funkcji De Jonga wzbogaconej o czynnik kosinusowy. Powoduje on utworzenie bardzo dużej liczby równomiernie rozmieszczonych minimów lokalnych [137].

Podobnie jak w przypadku pierwszej funkcji De Jonga, obszar optymalizacji zawęża się do zakresu $-5{,}12 \le x_i \le 5{,}12$, dla $i = 1, \ldots, n$ [11]. Definicję funkcji Rastrigina można przedstawić za pomocą wzoru [128]:

$$f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{62}$$



Rys. 22. Dwuwymiarowa funkcja Rastrigina: a) widok ze „średniej" odległości; b) widok z bliska.



Rys. 23. Dwuwymiarowa funkcja Griewanka: a) widok z dużej odległości; b) widok ze średniej odległości; c) widok z bliska.

Funkcja Griewanka [55], należąca również do czwartej grupy funkcji, jest scharakteryzowana formułą [37]:

$$f(x) = \frac{1}{4000} \cdot \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{63}$$

Wygląd funkcji zmienia się wraz z odległością, z jakiej się ją obserwuje (rys. 23). Przypomina ona pierwszą funkcję De Jonga, natomiast jej przybliżenia ukazują liczne minima lokalne. Analizowana jest zwykle hiperkostka o wymiarach $-600 \le x_i \le 600$, dla $i = 1, \ldots, n$ [106].

Ostatnią wybraną funkcją (czwarta grupa funkcji), zawierającą bardzo dużą liczbę minimów lokalnych, jest funkcja Ackleya [3] (rys. 24), którą można przedstawić w następujący sposób [106]:

$$f(x) = -a \cdot \exp\left(-b \cdot \sum_{i=1}^{n} x_i^2\right) - \exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} \cos(c \cdot x_i)\right) + a + \exp(1) \tag{64}$$

gdzie zwyczajowo współczynniki przyjmują następujące wartości: $a = 20$; $b = 0{,}2$; $c = 2\pi$ [128]. Przestrzeń testowa ograniczona jest do przedziału $-32{,}768 \le x_i \le 32{,}768$, dla $i = 1, \ldots, n$ [106].

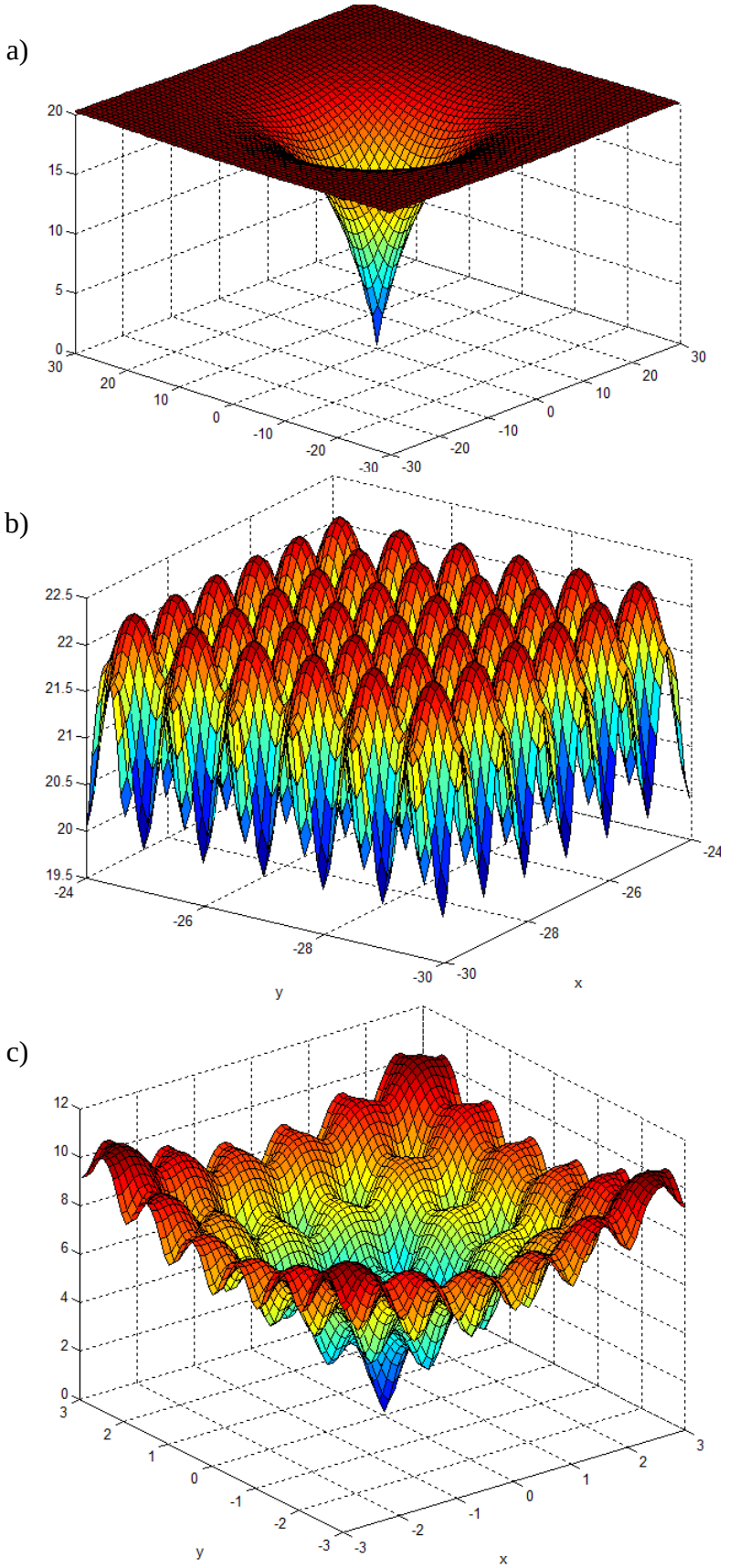
6.1.2. Metodyka przeprowadzonych badań

Wszystkie badania przeprowadzono dwuetapowo. Wstępną fazą było wykonanie eksperymentów mających na celu wyznaczenie jak najlepszych współczynników algorytmu (przedstawionych w tabelach 2 i 3) dla dalszych testów. Po określeniu współczynników realizowano badania właściwe.

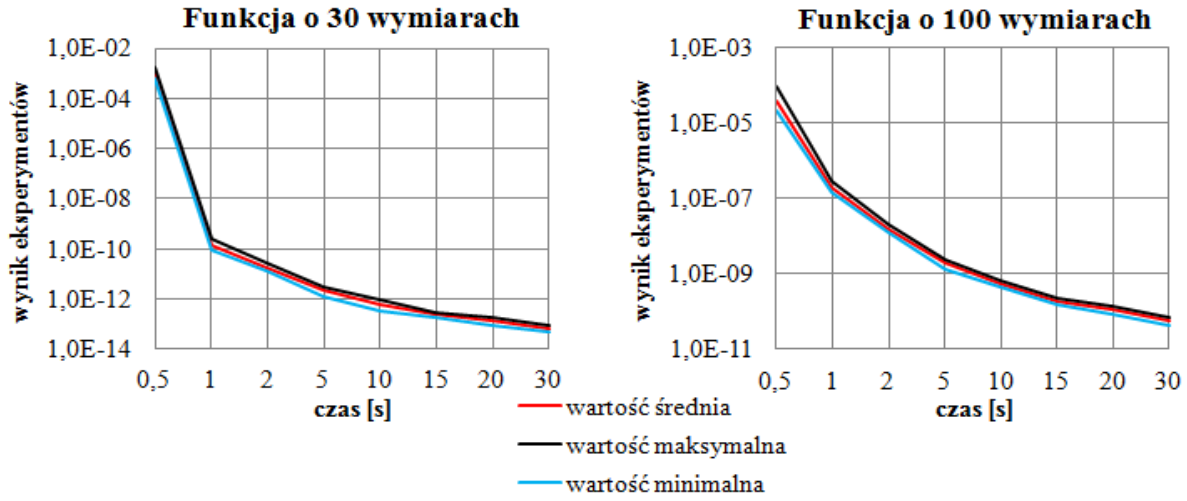
Obliczenia przeprowadzono na komputerze z procesorem Intel Core2 Quad Q6600 2,4 GHz, pamięcią o pojemności 2 GB taktowaną sygnałem zegarowym o częstotliwości 800 MHz oraz systemem operacyjnym Microsoft Windows XP SP3.

6.1.3. Badania z użyciem wybranych funkcji testowych

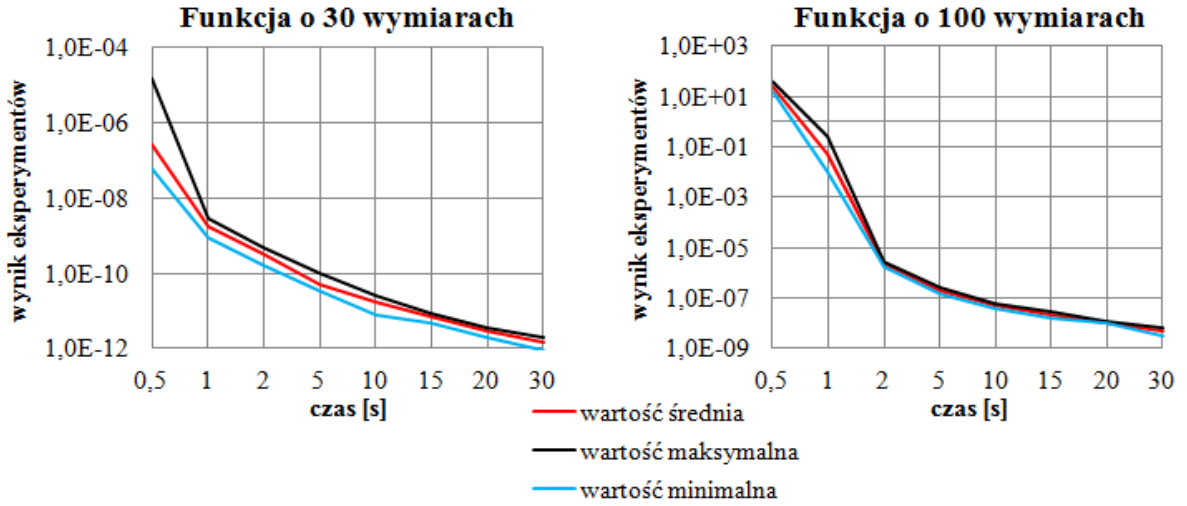
Pierwsza faza badań skupiła się wokół wyznaczenia wartości minimów globalnych dla następujących funkcji testowych: pierwsza funkcja De Jonga, funkcje Griewanka, Rastrigina, Rosenbrocka i Ackleya. Obliczenia wykonano dla funkcji o 30 i 100 wymiarach. Wszystkie testy przeprowadzono przy użyciu autorskiej, zmodyfikowanej wersji IWO, a testy powtórzono stukrotnie. Eksperymenty przeprowadzono dla typowych zakresów początkowych tych funkcji [106]. Zastosowano dwa kryteria zatrzymania algorytmu – interwał czasowy oraz liczbę iteracji, z różnymi wartościami.



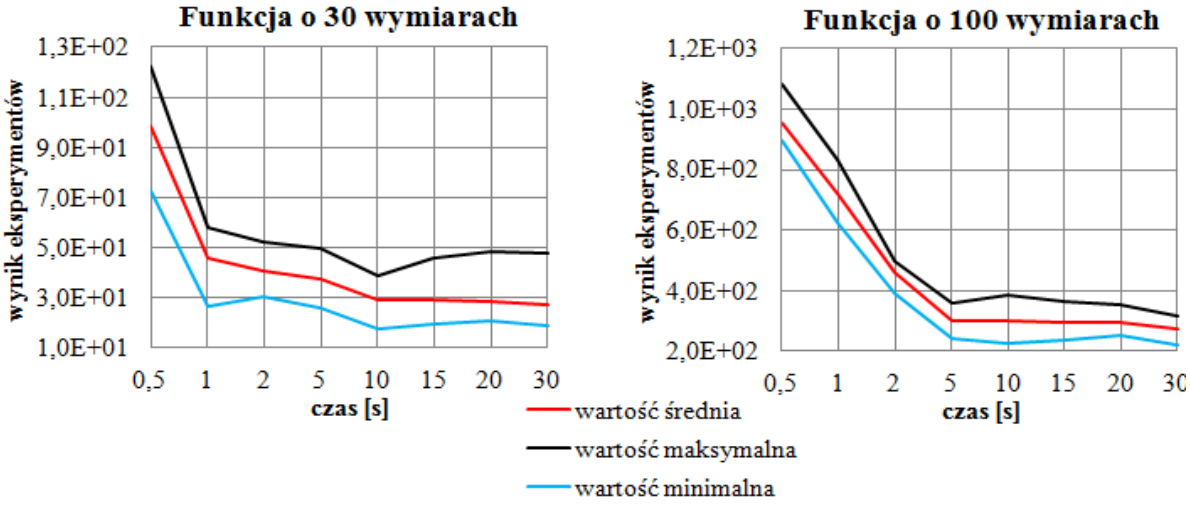
Rys. 24. Dwuwymiarowa funkcja Ackleya: a) widok z dużej odległości; b) powiększenie płaskiej części funkcji; c) powiększenie fragmentu z minimum globalnym.



Rys. 25. Wykresy prezentujące wyniki obliczeń przeprowadzone dla pierwszej funkcji De Jonga.

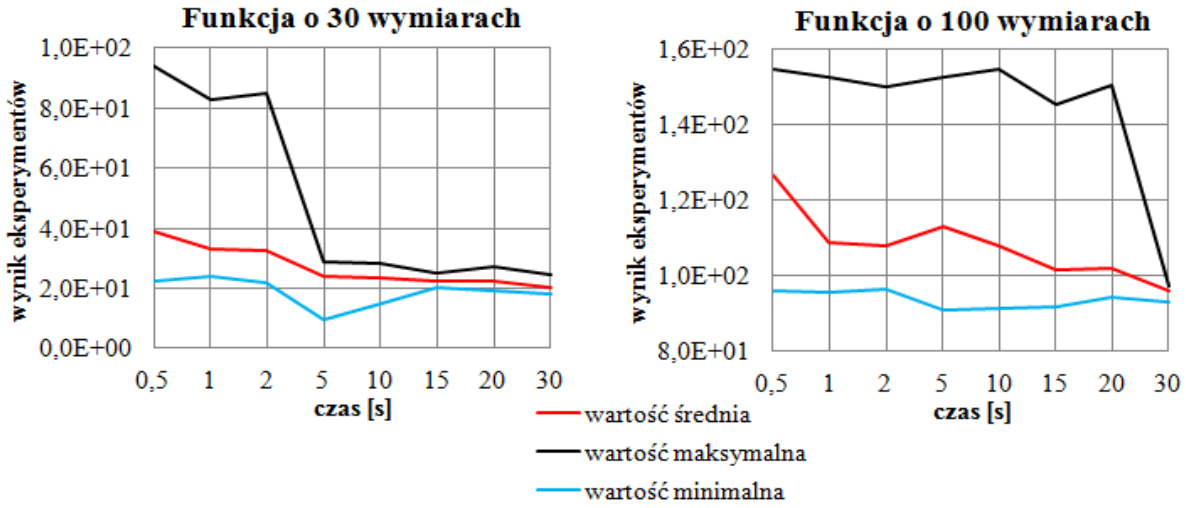


Rys. 26. Wykresy prezentujące wyniki obliczeń przeprowadzone dla funkcji Griewanka.

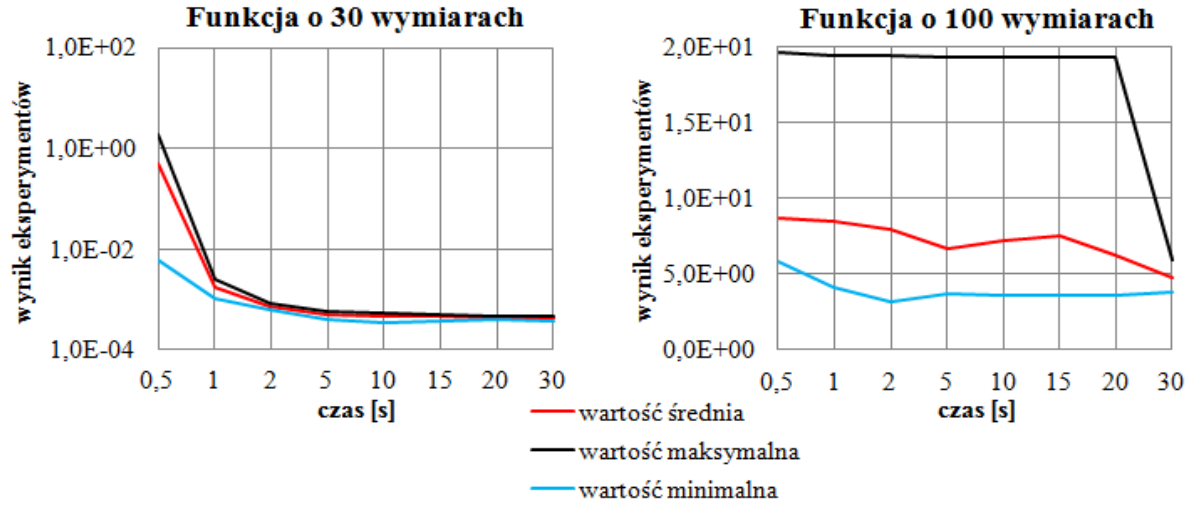


Rys. 27. Wykresy prezentujące wyniki obliczeń przeprowadzone dla funkcji Rastrigina.

Rysunki 25-29 prezentują wyniki obliczeń dla poszczególnych funkcji testowych w formie wykresów. Na każdym wykresie istnieją trzy linie, które pokazują wartości średnie, maksymalne i minimalne ze stu uruchomień algorytmu. Szczegółowe wyniki badań zawarto w formie tabelarycznej w końcowej części pracy w podrozdziale A.1.1.



Rys. 28. Wykresy prezentujące wyniki obliczeń przeprowadzone dla funkcji Rosenbrocka.



Rys. 29. Wykresy prezentujące wyniki obliczeń przeprowadzone dla funkcji Ackleya.

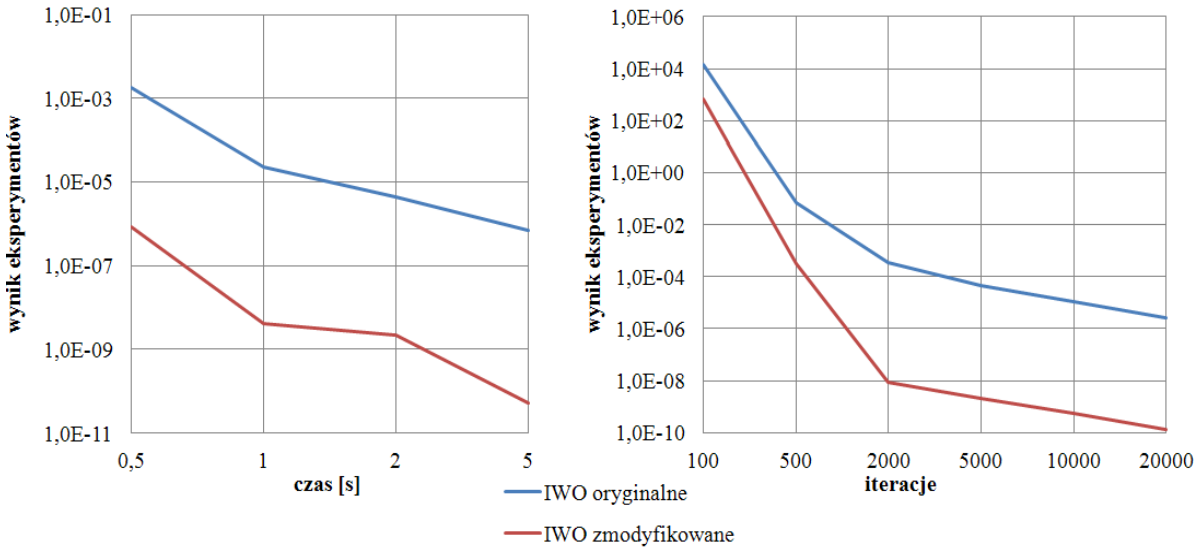
Na podstawie zamieszczonych wyników nie ma możliwości jednoznacznego stwierdzenia, czy exIWO pozwala na otrzymanie dobrych rezultatów. Aby w pełni potwierdzić to twierdzenie, wymagane jest przeprowadzenie testów porównawczych z innymi algorytmami optymalizacyjnymi.

6.1.4. Porównanie exIWO z wersją oryginalną

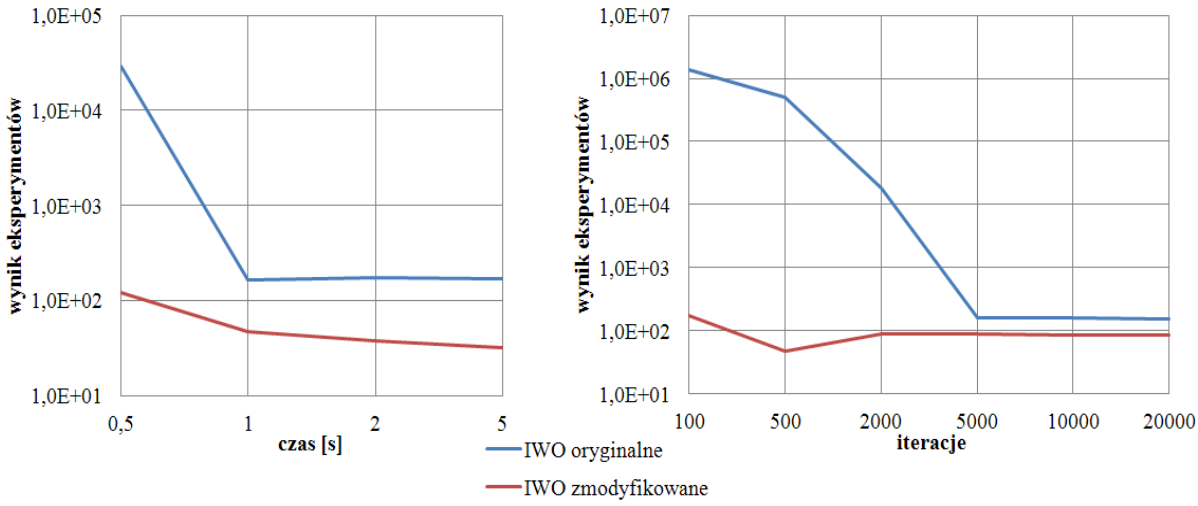
Uzyskane wyniki badań zmodyfikowanego algorytmu IWO porównano z wynikami dla wersji oryginalnej, przedstawionej w pracy [101]. Oparto się na funkcjach Griewanka i Ra-

strigina o 30 wymiarach. Przestrzeń testową ograniczono zgodnie z [101] do hiperkostek o wymiarach odpowiednio: $-512 \le x_i \le 512$ oraz $-100 \le x_i \le 100$ dla $i = 1, \ldots, n$. Badania przeprowadzono dla następujących kryteriów stopu algorytmu: czasu (5, 2, 1 i 0,5 sekundy) oraz liczby iteracji (20 000, 10 000, 5000, 2000, 500 i 100 iteracji). Współczynniki algorytmu dla oryginalnej wersji IWO również zaczerpnięto z pracy [101]. Szczegółowe wyniki eksperymentów zawarto w tabelach 14 (funkcja Griewanka) i 15 (funkcja Rastrigina). Tabele te zamieszczone są w dodatku A.1.2. Jako że algorytmy heurystyczne charakteryzują się wysoką niedeterministycznością, wszystkie obliczenia powtórzono stukrotnie.

Rysunki 30 i 31 ilustrują wyniki numeryczne otrzymane przy użyciu oryginalnej i zmodyfikowanej wersji algorytmu IWO.



Rys. 30. Porównanie wyników otrzymanych przy użyciu oryginalnej i zmodyfikowanej wersji IWO dla 30 wymiarowej funkcji Griewanka.



Rys. 31. Porównanie wyników otrzymanych przy użyciu oryginalnej i zmodyfikowanej wersji IWO dla 30 wymiarowej funkcji Rastrigina.

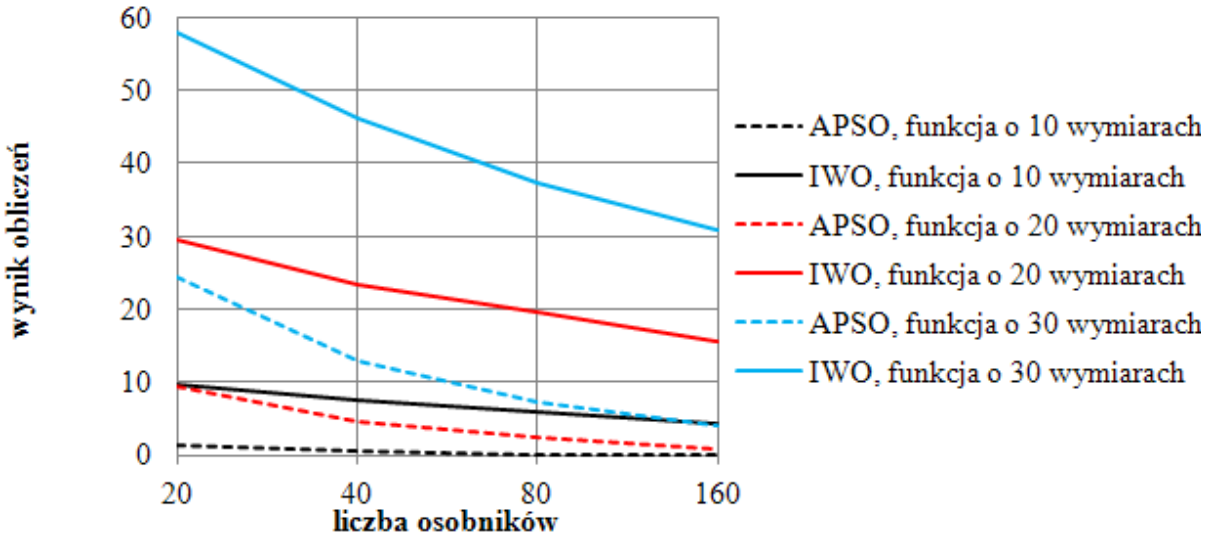
Z danych zestawionych zarówno na rysunkach 30 i 31, jak i w tabelach 14 i 15 wynika, że dla badanych funkcji testowych, przy założonych kryteriach stopu, zmodyfikowana wersja algorytmu IWO pozwala na otrzymanie zdecydowanie lepszych jakościowo wyników. Oznacza to odnalezienie takiego punktu w hiperprzestrzeni, któremu odpowiada mniejsza wartość funkcji testowej. Na uwagę zasługuje fakt, że osie pionowe na obu wykresach zostały opisane skalą logarytmiczną, co potęguje przewagę wersji autorskiej nad oryginalną.

Obydwie wersje IWO cechują się różnymi złożonościami obliczeniowymi. Czas potrzebny na realizację jednej iteracji exIWO jest dłuższy niż w przypadku oryginalnej. Jednak ten nakład czasowy jest opłacalny, gdyż w jego wyniku otrzymuje się rozwiązania lepszej jakości. Przewagi wersji autorskiej upatruje się przede wszystkim w rozbudowanym rozpraszaniu ziaren wokół chwastu macierzystego.

W związku z różnym czasem realizacji jednej iteracji bardziej miarodajnym kryterium zatrzymania algorytmu wydaje się być z góry założony interwał czasowy.

6.1.5. Porównanie exIWO z algorytmem APSO

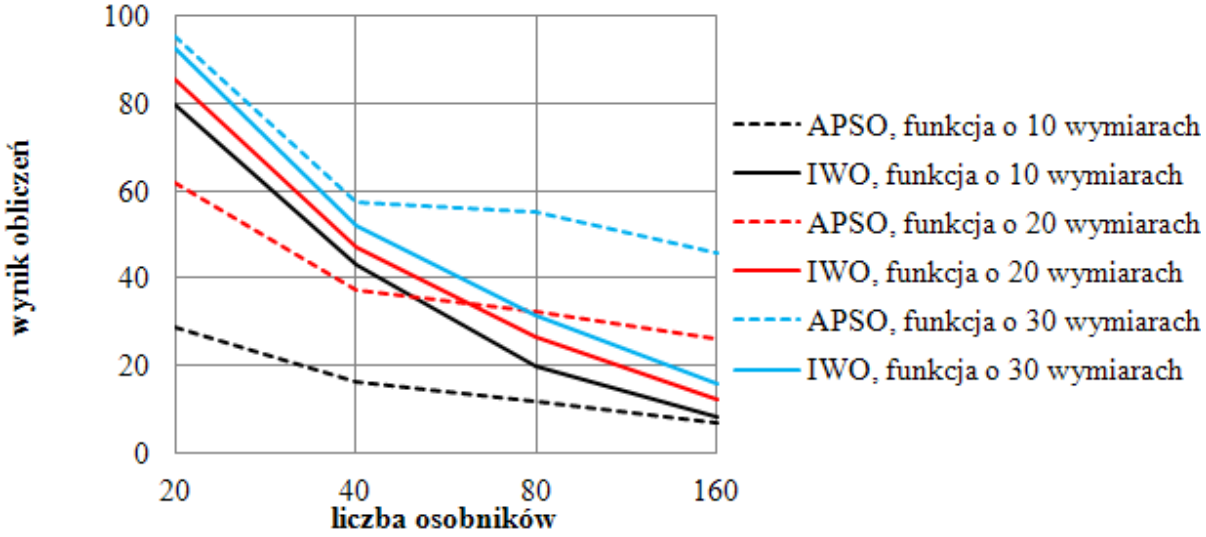
Ostatnią grupę stanowią badania porównawcze z adaptacyjnym algorytmem roju cząstek (ang. *adaptive particle swarm optimization*, APSO) [143]. Wybrano ten algorytm ze względu na jego wysoką skuteczność oraz szerokie rozpowszechnienie w literaturze.



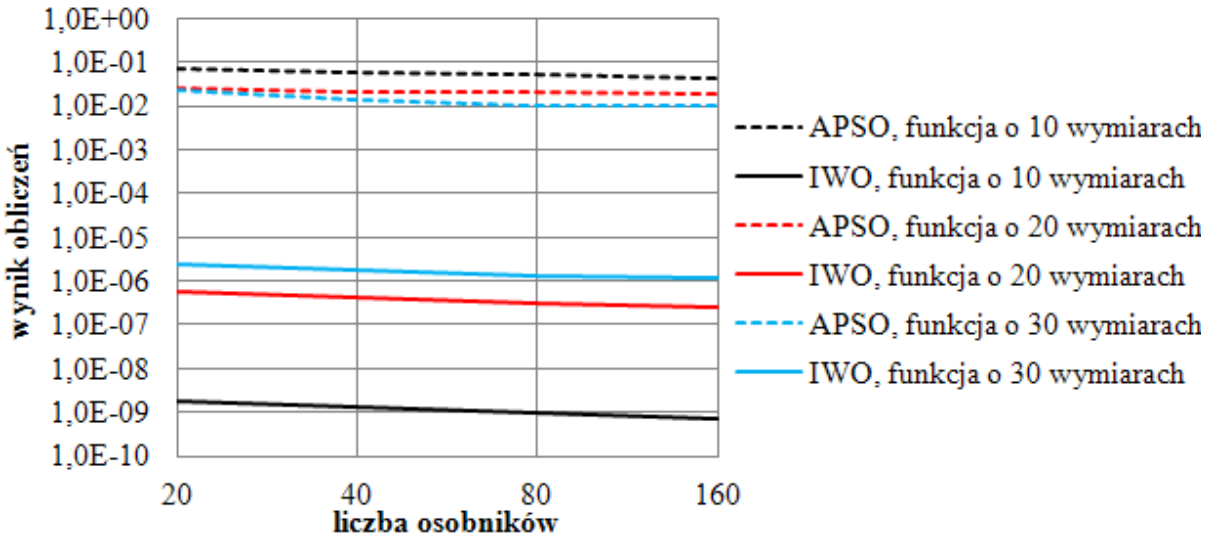
Rys. 32. Porównanie wyników uzyskanych przy pomocy APSO i exIWO na podstawie funkcji Rastrigina.

Podstawą eksperymentów stały się funkcje: Rastrigina, Rosenbrocka i Griewanka. Zgodnie z [143], w każdej iteracji znajdowało się 20, 40, 80 lub 160 osobników. W przypadku algorytmu APSO liczba osobników oznacza liczbę cząstek w populacji, natomiast dla zmodyfikowanego IWO – liczbę chwastów w pokoleniu. Funkcje zbudowane były z 10, 20 lub 30 wymiarów. Liczbie wymiarów ściśle odpowiadały liczby iteracji, równe odpowiednio: 1000,

1500 oraz 2000. Wszystkie obliczenia powtórzono 500 razy. Wyniki obliczeń zebrano w tabelach 16-18 (dodatek A.1.3) oraz zaprezentowano na rysunkach 32-34.



Rys. 33. Porównanie wyników uzyskanych przy pomocy APSO i exIWO na podstawie funkcji Rosenbrocka.



Rys. 34. Porównanie wyników uzyskanych przy pomocy APSO i exIWO na podstawie funkcji Griewanka.

Przedstawione wyniki nie pozwalają jednoznacznie określić, który algorytm jest lepszy. Badania dotyczące funkcji Rastrigina wskazują na wyższość algorytmu APSO, natomiast funkcji Griewanka – zmodyfikowanego algorytmu IWO. Należy zwrócić uwagę, że osie pionowe dla funkcji Rastrigina i Rosenbrocka mają skalę liniową, a dla funkcji Griewanka – logarytmiczną. Fakt ten może delikatnie przechylać szalę na korzyść exIWO.

Przeprowadzone trzy serie eksperymentów dowodzą, że dla problemu znajdowania wartości minimalnej funkcji wielowymiarowej autorska, zmodyfikowana wersja algorytmu IWO

generuje zdecydowanie lepsze wyniki niż wersja oryginalna. Może również z powodzeniem konkurować z innymi znanymi i szeroko rozpowszechnionymi w literaturze heurystycznymi algorytmami optymalizacyjnymi.

Uzyskanie bardzo dobrych wyników w klasycznym problemie o charakterze ciągłym sugeruje, że opracowane metody mogą być równie wartościowe w przypadku dyskretnych problemów optymalizacyjnych.

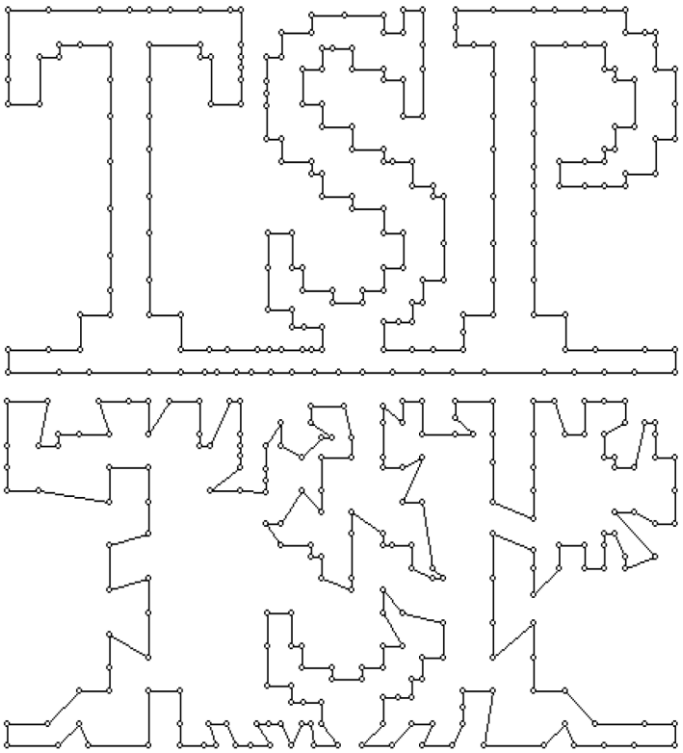
6.2. Problem komiwojażera

Problem komiwojażera jest ciągle ważnym i aktualnym klasycznym problemem optymalizacyjnym o charakterze dyskretnym. Jego odmiany spotyka się m.in. w produkcji elektronicznych płytek drukowanych, jak i w firmach przewozowych. Wyniki przeprowadzonych eksperymentów będą bardzo ważne ze względu na ogólność i rozpowszechnienie niniejszego problemu optymalizacyjnego.

6.2.1. Dobór danych testowych

a)

b)

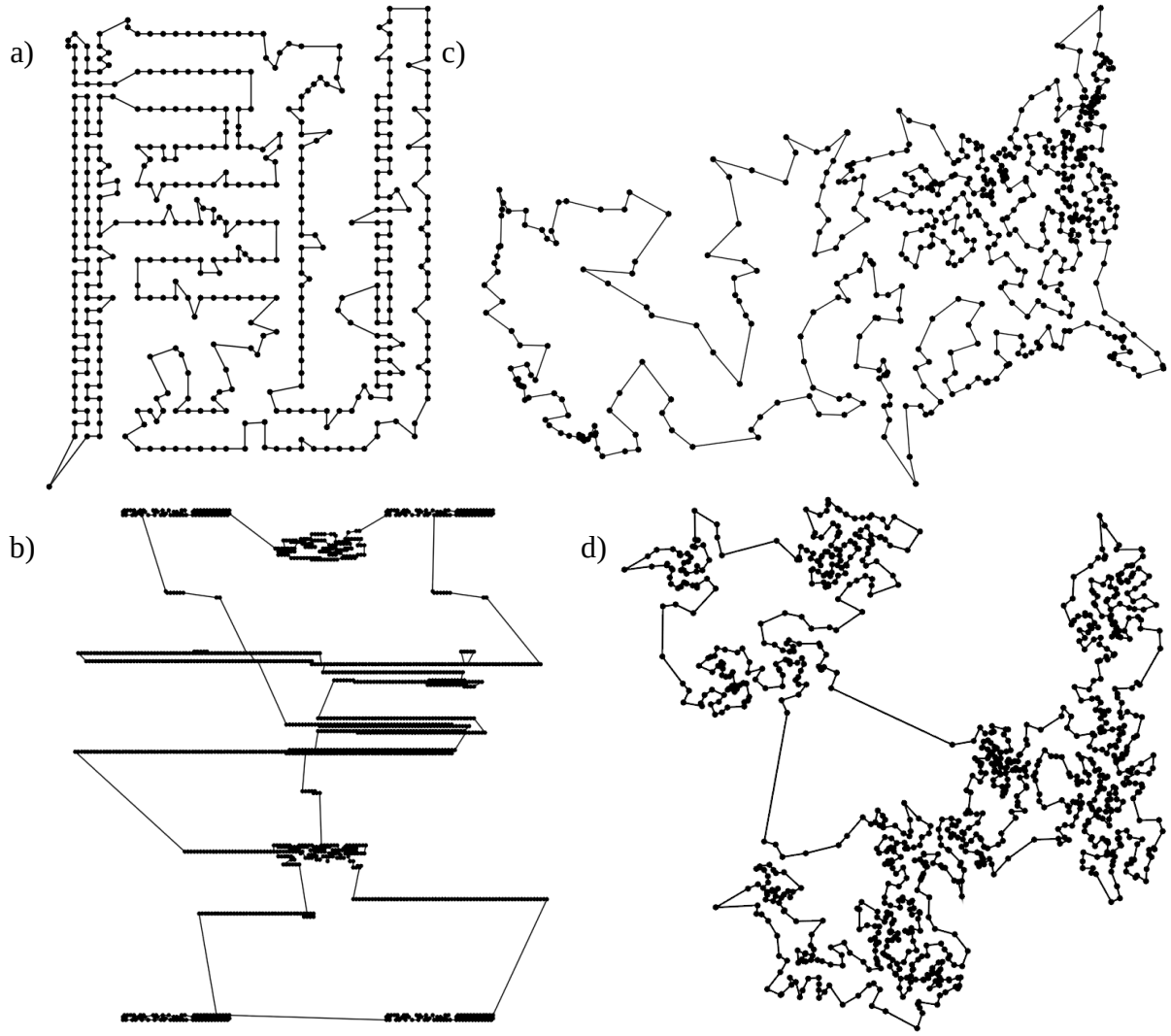


Rys. 35. Przykładowy zestaw danych z odległościami euklidesowymi (*tsp225*) [15]: a) rozwiązanie optymalne; b) rozwiązanie gorsze od optymalnego o 5,19%.

Źródłem danych testowych stały się zbiory udostępnione przez *Research Group Combinatorial Optimization* z Ruprecht-Karls-Universität w Heidelbergu (Niemcy) [120]. Strony tej grupy naukowców udostępniają m.in. ponad sto zestawów danych dla klasycznego symetrycznego problemu komiwojażera (rys. 35) oraz kilkanaście unikalnych zestawów dla nie-

symetrycznego problemu komiwojażera. Większość z nich opisuje prawdziwe dane: miejsca wierceń płytek drukowanych (rys. 36a, 36b), współrzędne miast (rys. 36c), lotnisk i inne (rys. 36d).

Wybrano te zestawy testowe z dwóch przyczyn: dla znacznej większości zestawów znane są wartości optymalne oraz istnieje wiele odniesień w literaturze światowej. Fakt ten pozwala na konfrontację opracowanych metod z wynikami uzyskanymi przy użyciu innych popularnych i skutecznych algorytmów heurystycznych.



Rys. 36. Przykładowe zestawy danych z optymalnymi rozwiązaniami [102]: a) zestaw *pcb442*; b) zestaw *fl1577*; c) zestaw *att532*; d) zestaw *dsj1000*.

Odległości dróg łączących poszczególne miasta obliczane są w zestawach danych na kilka sposobów: euklidesowy, zaokrąglony euklidesowy, pseudo-euklidesowy, geograficzny oraz odległości podane wprost w postaci macierzowej.

Zakładając istnienie dwóch miast A i B o współrzędnych $A = (x_A, y_A)$ oraz $B = (x_B, y_B)$ odległość euklidesowa jest obliczana przy użyciu następującego wzoru:

$$d_{Eukl} = \text{nint}\left(\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}\right) \tag{65}$$

gdzie *nint* jest funkcją zwracającą najbliższą liczbę całkowitą (ang. *nearest integer*). W przypadku zaokrąglonej odległości euklidesowej zamiast funkcji *nint* stosuje się funkcję *sufit* ($\lceil\ \rceil$), zaokrąglającą wartości liczbowe w górę do najbliższej liczby całkowitej. Bardzo podobnie oblicza się odległość pseudo-euklidesową [120]:

$$d_{pseud} = \left\lceil \sqrt{0{,}1 \cdot \left[(x_A - x_B)^2 + (y_A - y_B)^2\right]} \right\rceil \tag{66}$$

Odległość geograficzną oblicza się pomiędzy punktami (miastami) położonymi na sferze o promieniu R = 6378,388 km. Współrzędne każdego punktu są opisane szerokością i długością geograficzną. Obydwie wartości są podane w stopniach i minutach. Dodatnia wartość szerokości geograficznej oznacza półkulę północną, ujemna – południową. Natomiast dodatnia wartość długości to wschód, ujemna – zachód [119]. Pierwszym krokiem obliczania odległości geograficznej jest przeliczenie szerokości i długości ze stopni na radiany. Następnie, zgodnie z [119], korzysta się z następujących zależności:

$$\begin{aligned} q_1 &= \cos(dl_A - dl_B) \\ q_2 &= \cos(sz_A - sz_B) \\ q_3 &= \cos(sz_A + sz_B) \\ d_{geo} &= \left\lfloor \arccos\{0{,}5 \cdot [(1{,}0 + q_1) \cdot q_2 - (1{,}0 - q_1) \cdot q_3]\} + 1{,}0 \right\rfloor \end{aligned} \tag{67}$$

gdzie: dl_A, dl_B – długość geograficzna miast A i B; sz_A, sz_B – szerokość geograficzna miast A i B; arccos – funkcja odwrotna do funkcji kosinus; $\lfloor\ \rfloor$ – funkcja *podłoga* zaokrąglająca liczbę w dół do najbliższej liczby całkowitej.

W przypadku odległości podanych wprost, dane zawierają macierz z wartościami oznaczającymi odległość pomiędzy poszczególnymi miastami.

6.2.2. Metodyka przeprowadzonych badań

Zestawy danych podzielono na grupy ze względu na typ obliczania odległości między miastami, jak i na liczbę miast w nich zawartych. Podobnie jak w przypadku problemu określania wartości minimalnej funkcji wielowymiarowej, wszystkie eksperymenty prowadzono dwuetapowo. Pierwszą fazą było wykonanie testów mających na celu wyznaczenie współczynników algorytmu dla poszczególnych grup. Następnie zrealizowano badania właściwe.

Obliczenia wykonano na komputerze z dwoma procesorami Intel Xeon E5620 2,40 GHz, pamięcią RAM o pojemności 16 GB oraz systemem operacyjnym Microsoft Windows Server 2008 R2 Datacenter 64-bit SP1.

6.2.3. Porównanie operatorów inver-over i odwracania

Seria badań opisana w niniejszym podrozdziale miała na celu przetestowanie przydatności autorskiego algorytmu exIWO do rozwiązywania problemu komiwojażera oraz porównanie operatorów transformacji pojedynczego osobnika: *inver-over* i odwracania. Zamiana, trzeci z opisanych w podrozdziale 5.3.2 operatorów, już w badaniach wstępnych wykazywał bardzo niską skuteczność, przez co zrezygnowano z dalszych badań z jego udziałem. Rezygnacja ta nie niesie za sobą żadnych konsekwencji i nie wpływa na wyniki osiągane przez opracowaną metodę.

Do przeprowadzenia badań postanowiono wykorzystać jak największą liczbę danych testowych. Wybrano niemal wszystkie zestawy opracowane dla symetrycznego problemu komiwojażera o liczbie miast do 10 000 (101 zestawów) oraz wszystkie unikalne zestawy przygotowane dla problemu asymetrycznego (17 zestawów).

Szczegółowe wyniki obliczeń dla problemu symetrycznego zebrano w tabeli 19, dla asymetrycznego – w tabeli 20 (dodatek A.2). Kryterium zatrzymania algorytmu to 10 000 iteracji. Eksperymenty przeprowadzano stukrotnie.

Dla symetrycznego problemu komiwojażera w 22 na 101 przypadków średni rezultat ze 100 uruchomień algorytmu był lepszy dla operatora *inver-over*. W 19 przypadkach odnotowano wartości identyczne, zaś aż dla 60 przypadków lepszy okazał się operator odwracania. Natomiast dla komiwojażera asymetrycznego dzięki operatorowi *inver-over* uzyskano wynik lepszy zaledwie w 1 na 17 przypadków, wartości identyczne dla 2 przypadków, a dla pozostałych 14 – lepsze wyniki uzyskano dla operatora odwracania.

Tabela 4 zawiera uśrednione wyniki z poszczególnych kolumn tabel 19 i 20. Wartości w kolumnach średnia („śr.”), maksimum („maks.”), minimum („min.”) i odchylenie standardowe wyznaczone dla uzyskanych wyników („odch.”) są procentowym odchyleniem od wartości optymalnych. Kolumna numer iteracji („iter.”) jest wartością określającą numer pokolenia algorytmu, w którym pojawiło się najlepsze rozwiązanie; zaś czas, podawany w sekundach, jest interwałem, jaki upłynął od rozpoczęcia do zakończenia algorytmu optymalizacyjnego. Wszystkie opisane wartości są średnimi ze wszystkich zestawów testowych, które z kolei są średnimi ze 100 uruchomień exIWO.

Tabela 4

Podsumowanie wyników badań dla operatorów *inver-over* i odwracania

Typ problemu	**Operator**	**Śr.**	**Maks.**	**Min.**	**Odch.**	**Czas**	**Iter.**
Symetryczny	inver-over	4,37	5,59	3,27	0,46	285,1	6741
	odwracanie	4,16	5,34	3,12	0,44	247,2	6749
Asymetryczny	inver-over	11,66	13,74	9,98	0,77	21,9	6665
	odwracanie	11,23	13,22	9,78	0,73	30,0	6570

Dla symetrycznego problemu komiwojażera przy zastosowaniu operatora *inver-over* wartości optymalne uzyskano dla 28 ze 101 zestawów danych, natomiast dla operatora odwracania – dla 30 zestawów. Zaś dla problemu asymetrycznego wartości optymalne zostały znalezione dla 7 z 17 zestawów zarówno dla operatora *inver-over*, jak i odwracania.

Mimo niewielkiej przewagi operatora odwracania zebrane wyniki eksperymentów nie pozwalają na jednoznaczne określenie, który z badanych operatorów transformacji pojedynczego osobnika pozwala na osiągnięcie lepszych rezultatów. Z tej też przyczyny w dalszych testach postanowiono wykorzystać obydwa operatory.

6.2.4. Porównanie exIWO z algorytmem Meta-RaPS i innymi algorytmami heurystycznymi

Celem niniejszej serii badań było porównanie jakości rozwiązań otrzymywanych za pomocą exIWO z rezultatami uzyskiwanymi przy użyciu innych algorytmów heurystycznych.

Źródłem danych do analizy porównawczej stała się praca [32]. Zawiera ona zestawienie wyników uzyskanych dzięki metaheurystyce losowego przeszukiwania priorytetowego (Meta-RaPS). Wybór padł na metodę Meta-RaPS, ponieważ osiąga ona bardzo dobre wyniki dla przedstawionych w pracy [32] zbiorów testowych, oraz zestawiono je z rezultatami wielu innych popularnych algorytmów optymalizacyjnych.

Tabela 5

Porównanie rezultatów uzyskanych za pomocą IWO i innych strategii

Nazwa metody	**Nazwa zbioru testowego**				
	kroA100	**kroB100**	**kroC100**	**kroD100**	**kroE100**
Meta-RaPS TSP (DePuy i in. [32])	0	0,25	**0**	**0**	0,17
Priority rule (DePuy i in. [32])	0,5	2,46	0,82	1,43	1,1
GRASP (DePuy i in. [32])	0	0,55	0,31	0,42	0,37
Christofides & 2opt 3opt (Lawler i in. [94])	2,51	1,4	1,53	0,17	3,03
Convex hull & 3opt (Lawler i in. [94])	0,37	1,46	1,06	**0,04**	2,46
Nearest neighbour & 2opt 3opt (Lawler i in. [94])	0,14	1,46	1,06	0,73	2,46
2opt 3opt (Lawler i in. [94])	0,81	1,44	0,53	1,74	0,18
Lin-Kernighan (Padberg i Rinaldi, 1991)	0,26	0	0,7	0,17	0,16
Modified Lin-Kernighan (Mark i Morton, 1992)	0	0,17	**0**	**0**	0,21
Composite heuristic, CCAO (Golden i Stewart, 1985)	0	0,97	0,5	0,97	2,54

Tabela 5

Porównanie rezultatów uzyskanych za pomocą IWO i innych strategii

Nazwa metody	**Nazwa zbioru testowego**				
	kroA100	**kroB100**	**kroC100**	**kroD100**	**kroE100**
Delaunay triangulation (Krasnogor i in., 1995)	0,51	2,13	2,79	3,81	2
I^3 (Renaud i in., 1996)	0	0,9	0,5	2	2,6
P-SEC (Rego, 1998)	0	0,32	**0,02**	0,75	0,33
F-SEC (Rego, 1998)	0	0	**0**	**0**	0
Guided local search (Voudouris i Tsang, 1999)	0	–	**0**	–	–
Genetic algorithm, A (Chatterjee i in., 1996)	0,70	–	1,78	1,45	–
Genetic algorithm, B (Chatterjee i in., 1996)	1,80	–	2,10	1,30	–
Neural net (Modares i in., 1999)	0,31	1,43	–	–	–
Neural net (Matsuyama i in., 1992)	1,81	3,37	–	–	–
Neural net (Burke i Damany, 1992)	5,00	4,31	–	–	–
Simulated annealing (Voudouris i Tsang, 1999)	0,42	–	0,80	–	–
Simulated annealing (Malek i in., 1989)	0,09	–	–	–	–
Tabu search (Voudouris i Tsang, 1999)	0	–	0,25	–	–
Tabu search (Malek i in., 1989)	0,33	–	–	–	–
Tabu search & 3opt (Tsubakitani i Evans, 1998)	1,37	–	–	–	–
IWO, inver-over, średnia	0	0,005	0,246	0,508	0,202
IWO, inver-over, minimum	0	0	0,096	0	0
IWO, odwracanie, średnia	0	0,007	0,239	0,488	0,194
IWO, odwracanie, minimum	0	0	0	0,169	0

Przedstawione w tabeli 5 wyniki stanowią porównanie IWO z innymi strategiami optymalizacyjnymi i są procentowym odchyleniem od wartości optymalnych. Liczba osobników dla każdego zestawu danych wynosiła 200, natomiast liczba pokoleń – 10 000. Rezultaty uzyskane dzięki exIWO zostały przedstawione w czterech ostatnich wierszach. Zawierają one wartości minimalne i średnie dla 100 uruchomień algorytmu przy zastosowaniu dwóch operatorów transformacji. Wszystkie wartości liczbowe dla pozostałych algorytmów (poza IWO) zostały zaczerpnięte z pracy [32]. W zestawieniu użyto oryginalnych nazw. Należy zwrócić

uwagę na fakt, że zaprezentowane w tabeli 5 dane, pochodzące z pracy [32], zawierają wartości minimalne uzyskane przy użyciu wymienionych algorytmów.

Pogrubioną czcionką zaznaczono te wartości, w których wartość minimalna uzyskana za pomocą algorytmu IWO okazała się w przypadku co najmniej jednego z operatorów gorsza od danej metody. W 9 na 91 przypadków minimum ze 100 uruchomień exIWO okazuje się gorsze od innych metod, w 12 przypadkach wartości są takie same (i wynoszą 0), natomiast w pozostałych 70 przypadkach – IWO okazuje się lepsze. Na uwagę zasługuje fakt, iż dla wszystkich pięciu zestawów danych osiągnięto wartość optymalną (wartość minimum). Zestawienie to jednoznacznie wskazuje, że zmodyfikowany algorytm IWO może z powodzeniem konkurować z innymi uznanymi strategiami ewolucyjnymi.

6.2.5. Porównanie exIWO ze strategią sieci samoorganizujących

Niniejsza seria badań miała na celu porównanie zmodyfikowanego IWO ze strategiami nienależącymi do rodziny algorytmów ewolucyjnych. Wybór padł na sieci samoorganizujące [78, 79, 80], gdyż wykazują one bardzo dużą skuteczność w rozwiązywaniu problemu komiwojażera. Źródłem danych porównawczych stała się praca [10]. Praca zawiera wyniki uzyskane za pomocą różnych metod opartych na idei sieci samoorganizujących.

Tabela 6

Porównanie rezultatów uzyskanych za pomocą IWO i sieci samoorganizujących

Zestaw	PKN	GN	AVL	KL	KG	SETSP	MGSOM	IWO i-o	IWO od
bier127	3,322	31,181	3,713	2,762	3,079	1,850	1,097	0,069	**0,061**
eil51	4,202	10,493	4,108	2,864	2,864	2,221	1,398	**0,099**	0,127
eil76	6,171	14,182	6,190	4,981	5,483	4,234	3,384	0,338	**0,292**
kroA200	5,311	34,058	5,540	2,836	3,667	3,119	1,972	0,757	**0,716**
lin105	6,921	7,584	6,487	1,985	1,291	1,301	**0,028**	0,033	0,070
pcb442	–	–	17,472	11,072	10,447	10,160	8,577	3,011	**2,580**
pr107	0,454	81,661	1,791	0,734	0,425	0,409	0,172	**0,000**	**0,000**
pr136	7,343	–	6,893	4,531	5,147	4,400	**2,154**	3,834	3,768
pr152	1,523	42,817	1,302	0,968	1,285	1,169	**0,741**	0,924	0,863
rat195	–	–	15,420	12,238	11,916	11,192	5,984	1,868	**1,721**
rd100	–	10,382	4,498	2,095	2,622	2,601	1,172	**0,105**	0,127
st70	2,637	11,956	2,711	1,511	2,326	1,600	1,183	0,594	**0,533**
średnia	–	–	6,344	4,048	4,213	3,688	2,322	0,969	**0,905**

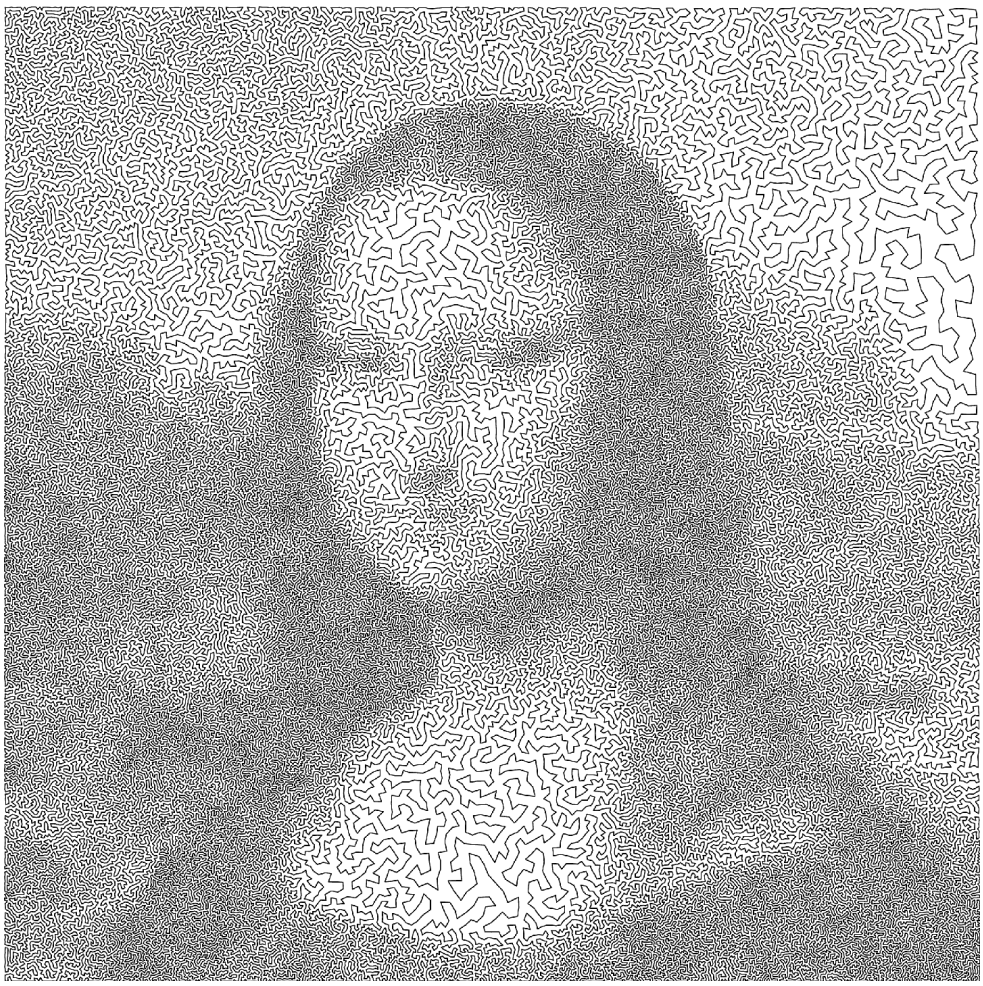
W tabeli 6 zaprezentowano zestawienie wyników uzyskanych przy użyciu exIWO z dwoma różnymi operatorami transformacji oraz różnych sieci samoorganizujących. Wykorzystano następujące skróty: *PKN* – Pure Kohonen Network (Hueter, 1988 oraz Fort, 1988), *GN* – Guilty Net (Burke i Damany, 1992), *AVL* – strategia Angéniol, de la Croix Vaubois

i Le Texier (Angéniol i in., 1988), *KL*, *KG* – warianty lokalny i globalny sieci Kohonena (Local/Global Kohonen Network Incorporating Explicit Statistics – KNIES, Aras i.in., 1999), *SETSP* – SOM Efficiently applied in the TSP (Vieira i in., 2003), *MGSOM* – Modified Growing ring SOM approach for TSP (Bai i in., 2006 [10]). Dwie ostatnie kolumny zawierają wyniki zebrane dla algorytmu IWO. Przeprowadzono eksperymenty dla operatora *inver-over* (kolumna „IWO i-o") oraz odwracania (kolumna „IWO od"). Poszczególne wartości w wierszu „średnia" są wartościami średnimi odpowiadających im kolumn.

Liczba osobników dla zestawów zawierających do 150 miast wynosiła 200, natomiast powyżej 150 miast – 50. Liczba pokoleń algorytmu była równa 10 000. Obliczenia przeprowadzono stukrotnie.

Wartości oznaczone pogrubioną czcionką wskazują metodę optymalizacji, która osiągnęła wartości najbliższe globalnemu minimum dla poszczególnych zestawów testowych. Na uwagę zasługuje fakt, że rezultaty otrzymane dzięki operatorowi odwracania są odrobinę lepsze niż wartości otrzymane za pomocą *inver-over*.

6.2.6. Zestaw Mona Lisa



Rys. 37. Ilustracja zestawu testowego *mona-lisa*.

W lutym 2009 r. Robert Bosch stworzył zestaw danych testowych zawierający 100 000 miast. Jest on reprezentacją słynnego obrazu Mona Lisa autorstwa Leonarda da Vinci. Zestaw ten jest tematem konkursu, który ma na celu znalezienie jak najkrótszej drogi łączącej wszystkie punkty [107] (rys. 37).

Aktualnie znane najlepsze rozwiązanie wynoszące 5 757 191 podał Yuichi Nagata w dniu 17 marca 2009 roku (czas wykonania optymalizacji nie jest podany, stan na dzień 3 stycznia 2014r.). Mimo wielu starań naukowców z całego świata wynik ten dotychczas nie został poprawiony [107]. 27 lipca 2012 r. ogłoszono wartość dolnej granicy, wynoszącej 5 757 084. Wiadomym jest, że rezultat końcowy nie może być niższy od tej wartości [4]. Granicę tę otrzymano w czasie 11,5 procesoro-lat. Całkowita liczba możliwych dróg jest ogromna i wynosi $\frac{1}{2}\cdot(100000-1)!\approx 1{,}412\cdot 10^{456568}$.

Postanowiono sprawdzić jakość otrzymywanych dzięki exIWO rozwiązań dla tak dużego problemu optymalizacyjnego. Przyjęto następujące wartości parametrów algorytmu:

- rozkład – normalny;
- operator – odwracanie;
- selekcja – rodzinna;
- liczba osobników w populacji – 20;
- liczba pokoleń algorytmu – 1 000 000;
- maksymalna liczba ziaren – 5;
- minimalna liczba ziaren – 1;
- początkowe odchylenie standardowe odległości pomiędzy chwastem a ziarnem – 10,0;
- końcowe odchylenie standardowe – 0,01;
- współczynnik modulacji nieliniowej – 3,0;
- liczba przejść dla staczania – 3;
- prawdopodobieństwa poszczególnych strategii transformacji osobnika były zmienne.

Najlepszy wynik, jaki uzyskano przy zastosowaniu autorskiej zmodyfikowanej wersji algorytmu IWO, jest równy 5 919 404, czyli o 2,818% gorszy od najlepszego znanego. Na uwagę zasługuje fakt, że rezultat ten otrzymano w czasie 19 dni 14 godzin i 22 minut (czyli ponad 200 razy krótszym niż wyznaczenie aktualnie obowiązującej granicy dolnej [4]). Informacje te sugerują, że opracowane metody pozwalają na optymalizację z dobrym skutkiem dużych zestawów danych. Niestety, trudno dokładnie ocenić wartość osiągniętego rozwiązania i przewagi czasowej, gdyż, według wiedzy autora niniejszej pracy, literatura naukowa dotycząca tego zestawu danych jest bardzo uboga. Autorzy artykułów wykorzystują inne (nieewolucyjne) metody optymalizacji [12]. Co więcej, nie ma danych dotyczących czasów i mocy obliczeniowych komputerów używanych do innych badań.

Z przedstawionych czterech serii badań dotyczących problemu komiwojażera wynika, że autorska zmodyfikowana wersja algorytmu IWO doskonale sprawdza się w rozwiązywaniu niniejszego problemu. Może ona z bardzo dobrym skutkiem konkurować z wieloma algorytmami optymalizacyjnymi, należącymi zarówno do grupy algorytmów ewolucyjnych, jak i z metodami opartymi na innych ideach. Radzi sobie także z bardzo dużymi problemami zawierającymi nawet 100 000 miast.

6.3. Problem określania kolejności złączeń w realizacji zapytań

Przeprowadzone eksperymenty numeryczne pozwalają twierdzić, że autorska wersja algorytmu IWO bardzo dobrze sprawdza się w optymalizacji klasycznych zadań (znajdowanie minimum wielowymiarowej funkcji matematycznej oraz symetryczny i asymetryczny problem komiwojażera). Uzyskane wyniki sugerują, iż zasadne jest wykorzystanie i przetestowanie opracowanej strategii na centralnym problemie optymalizacyjnym niniejszej pracy, jakim jest określanie kolejności złączeń w realizacji zapytań kierowanych do baz danych. Warto zaznaczyć, że opisywane zagadnienie jest znacznie bardziej złożone obliczeniowo niż poprzednio opisane. Jest to spowodowane znacznie większą ilością danych, które należy uwzględnić w prowadzonych badaniach i koniecznością obliczania dość skomplikowanej funkcji celu (aspekty te opisano szerzej w podrozdziale 5.3.3).

6.3.1. Dobór danych testowych

Do przeprowadzenia kompleksowych badań wymagana jest odpowiednia liczba danych. Dane te powinny zawierać zapytania o różnych grafach złączeń, zarówno o charakterze gwiazdy, łańcucha, jak i nieregularnym. Zapytania testowe winny wykorzystywać jak największą liczbę tabel. Przypuszcza się, że im większa będzie liczba tabel, tym bardziej będą uwydatnione różnice w wynikach otrzymywanych przy użyciu różnych metod.

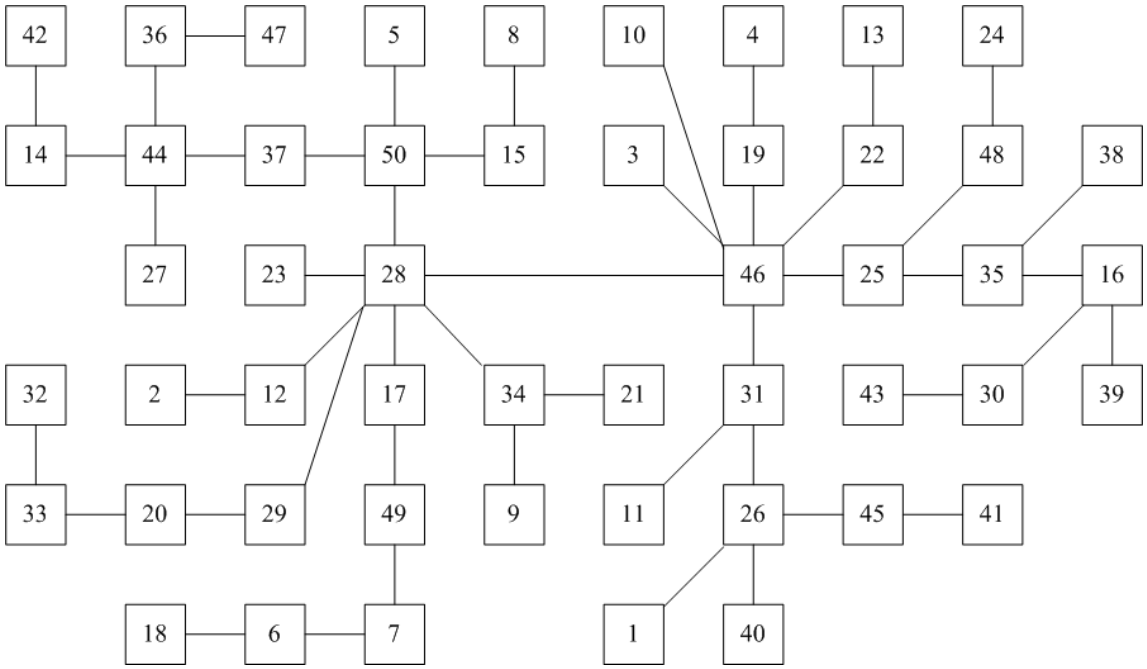
Niestety, pomimo wielu starań nie znaleziono wystarczającej liczby gotowych zestawów danych pozwalających na przeprowadzenie opisanych testów. Skonstruowano zatem generator danych oraz zapytań testowych. Wzorowano się przy tym na pracy [108].

Na potrzeby niniejszych badań utworzono 90 baz danych zawierających po 50 tabel każda (rys. 38). W tabeli 7 zebrano najważniejsze informacje dotyczące przygotowanych zestawów testowych.

Opracowane zestawy można podzielić pod względem wykorzystanego grafu złączeń na trzy grupy:

- zestawy oparte na grafach o charakterze łańcucha (20 zestawów);
- zestawy oparte na grafach o charakterze gwiazdy (30 zestawów); liczba zestawów jest zwiększona w stosunku do zestawów opartych na grafach o charakterze łańcucha ze względu na dodatkowe kryterium, jakim jest liczba wierszy w tabeli faktów;

- zestawy oparte na grafach o charakterze nieregularnym (40 zestawów); zdecydowano się na podwojenie liczby zestawów testowych z grafem o charakterze nieregularnym, w stosunku do danych z grafem łańcuchowym, z dwóch przyczyn: tego typu zapytania występują najczęściej; optymalizacja kolejności realizacji złączeń jest bardziej skomplikowana z powodu większej liczby możliwych do zrealizowania złączeń.



Rys. 38 Przykładowy zestaw danych testowych – nieregularny graf złączeń oparty na 50 tabelach.

Dodatkowo każdą z wymienionych grup można podzielić na dwie równoliczne podgrupy pod względem liczby wierszy, jakie znajdują się w tabelach. Zestawy z tabelami zawierającymi od 10 do 5 000 wierszy nazywane są skrótowo *bazami* z *małymi tabelami*, natomiast zestawy z tabelami zawierającymi od 100 do 50 000 wierszy – *bazami* z *dużymi tabelami*.

Liczba wierszy dla poszczególnych tabel jest losowana z różnym prawdopodobieństwem (szczegółowe informacje zawarto w tab. 7).

Jako dodatkową długość wiersza należy rozumieć kolumnę dodaną do każdej z tabel. Ma ona losowy rozmiar w granicach od 50 do 500 znaków. Kolumna ta reprezentuje dane zawarte w każdym wierszu, a nie będące kluczem głównym lub obcym. Dodatkowa długość wiersza pozwala na symulację obciążenia systemu bazodanowego związanego z odczytem danych z dysku twardego. Wspomniany odczyt ma duże znaczenie przy wyznaczaniu wartości funkcji celu, co zostało dokładniej opisane w podrozdziale 5.3.3.

Do każdego zestawu danych przyporządkowane jest jedno zapytanie pobierające wszystkie informacje ze wszystkich tabel (SELECT * FROM…). Dane testowe skonstruowano w taki sposób, aby nie było sytuacji wymuszającej realizację iloczynu kartezjańskiego, ani żeby nie było możliwości utworzenia zamkniętych cykli. Wszystkie zapytania zwracały jako wynik końcowy od 100 000 do 1 000 000 wierszy.

Tabela 7

Opis przygotowanych zestawów testowych

Opis	Liczba zestawów	Typ grafu	Liczba wierszy w tabeli	Dodatkowa długość wiersza
G 10 M	5	gwiazda	10-99 z prawdop. 0,2 100-999 z prawdop. 0,6 1 000-5 000 z prawdop. 0,2 10 000 w tabeli faktów	od 50 do 500 znaków
G 10 D	5	gwiazda	100-999 z prawdop. 0,2 1 000-9 999 z prawdop. 0,6 10 000-50 000 z prawdop. 0,2 10 000 w tabeli faktów	od 50 do 500 znaków
G 100 M	5	gwiazda	10-99 z prawdop. 0,2 100-999 z prawdop. 0,6 1 000-5 000 z prawdop. 0,2 100 000 w tabeli faktów	od 50 do 500 znaków
G 100 D	5	gwiazda	100-999 z prawdop. 0,2 1 000-9 999 z prawdop. 0,6 10 000-50 000 z prawdop. 0,2 100 000 w tabeli faktów	od 50 do 500 znaków
G 1000 M	5	gwiazda	10-99 z prawdop. 0,2 100-999 z prawdop. 0,6 1 000-5 000 z prawdop. 0,2 1 000 000 w tabeli faktów	od 50 do 500 znaków
G 1000 D	5	gwiazda	100-999 z prawdop. 0,2 1 000-9 999 z prawdop. 0,6 10 000-50 000 z prawdop. 0,2 1 000 000 w tabeli faktów	od 50 do 500 znaków
Ł M	10	łańcuch	10-99 z prawdop. 0,2 100-999 z prawdop. 0,6 1 000-5 000 z prawdop. 0,2	od 50 do 500 znaków
Ł D	10	łańcuch	100-999 z prawdop. 0,2 1 000-9 999 z prawdop. 0,6 10 000-50 000 z prawdop. 0,2	od 50 do 500 znaków
N M	20	nieregularny	10-99 z prawdop. 0,2 100-999 z prawdop. 0,6 1 000-5 000 z prawdop. 0,2	od 50 do 500 znaków
N D	20	nieregularny	100-999 z prawdop. 0,2 1 000-9 999 z prawdop. 0,6 10 000-50 000 z prawdop. 0,2	od 50 do 500 znaków

W tabeli 7 pierwsza litera opisu wskazuje na typ grafu złączeń danego zestawu (G – gwiazda, Ł – łańcuch, N – nieregularny). Następująca po niej liczba, dotycząca jedynie danych z grafem o charakterze gwiazdy, oznacza liczbę wierszy (w tysiącach) w tabeli centralnej, zwanej tabelą faktów. Ostatnia litera dotyczy liczby wierszy w tabelach (M – *bazy z tabelami małymi* zawierającymi z odpowiednimi prawdopodobieństwami od 10 do 5 000 wierszy; D – *bazy z tabelami dużymi* zawierającymi od 100 do 50 000 wierszy).

6.3.2. Metodyka przeprowadzonych badań

Celem przeprowadzanych eksperymentów było porównanie czasów wykonania zapytania uzyskanych w systemie Microsoft SQL Server 2008. Dla każdego zapytania realizowano z jednej strony plan wyznaczony przez moduł optymalizacyjny systemu zarządzania bazą danych, z drugiej zaś – plany skonstruowane przez algorytm IWO. Wykonanie planów utworzonych przy użyciu opracowanej strategii było możliwe dzięki zestawowi *wskazówek* (ang. *hints*) pozwalających użytkownikowi na wymuszenie określonych zachowań modułu optymalizacyjnego [30].

Podobnie jak w przypadku określenia minimum funkcji wielowymiarowej oraz problemu komiwojażera wszystkie badania przeprowadzono dwuetapowo. Fazą wstępną było określenie współczynników algorytmu dla każdego z typów grafu złączeń z osobna. Jest to możliwe ze względu na łatwą metodę sprawdzenia typu grafu (szczegóły zawarto w podrozdziale 5.3.3). Należy podkreślić, że w tej fazie nie wykorzystywano danych testowych.

Badania właściwe przeprowadzono w następujących krokach:

1. Określenie planu realizacji zapytania testowego przy użyciu algorytmu IWO. Przyjęto, że minimalny czas obliczeń wynosi 5 sekund, zaś chwilę ich zakończenia wyznacza utworzenie pierwszej populacji, którą skonstruowano już po upływie tego czasu. Wartość interwału czasowego wybrano arbitralnie na podstawie stwierdzenia zamieszczonego w rozdziale 2 niniejszej pracy, a mówiącego, że z praktycznego punktu widzenia nie do zaakceptowania jest sytuacja, w której czas poszukiwania rozwiązania optymalnego znacznie przekracza czas realizacji zapytania według wyznaczonego planu. Ze względu na niedeterministyczny charakteru algorytmu IWO obliczenia powtarzano dziesięciokrotnie, otrzymując w ten sposób 10 różnych planów realizacji zapytania.
2. Wymuszenie realizacji zapytania testowego w systemie Microsoft SQL Server 2008 według planu wyznaczonego w punkcie 1. Zastosowanie wskazówki OPTION(FORCE ORDER) [136] pozwoliło na narzucenie takiego porządku złączeń, jaki został wyznaczony przez algorytm IWO. Ponadto dzięki wskazówce OPTION(LOOP JOIN) wymuszono realizację złączeń zgodnie z algorytmem pętli zagnieżdżonych [45]. Użycie tej metody zakłada aktualna wersja algorytmu IWO. W przyszłości planuje się rozszerzenie możliwości algorytmu o inne metody realizacji złączeń. Ponieważ dla każdego wariantu planu

otrzymanego dla danego zapytania wykonywano 10 uruchomień, rejestrowano w ten sposób 100 pomiarów czasu realizacji.

3. Określenie planu realizacji zapytania testowego za pomocą metody wbudowanej w systemie Microsoft SQL Server 2008. Optymalizator dysponował histogramami opisującymi aktualne dane zawarte w tabelach.
4. Następnie wymuszano wykonanie zapytania według planu określonego w punkcie 3. Dla zapewnienia identycznych warunków realizacji zapytania złączenia były wykonywane przy użyciu metody pętli zagnieżdżonych, gdyż tylko ta metoda jest wspierana aktualnie przez algorytm IWO. Zapytania wykonywano 10-krotnie otrzymując 10 pomiarów czasu realizacji zapytania.

Należy zaznaczyć, że na wszystkich kolumnach z kluczami głównymi oraz obcymi utworzono indeksy typowe dla tego systemu zarządzania bazą danych. Ponadto dla uniknięcia wpływu wcześniej wykonywanych operacji na rezultat kolejnej każdorazowo opróżniano bufory pamięci podręcznej (*cache*) wykonując zestaw poleceń: CHECKPOINT oraz DBCC DROPCLEANBUFFERS.

Wszystkie obliczenia przeprowadzono na laptopie Dell Vostro 3500 z procesorem Intel Core i5 M 460 2,53 GHz oraz pamięcią DDR 3 o pojemności 4 GB taktowanej z częstotliwością 1066 MHz.

6.3.3. Porównanie czasów wykonania zapytań w systemie SQL Server 2008

W związku z bardzo dużą liczbą przeprowadzonych eksperymentów szczegółowe rezultaty badań zamieszczono w dodatku A.3. Dane zbiorcze zawarto w tabeli 8. Wartości w tej tabeli koncentrują się na czasach wykonania zapytań. Wartości w kolumnie *względna jakość rozwiązań* są zakresami odchylenia pomiędzy średnim czasem wykonania zapytania na podstawie planu wygenerowanego przez metodę zawartą w systemie SQL Server 2008, a planu zbudowanego przez exIWO. Można to zapisać następującą relacją:

$$\text{względna\_jakość\_rozwiązań} = \frac{\text{śr\_czas\_SQLServera} - \text{śr\_czas\_IWO}}{\text{śr\_czas\_SQLServera}} \cdot 100\% \quad (68)$$

Wartość dodatnia względnej jakości rozwiązań oznacza, że zapytanie zrealizowane przy użyciu IWO zostało wykonane szybciej niż na podstawie metody wbudowanej. W przypadku, gdy czas realizacji zapytania przy użyciu zmodyfikowanego algorytmu IWO jest dłuższy niż dwukrotna wartość czasu realizacji zapytania przy użyciu metody wbudowanej w optymalizator Microsoft SQL Server 2008, względna jakość rozwiązań jest niższa niż -100%.

Liczby zawarte w poszczególnych komórkach tabeli 8 oznaczają liczbę realizacji zapytań, opartych na algorytmie exIWO, które mieszczą się w danym przedziale odchylenia.

Rysunek 39 ilustruje wyniki zawarte w tabeli 8. Dane te zostały przeskalowane w taki sposób, aby suma wartości dla każdego typu grafu była równa jedności. Oś pozioma stanowi

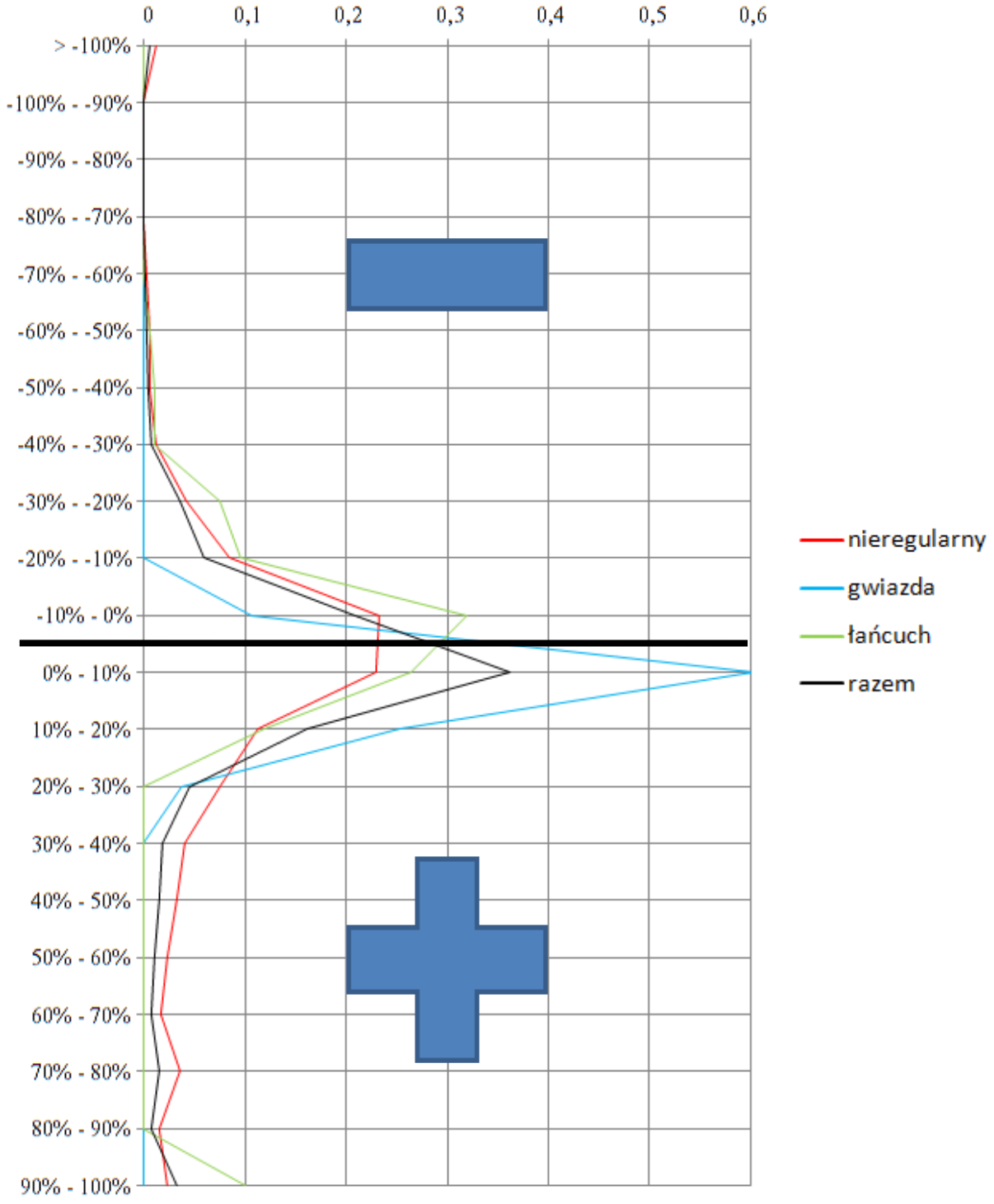
zatem prawdopodobieństwo wystąpienia czasu wykonania zapytania, zrealizowanego przy użyciu algorytmu IWO, w danym przedziale odchylenia w stosunku do czasu realizacji zapytania zrealizowanego na podstawie metody wbudowanej w system Microsoft SQL Server 2008.

Tabela 8

Zestawienie wyników badań dotyczących optymalizacji zapytań przy użyciu IWO

Względna jakość rozwiązań	**Gwiazda**							**Łańcuch**			**Nieregularny**			**Suma**
	10 M	**10 D**	**100 M**	**100 D**	**1000 M**	**1000 D**	**Σ**	**M**	**D**	**Σ**	**M**	**D**	**Σ**	**Σ**
< -100%	0	0	0	0	0	0	0	0	0	0	4	1	5	5
-100% – -90%	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-90% – -80%	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-80% – -70%	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-70% – -60%	0	0	0	0	0	0	0	0	0	0	0	1	1	1
-60% – -50%	0	0	0	0	0	0	0	1	0	1	1	1	2	3
-50% – -40%	0	0	0	0	0	0	0	0	2	2	1	1	2	4
-40% – -30%	0	0	0	0	0	0	0	2	0	2	2	3	5	7
-30% – -20%	0	0	0	0	0	0	0	10	5	15	7	10	17	32
-20% – -10%	0	0	0	0	0	0	0	11	8	19	13	21	34	53
-10% – 0%	8	0	9	0	0	15	32	23	41	64	44	49	93	189
0% – 10%	42	20	20	44	20	35	181	17	36	53	41	51	92	326
10% – 20%	0	19	21	6	30	0	76	16	8	24	17	28	45	145
20% – 30%	0	11	0	0	0	0	11	0	0	0	17	13	30	41
30% – 40%	0	0	0	0	0	0	0	0	0	0	9	7	16	16
40% – 50%	0	0	0	0	0	0	0	0	0	0	9	4	13	13
50% – 60%	0	0	0	0	0	0	0	0	0	0	8	1	9	9
60% – 70%	0	0	0	0	0	0	0	0	0	0	1	6	7	7
70% – 80%	0	0	0	0	0	0	0	0	0	0	11	3	14	14
80% – 90%	0	0	0	0	0	0	0	0	0	0	6	0	6	6
90% – 100%	0	0	0	0	0	0	0	20	0	20	9	0	9	29

Z informacji zawartych w tabeli 8 i na rysunku 39 można wywnioskować, że w przypadku zapytań o grafie łańcuchowym algorytm IWO generuje plany trochę gorsze niż metoda wbudowana (103 na 200 przypadków), natomiast w przypadku zapytań z grafem o charakterze gwiazdy (268 na 300 przypadków) oraz, w mniejszym stopniu, nieregularnym (241 na 400 przypadków) zaobserwować można sytuację odwrotną. Na uwagę zasługuje fakt, iż w wielu przypadkach (wiersz, dla którego względna jakość rozwiązań wynosi 90% – 100 %) algorytm IWO wygenerował plan realizacji zapytania, którego czas wykonania był kilka-kilkanaście razy krótszy niż w przypadku metody wbudowanej w system SQL Server 2008.



Rys. 39. Wykres ilustrujący zestawienie rezultatów badań zawartych w tabeli 8.

W analizie otrzymanych wyników posłużono się dwoma wielkościami: szacunkowym kosztem realizacji planu oraz czasem realizacji zapytania. Szacunkowy koszt realizacji planu jest wielkością bezwymiarową. Liczba ta jest generowana przez optymalizator wbudowany w system Microsoft SQL Server 2008 i jest podstawą podejmowanych przez ten optymalizator decyzji. Odzwierciedla ona szacowany czas wykonania zapytania na podstawie dostar-

czonego planu. Należy zwrócić uwagę na fakt, iż liczba ta jest wyznaczana przed przystąpieniem do realizacji zapytania, w związku z czym może być obarczona dużym błędem. Czas realizacji zapytania jest z kolei rzeczywistym czasem, który upływa od chwili wysłania zapytania do systemu zarządzania bazą danych do momentu otrzymania wyniku końcowego.

Na podstawie wyników szczegółowych zawartych w załączniku A.3 (tabele 21-30) można wyciągnąć następujące wnioski:

1. W 59 na 90 przypadków (65,56%) szacunkowy koszt realizacji planu skonstruowanego przez system SQL Server 2008 jest niższy o ponad 25% od średniego szacunkowego kosztu realizacji planu wygenerowanego przez algorytm exIWO. W 17 na 59 przypadków relacja ta została zachowana dla czasów realizacji zapytania (czas wykonania zapytania na podstawie planu wygenerowanego przez SQL Server był krótszy niż czas wykonania zapytania dla planu wygenerowanego przez algorytm IWO). Jednak dysproporcja pomiędzy czasami realizacji zapytania była zdecydowanie mniejsza niż między szacowanymi kosztami realizacji planów. W pozostałych 42 sytuacjach czasy realizacji zapytania przy użyciu planu zbudowanego na podstawie algorytmu IWO okazały się krótsze niż na podstawie metody wbudowanej w system SQL Server.
2. W 9 na 90 przypadków (10%) szacunkowy koszt realizacji planu skonstruowanego przez system SQL Server 2008 był nieco niższy (o mniej niż 25%) niż średni koszt realizacji planów wygenerowanych przez algorytm IWO. Jednak w 6 przypadkach okazało się, że relacja między czasami realizacji zapytania testowego jest odwrotna (czas wykonania zapytania na podstawie planu wyznaczonego przez algorytm IWO był krótszy niż czas wykonania zapytania przy użyciu planu pochodzącego z metody wbudowanej).
3. Szacunkowy koszt realizacji planu skonstruowanego przez system SQL Server 2008 był trochę wyższy w 5 na 90 przypadków (5,55%) niż średni szacowany koszt realizacji planów wygenerowanych przez algorytm IWO. Zostało to potwierdzone w 3 przypadkach przez porównanie czasów realizacji zapytania testowego według każdego z planów.
4. Szacunkowy koszt realizacji planu skonstruowanego przez system SQL Server 2008 był wyższy o ponad 25% niż średni szacunkowy koszt realizacji planów wygenerowanych przez algorytm IWO w 17 na 90 przypadków (18,89%). W 5 sytuacjach tak duża dysproporcja została potwierdzona czasami realizacji zapytań (czas realizacji zapytania na podstawie IWO okazał się znacznie krótszy), w kolejnych 2 przypadkach dysproporcja nie była tak znaczna, w pozostałych 9 – relacja między czasami realizacji zapytania testowego według każdego z planów okazała się odwrotna (wykonanie zapytania przy użyciu algorytmu IWO było wolniejsze w stosunku do realizacji zapytania na podstawie metody wbudowanej).

Kompleksowo przeprowadzone badania dotyczące konstruowania planów realizacji zapytań dowodzą, że w bardzo wielu przypadkach algorytm IWO generuje plany lepsze od optymalizatora wbudowanego w system Microsoft SQL Server 2008. W 58 na 90 przypadków czas realizacji zapytania opartego na planie stworzonym przez IWO jest krótszy.

Dla 27 z 30 różnych baz danych z grafem złączeń o charakterze gwiazdy otrzymano krótszy czas realizacji zapytania (268 na 300 eksperymentów) używając planów wyznaczonych przez autorski, zmodyfikowany algorytm IWO. Można wysnuć wniosek, iż dla baz danych z grafem złączeń o charakterze gwiazdy exIWO wyraźnie przewyższa metodę wbudowaną w optymalizator kosztowy MS SQL Server 2008.

Zastanawiający jest fakt bardzo dużej dysproporcji pomiędzy szacunkowymi kosztami realizacji planów: wyznaczonego przez system SQL Server, a wygenerowanego przez exIWO. W większości przypadków szacunkowy koszt realizacji planu jest niższy o ponad 25% dla planu skonstruowanego przez metodę wbudowaną w SQL Server 2008. Jednak ta tendencja nie została potwierdzona przez rzeczywiste czasy realizacji zapytań. Z badań wynika, iż wbudowany optymalizator niepoprawnie szacuje liczbę wierszy biorących udział w złączeniach. Prowadzi to do błędnego oszacowania kosztu realizacji zapytania. W tym też miejscu upatruje się możliwości znacznej poprawy optymalizatora systemu Microsoft SQL Server 2008.

6.4. Podsumowanie przeprowadzonych badań

Przeprowadzone serie badań wykazują, że zmodyfikowana wersja algorytmu IWO sprawdza się w optymalizacji wielu problemów zarówno klasycznych, takich jak wyznaczanie minimum globalnego funkcji wielowymiarowych oraz symetrycznego i asymetrycznego komiwojażera, jak i bardzo złożonych.

Wykazano, iż algorytm exIWO, dzięki większym możliwościom dopasowania do rozwiązywanego problemu, często znajduje rozwiązania lepszej jakości niż jego oryginalny odpowiednik. Może ona konkurować z wieloma uznanymi algorytmami ewolucyjnymi oraz innymi metodami optymalizacyjnymi. Radzi sobie również z zestawami danych o ogromnej liczbie możliwych rozwiązań.

Autorska wersja IWO w niektórych przypadkach może z powodzeniem rywalizować z systemem komercyjnym w optymalizacji planów realizacji złączeń zapytań bazodanowych.

Można przypuszczać, że dzięki swoim atutom potencjalne zastosowania zmodyfikowanej wersji algorytmu IWO są bardzo szerokie.

7. Wnioski końcowe

Rozprawa dotyczy opracowania ogólnej metody optymalizacji, która może być wykorzystana do optymalizacji zapytań w bazach danych. Celem rozprawy było opracowanie metody wyznaczenia kolejności realizacji operacji złączeń, wykorzystującej zmodyfikowany algorytm *Invasive Weed Optimization* (exIWO).

Osiągnięcie tak postawionego celu wymagało zarówno dokładnego zapoznania się ze znanymi rozwiązaniami, jak i przeprowadzenia szczegółowej analizy algorytmu IWO. Czynniki te dały podstawę do opracowania metody autorskiej.

Realizacja celu postawionego w pracy przebiegała w kilku etapach. W pierwszym szczególny nacisk położono na sformułowanie podstawowych pojęć związanych z optymalizacją zapytania oraz na analizę całego procesu optymalizacji w systemie zarządzania bazą danych. Definicje pojęć i opis procesu zawarto w rozdziale drugim. Na podstawie dokonanej analizy sformułowano problem wyznaczenia kolejności złączeń w realizacji zapytań bazodanowych, który jest jednym z podstawowych zagadnień zadania optymalizacji zapytania. Jest to zarazem główny problem badawczy rozprawy. Wzięto pod uwagę również dwa klasyczne problemy optymalizacyjne: znajdowanie minimum funkcji wielowymiarowej (problem o charakterze ciągłym) i problem komiwojażera (problem o charakterze dyskretnym). Celem, który przyświecał wyborowi dwóch dodatkowych zagadnień, było dogłębne zbadanie opracowanych metod i strategii optymalizacyjnych. Charakterystyka wszystkich trzech wspomnianych zadań stanowi treść rozdziału trzeciego.

Bardzo dobre wyniki badań, jakie otrzymano stosując algorytm *Invasive Weed Optimization* do rozwiązania różnych problemów optymalizacyjnych, stanowiły dla autora niniejszej pracy bodziec do podjęcia próby rozwiązania przy jego użyciu problemu wyznaczenia optymalnej kolejności złączeń. Dokładny opis IWO, ogólną analizę jego złożoności obliczeniowej oraz przegląd innych algorytmów optymalizacyjnych o charakterze ewolucyjnym zamieszczono w rozdziale czwartym.

Efektem prowadzonych prac stała się zmodyfikowana wersja algorytmu IWO. Metodę tę przystosowano do rozwiązywania trzech zadań optymalizacyjnych (problem wyznaczania kolejności realizacji złączeń, znajdowanie minimum funkcji wielowymiarowej oraz problem komiwojażera). Założenia i modyfikacje opracowanych strategii przedstawiono w rozdziale piątym. Rozdział szósty natomiast zawiera wyniki przeprowadzonych eksperymentów, ich analizę oraz podsumowanie wszystkich badań.

Opracowanie nowych strategii optymalizacyjnych pozwoliło na sformułowanie następujących wniosków:

1. Autorska wersja IWO pozwala na optymalizację planów realizacji złączeń zapytań bazodanowych.
2. Opracowana strategia optymalizacyjna pozwala na znalezienie bardzo dobrych wyników w przypadku baz danych z grafami złączeń o charakterze gwiazdy. W tym przypadku przygotowana przez autora metoda może konkurować z komercyjnym systemem zarządzania bazą danych.
3. Zmodyfikowany algorytm IWO znajduje bardzo dobre rozwiązania dla klasycznych problemów optymalizacyjnych (wyznaczanie minimum globalnego funkcji wielowymiarowej oraz symetrycznego i asymetrycznego problemu komiwojażera).
4. Złożoność obliczeniowa opracowanej strategii jest wyższa niż złożoność oryginalnej wersji IWO, co jest rekompensowane jakością znajdowanych rozwiązań. Dzięki temu algorytm exIWO często znajduje rozwiązania lepszej jakości niż jego oryginalny odpowiednik, co potwierdzono z podrozdziale 6.1.4.
5. Zmodyfikowana wersja algorytmu IWO może konkurować z innymi algorytmami ewolucyjnymi oraz innymi metodami optymalizacyjnymi (np. sieciami samoorganizującymi, wyniki takiego porównania zamieszczono w podrozdziale 6.2.5.).
6. ExIWO może znaleźć zastosowanie dla zestawów danych o ogromnej liczbie możliwych rozwiązań (dla symetrycznego zadania komiwojażera zestaw *mona-lisa* o 100 000 lokalizacji).

Do autorskich oryginalnych wyników uzyskanych w rozprawie można zaliczyć:

1. Stworzenie zmodyfikowanej wersji algorytmu IWO:
   a) opracowanie mieszanej metody rozpraszania ziaren wokół chwastu macierzystego (strategie: rozwiewania, rozsiewania i staczania) oraz możliwość zastosowania rozkładu *t*-Studenta dla rozwiewania;
   b) użycie trzech metod selekcji chwastów do kolejnego pokolenia (wybór ze wszystkich osobników, wybór z osobników potomnych, wybór w ramach jednej rodziny);
   c) zastosowanie różnych kryteriów zatrzymania algorytmu.
2. Wykorzystanie zmodyfikowanego IWO do rozwiązania problemu wyznaczenia kolejności złączeń dla zapytań bazodanowych.
3. Określenie złożoności obliczeniowej dla algorytmu IWO zarówno w wersji podstawowej, jak i zmodyfikowanej oraz analiza złożoności metody w obu wersjach dla problemu określania kolejności złączeń.
4. Zastosowanie opracowanych metod do problemów znajdowania minimum funkcji wielowymiarowych i komiwojażera (nawet dla bardzo dużych zestawów danych sięgających 100 000 lokalizacji).

Należy nadmienić, iż część wyników uzyskanych przez autora dla różnych problemów optymalizacyjnych została opublikowana w następujących pracach:

- dla problemu określania kolejności złączeń [82, 83, 85, 86, 89, 90],
- dla znajdowania minimum funkcji wielowymiarowych [66, 87],
- dla problemu komiwojażera [81, 88, 91].

Uzyskane wyniki pozwalają na stwierdzenie, że cel pracy został osiągnięty, a teza pracy udowodniona.

Efektem przeprowadzonych badań było także stworzenie kilku programów komputerowych, co dało możliwość weryfikacji opracowanych metod i składowych algorytmu IWO. Z ich pomocą wyznaczono wartości numeryczne dla przygotowanych wcześniej scenariuszy testowych. Jest to pierwszy krok w planie dalszych badań autora, a zaimplementowane aplikacje stanowią bazę dla dalszych prac. Kolejne zamierzenia dotyczą następujących zagadnień:

1. Rozszerzenie exIWO w zakresie dodania kolejnych algorytmów złączeń. Planuje się dodanie metody z użyciem funkcji haszującej (ang. *hash join*) oraz strategii sortowania i złączenia (ang. *sort merge*).
2. Rozszerzenie możliwości algorytmu IWO o prowadzenie optymalizacji dla baz danych rozproszonych (wymaga to określenia innej reprezentacji rozwiązania). Warto nadmienić, że prace nad tym zagadnieniem zostały rozpoczęte, a osiągnięte rezultaty zaprezentowano w [90, 85].
3. Testowanie różnych modeli obliczeniowych służących do oszacowania czasu realizacji zadań złączenia i przesyłu danych. Wstępne prace dotyczące szacowania selektywności zapytań, w których autor pracy brał udział, zamieszczono w [7].
4. Wykorzystanie innych operatorów transformujących osobniki (np. MSFX).
5. Użycie różnych tzw. miękkich metod selekcji osobników. Wstępne prace i wyniki eksperymentów dotyczących zastosowania algorytmu symulowanego wyżarzania zamieszczono w [81].
6. Stworzenie algorytmu hybrydowego łączącego zalety algorytmu exIWO z algorytmem ewolucyjnym. Wcześniejsze badania dotyczące problemu określania kolejności realizacji złączeń przy użyciu algorytmu ewolucyjnego zostały przez autora niniejszej pracy zaprezentowane w [85, 84].
7. Zastosowanie zmodyfikowanego algorytmu IWO do selekcji cech. Technika ta polega na wyborze podzbioru cech z całego zbioru. Kryterium doboru składowych podzbioru jest skuteczność zadania klasyfikacji na podstawie wybranego podzbioru. Metoda ta posłuży rozwiązywaniu problemu rozpoznawania osób na podstawie ich sposobu poruszania się.

Jedną z głównych przyczyn hamujących rozwój tej dziedziny jest duża ilość danych, które należy przeanalizować. Pierwsze wyniki badań prowadzonych w tej dziedzinie zamieszczono w [66, 67]. Autor brał także udział w innych badaniach dotyczących redukcji liczby cech na potrzeby klasyfikacji osób na podstawie chodu ([68, 70, 65, 69]).

Bibliografia

1. Aarts E., Lenstra J. K. (eds.): Local search in combinatorial optimization. Princeton University Press, Princeton, New Jersey, USA 2003.
2. Aarts E., van Laarhoven P. J.: Simulated Annealing: An introduction. Statistica Neerlandica, Vol. 43, Issue 1, Wiley, 1989, s. 31-52.
3. Ackley D. H.: A connectionist machine for genetic hillclimbing. The Kluwer International Series in Engineering and Computer Science, Vol. 28, Kluwer Academic Publishers, Boston, MA, USA 1987.
4. Al Kasassbeh M., Alabadleh A., Al-Ramadeen T.: Shared crossover method for solving traveling salesman problem. International Journal of Information and Computer Science, Vol. 1, Issue 6, Science and Engineering Publishing Company, 2012, s. 153-158.
5. Applegate D. L., Bixby R. E., Chvátal V., Cook W. J.: The travelling salesman problem: A Computational Study. Princeton University Press, Princeton, New Jersey, USA 2006.
6. Arabas J.: Wykłady z algorytmów ewolucyjnych. Wydawnictwa Naukowo-Techniczne, Warszawa 2004.
7. Augustyn D. R., **Kostrzewa D.**: Szacowanie selektywności zapytań oparte na transformacie Hougha i metodzie PCA. Studia Informatica, Vol. 32, No. 2A(105), Wydawnictwo Politechniki Śląskiej, Gliwice 2012, str. 211-227.
8. Ausiello G., Crescenzi P., Gambosi G., Kann V., Marchetti-Spaccamela A., Protasi M.: Complexity and approximation: combinatorial optimization problems and their approximability properties. Springer-Verlag, Berlin Heidelberg 1999.
9. Bäck T., Fogel D., Michalewicz Z.: Handbook of evolutionary computation. IOP Publishing Ltd., Bristol, UK 1997.
10. Bai Y., Zhang W., Jin Z.: An self-organizing maps strategy for solving the travelling salesman problem. Chaos, Solitons and Fractals, Vol. 28, Elsevier, 2006, s. 1082-1089.
11. Barcsák C., Jármai K.: Benchmark for testing evolutionary algorithms. 10th World Conference on Structural and Multidisciplinary Optimization, Orlando, Florida, USA 2013.
12. Bazylevych R., Dupas R., Prasad B., Kuz B., Kutelmakh R., Bazylevych L.: A parallel approach for solving a large-scale traveling salesman problem. Proceedings of the Fifth Indian International Conference on Artificial Intelligence, 2011, s. 566-579.
13. Beaulieu A.: Learning SQL, 2nd edition. O'Reilly, USA 2009.
14. Beynon-Davies P.: Database systems, third edition. Palgrave Macmillan, Basingstoke, UK 2004.

15. Brocki Ł., Koržinek D.: Kohonen self-organizing map for the travelling salesperson problem. Recent Advances In Mechatronics, Part 1, Springer-Verlag, Berlin Heidelberg 2007, s. 116-119.
16. Černy V.: Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications, Vol. 45, No. 1, Springer, Kluwer Academic Publishers – Plenum Publishers, 1985, s. 41-51.
17. Chamberlin D. D., Boyce R. F.: Sequel: a structured English query language. Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control (Association for Computing Machinery), 1974, s. 249-264.
18. Chambers L. D.: The practical handbook of genetic algorithms: applications. Second edition. CRC Press, Inc., Boca Raton, FL, USA 2000.
19. Chaudhuri S.: An overview of query optimization in relational systems. Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, ACM, New York, USA 1998, s. 34-43.
20. Chelouah R., Siarry P.: Tabu search applied to global optimization. European Journal of Operational Research, Vol. 123, Issue 2, Elsevier, 2000, s. 256-270.
21. Chen Y.: A systematic metod for query evaluation in distributed heterogeneous databases. Journal of Information Science and Engineering, Vol. 16, 2000, s. 463-497.
22. Codd E. F.: A relational model of data for large shared data banks. Communications of ACM, Vol. 13, No. 6, ACM, 1970, s. 377-387.
23. Codd E. F.: Relational completeness of data base sublanguages. IBM Research Report, IBM Research Laboratory, San Jose, California, USA 1972.
24. Coello C. A., Lamont G. B., Van Veldhuizen D. A.: Evolutionary algorithms for solving multi-objective problems, second edition. Genetic and Evolutionary Computation, Springer, New York, NY, USA 2007.
25. Colomi A., Dorigo M., Maniezzo V.: Distributed optimization by ant colonies. Proceeding of European Conference on Artificial Life ’91, Elsevier, Paris, France 1991, s. 134-142.
26. Cormen T. H., Leiserson C. E., Rivest R. L.: Wprowadzenie do algorytmów. Wydawnictwa Naukowo-Techniczne, Warszawa 2005.
27. Czarnas P., Czech Z. J., Gocyła P.: Parallel simulated annealing for bicriterion optimization problems. Parallel Processing and Applied Mathematics, LNCS, Vol. 3019, Springer, Berlin, Heidelberg 2004, s. 233-240.
28. Date C. J.: Wprowadzenie do systemów baz danych. Wydawnictwa Naukowo-Techniczne, Warszawa 2000.
29. De Jong K. A.: Analysis of the behavior of a class of genetic adaptive systems, PhD dissertation. University of Michigan, USA 1975.

30. Delaney K., Randal P. S., Tripp K. L., Cunningham C., Mechanic A., Navarez B.: Microsoft SQL Server 2008 Internals. Microsoft Press, Redmond, Washington, USA 2009.
31. Delobel C., Adiba M.: Relational database systems. Elsevier, New York, NY, USA 1985.
32. DePuy G. W., Moraga R. J., Whitehouse G. E.: Meta-RaPS: a simple and effective approach for solving the travelling salesman problem. Transportation Research, Part E, Vol .41, Elsevier, 2005, s. 115-130.
33. Deshpande A., Ives Z., Raman V.: Adaptive query processing. Foundations and Trends in Databases, Vol. 1, Issue 1, Now Publishers Inc., Hanover, MA, USA 2007, s. 1-140.
34. Dong H., Liang Y.: Genetic algorithms for large join query optimization. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation GECCO '07, ACM, New York, NY, USA 2007, s. 1211-1218.
35. Dorigo M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, Vol. 1, Issue 1, IEEE, 1997, s. 53-66.
36. Dorigo M.: Optimization, learning and natural algorithms. PhD thesis, Politechnico di Milano, Italie 1992.
37. Eberhart R. C., Shi Y.: Comparing inertia weights and constriction factors in particle swarm optimization. IEEE Proceedings of the 2000 Congress on Evolutionary Computation, Vol. 1, IEEE, La Jolla, CA, USA 2000, s. 84-88.
38. Elmasri R., Navathe S. B.: Fundamentals of database systems, 6th edition. Addison-Wesley, Boston, MA, USA 2011.
39. Evrendilek C., Dogac A., Nural S., Ozcan F.: Multidatabase query optimization. Distributed and Parallel Databases, Vol. 5, Issue 1, Kluwer Academic Publishers, Hingham, MA, USA 1997, s. 77-114.
40. Ferreira C.: Combinatorial optimization by gene expression programming: inversion revisited. Proceedings of the Argentine Symposium on Artificial Intelligence, Santa Fe, Argentina 2002, s. 160-174.
41. Finnila A. B., Gomez M. A., Sebenik C., Stenson C., Doll J. D.: Quantum annealing: a new method for minimizing multidimensional functions. Chemical Physics Letters, Vol. 219, Issue 5-6, Elsevier, 1994, s. 343-348.
42. Gajek L., Kałuszka M.: Wnioskowanie statystyczne dla studentów. Modele i metody. WNT, Warszawa 1999.
43. Galindo-Legaria C., Pellenkoft A., Kersten M.: Fast, randomized join-order selection – Why use transformations? Proceedings of the 20th VLDB Conference, Santiago 1994.

44. Garcia-Molina H., Ullman J. D., Widom J.: Database system implementation, second edition. Prentice Hall, New Jersey, USA 2002.
45. Garcia-Molina H., Ullman J. D., Widom J.: Systemy baz danych pełny wykład. Wydawnictwa Naukowo-Techniczne, Warszawa 2006.
46. Garey M. R., Johnson D. S.: Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., New York, NY, USA 1990.
47. Gendreau M., Hertz A., Laporte G.: A tabu search heuristic for the vehicle routing problem. Management Science, Vol. 40, No. 10, INFORMS, Hanover, Maryland, USA 1994, s. 1276-1290.
48. Ghalenoei M. R., Hajimirsadeghi H., Lucas C.: Discrete invasive weed optimization algorithm: Application to cooperative multiple task assignment of UAVs. IEEE Conference on Decision and Control, IEEE, Shanghai 2009, s. 1665-1670.
49. Giri R., Chowdhury A., Ghosh A., Das S., Abraham A., Snasel V.: A modified invasive weed optimization algorithm for training of feed-forward neural networks. IEEE International Conference on Systems Man and Cybernetics 2010, IEEE, Istanbul 2010, s. 3166-3173.
50. Glover F.: Tabu search – part I. ORSA Journal on Computing, Vol. 1, No. 3, Operations Research of America, USA 1989, s. 190-206.
51. Glover F.: Tabu search – part II. ORSA Journal on Computing, Vol. 2, No. 1, Operations Research of America, USA 1990, s. 4-32.
52. Główny Urząd Statystyczny: Rocznik Statystyczny Rzeczypospolitej Polskiej 2011. GUS, Warszawa 2011.
53. Graefe G.: Query evaluation techniques for large databases. ACM Computing Surveys, Vol. 25, Issue 2, ACM, New York, NY, USA 1993, s. 73-169.
54. Greco F. (ed.): Travelling salesman problem. In-Teh, Croatia 2008.
55. Griewank A. O.: Generalized descent for global optimization. Journal of Optimization Theory and Applications, Vol. 34, No. 1, Plenum Publishing Corporation, 1981, s. 11-39.
56. Grzymkowski R.: Matematyka dla studentów wyższych uczelni technicznych. Pracownia Komputerowa Jacka Skalmierskiego, Gliwice 2000.
57. Gutin G., Punnen A. (eds.): The traveling salesman problem and its variations. Kluwer Academic Publishers, Boston / Dordrecht / London 2002.
58. Hedar A.-R., Fukushima M.: Tabu search directed by direct search methods for nonlinear global optimization. European Journal of Operational Research, Vol. 170, Issue 2, Elsevier, 2006, s. 329-349.

59. Helshaun K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research, Vol. 126, Issue 1, Elsevier, 2000, s. 106-130.
60. Ibaraki T.: On the optimal nesting order for computing N-relational joins. ACM Transactions on Database Systems, Vol. 9, Issue 3, ACM, New York, NY, USA 1984, s. 482-502.
61. Ioannidis Y. E.: Query optimization. ACM Computing Surveys, Vol. 28, Issue 1, ACM, New York, USA 1996, s. 121-123.
62. Ioannidis Y. E., Kang Y. C.: Left-deep vs. bushy trees: an analysis of strategy spaces and its implications for query optimization. Proceedings of the ACM SIGMOD Conference on Management of Data, Denver, USA 1991, s. 168-177.
63. Ioannidis Y. E., Kang Y. C.: Randomized algorithms for optimizing large join queries. Proceedings of the ACM SIGMOD Conference on Management Data, Atlantic City, NJ, USA 1990, s. 312-321.
64. Jarke M., Koch J.: Query optimization in database systems. ACM Computing Surveys, Vol. 16, Issue 2, ACM, New York, USA 1984, s. 111-152.
65. Josiński H., **Kostrzewa D.**, Michalczuk A., Świtoński A.: The exIWO metaheuristic for solving continuous and discrete optimization problems. The Scientific World Journal (**Impact Factor 1.730**), Vol. 2014, Article ID 831691, Hindawi Publishing Corporation, 2014.
66. Josiński H., **Kostrzewa D.**, Michalczuk A., Świtoński A., Wojciechowski K.: Feature Extraction and HMM-Based Classification of Gait Video Sequences for the Purpose of Human Indentification. Vision Based Systems for UAV Applications, SCI 481, Springer, Switzerland 2013.
67. Josiński H., Michalczuk A., **Kostrzewa D.**, Świtoński A., Wojciechowski K.: Heuristic method of feature selection for person reindentification based on gait motion capture data. Intelligent Information and Database Systems, Lecture Notes in Computer Science, Vol. 8398, Springer, 2014, s. 585-594.
68. Josiński H., Świtoński A., Jędrasiak K., **Kostrzewa D.**: Human Identification Based on Gait Motion Capture Data. Lecture Notes in Engineering and Computer Science, Vol. 2195, Hong Kong 2012, str. 507-510.
69. Josiński H., Świtoński A., Jędrasiak K., **Kostrzewa D.**: Human Identification Based on Tensor Representation of the Gait Motion Capture Data. IAENG Transactions on Electrical Engineering, Vol. 1 – Special Issue of the International MultiConference of Engineers and Computer Scientists 2012, World Scientific Publishing 2013, s. 111-122.

70. Josiński H., Świtoński A., Michalczuk A., **Kostrzewa D.**, Wojciechowski K.: Human Identification Based on Gait Video Sequences. International Conference on Computer Science and Engineering (ICCSE 2013), World Academy of Science, Engineering and Technology (WASET), Kuala Lumpur, 14-15.02.2013, s. 312-317.
71. Josiński H.: Model realizacji zapytania dla danych rozproszonych. Wysokowydajne sieci komputerowe, Tom 1, WKiŁ, Warszawa 2005.
72. Kang R.-G., Jung C.-Y.: The optimal solution of TSP using the new mixture initialization and sequential transformation method in genetic algorithm. Lecture Notes in Artificial Intelligence, Vol. 4099, Springer-Verlag, Berlin Heidelberg 2006, s. 1181-1185.
73. Karaboga D., Basturk B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, Vol. 39, No. 3, Springer Netherlands, 2007, s. 459-471.
74. Karimkashi S., Kishk A. A.: Invasive weed optimization and its features in electromagnetics. IEEE Transactions on antennas and propagation, Vol. 58, No. 4. IEEE, 2010, s. 1269-1278.
75. Kennedy J.: Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. IEEE, Proceedings of the 1999 Congress on Evolutionary Computation, Vol. 3, IEEE, 1999, s. 1931-1938.
76. Kennedy J., Eberhart R.: Particle swarm optimization. 1995. Proceedings IEEE International Conference on Neural Networks, Vol 4, IEEE, 1995, s. 1942-1948.
77. Kirkpatrick S., Gelatt C. D., Vecchi M.P.: Optimization by simulated annealing. Science, Vol. 220, No. 4598, USA 1983, s. 671-680.
78. Kohonen T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics, Vol. 43, Issue 1, Springer-Verlag, 1982, s. 59-69.
79. Kohonen T.: The self-organizing map. Neurocomputing, Vol. 21, Elsevier, 1998, s. 1-6.
80. Kohonen T.: The self-organizing map. Proceedings of the IEEE, Vol. 78, Issue 9, IEEE, 1990, s. 1464-1680.
81. **Kostrzewa D.**, Josiński H.: Evaluation of the exIWO Algorithm Based on the Travelling Salesman Problem. International Journal of Mathematical, Computational, Natural and Physical Engineering, Vol. 8, No. 9, WASET 2014, International Conference on Computing and Artificial Intelligence. Singapore 2014, s. 1175-1178.
82. **Kostrzewa D.**, Josiński H.: Metody przeszukiwania przestrzeni planów realizacji zapytań za pomocą algorytmu IWO. Studia Informatica, Vol. 31, No. 2A(89). Wydawnictwo Politechniki Śląskiej, Gliwice 2010.
83. **Kostrzewa D.**, Josiński H.: Ocena jakości strategii eksploracji przestrzeni poszukiwań dla problemu określenia kolejności realizacji złączeń. Studia Informatica, Vol. 32, No. 2A (96). Wydawnictwo Politechniki Śląskiej, Gliwice, 2011, s. 37-46.

84. **Kostrzewa D.**, Josiński H.: Planowanie procesu scalania danych rozproszonych za pomocą algorytmu ewolucyjnego. Bazy danych: Rozwój metod i technologii, Tom 1, WKiŁ, Warszawa 2008.
85. **Kostrzewa D.**, Josiński H.: The comparison of an adapted evolutionary algorithm with the invasive weed optimization algorithm based on the problem of predetermining progress of distributed data merging process. Advances in Intelligent and Soft Computing: Man-Machine Interactions, Springer-Verlag, Berlin Heidelberg 2009.
86. **Kostrzewa D.**, Josiński H.: The exIWO metaheuristic – a recapitulation of the research on the join ordering problem. Communications in Computer and Information Science, Vol. 424, Springer, 2014, s. 10-19.
87. **Kostrzewa D.**, Josiński H.: The modified IWO algorithm for optimization of numerical functions. Lecture Notes in Computer Science, Vol. 7269, Springer-Verlag, Berlin Heidelberg 2012, s. 267-274.
88. **Kostrzewa D.**, Josiński H.: Using the Expanded IWO Algorithm to Solve the Traveling Salesman Problem. Proceedings of the 5th International Conference on Agents and Artificial Intelligence Barcelona 2013, Vol. 2. SCITEPRESS – Science and Technology Publications, Portugal 2013, str. 451-456.
89. **Kostrzewa D.**, Josiński H.: Verification of the search space exploration strategy based on the solutions of the join ordering problem. Advances in Intelligent and Soft Computing: Man-Machine Interactions 2, Springer-Verlag, Berlin Heidelberg 2011 s. 447-455.
90. **Kostrzewa D.**, Josiński H.: Zastosowanie algorytmu IWO do planowania procesu scalania danych rozproszonych. Studia Informatica, Vol. 30, No. 2A (83), Wydawnictwo Politechniki Śląskiej, Gliwice 2009.
91. **Kostrzewa D.**, Josiński H.: Zastosowanie algorytmu IWO do rozwiązania problemu komiwojażera. Technologie Przetwarzania Danych, Wydawnictwa Naukowo-Techniczne, Warszawa 2010.
92. Lanzelotte R. S., Valduriez P., Zaït M.: On the effectiveness of optimization search strategies for parallel execution spaces. Proceedings of the 19th VLDB Conference, Dublin, Ireland 1993, s. 493-504.
93. Larrañaga P., Kuijpers C. M., Murga R. H., Inza I., Dizdarevic S.: Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review, Vol. 13, No. 2, Springer Netherlands, 1999, s. 129-170.
94. Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G., Shmoys D. B. (eds.): The travelling salesman problem a guided tour of combinatorial optimization. John Wiley & Sons, Chichester, UK 1985.

95. Liang J. J., Qin A. K., Suganthan P. N., Baskar S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation, Vol. 10, No. 3, IEEE, 2006, s. 281-295.
96. Liu L., Özsu M. T. (eds.): Encyclopedia of database systems. Springer Science+Business Media, New York, NY, USA 2009.
97. Mallahzadeh A. R., Es'haghi S., Hassani H. R.: Compact U-array MIMO antenna designs using IWO algorithm. International Journal of RF and Microwave Computer-Aided Engineering, Vol. 19, Issue 5. Wiley, 2009, s. 568-576.
98. Mallahzadeh A. R., Oraizi H., Davoodi-Rad Z.: Application of the invasive weed optimization technique for antenna configurations. Progress In Electromagnetics Research, Vol. 79. EMW Publishing, 2008, s. 137-150.
99. Mamaghani A. S., Asghari K., Mahmoudi F., Meybodi M. R.: A novel hybrid algorithm for join ordering problem in database queries. Proceedings of the 6th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics, Tenerife, Spain 2007, s. 104-109.
100. Matysiak M.: Efficient optimization of large join queries using tabu search. Information Sciences, Vol. 83, Issue 1-2, Elsevier, 1995, s. 77-88.
101. Mehrabian A. R., Lucas C.: A novel numerical optimization algorithm inspired from weed colonization. Ecological Informatics, Vol. 1, Issue 4. Elsevier, 2006, s. 355-366.
102. Merz P., Freisleben B.: Memetic algorithms for the travelling salesman problem. Complex Systems, Vol. 13, Complex Systems Publications, 2001, s. 297-345.
103. Michalewicz Z.: Algorytmy genetyczne + struktury danych = programy ewolucyjne. WNT, Warszawa 2003.
104. Michalewicz Z., Fogel D. B.: Jak to rozwiązać, czyli nowoczesna heurystyka. WNT, Warszawa 2006.
105. Moerkotte G., Neumann T.: Analysis of two existing and one new dynamic programming algorithm for the generation of optimal bushy join trees without cross products. Proceedings of the 32nd VLDB '06 Conference, Seoul, Korea 2006, s. 930-941.
106. Molga M., Smutnicki C.: Test functions for optimization needs. http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf, 2005.
107. Mona Lisa TSP Challenge. http://www.tsp.gatech.edu/data/ml/monalisa.html (data ostatniej wizyty 10 sierpnia 2012r.).
108. Morzy T., Matysiak M., Salza S.: Tabu search optimization of large join queries. [w:] Jarke M. i in. (red.): Proceedings of the 4th International Conference on Extending Database Technology, Cambridge 1994, s. 309-322.
109. Mühlenbein H., Schomisch M., Born J.: The parallel genetic algorithm as function optimizer. Parallel Computing, Vol. 17, Issue 6-7, Elsevier, 1991, s. 619-632.

110. Nemhauser G. L., Wolsey L. A.: Integer and combinatorial optimization. John Wiley & Sons, USA 1999.
111. Osman I. H., Kelly J. P.: Meta-heuristics: theory & applications. Kluwer Academic Publishers, Norwell, MA, USA 1996.
112. Pahlavani P., Delavar M. R., Frank A. U.: Using a modified invasive weed optimization algorithm for a personalized urban multi-criteria path optimization problem. International Journal of Applied Earth Observation and Geoinformation, Vol. 18, Elsevier, 2012, s. 313-328.
113. Papadimitriou C. H., Yannakakis M.: Optimization, approximation, and complexity classes. Journal of Computer and System Sciences, Vol. 43, Issue 3, Elsevier 1991, s. 425-440.
114. Pawlak M.: Algorytmy ewolucyjne jako narzędzie do harmonogramowania produkcji. Wydawnictwo Naukowe PWN, Warszawa 1999.
115. Poli R., Kennedy J., Blackwell T.: Particle swarm optimization: An overview. Swarm Intelligence, Vol. 1, No. 1, Springer, New York, USA 2007, s. 33-57.
116. Potter M. A., De Jong K. A.: A cooperative coevolutionary approach to function optimization. Lecture Notes in Computer Science, Vol. 866, Springer-Verlag, London 1994, s. 249-257.
117. Purcell T.: Evolution of star join optimization DB2 UDB for z/OS and OS/390. White Paper, Yevich, Lawson & Associates, 2002, ftp://ftp.software.ibm.com/software/data/db2/zos/presentations/performance/evolution-star-schema-optimization-whitepaper-2002-purcell.pdf.
118. Rastrigin L. A.: External control by the method of random scanning. Automatika i Telemechanika, Vol. 21, No. 9, ZSRR 1960, s. 1264-1271.
119. Reinelt G.: TSPLIB95. Institut für Angewandte Mathematik, Universität Heidelberg, Heidelberg, Niemcy 1995. http://comopt.ifi.uni-heidelberg.de/software/tsplib95/doc.ps (data ostatniej wizyty 5 sierpnia 2012r.).
120. Research Group Combinatorial Optimization, Ruprecht-Karls-Universität Heidelberg. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html (data ostatniej wizyty 5 sierpnia 2012r.).
121. Rosenbrock H. H.: An automatic method for finding the greatest or least value of a function. Computer Journal, Vol. 3, Issue 3, Oxford University Press, 1960, s. 175-184.
122. Sahraei-Ardakani M., Roshanaei M., Rahimi-Kian A., Lucas C.: A study of electricity market dynamics using invasive weed colonization optimization. IEEE Symposium on Computational Intelligence and Games 2008. IEEE, Perth, WA 2008, s. 276-282.

123. Scheufele W., Moerkotte G.: Optimal ordering of selections and joins in acyclic queries with expensive predicates. Interner Bericht, RWTH Aachen, Lehrstuhl für Informatik III, Aachen 1996.
124. Sellis T. K.: Multiple-query optimization. ACM Transactions on Database Systems, Vol. 13, Issue 1, ACM, New York, USA 1988, s. 23-52.
125. Sepehri H., Lucas C.: A recommender system based on invasive weed optimization algorithm. IEEE Congres on Evolutionary Computation 2007. IEEE, Singapore 2007, s. 4297-4304.
126. Shang Y.-W., Qiu Y.-H.: A note of the extended Rosenbrock function. Evolutionary Computation, Vol. 16, No. 1, MIT Press, 2006, s. 119-126.
127. Socha K., Dorigo M.: Ant colony optimization for continuous domains. European Journal of Operational Research, Vol. 185, Issue 3, Elsevier, 2008, s. 1155-1173.
128. Stacey A., Jancic M., Grudny I.: Particle swarm optimization with mutation. IEEE The 2003 Congress on Evolutionary Computation, Vol. 2, IEEE, 2003, s. 1425-1430.
129. Steinbrunn M., Moerkotte G., Kemper A.: Heuristic and randomized optimization for the join ordering problem. The VLDB Journal, Vol. 6, No. 3, 1997, s. 191-208.
130. Steinbrunn M., Moerkotte G., Kemper A.: Optimizing join orders. Technical Report MIP9307, Faculty of Mathematics, University of Passau, Germany 1993.
131. Stillger M., Spiliopoulou M.: Genetic programming in database query optimization. Proceedings of the 1st Annual Conference on Genetic Programming, GECCO '96, MIT Press Cambridge, MA, USA 1996, s. 388-393.
132. Subieta K.: Słownik terminów z zakresu obiektowości. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.
133. Swami A., Gupta A.: Optimization of large join queries. Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA 1988, s. 8-17.
134. Swami A., Iyer B. R.: A polynomial time algorithm for optimizing join queries. Proceedings of the 9th International Conference on Data Engineering, IEEE, 1993, s. 345-354.
135. Tao G., Michalewicz Z.: Inver-over operator for the TSP. Lecture Notes in Computer Science, Vol. 1498, Springer, Berlin Heidelberg 1998, s. 803-812.
136. Tow D.: SQL. Optymalizacja. Wydawnictwo Helion, Gliwice 2004.
137. Trojanowski K.: Metaheurystyki praktycznie. Wydawnictwo WIT, Warszawa 2005.
138. Ullman J. D., Widom J.: Podstawowy wykład z systemów baz danych. Wydawnictwa Naukowo-Techniczne, Warszawa 2001.

139. Waas F., Pellenkoft A.: Join order selection – good enough s easy. Proceedings of the 17th British National Conference on Databases: Advances in Databases, BNCOD 17, Springer-Verlag, London, UK 2000, s. 51-67.
140. Wrembel R., Bębel B.: Oracle. Projektowanie rozproszonych baz danych. Helion, Gliwice 2003.
141. Wróblewski J.: Niekonwencjonalne metody obliczeniowe. Algorytmy genetyczne – ćwiczenia. Polsko-Japońska Wyższa Szkoła Technik Komputerowych, Warszawa 1996.
142. Wu S., Li F., Mehrotra S., Ooi B. C.: Query optimization for massively parallel data processing. Proceedings of the 2nd ACM Symposium on Cloud Computing, ACM, New York, USA 2011.
143. Xie X.-F., Zhang W.-J., Yang Z.-L.: Adaptive particle swarm optimization on individual level. 6th International Conference on Signal Processing, Vol. 2. IEEE, China 2002, s. 1215-1218.
144. Yu C., Meng W.: Principles of database query processing for advanced applications. Morgan Kaufmann Publishers, San Francisco, CA, USA 1998.
145. Yu Y., Chen Y., Li T.: A new design of genetic algorithm for solving TSP. Fourth International Joint Conference on Computational Sciences and Optimization, IEEE, Yunnan, China 2011, s. 309-313.
146. Zhang X., Wang Y., Cui G., Niu Y., Xu J.: Application of a novel IWO to the design of encoding sequences for DNA computing. International Journal of Computers & Mathematics with Applications, Vol. 57, Elsevier, 2009, s. 2001-2008.
147. Zhao P., Han J.: On graph query optimization in large networks. Proceedings of the VLDB Endowment, Vol. 3, Issue 1-2, VLDB Endowment, 2010, s. 340-351.
148. Zitzler E., Deb K., Thiele L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evolutionary Computation, Vol. 8, No. 2, MIT Press, USA 2000, s. 173-195.

Spis rysunków

Spis tabel

Dodatek A. Szczegółowe wyniki badań

Ze względu na dużą liczbę przeprowadzonych eksperymentów szczegółowe zestawienia wyników badań postanowiono zamieścić w niniejszym dodatku.

A.1. Problem znajdowania minimum funkcji wielowymiarowej

Podrozdział ten zawiera wyniki eksperymentów przeprowadzonych na potrzeby rozwiązania problemu znajdowania minimum funkcji wielowymiarowej.

A.1.1. Badania z użyciem wybranych funkcji testowych

Tabele 9-13 zawierają wyniki badań dla: pierwszej funkcji De Jonga, funkcji Griewanka, Rastrigina, Rosenbrocka i Ackleya. Testy przeprowadzono dla funkcji o 30 i 100 wymiarach, obliczenia powtórzono stukrotnie. Eksperymenty przeprowadzono dla typowych zakresów początkowych tych funkcji [106].

Tabela 9
Zestawienie wyników dla autorskiej, zmodyfikowanej wersji algorytmu IWO na podstawie pierwszej funkcji De Jonga

Liczba wymiarów	Kryterium stopu	Średni wynik	Odch. standardowe	Min. wynik	Maks. wynik	Śr. czas obliczeń	Śr. liczba iteracji
30 wymiarów	30 sekund	$6{,}21\cdot10^{-14}$	$1{,}08\cdot10^{-14}$	$4{,}56\cdot10^{-14}$	$8{,}89\cdot10^{-14}$	30	251784
	20 sekund	$1{,}38\cdot10^{-13}$	$2{,}80\cdot10^{-14}$	$9{,}06\cdot10^{-14}$	$1{,}88\cdot10^{-13}$	20	162405,3
	15 sekund	$2{,}43\cdot10^{-13}$	$4{,}35\cdot10^{-14}$	$1{,}77\cdot10^{-13}$	$2{,}93\cdot10^{-13}$	15	129397,7
	10 sekund	$5{,}80\cdot10^{-13}$	$1{,}72\cdot10^{-13}$	$3{,}30\cdot10^{-13}$	$9{,}08\cdot10^{-13}$	10	80210
	5 sekund	$2{,}23\cdot10^{-12}$	$5{,}91\cdot10^{-13}$	$1{,}31\cdot10^{-12}$	$3{,}10\cdot10^{-12}$	5	41055,2
	2 sekundy	$1{,}66\cdot10^{-11}$	$4{,}33\cdot10^{-12}$	$1{,}23\cdot10^{-11}$	$2{,}65\cdot10^{-11}$	2	15144,9
	1 sekunda	$1{,}43\cdot10^{-10}$	$5{,}28\cdot10^{-11}$	$8{,}40\cdot10^{-11}$	$2{,}42\cdot10^{-10}$	1	6382,7
	0,5 sekundy	$1{,}20\cdot10^{-3}$	$3{,}00\cdot10^{-4}$	$6{,}27\cdot10^{-4}$	$1{,}73\cdot10^{-3}$	0,5	3647
	25000 iteracji	$1{,}88\cdot10^{-12}$	$3{,}82\cdot10^{-13}$	$1{,}32\cdot10^{-12}$	$2{,}76\cdot10^{-12}$	16,84	25000
	20000 iteracji	$3{,}15\cdot10^{-12}$	$6{,}21\cdot10^{-13}$	$2{,}03\cdot10^{-12}$	$4{,}20\cdot10^{-12}$	13,79	20000
	15000 iteracji	$5{,}17\cdot10^{-12}$	$1{,}55\cdot10^{-12}$	$3{,}69\cdot10^{-12}$	$9{,}27\cdot10^{-12}$	10,02	15000
	10000 iteracji	$1{,}31\cdot10^{-11}$	$2{,}94\cdot10^{-12}$	$9{,}27\cdot10^{-12}$	$1{,}98\cdot10^{-11}$	6,63	10000
	5000 iteracji	$4{,}70\cdot10^{-11}$	$8{,}80\cdot10^{-12}$	$3{,}10\cdot10^{-11}$	$6{,}26\cdot10^{-11}$	3,25	5000
	2000 iteracji	$3{,}80\cdot10^{-9}$	$5{,}86\cdot10^{-10}$	$2{,}72\cdot10^{-9}$	$4{,}52\cdot10^{-9}$	0,95	2000
	500 iteracji	$6{,}50\cdot10^{-8}$	$9{,}70\cdot10^{-9}$	$5{,}02\cdot10^{-8}$	$7{,}89\cdot10^{-8}$	0,38	500
	100 iteracji	$1{,}73\cdot10^{-2}$	$5{,}51\cdot10^{-3}$	$9{,}20\cdot10^{-3}$	$2{,}61\cdot10^{-2}$	0,15	100
100 wymiarów	30 sekund	$5{,}63\cdot10^{-11}$	$7{,}38\cdot10^{-12}$	$4{,}17\cdot10^{-11}$	$6{,}95\cdot10^{-11}$	30	93173,1
	20 sekund	$1{,}13\cdot10^{-10}$	$1{,}66\cdot10^{-11}$	$8{,}94\cdot10^{-11}$	$1{,}38\cdot10^{-10}$	20	61775,3
	15 sekund	$1{,}83\cdot10^{-10}$	$2{,}27\cdot10^{-11}$	$1{,}50\cdot10^{-10}$	$2{,}21\cdot10^{-10}$	15	47264,2
	10 sekund	$5{,}55\cdot10^{-10}$	$6{,}54\cdot10^{-11}$	$4{,}73\cdot10^{-10}$	$6{,}69\cdot10^{-10}$	10	30250,1
	5 sekund	$1{,}90\cdot10^{-9}$	$3{,}54\cdot10^{-10}$	$1{,}32\cdot10^{-9}$	$2{,}46\cdot10^{-9}$	5	15599
	2 sekundy	$1{,}58\cdot10^{-8}$	$2{,}62\cdot10^{-9}$	$1{,}21\cdot10^{-8}$	$1{,}98\cdot10^{-8}$	2	8914,3
	1 sekunda	$2{,}02\cdot10^{-7}$	$4{,}50\cdot10^{-8}$	$1{,}40\cdot10^{-7}$	$3{,}00\cdot10^{-7}$	1	4216,1
	0,5 sekundy	$4{,}05\cdot10^{-5}$	$2{,}05\cdot10^{-5}$	$2{,}07\cdot10^{-5}$	$9{,}54\cdot10^{-5}$	0,5	2030,8
	25000 iteracji	$3{,}59\cdot10^{-9}$	$5{,}04\cdot10^{-10}$	$2{,}79\cdot10^{-9}$	$4{,}55\cdot10^{-9}$	37,65	25000
	20000 iteracji	$6{,}01\cdot10^{-9}$	$8{,}93\cdot10^{-10}$	$4{,}53\cdot10^{-9}$	$7{,}67\cdot10^{-9}$	28,98	20000
	15000 iteracji	$1{,}15\cdot10^{-8}$	$1{,}60\cdot10^{-9}$	$8{,}64\cdot10^{-9}$	$1{,}36\cdot10^{-8}$	22,27	15000
	10000 iteracji	$2{,}49\cdot10^{-8}$	$2{,}69\cdot10^{-9}$	$1{,}98\cdot10^{-8}$	$2{,}86\cdot10^{-8}$	14,95	10000
	5000 iteracji	$1{,}29\cdot10^{-7}$	$1{,}08\cdot10^{-8}$	$1{,}16\cdot10^{-7}$	$1{,}48\cdot10^{-7}$	6,84	5000
	2000 iteracji	$1{,}27\cdot10^{-7}$	1,11·10-8	$1{,}06\cdot10^{-7}$	$1{,}46\cdot10^{-7}$	3,10	2000
	500 iteracji	$1{,}15\cdot10^{-2}$	$1{,}58\cdot10^{-3}$	$8{,}55\cdot10^{-3}$	$1{,}47\cdot10^{-2}$	0,91	500
	100 iteracji	$5{,}13\cdot10^{1}$	$5{,}34\cdot10^{0}$	$4{,}28\cdot10^{1}$	$5{,}81\cdot10^{1}$	0,51	100

Tabela 10
Zestawienie wyników dla autorskiej, zmodyfikowanej wersji algorytmu IWO na podstawie funkcji Griewanka

Liczba wymiarów	Kryterium stopu	Średni wynik	Odch. standardowe	Min. wynik	Maks. wynik	Śr. czas obliczeń	Śr. liczba iteracji
30 wymiarów	30 sekund	$1{,}42\cdot10^{-12}$	$3{,}40\cdot10^{-13}$	$9{,}38\cdot10^{-13}$	$1{,}94\cdot10^{-12}$	30	282886,2
	20 sekund	$2{,}87\cdot10^{-12}$	$4{,}80\cdot10^{-13}$	$2{,}07\cdot10^{-12}$	$3{,}49\cdot10^{-12}$	20	190793,9
	15 sekund	$7{,}27\cdot10^{-12}$	$1{,}37\cdot10^{-12}$	$4{,}92\cdot10^{-12}$	$9{,}13\cdot10^{-12}$	15	139777,1
	10 sekund	$1{,}68\cdot10^{-11}$	$5{,}05\cdot10^{-12}$	$7{,}74\cdot10^{-12}$	$2{,}68\cdot10^{-11}$	10	90076,9
	5 sekund	$5{,}07\cdot10^{-11}$	$2{,}00\cdot10^{-11}$	$3{,}36\cdot10^{-11}$	$1{,}03\cdot10^{-10}$	5	47797,8
	2 sekundy	$3{,}28\cdot10^{-10}$	$9{,}40\cdot10^{-11}$	$1{,}71\cdot10^{-10}$	$5{,}04\cdot10^{-10}$	2	17763,7
	1 sekunda	$1{,}82\cdot10^{-9}$	$8{,}35\cdot10^{-10}$	$9{,}15\cdot10^{-10}$	$3{,}01\cdot10^{-9}$	1	8388,7
	0,5 sekundy	$4{,}79\cdot10^{5}$	$1{,}80\cdot10^{5}$	$2{,}62\cdot10^{5}$	$8{,}72\cdot10^{5}$	0,5	3761,6
	25000 iteracji	$4{,}88\cdot10^{-11}$	$9{,}52\cdot10^{-12}$	$3{,}08\cdot10^{-11}$	$6{,}24\cdot10^{-11}$	24,38	25000
	20000 iteracji	$7{,}59\cdot10^{-11}$	$1{,}50\cdot10^{-11}$	$5{,}80\cdot10^{-11}$	$1{,}03\cdot10^{-10}$	20,31	20000
	15000 iteracji	$1{,}32\cdot10^{-10}$	$1{,}82\cdot10^{-11}$	$9{,}76\cdot10^{-11}$	$1{,}55\cdot10^{-10}$	14,76	15000
	10000 iteracji	$3{,}22\cdot10^{-10}$	$6{,}29\cdot10^{-11}$	$2{,}14\cdot10^{-10}$	$4{,}61\cdot10^{-10}$	9,84	10000
	5000 iteracji	$1{,}37\cdot10^{-9}$	$2{,}35\cdot10^{-10}$	$1{,}10\cdot10^{-9}$	$1{,}81\cdot10^{-9}$	4,91	5000
	2000 iteracji	$8{,}77\cdot10^{-9}$	$2{,}20\cdot10^{-9}$	$5{,}44\cdot10^{-9}$	$1{,}29\cdot10^{-8}$	1,77	2000
	500 iteracji	$2{,}68\cdot10^{-4}$	$8{,}57\cdot10^{-5}$	$1{,}24\cdot10^{-4}$	$4{,}35\cdot10^{-4}$	0,50	500
	100 iteracji	$8{,}47\cdot10^{2}$	$2{,}41\cdot10^{2}$	$5{,}36\cdot10^{2}$	$1{,}36\cdot10^{3}$	0,39	100
100 wymiarów	30 sekund	$5{,}09\cdot10^{-9}$	$9{,}47\cdot10^{-10}$	$3{,}35\cdot10^{-9}$	$6{,}76\cdot10^{-9}$	30	197815,9
	20 sekund	$1{,}10\cdot10^{-8}$	$9{,}45\cdot10^{-10}$	$9{,}71\cdot10^{-9}$	$1{,}29\cdot10^{-8}$	20	130261,7
	15 sekund	$2{,}12\cdot10^{-8}$	$3{,}50\cdot10^{-9}$	$1{,}62\cdot10^{-8}$	$2{,}87\cdot10^{-8}$	15	96801,5
	10 sekund	$5{,}15\cdot10^{-8}$	$8{,}49\cdot10^{-9}$	$4{,}13\cdot10^{-8}$	$6{,}32\cdot10^{-8}$	10	62473,1
	5 sekund	$1{,}83\cdot10^{-7}$	$3{,}14\cdot10^{-8}$	$1{,}43\cdot10^{-7}$	$2{,}52\cdot10^{-7}$	5	31868,1
	2 sekundy	$2{,}25\cdot10^{-6}$	$4{,}09\cdot10^{-7}$	$1{,}67\cdot10^{-6}$	$2{,}92\cdot10^{-6}$	2	12098,2
	1 sekunda	$5{,}37\cdot10^{-2}$	$6{,}94\cdot10^{-2}$	$1{,}06\cdot10^{-2}$	$2{,}50\cdot10^{-1}$	1	5720,4
	0,5 sekundy	$2{,}52\cdot10^{1}$	$5{,}53\cdot10^{0}$	$1{,}68\cdot10^{1}$	$3{,}79\cdot10^{1}$	0,5	2694,5
	25000 iteracji	$5{,}56\cdot10^{-8}$	$6{,}73\cdot10^{-9}$	$4{,}55\cdot10^{-8}$	$6{,}86\cdot10^{-8}$	55,49	25000
	20000 iteracji	$8{,}12\cdot10^{-8}$	$1{,}00\cdot10^{-8}$	$6{,}74\cdot10^{-8}$	$9{,}84\cdot10^{-8}$	42,36	20000
	15000 iteracji	$1{,}51\cdot10^{-7}$	$2{,}11\cdot10^{-8}$	$1{,}21\cdot10^{-7}$	$1{,}86\cdot10^{-7}$	33,70	15000
	10000 iteracji	$3{,}60\cdot10^{-7}$	$4{,}17\cdot10^{-8}$	$2{,}82\cdot10^{-7}$	$4{,}25\cdot10^{-7}$	21,30	10000
	5000 iteracji	$1{,}65\cdot10^{-6}$	$2{,}31\cdot10^{-7}$	$1{,}27\cdot10^{-6}$	$1{,}89\cdot10^{-6}$	11,04	5000
	2000 iteracji	$2{,}05\cdot10^{-5}$	$2{,}64\cdot10^{-6}$	$1{,}61\cdot10^{-5}$	$2{,}57\cdot10^{-5}$	4,38	2000
	500 iteracji	$3{,}63\cdot10^{6}$	$3{,}42\cdot10^{5}$	$3{,}00\cdot10^{6}$	$4{,}21\cdot10^{6}$	1,15	500
	100 iteracji	$7{,}67\cdot10^{6}$	$4{,}79\cdot10^{5}$	$6{,}48\cdot10^{6}$	$8{,}15\cdot10^{6}$	0,24	100

Tabela 11
Zestawienie wyników dla autorskiej, zmodyfikowanej wersji algorytmu IWO na podstawie funkcji Rastrigina

Liczba wymiarów	Kryterium stopu	Średni wynik	Odch. standardowe	Min. wynik	Maks. wynik	Śr. czas obliczeń	Śr. liczba iteracji
30 wymiarów	30 sekund	$2{,}76\cdot10^{1}$	$8{,}94\cdot10^{0}$	$1{,}89\cdot10^{1}$	$4{,}78\cdot10^{1}$	30	11896,9
	20 sekund	$2{,}88\cdot10^{1}$	$7{,}90\cdot10^{0}$	$2{,}09\cdot10^{1}$	$4{,}88\cdot10^{1}$	20	7585,3
	15 sekund	$2{,}92\cdot10^{1}$	$7{,}42\cdot10^{0}$	$1{,}99\cdot10^{1}$	$4{,}58\cdot10^{1}$	15	5807,3
	10 sekund	$2{,}90\cdot10^{1}$	$5{,}79\cdot10^{0}$	$1{,}79\cdot10^{1}$	$3{,}88\cdot10^{1}$	10	3731,6
	5 sekund	$3{,}73\cdot10^{1}$	$6{,}62\cdot10^{0}$	$2{,}59\cdot10^{1}$	$4{,}98\cdot10^{1}$	5	1957,1
	2 sekundy	$4{,}11\cdot10^{1}$	$7{,}32\cdot10^{0}$	$3{,}02\cdot10^{1}$	$5{,}24\cdot10^{1}$	2	738,4
	1 sekunda	$4{,}58\cdot10^{1}$	$9{,}25\cdot10^{0}$	$2{,}64\cdot10^{1}$	$5{,}80\cdot10^{1}$	1	372,4
	0,5 sekundy	$9{,}87\cdot10^{1}$	$1{,}21\cdot10^{1}$	$7{,}30\cdot10^{1}$	$1{,}22\cdot10^{2}$	0,5	175,1
	25000 iteracji	$2{,}68\cdot10^{1}$	$4{,}55\cdot10^{0}$	$1{,}99\cdot10^{1}$	$3{,}38\cdot10^{1}$	78,59	25000
	20000 iteracji	$3{,}07\cdot10^{1}$	$6{,}77\cdot10^{0}$	$2{,}19\cdot10^{1}$	$4{,}48\cdot10^{1}$	60,15	20000
	15000 iteracji	$3{,}15\cdot10^{1}$	$7{,}39\cdot10^{0}$	$1{,}89\cdot10^{1}$	$4{,}18\cdot10^{1}$	49,07	15000
	10000 iteracji	$3{,}26\cdot10^{1}$	$4{,}49\cdot10^{0}$	$2{,}69\cdot10^{1}$	$4{,}28\cdot10^{1}$	31,08	10000
	5000 iteracji	$3{,}15\cdot10^{1}$	$4{,}83\cdot10^{0}$	$2{,}49\cdot10^{1}$	$3{,}98\cdot10^{1}$	15,83	5000
	2000 iteracji	$3{,}49\cdot10^{1}$	$8{,}08\cdot10^{0}$	$2{,}49\cdot10^{1}$	$4{,}98\cdot10^{1}$	5,60	2000
	500 iteracji	$4{,}21\cdot10^{1}$	$8{,}84\cdot10^{0}$	$2{,}88\cdot10^{1}$	$5{,}70\cdot10^{1}$	1,30	500
	100 iteracji	$1{,}86\cdot10^{2}$	$2{,}40\cdot10^{1}$	$1{,}36\cdot10^{2}$	$2{,}17\cdot10^{2}$	0,26	100
100 wymiarów	30 sekund	$2{,}72\cdot10^{2}$	$2{,}80\cdot10^{1}$	$2{,}22\cdot10^{2}$	$3{,}16\cdot10^{2}$	30	8009,6
	20 sekund	$2{,}95\cdot10^{2}$	$3{,}03\cdot10^{1}$	$2{,}55\cdot10^{2}$	$3{,}56\cdot10^{2}$	20	5268,2
	15 sekund	$2{,}94\cdot10^{2}$	$3{,}89\cdot10^{1}$	$2{,}35\cdot10^{2}$	$3{,}63\cdot10^{2}$	15	4181
	10 sekund	$3{,}03\cdot10^{2}$	$4{,}33\cdot10^{1}$	$2{,}25\cdot10^{2}$	$3{,}86\cdot10^{2}$	10	2708,9
	5 sekund	$3{,}01\cdot10^{2}$	$3{,}60\cdot10^{1}$	$2{,}40\cdot10^{2}$	$3{,}61\cdot10^{2}$	5	1462,4
	2 sekundy	$4{,}62\cdot10^{2}$	$3{,}16\cdot10^{1}$	$3{,}90\cdot10^{2}$	$4{,}99\cdot10^{2}$	2	611,7
	1 sekunda	$7{,}23\cdot10^{2}$	$6{,}32\cdot10^{1}$	$6{,}27\cdot10^{2}$	$8{,}34\cdot10^{2}$	1	298,1
	0,5 sekundy	$9{,}55\cdot10^{2}$	$5{,}34\cdot10^{1}$	$8{,}94\cdot10^{2}$	$1{,}08\cdot10^{3}$	0,5	154,9
	25000 iteracji	$2{,}82\cdot10^{2}$	$4{,}51\cdot10^{1}$	$2{,}14\cdot10^{2}$	$3{,}36\cdot10^{2}$	121,32	25000
	20000 iteracji	$2{,}94\cdot10^{2}$	$3{,}10\cdot10^{1}$	$2{,}43\cdot10^{2}$	$3{,}38\cdot10^{2}$	91,47	20000
	15000 iteracji	$2{,}97\cdot10^{2}$	$5{,}11\cdot10^{1}$	$2{,}25\cdot10^{2}$	$3{,}74\cdot10^{2}$	75,47	15000
	10000 iteracji	$3{,}06\cdot10^{2}$	$4{,}33\cdot10^{1}$	$2{,}07\cdot10^{2}$	$3{,}63\cdot10^{2}$	51,61	10000
	5000 iteracji	$3{,}08\cdot10^{2}$	$3{,}95\cdot10^{1}$	$2{,}38\cdot10^{2}$	$3{,}81\cdot10^{2}$	23,51	5000
	2000 iteracji	$3{,}22\cdot10^{2}$	$3{,}54\cdot10^{1}$	$2{,}72\cdot10^{2}$	$3{,}81\cdot10^{2}$	8,87	2000
	500 iteracji	$4{,}86\cdot10^{2}$	$4{,}00\cdot10^{1}$	$4{,}23\cdot10^{2}$	$5{,}48\cdot10^{2}$	2,08	500
	100 iteracji	$9{,}51\cdot10^{2}$	$3{,}58\cdot10^{1}$	$8{,}91\cdot10^{2}$	$9{,}92\cdot10^{2}$	0,38	100

Tabela 12
Zestawienie wyników dla autorskiej, zmodyfikowanej wersji algorytmu IWO na podstawie funkcji Rosenbrocka

Liczba wymiarów	Kryterium stopu	Średni wynik	Odch. standardowe	Min. wynik	Maks. wynik	Śr. czas obliczeń	Śr. liczba iteracji
30 wymiarów	30 sekund	$2{,}04\cdot10^{1}$	$1{,}99\cdot10^{0}$	$1{,}80\cdot10^{1}$	$2{,}46\cdot10^{1}$	30	50033,3
	20 sekund	$2{,}26\cdot10^{1}$	$2{,}07\cdot10^{0}$	$1{,}94\cdot10^{1}$	$2{,}71\cdot10^{1}$	20	31095,8
	15 sekund	$2{,}25\cdot10^{1}$	$1{,}69\cdot10^{0}$	$2{,}05\cdot10^{1}$	$2{,}53\cdot10^{1}$	15	29366,5
	10 sekund	$2{,}38\cdot10^{1}$	$3{,}75\cdot10^{0}$	$1{,}49\cdot10^{1}$	$2{,}82\cdot10^{1}$	10	18856,4
	5 sekund	$2{,}38\cdot10^{1}$	$5{,}88\cdot10^{0}$	$9{,}81\cdot10^{0}$	$2{,}89\cdot10^{1}$	5	9759,1
	2 sekundy	$3{,}27\cdot10^{1}$	$1{,}76\cdot10^{1}$	$2{,}19\cdot10^{1}$	$8{,}51\cdot10^{1}$	2	5451,1
	1 sekunda	$3{,}31\cdot10^{1}$	$1{,}67\cdot10^{1}$	$2{,}40\cdot10^{1}$	$8{,}31\cdot10^{1}$	1	1786,1
	0,5 sekundy	$3{,}91\cdot10^{1}$	$2{,}49\cdot10^{1}$	$2{,}26\cdot10^{1}$	$9{,}42\cdot10^{1}$	0,5	1210,6
	25000 iteracji	$3{,}01\cdot10^{0}$	$7{,}67\cdot10^{-1}$	$1{,}38\cdot10^{0}$	$3{,}97\cdot10^{0}$	13,78	25000
	20000 iteracji	$5{,}92\cdot10^{0}$	$5{,}58\cdot10^{-1}$	$4{,}48\cdot10^{0}$	$6{,}68\cdot10^{0}$	10,18	20000
	15000 iteracji	$9{,}87\cdot10^{0}$	$7{,}10\cdot10^{-1}$	$8{,}66\cdot10^{0}$	$1{,}08\cdot10^{1}$	7,88	15000
	10000 iteracji	$1{,}42\cdot10^{1}$	$8{,}69\cdot10^{-1}$	$1{,}20\cdot10^{1}$	$1{,}54\cdot10^{1}$	5,93	10000
	5000 iteracji	$2{,}58\cdot10^{1}$	$1{,}69\cdot10^{1}$	$1{,}82\cdot10^{1}$	$7{,}63\cdot10^{1}$	3,02	5000
	2000 iteracji	$2{,}73\cdot10^{1}$	$2{,}50\cdot10^{0}$	$2{,}13\cdot10^{1}$	$2{,}92\cdot10^{1}$	0,52	2000
	500 iteracji	$3{,}30\cdot10^{1}$	$1{,}70\cdot10^{1}$	$2{,}17\cdot10^{1}$	$8{,}37\cdot10^{1}$	0,43	500
	100 iteracji	$3{,}54\cdot10^{1}$	$1{,}88\cdot10^{1}$	$2{,}65\cdot10^{1}$	$9{,}17\cdot10^{1}$	0,19	100
100 wymiarów	30 sekund	$9{,}60\cdot10^{1}$	$1{,}41\cdot10^{0}$	$9{,}33\cdot10^{1}$	$9{,}73\cdot10^{1}$	30	63295,1
	20 sekund	$1{,}02\cdot10^{2}$	$1{,}62\cdot10^{1}$	$9{,}45\cdot10^{1}$	$1{,}50\cdot10^{2}$	20	43179,4
	15 sekund	$1{,}01\cdot10^{2}$	$1{,}49\cdot10^{1}$	$9{,}19\cdot10^{1}$	$1{,}46\cdot10^{2}$	15	31536,8
	10 sekund	$1{,}08\cdot10^{2}$	$2{,}28\cdot10^{1}$	$9{,}13\cdot10^{1}$	$1{,}55\cdot10^{2}$	10	20374,5
	5 sekund	$1{,}13\cdot10^{2}$	$2{,}51\cdot10^{1}$	$9{,}11\cdot10^{1}$	$1{,}53\cdot10^{2}$	5	10753,1
	2 sekundy	$1{,}08\cdot10^{2}$	$2{,}09\cdot10^{1}$	$9{,}67\cdot10^{1}$	$1{,}50\cdot10^{2}$	2	3705,1
	1 sekunda	$1{,}09\cdot10^{2}$	$2{,}20\cdot10^{1}$	$9{,}56\cdot10^{1}$	$1{,}53\cdot10^{2}$	1	3061,1
	0,5 sekundy	$1{,}27\cdot10^{2}$	$2{,}53\cdot10^{1}$	$9{,}63\cdot10^{1}$	$1{,}55\cdot10^{2}$	0,5	1517,1
	25000 iteracji	$8{,}70\cdot10^{1}$	$2{,}63\cdot10^{0}$	$8{,}20\cdot10^{1}$	$9{,}08\cdot10^{1}$	126,82	25000
	20000 iteracji	$8{,}87\cdot10^{1}$	$2{,}44\cdot10^{0}$	$8{,}55\cdot10^{1}$	$9{,}28\cdot10^{1}$	100,12	20000
	15000 iteracji	$8{,}89\cdot10^{1}$	$2{,}93\cdot10^{0}$	$8{,}30\cdot10^{1}$	$9{,}26\cdot10^{1}$	78,11	15000
	10000 iteracji	$9{,}08\cdot10^{1}$	$1{,}49\cdot10^{0}$	$8{,}76\cdot10^{1}$	$9{,}31\cdot10^{1}$	54,61	10000
	5000 iteracji	$9{,}13\cdot10^{1}$	$1{,}44\cdot10^{0}$	$8{,}84\cdot10^{1}$	$9{,}36\cdot10^{1}$	26,50	5000
	2000 iteracji	$9{,}32\cdot10^{1}$	$1{,}95\cdot10^{0}$	$9{,}06\cdot10^{1}$	$9{,}58\cdot10^{1}$	10,08	2000
	500 iteracji	$1{,}75\cdot10^{2}$	$6{,}32\cdot10^{1}$	$1{,}03\cdot10^{2}$	$2{,}96\cdot10^{2}$	0,70	500
	100 iteracji	$9{,}20\cdot10^{2}$	$8{,}63\cdot10^{1}$	$8{,}15\cdot10^{2}$	$1{,}14\cdot10^{3}$	0,72	100

Tabela 13
Zestawienie wyników dla autorskiej, zmodyfikowanej wersji algorytmu IWO na podstawie funkcji Ackleya

Liczba wymiarów	Kryterium stopu	Średni wynik	Odch. standardowe	Min. wynik	Maks. wynik	Śr. czas obliczeń	Śr. liczba iteracji
30 wymiarów	30 sekund	$4{,}25\cdot10^{-4}$	$3{,}03\cdot10^{-5}$	$3{,}80\cdot10^{-4}$	$4{,}75\cdot10^{-4}$	30	18805,2
	20 sekund	$4{,}31\cdot10^{-4}$	$2{,}56\cdot10^{-5}$	$3{,}94\cdot10^{-4}$	$4{,}68\cdot10^{-4}$	20	12430,4
	15 sekund	$4{,}47\cdot10^{-4}$	$3{,}57\cdot10^{-5}$	$3{,}73\cdot10^{-4}$	$4{,}92\cdot10^{-4}$	15	9309,2
	10 sekund	$4{,}53\cdot10^{-4}$	$5{,}90\cdot10^{-5}$	$3{,}47\cdot10^{-4}$	$5{,}42\cdot10^{-4}$	10	5948,1
	5 sekund	$4{,}88\cdot10^{-4}$	$4{,}47\cdot10^{-5}$	$4{,}02\cdot10^{-4}$	$5{,}58\cdot10^{-4}$	5	3172,1
	2 sekundy	$7{,}02\cdot10^{-4}$	$6{,}95\cdot10^{-5}$	$5{,}94\cdot10^{-4}$	$8{,}11\cdot10^{-4}$	2	1200,5
	1 sekunda	$1{,}73\cdot10^{-3}$	$4{,}97\cdot10^{-4}$	$1{,}01\cdot10^{-3}$	$2{,}51\cdot10^{-3}$	1	629,6
	0,5 sekundy	$4{,}84\cdot10^{-1}$	$7{,}30\cdot10^{-1}$	$6{,}15\cdot10^{-3}$	$1{,}90\cdot10^{0}$	0,5	300,1
	25000 iteracji	$4{,}18\cdot10^{-4}$	$2{,}26\cdot10^{-5}$	$3{,}86\cdot10^{-4}$	$4{,}55\cdot10^{-4}$	45,42	25000
	20000 iteracji	$4{,}14\cdot10^{-4}$	$2{,}95\cdot10^{-5}$	$3{,}74\cdot10^{-4}$	$4{,}70\cdot10^{-4}$	35,98	20000
	15000 iteracji	$4{,}38\cdot10^{-4}$	$2{,}37\cdot10^{-5}$	$3{,}95\cdot10^{-4}$	$4{,}79\cdot10^{-4}$	27,06	15000
	10000 iteracji	$4{,}51\cdot10^{-4}$	$1{,}71\cdot10^{-5}$	$4{,}25\cdot10^{-4}$	$4{,}75\cdot10^{-4}$	18,53	10000
	5000 iteracji	$4{,}59\cdot10^{-4}$	$2{,}59\cdot10^{-5}$	$4{,}10\cdot10^{-4}$	$4{,}88\cdot10^{-4}$	9,20	5000
	2000 iteracji	$5{,}90\cdot10^{-4}$	$3{,}53\cdot10^{-5}$	$5{,}41\cdot10^{-4}$	$6{,}41\cdot10^{-4}$	3,32	2000
	500 iteracji	$6{,}10\cdot10^{-3}$	$9{,}78\cdot10^{-4}$	$4{,}56\cdot10^{-3}$	$7{,}74\cdot10^{-3}$	0,72	500
	100 iteracji	$1{,}25\cdot10^{1}$	$2{,}41\cdot10^{0}$	$7{,}59\cdot10^{0}$	$1{,}65\cdot10^{1}$	0,16	100
100 wymiarów	30 sekund	$4{,}76\cdot10^{0}$	$7{,}04\cdot10^{-1}$	$3{,}83\cdot10^{0}$	$5{,}94\cdot10^{0}$	30	34678
	20 sekund	$6{,}24\cdot10^{0}$	$4{,}40\cdot10^{0}$	$3{,}59\cdot10^{0}$	$1{,}93\cdot10^{1}$	20	22694,1
	15 sekund	$7{,}57\cdot10^{0}$	$5{,}92\cdot10^{0}$	$3{,}64\cdot10^{0}$	$1{,}93\cdot10^{1}$	15	16553,4
	10 sekund	$7{,}18\cdot10^{0}$	$6{,}01\cdot10^{0}$	$3{,}66\cdot10^{0}$	$1{,}93\cdot10^{1}$	10	10924
	5 sekund	$6{,}74\cdot10^{0}$	$4{,}34\cdot10^{0}$	$3{,}67\cdot10^{0}$	$1{,}93\cdot10^{1}$	5	5849,3
	2 sekundy	$7{,}93\cdot10^{0}$	$5{,}79\cdot10^{0}$	$3{,}17\cdot10^{0}$	$1{,}94\cdot10^{1}$	2	2190,7
	1 sekunda	$8{,}50\cdot10^{0}$	$5{,}53\cdot10^{0}$	$4{,}16\cdot10^{0}$	$1{,}94\cdot10^{1}$	1	1111,5
	0,5 sekundy	$8{,}70\cdot10^{0}$	$3{,}87\cdot10^{0}$	$5{,}82\cdot10^{0}$	$1{,}96\cdot10^{1}$	0,5	556,1
	25000 iteracji	$3{,}42\cdot10^{0}$	$4{,}48\cdot10^{-1}$	$2{,}63\cdot10^{0}$	$4{,}04\cdot10^{0}$	38,89	25000
	20000 iteracji	$3{,}73\cdot10^{0}$	$5{,}85\cdot10^{-1}$	$2{,}81\cdot10^{0}$	$4{,}56\cdot10^{0}$	30,52	20000
	15000 iteracji	$3{,}88\cdot10^{0}$	$7{,}41\cdot10^{-1}$	$2{,}68\cdot10^{0}$	$5{,}33\cdot10^{0}$	23,47	15000
	10000 iteracji	$4{,}14\cdot10^{0}$	$7{,}05\cdot10^{-1}$	$2{,}79\cdot10^{0}$	$5{,}29\cdot10^{0}$	15,73	10000
	5000 iteracji	$4{,}04\cdot10^{0}$	$6{,}60\cdot10^{-1}$	$2{,}92\cdot10^{0}$	$5{,}36\cdot10^{0}$	7,13	5000
	2000 iteracji	$5{,}68\cdot10^{0}$	$4{,}60\cdot10^{0}$	$2{,}99\cdot10^{0}$	$1{,}94\cdot10^{1}$	3,33	2000
	500 iteracji	$9{,}12\cdot10^{0}$	$5{,}37\cdot10^{0}$	$5{,}52\cdot10^{0}$	$1{,}99\cdot10^{1}$	1,11	500
	100 iteracji	$1{,}97\cdot10^{1}$	$4{,}33\cdot10^{-1}$	$1{,}88\cdot10^{1}$	$2{,}02\cdot10^{1}$	0,13	100

A.1.2. Porównanie zmodyfikowanej wersji IWO z wersją oryginalną

Podstawą niniejszej serii badań stały się eksperymenty porównawcze zmodyfikowanej i oryginalnej wersji algorytmu IWO zaprezentowanej w [101]. Oparto się na funkcjach Griewanka i Rastrigina o 30 wymiarach. Szczegółowe wyniki eksperymentów zawarto w tabelach 14 (funkcja Griewanka) i 15 (funkcja Rastrigina). W tabelach dodano kolumnę zawierającą znak graficzny wskazujący, czy autorska zmodyfikowana wersja algorytmu IWO jest lepsza (✓) lub gorsza (✗) od wersji oryginalnej.

Tabela 14

Porównanie oryginalnej i zmodyfikowanej wersji algorytmu IWO na podstawie 30-wymiarowej funkcji Griewanka

Kryterium stopu	**Wersja algorytmu**	✓ ✗	**Średni wynik**	**Odch. stand.**	**Min. wynik**	**Maks. wynik**	**Śr. czas obliczeń**	**Śr. liczba iteracji**
5 sekund	oryg.	✓	$6{,}91\cdot10^{-7}$	$2{,}06\cdot10^{-7}$	$2{,}72\cdot10^{-7}$	$1{,}26\cdot10^{-6}$	5s	56651
	zmodyf.		$5{,}32\cdot10^{-11}$	$1{,}45\cdot10^{-11}$	$2{,}41\cdot10^{-11}$	$1{,}19\cdot10^{-10}$	5s	35246
2 sekundy	oryg.	✓	$4{,}29\cdot10^{-6}$	$1{,}29\cdot10^{-6}$	$2{,}04\cdot10^{-6}$	$8{,}47\cdot10^{-6}$	2s	22789
	zmodyf.		$2{,}18\cdot10^{-9}$	$5{,}57\cdot10^{-10}$	$1{,}09\cdot10^{-9}$	$3{,}91\cdot10^{-9}$	2s	9690
1 sekunda	oryg.	✓	$2{,}23\cdot10^{-5}$	$4{,}00\cdot10^{-5}$	$7{,}37\cdot10^{-6}$	$4{,}13\cdot10^{-4}$	1s	10634
	zmodyf.		$4{,}02\cdot10^{-9}$	$1{,}95\cdot10^{-9}$	$1{,}51\cdot10^{-9}$	$2{,}07\cdot10^{-8}$	1s	7030
0,5 sekundy	oryg.	✓	$1{,}82\cdot10^{-3}$	$9{,}22\cdot10^{-3}$	$2{,}88\cdot10^{-5}$	$6{,}44\cdot10^{-2}$	0,5s	4914
	zmodyf.		$8{,}65\cdot10^{-7}$	$2{,}53\cdot10^{-7}$	$3{,}25\cdot10^{-7}$	$2{,}19\cdot10^{-6}$	0,5s	1768
20 000 iteracji	oryg.	✓	$2{,}67\cdot10^{-6}$	$6{,}24\cdot10^{-7}$	$1{,}49\cdot10^{-6}$	$4{,}28\cdot10^{-6}$	11,77s	20 000
	zmodyf.		$1{,}31\cdot10^{-10}$	$30{,}8\cdot10^{-11}$	$5{,}69\cdot10^{-11}$	$2{,}11\cdot10^{-10}$	16,43s	20 000
10 000 iteracji	oryg.	✓	$1{,}10\cdot10^{-5}$	$2{,}34\cdot10^{-6}$	$5{,}17\cdot10^{-6}$	$1{,}76\cdot10^{-5}$	5,89s	10 000
	zmodyf.		$5{,}24\cdot10^{-10}$	$1{,}14\cdot10^{-10}$	$2{,}95\cdot10^{-10}$	$7{,}88\cdot10^{-10}$	8,18s	10 000
5000 iteracji	oryg.	✓	$4{,}55\cdot10^{-5}$	$8{,}68\cdot10^{-6}$	$2{,}31\cdot10^{-5}$	$6{,}39\cdot10^{-5}$	2,96s	5000
	zmodyf.		$2{,}17\cdot10^{-9}$	$4{,}98\cdot10^{-10}$	$1{,}34\cdot10^{-9}$	$3{,}90\cdot10^{-9}$	3,98s	5000
2000 iteracji	oryg.	✓	$3{,}52\cdot10^{-4}$	$7{,}10\cdot10^{-5}$	$1{,}77\cdot10^{-4}$	$5{,}96\cdot10^{-4}$	1,15s	2000
	zmodyf.		$8{,}38\cdot10^{-9}$	$1{,}81\cdot10^{-9}$	$5{,}26\cdot10^{-9}$	$1{,}39\cdot10^{-8}$	2,12s	2000
500 iteracji	oryg.	✓	$6{,}96\cdot10^{-2}$	$2{,}73\cdot10^{-2}$	$2{,}85\cdot10^{-2}$	$1{,}52\cdot10^{-1}$	0,28s	500
	zmodyf.		$3{,}19\cdot10^{-4}$	$6{,}30\cdot10^{-5}$	$1{,}75\cdot10^{-4}$	$4{,}97\cdot10^{-4}$	0,51s	500
100 iteracji	oryg.	✓	$1{,}42\cdot10^{4}$	$4{,}77\cdot10^{3}$	$5{,}92\cdot10^{3}$	$2{,}83\cdot10^{4}$	0,03s	100
	zmodyf.		$6{,}59\cdot10^{2}$	$1{,}43\cdot10^{2}$	$3{,}44\cdot10^{2}$	$9{,}71\cdot10^{2}$	0,40s	100

Tabela 15
Porównanie oryginalnej i zmodyfikowanej wersji algorytmu IWO na podstawie 30-wymiarowej funkcji Rastrigina

Kryterium stopu	Wersja algorytmu	✓ ✗	Średni wynik	Odch. stand.	Min. wynik	Maks. wynik	Śr. czas obliczeń	Śr. liczba iteracji
5 sekund	oryg.	✓	$1{,}68\cdot10^{2}$	$5{,}24\cdot10^{1}$	$6{,}37\cdot10^{1}$	$2{,}95\cdot10^{2}$	5s	19133
	zmodyf.		$3{,}19\cdot10^{1}$	$8{,}09\cdot10^{0}$	$1{,}59\cdot10^{1}$	$5{,}47\cdot10^{1}$	5s	1949
2 sekundy	oryg.	✓	$1{,}75\cdot10^{2}$	$5{,}36\cdot10^{1}$	$7{,}56\cdot10^{1}$	$3{,}18\cdot10^{2}$	2s	5144
	zmodyf.		$3{,}71\cdot10^{1}$	$8{,}02\cdot10^{0}$	$2{,}02\cdot10^{1}$	$5{,}62\cdot10^{1}$	2s	750
1 sekunda	oryg.	✓	$1{,}65\cdot10^{2}$	$4{,}32\cdot10^{1}$	$7{,}98\cdot10^{1}$	$2{,}96\cdot10^{2}$	1s	2567
	zmodyf.		$4{,}66\cdot10^{1}$	$1{,}07\cdot10^{1}$	$2{,}59\cdot10^{1}$	$8{,}36\cdot10^{1}$	1s	441
0,5 sekundy	oryg.	✓	$2{,}90\cdot10^{4}$	$5{,}32\cdot10^{4}$	$1{,}94\cdot10^{2}$	$2{,}95\cdot10^{5}$	0,5s	1804
	zmodyf.		$1{,}20\cdot10^{2}$	$2{,}76\cdot10^{1}$	$4{,}79\cdot10^{1}$	$1{,}90\cdot10^{2}$	0,5s	297
20 000 iteracji	oryg.	✓	$1{,}51\cdot10^{2}$	$3{,}98\cdot10^{1}$	$7{,}96\cdot10^{1}$	$2{,}50\cdot10^{2}$	7,28s	20 000
	zmodyf.		$8{,}54\cdot10^{1}$	$1{,}96\cdot10^{1}$	$4{,}88\cdot10^{1}$	$1{,}45\cdot10^{2}$	26,71s	20 000
10 000 iteracji	oryg.	✓	$1{,}59\cdot10^{2}$	$4{,}73\cdot10^{1}$	$5{,}57\cdot10^{1}$	$3{,}74\cdot10^{2}$	3,60s	10 000
	zmodyf.		$8{,}41\cdot10^{1}$	$1{,}97\cdot10^{1}$	$3{,}48\cdot10^{1}$	$1{,}42\cdot10^{2}$	13,44s	10 000
5000 iteracji	oryg.	✓	$1{,}58\cdot10^{2}$	$4{,}27\cdot10^{1}$	$6{,}67\cdot10^{1}$	$3{,}14\cdot10^{2}$	1,74s	5000
	zmodyf.		$8{,}74\cdot10^{1}$	$2{,}18\cdot10^{1}$	$4{,}58\cdot10^{1}$	$1{,}51\cdot10^{2}$	6,71s	5000
2000 iteracji	oryg.	✓	$1{,}80\cdot10^{4}$	$1{,}15\cdot10^{4}$	$7{,}31\cdot10^{2}$	$4{,}48\cdot10^{4}$	0,68s	2000
	zmodyf.		$8{,}89\cdot10^{1}$	$2{,}34\cdot10^{1}$	$4{,}78\cdot10^{1}$	$1{,}84\cdot10^{2}$	2,15s	2000
500 iteracji	oryg.	✓	$5{,}03\cdot10^{5}$	$8{,}64\cdot10^{4}$	$2{,}63\cdot10^{5}$	$7{,}24\cdot10^{5}$	0,16s	500
	zmodyf.		$4{,}69\cdot10^{1}$	$1{,}07\cdot10^{1}$	$2{,}63\cdot10^{1}$	$7{,}69\cdot10^{1}$	0,98s	500
100 iteracji	oryg.	✓	$1{,}40\cdot10^{6}$	$1{,}52\cdot10^{5}$	$9{,}72\cdot10^{5}$	$1{,}69\cdot10^{6}$	0,03s	100
	zmodyf.		$1{,}74\cdot10^{2}$	$1{,}65\cdot10^{1}$	$1{,}36\cdot10^{2}$	$2{,}12\cdot10^{2}$	0,59s	100

A.1.3. Porównanie exIWO z algorytmem APSO

Niniejszy podrozdział stanowi zestawienie wyników badań porównawczych z adaptacyjnym algorytmem roju cząstek (ang. *adaptive particle swarm optimization*, APSO) [143].

Podstawą eksperymentów stały się funkcje: Rastrigina, Rosenbrocka i Griewanka. W tabelach 16-18 zestawiono wartości średnie wartości minimalnych funkcji testowych przy użyciu algorytmów APSO (kolumna „APSO wyniki”) i exIWO (kolumna „IWO wyniki”). Wszystkie obliczenia testowe przeprowadzono 500 razy.

Podobnie jak w poprzednim podrozdziale, w tabelach dodano kolumnę zawierającą znak graficzny wskazujący, czy autorska zmodyfikowana wersja algorytmu IWO jest lepsza (✓) lub gorsza (✗) od algorytmu APSO.

Tabela 16

Porównanie wyników uzyskanych przy użyciu algorytmów APSO i IWO na podstawie funkcji Rastrigina

Liczba osobników	Liczba wymiarów	Liczba iteracji	Wyniki dla algorytmu APSO	Wyniki dla algorytmu IWO	✓ ✗
20	10	1000	$1{,}36 \cdot 10^{0}$	$9{,}76 \cdot 10^{0}$	✗
	20	1500	$9{,}40 \cdot 10^{0}$	$2{,}95 \cdot 10^{1}$	✗
	30	2000	$2{,}44 \cdot 10^{1}$	$5{,}81 \cdot 10^{1}$	✗
40	10	1000	$4{,}77 \cdot 10^{-1}$	$7{,}60 \cdot 10^{0}$	✗
	20	1500	$4{,}62 \cdot 10^{0}$	$2{,}33 \cdot 10^{1}$	✗
	30	2000	$1{,}28 \cdot 10^{1}$	$4{,}61 \cdot 10^{1}$	✗
80	10	1000	$7{,}48 \cdot 10^{-2}$	$5{,}89 \cdot 10^{0}$	✗
	20	1500	$2{,}32 \cdot 10^{0}$	$1{,}96 \cdot 10^{1}$	✗
	30	2000	$7{,}25 \cdot 10^{0}$	$3{,}74 \cdot 10^{1}$	✗
160	10	1000	$2{,}00 \cdot 10^{-3}$	$4{,}30 \cdot 10^{0}$	✗
	20	1500	$9{,}62 \cdot 10^{-1}$	$1{,}56 \cdot 10^{1}$	✗
	30	2000	$4{,}06 \cdot 10^{0}$	$3{,}09 \cdot 10^{1}$	✗

Tabela 17

Porównanie wyników uzyskanych przy użyciu algorytmów APSO i IWO na podstawie funkcji Rosenbrocka

Liczba osobników	Liczba wymiarów	Liczba iteracji	Wyniki dla algorytmu APSO	Wyniki dla algorytmu IWO	✓ ✗
20	10	1000	$2{,}88 \cdot 10^{1}$	$7{,}99 \cdot 10^{1}$	✗
	20	1500	$6{,}20 \cdot 10^{1}$	$8{,}54 \cdot 10^{1}$	✗
	30	2000	$9{,}56 \cdot 10^{1}$	$9{,}29 \cdot 10^{1}$	✓
40	10	1000	$1{,}64 \cdot 10^{1}$	$4{,}30 \cdot 10^{1}$	✗
	20	1500	$3{,}72 \cdot 10^{1}$	$4{,}70 \cdot 10^{1}$	✗
	30	2000	$5{,}73 \cdot 10^{1}$	$5{,}20 \cdot 10^{1}$	✓
80	10	1000	$1{,}16 \cdot 10^{1}$	$1{,}99 \cdot 10^{1}$	✗
	20	1500	$3{,}26 \cdot 10^{1}$	$2{,}65 \cdot 10^{1}$	✓
	30	2000	$5{,}53 \cdot 10^{1}$	$3{,}17 \cdot 10^{1}$	✓
160	10	1000	$6{,}95 \cdot 10^{0}$	$8{,}33 \cdot 10^{0}$	✗
	20	1500	$2{,}60 \cdot 10^{1}$	$1{,}22 \cdot 10^{1}$	✓
	30	2000	$4{,}57 \cdot 10^{1}$	$1{,}60 \cdot 10^{1}$	✓

Tabela 18

Porównanie wyników uzyskanych przy użyciu algorytmów APSO i IWO na podstawie funkcji Griewanka

Liczba osobników	Liczba wymiarów	Liczba iteracji	Wyniki dla algorytmu APSO	Wyniki dla algorytmu IWO	✓ ✗
20	10	1000	$6{,}82 \cdot 10^{-2}$	$1{,}83 \cdot 10^{-9}$	✓
	20	1500	$2{,}59 \cdot 10^{-2}$	$5{,}72 \cdot 10^{-7}$	✓
	30	2000	$2{,}32 \cdot 10^{-2}$	$2{,}35 \cdot 10^{-6}$	✓
40	10	1000	$5{,}57 \cdot 10^{-2}$	$1{,}28 \cdot 10^{-9}$	✓
	20	1500	$2{,}11 \cdot 10^{-2}$	$4{,}24 \cdot 10^{-7}$	✓
	30	2000	$1{,}36 \cdot 10^{-2}$	$1{,}70 \cdot 10^{-6}$	✓
80	10	1000	$5{,}26 \cdot 10^{-2}$	$9{,}46 \cdot 10^{-10}$	✓
	20	1500	$2{,}04 \cdot 10^{-2}$	$3{,}15 \cdot 10^{-7}$	✓
	30	2000	$1{,}05 \cdot 10^{-2}$	$1{,}33 \cdot 10^{-6}$	✓
160	10	1000	$4{,}34 \cdot 10^{-2}$	$7{,}10 \cdot 10^{-10}$	✓
	20	1500	$1{,}82 \cdot 10^{-2}$	$2{,}55 \cdot 10^{-7}$	✓
	30	2000	$1{,}04 \cdot 10^{-2}$	$1{,}12 \cdot 10^{-6}$	✓

A.2. Problem komiwojażera – porównanie operatorów

W tym podrozdziale zawarto zestawienie wyników badań rozwiązujących klasyczny problem komiwojażera zarówno symetryczny, jak i asymetryczny.

Wyniki obliczeń dla problemu symetrycznego zebrano w tabeli 19, dla asymetrycznego – w tabeli 20. Kryterium zatrzymania algorytmu to 10 000 iteracji. Eksperymenty przeprowadzano stukrotnie. Wartości w kolumnach średnia („śr.”), maksimum („maks.”), minimum („min.”), mediana („med.”) i odchylenie standardowe wyznaczone dla uzyskanych wyników („odch.”) są procentowym odchyleniem od wartości optymalnych. Kolumna numer iteracji („iter.”) jest wartością określającą numer pokolenia algorytmu, w którym pojawiło się najlepsze rozwiązanie; zaś czas, podawany w sekundach, jest interwałem, jaki upłynął od rozpoczęcia do zakończenia algorytmu optymalizacyjnego. Wszystkie opisane wartości są średnimi ze 100 uruchomień IWO. Kolumna odległość („odl.”) zawiera oznaczenie literowe typu obliczania dystansu pomiędzy miastami: E – odległość euklidesowa, C – zaokrąglona odległość euklidesowa, A – odległość pseudo-euklidesowa, G – odległość geograficzna oraz M – odległość podana wprost w postaci macierzowej. Koszt zawiera wartości optymalne, natomiast operator („op.”) zawiera informację o typie transformacji: *inver-over* (I-O) lub odwracania (odw.).

Tabela 19

Zestawienie wyników dla operatorów *inver-over* i odwracania – symetryczny problem komiwojażera

Nazwa	Odl.	Koszt	Op.	Śr.	Maks.	Min.	Odch.	Med.	Czas	Iter.
att48	A	10628	I-O	0,09	0,74	0	0,12	0	4,0	1146
			Odw.	0,04	0,24	0	0,08	0	9,3	996
att532	A	27686	I-O	4,24	5,51	3,02	0,52	4,16	44,4	9568
			Odw.	3,34	4,34	2,53	0,39	3,31	93,8	9188
dsj1000	C	18659688	I-O	8,95	11,07	7,32	0,67	9,04	232,1	9814
			Odw.	9,26	11,17	7,54	0,70	9,25	184,3	9818
pla7397	C	23260728	I-O	17,05	18,04	15,74	0,47	17,13	1298,9	9918
			Odw.	17,17	18,08	15,05	0,53	17,19	1116,3	9931
eil51	E	426	I-O	0,13	0,24	0	0,12	0,24	30,0	3258
			Odw.	0,12	0,24	0	0,12	0,24	29,1	3410
berlin52	E	7542	I-O	0	0	0	0	0	39,1	185
			Odw.	0	0	0	0	0	38,8	201
st70	E	675	I-O	0,61	0,89	0,15	0,32	0,74	51,3	4722
			Odw.	0,56	0,89	0	0,32	0,59	51,0	5008
eil76	E	538	I-O	0,30	0,56	0	0,23	0,19	46,2	4057
			Odw.	0,31	0,930	0	0,25	0,19	45,6	4041
pr76	E	108159	I-O	0,66	1,38	0	0,34	0,83	40,7	4670
			Odw.	0,66	1,38	0	0,34	0,82	40,7	5097
rat99	E	1211	I-O	0,18	0,99	0	0,20	0,17	69,5	4515
			Odw.	0,16	1,16	0	0,21	0,17	70,3	4428
kroA100	E	21282	I-O	0	0	0	0	0	65,9	1570
			Odw.	0	0	0	0	0	65,0	1467
kroB100	E	22141	I-O	0	0,23	0	0,02	0	70,0	2360
			Odw.	0,01	0,23	0	0,04	0	68,1	2510
kroC100	E	20749	I-O	0,23	0,57	0	0,06	0,24	75,5	2931
			Odw.	0,23	0,33	0	0,04	0,24	73,3	3131
kroD100	E	21294	I-O	0,49	1,37	0	0,28	0,54	74,2	6744
			Odw.	0,50	1,32	0	0,24	0,54	73,2	6805
kroE100	E	22068	I-O	0,21	0,35	0	0,10	0,24	79,7	5561
			Odw.	0,19	0,35	0	0,12	0,24	78,2	6275
rd100	E	7910	I-O	0,07	0,43	0	0,16	0	68,6	6181
			Odw.	0,10	0,43	0	0,18	0	67,8	6077
eil101	E	629	I-O	0,01	0,16	0	0,05	0	69,7	2406
			Odw.	0,02	0,16	0	0,06	0	68,4	1834
lin105	E	14379	I-O	0,09	0,77	0	0,21	0	70,6	3315
			Odw.	0,09	0,79	0	0,20	0	68,5	3807
pr107	E	44303	I-O	0	0	0	0	0	77,8	941
			Odw.	0	0	0	0	0	75,5	990
pr124	E	59030	I-O	0,08	0,35	0	0,05	0,10	95,6	5439
			Odw.	0,09	0,22	0	0,05	0,10	91,2	5797

Nazwa	Odl.	Koszt	Op.	Śr.	Maks.	Min.	Odch.	Med.	Czas	Iter.
bier127	E	118282	I-O	0,10	0,40	0	0,14	0	100,2	4337
			Odw.	0,09	0,40	0	0,13	0	94,8	3964
ch130	E	6110	I-O	1,08	1,73	0,47	0,32	1,10	100,2	7308
			Odw.	1,07	1,62	0,47	0,29	1,06	95,5	7408
pr136	E	96772	I-O	3,79	5,34	1,69	0,68	3,86	99,3	8041
			Odw.	3,73	4,83	1,52	0,63	3,79	94,6	8555
pr144	E	58537	I-O	0,16	0,20	0,11	0,02	0,17	107,7	5368
			Odw.	0,17	0,26	0,11	0,02	0,17	102,3	5800
ch150	E	6528	I-O	0,46	0,67	0,18	0,12	0,46	18,8	6153
			Odw.	0,40	0,66	0	0,11	0,40	25,6	5097
kroA150	E	26524	I-O	1,98	3,08	0,54	0,56	2,05	18,2	7831
			Odw.	1,69	2,94	0,46	0,56	1,66	24,4	7834
kroB150	E	26130	I-O	1,15	2,32	0,04	0,44	1,02	17,2	6722
			Odw.	0,92	1,87	0,15	0,28	0,85	26,7	6227
pr152	E	73682	I-O	0,96	2,23	0,55	0,32	0,84	21,5	5979
			Odw.	0,89	2,07	0,21	0,29	0,79	27,5	5527
u159	E	42080	I-O	4,85	9,89	1,78	1,50	4,84	12,6	7842
			Odw.	3,27	8,14	0,43	1,54	3,20	22,4	7632
rat195	E	2323	I-O	1,91	2,84	0,65	0,46	1,89	24,3	7565
			Odw.	1,72	2,71	0,39	0,47	1,81	34,7	6949
d198	E	15780	I-O	0,73	1,65	0,07	0,33	0,67	25,8	7902
			Odw.	0,61	2,08	0,20	0,33	0,54	32,8	8010
kroA200	E	29368	I-O	0,72	1,62	0,43	0,22	0,67	27,3	5501
			Odw.	0,70	1,45	0,45	0,21	0,64	35,3	6532
kroB200	E	29437	I-O	3,20	5,31	1,42	0,76	3,21	24,0	7962
			Odw.	2,82	5,00	1,27	0,64	2,73	31,3	8133
ts225	E	126643	I-O	1,35	1,93	0,70	0,25	1,35	22,2	6819
			Odw.	1,15	1,66	0,57	0,26	1,14	34,2	7240
tsp225	E	3919	I-O	2,08	3,96	0,79	0,71	2,04	27,0	8235
			Odw.	1,69	3,19	0,36	0,66	1,51	36,0	8000
pr226	E	80369	I-O	2,23	3,09	1,50	0,30	2,23	31,6	8331
			Odw.	2,12	3,64	1,39	0,39	2,06	40,7	8164
gil262	E	2378	I-O	3,81	5,26	2,40	0,59	3,87	36,3	8532
			Odw.	3,51	5,42	1,68	0,61	3,55	43,0	8476
pr264	E	49135	I-O	6,48	8,79	2,61	1,09	6,39	25,8	7421
			Odw.	5,84	8,57	1,45	1,44	6,09	36,3	8044
a280	E	2579	I-O	3,31	5,43	1,59	0,82	3,26	33,5	8621
			Odw.	2,80	4,58	0,70	0,73	2,79	42,6	8029
pr299	E	48191	I-O	4,91	7,03	2,26	0,90	5,00	31,4	9133
			Odw.	4,31	6,18	1,87	0,75	4,32	44,4	9021
lin318	E	42029	I-O	3,30	5,85	1,44	0,77	3,20	38,0	9269
			Odw.	2,94	4,62	1,51	0,66	2,96	48,1	9108

Nazwa	Odl.	Koszt	Op.	Śr.	Maks.	Min.	Odch.	Med.	Czas	Iter.
linhp318	E	41345	I-O	5,17	7,24	3,22	0,79	5,09	35,1	9301
			Odw.	4,62	6,33	3,23	0,63	4,67	48,7	9122
rd400	E	15281	I-O	4,11	5,64	2,28	0,60	4,10	47,2	9320
			Odw.	3,61	4,91	2,30	0,54	3,55	61,4	9162
fl417	E	11861	I-O	4,51	8,80	3,79	0,56	4,41	52,8	9051
			Odw.	4,16	6,95	3,29	0,37	4,16	71,8	8864
pr439	E	107217	I-O	5,17	7,36	3,42	0,97	4,95	51,0	9151
			Odw.	4,78	7,23	3,16	0,92	4,57	67,0	9023
pcb442	E	50778	I-O	2,99	4,22	1,98	0,56	2,92	53,3	9380
			Odw.	2,73	3,84	1,31	0,50	2,73	66,9	9333
d493	E	35002	I-O	3,19	4,83	2,18	0,53	3,23	60,6	9377
			Odw.	2,84	3,88	2,06	0,39	2,80	73,6	9426
u574	E	36905	I-O	6,25	8,18	3,69	0,82	6,29	62,9	9583
			Odw.	5,80	7,55	4,49	0,73	5,71	75,0	9554
rat575	E	6773	I-O	4,51	5,94	2,94	0,57	4,55	68,1	9337
			Odw.	4,20	5,39	2,86	0,48	4,19	83,2	9325
p654	E	34643	I-O	4,90	7,53	2,21	0,98	4,96	69,6	9627
			Odw.	4,44	6,65	2,39	0,95	4,41	84,2	9478
d657	E	48912	I-O	6,03	7,56	4,25	0,73	6,05	81,3	9652
			Odw.	5,43	7,36	4,00	0,66	5,39	97,6	9535
u724	E	41910	I-O	5,74	7,52	4,11	0,68	5,73	86,7	9707
			Odw.	5,27	6,61	4,10	0,50	5,23	102,8	9558
rat783	E	8806	I-O	6,07	7,57	4,62	0,57	6,02	88,3	9689
			Odw.	5,44	6,63	4,35	0,46	5,37	108,0	9626
pr1002	E	259045	I-O	8,42	10,80	6,61	0,82	8,30	103,0	9811
			Odw.	7,84	9,64	6,10	0,77	7,87	124,6	9791
u1060	E	224094	I-O	10,55	13,00	7,92	0,96	10,58	85,4	9824
			Odw.	9,42	11,68	7,70	0,84	9,34	108,9	9829
vm1084	E	239297	I-O	9,76	11,79	6,91	0,85	9,68	114,5	9690
			Odw.	9,04	11,21	6,75	0,87	9,05	140,4	9691
pcb1173	E	56892	I-O	9,92	11,90	8,32	0,76	9,84	121,4	9844
			Odw.	9,05	10,40	7,79	0,54	9,05	145,2	9826
d1291	E	50801	I-O	9,68	11,49	7,47	0,80	9,77	120,0	9567
			Odw.	9,21	11,96	7,27	0,89	9,22	140,3	9522
rl1304	E	252948	I-O	10,48	12,28	8,24	0,83	10,61	137,4	9733
			Odw.	9,72	11,93	7,18	0,94	9,75	160,8	9677
rl1323	E	270199	I-O	8,43	10,44	6,40	0,87	8,41	125,6	6907
			Odw.	7,86	9,96	6,16	0,80	7,86	148,3	9630
nrw1379	E	56638	I-O	9,48	11,73	8,03	0,67	9,45	125,9	9885
			Odw.	8,65	10,36	7,12	0,65	8,69	156,7	9841
fl1400	E	20127	I-O	10,88	15,96	7,13	1,86	10,73	148,9	9919
			Odw.	9,33	14,27	5,58	1,60	9,01	178,3	9915

Nazwa	Odl.	Koszt	Op.	Śr.	Maks.	Min.	Odch.	Med.	Czas	Iter.
u1432	E	152970	I-O	10,71	12,09	9,29	0,67	10,71	138,1	9859
			Odw.	9,84	11,08	7,88	0,64	9,84	171,7	9841
fl1577	E	22204	I-O	12,50	14,39	9,60	1,07	12,67	129,2	9766
			Odw.	11,52	14,34	9,03	1,20	11,63	157,7	9797
d1655	E	62128	I-O	10,30	11,95	8,44	0,82	10,33	163,2	9753
			Odw.	9,64	11,24	7,20	0,87	9,72	194,2	9721
vm1748	E	336556	I-O	9,22	13,14	7,12	1,16	8,97	153,4	9852
			Odw.	8,40	10,80	6,52	0,80	8,27	179,8	9780
u1817	E	57201	I-O	9,47	11,87	8,17	0,67	9,40	176,7	9798
			Odw.	8,84	10,42	6,65	0,72	8,90	210,1	9808
rl1889	E	316536	I-O	11,33	12,88	8,99	0,73	11,31	174,9	9795
			Odw.	10,64	12,41	9,29	0,65	10,62	217,4	9817
d2103	E	79952	I-O	5,77	6,67	4,92	0,37	5,83	447,5	9631
			Odw.	6,02	7,27	4,86	0,43	6,05	303,2	9604
u2152	E	64253	I-O	10,67	12,33	9,38	0,49	10,68	419,4	9840
			Odw.	11,21	12,63	9,88	0,52	11,15	280,7	9833
u2319	E	234256	I-O	7,45	8,60	6,26	0,45	7,38	395,6	9860
			Odw.	7,97	8,98	6,54	0,46	7,96	274,5	9905
pr2392	E	378032	I-O	13,64	14,91	12,08	0,63	13,69	452,3	9910
			Odw.	14,30	15,62	12,53	0,60	14,35	322,0	9889
pcb3038	E	137694	I-O	13,37	14,68	11,92	0,57	13,37	642,5	9953
			Odw.	14,25	15,38	12,66	0,61	14,37	442,0	9942
fl3795	E	28723	I-O	16,68	18,92	14,26	0,86	16,62	736,5	9921
			Odw.	17,30	18,95	14,73	0,86	17,33	540,6	9932
fnl4461	E	182566	I-O	14,49	15,56	13,51	0,42	14,48	828,9	9956
			Odw.	15,06	15,80	13,77	0,42	15,09	611,0	9961
rl5915	E	565040	I-O	14,64	15,64	12,42	0,66	14,79	1116,2	9908
			Odw.	15,16	16,61	13,64	0,62	15,20	846,3	9912
rl5934	E	554070	I-O	14,94	16,34	13,60	0,44	14,97	1193,9	9925
			Odw.	15,41	16,30	14,08	0,43	15,47	882,2	9920
burma14	G	3323	I-O	3,94	3,94	3,94	0	3,94	63,4	22
			Odw.	3,94	3,94	3,94	0	3,94	61,9	15
ulyssess16	G	6859	I-O	0,92	0,92	0,92	0	0,92	92,3	77
			Odw.	0,92	0,92	0,92	0	0,92	106,5	116
ulyssess22	G	7013	I-O	1,17	1,97	1,15	0,12	1,15	94,8	762
			Odw.	1,21	1,97	1,15	0,21	1,15	127,5	552
gr96	G	55209	I-O	1,63	3,40	1,52	0,31	1,52	1754,1	3696
			Odw.	1,63	3,40	1,52	0,31	1,52	1231,9	5019
gr137	G	69853	I-O	1,06	1,62	0,42	0,25	1,13	2674,7	5654
			Odw.	1,13	2,08	0,53	0,25	1,13	1818,2	6716
gr202	G	40160	I-O	3,10	3,94	2,38	0,29	3,15	3592,5	7228
			Odw.	3,30	3,86	2,38	0,29	3,31	2562,7	7817

Nazwa	Odl.	Koszt	Op.	Śr.	Maks.	Min.	Odch.	Med.	Czas	Iter.
gr229	G	134602	I-O	1,13	1,68	0,49	0,24	1,16	4241,2	7805
			Odw.	1,26	2,48	0,34	0,35	1,26	3013,3	8002
gr431	G	171414	I-O	4,22	5,69	2,26	0,69	4,20	911,3	9392
			Odw.	3,68	4,94	2,43	0,57	3,69	1147,7	9297
ali535	G	202310	I-O	6,84	9,55	3,97	1,14	6,90	891,2	9672
			Odw.	5,91	11,34	3,81	1,19	5,69	1148,9	9529
gr666	G	294358	I-O	6,99	8,16	5,44	0,54	7,02	1450,0	9726
			Odw.	6,40	7,76	4,81	0,58	6,42	1736,2	9635
gr17	M	2085	I-O	0	0	0	0	0	16,0	11
			Odw.	0	0	0	0	0	21,7	8
gr21	M	2707	I-O	0	0	0	0	0	18,3	68
			Odw.	0	0	0	0	0	25,1	50
gr24	M	1272	I-O	0	0	0	0	0	15,3	102
			Odw.	0	0	0	0	0	19,3	84
fri26	M	937	I-O	0	0	0	0	0	13,0	66
			Odw.	0	0	0	0	0	19,6	45
bayg29	M	1610	I-O	0	0	0	0	0	18,8	430
			Odw.	0	0	0	0	0	26,7	296
bays29	M	2020	I-O	0	0	0	0	0	21,0	50
			Odw.	0	0	0	0	0	28,9	36
dantzig42	M	699	I-O	0	0	0	0	0	26,6	258
			Odw.	0	0	0	0	0	39,2	200
swiss42	M	1273	I-O	0	0	0	0	0	28,3	148
			Odw.	0	0	0	0	0	40,7	113
gr48	M	5046	I-O	0,10	0,40	0	0,12	0,06	25,3	4003
			Odw.	0,06	0,63	0	0,11	0	38,6	4448
hk48	M	11461	I-O	0	0	0	0	0	26,3	292
			Odw.	0	0,08	0	0,02	0	40,1	214
brazil58	M	25395	I-O	0	0	0	0	0	35,4	527
			Odw.	0	0	0	0	0	44,4	377
si535	M	48450	I-O	1,34	1,75	0,98	0,17	1,34	91,3	9394
			Odw.	1,23	1,58	0,90	0,15	1,24	104,6	9317
pa561	M	2763	I-O	4,95	6,84	2,79	0,68	4,85	91,2	9287
			Odw.	4,75	6,30	3,18	0,53	4,71	111,1	9297
si1032	M	92650	I-O	0,81	1,06	0,55	0,09	0,81	152,0	9198
			Odw.	0,77	0,94	0,51	0,07	0,77	191,7	9379

Tabela 20

Zestawienie wyników dla operatorów *inver-over* i odwracania – asymetryczny problem komiwojażera

Nazwa	Odl.	Koszt	Op.	Śr.	Maks.	Min.	Odch.	Med.	Czas	Iter.
br17	M	39	I-O	0	0	0	0	0	14,3	6
			Odw.	0	0	0	0	0	19,2	5
flv33	M	1286	I-O	2,38	4,59	0	1,48	2,95	15,4	6531
			Odw.	1,81	5,68	0	1,62	2,33	21,7	6608
flv35	M	1473	I-O	0,01	0,41	0	0,06	0	16,8	3709
			Odw.	0,02	1,15	0	0,12	0	23,2	3420
flv38	M	1530	I-O	0,48	1,90	0	0,51	0,39	16,9	5444
			Odw.	0,46	1,44	0	0,51	0,39	23,1	5436
p43	M	5620	I-O	0	0,02	0	0,01	0	19,9	1388
			Odw.	0	0,02	0	0,01	0	28,2	1561
flv44	M	1613	I-O	0,60	2,11	0	0,40	0,62	18,3	6571
			Odw.	0,49	1,18	0	0,29	0,62	26,1	5670
flv47	M	1776	I-O	2,05	3,94	0	0,77	1,91	21,7	8349
			Odw.	1,67	3,77	0	0,82	1,75	27,6	8146
ry48p	M	14422	I-O	0,55	1,28	0,17	0,16	0,59	22,1	4675
			Odw.	0,54	1,23	0,17	0,16	0,59	28,5	4373
ft53	M	6905	I-O	6,25	10,98	3,98	1,57	5,97	19,8	8474
			Odw.	5,82	9,91	3,32	1,58	5,39	27,4	8382
flv55	M	1608	I-O	4,35	7,96	1,68	1,74	4,73	21,8	7637
			Odw.	3,74	7,34	1,37	1,65	3,86	30,9	7887
flv64	M	1839	I-O	2,77	5,82	1,25	0,67	2,45	24,4	8128
			Odw.	2,62	5,44	0,92	0,53	2,45	34,8	7706
ft70	M	38673	I-O	2,36	3,48	1,10	0,52	2,43	27,0	8356
			Odw.	2,25	4,58	1,10	0,54	2,20	36,8	8124
flv170	M	2755	I-O	25,23	33,36	18,58	2,63	25,06	13,6	8007
			Odw.	23,84	28,60	19,31	1,97	23,45	18,2	8295
rbg323	M	1326	I-O	27,92	29,34	26,55	0,55	27,90	26,2	7855
			Odw.	27,73	28,96	26,02	0,54	27,75	36,5	7982
rbg358	M	1163	I-O	48,06	50,13	45,31	0,84	47,98	30,9	8957
			Odw.	47,14	49,53	44,37	0,90	47,16	40,4	8876
rbg403	M	2465	I-O	36,88	38,70	34,16	0,76	36,92	39,6	9561
			Odw.	35,59	37,57	33,96	0,67	35,62	40,0	9541
rbg443	M	2720	I-O	38,34	39,63	36,95	0,48	38,36	34,4	9661
			Odw.	37,14	38,42	35,66	0,54	37,17	46,7	9686

A.3. Problem określania kolejności złączeń w realizacji zapytań

Niniejszy załącznik zawiera szczegółowe zestawienie rezultatów badań porównawczych planów realizacji zapytań wyznaczonych przez wbudowany optymalizator systemu Microsoft

SQL Server 2008 oraz planów zbudowanych przez zmodyfikowaną autorską wersję algorytmu IWO.

Wyniki eksperymentów zgrupowano w 10 tabelach. Tabele 21-26 zawierają rezultaty badań dotyczących zapytań z grafem o charakterze gwiazdy, tabele 27 i 28 – zapytań z grafem łańcuchowym, natomiast 29 oraz 30 – dla grafów nieregularnych.

Wszystkie tabele składają się z następujących kolumn:

- baza – oznaczenie zestawu danych;
- optymalizator – oznaczenie, czy optymalizacja była wykonana przy użyciu metody wbudowanej w system Microsoft SQL Server 2008, czy z wykorzystaniem algorytmu IWO;
- koszt – wartość jest średnim szacunkowym kosztem realizacji zapytania, wyznaczonym przez optymalizator Microsoft SQL Server 2008 przed wykonaniem zapytania;
- czas proc. – średni czas procesora, wyrażony w sekundach, jest czasem zajęcia procesora obliczeniami niezbędnymi do wykonania zapytania;
- odch. czasu proc. – średnie odchylenie standardowe od wartości czasu procesora;
- liczba odcz. – średnia liczba odczytów z dysku;
- odch. liczby odcz. – średnie odchylenie standardowe od liczby odczytów;
- czas real. – średni czas realizacji zapytania, jest to wartość najważniejsza i oznacza czas oczekiwania użytkownika na wynik zapytania;
- odch. czasu real. – średnie odchylenie standardowe od wartości czasu realizacji;
- ostatnia kolumna zawiera oznaczenie graficzne wskazujące, czy autorska zmodyfikowana wersja algorytmu IWO jest lepsza (✓) lub gorsza (✗) od metody wbudowanej w system SQL Server 2008.

Tabela 21

Porównanie rezultatów badań dla małych baz z grafem złączeń o charakterze gwiazdy z liczbą 10 000 wierszy w tabeli faktów

Baza	Optymalizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
G 10 M 1	SQL Server	10,58	3,92	0,09	1 183 451	80	6,50	0,19	✓
	IWO	14,05	3,47	0,11	1 087 751	82	6,16	0,28	
G 10 M 2	SQL Server	9,86	3,57	0,17	1 135 878	66	6,27	0,21	✓
	IWO	11,77	3,25	0,11	1 078 454	71	6,09	0,41	
G 10 M 3	SQL Server	10,47	2,98	0,12	1 141 339	69	5,61	0,15	✓
	IWO	13,17	2,75	0,07	1 073 201	67	5,25	0,13	
G 10 M 4	SQL Server	10,56	2,84	0,06	1 077 142	42	5,18	0,17	✗
	IWO	14,13	2,81	0,07	1 072 852	64	5,18	0,24	
G 10 M 5	SQL Server	10,73	3,10	0,06	1 139 456	98	5,53	0,20	✓
	IWO	10,55	2,93	0,07	1 099 087	104	5,43	0,13	

Należy podkreślić, że wszystkie eksperymenty wykonywano wielokrotnie. Dla każdego zestawu danych przygotowano 10 planów realizacji skonstruowanych za pomocą algorytmu IWO oraz 1 zbudowany przez metodę wbudowaną w system SQL Server 2008. Obliczenia przeprowadzono dziesięć razy przy użyciu każdego z planów, stąd wszystkie wyniki w tabelach są średnimi wynikami osiągniętymi podczas realizacji opisanych uruchomień.

Tabela 22

Porównanie rezultatów badań dla dużych baz z grafem złączeń o charakterze gwiazdy z liczbą 10 000 wierszy w tabeli faktów

Baza	Optymalizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
G 10 D 1	SQL Server	39,14	6,38	0,22	1 599 609	278	20,34	0,41	✓
	IWO	54,20	5,83	0,12	1 527 288	325	19,31	0,60	
G 10 D 2	SQL Server	33,85	7,04	0,14	1 676 036	229	19,52	0,18	✓
	IWO	49,38	6,11	0,14	1 558 670	277	18,18	0,32	
G 10 D 3	SQL Server	32,41	7,38	0,37	1 613 494	173	18,63	2,30	✓
	IWO	51,57	6,02	0,27	1 438 902	215	15,97	0,70	
G 10 D 4	SQL Server	26,68	8,87	0,32	1 724 488	379	24,39	0,93	✓
	IWO	56,15	6,30	0,22	1 485 814	463	20,05	0,51	
G 10 D 5	SQL Server	25,59	9,04	0,42	1 733 189	310	22,27	0,61	✓
	IWO	50,00	6,12	0,20	1 483 381	311	17,38	0,57	

Tabela 23

Porównanie rezultatów badań dla małych baz z grafem złączeń o charakterze gwiazdy z liczbą 100 000 wierszy w tabeli faktów

Baza	Optymalizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
G 100 M 1	SQL Server	73,69	27,73	0,94	11 143 619	54	36,82	0,64	✓
	IWO	101,09	25,98	0,45	10 378 441	56	34,39	0,65	
G 100 M 2	SQL Server	76,31	30,60	0,48	11 583 527	43	38,79	0,53	✓
	IWO	97,53	27,98	0,41	10 797 214	69	36,40	0,34	
G 100 M 3	SQL Server	79,32	42,55	0,63	13 775 927	133	52,08	0,94	✓
	IWO	108,43	35,04	0,46	11 586 979	99	44,36	0,42	
G 100 M 4	SQL Server	76,56	35,94	0,22	12 072 218	90	43,63	0,22	✓
	IWO	106,31	30,97	0,26	10 563 255	89	38,42	0,22	
G 100 M 5	SQL Server	78,36	26,34	0,34	10 085 978	46	33,76	0,29	✗
	IWO	93,42	26,36	0,30	10 086 019	38	33,92	0,43	

Tabela 24
Porównanie rezultatów badań dla dużych baz z grafem złączeń o charakterze gwiazdy z liczbą 100 000 wierszy w tabeli faktów

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
G 100 D 1	SQL Server	219,83	40,19	0,68	12 611 052	274	62,53	0,88	✓
	IWO	270,16	38,88	0,44	12 146 647	836	59,85	0,55	
G 100 D 2	SQL Server	133,98	39,43	0,63	13 188 856	214	54,31	0,64	✓
	IWO	288,98	35,92	0,48	11 658 438	236	50,41	0,43	
G 100 D 3	SQL Server	230,21	39,18	0,69	12 099 503	205	57,85	0,81	✓
	IWO	470,55	37,77	0,53	11 312 558	386	56,35	0,86	
G 100 D 4	SQL Server	303,92	36,81	0,50	11 329 994	266	55,72	1,32	✓
	IWO	538,46	36,09	0,34	11 117 530	283	54,24	0,55	
G 100 D 5	SQL Server	192,95	42,46	0,68	12 928 461	287	64,53	0,74	✓
	IWO	388,82	38,69	0,28	11 891 436	1236	58,97	0,35	

Tabela 25
Porównanie rezultatów badań dla małych baz z grafem złączeń o charakterze gwiazdy z liczbą 1 000 000 wierszy w tabeli faktów

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
G 1000 M 1	SQL Server	759,34	294,24	7,46	121 343 791	108	358,24	6,91	✓
	IWO	888,97	243,52	4,95	104 556 085	128	306,10	5,27	
G 1000 M 2	SQL Server	729,27	297,17	4,91	113 815 752	95	364,70	5,92	✓
	IWO	1 149,39	252,27	2,55	101 065 551	69	318,02	2,30	
G 1000 M 3	SQL Server	764,47	343,70	1,63	120 086 994	93	408,67	1,61	✓
	IWO	893,06	299,44	3,65	107 336 792	86	363,00	2,65	
G 1000 M 4	SQL Server	771,91	318,65	5,61	115 955 514	88	388,01	5,08	✓
	IWO	846,98	289,10	3,33	108 092 773	83	360,65	3,29	
G 1000 M 5	SQL Server	754,25	296,95	8,97	103 562 342	46	359,53	6,04	✓
	IWO	1 011,00	280,22	3,33	101 437 260	82	342,46	5,29	

Tabela 26

Porównanie rezultatów badań dla dużych baz z grafem złączeń o charakterze gwiazdy z liczbą 1 000 000 wierszy w tabeli faktów

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
G 1000 D 1	SQL Server	1 721,53	352,72	6,08	113 247 353	381	427,42	6,45	✓
	IWO	2 779,96	351,36	4,88	113 247 583	388	425,82	5,27	
G 1000 D 2	SQL Server	1 142,63	333,30	6,12	113 714 134	273	416,37	6,47	✓
	IWO	2 572,10	333,82	4,82	113 714 119	421	415,90	4,71	
G 1000 D 3	SQL Server	3 057,76	349,31	4,58	113 261 777	291	428,95	5,09	✗
	IWO	3 616,31	351,19	3,54	113 261 978	292	431,20	3,95	
G 1000 D 4	SQL Server	1 901,91	353,79	7,00	105 691 995	175	426,45	5,30	✓
	IWO	3 324,15	350,32	4,40	105 692 276	223	423,09	4,03	
G 1000 D 5	SQL Server	3 488,74	375,51	3,32	111 261 066	280	451,85	4,29	✓
	IWO	4 579,22	356,67	2,71	111 262 210	760	430,15	2,77	

Tabela 27

Porównanie rezultatów badań dla małych baz z grafem złączeń o charakterze łańcucha

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
Ł M 1	SQL Server	5 888,03	76,88	1,49	6 232 715	237	132,65	1,11	✗
	IWO	1 421,60	84,56	1,16	6 541 234	200	140,37	1,36	
Ł M 2	SQL Server	512,01	63,25	2,01	5 321 531	78	102,64	1,20	✗
	IWO	2 612,95	71,38	1,67	10 533 864	169	110,05	0,90	
Ł M 3	SQL Server	46,45	72,35	0,61	5 649 875	21	110,24	0,69	✓
	IWO	4 741,43	71,68	1,04	9 446 895	145	110,13	1,25	
Ł M 4	SQL Server	20,77	16,08	0,25	1 083 995	62	27,79	0,29	✓
	IWO	50,08	13,25	0,35	692 987	79	24,90	0,31	
Ł M 5	SQL Server	687,65	1610,01	31,46	$1{,}289 \cdot 10^9$	129	1641,94	31,83	✓
	IWO	76,28	48,76	0,67	11 570 923	105	68,96	0,73	
Ł M 6	SQL Server	13 388,70	2086,04	18,10	$1{,}939 \cdot 10^9$	101	2108,95	17,93	✓
	IWO	4 129,48	60,23	0,48	34 023 095	82	73,67	0,37	
Ł M 7	SQL Server	1 843,43	79,32	1,26	6 996 508	297	131,60	1,24	✗
	IWO	520,29	86,68	1,41	14 661 013	173	138,96	1,61	
Ł M 8	SQL Server	83,44	69,52	0,53	7 677 734	162	113,04	0,50	✗
	IWO	1 829,84	98,84	0,85	31 886 412	129	142,68	1,23	
Ł M 9	SQL Server	271,53	81,38	1,03	27 217 152	231	108,16	0,96	✓
	IWO	1 754,13	65,95	0,77	16 580 430	157	92,54	1,11	
Ł M 10	SQL Server	110,25	52,18	0,76	4 824 122	66	81,83	0,73	✗
	IWO	53,49	65,62	0,64	12 776 570	101	95,62	0,78	

Tabela 28

Porównanie rezultatów badań dla dużych baz z grafem złączeń o charakterze łańcucha

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
Ł D 1	SQL Server	60 090,9	122,26	1,20	69 414 916	590	148,92	1,11	✓
	IWO	124 019,1	113,41	1,15	54 458 103	919	141,86	1,54	
Ł D 2	SQL Server	109 890,0	37,23	0,18	8 907 891	494	55,05	0,31	✗
	IWO	581 757,3	47,75	0,32	207 768 854	713	65,89	0,28	
Ł D 3	SQL Server	149 728,0	78,14	1,09	11 840 188	488	113,13	2,69	✓
	IWO	57 985,3	75,91	0,94	10 800 333	625	110,94	1,18	
Ł D 4	SQL Server	335 218,0	65,97	0,74	9 118 457	663	109,32	0,72	✗
	IWO	190 625,2	82,69	0,73	11 505 207	934	124,80	0,87	
Ł D 5	SQL Server	701 533,0	10,85	0,59	26 730 118	584	140,34	0,85	✓
	IWO	2 676 959,0	94,86	0,84	27 953 987	558	135,90	0,96	
Ł D 6	SQL Server	95 384,1	228,15	1,58	87 494 505	551	279,95	1,39	✗
	IWO	157 325,3	239,10	1,82	93 974 162	456	291,44	2,31	
Ł D 7	SQL Server	2 459 030,0	161,60	0,60	42 794 774	344	215,83	1,58	✓
	IWO	3 741 921,0	133,97	1,41	30 985 982	863	188,34	1,55	
Ł D 8	SQL Server	98 098,3	83,44	1,09	4 864 035	809	126,02	0,90	✗
	IWO	36 392,5	85,79	0,71	15 722 266	888	126,03	0,89	
Ł D 9	SQL Server	9 034 820	144,29	1,84	26 809 954	820	208,32	1,95	✗
	IWO	23 008 320	146,32	1,73	26 845 922	806	212,48	1,76	
Ł D 10	SQL Server	187 058,0	175,74	1,94	105 135 282	723	206,65	1,84	✗
	IWO	990 055,9	184,96	1,72	95 259 251	781	214,87	1,60	

Tabela 29

Porównanie rezultatów badań dla małych baz z grafem nieregularnym

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
N M 1	SQL Server	9,85	62,03	1,10	5 662 077	32	103,97	0,82	✗
	IWO	70,55	63,13	1,11	4 052 830	52	105,64	1,17	
N M 2	SQL Server	172,91	54,51	1,22	15 437 628	248	73,69	0,85	✓
	IWO	865,04	49,42	0,85	16 944 536	121	69,11	0,89	
N M 3	SQL Server	128,69	109,78	0,49	49 141 972	115	152,65	1,22	✓
	IWO	325,01	82,07	0,92	14 310 227	88	122,12	0,78	
N M 4	SQL Server	264,32	96,98	1,99	2 625 366	25	149,63	1,84	✓
	IWO	619,30	95,86	1,48	11 414 714	105	148,73	1,37	
N M 5	SQL Server	347,85	21,93	0,55	687 236	144	35,45	0,33	✓
	IWO	1 550,99	13,79	0,35	597 019	95	26,69	0,27	
N M 6	SQL Server	302,54	26,78	0,52	3 046 699	184	41,12	0,36	✗
	IWO	573,14	33,91	0,48	8 991 769	115	48,61	0,44	

Tabela 29

Porównanie rezultatów badań dla małych baz z grafem nieregularnym

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
N M 7	SQL Server	3 629,72	130,22	1,91	77 843 251	85	163,33	1,48	✓
	IWO	22 957,58	70,58	1,23	21 255 842	64	103,26	1,10	
N M 8	SQL Server	76,62	58,75	1,59	5 087 660	192	95,90	0,65	✗
	IWO	90,80	62,31	1,14	2 516 725	154	98,61	1,02	
N M 9	SQL Server	131,15	80,57	1,18	3 594 954	96	135,92	0,83	✗
	IWO	333,58	89,05	1,16	11 140 933	155	146,29	1,36	
N M 10	SQL Server	428,60	64,60	1,49	1 145 567	56	98,90	0,77	✗
	IWO	1 030,72	70,52	0,90	5 847 076	78	106,26	0,80	
N M 11	SQL Server	149,76	34,58	0,59	6 430 701	140	51,17	0,80	✗
	IWO	42,22	35,45	0,56	4 080 956	187	52,08	0,64	
N M 12	SQL Server	114,05	53,22	1,98	12 898 638	70	80,14	0,49	✓
	IWO	1 602,00	47,69	0,96	8 043 031	65	75,26	1,11	
N M 13	SQL Server	2 738,41	1 109,10	5,17	358 312 473	205	1 136,87	5,29	✓
	IWO	4 174,54	61,13	0,78	11 690 474	173	86,79	0,92	
N M 14	SQL Server	31,85	90,54	1,20	12 105 010	122	137,24	1,24	✓
	IWO	40,23	89,49	1,75	6 926 464	100	136,12	1,36	
N M 15	SQL Server	57 103,00	129,08	1,51	86 863 378	73	139,22	1,38	✓
	IWO	7 802,70	25,75	0,35	10 994 138	73	35,21	0,33	
N M 16	SQL Server	1 123,30	188,89	1,86	56 998 831	428	228,24	1,87	✗
	IWO	15 069,15	218,38	1,70	120 653 101	143	257,90	1,88	
N M 17	SQL Server	4 450,68	20,54	0,46	5 463 393	20	29,39	0,34	✗
	IWO	2 964,59	31,51	0,34	14 375 447	149	40,24	0,44	
N M 18	SQL Server	44 082,90	603,70	4,37	136 009 478	60	629,29	4,05	✓
	IWO	4 639,95	160,66	1,37	103 897 327	84	183,77	1,42	
N M 19	SQL Server	27,84	23,06	0,60	670 935	43	37,25	0,48	✓
	IWO	209,00	20,48	0,60	1 406 044	78	34,59	0,73	
N M 20	SQL Server	303,89	263,56	4,60	155 528 758	35	317,47	3,38	✓
	IWO	100,35	100,94	1,19	20 138 262	122	151,06	1,08	

Tabela 30

Porównanie rezultatów badań dla dużych baz z grafem nieregularnym

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
N M 1	SQL Server	198 285	120,37	1,04	15 725 953	464	167,68	2,14	✗
	IWO	269 535	124,37	2,06	18 530 816	422	168,46	2,04	
N D 2	SQL Server	3 695 190	68,30	2,10	8 869 112	1 117	109,07	2,56	✗
	IWO	17 797 235	83,83	1,52	16 615 590	807	121,77	1,31	

Tabela 30

Porównanie rezultatów badań dla dużych baz z grafem nieregularnym

Baza	Optyma-lizator	Koszt	Czas proc.	Odch. czasu proc.	Liczba odcz.	Odch. liczby odcz.	Czas real.	Odch. czasu real.	✓ ✗
N D 3	SQL Server	20 987	61,85	1,97	2 800 463	647	95,86	1,03	✗
	IWO	101 439	65,16	0,90	3 649 355	722	95,98	1,15	
N D 4	SQL Server	2 544 620	134,76	1,24	29 697 057	996	191,38	1,31	✗
	IWO	2 060 918	137,26	1,41	24 216 711	619	192,09	1,67	
N D 5	SQL Server	103 625	134,00	1,07	13 857 052	906	202,05	1,55	✓
	IWO	2 700 522	132,91	0,90	19 457 211	813	199,80	0,91	
N D 6	SQL Server	1 694 890	19,70	0,62	1 514 453	359	37,56	0,90	✓
	IWO	5 465 440	22,47	0,56	2 068 847	230	36,93	1,04	
N D 7	SQL Server	190 556	85,65	0,59	26 371 654	1 145	126,01	1,82	✓
	IWO	498 563	64,73	0,52	10 146 364	628	99,65	0,58	
N D 8	SQL Server	45 298	47,28	0,75	5 970 308	849	71,00	1,18	✓
	IWO	52 614	40,79	1,22	3 912 262	428	63,78	1,16	
N D 9	SQL Server	592 556	100,62	0,89	5 849 199	321	148,71	0,74	✗
	IWO	948 023	105,89	2,10	16 586 565	905	154,82	1,87	
N D 10	SQL Server	1 825 860	54,91	1,09	6 026 391	92	93,12	1,27	✗
	IWO	54 079 944	81,86	0,95	28 613 650	720	112,06	1,24	
N D 11	SQL Server	499 593	160,89	0,87	101 141 904	396	184,49	0,95	✓
	IWO	1 375 480	40,94	0,77	8 608 735	548	62,49	0,71	
N D 12	SQL Server	480 584	126,45	4,32	19 406 820	196	176,27	2,88	✓
	IWO	92 612	90,25	2,14	5 460 173	471	141,04	1,74	
N D 13	SQL Server	2 000	21,29	0,25	3 776 502	614	35,28	0,17	✗
	IWO	4 330	25,20	0,28	7 857 525	643	41,24	0,46	
N D 14	SQL Server	621 309	66,34	1,77	5 680 362	2 031	111,58	2,35	✗
	IWO	523 824	72,48	1,49	4 900 735	1 363	115,17	2,53	
N D 15	SQL Server	14 649	128,89	1,07	38 225 246	937	180,44	1,33	✓
	IWO	83 125	101,07	1,27	19 139 102	839	149,22	1,55	
N D 16	SQL Server	35 823 900	145,84	1,19	8 296 435	737	209,41	1,27	✓
	IWO	28 362 745	130,90	1,01	16 689 653	915	197,24	0,89	
N D 17	SQL Server	1 289 240	270,58	1,61	141 404 947	1 077	314,21	1,46	✓
	IWO	1 226 797	154,49	1,12	41 976 037	548	197,86	1,44	
N D 18	SQL Server	20 066 400	69,12	1,64	3 648 511	631	110,20	1,23	✗
	IWO	7 492 954	78,87	0,98	6 387 717	417	120,42	1,43	
N D 19	SQL Server	2 758 220	44,91	1,02	11 917 913	621	68,02	0,71	✗
	IWO	304 780	51,59	0,83	14 410 328	546	74,46	0,96	
N D 20	SQL Server	177 347	60,31	1,13	4 542 997	180	92,37	2,06	✗
	IWO	65 929	72,95	1,40	10 396 807	737	108,44	1,15	

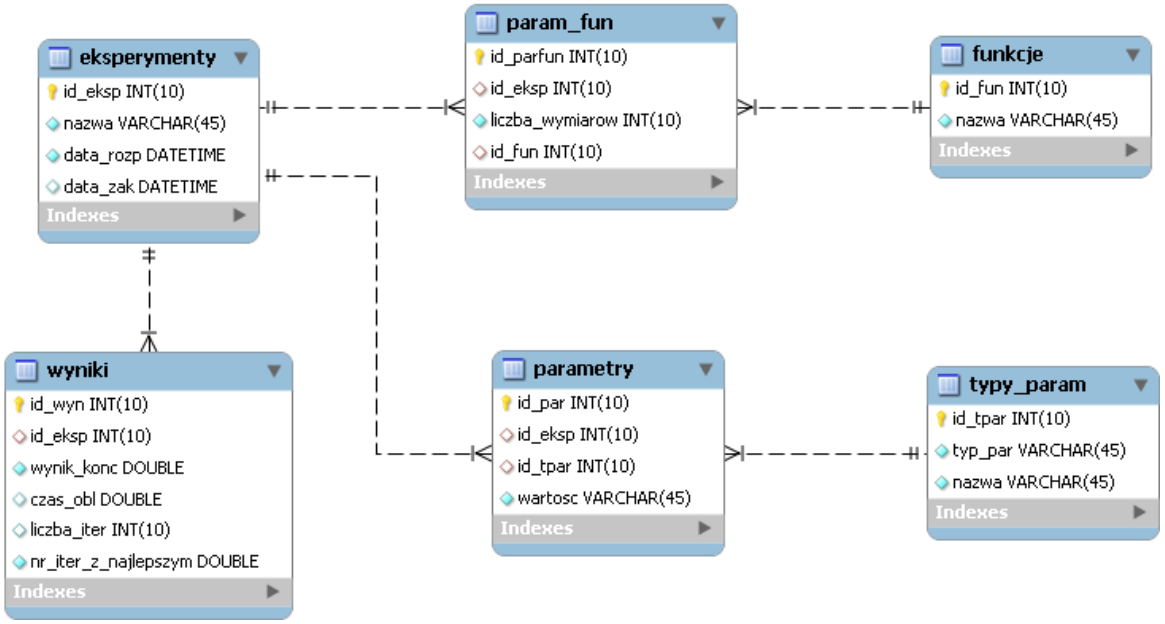
Dodatek B. Opis aplikacji badawczych

Do zrealizowania eksperymentów wymagane było stworzenie kilku programów komputerowych. Wszystkie aplikacje napisane zostały w języku Java, w środowisku Eclipse. Niniejszy dodatek zawiera ogólny opis tych programów.

B.1. Problem znajdowania minimum funkcji wielowymiarowej

Aplikacja badawcza rozwiązująca niniejszy problem optymalizacyjny ma charakter wsadowy. Oznacza to, że nie posiada ona interfejsu graficznego, tylko jest uruchamiana z linii poleceń. Wszystkie dane, jakie są wymagane przez program, są podawane przez plik typu XML. Zawiera on opis eksperymentów, jakie mają zostać wykonane przez aplikację. Na opis ten składają się m.in.: nazwa optymalizowanej funkcji, liczba wymiarów funkcji, parametry algorytmu oraz liczba powtórzeń eksperymentu przy takich samych danych. Wszystkie obliczenia wykonywane były wielokrotnie ze względu na niedeterministyczny charakter algorytmów heurystycznych.

Dla wygody użytkowania aplikacji, wyniki testów są przechowywane w bazie danych MySQL. Schemat fizyczny tej bazy przedstawiono na rys. 40. Taki sposób gromadzenia danych ułatwił agregowanie wyników obliczeń oraz zwiększył bezpieczeństwo ich przechowywania.



Rys. 40. Schemat fizyczny bazy zastosowanej w aplikacji badawczej.

Aplikacja badawcza została napisana wielowątkowo. Pozwoliło to na realizację obliczeń eksperymentalnych w sposób równoległy, wykorzystując tym samym możliwości współczesnych procesorów wielordzeniowych.

B.2. Problem komiwojażera

Program do przeprowadzenia obliczeń dotyczących problemu komiwojażera został napisany w bardzo podobny sposób jak aplikacja znajdująca minimum globalne funkcji wielowymiarowej.

Jest to program wsadowy, pobierający dane z pliku XML (rys. 41) i zapisujący wyniki obliczeń w bazie danych. Aplikacja pozwala również na realizację obliczeń równoległych.

```
<?xml version="1.0" encoding="UTF-8" ?>
<eksperymenty>
    <eksperyment>
        <nazwa_eksp>Badanie 1 - kroA100</nazwa_eksp>
        <parametry_alg>
            <algorytm>IWO</algorytm>

            <metody_alg>
                <rozprzestrzenianie>mieszane</rozprzestrzenianie>
                <rozklad>normalny</rozklad>
                <nowy_chwast>inverover</nowy_chwast>
                <selekcja>rodzinna</selekcja>
                <war_stopu>iteracje</war_stopu>
            </metody_alg>

            <param_ogolne>
                <liczba_osobnikow>200</liczba_osobnikow>
                <liczba_pokolen>10000</liczba_pokolen>
                <czas>10</czas>
            </param_ogolne>

            <param_IWO>
                <max_ziaren>5</max_ziaren>
                <min_ziaren>0</min_ziaren>
                <pocz_odch_stand>2.5</pocz_odch_stand>
                <konc_odch_stand>0.001</konc_odch_stand>
                <liczba_st_swobody>7</liczba_st_swobody>
                <pocz_gamma>10.0</pocz_gamma>
                <konc_gamma>0.1</konc_gamma>
                <wsp_nieliniowosci>0.5</wsp_nieliniowosci>
                <liczba_przejsc>2</liczba_przejsc>
                <inverover_prog>0.1</inverover_prog>
                <proc_staczanie>10</proc_staczanie>
                <proc_rozwiewanie>90</proc_rozwiewanie>
                <proc_rozsiewanie>0</proc_rozsiewanie>
            </param_IWO>
        </parametry_alg>

        <parametry_eksp>
            <tsp_prob>kroA100</tsp_prob>
            <tsp_prob_path>C:\TSP\kroA100.tsp</tsp_prob_path>
            <liczba_uruchomien>100</liczba_uruchomien>
        </parametry_eksp>
    </eksperyment>
</eksperymenty>
```

Rys. 41. Część przykładowego pliku XML z opisem eksperymentów.

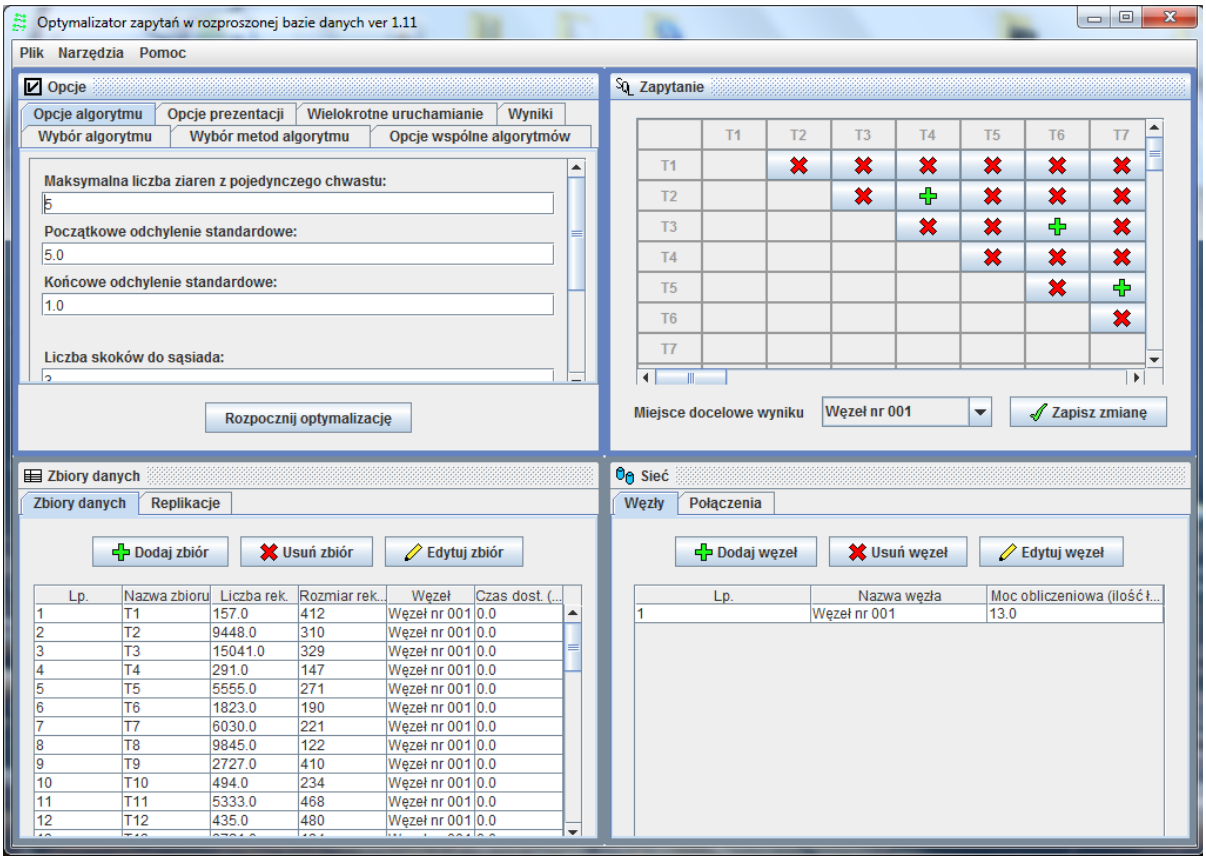
B.3. Problem określania kolejności złączeń w realizacji zapytań

W celu przeprowadzenia kompleksowych badań nad problemem wyznaczania kolejności złączeń zapytań bazodanowych stworzono kilka programów komputerowych.

W związku z brakiem wymaganej ilości danych testowych napisano aplikację generującą bazy danych. Program umożliwia tworzenie baz o zadanej liczbie tabel i określonym grafie złączeń (gwiazda, łańcuch, nieregularny). Następnie generowane są dane o założonej z pewnym prawdopodobieństwem liczbie wierszy.

Kolejna aplikacja zajmuje się zbudowaniem opracowanej wcześniej bazy danych w systemie Microsoft SQL Server 2008. Prócz kreowania tabel, tworzone są klucze i indeksy, a następnie ładowane są przygotowane dane wypełniające wszystkie tabele.

Najważniejszym i najbardziej złożonym programem komputerowym jest optymalizator wyznaczający kolejność wykonywania złączeń podczas realizacji zapytań bazodanowych (rys. 42).



Rys. 42. Okno główne aplikacji wyznaczającej kolejność złączeń w zapytaniach bazodanowych.

Jest to aplikacja okienkowa, niepowiązana z żadnym systemem zarządzania bazami danych. Umożliwia ona określenie warunków wstępnych: danych dotyczących tabel; węzłów, w których tabele się znajdują; przepustowości łączy między poszczególnymi węzłami oraz warunków określających złączenia tabel. Optymalizator pozwala na ustawienie parametrów

algorytmu exIWO oraz na wielokrotne uruchamianie obliczeń numerycznych. Na uwagę zasługuje fakt, że program umożliwia wyznaczanie kolejności złączeń dla baz rozproszonych. Jednak nie przeprowadzono jeszcze dostatecznej liczby badań potwierdzających skuteczność algorytmu dla takich danych.