Uniwersytet Śląski w Katowicach
Wydział Informatyki i Nauki o Materiałach
Instytut Informatyki

Adaptacyjny algorytm optymalizacji stadnej cząsteczek dla dynamicznego problemu komiwojażera

mgr Łukasz Strąk

Rozprawa doktorska napisana pod kierunkiem
dr hab. Urszuli Boryczki

Promotor pomocniczy
dr Rafał Skinderowicz

Sosnowiec 2017

Chciałbym serdecznie podziękować mojemu promotorowi dr hab. Urszuli Boryczce, promotorowi pomocniczemu dr Rafałowi Skinderowiczowi, kolegom z zakładu Algorytmiki i Inteligencji Obliczeniowej dr Wojtkowi Wieczorkowi i mgr. Arkadiuszowi Nowakowskiemu za bardzo cenne uwagi i pomoc. Chciałbym również podziękować mojej żonie i rodzicom za wsparcie.

Spis treści

Wykaz terminów i skrótów

Cząsteczka Osobnik w algorytmie PSO i w jego wariantach. Reprezentuje krotkę $\langle X, V, pBest_i \rangle$, gdzie X to pozycja cząsteczki (jedno z rozwiązań problemu), V to prędkość (kierunek przeszukiwania przestrzeni rozwiązań) oraz $pBest_i$ najlepsze (na podstawie funkcji oceny) znalezione rozwiązanie przez cząsteczkę.

Feromon W środowisku naturalnym, substancja chemiczna pełniąca rolę m.in. kanału komunikacyjnego mrówek. W algorytmach uczenia maszynowego wartość zmiennoprzecinkowa, która zostaje powiększona (wzmocniona) lub zmniejszona (parowanie) wraz z kolejnymi iteracjami algorytmu.

α-miara Miara sąsiedztwa zaproponowana przez Kelda Helsgauna na potrzeby algorytmu LKH (ang. Lin-Kernighan Heuristic). Więcej informacji można znaleźć w rozdziale 4 na stronie 73.

ACO (skrót od ang. *Ant Colony Optimization*), metaheurystyka inspirowana wyszukiwaniem najkrótszej trasy do pożywienia przez mrówki w środowisku naturalnym. Patrz strony: v–vii, xix, 10–12, 33, 36, 46, 107, 109, 110.

ACS (skrót od ang. *Ant Colony System*), jedna z wersji ACO, algorytm zaproponowany przez Marco Dorigo i Luca Maria Gambardellę w 1997 roku [1]. Patrz strony: 33, 97, 98.

APX (skrót od ang. *Approximable*), zbiór algorytmów aproksymacyjnych dla problemów optymalizacyjnych o złożoności NP, których czas działania jest wielomianowy w stosunku do rozmiaru problemu, a wynik algorytmu jest oszacowany z góry (nie gorszy niż założona wartość) postaci $c \cdot x^*, c > 0$, gdzie c to pewna stała. Patrz strony: viii, 6, 7.

AS (skrót od ang. *Ant System*), jedna z wersji ACO, algorytm mrówkowy stworzony przez Marco Dorigo w 1992 roku [2, 3]. Przegląd implementacji można znaleźć w [4, 5, 6]. Patrz strony: 33, 34.

BG (skrót od ang. *Benchmark Generator*), generator instancji problemów optymalizacyjnych. Patrz strony: xvii, xix.

BPSO (skrót od ang. *Binary PSO*), binarny algorytm optymalizacji stadnej cząsteczek. Więcej informacji można znaleźć w rozdziale 2 na stronie 19. Patrz strony: 26, 28.

DPSO (skrót od ang. *Discrete PSO*), dyskretny algorytm optymalizacji stadnej cząsteczek. Więcej informacji można znaleźć w rozdziale 3 na stronie 39. Patrz strony: xii–xiv, xvii–xix, 10, 19, 26, 27, 29, 30, 32, 40, 42–55, 58, 61–70, 73, 85–89, 91, 93–99, 102, 103, 105–111, 113–116.

DTSP (skrót od ang. *Dynamic TSP*), dynamiczny problem komiwojażera. Więcej informacji można znaleźć w rozdziale 1 na stronie 1. Patrz strony: x, xi, xiii–xix, 1, 8–12, 14, 15, 17, 19, 39, 48, 60, 63, 68, 71, 72, 74, 94, 99, 105, 106, 108–111, 113, 115.

FPTAS (skrót od ang. *Fully Polynomial Time Approximation Scheme*), w pełni wielomianowy schemat aproksymacji jest podzbiorem problemów należących do PTAS, jednak zależność między rozmiarem problemu n i $\frac{1}{\epsilon}$ w złożoności czasowej musi być wielomianowa (w PTAS zależność ta mogła być wykładnicza). Patrz strony: 6, 7.

HKLO (skrót od ang. *Held-Karp Lower Bound*), algorytm szacowania długości trasy optymalnej problemu TSP. Więcej informacji można znaleźć w rozdziale 4 na stronie 73. Patrz strony: 75.

ILP (skrót od ang. *Integer Linear Programming*), programowanie liniowe całkowitoliczbowe. Jedna z technik optymalizacyjnych wchodząca w skład metod programowania matematycznego. Głównym założeniem jest przekształcenie problemu do postaci modelu optymalizacyjnego, w skład którego wchodzą warunki ograniczające przestrzeń rozwiązań oraz cel optymalizacji (minimalizacja lub maksymalizacja). Patrz strony: 3.

LKH (skrót od ang. *Lin-Kernighan Heuristic*), algorytm przeszukiwania przestrzeni rozwiązań dla problemu TSP, stworzony przez Kelda Helsgauna [7, 8]. Więcej informacji można znaleźć w rozdziale 4 na stronie 73. Patrz strony: v, 73.

MMAS (skrót od ang. $\mathcal{MAX} - \mathcal{MIN}$ *Ant System*), algorytm mrówkowy $\mathcal{MAX}-\mathcal{MIN}$ zaproponowany przez Thomasa Stützle i Holgera Hoosa [9] w 2000 roku. Przegląd najnowszych implementacji można znaleźć w [10]. Patrz strony: 36, 37, 46, 47.

MST (skrót od ang. *Minimal Spanning Tree*), przypadek drzewa, które zostało otrzymane z algorytmu wyznaczającego *minimalne drzewo rozpinające* - zbiór krawędzi, które zawierają każdy wierzchołek grafu i mają minimalną wagę. Patrz strony: 77, 78, 80.

NP W teorii złożoności obliczeniowej, klasa problemów decyzyjnych, w których znalezienie optimum wymaga niewielomianowego czasu obliczeń w stosunku do rozmiaru problemu n. Dodatkowo sprawdzenie poprawności rozwiązania musi być możliwe w czasie wielomianowym. Patrz strony: v, xv, 2, 6, 7, 36, 74, 75.

P W teorii złożoności obliczeniowej, klasa problemów, których optimum można znaleźć w czasie wielomianowym. Złożoność czasowa algorytmów dla każdego z problemów jest postaci $\mathrm{O}(n^p), p \in \mathbb{N}$. Patrz strony: 7.

PACO (skrót od ang. *Population ACO*), populacyjny algorytm mrowiskowy, modyfikacja algorytmu ACO. Polega na wprowadzeniu listy rozwiązań, do której trafiają najlepsze rozwiązania z każdej iteracji (generacje) [11, 12]. W stosunku do klasycznego ACO zmieniono również podejście do uaktualnienia macierzy feromonowej. Patrz strony: xix, 97, 98, 107, 109.

PSO (skrót od ang. *Particle Swarm Optimization*), algorytm optymalizacji stadnej cząsteczek. Więcej informacji można znaleźć w rozdziale 2 na stronie 19. Patrz strony: v, vi, xii, xvi–xix, 19–26, 28, 39, 85, 86, 105, 106, 111, 115.

PTAS (skrót od ang. *Polynomial Time Approximation Scheme*), wielomianowy schemat aproksymacji jest podzbiorem problemów należących do APX, a których wartość oszacowania otrzymanego rozwiązania zależy od parametru ϵ. Zależność opisuje wzór: $(1-\epsilon)x^*$ dla maksymalizacji, $(1+\epsilon)x^*$ dla minimalizacji, $\epsilon > 1$. Z kolei złożoność czasowa musi być wielomianowa, np. $O(n^{1/\epsilon})$ lub $O(2^{1/\epsilon}n^2)$. Patrz strony: vi, 6, 7.

TSP (skrót od ang. *Traveling Salesman Problem*), problem komiwojażera. Więcej informacji można znaleźć w rozdziale 1 na stronie 1. Patrz strony: vi–viii, xiv–xvii, xix, 1–10, 12, 21, 27, 29, 34, 40–44, 47, 69, 73–76, 78, 79, 83, 85, 87, 92–94, 100, 102, 105, 106, 108, 109, 114, 115.

TSPLIB Najpopularniejsza biblioteka instancji problemów TSP wraz z dokumentacją dotyczącą struktury instancji. Dane problemów znajdują się pod adresem:
`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`. Patrz strony: xvii, 13–15, 55, 69, 75, 81, 103, 109, 110, 114.

Wykaz symboli

$\mathcal{G}$ Skierowany graf ważony składający się z następującej trójki: $\langle \mathcal{V}, \mathcal{E}, w \rangle$. Patrz strony: ix, xii, xiii, 2, 4, 27.

$\mathcal{V}$ Zbiór wierzchołków skierowanego grafu ważonego $\mathcal{G}$. Patrz strony: ix, 2, 3, 5, 8, 27, 45, 49, 77.

$\mathcal{E}$ Zbiór krawędzi skierowanego grafu ważonego $\mathcal{G}$. Patrz strony: ix, 2, 3, 27, 41.

v Jeden z wierzchołków należący do zbioru wierzchołków $v \in \mathcal{V}$ skierowanego grafu ważonego $\mathcal{G}$. Patrz strony: ix, 3, 27, 35, 41, 76–79.

e Krawędź należąca do zbioru krawędzi $\mathcal{E}$ skierowanego grafu ważonego $\mathcal{G}$. Patrz strony: xiii, 3, 4, 27, 74.

$\langle a, b \rangle$ Krawędź o końcach a, b należąca do zbioru krawędzi $\langle a, b \rangle \in \mathcal{E} \quad a, b \in \mathcal{V}$ skierowanego grafu ważonego $\mathcal{G}$, w którym kolejność wierzchołków w krawędzi ma znaczenie. Patrz strony: ix, xi, xiii, 3, 4, 27–30, 35, 36, 40, 41, 45, 46, 77, 78, 80, 92, 99–101, 114.

w Funkcja przyporządkowująca wagę do krawędzi skierowanego grafu ważonego $\mathcal{G}$. Patrz strony: ix, xi, 2, 3, 77, 78, 80.

$\mathcal{G}_u$ Nieskierowany graf ważony składający się z następującej trójki: $\langle \mathcal{V}_u, \mathcal{E}_u, w_u \rangle$. Patrz strony: ix, x, xiii, 4, 31.

$\mathcal{V}_u$ Zbiór wierzchołków nieskierowanego grafu ważonego $\mathcal{G}_u$. Patrz strony: ix, x, 4, 31.

$\mathcal{E}_u$ Zbiór krawędzi nieskierowanego grafu ważonego $\mathcal{G}_u$. Patrz strony: ix, x, 4, 31.

v_u Jeden z wierzchołków należący do zbioru wierzchołków $v_u \in \mathcal{V}_u$ nieskierowanego grafu ważonego $\mathcal{G}_u$. Patrz strony: x, 4.

e_u Krawędź należąca do zbioru krawędzi $\mathcal{E}_u$ nieskierowanego grafu ważonego $\mathcal{G}_u$. Patrz strony: xiii.

$\{a, b\}$ Krawędź o końcach a, b należąca do zbioru krawędzi $\{a, b\} \in \mathcal{E}_u \quad a, b \in \mathcal{V}_u$ nieskierowanego grafu ważonego $\mathcal{G}_u$, w którym kolejność wierzchołków w krawędzi nie ma znaczenia. Patrz strony: x, xiii, 4, 5, 29, 31, 32, 42, 89.

w_u Funkcja nadająca wagę krawędzi nieskierowanego grafu ważonego $\mathcal{G}_u$. Patrz strony: ix.

$\deg(v)$ Stopień wierzchołka - liczba krawędzi incydentnych do danego wierzchołka lub inaczej liczba połączeń danego wierzchołka z samym sobą lub innymi wierzchołkami. Patrz strony: 4, 31, 45, 76, 79.

O Notacja dużego O. Określa asymptotyczne tempo wzrostu średniego czasu obliczeń w stosunku do rozmiaru danych wejściowych (rozmiaru problemu). Patrz strony: vii, viii, 6, 7, 99, 100.

$\mathcal{I}$ Instancja problemu komiwojażera wraz z informacją o wierzchołkach, krawędziach oraz macierzą odległości. Patrz strony: x, 8.

$\mathcal{I}^t$ Instancja problemu komiwojażera $\mathcal{I}$ o indeksie t (podproblem w DTSP). Patrz strony: x, xi, 12, 14, 15, 47.

$\mathcal{S}$ Przestrzeń rozwiązań problemu DTSP. Patrz strony: 8.

$\mathcal{S}^t$ Przestrzeń rozwiązań problemu komiwojażera. W kontekście DTSP przestrzeń rozwiązań podproblemu o indeksie t (instancja $\mathcal{I}^t$). Rozmiar przestrzeni rozwiązań podproblemu w przypadku ogólnym wynosi $n!$. Patrz strony: x, 8.

s Jedno z rozwiązań problemu DTSP postaci $s = \langle s^0, s^1, ..., s^{t_{max}} \rangle$. Jest to wektor, którego składowa s^t jest jednym z rozwiązań problemu komiwojażera w czasie t (o indeksie t). Patrz strony: 8.

s^t Jedno z rozwiązań podproblemu DTSP (rozwiązanie problemu komiwojażera) dla instancji $\mathcal{I}^t$, $s^t \in \mathcal{S}^t$. Patrz strony: x, 8.

x^* Optimum problemu. Patrz strony: v, viii, 6, 7, 74, 75.

x^+ Zbiór rozwiązań nie należących do optimum. Patrz strony: 74.

D Macierz odległości między wierzchołkami w grafie ważonym o rozmiarze $n \times n$. Wartość na przecięciu wiersza a oraz kolumny b ($a \neq b$) oznacza wagę krawędzi, $w(a, b)$. Patrz strony: 3, 4, 8, 29, 74.

d_{ab} Odległość między wierzchołkami a, b. Zapis równoznaczny z $w(a, b)$ lub w przypadku, gdy krawędzie obliczono na podstawie metryki równoznaczny z $d(a, b)$. Patrz strony: 3, 4, 35, 36, 92.

$D(t)$ Macierz odległości dla instancji problemu $\mathcal{I}^t$ w czasie t. Patrz strony: xi, 8.

$d_{ab}(t)$ Odległość między wierzchołkami a, b w czasie t. Patrz strony: 8.

$n(t)$ Funkcja zwracająca liczbę wierzchołków w czasie t wykorzystywana w macierzy odległości $D(t)$. Patrz strony: 8.

p_m Liczba zmian w podproblemie problemu DTSP (problem komiwojażera). Patrz strony: 14.

f_{inf} Wpływ losowych wartości na przesunięcie współrzędnych miasta. Wartość z przedziału $[0, 1]$. Patrz strony: 14.

$\tau_{\max}$ Maksymalna wartość feromonu w macierzy feromonowej τ. Patrz strony: 36, 37, 41, 47–49, 53, 55.

$\tau_{\min}$ Minimalna wartość feromonu w macierzy feromonowej τ. Patrz strony: 36, 37, 41, 47, 49, 55.

$\Delta\tau$ Funkcja obliczająca wartość wzmocnienia prawdopodobieństwa wyboru krawędzi do następnej pozycji p, m.in. na podstawie macierzy feromonowej τ. Patrz strony: 40.

$\tau_{a,b}$ Element macierzy feromonowej ($\tau_{a,b} \in \mathbb{R}$), oznaczający ilość feromonu dla krawędzi $\langle a, b\rangle$. Patrz strony: xi, 35, 36, 41, 42, 48.

τ Macierz feromonowa, której każdy element to wielkość feromonu na krawędzi $\langle a, b\rangle$. Macierz ta ma wymiar $n \times n$. Patrz strony: xi, 41.

$\vec{v}_i^k$ Prędkość cząsteczki i w iteracji k w algorytmie PSO. Formalnie jest to d-wymiarowy wektor. Patrz strony: xii, 22–26.

$\vec{v}_i^{k+1}$ Kolejna prędkość cząsteczki i w algorytmie PSO. Patrz strony: 22–26.

$\vec{x}_i^{k+1}$ Pozycja cząsteczki i w kolejnej iteracji algorytmu PSO. Patrz strony: xiv, 22–26.

$\vec{x}_i^k$ Pozycja cząsteczki i w iteracji k. Formalnie w algorytmie PSO to wektor o wymiarze równym wymiarowi przestrzeni rozwiązań, będący jednocześnie jednym z rozwiązań problemu. Patrz strony: 22–26.

V_i^k Prędkość cząsteczki i w iteracji k w algorytmie DPSO. Formalnie zbiór krawędzi należących do grafu $\mathcal{G}$ problemu. Patrz strony: 27, 28, 31, 40, 42, 44.

V_i^{k+1} Kolejna prędkość cząsteczki i w algorytmie DPSO. Patrz strony: xiii, 28, 30–32, 40, 42–44, 46.

X_i^k Pozycja cząsteczki i w iteracji k w algorytmie DPSO. Formalnie zbiór krawędzi, który tworzy cykl Hamiltona. Patrz strony: 27, 28, 30, 31, 40, 42, 44–46, 51–53, 88–92, 99, 115, 116.

X_i^{k+1} Pozycja cząsteczki i w iteracji kolejnej iteracji algorytmu DPSO. Patrz strony: xiii, 28, 30–32, 40, 43, 46.

$g\vec{B}est$ Najlepsza pozycja stada (najlepsze znalezione rozwiązanie). Formalnie w algorytmie PSO jest to wektor wielowymiarowy. Patrz strony: xiv, 22–25.

$gBest$ Najlepsza pozycja stada (najlepsze znalezione rozwiązanie). Formalnie w algorytmie DPSO jest to zbiór krawędzi, który tworzy cykl Hamiltona. Patrz strony: 22, 24, 28–31, 40, 42–46, 51, 52, 56–60, 86, 87, 89–92, 94, 95, 99, 107, 114–116.

$p\vec{B}est_i$ Najlepsza pozycja cząsteczki i (najlepsze znalezione rozwiązanie przez cząsteczkę i). Formalnie w algorytmie PSO jest to wektor wielowymiarowy. Patrz strony: xiv, 22–25.

V_{max} Parametr ograniczający wartość prędkości cząsteczki w algorytmie PSO. Patrz strony: 23.

ω Parametr bezwładności cząsteczki, który skaluje poprzednią prędkość cząsteczki. Patrz strony: 25, 28, 31, 39, 40, 42, 49, 61, 86–88, 90–97.

$pBest_i$ Najlepsza pozycja cząsteczki i (najlepsze znalezione rozwiązanie przez cząsteczkę i). Formalnie w algorytmie DPSO jest to zbiór krawędzi, który tworzy cykl Hamiltona. Patrz strony: v, 22, 28, 30, 31, 40, 42–46, 87, 89–92, 94, 116.

$\hat{e}$ Krawędź w algorytmie DPSO dla skierowanego grafu ważonego $\mathcal{G}$. Zawiera dodatkową zmienną p, która oznacza prawdopodobieństwo wyboru danej krawędzi do kolejnej pozycji cząsteczki V_i^{k+1}. Formalnie to uporządkowana dwójka postaci: $\langle p, e\rangle$ ($\langle p, \langle a, b\rangle\rangle$). Patrz strony: xiii, 27–32, 40–43, 45, 46, 89.

$\hat{e}_u$ Krawędź nieskierowanego grafu ważonego $\mathcal{G}_u$ w algorytmie DPSO. Zawiera dodatkową zmienną p, która oznacza prawdopodobieństwo wyboru danej krawędzi do kolejnej pozycji cząsteczki V_i^{k+1}. Formalnie to uporządkowana dwójka postaci: $\{p, e_u\}$ ($\langle p, \{a, b\}\rangle$). Patrz strony: 29.

p Składowa krawędzi. Prawdopodobieństwo wyboru krawędzi do następnej pozycji X_i^{k+1} w algorytmie DPSO. Patrz strony: xi, xiii, 27–32, 40–43, 45–47, 87–89, 93.

$\mathcal{N}_a$ Sąsiedztwo wierzchołka, zbiór najbliższych sąsiadów ustalonych na podstawie pewnej miary, np. euklidesowej. Patrz strony: 35, 45.

$\mathcal{N}_{size}$ Parametr oznaczający maksymalny rozmiar sąsiedztwa każdego wierzchołka. Patrz strony: 49, 60, 61, 88, 92, 94, 97, 102, 107.

p_{mu} Mnożnik liczby ewaluacji, którą wyznacza się zgodnie ze wzorem: $p_{mu} \cdot n$. Patrz strony: 67, 68, 71.

π Funkcja kary zwiększająca wagę wszystkich przystających krawędzi do wierzchołka. Patrz strony: 78–80.

t Dyskretna zmienna oznaczająca czas lub inaczej numer podproblemu. Patrz strony: x, xi, xiii, xv, 8, 11, 14, 48, 74.

t_{max} Całkowita liczba podproblemów dla pewnej instancji problemu DTSP. Jest to również maksymalny indeks t. W pracy parametr ten ustalono a priori na wartość 10 (oryginalny problem z TSPLIB oraz 10 zmodyfikowanych podproblemów). Patrz strony: 8, 14.

i Numer cząsteczki w stadzie. Patrz strony: xii–xiv, 22, 26–28, 30, 40, 42, 44–46, 115.

i_{max} Rozmiar stada (populacji). Maksymalny indeks i. Patrz strony: 24, 30, 45, 46, 49, 53, 61, 67, 88, 92, 94, 97, 99, 102, 107.

k Numer iteracji algorytmu. Patrz strony: xii, 22, 24, 26–28, 30, 40–43, 46, 47, 78–80.

k_{max} Liczba iteracji algorytmu. Patrz strony: 24, 30, 41, 42, 46, 61, 67, 69, 80, 99, 102, 107, 116.

n Rozmiar problemu. W TSP i DTSP liczność zbioru wierzchołków. Patrz strony: vi–viii, x, xi, xv, 3–8, 34, 41, 48, 53, 61, 67, 68, 74, 78–80, 99, 102, 107.

d Wymiar przestrzeni. Patrz strony: 7, 21, 22, 24, 26.

c_1 Parametr kognitywny, skaluje wpływ najlepszego znalezionego rozwiązania przez cząsteczkę i, $p\vec{B}est_i$, na kolejną pozycję cząsteczki $\vec{x}_i^{k+1}$. Patrz strony: 22–25, 28, 29, 31, 39, 40, 42, 48, 49, 61, 87–97.

c_2 Parametr społeczny, skaluje wpływ najlepszego znalezionego rozwiązania przez stado $g\vec{B}est$ na kolejną pozycję cząsteczki $\vec{x}_i^{k+1}$. Patrz strony: 22–25, 28, 29, 31, 39, 40, 42, 48, 49, 61, 87–97.

c_3 Parametr wprowadzony dla algorytmu DPSO [13], skaluje prawdopodobieństwo wyboru krawędzi znajdujących się w aktualnym zbiorze pozycji do kolejnej pozycji cząsteczki. Patrz strony: 28–30, 39, 40, 46, 48, 49, 61, 87, 88, 90–97.

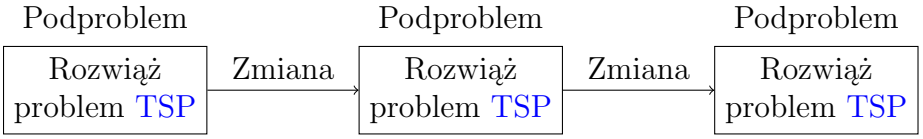
$rand()$ Funkcja losująca wartość na podstawie rozkładu jednostajnego. Domyślnie z przedziału $[0, 1]$, chyba, że przedział został określony wprost np. $rand(0, 2)$. Patrz strony: xiv, 14, 22, 24–26, 28–31, 40, 42, 46, 87, 91.

Wstęp

Problem komiwojażera (TSP) został zdefiniowany w 1930 roku i jest jednym z najlepiej zbadanych problemów optymalizacji dyskretnej. Pod względem klasy złożoności obliczeniowej jest problemem $\mathcal{NP}$-*trudnym*. Dla wersji decyzyjnej, sprawdzającej istnienie krótszej trasy niż pewna założona długość, jest problemem $\mathcal{NP}$-*zupełnym*. W wersji klasycznej polega on na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Nazwa pochodzi od następującej definicji problemu: dla n miast oraz wyznaczonych wszystkich odległościach pomiędzy nimi, należy znaleźć najkrótszą drogę przechodzącą przez wszystkie miasta dokładnie raz. W przypadku ogólnym rozmiar przestrzeni rozwiązań wynosi $n!$. Najczęściej miasta – wierzchołki znajdują się na płaszczyźnie euklidesowej, a odległości miedzy nimi – wagi krawędzi, wyznacza się na podstawie metryki euklidesowej. Problem ma zastosowanie w takich dziedzinach jak: planowanie, logistyka, tworzenie układów scalonych oraz sekwencjonowanie DNA. Mimo dużej popularności nie istnieje algorytm dokładny rozwiązujący ten problem w czasie wielomianowym w ogólnym przypadku [14].

W dynamicznej wersji problemu (DTSP) wprowadzono dodatkowy, dyskretny parametr t oznaczający czas. W kolejnych iteracjach t mogą zmieniać się odległości między miastami i/lub liczba miast. W pracy rozpatrywana jest tylko zmiana odległości między miastami. Dynamiczną wersję problemu należy rozpatrywać jako sekwencję następujących po sobie zmian w pierwotnych danych wejściowych lub jako sekwencję problemów komiwojażera nazywanych podproblemami. Po każdej zmianie uruchamiany jest ponownie algorytm rozwiązujący problem. Pod względem złożoności obliczeniowej dynamiczny problem komiwojażera również jest $\mathcal{NP}$-*trudny* (należy rozwiązać wiele razy problem komiwojażera). Rysunek 1 przedstawia przykładowy problem DTSP. Zmiana w podproblemie może dotyczyć zarówno odległości między miastami jak i zmiany liczby wierzchołków. Diagram prezentuje dwie

takie zmiany.



Rysunek 1: Schemat rozwiązywania problemu DTSP składający się z trzech podproblemów TSP.

Konkurencja oraz ewolucja sprawiają, że rozwiązania stosowane przez naturę są efektywne. Organizmy żywe charakteryzują się homeostazą – zdolnością do opierania się zmianom środowiskowym i trwania w stanie równowagi biologicznej. Jest to proces adaptacji lub kooptacji stanowiący proces przystosowania się do zmieniających się warunków panujących w przyrodzie. Determinuje on zachowanie, a w przypadku zwierząt stadnych zachowanie i organizację pracy w kolektywie. Badania nad tymi zjawiskami doprowadziły m.in. do powstania terminu synergia, który jest próbą sformułowania mechanizmu kooperacji osobników w ramach pewnej gromady osobników (np. stada) [15]. Termin ten został wprowadzony przez francuskiego entomologa Pierre-Paul Grassé w latach czterdziestych i pięćdziesiątych XX wieku na podstawie badań nad sposobem komunikacji termitów [16, 17, 15, 18, 19]. Komunikacja ta jest niebezpośrednia, a pośrednikiem jest modyfikowane środowisko. Dodatkowo informacja jest lokalna i może zostać odczytana jedynie przez osobniki znajdujące się w najbliższym otoczeniu. Nie występuje w niej porozumiewanie się przy pomocy dźwięków. Stanowi jednak filar samoorganizacji kolektywnych grup insektów, np. mrówek. Przykładowo, pojedyncza mrówka ma bardzo ograniczone możliwości w poszukiwaniu pożywienia wynikające z jej anatomii. Dzięki kooperacji (synergii), mrówki są w stanie podołać skompilowanym zadaniom, takim jak budowa gniazda czy wychowanie potomstwa. W przypadku tych owadów synergia objawia się poprzez feromon, który odkłada każda z nich na swojej drodze. Inteligencja obliczeniowa jest dziedziną wchodzącą w skład sztucznej inteligencji, która polega na zastosowaniu zachowań i strategii organizmów żywych w środowisku naturalnym do rozwiązywania problemów algorytmicznych [20]. Algorytmy inspirowane zachowaniem mrówek są przykładową formą naśladownictwa natury [21, 16]. Innym przykładem algorytmu inspirowanego naturą jest optymalizacja stadna cząsteczek (PSO). Technika ta jest inspirowana ruchem stada ptaków pod-

czas lotu. W algorytmie, osobnikiem jest wirtualna cząsteczka, a populacja to zbiór wirtualnych cząsteczek tworzących stado. Dla optymalizacji funkcji w dziedzinie liczb rzeczywistych, każda pozycja cząsteczki to jedno z rozwiązań problemu (wektor liczb rzeczywistych). Poruszając się po wielowymiarowej przestrzeni rozwiązań, cząsteczka eksploruje kolejne rozwiązania próbując znaleźć rozwiązanie optymalne lub rozwiązanie znajdujące się w sąsiedztwie optimum. Jakość pozycji cząsteczki wyznacza funkcja oceny (ewaluacji). Algorytm PSO definiuje zależności między cząsteczkami i wyznacza sposób tworzenia kolejnych rozwiązań (kolejnych pozycji cząsteczek). Algorytm powstał na potrzeby optymalizacji w wielowymiarowej przestrzeni ciągłej ze względu na bezpośrednie odwzorowanie koncepcji ruchu stada. Wkrótce jednak zaproponowano różne wersje dla problemów dyskretnych. Przedstawiony w pracy dyskretny algorytm optymalizacji stadnej cząsteczek – DPSO (ang. *Discrete Particle Swarm Optimization*), wzbogacony o wirtualny feromon, posiada cechy samoadaptacji i samoorganizacji, które umożliwiają mu przystosowanie się do problemu w środowisku dynamicznych zmian. Przy każdej modyfikacji odległości między miastami ulegają częściowej zmianie. Proponowane rozwiązanie wykorzystuje informację o rozwiązaniach sprzed zmian w celu minimalizacji otrzymanego rozwiązania w kolejnym problemie TSP (kolejnym podproblemie DTSP). Dzięki temu algorytm efektywniej przeszukuje przestrzeń rozwiązań. W pracy zastosowano również sąsiedztwo, którego odległości wierzchołków obliczane są na podstawie α-miary (ang. α-*measure*) oraz zaproponowano repozytorium do testowania jakości otrzymywanych wyników (BG). To ostatnie opracowanie powstało w wyniku braku zunifikowanego oraz spójnego podejścia do testowania algorytmów rozwiązujących dynamiczny problem komiwojażera, tak jak ma to miejsce w statycznej wersji problemu (repozytorium TSPLIB).

Teza

Zastosowanie pamięci feromonowej oraz heterogeniczności wartości parametrów w dyskretnym algorytmie PSO dla dynamicznego problemu komiwojażera pozwala poprawić jakość otrzymywanych wyników.

Motywacja

Jednym ze sposobów rozwiązywania problemu (DTSP) może być rozpa-

trywanie każdego podproblemu niezależnie. W takim przypadku istnieje możliwość zastosowania całej gamy algorytmów stosowanych do rozwiązywania problemu komiwojażera. Nie dostarcza to jednak żadnej nowej wiedzy na temat charakterystyki problemu DTSP – m.in. wpływu zmian na optimum problemu. Celem pracy jest opracowanie algorytmu stricte dla problemu DTSP i dla którego zbadana zostanie jakość adaptacji do przestrzeni rozwiązań kolejnego podproblemu. Ważnym problemem badawczym jest tutaj oszacowanie, dla jakiego poziomu zmian w kolejnych podproblemach wykorzystanie informacji o poprzednich rozwiązaniach jest efektywne, tzn. poprawia efektywność przeszukiwania przestrzeni rozwiązań, umożliwiając znalezienie optimum lub krótszego cyklu Hamiltona, niż miałoby to miejsce bez zastosowania tej informacji. Algorytmy inspirowane naturą charakteryzują się samoorganizacją. Dynamiczny problem komiwojażera jest w swej naturze zmienny. Użycie algorytmu DPSO, mającego cechy samoorganizacji, ma na celu wykorzystanie tych cech do efektywnego wyszukiwania rozwiązania optymalnego w tak zmieniającym się środowisku.

Cel pracy

Głównym założeniem pracy jest zastosowanie pamięci feromonowej w dyskretnej wersji algorytmu PSO w celu przystosowania algorytmu do dynamicznego problemu komiwojażera. Dodatkowo praca ma na celu zbadanie efektywności opracowanego algorytmu w stosunku do zmienności problemu (liczby zmian w kolejnym podproblemie). Kolejne etapy realizacji celu nadrzędnego obejmują:

- stworzenie podstawowej wersji algorytmu optymalizacji stadnej cząsteczek z feromonem oraz dostosowanie do omawianego problemu,
- utworzenie repozytorium problemów DTSP,
- analiza wpływu wartości parametrów opracowanego algorytmu na jakość uzyskanych wyników,
- opracowanie samoadaptacyjnej wersji algorytmu DPSO pod kątem szybszego i lepszego dopasowania się do problemu (minimalizacja odległości od optimum),

- oszacowanie progu użyteczności wiedzy o rozwiązaniu sprzed zmian do przyspieszenia zbieżności algorytmu DPSO po wprowadzeniu zmian w rozwiązywanym problemie DTSP,

- porównanie hybrydowego algorytmu DPSO z feromonem z klasycznymi algorytmami inteligencji obliczeniowej: ACO oraz PACO.

Praca składa się z 5 rozdziałów. W rozdziale 1 omówiony zostanie problem TSP oraz jego dynamiczna wersja (DTSP). W rozdziale zostanie opisany również generator testów (BG), który powstał na potrzeby zbadania efektywności zaproponowanego algorytmu. Przedstawiony zostanie również przegląd literatury związany z problemem DTSP. W rozdziale 2 przedstawione zostaną wybrane zagadnienia związane z inteligencją obliczeniową, które są niezbędne do zrozumienia działania algorytmu DPSO z feromonem. Opisane zostaną następujące pojęcia: inteligencja obliczeniowa, algorytmy mrowiskowe oraz różne warianty algorytmu PSO – od wersji klasycznej do wersji dyskretnych. Rozdział 3 zawiera opis oraz podstawowe badania nad algorytmem będącym przedmiotem dysertacji doktorskiej. W rozdziale tym opisane będą również badania związane z wpływem liczby zmian na efektywność proponowanego rozwiązania. W rozdziale 4 zostanie opisana funkcja sąsiedztwa stosowana w pracy. Stanowi ona część algorytmu DPSO z feromonem. W rozdziale 5 zastanie omówiony wariant heterogeniczny algorytmu DPSO z feromonem. Wariant ten zostanie porównany z algorytmami ACO oraz PACO. W celu weryfikacji porównania użyte zostaną testy statystyczne. W ostatnim rozdziale przedstawione zostanie podsumowanie oraz wnioski.

Rozdział 1

Dynamiczny problem komiwojażera

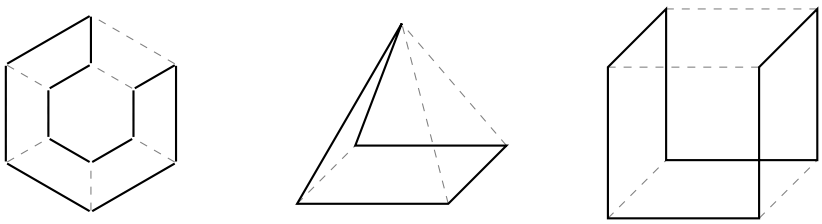
W rozdziale zostaną omówione podstawowe pojęcia związane z problemem komiwojażera oraz jego dynamiczną wersją. W podrozdziale 1.1 omówiona zostanie historia problemu TSP oraz jego definicja formalna. W dalszej części zostaną przedstawione różne warianty problemu. Następnie zostaną przedstawione algorytmy aproksymacyjne oraz dokładne rozwiązujące problem TSP i różne jego warianty. W podrozdziale 1.2 znajduje się przegląd literatury oraz zostaną opisane wybrane zastosowania problemu komiwojażera i jego dynamicznej wersji. Ze względu na ograniczony rozmiar pracy, skupiono się na publikacjach związanych z inteligencją obliczeniową. W podrozdziale 1.3 zostanie omówione środowisko testowe, które ma pomóc w rzetelnym porównaniu różnych rozwiązań dla problemu DTSP.

1.1 Definicja problemu

Problem TSP

Nie istnieją jednoznaczne przesłanki, kto pierwszy sformułował problem komiwojażera. Pierwszą wzmianką na temat praktycznego zastosowania problemu znaleziono w podręczniku dla obwoźnych sprzedawców z 1832 roku, w którym znalazły się na trasy komiwojażera przedstawione na mapie [22]. Matematyczny model TSP został zdefiniowany w XIX wieku niezależnie przez dwóch matematyków Wiliama Hamiltona (za sprawą gry Icosian) oraz Tho-

masa Kirkmana [22, 23]. Pierwotna definicja problemu dotyczyła teorii grafów i polegała na znalezieniu w grafie cyklu Hamiltona, który przechodzi przez wszystkie wierzchołki dokładnie raz (Rys. 1.1) [23].



(a) (b) (c)

Rysunek 1.1: Przykładowe cykle Hamiltona dla grafu o 12 wierzchołkach (a), ostrosłupie (b) i sześcianie (c).

Pierwszymi badaniami naukowymi były prace Karla Mengera oraz Hasslera Whitney'a w latach 30. ubiegłego wieku [22, 24]. Teraźniejsza nazwa problemu oraz powiązanie z praktycznym wykorzystaniem problemu, gdzie jako wierzchołki przyjmuje się miasta, a jako krawędzie drogi między nimi została zaproponowana właśnie na bazie tych prac. Więcej informacji na ten temat można znaleźć w [24]. W nowej definicji problem można sformułować następująco: komiwojażer ma przejść przez wszystkie miasta dokładnie raz i wrócić do miasta, z którego rozpoczął podróż. Dodatkowo w nowej formule trasa ma być najkrótsza (o najmniejszej sumie wag krawędzi tworzących cykl). Liczne zastosowania problemu w różnych dziedzinach wynikają z jego uogólnienia. Jako miarę odległości między miastami (wagami krawędzi) można przyjąć np. czas, koszt pieniężny lub dowolną metrykę – np. odległość euklidesową. Dla zastosowań w logistyce przydatny może być czas. Z kolei dla zastosowań w wytwarzaniu układów scalonych może to być odległość między kolejnymi elementami, które należy przymocować do budowanego układu scalonego. W 1972 roku Richard M. Karp udowodnił, że problem znalezienia minimalnego cyklu Hamiltona jest NP-*zupełny*, co oznacza, że problem TSP należy do tej samej klasy złożoności obliczeniowej [22, 25, 26].

Formalnie klasyczny problem komiwojażera można modelować za pomocą pełnego skierowanego grafu ważonego $\mathcal{G}$:

$$\mathcal{G} =< \mathcal{V}, \mathcal{E}, w >,$$

gdzie:

- $\mathcal{V}$ jest zbiorem wierzchołków,
- $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ to zbiór krawędzi,
- e oznacza dowolną krawędź taką, że $e \in \mathcal{E}$,
- $\langle a, b \rangle$ oznacza krawędź o końcach a, b, taką, że $a, b \in \mathcal{V}, \langle a, b \rangle \in \mathcal{E}$,
- $\forall_{a \in \mathcal{V}} \langle a, a \rangle \notin \mathcal{E}$ – brak pętli,
- $n = |\mathcal{V}|$ – liczność zbioru wierzchołków (rozmiar problemu),
- $n(n-1) = |\mathcal{E}|$ liczność zbioru krawędzi (pełny graf bez pętli),
- w jest funkcją wag: $\mathcal{E} \to \mathbb{R}, \quad \forall e \in \mathcal{E}$, $w(e)$ nazywamy wagą krawędzi. Grupuje się je w macierz odległości D postaci:

$$D = \begin{pmatrix} - & w(1,2) & \cdots & w(1,n) \\ w(2,1) & - & \cdots & w(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ w(n,1) & w(n,2) & \cdots & - \end{pmatrix}.$$

  Przyjmuje się, że $d_{ab} = w(a,b)$, co nawiązuje do elementów macierzy odległości D. W literaturze, w definicji grafu skierowanego zamiast terminów wierzchołek i krawędź używa się terminów węzeł i łuk [27, 28]. Na poziomie syntaktycznym rozróżnia to graf skierowany (z węzłami i łukami) od grafu nieskierowanego (z wierzchołkami i krawędziami). W pracy używanymi pojęciami są wierzchołek i krawędź, ale z kontekstu będzie jasno wynikać czy chodzi o graf skierowany, czy nieskierowany. Warto w tym miejscu nadmienić, że istnieją inne ujęcia problemu, np. na potrzeby techniki programowania liniowego całkowitoliczbowego (ILP). Niemniej jednak zastosowanie tak sformułowanego problemu na potrzeby innych algorytmów jest bardzo ograniczone.

W praktycznym ujęciu problemu komiwojażera, wierzchołki reprezentują miasta, a wagi – odległości między nimi. Często w literaturze używa się zamiennie tych pojęć, odwołując się do teorii grafów lub praktycznego wykorzystania problemu.

W literaturze często można spotkać dwa pojęcia związane z problemem TSP – ścieżka Hamiltona oraz cykl Hamiltona. Ścieżka Hamiltona to taki ciąg krawędzi $e \in \mathcal{E}$, w którym każdy wierzchołek $v \in \mathcal{V}$ został odwiedzony dokładnie jeden raz. W takim przypadku tylko dwa wierzchołki – wybrany

na początku e_b i na końcu e_e są stopnia pierwszego. Pozostałe wierzchołki są stopnia drugiego, co można zapisać formalnie: $\forall_{e \in s_h \setminus \{e_b, e_e\}} \deg(e) = 2$, $\deg(e_b) = 1$, $\deg(e_e) = 1$, gdzie: s_h to ścieżka Hamiltona. W ścieżce Hamiltona istotny jest wybór początkowego i końcowego wierzchołka. Cykl Hamiltona zawiera dodatkową krawędź (e_e, e_b), co oznacza, że wybór wierzchołka początkowego i końcowego nie ma znaczenia. Graf hamiltonowski to graf $\mathcal{G}$, który zawiera przynajmniej jedną ścieżkę Hamiltona.

Warianty TSP

W zależności od warunków nałożonych na wagi, wyróżnia się wiele wariantów problemu TSP. Najważniejsze z nich zostały opisane poniżej.

- W wersji asymetrycznej pary krawędzi $\langle a, b \rangle$ i $\langle b, a \rangle$ mogą mieć przypisane różne wagi. Ten rodzaj problemu jest szczególnie często wykorzystywany do modelowania zatoru na drodze (droga może być przejezdna tylko w jedną stronę lub droga może być jednokierunkowa). W praktyce, ostatni przypadek jest realizowany przez przypisanie wartości $+\infty$ dla nieistniejącego kierunku.

- Symetryczny TSP jest szczególnym przypadkiem wersji asymetrycznej. Krawędzie spełniają warunek symetryczności, tj. $d_{ab} = d_{ba}$. Istotny jest fakt, że w przypadku tego problemu rozważany graf ważony $\mathcal{G}_u$ jest grafem nieskierowanym. Krawędzie w takim grafie definiujemy jako: $\mathcal{E}_u \subseteq \{\{u, v\} : u, v \in \mathcal{V}_u\}$. Liczność zbioru krawędzi wynosi $|\mathcal{E}_u| = \binom{n}{2}$ (wszystkie kombinacje dwuelementowe zbioru $\mathcal{V}_u$). Macierz odległości D również jest symetryczna. Wprowadzenie warunku symetryczności redukuje rozmiar przestrzeni rozwiązań problemu do $\frac{(n-1)!}{2}$.

- Problem TSP nazywamy metrycznym, gdy wagi zostały otrzymane na podstawie przyjętej metryki, co oznacza, że wagi spełniają następujące kryteria ($a, b, c \in \mathcal{V}_u$):

  1. identyczność nierozróżnialnych: $d_{ab} = 0 \iff a = b$,
  2. symetria: $d_{ab} = d_{ba}$,
  3. nierówność trójkąta: $d_{ac} \leqslant d_{ab} + d_{bc}$.

  Wariant ten występuje również w literaturze pod nazwami: delta-TSP lub Δ-TSP.

- Problem euklidesowy jest specjalnym przypadkiem problemu metrycznego, w którym jako metryki użyto odległości euklidesowej. Każdemu wierzchołkowi odpowiadają przyporządkowane mu współrzędne $\mathbb{R}^d$ (najczęściej dwuwymiarowe).

- W uogólnionym problemie TSP (GTSP) zbiór wierzchołków $\mathcal{V}$ dzieli się na grupy. Problem polega na znalezieniu minimalnego cyklu Hamiltona przechodzącego przez wszystkie grupy. Wybór wierzchołka, który będzie reprezentował grupę jest problemem nietrywialnym. Klasyczny problem komiwojażera jest specjalnym przypadkiem GTSP, w którym każda grupa zawiera jeden wierzchołek.

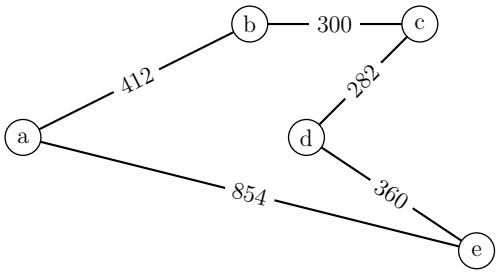
- W geometrycznym TSP, dwuwymiarowe współrzędne każdego wierzchołka losowane są z rozkładu jednostajnego z przedziału $[0, 1]$. Następnie wagi wyznaczone są na podstawie miary euklidesowej. Statystyk Prasanta Chandra Mahalanobis intuicyjnie oszacował, że rozmiar cyklu Hamiltona powinien rosnąć wraz z rosnącym rozmiarem problemu (liczbą wierzchołków) i przy odpowiednio dużej liczbie wierzchołków wynosić około $\sqrt{n}$ [29]. Eli Marks oszacował dolną granicę trasy optymalnej w instancjach geometrycznych na poziomie $\sqrt{n} - 1/\sqrt{n}/\sqrt{2}$ [30]. M.N. Ghosh oszacował górną granicę trasy optymalnej w instancjach geometrycznych na poziomie $1,27\sqrt{n}$ [29]. Oba artykuły potwierdziły przypuszczenie Mahalanobisa.

Należy zauważyć, że problem euklidesowy zawsze jest metryczny oraz symetryczny. Jednak problem metryczny oraz symetryczny nie musi być euklidesowy. W szczególności problem symetryczny nie musi być metryczny.

Rysunek 1.2 przedstawia przykładowy problem TSP, w którym każdemu wierzchołkowi odpowiadają współrzędne w przestrzeni euklidesowej. Zapis trasy w notacji wierzchołkowej $\langle a, b, c, d, e, a\rangle$, w notacji krawędziowej $\{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, a\}\}$.

Istnieje technika gwarantująca transformację asymetrycznego problemu komiwojażera do wersji symetrycznej. Wadą takiego podejścia jest jednak to, że liczba wierzchołków $\mathcal{V}$ musi zostać podwojona w stosunku do pierwotnego rozmiaru problemu n ($n \to 2n$) [31].

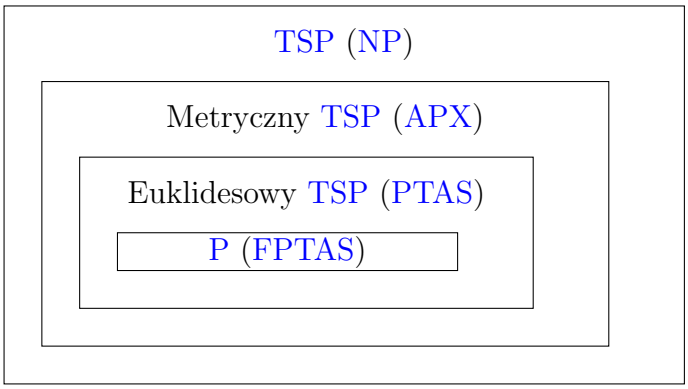
Algorytmy przybliżone dla TSP



Rysunek 1.2: Przykładowy cykl Hamiltona o długości 2208.

Algorytmy aproksymacyjne służą do znajdowania rozwiązań przybliżonych problemów optymalizacyjnych. Cechą charakterystyczną tych algorytmów w stosunku do heurystyk jest informacja o jakości przybliżenia, tzn. z góry wiadomo jakie będzie rozwiązanie w najgorszym przypadku. Dla problemu TSP klasa tych algorytmów jest licznie reprezentowana ze względu na trudność problemu. Klasa algorytmów aproksymacyjnych to zbiór algorytmów dla problemów z klasy NP, które działają w czasie wielomianowym w stosunku do rozmiaru problemu n, a których otrzymane rozwiązanie jest przybliżone (oszacowane w stosunku do optimum). W kontekście TSP takie oszacowanie z góry (problem minimalizacji) oznacza długość cyklu Hamiltona nie gorszą, niż założona wartość. Oszacowanie odróżnia algorytmy aproksymacyjne od heurystyk, w których takiego oszacowania nie ma. Przybliżenie może być wyrażone przez stałą lub dane wzorem. Algorytmy o stałej aproksymacji należą do klasy APX, a oszacowanie jest postaci: $c \cdot x^*$, gdzie $c \geqslant 1$ jest oszacowaniem wynikającym z algorytmu, x^* to optimum. Aproksymacja dana wzorem zależy od współczynnika aproksymacji $\epsilon, \epsilon > 0$. Złożoność czasowa takiego algorytmu jest wielomianowa w stosunku do rozmiaru problemu n, ale zależy od ϵ. Przykładem takiej złożoności czasowej może być $O(n^{1/\epsilon})$. Algorytm aproksymacyjny o takiej złożoności czasowej należy do klasy PTAS. W klasie PTAS zależność między rozmiarem problemu n a parametrem $1/\epsilon$ może być wykładnicza. W klasie FPTAS zależność obu zmiennych musi być wielomianowa. Rysunek 1.3 przedstawia warianty problemu TSP oraz najlepszą znaną klasę aproksymacji.

Nie jest znany żaden algorytm aproksymacyjny dla TSP o dowolnych wagach. Dla metrycznego TSP, algorytm Christofidesa pozwala otrzymać rozwiązanie co najmniej $\frac{3}{2}x^*$, gdzie x^* to rozwiązanie optymalne [32]. Złożoność czasowa takiego algorytmu to: $O(n^3)$, gdzie n oznacza rozmiar problemu. Nie istnieje dla tego wariantu algorytm o klasie PTAS (którego parametr



Rysunek 1.3: Warianty problemu TSP oraz odpowiadające im klasy aproksymacji.

aproksymacji ϵ można kontrolować). Dla wariantu euklidesowego najlepszy algorytm aproksymacyjny pozwala otrzymać trasę TSP o długości $(1+\epsilon)x^*$, gdzie $\epsilon > 0$ to współczynnik aproksymacji [33]. Złożoność czasowa tego algorytmu wynosi:

$$\mathrm{O}\left(n(\log n)^{(\mathrm{O}(\sqrt{d}/\epsilon))^{d-1}}\right),$$

gdzie d to liczba wymiarów współrzędnych określonych dla każdego wierzchołka. Niemniej jednak nie istnieje algorytm o klasie FPTAS. Za opracowanie tego algorytmu, dwóch uczonych: Sanjeev Arora oraz Joseph S. B. Mitchell zostali odznaczeni nagrodą Gödel Prize w 2010 [34].

Algorytmy dokładne dla TSP

Jest to klasa algorytmów najbardziej pożądana w praktycznych zastosowaniach ze względu na gwarancje znalezienia optimum. W tej klasie algorytmów najważniejsza jest złożoność obliczeniowa oraz pamięciowa. David Applegate, Robert E. Bixby, Vašek Chvátal, i William J. Cook są autorami najpopularniejszego pakietu *Concorde* [22]. W skład pakietu wchodzi wiele zaimplementowanych algorytmów dla problemu TSP, m. in. *Branch and Cut* oraz *Chained Lin-Kernighan* [35, 36]. Pakiet *Concorde* jest obecnie jednym z najszybszych i najbardziej zaawansowanych pakietów przeszukujących przestrzeń rozwiązań tego problemu. Pakiet ten w roku 2004 potrzebował ponad 80 CPU lat (rok na maszynie o mocy obliczeniowej 1GFLOPS), aby znaleźć rozwiązanie optymalne dla problemu *sw24978* [22, 37]. Ten sam pakiet uży-

ty około roku 2006 roku potrzebował 136 CPU lat do znalezienia optimum dla problemu *pla85900* [22, 38]. Największą wadą tego podejścia jest brak informacji na temat czasu obliczeń samego algorytmu.

Dynamiczna wersja TSP

Dynamiczny problem komiwojażera jest sekwencją (ciągiem) statycznych podproblemów $\mathcal{I}$, w których każda zmiana oznacza modyfikację odległości i/lub liczbę wierzchołków [39]. Pod względem formalnym można go zdefiniować następująco [40]:

$$D(t) = \{d_{ab}(t)\}_{n(t)\times n(t)}\,, \tag{1.1}$$

gdzie:

- D to macierz odległości (wag) między miastami (wierzchołkami),
- t to dyskretny parametr czasu lub numer podproblemu,
- a, b oznaczają końce krawędzi, takie, że $a, b \in \mathcal{V}$,
- $n(t)$ określa liczbę wierzchołków w funkcji czasu.

Zmiana dokonywana jest losowo oraz dotyczy tylko wybranego podzbioru wierzchołków. W pracy rozpatrywany jest tylko wariant ze zmianą macierzy odległości. Jednak informacje takie jak: czy krawędź należy do rozwiązania optymalnego są rozpatrywane z ograniczonym zaufaniem, gdyż każda modyfikacja może wpłynąć na zmianę optimum.
Przestrzeń rozwiązań DTSP $\mathcal{S}$ wynosi $\mathcal{S} = \mathcal{S}^0 \times \mathcal{S}^1 \times ... \times \mathcal{S}^{t_{max}}$, gdzie $\mathcal{S}^t$ oznacza zbiór wszystkich możliwych rozwiązań problemu komiwojażera w podproblemie o numerze t ($0 \leqslant t \leqslant t_{max}$). Jednym z rozwiązań problemu DTSP jest wektor $s \in \mathcal{S}$, którego każdy element $s^t \in \mathcal{S}^t$ jest jednym z rozwiązań problemu komiwojażera w czasie t (rozwiązaniem podproblemu o indeksie t). W przypadku braku ograniczeń nakładanych na wagi krawędzi i przy założeniu braku zmian w liczności zbioru wierzchołków $|\mathcal{V}|$ rozmiar przestrzeni rozwiązań problemu DTSP wynosi $n!^{t_{max}}$.

1.2 Przegląd literatury

Środowisko testowe dla problemu DTSP

W literaturze zmianę w problemie DTSP interpretuje się w rozmaity sposób, np. liczba zmian może być losowa lub zmiana danych może dotyczyć tylko tych krawędzi, które należą do rozwiązania optymalnego. W przypadku instancji problemów DTSP zaczerpniętych z rzeczywistych danych, najczęściej zmiana dotyczy współrzędnych wierzchołków. Tym samym wraz z ich zmianą, wszystkie wagi krawędzi incydentalne do zmienionych współrzędnych wierzchołków są przeliczane od nowa. Środowisko testowe zaproponowane przez Guntscha i innych [41], polega na symulowaniu dynamicznych zmian poprzez wymianę pewnej liczby miast między instancją problemu a dodatkową pulą miast. Powoduje to zmianę optimum, dlatego techniki optymalizacyjne są porównywane między sobą za pomocą różnicy długości najlepszych znalezionych rozwiązań. Metoda ta pozwala na porównanie wyników bez konieczności znajomości optimum [41]. Bardziej ogólną metodę zaproponował Younes i inni [42], która może zostać użyta jako środowisko testowe dla dynamicznego problemu optymalizacji kombinatorycznej. Pomysł polega na wykorzystaniu faktu, że wiele populacyjnych technik optymalizacyjnych używa jakiejś formy mapowania osobników na przestrzeń rozwiązań, np. permutację zbioru. Symulacja środowiska dynamicznego polega na zamianie parametrycznej liczby etykiet danego wierzchołka na inne w zakodowanym osobniku. W przypadku TSP zmiana miasta powoduje zmianę długości rozwiązania (odczytywane są inne długości w macierzy odległości). Zmianie ulegają długości rozwiązań, ale krajobraz przestrzeni rozwiązań oraz optimum pozostają takie same [42]. Podobny pomysł znajduje się w pracy Mavrovouniotis i innych [43], jednak nowe podejście jest bardziej wszechstronne. Zmiana kodowania dotyczy nie osobników, a instancji problemu. W przypadku problemu komiwojażera oraz marszrutowania, realizacja koncepcji polega na zamianie lokalizacji par miast. Podobnie jak w poprzednim podejściu, optimum nie zmienia się. Jednakże metoda ta nie odzwierciedla rzeczywistego problemu. Zupełnie inne podejście przedstawia Younes i inni [44]. Pomysł polega na dodawaniu losowych zmian do instancji podproblemu TSP i zapisanie ich na stosie. Następnie w połowie testu zmiany są kolejno wycofywane (cofając zmiany ze stosu). Pierwszy i ostatni podproblem jest jednakowy. Porównanie wyników technik optymalizacyjnych następuje w ostatniej iteracji, kiedy znane jest optimum. Innym podejściem jest połączenie statycznych współrzędnych miast oraz satelitów leżących na orbicie geostacjonarnej. Przykładem takiego problemu może być instancja *CHN146 + 3*, która zawiera 146 statycznych miast oraz 3 satelity. Jednak dynamika problemu jest bardzo mała. Niemniej jednak problem jest odtwarzalny, tzn. dla takich samych danych można uruchomić wiele in-

stancji algorytmów. Każda z wymienionych metod ma swoje wady i zalety. Zaproponowano własne rozwiązanie opisane w dalszej części pracy.

Algorytmy DPSO dla dynamicznej optymalizacji kombinatorycznej

W literaturze można znaleźć relatywnie niewiele pozycji dotyczących algorytmu DPSO dla dynamicznej optymalizacji kombinatorycznej oraz brak pozycji dla problemu DTSP. Okulewicz i Mańdziuk [45] zaproponowali dwuetapowy algorytm DPSO dla dynamicznego marszrutowania. W pierwszej części instancja algorytmu DPSO odpowiada za znalezienie przydziału klienta i pojazdu, podczas gdy druga niezależna od pierwszej instancja algorytmu DPSO znajduje odpowiednią kolejność, w której klienci powinni być obsłużeni (część statyczna). Podobne podejście można znaleźć w pracy Demirtacsa i inni [46]. Khouadjia i inni [47] zaproponowali adaptacyjną wersję DPSO dla problemu dynamicznego marszrutowania z obsługą dynamicznych żądań. Technika przechowuje poprzednie rozwiązania oraz aktualizuje je w czasie działania. Po detekcji zmian w kolejnym podproblemie, poprzednie rozwiązanie (przechowywane w pamięci) jest użyte jako punkt startu. Publikacja Mavrovouniotisa i inni [48] zawiera przegląd najnowszych metod inteligencji obliczeniowej dla dynamicznej optymalizacji kombinatorycznej.

Wybrane algorytmy inteligencji obliczeniowej dla DTSP

Istnieje bardzo wiele prac na temat dynamicznego problemu TSP. Przede wszystkim jest to grupa algorytmów inspirowanych naturą ze względu na samoorganizację. Tabela 1.1 przedstawia wybrane pozycje związane z tematyką.

Tabela 1.1: Wybrane pozycje literatury związanej z DTSP oraz technik inteligencji obliczeniowej użytych do rozwiązywania tego problemu.

Rok	Algorytm	Wariant zmian w DTSP
2001	ACO z lokalnym i globalnym resetem feromonu [49]	Dodawanie/usuwanie wierzchołków

Rok	Algorytm	Wariant DTSP
2002	ACO z różnymi wariantami uaktualnienia macierzy feromonowej, aby utrzymać różnorodność [50]	Zmiana wag krawędzi w czasie (symulowany korek na drodze)
2006	GSInver-Over i pula genów z α-measure [8, 40]	*CHN145*+1 – 145 miast oraz jeden satelita
2010	ACO ze schematem migracji, aby zwiększyć różnorodność populacji [51]	Współczynniki – częstość zmian i wielkość zmian
2011	CHC [52]	Test polegający na dodawaniu zmian, a następnie ich wycofywaniu [44]. W ten sposób na końcu i na początku testu optimum jest takie samo
2012	Równoległy ACO [53]	Zmiana współrzędnych wierzchołków
2014	Algorytm ewolucyjny [54]	Losowe zmiany w problemie
2014	Sieci neuronowe Hopfielda [55]	Symulacja różnego rodzaju rzeczywistych zdarzeń losowych na drodze
2016	ACO ze schematem migracji [56]	Zmiana długości krawędzi. Symulowane opóźnienia pociągów

Pozycje w Tab. 1.1 zawierają opis badań algorytmów populacyjnych dla problemu DTSP. Podstawową kwestią w podejściu do rozwiązywania tego problemu jest strategia uruchomienia algorytmu po zmianie danych – w kolejnej iteracji $t + 1$. Różnorodność podejść wynika z odmienności zastosowanych algorytmów. Jednak cechą wspólną algorytmów populacyjnych jest malejąca różnorodność osobników, która wynika ze zbieżności algorytmu do optimum [57]. Poszukiwanie kolejnych rozwiązań, których długość znajduje się bliżej rozwiązania optymalnego (wg. funkcji ewaluacji) jest wolniejsze. Taka prawidłowość może nie wystąpić tylko w przypadku algorytmów dokładnych, które przeszukują całą dostępną przestrzeń rozwiązań lub stosujących technikę, która gwarantuje znalezienie optimum. Jednak zestawienie

algorytmów w Tab. 1.1 zawiera stochastyczne algorytmy typu Monte Carlo (długość działania algorytmu jest znana, ale nieprzewidywalna jest jakość znalezionego rozwiązania). Przeniesienie całej populacji wraz ze znalezionymi rozwiązaniami i uruchomienie implementacji tego samego algorytmu dla danych z kolejnego podproblemu I^{t+1} może prowadzić do stagnacji. W algorytmach optymalizacyjnych, w tym algorytmach populacyjnych, wyróżnia się dwa etapy działania – eksplorację i eksploatację. W przypadku TSP, w pierwszym etapie przeszukiwanie jest bardzo „szerokie", w drugim etapie następuje zawężenie przeszukiwania, najczęściej ograniczone do poprawy rozwiązań otrzymanych z pierwszego etapu. W kontekście DTSP, uruchomienie implementacji algorytmu bez modyfikacji populacji, przeszukiwanie ograniczałoby się do eksploatacji. Strategia taka może być skuteczna tylko w przypadku niewielkiej liczbie zmian w kolejnych podproblemach. Z tego powodu osobniki najczęściej są losowane dla każdego podproblemu, jak np. w algorytmie zaproponowanym przez Mavrovouniotis [51], w którym implementacja algorytmu dla kolejnego podproblemu uruchamiana jest z resetem całej populacji, oprócz najlepszego osobnika (tzw. imigrant), który wyjątkowo wchodzi w skład nowej populacji. Jeszcze inną strategią jest częściowy reset macierzy feromonowej zastosowany dla algorytmu ACO przez Eyckelhofa oraz Snoeka, co ma wymusić etap eksploracji [50].

Zastosowania w praktyce

W pracy wielokrotnie podkreślano wymiar teoretyczny i praktyczny problemu komiwojażera. Wymiar praktyczny to nie tylko zastosowania w logistyce, ale również w tak odległych dyscyplinach naukowych jak genetyka, czy astronomia [22]. W genetyce za miasto przyjmuje się fragment DNA, a odległością jest miara podobieństwa między fragmentami. W przypadku astronomii, za miasto przyjmuję się obserwację, a za odległości (wagi) ruch teleskopu w ich kierunku. Dynamiczną wersję problemu można zastosować wszędzie tam, gdzie może wystąpić zmiana w danych wejściowych. Przykładowo w logistyce, klienci reprezentują miasta, a zadanie optymalizacyjne polega na skróceniu kosztu obsługi wszystkich klientów. Niektórzy z nich mogą przenieść siedzibę, np. na czas remontu. Ciągła zmiana natężenia ruchu powoduje zator drogowy, co z kolei wpływa na zmianę odległości między punktami dostaw (miastami) [58, 59, 60]. Innym realnym powodem zmiany danych może być wypadek na drodze. We wszystkich tych przypadkach, szyb-

ka adaptacja do zmian jest bardzo pożądana. Jeszcze innym zastosowaniem może być planowanie bezpiecznej trasy dla transportu, np. krwi lub organów. Posiadając dane dotyczące prawdopodobieństwa zatoru na drodze oraz maksymalny czas potrzebny na dostarczenie danego towaru można zaplanować trasę tak, aby z jednej strony dostarczyć ładunek na czas, a z drugiej zrobić to bezpiecznie. Wymaga to algorytmu umożliwiającego szybką adaptację do zmian. W rzeczywistych problemach dane mogą sięgać tysięcy wierzchołków, a uruchamianie po każdej zmianie algorytmu, który nie wykorzystuje informacji o instancji problemu sprzed zmian może nie być dostatecznie wydajne. Takie przykłady można mnożyć, gdyż w praktycznych zastosowaniach dynamiczna zmienność natury wymusza częste zmiany w instancjach problemu.

1.3 Środowisko testowe

Niedeterminizm problemu powoduje brak możliwości porównania dwóch różnych algorytmów, nawet pomimo posiadania ich implementacji, ponieważ nie działają one w środowisku testowym na tych samych danych wejściowych. Badania przeprowadzone przez Bilu i Linial [61] pokazały, że każda modyfikacja danych oprócz możliwości zmiany optimum ma również inne następstwa, takie jak zmiana stopnia trudności problemu. Trudność problemu komiwojażera to odległość rozwiązań od rozwiązania optymalnego oraz minimów lokalnych od siebie. W przypadku, gdy odległość ta jest mała, łatwo trafić na optimum lokalne [61], tym samym trudność problemu wzrasta. W pracach Farhana Ahammeda oraz Pablo Moscato [62] można znaleźć przykłady instancji problemów, dla których algorytm *Concorde* potrzebował około 30 000 razy więcej czasu do znalezienia optimum w stosunku do oryginalnego problemu (bez zmiany rozmiaru problemu) [62].

W pracy przedstawiono repozytorium wygenerowanych problemów bazujących na danych i formacie TSPLIB. W pierwszym podproblemie testowany algorytm rozwiązuje niezmodyfikowany problem komiwojażera. Następnie po każdej zmianie danych (kolejnych podproblemach) ponownie uruchamiany jest ten sam algorytm (Rys. 1). W pracy założono, że każdy problem będzie składał się z jedenastu podproblemów (uruchomienie algorytmu na pierwotnych danych i sekwencji dziesięciu modyfikacji).

Etap wstępny pracy związany jest z przygotowaniem repozytorium spójnych i zunifikowanych danych do testowania algorytmów rozwiązujących dynamiczny problem komiwojażera. Repozytorium utworzone jest na podstawie

Alg. 1 i wzoru (1.2) na zmianę współrzędnych.

Algorytm 1: Generator problemów DTSP.

wejście: liczba zmian w podproblemie p_m,
liczba podproblemów t_{max};
Wczytaj dla $\mathcal{I}^0$ dane z biblioteki TSPLIB;
while $t = 0 \leqslant t_{max}$ **do**
  Rozwiąż $\mathcal{I}^t$ przy użyciu algorytmu dokładnego;
  Zapisz $\mathcal{I}^t$ (współrzędne oraz trasę optymalną);
  $\mathcal{I}^{t+1}$ = kopiuj($\mathcal{I}^t$);
  $w \leftarrow$ Losowe (unikalne) współrzędne wierzchołków z $\mathcal{I}^{t+1}$;
  forall $w_i \in w$ **do**
    Zmień współrzędne zgodnie ze wzorem (1.2);
  end
end

Porównanie rezultatu otrzymanego z implementacji algorytmu wymaga znajomości rozwiązania optymalnego. Informacja ta jest zapisywana w instancji problemu[1]. Równanie (1.2) opisuje zmianę współrzędnych miasta.

$$\begin{aligned} w_x &= w_x + r \cdot \cos(\phi)\,, \\ w_y &= w_y + r \cdot \sin(\phi)\,, \end{aligned} \tag{1.2}$$

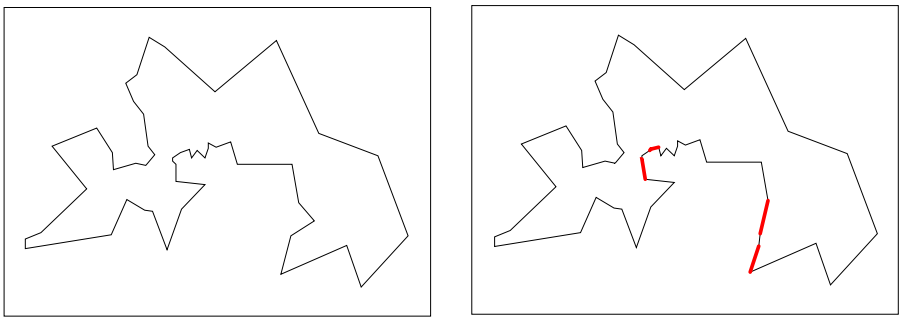
gdzie:

- w_x, w_y to współrzędne miasta,
- r to promień przesunięcia równy $f_{inf} \cdot rand(0, d_{avg})$,
- $f_{inf} \in (0, 1]$ oznacza wpływ nowych współrzędnych na pozycję,
- d_{avg} to średnia odległość wszystkich miast od siebie,
- ϕ to losowy kąt z przedziału $[0, 2\pi]$, w kierunku którego zostaną przesunięte współrzędne miasta.

Parametr f_{inf} określa jak duży wpływ będzie miała zmiana na współrzędne miasta. W pracy założono, że wartość ta będzie wynosić 0,3. Zaletą stosowania powyższego podejścia w stosunku do całkowicie losowej zmiany krawędzi

[1] Jako narzędzie znajdujące optimum przyjęto *NEOS Server*, będący serwisem opartym na protokole *xml-rpc*. Bazuje on na pakiecie do rozwiązywania problemu komiwojażera *Concorde* [63, 64, 65]

jest zachowanie nierówności trójkąta (problem pomimo zmian dalej jest metryczny) oraz kontrola wielkości zmian danych w kolejnych podproblemach $\mathcal{I}^t$. Oprócz wygenerowania danych (współrzędnych miast) zaproponowane środowisko tworzy również dwa rodzaje wykresów: z zaznaczoną trasą optymalną oraz z zaznaczonymi zmianami w rozwiązaniu optymalnym. Rysunek 1.4 przedstawia tę wizualizację dla problemu DTSP opartego na problemie *berlin52*. Po lewej stronie (a) wizualizacja trasy optymalnej zmodyfikowanego problemu (*berlin52*), po prawej stronie (b) kolorem czerwonym zaznaczono krawędzie, którymi różni się rozwiązanie optymalne sprzed zmian w stosunku do rozwiązania po zmianach.

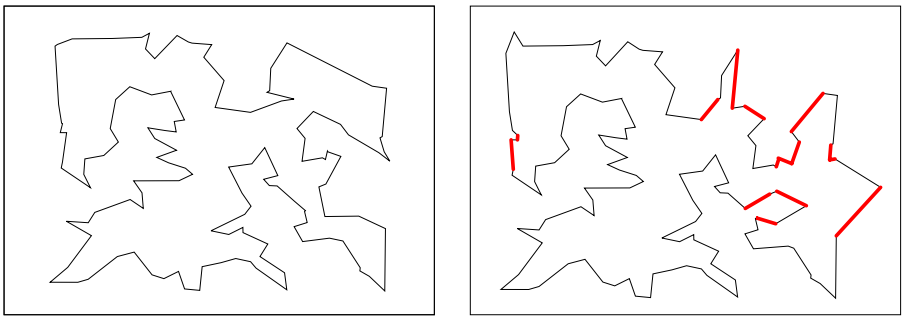


(a) (b)

Rysunek 1.4: Wizualizacja zmian w zmodyfikowanym problemie *berlin52*.

W kolejnej instancji problemu DTSP bazującej na danych pochodzących z problemu komiwojażera *ch130* można zauważyć większe zmiany w rozwiązaniu optymalnym (14 krawędzi zostało zmienionych). Po lewej stronie (a) wizualizacja trasy optymalnej zmodyfikowanego problemu (*ch130*), po prawej stronie (b) kolorem czerwonym zaznaczono krawędzie, którymi różni się rozwiązanie optymalne sprzed zmian w stosunku do rozwiązania po zmianach.
Na Rys. 1.5 w lewym górnym rogu (skrajny punkt) można zaobserwować zmianę współrzędnych miasta bez zmiany w tym miejscu krawędzi należących do rozwiązania optymalnego.

Bieżące repozytorium zawiera 64 problemy. Bazują one na ośmiu problemach pochodzących z biblioteki TSPLIB: *berlin52*, *kroA100*, *ch130*, *kroA200*, *gil262*, *gr202*, *pcb442*, *gr666* i dla każdego z nich przyjęto osiem stopni mo-

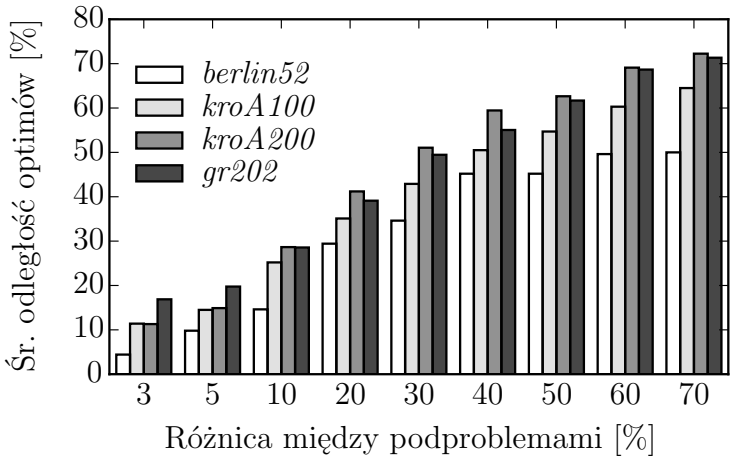


(a) (b)

Rysunek 1.5: Wizualizacja zmian w zmodyfikowanym problemie *ch130*.

dyfikacji oryginalnych danych w każdym podproblemie: 3%, 5%, 10%, 20%, 30%, 40%, 50%, 60% (tj. procent współrzędnych podlegających modyfikacji).

Rysunek 1.6 przedstawia wykres liczby zmian w krawędziach tworzących nowe rozwiązanie optymalne (po zmianach) w stosunku do procentowej liczby zmian w danych problemu (zmian współrzędnych miast – zmian w macierzy odległości). Porównanie dotyczyło liczby nowych krawędzi a procentowa liczba zmian w podproblemie (która jest równa liczbie zmienionych współrzędnych miast do wszystkich miast).



Rysunek 1.6: Średnia różnica w odległości rozwiązań optymalnych między kolejnymi podproblemami w ramach jednego problemu DTSP.

Rozdział 2

Wybrane zagadnienia inteligencji obliczeniowej

W rozdziale tym zostaną opisane pokrótce najważniejsze pojęcia związane z inteligencją obliczeniową. Podrozdział 2.1 zawiera wprowadzenie do tematyki. W podrozdziale 2.2 zostanie opisany klasyczny algorytm PSO oraz jego najważniejsze modyfikacje, w tym wersja dyskretna. W podrozdziale 2.3 zostanie opisany feromon stanowiący charakterystyczny element algorytmów mrowiskowych. Feromon stanowi ważny element algorytmu DPSO zaproponowanego w pracy oraz główny mechanizm adaptacji do zmian w problemie DTSP.

2.1 Wprowadzenie

Systemem naturalnym nazywamy system pochodzący z inspiracji fizycznych, biologicznych i społecznych [20]. Ewolucja oraz konkurencja wytworzyła wiele zachowań, które po wnikliwych analizach naukowców uznawane są za bardzo efektywne. Z tego też powodu od lat inspirują one naukowców, architektów i inżynierów. Jako przykład można podać termitiery (gniazda) budowane przez afrykański gatunek termitów *Macrotermitinae* [66, 67]. Wewnątrz tych budowli panują bardzo stabilne warunki pod względem wilgotności, temperatury i zawartości dwutlenku węgla, co powoduje, że termitiery są pod względem budowy, konstrukcjami bardzo skomplikowanymi. Owady, takie jak termity, mrówki czy pszczoły wykorzystują efekt synergii czyli kooperacji osobników w celu wykonywania prac, które indywidualnie nie byłyby w stanie wyko-

nać. Tworzą one kolektyw oraz hierarchię społeczną, w której każda grupa ma określoną funkcję. Jest to tzw. zachowanie kolektywne. Wielu badaczy próbowało naśladować zachowania stadne w symulacjach komputerowych.

Prace Craiga Reynoldsa były próbą stworzenia ogólnego algorytmu stada. W roku 1987 na konferencji *SIGGRAPH* Reynolds przedstawił swój algorytm, w którym pojedynczy obiekt nazywany jest boidem [68]. Wnioski z pracy prowadzą do konkluzji, że łącząc kilka prostych reguł można odzwierciedlić bardzo skomplikowane, naturalne zachowania stadne w symulacji komputerowej. Reguły wprowadzone w podstawowej wersji algorytmu to:

- rozdzielność (ang. *separation*) – unikanie zatłoczonych miejsc wewnątrz lokalnej grupy boidów,
- spójność (ang. *cohesion*) – poruszanie się w kierunku centrum lokalnej grupy,
- wyrównanie (ang. *alignment*) – dopasowanie kierunku ruchu boida do uśrednionej wartości funkcji celu (dla lokalnej grupy).

Lokalność w schemacie działania oraz przynależność do grupy określana jest na podstawie przyjętej funkcji sąsiedztwa. Podobne badania prowadzili Frank Heppner (zoolog) i Ulf Grenander (naukowiec) [69]. W pracy, autorzy starali się odkryć wzorzec ruchu osobników w stadzie lub roju. Stado często w sposób nagły zmienia kierunek lotu, ulega rozproszeniu oraz przegrupowaniu, a pomimo tej dynamiki zmian nie występują kolizje między osobnikami. Autorzy usiłowali wyrazić poprzez nieliniowe równania różniczkowe reguły sterujące lotem. Prace Jamesa Kennedy'ego oraz Russella Eberharta [70] zaowocowały powstaniem algorytmu PSO, który inspirowany jest pracami Reynoldsa oraz Heppnera i Grenandera oraz wykorzystuje synergię [71, 20]. Interpretując działanie algorytmu, wirtualne cząsteczki (osobniki) naśladują zachowania społeczne, komunikują się między sobą wykorzystując swoje doświadczenie oraz wiedzę dostępną w obrębie swojego stada (lokalnej grupy). Stopniowo członkowie populacji poprzez wzajemne interakcje przemieszczają się w stronę lepszych regionów przestrzeni rozwiązań, doprowadzając do osiągnięcia potencjalnego optimum.

Innym przykładem techniki wykorzystującej efekt synergiczny to algorytmy mrowiskowe i mrówkowe. Ogólna idea polega na wykorzystaniu dodatniego sprzężenia zwrotnego, otrzymanego z kooperacji mrówek. Podstawowym nośnikiem informacji jest feromon, który w klasycznym algorytmie PSO nie

występuje. Algorytmy inspirowane mrówkami z sukcesem zostały zastosowane w wielu problemach dyskretnych, w tym w rozwiązywaniu problemu TSP [9, 20]. W ramach algorytmów uczenia maszynowego jest to przykład uczenia ze wzmocnieniem [21, 72].

Powyższe przykłady algorytmów zaklasyfikowano do rodziny metaheurystyk. Pojęcie heurystyki w kontekście problemów optymalizacyjnych oznacza pewną metodę znajdowania rozwiązania przybliżonego (bez gwarancji znalezienia optimum). Metaheurystyka to uogólniona heurystyka, tj. pewna idea lub inaczej pewien ogólny schemat postępowania. Użycie metaheurystyki do konkretnego problemu wymaga uściślenia ogólnych pojęć używanych w definicji metaheurystyki. W przypadku algorytmu PSO pojęcia: pozycji i prędkości muszą być dopasowane do konkretnego problemu. Dla optymalizacji funkcji będzie to wektor, dla optymalizacji dyskretnej będzie to zbiór krawędzi.

2.2 Algorytm PSO i jego warianty

W podrozdziale zostaną omówione najważniejsze warianty algorytmu PSO. Dokładne zrozumienie schematu działania podstawowej wersji algorytmu jest niezbędne do zrozumienia wariantów algorytmu dla innych przestrzeni i problemów niż ciągła optymalizacja globalna. Ze względu na ograniczony obszar tematyki, który będzie pomocny w kolejnych rozdziałach pracy, zostanie omówiony model PSO klasyczny, ze ściskiem oraz wersja binarna i dyskretna.

Klasyczny algorytm PSO

Każdy osobnik (cząsteczka) reprezentuje pojedyncze rozwiązanie w d-wymiarowej przestrzeni rozwiązań. Populacja (stado) porusza się po wirtualnej przestrzeni rozwiązań i wykorzystuje kooperację w celu znalezienia optimum. Wzory (2.1) i (2.2) opisują ruch stada cząsteczek w przestrzeni rozwiązań w $\mathbb{R}^d$.

$$\begin{aligned} \vec{v}_i^{k+1} \leftarrow \vec{v}_i^{k} & \\ & + c_1 rand() \otimes (p\vec{Best}_i - \vec{x}_i^{k}) \\ & + c_2 rand() \otimes (g\vec{Best} - \vec{x}_i^{k}) \,, \end{aligned} \tag{2.1}$$

$$\vec{x}_i^{k+1} \leftarrow \vec{x}_i^{k} + \vec{v}_i^{k+1} \,, \tag{2.2}$$

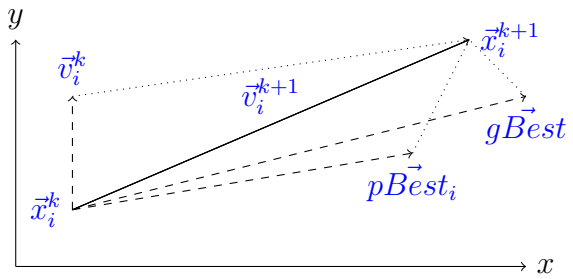
$$\vec{v},\ \vec{x},\ g\vec{Best},\ p\vec{Best}_i \in \mathbb{R}^d \,,$$

gdzie:

- $\vec{v}_i^{k}$ to prędkość cząsteczki i w iteracji k (kierunek przeszukiwania przestrzeni rozwiązań),
- $\vec{x}_i^{k}$ to pozycja cząsteczki i w iteracji k, będąca jednym z rozwiązań problemu,
- $rand()$ to funkcja zwracająca wartość z rozkładu jednostajnego z przedziału $[0, 1]$,
- $p\vec{Best}_i$ i $g\vec{Best}$ oznaczają: najlepsze rozwiązanie znalezione przez cząsteczkę i oraz najlepsze dotychczas znalezione rozwiązanie,
- c_1 i c_2 to parametry określane jako stałe przyspieszenia: kognitywny i społeczny, które skalują znaczenie wpływu kolejno $pBest_i$ i $gBest$,
- operator $\otimes$ w przypadku przestrzeni ciągłej oznacza iloczyn wartości skalarnej i wektora.

Kolejna pozycja cząsteczki zależy od: wartości losowej $rand()$ (czynnik losowy), pozycji innych cząsteczek (poprzez $g\vec{Best}$), najlepszego znalezionego rozwiązania przez cząsteczkę (poprzez $p\vec{Best}_i$), poprzedniej pozycji oraz poprzedniej prędkości. Cząsteczki mają tendencję do kierowania się w stronę coraz to lepszych rozwiązań. Rysunek 2.1 prezentuje schemat obliczania kolejnej pozycji $\vec{x}_i^{k+1}$ przez cząsteczkę w przestrzeni $\mathbb{R}^2$.

Algorytm 2 przedstawia schemat działania PSO dla optymalizacji w przestrzeni ciągłej (minimalizacji). Na początku pozycja każdej cząsteczki jest losowa. Następnie, w każdej iteracji algorytmu obliczana jest jej kolejna prędkość (kierunek przeszukiwania), a na jej podstawie – kolejna pozycja (nowe



Rysunek 2.1: Ruch pojedynczej cząsteczki w PSO.

rozwiązanie). W przypadku znalezienia lepszego rozwiązania następuje aktualizacja wektorów $\vec{gBest}$ i $\vec{pBest}_i$. W przeciwieństwie do wielu algorytmów populacyjnych, m.in. algorytmów genetycznych (w wyniku działania sukcesji), otrzymanie gorszego rozwiązania nie powoduje powrotu do poprzedniego rozwiązania.

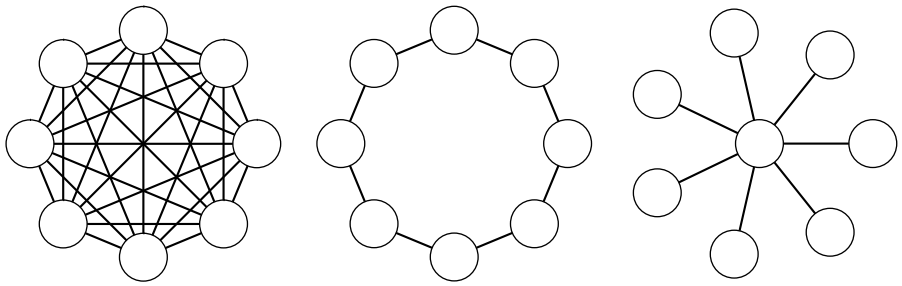
Pod wpływem wielu badań empirycznych oszacowano, że wartości parametrów c_1 oraz c_2 powinny spełniać równanie $c_1 + c_2 = 4$ [73, 74]. Gdy prędkość cząsteczki v jest relatywnie mała, algorytm eksploatuje przestrzeń rozwiązań przeszukując swoje otoczenie (rozwiązania sąsiednie). Relatywnie duża wartość powoduje łatwiejsze opuszczenie lokalnego optimum, ale może powodować zbyt chaotyczne przeszukiwanie przestrzeni rozwiązań. Drugi przypadek oznacza fazę eksploracji. W celu eliminacji sytuacji, w której prędkość będzie zbyt duża, wprowadzono dodatkowe ograniczenie w postaci przedziału dopuszczalnej wartości prędkości cząsteczki $[-V_{max}, V_{max}]$. W ramach jednego algorytmu można przyjąć różne topologie połączeń między cząsteczkami. Można wtedy mówić o podzbiorze cząsteczek (rodzinie) w ramach stada, które są w relacji sąsiedztwa między sobą. Komunikacja dotyczy rozwiązania $\vec{gBest}$, które współdzielą tylko cząsteczki połączone ze sobą. Powiązania pełnią rolę sieci społecznościowej, co ma nawiązywać do fenomenu biologicznego na bazie którego algorytm PSO powstał. Topologię można stworzyć na bazie grafu nieskierowanego, w którym wierzchołki reprezentują cząsteczki, a krawędzie to pary cząsteczek, które współdzielą $\vec{gBest}$. Rysunek 2.2 przedstawia wybrane regularne topologie połączeń między cząsteczkami. Kolejno od lewej: (a) pełne połączenie (ang. *fully connected topology*); (b) pierścień (ang. *ring topology*); (c) gwiazda (ang. *star topology*). W literaturze można spotkać dynamiczne topologie, które zmieniają się w kolejnych iteracjach [74].

Algorytm 2: Algorytm PSO dla problemu minimalizacji w przestrzeni ciągłej.

wejście: liczba iteracji k_{max},
rozmiar stada i_{max},
wymiar przestrzeni d,
funkcja ewaluacji (oceny) rozwiązania $f : \mathbb{R}^d \to \mathbb{R}$;
Utwórz losowe stado;
Oceń każdą pozycję cząsteczek w stadzie;
Wyznacz $g\vec{B}est$ na podstawie ww. oceny;
for $k = 0 \to k_{max}$ **do**
  for $i = 0 \to i_{max}$ **do**
    for $l = 0 \to d$ **do**
      $r_1 \leftarrow rand() \cdot c_1$ i $r_2 \leftarrow rand() \cdot c_2$;
      $\vec{v}_{i,l}^{k+1} \leftarrow \vec{v}_{i,l}^{k} + r_1 \cdot (p\vec{B}est_{i,l} - \vec{x}_{i,l}^{k}) + r_2 \cdot (g\vec{B}est_{l} - \vec{x}_{i,l}^{k})$;
      $\vec{x}_{i,l}^{k+1} \leftarrow \vec{x}_{i,l}^{k} + \vec{v}_{i,l}^{k+1}$;
    end
    if $f(p\vec{B}est_i) > f(\vec{x}_i^{k+1})$ **then**
      $p\vec{B}est_i \leftarrow \vec{x}_i^{k+1}$;
      if $f(p\vec{B}est_i) > f(\vec{x}_i^{k+1})$ **then**
        $g\vec{B}est \leftarrow \vec{x}_i^{k+1}$;
      end
    end
  end
end
wyjście: $gBest$

Model ze ściskiem

Klasyczny model algorytmu PSO nie posiadał mechanizmów umożliwiających płynne przejście od eksploracji (duże skoki algorytmu w przestrzeni rozwiązań) do eksploatacji (przegląd sąsiedztwa). Każdy z tych etapów jest niezwykle istotny dla optymalizacji. Bez etapu eksploracji proces przeszukiwania przestrzeni rozwiązań mógłby utknąć w minimum/maksimum lokal-



(a) (b) (c)

Rysunek 2.2: Rodzaje sieci społecznościowych w algorytmie PSO.

nym. Bez etapu eksploatacji proces ten mógłby zatrzymać się zbyt wcześnie, chociaż optimum globalne byłoby w najbliższym sąsiedztwie bieżącego rozwiązania. Kolejny model PSO zaproponowany przez Yuhui Shi i Russella Eberharta zawierał dodatkowy parametr bezwładności ω [75, 21]. Wartość tego parametru empirycznie została oszacowana na 0,9 w pierwszej iteracji algorytmu i powinna maleć, aż do wartości 0,4 w ostatniej iteracji (spadek liniowy). Wzory (2.1) i (2.2) z uwzględnieniem bezwładności cząsteczki przyjmują postaci wzorów (2.3) i (2.4).

$$\begin{aligned} \vec{v}_i^{k+1} &\leftarrow \omega \vec{v}_i^{k} \\ &\quad + rand() c_1 \otimes (p\vec{Best}_i - \vec{x}_i^{k}) \\ &\quad + rand() c_2 \otimes (g\vec{Best} - \vec{x}_i^{k}) \,, \end{aligned} \tag{2.3}$$

$$\vec{x}_i^{k+1} \leftarrow \vec{x}_i^{k} + \vec{v}_i^{k+1} \,, \tag{2.4}$$

$$\vec{v}, \vec{x}, g\vec{Best}, p\vec{Best}_i \in \mathbb{R}^d \,,$$

gdzie ω to wartość bezwładności cząsteczki.

Współczynnik ω można interpretować jako miarę inercji (bezwładności) cząsteczki. Duże wartości ω powodują, że prędkość $\vec{v}_i^{k}$ będzie proporcjonalnie większa. Determinuje to skokowe przejścia do kolejnych pozycji $\vec{x}_i^{k+1}$. Malejąca wartość bezwładności powoduje, że prędkość cząsteczki $\vec{v}_i^{k}$ będzie maleć, a cząsteczka będzie przemieszczać się w obrębie swojego najbliższego otoczenia [74, 21]. W literaturze można znaleźć inne metody wyznaczania wartości ω w kolejnych iteracjach, np. poprzez system rozmyty [76].

Binarna wersja PSO

Asumptem do zastosowania algorytmu PSO dla przestrzeni dyskretnej i binarnej były liczne wdrożenia i rosnąca popularność metaheurystki [74]. Adaptacja polegała na zmianie definicji takich pojęć jak pozycja i prędkość, a operatory dodawania, odejmowania i mnożenia oznaczały operacje binarne. Prekursorzy algorytmu PSO – James Kennedy i Russell Eberhart zaproponowali prostą implementację techniki do przestrzeni binarnej [77], która przyjęła nazwę binarnego PSO (BPSO). W nowym ujęciu pozycja cząsteczki i w iteracji k $\vec{x}_i^k$ to wektor binarny, a prędkość $\vec{v}_i^k$ to wektor prawdopodobieństw przypisania każdej składowej pozycji cząsteczki bitu o wartości zero lub jeden. Wektor prędkości i wektor pozycji mają równy rozmiar, a każdemu elementowi wektora prędkości odpowiada dokładnie jeden element wektora pozycji (element o tych samym indeksie). Wzór na prędkość cząsteczki pozostaje taki sam, jak w klasycznym PSO. Zmianie uległ wzór na obliczenie kolejnego elementu wektora pozycji (2.5), zastępując wzory (2.2) lub (2.4).

$$\vec{x}_i^{k+1} = \begin{cases} 0 & \text{jeżeli } rand() \geqslant sig(\vec{v}_i^{k+1}), \\ 1 & \text{jeżeli } rand() < sig(\vec{v}_i^{k+1}), \end{cases} \tag{2.5}$$

gdzie:

- sig to funkcja sigmoidalna przekształcająca prędkość cząsteczki w jej pozycję daną wzorem:

$$sig(\vec{v}_i^{k+1}) = \frac{1}{e^{-\vec{v}_i^{k+1}}}, \tag{2.6}$$

Wzór (2.5) należy obliczyć dla każdego elementu wektora pozycji. Należy również zauważyć, że powyższa heurystyka jest podstawowym schematem adaptacji metaheurytyki PSO dla przestrzeni binarnej. W literaturze można znaleźć inne propozycje [78, 79, 80, 81].

Dyskretna wersja PSO

W metaheurystyce PSO pozycja i prędkość w każdej adaptacji mają inne znaczenie – dla przestrzeni ciągłej to d-wymiarowe wektory liczb rzeczywistych, dla przestrzeni binarnej d-wymiarowe wektory liczb binarnych. Algorytm dyskretnego PSO (DPSO) zaproponowany przez Zhonga Wen-lianga,

Zhanga Juna, Chena Wei-nenga bazuje na zbiorach krawędzi [13]. W celu adaptacji metaheurystyki do tej przestrzeni wiele koncepcji należało przedefiniować. Pozycja cząsteczki i w iteracji k (X_i^k) jest zbiorem krawędzi tworzących cykl Hamiltona. Prędkość V_i^k to „kierunek" przeszukiwania przestrzeni rozwiązań w iteracji k dla cząsteczki i. Jest to zbiór krawędzi, który może zawierać częściowe rozwiązanie problemu (nie zawierać wszystkich krawędzi potrzebnych do stworzenia cyklu Hamiltona) lub posiadać krawędzie nadmiarowe. Zbiór ten zawiera krawędzie i można go interpretować jako zbiór krawędzi, o które kolejna pozycja cząsteczki może zostać uzupełniona, przemieszczając cząsteczkę w inne miejsce dyskretnej przestrzeni rozwiązań. Pojedyncza krawędź algorytmu DPSO $\hat{e}$ to uporządkowana dwójka postaci:

$$\hat{e} = \langle p, \langle a, b \rangle \rangle \,, \tag{2.7}$$

z ograniczeniami:

$$p \in [0, 1] \,, \tag{2.8a}$$
$$a, b \in \mathcal{V} \,, \tag{2.8b}$$
$$a \neq b \,, \tag{2.8c}$$
$$\langle a, b \rangle \in \mathcal{E} \,, \tag{2.8d}$$

gdzie:

- $\mathcal{V}$ to zbiór wierzchołków grafu skierowanego $\mathcal{G}$ instancji problemu TSP,
- $\mathcal{E}$ to zbiór krawędzi grafu skierowanego $\mathcal{G}$ instancji problemu TSP,
- p oznacza prawdopodobieństwo wyboru krawędzi do następnej pozycji. Znaczenie i sposób wyznaczania parametru p zostaną wyjaśnione w dalszej części pracy.

Nowa formalna definicja krawędzi może prowadzić do nieścisłości, gdyż omawiany problem, jak i algorytm inaczej definiują termin „krawędź". W pracy przyjęto następującą terminologię: krawędź problemu e to krawędź instancji problemu TSP, krawędź algorytmu $\hat{e}$ to krawędź opisana wzorem (2.7).
Cechą wspólną wszystkich algorytmów optymalizacji stadnej cząsteczek jest działanie bazujące na równaniu opisującym ruch cząsteczki w przestrzeni rozwiązań. Wzory (2.9) i (2.10) przedstawiają dyskretną wersję tych równań zaproponowanych w [13].

$$\begin{aligned} V_i^{k+1} &= c_2 rand() \cdot (gBest \setminus X_i^k) \\ &\cup c_1 rand() \cdot (pBest_i \setminus X_i^k) \\ &\cup \omega \cdot V_i^k \,, \end{aligned} \tag{2.9}$$

$$X_i^{k+1} = \underbrace{V_i^{k+1}}_{\text{krok 2.10a}} \oplus \underbrace{c_3 rand() \cdot X_i^k}_{\text{krok 2.10b}} , \tag{2.10}$$

gdzie:

- i oznacza numer cząsteczki,
- k to numer iteracji,
- X_i^k to zbiór krawędzi algorytmu postaci (2.7) oznaczający pozycję cząsteczki i, który jest rozwiązaniem problemu komiwojażera,
- V_i^k to zbiór krawędzi $\hat{e}$ (2.7), oznaczający prędkość cząsteczki,
- $rand()$ oznacza wartość losową z przedziału $[0, 1]$,
- $pBest_i$ i $gBest$ to zbiory krawędzi $\hat{e}$, oznaczają kolejno najlepszą pozycję cząsteczki i oraz najlepsze znalezione rozwiązanie,
- współczynnik ω nazywany jest współczynnikiem inercji stada,
- parametry c_1 i c_2 to odpowiednio parametr kognitywny i socjalny,
- parametr c_3 nie występuje w poprzednich algorytmach – PSO i BPSO. Skaluje wpływ poprzedniej pozycji na jej kolejną postać,
- operator $\oplus$ dodaje do zbioru X_i^{k+1} krawędzi algorytmu z poprzedniej pozycji. Działanie tego operatora zostało wyjaśnione w Alg. 3,
- iloczyn liczby i zbioru krawędzi $\hat{e}$ to mnożenie liczby z każdym parametrem p znajdującym się w elementach zbioru. Niech M oznacza pewien zbiór krawędzi $\hat{e}$, $r \in \mathbb{R}$, formalnie operację można zapisać jako:

$$\begin{aligned} r \cdot M &= r \cdot \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_{|M|}\} \,, \\ &= \{r \cdot \hat{e}_1, r \cdot \hat{e}_2, \ldots, r \cdot \hat{e}_{|M|}\} \,, \\ &:= \{\forall_{\hat{e} \in M} : r \cdot \hat{e}\} \,. \end{aligned} \tag{2.11}$$

  Mnożenie liczby r ($r \in \mathbb{R}$) i krawędzi $\hat{e}$ ($\hat{e} \in M$) formalnie można zdefiniować następująco:

$$\begin{aligned} r \cdot \hat{e} &= r \cdot \langle p, \langle a, b \rangle \rangle \,, \\ &= \langle r \cdot p, \langle a, b \rangle \rangle \,. \end{aligned} \tag{2.12}$$

Powyższe definicje związane z krawędziami postaci (2.7) oraz wszystkie operacje z nimi związane zostały przedstawione bazując na grafie skierowanym (ważonym z powodu adaptacji do problemu TSP). Jednak w przypadku DPSO dla symetrycznego problemu TSP (z symetrycznymi odległościami w macierzy odległości D), definicja krawędzi $\hat{e}$ i wszystkich z nią związanych operacji musi uwzględnić nową postać. Krawędź algorytmu $\hat{e}_u$ przyjmuje formę: $\langle p, \{a, b\}\rangle$. Operacje na krawędziach algorytmu są analogiczne, ponieważ dotyczą parametru p. Dwie krawędzie algorytmu $\langle p_1, \langle a, b\rangle\rangle$ i $\langle p_2, \langle c, d\rangle\rangle$ lub $\langle p_1, \{a, b\}\rangle$ i $\langle p_2, \{c, d\}\rangle$ są równe jeżeli:

$$\begin{aligned} \langle p_1, \langle a, b\rangle\rangle = \langle p_2, \langle c, d\rangle\rangle &\iff a = c \wedge b = d\,, \\ \langle p_1, \{a, b\}\rangle = \langle p_2, \{c, d\}\rangle &\iff a = c \wedge b = d \vee a = d \wedge b = c\,. \end{aligned} \tag{2.13}$$

Przykładowe rozwiązanie asymetrycznego problemu komiwojażera:

$$\{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 4\rangle, \langle 4, 1\rangle\}\,,$$

w DPSO przyjmuje postać:

$$\{\langle 1, \langle 1, 2\rangle\rangle, \langle 1, \langle 2, 3\rangle\rangle, \langle 1, \langle 3, 4\rangle\rangle, \langle 1, \langle 4, 1\rangle\rangle\}\,.$$

Z kolei przykładowe rozwiązanie symetrycznego problemu komiwojażera to:

$$\{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}\}\,,$$

w DPSO przyjmuje postać:

$$\{\langle 1, \{1, 2\}\rangle, \langle 1, \{2, 3\}\rangle, \langle 1, \{3, 4\}\rangle, \langle 1, \{4, 1\}\rangle\}.$$

Alg. 3 przedstawia schemat działania DPSO na podstawie wzorów (2.9) i (2.10).
W pierwszych krokach tworzone jest losowe położenie (w przestrzeni rozwiązań) stada cząsteczek. Na jego podstawie wybierana jest najlepsza pozycja – $gBest$. Następnie obliczana jest kolejna prędkość każdej cząsteczki zgodnie ze wzorem (2.9) (linia 8, Alg. 3). Operacja mnożenia liczby ze zbiorem wykonywana jest jako mnożenie liczby i każdego współczynnika p w elementach zbioru (krawędziach algorytmu), zgodnie ze wzorami (2.11) i (2.12). Prawdopodobieństwo wyboru krawędzi algorytmu do następnej pozycji uzależnione jest od współczynników c_1, c_2, c_3 oraz wartości losowej $rand()$. Operacja dodawania i odejmowania to odpowiednio suma i różnica zbiorów. W ten

Algorytm 3: Algorytm DPSO.

```
1  wejście: liczba iteracji k_max,
             rozmiar stada i_max;
2  Utwórz losowe stado;
3  Oceń każdą pozycję cząsteczek w stadzie;
4  Wyznacz gBest na podstawie ww. oceny;
5  for k = 0 → k_max do
6      for i = 0 → i_max do
7          Oblicz prędkość V_i^{k+1} (wzór (2.9));
8          Filtrowanie krawędzi;
9          X_i^{k+1} = ∅;
10         forall ⟨p, ⟨a, b⟩⟩ ∈ V_i^{k+1} do          // wzór (2.10a)
11             if rand() ⩽ p then
12                 X_i^{k+1} ∪ ⟨p, ⟨a, b⟩⟩;
13             end
14         end
15         forall ⟨p, ⟨a, b⟩⟩ ∈ X_i^k do              // wzór (2.10b)
16             if rand() ⩽ rand() · c_3 then
17                 X_i^{k+1} ∪ ⟨p, ⟨a, b⟩⟩;
18             end
19         end
20         Etap dopełnienia;
21         Dopełnij X_i^{k+1} przeglądając sąsiedztwo;
22         Uaktualnij pBest_i i gBest;
23     end
24 end
25 wyjście: gBest
```

sposób obliczana jest kolejna prędkość cząsteczki. Tworzenie nowego rozwiązania składa się z dwóch etapów – filtrowania i dopełnienia. W pierwszym z nich, każda krawędź algorytmu ze zbioru kolejnej prędkości V_i^{k+1} wybierana jest do niepełnego zbioru kolejnej pozycji X_i^{k+1}, jeśli jej współczynnik p jest większy bądź równy losowej wartości otrzymanej z funkcji $rand()$ (linia 12). W kolejnym etapie następuje dopełnienie tego zbioru do cyklu Hamiltona. Etap ten składa się z dwóch kroków. W pierwszym, niepełny zbiór następnej pozycji X_i^{k+1} dopełnia się krawędziami algorytmu z poprzedniej pozycji

X_i^k (linie 16-20). Następnie zbiór ten dopełniany jest krawędziami algorytmu pochodzącymi z heurystyki najbliższego sąsiada, co prowadzi do uzupełniania zbioru do pełnego cyklu Hamiltona. Cały proces w kolejnej iteracji jest powtarzany.

Niech $\mathcal{G}_u$ oznacza nieskierowany graf zdefiniowany następująco: $\mathcal{V}_u$= $\{1,2,3,4,5,6\}$, $|\mathcal{E}_u| = \binom{n}{2}$ (wszystkie kombinacje dwuelementowe zbioru $\mathcal{V}_u$). Dana jest cząsteczka o indeksie 0 w iteracji 1:

$$X_0^1 = \{\langle 1,\{1,4\}\rangle,\langle 1,\{4,2\}\rangle,\langle 1,\{2,5\}\rangle,\langle 1,\{5,3\}\rangle,\langle 1,\{3,6\}\rangle,\langle 1,\{6,1\}\rangle\},$$
$$V_0^1 = \{\langle 1,\{2,5\}\rangle\},$$
$$gBest = \{\langle 1,\{1,2\}\rangle,\langle 1,\{2,3\}\rangle,\langle 1,\{3,4\}\rangle,\langle 1,\{4,5\}\rangle,\langle 1,\{5,6\}\rangle,\langle 1,\{6,1\}\rangle\},$$
$$pBest_0 = \{\langle 1,\{1,2\}\rangle,\langle 1,\{2,3\}\rangle,\langle 1,\{3,5\}\rangle,\langle 1,\{5,4\}\rangle,\langle 1,\{4,6\}\rangle,\langle 1,\{6,1\}\rangle\}.$$

Wynikiem wzoru (2.9) są następujące zbiory:

$$gBest \setminus X_0^1 = \{\langle 1,\{1,2\}\rangle,\langle 1,\{2,3\}\rangle,\langle 1,\{3,4\}\rangle,\langle 1,\{4,5\}\rangle,\langle 1,\{5,6\}\rangle\},$$
$$pBest_0 \setminus X_0^1 = \{\langle 1,\{1,2\}\rangle,\langle 1,\{2,3\}\rangle,\langle 1,\{5,4\}\rangle,\langle 1,\{4,6\}\rangle\}.$$

Kolejna prędkość cząsteczki V_0^2 po wymnożeniu przez ($c_1 rand()$, $c_2 rand()$ lub $\omega rand()$) jest równa:

$$(gBest \setminus X_0^1) \cup (pBest_i \setminus X_0^1) \cup V_0^1 = \{\langle 0,3,\{1,2\}\rangle,\ \langle 0,1,\{2,3\}\rangle,$$
$$\langle 0,5,\{3,4\}\rangle,\ \langle 0,6,\{4,5\}\rangle,\ \langle 0,1,\{5,6\}\rangle,\ \langle 0,2,\{1,2\}\rangle,$$
$$\langle 0,9,\{2,3\}\rangle,\langle 0,7,\{5,4\}\rangle,\ \langle 0.4,\{4,6\}\rangle\}.$$

Krawędź pochodząca z poprzedniej prędkości nie została dodana do sumy, ponieważ wierzchołek nie może wystąpić w kolejnej prędkości więcej niż cztery razy (wierzchołek 2 byłby stopnia 5 ($\deg(2) = 5$) [13]. Zakładając, że wylosowano następujące wartości: 0,1; 0,7; 0,49; 0,5; 0,9; 0,3; 0,6; 0,55; 0,39, po procesie filtracji, niepełny zbiór kolejnej pozycji cząsteczki jest równy:

$$X_0^2 = \{\langle 0,3,\{1,2\}\rangle,\ \langle 0,5,\{3,4\}\rangle,\ \langle 0,6,\{4,5\}\rangle,\langle 0,9,\{2,3\}\rangle,\ \langle 0,7,\{5,4\}\rangle,$$
$$\langle 0,4,\{4,6\}\rangle\}.$$

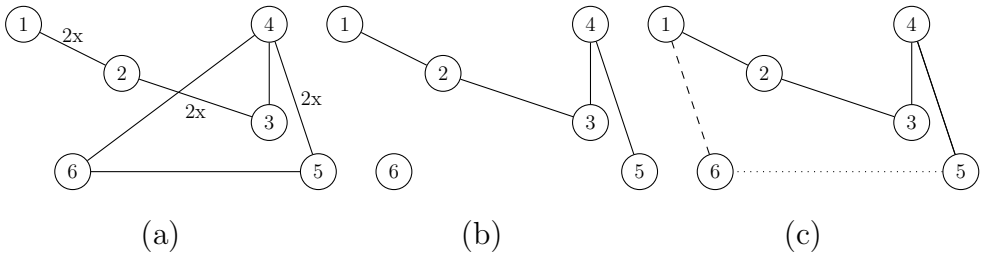
Krawędź algorytmu $\langle 0,4,\{4,6\}\rangle$ jest odrzucona, ponieważ wierzchołek 4 jest stopnia 3. Krawędź algorytmu $\langle 0,7,\{5,4\}\rangle$ jest również odrzucona, ponieważ krawędź algorytmu o końcach $\{4,5\}$ znajduje się już w zbiorze kolejnej pozycji. Kolejna, niepełna pozycja cząsteczki wynosi:

$$X_0^2 = \{\langle 0,3,\{1,2\}\rangle,\ \langle 0,5,\{3,4\}\rangle,\ \langle 0,6,\{4,5\}\rangle,\ \langle 0,9,\{2,3\}\rangle\}.$$

Rezultat ten jest spełnieniem wzoru (2.10a). Operacja $\oplus$ dopełnia krawędź algorytmu $\langle 1, \{6, 1\}\rangle$ do X_0^2 na podstawie zbioru określającego poprzednią pozycję cząsteczki. W celu stworzenia pełnego cyklu Hamiltona, proces dopełnia ma na celu uzupełnienie brakujących krawędzi z heurystyki najbliższego sąsiada: $\langle 1, \{5, 6\}\rangle$. Finalnie, pozycja cząsteczki wynosi:

$$X_0^2 = \{\langle 1, \{1, 2\}\rangle, \langle 1, \{2, 3\}\rangle, \langle 1, \{3, 4\}\rangle, \langle 1, \{4, 5\}\rangle, \langle 1, \{5, 6\}\rangle, \langle 1, \{6, 1\}\rangle\}.$$

Rysunek 2.3 przestawia wizualizację wszystkich operacji. Krawędzie algorytmu ze zbioru V_i^{k+1} przed procesem filtracji (a), po procesie filtracji (b) i po utworzeniu pełnego cyklu Hamiltona (c). Przerywanymi liniami oznaczono krawędzie algorytmu pochodzące ze wzoru (2.10) krok (2.10b), kropkowane linie oznaczają krawędzie algorytmu otrzymane z procesu dopełnienia (c).

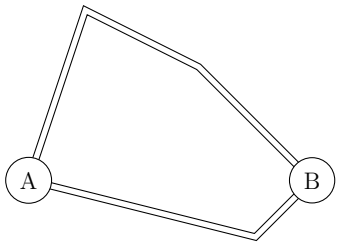


Rysunek 2.3: Przykład tworzenia nowej pozycji w algorytmie DPSO.

2.3 Feromon w algorytmach mrowiskowych

System mrowiskowy jest niedeterministycznym algorytmem populacyjnym, osobniki – wirtualne mrówki nie posiadają inteligencji własnej, natomiast dysponują inteligencją zbiorową [20]. Asumptem do powstania algorytmów mrowiskowych były prace naukowe poświęcone strategii wyszukiwaniu najkrótszej drogi od mrowiska do źródła pożywienia przez mrówki w środowisku naturalnym. Najważniejszymi początkowymi pracami były eksperymenty prowadzone przez Deneubourga [82, 83]. Proces wyszukiwania najkrótszej ścieżki składa się z dwóch etapów: odkładania feromonu (substancji chemicznej) wydzielanego przez mrówki oraz naturalny proces parowania. Podczas wędrówki, w celu znalezienia pożywienia, mrówki odkładają pewną ilość feromonu na swojej drodze. Początkowo wybór kierunku jest czysto losowy.

Po pewnym czasie wzmocnienie feromonu powoduje, że każda kolejna mrówka rozpoczynając wędrówkę podąża za feromonem. Heurystykę wyznaczania najkrótszej trasy przedstawia uproszczony model na rysunku 2.4.



Rysunek 2.4: Znajdowanie najkrótszej trasy przez mrówki. Symbolem B oznaczono źródło pożywienia, A oznacza mrowisko (punkt początkowy i docelowy).

Źródło pożywienia oznaczone jest symbolem B, z kolei kopiec mrówek oznaczono symbolem A. Kiedy dwie mrówki znajdą źródło pożywienia (B), ale podążają inną ścieżką, mrówka, której droga jest krótsza jest w stanie odłożyć większą ilość feromonu w czasie (pokonuje krótszy dystans). Na dłuższej drodze, feromon do czasu wzmocnienia może wyparować całkowicie lub znacznie zredukować swoją wartość. Tym samym, krótsza droga będzie wzmacniana coraz częściej. Większa liczba śladów feromonowych w bliskiej odległości powoduje powstanie tuneli osmotycznych [20]. Jest to nagromadzenie znacznej ilości feromonu na pewnej ograniczonej powierzchni. W tunelu takim, decyzja o wyborze konkretnej ścieżki jest bardziej losowa (rozmyta) z powodu tego, że niektóre fragmenty ścieżki zawierają przybliżoną ilość feromonu. Można to porównać do przeszukiwania lokalnego. Kiedy źródło pożywienia się wysyci, mrówki przestają odkładać w drodze powrotnej feromon. Z czasem ślad prowadzący do wysyconego źródła pożywienia całkowicie wyparuje. Cały proces i jego prostota była inspiracją do przeniesienia tego zjawiska na rzecz algorytmiki pod nazwą metaheurystyki ACO. Najpopularniejszymi wariantami ACO zastosowanymi z sukcesem w wielu problemach optymalizacji dyskretnej są: system mrówkowy AS zaproponowany przez Marco Dorigo w 1992 roku [2, 3], algorytm mrowiskowy ACS zaproponowany przez Marco Dorigo i Luca Maria Gambardella Gambardella w 1997 roku [1] oraz algorytm mrowiskowy $\mathcal{MAX} - \mathcal{MIN}$ opublikowany w roku 2000, a którego autorami są Thomas Stützle i Holger Hoos.

System mrówkowy

W algorytmie mrówkowym (AS), iteracyjnie każdy osobnik przechodzi z jednego wierzchołka do kolejnego, aż utworzy poprawne rozwiązanie. Funkcja przejścia do kolejnego wierzchołka określana jest na podstawie prawdopodobieństwa przejścia. Jest ono w algorytmach mrowiskowych związane z wartością feromonu odłożonego na konkretnej krawędzi. Na początku prawdopodobieństwo to jest zupełnie losowe (wartość inicjalna). Proces wzmocnienia powoduje, że pewne krawędzie skumulują znaczną ilość feromonu, a prawdopodobieństwo będzie odpowiednio większe. Dla pozostałych krawędzi, w wyniku parowania, prawdopodobieństwo wyboru zmaleje lub będzie równe wartości progowej. Jest to typowy przykład uczenia ze wzmocnieniem. Pseudokod 4 przedstawia uproszczony model działania algorytmu.

Algorytm 4: Szablon algorytmu mrowiskowego.

```
1 wejście: Ustaw parametry wejściowe;
2 Zainicjuj macierz feromonową;
3 while Nie spełniono warunku stopu do
4 |   Utwórz rozwiązanie;
5 |   Zastosuj przeszukiwanie lokalne;          // opcjonalnie
6 |   Uaktualnij macierz feromonową;
7 end
8 wyjście: Najlepsze rozwiązanie
```

Uruchomienie algorytmu Algorytm rozpoczyna działanie od utworzenia losowej populacji. Każdy osobnik reprezentuje poprawne rozwiązanie (w przypadku algorytmu TSP – cykl Hamiltona). Następnie zostaje zainicjalizowany feromon (macierz feromonowa). W przypadku ACO wszystkie wartości w macierzy mają wartość inicjalną τ_0. Rozmiar macierzy feromonowej jest taki sam jak macierzy odległości, tj. $n \times n$, gdzie n oznacza liczbę wierzchołków. Następnie populacja jest modyfikowana wraz z kolejnymi iteracjami algorytmu.

Tworzenie rozwiązania Za każdym razem mrówka rozpoczyna tworzenie nowej pozycji od nowa, wybierając kolejno nowy wierzchołek z sąsiedztwa

ostatnio wybranego wierzchołka. Wybór realizowany jest za pomocą prawdopodobieństwa wyboru kolejnego wierzchołka zgodnie ze wzorem (2.14):

$$p_{a,b} = \frac{[\tau_{a,b}]^{\alpha}[\eta_{a,b}]^{\beta}}{\sum_{l \in \mathcal{N}_a}[\tau_{a,l}]^{\alpha}[\eta_{a,l}]^{\beta}}, \quad \text{if } b \in \mathcal{N}_a, \tag{2.14}$$

gdzie:

- a, b są wierzchołkami,
- $\tau_{a,b}$ to wartość feromonu odczytana z macierzy feromonowej dla krawędzi $\langle a, b \rangle$,
- $\eta_{a,b}$ to tzw. informacja heurystyczna, czyli dodatkowa wiedza o problemie, w formie wag przypisywanych krawędziom, najczęściej jako: $\frac{1}{d_{ab}}$ [21], gdzie d_{ab} jest odległością,
- α, β to parametry skalujące wagi kolejno: wartości feromonu i informacji heurystycznej,
- $\mathcal{N}_a$ to sąsiedztwo (najbliższe otoczenie) wierzchołka a.

Wybór kolejnej krawędzi odbywa się określoną metodą. Najczęściej jest to metoda ruletki. Gdy zadany parametr β jest równy zero, informacja na temat długości krawędzi nie jest brana pod uwagę. Tym samym decydującym parametrem odpowiedzialnym za wybór kolejnej pozycji jest feromon. Gdy obie wartości α, β są różne od zera, w pierwszej iteracji prawdopodobieństwo przejścia do następnej pozycji jest zależne jedynie od informacji heurystycznej ze względu na identyczne wartości w macierzy feromonowej. Dopiero w kolejnych iteracjach wraz ze wzmocnieniem niektórych krawędzi wartość feromonu nabiera znaczenia.

Uaktualnienie macierzy feromonowej Aktualizacja macierzy feromonowej wykonywana jest dla każdej mrówki, po utworzeniu wszystkich rozwiązań. Składa się z dwóch etapów: parowania i wzmocnienia. Formalnie opisuje je wzór (2.15):

$$\tau_{a,b} = (1 - \rho)\tau_{a,b} + \Delta\tau_{a,b}, \tag{2.15}$$

gdzie:

- ρ oznacza tempo parowania feromonu (wartość z przedziału $[0, 1]$),
- $\Delta\tau_{a,b}$ wartość wzmocnienia, najczęściej [21]:

$$\Delta\tau_{a,b} = \begin{cases} \frac{Q}{L} & \text{jeżeli } \langle a, b\rangle \text{ znajduje się rozwiązaniu mrówki}, \\ 0 & \text{w przeciwnym przypadku}, \end{cases} \tag{2.16}$$

  gdzie:

  - L to najczęściej długość krawędzi (waga) postaci d_{ab},
  - Q to stała, najczęściej wartość 1.

Parowanie feromonu ma na celu zmniejszenie prawdopodobieństwa wyboru krawędziom, które dają słabsze rezultaty. Wzmocnienie ma na celu zwiększenie prawdopodobieństwa wyboru dla krawędzi, które zostały wybrane przez mrówki.

Algorytm MMAS

Algorytm $\mathcal{MAX} - \mathcal{MIN}$ jest techniką zaproponowaną przez Thomasa Stützle i Holgera Hoosa na podstawie algorytmu ACO Marco Dorigo [3, 9]. Z powodzeniem został zaimplementowany w wielu kombinatorycznych problemach optymalizacyjnych należących do klasy NP. W stosunku do pierwowzoru poczyniono wiele zmian.

- Algorytm dynamicznie (w czasie działania) ogranicza wartość feromonu do przedziału $[\tau_{\min}, \tau_{\max}]$. Jeśli ilość feromonu będzie przekraczać wartość brzegową, zostanie ona skorygowana do najbliższej wartości brzegowej.
- Wartość inicjalna w macierzy feromonowej wynosi $\tau_{\max}$ i wraz z kolejnymi iteracjami maleje.
- W MMAS tylko jedna mrówka zostawia ślad (wzmocnienie feromonowe). W zależności od przyjętej strategii jest to najlepsza mrówka w iteracji lub najlepsza mrówka w całym algorytmie.

Dodatkowo w algorytmie zastosowano "wygładzanie" wartości feromonu, które uśrednia jego wartość. Autor [9] pokazał, że taki zabieg daje bardzo dobre rezultaty. W klasycznym algorytmie ACO nie zastosowano ograniczenia wartości feromonu.

Stagnacja Może się zdarzyć, że mrówki mogą utknąć w optimum lokalnym. Dzieje się tak, gdy wartość feromonu dla malej liczby krawędzi jest na tyle duża, że wciąż wybierane są te same krawędzie. Tym samym każda nowa pozycja jest taka sama. Jest to zjawisko bardzo niepożądane, gdyż prowadzi do przedwczesnej stagnacji. Rozwiązaniem zastosowanym w MMAS jest przypisanie początkowej wartości feromonu na $\tau_{\max}$ oraz ustalenie wartości $\tau_{\min}$ blisko wartości $\tau_{\max}$. W kolejnych iteracjach algorytmu różnica ta jest zwiększana. Dzięki temu żadna z krawędzi nie osiągnie znaczącego wzrostu feromonu w stosunku do pozostałych krawędzi w początkowej fazie działania algorytmu. Kolejną metodą unikania stagnacji jest odkładanie feromonu tylko przez jednego osobnika. W zależności od wybranej strategii może to być najlepszy osobnik dotychczas znaleziony lub najlepszy w bieżącej iteracji. Innym mechanizmem umożliwiającym ograniczeniem stagnacji jest reset feromonu. Gdy algorytm przestaje generować nowe trasy, reset powoduje wybicie populacji z optimum lokalnego [9]. Zwykle odbywa się to po pewnej liczbie iteracji.

Rozdział 3

Dyskretny algorytm PSO z pamięcią feromonową

W rozdziale tym zostanie omówiony zaproponowany w pracy dyskretny algorytm PSO z pamięcią feromonową. W podrozdziale 3.1 zostanie przedstawione uzasadnienie stworzenia tego algorytmu, a następnie opisane podstawowe pojęcia z nim związane. Ważnym fragmentem rozdziału, który zostanie opisany w podrozdziale 3.2 stanowią operacje, które definiują sposób otrzymywania kolejnych rozwiązań problemu. W dalszej części zostaną przestawione przykłady działania algorytmu oraz krok po kroku zademonstrowany proces przebiegu optymalizacji. W podrozdziale 3.3 omówiona zostanie rola feromonu w procesie adaptacji do problemu DTSP. W podrozdziale 3.4 zostanie wykazana efektywność omawianego rozwiązania.

3.1 Motywacja

Nowa wersja dyskretnego algorytmu PSO z pamięcią feromonową bazuje na pracy Zhonga i in. [13]. Opis tego algorytmu można znaleźć w rozdziale 2. Asumptem do powstania nowej wersji jest brak korelacji między prawdopodobieństwem wyboru krawędzi wchodzących w skład kolejnego rozwiązania, a wpływem na sumaryczną długość znalezionego rozwiązania w pierwotnym algorytmie. Każdorazowy wybór jest losowy i sterowany jedynie przez stałe w czasie działania algorytmu współczynniki c_1, c_2, c_3, ω oraz losową wartość z przedziału $[0, 1]$ o rozkładzie jednorodnym. W celu wzmocnienia prawdopodobieństwa wyboru krawędziom algorytmu, które poprawiają (skracają)

długość rozwiązania zastosowano wirtualny feromon, którego przeznaczenie w algorytmach inspirowanych zachowaniem mrówek w naturalnym środowisku zostało omówione w rozdziale 2. Modyfikacja ta poprawia również adaptacyjność algorytmu w kontekście dynamicznego problemu komiwojażera, co zostało pokazane w wynikach badań przedstawionych w dalszej części pracy.

3.2 Opis algorytmu

Równanie opisujące ruch cząsteczki w algorytmie DPSO z feromonem przyjmuje postać daną wzorem (3.1) i (3.2):

$$\begin{aligned} V_i^{k+1} &= c_2 rand() \cdot (gBest \setminus X_i^k) \\ &\cup c_1 rand() \cdot (pBest_i \setminus X_i^k) \\ &\cup \omega \cdot V_i^k \,, \end{aligned} \tag{3.1}$$

$$X_i^{k+1} = \underbrace{\Delta\tau(V_i^{k+1})}_{\text{krok 3.2a}} \oplus \underbrace{c_3 rand() \cdot X_i^k}_{\text{krok 3.2b}} , \tag{3.2}$$

gdzie:

- krawędź algorytmu $\hat{e}$ jest zgodna z definicją we wzorze (2.7),
- prędkość cząsteczki i w iteracji k (V_i^k) to kierunek przeszukiwania przestrzeni rozwiązań. W dyskretnej wersji jest to zbiór krawędzi,
- pozycja cząsteczki X_i^k to zbiór krawędzi algorytmu tworzących cykl Hamiltona, w TSP jedno z rozwiązań problemu,
- operacja iloczynu wartości skalarnej i zbioru krawędzi przeprowadzona jest zgodnie ze wzorami (2.12) i (2.11).

Feromon został do algorytmu dołączony poprzez funkcję $\Delta\tau$, która zmienia prawdopodobieństwo przejścia do następnej pozycji każdemu elementowi zbioru prędkości, wykorzystując pamięć feromonową znaną z algorytmów mrowiskowych. Funkcja wzmocnienia dowolnego zbioru krawędzi M algorytmu określona jest wzorem (3.3):

$$\Delta\tau(M) = \forall_{\langle p,\langle a,b\rangle\rangle \in M} \, \Delta\tau_p(\langle p, \langle a, b\rangle\rangle) \,, \tag{3.3}$$

gdzie:

- M oznacza dowolny zbiór krawędzi algorytmu postaci (2.7),
- $\Delta\tau_p$ to funkcja zmieniająca wartość parametru p dla każdej krawędzi w M.

Funkcja $\Delta\tau_p$, która wiąże bieżącą wartość parametru p i wartość feromonu określona jest wzorem (3.4):

$$\Delta\tau_p(\langle p, \langle a, b\rangle\rangle) = \langle p + \Big((\tau_{a,b} - 0,5) \cdot \frac{k}{k_{max}}\Big), \langle a, b\rangle\rangle\,, \tag{3.4}$$

gdzie:

- $\langle a, b\rangle \in \mathcal{E}$ to krawędź problemu TSP,
- $\tau_{a,b}$ to wartość feromonu dla krawędzi o końcach a, b z przedziału $[\tau_{\min}, \tau_{\max}]$. Macierz feromonowa agreguje wszystkie wartości feromonu na każdej dozwolonej krawędzi i przyjmuje postać:

$$\tau = \begin{pmatrix} - & \tau_{1,2} & \cdots & \tau_{1,n} \\ \tau_{2,1} & - & \cdots & \tau_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{n,1} & \tau_{n,2} & \cdots & - \end{pmatrix}, \tag{3.5}$$

- współczynnik $\frac{k}{k_{max}}$ skaluje moc wzmocnienia (dodatniego lub ujemnego) w zależności od liczby wykonanych iteracji algorytmu.

Operacja wzmocnienia polega na modyfikacji współczynnika p krawędzi algorytmu $\hat{e}$. Wynikiem działania funkcji (3.3) jest zbiór krawędzi ze wzmocnionym dodatnio lub ujemnie współczynnikiem p (prawdopodobieństwem dodania krawędzi do kolejnego zbioru pozycji cząsteczki, czyli kolejnego rozwiązania problemu). Przy założeniu, że $\tau_{\min}$ jest równe zero, a $\tau_{\max}$ wynosi jeden, wartość wzmocnienia jest z przedziału $[-0,5, 0,5]$ (takie wartości przyjęto w pracy). Feromon o ujemnej wartości (repelent) jest interpretowany jako funkcja kary dla krawędzi, które nie poprawiają jakości otrzymanego rozwiązania. W ten sposób algorytm przekazuje do kolejnej pozycji cząsteczki tylko te krawędzie, które mają dużą szansę należeć do rozwiązania optymalnego. Niepewność z tym związana ma swoje odzwierciedlenie w wartości feromonu. Im większa jest to wartość, tym większe jest prawdopodobieństwo, że dana

krawędź należy do rozwiązania optymalnego. W kolejnych iteracjach algorytmu, poprzez parowanie i wzmacnianie feromonu wartość ta jest modyfikowana. Krawędzie często wchodzące w skład najlepszego rozwiązania będą miały dużą wartość feromonu i tym samym dodatnią wartość wzmocnienia feromonowego (wzór (3.3)). Po uwzględnieniu wszystkich operacji, wartość prawdopodobieństwa wyboru krawędzi do następnej pozycji $k+1$ dane jest wzorem (3.6):

$$p = \underbrace{c \cdot rand()}_{\text{wzór (3.1)}} + \underbrace{\left((\tau_{a,b} - 0,5) \cdot \frac{k}{k_{max}}\right)}_{\text{wzór (3.4)}}, \tag{3.6}$$

gdzie: c to stały modyfikator w czasie działania algorytmu. W zależności od tego czy wzmacniana krawędź pochodzi ze zbioru: $pBest_i$, $gBest$, V_i^k przyjmuje wartość kolejno: c_1, c_2 lub ω.
Domyślna wartość współczynnika p wynosi 1 [84, 13]. Przykładowo, poprawne rozwiązanie symetrycznego problemu TSP:

$$\{\{1,2\},\{2,3\},\{3,4\},\{4,1\}\},$$

w algorytmie DPSO będzie miało postać:

$$\{\langle 1,\{1,2\}\rangle, \langle 1,\{2,3\}\rangle, \langle 1,\{3,4\}\rangle, \langle 1,\{4,1\}\rangle\}.$$

Rysunek 3.1 przedstawia proces tworzenia kolejnej pozycji cząsteczki w algorytmie DPSO z feromonem (kolejnego cyklu Hamiltona). Wszystkie zmienne oznaczają zbiory krawędzi. W pierwszym kroku ($k = 0$) wykonywana jest operacja odejmowania dwóch zbiorów: $gBest$ i aktualnej pozycji X_i^k (jedno z rozwiązań problemu). Wynikiem tej operacji są krawędzie, dzięki którym rozwiązanie oznaczone jako $gBest$ jest lepsze od aktualnego rozwiązania X_i^k. Ta sama czynność realizowana jest dla zbioru $pBest_i$ (najlepsze rozwiązanie znalezione przez aktualnie rozpatrywaną cząsteczkę i). Zbiór $pBest_i$ w pierwszym kroku jest równy pozycji X_i^k. Następnie do wyniku obu działań dodawana jest prędkość (V_i^k), która w pierwszej iteracji ($k = 0$) jest zbiorem pustym. Wynikowy zbiór jest wzmacniany przez feromon – do każdego współczynnika p krawędzi dodawana jest wartość feromonu. Następnie wynikowy zbiór jest filtrowany na podstawie parametru p każdej krawędzi, która się w nim zawiera. Obie operacje zostały opisane w dalszej części pracy. Po tym etapie powstaje kolejna prędkość cząsteczki (V_i^{k+1}). Zbiór ten zwykle

nie tworzy cyklu Hamiltona (rozwiązania problemu TSP). Zadaniem kolejnych kroków jest dopełnienie zbioru, tak aby stanowił on kolejną pozycję cząsteczki (X_i^{k+1}). Następnie każdemu współczynnikowi p krawędzi znajdującym się w tym zbiorze, zostanie przypisana wartość 1. Powyższe operacje wykonywane są dla każdej cząsteczki w stadzie.

Faza inicjalna

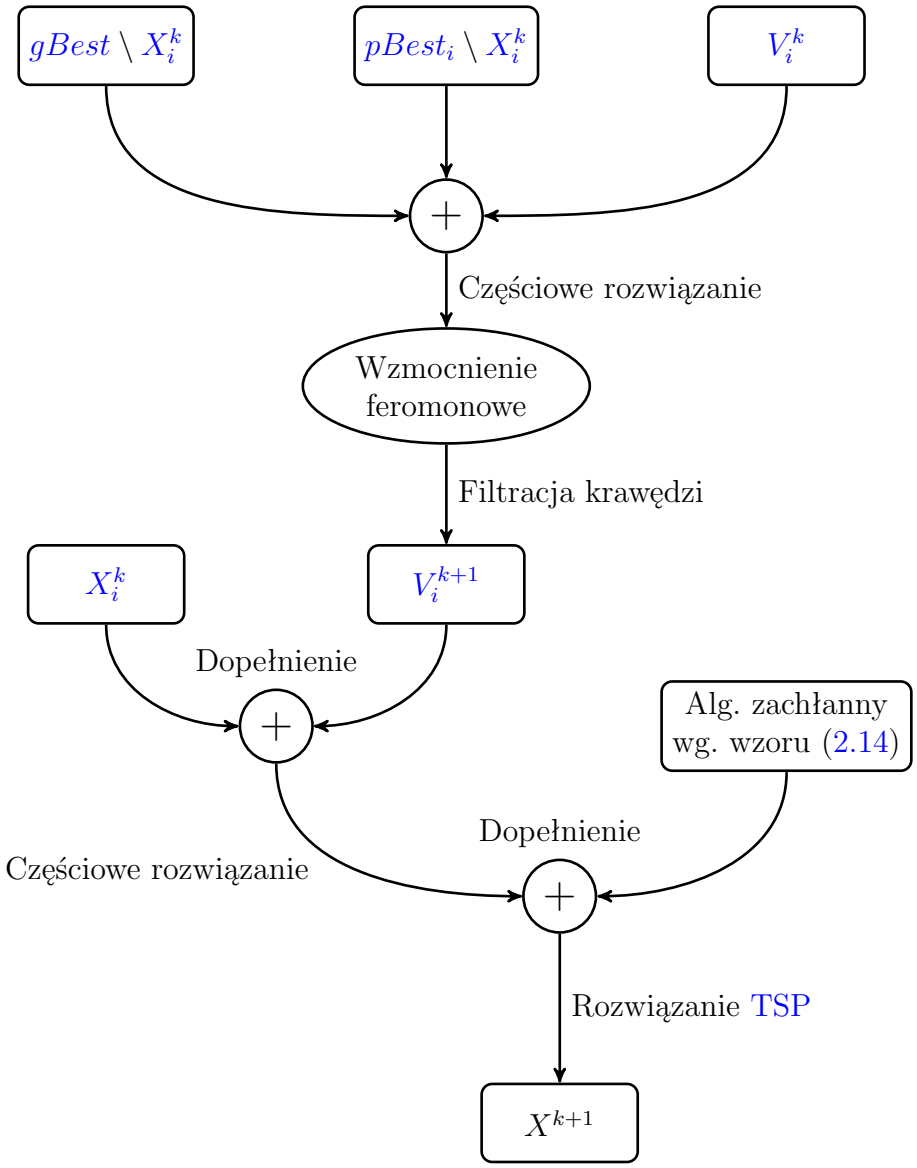
Na początku działania algorytmu każdy osobnik reprezentuje losowe rozwiązanie. Najlepsze rozwiązanie zostaje oznaczone jako $gBest$ i w algorytmie reprezentuje najlepsze znalezione rozwiązanie przez stado. W trakcie działania algorytmu zbiór ten jest uaktualniany. Zbiór $pBest_i$ oznacza najlepsze znalezione rozwiązanie przez cząsteczkę. W pierwszej iteracji (k=0) jest równy pozycji cząsteczki. Każdy osobnik posiada swój zbiór $pBest_i$ w przeciwieństwie do zbioru $gBest$, który jest współdzielony przez całe stado. Pseudokod 5 przedstawia tworzenie losowego stada przez algorytm.

Faza optymalizacji

Algorytm 6 w sposób bardziej szczegółowy prezentuje działanie DPSO z feromonem, uwzględniając ruch całego stada cząsteczek, aktualizacje zbiorów $pBest_i$ i $gBest$ oraz uaktualnienie macierzy feromonowej.

Filtracja krawędzi

Po wyznaczeniu kolejnego zbioru prędkości V_i^{k+1}, w procesie optymalizacji następuje wzmocnienie prawdopodobieństwa wyboru p każdej krawędzi $\hat{e}$ zgodnie ze wzorem (3.3). Kolejnym etapem jest filtracja krawędzi algorytmu ze zbioru V_i^{k+1}. Zmieniona wartość feromonu odłożona na krawędzi (współczynnik p) jest wykorzystywana na tym etapie, ma ocenić jakość krawędzi (jak duże jest prawdopodobieństwo, że krawędź należy do rozwiązania optymalnego). Proces przebiega poprzez porównanie każdego współczynnika p krawędzi w zbiorze z wartością losową r z przedziału $[0, 1]$. Jeśli $r \leqslant p$ to wówczas krawędź pozostaje w zbiorze. W przeciwnym przypadku, krawędź jest z niego usuwana. Celem etapu filtracji jest ekstrakcja krawędzi z największym prawdopodobieństwem, że krawędź należy do rozwiązania optymalnego oraz wprowadzanie losowości do rozwiązania (co zapobiega utykaniu



Rysunek 3.1: Tworzenie nowej pozycji cząsteczki i w homogenicznym algorytmie DPSO.

w optimum lokalnym).

Algorytm 5: Tworzenie stada w algorytmie DPSO z feromonem.

```
1  for i = 0 → i_max do
2      Wybierz losowy wierzchołek a ∈ 𝒱;
3      Ustaw X_i^0 = ∅;
4      while |X_i^0| ⩽ |𝒱| do
5          forall b ∈ 𝒩_a do
6              if deg(b) < 2 then
7                  X_i^0 ∪ ⟨1, ⟨a, b⟩⟩;
8                  Zakończ pętlę;
9              end
10         end
11         if Nie znaleziono żadnego wierzchołka w 𝒩_a then
12             Przypisz dowolny wierzchołek z 𝒱 do b;
13             X_i^0 ∪ ⟨1, ⟨a, b⟩⟩;
14         end
15         a = b;
16     end
17     pBest_i = X_i^0;
18 end
19 Uaktualnij gBest;
```

Dopełnienie zbioru do cyklu Hamiltona

Po etapie filtracji następuje etap dopełnienia, którego celem jest dodanie brakujących krawędzi algorytmu do pełnego cyklu Hamiltona. W początkowej fazie do niepełnego zbioru pozycji cząsteczki dodawane są krawędzie z poprzedniego zbioru pozycji, zgodnie ze wzorem (3.2b). W przypadku nie utworzenia cyklu Hamiltona następuje drugi etap – dopełnienia. Oryginalny algorytm zaproponowany przez Zhonga i innych [13] dodaje krawędzie na podstawie heurystyki najbliższego sąsiada i miary euklidesowej. Proponowane w pracy rozwiązanie stosuje dwie techniki. Pierwszą z nich jest heurystyka najbliższego sąsiada oparta na α-mierze [7, 8]. Drugą z nich jest funkcja przejścia znana z algorytmów mrowiskowych [9] (w celu maksymalizacji wpływu feromonu na algorytm). Po etapie filtracji tworzona jest lista stopni wierzchołków. Każdy wierzchołek w cyklu Hamiltona jest stopnia drugiego. Obie techniki operując na tej liście, łączą wierzchołki, których stopień

Algorytm 6: Algorytm DPSO z pamięcią feromonową.

wejście: liczba iteracji k_{max},
rozmiar stada i_{max};
Utwórz losowe stado (Alg. 5);
for $k = 0 \rightarrow k_{max}$ **do**
  for $i = 0 \rightarrow i_{max}$ **do**
    Oblicz prędkość V_i^{k+1} (wzór (3.1));
    Filtrowanie krawędzi;
    $X_i^{k+1} = \emptyset$;
    forall $\langle p, \langle a, b\rangle\rangle \in V_i^{k+1}$ **do** `// wzór (3.2a)`
      Zastosuj wzmocnienie feromonowe p (wzór (3.3));
      if $rand() \leqslant p$ **then**
        $X_i^{k+1} \cup \langle p, \langle a, b\rangle\rangle$;
      end
    end
    forall $\langle p, \langle a, b\rangle\rangle \in X_i^k$ **do** `// wzór (3.2b)`
      if $rand() \leqslant rand() \cdot c_3$ **then**
        $X_i^{k+1} \cup \langle p, \langle a, b\rangle\rangle$;
      end
    end
    Etap dopełnienia;
    Dopełnij X_i^{k+1} używając sąsiedztwa;
    Uaktualnij $pBest_i$ i $gBest$;
  end
  Uaktualnij macierz feromonową;
end
wyjście: $gBest$

jest mniejszy niż 2. W przypadku heurystyki najbliższego sąsiada wybierany jest wierzchołek, którego odległość od rozpatrywanego wierzchołka jest najmniejsza. W drugim przypadku stosuje się funkcję przejścia znaną z algorytmu ACO i MMAS (2.14) oraz metodę koła ruletki. Zastosowanie obu metod rozszerza przegląd przestrzeni rozwiązań. Stosowanie tylko pierwszej metody powoduje, że istnieje bardzo wysokie prawdopodobieństwo wyboru tych samych sąsiednich wierzchołków. Zastosowanie tylko drugiej metody powoduje, że informacja o najbliższym sąsiedzie nie jest wykorzystywana. Obie

metody dopełnienia zbioru krawędzi stosuje się wg zasady – na każde 50 iteracji dopełnienia funkcją przejścia, wykonywana jest jedna iteracja heurystyki najbliższego sąsiada. Po utworzeniu pełnego cyklu Hamiltona, wszystkie wartości prawdopodobieństwa wyboru krawędzi p są resetowane do wartości inicjalnej równej jeden.

Aktualizacja macierzy feromonowej w DPSO

Proces aktualizacji macierzy feromonowej jest następujący: w pierwszej iteracji algorytmu, każdemu elementowi macierzy przypisywana jest wartość domyślna $\tau_{\max}$. Jest to jednocześnie wartość początkowa. Pod koniec każdej iteracji algorytmu k, najpierw odparowywany jest feromon poprzez pomnożenie macierzy feromonowej przez współczynnik $\rho < 1$, następnie wzmocnione zostają te krawędzie, które znajdują się w najlepszym znalezionym rozwiązaniu. Wartości przechowywane w macierzy są z przedziału $[\tau_{\min}, \tau_{\max}]$. Proces aktualizacji feromonu jest tożsamy z algorytmem MMAS, opracowanym przez Thomasa Stützle and Holgera Hoosa [9].

3.3 Rola feromonu w DTSP

W kontekście dynamicznego problemu komiwojażera ważną cechą pamięci feromonowej jest jej samoadaptacja. W przypadku, gdy w początkowych iteracjach algorytmu, dana krawędź często będzie składową najlepszego rozwiązania, wartość w macierzy feromonowej dla tej krawędzi będzie stała i będzie wynosić $\tau_{\max}$. Jednocześnie, gdy w pewnym momencie przestanie ona wchodzić w skład najlepszego rozwiązania, jej wartość w macierzy feromonowej zacznie spadać, aż w końcu osiągnie wartość minimalną $\tau_{\min}$. Po każdej zmianie danych (w kolejnym podproblemie), macierz feromonowa jest kopiowana z poprzedniego uruchomienia algorytmu. Jest to kolejny parametr wejściowy algorytmu DPSO z feromonem, który przekazywany jest wraz z innymi parametrami, takimi jak macierz odległości dla podproblemu. W dynamicznym problemie TSP można wyróżnić sekwencję macierzy odległości oraz w przypadku omawianego algorytmu DPSO, sekwencję macierzy feromonowej. Można ją interpretować jako miarę jakości krawędzi zależną od macierzy odległości. W ostatniej iteracji algorytmu dla podproblemu $\mathcal{I}^t$ macierz ta przechowuje informacje o przestrzeni rozwiązań tego podproblemu. Mała wartość feromonu odłożona na krawędzi może świadczyć o małym prawdo-

podobieństwie, że krawędź należy do rozwiązania optymalnego. W idealnym przypadku krawędzie należące do rozwiązania optymalnego powinny zawierać maksymalne wartości w macierzy feromonowej. Taka sytuacja ma miejsce jedynie wtedy, gdy algorytm wykorzystuje technikę uczenia ze wzmocnieniem (algorytmy mrowiskowe) oraz znajdzie rozwiązanie optymalne odpowiednio długo przed zakończeniem ostatniej iteracji algorytmu (czas na wzmocnienie krawędzi należących do rozwiązania optymalnego). Sytuacja taka jest dość rzadka, z tego powodu pamięć feromonową należy traktować z pewną miarą niepewności. Poniżej znajduje się schemat przedstawiający macierz feromonową w problemie DTSP dla trzech pierwszych podproblemów:

$$\overset{t=0}{\begin{pmatrix} - & \tau_{\max} & \cdots & \tau_{\max} \\ \tau_{\max} & - & \cdots & \tau_{\max} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{\max} & \tau_{\max} & \cdots & - \end{pmatrix}} \rightarrow \overset{t=1}{\begin{pmatrix} - & \tau_{1,2} & \cdots & \tau_{1,n} \\ \tau_{2,1} & - & \cdots & \tau_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{n,1} & \tau_{n,2} & \cdots & - \end{pmatrix}} \rightarrow \overset{t=2}{\begin{pmatrix} - & \tau_{1,2} & \cdots & \tau_{1,n} \\ \tau_{2,1} & - & \cdots & \tau_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{n,1} & \tau_{n,2} & \cdots & - \end{pmatrix}}.$$

3.4 Efektywność algorytmu

Przedstawiony w pracy algorytm DPSO, wzbogacony o wirtualny feromon posiada cechy samoadaptacji, który ma na celu bardziej efektywne rozwiązywanie kolejnych podproblemów. W każdym kolejnym podproblemie, macierz odległości ulega zmianie tylko częściowo. Proponowane rozwiązanie wykorzystuje informację o rozwiązaniach w podproblemie t do optymalizacji, po modyfikacji macierzy odległości, w podproblemie $t+1$. Dzięki temu algorytm efektywniej przeszukuje przestrzeń rozwiązań. Dodatkowo macierz feromonowa otrzymana w poprzednim podproblemie jest kopiowana, co ma poprawić zbieżność algorytmu do optimum w kolejnych podproblemach.

W badaniach stosowano ustawienia znajdujące się w tabeli 3.1. Każdy podproblem zawierał 3% zmodyfikowanych współrzędnych wierzchołków (miast). Wyjątkiem są badania porównawcze między różnymi wariantami algorytmu DPSO, gdzie wzięto pod uwagę większą liczbę modyfikacji. Parametry dla wersji bez feromonu ustawiono następująco: $c_1 = 1,5$; $c_2 = 2$; $c_3 = 2$, $\omega = 0,6$. Są to parametry zaproponowane w pracy [13]. Wersja z feromonem uruchomiona została z parametrami: $c_1 = 0,5$; $c_2 = 0,5$; $c_3 = 0,5$. Wartość ω to: *berlin52* – 0,2 dla *gr202* i *kroA100* – 0,5. Wartości te zostały

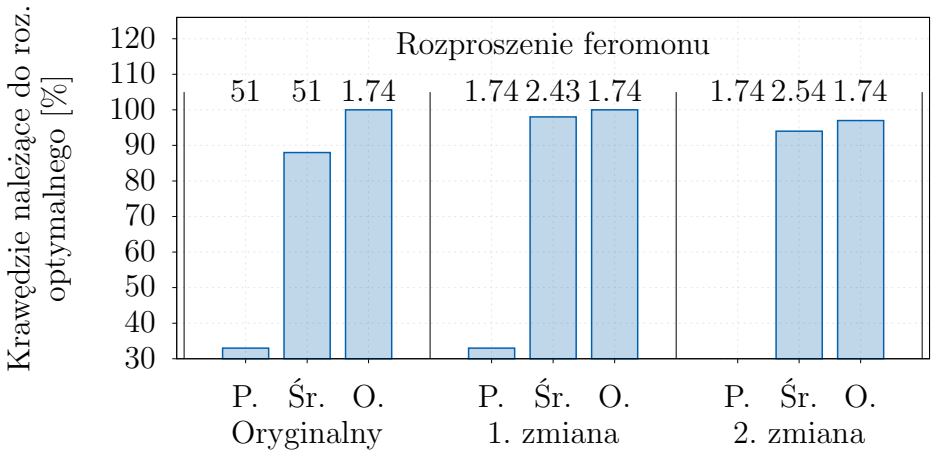
wyznaczone poprzez próbkowanie różnych wartości tych parametrów.

Tabela 3.1: Ustawienia algorytmu DPSO z feromonem.

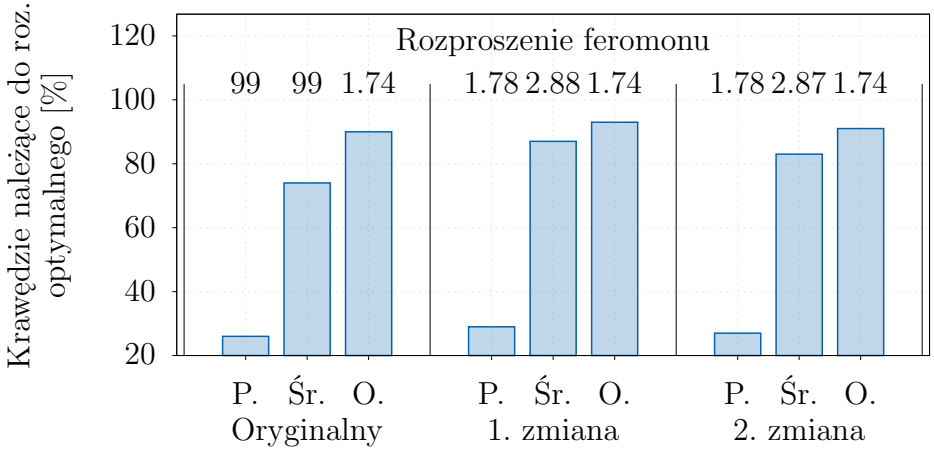
Problemy	Parametry					
	c_1	c_2	c_3	ω	i_{max}	$\mathcal{N}_{size}$
berlin52	0,5	0,5	0,5	0,2	32	7
kroA100	0,5	0,5	0,5	0,6	60	10

Rozproszenie feromonu

Samoadaptacyjny charakter algorytmu można zaobserwować na Rys. 3.2 i Rys. 3.3. Badania powstały na podstawie obliczeń dla problemów *berlin52* i *kroA100*. Oś y przedstawia procentowy stosunek wspólnych krawędzi z rozwiązaniem optymalnym do liczby wszystkich krawędzi należących do rozwiązania optymalnego, który jest stały i wynosi $|\mathcal{V}|$ (liczba krawędzi w rozwiązaniu). Jeśli stosunek ten wynosi 100%, znaleziona trasa jest rozwiązaniem optymalnym. Pionową czarną linią oddzielono kolejne podproblemy (zmiany). Kolejnymi punktami na osi odciętych oznaczono miejsca charakterystyczne dla algorytmu – pierwszą (P.) i ostatnią (Os.) iterację. Wartość „Śr." oznacza agregację wszystkich iteracji algorytmu. Wysokość słupka oznacza procent krawędzi należących do rozwiązania optymalnego w stosunku do liczby krawędzi w rozwiązaniu. Wartości nad słupkami to suma feromonu na krawędziach, których ilość jest większa, niż $\tau_{\min}$ w stosunku do iloczynu $|\mathcal{V}|\cdot\tau_{\max}$. Jeśli stosunek ten wynosi jeden, cały feromon skupiony jest na krawędziach należących do rozwiązania optymalnego. W praktyce trudno osiągnąć ten wynik, co jest związane z różnorodnością w populacji.
Na Rys. 3.2 (problem *berlin52*) i Rys. 3.3 (problem *kroA100*) można również zaobserwować rozproszenie feromonu. W ostatnich iteracjach znalezione zostało rozwiązanie optymalne. Dotyczy to wszystkich trzech podproblemów. W pierwszym podproblemie feromon był bardzo rozproszony po całej macierzy. Wynika to z faktu, że algorytm rozpoczął swoje działanie bez informacji o poprzednich rozwiązaniach (było to jego pierwsze uruchomienie). W kolejnych podproblemach, współczynnik rozproszenia na początku jest mniejszy (feromon jest bardziej skupiony). Niemniej jednak po pierwszej iteracji jego wielkość wzrasta, gdyż algorytm znajduje coraz lepsze rozwiązania. Tym samym feromon dostraja się do nowych danych (po zmianie). W ostatnich



Rysunek 3.2: Adaptacja feromonu w algorytmie DPSO w kolejnych podproblemach na przykładzie problemu *berlin52*.

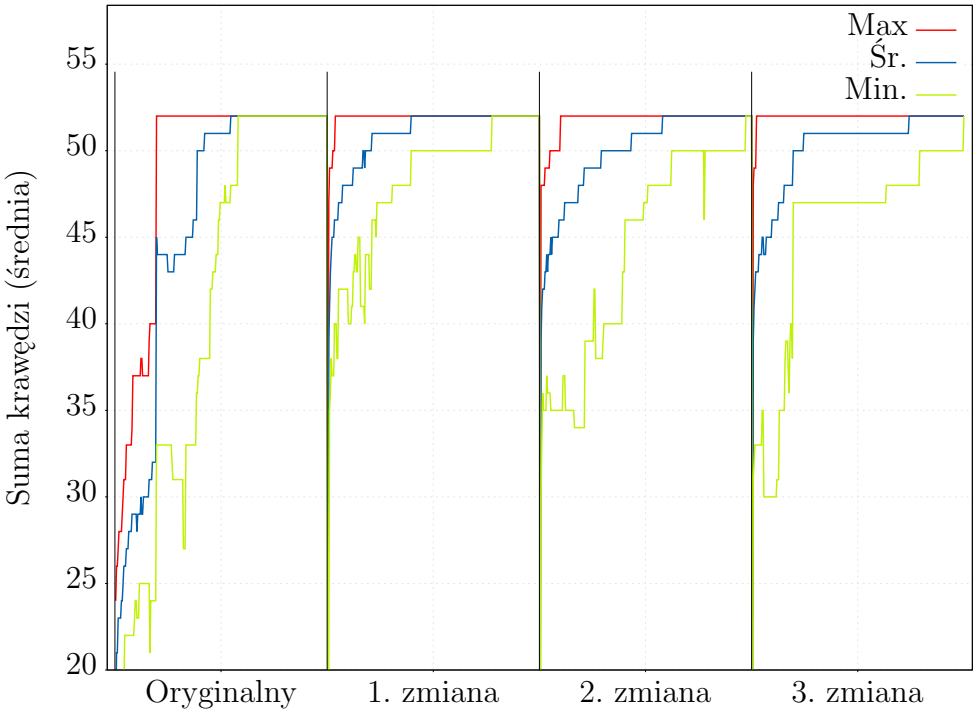


Rysunek 3.3: Adaptacja feromonu w algorytmie DPSO w kolejnych podproblemach na przykładzie problemu *kroA100*.

iteracjach, wartość feromonu ponownie rośnie na pojedynczych krawędziach co jest spowodowane znalezieniem trasy optymalnej przez algorytm.

Podobieństwo populacji

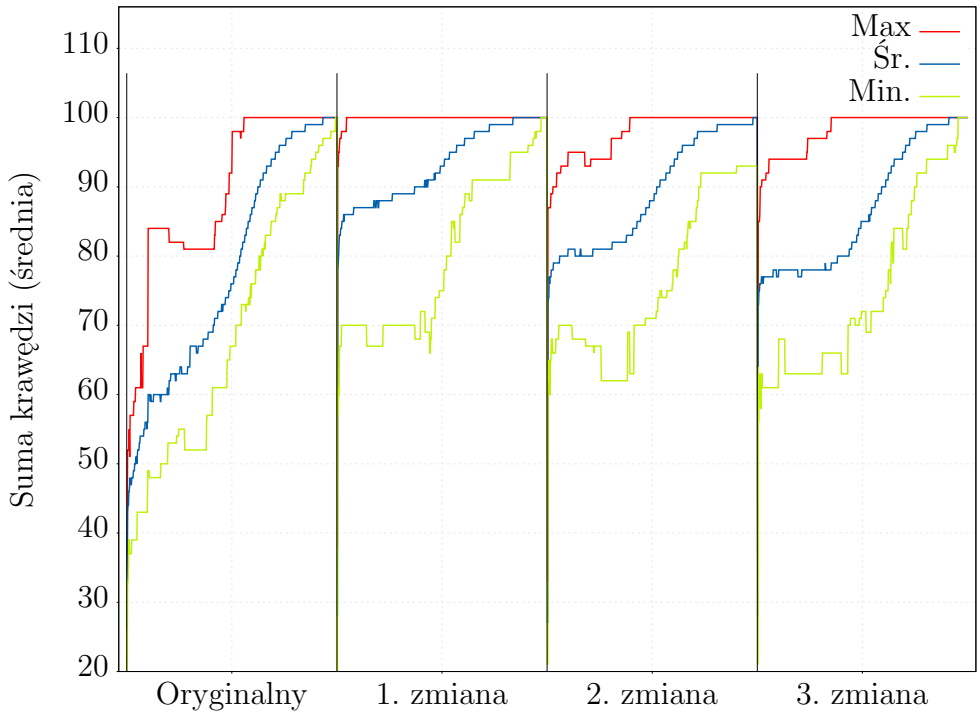
Na kolejnym rysunku Rys. 3.4 można zaobserwować średnią liczność przecięcia zbioru *gBest* ze zbiorem aktualnej pozycji cząsteczki. Na wykresie uwzględniono również minimalną i maksymalną liczbę wspólnych krawędzi. Wykres otrzymano na podstawie wyników otrzymanych z algorytmu DPSO z feromonem.



Rysunek 3.4: Liczność przecięcia zbioru *gBest* i pozycji X_i^k dla problemu *berlin52*.

W początkowych iteracjach algorytmu (Rys. 3.4), liczba wspólnych krawędzi zbioru aktualnej pozycji X_i^k ze zbiorem *gBest* wynosi około 40% (nominalnie 20). Wartość ta sukcesywnie rośnie wraz z postępującym procesem optymalizacji – znajdowaniem coraz lepszych rozwiązań. W połowie zakładanej liczby iteracji średnia przecięć wynosi już 100%, co może oznaczać, że algorytm znalazł rozwiązanie optymalne i rozproszenie populacji na przestrzeni rozwiązań maleje. Po zmianie danych w kolejnym podproblemie i uruchomieniu proce-

su optymalizacji dla nowego podproblemu średnia liczba wspólnych krawędzi zbioru aktualnej pozycji X_i^k i $gBest$ rośnie znacznie szybciej. Można to interpretować w ten sposób, że w kolejnych iteracjach za sprawą kopiowania macierzy feromonowej już na początku algorytm otrzymuje bardzo dobre rozwiązania (relatywnie bliskie optimum). W dwóch następnych podproblemach tendencja jest podobna. Kolejny rysunek Rys. 3.5 przedstawia te same zależności z tą różnicą, że algorytm DPSO z feromonem uruchomiono dla problemu *kroA100*.



Rysunek 3.5: Liczność przecięcia zbioru $gBest$ i pozycji X_i^k dla problemu *kroA100*.

Podobne wnioski jak w przypadku Rys. 3.4 można wyciągnąć analizując Rys. 3.5. Z tą różnicą, że algorytm do ostatniej iteracji charakteryzował się zróżnicowaną populacją. Taką interpretację można wywnioskować z braku pokrycia liczności przecięcia $gBest$ i X_i^k maksymalnego, minimalnego z wartością średnią.

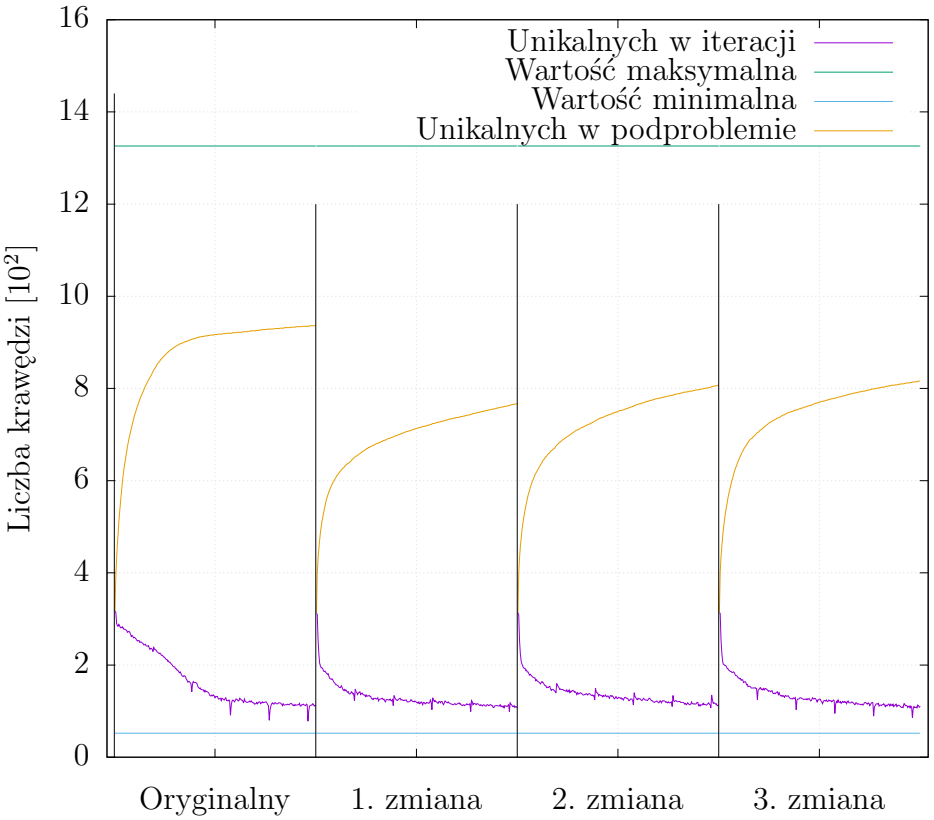
Eksploracja przestrzeni rozwiązań

Wynikiem kolejnych badań są wykresy przedstawione na Rys. 3.6 i Rys. 3.7. Przedstawiają one eksplorację przestrzeni rozwiązań na przykładzie sumarycznej liczby unikalnych krawędzi, znajdujących się w zbiorze pozycji wszystkich cząsteczek w stadzie. Szarą linią oznaczono krawędzie unikalne w danej iteracji, kolorem pomarańczowym liczbę wszystkich unikalnych krawędzi w każdej dotąd uruchomionej iteracji (w ramach jednego podproblemu). Minimalna i maksymalna liczba na wykresie oznacza pojemność populacji, tzn. unikalność osobników w populacji (różnorodność przechowywanych rozwiązań przez stado). Minimalna wartość bierze się z założenia, że każdy osobnik w populacji jest taki sam (liczba unikalnych krawędzi w populacji jest równa rozmiarowi problemu). Maksymalna wartość bierze się z założenia, że każdy osobnik zawiera w swojej bieżącej pozycji (X_i^k) unikalne, w obrębie populacji krawędzie; przy założeniu że pozwala na to rozmiar populacji (i_{max}) w stosunku do rozmiaru problemu (n). Przykładowo dla problemu *berlin52* istnieje tylko $52^2 - 52$ (macierz $n \times n$ minus pętle) unikalnych krawędzi dla problemu asymetrycznego. Maksymalny rozmiar stada, w którym każda cząsteczka miałaby unikalne krawędzie w swojej pozycji wynosi $(52^2 - 52)/52$, czyli 51. Badania przeprowadzono dla dwóch problemów *berlin52* i *kroA100*.

Największa eksploracja następuje w początkowych iteracjach algorytmu. Po czym jej tempo zmian zaczyna spadać. Tę zależność można zaobserwować dla obu problemów i dla obu wartości „unikalnych w iteracji" i „unikalnych w podproblemie". W żadnym z problemów algorytm DPSO z feromonem nie osiągnął minimalnej i maksymalnej wartości.

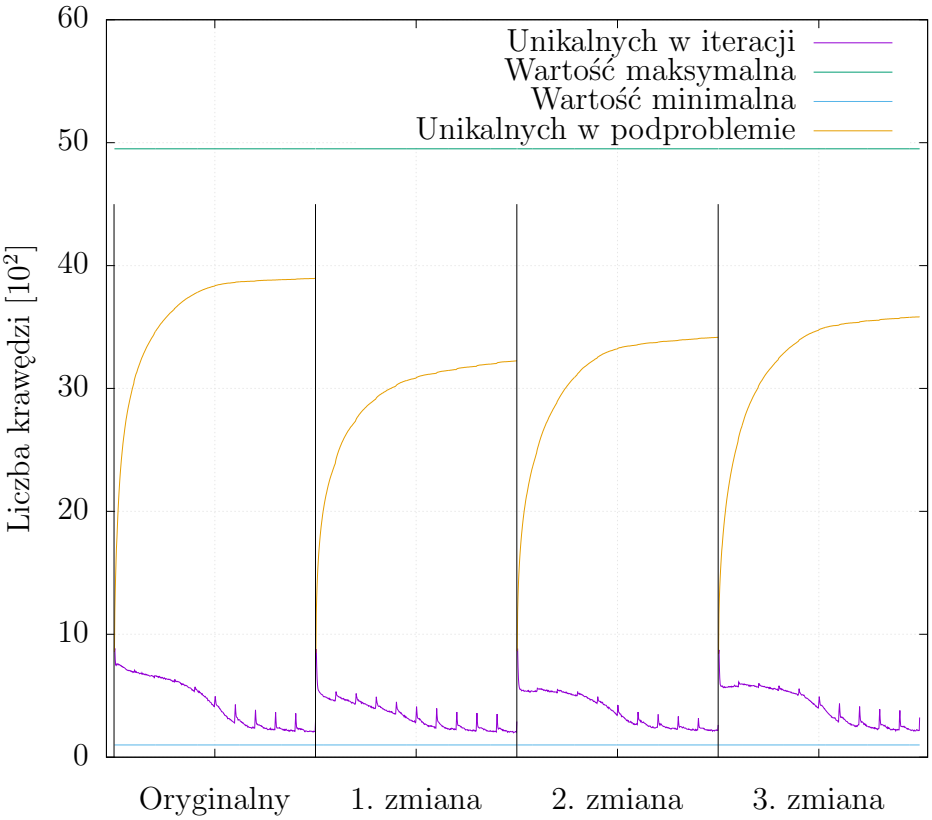
Rozproszenie feromonu

Kolejne badania polegały na obserwacji macierzy feromonowej w kolejnych iteracjach algorytmu. Miało to na celu sprawdzenie korelacji między wierzchołkami znajdującymi się w najbliższym sąsiedztwie, krawędziami znajdującymi się w optimum problemu a ich wartością feromonu. W idealnym przypadku krawędzie znajdujące się w optimum powinny zawierać się w najbliższym sąsiedztwie, a wartość feromonu odłożona na tych krawędziach powinna być maksymalna ($\tau_{\max}$). Na potrzeby badań zastosowano wizualizację macierzy feromonowej przy pomocy mapy ciepła (Rys. 3.8). Ze względu na czytelność macierzy, w pracy przedstawiono jedynie macierze dla problemu *berlin52*



Rysunek 3.6: Eksploracja przestrzeni rozwiązań algorytmu DPSO z feromonem na przykładzie problemu *berlin52*.

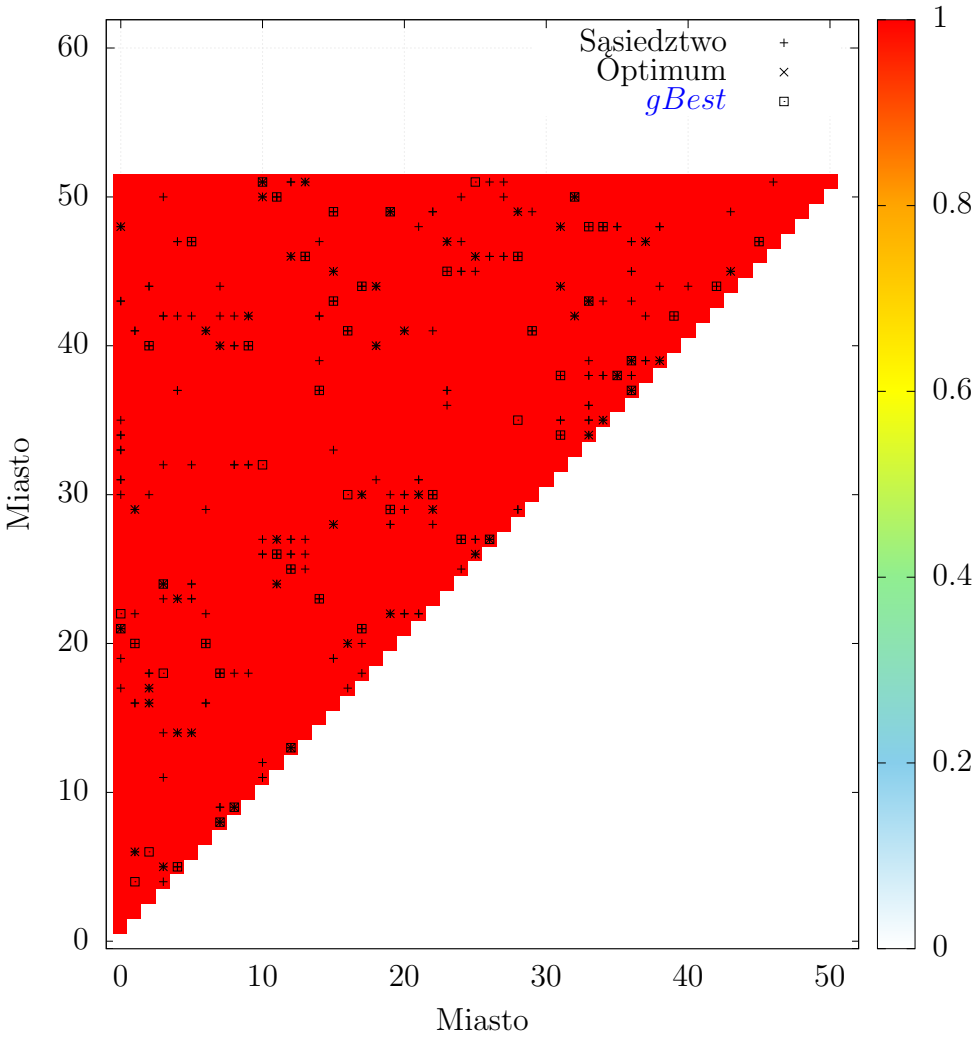
i *kroA100*. Ze względu na symetryczność macierzy pokazano jedynie górną jej część. Markery można analizować łącznie, tzn. jeśli krawędź przedstawia gwiazdę w kwadracie to oznacza, że krawędź znajduje się w: najbliższym sąsiedztwie, rozwiązaniu optymalnym oraz w zbiorze najlepszego znalezionego rozwiązania. Jeśli liczba tych punktów (gwiazdek) na rysunku jest równa rozmiarowi problemu, algorytm znalazł optimum podproblemu. Z drugiej strony przypadek, w którym krawędź należy do optimum, ale nie należy do najbliższego sąsiedztwa powoduje, że jedynie heurystyka najbliższego sąsiada na etapie dopełnienia jest w stanie znaleźć tę krawędź. Tym samym prawdopodobieństwo wybrania tej krawędzi do następnej pozycji jest relatywnie małe (w przeciwieństwie do krawędzi znajdujących się w sąsiedztwie). Na Rys. 3.8 znajduje się macierz wartości feromonu w pierwszej iteracji algo-



Rysunek 3.7: Eksploracja przestrzeni rozwiązań algorytmu DPSO z feromonem na przykładzie problemu *kroA100*.

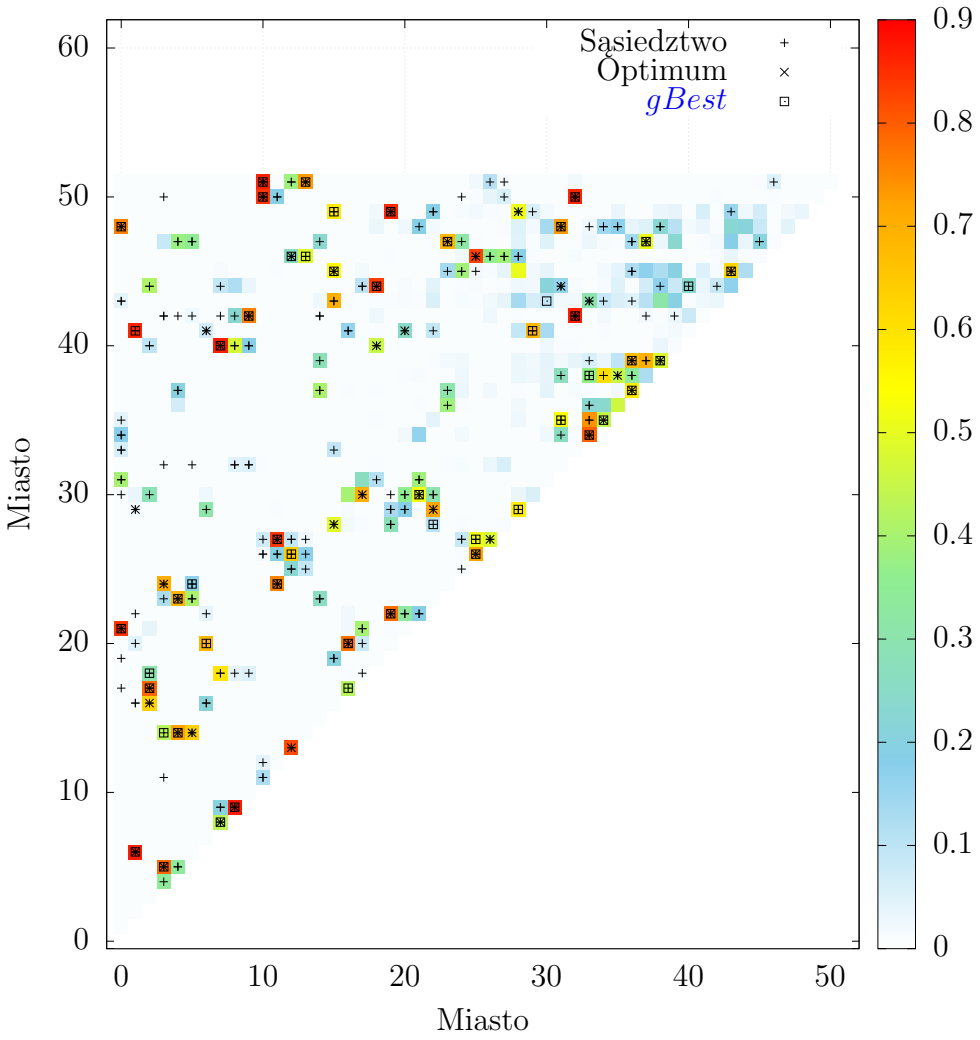
rytmu DPSO z feromonem dla problemu *berlin52* i podproblemu zerowego (oryginalny problem z biblioteki TSPLIB).
Cała macierz wartości feromonu posiada wartość 1, gdyż jest to wartość inicjalna, zgodna z podejściem zastosowanym w algorytmie $\mathcal{MAX} - \mathcal{MIN}$ zaproponowaną przez Thomasa Stützle i Holgera Hoosa [9].

W kolejnych iteracjach wartości w macierzy feromonowej dążą do wartości minimalnej – $\tau_{\min}$, co jest efektem wyparowania feromonu. Niektóre wartości kumulują znaczą wartość feromonu bliską wartości $\tau_{\max}$. Można zaobserwować również to, że feromon ma większą wartość na krawędziach należących do sąsiedztwa.
Kolejne iteracje algorytmu to wzmocniony efekt parowania feromonu, co można zaobserwować na Rys. 3.10 i Rys. 3.11.



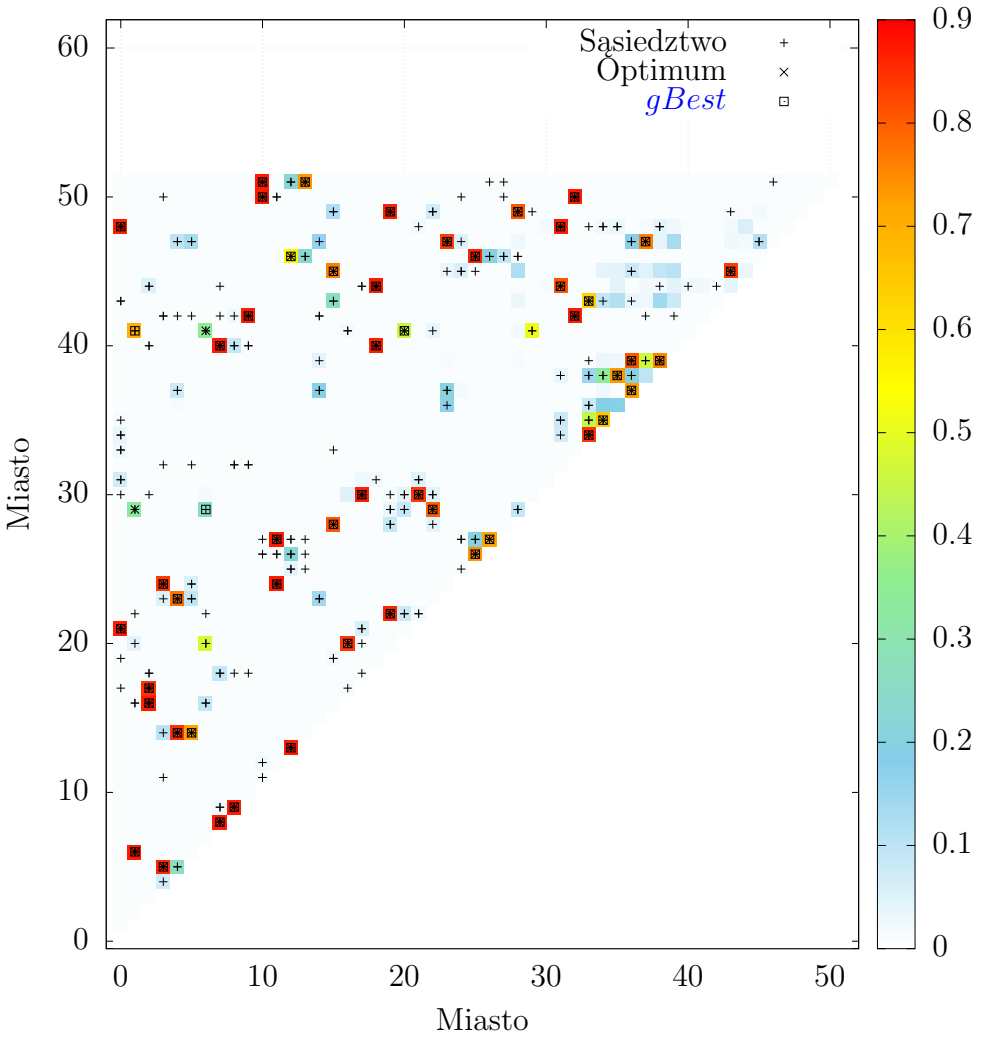
Rysunek 3.8: Wizualizacja macierzy wartości feromonu dla problemu *berlin52* i podproblemu zerowego (1 iteracja algorytmu).

W kolejnym podproblemie macierz feromonowa przekazywana jest do kolejnego podproblemu. W przypadku uruchomienia metody restartu macierzy feromonowej wszystkie jej wartości byłyby tożsame z przedstawionymi na Rys. 3.8. Kopiowanie macierzy powoduje, że niektóre wartości mają wysoką ilość feromonu, ale nie pokrywają się z krawędziami należącymi do sąsiedztwa *gBest* oraz optimum problemu. Jednak za pomocą wyparowania feromonu,



Rysunek 3.9: Wizualizacja macierzy wartości feromonu dla problemu *berlin52* i podproblemu pierwszego (50 iteracja algorytmu).

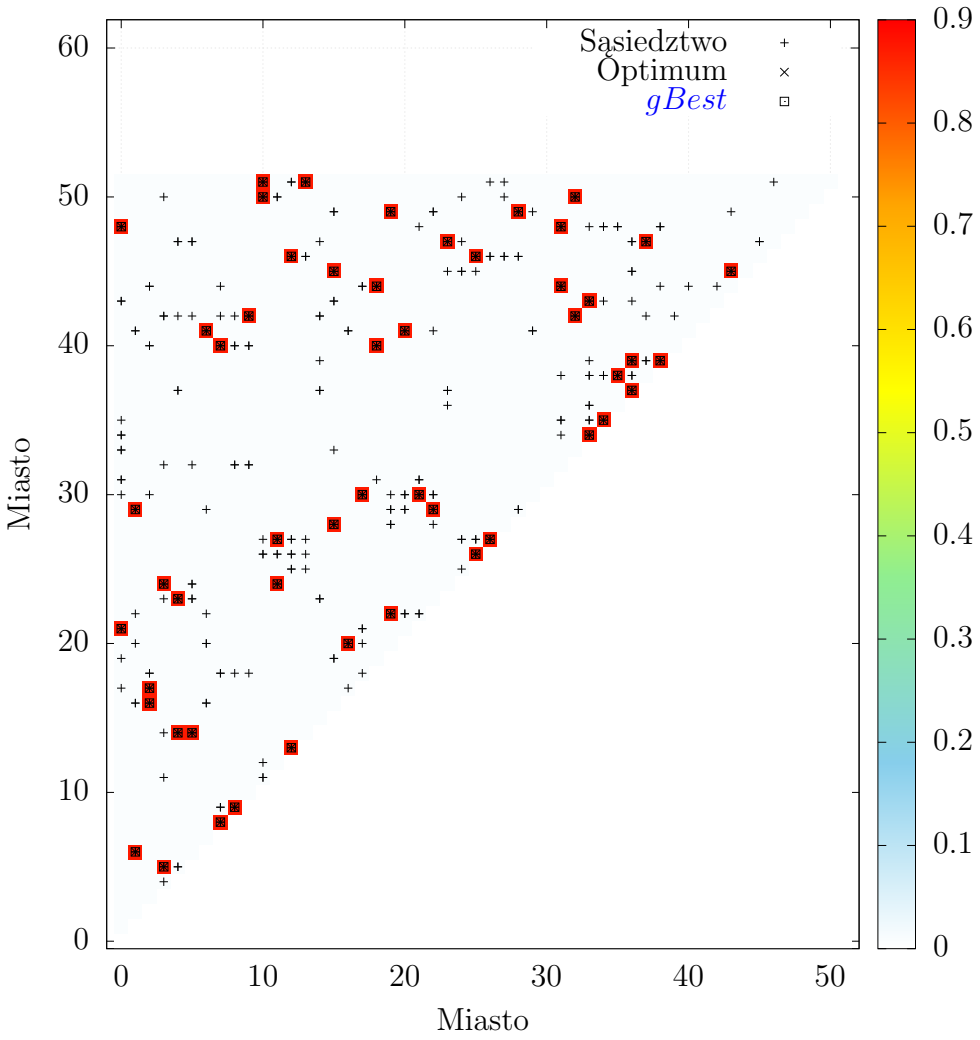
wartości te zostaną skorygowane, a wartość feromonu zostanie zwiększona dla bardziej obiecujących krawędzi. Z drugiej strony wiele krawędzi z największą wartością feromonu pokrywają się z krawędziami tworzącymi optimum problemu, co jest nie bez znaczenia w procesie optymalizacji.



Rysunek 3.10: Wizualizacja macierzy wartości feromonu dla problemu *berlin52* i podproblemu zerowego (100 iteracja algorytmu).

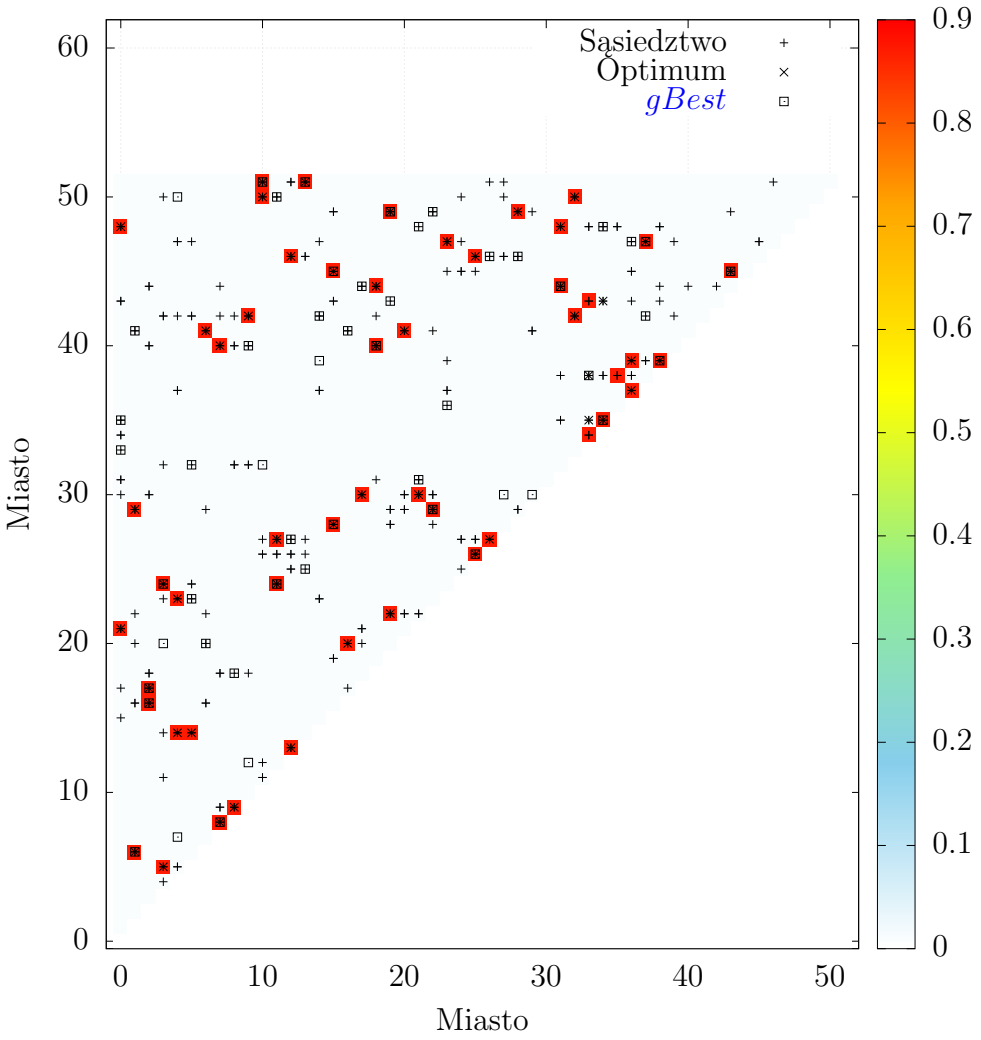
Porównanie DPSO z i bez feromonu

W Tab. 3.2 zostały przedstawione ustawienia obu algorytmów. W przypadku DPSO bez feromonu zastosowano oryginalne ustawienia zaproponowane przez Zhonga i innych [13]. Pomimo, iż oryginalny algorytm [13] używa heurystyki najbliższego sąsiada z metryką euklidesową, a zaproponowany w



Rysunek 3.11: Wizualizacja macierzy wartości feromonu dla problemu *berlin52* i zerowego podproblemu (200 iteracja algorytmu).

pracy – sąsiedztwo oparte na α-mierze dla bardziej rzetelnego porównania, obie implementacje wykonane na potrzeby pracy używają metryki α-miary do wyznaczenia najbliższego sąsiedztwa każdego wierzchołka. Znacznie poprawia ona wyniki, co zostało pokazane w pracach Boryczki i innych oraz Helsgauna [84, 8]. Wersja z przeszukiwaniem lokalnym wykonuje co 50 iteracji algorytm 3-optymalny dla każdego rozwiązania w populacji osobników.



Rysunek 3.12: Wizualizacja macierzy wartości feromonu dla problemu *berlin52* i drugiego podproblemu (1 iteracja algorytmu).

Podział w ramach jednego problemu DTSP na różne procentowe zmiany długości krawędzi w podproblemie i inne wartości parametru rozmiar sąsiedztwa ($\mathcal{N}_{size}$) wynika z badań empirycznych. Liczba sąsiednich wierzchołków dla każdego z wierzchołków w grafie problemu została dopasowana do jego rozmiaru, co ma za zadanie zwiększyć prawdopodobieństwo istnienia krawędzi należących do rozwiązania optymalnego w sąsiedztwie wierzchołków.

Tabela 3.2: Ustawienia algorytmu DPSO z i bez feromonu.

Problem	Procent zmian	DPSO										
		Bez feromonu				Z feromonem				Wspólne		
		c_1	c_2	c_3	ω	c_1	c_2	c_3	ω	i_{max}	$\mathcal{N}_{size}$	k_{max}
berlin52	3% - 20%	1,5	2	2	0,6	0,5	0,5	0,5	0,2	30	7	312
	30% - 70%	1,5	2	2	0,6	0,5	0,5	0,5	0,2	30	10	312
kroA100	3%, 10% - 30%	1,5	2	2	0,6	0,5	0,5	0,5	0,5	60	10	700
	5%, 40% - 70%	1,5	2	2	0,6	0,5	0,5	0,5	0,5	60	7	700
kroA200	3% - 20%	1,5	2	2	0,6	0,5	0,5	0,5	0,5	80	7	1800
	30% - 70%	1,5	2	2	0,6	0,5	0,5	0,5	0,5	80	15	1800
gr202	3% - 20%	1,5	2	2	0,6	0,5	0,5	0,5	0,5	80	10	2424
	30% - 70%	1,5	2	2	0,6	0,5	0,5	0,5	0,5	80	15	2424
pcb442	3% - 20%	1,5	2	2	0,6	1	1	1	1	100	15	7956
	30% - 70%	1,5	2	2	0,6	1	1	1	1	100	20	7956

Wspólna wartość tego parametru dla obu algorytmów nie powoduje, że jest ona dopasowana do konkretnego algorytmu, a jest cechą konkretnej instancji problemu. Parametry od c_1 do c_3 są stałe w obu wariantach algorytmu. Dobór parametrów jest wynikiem badań empirycznych. Na ich podstawie znaleziono takie wartości parametrów, które nie są dopasowane do konkretnej instancji problemu, a dają dobre rezultaty w przypadku ogólnym. W drugim wariancie algorytmu DPSO (z feromonem) wartość parametru ω rośnie wraz ze wzrostem liczby wierzchołków problemu. Można ją jednak wyznaczyć na podstawie rozmiaru problemu (n). W przypadku problemu *pcb442* można zauważyć wyższe wartości wszystkich parametrów. Są to wartości parametrów dla instancji problemów większych od 300 miast. Tabela 3.3 przedstawia wyniki otrzymane z obu wersji algorytmu DPSO tj. z feromonem i bez niego. Obie wersje działają na innych parametrach, poza takimi jak: wielkość sąsiedztwa $\mathcal{N}_{size}$, rozmiar populacji i_{max} i liczba iteracji k_{max}. Różniące się wartości parametrów - c_1, c_2, c_3, ω mają wpływ na prawdopodobieństwo wyboru krawędzi. Wersja z feromonem, dzięki wzmocnieniu krawędzi nie wymaga dużych wartości początkowych. Rozmiar problemów został ograniczony ze względu na bardzo czasochłonne działanie zaimplementowanych algorytmów. Każde uruchomienie zostało powtórzone 30 razy, a wyniki uśredniono. Przyjęto następujące oznaczenia: F+ i F- kolejno algorytmy z i bez zastosowania feromonu. Po kolumnie z nazwą problemu oznaczono procent zmian w każdym kolejnym podproblemie w stosunku do swojego poprzednika. Pogrubioną czcionką oznaczono najlepsze wartości, osobno dla wersji z przeszukiwaniem lokalnym

i bez niego.

Tabela 3.3: Otrzymane wyniki różnych wariantów algorytmu DPSO.

Problem	Zmian	Bez przesz. lokalnego				Z przesz. lokalnym			
		Czas [s]		Opt. [%]		Czas [s]		Opt. [%]	
		F-	F+	F-	F+	F-	F+	F-	F+
berlin52	3%	0,12	0,22	1,76	**0,01**	0,12	0,22	**0**	**0**
berlin52	5%	0,1	0,21	1	**0,22**	0,11	0,23	**0,03**	0,08
berlin52	10%	0,11	0,21	1	**0,14**	0,12	0,22	0,01	**0**
berlin52	20%	0,12	0,22	0,67	**0,35**	0,12	0,22	0,04	**0,02**
berlin52	30%	0,12	0,25	0,83	**0,67**	0,12	0,26	0,03	**0,01**
berlin52	40%	0,12	0,25	1,82	**1,22**	0,12	0,25	0,04	**0,01**
berlin52	50%	0,12	0,24	2,08	**0,98**	0,12	0,25	0,05	**0,02**
berlin52	60%	0,11	0,24	**1,12**	1,26	0,12	0,25	0,06	**0,04**
berlin52	70%	0,12	0,25	1,11	**0,69**	0,12	0,26	0,03	**0,01**
kroA100	3%	0,84	1,96	1,65	**0,91**	0,89	2,02	0,03	**0,02**
kroA100	5%	0,82	1,7	2,06	**1,03**	0,88	1,74	**0,01**	**0,01**
kroA100	10%	0,79	1,9	2,01	**1,35**	0,84	1,96	0,04	**0,02**
kroA100	20%	0,78	1,91	1,98	**1,01**	0,85	1,99	0,07	**0,03**
kroA100	30%	0,75	1,88	1,49	**1,08**	0,81	1,95	0,05	**0,03**
kroA100	40%	0,85	1,69	2,07	**1,28**	0,91	1,75	0,07	**0,05**
kroA100	50%	0,87	1,73	2,5	**1,11**	0,94	1,77	0,04	**0,02**
kroA100	60%	0,92	1,76	2,39	**1,27**	0,99	1,82	0,06	**0,05**
kroA100	70%	0,84	1,7	1,85	**1,3**	0,91	1,74	0,03	**0,02**
kroA200	3%	4,08	10,43	2,67	**1,73**	4,79	10,72	0,11	**0,05**
kroA200	5%	3,72	10,17	2,12	**1,92**	4,35	10,52	0,09	**0,06**
kroA200	10%	4,72	11,19	2,55	**2,05**	5,5	11,53	0,13	**0,1**
kroA200	20%	4,26	10,59	2,93	**2,32**	4,97	10,96	0,12	**0,09**
kroA200	30%	4,19	14,73	2,45	**2,1**	4,94	15,46	0,11	**0,1**
kroA200	40%	4,44	14,86	2,6	**2,21**	5,21	15,64	0,15	**0,13**
kroA200	50%	4,86	15,32	2,7	**2,28**	5,59	16,14	**0,1**	0,11
kroA200	60%	4,59	15,46	1,67	**1,29**	5,31	16,1	0,07	**0,05**
kroA200	70%	4,71	15,34	2,55	**1,77**	5,45	16,18	0,13	**0,1**
gr202	3%	10,66	20,26	2,58	**1,59**	11,52	21,17	**0,17**	0,18
gr202	5%	10,13	19,92	2,5	**1,58**	10,94	20,96	**0,12**	0,14
gr202	10%	9,32	19,3	2,17	**1,2**	10,11	20,18	0,11	**0,09**
gr202	20%	9,23	19,3	1,88	**1,12**	10,16	20,15	0,14	**0,13**
gr202	30%	10,29	24	2,56	**1,54**	11,15	24,91	**0,11**	0,12

Tabela 3.3: Ciąg dalszy.

Problem	Zmian	Bez przesz. lokalnego				Z przesz. lokalnym			
		Czas [s]		Opt [%]		Czas [s]		Opt [%]	
		F-	F+	F-	F+	F-	F+	F-	F+
gr202	40%	9,28	23,18	3,17	**1,67**	10,25	24,22	0,15	**0,14**
gr202	50%	9,06	22,81	2,26	**1,4**	9,93	23,88	**0,08**	0,1
gr202	60%	9,26	23,34	1,79	**1,13**	10,18	24,26	**0,1**	0,11
gr202	70%	8,66	22,89	1,89	**1,03**	9,52	23,8	**0,09**	**0,09**
pcb442	3	41,76	127,95	2,97	**2,05**	53,93	138,78	0,25	**0,2**
pcb442	5	43,35	128,76	4,37	**2,76**	56,86	138,73	0,33	**0,23**
pcb442	10	48,57	126,53	3,64	**2,65**	64,68	136,22	0,41	**0,29**
pcb442	20	49,61	127,95	3,26	**2,38**	65,22	138,07	0,32	**0,23**
pcb442	30	34,32	100,49	4,02	**2,88**	44,86	108,4	0,4	**0,3**
pcb442	40	40,97	106,63	3,98	**2,89**	51,38	115,18	0,49	**0,33**
pcb442	50	39,63	106,14	3,3	**2,71**	50,12	114,07	0,39	**0,31**
pcb442	60	37,11	104,72	3,53	**2,65**	47,5	112,72	0,34	**0,25**
pcb442	70	37,48	104,72	3,52	**2,53**	48,42	112,65	0,39	**0,29**

W prawie każdym badanym problemie, w wyniku działania algorytmu DPSO z feromonem otrzymywano krótsze rozwiązanie (sumę długości krawędzi w zbiorze pozycji). Lepsze rozwiązanie algorytm z feromonem uzyskał 35 razy, a gorszy wynik uzyskał 1 raz. Najczęściej, największą różnicę odnotowano w przypadku najmniejszej liczby zmian w kolejnych podproblemach – 3%. W wynikach dla wersji z zastosowaniem feromonu, w stosunku do wariantu bez feromonu, można zaobserwować trend spadkowy otrzymanych wyników w stosunku do procentowej liczby zmian w kolejnych podproblemach konkretnej instancji problemu DTSP. Czas przedstawiony w tabeli dotyczył jednej instancji podproblemu DTSP. Wersja z feromonem ze względu na odczyt z macierzy feromonowej oraz aktualizację śladu feromonowego działała dłużej. Najlepsze wyniki otrzymano dla wersji z przeszukiwaniem lokalnym. Zastosowanie przeszukiwania lokalnego, w tym przypadku algorytmu 3-optymalnego spowodowało, że wyniki są bardzo blisko wartości optymalnej (poniżej 1%), w każdym badanym przypadku. Miało to negatywny wpływ na działanie macierzy feromonowej ze względu na zbyt szybką zbieżność i tym samym ograniczoną eksplorację przestrzeni rozwiązań. Zaburzyło to adaptację feromonu do przestrzeni rozwiązań, przez co wyniki nie są jednoznacznie lepsze (poza największym analizowanym problem *pcb442*).

Analiza statystyczna

W celu uzupełnienia wyników badań wykonano analizę statystyczną otrzymanych wyników z poprzedniej części badań. W tym celu zastosowano test Wilcoxona dla par obserwacji. Jest to jeden z nieparametrycznych testów statystycznych. Przyjęto poziom istotności wyników na poziomie 5%. Jako „wygraną" należy rozumieć otrzymany wynik algorytmu DPSO z feromonem niższy, niż w przypadku algorytmu bez feromonu. Analogicznie należy rozumieć sytuację przeciwną („przegraną"). W podsumowaniu w Tab. 3.4, Tab. 3.5, Tab. 3.6, Tab. 3.7 suma wygranych, przegranych i remisów dotyczy tylko statystycznie istotnych uśrednionych wyników z 30. uruchomień. Liczba próbek jest równa liczbie podproblemów (11). Maksymalna wartość w przypadku wszystkich wyników statystycznie istotnych wynosi $9 \cdot 2 \cdot 11$ (liczba różnych procentowych zmian w danych, wersja z i bez przeszukiwania lokalnego, liczba podproblemów).

Tabela 3.4: Statystyczna istotność otrzymanych wyników dla różnych wariantów algorytmów DPSO bez i z feromonem dla problemu *berlin52*.

Problem	Zmian	Przesz. lokalne	Stat. istotne	Wygrane	Remis	Przegrane
berlin52	0,03	-	+	8	1	2
	0,05	-	+	7	3	1
	0,1	-	+	10	1	0
	0,2	-	-	-	-	-
	0,3	-	-	-	-	-
	0,4	-	-	-	-	-
	0,5	-	+	8	1	2
	0,6	-	-	-	-	-
	0,7	-	-	-	-	-
	0,03	+	-	-	-	-
	0,05	+	-	-	-	-
	0,1	+	+	5	6	0
	0,2	+	+	3	5	3
	0,3	+	+	3	8	0
	0,4	+	+	4	5	2
	0,5	+	+	7	4	0
	0,6	+	+	5	4	2
	0,7	+	+	5	4	2
Podsumowanie				65	42	14

Statystycznie nieistotne różnice w wynikach otrzymano 6 razy. Ze względu na

otrzymane wyniki blisko wartości optymalnej, można zauważyć relatywnie dużą liczbę remisów (42). Algorytm z feromonem uzyskał lepszy wynik 65 razy, gorszy 14 razy. W Tab. 3.5 przedstawiono wyniki analizy statystycznej dla problemu *kroA100*.

Tabela 3.5: Statystyczna istotność otrzymanych wyników dla różnych wariantów algorytmów DPSO bez i z feromonem dla problemu *kroA100*.

Problem	Zmian	Przesz. lokalne	Stat. istotne	Wygrane	Remis	Przegrane
kroA100	0,03	-	+	10	0	1
	0,05	-	-	-	-	-
	0,1	-	-	-	-	-
	0,2	-	+	9	0	2
	0,3	-	-	-	-	-
	0,4	-	+	9	0	2
	0,5	-	+	11	0	0
	0,6	-	+	9	1	1
	0,7	-	+	7	0	4
	0,03	+	+	5	6	0
	0,05	+	+	5	6	0
	0,1	+	+	5	6	0
	0,2	+	+	8	3	0
	0,3	+	-	-	-	-
	0,4	+	+	6	4	1
	0,5	+	+	5	5	1
	0,6	+	-	-	-	-
	0,7	+	+	5	3	3
Podsumowanie				94	34	15

Statystycznie nieistotne różnice w wynikach otrzymano 5 razy. Ze względu na otrzymane wyniki blisko wartości optymalnej, można zauważyć relatywnie dużą liczbę remisów (34), ale jest to wartość niższa niż w przypadku problemu *berlin52*. Algorytm z feromonem uzyskał lepszy wynik 94 razy, gorszy 15 razy. W Tab. 3.6 przedstawiono wyniki analizy statystycznej dla problemu *kroA200*.

Statystycznie nieistotne różnice w wynikach otrzymano 8 razy na 18 możliwych (ponad 44%). Otrzymano relatywnie mniej remisów (15) niż w poprzednich przypadkach, ale na taki stan wpłynęła liczba statystycznie nieistotnych różnic w wynikach. Algorytm z feromonem uzyskał lepszy wynik 81 razy, gorszy 14 razy. W

Tabela 3.6: Statystyczna istotność otrzymanych wyników dla różnych wariantów algorytmów DPSO bez i z feromonem dla problemu *kroA200*.

Problem	Zmian	Przesz. lokalne	Stat. istotne	Wygrane	Remis	Przegrane
kroA200	0,03	-	+	10	0	1
	0,05	-	-	-	-	-
	0,1	-	-	-	-	-
	0,2	-	-	-	-	-
	0,3	-	+	10	0	1
	0,4	-	-	-	-	-
	0,5	-	-	-	-	-
	0,6	-	+	9	0	2
	0,7	-	+	11	0	0
	0,03	+	+	8	2	1
	0,05	+	+	8	1	2
	0,1	+	+	7	2	2
	0,2	+	-	-	-	-
	0,3	+	+	5	4	2
	0,4	+	-	-	-	-
	0,5	+	-	-	-	-
	0,6	+	+	5	4	2
	0,7	+	+	8	2	1
Podsumowanie				81	15	14

Tab. 3.7 przedstawiono wyniki analizy statystycznej dla problemu *gr202*.

Przedostatnia analiza statystyczna dotyczyła największego analizowanego problemu - *gr202*. Statystycznie nieistotne różnice w wynikach otrzymano również 8 razy na 18 możliwych (ponad 44%). Otrzymano relatywnie mniej remisów niż w poprzednich przypadkach, bo tylko 2. Algorytm z feromonem uzyskał lepszy wynik 103 razy, gorszy 5 razy, uzyskując w ten sposób największą różnicę. W Tab. 3.8 przedstawiono wyniki analizy statystycznej dla problemu *pcb442*.

Ostatnia analiza statystyczna dotyczyła największego analizowanego problemu - *pcb442*. Statystycznie istotne różnice w wynikach otrzymano 18 razy na 18 możliwych (100%). Otrzymano 2 remisy. Algorytm z feromonem uzyskał lepszy wynik 185 razy, gorszy 11 razy.

Tabela 3.7: Statystyczna istotność otrzymanych wyników dla różnych wariantów algorytmu DPSO bez i z feromonem dla problemu *gr202*.

Problem	Zmian	Przesz. lokalne	Stat. istotne	Wygrane	Remis	Przegrane
gr202	0,03	-	+	11	0	0
	0,05	-	+	10	0	1
	0,1	-	+	11	0	0
	0,2	-	+	11	0	0
	0,3	-	+	11	0	0
	0,4	-	+	10	0	1
	0,5	-	+	11	0	0
	0,6	-	+	11	0	0
	0,7	-	+	11	0	0
	0,03	+	-	-	-	-
	0,05	+	-	-	-	-
	0,1	+	-	-	-	-
	0,2	+	+	6	2	3
	0,3	+	-	-	-	-
	0,4	+	-	-	-	-
	0,5	+	-	-	-	-
	0,6	+	-	-	-	-
	0,7	+	-	-	-	-
Podsumowanie				103	2	5

Zależność w kontekście liczby iteracji

Asumptem do kolejnych badań było określenie zależności między liczbą iteracji (k_{max}), a jakością otrzymanych wyników. Jest to kluczowy związek w przypadku, gdy istnieje ograniczony budżet obliczeniowy. Jest to również istotny instrument do badania zbieżności algorytmu do optimum. W Tab. 3.9 przedstawiono liczbę ewaluacji - sumaryczną liczbę sprawdzonych rozwiązań. W przypadku obu algorytmów DPSO, liczba ta przedstawia iloczyn liczby iteracji k_{max} i rozmiaru populacji i_{max}. Liczba ewaluacji została sprzężona z rozmiarem problemu n i określona wzorem: $n \cdot p_{mu}$.

Rysunek 3.13 przedstawia rezultat otrzymany z implementacji algorytmu w stosunku do liczby iteracji. Kolorem niebieskim oznaczono wyniki otrzymane z algorytmu DPSO z feromonem. Kolorem zielonym i pomarańczowym oznaczono

Tabela 3.8: Statystyczna istotność otrzymanych wyników dla różnych wariantów algorytmów DPSO bez i z feromonem dla problemu *pcb442*.

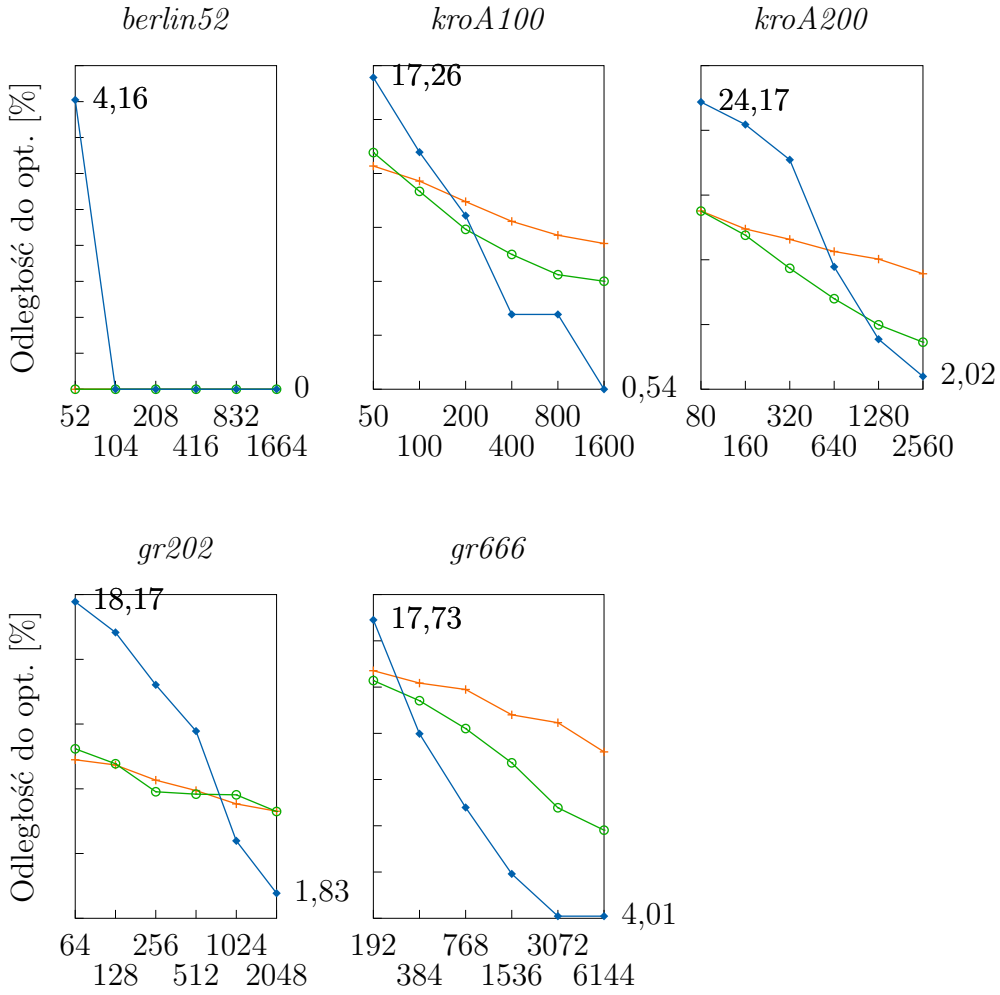
Problem	Zmian	Przesz. lokalne	Stat. istotne	Wygrane	Remis	Przegrane
pcb442	0,03	-	+	11	0	0
	0,05	-	+	11	0	0
	0,1	-	+	10	0	1
	0,2	-	+	9	0	2
	0,3	-	+	10	0	1
	0,4	-	+	11	0	0
	0,5	-	+	9	0	2
	0,6	-	+	10	0	1
	0,7	-	+	11	0	0
	0,03	+	+	9	0	2
	0,05	+	+	11	0	0
	0,1	+	+	11	0	0
	0,2	+	+	10	1	0
	0,3	+	+	11	0	0
	0,4	+	+	11	0	0
	0,5	+	+	9	0	2
	0,6	+	+	10	1	0
	0,7	+	+	11	0	0
Podsumowanie				185	2	11

Tabela 3.9: Liczba sprawdzonych rozwiązań dla jednego podproblemu DTSP.

Problem		Mnożnik p_{mu} i całkowita liczba ewaluacji					
Nazwa	n	32	64	128	256	512	1024
berlin52	52	1664	3328	6656	13312	26624	53248
kroA100	100	3200	6400	12800	25600	51200	102400
kroA200	200	6400	12800	25600	51200	102400	204800
gr202	202	6464	12928	25856	51712	103424	206848
gr666	666	21312	42624	85248	170496	1875456	681984

algorytm DPSO bez feromonu z ustawieniami kolejno oryginalnym i z Tab. 3.2 (dla wersji z feromonem). Na wykresie oznaczono również wynik otrzymany dla algorytmu DPSO z feromonem w pierwszej i ostatniej fazie testów - kolejno przy

najmniejszej i największej liczbie iteracji. Liczba zmian w kolejnych podproblemach wynosi 3% oprócz problemu *gr666*, w którym wykorzystano problem TSP z biblioteki TSPLIB.



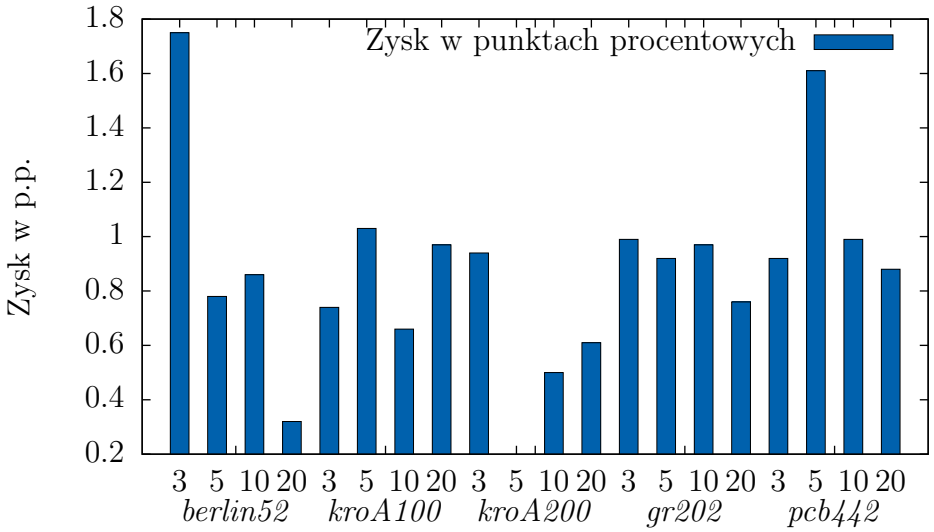
Rysunek 3.13: Zbieżność algorytmu DPSO z i bez feromonu wyrażona w procentach od optimum w stosunku do różnych wartości parametru k_{max} (liczba iteracji).

Dla problemu *berlin52* powyżej 104 iteracji, uruchomienie algorytmu pozwala otrzymać te same wyniki. W przypadku bardzo małej liczby iteracji algorytm DPSO

bez feromonu daje lepsze wyniki. Czas potrzebny dla przystosowania macierzy feromonowej do przestrzeni rozwiązań powoduje, że algorytm DPSO z feromonem pozwala otrzymać lepsze wyniki dla większej liczby iteracji. Znacząco różnią się charakterystyki obu algorytmów. Wersja z feromonem charakteryzuje się bardziej dynamiczną zbieżnością do optimum. Przy małej liczbie iteracji daje słabe wyniki, ale przy większej liczbie iteracji znacznie poprawia otrzymane wyniki. Przy maksymalnej liczbie iteracji, we wszystkich badanych problemach, najlepszy wynik otrzymano z algorytmu DPSO z feromonem. Różnicę widać nie tylko nominalnie, ale również można ją zaobserwować na rysunku 3.13. W największym badanym problemie (*gr666*) można również zauważyć znaczną różnicę wyników dla mniejszej liczby iteracji. Dodatkowo należy zaznaczyć, że największa liczba iteracji w każdym badanym problemie nie jest wygórowana - w stosunku do rozmiaru problemu.

Transfer wiedzy między podproblemami

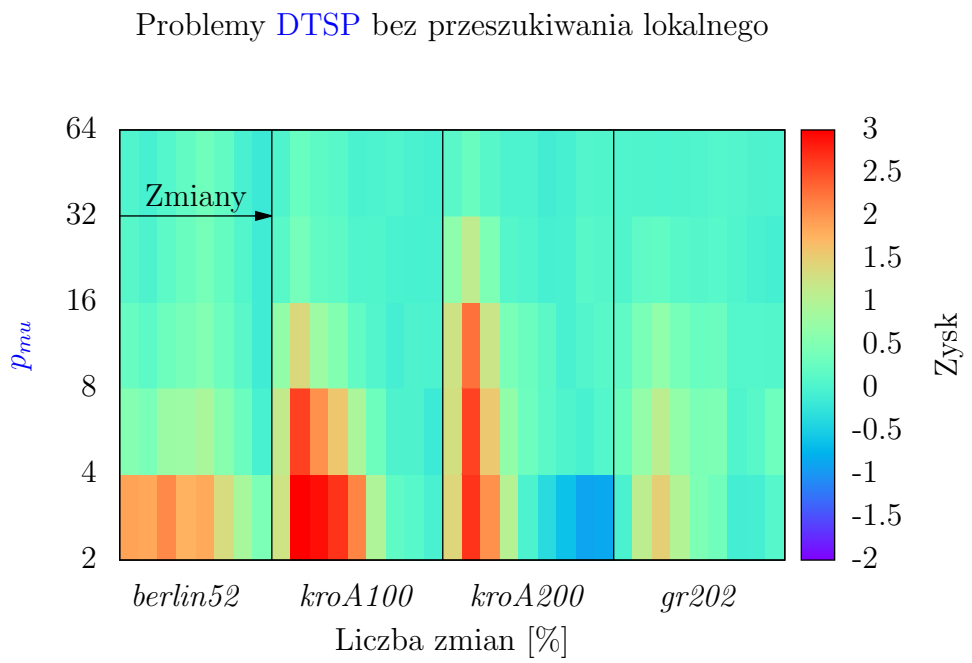
Rysunek 3.14 przedstawia omawiany zysk z kopiowania macierzy feromonowej w kolejnych podproblemach w stosunku do różnej wielkości zmian (modyfikacji macierzy odległości).



Rysunek 3.14: Różnica wyników otrzymanych z algorytmu DPSO bez i z feromonem (bez przeszukiwania lokalnego).

Intuicyjnie wydaje się oczywiste, że zysk z kopiowania macierzy feromonowej jest zależny od tego, jak bardzo kolejny podproblem różni się od swojego poprzedni-

ka. Zostało to potwierdzone empirycznie w postaci otrzymanych wyników. Należy również zaznaczyć, że żaden parametr nie został zmieniony w trakcie działania algorytmu. Ostateczny wynik powstał z różnicy między otrzymanym wynikiem bez kopiowania macierzy i wynikiem otrzymanym z wariantu z transferem wiedzy, co przedstawia oś pionowa (punkty procentowe różnicy wyników). Kolejny rysunek 3.15 przedstawia mapę ciepła przedstawiającą różnicę między kopiowaniem macierzy feromonowej a jej resetem w stosunku do liczby iteracji oraz liczby zmian między następującymi po sobie zmianami danych w kolejnych podproblemach. Na osi x przedstawiono kolejne problemy. W ramach jednego problemu ograniczonego linią pionową znajdują się coraz większa liczba zmian w danych kolejnych podproblemów. Ich wielkość jest następująca – 3%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%. Na osi y oznaczono mnożnik, który wpływa na liczbę iteracji algorytmu.



Rysunek 3.15: Różnica między kopiowaniem macierzy feromonowej między podproblemami a jej resetem (bez przeszukiwania lokalnego).

Największy zysk można odnotować dla najmniejszej liczby zmian. Determinuje to relacje między podobieństwem rozwiązania optymalnego sprzed zmian długości krawędzi w stosunku do rozwiązania optymalnego po zmianach. Formalnie jest to

przecięcie krawędzi rozwiązania optymalnego sprzed zmian danych podproblemu w ramach problemu DTSP, a rozwiązania optymalnego po zmianie długości krawędzi. Na podstawie wyników przedstawionych w na rysunku 3.15 można wywnioskować, że kopiowanie macierzy najlepiej sprawdza się dla mniejszej liczby modyfikacji.

Rozdział 4

Sąsiedztwo w algorytmie

Użycie sąsiedztwa w technice optymalizacyjnej ma na celu redukcję rozmiaru przeszukiwanej przestrzeni rozwiązań. Jest to szczególnie przydatne w trudnych problemach (pod względem złożoności obliczeniowej). W rozdziale tym zostanie omówiona teoria tolerancji oraz teoria Helda-Karpa, na bazie których powstała α-miara Kelda Helsgauna. Ma ona zastosowanie w funkcji sąsiedztwa, która została zastosowana w algorytmie DPSO z feromonem. Efektywność sąsiedztwa została niejednokrotnie wykazana w badaniach [7, 84, 85]. Przykładowo dla problemu *att532* jedna z krawędzi należąca do rozwiązania optymalnego jest na 22 pozycji na liście najbliższych sąsiadów jednego z wierzchołków używając miary euklidesowej. Używając α-miary ta sama krawędź jest na 14 miejscu [7, 8]. Sąsiedztwo znacząco przyspieszyło działanie heurystyki LKH [86]. Efektywność algorytmu została przedstawiona w wielu eksperymentach na instancjach euklidesowych TSP o rozmiarze od 10 000 do 10 000 000 miast. Pomimo, iż użycie sąsiedztwa bazującego na α-mierze znacząco poprawia efektywność algorytmu LKH, algorytm jego tworzenia jest rzadko opisany w literaturze. Szczególnie dotyczy to literatury polskojęzycznej, w której brakuje jakichkolwiek opracowań.

4.1 Teoria tolerancji

Przedmiotem badań teorii tolerancji jest wpływ zmian danych wejściowych na przestrzeń rozwiązań, w szczególności na rozwiązanie optymalne [87, 88, 89, 90]. Często w literaturze obcojęzycznej teoria tolerancji nazywana jest ang. *Sensitivity Analysis*. Ma ona szerokie zastosowanie w problemach grafowych, takich jak TSP czy problemie przydziału [91]. W dalszej części pracy zostanie omówiona teoria tolerancji dla TSP.

Każdej krawędzi można przypisać górną i dolną tolerancję. Wartością tolerancji

może być wartość skalarna $c \in \mathbb{Z}$, $+\infty$ lub $-\infty$. Niech e oznacza dowolną krawędź, a x^* rozwiązanie optymalne (lub jedno z rozwiązań optymalnych), a x^+ oznacza jedno z rozwiązań problemu nie należące do rozwiązań optymalnych.

- Górną tolerancję można interpretować jako informację o tym, jak bardzo wagi krawędzi należące do rozwiązania optymalnego x^* można wydłużyć (zwiększyć wagę krawędzi, przy założeniu problemu minimalizacji), aby nie zmieniło się optimum. Dla każdej krawędzi wchodzącej w skład rozwiązania optymalnego x^* będzie to wartość skalarna. W przeciwnym wypadku zostanie przypisana wartość $+\infty$, ponieważ zwiększenie wagi krawędzi nie należącej do x^* (wartością większą od zera) nie zmieni rozwiązania optymalnego x^* [88]. Formalnie, górną tolerancję można zapisać następująco:

$$u_{x^*}(e) := \sup\{c \in \mathbb{R}_0^+ | x^*\}\,.$$

- Dolna tolerancja z kolei odpowiada na pytanie, o ile można skrócić krawędzie nie należące do rozwiązania optymalnego x^+, aby optimum x^* się nie zmieniło. Formalnie, dolną tolerancję można zapisać następująco:

$$l_{x^+}(e) := \sup\{c \in \mathbb{R}_0^+ | x^+\}\,.$$

  Dolna tolerancja jest konkretną wartością dla $e \in x^+$ oraz $-\infty$ dla $e \in x^*$.

Jednym z tematów związanych z problemem DTSP jest właśnie teoria tolerancji. Gdyby istniał algorytm dokładny należący do klasy złożoności innej, niż NP, byłaby możliwość oszacowania wpływu zmian wag na optimum [87]. Nawet nie znając a priori zmian wystarczy porównać macierze odległości D dla podproblemu t oraz $t+1$, co można wykonać w czasie kwadratowym w stosunku do n. Jednak obliczenie wartości dolnej lub górnej tolerancji jest bardziej złożone czasowo, niż rozwiązanie problemu TSP. Pomimo tego, teorię tolerancji z sukcesem stosuje się, m.in. w tworzeniu sąsiedztwa wierzchołków, zastępując inne miary, np. euklidesową. Zamiast rozwiązania optymalnego stosuje się rozwiązanie pseudo-optymalne, co do którego nie ma pewności, że jest optymalne lub jest w pewnym sąsiedztwie (w pewnej odległości) od rozwiązania optymalnego. Tak zastosowana teoria tolerancji została użyta przez Kelda Helsgauna do tworzenia sąsiedztwa wierzchołków bazującego na α-mierze [7], opisana w dalszej części pracy.

Niezbadanym dotąd problemem badawczym jest znalezienie algorytmu poszukującego rozwiązania problemu TSP, nie według kryterium najkrótszego rozwiązania, a najbardziej odpornego na losowe zmiany długości krawędzi (zarówno wydłużenia jak i skrócenia dowolnych krawędzi) oraz leżącego w sąsiedztwie rozwiązania optymalnego.

4.2 Teoria Helda-Karpa

Held i Karp opublikowali w 1970 i 1971 roku dwie prace zatytułowane „The Traveling Salesman Problem and Minimum Spanning Trees" i „The Traveling Salesman Problem and Minimum Spanning Trees part II" [92, 93]. Dotyczyły one szacowania długości trasy optymalnej dla problemu TSP oraz wykorzystania tej informacji w procesie optymalizacji dla problemu komiwojażera. Sama teoria Helda-Karpa, jak i metoda wspinaczki, zaproponowana przez autorów, nie gwarantują, że otrzymane rozwiązanie będzie poprawne z punktu widzenia problemu TSP. Jednak w krótkim czasie algorytm znajduje bardzo dobre rozwiązania – zarówno pod względem liczby wspólnych krawędzi otrzymanego grafu z optimum problemu oraz sumie długości krawędzi tego grafu, w stosunku do sumy długości krawędzi rozwiązania optymalnego. Otrzymany wynik jest oszacowaniem z dołu, tzn. $x \leqslant x^*$, gdzie: x to oszacowane optimum, x^* to optimum problemu). Przedstawione badania na bibliotece TSPLIB pokazały, że zastosowanie teorii Helda-Karpa pozwala oszacować optimum w wielu przypadkach z dokładnością do mniej niż 1% długości rzeczywistego optimum problemu [8]. Badania nad analizą probabilistyczną wyników otrzymanych z algorytmu HKLO (Held-Karp Lower Bound) można znaleźć w [94, 95, 96]. Warta uwagi jest również praca Westerlund i inni [97], w której można znaleźć analizę specjalnych przypadków dla grafów otrzymanych tą metodą oraz jej generalizacji. Poniżej zostaną zaprezentowane dwa podstawowe nurty zastosowań teorii Helda-Karpa.

Szacowanie długości trasy optymalnej

W dynamicznym problemie TSP, układ miast ulega okresowym zmianom. Po każdej z nich, nowy podproblem, wywodzący się z poprzedniego, może mieć inną trasę optymalną [84]. Problem ten można rozpatrywać jako serię statycznych problemów TSP. Określenie maksymalnych zmian, jakie mogą wystąpić w krawędziach, aby nie zmieniło się optimum jest przedmiotem badań teorii tolerancji. Nie istnieje jednak algorytm nie należący do klasy NP, który jest w stanie wyznaczyć te wartości [87]. Z tego powodu zastosowanie algorytmu przybliżonego wyznaczającego oszacowaną długość trasy komiwojażera może pomóc w sprawdzeniu, czy trasa optymalna została zmieniona lub odpowiedź na pytanie, jak daleko od optimum jest algorytm optymalizacyjny.

Estymacja długości trasy optymalnej jest podstawą szybkich algorytmów dokładnych, np. *podziału i ograniczeń*. Wybrane algorytmy tego typu, które używają teorii Helda-Karpa można znaleźć w wielu pracach [98, 99, 100, 101].

Zastosowanie w funkcji sąsiedztwa

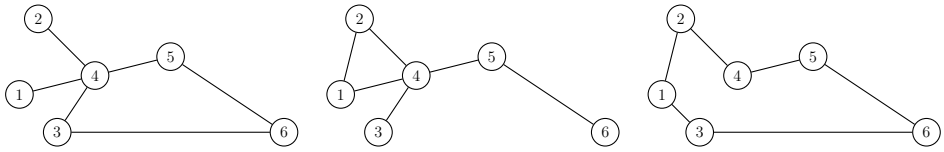
Jedną z metod redukcji rozmiaru przeszukiwanej przestrzeni rozwiązań dla problemu TSP jest sąsiedztwo. Dla każdego wierzchołka wyznacza się odległość do pozostałych wierzchołków na podstawie przyjętej miary. W trakcie optymalizacji dla każdego wierzchołka badane jest jego najbliższe otoczenie, pomijając pozostałe wierzchołki leżące dalej. Odległość ta może być mierzona za pomocą różnych miar, np. euklidesowej. Helsgaun przedstawił swoją miarę i użył jej do konstrukcji funkcji sąsiedztwa [8, 7]. Bazuje ona na dolnej tolerancji krawędzi należących do grafu otrzymanego z estymacji Held-Karpa.

Definicje stosowane w teorii Held-Karpa

Podstawą teorii tolerancji jest 1-drzewo. Jest to relaksacja pojęcia drzewa. W przeciwieństwie do niego, 1-drzewo nie jest acykliczne.

Definicja 4.2.1. 1-drzewo To spójny graf ważony zawierający dokładnie jeden cykl.

Rysunek 4.1 przedstawia trzy przykładowe 1-drzewa otrzymane na podstawie tego samego zbioru wierzchołków (punktów) na płaszczyźnie.



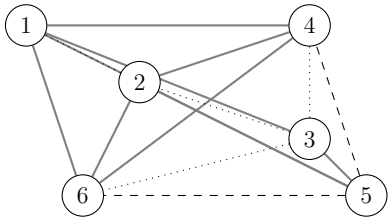
(a) (b) (c)

Rysunek 4.1: Przykład trzech różnych 1-drzew z tego samego zbioru wierzchołków.

Uwaga. Każde poprawne rozwiązanie problemu komiwojażera to 1-drzewo (każdy wierzchołek jest stopnia 2).

Held i Karp w swojej pracy przedstawili algorytm do tworzenia specjalnego przypadku 1-drzewa, tzn. minimalnego 1-drzewa (min-1-drzewa). Wygląda on następująco: utwórz *minimalne drzewo rozpinające* T dla zbioru $\mathcal{V} - \{v_s\}$. Następnie połącz wierzchołek v_s z dwoma wierzchołkami znajdującymi się w *minimalnym drzewie rozpinającym* T, których krawędzie mają najmniejsze wagi. Rysunek 4.1 przestawia przykładowe 1-drzewo otrzymane z powyższego algorytmu. Wybrany

wierzchołek, który początkowo nie wchodzi w skład MST to 5. W późniejszym etapie został połączony z drzewem krawędziami zaznaczonymi linią przerywaną. Linią wykropkowaną pokazano krawędzie tworzące MST. Pozostałe nieużyte krawędzie zaznaczono linią szarą.



Rysunek 4.2: Przykładowe min-1-drzewo (linie przerywane i wykropkowane).

Uwaga. Koszt min-1-drzewa jest związany z wyborem wierzchołka v_s.

4.3 Sąsiedztwo i miara Helsgauna

Koncepcja α miary stosowana podczas wyznaczania najbliższych sąsiadów bazuje na dolnej tolerancji min-1-drzewa [8, 92]. Odległość pomiędzy parą wierzchołków $\alpha(a, b)$ $a, b \in \mathcal{V}$ można zdefiniować jako różnicę między sumą krawędzi należących do min-1-drzewa, a kosztem tego grafu (pseudo-drzewa) po dodaniu do niego wagi krawędzi $\langle a, b\rangle$. Model formalny przedstawia wzór 4.1:

$$\alpha(a, b) = w(T^{+}(\langle a, b\rangle)) - w(T), \tag{4.1}$$

gdzie:

- T oznacza zbiór krawędzi min-1-drzewa,
- T^{+} zawiera krawędzie z T oraz dodatkowo krawędź $\langle a, b\rangle$ z zastosowaniem następujących reguł:
  - jeżeli $\langle a, b\rangle$ należy do T, wtedy grafy są równe,
  - jeżeli a lub b to wybrany wierzchołek v_s, wtedy $\langle a, b\rangle$ zmienia dłuższą krawędź spośród dwóch, które łączą v_s z 1-drzewem (Rys. 4.2),
  - w przeciwnym wypadku, krawędź $\langle a, b\rangle$ zostaje dodana do T, co powoduje powstanie drugiego cyklu. Ostateczna postać T^{+} powstaje po usunięciu najdłuższej krawędzi z pozostałych krawędzi tworzących ten dodatkowy cykl.

Wartość $\alpha(a, b)$ określa ważność krawędzi $\langle a, b\rangle$ w rozwiązaniu pseudo-optymalnym. W przypadku, gdy krawędź należy do min-1-drzewa otrzymuje wartość 0, co oznacza, że oba wierzchołki krawędzi są najbliższymi sąsiadami. Sąsiedztwo powstaje poprzez obliczenie wartości α dla każdej krawędzi problemu TSP. Dla każdego wierzchołka wszystkie do niego przystające krawędzie są sortowane na podstawie otrzymanej α-miary (rosnąco). Najbliższym sąsiadem wierzchołka a jest ten wierzchołek, którego wartość α-miary jest najmniejsza. W przypadku, gdy należy wyznaczyć zadaną liczbę najbliższych sąsiadów, dla każdego wierzchołka lista przystających wierzchołków jest sortowana i wybierane są z niej wszystkie wierzchołki leżące w pewnym otoczeniu wierzchołka, dla którego obliczana jest miara α. Otrzymane krawędzie min-1-drzewa są minimalne, co jest zgodne z definicją MST. Dlatego spełnione jest równanie $\alpha(a, b) \geqslant 0$ [8]. Helsgaun w swojej pracy pokazał, że algorytm tworzenia sąsiedztwa przy użyciu α-miary może mieć złożoność obliczeniową $O(n^2)$.

4.4 Metoda wspinaczki

Im więcej jest krawędzi wspólnych min-1-drzewa z rozwiązaniem optymalnym, tym sąsiedztwo będzie lepszej jakości. Held i Karp w swojej przełomowej pracy [92, 93] pokazali, że zmiana długości krawędzi przeprowadzona zgodnie ze wzorem (4.2), nie powoduje zmiany zbioru krawędzi należących do optimum problemu, tylko zmianę krawędzi należących do min-1-drzewa. Obserwacja ta została wykorzystana w algorytmach, które modyfikują wagi krawędzi, tak aby jak najwięcej wierzchołków znajdujących się w min-1-drzewo była stopnia drugiego (funkcja celu):

$$w^+(a, b) = w(a, b) + \pi_a + \pi_b\,, \tag{4.2}$$

gdzie:

- a, b to wierzchołki,
- $w(a, b)$ to waga krawędzi z oryginalnej macierzy odległości,
- π_a oraz π_b to dodatkowe koszty, które powodują zmianę wag krawędzi w min-1-drzewie [92, 93].

Suma wag min-1-drzewa zmienia się zgodnie ze wzorem (4.3):

$$\max_k \hat{w}(\pi^k) = \sum_{\langle a,b\rangle \in T^k} w^+(a, b) - 2\sum_{i=1}^{n} \pi_i^k\,, \tag{4.3}$$

gdzie:

- k to numer iteracji,
- n to liczba wierzchołków,
- T^k to zbiór krawędzi należących do min-1-drzewa w iteracji k, który tworzony jest na podstawie długości krawędzi danych wzorem (4.2).

Wzór (4.3) stanowi funkcję celu optymalizacji. Polega ona na znalezieniu wektora $\pi = (\pi_1, \pi_2, ..., \pi_n)$, który maksymalizuje funkcję celu daną wzorem (4.3).

Dowód. Niech T^1 będzie 1-drzewem oraz wierzchołek $a \in T^1$. Zgodnie ze wzorem (4.2) wszystkie krawędzie przystające do wierzchołka a są zwiększane o π_a. Suma kar jest stała, jeżeli każdy wierzchołek jest tego samego stopnia w 1-drzewie. To implikuje, że optimum problemu TSP, które jest min-1-drzewem (zwiększone o kary z π) jest równe $2\sum_{i=1}^{n} \pi_i$ (każdy wierzchołek w rozwiązaniu problemu TSP jest stopnia drugiego). □

Algorytm optymalizacji

Efektywnym algorytmem optymalizacyjnym, który znajduje taki wektor π, który maksymalizuje $\hat{w}$ jest technika subgradientowa nazywana metodą wspinaczki (*ascent*)[92, 93]. Jest to iteracyjna metoda bazująca na uogólnieniu pojęcia gradientu. W kolejnym kroku następuje przemieszczanie się w kierunku coraz bardziej obiecujących rozwiązań, otrzymując kolejne wektory π. Kierunek wyznacza subgradient, zgodnie ze wzorem (4.4):

$$\pi^{k+1} = \pi^k + t^k \cdot v^k \,, \tag{4.4}$$

gdzie:

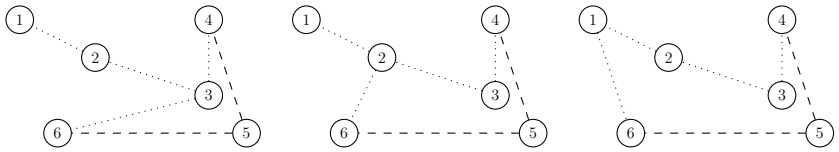
- v^k to gradient (wektor kierunku),
- t^k to krok algorytmu, wartość skalarna większa od zera.

Przykładowy proces optymalizacji subgradientowej przedstawia Rys. 4.3.
Pseudokod 6 przedstawia schemat optymalizacji subgradientowej.

W linii 6 metoda Helsgaun, wykorzystuje wzór (4.5), który jest logiczną konsekwencją wzoru (4.4):

$$\pi^{k+1} = \pi^k + t^k(0.7v^k + 0.3v^{k-1}) \,, \tag{4.5}$$

gdzie $v^k = d^k - 2$ – stopień wierzchołka pomniejszony o dwa. Wartość d^k równa 2 powoduje, że koszt π_i jest niezmieniany w kroku $k + 1$.



Rysunek 4.3: Przykładowe 1-drzewo składające się z 6 wierzchołków oraz zgodnie z definicją, jeden cykl.

Algorytm 7: Optymalizacja subgradientowa.

```
1 k = 0, π^0 = 0 and Ŵ = −∞;
2 for k = 0 → k_max do
3 |   ŵ(π^k) = Nowe rozwiazanie(π^k);
4 |   W = max(W, ŵ(π^k));
5 |   π^{k+1} = π^k + t^k · v^k;
6 end
```

Zostało udowodnione, że w przypadku, gdy $t^k \to 0$ dla $k \to \infty$ i $\sum t^k = \infty$, wzór (4.4) zbiega się do optimum [102, 8]. W praktyce proces ten ma zbyt wolną zbieżność, aby był on użyteczny. Kolejny algorytm 8 przedstawia stosowaną funkcję oceny oraz tworzenie nowego rozwiązania.

Algorytm 8: Tworzenie nowego rozwiązania.

```
1 wejście: π^k;
2 T' = MST(π^k);
3 T^k = MinOneTree(T');                          // Wzór (4.3)
4 wyjście: ŵ(π^k) = Σ_{⟨a,b⟩∈T^k} w⁺(a,b) − 2 Σ_{i=1}^{n} π_i^k
```

Ocena rozwiązania następuje po otrzymaniu minimalnego drzewa rozpinającego oraz po utworzeniu na jego podstawie min-1-drzewa. Ocena jest zgodna ze wzorem (4.3). Oceniane rozwiązanie zbudowane jest na podstawie macierzy odległości oraz wektora π^k, zgodnie ze wzorem (4.2).

Długość oszacowanej trasy

Im więcej wspólnych krawędzi zawiera min-1-drzewo z rozwiązaniem optymalnym, tym lepsze wyniki można otrzymać przy pomocy sąsiedztwa. Tabela 4.1

powstała na podstawie implementacji teorii Helda-Karpa, wraz z pewnymi modyfikacjami zaproponowanym przez Helsgauna, które można znaleźć w pracy [8, 7]. Należy również zaznaczyć, że w tej wersji jest to algorytm bezparametrowy. Badania przeprowadzono na zbiorze wybranych problemów z biblioteki TSPLIB. Ostatnia liczba w nazwie problemu oznacza liczbę wierzchołków.

Tabela 4.1: Wyniki porównania algorytmu Helsgouna z optimum problemu.

Problem	Czas [s]	Optimum	Estymacja
bayg29	0,01	1610	1607,78
bays29	0,01	2020	2013,46
berlin52	0,04	7542	7542
eil76	0,03	538	536,92
kroA100	0,33	21282	20936,42
eil101	0,1	629	627,46
gr120	0,34	6942	6886,97
pr124	0,24	59030	54826,09
ch130	0,26	6110	6073,9
pr136	0,64	96772	94012,17
pr152	0,23	73682	62728,77
kroA200	1,02	29368	29016,27
gr202	7,38	40160	39984,04
tsp225	0,76	3861	3827,25
pr226	3,1	80369	75469,48
a280	1,63	2579	2566
pr299	7,46	48191	47244,01
pr439	29,86	107217	104279,66
pcb442	4,87	50778	50485,7
ali535	78,1	202339	200296,17
gr666	62,25	294358	290042,78
pcb1173	112,57	56892	56351,01
pr2392	686,97	378032	373489,22

Należy zwrócić uwagę na dwa aspekty działania algorytmu. Czas działania algorytmu nie jest wprost proporcjonalny do rozmiaru problemu, co można zaobserwować na przykładzie problemu *pr439* oraz *pcb442*. Jakość oszacowania nie zależy również od rozmiaru problemu, gdyż zarówno dla wielu mniejszych jak i największych

badanych problemach, jakość estymacji oscyluje w granicach 1% poniżej optimum (oszacowanie z dołu).

Podobieństwo do trasy optymalnej

Poprzednie badania miały na celu porównanie otrzymanej długości trasy z min-1-drzewa, na tle rozwiązania optymalnego. Kolejne badania dotyczą przecięcia zbioru wierzchołków w min-1-drzewie i optimum. Wyniki przedstawiono w Tab. 4.2. Ostatnia kolumna zawiera wartość wyrażoną w procentach. Jeśli wartość wynosi 100% otrzymana trasa jest identyczna z rozwiązaniem optymalnym problemu.

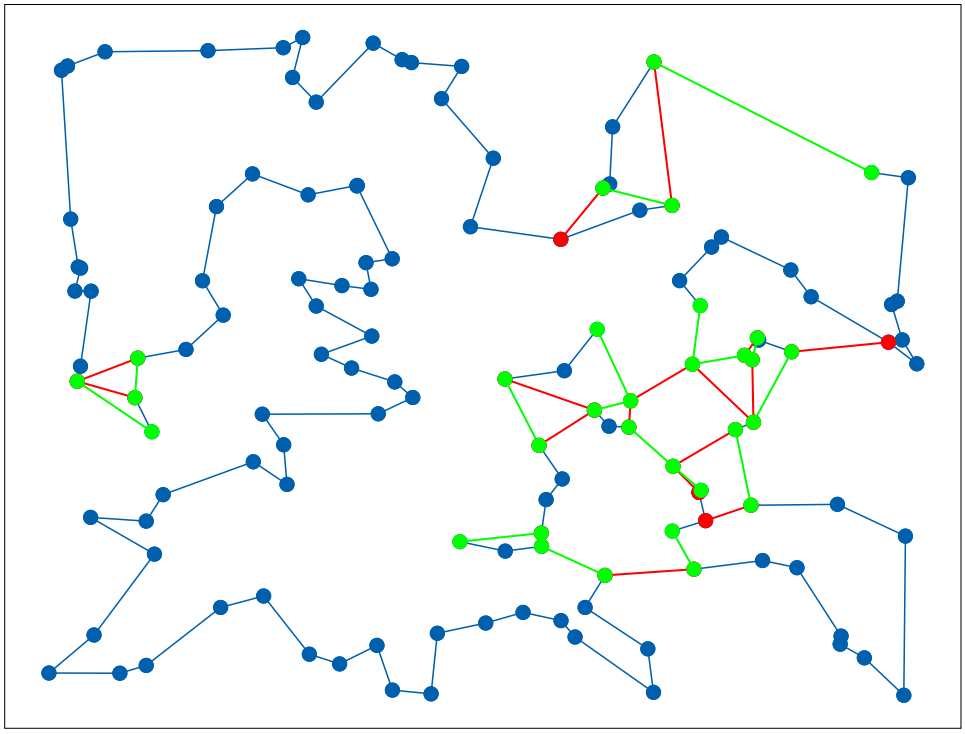
Tabela 4.2: Liczba wspólnych krawędzi otrzymanej trasy z rozwiązaniem optymalnym.

Problem	Czas w [s]	Liczba wspólnych krawędzi z opt. problem [%]
bayg29	0,14	89,66
bays29	0,32	86,21
berlin52	0,00	100,00
eil76	0,20	85,53
kroA100	1,62	86,00
eil101	0,24	83,17
gr120	0,79	85,83
pr124	7,12	89,52
ch130	0,59	85,38
pr136	2,85	77,94
pr152	14,87	89,47
kroA200	1,20	87,00
gr202	0,44	88,12
tsp225	0,87	89,78
pr226	6,10	85,40
a280	0,50	85,36
pr299	1,97	84,62
pr439	2,74	87,24
pcb442	0,58	83,94
ali535	1,01	82,24
gr666	1,47	85,89
pcb1173	0,95	88,49
pr2392	1,20	88,38

Można zauważyć, że liczba krawędzi oscyluje na poziomie powyżej 80%, a w niektórych przypadkach wynosi nawet 90%. W przypadku problemu *berlin52*, otrzymana trasa jest identyczna z rozwiązaniem optymalnym.

Wizualizacja otrzymanego rozwiązania

Rysunek 4.4 przestawia otrzymane rozwiązanie z metody wspinaczki. Zielonym kolorem zaznaczono krawędzie znajdujące się w optimum, a nie występujące w otrzymanym min-1-drzewie. Kolorem czerwonym z kolei oznaczono krawędzie, które występują w min-1-drzewie, a nie występują w optimum problem (krawędzie nadmiarowe). Badanie wykonano dla instancji problemu *ch130* zawierającym 130 wierzchołków.



Rysunek 4.4: Widok min-1-drzewa na przykładzie problemu *ch130*.

Wiele krawędzi nadmiarowych, jak i tych, które nie występują w min-1-drzewie, leżą w swoim bliskim otoczeniu. Jest to wynikiem tego, że min-1-drzewo i każde poprawne rozwiązanie problemu TSP jest spójne. Interesującym problemem

badawczym mogłaby być próba stworzenia algorytmu, który byłby uruchomiony po algorytmie wspinaczki i w sposób deterministyczny modyfikował krawędzie w min-1-drzewie.

Rozdział 5

Heterogeniczny algorytm DPSO

W rozdziale tym znajduje się opis heterogenicznego algorytmu DPSO z feromonem. Jest to modyfikacja, której zadaniem jest z jednej stronie zmniejszyć liczbę parametrów algorytmu, z drugiej poprawić jakość otrzymywanych wyników. W podrozdziale 5.1 znajduje się opis różnych heterogenicznych wariantów algorytmu PSO. W podrozdziale 5.2 znajduje się analiza parametrów homogenicznego algorytmu DPSO z feromonem, przedstawionego w rozdziale 3. Podrozdział 5.3 przedstawia analizę zachowania populacji heterogenicznej, w której wartości parametrów cząsteczek różnią się między sobą. Podrozdział 5.4 zawiera porównanie wyników otrzymanych z wariantu homogenicznego i heterogenicznego. Dwa ostatnie podrozdziały (podrozdział 5.5 i podrozdział 5.6) zawierają analizę złożoności obliczeniowej heterogenicznego wariantu algorytmu DPSO z feromonem oraz jego efektywność dla dużych instancji problemów TSP.

5.1 Heterogeniczność w algorytmie PSO

Ogólnie pojęcie „heterogeniczność" można zdefiniować jako niejednorodność rozumianą jako zróżnicowanie. W algorytmach populacyjnych może przejawiać się ona na wiele sposobów. Taksonomię heterogenicznych wariantów algorytmu PSO przedstawiono w pracy Montes de Oca i in. [103]. Autorzy tej publikacji podzielili heterogeniczność na cztery kategorie.

1. Różnorodność sąsiedztwa dotyczy przypadku, w którym wielkość sąsiedztwa dla każdej cząsteczki jest inna. W takim przypadku topologia połączeń cząsteczek w ramach stada nie jest regularna (wierzchołki mają różne stopnie),

a niektóre cząsteczki mają większy wpływ na pozostałe (cecha wykorzystywana podczas obliczenia następnej pozycji). Regularne topologie komunikacyjne w algorytmie PSO zostały omówione w niniejszej pracy w rozdziale 2.

2. Różnorodność wyboru informatora polega na odmiennym podejściu każdego osobnika do wyboru najlepszej cząsteczki $gBest$, która ma wpływ na jej następną pozycję. Przykładowo jedna cząsteczka może uaktualnić swoją pozycję podążając za najlepszą cząsteczką jedynie w jej sąsiedztwie, podczas gdy inne cząsteczki mogą być w pełni poinformowane, czyli podążać za globalnie najlepszą cząsteczką (najlepszym rozwiązaniem).

3. Różnorodność strategii uaktualnienia pozycji bazuje na grupowaniu cząsteczek pod względem sposobu przeszukiwania przestrzeni rozwiązań – nadając im specjalizację. Przykładowo jedna grupa może eksplorować przestrzeń rozwiązań, z kolei inna grupa wykonywać przeszukiwanie lokalne. Ten rodzaj heterogeniczności najbardziej urozmaica populację, gdyż każda cząsteczka może przeszukiwać przestrzeń rozwiązań w inny sposób.

4. Różnorodność parametrów w stadzie polega na tym, że każda cząsteczka może posiadać parametry inne niż pozostałe cząsteczki w obrębie populacji. W ten sposób niektóre cząsteczki mogą posiadać dużą inercję ω i eksplorować przestrzeń rozwiązań, podczas gdy inne mogą mieć małą jej wartość i lokalnie (wokół własnej najlepszej znalezionej pozycji) szukać optimum globalnego.

Oprócz taksonomii, artykuł [103] zawiera również pełny przegląd literatury dotyczący heterogeniczności w algorytmie PSO do roku 2009. W pracy można znaleźć opis algorytmów dla różnych zastosowań oraz ich porównanie. Niestety w literaturze brakuje pozycji na temat heterogeniczności w dyskretnym algorytmie PSO. Jednak podobieństwo koncepcji działania wersji klasycznej i dyskretnej umożliwia przeniesienie heterogenicznych wariantów klasycznego algorytmu PSO i ich koncepcji działania do wersji dyskretnej.

Heterogeniczność może mieć adaptacyjny charakter. W przypadku, gdy pewne zachowania (heurystyki) pozwalają otrzymać lepsze wyniki, algorytm powinien zwiększyć liczbę osobników z tymi cechami. W odwrotnym wariancie, przy braku dobrych wyników, algorytm może zmniejszyć odpowiednio liczbę osobników z niepożądanymi cechami. Taka strategia może prowadzić do adaptacji ustawień dla danego problemu przy założeniu odpowiednio dużej liczby iteracji. Przedstawiony w tym rozdziale algorytm DPSO należy do ostatniej kategorii w opisanej wyżej taksonomii.

5.2 Analiza parametrów algorytmu

Projektowanie algorytmu heterogenicznego wymaga znajomości wpływu wartości parametrów na działanie algorytmu DPSO z feromonem. W tym celu należało wyznaczyć i zbadać najbardziej charakterystyczne wartości wszystkich parametrów algorytmu DPSO z feromonem. Zakres wartości parametrów jest następujący: $c_1 = [0; 1, 5]$, $c_2, c_3 = [0; 2]$, $\omega = [0; 0, 6]$ [13]. Małe wartości parametrów c_1, c_2, c_3, ω (blisko początkowej granicy zakresu) powodują, że cząsteczka często zmienia krawędzie w kolejnych iteracjach (zmienia swoje położenie). Jest to związane z tym, że wartości parametrów mają bezpośredni wpływ na prawdopodobieństwo wyboru krawędzi należącej do bieżącego rozwiązania do jego kolejnej postaci. Na początku wzmocnienie krawędzi feromonem (zwiększenie prawdopodobieństwa wyboru krawędzi p) jest prawie nieistotne. W kolejnych iteracjach algorytmu wzmocnienie rośnie. Następuje kompensacja małych wartości parametrów wzmocnieniem feromonowym. Dla wartości z końca przedziału wartości, wpływ feromonu nie jest aż tak istotny, gdyż prawdopodobieństwo wyboru krawędzi jest bliskie 1. Przykładowo dla wartości losowej $rand() = 0,5$ i parametru $c_1 = 2$ jest ono równe $0,5 \cdot 2 = 1$. Należy również zauważyć, że jest to maksymalna wartość prawdopodobieństwa wyboru krawędzi. Otrzymanie większej wartości niż 1 spowoduje jej ograniczenie do wartości maksymalnej równej 1. Na podstawie analizy różnych wartości parametrów wybrano najbardziej charakterystyczne. Znajdują się one w tabeli 5.1.

Tabela 5.1: Zestawy charakterystycznych wartości parametrów algorytmu DPSO.

Nr	c_1	c_2	c_3	ω	Opis
1	0,1	0,1	0,1	0,1	szybkie zmiany pozycji
2	2,0	0,1	0,1	0,1	wykorzystanie informacji z $pBest_i$
3	0,1	2,0	0,1	0,1	wykorzystanie informacji z $gBest$
4	0,1	0,1	2,0	0,5	bardzo wolne zmiany pozycji
5	0,75	1,0	1,0	0,25	słabe oddziaływanie $pBest_i$, $gBest$
6	1,25	1,5	1,5	0,5	silniejsze oddziaływanie $pBest_i$, $gBest$
7	1,5	2,0	2,0	0,5	silne oddziaływanie $pBest_i$, $gBest$
8	1,75	2,0	2,0	0,75	bardzo silne oddziaływanie $pBest_i$, $gBest$

Badania przeprowadzono na problemach TSP następująco: na etapie tworzenia populacji początkowej wybierany jest badany zestaw wartości parametrów. Następnie implementacja algorytmu zostaje uruchomiona bez żadnych modyfikacji

Tabela 5.2: Wartości parametrów homogenicznego algorytmu DPSO z feromonem.

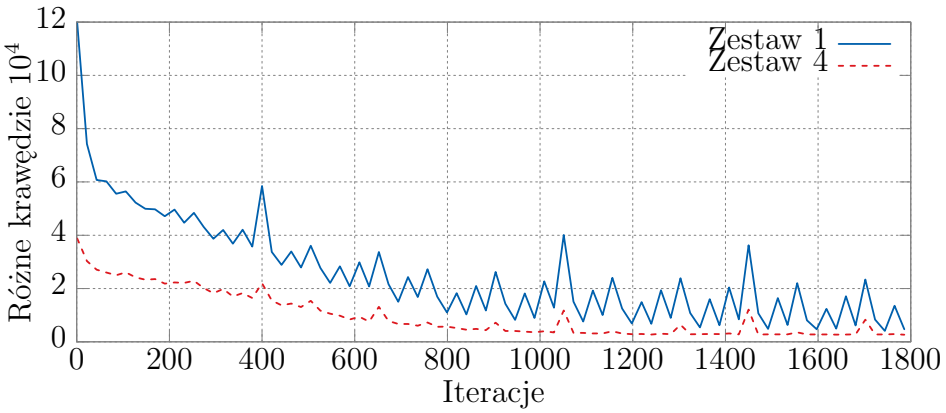
Problemy	Wartości parametrów					
	c_1	c_2	c_3	ω	i_{max}	$\mathcal{N}_{size}$
berlin52	Zgodnie z tabelą 5.1				32	7
kroA100					64	7
kroA200					80	7
gr202					101	10
gr666					112	30

działania. Wszystkie testy powtórzono 30 razy, a wyniki uśredniono. Do badań użyto parametrów przedstawionych w tabeli 5.2.

Rozmiar stada (i_{max}) i sąsiedztwa ($\mathcal{N}_{size}$) zostały wyznaczone na podstawie wpływu tych wartości parametrów na działanie algorytmu. Rozmiar stada wpływa na wydłużenie lub skrócenie czasu obliczeń kosztem poprawy lub pogorszenia jakości otrzymywanych rozwiązań. Z kolei mniejsze sąsiedztwo ogranicza przestrzeń rozwiązań. Zbyt mała jego wielkość może uniemożliwić znalezienie rozwiązania optymalnego (brak krawędzi wchodzących w skład rozwiązania optymalnego w sąsiedztwie wierzchołka). Podane wartości zostały wyznaczone na podstawie wstępnych eksperymentów obliczeniowych, jednak wydaje się, że korzyści płynące z heterogeniczności algorytmu będą zauważalne również dla innych wartości rozmiaru populacji, czy sąsiedztwa. Wszystkie algorytmy użyte w pracy zostały zaimplementowane w języku C# i uruchomione na jednym wątku na maszynie z procesorem Intel i7 3,2 GHz i pamięcią 12 GB.

Rysunek 5.1 przestawia liczbę nowych krawędzi dla zbiorów X_i^{k-1} oraz X_i^k (poprzednia i bieżąca pozycja). Niebieska linia przedstawia wartości parametrów cząsteczek, które często zmieniają krawędzie (zestaw 1), a czerwona dla cząsteczek bardziej stabilnych – rzadsze zmiany (zestaw 4). Numeracja oznaczeń określona jest zgodnie z tabelą 5.1. Pozostałe parametry ustawiono zgodnie z tabelą 5.2. Wartości uśredniono dla 30 uruchomień.

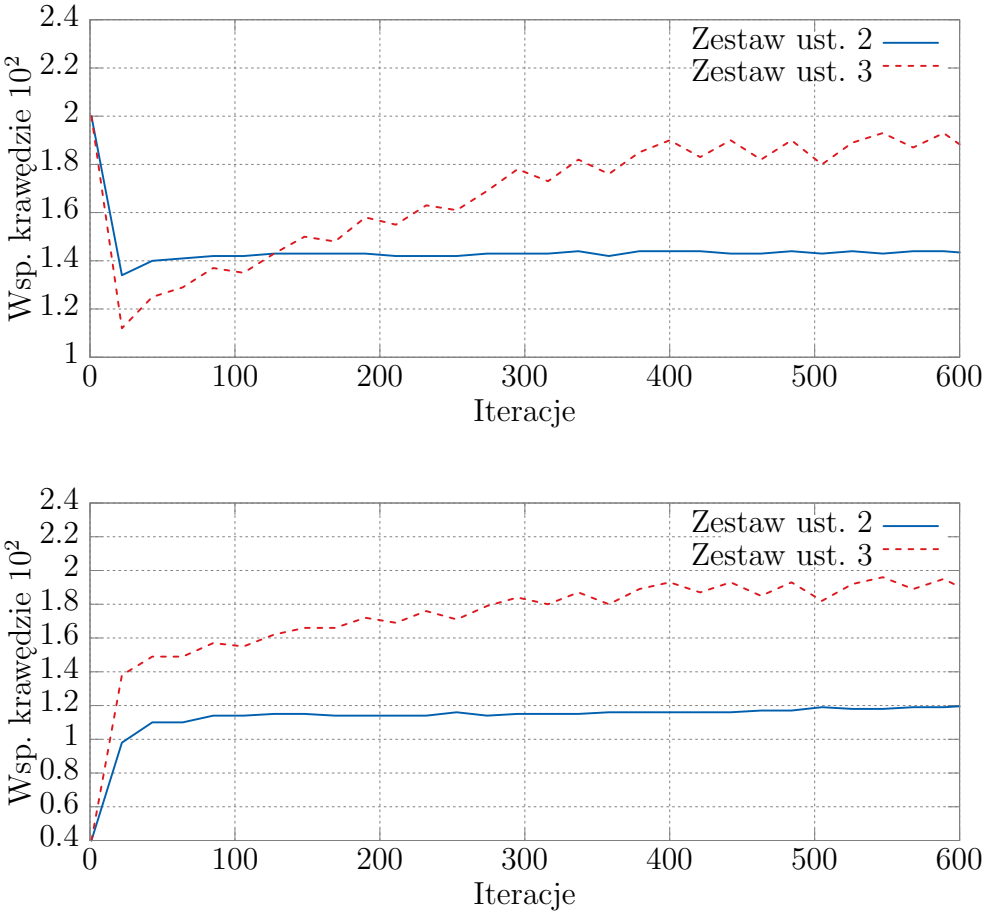
Pierwszy i czwarty zestaw ustawień cząsteczek z Tab. 5.1 różni się dynamiką zmian w liczbie krawędzi dla aktualnej i poprzedniej pozycji cząsteczki. Małe wartości parametrów w przypadku pierwszym powodują, że prawdopodobieństwo wyboru krawędzi do następnej pozycji (p) jest bardzo małe. Jedynie wzmocnienie wirtualnego feromonu może zwiększyć tę wartość. Z kolei w czwartym zestawie ustawień parametr c_3 posiada najwyższą wartość z zakresu. Ustawienie to powoduje, że do nowej pozycji z dużym prawdopodobieństwem będą dodane krawędzie z poprzedniej pozycji. Obie charakterystyki można zaobserwować na Rys. 5.1. Nie-



Rysunek 5.1: Średnia liczba różnych krawędzi między poprzednią i bieżącą pozycją cząsteczki w kolejnych iteracjach heterogenicznego algorytmu DPSO dla problemu *kroA200*.

bieska linia znajduje się ponad linią czerwoną, co oznacza, że pozycja cząsteczki z zestawu pierwszego ma więcej zmian w krawędziach. Przy sumowaniu liczby krawędzi nie uwzględniono współczynnika p oraz kierunku krawędzi, np.: $\langle 0, 2, \{1, 2\}\rangle$ jest równa $\langle 0, 3, \{2, 1\}\rangle$. Rysunek 5.2 przestawia zestawienie dwóch zbiorów ustawień o numerach – 2 i 3. Pierwszy z nich przedstawia liczbę wspólnych krawędzi zbiorów: bieżącej pozycji cząsteczki X_i^k oraz $pBest_i$, drugi bieżącej pozycji X_i^k oraz $gBest$. Numeracja oznaczeń określona jest zgodnie z tabeli 5.1. Pozostałe parametry ustawiono zgodnie z tabelą 5.2. Wartości uśredniono dla 30 uruchomień algorytmu.

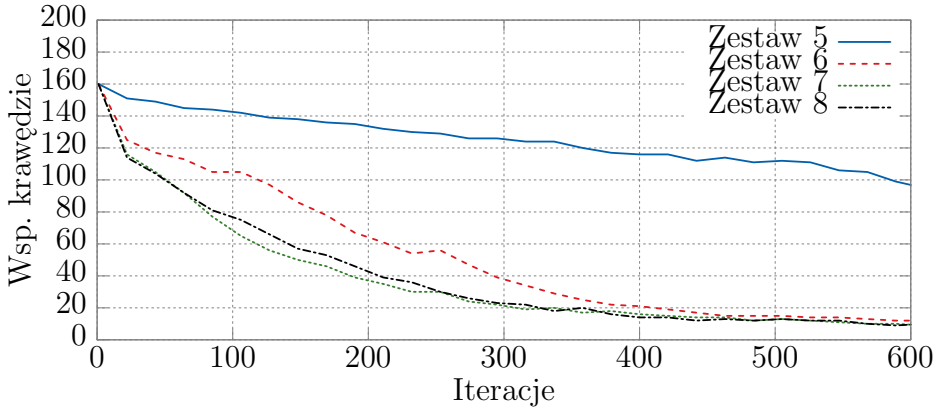
Na pierwszym wykresie znajduje się liczba wspólnych krawędzi dla zbiorów X_i^k i $pBest_i$. Dla zestawu 2 otrzymano większą liczbę wspólnych krawędzi z $pBest_i$ niż dla zestawu 3 z powodu większej wartości parametru c_1 (równego 2). Dotyczy to początkowych iteracji algorytmu. Po 100 iteracjach liczba wspólnych krawędzi zaczyna się zmieniać. W przypadku zestawu numer 3, jest to spowodowane tym, że zbiory $pBest_i$ i $gBest$ zaczynają być do siebie podobne. Wynika to z wysokiej wartości parametru c_2 dla tego zestawu. Drugi wykres przedstawia liczbę wspólnych krawędzi zbiorów X_i^k i $gBest$. Na wykresie tym można zaobserwować rosnące podobieństwo aktualnej pozycji X_i^k w stosunku do najlepszej znalezionej pozycji $pBest_i$. Wynika to ze zbieżności algorytmu do optimum i trudności w znalezieniu lepszych krawędzi (z większym prawdopodobieństwem, że są one optymalne). Efekt ten można zaobserwować dla obu zestawów. Liczba krawędzi wspólnych jest większa dla cząsteczek z trzecim zestawem parametrów, co wynika z największej



Rysunek 5.2: Liczba wspólnych krawędzi zbioru X_i^k ze zbiorami $pBest_i$ i $gBest$ w kontekście różnych zestawów wartości parametrów dla problemu *kroA200* dla pierwszych 600 iteracji.

wartości z zakresu parametru c_1 równej 2. Ostatni wykres (Rys. 5.3) przedstawia łączną liczbę wspólnych krawędzi – $pBest_i$, X_i^k oraz $gBest$, X_i^k. Numeracja oznaczeń określona jest zgodnie z tabelą 5.1. Pozostałe parametry ustawiono zgodnie z tabelą 5.2. Wartości uśredniono dla 30 uruchomień.

Największe różnice można zaobserwować dla zestawów 5 oraz 6. Najmniejszą różnicę odnotowano dla zestawów 7 oraz 8. Wynika to głównie z małej różnicy w wartościach parametrów, która wynosi dla Δc_1=0,25; $\Delta\omega$=0,25 (pozostałe parametry c_2, c_3, mają taką samą wartość).



Rysunek 5.3: Łączna liczba wspólnych krawędzi $pBest_i$ i bieżącej pozycji cząsteczki, $gBest$ i bieżącej pozycji X_i^k dla problemu *kroA200* (bez zmian dla pierwszych 600 iteracji).

Wybranie dowolnych wartości każdego parametru może powodować zbyt chaotyczne przeszukiwanie przestrzeni rozwiązań. Zaproponowany w pracy rozkład prawdopodobieństwa wyboru wartości parametrów bazuje na analizie zachowania cząsteczek. Listę ośmiu charakterystycznych zestawów wartości parametrów znajdujących się w Tab. 5.1 powielono 14 razy, aby otrzymać listę 112 zestawów wartości parametrów (największy analizowany problem *gr666* zawiera taki rozmiar populacji). Na jej podstawie utworzono histogram dla każdego parametru. Parametry c_1, c_2, c_3 posiadały następujące przedziały na histogramie: $\leqslant 0,1$; $\leqslant 0,25$; $\leqslant 0,5$; $\leqslant 0,75$; $\leqslant 1,0$; $\leqslant 1,25$; $\leqslant 1,5$; $\leqslant 1,75$. Wartość 2 została pominięta. Parametr ω posiadał trzy przedziały: $\leqslant 0,1$; $0,25$; $0,5$. Na podstawie częstości występowania danej wartości parametru, uzyskano prawdopodobieństwo wyboru każdego z nich. Przykładowo dla parametru c_1 powstały następujące największe przedziały histogramu: $\leqslant 0,1$ – 42; $\leqslant 0,75$ – 14; $\leqslant 1,5$ – 28; $\leqslant 1,75$ – 14. Na podstawie częstości występowania uzyskano następujący procent (zaokrąglony) wystąpień wartości parametru c_1: 0,1 – 40%; 0,75 – 15%; 1,5 – 30%; 1,75 – 15%. Procentowy udział stanowi jednocześnie prawdopodobieństwo wyboru danej wartości parametru. Powyższą technikę zastosowano do pozostałych parametrów. Tabela 5.3 przedstawia otrzymany dyskretny rozkład prawdopodobieństwa wyboru konkretnej wartości parametru algorytmu DPSO z feromonem.

Przykładowo dla parametru c_1, dla wartości losowej $rand()$ równej 0,3 zostanie ustawiona wartość tego parametru na 0,1.

Tabela 5.3: Dyskretny rozkład prawdopodobieństwa wyboru wartości każdego parametru.

Parametr	Wartość parametru	Prawdopodobieństwo wyboru tej wartości
c_1	0,1	0,4
	0,75	0,15
	1,5	0,3
	1,75	0,15
c_2, c_3	0,1	0,4
	1	0,15
	1,5	0,15
	2	0,3
ω	0,1	0,4
	0,25	0,2
	0,5	0,4

5.3 Analiza populacji heterogenicznej

Badania przeprowadzono na problemach TSP następująco: na początku działania algorytmu, dla każdej cząsteczki w stadzie, losowo wybierane są wartości parametrów na podstawie dyskretnego rozkładu przedstawionego w tabeli 5.3. Następnie implementacja algorytmu zostaje uruchomiona bez dodatkowych modyfikacji działania. Zastosowanie heterogenicznej populacji jest dla algorytmu transparentne poza etapem tworzenia populacji. Wszystkie testy powtórzono 30 razy, a wyniki uśredniono. Do badań użyto parametrów i_{max} oraz $\mathcal{N}_{size}$ przedstawionych w tabeli 5.2.

W pracy użyto dwóch funkcji oceny cząsteczek – ilościowej i jakościowej. Pierwsza z nich określa liczbę ulepszeń najlepszego znalezionego rozwiązania $gBest$. Druga, jakościowa wyznaczana jest na podstawie wzoru (5.1):

$$\begin{aligned} p_{gain} = 0,9 \cdot \sup\left\{ \sum_{\langle a,b\rangle \in gBest} d_{ab} - \sum_{\langle a,b\rangle \in X_i^k} d_{ab}, 0 \right\} \\ + 0,1 \cdot \sup\left\{ \sum_{\langle a,b\rangle \in pBest_i} d_{ab} - \sum_{\langle a,b\rangle \in X_i^k} d_{ab}, 0 \right\}, \end{aligned} \tag{5.1}$$

gdzie: p_{gain} oznacza zysk cząsteczki, pierwszy składnik operacji dodawania to poprawa najlepszego znalezionego rozwiązania przez stado, drugi zaś składnik oznacza jakościową poprawę swojego najlepszego znalezionego rozwiązania. Heteroge-

niczny algorytm DPSO uruchomiono dla problemu *gr666*. Wartości parametrów algorytmu znajdują się w tabeli 5.2.

Celem badań była analiza wpływu różnorodności na zbieżność heterogenicznego algorytmu DPSO w problemie TSP. Tabela 5.4 przedstawia wartości parametrów cząsteczek, dla którego najlepsze rozwiązanie poprawiane było najwięcej razy. Badanie powtórzono 30 razy. Na pierwszych miejscach znalazły się ustawienia cząsteczek, których wartości parametrów c_1, c_2, c_3, ω miały małą wartość. Takie wartości wpływają na prawdopodobieństwo wyboru krawędzi p, umożliwiając eksplorację dużej części przestrzeni rozwiązań. Dopiero na trzecim miejscu znalazła się cząsteczka, która jest bardziej stabilna (wartość parametru c_2 równa 2). Należy również zwrócić uwagę na dużą różnicę między drugim a trzecim miejscem, która wynosi 53. Ranking należy interpretować w kontekście liczby możliwych wystąpień danego parametru. Losowanie nie jest zgodne z rozkładem jednorodnym, a dyskretnym rozkładem przedstawionym w tabeli 5.3. Największe prawdopodobieństwo wyboru parametru c_1 ma wartość $P(c_1 = 0,1) = 0,4$. Wysokie jest również prawdopodobieństwo wyboru $P(c_1 = 1,5) = 0,3$, ale w rankingu występuje bardzo rzadko. Każda cząsteczka ma unikalny zestaw wartości parametrów – c_1, c_2, c_3 i ω (żaden zestaw wartości parametrów się nie powtarza). Łączna liczba wszystkich kombinacji wartości parametrów cząsteczek wynosi $4 \cdot 4 \cdot 4 \cdot 3 = 192$.

Kolejne badania miały na celu określenie, które wartości parametrów najlepiej sprawdzają się w różnych fazach działania implementacji algorytmu. W tym celu podzielono liczbę iteracji algorytmu na równe części i z każdego przedziału wyznaczono te wartości parametrów, które najczęściej poprawiały najlepsze rozwiązanie. Badanie zostało powtórzone 30 razy. Tabela 5.5 przedstawia najlepsze cząsteczki reprezentowane przez wartości parametrów w różnych przedziałach. Na potrzeby analizy tabeli dodano Rys. 5.4, który przedstawia wykres zbieżności do optimum. W celu poprawienia czytelności wykresu usunięto pierwsze 50 wyników (ze względu na bardzo wysokie wartości).

W przypadku tabeli 5.5 rozkład wartości parametrów jest pokazywany wg kolejnych faz działania algorytmu. Przejście do kolejnej fazy algorytmu może pociągać za sobą zmianę wartości parametrów najlepszych cząsteczek. W pierwszej fazie (0–1250) dominują wartości z małą wartością – szybko zmieniające krawędzie w pozycji cząsteczki (rozwiązanie problemu). Jest to faza eksploracji przestrzeni rozwiązań. W trzecim przedziale (2500–3750) wartości parametrów najlepszych cząsteczek są wymieszane – rozpoczyna się faza eksploatacji przestrzeni rozwiązań. W ostatnim przedziale (5000–6144) najlepsze cząsteczki mają większe wartości parametrów. Wysoki modyfikator prawdopodobieństwa wyboru krawędzi wchodzących w skład aktualnej pozycji wraz z dużym wzmocnieniem feromonu powoduje jedynie drobną zmianę aktualnej pozycji.

Ostatnie badanie dotyczy jakościowej oceny najlepszych cząsteczek. Tabela

Tabela 5.4: Ranking skumulowanych najlepszych zestawów wartości parametrów cząsteczek we wszystkich iteracjach heterogenicznego algorytmu DPSO dla jednego podproblemu *gr666*.

Ranking	Parametry				Liczba poprawień $gBest$
	c_1	c_2	c_3	ω	
1	0,1	0,1	0,1	0,5	113
2	0,1	0,1	0,1	0,1	102
3	0,1	2	0,1	0,1	49
4	0,1	2	2	0,5	46
5	0,1	2	2	0,1	42
6	0,1	1,5	0,1	0,5	39
7	0,1	1	0,1	0,1	38
8	0,1	1	0,1	0,25	34
9	0,75	2	2	0,25	27
10	0,1	1	2	0,1	26
11	0,1	2	0,1	0,25	24
12	0,75	2	2	0,1	22
13	1,5	1,5	2	0,5	21
14	1,5	2	0,1	0,1	21
15	1,5	2	0,1	0,25	20

5.6 przedstawia zysk (wzór (5.1)) w ostatnich iteracjach algorytmu dla problemu *gr666*. Małe wartości zysku oznaczają, że najczęściej pochodzi on od poprawy najlepszej pozycji cząsteczki ($pBest_i$), która posiada modyfikator 0,1 (wzór (5.1)). Niemniej jednak, na tym etapie obliczeń pozycje cząsteczek są blisko wartości optymalnej. W kolejnych iteracjach może to spowodować poprawę rozwiązania $gBest$.

5.4 Porównanie wariantów DPSO

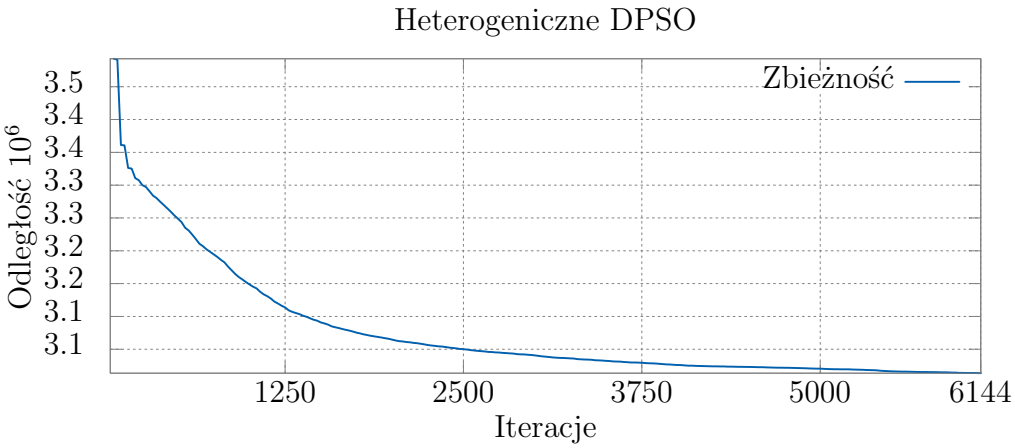
W kolejnym etapie badań porównano dwa warianty algorytmu DPSO dla kilku instancji problemu DTSP oraz jednej TSP. Na potrzeby porównania należało określić wartości parametrów algorytmu homogenicznego. Wartości te zostały ustalone na podstawie próbkowania wartości c_1, c_2, c_3, ω, i_{max} i rozmiaru sąsiedztwa ($\mathcal{N}_{size}$). Dla każdej wartości parametrów uruchomiono implementację algorytmu. Każdy test powtórzono 30 razy. Najlepszą szóstkę dla konkretnej instancji proble-

Tabela 5.5: Cząsteczki z największą liczbą ulepszeń *gBest* wraz z parametrami zgrupowane i zsumowane wg numeru iteracji dla jednego problemu *gr666*.

Iteracje	Parametry				Liczba ulepszeń *gBest*
	c_1	c_2	c_3	ω	
0-1250	0,1	0,1	0,1	0,1	94
	0,1	0,1	0,1	0,5	93
	0,1	2	2	0,5	38
	0,1	2	0,1	0,1	38
	0,1	2	2	0,1	32
1250–2500	0,75	2	2	0,25	12
	1,5	2	0,1	0,1	10
	0,1	2	0,1	0,1	10
	0,1	1,5	0,1	0,5	9
	0,1	1	2	0,1	8
2500–3750	0 ,1	0,1	0,1	0,5	10
	1,5	1,5	2	0,5	4
	1,5	2	0,1	0,25	4
	0,75	2	2	0,25	3
	0,1	1	2	0,1	3
3750–5000	0,1	1	0,1	0,1	2
	0,1	1	2	0,1	2
	0,75	0,1	2	0,5	2
	1,5	0,1	2	0,1	2
	1,5	2	1,5	0,1	2
5000–6144	1,75	2	1	0,5	3
	1,5	2	1,5	0,1	2
	1,75	0,1	2	0,5	2
	0,1	1,5	0,1	0,5	2
	1,5	1	1	0,1	1

mu przedstawia tabela 5.7.

Tabela 5.8 zawiera zestawienie otrzymanych wyników. Porównanie dotyczy dwóch wariantów algorytmu DPSO – homogenicznego (a) i heterogenicznego (b) dla 3% zmian między podproblemami. „T.” oznacza czas działania algorytmu, „G.” oznacza odległość od optimum, „D.” oznacza średnie odchylenie standardowe dla 11 podproblemów. Liczba iteracji dana jest per podproblem. Pogrubioną



Rysunek 5.4: Wykres zbieżności do optimum heterogenicznej wersji algorytmu DPSO dla problemu *gr666*.

Tabela 5.6: Skumulowany zysk grup obliczony na podstawie wzoru (5.1) w ostatnich iteracjach heterogenicznego algorytmu DPSO dla 30 powtórzeń.

Rank.	c_1	c_2	c_3	ω	Zysk grupy	Rank.	c_1	c_2	c_3	ω	Zysk grupy
1	0,1	1	2	0,1	2296	6	0,1	2	2	0,5	830
2	0,1	0,1	2	0,1	1826	7	0,1	0,1	2	0,5	735
3	0,75	0,1	2	0,5	1260	8	0,1	1	2	0,5	682
4	0,75	2	2	0,1	945	9	0,1	0,1	2	0,25	659
5	1,5	1,5	2	0,5	883	10	0,1	2	1,5	0,1	642

czcionką zaznaczono najlepsze rozwiązania znalezione przez jeden z algorytmów DPSO. W przypadku algorytmu homogenicznego wybrane wartości parametrów były zaproponowane w publikacji Boryczka i inni [57].

Tabela 5.7: Wartości parametrów DPSO z feromonem w wersji homogenicznej i heterogenicznej.

Problem	Parametry									
	Homogen. DPSO				Heterogen. DPSO				Wspólne par.	
	c_1	c_2	c_3	ω	c_1	c_2	c_3	ω	i_{max}	$\mathcal{N}_{size}$
berlin52	0,5	0,5	0,5	0,2	Zgodnie z tabelą 5.1				32	7
kroA100	0,5	0,5	0,5	0,5					64	7
kroA200	0,5	0,5	0,5	0,5					80	7
gr202	0,5	0,5	0,5	0,5					101	10
gr666	0,5	1,0	1,5	0,6					112	30

Tabela 5.8: Porównanie dwóch wariantów algorytmu DPSO – homogenicznego i heterogenicznego.

Problem	Iter.	Warianty DPSO						Inne	
		Homogeniczny			Heterogeniczny			ACS	PACO
		T. [s]	G. [%]	D. [%]	T. [s]	G. [%]	D. [%]	G. [%]	G. [%]
berlin52	52	0,1	0,67	46,88	0,1	**0,46**	43,53	-	-
berlin52	104	0,13	0,15	0,32	0,13	**0,13**	0,15	0,96	0,96
berlin52	208	0,19	0,05	8,64	0,18	**0,04**	8,12	0,69	0,69
berlin52	416	0,3	0,01	0,04	0,28	0,01	0,05	0,5	0,5
berlin52	832	0,53	0,01	2,37	0,48	0,01	1,96	0,36	0,36
berlin52	1664	0,98	**0**	0	0,89	0,01	0,05	0,46	0,46
kroA100	50	0,89	11,61	685,86	0,73	**5,82**	483,97	-	-
kroA100	100	1,03	5,44	2,47	0,86	**2,68**	1,4	1,8	2,97
kroA100	200	1,21	2,65	345,75	0,97	**1,64**	246,07	1,61	2,77
kroA100	400	1,63	1,28	1,02	1,27	**1,05**	0,81	1,31	2,13
kroA100	800	2,46	**0,88**	176,29	2,01	0,95	219,41	1,07	1,85
kroA100	1600	4,11	**0,64**	0,69	3,38	0,78	0,77	0,82	1,36
kroA200	80	2,11	19,97	912,98	1,8	**8,1**	609,99	-	-
kroA200	160	2,49	15,63	2,77	2,18	**5,14**	1,84	2,41	3,33
kroA200	320	3,41	9,61	967,99	2,91	**3,1**	329,85	1,99	3,14
kroA200	640	5,13	4,45	1,62	4,46	**2,89**	1,09	1,62	2,71

Problem	Iter.	Warianty DPSO						Inne	
		Homogeniczny			Heterogeniczny			ACS	PACO
		T. [s]	G. [%]	D. [%]	T. [s]	G. [%]	D. [%]	G. [%]	G. [%]
kroA200	1280	8,6	**2,27**	299,95	7,48	2,36	310,76	1,5	2,48
kroA200	2560	15,6	**1,62**	0,81	13,18	2,02	0,8	1,47	2,28
gr202	64	8,36	16,93	751,51	7,71	**8,61**	744,03	-	-
gr202	128	8,82	13,75	2,06	8,17	**4,19**	1,2	6,26	4,91
gr202	256	9,63	10,7	881,15	9,1	**2,61**	313,86	5,29	4,03
gr202	512	11,54	6,81	2,11	10,88	**1,97**	0,66	4,88	3,9
gr202	1024	15	3,14	613,14	14,57	**1,69**	228,77	4,66	3,23
gr202	2048	23,01	**1,52**	0,6	21,98	1,53	0,55	3,93	3,34
pcb442	136	8,27	36,95	2169,38	7,91	**12,54**	1401,82	7,53	5,22
pcb442	272	11,22	29,31	5,33	11,16	**6,73**	1,68	6,18	4,44
pcb442	544	16,75	21,65	2692	17,61	**4,14**	642,5	5,25	4,03
pcb442	1088	28,52	13,41	5	30,69	**2,87**	0,89	4,87	3,56
pcb442	2176	52,9	7,07	1982,77	56,87	**2,41**	486,69	4,16	3,31
pcb442	4352	102,78	3,13	1,52	108,25	**1,92**	0,79	3,91	3,3
gr666	192	78,03	15,89	8186,95	79,9	**13,66**	5006,11	-	-
gr666	384	85,19	10,84	1,52	91,83	**9,58**	0,86	9,18	5,89
gr666	768	98,36	7,37	2938,01	115,19	**6,88**	2283,94	7,46	4,77
gr666	1536	124,84	5,62	0,84	163,48	**5,33**	0,57	6,09	4,51
gr666	3072	180,66	4,88	1864,37	259	**4,52**	2593,57	5,67	4,14
gr666	6144	296,83	3,99	0,77	453,83	**3,8**	0,78	4,92	4,21

Liczba iteracji jest określona dla jednego podproblemu. Łączna liczba iteracji dla całego testu to wartość pomnożona przez 11 (liczba podproblemów). Wyjątkiem jest problem *gr666*, który nie zawiera podproblemów. Jedynie dla najmniejszego problemu (*berlin52*) wyniki są porównywalne. Wynika to z szybkiej zbieżności do optimum obu zaimplementowanych algorytmów. Największą zaobserwowaną różnicę odnotowano dla problemów *pcb442* i *gr202*. Największą zaletą algorytmu heterogenicznego jest fakt, że poprawa rozwiązań jest widoczna wraz z rosnącą liczbą iteracji, co może wskazywać, że algorytm nie ma tendencji do wpadania w optimum lokalne. Średnie odchylenie standardowe (dla 11 podproblemów) jest wyraźnie niższe dla algorytmu heterogenicznego. Mniejsze rozproszenie wyników

dla wielu powtórzeń algorytmu dla tego samego problemu (30 powtórzeń) jest bardzo pożądanym zjawiskiem. Średnie optymalne rozwiązania dla przeanalizowanych problemów DTSP (średnia dla 11 podproblemów) przy zachowaniu kolejności z Tab. 5.8 wynoszą: 7635; 22357,3; 31073; 40582,1; 52497,5; 294358. W żadnym z algorytmów nie stosowano przeszukiwania lokalnego.

5.5 Złożoność obliczeniowa operacji

Dość łatwo określić w algorytmie DPSO z feromonem liczbę ewaluacji rozwiązań (liczbę sprawdzonych rozwiązań). Wartość tą określa wzór: $i_{max} \cdot k_{max}$. Pierwszy czynnik pochodzi od głównej liczby iteracji algorytmu, drugi od rozmiaru populacji. Wzór ten nie uwzględnia wielkości sąsiedztwa oraz rozmiaru problemu (n). Bardzo trudno znaleźć dokładną złożoność algorytmu uzależnioną od n, ze względu na etap dopełnienia krawędzi do pełnego cyklu Hamiltona dla każdej cząsteczki w trakcie określenia jej kolejnej pozycji. Etap ten jest wielostopniowy i stochastyczny co utrudnia analizę algorytmu. Poniżej przedstawiono użyte struktury danych, których zadaniem jest przyspieszenie działania algorytmu.

Operacje i struktury

Dominującą operacją wykonywaną w każdej iteracji dla każdej cząsteczki jest operacja przecięcia dwóch zbiorów. Przyjęto reprezentację cyklu Hamiltona za pomocą tablicy, w której indeks wskazuje na początek krawędzi, a wartość tego indeksu w tabeli wskazuje na drugi koniec krawędzi. Rysunek 5.9 przestawia trasę w zapisie krawędziowym $\{\langle 1,3\rangle,\langle 3,5\rangle,\langle 5,2\rangle,\langle 2,4\rangle,\langle 4,1\rangle\}$.

Tabela 5.9: Reprezentacja rozwiązania w implementacji algorytmu DPSO.

Indeks	1	2	3	4	5
Wartość	3	4	5	1	2

Taka reprezentacja umożliwia wyznaczenie przecięcia dwóch cykli Hamiltona w czasie liniowym. Proces mnożenia elementów zbioru jest połączony z procesem filtracji. Po wyznaczeniu przecięcia zbiorów, np. X z $gBest$, obliczane jest prawdopodobieństwo wyboru każdej krawędzi z wynikowego zbioru przecięcia. Jeśli zmienna losowa z przedziału $[0,1]$ o rozkładzie jednorodnym jest mniejsza od obliczonego prawdopodobieństwa, krawędź staje się kandydatem do kolejnego zbioru pozycji cząsteczki. Najpierw jednak należy sprawdzić, czy któryś wierzchołek w krawędzi nie jest stopnia drugiego. Przechowując tablicę dodanych wierzchołków i ich stopni sprawdzenie stopnia konkretnego wierzchołka jest operacją o czasie stałym $O(1)$.

Kolejny warunek wymaga, aby dodana krawędź nie powodowała cyklu, chyba że jest to brakująca krawędź do utworzenia cyklu Hamiltona. Przykład niepoprawnej trasy znajduje się w tabeli 5.10.

Tabela 5.10: Przykład niepoprawnego rozwiązania problemu TSP.

1	2	3	4	5
2	3	1	5	4

Tabela 5.10 reprezentuje rozwiązanie składające się z dwóch podcykli w zapisie wierzchołkowym $\langle 1, 2, 3, 1\rangle$ oraz $\langle 4, 5\rangle$. Efektywność tej operacji ma decydujące znaczenie dla czasu obliczeń. Przyjęto następującą metodę: dodawana krawędź jest dołączona do istniejącej na liście sekwencji krawędzi lub tworzona jest nowa sekwencja. Każda z tych sekwencji zawiera pierwszy i ostatni wierzchołek w sekwencji. Przykładowo, niech lista sekwencji będzie pusta. Następnie dodawana zostanie krawędź $\langle 1, 2\rangle$. Nowa sekwencja zostaje dodana do listy i przyjmuje postać:

$$(\langle 1, 2\rangle)\,.$$

Następnie niech zostanie dodana kolejna krawędź $\langle 3, 4\rangle$. Na liście nie istnieje żadna sekwencja zawierająca wierzchołek 3 lub 4. Z tego powodu powstaje kolejna, druga sekwencja na liście, która po dodaniu kolejnego elementu przyjmuje postać:

$$(\langle 1, 2\rangle, \langle 3, 4\rangle)\,.$$

Niech kolejna krawędź będzie postaci $\langle 2, 5\rangle$. Ponieważ krawędź 2 istnieje na liście sekwencji, zostaje ona powiększona, lista będzie postaci:

$$(\langle 1, 5\rangle, \langle 3, 4\rangle)\,.$$

Jeśli wierzchołek 2, będzie występował w kolejnej krawędzi, nie zostanie dodany ze względu na swój stopień równy 2. Z tego powodu nie ma potrzeby pamiętać o pośrednich wierzchołkach (wewnątrz sekwencji). Należy również zaznaczyć, że dwie sekwencje mogą zostać połączone, np. krawędzią $\langle 5, 3\rangle$. W takim przypadku lista przyjmuje postać:

$$(\langle 1, 4\rangle)\,.$$

Nigdy jednak nie powstanie podcykl, gdyż algorytm sprawdza, czy w ramach jednej sekwencji wierzchołki krawędzi nie występują dwukrotnie, np. biorąc pod uwagę ostatnią postać listy, dodawana krawędź $\langle 1, 4\rangle$ lub $\langle 4, 1\rangle$ jest niepoprawna. Chyba, że jest to brakująca, ostatnia krawędź zamykająca cykl Hamiltona. To ostatnie można sprawdzić w czasie stałym $O(1)$ posługując się licznikiem dodanych krawędzi. Po fazie filtracji powstałe rozwiązanie należy dopełnić do cyklu Hamiltona

(faza dopełnienia). Czas przeglądu listy można skrócić posługują się tablicą, w której indeks wskazuje wierzchołek, a wartość na indeks elementu sekwencji w liście. Dla listy:

$$(\langle 1,2\rangle, \langle 3,5\rangle, \langle 6,8\rangle)\,,$$

tablica może przyjąć postać Tab. 5.11.

Tabela 5.11: Tablica indeksująca listę $(\langle 1,2\rangle, \langle 3,5\rangle, \langle 6,8\rangle)$.

wierzchołek	1	2	3	4	5	6	7	8
indeks elementu	1	1	2	0	2	3	0	3

Wartość zero oznacza, że wierzchołek nie jest użyty lub jest użyty dwa razy. Należy zauważyć, że w przypadku dodawania kolejnego wierzchołka i w wyniku tego, scalaniu elementów, odpowiednie wierzchołki należy wyzerować. Wszystkie operacje można wykonać w czasie stałym.

Etap dopełnienia wykonywany jest przez dwa różne algorytmy: funkcję przejścia znaną z algorytmów mrowiskowych oraz najbliższego sąsiada. Pierwszy realizowany jest następująco: dla każdego wierzchołka sprawdzany jest jego stopień. Jeśli jest on mniejszy, niż 2 odkładane są na listę wszystkie krawędzie, których początek stanowi bieżący wierzchołek oraz każdy wierzchołek należący do jego sąsiedztwa. Następnie dla każdego wierzchołka obliczane jest prawdopodobieństwo wyboru krawędzi (na podstawie funkcji przejścia (2.14)). Metodą ruletki wybierana jest krawędź, która zostanie dodana do kolejnego rozwiązania cząsteczki. Dla wierzchołka o stopniu 1, algorytm przechodzi do kolejnego wierzchołka. W przypadku wierzchołka o stopniu 0, wybierana jest kolejna krawędź dla bieżącego wierzchołka (każdy wierzchołek musi dwukrotnie pojawić się na liście krawędzi). W algorytmie najbliższego sąsiada procedura uproszczona jest do wybrania pierwszego, najbliższego sąsiada. W obu omówionych metodach przed dodaniem krawędzi do kolejnego rozwiązania cząsteczki, sprawdzana jest poprawność rozwiązania – czy nie tworzy subcykli. Dlatego może się zdarzyć, że żaden wierzchołek w najbliższym sąsiedztwie nie pozwoli na stworzenie krawędzi, która po dodaniu do bieżącego rozwiązania utworzy poprawny cykl Hamiltona. Wtedy sąsiedztwo rozszerza się na wszystkie wierzchołki. Przykładowo mając listę postaci:

$$(\langle 1,2\rangle, \langle 3,5\rangle, \langle 6,8\rangle)\,,$$

zostanie obliczone prawdopodobieństwo wyboru wierzchołka z sąsiedztwa wierzchołka 2. Jeśli będzie to wierzchołek 3, dodana zostanie krawędź $\langle 2,3\rangle$.

5.6 Efektywność algorytmu heterogenicznego

O efektywności dowolnego algorytmu dla dynamicznego problemu TSP decyduje nie tylko adaptacja do zmian, ale również efektywność optymalizacji w kontekście statycznego problemu TSP (podproblemu). W ramach badań uruchomiono heterogeniczny algorytm DPSO z feromonem. Użyto ustawień przedstawionych w tabeli 5.12. Ze względu na wielkość instancji problemów, implementacja algorytmu została uruchomiona z przeszukiwaniem lokalnym co 50 iteracji (algorytm 3-optymalny). Ze względu na długi czas obliczeń. Przedstawionych parametrów nie dobrano na postawie żadnej analizy wyników. Są to parametry uniwersalne.

Tabela 5.12: Wartości parametrów heterogenicznego algorytmu DPSO z feromonem.

Problem	Rozmiar problemu (n)	Liczba iteracji (k_{max})	Rozmiar stada (i_{max})	Wielkość sąsie. ($\mathcal{N}_{size}$)
rat783	783	$n \cdot 10$	100	30
vm1084	1084	$n \cdot 10$	100	30
pcb1173	1173	$n \cdot 10$	100	30
d1291	1291	$n \cdot 10$	100	30
fl1400	1400	$n \cdot 10$	100	30
fl1577	1577	$n \cdot 10$	100	30
d1655	1655	$n \cdot 10$	100	30
vm1748	1748	$n \cdot 10$	100	30
rl1889	1889	$n \cdot 10$	100	30
pr2392	2392	$n \cdot 15$	100	30
pcb3038	3038	$n \cdot 15$	100	30
fl3795	3795	$n \cdot 15$	100	30
pla7397	7397	$n \cdot 15$	100	30

Wielkość problemu (liczba wierzchołków) determinuje wielkość przestrzeni rozwiązań. Z tego powodu liczba iteracji została uzależniona od liczby wierzchołków. Przyjęto dwa mnożniki, dla rozmiaru instancji poniżej 2000 wierzchołków i powyżej tej liczby. Dwa pozostałe parametry – wielkość stada i wielkość sąsiedztwa są stałe. Tabela 5.13 przedstawia wyniki działania instancji algorytmu uruchomionego za pomocą ustawień z tabeli 5.12. Badania powtórzono 30 razy dla instancji poniżej 3000 wierzchołków, 10 razy dla instancji poniżej 7000 wierzchołków, a dla największej instancji 5 razy, wszystkie wyniki uśredniono.

Tabela 5.13: Wyniki działania heterogenicznego algorytmu DPSO z feromonem dla wybranych instancji z TSPLIB.

Problem	Czas [s]	Odl. od opt. [%]	Odchylenie standardowe
rat783	474,51	1	19,42
vm1084	973,74	0,22	301,51
pcb1173	1139,41	0,9	133,32
d1291	1474,99	0,41	147,93
fl1400	1804,25	0,37	29,78
fl1577	2290,77	0,15	8,99
d1655	2767,53	0,66	161,33
vm1748	2852,54	0,36	511,66
rl1889	3763,45	0,43	659,87
pr2392	8335,23	0,95	739,98
pcb3038	13986,75	1,2	285,44
fl3795	23970,2	1,05	58,33
pla7397	156986,71	0,77	14914,52

Wszystkie wyniki dla instancji mniejszej niż 2392 wierzchołków są równe lub poniżej 1% optimum. Nie zaobserwowano korelacji miedzy jakością otrzymanych wyników, a liczbą wierzchołków w tym przedziale. Jedynie w przypadku czasu obliczeń taka korelacja występuje ze względu na czas ewaluacji rozwiązania oraz liczbę iteracji (uzależnioną od rozmiaru problemu).

Podsumowanie

Problem komiwojażera ma znaczenie nie tylko teoretyczne (wiele problemów kombinatorycznych da się sprowadzić do problemu TSP), ale również praktyczne – szczególnie w transporcie, z którego się wywodzi. Asumptem do powstania dynamicznej wersji problemu są względy praktyczne. Niejednokrotnie może zdarzyć się korek na drodze, skutkiem czego trasa ulega wydłużeniu. Odległość między wierzchołkami może oznaczać nie tylko dystans, ale również, np. czas, czy pewien poniesiony koszt. Dzięki temu zakres zastosowań problemu statycznego i dynamicznego TSP znacząco się powiększa. W pracy, dynamiczny problem komiwojażera zdefiniowano jako sekwencję następujących po sobie statycznych problemów komiwojażera (podproblemów). Różnią się one od siebie pewnym procentem zmian w macierzy odległości. Pomimo dużej liczby prac poświęconych problemowi statycznemu, jak i dynamicznemu, wciąż wiele pytań pozostaje otwartych. Szczególnie w dynamicznej wersji.

Cechą charakterystyczną algorytmu PSO jest inspiracja fenomenem ruchu stada, np. ptaków, jaki zaobserwować można w środowisku naturalnym. Pierwotnie technika nie była wykorzystywana praktycznie, a prezentowana jako algorytm „proof of concept". Liczne badania i udoskonalenia doprowadziły do tego, że w niektórych dziedzinach wyparła ona tzw. klasyczne podejścia i niejednokrotnie została użyta w praktyce. Niemniej jednak, pomimo licznych publikacji na temat algorytmu PSO, trudno znaleźć dyskretny wariant algorytmu, który bezpośrednio pozwalałby na adaptację do dynamicznego problemu komiwojażera. Praca stanowiła próbę adaptacji algorytmu DPSO do rozwiązywania problemu DTSP. Nowa wersja algorytmu bazowała na dyskretnej wersji algorytmu DPSO Zhonga i in., który jest przystosowany do rozwiązywania problemu TSP. W celu adaptacji tej techniki do dynamicznej wersji problemu zastosowano macierz feromonową znaną z algorytmów mrowiskowych. Wirtualny feromon pełnił różnorakie role w algorytmie, odpowiadał, m.in. za transfer wiedzy między podproblemami oraz miał przyspieszyć zbieżność algorytmu do optimum.

Teza pracy „Zastosowanie pamięci feromonowej oraz heterogeniczności warto-

ści parametrów w dyskretnym algorytmie PSO dla dynamicznego problemu komiwojażera pozwala poprawić jakość otrzymywanych wyników" została potwierdzona na podstawie wyników eksperymentów obliczeniowych poddanych wstępnie analizie statystycznej. Zrealizowane zostały następujące cele pracy (w kolejności chronologicznej):

1. Sporządzono przegląd literatury w kontekście tematyki związanej z problemami DTSP i TSP, co można znaleźć w rozdziale 1. Dodatkowo zrealizowano przegląd literatury dla algorytmów inteligencji obliczeniowej ze szczególnym uwzględnieniem algorytmu PSO i jego wariantów (rozdział 2 i 3). Przeprowadzona analiza wykazała braki w tej materii, uzasadniając tym samym konieczność podjęcia badań nad tą tematyką.

2. W rozdziale 1 formalnie zdefiniowano dynamiczny problem komiwojażera oraz środowisko testowe. W literaturze wiele pozycji nie zawiera odpowiednich formalizmów, a zmienność środowiska traktowana jest dowolnie. Takie podejście prowadzi do trudności w obiektywnej ocenie jakości wyników otrzymanych z implementacji testowanego algorytmu. Środowisko użyte w pracy składa się z wygenerowanych wcześniej sekwencji problemów, które mogą zostać wczytane przez dowolny program. Za pomocą algorytmów dokładnych dla TSP uzyskano optima każdego podproblemu. Dodatkowo zamieszczono wizualizację przedstawiającą zmiany między kolejnymi podproblemami z podkreśleniem nowych krawędzi optimum. Podejście zastosowane w pracy, gwarantuje, że wszystkie otrzymane wyniki z implementacji algorytmów zostały otrzymane dla tych samych danych wejściowych.

3. Na podstawie przeglądu literatury został wybrany wariant algorytmu DPSO, a następnie zaproponowano jego nową wersję z feromonem do rozwiązywania dynamicznego problemu komiwojażera, co zostało przedstawione w rozdziale 3.

4. Przybliżono rolę feromonu w kontekście problemu DTSP oraz sposobie jego adaptacji do istniejącego algorytmu DPSO. W trakcie prac nad algorytmem dla dynamicznego problemu komiwojażera należało rozwiązań szereg problemów m.in. związanych z wirtualnym feromonem. Rozdział 3 przedstawia pełną analizę algorytmu DPSO z feromonem oraz szczegółowo opisuje jego schemat działania.

5. Zaprezentowano również liczne badania związane z charakterystyką działania algorytmu, w tym: analizę eksploracji i eksploatacji algorytmu w różnych fazach jego działania, rozproszenia feromonu w kontekście adaptacji do kolejnych podproblemów, podobieństwa populacji do najlepszego znalezionego

rozwiązania ($gBest$) liczonego na podstawie przecięć zbiorów, zmiany macierzy feromonowej w kolejnych iteracjach algorytmów, na co wypływ ma proces parowania wirtualnego feromonu oraz znajdowanie coraz lepszych rozwiązań. Asumptem do badań była próba poznania procesów zachodzących w algorytmie, co zostało pokazane w rozdziale 3.

6. W pracy, proponowany algorytm został porównany z innymi algorytmami inteligencji obliczeniowej: populacyjny algorytm mrówkowy (PACO) oraz algorytm mrówkowy (ACO).

7. Nowy algorytm używa ograniczonej listy sąsiedztwa dla każdego wierzchołka. Zabieg ten redukuje przegląd przestrzeni rozwiązań i tym samym poprawia zbieżność algorytmu. Ma to swoje wady wśród których należy wymienić, to że w przypadku braku krawędzi należącej do rozwiązania optymalnego w sąsiedztwie wierzchołka prawdopodobieństwo znalezienia rozwiązania optymalnego jest bardzo małe. W pracy zastosowano sąsiedztwo Helsgauna, którego opis znajduje się w rozdziale 4.

8. W pracy dużo miejsca poświęcono badaniu wpływu różnych wartości parametrów na działanie algorytmu DPSO z feromonem, co było asumptem do powstania algorytmu heterogenicznego. W algorytmie tym każda cząsteczka może posiadać inne wartości parametrów. Jednak zupełna ich dowolność może prowadzić do chaotycznego przeszukiwania przestrzeni rozwiązań. Dlatego dobrano taki rozkład prawdopodobieństw wyboru konkretnych wartości parametrów, aby temu zapobiec. Poprzedzała je analiza charakterystycznych wartości algorytmu DPSO z feromonem. Różnorodność wartości parametrów poprawiła jakość otrzymanych wyników. Jednak głównym celem powstania wersji heterogenicznej była redukcja parametrów algorytmu wymagających podania przed uruchomieniem algorytmu. Ostatecznie parametry ograniczono do liczby iteracji (i_{max}), wielkości stada (k_{max}) oraz rozmiaru sąsiedztwa ($\mathcal{N}_{size}$). Są to parametry, których wartości powinny być wyznaczane na podstawie rozmiaru problemu (n). Ich wybór ogranicza się do budżetu obliczeniowego, gdyż dwa pierwsze wpływają na czas obliczeń. Trzeci parametr wpływa na stopień eksploracji i eksploatacji przestrzeni rozwiązań.

9. Efektywna implementacja algorytmu DPSO z feromonem jest zadaniem trudnym, głównie z powodu etapu dopełnienia zbioru krawędzi do cyklu Hamiltona. W pracy przedstawiono najważniejsze kwestie implementacyjne algorytmu DPSO z feromonem, dotyczy to użytych struktur danych oraz sposobu ich przetwarzania.

10. Zbadano dodatkowo efektywność algorytmu DPSO z feromonem dla dużych

instancji TSP od 783 do 7397 wierzchołków.

W wyniku przedstawionych w pracy badań możliwe było wysunięcie następujących wniosków:

1. Zaobserwowano na podstawie badań, że głównym czynnikiem, który wpływa na zysk związany z wykorzystaniem feromonu jest liczba zmian między podproblemami. Czym większa była to liczba, tym mniejszy zysk odnotowano w badaniach. Jednym z celów pracy było pokazanie, że feromon dla optymalizacji po zmianach danych wejściowych (macierzy odległości) poprawia jakość otrzymywanych wyników. Dzięki niemu proponowane rozwiązanie nie rozpoczyna swojego działania od przeszukiwania pełnej przestrzeni rozwiązań, a skupia się na poprawie najbardziej obiecujących rozwiązań, co znacząco wpływało na odległość od optimum ostatecznie otrzymanego rozwiązania. W badaniach wykazano, że cel ten został zrealizowany.

2. Liczność zmian między podproblemami określa ich podobieństwo w sekwencji, co przekładało się na różnice w przestrzeni rozwiązań. Wykorzystanie tej informacji pozwoliło na otrzymanie lepszych wyników, co zostało wykazane w pracy.

3. Zastosowanie wirtualnego feromonu poprawiło zbieżność algorytmu do optimum, zarówno dla wersji statycznej jak i dynamicznej. W obu problemach (TSP i DTSP). Wersja algorytmu z feromonem pozwalała otrzymać lepszy wynik, niż wariant algorytmu bez feromonu (w przeważającej liczbie przypadków), co zostało potwierdzone wynikami testów statystycznych w rozdziałach 3, 5).

4. Wirtualny feromon za sprawą parowania (zmniejszenia wartości feromonu odłożonego na krawędzi) oraz wzmocnienia (dla krawędzi wchodzących w skład najlepszego znalezionego rozwiązania) okazał się być bardzo elastycznym mechanizmem, który dość szybko adaptował się do zmian w przestrzeni rozwiązań (dla kolejnego podproblemu), co udowodniły przeprowadzone badania.

5. Dodanie zróżnicowania w wartościach parametrów dla każdego osobnika ma pozytywny wpływ na proces optymalizacji. Wersja heterogeniczna algorytmu DPSO z feromonem pozwalała otrzymać lepsze wyniki od wersji homogenicznej. Może to być efektem tego, że jak pokazały badania, różne wartości parametrów są przydatne na różnych etapach działania algorytmu. Proces ten jest ściśle związany z fazą eksploracji i eksploatacji przestrzeni rozwiązań.

6. Heterogeniczny algorytm DPSO z feromonem dla dużych instancji problemów TSP, zaczerpniętych z biblioteki TSPLIB pozwalał otrzymać w przeważającej liczbie rozwiązania o długości poniżej 1% optimum (ostatni podrozdział rozdziału 5).

7. Porównując wyniki otrzymane dla algorytmów heterogenicznego i dwóch innych popularnych algorytmów inteligencji obliczeniowej ACO i PACO, można stwierdzić, że zaproponowany w pracy algorytm uzyskał lepsze wyniki.

Oprócz wniosków związanych z algorytmem DPSO z feromonem możliwe było także zaobserwowanie pewnych zależności:

1. Trudność problemu należy rozumieć w kontekście odległości optimów lokalnych i globalnych od siebie, a nie tylko jako efekt liczby wierzchołków. W wielu przypadkach zaobserwowano otrzymanie gorszych wyników dla zmodyfikowanych problemów, w porównaniu do oryginalnych danych z biblioteki TSPLIB.

2. W każdym teście DTSP zaobserwowano, że nawet niewielkie zmiany w macierzy odległości powodują zmianę optymalnej trasy komiwojażera, co uzasadnia podjęcie tematyki pracy.

Przeprowadzone badania oraz ich wyniki mogą stanowić punkt wyjścia do kolejnych badań:

1. Istotnym kierunkiem badawczym może być poszukiwanie rozwiązań najbardziej odpornych na zmiany, tj. leżących relatywnie blisko najkrótszej trasy komiwojażera, ale których sąsiednie rozwiązania (pod względem długości) są relatywnie daleko. Ma to fundamentalne znaczenie w przypadku transportu priorytetowego, np. krwi lub organów. Przy założeniu, że jedynym kryterium jest długość trasy lub czas jej pokonania, może się okazać, że jeden z odcinków jest zablokowany lub zamknięty, a pozostałe objazdy znacznie wydłużą całą trasę. Innym aspektem tego samego problemu jest unikanie tzw. wąskich gardeł przy projektowaniu głównych węzłów w miastach. Przykładowo takie podejście pozwala na budowanie takich połączeń między ważnymi punktami miasta, że zamknięcie jednej drogi nie spowoduje paraliżu komunikacyjnego. Oba zagadnienia są częścią zagadnienia podatności rozwiązań znajdujących się w przestrzeni rozwiązań na zmiany i teorii tolerancji.

2. Kolejnym istotnym kierunkiem badań może być zastosowanie różnorakich strategii przy uruchomieniu algorytmu dla kolejnego podproblemu. Jedną z nich jest częściowy reset macierzy feromonowej zamiast jej kopiowania,

bądź resetowania do wartości domyślnych. Najczęściej strategie te są omawiane w literaturze w kontekście algorytmu ACO. Ze względu na efektywność algorytmu DPSO z feromonem przedstawionym w tej pracy, naturalnym kierunkiem kolejnych badań wydaje się zaimplementowanie najbardziej obiecujących strategii.

3. W pracy zastosowano wariant problemu DTSP, w którym liczność wierzchołków w kolejnych podproblemach była stała. Dodatkowo każdy badany problem był symetryczny i metryczny. Ostatnie ograniczenie wynikało jedynie z biblioteki TSPLIB, nie konstrukcji samego algorytmu. Rozwinięcie badań na problemy niesymetryczne jest kolejnym punktem dalszych badań. Wiąże się z nim również rozszerzenie środowiska testowego na problemy niesymetryczne.

Spis algorytmów

Spis tabel

Spis rysunków

Bibliografia

[1] Marco Dorigo and Luca Maria Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem". In: *IEEE Transactions on evolutionary computation* 1.1 (1997), pp. 53–66.

[2] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. "Distributed optimization by ant colonies". In: *Proceedings of the first European conference on artificial life*. Vol. 142. 1991, pp. 134–142.

[3] Marco Dorigo. "Optimization, learning and natural algorithms". In: *Ph. D. Thesis, Politecnico di Milano, Italy* (1992).

[4] Christian Blum. "Ant colony optimization: Introduction and recent trends". In: *Physics of Life Reviews* 2.4 (2005), pp. 353–373. ISSN: 1571-0645.

[5] Marco Dorigo and Christian Blum. "Ant colony optimization theory: A survey". In: *Theoretical Computer Science* 344.2 (2005), pp. 243–278. ISSN: 0304-3975.

[6] Marco Dorigo and Thomas Stützle. "Ant colony optimization: overview and recent advances". In: *Handbook of metaheuristics*. Springer, 2010, pp. 227–263.

[7] Keld Helsgaun. *An effective implementation of k-opt moves for the Lin-Kernighan TSP heuristic*. Tech. rep. Roskilde University, 2006.

[8] Keld Helsgaun. "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic". In: *European Journal of Operational Research* 126 (2000), pp. 106–130.

[9] Thomas Stützle and Holger H Hoos. "MAX–MIN Ant System". In: *Future generation computer systems* 16.8 (2000), pp. 889–914.

[10] Marco Dorigo and Thomas Stützle. "Ant colony optimization: overview and recent advances". In: *Techreport, IRIDIA, Universite Libre de Bruxelles* (2009).

[11] Michael Guntsch and Martin Middendorf. "A population based approach for ACO". In: *Applications of Evolutionary Computing*. Springer, 2002, pp. 72–81.

[12] Rafał Skinderowicz. "Implementing Population-Based ACO". In: *Computational Collective Intelligence. Technologies and Applications*. Springer International Publishing, 2014, pp. 603–612.

[13] Wen-liang Zhong, Jun Zhang, and Wei-neng Chen. "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems". In: *Evolutionary Computation, 2007. CEC 2007*. Vol. 14. IEEE, 1997, pp. 3283–3287.

[14] Rainer E Burkard et al. "Well-solvable special cases of the traveling salesman problem: a survey". In: *SIAM review* 40.3 (1998), pp. 496–546.

[15] Guy Theraulaz and Eric Bonabeau. "A brief history of stigmergy". In: *Artificial life* 5.2 (1999), pp. 97–116.

[16] Marco Dorigo, Mauro Birattari, and Thomas Stützle. "Ant Colony Optimization – Artificial Ants as a Computational Intelligence Technique". In: *IEEE COMPUT. INTELL. MAG* 1 (2006), pp. 28–39.

[17] Plerre-P Grassé. "La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs". In: *Insectes sociaux* 6.1 (1959), pp. 41–80.

[18] Francis Heylighen. "Stigmergy as a universal coordination mechanism I: Definition and components". In: *Cognitive Systems Research* 38 (2016), pp. 4–13.

[19] Leslie Marsh and Christian Onof. "Stigmergic epistemology, stigmergic cognition". In: *Cognitive Systems Research* 9.1 (2008), pp. 136–149.

[20] U. Boryczka. *Algorytmy optymalizacji mrowiskowej*. Prace Naukowe Uniwersytetu Śląskiego w Katowicach. Wydawn. Uniw. Śląskiego, 2006. ISBN: 9788322615850.

[21] O. Olorunda and A.P. Engelbrecht. "Measuring exploration/exploitation in particle swarms using swarm diversity". In: *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*. June 2008, pp. 1128–1134.

[22] David L. Applegate et al. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007. ISBN: 0691129932, 9780691129938.

[23] N. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory, 1736-1936*. New York, NY, USA: Clarendon Press, 1986. ISBN: 0-198-53916-9.

[24] Alexander Schrijver. "On the History of Combinatorial Optimization (Till 1960)". In: *Discrete Optimization*. Ed. by G.L. Nemhauser K. Aardal and R. Weismantel. Vol. 12. Handbooks in Operations Research and Management Science. Elsevier, 2005, pp. 38–46.

[25] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990, p. 199. ISBN: 0716710455.

[26] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2.

[27] Frank Harary and Edgar M Palmer. *Graphical enumeration*. New York: Elsevier, 1973. Chap. Digraphs, pp. 120–134.

[28] Chris Godsil and Gordon Royle. *Algebraic graph theory, volume 207 of Graduate Texts in Mathematics*. 2001.

[29] William Cook. *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press, 2012. ISBN: 9780691163529.

[30] Eli S. Marks. "A Lower Bound for the Expected Travel Among m Random Points". In: *Ann. Math. Statist.* 19.3 (Sept. 1948), pp. 419–422.

[31] Roy Jonker and Ton Volgenant. "Transforming asymmetric into symmetric traveling salesman problems". In: *Operations Research Letters* 2.4 (1983), pp. 161–163. ISSN: 0167-6377.

[32] Nicos Christofides. *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem*. Management sciences research report. Defense Technical Information Center, 1976.

[33] Sanjeev Arora. "Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems". In: *J. ACM* 45.5 (Sept. 1998), pp. 753–782. ISSN: 0004-5411.

[34] Jack Rosenberger. "GÖDel Prize and Other CS Awards". In: *Commun. ACM* 53.8 (Aug. 2010), pp. 21–21. ISSN: 0001-0782.

[35] David Applegate et al. "TSP Cuts Which Do Not Conform to the Template Paradigm". In: *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*. Ed. by Michael Jünger and Denis Naddef. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 261–303. ISBN: 978-3-540-45586-8.

[36] David Applegate, William Cook, and André Rohe. "Chained Lin-Kernighan for Large Traveling Salesman Problems". In: *INFORMS J. on Computing* 15.1 (Jan. 2003), pp. 82–92. ISSN: 1526-5528.

[37] Willliam Cook. *Sweden Computation Log*. June 2013. URL: http://www.tsp.gatech.edu/world/swlog.html.

[38] David L. Applegate et al. "Certification of an optimal TSP tour through 85,900 cities". In: *Operations Research Letters* 37.1 (2009), pp. 11–15. ISSN: 0167-6377.

[39] Harilaos N Psaraftis. "Dynamic Vehicle Routing problems". In: *Vehicle routing: Methods and studies* 16 (1988), pp. 223–248.

[40] Changhe Li, Ming Yang, and Lishan Kang. "A new approach to solving Dynamic Traveling Salesman problems". In: *Proceedings of the 6th international conference on Simulated Evolution And Learning*. SEAL'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 236–243. ISBN: 3-540-47331-9, 978-3-540-47331-2.

[41] Michael Guntsch, Martin Middendorf, and Hartmut Schmeck. "An Ant Colony Optimization Approach to Dynamic TSP". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. GECCO'01. San Francisco, California: Morgan Kaufmann Publishers Inc., 2001, pp. 860–867. ISBN: 1-55860-774-9. URL: http://dl.acm.org/citation.cfm?id=2955239.2955396.

[42] Abdunnaser Younes, Paul Calamai, and Otman Basir. “Generalized Benchmark Generation for Dynamic Combinatorial Problems”. In: *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation*. GECCO ’05. Washington, D.C.: ACM, 2005, pp. 25–31. DOI: 10.1145/1102256.1102262. URL: http://doi.acm.org/10.1145/1102256.1102262.

[43] Michalis Mavrovouniotis, Shengxiang Yang, and Xin Yao. “A Benchmark Generator for Dynamic Permutation-Encoded Problems”. In: *Parallel Problem Solving from Nature - PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part II*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 508–517. ISBN: 978-3-642-32964-7.

[44] Abdunnaser Younes, Otman Basir, and Paul Calamai. “A benchmark generator for dynamic optimization”. In: *Proceedings of the 3rd International Conference on Soft Computing, Optimization, Simulation & Manufacturing Systems*. 2003.

[45] Michał Okulewicz and Jacek Mańdziuk. “Application of particle swarm optimization algorithm to dynamic vehicle routing problem”. In: *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2013, pp. 547–558.

[46] Yonca Erdem Demirtaş, Erhan Özdemir, and Umut Demirtaş. “A particle swarm optimization for the dynamic vehicle routing problem”. In: *Modeling, Simulation, and Applied Optimization (ICMSAO), 2015 6th International Conference on*. IEEE. 2015, pp. 1–5.

[47] Mostepha Redouane Khouadjia, Laetitia Jourdan, and El-Ghazali Talbi. “Adaptive particle swarm for solving the dynamic vehicle routing problem”. In: *Computer Systems and Applications (AICCSA), 2010 IEEE/ACS International Conference on*. IEEE. 2010, pp. 1–8.

[48] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. “A survey of swarm intelligence for dynamic optimization: Algorithms and applications”. In: *Swarm and Evolutionary Computation* 33.Supplement C (2017), pp. 1–17. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2016.12.005.

[49] Michael Guntsch and Martin Middendorf. “Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP”. In: *Applications of Evolutionary Computing*. Vol. 2037. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, pp. 213–222.

[50] Casper Joost Eyckelhof and Marko Snoek. “Ant Systems for a Dynamic TSP”. In: *Ant Algorithms: Third International Workshop, ANTS 2002 Brussels, Belgium, September 12–14, 2002 Proceedings*. Ed. by Marco Dorigo, Gianni Di Caro, and Michael Sampels. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 88–99. ISBN: 978-3-540-45724-4.

[51] Michalis Mavrovouniotis and Shengxiang Yang. “Ant Colony Optimization with Immigrants Schemes in Dynamic Environments”. English. In: *Parallel Problem Solving from Nature, PPSN XI*. Vol. 6239. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 371–380.

[52] Anabela Simões and Ernesto Costa. “CHC-Based Algorithms for the Dynamic Traveling Salesman Problem”. English. In: *Applications of Evolutionary Computation*. Vol. 6624. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 354–363.

[53] Camelia-Mihaela Pintea, Gloria Cerasela Crisan, and Mihai Manea. “Parallel ACO with a Ring Neighborhood for Dynamic TSP”. In: *JITR* 5.4 (2012), pp. 1–13. DOI: 10.4018/jitr.2012100101. URL: http://dx.doi.org/10.4018/jitr.2012100101.

[54] Renato Tinós, Darrell Whitley, and Adele Howe. “Use of Explicit Memory in the Dynamic Traveling Salesman Problem”. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO ’14. Vancouver, BC, Canada: ACM, 2014, pp. 999–1006. ISBN: 978-1-4503-2662-9.

[55] Yitong Zhang and Gang Zhao. “Research on Multi-service Demand Path Planning Based on Continuous Hopfield Neural Network”. In: *Proceedings of China Modern Logistics Engineering: Inheritance, Wisdom, Innovation and Cooperation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 417–430. ISBN: 978-3-662-44674-4.

[56] Jayne Eaton, Shengxiang Yang, and Michalis Mavrovouniotis. "Ant colony optimization with immigrants schemes for the dynamic railway junction rescheduling problem with multiple delays". In: *Soft Computing* 20.8 (2016), pp. 2951–2966. ISSN: 1433-7479. DOI: 10.1007/s00500-015-1924-x. URL: http://dx.doi.org/10.1007/s00500-015-1924-x.

[57] Urszula Boryczka and Łukasz Strąk. "Diversification and Entropy Improvement on the DPSO Algorithm for DTSP". In: *Intelligent Information and Database Systems: 7th Asian Conference, ACIIDS 2015, Bali, Indonesia, March 23-25, 2015, Proceedings, Part I*. Ed. by Ngoc Thanh Nguyen, Bogdan Trawiński, and Raymond Kosala. Springer International Publishing, 2015, pp. 337–347. ISBN: 978-3-319-15702-3.

[58] T. Cheong and C.C. White III. "Dynamic traveling salesman problem: Value of real-time traffic information". In: *IEEE Transactions on Intelligent Transportation Systems* 13.2 (2012). cited By 16, pp. 619–630. DOI: 10.1109/TITS.2011.2174050. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84861919836&doi=10.1109%2fTITS.2011.2174050&partnerID=40&md5=1c951ce711daefad9c8f19a0441932d6.

[59] R. Tinós. "Analysis of the dynamic traveling salesman problem with weight changes". In: *2015 Latin America Congress on Computational Intelligence (LA-CCI)*. Oct. 2015, pp. 1–6. DOI: 10.1109/LA-CCI.2015.7435936.

[60] A. Larsen, O.B.G. Madsen, and M.M. Solomon. "Recent developments in dynamic vehicle routing systems". In: *Operations Research/Computer Science Interfaces Series* 43 (2008). cited By 35, pp. 199–218. DOI: 10.1007/978-0-387-77778-8_9. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84888588363&doi=10.1007%2f978-0-387-77778-8_9&partnerID=40&md5=01ee57d6743b1e766c5c318d8a64fe74.

[61] Y. Bilu and N. Linial. "Are stable instances easy?" In: *Combinatorics Probability and Computing* 21.5 (2012), pp. 643–660.

[62] Farhan Ahammed and Pablo Moscato. "Evolving L-Systems as an Intelligent Design Approach to Find Classes of Difficult-to-Solve Traveling Salesman Problem Instances". In: *Applications of Evolutionary Computation: EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, To-*

rino, Italy, April 27-29, 2011, Proceedings, Part I. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–11. ISBN: 978-3-642-20525-5.

[63] Joseph Czyzyk, Michael P. Mesnier, and Jorge J. Moré. "The NEOS Server". In: *IEEE Journal on Computational Science and Engineering* 5.3 (1998), pp. 68–75.

[64] Elizabeth D. Dolan. *The NEOS Server 4.0 Administrative Guide*. Technical Memorandum ANL/MCS-TM-250. Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

[65] William Gropp and Jorge J. Moré. "Optimization Environments and the NEOS Server". In: *Approximation Theory and Optimization*. Ed. by Martin D. Buhman and Arieh Iserles. Cambridge University Press, 1997, pp. 167–182.

[66] Judith Korb. "Thermoregulation and ventilation of termite mounds". In: *Naturwissenschaften* 90.5 (2003), pp. 212–219. ISSN: 1432-1904.

[67] Pascal Jouquet, Daniel Tessier, and Michel Lepage. "The soil structural stability of termite nests: role of clays in Macrotermes bellicosus (Isoptera, Macrotermitinae) mound soils". In: *European Journal of Soil Biology* 40.1 (2004), pp. 23–29. ISSN: 1164-5563.

[68] Craig W Reynolds. "Flocks, herds and schools: A distributed behavioral model". In: *ACM SIGGRAPH computer graphics* 21.4 (1987), pp. 25–34.

[69] Frank Heppner and Ulf Grenander. "A stochastic nonlinear model for coordinated bird flocks". In: *The ubiquity of chaos* (1990), pp. 233–238.

[70] James Kennedy and Russell Eberhart. "Particle Swarm Optimization". In: *In Proceedings of the IEEE International Conference on Neural Networks* (1995), pp. 1942–1948.

[71] A. P. Engelbrecht. "Computational Intelligence: An Introduction: Second Edition". English. In: Computational Intelligence: An Introduction: Second Edition. 2007, pp. 1–597. ISBN: 978-0-470-03561-0.

[72] P. Cichosz. *Systemy uczące się*. Wydawnictwa Naukowo-Techniczne, 2007. ISBN: 9788320433104.

[73] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006. ISBN: 0470091916.

[74] Riccardo Poli, James Kennedy, and Tim Blackwell. "Particle swarm optimization". In: *Swarm Intelligence* 1.1 (2007), pp. 33–57. ISSN: 1935-3820.

[75] Yuhui Shi and Russell Eberhart. "A modified particle swarm optimizer". In: *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE. 1998, pp. 69–73.

[76] Russell Eberhart and Yuhui Shi. "Comparing inertia weights and constriction factors in particle swarm optimization". In: *Proceedings of the 2000 Congress on Evolutionary Computation*. Vol. 1. 2000, pp. 84–88.

[77] James Kennedy and Russell C. Eberhart. "A Discrete Binary Version of The Particle Swarm Algorithm". In: *PROC. OF CONF. ON SYSTEM, MAN, AND CYBERNETICS*. 1997, pp. 4104–4109.

[78] Y. Zhang et al. "Binary PSO with mutation operator for feature selection using decision tree applied to spam detection". English. In: *Knowledge-Based Systems* 64 (2014), pp. 22–31. ISSN: 09507051.

[79] L.-Y. Chuang et al. "Improved binary PSO for feature selection using gene expression data". In: *Computational Biology and Chemistry* 32.1 (2008), pp. 29–37. ISSN: 14769271.

[80] A. Unler and A. Murat. "A discrete particle swarm optimization method for feature selection in binary classification problems". In: *European Journal of Operational Research* 206.3 (2010), pp. 528–539. ISSN: 03772217. DOI: 10.1016/j.ejor.2010.02.032.

[81] Y.-W. Jeong et al. "A new quantum-inspired binary PSO: Application to unit commitment problems for power systems". In: *IEEE Transactions on Power Systems* 25.3 (2010), pp. 1486–1495. ISSN: 08858950.

[82] Jean-Louis Deneubourg, Jacques M Pasteels, and Jean-Claude Verhaeghe. "Probabilistic behaviour in ants: a strategy of errors?" In: *Journal of Theoretical Biology* 105.2 (1983), pp. 259–271.

[83] Simon Goss et al. "Self-organized shortcuts in the Argentine ant". In: *Naturwissenschaften* 76.12 (1989), pp. 579–581.

[84] Urszula Boryczka and Łukasz Strąk. “Efficient DPSO Neighbourhood for Dynamic Traveling Salesman Problem”. English. In: *Computational Collective Intelligence. Technologies and Applications*. Vol. 8083. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 721–730. ISBN: 978-3-642-40494-8. URL: http://dx.doi.org/10.1007/978-3-642-40495-5_72.

[85] Urszula Boryczka and Łukasz Strąk. “A hybrid discrete particle swarm optimization with pheromone for dynamic traveling salesman problem”. In: *Proceedings of the 4th international conference on Computational Collective Intelligence: technologies and applications - Volume Part II*. ICCCI’12. Ho Chi Minh City, Vietnam: Springer-Verlag, 2012, pp. 503–512. ISBN: 978-3-642-34706-1.

[86] Dirk Richter et al. “Improving the efficiency of Helsgaun’s Lin-Kernighan Heuristic for the symmetric TSP”. In: *Proceedings of the 4th conference on Combinatorial and algorithmic aspects of networking*. CAAN’07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 99–111. ISBN: 3-540-77293-6, 978-3-540-77293-4.

[87] Marek Libura et al. “Stability aspects of the traveling salesman problem based on k-best solutions”. In: *Discrete Applied Mathematics* 87.1–3 (1998), pp. 159–185. ISSN: 0166-218X.

[88] Marek Libura. “Sensitivity analysis for minimum Hamiltonian path and traveling salesman problems”. In: *Discrete Applied Mathematics* 30.2 (1991), pp. 197–211. ISSN: 0166-218X.

[89] Gerold Jäger, Boris Goldengorin, and Panos M. Pardalos. “The Theory of Set Tolerances”. In: *Learning and Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers*. Ed. by Panos M. Pardalos et al. Cham: Springer International Publishing, 2014, pp. 362–377. ISBN: 978-3-319-09584-4.

[90] Boris Goldengorin, Gerold Jäger, and Paul Molitor. “Tolerances Applied in Combinatorial Optimization”. In: *J. COMPUT. SCI* 2.9 (2006), pp. 716–734.

[91] Gerold Jäger. “The Theory of Tolerances with Applications to the Traveling Salesman Problem”. In: Habilitation Thesis, Christian Albrechts Universität Kiel (2011), p. 240.

[92] M. Held and R. M. Karp. “The traveling-salesman problem and minimum spanning trees”. In: *Operations Research* 18 (1970), pp. 1138–1162.

[93] Michael Held and Richard M. Karp. “The traveling-salesman problem and minimum spanning trees: Part II”. In: *Mathematical Programming* 1.1 (1971), pp. 6–25. ISSN: 1436-4646.

[94] Michel X. Goemans and Dimitris J. Bertsimas. “Probabilistic Analysis of the Held and Karp Lower Bound for the Euclidean Traveling Salesman Problem”. In: *Mathematics of Operations Research* 16.1 (1991), pp. 72–89.

[95] Christine L. Valenzuela and Antonia J. Jones. “Estimating the Held-Karp lower bound for the geometric {TSP}”. In: *European Journal of Operational Research* 102.1 (1997), pp. 157–175. ISSN: 0377-2217.

[96] D. S. Johnson, L. A. McGeoch, and E. E. Rothberg. “Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound”. In: *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’96. Atlanta, Georgia, USA: Society for Industrial and Applied Mathematics, 1996, pp. 341–350. ISBN: 0-89871-366-8.

[97] Andreas Westerlund, Maud Göthe-Lundgren, and Torbjörn Larsson. “A note on relatives to the Held and Karp 1-tree problem”. In: *Operations Research Letters* 34.3 (2006), pp. 275–282. ISSN: 0167-6377.

[98] T. H. C. Smith. “Combinatorial Optimization”. In: ed. by M. W. Padberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1980. Chap. A LIFO implicit enumeration algorithm for the asymmetric travelling salesman problem using a one-arborescence relaxation, pp. 108–114. ISBN: 978-3-642-00802-3.

[99] Ton Volgenant and Roy Jonker. “A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation”. In: *European Journal of Operational Research* 9.1 (1982), pp. 83–89. ISSN: 0377-2217.

[100] Ton Volgenant and Roy Jonker. “The symmetric traveling salesman problem and edge exchanges in minimal 1-trees”. In: *European Journal of Operational Research* 12.4 (1983), pp. 394–403. ISSN: 0377-2217.

[101] P. M. E. Shutler. “An Improved Branching Rule for the Symmetric Travelling Salesman Problem”. In: *The Journal of the Operational Research Society* 52.2 (2001), pp. 169–175. ISSN: 01605682, 14769360.

[102] Boris T Polyak. “A general method of solving extremum problems”. In: *Doklady Akademii Nauk SSSR* 174.1 (1967), p. 33.

[103] Marco A. Montes de Oca et al. “Heterogeneous Particle Swarm Optimizers”. In: *IEEE Congress on Evolutionary Computation (CEC 2009)*. Ed. by Pauline Haddow et al. Piscatway, NJ: IEEE Press, 2009, pp. 698–705.