POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK INFORMATYKA

Praca dyplomowa magisterska

Rozwiązanie problemu sekwencyjnego porządkowania z użyciem algorytmu mrówkowego

Autor: Wojciech Grzenia

Kierujący pracą: dr inż. Jacek Widuch

Gliwice, wrzesień 2014

Spis treści

1. WSTĘP

Mianem algorytmu mrówkowego nazywa się zbiór pewnych charakterystycznych działań stosowanych w celu rozwiązania problemu poprzez szukanie dróg w grafach. Działania te zostały zainspirowane przez zachowanie mrówek, które poprzez pozostawianie za sobą śladu feromonowego tworzą swego rodzaju pamięć grupową. Pierwszy algorytm mrówkowy został zaproponowany przez Marco Dorigo w pracy doktorskiej, która została opublikowana w 1992 roku [12]. Algorytmy mrówkowe nie są algorytmami z punktu widzenia klasycznej definicji, ponieważ zawierają element losowy, a zatem nie spełniają założenia o jednoznaczności. Nie dają również gwarancji znalezienia najlepszego rozwiązania. Umożliwiają jednak wydajne przeczesywanie przestrzeni możliwych rozwiązań w celu znalezienia stosunkowo dobrego rozwiązania w ograniczonym czasie.

Wiele zadań można przedstawić za pomocą grafów, dlatego też algorytmy mrówkowe znajdują zastosowanie w szerokiej gamie problemów, na przykład harmonogramowanie, poszukiwanie partycji czy wyznaczanie drogi przejazdu. Do zobrazowania działania tych algorytmów często wykorzystuje się problem komiwojażera, polegający na znalezieniu jak najkrótszej, zamkniętej ścieżki łączącej wszystkie węzły grafu. Podobnym zagadnieniem jest sekwencyjne porządkowanie, gdzie celem jest odnalezienie jak najkrótszej ścieżki od węzła początkowego do węzła końcowego, biorąc pod uwagę ograniczenia dotyczące kolejności odwiedzania kolejnych wierzchołków. W ten sposób można zamodelować wiele rzeczywistych zagadnień, jak na przykład planowanie produkcji [13], planowanie trasy przejazdu dla pojazdu dostawczego [14] czy planowanie transportu w elastycznych systemach produkcji [15].

W momencie pisania niniejszej pracy najlepszym znanym algorytmem mrówkowym przystosowanym do rozwiązywania problemu sekwencyjnego porządkowania jest *Enhanced Ant Colony System* (EACS) [7]. Porównywalne wyniki osiąga algorytm *EigenAnt* [16], który po raz pierwszy został przedstawiony w pracy [17]. Istnieją również algorytmy *Probabilistic Enhanced Ant Colony System* i *EigenAnt Ant System* przedstawione w pracy [9], które dają lepsze wyniki od EACS dla pewnej specyficznej grupy problemów.

Celem niniejszej pracy jest ulepszenie algorytmu EACS w taki sposób, aby w tym samym czasie przetwarzania osiągał on lepsze wyniki. Elementem

motywującym są szerokie możliwości wykorzystania tego algorytmu w rzeczywistych problemach, jak również fakt, że jest to jeden z najlepszych istniejących algorytmów wykorzystywanych do rozwiązywania problemów sekwencyjnego porządkowania.

Praca składa się z sześciu rozdziałów. W rozdziale drugim przedstawiono podstawy teoretyczne, których znajomość jest przydatna, a czasami konieczna do dobrego zrozumienia wielu aspektów poruszanej problematyki. Rozdział trzeci zawiera dokładny opis problemu sekwencyjnego porządkowania oraz algorytmu EACS. W rozdziale czwartym zawarto informacje o szczegółach implementacji tego algorytmu. Znajduje się tam też opis wprowadzonych modyfikacji wraz z motywacją zastosowania każdej z nich. W rozdziale piątym przedstawiono wyniki badań eksperymentalnych oraz płynące z nich wnioski. Rozdział szósty zawiera podsumowanie przeprowadzonych prac badawczych.

2. PODSTAWY TEORETYCZNE

2.1. WYBRANE ZAGADNIENIA Z TEORII GRAFÓW[1]

W niniejszym podrozdziale przedstawione zostaną podstawowe pojęcia z teorii grafów. Graf jest powszechnie stosowanym sposobem reprezentacji wielu problemów.

Graf skończony $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ składa się ze skończonego i niepustego zbioru wierzchołków $\mathrm{V} = \{\mathrm{v}_1, \mathrm{v}_2, \dots, \mathrm{v}_\mathrm{n}\}$ i zbioru krawędzi $\mathrm{E} = \{\mathrm{e}_1, \mathrm{e}_2, \dots, \mathrm{e}_\mathrm{m}\}$. Każdej krawędzi e odpowiada pewna para wierzchołków (v, w), gdzie v i w należą do zbioru V. Przykładowy graf skończony znajduje się na rysunku 2.1.

Graf skierowany to graf, którego każda para wierzchołków jest parą uporządkowaną. O krawędzi e mówi się wtedy, że jest ona skierowana od wierzchołka v do wierzchołka w. Jeżeli graf nie jest grafem skierowanym, to para wierzchołków jest nieuporządkowana i zapisy (w, v) i (v, w) oznaczają tę samą krawędź. Przykładowy graf skierowany znajduje się na rysunku 2.1.

Sąsiedztwo wierzchołków występuje pomiędzy wierzchołkami v i w gdy istnieje krawędź $e \in E$, która zawiera te wierzchołki. Dla grafu z rysunku 2.1 wierzchołkami sąsiadującymi z wierzchołkiem v_1 są: v_2, v_6, v_7, v_8.

Stopień wierzchołka to liczba wierzchołków z nim sąsiadujących. Dla grafu z rysunku 2.1 stopień wierzchołka v_1 wynosi cztery.

Droga w grafie jest ciągiem krawędzi $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)$. Często stosowaną reprezentacją drogi jest ciąg wchodzących w jej skład wierzchołków $v_1, v_2, \dots, v_n$. Przykładem drogi dla grafu z rysunku 2.1 może być następujący ciąg wierzchołków: v_1, v_2, v_8, v_3.

Sumą grafów $G_1 = (V_1, E_1)$ i $G_2 = (V_2, E_2)$, gdzie zbiory V_1 i V_2 są rozłączne, jest graf, którego zbiorem wierzchołków jest zbiór $V_1 \cup V_2$, a zbiorem krawędzi zbiór $E_1 \cup E_2$. Przykład sumy grafów został zaprezentowany na rysunku 2.2.

Graf spójny to graf, którego nie można przedstawić w postaci sumy dwóch grafów. Przykładem grafu spójnego jest graf znajdujący się na rysunku 2.1.

Pętla jest krawędzią łączącą wierzchołek ze samym sobą. Przykładem pętli dla grafu z rysunku 2.1 jest krawędź (v_5, v_5).

---

[1] Do opracowania wykorzystano pozycje [1], [2], [3] i [4].

Krawędzie wielokrotne to krawędzie łączące te same wierzchołki. Dla grafu skierowanego krawędzie te muszą mieć ten sam zwrot.

Graf prosty to graf w którym nie występują krawędzie wielokrotne oraz pętle.

Graf pełny to graf prosty w którym każda para różnych wierzchołków jest połączona krawędzią.

Graf regularny to graf w którym każdy wierzchołek ma ten sam stopień.

Stopniem grafu regularnego nazywa się stopień wierzchołków należących do tego grafu.

Graf cykliczny to graf spójny i regularny stopnia drugiego.

Długością drogi nazywa się liczbę należących do niej krawędzi, za wyjątkiem pętli. W szczególnym przypadku pojedynczy wierzchołek oznacza drogę długości zero z tego wierzchołka do niego samego. Dla grafu z rysunku 2.1 długość drogi v_1, v_2, v_8, v_3 wynosi trzy.

Droga elementarna to droga, której wszystkie krawędzie i wierzchołki są różne. Wyjątkiem jest droga elementarna zaczynająca i kończąca się w tym samym wierzchołku. W takiej sytuacji jedynym wierzchołkiem, który może się powtórzyć, jest wierzchołek pierwszy będący zarazem wierzchołkiem ostatnim. Przykładem drogi elementarnej dla grafu z rysunku 2.1 jest droga v_1, v_2, v_8, v_3.

Cykl elementarny to droga elementarna zaczynająca i kończąca się w tym samym wierzchołku, której długość jest większa od zera. Przykładem cyklu elementarnego dla grafu z rysunku 2.1 jest droga elementarna v_1, v_8, v_7, v_1.

Cykl Hamiltona to cykl elementarny, który zawiera wszystkie wierzchołki grafu. Cyklem Hamiltona dla grafu z rysunku 2.1 jest cykl elementarny $v_1, v_2, v_8, v_3, v_4, v_5, v_7, v_6, v_1$.

Ścieżka Hamiltona to droga elementarna, która zawiera wszystkie wierzchołki grafu. Przykładem ścieżki Hamiltona dla grafu z rysunku 2.1 jest droga elementarna $v_1, v_2, v_8, v_3, v_4, v_5, v_7, v_6$.

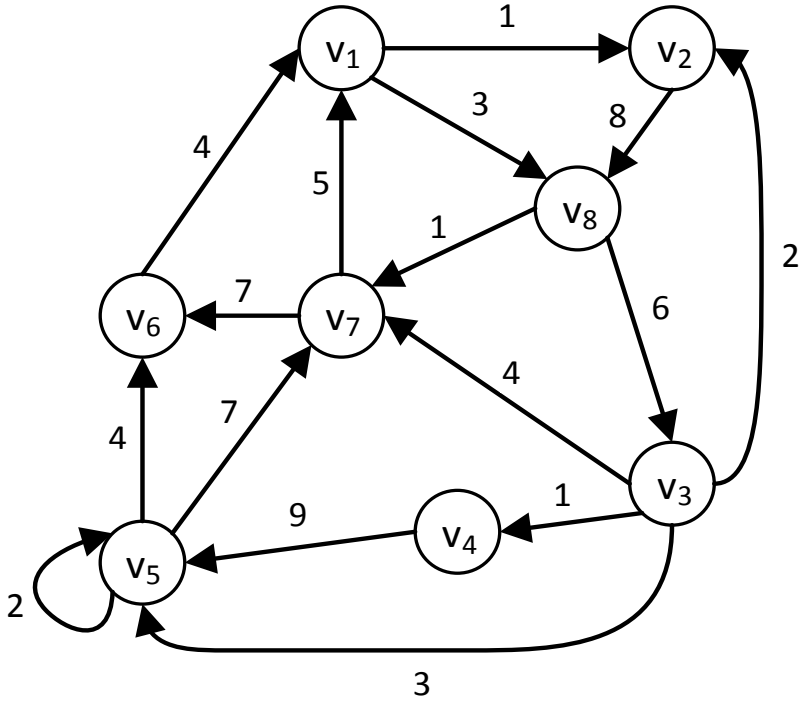
Cykl Eulera to droga, które zawiera wszystkie wierzchołki grafu oraz zaczyna i kończy się w tym samym wierzchołku. Przykładem dla grafu z rysunku 2.1 jest droga $v_1, v_8, v_3, v_2, v_8, v_3, v_4, v_5, v_7, v_6, v_1$.

Waga krawędzi to wartość liczbowa przypisana do krawędzi. Graf zawierający wagi jest rozszerzony o funkcję $f: E \to R$ taką, że dla każdej krawędzi

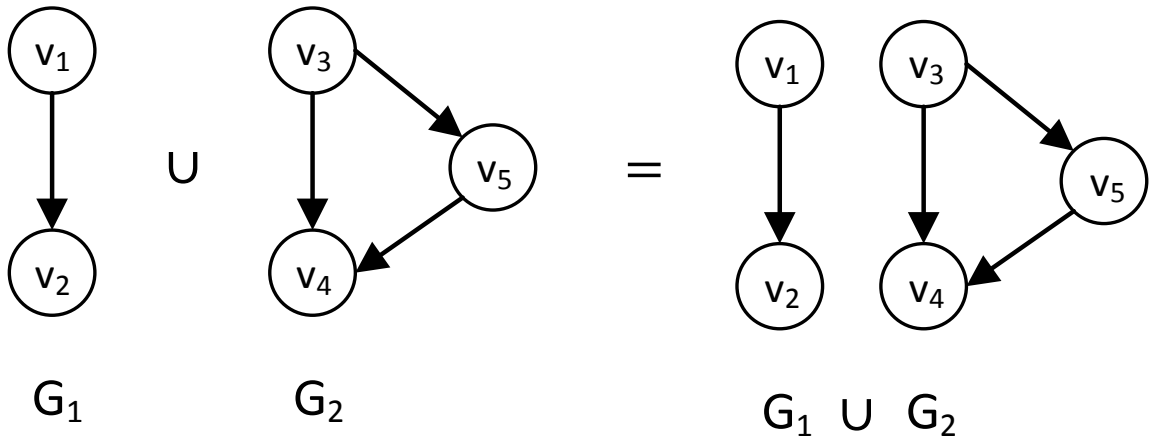
$e \in E$, $f(e)$ jest wagą danej krawędzi. Funkcja ta odzwierciedla pewne właściwości problemu opisanego przez graf. Wagi krawędzi często stosuje się do odzwierciedlenia kosztów związanych z przejściem pomiędzy wierzchołkami. Dla grafu z rysunku 2.1 waga krawędzi (v_1, v_8) wynosi trzy.

Waga drogi to suma wag przypisanych do kolejnych elementów ciągu krawędzi tworzącego drogę. Dla grafu z rysunku 2.1 waga drogi v_1, v_2, v_8, v_3 wynosi piętnaście.

Istnieje wiele sposobów reprezentowania grafów. Najprostszy z nich polega na odzwierciedleniu grafu na rysunku. W tym celu stosuje się podpisane okręgi symbolizujące węzły oraz linie odpowiadające krawędziom. Jeżeli krawędź jest skierowana to oznacza się to poprzez strzałkę. Przykład tej reprezentacji przedstawiono na rysunku 2.1.



Rysunek 2.1 Graf skierowany



Rysunek 2.2 Suma grafów

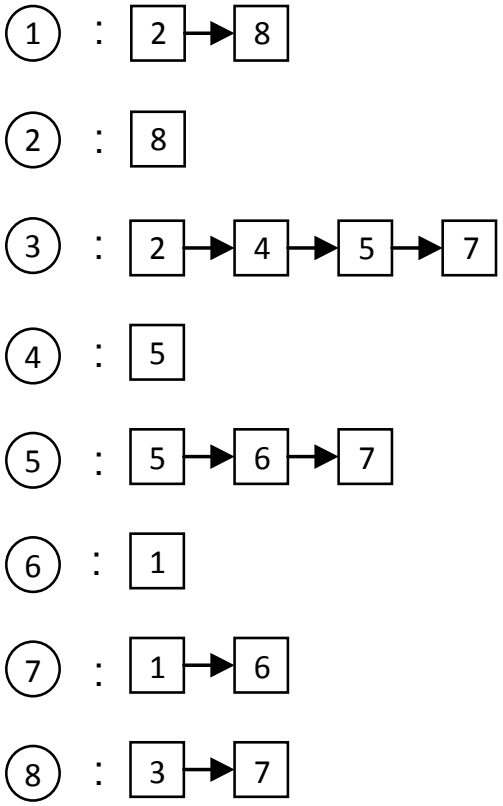
Taki sposób przedstawienia grafu, choć bardzo czytelny dla człowieka, nie nadaje się na źródło danych do przetwarzania w programie komputerowym. Najczęstszymi postaciami grafów w tym zastosowaniu są macierze i listy sąsiedztwa.

Macierz sąsiedztwa A dla grafu jest zero-jedynkową macierzą o wymiarze $n * n$, gdzie n oznacza liczbę wierzchołków. Element $A[i,j]$ jest równy jeden wtedy i tylko wtedy, gdy istnieje krawędź prowadząca od i do j. Zastosowanie takiej macierzy umożliwia sprawdzenie w czasie stałym czy istnieje krawędź pomiędzy dwoma wierzchołkami. Reprezentacja ta wymaga stosunkowo dużo pamięci niezależnie od liczby krawędzi w grafie. Rysunek 2.3 przedstawia macierz sąsiedztwa dla grafu z rysunku 2.1.

	1	2	3	4	5	6	7	8
1	0	1	0	0	0	0	0	1
2	0	0	0	0	0	0	0	1
3	0	1	0	1	1	0	1	0
4	0	0	0	0	1	0	0	0
5	0	0	0	0	1	1	1	0
6	1	0	0	0	0	0	0	0
7	1	0	0	0	0	1	0	0
8	0	0	1	0	0	0	1	0

Rysunek 2.3 Graf skierowany przedstawiony w postaci macierzy sąsiedztwa

Listą sąsiedztwa dla wierzchołka v nazywa się listę zawierającą wszystkie wierzchołki sąsiadujące z wierzchołkiem v. Graf może być reprezentowany przez n list sąsiedztwa dla n równego liczbie wierzchołków grafu. Rysunek 2.4 przedstawia listy sąsiedztwa dla grafu z rysunku 2.1.



Rysunek 2.4 Graf skierowany przedstawiony w postaci list sąsiedztwa

2.2. ZŁOŻONOŚĆ OBLICZENIOWA ALGORYTMÓW[2]

Algorytmy można oceniać na podstawie wielu różnych czynników. Najczęściej wykorzystuje się do tego celu szybkość, z jaką wzrastają czas lub pamięć potrzebne do wykonania zadania, w zależności od jego rozmiaru. Rozmiar zadania to pewna liczba całkowita, która jest miarą wielkości danych. To, co dokładnie jest reprezentowane przez rozmiar, jest wysoce zależne od problemu, którego dotyczy algorytm. Rozmiarem zadania sortowania może być liczba elementów w sortowanej kolekcji. Dla problemu, w którym dane wejściowe stanowi graf, rozmiar może być zadany liczbą wierzchołków lub krawędzi.

[2]Do opracowania wykorzystano pozycje [1], [2] i [3].

Złożoność czasowa algorytmu to czas potrzebny do jego wykonania, przedstawiony w postaci funkcji rozmiaru problemu. Czas działania algorytmu jest wyrażony liczbą wykonanych operacji dominujących, czyli czynności charakterystycznych dla danego algorytmu, których wykonywanie na ogół zajmuje najwięcej czasu. Dzięki takiemu podejściu analiza algorytmu nie jest zależna od implementacji ani sprzętu. Podczas analizy algorytmów często upraszcza się wzory określające dokładny nakład zasobów potrzebnych do ich wykonania. Robi się to poprzez opuszczenie tych fragmentów, których wpływ na wynik maleje wraz ze wzrostem rozmiaru. Asymptotyczna złożoność czasowa określa zachowanie złożoności czasowej w granicy dla rosnącego rozmiaru zadania. Analogicznie definiuje się złożoność pamięciową i asymptotyczną złożoność pamięciową. To właśnie złożoności asymptotyczne najczęściej służą do porównywania algorytmów. Zakładając, że funkcja $g(n) = cn^2$ przedstawia czas potrzebny do wykonania pewnego algorytmu, gdzie n reprezentuje rozmiar problemu a c pewną stałą, to złożoność czasową określa się jako $O(n^2)$. Mówi się wtedy, że złożoność ta jest co najwyżej rzędu n^2.

Aby zilustrować wpływ rzędu złożoności na maksymalny rozmiar zadania założono, że dla pewnej maszyny wykonującej algorytmy jednostka czasu jest równa jednej milisekundzie, oraz istnieją trzy algorytmy o różnej złożoności czasowej, odpowiednio: $A_1: O(n)$; $A_2: O(n^2)$; $A_3: O(2^n)$. Algorytm pierwszy w ciągu sekundy będzie w stanie rozwiązać problem o rozmiarze 1000. Algorytm drugi w tym samym czasie rozwiąże problem o rozmiarze nie większym niż 31. Natomiast maksymalny rozmiar zadania dla algorytmu trzeciego to tylko 9.

Powyższe zestawienie obrazuje jak wielki wpływ na możliwości doboru rozmiaru zadań ma rząd złożoności algorytmu. Należy pamiętać, że w pewnych sytuacjach algorytmy o wysokim rzędzie złożoności mogą mieć mniejszą wydajność niż algorytmy o małym rzędzie. Sytuacja taka może mieć miejsce dla algorytmów o złożonościach: $A_1: 100n$; $A_2: 2^n$, gdy rozmiar problemu jest mniejszy niż 10. Możliwe jest również, że dla danego problemu rozmiar nigdy nie osiągnie wartości, dla której algorytm o wyższym rzędzie złożoności będzie mniej wydajny, jednak jest to sytuacja stosunkowo rzadka. Rząd złożoności staje bardziej istotny wraz ze wzrostem rozmiaru problemu, dlatego też dla dużych problemów algorytmy o mniejszym rzędzie złożoności są wydajniejsze.

O problemach rozwiązywalnych za pomocą algorytmów wielomianowych lub algorytmów o złożoności niższej niż wielomianowa mówi się, że są problemami łatwo rozwiązywalnymi i należą do klasy $\mathcal{P}$. Algorytm wielomianowy to taki algorytm, którego czas działania w pesymistycznym przypadku dla danych wejściowych o rozmiarze n wynosi nie więcej niż $O(n^k)$, gdzie k jest stałą. O problemach dla których nie istnieją algorytmy o złożoności wielomianowej lub niższej mówi się, że są trudno rozwiązywalne.

Algorytm nazywa się deterministycznym, jeżeli w danej chwili może realizować tylko jedno określone polecenie. Algorytm jest niedeterministyczny, jeżeli może on z danego stanu przejść do więcej niż jednego stanu następnego. Innymi słowy może on realizować jednocześnie więcej niż jedno działanie. Problemy klasy $\mathcal{NP}$ to problemy, które mogą być rozwiązane w czasie wielomianowym za pomocą algorytmu niedeterministycznego. Wszystkie problemy klasy $\mathcal{P}$ należą również do klasy $\mathcal{NP}$. Nie istnieje jednak dowód określający czy wszystkie problemy klasy $\mathcal{NP}$ należą do klasy $\mathcal{P}$.

Problem nazywa się $\mathcal{NP}$–trudnym, jeżeli rozwiązujący go algorytm deterministyczny o wielomianowym czasie przetwarzania da się wykorzystać do uzyskania deterministycznego algorytmu o wielomianowym czasie przetwarzania dla każdego problemu z klasy $\mathcal{NP}$. Innymi słowy problem jest $\mathcal{NP}$–trudny, gdy jest on co najmniej tak trudny, jak każdy problem z klasy $\mathcal{NP}$. Problem $\mathcal{NP}$–trudny należący do klasy $\mathcal{NP}$ nazywa się $\mathcal{NP}$–zupełnym, gdy jest on co najmniej tak trudny, jak każdy problem z klasy $\mathcal{NP}$, ale nie trudniejszy. Należy zauważyć, że problem $\mathcal{NP}$–trudny nie musi należeć do klasy $\mathcal{NP}$.

Dla żadnego problemu $\mathcal{NP}$–zupełnego nie znaleziono algorytmu, dzięki któremu można by uznać go za problem łatwo rozwiązywalny, jednak nie udowodniono, że taki algorytm nie istnieje. Wykazano natomiast, że jeżeli jeden z problemów tej klasy jest problemem łatwo rozwiązywalnym, to wszystkie problemy tej klasy są łatwo rozwiązywalne.

Przykładem problemu $\mathcal{NP}$–zupełnego jest problem spełnialności, którego rozwiązanie polega na określeniu, czy dane wyrażenie logiczne może być prawdziwe. Przykładem problemu $\mathcal{NP}$–trudnego jest problem plecakowy. Problem ten polega na takim doborze przedmiotów z danego zbioru, aby ich wartość była

jak największa, ale sumaryczna waga nie przekroczyła pewnej narzuconej maksymalnej wartości.

2.3. HEURYSTYKI[3]

Optymalizacja jest procesem znajdowania jak najlepszego rezultatu dla danego problemu. Najlepsze istniejące rozwiązanie nazywa się optymalnym. W trakcie optymalizacji wykorzystuje się funkcję oceny, dzięki której można określić jakość każdego z badanych rozwiązań. Wartość tej funkcji, w zależności od problemu, rośnie lub maleje wraz ze wzrostem jakości rozwiązania, dlatego też zadanie optymalizacji sprowadza się do znalezienia jej minimum lub maksimum.

Praktyczne próby znalezienia optymalnego rozwiązania dla problemów trudno rozwiązywalnych często nie są realizowalne w akceptowalnym czasie, ze względu na złożoność tych problemów. Najprostszy algorytm dokładny, to znaczy znajdujący optymalny wynik, musi wyznaczyć i ocenić wszystkie wyniki z przestrzeni możliwych rozwiązań, która bywa bardzo rozległa. Jednak w wielu przypadkach zadowalające jest rozwiązanie zbliżone do optymalnego. W takich sytuacjach często stosuje się algorytmy przeczesujące przestrzeń rozwiązań w poszukiwaniu jak najlepszego rezultatu. Algorytmy te często kończą swoje działanie po upływie pewnego narzuconego czasu lub po otrzymaniu wystarczająco dobrego wyniku.

Ze względu na powtarzalność otrzymywanych wyników algorytmy można podzielić na deterministyczne i probabilistyczne. Algorytmy deterministyczne dla tego samego zestawu danych wejściowych zawsze zwrócą ten sam wynik. Nie ma takiej gwarancji dla algorytmów probabilistycznych, a to dlatego, że ich działanie opiera się na elemencie losowym. Na pewnym etapie rozwiązania jest on wykorzystywany podczas wyboru kolejnego kroku algorytmu.

Reprezentacją nazywa się sposób, w jaki są przechowywane informacje o problemie i rozwiązaniach. Dla problemu komiwojażera, który można przedstawić jako znalezienie cyklu Hamiltona o najmniejszej wadze w grafie, reprezentacją rozwiązania może być ciąg kolejnych wierzchołków należących do cyklu.

[3]Do opracowania wykorzystano pozycje [5] i [6].

Rozwiązanie A nazywa się sąsiednim względem rozwiązania B, gdy różni się ono nieznacznie pod względem budowy. Często rozwiązania sąsiednie posiadają zbliżone wartości funkcji oceny, choć nie jest to regułą. Dokładne określenie sąsiedztwa zależy od problemu i jego reprezentacji. Dla problemu komiwojażera rozwiązanie sąsiednie można określić jako ciąg odwiedzonych miast w którym dwa miasta są ze sobą zamienione miejscami. Można również rozszerzyć to sąsiedztwo poprzez zdefiniowanie go jako zbiór rozwiązań, w których co najwyżej k miast jest zamienionych kolejnością, gdzie k jest pewną stałą.

W technice iteracyjnego ulepszania rozwiązania podczas kolejnych kroków algorytmu analizuje się rozwiązania sąsiednie względem rozwiązania bieżącego. Jeżeli któryś z sąsiadów okazuje się lepszy, przechodzi się do niego i traktuje jako bieżące rozwiązanie. Niestety za pomocą tej techniki można znaleźć jedynie optimum lokalne, czyli rozwiązanie, dla którego wszystkie rozwiązania sąsiednie dają gorszy wynik funkcji oceny. Nie ma jednak gwarancji, że nie istnieje rozwiązanie lepsze znajdujące się poza sąsiedztwem optimum lokalnego. Powstaje zatem niebezpieczeństwo utknięcia w takim punkcie przestrzeni rozwiązań, z którego wyjście za pomocą techniki iteracyjnego ulepszania rozwiązania nie jest możliwe. Istnieje wiele metod wychodzenia z lokalnych optimów jednak wszystkie sprowadzają się do tego, aby w odpowiednim momencie przejść w inny rejon przestrzeni możliwych rozwiązań.

Heurystyka to pewien sposób wyznaczania rozwiązań wykorzystujący wiedzę o problemie. Nie gwarantuje ona znalezienia optimum, a często nawet uzyskania prawidłowego wyniku. Jej celem jest takie ukierunkowanie procesu budowania rozwiązań, aby były one jak najlepsze przy jak najmniejszym nakładzie obliczeniowym lub pamięciowym. Wspomniane wcześniej algorytmy przeszukujące często wykorzystują heurystyki w celu lepszego ukierunkowania procesu budowania rozwiązań. Metody tej używa się również do otrzymywania rezultatów zbliżonych do optimum, na podstawie których wylicza się rozwiązanie optymalne, skracając tym samym czas działania całego algorytmu.

Algorytmy heurystyczne posiadają pewną funkcję heurystyczną. Za jej pomocą dokonują oceny możliwych do podjęcia kolejnych kroków, a następnie, biorąc tę ocenę pod uwagę, wykonują kolejne działania. Proces wyboru kolejnego

kroku może polegać na wybraniu elementu o najlepszej ocenie, lecz nie jest to regułą. Inną powszechnie stosowaną praktyką w algorytmach probabilistycznych jest wykorzystanie funkcji heurystycznej do określenia prawdopodobieństwa wyboru poszczególnych działań.

Najbardziej znanymi algorytmami heurystycznymi są: symulowane wyżarzanie, przeszukiwanie z tabu, algorytmy ewolucyjne, systemy mrówkowe oraz rój cząstek.

3. PROBLEM SEKWENCYJNEGO PORZĄDKOWANIA I ALGORYTM ENHANCED ANT COLONY SYSTEM[4]

W niniejszym rozdziale omówiony zostanie problem sekwencyjnego porządkowania oraz algorytm Enhanced Ant Colony System.

3.1. DEFINICJA PROBLEMU SEKWENCYJNEGO PORZĄDKOWANIA

Dany jest graf pełny $G = (V, E)$, którego węzły $0, \dots, i, \dots, n$ reprezentują zadania. Każda krawędź (i, j) posiada nieujemną wagę $t_{ij} \in R$ reprezentującą czas oczekiwania pomiędzy zakończeniem zadania i, a rozpoczęciem zadania j. Waga ta w dalszej części pracy będzie nazywana również kosztem. Dane są również ograniczenia kolejności wykonywania zadań w postaci acyklicznego grafu skierowanego $P = (V, F)$ zdefiniowanego na tym samym zbiorze wierzchołków V. Jeżeli istnieje krawędź $(i, j) \in F$, oznacza to, że zadanie i musi być wykonane przed zadaniem j, aby rozwiązanie było poprawne. Jeżeli istnieje krawędź $(i, j) \in F$ oraz $(j, k) \in F$ to istnieje również $(i, k) \in F$. Jako, że ciąg kolejno wykonywanych zadań zawsze zaczyna się od zadania 0, a kończy na zadaniu n, to $(0, i) \in F \; \forall i \in V \backslash \{0\}$, oraz $(i, n) \in F \; \forall i \in V \backslash \{n\}$. Problem sekwencyjnego porządkowania polega na znalezieniu sekwencji wykonywania zadań, której koszt jest jak najmniejszy, uwzględniając przy tym ograniczenia kolejności. Jest to równoznaczne ze znalezieniem ścieżki Hamiltona w grafie G uwzględniającej ograniczenia zawarte w grafie P.

3.2. REPREZENTACJE PROBLEMU SEKWENCYJNEGO PORZĄDKOWANIA

Problem sekwencyjnego porządkowania można przedstawić w postaci dwóch macierzy: kosztów i ograniczeń. Macierz kosztów jest macierzą kwadratową składającą się z $(n + 1) * (n + 1)$ elementów, gdzie n jest indeksem ostatniego zadania. Poszczególne jej elementy odzwierciedlają wagi z grafu G w taki sposób, że waga t_{ij} znajduje się w elemencie macierzy o indeksie $[i + 1, j +$

[4] Do opracowania wykorzystano pozycje: [7], [8] i [9]

$1]$, zakładając indeksację macierzy rozpoczynającą się od jedynki. Macierz ograniczeń również jest macierzą kwadratową o tym samym rozmiarze, co macierz kosztów. Przedstawia ona ograniczenia zawarte w grafie P. Element macierzy ograniczeń o indeksie $[i,j]$ przyjmuje wartość 1, jeżeli istnieje krawędź $(i-1, j-1) \in F$. Jeżeli warunek ten nie jest spełniony, wartość elementu jest równa 0.

Często stosowaną reprezentacją jest odzwierciedlenie danych problemu za pomocą jednej macierzy łączącej informacje z macierzy kosztów i macierzy ograniczeń. Należy zauważyć, że jeżeli istnieje krawędź $(i,j) \in F$, to krawędź $(j,i) \in E$ nie może zostać wykorzystana do zbudowania poprawnego rozwiązania, ponieważ naruszałoby to jedno z ograniczeń. Dlatego też informacja o koszcie związanym z przejściem z węzła j do węzła i z punktu widzenia części algorytmów jest zbędna. W takiej sytuacji do elementu macierzy $[j+1, i+1]$ wpisuje się wartość -1. Wartość ta nie może być mylona z kosztem krawędzi, ponieważ zgodnie z założeniami musi on być nieujemny. Na rysunku 3.1 przedstawiono przykładową macierz kosztów, macierz ograniczeń oraz macierz wynikającą z ich połączenia.

	1	2	3	4	5	6
1	0	1	3	5	8	7
2	2	0	6	7	5	6
3	5	1	0	5	3	1
4	4	8	7	0	1	5
5	3	3	5	6	0	1
6	8	2	4	6	8	0

(a)

	1	2	3	4	5	6
1	0	1	1	1	1	1
2	0	0	0	1	0	1
3	0	1	0	1	0	1
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

(b)

	1	2	3	4	5	6
1	0	1	3	5	8	7
2	-1	0	-1	7	5	6
3	-1	1	0	5	3	1
4	-1	-1	-1	0	1	5
5	-1	3	5	6	0	1
6	-1	-1	-1	-1	-1	0

(c)

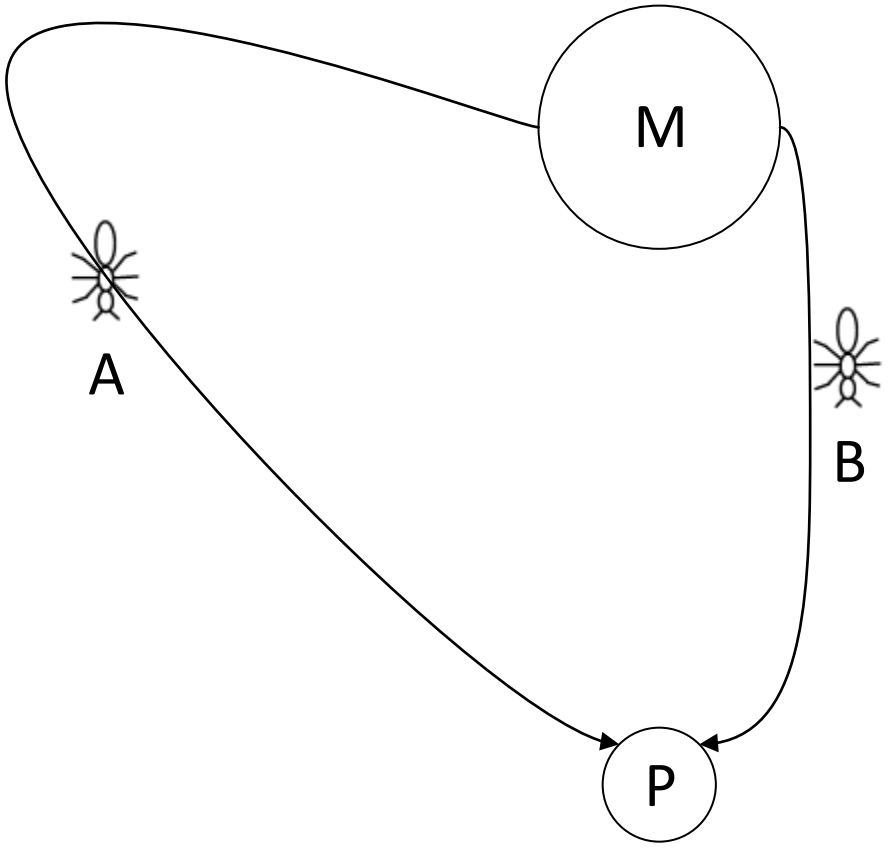
Rysunek 3.1 Macierz kosztów (a), macierz ograniczeń (b), macierz łącząca macierz kosztów i macierz ograniczeń (c)

3.3. OGÓLNY OPIS ALGORYTMÓW MRÓWKOWYCH

Algorytmy mrówkowe, jak nazwa sugeruje, zostały zainspirowane przez zachowania mrówek, a konkretniej proces szukania pożywienia. Robotnica wędrując po okolicy mrowiska odbiera swoimi zmysłami informacje tylko o swoim

najbliższym otoczeniu. Często dociera do miejsc, z których nie „widzi” mrowiska. Wie natomiast jak do niego wrócić dzięki feromonom, które zostawia za sobą podczas wędrówki. Ślad feromonowy to swego rodzaju pamięć zbiorowa mrówek o otoczeniu mrowiska. Mrówka podczas wyboru trasy swojej wędrówki bierze pod uwagę ślady feromonowe zostawione przez inne mrówki. Im więcej feromonu w danym miejscu tym większa szansa, że mrówka będzie za nim wędrowała. W połączeniu z właściwością wyparowywania feromonu tworzy to ciekawy mechanizm umożliwiający znajdowanie krótkich ścieżek do pożywienia i szybki jego transport do mrowiska.

Na rysunku 3.2 zaprezentowana została przykładowa sytuacja, w której dwie mrówki, oznaczone literami A i B szukają drogi od mrowiska M do pożywienia P.



Rysunek 3.2 Mrówki wędrujące do pożywienia

Odległość, którą musi przebyć mrówka A jest znacznie większa niż odległość, którą musi przebyć mrówka B. W związku z tym w momencie pokonania całej drogi $(M \rightarrow P \rightarrow M)$ ilość feromonu na drodze przebytej przez mrówkę A będzie mniejsza niż na drodze mrówki B. Dzieje się tak, ponieważ

przebycie drogi mrówki A zajmuje więcej czasu, a co za tym idzie, więcej feromonu zdąży odparować. Kolejne mrówki wyruszając w trasę będą preferowały drogę odkrytą przez mrówkę B. Oczywiście możliwe jest też, że będą próbowały znaleźć zupełnie nową drogę do pożywienia w okolicy mrowiska.

Pierwsze wersje algorytmów mrówkowych dość dokładnie odwzorowywały zachowanie mrówek. Jednak cel mrówek różni się od celów stawianych algorytmom mrówkowym. Mrówki mają przetransportować jak najwięcej pokarmu w jak najkrótszym czasie, natomiast algorytm mrówkowy najczęściej ma za zadanie znaleźć jak najkrótszą drogę pomiędzy danymi punktami. O ile w przypadku mrówek poruszanie się wielu osobników po krótkiej trasie jest jak najbardziej oczekiwane, o tyle w algorytmach mrówkowych takie zachowanie jest stratą czasu. Mrówki, które wędrują po już zbadanych trasach, nigdy nie znajdą rozwiązania lepszego. Dlatego też algorytmy mrówkowe zostały zmodyfikowane w taki sposób, aby zróżnicowanie dróg było jak największe.

3.4. Opis algorytmu Enhanced Ant Colony System

Algorytm Enhanced Ant Colony System (EACS) został po raz pierwszy przedstawiony w pracy L. M. Gambardella'ego, R. Montemanni'ego i D. Weylanda [7]. Jest to algorytm dedykowany dla problemu sekwencyjnego porządkowania. Opiera się on na algorytmie Hybrid Ant System for the Sequential Ordering Problem (HAS-SOP) przedstawionym przez L. M. Gambardella'ego i M. Doriga [8].

W algorytmie EACS każda mrówka iteracyjnie buduje kompletne rozwiązanie, czyli ścieżkę Hamiltona zaczynającą się w węźle 0 i kończącą w węźle n. Będąc w węźle i mrówka wybiera kolejny węzeł do odwiedzenia. Podczas tego wyboru bierze pod uwagę węzły należące do zbioru $F(i)$, które mogą w danym momencie zostać wykorzystane do zbudowania poprawnego rozwiązania. Węzły znajdujące się w zbiorze $F(i)$ nie zostały jeszcze odwiedzone, a dodatkowo wszystkie dotyczące ich ograniczenia kolejności zostały spełnione.

EACS podczas swojej pracy silnie bazuje na najkrótszej dotychczas znalezionej ścieżce Hamiltona. Podczas wyboru kolejnej krawędzi mrówka wybierze dokładnie tę samą drogę z węzła i, która była wybrana w najlepszym dotychczas znalezionym rozwiązaniu, z prawdopodobieństwem q_0. Jeżeli droga ta nie może w danym momencie być wykorzystana do zbudowania poprawnego

rozwiązania, mrówka wybierze drogę o największej wartości $\tau_{ij} * \eta_{ij}$. Symbol τ_{ij} oznacza ilość feromonu na drodze z węzła i do węzła j, natomiast η_{ij} reprezentuje wartość funkcji oceny jakości tej drogi wyrażającą się wzorem $\eta_{ij} = 1/t_{ij}$ (t_{ij} to waga krawędzi). Taki wybór kolejnego węzła nazywany będzie wyborem deterministycznym.

Z prawdopodobieństwem $1 - q_0$ mrówka wybierze kolejny węzeł w sposób probabilistyczny. W takim wypadku mrówka dokonuje oceny atrakcyjności każdej z dróg prowadzących do węzłów należących do $F(i)$ i na jej podstawie określa prawdopodobieństwo przejścia do danego węzła $j \in F(i)$ zgodnie ze wzorem:

$$p_{ij} = \frac{\tau_{ij} * \eta_{ij}}{\sum_{l \in F(i)} \tau_{il} * \eta_{il}}$$

Wartość q_0 wyrażona jest wzorem $q_0 = 1 - s/n$, gdzie s oznacza oczekiwaną liczbę węzłów wybranych w sposób probabilistyczny.

Na iterację algorytmu mrówkowego składają się dwie istotne operacje: zbudowanie rozwiązań przez wszystkie mrówki i wykonanie globalnej aktualizacji śladów feromonowych. Funkcja aktualizująca ślad feromonowy w EACS zmienia ilość feromonu na krawędziach należących do globalnie najlepszego rozwiązania, znalezionego od początku działania algorytmu. Rozwiązanie to nie musi należeć do aktualnej iteracji. Do zmiany śladu feromonowego wykorzystuje się wzór:

$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \rho / L_{best}$$

L_{best} jest kosztem globalnie najlepszego rozwiązania natomiast ρ jest parametrem algorytmu i przyjmuje wartości w przedziale $\langle 0,1 \rangle$. Łatwo zauważyć, że powyższy wzór składa się z dwóch części. Pierwsza z nich $(1 - \rho) * \tau_{ij}$ jest odpowiedzialna za odzwierciedlenie efektu wyparowywania, natomiast druga ρ / L_{best} symuluje pozostawianie feromonu przez mrówkę.

Ślad feromonowy zostaje również aktualizowany podczas budowania rozwiązań przez mrówki. Każda mrówka przechodząc po krawędzi (i, j) pomniejsza ilość feromonu związaną z tą krawędzią. To działanie nazywane jest też zjadaniem feromonu. Nie ma ono odzwierciedlenia w świecie przyrody, jednak powoduje większe zróżnicowanie pomiędzy rozwiązaniami budowanymi przez mrówki. Do zjadania feromonu stosuje się następujący wzór:

$$\tau_{ij} = (1 - \varphi) * \tau_{ij} + \varphi * \tau_0$$

gdzie φ jest parametrem algorytmu, a τ_0 reprezentuje początkową ilość feromonu na krawędziach. Dobrymi wartościami parametrów są $\tau_0 = (FS * n)^{-1}$, $\rho = \varphi = 0{,}1$, $s = 10$. FS to koszt pierwszego wygenerowanego rozwiązania z pominięciem etapu lokalnej optymalizacji.

Jeżeli koszt zbudowanego przez mrówkę rozwiązania jest mniejszy lub równy $1{,}2 * L_{best}$ to rozwiązanie to jest poddawane lokalnej optymalizacji. Proces ten polega na wprowadzaniu nieznacznych zmian w znalezionym rozwiązaniu i weryfikacji ich korzystności.

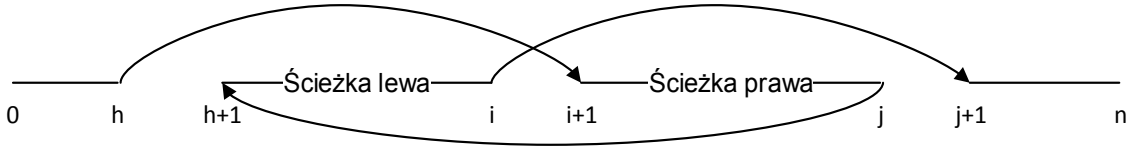
Warunek zatrzymania algorytmu nie jest jednoznacznie zdefiniowany i zależy od konkretnej implementacji. Najczęściej stosowane warunki zatrzymania to przekroczenie zadanego czasu wykonania, wykonanie zadanej liczby iteracji oraz wystąpienie zadanej liczby iteracji przy równoczesnym braku poprawy rozwiązania.

Pseudokod 3.1 przedstawia działanie algorytmu EACS.

```
procedure EACS (problemSekwencyjnegoPorzadkowania)
{
    ResetujFeromony(problemSekwencyjnegoPorzadkowania)
    najepszeRozwiazanie = null
    while(CzyWarunekStopuNieSpelniony())
    {
        ResetujMrowki()
        DlaKazdejMrowki
        {
            mrowka.ZnajdzRozwiazanie(problemSekwencyjnegoPorzadkowania,
                                     najlepszeRozwiazanie)
            if(najlepszeRozwiazanie == null ||
               mrowka.Rozwiazanie.Koszt <= najlepszeRozwiazanie.Koszt * 1.2)
            {
                LokalnaOptymalizacja(mrowka.Rozwiazanie, najlepszeRozwiazanie,
                                     problemSekwencyjnegoPorzadkowania)
                if(najlepszeRozwiazanie == null ||
                   mrowka.Rozwiazanie.Koszt <= najlepszeRozwiazanie.Koszt)
                {
                   najlepszeRozwiazanie = mrowka.Rozwiazanie
                }
            }
        }
        AktualizujFeromony(najlepszeRozwiazanie)
    }
    return najlepszeRozwiazanie
}
```

Pseudokod 3.1 Algorytm Enhanced Ant Colony System

Procedura lokalnej optymalizacji tworzy nowe rozwiązanie poprzez zamianę trzech krawędzi na inne. Charakteryzuje ją kierunek przeszukiwania sąsiedztwa rozwiązania, który określa, jak zmienia się dobór zamienianych krawędzi w kolejnych iteracjach. Pojedynczą zamianę dla kierunku w przód zobrazowano na rysunku 3.3.

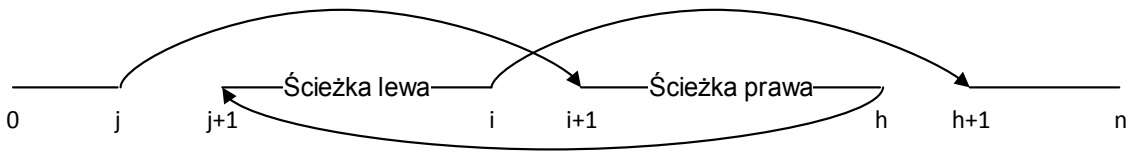


Rysunek 3.3 Pojedyncza zamiana krawędzi dla kierunku przeszukiwania w przód

Krawędzie $(h, h+1)$, $(i, i+1)$ i $(j, j+1)$ zostały zamienione na $(h, i+1)$, $(i, j+1)$ i $(j, h+1)$ przez co węzły ścieżki prawej zostaną odwiedzone przed węzłami ścieżki lewej.

Na początku działania procedury dobiera się punkty h, które mają zostać wykorzystane podczas przeszukiwania sąsiedztwa. Następnie wykonuje się przeszukiwanie w przód. Dla każdego punktu h iteracyjnie zwiększa się punkty $i \in \langle h+1, n-2 \rangle$ oraz $j \in \langle i+1, n-1 \rangle$. Przy każdym zwiększeniu indeksu j sprawdza się, czy znaleziono lepsze rozwiązanie. Jeżeli tak, zapamiętuje się wartości indeksów h, i oraz j i dalej iteruje pętlę zmieniającą wartość j szukając jeszcze lepszej zamiany dla tych samych h oraz i. Gdy pętla ta się skończy i znaleziono lepsze rozwiązanie przechodzi się do zamiany krawędzi zaniechując sprawdzanie rozwiązań dla pozostałych możliwych wartości indeksu i. Jeżeli natomiast nie znaleziono żadnej zamiany polepszającej rozwiązanie, następuje iteracja pętli po indeksie i. Jeżeli podczas całego przeszukiwania w przód nie znaleziono lepszego rozwiązania następuje przeszukiwanie w tył dla tej samej wartości indeksu h.

Dla kierunku przeszukiwania w tył pojedyncza zamiana została przedstawiona na rysunku 3.4.



Rysunek 3.4 Pojedyncza zamiana krawędzi dla kierunku przeszukiwania w tył

Podczas przeszukiwania w tył indeksy i oraz j są zmniejszane w kolejnych iteracjach. Ich wartości początkowe to: $i = h - 1$ oraz $j = i - 1$, natomiast zakres możliwych wartości to: $i \in \langle 1, h - 1 \rangle$ oraz $j \in \langle 0, i - 1 \rangle$.

Punkty h, które należy sprawdzić podczas wykonywania procedury, przechowuje się na stosie. Niestety opis jego inicjalizacji zawarty w pracy [7] nie jest precyzyjny. Mówi on, że stos jest inicjalizowany elementami, których kolejność nie jest zgodna z kolejnością elementów w najlepszym dotychczas znalezionym rozwiązaniu. Wzorując się na pracy A. Ezzata [9] przyjęto, że indeks i należy dodać do stosu, jeżeli dla ulepszanej drogi przedstawionej w postaci kolejnych odwiedzanych węzłów, i-ty węzeł jest różny od i-tego węzła w najlepszym dotychczas znalezionym rozwiązaniu.

Po wykonaniu zamiany aktualną wartość h pobraną ze stosu uznaje się za przetworzoną i algorytm wraca do punktu, w którym pobiera kolejną wartość indeksu h. Jednak zanim to nastąpi, do stosu dodawane są wartości indeksów związanych z zamianą: $j + 1$, j, $i + 1$, i, $h + 1$, h. Jeżeli dana wartość znajduje się już na stosie, to jej dodawanie jest pomijane. W pracy L. M. Gambardella'ego [8] nie podano dokładnej kolejności wstawiania elementów do stosu. Dlatego też ponownie na wzór pracy A. Ezzata [9] przyjęto, że elementy są wkładane w takiej kolejności, aby najniższa wartość znajdowała się na szczycie stosu. To znaczy, że w przypadku przeszukiwania w przód na szczycie stosu znajdzie się wartość h, a dla przeszukiwania w tył wartość j.

Ważnym elementem procedury lokalnej optymalizacji rozwiązania jest sposób, w jaki radzi sobie ona z ograniczeniami dotyczącymi kolejności odwiedzanych miast. Jak nie trudno zauważyć, węzły należące do ścieżki prawej zostaną odwiedzone przed węzłami należącymi do ścieżki lewej, a co za tym idzie, aby zamiana stworzyła poprawne rozwiązanie, żaden z węzłów ścieżki prawej nie może wymagać odwiedzenia któregokolwiek węzła ścieżki lewej. Do efektywnego sprawdzenia, czy dana zamiana stworzy poprawne rozwiązanie, algorytm wykorzystuje dodatkowe zmienne. Pierwszą z nich jest zmienna $count\_h$, przyjmująca na początku wartość 0. Jest ona inkrementowana gdy kolejna wartość h jest pobierana ze stosu, lub gdy zmienia się kierunek przeszukiwania. Kolejną zmienną jest $mark$, która jest tablicą liczb całkowitych zainicjalizowaną zerami. Jej elementy przechowują informacje powiązane z każdym z węzłów

optymalizowanego rozwiązania. Poprzez $następca[k]$ oznaczany będzie zbiór węzłów, które wymagają odwiedzenia węzła z rozwiązania o indeksie k, natomiast poprzez $poprzednik[k]$ oznaczany będzie zbiór węzłów, które muszą zostać odwiedzone zanim będzie możliwe odwiedzenie węzła k. Dla przeszukiwania w przód podczas każdego ustalania wartości indeksu i, dla każdego węzła $s \in następca[i]$ wykonywane jest przypisanie: $mark[s] = count\_h$. Dzięki temu wszystkie węzły, które muszą zostać odwiedzone po którymkolwiek z węzłów w ścieżce lewej są oznaczone wartością $count\_h$. Następnie, gdy ścieżka prawa jest zwiększana poprzez inkrementację indeksu j, wykonywane jest sprawdzenie, czy wartość $mark[j]$ jest równa $count\_h$. Jeżeli tak, oznacza to, że węzeł o indeksie j wymaga odwiedzenia któregoś z węzłów w ścieżce lewej, a co za tym idzie indeksy i, j oraz h nie mogą zostać wykorzystane do stworzenia poprawnego rozwiązania. W takim wypadku przerywa się działanie pętli zmieniającej indeks j i kontynuuje poszerzanie ścieżki lewej. Dla kierunku w tył procedura działa analogicznie z tą różnicą, że podczas ustalania wartości indeksu i, węzeł $s \in poprzednik[i]$. Kolejna różnica występuje podczas sprawdzania, czy dana zamiana stworzy poprawne rozwiązanie. Porównuje się tutaj $mark[j+1]$ i $count\_h$, a nie $mark[j]$, jak to miało miejsce dla kierunku w przód.

Pseudokod 3.2 przedstawia logikę działania procedury.

```
procedure LokalnaOptymalizacja(ulepszaneRozwiazanie, najlepszeRozwiazanie,
                               problemSekwencyjnegoPorzadkowania)
{
    stos = InicjalizujStos(ulepszaneRozwiazanie, najlepszeRozwiazanie)
    count_h = 0
    mark = InicjalizujMark()
    while(stos.CzyMaElementy())
    {
        h = stos.Pop()
        count_h++
        czyDokonanoZamiany = Szukaj(true, h, count_h, mark,
                                    ulepszaneRozwiazanie, stos)
        if(czyDokonanoZamiany)
        {
            continue
        }
        count_h++
        Szukaj(false, h, count_h, mark, ulepszaneRozwiazanie, stos)
    }
}

procedure Szukaj(czyWPrzod, h, count_h, mark, ulepszaneRozwiazanie, stos)
{
```

```
    czyZnalezionoZamiane = false
    i = InicjalizujI(h, czyWPrzod)
    najlepszeH = -1, najlepszeI = -1, najlepszeJ = -1
    while(CzyKontynuowacPetleI(i, czyWPrzod))
    {
        j = InicjalizujJ(i, czyWPrzod))
        najlepszyZysk = 0
        OznaczWezly(solutionToImprove, i, mark, count_h, czyWPrzod)
        czyPoprawne = true
        while(CzyKontynuowacPetleJ(j, czyPoprawne, czyWPrzod))
        {
            czyPoprawne = CzyZamianaPoprawna(j, mark, countH, czyWPrzod)
            if(czyPoprawne)
            {
                zysk = ObliczZysk(h, i, j, ulepszaneRozwiazanie, czyWPrzod)
                if(zysk > najlepszyZysk)
                {
                    czyZnalezionoZamiane = true
                    najlepszeH = h
                    najlepszeI = i
                    najlepszeJ = j
                    najlepszyZysk = zysk
                }
            }
            j = WyznaczNastepneJ(j, czyWPrzod)
        }
        if(czyZnalezionoZamiane)
        {
            WykonajZamiane(najlepszeH, najlepszeI, najlepszeJ,
                           ulepszaneRozwiazanie)
            WlozNaStos(stos, najlepszeH, najlepszeI, najlepszeJ,
                       najlepszeH + 1, najlepszeI + 1, najlepszeJ + 1,
                       czyWPrzod)
            return true
        }
        i = WyznaczNastepneI(i, czyWPrzod)
    }
    return false
}
```

Pseudokod 3.2 Procedura lokalnej optymalizacji

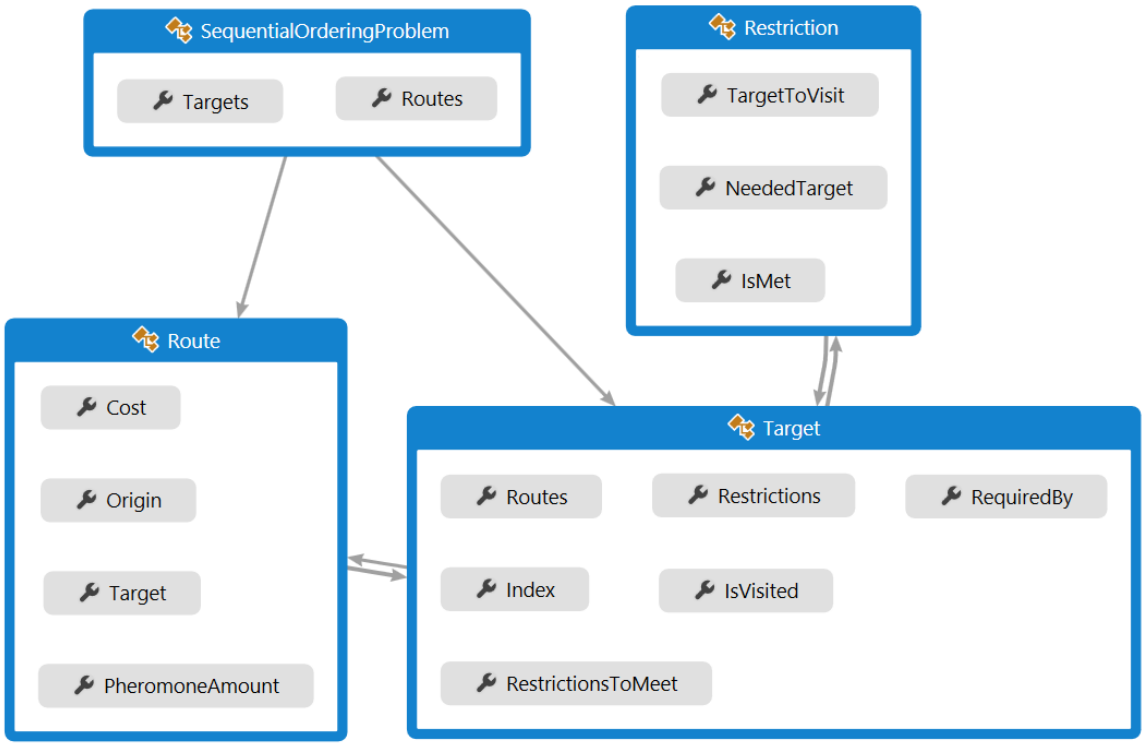
4. Opis implementacji algorytmu EACS wraz z wprowadzonymi modyfikacjami

Podczas prac badawczych wykonano własną implementację algorytmu EACS, ponieważ ta stworzona przez autorów algorytmu w pracy [7] nie jest ogólnie dostępna. Rozdział ten opisuje jej szczegóły, jak również zaproponowane modyfikacje.

4.1. Szczegóły implementacji

Podrozdział ten opisuje niektóre elementy implementacji algorytmu EACS. Program wykorzystany do badań został stworzony w języku C# z wykorzystaniem technologii .NET w wersji 4.5.

Stworzono cztery klasy przechowujące informacje o problemie: SequentialOrderingProblem, Route, Target oraz Restriction. Rysunek 4.1 przedstawia diagram zależności tych klas oraz najważniejsze elementy w nich zawarte.



Rysunek 4.1 Diagram zależności niektórych klas programu odpowiedzialnych za przechowywanie informacji o problemie sekwencyjnego porządkowania

Klasa Target reprezentuje pojedynczy węzeł. Zawiera ona następujące właściwości:

- Routes – lista dróg możliwych do odwiedzenia z danego węzła;
- Restrictions – lista ograniczeń, które muszą zostać spełnione, aby było możliwe odwiedzenie danego węzła;
- RequiredBy – lista ograniczeń, które stają się spełnione po odwiedzeniu danego węzła;
- Index – indeks danego węzła;
- IsVisited – flaga informująca o tym, czy dany węzeł został już wykorzystany w budowanym rozwiązaniu;
- RestrictionsToMeet – liczba ograniczeń, które muszą zostać spełnione, aby dany węzeł mógł zostać odwiedzony.

Klasa Restriction reprezentuje pojedyncze ograniczenie występujące pomiędzy dwoma węzłami. Zawiera następujące właściwości:

- TargetToVisit – węzeł, który nie może zostać odwiedzony dopóki ograniczenie nie zostanie spełnione;
- NeededTarget – węzeł, który musi zostać odwiedzony, aby możliwe było odwiedzenie węzła TargetToVisit;
- IsMet – flaga informująca o tym, czy dane ograniczenie zostało spełnione podczas budowania bieżącego rozwiązania.

Klasa Route reprezentuje pojedynczą skierowaną krawędź. Zawiera ona następujące właściwości:

- Cost – koszt (waga) związany z daną krawędzią;
- Origin – węzeł będący początkiem krawędzi;
- Target – węzeł, do którego skierowana jest krawędź;
- PheromoneAmount – ilość feromonu powiązana z daną krawędzią.

Klasa SequentialOrderingProblem reprezentuje problem sekwencyjnego porządkowania. Jej właściwości to:

- Targets – lista węzłów związanych z daną instancją problemu;

- Routes – tablica kwadratowa posiadająca $(n+1)*(n+1)$ elementów, gdzie n jest indeksem ostatniego węzła. Reprezentuje drogi możliwe do wykorzystania w poprawnym rozwiązaniu. Droga pomiędzy węzłami o indeksach i oraz j znajduje się w elemencie $[i,j]$. Jeżeli taka droga nie istnieje, w elemencie $[i,j]$ znajduje się *null*.

Podczas inicjalizacji danych problemu zadbano o to, aby każda krawędź, węzeł czy ograniczenie było reprezentowane przez dokładnie jedną instancję odpowiedniego obiektu. Dzięki temu nie występuje redundancja danych, a stworzone zależności umożliwiają efektywne przeszukiwanie informacji zależnie od wykonywanej operacji i jej parametrów.

Stworzono również klasę Solution, która przechowuje rozwiązanie za pomocą listy kolejno przebytych krawędzi. Klasa ta na bieżąco zlicza aktualny koszt rozwiązania przy jakichkolwiek jego modyfikacjach. Dzięki temu uniknięto konieczności iteracyjnego sumowania wag krawędzi podczas pobierania informacji o koszcie rozwiązania.

Dla szczególnego przypadku, gdy waga krawędzi t_{ij} jest równa zero, wartość funkcji oceny jakości wyrażona w normalny przypadku wzorem $\eta_{ij} = 1/t_{ij}$, przyjmuje wartość 2. Dane testowe wykorzystane podczas badań posiadają koszty wyrażone w postaci liczb całkowitych, dlatego też krawędź o zerowym koszcie jest preferowana przez funkcję oceny jakości przed wszystkimi innymi krawędziami.

4.2. MODYFIKACJE WPROWADZONE W ALGORYTMIE EACS

Podrozdział ten zawiera opis badanych implementacji oraz modyfikacji algorytmu EACS.

4.2.1. Przerywanie budowania kosztownych rozwiązań

Budowane przez mrówkę rozwiązanie na pewno zostanie odrzucone przez algorytm EACS, jeżeli jego koszt przekroczy koszt dotychczas znalezionego najlepszego rozwiązania o więcej niż 20%. Proponowana modyfikacja polega na tym, aby w trakcie budowania rozwiązania sprawdzać, czy wartość ta została przekroczona, a gdy to nastąpi, porzucić dalsze budowanie rozwiązania. W ten sposób pomija się zbędne operacje związane z wędrówką mrówki, której ścieżka i

tak jest niskiej jakości. Z drugiej jednak strony dodawany jest pewien narzut obliczeniowy związany ze sprawdzaniem, czy przekroczono wartość progową. Dzięki temu, że koszt budowanego rozwiązania jest modyfikowany na bieżąco i nie ma konieczności iteracyjnego sumowania kosztów krawędzi, narzut ten sprowadza się do $n - 1$ operacji porównania.

4.2.2. Licznik spełnionych ograniczeń

Podczas budowania rozwiązania mrówka musi wielokrotnie sprawdzać, czy dla danego węzła wszystkie ograniczenia zostały spełnione. Przechowywanie informacji o tym zaimplementowano na dwa różne sposoby i uzależniono od parametru algorytmu.

Pierwszy sposób wykorzystuje właściwość IsMet klasy Restriction. Gdy mrówka odwiedza dany węzeł, oznacza wszystkie ograniczenia z listy RequiredBy jako spełnione. Samo sprawdzanie polega na weryfikacji, czy wszystkie ograniczenia z listy Restriction danego węzła są oznaczone jako spełnione.

Drugi sposób wykorzystuje licznik ograniczeń, które muszą zostać spełnione, aby dany węzeł mógł zostać odwiedzony. Mrówka, podczas odwiedzania węzła, dla każdego elementu listy RequiredBy odwołuje się do wierzchołka, którego odwiedzenie jest ograniczone (właściwość TargetToVisit) i dekrementuje jego licznik RestrictionsToMeet. Sprawdzenie czy wszystkie ograniczenia zostały spełnione sprowadza się do porównania licznika z zerem. Dzięki temu unika się iterowania listy ograniczeń związanych z danym węzłem.

4.2.3. Dodawanie feromonu proporcjonalnie do kosztu

Dodawanie feromonu odbywa się podczas globalnej aktualizacji śladów feromonowych. Proponowana modyfikacja ma na celu zbliżyć proces symulacji działania mrówek do sytuacji rzeczywistej. Feromon w środowisku naturalnym jest rozkładany równomiernie na całej trasie, natomiast w algorytmach mrówkowych ilość feromonu dodawanego do krawędzi nie jest zależna od jej wagi. Z punktu widzenia algorytmu nieistotne jest, czy koszt przebycia krawędzi jest bardzo duży i równy niemal całemu kosztowi rozwiązania, czy raczej jest bliski zeru. Do każdej krawędzi dodawana jest ta sama ilość feromonu. Proponowana modyfikacja polega na zmianie wzoru odpowiedzialnego za globalną aktualizację feromonu do następującej formy:

$$\tau_{ij} = (1-\rho) * \tau_{ij} + \left(t_{ij}/t_{avg}\right) * \rho/L_{best}$$

Parametr t_{avg} oznacza średni koszt drogi w najlepszym dotychczas znalezionym rozwiązaniu. Dzięki tej modyfikacji ilość pozostawionego feromonu jest większa na krawędziach długich i odpowiednio mniejsza na krawędziach krótkich.

Podczas implementacji przyjęto, że w przypadku, gdy $t_{ij} = 0$ w miejsce t_{ij} wstawiana jest wartość $0{,}5$. Bez tego założenia do krawędzi o koszcie równym zero nigdy nie zostałby dodany ślad feromonowy.

4.2.4. Wyparowywanie feromonu proporcjonalnie do kosztu

Modyfikacja ta jest motywowana podobnymi pobudkami, co modyfikacja dodawania śladu feromonowego przedstawiona w punkcie 4.2.3. Ma ona na celu zwiększenie ilości odparowującego feromonu na długich krawędziach i zmniejszenie tej ilości dla krawędzi krótkich. Uzyskano to poprzez modyfikację wzoru globalnej aktualizacji feromonu do formy:

$$\tau_{ij} = (1-\rho') * \tau_{ij} + * \rho/L_{best}$$

Parametr ρ' wyraża się wzorem:

$$\rho' = \begin{cases} \rho * (t_{ij}/t_{avg}), & \rho * (t_{ij}/t_{avg}) < 1 \\ 1, & \rho * (t_{ij}/t_{avg}) \geq 1 \end{cases}$$

Również podczas implementacji tej modyfikacji przyjęto, że w przypadku, gdy $t_{ij} = 0$ w miejsce t_{ij} wstawiana jest wartość $0{,}5$. Bez tego założenia zastosowanie globalnej aktualizacji feromonu dla krawędzi o zerowym koszcie zawsze powodowałoby zupełne odparowanie feromonu.

4.2.5. Początkowa wartość feromonu na krawędzi zależna od kosztu

Jest to trzecia modyfikacja uzależniająca ilość feromonu na krawędzi od jej kosztu. Przed przypisaniem wartości początkowej τ_{init} do krawędzi następuje jej przemnożenie przez proporcję kosztu zgodnie ze wzorem:

$$\tau_{ij} = \tau_{init} * (t_{ij}/t_{favg})$$

Parametr t_{favg} to średni koszt krawędzi należących do pierwszego wygenerowanego rozwiązania z pominięciem etapu lokalnej optymalizacji.

Ponownie podczas implementacji przyjęto, że w przypadku, gdy $t_{ij} = 0$ w miejsce t_{ij} wstawiana jest wartość $0{,}5$. Bez tego założenia początkowa wartość feromonu na krawędziach o zerowym koszcie byłaby równa zero.

4.2.6. Wybór deterministyczny jedynym sposobem wyboru krawędzi należącej do najlepszego rozwiązania

Podczas obserwacji zachowania algorytmu EACS zauważono, że mrówki pomimo wylosowania probabilistycznego sposobu wyboru kolejnej krawędzi, często podążają za drogą z najlepszego rozwiązania. Efekt ten staje się coraz bardziej widoczny wraz z kolejnymi iteracjami. Jest to powodowane tym, że po każdej z nich ilość feromonu na krawędziach najlepszego rozwiązania zwiększa się. Dodatkowo krawędzie te niejednokrotnie są preferowane przez funkcję oceny jakości, dlatego też prawdopodobieństwo ich wyboru jest bardzo duże.

Proponowana modyfikacja algorytmu EACS polega na uniemożliwieniu wyboru kolejnego węzła zgodnie z najlepszym znalezionym rozwiązaniem, gdy mrówka wylosowała probabilistyczny sposób wyłonienia kolejnego kroku swojej podróży.

4.2.7. Zmniejszanie znaczenia heurystyki z kolejnymi iteracjami

W początkowej fazie działania heurystyka, wyrażona w funkcji oceny jakości krawędzi, ma na celu naprowadzenie algorytmu w najbardziej obiecujące obszary przestrzeni rozwiązań. Jednak w dalszych etapach pracy, gdy eksploracja w znacznym stopniu wyczerpała sąsiedztwo najlepszego rozwiązania, heurystyka może utrudniać szukanie lepszych rezultatów w obszarach jeszcze niezbadanych.

Proponowana modyfikacja ma na celu stopniowe zmniejszanie znaczenia heurystyki, a dzięki temu poszerzenie obszaru przeszukiwań w późnych etapach pracy algorytmu. Osiągnięto to poprzez zmianę wzoru funkcji oceny jakości na:

$$\eta_{ij} = 1/{t_{ij}}^{\omega_k}$$

Parametr ω_k na początku działania przyjmuje wartość ω_0 będącą parametrem algorytmu, a następnie z każdą kolejną iteracją o numerze k jest modyfikowany według wzoru:

$$\omega_k = \omega_{k-1} * (1 - \gamma)$$

Parametr γ jest parametrem algorytmu nazwanym współczynnikiem zmniejszania znaczenia heurystyki.

4.2.8. Zwiększanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny

Zmiana ta ma na celu stopniowe poszerzanie obszaru poszukiwań poprzez zwiększanie parametru s, który reprezentuje oczekiwaną liczbę węzłów wybranych w sposób probabilistyczny. Parametr ten traktuje się jako liczbę zmiennoprzecinkową, a nie całkowitą. W modyfikacji można wyróżnić trzy elementy: zwiększanie s poprzez mnożenie, zwiększanie s poprzez dodawanie oraz wartość graniczną parametru s. Zwiększanie może być realizowane przez dowolne kombinacje mnożenia i dodawania, jednak ustalenie wartości granicznej jest nieodzownym elementem tej modyfikacji. Odbywa się to za pomocą parametru s_b, który jest wykorzystywany do wyznaczenia wartości maksymalnej parametru s według wzoru:

$$s_{max} = s_b * n$$

Parametr ten przyjmuje wartości należące do przedziału $\langle 0, 1\rangle$.

W zwiększaniu przez mnożenie w kolejnych iteracjach algorytmu zmienia się wartość s poprzez przemnażanie go przez μ. Na początku działania $s = s_0$, gdzie s_0 jest parametrem algorytmu. Następnie $s = s_k$, gdzie parametr s_k jest modyfikowany z każdą kolejną iteracją o numerze k w następujący sposób:

$$s_k = s_{k-1} * \mu$$

Można również ustalić, czy wartość s ma być resetowana do wartości s_0 w przypadku znalezienia lepszego rozwiązania. Dzięki temu obszar przeszukiwania zmniejsza się do bliskiego sąsiedztwa najlepszego rozwiązania w przypadku jego znalezienia.

Zwiększanie przez dodawanie polega na zmianie wzoru na prawdopodobieństwo q_0 do następującej formy:

$$q_0 = 1 - (s + l * \sigma)/n$$

Parametr l określa liczbę iteracji bez poprawy najlepszego znalezionego rozwiązania, natomiast σ jest parametrem algorytmu.

4.2.9. *Wymuszanie przynajmniej jednej zmiany w stosunku do najlepszego rozwiązania*

Podczas obserwacji zachowania algorytmu EACS zauważono, że mrówki wielokrotnie wyznaczają dokładnie to samo najlepsze rozwiązanie. Proponowana modyfikacja ma na celu wymuszenie wyboru przynajmniej jednej niewchodzącej w jego skład krawędzi. Mrówka zanim rozpocznie proces iteracyjnego budowania rozwiązania losuje numer iteracji, w której wymusi wybranie innej drogi. Wymuszenie to nie będzie miało miejsca, jeżeli w czasie jego potencjalnego wykonania wiadomo, że budowane rozwiązanie będzie różne od najlepszego dotychczas znalezionego. Ta operacja może zakończyć się porażką, gdy krawędź wykorzystana w rozwiązaniu najlepszym jest jedyną, która w danym momencie może zostać wykorzystana do zbudowania poprawnej ścieżki. W takim przypadku próba wymuszenia wybrania innej drogi jest podejmowana w kolejnych iteracjach, aż do skutku lub zakończenia budowania rozwiązania.

4.2.10. *Zjadanie feromonu podczas lokalnej optymalizacji*

Celem zjadania feromonu jest zmniejszenie atrakcyjności użytych krawędzi dla innych mrówek, a tym samym osiągnięcie większego zróżnicowania sprawdzanych rozwiązań. W algorytmie EACS zjadanie feromonu odbywa się podczas budowania ścieżki przez mrówkę. Należy jednak zauważyć, że podczas lokalnej optymalizacji rozważanych jest wiele innych możliwości, a ilość feromonu na krawędziach do nich należących nie jest wtedy zmniejszana.

Proponowana modyfikacja ma na celu zmniejszenie atrakcyjności krawędzi należących do rozwiązań rozważanych podczas lokalnej optymalizacji. Istnieją dwa warianty tej modyfikacji. W pierwszym z nich zjadanie feromonu odbywa się na całej trasie ocenianej podczas lokalnej optymalizacji. Drugi wariant polega na zjadaniu feromonu jedynie na krawędziach nowych z perspektywy ulepszanego rozwiązania.

4.2.11. *Akceptowanie rozwiązania o tym samym koszcie*

Po wprowadzeniu tej modyfikacji algorytm akceptuje rozwiązanie jako najlepsze nie tylko w momencie, gdy jego koszt jest niższy od dotychczas znalezionego najlepszego rezultatu, ale również, gdy jest on taki sam. Nie jest to traktowane jako znalezienie rozwiązania lepszego – przykładowo dla modyfikacji z

punktu 4.2.8 nie nastąpi resetowanie wartości s nawet, gdy jest włączone. Sprawia to jednak, że globalna aktualizacja feromonów odbywa się na innej drodze, a tym samym przestrzeń sąsiedztwa najlepszego rozwiązania zmienia się na potencjalnie bardziej atrakcyjną.

4.2.12. Wyłączenie heurystyki z algorytmu

Modyfikacja ta polega na zmianie wzoru na prawdopodobieństwo przejścia do węzła p_{ij} w przypadku zastosowania podejścia probabilistycznego na:

$$p_{ij} = \frac{\tau_{ij}}{\sum_{l \in F(i)} \tau_{il}}$$

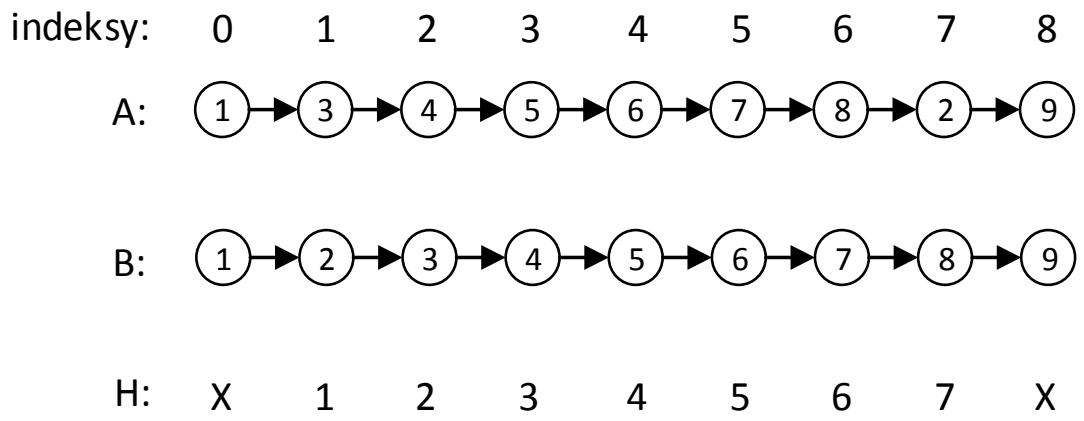
Motywacja zastosowania takiej zmiany ma źródło w analizie zachowania algorytmu przy wykorzystaniu przerywania budowania zbyt kosztownych rozwiązań opisanego w punkcie 4.2.1. Okazuje się, że przyrost kosztu budowanych ścieżek jest znacznie większy w końcowej fazie ich tworzenia. Powodem takiego zachowania jest silne preferowanie dróg o niskim koszcie. Na początku procesu budowania rozwiązania algorytm nie ma problemu ze znalezieniem takich dróg, jednak często w końcowej fazie wszystkie wagi możliwych do wyboru krawędzi są wysokie. Z tego też powodu tworzone rozwiązania często przekraczają próg przerwania z punktu 4.2.1 na późnym etapie pracy i zysk z samego przerwania jest niewielki. Należy również zauważyć, że z powodu heurystyki niemal wcale nie przeszukuje się rozwiązań, w których początkowo wybierane krawędzie mają wysokie koszty. Proponowana modyfikacja ma na celu wyeliminowanie tego efektu, a w połączeniu z modyfikacją z punku 4.2.1 nie powoduje ona aż tak dużej straty czasu na rozważanie mało obiecujących przestrzeni.

4.2.13. Użycie wykorzystanych dróg podczas inicjalizacji stosu w procesie lokalnej optymalizacji[5]

W algorytmie EACS indeks i należy dodać do stosu, jeżeli dla ulepszanej drogi przedstawionej w postaci kolejnych odwiedzanych węzłów, i-ty węzeł jest

[5] Jak już wspomniano w podrozdziale 3.4 opis inicjalizacji stosu wartości h zawarty w pracy [7] nie jest precyzyjny, dlatego też trudno jednoznacznie stwierdzić, czy przedstawione tutaj podejście jest faktyczną modyfikacją, czy raczej poprawną implementacją algorytmu EACS.

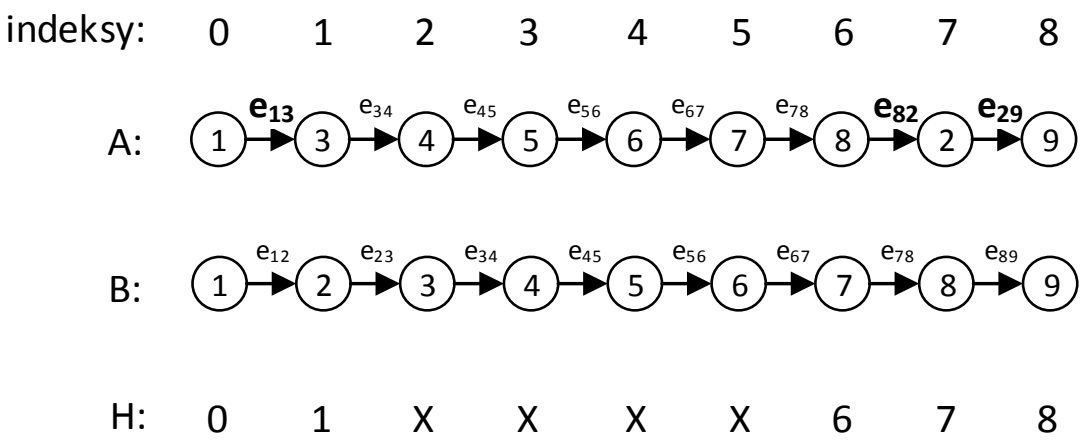
różny od i-tego węzła w najlepszym dotychczas znalezionym rozwiązaniu. Podejście to jednak sprawia, że prawdopodobne jest wielokrotne podjęcie prób poprawy tego samego ciągu krawędzi. Zostanie teraz zaprezentowany przykład takiej sytuacji. Dane są dwa rozwiązania: ulepszane A i najlepsze dotychczas znalezione B. Rysunek 4.2 przedstawia proces inicjalizacji stosu H.



Rysunek 4.2 Proces inicjalizacji stosu podczas lokalnej optymalizacji dla algorytmu EACS

Można zauważyć, że w tym przypadku do stosu trafią niemal wszystkie możliwe wartości. Łatwo jednak zauważyć, że droga od węzła 3 do węzła 8 powtarza się w obu rozwiązaniach. Wszystkie zamiany rozwiązania A zawierające się pomiędzy indeksami 1 do 6 zostały już zbadane podczas budowania B, a ich wykonanie nie przyniosło poprawy (gdyby któraś z nich powodowała poprawę, zostałaby wykonana i B miałoby inną formę).

Proponowana modyfikacja polega na inicjalizacji stosu na podstawie zbioru krawędzi, które nie występują w najlepszym rozwiązaniu. Rysunek 4.3 prezentuje zmodyfikowaną inicjalizację stosu.



Rysunek 4.3 Zmodyfikowana inicjalizacja stosu podczas lokalnej optymalizacji

Na rysunku krawędzie oznaczono za pomocą litery e oraz pogrubiono te, które nie występują w B i zostały użyte do wyznaczenia zawartości stosu H.

4.2.14. Brak odparowywania feromonu podczas globalnej aktualizacji

W pierwszych wersjach algorytmów mrówkowych odparowywanie feromonu odbywało się na wszystkich krawędziach. W algorytmie EACS jest ono aplikowane jedynie na krawędziach należących do najlepszego rozwiązania podczas procesu globalnej aktualizacji śladu feromonowego. W trakcie tego samego procesu do tych samych krawędzi jest dodawany ślad feromonowy, aby zwiększyć atrakcyjność obiecujących dróg. Działania tych efektów niwelują się wzajemnie. Dominującym tutaj jest dodawanie feromonu, więc po zaaplikowaniu globalnej aktualizacji ślad feromonowy na krawędziach z najlepszego rozwiązania się zwiększa.

Proponowana modyfikacja polega na wyeliminowaniu efektu wyparowywania feromonu. Dzięki temu globalna aktualizacja nie wykonuje dwóch przeciwnych działań. Feromon nadal jest pomniejszany poprzez zjadanie go przez mrówki.

5. BADANIA EKSPERYMENTALNE

W tym rozdziale opisane zostały wykonane badania eksperymentalne oraz wynikające z nich wnioski.

5.1. DANE WYKORZYSTANE PODCZAS BADAŃ[6]

Do badań wykorzystano istniejący zbiór 48 problemów sekwencyjnego porządkowania znany pod nazwą *SOPLIB2006* [10]. Każda instancja problemu z tej biblioteki jest zapisana w osobnym pliku tekstowym o rozszerzeniu *.sop*. Ich nazewnictwo jest znormalizowane do postaci $\mathrm{R}.n.r.p$, gdzie

- n oznacza liczbę węzłów;
- r oznacza maksymalny koszt krawędzi;
- p reprezentuje przybliżony, procentowy stosunek liczby ograniczeń do liczby krawędzi w problemie. Liczba ograniczeń występujących w danym problemie wynosi około $(p/100) * (n * (n-1)/2)$.

Opisane parametry przyjmują następujące wartości:

- $n \in \{200, 300, 400, 500, 600, 700\}$;
- $r \in \{100, 1000\}$
- $p \in \{1, 15, 30, 60\}$

Przykładowo plik o nazwie *R.400.1000.30.sop* opisuje problem składający się z 400 węzłów połączonych krawędziami o wagach należących do zbioru $\{0, 1, 2, 3, \dots, 1000\}$. Dodatkowo występuje w nim około $(30/100) * (400 * (400-1)/2) = \mathbf{23940}$ ograniczeń.

Pliki zawierają opis problemu w postaci macierzy łączącej informacje z macierzy kosztów i macierzy ograniczeń. Ten sposób reprezentacji został opisany w podrozdziale 3.2. Na rysunku 5.1 zaprezentowano fragment pliku *R.400.1000.30.sop.*

[6] Do opracowania opisu biblioteki SOPLIB2006 wykorzystano pracę [11], w której została ona przedstawiona po raz pierwszy.

```
0   0   0   0   0   0   0   0   0
-1  0   -1  938 -1  431 787 747 845
-1  111 0   511 936 217 -1  319 -1
-1  618 754 0   537 320 -1  -1  959
-1  614 838 469 0   -1  714 129 395
-1  464 555 272 774 0   -1  892 55
-1  127 481 666 321 295 0   865 688
-1  672 461 391 176 417 228 0   -1
-1  74  838 998 647 108 674 0   0
```

Rysunek 5.1 Fragment pliku R.400.1000.30.sop *należącego do biblioteki SOPLIB2006*

Dane zawarte w plikach biblioteki *SOPLIB2006* posiadają pewną ciekawą właściwość. Wszystkie ograniczenia, które nie dotyczą drogi do węzła 0, ani drogi z węzła n, występują powyżej głównej przekątnej macierzy. Powoduje to nierównomierny rozkład ograniczeń pomiędzy węzłami – im mniejszy indeks węzła tym więcej ograniczeń. Właściwość ta nie została opisana przez autora biblioteki, dlatego też nie wiadomo, jaki był cel spreparowania danych w taki sposób.

5.2. PARAMETRY ALGORYTMU

Przyjęto następujące wartości parametrów:

- Współczynnik wyparowania feromonów: $\rho = 0{,}1$
- Liczba mrówek: $m = 10$
- Oczekiwana liczbę węzłów wybranych w sposób probabilistyczny: $n = 10$
- Współczynnik zjadania feromonów: $\varphi = 0{,}1$
- Początkowa ilość feromonu na krawędziach: $\tau_0 = (FS * n)^{-1}$

Dodatkowo przyjęto, że w oryginalnym EACS zastosowano licznik spełnionych ograniczeń opisany w punkcie 4.2.2.

5.3. ŚRODOWISKO TESTOWE

Zaimplementowany program został skompilowany w trybie Release dla procesorów 64-bitowych. Wszystkie badania przeprowadzono przy użyciu komputera z systemem Windows 8.1, wyposażonego w procesor Intel Core

i7-2600K o prędkości taktowania ustawionej na 3,74 GHz. Implementacja programu badawczego umożliwia zrównoleglenie prowadzonych badań, a procesor i7-2600K dzięki technologii Hyper-Threading posiada osiem procesorów logicznych, jednak w trakcie symulacji wykorzystywano maksymalnie pięć z nich. Dzięki temu działania systemu niezwiązane z badaniami mogły być wykonywane z wykorzystaniem wolnych zasobów i bez odzwierciedlenia na wynikach.

5.4. ZAIMPLEMENTOWANY MECHANIZM TESTUJĄCY

W programie wykorzystywanym do badań zaimplementowano mechanizm oparty o algorytm symulowanego wyżarzania, mający na celu dobór jak najlepszych parametrów wykonania algorytmu. Mechanizm ten wielokrotnie wykonuje badanie algorytmu mrówkowego na zadanym problemie zmieniając wskazane parametry i zapamiętując ten zestaw, dla którego uzyskano najlepszy wynik. Proces ten trwa do momentu przekroczenia zadanego czasu wykonania.

5.5. WYNIKI BADAŃ POSZCZEGÓLNYCH MODYFIKACJI I ICH KOMBINACJI

Dla każdej konfiguracji wykonano dziesięć symulacji z czasem wykonania ograniczonym do dziesięciu minut.

5.5.1. Przerywanie budowania kosztownych rozwiązań

W tabeli 5.1 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zastosowano przerywanie budowania kosztownych rozwiązań, opisane w punkcie 4.2.1.

Tabela 5.1 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zastosowano przerywanie budowania kosztownych rozwiązań

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	73,4	2,00%	11381,7	11759,9	3,32%
R.200.100.15.sop	1925,5	1962,1	-1,90%	21796,4	17130,6	-21,41%
R.200.100.30.sop	4236,2	4238,9	-0,06%	1574,3	1500,9	-4,66%
R.200.100.60.sop	71749	71773,9	-0,03%	5451	5253,3	-3,63%
R.200.1000.1.sop	1471,5	1467,6	0,27%	4337,2	3733,8	-13,91%
R.200.1000.15.sop	22362,9	22244,5	0,53%	9097,7	8723,3	-4,12%
R.200.1000.30.sop	41353,3	41362,8	-0,02%	2006,8	1948,2	-2,92%

R.200.1000.60.sop	71570,5	71614	-0,06%	4432	4550	2,66%
R.300.100.1.sop	46,2	44,9	2,81%	3947,9	3718,2	-5,82%
R.300.100.15.sop	3528,1	3523,6	0,13%	965,3	1136,1	17,69%
R.300.100.30.sop	6204,3	6206	-0,03%	541,9	525,4	-3,04%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2298	1,18%
R.300.1000.1.sop	1462,9	1462,6	0,02%	848	861,1	1,54%
R.300.1000.15.sop	33301,4	33645,9	-1,03%	1532,1	1507,4	-1,61%
R.300.1000.30.sop	55545,1	55279,3	0,48%	331,4	304,2	-8,21%
R.300.1000.60.sop	109783,9	109728,5	0,05%	1586,7	1551,8	-2,20%
R.400.100.1.sop	39,9	44,2	-10,78%	950,9	993,4	4,47%
R.400.100.15.sop	4761,9	4748,1	0,29%	266,6	260,5	-2,29%
R.400.100.30.sop	8511	8521,2	-0,12%	158,5	163,9	3,41%
R.400.100.60.sop	15273,6	15271,6	0,01%	593,8	594,3	0,08%
R.400.1000.1.sop	1575,4	1588,2	-0,81%	229,3	187,8	-18,10%
R.400.1000.15.sop	46890,7	47106,4	-0,46%	308	280	-9,09%
R.400.1000.30.sop	87839,1	87478,9	0,41%	173,9	161,7	-7,02%
R.400.1000.60.sop	141531,9	141576,7	-0,03%	650	622,1	-4,29%
R.500.100.1.sop	34,9	37,5	-7,45%	452	419,2	-7,26%
R.500.100.15.sop	6701	6684,4	0,25%	68,6	87,8	27,99%
R.500.100.30.sop	10336,8	10333,1	0,04%	82,8	76,4	-7,73%
R.500.100.60.sop	18407,9	18528,3	-0,65%	275,2	282,6	2,69%
R.500.1000.1.sop	1683,1	1652,7	1,81%	88,1	90,3	2,50%
R.500.1000.15.sop	62426,6	61816,2	0,98%	79,2	108	36,36%
R.500.1000.30.sop	104370,2	104709,4	-0,32%	90,6	84,6	-6,62%
R.500.1000.60.sop	179575,3	179952,4	-0,21%	364,6	356	-2,36%
R.600.100.1.sop	29,2	30,2	-3,42%	304	338,5	11,35%
R.600.100.15.sop	7650,8	7640,4	0,14%	30,8	32,2	4,55%
R.600.100.30.sop	13411,5	13636,3	-1,68%	63,8	55,9	-12,38%
R.600.100.60.sop	23996,6	24006,9	-0,04%	209,9	208,4	-0,71%
R.600.1000.1.sop	1799,9	1806,7	-0,38%	47,5	43,6	-8,21%
R.600.1000.15.sop	74328,1	74580,6	-0,34%	49,2	47,2	-4,07%
R.600.1000.30.sop	137129,7	137587,8	-0,33%	55,2	47	-14,86%
R.600.1000.60.sop	224034,2	225694,5	-0,74%	164,4	182,6	11,07%
R.700.100.1.sop	26,6	27,2	-2,26%	167,4	179,2	7,05%
R.700.100.15.sop	9604,7	9546	0,61%	15,6	13,7	-12,18%
R.700.100.30.sop	16282,2	16220,1	0,38%	37,5	34,9	-6,93%
R.700.100.60.sop	25309,8	25255,7	0,21%	148,3	148,8	0,34%
R.700.1000.1.sop	1750,2	1767,2	-0,97%	31,3	26,3	-15,97%
R.700.1000.15.sop	92592	92005,8	0,63%	18,2	18,4	1,10%
R.700.1000.30.sop	148859,8	150114,2	-0,84%	38,1	31,6	-17,06%
R.700.1000.60.sop	257040,2	258286,3	-0,48%	149,9	160,5	7,07%
		Średnio:	**-0,49%**		**Średnio:**	**-1,71%**

Niestety proponowana modyfikacja przynosi pogorszenie otrzymywanych wyników. Dla problemu z pliku *R.400.100.1.sop* pogorszenie wynosi 10,78%,

jednak należy zauważyć, że średnie koszty (EACS: $39{,}9$; zEACS: $44{,}2$) różnią się nieznacznie, biorąc pod uwagę fakt, że koszt jednej krawędzi w tym problemie może wynosić 100. Należy też zauważyć, że liczba iteracji w dużym stopniu zależy od tego jak, długo trwa lokalna optymalizacja, co z kolei jest zależne od różnic pomiędzy rozwiązaniem poprawianym a najlepszym. Dlatego też wahania w przypadku tej wartości są znacznie większe niż w przypadku kosztu.

Pogorszenie otrzymywanych rezultatów może płynąć z dodatkowego narzutu związanego ze sprawdzaniem, czy po dodaniu kolejnej krawędzi sumaryczny koszt rozwiązania przekroczył już wartość graniczną. Dodatkowo przyrost kosztu budowanych ścieżek jest znacznie większy w końcowej fazie ich tworzenia, co sprawia, że przerwanie następuje po wykonaniu większości obliczeń, a zysk z jego zastosowania jest niewielki. Należy również zauważyć, że zatrzymując budowanie rozwiązania przedwcześnie powstrzymujemy mrówkę od zjadania feromonu na krawędziach, które zostałyby wykorzystane przy zastosowaniu standardowej wersji algorytmu EACS.

5.5.2. Licznik spełnionych ograniczeń

W tabeli 5.2 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której wyłączono użycie licznika spełnionych ograniczeń. Jego opis znajduje się w punkcie 4.2.2.

Tabela 5.2 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której wyłączono użycie licznika spełnionych ograniczeń

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	74,3	0,80%	11381,7	10744,7	-5,60%
R.200.100.15.sop	1925,5	1987,2	-3,20%	21796,4	16660,6	-23,56%
R.200.100.30.sop	4236,2	4235,3	0,02%	1574,3	1453,7	-7,66%
R.200.100.60.sop	71749	71749	0,00%	5451	4810,2	-11,76%
R.200.1000.1.sop	1471,5	1468,7	0,19%	4337,2	3624,4	-16,43%
R.200.1000.15.sop	22362,9	22035,9	1,46%	9097,7	8012,8	-11,92%
R.200.1000.30.sop	41353,3	41353,3	0,00%	2006,8	1924	-4,13%
R.200.1000.60.sop	71570,5	71614	-0,06%	4432	4174,6	-5,81%
R.300.100.1.sop	46,2	44,8	3,03%	3947,9	4178,7	5,85%
R.300.100.15.sop	3528,1	3514,7	0,38%	965,3	1131,5	17,22%
R.300.100.30.sop	6204,3	6202,4	0,03%	541,9	500,8	-7,58%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2065,3	-9,07%

R.300.1000.1.sop	1462,9	1440,2	1,55%	848	799,2	-5,75%
R.300.1000.15.sop	33301,4	33336,6	-0,11%	1532,1	1430	-6,66%
R.300.1000.30.sop	55545,1	55343,3	0,36%	331,4	328,7	-0,81%
R.300.1000.60.sop	109783,9	109835,4	-0,05%	1586,7	1416,1	-10,75%
R.400.100.1.sop	39,9	43,8	-9,77%	950,9	899,9	-5,36%
R.400.100.15.sop	4761,9	4772,3	-0,22%	266,6	232	-12,98%
R.400.100.30.sop	8511	8536,7	-0,30%	158,5	156,3	-1,39%
R.400.100.60.sop	15273,6	15276,4	-0,02%	593,8	558	-6,03%
R.400.1000.1.sop	1575,4	1593,1	-1,12%	229,3	204,9	-10,64%
R.400.1000.15.sop	46890,7	46963,8	-0,16%	308	259,2	-15,84%
R.400.1000.30.sop	87839,1	87717,1	0,14%	173,9	173,7	-0,12%
R.400.1000.60.sop	141531,9	141465,3	0,05%	650	641,3	-1,34%
R.500.100.1.sop	34,9	34,9	0,00%	452	501,1	10,86%
R.500.100.15.sop	6701	6719,4	-0,27%	68,6	71,1	3,64%
R.500.100.30.sop	10336,8	10306,7	0,29%	82,8	79,9	-3,50%
R.500.100.60.sop	18407,9	18565,9	-0,86%	275,2	289,4	5,16%
R.500.1000.1.sop	1683,1	1637,2	2,73%	88,1	94,4	7,15%
R.500.1000.15.sop	62426,6	62595,1	-0,27%	79,2	82,9	4,67%
R.500.1000.30.sop	104370,2	104323,6	0,04%	90,6	93,1	2,76%
R.500.1000.60.sop	179575,3	179461,7	0,06%	364,6	330,7	-9,30%
R.600.100.1.sop	29,2	25,5	12,67%	304	322,4	6,05%
R.600.100.15.sop	7650,8	7662,6	-0,15%	30,8	37,3	21,10%
R.600.100.30.sop	13411,5	13486,6	-0,56%	63,8	62	-2,82%
R.600.100.60.sop	23996,6	23934,8	0,26%	209,9	223,8	6,62%
R.600.1000.1.sop	1799,9	1793	0,38%	47,5	49,7	4,63%
R.600.1000.15.sop	74328,1	74907,4	-0,78%	49,2	44,2	-10,16%
R.600.1000.30.sop	137129,7	136707,2	0,31%	55,2	47,3	-14,31%
R.600.1000.60.sop	224034,2	223817,7	0,10%	164,4	181	10,10%
R.700.100.1.sop	26,6	25	6,02%	167,4	206,1	23,12%
R.700.100.15.sop	9604,7	9545	0,62%	15,6	17	8,97%
R.700.100.30.sop	16282,2	16106,4	1,08%	37,5	40,9	9,07%
R.700.100.60.sop	25309,8	25267,3	0,17%	148,3	152	2,49%
R.700.1000.1.sop	1750,2	1775,7	-1,46%	31,3	26,1	-16,61%
R.700.1000.15.sop	92592	90801,3	1,93%	18,2	20	9,89%
R.700.1000.30.sop	148859,8	149785,9	-0,62%	38,1	37,5	-1,57%
R.700.1000.60.sop	257040,2	257270,1	-0,09%	149,9	163,8	9,27%
		Średnio:	**0,30%**		**Średnio:**	**-1,48%**

Badana implementacja miała na celu polepszenie uzyskiwanych rezultatów poprzez skrócenie czasu wykonywania iteracji, a przez to przebadanie większej przestrzeni możliwych rozwiązań w tym samym czasie. Pomimo iż średnio uzyskano poprawę rezultatów na poziomie $0{,}3\%$, to jednak średnia liczba wykonanych iteracji zmniejszyła się o $1{,}48\%$. Wyniki te nie określają jednoznacznie oddziaływania proponowanej modyfikacji. Tak małe wahania

wartości średnich mogą być spowodowane zróżnicowanym czasem wykonania lokalnej optymalizacji. Wiadomo jednak, że wpływ zastosowania licznika jest nieznaczny.

5.5.3. Dodawanie feromonu proporcjonalnie do kosztu

W tabeli 5.3 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której dodawanie feromonu odbywa się proporcjonalnie do kosztu. Dokładny opis tej modyfikacji znajduje się w punkcie 4.2.3.

Tabela 5.3 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której dodawanie feromonu odbywa się proporcjonalnie do kosztu

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	75,4	-0,67%	11381,7	12387,9	8,84%
R.200.100.15.sop	1925,5	1960,2	-1,80%	21796,4	16507,2	-24,27%
R.200.100.30.sop	4236,2	4242,8	-0,16%	1574,3	1200,3	-23,76%
R.200.100.60.sop	71749	71749	0,00%	5451	4113,4	-24,54%
R.200.1000.1.sop	1471,5	1456,4	1,03%	4337,2	3719,8	-14,23%
R.200.1000.15.sop	22362,9	22332,2	0,14%	9097,7	8096,1	-11,01%
R.200.1000.30.sop	41353,3	41406,5	-0,13%	2006,8	1582,3	-21,15%
R.200.1000.60.sop	71570,5	71614	-0,06%	4432	3683,2	-16,90%
R.300.100.1.sop	46,2	46,5	-0,65%	3947,9	4926,4	24,79%
R.300.100.15.sop	3528,1	3583,3	-1,56%	965,3	813,8	-15,69%
R.300.100.30.sop	6204,3	6172,9	0,51%	541,9	423,5	-21,85%
R.300.100.60.sop	9726	9726	0,00%	2271,2	1525,9	-32,82%
R.300.1000.1.sop	1462,9	1466,8	-0,27%	848	798,5	-5,84%
R.300.1000.15.sop	33301,4	33695,6	-1,18%	1532,1	1205,8	-21,30%
R.300.1000.30.sop	55545,1	55526,5	0,03%	331,4	247,6	-25,29%
R.300.1000.60.sop	109783,9	109661,2	0,11%	1586,7	1043	-34,27%
R.400.100.1.sop	39,9	45,8	-14,79%	950,9	1215	27,77%
R.400.100.15.sop	4761,9	4760,4	0,03%	266,6	299,8	12,45%
R.400.100.30.sop	8511	8524,4	-0,16%	158,5	139,6	-11,92%
R.400.100.60.sop	15273,6	15238,8	0,23%	593,8	521,9	-12,11%
R.400.1000.1.sop	1575,4	1605,6	-1,92%	229,3	210,5	-8,20%
R.400.1000.15.sop	46890,7	46845,8	0,10%	308	281,1	-8,73%
R.400.1000.30.sop	87839,1	87630	0,24%	173,9	140,6	-19,15%
R.400.1000.60.sop	141531,9	141399,1	0,09%	650	503,9	-22,48%
R.500.100.1.sop	34,9	38,2	-9,46%	452	506,9	12,15%
R.500.100.15.sop	6701	6760,9	-0,89%	68,6	68,1	-0,73%
R.500.100.30.sop	10336,8	10335,6	0,01%	82,8	64,4	-22,22%

R.500.100.60.sop	18407,9	18497,5	-0,49%	275,2	218,5	-20,60%
R.500.1000.1.sop	1683,1	1665,4	1,05%	88,1	84,8	-3,75%
R.500.1000.15.sop	62426,6	62415,8	0,02%	79,2	65,1	-17,80%
R.500.1000.30.sop	104370,2	104929,6	-0,54%	90,6	64,4	-28,92%
R.500.1000.60.sop	179575,3	179745,7	-0,09%	364,6	268,3	-26,41%
R.600.100.1.sop	29,2	33,4	-14,38%	304	322,8	6,18%
R.600.100.15.sop	7650,8	7612,3	0,50%	30,8	27,6	-10,39%
R.600.100.30.sop	13411,5	13546	-1,00%	63,8	48,4	-24,14%
R.600.100.60.sop	23996,6	23938	0,24%	209,9	171,7	-18,20%
R.600.1000.1.sop	1799,9	1808,2	-0,46%	47,5	41,8	-12,00%
R.600.1000.15.sop	74328,1	76169,3	-2,48%	49,2	31,6	-35,77%
R.600.1000.30.sop	137129,7	137264,2	-0,10%	55,2	37,8	-31,52%
R.600.1000.60.sop	224034,2	223023,9	0,45%	164,4	156,9	-4,56%
R.700.100.1.sop	26,6	33,7	-26,69%	167,4	387,8	131,66%
R.700.100.15.sop	9604,7	9621,6	-0,18%	15,6	10,8	-30,77%
R.700.100.30.sop	16282,2	16219,3	0,39%	37,5	28,3	-24,53%
R.700.100.60.sop	25309,8	25184,5	0,50%	148,3	119,8	-19,22%
R.700.1000.1.sop	1750,2	1749,8	0,02%	31,3	27,9	-10,86%
R.700.1000.15.sop	92592	92186,3	0,44%	18,2	13,3	-26,92%
R.700.1000.30.sop	148859,8	151491,5	-1,77%	38,1	31,8	-16,54%
R.700.1000.60.sop	257040,2	255976,5	0,41%	149,9	126,5	-15,61%
		Średnio:	**-1,57%**		**Średnio:**	**-11,52%**

Z uzyskanych danych badawczych wynika, że proponowana modyfikacja powoduje pogorszenie otrzymywanych rezultatów. Najbardziej negatywne zmiany odnotowano dla problemów o parametrach: $n \geq 400$, $r = 100$ i $p = 1$. Średnie pogorszenie otrzymywanych rezultatów dla wszystkich pozostałych problemów wynosi $0{,}23\%$. Co ciekawe, dla wszystkich problemów o parametrach: $r = 100$ i $p = 1$ nastąpiło zwiększenie liczby wykonanych iteracji, a jednocześnie są to niemal jedyne instancje, dla których wystąpiła pozytywna zmiana w tym aspekcie. Wyjątkiem jest problem opisany w pliku *R.400.100.15.sop*.

Dodawanie feromonów odbywa się $i * (n - 1)$ razy, gdzie i jest liczbą wykonanych iteracji. Jest to stosunkowo niewiele w porównaniu np. do liczby akcji koniecznych do zbudowania rozwiązania przez mrówkę. Dlatego też powodu pogorszenia rezultatów otrzymywanych przez zEACS nie należy doszukiwać się w zwiększeniu złożoności tego procesu.

Należy zauważyć, że proponowana modyfikacja powoduje pozostawianie większej ilości feromonu na krawędziach długich, a co za tym idzie, zwiększenie ich atrakcyjności w stosunku do krawędzi krótkich. Prawdopodobnie powoduje to

przesunięcie obszaru poszukiwań w mniej obiecujące rejony. Zmiany w średniej liczbie iteracji sugerują, że dla większości przypadków powoduje to również wydłużenie procesu lokalnej optymalizacji. Wyjątkiem są wspomniane wcześniej problemy o parametrach: $r = 100$ i $p = 1$, oraz problem opisany w pliku *R.400.100.15.sop*.

5.5.4. Wyparowywanie feromonu proporcjonalnie do kosztu

W tabeli 5.4 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której wyparowywanie feromonu odbywa się proporcjonalnie do kosztu. Dokładny opis tej modyfikacji znajduje się w punkcie 4.2.4.

Tabela 5.4 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której wyparowywanie feromonu odbywa się proporcjonalnie do kosztu

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	74,8	0,13%	11381,7	10502,5	-7,72%
R.200.100.15.sop	1925,5	1958,2	-1,70%	21796,4	18922,1	-13,19%
R.200.100.30.sop	4236,2	4238,7	-0,06%	1574,3	1512	-3,96%
R.200.100.60.sop	71749	71749	0,00%	5451	4904,1	-10,03%
R.200.1000.1.sop	1471,5	1464,2	0,50%	4337,2	3853,2	-11,16%
R.200.1000.15.sop	22362,9	22391,7	-0,13%	9097,7	7995,8	-12,11%
R.200.1000.30.sop	41353,3	41407,5	-0,13%	2006,8	1940,8	-3,29%
R.200.1000.60.sop	71570,5	71614	-0,06%	4432	4337,4	-2,13%
R.300.100.1.sop	46,2	44,7	3,25%	3947,9	3652,5	-7,48%
R.300.100.15.sop	3528,1	3572,6	-1,26%	965,3	1036,7	7,40%
R.300.100.30.sop	6204,3	6200,7	0,06%	541,9	521,6	-3,75%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2360,6	3,94%
R.300.1000.1.sop	1462,9	1470	-0,49%	848	888	4,72%
R.300.1000.15.sop	33301,4	33828,7	-1,58%	1532,1	1476,2	-3,65%
R.300.1000.30.sop	55545,1	55453,1	0,17%	331,4	320,1	-3,41%
R.300.1000.60.sop	109783,9	109783,9	0,00%	1586,7	1521,3	-4,12%
R.400.100.1.sop	39,9	39,3	1,50%	950,9	1169,1	22,95%
R.400.100.15.sop	4761,9	4761,5	0,01%	266,6	230,1	-13,69%
R.400.100.30.sop	8511	8532,6	-0,25%	158,5	171,8	8,39%
R.400.100.60.sop	15273,6	15291,5	-0,12%	593,8	521,8	-12,13%
R.400.1000.1.sop	1575,4	1628,3	-3,36%	229,3	206,7	-9,86%
R.400.1000.15.sop	46890,7	46843,8	0,10%	308	276,8	-10,13%
R.400.1000.30.sop	87839,1	87483,6	0,40%	173,9	165,8	-4,66%

R.400.1000.60.sop	141531,9	141435,5	0,07%	650	597,7	-8,05%
R.500.100.1.sop	34,9	33,5	4,01%	452	399,9	-11,53%
R.500.100.15.sop	6701	6652	0,73%	68,6	76,1	10,93%
R.500.100.30.sop	10336,8	10328,3	0,08%	82,8	80,8	-2,42%
R.500.100.60.sop	18407,9	18517,2	-0,59%	275,2	263,4	-4,29%
R.500.1000.1.sop	1683,1	1650	1,97%	88,1	86,8	-1,48%
R.500.1000.15.sop	62426,6	62474	-0,08%	79,2	89,2	12,63%
R.500.1000.30.sop	104370,2	105008,2	-0,61%	90,6	82,1	-9,38%
R.500.1000.60.sop	179575,3	180131	-0,31%	364,6	322,3	-11,60%
R.600.100.1.sop	29,2	28,9	1,03%	304	226,9	-25,36%
R.600.100.15.sop	7650,8	7615,9	0,46%	30,8	33,3	8,12%
R.600.100.30.sop	13411,5	13509,9	-0,73%	63,8	64,2	0,63%
R.600.100.60.sop	23996,6	23978,9	0,07%	209,9	207,5	-1,14%
R.600.1000.1.sop	1799,9	1764,8	1,95%	47,5	44,7	-5,89%
R.600.1000.15.sop	74328,1	75851,3	-2,05%	49,2	45,6	-7,32%
R.600.1000.30.sop	137129,7	137014,4	0,08%	55,2	40,5	-26,63%
R.600.1000.60.sop	224034,2	225827,1	-0,80%	164,4	159	-3,28%
R.700.100.1.sop	26,6	28,8	-8,27%	167,4	198,1	18,34%
R.700.100.15.sop	9604,7	9657,9	-0,55%	15,6	15,1	-3,21%
R.700.100.30.sop	16282,2	16340	-0,35%	37,5	36,7	-2,13%
R.700.100.60.sop	25309,8	25136,1	0,69%	148,3	137,3	-7,42%
R.700.1000.1.sop	1750,2	1777,2	-1,54%	31,3	28,6	-8,63%
R.700.1000.15.sop	92592	91869,2	0,78%	18,2	18,8	3,30%
R.700.1000.30.sop	148859,8	149885,9	-0,69%	38,1	31,9	-16,27%
R.700.1000.60.sop	257040,2	256346,4	0,27%	149,9	145,4	-3,00%
		Średnio:	**-0,15%**		**Średnio:**	**-4,05%**

Rezultaty symulacji ukazują, iż proponowany algorytm zEACS daje gorsze wyniki od oryginalnego EACS. Pogorszenie to wynosi $0{,}15\%$ w przypadku średniego kosztu, natomiast dla średniej liczby wykonanych iteracji jest to $4{,}05\%$. Otrzymane wyniki nie wskazują na to, aby istniała jakaś zależność pomiędzy zmianą wydajności a którymś z parametrów problemu.

5.5.5. Początkowa wartość feromonu na krawędzi zależna od kosztu

W tabeli 5.5 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której początkowa wartość feromonu na krawędziach jest zależna od ich kosztu. Dokładny opis tej modyfikacji znajduje się w punkcie 4.2.5.

Tabela 5.5 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której początkowa wartość feromonu na krawędziach jest zależna od ich kosztu

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	74,3	0,80%	11381,7	22134,7	94,48%
R.200.100.15.sop	1925,5	1940,7	-0,79%	21796,4	19129,1	-12,24%
R.200.100.30.sop	4236,2	4232,3	0,09%	1574,3	1525,4	-3,11%
R.200.100.60.sop	71749	71773,9	-0,03%	5451	5126,7	-5,95%
R.200.1000.1.sop	1471,5	1462,8	0,59%	4337,2	12428,5	186,56%
R.200.1000.15.sop	22362,9	22146,4	0,97%	9097,7	8607,8	-5,38%
R.200.1000.30.sop	41353,3	41399,2	-0,11%	2006,8	2015,4	0,43%
R.200.1000.60.sop	71570,5	71556	0,02%	4432	4610,2	4,02%
R.300.100.1.sop	46,2	45,5	1,52%	3947,9	14469,5	266,51%
R.300.100.15.sop	3528,1	3469	1,68%	965,3	1245,2	29,00%
R.300.100.30.sop	6204,3	6175,9	0,46%	541,9	457,6	-15,56%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2162,7	-4,78%
R.300.1000.1.sop	1462,9	1434,1	1,97%	848	6939	718,28%
R.300.1000.15.sop	33301,4	33093,2	0,63%	1532,1	1734,5	13,21%
R.300.1000.30.sop	55545,1	54886,6	1,19%	331,4	333,3	0,57%
R.300.1000.60.sop	109783,9	109661,2	0,11%	1586,7	1617	1,91%
R.400.100.1.sop	39,9	41,6	-4,26%	950,9	7399,3	678,14%
R.400.100.15.sop	4761,9	4581,9	3,78%	266,6	445,3	67,03%
R.400.100.30.sop	8511	8392,5	1,39%	158,5	157,7	-0,50%
R.400.100.60.sop	15273,6	15282	-0,05%	593,8	655	10,31%
R.400.1000.1.sop	1575,4	1593,1	-1,12%	229,3	3604,4	1471,91%
R.400.1000.15.sop	46890,7	44829,4	4,40%	308	478,6	55,39%
R.400.1000.30.sop	87839,1	86784,7	1,20%	173,9	151,2	-13,05%
R.400.1000.60.sop	141531,9	141656,9	-0,09%	650	656,6	1,02%
R.500.100.1.sop	34,9	38,1	-9,17%	452	5821,4	1187,92%
R.500.100.15.sop	6701	6396,2	4,55%	68,6	125,6	83,09%
R.500.100.30.sop	10336,8	10160,2	1,71%	82,8	66,8	-19,32%
R.500.100.60.sop	18407,9	18407,1	0,00%	275,2	269,3	-2,14%
R.500.1000.1.sop	1683,1	1664,6	1,10%	88,1	2546,1	2790,01%
R.500.1000.15.sop	62426,6	60147,1	3,65%	79,2	126,3	59,47%
R.500.1000.30.sop	104370,2	103002,2	1,31%	90,6	66,7	-26,38%
R.500.1000.60.sop	179575,3	179745	-0,09%	364,6	329,7	-9,57%
R.600.100.1.sop	29,2	36	-23,29%	304	4004,5	1217,27%
R.600.100.15.sop	7650,8	7483,3	2,19%	30,8	53,3	73,05%
R.600.100.30.sop	13411,5	13092	2,38%	63,8	58,2	-8,78%
R.600.100.60.sop	23996,6	23773,2	0,93%	209,9	196,5	-6,38%
R.600.1000.1.sop	1799,9	1796,5	0,19%	47,5	2011,1	4133,89%
R.600.1000.15.sop	74328,1	72207,7	2,85%	49,2	65,9	33,94%

R.600.1000.30.sop	137129,7	135024,2	1,54%	55,2	38,7	-29,89%
R.600.1000.60.sop	224034,2	221840,3	0,98%	164,4	169,2	2,92%
R.700.100.1.sop	26,6	38,5	-44,74%	167,4	3613	2058,30%
R.700.100.15.sop	9604,7	9293,3	3,24%	15,6	24,2	55,13%
R.700.100.30.sop	16282,2	15885,1	2,44%	37,5	36	-4,00%
R.700.100.60.sop	25309,8	25055,5	1,00%	148,3	130,6	-11,94%
R.700.1000.1.sop	1750,2	1797,1	-2,68%	31,3	1726	5414,38%
R.700.1000.15.sop	92592	87339,2	5,67%	18,2	30,3	66,48%
R.700.1000.30.sop	148859,8	147673,8	0,80%	38,1	24,5	-35,70%
R.700.1000.60.sop	257040,2	254362,3	1,04%	149,9	143,5	-4,27%
		Średnio:	**-0,58%**		**Średnio:**	**428,24%**

Wyniki symulacji ukazują, że proponowana modyfikacja zEACS powoduje pogorszenie otrzymywanych wyników średnio o $0{,}58\%$. Należy jednak zauważyć, że wynik ten jest mocno zaniżany przez problemy o parametrach $n \geq 400$, $r = 100$ i $p = 1$. Nie bez znaczenia jest też fakt, że choć procentowe pogorszenie rezultatów dla tych instancji jest znaczące, to różnice w otrzymywanych wynikach są niewielkie w stosunku do maksymalnego możliwego kosztu ścieżki. W przypadku problemu *R.700.100.1.sop*, dla którego odnotowano największe procentowe pogorszenie wyniku, pogorszenie rzeczywiste wynosi $11{,}9$, natomiast maksymalny teoretyczny koszt ścieżki to $69\,900$. Średnia poprawa otrzymywanych rezultatów dla pozostałych problemów to 1,21%.

Zaobserwowano również duże wahania w ilości wykonanych iteracji, szczególnie dla problemów o parametrze $p = 1$. Największą poprawę, wynoszącą $5414{,}38\%$**,** odnotowano dla problemu *R.700.1000.1*. Jednocześnie wystąpiło pogorszenie otrzymywanych rezultatów o $2{,}68\%$. W algorytmie zEACS zmiany w inicjalizacji ilości feromonu powodują niwelowanie działania heurystyki w początkowej fazie działania algorytmu poprzez zwiększanie atrakcyjności długich krawędzi względem krótkich. To z kolei skutkuje rozpoczęciem przeszukiwania przestrzeni możliwych rozwiązań od rejonów innych niż w przypadku tradycyjnego EACS. Prawdopodobnie tak wielkie różnice w ilości iteracji dla problemów o parametrze $p = 1$ wynikają z przeszukiwania przestrzeni, dla której koszt kolejnych produkowanych rozwiązań był gorszy od najlepszego wyniku o ponad 20%. zEACS pomijał więc wykonanie kosztownej lokalnej optymalizacji przechodząc do kolejnej iteracji.

Kolejną właściwością rezultatów symulacji jest fakt, że dla problemów o parametrach $p = 1,\ r = 100$ i $n \geq 300$ średnia zmiana otrzymywanych wyników jest coraz gorsza wraz ze wzrostem parametru n. Powodów tej zależności również należy doszukiwać się w zmianie przeszukiwanej przestrzeni. Zmiana zaproponowana w zEACS powoduje uśrednienie atrakcyjności przestrzeni możliwych rozwiązań w początkowej fazie działania algorytmu, co z kolei skutkuje przeszukiwaniem miejsc mało obiecujących z punktu widzenia heurystyki, która preferuje drogi zaczynające się krótkimi ścieżkami. Oczywiście drogi zaczynające się ścieżkami długimi a kończące krótkimi również mogą być bardzo atrakcyjne, jednak w tradycyjnym EACS są zazwyczaj odrzucane przez heurystykę. Algorytm zEACS rozpoczyna odrzucanie rozwiązań zaczynających się od długich krawędzi dopiero, gdy mrówki wyznaczą słabe rozwiązania z tymi krawędziami. Następuje wtedy zjadanie feromonu na danej krawędzi, a im krawędź jest dłuższa, tym zmniejszenie jej atrakcyjności jest większe. Dzieje się tak dlatego, że w początkowej fazie działania zEACS na dłuższych krawędziach znajduje się więcej feromonu, dlatego też zjadana ilość jest większa, co bezpośrednio przekłada się na atrakcyjność krawędzi. Należy zauważyć, że wzrost przestrzeni możliwych rozwiązań względem wzrostu parametru n jest znacznie większy dla problemów o małej liczbie ograniczeń (mała wartość p). Dlatego dla dużych problemów potrzeba znacznie więcej czasu na odsianie przestrzeni mało obiecujących.

Główną zaletą zEACS jest branie pod uwagę obiecujących rejonów, które zostałyby odrzucone przez heurystykę w EACS. Umożliwia to znalezienie lepszych rozwiązań, lecz wiąże się z koniecznością sprawdzania sporych przestrzeni, dla których zwracane rezultaty nie są atrakcyjne. Dlatego też im dłuższy czas działania algorytmów tym bardziej prawdopodobne, że zEACS zwróci lepsze wyniki niż EACS, co sugerują uzyskane rezultaty dla problemów o parametrach $p = 1,\ r = 100$ i $n \geq 300$. Należy zauważyć, że dla problemu *R.200.100.1* również odnotowano poprawę, a fakt, że jej procentowy wymiar jest mniejszy niż dla problemu *R.300.100.1* może być spowodowany bardzo małymi różnicami w średnich kosztach uzyskanych dróg odpowiednio $0{,}6$ dla *R.200.100.1* oraz $0{,}7$ dla *R.300.100.1*.

W uzyskanych rezultatach dla dziewiętnastu problemów wystąpiło pogorszenie liczby wykonanych iteracji. W piętnastu z nich odnotowano poprawę

otrzymywanych rezultatów. Z pozostałych czterech tylko dla problemu *R.200.100.15.sop* odnotowano pogorszenie większe niż 0,1%. Prawdopodobnie dla tych przypadków algorytm zEACS znalazł bardziej obiecującą przestrzeń rozwiązań, której przeszukiwanie powodowało intensywniejsze wykorzystywanie lokalnej optymalizacji, a tym samym wydłużenie średniego czasu iteracji.

5.5.6. *Dodawanie, wyparowywanie i początkowa wartość feromonu na krawędzi zależna od kosztu*

W tabeli 5.6 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której dodawanie, wyparowywanie oraz początkowa wartość feromonu na krawędziach jest zależna od ich kosztu. Dokładny opis wykorzystanych modyfikacji znajduje się w punktach 4.2.3, 4.2.4 i 4.2.5.

Tabela 5.6 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której dodawanie, wyparowywanie oraz początkowa wartość feromonu na krawędziach jest zależna od ich kosztu

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	76,2	-1,74%	11381,7	19203,5	68,72%
R.200.100.15.sop	1925,5	1929,9	-0,23%	21796,4	15296,7	-29,82%
R.200.100.30.sop	4236,2	4229,5	0,16%	1574,3	1321,5	-16,06%
R.200.100.60.sop	71749	71773,9	-0,03%	5451	4052,7	-25,65%
R.200.1000.1.sop	1471,5	1473,5	-0,14%	4337,2	11145,2	156,97%
R.200.1000.15.sop	22362,9	21833,4	2,37%	9097,7	10275,3	12,94%
R.200.1000.30.sop	41353,3	41330,4	0,06%	2006,8	1659	-17,33%
R.200.1000.60.sop	71570,5	71614	-0,06%	4432	3865,4	-12,78%
R.300.100.1.sop	46,2	48,2	-4,33%	3947,9	12255,3	210,43%
R.300.100.15.sop	3528,1	3469	1,68%	965,3	1219,6	26,34%
R.300.100.30.sop	6204,3	6163,5	0,66%	541,9	379,6	-29,95%
R.300.100.60.sop	9726	9726	0,00%	2271,2	1588,1	-30,08%
R.300.1000.1.sop	1462,9	1471,8	-0,61%	848	6234,1	635,15%
R.300.1000.15.sop	33301,4	33047,5	0,76%	1532,1	1520	-0,79%
R.300.1000.30.sop	55545,1	54683,1	1,55%	331,4	282,8	-14,67%
R.300.1000.60.sop	109783,9	109609,7	0,16%	1586,7	1088,3	-31,41%
R.400.100.1.sop	39,9	49,9	-25,06%	950,9	6510,6	584,68%
R.400.100.15.sop	4761,9	4558	4,28%	266,6	484,5	81,73%
R.400.100.30.sop	8511	8362,8	1,74%	158,5	143,4	-9,53%

R.400.100.60.sop	15273,6	15248,9	0,16%	593,8	539,9	-9,08%
R.400.1000.1.sop	1575,4	1571,6	0,24%	229,3	4191,9	1728,13%
R.400.1000.15.sop	46890,7	44842,4	4,37%	308	436,5	41,72%
R.400.1000.30.sop	87839,1	86426,9	1,61%	173,9	148,3	-14,72%
R.400.1000.60.sop	141531,9	141033,3	0,35%	650	528,3	-18,72%
R.500.100.1.sop	34,9	39,3	-12,61%	452	3667,7	711,44%
R.500.100.15.sop	6701	6433,4	3,99%	68,6	124,9	82,07%
R.500.100.30.sop	10336,8	10152,7	1,78%	82,8	67,8	-18,12%
R.500.100.60.sop	18407,9	18415,6	-0,04%	275,2	274,6	-0,22%
R.500.1000.1.sop	1683,1	1648,7	2,04%	88,1	2378,9	2600,23%
R.500.1000.15.sop	62426,6	58865,2	5,70%	79,2	149,7	89,02%
R.500.1000.30.sop	104370,2	103278	1,05%	90,6	56,5	-37,64%
R.500.1000.60.sop	179575,3	179204	0,21%	364,6	293,4	-19,53%
R.600.100.1.sop	29,2	39,7	-35,96%	304	2612,3	759,31%
R.600.100.15.sop	7650,8	7329,7	4,20%	30,8	60	94,81%
R.600.100.30.sop	13411,5	13141,2	2,02%	63,8	47	-26,33%
R.600.100.60.sop	23996,6	23714,2	1,18%	209,9	184,9	-11,91%
R.600.1000.1.sop	1799,9	1855,1	-3,07%	47,5	2309,8	4762,74%
R.600.1000.15.sop	74328,1	71665	3,58%	49,2	72,5	47,36%
R.600.1000.30.sop	137129,7	135685,9	1,05%	55,2	35,5	-35,69%
R.600.1000.60.sop	224034,2	221828,8	0,98%	164,4	142,9	-13,08%
R.700.100.1.sop	26,6	38,7	-45,49%	167,4	1969,8	1076,70%
R.700.100.15.sop	9604,7	9398,1	2,15%	15,6	23,3	49,36%
R.700.100.30.sop	16282,2	15858,4	2,60%	37,5	31,1	-17,07%
R.700.100.60.sop	25309,8	25021,4	1,14%	148,3	110,4	-25,56%
R.700.1000.1.sop	1750,2	1832,4	-4,70%	31,3	1753	5500,64%
R.700.1000.15.sop	92592	87802,9	5,17%	18,2	28	53,85%
R.700.1000.30.sop	148859,8	147630	0,83%	38,1	20,3	-46,72%
R.700.1000.60.sop	257040,2	253763	1,27%	149,9	116,6	-22,21%
		Średnio:	**-1,52%**		**Średnio:**	**392,49%**

Proponowany algorytm zEACS jest połączeniem trzech modyfikacji, których indywidualne zastosowanie zbadano w punktach 5.5.3, 5.5.4 i 5.5.5. Daje on pogorszenie wyniku średnio o $1{,}52\%$. Otrzymane rezultaty prezentują podobne cechy, jak algorytmy powstałe w wyniku zastosowania pojedynczych modyfikacji. Średnia procentowa zmiana jest mocno zaniżana przez problemy o parametrach $n \geq 400$, $r = 100$ i $p = 1$, jak to miało miejsce przy zastosowaniu jedynie zmienionego sposobu inicjalizacji ilości feromonu. Średnia poprawa otrzymywanych rezultatów dla pozostałych problemów wynosi $1{,}05\%$, co jest wynikiem gorszym, niż w przypadku algorytmu, którego badania przedstawiono w punkcie 5.5.5. Połączenie zmian mających na celu lepsze odwzorowanie sytuacji rzeczywistej nie spowodowało powstania nowych zależności pomiędzy

otrzymanymi rezultatami, a parametrami problemu. Wszystkie zależności można dostrzec podczas badania poszczególnych modyfikacji z osobna.

5.5.7. *Początkowa wartość feromonu na krawędzi zależna od kosztu i przerywanie budowania kosztownych rozwiązań*

W tabeli 5.7 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której początkowa wartość feromonu na krawędziach jest zależna od ich kosztu oraz zastosowano przerywanie budowania kosztownych rozwiązań. Dokładny opis wykorzystanych modyfikacji znajduje się w punktach 4.2.5 i 4.2.1.

Tabela 5.7 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której początkowa wartość feromonu na krawędziach jest zależna od ich kosztu oraz zastosowano przerywanie budowania kosztownych rozwiązań

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	75,4	-0,67%	11381,7	26867,5	136,06%
R.200.100.15.sop	1925,5	1920,4	0,26%	21796,4	17670,4	-18,93%
R.200.100.30.sop	4236,2	4222,5	0,32%	1574,3	1596,5	1,41%
R.200.100.60.sop	71749	71749	0,00%	5451	4438,8	-18,57%
R.200.1000.1.sop	1471,5	1471,3	0,01%	4337,2	16397,3	278,06%
R.200.1000.15.sop	22362,9	22052,7	1,39%	9097,7	8989,4	-1,19%
R.200.1000.30.sop	41353,3	41404,1	-0,12%	2006,8	1968,9	-1,89%
R.200.1000.60.sop	71570,5	71657,5	-0,12%	4432	4202,9	-5,17%
R.300.100.1.sop	46,2	48,6	-5,19%	3947,9	19626,5	397,14%
R.300.100.15.sop	3528,1	3481,3	1,33%	965,3	1475	52,80%
R.300.100.30.sop	6204,3	6174,9	0,47%	541,9	417,5	-22,96%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2059,2	-9,33%
R.300.1000.1.sop	1462,9	1444,4	1,26%	848	10398	1126,18%
R.300.1000.15.sop	33301,4	32888,5	1,24%	1532,1	1799,6	17,46%
R.300.1000.30.sop	55545,1	54651	1,61%	331,4	313,9	-5,28%
R.300.1000.60.sop	109783,9	109705,1	0,07%	1586,7	1556,1	-1,93%
R.400.100.1.sop	39,9	45,7	-14,54%	950,9	12086,7	1171,08%
R.400.100.15.sop	4761,9	4547,5	4,50%	266,6	433,8	62,72%
R.400.100.30.sop	8511	8372,1	1,63%	158,5	148,2	-6,50%
R.400.100.60.sop	15273,6	15311,9	-0,25%	593,8	599,6	0,98%
R.400.1000.1.sop	1575,4	1639,1	-4,04%	229,3	4864,8	2021,59%
R.400.1000.15.sop	46890,7	44879,5	4,29%	308	430,7	39,84%
R.400.1000.30.sop	87839,1	87059,3	0,89%	173,9	151,6	-12,82%

R.400.1000.60.sop	141531,9	141477,7	0,04%	650	575,4	-11,48%
R.500.100.1.sop	34,9	41,8	-19,77%	452	6737,5	1390,60%
R.500.100.15.sop	6701	6325,4	5,61%	68,6	109,9	60,20%
R.500.100.30.sop	10336,8	10128,8	2,01%	82,8	66,7	-19,44%
R.500.100.60.sop	18407,9	18371,8	0,20%	275,2	286	3,92%
R.500.1000.1.sop	1683,1	1695,3	-0,72%	88,1	3723,7	4126,67%
R.500.1000.15.sop	62426,6	60322,3	3,37%	79,2	119	50,25%
R.500.1000.30.sop	104370,2	103128,3	1,19%	90,6	70,2	-22,52%
R.500.1000.60.sop	179575,3	179205	0,21%	364,6	353,9	-2,93%
R.600.100.1.sop	29,2	39,8	-36,30%	304	5179,5	1603,78%
R.600.100.15.sop	7650,8	7364,2	3,75%	30,8	59,2	92,21%
R.600.100.30.sop	13411,5	13154	1,92%	63,8	57,1	-10,50%
R.600.100.60.sop	23996,6	23686,6	1,29%	209,9	199,1	-5,15%
R.600.1000.1.sop	1799,9	1825,3	-1,41%	47,5	2321,2	4786,74%
R.600.1000.15.sop	74328,1	71307,6	4,06%	49,2	73,8	50,00%
R.600.1000.30.sop	137129,7	134730,8	1,75%	55,2	42	-23,91%
R.600.1000.60.sop	224034,2	222054,1	0,88%	164,4	149,5	-9,06%
R.700.100.1.sop	26,6	39,2	-47,37%	167,4	5147,5	2974,97%
R.700.100.15.sop	9604,7	9311,9	3,05%	15,6	23,2	48,72%
R.700.100.30.sop	16282,2	15989,6	1,80%	37,5	30,3	-19,20%
R.700.100.60.sop	25309,8	25002,3	1,21%	148,3	124,7	-15,91%
R.700.1000.1.sop	1750,2	1874,8	-7,12%	31,3	1737,1	5449,84%
R.700.1000.15.sop	92592	87460,8	5,54%	18,2	32,2	76,92%
R.700.1000.30.sop	148859,8	146671,1	1,47%	38,1	27,4	-28,08%
R.700.1000.60.sop	257040,2	254489,6	0,99%	149,9	116,2	-22,48%
		Średnio:	**-1,63%**		**Średnio:**	**535,94%**

Dla algorytmu zEACS otrzymano rezultaty gorsze, niż gdy zastosowano jedynie modyfikację inicjalizacji feromonu[7]. Średnie pogorszenie wyniosło $1{,}63\%$. Po przerwaniu budowania rozwiązania nie jest zmniejszana ilość feromonu na krawędziach, które w przypadku zastosowania EACS zostałyby włączone ścieżki. To z kolei powoduje zwiększenie prawdopodobieństwa ich wyboru w kolejnych krokach algorytmu i jest prawdopodobnie powodem spadku jakości rozwiązań.

Zwiększenie liczby iteracji w stosunku do wyników badań z punktu 5.5.5 sugeruje, że nie należy doszukiwać się powodu pogorszenia wyników w dodatkowym narzucie obliczeniowym wynikającym z konieczności sprawdzania, czy należy przerwać tworzenie ścieżki.

[7] Wyniki badań dla wspomnianej modyfikacji znajdują się w punkcie 5.5.5.

5.5.8. *Wybór deterministyczny jedynym sposobem wyboru krawędzi należącej do najlepszego rozwiązania*

W tabeli 5.8 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której sposób deterministyczny jest jedyną możliwością wyboru krawędzi należącej do najlepszego rozwiązania. Dokładny opis wykorzystanych modyfikacji znajduje się w punkcie 4.2.6.

Tabela 5.8 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której sposób deterministyczny jest jedyną możliwością wyboru krawędzi należącej do najlepszego rozwiązania

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	96,7	-29,11%	11381,7	48531,2	326,40%
R.200.100.15.sop	1925,5	2043,6	-6,13%	21796,4	50664,5	132,44%
R.200.100.30.sop	4236,2	4243,9	-0,18%	1574,3	3921,3	149,08%
R.200.100.60.sop	71749	71749	0,00%	5451	313,6	-94,25%
R.200.1000.1.sop	1471,5	1573,3	-6,92%	4337,2	54386,5	1153,95%
R.200.1000.15.sop	22362,9	22786	-1,89%	9097,7	45287,9	397,80%
R.200.1000.30.sop	41353,3	41263,5	0,22%	2006,8	1853,8	-7,62%
R.200.1000.60.sop	71570,5	71556	0,02%	4432	254,3	-94,26%
R.300.100.1.sop	46,2	75,9	-64,29%	3947,9	26004	558,68%
R.300.100.15.sop	3528,1	3672,5	-4,09%	965,3	10504,5	988,21%
R.300.100.30.sop	6204,3	6237,1	-0,53%	541,9	124,7	-76,99%
R.300.100.60.sop	9726	9726	0,00%	2271,2	80,9	-96,44%
R.300.1000.1.sop	1462,9	1630,1	-11,43%	848	23258	2642,69%
R.300.1000.15.sop	33301,4	34444,2	-3,43%	1532,1	12391	708,76%
R.300.1000.30.sop	55545,1	55487,7	0,10%	331,4	712,2	114,91%
R.300.1000.60.sop	109783,9	109590	0,18%	1586,7	74,8	-95,29%
R.400.100.1.sop	39,9	79,2	-98,50%	950,9	14298,5	1403,68%
R.400.100.15.sop	4761,9	4941,6	-3,77%	266,6	4112,7	1442,65%
R.400.100.30.sop	8511	8609,8	-1,16%	158,5	24,6	-84,48%
R.400.100.60.sop	15273,6	15240,7	0,22%	593,8	34,4	-94,21%
R.400.1000.1.sop	1575,4	1784,5	-13,27%	229,3	11718,6	5010,60%
R.400.1000.15.sop	46890,7	48373,8	-3,16%	308	3224,4	946,88%
R.400.1000.30.sop	87839,1	89430,9	-1,81%	173,9	19,6	-88,73%
R.400.1000.60.sop	141531,9	140948,2	0,41%	650	31,3	-95,18%
R.500.100.1.sop	34,9	64	-83,38%	452	9074,2	1907,57%
R.500.100.15.sop	6701	6843,7	-2,13%	68,6	455	563,27%
R.500.100.30.sop	10336,8	10514,3	-1,72%	82,8	7,5	-90,94%
R.500.100.60.sop	18407,9	18332,8	0,41%	275,2	14,2	-94,84%

R.500.1000.1.sop	1683,1	1832,5	-8,88%	88,1	6877,3	7706,24%
R.500.1000.15.sop	62426,6	64098,2	-2,68%	79,2	538,7	580,18%
R.500.1000.30.sop	104370,2	106171,8	-1,73%	90,6	8,1	-91,06%
R.500.1000.60.sop	179575,3	178824,5	0,42%	364,6	14,6	-96,00%
R.600.100.1.sop	29,2	59,5	-103,77%	304	5838,1	1820,43%
R.600.100.15.sop	7650,8	7794,6	-1,88%	30,8	219,1	611,36%
R.600.100.30.sop	13411,5	13649,4	-1,77%	63,8	4,1	-93,57%
R.600.100.60.sop	23996,6	23482,4	2,14%	209,9	10,7	-94,90%
R.600.1000.1.sop	1799,9	1930,5	-7,26%	47,5	3927,2	8167,79%
R.600.1000.15.sop	74328,1	75455	-1,52%	49,2	491,1	898,17%
R.600.1000.30.sop	137129,7	138125,9	-0,73%	55,2	3,5	-93,66%
R.600.1000.60.sop	224034,2	217517	2,91%	164,4	8,1	-95,07%
R.700.100.1.sop	26,6	50,5	-89,85%	167,4	4234,3	2429,45%
R.700.100.15.sop	9604,7	9767,5	-1,70%	15,6	19	21,79%
R.700.100.30.sop	16282,2	16174,7	0,66%	37,5	2,8	-92,53%
R.700.100.60.sop	25309,8	24538,7	3,05%	148,3	6,2	-95,82%
R.700.1000.1.sop	1750,2	1902,5	-8,70%	31,3	2935,4	9278,27%
R.700.1000.15.sop	92592	93108,2	-0,56%	18,2	53,7	195,05%
R.700.1000.30.sop	148859,8	151359,4	-1,68%	38,1	2,8	-92,65%
R.700.1000.60.sop	257040,2	250624,5	2,50%	149,9	6,4	-95,73%
		Średnio:	**-11,59%**		**Średnio:**	**1004,21%**

Proponowana modyfikacja pogarsza średnią wartość uzyskiwanych rezultatów o $11{,}59\%$. Znaczący wzrost liczby wykonywanych iteracji w części problemów prawdopodobnie jest rezultatem wyznaczania wielu rozwiązań niekwalifikujących się do podjęcia procesu lokalnej optymalizacji. Należy zauważyć, że zEACS powoduje poprawę uzyskiwanych rezultatów dla problemów o parametrze $p = 60$ (wyjątkiem jest problem *R.200.100.60*, dla którego uzyskany wynik jest identyczny dla obydwu algorytmów). Dodatkowo można zauważyć zależność, że im większe n, tym większa jest to poprawa. Co ciekawe, liczba wykonywanych iteracji dla tych instancji drastycznie spadła, średnio o $\mathbf{95{,}26}\%$. Sugeruje to, że większość czasu zostało poświęcone na wykonanie lokalnej optymalizacji.

Podjęto próbę wyznaczenia dobrej wartości parametru s (oczekiwana liczba węzłów wybranych w sposób probabilistyczny) dla algorytmu zEACS za pomocą mechanizmu zaimplementowanego w programie badawczym. Badania te zostały wykonane na problemie *R.200.100.15* dla czasu pracy algorytmu ustawionego na 30 sekund. Najlepsze wyniki uzyskano dla $s = 3{,}40099596133855$. Wartość ta

została wykorzystana do przeprowadzenia standardowej serii badań, których rezultaty znajdują się w tabeli 5.9.

Tabela 5.9 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której sposób deterministyczny jest jedyną możliwością wyboru krawędzi należącej do najlepszego rozwiązania po zmianie parametru s

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	73,2	2,27%	11381,7	23985,5	110,74%
R.200.100.15.sop	1925,5	1970,1	-2,32%	21796,4	24862,4	14,07%
R.200.100.30.sop	4236,2	4241,4	-0,12%	1574,3	660,8	-58,03%
R.200.100.60.sop	71749	71749	0,00%	5451	801,5	-85,30%
R.200.1000.1.sop	1471,5	1456,5	1,02%	4337,2	11325,4	161,12%
R.200.1000.15.sop	22362,9	22208,5	0,69%	9097,7	11189,9	23,00%
R.200.1000.30.sop	41353,3	41281,2	0,17%	2006,8	573,1	-71,44%
R.200.1000.60.sop	71570,5	71614	-0,06%	4432	676,6	-84,73%
R.300.100.1.sop	46,2	44,1	4,55%	3947,9	10405,1	163,56%
R.300.100.15.sop	3528,1	3601,5	-2,08%	965,3	1037,3	7,46%
R.300.100.30.sop	6204,3	6222	-0,29%	541,9	101,5	-81,27%
R.300.100.60.sop	9726	9726	0,00%	2271,2	226,8	-90,01%
R.300.1000.1.sop	1462,9	1425,7	2,54%	848	3074,2	262,52%
R.300.1000.15.sop	33301,4	33972,5	-2,02%	1532,1	1636,7	6,83%
R.300.1000.30.sop	55545,1	55473,6	0,13%	331,4	138,1	-58,33%
R.300.1000.60.sop	109783,9	109704,7	0,07%	1586,7	211,1	-86,70%
R.400.100.1.sop	39,9	37,9	5,01%	950,9	4420,5	364,88%
R.400.100.15.sop	4761,9	4795,1	-0,70%	266,6	263,9	-1,01%
R.400.100.30.sop	8511	8459,7	0,60%	158,5	35,2	-77,79%
R.400.100.60.sop	15273,6	15233,2	0,26%	593,8	96,1	-83,82%
R.400.1000.1.sop	1575,4	1599,8	-1,55%	229,3	760,9	231,84%
R.400.1000.15.sop	46890,7	46502,7	0,83%	308	453,1	47,11%
R.400.1000.30.sop	87839,1	87865,6	-0,03%	173,9	39,1	-77,52%
R.400.1000.60.sop	141531,9	140896,8	0,45%	650	105,3	-83,80%
R.500.100.1.sop	34,9	28,6	18,05%	452	2197,3	386,13%
R.500.100.15.sop	6701	6607	1,40%	68,6	75,3	9,77%
R.500.100.30.sop	10336,8	10254,6	0,80%	82,8	15,9	-80,80%
R.500.100.60.sop	18407,9	18279,5	0,70%	275,2	47,3	-82,81%
R.500.1000.1.sop	1683,1	1654,8	1,68%	88,1	309,9	251,76%
R.500.1000.15.sop	62426,6	62360,7	0,11%	79,2	69,1	-12,75%
R.500.1000.30.sop	104370,2	104283,3	0,08%	90,6	17,4	-80,79%
R.500.1000.60.sop	179575,3	178399,5	0,65%	364,6	51,6	-85,85%
R.600.100.1.sop	29,2	23,1	20,89%	304	1491,7	390,69%
R.600.100.15.sop	7650,8	7584,4	0,87%	30,8	27,3	-11,36%

R.600.100.30.sop	13411,5	13321,5	0,67%	63,8	11,7	-81,66%
R.600.100.60.sop	23996,6	23444,7	2,30%	209,9	35,5	-83,09%
R.600.1000.1.sop	1799,9	1790,3	0,53%	47,5	145,4	206,11%
R.600.1000.15.sop	74328,1	74610,3	-0,38%	49,2	39,4	-19,92%
R.600.1000.30.sop	137129,7	135077,3	1,50%	55,2	10,9	-80,25%
R.600.1000.60.sop	224034,2	216739,7	3,26%	164,4	25,8	-84,31%
R.700.100.1.sop	26,6	23,9	10,15%	167,4	879,1	425,15%
R.700.100.15.sop	9604,7	9656,1	-0,54%	15,6	7,7	-50,64%
R.700.100.30.sop	16282,2	15872,4	2,52%	37,5	6,8	-81,87%
R.700.100.60.sop	25309,8	24401,5	3,59%	148,3	18,7	-87,39%
R.700.1000.1.sop	1750,2	1773,9	-1,35%	31,3	103,1	229,39%
R.700.1000.15.sop	92592	92305,3	0,31%	18,2	11,6	-36,26%
R.700.1000.30.sop	148859,8	148264,4	0,40%	38,1	6,4	-83,20%
R.700.1000.60.sop	257040,2	249974,7	2,75%	149,9	17,2	-88,53%
		Średnio:	**1,67%**		**Średnio:**	**25,44%**

Po zmianie parametru s uzyskano średnią poprawę otrzymywanych rezultatów na poziomie $1{,}67\%$. Największa poprawa występuje dla problemów o parametrze $p = 1$, średnio $5{,}32\%$, natomiast największe pogorszenie dla $p = 15$, średnio o $0{,}32\%$. Należy zwrócić uwagę, że również tym razem średnia liczba iteracji silnie zależy od liczby ograniczeń.

5.5.9. Zmniejszanie znaczenia heurystyki z kolejnymi iteracjami

W niniejszym punkcie opisano badania przeprowadzone na zmienionej wersji algorytmu EACS, w której zastosowano zmniejszanie znaczenia heurystyki wraz z kolejnymi iteracjami. Algorytm ten w dalszej części będzie oznaczany jako zEACS. Dokładny opis wykorzystanych modyfikacji znajduje się w punkcie 4.2.7.

W początkowej fazie badania zEACS skupiono się na wyznaczeniu dobrych wartości dla parametrów γ oraz ω_0. W tym celu wykonano serię pięciu badań za pomocą mechanizmu zaimplementowanego w programie badawczym. Badania te zostały wykonane na problemie R.200.100.15 dla czasu pracy algorytmu ustawionego na 30 sekund. Uzyskane wyniki pozwoliły wyłonić najbardziej obiecujące wartości parametrów: $\gamma = 0{,}108329395992977$ i $\omega_0 = 1{,}82509005287434$. Następnie zostały one wykorzystane do przeprowadzenia standardowej serii badań, których rezultaty znajdują się w tabeli 5.10.

Tabela 5.10 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zastosowano zmniejszanie znaczenia heurystyki wraz z kolejnymi iteracjami

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	87,1	-16,29%	11381,7	48871,8	329,39%
R.200.100.15.sop	1925,5	1939,1	-0,71%	21796,4	22670,2	4,01%
R.200.100.30.sop	4236,2	4220,3	0,38%	1574,3	1561,7	-0,80%
R.200.100.60.sop	71749	71773,9	-0,03%	5451	4560,9	-16,33%
R.200.1000.1.sop	1471,5	1673,4	-13,72%	4337,2	52898,9	1119,66%
R.200.1000.15.sop	22362,9	21829,9	2,38%	9097,7	20286	122,98%
R.200.1000.30.sop	41353,3	41235,4	0,29%	2006,8	2111,6	5,22%
R.200.1000.60.sop	71570,5	71614	-0,06%	4432	4637,9	4,65%
R.300.100.1.sop	46,2	66,1	-43,07%	3947,9	25845,3	554,66%
R.300.100.15.sop	3528,1	3430,3	2,77%	965,3	2910	201,46%
R.300.100.30.sop	6204,3	6171,7	0,53%	541,9	378,7	-30,12%
R.300.100.60.sop	9726	9726	0,00%	2271,2	1945,8	-14,33%
R.300.1000.1.sop	1462,9	1683,4	-15,07%	848	24850,8	2830,52%
R.300.1000.15.sop	33301,4	32046,7	3,77%	1532,1	3227,9	110,68%
R.300.1000.30.sop	55545,1	55012,4	0,96%	331,4	264,2	-20,28%
R.300.1000.60.sop	109783,9	109542,4	0,22%	1586,7	1459,5	-8,02%
R.400.100.1.sop	39,9	52,5	-31,58%	950,9	13248	1293,21%
R.400.100.15.sop	4761,9	4613,3	3,12%	266,6	282,7	6,04%
R.400.100.30.sop	8511	8414,8	1,13%	158,5	144,4	-8,90%
R.400.100.60.sop	15273,6	15249,1	0,16%	593,8	565	-4,85%
R.400.1000.1.sop	1575,4	1804,6	-14,55%	229,3	12903,1	5527,17%
R.400.1000.15.sop	46890,7	46867,2	0,05%	308	217,2	-29,48%
R.400.1000.30.sop	87839,1	87201,8	0,73%	173,9	135,8	-21,91%
R.400.1000.60.sop	141531,9	141131,5	0,28%	650	578	-11,08%
R.500.100.1.sop	34,9	42,2	-20,92%	452	7213,9	1496,00%
R.500.100.15.sop	6701	6615,9	1,27%	68,6	80,4	17,20%
R.500.100.30.sop	10336,8	10175,4	1,56%	82,8	64,8	-21,74%
R.500.100.60.sop	18407,9	18451	-0,23%	275,2	213,3	-22,49%
R.500.1000.1.sop	1683,1	1796,2	-6,72%	88,1	7718,8	8661,41%
R.500.1000.15.sop	62426,6	61070,4	2,17%	79,2	79,5	0,38%
R.500.1000.30.sop	104370,2	105035	-0,64%	90,6	65,5	-27,70%
R.500.1000.60.sop	179575,3	178918,7	0,37%	364,6	272,3	-25,32%
R.600.100.1.sop	29,2	34	-16,44%	304	3376,9	1010,82%
R.600.100.15.sop	7650,8	7597,9	0,69%	30,8	31,5	2,27%
R.600.100.30.sop	13411,5	13393,7	0,13%	63,8	48,6	-23,82%
R.600.100.60.sop	23996,6	23835,1	0,67%	209,9	179,8	-14,34%
R.600.1000.1.sop	1799,9	1916,8	-6,49%	47,5	3508,4	7286,11%
R.600.1000.15.sop	74328,1	74972,4	-0,87%	49,2	33,7	-31,50%

R.600.1000.30.sop	137129,7	137083,1	0,03%	55,2	40,5	-26,63%
R.600.1000.60.sop	224034,2	222579,5	0,65%	164,4	128,8	-21,65%
R.700.100.1.sop	26,6	24,6	7,52%	167,4	1949,3	1064,46%
R.700.100.15.sop	9604,7	9589,7	0,16%	15,6	13,8	-11,54%
R.700.100.30.sop	16282,2	16180,9	0,62%	37,5	32,6	-13,07%
R.700.100.60.sop	25309,8	25024,3	1,13%	148,3	124,3	-16,18%
R.700.1000.1.sop	1750,2	1816,6	-3,79%	31,3	1761,2	5526,84%
R.700.1000.15.sop	92592	91734,4	0,93%	18,2	18,2	0,00%
R.700.1000.30.sop	148859,8	149059,1	-0,13%	38,1	32,6	-14,44%
R.700.1000.60.sop	257040,2	255264,8	0,69%	149,9	108,9	-27,35%
		Średnio:	**-3,25%**		**Średnio:**	**764,82%**

Proponowana modyfikacja powoduje pogorszenie otrzymywanych rezultatów średnio o $3{,}25\%$. Problemy, dla których spadek jakości wyników był największy cechuje parametr $p = 1$. W ich przypadku średnie pogorszenie wynosi $15{,}09\%$ przy równoczesnym zwiększeniu liczby iteracji o $3058{,}35\%$. Tak znacząca zmiana w liczbie iteracji najprawdopodobniej jest rezultatem wielokrotnego wyznaczania rozwiązań, dla których algorytm nie podejmował próby lokalnej optymalizacji. Dla pozostałych instancji odnotowano poprawę na poziomie $0{,}7\%$ przy równoczesnym wzroście średniej liczby iteracji o $0{,}31\%$.

5.5.10. Zmniejszanie znaczenia heurystyki z kolejnymi iteracjami i przerywanie budowania kosztownych rozwiązań

W niniejszym punkcie opisano badania przeprowadzone na zmienionej wersji algorytmu EACS, w której zastosowano zmniejszanie znaczenia heurystyki wraz z kolejnymi iteracjami oraz przerywanie budowania kosztownych rozwiązań. Dokładny opis wykorzystanych modyfikacji znajduje się w punktach 4.2.7 i 4.2.1.

W początkowej fazie badania zEACS skupiono się na wyznaczeniu dobrych wartości dla parametrów γ oraz ω_0. W tym celu wykonano serię pięciu badań za pomocą mechanizmu zaimplementowanego w programie badawczym. Zostały one wykonane na problemie R.200.100.15 dla czasu pracy algorytmu ustawionego na 30 sekund. Uzyskane wyniki pozwoliły wyłonić najbardziej obiecujące wartości parametrów: $\gamma = 0{,}0486351410159278$ i $\omega_0 = 0{,}248158903748184$. Następnie zostały one wykorzystane do przeprowadzenia standardowej serii badań, których rezultaty znajdują się w tabeli 5.11.

Tabela 5.11 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zastosowano zmniejszanie znaczenia heurystyki wraz z kolejnymi iteracjami oraz przerywanie budowania kosztownych rozwiązań

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	79,9	-6,68%	11381,7	88420,5	676,87%
R.200.100.15.sop	1925,5	1925,7	-0,01%	21796,4	31681,9	45,35%
R.200.100.30.sop	4236,2	4226,6	0,23%	1574,3	1941,6	23,33%
R.200.100.60.sop	71749	71773,9	-0,03%	5451	5053,3	-7,30%
R.200.1000.1.sop	1471,5	1524,6	-3,61%	4337,2	87156,2	1909,50%
R.200.1000.15.sop	22362,9	21735,9	2,80%	9097,7	24670,1	171,17%
R.200.1000.30.sop	41353,3	41311,7	0,10%	2006,8	2381,4	18,67%
R.200.1000.60.sop	71570,5	71614	-0,06%	4432	5258,1	18,64%
R.300.100.1.sop	46,2	53,3	-15,37%	3947,9	30061,6	661,46%
R.300.100.15.sop	3528,1	3422,4	3,00%	965,3	1860,4	92,73%
R.300.100.30.sop	6204,3	6167,5	0,59%	541,9	395,5	-27,02%
R.300.100.60.sop	9726	9726	0,00%	2271,2	1933,1	-14,89%
R.300.1000.1.sop	1462,9	1529,1	-4,53%	848	36056,5	4151,95%
R.300.1000.15.sop	33301,4	31856,8	4,34%	1532,1	3171,5	107,00%
R.300.1000.30.sop	55545,1	54965,2	1,04%	331,4	280	-15,51%
R.300.1000.60.sop	109783,9	109566,2	0,20%	1586,7	1536,5	-3,16%
R.400.100.1.sop	39,9	42,8	-7,27%	950,9	8650,8	809,75%
R.400.100.15.sop	4761,9	4661	2,12%	266,6	261,6	-1,88%
R.400.100.30.sop	8511	8382,2	1,51%	158,5	136,3	-14,01%
R.400.100.60.sop	15273,6	15232,5	0,27%	593,8	612,5	3,15%
R.400.1000.1.sop	1575,4	1617,5	-2,67%	229,3	14100,8	6049,50%
R.400.1000.15.sop	46890,7	46225,8	1,42%	308	273,7	-11,14%
R.400.1000.30.sop	87839,1	87073,2	0,87%	173,9	147,7	-15,07%
R.400.1000.60.sop	141531,9	140866,5	0,47%	650	658,8	1,35%
R.500.100.1.sop	34,9	39,1	-12,03%	452	2689,5	495,02%
R.500.100.15.sop	6701	6637,7	0,94%	68,6	87,6	27,70%
R.500.100.30.sop	10336,8	10148,6	1,82%	82,8	69,7	-15,82%
R.500.100.60.sop	18407,9	18388	0,11%	275,2	268,8	-2,33%
R.500.1000.1.sop	1683,1	1680,6	0,15%	88,1	4762,9	5306,24%
R.500.1000.15.sop	62426,6	62291	0,22%	79,2	85,8	8,33%
R.500.1000.30.sop	104370,2	104161,7	0,20%	90,6	73,2	-19,21%
R.500.1000.60.sop	179575,3	178976,9	0,33%	364,6	306,2	-16,02%
R.600.100.1.sop	29,2	29,3	-0,34%	304	602,6	98,22%
R.600.100.15.sop	7650,8	7570,7	1,05%	30,8	27	-12,34%
R.600.100.30.sop	13411,5	13386,8	0,18%	63,8	48,5	-23,98%
R.600.100.60.sop	23996,6	23710,4	1,19%	209,9	182	-13,29%
R.600.1000.1.sop	1799,9	1775,5	1,36%	47,5	659	1287,37%
R.600.1000.15.sop	74328,1	74055,3	0,37%	49,2	42,9	-12,80%

R.600.1000.30.sop	137129,7	136495	0,46%	55,2	44,9	-18,66%
R.600.1000.60.sop	224034,2	219876,4	1,86%	164,4	153,6	-6,57%
R.700.100.1.sop	26,6	27,4	-3,01%	167,4	490	192,71%
R.700.100.15.sop	9604,7	9583,3	0,22%	15,6	15,7	0,64%
R.700.100.30.sop	16282,2	15828,3	2,79%	37,5	38,9	3,73%
R.700.100.60.sop	25309,8	24963,1	1,37%	148,3	141,3	-4,72%
R.700.1000.1.sop	1750,2	1801	-2,90%	31,3	327,6	946,65%
R.700.1000.15.sop	92592	90004,2	2,79%	18,2	19,3	6,04%
R.700.1000.30.sop	148859,8	148281,8	0,39%	38,1	30,8	-19,16%
R.700.1000.60.sop	257040,2	254567,6	0,96%	149,9	122,6	-18,21%
		Średnio:	**-0,43%**		**Średnio:**	**475,42%**

Otrzymane wyniki przeprowadzonych badań posiadają podobne cechy, jak wyniki otrzymane podczas zastosowania jedynie zmniejszania znaczenia heurystyki, które zostały opisane w punkcie 5.5.9. Jednak rezultaty otrzymane po zastosowaniu innych parametrów γ i ω_0 w połączeniu z przerywaniem budowania kosztownych rozwiązań są znacznie lepsze. Proponowana modyfikacja powoduje pogorszenie otrzymywanych rezultatów średnio o $0{,}43\%$. Problemy, dla których spadek jakości wyników był największy cechuje parametr $p = 1$. W ich przypadku średnie pogorszenie wynosi $4{,}74\%$ przy równoczesny zwiększeniu liczby iteracji o $1882{,}1\%$. Tak znacząca zmiana w liczbie iteracji najprawdopodobniej jest rezultatem wielokrotnego przerywania budowanych rozwiązań. Dla pozostałych instancji odnotowano poprawę na poziomie 1% przy równoczesnym wzroście średniej liczby iteracji o $6{,}52\%$.

5.5.11. Zwiększanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny

Modyfikacja polegająca na zwiększaniu oczekiwanej liczby węzłów wybranych w sposób probabilistyczny[8] posiada trzy warianty różniące się sposobem, w jaki zmieniana jest wartość parametru s:

- zwiększanie przez dodawanie (EACSa);
- zwiększanie przez mnożenie (EACSm);
- zwiększanie przez dodawanie i mnożenie (EACSam).

[8] Dokładny opis modyfikacji znajduje się w punkcie 4.2.8.

W początkowej fazie badania EACSa skupiono się na wyznaczeniu dobrych wartości dla parametrów σ oraz s_b. W tym celu wykonano serię pięciu badań za pomocą mechanizmu zaimplementowanego w programie badawczym. Badania te zostały wykonane na problemie *R.200.100.15* dla czasu pracy algorytmu ustawionego na 30 sekund. Uzyskane wyniki pozwoliły wyłonić najbardziej obiecujące wartości parametrów: $\sigma = 0{,}090699866223344$ i $s_b = 0{,}161666909182415$. Następnie zostały one wykorzystane do przeprowadzenia standardowej serii badań, których rezultaty znajdują się w tabeli 5.12.

Tabela 5.12 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (EACSa), w której zastosowano zwiększanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny poprzez dodawanie

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	EACSa	Poprawa	EACS	EACSa	Poprawa
R.200.100.1.sop	74,9	74	1,20%	11381,7	12173,8	6,96%
R.200.100.15.sop	1925,5	1928,9	-0,18%	21796,4	16278,8	-25,31%
R.200.100.30.sop	4236,2	4232,1	0,10%	1574,3	1031,2	-34,50%
R.200.100.60.sop	71749	71749	0,00%	5451	2019,7	-62,95%
R.200.1000.1.sop	1471,5	1458,2	0,90%	4337,2	5481,8	26,39%
R.200.1000.15.sop	22362,9	21798,4	2,52%	9097,7	10694,1	17,55%
R.200.1000.30.sop	41353,3	41315,8	0,09%	2006,8	1264,1	-37,01%
R.200.1000.60.sop	71570,5	71570,5	0,00%	4432	2097,6	-52,67%
R.300.100.1.sop	46,2	44,6	3,46%	3947,9	4838,2	22,55%
R.300.100.15.sop	3528,1	3474	1,53%	965,3	1220,7	26,46%
R.300.100.30.sop	6204,3	6179,6	0,40%	541,9	539,4	-0,46%
R.300.100.60.sop	9726	9726	0,00%	2271,2	870,2	-61,69%
R.300.1000.1.sop	1462,9	1464,4	-0,10%	848	1034,2	21,96%
R.300.1000.15.sop	33301,4	33488,1	-0,56%	1532,1	1324	-13,58%
R.300.1000.30.sop	55545,1	55407	0,25%	331,4	276,6	-16,54%
R.300.1000.60.sop	109783,9	109700,8	0,08%	1586,7	687,7	-56,66%
R.400.100.1.sop	39,9	41,9	-5,01%	950,9	1037,8	9,14%
R.400.100.15.sop	4761,9	4778	-0,34%	266,6	205,4	-22,96%
R.400.100.30.sop	8511	8523	-0,14%	158,5	150	-5,36%
R.400.100.60.sop	15273,6	15293	-0,13%	593,8	458,7	-22,75%
R.400.1000.1.sop	1575,4	1615,6	-2,55%	229,3	200,5	-12,56%
R.400.1000.15.sop	46890,7	46973,8	-0,18%	308	270	-12,34%
R.400.1000.30.sop	87839,1	87580,4	0,29%	173,9	161,9	-6,90%
R.400.1000.60.sop	141531,9	141669,7	-0,10%	650	484,8	-25,42%
R.500.100.1.sop	34,9	33,5	4,01%	452	467,7	3,47%
R.500.100.15.sop	6701	6706,2	-0,08%	68,6	83	20,99%

R.500.100.30.sop	10336,8	10317,6	0,19%	82,8	71,4	-13,77%
R.500.100.60.sop	18407,9	18495,8	-0,48%	275,2	259,5	-5,70%
R.500.1000.1.sop	1683,1	1669	0,84%	88,1	87,5	-0,68%
R.500.1000.15.sop	62426,6	62792,2	-0,59%	79,2	84,4	6,57%
R.500.1000.30.sop	104370,2	104209,3	0,15%	90,6	81,3	-10,26%
R.500.1000.60.sop	179575,3	180111,5	-0,30%	364,6	300,3	-17,64%
R.600.100.1.sop	29,2	30,8	-5,48%	304	270,7	-10,95%
R.600.100.15.sop	7650,8	7610,3	0,53%	30,8	36	16,88%
R.600.100.30.sop	13411,5	13488,2	-0,57%	63,8	63,7	-0,16%
R.600.100.60.sop	23996,6	24034,2	-0,16%	209,9	195,5	-6,86%
R.600.1000.1.sop	1799,9	1786,9	0,72%	47,5	42,3	-10,95%
R.600.1000.15.sop	74328,1	74237,4	0,12%	49,2	45,4	-7,72%
R.600.1000.30.sop	137129,7	136252,6	0,64%	55,2	45,2	-18,12%
R.600.1000.60.sop	224034,2	224436,4	-0,18%	164,4	165,8	0,85%
R.700.100.1.sop	26,6	26,4	0,75%	167,4	174,5	4,24%
R.700.100.15.sop	9604,7	9521,4	0,87%	15,6	14,1	-9,62%
R.700.100.30.sop	16282,2	16112,1	1,04%	37,5	36,7	-2,13%
R.700.100.60.sop	25309,8	25283,8	0,10%	148,3	139,1	-6,20%
R.700.1000.1.sop	1750,2	1760,4	-0,58%	31,3	25,7	-17,89%
R.700.1000.15.sop	92592	92251,9	0,37%	18,2	15,9	-12,64%
R.700.1000.30.sop	148859,8	148837,3	0,02%	38,1	32,7	-14,17%
R.700.1000.60.sop	257040,2	255884,5	0,45%	149,9	156,6	4,47%
		Średnio:	**0,08%**		**Średnio:**	**-9,30%**

Algorytm EACSa nie powoduje większych zmian jakości otrzymywanych rozwiązań, zmniejsza natomiast liczbę wykonanych iteracji średnio o $9{,}3\%$. Nie odnotowano powiązania pomiędzy kierunkiem zmian średnich wyników, a parametrami problemów.

W celu zbadania EACSm również wykonano podobną serię pięciu badań w celu wyznaczenia dobrych wartości dla parametrów s_b oraz μ. Badano również, czy resetowanie wartości s wpływa pozytywnie na rezultaty. Najlepsze wyniki uzyskano dla $s_b = 0{,}271577729269073$, $\mu = 1{,}18337445543562$ i włączonego resetowania wartości s. Następnie wykorzystano te wartości do przeprowadzenia standardowej serii badań, której wyniki znajdują się w tabeli 5.13.

Tabela 5.13 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (EACSm), w której zastosowano zwiększanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny poprzez mnożenie

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	EACSm	Poprawa	EACS	EACSm	Poprawa
R.200.100.1.sop	74,9	72,1	3,74%	11381,7	13073,1	14,86%
R.200.100.15.sop	1925,5	1879,7	2,38%	21796,4	17155,3	-21,29%
R.200.100.30.sop	4236,2	4223,8	0,29%	1574,3	764	-51,47%
R.200.100.60.sop	71749	71749	0,00%	5451	1124,9	-79,36%
R.200.1000.1.sop	1471,5	1468,9	0,18%	4337,2	9645,7	122,39%
R.200.1000.15.sop	22362,9	21441	4,12%	9097,7	10136,1	11,41%
R.200.1000.30.sop	41353,3	41274,1	0,19%	2006,8	664	-66,91%
R.200.1000.60.sop	71570,5	71556	0,02%	4432	945,5	-78,67%
R.300.100.1.sop	46,2	45,1	2,38%	3947,9	4011,3	1,61%
R.300.100.15.sop	3528,1	3430,8	2,76%	965,3	1289,3	33,56%
R.300.100.30.sop	6204,3	6231	-0,43%	541,9	173,3	-68,02%
R.300.100.60.sop	9726	9726	0,00%	2271,2	376,2	-83,44%
R.300.1000.1.sop	1462,9	1454	0,61%	848	3240,4	282,12%
R.300.1000.15.sop	33301,4	32708,1	1,78%	1532,1	1682,2	9,80%
R.300.1000.30.sop	55545,1	55004	0,97%	331,4	165,8	-49,97%
R.300.1000.60.sop	109783,9	109645,8	0,13%	1586,7	283	-82,16%
R.400.100.1.sop	39,9	40,7	-2,01%	950,9	1709,1	79,73%
R.400.100.15.sop	4761,9	4696	1,38%	266,6	285,3	7,01%
R.400.100.30.sop	8511	8537,3	-0,31%	158,5	79,1	-50,09%
R.400.100.60.sop	15273,6	15286,5	-0,08%	593,8	179	-69,86%
R.400.1000.1.sop	1575,4	1609,3	-2,15%	229,3	1045,6	356,00%
R.400.1000.15.sop	46890,7	46556,6	0,71%	308	289,6	-5,97%
R.400.1000.30.sop	87839,1	87680,5	0,18%	173,9	93,3	-46,35%
R.400.1000.60.sop	141531,9	141310,4	0,16%	650	181,1	-72,14%
R.500.100.1.sop	34,9	34,1	2,29%	452	790,8	74,96%
R.500.100.15.sop	6701	6684,7	0,24%	68,6	76,5	11,52%
R.500.100.30.sop	10336,8	10349,9	-0,13%	82,8	49,6	-40,10%
R.500.100.60.sop	18407,9	18415,2	-0,04%	275,2	136,4	-50,44%
R.500.1000.1.sop	1683,1	1670,8	0,73%	88,1	443,2	403,06%
R.500.1000.15.sop	62426,6	62735	-0,49%	79,2	74,4	-6,06%
R.500.1000.30.sop	104370,2	104777,5	-0,39%	90,6	60,3	-33,44%
R.500.1000.60.sop	179575,3	179960,6	-0,21%	364,6	144,6	-60,34%
R.600.100.1.sop	29,2	27,2	6,85%	304	447,1	47,07%
R.600.100.15.sop	7650,8	7603,4	0,62%	30,8	28,1	-8,77%
R.600.100.30.sop	13411,5	13442,7	-0,23%	63,8	48,2	-24,45%

R.600.100.60.sop	23996,6	23865,3	0,55%	209,9	125	-40,45%
R.600.1000.1.sop	1799,9	1789,7	0,57%	47,5	263,2	454,11%
R.600.1000.15.sop	74328,1	75398,6	-1,44%	49,2	34	-30,89%
R.600.1000.30.sop	137129,7	136779,8	0,26%	55,2	41,7	-24,46%
R.600.1000.60.sop	224034,2	222863,2	0,52%	164,4	106,5	-35,22%
R.700.100.1.sop	26,6	26,2	1,50%	167,4	275,2	64,40%
R.700.100.15.sop	9604,7	9576,3	0,30%	15,6	12	-23,08%
R.700.100.30.sop	16282,2	16109,4	1,06%	37,5	32,6	-13,07%
R.700.100.60.sop	25309,8	25257,8	0,21%	148,3	94,7	-36,14%
R.700.1000.1.sop	1750,2	1755,3	-0,29%	31,3	149,3	377,00%
R.700.1000.15.sop	92592	90330,4	2,44%	18,2	17,3	-4,95%
R.700.1000.30.sop	148859,8	148418,2	0,30%	38,1	37	-2,89%
R.700.1000.60.sop	257040,2	255266,1	0,69%	149,9	109	-27,28%
		Średnio:	**0,69%**		**Średnio:**	**22,14%**

Dla EACSm otrzymano poprawę wyników średnio o $0{,}69\%$. Dla problemów o parametrach $r = 1000$ i $p = 1$ diametralnie wzrosła liczba iteracji, co z kolei sugeruje częste odrzucanie rozwiązań przed podjęciem próby lokalnej optymalizacji.

W celu zbadania EACSam również wykonano podobną serię pięciu badań w celu wyznaczenia dobrych wartości dla parametrów σ, s_b oraz μ. Badano również, czy resetowanie wartości s wpływa pozytywnie na rezultaty. Najlepsze wyniki uzyskano dla $\sigma = 0{,}0705622814714942$, $s_b = 0{,}181542536012172$, $\mu = 1{,}18916278730027$ i wyłączonego resetowania wartości s. Następnie wykorzystano te wartości do przeprowadzenia standardowej serii badań, której wyniki znajdują się w tabeli 5.14.

Tabela 5.14 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (EACSam), w której zastosowano zwiększanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny poprzez dodawanie i mnożenie

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	EACSam	Poprawa	EACS	EACSam	Poprawa
R.200.100.1.sop	74,9	72,6	3,07%	11381,7	9287,9	-18,40%
R.200.100.15.sop	1925,5	1912,4	0,68%	21796,4	15353,7	-29,56%
R.200.100.30.sop	4236,2	4227,5	0,21%	1574,3	740	-52,99%
R.200.100.60.sop	71749	71749	0,00%	5451	1515	-72,21%
R.200.1000.1.sop	1471,5	1457,4	0,96%	4337,2	5386,4	24,19%
R.200.1000.15.sop	22362,9	21751,1	2,74%	9097,7	8880,7	-2,39%

R.200.1000.30.sop	41353,3	41276,5	0,19%	2006,8	766	-61,83%
R.200.1000.60.sop	71570,5	71585	-0,02%	4432	1375	-68,98%
R.300.100.1.sop	46,2	48	-3,90%	3947,9	3946,1	-0,05%
R.300.100.15.sop	3528,1	3447,1	2,30%	965,3	894	-7,39%
R.300.100.30.sop	6204,3	6216,4	-0,20%	541,9	194,2	-64,16%
R.300.100.60.sop	9726	9726	0,00%	2271,2	498	-78,07%
R.300.1000.1.sop	1462,9	1448,8	0,96%	848	1941,5	128,95%
R.300.1000.15.sop	33301,4	32945,7	1,07%	1532,1	1336,7	-12,75%
R.300.1000.30.sop	55545,1	55238,9	0,55%	331,4	175,4	-47,07%
R.300.1000.60.sop	109783,9	109605,8	0,16%	1586,7	383,4	-75,84%
R.400.100.1.sop	39,9	37,8	5,26%	950,9	1193,1	25,47%
R.400.100.15.sop	4761,9	4652,6	2,30%	266,6	270,3	1,39%
R.400.100.30.sop	8511	8499,9	0,13%	158,5	82	-48,26%
R.400.100.60.sop	15273,6	15275,6	-0,01%	593,8	188,5	-68,26%
R.400.1000.1.sop	1575,4	1606,7	-1,99%	229,3	656,7	186,39%
R.400.1000.15.sop	46890,7	46185,6	1,50%	308	243,2	-21,04%
R.400.1000.30.sop	87839,1	87827	0,01%	173,9	93,4	-46,29%
R.400.1000.60.sop	141531,9	141543,5	-0,01%	650	199,4	-69,32%
R.500.100.1.sop	34,9	35,5	-1,72%	452	658	45,58%
R.500.100.15.sop	6701	6671,8	0,44%	68,6	69,1	0,73%
R.500.100.30.sop	10336,8	10325,1	0,11%	82,8	53,6	-35,27%
R.500.100.60.sop	18407,9	18478,9	-0,39%	275,2	115,2	-58,14%
R.500.1000.1.sop	1683,1	1666,3	1,00%	88,1	295,5	235,41%
R.500.1000.15.sop	62426,6	62984,1	-0,89%	79,2	72,2	-8,84%
R.500.1000.30.sop	104370,2	105319,5	-0,91%	90,6	56,8	-37,31%
R.500.1000.60.sop	179575,3	180495,2	-0,51%	364,6	130,8	-64,13%
R.600.100.1.sop	29,2	27,4	6,16%	304	344	13,16%
R.600.100.15.sop	7650,8	7692,1	-0,54%	30,8	22,4	-27,27%
R.600.100.30.sop	13411,5	13596,7	-1,38%	63,8	40,5	-36,52%
R.600.100.60.sop	23996,6	24017,1	-0,09%	209,9	100,5	-52,12%
R.600.1000.1.sop	1799,9	1797,8	0,12%	47,5	134,8	183,79%
R.600.1000.15.sop	74328,1	75750,8	-1,91%	49,2	28,4	-42,28%
R.600.1000.30.sop	137129,7	138091,2	-0,70%	55,2	37,4	-32,25%
R.600.1000.60.sop	224034,2	225190,7	-0,52%	164,4	100	-39,17%
R.700.100.1.sop	26,6	27,9	-4,89%	167,4	236,2	41,10%
R.700.100.15.sop	9604,7	9607,4	-0,03%	15,6	10,6	-32,05%
R.700.100.30.sop	16282,2	16209,8	0,44%	37,5	30,3	-19,20%
R.700.100.60.sop	25309,8	25304,8	0,02%	148,3	84,9	-42,75%
R.700.1000.1.sop	1750,2	1787,6	-2,14%	31,3	69,3	121,41%
R.700.1000.15.sop	92592	91976	0,67%	18,2	13,3	-26,92%
R.700.1000.30.sop	148859,8	151410,2	-1,71%	38,1	26,4	-30,71%
R.700.1000.60.sop	257040,2	257003,6	0,01%	149,9	79	-47,30%
		Średnio:	**0,14%**		**Średnio:**	**-9,78%**

Dla proponowanej modyfikacji uzyskano poprawę otrzymywanych rezultatów średnio o $0{,}14\%$ oraz spadek liczby iteracji o $9{,}78\%$. EACSam nie powoduje większych różnic w jakości uzyskiwanych wyników, a odnotowane pozytywne i negatywne zmiany nie przejawiają zależności względem parametrów problemu.

5.5.12. Wymuszanie przynajmniej jednej zmiany w stosunku do najlepszego rozwiązania

W tabeli 5.15 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zastosowano wymuszanie przynajmniej jednej zmiany w stosunku do najlepszego rozwiązania. Dokładny opis wykorzystanej modyfikacji znajduje się w punkcie 4.2.9.

Tabela 5.15 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zastosowano wymuszanie przynajmniej jednej zmiany w stosunku do najlepszego rozwiązania

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	74,6	0,40%	11381,7	7879,1	-30,77%
R.200.100.15.sop	1925,5	1966,2	-2,11%	21796,4	16584,5	-23,91%
R.200.100.30.sop	4236,2	4239,3	-0,07%	1574,3	1040,1	-33,93%
R.200.100.60.sop	71749	71773,9	-0,03%	5451	3103,3	-43,07%
R.200.1000.1.sop	1471,5	1461,4	0,69%	4337,2	3415,8	-21,24%
R.200.1000.15.sop	22362,9	21998,4	1,63%	9097,7	9996,5	9,88%
R.200.1000.30.sop	41353,3	41396,6	-0,10%	2006,8	1491	-25,70%
R.200.1000.60.sop	71570,5	71686,5	-0,16%	4432	2339,1	-47,22%
R.300.100.1.sop	46,2	47	-1,73%	3947,9	2720	-31,10%
R.300.100.15.sop	3528,1	3582,7	-1,55%	965,3	1128,6	16,92%
R.300.100.30.sop	6204,3	6219,4	-0,24%	541,9	355,6	-34,38%
R.300.100.60.sop	9726	9726	0,00%	2271,2	1079,9	-52,45%
R.300.1000.1.sop	1462,9	1446,7	1,11%	848	681	-19,69%
R.300.1000.15.sop	33301,4	33642,9	-1,03%	1532,1	888,6	-42,00%
R.300.1000.30.sop	55545,1	55475,9	0,12%	331,4	237,2	-28,42%
R.300.1000.60.sop	109783,9	109842	-0,05%	1586,7	792,7	-50,04%
R.400.100.1.sop	39,9	40,9	-2,51%	950,9	776,6	-18,33%
R.400.100.15.sop	4761,9	4706	1,17%	266,6	222,7	-16,47%
R.400.100.30.sop	8511	8565,4	-0,64%	158,5	93,4	-41,07%
R.400.100.60.sop	15273,6	15277,5	-0,03%	593,8	319,1	-46,26%

R.400.1000.1.sop	1575,4	1602,5	-1,72%	229,3	200,3	-12,65%
R.400.1000.15.sop	46890,7	46927,5	-0,08%	308	195	-36,69%
R.400.1000.30.sop	87839,1	87986,9	-0,17%	173,9	116,6	-32,95%
R.400.1000.60.sop	141531,9	141309,2	0,16%	650	366,5	-43,62%
R.500.100.1.sop	34,9	33,6	3,72%	452	357,5	-20,91%
R.500.100.15.sop	6701	6750,2	-0,73%	68,6	59,2	-13,70%
R.500.100.30.sop	10336,8	10253,4	0,81%	82,8	53	-35,99%
R.500.100.60.sop	18407,9	18540,3	-0,72%	275,2	147,6	-46,37%
R.500.1000.1.sop	1683,1	1663,7	1,15%	88,1	85,8	-2,61%
R.500.1000.15.sop	62426,6	62202,6	0,36%	79,2	72,2	-8,84%
R.500.1000.30.sop	104370,2	106258,9	-1,81%	90,6	58,5	-35,43%
R.500.1000.60.sop	179575,3	179908,2	-0,19%	364,6	180,8	-50,41%
R.600.100.1.sop	29,2	30,3	-3,77%	304	207,4	-31,78%
R.600.100.15.sop	7650,8	7663,4	-0,16%	30,8	23,6	-23,38%
R.600.100.30.sop	13411,5	13504,2	-0,69%	63,8	44,3	-30,56%
R.600.100.60.sop	23996,6	24049,3	-0,22%	209,9	99,9	-52,41%
R.600.1000.1.sop	1799,9	1786,2	0,76%	47,5	43,8	-7,79%
R.600.1000.15.sop	74328,1	74597	-0,36%	49,2	32,5	-33,94%
R.600.1000.30.sop	137129,7	138935,8	-1,32%	55,2	32	-42,03%
R.600.1000.60.sop	224034,2	229161,1	-2,29%	164,4	88	-46,47%
R.700.100.1.sop	26,6	26,9	-1,13%	167,4	174,6	4,30%
R.700.100.15.sop	9604,7	9717,8	-1,18%	15,6	11	-29,49%
R.700.100.30.sop	16282,2	16270,6	0,07%	37,5	21,7	-42,13%
R.700.100.60.sop	25309,8	25507,7	-0,78%	148,3	64,5	-56,51%
R.700.1000.1.sop	1750,2	1726,3	1,37%	31,3	26	-16,93%
R.700.1000.15.sop	92592	91513,8	1,16%	18,2	13,5	-25,82%
R.700.1000.30.sop	148859,8	151368,4	-1,69%	38,1	24	-37,01%
R.700.1000.60.sop	257040,2	257757,1	-0,28%	149,9	68,3	-54,44%
		Średnio:	**-0,31%**		**Średnio:**	**-30,12%**

Proponowana modyfikacja powoduje pogorszenie otrzymywanych wyników średnio o $0{,}31\%$, natomiast średnia liczba wykonanych iteracji spadła o $30{,}12\%$. Spadek liczby iteracji może wynikać z narzutu obliczeniowego. Innym powodem może być wydłużenie procesu lokalnej optymalizacji spowodowane większym zróżnicowaniem kolejnych rozwiązań względem najlepszego znalezionego wyniku lub rzadszym odrzucaniem rozwiązań przed optymalizacją.

5.5.13. Zjadanie feromonu podczas lokalnej optymalizacji

W tabeli 5.16 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (EACSw), w której zastosowano zjadanie feromonu podczas lokalnej optymalizacji na wszystkich krawędziach należących do sprawdzanego

rozwiązania. Dokładny opis wykorzystanej modyfikacji znajduje się w punkcie 4.2.10.

Tabela 5.16 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (EACSw), w której zastosowano zjadanie feromonu podczas lokalnej optymalizacji na wszystkich krawędziach należących do sprawdzanego rozwiązania

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	EACSw	Poprawa	EACS	EACSw	Poprawa
R.200.100.1.sop	74,9	85	-13,48%	11381,7	310,3	-97,27%
R.200.100.15.sop	1925,5	1947,4	-1,14%	21796,4	18369,4	-15,72%
R.200.100.30.sop	4236,2	4232,3	0,09%	1574,3	782,4	-50,30%
R.200.100.60.sop	71749	71749	0,00%	5451	1187,8	-78,21%
R.200.1000.1.sop	1471,5	1595,4	-8,42%	4337,2	160,6	-96,30%
R.200.1000.15.sop	22362,9	22335,7	0,12%	9097,7	9724,7	6,89%
R.200.1000.30.sop	41353,3	41337,9	0,04%	2006,8	606,9	-69,76%
R.200.1000.60.sop	71570,5	71556	0,02%	4432	861,7	-80,56%
R.300.100.1.sop	46,2	76	-64,50%	3947,9	55,6	-98,59%
R.300.100.15.sop	3528,1	3567,2	-1,11%	965,3	1070,9	10,94%
R.300.100.30.sop	6204,3	6227,4	-0,37%	541,9	103,3	-80,94%
R.300.100.60.sop	9726	9726	0,00%	2271,2	477,5	-78,98%
R.300.1000.1.sop	1462,9	1727,3	-18,07%	848	22,3	-97,37%
R.300.1000.15.sop	33301,4	33702,7	-1,21%	1532,1	1510,4	-1,42%
R.300.1000.30.sop	55545,1	55631,9	-0,16%	331,4	176,7	-46,68%
R.300.1000.60.sop	109783,9	109843,9	-0,05%	1586,7	308,8	-80,54%
R.400.100.1.sop	39,9	95,1	-138,35%	950,9	10,4	-98,91%
R.400.100.15.sop	4761,9	4817,4	-1,17%	266,6	396,6	48,76%
R.400.100.30.sop	8511	8582,8	-0,84%	158,5	37,1	-76,59%
R.400.100.60.sop	15273,6	15232,2	0,27%	593,8	216,4	-63,56%
R.400.1000.1.sop	1575,4	2099,9	-33,29%	229,3	5,1	-97,78%
R.400.1000.15.sop	46890,7	47130,5	-0,51%	308	476,1	54,58%
R.400.1000.30.sop	87839,1	88336	-0,57%	173,9	39,9	-77,06%
R.400.1000.60.sop	141531,9	141027,6	0,36%	650	180,6	-72,22%
R.500.100.1.sop	34,9	343	-882,81%	452	2,6	-99,42%
R.500.100.15.sop	6701	6829,6	-1,92%	68,6	73	6,41%
R.500.100.30.sop	10336,8	10453,9	-1,13%	82,8	18,9	-77,17%
R.500.100.60.sop	18407,9	18303,7	0,57%	275,2	111,9	-59,34%
R.500.1000.1.sop	1683,1	6648,9	-295,04%	88,1	1,3	-98,52%
R.500.1000.15.sop	62426,6	62742,7	-0,51%	79,2	97,7	23,36%
R.500.1000.30.sop	104370,2	106712,7	-2,24%	90,6	19,1	-78,92%
R.500.1000.60.sop	179575,3	178544,1	0,57%	364,6	143,1	-60,75%
R.600.100.1.sop	29,2	508,4	-1641,10%	304	1,5	-99,51%
R.600.100.15.sop	7650,8	7760,3	-1,43%	30,8	38,9	26,30%

R.600.100.30.sop	13411,5	13455,5	-0,33%	63,8	12,2	-80,88%
R.600.100.60.sop	23996,6	23477,6	2,16%	209,9	140,8	-32,92%
R.600.1000.1.sop	1799,9	6776,9	-276,52%	47,5	1,2	-97,47%
R.600.1000.15.sop	74328,1	75091,6	-1,03%	49,2	84,1	70,93%
R.600.1000.30.sop	137129,7	136353	0,57%	55,2	12,5	-77,36%
R.600.1000.60.sop	224034,2	216531,6	3,35%	164,4	88,4	-46,23%
R.700.100.1.sop	26,6	608,2	-2186,47%	167,4	1,7	-98,98%
R.700.100.15.sop	9604,7	9744,1	-1,45%	15,6	11,4	-26,92%
R.700.100.30.sop	16282,2	15993	1,78%	37,5	8,7	-76,80%
R.700.100.60.sop	25309,8	24552,9	2,99%	148,3	71,5	-51,79%
R.700.1000.1.sop	1750,2	6610,6	-277,71%	31,3	1,1	-96,49%
R.700.1000.15.sop	92592	91764,6	0,89%	18,2	14,8	-18,68%
R.700.1000.30.sop	148859,8	148839,5	0,01%	38,1	9,5	-75,07%
R.700.1000.60.sop	257040,2	251427	2,18%	149,9	72,5	-51,63%
		Średnio:	**-121,60%**		**Średnio:**	**-54,49%**

Ten wariant diametralnie pogarsza otrzymywane wyniki oraz liczbę wykonanych iteracji. Powodu należy doszukiwać się w dużej złożoności obliczeniowej proponowanej operacji. Podczas lokalnej optymalizacji sprawdzanych jest wiele sąsiednich rozwiązań, a dla każdego z nich wykonywane jest pomniejszanie feromonu na każdej jego krawędzi.

Znacznie lepiej prezentują się wyniki dla drugiego wariantu tej modyfikacji. W tabeli 5.17 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (EACSc), w której zastosowano zjadanie feromonu podczas lokalnej optymalizacji na krawędziach nowych z perspektywy ulepszanego rozwiązania.

Tabela 5.17 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (EACSc), w której zastosowano zjadanie feromonu podczas lokalnej optymalizacji na krawędziach nowych z perspektywy ulepszanego rozwiązania

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	EACSc	Poprawa	EACS	EACSc	Poprawa
R.200.100.1.sop	74,9	73,8	1,47%	11381,7	9044,2	-20,54%
R.200.100.15.sop	1925,5	1934,8	-0,48%	21796,4	14450,4	-33,70%
R.200.100.30.sop	4236,2	4233,2	0,07%	1574,3	1309,9	-16,79%
R.200.100.60.sop	71749	71749	0,00%	5451	4685,5	-14,04%
R.200.1000.1.sop	1471,5	1464,7	0,46%	4337,2	3167,9	-26,96%
R.200.1000.15.sop	22362,9	22227,6	0,61%	9097,7	8171,6	-10,18%
R.200.1000.30.sop	41353,3	41337,5	0,04%	2006,8	1926,1	-4,02%

R.200.1000.60.sop	71570,5	71614	-0,06%	4432	4426,6	-0,12%
R.300.100.1.sop	46,2	46	0,43%	3947,9	4545,5	15,14%
R.300.100.15.sop	3528,1	3598	-1,98%	965,3	1065,9	10,42%
R.300.100.30.sop	6204,3	6228,8	-0,39%	541,9	582,6	7,51%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2358,4	3,84%
R.300.1000.1.sop	1462,9	1454,7	0,56%	848	944,4	11,37%
R.300.1000.15.sop	33301,4	34223,8	-2,77%	1532,1	1643,4	7,26%
R.300.1000.30.sop	55545,1	55562,6	-0,03%	331,4	405,8	22,45%
R.300.1000.60.sop	109783,9	109811,6	-0,03%	1586,7	1742,5	9,82%
R.400.100.1.sop	39,9	37,4	6,27%	950,9	2193,5	130,68%
R.400.100.15.sop	4761,9	4673,5	1,86%	266,6	433,3	62,53%
R.400.100.30.sop	8511	8498,6	0,15%	158,5	242,6	53,06%
R.400.100.60.sop	15273,6	15244,4	0,19%	593,8	926,5	56,03%
R.400.1000.1.sop	1575,4	1605,7	-1,92%	229,3	283	23,42%
R.400.1000.15.sop	46890,7	46574	0,68%	308	426,2	38,38%
R.400.1000.30.sop	87839,1	87142,7	0,79%	173,9	256,4	47,44%
R.400.1000.60.sop	141531,9	141379,3	0,11%	650	869,2	33,72%
R.500.100.1.sop	34,9	31	11,17%	452	1274,4	181,95%
R.500.100.15.sop	6701	6720,4	-0,29%	68,6	115	67,64%
R.500.100.30.sop	10336,8	10268,1	0,66%	82,8	106,6	28,74%
R.500.100.60.sop	18407,9	18368	0,22%	275,2	445,4	61,85%
R.500.1000.1.sop	1683,1	1656,7	1,57%	88,1	135,4	53,69%
R.500.1000.15.sop	62426,6	62625	-0,32%	79,2	125,9	58,96%
R.500.1000.30.sop	104370,2	103980,9	0,37%	90,6	138,7	53,09%
R.500.1000.60.sop	179575,3	179444,3	0,07%	364,6	548,5	50,44%
R.600.100.1.sop	29,2	27,7	5,14%	304	890,2	192,83%
R.600.100.15.sop	7650,8	7548,4	1,34%	30,8	52,2	69,48%
R.600.100.30.sop	13411,5	13303,8	0,80%	63,8	102,1	60,03%
R.600.100.60.sop	23996,6	23679,4	1,32%	209,9	349,3	66,41%
R.600.1000.1.sop	1799,9	1761,2	2,15%	47,5	66,7	40,42%
R.600.1000.15.sop	74328,1	74728	-0,54%	49,2	78,7	59,96%
R.600.1000.30.sop	137129,7	134886,1	1,64%	55,2	70,2	27,17%
R.600.1000.60.sop	224034,2	221158,3	1,28%	164,4	264,2	60,71%
R.700.100.1.sop	26,6	25,1	5,64%	167,4	726,5	333,99%
R.700.100.15.sop	9604,7	9730	-1,30%	15,6	15,4	-1,28%
R.700.100.30.sop	16282,2	16045,3	1,45%	37,5	50,2	33,87%
R.700.100.60.sop	25309,8	25113,6	0,78%	148,3	175,8	18,54%
R.700.1000.1.sop	1750,2	1777,6	-1,57%	31,3	39,5	26,20%
R.700.1000.15.sop	92592	89452,8	3,39%	18,2	29,3	60,99%
R.700.1000.30.sop	148859,8	148228,7	0,42%	38,1	46,4	21,78%
R.700.1000.60.sop	257040,2	254692,9	0,91%	149,9	210,4	40,36%
		Średnio:	**0,88%**		**Średnio:**	**43,22%**

Proponowana modyfikacja EACSc powoduje poprawę otrzymywanych wyników średnio o $0{,}88\%$, natomiast średnia liczba wykonanych iteracji wzrosła o

$43{,}22\%$. Dla dziewięciu problemów odnotowano pogorszenie rezultatów o więcej niż $0{,}1\%$, z czego sześć instancji charakteryzuje wartość parametru $p = 15$. Średnia wartość poprawy dla wszystkich problemów spełniających warunek $p = 15$ wynosi $0{,}02\%$.

Zwiększenie liczby iteracji średnio o $43{,}22\%$ może wynikać z częstszego odrzucania badanych rozwiązań przed podjęciem próby lokalnej optymalizacji. Powodów takiego zachowania można doszukiwać się w większym zróżnicowaniu w wyborach mrówek, czego osiągnięcie jest celem EACSc.

5.5.14. Akceptowanie rozwiązania o tym samym koszcie

W tabeli 5.18 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zastosowano akceptowanie rozwiązania o tym samym koszcie, co najlepsze dotychczas znalezione rozwiązanie. Dokładny opis wykorzystanej modyfikacji znajduje się w punkcie 4.2.11.

Tabela 5.18 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zastosowano akceptowanie rozwiązania o tym samym koszcie, co najlepsze dotychczas znalezione rozwiązanie

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	71,6	4,41%	11381,7	9305,1	-18,25%
R.200.100.15.sop	1925,5	1944,5	-0,99%	21796,4	17837,2	-18,16%
R.200.100.30.sop	4236,2	4244,3	-0,19%	1574,3	1352,2	-14,11%
R.200.100.60.sop	71749	71749	0,00%	5451	4830,9	-11,38%
R.200.1000.1.sop	1471,5	1470,7	0,05%	4337,2	3582,6	-17,40%
R.200.1000.15.sop	22362,9	21957,8	1,81%	9097,7	9601,4	5,54%
R.200.1000.30.sop	41353,3	41339,1	0,03%	2006,8	1920,6	-4,30%
R.200.1000.60.sop	71570,5	71628,5	-0,08%	4432	4288,6	-3,24%
R.300.100.1.sop	46,2	40,7	11,90%	3947,9	2594,7	-34,28%
R.300.100.15.sop	3528,1	3512,6	0,44%	965,3	998,9	3,48%
R.300.100.30.sop	6204,3	6211,9	-0,12%	541,9	501,5	-7,46%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2198,8	-3,19%
R.300.1000.1.sop	1462,9	1477,2	-0,98%	848	825,7	-2,63%
R.300.1000.15.sop	33301,4	33728,7	-1,28%	1532,1	1370,1	-10,57%
R.300.1000.30.sop	55545,1	55366,8	0,32%	331,4	284,4	-14,18%
R.300.1000.60.sop	109783,9	109756,2	0,03%	1586,7	1557,6	-1,83%

R.400.100.1.sop	39,9	35,1	12,03%	950,9	881,9	-7,26%
R.400.100.15.sop	4761,9	4761,7	0,00%	266,6	264,8	-0,68%
R.400.100.30.sop	8511	8499,5	0,14%	158,5	151,9	-4,16%
R.400.100.60.sop	15273,6	15276,7	-0,02%	593,8	616,4	3,81%
R.400.1000.1.sop	1575,4	1601,5	-1,66%	229,3	218,3	-4,80%
R.400.1000.15.sop	46890,7	47011,7	-0,26%	308	273,3	-11,27%
R.400.1000.30.sop	87839,1	87542	0,34%	173,9	155,8	-10,41%
R.400.1000.60.sop	141531,9	141516,2	0,01%	650	681	4,77%
R.500.100.1.sop	34,9	31,1	10,89%	452	371,5	-17,81%
R.500.100.15.sop	6701	6639,1	0,92%	68,6	95,3	38,92%
R.500.100.30.sop	10336,8	10364	-0,26%	82,8	77,5	-6,40%
R.500.100.60.sop	18407,9	18483,3	-0,41%	275,2	280,2	1,82%
R.500.1000.1.sop	1683,1	1677,6	0,33%	88,1	88,3	0,23%
R.500.1000.15.sop	62426,6	62254,4	0,28%	79,2	95,6	20,71%
R.500.1000.30.sop	104370,2	104093,6	0,27%	90,6	84,4	-6,84%
R.500.1000.60.sop	179575,3	179712,5	-0,08%	364,6	338,2	-7,24%
R.600.100.1.sop	29,2	26,1	10,62%	304	254,6	-16,25%
R.600.100.15.sop	7650,8	7656,5	-0,07%	30,8	35,7	15,91%
R.600.100.30.sop	13411,5	13495,4	-0,63%	63,8	63,9	0,16%
R.600.100.60.sop	23996,6	24034	-0,16%	209,9	208	-0,91%
R.600.1000.1.sop	1799,9	1755,6	2,46%	47,5	48,3	1,68%
R.600.1000.15.sop	74328,1	74092,5	0,32%	49,2	44,7	-9,15%
R.600.1000.30.sop	137129,7	135904,9	0,89%	55,2	47,8	-13,41%
R.600.1000.60.sop	224034,2	226028	-0,89%	164,4	172	4,62%
R.700.100.1.sop	26,6	23,9	10,15%	167,4	161,7	-3,41%
R.700.100.15.sop	9604,7	9609,1	-0,05%	15,6	15,7	0,64%
R.700.100.30.sop	16282,2	16178,6	0,64%	37,5	42,3	12,80%
R.700.100.60.sop	25309,8	25212,7	0,38%	148,3	150,2	1,28%
R.700.1000.1.sop	1750,2	1764,5	-0,82%	31,3	26,4	-15,65%
R.700.1000.15.sop	92592	90860,1	1,87%	18,2	15,8	-13,19%
R.700.1000.30.sop	148859,8	149678,5	-0,55%	38,1	36,5	-4,20%
R.700.1000.60.sop	257040,2	257167,8	-0,05%	149,9	168,1	12,14%
		Średnio:	**1,29%**		**Średnio:**	**-3,86%**

Proponowana modyfikacja powoduje średnią poprawę uzyskiwanych wyników na poziomie $1{,}29\%$. Należy jednak zwrócić uwagę na fakt, że wartość ta jest zawyżana przez problemy o parametrach $r = 100$ i $p = 1$, dla których średnia poprawa wynosi 10%. Dla pozostałych problemów poprawa jest równa jedynie 0,02%. Wynika z tego, że zEACS uzyskuje znaczną poprawę otrzymywanych rezultatów dla pewnej szczególnej grupy problemów, podczas gdy nie wykazuje znaczącej różnicy względem EACS dla pozostałych instancji.

5.5.15. Wyłączenie heurystyki z algorytmu

W tabeli 5.19 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zrezygnowano z zastosowania heurystyki. Dokładny opis wykorzystanej modyfikacji znajduje się w punkcie 4.2.12.

Tabela 5.19 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zrezygnowano z zastosowania heurystyki

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	100,5	-34,18%	11381,7	48041	322,09%
R.200.100.15.sop	1925,5	2016	-4,70%	21796,4	26236,9	20,37%
R.200.100.30.sop	4236,2	4223,2	0,31%	1574,3	2333	48,19%
R.200.100.60.sop	71749	71773,9	-0,03%	5451	4927,2	-9,61%
R.200.1000.1.sop	1471,5	1576,8	-7,16%	4337,2	50543	1065,34%
R.200.1000.15.sop	22362,9	21870,4	2,20%	9097,7	26059,7	186,44%
R.200.1000.30.sop	41353,3	41483,7	-0,32%	2006,8	3318,3	65,35%
R.200.1000.60.sop	71570,5	71570,5	0,00%	4432	5131	15,77%
R.300.100.1.sop	46,2	92,6	-100,43%	3947,9	25882,8	555,61%
R.300.100.15.sop	3528,1	3558,9	-0,87%	965,3	6068,2	528,63%
R.300.100.30.sop	6204,3	6148,2	0,90%	541,9	569,4	5,07%
R.300.100.60.sop	9726	9727,2	-0,01%	2271,2	1837,7	-19,09%
R.300.1000.1.sop	1462,9	1776,9	-21,46%	848	21224,6	2402,90%
R.300.1000.15.sop	33301,4	32914,7	1,16%	1532,1	6039,4	294,19%
R.300.1000.30.sop	55545,1	54449,3	1,97%	331,4	726,9	119,34%
R.300.1000.60.sop	109783,9	109682,9	0,09%	1586,7	1465,5	-7,64%
R.400.100.1.sop	39,9	104,9	-162,91%	950,9	13217,9	1290,04%
R.400.100.15.sop	4761,9	4574,9	3,93%	266,6	2246,4	742,61%
R.400.100.30.sop	8511	8394,4	1,37%	158,5	190,8	20,38%
R.400.100.60.sop	15273,6	15273,8	0,00%	593,8	589,8	-0,67%
R.400.1000.1.sop	1575,4	1960,6	-24,45%	229,3	10891,4	4649,85%
R.400.1000.15.sop	46890,7	45149,9	3,71%	308	2150,6	598,25%
R.400.1000.30.sop	87839,1	86764,2	1,22%	173,9	164,7	-5,29%
R.400.1000.60.sop	141531,9	141664,1	-0,09%	650	525,1	-19,22%
R.500.100.1.sop	34,9	100,6	-188,25%	452	9424,1	1984,98%
R.500.100.15.sop	6701	6419,6	4,20%	68,6	556,5	711,22%
R.500.100.30.sop	10336,8	10205,5	1,27%	82,8	69,3	-16,30%
R.500.100.60.sop	18407,9	19138,8	-3,97%	275,2	202,9	-26,27%
R.500.1000.1.sop	1683,1	2227,3	-32,33%	88,1	7360	8254,14%
R.500.1000.15.sop	62426,6	60595,2	2,93%	79,2	919,8	1061,36%
R.500.1000.30.sop	104370,2	105154	-0,75%	90,6	84,3	-6,95%
R.500.1000.60.sop	179575,3	181880,2	-1,28%	364,6	209,5	-42,54%

R.600.100.1.sop	29,2	94,8	-224,66%	304	6578	2063,82%
R.600.100.15.sop	7650,8	7352,3	3,90%	30,8	304,3	887,99%
R.600.100.30.sop	13411,5	13442,6	-0,23%	63,8	61,9	-2,98%
R.600.100.60.sop	23996,6	24668,9	-2,80%	209,9	124,4	-40,73%
R.600.1000.1.sop	1799,9	2234,7	-24,16%	47,5	4444,7	9257,26%
R.600.1000.15.sop	74328,1	73017,9	1,76%	49,2	446,7	807,93%
R.600.1000.30.sop	137129,7	136637,7	0,36%	55,2	47,7	-13,59%
R.600.1000.60.sop	224034,2	234511,8	-4,68%	164,4	111,6	-32,12%
R.700.100.1.sop	26,6	76,6	-187,97%	167,4	4885,1	2818,22%
R.700.100.15.sop	9604,7	9433	1,79%	15,6	181,6	1064,10%
R.700.100.30.sop	16282,2	15634,7	3,98%	37,5	40,2	7,20%
R.700.100.60.sop	25309,8	28029,6	-10,75%	148,3	74,5	-49,76%
R.700.1000.1.sop	1750,2	2287,9	-30,72%	31,3	3698,8	11717,25%
R.700.1000.15.sop	92592	89429,4	3,42%	18,2	188,2	934,07%
R.700.1000.30.sop	148859,8	146104,1	1,85%	38,1	35,8	-6,04%
R.700.1000.60.sop	257040,2	279453,4	-8,72%	149,9	79,2	-47,16%
		Średnio:	**-21,57%**		**Średnio:**	**1128,21%**

Proponowana modyfikacja powoduje pogorszenie uzyskiwanych rezultatów średnio o $21{,}57\%$. Zwiększona o $1128{,}21\%$ przeciętna liczba iteracji najprawdopodobniej wynika z wyznaczania wielu mało atrakcyjnych rozwiązań, dla których nie była podejmowana próba poprawienia ich za pomocą procesu lokalnej optymalizacji. Zaskakujący jest fakt, że odnotowano średnią poprawę uzyskiwanych rezultatów na poziomie $1{,}47\%$ dla problemów o parametrze $p \in \{15, 30\}$. Sugeruje to, że heurystyka dla tych problemów działa odwrotnie do swojego założenia i odpycha algorytm EACS od obiecujących obszarów przestrzeni rozwiązań.

5.5.16. Wyłączenie heurystyki z algorytmu w połączeniu z przerywaniem budowania kosztownych rozwiązań

W tabeli 5.20 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zrezygnowano z zastosowania heurystyki oraz zastosowano przerywanie budowania kosztownych rozwiązań. Dokładny opis wykorzystanych modyfikacji znajduje się w punktach 4.2.12 i 4.2.1.

Tabela 5.20 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zrezygnowano z zastosowania heurystyki oraz zastosowano przerywanie budowania kosztownych rozwiązań

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	101	-34,85%	11381,7	55359,3	386,39%
R.200.100.15.sop	1925,5	1991	-3,40%	21796,4	31672,5	45,31%
R.200.100.30.sop	4236,2	4224,4	0,28%	1574,3	2500	58,80%
R.200.100.60.sop	71749	71823,7	-0,10%	5451	4649,1	-14,71%
R.200.1000.1.sop	1471,5	1568,4	-6,59%	4337,2	53106,9	1124,45%
R.200.1000.15.sop	22362,9	21887,6	2,13%	9097,7	26763,2	194,18%
R.200.1000.30.sop	41353,3	41401,9	-0,12%	2006,8	3671,5	82,95%
R.200.1000.60.sop	71570,5	71628,5	-0,08%	4432	5504,1	24,19%
R.300.100.1.sop	46,2	105,6	-128,57%	3947,9	29805,6	654,97%
R.300.100.15.sop	3528,1	3545,6	-0,50%	965,3	7101,1	635,64%
R.300.100.30.sop	6204,3	6192,9	0,18%	541,9	528	-2,57%
R.300.100.60.sop	9726	9727,6	-0,02%	2271,2	2172,5	-4,35%
R.300.1000.1.sop	1462,9	1784,2	-21,96%	848	22414,7	2543,24%
R.300.1000.15.sop	33301,4	32905,1	1,19%	1532,1	8072,9	426,92%
R.300.1000.30.sop	55545,1	54685,8	1,55%	331,4	774,1	133,58%
R.300.1000.60.sop	109783,9	109715,3	0,06%	1586,7	1715,6	8,12%
R.400.100.1.sop	39,9	110,2	-176,19%	950,9	16303,6	1614,54%
R.400.100.15.sop	4761,9	4591,3	3,58%	266,6	2169,7	713,84%
R.400.100.30.sop	8511	8370	1,66%	158,5	239,6	51,17%
R.400.100.60.sop	15273,6	15298,9	-0,17%	593,8	558,4	-5,96%
R.400.1000.1.sop	1575,4	1974	-25,30%	229,3	10938,2	4670,26%
R.400.1000.15.sop	46890,7	45352,4	3,28%	308	2459,9	698,67%
R.400.1000.30.sop	87839,1	86967,4	0,99%	173,9	171,8	-1,21%
R.400.1000.60.sop	141531,9	141302,6	0,16%	650	540,9	-16,78%
R.500.100.1.sop	34,9	94,1	-169,63%	452	10296,6	2178,01%
R.500.100.15.sop	6701	6436,5	3,95%	68,6	675,6	884,84%
R.500.100.30.sop	10336,8	10145,8	1,85%	82,8	83,2	0,48%
R.500.100.60.sop	18407,9	18989,5	-3,16%	275,2	226,9	-17,55%
R.500.1000.1.sop	1683,1	2191,8	-30,22%	88,1	8393,7	9427,47%
R.500.1000.15.sop	62426,6	60821,6	2,57%	79,2	859,7	985,48%
R.500.1000.30.sop	104370,2	104257,2	0,11%	90,6	91,8	1,32%
R.500.1000.60.sop	179575,3	181313,2	-0,97%	364,6	240,5	-34,04%
R.600.100.1.sop	29,2	96,8	-231,51%	304	7298,7	2300,89%
R.600.100.15.sop	7650,8	7300,1	4,58%	30,8	339,3	1001,62%
R.600.100.30.sop	13411,5	13259,8	1,13%	63,8	55,6	-12,85%
R.600.100.60.sop	23996,6	24432,6	-1,82%	209,9	152,6	-27,30%
R.600.1000.1.sop	1799,9	2248,7	-24,93%	47,5	5074,8	10583,79%
R.600.1000.15.sop	74328,1	73238,7	1,47%	49,2	466,3	847,76%

R.600.1000.30.sop	137129,7	134632	1,82%	55,2	53,4	-3,26%
R.600.1000.60.sop	224034,2	233213,8	-4,10%	164,4	104,2	-36,62%
R.700.100.1.sop	26,6	70,5	-165,04%	167,4	5013,9	2895,16%
R.700.100.15.sop	9604,7	9410,4	2,02%	15,6	152,6	878,21%
R.700.100.30.sop	16282,2	15808,6	2,91%	37,5	37,5	0,00%
R.700.100.60.sop	25309,8	27537,8	-8,80%	148,3	80,1	-45,99%
R.700.1000.1.sop	1750,2	2339,2	-33,65%	31,3	4014,5	12725,88%
R.700.1000.15.sop	92592	90452	2,31%	18,2	172,7	848,90%
R.700.1000.30.sop	148859,8	146955,6	1,28%	38,1	34,8	-8,66%
R.700.1000.60.sop	257040,2	276204,5	-7,46%	149,9	80,5	-46,30%
		Średnio:	**-21,63%**		**Średnio:**	**1236,44%**

Otrzymane wyniki przeprowadzonego badania są bardzo zbliżone do rezultatów otrzymanych dla zmodyfikowanej wersji EACS, w której zrezygnowano z zastosowania heurystyki, jednak nie zastosowano przerywania budowania kosztownych rozwiązań. Można również zauważyć te same zależności pomiędzy parametrami problemów, a uzyskanymi wynikami. Średnie pogorszenie dla zEACS wynosi $21{,}63\%$. Odnotowano średnią poprawę uzyskiwanych rezultatów na poziomie $1{,}53\%$ dla problemów o parametrze $p \in \{15, 30\}$.

5.5.17. Użycie wykorzystanych dróg podczas inicjalizacji stosu w procesie lokalnej optymalizacji

W tabeli 5.21 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zmieniono sposób inicjalizacji stosu w procesie lokalnej optymalizacji. Dokładny opis wykorzystanej modyfikacji znajduje się w punkcie 4.2.13.

Tabela 5.21 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której zmieniono sposób inicjalizacji stosu w procesie lokalnej optymalizacji

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	74,9	0,00%	11381,7	32877,8	188,87%
R.200.100.15.sop	1925,5	1959,6	-1,77%	21796,4	20419,5	-6,32%
R.200.100.30.sop	4236,2	4231,9	0,10%	1574,3	1535,1	-2,49%
R.200.100.60.sop	71749	71749	0,00%	5451	4804,8	-11,85%
R.200.1000.1.sop	1471,5	1450,2	1,45%	4337,2	18872,2	335,12%

R.200.1000.15.sop	22362,9	22157,7	0,92%	9097,7	8844,8	-2,78%
R.200.1000.30.sop	41353,3	41346,7	0,02%	2006,8	2110,9	5,19%
R.200.1000.60.sop	71570,5	71556	0,02%	4432	3993,9	-9,88%
R.300.100.1.sop	46,2	45,7	1,08%	3947,9	20060,3	408,13%
R.300.100.15.sop	3528,1	3560,9	-0,93%	965,3	1361,7	41,06%
R.300.100.30.sop	6204,3	6188,9	0,25%	541,9	544,9	0,55%
R.300.100.60.sop	9726	9726	0,00%	2271,2	1967,8	-13,36%
R.300.1000.1.sop	1462,9	1427,8	2,40%	848	7141,9	742,21%
R.300.1000.15.sop	33301,4	33094,9	0,62%	1532,1	1600,6	4,47%
R.300.1000.30.sop	55545,1	55688	-0,26%	331,4	347,4	4,83%
R.300.1000.60.sop	109783,9	109680,9	0,09%	1586,7	1352,6	-14,75%
R.400.100.1.sop	39,9	34,3	14,04%	950,9	11517,4	1111,21%
R.400.100.15.sop	4761,9	4764,2	-0,05%	266,6	283	6,15%
R.400.100.30.sop	8511	8415,5	1,12%	158,5	148,7	-6,18%
R.400.100.60.sop	15273,6	15270,9	0,02%	593,8	598,2	0,74%
R.400.1000.1.sop	1575,4	1522,2	3,38%	229,3	2248,4	880,55%
R.400.1000.15.sop	46890,7	46382	1,08%	308	378,9	23,02%
R.400.1000.30.sop	87839,1	87999,3	-0,18%	173,9	182,8	5,12%
R.400.1000.60.sop	141531,9	141506,6	0,02%	650	552,3	-15,03%
R.500.100.1.sop	34,9	24,7	29,23%	452	7426,8	1543,10%
R.500.100.15.sop	6701	6653,1	0,71%	68,6	78,4	14,29%
R.500.100.30.sop	10336,8	10261,1	0,73%	82,8	87,1	5,19%
R.500.100.60.sop	18407,9	18488,1	-0,44%	275,2	277,5	0,84%
R.500.1000.1.sop	1683,1	1558,7	7,39%	88,1	987,4	1020,77%
R.500.1000.15.sop	62426,6	62553,7	-0,20%	79,2	106,3	34,22%
R.500.1000.30.sop	104370,2	104338,3	0,03%	90,6	100,7	11,15%
R.500.1000.60.sop	179575,3	180072,7	-0,28%	364,6	296	-18,82%
R.600.100.1.sop	29,2	16,8	42,47%	304	5112,4	1581,71%
R.600.100.15.sop	7650,8	7700,5	-0,65%	30,8	36,4	18,18%
R.600.100.30.sop	13411,5	13361,1	0,38%	63,8	68,7	7,68%
R.600.100.60.sop	23996,6	23938,3	0,24%	209,9	197,4	-5,96%
R.600.1000.1.sop	1799,9	1665,1	7,49%	47,5	465,3	879,58%
R.600.1000.15.sop	74328,1	73682,3	0,87%	49,2	49,5	0,61%
R.600.1000.30.sop	137129,7	137235	-0,08%	55,2	52	-5,80%
R.600.1000.60.sop	224034,2	224541,4	-0,23%	164,4	156,9	-4,56%
R.700.100.1.sop	26,6	14,7	44,74%	167,4	3690,1	2104,36%
R.700.100.15.sop	9604,7	9608,6	-0,04%	15,6	17,3	10,90%
R.700.100.30.sop	16282,2	16120,3	0,99%	37,5	42	12,00%
R.700.100.60.sop	25309,8	25194,6	0,46%	148,3	136,3	-8,09%
R.700.1000.1.sop	1750,2	1602,7	8,43%	31,3	309	887,22%
R.700.1000.15.sop	92592	91033,4	1,68%	18,2	22,1	21,43%
R.700.1000.30.sop	148859,8	150047,6	-0,80%	38,1	40,1	5,25%
R.700.1000.60.sop	257040,2	256455,9	0,23%	149,9	132,3	-11,74%
		Średnio:	**3,47%**		**Średnio:**	**245,38%**

Proponowana modyfikacja powoduje średnią poprawę uzyskiwanych wyników na poziomie $3{,}47\%$. Należy jednak zauważyć, że najbardziej znacząca poprawa wynosząca $13{,}51\%$ występuje dla problemów, których parametr $p = 1$. Dla pozostałych instancji średnia poprawa to jedynie $0{,}13\%$. Również poprawa średniej liczby iteracji, jest znacznie większa, gdy $p = 1$ i wynosi $973{,}57\%$ podczas gdy dla $p \neq 1$ jest równa $2{,}47\%$.

5.5.18. Brak odparowywania feromonu podczas globalnej aktualizacji

W tabeli 5.22 zestawiono wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której nie zastosowano odparowywania feromonu podczas globalnej aktualizacji. Dokładny opis wykorzystanej modyfikacji znajduje się w punkcie 4.2.14.

Tabela 5.22 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której nie zastosowano odparowywania feromonu podczas globalnej aktualizacji

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	74,3	0,80%	11381,7	9981,8	-12,30%
R.200.100.15.sop	1925,5	1974,9	-2,57%	21796,4	16848,4	-22,70%
R.200.100.30.sop	4236,2	4254,2	-0,42%	1574,3	1290,8	-18,01%
R.200.100.60.sop	71749	71773,9	-0,03%	5451	4948,9	-9,21%
R.200.1000.1.sop	1471,5	1492,5	-1,43%	4337,2	3541,1	-18,36%
R.200.1000.15.sop	22362,9	22408,4	-0,20%	9097,7	8501,3	-6,56%
R.200.1000.30.sop	41353,3	41415,6	-0,15%	2006,8	2017,5	0,53%
R.200.1000.60.sop	71570,5	71628,5	-0,08%	4432	4474,6	0,96%
R.300.100.1.sop	46,2	48,7	-5,41%	3947,9	3962,6	0,37%
R.300.100.15.sop	3528,1	3562,9	-0,99%	965,3	1078,3	11,71%
R.300.100.30.sop	6204,3	6194,7	0,15%	541,9	540,5	-0,26%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2415	6,33%
R.300.1000.1.sop	1462,9	1470,4	-0,51%	848	902	6,37%
R.300.1000.15.sop	33301,4	33944,4	-1,93%	1532,1	1267,2	-17,29%
R.300.1000.30.sop	55545,1	55352,1	0,35%	331,4	362,1	9,26%
R.300.1000.60.sop	109783,9	109700,8	0,08%	1586,7	1659,8	4,61%
R.400.100.1.sop	39,9	39	2,26%	950,9	1452,1	52,71%
R.400.100.15.sop	4761,9	4746,3	0,33%	266,6	265,7	-0,34%
R.400.100.30.sop	8511	8500,1	0,13%	158,5	156,4	-1,32%
R.400.100.60.sop	15273,6	15284,4	-0,07%	593,8	654,4	10,21%

R.400.1000.1.sop	1575,4	1590	-0,93%	229,3	214,8	-6,32%
R.400.1000.15.sop	46890,7	47059,3	-0,36%	308	300,7	-2,37%
R.400.1000.30.sop	87839,1	87638	0,23%	173,9	170,7	-1,84%
R.400.1000.60.sop	141531,9	141512,4	0,01%	650	739,7	13,80%
R.500.100.1.sop	34,9	33,4	4,30%	452	470,8	4,16%
R.500.100.15.sop	6701	6686,4	0,22%	68,6	90,2	31,49%
R.500.100.30.sop	10336,8	10222,2	1,11%	82,8	86,9	4,95%
R.500.100.60.sop	18407,9	18451,4	-0,24%	275,2	313,2	13,81%
R.500.1000.1.sop	1683,1	1659,7	1,39%	88,1	100,8	14,42%
R.500.1000.15.sop	62426,6	61903,1	0,84%	79,2	92,7	17,05%
R.500.1000.30.sop	104370,2	104689,1	-0,31%	90,6	94,8	4,64%
R.500.1000.60.sop	179575,3	179895,4	-0,18%	364,6	406,3	11,44%
R.600.100.1.sop	29,2	27,1	7,19%	304	265,3	-12,73%
R.600.100.15.sop	7650,8	7630,3	0,27%	30,8	34,2	11,04%
R.600.100.30.sop	13411,5	13405,6	0,04%	63,8	60,7	-4,86%
R.600.100.60.sop	23996,6	23887,7	0,45%	209,9	232,2	10,62%
R.600.1000.1.sop	1799,9	1788,2	0,65%	47,5	49,9	5,05%
R.600.1000.15.sop	74328,1	75939,9	-2,17%	49,2	37,4	-23,98%
R.600.1000.30.sop	137129,7	136813,3	0,23%	55,2	53,4	-3,26%
R.600.1000.60.sop	224034,2	224192,4	-0,07%	164,4	188,3	14,54%
R.700.100.1.sop	26,6	28,3	-6,39%	167,4	204	21,86%
R.700.100.15.sop	9604,7	9729,4	-1,30%	15,6	15,7	0,64%
R.700.100.30.sop	16282,2	16055,1	1,39%	37,5	42,9	14,40%
R.700.100.60.sop	25309,8	25259,4	0,20%	148,3	155,1	4,59%
R.700.1000.1.sop	1750,2	1762,7	-0,71%	31,3	25,2	-19,49%
R.700.1000.15.sop	92592	91781,3	0,88%	18,2	17,5	-3,85%
R.700.1000.30.sop	148859,8	148596,8	0,18%	38,1	35,7	-6,30%
R.700.1000.60.sop	257040,2	257346,7	-0,12%	149,9	157,3	4,94%
		Średnio:	**-0,06%**		**Średnio:**	**2,40%**

Proponowana modyfikacja powoduje średnie pogorszenie uzyskiwanych wyników na poziomie $0{,}06\%$ przy równoczesnej poprawie średniej liczby iteracji o $2{,}40\%$. Wahania te są nieznaczne i mogą wynikać z losowego charakteru algorytmów mrówkowych. Chociaż po zastosowaniu modyfikacji nie odnotowano znaczącej różnicy w otrzymywanych wynikach, to korzyścią z jej zastosowania jest nieznaczne uproszczenie algorytmu.

W trakcie badań podjęto próbę wyznaczenia dobrej wartości współczynnika wyparowywania ρ dla algorytmu zEACS. Pomimo iż symulowanie wyparowywania nie ma miejsca, to jest on wykorzystywany podczas dodawania feromonu. Badania te zostały wykonane na problemie *R.200.100.15* dla czasu pracy algorytmu ustawionego na 30 sekund. Najlepsze wyniki uzyskano dla $\rho =$

0,0257171923795521. Wartość ta została wykorzystana do przeprowadzenia standardowej serii badań, których rezultaty znajdują się w tabeli 5.23.

Tabela 5.23 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS), w której nie zastosowano odparowywania feromonu podczas globalnej aktualizacji dla zmienionego parametru ρ

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS	Poprawa	EACS	zEACS	Poprawa
R.200.100.1.sop	74,9	73,5	1,87%	11381,7	8619,3	-24,27%
R.200.100.15.sop	1925,5	1943,4	-0,93%	21796,4	13539,9	-37,88%
R.200.100.30.sop	4236,2	4251,6	-0,36%	1574,3	753,4	-52,14%
R.200.100.60.sop	71749	71749	0,00%	5451	2279,7	-58,18%
R.200.1000.1.sop	1471,5	1462,8	0,59%	4337,2	4144,6	-4,44%
R.200.1000.15.sop	22362,9	22223	0,63%	9097,7	7778,4	-14,50%
R.200.1000.30.sop	41353,3	41346,6	0,02%	2006,8	902,8	-55,01%
R.200.1000.60.sop	71570,5	71585	-0,02%	4432	1904,9	-57,02%
R.300.100.1.sop	46,2	43,1	6,71%	3947,9	3629,7	-8,06%
R.300.100.15.sop	3528,1	3546,7	-0,53%	965,3	667,3	-30,87%
R.300.100.30.sop	6204,3	6189,8	0,23%	541,9	251	-53,68%
R.300.100.60.sop	9726	9726	0,00%	2271,2	957,6	-57,84%
R.300.1000.1.sop	1462,9	1426,1	2,52%	848	1054,9	24,40%
R.300.1000.15.sop	33301,4	33311,9	-0,03%	1532,1	1206,1	-21,28%
R.300.1000.30.sop	55545,1	55450,2	0,17%	331,4	200,6	-39,47%
R.300.1000.60.sop	109783,9	109728,5	0,05%	1586,7	683,1	-56,95%
R.400.100.1.sop	39,9	37,1	7,02%	950,9	1137,1	19,58%
R.400.100.15.sop	4761,9	4811,9	-1,05%	266,6	236,8	-11,18%
R.400.100.30.sop	8511	8453,4	0,68%	158,5	78,6	-50,41%
R.400.100.60.sop	15273,6	15247,9	0,17%	593,8	311	-47,63%
R.400.1000.1.sop	1575,4	1595,4	-1,27%	229,3	284,3	23,99%
R.400.1000.15.sop	46890,7	47035,2	-0,31%	308	228	-25,97%
R.400.1000.30.sop	87839,1	87290,1	0,63%	173,9	101,1	-41,86%
R.400.1000.60.sop	141531,9	141300,6	0,16%	650	347,8	-46,49%
R.500.100.1.sop	34,9	32,6	6,59%	452	468,8	3,72%
R.500.100.15.sop	6701	6645,4	0,83%	68,6	73,2	6,71%
R.500.100.30.sop	10336,8	10255,5	0,79%	82,8	41,8	-49,52%
R.500.100.60.sop	18407,9	18364,3	0,24%	275,2	160,2	-41,79%
R.500.1000.1.sop	1683,1	1612,1	4,22%	88,1	120,3	36,55%
R.500.1000.15.sop	62426,6	62161,1	0,43%	79,2	68,1	-14,02%
R.500.1000.30.sop	104370,2	103972,9	0,38%	90,6	45,5	-49,78%
R.500.1000.60.sop	179575,3	178987,2	0,33%	364,6	189,5	-48,03%
R.600.100.1.sop	29,2	27,9	4,45%	304	273	-10,20%
R.600.100.15.sop	7650,8	7701,2	-0,66%	30,8	23,6	-23,38%

R.600.100.30.sop	13411,5	13428,3	-0,13%	63,8	33,9	-46,87%
R.600.100.60.sop	23996,6	23685,8	1,30%	209,9	123,8	-41,02%
R.600.1000.1.sop	1799,9	1770	1,66%	47,5	54,9	15,58%
R.600.1000.15.sop	74328,1	74454	-0,17%	49,2	32,1	-34,76%
R.600.1000.30.sop	137129,7	135320,7	1,32%	55,2	28,3	-48,73%
R.600.1000.60.sop	224034,2	219650,1	1,96%	164,4	96,8	-41,12%
R.700.100.1.sop	26,6	22,3	16,17%	167,4	242,2	44,68%
R.700.100.15.sop	9604,7	9529,8	0,78%	15,6	10,7	-31,41%
R.700.100.30.sop	16282,2	16068,4	1,31%	37,5	20,5	-45,33%
R.700.100.60.sop	25309,8	24901,2	1,61%	148,3	83,9	-43,43%
R.700.1000.1.sop	1750,2	1753,9	-0,21%	31,3	33,5	7,03%
R.700.1000.15.sop	92592	90779,1	1,96%	18,2	16,4	-9,89%
R.700.1000.30.sop	148859,8	147645,6	0,82%	38,1	19,4	-49,08%
R.700.1000.60.sop	257040,2	253054,5	1,55%	149,9	80,8	-46,10%
		Średnio:	**1,34%**		**Średnio:**	**-26,82%**

Po zmianie parametru ρ otrzymano średnią poprawę rezultatów na poziomie $1{,}34\%$ przy jednoczesnym spadku liczby iteracji o $26{,}82\%$. Największą poprawę, która wyniosła $4{,}19\%$ odnotowano dla problemów o parametrze $p = 1$. Poprawa dla pozostałych problemów wyniosła $0{,}39\%$.

5.5.19. Połączenie modyfikacji

Po wykonaniu badań opisanych w punktach 5.5.1 – 5.5.18 podjęto próbę połączenia modyfikacji, których zastosowanie powodowało poprawę otrzymywanych wyników. W tym celu wykonano 28 badań z wykorzystaniem zaimplementowanego mechanizmu testującego różne kombinacje parametrów. Badania te miały czas wyznaczania parametrów ograniczony do 24 godzin, oraz czas wykonania algorytmu mrówkowego do 30 sekund. Zostały one przeprowadzone na problemie *R.200.100.15*. Dodatkowo dla pięciu z wyznaczonych w ten sposób kombinacji wykonano standardową serię badań. Przeprowadzono również jedną standardową serię badań dla wybranych modyfikacji. W dwóch z sześciu przeprowadzonych serii badań uzyskano dobre wyniki. W pierwszej z nich zastosowano następujące modyfikacje[9]:

- początkowa wartość feromonu na krawędzi zależna od kosztu (4.2.5);

[9] W nawiasach podano punkty, w których znajduje się dokładny opis danej modyfikacji.

- zjadanie feromonu podczas lokalnej optymalizacji – wariant zjadania na krawędziach nowych (4.2.10);
- akceptowanie rozwiązania o tym samym koszcie (4.2.11);
- użycie wykorzystanych dróg podczas inicjalizacji stosu w procesie lokalnej optymalizacji (4.2.13);
- brak odparowywania feromonu podczas globalnej aktualizacji (4.2.14).

Algorytm powstały w wyniku zastosowania powyższych modyfikacji w dalszej części tego punktu nazywany będzie zEACS1. Rezultaty badań przeprowadzonych na tym algorytmie przedstawiono w tabeli 5.24.

Tabela 5.24 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS1), w której zastosowano pięć różnych modyfikacji

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS1	Poprawa	EACS	zEACS1	Poprawa
R.200.100.1.sop	74,9	69,6	7,08%	11381,7	38895,3	241,74%
R.200.100.15.sop	1925,5	1937,8	-0,64%	21796,4	19951,9	-8,46%
R.200.100.30.sop	4236,2	4235,7	0,01%	1574,3	1485,4	-5,65%
R.200.100.60.sop	71749	71749	0,00%	5451	5160,7	-5,33%
R.200.1000.1.sop	1471,5	1465	0,44%	4337,2	15318	253,18%
R.200.1000.15.sop	22362,9	22098,1	1,18%	9097,7	8731,7	-4,02%
R.200.1000.30.sop	41353,3	41363,8	-0,03%	2006,8	2287,4	13,98%
R.200.1000.60.sop	71570,5	71556	0,02%	4432	4352,8	-1,79%
R.300.100.1.sop	46,2	35,1	24,03%	3947,9	21390	441,81%
R.300.100.15.sop	3528,1	3539,3	-0,32%	965,3	1888	95,59%
R.300.100.30.sop	6204,3	6183,3	0,34%	541,9	549,4	1,38%
R.300.100.60.sop	9726	9726	0,00%	2271,2	2474,7	8,96%
R.300.1000.1.sop	1462,9	1449,5	0,92%	848	5738,5	576,71%
R.300.1000.15.sop	33301,4	33521,7	-0,66%	1532,1	2225,3	45,25%
R.300.1000.30.sop	55545,1	55688,6	-0,26%	331,4	421,4	27,16%
R.300.1000.60.sop	109783,9	109748,2	0,03%	1586,7	1568,6	-1,14%
R.400.100.1.sop	39,9	28,4	28,82%	950,9	11967,5	1158,54%
R.400.100.15.sop	4761,9	4703,1	1,23%	266,6	455,2	70,74%
R.400.100.30.sop	8511	8395,5	1,36%	158,5	208	31,23%
R.400.100.60.sop	15273,6	15281,6	-0,05%	593,8	904,9	52,39%
R.400.1000.1.sop	1575,4	1510,8	4,10%	229,3	2718,9	1085,74%
R.400.1000.15.sop	46890,7	46490,6	0,85%	308	554	79,87%
R.400.1000.30.sop	87839,1	87323,3	0,59%	173,9	302,1	73,72%
R.400.1000.60.sop	141531,9	141338,4	0,14%	650	812,1	24,94%

R.500.100.1.sop	34,9	20,4	41,55%	452	7720,1	1607,99%
R.500.100.15.sop	6701	6625,4	1,13%	68,6	163	137,61%
R.500.100.30.sop	10336,8	10206,5	1,26%	82,8	122,3	47,71%
R.500.100.60.sop	18407,9	18382,5	0,14%	275,2	469,6	70,64%
R.500.1000.1.sop	1683,1	1540,5	8,47%	88,1	1524	1629,85%
R.500.1000.15.sop	62426,6	63036,1	-0,98%	79,2	166,6	110,35%
R.500.1000.30.sop	104370,2	103732,2	0,61%	90,6	125,2	38,19%
R.500.1000.60.sop	179575,3	178868	0,39%	364,6	539,2	47,89%
R.600.100.1.sop	29,2	17,1	41,44%	304	5574,8	1733,82%
R.600.100.15.sop	7650,8	7498,4	1,99%	30,8	78	153,25%
R.600.100.30.sop	13411,5	13095,7	2,35%	63,8	83,6	31,03%
R.600.100.60.sop	23996,6	23606,4	1,63%	209,9	296,3	41,16%
R.600.1000.1.sop	1799,9	1639,3	8,92%	47,5	828	1643,16%
R.600.1000.15.sop	74328,1	73133,5	1,61%	49,2	106	115,45%
R.600.1000.30.sop	137129,7	134761,5	1,73%	55,2	61,1	10,69%
R.600.1000.60.sop	224034,2	220308,7	1,66%	164,4	226,1	37,53%
R.700.100.1.sop	26,6	17,1	35,71%	167,4	3887,7	2222,40%
R.700.100.15.sop	9604,7	9404,2	2,09%	15,6	23,5	50,64%
R.700.100.30.sop	16282,2	15659,6	3,82%	37,5	40,8	8,80%
R.700.100.60.sop	25309,8	24736	2,27%	148,3	144,6	-2,49%
R.700.1000.1.sop	1750,2	1605,1	8,29%	31,3	584,3	1766,77%
R.700.1000.15.sop	92592	89502,4	3,34%	18,2	40,7	123,63%
R.700.1000.30.sop	148859,8	147155,1	1,15%	38,1	32,1	-15,75%
R.700.1000.60.sop	257040,2	253755,1	1,28%	149,9	162	8,07%
		Średnio:	**5,02%**		**Średnio:**	**330,73%**

Proponowany zestaw modyfikacji powoduje średnią poprawę uzyskiwanych wyników na poziomie $5{,}02\%$ przy równoczesnej poprawie średniej liczby iteracji o $330{,}73\%$. W tabeli 5.25 przedstawiono średnią poprawę wyników oraz liczby iteracji dla problemów zgrupowanych po parametrze p.

Tabela 5.25 Średnia poprawa wyników oraz liczby iteracji dla zEACS1 w zależności od parametru problemu p

Wartość p	*Poprawa średniego wyniku*	*Poprawa średniej liczby iteracji*
1	17,48%	1196,81%
15	0,90%	80,82%
30	1,08%	21,87%
60	0,63%	23,40%

Rezultaty badań wskazują na to, że zEACS1 znacząco poprawia uzyskiwane wyniki dla problemów o małej liczbie ograniczeń, poprawiając również, średnio o $0{,}87\%$, jakość rozwiązań dla pozostałych problemów.

Drugi algorytm (zEACS2), dla którego uzyskano dobre wyniki, zawierał następujące modyfikacje[10]:

- przerywanie budowania kosztownych rozwiązań (4.2.1);
- zwiększanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny (4.2.8):
  - realizacja wyłącznie przez mnożenie;
  - $\mu = 1{,}18752824765987$;
  - $s_b = 0{,}271577729269073$;
  - wyłączone resetowanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny s;
- zjadanie feromonu podczas lokalnej optymalizacji – wariant zjadania na krawędziach nowych (4.2.10);
- akceptowanie rozwiązania o tym samym koszcie (4.2.11);
- użycie wykorzystanych dróg podczas inicjalizacji stosu w procesie lokalnej optymalizacji (4.2.13);
- brak odparowywania feromonu podczas globalnej aktualizacji (4.2.14).

Dodatkowo część parametrów algorytmu została zmodyfikowana:

- oczekiwana liczba węzłów wybranych w sposób probabilistyczny $s = 9{,}65145604555136$;
- współczynnik wyparowywania $\rho = 0{,}113918090968428$.

Rezultaty badań przeprowadzonych na zEACS2 przedstawiono w tabeli 5.26.

[10] W nawiasach podano punkty, w których znajduje się dokładny opis danej modyfikacji.

Tabela 5.26 Wyniki symulacji dla algorytmu EACS oraz jego zmienionej wersji (zEACS2), w której zastosowano sześć różnych modyfikacji

	Średni koszt			*Średnia liczba iteracji*		
Plik	EACS	zEACS2	Poprawa	EACS	zEACS2	Poprawa
R.200.100.1.sop	74,9	67,7	9,61%	11381,7	31284,4	174,87%
R.200.100.15.sop	1925,5	1897,7	1,44%	21796,4	18865,9	-13,44%
R.200.100.30.sop	4236,2	4216,8	0,46%	1574,3	803,1	-48,99%
R.200.100.60.sop	71749	71749	0,00%	5451	1029,5	-81,11%
R.200.1000.1.sop	1471,5	1455,1	1,11%	4337,2	23104,8	432,71%
R.200.1000.15.sop	22362,9	21454,9	4,06%	9097,7	11899,1	30,79%
R.200.1000.30.sop	41353,3	41294,7	0,14%	2006,8	633,5	-68,43%
R.200.1000.60.sop	71570,5	71556	0,02%	4432	913,7	-79,38%
R.300.100.1.sop	46,2	34,4	25,54%	3947,9	17819,3	351,36%
R.300.100.15.sop	3528,1	3397,4	3,70%	965,3	1320,3	36,78%
R.300.100.30.sop	6204,3	6213,2	-0,14%	541,9	116,2	-78,56%
R.300.100.60.sop	9726	9726	0,00%	2271,2	336,6	-85,18%
R.300.1000.1.sop	1462,9	1426,2	2,51%	848	9394,9	1007,89%
R.300.1000.15.sop	33301,4	32478,1	2,47%	1532,1	1609,5	5,05%
R.300.1000.30.sop	55545,1	55006,3	0,97%	331,4	148,1	-55,31%
R.300.1000.60.sop	109783,9	109498,7	0,26%	1586,7	236,7	-85,08%
R.400.100.1.sop	39,9	23,8	40,35%	950,9	9316,5	879,76%
R.400.100.15.sop	4761,9	4615,1	3,08%	266,6	408,5	53,23%
R.400.100.30.sop	8511	8465,5	0,53%	158,5	36,3	-77,10%
R.400.100.60.sop	15273,6	15281,1	-0,05%	593,8	101,9	-82,84%
R.400.1000.1.sop	1575,4	1536,5	2,47%	229,3	3596,1	1468,29%
R.400.1000.15.sop	46890,7	45198,2	3,61%	308	328,5	6,66%
R.400.1000.30.sop	87839,1	88411,5	-0,65%	173,9	36,3	-79,13%
R.400.1000.60.sop	141531,9	141315,9	0,15%	650	96,4	-85,17%
R.500.100.1.sop	34,9	16,1	53,87%	452	5587,6	1136,19%
R.500.100.15.sop	6701	6490,9	3,14%	68,6	101,9	48,54%
R.500.100.30.sop	10336,8	10375,4	-0,37%	82,8	20,1	-75,72%
R.500.100.60.sop	18407,9	18426,1	-0,10%	275,2	47,2	-82,85%
R.500.1000.1.sop	1683,1	1594,5	5,26%	88,1	2110,3	2295,35%
R.500.1000.15.sop	62426,6	62205,7	0,35%	79,2	104,8	32,32%
R.500.1000.30.sop	104370,2	106297,1	-1,85%	90,6	20,9	-76,93%
R.500.1000.60.sop	179575,3	179326,3	0,14%	364,6	53,3	-85,38%
R.600.100.1.sop	29,2	11,2	61,64%	304	3371,4	1009,01%
R.600.100.15.sop	7650,8	7629,1	0,28%	30,8	27,9	-9,42%
R.600.100.30.sop	13411,5	13534,2	-0,91%	63,8	16,8	-73,67%
R.600.100.60.sop	23996,6	23730,1	1,11%	209,9	34,7	-83,47%
R.600.1000.1.sop	1799,9	1712,2	4,87%	47,5	1137,3	2294,32%
R.600.1000.15.sop	74328,1	75223,7	-1,20%	49,2	38,3	-22,15%
R.600.1000.30.sop	137129,7	138064,8	-0,68%	55,2	15,3	-72,28%
R.600.1000.60.sop	224034,2	223161,9	0,39%	164,4	28,6	-82,60%

R.700.100.1.sop	26,6	9,9	62,78%	167,4	2188,6	1207,41%
R.700.100.15.sop	9604,7	9650,9	-0,48%	15,6	11,3	-27,56%
R.700.100.30.sop	16282,2	16055,2	1,39%	37,5	14,3	-61,87%
R.700.100.60.sop	25309,8	25082,5	0,90%	148,3	22,9	-84,56%
R.700.1000.1.sop	1750,2	1681,2	3,94%	31,3	973,2	3009,27%
R.700.1000.15.sop	92592	91324,8	1,37%	18,2	15,3	-15,93%
R.700.1000.30.sop	148859,8	150308,1	-0,97%	38,1	14	-63,25%
R.700.1000.60.sop	257040,2	255382,8	0,64%	149,9	23,7	-84,19%
		Średnio:	**6,19%**		**Średnio:**	**282,46%**

Proponowany zestaw modyfikacji powoduje średnią poprawę uzyskiwanych wyników na poziomie $6{,}19\%$ przy równoczesnej poprawie średniej liczby iteracji o $282{,}46\%$. W tabeli 5.27 przedstawiono średnią poprawę wyników oraz liczby iteracji dla problemów zgrupowanych po parametrze p.

Tabela 5.27 Średnia poprawa wyników oraz liczby iteracji dla zEACS1 w zależności od parametru problemu p

Wartość p	*Poprawa średniego wyniku*	*Poprawa średniej liczby iteracji*
1	22,83%	1272,20%
15	1,82%	10,40%
30	-0,17%	-69,27%
60	0,29%	-83,48%

Rezultaty badań wskazują na to, że zEACS2 znacząco poprawia uzyskiwane wyniki dla problemów o małej liczbie ograniczeń, poprawiając przy tym, średnio o $0{,}64\%$, jakość rozwiązań w przypadku pozostałych problemów. Na uwagę zasługuje fakt, że odnotowano średnie pogorszenie uzyskiwanych rezultatów na poziomie $0{,}17\%$ dla problemów o parametrze $p = 30$.

Porównując rezultaty zEACS1 i zEACS2 łatwo zauważyć, że zEACS1 uzyskuje lepsze wyniki dla problemów o parametrze $p \geq 30$, natomiast dla pozostałych lepszy jest zEACS2.

5.6. PODSUMOWANIE NAJBARDZIEJ OBIECUJĄCYCH REZULTATÓW

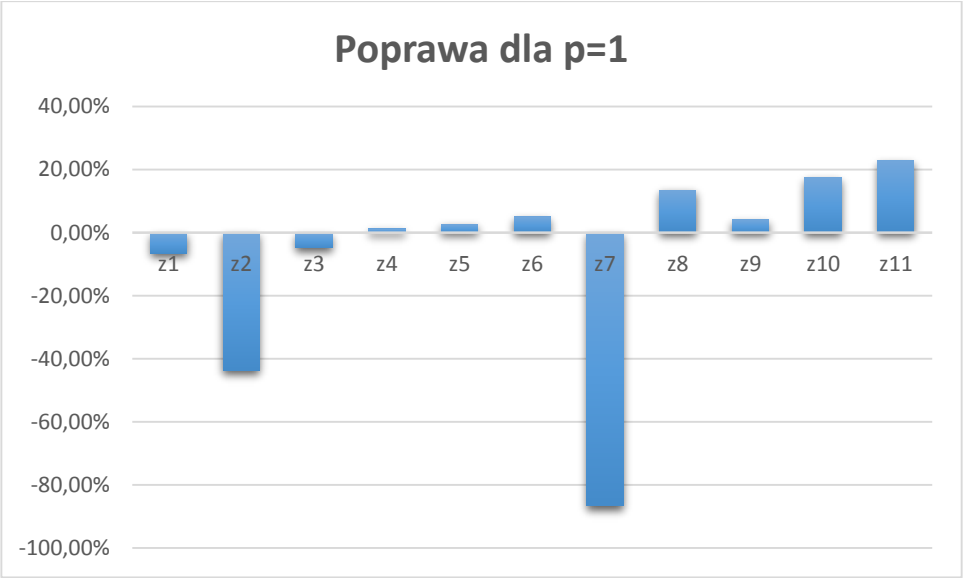
W niniejszym podrozdziale wykonano zestawienie wyników uzyskanych dla wybranych modyfikacji. Modyfikacje wykorzystane w zestawieniu wraz z oznaczeniami[11]:

- z1 – początkowa wartość feromonu na krawędzi zależna od kosztu (5.5.5);
- z2 – wybór deterministyczny jedynym sposobem wyboru krawędzi należącej do najlepszego rozwiązania (5.5.8);
- z3 – zmniejszanie znaczenia heurystyki z kolejnymi iteracjami i przerywanie budowania kosztownych rozwiązań (5.5.10);
- z4 – zwiększanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny (5.5.11):
  - realizacja wyłącznie przez mnożenie;
  - $\mu = 1{,}18337445543562$;
  - $s_b = 0{,}271577729269073$;
  - włączone resetowanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny s;
- z5 – zjadanie feromonu podczas lokalnej optymalizacji – wariant zjadania na krawędziach nowych (5.5.13);
- z6 – akceptowanie rozwiązania o tym samym koszcie (5.5.14);
- z7 – wyłączenie heurystyki z algorytmu (5.5.15);
- z8 – użycie wykorzystanych dróg podczas inicjalizacji stosu w procesie lokalnej optymalizacji (5.5.17);
- z9 – brak odparowywania feromonu podczas globalnej aktualizacji (5.5.18):
  - współczynnik wyparowywania $\rho = 0{,}0257171923795521$;
- z10 – połączenie modyfikacji – algorytm zEACS1 (5.5.19):
  - początkowa wartość feromonu na krawędzi zależna od kosztu;
  - zjadanie feromonu podczas lokalnej optymalizacji – wariant zjadania na krawędziach nowych;

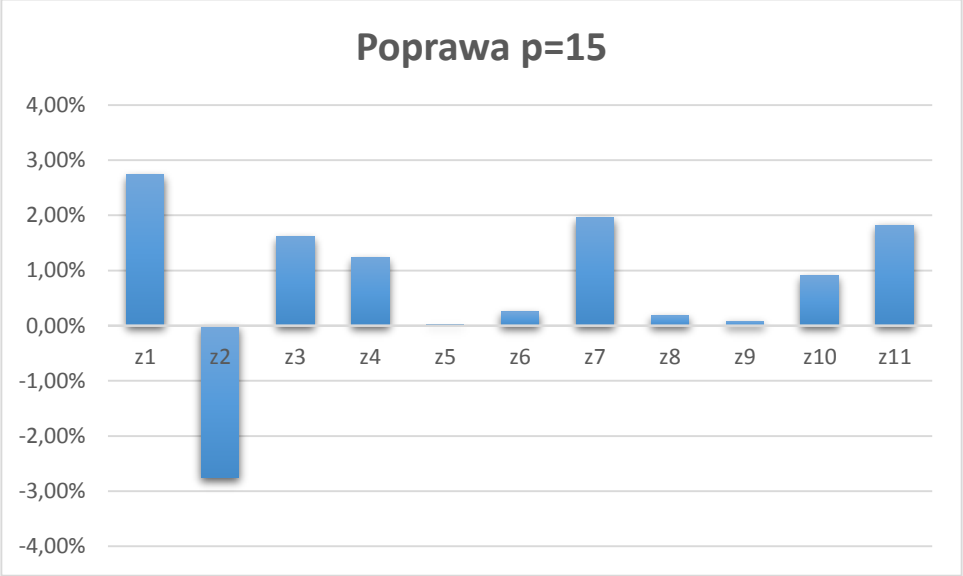
[11] W nawiasach podano punkty, w których znajdują się szczegółowe wyniki badań.

    - akceptowanie rozwiązania o tym samym koszcie;
    - użycie wykorzystanych dróg podczas inicjalizacji stosu w procesie lokalnej optymalizacji;
    - brak odparowywania feromonu podczas globalnej aktualizacji;
- z11 – połączenie modyfikacji – algorytm zEACS2 (5.5.19):
    - przerywanie budowania kosztownych rozwiązań;
    - zwiększanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny:
        - realizacja wyłącznie przez mnożenie;
        - $\mu = 1{,}18752824765987$;
        - $s_b = 0{,}271577729269073$;
        - wyłączone resetowanie oczekiwanej liczby węzłów wybranych w sposób probabilistyczny s;
    - zjadanie feromonu podczas lokalnej optymalizacji – wariant zjadania na krawędziach nowych;
    - akceptowanie rozwiązania o tym samym koszcie;
    - użycie wykorzystanych dróg podczas inicjalizacji stosu w procesie lokalnej optymalizacji;
    - brak odparowywania feromonu podczas globalnej aktualizacji;
    - oczekiwana liczba węzłów wybranych w sposób probabilistyczny $s = 9{,}65145604555136$;
    - współczynnik wyparowywania $\rho = 0{,}113918090968428$.

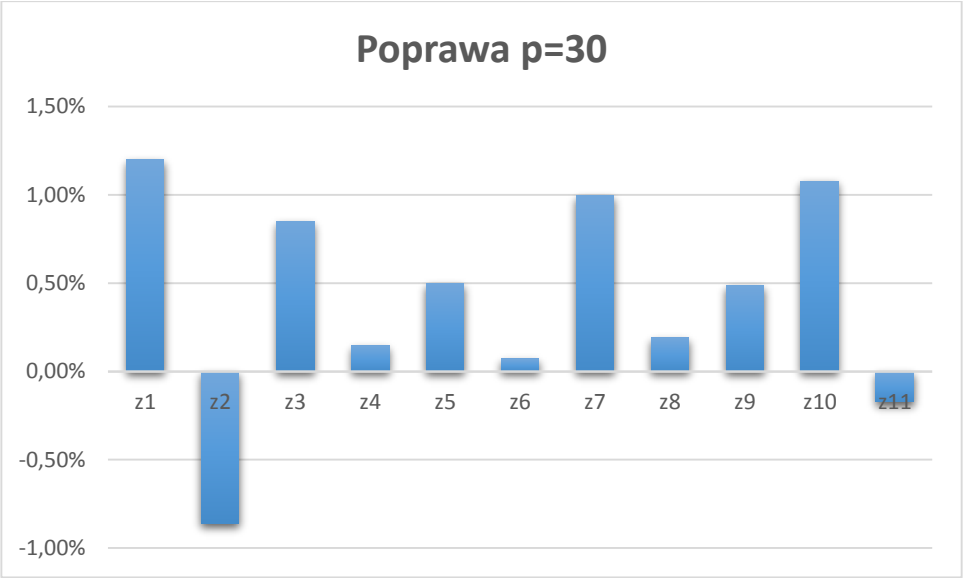
Zachowania poszczególnych modyfikacji najczęściej różniły się w zależności od liczby ograniczeń, dlatego też zestawienie wykonano według parametru p instancji problemów. Wykresy na rysunkach 5.2 – 5.5 przedstawiają uzyskaną średnią poprawę rezultatów dla różnych wartości parametru p.



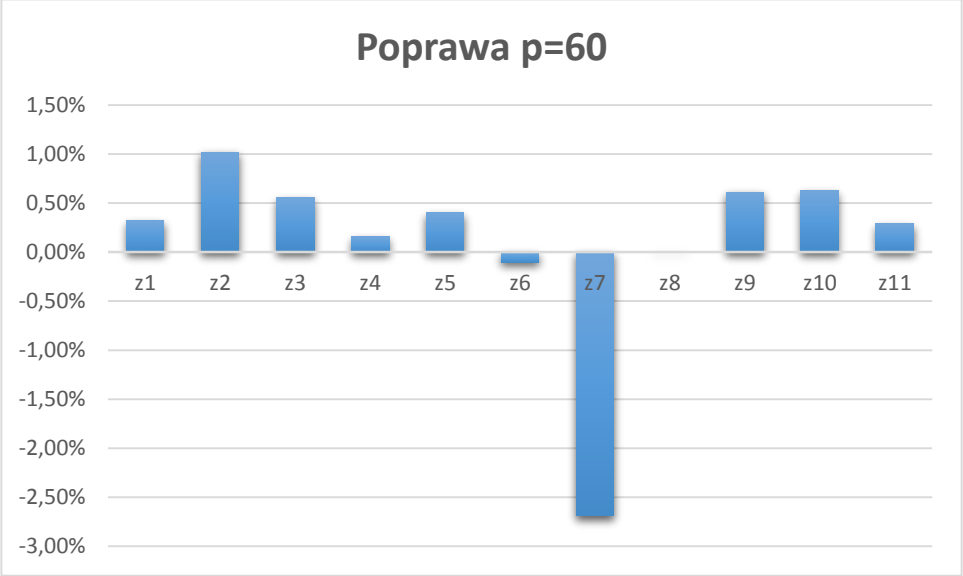
Rysunek 5.2 Wykres prezentujący średnią zmianę uzyskiwanych wyników dla różnych algorytmów i parametru $p = 1$



Rysunek 5.3 Wykres prezentujący średnią zmianę uzyskiwanych wyników dla różnych algorytmów i parametru $p = 15$



Rysunek 5.4 Wykres prezentujący średnią zmianę uzyskiwanych wyników dla różnych algorytmów i parametru $p = 30$



Rysunek 5.5 Wykres prezentujący średnią zmianę uzyskiwanych wyników dla różnych algorytmów i parametru $p = 60$

Algorytmy, dla których uzyskano poprawę średniego rezultatu dla każdej grupy problemów w zależności od parametru p to: z4, z5, z8, z9 i z10. Z tych algorytmów największą średnią poprawę uzyskał algorytm z10 $(5{,}02\%)$.

6. ZAKOŃCZENIE

Cel pracy, czyli ulepszenie najlepszego istniejącego algorytmu mrówkowego wykorzystywanego do rozwiązywania problemów sekwencyjnego porządkowania, został osiągnięty. Zaproponowano czternaście różnych modyfikacji oraz przeprowadzono szereg badań dla każdej z nich, jak również dla wielu ich kombinacji. Najlepsze wyniki otrzymano podczas wykorzystania jednocześnie pięciu z proponowanych zmian – uzyskano średnią poprawę rezultatów na poziomie $5{,}02\%$. Zmiany te miały na celu ulepszenie różnych aspektów EACS. Uzależnienie początkowej ilości feromonu od kosztu krawędzi miało na celu upodobnienie działania algorytmu, a dokładniej danych, na których operuje, do środowiska naturalnego. Poskutkowało ono lepszym doborem przestrzeni poszukiwań przez mrówki. Zjadanie feromonu podczas lokalnej optymalizacji jak i akceptowanie rozwiązania o tym samym koszcie, co rozwiązanie najlepsze, miały służyć zwiększeniu różnorodności badanych ścieżek. Użycie wykorzystanych dróg podczas inicjalizacji stosu w procesie lokalnej optymalizacji miało przyspieszyć działanie algorytmu, a brak odparowywania feromonu uprościć zasadę działania.

Wiele z proponowanych zmian nie odniosło zamierzonego skutku. Rezultaty niektórych badań eksperymentalnych były zaskakujące, jak na przykład dla modyfikacji polegającej na usunięciu heurystyki z algorytmu otrzymano poprawę jakości otrzymywanych rozwiązań dla pewnej grupy problemów.

Podczas wykonywanych prac napotkano kilka trudności. Pierwszą z nich był dobór parametrów algorytmu. Niektóre z proponowanych modyfikacji zostały oparte o parametry ciągłe, inne zmieniają wpływ istniejących parametrów na pracę algorytmu. Również dobór atrybutów podczas wykorzystywania kilku modyfikacji, jak i wybór, które zmiany powinny być stosowane w połączeniu ze sobą, okazały się aspektami problematycznymi. Z problemem tym poradzono sobie poprzez zaimplementowanie mechanizmu testującego opartego o algorytm symulowanego wyżarzania. Jego zadaniem jest iteracyjna zmiana zadanych parametrów, w tym parametrów uruchamiających/wyłączających dane modyfikacje, i badanie otrzymywanych rezultatów. Kolejnym problemem był długi czas potrzebny do wykonania badań. Zbiór danych testowych [10] zawiera 48 instancji problemów sekwencyjnego porządkowania. Badania wykonywano poprzez dziesięciokrotne

uruchomienie algorytmu dla każdej z nich, z czasem wykonania ograniczonym do dziesięciu minut. Wykonano 25 serii takich badań, a dodatkowo 48 badań wykorzystujących mechanizm testujący, z czasem wykonania ograniczonym do dwudziestu czterech godzin. Daje to w sumie ponad sto trzydzieści jeden dni obliczeń. Problem ten rozwiązano poprzez zastosowanie wielowątkowości.

Podczas badań eksperymentalnych zauważono, że najczęściej uzyskiwana poprawa lub pogorszenie wyników dla różnych wersji EACS występowało w dużej zależności od liczby ograniczeń występujących w problemie. Podobne zależności odnotowano również w pracy [9]. Dlatego też sugerowanym kierunkiem dalszych badań jest podjęcie próby wyznaczenia dokładniejszych zależności pomiędzy proponowanymi zmianami a liczbą ograniczeń dla danego problemu. Umożliwiłoby to implementację algorytmu mrówkowego, który dynamicznie wybierałby, których modyfikacji użyć w zależności od specyfiki rozwiązywanego problemu.

Bibliografia

1. Aho A. V., Hopcroft J. E., Ullman J. D.: *Projektowanie i analiza algorytmów*. Wydawnictwo Helion, Gliwice 2003.
2. Cormen T. H., Leiserson C. E., Stein C., Rivest R. L.: *Wprowadzenie do algorytmów*. WNT, Warszawa 2004.
3. Reingold E. M., Nievergelt J., Deo N.: *Algorytmy kombinatoryczne*. PWN, Warszawa 1985.
4. Wilson R. J.: *Wprowadzenie do teorii grafów*. Wydawnictwo Naukowe PWN, Warszawa 2008.
5. Michalewicz Z., Fogel D.B.: *How to Solve It: Modern Heuristics*, Springer, 2000.
6. Rayward-Smith V.J., Osman I.H., Reeves C.R., Smith C.R.: *Modern Heuristic Search Methods*. Wiley, 1996.
7. Gambardella L. M., Montemanni R., Weyland D.: *An Enhanced Ant Colony System for the Sequential Ordering Problem*. Operations Research Proceedings 2011, 355-360, 2011.
8. Gambardella L. M., Dorigo M.: *An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem.* INFORMS Journal on Computing, vol. 12, 237-255, 2000.
9. Ezzat A.: *Ant Colony Optimization Approaches for the Sequential Ordering Problem*, Master of Science Thesis, The American University in Cairo, Department of Computer Science and Engineering, 2013. https://dar.aucegypt.edu/handle/10526/3629
10. Zbiór danych testowych *SOPLIB2006.* http://www.idsia.ch/~roberto/SOPLIB06.zip
11. Montemanni R., Smith D.H., Gambardella L.M.: *A heuristic manipulation technique for the sequential ordering problem*. Computers & Operations Research, vol. 35, 3931-3944, 2008.
12. Dorigo M.: *Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, 1992.

13. Escudero L. F.: *An inexact algorithm for the sequential ordering problem*. European Journal of Operational Research, vol. 37, 236-249, 1988.
14. Savelsbergh M. W. P.: *An efficient implementation of local search algorithms for constrained routing problems*. European Journal of Operational Research, vol. 47, 75-85, 1990.
15. Ascheuer N.: *Hamiltonian path problems in the on-line optimization of flexible manufacturing systems*. PhD thesis, Technische Universität, 1995.
16. Ezzat A., Abdelbar A. M., Wunsch II D. C.: *A Bare-Bones ACO Algorithm that Performs Competitively on the Sequential Order Problem*. Numeric Computing, vol. 6, 19-29, 2014.
17. Jayadeva, Shah S., Bhaya A., Kothari R., Chandra S.: *Ants find the shortest path: a mathematical proof*. Swarm Intelligence, vol. 7, 43-62, 2013.