Spis treści

Rozdział 1

Wstęp

Rozdział 2

Przegląd literatury

2.1 Cele badania przestrzeni rozwiązań

Analiza przestrzeni rozwiązań pozwala na lepsze poznanie niektórych cech problemu, a także zbadanie, w jakim stopniu cechy te zależne są od rodzaju badanej instancji. Przedmiotem badań z tej dziedziny najczęściej są znane problemy NP-trudne, takie jak problem komiwojażera czy problem kwadratowego przydziału. Jednym z proponowanych zastosowań analizy przestrzeni rozwiązań jest wykorzystanie jej do wyboru najlepszego algorytmu heurystycznego dla danej instancji problemu. W pracy *Local Optima Networks in Solving Algorithm Selection Problem for TSP*[1] sieci optimów lokalnych wygenerowane na podstawie próbkowania przestrzeni rozwiązań wykorzystano do nauczenia modeli regresji, których zadaniem było przewidzenie, który algorytm heurystyczny da lepsze wyniki dla danej instancji problemu. Badanie wykazało, że analiza przestrzeni rozwiązań może zostać z powodzeniem wykorzystana w tym celu. Problemem pozostaje natomiast długi czas trwania próbkowania przestrzeni, zwykle dłuższy niż czas działania samego algorytmu heurystycznego. W pracy *Mapping the global structure of TSP Fitness landscapes*[7] zbadano przestrzeń rozwiązań dla różnych instancji problemu komiwojażera. Zauważono, że instancje wygenerowane losowo zwykle mają mniejszą neutralność i mniej globalnych optimów od instancji rzeczywistych. Zaobserwowano również, że sposób rozłożenia miast (w klastrach, równomierny) wpływa na wzajemną korelację wartości różnych miar. W ostatnich latach w podobny sposób przeprowadzono analizy przestrzeni konfiguracji wieloparametrowych algorytmów optymalizacyjnych. W pracy *Understanding Parameter Spaces using Local Optima Networks: A Case Study on Particle Swarm Optimization*[2] wykorzystano sieci lokalnych optimów, oraz pochodne struktury CMLON do analizy i wizualizacji przestrzeni parametrów algorytmu roju cząstek. Analiza wykazała istnienie dużej ilości lokalnych optimów, niską neutralność, oraz istnienie wielu ścieków (ang sinks) nie znajdujących się w optimum globalnym, Sugeruje to, że naiwne metody dobierania parametrów mogą łatwo doprowadzić do suboptymalnej konfiguracji i w efekcie nie otrzymaniu najlepszych wyników. Podobne badanie wykonano dla przestrzeni parametrów procesu AutoML w pracy *Understanding AutoML Search Spaces with Local Optima Networks*[10]. AutoML jest procesem automatyzacji konfiguracji procesów (ang. pipelines) uczenia maszynowego obejmującym m. in. wybór zastosowanego przetwarzania wstępnego, właściwego algorytmu uczenia i jego hiperparametrów. W pracy zbadano przestrzeń konfiguracji AutoML dla zadania klasyfikacji. TODO: expand

2.2 Próbkowanie przestrzeni rozwiązań

Przegląd zupełny przestrzeni rozwiązań problemów trudnych obliczeniowo jest w praktyce niemożliwy poza bardzo małymi instancjami problemu. Z tego powodu analizę przestrzeni rozwiązań przeprowadza się na części przestrzeni zbadanej w procesie próbkowania. W tym miejscu pojawiają się pytanie: w jaki sposób próbkować przestrzeń, aby cechy jej zbadanego fragmentu jak najlepiej odzwierciedlały cechy całej przestrzeni?

Próbkowanie przestrzeni rozwiązań i budowanie na jej podstawie sieci optimów lokalnych odbywa się poprzez zastosowanie metody

Najczęściej stosowane algorytmy próbkowania można podzielić na trzy kategorie: Próbkowanie dwufazowe, *Markov-chain* oraz *Snowball*.

Próbkowanie dwufazowe składa się z dwóch faz - najpierw próbkowane są wierzchołki poprzez zastosowanie metody iteracyjnego przeszukiwania lokalnego[8]. Metoda ta polega na generowaniu losowego rozwiązania, a następnie wykorzystaniu algorytmu optymalizacji lokalnej do znalezienia najbliższego lokalnego optimum. Następuje po tym wylosowanie nowego punktu i powtórzenie procedury. Druga faza to faza próbkowania krawędzi, polegająca na poddaniu znalezionych wcześniej rozwiązań operacji perturbacji, a następnie wykonaniu optymalizacji lokalnej. Jeśli otrzymane w ten sposób rozwiązanie jest innym lokalnym optimum znajdującym się w zbiorze wierzchołków, to dodawana jest odpowiednia krawędź. Metoda ta po raz pierwszy została zaprezentowana w pracy[4] do analizy problemu kwadratowego przypisania. W pracy[1] algorytm oparty o tę samą metodę został wykorzystany do badania przestrzeni problemu komiwojażera. Zastosowana tam odmiana wykorzystuje algorytm 2-opt do optymalizacji lokalnej i procedurę *2-exchange* jako operację perturbacji.

Metoda *Markov-chain* została po raz pierwszy zaprezentowana w pracy [6]. Zaczyna się od wybrania losowego rozwiązania i jego optymalizacji. Następnie otrzymane optimum lokalne zostaje poddane operacji perturbacji i otrzymywane w ten sposób rozwiązanie staje się nowym punktem startowym. Procedura jest powtarzana przez określoną liczbę iteracji, z każdą iteracją zapisywane są informacje o nowych krawędziach i wierzchołkach. Metoda została wykorzystana w pracy[7] do badania przestrzeni problemu komiwojażera. Do optymalizacji lokalnej wykorzystany został algorytm Lin-Kerninghan, a jako operacja perturbacji procedura *Double-bridge*.

Metoda *Snowball* składa się z dwóch etapów wykonywanych naprzemiennie. Pierwszy polega na kilkukrotnym poddaniu rozwiązania początkowego perturbacji w celu uzyskania sąsiednich rozwiązań, a następnie ich optymalizacji. Procedura jest rekurencyjnie powtarzana dla znalezionych w ten sposób lokalnych optimów aż do osiągnięcia pewnej z góry ustalonej głębokości. Jedno z lokalnych optimów sąsiadujących z rozwiązaniem początkowym zostaje wybrane jako rozwiązanie początkowe kolejnej iteracji. Dodatkowo w pamięci przechowywana jest lista rozwiązań początkowych - jeśli rozwiązanie już się w nim znajduje, to nie może być wybrane ponownie. Jeśli nie istnieje rozwiązanie sąsiednie spełniające ten warunek, za kolejne rozwiązanie początkowe przyjmowane jest lokalne optimum otrzymane z optymalizacji rozwiązania losowego. Algorytm *Snowball* wywodzi się z technik wykorzystywanych w badaniach z dziedziny socjologii, a w kontekście badania przestrzeni rozwiązań problemów optymalizacyjnych został zaprezentowany po raz pierwszy w pracy [14].

W pracy[11] zostało przeprowadzone statystyczne porównanie algorytmów próbkowania *Snowball* oraz *Markov-chain*. Eksperyment polegał na spróbkowaniu 30 instancji problemu kwadratowego przypisania, a następnie użyciu zebranych danych do stworzenia modeli regresji przewidujących jakość rozwiązań, które zostaną uzyskane przez algorytm

optymalizacji uruchomiony na instancji. Wybranymi algorytmami heurystycznymi były *Robust Taboo Search* Taillarda oraz *Improved ILS* Stützla. Z zebranych danych utworzono grafy LON i zbadano takie właściwości, jak średnia wartość funkcji dostosowania celu, średni stopień wychodzący wierzchołków w grafie, promień grafu, liczba lokalnych optimów i liczba krawędzi. Stworzono modele regresji typu liniowego oraz lasu losowego. Obliczono wartości korelacji pomiędzy właściwościami przestrzeni rozwiązań a jakością rozwiązań zwracanych przez algorytmy optymalizacji. Badania wykazały, że dane pozyskane z próbkowania *Markov-chain* były w większym stopniu skorelowane z jakością rozwiązań algorytmów optymalizacji, a modele regresji utworzone na ich podstawie dokonywały lepszej predykcji niż te utworzone na podstawie danych ze *Snowball*. Z kolei *Snowball* okazał się bardziej przewidywalny i łatwiejszy w doborze parametrów.

Podobne badanie przeprowadzono w pracy[12], tym razem na większej liczbie instancji - 124 instancji ze zbioru QAPLIB i dodatkowych 60 instancji o rozmiarze N=11. Dla tych dodatkowych instancji, oprócz próbkowania wykonano również przegląd zupełny. Zauważono, że algorytm *Snowball* produkuje gęstszą sieć od algorytmu *Markov-chain*. Dostrzeżono wadę algorytmu *Snowball* - duży wpływ parametrów próbkowania na wartości metryk opartych o gęstość i wzory połączeń krawędzi, oraz metryk opisujących struktury lejowe.

TODO: dokończyć

Z innych prac z dziedziny warto wymienić[9], w którym porównano algorytmy wstępujące (hillclimb) typu *best-improvement* i *first-improvement* w próbkowaniu przestrzeni NK. Wykorzystanie algorytmu typu *first-improvement* prowadziło do powstania gęstszej, bardziej kompletnej sieci optimów lokalnych. Pętle obecne w sieci miały mniejszą wagę w przypadku algorytmu *first-improvement* co sugeruje, że łatwiej wychodzi z optimum lokalnego. Dodatkową zaletą tego algorytmu jest krótszy czas wykonywania, spowodowany brakiem konieczności przeglądania za każdym razem całego sąsiedztwa rozwiązania początkowego.

Z kolei w pracy[5] zbadano wpływ siły perturbacji (stopnia, w jakim operacja perturbacji zmienia rozwiązanie początkowe) na właściwości spróbkowanej przestrzeni. Wybranym algorytmem próbkującym był algorytm typu *Markov-chain* pochodzący z artykułu[7]. Operacją perturbacji tego algorytmu jest procedura double-bridge. Badania przeprowadzono na 180 instancjach o znanych rozwiązaniach optymalnych, uzyskanych przy pomocy generatora DIMACS TSP . Celem było znalezienie takiego parametru k - siły perturbacji - aby uzyskać jak największy wskaźnik sukcesu. Za wskaźnik sukcesu przyjęto stosunek liczby przebiegów algorytmu, które znalazły globalne optimum do wszystkich przebiegów. Nie znaleziono uniwersalnej najlepszej wartości parametru k. Zauważono jednak, że niższa siła perturbacji sprawdzała się dla instancji, w których miasta rozmieszczone są w klastrach. Dla instancji z miastami rozmieszczonymi równomiernie zwiększanie wartości k powodowało zmniejszenie ilości suboptymalnych lejów w przestrzeni. Zaobserwowano również, że instancje z miastami ułożonymi w klastrach były generalnie prostsze do rozwiązania, niż instancje z rozkładem równomiernym, a także, że rozkład optimów lokalnych w przestrzeni rozwiązań ma większy wpływ na trudność znalezienia optymalnego rozwiązania dla danej instancji, niż ich ilość.

2.3 O przestrzeni rozwiązań

Fitness landscape(Przestrzeń rozwiązań? Krajobraz dopasowania?) jest pojęciem wywodzącym się z biologii ewolucyjnej. W kontekście biologicznym jest to model opisujący

relację między genotypem i fenotypem organizmów, a ich przystosowaniem (ang. fitness), które jest miarą opisującą sukces reprodukcyjny[3].

W kontekście optymalizacji Fitness landscape to trójka (S, V, f), gdzie:

- S jest zbiorem wszystkich możliwych rozwiązań - przestrzenią przeszukiwania,
- V jest funkcją przypisującą każdemu rozwiązaniu $s \in S$ zbiór sąsiadów $V(s)$,
- f jest funkcją $f : S \to \mathbb{R}$ przypisującą danemu rozwiązaniu wartość przystosowania - zwykle jest to wartość funkcji celu dla danego rozwiązania.

2.3.1 Sieć optimów lokalnych

Sieć optimów lokalnych(ang. Local Optima Network, LON) jest konstruktem zaprezentowanym po raz pierwszy w artykule[13], i rozwiniętym w pracy[8].

Jest to graf $G = (N, E)$ przedstawiający występujące w przestrzeni rozwiązań optima lokalne (zbiór wierzchołków N) i relacje między nimi (zbiór krawędzi E).

2.3.2 Wierzchołki

Wierzchołki w sieci optimów lokalnych reprezentują optima lokalne w przestrzeni rozwiązań. Do optimów zaliczamy minima i maksima; w problemach optymalizacyjnych zazwyczaj poszukujemy tych pierwszych. Minimum lokalne to takie rozwiązanie s, w którego sąsiedztwie $V(s)$ nie znajduje się żadne rozwiązanie x, dla którego $f(x) < f(s)$. Każde rozwiązanie w przestrzeni rozwiązań można przyporządkować do pewnego lokalnego minimum. Aby znaleźć lokalne minimum $n \in N$, do którego "prowadzi" dane rozwiązanie $s \in S$, wykonuje się lokalną optymalizację z tym rozwiązaniem przyjętym jako punkt startowy. W dalszej części pracy takie przyporządkowanie będzie oznaczane jako $h(s) \to n \in N$. Wierzchołkom w sieci LON można przypisać wagę równą wartości funkcji celu w danym optimum lokalnym.

2.3.3 Krawędzie

Krawędzie w sieci lokalnych optimów mogą być zdefiniowane na jeden z kilku sposobów. W literaturze[8][10] zostały opisane trzy różne modele: *basin-transition*, *escape edges* i *perturbation edges*.

Basin-transition

Basen przyciągania optimum lokalnego n jest zdefiniowany jako zbiór:

$$b_i = \{s \in S \mid h(s) = n\}$$

Rozmiarem basenu jest liczność tego zbioru oznaczana jako $|b_i|$. Dla każdej pary rozwiązań w przestrzeni można obliczyć prawdopodobieństwo przejścia z jednego rozwiązania do drugiego $p(s \to s')$. Dla rozwiązań reprezentowanych permutacją o długości M, prawdopodobieństwo takie wynosi:

$$p(s \to s') = \frac{1}{M(M-1)/2}, \quad \text{jeżeli} \quad s' \in V(s),$$

$$p(s \to s') = 0, \quad \text{jeżeli} \quad s' \notin V(s),$$

Mając informacje o prawdopodobieństwach przejścia między poszczególnymi rozwiązaniami można obliczyć prawdopodobieństwo przejścia od rozwiązania s do dowolnego rozwiązania należącego do basenu b_j:

$$p(s \to b_j) = \sum_{s' \in b_j} p(s \to s')$$

Całkowite prawdopodobieństwo przejścia z basenu jednego optimum lokalnego do drugiego wynosi więc:

$$p(b_i \to b_j) = \frac{1}{|b_i|} \cdot \sum_{s \in b_i} p(s \to b_j)$$

To całkowite prawdopodobieństwo stanowi wagę krawędzi w grafie.

Krawędzie typu *basin-transition* tworzą gęstszą sieć od krawędzi typu *escape edges*.

Escape Edges

Escape Edges zdefiniowane są przy pomocy funkcji dystansu d zwracającej najmniejszą odległość między dwoma rozwiązaniami, oraz liczby całkowitej D. Krawędź e_{ij} między lokalnymi optimami n_i i n_j istnieje, jeśli istnieje rozwiązanie s takie, że:

$$d(s, n_i) \leqslant D \wedge h(s) = n_j \tag{2.1}$$

Wagą takiej krawędzi jest ilość rozwiązań spełniających powyższy warunek.

Perturbation edges

W tym modelu wagę krawędzi pomiędzy lokalnymi optimami n_i i n_j uzyskuje się poprzez kilkukrotne wykonanie operacji perturbacji(mutacji) na n_i, a następnie optymalizacji lokalnej otrzymanego rozwiązania. Liczba przypadków, w których po optymalizacji otrzymujemy rozwiązanie n_j podzielona przez liczbę prób stanowi wagę krawędzi.

$$w_{ij} = \frac{|\{opt(mut(n_i)) = n_j\}|}{trials}$$

TODO: czy zapis jest poprawny $\wedge$?

2.3.4 Struktury lejowe

W sieci optimów lokalnych możemy wyróżnić sekwencje złożone z optimów lokalnych, których wartość dopasowania jest niemalejąca. Sekwencje te zwane są sekwencjami monotonicznymi[7]. Sekwencje monotoniczne zmierzające do tego samego ścieku (ang. sink, wierzchołek bez krawędzi wychodzących) tworzy strukturę zwaną lejem. Wspomniany ściek stanowi spód leja (ang. funnel bottom), a liczba wierzchołków zawartych w tej strukturze określa jej rozmiar. Ponadto w przestrzeni rozwiązań można wyróżnić lej pierwszorzędny(ang. primary funnel), kończący się w globalnym optimum i leje drugorzędne, kończące się w optimach lokalnych. Jeden wierzchołek w grafie może przynależeć jednocześnie do wielu lejów.

Leje w sieci optimów lokalnych można zidentyfikować poprzez usunięcie z grafu krawędzi prowadzących od lepszych rozwiązań do gorszych, zidentyfikowanie ścieków, odwrócenie krawędzi i wykonanie przeszukiwania wgłąb lub wszerz w celu odnalezienia wierzchołków należących do leja.

2.3.5 Metryki przestrzeni rozwiązań

- **num_sinks** - liczba ścieków. Ściek (ang. sink) jest wierzchołkiem grafu nie posiadającym krawędzi wychodzących. Pętle nie są uwzględniane.
- **num_sources** - liczba źródeł. Źródło (ang. source) jest wierzchołkiem grafu nie posiadającym krawędzi wchodzących. Pętle nie są uwzględniane.
- **num_subsinks** - Liczba wierzchołków, które nie posiadają krawędzi wychodzących do rozwiązań o niższej wartości funkcji celu.
- **edge_to_node** - stosunek liczby krawędzi do liczby wierzchołków w grafie.
- **avg_fitness** - średnia wartość funkcji celu lokalnych optimów w sieci.
- **distLO** - średnia odległość rozwiązań od rozwiązania z najniższą wartością funkcji celu. Odległość jest zdefiniowana jako odwrotność wagi krawędzi łączących rozwiązania. Rozwiązania nie połączone krawędzią z najlepszym rozwiązaniem nie są brane pod uwagę.
- **conrel** - Stosunek liczby rozwiązań połączonych krawędzią z najlepszym rozwiązaniem do liczby pozostałych rozwiązań.
- **avg_out_degree, max_out_degree** - średni i maksymalny stopień wychodzący rozwiązań w grafie. Stopień wychodzący wierzchołka to liczba wychodzących z niego krawędzi. Pętle nie są brane pod uwagę.
- **avg_in_degree, max_in_degree** - średni i maksymalny stopień wchodzący rozwiązań w grafie. Stopień wchodzący wierzchołka to liczba wchodzących do niego krawędzi. Pętle nie są brane pod uwagę.
- **assortativity** - współczynnik różnorodności grafu. Różnorodność (ang. assortativity) grafu skierowanego jest zdefiniowana następującym wzorem:

$$assortativity = \frac{1}{\sigma_o \sigma_i} \sum_{(j,k) \in E} deg(j) \cdot deg(k) \cdot (e_{jk} - q_j^o q_k^i)$$

  Gdzie:

  - e_{ij} - część krawędzi łączących wierzchołki i i j w stosunku do liczby wszystkich krawędzi (ułamek z zakresu 0 do 1),
  - $q_i^o = \sum_{j \in V} e_{ij}$
  - $q_i^i = \sum_{j \in V} e_{ji}$
  - σ_o - odchylenie standardowe q^o
  - σ_i - odchylenie standardowe q^i

- **clustering_coeff** - współczynnik klasteryzacji grafu (ang. clustering coefficient, transitivity) opisuje prawdopodobieństwo istnienia połączenia pomiędzy sąsiednimi wierzchołkami. Współczynnik klasteryzacji opisany jest wzorem:

$$cc = \frac{N_{triangles}}{N_{ctriples}}$$

Gdzie $N_{triangles}$ to liczba trójkątów w grafie, a $N_{ctriples}$ to liczba połączonych trójek.

Trójkąt to trójka wierzchołków (x, y, z) taka, że $(x, y), (y, z), (x, z) \in E$.

Połączona trójka to trójka wierzchołków (x, y, z) taka, że $(x, y), (y, z) \in E$.

- **density** - gęstość grafu. Jest to stosunek liczby krawędzi w grafie do maksymalnej liczby krawędzi, jaka mogłaby istnieć w tym grafie. dana jest wzorem:

$$density = \frac{|E|}{|V|(|V| - 1)}$$

- **cliques_num** - Liczba klik. Klika w grafie jest podzbiorem zbioru wierzchołków, w którym wszystkie wierzchołki są sąsiednie - istnieje krawędź pomiędzy każdą parą wierzchołków należących do zbioru.

- **maximal_cliques_num** - Liczba klik maksymalnych w grafie. Klika maksymalna to klika, której nie można powiększyć poprzez dołączenie do niej sąsiedniego wierzchołka.

- **largest_clique_size** - rozmiar największej kliki.

- **reciprocity** - Wzajemność. Wzajemność jest miarą zdefiniowaną tylko dla grafów skierowanych. Jest to stosunek wierzchołków wzajemnie połączonych do wierzchołków, które są połączone krawędzią tylko w jednym kierunku. Dana jest wzorem:

$$reciprocity = \frac{|(i, j) \in E \mid (j, i) \in E|}{|(i, j) \in E \mid (j, i) \notin E|}$$

- **funnel_num** - liczba lejów w przestrzeni rozwiązań

- **mean_funnel_size** - średnia wielkość leja

- **max_funnel_size** - wielkość największego leja

- **go_path_ratio** - stosunek liczby wierzchołków ze ścieżką do najlepszego rozwiązania do liczby wszystkich wierzchołków.

- **avg_go_path_len** - średnia długość ścieżki do najlepszego rozwiązania. Wierzchołki bez ścieżki do najlepszego rozwiązania nie są brane pod uwagę Długość ścieżki definiowana jest jako liczba krawędzi wchodzących w skład ścieżki pomiędzy wierzchołkami.

- **max_go_path_len** - długość najdłuższej ścieżki do najlepszego rozwiązania.

- **num_cc** - Liczba spójnych podgrafów grafu

- **largest_cc** - Wielkość (liczba wierzchołków) największego spójnego podgrafu.

- **largest_cc_radius** - Promień największego spójnego podgrafu. Promień grafu to najmniejsza acentryczność wierzchołka wśród wszystkich wierzchołków grafu. Acentryczność(ang. eccentricity) wierzchołka to największa z odległości wierzchołka do innych wierzchołków grafu.

2.3.6 Problem komiwojażera

Problem komiwojażera(ang. Traveling Salesman Problem, TSP) jest znanym problemem optymalizacyjnym sformułowanym w następujący sposób: mając do dyspozycji listę miast i odległości między nimi, należy odnaleźć najkrótszą ścieżkę przechodzącą przez wszystkie miasta zaczynającą i kończącą się w ustalonym punkcie. Problem ten jest problemem NP-trudnym i z tego powodu do rozwiązywania większych jego instancji konieczne jest stosowanie algorytmów heurystycznych.

2.3.7 Operacja 2-exchange

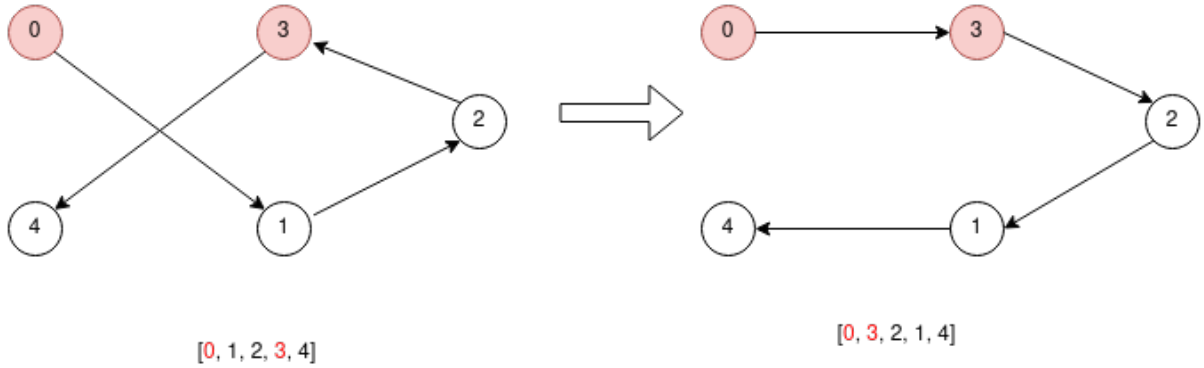
Operacja 2-exchange jest operacją wybrania dwóch wierzchołków i zamiany krawędzi prowadzących do następnych wierzchołków. Procedura jest wykorzystywana w algorytmie heurystycznym 2opt, w którym w ten sposób "rozplatane" są skrzyżowane krawędzie.

Algorytmy przeszukiwania przestrzeni rozwiązań zaprezentowane w tej pracy wykorzystują operację 2-exchange jako operację mutacji. Mutacja permutacji polega na D-krotnym wykonaniu operacji 2-exchange na losowych wierzchołkach, gdzie D to maksymalna odległość z równania 2.1.

Operacja 2-exchange oraz operator mutacji zostały przedstawione na listingu 1.

Algorithm 1: Operacja 2exchange - pseudokod

```
function 2exchange(a, b, perm):
    a ← a + 1;
    while a < b do
        zamien(perm[a], perm[b]);
        a ← a + 1;
        b ← b + 1;
    end
    return perm;
end

function 2exchangeMutacja(perm, D):
    for i ← 1 to D do
        a ← losowaZZakresu(0, length(perm) − 3);
        b ← losowaZZakresu(a + 2, length(perm) − 1);
        perm ← 2exchange(a, b, perm);
    end
    return perm;
end

function 2exchangeWszystkiePermutacje(perm):
    perms = {};
    for a ← 0 to n − 3 do
        for b ← a + 2 to n − 1 do
            perm ← 2exchange(a, b);
            perms ← perms ∪ {perm};
        end
    end
    return perms;
end
```



Rysunek 2.1 Przykład procedury 2-exchange

Rozdział 3

Badania eksperymentalne

3.1 Opis eksperymentów

3.2 Zaimplementowane algorytmy

3.2.1 Próbkowanie dwufazowe

Próbkowanie dwufazowe swoją nazwę zawdzięcza procesowi próbkowania składającemu się z dwóch oddzielnych faz - próbkowania wierzchołków oraz próbkowania krawędzi - wykonywanych jedna po drugiej. Istotną zaletą tego podejścia jest jego stosunkowo prosta implementacja.

Zaimplementowany algorytm pochodzi z pracy[1]. Został on przygotowany specjalnie do próbkowania przestrzeni rozwiązań problemu komiwojażera. Próbkowanie wierzchołków odbywa się poprzez generowanie losowych rozwiązań, a następnie ich optymalizacji algorytmem 2-opt. Próbkowanie krawędzi polega na wielokrotnym poddaniu każdego ze znalezionych wcześniej lokalnych optimów n_i operacji perturbacji typu 2-exchange, a następnie poddaniu powstałego rozwiązania optymalizacji algorytmem 2-opt typu *first-improvement* uzyskując w ten sposób lokalne optimum n_j. Następnie dodawana jest krawędź między n_i a n_j, lub - jeśli już taka istnieje - jej waga jest zwiększana o 1.

Algorytm przyjmuje trzy parametry: pożądaną liczbę wierzchołków do wygenerowania (n_{max}), maksymalną liczbę prób generowania wierzchołka (n_{att}) oraz maksymalną liczbę prób generowania krawędzi(e_{att}). Implementacja zastosowana w tej pracy dodatkowo powtarza cały proces kilkukrotnie, za każdym razem zapisując zebrane próbki do pliku.

Algorytm w postaci pseudokodu został przedstawiony na listingu 2.

3.2.2 Snowball

Próbkowanie typu Snowball wywodzi się z techniki używanej w badaniach z dziedziny socjologii, w której ludzie należący do próby z populacji rekrutują kolejnych uczestników badania spośród swoich znajomych. W kontekście badania przestrzeni rozwiązań technika ta została zaprezentowana w pracy[14], gdzie została wykorzystana do próbkowanie przestrzeni problemu kwadratowego przypisania (QAP).

Próbkowanie składa się z etapów procedury *snowball* próbkującej "wgłąb" i losowego spaceru(ang. *random walk*). Próbkowanie *snowball* polega na wybraniu rozwiązania startowego i przeszukaniu jego najbliższego sąsiedztwa. Następnie operacja ta jest powtarzana dla każdego rozwiązania w tym sąsiedztwie. Proces powtarza się aż do osiągnięcia z góry ustalonej głębokości przeszukiwania. Następnie rozpoczyna się procedura losowego

spaceru - wybierane jest kolejne rozwiązanie startowe ze zbioru sąsiadów poprzedniego rozwiązywania startowego (lub rozwiązanie losowe, jeśli to sąsiedztwo jest puste) i proces *snowball* rozpoczyna się od nowa. Procedura jest powtarzana aż osiągnięty zostanie z góry ustalony limit długości spaceru.

Zaimplementowany algorytm jest próbą adaptacji tej techniki do zadania przeszukiwania przestrzeni problemu komiwojażera. Do najważniejszych modyfikacji należy zastąpienie funkcji optymalizacji lokalnej *hillclimb* optymalizacją 2opt, implementacja odpowiedniej funkcji celu oraz operacji mutacji typu 2-exchange.

Algorytm w postaci pseudokodu został przedstawiony na listingu 3.

Algorithm 2: Próbkowanie dwufazowe - pseudokod

```
Data:
    n_max - żądana liczba wierzchołków
    n_att - liczba prób generowania wierzchołków
    e_att - liczba prób generowania krawędzi
    n_runs - liczba powtórzeń
    D - stała D krawędzi

N ← {};
E ← {};
for i ← 1 to n_runs do
    probkujWierzcholki(N, n_max, n_att);
    probkujKrawedzie(N, E, e_att);
    zapiszDoPliku(N, E);
end

function probkujWierzcholki(N, n_max, n_att):
    for i ← 1 to n_max do
        for i ← 1 to n_att do
            s ← losoweRozwiazanie();
            s ← 2opt(s);
            N ← N ∪ {s};
        end
    end
end

function probkujKrawedzie(N, E, e_att):
    foreach n ∈ N do
        for i ← 1 to e_att do
            s ← 2exchangeMutacja(n, D);
            s ← 2optFirstImprovement(s);
            if s ∈ N then
                E ← E ∪ {(n, s)};
                w_ns ← w_ns + 1;
            end
        end
    end
end
```

Algorithm 3: Próbkowanie snowball - pseudokod

Data:
w_{len} - długość losowego spaceru
m - liczba prób przeszukania sąsiedztwa
$depth$ - głębokość przeszukiwania
n_{runs} - liczba powtórzeń
D - stała D krawędzi

$s_1 \leftarrow losoweRozwiazanie()$;
$n_1 \leftarrow 2opt(s_1)$;
$N \leftarrow \{n_1\}$;
$E \leftarrow \{\}$;
for $j \leftarrow 1$ **to** n_{runs} **do**
  for $i \leftarrow 1$ **to** w_{len} **do**
    $snowball(n_i, m, depth)$;
    $n_{i+1} \leftarrow losowySpacer(n_i)$;
  end
  $zapiszDoPliku(N, E)$;
end

function `snowball`$(n, m, depth)$**:**
  if $d > 0$ **then**
    for $i \leftarrow 1$ **to** m **do**
      $s \leftarrow 2opt(2exchangeMutacja(n, D))$;
      $N \leftarrow N \cup \{s\}$;
      if $(n, s) \in E$ **then**
        $w_{ns} \leftarrow w_{ns} + 1$;
      else
        $E \leftarrow E \cup \{(n, s)\}$;
        $w_{ns} \leftarrow 1$;
        $snowball(s, m, d - 1)$;
      end
    end
  end
end

function `losowySpacer`(n_i)**:**
  $neighbours \leftarrow \{s : (n_i, s) \in E \wedge s \notin \{n_0 ... n_i\}\}$;
  if $neighbours \neq \emptyset$ **then**
    $n_{i+1} \leftarrow losowyElementZeZbioru(neighbours)$;
  else
    $s \leftarrow losoweRozwiazanie()$;
    $n_{i+1} \leftarrow 2opt(s)$;
    $N \leftarrow N \cup \{n_{i+1}\}$;
  end
  return n_{i+1}
end

3.2.3 Przegląd zupełny

Ze względu na złożoność problemu komiwojażera przegląd zupełny można zastosować tylko do bardzo małych instancji problemu. Przegląd polega na wygenerowaniu wszystkich możliwych rozwiązań danej instancji, wykonaniu na nich optymalizacji 2-opt w celu znalezienia optimów lokalnych a następnie znalezieniu krawędzi oraz obliczeniu ich wag. Dla każdego z rozwiązań generowane są wszystkie permutacje, które mogą powstać poprzez D-krotne wykonanie na rozwiązaniu operacji 2-exchange. Jeśli wśród tych permutacji znajduje się jedno ze znalezionych wcześniej lokalnych optimów, oznacza to, że spełniony jest warunek 2.1 i dodawana jest nowa krawędź lub zwiększona zostaje waga istniejącej.

Algorithm 4: Przegląd zupełny

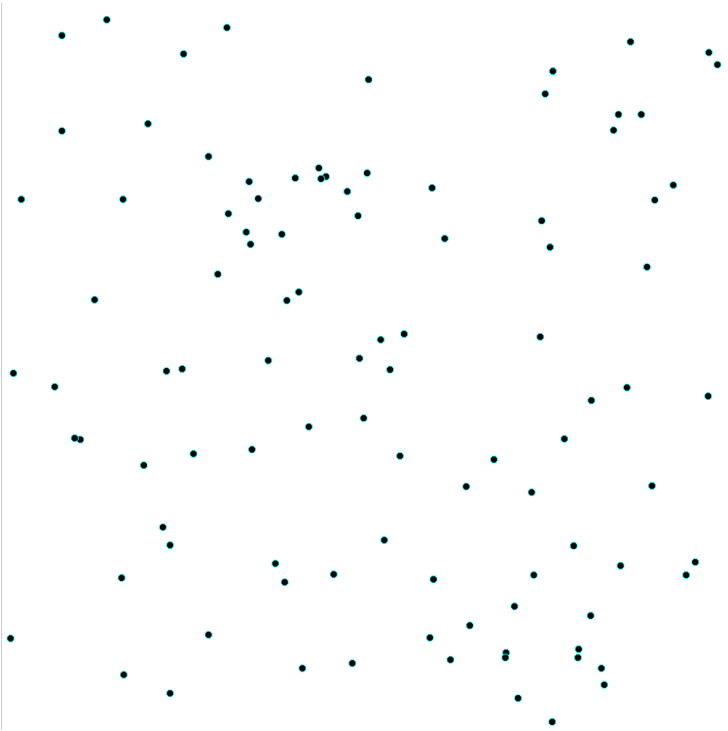
```
S ← {};
P ← wygenerujWszystkiePermutacje();
foreach p ∈ P do
    lo ← 2opt(p);
    S ← S ∪ {(p, lo)};
end
foreach (p, lo) ∈ S do
    foreach n ∈ N do
        if wZasiegu2Exchange(p, n, D) then
            if (n, lo) ∈ E then
                w_{n,lo} ← w_{n,lo} + 1;
            else
                E ← E ∪ {(n, lo)};
                w_{n,lo} = 1;
            end
        end
    end
end

function wZasiegu2Exchange(p, n, D):
    permutacje ← {p};
    for i ∈ 1..D do
        nowe_perm ← {};
        foreach perm ∈ permutacje do
            pochodne_perm ← 2exchangeWszystkiePermutacje(permutacje);
            foreach poch ∈ pochodne_perm do
                if poch = n then
                    return true;
                end
                nowe_perm ← nowe_perm ∪ {poch};
            end
        end
        permutacje ← nowe_perm;
    end
    return false;
end
```

3.3 Instancje testowe

Do badań wykorzystano instancje testowe wygenerowane losowo oraz wybrane instancje ze zbioru TSPLIB. Zaimplementowano trzy generatory tworzące różne typy instancji testowych:

Miasta rozmieszczone równomiernie

Generator losowo rozmieszcza miasta na wirtualnej planszy o ustalonym rozmiarze. Współrzędne miast generowane są losowo z rozkładu równomiernego. Przykład wygenerowanej instancji został przedstawiony na rysunku 3.1.



Rysunek 3.1 Wizualizacja wygenerowanej instancji z miastami rozmieszczonymi równomiernie dla N=100

Miasta rozmieszczone w klikach

Miasta umieszczane są blisko siebie w kilku grupach oddzielonych większymi odległościami. Przykład wygenerowanej instancji został przedstawiony na rysunku 3.2.

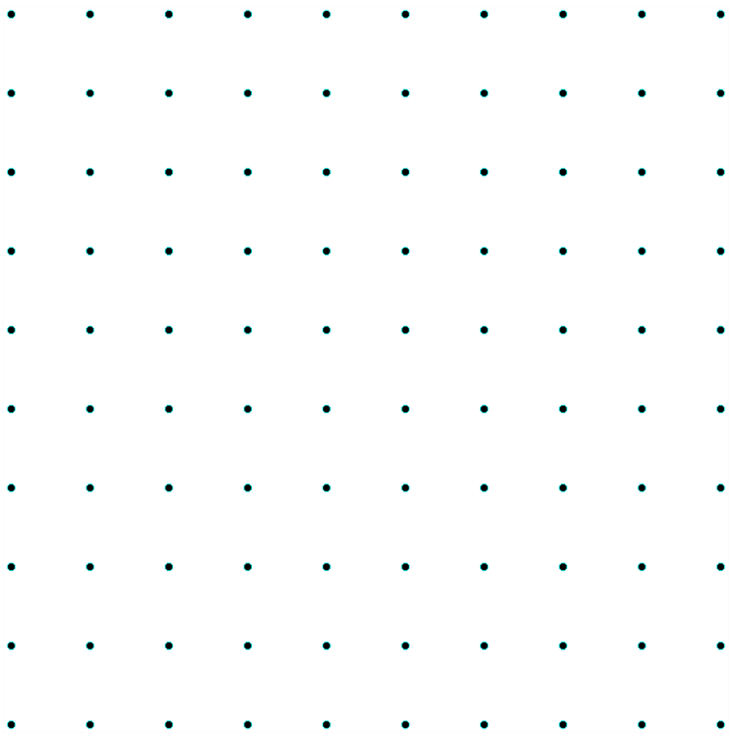
Miasta rozmieszczone na siatce

Miasta umieszczane są na siatce, w stałej odległości od swoich sąsiadów. Przykład wygenerowanej instancji został przedstawiony na rysunku 3.3.

Wygenerowano instancje testowe każdego z trzech typów instancji losowych o rozmiarach 7,8,9,10,11 oraz 20,50,100,200 i 500. Uzyskano w ten sposób 30 instancji problemu.

Ze zbioru TSPLIB wybrano instancje o podobnych rozmiarach: **burma14**, **ulysses22**, **att58**, **berlin52**, **rat99**, **bier127**, **d198**, **a280**, **pa561**, **u574**.

Rysunek 3.2 Wizualizacja wygenerowanej instancji z miastami rozmieszczonymi w kilkach dla N=100



Rysunek 3.3 Wizualizacja wygenerowanej instancji z miastami rozmieszczonymi na siatce dla N=100

3.4 Experimental setup but po polsku

3.5 Wyniki

Rozdział 4

Opis implementacji

Rozdział 5

Podsumowanie

Literatura

[1] W. Bozejko, A. Gnatowski, T. Nizynski, M. Affenzeller, and A. Beham, *Local optima networks in solving algorithm selection problem for TSP*, in Contemporary Complex Systems and Their Dependability - Proceedings of the 13th International Conference on Dependability and Complex Systems DepCoS-RELCOMEX, July 2-6, 2018, Brunów, Poland, W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, and J. Kacprzyk, eds., vol. 761 of Advances in Intelligent Systems and Computing, Springer, 2018, pp. 83–93.

[2] C. W. Cleghorn and G. Ochoa, *Understanding parameter spaces using local optima networks: a case study on particle swarm optimization*, in GECCO '21: Genetic and Evolutionary Computation Conference, Companion Volume, Lille, France, July 10-14, 2021, K. Krawiec, ed., ACM, 2021, pp. 1657–1664.

[3] I. Fragata, A. Blanckaert, M. A. Dias Louro, D. A. Liberles, and C. Bank, *Evolution in the light of fitness landscape theory*, Trends in Ecology & Evolution, 34 (2019), pp. 69–82.

[4] D. Iclanzan, F. Daolio, and M. Tomassini, *Data-driven local optima network characterization of qaplib instances*, in Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO '14, New York, NY, USA, 2014, Association for Computing Machinery, p. 453–460.

[5] P. McMenemy, N. Veerapen, and G. Ochoa, *How perturbation strength shapes the global structure of TSP fitness landscapes*, in Evolutionary Computation in Combinatorial Optimization - 18th European Conference, EvoCOP 2018, Parma, Italy, April 4-6, 2018, Proceedings, A. Liefooghe and M. López-Ibáñez, eds., vol. 10782 of Lecture Notes in Computer Science, Springer, 2018, pp. 34–49.

[6] G. Ochoa and S. Herrmann, *Perturbation Strength and the Global Structure of QAP Fitness Landscapes: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part II*, 01 2018, pp. 245–256.

[7] G. Ochoa and N. Veerapen, *Mapping the global structure of TSP fitness landscapes*, J. Heuristics, 24 (2018), pp. 265–294.

[8] G. Ochoa, S. Vérel, F. Daolio, and M. Tomassini, *Local optima networks: A new model of combinatorial fitness landscapes*, CoRR, abs/1402.2959 (2014).

[9] G. Ochoa, S. Vérel, and M. Tomassini, *First-improvement vs. best-improvement local optima networks of NK landscapes*, CoRR, abs/1207.4455 (2012).

[10] M. C. Teixeira and G. L. Pappa, *Understanding automl search spaces with local optima networks*, in GECCO '22: Genetic and Evolutionary Computation Conference,

Boston, Massachusetts, USA, July 9 - 13, 2022, J. E. Fieldsend and M. Wagner, eds., ACM, 2022, pp. 449–457.

[11] S. L. Thomson, G. Ochoa, and S. Vérel, *Clarifying the difference in local optima network sampling algorithms*, in Evolutionary Computation in Combinatorial Optimization - 19th European Conference, EvoCOP 2019, Held as Part of EvoStar 2019, Leipzig, Germany, April 24-26, 2019, Proceedings, A. Liefooghe and L. Paquete, eds., vol. 11452 of Lecture Notes in Computer Science, Springer, 2019, pp. 163–178.

[12] S. L. Thomson, G. Ochoa, S. Vérel, and N. Veerapen, *Inferring future landscapes: Sampling the local optima level*, Evol. Comput., 28 (2020), pp. 621–641.

[13] M. Tomassini, S. Vérel, and G. Ochoa, *Complex-network analysis of combinatorial spaces: The nk landscape case*, Phys. Rev. E, 78 (2008), p. 066114.

[14] S. Vérel, F. Daolio, G. Ochoa, and M. Tomassini, *Sampling local optima networks of large combinatorial search spaces: The QAP case*, in Parallel Problem Solving from Nature - PPSN XV - 15th International Conference, Coimbra, Portugal, September 8-12, 2018, Proceedings, Part II, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and L. D. Whitley, eds., vol. 11102 of Lecture Notes in Computer Science, Springer, 2018, pp. 257–268.