Spis treści

1 Wstęp **3**

2 Przegląd literatury **5**
2.1 Przegląd - notatki 5
2.2 O przestrzeni rozwiązań 10
2.2.1 Sieć optimów lokalnych 11
2.2.2 Wierzchołki 11
2.2.3 Krawędzie 11
2.2.4 Struktury lejowe 12
2.2.5 Metryki przestrzeni rozwiązań 13
2.2.6 Problem komiwojażera 15
2.2.7 Operacja 2-exchange 15

3 Badania eksperymentalne **17**
3.1 Opis eksperymentów 17
3.2 Zaimplementowane algorytmy 17
3.2.1 Próbkowanie dwufazowe 17
3.2.2 Snowball 17
3.2.3 Przegląd zupełny 20
3.3 Instancje testowe 21
3.4 Experimental setup but po polsku 23
3.5 Wyniki 23

4 Opis implementacji **25**

5 Podsumowanie **27**

Literatura **27**

Rozdział 1

Wstęp

Rozdział 2

Przegląd literatury

2.1 Przegląd - notatki

First-improvement vs. Best-improvement Local Optima Networks of NK Landscapes (2012)[7]

W pracy porównane zostały algorytmy hillclimb first-improvement i best-improvement w próbkowaniu NK landscape. Porównano sieci optimów lokalnych wygenerowane przez oba algorytmy. Wśród miar można wymienić: Liczbę wierzchołków i krawędzi, współczynnik klasteryzacji, disparity, długość najkrótszej ścieżki między wierzchołkami. Zbierano także informacje o basenach przyciągania: rozmiar basenu globalnego optimum, porównanie rozmiarów basenów w obu sieciach, liczba basenów na rozwiązanie. Badania wykazały, że algorytm typu first-improvement tworzy gęstrzą, bardziej kompletną sieć LON, z mniejszymi wartościami wag i mniejszą liczbą pętli. First improvement tworzył zwykle dłuższe ścieżki w sieci. Dodatkową zaletą first-improvement jest szybsze działanie, gdyż nie musi być przeglądane całe sąsiedztwo.

Mapping the global structure of TSP fitness landscapes (2018) [5]

Celem pracy było zbadanie przestrzeni rozwiązań różnych instancji problemu komiwojażera(TSP), oraz opracowanie procedury próbkowania tych przestrzeni. W pracy przedstawiony został algorytm próbkujący oparty o algorytm *Chained-Lin-Kerninghan*. Do próbkowania wykorzystano implementację tego algorytmu z biblioteki *Concorde*. Algorytm konstruuje sieć optimów lokalnych(LON) - odmianę z definicją krawędzi *escape edges*.

Wykonane zostało 1000 niezależnych przebiegów algorytmu. Połowa z nich korzystała z metody inicjalizacji Quicka-Borůvki, a druga połowa za punkt startowy przyjmowała losowe rozwiązanie. Algorytm opiera się na generowaniu nowych rozwiązań poprzez zastosowanie perturbacji typu double-bridge, a następnie uruchomieniu algorytmu Lin-Kerninghan, by znaleźć lokalne minimum. Operacja jest powtarzana 10000 razy i budowana jest w ten sposób sieć optimów lokalnych.

Jednym z zadań, które postawili sobie autorzy pracy, była identyfikacja struktur lejowych w przestrzeni rozwiązań. Lej jest sekwencją lokalnych optimów, których wartość dopasowania się nie pogarsza, i która zmierza do pewnego optimum - globalnego (lej pierwszorzędny - ang. primary funnel) lub lokalnego (lej drugorzędny - ang. secondary funnel). To optimum nazywane jest spodem leja (ang. funnel floor/bottom).

Sieci optimów lokalnych wygenerowane przez kolejne uruchomienia algorytmu próbkującego zawyżały liczbę lejów w przestrzeni rozwiązań. Badacze scalili więc znalezione

optima lokalne z wszystkich przebiegów algorytmu. Znalezione w ten sposób rozwiązania nazwali atraktorami i utworzyli graf sieci atraktorów, w której są one wierzchołkami. Następnie spody lejów zostały zidentyfikowane jako wierzchołki, z których nie ma krawędzi wychodzących. Przynależność optimów lokalnych do poszczególnych lejów odbywała się poprzez sprawdzenie, czy istnieje ścieżka w grafie pomiędzy danym optimum a spodem leja.

Do badań wykorzystano 20 instancji TSP o rozmiarach od 500 do 1500 miast. Wykorzystano instancje o miastach rozłożonych równomiernie i w klastrach. Część została wygenerowana losowo, a część pochodziła z biblioteki TSPLIB.

Zbadano właściwości powstałych sieci optimów lokalnych, m.in. liczbę unikalnych optimów lokalnych i globalnych, liczba wywołań funkcji celu, stosunek liczby wierzchołków i wywołań funkcji celu, stosunek wierzchołków jest połączonych ścieżką z optimum globalnym do całkowitej liczby wierzchołków, średnią siłę(ang. strength) wierzchołków. Zbadano również własności lejów, takie jak rozmiar największego leja czy liczba atraktorów.

Badania wykazały, że przestrzeń rozwiązań badanych instancji TSP zawiera wiele lejów a nie, jak wcześniej sądzono, jeden lej kończący się w globalnym minimum. W przestrzeni rozwiązań instancji o regularnej strukturze wykryto istnienie rozległych płaskowyżów(ang. plateaus). Zauważono różnicę między instancjami wygenerowanymi losowo a rzeczywistymi. Te pierwsze miały zwykle jedno globalne optimum i małą neutralność, podczas gdy instancje rzeczywiste zwykle miały więcej globalnych optimów, czasem formujących płaskowyż globalnych optimów. Zauważono zależność między rozmiarem instancji a liczbą lejów - ich liczba zwiększa się wraz ze wzrostem rozmiaru instancji. Najwięcej lejów pojawiło się w instancjach z równomiernym losowym rozmieszczeniem miast. Zbadano korelację między różnymi właściwościami przestrzeni rozwiązań. Wykazano, że w instancjach, w których miasta rozmieszczone są w klastrach, występuje większa korelacja właściwości, niż w instancjach o rozłożeniu równomiernym.

Local Optima Networks in Solving Algorithm Selection Problem for TSP (2018) [1]

W pracy badano możliwość wykorzystania analizy sieci optimów lokalnych w problemie wyboru algorytmu heurystycznego dla danej instancji problemu komiwojażera (Algorithm selection problem).

Wykorzystano algorytm dwufazowy (ang. Two-Phase Sampling). Algorytm tego typu podzielony jest na dwa etapy - poszukiwanie wierzchołków i poszukiwanie krawędzi. Wierzchołki wyszukiwane są poprzez generowanie losowych rozwiązań i uruchamianiu na nich algorytmu 2-opt typu best-improvement. Algorytm generuje LON opartą o escape edges. Próbkowanie krawędzi polega na poddaniu każdego ze znalezionych wcześniej lokalnych optimów perturbacji typu 2-change i optymalizacji algorytmem 2-opt typu first-improvement. Sprawdzany jest następnie warunek istnienia krawędzi i, jeśli jest spełniony, do zbioru dodawana jest odpowiednia krawędź bądź zwiększana jest jej waga, jeśli już istnieje ona w zbiorze. Wśród badanych parametrów wymienić można m.in stosunek liczby krawędzi do liczby wierzchołków, średnia odległość optimum lokalnego od optimum o najniższej wartości funkcji celu, stosunek wierzchołków połączonych z tym optimum do pozostałych oraz różnorodność(ang. assortativity) i współczynnik klasteryzacji grafu. Do eksperymentów wykorzystano instancje problemu z biblioteki TSPLIB, oraz instancje generowane losowo o równomiernym rozkładzie miast. Obliczono korelację różnych miar, aby wyeliminować redundantne dane.

Na każdej instancji problemu uruchomiono algorytmy heurystyczne i porównano parami jakość ich rozwiązań. Obliczono również średnią względnej jakości rozwiązań dla każdego algorytmu. Porównane zostały algorytmy *Greedy Descent*, *Guided Local Search*, *Simulated Annealing*, *Taboo Search*, *Objective Taboo Search*. Wykorzystano implementacje zawarte w pakiecie *Google Optimization Tools*. Sprawdzono również obecny w pakiecie tryb automatyczny, który sam dobiera algorytm dla danej instancji.

Na podstawie zebranych danych nauczono klasyfikatory dla każdej pary algorytmów. Klasyfikatory przypisywały wyniki do jednej z trzech klas: 1 - algorytm A daje lepszy wynik od algorytmu B, 2 - algorytm B daje lepszy wynik od algorytmu A, 3 - algorytmy dają ten sam wynik. Utworzone klasyfikatory porównano z klasyfikatorem ZeroR. Przygotowane klasyfikatory dokonały poprawnej predykcji częściej niż ZeroR w przypadku instancji ze zbioru TSPLIb. W przypadku instancji generowanych losowo, tylko w jednym przypadku klasyfikator okazał się lepszy od ZeroR.

Eksperymenty wykazały, że analiza sieci optimów lokalnych może zostać z powodzeniem wykorzystana do selekcji algorytmu heurystycznego, jednak czas próbkowania, dłuższy od czasu samego rozwiązywania problemu, ogranicza jej użyteczność.

How Perturbation Strength Shapes the Global Structure of TSP Fitness Landscapes[4]

W pracy zbadany został wpływ siły permutacji na wyniki generowane przez algorytm próbkowania Chained-LK. Do badań wykorzystano dwa rodzaje generowanych instancji - miasta rozłożone równomiernie i w klastrach. Sieć LON była tworzona poprzez agregowanie danych z 1000 niezależnych uruchomień algorytmu. Przebadano siłę perturbacji z zakresu 1 do 10. Zastosowanym typem perturbacji był double-bridge. Wśród badanych metryk znalazły się: success rate - czy globalne optimum znane a priori zostało znalezione?, średnia liczba iteracji Chained-LK potrzebna do znalezienia globalnego optimum, liczba wierzchołków i krawędzi w lon, in-strength wierzchołków z globalnym optimum, część wierzchołków należących do leja globalnego optimum, część lokalnych optimów z unikalnymi wartościami funkcji celu. Celem było takie dobranie siły perturbacji z, aby otrzymać jak największy success rate. Badania wykazały, że dla instancji z miastami ułożonymi w klastrach, lepsze wyniki osiągnięto przy zastosowaniu niskiej siły perturbacji - za duża wartość tworzyła dużo sub-optymalnych lejów w przestrzeni rozwiązań. Z kolei dla instancji z równomiernym rozłożeniem miast było wręcz przeciwnie - mała siła perturbacji tworzyła przestrzeń pełną suboptymalnych lejów, a większa nieco ją wygładzała. Badanie wykazało również, że instancje z miastami w klastrach były generalnie prostsze do rozwiązania niż instancje równomierne, wynika z tego że rozkład optimów lokalnych bardziej wpływa na trudność problemu niż ich ilość.

Clarifying the Difference in Local Optima Network Sampling Algorithms (2019) [9]

W pracy wykonane zostało statystyczne porównanie dwóch algorytmów próbkowania sieci lokalnych optimów dla kwadratowego problemu przydziału(QAP). Do badań wykorzystano instancje problemu z biblioteki QAPLIB. Wybrano 30 instancji o różnych klasach (rozłożone równomiernie, na siatce, instancje rzeczywiste, instancje losowe symulujące rzeczywiste). Porównano algorytmy *Snowball* oraz *Iterated Local Search*. Spróbkowane dane wykorzystano do stworzenia modelów regresji do przewidywania jakości rozwiązań dwóch algorytmów heurystycznych: *Robust Taboo Search* Taillarda oraz *Improved ILS* Stützla.

Wykonano próbkowanie przestrzeni rozwiązań i sprawdzono, czy wartości poszczególnych miar pochodzące od dwóch algorytmów są ze sobą skorelowane. Zbadanymi miarami były m.in: liczba optimów, liczba krawędzi, średnia liczba krawędzi wychodzących z wierzchołka grafu(ang. outdegree), średnica grafu, średnie dopasowanie lokalnych optimów(ang. meanfitness), sinkfitness (dopasowanie spływów? TODO: translate). Zauważono korelację metryk meanfitness i sinkfitness na poziomie 0.99. Dla wszystkich innych metryk korelacja było mniejsza od 0.5, dla średnicy grafu była ona ujemna. Na podstawie zebranych danych nauczono modele typu liniowego oraz *RandomForest*. Zbadano wartości metryki R^2 dla różnych parametrów algorytmów próbkowania, algorytmów heurystycznych i modelów regresji. Obliczono również współczynniki korelacji pomiędzy różnymi właściwościami sieci lokalnych optimów a performance gap metric(TODO: translate) algorytmów optymalizujących.

Badania wykazały, że sieci wygenerowane przez dwa różne algorytmy próbkujące mają pewne wspólne cechy. *Snowball* okazał się bardziej przewidywalny i łatwiejszy w dobieraniu parametrów. ITS natomiast lepiej znajdował struktury skupiające (ang. hub-and-spoke structures, hubs) w przestrzeni rozwiązań. Wartości metryk LON uzyskane na podstawie algorytmu ITS były bardziej skorelowane z jakością rozwiązania heurystyki(ang. heuristic performance metric). Modele regresji utworzone na podstawie danych uzyskanych z ITS dokonywały lepszej predykcji jakości rozwiązań algorytmów heurystycznych, niż modele utworzone na podstawie algorytmu Snowball. Zauważono, że wartości funkcji celu w poszczególnych lokalnych optimach sieci LON były lepszymi predyktorami, niż informacje o krawędziach tej sieci.

Inferring Future Landscapes: Sampling the Local Optima Level (2020) [10]

W pracy zostały porównane algorytmy próbkowania sieci optimów lokalnych(LON) dla kwadratowego problemu przydziału (QAP). Porównane zostały algorytmy *Markov-Chain* oraz *Snowball*. Dla małych instancji problemu wykonano kompletne przeszukanie przestrzeni rozwiązań, dla większych - przeszukiwanie z limitowanym budżetem obliczeniowym.

Jako instancje testowe wykorzystano przykłady ze zbioru QAPLIB. Instancje należały do pięciu kategorii: instancje z wartościami losowymi z rozkładu równomiernego, instancje z wartościami odległości ułożonymi "na siatce" i losowymi wartościami przepływu, rzeczywiste instancje problemu, instancje generowane symulujące rzeczywiste, oraz instancje nie pasujące do żadnej z poprzednich kategorii.

Badane właściwości sieci LON różniły się w zależności od algorytmu próbkowania. Dla Algorytmu *Markov-chain* były to: liczba wierzchołków i krawędzi grafu, średnie dopasowanie(ang. meanfitness), średnia liczba krawędzi wychodzących z wierzchołka (ang. mean outdegree), oraz najdłuższa ścieżka między wierzchołkami. Dodatkowo zbadano tzw. metryki lejowe(ang. Funnel metrics) - liczba krawędzi do globalnego optimum (ang. incoming global) oraz średnie dopasowanie spodu leja(ang. mean funnel-floor fitness).

Dla algorytmu *Snowball* metryki lejowe, oraz te oparte o gęstość i wzory połączeń krawędzi nie dają użytecznych informacji, zamiast tych metryk wybrano: średnią wagę pętli w grafie, średnią różnicę wag krawędzi wychodzących (ang. mean weight disparity), korelację dopasowania między sąsiadami (ang. fitness-fitness correlation), średnia długość "wspinaczki" do optimum lokalnego, maksymalna długość "wspinaczki" do lokalnego optimum oraz maksymalna liczba ścieżek prowadzących do lokalnego optimum.

Na podstawie danych zbudowano modele regresji, dokonujące predykcji jakości rozwiązań uzyskanych przy wykorzystaniu heurystyk *Improved Iterated Local Search* i *Robust Taboo Search*. Sprawdzono modele liniowe i *RandomForest*. Modele oparte o *RandomForest* okazały się lepsze.

Badania nie wykazały wyższości jednego algorytmu próbkowania nad drugim. Autorzy zwracają uwagę na fakt, że niektóre z cech sieci optimów lokalnych są lepiej odwzorowywane przez algorytm *Markov-chain*, a inne lepiej przez *Snowball*. Najlepsze wśród zbudowanych modeli regresji okazały się zaś te, które korzystały z kombinacji danych pozyskanych z obu algorytmów.

Understanding Parameter Spaces using Local Optima Networks: A Case Study on Particle Swarm Optimization[2]

W pracy sieci optimów lokalnych zostały wykorzystane do analizy i wizualizacji przestrzeni parametrów heurystyki Particle Swarm Optimization dla różnych funkcji celu. Przebadano 26 różnych funkcji celu. Utworzono sieci LON, oraz CMLON (compressed LON). Zbadano: liczbę wierzchołków i krawędzi w LON, liczbę pętli w sieci, Liczbę konfiguracji, dla których osiągnięto wynik zbliżony do najlepszego o ustalony ϵ, stosunek krawędzi zmierzających do rozwiązania lepszego o ponad ϵ do ogółu krawędzi (nie licząc pętli), stosunek krawędzi neutralnych (nie zmierzających do rozwiązania lepszego lub gorszego o ϵ) do ogółu krawędzi, i krawędzi pogarszających do ogółu krawędzi. Dla sieci CMLON zbadana została liczba ścieków, optymalnych globalnie ścieków, rozmiar leja dla najlepszego ścieku i inne. Badania wykazały istnienie dużej ilości optimów lokalnych, co sugeruje że naiwne metody dobierania parametrów mogą łatwo doprowadzić do suboptymalnej konfiguracji, Przestrzeń konfiguracji charakteryzowała się niską neutralnością. Dla 20 z 26 celu istniało tylko jedno globalne optimum, co oznacz że najlepsze konfiguracje są do siebie zbliżone, a nie rozsianie po przestrzeni. Podsumowując: przestrzeń konfiguracji PSO zwykle ma jeden duży lej zmierzający do najlepszej konfiguracji, jednak istnieje bardzo dużo lokalnych optimów, oraz ścieków suboptymalnych, co sprawia że dobór parametrów jest nietrywialny.

MLON jest grafem tworzonym na podstawie sieci LON, ale zawierającym tylko niepogarszające krawędzie. CMLON jest grafem tworzonym na podstawie MLON, w którym wierzchołki stanowią płaskowyże sieci MLON. Płaskowyż jest spójnym podgrafem, w którym wszystkie wierzchołki mają tą samą wartość funkcji celu. krawędzi są tworzone poprzez agregację krawędzi z MLON.

Understanding AutoML Search Spaces with Local Optima Networks (2022) [8]

AutoML jest procesem automatyzacji konfiguracji procesów(pipelines) uczenia maszynowego. Może obejmować on takie zadania jak wstępne przetwarzanie danych, ekstrakcja cech, dobór odpowiedniego algorytmu i jego hiperparametrów. Mnogość możliwych konfiguracji procesu sprawia, że przestrzeń przeszukiwania AutoML jest duża. W opisanej pracy autorzy podejmują próbę analizy przestrzeni AutoML dla zadania klasyfikacji przy wykorzystaniu sieci optimów lokalnych.

Przestrzeń przeszukiwania AutoML cechuje się dużą liczbą parametrów warunkowych - parametrów, których istnienie zależy od wartości innych parametrów. Jest to spowodowane tym, że algorytmy uczenia maszynowego przyjmują inny zestaw hiperparametrów.

Z tego powodu do reprezentacji przestrzeni rozwiązań wykorzystano odpowiednią gramatykę BNF z rozwiązaniami generowanymi przy użyciu drzewa wyprowadzania gramatyki (grammar derivation tree). Powstała gramatyka posiada 38 reguł decydujących o zastosowanych algorytmach wstępnego przetwarzania danych, zastosowanego algorytmu klasyfikacji oraz jego hiperparametrów. Spośród algorytmów klasyfikacji wykorzystano: Regresję logistyczną, MLP, KNN, Random Forest i Ada Boost.

Wierzchołki LON zdefiniowano jako lokalne maksima przestrzeni rozwiązań. Jako wartość dopasowania wybrano uśrednioną wartość metryki F-score dla danej konfiguracji procesu. Sąsiedztwo rozwiązania zdefiniowano jako zbiór rozwiązań możliwych do uzyskanie poprzez zastosowanie operatora mutacji na danym rozwiązaniu. Za operator mutacji przyjęto losowe wybranie jednego wierzchołka z drzewa rozwiązania i zastąpienie go losowym drzewem (przy zachowaniu zasad gramatyki procesu). Szansa na wylosowanie wierzchołka została uzależniona od głębokości jego pozycji w drzewie. W ten sposób operacje zmieniające proces w znaczny sposób (np. zmiana algorytmu) mają mniejsze prawdopodobieństwo wystąpienia niż operacje zmieniające jedynie wartość hiperparametrów. Zbudowano sieci oparte o 3 różne definicje krawędzi - basin-transition edges, escape edges i perturbation edges. Do próbkowania przestrzeni rozwiązań w celu utworzenia LON wykorzystano algorytm wstępujący(hill-climb).

Do badań wykorzystano 7 zbiorów danych, zbadano 30 różnych przestrzeni konfiguracji. Ze względu na złożoność obliczeniową badania sąsiedztwa, próbkowano ograniczoną liczbę sąsiadów. Parametr ten nazwano wielkością sąsiedztwa i zbadano 4 warianty - 20, 30, 50 i 100 sąsiadów. W każdym przypadku wartość dopasowania została obliczona dla wszystkich rozwiązań w przestrzeni. Nadano budżet 120 sekund na wykonanie obliczeń określających wartość dopasowania danego rozwiązania. W przypadku przekroczenia otrzymywał on wartość 0.

Zbadano korelację między wartością dopasowania optimum lokalnego a rozmiarem jego basenu przyciągania w zależności od rozmiaru sąsiedztwa. Wykazano, że istnieje silna korelacja i zwiększanie sąsiedztwa w większości przypadków zmniejsza liczbę basenów przyciągania w przestrzeni i zwiększa rozmiar istniejących. Ta zależność nie zachodziła jednak w przypadku jednego z badanych zbiorów danych.

Zauważono wpływ zastosowanej definicji krawędzi na powstałą sieć LON - w sieciach z krawędziami typu basin-transition nie zauważono obecności ścieków (wierzchołków bez krawędzi wychodzących, ang. sinks), natomiast istniało wiele źródeł (wierzchołków bez krawędzi wchodzących, ang. sources). Ilość źródeł zmniejszała się wraz ze wzrostem sąsiedztwa. Sieci oparte o escape edges miały natomiast dużo ścieków i mało źródeł. Zwiększanie rozmiaru sąsiedztwa prowadziło do zmniejszenia liczby ścieków. Podobny efekt dawało zwiększenie parametru D. W przypadku perturbation edges w sieci nie było ścieków ani źródeł niezależnie od rozmiaru sąsiedztwa.

Obliczono odległości między parami lokalnych i globalnych optimów. Stwierdzono, że odległości między optimami globalnymi były mniejsze od odległości między optimami lokalnymi i odległości pomiędzy optimami globalnymi i lokalnymi. Zwiększanie sąsiedztwa zmniejszało odległość między globalnymi optimami. Zauważono, że optima są skupione w pewnym rejonie przestrzeni rozwiązań, a nie rozłożone równomiernie.

2.2 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.

2.2.1 Sieć optimów lokalnych

Sieć optimów lokalnych(ang. Local Optima Network, LON) jest konstruktem zaprezentowanym po raz pierwszy w artykule[11], i rozwiniętym w pracy[6].

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

2.2.2 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.

2.2.3 Krawędzie

Krawędzie w sieci lokalnych optimów mogą być zdefiniowane na jeden z kilku sposobów. W literaturze[6][8] 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 $\wedge$?

2.2.4 Struktury lejowe

W sieci optimów lokalnych możemy wyróżnić sekwencje złożone z optimów lokalnych, których wartość dopasowania jest niemalejąca. Sekwencje te zwane są sekwencjami monotonicznymi[5]. 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.

2.2.5 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.

2.2.6 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.

2.2.7 Operacja 2-exchange

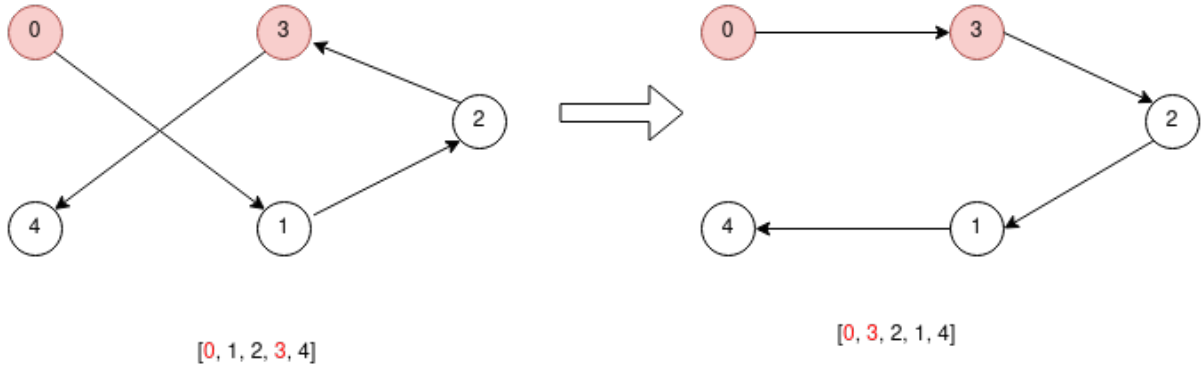
Operacja 2-exchange jest operacją wybrania dwóch wierzchołków i zamiany krawędzi prowadzących do następnych wierzchołków. Procedura jest wykorzystywana w algorytmie heurystycznym 2opt, w którym w ten sposób "rozplatane" są skrzyżowane krawędzie.

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[12], 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

$N \leftarrow \{\}$;
$E \leftarrow \{\}$;
for $i \leftarrow 1$ **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 $i \leftarrow 1$ **to** n_{max} **do**
    for $i \leftarrow 1$ **to** n_{att} **do**
      $s \leftarrow losoweRozwiazanie()$;
      $s \leftarrow 2opt(s)$;
      $N \leftarrow N \cup \{s\}$;
    end
  end
end

function `probkujKrawedzie`(N, E, e_{att})**:**
  foreach $n \in N$ **do**
    for $i \leftarrow 1$ **to** e_{att} **do**
      $s \leftarrow 2exchangeMutacja(n, D)$;
      $s \leftarrow 2optFirstImprovement(s)$;
      if $s \in N$ **then**
        $E \leftarrow E \cup \{(n, s)\}$;
        $w_{ns} \leftarrow w_{ns} + 1$;
      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
        $E \leftarrow E \cup \{(n, s)\}$;
        $w_{ns} \leftarrow 1$;
        $snowball(d - 1, m, s)$;
      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

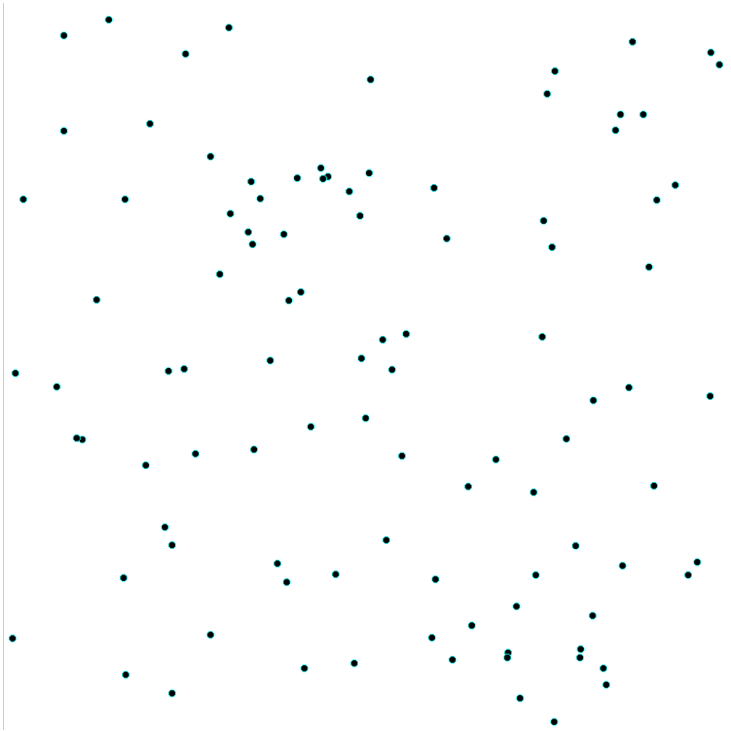
```
S ← {};
P ← wygenerujWszystkiePermutacje();
foreach p ∈ P do
    lo ← 2opt(p);
    S ← S ∪ {(p, lo)};
end
foreach (p, lo) ∈ S do
    foreach n ∈ N do
        if wZasiegu2Exchange(p, n, D) then
            if (n, lo) ∈ E then
                w_{n,lo} ← w_{n,lo} + 1;
            else
             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

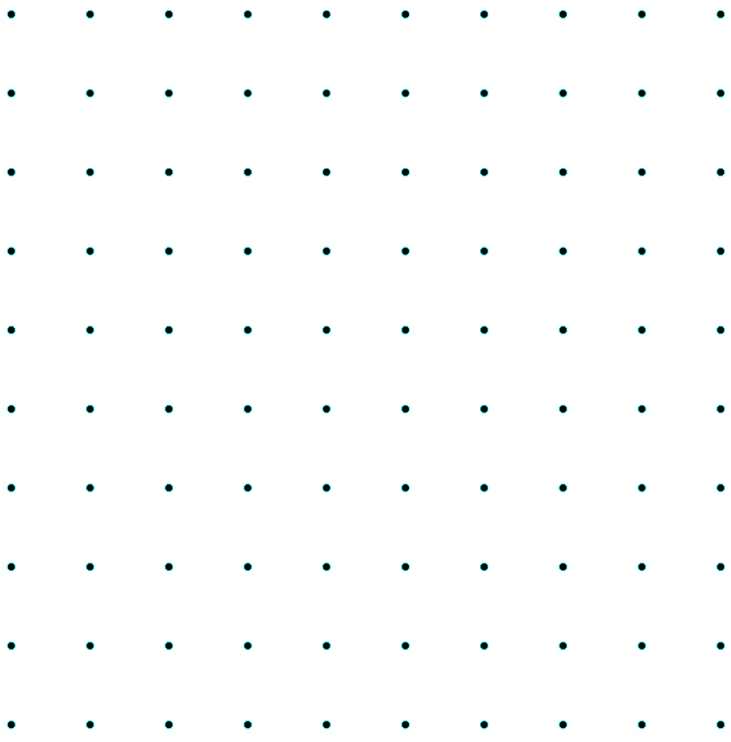
Miasta rozmieszczone na siatce

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] 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.

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

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

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

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.