Uniwersytet Mikołaja Kopernika
Wydział Matematyki i Informatyki

Jarosław Piersa
Numer albumu 183651

Praca magisterska
na kierunku informatyka

Algorytm optymalnej ramifikacji transportu na płaszczyźnie

Praca wykonana pod kierunkiem dr. hab. Tomasza Schreibera
Wydział Matematyki i Informatyki
Katedra Teorii Prawdopodobieństwa i Analizy Stochastycznej

Toruń 2008r

. . . . . .
Pracę przyjąłem i zaakceptowałem

. . . . . .
Potwierdzam przyjęcie pracy do dziekanatu

Spis treści

Abstrakt W poniższej pracy zostanie zaproponowany algorytm wyszukujący optymalną sieć drogową między dwiema dyskretnymi miarami probabilistycznymi w $\mathbb{R}^2$ przy wklęsłych funkcjonałach kosztu wymuszających ramifikacje, przedstawiony również zostanie opis jego implementacji w Javie.

Słowa kluczowe Ramifikacja, sieć drogowa, sieć transportowa, optymalizacja grafowa, symulowane wyżarzanie.

Rozdział 1

Wstęp

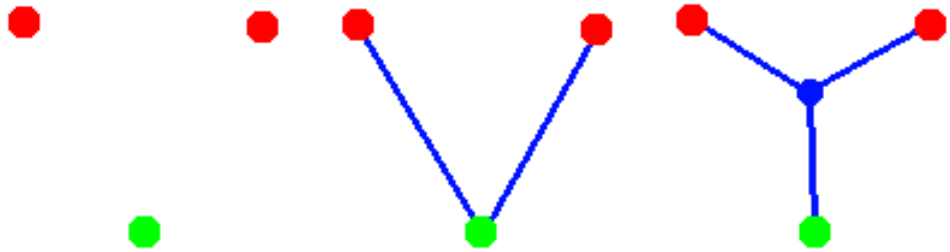
1.1 Wstęp

Problemy znalezienia właściwej drogi czy to w zatłoczonej metropolii, czy w grafach są od dawna znane i dobrze opisane. Znane są efektywne algorytmy dla wielu problemów dotyczących wyszukiwania ścieżek pomiędzy dwoma określonymi punktami, jak również dla całych zbiorów punktów. Większość z tych problemów na wstępie zakłada, że znana jest całkowita struktura dróg.

Jednakże tworzenie samej sieci drogowej łączącej określone punkty jest znacznie trudniejszym problemem. Świadczą o tym kilometry zawiłych uliczek, które wieki temu powstawały w sposób spontaniczny, bez dokładnego planowania (niejednokrotnie bez żadnego), a dziś są zmorą kierowców. Świadczą również o tym koryta rzek, które stale się zmieniają próbując dostosować swój kształt do bieżącego poziomu wody, warunków klimatycznych i innych czynników, a przy tym często powodują problemy podmywając grunty, podtapiając niskopołożone tereny, czy wdzierając się na obszary zabudowane.

Wracając do wspomnianych algorytmów wyszukujących ścieżki w grafach, warto zauważyć, że w wielu wypadkach działają one w sposób zachłanny — wybierają ścieżkę wyłącznie pod kątem trasy, którą same będą podążać. Projektowanie sieci drogowej może wymagać spojrzenia z szerszej perspektywy. Lepszym rozwiązaniem może się okazać wydłużenie dróg między dwoma indywidualnymi węzłami, w celu poprawienia globalnych cech sieci, jak sumaryczna długość wszystkich dróg.

Za modelowy przykład podaje się tu układ złożony z jednego źródła i dwóch odbiorców, umieszczonych w wierzchołkach trójkąta równobocznego. Problem ten sformułowany został przez Gaussa (zobacz [4]). Zachłanny algorytm doboru ścieżek każe połączyć odbiorcę i źródło bezpośrednią krawędzią. Daje to V–kształtną strukturę sieci. Jeżeli za miarę kosztu przyjąć suma-



(a) Rozmieszczenie źródła i odbiorców (b) Sieć V–kształtna (c) Sieć Y–kształtna

Rysunek 1.1: Różne sieci drogowe dla źródła i odbiorców rozmieszczonych w wierzchołkach trójkąta równobocznego

ryczną długość dróg, to „niedostosowanie" tego układu wyniesie $2 \cdot a$, gdzie a jest odległością między wierzchołkami trójkąta (patrz rysunek 1.1(b)).

Jednakże spoglądając na tę sieć z perspektywy globalnej, można znaleźć lepsze (tańsze) rozwiązanie. Należy w tym celu poprowadzić ścieżkę ze źródła wybranego wspólnego punktu wewnątrz trójkąta; ścieżką tą płyną jednocześnie transporty do obu odbiorców. Następnie należy rozdzielić transport, tj. poprowadzić oddzielne ścieżki od wybranego poprzednio punktu do obu odbiorców. Im bliżej wybrany punkt znajduje się od środka okręgu opisanego na trójkącie tym mniejszy będzie koszt sieci. W optymalnym przypadku, gdy punkt rozdzielenia pokrywa się ze środkiem okręgu dostosowanie wyniesie $\sqrt{3} \cdot a$. Takie rozlokowanie dróg określa się mianem Y–kształtnego (patrz rysunek 1.1(c)).

Jeśli jednak bliżej przyjrzeć się naturze, można stwierdzić, że jesteśmy otoczeni naturalnymi sieciami drogowymi uwzględniającymi Y–kształtne rozgałęzienia. Unerwienie liścia, sieć naczyń krwionośnych, wspomniane już rzeki, czy kształt błyskawicy są kilkoma zaledwie przykładami, które z mniejszym lub większym sukcesem próbujemy skopiować.

1.2 Specyfikacja problemu

1.2.1 Dane wejściowe i wyjściowe

Dane niech będą dwa skończone, niepuste i rozłączne zbiory punktów na płaszczyźnie

$$X, Y \subset \mathbb{R}^2, \quad X \cap Y = \emptyset, \tag{1.1}$$

nazywane dalej zbiorami źródeł i odbiorców odpowiednio. Dane są również funkcje

$$f : X \to \mathbb{R}_{>0} \tag{1.2}$$

$$g : Y \to \mathbb{R}_{>0}, \tag{1.3}$$

które spełniają

$$\sum_{x \in X} f(x) = \sum_{y \in Y} g(y),$$

nazywane dalej intensywnością źródła i odbiorcy odpowiednio. Określają one możliwości produkcyjne źródeł i konsumpcyjne odbiorców. Sumaryczna intensywność źródeł musi być równa sumarycznej intensywności odbiorców.

Wynikiem algorytmu ma być sieć drogowa reprezentowana jako trójka uporządkowana

$$(G, h, j) \tag{1.4}$$

Gdzie:

- $G = (V, E)$ jest grafem skierowanym i acyklicznym, reprezentującym wynikową sieć drogową na płaszczyźnie; $X, Y \subset V \subset \mathbb{R}^2$,
- $h : E \to \mathbb{R}_{>0}$ jest funkcją wag na krawędziach, określającą ile towaru transportowane jest przez daną krawędź e. Krawędzie o zerowym przepływie nie należą do grafu,
- $j : X \times Y \to \mathbb{R}_{\geq 0}$ jest ilością towaru transportowanego ze źródła $x \in X$ do odbiorcy $y \in Y$.

Dodatkowo żądamy aby spełnione były następujące wymagania:

- Jeżeli $j(x, y) > 0$ dla pewnego $(x, y) \in X \times Y$, to istnieje w G ścieżka z x do wierzchołka y, ozn. $p_{x,y}$,
- Wymagamy, by przepływ przez krawędź e był równy dokładnie sumie transportów między źródłami, a odbiorcami takimi, że ścieżka $p_{x,y}$ między nimi zawiera krawędź e.

$$\forall_{e \in E} h(e) = \sum_{(x,y) \in X \times Y \wedge e \in p_{x,y}} j(x, y) \tag{1.5}$$

- Wreszcie żądamy, by całkowita podaż źródła była zagospodarowana.

$$\forall_{x \in X} f(x) = \sum_{y \in Y} j(x, y) \tag{1.6}$$

W optymalnej sytuacji należałoby dodatkowo żądać, by zachodziła dualna równość:

$$\forall_{y \in Y} g(y) = \sum_{x \in X} j(x, y) \tag{1.7}$$

Innymi słowy, by popyt każdego odbiorcy był zaspokojony. Jednak ze względu na charakter algorytmu nie zdecydowano się na wprowadzenie tego warunku. Jest on zastąpiony składnikiem penalizującym w funkcji kosztu, o czym poniżej.

1.2.2 Funkcja kosztu

Funkcja kosztu określa stopień „niedostosowania" sieci drogowej. Przyjmuje wartości rzeczywiste i nieujemne. Im mniejsza jest jej wartość tym, lepszy jest system dróg.

W pracach [2] oraz[3] Quinglan Xia zaproponował następującą postać:

$$K(G, h) = \sum_{e \in E} |e| \cdot h(e)^{\alpha} \quad \alpha \in [0..1) \tag{1.8}$$

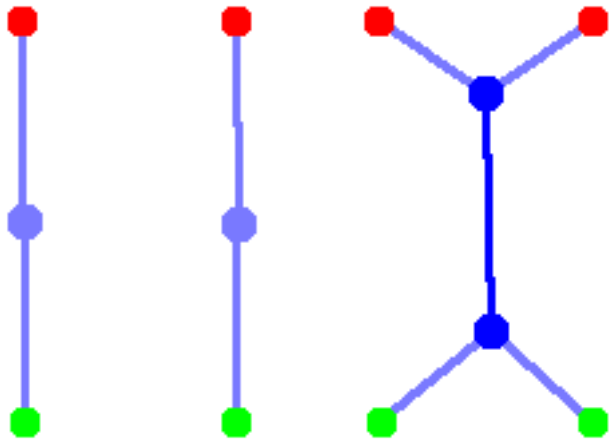
Koszt jest liniowo proporcjonalny względem sumarycznej długości krawędzi w grafie. Jednak posiada on również nieliniowy czynnik, zależny w sposób potęgowy od przepływu przez daną krawędź e. Powoduje to, że większość klasycznych algorytmów grafowych m. in. Prima i Kruskala (patrz np. [1]) nie daje optymalnego rezultatu. Z drugiej strony jedną z właściwości funkcji wklęsłej, a taką jest $l(x) = x^{\alpha}$ dla $\alpha \in (0..1)$, jest:

$$l(a + b) \leq l(a) + l(b), \quad a, b \geq 0$$

Warunek ten powoduje, że badziej opłacalne staje się łączenie mniejszych ścieżek transportujących w większe, po których przenoszone są liczne przepływy (rysunek 1.2).

Niestety tak sformułowana funkcja kosztu wymaga bardzo restrykcyjnego ograniczenia na zależności w grafie 1.7. Powoduje to trudności w przeszukiwaniu przestrzeni rozwiązań. Dlatego też założenie 1.7 zostało pominięte. W jego miejsce został wprowadzony dodatkowy składnik penalizujący do funkcji kosztu 1.8:

$$K(G, h, j) = c_1 \sum_{e \in E} |e| \cdot h(e)^{\alpha} + c_2 \sum_{y \in Y} (\sum_{x \in X} j(x, y) - g(y))^2 \tag{1.9}$$



(a) Ścieżki biegnące niezależnie.

(b) Ścieżki połączone w większą drogę.

Rysunek 1.2: Porównanie sieci drogowych dla tych samych źródeł i odbiorców.

$$\alpha \in [0..1)$$

$$c_1, c_2 \in \mathbb{R}_{\geq 0}$$

Dodany składnik określa dodatkową karę za niezgodne z danymi wejściowymi nasycenia odbiorców. Założono w pracy, że transportowany towar nie jest rozróżnialny i nie ma znaczenia, z którego źródła pochodzi, ani do którego odbiorcy dociera. Znaczenie posiada natomiast ilość towaru, jaką potrzebuje dany odbiorca i niezgodność z właśnie tym parametrem jest karana. Przyjęto, że intensywność źródeł jest stała, zadana w wejściowych danych i nie jest modyfikowana. Stąd też brak analogicznego składnika, który dodatkowo karałby za niezgodność z intensywnością źródeł. Skalary c_1 oraz c_2 są wagami, z jakimi składniki wchodzą do sumarycznego kosztu rozwiązania.

Mimo iż algorytm projektowany był z uwzględnieniem możliwości modyfikacji przepływów pomiędzy konkretnymi źródłami, a odbiorcami, możliwe jest zablokowanie tych zmian z poziomu interfejsu użytkownika. Wówczas w trakcie poszukiwań rozwiązania pomijane są modyfikacje zadanego typu (o czym dokładniej w dalszej części pracy). W takim wypadku wprowadzone początkowe przesyły będą zachowane, a algorytm „rozróżnia" typ transportowanego towaru. Wówczas składnik penalizujący zawsze będzie wynosił 0.

W szczególnej sytuacji przy parametrze $\alpha = 0$, gdy zbiór źródeł X jest jednoelementowy problem sprowadza się do geometrycznego problemu minimalnego drzewa Steinera (patrz [4]) ze zbiorem terminali $X \cup Y$.

W kontekście tych rozważań na miejscu jest podanie definicji klasycznego problemu Steinera.

Niech $G = (V, E)$ będzie grafem spójnym (nieskierowanym), a $h : V \to \mathbb{R}_{>0}$ będzie funkcją wag krawędzi. Niech $X \subset V$ będzie niepusty. Drzewem Steinera o zbiorze terminali X nazwiemy podgraf $T = (V_1, E_1)$ grafu G, taki że T jest drzewem oraz $X \subset V_1$.

Dla $G = (V, E)$, $h : V \to \mathbb{R}_{>0}$ oraz $X \subset V$, określonych jak wyżej, znalezienie drzewa Steinera o zbiorze terminali X o minimalnym koszcie nazywane jest grafowym problemem drzewa Steinera.

Dany niech będzie niepusty i skończony zbiór punktów $X \subset \mathbb{R}^k, k \in \mathbb{N}$, oraz metryka $d : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}_{\geq 0}$. Geometrycznym problemem Steinera nazwiemy znalezienie drzewa $T = (V_1, E_1)$ w przestrzeni $\mathbb{R}^k$, takiego że $X \subset V_1$ i T ma minimalny koszt (w sensie metryki d zadanej na krawędziach grafu).

Problem najczęściej jest rozważany dla $k = 2$ lub $k = 3$. Można pokazać, oba problemy są NP-zupełne (patrz [4]).

Rozdział 2

Idea algorytmu

2.1 Proponowany algorytm

Zaproponowany poniżej algorytm jest pewną modyfikacją symulowanego wyżarzania (ang. simulated annealing). Idea sprowadza się do przeszukiwania przestrzeni rozwiązań problemu poprzez poddawanie bieżącego rozwiązania pewnym losowym modyfikacjom. Zmodyfikowane rozwiązanie jest akceptowane, bądź odrzucane z prawdopodobieństwem zależnym od zmiany jakości rozwiązania w porównaniu do poprzedniego, numeru bieżącej iteracji, charakteru modyfikacji i innych czynników. Poza modyfikacjami losowymi graf poddawany jest również modyfikacjom o charakterze optymalizującym. Ich celem jest usunięcie lokalnych nieoptymalnych rozwiązań, które mogą być trudne, bądź czasochłonne do usunięcia poprzez modyfikacje losowe. Ten typ zmian nie podlega testom na akceptację — przyjmowany jest zawsze. Wraz z postępem obliczeń modyfikowane są niektóre parametry algorytmu, jak prawdopodobieństwa akceptacji, parametry natężenia zmian dokonywanych przez modyfikacje itp. Zatrzymanie obliczeń dokonywane jest przez użytkownika, gdy uzna, że aktualny wynik jest zadowalający.

2.2 Struktura danych

Struktura danych reprezentująca rozwiązanie problemu jest parą uporządkowaną składającą się z grafu oraz tablicy przepływu.

2.2.1 Tablica przepływu

Tablica przepływu przechowuje informacje o ilościach transportu pomiędzy źródłami, a odbiorcami (patrz funkcja j — 1.4). Przechowywane są

również informacje o intensywności źródeł i odbiorców (funkcje f oraz g — 1.2, 1.3). Można je obliczyć jako sumę danego wiersza i odpowiednio danej kolumny. Jak zostało wspomniane w specyfikacji problemu, sumaryczna intensywność źródła jest stała i nie może być modyfikowana. Może być co najwyżej zmieniany sposób dystrybucji ze źródła. Wadą takiego podejścia jest pewne uproszczenie problemu, co za tym idzie również funkcjonalności. Zaletą — uproszczenie postaci funkcji kosztu 1.9. Gdyby przyjąć możliwość zmian podaży, doszedłby do niej trzeci składnik odpowiedzialny za niezgodność intensywności źródeł z danymi wejściowymi.

Ponadto należy zauważyć, że przy ustalonej architekturze, koszt zależy monotonicznie od sumarycznej ilości transportu. Gdyby skalar odpowiedzialny za składnik penalizujący był zbyt niski, mogłaby występować tendencja do zerowania przepływu, by zminimalizować koszt, który byłby zdominowany przez wagę grafu. Z drugiej strony, wysoka waga dla składnika penalizującego, choć zapobiega powyższym problemom, skutkuje silnymi ograniczeniami w przeszukiwaniu całej przestrzeni rozwiązań. W skrajnej sytuacji nie byłoby możliwe wyskoczenie z pewnego minimum lokalnego, gdyż wzrost kosztu nowego rozwiązania uniemożliwiłby jego zaakceptowanie.

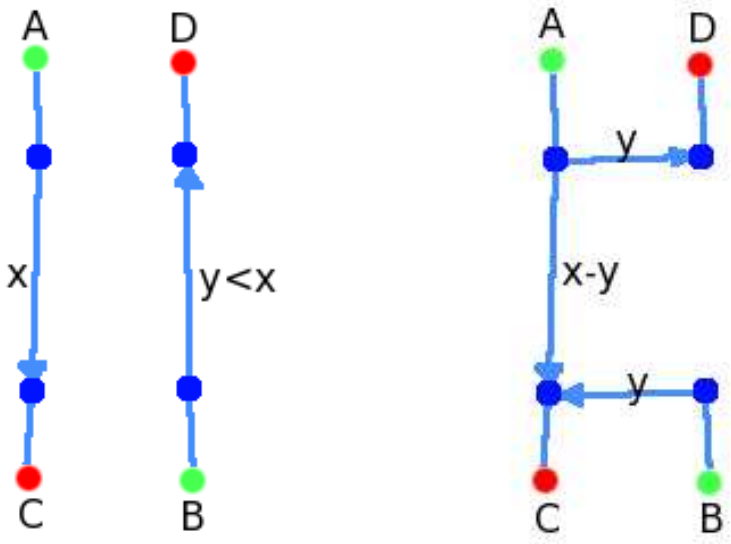
2.2.2 Graf transportu

Graf $G = (V, E)$ jest reprezentacją przebiegu sieci drogowej na płaszczyźnie (1.4). Zbiór wierzchołków V jest sumą zbiorów źródeł, odbiorców oraz wierzchołków pośrednich[1]. Wierzchołki pośrednie są punktami, w których mniejsze ścieżki łączą się w większą, bądź większa trasa rozdziela się na mniejsze. Przyjęto, że G jest skierowany. Krawędzie zorientowane są od źródeł do odbiorców.

W pewnych uogólnionych sytuacjach to założenie mogłoby uniemożliwić uzyskanie wyników choćby zbliżonych do optymalnych. Przykładowo, gdy dwie bliskie krawędzie biegną w przeciwnych kierunkach. Jednakże w przyjętej specyfikacji zawsze bardziej opłacalne będzie przeniesienie transportów do bliższych odbiorców, zaś przez krawędź popłynie różnica, pomiędzy przepływem silniejszym, a słabszym (patrz rysunek 2.1).

W dalszej części pracy poprzez „cykl" będzie rozumiana ścieżka o tym samym początku i końcu w grafie skierowanym, czyli z uwzględnieniem kierunków krawędzi. Jako „pętlę" będziemy określać ścieżkę o wspólnym początku i końcu, ale w grafie nieskierowanym, tj. nie uwzględniając orientacji krawędzi. Przyjmujemy, że krawędzie na ścieżce nie powtarzają się. Każdy cykl jest jednocześnie pętlą, lecz nie na odwrót.

[1] W problemie drzew Steinera nazywane są one wierzchołkami Steinera.



(a) Krawędzie transportujące w przeciwnych kierunkach.

(b) Przeniesienie transportu do bliższych odbiorców.

Rysunek 2.1: Przenoszenie transportu.

Przy przyjętej specyfikacji rozwiązania optymalne są zawsze pozbawione cykli i pętli. Przyjęto, że graf G, reprezentujący rozwiązanie pośrednie w trakcie optymalizacji, jest acykliczny, lecz mogą występować w nim pętle. Praca z grafami acyklicznymi pozwala pominąć niektóre testy warunków przy implementowaniu innych fragmentów algorytmu, między innymi wyszukiwanie drogi (skierowanej) w grafie.

W grafach nieskierowanych acykliczność implikuje jednoznaczność ścieżek. Gdy graf jest skierowany implikacja ta nie ma miejsca. Istnienie pętli w grafie może powodować istnienie wielu ścieżek między ustaloną parą wierzchołków.

W implementacji dopuszczono możliwość istnienia pętli w grafie reprezentującym rozwiązania pośrednie. Wadą takiego podejścia jest utrudniona praca ze ścieżkami w grafie; należy każdorazowo sprawdzać czy rozpatrywana ścieżka między wierzchołkami jest jedyna. Główną zaletą dopuszczenia pętli jest ułatwione przeszukiwanie przestrzeni rozwiązań. Jest ona obszerniejsza, niż przestrzeń grafów bez cykli i pętli, ale poruszanie się po niej poprzez losowe modyfikacje jest prostsze.

Dodatkowo obecność warunku jednoznaczności istnienia ścieżek wymusiłaby wprowadzanie złączeń i rozłączeń progresywnie zależnie od pierwszych zmian w grafie. Bez niego operacje mogą być dokonywane w niemal dowolnej części sieci od samego początku, co przyśpiesza działanie.

Problem, choć „wyrzucony", powraca jak bumerang przy niektórych modyfikacjach, gdzie brak tego założenia może zaburzyć strukturę grafu.

Choć istnienie więcej niż jednej ścieżki nie jest rozwiązaniem optymalnym, ostateczne rozwiązanie zwracane przez algorytm powinno być pozbawione pętli. W końcowych etapach działania algorytmu, pętle zostaną usunięte, zaś rozwiązania, które je posiadają, będą omijane ze względu na wysoki koszt.

Przyjęte zostały następujące ograniczenia na stopnie wierzchołków w grafie:

źródło stopnień wejściowy — 0 ($degin = 0$), stopień wyjściowy — dodatni ($degout \geq 1$),

odbiorca stopień wejściowy — nieujemny ($degin \geq 0$), stopień wyjściowy — 0 ($degout = 0$)

wierzchołek pośredni $1 \leq degin, degout \leq 2$

Ograniczenia te nie zmniejszają ogólności problemu, gdyż wierzchołki pośrednie o wyższych stopniach mogą być przybliżane poprzez większą ilość wierzchołków spełniających powyższe wymogi.

Źródła ani odbiorcy nie mogą być wierzchołkami tranzytowymi, trasa może zaczynać swój bieg (w wypadku źródeł), albo kończyć (odbiorcy), ale w jednym węźle nie mogą zachodzić obie sytuacje jednocześnie. Przez wierzchołek pośredni musi przebiegać trasa. Ale w jednym wierzchołku droga nie może się rozdzielać na więcej niż dwie mniejsze i podobnie, w jednym wierzchołku łączyć się mogą tylko dwie mniejsze trasy w większą. Możliwe są również „skrzyżowania" w których zbiegają się i wychodzą po dwie trasy.

Fakt Każdy węzeł Steinera w minimalnym drzewie Steinera na płaszczyźnie ma stopień 3, a kąt pomiędzy sąsiednimi krawędziami wynosi $\frac{2\pi}{3}$ (patrz [4] rozdział 3.3, zadanie 3.8 s. 35).

Założenia dotyczące stopni dają górne oszacowanie ilości krawędzi w grafie na poziomie $\frac{1}{2}\sum_{v \in V} degin(v) + degout(v) \leq 2|V|$, z dokładnością do stopni źródeł i odbiorców.

Przy dodatkowym założeniu, że końcowe rozwiązanie będzie drzewem (czyli nie posiada pętli), to oszacowanie może zostać poprawione.

Oznaczmy: $n = |X| + |Y|$ — ilość liści, $m = |V| - n$ — ilość węzłów pośrednich. Suma stopni wszystkich węzłów wynosi $1 \cdot n + 3 \cdot m$. Ponieważ

graf jest drzewem ilość krawędzi wynosi $|E| = |V| - 1 = m + n - 1$. Co daje:

$$m + n - 1 = \frac{1 \cdot n + 3 \cdot m}{2}$$

Po uproszczeniu $m = n - 2 = |V| - 2$. W efekcie mamy jeszcze lepszą liniową zależność ilości węzłów pośrednich od ilości liści w grafie, co za tym idzie również liniową ilość krawędzi. W praktyce jednak może być ich więcej, gdyż w rozwiązaniach pośrednich dopuszczane są węzły „proste" o jednym wejściu i jednym wyjściu. Z drugiej strony brak wymogu spójności zaniża szacowanie. Dokładny wynik jest zbyt zależny od danych wejściowych żeby cokolwiek poprawić, lecz nadal jednak jest to wynik optymistyczny dla ilości krawędzi w grafie i co za tym idzie szybkości algorytmu.

2.3 Modyfikacje grafu

Nieustanne poddawanie modyfikacjom bieżącego rozwiązania jest zasadniczą ideą algorytmu. Modyfikacje powinny z jednej strony błądzić po całej przestrzeni rozwiązań, z drugiej zaś unikać tych, które są tak złe, że niewiele da się je poprawić.

2.3.1 Przesunięcia typu Kohonena wierzchołków grafu

Idea została zapożyczona z algorytmu samoorganizacji topologicznej Kohonena. Można go znaleźć np. w [6].

Algorytm:

- Wylosuj punkt $p \in \mathbb{R}^2$ leżący w „pobliżu" aktualnego grafu;
- Znajdź wierzchołek $v \in V$ z grafu G leżący najbliżej p;
- Przesuń wierzchołek p oraz jego sąsiadów zgodnie ze wzorami

$$v.x = v.x(1 - \theta) + p.x \cdot \theta$$

$$v.y = v.y(1 - \theta) + p.y \cdot \theta$$

$$\theta \in (0..1)$$

Nie dotyczy to wierzchołków, które są elementami zbiorów X lub Y, przesuwać można tylko wierzchołki pośrednie.

Losowanie punktu z płaszczyzny powinno uwzględniać odległość od krawędzi grafu oraz ilość krawędzi leżących w pobliżu losowanego punktu. Dodatkowo należy zauważyć, że rozkład prawdopodobieństwa wylosowania punktów powinien się zmieniać wraz postępem obliczeń i zmianami grafu. Parametr θ należy do przedziału (0..1) i wraz z postępem obliczeń powinien maleć do zera. Np. mając zadaną pewną odgórną ilość kroków T i numer bieżącego kroku t parametr θ może przyjmować postać:

$$\theta = 1 - \frac{t-1}{T}$$

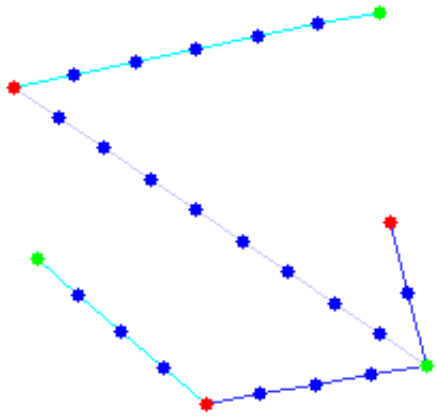
Alternatywną do zależności od odległości od krawędzi może być odległość od wierzchołków w grafie. Jednakże w takiej sytuacji należy dodatkowo zadbać o równomierne odległości między węzłami.

Idea wykorzystuje pewne właściwości algorytmu Kohonena. Pierwszą jest możliwość rozmieszczenia pewnych typów grafów planarnych (w tym drzew) na płaszczyźnie bez przecinania krawędzi. Drugą jest tendencja do przyciągania do siebie wierzchołków sąsiadujących — przynajmniej we wczesnych etapach obliczeń. Trzecią — sposób losowania punktów z płaszczyzny, który preferuje miejsca leżące w pobliżu wielu krawędzi. Takie bliskie krawędzie są dobrymi kandydatami do połączenia w większą ścieżkę, choć za to odpowiedzialny jest inny typ zmian. Kolejną specyficzną tendencją jest kurczenie i rozplątywanie pętli — sytuacji gdy dwa wierzchołki są połączone dwiema różnymi ścieżkami. Jak zostało zaznaczone wcześniej, takie sytuacje są dopuszczalne, choć nieoptymalne i dość kłopotliwe w rozwiązywaniu. Gdy taka pętla zostanie skurczona do trzech – czterech krawędzi możliwe jest, w dość prosty sposób, bezpośrednie znalezienie jej i usunięcie.

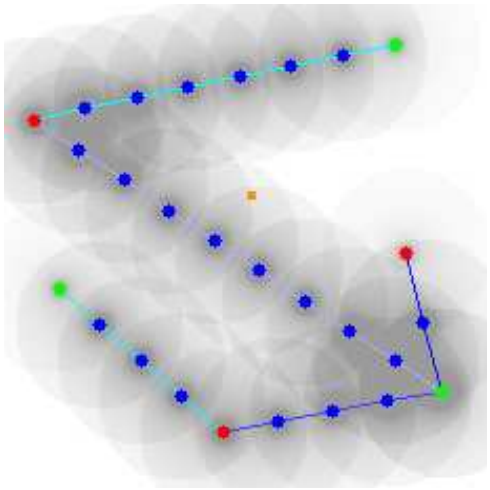
Ten typ modyfikacji nie zmienia struktury krawędzi w grafie; wpływ ma tylko na położenie wierzchołków. Działanie tego typu zmian w pierwszych iteracjach może być bardzo chaotyczne, natomiast po przekroczeniu maksymalnej ilości iteracji zanika. Z tego powodu wskazane jest unikanie wartości skrajnych dla parametru θ, lub wprowadzanie możliwości regulacji parametru w trakcie obliczeń. Działanie tego typu przesunięć akceptowane jest zawsze, niezależnie od zmiany kosztu.

2.3.2 Łączenia wierzchołków

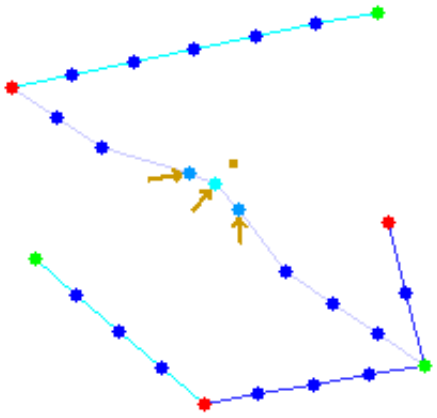
Łączenie wierzchołków jest podstawową modyfikacją operującą na strukturze krawędzi grafu. Jednocześnie jest to modyfikacja, która dokonuje wspomnianego w założeniach programu łączenia małych tras transportowych w większe „autostrady”.



(a) Graf przed modyfikacją.



(b) Mapa odległości punktów od wierzchołków grafu wraz z zaznaczonym wylosowanym punktem.



(c) Przesunięcie najbliższego węzła wraz z sąsiadami w kierunku wylosowanego punktu.

Rysunek 2.2: Przesunięcie wierzchołka.

Idea:

- Wylosuj wierzchołek pośredni p_1 z grafu G, $p_1 \notin X \cup Y$;
- Znajdź w G wierzchołek pośredni p_2, różny od p_1, leżący w jego „pobliżu" i nie połączony ścieżką z p_1. Jeżeli nie ma takiego wierzchołka zakończ modyfikację;
- Sprawdź ile p_1 i p_2 mają w sumie sąsiadów wchodzących. Jeżeli trzech lub więcej zakończ modyfikację;
- Sprawdź ile p_1 i p_2 mają w sumie sąsiadów wychodzących. Jeżeli trzech lub więcej zakończ modyfikację;
- Dodaj do grafu punkt p_3 typu pośredniego

$$p_3.x = \frac{p_1.x + p_2.x}{2}$$

$$p_3.y = \frac{p_1.y + p_2.y}{2}$$

- Przekieruj przepływ z przodków p_1 i p_2 do ich potomków tak by przechodził przez dodany wierzchołek p_3:
  - Dla każdej krawędzi $(p_1, u), u \in V$ w G dodaj krawędź (p_3, u) o tym samym przepływie;
  - Dla każdej krawędzi $(v, p_1), v \in V$ w G dodaj krawędź (v, p_3) o tym samym przepływie;
  - Dla każdej krawędzi $(p_2, u), u \in V$ w G dodaj krawędź (p_3, u) o tym samym przepływie. Jeżeli krawędź już istnieje to tylko zwiększ przepływ;
  - Dla każdej krawędzi $(v, p_2), v \in V$ w G dodaj krawędź (v, p_3) o tym samym przepływie. Jeżeli krawędź już istnieje to tylko zwiększ przepływ;
- Oblicz zmianę kosztu $k_2 - k_1$, gdzie:

$$k_1 = \sum_{e=(u,v)\in E, u=p_1 \vee v=p_1} |e| \cdot h(e)^\alpha + \sum_{e=(u,v)\in E, u=p_2 \vee v=p_2} |e| \cdot h(e)^\alpha$$

$$k_2 = \sum_{e=(u,v)\in E, u=p_3 \vee v=p_3} |e| \cdot h(e)^\alpha$$

- Jeżeli koszt zmalał to zaakceptuj modyfikację (usuń p_1 i p_2 z grafu wraz z wszystkimi incydentnymi do nich krawędziami);
- Jeżeli koszt wzrósł to zaakceptuj modyfikację z prawdopodobieństwem

$$P = \exp(-\beta(k_2 - k_1))$$

W przeciwnym przypadku odrzuć modyfikację (usuń p_3);

β jest tak zwaną temperaturą odwrotną, $\beta \geq 0$ i wraz z postępem obliczeń powinno rosnąć. Przykładowo może się zmieniać według wzorów (zaproponowanych w [7]):

$$\beta = \frac{c}{T}, \tag{2.1}$$

gdzie T jest temperatują (dodatnią i rosnącą), np.

$$T = \frac{1}{\lg(1+t)}, \tag{2.2}$$

gdzie t jest numerem bieżącej iteracji, natomiast $c > 0$ jest stałą.

W termodynamice temperatura jest zależna od średniej energii kinetycznej atomów układu. W wysokiej temperaturze ruch atomów w ośrodku charakteryzuje się dużą chaotycznością. W niskich temperaturach (w skali Kelvina tj. wciąż dodatnich) następuje stabilizacja (przykładowo substancja przechodzi ze stanu ciekłego w stały). Substancje fizyczne zmierzają do osiągnięcia tak zwanego stanu równowagi termodynamicznej, to jest stanu, w którym temperatura, ciśnienie i objętość pozostają stałe w czasie. Energia swobodna układu przyjmuje wartość ekstremalną w punkcie równowagi.

Gdy przeprowadzamy proces schładzania odpowiednio powoli, rozpatrywany w algorytmie układ (G, h, j) doprowadzany jest asymptotycznie do równowagi zerotemperaturowej, w której energia (tu nazywana kosztem grafu 1.9) osiąga minimum. Z prawdopodobieństwem równym 1 będzie to minimum globalne.

Dokładniejszy opis symulowanego wyżarzania wraz z teoretycznymi podstawami można znaleźć w [7] lub [5].

Liczne opisane wyżej warunki, które muszą zostać spełnione przez wierzchołki przed połączeniem, są niezbędne. Jak widać modyfikowana jest struktura krawędzi w grafie, więc ważnym jest aby nie stworzyć cyklu, przerwań ścieżki, bądź innych zmian niezgodnych z założeniami. Warunek dotyczący ograniczenia na ilość sąsiadów po połączeniu może być traktowany w sposób dość liberalny, o ile nie jest wykorzystywany jako założenie przy implementacji innych fragmentów algorytmu.

W sytuacji gdy łączone punkty nie posiadają wspólnych sąsiadów, wynikowe rozwiązanie zawsze ma koszt większy niż przed połączeniem. Z drugiej strony jest to pewien próg, po przeskoczeniu którego możliwe są dalsze łączenia krawędzi o wspólnym sąsiedzie, co już niemal zawsze prowadzi do pewnej redukcji kosztu. Dlatego należy zachować ostrożność przy dobieraniu parametru β. Jeżeli będzie zbyt wysoki, takie zmiany nie będą akceptowane, co ograniczy łączenia tylko do węzłów wchodzących do wspólnego odbiorcy, bądź wychodzących ze wspólnego źródła.

2.3.3 Rozłączenia wierzchołków

Ten typ modyfikacji jest przeciwny do łączeń. Powoduje rozłączenia wierzchołków, w których zbiegają się bądź rozbiegają dwie trasy. Użyteczność jej może wydawać się wątpliwa, lecz okazuje się ona niezbędna by nie dopuścić do utknięcia w ślepym zaułku. O ile jednak łączenia zmniejszają ilość wierzchołków, rozłączenia tworzą nowe (dokładniej — jeden nowy). Należy się zatem spodziewać pewnego lokalnego podproblemu do optymalizacji. Jest to istotnie trudniejsza modyfikacja i powinna być rozważana ze szczególną uwagą.

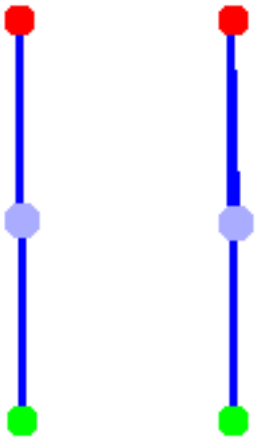
Dla wygody można wyodrębnić trzy sytuacje, w których można rozdzielić wierzchołek.

- Gdy posiada on dwóch sąsiadów wchodzących i jednego wychodzącego (λ-kształtne rysunki 2.4(a) i 2.4(b)),
- Gdy posiada on jednego sąsiada wchodzącego i dwóch wychodzących (Y-kształtne, patrz rysunki 2.4(c) i 2.4(d)) ,
- Gdy posiada on po dwóch sąsiadów wchodzących i wychodzących (X-kształtne, rysunki 2.4(e) i 2.4(f)).

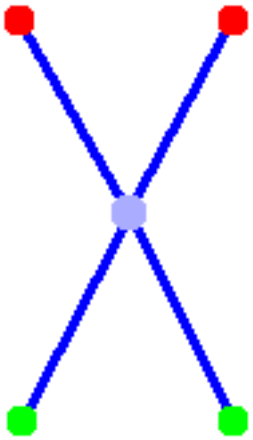
Pierwsze dwa przypadki są analogiczne, różnią się jedynie osobnym traktowaniem sąsiadów wejściowych i wyjściowych. Algorytm dla sytuacji λ-kształtnej (rysunek 2.4(a)):

- Wylosuj wierzchołek pośredni p o dwóch poprzednikach (p_1,p_2) i jednym następniku (p_3);
- Dodaj do grafu wierzchołki n_1 i n_2 typu pośredniego:

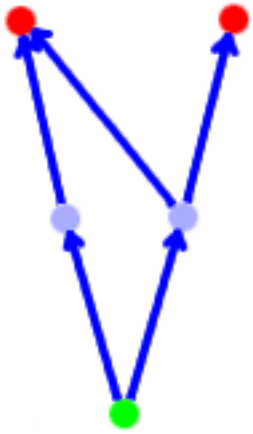
$$n_1.x = \frac{p_1.x + p_3.x}{2}$$



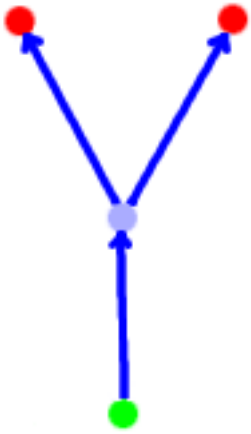
(a) Bez wspólnych sąsiadów, przed połączeniem.



(b) Po połączeniu.



(c) Łączone wierzchołki posiadające wspólnych sąsiadów.



(d) Po połączeniu.

Rysunek 2.3: Łączenie wierzchołków

$$n_1.y = \frac{p_1.y + p_3.y}{2}$$
$$n_2.x = \frac{p_2.x + p_3.x}{2}$$
$$n_2.y = \frac{p_2.y + p_3.y}{2}$$

- Przekieruj przepływ z p_1 i p_2 do p_3 przez nowo dodane wierzchołki. Dodaj do grafu krawędzie:
  - (p_1, n_1) o przepływie równym przepływowi (p_1, p);
  - (n_1, p_3) o przepływie (p_1, p);
  - (p_2, n_2) o przepływie (p_2, p);
  - (n_2, p_3) o przepływie (p_2, p);
- Oblicz zmianę kosztu $k_2 - k_1$ (k_1 jest kosztem rozwiązania przed modyfikacją, k_2 — po modyfikacji);
- Jeżeli koszt zmalał ($k_2 - k_1 < 0$) zaakceptuj modyfikację (usuń p);
- Jeżeli wzrósł — zaakceptuj modyfikację z prawdopodobieństwem,

  $$P = \exp(-\beta(k_2 - k_1))$$

  W przeciwnym przypadku odrzuć modyfikację — usuń n_1 i n_2;

β jak poprzednio jest temperaturą odwrotną, parametr jest dodatni i rośnie wraz z przebiegiem algorytmu. Parametr jednak nie musi być taki sam jak dla łączenia wierzchołków, mimo iż pełni identyczną funkcję.

Rozłączanie przypadków X-kształtnych (rysunek 2.4(e)) wygląda nieco inaczej. Poza połączeniem z czterema, zamiast trzema jak wcześniej, sąsiadami należy dodatkowo zwrócić szczególną uwagę na zachowanie zgodności ścieżek transportowych w grafie z wymaganiami opisywanymi przez tablicę przepływu.

Ogólna idea wygląda następująco:

- Wylosuj wierzchołek p do rozłączenia. p musi być wierzchołkiem pośrednim o dwóch przodkach (oznaczonych dalej p_1 i p_2) i dwóch potomkach (n_1, n_2);

- Dodaj do grafu dwa wierzchołki: s_1 i s_2 leżące na drodze między poprzednikami, a następnikami. Np.

$$s_1.x = \frac{p.x + n_1.x + 0.5p_1.x + 0.5p_2.x}{3}$$

$$s_1.y = \frac{p.y + n_1.y + 0.5p_1.y + 0.5p_2.y}{3}$$

$$s_2.x = \frac{p.x + n_2.x + 0.5p_1.x + 0.5p_2.x}{3}$$

$$s_2.y = \frac{p.y + n_2.y + 0.5p_1.y + 0.5p_2.y}{3}$$

- Przekieruj transport pomiędzy p_1 i następnikami (n_1 i n_2) i przechodzący przez p, w taki sposób, by przechodził przez s_1;

- Podobnie przekieruj transport pomiędzy p_2, a następnikami (n_1 i n_2) w taki sposób by przechodził przez s_2;

- Jeżeli w wyniku powyższego przekierowania w grafie istnieją wszystkie cztery następujące krawędzie to odrzuć zmianę (usuń s_1 i s_2);

$$\{(s_1, n_1), (s_1, n_2), (s_2, n_1), (s_2, n_2)\}$$

W sytuacji gdy w krawędziach nie są pamiętane przepływy pomiędzy źródłami, a odbiorcami, które przechodzą przez krawędź, operacje przekierowania transportu można wykonać w poniższy sposób. Wymagany jest jednak warunek, by ścieżki od źródeł do p i od p do odbiorców były jednoznaczne. Jest to wspomniany wcześniej lokalny problem optymalizacyjny do rozwiązania.

- Jeżeli istnieje źródło połączone z p więcej niż jedną ścieżką nie można wykonać modyfikacji.

- Jeżeli istnieje odbiorca połączony z p więcej niż jedną ścieżką nie można wykonać modyfikacji.

- Dane są dwa „lokalne źródła" czyli p_1 i p_2 oraz dwóch „lokalnych odbiorców" s_1 i s_2.

$$p_1.w, p_2.w, s_1.w, s_2.w > 0$$

$$p_1.w + p_2.w = s_1.w + s_2.w$$

Należy rozdysponować przepływ ze źródeł do odbiorców w sposób minimalizujący koszt, to jest dla każdej z krawędzi znaleźć przepływ tak aby:

$$(p_1, s_1).w, (p_1, s_2).w, (p_2, s_1).w, (p_2, s_2).w > 0$$

$$(p_1, s_1).w + (p_1, s_2).w = p_1.w$$

$$(p_2, s_1).w + (p_2, s_2).w = p_2.w$$

$$(p_1, s_1).w + (p_2, s_1).w = s_1.w$$

$$(p_1, s_2).w + (p_2, s_2).w = s_2.w$$

Ostatnie dwie równości dają liniowe zależności na szukane. Zmniejsza to ilość niewiadomych z czterech do dwóch.

$$(p_2, s_1).w = s_1.w - (p_1, s_1).w$$

$$(p_2, s_2).w = s_2.w - (p_1, s_2).w$$

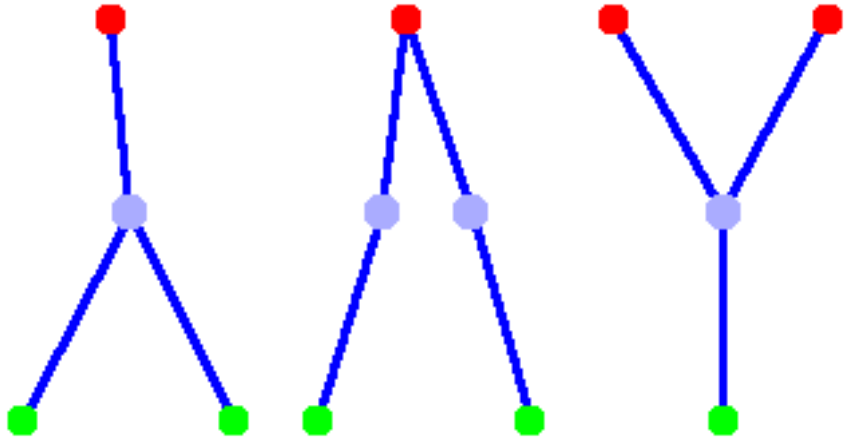
Z własności funkcji kosztu wynika, że optymalne rozwiązanie nasyca pewnego odbiorcę albo eksploatuje pewne źródło, innymi słowy jedna z niewiadomych wynosi zero. Wówczas również automatycznie mamy resztę rozwiązań. Możliwe są zatem cztery przypadki:

Lp.	$(p_1, s_1).w$	$(p_1, s_2).w$	$(p_2, s_1).w$	$(p_2, s_2).w$
1	$p_1.w$	0	$s_1.w - p_1.w$	$s_2.w$
2	0	$p_1.w$	$s_1.w$	$s_2.w - p_1.w$
3	$s_1.w - p_2.w$	$s_2.w$	$p_2.w$	0
3	$s_1.w$	$s_2.w - p_2.w$	0	$p_2.w$

W sytuacji gdy wartość przepływu wynosi zero krawędź nie jest dodawana. Nie każda z powyższych sytuacji jest możliwa. Może się zdarzyć, że źródło nie może nasycić odbiorcy i wówczas powstaną wyniki ujemne — nie są one dalej rozpatrywane.

- Wybierz najmniej kosztowną z powyższych czterech sytuacji, jednakże dodatkowo spełniającą warunek, że nadal istnieją ścieżki między źródłami, a odbiorcami. Jeżeli żaden z przypadków nie spełnia tego warunku odrzuć modyfikację (usuń s_1 i s_2);

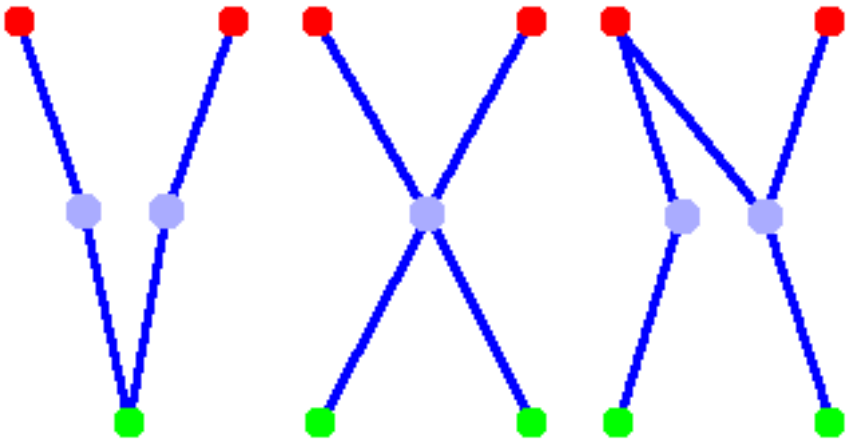
Należy zauważyć, że ilość niewiadomych jest iloczynem stopni wejściowego i wyjściowego wierzchołka p. Bez ograniczeń na te wartości problem staje się bardziej skomplikowany i wymaga zastosowania bardziej ogólnych narzędzi do rozwiązania.



(a) Kształt λ przed rozłączeniem.

(b) Kształt λ po rozłączeniu.

(c) Kształt Y przed rozłączeniem.



(d) Kształt Y po rozłączeniu.

(e) Kształt X przed rozłączeniem.

(f) Kształt X po rozłączeniu.

Rysunek 2.4: Rozważane przypadki rozłączeń wierzchołków.

Opisana powyżej modyfikacja rozdzielania X-kształtnego zawsze prowadzi do obniżenia kosztu. Wyjątkiem może być sytuacja gdy pamiętane są przepływy przez krawędź i istnieją różne ścieżki ze źródeł do punktu rozdzielanego.

2.3.4 Losowe przesunięcia wierzchołków

Modyfikacja przebiega następująco:

- Wybierz losowy wierzchołek pośredni z grafu — $p \in V \backslash (X \cup Y)$;
- Wylosuj niezależnie liczby δ_x i δ_y z rozkładu normalnego $N(0, \sigma^2)$;
- Przesuń wierzchołek p o wektor (δ_x, δ_y);

$$p.x = p.x + \delta_x$$

$$p.y = p.y + \delta_y$$

- Oblicz zmianę kosztu ΔK;
- Jeżeli koszt zmalał to zaakceptuj modyfikację;
- Jeżeli wzrósł to zaakceptuj modyfikację z prawdopodobieństwem

$$P = \exp(-\beta \Delta K)$$

Parametr $\beta \geq 0$ jest temperaturą odwrotną i wraz z postępem obliczeń powinien rosnąć. Podobnie jak poprzednio, choć oznaczony jest w sposób identyczny, nie musi być równy temperaturze odwrotnej rozważanej przy innych modyfikacjach. Parametr $\sigma > 0$ jest wariancją rozkładu normalnego. Wraz z postępem obliczeń jego wartość może być zmniejszana, choć nie jest to konieczne. Modyfikacji nie podlegają wierzchołki, które są źródłami lub odbiorcami.

Wprowadzane zmiany mają charakter lokalny — nie wpływają ani na przepływ między źródłami, a odbiorcami, ani na strukturę połączeń w grafie. Mają natomiast istotne znaczenie w końcowym etapie algorytmu, kiedy to uzyskana już została optymalna struktura sieci, a optymalizacji podlega położenie wierzchołków. Pozwalają one na usunięcie zakrętów na drogach oraz optymalizację kątów między sąsiednimi krawędziami.

2.3.5 Losowe usunięcia wierzchołków

Jest to pomocniczy typ zmian, nie prowadzi do znacznego zmniejszenia kosztu rozwiązania. Ma jednak istotny wpływ na działanie algorytmu, szczególnie opisane wcześniej przesuwanie typu Kohonena (zob. 2.3.1). Przebieg modyfikacji:

- Wybierz losowy wierzchołek pośredni $p \in V \backslash (X \cup Y)$, taki że posiada on dokładnie jednego poprzednika (oznaczonego jako p_1) i dokładnie jednego następnika (p_2). Jeżeli taki wierzchołek nie istnieje przerwij modyfikację.
- Oblicz $d = |(p_1, p)| + |(p, p_2)|$;
- Z prawdopodobieństwem zależnym od d wykonaj (im większa wartość d tym mniejsze powinno być prawdopodobieństwo):
  - Usuń wierzchołek p;
  - Uzupełnij lukę na ścieżce. Dodaj do grafu krawędź (p_1, p_2) o przepływie równym przepływowi przez krawędź (p_1, p). Jeżeli krawędź już istnieje to tylko zwiększ przepływ.

Powyższa modyfikacja nie spowoduje wzrostu kosztu, więc mechanizm jej akceptowania bądź odrzucania nie jest stosowany. Zasadniczym celem wprowadzania tego typu modyfikacji jest równomierne rozmieszczenie wierzchołków w grafie, zachowanie podobnych odległości między sąsiednimi wierzchołkami — oczywiście w granicach rozsądku.

Usuwanie blisko leżących punktów prowadzi do zmniejszania ilości krawędzi w pętli w grafie. (tj. fragmentów dwóch różnych ścieżek łączących te same punkty). Gdy długość pętli zostanie zredukowana do czterech wierzchołków możliwe jest jej znalezienie i usunięcie. Dodatkowo, w sytuacji gdy w losowym przesunięciu typu Kohonena punkt z płaszczyzny losowany jest jako leżący w pewnej odległości od wierzchołków problem nabiera istotności. Wówczas powoduje w miarę równomierny rozkład prawdopodobieństwa wylosowania punktów z płaszczyzny.

Wreszcie, gdy na pewnym obszarze znajduje się wiele krawędzi, dodawanie nowych wierzchołków na losowych połączeniach będzie preferowało właśnie ten obszar. Może to spowodować dalszy wzrost liczby krawędzi, a w konsekwencji lekceważenie innych fragmentów grafu, braki pamięci i inne błędy

natury numerycznej. Usuwanie wierzchołków (oraz opisane poniżej wstawianie) zapobiega tym niepożądanym sytuacjom.

Funkcja zależności prawdopodobieństwa od odległości do sąsiadów (występująca w punkcie trzeciem) powinna spełniać następujące warunki. Dla małych odległości powinna preferować usuwanie punktu, dla dużych odległości powinna pozostawić wierzchołek. Przykładowe funkcje:

$$f(x) = \begin{cases} 1 & x < a \\ \frac{x-b}{a-b} & a \leq x < b \\ 0 & b \leq x \end{cases}$$

$$f(x) = \frac{1}{1 + \exp(\beta(x - c))}$$

Dokładny dobór parametrów jest zależny od implementacji.

2.3.6 Losowe wstawianie wierzchołków

Jest to zmiana odwrotna do usuwania wierzchołków. Przebieg jest następujący:

- Wylosuj krawędź $e = (p_1, p_2)$ z grafu;
- Z prawdopodobieństwem zależnym od długości krawędzi e wykonaj:
  - Wstaw nowy wierzchołek p do grafu:

$$p.x = \frac{p_1.x + p_2.x}{2}$$

$$p.y = \frac{p_1.y + p_2.y}{2}$$

  - Przekieruj przepływ z p_1 do p_2 przez p. Dodaj krawędzie (p_1, p) i (p, p_2) o przepływie równym przepływowi przez e. Usuń e;

Działanie modyfikacji jest odwrotne do poprzedniej, natomiast cele bardzo podobne. Podstawowym jest zachowanie równomiernych długości krawędzi między wierzchołkami w grafie. Dodatkowo zwiększana jest liczba wierzchołków o stopniu (wejściowym i wyjściowym) równym jeden. Ten typ punktów jest wykorzystywany przy innych modyfikacjach, które istotnie zmieniają strukturę sieci, jak łączenie wierzchołków. Gdy jest ich zbyt mało, niektóre ze zmian nie mogą zostać wprowadzone.

Wspomniana funkcja zależności prawdopodobieństwa od długości krawędzi powinna wstawiać nowy punkt gdy krawędź jest długa, a odrzucać wstawianie w przeciwnym przypadku. Przykładowe funkcje:

$g(x) = 1 - f(x)$ gdzie f jest funkcją z modyfikacji usuwającej wierzchołki

$$g(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \leq x < b \\ 1 & b \leq x \end{cases}$$

$$g(x) = \frac{1}{1 + \exp(-\beta(x - c))}$$

Jak poprzednio, dobór parametrów zależny jest od implementacji.

2.3.7 Modyfikacje przepływu

Jest to bardziej subtelny typ zmian, który wpływa nie tylko na strukturę sieci drogowej, ale również na wartości transportu pomiędzy źródłami, a odbiorcami. Wprowadzane modyfikacje nie mają już charakteru wyłącznie lokalnego. Zmiany należy wprowadzać z zachowaniem ostrożności. Nieprzemyślane, lub niezabezpieczone mogą zaburzyć strukturę problemu.

Idea zmian polega na modyfikacji wartości przepływu pomiędzy źródłem, a dwoma odbiorcami w tablicy przepływu i następnie przeniesieniu tych zmian na graf. Algorytm modyfikacji przedstawia się następująco:

- Wybierz losowe źródło z;
- Wybierz losowo odbiorców o_1 i o_2, tak aby przepływ z z do o_1 był dodatni czyli $j(z, o_1) > 0$;
- Jeżeli ścieżki z z do o_1 i z z do o_2 (o ile istnieje) nie są jednoznaczne to nie można wykonać modyfikacji;
- Wybierz wartość przepływu d, który zostanie przeniesiony z o_1 do o_2, $d \leq j(z, o_1)$;
- Jeżeli istnieje już przepływ z z do o_2, czyli $j(z, o_2) > 0$ lub jeżeli istnieje ścieżka z z do o_2 (porównaj rysunek 2.5) wykonaj:
  - Zwiększ przepływ z z do o_2 o wartość d;
    - W tabeli przepływu: $j(z, o_2) += d$;
    - W grafie: znajdź ścieżkę s_2 z z do o_2;

        * Dla każdej krawędzi $e \in s_2$ zwiększ przepływ e o d;
    - Zmniejsz przepływ z z do o_1 o wartość d;
        * W tabeli przepływu: $j(z, o_1) - = d$;
        * W grafie: znajdź ścieżkę s_1 z z do o_1;
        * Dla każdej krawędzi $e \in s_1$ zmniejsz przepływ e o d

- W przeciwnym przypadku (to jest nie istnieje ścieżka z z do o_2; rysunek 2.6) wykonaj
    - Znajdź wierzchołek pośredni $p \in V \backslash (X \cup Y)$ najbliższy do o_2 i połączony ścieżką do z. Ewentualnie, dodatkowo niech p ma dokładnie jednego sąsiada wychodzącego;
    - Zwiększ przepływ z z do o_2 o wartość d;
        * W tabeli przepływu: $j(z, o_2) + = d$;
        * W grafie: znajdź ścieżkę s_2 z z do p;
        * Dla każdej krawędzi $e \in s_2$ zwiększ przepływ e o d;
        * Dodaj do grafu ścieżkę (ewentualnie pojedynczą krawędź) z p do o_2 o przepływie d;
    - Zmniejsz przepływ z z do o_1 o wartość d;
        * W tabeli przepływu: $j(z, o_1) - = d$;
        * W grafie: znajdź ścieżkę s_1 z z do o_1;
        * Dla każdej krawędzi $e \in s_1$ zmniejsz przepływ e o d

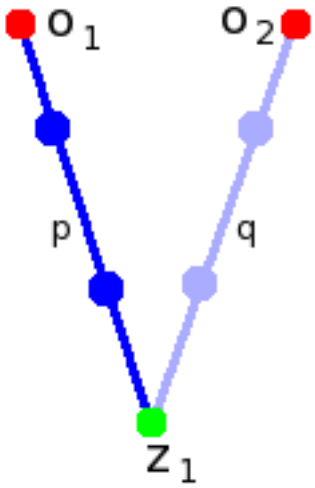
- Jeżeli koszt rozwiązania zmalał, to zaakceptuj modyfikację;

- Jeżeli wzrósł o ΔK to zaakceptuj z prawdopodobieństwem

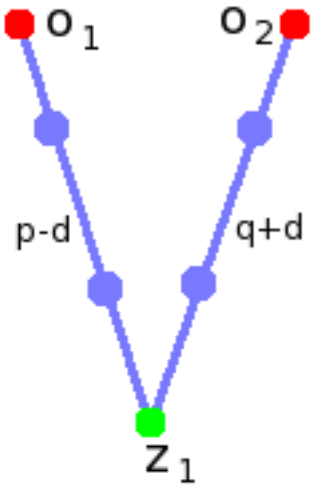
$$P = \exp(-\beta \Delta K)$$

Zmiana przepływu jest jedyną modyfikacją mającą możliwość zmiany przydziału transportu między źródłami, a odbiorcami. Wymaga jednak pamiętania jaki jest aktualny przebieg przepływu, lub przynajmniej możliwości jego odtworzenia. Stąd wymagania dotyczące jednoznaczności istnienia ścieżek. W przeciwnym wypadku możliwe byłoby obniżanie traportu na krawędziach, przez które nie biegnie przepływ.

Należy zauważyć, że jest to modyfikacja mająca wpływ na drugi ze składników funkcji kosztu, odpowiedzialny za penalizację rozwiązań odbiegających



(a) Sytuacja przed modyfikacją.



(b) Po modyfikacji, przepływ z z_1 do o_2 zwiększono o wartość d.

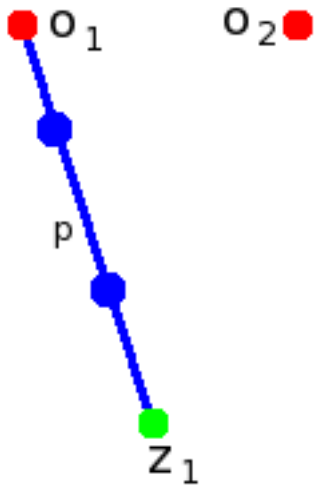
	o_1	o_2
z_1	$p \geq d$	q

(c) Tablica przepływu przed modyfikacją.

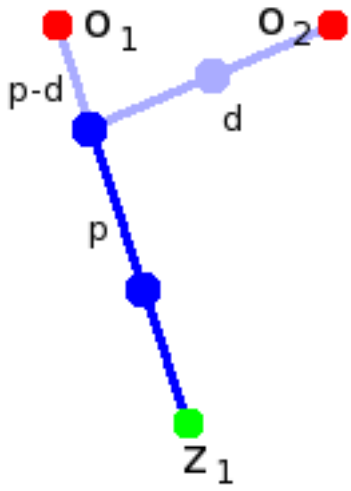
	o_1	o_2
z_1	$p - d$	$q + d$

(d) Tablica po modyfikacji.

Rysunek 2.5: Modyfikacja przepływu gdy istnieje ścieżka z z_1 do o_2.



(a) Sytuacja przed modyfikacją.



(b) Po modyfikacji, dodano przepływ z z_1 do o_2 o wartości d.

	o_1	o_2
z_1	$p \geq d$	0

(c) Tablica przepływu przed modyfikacją.

	o_1	o_2
z_1	$p - d$	d

(d) Tablica po modyfikacji.

Rysunek 2.6: Modyfikacja przepływu gdy nie istnieje ścieżka z z_1 do o_2.

od wejściowej specyfikacji (1.9). Ponieważ przyjęto założenie, że intensywność źródeł nie ulega zmianom, modyfikacja następuje w obrębie jednego źródła.

Wybór wierzchołka p najbliższego do o_2 i połączonego z z w algorytmie jest kompromisem między czasem obliczeń, a wyborem optymalnego wierzchłka. Przeliczanie dokładnych zmian kosztu dla każdego wierzchołka jest czasochłonne i powinno być unikane w często wykonywanych modyfikacjach. Z drugiej strony, wzięcie pod uwagę tylko sumarycznej długości ścieżek (z, p) i (p, o_2) zawsze kazałoby wybrać źródło jako punkt przecięcia $(p = z)$.

Sposób wyboru p nie jest optymalny, choć ma swoje zalety. Jako że przepływ jest zwiększany na istniejącej ścieżce z z do p, koszt zwiększa się wolniej niż linowo — tak została dobrana funkcja energetyczna (1.9). Można spodziewać się, że wstawiona krawędź (lub ścieżka) (p, o_2) będzie miała znaczący wpływ na zmianę kosztu.

Zauważmy również, że brak transportu między źródłem, a odbiorcą nie musi oznaczać, że oba wierzchołki nie są połączone ścieżką.

2.3.8 Przesunięcia rozgałęzień

Ostatnią modyfikacją są przesunięcia rozgałęzień w grafie. Modyfikacja ma dwie formy w zależności, czy rozważana jest przy rozgałęzieniu wchodzącym, czy wychodzącym. Podana poniżej wersja działa dla rozgałęzień wychodzących (Y-kształtnych — rysunek 2.7). Idea jest następująca:

- Wybierz wierzchołek pośredni $p_1 \in V \backslash (X \cup Y)$, który posiada przynajmniej dwóch sąsiadów wychodzących;
- Ustal losowo kolejność wierzchołków wychodzących, oznacz je jako p_2, p_3;
- Jeżeli ilość poprzedników p_1 wynosi dokładnie jeden to oznacz jedynego sąsiada wchodzącego jako p_4. W przeciwnym wypadku nie jest możliwe przesuwanie wstecz ścieżki.
- Jeżeli p_4 ma dwóch potomków, nie jest możliwe przesuwanie rozgałęzienia wstecz ścieżki;
- Jeżeli p_2 jest odbiorcą, przesuwanie wprzód nie jest możliwe;
- Dodatkowym można narzucić ograniczenie: jeżeli ilość następników p_2 wynosi dwa, to przesuwanie wprzód nie jest możliwe;
- Wybierz losowo typ modyfikacji: przesuwanie wstecz, lub przesuwanie wprzód. Jeżeli żadna sytuacja nie jest możliwa zakończ modyfikację.

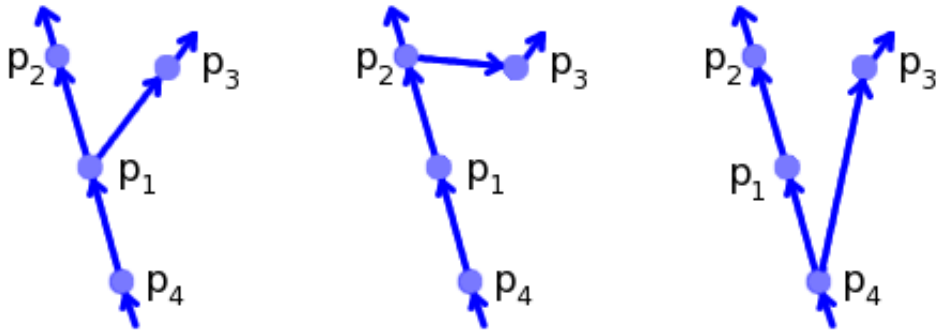
- Przesunięcie wstecz polega na przekierowaniu przepływu z p_4 do p_3 z pominięciem p_1;
  - Oznacz przepływ przez krawędź (p_1, p_3) jako d;
  - Dodaj krawędź (p_4, p_3) o przepływie równym d;
  - Usuń krawędź (p_1, p_3);
  - Zmniejsz przepływ przez krawędź (p_4, p_1) o d;
- Przesunięcie wprzód polega na przekierowaniu przepływu z p_1 do p_3 poprzez p_2;
  - Oznacz przepływ przez krawędź (p_1, p_3) jako d;
  - Dodaj krawędź (p_2, p_3) o przepływie równym d;
  - Usuń krawędź (p_1, p_3);
  - Zwiększ przepływ przez krawędź (p_1, p_2) o d;
- Oblicz zmianę kosztu ΔK
- Jeżeli koszt zmalał zatwierdź modyfikację;
- Jeżeli koszt wzrósł to zaakceptuj modyfikację z prawdopodobieństwem

$$P = \exp(-\beta \Delta K)$$

Analogicznie wygląda modyfikacja dla przesunięć rozgałęzień typu λ-kształtnego (rysunek 2.8).

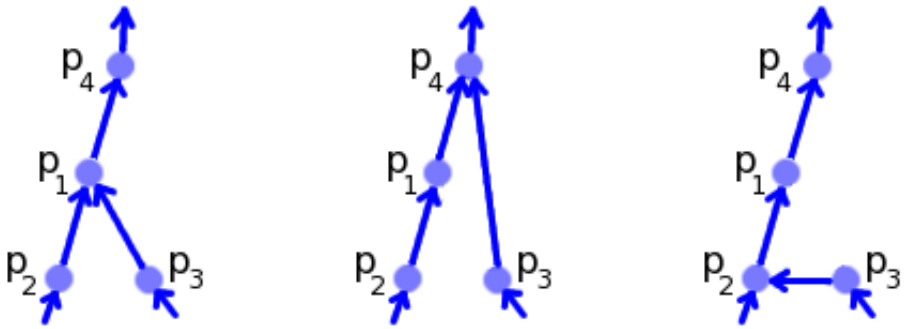
Podczas wprowadzania zmian należy zwrócić uwagę na przesunięcia rozgałęzień wychodzących wstecz (i analogicznie wchodzących wzdłuż ścieżki). Nie należy dopuszczać do cofnięcia rozgałęzienia przed węzeł, w którym dochodzi transport, analogicznie nie można dopuszczać do przesunięcia rozgałęzienia wchodzącego za węzeł, z którego wypływa transport.

Wymóg by wierzchołki oznaczone jako p_2 lub p_4 miały co najwyżej po jednym potomku, może zostać pominięty, lecz dopuszczane są wówczas sytuacje gdy ilość sąsiadów wchodzących (bądź wychodzących) jest większa niż dwa.



(a) Kształt Y przed przesunięciem. (b) Sytuacja po przesunięciu rozgałęzienia w przód ścieżki. (c) Sytuacja po przesunięciu rozgałęzienia w tył ścieżki.

Rysunek 2.7: Przesunięcia rozgałęzień w przypadku Y-kształtnym.



(a) Kształt λ przed przesunięciem. (b) Sytuacja po przesunięciu rozgałęzienia w przód ścieżki. (c) Sytuacja po przesunięciu rozgałęzienia w tył ścieżki.

Rysunek 2.8: Przesunięcia rozgałęzień w przypadku λ-kształtnym.

2.4 Usuwanie anomalii

Anomalią będą określane sytuacje w grafie, które choć dopuszczalne są jednak nieoptymalne i mogą być trudne do usunięcia przez powyższe losowe modyfikacje. Zasadniczym typem anomalii są pętle tj. dwie różne ścieżki łączące parę wierzchołków. Takie pętle same są źródłem niejednoznaczności ścieżek w grafie i uniemożliwiają wykonywanie niektórych modyfikacji jak zmiana przepływu. Dopuszczenie modyfikacji przy niejednoznacznych ścieżkach może natomiast prowadzić do powstania błędów.

2.4.1 Usuwanie pętli trzy-elementowych

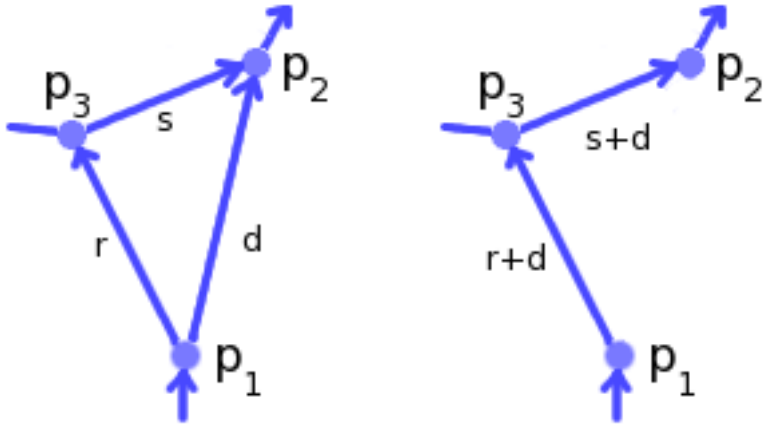
Najbardziej podstawowym typem pętli są pętle trzy-elementowe (rysunek 2.9). Algorytm wykrywania i usuwania przedstawia się następująco.

- Wybierz wierzchołek p_1;
- Dla każdego sąsiada wychodzącego p_2 węzła p_1 wykonaj;
  - Jeżeli istnieje ścieżka w dokładnie dwóch krokach (krawędziach) z p_1 do p_2 to
    - Oznacz wierzchołek pośredni tej ścieżki przez p_3;
    - Oznacz przepływ przez krawędź (p_1, p_2) przez d;
    - Zwiększ przepływ przez krawędzie (p_1, p_3) i (p_3, p_2) o d;
    - Usuń krawędź (p_1, p_2);
- Wybierz następny wierzchołek w grafie;

Efektywność algorytmu bazuje na założeniu, że stopnie węzłów są ograniczone przez dwa. Wówczas ilość iteracji po sąsiadach sprowadza się do dwóch obrotów pętli. Sprawdzenie czy istnieje ścieżka w dokładnie dwóch krawędziach zajmie co najwyżej cztery kroki. Łącznie daje do co najwyżej $8 \cdot |V|$ operacji, na pojedynczy wierzchołek. Bez tych założeń złożoność rośnie do $|V|^3$.

2.4.2 Usuwanie pętli cztero-elementowych

Pętle cztero-elementowe są drugim i najczęściej spotykanym typem anomalii. Bardzo rzadko dają się sprowadzić do pętli trzy-elementowej, więc ich odnajdowanie i usuwanie również zostało zaimplementowane. Pętle tego typu mogą występować w kilku postaciach, zależnych od orientacji krawędzi na pętli. Najczęściej występujące to:



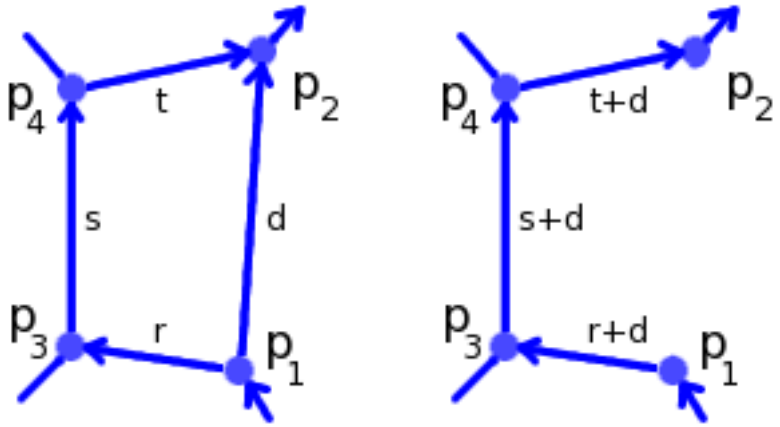
(a) Pętla trójelementowa w grafie. (b) Usunięcie pętli.

Rysunek 2.9: Usuwanie pętli trzy-elementowych.

- Pętle typu 1-3, łączą punkty ścieżką trzy-krawędziową i pojedynczą krawędzią (rysynek 2.10),
- Pętle typu 2-2 łączą wierzchołki dwiema różnymi ścieżkami dwukrawędziowymi (rysunek 2.11).

Algorytm usuwania pętli typu 1-3, jest niemal analogiczny do usuwania pętli trzy-elementowych:

- Wybierz wierzchołek $p_1 \in V$;
- Dla każdego sąsiada wychodzącego p_2 węzła p_1 wykonaj;
  - Jeżeli istnieje ścieżka w dokładnie trzech krokach (krawędziach) z p_1 do p_2 to
    - Oznacz wierzchołki pośrednie tej ścieżki przez p_3 i p_4;
    - Oznacz przepływ przez krawędź (p_1, p_2) przez d;
    - Zwiększ przepływ przez krawędzie (p_1, p_3), (p_3, p_4) i (p_4, p_2) o d;
    - Usuń krawędź (p_1, p_2);
- Wybierz następny wierzchołek w grafie;



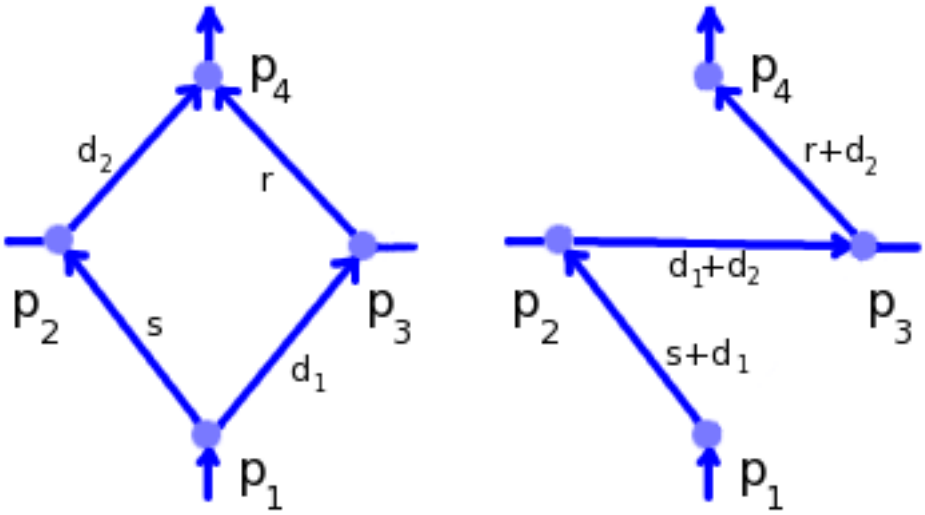
(a) Pętla cztero-elementowa w grafie. (b) Usunięcie pętli.

Rysunek 2.10: Usuwanie pętli typu 1-3.

Stwierdzenie istnienia ścieżki w trzech krawędziach wymaga co najwyżej 8 kroków. Ponownie daje to stałe oszacowanie na czas działania na pojedynczy wierzchołek. Bez ograniczeń na stopnie wierzchołków, złożoność się zwiększa do $O(|V|^4)$

Algorytm usuwania pętli typu 2-2:

- Wybierz wierzchołek $p_1 \in V$ o dwóch potomkach, oznacz ich jako p_2 i p_3;
- Znajdź wspólnego potomka dla p_2 i p_3, oznacz go jako p_4, jeżeli nie istnieje to wybierz następny wierzchołek, jeżeli istnieje więcej niż jeden, wybierz dowolnego z nich;
- Jeżeli w grafie nie istnieje krawędź (p_3, p_2) to przekieruj przepływ krawędzi (p_1, p_3) przez krawędź (p_2, p_3) oraz przekieruj przepływ (p_2, p_4) przez krawędź (p_2, p_3);
  - Oznacz przepływ przez krawędź (p_1, p_3) przez d_1;
  - Oznacz przepływ przez krawędź (p_2, p_4) przez d_2;
  - Zwiększ przepływ przez krawędź (p_1, p_2) o d_1;
  - Dodaj do grafu krawędź (p_2, p_3) o przepływie d_1+d_2, jeżeli krawędź już istnieje to tylko zwiększ przepływ;



(a) Pętla cztero-elementowa w grafie. (b) Usunięcie pętli.

Rysunek 2.11: Usuwanie pętli typu 2-2.

  - Zwiększ przepływ przez krawędź (p_3, p_4) o d_2;
  - Usuń krawędzie (p_1, p_3) oraz (p_2, p_4) z grafu;

- w przeciwnym wypadku (tj. istnieje krawędź (p_3, p_2)) przekieruj przepływ krawędzi (p_1, p_2) przez krawędź (p_3, p_2) oraz przekieruj przepływ (p_3, p_4) przez krawędź (p_3, p_2);
  - Oznacz przepływ przez krawędź (p_1, p_2) przez d_1;
  - Oznacz przepływ przez krawędź (p_3, p_4) przez d_2;
  - Zwiększ przepływ przez krawędź (p_1, p_3) o d_1;
  - Zwiększ przepływ przez krawędź (p_3, p_2) o $d_1 + d_2$;
  - Zwiększ przepływ przez krawędź (p_2, p_4) o d_2;
  - Usuń krawędzie (p_1, p_2) oraz (p_3, p_4) z grafu;
- Wybierz następny wierzchołek w grafie;

Rozdział 3

Implementacja algorytmu

3.1 Wykorzystane narzędzia

Program, w którym zaimplementowano powyżej opisany algorytm, został napisany w Javie — JDK 6 oraz Java IcedTea 1.7, w środowisku Eclipse 3.3. Interfejs użytkownika został zaprojektowany z użyciem bibliotek Swing. Do elementów graficznych wykorzystano edytor GIMP. Sama praca napisana została środowisku KILE 2.0.

3.2 Implementacja struktury danych

3.2.1 Tablica przepływu

Tablica przepływu, która reprezentuje funkcję j — ilość transportowanego towaru pomiędzy źródłami, a odbiorcami (patrz 1.4), jest reprezentowana poprzez trzy tablice typu `double`.

- `intensywnoscZrodla` reprezentuje możliwości produkcyjne źródeł (porównaj 1.1), jest ona wyspecyfikowana poprzez funkcję f (1.2) będącą parametrem wejściowym do programu i nie jest modyfikowana.

- `intensywnoscOdbiorcy` określa oczekiwany popyt odbiorców (1.1), tak jak powyższa tablica jest ona zadana poprzez funkcję g (1.3) i również nie jest modyfikowana.

- `macierzTransportu` jest tablicą dwuwymiarową (dokładniej — tablicą tablic) o wymiarach $|X| \times |Y|$. Określa ona jaki ułamek podaży źródła jest transportowany do poszczególnych odbiorców. Przyjęto założenie,

że jest to macierz stochastyczna, tj. zachodzi zależność

$$\forall_{x \in X} \sum_{y \in Y} \texttt{macierzTransportu[x][y]} = 1 \tag{3.1}$$

Przy tych założeniach funkcja wartość funkcji j wyraża się jako

$$\forall_{(x,y) \in X \times Y} j(x,y) = \texttt{intensywnoscZrodla[x]} \cdot \texttt{macierzTransportu[x][y]} \tag{3.2}$$

Faktyczny wpływ do odbiorców ma postać:

$$g_1(y) = \sum_{x \in X} \texttt{intensywnoscZrodla[x]} \cdot \texttt{macierzTransportu[x][y]}$$

Zauważmy, że zależności 3.1 oraz 3.2 dają wymaganą w specyfikacji własność 1.6.

Wybór macierzy stochastycznej odzwierciedla pewne własności algorytmu między innymi, brak możliwości modyfikowania podaży źródeł, czy dopuszczenie możliwości zmian popytu odbiorców. Gdyby zablokować tę drugą możliwość `macierzTransportu` byłaby dodatkowo podwójnie stochastyczna, lecz wówczas wrócilibyśmy do „sztywnej" wersji problemu opisanej w rozdziale pierwszym (patrz zależność 1.8).

Wybór typu double wiąże się z pewnymi problemami numerycznymi przy obliczaniu wartości przepływów. Jako że niezerowy przepływ w tablicy implikuje istnienie ścieżki w grafie, muszą zostać zastosowane pewne środki zabezpieczeń. Usunięcie przepływu może wiązać się z przerwaniem ścieżki, dlatego też konieczne jest upewnienie się, że w tym przypadku wartość w macierzy zostanie wyzerowana. W implementacji przyjęto, że wartości przepływów są zmieniane o wartość postaci $2^k, k \in \mathbb{Z}$. Jako, że są to liczby maszynowe są dokładnie reprezentowane w komputerze i minimalizują szanse wystąpienia błędów numerycznych. Problem rozwiązano poprzez unikanie odejmowania zmiennych „niebezpiecznych", tj takich które były poddawane operacjom arytmetycznym po przypisaniu.

3.2.2 Graf transportu

Graf transportu reprezentuje parę (G, h) — graf oraz funkcję wag (patrz 1.4).

Przyjęto, że wszystkie wierzchołki grafu są zawarte w zbiorze $[0..300) \times [0..300)$. Jako że jest to zbiór wypukły, wszystkie krawędzie grafu również są w nim zawarte.

Pierwszym istotnym ograniczeniem jakie zostało wprowadzone jest minimalna odległość między wierzchołkami w grafie. Daje to pewien minimalny próg długości krawędzi. Zapobiega to koncentrowaniu wierzchołków i krawędzi w jednym regionie. Gdy modyfikacji poddawana jest losowa krawędź lub losowy wierzchołek, taka centralizacja powodowałaby ignorowanie innych obszarów grafu. Wadą wprowadzenia tego progu jest wymóg sprawdzania czy w grafie istnieje już pobliski węzeł, przy każdej próbie wstawiania nowego, bądź przesuwania istniejącego. Wydaje się jednak, że niedogodności takiej nie da się uniknąć.

Wierzchołki w grafie reprezentowane są jako lista. Stosowanie tablicy jest dyskusyjne w sytuacji gdy graf (w tym zbiór wierzchołków) jest poddawany częstym zmianom (w tym usuwaniu i wstawianiu nowych).

```
private Vector<Punkt> wierzcholki;
```

Podobnie krawędzie są reprezentowane jako listy sąsiedztwa również jako obiekt klasa `java.util.Vector`.

```
private Vector<Vector<Krawedz>> sasiedzi;
```

Ponownie, przy częstych modyfikacjach reprezentacja poprzez macierz sąsiedztwa byłaby nieefektywna. Dodatkowo przy ograniczeniach na stopnie wierzchołków, stosowanie macierzy powodowałoby alokacje dużej ilości pamięci, która nie jest w ogóle wykorzystywana. Lista sąsiedztwa zawiera wyłącznie krawędzie wychodzące. Krawędzie wchodzące nie są zapamiętywane. W krawędziach zapamiętywane są oba wierzchołki oraz jej przepływ.

Jak poprzednio wartości zmiennoprzecinkowe są źródłem błędów numerycznych. Jako że, przepływ przez krawędź jest liczony jako iloczyn podaży źródła i elementu w macierzy przepływu, nie da się uciec do liczb maszynowych. Problem czy krawędź po odjęciu przepływu powinna zostać usunięta, rozwiązano wprowadzając minimalny próg błędu. Jeżeli po zmianie przepływ jest mniejszy niż wartość progowa, jest uznawany za zerowy, a krawędź usuwana. Podobne zabezpieczenia dotyczą ujemnych wartości przepływu.

3.3 Implementacja algorytmu

3.3.1 Przesunięcia typu Kohonena

W przypadku tego fragmentu algorytmu pewna swoboda jest pozostawiona w doborze sposobu losowania punktu, do którego przyciągane będą wierzchołki. Jak zostało zaznaczone w rozdziale opisującum algorytm, sposób losowania powinien uwzględniać odległość punktu od krawędzi w grafie oraz ilość krawędzi w pobliżu wierzchołka.

Zdecydowałem się na przybliżenie układu krawędzi poprzez układ wierz-

chołków w grafie. Wymaga to dodatkowo zachowania pewnej równomierności odległości pomiędzy węzłami, co było akcentowane w poprzednim rozdziale. Dodatkowo postanowiłem, że cała mapa punktów wraz z stopniem ich oddalenia od grafu będzie przechowywana w pamięci. Oczywiście losowanie może być wykonane bez obliczania takiej mapy, jednakże jest ona niezbędna dla wizualizacji sposobu losowania (patrz rysunek 3.1).

Ponadto losowanie bez mapy zawsze uwzględnia jedynie aktualny stan grafu. Mapa, o ile nie jest uaktualniana w każdym kroku, pamięta „kształt" sieci we wcześniejszych krokach. Powoduje to pewną bezwładność, ale z drugiej strony również odporność na nagłe zmiany, które mogą drastycznie zmieniać kształt grafu i przedwcześnie wykluczyć pewne obszary z losowania. Dodatkowo, w celu zwiększenia tej bezwładności nowa mapa jest średnią arytmetyczną poprzedniej oraz odpowiadającej stanowi grafu w chwili uaktualnienia.

W implementacji mapa odległości jest uaktualniana co 50 iteracji.

Dla każdego punktu (o obu współrzędnych całkowitych) stopień jego „bliskości" od grafu jest obliczany według wzoru.

$$p(x,y) = \max(\sum_{v \in V} odl(x,y,v), 1)$$

$$odl(x,y,v) = \begin{cases} 1 & d((x,y),(v.x,v.y)) = 0 \\ \frac{c}{d((x,y),(v.x,v.y))} & 0 < d((x,y),(v.x,v.y)) \leq r \\ 0 & r < d((x,y),(v.x,v.y)) \end{cases}$$

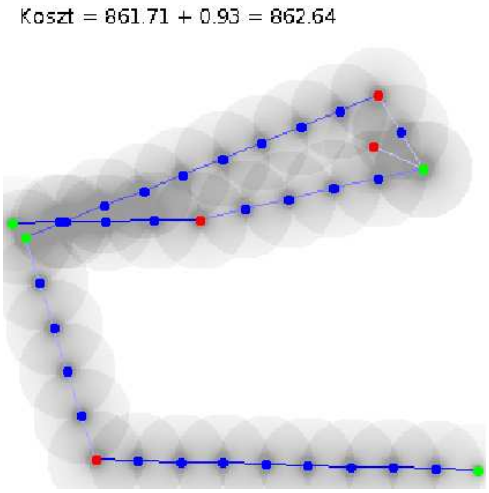
Gdzie d jest odległością euklidesową, r jest zależne od ustawień programu, jednakże powinno być porównywalne z odległościami między sąsiednimi krawędziami.

Losowanie punktu przebiega dwuetapowo.

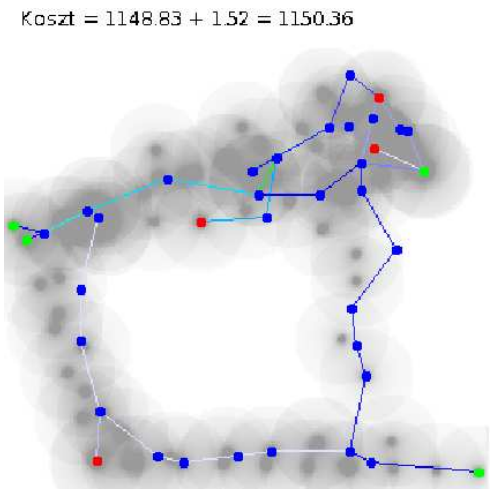
1. Wylosuj punkt (x, y) z płaszczyzny o niezerowej wartości $p(x, y)$ z rozkładu jednostajnego;

2. Zaakceptuj (x, y) z prawdopodobieństwem $p(x, y)$, jeżeli nie został zaakceptowany wróć do punktu 1;

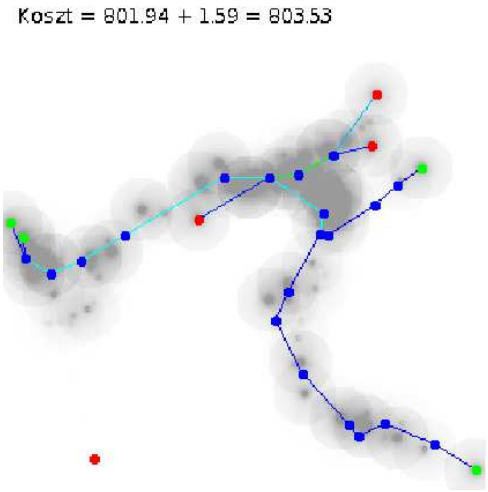
Jako, że graf jest niepusty $(X, Y \neq \emptyset)$ zawsze będzie istniał punkt o niezerowej p. Domyślnie $r = 40$, jednakże istnieje możliwość modyfikowania wartości tego parametru przez użytkownika. W programie nałożone zostały warunki by $r \in [10, 100]$.



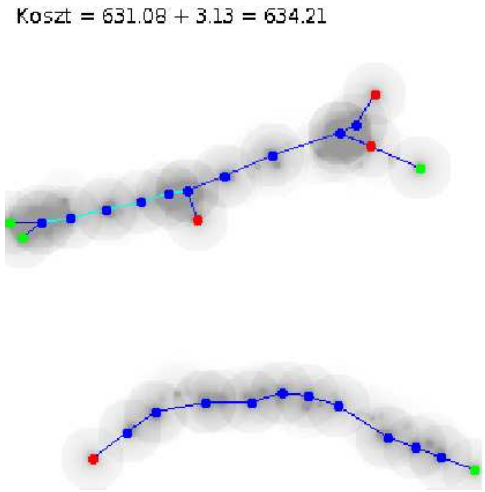
(a) Przed rozpoczęciem algorytmu.



(b) Początkowa faza obliczeń (ok. 100 iteracji).



(c) Zaawansowana faza obliczeń (ok. 1200 iteracji).



(d) Końcowa faza algorytmu (około 3000 iteracji).

Rysunek 3.1: Mapa odległości i struktura grafu w różnych etapach obliczeń

3.3.2 Łączenia wierzchołków

Zasadniczy algorytm został sprecyzowany w rozdziale drugim. Pewnego komentarza może wymagać dobór wierzchołków do połączenia.

W programie pierwszy z wierzchołków p_1 wybierany jest jako najbliższy do punktu z płaszczyzny wylosowanego według mapy odległości (patrz powyższa modyfikacja). Jako drugi wierzchołek p_2 wybierany jest taki, który spełnia:

- nie istnieje ścieżka z p_1 do p_2, ani z p_2 do p_1,
- p_2 nie jest oddalony od p_1 o więcej niż r, w programie przyjęto $r = 40$ z możliwością zmiany tej wartości przez użytkownika,
- jeżeli istnieje więcej niż jeden taki wierzchołek wybierany jest najbliższy do p_1, jeżeli nie istnieje żaden, modyfikacja nie jest wykonywana.

Temperatura odwrotna β, od której zależy prawdopodobieństwo zaakceptowania zmian, jest obliczana nieco inaczej niż w literaturze. Wprowadzona została możliwość modyfikacji temperatury przez użytkownika i, co za tym idzie, zmiany temperatury powinny uwzględniać również poprzednią wartość. Wartość temperatury odwrotnej kroku $i + 1$ jest opisana wzorem:

$$\beta_{i+1} = \min(\beta_i + \delta, \beta_{max}) \tag{3.3}$$

Przyjęto wartości $\delta = 0.01$ oraz górne ograniczenie na wartość temperatury odwrotnej wynoszące $\beta_{max} = 1000$. Przy automatyzacji algorytmu rozsądniej jednak będzie przyjąć wzory 2.1, 2.2.

3.3.3 Rozłączenia wierzchołków

Podobnie jak w łączeniu, wierzchołek jest losowany z mapy odległości. Jeżeli wylosowany wierzchołek nie spełnia wymogów, modyfikacja jest przerywana.

Temperatura odwrotna β w tym procesie jest identyczna jak przy łączeniu wierzchołków.

3.3.4 Losowe przesunięcia wierzchołków

Przyjęto następujące parametry dla rozkładu normalnego, z którego losowane są przesunięcia wierzchołków: $\mu = 0$, $\sigma^2 = 16$. Z twierdzenia o trzech σ wynika, że większość modyfikacji przesuwa wierzchołek nie dalej

niż o $12\sqrt{2} \simeq 17$ pikseli. Do generowania niezależnych liczb pseudolosowych wykorzystano klasę `java.util.Random` dostępną w bibliotekach Javy.

Temperatura odwrotna β jest obliczana według wzoru 3.3, jednakże, ze względu zastosowanie tego typu zmian w późnych etapach algorytmu, dodatkowo przemnożoną przez skalar $c = 0.1$.

3.3.5 Losowe wstawianie i usuwanie wierzchołków

W programie została zastosowana następująca funkcja zależności prawdopodobieństwa usunięcia wierzchołka w zależności od odległości do jego sąsiadów.

$$f(x) = \begin{cases} 1 & x < a \\ \frac{x-b}{a-b} & a \leq x < b \\ 0 & b \leq x \end{cases}$$

Przyjęto wartości $a = 30$, $b = 60$ pikseli. Brak potęgowania przyśpiesza nieco obliczenia.

Dla wstawiania wierzchołków przyjęto $g = 1 - f$ przy identycznych wartościach a i b.

3.3.6 Modyfikacje przepływu

Ten typ modyfikacji ingeruje zarówno w strukturę grafu jak i tablicę przepływu. Co za tym idzie należy dbać, o to by obie struktury pozostawały w ścisłej zależności. Przed wprowadzeniem jakiejkolwiek zmiany należy sprawdzić jej wykonalność zarówno w grafie jak i w tablicy.

Wymóg by macierz była stochastyczna, powoduje że wartości przepływu przez krawędzie są sumami iloczynów elementu macierzy i intensywności źródła (patrz 3.2). W efekcie pojawiają się błędy numeryczne przy próbach pomniejszania przepływu przez krawędź. Jako że przepływ równy zero powinien oznaczać usunięcie krawędzi, konieczne staje się wprowadzenie pewnego progu błędu. Jeżeli waga krawędzi ma wartość mniejszą niż próg, jest uznawana za zero, a krawędź usuwana.

Do wyszukiwania ścieżek (lub stwierdzania ich braku) zostało wykorzystane przeszukiwanie grafu w głąb (DFS), algorytm można znaleźć np. w [1]. Operacje na grafie dotyczące ścieżek wymagają uprzedniego stwierdzenia ich jednoznaczności istnienia, więc stosowanie bardziej zaawansowanych algorytmów wyszukiwania najkrótszych dróg mija się z celem.

Do testowania jednoznaczności istnienia ścieżek w grafie zastosowany został następujący algorytm:

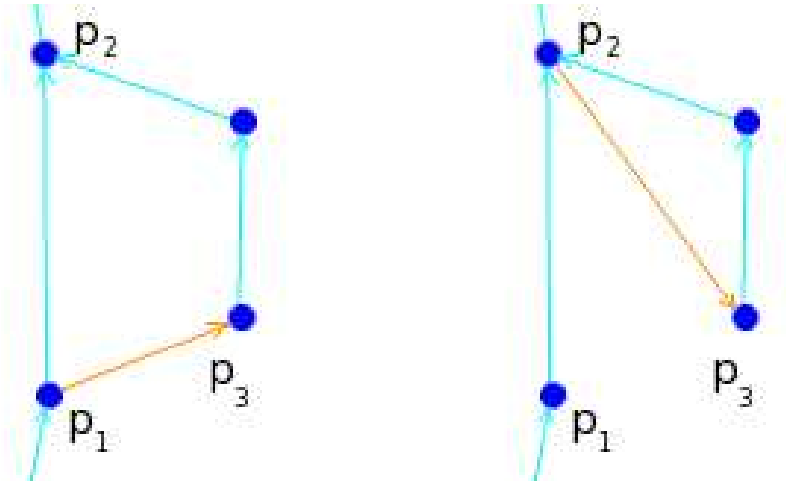
Dane: dwa wierzchołki $p_1, p_2 \in V$.
Algorytm zwróci `true` wtedy i tylko wtedy, gdy istnieje dokładnie jedna ścieżka skierowana z p_1 do p_2.

- Jeżeli $p_1 = p_2$ zwróć `false`;
- Jeżeli nie istnieje ścieżka z p_1 do p_2 zwróć `false`;
- Oznacz wszystkie wierzchołki jako nieodwiedzone i niesprawdzone;
- Oznacz p_1 jako odwiedzony, dodaj p_1 na stos s (lub kolejkę);
- Dopóki stos s jest niepusty wykonuj:
  - Zdejmij wierzchołek ze stosu, oznacz go jako p;
  - Dla każdego sąsiada v wychodzącego z p wykonaj:
    - Jeżeli v jest nieodwiedzony to oznacz v jako odwiedzony i dodaj go na stos;
    - Jeżeli v jest już odwiedzony to
      - Jeżeli v jest już sprawdzony, kontynuuj pętlę;
      - Jeżeli v nie jest sprawdzony i istnieje ścieżka z v do p_2 to zwróć `false` w przeciwnym wypadku oznacz v jako sprawdzony;
- Zwróć `true`;

3.3.7 Przesunięcia rozgałęzień

Tak jak przy innych modyfikacjach, temperatura odwrotna β jest obliczana zgodnie ze wzorami 3.3.

Przy implementacji przesunięć rozgałęzień należy zwrócić uwagę na stan sieci drogowej po wprowadzeniu zmian. W pewnych sytuacjach, operacja ta może prowadzić do powstania cyklu w grafie (przypomnijmy, że mowa tu o cyklu zorientowanym, a zatem niedopuszczonym w implementacji) — patrz rysunek 3.2. Jest to krytyczna sytuacja, która powinna zostać odrzucona, a modyfikacja wycofana. Powstanie cyklu jest możliwe przy przesuwaniu rozgałęzień wychodzących z wierzchołka w przód ścieżki, oraz przy przesuwaniu rozgałęzień wchodzących do wierzchołka wstecz ścieżki. Do wyszukiwania cyklów skierowanych można zastosować np. przeszukiwanie grafu w szerz (BFS).



(a) Sytuacja przed przesunieciem rozgałęzienia.

(b) Przepływ przez krawędź (p_1, p_3) został przesunięty do p_2.

Rysunek 3.2: Powstanie cyklu na skutek przesunięcia rozgałęzienia.

3.4 Interfejs użytkownika

3.4.1 Edytor danych wejściowych

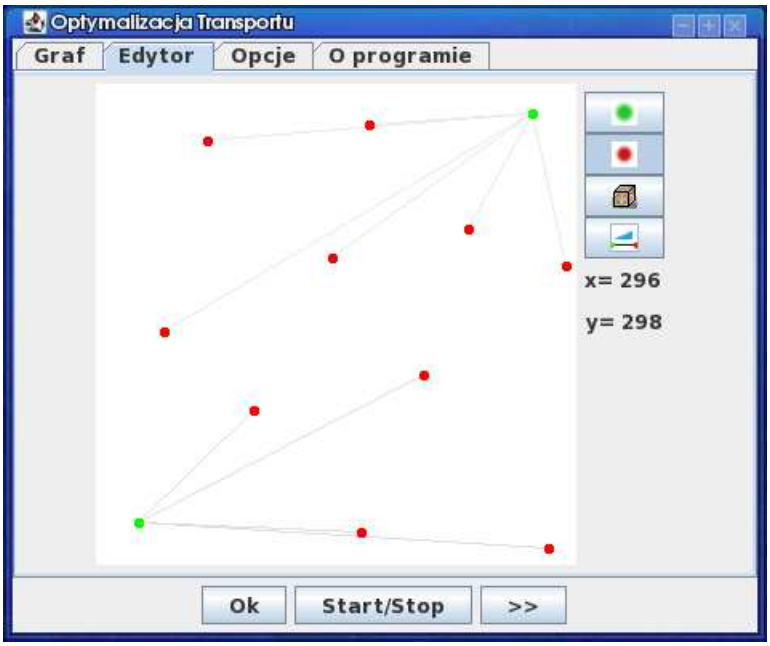
Dane wejściowe do programu można edytować w panelu dostępnym w zakładce **Edytor**. W tym panelu (rysunek 3.3) należy sprecyzować liczbę i położenie źródeł i odbiorców. Wszystkie węzły muszą leżeć w zbiorze $[0, 300) \times [0..300)$. Podaż i popyt, czyli funkcje f oraz g można edytować w osobnym oknie dialogowym, dostępnym poprzez przycisk **Ustaw transport**. Dodatkowo można również sprecyzować początkowe przepływy między źródłami i odbiorcami.

3.4.2 Podgląd bieżącego wyniku algorytmu

Zakładka **Graf** wyświetla bieżący stan sieci drogowej (rysunek 3.4).

W programie oraz w ilustracjach prezentujących wyniki przyjęta została następująca konwencja:

- źródła w grafie zostały zaznaczone kolorem zielonym,
- odbiorcy w grafie zostały oznaczone kolorem czerwonym,
- wierzchołki tranzytowe zostały oznaczone kolorem niebieskim,



Rysunek 3.3: Zrzut ekranowy panelu edycji danych wejściowych.

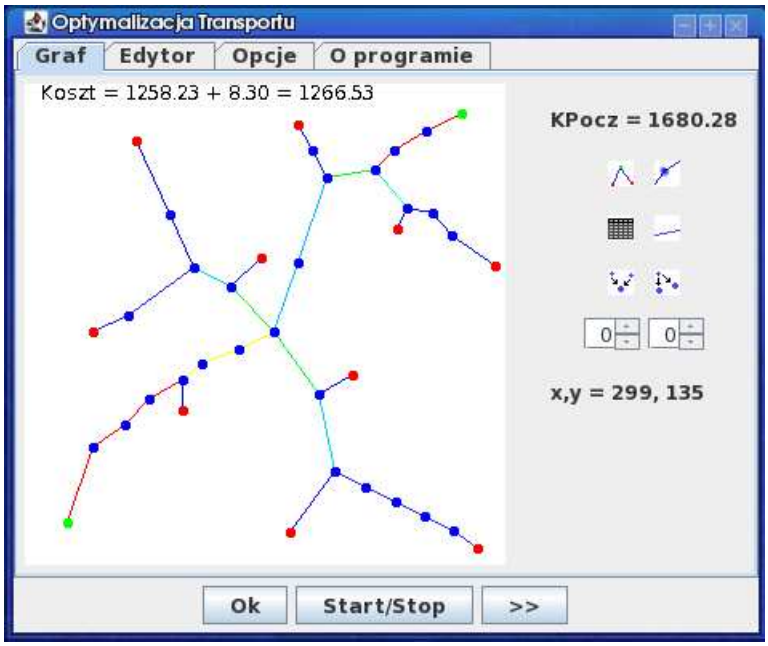
- kolor krawędzi między wierzchołkami zależy od przepływu przez daną krawędź. Kolor zmienia się stopniowo:
  - od białego do niebieskiego (przepływ 0 do 1),
  - od niebieskiego do błękitnego — cyjan (1 do 2),
  - od błękitnego do zielonego (2 do 3),
  - do zielonego do żółtego (3 do 4),
  - od żółtego do czerwonego (4 do 5),
  - od czerwonego do fioletowego — magenta (5 do 6),
  - fioletowy (przepływ przez krawędź powyżej 6).

Dostępne są opcje podglądu mapy odległości, indeksu wierzchołków i dokładnego przepływu przez krawędzie.

3.4.3 Parametry opcjonalne

W zakładce **Opcje** mogą być skonfigurowane opcjonalne parametry programu (rysunek 3.5).

- Skalar $\alpha \in [0..1)$, występujący w funkcji kosztu (1.9),



Rysunek 3.4: Zrzut ekranowy podglądu wyniku algorytmu.

- Skalary $c_1, c_2 \in [0.1, 100]$, oznaczające wagi składnika kosztu i składnika penalizującego odpowiednio (1.9),
- Maksymalna ilość iteracji przy algorytmie samoorganizacyjnym Kohonena,
- Promień zasięgu mapy odległości,
- Maksymalna zmiana przepływu pomiędzy źródłem a dwoma odbiorcami, a algorytmie oznaczona poprzez d,
- Czas odświeżania, wyniku przy ciągłej iteracji (w milisekundach),
- Aktualna wartość temperatury odwrotnej β, parametr jest wspólny dla wszystkich modyfikacji (3.3); należy pamiętać, że ten parametr stale rośnie w trakcie obliczeń,
- Lista dopuszczalnych modyfikacji grafu:
  - Przesunięcia typu Kohonena węzłów grafu,
  - Łączenie i rozłączenie wierzchołków (jako że są to zmiany przeciwne są dopuszczane razem),
  - Przesunięcia losowe wierzchołków,

Rysunek 3.5: Zrzut ekranowy zakładki konfigurowalnych parametrów.

– Modyfikacje przepływu,
– Przesunięcia rozgałęzień.

Usuwanie anomalii oraz losowe wstawianie i usuwanie węzłów zawsze są na liście operacji dopuszczalnych.

3.5 Wymagania sprzętowe

Wymagania systemowe aplikacji:

- procesor 2000 MHz lub szybszy,
- przynajmniej 512 MB pamięci operacyjnej,
- system operacyjny posiadający wsparcie Javy,
- środowisko Java SE 6 lub odpowiednik (GCJ, Java IcedTea),
- urządzenia wskazujące (myszka, klawiatura).

Rozdział 4

Omówienie wyników

Poniżej przedstawione zostaną przykładowe wyniki działania zaimplementowanego algorytmu. Charakterystyczną cechą algorytmu symulowanego wyżarzania jest zwracanie rezultatów suboptymalnych tj. dobrze przybliżających rozwiązanie optymalne, lecz w nielicznych przypadkach pokrywających się z nim. Przyjęto grupowanie ze względu na ilość źródeł w specyfikacji wejściowej.

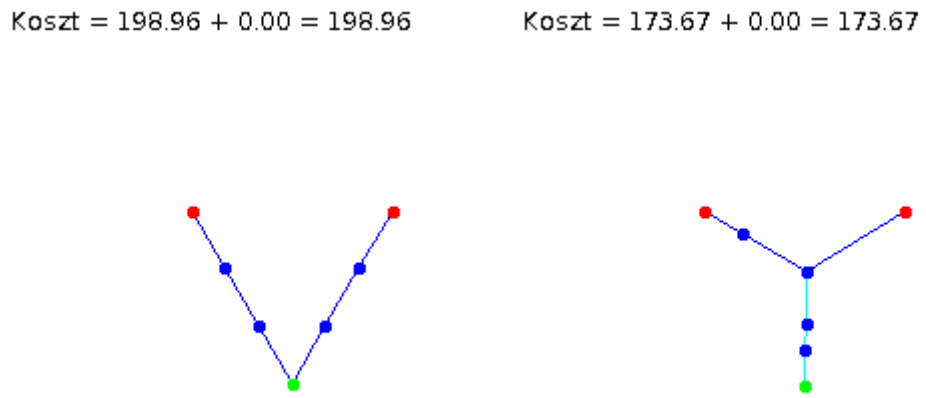
4.1 Dane z jednym źródłem

Jedno źródło jest charakterystyczne między innymi dla liści drzew (patrz [2]). Są to o tyle prostsze problemy, że mają wyspecyfikowany przydział transportu — funkcję j — pokrywa się ona z wektorem popytu odbiorców. Znacznie upraszcza to przestrzeń rozwiązań.

Jak zostało już stwierdzone, dla jednego źródła i dwóch odbiorców ustawionych w wierzchołkach trójkąta równobocznego, przy parametrze $\alpha = 0$ (patrz funkcja kosztu 1.8 oraz 1.9) otrzymujemy problem Gaussa.

Wynik działania zaprezentowany jest na rysunku 4.1. Przy odległości między wierzchołkami równej $a = 100$, koszt rozwiązania początkowego wynosi $2a = 200$. Koszt optymalnego rozwiązania wynosi $3 \cdot \frac{\sqrt{3}}{3} a \simeq 173$. Otrzymane rozwiązane ma koszt 173.67, ale należy zauważyć, że wierzchołki nie tworzą dokładnego trójkąta równobocznego — są one przybliżane współrzędnymi całkowitymi.

Przy większej ilości odbiorców dokładne obliczenie kształtu optymalnej sieci i optymalnego kosztu staje się skomplikowane. Nie ma zatem obiektywnych kryteriów oceny jakości rozwiązania zwracanego przez algorytm. Posiłkować się można wyglądem rozwiązania i ewentualnie analogią do faktycznych kształtów unerwień liści.



(a) Dane wejściowe. (b) Wynik (ok. 2000 iteracji).

Rysunek 4.1: Problem Gaussa: parametry $\alpha = 0$, $c_1 = 1$, $c_2 = 100$, odległość między wierzchołkami wynosi ok 100 pikseli.

Poniżej zamieszczone zostały przykłady działania algorytmu dla jednego źródła oraz kilku odbiorców ułożonych w kształt liścia (rysunek 4.2). Każdy odbiorca przyjmuje po jednej jednostce transportu. Intensywność źródła pokrywa się z sumą popytu wszystkich odbiorców. Parametr α nie jest już zerowy, co oznacza że koszt krawędzi jest istotnie zależny od przepływu.

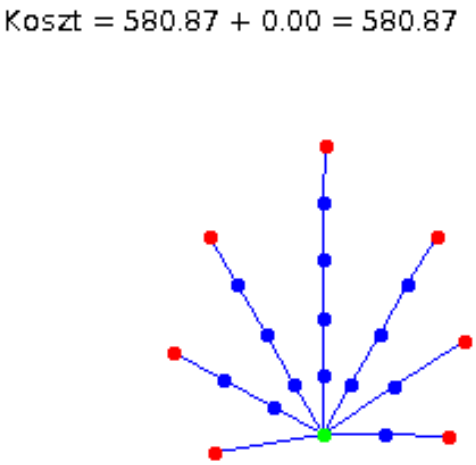
Uzyskane w wyniku działania algorytmu rozwiązania są pozbawione pętli i przecięć krawędzi. Można doszukać się pewnych analogii do sieci nerwowej prawdziwych liści.

Można można zaobserwować tendencję do silniejszego nasycania odbiorców leżących bliżej źródła. Jest to spowodowane, tym że odległość do odbiorcy ma wpływ na koszt. Okazuje się, że lepiej jest część towaru dostarczać bliżej, za cenę zwiększenia wartości składnika penalizującego. Problem ten można rozwiązać poprzez zwiększenie wagi dla kary c_2, lub alternatywnie poprzez zmniejszenie wagi kosztu grafu c_1. W prezentowanych przykładach różnica między wpływem oczekiwanym, a faktycznym nie przekracza 5%.

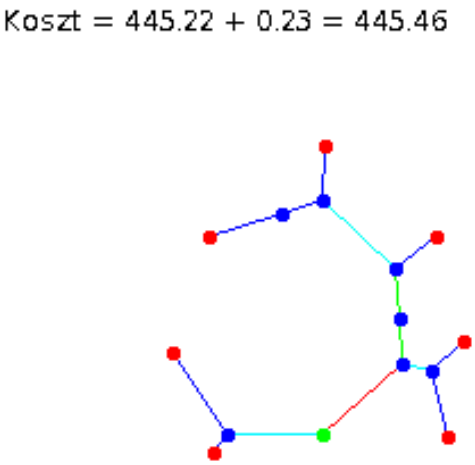
Czas obliczeń wynosi około 3 do 5 minut na komputerze z procesorem jednordzeniowym 2800 MHz, zależnie od parametrów wejściowych między innymi temperatury odwrotnej, wag dla funkcji kosztu.

4.2 Dane z wieloma źródłami

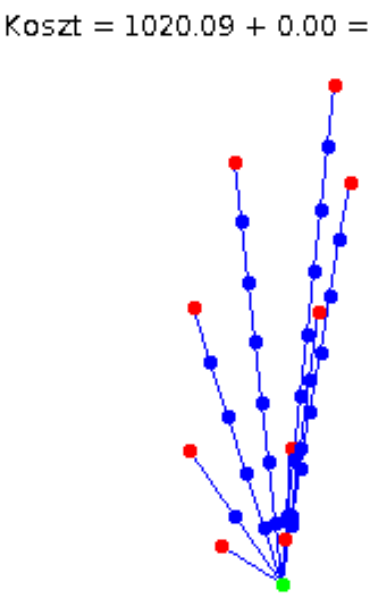
Dane, w których istnieje wiele źródeł, są trudniejsze do rozwiązania. Funkcja j opisująca przepływ pomiędzy źródłami, a odbiorcami będzie się zmie-



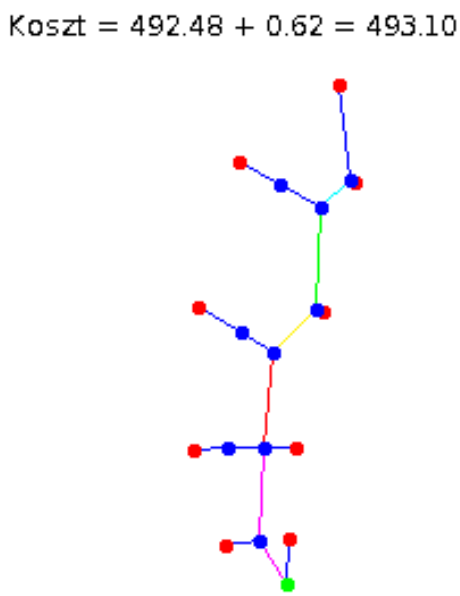
(a) Dane wejściowe (7 odbiorców).



(b) Wynik.



(c) Dane wejściowe (9 odbiorców).



(d) Wynik.

Rysunek 4.2: Problemy „liściowe”: parametry $\alpha = 0.25$, $c_1 = 1$, $c_2 = 100$.

niała w trakcie obliczeń. Dodatkowo przy wielu źródłach rozwiązania mogą charakteryzować się, zarówno obecnością rozgałęzień Y-kształtnych jak i λ-kształtnych.

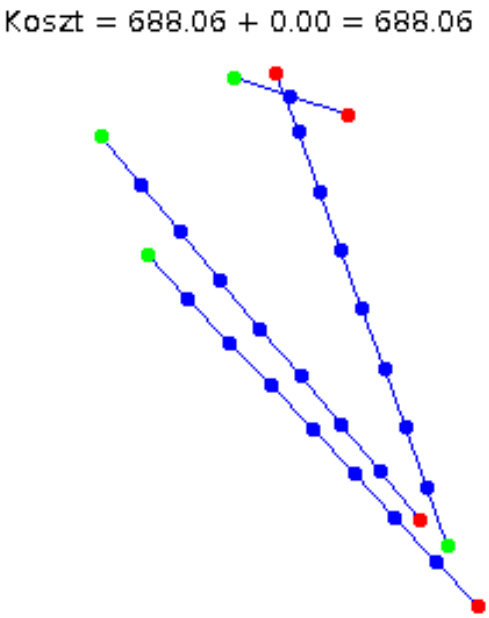
Należy zwrócić uwagę na dobór skalarów c_1 i c_2 adekwatnych do kosztu rozwiązania. Przy wysokim skalarze dla składnika penalizującego (c_2) algorytm kurczowo trzyma się pierwszego rozwiązania, które odpowiada wyspecyfikowanemu w danych wejściowych popytowi. Próby poprawy sieci powodują gwałtowny skok funkcji kosztu i są z reguły odrzucane.

Dlatego dobór odpowiedniego rozwiązania początkowego ma wpływ szybkość działania. Dobrze spisują się te, w których źródła transportują towar do bliskich odbiorców. Charakteryzują się one małą ilością węzłów tranzytowych i krawędzi, więc przyśpieszają obliczenia. Dodatkowo, z punktu widzenia algorytmu, znacznie łatwiejsze jest stworzenie nowego połączenia między źródłem, a odbiorcą, niż wyzerowanie istniejącego. Jest to spowodowane wymogiem istnienia dokładnie jednej ścieżki, przy modyfikacjach przepływu.

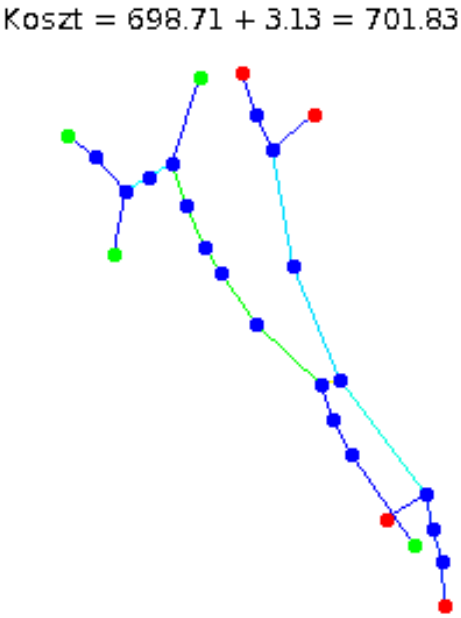
Innym rozwiązaniem jest zmiana wagi parametru c_2 w trakcie obliczeń. Przy niskiej wartości algorytm wykazuje tendencje do łączenia źródeł z najbliższymi odbiorcami. Zwiększanie tej wagi powoduje stopniowy wzrost składnika penalizującego. Mając połączenia do bliskich odbiorców można łatwo stworzyć odnogę do węzła, który nie został jeszcze nasycony, bez generowania chaosu w strukturze grafu. Spowoduje do spadek wartości składnika penalizującego, ale i jednoczesny wzrost składnika kosztu samego grafu. Kontynuując zwiększanie c_2 można uzyskać rozwiązanie nasycające wszystkich odbiorców (rysunek 4.3).

Na rysunku 4.4 zamieszczone są wyniki działania algorytmu dla większej ilości węzłów. Jak widać udało się uzyskać rozwiązanie przy wyraźnie odseparowanych od siebie zbiorach źródeł i odbiorców jak również w przypadku gdy są one wymieszane. Ponadto algorytm poradził sobie z dość kłopotliwym rozwiązaniem początkowym, które w obu przypadkach jest dość odległe od optymalnego. W obu przykładach obliczenia zostały rozpoczęte przy wartościach parametrów równych $c_1 = c_2 = 1$, wraz z postępem obliczeń c_2 był zwiększany do wartości 100.

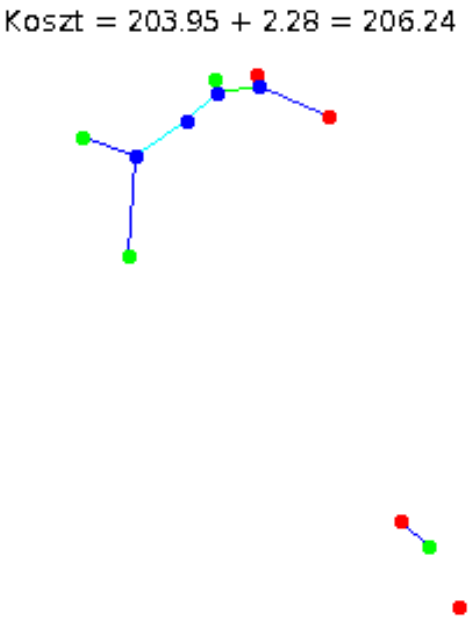
Obliczenia dla ośmiu węzłów wejściowych trwają około pięciu minut, dla szesnastu węzłów czas działania rośnie do ok. dziesięciu – piętnastu minut. Udaje się uzyskać prędkość na poziomie ok. 20 – 40 iteracji na sekundę. Długość obliczeń silnie zależy od startowej temperatury odwrotnej. Niskie jej wartości powodują akceptowanie każdych zmian i co za tym idzie tworzenie się wielu chaotycznych ścierzek, których usunięcie może długo trwać.



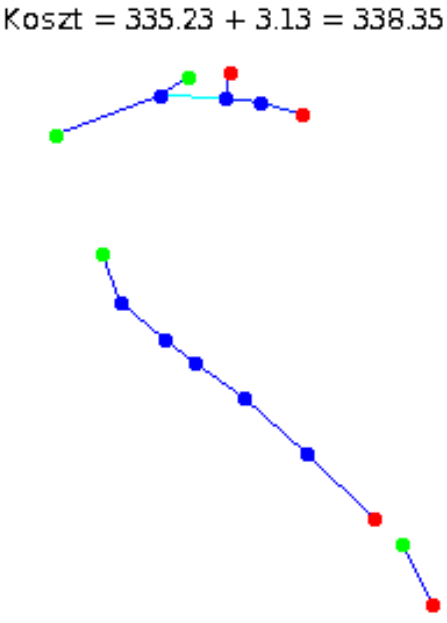
(a) Dane wejściowe.



(b) Wynik po ok 6000 iteracji, wysoka wartość skalaru $c_2 = 100$.

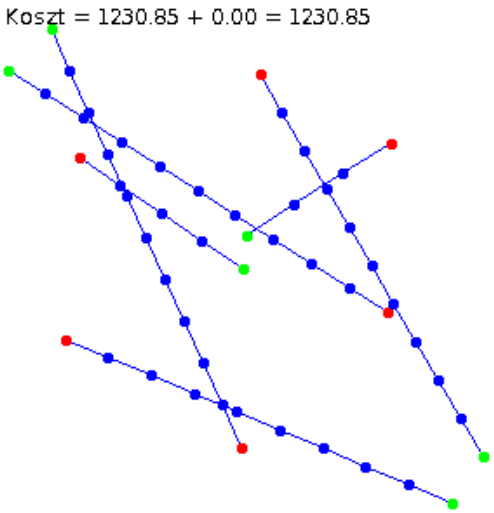


(c) Wynik po ok 1500 iteracji przy $c_2 = 1$.

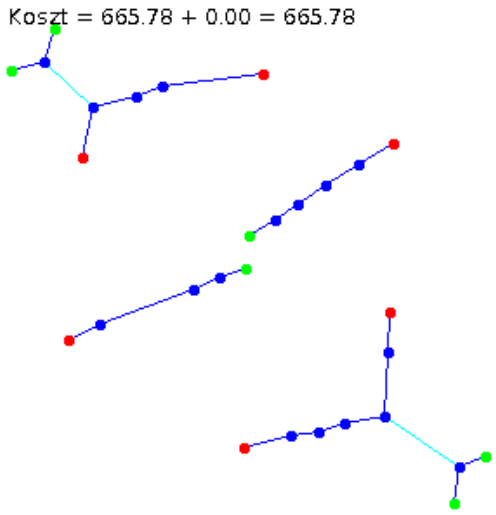


(d) Wynik (c) po stopniowym zwiększaniu c_2 od 1 do 100, ok 6000 iteracji.

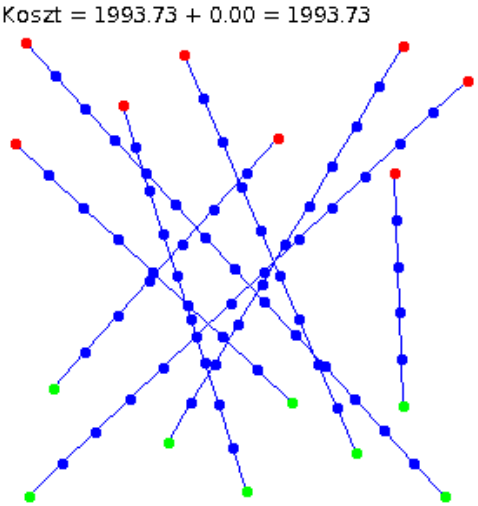
Rysunek 4.3: Grafy z czterema źródłami i czterema odbiorcami: parametry $\alpha = 0.25$, $c_1 = 1$.



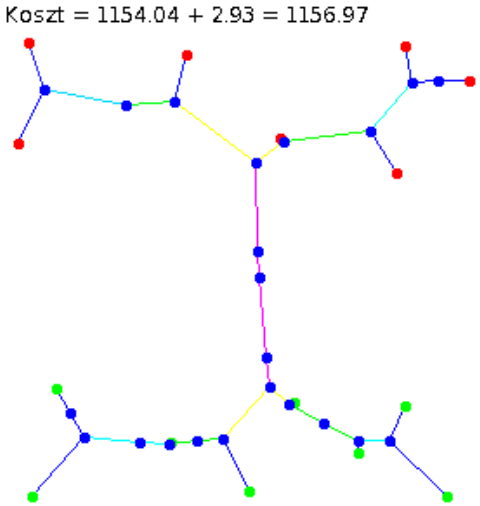
(a) Dane wejściowe, po sześć źródeł i odbiorców.



(b) Wynik po ok 7000 iteracji.



(c) Dane wejściowe, po osiem źródeł i odbiorców, wyraźna separacja węzłów jednego typu.



(d) Wynik po ok 9000 iteracji.

Rysunek 4.4: Grafy dla 12 i 16 węzłów.

Bibliografia

[1] M. M. Sysło, N. Deo, J. Kowalik, *Discrete Optimization Algorithms With Pascal Programs*, Dover Publications, Inc. Mineola, New York

[2] Quinglan Xia, *The Formation of the Tree Leaf*

[3] Quinglan Xia, *Optimal Paths Related to Transport Problems*

[4] Vijay V. Vazirani, *Algorytmy aproksymacyjne*, Wydawnictwa Naukowo-Techniczne, Warszawa 2005.

[5] Pierre Peretto, *An Introduction to the Modelling of Neural Networks*, Cambridge University Press, Cambridge 1994.

[6] Raul Rojas, *Neural Networks : a Systematic Introduction*, Springer, Berlin 1996.

[7] Jacek Mańdziuk, *Sieci Neuronowe Typu Hopfielda. Teoria i Przykłady Zastosowań*, Akademica Oficyna Wydawnicza EXIT, Warszawa 2000.