Biuletyn WAT
Vol. LX, Nr 4, 2011

Metody rozwiązywania problemu najkrótszych dróg wierzchołkowo rozłącznych przechodzących przez wybrane wierzchołki w sieciach o strukturze kraty

ZBIGNIEW TARAPATA, STEFAN WROCŁAWSKI

Wojskowa Akademia Techniczna, Wydział Cybernetyki, Instytut Systemów Informatycznych,
00-908 Warszawa, ul. S. Kaliskiego 2,
zbigniew.tarapata@wat.edu.pl, stefan.wroclawski@gmail.com

Streszczenie. W pracy zaprezentowano modele i metody służące do rozwiązywania problemu wyznaczania *K* dróg wierzchołkowo rozłącznych przechodzących przez wybrane wierzchołki, o najmniejszym sumarycznym koszcie w sieci prostokątnej (tzw. kracie) opartej o graf *G*. Zdefiniowano problem jako zadanie optymalizacji liniowej ciągłej oraz przedstawiono dwie metody przybliżone jego rozwiązania: metodę SGDP (bazującą na pewnej iteracyjnej procedurze wyznaczania dróg najkrótszych w podgrafach grafu *G*) oraz modyfikację metody Edmondsa-Karpa rozwiązywania problemu wyznaczania przepływu zaspokajającego o minimalnym koszcie. Przeprowadzono analizę ich złożoności oraz dokonano porównania jakości obu metod na podstawie eksperymentalnych wyników.
Słowa kluczowe: drogi rozłączne wierzchołkowo; symulacja pola walki; generowanie podgrafów; planowanie przemieszczania

1. Wprowadzenie

Problem dróg rozłącznych jest znanym problemem optymalizacyjnym w sieciach. Obejmuje on następujące zagadnienia: planowanie manewrów oddziałów wojskowych i planowanie przemieszczania obiektów (np. pojazdów) [32], harmonogramowanie zadań w równoległych i rozproszonych systemach obliczeń [26], problem kurierów, problem konstruowania niezawodnej sieci [5] czy problem routingu w sieciach telekomunikacyjnych (w szczególności optycznych) [1], [2], [14], [17], [20] etc. Prezentowanym problemem oraz metodami jego rozwiązania mocno zainteresowany jest również przemysł związany z projektowaniem i produkcją układów elektronicznych

bardzo dużej skali integracji (VLSI — ang. *Very Large Scale Integration*) [1]. Kluczowym zagadnieniem jest znalezienie takich ścieżek na płytkach układów, aby żadne z nich się nie przecinały, a ponadto aby były one jak najkrótsze.

Problem przemieszczania obiektów jest jednym z podstawowych problemów w symulatorach pola walki i jest związany z przegrupowaniem jednostek wojskowych na polu walki zarówno w czasie działań bojowych, jak i na etapie ich planowania. Przemieszczanie obiektów jest bardzo ważne z punktu widzenia złożonych symulacji, ponieważ może wpływać na dokładność, adekwatność, efektywność i inne charakterystyki tych systemów. W przypadku systemów DIS (ang. *Distributed Interactive Simulation*), system ma za zadanie symulować procesy i obiekty pola walki w celu szkolenia personelu militarnego, czyli prowadzenia tzw. komputerowo wspomaganych ćwiczeń CAX (ang. *Computer Assisted Exercises*). Jedną z technik służących do zapewnienia działań wojsk strony przeciwnej na wirtualnym polu walki jest użycie komputera do generowania i kontrolowania wielu obiektów symulacyjnych. Systemy takie nazywa się CGF (ang. *Computer Generated Forces*) [21] lub SAF (ang. *Semi-Automated Forces*). Nieodłączną częścią tego typu systemów są moduły planowania tras przemieszczania bazujące na modelach terenu i wykorzystujące przetworzone z nich informacje. W [19] opisany został moduł tego typu stosowany w systemie ModSAF. W [4] autorzy opisują system mieszany planowania tras („po drogach i na przełaj") z użyciem systemów typu GIS w symulatorach. Opisują m.in. tzw. hierarchiczny system planowania dróg jako część prognostycznego systemu analiz taktycznych PIMTAS (ang. *Predictive Intelligence Military Tactical Analysis System*). Moduł planowania tras w systemie CCTT (*Close Combat Tactical Trainer*) opisany został w [8], natomiast autorzy [16] opisali analizator przemieszczania TMA (ang. *Tactical Movement Analyzer*). System ten wykorzystuje mapy cyfrowe, zdjęcia satelitarne, warunki pogodowe i rodzaj pojazdu, aby określić przejezdność kwadratu terenu. Moduł odpowiedzialny za planowanie przemieszczania znajduje się również w polskim systemie wspomagania szkolenia operacyjnego wojsk „Złocień", który został wytworzony w Wydziale Cybernetyki Wojskowej Akademii Technicznej [3]. Przegląd możliwych rozwiązań w tym zakresie został przedstawiony m.in. w [28]. W pracach [26], [32] i [33] przedstawiono kilka problemów harmonogramowania przemieszczania oraz jego synchronizacji dla wielu obiektów oraz algorytmy rozwiązujące ten problem wraz z teoretyczną i eksperymentalną analizą w zastosowaniach militarnych. W pracy [34] porusza się ważny problem wielorozdzielczego planowania i symulacji przemieszczania.

Problem optymalizacyjny znajdowania K najkrótszych dróg rozłącznych dla $K > 1$ pomiędzy K parami różnych wierzchołków jest NP trudny (nawet dla $K = 2$) [12], [15], [20], również jeśli na graf będzie narzucone wymaganie planarności [13]. Problem znalezienia dwóch lub więcej rozłącznych dróg pomiędzy ustalonymi parami wierzchołków końcowych jest intensywnie studiowany. Pierwsze znaczące wyniki zostały uzyskane przez Suurballe [24]. Zaprezentowany przez

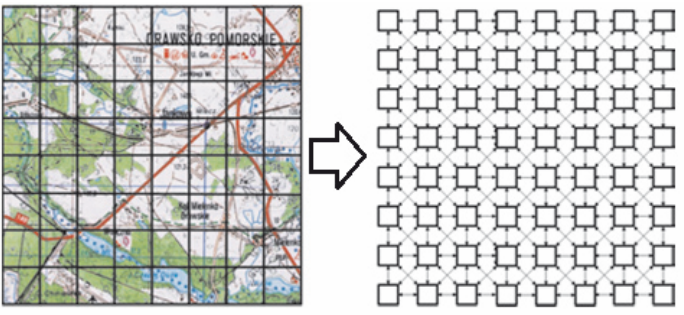
niego algorytm dla jednego źródła i jednego wierzchołka docelowego ma złożoność $O(A\log_{(1+A/V)}V)$, gdzie V to liczba wierzchołków sieci, A — liczba łuków w sieci. Metoda ta rozwiązuje omawiany problem jako specjalny przypadek problemu przepływu o minimalnym koszcie i wykorzystuje dwie efektywne implementacje algorytmu Dijkstry do znalezienia najkrótszej drogi z pojedynczego wierzchołka. Efektywny algorytm znajdujący drogi rozłączne z pojedynczego źródła do pozostałych wierzchołków sieci został przedstawiony w [25]. Pary dróg rozłącznych są wyznaczane tu za pomocą algorytmu zbliżonego do algorytmu Dijkstry o złożoności $O(A\log_{(1+A/V)}V)$. Perl i Shiloach [11] studiowali złożoność poszukiwań dwóch dróg rozłącznych pomiędzy dwoma różnymi wierzchołkami początkowymi i dwoma różnymi wierzchołkami końcowymi w skierowanych grafach acyklicznych (DAG). Zaproponowali algorytm, który daje się w łatwy sposób uogólnić do znajdowania najkrótszych par dróg (mierzonych jako sumaryczna długość dróg) lub do znajdowania pęku d dróg rozłącznych pomiędzy określonymi punktami końcowymi. W tym drugim przypadku złożoność obliczeniowa wynosiłaby $O(AV^{d-1})$. Eppstein w [11] rozważał problem poszukiwania par dróg wierzchołkowo rozłącznych w DAG, zarówno łącząc dwa określone wierzchołki z ich wspólnym przodkiem, albo łącząc dwie pary wierzchołków początkowych i końcowych. Pokazał, jak znaleźć K par dróg o najkrótszej łącznej długości w czasie $O(AV + K)$ oraz jak zliczyć wszystkie takie pary dróg, wykorzystując $O(AV)$ operacji arytmetycznych. Wyniki te mogą być rozszerzone na znajdowanie lub zliczanie pęków złożonych z d rozłącznych dróg w czasie $O(AV^{d-1}+K)$ lub $O(AV^{d-1})$. Li i inni [18] podali pseudowielomianowy algorytm dla optymalizacyjnej wersji problemu dla dwóch dróg rozłącznych, przy czym minimalizowana jest długość dłuższej z nich. W pracy [14] poruszany był problem poszukiwania dwóch wierzchołkowo rozłącznych dróg o minimalnym sumarycznym koszcie w sieciach, w których koszt przypisany był do każdego łuku lub krawędzi oraz koszt przejścia przypisany był do każdego wierzchołka tej sieci. Koszt przejścia miał oznaczać opóźnienia przesyłu związane ze zmianą technologii przesyłu po obu stronach węzła sieci. W pracy [31] przedstawiono opis sposobu wykorzystania metod wyznaczania przepływu w sieciach do rozwiązania specyficznego problemu planowania manewru wojsk, wykorzystując zmodyfikowany algorytm Busackera-Gowena [7] oraz pewną procedurę konstrukcji sieci przepływowej. Dobre studium bardzo ważnego problemu poszukiwania dróg rozłącznych w grafach planarnych zaprezentowano w [22]. Bardzo ciekawe podejście do problemu czasowo zależnych najkrótszych par dróg rozłącznych zostało przedstawione w [23]. W [26] zaproponowano nowe podejście do problemu K dróg rozłącznych: bazuje ono na budowaniu, poczynając od sieci początkowej, tzw. K-wierzchołków (K-wymiarowy wektor wierzchołków sieci), K-łuków i „wirtualnej" K-sieci, i poszukiwaniu w takiej K-sieci najkrótszej K-drogi (K-wymiarowy wektor dróg prostych), używając algorytmu podobnego do algorytmu Dijkstry.

W niniejszej pracy przedstawiono algorytm bazujący na generowaniu podgrafów (*Subgraphs Generating-based Disjoint Paths* — SGDP) oraz modyfikację metody Edmondsa-Karpa rozwiązującą problem *K* najkrótszych dróg rozłącznych wierzchołkowo, przechodzących przez wybrane wierzchołki pośrednie. Metody te bazują na założeniu, że topologia sieci reprezentowana jest przez graf prostokątny, który jest typową reprezentacją terenu, np. w symulatorach pola walki. W rozdziale 2 zdefiniowano problem wyznaczania *K* dróg rozłącznych między *K* wektorami wierzchołków pośrednich (w skrajnym przypadku — między *K* parami wierzchołków), rozdziały 3 i 4 zawierają opisy wspomnianych metod, w rozdziale 5 przedstawiono wybrane doświadczalne wyniki i dokonano porównania omawianych metod. Pracę kończą wnioski oraz opis możliwych kierunków rozwoju prezentowanych metod.

2. Definicja problemu

Sieć oparta na grafie prostokątnym może modelować m.in. regularną siatkę kwadratowych obszarów terenu używaną do planowania przemieszczania „na przełaj" (rys. 1). Siatka ta dzieli teren na identycznego rozmiaru kwadratowe obszary, zaś każdy z nich jest homogeniczny z punktu widzenia charakterystyki terenowej (współczynnik spowolnienia, zalesienie, stopień widoczności etc.). Nie ma możliwości przemieszczania się po obszarach, które nie są do siebie geograficznie przyległe — brana pod uwagę jest jedynie przyległość w poziomie, pionie oraz na ukos — każdy obszar może mieć maksymalnie 8 sąsiadów (patrz rys. 1). Wprowadzony jest również koszt przejścia między sąsiednimi (i tylko sąsiednimi) obszarami.

Taka struktura może być reprezentowana przez „prostokątny" digraf $G = \langle V_G, A_G \rangle$ nazywany również kratą, gdzie V_G — zbiór wierzchołków grafu (wierzchołek odpowiada środkowi kwadratowego obszaru), A_G — zbiór łuków grafu, $A_G \subset V_G \times V_G$,



Rys. 1. Siatka prostokątna oraz odpowiadający jej graf typu krata

$A = |A_G|$. Łuki istnieją jedynie pomiędzy wierzchołkami odpowiadającymi sąsiadującym obszarom. Funkcja kosztu opisana na łukach ma charakter addytywny, o nieujemnych wartościach.

Niech $V_G = \{1, \ldots, V\}$, tzn. każdy wierzchołek grafu G utożsamiany będzie z jego numerem. Niech M będzie liczbą odcinków, z jakich ma składać się każda droga ($M = N$-1, N oznacza liczbę wierzchołków pośrednich łącznie z początkowym i końcowym), K oznacza liczbę obiektów, dla których wyznaczać będziemy drogi. Wektor wierzchołków przez które musi przechodzić droga dla k-tego obiektu oznaczymy $v_k = \langle v_{1k}, v_{2k}; \ldots, v_{Nk} \rangle$, gdzie v_{1k} jest wierzchołkiem początkowym, a v_{Nk} — końcowym drogi dla k-tego obiektu, pozostałe wierzchołki są pośrednimi w drodze. Niech $A = [a_{imk}]_{V \times M \times K}$ będzie macierzą wierzchołków początkowych, pośrednich oraz końcowych dla każdego z obiektów: $a_{imk} = 1$, gdy i-ty wierzchołek jest początkiem dla m-tego odcinka drogi k-tego obiektu, $a_{imk} = -1$, jeśli i-ty wierzchołek jest końcem dla m-tego odcinka drogi k-tego obiektu, $a_{imk} = 0$ w pozostałych przypadkach. Dodatkowo, następujące warunki muszą być spełnione:

- $a_{i1k} = 1 \Leftrightarrow i = v_{1k}$, co oznacza, że wierzchołek v_{1k} musi być wierzchołkiem początkowym pierwszego odcinka drogi dla k-tego obiektu;
- $a_{i1k} = -1 \Leftrightarrow i = v_{2k}$, co oznacza, że pierwszy wierzchołek pośredni v_{2k} dla k-tego obiektu jest wierzchołkiem końcowym dla pierwszego odcinka drogi dla tego obiektu;
- $a_{iMk} = 1 \Leftrightarrow i = v_{Mk}$, co oznacza, że ostatni wierzchołek pośredni v_{Mk} dla k-tego obiektu jest wierzchołkiem początkowym dla ostatniego odcinka drogi dla tego obiektu;
- $a_{iMk} = -1 \Leftrightarrow i = v_{Nk}$, co oznacza, że v_{Nk} jest wierzchołkiem końcowym w ostatnim odcinku drogi dla k-tego obiektu;
- $\forall_{m \in \{1,\ldots,M-1\}} a_{imk} = -1 \Rightarrow a_{i(m+1)k} = 1$, co oznacza, że wierzchołek końcowy m-tego odcinka drogi dla k-tego obiektu jest równocześnie wierzchołkiem początkowym dla (m+1)-tego odcinka drogi dla tego obiektu.

Ponadto niech $H = [h_{ik}]_{V \times K}$ będzie macierzą wierzchołków (generującą podgraf grafu G), które mogą być brane pod uwagę podczas wyznaczania dróg dla każdego obiektu: $h_{ik} = 1$, jeśli i-ty wierzchołek może być wzięty pod uwagę podczas ustalania dróg dla k-tego obiektu; $h_{ik} = 0$ w przeciwnym wypadku. Niech $OUT = [out_{ij}]_{V \times A}$ będzie binarną macierzą zawierającą informacje o łukach wychodzących z wierzchołków grafu G:$out_{ij} = 1$, jeśli j-ty łuk „wychodzi" z wierzchołka i-tego, $out_{ij} = 0$ w przeciwnym przypadku; $IN = [in_{ij}]_{V \times A}$ będzie binarną macierzą przechowującą informacje o łukach wchodzących do wierzchołków z grafu G:$in_{ij} = 1$, jeśli j-ty łuk „wchodzi" do i-tego wierzchołka, $in_{ij} = 0$ w przeciwnym przypadku. Niech $D = [d_j]_A$ będzie wektorem nieujemnych kosztów przejścia łuków, $X = [x_{jmk}]_{A \times M \times K}$ — macierz zmiennych decyzyjnych: $x_{jmk} = 1$ jeśli j-ty łuk z grafu G należy do m-tego odcinka drogi dla k-tego obiektu, $x_{jmk} = 0$ w przeciwnym przypadku.

Zadanie wyznaczania K rozłącznych dróg przechodzących przez wybrane wierzchołki sformułujemy następująco: wyznaczyć macierz X tak, aby

$$\sum_{j=1}^{A}\sum_{m=1}^{M}\sum_{k=1}^{K} d_j x_{jmk} \to \min \tag{1}$$

przy następujących ograniczeniach:

$$\sum_{j=1}^{A}\left(out_{ij} - ini_j\right)x_{jmk} = a_{imk}, \quad i=\overline{1,V}; m=\overline{1,M}; k=\overline{1,K} \tag{2}$$

$$\sum_{j=1}^{A}\sum_{m=1}^{M}\sum_{k=1}^{K} out_{ij} x_{jmk} \le 1, \quad i=\overline{1,V}, \tag{3}$$

$$\sum_{j=1}^{A}\sum_{m=1}^{M}\sum_{k=1}^{K} in_{ij} x_{jmk} \le 1, \quad i=\overline{1,V}, \tag{4}$$

$$\sum_{j=1}^{A}\sum_{m=1}^{M} out_{ij} x_{jmk} \le h_{ik}, \quad i=\overline{1,V}; k=\overline{1,K}, \tag{5}$$

$$\sum_{j=1}^{A}\sum_{m=1}^{M} in_{ij} x_{jmk} \le h_{ik}, \quad i=\overline{1,V}; k=\overline{1,K}, \tag{6}$$

$$x_{jmk} \in \{0,1\}, \quad j=\overline{1,A}; m=\overline{1,M}; k=\overline{1,K}. \tag{7}$$

Poszczególne ograniczenia mają następującą interpretację:

(2): dla każdego wierzchołka (wyłączając startowy i końcowy), dla każdego obiektu oraz dla każdego odcinka drogi liczba łuków wychodzących z wierzchołka i liczba łuków wchodzących do wierzchołka, które zostały wybrane do drogi jest taka sama (następne dwa ograniczenia zapewniają, ze wartość ta jest ≤ 1). Dla wierzchołka początkowego różnica ta jest równa 1 (tylko jeden odcinek drogi może zaczynać się w danym wierzchołku startowym), a dla wierzchołka końcowego równa –1 (tylko jeden odcinek drogi może kończyć się w danym wierzchołku końcowym).

(3) i (4): dla każdego wierzchołka co najwyżej jeden łuk wychodzący (wchodzący) z (do) niego może należeć do jakiejkolwiek drogi.

(5) i (6): tylko wierzchołki dozwolone należą do drogi dla k-tego obiektu.

Można zauważyć, że macierz współczynników ograniczeń, zbudowana na podstawie lewych stron ograniczeń (2)-(6), jest całkowicie unimodularna, a prawe strony ograniczeń, elementy a_{imk} oraz h_{ik} są całkowitoliczbowe. Możemy zatem zastąpić ograniczenie $x_{jmk} \in \{0,1\}$ ograniczeniem $x_{jmk} \ge 0$ (otrzymując ciągły problem programowania liniowego zamiast binarny), gdyż stosując np. algorytm

simpleks, każde rozwiązanie dopuszczalne będzie całkowitoliczbowe (w tym przypadku $x_{jmk} \in \{0,1\}$).

W powyższym problemie optymalizacyjnym zdefiniowano *AMK* zmiennych decyzyjnych oraz *V*(*MK*+2*K*+2) ograniczeń (wyłączając ograniczenie (7)). W pracach [29], [30] zdefiniowano podobny problem z wykorzystaniem binarnej macierzy incydencji grafu (zamiast macierzy *IN* i *OUT*, jak w tym przypadku), ale nie zmienia to w znaczący sposób złożoności tak sformułowanego problemu. W obu tych przypadkach rozwiązanie problemu (1)-(7) standardowymi metodami rozwiązywania zadań programowania liniowego jest bardzo nieefektywne (zwłaszcza dla dużych sieci), dlatego też w rozdziałach 3 i 4 zaprezentowano przybliżone algorytmy rozwiązania problemu.

3. Opis metody SGDP

Jedną z metod rozwiązania problemu (1)-(7) jest metoda SGDP (*Subgraphs Generating-based Disjoint Paths*), w szczegółach zaprezentowana w [32]. Znajduje ona pęk *K* najkrótszych wierzchołkowo rozłącznych dróg przechodzących przez ustalone różne wierzchołki pośrednie, dla każdego $k = 1,\ldots, K$. Jej ideą jest generowanie tzw. pasa, czyli podgrafu G'_{mk} grafu wejściowego *G* dla każdej pary wierzchołków początkowego i końcowego *m*-tego odcinka *k*-tego obiektu, bazując na założeniu, że teren reprezentowany jest jako sieć prostokątna. W podgrafie wyszukiwana jest najkrótsza droga przy użyciu algorytmu Dijkstry [9], a następnie usuwana z grafu *G*, aby zapewnić, że żadna inna droga już nie skorzysta z wierzchołków i łuków należących do wyznaczonej poprzednio drogi. W przypadku gdy nie zostanie znalezione rozwiązanie dopuszczalne (czyli *K* dróg rozłącznych wierzchołkowo), albo w celu poprawienia rozwiązania, algorytm w kolejnych iteracjach ponawia wyszukiwanie, modyfikując kolejność lub szerokość pasów według procedury opisanej w dalszej części rozdziału.

Dla *m*-tego odcinka *k*-tego obiektu generowany jest oddzielny podgraf w następujący sposób: $G'_{mk} = \langle V_{G_{mk}}, A_{G_{mk}} \rangle$, gdzie $V_{G_{mk}}$ — jest zbiorem wierzchołków podgrafu, a $A_{G_{mk}}$ zbiorem łuków pomiędzy wierzchołkami należącymi do $V_{G_{mk}}$. Jeśli sw_k jest szerokością pasa (identyczną dla wszystkich odcinków tego samego obiektu), w którym dozwolona jest droga dla *k*-tego obiektu, to należy wyznaczyć prostą przechodzącą przez wierzchołek początkowy v_{mk} i końcowy $v_{(m+1)k}$ (każdy z wierzchołków musi mieć ustalone współrzędne, czy to geograficzne, czy względne w sieci). Do pasa, czyli podgrafu G'_{mk}, należy zaliczyć wszystkie te wierzchołki grafu *G*, które są odległe (w sensie współrzędnych, metryka euklidesowa) od dowolnego punktu leżącego na wyznaczonej prostej maksymalnie o $sw_k/2$ (patrz rys. 2). Można to wykonać, posługując się następującą zależnością:

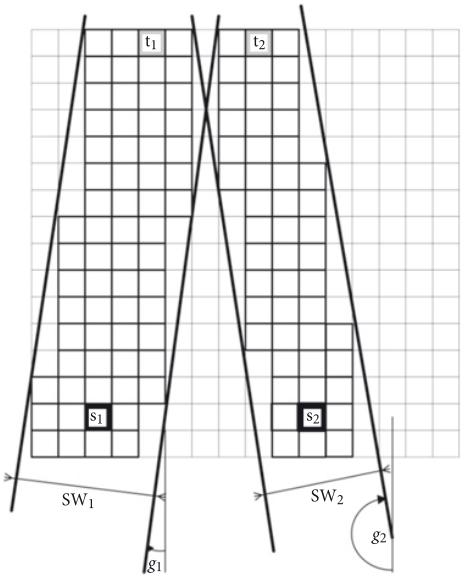
$$V_{G_{mk}} = \{v \in V_G :$$

$$x(v_{mk}) + \tan g_k \cdot (y(v) - y(v_{mk})) - \frac{sw_k}{\cos g_k}$$

$$\leq x(v) \leq$$

$$x(v_{mk}) + \tan g_k \cdot (y(v) - y(v_{mk})) + \frac{sw_k}{\cos g_k}\}$$

$$A_{G_{mk}} = \{(v, v') \in V_{G_{mk}} \times V_{G_{mk}} : (v, v') \in A_G\}.$$



Rys. 2. Idea wycinania podgrafów (pasów): s_1 i t_1 to wierzchołki terminalne m-tego odcinka k-tego obiektu $s_1 = v_{mk}$, $t_1 = v_{(m+1)k}$, zaś s_2 i t_2 to wierzchołki terminalne m-tego odcinka (k+1)-tego obiektu $s_2 = v_{m(k+1)}$, $t_2 = v_{(m+1)(k+1)}$

W powyższych wzorach $x(v)$ oraz $y(v)$ oznaczają odpowiednio współrzędną x (odcięta w kartezjańskim układzie współrzędnych) i y (rzędna w kartezjańskim układzie współrzędnych) wierzchołka v. Kąt, jaki tworzy oś rzędnych i prosta przeprowadzona między wierzchołkiem początkowym i końcowym, oznaczony został symbolem g_k.

Metoda zakłada możliwość odrzucenia części łuków, jeśli ich koszt jest większy niż zdefiniowany próg przejezdności (*passabilityThreshold*). Mając wyznaczony pas dla każdego obiektu, możliwe jest znalezienie najkrótszej drogi za pomocą algorytmu Dijkstry, używając różnych strategii poszukiwań. Są to m.in. trzy strategie wyznaczania kolejności obiektów, w jakiej będą wyszukiwane dla nich najkrótsze drogi: *stripeOrderStrategies* = {*Ascending, Descending, Random*}, gdzie *Ascending*

— kolejność rosnąca zgodnie z numeracją obiektów; *Descending* — kolejność malejąca numeracji obiektów oraz *Random* — kolejność losowa (rozkład równomierny). Dwie pierwsze deterministyczne strategie generują tylko jedną możliwość kolejności, natomiast strategia *Random* generuje *K*! różnych możliwości. Liczba sprawdzanych możliwości jest ograniczana przez warunki stopu, o których będzie mowa w dalszej części tego rozdziału.

Istnieje również możliwość parametryzacji strategii generowania szerokości pasa (sw_k). Szerokość może być albo stała i nigdy nie zmieniana, albo losowana z określonego zakresu z rozkładem równomiernym: *widthOfStripeGenerationStrategy* = {*Constant, Random*}. Strategia losowa może wygenerować maksymalnie *L* różnych szerokości (*L* zależy od zdefiniowanego przedziału oraz minimalnego przyrostu szerokości).

Iteracją w algorytmie SGDP jest próba znalezienia pęku *K* dróg przy zadanej kolejności oraz szerokości pasów. Liczba iteracji ograniczana jest przez warunki stopu zdefiniowane w strategiach stopu.

Zdefiniowano kilka strategii stopu, które można łączyć w dowolne konfiguracje, a każdą z nich można kalibrować w zależności od potrzeb. Strategie stopu mogą spowodować zatrzymanie algorytmu, jeśli osiągnięty został: limit czasu (*TimeLimit*), maksymalna liczba iteracji (*MaxIterationNumber*), maksymalna liczba znalezionych rozwiązań dopuszczalnych (*NFeasibleSolutionFound*) lub jeśli kolejne rozwiązanie dopuszczalne nie jest lepsze od poprzedniego o określoną wartość (*NextSolutionIsBetterThan*):

stopStrategies = {*MaxIterationNumber, NFeasibleSolutionFound, NextSolutionIsBetterThan, TimeLimit*}.

Dodatkowym parametrem jest tryb zwiększania szerokości pasa:

widthGenerationModes = {*SameWidth, VariousWidth*}.

Dla trybu *SameWidth* każdy z obiektów ma wyznaczany pas o identycznej szerokości, tzn. jeśli dla któregokolwiek obiektu w iteracji nie udało się znaleźć drogi przy określonej szerokości pasa, wyznaczana jest nowa szerokość i proces poszukiwania dróg rozpoczynany jest dla każdego obiektu od nowa. Drogi, które wcześniej udało się wyznaczyć, są zapominane. Dla trybu *VariousWidth* każdy obiekt może mieć różną szerokość pasa w pojedynczej iteracji, czyli brak rozwiązania dla *k*-tego obiektu powoduje zwiększenie szerokości pasa tylko dla tego obiektu. Poprzednio wyznaczone drogi są pamiętane.

Strategie wyznaczania kolejności obiektów zapewniają, że żadna kolejność nie jest sprawdzana dwukrotnie. Również strategie wyznaczania szerokości pasa generują zawsze niesprawdzaną do tej pory szerokość pasa. W przypadku losowej strategii wyznaczania szerokości pasa każda kolejna szerokość jest większa od poprzedniej.

Analizując złożoność obliczeniową algorytmu, można zauważyć, że wyznaczenie podgrafu dla pojedynczego odcinka (mając dany wierzchołek początkowy, końcowy

oraz graf początkowy) wymaga $O(V)$ operacji, zaś poszukiwanie drogi za pomocą algorytmu Dijkstry opartego na kopcach binarnych ma złożoność rzędu $O(A \log V)$. Zarówno wyznaczenie pasa jak i poszukiwanie drogi jest powtarzane dla każdego odcinka każdego obiektu, a więc KM razy. Ze względu na możliwość powrotów przy braku satysfakcjonującego rozwiązania, poszukiwania pęku dróg mogą być prowadzone dla różnych szerokości pasa, czyli L razy oraz dla różnych kolejności obiektów, czyli w najgorszym przypadku $K!$ razy. Ostateczna złożoność pesymistyczna algorytmu SGDP jest rzędu $O(K!(KLM\ (V+A \log V))$.

Pseudokod algorytmu SGDP:

```
0) Zapisz początkowy stan grafu
1) Dopóki żaden z warunków stopu nie jest spełniony {
  2) Wygeneruj kolejność pasów używając wybranej stripeOrderStrategy
  3) Jeśli nie pozostała żadna nie sprawdzana kolejność pasów → Zakończ
  4a) Jeśli tryb poszukiwań = SameWidth{
    5a) Dopóki żaden z warunków stopu nie jest spełniony {
      6a) Wygeneruj szerokość pasa używając wybranej widthOfStripeGenera-
      tionStrategy
        7a) Jeśli nie pozostała żadna niesprawdzona szerokość -> przywróć
        stan grafu i przejdź do kroku 5a);
          8a) Dla każdego k-tego obiektu, k=1,…,K{
          9a) Dla każdego odcinka w PS(k) {
            10a) Szukaj drogi dla odcinka w wygenerowanym podgrafie}
              11a) Jeśli droga została znaleziona -> zapisz drogę i usuń
              wykorzystane wierzchołki i łuki z grafu}
          12a) Jeśli wszystkie drogi zostały znalezione -> zapisz rozwiązanie
          dopuszczalne}}
4b) Jeśli tryb poszukiwań = VariousWidth{
  5b) Dopóki żaden z warunków stopu nie jest spełniony {
    6b) Dla każdego k-tego spośród K obiektów {
      7b) Wygeneruj szerokość pasa używając wybranej widthOfStripeGenera-
      tionStrategy
      8b) Jeśli nie pozostała żadna niesprawdzona szerokość -> przywróć stan
      grafu i przejdź do kroku 5b);
        9b) Dla każdego odcinka w PS(k) {
          10b) Szukaj drogi dla odcinka w wygenerowanym podgrafie;}
            11b) Jeśli droga została znaleziona -> zapisz drogę i usuń
            wykorzystane wierzchołki i łuki z grafu
            W przeciwnym przypadku przywróć początkowy stan grafu }
          12b) Jeśli wszystkie drogi zostały znalezione -> zapisz rozwiązanie
          dopuszczalne i przywróć początkowy stan grafu }}}
```

W praktyce złożoność algorytmu SGDP jest znacznie mniejsza, ponieważ wygenerowane podgrafy składają się z dużo mniejszej liczby wierzchołków, niż graf wejściowy, zaś warunki stopu zdefiniowane w strategiach stopu znacznie ograniczają liczbę przeszukiwanych możliwości. Możliwe są dalsze modyfikacje w celu zmniejszenia złożoności obliczeniowej, np. oparcie algorytmu Dijkstry na innych strukturach danych: kopcu Fibonacciego czy kolejkach priorytetowych Brodala [6]. Te ostatnie charakteryzują się bardzo dobrą złożonością obliczeniową najczęściej wykorzystywanych operacji. Inną możliwością jest sprawdzanie przynależności wierzchołków do podgrafu (pasa) „w locie", czyli w trakcie wyznaczania najkrótszej drogi lub zastosowanie zmodyfikowanego algorytmu A*.

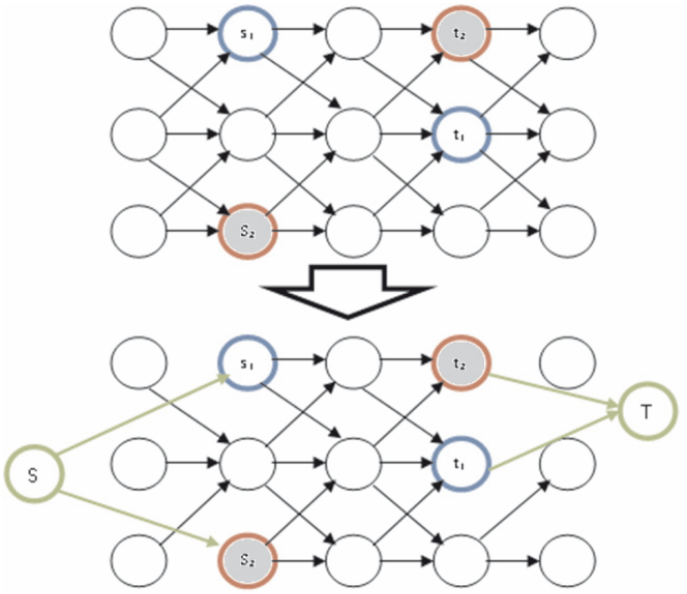
4. Opis modyfikacji metody Edmondsa-Karpa [36]

Jak zauważono wcześniej, problem znajdowania przepływu zaspokajającego o minimalnym koszcie można zastosować do wyznaczania *K* dróg rozłącznych. Wymagana jest modyfikacja modelu opartego na reprezentacji grafowo-sieciowej polegająca na dodaniu do każdego łuku dodatkowej informacji oznaczającej maksymalną przepustowość łuku. Rozwiązanie tego problemu polega na wyznaczeniu dróg w taki sposób, aby można było przez nie przesłać przepływ o wartości dokładnie *K*, przy czym przepływ może składać się z wielu dróg, a przez każdą z nich można przesłać maksymalnie taką wartość, jaką jest minimalna wartość przepustowości łuku wchodzącego w skład danej drogi.

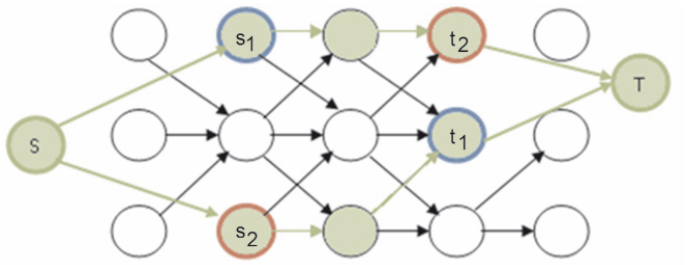
W klasycznym problemie przepływu zaspokajającego o minimalnym koszcie [10] wyznacza się przepływ pomiędzy parą wierzchołków *s*-*t*, a nie jak w opisywanym do tej pory problemie, pomiędzy *KM* parami wierzchołków

$(v_{11} \rightarrow \ldots \rightarrow v_{N1}), \ldots, (v_{1K} \rightarrow \ldots \rightarrow v_{NK})$. Aby wyznaczyć przepływ przechodzący przez wiele par, wprowadza się dodatkowe superwierzchołki [5] *S* oraz *T*, w taki sposób, że wierzchołek *S* połączony jest z każdym wierzchołkiem $v_{1k}, k = \overline{1,K}$ łukiem o koszcie 0 i przepustowości 1, zaś wszystkie inne połączenia do wierzchołków v_{1k} (wierzchołków początkowych dla każdego obiektu) są usuwane. Podobnie wszystkie łuki wychodzące z wierzchołków $v_{Nk}, k = \overline{1,K}$ są usuwane, zaś wprowadzane są nowe łuki o koszcie 0 i przepustowości 1 z wierzchołków v_{Nk} do wierzchołka *T* (patrz rys. 3).

Przepływ zaspokajający (o wartości *K*) o minimalnym koszcie wyznaczany jest z superwierzchołka *S* do superwierzchołka *T*. Ta modyfikacja zapewnia jednak jedynie wyznaczenie *K* rozłącznych dróg z *S* do *T* przechodzących przez wierzchołki v_{1k_1} i v_{Nk_2} $(k_1, k_2 = \overline{1,K})$, nie gwarantując, że $k_1 = k_2$ tzn. nie ma gwarancji, że wyznaczona droga będzie przechodziła przez wierzchołki pośrednie zdefiniowane dla jednego obiektu (patrz rys. 4). Aby problem przepływu w pełni odpowiadał omawianemu problemowi wyznaczania rozłącznych dróg, zostanie zaproponowana prosta modyfikacja metody Edmondsa-Karpa [10].



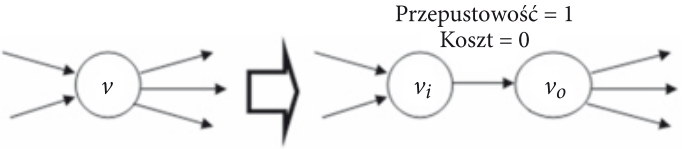
Rys. 3. Dodanie superwierzchołków S i T do grafu. Wierzchołki $s_1 = v_{11}$, $s_2 = v_{12}$ są wierzchołkami początkowymi odpowiednio pierwszego i drugiego obiektu, zaś $t_1 = v_{21}$, $t_2 = v_{22}$ wierzchołkami końcowymi dla tych obiektów ($K = 2$, $N = 2$)



Rys. 4. Przykład rozwiązania problemu przepływu bez modyfikacji. Wyznaczone drogi przechodzą przez wierzchołki, które miały należeć do różnych dróg $s_1 \rightarrow t_2$; $s_2 \rightarrow t_1$

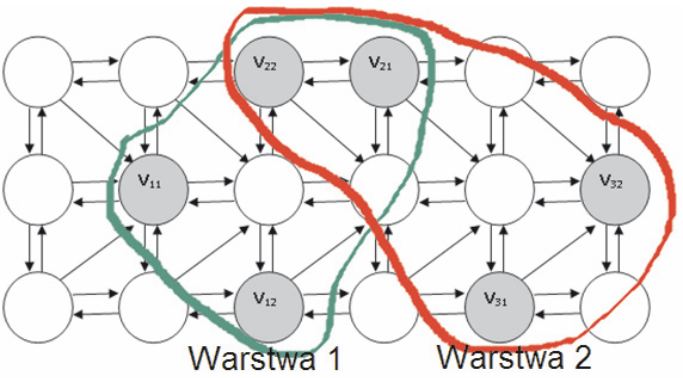
Ponieważ jednym z ograniczeń nałożonych na wyznaczone drogi jest ich rozłączność, każdemu łukowi należy przypisać identyczną wartość przepustowości równą 1 — wyklucza to możliwość wykorzystania łuku więcej niż raz, a więc zapewnia rozłączność krawędziową. Rozwiązaniem problemu rozłączności wierzchołkowej jest zastąpienie wierzchołka v w wejściowym grafie G dwoma wierzchołkami v_i i v_o w taki sposób, że do wierzchołka v_i wchodzą wszystkie te łuki, które wchodziły do wierzchołka v (o identycznych kosztach i przepustowościach), zaś wychodzi jeden łuk wchodzący do wierzchołka v_o. Łukowi temu nadajemy jednostkową przepustowość oraz zerowy koszt przejścia. Do wierzchołka v_o wchodzi tylko jeden

łuk wychodzący z v_i, zaś wychodzą z niego wszystkie te łuki, które wychodziły z wierzchołka v (patrz rys. 5).



Rys. 5. Transformacja wierzchołka v w wierzchołki v_i i v_o

Ostatnią kwestią do rozwiązania pozostaje uwzględnienie wierzchołków pośrednich w wyznaczanych drogach. Dokonano tego, rozwiązując problem przepływu zaspokajającego (o wartości K) o minimalnym koszcie kilkukrotnie dla różnego zestawu wierzchołków początkowych oraz końcowych, tzn. jeśli k-ta droga ma przechodzić przez wierzchołki $v_{1k},\ldots,v_{Nk}$, to należy rozwiązać problem przepływu $M = N\text{-}1$ razy, szukając przepływu najpierw między K parami wierzchołków v_{1k}, a v_{2k} następnie między K parami v_{mk}, *a* $v_{(m+1)k}$, kończąc na przepływie między K parami wierzchołków $v_{(M-1)k}$, *a* v_{Mk} (patrz rys. 6). Każdy zestaw K par wierzchołków, pomiędzy którymi poszukiwany jest przepływ zaspokajający o wartości K, nazywany będzie warstwą; w problemie wejściowym można więc wyróżnić M warstw.



Rys. 6. Sieć z zaznaczonymi warstwami. Obiekt 1 musi przechodzić przez wierzchołki v_{11}, v_{21} oraz v_{31}, zaś obiekt 2 przez wierzchołki v_{12}, v_{22} i v_{32}. W warstwie 1 wierzchołkami terminalnymi są wierzchołki v_{11}, v_{12}, v_{21} i v_{22}, zaś w warstwie 2 wierzchołkami terminalnymi są v_{21}, v_{22}, v_{31} i v_{32}. Przykład podano dla $K = 2$, $N = 3$

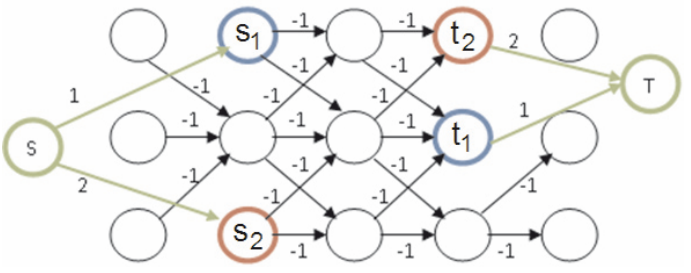
Do rozwiązania problemu przepływu zaspokajającego o minimalnym koszcie użyto klasycznej metody Edmondsa-Karpa, zaprezentowanej przez autorów w [10]. Ideą tej metody jest, zaczynając od zerowego przepływu, powiększanie go za pomocą „dróg powiększających" (ang. *augmenting paths*). Drogi wyszukiwane są za pomocą uogólnienia algorytmu BFS, algorytmu Dijkstry tak, aby znaleziona droga powiększająca była najkrótsza (najmniejszy koszt) oraz zawierała tylko takie łuki,

które mają dostatecznie dużą przepustowość. Po wyznaczeniu drogi, przepustowości na łukach wchodzących w jej skład są pomniejszane o największą wartość przepływu, jaką można przesłać przez całą drogę, co w rozważanym przypadku oznacza ich zerowanie.

Pseudokod metody Edmondsa-Karpa (*s* — wierzchołek początkowy, *t* — wierzchołek końcowy):

(1) ustaw maksymalny przepływ na zero;
(2) wyszukaj najmniej kosztowną drogę z *s* do *t*; wszystkie łuki wchodzące do drogi muszą mieć przepustowość >0;
(3) jeśli droga istnieje:
(a) zwiększ wartość maksymalnego przepływu o wartość najmniejszej przepustowości łuku należącego do wyznaczonej drogi;
(b) zmniejsz przepustowość każdego łuku należącego do drogi o wartość najmniejszej przepustowości łuku należącego do wyznaczonej drogi;
(4) jeśli droga nie istnieje — zakończ.

Jak wspomniano wcześniej, aby zapewnić, że drogi będą wyszukiwane pomiędzy wierzchołkami v_{nk} a $v_{(n+1)k}$, $n=\overline{1,N-1}$, czyli że w każdej warstwie łączone będą odpowiednie wierzchołki zdefiniowane dla tego samego obiektu, należy wprowadzić dodatkową modyfikację, która polega na tym, że każdy łuk etykietowany jest liczbą całkowitą w sposób następujący: każdy łuk wychodzący z superwierzchołka *S* jest zaetykietowany wartością *k* wtedy, gdy wchodzi do wierzchołka v_{nk} oraz każdy łuk wchodzący do superwierzchołka *T* jest zaetykietowany wartością *k* wtedy, gdy wychodzi z wierzchołka $v_{(n+1)k}$. Pozostałe łuki zostają zaetykietowane wartością –1 (patrz rys. 7).



Rys. 7. Sieć z nadanymi etykietami łuków

W tak zaetykietowanej sieci wyszukiwana jest droga za pomocą zmodyfikowanego algorytmu Dijkstry. Droga wychodząca z wierzchołka *S* po przejściu przez pierwszy łuk, który ma niezerową przepustowość, zostaje na stałe zaetykietowana wartością jego etykiety (etykieta drogi nie zmienia się do końca poszukiwań). Następnie przeszukiwane są kolejne wierzchołki, do których prowadzą łuki o etykiecie –1, czyli przeszukiwanie odbywa się w sposób zgodny z klasyczną metodą

Dijkstry (oczywiście z uwzględnieniem wymogu na niezerową przepustowość łuków). Gdy podczas wyszukiwań napotkany zostanie superwierzchołek *T*, do którego wchodzi łuk o etykiecie różnej od –1, następuje porównanie, czy wartość etykiety łuku jest identyczna z etykietą drogi. Jeśli wartości są zgodne, to przejście do wierzchołka *T* jest możliwe (aktualna wartość przepustowości jest ustalana na 1), jeśli wartości etykiet różnią się, aktualna wartość przepustowości jest ustawiana na 0, a więc wierzchołek *T* jest niedostępny.

Ostatnią modyfikacją wprowadzoną, aby zapewnić, że obie metody rozwiązują ten sam problem, jest wprowadzenie progu przejezdności, czyli wymogu, że wszystkie wyznaczone drogi mogą zawierać tylko te łuki, których koszt przejścia nie przekracza progu przejezdności (*passabilityThreshold*).

Analiza złożoności obliczeniowej zmodyfikowanej metody Edmondsa-Karpa pokazuje, że złożoność ta jest rzędu $O(M(A+KA\log V))$. Każda z *K* dróg podzielona jest na *M* odcinków. Dla każdego zestawu *M* odcinków (*M* warstw) wykonywana jest transformacja grafu polegająca na dodaniu superwierzchołków oraz poszukiwaniu *K* dróg powiększających przepływ za pomocą algorytmu Dijkstry. Transformacja dokonywana jest w czasie $O(A)$, ponieważ należy sprawdzić każdy z łuków, aby dokonać usunięcia niepotrzebnych łuków, zaś algorytm Dijkstry działa w czasie $O(A \log V)$.

Choć teoretyczna złożoność tego algorytmu jest lepsza od złożoności obliczeniowej metody SGDP, w praktyce czas wykonania jest większy, ponieważ w omawianej metodzie algorytm Dijkstry przeszukuje cały graf wejściowy, a nie, jak w przypadku metody SGDP, mały podgraf grafu wejściowego (patrz wyniki w rozdziale 5).

5. Wyniki eksperymentów

Badania zostały przeprowadzone na grafach przedstawiających rzeczywiste obszary terenu. Grafy zawierały od 5000 do 30000 wierzchołków, dla liczby obiektów $K = \{2, 3, 4, 5\}$ i liczby punktów pośrednich $N = \{2, 3, 4, 5\}$. Punkty pośrednie były losowane. Badania metody SGDP przeprowadzono dla prawie każdego możliwego zestawu wartości parametrów, przy czym użyto konfiguracji, jak poniżej.

Strategie generowania szerokości pasa:

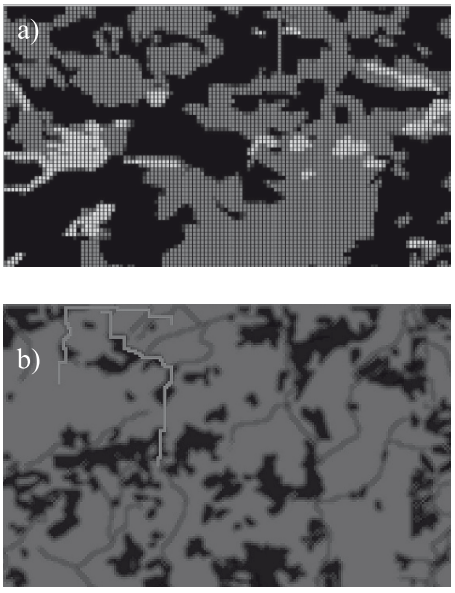
— *ConstantWidthOfStripStrategy* — domyślna wartość szerokości pasa = 10,0;
— *RandomWidthOfStripStrategy* — zakres losowanych szerokości < 5,0; 10,0 >; minimalny przyrost szerokości 0,5.

Metodę SGDP badano przy trzech zestawach strategii stopu:

— {*TimeLimit*; *MaxIterationNumber*};
— {*TimeLimit*, *MaxIterationNumber*, *NextSolutionBetterThan*};
— {*TimeLimit*, *MaxIterationNumber*; *NFeasibleSolutionFound*}.

Domyślnymi parametrami były dla nich:
- *TimeLimit* — limit czasu = 5000 ms;
- *MaxIterationNumber* — maksymalna liczba iteracji = 10;
- *NextSolutionBetterThan* — minimalna poprawa wartości funkcji celu = 5,0;
- *NFeasibleSolutionFound* — maksymalna liczba znalezionych rozwiązań dopuszczalnych = 4.

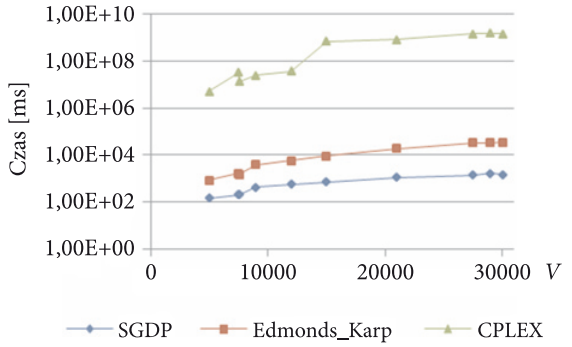
Wyniki otrzymane z obu metod zostały porównane również z rozwiązaniami optymalnymi problemu (1)-(7) otrzymanymi z solvera CPLEX 7.0 znajdującego się w pakiecie GAMS 2.0. Obliczenia przeprowadzono na procesorze Intel® Core™ i5 430i, 2×2.27 GHz.



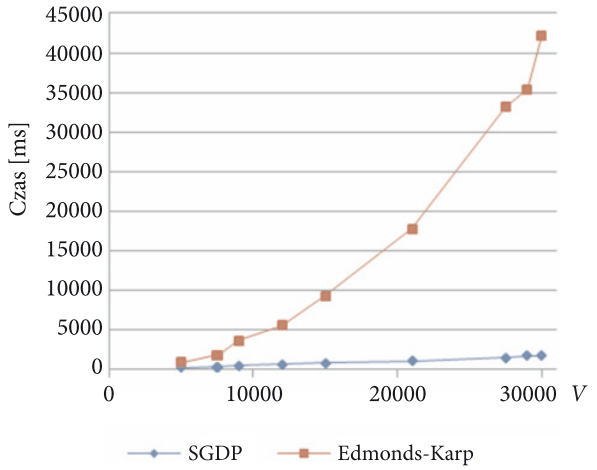
Rys. 8. Typowe grafy prostokątne reprezentujące fragment terenu. Kolorem oznaczono koszt przypisany do wierzchołków: jaśniejszy kolor oznacza lepiej przejezdne obszary, najciemniejszy oznacza przeszkody takie jak lasy, rzeki, jeziora, zabudowania. Im jaśniejszy kolor, tym mniejszy koszt przejścia danego obszaru; a) graf o 7540 = 65 × 116 wierzchołkach reprezentujący teren w pobliżu Drawska (Polska); b) graf o 25000 = 125 × 200 wierzchołkach reprezentujący teren w pobliżu Radomia (Polska) wraz z dwoma wierzchołkowo-rozłącznymi drogami wyznaczonymi metodą SGDP

Na wykresie średniego czasu obliczeń dla poszczególnych metod (wykres 1) zastosowano skalę logarytmiczną. Widać, że metoda SGDP jest średnio o jeden rząd wielkości szybsza od metody Edmondsa-Karpa (E-K) i pięć rzędów wielkości od metody z solvera CPLEX. Różnicę w średnim czasie wykonania tych samych zadań przez SGDP oraz E-K dobrze pokazuje wykres 2.

Przeprowadzono również szczegółowe badania wpływu zastosowania różnych strategii w metodzie SGDP. Badania pokazały, że algorytm działa szybciej przy



Wykres 1. Średni czas wykonania dla metody SGDP, Edmondsa-Karpa oraz zadania optymalizacyjnego rozwiązywanego przez CPLEX (skala logarytmiczna)



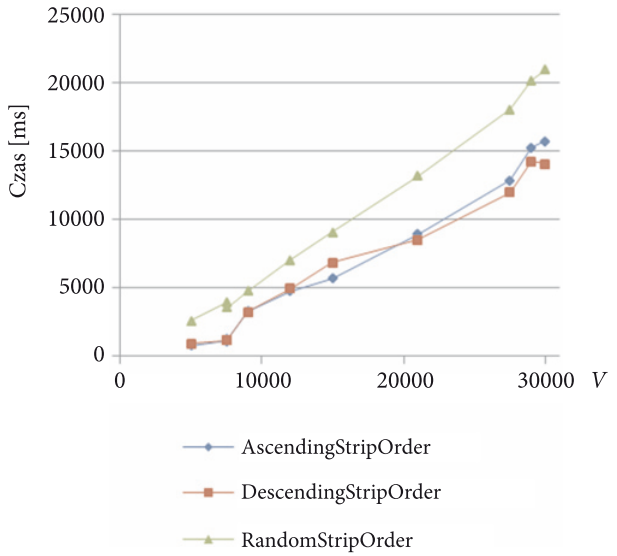
Wykres 2. Średni czas wykonania dla metody SGDP i Edmondsa-Karpa

zastosowaniu strategii generowania stałej szerokości pasa (*ConstantWidthOfStrip)* i choć różnica dla małych wartości *V* (liczby wierzchołków) jest znikoma (ok. 20 ms), powiększa się w miarę wzrostu *V* (do 220 ms).

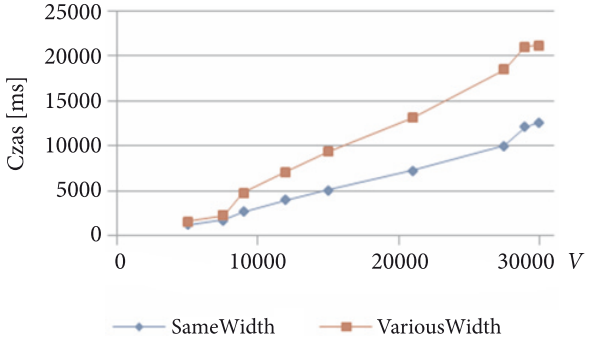
Znacznie większa różnica w czasie wykonania występuje dla różnych strategii generowania kolejności obiektów (wykres 3). O ile nie ma znaczących różnic pomiędzy stosowaniem strategii *Ascending* i *Descending*, o tyle czas potrzebny do obliczeń przy wykorzystaniu strategii *Random* jest dłuższy.

Jeszcze większe różnice w czasie wykonania można zauważyć z danych przedstawionych na wykresie 4. Algorytm działa istotnie szybciej (prawie dwukrotnie) dla trybu zakładającego jednakową szerokość dla wszystkich pasów.

Istotną różnicę w czasie wykonania algorytmu powoduje również wybór zestawów strategii stopu (wykres 5). Jest to spodziewany wynik, ponieważ wybór



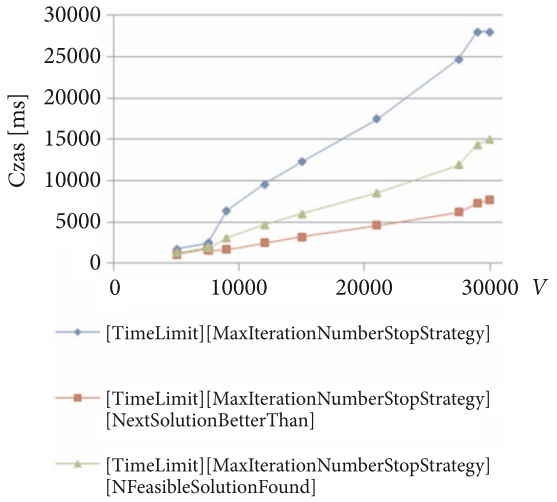
Wykres 3. Średni czas wykonania metody SGDP z podziałem na strategie generowania kolejności obiektów



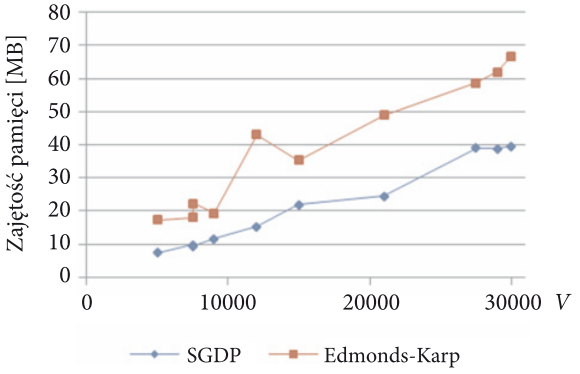
Wykres 4. Średni czas wykonania metody SGDP dla dwóch trybów: jednakowa szerokość pasa dla wszystkich obiektów (*SameWidth*) oraz różna szerokość pasa dla różnych obiektów (*VariousWidth*)

bardziej restrykcyjnych strategii stopu zmusi algorytm do wcześniejszego zakończenia działania, może jednak również spowodować nieznalezienie rozwiązania lub znalezienie rozwiązania odległego od optymalnego.

Z zaprezentowanych wyników można stwierdzić, że w praktyce czas obliczeń dla metody SDGP jest liniowy względem rozmiaru danych (liczby wierzchołków grafów wejściowych). Stwierdzenia takiego nie można sformułować o metodzie Edmondsa-Karpa, ponieważ obserwowany fragment krzywej sugeruje wyższy rząd funkcji złożoności. Metoda Edmondsa-Karpa wymaga zdecydowanie większej ilości pamięci (wykres 6), co nie powinno dziwić, ponieważ metoda SGDP bazuje



Wykres 5. Średni czas wykonania metody SGDP dla różnych zestawów strategii stopu

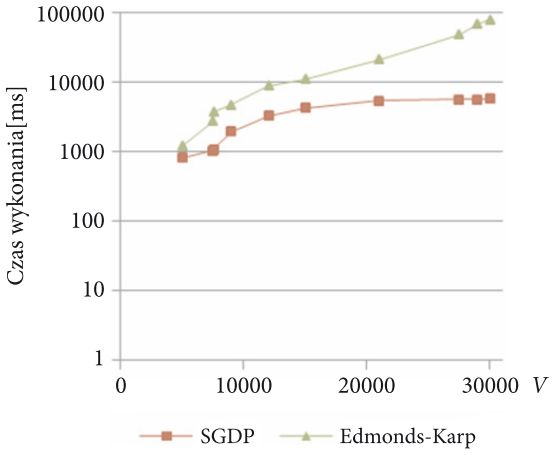


Wykres 6. Porównanie średniej liczby zajętych bajtów pamięci przez metody SGDP i Edmondsa-Karpa

głównie na podgrafach grafu wejściowego, co znacznie redukuje nie tylko czas obliczeń, ale i wymaganą ilość pamięci.

Podobnie jak dla średniego czasu wykonania, przeprowadzono szczegółowe badania metody SGDP z podziałem na różne strategie i tryby poszukiwań. Nie wykazały one jednak praktycznie żadnych różnic, dlatego też wyniki nie zostaną zaprezentowane i będą traktowane jako nieistotne dla dalszej analizy.

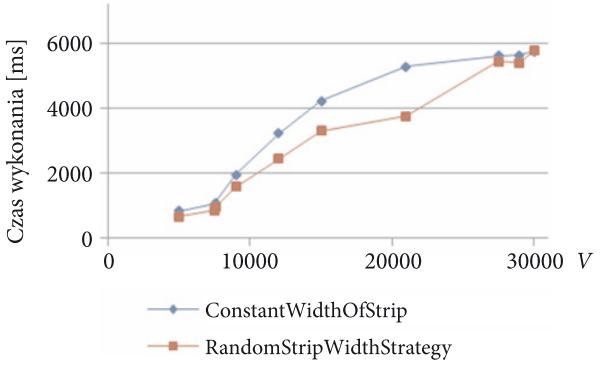
Omawiane metody rozwiązania problemu wyznaczania K dróg rozłącznych zostały poddane również badaniu wrażliwości ze względu na czas wykonania. Miara wrażliwości rozumiana jest jako największa różnica między czasami wykonania algorytmu dla dowolnych różnych zestawów danych wejściowych. Na wykresie 7 widać, że metoda SGDP jest zdecydowanie mniej wrażliwa. Wykres wrażliwości



Wykres 7. Porównanie średniej wrażliwości ze względu na czas wykonania metod SGDP, Edmondsa-Karpa (skala logarytmiczna)

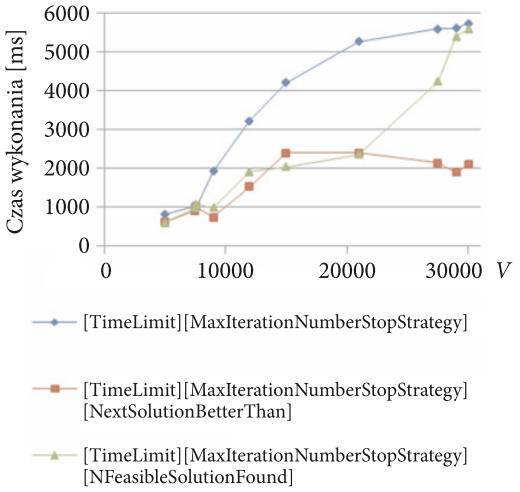
sugeruje zależność logarytmiczną, natomiast wykres dla metody Edmondsa-Karpa sugeruje zależność wielomianową. Należy dodać, że obie metody są bardzo wrażliwe, ponieważ średni czas wykonania dla metody SGDP wynosił od 139 do 1691 ms (w zależności od rozmiaru grafu), a średnia eksperymentalna wrażliwość ze względu na czas wykonania wynosiła 820-5754 ms, natomiast dla metody Edmondsa-Karpa średni czas wynosi 800-42237 ms przy wrażliwości 1187-78102 ms.

Metoda SGDP wykazuje się dużą wrażliwością ze względu na czas wykonania przy zmianach strategii generowania szerokości pasa (wykres 8). Może być to spowodowane występującymi przypadkami nieznajdowania dróg, dla których wynik może pojawić się już w pierwszych kilkudziesięciu milisekundach (szczególnie dla deterministycznych strategii, które umożliwiają sprawdzenie tylko jednej wartości danego parametru).

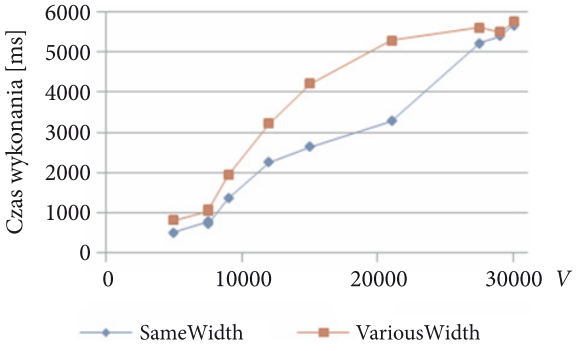


Wykres 8. Wrażliwość metody SGDP ze względu na czas wykonania z podziałem na strategie generowania szerokości pasa

Dla badanych strategii generowania kolejności obiektów dużych różnic nie odnotowano. Największe różnice wrażliwości otrzymano, badając metodę SGDP przy stosowaniu różnych zestawów strategii stopu (wykres 9).



Wykres 9. Wrażliwość metody SGDP ze względu na czas wykonania z podziałem na zestawy strategii stopu



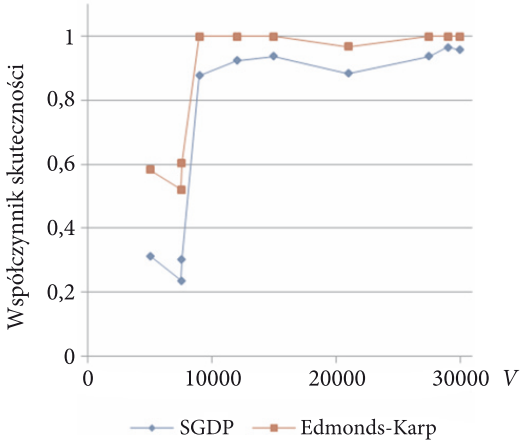
Wykres 10. Wrażliwość metody SGDP ze względu na czas wykonania z podziałem na tryb poszukiwań: wspólna szerokość dla wszystkich obiektów (*SameWidth*) oraz różna szerokość pasów dla różnych obiektów (*VariousWidth*)

Wrażliwość SGDP może być nawet pięciokrotnie razy większa od średniej wartości. Jest to własność, która może wykluczyć metodę SGDP z zastosowań, gdzie potrzeba bardzo dokładnego oszacowania czasu wykonania algorytmu (np. w systemach czasu rzeczywistego).

Badania pokazały, że stosowanie wspólnej szerokości dla wszystkich obiektów jest mniej wrażliwe ze względu na czas wykonania.

Współczynnik skuteczności pozwoli oszacować, dla jakiego procentu liczby przypadków danych zostanie znalezione rozwiązanie dopuszczalne, jeśli takie rozwiązanie istnieje (istnienie rozwiązania dopuszczalnego weryfikowane jest przez solver CPLEX). Jest to pierwszy krok na drodze do wyznaczenia eksperymentalnych współczynników aproksymacji.

Na wykresie 11 przedstawiono współczynniki skuteczności dla badanych przypadków danych wejściowych. Można zauważyć, że metoda Edmondsa-Karpa daje bardzo dobre wyniki (na poziomie 0,99) w teście skuteczności, natomiast metoda SGDP gorsze, ale również do przyjęcia (na poziomie 0,95). Jest to wynik spodziewany, ponieważ jeśli droga istnieje w mniejszym podgrafie, na pewno będzie istnieć również i w większym (zbiór wierzchołków podgrafu większego zawiera wszystkie wierzchołki podgrafu mniejszego). Jednocześnie z analizy wykresu wynika, że dla zastosowanych danych testowych skuteczność metod była bardzo dobra w przypadku grafów o liczbie wierzchołków większej od ok. 8000. Choć metoda SGDP, w przeciwieństwie do Edmondsa-Karpa, potrafi poszukiwać dróg z różną kolejnością obiektów, jest to jednak element, który niezwykle rzadko wpływa na poprawę współczynnika skuteczności, częściej natomiast wpływa na współczynnik dokładności rozwiązania.

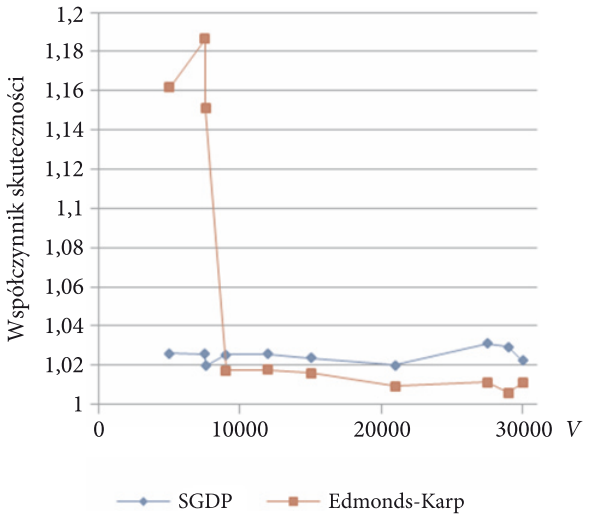


Wykres 11. Współczynnik skuteczności dla metody SGDP i metody Edmondsa-Karpa

Ze względu na przyjęte parametry domyślne strategii generowania szerokości pasa, a więc takie, że maksymalna szerokość pasa generowana strategią *RandomStripWidthStrategy* jest równa stałej szerokości generowanej przez strategię *ConstantWidthOfStrip*, zmiana tych strategii nie ma praktycznie żadnego wpływu na skuteczność. Również żadnych znaczących zmian w wartości eksperymentalnego współczynnika skuteczności nie pokazały ani zmiany zestawów strategii stopu, ani zmiana parametru odpowiedzialnego za wyszukiwanie dróg ze wspólną lub różną szerokością pasa.

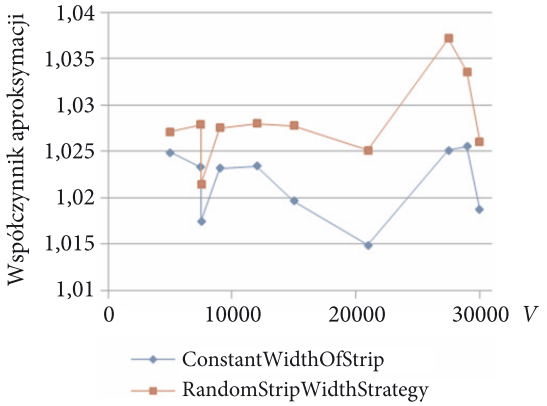
Ponieważ współczynnik skuteczności metody SGDP i Edmondsa-Karpa jest na rozsądnie wysokim poziomie (średnio powyżej 0,95 dla dużych wartości *V*), warto zbadać eksperymentalne współczynniki aproksymacji. Ich wartości mogą przesądzić o przyjęciu bądź odrzuceniu metod jako sposobów rozwiązania badanego problemu.

Badania wykazały, że średni eksperymentalny współczynnik aproksymacji dla metod SGDP i Edmondsa-Karpa jest bardzo blisko 1 dla większości badanych przypadków. Średni eksperymentalny współczynnik aproksymacji na poziomie 1,03 oznacza, że znalezione rozwiązania są gorsze od optymalnych zaledwie o 3%. Pozwala to zaliczyć badane metody do grona metod dających bardzo dobre wyniki. Na wykresie 12 widać, że dla badanych grafów o małych liczbach wierzchołków (*V*) metoda SGDP uzyskała przewagę nad metodą Edmondsa-Karpa, co musiało być spowodowane zmianą kolejności przeszukiwań obiektów. Dla większości przypadków jednak metoda Edmondsa-Karpa okazała się bliższa znalezieniu rozwiązań optymalnych, choć nieznacznie.



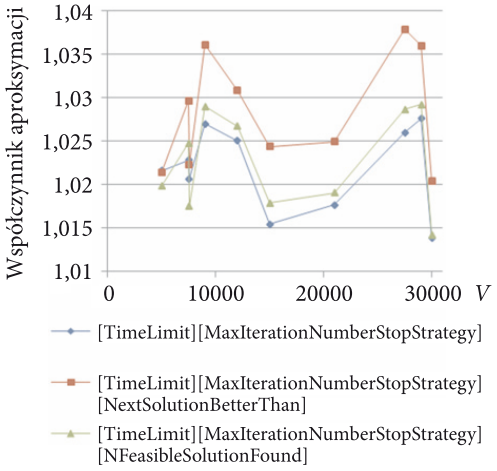
Wykres 12. Średni eksperymentalny współczynnik aproksymacji dla metod SGDP i Edmondsa-Karpa

Przeprowadzono również szczegółową analizę metody SGDP i wpływu doboru jej strategii na wartość średniego eksperymentalnego współczynnika aproksymacji. Obliczany jest jako średnia arytmetyczna z wartości eksperymentalnego współczynnika aproksymacji z wielu uruchomień algorytmu dla wielu zestawów danych określonego rozmiaru. Im mniejsza wartość tego współczynnika, tym algorytm jest bardziej preferowany. Wykazano, że lepsze (bliższe optymalnym) rozwiązania daje strategia generowania szerokości pasa *ConstantWidthOfStrip* (wykres 13). Wpływ



Wykres 13. Średni eksperymentalny współczynnik aproksymacji dla metody z podziałem na strategie generowania szerokości pasa

strategii generowania kolejności obiektów wydaje się nie mieć znaczenia, ale za to wykres zależności średniego eksperymentalnego współczynnika aproksymacji od wybranych zestawów strategii stopu każe skłaniać się ku zestawowi [*TimeLimit*] [*MaxIterationNumberStopStrategy*], jednak przewaga tego zestawu nad innymi badanymi jest niewielka (wykres 14).



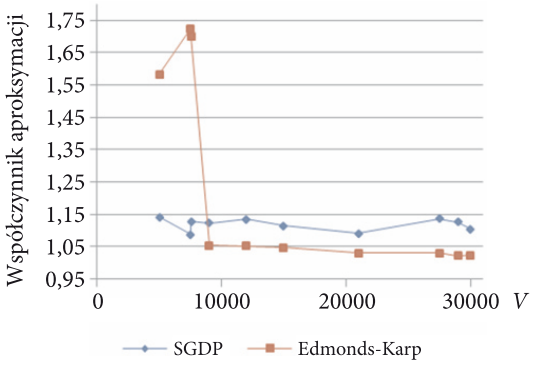
Wykres 14. Średni eksperymentalny współczynnik aproksymacji dla metody z podziałem na wybrane zestawy strategii stopu

W celu sprawdzenia, jak silnie mogą różnić się znajdowane rozwiązania dopuszczalne przez metody SGDP i Edmondsa-Karpa od rozwiązań optymalnych znajdowanych przez solver CPLEX 7.0, wyznaczono maksymalny eksperymentalny

współczynnik aproksymacji. Okazuje się, że współczynnik ten jest również na bardzo dobrym, niskim poziomie (poniżej 1,05 dla metody Edmondsa-Karpa i nieco gorszy, bo na poziomie 1,05-1,15 dla SGDP).

Analiza metody SGDP pod kątem zmian strategii generowania szerokości pasa ponownie wskazała na wyższość strategii generującej stałą szerokość pasa nad strategią *RandomStripWidthStrategy*.

Analiza zmian pozostałych strategii nie wykazała zdecydowanego wpływu którejkolwiek z nich na badany współczynnik. Podobnie analiza zmian parametru odpowiedzialnego za poszukiwanie dróg z jednakową lub różną szerokością pasów dla wszystkich obiektów nie wykazała praktycznie żadnych zmian w wartości maksymalnego eksperymentalnego współczynnika aproksymacji.



Wykres 15. Maksymalny eksperymentalny współczynnik aproksymacji dla metod SGDP i Edmondsa-Karpa

Wykres 16. Maksymalny eksperymentalny współczynnik aproksymacji dla metody SGDP z uwzględnieniem zmian strategii generowania szerokości pasa

6. Podsumowanie

W pracy zaprezentowano metody rozwiązywania problemu znajdowania najkrótszych dróg rozłącznych w grafach prostokątnych. Algorytmy te mogą być użyte do planowania przemieszczenia, np. planowania tras w symulatorach pola walki.

Metody dokładne (np. zaimplementowane w solverze CPLEX) mają niedopuszczalnie duży czas działania, dlatego naturalnym kierunkiem rozwoju badań w tej dziedzinie jest konstrukcja algorytmów aproksymacyjnych lub takich, które, choć nie zapewniają 100% skuteczności, charakteryzują się bardzo dobrymi wartościami eksperymentalnych współczynników aproksymacji.

Analiza wybranych metod ze względu na zdefiniowane miary oceny wykazała, że zaimplementowane metody (SGDP, Edmondsa-Karpa) są:

a) szybkie — średni czas wykonania jest kilka rzędów wielkości lepszy niż średni czas wykonania przez solver CPLEX;
b) skuteczne — współczynnik skuteczności na poziomie 0,99 dla metody Edmondsa-Karpa i 0,95 dla SGDP pokazują wysoki odsetek przypadków, w których metody te znajdują rozwiązania dopuszczalne (o ile one istnieją);
c) bardzo dokładne — współczynniki aproksymacji bliskie 1; bardzo dobra wartość średniego eksperymentalnego współczynnika aproksymacji i nieco tylko gorsza wartość maksymalnego eksperymentalnego współczynnika aproksymacji;
d) wrażliwe ze względu na czas wykonania — wartości maksymalnych wartości wrażliwości są duże, jednak nawet dla najgorszych przypadków czas znalezienia rozwiązań przez te metody jest do przyjęcia.

Zastosowane ograniczenie przestrzeni poszukiwań przez metodę SGDP prowadzi do znacznego przyspieszenia działania algorytmu kosztem niewielkiej straty skuteczności i dokładności.

Zauważono, ze zastosowanie losowości w metodzie SGDP, zarówno podczas wyznaczania szerokości pasa jak i przy wyznaczaniu kolejności obiektów nie poprawiło badanych charakterystyk, z wyjątkiem nieznacznej poprawy wrażliwości ze względu na czas wykonania.

Ponieważ oba zaprezentowane algorytmy są algorytmami aproksymacyjnymi, wydaje się niezbędne określenie warunków koniecznych i wystarczających do znalezienia rozwiązań optymalnych oraz oszacowanie teoretycznego współczynnika aproksymacji. Dodatkowo wydaje się kluczowe zbadanie wpływu liczby wierzchołków pośrednich oraz liczby obiektów na wrażliwość algorytmu oraz dokonanie kalibracji parametrów dla różnych strategii poszukiwań dla metody SGDP.

Możliwe jest rozszerzenie badanego problemu na inne kryteria, np. minimalizacja kosztu najdłuższej z dróg, problem wielokryterialny itd. Warto również zwrócić większą uwagę na możliwość zrównoleglenia obliczeń.

Artykuł wpłynął do redakcji 25.01.2011 r. Zweryfikowaną wersję po recenzji otrzymano w lutym 2011 r.

LITERATURA

[1] A. Aggarwal, J. Kleinberg, D. Williamson, *Node-Disjoint Paths on the Mesh and a New Trade-Off in VLSI Layout*, SIAM Journal on Computing, 29, 4, 2000, 1321-1333.

[2] R. Andersen, F. Chung, A. Sen, G. Xue, *On Disjoint Path Pairs with Wavelength Continuity Constraint in WDM Networks*, Proceedings of the IEEE Infocom'2004, Hong Kong (China), 7-11 March, 2004, 524-535.

[3] R. Antkiewicz, W. Kulas, A. Najgebauer, D. Pierzchała, J. Rulka, Z. Tarapata, R. Wantoch-Rekowski, *The Automation of Combat Decision Processes in the Simulation Based Operational Training Support System*, Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA), Honolulu (Hawaii, USA), April 1-5 2007.

[4] J. R. Benton, S. S. Iyengar, W. Deng, N. Brener, V. S. Subrahmanian, *Tactical route planning: new algorithms for decomposing the map*, Proceedings of the IEEE International Conference on Tools for AI, 268-277, 1995.

[5] R. Bhandari, *Survivable networks. Algorithms for Diverse Routing*, Boston, 2001.

[6] G. S. Brodal, *Fast Meldable Priority Queues*, Proceedings of the 4th International Workshop on Algorithms and Data Structures, London, 282-290, 1995.

[7] R. G. Busacker, P. J. Gowen, *A procedure for determining a family of minimum-cost flow patterns*, Operations Research Office Technical Report 15, John Hopkins University, Baltimore, 1961.

[8] C. Campbell, R. Hull, E. Root, L. Jackson, *Route planning in CCTT*, Proceedings of the 5th Conference on Computer Generated Forces and Behavioural Representation, University of Central Florida, 233-244, 1995.

[9] E. W. Dijkstra, *A Note on Two Problems in Connexion with Graphs*, Numerische Mathematik, 1, 1959, 269-271.

[10] J. Edmonds, R. M. Karp, *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems*, Journal of the ACM, 19, 2, 1972, 248-264.

[11] D. Eppstein, *Finding common ancestors and disjoint paths in DAGs*, Technical Report 95-52, Department of Information and Comp. Science, Univ. of California, Irvine, 1995.

[12] S. Even, A. Itai, A. Shamir, *On the complexity of time-table and multicommodity flow problems*, SIAM Journal on Computing, 5, 1976, 691-703.

[13] J. F. Lynch, *The equivalence of theorem proving and the interconnection problem*, SIGDA Newsletter, 5, 3, 1975, 31-36.

[14] A. Jongh, M. Gendreau, M. Labbe, *Finding disjoint routes in telecommunications networks with two technologies*, Operations Research, 47, 1999, 81-92.

[15] R. M. Karp, *On the computational complexity of combinatorial problems*, Networks, 5, 1975, 48-68.

[16] T. Kreitzberg, T. Barragy, B. Nevin, *Tactical movement analyzer: a battlefield mobility tool*, Proceedings of the 4th Join Tactical Fusion Symposium, 1990, 363-383.

[17] C. L. Li, S. T. McCormick, D. Simchi-Levi, *Finding disjoint paths with different path-costs: Complexity and algorithms*, Networks, 22, 1992, 653-667.

[18] C. L. Li, S. T. McCormick, D. Simchi-Levi, *The complexity of finding two disjoint paths with min-max objective function*, Discrete Applied Math., 26, 1990, 105-115.

[19] M. Longtin, D. Megherbi, *Concealed routes in ModSAF*, Proceedings of the 5th Conference on Computer Generated Forces and Behavioral Representation, University of Central Florida, 1995, 305-314.

[20] Y. Perl, Y. Shiloach, *Finding two disjoint paths between two pairs of vertices in a graph*, Journal of the ACM, 25, 1978, 1-9.

[21] M. D. Petty, *Computer generated forces in Distributed Interactive Simulation*, Proceedings of the Conference on Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment, The International Society for Optical Engineering, Orlando, 1995, 251-280.

[22] A. Schrijver, P. Seymour, *Disjoint paths in a planar graph — a general theorem*, SIAM Journal of Discrete Mathematics, 5, 1, 1992, 112-116.

[23] H. Sherali, K. Ozbay, S. Subramanian, *The time-dependent shortest pair of disjoint paths problem: complexity, models and algorithms*, Networks, 31, 1998, 259-272.

[24] J. W. Suurballe, *Disjoint paths in a network*, Networks, 4, 1974, 125-145.

[25] J. W. Suurballe, R. E. Tarjan, *A quick method for finding shortest pairs of disjoint paths*, Networks, 14, 1984, 325-336.

[26] Z. Tarapata, *Multi-paths optimization in unreliable time-dependent networks*, Proceedings of The 2nd NATO Regional Conference on Military Communication and Information Systems, 04-06 October, Zegrze, 1, 2000, 181-189.

[27] Z. Tarapata, *Modeling, optimization and simulation of groups movement according to group pattern in multiresolution terrain-based grid network*, Proceedings of The Regional Conference on Military Communication and Information Systems, Zegrze, 1, 2001, 241-251.

[28] Z. Tarapata, *Military route planning in battlefield simulation: effectiveness problems and potential solutions*, Journal of Telecommunications and Information Technology, 4, 2003, 47-56.

[29] Z. Tarapata, *Modele harmonogramowania zsynchronizowanego przemieszczania wielu obiektów*, Badania Operacyjne i Decyzje, 2, 2007, 83-103.

[30] Z. Tarapata, *Algorytmy harmonogramowania zsynchronizowanego przemieszczania wielu obiektów*, Badania Operacyjne i Decyzje, 4, 2008, 101-132.

[31] Z. Tarapata, *Zastosowanie metod wyznaczania przepływu w sieciach do planowania manewru wojsk*, Biuletyn Instytutu Systemów Informatycznych, 2, 2008, 31-44.

[32] Z. Tarapata, *Approximation Scheduling Algorithms for Solving Multi-objects Movement Synchronization Problem*, ICANNGA'2009, Lecture Notes in Computer Science, 5495, Springer, Heidelberg, 2009, 577-589.

[33] Z. Tarapata, *Movement Simulation and Management of Cooperating Objects in CGF Systems: a Case Study*, KES-AMSTA'2010, Lecture Notes in Artificial Intelligence, 6070, Springer, Heidelberg, 2010, 293-304.

[34] Z. Tarapata, *Multiresolution models and algorithms of movement planning and their application for multiresolution battlefield simulation*, ACIIDS'2010, Lecture Notes in Artificial Intelligence, 5991, Springer, Heidelberg, 2010, 378-389.

[35] Z. Tarapata, S. Wrocławski, *Subgraphs Generating Algorithm for Obtaining Set of Node-Disjoint Paths in Terrain-Based Mesh Graphs*, MIG'2010, Lecture Notes in Computer Science, 6459, Springer, Heidelberg, 2010, 398-409.

[36] S. Wrocławski, *Analiza wybranych modeli i metod wspomagania decyzji w sytuacjach konfliktowych*, praca magisterska pod kierunkiem Z. Tarapaty, Wydział Cybernetyki, WAT, Warszawa, 2010.

Z. TARAPATA, S. WROCŁAWSKI

Methods for solving node-disjoint shortest paths visiting specified nodes problem in mesh networks

Abstract. In the paper, models and methods for solving *K* node-disjoint shortest paths visiting specified nodes problem in mesh networks (based on a graph G) have been presented. The problem has been defined as continuous linear programming problem and two approximation methods for solving it have been presented: SGDP method (based on some iterative procedure of finding shortest paths in subgraphs of G) and modification of Edmond's-Karp method for solving minimal cost flow problem. Complexity and quality analysis of presented methods based on experimental results using real terrain models have been done.

Keywords: node-disjoint paths; battlefield simulation; subgraphs generating; movement planning