Symulacja w Badaniach i Rozwoju
Vol. 1, No. 4/2010

Zbigniew TARAPATA, Paweł GODLEWSKI
Wojskowa Akademia Techniczna, Wydział Cybernetyki,
00-908 Warszawa, ul. Kaliskiego 2
E–mail: zbigniew.tarapata@wat.edu.pl; pgodlew@gmail.com

Wielorozdzielcze modele i algorytmy planowania przemieszczania oraz ich zastosowanie w wielorozdzielczej symulacji pola walki

1 Wprowadzenie

Planowanie wielorozdzielczych tras jest problemem istotnym z punktu widzenia wielu zastosowań (mobilne roboty [7, 9], symulacja pola walki [15], systemy typu CGF [13] (ang. Computer Generated Forces), transport lub/i nawigacja [4, 12]). Dla przykładu w symulacyjnych modelach przemieszczania na polu walki wykorzystuje się środowisko wielorozdzielcze [16]. Wielorozdzielczość wynika z hierarchicznej struktury jednostek oraz sposobów ich zachowania na polu walki. Dla szczebla kompanii wymagane jest bardziej precyzyjne odzwierciedlenie środowiska (terenu), aniżeli np. dla brygady. Automatyzacja pola walki jest domeną systemów klasy CGF [13] (ang. Computer Generated Forces) lub SAF-SAFOR (ang. Semi-Automated Forces). Systemy typu CGF (SAF, SAFOR) umożliwiają generowanie elementów wirtualnego (symulowanego) pola walki i zarządzanie nimi. Automatyczne generowanie i symulowanie niektórych elementów pola walki (w szczególności związanych z przeciwnikiem) umożliwia sztabom ćwiczących wojsk przeprowadzenie ćwiczeń, gdyż wojska przeciwnika są automatycznie „podgrywane". Problem poszukiwania wielorozdzielczych dróg jest ściśle związany z problemem poszukiwania dróg w sieciach wielkich rozmiarów. Przy rozwiązywaniu tego typu problemów stosuje się dwa główne podejścia: (a) dekompozycja problemu lub modelu środowiska (sieci terenu), w którym odbywa się planowanie na mniejsze podproblemy i następnie rozwiązywanie podproblemów [7, 12, 16]; (b) zastosowanie algorytmów on-line wyznaczania dróg, które znajdują rozwiązanie krok po kroku (komórka (terenu) po komórce) [5, 8]. Pierwsza grupa metod nosi nazwę wielorozdzielczych. Jako lokalne algorytmy wyznaczania dróg w tej grupie stosuje się: zmodyfikowany algorytm Dijkstry z kolejką priorytetową reprezentowaną przez kopiec d-arny ($O(A \log_d V)$, gdzie V – liczba wierzchołków sieci, A – liczba krawędzi (lub łuków) sieci) zaproponowany przez Tarjan'a, z kolejką priorytetową reprezentowaną przez kopiec Fibonacci'ego ($O(E+V \log V)$) zaproponowany przez Fredman'a i Tarjan'a, algorytm A* (o oczekiwanej złożoności $O(\sqrt{V} \cdot V)$). Ponadto możemy zastosować algorytm Bellmann'a-Ford'a ($O(VE)$), algorytm Gabow'a-Tarjan'a ($O(\sqrt{V}E\log(VW))$, gdzie W oznacza największą, co do bezwzględnej wartości, wagę łuku/krawędzi) lub algorytm zaprezentowany przez Ahuja i innych ($O(E+V\sqrt{\log W})$). Do poszukiwania dróg najkrótszych między każdą parą wierzchołków możemy zastosować V razy (dla każdego wierzchołka) zmodyfikowany algorytm Dijkstry ($O(VE\log_d V)$), algorytm Johnson'a

dla sieci rzadkich ($O(V^2 \log V+VE)$) lub algorytm Bellman'a dla sieci acyklicznych ($O(V+E)$). Celem wszystkich metod wielorozdzielczych jest redukcja złożoności obliczeniowej poprzez redukcję rozmiaru problemu (rozmiaru sieci). Dla przykładu, niektórzy autorzy prezentują metody dekompozycji sieci oraz podziału obszaru terenu w postaci drzew czwórkowych, a następnie użycie wyszukiwania etapowego (podobnego do algorytmu A*), aby wykorzystać hierarchię zależności w drzewie czwórkowym [7].

W artykule zaprezentowano przegląd modeli i metod poszukiwania dróg w wielorozdzielczych sieciach oraz ich analizę pod kątem efektywnościowym (dokładności i czasu obliczeń). Istota jednej z opisywanych metod opiera się o „agregowanie" geograficznie sąsiednich wierzchołków (kwadratów terenu) i planowanie tras w „zagregowanej" sieci z wykorzystaniem specyficznej transformacji. Celem prezentowanych metod jest nie tylko redukcja czasu obliczeń, ale - przede wszystkim - użycie ich do planowania wielorozdzielczych tras, w szczególności w problemach symulacji pola walki.

2 Definicja wielorozdzielczego modelu terenu

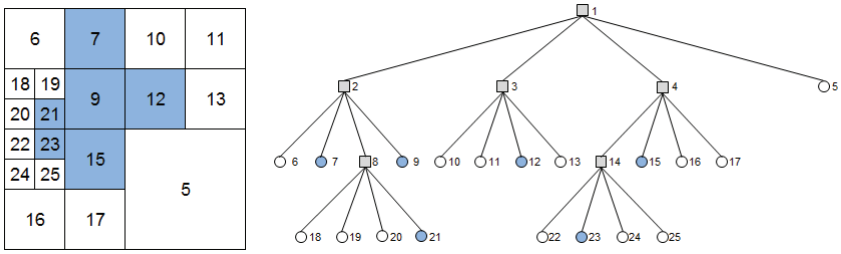
Modelowanie wielorozdzielcze rozwiązuje dwa zasadnicze problemy związane z wyszukiwaniem dróg w rozległych sieciach – problem odwzorowania naturalnej hierarchiczności zjawiska (np. w zagadnieniu przemieszczania rozmaitych formacji wojsk) oraz problem efektywnego przetwarzania dużej ilości danych (związanych np. z dużą szczegółowością lub wielkością modelu terenu). Pojęcie wielorozdzielczości zostało zdefiniowane m.in. w [10]: „Pojęcie *wielorozdzielczości* odnosi się do możliwości korzystania z różnych reprezentacji obiektów przestrzennych, posiadających różne poziomy dokładności i złożoności. Modele wielorozdzielcze pozwalają na sterowanie dokładnością odwzorowania oraz ilością związanych z nim danych. Wielorozdzielcze reprezentacje terenów sprawdzają się szczególnie w sytuacji, gdy dostępne są duże ilości danych i/lub modelowaniu podlegają znaczne obszary".

Głównymi zaletami, przemawiającymi za stosowaniem modeli i metod wielorozdzielczych jest przyspieszenie obliczeń, możliwość sterowania stopniem szczegółowości bieżącego rozwiązania zachowując zgodność rozwiązań z różnych poziomów rozdzielczości oraz możliwość przetrzymywania w pamięci podręcznej wyników części obliczeń (czyli wykonanie części pracy przed właściwym eksperymentem). Należy jednak zdawać sobie sprawę z tego, że metody te zwykle nie znajdują rozwiązań optymalnych, są bardziej skomplikowane w implementacji i stosowaniu (potrzebują np. odpowiednio przygotowanego modelu terenu) i nie są tak uniwersalne jak klasyczne metody planowania przemieszczania.

3 Analiza wielorozdzielczych algorytmów planowania tras (WAPT)

W literaturze odnaleźć można wiele przykładów metod wykorzystujących modelowanie wielorozdzielcze w procesie planowania przemieszczania [2, 3, 7]. Zwykle związane są z konkretną klasą problemów, taką jak przemieszczanie w sieci drogowej, w otwartym terenie, itp. Jedną z najbardziej znanych i podstawowych metod jest wykorzystanie drzew czwórkowych w celu wielorozdzielczej reprezentacji rozpatrywanego terenu. Drzewa te stanowią rekursywny podział dwuwymiarowej mapy (istnieje również wersja trójwymiarowa – drzewa ósemkowe) na homogeniczne obszary o rozmiarach $2^i q \times 2^i q$ (q to długość boku najmniejszego pola) (Rys. 1). Taka struktura danych zapewnia

efektywną reprezentację terenu i szybsze działanie algorytmów wyszukujących drogi. Niestety słabo nadaje się do reprezentacji innego terenu, niż binarny (tzn. reprezentującego wyłącznie przejezdne (1) i nieprzejezdne (0) obszary).



Rys. 1. Model terenu i odpowiadające mu drzewo czwórkowe
Fig. 1. Terrain model with its quadtree representation

Występują trzy rodzaje węzłów takiego drzewa – *węzeł wolny* (*free node*) to obszar dostępny dla ruchu, w przeciwieństwie do *węzła przeszkody* reprezentującego obszar nieprzejezdny. *Węzły szare* to obszary zawierające zarówno węzły wolne i przeszkody. W normalnym drzewie liście zawsze są węzłami prostymi lub przeszkodami. Czasami jednak odstępuje się od tej zasady, a wynikowe drzewo nazywa się drzewem *przyciętym* (*pruned*). Drzewo przycięte reprezentuje taką samą powierzchnię terenu, ale z mniejszą dokładnością. Jeden z możliwych algorytmów wykorzystujących drzewa czwórkowe przedstawiony został w [7]. Metoda tam opisana kładzie nacisk na utrzymanie przemieszczanego agenta możliwie daleko od przeszkód. Jej zaletą jest efektywne przetwarzanie mapy rastrowej do drzewa czwórkowego w czasie liniowym (względem liczby pikseli w obrazie). Istnieją również metody poprawiające jakość dróg znalezionych przy zastosowaniu drzew czwórkowych (np. poprzez otoczenie dużych pól ramkami złożonymi z minimalnych obszarów [2, 3]).

Drzewa czwórkowe ze względu na hierarchiczny charakter dobrze nadają się do wyszukiwania etapowego. W pierwszym kroku przycina się drzewo do wybranego poziomu i znajduje wstępną drogę. Następnie stopniowo zwiększa się rozdzielczość (przez uwzględnienie wżerów niższego poziomu) i uszczegóławia pierwotne rozwiązanie. Wyszukiwanie kończy się w momencie dotarcia do ostatniego poziomu lub na dowolnie wybranej rozdzielczości (jeżeli pełna szczegółowość nie jest wymagana).

W [12] zaprezentowano inną metodę służącą do planowania ruchu robota w otwartym terenie reprezentowanym przez mapę wysokości. Głównym kryterium wyszukiwania drogi jest jej możliwie łagodny przebieg (aby unikać miejsc niemożliwych lub niebezpiecznych do pokonania prze agenta). Algorytm składa się z czterech głównych kroków:

- wstępne przygotowanie modelu,
- uwzględnienie przeszkód i innych dodatkowych aspektów terenu,
- wyznaczenie drogi w najmniejszej rozdzielczości,
- uszczegółowianie drogi na kolejnych poziomach.

W pierwszym kroku, przy zastosowaniu transformaty falkowej, teren dzielony jest na hierarchiczne warstwy (będące aproksymacjami modelu wyrażonego jako funkcja wysokości $z = f(u,v)$ na kolejnych poziomach rozdzielczości). Ponieważ zmniejszanie rozdzielczości modelu powoduje powstawanie błędów, z każdym punktem wiąże się współczynnik wielkości tego błędu (jak bardzo teren w tej rozdzielczości różni się od oryginalnego). Przy wyborze drogi brane są pod uwagę te współczynniki tak, że droga prowadzona jest po obszarach, których zmiana rozdzielczości zbytnio nie „zepsuła" (tzn. zmniejsza się ryzyko, że np. po zwiększeniu rozdzielczości w miejscu którędy biegnie droga pojawiłaby się nagle głęboka szczelina). Wykorzystuje się w tym celu współczynniki rozwinięcia falkowego mierzące stopień *surowości* poszczególnych obszarów terenu. Poprzez teren *surowy* rozumie się taki obszar, który w wyniku zmniejszenia rozdzielczości istotnie się zmienił – posiadał wiele gwałtownych spadków i wzniesień, które zostały „wygładzone". Jego przeciwieństwem jest teren *gładki* – taki, którego postać w niskiej rozdzielczości nie zmienia się znacząco. Oznacza to, że przy zwiększeniu szczegółowości modelu w tym miejscu nie pojawią się niespodziewanie miejsca trudniejsze do pokonania przez agenta ze względu na uformowanie terenu. Stopień surowości terenu jest głównym czynnikiem wpływającym na koszt przemieszczania przez taki obszar. Preferowane są obszary *gładkie* – nie niosące ryzyka wyznaczenia drogi przez teren o niesprzyjającej charakterystyce. Następnie przygotowany model wzbogaca się o dodatkowe ograniczenia w postaci przeszkód i innych dodatkowych aspektów wpływających na ruch agenta i wyznaczaną drogę.

Kosztem drogi jest wektor zawierający uporządkowane w kolejności nierosnącej koszty przejścia przez poszczególne pola wchodzące w jej skład. Oznacza to, że dwie wartości kosztu $\{u_1,u_2,\ldots,u_l\} < \{v_1,v_2,\ldots,v_m\}$ w dwóch przypadkach:

- istnieje $j \in N$, takie, że $u_j < v_j$ oraz dla każdego $i < j \quad u_i = v_i$

lub

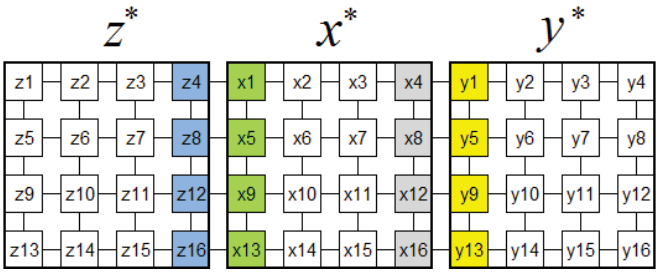
- $l < m$ oraz $u_i = v_i$ dla $1 < i < l$.

Wyszukanie drogi na początkowym poziomie (o najniższej rozdzielczości) odbywa się przy wykorzystaniu standardowych algorytmów (np. Dijkstry zatrzymywanego po znalezieniu punktu docelowego). Następnie droga taka jest uszczegółowiana w coraz wyższej rozdzielczości. Przy przechodzeniu na kolejny poziom, brane są pod uwagę tylko pola leżące w sąsiedztwie pól odpowiadających tym, przez które przechodzi droga na poprzednim poziomie, co znacząco przyspiesza obliczenia. Po wyznaczeniu drogi na określonym poziomie następuje zejście do większej rozdzielczości. W pierwszym kroku droga p przekształcana jest w kanał P o zadanym marginesie m. Oznacza to, że jeżeli p jest drogą na poziomie l, to komórka $(u, v) \in P$ jeżeli istnieje komórka $(i, j) \in p$ oraz $|u - i| \le m$ i $|v - j| \le$ m. W kolejnym kroku pola, które nie należą do kanału oznaczane są jako przeszkody, zatem nie biorą już udziału w wyszukiwaniu drogi na kolejnych poziomach. W zależności od parametru m kanał biorący udział w kolejnych przeszukiwaniach będzie miał różną szerokość. W zależności od potrzeb wyszukiwanie nie musi dochodzić do samego dołu – może zatrzymać się na dowolnym poziomie, co oczywiście wiąże się z mniej szczegółowym rozwiązaniem.

Inny problem rozwiązuje metoda przedstawiona w [1]. Geneza jej powstania związana jest jednoznacznie z wojskiem – opracowana została przez U.S. Army Topographic Engineering Center oraz Uniwersytet Stanowy Luisiany w celu znajdowania dróg dla modeli terenu o postaci siatek prostokątnych o bardzo dużych rozmiarach. Zatem główny nacisk położony został na efektywnościowy aspekt stosowania metod wielorozdzielczych. Punktem wyjściowym jest model terenu w postaci siatki (o dużej rozdzielczości, niekoniecznie kwadratowej). Na podstawie tego modelu tworzy się graf (podobny do diagramu Voronoi (patrz np. w [15]), czyli że z każdą przeszkodą wiąże się decyzja czy ominąć ją z lewej, czy z prawej). Liczba węzłów w tym grafie powinna być wielokrotnie mniejsza niż pól w siatce (rzędu 1 do 1000). Wagi krawędzi tego grafu wyznaczane są poprzez wyznaczenie drogi pomiędzy jego wierzchołkami na poziomie siatki. Planowanie przemieszczania odbywa się na dwóch poziomach hierarchii. W pierwszej kolejności odnajdowana jest droga w grafie. Następnie następuje zejście na poziom siatki i odnajdowana jest dokładna droga pomiędzy wierzchołkami znalezionymi wcześniej (i wyznaczany końcowy koszt). W następnym kroku następuje relaksacja drogi. Polega ona na zdefiniowaniu nowych wierzchołków (w grafie wyższego poziomu) leżących w połowie drogi między oryginalnymi. Następnie znajdowana jest droga pomiędzy nowymi wierzchołkami. Tak znaleziona droga, która nie przechodzi dokładnie przez wierzchołki grafu zwykle jest tańsza o około 10 procent. Wyszukiwanie drogi na obu poziomach odbywa się z wykorzystaniem algorytmu A*. Metoda ta została zaimplementowana w module Hierachical Route Planner stanowiącym część systemu PIMTAS (ang. Predictive Intelligence Military Tactical Analysis System).

Nieco inne podejście do wyszukiwania dróg w modelach o postaci siatki kwadratowej można odnaleźć w [16]. Zaprezentowano tam algorytm DSP (ang. Decomposition Shortest Paths) polegający na łączeniu sąsiadujących pól siatki (węzłów grafu) w grupy stanowiące węzły (nazywane *b-węzłami*) grafu wyższego poziomu (*b-grafu*).

Na wejściu algorytmu znajduje się graf $G=(V, A)$ reprezentujący siatkę kwadratową modelującą określony teren, gdzie V jest zbiorem węzłów, natomiast $A=\{(x, y) \subseteq V \times V\}$ to zbiór łączących ich krawędzi. Kolejnym krawędziom $(x, y) \in A$ przyporządkowana jest wartość funkcji kosztu $c(x, y)$ ($c(x, x)=0$, $c(x,u)=+\infty$ gdy $(x, y) \notin A$) reprezentująca np. czas przejścia z węzła x do y. W ramach działania metody utworzony zostaje nowy graf G^* (zwany *b-grafem*) powstały poprzez łączenie sąsiadujących węzłów w większe grupy (*b-węzły*). Następnie te b-węzły, między którymi można bezpośrednio przejść po ich węzłach składowych łączone są *b-krawędziami*.



Rys. 2. Węzły brzegowe b-węzłów: W(z,x*)={z4,z8,z12,z16}, W(x*z*)={x1,x5,x9,x13}, W(x*,y*)={x4,x8,x12,x16}, W(y*,x*)={y1,y5,y9,y13}*

Fig. 2. Border nodes for b-nodes: W(z,x*)={z4,z8,z12,z16}, W(x*z*)={x1,x5,x9,x13}, W(x*,y*)={x4,x8,x12,x16}, W(y*,x*)={y1,y5,y9,y13}*

Formalnie graf G^* opisać można jako $G^*=(V^*, A^*)$, gdzie $V^*=\{x^*_1, x^*_2, \ldots, x^*_n\}$ to zbiór b-węzłów $(x^*_i=\{x_{i1}, x_{i2}, \ldots, x_{im_i}\}, \quad i=1,\ldots,n)$, natomiast $A^*=\left\{(x^*, y^*)\subset V^*\times V^*: \exists_{x\in x^*, y\in y^*}(x,y)\in A\right\}$ stanowi zbiór b-krawędzi. Postać funkcji kosztu w b-grafie jest znacznie bardziej złożona niż w grafie pierwotnym. Z każdą b-krawędzią związane są wartości dwóch funkcji kosztu – $c^{*\min}(x^*, y^*)$ oraz $c^{*\max}(x^*, y^*)$. Wartości tych funkcji mają postać wektora, o elementach reprezentujących koszt przejścia pomiędzy węzłami x^* i y^* dla wszystkich węzłów mogących być poprzednikami x^* w ścieżce, z czego wynika, że koszt ten jest różny w zależności od tego, z jakiego wcześniejszego b-węzła został osiągnięty x^*; $c^{*\min}(x^*, y^*)$ to minimalny koszt drogi, spośród najkrótszych dóg w grafie G pomiędzy węzłami należącymi do x^* i y^* dla różnych poprzedników x^*. Formalnie zapisać można

$$c^{*\min}(x^*, y^*)=\left\langle c^{*\min}_{z^*}(x^*, y^*)\right\rangle_{z^*\in\{v^*\in V^*:(v^*,x^*)\in A^*\}}.$$

Analogicznie $c^{*\max}(x^*, y^*)$ jest maksymalnym kosztem spośród takich dróg, czyli

$$c^{*\max}(x^*, y^*)=\left\langle c^{*\max}_{z^*}(x^*, y^*)\right\rangle_{z^*\in\{v^*\in V^*:(v^*,x^*)\in A^*\}}.$$

Aby objaśnić zasadę działania tych funkcji, należy zastosować dodatkowe oznaczenia. Niech $W(x^*, y^*)=\left\{x\in x^*: \exists_{y\in y^*}(x,y)\in A\right\}$, będzie zbiorem węzłów należących do x^*, przyległych do dowolnego z węzłów składowych y^* (Rys. 2), natomiast $D^{\min}(W(x^*, z^*), W(y^*, v^*))$ oznacza zbiór najkrótszych dróg, dla których węzeł początkowy należy do $W(x^*, z^*)$, natomiast docelowy do $W(y^*, v^*)$. Droga pomiędzy węzłami w grafie bazowym oznaczana będzie jako $d(x,y)$ (a jej długość: $L(d(x,y))$), natomiast w b-grafie - $d^*(x^*, y^*)$.

Wykorzystując wcześniejsze oznaczenia można formalnie zapisać:

$$c^{*\min}_{z^*}(x^*, y^*)=\min_{d(\cdot,\cdot)\in D^{\min}(W(x^*,z^*),W(x^*,y^*))} L(d(\cdot,\cdot))+\min_{d(\cdot,\cdot)\in D^{\min}(W(x^*,y^*),W(y^*,x^*))} L(d(\cdot,\cdot))$$

oraz

$$c^{*\max}_{z^*}(x^*, y^*) = \max_{d(\cdot,\cdot)\in D^{\min}(W(x^*,z^*),W(x^*,y^*))} L(d(\cdot,\cdot)) + \max_{d(\cdot,\cdot)\in D^{\min}(W(x^*,y^*),W(y^*,x^*))} L(d(\cdot,\cdot))$$

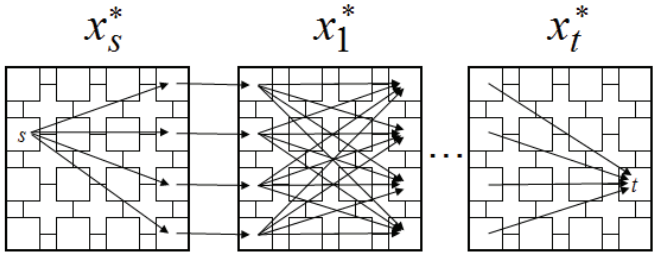
Suma kosztów przejść pomiędzy kolejnymi b-węzłami stanowi łączny koszt drogi w b-grafie. Do wyszukiwania takiej drogi można użyć wybranej z dwóch funkcji kosztu w b-grafie - $c^{*\min}(x^*, y^*)$ lub $c^{*\max}(x^*, y^*)$. Algorytm ten ma dodatkową bardzo interesującą cechę – w zależności od wybranej funkcji kosztu, koszt drogi w b-grafie stanowi ograniczenie dolne (dla $c^{*\min}(x^*, y^*)$) lub górne (dla $c^{*\max}(x^*, y^*)$) kosztu drogi optymalnej w grafie bazowym (dowód w [16]).

Działanie algorytmu DSP można podzielić na dwie fazy: konstrukcji grafu *G** (kroki 1-3) oraz znalezienie drogi pomiędzy wymaganymi węzłami (kroki 4-5). Cele te realizowane są w pięciu krokach.

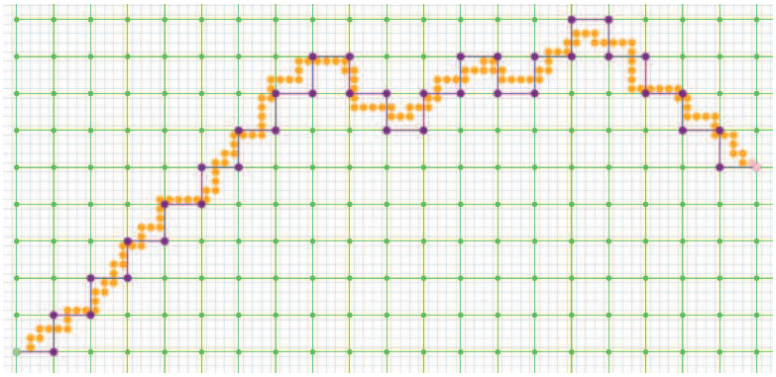
1. Podział grafu wejściowego *G* na *n* podgrafów poprzez połączenie pól siatki w większe kwadraty (*n* jest parametrem wejściowym algorytmu).
2. Wyznaczenie silnie spójnych składowych z podgrafów otrzymanych w poprzednim kroku, otrzymując co najmniej *n* podgrafów (b-węzłów) następnie wyznaczenie b-krawędzi tworząc w ten sposób graf *G** (b-graf).
3. Wyznaczenie drzew najkrótszych dóg (SPT) wewnątrz podgrafów wyznaczonych przez b-węzły pomiędzy wszystkimi parami wierzchołków brzegowych sąsiadujących z innymi b-węzłami (tzn. z których można przejść do podgrafu innego b-węzła). Następnie wyznaczenie dla każdej b-krawędzi kosztów $c^{min}(*,*)$ i $c^{max}(*,*)$.
4. Znalezienie najkrótszej drogi wewnątrz b-grafu *G** pomiędzy b-węzłami zawierającymi w sobie wierzchołki początkowy i końcowy szukanej drogi z grafu G. Wyszukiwanie odbywa się stosując koszty c^{min} lub c^{max}.
5. Uszczegółowienie drogi w grafie *G*. Można to zrobić konstruując acykliczny graf skierowany (DAG) złożony z odpowiednich węzłów z podgrafów należących do b-węzłów ze drogi znalezionej w kroku 4 (Rys.3).

Acykliczny graf skierowany (DAG) z kroku 5 ma budowę warstwową. Dla drogi $d^*(x^*_s, x^*_t) = (x^*_s, x^*_1, x^*_2 \ldots, x^*_{t-1}, x^*_t)$ pierwsza warstwa będzie się składać z łuków wychodzących z węzła początkowego *s* do węzłów z $W(x^*_s, x^*_1)$. Następnie z $W(x^*_s, x^*_1)$ do $W(x^*_1, x^*_s)$, potem z $W(x^*_1, x^*_s)$ do $W(x^*_2, x^*_1)$, itd. Ostatecznie z $W(x^*_t, x^*_{t-1})$ łuki kierują się do węzła końcowego *t* (Rys. 3).

Przykład drogi znalezionej przez algorytm DSP pokazano na Rys. 4. Bazowa siatka (graf *G*) zaznaczona jest najdrobniejszą podziałką. Najmniejsze kropki (kółka) oznaczają węzły b-grafu *G**, szarymi dużymi kropkami (kółkami) oznaczona jest droga w sieci bazowej (grafie *G*) natomiast czarnymi dużymi kropkami – droga w b-grafie *G**.



Rys. 3. DAG skonstruowany w kroku 5 algorytmu DSP
Fig. 3. DAG constructed in the 5th step of the DSP algorithm



Rys. 4. Przykład drogi znalezionej przez algorytm DSP
Fig. 4. An example of path found by DSP algorithm

Jeżeli istnieje droga pomiędzy węzłem początkowym i końcowym, to zostanie ona znaleziona przez algorytm DSP, jednak jej optymalność nie jest zapewniona. W [16] udowodniono, że złożoność tego algorytmu w zadaniu znalezienia drogi pomiędzy dwoma węzłami wynosi $O\left(\sqrt{\frac{V^3}{n}}\log_k \frac{V}{n} + n\log_k n\right)$, gdzie V to liczba węzłów w grafie G, n – liczba węzłów w grafie G^*, natomiast $k = \max\{2, \lceil A/V \rceil\}$ (A to liczba łuków w grafie G).

Algorytm Hierarchiczny (opisany w [4]) powstał w celu przyspieszenia wyszukiwania dróg w dużych sieciach modelujących sieci drogowe. Punktem wyjścia jest model w postaci sieci (reprezentujący typową sieć ulic). Dzielona jest ona na niskopoziomowe podsieci (zwane mikrosieciami). Jeżeli dwa punkty, pomiędzy którymi wyznacza się drogę znajdują się w tej samej podsieci, wówczas szukanie ograniczane jest tylko do węzłów z tej podsieci (nawet jeżeli optymalna droga dla całej sieci przebiegałaby na jakimś odcinku przez inną podsieć). Jeżeli punkty znajdują się w innych podsieciach korzysta się z dodatkowej wysokopoziomowej sieci, nazywanej makrosiecią (będącej

podsiecią sieci oryginalnej). Poszczególne wierzchołki tej sieci należą do kolejnych małych podsieci. Jeżeli podsieci można by utożsamiać z sieciami małych dróg lokalnych to makrosieć reprezentuje autostrady i drogi szybkiego ruchu. Znalezienie najkrótszej drogi pomiędzy węzłami z różnych podsieci polega na odnalezieniu drogi z mikrosieci początkowej do węzła z makrosieci, wyszukaniem drogi w obrębie makrosieci, a następnie w obrębie mikrosieci końcowej. Ruch pomiędzy mikrosieciami może odbywać się więc tylko za pośrednictwem makrosieci.

Rys. 5. Sieć drogowa z naniesioną makrosiecią
Fig. 5. Road network with macronetwork

Niech $G=(V,A,C)$ będzie skierowanym grafem silnie spójnym o zbiorze węzłów *V*, krawędzi *A* oraz nieujemnej funkcji kosztu krawędzi $C:A\to R^{+}\cup\{0\}$. Wyodrębnienie makrosieci polega na utworzeniu nowej sieci (makrosieci) $\tilde{G}=(\tilde{V},\tilde{A},\tilde{C})$, $\tilde{V}\subseteq V$ (makrowęzły), $\tilde{A}\subseteq\tilde{V}\times\tilde{V}$ (makrokrawędzie), oraz $\tilde{C}:\tilde{A}\to R^{+}\cup\{0\}$. Każdej makrokrawędzi $(I,J)\in\tilde{A}$ odpowiada droga w *G*. Następnie graf *G* dzieli się na na *H* silnie spójnych mikrosieci $G_h=(V_h,A_h,C_h)$, gdzie $V_h\subseteq V$, $V=\bigcup_h V_h$, $A_h=A\cap(V_h\times V_h)$ oraz $\tilde{V}\cap V_h\neq\phi$ dla $h=1,\ldots,H$. Wyszukiwanie dróg odbywa się w pierwszym kroku w makrosieci $\tilde{G}$ i poszczególnych mikrosieciach G_h. Dalej $\tilde{f}:\tilde{V}\times\tilde{V}\to R$ będzie oznaczać funkcję zwracającą koszt najkrótszej drogi w makrosieci $\tilde{G}$ oraz $f_h:V_h\times V_h\to R$ funkcję zwracającą koszt najkrótszej drogi w mikrosieci G_h, $h=1,\ldots,H$.

Jeżeli dwa węzły $i\in V_h$ oraz $j\in V_l$ znajdują się w innych mikrosieciach, wówczas wyznaczona droga będzie składać się z: najkrótszej drogi w G_h od *i* do wybranego makrowęzła, najkrótszej drogi w makrosieci $\tilde{G}$ od *I* do wybranego makrowęzła $J\in\tilde{V}\cap V_l$ oraz najkrótszej drogi od makrowęzła *J* do *j*.

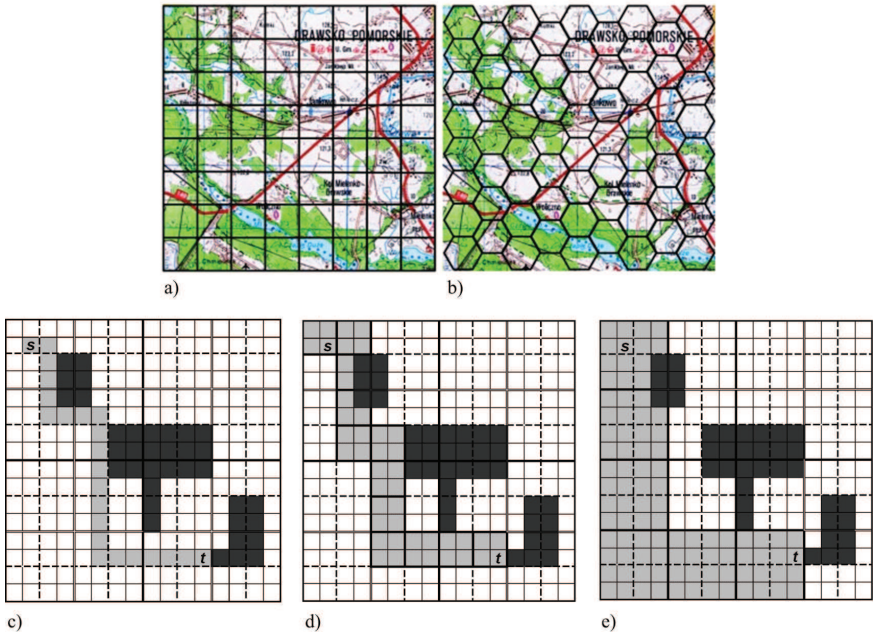
Jeżeli w mikrosieci znajduje się kilka makrowęzłów, należy zdecydować, z którego skorzystać w celu wejścia do i wyjścia z makrosieci. Sposób dokonywania tego wyboru ma kluczowe znaczenie dla tego algorytmu. Wyróżnia się dwa podejścia do tego problemu. W pierwszym wybierany jest makrowęzeł znajdujący się najbliżej mikrowęzła początkowego/końcowego. Jeżeli $i \in V_h$ oraz $j \in V_l$ to węzły początkowy i końcowy, wówczas makrowęzłem początkowym będzie $I^* = \arg\min_{I \in V_h \cap \tilde{V}} f_h(i, I)$ natomiast końcowym - $J^* = \arg\min_{J \in V_l \cap \tilde{V}} f_l(J, j)$. Można jednak postąpić w bardziej wyrafinowany sposób: wybrać taki makrowęzeł, dla którego długość znalezionej trasy jest najmniejsza. Wówczas zostaną następujące wybrane mikrowierzchołki: $(I^*, J^*) = \arg\min_{(I,J) \in (V_h \cap \tilde{V}) \times (V_l \cap \tilde{V})} \{f_h(i, I) + \tilde{f}(I, J) + f_l(J, j)\}$. Wariant algorytmu wykorzystujący pierwsze podejście nosi nazwę *Nearest HA*, natomiast drugie – *Best HA*.

Może się zdarzyć, że makrowęzeł początkowy lub końcowy nie należą tylko do jednej mikrosieci (ponieważ nie muszą być one rozłączne). W takiej sytuacji wyboru makrowęzła dokonuje się ze wszystkich mikrosieci zawierających dany mikrowęzeł.

Jak łatwo się domyślić, wariant algorytmu ma wpływ na jego złożoność obliczeniową. Niech $\Theta(N)$ będzie oszacowaniem liczby węzłów sieci G, $\Theta(N^m)$, $1 < m < 1$ - liczby makrowęzłów, natomiast $\Theta(N^k)$, $0 < k < 1$ to oszacowanie liczby mikrosieci. Ponieważ w każdej mikrosieci musi być przynajmniej jeden makrowęzeł, dlatego $m \geq k$. W 4 wykazano, że w zadaniu znalezienia drogi pomiędzy wszystkimi parami wierzchołków algorytm w wariancie *Nearest HA* cechuje się złożonością $C_N = O(N^2)$, co jest rezultatem lepszym , od najszybszej wersji algorytmu Dijkstry, który w takiej sytuacji ma złożoność $C_D = O(N^2 \log N)$ (N razy wykonywane jest zadanie wyszukiwania drogi z jednym źródłem, jednak w przeciwieństwie do Algorytmu Hierarchicznego algorytm Dijkstry znajduje zawsze optymalne rozwiązanie). *Best HA* jest już bardziej wymagający, ponieważ potrzebuje do tego $C_B = O(N^{2+2(m-k)})$ (należy pamiętać o warunku $m \geq k$). Zatem tylko dla $m = k$ algorytm ten jest szybszy od algorytmu Dijkstry. Należy brać to pod uwagę projektując strukturę makro- i mikrosieci.

4 Zastosowanie WAPT w wielorozdzielczej symulacji pola walki

Wielorozdzielczy model terenu jest naturalnym modelem odwzorowania środowiska (terenu) dla struktury hierarchicznej jednostek na symulowanym polu walki. Dla szczebla kompanii wymagane jest bardziej precyzyjne odwzorowanie terenu niż, np. dla szczebla brygady. W symulacji pola walki stosuje się wiele modeli terenu. Najbardziej popularne są dwie reprezentacje: regularna siatka kwadratów terenu (Rys. 6a) i regularna siatka heksagonów terenu (Rys. 6b).



Rys. 6. Przykłady reprezentacji terenu w symulacji pola walk: a) regularna siatka kwadratów; b) regularna siatka heksagonów. Wielorozdzielcza droga najkrótsza z s do t z wykorzystaniem algorytmu DSP w G^*: c) $G=G^*$ stąd G^* zawiera 16×16 b-wierzchołków; d) G^* zawiera 8×8 b-wierzchołków; e) G^* zawiera 4×4 b-wierzchołki

Fig. 6. Examples of terrain representation in a simulated battlefield: a) regular grid of terrain squares; b) regular grid of terrain hexagons. Multiresolution shortest path from s to t using DSP algorithm in G: c) G=G* thus G* contains 16x16 nodes; d) G* contains 8x8 nodes; e) G* contains 4x4 nodes*

Zaleta pierwszego odwzorowania (kwadratowego) jest widoczna zwłaszcza w kontekście wielorozdzielczości (patrz Rys. 6c÷e). Rozmiar kwadratu terenu może być dynamicznie zmieniany i może zależeć od wymaganego szczebla jednostek. Kwadrat o większym rozmiarze niż kwadrat bazowy może być definiowany jako kwadratowa macierz kwadratów bazowych (dla przykładu, na Rys. 6d każdy kwadrat ma rozmiar 2x2 kwadraty bazowe). Taka reprezentacja nie jest możliwa dla heksagonów, tak więc siatka kwadratów jest bardziej użyteczna z punktu widzenia wielorozdzielczego modelowania terenu i planowania tras. Na Rys. 6c÷e podano przykład wyznaczania tras w trzy-poziomowym grafie: (c) pierwszy poziom jest najbardziej szczegółowy (bazowy); (d) drugi poziom jest dwa razy mniej szczegółowy niż pierwszy; (e) trzeci poziom jest cztery razy mniej szczegółowy niż pierwszy. Modele te mogą opisywać szczeble np. plutonu, kompanii i batalionu. Zauważmy, że bardzo łatwo możemy otrzymać wielorozdzielczy model terenu definiując graf G^* rekurencyjnie. Jeśli założymy, że graf G definiuje model terenu pierwszego poziomu (np. szczebla kompanii) wówczas G^* definiuje model drugiego (lub wyższego) poziomu (np. szczebla batalionu). To podejście może być wykorzystane

do zwiększania lub zmniejszania wymaganej rozdzielczości modelu. Parametr n algorytmu DSP ($n \in \{1,\ldots,V\}$) może być użyty do wpływania na wymiar (rozdzielczość) grafu G^*. Następnie algorytm DSP może być wykorzystany do poszukiwania wielorodzielczych tras w takim wielorozdzielczym modelu terenu. Dla przykładu, na Rys. 6c $G^*=G$ i zawiera n=256 b-wierzchołków (np. dla szczebla plutonu), na Rys. 6d G^* zawiera n=64 b-wierzchołków (np. dla szczebla kompanii) a na Rys. 6e G^* zawiera n=16 b-wierzchołków (np. dla szczebla batalionu).

Warto podkreślić, że prezentowane podejście różni się od bardzo efektywnej metody reprezentacji terenu w postaci drzewa czwórkowego [7] z dwóch głównych powodów: (1) elementy (kwadraty) drzewa czwórkowego, które reprezentują teren są różnej wielkości, (2) w większości zastosowań (dla przykładu w [7]) drzewo czwórkowe reprezentuje tylko teren „binarny" z dwoma rodzajami obszarów (kwadratów): otwarty (przejezdny) i zamknięty (nieprzejezdny). Takie podejście jest bardzo efektywne dla mobilnych robotów, ale nie odzwierciedla w sposób wystarczająco adekwatny środowiska pola walki [15].

Zauważmy również, że wielorozdzielcze podejście do planowania tras reprezentowane przez poszukiwanie najkrótszych dróg w rekurencyjnie definiowanym grafie G^* może być wykorzystane do wieloetapowego planowania: najpierw znajdujemy "zgrubną" drogę $d^{*\min}(x_s^*, x_t^*)$ (lub $d^{*\max}(x_s^*, x_t^*)$) w "zgrubnym" modelu terenu reprezentowanym przez G* (dla przykładu na Rys.3e) a następnie poszukujemy dokładnej drogi w bardziej szczegółowym modelu terenu (reprezentowanym przez G z kwadratami bazowymi, Rys.3c; bardziej precyzyjnie: znajdujemy najkrótszą drogę z s do t wewnątrz podgrafu generowanego przez wierzchołki grafu G należące do b-wierzchołków drogi $d^{*\min}(x_s^*, x_t^*)$ (lub ($d^{*\max}(x_s^*, x_t^*)$, patrz krok 5 algorytmu DSP). Jest to przykład zastosowania modelowania „od ogółu do szczegółu".

5 Analiza eksperymentalna wybranych algorytmów planowania tras wielorozdzielczych

Przeprowadzone zostały badania charakterystyk wybranych spośród opisanych wcześniej algorytmów. Eksperymenty podzielone były na dwie części. W pierwszej zaimplementowano i przeprowadzono pomiary dla algorytmów DSP oraz Algorytmu Hierarchicznego. Następnie przeprowadzono porównanie obu tych algorytmów posługując się tymi samymi danymi wejściowymi.

W celu zbadania algorytmu DSP wykorzystano losowo wygenerowany model sieci w formie siatki kwadratowej o boku 200 węzłów (łącznie 40000 węzłów). Losowe z krawędzi pomiędzy węzłami zostały usunięte, a innym nadano losowo wartość funkcji kosztu z przedziału <1; 4>. Ostatecznie sieć posiadała 119574 krawędzie. Badania przeprowadzono dla różnych funkcji kosztu ($c^{\min}$ i $c^{\max}$) oraz różnych wartości parametru n. Na potrzeby tych badań parametr n rozumiany był jako długość boku pojedynczego b-węzła (wyrażona jako liczba węzłów do niego należąca). Zatem dla n=1 każdy b-węzeł zawierał dokładnie jeden węzeł (czyli $G^*=G$), natomiast dla n=200 G^* składa się z jednego b-węzła zawierającego cały graf G. Pomiary przeprowadzono dla 500 losowo wybranych par węzłów sieci. Dla porównania, jednocześnie wykonywano poszukiwanie algorytmem A* znajdującym rozwiązania optymalne. Pozwoliło to na oszacowanie błędu

algorytmu DSP oraz różnicy w czasie działania z typowym algorytmem znajdującym rozwiązania optymalne. Wyniki pomiarów znajdują się w Tabeli 1.

Tabela 1. Wyniki algorytmu DSP

Table 1. Results of the DSP algorithm

n	Czas gen. [s]	Liczba b-węzłów	Liczba b-kraw.	Czas DSP [s]		Czas A* [s]		Błąd [%]	
				c^{min}	c^{max}	c^{min}	c^{max}	c^{min}	c^{max}
1	3,95	39843	119574	107,12	106,01	38,74	36,48	0	0
2	1,93	12978	43024	30,31	32,14	34,49	36,69	6,44	12,21
3	1,66	6383	22050	14,23	14,09	32,97	33,44	12,12	14,65
4	1,39	3791	13202	9,06	8,32	34,32	31,67	16,75	14,88
5	1,50	2519	8748	6,05	5,57	33,54	32,38	20,98	14,11
10	2,66	748	2384	2,90	2,15	32,00	33,40	25,22	9,77
20	4,48	241	676	2,75	1,7	35,51	31,18	21,09	8,39
50	10,65	60	140	7,16	5,52	32,64	33,00	15,46	4,41
100	16,93	20	40	17,93	18,13	32,44	32,95	0,56	0,61

Kolumny tabeli zawierają kolejno: rozmiar boku podsieci kwadratowej użytej do wygenerowania b-węzłów, czas przygotowania modelu danych (kroki 1 – 3 algorytmu), liczbę wygenerowanych b-węzłów i b-krawędzi, następnie czasy wyszukiwania wszystkich dróg algorytmem DSP (kroki 4 – 5) i A* oraz błąd długości drogi liczony jako względna różnica pomiędzy długością drogi znalezionej algorytmem DSP, a optymalną. Analogiczne pomiary przeprowadzono dla sieci posiadającej wszystkie krawędzie pomiędzy sąsiadującymi węzłami o koszcie z zakresu <1; 5>. Zestawione zostały w Tabeli 2.

Tabela 2. Wyniki algorytmu DSP działającego na pełnej siatce

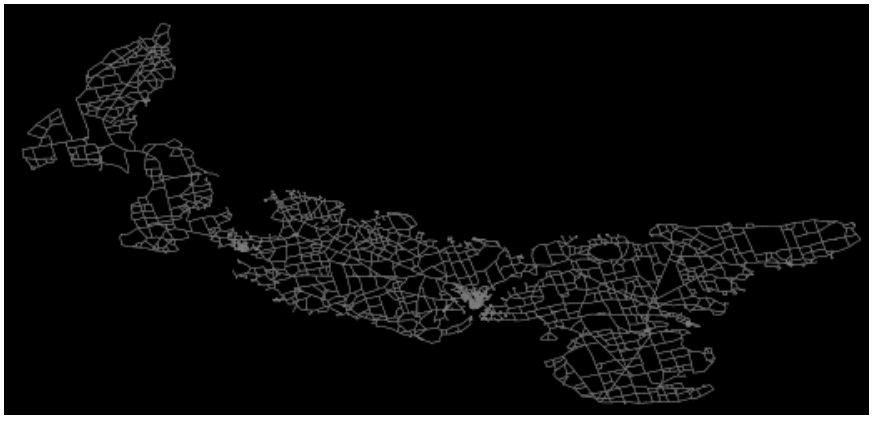
Table 2. Results of the DSP algorithm working on a full network

n	Czas gen. [s]	Liczba b-węzłów	Liczba b-kraw.	Czas DSP [s]		Czas A* [s]		Błąd [%]	
				c^{min}	c^{max}	c^{min}	c^{max}	c^{min}	c^{max}
1	4,98	40000	159200	108,11	107,46	33,12	35,83	0	0
2	1,89	10000	39600	25,92	26,08	37,17	35,19	12,68	9,85

3	1,69	4489	17688	11,43	10,85	37,95	38,6	25,99	9,59
4	1,8	2500	9800	6,62	6,43	36,18	36,77	27,86	7,75
5	1,91	1600	6240	4,49	4,17	32,13	34,21	25,56	6,88
10	3,38	400	1520	3,15	2,04	37,72	32,82	19,99	5,45
20	6,2	100	360	4,11	2,71	36,64	38,39	16,5	4,47
50	14,74	16	48	9,59	7,43	36,93	34,38	18,32	2,5
100	24,92	4	8	19,03	19,32	33,85	32,72	4,38	0,54

Wyniki pomiarów pokazują, że parametr n ma duży wpływ na działania algorytmu. Zarówno wartości ekstremalnie duże i ekstremalnie małe powodują istotne pogorszenie efektów działania tej metody. Ujawnia tu się też dość duża różnica w dokładności algorytmu przy wykorzystaniu funkcji $c^{\min}$ i $c^{\max}$. W 6 znajduje się próba omówienia przyczyn tego zjawiska.

Badania Algorytmu Hierarchicznego przeprowadzono przy wykorzystaniu sieci zawierającej 7079 węzłów i 18954 krawędzie modelujących przykładową sieć drogową jednej z prowincji Kanady (Rys. 7).



Rys. 7. Sieć bazowa dla Algorytmu Hierarchicznego

Fig. 7. Base network for Hierarchical Algorithm

Następnie na powyższą sieć nałożono odpowiednią strukturę hierarchiczną w postaci wyznaczonych mikrosieci i makrosieci. Wydzielona makrosieć składała się z 169 makrowęzłów i 362 makrokrawędzi (Rys. 8). Podzieliła ona całą sieć na 38 mikrosieci o średnim rozmiarze 202 węzłów (należy pamiętać, że mikrowęzły mogą należeć do więcej niż jednej mikrosieci) oraz przeciętnie 7 makrowęzłach przypadających na pojedynczą mikrosieć. Algorytm zaimplementowany do badań działał w dwóch fazach.

Pierwszą jest faza inicjacji, w której dla makrosieci oraz każdej mikrosieci budowane jest drzewo najkrótszych dróg (SPT) z wykorzystaniem algorytmu Dijkstry z pojedynczym źródłem. Właściwe drogi szukane są w fazie drugiej. Dzięki wcześniej wyznaczonym drzewom polega ona tylko na złożeniu odpowiednich dróg w mikrosieciach i makrosieci. Szukanie zrealizowano dla 10000 losowo wybranych par węzłów. Jednocześnie działał algorytm A* umożliwiając porównanie wyników. W oparciu o ten algorytm liczony był błąd Algorytmu Hierarchicznego, rozumiany jako względna różnica pomiędzy długością wyznaczonej drogi a długością drogi optymalnej, wyrażona w procentach. Wyniki znajdują się w Tabeli 3. Jak można było przewidzieć, algorytm w wersji NearestHA jest bardzo szybki i dość niedokładny, natomiast w wersji BestHA generuje już dość dobre drogi w czasie wciąż znacznie szybszym niż algorytm optymalny.

Rys. 8. Makrosieć uzyskana z sieci bazowej
Fig. 8. The macronetwork obtained from the base network

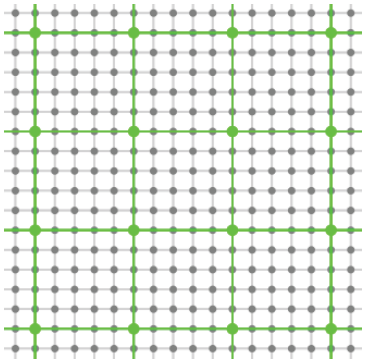
Tabela 3. Wyniki Algorytmu Hierarchicznego
Table 3. Results of the Hierarchical Algorithm

Charakterystyka	NearestHA	BestHA
Łączny czas [s]	3,97	14,69
Faza I [s]	3,59	3,99
Faza II [s]	0,38	10,70
Średni błąd [%]	21,64	9,05
Czas działania A* [s]	110,99	115,34

Na koniec przeprowadzona została próba porównania zbadanych wcześniej algorytmów. W tym celu posłużono się drugą z sieci, którą wykorzystano do badania algorytmu DSP.

Sieć ta składa się z 40000 węzłów i 159200 krawędzi. Wartości kosztu krawędzi mają rozkład równomierny z przedziału <1; 5>. Na potrzeby Algorytmu Hierarchicznego, na sieć tą nakładana była makrosieć (o charakterze siatki kwadratowej). O odległości (liczonej w krawędziach) pomiędzy kolejnymi makrokrawędziami decydował parametr *d* (Rys. 9). Czas generacji tej sieci został umieszczony w wynikach pomiarów. Badania wykonano dla 500 losowo wybranych par węzłów.

Wyniki pomiarów pokazują, że Algorytm Hierarchiczny jest znaczenie szybszy od DSP, ale jednocześnie rozwiązania przez niego znalezione znacznie bardziej odbiegają od rozwiązań optymalnych. Badania te pokazują jednak również dużą wrażliwość tych metod na odpowiednią konfigurację. Niewłaściwy dobór parametrów (np. *n* dla DSP lub *d* dla Algorytmu Hierarchicznego) drastycznie pogarsza jakość rozwiązań, a w skrajnych okolicznościach zupełnie niweluje jakiekolwiek zalety stosowania takich metod.



Rys. 9. Makrosieć skonstruowana dla d=5
Fig. 9. Macronetwork constructed for d=5

Tabela 4. Porównanie algorytmu DSP (a) i Algorytmu Hierarchicznego (b)
Table 4. The comparison of DSP (a) and Hierarchical Algorithm (b)

a)

n	**Czas gen. [s]**	**Liczba b-węzłów**	**Liczba b-kraw.**	**Czas DSP [s]**		**Czas A* [s]**		**Błąd [%]**	
				c^{min}	c^{max}	c^{min}	c^{max}	c^{min}	c^{max}
4	1,8	2500	9800	6,62	6,43	36,18	36,77	27,86	7,75
5	1,91	1600	6240	4,49	4,17	32,13	34,21	25,56	6,88
10	3,38	400	1520	3,15	2,04	37,72	32,82	19,99	5,45
20	6,2	100	360	4,11	2,71	36,64	38,39	16,5	4,47

b)

d	Czas gen. sieci [s]	Czas fazy I	Czas HA [s]		Czas A* [s]		Błąd [%]	
			NearestHA	BestHA	NearestHA	BestHA	NearestHA	BestHA
4	168,30	30,40	0,08	0,21	39,17	40,90	25,42	21,44
5	105,70	15,63	0,09	0,19	39,32	41,54	28,65	22,68
10	28,16	12,67	0,09	0,15	40,19	40,46	38,83	22,89
20	7,80	39,40	0,08	0,63	38,58	39,04	44,60	19,30

Porównując te dwie metody należy również wspomnieć, że algorytm DSP jest bardziej tolerancyjny w stosunku do modelu wejściowego. Algorytm Hierarchiczny do prawidłowego działania wymaga, aby zarówno cały graf wejściowy, wszystkie mikrosieci i makrosieć były silnie spójne. W innych okolicznościach może nie znajdować rozwiązań, pomimo, że w rzeczywistości istnieją.

6 Podsumowanie

Metody wspierające planowanie i symulację przemieszczania wykorzystujące wielorozdzielcze modele terenu stanowią przydatne narzędzie, gdy problem cechuje się dużą skalą lub hierarchicznym charakterem, jak w przypadku symulacji pola walki. Podejście prezentowane w pracy jest przeznaczone przede wszystkim do wielorozdzielczego planowania dróg w wielorozdzielczym modelu terenu reprezentowanym przez regularną siatkę kwadratów. Jest to element wielorozdzielczej symulacji pola walki, który jest ściśle związany z planowaniem i symulacją działań. Pokazano również, że wielorozdzielcze podejście do planowania tras reprezentowane przez poszukiwanie najkrótszych dróg w rekurencyjnie definiowanym grafie G^* może być wykorzystane do wieloetapowego planowania tras: najpierw znajdujemy "zgrubną" drogę w "zgrubnym" modelu terenu reprezentowanym przez G^*, a następnie poszukujemy dokładnej drogi w bardziej szczegółowym modelu terenu. Algorytm DSP daje dobre wyniki nie tylko dla problemu wyznaczania dróg między każdą parą wierzchołków (Tabela 1). Ponieważ kroki 1-3 algorytmu o największej złożoności ("wąskie gardło") wykonywane są tylko raz (konstruujemy b-graf G^* tylko raz – przetwarzanie wstępne), więc jeśli wyznaczamy drogi najkrótsze między pojedynczymi parami wierzchołków wielokrotnie wówczas zmniejszamy wpływ „wąskiego gardła" na czas obliczeń. Ponadto w pracy [17] pokazano, że algorytm DSP bardzo dobrze nadaje się do zrównoleglenia obliczeń.

Podziękowania

Praca częściowo finansowa z projektów: MNiSW OR00005506 pt. „Symulacja działań bojowych w heterogenicznym środowisku wielorozdzielczej i rozproszonej symulacji" oraz MNiSW OR00005006 pt. „Integracja systemów dowodzenia".

Literatura:

1. Benton J.R., Iyengar S. S., Deng W., Brener N., Subrahmanian V. S.: *Tactical Route Planning: New Algorithms for Decomposing the Map*, International Journal on Artificial Intelligence Tools, vol. 5 (1-2), 1996
2. Chen D. Z., Szczerba R. J., Uhran Jr. J. J.: *Using Framed-Quadtrees to Find Conditional Shortest Paths in an Unknown 2-D Environment*, Technical Report: #95-2, Uniwersytet Notre Dame, Indiana, USA, 1995
3. Chen D. Z., Szczerba R. J., Uhran Jr. J. J.: *Using Framed-Octrees to Find Conditional Shortest Paths in an Unknown 3-D Environment*, Technical Report: #95-9, Uniwersytet Notre Dame, Indiana, USA, 1995
4. Chou Y., Romeijn H. E., Smith R. L.: *Approximating Shortest Paths in Large-scale Networks with an Application to Intelligent Transportation Systems*, INFORMS Journal on Computing Vol. 10, Issue 2 (1998) s. 163 – 179
5. Djidjev H., Pantziou G., Zaroliagis C.D.: *On-line and dynamic algorithms for shortest path problems,* Lecture Notes in Computer Science, vol.900 (1995), 193-204.
6. Godlewski P.: *Modelowanie i algorytmizacja procesów planowania i symulacji przemieszczania z wykorzystaniem wielorozdzielczych modeli terenu*, praca magisterska pod kierunkiem Z.Tarapaty, Wojskowa Akademia Techniczna, Warszawa, 2010.
7. Kambhampati S., Davis L. S.: *Multiresolution Path Planning for Mobile Robots*, IEEE Journal of Robotics and Automation, Vol. RA-2, nr. 3, 1986
8. Korf R.E.: *Artificial intelligence search algorithms*, in Algorithms Theory Computation Handbook, Boca Raton, FL: CRC Press (1999)
9. LaValle S. M.: *Planning Algorithms*, Cambridge University Press 2006
10. Magillo P., Bertocci V.: *Managing Large Terrain Data Sets with a Multiresolution Structure*, INFORMS Journal on Computing Vol. 10, nr 2, 1998 s. 163 - 179
11. Mitchell J. S. B.: *An Algorithmic Approach to Some Problems in Terrain Navigation*, Artificial Intelligence Vol. 37, nr 1-3 (1988) s. 171 – 201
12. Pai D. K., Reissell L.M.: *Multiresolution Rough Terrain Motion Planning*, IEEE Transactions on Robotics and Automation, Vol. 14, nr 1, 1998, s. 19 – 33
13. Petty M.D.: *Computer generated forces in Distributed Interactive Simulation*, Proceedings of the Conference on Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment, 19-20 April, Orlando (USA) (1995), 251-280
14. Reece D., Kraus M., Dumanoir P.: *Tactical Movement Planning for Individual Combatants*, Proceedings of the 9th Conference on Computer Generated Forces and Behavior Representation, Orlando, Floryda, 2000
15. Tarapata Z.: *Military route planning in battlefield simulation: effectiveness problems and potential solutions*, Journal of Telecommunications and Information Technology, nr 4 (2003), s. 47-56
16. Tarapata Z.: *Multiresolution models and algorithms of movement planning and their application for multiresolution battlefield simulation*, ACIIDS 2010, Lecture Notes in Computer Science/LNAI, 5991 (2010) Springer-Verlag Berlin Heidelberg, s. 378-389
17. Tarapata Z.: *A Parallel Decomposition Algorithm for Shortest Path Problem in Large-Size Mesh Networks*, Biuletyn WAT, vol. LIX, Nr 3, 2010, 295-306

Streszczenie

W pracy zaprezentowano przegląd modeli i metod poszukiwania dróg w wielorozdzielczych sieciach oraz ich analizę pod kątem efektywnościowym (dokładności i czasu obliczeń). Jedna z opisywanych metod opiera się na „agregowaniu" geograficznym sąsiednich wierzchołków (kwadratów terenu) w grafie, który reprezentuje teren w postaci tzw. kraty i planowaniu tras w „zagregowanej" sieci z wykorzystaniem specyficznej transformacji. Druga z metod wykorzystywana jest do planowania wielorozdzielczych tras w sieci drogowej. Opisano zastosowanie prezentowanych metod w wielorozdzielczej symulacji pola walki.

Multiresolution models and algorithms of movement planning and their application for multiresolution battlefield simulation

Summary

In the paper a review of models and methods for finding paths in multiresolution networks and their effectiveness analysis have been presented. One of the methods is based on merging the geographically adjacent nodes (squares) and the planning path to a "merged" graph. The merging is done by using geographically adjacent squares of primary graph (thus, we obtain nodes of a "merged" graph) and calculating costs in the "merged" graph as longest (or shortest) of the shortest paths between some subsets of nodes belonging to "merged" nodes. Second method is applied for planning multiresolution paths in roads network. Application of presented methods in multiresolution battlefield movement planning and simulation is discussed.