Zbigniew TARAPATA

Wojskowa Akademia Techniczna
Wydział Cybernetyki, Instytut Systemów Informatycznych
ul. Gen. Sylwestra Kaliskiego 2, 00-908 Warszawa
zbigniew.tarapata@wat.edu.pl

OPARTE NA WIEDZY MODELE I ALGORYTMY PLANOWANIA ORAZ SYMULACJI PRZEMIESZCZANIA W SYSTEMACH WOJSKOWYCH KLASY DSS I CGF

Streszczenie:

W pracy zaprezentowano modele i algorytmy planowania oraz symulacji przemieszczania wielu obiektów w wojskowych systemach klasy DSS i CGF bazujących na wiedzy. Studium przypadku dotyczy idei oraz modelu decyzyjnego i sterowania związanego z automatem decyzyjnym do marszu na szczeblu batalionu. Automat realizuje dwa główne procesy: planowania decyzji do marszu i bezpośredniego sterowania przebiegiem (symulacją) marszu. Rozpatrzono wybrane aspekty teoretyczne i praktyczne modelowania i optymalizacji procesu planowania marszu, symulacji i sterowania jego przebiegiem. Opisano sposób implementacji automatu oraz wybrane wyniki symulacji w interaktywnym systemie wspomagania szkolenia operacyjnego *Złocień*.

Słowa kluczowe: planowanie tras, symulacja przemieszczania, synchronizacja przemieszczania, systemy wspomagania decyzji, algorytmy transportowe

WPROWADZENIE

Jednymi z najistotniejszych problemów transportowych spotykanych w literaturze są problemy związane z planowaniem przemieszczania. Problemy te są istotne nie tylko w zastosowaniach wojskowych, ale również w: sieciach komputerowych, mobilnych robotach, systemach ewakuacji, systemach nawigacji samochodowej, grach komputerowych [8]. W zastosowaniach wojskowych problemy te spotykane są zarówno w symulatorach pola walki (wyznaczanie tras przed rozpoczęciem symulacji działań, jak i w trakcie ich trwania), jak również w systemach wspomagania decyzji, które wspierają zautomatyzowane systemy dowodzenia klasy *C4ISR* (*Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance*) [3], [5], [7], [9], [10], [14], [15].

Opisywane systemy powinny bazować na wiedzy, a modele i algorytmy wspomagania decyzji oraz symulacji powinny z tej wiedzy korzystać [4], [13]. Wiedza jest przetworzoną, na podstawie pewnych reguł, informacją pochodzącą z różnego rodzaju tematycznych baz danych (o terenie, regulaminów działań taktycznych, uzbrojenia i sprzętu wojskowego, struktur wojsk, itp.) i obejmuje takie elementy, jak: przejezdność wskazanych fragmentów terenu, wzorce sytuacji decyzyjnych, wzorce wariantów działań dla różnych rodzajów działań, itd. Symulacja oparta na wiedzy, czyli z wykorzystaniem modeli i metod sztucznej inteligencji, rozwijana jest od późnych lat siedemdziesiątych i wczesnych osiemdziesiątych poprzedniego wieku, a pionierem tych badań była RAND Corporation w USA. W szczególności dotyczy to systemów klasy *CGF* (*Computer Generated Forces*) lub *SAF*/*SAFOR* (*Semi-Automated Forces*), w których komputer zarządza zachowaniem symulowanych obiektów jednej lub obu stron konfliktu na wirtualnym polu walki [4], [12]. 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". Wszystkie procesy (strzelanie, przemieszczanie, wykrywanie, itp.) realizowane przez taki automat są bardzo złożone i mają wpływ na dokładność i adekwatność modelu symulacyjnego. Jednym z elementów, które w tego typu systemach są istotne, są moduły planowania tras dla wirtualnych obiektów symulowanego/generowanego pola walki [3], [9], [14], [18]. W rozdziałach 1 oraz 3 pracy [19], Autor uzasadnia potrzebę stosowania tego typu rozwiązań powołując się szeroko na literaturę związaną z systemami agentowymi (*agent-based simulation systems*), które są podstawą systemów klasy *CGF*. Systemy wspomagania decyzji (*Decision Support Systems* (*DSS*)) oparte na wiedzy związane są przede wszystkim z systemami eksperckimi [13].

W niniejszej pracy zostaną zaprezentowane elementy automatu decyzyjnego do marszu, który realizuje dwa główne procesy: planowania decyzji do marszu i sterowania (zarządzania) przebiegiem symulacji marszu. Proces planowania decyzji do marszu obejmuje: organizację marszu (ustalenie liczby kolumn marszowych jednostek, ustalenie porządku jednostek w tych kolumnach, wyznaczenie liczby i miejsc postojów), wyznaczenie optymalnych dróg dla jednostek, wyznaczenie szczegółowego harmonogramu marszu (optymalizującego osiągnięcie pewnych punktów synchronizacji) dla każdej jednostki z kolumn marszowych. Problemy planowania zsynchronizowanego przemieszczania znane są w literaturze, np. w planowaniu wszelkiego rodzaju rozkładów jazdy (pociągów, autobusów, samolotów, mieszanych środków komunikacji). W pracy wykorzystuje się zaproponowane modele (rozdział 2) i metody (rozdział 3) do planowania marszu wielu kolumn/obiektów wojskowych na wirtualnym polu walki. Proces zarządzania symulacją marszu (rozdział 4) obejmuje: dowodzenie, meldowanie oraz reakcję na sytuacje awaryjne powstałe w czasie symulacji marszu (np. pojawienie się przeciwnika na trasie marszu, niemożność przemieszczania się zaplanowaną trasą, itd.). Automat został zaimplementowany w języku ADA i reprezentuje dowódcę szczebla batalionu (najniższym szczeblem ćwiczących wojsk jest szczebel brygady). Stanowi element składowy rozproszonego, interaktywnego systemu symulacyjnego SSWSO *Złocień* przeznaczonego do przeprowadzania komputerowych ćwiczeń sztabów wojsk [1], [11]. System *Złocień* został zbudowany i wdrożony w Siłach Zbrojnych RP przez zespół z Wydziału Cybernetyki WAT.

1. MODEL TERENU OPARTY NA WIEDZY

Model terenu S_0, którego używać będziemy, jako modelu środowiska pola walki (stosowany w systemie *Złocień*), bazuje na informacjach przetworzonych z cyfrowej mapy terenu w formacie *VPF* (*Vector Product Format*). Model składa się z dwóch sieci: Z_1 w postaci regularnej siatki kwadratów terenu, która reprezentuje wybrany fragment terenu oraz Z_2 reprezentującej sieć drogowo-kolejową [15]:

$$S_O(t) = \langle Z_1(t), Z_2(t) \rangle \tag{1}$$

Regularna siatka kwadratów Z_1 dzieli obszar zainteresowania na kwadraty o tych samych wymiarach (200m×200m), a każdy z kwadratów jest jednorodny z punktu widzenia charakterystyk terenu (stopień osłabienia prędkości, zdolność do maskowania, stopień widoczności, itd.). Sieć Z_1 używana jest do planowania i symulacji przemieszczania na przełaj, np. podczas planowania i symulacji natarcia. W sieci Z_2 wierzchołkami są skrzyżowania, a łukami – odcinki dróg. Model ten używany jest do planowania szybkiego przemieszczania po drogach, np. podczas planowania i symulacji marszu. Oba te modele zależą od czasu i mogą być reprezentowane przez graf Berge'a G definiujący strukturę terenu

lub/i sieci drogowo-kolejowej, $G = \langle V_G, A_G \rangle$, V_G – zbiór wierzchołków grafu (V_G=W_1 dla Z_1 opisuje środki kwadratów terenu, a V_G=W_2 dla Z_2 opisuje skrzyżowania), A_G – zbiór łuków grafu, $A_G \subset V_G \times V_G$. Zakładamy, że na każdym łuku grafu zdefiniowana jest funkcja $d : V_G \times V_G \to R^+ \cup \{0\}$, która opisuje odległość geograficzną między dwoma wierzchołkami, funkcja $f^{slowd} : V_G \times V_G \to [0,1]$, która opisuje stopień osłabienia prędkości między dwoma wierzchołkami wynikający z warunków topograficznych (postać tej funkcji bazuje na pewnych regułach opartych o wiedzę i doświadczenie ekspertów) i funkcja *FCam* opisująca zdolność do maskowania w kwadracie (w chwili $t \in T$), *FCam*: $V_G \times T \to [0,1]$. Model ten jest bardziej rozbudowany (tutaj przedstawiamy jedynie jego niezbędny do dalszych rozważań fragment), a jego szczegóły można znaleźć w [15], [19]. Stosuje się również wielorozdzielcze modele terenu, a ich opis można znaleźć, np. w [19].

2. PLANOWANIE PRZEMIESZCZANIA OBIEKTÓW NA PRZYKŁADZIE MARSZU

Proces planowania marszu związany z automatem decyzyjnym zawiera wyznaczenie takich elementów, jak: organizacja marszu (kolejność jednostek w kolumnie, liczba i miejsca postojów), optymalne trasy dla jednostek, szczegółowy harmonogram marszu dla każdej jednostki. Proces planowania marszu startuje w chwili *t*, kiedy batalion *id* otrzymuje rozkaz do marszu *SO*(*id*, *t*) od przełożonego (dowódcy brygady). Struktura *SO*(*id*, *t*) jest następująca:

$$SO(id,t) = \left(t_0(id,t), t_S(id,t), MD(id,t)\right) \tag{2}$$

gdzie: *SO*(*id*, *t*) – rozkaz przełożonego batalionu *id* do marszu; $t_0(id,t)$ - czas gotowości dla jednostki *id*; $t_S(id,t)$ - chwila rozpoczęcia marszu przez jednostkę *id*; $MD(id,t)$ - szczegółowy opis rozkazu do marszu. Definicja $MD(id)$ (pomijamy *t*) jest następująca:

$$MD(id) = \left\langle S(id), D(id), RP(id), IP(id) = \left(in_p(id), it_p(id)\right)_{p=\overline{1,NIP}} \right\rangle \tag{3}$$

gdzie: $S(id), D(id)$ - odpowiednio, rejony wyjściowy i docelowy dla *id*; *RP*(*id*) – rejon odpoczynku dla *id* (po 24. godzinach marszu), opcjonalny; *IP*(*id*) – wektor punktów pośrednich dla *id* (trasa marszu musi przechodzić przez te punkty), in_p(*id*) – *p*-ty punkt pośredni, $in_p(id) \in W_1 \cup W_2$, in_1(*id*)=*PS*(*id*) jest punktem wyjściowym marszu (w tym punkcie formuje się kolumna marszowa) i jest wymagany, inne punkty pośrednie są opcjonalne, it_p(*id*) – moment osiągnięcia *p*-tego punktu pośredniego (opcjonalny); *NIP* – liczba punktów pośrednich. Po otrzymaniu przez batalion *id* brygadowego rozkazu do marszu, automat decyzyjny rozpoczyna planowanie decyzji do marszu dla *id*. Biorąc pod uwagę $SO(id,t)$, dla każdej jednostki *id'* (szczebla kompanii i równoważnego) bezpośrednio podległej batalionowi *id* wyznaczana jest przez automat decyzja do marszu *MDS*(*id'*):

$$MDS(id') = \left\langle S(id'), D(id'), PS(id'), PD(id'), RP(id'), \mu\left(id', S(id'), D(id')\right)\right\rangle \tag{4}$$

gdzie: $S(id'), D(id')$ - odpowiednio, rejony wyjściowy i docelowy dla *id'*, $S(id') \subset S(id)$, $D(id') \subset D(id)$; *RP*(*id'*) – rejon odpoczynku dla *id'* (po 24. godzinach marszu), $RP(id') \subset RP(id)$, parametr opcjonalny; *PS*(*id'*) – punkt wyjściowy dla *id'*, taki sam dla wszystkich *id'*$\in$*id* oraz $PS(id') = in_1(id) \in W_1 \cup W_2$; *PD*(*id'*) – punkt docelowy dla *id'*, taki sam dla wszystkich *id'*$\in$*id* oraz $PD(id') \in W_1 \cup W_2$; $\mu(id', S, D)$ - droga dla *id'* z obszaru

S(*id'*)=*S* do obszaru *D*(*id'*)=*D*, $\mu(id',S,D)=\left(w(id',m),v(id',m)\right)_{m=\overline{1,LW(\mu(id',S,D))}}$, $w(id',m)$ - *m*-ty wierzchołek na drodze dla *id'*, $w(id',m)\in W_1\cup W_2$, $S,D\subset W_1\cup W_2$ i $w(id',1)\in S$, $w\left(id',LW\left(\mu(id',S,D)\right)\right)\in D$; *LW*($\mu$(*id'*, *S*, *D*)) – liczba wierzchołków (kwadratów lub/i skrzyżowań) na drodze μ(*id'*,*S*,*D*) dla *id'*; $v(id',m)$ - prędkość dla *id'* na łuku rozpoczynającym się w *m*-tym wierzchołku drogi. Warto podkreślić, że droga $\mu(id',S,D)$ może składać się z ciągów wierzchołków sieci $Z_1(t)$ oraz $Z_2(t)$ (kiedy dopuszczamy zejście z drogi na kwadraty (jeżeli jest to możliwe) i na odwrót).

Organizacja marszu zawiera wyznaczenie takich elementów, jak: liczba kolumn, kolejność jednostek w kolumnie marszowej oraz liczbę i miejsca postojów. Liczba (#) kolumn wynika z regulaminów taktycznych i zależy od szczebla dowodzenia: dla batalionu #kolumn=1, dla brygady #kolumn$\in\{1,2,3\}$; dla dywizji #kolumn$\in\{3,4,5\}$. Kolejność jednostek w kolumnie marszowej wynika również z regulaminów taktycznych (patrz Tabela 2, algorytm *Units_Order_In_March_Column_Determ*(*id'*)). Liczba postojów $c_{stops}(id)$ wyliczana jest następująco (algorytm *Number_of_Stops_Determ*(*id'*), Tabela 2):

$$c_{stops}(id)=\max\left\{\left\lfloor\frac{\left(t_D(id,t)-t_S(id,t)-t_{rest}(id)\right)\cdot v_{avg}(id)-L_{path}(id)}{v_{avg}(id)\cdot\left(t_{stop}(id)+\Delta s\right)}\right\rfloor,0\right\} \tag{5}$$

gdzie: $t_D(id,t)$ - pożądana chwila zakończenia marszu przez *id*, $t_S(id,t)$ - chwila rozpoczęcia marszu dla *id* (jak w (2)), $t_D(id,t)>t_S(id,t)\geq 0$, $t_{rest}(id)$ - czas trwania odpoczynku dla *id*, $v_{avg}(id)$ - średnia prędkość marszu dla *id*, $L_{path}(id)$ - długość drogi wyznaczonej dla *id* (w km), $t_{stop}(id)$ - czas trwania postoju dla *id*, Δs - odstęp czasu między postojami. W praktyce, wartości parametrów są następujące: $t_{rest}(id)\approx$24h, $v_{avg}(id)\in[30,40]$ km/h, $t_{stop}(id)\approx 1$ h, $\Delta s\in[3,4]$ h. Miejsca postojów wyznaczane są po ustaleniu dróg, a algorytm *Place_Of_Stops_Determ(id')* (Tabela 2) bierze pod uwagę $c_{stops}(id)$ i funkcję *FCam* (z rozdziału 2), aby wyznaczyć optymalne miejsca postojów.

Szczegółowy harmonogram przemieszczania dla jednostki *id'* jest definiowany następująco:

$$H(id',t_0)=\left\langle S,D,\mu(id',S,D),T(id',S,D)\right\rangle \tag{6}$$

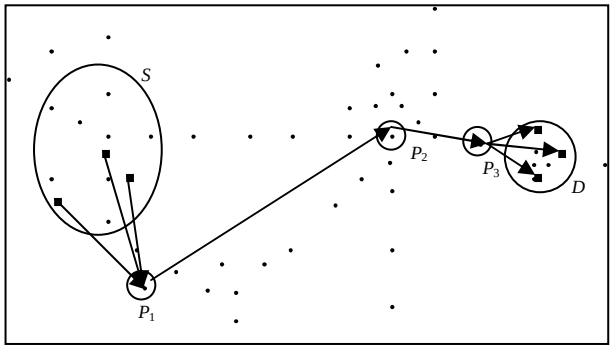
gdzie: t_0 – moment rozpoczęcia realizacji harmonogramu; $T(id',S,D)$ - wektor chwil osiągnięcia wierzchołków drogi dla *id'*, $T(id',S,D)=\left\langle t(id',m)\right\rangle_{m=\overline{1,LW(\mu(id',S,D))}}$, $t(id',m)$ - chwila osiągnięcia *m*-tego wierzchołka z drogi dla *id'*,

$$t(id',m)=t_0+\sum_{j=1}^{m-1}\frac{L\left(w(id',j),w(id',j+1)\right)}{v(id',j)} \tag{7}$$

gdzie *L*(*w*(*id'*,*j*),*w*(*id'*,*j*+*1*)) opisuje odległość geometryczną między *j*-tym i (*j*+1)-szym wierzchołkiem drogi, $LW(\mu(id',S,D))$ - liczba wierzchołków drogi dla *id'*. Po wyznaczeniu *MDS*(*id'*) dla każdej jednostki *id'* podległej batalionowi *id*, wypracowana decyzja jest przesyłana do każdej z tych jednostek. Idea wyznaczania trasy marszu dla jednostki *id* została zaprezentowana na Rys.1 i jest realizowana przez procedurę *March_Schedule_Determ*(*id'*).

W ogólności, automat wykorzystuje dwie kategorie kryteriów dla harmonogramu zsynchronizowanego przemieszczania *K* kolumn obiektów (jednostek). Dla uproszczenia

przyjmijmy, że batalion *id* jest równoważny *k*-tej kolumnie, *k*=1,…,*K*, tzn. $k \equiv id$. Ponadto, przyjmijmy następujące oznaczenia: $I_k(s_k,t_k) = I_k = (i^0(k) = s_k,\ i^1(k),...,i^r(k),...,\ i^{R_k}(k) = t_k)$ - wektor wierzchołków drogi *k*-tego obiektu, $s_k \in S$, $t_k \in D$, $i^r(k)$ - *r*-ty wierzchołek drogi *k*-tego obiektu, $\tau^r(k)$ - chwila osiągnięcia wierzchołka $i^r(k)$ przez czoło *k*-tego obiektu, $v_{i^r(k),i^{r+1}(k)}$ - prędkość *k*-tego obiektu na łuku $\left(i^r(k),i^{r+1}(k)\right)$ jego drogi, $d_{i^r(k),i^{r+1}(k)}$ - odległość terenowa między wierzchołkami $i^r(k)$ oraz $i^{r+1}(k)$, R_k - liczba łuków należących do drogi I_k.



Rys. 1. Przykład trasy marszu dla trzech jednostek *id'*∈ *id* (wypełnione kwadraty) z rejonu wyjściowego *S* do rejonu docelowego *D* (kropki reprezentują skrzyżowania dróg). Zdefiniowano trzy punkty pośrednie: P_1=*PS*, P_2 i P_3=*PD* (drogi dla wszystkich jednostek muszą przechodzić przez te punkty). P_1 jest punktem wyjściowym marszu (w tym punkcie formowana jest kolumna marszowa rozważanych trzech jednostek), P_3 jest punktem końcowym marszu (w tym punkcie kolumna marszowa jest rozwiązywana), P_2 jest innym punktem pośrednim marszu. Droga między P_1 i P_3 jest wspólna dla wszystkich jednostek, jednakże każda z jednostek ma inną drogę ze swojego podobszaru w *S* do P_1 oraz z P_3 do podobszaru w *D*.

Źródło: opracowanie własne.

Pierwszą kategorią kryteriów jest czas przemieszczania *K* obiektów z dwoma podstawowymi funkcjami:

$$\max_{k\in\{1,...,K\}} \tau^{R_k}(k) \qquad \text{lub} \qquad \sum_{k=1}^{K} \tau^{R_k}(k) \tag{8), (9}$$

Drugą kategorią kryteriów jest "odległość" między czasami osiągnięcia punktów wyrównania przez wszystkie *K* obiekty. Definiujemy dwie funkcje w ramach tej kategorii:

$$\sum_{p=1}^{NIP}\sum_{k=1}^{K} \tau_p^{\max} - \tau_p(k) \qquad \text{lub} \qquad \min_{p\in\{1,...,NIP\}} \max_{k\in\{1,...,K\}} \left(\tau_p^{\max} - \tau_p(k)\right) \tag{10), (11}$$

gdzie: $\tau_p(k)$ chwila osiągnięcia *p*-tego wierzchołka wyrównania ($in_p(id)$ z (3)),

$$\tau_p(k) = \tau^0(k) + \sum_{\substack{r\in\{0,...,R_k-1\} \\ r\le r_p(k)}} \frac{d_{i^r(k),i^{r+1}(k)}}{v_{i^r(k),i^{r+1}(k)}} \tag{12}$$

i $r_p(k) = r \in \{1,...,R_k\} \Leftrightarrow in_p(k) = i^r(k)$, $\tau_p^{\max} = \max_{k\in\{1,...,K\}} \tau_p(k)$. Biorąc pod uwagę, że jednostka *id* odpowiada *k*-temu obiektowi możemy zapisać: $k \equiv id$, $v_{i^r(k),i^{r+1}(k)} \equiv v(k,r)$, $i^r(k) \equiv w(k,r)$, $d_{i^r(k),i^{r+1}(k)} \equiv L\left(w(k,r),w(k,r+1)\right)$, $r_p(k) = in_p\left(id\right)$.

Jeden z problemów synchronizacji przemieszczania *K* obiektów wykorzystujący miary (8)-(9) może być zdefiniowany następująco: dla ustalonych dróg I_k dla każdego *k*-tego obiektu wyznaczyć takie $v_{i^r(k),i^{r+1}(k)}$, $r=\overline{0,R_k-1}$, $k=\overline{1,K}$, że

$$\sum_{p=1}^{NIP}\sum_{k=1}^{K}\left(\tau_p^{\max}-\tau_p(k)\right)\to\min \tag{13}$$

przy ograniczeniach:

$$v_{i^r(k),i^{r+1}(k)}\le v^{\max}(k),\qquad r=\overline{0,R_k-1},\ \ k=\overline{1,K} \tag{14}$$

$$v_{i^r(k),i^{r+1}(k)}>0,\qquad r=\overline{0,R_k-1},\ \ k=\overline{1,K} \tag{15}$$

gdzie $v^{\max}(k)$ oznacza maksymalną prędkość *k*-tego obiektu wynikającą z jego parametrów technicznych.

Pewne rozszerzenia opisywanego problemu oraz metody ich rozwiązania zawiera rozdział 3.

3. ROZSZERZENIA PROBLEMU PLANOWANIA PRZEMIESZCZANIA ORAZ METODY ICH ROZWIĄZANIA

Biorąc po uwagę planowanie tras dla wielu obiektów możemy rozpatrywać różne rozszerzenia problemu (13)-(15):

(a) dodając następującą grupę ograniczeń,

$$\tau^0(k)+\sum_{r\in\{0,\ldots,R_k-1\}}\frac{d_{i^r(k),i^{r+1}(k)}}{v_{i^r(k),i^{r+1}(k)}}\le\tau^*,\quad k=\overline{1,K} \tag{16}$$

poszukiwać będziemy takiego harmonogramu przemieszczania, w którym chwila osiągnięcia wierzchołka docelowego przez najwolniejszy obiekt jest nie większa, niż chwila optymalna τ^* (lub pewna ustalona chwila $T^0\ge\tau^*$);

(b) możemy poszukiwać zarówno dróg I_k oraz prędkości $v_{i^r(k),i^{r+1}(k)}$, $r=\overline{0,R_k-1}$, $k=\overline{1,K}$; drogi dla *K* obiektów muszą być rozłączne lub nie (b1) albo muszą przechodzić przez z góry ustalone punkty (wierzchołki) lub te punkty wyliczane są dynamicznie (b2). W pierwszym przypadku (b1) mamy do czynienia z problemem NP-trudnym obliczeniowo, który możemy rozwiązać stosując pewne algorytmy przybliżone dla znajdowania dróg rozłącznych, np. zawarte w [19]. Autor wspomnianej monografii [19] definiuje pewne modyfikacje klasycznych problemów oraz opisuje aproksymacyjny algorytm rozwiązania tych problemów w sieci typu krata bazującej na danych pochodzących z cyfrowych baz terenu. Skupia się przy tym na najtrudniejszej (z punktu widzenia złożoności obliczeniowej) wersji problemu, rzadziej występującej w literaturze, związanej z wyznaczaniem *K*>1 dróg rozłącznych między wskazanymi *K* parami węzłów, dodatkowo z zadanymi punktami pośrednimi, przez które drogi muszą przechodzić. Porównuje otrzymane rozwiązania z rozwiązaniami uzyskanymi innymi metodami, np. rozwiązując zadanie programowania liniowego algorytmem simpleks (zaimplementowanym w solverze GAMS/CPLEX). W drugim przypadku (b2) możemy zastosować podejście dwuetapowe: (*) znajdując iteracyjnie najlepsze drogi dla *K* obiektów używając metod znajdowania *m*-tej (1-szej, 2-giej, 3-ciej, itd.) najkrótszej drogi dla każdego

z *K* obiektów; (**) synchronizując przemieszczanie *K* obiektów poprzez rozwiązanie problemu (13)-(15) z użyciem algorytmów opisanych, np. w [17];

(c) możemy sformułować zadanie optymalizacji dwukryterialnej używając jednego z kryteriów (8)-(9) oraz jednego z kryteriów (10)-(11) i rozwiązując je z wykorzystaniem jednej z metod opisanych, np. w [16], [19].

Szczegółowy opis modeli i algorytmów harmonogramowania zsynchronizowanego przemieszczania wielu obiektów zawarty jest w [17], [19]. W celu poszukiwania dróg dla obiektów, w systemie *Złocień* stosowanych jest szereg zmodyfikowanych algorytmów poszukiwania dróg najkrótszych (ADN) takich, jak Dijkstry, A* [6], geometryczny ADN [11]. Algorytm wyznaczania dróg geometrycznych uzupełnia dwa klasyczne algorytmy (Dijkstry, A*) i jest używany w przypadku dużych rozmiarów sieci (domyślnie powyżej 10000 wierzchołków, ale jest to parametr ustalany w tzw. kalibratorze systemu [2]). Modyfikacje wspomnianych algorytmów dotyczą następujących rozszerzeń: (a) wyznaczania dróg
w różnych konfiguracjach – (a1) z punktu (rejonu) do punktu (rejonu); (a2) przechodzące przez wybrane punkty (rejony); (a3) omijające wybrane punkty (rejony, przeszkody); (a4) wewnątrz lub na zewnątrz pewnego rejonu; (a5) tylko po drogach; (a6) tylko „na przełaj”; (a7) po drogach i „na przełaj”; (b) jeżeli nie zostanie podany obszar wewnątrz którego ma być poszukiwana droga, to algorytm sam dobiera iteracyjnie obszar prostokątny bazujący na prostej łączącej punkt startowy z docelowym, w celu minimalizacji czasu obliczeń; (c) jeżeli wymagamy znalezienia drogi po sieci drogowej, a w podanych wierzchołkach (kwadratach), przez które droga ma przechodzić nie ma żadnego węzła sieci drogowej, to algorytm może zawsze poszukiwać (jeśli wybierzemy taką opcję) węzłów drogowych najbliższych kwadratom, przez które ma przechodzić droga na kierunku od źródła do celu [15]; (d) wyznaczania dróg dwukryterialnych [16].

W ogólności, modelowanie i optymalizacja przemieszczania wielu obiektów (w celu zsynchronizowanego ich przemieszczania) są procesami bardzo skomplikowanymi. Złożoność tych procesów zależy od wielu czynników: liczby konwojów (im większa liczba konwojów tym problem harmonogramowania bardziej złożony); liczby obiektów w każdym konwoju (im dłuższy konwój, tym bardziej skomplikowane harmonogramowanie); Czy konwoje mają być synchronizowane w czasie przemieszczania? Czy konwoje mogą być niszczone (atakowane) w czasie przemieszczania? Czy struktura oraz parametry sieci drogowej mogą zmieniać się w czasie? Czy konwoje mają być przemieszczane rozłącznymi drogami? Czy konwoje muszą osiągać pewne wyznaczone punkty w określonym czasie? Czy konwoje muszą startować w tym samym czasie? Czy konwoje muszą przemieszczać się tylko w pewnych pasach terenu? Czy konwoje mogą być rozłączane i złączane w czasie marszu? Czy trasy dla konwojów muszą przechodzić przez pewne ustalone punkty?, itd.

4. SYMULACJA PRZEMIESZCZANIA I PROCES STEROWANIA MARSZEM

Proces bezpośredniego sterowania marszem zawiera takie fazy, jak: dowodzenie, meldowanie i reagowanie na sytuacje awaryjne w czasie marszu [18]. Automat do marszu na szczeblu batalionu reaguje na pewne sytuacje awaryjne zaprezentowane w Tabeli 1 (procedura *React_To_Fault_Situations*(*id'*), Tabela 2). Sytuacje, które wymagają meldowania do przełożonego dowódcy batalionu są następujące: osiągnięcie punktów pośrednich, rejonu postoju lub odpoczynku; zmniejszenie prędkości powodujące opóźnienie; wykrycie skażenia; wykrycie pola minowego; osiągnięcie stanu zapasów paliwa na poziomie 75% i 50% stanu normatywnego; utracenie zdolności do prowadzenia marszu (meldowanie przyczyny utraty zdolności); wykrycie jednostek przeciwnika.

Podczas symulacji przemieszczania, jednostki na drodze "widziane" są dwojako: (a) jako zajmujące pewne odcinki drogi i skrzyżowań (elementy sieci Z_2), (b) jako zajmujące ciąg kwadratów sieci Z_1, przez które odcinek drogi przebiega. W przypadku (a) przemieszczane jest czoło i ogon kolumny i rejestrowane są łuki sieci Z_2, na których znajdują się czoło i ogon kolumny oraz stopień pokonania każdego z tych łuków. W przypadku (b) wyznaczane jest położenie czoła i ogona kolumny na kwadratach sieci Z_2 i przemieszczany jest taki "ciąg" kwadratów (od czoła do ogona). Przemieszczanie jednostki (rozwiniętej w kolumnę) po drodze realizowane jest poprzez wyznaczenie ciągu wierzchołków (skrzyżowań) i/lub kwadratów oraz łuków (odcinków dróg) sieci Z_2 używając algorytmów opisanych w poprzednim rozdziale i następnie realizację (symulację) przemieszczania od jednego wierzchołka do drugiego wierzchołka (procedura *Simulate_Unit_Movement(id')*, Tabela 2).

Ważnym problemem podczas symulacji jest ustalenie aktualnej prędkości przemieszczania się jednostki *id* po odcinku drogi (procedura *Adapt_March_Velocity*(*id'*), Tabela 2). Procedura ustalania prędkości wewnątrz *j*-tego kwadratu (na *j*-tym odcinku) rozpatruje dwa przypadki: (a) kiedy jednostka *id* nie prowadzi walki w *j*-tym kwadracie; (b) kiedy jednostka *id* prowadzi walkę w *j*-tym kwadracie.

Tabela 1. Wybrane sytuacje awaryjne w czasie symulacji marszu i reakcje automatu

Sytuacje awaryjne podczas symulacji marszu	**Reakcje automatu**
Aktualna prędkość podległej jednostki różni się od prędkości planowanej	– Jeżeli jednostka jest czołem kolumny i nie porusza się z zaplanowaną prędkością, to zwiększ prędkość (w przypadku opóźnienia) lub zmniejsz prędkość (w przypadku przyspieszenia); – Jeżeli jednostka nie jest czołem kolumny, to przystosuj prędkość jednostki do prędkości jednostki poprzedzającej ją w kolumnie.
Osiągnięcie jednego z krytycznych poziomów paliwa w jednostce podległej	Meldowanie do automatycznego dowódcy. Próba tankowania na najbliższym postoju lub tankowanie niezwłoczne, jeśli to możliwe
Wykrycie jednostki przeciwnika	Jeżeli potencjał jednostki przeciwnika jest odpowiednio duży (powyżej pewnej wartości granicznej) i odległość między jednostką własną, a jednostką przeciwnika jest mała, to zatrzymanie jednostki, przejście do obrony i meldowanie do dowódcy. W przeciwnym przypadku, tylko meldowanie do dowódcy
Wykrycie pola minowego	Zatrzymanie i meldowanie do dowódcy
Brak możliwości prowadzenia marszu (zniszczenie części trasy marszu (np. mostu) lub inna przyczyna braku przejezdności)	– Jeżeli odcinek drogi jest nieprzejezdny z powodu zniszczenia – próba znalezienia objazdu. Meldowanie do dowódcy; – Jeżeli inny powód nieprzejezdności – przejście do obrony i meldowanie do dowódcy

Źródło: opracowanie własne.

Tabela 2. Procedury automatu decyzyjnego zaimplementowane na potrzeby procesu planowania marszu i sterowania jego przebiegiem

Procedury zaimplementowane i używane dla każdej jednostki *id'*∈ *id* w procesie planowania decyzji	Procedury zaimplementowane i używane dla każdej jednostki *id'*∈ *id* w procesie bezpośredniego sterowania marszem
Units_Order_In_March_Column_Determ(id') *Column_Length_Determ(id')* *Number_of_Stops_Determ(id')* *Place_Of_Stops_Determ(id')* *Ending_Point_PD_Determ(id')* *March_Schedule_Determ(id'):* *Paths_Determ(id')* *Path_ S_To_PS_Determ(id')* *Common_Path_PS_To_PD(id')* *Path_ PD_To_D_Determ(id')* *Detailed_Schedule_Determ(id')*	*March_Simulation(id'):* *Simulate_Unit_Movement(id')* *React_To_Fault_Situations(id')* *Fuel_Consumption_Determ(id')* *Adapt_March_Velocity(id')* *Report_To_Commander(id')*

Źródło: opracowanie własne.

W przypadku (a) aktualna prędkość $v_{cur}(id, j)$ jednostki *id* w *j*-tym kwadracie obliczana jest następująco:

$$v_{cur}(id,j)=\min\{v^{slowd}(id,j), v_{dec}(id,j)\} \tag{17}$$

gdzie: $v^{slowd}(id,j)$ - maksymalna prędkość jednostki *id* w *j*-tym kwadracie z uwzględnieniem warunków topograficznych, $v^{slowd}(id,j)=v^{\max}(id)\cdot FOP(id,\bullet)$, $v^{\max}(id)$ - maksymalna prędkość jednostki *id* wynikająca z parametrów technicznych pojazdów należących do jednostki, $v^{\max}(id)=\min_{p\in ZVeh(id)} v^{tech}(p)$, *ZVeh(id)* – zbiór pojazdów należących do jednostki *id*, $v^{tech}(p)$ – maksymalna prędkość pojazdu *p* (wynikająca z parametrów technicznych), $FOP(id,\bullet)\equiv f_{j,j}^{slowd}$ - funkcja osłabienia prędkości dla jednostki *id* w *j*-tym kwadracie (na łuku (*j*,*j*)); $v_{dec}(id,j)$ – prędkość wynikająca z decyzji dowódcy (równa $v(id,j)$ we wzorze (7)).

Jeśli jednostka *id* jest czołem kolumny i nie przemieszcza się z zaplanowaną prędkością $v_{dec}(id,j)$ wówczas prędkość jest zwiększana (w przypadku opóźnienia) lub zmniejszana (w przypadku przyspieszenia). Jeżeli jednostka *id* nie jest czołem kolumny wówczas prędkość jednostki *id* jest dostosowywana do prędkości jednostki poprzedzającej w kolumnie.

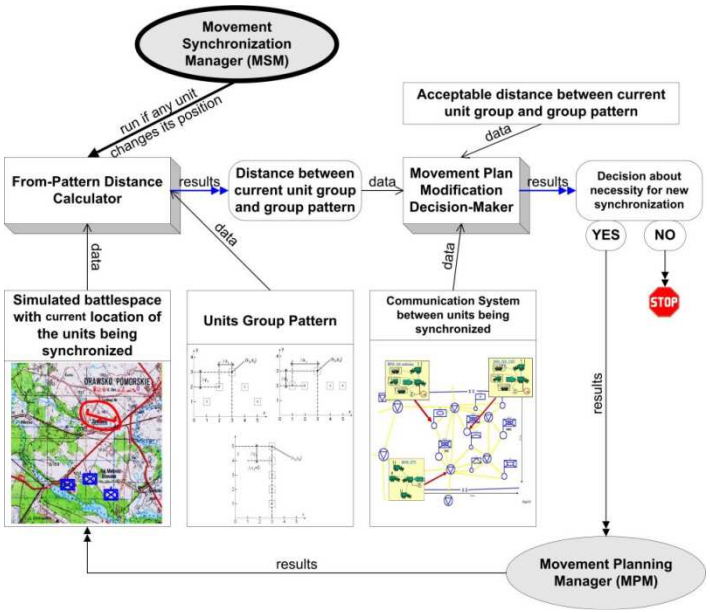
W przypadku (b) aktualna prędkość $v_{cur}(id, j)$ jednostki *id* w *j*-tym kwadracie wyliczana jest jak poniżej:

$$v_{cur}(id,j)=\min\left\{f\left(v^{slowd}(id,\bullet),U_A,U_B,dist\right), v_{dec}(id,j)\right\} \tag{18}$$

gdzie: $f(\bullet,\bullet,\bullet,\bullet)$ – funkcja opisująca prędkość w kwadracie w zależności od $v^{slowd}(id,\bullet)$, potencjałów jednostki *id* strony A (U_A) oraz B (U_B) z którymi walczą, odległości (*dist*) między walczącymi stronami.

Procedury zaimplementowane i używane na etapie planowania decyzji i bezpośredniego sterowania marszem zawiera Tabela 2.

W celu planowania i sterowania przemieszczaniem *K* jednostek podczas symulacji marszu zaproponowany został *Movement Synchronization Manager* (*MSM*) (Rys.2).



Rys. 2. (a) Idea *Movement Synchronization Managera*; (b) Diagramy klas dla pakietu jądra symulacji. *Źródło: opracowanie własne*.

Pierwszy krok (przed symulacją) polega na uruchomieniu *Movement Planning Manager* (*MPM*), który planuje przemieszczanie *K* obiektów, rozwiązując jeden z problemów optymalizacyjnych zdefiniowanych w rozdziale 3 (w zależności od preferencji użytkownika). Moduł *MSM* rozpoczyna działanie wówczas, gdy startuje symulacja przemieszczania. Przechowywana jest w nim m.in. informacja o *wzorcu ugrupowania* (*GP*) *K* monitorowanych jednostek, typie "*miary odległości*" (*TDM*) między aktualnym ugrupowaniem, a wzorcem ugrupowania oraz *dopuszczalna wartość "miary odległości"* (*AVD*). Kiedy rozpoczyna się symulacja przemieszczania, *MSM* informowany jest o każdej zmianie położenia monitorowanych jednostek i w takiej sytuacji uruchamiana jest procedura *From-Pattern Distance Calculator*. Procedura ta wylicza "odległość" pomiędzy aktualnym ugrupowaniem, a *GP* biorąc pod uwagę *TDM*, *AVD* i aktualne położenie *K* jednostek podlegających monitorowaniu. Następnie uruchamiana jest procedura *Movement Plan Modification Decision-Maker*. Jeżeli wyliczona "odległość" jest większa niż dopuszczalna jej wartość *AVD* i istnieje łączność między dowódcą jednostki, a jednostką monitorowaną (symulujemy dowódcę, który widzi lub wie o odstępstwie od planu marszu i podejmuje decyzję o synchronizacji przemieszczania jednostek mu podległych przekazując ją przez sieć łączności) wtedy uruchamiany jest *Movement Planning Manager* (*MPM*), aby znaleźć nowy harmonogram dla *K* jednostek. Szczegółowo procedura ta opisana jest w [19], rozdz.5.3.

5. PODSUMOWANIE

Modele i metody zaprezentowane w pracy używane są w rzeczywistym systemie symulacyjnego wspomagania szkolenia operacyjnego wojsk [1] oraz/lub mogą być użyte w systemach typu *Computer Generated Forces*. Zaprezentowany automat decyzyjny do marszu został zaimplementowany i przetestowany na wielu scenariuszach (nie tylko do marszu). W typowym scenariuszu liczba kwadratów terenu wynosiła kilkadziesiąt tysięcy, a liczba przemieszczanych jednostek – nawet setki. Zaprezentowane metody wraz z ich implementacją są bardzo obiecujące w kontekście efektywności i zarządzania ćwiczeniami wspomaganymi komputerowo typu *CAX*. Używając, dla przykładu, automatu decyzyjnego na szczeblu batalionu możemy oszczędzić wiele czasu oraz ludzi (brak podgrywki zastępowanej

przez automat), dlatego też nawet złożone ćwiczenia mogą być przeprowadzone efektywnie i niskokosztowo
z uruchomieniem wielu scenariuszy działań. Jeden z aspektów automatyzacji procesów decyzyjnych – planowanie, synchronizacja i symulacja przemieszczania jest istotny nie tylko w systemach typu *CGF*. Systemy symulacyjne wspomagające ćwiczenia sztabowe lub zarządzanie kryzysowe powinny być wyposażone w moduły zarządzania przemieszczaniem wielu obiektów. Jakość tego zarządzania ma wpływ na dokładność, efektywność i inne charakterystyki symulowanego systemu. Bardzo ważnym problemem, który dotyczy automatyzacji procesów decyzyjnych jest kalibracja modeli symulacyjnych złożonych procesów [2]. Umożliwia ona dostrajanie tych modeli. Proces ten ma wpływ na jedną z najważniejszych cech modelu symulacyjnego, jaką jest adekwatność.

BIBLIOGRAFIA

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

[2] Antkiewicz R., Najgebauer A., Rulka J., Tarapata Z.: Calibration of simulation models of selected battlefield processes, Proceedings of the 1st Military Communication and Information Systems Conference, ISBN 83-920120-1-1, 18-19.09, Gdynia (Poland) 2006.

[3] Campbell C., Hull R., Root E., Jackson L.: Route Planning in CCTT, in Proceedings of the 5th Conference on Computer Generated Forces and Behavioural Representation, Technical Report, Institute for Simulation and Training, pp. 233-244, 1995.

[4] Dompke U.: Computer Generated Forces - Background, Definition and Basic Technologies, RTO-EN-017, SAS Lecture Series on "Simulation of and for Military Decision Making", Rome, Italy, 15-16 October, 2001.

[5] Karr C.R., Craft M.A., Cisneros J.E.: Dynamic Obstacle Avoidance, Proceedings of the Conference on Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment, The International Society for Optical Engineering, 19-20 April, Orlando (USA) 1995, pp. 195-219.

[6] Korf R.E.: Artificial Intelligence Search Algorithms, in Algorithms Theory Computation Handbook, Boca Raton, FL: CRC Press (1999).

[7] Kreitzberg T., Barragy T., Nevin B.: Tactical Movement Analyzer: A Battlefield Mobility Tool, Proceedings of the 4th Join Tactical Fusion Symposium, Laurel (USA), 1990.

[8] Lavalle S.: Planning Algorithms, Cambridge University Press, 2006.

[9] Longtin M., Megherbi D.: Concealed Routes in ModSAF, in Proceedings of the 5th Conference on Computer Generated Forces and Behavioural Representation, Technical Report, Institute for Simulation and Training, Orlando (USA), 1995, pp. 305-314.

[10] Najgebauer A.: Informatyczne systemy wspomagania decyzji w sytuacjach konfliktowych. Modele, metody i środowiska symulacji interaktywnej, Suplement do Biuletynu Wojskowej Akademii Technicznej, Warszawa, 1999.

[11] Najgebauer A.: Polish Initiatives in M&S and Training. Simulation Based Operational Training Support System (SBOTSS) Zlocien, Proceedings of the ITEC'2004, London (UK) 2004.

[12] 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, The International Society for Optical Engineering, 19-20 April, Orlando (USA) 1995, pp. 251-280.

[13] Pohl J., Chapman A., Pohl K., Primrose J., Wozniak A.: Decision-Support Systems: Notions, Prototypes, and In-Use Applications With Emphasis on Military Applications, Design Institute Report: CADRU-11-97, Collaborative Agent Design Research Center (CADRC), California Polytechnic State University, San Luis Obispo (CA, USA), 2003.

[14] Reece D., Kraus M., Dumanoir P.: Tactical Movement Planning for Individual Combatants, Conf. on Computer Generated Forces and Behavioral Representation, Orlando (USA) 2000.
[15] Tarapata Z.: Models and Methods of Movement Planning and Simulation in Simulation Aided System for Operational Training, Proceedings of the 6th NATO Regional Conference on Military Communication and Information Systems, ISBN 83-920120-0-3, 06-08 October, Zegrze (Poland) 2004, pp. 152-161.
[16] Tarapata Z.: Selected Multicriteria Shortest Path Problems: an Analysis of Complexity, Models and Adaptation of Standard Algorithms, International Journal of Applied Mathematics and Computer Science, vol.17, No.2 (2007), pp. 269-287.
[17] Tarapata Z.: Approximation Scheduling Algorithms for Solving Multi-objects Movement Synchronization Problem, ICANNGA'2009, Lecture Notes in Computer Science, vol.5495 (2009), Springer, Heidelberg, pp.577-589.
[18] Tarapata Z.: Movement Simulation and Management of Cooperating Objects in CGF Systems: a Case Study, KES-AMSTA'2010, Lecture Notes in Artificial Intelligence, vol.6070 (2010), Springer, Heidelberg, pp.293-304.
[19] Tarapata Z.: Models And Algorithms For Knowledge-Based Decision Support And Simulation In Defence And Transport Applications, Rozprawa habilitacyjna, Wojskowa Akademia Techniczna, Warszawa, 2011 (w druku).

KNOWLEDGE-BASED MODELS AND ALGORITHMS OF MOVEMENT PLANNING AND SIMULATION IN MILITARY DSS AND CGF SYSTEMS

Abstract:

In the paper models and algorithms of movement planning and simulation in knowledge-based military Decision Support Systems and Computer Generated Forces are presented. A case study deals with the idea and model of command and control process applied for the decision automata for march on the battalion level. The automaton executes two main processes: decision planning process and direct march control. Some theoretical and practical aspects of modelling and optimization of planning process, march simulation and control are considered. The automata implementation process and experimentation approach are presented. Some results of march planning and realization (simulation) in distributed interactive simulation system SBOTSS "Zlocien" for CAX'es (Computer Assisted Exercises) are discussed.

Key words: route planning, movement simulation, movement synchronization, decision support systems, transport algorithms