DOI: 10.24427/978-83-66391-28-4_9

Metoda wyznaczania stanu agentów w symulacji wieloagentowej o zmiennej rozdzielczości

Dariusz PIERZCHAŁA [*]
Przemysław CZUBA [*]

1. Wprowadzenie

Świat w swojej naturze jest złożony z ogromnej liczby różnych współistniejących obiektów, które nieustannie ewoluują oraz stale oddziałują na siebie. Ich badanie za pomocą klasycznych metod, o wybiórczym zastosowaniu, często nie jest wystarczające z uwagi na szczegółowość i złożoność obiektów oraz konstruowanych dla nich modeli. W ramach poszukiwania nowych metod uformowała się gałąź inżynierii systemów, zwana systemami wieloagentowymi (ang. *multi-agent systems*, MAS). Jest to wiedza interdyscyplinarna, łącząca w sobie m.in. elementy systemów rozproszonych, sztucznej inteligencji, symulacji, teorii gier, a także nauk społecznych. Idea tychże systemów jest relatywnie prosta – składają się one z wielu agentów, które komunikują się ze sobą w ramach wspólnego środowiska (w tym są: cele, zasady, semantyka i ograniczenia). Autorzy Franklin i Gasser definiują agenta następująco [1]: „Autonomiczny agent jest systemem usytuowanym wewnątrz środowiska, które obserwuje i podejmuje w nim działania w celu osiągnięcia własnych celów". Bezpośrednią konsekwencją takiego stwierdzenia jest sformułowanie założenia o posiadaniu przez agenta zdolności do stawiania sobie zadań oraz dostosowywania swoich akcji w celu ich realizacji. Autonomiczna jednostka wykonuje swoje czynności w środowisku, w którym znajdują się inne agenty. Agent podczas deliberacji kolejnych działań musi brać pod uwagę również akcje innych agentów. Stąd też komunikacja oraz koordynacja działań pomiędzy nimi są niezbędne dla pomyślnej realizacji postawionych agentom zadań. To właśnie koordynacja działań w grupie agentów jest jednym z fundamentalnych problemów systemów wieloagentowych oraz stanowi jeden z aspektów rozważanych w niniejszym artykule.

Adekwatność w modelowaniu agentów i ich relacji sprawia, że symulacja złożonych systemów wieloagentowych może być bardzo wymagająca (co do zasobów i czasu realizacji). Każdy model jest pewną subiektywną abstrakcją wycinka rzeczywistości, a zatem modele mogą różnić się poziomem szczegółowości opisu struktury i dynamiki. Poziom szczegółowości zależy od trzech czynników: zakresu (systemu,

[*] Wojskowa Akademia Techniczna

domeny wejściowej i przetwarzanej informacji wyjściowej), rozdzielczości (dokładności, z którą przedstawiane są elementy systemu i ich zachowania) oraz perspektywy. Modelowanie wielorozdzielcze (ang. *multi-resolution modeling*, MRM) pozwala zróżnicować widzenie obiektu symulacyjnego i dostosować poziom rozdzielczości do bieżących oczekiwań. Jednostka (encja) o niskim poziomie rozdzielczości (wysokim stopniu agregacji) zwana jest LRE (ang. *low-resolution entity*) – reprezentuje często wiele jednostek zagregowanych w jednym obiekcie (np. batalion grupujący kompanie). Analogicznie, wysoki poziom rozdzielczości pociąga za sobą bardziej szczegółowy opis atrybutów oraz dynamiki zmian, zatem odpowiada niskiemu stopniowi agregacji – jednostki są reprezentowane indywidualnie (HRE – ang. *high-resolution entity*). Mimo że modele o wysokiej rozdzielczości odwzorowują systemy precyzyjniej, to modele zagregowane w dalszym ciągu są i będą implementowane w wielu systemach. Ma to związek z ograniczeniami mocy obliczeniowej i pamięci komputerów, a także uogólnionymi oczekiwaniami wobec modeli w określonych zastosowaniach. Złożone systemy adaptacyjne z zachowaniami emergentnymi (ang. *complex adaptive systems and emergent behaviors*) to przykładwystępowania pewnych zjawisk dopiero na poziomie makroskopowym (czyli niskiej rozdzielczości). Nie są one zrozumiałe przy zastosowaniu mikroskopowych praw rządzących pojedynczymi obiektami, ponieważ zachowanie emergentne pojawia się dopiero w analizie grupy prostych jednostek jako jednego agregatu. Jego uogólnione zachowanie jest bardziej złożone niż zachowania składowych obiektów, a w przypadku emergencji związanej z różnicami poziomów powodem może być uporządkowanie i wzmocnienie specyficznych oddziaływań między jednostkami, skutkujące synergicznymi efektami w całej grupie.

Jednostka, którą można obserwować na wielu różnych poziomach rozdzielczości nazywana jest encją wielorozdzielczą (ang. *Multi-resolution Entity,* MRE). W momencie interakcji między jednostkami na różnych poziomach rozdzielczości dochodzi z reguły do problemu spójności stanów – obiekty nie mogą poprawnie współpracować, gdy posiadają różne zestawy atrybutów, a ich dynamikę odwzorowują różne procesy. Typowym rozwiązaniem jest adaptacyjne dostosowanie rozdzielczości w taki sposób, aby interakcja zachodziła pomiędzy obiektami na tym samym poziomie. Na przestrzeni ostatnich 30 lat powstało wiele rozwiązań tego problemu – w opracowaniu opieramy się na metodzie cross-resolution (CRM). Proponujemy wykorzystanie technik wieloagentowych w symulacji o zmiennej rozdzielczości, gdzie w agregacji oraz deagregacji stanu agenta stosowane są algorytmy konsensusu [2] oraz sterowania formacją [3].

W kolejnej części opracowania zamieszczone jest zwięzłe wprowadzenie do sieci wieloagentowych, które są istotne dla prezentowanych prac, a także opis propozycji metody agregacji i deagregacji stanu agenta. Kolejna część zawiera charakterystykę pakietu symulacyjnego, który został wykorzystany do autorskiej implementacji metody. Finalnie opisano studium przypadku i zaprezentowano wyniki eksperymentalne.

2. Podstawowe pojęcia i algorytmy sieci wieloagentowych

2.1. Sieć wieloagentowa

Sieć wieloagentową (ang. *multiagent network*) możemy widzieć jako zbiór dynamicznych jednostek, które współdziałają przez wymianę sygnałów dla skoordynowania własnego zachowania oraz osiągnięcia wspólnego celu [3]. Podstawowym założeniem dla sieci jest fakt, iż jej struktura oraz cechy wpływają na dynamiczne właściwości systemu. Rozproszone sieci wieloagentowe (np. system rozproszonych robotów) wniosły do teorii sieci nowe problemy związane z ich analizą. Agenty w takich sieciach powinny współdziałać w sposób skoordynowany, aby osiągnąć zbiorowy synergiczny cel, posiadając do dyspozycji ograniczone zasoby obliczeniowe oraz możliwości komunikacji i percepcji.

W sieciach wieloagentowych zakłada się istnienie:

- autonomicznych i dynamicznych jednostek (agentów), które posiadają umiejętności podejmowania decyzji oraz wymiany informacji z sąsiadami;
- struktury wymiany informacji, którą modeluje się przy użyciu teorii grafów – wierzchołki reprezentują jednostki z ograniczonymi zasobami, a gałęzie wirtualne encje kodujące przepływ informacji pomiędzy wierzchołkami.

Poniżej przedstawione zostały wybrane pojęcia sieci wieloagentowych:

- $x_i(t)$, $i \in N$ – stan wierzchołka (agenta) *i-tego* w chwili *t*;
- N_i – dla grafu nieskierowanego: zbiór wszystkich gałęzi incydentnych (sąsiadujących) z wierzchołkiem (agentem) *i-tym*, natomiast dla grafu skierowanego zbiór jego poprzedników, czyli łuków wchodzących do wierzchołka *i-tego*;
- $I_i(t) = \{x_j(t) | \, j \in N_i$ – opis wiedzy agenta *i-tego* o sąsiadach w chwili *t*; jest to zbiór stanów agentów będących sąsiadami (w przypadku grafu nieskierowanego) bądź poprzednikami (graf skierowany);
- $x_i(t+1) = F_i(x_i(t), I_i(t))$ – reguła wyznaczenia stanu agenta *i-tego* w kolejnej chwili, oparta na jego aktualnym stanie oraz wiedzy o sąsiadach.

Sieci wieloagentowe wykorzystywane są w szczególności do rozwiązywania problemów związanych z osiąganiem konsensusu oraz ustalaniem formacji agentów.

2.2. Algorytm konsensusu

W systemach wieloagentowych algorytm konsensusu wpisuje się w klasę problemów związanych ze sterowaniem kooperacyjnym (ang. *cooperative control for multiagent systems*). W przypadku sieci wieloagentowych konsensus oznacza osiągnięcie zgody (ang. *agreement*) co do szczególnej wartości, która jest zależna od stanu wszystkich

agentów w sieci. Ogólna zasada działania algorytmu konsensusu (zwanego także protokołem konsensusu) polega na określeniu i zastosowaniu reguł interakcji, opisujących sposób wymiany informacji pomiędzy agentem i jego sąsiadami w sieci, w wyniku których uzgodniona zostanie wspólna wartość. Podczas procesu uzgadniania stany poszczególnych agentów ewoluują w czasie. Mówi się, że system osiągnął konsensus, gdy wartości dla uzgadnianego stanu *u* każdego z agentów w sieci są równe. Reguły w procesie uzgadniania wykorzystują funkcje definiowane adekwatnie do specyfiki problemu (np. wartość minimalna, maksymalna, średnia itp.)

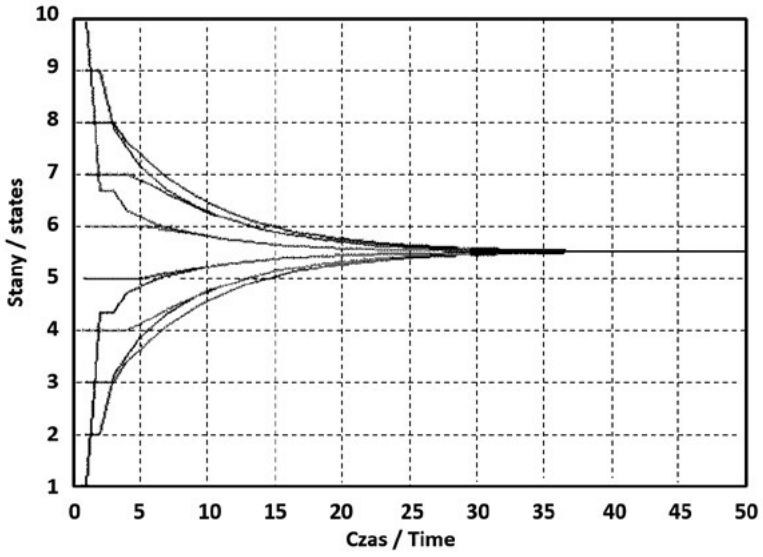
Proponowana metoda agregacji stanów agentów jest oparta na poniższym protokole konsensusu [4], z założeniem dyskretnego czasu oraz topologii stałej w czasie:

$$x_i(t+1) = \frac{1}{N_i + 1}\left(x_i(t) + \sum_{j \in N_i} x_j(t)\right)$$

Z protokołu wynika, że stan agenta (wartość uzgadniana) w każdej następnej chwili jest średnią arytmetyczną z jego stanu oraz stanów sąsiadów. Zaznaczymy, że w przypadku dynamicznych topologii zbiór sąsiadów agenta jest zmienny w czasie [3].

Poniższa granica przedstawia spodziewany wynik algorytmu. Powinien on być równy średniej arytmetycznej początkowych stanów agentów:

$$\lim_{t \to \infty} x_i(t) = \frac{1}{N} \sum_{j \in N} x_j(0), i \in N$$



RYS. 1. Ewolucja stanu agentów podczas działania algorytmu konsensusu
FIG. 1. Agents' states evolution according to consensus algorithm

ŹRÓDŁO: [3].
SOURCE: [3].

Na rysunku 1 przedstawiono przykład ewolucji stanów agentów podczas uzgadniania wspólnej wartości dla stanu encji o niższym poziomie rozdzielczości.

2.3. Sterowanie formacją

Formację można zdefiniować jako wzorzec geometryczny realizowany przez grupę składającą się z wielu agentów. Idea formacji wywodzi się z badań inspirowanych biologią – np. ptaki, używając ustalonej formacji, minimalizują utratę sił podczas lotu. Sterowanie formacją (ang. *formation control*) jest jednym z istotniejszych problemów poruszanych w zagadnieniach sterowania oraz koordynacji agentów w systemach wieloagentowych lub wielorobotowych.

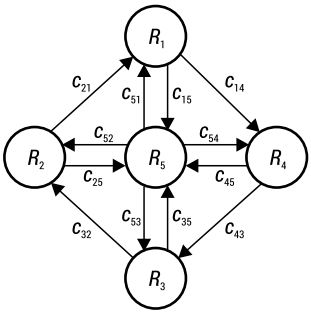
Ideą rozważanej metody jest poruszanie się agentów w ich środowisku tak, aby pozostał utrzymany kształt (wzorzec) formacji. Jednostki w formacji nie poszukują konsensusu co do swoich stanów w całości, lecz co do ich relatywnej pozycji względem reszty agentów. Wzorując się na regułach koordynacji obserwowanych w naturze, wszystkie agenty programowe muszą przestrzegać lokalnych zasad opartych na częściowej wiedzy o pozycji określonych członków grupy. Przykładem w systemie wielorobotowym może być zadanie eksploracji, gdy roboty poruszają się w konkretnej formacji tak, aby maksymalizować przeszukiwany obszar.

W sterowaniu formacją wyróżnia się zasadniczo dwa podejścia:

- pierwsze, gdzie agenty działają według prostych reakcyjnych reguł zachowania, utrzymując dystans od sąsiadów bez zachowania konkretnej pozycji w formacji oraz jej kształtu – schemat takiego działania jest charakterystyczny dla metod inteligencji roju (ang. *swarm intelligence*), a przykładem realizacji może być model boidów Reynoldsa [5];
- drugie oparte jest na określonej odgórnie topologii sieci komunikacji pomiędzy agentami, nazywanej grafem formacji (ang. *formation graph, FG*); wierzchołek w grafie reprezentuje pozycję agenta, a gałęzie – możliwe kanały wymiany informacji pomiędzy parami jednostek (jedno lub dwukierunkowe); rysunek 2 przedstawia graficzny przykład takiego grafu formacji.

Poprawnie zdefiniowany graf formacji musi być spójny, tzn. bez wyizolowanych wierzchołków. Dla każdej gałęzi w grafie formacji określony jest wektor pożądanej relatywnej pozycji pomiędzy parą wierzchołków (pozycjami agentów).

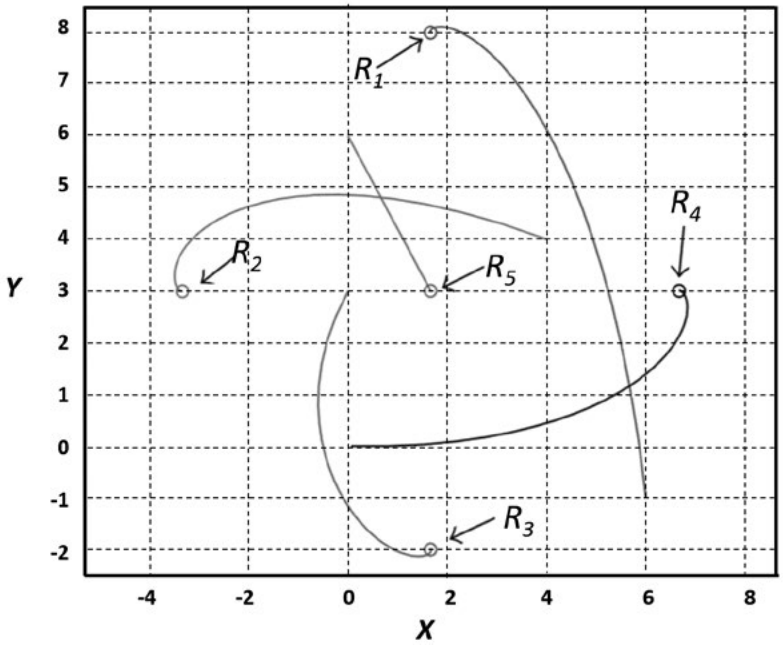
Większość podejść do sterowania formacją zakłada ciągły upływ czasu. Przykładami realizacji są: konfiguracja konwojów w pracy Belkhouche'a [6] czy formacja cyklicznego pościgu Francisa, Broucke'a i Lina [7]. Z drugiej strony, sterowanie formacją w oparciu o czas dyskretny jest stosowane w algorytmie konsensusu, co konceptualnie różni się od podejścia z czasem ciągłym (dodane wektory relatywnych pozycji pomiędzy agentami).



RYS. 2. Przykład grafu formacji
FIG. 2. Example of formation graph

ŹRÓDŁO: opracowanie własne.
SOURCE: own elaboration.

W niniejszym opracowaniu do realizacji metody deagregacji stanu agentów użyto algorytmu sterowania formacją, który jest dyskretny w czasie oraz zakłada stałą topologię [8]. Jego realizacja opiera się na grafie formacji. Stan początkowy agenta jest równy wartości stanu jednostki (encji) o niższej rozdzielczości.



RYS. 3. Przykład grafu formacji
FIG. 3. Example of formation graph

ŹRÓDŁO: [8].
SOURCE: [8].

Niech N_i będzie zbiorem poprzedników agenta R_i, czyli zbiorem agentów, które wykrywają jego pozycję. Niech c_{ij} dla wszystkich $j \in N_i$ oznacza niezależny od czasu wektor pożądanych relatywnych pozycji R_i w zależności od pozycji R_j. Oznacza on odległości, jakie powinny być utrzymywane w konkretnej formacji pomiędzy parami agentów. Stąd pożądaną relatywną pozycję dla każdego R_i w formacji opisujemy przez:

$$z_i^*(t) = \frac{1}{N_i} \sum_{j \in N_i} \left(z_j + c_{ji} \right), i \in N$$

Zatem pożądane relatywne pozycje agentów możemy interpretować jako kombinację pożądanych pozycji z_i względem pozycji wszystkich elementów N_i (poprzedników danego agenta).

Strategia kontroli formacji dla deagregacji stanu agenta posiada następującą postać:

$$u_i(t) = -k\left(z_i(t) - z_i^*(t) \right), i \in N$$

gdzie $k > 1$ jest parametrem przyrostu. Na rysunku 3 przedstawiono wyniki symulacji dla grafu formacji z rysunku 2.

3. Symulacja dyskretna w pakiecie DisSim

W prowadzonych pracach przyjęto, że symulacja komputerowa to ilościowa i jakościowa metoda modelowania w języku formalnym oraz odwzorowania w programie komputerowym strukturalnych i behawioralnych cech systemów (rzeczywistych lub projektowanych), umożliwiająca eksperymentowanie z modelem (zamiast z systemem) i obserwowanie w nim procesów zachodzących w symulacyjnym czasie. W praktyce modele symulacyjne implementowane są z wykorzystaniem bibliotek programowych o różnych możliwościach (np. w zakresie zarządzania czasem i zdarzeniami symulacyjnymi), a zatem o różnych wymaganiach sprzętowych. Dobór właściwego języka programowania i jego symulacyjnych rozszerzeń może być pokierowany bardzo zróżnicowanymi kryteriami – od znajomości języka, przez koszty czasowo-finansowe, aż po możliwość wykonania symulacji w wybranym języku. Oprócz „ciężkich" pakietów symulacyjnych można znaleźć tzw. lekkie rozwiązania, które budowane są od podstaw i tym samym krojone na potrzeby konkretnego zastosowania. W przypadku badań, będących tematem niniejszej pracy, istotne jest, aby czas w symulacji upływał w sposób dyskretny lub quasi-ciągły – ma to bezpośredni związek z algorytmami konsensusu i sterowania formacją.

Autorska propozycją, wychodzącą naprzeciw takim potrzebom, jest pakiet do symulacji dyskretnej DisSim. Agenty wchodzące w skład systemu, wraz z ich cechami oraz relacjami łączącymi je w związki, są odwzorowane w zbiorze obiektów O [9, 10]:

- $O = \{o = < id, c >\}, c \in C°, id \in \mathrm{N}$ – zbiór symulowanych obiektów klasy c identyfikowanych przez id o wartości niepowtarzalnej w zbiorze obiektów;
- $C°$ – niepusty zbiór klas modelowanych obiektów.

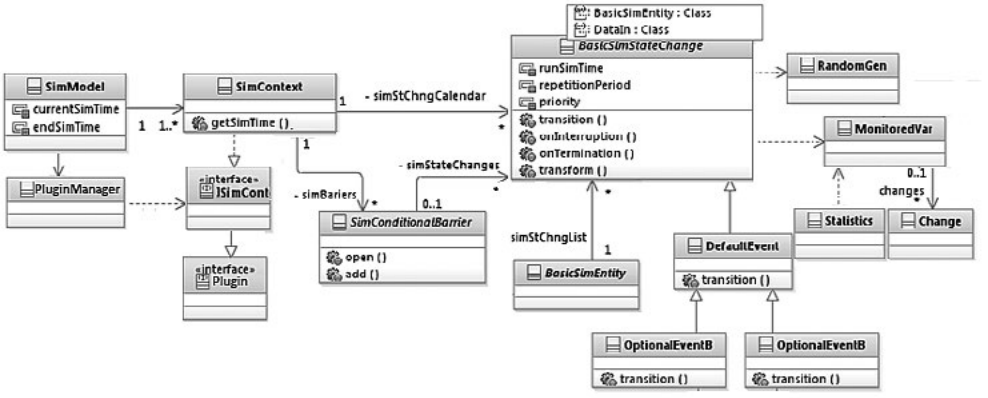
Do każdej istotnej dla celu modelowania, mierzalnej cechy agenta należy przypisać atrybut opisany dziedziną dopuszczalnych wartości:

- A_c – niepusty zbiór atrybutów określonych dla klasy obiektów $c \in C°$;
- V_a^c – zbiór dopuszczalnych wartości atrybutu $a \in A_c$ klasy obiektów $c \in C°$.

W praktyce zbiory $C°$ oraz A_c przedstawia się jako zbiory numerów odpowiednio modelowanych klas obiektów i ich atrybutów.

W symulacji dynamicznej niezbędne jest zdefiniowanie metod i algorytmów programowych wyznaczania kolejnych stanów agenta. W każdym punkcie t czasu symulacyjnego każda składowa stanu systemu, a tym samym wyróżnione cechy agentów, widziana będzie jako czwórka uporządkowana: $s_{o,a} = <o, a, v, t>$. Zatem stan modelowanego systemu $S(t) = \{<o, a, v, t>\}$, $o \in O$ będzie zbiorem utworzonym przez atrybuty wszystkich obiektów (agentów) istniejących w chwili symulacyjnej t. W symulacji dynamicznej wartości $v \in V_a^c$ atrybutów $a \in A_c$ wyznaczane są przez funkcje zmiany stanu, tworząc tzw. zdarzenia.

W *podejściu zorientowanym na zdarzenia* pod pojęciem zdarzenia e ze skończonego zbioru zdarzeń E rozumiana jest planowa zmiana stanu obiektu w określonym punkcie czasu symulacyjnego: $e = <t, f_e^S(t)>$, $t \in \mathrm{T}$. Funkcja zmiany stanu f_e^S: $SxT \rightarrow SxT$ wyznacza stan, w jakim znajdzie się system w chwili t po zajściu zdarzenia e. W metodach symulacji dyskretnej krokowej oraz zdarzeniowej przyjmuje się następujące uproszczenie modelowe – stan systemu nie ulega zmianie do czasu realizacji kolejnego zdarzenia, czyli w przedziale $[t_i, t_{i+1}) \subset T$. Jeżeli zdarzenia występują w odstępach czasu pomijalnych z punktu widzenia uproszczeń modelowych, możemy traktować upływ czasu i symulację jako quasi-ciągłą.



RYS. 4. Podstawowe klasy pakietu DisSim
FIG. 4. Basic classes of DisSim package

ŹRÓDŁO: opracowanie własne.
SOURCE: own elaboration.

Pakiet DisSim zrealizowano w języku Java. Stanowi bazową warstwę programową symulatora realizującego w praktyce proponowane modele i metody. Podstawowe znaczenie mają klasy zdarzenia (*BasicSimStateChange*) oraz obiektu symulacyjnego (*BasicSimEntity*). Inne klasy to m.in. klasy odpowiadające za generowanie liczb pseudolosowych (*RandomGen*), monitorowanie zmiennych (*MonitoredVar*), oszacowania statystyczne (*Statistics*) czy przesyłanie komunikatów (*EventBroker*). Z ich pomocą opracowano drugą warstwę programową symulatora, odpowiedzialną za odwzorowanie agentów oraz wielorozdzielczości. Do implementacji jednostki (encji) wielorozdzielczej wykorzystano klasę obiektu symulacyjnego *BasicSimEntity* – oparta na niej klasa *ResolutionLevel* jest abstrakcją reprezentującą obiekt na określonym poziomie rozdzielczości. Dwa interfejsy – *IAggregation* i *IDeaggregation* – są zapowiedzią metod przejść pomiędzy poziomami szczegółowości, czyli agregacji i deagregacji Dzięki zastosowaniu „wstrzykiwania zależności" implementacje metod agregacji i deagregacji mogą być definiowane na zewnątrz klasy *ResolutionLevel*, a zatem mogą być modyfikowane w trakcie symulacji. Grupując obiekty *ResolutionLevel* w ramach jednej klasy, otrzymujemy jednostkę Multiple-Resolution Entity (MRE) o wielu poziomach rozdzielczości. Zmiana poziomu rozdzielczości jest publikowana jako zdarzenia klas: *AggregationEvent*, *DeaggregationEvent*.

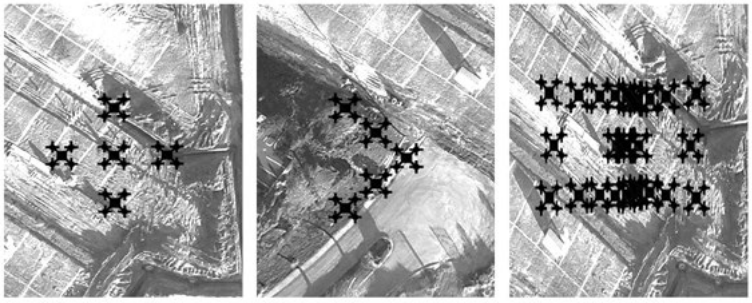
Klasa *BasicSimEntity* jest bazową również dla *BasicAgent* – klasy agenta programowego. Jedną z cech agenta jest możliwość reagowania na zmiany występujące w otaczającym go środowisku. Zatem, każdy taki obiekt implementuje interfejs IEventSubscriber, dzięki czemu otrzymuje dostęp do funkcji pakietu DisSim dotyczących subskrypcji zdarzeń. Ponadto, wykorzystany jest model filtrowania z pakietu *DisSim*, ograniczający przesyłanie komunikatów między publikującymi a subskrybentami. Klasa *BasicAgent* zawiera atrybuty identyfikujące agenta (*id*) i pozycjonujące agenta w przestrzeni (*position*). Model komunikacji (interakcji) odwzorowany jest w klasach pakietu *Network*, opartego na bibliotece *JGraphT*. Główna klasa *Network* reprezentuje sieć wieloagentową, złożoną z obiektu klasy *DirectedWeightedMultigraph* (z biblioteki *JGraphT*), wierzchołków sieci w postaci obiektów klasy dziedziczącej z *BasicAgent* oraz połączeń pomiędzy parami agentów jako klasy *Link*. W obiekcie klasy *Link* atrybuty odziedziczone z klasy *DefaultWeightedEdge* (biblioteki *JGraphT*) opisują wierzchołki (agentów) źródłowych i docelowych połączenia, wagę wierzchołka oraz wektor *Point2D*. Docelowo klasa *Network* zastosowana została do implementacji klasy *FormationGraph*, która reprezentuje możliwe kierunki interakcji pomiędzy agentami oraz wektory pożądanych relatywnych pozycji *PositonVector*. Wektory pozycji są definiowane przez agentów (sourceAgentId, targetAgentId) będących w połączeniu, a także wektor pożądanej relatywnej pozycji dla połączenia.

Przyjęta koncepcja symulacji oraz jej realizacja w opisanych warstwach pakietów i klas programowych języka Java pozwalają na definiowanie i implementację modeli symulacyjnych z rozdzielczością adekwatną do modelowanego problemu oraz z upływem czasu dyskretnym lub quasi-ciągłym.

4. Eksperyment symulacyjny z grupą statków powietrznych BSP

Model symulacyjny wykorzystany w eksperymentach odwzorowuje grupę (drużynę) bezzałogowych statków powietrznych (BSP) z jej strukturą i dynamiką. Zaimplementowany został z wykorzystaniem opisanego w poprzednim punkcie pakietu DisSim, rozszerzonego o warstwę grafiki z biblioteką JavaFX. Agentem programowym jest statek BSP, a drużyna to zagregowany obiekt w ramach MRE.

Przyjmiemy dla potrzeb symulacji następujące założenia dotyczące programowej realizacji modelu i przebiegu eksperymentu. W chwili startu symulacji drużyny dronów są prezentowane na poziomie zagregowanym i rozpoczynają od zadania patrolowania zadanego obszaru. W momencie, gdy drużyna dronów otrzyma sygnał, że na obszarze pojawił się obiekt wymagający ich działania (dla drużyny poszukującej – ratunek, dla drużyny atakującej – eliminacja), zagregowane statki BSP lecą do miejsca zdarzenia. Po dotarciu wykonana zostaje deagregacja obiektu MRE w celu wykonania zadania przez poszczególne agenty (pojedyncze BSP). Po zakończeniu działania drużyna agreguje się do obiektu grupowego i powraca do patrolowania. Kolejny rysunek (5) przedstawia różne formacje zaimplementowane w testach.



RYS. 5. Formacje statków BSP na wysokim poziomie rozdzielczości: formacja „poszukiwawcza", formacja „atakująca" i formacja „żółwia" bez punktów stabilizacyjnych

FIG. 5. High-resolution formations of BSP ships: "search", "attacking", "turtle" without stabilization points

ŹRÓDŁO: opracowanie własne.
SOURCE: own elaboration.

Badania testowe zostały przeprowadzone na następującej konfiguracji sprzętowej: Intel Core i5-3230M 2.60GHz, 12GB RAM, system operacyjny Kubuntu 64-bit. Podczas badań zmierzone zostały czasy wykonania metod agregacji i deagregacji drużyny dronów w celu określenia ich zależności od topologii sieci wieloagentowej. Pomiary wykonano przy użyciu dostępnej w języku Java metody *System.currentTimeMillis*().

Warianty eksperymentów różniły się następującymi parametrami:

- trybem symulacji – z aktywną wizualizacją z krokiem czasu symulacyjnego 0.5 sekundy dla wyznaczania kolejnych pozycji agentów oraz w trybie ASAP (najszybszy możliwy czas wykonania symulacji);
- rodzajem formacji – badania zostały przeprowadzone dla czterech różnych typów formacji, różniących się zasadniczo topologią (rys. 4):
  - formacja atakująca składająca się z pięciu wierzchołków,
  - formacja poszukiwawcza składająca się z pięciu wierzchołków,
  - formacja konwojowania składająca się z pięciu wierzchołków,
  - formacja „żółwia" składająca się z piętnastu wierzchołków.

Czas był mierzony od momentu rozpoczęcia procesu agregacji (deagregacji) aż do momentu jego zakończenia. Wyniki pomiarów zostały przedstawione w tabeli 1.

TAB. 1. Wyniki symulacji
TAB. 1. Simulation results

TRYB Z AKTYWNĄ WIZUALIZACJĄ					
Formacja atakująca					
Lp.	1.	2.	…	9.	10.
Czas agregacji [ms]	503	500		511	500
Czas deagregacji [ms]	6837	7101		7108	7106
Formacja poszukiwawcza					
Lp.	1.	2.	…	9.	10.
Czas agregacji [ms]	1005	1000		699	700
Czas deagregacji [ms]	4825	4800		4800	4799
Formacja konwojowania					
Lp.	1.	2.	…	9.	10.
Czas agregacji [ms]	506	502		504	502
Czas deagregacji [ms]	24739	28415		28414	29212

TRYB Z AKTYWNĄ WIZUALIZACJĄ					
Formacja „żółwia”					
Lp.	1.	2.	…	9.	10.
Czas agregacji [ms]	1528	1601		1599	1603
Czas deagregacji [ms]	32539	35524		35703	43869
TRYB ASAP					
Formacja atakująca					
Lp.	1.	2.	…	9.	10.
Czas agregacji [ms]	4	2		11	14
Czas deagregacji [ms]	92	67		69	152
Formacja poszukiwawcza					
Lp.	1.	2.	…	9.	10.
Czas agregacji [ms]	9	9		23	22
Czas deagregacji [ms]	43	79		104	64
Formacja konwojowania					
Lp.	1.	2.	…	9.	10.
Czas agregacji [ms]	11	7		8	7
Czas deagregacji [ms]	272	280		302	241
Formacja „żółwia”					
Lp.	1.	2.	…	9.	10.
Czas agregacji [ms]	55	22		45	34
Czas deagregacji [ms]	641	645		565	581

ŹRÓDŁO: opracowanie własne.
SOURCE: own elaboration.

Wykonany eksperyment pokazał, iż w obu trybach (z aktywną wizualizacją oraz ASAP) dla formacji „atakująca" oraz „poszukiwawcza" zarejestrowano różne czasy, pomimo tej samej liczby wierzchołków w sieci. O ile czasy dla agregacji (algorytm konsensusu) są wyraźnie zbliżone, to dla metody deagregacji (sterowanie formacją) różnią się nawet o tysiące milisekund w obu trybach. Prowadzi to do wniosku, że topologia sieci ma wpływ na czas wykonania deagregacji agentów. W zależności od połączeń pomiędzy agentami (tu statkami BSP) ustalanie formacji może być mniej lub bardziej złożone czasowo. Czas wykonania rośnie także z rozmiarem sieci. Jest to przewidywalny wniosek i potwierdza zależność, iż zwiększanie liczby wierzchołków wiąże się z większą ilością obliczeń.

5. Podsumowanie

W badaniach zaproponowano wykorzystanie technik wieloagentowych w symulacji o zmiennej rozdzielczości, przy czym w agregacji oraz deagregacji stanu agenta zastosowano algorytmy konsensusu oraz sterowania formacją. Podejście wieloagentowe oraz rozdzielcze w modelowaniu systemów są ze sobą połączone w sposób naturalny.

Przyjęta koncepcja symulacji oraz jej realizacja w postaci biblioteki DisSim z dziedzinowymi rozszerzeniami do implementacji modeli MRE pozwala na odwzorowanie w symulacji obiektów z rozdzielczością adekwatną do modelowanego problemu oraz z upływem czasu dyskretnym lub quasi-ciągłym.

Proponowane multidyscyplinarne podejście wydaje się bardzo obiecujące w symulacji wielorozdzielczej. Zaadaptowane algorytmy konsensu i sterowania formacją ograniczają konieczność definiowania znacznie bardziej złożonych algorytmów na potrzeby agregacji i deagregacji w tej klasie problemów. Następnym krokiem może być wykorzystanie technik uczenia maszynowego, co mogłyby poprawić efekty szukania przez agentów optymalnych formacji w zależności od warunków otoczenia.

Literatura

1. Franklin S, Gasser A. Is it an agent, or just a program?: A taxonomy for autonomous agents. In: Muller J, Wooldridge MJ, Jennings NR. eds *Intelligent Agents III. Agent Theories, Architectures, and Languages*. Budapest: Springer Verlag; 1997; 21-35.
2. Saber RO, Murray RM. Consensus protocols for networks of dynamic agents. *American Control Conference Proceedings*. 2003; 951-956.
3. Mesbahi M, Egerstedt M. *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press; 2010.
4. Zhipu J. *Consensus Problem and Algorithms*. CDS 270-2: Lecture 8-1; 2006.

5. Reynolds CW. Flocks, Herds, Schools. *A Distributed Behavioral Model*. ACM SIGGRAPH; 1987.
6. Belkhouche F, Belkhouche B. Modeling and controlling a robotic convoy using guidance laws strategies. *IEEE Transactions on Systems, Man, and Cybernetics B*. 2005; vol. 35, no. 4: 813-825.
7. Francis B, Broucke M, Lin Z. Local control strategies for groups of mobile autonomous agents. *IEEE Transactions on Automatic Control*. 2004; vol. 49, no. 4: 622-629.
8. Hernandez-Martinez EG, Flores-Godoy JJ, Fernandez-Anaya G. *Decentralized Discrete-Time Formation Control for Multirobot Systems*. Universidad Iberoamericana; 2013.
9. Pierzchała D., *Symulacja komputerowa – od procedury do chmury*, w monografii ISBN/ISSN: 978-83-7938-038-1. Warszawa; 2014; 104-118.
10. Dyk M, Najgebauer A, Pierzchała D. Agent-based M&S of smart sensors for knowledge acquisition inside the Internet of Things and sensor networks. *Int. Inf. and Database Syst*., LNCS, 9012, Subseries: LNAI, XXXVI; 212-223; 2015.

Streszczenie

W opracowaniu zaproponowano wieloagentowe podejście do wyznaczania stanu agenta w symulacji wielorozdzielczej (o zmiennej rozdzielczości) i wieloagentowej. Dwie kluczowe metody zastosowane do realizacji procesu agregacji i deagregacji stanów to algorytm konsensusu i kontroli formacji. Idea koordynacji działań wielu agentów wyłoniła się z obserwacji oraz symulacji zbiorowych zachowań żywych istot. Algorytmy konsensusu są powszechnie stosowane w przypadku problemów sterowania kooperacyjnego w systemach wieloagentowych (konsensus oznacza osiągnięcie zgody na temat szczególnej wartości, która jest zależna od stanu wszystkich agentów w sieci). Kontrola formacji jest natomiast najpopularniejszym algorytmem w problemie koordynacji ruchu w systemach wielorobotowych, gdzie musi być spełniony warunek utrzymania predefiniowanego kształtu geometrycznego formacji.

Przedstawione w pracy podejście pokazuje, że metody wielodyscyplinarne wydają się bardzo obiecujące w symulacji wielorozdzielczej. Algorytmy konsensu i kontroli formacji eliminują konieczność definiowania znacznie bardziej złożonych algorytmów na potrzeby agregacji i deagregacji.

Słowa kluczowe: wielorozdzielcza symulacja, symulacja wieloagentowa, system wieloagentowy, sterowanie formacją grupy

Summary
The method of state estimation in multiagent multiresolution simulation

The paper proposes the multiagent techniques for estimation of agent's state in the multiresolution multiagent simulation. The key methods we have used for state aggregation and disaggregation are: consensus algorithm and formation control. The idea of the coordination of multiple agents has emerged from both observation and simulation of a collective behaviour of biological entities.

The consensus algorithms are commonly used for the cooperative control problems in the multiagent systems, whilst the formation control is the most popular and fundamental motion coordination problem in the multiagent systems, where agents converge to predefined geometric shapes.

The presented approach shows that multiagent methods seem to be very promising in multiresolution simulation. Consensus and formation control algorithms remove necessity to specify the much more complex algorithms for the aggregation and disaggregation needs.

Keywords: multiresolution multiagent simulation, multiagent system, formation control