DARIUSZ KAŚL

# WYBRANE METODY PROPAGACJI DANYCH W SYSTEMACH ROZPROSZONYCH


OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ

**Dariusz Król**

# Wybrane metody propagacji danych w systemach rozproszonych

Oficyna Wydawnicza Politechniki Wrocławskiej
Wrocław 2012

Publikacja dofinansowana przez Ministerstwo Nauki
i Szkolnictwa Wyższego, projekt badawczy N N516 448538

Recenzenci
Juliusz Lech KULIKOWSKI
Tadeusz MORZY

Opracowanie redakcyjne i korekta
Dorota RAWA

Projekt okładki
Marcin ZAWADZKI





OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
http://www.oficyna.pwr.wroc.pl
e-mail: oficwyd@pwr.wroc.pl
zamawianie.ksiazek@pwr.wroc.pl

ISBN 978-83-7493-669-9

Drukarnia Oficyny Wydawniczej Politechniki Wrocławskiej. Zam. nr 267/2012.

# Spis treści

# Ważniejsze oznaczenia i skróty

$\Delta\tau$ – ilość feromonu wygenerowana przez pojedynczy pojazd
$\rho_{wf}$ – współczynnik wzmocnienia śladu feromonowego
$\rho_{pf}$ – współczynnik parowania feromonu
$\tau_i$ – poziom feromonu w chwili $t_i$
$o_b$ – odsetek błędów
$\overline{o_b}$ – średni odsetek błędów
$o_s$ – odchylenie standardowe z próby
$ods_{bez}$ – bezpieczny odstęp między pojazdami
$p_b$ – prawdopodobieństwo wystąpienia błędu
$p_w$ – przyspieszenie wykonania
$t_o$ – minimalny czas oczekiwania między raportami
$t_{ops}$ – czas oczekiwania na przejazd przez skrzyżowanie
$t_{pcw}$ – prognozowany czas obliczeń w węźle
$\overline{t_{pcw}}$ – średni prognozowany czas obliczeń w węźle
$t_{pcz}$ – prognozowany czas obliczeń dla zadania w węźle
$\overline{t_{pcz}}$ – średni prognozowany czas obliczeń dla zadania w węźle
$t_{pow}$ – próg opłacalności węzła
$t_{pps}$ – czas przejazdu przez skrzyżowanie
$t_{pr}$ – czas pierwszego raportowania
$\overline{t_r}$ – średni czas raportowania
$t_w$ – czas wykonania
$t_z$ – czas zdarzenia
$v$ – prędkość pojazdu,

*JRZL* – jednostkowy rozmiar zakresu liczb
*LK* – liczba komunikatów
*LP* – liczba propagacji
*LR* – liczba raportów
*LW* – liczba węzłów
*LWB* – liczba węzłów obciążonych błędem
*LZW* – liczba zadań w węźle
*MW* – moc węzła
*MZLZ* – maksymalna dopuszczalna zmiana liczby zadań w węźle
*RP* – rodzaj propagacji
*SD* – struktura drzewiasta
*SE* – strategia ekstensywna

*SG* – struktura gwiaździsta
*SI* – strategia intensywna
*SPE* – struktura pełna
*SPI* – struktura pierścieniowa
*SPR* – struktura przypadkowa
*TB* – typ błędu
*TST* – typ struktury topologicznej
*ZL* – zakres wyszukiwanych liczb
*ZWL* – zbiór wyszukanych liczb

# 1. Wstęp

Celem publikacji jest zaprezentowanie wybranych metod propagacji danych stosowanych do rozwiązywania przykładowych zadań kolektywnej inteligencji we współczesnych systemach rozproszonych. Metody te umożliwiają wielokrotne i powtarzalne przekazywanie danych między wieloma podmiotami występującymi w środowisku w kontekście realizacji zamierzonego celu. W pracy badano, jak proces propagacji danych może być praktycznie wykorzystany do rozproszonego przetwarzania zadań, projektowania odpornych na błędy rozwiązań sieciowych oraz do optymalizacji natężenia ruchu drogowego w systemach sterowania ruchem pojazdów w mieście.

## 1.1. Problematyka i cel publikacji

W czasach społeczeństwa informacyjnego konieczna stała się budowa dużych systemów sieciowych wspierających rozwiązywanie różnych, często bardzo złożonych, zadań. Praktycznie każdy użytkownik takiego systemu spotyka się z propagacją danych, czyli wielokrotnym i powtarzalnym procesem przekazywania danych między wieloma podmiotami występującymi w danym środowisku w kontekście realizacji zamierzonego celu. Zwykle jest to zadanie skomplikowane wymagające nie tylko umiejętności komunikacyjnych, ale także poznawczych w znaczeniu aktywnej eksploracji danego środowiska.

Obecnie pojęcie propagacji często pojawia się w różnych dziedzinach życia. W fizyce występuje propagacja światła, fal radiowych i dźwiękowych, w programowaniu – propagacja wyjątków i kodu źródłowego, a w systemach sterowania – propagacja uszkodzeń i niepewności pomiarów. W informatyce spotyka się m.in. wsteczną propagację błędu, propagację wpisów w DNS, propagację zmian w bazie danych, propagację schematów i ograniczeń integralnościowych, propagację wirusów i wiadomości, propagację wiedzy. Znanymi pojęciami związanymi z propagacją danych są migracja i replikacja, np. migracja agentów mobilnych czy replikacja baz danych. Definicje poszczególnych pojęć różnią się, łączy je jednak idea wielokrotnego i powtarzalnego procesu przekazywania informacji. Wśród perspektywicznych zastosowań metod propagacji danych jako charakterystyczne wymienić należy następujące obszary szeroko rozumianej informatyki:

systemy i sieci mobilne, złożone systemy decyzyjne, platformy wieloagentowe, systemy inspirowane przyrodą, światy wirtualne oraz systemy społeczne.

Tak szeroki zakres wykorzystania metod propagacyjnych wcale nie oznacza, że problem propagacji danych został dobrze rozpoznany. Nadal istnieje wiele pytań, które pozostają bez jednoznacznej odpowiedzi, np. czy możemy w pełni sterować procesem propagacji, „pozytywnych" jak i „negatywnych", w systemie komputerowym oraz w jakim zakresie współczesny system sieciowy powinien uwzględniać propagację danych.

Następujące zagadnienia stanowiły motywację do rozpoczęcia badań i w konsekwencji poszukiwania rozwiązań, które zostały omówione w pracy:

1. Specyfikacja algorytmu propagacji danych w dużym stopniu zależy od wykorzystywanego schematu komunikacji, dlatego opracowanie uniwersalnego algorytmu propagacji dla wszystkich rodzajów architektur i różnych zastosowań jest praktycznie niemożliwe. Każdy przypadek wymaga osobnego podejścia uwzględniającego jego specyfikę. W pracy zostały szczegółowo opisane problemy związane z propagacją zadań, propagacją błędów i propagacją wiedzy.
2. Praktyczne wykorzystanie metod propagacyjnych w nowoczesnych systemach sieciowych może prowadzić do wzrostu efektywności działania tych systemów lub też do zwiększenia bezpieczeństwa ich działania. Metody opracowane w ramach prezentowanej pracy z jednej strony są przykładem na to, jak propagacja danych może być efektywnie wykorzystywana, np. do rekonfiguracji połączeń – z drugiej zaś, jak mogą służyć do minimalizacji niekorzystnych procesów propagacyjnych. Niestety, bardzo wiele rozwiązań przyjętych w codziennym życiu nie spełnia choćby minimum wymagań, by można było z powodzeniem rozwiązywać problemy związane z propagacją danych, np. zmiana adresu zamieszkania rejestrowana w jednym systemie nie jest automatycznie propagowana do innych systemów, choć byłoby to technicznie możliwe.
3. Proces propagacji danych może być też zastosowany na niekorzyść użytkownika systemu informatycznego, np. do rozpowszechniania błędów, wirusów czy plotek. Zbadanie tych zjawisk, szczególnie na płaszczyźnie systemów społecznych, P2P oraz systemów mobilnych, jest jedną z metod ograniczania przestępczości elektronicznej, która każdego roku powoduje coraz większe straty wśród użytkowników systemów internetowych.

Obszarem zastosowań opracowanych metod propagacji danych jest kolektywna inteligencja rozumiana jako zdolność grupy inteligentnych podmiotów, np. programów i użytkowników, do racjonalnego działania dzięki współpracy lub rywalizacji [113]. Podmioty takie charakteryzują się autonomicznością, możliwą niespójnością wiedzy oraz dążeniem do określonego celu. Podstawowe kwestie dotyczące kolektywnej inteligencji z punktu widzenia procesów propagacyjnych można opisać, poszukując odpowiedzi na następujące pytania:

1. Jak wykorzystać proces propagacyjny we współpracy lub rywalizacji wielu podmiotów występujących w środowisku w kontekście realizacji zamierzonego celu?
2. Jak propagować i integrować dane, aby uzyskać kolektywną inteligencję?
3. Jak optymalizować metody propagacji danych, aby działania grup inteligentnych podmiotów były efektywniejsze?

Odwołanie się do idei kolektywnej inteligencji pozwala wyraźnie wskazać na aspekty praktyczne metod propagacyjnych w przypadku następujących grup podmiotów: agentów, modułów systemów sieciowych, ekspertów czy użytkowników systemów społecznych.

Wkład przedstawianego opracowania do dziedziny kolektywnej inteligencji czy – szerzej – do dziedziny systemów rozproszonych polega na uporządkowaniu i poszerzeniu wiedzy w odniesieniu do wybranych mechanizmów propagacyjnych. Dotyczy to zarówno propagacji danych w systemach ukierunkowanych na efektywną dystrybucję zadań, jak i propagacji błędów w systemach sieciowych. Ważnym elementem proponowanych rozwiązań jest pokazanie, że metody inspirowane przyrodą wykorzystywane w propagacji wiedzy są efektywniejsze od metod tradycyjnych bazujących na wymianie komunikatów.

Wybór problemów propagacyjnych nie jest przypadkowy. Pierwszy z nich dotyczący propagacji zadań wydaje się dobrze znany. Z punktu widzenia mechanizmów propagacyjnych jednak, przy bliższej ich analizie, wiele pytań pozostaje bez odpowiedzi. Podobną sytuację odnaleźć można przy zjawisku propagacji błędów czy propagacji wiedzy. Wymieniona trójka metod propagacyjnych w przypadku zadań, błędów i wiedzy tworzy podstawy do opracowania praktycznych rozwiązań propagacyjnych w obszarze kolektywnej inteligencji.

Celem publikacji jest zaprezentowanie metod propagacji danych stosowanych do rozwiązywania wybranych zadań kolektywnej inteligencji we współczesnych systemach rozproszonych. W pracy badano, jak proces propagacji danych może być praktycznie wykorzystany do rozproszonego przetwarzania zadań, projektowania odpornych na błędy rozwiązań sieciowych oraz do optymalizacji natężenia ruchu drogowego w systemach sterowania ruchem pojazdów w mieście. Skonkretyzowanie podanego kierunku badawczego prowadzi do poszukiwania odpowiedzi na ogólniejsze pytanie: w jakim zakresie współczesne systemy sieciowe powinny uwzględniać propagację danych.

Podstawowy cel prowadzonych badań został osiągnięty przez realizację następujących celów szczegółowych:

1. Przedstawienie propagacji danych jako mechanizmu ułatwiającego takie formy interakcji, jak: kolektywność, kooperatywność lub kolaboratywność.
2. Zaproponowanie architektury sieciowej umożliwiającej efektywną propagację zadań, w tym zdefiniowanie progu opłacalności oraz strategii limitowania procesu propagacji, aby można było ograniczyć negatywne zjawisko cyklu propagacyjnego.

3. Pokazanie, że przebieg i zakres propagacji błędów w systemie rozproszonym zależy przede wszystkim od struktury topologicznej systemu oraz typu i momentu wystąpienia błędu.
4. Wykazanie, że mechanizm propagacji wiedzy bazujący na algorytmie inspirowanym przyrodą może być efektywniejszy od tradycyjnej metody propagacji realizowanej za pomocą wymiany komunikatów.

Metody opracowane w ramach przedstawionej publikacji mają praktyczne znaczenie i mogą być z powodzeniem stosowane ze względu na ich dobrą efektywność, zwłaszcza tam, gdzie konieczna jest wydajna propagacja danych do wybranych komponentów sieci lub też jej skuteczne zablokowanie. Problem tego typu pojawia się na przykład przy dystrybucji decyzji na niższe szczeble zarządzania oraz przy przeciwdziałaniu atakom komputerowym lub rozpowszechnianiu plotek w sieci społecznej. Decentralizacja zasobów sieci ma w tym przypadku pozytywny wpływ na możliwość propagowania danych „pozytywnych", z drugiej strony jednak utrudnia zatrzymanie propagacji „negatywnych".

Książka jest rezultatem doświadczeń, które autor nabył, prowadząc badania naukowe na Politechnice Wrocławskiej. Zaproponowany jeszcze w rozprawie doktorskiej [76] mechanizm migracji obiektów wieloklasowych z bazy danych ewoluował do postaci złożonego procesu propagacji danych w systemie rozproszonym. Większość prezentowanych tutaj rozwiązań zostało już opublikowanych – najważniejsze publikacje to: [68], [71], [77], [79], [80], [84], [86], [87], [88]. Część z nich powstała w wyniku współpracy, prowadzonej głównie w obszarze działań programistycznych, z moimi studentami: Łukaszem Korzeniowskim, Michałem Zelmozerem, Aleksandrem Lupą, Grzegorzem Kuklą, Łukaszem Popielą, Maciejem Drożdżowskim, Maciejem Mrożkiem, Tomaszem Kolasą oraz Piotrem Karasiem – za co składam Im serdeczne podziękowania.

## 1.2. Układ i zawartość kolejnych rozdziałów

W następnych rozdziałach zostaną przedstawione wybrane metody propagacji danych, dla różnych architektur i schematów komunikacyjnych. Znaczenie i oryginalność tych rozwiązań wzrasta zwłaszcza w wyniku szerokiego zakresu ich stosowania. Dlatego w celu sprawdzenia i zilustrowania przyjętych rozwiązań zaprojektowano i zrealizowano pod kierunkiem autora kilka prototypów systemów rozproszonych. Spośród nich zostały wybrane do omówienia – z podaniem istotnych elementów specyfikacji oraz zasad ich działania – te, które są najlepiej ilustrującymi przykładami omawianych mechanizmów. Najczęściej wybieraną platformą implementacyjną w przypadku wyróżnionych prototypów było środowisko wieloagentowe, zdecydowano się więc na używanie języka opisu charakterystycznego dla systemów wieloagentowych.

Praca składa się z sześciu rozdziałów. W rozdziale 2 omówiono motywację, która zadecydowała o podjęciu badań, na tle problematyki mechanizmów interakcyjnych i ogólnego zagadnienia propagacji danych z uwzględnieniem fizycznych modeli przemian fazowych. Dodatkowo opisano dwa przykłady zastosowań tych mechanizmów, a mianowicie – propagację zapytań w systemach społecznych i metodę identyfikacji komponentów krytycznych w systemie rozproszonym. Mechanizm propagacyjny dystrybucji zadań jest tematem rozdz. 3 poświęconego propagacyjnej architekturze sieciowej pozwalającej na rozproszone wyznaczanie liczb pierwszych. Wzięto pod uwagę kolejno trzy rodzaje propagacji: migrację, replikację i promocję. Na koniec rozdziału zaprezentowano wyniki badań efektywności mechanizmu propagacji podczas rozproszonych obliczeń. Szczególnie interesujące okazały się próby znalezienia równowagi między zwiększaniem liczby aktywnych węzłów obliczeniowych a ograniczaniem możliwości wystąpienia cyklu propagacyjnego przez zmniejszanie tej samej liczby aktywnych węzłów. W rozdziale 4 zaproponowano ilościowe i jakościowe metody analizy różnych scenariuszy propagacyjnych dla kilku wybranych struktur topologicznych, typów błędów i rozkładów prawdopodobieństw. Wyniki przeprowadzonych badań symulacyjnych jednoznacznie potwierdzają, że przebieg i zakres propagacji błędów zależy przede wszystkim od topologii systemu sieciowego oraz od typu i momentu wystąpienia błędu. Metoda propagacji pośredniej inspirowana przyrodą została przedstawiona w rozdz. 5. Wcześniej przyjętym głównym nośnikiem informacji między elementami systemu był mechanizm bazujący na komunikacji bezpośredniej. Teraz rolę tę przejęła komunikacja pośrednia wykorzystywana m.in. w algorytmach mrówkowych. W pierwszej części rozdz. 5 omówiono problem przekazywania wiedzy inspirowany przyrodą. W drugiej – przedyskutowano wady i zalety propagacji pośredniej w porównaniu do mechanizmu propagacji bezpośredniej opierającej się na wymianie komunikatów na podstawie analizy natężenia ruchu drogowego w systemie sterowania ruchem pojazdów w mieście. W rozdziale 6 zawarto podsumowanie, w którym zaprezentowano ogólne wnioski dotyczące przeprowadzonych badań oraz sformułowano kierunki dalszych prac obejmujące mechanizmy propagacyjne nieanalizowane w prezentowanej publikacji.

# 2. Przegląd metod i modeli

Analizę problemu propagacyjnego rozpoczyna się na przykład od określenia typu stosowanych mechanizmów interakcyjnych zachodzących między elementami systemu rozproszonego. W związku z tym propagacja danych może być kolektywna, kooperatywna lub kolaboratywna. Następnie konieczny jest wybór właściwego paradygmatu badawczego, często nawet kilku równocześnie występujących. Wybór ten nie jest już jednak tak oczywisty, ponieważ zależy od wymagań funkcjonalnych konkretnych aplikacji. Istniejące opracowania literaturowe nie rozwiązują wyczerpująco wymienionych problemów pojawiających się w kontekście procesów propagacyjnych, konieczne jest zatem prowadzenie dalszych badań. Uzupełnienie omawianych kwestii i opisanych metod stanowią dwa przykłady zastosowań mechanizmów propagacyjnych, a mianowicie – propagacja zapytań w systemach społecznych oraz propagacyjna metoda identyfikacji komponentów krytycznych w systemie sieciowym.

## 2.1. Mechanizmy interakcyjne i paradygmaty badawcze

Przez ostatnie lata znaczenie systemów rozproszonych stale rośnie. Główną ich zaletą jest nie tylko dzielenie zasobów i zwiększanie mocy obliczeniowej, ale także rozszerzanie zakresu działania. Są one udostępniane użytkownikowi w coraz przystępniejszej formie. Poza wskazanymi cechami systemów rozproszonych najczęściej wymienianymi zaletami są: otwartość, transparentność, współbieżność i tolerowanie awarii. Niestety, posiadają one też wady – np. wrażliwość na przepustowość łączy komunikacyjnych oraz podatność na wszelkiego rodzaju ataki.

Wymienione możliwości systemów zależą od mechanizmów interakcyjnych zachodzących między elementami systemu. Mechanizmy te mogą mieć charakter bezpośredni, jak w przypadku wymiany komunikatów, lub pośredni w komunikacji feromonowej (zob. rozdz. 5). Komponenty systemu mogą współpracować ze sobą, tworząc zintegrowane zespoły, albo mogą działać samodzielnie. Najczęściej przytacza się trzy podstawowe formy interakcji: kolektywną, kooperatywną i kolaboratywną [106], [118].

W najprostszej formie interakcji – oddziaływaniu kolektywnym – jednostki nie wiedzą o innych, również działających w systemie rozproszonym, mimo to pośrednio korzystają z ich działania. Przykładem tego typu interakcji są systemy wieloagentowe stosujące mechanizmy inspirowane przyrodą. W prezentowanej pracy ten schemat interakcji uwzględniono w rozdz. 5 podczas propagacji wiedzy metodą pośrednią.

Drugi rodzaj interakcji to kooperatywność, w której jednostki wiedzą o innych i bezpośrednio korzystają z ich działania, np. w celu wykonania wspólnego zadania. Także w tym przypadku dobrym przykładem są systemy wieloagentowe – agenty współpracują między sobą przez wysyłanie komunikatów, aby osiągnąć zamierzony cel. Sposób takiej interakcji zostanie omówiony w rozdz. 3 przy opisie propagacji zadań.

Trzecim rodzajem interakcji jest oddziaływanie kolaboratywne. Różni się ono od poprzednich tym, że jednostki posiadają wiedzę o innych i pomagają sobie wzajemnie w dążeniu do spełnienia indywidualnych celów, choć mogą ich początkowo nie znać. Różnica między oddziaływaniem kooperatywnym a kolaboratywnym nie jest duża i znaczna większość spotykanych działań kolaboratywnych jest równocześnie kooperatywna. Ten schemat zostanie bliżej przedstawiony w rozdz. 4 przy analizie zjawiska propagacji błędów w systemie sieciowym.

Propagacja danych jest mechanizmem, który realizuje wszystkie wymienione rodzaje interakcji, a więc może być kolektywna, kooperatywna lub kolaboratywna. Należy jednak dodać, że dla systemu rozproszonego interakcja może przynosić korzyść, ale może mieć też charakter destrukcyjny albo neutralny. Klasyczny przypadek interakcji „negatywnej" zostanie opisany w rozdz. 3 dotyczącym procesu propagacji błędów. Przypadek „neutralny" nie będzie rozpatrywany, gdyż ma on jedynie charakter hipotetyczny.

Analogicznie do kilku rodzajów interakcji wymienić można kilka paradygmatów badawczych, które m.in. pozwalają na wybór odpowiedniej strategii współpracy między elementami rozproszonego systemu. Łatwo zauważyć, że zachodzi zależność między wyborem typu mechanizmu interakcyjnego i wyborem właściwego paradygmatu. Nie ma pełnej dowolności w tym zakresie. Wybór paradygmatu ma ogromny wpływ na dobór odpowiednich metod badawczych, a przez to istotnie wpływa na używany mechanizm interakcyjny.

Nadrzędną zasadą paradygmatów stosowanych w systemach rozproszonych jest uzyskanie jak najlepszej spójności między komponentami systemu, tak jak np. w standardach CORBA, RMI oraz SOA. Wykorzystanie paradygmatów pozwala na abstrakcyjne traktowanie problemu, co w konsekwencji daje więcej potencjalnych rozwiązań, sprawdzonych już wielokrotnie w podobnych przypadkach. Do najbardziej istotnych paradygmatów dla systemów rozproszonych związanych z problemem propagacji danych zaliczamy: paradygmat inspirowany przyrodą, paradygmat socjologiczny (społeczny) oraz paradygmat semantyczny.

Z grupy paradygmatów inspirowanych przyrodą najbardziej interesujące w przypadku struktur sieciowych okazały się teorie wywodzące się z zachowania roju i życia

mrówek. Wprawdzie sposób komunikacji między jednostkami nie jest skomplikowany i polega na możliwości rozpoznania relewantnej informacji z otaczającego świata, ale za to uzyskiwane rezultaty są bardzo dobre, często lepsze od wyników generowanych standardowymi procedurami.

Socjologia, badająca społeczne reguły, procesy i struktury w powiązaniu z ekonomią i psychologią dały podstawy do tworzenia inteligentnych systemów, w których jednostki współpracują w celu rozwiązania złożonego problemu. Podejście takie jest widoczne zarówno w systemach kooperatywnych, jak i kolaboratywnych stosujących m.in. paradygmat funkcjonalistyczny traktujący społeczeństwo jako system wzajemnie powiązanych ze sobą części o jasno zdefiniowanych funkcjach społecznych. Praktycznym przypadkiem wykorzystania nauk ekonomicznych jest dynamiczna alokacja zadań i zasobów służąca do ustalania optymalnych warunków bilansowania zysków i ewentualnych strat.

W przypadku paradygmatu semantycznego, trzeciego paradygmatu używanego w systemach rozproszonych, podstawowym przedmiotem badań jest rozproszona wiedza, z której można wspólnie korzystać, np. za pomocą narzędzi ontologicznych. Dzięki usystematyzowaniu pojęć jesteśmy w stanie dokonać znacznego zredukowania wymiany informacji w systemie, a przez to uzyskać obniżenie ponoszonych kosztów.

Podstawowy wniosek nasuwający się z analizy przedstawionych paradygmatów jest potwierdzeniem, że wybór odpowiedniego paradygmatu nie jest oczywisty, ponieważ zależy od wymagań konkretnych aplikacji. Coraz częściej w złożonych systemach sieciowych można spotkać kilka przeplatających się paradygmatów. Da się to zauważyć w setkach propozycji hybrydyzacji różnych metod i algorytmów, np. algorytm mrówkowo-genetyczny, genetyczne przeszukiwanie tabu, system genetyczno-rozmyty.

W prezentowanej pracy najwięcej miejsca poświęcono paradygmatowi rozproszonych obiektów, następnie paradygmatowi inspirowanemu przyrodą, a najmniej – społecznemu. Oparcie procesu propagacji danych na wymienionych teoriach i pojęciach pozwala wykorzystać wyniki wielu prac badawczych dotyczących efektywności i sposobów implementowania poszczególnych modeli przetwarzania danych.

## 2.2. Komputerowe metody i modele propagacyjne

Przegląd komputerowych metod propagacji danych należy rozpocząć od wstecznej propagacji błędów. Jest to podstawowy algorytm uczenia z nauczycielem dla wielowarstwowych jednokierunkowych sieci neuronowych zaproponowany przez Paula Werbosa w jego pracy doktorskiej [147]. Polega on na zmianie wag neuronów w są-

siednich warstwach sieci zgodnie z metodą najszybszego spadku, tak aby minimalizować sumy kwadratów błędów uczenia przesyłanych od warstwy wyjściowej do warstwy wejściowej [42].

Obecnie można wyróżnić kilka grup komputerowych metod i modeli propagacyjnych. Najważniejsze z nich to te, które są inspirowane przyrodą. Pozostałe dotyczą propagacji w systemach wieloagentowych, społecznych, kolektywnych i kolaboratywnych.

### 2.2.1. Metody inspirowane przyrodą

Wydajne wykorzystanie ogromnych zasobów sieciowych stało się głównym zadaniem inżynierii ruchu w sieciach teleinformatycznych [51]. Wiele algorytmów wyznaczania tras czerpie inspiracje z biologii – z przykładów zachowania organizmów żywych [17], [37], [39], [149]. W pracy [145] dokonano przeglądu propagacyjnych metaheurystyk wyznaczania tras dla różnych rozwiązań topologicznych. Wyróżniono trzy grupy algorytmów inspirowanych przyrodą: mrówkowe, ewolucyjne oraz rojowe. W grupie algorytmów mrówkowych komunikacja odbywa się za pośrednictwem zmian wprowadzanych w środowisku przez pozostawianie śladu feromonowego. Algorytmy ewolucyjne adaptują się przez stosowanie operatorów krzyżowania, mutacji oraz selekcji. Grupa algorytmów rojowych natomiast pozwala na bezpośrednią komunikację, zgodnie z przyjętą topologią sąsiedztwa.

Jeszcze wyraźniej widać praktyczne działanie mechanizmu propagacyjnego w przypadku systemów komunikacji drogowej. Choć istnieje wiele różnych podejść, ostatnio na pierwszy plan wysunęły się rozwiązania inspirowane przyrodą [138]. I tak w pracy [2] zaproponowano algorytm mrówkowy rekomendujący trasę przejazdu dla każdego kierowcy znajdującego się w granicach miasta. Inni opisali metodę krótkoterminowej predykcji natężenia ruchu drogowego bazującej na komunikacji feromonowej adaptacyjnie reagującej na zwiększający się ruch pojazdów i powstające korki uliczne [5], [92]. Podobne rozwiązania zaproponowano w pracy [109], przy czym pojazdy poruszają się zgodnie z zasadami ruchu drogowego, uwzględniając ograniczenia prędkości i bezpieczną odległość między nimi. W publikacji [126] podano, że wystarczy wziąć pod uwagę charakterystykę lokalnego ruchu pojazdów, by efekty optymalizacji były porównywalne z tymi, które uwzględniają globalne parametry ruchu drogowego.

W literaturze można znaleźć również takie rozwiązania, jak hybrydyzacja algorytmów skutkująca przede wszystkim większą efektywnością obliczeniową. Zastosowali ją Pellegrini i Moretti, proponując sekwencyjne złożenie algorytmu Lin-Kernighan z rozszerzeniem algorytmu mrówkowego o przeszukiwanie lokalne przy wyborze punktu początkowego [120].

### 2.2.2. Dystrybucja danych i obliczeń w systemach wieloagentowych

Problem propagacji danych w systemie wieloagentowym należy do szerokiej klasy zagadnień badawczych zajmujących się zachowaniem indywidualnych agentów [3], [4], [59], [142]. Systemy wieloagentowe sprawdziły się jako platformy dystrybucji obliczeń rozproszonych z wykorzystaniem asynchronicznej wymiany komunikatów. Najczęściej stosowaną w tym celu agentową platformą pośredniczącą był JADE [12], [93]. Interesującym rozwiązaniem w tej grupie jest środowisko A-Team o nazwie JABAT, umożliwiające konstruowanie dedykowanych architektur implementacyjnych dla populacyjnych algorytmów heurystycznych, w tym dla problemu komiwojażera [11].

Kolejnym przykładem platformy kooperujących agentów na zasadach relacji społecznych, indywidualnego uczenia się i pośredniego systemu komunikacyjnego jest MAOS [154]. Ostatnio opublikowane wyniki dotyczące problemu komiwojażera potwierdziły konkurencyjność tego rozwiązania z najlepszym znanym podejściem Lin-Kernighan [7], [95].

Nie można jednak pominąć barier skutecznie przeciwdziałających rozwojowi systemów wieloagentowych. Po pierwsze, ze względu na ogromną dynamikę działania implementowanych algorytmów nie można w pełni kontrolować zachowania agentów (przykłady podano w pracach [35] oraz [65]). Po drugie, nie ma przyjętego standardu projektowania systemów wieloagentowych [79], co prowadzi często do nieporozumień przy pracach implementacyjnych.

### 2.2.3. Propagacje złożone w systemach społecznych

W systemach społecznych występuje jeszcze jedno osobliwe zagadnienie związane z propagacją danych. Do tej pory mowa była o jednostkowej propagacji danych między dwoma komponentami systemu sieciowego. Okazało się jednakże, że w systemach społecznych bardzo częstym przypadkiem jest jednoczesna propagacja informacji między wybranym węzłem a wieloma jego sąsiadami. Taki typ przetwarzania został nazwany propagacją złożoną lub kaskadową [23]. Do jego analizy został wykorzystany tzw. model progowy o określonym współczynniku progu. W omawianym modelu występują węzły aktywne i bierne. Jedynie węzły aktywne mogą uczestniczyć w procesie propagacyjnym. Aby węzeł był w stanie aktywnym (wzbudzonym), odsetek jego aktywnych (wzbudzonych) sąsiadów musi być równy lub większy od zadanego progu. Model progowy przypomina model epidemiologiczny [150], ale różni się od niego tym, że w modelu progowym prawdopodobieństwo pobudzenia węzła poniżej progu jest równe 0 i 1 – dla odsetka aktywnych sąsiadów przy równym progu lub powyżej zadanego. W takim modelu warunkiem wystąpienia propagacji złożonej jest

wzbudzenie co najmniej kilku węzłów. Przeprowadzone badania wskazują jednak, że model progowy może też skutecznie blokować proces propagacyjny ze względu na wymagalność odpowiedniej liczby aktywnych sąsiadów. Nawet jeśli się uda uruchomić proces propagowania, natrafia się dalej na problem koniecznej współpracy wielu węzłów. Złożona propagacja wykorzystująca model progowy nie jest łatwo realizowalna i zależy przede wszystkim od lokalnej struktury połączeń węzła początkowego i jego sąsiadów. Uzyskanie dużego wpływu na działanie tego typu systemu sieciowego jest więc bardzo ograniczone. Wydaje się, że ewentualna rezygnacja z modelu progowego mogłaby poprawić skuteczność procesu propagacyjnego, jednak wymagane jest przeprowadzenie w tym celu dodatkowych eksperymentów.

Podobny kierunek badań został podjęty w przypadku propagacji zawartości stron internetowych [155], propagacji zachowań społecznych [22] i propagacji zaufania [58]. Autorzy dwóch pierwszych prac symulują odpowiadające rzeczywistości scenariusze zachowań, które wymagają różnych mechanizmów propagacyjnych. I tak, w pierwszym przypadku do realizacji propagacji zawartości stron internetowych wybrano czasowo zależny model Markowa, podczas gdy w drugim przypadku badano rozwój epidemii dla różnych struktur topologicznych. Ostatnia publikacja dotyczy formalnego modelu zaufania w systemach rozproszonych, w których zdefiniowano m.in. relację zaufania, operację agregacji oraz mechanizm propagacji zaufania.

Przy okazji omawiania grupy metod dedykowanych systemom społecznym nie sposób pominąć zjawiska propagacji epidemii [43], propagacji wirusów komputerowych [121], [128] oraz propagacji podobieństw [49], [102].

### 2.2.4. Metody propagacyjne w systemach kolektywnych i kolaboratywnych

Proces propagacji danych w systemach rozproszonych widoczny jest szczególnie w systemach kolektywnych i kolaboratywnych opartych na SOA, technologiach związanych z usługami sieciowymi i zarządzaniem treścią, w negocjacjach związanych z alokacją zasobów [99], systemach mobilnych [94], [110], algorytmach wyznaczania tras [137], [145] i systemach samoorganizujących [14], [18], [48]. Ponadto propagacja danych pełni ważną rolę integrującą przy utrzymywaniu spójności systemów kolaboratywnych przez udział w optymalizacji algorytmów trasujących. Takim przykładem jest praca [137], w której m.in. opisano nowy algorytm wyznaczania tras wykorzystujący metrykę określającą długość ścieżki połączeń.

Koncepcja kierunkowej dystrybucji informacji występuje także w formie kolaboratywnej propagacji danych [64], [106], [123], adaptacyjnej alokacji zadań [44], [60], propagacji komunikatów [71], [75] i rekomendacji [157].

Wymienione przykłady nie wyczerpują wszystkich istniejących komputerowych metod i modeli propagacyjnych. Potwierdzają jednak szerokie wykorzystywanie mechanizmu propagacji danych w nowoczesnych systemach informatycznych.

## 2.3. Fizyczne modele propagacyjne

Przez ostatnie lata fizycy intensywnie badali i starali się wytłumaczyć przemiany fazowe i związane z nimi słabo przewidywalne zjawiska krytyczne [67]. Co więcej, zauważono, że układy przemian fazowych w pobliżu punktów krytycznych w związku z brakiem powtarzalności i dużymi fluktuacjami wyników stają się bezskalowe, co oznacza możliwość stosowania praw potęgowych podczas prowadzonych badań. Cecha bezskalowości okazała się powszechna wśród układów rzeczywistych nie tylko z obszaru zastosowań fizyki (sieci energetyczne, sieci komunikacyjne), ale także z biologii (sieci metabolizmu komórkowego, sieci oddziaływań między białkami), z techniki (sieci połączeń lotniczych, WWW) i socjologii (sieci społeczne). Najczęściej do opisu zjawisk krytycznych, takich jak: powstawanie/wymieranie gatunków, krachy/wzrosty na giełdzie, panika oraz rewolucja, są wykorzystywane dwa propagacyjne modele fizyki statystycznej – model Isinga oraz model perkolacji węzłowej.

Ogólny (magnetyczny) model Isinga polega na przypisaniu każdemu punktowi sieci zmiennej o wartości +1 lub –1, nazywanej spinem. Zakłada się, że spiny umieszczone w sąsiadujących węzłach dążą do tych samych wartości, gdyż spin można traktować jako źródło lokalnego pola magnetycznego. W przypadku występowania zewnętrznego pola magnetycznego spiny porządkują się zgodnie z tym polem. W przypadku braku zewnętrznego pola magnetycznego zachowanie spinów zależy natomiast od temperatury. Spiny posiadają ponadto własność nieskończonej korelacji, co oznacza, że nawet bardzo odległe spiny mają wpływ na swoje wartości.

Model Isinga najczęściej jest używany do modelowania opinii społecznych [134], [135], [140]. Spiny reprezentujące opinie poszczególnych jednostek odwzorowują sieć wzajemnych oddziaływań. Ponieważ spin ferromagnetyka jest źródłem pola magnetycznego, w ten sposób bezpośrednio wpływa na sąsiadów i propaguje swój stan na innych. Tak samo dzieje się z opinią przekazywaną najbliższym znajomym. Ponadto okazuje się, że układ spinów dąży do stanu o najniższej energii i osiąga go wtedy, kiedy spiny mają taką samą wartość. Jednakże możliwe fluktuacje parametrów, np. szum medialny w postaci plotki czy wykrycia afery gospodarczej, powodują, że osiągnięcie stanu minimalnego staje się niemożliwe. W takiej sytuacji ciągła przemiana fazowa nie pozwala na uzyskanie porozumienia społecznego. Niestabilność układu zachodzi zarówno powyżej temperatury krytycznej dla przemiany fazowej, jak

i w pobliżu punktu krytycznego. W drugim przypadku układ jest nieprzewidywalny i charakteryzuje się dużą zmiennością. Jeżeli w układzie nie ma zgodności opinii, to różnice są propagowane na otoczenie. W rezultacie społeczeństwo nie jest zdolne do podejmowania jakichkolwiek decyzji.

Najważniejszą właściwością modelu Isinga w przypadku sieci bezskalowej jest to, że znak magnetyzacji sieci różnej od zera zależy od najlepiej usieciowanego węzła. Dzięki temu pojedyncza jednostka o największym wpływie na otoczenie może praktycznie sterować opinią publiczną.

Drugi fizyczny model – model perkolacji węzłowej, znany także jako model przepływu, służy przede wszystkim epidemiologom. Teoria perkolacji w odniesieniu do sieci złożonych opisuje zachowanie połączonych grup wierzchołków i polega na losowym podziale wierzchołków lub krawędzi na dwa rozłączne zbiory [46]. Znany z literatury model epidemiologiczny [150] może być utożsamiany z modelem perkolacji wierzchołkowej. Model perkolacji dostarcza nam wielu informacji na temat możliwości rozprzestrzeniania się choroby. Przejście fazowe to tzw. próg epidemiologiczny, powyżej którego możliwy jest wybuch epidemii. Gdy skoreluje się proces perkolacji ze stopniem wierzchołka, wówczas można uzyskać odpowiedź na pytanie, jak przeciwdziałać rozprzestrzenianiu się choroby. Niestety, w tym modelu nie można wywnioskować o hipotetycznych zmianach ewolucji ognisk choroby w czasie.

Ze względu na istniejące, wyczerpujące opracowania teoretyczne oraz położony przez autora prezentowanej pracy nacisk na praktyczną weryfikację metod używanych w systemach rozproszonych, w dalszych rozdziałach nie będzie odwołań do modeli fizycznych. Obszerny materiał dotyczący ich wykorzystania do analizy zachowań sieci złożonych jest dostępny m.in. w pracach [46], [140], [148].

## 2.4. Praktyczne wykorzystanie mechanizmów propagacyjnych

Zagadnienie propagacji danych pojawia się coraz częściej w rozwiązaniach informatycznych i sprawia przy tym wiele problemów. Nieraz nie umiemy poradzić sobie z negatywnymi skutkami procesów propagacyjnych. Tak się dzieje w przypadku wirusów komputerowych, przed którymi do dzisiaj nie ma pełnej ochrony. Brakuje też umiejętności korzystania w pełni z dobrodziejstw fenomenu propagacji danych. W punktach 2.4.1 i 2.4.2 przedstawiono dwa przykłady dotyczące praktycznego wykorzystania mechanizmów propagacyjnych w systemach społecznych i sieciowych. Choć nie są to jeszcze gotowe rozwiązania informatyczne, gdyż m.in. nie zostały zweryfikowane w praktyce, uzyskane wstępne wyniki są bardzo interesujące i pokazują, jak duże możliwości kryją w sobie mechanizmy propagacyjne.

### 2.4.1. Przekazywanie zapytań w systemach społecznych

W społecznościach internetowych nie występuje tradycyjny model organizacji wiedzy i przepływu informacji bazujący na strukturze silnie zhierarchizowanej [146]. W jaki sposób zatem ludzie współpracują ze sobą, będąc równocześnie członkami wielu lokalizacyjnie rozproszonych zespołów. W szczególności – w jaki sposób uzyskują informacje od innych, jeśli nie wiedzą nawet tego, do kogo skierować pytanie. W ostatnich latach otworzyły się różne możliwości wspomagające szybką wymianę informacji między użytkownikami systemów społecznych, począwszy od poczty elektronicznej i mobilnych telefonów, aż po komunikatory internetowe i portale społecznościowe.

Jednym z ważnych problemów pojawiających się wraz z dużą liczbą różnych aplikacji jest identyfikacja użytkownika i propagacja jego profilu. Innym – uwzględnienie znajomych moich znajomych, którzy „mnożą się" w bardzo szybkim tempie. Wykrywanie wspólnych znajomych i rekomendowanie ich w sieciach kontaktów stało się już stałym zwyczajem wielu portali społecznościowych. Podobnie dzieje się z uporządkowaniem komunikacji według przyjętych zasad, priorytetów i zakazów. Praktycznie każdy serwer pocztowy oferuje mechanizm wykrywania i filtrowania niechcianego spamu. Jeszcze innym zagadnieniem jest tworzenie oraz korzystanie z repozytoriów zadawanych zapytań i uzyskiwanych odpowiedzi. Takim repozytorium mogą być listy kontaktów do działów obsługi tworzone automatycznie w wybranym okresie lub też listy problemów i ich rozwiązań utrzymywane przez samych użytkowników. Drugie rozwiązanie jest bardzo obiecujące, ponieważ angażuje bezpośrednio zainteresowanych członków społeczności i interaktywnie weryfikuje uzyskaną wiedzę.

Prezentowana propagacja zapytań w systemie społecznym polega na przekazywaniu pytania swoim znajomym z prośbą przesyłania go dalej, o ile odpowiedź nie będzie nadal znana. Proces ten odróżnia od innych podobnych mechanizmów to, że pytanie może być przekazywane w różny sposób, np. przez e-mail lub SMS, w czasie rozmowy telefonicznej, na zebraniu pracowników. Rozprzestrzenianie się zapytania jest ograniczone jedynie wiedzą i możliwościami pierwszych i kolejnych odbiorców. Nie ma tam wyodrębnionej jednostki zarządzającej. Odbiorcy generowanych zapytań komunikują się z kim chcą. Przede wszystkim jednak łączą się z tymi, kogo preferują oraz od kogo mogą liczyć na szybką i kompetentną odpowiedź. Przykładem systemu społecznego z propagacją zapytań jest SocLaKE [90] (omówienie jego działania będzie podstawą dalszych rozważań).

W systemie SocLaKE poza różnymi metodami komunikacji wykorzystywana jest wiedza na temat struktury organizacyjnej społeczności. System taki idealnie pasuje do nowoczesnych przedsiębiorstw o precyzyjnie zdefiniowanym zakresie obowiązków każdego pracownika. Dane o pracownikach, kompetencjach i ich służbowych obowiązkach na co dzień są wykorzystywane przez działy osobowe przedsiębiorstw.

W pierwszej chwili wydaje się, że system SocLaKE całkowicie nie nadaje się do struktur niezorganizowanych – mimo wszystko już po krótkim czasie działania systemu w strukturze o nieznanej organizacji można odkryć dotąd ukrytą przed nami wiedzę, np. ze ścieżek komunikacyjnych, w tym sprzężeń zwrotnych istniejących między użytkownikami.

Znalezienie eksperta, który może odpowiedzieć na pytanie w systemie rekomendacyjnym stosującym propagację zapytań, nie jest skomplikowane. System za każdym razem wyszukuje najlepszą drogę do eksperta, opierając się na indywidualnych preferencjach użytkowników. Można wybrać jedną z kilku proponowanych strategii, np. najszybszą albo najbardziej wiarygodną. Ponieważ system rekomenduje listę ekspertów, do których można wysłać pytanie, użytkownik dokonuje ostatecznego wyboru adresata. Jeśli uzyskana odpowiedź satysfakcjonuje pytającego, to proces propagacji jest zatrzymywany. W tym momencie należałoby przesłać wszystkim uczestnikom procesu poszukiwawczego informację, że problem został rozwiązany i nie jest już oczekiwana odpowiedź na to pytanie. Jeżeli jednak nikt ze znajomych nie zna odpowiedzi, mogą oni propagować pytanie do innych członków sieci społecznej. Nie powinni jednak wysyłać tej informacji do odbiorców znających problem. Wynika stąd, że wraz z pytaniem należy wysyłać listę adresów, co jednocześnie powoduje bardzo szybkie zwiększanie objętości propagowanej wiadomości. Ponadto uciążliwe stałoby się odfiltrowywanie tych, którzy otrzymali już to zapytanie. Z wymienionych powodów nie wprowadza się takiej kontroli kosztem często nadmiernie generowanego ruchu w systemie społecznym i zalewania użytkowników różnymi pytaniami z prośbą o pomoc. Jednym ze stosowanych w takiej sytuacji ograniczeń jest maksymalna liczba użytkowników, do których może trafić pytanie. Innym sposobem zachęcającym do działania może być informowanie użytkownika, jak długo zwleka z reakcją na przysłane pytanie i ile osób oczekuje od niego odpowiedzi. Jeszcze innym rozwiązaniem problemu nadmiernego rozpowszechniania informacji może być wprowadzenie czasu ważności określonego dla każdego propagowanego pytania. Po jego upłynięciu pytanie nie jest dalej rozpowszechniane, co blokuje ewentualną propagację nieaktualnych zapytań.

Jeśli już pytanie zostało przesłane ekspertowi, zwykle nie ma zainteresowania, co z nim dalej się dzieje. Często w ten sposób trafia się na użytkownika przetrzymującego pytania lub w ogóle nieodbierającego wiadomości. Aby temu zapobiec, można wprowadzić czas reakcji dla adresata – przed jego upłynięciem powinny przyjść potwierdzenia odebrania wiadomości. W przypadku braku takiego potwierdzenia zapytanie powinno być przekazane innemu użytkownikowi.

Na wysłane pytanie w sieci społecznej nadawca może otrzymać kilka odpowiedzi. Jeśli któraś z nich spełnia jego oczekiwania, można anulować pytanie. Wiąże się to z wygenerowaniem wiadomości o wycofaniu zapytania i uruchomieniem nowego procesu propagacyjnego. Jeśli proces anulowania będzie odbywał się bez udziału użytkowników, to jest szansa, że zostanie zaakceptowany, w przeciwnym wypadku

nie ma on sensu. Spełnienie oczekiwań nadawcy zapytania może też być rozłożone na kilka wiadomości i w podobny sposób należałoby rozdzielić przypadający stopień kompetencji na kilku ekspertów. W przypadku spełnienia oczekiwań nadawcy stopień kompetencji eksperta powinien wzrosnąć, natomiast w przypadku błędnej odpowiedzi stopień ten może zostać zmniejszony. Czasami wskazane by było utrzymywanie kilku stopni kompetencji, np. ze względu na dziedzinę problemu. Mówi się wtedy o kompetencji domenowej lub dziedzinowej. Takie uszczegóławianie prowadzi jednak do zwiększenia złożoności problemu, gdyż już samo przyporządkowanie pytania do jednej z klas sprawia do dzisiaj wiele trudności.

W systemie SocLaKE wykorzystano model stochastyczny, w którym każdy węzeł jest opisany przez 5 zmiennych: strategię, stopień współpracy, podobieństwo, kompetencję i poprawność. Prawdopodobieństwo znalezienia odpowiedzi przez użytkownika na odebrane pytanie zależy od tego, na ile jest on ekspertem, zna poprawną odpowiedź i jak zareagują pozostali zapytani użytkownicy sieci. Aby obliczyć to prawdopodobieństwo, należy znać wszystkie wartości wymienionych zmiennych dla każdego węzła sieci. Do wyznaczania listy polecanych ekspertów poza wartościami zmiennych wykorzystywano również algorytm genetyczny – okazał się on najlepszą strategią rekomendacyjną z 8 testowanych metod. Przeprowadzone eksperymenty pokazały, że zastosowanie rekomendacji do propagacji zapytań dało średnio ponad 5% wzrostu prawdopodobieństwa znalezienia akceptowalnej odpowiedzi oraz ponad 192% wzrostu prawdopodobieństwa w najlepszym uzyskanym przypadku.

### 2.4.2. Metoda identyfikacji komponentów krytycznych w systemie sieciowym

Nowoczesne systemy sieciowe charakteryzują się z jednej strony dużą dynamiką i złożonością, a z drugiej także wysoką awaryjnością pojedynczych elementów. Węzły mogą być wyłączane w dowolnym momencie, zarówno w bezprzewodowych sieciach sensorowych (ze względu na ograniczenia energetyczne), jak i w sieciach P2P po podjęciu takiej decyzji przez użytkownika. Chociaż oba rozwiązania dość dobrze tolerują przypadki wyłączenia pojedynczego węzła, istnieją takie komponenty, których ewentualna awaria może wygenerować olbrzymie koszty. Ponieważ nie wszystkie węzły są jednakowo ważne, więc można wybrać te najważniejsze i zadbać jedynie o ich stabilne działanie, czasami wiąże się to z oszczędzaniem ich zapasów energetycznych lub z częściową rekonfiguracją sieci. Obszary krytyczne mogą dotyczyć węzłów ważnych ze względu na udostępniane usługi sieciowe, posiadane zasoby czy strukturę połączeń z pozostałymi węzłami. W dwóch pierwszych przypadkach identyfikacja komponentów krytycznych jest niepotrzebna, gdyż zwykle parametry węzłów są raz specyfikowane i potem niezmieniane. Nawet w przypadku zmian można ponownie podać pełną ich listę. Ze względu na dynamikę współczesnego systemu sie-

ciowego inne rozwiązanie musi być zaproponowane do ewentualnych zmian strukturalnych.

W przypadku sieci statycznych znalezienie elementów krytycznych jest trywialne, np. przez przeszukiwanie w głąb. W przypadku sieci dynamicznych uzyskanie pełnej informacji o aktualnym stanie sieci wiąże się z dużymi kosztami komunikacyjnymi, dlatego takie podejście jest niepraktyczne. Wymagane jest inne rozwiązanie bazujące jedynie na informacji lokalnej przechowywanej przez każdy węzeł. Ponadto ze względu na redundancję połączeń nie jest też wymagane, aby metoda identyfikacji była bezbłędna, w pełni wystarczający jest poziom dokładności rzędu 90%.

Zarówno sieci P2P, jak i bezprzewodowe sieci sensorowe najczęściej implementują algorytm zalewowy (zob. rozdz. 4.4). Węzeł propaguje więc informację do wszystkich swoich sąsiadów z wyjątkiem tego, od którego otrzymał wiadomość. Ponadto każda wiadomość jest oznaczona identyfikatorem węzła początkowego oraz czasem wygenerowania wiadomości. Proces propagacji dla pary (wiadomość, nadawca) jest niepowtarzalny, co oznacza, że jeśli wiadomość zostanie powtórnie odebrana przez ten sam węzeł, to nie będzie dalej przesyłana. Takie założenia przyjęto w publikacji [55] – rozwiązanie z wykorzystaniem w opisie pojęcia z klasycznej teorii grafów [30] zostanie zaprezentowane w dalszej części pracy.

W skrócie metoda identyfikacji i eliminacji komponentów krytycznych w grafie działa w trzech zasadniczych krokach. W pierwszym konstruowana jest struktura przechowująca informacje o składowych dwuspójnych, które mogą tworzyć sąsiedzi danego węzła. Dodatkowo węzły zapamiętują wszystkie odebrane wiadomości w postaci list par (wiadomość, nadawca). Jeśli po odebraniu wszystkich wiadomości dla danego węzła powstała z jego sąsiadów tylko jedna składowa dwuspójna, to nie jest on węzłem rozcinającym (punktem artykulacji). W przypadku jednak utworzenia wielu składowych dwuspójnych nie mamy jeszcze pewności, czy badany element jest węzłem rozcinającym. Wtedy rozpoczyna się drugi krok polegający na minimalizacji liczby składowych dwuspójnych przez ich łączenie. Jeśli dalej nie uzyskamy jednej dwuspójnej składowej, to dany element jest węzłem rozcinającym. W trzecim kroku po identyfikacji węzła rozcinającego w łatwy sposób można zabezpieczyć sieć przed awarią przez dodanie nowego połączenia między węzłami z różnych wygenerowanych składowych dwuspójnych. W konsekwencji takiego działania dwie składowe dwuspójne zostaną połączone i węzeł przestanie być punktem artykulacji.

Zaprezentowana metoda została zweryfikowana eksperymentalnie. Uzyskane wyniki potwierdziły skuteczność jej działania i pokazały, że propagacja zalewowa może być efektywnie wykorzystana najpierw do wykrywania krytycznych komponentów struktury sieciowej, a następnie do jej rekonfiguracji w celu zwiększenia niezawodności połączeń między węzłami.

## 2.5. Podsumowanie

Bezpośrednie przeniesienie zachowań ze schematów interakcyjnych do metod propagacji w pełni uzasadnia wprowadzenie podziału metod propagacyjnych na grupy: kolektywną, kooperatywną i kolaboratywną. Ponadto ze względu na ocenę rezultatu otrzymanego w wyniku działania procesu propagacyjnego można wyróżnić propagacje „pozytywne" i „negatywne".

Z wyborem mechanizmu interakcji łączy się selekcja odpowiedniego paradygmatu. Jego wybór jednak nie jest już tak prosty, jak w przypadku schematu interakcyjnego. Coraz częściej w złożonych systemach sieciowych możemy spotkać kilka jednocześnie przeplatających się różnych teorii. Powszechnie stosowana hybrydyzacja metod i modeli jest bezpośrednim następstwem takiego zjawiska i niesie ze sobą duże możliwości, ale też większą złożoność. Oparcie procesu propagacji danych na nowych paradygmatach, szczególnie na inspiracji przyrodą, daje szansę uzyskania lepszych rezultatów niż w przypadku standardowej wymiany komunikatów.

Opisane w rozdz. 2 komputerowe metody i modele propagacyjne dla systemów rozproszonych można przyporządkować do następujących grup systemów: metody propagacji w systemach wieloagentowych, społecznych, kolektywnych i kolaboratywnych. Na osobne wyróżnienie zasługuje grupa metod propagacyjnych bazujących na inspiracjach biologicznych. Dwa najbardziej znane fizyczne modele propagacyjne – model Isinga oraz model perkolacji węzłowej zostały pozytywnie zweryfikowane przy badaniu opinii społecznych oraz w analizie przebiegu epidemii. Nie ogranicza to w żadnym stopniu innych możliwych zastosowań.

W literaturze naukowej ciągle brak jest wielu opracowań dotyczących efektywnych metod propagacji danych używanych we współczesnych systemach. Przy tym warto mieć na uwadze, że rozwiązanie problemów propagacyjnych *post factum* w działających systemach jest zagadnieniem bardzo kosztownym i czasochłonnym, wymagającym częściowej lub nawet całkowitej rekonfiguracji systemu. Ewentualne modyfikacje mogą znaleźć się w konflikcie z innymi wymaganiami podanymi przez współpracujące ze sobą systemy. Aby więc w pełni wykorzystać dowolny mechanizm propagacyjny, należy go uwzględnić na każdym etapie tworzenia systemu, zaczynając już od etapu projektowania. Przytoczone przykłady systemu społecznego z rekomendacją i systemu sieciowego o zidentyfikowanych węzłach krytycznych pokazują, że wykorzystanie metod propagacyjnych może prowadzić do spełnienia bardzo praktycznych oczekiwań użytkowników współczesnych systemów komputerowych.

# 3. Propagacja zadań

W rozdziale 3 zaproponowano i zbadano mechanizm propagacji zadań pozwalający na uzyskanie większej mocy obliczeniowej systemu rozproszonego. Dynamiczny proces przesyłania zadań do wykonania jest ściśle skorelowany z jednostkowym obciążeniem poszczególnych elementów systemu. Wprowadzone pojęcia progu opłacalności i strategii limitowania propagacji skutecznie ograniczają częste, a przy tym negatywne, zjawisko cyklu propagacyjnego, które może pojawić się między podobnymi węzłami architektury rozproszonej. Przytoczone i omówione przykłady obrazują szeroki zakres występujących tu problemów metodologicznych oraz implementacyjnych.

## 3.1. Migracja, klonowanie czy propagacja danych

W badaniach nad metodami propagacyjnymi szczególne znaczenie ma odwołanie się do idei migracji danych, znanej przede wszystkim z baz danych [76], i poszerzenie tego zjawiska do propagacji danych. Odwołanie się do tego przykładu pozwala wyraźnie wskazać na aspekty praktyczne proponowanych metod także dla systemów wieloagentowych czy sieci społecznych. Idea migracji danych została wprowadzona do literatury światowej na początku lat 90. i stała się niemal natychmiast jednym z najczęściej używanych pojęć o charakterze dynamicznym, np. w dziedzinie rozproszonych baz danych jest używana w postaci operacji replikacji schematów oraz danych. Mimo jednak relatywnie dużej popularności tego procesu w rzeczywistych wdrożeniach, dotychczasowe prace naukowe dotyczące migracji czy propagacji danych są nadal nieliczne, szczególnie w zakresie współczesnych złożonych systemów sieciowych. W istniejących opublikowanych pracach problem propagacji bardzo często traktowany jest jako prosta kontynuacja badań nad migracją danych.

W informatyce oprócz pojęcia migracji występuje operacja klonowania danych polegająca na tym, że obiekt podstawowy pozostaje w pierwotnej lokalizacji, jego kopia natomiast jest tworzona w innym miejscu. Nowy obiekt różni się od podsta-

wowego jedynie identyfikatorem. W prezentowanej pracy, generalnie, przyjęte zostanie ogólniejsze pojęcie propagacji, gdyż według autora migracja i klonowanie są tylko jego szczególnymi przypadkami.

Efektywne wykorzystywanie zasobów sieciowych jest jednym z ważniejszych zadań stojących m.in. przed projektantami i programistami systemów informatycznych. Na złożoność problemu wpływa rozproszona natura sieci komputerowych i wysokie wymagania co do szybkości transmisji przesyłanych danych. Wśród przyjętych rozwiązań jest zastosowanie zdecentralizowanej i odpornej na błędy propagacji danych. Mechanizm propagacyjny w dużym stopniu eliminuje problemy związane ze skalowalnością i niezawodnością. Innym alternatywnym rozwiązaniem jest, stosowany np. w sieciach komputerowych typu P2P, mechanizm samoorganizacji [25].

Rosnąca liczba komputerów połączonych w jednej sieci oznacza więcej dostępnych zasobów, co w rezultacie daje szersze możliwości dystrybucji zadań. Jeśli jednak liczba węzłów będzie za duża, możliwości scentralizowanego efektywnego sterowania spadają, podobnie jak skalowalność całego systemu. Koszty transmisji też zwiększają się, bo wzrasta czas połączeń. Wysłanie komunikatu z wybranego komputera do pozostałych i odebranie potwierdzenia trwa za długo. Dobrym rozwiązaniem w takim przypadku jest zdecentralizowanie systemu i wykorzystanie migracji danych [52], [74].

W rozdziale 3 zwrócono przede wszystkim uwagę na możliwości adaptacyjne procesu propagacji zadań wobec zmieniających się warunków zewnętrznych środowiska implementacyjnego. Analiza problemu najczęściej skupia się na skróceniu czasu wykonania oraz zmniejszeniu wielkości przesyłanych danych. Można to osiągnąć m.in. dzięki odpowiedniej zmianie liczby uruchomionych zadań w węźle sieci. Dopasowanie metody dystrybucji zadań do używanej struktury sieciowej jest możliwe na przykład przy zastosowaniu algorytmu zrównoważonego obciążenia [20], [63].

Powszechnie przyjmuje się, że propagacja zadań musi spełniać następujące wymagania:

- Transparentność – proces propagacyjny nie powinien być zauważalny dla użytkownika.
- Dostępność – „wędrujące dane” powinny być dostępne na dowolnym etapie przetwarzania, np. propagowane zadania powinny odbierać wiadomości, jak i generować je dla innych.
- Atomowość – pojedyncza propagacja musi się zakończyć powodzeniem lub zostać odwołana.
- Spójność – po wykonywaniu propagacji nie będą naruszane zasady integralności danych.
- Wydajność – proces propagacyjny nie może powodować widocznego opóźnienia wykonywania zadania, a wręcz przeciwnie, oczekiwany jest zysk z jego sto-

sowania. Decyzja o ponownej relokacji zadania może być podjęta dopiero po powtórnym oszacowaniu potrzeb i kosztów.

W omawianym przypadku sposób implementacji propagacji zadań bazuje na przesłance, że w czasie zmiany lokalizacji zadań nie są wykonywane żadne inne operacje. Ograniczenie to, narzucone przez wybraną platformę implementacyjną, jest dopuszczalne, gdyż nie wpływa na wykonywanie zaplanowanego podstawowego zadania. Podczas procesu propagacyjnego zmienia się stan sieci i bardzo często nie możemy przewidzieć wszystkich jego skutków, mimo przestrzegania zasad spójności. Ponadto pojedyncze zadanie może być poddane procesowi propagacji wielokrotnie. Standardowo, za poprawność złożonych operacji odpowiadają m.in. ograniczenia integralnościowe, polityka bezpieczeństwa. Sprawa bardzo się jednak komplikuje, gdy propagacja będzie dotyczyć właśnie ograniczeń integralnościowych czy zasad bezpieczeństwa [13], [16], [32].

W dalszej części rozdziału zaprezentowano architekturę rozproszoną umożliwiającą propagację zadań, następnie przeprowadzono analizę różnych metod propagacji, a na koniec wykonano serię testów w środowisku heterogenicznym i homogenicznym. W celu uzyskania lepszej przejrzystości opisu pojęciem obiektu i zadania posłużono się zamiennie.

## 3.2. Wyznaczanie liczb pierwszych

Przy wyborze odpowiedniego algorytmu używanego w procesie propagacji zadań przede wszystkim kierowano się możliwością jego dekompozycji na jednostkowe zadania, które mogą być wykonywane jednocześnie [31]. Z istniejących 5 podstawowych sposobów wyodrębniania zadań: funkcjonalnego, danych, rekursywnego, eksploracyjnego i spekulatywnego zdecydowano się wybrać drugi z wymienionych sposobów. Realizacja dekompozycji danych jest stosunkowo prosta, szczególnie dla środowiska wieloagentowego. Z tego względu zdecydowano się na implementację środowiska testowego na tej platformie. Jednakże nic nie stoi ma przeszkodzie, by proponowane rozwiązania także mogły być używane na innych platformach sieciowych po uprzednim dostosowaniu do nowych wymagań.

Liczbę jednostkowych zadań, które mogą być rozwiązywane jednocześnie, wyznacza tzw. stopień współbieżności problemu. Im jest on większy, tym lepiej będzie wykorzystywał właściwości systemu rozproszonego. Ponadto im częstość komunikacji i synchronizacji operacji jest mniejsza, tym opracowanie efektywnego algorytmu jest łatwiejsze. Tak jest w przypadku problemu wyznaczania liczb pierwszych – w nim jednostkowe zadania są niezależne od siebie i nie wymagają złożonej komunikacji. Wystarczy przekazywać dane do obliczeń, które tylko czasami muszą być modyfikowane ze względu na występowanie propagacji. Ostatnim etapem przetwarzania będzie scalenie dostarczanych przez węzły wyników.

**Algorytm** WLP(*a*, *b*)

Parametry wejściowe: przedział [*a*, *b*].

Wynik: lista wyznaczonych liczb pierwszych.

```
1:  begin
2:  for each liczby naturalnej x z przedziału [a, b] do
3:      oblicz pierwiastek kwadratowy c = sqrt(x);
4:      sprawdź, czy istnieje liczba naturalna z przedziału [2, c], która dzieli x bez
        reszty;
5:      jeśli nie istnieje, dodaj x do listy liczb pierwszych;
6:  end for;
7:  end;
```

Algorytm 3.1. Wyznaczanie liczb pierwszych

W celu analizy zagadnienia propagacji zadań w architekturze rozproszonej odniesiono się do problemu wyznaczania liczb pierwszych. W skrócie wygenerowano określoną liczbę liczb pierwszych począwszy od zadanej. W środowisku sieciowym wszystkie dostępne jednostki obliczeniowe będą wyznaczać liczby pierwsze równolegle, by efektywnie znaleźć rozwiązanie. Mając $n$ danych wejściowych, można podzielić je na $p$ części, $p < n$, o rozmiarach co najwyżej $[n/p]$. Wtedy stopień współbieżności algorytmu rozproszonego skonstruowanego przy użyciu dekompozycji danych wynosi $p$. Są więc dokładnie określone przedziały liczb i wykonanie tego zadania nie jest trudne. Problem polega jednak na tym, żeby jednostki szybsze mogły wyszukać więcej liczb. W tym celu wykorzystano mechanizm propagacyjny.

Algorytm 3.1 bazuje na następującej regule: jeśli liczba naturalna $x$ większa od 1 nie jest podzielna przez żadną z liczb pierwszych nie większych od pierwiastka kwadratowego z $x$, to $x$ jest liczbą pierwszą. Implementacja algorytmu polega na tym, że dla każdej liczby naturalnej $x$ z przedziału [$a$, $b$] wylicza się pierwiastek kwadratowy $c$ = sqrt($x$) i sprawdza, czy istnieje liczba naturalna z przedziału [2, $c$], która dzieli $x$ bez reszty. Jeśli nie istnieje, dodaje się $x$ do listy liczb pierwszych. Nie jest to najszybsza metoda, jednak celem nie jest projektowanie nowego algorytmu wyznaczania liczb pierwszych, ale pokazanie, jak można wykorzystać mechanizm propagacyjny do przyspieszenia obliczeń w systemie rozproszonym. Istnieją ku temu dwie przesłanki. Po pierwsze, podstawowy przedział wyszukiwanych liczb pierwszych może być dzielony na mniejsze przedziały. Po drugie, wynikowa lista wyznaczonych liczb pierwszych może być scalana z innymi uzyskanymi listami od pozostałych jednostek obliczeniowych.

## 3.3. Propagacyjna architektura rozproszona

Aby proces propagacji danych mógł zachodzić, należy zdefiniować odpowiednią architekturę rozproszoną. Pobocznym problemem do rozwiązania w proponowanej architekturze jest znalezienie takiej konfiguracji dla mechanizmu propagacyjnego, by działał on prawidłowo zarówno w środowisku heterogenicznym, jak i homogenicznym. Jeżeli za kryterium oceny przyjęto czas wykonania powierzonego zadania, to dzięki propagacji danych powinno się osiągnąć znaczące skrócenie tego czasu w zależności od używanego algorytmu i przyjętych założeń.

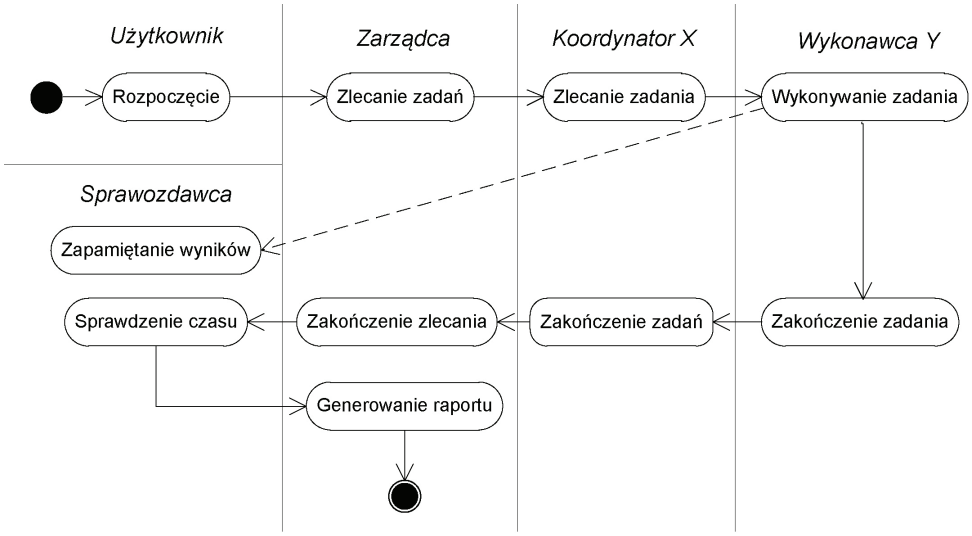
W architekturze rozproszonej, która ma umożliwić wyznaczanie liczb pierwszych, wyróżnione zostały dwa podstawowe obiekty – *Wykonawca*, w postaci węzła obliczeniowego, oraz *Zarządca*. Węzły obliczeniowe są dedykowane wyszukiwaniu liczb pierwszych, natomiast *Zarządca* dystrybuuje zadania jednostkowe do węzłów oraz zapamiętuje częściowe rezultaty w pliku zewnętrznym. *Zarządca*, w przeciwieństwie do *Wykonawcy*, nie uczestniczy w wyznaczaniu liczb pierwszych. *Zarządca* wie, do ilu węzłów obliczeniowych może propagować zadanie, gdyż na początku każdy węzeł rejestruje się przez wysłanie wiadomości do *Zarządcy* o stanie gotowości. W tym rozdziale przyjęto stałe założenie o niezmienności środowiska, bez uwzględniania ewentualnych zmian parametrów otoczenia w trakcie wykonywania zadania. Wprowadziłoby to jedynie niepotrzebne skomplikowanie działania algorytmu oraz niejednoznaczność interpretacji otrzymanych wyników. Na początku, ponieważ nic nie wiadomo o możliwościach węzłów obliczeniowych, zadany przedział liczbowy dzieli się przez liczbę dostępnych węzłów na równe części. W czasie obliczeń węzły przesyłają asynchronicznie, partiami do *Zarządcy* znalezione liczby pierwsze. Ostatnie przesłanie wyników obliczeń kończy wykonywanie zadania.

### 3.3.1. Diagram czynności

Podstawową funkcją proponowanego systemu jest dystrybucja zadania między węzłami obliczeniowymi oraz odebranie od nich wygenerowanych wyników. Można w nim wyróżnić pięć typów obiektów: *Użytkownika*, *Sprawozdawcę*, *Zarządcę*, *Koordynatora* i *Wykonawcę*, co pokazano na rys. 3.1. Widać, że przetwarzanie zaczyna się od odebrania przez *Zarządcę* wiadomości z podanym zakresem wyznaczenia liczb pierwszych. W następnym kroku *Zarządca* przesyła zadanie do *Koordynatora*. Na schemacie umieszczono jednego *Koordynatora* i jednego *Wykonawcę*, jednak w rzeczywistości jest ich wielu, stąd pojawiające się dodatkowe oznaczenia. *Wykonawca* wykonuje swoje zadanie, po którym każdorazowo przysyła wynik *Sprawozdawcy* (na rysunku oznaczono strzałką przerywaną). Możliwe procesy propagacji zadań zachodzą między *Zarządcą*, *Koordynatorem* i *Wykonawcą*.

Podczas ostatniego etapu wykonywania zadania, po wyznaczeniu wszystkich liczb pierwszych dla danego zakresu, *Wykonawca* raportuje koniec obliczeń *Koordynatorowi*,

a ten wysyła wiadomość do *Zarządcy* o zakończeniu przetwarzania. Ze względu na założoną asynchroniczność działania, co w tym przypadku znacznie zwiększa efektywność, *Sprawozdawca* sprawdza, czy nie został przekroczony dopuszczalny czas przetwarzania. Jeśli ma to miejsce, *Sprawozdawca* informuje *Zarządcę* o zakończeniu przetwarzania, a ten wyświetla uzyskany zbiorczy wynik, nazwany raportem.




Rys. 3.1. Diagram czynności

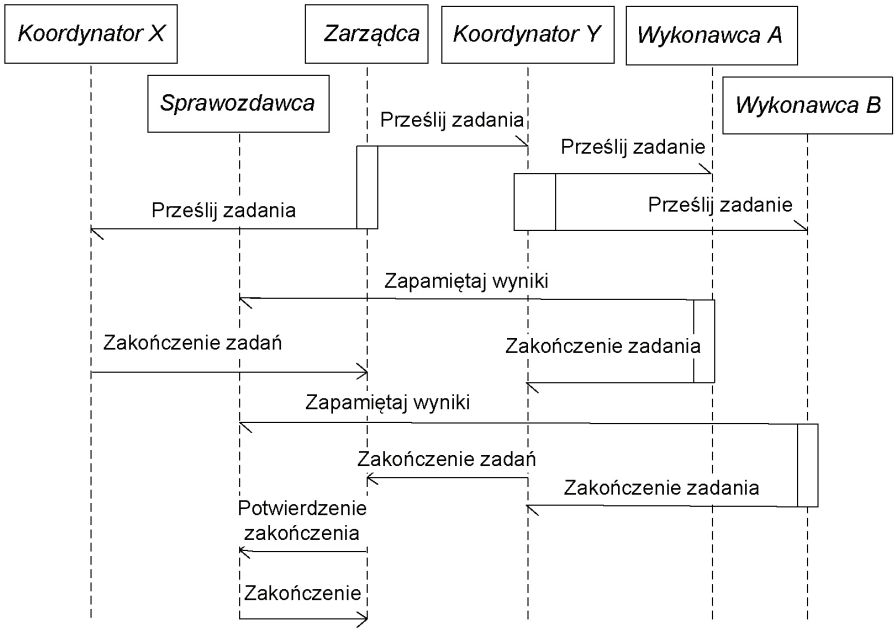
## 3.3.2. Diagram sekwencji

Na diagramie sekwencji (rys. 3.2) można prześledzić wszystkie przesyłane wiadomości między elementami systemu. W symboliczny sposób zobrazowano jednego *Sprawozdawcę*, jednego *Zarządcę*, ponadto *Wykonawcę A*, *Wykonawcę B* oraz *Koordynatora X* i *Koordynatora Y*.

*Zarządca* wysyła asynchronicznie *Koordynatorowi* wiadomość zawierającą zadanie do wykonania. Po tym, jak *Koordynator* odbierze wiadomość, natychmiast asynchronicznie przekazuje ją do *Wykonawcy*. W międzyczasie inna wiadomość wysłana przez *Zarządcę* mogła trafić do innego *Koordynatora*. W ten sposób, kiedy jeden *Wykonawca* może kończyć wykonywanie zleconego zadania, inny może je dopiero rozpoczynać.

Zaprezentowany sposób przesyłania wiadomości może powodować następujący problem. Gdy *Wykonawca* kończy pracę, wysyła o tym wiadomość z otrzymanymi wynikami do *Sprawozdawcy*. Wysyła też wiadomość do lokalnego *Koordynatora*. Kiedy *Koordynator* odbierze potwierdzenie zakończenia wykonywania zadań od wszystkich *Wykonawców*, generuje wiadomość do *Zarządcy*, a ten, jeśli otrzyma taką wiadomość od wszystkich *Koordynatorów*, wysyła wiadomość końcową do *Sprawoz-*

*dawcy*. Ponieważ *Koordynatorzy* i *Wykonawcy* działają asynchronicznie, może się zdarzyć, że wiadomość od *Wykonawcy* do *Sprawozdawcy* dotrze później niż sekwencja końcowych wiadomości od *Wykonawcy* do *Koordynatora*, a potem od *Koordynatora* do *Zarządcy* i od *Zarządcy* do *Sprawozdawcy*. Aby przeciwdziałać takiej sytuacji, wprowadzono dodatkowe opóźnienie dla *Sprawozdawcy*, które wymusza oczekiwanie na odebranie wszystkich wyników. Dopiero po upłynięciu tego czasu *Sprawozdawca* może wysłać kończącą wiadomość do *Zarządcy*, co przynosi wygenerowanie finalnego raportu. W tym momencie kończy się też cały proces.

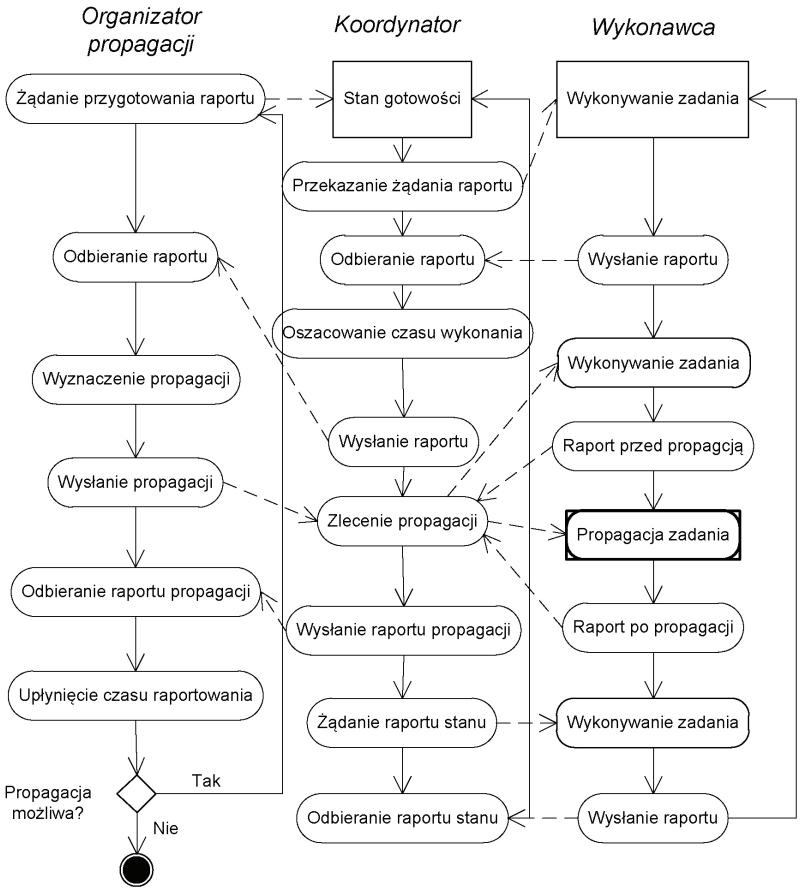



Rys. 3.2. Diagram sekwencji

## 3.3.3. Diagram propagacji

Działanie procesu propagacyjnego w schemacie powiązań między *Organizatorem* propagacji, *Koordynatorem* i *Wykonawcą* można prześledzić na diagramie propagacji (rys. 3.3). Proces zaczyna się od *Organizatora* propagacji, który otrzymuje od *Zarządcy* informację o rozpoczęciu działania. Po upłynięciu czasu potrzebnego na opracowanie raportu wydajności sprawdzane są warunki wstępne uruchomienia ewentualnych propagacji. Warunki przedstawiają się następująco: przede wszystkim musi istnieć aktywny *Wykonawca* i zadanie o oczekiwanym czasie wykonania dłuższym niż próg opłacalności propagacyjnej dla tego *Wykonawcy* (zob. p. 3.5.1). Tylko w takim przypadku algorytm propagacji rozpoczyna działanie.

Najpierw *Organizator* propagacji wysyła zapytanie o raport do wszystkich uczestniczących *Koordynatorów*. Następnie *Koordynatorzy* przekazują to pytanie do swoich *Wykonawców*. *Wykonawcy* natomiast raportują wstecz, ile obliczeń wykonali do tej pory. Potem *Koordynatorzy* odsyłają te informacje *Organizatorowi* propagacji. Dopiero wtedy *Organizator* propagacji wysyła propozycje zmian propagacyjnych. Na tej podstawie każdy *Koordynator* wie dokładnie, ile i gdzie pojawią się nowe zadania, a ile i skąd zostaną usunięte. Kiedy *Wykonawca* otrzyma taką wiadomość, przerywa obliczenia i uruchamia mechanizm propagacyjny. Prawie równocześnie *Wykonawca* informuje swojego lokalnego *Koordynatora* o stanie wykonanych obliczeń. Po przejściu zadania do nowej lokalizacji *Wykonawca* komunikuje się z nowym lokalnym *Koordynatorem* i powraca do wykonywania przerwanych obliczeń. W zależności od zleconego rodzaju propagacji *Wykonawca* może być zatrzymany i następnie usunięty lub też pozostawiony, jednak z ograniczonym zakresem liczb do




Rys. 3.3. Diagram propagacji

wyszukiwania. W tym czasie *Organizator* propagacji czeka na wykonanie zleconych operacji. Po uzyskaniu potwierdzenia od wszystkich *Koordynatorów*, do których zostały przesłane zlecenia propagacyjne, *Organizator* propagacji przechodzi do stanu początkowego.

Propagacja zlecona przez *Organizatora* propagacji może zostać odrzucona przez *Koordynatora*, jeżeli proponowany do propagacji *Wykonawca* zakończył działanie. Podobnie propagacja może zostać odrzucona, o ile dotyczy ostatniego *Wykonawcy* w węźle obliczeniowym. W środowisku jednorodnym takie ograniczenie ma sens, natomiast w środowisku niejednorodnym niekoniecznie. Jeśli tylko w wyniku działania algorytmu uzyskamy przyspieszenie wykonywania obliczeń, to taka propagacja powinna być przeprowadzona.

W celu uproszczenia opisu przyjęto dalej, że *Wykonawca* będzie uruchamiał jedno zadanie. Jeśli ma zostać przetworzona seria równoległych zadań w pojedynczym węźle, wystarczy utworzyć kilku *Wykonawców*. W związku z tym propagację zadań implementuje się tak samo, jak propagację *Wykonawców*. Dopuszczone wtedy zostanie przetwarzanie zadań heterogenicznych. W przypadku kilku takich zadań u jednego *Wykonawcy* sprawa wyboru zadania nie jest już trywialna. Ponadto nie ogranicza się w ten sposób mechanizmu propagacyjnego, gdyż dystrybucję zbioru zadań można rozłożyć na sekwencję propagacji pojedynczych zadań dla tego samego węzła docelowego. Jedyną widoczną różnicą może być zmniejszenie efektywności przetwarzania i w rezultacie dłuższy czas obliczeń.

### 3.3.4. Parametry i opis propagacji

Idea przedstawionego mechanizmu propagacyjnego opiera się na metodzie równoważenia obciążenia [44] i pozwala na efektywniejsze wykorzystanie dostępnych zasobów [8], [28]. Procedura ta może być realizowana programowo, sprzętowo lub przez integrację programowo-sprzętową. Istnieje wiele wariantów algorytmów bazujących na podstawowej wersji takich, jak: równoważenie przez odpytywanie, rozgłaszanie czy dyfuzję naturalną [127]. Można też maksymalnie obciążać najszybszy węzeł, by skrócić sumaryczny czas przetwarzania. Niemniej jednak znalezienie najlepszego algorytmu nie było celem przedstawionych badań. Analizowany algorytm ma jedynie zilustrować mechanizm propagacji i pokazać, że może być skutecznie wykorzystany przy rozwiązywaniu studiowanego zagadnienia.

Podstawową kwestią przy określaniu wydajności propagacji jest przyjęcie właściwej metody oceniającej. Na początku eksperymentu nie są znane parametry techniczne węzłów obliczeniowych. Dopiero, kiedy *Koordynatorzy* otrzymają informacje od podległych im *Wykonawców*, poznany zostaje przybliżony czas zakończenia przetwarzania zadań. Jest to kluczowy element algorytmu. W rzeczywi-

stości *Wykonawcy* dostarczają w raportach dwa parametry – zakres liczb do przeszukania oraz bieżącą pozycję sprawdzania. Na podstawie aktualnego i poprzedniego raportu można wywnioskować, ile liczb zostało sprawdzonych i w jakim czasie. W ten sposób można oszacować wydajność każdego węzła obliczeniowego w badanym środowisku.

Organizator propagacji przechowuje listę z danymi na temat wszystkich węzłów obliczeniowych. Jest ona wykorzystywana w procesie propagacyjnym. Na tej liście przechowujemy identyfikator węzła, moc węzła (*MW*), prognozowany czas obliczeń węzła ($t_{pcw}$), średni prognozowany czas obliczeń w węźle $(\overline{t_{pcw}})$, lokalny średni prognozowany czas obliczeń dla zadania w węźle $(\overline{t_{pcz}})$ oraz maksymalną dopuszczalną zmianę liczby zadań węzła (*MZLZ*). Moc węzła zdefiniowano za pomocą następującego wzoru:

$$MW(i) = \frac{\text{card}(ZWL(i,t)) - \text{card}(ZWL(i,t'))}{t - t'} \tag{3.1}$$

w którym: $t$ – aktualny czas, $t'$ – poprzedni czas, *ZWL* – zbiór wyszukanych liczb w węźle obliczeniowym $i$ w chwili $t$.

We wzorach (3.2)–(3.4) przedstawiono trzy kolejne parametry z listy: $t_{pcw}$, $\overline{t_{pcw}}$ i $\overline{t_{pcz}}$

$$t_{pcw}(i) = \frac{\text{card}(ZWL(i))}{MW(i)} \tag{3.2}$$

$$\overline{t_{pcw}} = \frac{\sum_{i=1}^{LW} t_{pcw}(i)}{LW} \tag{3.3}$$

przy czym *LW* – liczba węzłów obliczeniowych

$$\overline{t_{pcz}}(i) = \frac{t_{pcw}(i)}{LZW(i)} \tag{3.4}$$

*LZW* – liczba zadań w węźle. Na tej podstawie dopuszczalna zmiana liczby zadań w węźle

$$MZLZ(i) = \left\lfloor \frac{\overline{t_{pcw}} - t_{pcw}(i)}{\overline{t_{pcz}}(i)} \right\rfloor \tag{3.5}$$

Parametr *MW* charakteryzuje szybkość działania węzła względem ostatniego pomiaru. Wartość $t_{pcw}$ to prognozowany czas zakończenia obliczeń w węźle względ-

dem szybkości jego działania. Średni prognozowany czas zakończenia obliczeń dla dowolnego węzła określa $\overline{t_{pcw}}$, natomiast $\overline{t_{pcz}}$ – prognozowany czas przetwarzania dla pojedynczego zadania w węźle. Ostatni parametr *MZLZ* jest liczbą zadań, której nie można przekroczyć przy dodawaniu lub usuwaniu zadań z węzła. Zauważyć trzeba, że wartości *MZLZ* mogą być dodatnie lub ujemne. Ponieważ suma tych wartości może się nie bilansować, należy wprowadzić algorytm korygujący. W najprostszej wersji wybiera się parę węzłów o maksymalnych wartościach (max ujemna, max dodatnia) i planuje się dla tej pary propagację zadań o rozmiarze nieprzekraczającym żadnej z wartości *MZLZ*. Taka para węzłów automatycznie jest pomijana przy wyszukiwaniu kolejnej pary. Planowanie propagacji kończy się, gdy nie ma już niezaplanowanych węzłów lub nie można znaleźć par o wartościach ujemnych i dodatnich. Normalizacja liczby zadań na węźle może też przebiegać w inny sposób, np. przez wyszukanie maksymalnej wartości dodatniej i przesyłanie tam zadania z dowolnego węzła o ujemnej wartości *MZLZ*. Wprawdzie można nie uzyskać zrównoważenia obciążenia, ale można się spodziewać przyspieszenia obliczeń. Procedura zostaje powtarzana tak długo, aż będą przeniesione nadmiarowe zadania z przeciążonych węzłów. Pamiętać jednak trzeba o nieprzekraczaniu wyliczonych wartości *MZLZ*.

**Przykład 3.1**

Wzięty pod uwagę zostanie układ trzech węzłów obliczeniowych opisany następującymi parametrami: card(*ZWL*(1)) = 1000, *MW*(1) = 10, card(*ZWL*(2)) = 5000, *MW*(2) = 100 i card(*ZWL*(3)) = 45 000, *MW*(3) = 150 oraz początkową liczbą 10 zadań w każdym węźle. Ponieważ $t_{pcw}$(l) = 100, $t_{pcw}$(2) = 50 i $t_{pcw}$(3) = 30, więc po wyliczeniach: $\overline{t_{pcw}} = 60$, *MZLZ*(l) = –4, *MZLZ*(2) = +2 i *MZLZ*(3) = +10, zatem po normalizacji pary węzłów o skrajnych wartościach uzyskuje się *LZW*(1) = 6, *LZW*(2) = 10 i *LZW*(3) = 14.

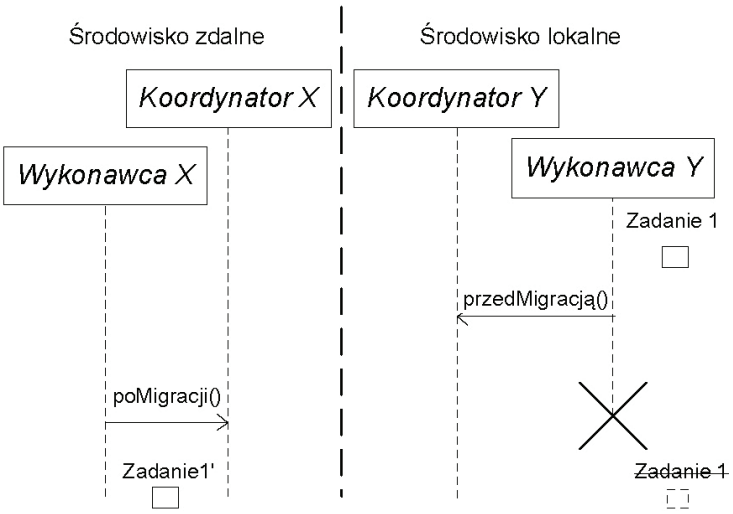
## 3.4. Rodzaje propagacji i schemat komunikacji

Wyróżnia się trzy rodzaje propagacji zadań: migrację, replikację i promocję.

### 3.4.1. Migracja zadań

Najczęściej propagację danych tworzy następująca sekwencja czynności: zapamiętanie stanu, uruchomienie serializacji, wysłanie binarnego obrazu danych pod

wskazane miejsce docelowe, deserializacja po dotarciu do celu i odtworzenie zapamiętanego stanu. Na podkreślenie zasługuje to, że proces propagacji zadań jedynie nieznacznie różni się od ogólnego schematu działania propagacji danych, dostosowując go do wymogów implementacyjnych przekazywania danych i kodu ze środowiska lokalnego do zdalnego. Dlatego też wymienione rodzaje propagacji zadań można uogólnić i nazwać odpowiednio: migracją, replikacja i promocją danych.

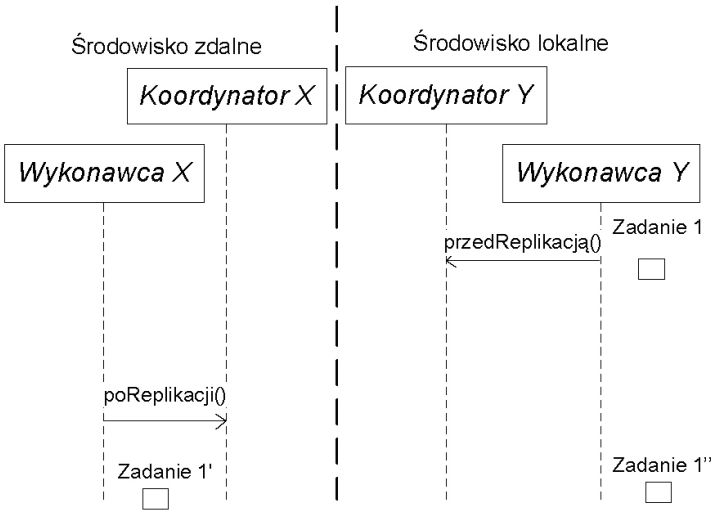



Rys. 3.4. Diagram migracji zadań

Podstawowym celem propagacji zadań jest realizacja postulatu mobilności, który jest rozumiany jako możliwość zmiany lokalizacji wykonywania zadania. W przypadku migracji zadań, tak jak pokazano na rys. 3.4, trzeba dodatkowo uwzględnić usunięcie zadania ze środowiska lokalnego oraz przesłanie *Koordynatorom* raportu z wykonanego zakresu wyznaczania liczb pierwszych. Wiadomość przesyłana do *Koordynatora* lokalnego zawiera bieżące informacje o wykonanym zakresie obliczeń. Na tej podstawie *Koordynator* aktualizuje pozostały do wyznaczenia zakres liczb. Zadanie po dotarciu na miejsce docelowe wysyła podobną wiadomość do nowego *Koordynatora*, tym razem jednak zawierającą zakres pozostałych liczb do przeszukania. Aby zwiększyć przydatność migracji zadań, dodano możliwość definiowania operacji wykonywanych przed uruchomieniem i po zakończonym procesie. Wykorzystuje się do tego metody: *przedMigracją*() i *poMigracji*(). Zadaniem tych metod jest zachowanie integracji w pierwszym przypadku ze środowiskiem lokalnym, w drugim ze środowiskiem zdalnym. Migracja zadań jest najbardziej powszechnym sposobem dystrybucji obliczeń spotykanym m.in. w systemach klastrowych, wieloagentowych, w systemach sterowania i zarządzania.

## 3.4.2. Replikacja zadań

Jak zaznaczono na rys. 3.5, replikacja zadań bazuje na asynchronicznej operacji klonowania zadania do nowego miejsca. Podobnie, jak w przypadku migracji, operacja ta może być zainicjowana przez *Wykonawcę*. W tym przypadku jednak poza wskazaniem docelowej lokalizacji jest wymagana nowa, niepowtarzalna nazwa dla zadania. Proces replikacji różni się od migracji tym, że po skopiowaniu danych i kodu nie następuje ich usunięcie ze środowiska lokalnego. Dzięki temu replikacja przebiega szybciej od migracji, ale może pojawić się intensywny wzrost liczby zadań, jeśli nie będzie odpowiednio kontrolowany proces. Klasycznym przykładem takiego zjawiska są infekcje wirusowe oraz wirusy komputerowe.



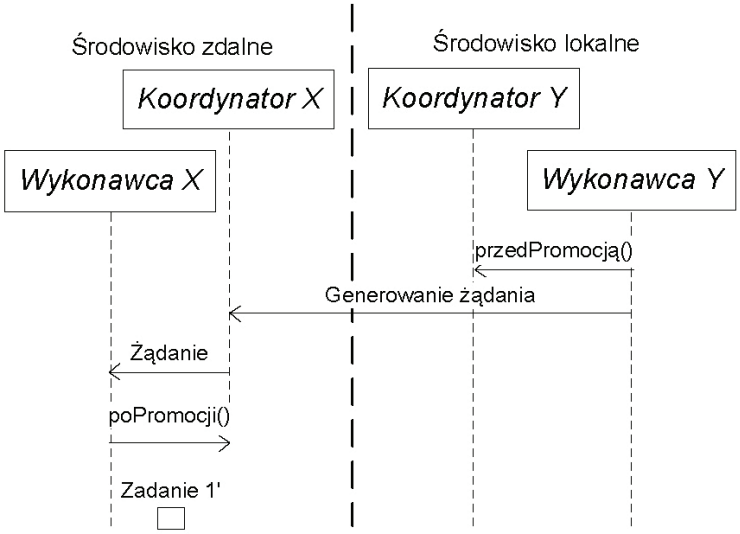

Rys. 3.5. Diagram replikacji zadań

Także w przypadku replikacji można przeprowadzić dodatkowe operacje przed rozpoczęciem i po jej zakończeniu. Służą do tego metody: *przedReplikacją*() i *poReplikacji*(). Można także usunąć obiekt z lokalizacji pierwotnej, wywołując standardową metodę *doDelete*(). Jeśli tak zostanie zrobione, uzyska się efekt końcowy taki, jaki jest po wykonaniu migracji danych.

## 3.4.3. Promocja zadań

Ostatnią formą propagacji zadań jest promocja polegająca na tworzeniu nowego obiektu w docelowej lokalizacji (rys. 3.6). W celu zachowania spójności rozwiązania,

podobnie jak w dwóch poprzednich przypadkach, mechanizm promocji może być zainicjowany przez *Wykonawcę*, który wysyła wiadomość do zdalnego *Koordynatora* z wszystkimi wymaganymi danymi do utworzenia nowego obiektu. W porównaniu z ogólnym podejściem serializacja danych nie zachodzi w środowisku lokalnym, natomiast deserializacja jest realizowana w środowisku zdalnym w postaci tworzenia zadania.

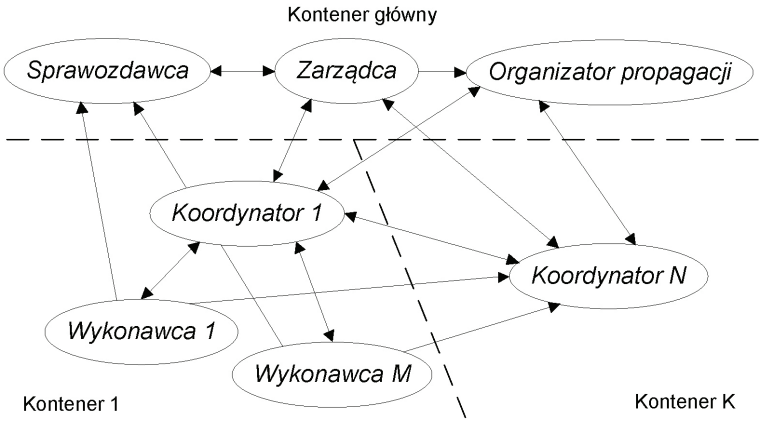



Rys. 3.6. Diagram promocji zadań

Ten typ propagacji używany jest najczęściej w pierwszym etapie przetwarzania rozproszonego, kiedy nie mamy jeszcze utworzonej infrastruktury systemu i trzeba uruchomić jak najwięcej komponentów obliczeniowych.

## 3.4.4. Komunikacja w architekturze propagacyjnej

Na rysunku 3.7 przedstawiono sposób połączeń najważniejszych ścieżek komunikacyjnych między komponentami rozproszonej architektury propagacyjnej. Ze schematu wynika, że *Wykonawcy* nie mogą bezpośrednio komunikować się z *Zarządcą* czy z *Organizatorem* propagacji. Mogą natomiast przesyłać wiadomości do *Sprawozdawcy* i *Koordynatorów*. W przypadku promocji konieczna jest możliwość komunikacji z dowolnym *Koordynatorem*. Także *Koordynatorzy* mogą się wzajemnie informować o dostępnych zasobach obliczeniowych, np. w przypadku odrzucenia proponowanej propagacji.




Rys. 3.7. Schemat komunikacji w architekturze propagacyjnej

Można zaobserwować, że architektura komunikacyjna przyjmuje postać prostej hierarchii. Na najniższym poziomie działają *Wykonawcy*, na poziomie pośrednim – *Koordynatorzy*, a na poziomie najwyższym – *Sprawozdawca*, *Zarządca* i *Organizator* propagacji.

## 3.5. Analiza symulacyjna procesu propagacji zadań

W tej części pracy zaprezentowane zostaną eksperymenty symulacyjne przeprowadzone na testowej implementacji propagacyjnej architektury rozproszonej – cele i założenia eksperymentów, uzyskane wyniki, a na koniec wnioski.

### 3.5.1. Cele i założenia eksperymentów

Zasadniczym celem przeprowadzonych eksperymentów symulacyjnych było sprawdzenie możliwości praktycznej implementacji rozproszonej architektury propagacyjnej oraz zbadanie efektywności mechanizmu propagacji zadań w heterogenicznym i dynamicznym środowisku. Heterogeniczność i dynamiczność polegała na tym, że możliwości przetwarzania w węzłach obliczeniowych mogły się różnić między sobą oraz zmieniać w czasie.

Na środowisko implementacyjne do przeprowadzenia analizy symulacyjnej wybrano platformę wieloagentową JADE 3.5[1] [12]. Za wyborem stała popularność tego

[1] http://jade.tilab.com/

narzędzia zarówno wśród rozwiązań komercyjnych, jak i prototypów akademickich. Dodatkowo za środowiskiem JADE przemawiała kompatybilność ze specyfikacją FIPA oraz dostępność na warunkach licencji LGPL. Założono, że eksperymenty będą wykonywane w dwóch konfiguracjach – heterogenicznej i homogenicznej. W ten sposób uzyskano możliwość porównania otrzymanych wyników.

W środowisku programistycznym JADE wyróżnia się dwa typy kontenerów – główny i podręczny. Pierwszy z nich pełni rolę *Zarządcy*, tymczasem drugi odpowiada *Wykonawcy*. Ponieważ drugi typ może występować wielokrotnie, można implementować węzły obliczeniowe z kilkoma *Wykonawcami*.

Dodatkowo przyjęto następujące założenia:

- Liczba pakietów wysłanych jest równa liczbie pakietów odebranych, czyli wszystkie pakiety bez żadnych zmian docierają do adresata.
- W systemie nie występują nagłe i niekontrolowane zmiany parametrów transferu danych.
- Na początku nieznana jest charakterystyka poszczególnych węzłów obliczeniowych. Przyjmuje się jedynie, że system potrafi rozwiązać zadany problem.

Podstawowym wskaźnikiem wydajności systemu testowego z propagacją obliczeń jest zysk czasowy, który zostanie osiągnięty podczas wyznaczania liczb pierwszych. Wynikowa charakterystyka czasowa środowiska symulacyjnego przekłada się w bezpośredni sposób na wartości parametrów przekazywanych do uruchamianych procedur testowych. Ponieważ parametrów jest wiele i obejmują one szerokie zakresy wartości, w eksperymentach skoncentrowano się na przypadkach istotnych dla procesu propagacji zadań – z tego względu do kluczowych parametrów należą:

- *LZW* – liczba zadań w węźle obliczeniowym. Najpierw tworzony jest *Koordynator*, następnie inicjowane są zadania. Jeśli nie jest specyfikowany numer węzła, parametr *LZW* dotyczy wszystkich węzłów. Początkowa wartość *LZW* to 10.
- *RP* – rodzaj propagacji. Parametr ten jest przekazywany przez *Koordynatora* podczas tworzenia zadania. *Organizator* propagacji decyduje o lokalizacji docelowej zadania, ale to *Wykonawca* stanowi o rodzaju propagacji. Parametr może przyjąć wartości: 1, 2 lub 3 oznaczające odpowiednio migrację, replikację i promocję. Domyślna wartość to 3, czyli promocja.
- *JRZL* – jednostkowy rozmiar zakresu liczb. Jeden z najważniejszych parametrów określający liczbę sprawdzanych liczb w jednym przebiegu obliczeń między raportami. Po tym, jak zadanie wyznaczy liczby pierwsze dla *JRZL*, *Wykonawca* sprawdza, czy jakieś wiadomości nie czekają w kolejce na obsługę, np. zlecenie propagacji. Dodatkowo zauważyć trzeba, że *JRZL* określa częstotliwość wysyłania raportów do *Sprawozdawcy*. W eksperymentach jednostkowy rozmiar zakresu liczb równa się 150.
- $t_o$ – minimalny czas oczekiwania między raportami. Jest podawany w sekundach i determinuje okres, jaki musi upłynąć od wysłania ostatniego raportu

przez *Koordynatorów*, aby można było przesłać ponowne żądanie raportowania przez *Organizatora* propagacji. W eksperymentach czas oczekiwania ustalono na 20 s.

- $t_{pow}$ – próg opłacalności węzła obliczeniowego. Kolejny parametr ograniczający zbędne i kosztowne propagacje. *Koordynatorzy* przesyłają *Organizatorowi* propagacji informacje o prognozowanym czasie zakończenia obliczeń i liczbie uruchomionych zadań. Jeśli prognozowany czas pozostałych obliczeń jest mniejszy od progu opłacalności, to taki węzeł nie jest brany pod uwagę przy planowaniu propagacji. Dzięki temu nie są wykonywane propagacje, które praktycznie nie powodują przyspieszenia, a jedynie bezużyteczny ruch w sieci. Na dodatek, jeżeli zachodzi taka potrzeba, możemy wykorzystać ten parametr do wyłączenia pojedynczego węzła z procesu propagacyjnego. Domyślna wartość progu opłacalności to 15 s.

## 3.5.2. Opis przeprowadzonych eksperymentów

Eksperymenty przeprowadzono w środowisku heterogenicznym i homogenicznym. Pierwsze środowisko składało się z 4 różnych komputerów typu PC: D1–D4, z kolei drugie środowisko tworzyło 46 jednakowych komputerów: A1–A46 zainstalowanych w laboratorium studenckim Wydziału Informatyki i Zarządzania Politechniki Wrocławskiej. W obu przypadkach komputery działy w sieci Ethernet o przepustowości do 100 Mb/s. Konfigurację używanych komputerów umieszczono w tab. 3.1.

Tabela 3.1. Konfiguracja komputerów używanych w eksperymentach

| Komputer | Opis konfiguracji |
|---|---|
| D1 | Intel Core Duo 1.86 GHz, 2GB RAM, Windows Vista Business SP1 |
| D2 | Intel Pentium 1.86 GHz, 448 MB RAM, Windows XP Professional SP2 |
| D3 | AMD Athlon XP 3200+ 2.2 GHz, 1 GB RAM, Windows XP Professional SP2 |
| D4 | AMD Athlon XP 1700+ 1.44 GHz, 256 RAM, Windows XP Professional SP2 |
| A1–A46 | Intel Pentium 2,81 GHz, 448 MB RAM, Windows XP Professional SP2 |

Wykonano 5 eksperymentów dotyczących: (A) wydajności poszczególnych węzłów, (B) strategii ekstensywnej i intensywnej, (C) wydajności propagacyjnej architektury rozproszonej, (D) analizy ilościowej mechanizmu propagacji i (E) analizy jakościowej mechanizmu propagacji. Wszystkie eksperymenty dotyczyły wyznaczania liczb pierwszych z wykorzystaniem mechanizmu propagacji zadań. Wyniki eksperymentów zostaną przedstawione w formie wykresów lub tabel. Na opis eksperymentu składają się: cele eksperymentu, wartości parametrów (o ile różnią się od wartości domyślnych) oraz wyniki z krótkim komentarzem.

### A. Wydajność węzłów obliczeniowych

**Cel**: Zbadanie wydajności pojedynczych węzłów jako samodzielnych jednostek obliczeniowych przez sprawdzenie zależności między liczbą zadań w węźle *LZW*, typem procesora i czasem wykonania.

**Parametry**: *LZW* = 20, *JRZL* = 1000, *ZL* = [1 mld, 1,003 mld].

Tabela 3.2. Czas wykonania uzyskany przez węzły w funkcji liczby zadań [s]

| *LZW* / Węzeł | 2 | 5 | 10 | 20 | 50 | 100 | 200 | 500 | 1000 |
|---|---|---|---|---|---|---|---|---|---|
| D1 | 92,3 | 93,9 | 93,9 | 93,6 | 94,1 | 93,9 | 93,8 | 95,1 | 95,6 |
| D2 | 185 | 185,8 | 184,9 | 185,1 | 185,3 | 187,2 | 188,5 | 193,2 | 197,5 |
| D3 | 328,7 | 328,1 | 325,1 | 321,9 | 318,8 | 318,4 | 318,6 | 324,9 | 329,6 |
| D4 | 478,9 | 481,1 | 481,1 | 481,4 | 482,1 | 482,5 | 483 | 493,3 | 511,6 |
| A1–A46 | 258,5 | 258,3 | 256,3 | 254,2 | 252,8 | 252,1 | 252,3 | 254,8 | 259,2 |

**Wyniki**: W tabeli 3.2 przedstawiono czas wykonania dla komputerów D1–D4 oraz A1–A46. Każdy test został powtórzony 3 razy. Standardowe odchylenie wahało się od 0,1 do 0,5 s. Ze względu na typ procesora globalne minimum zostało osiągnięte przez komputer D1 dla 2 zadań. Uzyskane wyniki dla pozostałych przypadków różniły się najczęściej o 1%, jednak dla konfiguracji między 5 a 200 zadaniem czas wykonania był najkrótszy. Jeśli porównuje się te wartości względem szybkości procesorów, to można zauważyć, że zależność czasowo-procesorowa została zachowana, np. D4/D3 = 478/328 (czas wykonania) i D3/D4 = 2,2/1,44 (szybkość procesorów).

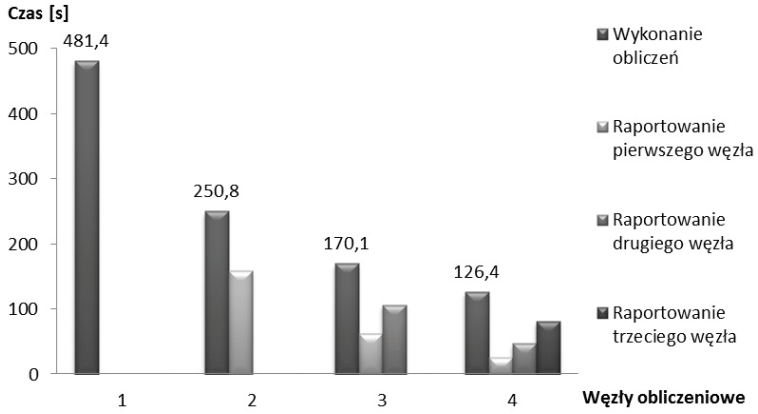
**Komentarz**: Czas wykonania był ściśle zależny od konfiguracji węzła, na którym zadanie zostało uruchomione. W kolejnych eksperymentach za wielkość początkową przyjęto liczbę 20 zadań przetwarzanych w jednym węźle.

### B. Strategia ekstensywna i intensywna

**Cel**: Zbadanie wpływu wyboru strategii ekstensywnej *SE* i intensywnej *SI* oraz mechanizmu propagacji na czas wykonania obliczeń $t_w$. Sprawdzenie zależności między czasem wykonania i czasem raportowania $t_{pr}$, $\overline{t_r}$.

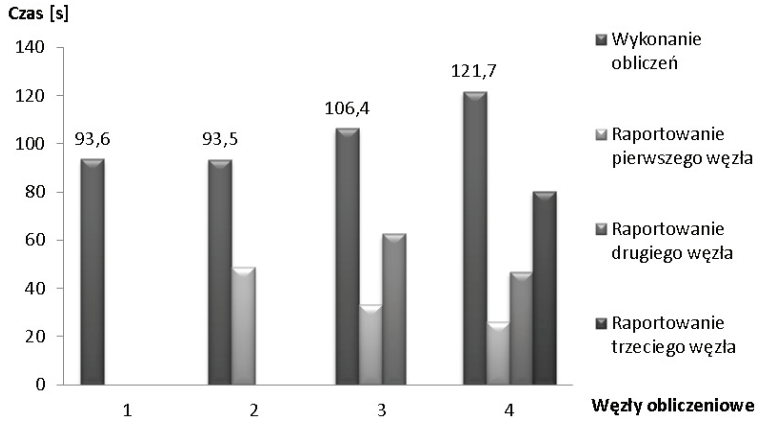
**Parametry**: *RP* = 1, *LZW* = 20, *JRZL* = 150, *TR* = 5, $t_{pow}$ = 15, *ZL* = [1 mld, 1,003 mld].

**Wyniki**: Strategia ekstensywna *SE* oznacza, że kolejność wykonywania zadania rozpoczyna się od najmniej wydajnego węzła, poprzez uaktywnianie kolejnych węzłów aż do najbardziej wydajnego, w rozpatrywanym przypadku będzie to sekwencja od D4 do D1. Strategia intensywna *SI* odwraca natomiast tę kolejność wykonywania zadania, wtedy zadanie jest wykonywane od D1 do D4.



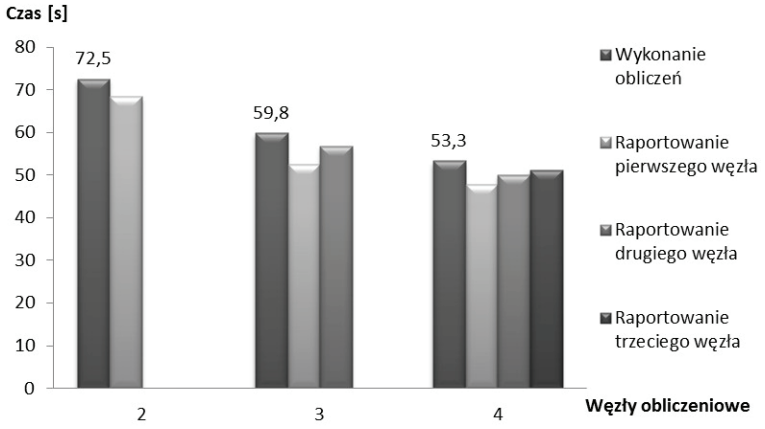

Rys. 3.8. Czas wykonania i raportowania w strategii ekstensywnej bez propagacji

Na rysunku 3.8 pokazano, jak zmienia się czas wykonania po uruchamianiu kolejnych węzłów obliczeniowych zgodnie ze strategią ekstensywną. Ponieważ momenty raportowania są zasadniczo różne, co wskazuje na różną wydajność węzłów, można się spodziewać, że wprowadzenie propagacji pozwoli lepiej wykorzystać dostępne zasoby.



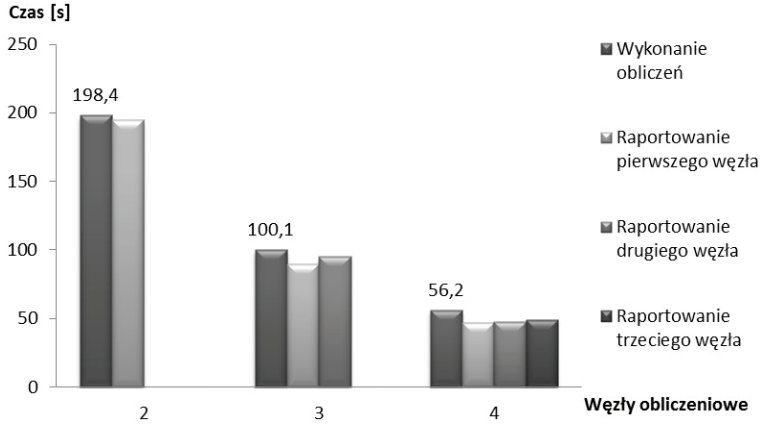

Rys. 3.9. Czas wykonania i raportowania w strategii intensywnej bez propagacji

Na rysunku 3.9 czas wykonania w przypadku strategii intensywnej wzrastał wraz z uruchamianiem kolejnych węzłów. Pierwszy – najbardziej wydajny węzeł wykonywał zadanie szybciej niż cztery współpracujące ze sobą węzły. Narzut komunikacyjny w tym przypadku jest bardzo kosztowny. Jeśli porówna się czas wykonania w obu strategiach, widać nieznaczną różnicę, która powstała m.in. z powodu wykonywania operacji dyskowych na węzłach znacznie różniących się wydajnością.




Rys. 3.10. Czas wykonania i raportowania w strategii intensywnej z propagacją

W przypadku strategii intensywnej z propagacją (rys. 3.10) widać stopniowe zmniejszanie czasu wykonania, z początkowych 72 s do 53 s. Dodanie mechanizmu propagacyjnego było opłacalne. Zysk wielkości 56% jest bardzo dobrym rezultatem. Uzyskana różnica między czasem wykonania a czasem raportowania wyniosła poniżej 20%.




Rys. 3.11. Czas wykonania i raportowania w strategii ekstensywnej z propagacją

W przypadku strategii ekstensywnej (rys. 3.11) zysk także jest duży i dobrze widoczny na każdym etapie eksperymentu, przy czym różnice czasów raportowania są minimalne.

**Komentarz**: Czas wykonania zadania jest jednoznacznie determinowany przez najsłabszy węzeł w sieci. W sieciach heterogenicznych wykorzystanie mechanizmu

propagacji może przynieść znaczący wzrost wydajności systemu. Operacje związane z zarządzaniem zadań powinny być wykonywane w najbardziej wydajnych węzłach.

### C. Wydajność propagacyjnej architektury rozproszonej

**Cel**: Zbadanie wpływu liczby węzłów obliczeniowych *LW* na wydajność systemu. Sprawdzenie zależności liczby komunikatów *LK* i raportów *LR* od liczby węzłów.

**Parametry**: *LZW* = 20, *JRZL* = 1000, *ZL* = [1 mld, 1,004 mld].

Tabela 3.3. Czas wykonania i raportowania, komunikaty i raporty w funkcji liczby węzłów

| *LW* | $t_w$ [s] | $t_{pr}$ [s] | *LK* | *LR* |
|---|---|---|---|---|
| 1 | 338,3 | 338,3 | 47 | 4000 |
| 3 | 113,5 | 112,4 | 133 | 4020 |
| 5 | 68,3 | 67,4 | 219 | 4000 |
| 20 | 17,6 | 16,9 | 864 | 4000 |
| 46 | 7,6 | 7,4 | 1982 | 4140 |

**Wyniki**: Uzyskane wyniki przedstawiono w tab. 3.3. Ponieważ jednostkowy rozmiar zakresu liczb *JRZL* był równy 1000 i zakres wyszukiwanych liczb *ZL* obejmował 4 mln liczb, wygenerowano 4 tys. raportów dla *Sprawozdawcy*. Tyle samo razy sprawdzono, czy nie pojawił się w kolejce komunikat, który należałoby obsłużyć. Zmniejszenie wielkości *JRZL* powodowało zwiększenie liczby generowanych raportów, co miało negatywny wpływ na sumaryczny czas wykonywania obliczeń. W środowisku węzłów A1–A46 mimo tego, że jednostki miały taką samą konfigurację, w rezultacie działały z różną efektywnością. Czas, po którym uzyskiwano pierwszy raport z wynikami, niewiele różnił się od czasu wykonania całego zadania. Liczba wysyłanych komunikatów rosła proporcjonalnie do liczby działających węzłów. Mimo zwiększania liczby węzłów liczba raportów dla *Sprawozdawcy* praktycznie pozostawała na tym samym poziomie.

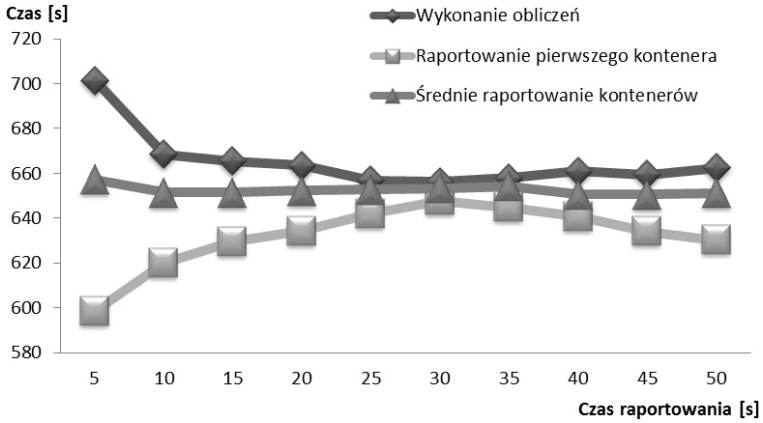
**Komentarz**: W warunkach eksperymentu wszystkie zaplanowane operacje zostały wykonane prawidłowo. Zmiana parametrów powodowała uzyskiwanie gorszych rezultatów. Można więc przyjąć, że wartości początkowe zostały dobrane prawidłowo.

### D. Analiza ilościowa mechanizmu propagacji

**Cel**: Znalezienie efektywnie przetwarzanej liczby zadań w węźle obliczeniowym *LZW*. Zbadanie wpływu liczby zadań na liczbę propagacji *LP* oraz rodzaju propagacji *RP* na czas wykonywania obliczeń $t_w$.

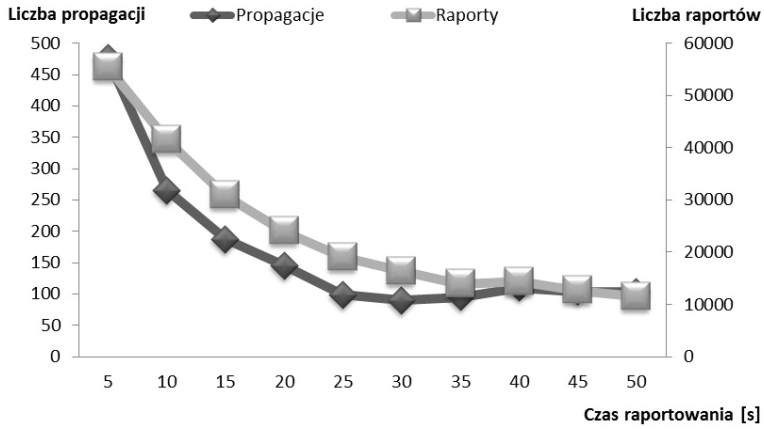
**Parametry**: *LZW* = 10, *JRZL* = 100, $t_{pcw}$ = 15 s, $t_r$ = 30 s, *RP* = 3, *ZL* = [1 mld, 1,11 mld].

**Wyniki**: W eksperymencie wykorzystano 15 węzłów, na których zainicjowano 30 kontenerów ze środowiska *JADE*. Na 5 węzłach po 1 kontenerze, na kolejnych 5 po 2 kontenery, a na ostatnich po 3 kontenery. To spowodowało zróżnicowanie mocy obliczeniowej węzłów. Bez wykorzystania mechanizmu propagacji wykonanie obliczeń zajęło 1000 s, przy czym pierwszy raport wynikowy pojawił się już po 317 s.




Rys. 3.12. Czas wykonania i raportowania w funkcji czasu oczekiwania

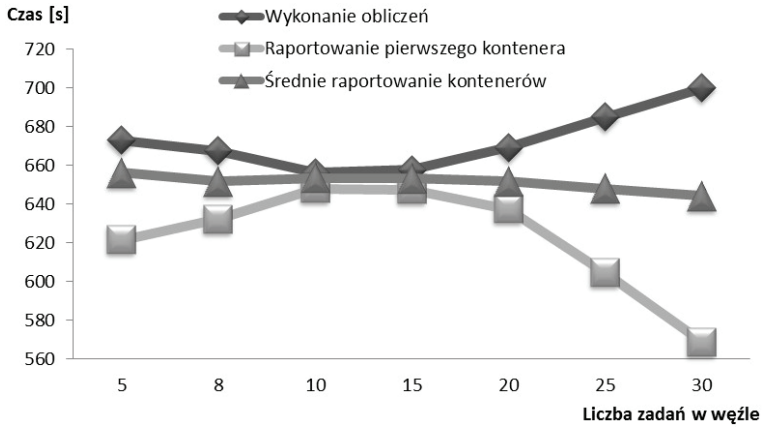
Na rysunku 3.12 pokazano, że jeśli czas oczekiwania jest niewielki, to czas wykonania eksperymentu jest długi przy jednocześnie szybkim pierwszym raportowaniu. Jeżeli wartość czasu oczekiwania oscyluje między 25–35 s, czas wykonania zmniejsza się.




Rys. 3.13. Liczba propagacji i raportów w funkcji czasu oczekiwania

Średni czas raportowania praktycznie nie zmienia się. Najlepsza uzyskana wartość czasu oczekiwania w odniesieniu do czasu wykonania to 30 s, co stanowi prawie 5% całkowitego czasu wykonania. Zmniejszanie czasu oczekiwania poniżej tej wartości powoduje zaktywizowanie środowiska, m.in. szybciej generowany jest pierwszy raport, odbywa się to jednak kosztem czasu wykonania.

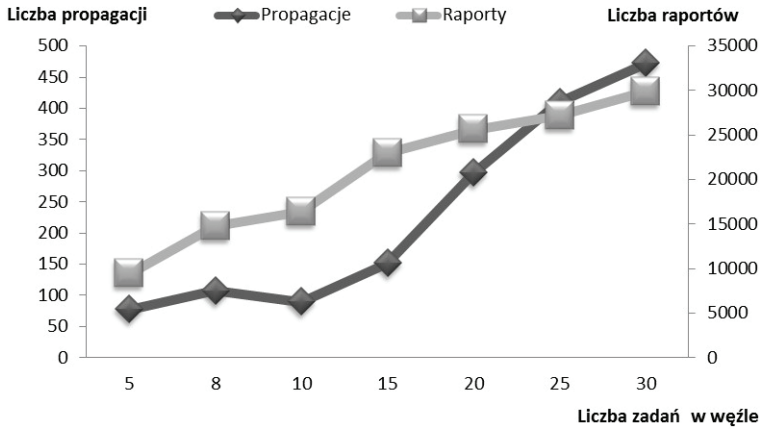
Na rysunku 3.13 wyznaczono zależność między liczbą propagacji i raportów a czasem oczekiwania. Jeśli czas oczekiwania rośnie, to liczba raportów tym samym maleje. Tak jak poprzednio, czas oczekiwania w przedziale 30–35 s pełni ważną rolę. Po jego przekroczeniu liczba raportów praktycznie stabilizuje się na tym samym poziomie.



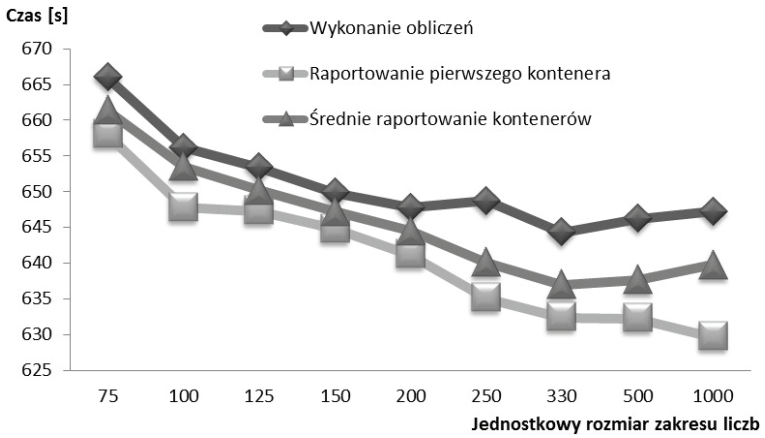

Rys. 3.14. Czas wykonania i raportowania w funkcji liczby zadań

Na rysunku 3.14 zobrazowano dynamikę mechanizmu propagacji. Czas wykonania dąży do minimum przy 10 zadaniach, podczas gdy linia pierwszego raportowania zachowuje się tak, jakby była lustrzanym odbiciem czasu wykonania. Średni czas raportowania przez węzeł jest praktycznie stały. Po przekroczeniu bariery 15 zadań można zauważyć lekki trend szybszego raportowania kosztem czasu wykonania. Wyjaśnienie tych zjawisk jest następujące: kiedy mamy do dyspozycji jedynie kilka węzłów, nie są one w stanie w pełni niwelować dysproporcji obliczeniowych, więc częste propagacje powodują opóźnienie. Podobnie dzieje się, gdy mamy za dużo węzłów, wówczas narzut komunikacyjny powoduje automatyczne wydłużenie czasu szukania rozwiązania.

Liczba propagacji (rys. 3.15) w zakresie od 5 do 10 zadań w węźle zmienia się nieznacznie, dopiero po przekroczeniu wartości 10 szybko rośnie. Ponadto zależności liczby propagacji i liczby raportów od liczby zadań są podobne i bliskie liniowym.



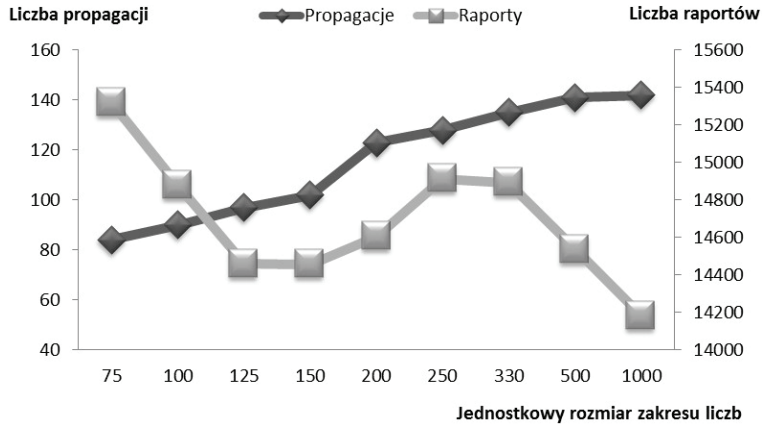

Rys. 3.15. Liczba propagacji i raportów w funkcji liczby zadań



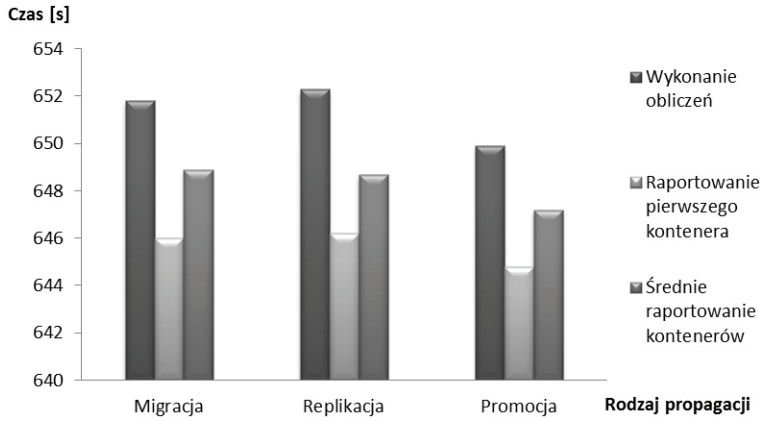

Rys. 3.16. Czas wykonania i raportowania w funkcji *JRZL*

Badając zależność czasu wykonania od wielkości jednostkowego rozmiaru zakresu liczb, można zauważyć (rys. 3.16), że im mniejsza wartość jednostkowego rozmiaru liczb *JRZL*, tym większy czas obliczeń. Zmienia się jednak także odległość między liniami czasu. Dla mniejszych wartości *JRZL* odległości są coraz mniejsze, natomiast dla większych *JRZL* zachodzi sytuacja odwrotna. Ponadto można zauważyć stabilizację linii czasu między 645–650 s, co jest związane z większym kosztem komunikacyjnym przy zmniejszającym się zakresie liczb.

Na rysunku 3.17 pokazano brak liniowej korelacji między liczbą propagacji a liczbą raportów w zależności od jednostkowego rozmiaru zakresu liczb. Widać natomiast, że przy zmniejszeniu *JRZL* dziesięciokrotnie – liczba propagacji zmalała jedynie dwukrotnie.




Rys. 3.17. Liczba propagacji i raportów w funkcji *JRZL*




Rys. 3.18. Czas wykonania i raportowania w funkcji rodzaju propagacji

Przy porównaniu wartości uzyskanych czasów wykonania dla różnego rodzaju propagacji (rys. 3.18) widać, że różnice oscylują na poziomie 1%. Oznacza to, że rodzaj propagacji nie ma wielkiego wpływu na uzyskiwane rezultaty przetwarzania. Dobór innych parametrów, np. *JRZL* czy *LZW*, ma dużo większe znaczenie.

**Komentarz**: Czas oczekiwania 25–35 s pozwolił uzyskać najlepsze wyniki w stosunku do czasu wykonania. Optymalna liczba zadań działających w węźle obliczeniowym w testowanym środowisku wynosiła od 10 do 15. W tym zakresie linie czasu wykonania, pierwszego raportowania i średniego raportowania maksymalnie zbliżają się do siebie. To oznacza, że różnice powodowane odmienną konfiguracją czy obciążeniem zanikają skutecznie w architekturze wykorzystującej mechanizm propagacyjny. Dodatkowo można zaobserwować, że zmniejszenie liczby zadań po-

woduje jednoczesne zmniejszenie liczby propagacji. *JRZL* powinien wynosić co najmniej 150, natomiast rodzaj propagacji nie ma wielkiego wpływu na czas wykonywanych obliczeń.

### E. Analiza jakościowa mechanizmu propagacji

**Cel**: Zbadanie zachowania procesu propagacji zadań na podstawie analizy raportów generowanych przez *Zarządcę*.

**Parametry**: $LZW = 10$ lub 20, $JRZL = 150$, $t_{pcw} = 20$ s, $t_o = 15$ s, $RP = 3$, $ZL = [1$ mld, 1, 025 mld].

**Wyniki**: Eksperyment podzielono na 2 etapy: dla 10 zadań i dla 20 zadań.

Tabela 3.4. Częściowy wykaz propagacji dla $LZW = 10$

| $t_z$ [ms] | *Zdarzenie* | *Początek* | *Koniec* | *LP* |
|---|---|---|---|---|
| 20 032 | Raport nr 1 – D1:10, D2:10, D3:10, D4:10 | | | |
| 22 864 | Propagacja | D4 | D1 | 7 |
| | Propagacja | D3 | D1 | 4 |
| | Propagacja | D3 | D2 | 1 |
| 45 875 | Raport nr 2 – D1:21, D2:11, D3:5, D4:3 | | | |
| 48 975 | Propagacja | D1 | D3 | 1 |
| | Propagacja | D1 | D4 | 1 |
| 70 368 | Raport nr 3 – D1:19, D2:11, D3:6, D4:4 | | | |
| 94 052 | Raport nr 4 – D1:19, D2:11, D3:6, D4:4 | | | |
| 115 817 | Raport nr 5 – D1:19, D2:11, D3:6, D4:4 | | | |
| 231 674 | Propagacja | D2 | D1 | 1 |
| 255 185 | Propagacja | D1 | D2 | 1 |

Podczas analizy wyników przedstawionych w tab. 3.4 nasuwa się wniosek, że system dąży do stabilnego układu zadań i zdecydowana większość procesów propagacyjnych (14 z 22) zachodzi głównie na początku, kiedy dostosowywane są parametry systemu do możliwości obliczeniowych węzłów. Stan stabilności uzyskano dla następującego układu zadań: D1:19, D2:11, D3:6 i D4:4. Struktura stanu stabilności wynika z różnicy szybkości procesorów, np. D3/D4 = 2,2/1,44 jest porównywalne z 6/4 (liczby zadań dla stanu stabilnego). Na późniejszym etapie możemy trafić na efekt cyklu propagacyjnego polegającego na odwróceniu procesu, np. propagacje po wygenerowaniu raportu nr 5. Tego rodzaju zmianom należy przeciwdziałać, na przykład zwiększając próg opłacalności węzła. Dalsze zmiany polegały już tylko na korekcie układu w postaci pojedynczych propagacji w reakcji na zakończenie obliczeń przez *Wykonawcę*. Całkowita liczba propagacji ostatecznie ustaliła się na 22. Średni czas wykonania zadania przez węzły to prawie 426 s, przy czym ostatni zakończył obliczenia po 429 s.

Tabela 3.5. Częściowy wykaz propagacji dla *LZW* = 20

| $t_z$ [ms] | *Zdarzenie* | *Początek* | *Koniec* | *LP* |
|---|---|---|---|---|
| 20 062 | Raport nr 1 – D1:20, D2:20, D3:20, D4:20 | | | |
| 30 231 | Propagacja | D4 | D1 | 14 |
| | Propagacja | D3 | D1 | 9 |
| 73 051 | Propagacja | D1 | D2 | 4 |
| 102 857 | Propagacja | D2 | D3 | 1 |
| | Propagacja | D2 | D1 | 1 |
| 135 061 | Propagacja | D1 | D4 | 3 |
| | Propagacja | D2 | D3 | 1 |
| 398 865 | Raport nr 14 – D1:18, D2:20, D3:12, D4:9 | | | |
| 404 780 | Propagacja | D4 | D1 | 1 |
| | Propagacja | D2 | D1 | 1 |

W tabeli 3.5 zawarto wyniki z drugiego etapu. Dla 20 zadań liczba propagacji zwiększyła się z 22 do 49. Podobnie jak poprzednio, większość propagacji (27) pojawiła się już na samym początku. Całkowity czas wykonania zadania był prawie taki sam, wystąpiła różnica 1 s, natomiast średni czas wykonania przez węzeł zwiększył się aż o prawie 3 s, co oznacza, że propagacje są szczególnie kosztowne dla mniej wydajnych komputerów. Dodatkowo wprowadziliśmy ograniczenie propagacji do jednego zadania po przeprowadzeniu kilku początkowych propagacji. Efekt był jednak dalej negatywny, gdyż spowodowało to usztywnienie systemu i brak możliwości szybkiego wyrównywania obciążeń węzłów.

**Komentarz**: Niewielka początkowa liczba *Wykonawców* dobrze wpływa na proces propagacji. Po początkowym okresie dużej aktywności w miarę upływu czasu liczby zadań na węzłach stabilizują się i dopiero w momencie kończenia obliczeń ponownie zmieniają się. Wtedy mogą pojawić się nowe propagacje, także o charakterze negatywnym. W ten sposób liczby aktywnych *Wykonawców* odzwierciedlają bieżący stan rozwiązywania problemu. Wprowadzenie ograniczenia na liczbę propagacji przeprowadzanych jednocześnie eliminuje nadmierną komunikację między węzłami, ale może powodować niepełne wykorzystanie zasobów obliczeniowych przez opóźnione bilansowanie różnic. Redukcja cykli propagacyjnych powinna być podstawowym zadaniem w pracach nad dalszą optymalizacją tego mechanizmu.

## 3.5.3. Wnioski z eksperymentów

Szczegółowe wnioski płynące z eksperymentów w postaci komentarza formułowano na bieżąco. Teraz podsumowane zostaną najważniejsze z nich.

Tabela 3.6. Najlepsze wartości konfiguracyjne uzyskane dla obu środowisk testowych

| *Parametr* | *Środowisko heterogeniczne* | *Środowisko homogeniczne* |
|---|---|---|
| $p_w$ | 58% | 35,4% |
| *JRZL* | 150 | 150–200 |
| $t_o$ | 20 s | 30 s |
| *LZW* | 5–20 | 10–15 |

Na podstawie analizy danych zawartych w tab. 3.6 można wnioskować, że uzyskane przyspieszenie wykonania w obu środowiskach testowych potwierdza hipotezę o zysku, jaki może wnieść mechanizm propagacji przy wykonywaniu rozproszonych obliczeń. Nie można jednak pominąć kosztu, gdyż może się okazać, że zamiast spodziewanego zysku poniesiona zostanie strata.

Większa liczba zadań powoduje zwielokrotnienie liczby propagacji, a tym samym większą aktywność środowiska. Ważnym elementem stają się wtedy ograniczenia transmisyjne, które mogą negatywnie wpływać na wykonywanie zadań. Zasadniczym parametrem sterującym w przypadku problemu rozproszonego wyznaczania liczb pierwszych jest *JRZL* bilansujący efektywność obliczeń i szybkość transmisji. Optymalnym momentem równowagi jest uzyskanie podobnego czasu zakończenia obliczeń przez wszystkie węzły przy równoczesnej minimalizacji kosztów transmisji. Wtedy zwiększa się stabilność całego systemu. Poszukiwana liczba zadań w węźle w rozpatrywanym przypadku – od 5 do 20 – ma umożliwić wyrównanie różnic wydajnościowych między heterogenicznymi węzłami obliczeniowymi. Dla środowiska homogenicznego liczba zadań w węźle oscyluje między 10 i 15. Wybór rodzaju propagacji nie jest tak ważny, jak liczba wykonywanych zadań. Najefektywniejszą z nich okazała się promocja pozbawiona kosztu związanego z serializacją zadania w środowisku lokalnym.

Przy propagacji zadań może pojawić się negatywne zjawisko, a mianowicie cykl propagacyjny. Najczęściej zachodzi on między podobnymi węzłami i polega na odwróceniu procesu propagacyjnego i powrocie do konfiguracji zadań sprzed zmiany. W ten sposób generowany jest tylko koszt wykonanych operacji przy znikomym lub żadnym zysku przetwarzania zadania. Aby ograniczyć tę osobliwość, wprowadzono próg opłacalności oraz strategię limitowania propagacji. Dodatkowo można wprowadzić ograniczenia przy tworzeniu zadań na węźle. Szczególnie niebezpiecznym momentem dla propagacji zadań jest uszkodzenie węzła obliczeniowego. Utrzymanie poziomu wydajności sprzed tego rodzaju awarii wprowadza konieczność na przykład uruchomienia masowej propagacji wśród pozostałych węzłów, co tym samym pociąga zwiększenie prawdopodobieństwa wygenerowania cyklu.

Biorąc pod uwagę otrzymane z eksperymentów wyniki, można stwierdzić, że podstawowym problemem jest znalezienie równowagi między wyrównywaniem mocy węzłów obliczeniowych, przez zwiększanie liczby wykonywanych zadań, a ograniczaniem propagacji i możliwości powstawania cykli przez zmniejszanie tej samej

liczby zadań. W takiej sytuacji najprostszym nasuwającym się rozwiązaniem jest selektywne dodawanie i usuwanie zadań w węzłach z równoczesną obserwacją liczby wykonywanych propagacji.

Należy podkreślić, że w rzeczywistości przeprowadzono znacznie więcej eksperymentów niż wcześniej opisane, m.in. dla większej liczby węzłów obliczeniowych i większej granulacji problemu. Przedstawionych tu sześć eksperymentów wybrano jako najbardziej reprezentatywne w przypadku zagadnień będących przedmiotem prezentowanej pracy.

## 3.6. Podsumowanie

W rozdziale 3 został przeanalizowany mechanizm propagacyjny pozwalający na uzyskanie większej mocy obliczeniowej w architekturze systemu rozproszonego. Pokazano, że wskazane rozwiązania, bazujące na promocji zadań przy jednoczesnym wyrównywaniu różnic wydajnościowych oraz ograniczaniu cyklu propagacyjnego, przynoszą najlepsze rezultaty w stosunku do ponoszonych kosztów operacji.

Jeden z kierunków dalszych badań może prowadzić do dokładnej analizy historii propagacji w systemie, o ile zmiany nie zachodzą zbyt często. Inny – do zagadnienia dekompozycji zadań złożonych z uwzględnieniem mechanizmów propagacyjnych. Zadanie wyznaczania liczb pierwszych nie wymaga dużego nakładu na jego dekompozycję. To ograniczyło potencjalny, negatywny wpływ dekompozycji na działanie systemu. Rozpatrując inny problem, należy mieć świadomość, że na przykład koszt transmisji może być na tyle duży, iż wprowadzenie nawet pojedynczej propagacji może powodować starty.

Najważniejsze wyniki przedstawione w rozdz. 3 zostały opublikowane w następujących pracach: [77], [81], [84], [85], [88].

# 4. Propagacja błędów

W rozdziale omówiono problem propagacji błędów w systemie rozproszonym. Zbadanie tego zagadnienia ma zasadnicze znaczenie w zapewnieniu maksymalnego bezpieczeństwa działania sieciowego systemu informacyjnego. Zaproponowano ilościowe i jakościowe metody analizy możliwych scenariuszy propagacyjnych dla różnych konstrukcji sieciowych, typów błędów i rozkładów prawdopodobieństw. Zweryfikowane statystycznie wyniki badań pozwalają na wybór najbardziej odpornej na błędy architektury rozproszonej oraz na określenie najmniej szkodliwych typów propagowanych błędów występujących w systemach sieciowych. Znajomość charakterystyki działania tego mechanizmu propagacyjnego pomaga m.in. projektować bardziej niezawodne systemy komputerowe, implementować bezpieczne protokoły sieciowe oraz lepiej oceniać ryzyko w zakresie zagrożeń dotyczących już wdrożonych rozwiązań.

## 4.1. Wprowadzenie

Rozwój technik internetowych przyniósł wiele nowych możliwości wzajemnego, swobodnego komunikowania się. Jednocześnie brak kontroli nad zawartością publikowanych informacji, w tym anonimowość użytkowników, pozwala na niczym nieograniczone zarażanie urządzeń i programów używanych w sieci. Istniejące zabezpieczenia nie są w stanie całkowicie chronić ani ogromnych zasobów komputerowych, ani urządzeń mobilnych czy konsol gier internetowych. Proponowane zasady ochrony przed szkodliwym oprogramowaniem najczęściej bazują na monitorowaniu przesyłanych w sieci danych. Do takiego oprogramowania zaliczamy m.in. wirusy, robaki, trojany i programy szpiegujące. Niestety, nawet najnowsze metody ochrony antywirusowej nie dają pełnej gwarancji bezpieczeństwa. Dlaczego infekcje komputerowe są tak skuteczne? Wśród kilku przyczyn jedna jest bardzo czytelna. Podstawowym sposobem rozpowszechniania złośliwego oprogramowania jest prosty, ale za to efektywny mechanizm propagacyjny. W dalszej części pracy nie zajęto się jednak propagacją wirusów, ale problemem ogólniejszym i bardziej złożonym, znanym w literaturze pod hasłem propagacji błędów lub uszkodzeń.

Propagacja błędów to proces, w wyniku którego błąd występujący w jednym komponencie systemu przenosi się na inne komponenty. Badanie tego procesu ma zasadnicze znaczenie w zapewnieniu poprawnego działania w każdym rozproszonym systemie informacyjnym, nie tylko ze względu na możliwą obecność szkodliwego oprogramowania.

Zjawisko propagacji błędów jest znane od lat. Ograniczanie rozprzestrzeniania się nieprawidłowego zachowania części czy nawet całych systemów leży m.in. w zakresie teorii niezawodności systemów oraz projektowania poprawnych i bezpiecznych algorytmów. Mimo istotności tego problemu fenomen mechanizmu propagacji w funkcji kompozycji typu, miejsca i momentu występowania błędu nie był dotąd przedmiotem wnikliwych badań. W przypadku tworzenia nowego systemu wiedza o miejscu występowania i charakterze hipotetycznej awarii może pomóc odpowiednio go zaprojektować, a następnie zaimplementować. Dla działającego systemu badanie zmian propagacyjnych także ma duże znaczenie, gdyż może wskazać komponenty najbardziej narażone na uszkodzenia. Pomimo szybkiego rozwoju sieci złożonych [23] oraz systemów wieloagentowych [116] nie istnieje jedna, uniwersalna metoda określająca skuteczność mechanizmu propagacji. Dotychczasowe prace najczęściej skupiały się na strukturalnych aspektach komunikacji w sieciach komputerowych [62], [88], [156], biorąc pod uwagę przede wszystkim złożoność przesyłanych wiadomości [57].

Istnieją dwa podstawowe kierunki badań dotyczących propagacji uszkodzeń. Pierwszy dotyczy metod uodparniania oprogramowania na ukryte wady. Do najczęściej stosowanych zalicza się: obsługę wyjątków, refaktoryzację i wstrzykiwanie kodu, ponowne uruchomienie procedury z wartościami domyślnymi, powielanie procesów i obsługę punktów kontrolnych. Na przykład Thain i Livn zdefiniowali mechanizm obsługi wyjątków w języku Java pozwalającym na kooperację procesów w celu podjęcia właściwej reakcji na wykryty błąd bez szczegółowej analizy jego charakteru i zachowania [139]. Podobną kwestią jest analiza ścieżek generowanych wyjątków w poszukiwaniu przyczyn ich występowania, np. określanie źródła błędu [26]. Problemem modelowania intruza i propagacyjnych schematów ataku zajmowano się w pracy [115].

Drugi kierunek badań dotyczący zachowania się systemów w momencie pojawienia się usterki. Określeniem najlepszego stanu odtworzenia zajmowali się Lin i Kuo [96]. Badania nad metrykami do ewaluacji systemów rozproszonych, również dla propagacji błędów, opublikowano w pracy [88]. W celu lepszego zrozumienia mechanizmów propagacji zaprojektowano sieć semantyczną pozwalającą identyfikować wszystkie możliwe scenariusze nieprawidłowych zachowań [47]. W pracy [96] opisano schemat znajdowania najlepszego punktu wycofania po rozpoznaniu nieprawidłowości w działaniu systemu. Inne podejście do procesów propagacyjnych zostało zaprezentowane podczas operacji zaokrąglania wyników obliczeń zmiennoprzecinkowych [100].

W publikacjach podkreśla się, że błędy mogą się szybko rozprzestrzeniać i należy umieć odwrócić ten proces, najczęściej przez powrót do stanu prawidłowego sprzed zmiany. Niewiele za to jest prac analizujących przebieg procesu propagacji błędów oraz to, czy można go przewidzieć i co zrobić, aby zminimalizować ewentualne skutki.

Rzeczywiste systemy sieciowe niewiele mają wspólnego z klasycznymi strukturami znanymi z teorii grafów [111]. Ostatnie badania dotyczące przepływu informacji w systemach internetowych, w tym portalach społecznych, dowodzą, że systemy te nie przypominają ani struktur regularnych, ani przypadkowych. Stąd też pojawiły się topologie małych światów oraz topologie bezskalowe. Znanymi przykładami są sieci współpracy naukowej czy sieci regulatorowe (oddziaływań) genów. Wyniki badań w dziedzinie modelowania i organizacji wirtualnych przedsiębiorstw [48] wyraźnie potwierdzają przewagę kilku trwałych połączeń nad wieloma połączeniami mniej odpornymi na błędy. To samo zaobserwowano w kontekście tworzenia wirtualnych zespołów. Krótkie, ale za to bezpieczne ścieżki komunikacyjne pozwalają na większą efektywność dynamicznie tworzonych zespołów. Sakata w pracy [124] zauważył topologiczne podobieństwo między sieciami społecznymi i biologicznymi, np. centralnym układem nerwowym ssaków. Liczba wzorców wzajemnych połączeń występujących w rzeczywistych sieciach znacznie przewyższa te generowane w sieciach przypadkowych [105].

Mimo to sieci rzeczywiste są analizowane i porównywane na podstawie pojęć z klasycznej teorii grafów, np. średniego stopnia wierzchołka, średniej długości drogi, rozkładu stopni wierzchołków, współczynnika gronowania (klasteryzacji). Sieci skonstruowane na danych rzeczywistych charakteryzują się trzema cechami. Po pierwsze, są rzadkie. Większość wierzchołków jest połączona z małym procentem pozostałych wierzchołków i liczba krawędzi jest zbliżona bardziej do liczby wierzchołków niż do ich kwadratu. Po drugie, charakteryzują się krótką średnią długością drogi i wysokim współczynnikiem lokalnego gronowania (klastrowania). Drugi z parametrów – współczynnik gronowania, znacznie odróżnia je od grafów przypadkowych. Po trzecie, rozkład stopni wierzchołków ma charakter potęgowy. Oznacza to, że mała liczba wierzchołków posiada wiele krawędzi i przez to wielu sąsiadów. Poza tym, istnieje duża grupa wierzchołków, które mają niewiele krawędzi, a więc i sąsiadów.

Jednym z najważniejszych problemów struktur sieciowych jest odporność na uszkodzenia i adaptowalność do zmian w dynamicznym środowisku [136]. Odporność na błędy wynika wprost z charakterystyki systemu sieciowego. Wraz ze wzrostem rozproszenia zwiększa się ryzyko ataków, czego skutkiem mogą być awarie, utrata danych oraz nieprawidłowości w działaniu. Z kolei adaptowalność to zdolność do przystosowywania własnej architektury i działania do zachodzących zmian w otaczającym środowisku. Obie wymienione cechy silnie zależą od typu struktury topologicznej połączeń występujących w badanym systemie.

W systemach wieloagentowych, które często wykorzystują mechanizmy propagacyjne, występują struktury regularne i nieregularne, takie jak: grafy pełne, struktury gwiaździste, pierścieniowe i drzewiaste [38], [69], [152]. Czasami dodatkowo wyróżnia się strukturę webową i gridową [158].

Ponieważ architektury współczesnych systemów rozproszonych różnią się od struktur regularnych i przypadkowych [111], można przypuszczać, że procesy propagacyjne będą się różnić ilościowo, jakościowo i statystycznie, ale merytorycznie pozostaną takie same. Struktury małego świata z krótkimi ścieżkami i lokalnym grupowaniem [132], [143], struktury bezskalowe [1], [10], [19], czyli struktury sieci WWW, P2P [97] oraz struktury społeczne [33], nie są jeszcze dobrze zbadane pod tym względem. Wydaje się, że propozycje rozwiązań opisanych w tym rozdziale mogą być solidnym fundamentem lepszego poznania także struktur rzeczywistych ze względu na przyjętą uniwersalną metodologię badań oraz występujące pewne podobieństwa w działaniu.

Co determinuje przebieg zjawiska propagacji błędów w rozproszonych systemach informacyjnych? Czy topologia systemu ma wpływ na ten proces? Jeżeli tak – która topologia jest najbardziej odporna na błędy? Aby znaleźć odpowiedź na te pytania, zostały sprecyzowane następujące zadania: (a) analiza znaczenia zjawiska propagacji błędów w systemach sieciowych, (b) określenie podstawowych pojęć z dziedziny systemów rozproszonych, w tym typów błędów i możliwości ich propagacji, (c) eksperymentalne badanie przebiegu zjawiska propagacji błędów w różnych topologiach i opracowanie wniosków.

W jednej z poprzednich publikacji [83] autor prezentowanej pracy pokazał, jak można zaprojektować środowisko programistyczne umożliwiające propagację pakietów klas i danych. Nacisk położono jednak na zyski, jakie można osiągnąć, stosując mechanizm propagacji, a nie na analizę sytuacji nieprawidłowych i ich konsekwencji.

Za punkt wyjścia do analizy propagacji błędów zostały przyjęte założenia opublikowane w: [15], [46], [153], w których zamieszczono przykłady modelowania procesu propagacji wiadomości w środowisku grafów przypadkowych.

W dalszej części rozdziału zaproponowano ilościowe, jakościowe i statystyczne metody analizy potencjalnych scenariuszy propagacyjnych dla różnych konstrukcji sieciowych, typów błędów i rozkładów prawdopodobieństw. Przedmiotem studiów są eksperymenty symulacyjne wyznaczania sumy elementów w środowisku rozproszonym z uwzględnieniem możliwości wystąpienia błędu.

## 4.2. Sieciowe struktury topologiczne

Struktury regularne zostały najlepiej poznane i opisane jako homogeniczne wzorce połączeń wierzchołków. W tych strukturach rozkład stopni wierzchołków jest trywial-

ny, wszystkie mają ten sam stopień. Przykładem takiego układu połączeń są grafy pełne, kraty i hiperkostki.

Struktury przypadkowe nie są dobrym reprezentantem sieci rzeczywistych, choć mogą charakteryzować się krótką średnią długością drogi, jednakże rozkład stopni wierzchołków ma charakter normalny.

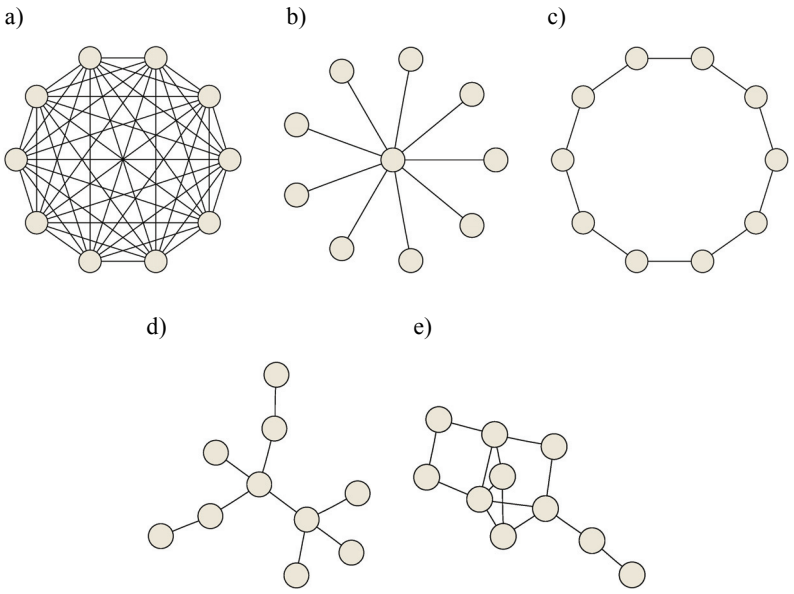
Struktury małych światów charakteryzują się wysokim lokalnym współczynnikiem gronowania (jak kraty) i krótką średnią długością drogi (jak grafy przypadkowe). Właściwość ta została nazwana małym światem. Jak można wywnioskować, ten typ połączeń da się umiejscowić między strukturami regularnymi a przypadkowymi. W grafach o takich cechach można szybko przemieszczać się od jednego wierzchołka do innego, najczęściej średnia liczba kroków nie przekracza wielkości 3–4. Dodawanie wierzchołka powoduje niewielką zmianę średniej długości drogi. Strogatz [132] i Watts [143] pokazali, że te dwie cechy, krótkich ścieżek i wysokiego gronowania, można odkryć w wielu rzeczywistych sieciach złożonych. Jest to zgodne z naszą intuicją, gdyż krótka długość drogi, bez wielu pośredników, pozwala na szybką komunikację między różnymi częściami systemu, przez co synchronizacja w dowolnym globalnym procesie zostanie znacznie ułatwiona.

Struktury bezskalowe [1], w tym webowe [19], typu P2P [97] oraz niektóre struktury społeczne, nie posiadają właściwości małych światów, jednakże zachowują się zgodnie z regułą wzrostu oraz preferencyjnego dołączania węzłów. Pierwsza z nich mówi o ciągłym ewoluowaniu, wzroście struktury, natomiast druga oznacza, że nowy wierzchołek jest częściej łączony z tymi, które już mają wielu sąsiadów. Jest to przydatne w procesie odwrotnym, gdyż losowe usuwanie wierzchołka zwykle nie wpływa znacząco na strukturę powiązań, chyba że zostanie wybrany taki, który jest połączony z wieloma innymi.

Wymienione cztery modele powiązań są najczęściej cytowane w literaturze i praktycznie w pełni pozwalają zamodelować sieci rzeczywiste.

Ważnym parametrem sieci jest jej spójność. Wyróżnia się spójność wierzchołkową i krawędziową. Nie wystarcza ona jednak do określenia wpływu topologii na mechanizm propagacji błędów. Znane są publikacje, takie jak [125], w którym pojawia się stwierdzenie, że im większa spójność topologiczna struktury połączeń, tym wyższa tolerancja na błędy. Jest to mimo wszystko bardzo duże uproszczenie, gdyż nie można uwzględniać jedynie tego parametru. Jednym z celów przeprowadzonych badań jest pokazanie, że różne struktury topologiczne o podobnej spójności mogą podczas procesu propagacyjnego różnie się zachowywać.

Topologia systemu determinuje sposób komunikacji między jego elementami. Widać to zarówno na płaszczyźnie logicznej, jak i fizycznej, choć nie każde fizyczne połączenie będzie wykorzystywane podczas transferu danych między poszczególnymi częściami systemu. Szerszej informacji o różnych metodach komunikacji zależnej od wybranej struktury topologicznej można znaleźć w pracy [88].




Rys. 4.1. Struktury topologiczne: a) *SPE*, b) *SG*, c) *SPI*, d) *SD*, e) *SPR*

Aby można było wyciągnąć ogólne wnioski, co do zachowania się błędu w systemach rozproszonych, rozpatrywane będą następujące struktury z klasy topologii regularnych: pełna (*SPE*), gwiaździsta (*SG*), pierścieniowa (*SPI*), drzewiasta (*SD*) i dodatkowo struktura przypadkowa (*SPR*), przykłady na rys. 4.1.

W strukturze pełnej nie ma wyodrębnionego elementu, wszystkie wierzchołki komunikują się bezpośrednio, na zasadzie każdy z każdym. Choć struktura przez to charakteryzuje się maksymalną redundancją połączeń – właśnie dzięki temu jest najbardziej wiarygodną wśród wszystkich pozostałych. Z drugiej strony, nadmiarowość połączeń wiąże się z dodatkowymi kosztami ich utrzymania. Ponadto im więcej wiadomości wysyłanych, tym bardziej prawdopodobne jest wystąpienie błędu podczas transmisji.

W strukturze gwiaździstej wyróżniamy jeden centralny wierzchołek oraz wiele końcowych połączonych tylko z tym wierzchołkiem. Dlatego nie występuje redundancja połączeń. Zawsze istnieje dokładnie jedna droga między dwoma wierzchołkami. Wystąpienie błędu połączenia z wierzchołkiem końcowym nie powoduje blokady całego systemu. Taka sytuacja ma miejsce tylko w przypadku awarii wierzchołka centralnego, gdyż jest on stale obciążany podczas komunikacji, stąd prawdopodobieństwo jego awarii znacznie wzrasta.

W strukturze pierścieniowej każdy wierzchołek ma dwóch sąsiadów, dlatego istnieją dwie drogi komunikacji między nimi. Dzięki temu została zwiększona wiarygodność połączeń. Wadą tego rozwiązania jest duża średnia odległość oraz wielka

liczba pośrednich wierzchołków, które muszą propagować otrzymywane wiadomości, o ile nie były ich adresatem.

W strukturze drzewiastej nie występują cykle, a wierzchołki łączy tylko jedna możliwa droga. Zaletą tej struktury jest równomierny podział obciążenia na wszystkie połączenia i wierzchołki. Wystąpienie błędu może jednak częściowo lub całkowicie zablokować działanie systemu.

Struktura przypadkowa nie posiada żadnych regularności czy hierarchiczności. Redundancja połączeń ma zatem także charakter losowy. Napotkamy więc tam gęsto i rzadko połączone grupy wierzchołków. W taki sposób powstają na przykład sieci P2P.

## 4.3. Typy błędów w systemach sieciowych

Błędem dla systemu sieciowego jest pojawienie się nieprawidłowego stanu jednego lub wielu jego komponentów. Wadliwe działanie może być spowodowane błędem projektowym czy programistycznym, mechanicznym uszkodzeniem, pogorszeniem stanu technicznego albo warunków działania, błędem administratora itd. Błędy mogą mieć charakter tymczasowy lub trwały, lokalny lub globalny.

Najczęściej błędy są grupowane w trzech klasach [125]: błędy wykonania, błędy transmisji oraz błędy komponentów. Komplementarnie do tych klas definiowane są odpowiednie modele błędnego działania – na poziomie połączeń albo komponentów. Ponieważ propagacja danych jest związana zarówno z transmisją danych, jak i ich przetwarzaniem w komponentach, w dalszej części będzie rozpatrywany model hybrydowy.

Tak jak w przypadku wytwarzania oprogramowania, wyróżnia się błędy składniowe i błędy logiczne, tak w systemie sieciowym nieprawidłowy stan może spowodować jeden z trzech typów błędów transmisji: *pominięcie*, *wstawienie* i *zanieczyszczenie*. W literaturze błędy te nazywane są błędami bizantyjskimi [125]. W pracy do wymienionej grupy został dodany czwarty typ nazwany *awarią*, który może być interpretowany jako szczególny przypadek błędu pominięcia. W przypadku jednak permanentnego zablokowania wszystkich możliwych połączeń tego komponentu, czyli kiedy dochodzi do jego pełnego uszkodzenia, następstwa działania systemu sieciowego zwykle są na tyle znaczące, by dodatkowo wyodrębnić ten typ zachowania.

*Pominięcie* oznacza zablokowanie wysłania danych, które powinny być przekazane dalej. Może to być związane z utratą fizycznego połączenia z pojedynczym wierzchołkiem.

*Wstawienie* z kolei generuje dane, które nie miały być wysłane. Często taka sytuacja ma miejsce podczas zamierzonego ataku na wybrany komponent systemu sieciowego.

*Zanieczyszczenie* wprawdzie transmituje dane, ale jednocześnie zmienia je. Najpowszechniejszym tego typu zjawiskiem są wirusy komputerowe doklejające się do kodu innych programów.

Ostatni wyróżniony typ błędu – *awaria*, kończy aktywność komponentu oraz wszelką możliwość komunikacji z nim. *Awaria* może być spowodowana na przykład uszkodzeniem procesora.

## 4.4. Rola komunikatów w procesie propagacji

Aby dobrze zrozumieć propagację błędów, można zacząć od wymiany komunikatów, nazywanej też przesyłaniem wiadomości. Taką formę komunikacji spotyka się w codziennym życiu, np. w czasie przesyłania innemu użytkownikowi otrzymanej wcześniej wiadomości SMS czy e-mail. Ostatnio coraz większą rolę w komunikacji natychmiastowej, oprócz poczty elektronicznej i SMS, odgrywają internetowe komunikatory, zarówno w postaci aplikacji stacjonarnych, jak i mobilnych [71].

Innym przykładem ilustrującym problem przesyłania wiadomości może być otwarcie sklepu w miasteczku. Pierwsi zadowoleni klienci będą przekazywać informację o nowym sklepie swoim znajomym. Równocześnie jest prowadzona akcja marketingowa służąca reklamie. Po pewnym czasie można oczekiwać, że duża grupa mieszkańców zostanie powiadomiona o otwarciu nowego punktu sprzedaży. Wtedy pojawia się zasadnicze pytanie, jak długo należy prowadzić jakiekolwiek działania marketingowe oraz czy prowadzona akcja dociera do nowych potencjalnych klientów i nie ogranicza się jedynie do osób, które już zostały powiadomione wcześniej. W odpowiedzi na rodzące się pytania może pomóc analiza procesu propagacji wiadomości, w tym wypadku w sieci społecznej.

Podstawowym celem propagacji wiadomości jest jak najszybsze dotarcie do wszystkich dostępnych części systemu sieciowego przy jak najmniejszej liczbie generowanych wiadomości. Zadanie wyszukania dostępnych elementów systemu sieciowego realizuje m.in. algorytm rozgłaszania zalewowego. Jest to jeden z podstawowych algorytmów dystrybucji informacji w sterowaniu ruchem w sieciach komputerowych [98], [125]. Ponadto jest on wykorzystywany w sieci Gnutella w propagacji zapytań [29] i w protokole OSPF w procesie monitorowania stanu sieci [51], [89]. Choć algorytm zalewowy propaguje wiadomość z maksymalną prędkością, jednocześnie generuje przy tym dużą liczbę wiadomości. Innym algorytmem często wykorzystywanym w procesie dystrybucji wiadomości jest błądzenie przypadkowe [50]. Już niewielka liczba wiadomości może uruchamiać proces propagacyjny, który jednak będzie dość wolny. Interesującym aspektem badań nad efektywnością tych algorytmów jest poszukiwanie odpowiedzi na pytanie o minimalną

liczbę wiadomości potrzebnych do odwiedzenia wszystkich węzłów sieci. Podczas działania algorytmu zalewowego lub błądzenia przypadkowego kluczowym pojęciem jest pokrycie propagacyjne grafu określające liczbę odwiedzonych wierzchołków. Zamiast niego czasami używa się pojęcia odsetka propagacyjnego. Wybierając drugie pojęcie, w rozdz. 4.5 zostanie zdefiniowany odsetek błędów, dzięki któremu będzie można oceniać rozmiar uszkodzeń.

Proces propagacji błędów polega na przenoszeniu się błędu występującego w jednym komponencie systemu na inne. Ważną cechą tego procesu jest jego powtarzalność zgodnie ze ścieżką przesyłania wiadomości. W normalnych warunkach propagację może jedynie zatrzymać awaria lub odpowiednie przeciwdziałanie. Proces ten nierozerwalnie jest związany z algorytmem przesyłania komunikatów oraz ciągiem stanów (prawidłowy, nieprawidłowy) uczestniczących podmiotów na ścieżce propagacyjnej.

## 4.5. Analiza symulacyjna procesu propagacji błędów

### 4.5.1. Cele i założenia eksperymentów

Podstawowym celem przeprowadzonych eksperymentów symulacyjnych było zbadanie procesu propagacji błędów w środowisku rozproszonym. Założono, że wszystkie eksperymenty będą wykonywane jednocześnie na pięciu wybranych strukturach topologicznych (zob. rozdz. 4.2). W ten sposób uzyskano możliwość porównania charakterystyk różnych struktur. Ponieważ parametrów propagacji błędów jest dość dużo, w eksperymentach skoncentrowano się na tych przypadkach, które są istotne z punktu widzenia specyfiki procesu propagacyjnego w każdym systemie sieciowym.

Na potrzeby eksperymentów zbudowano rozproszony, ale spójny system wymiany komunikatów z wyróżnionymi węzłami obliczeniowymi. Komunikacja między dowolnymi węzłami odbywa się asynchronicznie w obu kierunkach z możliwością aktywacji połączenia w dowolnym węźle. Węzły są homogeniczne, nie ma więc żadnego centralnego zarządcy. Kolejne założenie mówi o rozróżnialności i podstawowej wiedzy węzłów, tzn. każdy z nich ma jednoznaczny identyfikator i zna swoich bezpośrednich sąsiadów.

W symulacji procesu propagacyjnego należy dobrać taki algorytm rozgłaszania, aby można było w prosty sposób monitorować jego działanie zarówno przed, jak i po wprowadzeniu błędów. Wybór padł na rozproszony algorytm wyznaczania sumy elementów. Problem ten jest szczególnym przypadkiem redukcji [31]. W omawianym przykładzie jest to suma arytmetyczna wartości przechowywanych w węzłach. Na

początku węzły są losowo inicjowane. Następnie w wyniku pełnej wymiany przechowywanych informacji każdy z węzłów zna sumę wszystkich wygenerowanych wartości początkowych. Ta wymiana zachodzi zgodnie z rozgłaszaniem zalewowym.

Algorytm 4.1 wyznaczania sumy elementów, w skrócie WSE, jest uruchamiany na wszystkich węzłach w taki sam sposób. Najpierw każdy z nich jest inicjowany losowo generowaną wartością – dla uproszczenia – pochodzącą ze zbioru liczb naturalnych. Następnie węzły komunikują się między sobą (wiersze 2–11) celem wyznaczenia sumy wszystkich początkowych wartości (wiersz 6). Newralgiczną częścią algorytmu jest warunek stopu, gdyż węzły, nie znając schematu powiązań ani wielkości systemu, nie wiedzą, kiedy mogą przestać odbierać wiadomości. Ten problem można rozwiązać na dwa sposoby. Pierwsze rozwiązanie zajmuje więcej pamięci i polega na podaniu dodatkowej informacji o liczbie sumowanych składników. Drugie rozwiązanie natomiast obciąża czas wykonania i polega na czekaniu, aż węzły odczytają wszystkie wiadomości z kolejki. W praktyce najczęściej wyznacza się czasowy parametr nasycenia, po którym już nie zachodzą żadne zmiany w obliczanych sumach.

**Algorytm** WSE

Parametry wejściowe: *brak*.

Wynik: wyznaczona suma elementów.

```
1:  begin
2:  prześlij wiadomość z wartością początkową wszystkim sąsiadom;
3:  while nie jest spełniony warunek stopu do
4:          odbierz wiadomość z kolejki wiadomości;
5:          if nie odebrałeś wcześniej wiadomości od tego nadawcy i wiadomość nie
            pochodzi od siebie samego then
6:                  oblicz sumę odebranej wartości z aktualnie pamiętaną w węźle;
7:                  prześlij tę wiadomość wszystkim sąsiadom;
8:          else
9:                  pomiń tę wiadomość;
10:         end if;
11: end while;
12: end;
```

Algorytm 4.1. Wyznaczanie sumy elementów

Z punktu widzenia procesu propagacji błędów w przypadku algorytmu WSE spełnione są dwa podstawowe założenia: o intensywnej wymianie informacji i o zależnościach między węzłami. Wartości początkowe z węzłów muszą być rozpropagowane

do wszystkich pozostałych, co oznacza, że stany przejściowe, jak i końcowe, są zależne od aktywności bezpośrednich sąsiadów w sieci. Przyjęte założenia idealnie pasują do badanego procesu propagacyjnego. Po pierwsze, wzmożona komunikacja dotyczy wszystkich węzłów. Po drugie, stan elementu sieciowego zależy od wartości zainicjowanych w pozostałych węzłach. Po trzecie, algorytm wyznaczania rozproszonej sumy pozwala na wprowadzenie błędów wszystkich badanych typów (zob. rozdz. 4.3). W wyniku wygenerowanego błędu wyznaczone sumy nie są takie same. W celu rozwiązania konfliktu i ich ujednolicenia można by użyć jednej z metod wyboru consensusu [107], [112], [114], nie będzie to jednak przedmiotem dalszej analizy.

Aby przeprowadzić eksperymenty dotyczące propagacji błędów, należało zmodyfikować algorytm WSE. Algorytm 4.2, nazywany dalej WSE-B, uwzględnia cztery typy błędów: *pominięcie*, *wstawienie*, *zanieczyszczenie* i *awarię*. *Pominięcie* polega na tym, że węzeł nie wysyła wiadomości sąsiadowi, choć powinien to zrobić. Podobnie, ale na odwrót, *wstawienie* wysyła dodatkową wiadomość z przypadkowo wygenerowaną zawartością. *Zanieczyszczenie* wprowadza losowo wygenerowaną zmianę zawartości w wysyłanej wiadomości, a *awaria* powoduje całkowite zatrzymanie działania węzła. Losowa zmiana treści wiadomości może dotyczyć zarówno przekazywanej wartości początkowej, jak i identyfikatora nadawcy. W pojedynczym teście generowany jest tylko jeden błąd w losowo wybranym węźle. Generowanie sekwencji błędów na tym etapie zwiększy jedynie złożoność problemu i może zburzyć czytelność interpretacyjną uzyskanych wyników.

W ogólnym przypadku mamy kilka możliwości wyboru miejsca sprawdzenia ewentualnego pojawienia się błędu. Z punktu widzenia poprawności działania algorytmu przyjęto najbardziej niekorzystny wariant, który powoduje zmiany w obliczaniu sumy dla jak największej grupy węzłów. *Awaria* została zatem uwzględniona jeszcze przed odebraniem jakiejkolwiek wiadomości (wiersze 4–6). Ponieważ mamy do czynienia ze środowiskiem wielowątkowym, musimy też uwzględnić sytuację, kiedy proces wysyłania wartości początkowych może zostać z powodu *awarii* przerwany. Jeśli ten typ uszkodzenia wystąpi, węzeł automatycznie kończy swoje działanie. Błędy *pominięcia* (wiersze 11–12) i *zanieczyszczenia* (wiersze 13–15) są uwzględniane na etapie wysyłania wiadomości do sąsiadów. Wstawienie może mieć miejsce już po wysłaniu wszystkich wiadomości (wiersze 22–24).

Z uwagi na to, że błędy w rzeczywistym systemie pojawiają się w różnych punktach czasu, wprowadzono prawdopodobieństwo pojawienia się błędu $p_b$. Dzięki temu będzie można sprawdzić, czy istnieje zależność między momentem pojawienia się błędu a jego wpływem na działanie systemu. Ponieważ niektóre błędy nie zawsze wygenerują błędną sumę, zostanie również uwzględniony czas wyznaczenia sumy. Ponadto przyjęto, że rozmiar uszkodzeń będzie oceniany na podstawie odsetka błędów. Wprawdzie miara ta nic nie mówi o przebiegu procesu propagacyjnego, ale pozwala na określenie wielkości zmian i może być punktem wyjścia do analiz statystycznych. Odsetek błędów definiujemy następująco:

$$o_b = \frac{LWB}{LW} \tag{4.1}$$

*LWB* – liczba węzłów obciążonych błędem, *LW* – liczba wszystkich węzłów.

**Algorytm** WSE-B

Parametry wejściowe: brak.

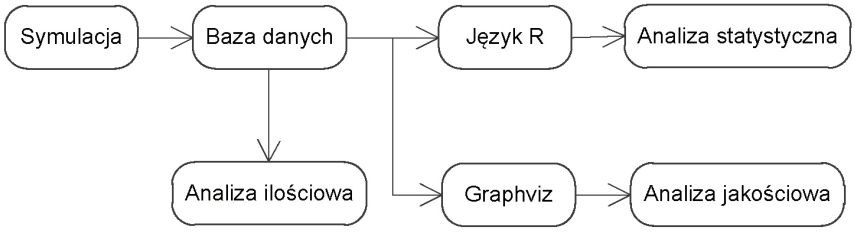
Wynik: wyznaczona suma elementów.

```
1:  begin
2:  prześlij wiadomość z wartością początkową wszystkim sąsiadom;
3:  while nie jest spełniony warunek stopu do
4:          if pojawiła się awaria then
5:              zakończ;
6:          end if;
7:          odbierz wiadomość z kolejki wiadomości;
8:          if nie odebrałeś wcześniej wiadomości od tego nadawcy i wiadomość nie
            pochodzi od siebie samego then
9:              oblicz sumę odebranej wartości z aktualnie pamiętaną w węźle;
10:             for each sąsiada do
11:                 if pojawiło się pominięcie then
12:                     kontynuuj, ale nie wysyłaj wiadomości do sąsiada;
13:                 else if pojawiło się zanieczyszczenie then
14:                     losowo zmień zawartość wiadomości;
15:                     prześlij tę wiadomość;
16:                 else prześlij tę wiadomość;
17:                 end if;
18:             end for;
19:         else
20:             pomiń tę wiadomość;
21:         end if;
22:         if pojawiło się wstawienie then
23:             prześlij losowo wygenerowaną wiadomość do przypadkowo wybranego
                sąsiada;
24:         end if;
25: end while;
26: end;
```

Algorytm 4.2. Wyznaczanie sumy elementów z uwzględnieniem wystąpienia błędu

## 4.5.2. Opis przeprowadzonych eksperymentów

Symulacje zostały przeprowadzone przy użyciu języka Python[2] w wersji 2.5. Wybór tego narzędzia o otwartym kodzie źródłowym był uzasadniony z kilku powodów. Po pierwsze, może być używany nieodpłatnie nawet w aplikacjach komercyjnych. Po drugie, istnieje duży wybór bibliotek wspomagających programistę. W eksperymentach wykorzystano pakiet symulacyjny SimPy[3] i interfejs bazodanowy PyGreSQL[4]. Po trzecie, Python Software Foundation wspiera badania naukowe, a język Python stał się standardem w eksperymentach symulacyjnych. Wyniki eksperymentów były zapamiętywane w bazie Postgre SQL 8.1[5]. Do obliczeń statystycznych i wizualizacji wyników wykorzystano język R[6]. Dodatkowo w analizie wizualnej było używane oprogramowanie Graphviz[7].




Rys. 4.2. Schemat przetwarzania wyników eksperymentów

Przeprowadzenie testu symulacyjnego polega na utworzeniu zbioru obiektów typu węzeł, przypisaniu początkowej wartości, wyborze architektury topologicznej i wygenerowaniu połączeń międzywęzłowych, wyborze typu błędu i wpisaniu go do jednego wybranego obiektu oraz uruchomieniu symulacji. Następnie zostaną zapamiętane i przetworzone wyniki eksperymentów zgodnie ze schematem 4.2.

Obiekt typu węzeł zawiera listę wiadomości, niepowtarzalny identyfikator, wartość początkową, wyznaczaną sumę elementów, zbiór odebranych wiadomości i dwie zmienne odpowiadające za błędy: rodzaj błędu oraz wartość logiczną 0–1. Wartością zmiennej *rodzaj błędu* może być: *pominięcie*, *wstawienie*, *zanieczyszczenie*, *awaria* lub *brak błędu*. Druga zmienna przechowuje wartość logiczną wskazującą, czy błąd został już wygenerowany. Ze względu na przyjęte założenie o różnym momencie pojawienia się błędu zapamiętanie tego faktu jest konieczne. Lista wiadomości przechowuje informacje przysłane do węzła, ale jeszcze nieprzetworzone. Wartość początko-

[2] http://python.org/
[3] http://simpy.sourceforge.net/
[4] http://www.pygresql.org/
[5] http://www.postgresql.org/
[6] http://www.r-proj ect.org/
[7] http://www.graphviz.org/

wa jest losowo generowana przy inicjalizacji obiektu. Zmienna przechowująca wartość wyznaczanej sumy elementów na początku wynosi tyle samo, co zmienna z wartością początkową. Zbiór odebranych wiadomości zawiera wszystkie odebrane i przesłane dalej wiadomości. Zbiór ten jest tworzony, by można było filtrować przychodzące wiadomości i pomijać te, które są od tych samych nadawców.

W bazie danych są zapamiętywane zarówno parametry testów, jak i uzyskane miary. W pierwszej grupie występują: typ struktury topologicznej, liczba węzłów, prawdopodobieństwo i rodzaj błędu. Do drugiej grupy zalicza się: odsetek błędów, graf reprezentujący stan końcowy węzłów i czas wykonania testu. Dane wynikowe są eksportowane do pliku i przetwarzane w środowisku R. Analogicznie dane stanów końcowych węzłów są eksportowane do narzędzia Graphviz, co umożliwia ich wizualizację.

W czasie badań symulacyjnych wykonano 6000 testów. Każdy z nich został opisany przez liczbę węzłów (10, 20, 30, 40, 50 i 100), typ struktury topologicznej (*SPE*, *SG*, *SPI*, *SD* i *SPR*), prawdopodobieństwo wystąpienia błędu $p_b$ (0,1, 0,3, 0,5, 0,8, 0,9) oraz typ błędu (*brak*, *pominięcie*, *wstawienie*, *zanieczyszczenie* i *awaria*). W ten sposób została utworzona czterowymiarowa przestrzeń symulacyjna. W każdym punkcie tej przestrzeni wykonano pomiar odsetka błędów i czasu trwania testu. Pojedynczy test był powtarzany 10-krotnie z tym samym zestawem parametrów. Zarówno moment pojawienia się błędu, jak i węzeł, który ulegał uszkodzeniu, były jednak losowane za każdym razem.

Wyniki przeprowadzonych eksperymentów zostaną przedstawione w formie tabel i grafów w trzech częściach: najpierw pod względem ilościowym, potem jakościowym i na końcu statystycznym.

### A. Analiza ilościowa uzyskanych wyników

Tabela 4.1. Odsetek błędów w zależności od struktury topologicznej i typu błędu

| *TST* \ *TB* | *Brak* | *Pominięcie* | *Wstawienie* | *Zanieczyszczenie* | *Awaria* | $\overline{o_b}$ |
|---|---|---|---|---|---|---|
| *SPE* | 0 | 0 | 0 | 0,04 | 0,51 | 0,11 |
| *SPR* | 0 | 0,05 | 0,16 | 0,28 | 0,55 | 0,21 |
| *SPI* | 0 | 0 | 0,39 | 0,44 | 0,51 | 0,27 |
| *SG* | 0 | 0,49 | 0,31 | 0,46 | 0,58 | 0,37 |
| *SD* | 0 | 0,52 | 0,31 | 0,52 | 0,69 | 0,41 |
| $\overline{o_b}$ | 0 | 0,21 | 0,23 | 0,35 | 0,57 | 0,27 |

W tabeli 4.1 przedstawiono zależność wielkości odsetka błędów od struktury topologicznej i typu błędu. W wierszu scharakteryzowano wybraną strukturę topologiczną, w kolumnie – określony typ błędu. Ostatni wiersz i ostatnia kolumna zawierają wartości uśrednione odsetka błędów. Wiersze zostały uporządkowane od topologii najmniej podatnej na błędy do najbardziej podatnej. Kolumny zostały uporządkowane od najmniej szkodliwego błędu do najbardziej szkodliwego. Uzyskane wartości zerowe

w kolumnie *Brak* potwierdzają poprawność algorytmu propagacyjnego, gdyż wyznaczone sumy we wszystkich węzłach były sobie równe. Struktury topologiczne, w których występują lub mogą wystąpić cykle (*SPE*, *SPR* i *SPI*) są bardziej odporne na błędy niż struktury acykliczne (*SG*, *SD*). Cykl ma charakter redundantny, czasami jednak jego występowanie jest pożądane, choćby w przypadku uszkodzenia jednego komponentu krytycznego (zob. p. 2.4.1.2). Błąd *pominięcia* w strukturach redundantnych praktycznie nie występuje. Ponadto można zauważyć, że poszczególne typy błędów w różny sposób oddziałują na struktury topologiczne. Mimo wszystko, prawie zawsze, kolejność odporności struktury na określony błąd jest taka sama, czyli odporność maleje od struktury pełnej (*SPE*) do drzewiastej (*SD*).

Tabela 4.2. Odsetek błędów w zależności od struktury topologicznej i liczby węzłów

| *LW* \ *TST* | *SPE* | *SPR* | *SPI* | *SG* | *SD* | $\overline{o_b}$ |
|---|---|---|---|---|---|---|
| 10 | 0,13 | 0,21 | 0,23 | 0,31 | 0,4 | 0,26 |
| 20 | 0,09 | 0,2 | 0,26 | 0,35 | 0,39 | 0,26 |
| 30 | 0,13 | 0,2 | 0,27 | 0,34 | 0,44 | 0,28 |
| 40 | 0,11 | 0,18 | 0,27 | 0,43 | 0,41 | 0,28 |
| 50 | 0,09 | 0,24 | 0,28 | 0,39 | 0,44 | 0,29 |
| 100 | 0,11 | 0,22 | 0,29 | 0,39 | 0,39 | 0,28 |
| $\overline{o_b}$ | 0,11 | 0,21 | 0,27 | 0,37 | 0,41 | 0,27 |
| $o_s$ | 0,02 | 0,02 | 0,02 | 0,04 | 0,02 | 0,01 |

W tabeli 4.2 zaprezentowano zależność wielkości odsetka błędów od struktury topologicznej (kolumny) i liczby węzłów (wiersze). Ostatnie dwa wiersze zawierają średnią arytmetyczną odsetka błędów oraz wartości odchylenia standardowego. Analiza wartości odsetka błędów dla wybranej struktury topologicznej wskazuje na stałą jej wartość, praktycznie niezależną od liczby węzłów. Oznacza to, że wielkość struktury praktycznie nie ma wpływu na jej odporność na błędy. Podobnie zachowają się topologie o małej, jak i dużej liczbie węzłów, mimo to odsetek błędów jest dobrym wskaźnikiem skali ewentualnej propagacji błędów w badanej strukturze sieciowej.

Tabela 4.3. Odsetek błędów w zależności od struktury topologicznej i prawdopodobieństwa wystąpienia błędu

| $p_b$ \ *TST* | *SPE* | *SPR* | *SPI* | *SG* | *SD* | $\overline{o_b}$ |
|---|---|---|---|---|---|---|
| 0,1 | 0,03 | 0,07 | 0,12 | 0,07 | 0,15 | 0,09 |
| 0,3 | 0,06 | 0,15 | 0,21 | 0,23 | 0,33 | 0,2 |
| 0,5 | 0,11 | 0,23 | 0,28 | 0,38 | 0,42 | 0,28 |
| 0,8 | 0,17 | 0,28 | 0,34 | 0,53 | 0,53 | 0,37 |
| 0,9 | 0,17 | 0,3 | 0,36 | 0,61 | 0,58 | 0,4 |

W tabeli 4.3 pokazano zależność wielkości odsetka błędów od struktury topologicznej i prawdopodobieństwa wystąpienia błędu. W kolumnach, tak jak poprzednio, zawarto różne struktury topologiczne, natomiast w wierszach występują użyte w symulacjach wartości prawdopodobieństw pojawienia się błędu. Prawdopodobieństwa pozwalają na zbadanie zachowania się struktury dla różnego nasycenia występowania błędów w czasie. Jeśli więc to prawdopodobieństwo jest duże, błędy będą pojawiać się często w małych odstępach czasu. Wniosek nasuwający się po analizie wyników o proporcjonalnej zależności prawdopodobieństwa wystąpienia błędu i uzyskanego odsetka można wytłumaczyć dużą wrażliwością algorytmu WSE-B na wczesne pojawienie się ewentualnego uszkodzenia. Podobną cechę wykazują też inne „negatywne" procesy propagacyjne, szczególnie te dotyczące propagacji wirusów czy plotek.

Tabela 4.4. Średni czas trwania wykonanych testów
w zależności od struktury topologicznej i typu błędu [s]

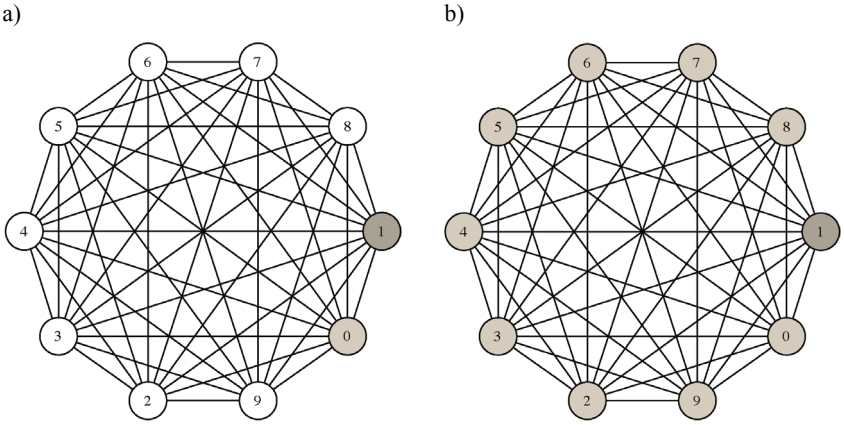
| *TST* / *TB* | *SPE* | *SPR* | *SPI* | *SG* | *SD* |
|---|---|---|---|---|---|
| *brak* | 7,9 | 2,8 | 0,8 | 7,8 | 1,4 |
| *pominięcie* | 11,9 | 91,5 | 0,9 | 1256 | 1684 |
| *wstawienie* | 7,9 | 2,8 | 0,8 | 7,8 | 1,4 |
| *zanieczyszczenie* | 7,9 | 2,8 | 0,8 | 7,8 | 1,4 |
| *awaria* | 1168 | 1335 | 1205 | 1421 | 2054 |

W tabeli 4.4 zawarto średni czas, jaki był potrzebny do zakończenia testów w każdej badanej strukturze topologicznej ze względu na typ błędu. Z tabeli można odczytać, że istotny wpływ na czas działania algorytmu WSE-B miała utrata danych w wyniku błędu *pominięcia* i wystąpienia *awarii*. W tych przypadkach czas działania wydłużył się znacznie. Nawet w strukturach odpornych na uszkodzenia (*SPE*, *SPR*) w przypadku wystąpienia błędu *pominięcia* algorytm działał dłużej. Wystąpienie awarii węzła powodowało w wielu przypadkach takie opóźnienie, że warunkiem stopu nie było ustabilizowanie się wyliczanych sum, ale wprowadzona górna granica czasu dla pojedynczego węzła, która przerywała działanie algorytmu. Wielkość jej została ustalona doświadczalnie na 100 s. W ten sposób była wielokrotnością uzyskiwanych czasów „normalnego" kończenia działania algorytmu. Osiągnięcie wartości granicznej oznaczało zatem brak możliwości ustabilizowania się sum wyznaczanych w węzłach. Innym wnioskiem, który się nasuwa po analizie danych odczytanych z tabeli, jest prawie niezauważalny wpływ pozostałych typów błędów (*wstawienia* i *zanieczyszczenia*) na uzyskany czas trwania przeprowadzanych testów. Wynika to z tego, że wymienione typy błędów nie powodują istotnego spowolnienia działania algorytmu, a jedynie mogą wprowadzać dezinformację.

### B. Analiza jakościowa uzyskanych wyników

Do tej pory uwaga skoncentrowana była na ilościowym podejściu w analizie uzyskanych wyników z eksperymentów. Podstawowa miara tego podejścia, odsetek błędów, niestety nie mówi nic więcej, ponad to, ile węzłów zostało zainfekowanych błędem. W praktyce bardzo ważnym kryterium oceny procesu propagacyjnego jest jego przebieg. Do analizy ścieżek propagacyjnych błędów zostanie wykorzystana metoda wizualna bazująca na obrazie końcowym stanu węzłów przy użyciu oprogramowania Graphviz.

Analiza jakościowa stanów końcowych będzie miała na celu znalezienie charakterystycznych ścieżek propagacyjnych, co pozwoli sformułować nowe wnioski. Za każdym razem obraz stanu końcowego sieci jest zapamiętywany w postaci nieskierowanego grafu. Węzły sieci są wierzchołkami, a połączenia są reprezentowane przez krawędzie. Liczba umieszczona w wierzchołku oznacza identyfikator węzła. Pierwszy węzeł, który wysyła błąd, ma kolor ciemnoszary, natomiast węzły zainfekowane w wyniku propagacji mają kolor jasnoszary. Pozostałe mają kolor biały. Przyjmując te uniwersalne założenia, nie ma potrzeby przeprowadzać analizy wizualnej dla każdego typu błędu oddzielnie. Rysunki dla różnych typów błędów będą przedstawiane w taki sam sposób.




Rys. 4.3. Wizualizacja w strukturze *SPE*: a) *zanieczyszczenia*, b) *awarii*

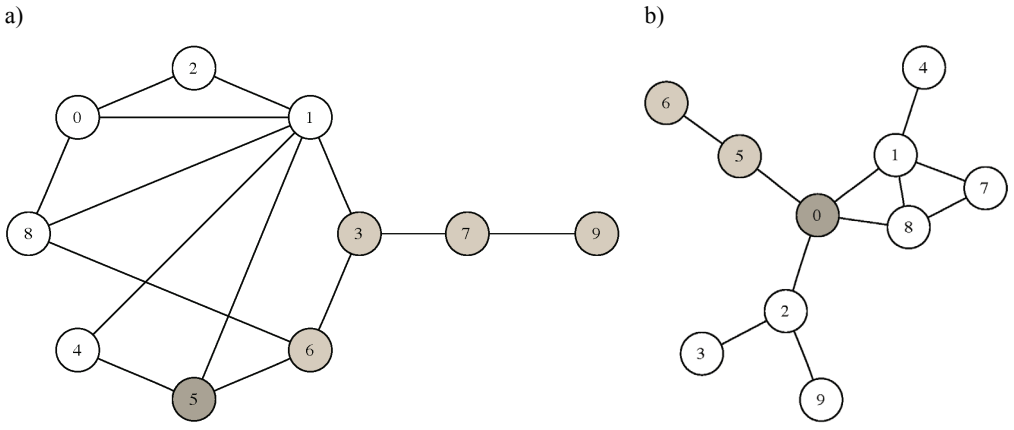
Na rysunku 4.3 pokazano ścieżkę propagacyjną zainicjowaną przez błąd *zanieczyszczenia* (a) i wystąpienie *awarii* (b) dla grafu pełnego. W obu przypadkach początkowym, infekującym węzłem był węzeł 1.

W przypadku (a) ścieżka propagacyjna błędu jest krótka i kończy się już na węźle 0. Prawdopodobnie wiadomość propagowana przez węzeł 1 do węzła 0 została zanieczyszczona. Inny scenariusz jest praktycznie niemożliwy. Warto jednak zwrócić uwagę,

że błąd zatrzymał się w węźle 0 i nie był dalej rozsyłany. Inne węzły odebrały w tym czasie poprawne wiadomości i pomijały kolejne od tych samych nadawców. Ponieważ strukturę pełną charakteryzuje redundancja połączeń, pozostałe węzły, w tym węzeł początkowy, uzyskały poprawną sumę wartości początkowych. Węzeł 0 nie mógł też dalej powodować błędów, gdyż pozostałe węzły otrzymały wcześniej poprawne dane bezpośrednio od innych i konsekwentnie pomijały zainfekowaną wiadomość.

Przypadek (b) dotyczy wystąpienia *awarii*. Mogą zajść dwa różne scenariusze. *Awaria* wystąpiła jeszcze przed wysłaniem lub już po wysłaniu wartości początkowej. Na rysunku widać pierwszy z nich. Węzeł 1 ze względu na *awarię* nie był w stanie wysłać informacji o swojej wartości początkowej do pozostałych węzłów. To spowodowało, że wszystkie wyliczone sumy były błędne. W przypadku udanego wysłania choćby jednej wiadomości z wartością początkową przez węzeł 1 jej odbiorca rozesłałby ją do pozostałych i sumy zostałyby wyznaczone prawidłowo (z wyjątkiem uszkodzonego węzła 1). Drugi scenariusz polega na tym, że *awaria* ma miejsce już po wysłaniu wiadomości innym węzłom. Wtedy wystarczy jedno udane przesłanie i sumy w pozostałych węzłach będą prawidłowe.

Z uwagi na redundancję struktury pełnej i bezpośrednio związaną z tym jej odpornością na błędy *wstawienia* czy *pominięcia* nie były brane pod uwagę te przypadki. Zgodnie z przyjętym algorytmem *wstawienie* jest generowane już po wysłaniu wartości początkowych wszystkim sąsiadom, a więc już po odebraniu wiadomości od każdego węzła. Wstawienie generuje dodatkową wiadomość, która zgodnie z algorytmem zostanie pominięta. Podobnie błąd *pominięcia* nie zakłóci wymiany informacji, gdyż wiadomość pominięta zostanie dostarczona przez inny węzeł. W ten sposób redundancja połączeń w strukturze *SPE* całkowicie uodparnia jej działanie na błędy *wstawienia* czy *pominięcia* w przyjętym schemacie procesu propagacyjnego.

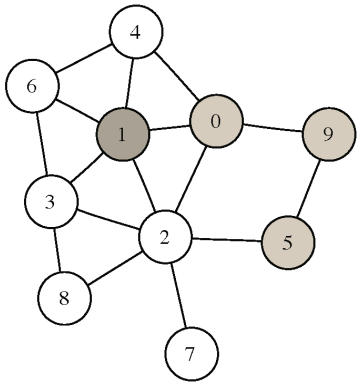



Rys. 4.4. Wizualizacja w strukturze *SPR*: a) *wstawienia*, b) *pominięcia*

Na rysunku 4.4 przedstawiono ścieżki propagacyjne w strukturze przypadkowej dla błędu *wstawienia* (a) i błędu *pominięcia* (b).

W przypadku (a) źródłem błędu *wstawienia* jest węzeł 5, a węzły 6, 3, 7 i 9 tworzą ścieżkę propagacyjną spowodowaną wysłaniem dodatkowej wiadomości. Proces infekcji przebiegał w następujący sposób: węzeł 5 wysłał fałszywą wiadomość od przypadkowego węzła do węzła 6. Tym przypadkowym węzłem mógł być jeden z listy: 0, 1, 2, 4. Węzły te nie mają bezpośredniego połączenia z węzłem 6. Fałszywa wiadomość musiała dotrzeć wcześniej niż wiadomość początkowa pochodząca od tego węzła, gdyż w przeciwnym razie zostałaby odrzucona. Węzeł 6 propagował błędną wiadomość kolejno dalej do węzłów 3, 7 i 9. Węzeł 1 nie został zarażony błędem, bowiem otrzymał wcześniej oryginalną wiadomość przed tą fałszywą, przysłaną przez węzeł 3. Podobnie węzły 0, 1, 2, 4 i 8 wyznaczyły prawidłowe sumy, ponieważ odebrały niezmienione wartości początkowe od swoich bezpośrednich sąsiadów. Najważniejszym wnioskiem, jaki można wyciągnąć na podstawie tego przypadku, jest to, że mechanizm propagacyjny ściśle zależy od kolejności wysyłania i odbierania wiadomości, a więc w rezultacie od momentu komunikacji z innymi elementami sieci.

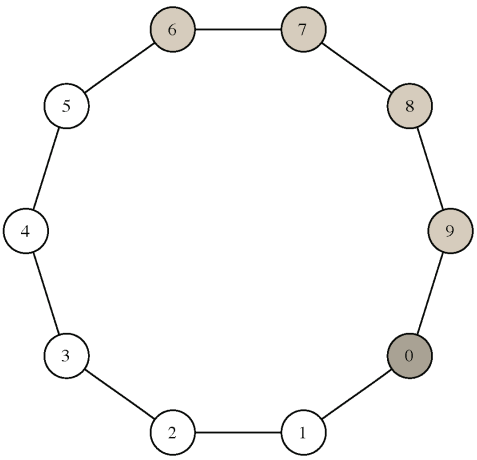
Przypadek (b) dotyczy błędu *pominięcia* w strukturze przypadkowej. Węzeł 0 nie wysyła propagowanej wiadomości do węzła 5, co w rezultacie wpływa także negatywnie na stan węzła 6. Dlatego że węzeł 0 odgrywa rolę pośredniczącą, każda wiadomość dla węzłów 5 i 6 od innego elementu struktury musi przejść przez ten węzeł. W rozpatrywanym przypadku węzeł 0 pomija jedną wiadomość przeznaczoną dla węzła 5, np. od węzła 4. Brak redundantnego połączenia węzłów 5 i 6 z innymi elementami poza węzłem 0 powoduje końcowy błąd wyznaczania sumy w tych węzłach. Węzły 2, 3 i 9 otrzymały wiadomości od węzła 4, gdyż pominięcie zostało wygenerowane na węźle 0 już po przekazaniu im tej wiadomości. Jak widać, nawet częściowe uszkodzenie działania jednego elementu pośredniczącego może skutecznie zakłócić prawidłowe funkcjonowanie struktury sieciowej.




Rys. 4.5. Wizualizacja *zanieczyszczenia* w strukturze *SPR*

Na rysunku 4.5 przedstawiono ścieżkę propagacyjną w strukturze przypadkowej dla błędu *zanieczyszczenia*. Przypadek ten jest tak samo groźny dla działania sieci. Z rysunku wynika, że węzeł 1 jest źródłem *zanieczyszczenia* wiadomości, np. z węzła 3 wysyłanej do węzła 0, który dalej propaguje błąd do węzła 9 i 5. Tak jak w przypadku błędu *wstawienia* zasadnicze znaczenie odegrał parametr czasu pojawienia się uszkodzenia. Warto zauważyć, że węzeł 2 odebrał niezmienioną wiadomość początkową i jego końcowy stan jest prawidłowy. Ponadto musiał później odebrać zanieczyszczoną wiadomość od węzła 0, ale mógł ją pominąć. Ścieżka propagacyjna (0–9–5) musiała jednak być krótsza niż ścieżka (2–5), gdyż węzeł 5 został zainfekowany błędem. Wynika to z tego, że węzeł 2 ma sześciu sąsiadów i wysłanie każdemu wiadomości trwa dłużej niż w przypadku węzła 9, który ma tylko dwóch sąsiadów.

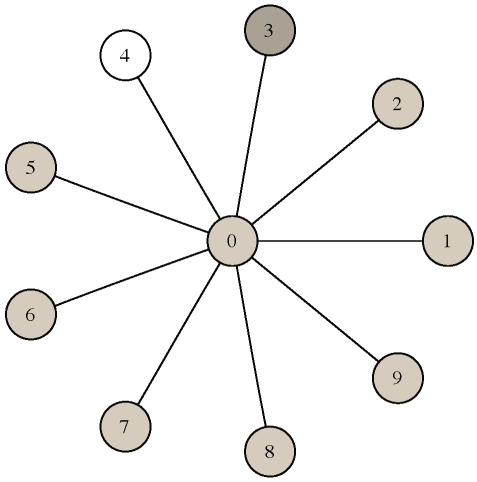
Przebieg procesu propagacyjnego awarii w strukturze *SPR* jest bardzo podobny do dystrybucji tego typu uszkodzenia w strukturze *SPE*. Czasami jednak różni się, gdyż możliwa jest sytuacja podziału struktury *SPR* na izolowane komponenty, które błędnie wyznaczą sumy wartości początkowych. Dodatkowo mogą wystąpić jeszcze dwa inne przypadki. Pierwszy z jednym elementem błędnym, który uległ uszkodzeniu, lub z zainfekowaną całą strukturą w drugim przypadku. Podobnie jak poprzednio o wielkości odsetka błędów będzie decydować moment wystąpienia awarii.




Rys. 4.6. Wizualizacja *wstawienia* lub *zanieczyszczenia* w strukturze *SPI*

Na rysunku 4.6 zobrazowano *wstawienie* lub *zanieczyszczenie* w strukturze pierścieniowej. Błąd jest generowany przez węzeł 0 na węzły 9, 8, 7 i 6. Propagacja jest zatrzymana w węźle 6, gdyż węzeł 5 odebrał wcześniej poprawną wiadomość od węzła 4 i pominął wiadomość przysłaną mu przez węzeł 6. Pominięcie w strukturze *SPI* jest całkowicie bezpieczne, ponieważ istnieje alternatywna droga dostarczenia prawidłowej wiadomości. Z kolei *awaria* w strukturze *SPI* jest podobna do awarii w struktu-

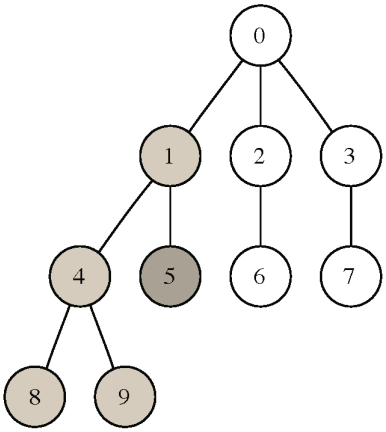
rze *SPE*. W przypadku wystąpienia *awarii* przed wysłaniem wszystkich wartości początkowych wyliczone sumy będą błędne. W przeciwnym przypadku, jeśli awaria wystąpi po wysłaniu wartości początkowej, jedynie uszkodzony węzeł nie będzie miał prawidłowo wyznaczonej sumy.




Rys. 4.7. Wizualizacja *wstawienia* w strukturze *SG*

Przedstawiona na rysunku 4.7 wizualizacja dotyczy struktury gwiaździstej i błędu *wstawienia*. Węzeł 3 generuje błąd. Fałszywa wiadomość, rzekomo pochodząca od węzła 4, trafia do węzła 0, a ten rozsyła ją do pozostałych węzłów. W ten sposób wszystkie węzły są zainfekowane błędem z wyjątkiem węzła 4, który pomija tę wiadomość. Tak jak poprzednio, gdyby błąd został wygenerowany już po przesłaniu poprawnej wartości początkowej z węzła 4 do węzła 0, propagacja nie przyniosłaby żadnych negatywnych skutków, gdyż fałszywa wiadomość zostałaby zablokowana przez węzeł centralny.

Przebieg procesu propagacyjnego w strukturze *SG* zależy od miejsca pojawienia się błędu. Paradoksalnie, jeśli błąd *wstawienia* będzie generowany przez węzeł centralny, szkody nie będą wielkie, ponieważ w przyjętym algorytmie wybierany jest tylko jeden węzeł peryferyjny, do którego wysyłany jest błąd. Pozostałe węzły odbierają prawidłowe wartości początkowe. Inaczej struktura zachowuje się w przypadku *awarii*. Jeśli *awaria* dotyczy węzła centralnego, cała struktura przestanie działać. Taki sam rezultat uzyskamy, gdy *awaria* dotyczy węzła peryferyjnego jeszcze przed wysłaniem wartości początkowej. Wtedy także wszystkie pozostałe węzły pokażą błędne sumy wartości początkowych. Gdy awaria nastąpi już po przekazaniu wartości początkowej, nie będzie ona miała żadnego wpływu na działanie pozostałych elementów systemu.




Rys. 4.8. Wizualizacja *wstawienia* w strukturze *SD*

Na rysunku 4.8 przedstawiono zachowanie struktury *SD*. Przypadek ten pokazuje, jak błąd *wstawienia* propaguje się w strukturze drzewiastej. Na rysunku widać, że węzeł 5 jest źródłem błędu, a węzły 1, 4, 8 i 9 zostały zainfekowane. Wygenerowano fałszywą wiadomość, rzekomo pochodzącą od węzła 0. Błąd rozprzestrzeniał się najpierw w górę hierarchii (węzeł 1), a potem w dół (węzły 4, 8 i 9). Dalsza propagacja w górę hierarchii jest zatrzymana, dlatego że węzeł 0 zablokuje fałszywą wiadomość. W strukturze drzewiastej pozbawionej redundancji połączeń propagacja błędów jest praktycznie niczym nieograniczona, zależy jednak od miejsca wystąpienia błędu. Im wyżej w hierarchii struktury znajduje się węzeł generujący błędy, tym większe szkody może poczynić. Podobnie duży wpływ na wielkość infekcji ma położenie fałszywego nadawcy w generowanej wiadomości w stosunku do źródła błędu.

### C. Analiza statystyczna uzyskanych wyników

Przeprowadzona w poprzednich rozdziałach analiza ilościowa i jakościowa uzyskanych wyników nie jest wystarczająca, żeby przyjąć albo odrzucić założoną na początku hipotezę o wpływie typu struktury topologicznej, typu i momentu wystąpienia błędu na sposób i rozmiar propagacji błędów. Analiza ilościowa przede wszystkim oparta została na średniej arytmetycznej odsetka błędów, podczas gdy analiza jakościowa wzięła pod uwagę kilka wybranych rezultatów w postaci stanów końcowych węzłów zliczających sumy. Aby można było zaakceptować wyciągnięte z analiz wnioski, wymagane są dalsze badania.

Duża liczba przeprowadzonych testów pozwala na weryfikację statystyczną otrzymanych wyników. Najbardziej odpowiednimi metodami statystycznymi są w rozpatrywanym przypadku testowanie hipotez i analiza zależności populacji [70]. Pierwsza

metoda pozwala na akceptację lub odrzucenie hipotezy o wpływie typu struktury topologicznej na proces propagacji błędów przy ustalonym poziomie istotności testu. Druga metoda bada zależności zmiennych ilościowych, dzięki czemu możemy porównywać dane wynikowe zebrane w czasie symulacji niezależnie od zaimplementowanej struktury topologicznej.

Testowanie hipotez jest jedną z metod wnioskowania statystycznego [129]. W ogólnym przypadku formułuje się dwie hipotezy, z których pierwsza – hipoteza zerowa ($H_0$) podlega weryfikacji i może zostać odrzucona na korzyść drugiej – alternatywnej ($H_1$). Następnie przyjmuje poziom istotności, dobiera statystykę testową i określa jej wartości prowadzące do przyjęcia lub odrzucenia $H_0$. W końcu sprawdza, czy wartość statystyki testowej odpowiada *p-wartości* pozwalającej odrzucić hipotezę zerową. Wybór odpowiedniej statystyki testowej jest bardzo ważny. Powinniśmy wziąć pod uwagę typ testu (np. parametryczny z określnym parametrem populacji lub nieparametryczny służący porównaniu właściwości wielu populacji), rozkład prawdopodobieństwa uzyskanych pomiarów i wielkość próby losowej.

Szczegółowy opis analizy statystycznej uzyskanych wyników przeprowadzonych symulacji został opublikowany w pracy [91]. Jak wynika z ogólnie przyjętego schematu postępowania, najpierw trzeba poznać rozkład prawdopodobieństwa testowanych populacji. Ponieważ ani histogram odsetka błędów, ani wykres kwantylowy nie wskazują na dobre dopasowanie zaobserwowanych wartości do rozkładu normalnego, przyjęto, że badane populacje nie mają takiego rozkładu. W zaistniałej sytuacji oparcie wnioskowania na teście nieparametrycznym jest uzasadnione. Do dalszej weryfikacji został wybrany rangowy test statystyczny Kruskala–Wallisa porównujący rozkłady dla badanych struktur topologicznych. Dzięki temu sprawdzono, czy struktura topologiczna wpływa na uzyskane wyniki. Zweryfikowano hipotezę zerową zakładającą, że rozkład prawdopodobieństwa pomiarów dla różnych struktur jest taki sam. Obliczona *p-wartość* przy założonym 5-procentowym poziomie istotności pozwala odrzucić $H_0$. Wyniki analizy pokazały, że struktura topologiczna wpływa statystycznie istotnie na poziom badanego odsetka błędów. Ponieważ test Kruskala–Wallisa nie jest dokładny, a wymagana była wiedza, które pary struktur topologicznych są podobne, dodatkowo zastosowano test Manna–Whitneya–Wilcoxona.

Tabela 4.5. *p-wartości* uzyskane dla par struktur topologicznych w teście Manna–Whitneya–Wilcoxona

| *TST* | *SPE* | *SPR* | *SPI* | *SG* | *SD* |
|---|---|---|---|---|---|
| *SPE* | x | $\ll$0,01 | $\ll$0,01 | $\ll$0,01 | $\ll$0,01 |
| *SPR* | $\ll$0,01 | x | $\ll$0,01 | 0,01 | $\ll$0,01 |
| *SPI* | $\ll$0,01 | $\ll$0,01 | x | 0,02 | $\ll$0,01 |
| *SG* | $\ll$0,01 | 0,01 | 0,02 | x | $\ll$0,01 |
| *SD* | $\ll$0,01 | $\ll$0,01 | $\ll$0,01 | $\ll$0,01 | x |

W tabeli 4.5 zawarto otrzymane *p-wartości* kolejno dla struktur: *SPE*, *SPR*, *SPI*, *SG* i *SD*. Wszystkie *p-wartości* są poniżej 0,05. Jedynie pary (*SG*, *SPR*) i (*SG*, *SPI*) są bliskie założonemu poziomowi istotności. Może to wskazywać na wzajemne większe podobieństwo tych struktur. Reasumując: pomiary wskazują na istniejące różnice w rozkładach prawdopodobieństw dla różnych topologii, co ostatecznie odrzuca przyjętą hipotezę zerową.

### 4.5.3. Wnioski z eksperymentów

Najważniejsze wnioski płynące z przeprowadzonych eksperymentów są potwierdzeniem, że przebieg i zakres propagacji błędów zależy przede wszystkim od struktury topologicznej systemu oraz typu i momentu wystąpienia błędu.

Po pierwsze, zbadane struktury można uporządkować od struktury najbardziej odpornej na błędy do najmniej odpornej: (1) struktura pełna *SPE*, (2) struktura przypadkowa *SPR*, (3) struktura pierścieniowa *SPI*, (4) struktura gwiaździsta *SG* i (5) struktura drzewiasta *SD*. Analiza jakościowa uzyskanych wyników dostarczyła szczegółowego wyjaśnienia przebiegu procesu propagacyjnego. Najbardziej odporną na uszkodzenia okazała się struktura pełna. Struktura przypadkowa działa trochę lepiej niż pierścieniowa z uwagi na mniejsze znaczenie węzłów pośredniczących. Propagacja w strukturze pierścieniowej powoduje większy odsetek błędów niż w strukturze przypadkowej. Najmniej odporne okazały się struktury nieredundantne, takie jak struktura gwiaździsta i drzewiasta. Pierwsza jest nieznacznie bardziej odporna m.in. ze względu na rolę, jaką pełni wyróżniony węzeł centralny w stosunku do wielu węzłów o podobnym znaczeniu w strukturze drzewiastej.

Po drugie, typy błędów występujących w systemach sieciowych można uporządkować od najmniej szkodliwego do najbardziej szkodliwego błędu: (1) *pominięcie*, (2) *wstawienie*, (3) *zanieczyszczenie* i (4) *awaria*. Błąd *pominięcia* może być łatwo zneutralizowany dzięki wprowadzeniu dodatkowych połączeń między wybranymi węzłami. Topologia pełna, przypadkowa i pierścieniowa są praktycznie całkowicie odporne na ten typ błędu. Utracona informacja po pojawieniu się tego błędu jest odzyskiwana z replikowanych danych przechowywanych w innych częściach systemu. W strukturach nieredundantnych informacja utracona nie może być odzyskana, gdyż nie została wcześniej zreplikowana. Pozostałe typy błędów są bardziej szkodliwe. Błąd *wstawienia* jest mniej szkodliwy niż błąd *zanieczyszczenia*, ponieważ wygenerowanie dodatkowej wiadomości łatwiej można wykryć niż w przypadku modyfikacji wiadomości, na którą czekamy. *Awaria* jest błędem, który powoduje zablokowanie działania pojedynczego elementu sieciowego. Mechanizm propagacji *awarii* na inne komponenty może jednakże spowodować wstrzymanie pracy nawet całego systemu.

Po trzecie, badania dotyczące momentu pojawienia się uszkodzenia i jego wpływu na mechanizm propagacyjny pokazały, że im wcześniej pojawi się błąd, tym więk-

szych szkód może dokonać. Widać więc, że wcześniejsze wykrycie pozwala na zablokowanie rozprzestrzeniania się uszkodzenia i uruchomienie procedury naprawczej. Na późniejszym etapie działania algorytmu wyznaczania sumy odrzucanie fałszywych wiadomości było bardziej prawdopodobne, gdyż węzły otrzymały wcześniej wiadomości prawidłowe.

Największym problemem w analizie symulacyjnej okazał się czas trwania eksperymentów. Mimo ograniczenia wielkości struktur do 100 komponentów i dobraniu prostego algorytmu propagacyjnego wszystkie testy zajęły prawie 200 h. Przyjęte parametry można krytykować. Uzyskane wyniki mimo wszystko jednoznacznie wskazują, że liczba komponentów miała niewielki wpływ na przebieg procesu propagacyjnego, a prostota algorytmu pozwoliła na jednoznaczną interpretację wyników.

Ponieważ struktury bezskalowe są w równej mierze wydajne jak struktury pełne [34], można więc implikować, że podobnie jak one charakteryzują się bardzo dobrą odpornością na błędy – ten fakt został niezależnie potwierdzony w pracy [48]. W przeciwieństwie do rozwiązań bezskalowych tzw. sieci małych światów sprzyjają szybkiej propagacji błędów. Można to zaobserwować w sieci społecznej utworzonej na bazie kontaktów SMS [101], w których dominują struktury gwiaździste i drzewiaste, czyli struktury mało odporne na procesy propagacyjne.

## 4.6. Podsumowanie

Problem propagacji błędów opisany w rozdz. 4 jest ważny w wielu obszarach zastosowań informatyki, m.in. takich jak: projektowanie systemów informacyjnych, implementacja protokołów sieciowych, ocena ryzyka w zakresie bezpieczeństwa systemów.

Przedstawione wyniki badań pozwalają na wybór najlepszej architektury systemu informatycznego oraz pokazują następstwa przyjętego rozwiązania. Można zatem wnioskować, że analiza procesu propagacji błędów umożliwia znalezienie sprawnych i odpornych na uszkodzenia metod na poziomie połączeń między różnymi komponentami, prowadzących z kolei do zmniejszenia negatywnych zachowań w użytkowanych systemach. Pierwszym rozwiązaniem, które nasuwa się po analizie struktur, jest możliwość dodawania lub usuwania bezpośrednich połączeń między węzłami. Inne rozwiązanie może polegać na wykorzystaniu kolaboracji w celu zatrzymania procesu propagującego błędy.

Na podstawie uzyskanych wyników wydawać by się mogło, że problem propagacji błędów został gruntownie przeanalizowany. Istnieje jednak przypadek wymagający dalszych badań. Dotyczy on nakładania się procesów propagacyjnych w podobny sposób, jak to zachodzi przy superpozycji niektórych wielkości fizycznych, np. interferencji fal akustycznych czy addytywności potencjału pola elektrycznego. W takiej

sytuacji możemy mówić o propagacji wielokrotnie złożonej. Przy sekwencyjnym wygenerowaniu kilku błędów efekt działania algorytmu WSE-B będzie niedeterministyczny. Zostaną obliczone różne sumy na węzłach w zależności od tego, który błąd pojawił się jako pierwszy. Także odsetek błędów ulegnie zmianie. Osobnym problemem jest analiza możliwych ścieżek propagacyjnych ściśle zależnych od typu pojawiających się błędów. Dla czterech rozpatrywanych typów, w wyniku propagacji wielokrotnie złożonej, ścieżki propagacyjne nie powinny odbiegać od przykładów podanych w niniejszym rozdziale. Za to faktycznym problemem może być identyfikacja uszkodzeń, ale to zagadnienie pominięto w prezentowanej pracy.

Dalsze badania w dziedzinie propagacji błędów mogą dotyczyć poszukiwania innych czynników mających wpływ na proces propagacyjny. Należy do nich na przykład rozszerzenie pojęcia odsetka błędów o przebieg ścieżek propagacyjnych. Wprowadzenie sterowanego mechanizmu wyboru sąsiada, któremu przesyłamy błąd może znacznie skrócić proces dystrybucji infekcji. Pewnym przybliżeniem tego zjawiska jest zagadnienie propagacji zadań opisane w rozdz. 3. Jeszcze innym kierunkiem przyszłych prac może być zbadanie korelacji między propagacją błędów a scenariuszami rozległych awarii czy przeprowadzonych ataków na elementy rzeczywistych systemów.

Wyniki przedstawione w rozdz. 4 zostały opublikowane w następujących pracach: [80], [82], [91].

# 5. Propagacja wiedzy

W rozdziale 5 przedstawiono analizę porównawczą komunikacji pośredniej i bezpośredniej w procesie propagacji wiedzy na platformie systemu wieloagentowego. Wykorzystanie w metodzie propagacyjnej stygmergii, pojęcia inspirowanego przyrodą, okazało się lepszym rozwiązaniem niż stosowanie tradycyjnej wymiany komunikatów między komponentami systemu rozproszonego, mimo wprowadzenia zindywidualizowanych parametrów progu, selektywności, trwałości i propagacyjności dla każdego węzła. Na podstawie wykonanej analizy i przeprowadzonych badań symulacyjnych można stwierdzić, że propagacja pośrednia jest efektywną metodą dystrybucji informacji i w szczególności nadaje się do optymalizacji ruchu pojazdów, zarówno w procesie modernizacji sieci dróg, jak i sterowania sygnalizacją świetlną.

## 5.1. Przekazywanie wiedzy inspirowane przyrodą

Do tej pory koncentrowano się na propagacji danych w formie komunikacji bezpośredniej za pomocą przesyłanych wiadomości w postaci elementarnych danych (zob. rozdz. 4), złożonych struktur (zob. p. 2.4.1), a nawet kompletnych algorytmów (zob. rozdz. 3). W rozdziale 5 zostanie rozpatrzone podejście inspirowane przyrodą, w którym występują generalnie dwa przypadki. W pierwszym brana jest pod uwagę grupa osobników komunikująca się między sobą w sposób pośredni. W drugim grupa osobników stosująca metodę bezpośrednią do wzajemnej komunikacji.

Propagacja wiedzy to proces, w wyniku którego wiedza jest przekazywana innym podmiotom występującym w danym środowisku. Pominięte będzie formalne definiowanie, czym jest wiedza i jak może być zorganizowana, ponieważ istnieje dużo opracowań na ten temat. Zainteresowanie autora skupi się jedynie na sposobie jej przekazywania. W poprzednich rozdziałach głównym nośnikiem informacji między komponentami systemu sieciowego była komunikacja bezpośrednia bazująca na przesyłaniu wiadomości. Taki sposób interakcji zapewniał dużą szybkość oraz niezawodność działania. Czy podobną rolę może pełnić komunikacja pośrednia? W badaniach dotyczących klasycznych struktur rozproszonych wykazano, że im więcej pośredni-

ków znajduje się na trasie propagowanych wiadomości, tym większe jest prawdopodobieństwo wystąpienia błędu. Taka sytuacja sprzyjałaby częstszej awaryjności i mniejszej szybkości transmisji. W przypadku systemów inspirowanych przyrodą mamy jednak do czynienia z odmiennym paradygmatem, a przez to z innymi możliwościami optymalizacyjnymi.

Poszukiwaniem metaheurystyki inspirowanej przyrodą ośrodki badawcze zajmują się już kilkadziesiąt lat. Typowymi przykładami proponowanych algorytmów są: algorytmy genetyczne, algorytmy mrówkowe czy algorytmy optymalizacji rojem cząsteczek. Szczegółowy stan wiedzy na ten temat został zebrany w pracach [61], [130]. Na ich podstawie można wyciągnąć wniosek, że najlepsze pod względem wydajności algorytmy dla dużej liczby węzłów bazują na heurystyce Lin–Kernighan [54]. Problem integracji paradygmatu inteligencji stadnej z podejściem wieloagentowym, choć wielokrotnie opisany w literaturze, nie jest w pełni rozwiązany. Ciągle brakuje na przykład standardowej metodologii projektowania tego typu systemów. Szczególnie interesujący pozostaje aspekt propagacji wiedzy między współpracującymi osobnikami określonej populacji.

Pojęcie inteligencji stadnej, zwanej też inteligencją roju, zostało wprowadzone do dziedziny systemów sieciowych na podstawie obserwacji zachowania się mrówek, pszczół, ryb czy ptaków – zorganizowanych w kasty, roje, ławice, gromady i stada [9], [117]. Najbardziej znanymi algorytmami zaproponowanymi w ramach metaheurystyk opartych na paradygmacie inteligencji stadnej są algorytmy mrówkowe i algorytmy optymalizacji rojem cząsteczek [36], [104], [141]. W środowisku naturalnym mrówki nie komunikują się między sobą w bezpośredni sposób. Nie operują na pojedynczym rozwiązaniu, ale na zbiorze rozwiązań. Używana jednak przez nie metoda identyfikowania na przykład najkrótszych dróg jest prosta i bardzo skuteczna, gdyż praktycznie nie zależy od stopnia złożoności problemu. W przypadku pszczół, ryb i ptaków komunikacja ma charakter bezpośredni i opiera się przede wszystkim na relacji sąsiedztwa [45].

Tak jak dla propagacji bezpośredniej podstawą jest relacja sąsiedztwa między krawędziami (wierzchołkami) grafu, tak dla propagacji pośredniej taką rolę pełni mechanizm stygmergii [41], [53], [119]. Jest to mechanizm zbiorowej komunikacjii pomiędzy użytkownikami wybranego środowiska, który umożliwia zmiany odczytywane następnie przez innych użytkowników. Środowisko stało się więc nośnikiem przekazu informacji stymulującym użytkowników do działania. Aby mimo wszystko przekaz informacji był możliwy, użytkownicy muszą być wyposażeni w możliwości oddziaływania na środowisko i odczytywania jego stanu. Pierwszym cytowanym przypadkiem stygmergii naturalnej występującej w przyrodzie było zachowanie termitów i mrówek. Także człowiek, często nieświadomie, wykorzystuje mechanizm stygmergii, np. tworząc graffiti i nielegalne wysypiska śmieci. Dodatkowo możemy zaobserwować stygmergię w świecie wirtualnym, np. w Wikipedii, w Second Life czy w repozytoriach plików otwartego oprogramowania.

Inspiracją dla systemów mrówkowych było poszukiwanie. Poruszające się mrówki pozostawiają feromon na swojej ścieżce, który w miarę upływu czasu odparowuje. Inne osobniki korzystają z tego śladu i wybierają drogę z największą ilością feromonu. W ten sposób kolektywnie konstruowana jest najkrótsza droga od gniazda do pożywienia. Deneubourg pierwszy zauważył tę umiejętność i opisał zjawisko wyboru krótszej ścieżki, z istniejących dwóch, na przykładzie mrówki argentyńskiej [36]. Pierwsza praca jednak prezentująca wykorzystanie algorytmu mrówkowego w rozwiązywaniu problemu komiwojażera pojawiła się dopiero 7 lat później [39]. Opis formalny modelu feromonowego można znaleźć m.in. w publikacji [17].

Na poziomie implementacyjnym, podobnie jak w przypadku propagacji zadań, przychodzącym na myśl rozwiązaniem jest zastosowanie systemu wieloagentowego. Kwestią integracji paradygmatu inteligencji stadnej z systemem wieloagentowym zajmowało się kilku autorów. Stali oni przed problemem wyboru odpowiedniej metodologii projektowania systemu wieloagentowego, która byłaby w stanie wykorzystać atuty komunikacji pośredniej lub bezpośredniej. Choć istnieje wiele różnych metodologii projektowania dla systemów wieloagentowych, np. AOSE, GAIA, Tropos i MaSE [21], [24], [133], do prezentowanych badań wybrana została ostatnia z nich. Głównym powodem takiego wyboru było przekonanie o odpowiednim dopasowaniu tej metodologii do systemów wieloagentowych bazujących na inteligencji stadnej, co zostało pokazane i zweryfikowane m.in. w pracy [79]. Prototypy systemów używanych w kolejnych eksperymentach zostały zaprojektowane zgodnie z cytowaną autorską metodologią MaSE.

W rozdziale 5 skupiono uwagę na problemie propagacji wiedzy opierającej się zarówno na klasycznej wymianie komunikatów, jak i na mechanizmach naśladujących naturę. W dalszej części przedstawiono, jak można wykorzystać mechanizm propagacji wiedzy do analizy i optymalizacji ruchu drogowego. Na przykładzie systemu sterowania sygnalizacją świetlną opisane zostały wady i zalety procesu propagacji pośredniej w porównaniu do propagacji bezpośredniej bazującej na wymianie komunikatów. Na końcu rozdziału umieszczono krótkie podsumowanie.

## 5.2. Propagacja pośrednia w analizie ruchu drogowego

Badania symulacyjne z wykorzystaniem systemu wieloagentowego z inteligencją stadną będą przedmiotem dalszych rozważań w tym rozdziale. Na podstawie rzeczywistych danych otrzymanych z Zarządu Dróg i Utrzymania Miasta we Wrocławiu (ZDiUM) zostanie wygenerowana i poddana analizie przechodząca przez centrum miasta tranzytowa sieć komunikacyjna obejmująca drogi krajowe i wojewódzkie miasta

Wrocławia. W opracowanym modelu wyróżniono trzy podstawowe elementy: sieć dróg krajowych i wojewódzkich, poruszanie się pojazdów na drodze, zmianę świateł na skrzyżowaniach i wpływ tej zmiany na ruch pojazdów. Podstawowe pytanie, na które poszukiwano odpowiedzi, brzmiało: czy można zidentyfikować krytyczne punkty (odcinki) w sieci komunikacyjnej miasta i wprowadzić takie, by uzyskać zwiększenie przepustowości ruchu drogowego? Do tego typu zmian zaliczamy m.in. konfigurowanie pasów ruchu, ustalanie długości czasu trwania kolejnych faz sygnalizacji świetlnej na skrzyżowaniach czy też poprawianie topologii połączeń w sieci dróg.

W dalszej części pracy zostanie przedstawiony model ruchu drogowego i środowisko badań symulacyjnych do optymalizacji ruchu pojazdów, następnie – omówione wyniki uzyskane dla zaprojektowanego algorytmu optymalizacyjnego, a na koniec zostaną wyciągnięte wnioski i podane kierunki kolejnych badań w tym zakresie.

## 5.2.1. Ogólny model ruchu drogowego

W dziedzinie modelowania ruchu drogowego do opisu procesów dynamicznych w systemach symulacji ruchu samochodowego [108] i pieszego [144] najczęściej używa się automaty komórkowe. Do podstawowych zalet automatów komórkowych można zaliczyć: dobre właściwości obliczeniowe, a przez to wysoką efektywność, niewielkie obciążanie pamięci, możliwość równoległego wykonywania procesów obliczeniowych, w końcu prostą implementację. Automaty komórkowe dzięki swoim zaletom są stosowane także w innych dziedzinach nauki. Wśród wielu przykładów wymienia się modelowanie takich zjawisk, jak: przepływ cząsteczek, mieszanie substancji płynnych, turbulencja smugi dymu, płynąca lawa, pożar lasu, rozprzestrzenianie się epidemii czy generowanie kluczy publicznych w systemach kryptograficznych.

Do symulacji jednopasmowej i jednokierunkowej drogi można wykorzystać automat komórkowy z regułą Wolframa 184 [151]. W omawianym przypadku wszystkie pojazdy poruszają się w tym samym kierunku, przesuwając się jednocześnie o co najwyżej jedną pozycję. Ten jednak prosty automat komórkowy nie uwzględniał podstawowych elementów, m.in. sygnalizacji świetlnej czy skrzyżowania dróg. Dużo lepszym rozwiązaniem jest propozycja automatów komórkowych Nagela–Schreckenberga, w którym ze stanu zajętości komórek wyznaczane są podstawowe parametry ruchu pojazdów, np. prędkość przejazdu przez skrzyżowanie, czas oczekiwania na przejazd, przepustowość skrzyżowania. Ponadto pojazdy mogą przyspieszać i zwalniać, uwzględniono też elementarne zdarzenia losowe występujące na drodze. Podstawowy model Nagela–Schreckenberga był wielokrotnie ulepszany, by lepiej modelować rzeczywiste zdarzenia na drodze, dodano obsługę wielu pasów ruchu [73]. Przy opisie założeń modelu ruchu drogowego z propagacją pośrednią będzie można zauważyć duże podobieństwo do reguł stosowanych przez adekwatne modele komórkowe.

Na podstawie publikacji [79] oraz [87] można stwierdzić, że najpopularniejszą platformą implementacyjną dla systemów optymalizacji ruchu drogowego są środowiska wieloagentowe.

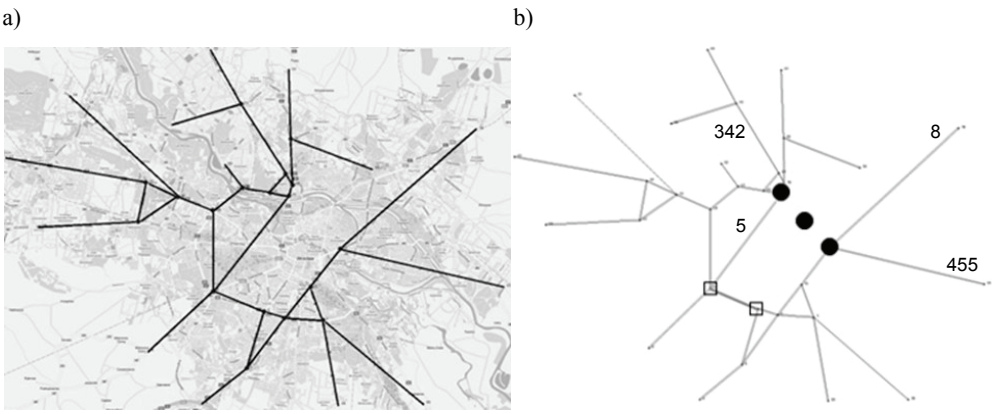
## 5.2.2. Model ruchu drogowego z propagacją pośrednią

Podstawowymi elementami modelu ruchu drogowego są następujące obiekty: trasy (drogi), pasy ruchu, skrzyżowania, systemy sygnalizacji świetlnej oraz pojazdy. Każda trasa składa się z uporządkowanej listy pasów ruchu, po których poruszają się pojazdy. Dodatkowo zawiera 24-elementową tablicę definiującą liczbę pojazdów generowanych na tej trasie w poszczególnych godzinach w ciągu doby. Pas ruchu ma jednoznacznie określony kierunek, zgodnie nim poruszają się pojazdy, oraz długość definiowaną od skrzyżowania początkowego do końcowego. Skrzyżowanie może łączyć kilka dróg. Wyróżniamy następujące rodzaje skrzyżowań: brzegowe z przypisaną jedną drogą, przelotowe z dwiema przypisanymi drogami i standardowe – łączące trzy lub więcej dróg. Dla sygnalizacji świetlnej związanej ze skrzyżowaniami określono minimalny i maksymalny czas trwania sygnału zielonego oddzielnie dla każdego pasu ruchu oraz dopuszczalny interwał zmiany czasu trwania sygnału. Wybrano sygnał zielony, gdyż zmiana wielkości czasu trwania tego sygnału pozwala sterować liczbą pojazdów przejeżdżających przez skrzyżowanie. Liczebność pojazdów generowana jest zgodnie z danymi przechowywanymi w elemencie droga. Przykładowo, jeśli dla określonej trasy należy w pierwszej godzinie wygenerować 3600 pojazdów, jeden pojazd będzie dodawany do pierwszego pasa ruchu tej trasy co 1 s. Wymienione założenia pozwalają na konstrukcję ważonego i skierowanego grafu sieci komunikacyjnej, którego wierzchołkami są skrzyżowania, a krawędziami pasy ruchu. Wagą krawędzi jest długość pasa ruchu początkowego i końcowego obliczana na podstawie podanych lokalizacji skrzyżowań.

Najważniejszymi parametrami opisującymi ruch drogowy poza standardową prędkością pojazdu są jeszcze dwa pojęcia: natężenie oraz przepływ. Pierwsze z nich określa liczbę pojazdów na danym odcinku drogi, np. [veh/km], drugie – liczbę pojazdów w jednostce czasu, np. [veh/h]. W opisywanych badaniach wykorzystano prędkość pojazdu oraz natężenie ruchu. Przyjęto uproszczoną definicję natężenia ruchu będącą stosunkiem liczby pojazdów aktualnie znajdujących się na pasie ruchu do maksymalnej liczby pojazdów, które może pomieścić dany pas drogi. W celu lepszej czytelności wyników w badaniach zastosowano jeszcze jedno uproszczenie – zamiast przepływu przyjęto czas podróży.

Żeby zrealizować ideę polegającą na wprowadzeniu zdecentralizowanego i efektywnego zarządzania ruchem pojazdów wykorzystano algorytm mrówkowy – zgodnie z nim pojazdy pozostawiają feromon w zależności od tego, jak długo czekają na przejazd przez skrzyżowanie. To właśnie poziom feromonu pośrednio odpowiada za stero-

wanie ruchem pojazdów, gdyż zależność między czasem oczekiwania na sygnał zielony a koncentracją feromonu wpływa na działanie sygnalizacji świetlnej, więc w rezultacie na ruch pojazdów. Podobne rozwiązanie zaproponowano w pracy [6] i [122].




Rys. 5.1. Rzeczywista sieć dróg tranzytowych uwzględniona w badaniach symulacyjnych

W modelu zostały uwzględnione jedynie te drogi krajowe i wojewódzkie, które przechodzą przez centrum miasta. Uznano, że pozostałe drogi mają zdecydowanie mniejszy wpływ na sumaryczną wielkość ruchu pojazdów w mieście. W badaniach uwzględniono drogi krajowe: 5, 8 i 94 oraz drogi wojewódzkie: 320, 322, 327, 336, 337, 342, 347, 349, 356, 362, 395, 452, 453 i 455 (patrz rys. 5.1a). Ponadto dodano połączenie między skrzyżowaniem drogi krajowej nr 8 z drogą wojewódzką 455 oraz skrzyżowaniem drogi krajowej 5 z drogą wojewódzką 342. Proponowana w eksperymencie nowa droga tranzytowa oznaczona została czarnymi kropkami, a odcinek testowy służący za punkt odniesienia przy interpretacji wyników – dwoma kwadratami (patrz rys. 5.1b).

Na poziomie implementacyjnym sieć dróg tranzytowych została zrealizowana jako skierowany, ważony graf, którego wierzchołkami są skrzyżowania, a krawędziami pasy ruchu. Waga krawędzi zależy od długości pasa ruchu i jest obliczana na podstawie odległości skrzyżowania początkowego od końcowego. Najczęściej spotykanymi typami skrzyżowań w mieście są skrzyżowania trzech lub czterech ulic. Poza tym ruch na skrzyżowaniach jest rozdzielany zgodnie z predefiniowanymi pasami ruchu. W ten sposób agenty współpracujące z sygnalizacją świetlną mogą decydować, jak długo sygnał zielony zezwala na jazdę wybranym pasem ruchu w danym kierunku.

W modelu przyjęto następujące ograniczenia dotyczące ruchu pojazdów:

- pojazdy poruszają się po drodze nie szybciej niż pozwalają na to ograniczenia prędkości;

- prędkość pojazdu $v$ zawsze jest dostosowana do innych pojazdów z zachowaniem umownego bezpiecznego odstępu $ods_{\text{bez}}$ określanego w następujący sposób: dla prędkości pojazdu mniejszej od 10 km/h bezpieczny odstęp wynosi 1 m, dla prędkości mniejszej od 20 km/h odpowiednio – 2 m itd.

Ruch pojazdów jest generowany w kilku krokach. W pierwszym kroku dodawane są pojazdy przejeżdżające przez miasto w ruchu tranzytowym. Następnie obliczana jest różnica występująca między wartością rzeczywistą zmierzoną na drodze przez ZDiUM i aktualnym natężeniem ruchu na tym samym odcinku drogi, o tej samej porze dnia. Uzyskanie wartości dodatniej implikuje konieczność wygenerowania większego natężenia ruchu dla tego fragmentu sieci tranzytowej, z kolei uzyskanie wartości ujemnej powoduje wstrzymanie procesu generowania nowych pojazdów i skierowanie istniejących na inne trasy.

Kluczowym parametrem algorytmu mrówkowego jest poziom feromonu $\tau$ pozostawiany przez pojazdy. W rozpatrywanym przypadku ilość pozostawianego feromonu $\tau$ jest odwrotnie proporcjonalna do czasu potrzebnego na przejazd pojazdu $t_{pps}$ przez najbliższe skrzyżowanie. W ten sposób niski poziom feromonu wskazuje na krótki czas oczekiwania $t_{ops}$ i w rezultacie szybki przejazd przez skrzyżowanie. Podczas działania algorytmu obliczana jest średnia arytmetyczna śladu feromonowego na drogach. Jeżeli samochód napotka ślad odbiegający od średniej o zadaną wartość, np. ustaloną procentowo, może zmienić swoją trasę pod warunkiem, że wybrana droga również doprowadzi go do pierwotnie wskazanego punktu docelowego. Ponieważ założono, że każdy pojazd pozostawia ślad feromonowy w zależności od czasu oczekiwania na przejazd, więc uaktualnienie poziomu feromonu odbywa się każdorazowo przed wjazdem na skrzyżowanie. Zmianę poziomu feromonu można obliczyć zgodnie ze wzorem

$$\Delta t = \frac{\rho_{wf}}{t_{pps}} \tag{5.1}$$

w którym: $\Delta t$ – ilość feromonu wygenerowana przez pojedynczy pojazd, $t_{pps}$ – czas przejazdu przez skrzyżowanie, $\rho_{wf}$ – współczynnik wzmocnienia śladu feromonowego, $0 < \rho_{wf} < 1$.

Ze względu na występowanie parowania po zakończeniu każdej iteracji symulacji poziom feromonu $\tau_1$ w chwili $t_1$ jest określany następująco:

$$\tau_1 = \tau_0 - (t_1 - t_0) \cdot \rho_{pf} \tag{5.2}$$

$\tau_1$, $\tau_0$ – poziom feromonu odpowiednio w chwili $t_1$, $t_0$, $(t_1 - t_0)$ – jednostka czasu między kolejnymi aktualizacjami poziomu feromonu, $\rho_{pf}$ – współczynnik parowania feromonu, $0 < \rho_{pf} < 1$.

Współczynniki wzmocnienia śladu i parowania feromonu były dobierane empirycznie z uwzględnieniem wyników eksperymentów opublikowanych w pracy [40].

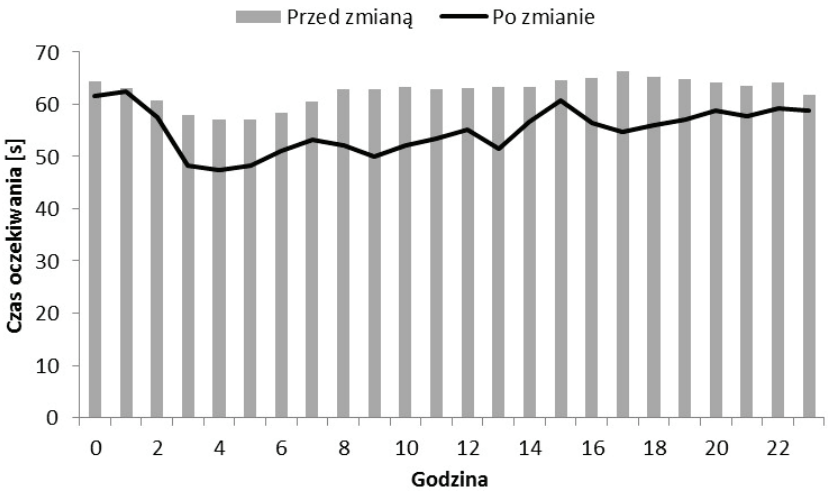
## 5.2.3. Analiza symulacyjna procesu propagacyjnego

Model został zaimplementowany na platformie JADE, w której skrzyżowania działają jako autonomiczne agenty. Mogą oni decydować o odpowiedniej kombinacji świateł oraz długości trwania pełnego cyklu zmian świateł, np. przez modyfikację czasu trwania sygnału zielonego.

Badania symulacyjne zostały przeprowadzone dwuetapowo. Pierwsza część badań dotyczyła optymalizacji ruchu na istniejącej sieci dróg tranzytowych. W drugiej – zaproponowano wyznaczenie nowej drogi tranzytowej, tj. brakującego odcinka do utworzenia wewnętrznej obwodnicy miasta. Celem tych badań było sprawdzenie, czy dodanie nowego, wewnętrznego połączenia pozwoli na obniżenie natężenia ruchu w innej części istniejącej sieci dróg. W obu przypadkach wykorzystano algorytm mrówkowy. Badania zostały przeprowadzone na komputerze z procesorem Centrino Duo 2.0 GHz i pamięcią 1 GB RAM.

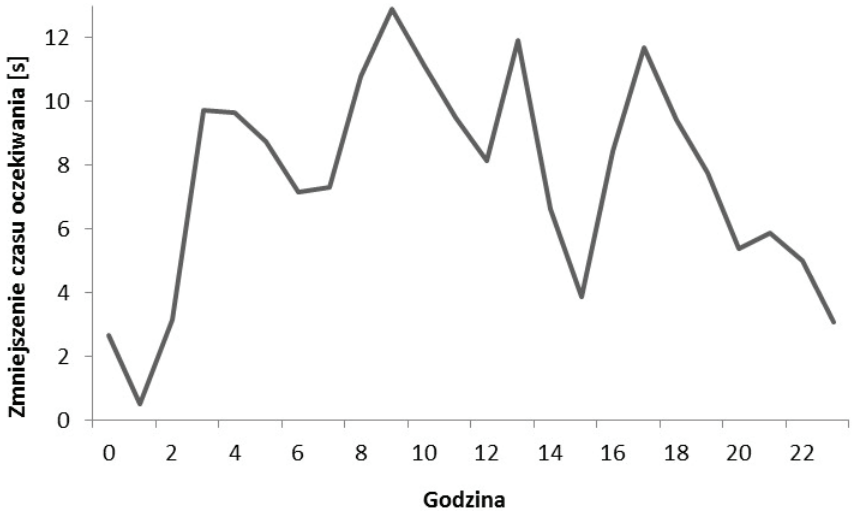
### A. Optymalizacja czasu oczekiwania istniejącej sieci dróg tranzytowych

Badania, które podjęto, pozwalają na wyciągnięcie wniosków dotyczących efektywności zastosowania algorytmu mrówkowego do optymalizacji czasu oczekiwania pojazdu na przejazd przez skrzyżowanie. Do testów wybrano Rondo Ronalda Reagana, ze względu na jego strategiczne położenie na skrzyżowaniu dróg tranzytowych oraz pełnienie roli jednego z najważniejszych węzłów komunikacyjnych miasta.




Rys. 5.2. Średni czas oczekiwania pojazdów na przejazd przez skrzyżowanie

Porównanie średniego czasu oczekiwania (patrz rys. 5.2) pozwala ocenić, czy zastosowane rozwiązanie przynosi dobre rezultaty. Średni czas oczekiwania na przejazd został skrócony o około 7,5 s. Co więcej, wartość osiągniętego zysku nie rozkłada się równomiernie w ciągu doby. Uzależnione jest to od wielkości natężenia ruchu oraz od współczynników wzmocnienia śladu i parowania feromonu.




Rys. 5.3. Zmniejszenia czasu oczekiwania pojazdów po zmianie

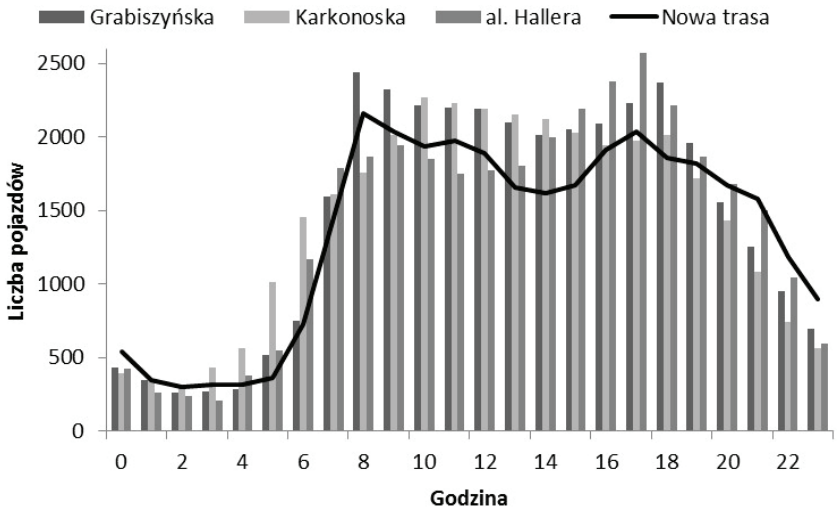
Podobnie jak poprzednio, wyniki przedstawione na rys. 5.3 świadczą o dużym zróżnicowaniu w ciągu dnia. Wieczorem i w porze nocnej zmniejszenie średniego czasu oczekiwania jest minimalne. Wynika to z tego, że podczas szczytu komunikacyjnego pojazdy pozostawiły wysoki poziom feromonu na trasach tranzytowych, który implikował wydłużony czas trwania sygnału zielonego w tych kierunkach. Pojazdy poruszające się w innych kierunkach nie mogły szybko wpłynąć na zmianę trwania sygnału zielonego, gdyż było ich za mało. Wtedy zasadniczą rolę odgrywają współczynniki wzmocnienia śladu i parowania feromonu. Można odczytać z wykresu, że dopiero po paru godzinach (około 2 w nocy) udało się dostosować czas trwania sygnału zielonego do rzeczywistych warunków natężenia ruchu drogowego. Test powtarzano wielokrotnie dla różnych wartości współczynników. Prezentowane na wykresach wyniki były najlepsze, jakie udało się uzyskać.

**B. Wyznaczenie nowej drogi tranzytowej**

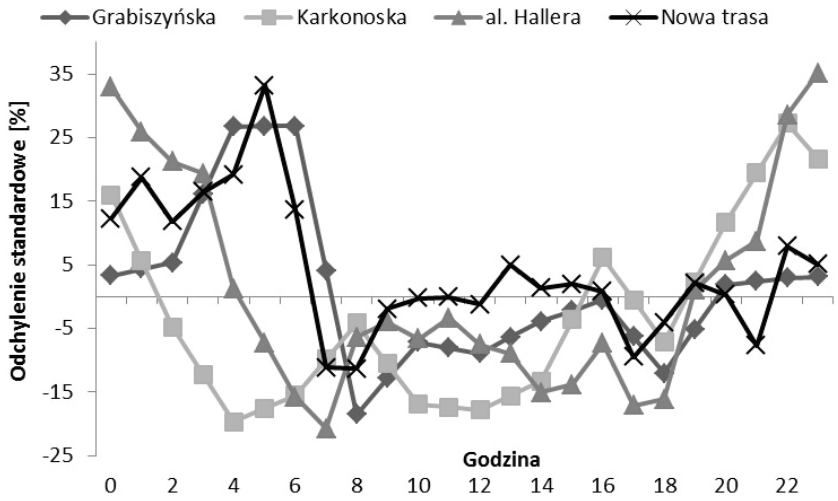
Sieć dróg tranzytowych przebiegających przez miasto jest dość słabo rozwinięta. Ponadto brakuje bezpośredniego połączenia części wschodniej miasta z północną. Dodanie połączenia w tym obszarze miasta powinno znacznie usprawnić ruch tranzytowy

z północy na wschód i na południe. W drugiej części badań początkowy schemat sieci dróg został więc rozszerzony o nową trasę.

Aby można było ocenić przyjęte rozwiązanie, pomiary symulacyjne natężenia ruchu pojazdów dokonano zarówno na wybranych istniejących odcinkach dróg, jak i na proponowanym nowym połączeniu (patrz rys. 5.4). Analiza wykrresu prowadzi do wniosku, że algorytm mrówkowy efektywnie wykorzystał nowe połączenie. Liczby pojazdów na wszystkich trasach są wzajemnie porównywalne.

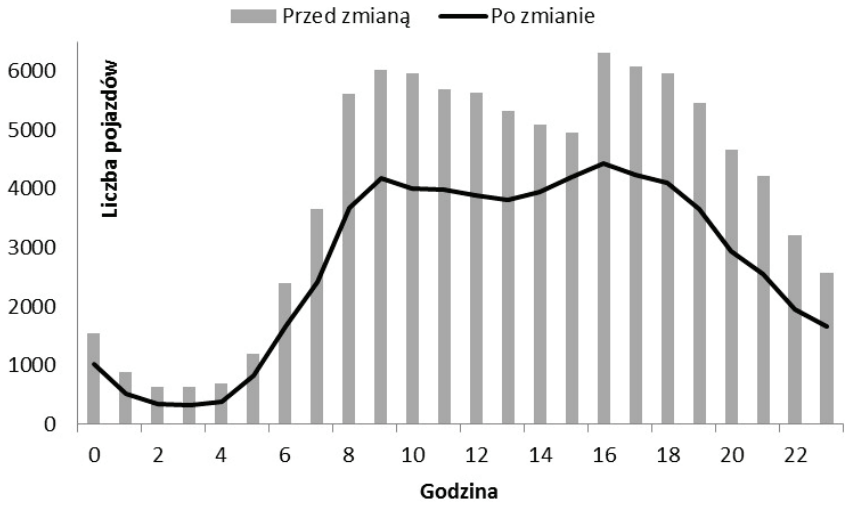



Rys. 5.4. Natężenie ruchu pojazdów na wybranych trasach

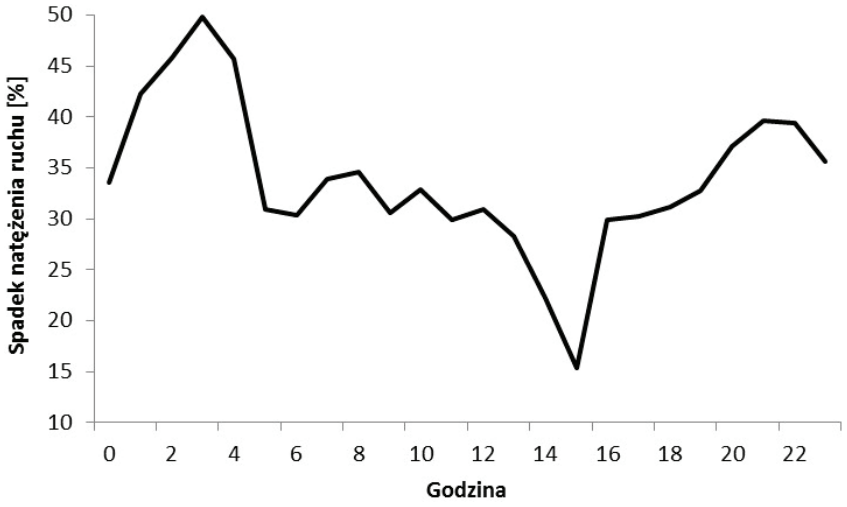



Rys. 5.5. Odchylenie standardowe natężenia ruchu pojazdów na wybranych trasach

Na rysunku 5.5 przedstawiono fluktuacje natężenia ruchu na wybranych trasach w zależności od pory dnia – między godziną 8 a 18 różnice te są niewielkie i wahają się maksymalnie do 15%, natomiast między 19 a 7 rano znacząco odbiegają od średniej. Na obu wykresach charakterystyki natężenia ruchu pojazdów na nowej trasie nie odbiegają od innych analizowanych tras. Oznacza to, że nowa trasa powinna odegrać rolę równie ważnego i uczęszczanego szlaku komunikacyjnego w mieście.




Rys. 5.6. Liczba pojazdów przed i po wyznaczeniu nowego połączenia




Rys. 5.7. Zmniejszenie natężenia ruchu pojazdów po wyznaczeniu nowego połączenia

Aby ocenić wpływ wprowadzonej zmiany na ruch pojazdów w mieście, należało zaobserwować występujące różnice natężenia po dodaniu nowego połączenia. Na podstawie analizy danych rzeczywistych wybrano w tym celu odcinek wzdłuż al. Hallera. Charakteryzuje go największe w mieście zarejestrowane natężenie ruchu drogowego. Z analizy rysunku 5.6 można wnioskować, że liczba pojazdów znacząco zmniejszyła się.

Procentowy obraz natężenia ruchu, widoczny na rys. 5.7, jeszcze dobitniej pokazuje zmniejszenie liczby pojazdów na najbardziej obciążonej trasie (al. Hallera). Jak można było się spodziewać, redukcja liczby pojazdów w ciągu doby nie jest rozłożona równomiernie. Największy spadek liczby pojazdów zaobserwowano w godzinach wieczornych i nocnych. Porównując z pierwotnym układem dróg, uzyskany wynik – średnio około 33% zmniejszenia wielkości natężenia ruchu pojazdów – jest dobrym prognostykiem dla efektywnej implementacji takiej zmiany w warunkach rzeczywistych.

### 5.2.4. Zalety i wady modelu z propagacją pośrednią

Model sieciowy wykorzystujący propagację pośrednią należy do tzw. podejścia makroskopowego i bardzo dobrze sprawdził się w zakresie modelowania ruchu drogowego. Uzyskane wyniki świadczą o tym, że przyjęte rozwiązania implementacyjne nadają się do efektywnej optymalizacji ruchu pojazdów zarówno w procesie sterowania sygnalizacją świetlną, jak i modernizacji sieci dróg. Nieodzowne jest jednak wskazanie w tym miejscu nie tylko zalet, ale także istniejących wad proponowanego rozwiązania.

Zacząć warto od zalet użytkowych, które wcześniej nie zostały wymienione. To, co zwykle nastręcza dużo trudności, czyli rozrost i modyfikacje sieci połączeń, w przypadku propagacji pośredniej nie powoduje widocznych zmian w działaniu systemu.

W ogólnym przypadku centralnym punktem badań nad systemami z propagacją pośrednią były trzy parametry: złożoność komunikacyjna, czas zakończenia działania algorytmu oraz transfer danych [27], [56]. W przypadku modelu ruchu drogowego zbierane przez agenty dane o natężeniu ruchu, czasie oczekiwania czy liczbie pojazdów poruszających się w określonym kierunku pozwalają na skuteczną predykcję tych parametrów także na nowych planowanych odcinkach dróg. Kolejną zaletą jest duża skalowalność algorytmu propagacyjnego oraz możliwość wykorzystania odpornych na błędy struktur topologicznych przy projektowaniu i implementacji tego typu systemów.

Podstawową wadą modeli ruchu drogowego jest niekompletność używanych sieci komunikacyjnych i przyjętych uproszczeń dotyczących ruchu pojazdów. Drogi krajowe oraz wojewódzkie, które wzięto pod uwagę w tym opracowaniu, nie odzwierciedlają w pełni panujących rzeczywistych warunków. Nie uwzględniono przecież wielu

skrzyżowań, które mimo lokalnego charakteru, mają duży wpływ na przemieszczanie się pojazdów. Bardzo często te skrzyżowania są sterowane sygnalizacją świetlną, co dodatkowo i niestety niezależnie od przyjętego rozwiązania wpływa na czas oczekiwania pojazdów i w rezultacie na natężenie ruchu. Dodatkowym mankamentem jest przyjęta stała prędkość poruszających się pojazdów.

Mimo iż osiągnięto skrócenie czasu oczekiwania na przejazd średnio o 7,5 s, co dało poprawę rzędu 12% w stosunku do stanu pierwotnego, nie są to liczby w pełni satysfakcjonujące. Uwidoczniły się dwie podstawowe przyczyny takiego stanu rzeczy. Po pierwsze, mimo wielu testów nie udało się uzyskać lepszych wartości dla współczynników wzmocnienia i parowania feromonu. Problem ten wymaga dalszych badań i zastosowania na przykład algorytmu genetycznego w celu optymalnego dobrania odpowiednich wartości. Po drugie, udostępnione dane przez ZDiUM o natężeniu ruchu pojazdów dotyczyły jedynie wybranych punktów w mieście, przez co nie można było zamodelować lokalnego ruchu pojazdów przecinającego kierunki tranzytowe. To z kolei ze względu na duże natężenie feromonu po szczycie komunikacyjnym powodowało opóźnienia w prawidłowym ustawianiu czasu trwania sygnału zielonego.

## 5.3. Porównanie propagacji pośredniej z propagacją bezpośrednią

Na podstawie wykonanej w poprzednim rozdziale analizy i badań symulacyjnych możemy stwierdzić, że propagacja pośrednia jest efektywną metodą dystrybucji informacji w środowisku rozproszonym. Wcześniej w przypadku dystrybucji zadań i propagacji błędów wykorzystaliśmy model propagacji bezpośredniej. W tym momencie nasuwa się pytanie, który model propagacyjny powinno się stosować – propagację pośrednią czy bezpośrednią. Z praktycznego punktu widzenia wydaje się, że zależy to od wielu czynników i nie znajdziemy jednej prostej odpowiedzi. Ponieważ uprzednio uwaga skupiona była na optymalizacji ruchu drogowego przy użyciu propagacji pośredniej, teraz sprawdzić trzeba, jak efektywny może być ten sam system przy zastosowaniu metody bezpośredniej.

Założenia przyjęte dla systemu sterowania sygnalizacją ruchu drogowego przy użyciu propagowanych wiadomości zostały zaczerpnięte z pracy [66]. Środowiskiem badań symulacyjnych pozostał system opisany w rozdz. 5.2. Wprowadzono jednak kilka zmian, by można było lepiej odzwierciedlić rzeczywiste warunki panujące na drogach. Najważniejszą modyfikacją jest umożliwienie dokonywania zmian prędkości poruszających się pojazdów. Pojazdy poruszają się zgodnie z ich uszeregowaniem na pasie ruchu. Dla każdego pojazdu obliczana jest prędkość, jaką mógłby osiągnąć na odcinku wolnej drogi, która znajduje się przed nim. Przyjęte uproszczenie dotyczące zachowania obiektów zakłada, że każdy pojazd ma długość 3 m, a minimalny odstęp

między nimi powinien wynosić 2 m. Podczas przeprowadzonych badań maksymalna prędkość pojazdów wynosiła 14 m/s. Ponadto każdy pojazd przechowywał następujący zestaw danych: całkowity czas podróży, czas oczekiwania na sygnał zielony, czas postoju w kolejce i liczbę zatrzymań pojazdu. Czas postoju pojazdu w kolejce był naliczany w sytuacji, kiedy generator chciał dodać kolejny pojazd, a na danym pasie ruchu nie było już miejsca, tzn. pojazdy znajdujące się na nim zajmowały całą jego długość.

Podobnie jak poprzednio, podstawowym celem badań było znalezienie takiego czasu trwania sygnału zielonego, aby czas podróży pojazdów był jak najkrótszy, a także jak najbardziej płynny. Płynność przejazdu głównie uzyskuje się przez minimalizację liczby zatrzymań pojazdów oraz jednoczesne zmniejszanie czasu oczekiwania przed skrzyżowaniami.

### 5.3.1. Model ruchu drogowego z propagacją bezpośrednią

Model ten różni się od poprzedniego przede wszystkim metodą komunikacji. Zamiast komunikacji feromonowej, która decyduje o ewentualnej zmianie trasy w przypadku przekroczenia progu koncentracji feromonu, używany jest prosty system wysyłania wiadomości. Steruje on czasem trwania sygnału zielonego, co wpływa na prędkość zbliżających się do niego pojazdów. Prędkość tym razem może się zmieniać w zależności od wolnego miejsca na pasie ruchu oraz określonej maksymalnej wartości dopuszczalnej. Dodatkowo system został tak przebudowany, że pojazdy nie zmieniają początkowo wyznaczonej trasy.

Za wydłużanie lub skracanie czasu trwania światła zielonego odpowiadają sterowniki sygnalizacji świetlnej zainstalowane nad skrzyżowaniami. Każdy sterownik działa na podstawie czterech parametrów: *progu* (określa poziom reakcji na korek uliczny), *selektywności* (określa poziom reakcji na żądanie zmiany), *trwałości* (określa czas trwania ostatnio wprowadzonej zmiany) i *propagacyjności* (określa listę sterowników informowanych o zmianie). Wszystkie parametry zostały ujednolicone i przyjmują wartości całkowite z zakresu od 1 do 10.

Jeśli sterownik zaobserwuje natężenie pojazdów większe od *progu*, wysyła sąsiadującym sterownikom informację o konieczności wydłużenia długości trwania sygnału zielonego na łączącym je pasie dojazdowym. Zgodnie z parametrem *propagacyjności* wysyłana wiadomość powinna dotrzeć do sąsiadów na skrzyżowaniach stanowiących kontynuację trasy pojazdów z pasa ruchu, na którym przekroczony został *próg* natężenia. Przykładowo, wartość progu równa 5 oznacza, że do sąsiadujących sterowników zostanie wysłana wiadomość dopiero, gdy natężenie na pasie dojazdowym osiągnie 50%. Parametr *progu* może być postrzegany jako odpowiednik współczynnika wzmocnienia śladu feromonowego w algorytmie mrówkowym.

Za decyzję o zmianie czasu trwania sygnału zielonego odpowiada też parametr *selektywność*, określający stopień reakcji na żądanie zmiany od sąsiedniego sterownika. Innymi słowy, *selektywność* odpowiada stopniowi prawdopodobieństwa odpowiedzi na otrzymaną informację o zwiększonym natężeniu i potrzebie zmiany. Wartość *selektywności* wynosząca 10 oznacza, że każda informacja otrzymana od sąsiada będzie skutkowała wprowadzeniem zmiany. Odpowiednio zmniejszenie (osłabienie) wartości *selektywności* do 5 oznacza, że sterownik zareaguje na otrzymany komunikat z prawdopodobieństwem 50%.

Trzeci z listy parametrów jest dopełnieniem parametru *selektywności – trwałość* pozwala na automatyczne skracanie wcześniej wydłużonego czasu trwania sygnału zielonego po upływie minut zdefiniowanych w tym parametrze. Parametr *trwałości* pełni podobną rolę, jaką odgrywa współczynnik parowania feromonu w algorytmie mrówkowym. Powrót do stanu poprzedniego, sprzed wydłużenia, pozwala na reakcję systemu w przypadku zmniejszenia natężenia ruchu drogowego. Podobnie jak dwa poprzednie parametry przyjmuje wartości od 1 do 10, tym razem jednak wartość 1 odpowiada 5 min, a wartość 10 – 50 min. Każdorazowe wydłużenie czasu trwania sygnału zielonego powoduje automatyczne odliczanie pozostałego czasu do ewentualnego skrócenia od początku.

Ostatni parametr – *propagacyjność* decyduje o rozmiarze procesu propagacyjnego wysyłanych wiadomości informujących o wydłużeniu długości trwania sygnału zielonego. Jak tylko zostanie przekroczony parametr *progu* na danym pasie ruchu, automatycznie uruchamia się mechanizm wysyłania komunikatów. Można wyróżnić dwa typy wiadomości odpowiadające za przyspieszenie (w uproszczeniu oznaczającym wydłużenie trwania sygnału zielonego) lub za spowolnienie ruchu (w uproszczeniu – skrócenie trwania sygnału). Odebranie komunikatu nie obliguje sterownika do podjęcia natychmiastowego działania, zależy to przecież od indywidualnych ustawień i aktualnie panujących warunków. Sposób przyporządkowania wiadomości wartościom od 1 do 10 jest następujący: 1 oznacza przyspieszenie przekazane sterownikom oddalonym jednym odcinkiem drogi, 2 oznacza spowolnienie przekazane sterownikom oddalonym jednym odcinkiem drogi, 3 oznacza przyspieszenie przekazane sterownikom oddalonym dwoma odcinkami drogi itd. Propagowanie wiadomości jest ograniczone, tak by wiadomość nie wracała do swojego miejsca wygenerowania. W tym celu można wykorzystać mechanizm identyfikatorów. W omawianych badaniach nie posługiwano się tym parametrem, gdyż przygotowany algorytm wymiany komunikatów uwzględniał jedynie wydłużenie trwania sygnału zielonego oraz najbliższych sąsiadów, a więc domyślnie *propagacyjność* przyjmowała wartość równą 1.

Kiedy wiadomość zostanie odebrana przez sterownik sygnalizacji świetlnej, wtedy, to co nastąpi później, zależy od wartości ustawionych w parametrach. W celu efektywnego działania modelu ruchu drogowego z propagacją bezpośrednią należało optymalnie dobrać wartości dla wszystkich wymienionych parametrów. Do realizacji tego zadania posłużył algorytm genetyczny [103], [72], [78]. Ponieważ problematyka

algorytmów genetycznych jest dobrze znana od kilkunastu lat niektóre szczegóły rozwiązania zostaną pominięte.

Ze względu na różnorodność cech skrzyżowań występujących w sieci dróg założono, że określanie wartości parametrów dla sterownika powinno mieć charakter indywidualny. Potencjalne rozwiązanie składa się więc z sekwencji trójek <*P*, *T*, *C*>: *P* oznacza *próg*, *T – trwałość* a *C – selektywność*. Długość sekwencji jest równa liczbie skrzyżowań. Przyjęto też, zgodnie z założeniami optymalizacji o usprawnieniu ruchu pojazdów, że funkcje celu i przystosowania będą uwzględniać średni czas podróży, średni czas postoju i średnią liczbę zatrzymań. Do wymienionych średnich dodano wagi, by można było podkreślić większe znaczenie liczby zatrzymań od czasu postoju i czasu podróży.

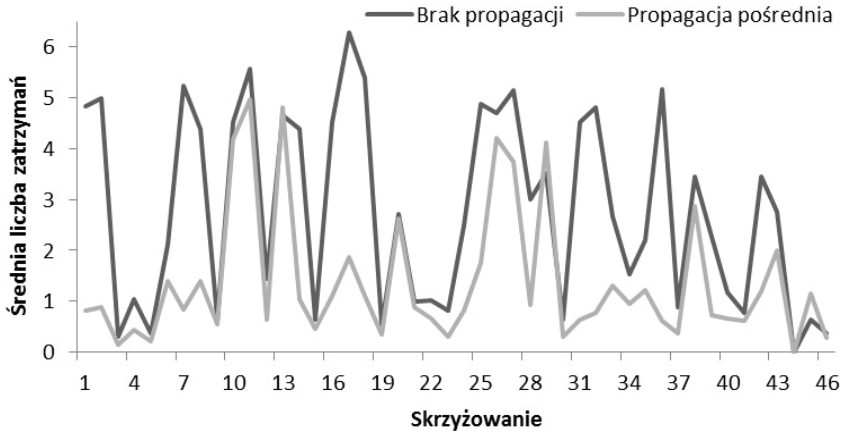
Zgodnie ze ogólnym schematem algorytmu genetycznego pozostały do ustalenia jeszcze trzy elementy: metoda reprodukcji, operatory genetyczne i metoda sukcesji. Na metodę reprodukcji wybrano selekcję proporcjonalną – tzw. mechanizm koła ruletki. Zbiór operatorów genetycznych tworzyły: krzyżowanie jednopunktowe, krzyżowanie zwane jednorodnym oraz wariant mutacji punktowej. Ostatni element – metoda sukcesji – wybierała najlepszego osobnika z populacji. Po przeprowadzeniu 50 iteracji wybrano najlepsze rozwiązanie, które wykorzystano do ostatecznej optymalizacji natężenia ruchu drogowego. Końcowe rozwiązanie o największej wartości funkcji przystosowania uzyskano na populacji składającej się z 50 osobników podczas dwugodzinnej symulacji.

### 5.3.2. Porównanie wyników badań symulacyjnych

W celu porównania wyników działania algorytmu z propagacją pośrednią i bezpośrednią opracowane zostały wykresy przedstawiające następujące zależności – średnią liczbę zatrzymań pojazdów, średni czas postoju oraz podróży pojazdów na każdej z 50 tras. Dane podsumowujące przebieg symulacji dla obu algorytmów zestawiono na końcu rozdziału w tab. 5.1.

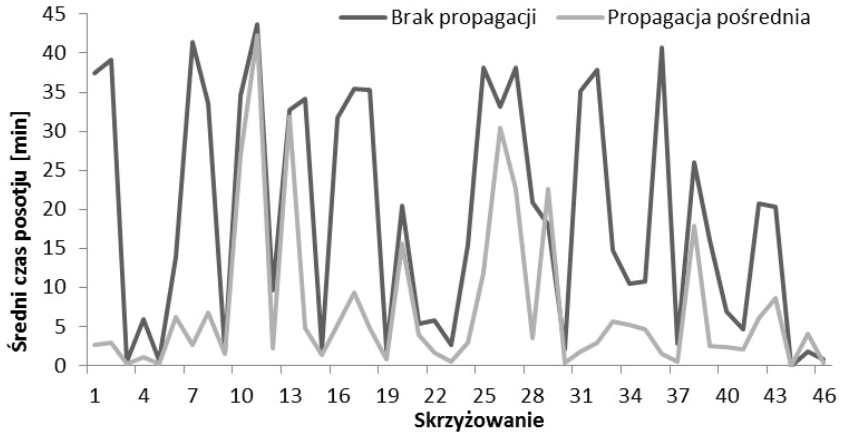
W badaniu porównawczym model ruchu drogowego składał się z 77 skrzyżowań, w tym z 12 skrzyżowań brzegowych, 10 skrzyżowań przelotowych, 13 skrzyżowań z trzema drogami dojazdowymi oraz 42 skrzyżowań z czterema drogami dojazdowymi połączonych 120 odcinkami dróg. Wprowadzono zestaw 50 tras przejazdowych, w tym 12 tras o charakterze tranzytowym oraz 38 tras o charakterze lokalnym. Trasy są różnej długości i mają określoną maksymalną dopuszczalną prędkość. Każda symulacja trwała 24 h. Liczba pojazdów generowanych na trasie w danej godzinie symulacji była losowana z przedziału (200, 500). Początkowa długość trwania sygnału zielonego na każdym pasie dojazdowym do skrzyżowania ustawiona była na 3 min.

### A. Wyniki badań dla propagacji pośredniej



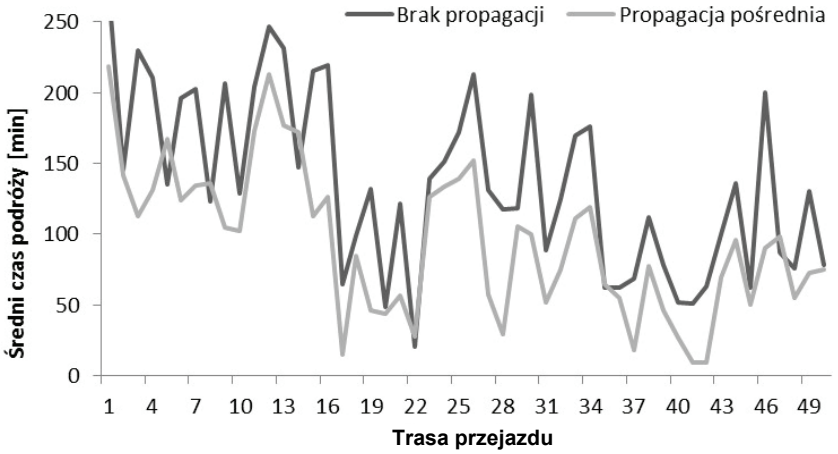

Rys. 5.8. Średnia liczba zatrzymań na pasach dojazdowych do skrzyżowań

W przypadku propagacji pośredniej (tak jak poprzednio wykorzystano algorytm mrówkowy), widać znaczną poprawę wszystkich obserwowanych parametrów ruchu. Suma średniej liczby zatrzymań pojazdów na pasach dojazdowych do poszczególnych skrzyżowań sterowanych algorytmem propagacyjnym w porównaniu z symulacją bez propagacji zmniejszyła się o 66 zatrzymań (patrz rys. 5.8).




Rys. 5.9. Średni czas postoju na pasach dojazdowych do skrzyżowań

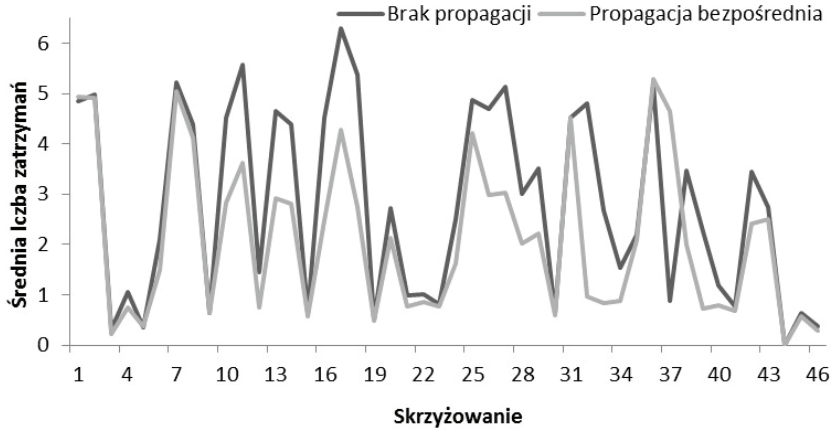
Podobnie jak na poprzednim wykresie linia odnosząca się do propagacji pośredniej znajduje się zazwyczaj poniżej linii symulacji bez propagacji. Suma średnich czasów postoju w środowisku propagacji pośredniej skróciła się o ok. 550 min (patrz rys. 5.9).




Rys. 5.10. Średni czas podróży na każdej z tras

Tymczasem suma średnich czasów podróży na wszystkich trasach zmniejszyła się o ok. 35 h (patrz rys. 5.10). Na uwagę zasługuje również fakt, że w czasie trwania symulacji z zastosowaniem algorytmu komunikacji pośredniej, swoją trasę ukończyło o 3919 pojazdów więcej niż podczas trwania symulacji bez żadnej optymalizacji. Oznaczało to znaczne zwiększenie przepustowości modelowanej sieci dróg.

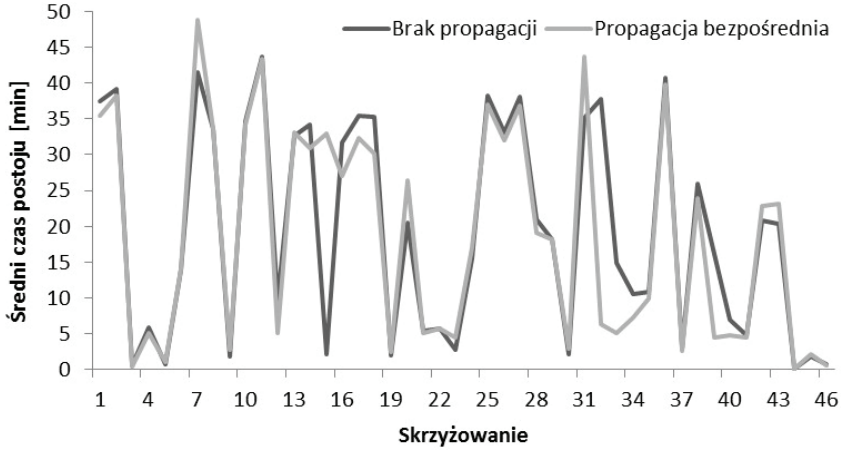
**B. Wyniki badań dla propagacji bezpośredniej**




Rys. 5.11. Średnia liczba zatrzymań na pasach dojazdowych do skrzyżowań

Na rysunku 5.11 linia odnosząca się do propagacji bezpośredniej także leży poniżej linii symulacji bez optymalizacji. W tym przypadku jednak różnica nie jest już tak widoczna. Suma średniej liczby zatrzymań pojazdów na pasach dojazdowych do po-

szczególnych skrzyżowań zmniejszyła się tylko o 32 zatrzymania. Dwa razy mniej niż w przypadku propagacji pośredniej.




Rys. 5.12. Średni czas postoju na pasach dojazdowych do skrzyżowań

Suma średnich czasów postoju pojazdów dla propagacji bezpośredniej skróciła się o ok. 31 min, podczas gdy w przypadku propagacji pośredniej zysk wynosił 550 min. Różnica linii symulacyjnych, widoczna na rys. 5.12, jest jeszcze mniejsza.


Rys. 5.13. Średni czas podróży na każdej z tras

Na rysunku 5.13, podobnie jak na poprzednim wykresie, obie linie symulacyjne różnią się już bardzo niewiele, mimo to suma średnich czasów podróży na wszystkich trasach zmniejszyła się o ok. 6 h. Choć w obu badanych przypadkach pojazdy zatrzy-

mywały się podczas jazdy podobną liczbę razy, ich sumaryczne czasy postoju oraz czasy podróży nadal znacznie różniły się. Pojazdy musiały więc poruszać się z większą prędkością, by wypracować taki zysk czasowy.

### C. Zestawienie wyników

Pełne zestawienie liczby pojazdów, które ukończyły wytyczoną trasę, liczby zmian czasu trwania sygnału zielonego oraz wartości średnich przedstawiono w tab. 5.1. Na jej podstawie można wywnioskować, że propagacja pośrednia okazała się dużo bardziej skuteczną metodą niż propagacja bezpośrednia. Wartości dla czterech z podanych w tabeli parametrów są zdecydowanie lepsze dla propagacji pośredniej. Jedynie średnia liczba zatrzymań była nieznacznie gorsza. Nie wpłynęło to jednak ani na średni czas podróży, ani na liczbę pojazdów, które ukończyły wytyczoną trasę.

Tabela 5.1. Zestawienie porównawcze wyników symulacji

| Kryterium | Brak propagacji | Propagacja bezpośrednia | Propagacja pośrednia |
|---|---|---|---|
| Liczba pojazdów, które ukończyły wytyczoną trasę | 33 286 | 34 723 | 37 205 |
| Liczba zmian czasu trwania sygnału zielonego na godzinę [liczba/h] | 0 | 5484 | 1043 |
| Średni czas podróży [min] | 102 | 86 | 64 |
| Średnia liczba zatrzymań | 9,6 | 5,7 | 6,2 |
| Średni czas postoju [min] | 63 | 54 | 36 |

Przy porównaniu opisanych metod propagacji warto zwrócić uwagę na problem możliwości implementacyjnych każdego z tych mechanizmów w rzeczywistym systemie. I tak, w przypadku metody opartej na algorytmie mrówkowym problemem technicznym jest sposób, w jaki pojazdy miałyby informować system o czasie swojego postoju na określonym pasie ruchu w celu aktualizacji ilości feromonu. Może to być realizowane na przykład przez dedykowane urządzenie. Póki co rozwiązanie to jednak wydaje się być kosztowne. W przypadku algorytmu opartego na komunikacji bezpośredniej istnieją już efektywne systemy określające liczby pojazdów na poszczególnych pasach ruchu i tego typu rozwiązania są z powodzeniem stosowane w warunkach rzeczywistych, np. w Londynie, Atenach, Pekinie i Hongkongu [66].

## 5.4. Podsumowanie

Zaproponowane algorytmy optymalizacji ruchu oparte na propagacji wiedzy między zarządzanymi skrzyżowaniami okazały się rozwiązaniami efektywnymi. Porów-

nanie metody pośredniej bazującej na algorytmie mrówkowym i bezpośredniej z wymianą komunikatów wypadło zdecydowanie na korzyść tej pierwszej. Mimo wprowadzenia dokładniejszej, zindywidualizowanej do każdego skrzyżowania informacji wstępnej obejmującej: *próg*, *selektywność* i *trwałość*, metoda bazująca jedynie na wielkości feromonu pozostawianego przez pojazdy okazała się lepsza.

Zarówno w pierwszym, jak i drugim przypadku mechanizm propagacyjny może być rozszerzony na większą liczbę sterowników sygnalizacji świetlnej (np. zgodnie z wyznaczonymi trasami tranzytowymi), tworząc tzw. zielone fale. Problemem do rozwiązania pozostaje konflikt propagacyjny wielu zielonych fal, które spotykają się przy tym samym skrzyżowaniu.

Ponieważ oba rozwiązania optymalizacyjne opierają się na propagacji wiedzy o zagęszczeniu pojazdów, zastosowanie bardziej zaawansowanego fizycznego modelu ruchu drogowego nie powinno spowodować zmian w działaniu algorytmów. Wprowadzenie możliwości dodawania dróg zawierających więcej niż dwa pasy ruchu, umożliwienie wzajemnego wyprzedzania się, uwzględnienie pojazdów specjalnych czy też możliwość modelowania ronda zasadniczo nie wpłyną na zmniejszenie efektywności procesu propagacyjnego.

Mimo korzystnych rezultatów działania metod propagacyjnych i uzyskania skrócenia czasu oczekiwania/postoju pojazdów krytycznie należy podsumować propozycję modelowania ruchu drogowego w podany sposób. Przedstawiona koncepcja jest tylko przybliżeniem rzeczywistego złożonego procesu, który w warunkach symulacyjnych działa prawidłowo. Nie wzięto jednak pod uwagę wielu występujących utrudnień na drodze w postaci na przykład przeszkody na pasie ruchu, zwężenia drogi, wpływu długości pasa czy łuku na prędkość skręcających pojazdów. Wielość i różnorodność czynników bezpośrednio wpływających na przemieszczanie się pojazdów powoduje dużą złożoność proponowanych rozwiązań. W takiej sytuacji najczęściej stosowanym modelem jest tzw. podejście mikroskopowe bazujące na automatach komórkowych [108]. Wydaje się, że połączenie podejścia makroskopowego i mikroskopowego może przynieść jeszcze lepsze wyniki, co skłania do prowadzenia dalszych badań w tym zakresie.

Celem opisanych badań nie było opracowanie modelu jak najwierniej odzwierciedlającego rzeczywistość, a jedynie pokazanie, że używając nawet bardzo uproszczonego modelu z funkcją propagacji pośredniej zaimplementowanej w postaci algorytmu mrówkowego lub propagacji bezpośredniej z wymianą komunikatów, można szybko uzyskać satysfakcjonujące wyniki. Ponadto algorytm propagacji pośredniej okazał się bardzo skuteczny zarówno pod względem czasu działania, jak i wyszukanego rozwiązania.

Część badań przedstawionych w rozdz. 5 została opublikowana w następujących pracach: [79], [86], [87].

# 6. Uwagi końcowe

W prezentowanej pracy: *Wybrane metody propagacji danych w systemach rozproszonych*: (a) sformułowano podstawowe typy propagacji zadań, (b) zbadano wpływ parametrów sterujących procesu propagacji danych na przyspieszenie wykonywania zadania, (c) zdefiniowano pojęcie progu opłacalności, które skutecznie ogranicza negatywne zjawisko cyklu propagacyjnego, (d) zbadano zależność przebiegu i zakresu propagacji błędów od struktury topologicznej systemu oraz typu i momentu wystąpienia błędu, (e) opracowano metodę określania parametrów: *progu*, *selektywności*, *trwałości* i *propagacyjności* dla komunikujących się węzłów w modelu ruchu drogowego, (f) w zakresie sterowania ruchem pojazdów zaproponowano model wykorzystujący propagację pośrednią bazującą na komunikacji feromonowej, (g) pokazano, w jaki sposób propagacyjna metoda pośrednia może być lepsza niż propagacyjna metoda bezpośrednia.

Realizacja wszystkich wymienionych punktów wniosła niekwestionowany wkład do dziedziny systemów rozproszonych w kontekście metod propagacji danych. Uzyskane wyniki pozwoliły autorowi potwierdzić tezę, zgodnie z którą metody propagacji danych mogą być skutecznym mechanizmem dystrybucji zadań, analizy błędów i integracji wiedzy w nowoczesnych systemach sieciowych.

Zaprezentowanie metod propagacji danych stosowanych do rozwiązywania wybranych zadań kolektywnej inteligencji we współczesnych systemach rozproszonych było podstawowym celem niniejszej publikacji. Zamiarem autora było uporządkowanie i poszerzenie wiedzy w odniesieniu do mechanizmów propagacyjnych zwykle dość pobieżnie reprezentowanych w literaturze, a równocześnie mających istotne znaczenie dla rozwoju systemów rozproszonych, szczególnie w sferze ich zastosowań. Po wnikliwej analizie podanego obszaru badawczego uwaga została skoncentrowana na trzech utylitarnych metodach propagacji danych: propagacji zadań, propagacji błędów i propagacji wiedzy. Najważniejsze wyniki dotyczą głównie propozycji związanej z efektywnym wykorzystywaniem metod propagacji bezpośredniej i pośredniej. Poszukiwanie rozwiązań prowadzone było z zastosowaniem metod algorytmicznych, w tym algorytmów inspirowanych przyrodą. Wszystkie badania zostały zweryfikowane eksperymentalnie.

Zastosowanie procesu propagacji danych jest jednym z najważniejszych kwestii w dziedzinie systemów rozproszonych, a efektywne rozwiązanie tego problemu ma

wiele aspektów praktycznych. Istotnym problemem, przed którym stoją użytkownicy systemów informatycznych, jest to, w jaki sposób można wykorzystać nie tylko ogromne zasoby, przede wszystkim obliczeniowe i magazynowe, ale także, jak spożytkować wiedzę i doświadczenia samych użytkowników. Na przykład wykorzystanie istniejących komputerów może polegać na coraz popularniejszym udostępnianiu ich przez wirtualizację w chmurze obliczeniowej. W procesie przeniesienia oprogramowania do chmury, jego użytkowania oraz pełnej personalizacji dużą rolę może pełnić odpowiedni mechanizm propagacyjny. Innym palącym problemem jest poszukiwanie nowych, łatwo modyfikowalnych, ale i odpornych na błędy struktur sieciowych. W tym przypadku dużą rolę może odegrać metoda propagacji pośredniej, często znacznie efektywniejsza niż metoda bezpośrednia. Prezentowana praca stanowi wkład do poszukiwania nowych rozwiązań w tych obszarach. Wykorzystanie mechanizmów propagacyjnych w poszukiwaniu eksperta odpowiadającego na zapytania w systemie społecznym czy w uodparnianiu na błędy systemów sieciowych już obecnie jest w pełni realne.

W podsumowaniu należy podkreślić, że najważniejszymi uzyskanymi wynikami, decydującymi o wartości teoretycznej i praktycznej prezentowanej publikacji – zdaniem autora – są:

1. Dowiedzenie, w ramach opracowanych rozwiązań, że uzyskane przyspieszenie wykonania zadania za pomocą mechanizmu propagacyjnego potwierdza hipotezę o zysku, jaki może wnieść ten mechanizm przy wykonywaniu rozproszonych obliczeń. Zasadniczym parametrem sterującym okazał się jednostkowy rozmiar zadania, który każdorazowo należy dopasowywać do środowiska uruchomieniowego. Optymalnym momentem równowagi było otrzymanie podobnego czasu zakończenia pracy przez wszystkie węzły obliczeniowe przy równoczesnej minimalizacji kosztów transmisji. Wprowadzono oryginalną, autorską taksonomię metod propagacji zadań: migrację, replikację i promocję. Najefektywniejszą metodą propagacji zadań okazała się promocja, pozbawiona kosztu związanego z serializacją danych.
2. Zdefiniowanie pojęcia progu opłacalności oraz opracowanie strategii limitowania procesu propagacyjnego, ponieważ często w procesie propagacji zadań pojawia się negatywne zjawisko cyklu propagacyjnego. Oba wymienione rozwiązania pozwalają na ograniczenie niepotrzebnych i kosztownych operacji w systemie rozproszonym.
3. Znalezienie równowagi między wyrównywaniem mocy węzłów obliczeniowych przez zwiększanie liczby zadań a ograniczaniem propagacji i występowania cykli przez zmniejszanie tej samej liczby realizowanych zadań stanowiących podstawowy problem przy propagacji zadań. Najprostszym rozwiązaniem, a zarazem bardzo skutecznym, okazało się dynamiczne dodawanie lub usuwanie zadań w wybranych węzłach.
4. Przeprowadzenie analizy propagacji błędów w środowisku rozproszonym. Wykonane eksperymenty symulacyjne pokazały, że przebieg i zakres propagacji

błędów zależy przede wszystkim od struktury topologicznej systemu oraz typu i momentu wystąpienia błędu. I tak, rozpatrując klasyczne topologie sieci, najbardziej odporną strukturą na błędy, jak można było się spodziewać, okazała się najbardziej redundantna struktura pełna. Z kolei najmniej odporne są struktury nieredundantne, takie jak – struktura gwiaździsta i drzewiasta. Zdefiniowane w pracy typy błędów występujących w systemach sieciowych można uporządkować od najmniej szkodliwego do najbardziej szkodliwego błędu w następujący sposób: *pominięcie*, *wstawienie*, *zanieczyszczenie* i na ogół fatalna w skutkach *awaria*. Ponadto badania dotyczące momentu pojawienia się błędu i jego wpływu na mechanizm propagacyjny potwierdziły, że im wcześniej zostanie wygenerowany błąd tym większych szkód może dokonać, o ile nie zostanie wcześniej wykryty i zneutralizowany.

5. Zdefiniowanie następujących parametrów dla komunikujących się węzłów w modelu ruchu drogowego: *próg*, *selektywność*, *trwałość* i *propagacyjność*. Pokazano, że uzyskanie zadawalających wyników działania systemu ruchu drogowego z propagacją bezpośrednią jest możliwe, o ile zostaną dobrane odpowiednie wartości dla wymienionych parametrów. W tym celu można posłużyć się na przykład algorytmem genetycznym.
6. Zaproponowanie modelu rozproszonego wykorzystującego propagację pośrednią bazującą na komunikacji feromonowej, który bardzo dobrze sprawdził się w zakresie symulowania ruchu drogowego. Przyjęte rozwiązania implementacyjne nadają się do efektywnej optymalizacji ruchu pojazdów, zarówno w procesie modernizacji sieci dróg, jak i sterowania sygnalizacją świetlną. To, co zwykle nastręcza dużo trudności, czyli rozrost i modyfikacja sieci połączeń, w przypadku propagacji pośredniej nie powoduje widocznych zmian w działaniu systemu.
7. Opracowanie metody optymalizacyjnej ruchu pojazdów opartej na propagacji wiedzy między zarządzanymi skrzyżowaniami, która okazała się rozwiązaniem efektywnym, choć nie uwzględnia wszystkich rzeczywistych zdarzeń mogących pojawić się na drodze.
8. Porównanie metody pośredniej opierającej się na algorytmie mrówkowym i bezpośredniej z wymianą komunikatów wypadło zdecydowanie na korzyść tej pierwszej. Mimo uwzględnienia w metodzie bezpośredniej szczegółowej informacji obejmującej *próg*, *selektywność* i *trwałość*, dodatkowo zindywidualizowanej do każdego skrzyżowania, metoda pośrednia, bazująca jedynie na wielkości feromonu pozostawianego przez pojazdy, okazała się dużo lepsza.

W przekonaniu autora uzyskane wyniki z przeprowadzonych badań świadczą o zrealizowaniu postawionego we wstępie pracy celu badawczego. W związku ze znacznym zróżnicowaniem wybranych metod propagacji danych oraz złożonością opracowanych rozwiązań nie obyło się bez wprowadzenia pewnych uproszczeń. Najważniejsze ograniczenia przyjęte w badaniach dotyczyły następujących zagadnień:

1. Do analizy wybrano jedynie przypadki propagacji danych dla małych i średnich systemów rozproszonych. Konsekwencją tego było ograniczenie się w przypadku propagacji zadań do sieci lokalnych, w przypadku propagacji błędów do pięciu wybranych struktur topologicznych, natomiast w przypadku propagacji wiedzy – do uproszczonego modelu ruchu drogowego.
2. Ze względu na wybór różnych modeli propagacyjnych nie podjęto próby formalnego zdefiniowania ogólnego modelu propagacyjnego. Solidne podstawy formalne pozwoliłyby na bardziej precyzyjne opisanie możliwych zachowań propagacyjnych oraz określenie złożoności i poprawności proponowanych metod.
3. Pominięto analizę przypadku propagacji zadań oraz propagacji błędów bazujących na komunikacji pośredniej. Na podstawie uzyskanych wyników porównawczych dla propagacji wiedzy między metodą pośrednią i bezpośrednią wydaje się wyjątkowo interesujące zastosowanie modelu pośredniego także w dwóch pozostałych przypadkach.

W pracy nie zostały wyczerpane wszystkie problemy związane z badaniami nad propagacją danych. Wobec tego, przedstawione opracowanie może stanowić podstawę dalszych badań rozwijających zagadnienie efektywnego wykorzystywania metod propagacji danych w nowoczesnych systemach sieciowych. W ostatnich latach obserwuje się intensywny rozwój następujących dziedzin związanych z tematem prezentowanej publikacji: (a) systemy mobilne i adaptacyjne, (b) systemy społeczne, (c) systemy inspirowane przyrodą, (d) systemy hybrydowe i czasu rzeczywistego, (e) sieci złożone, w tym sieci protein i genetyczne. Z pewnością przyniosą one nowe wyzwania związane z zachodzącymi w nich procesami propagacyjnymi.

Do najważniejszych kierunków przyszłych działań w zakresie badań nad propagacją danych można zaliczyć następujące zadania:

1. Dostosowanie metod propagacji zadań w procesie przenoszenia istniejących usług sieciowych do chmury obliczeniowej, dalsze ulepszanie metody promocji oraz poszukiwanie efektywnego wykorzystania metody pośredniej do personalizacji usług.
2. Mechanizm propagowania błędów może być podstawą kontroli jakości systemu rozproszonego, w szczególności algorytmu sprawdzającego odporność na błędy dla poszczególnych modułów systemu. Rozszerzenie zaproponowanego podejścia na systemy czasu rzeczywistego wydaje się być logicznym, następnym posunięciem. Wstępne założenia w tym zakresie zostały opracowane przez autora w ramach projektu badawczego PIEF-GA-2010-274375 pt.: *To what extent can design principles for complex networks be derived from the study of error propagation phenomenon in smart and bio-inspired network structures* zaakceptowanego do realizacji przez Komisję Europejską w ramach 7. Programu Ramowego.
3. Dotychczas stosowana metoda propagacji bezpośredniej w wielu przypadkach jest mało efektywna, np. w analizie niezawodności rzeczywistych systemów

sieciowych. Badania z wykorzystaniem propagacji pośredniej potwierdzają potrzebę dalszych prac nad przystosowaniem rozpatrywanej metody w tych właśnie przypadkach.

4. Opracowanie nowych metod propagacji dla rekomendacji w adaptacyjnych webowych systemach informacyjnych [131]. Zwłaszcza w przypadku rekomendacji interfejsu użytkownika metody propagacyjne powinny uwzględniać możliwość filtrowania demograficznego, bazującego na zawartości, czy kolaboratywnego oraz stosowanie reguł wnioskowania opartych na przypadkach użycia.
5. Prowadzenie dalszych badań nad procesami propagacyjnymi z jednoczesną redukcją ograniczeń przyjętych w pracy, włączając w to: przebadanie funkcjonujących propagacyjnych sieci złożonych oraz opracowanie modelu formalnego dla ogólnego procesu propagacji danych. Istnieje jeszcze duża grupa metod, które są pod względem naukowym interesujące i w większości przypadków zbadane w niedostatecznym stopniu, np. propagacja podobieństw w grupowaniu danych, propagacja zawartości stron internetowych, propagacja zachowań społecznych, propagacja plotki, propagacja consensusu dla systemów wspomagania podejmowania decyzji czy propagacja zaufania dla usług sieciowych.

W podsumowaniu należy podkreślić, że uzyskane wyniki można rozpatrywać zarówno lokalnie w sposób zindywidualizowany dla każdego przypadku osobno, jak i globalnie odnośnie do jednorodnych grup systemów sieciowych, nie tylko na poziomie projektowym (opracowane metody), lecz także implementacyjnym (wykonane prototypy). W przypadku opracowanych metod propagacji danych nie można przewidzieć wszystkich prawdopodobnych zastosowań. Niektóre z wymienionych rozwojowych zagadnień już są wstępnie opracowywane przez autora i współpracujące z nim osoby. Przewidywany w najbliższych latach rozwój wirtualnych systemów rozproszonych, a przy tym dalszy wzrost oczekiwań użytkowników permanentnie wymusza poszukiwanie coraz lepszych rozwiązań informatycznych, w tym również metod propagacji danych.

# Literatura


[1] Albert R., Jeong H., Barabasi A.L., *Error and Attack Tolerance of Complex Networks*, Nature, 406, 1999, s. 378–381.

[2] Alves D., van Ast J., Cong Z., De Schutter B., Babuska R., *Ant Colony Optimization for Traffic Dispersion Routing*, [w:] 13th International IEEE Conference on Intelligent Transportation Systems, IEEE Computer Society, 2010, s. 683–688.

[3] Amigoni F., *Dynamic Agency: a Methodology and Architecture for Multiagent Systems*, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Mediolan 2000.

[4] Amigoni F., Villa M., *An algebraic description of agency design*, [w:] *Workshop on Foundations and Applications Of Collective Agent Based Systems*, Utrecht University, Utrecht 1999, s. 321–332.

[5] Ando Y., Fukazawa Y., Masutani O., Iwasaki H., Honiden S., *Performance of pheromone model for predicting traffic congestion*, [w:] 5th International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, 2006, s. 73–80.

[6] Ando Y., Masutani O., Sasaki H., Iwasaki H., Fukuzawa Y., Honiden S., *Pheromone Model: Application to Traffic Congestion Prediction*, [w:] LNAI 3910, Springer-Verlag, 2006, s. 182–196.

[7] Applegate D.L., Bixby R.E., Chvatal V., Cook W., Espinoza D.G., Goycoolea M., Helsgaun K., *Certification of an optimal TSP tour through 85,900 cities*, Operations Research Letters, 37, 2009, s. 11–15.

[8] Aquiro A.C., Bancho A.C., Fereria M.G.V., da Silva J.D., *A Multi-agent Load Balancing Architecture for Distributed Object Applications*, [w:] Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications & Conference on Real-Time Computing Systems and Applications, IEEE Computer Society, Las Vegas, Nevada, USA, 2006, s. 1056–1061.

[9] Babaoglu O., Canright G., Deutsch A., Di Caro G., *Design patterns from biology for distributed computing*. ACM Trans. on Autonomous Adaptive Systems, 1, 1, 2006, s. 26–66.

[10] Bales M.E., Johnson S.B., *Graph theoretic modeling of large-scale semantic networks*, Journal of Biomedical Informatics, 39, 2006, s. 451–464.

[11] Barbucha D., Czarnowski I., Jedrzejewicz P., Ratajczak E., Wierzbowska I., *JADE-Based A-Team as a Tool for Implementing Population-Based Algorithms*, [w:] Proc. of the Sixth International Conference on Intelligent Systems Design and Applications, IEEE Computer Society, 2006, s. 144–149.

[12] Bellifemine F., Caire G., Poggi A., Rimassa G., *JADE: A software framework for developing multi-agent applications. Lessons learned*, Information and Software, 50, 2008, s. 10–21.

[13] Benbernou S., Hacid M.S., *Resolution and Constraint Propagation for Semantic Web Services Discovery*, Distributed and Parallel Databases, 18, 1, 2005, s. 65–81.

[14] Bernon C., Chevrier V., Hilaire V., Marrow P., *Applications of Self-Organising Multi-Agent Systems: An Initial Framework for Comparison*, Informatica, 30, 2006, s. 73–82.

[15] Boccaletti S., Latora V., Moreno Y., Chavez M., Hwang D.U., *Complex networks: Structure and dynamics*, Physics Reports, 424, 2006, s. 175–308.

[16] Brand S., Apt K.R., *Schedulers and redundancy for a class of constraint propagation rules*, Theory and Practice of Logic Programming, 5, 4–5, 2005, s. 441–465.
[17] Brueckner S., *Return from the Ant*, Uniwersytet Humboldta, Berlin 2000.
[18] Brueckner S., Hassa S., Jelasity M., Yamins D., *Engineering Self-Organising Systems*, [w:] 4th International Workshop, ESOA 2006, Springer-Verlag, 2007.
[19] Cami A., Deo N., *Techniques for analyzing dynamic random graph models of web-like networks: An overview*, Networks, 51, 4, 2008, s. 211–255.
[20] Cao J., Spooner D.P., Jarvis S.A., Nudd G.R., *Grid load balancing using intelligent agents*, Future Generation Computer Systems, 21, 1, 2005, s. 135–149.
[21] Castillo J.M., Ossowski S., Pastor L., *MECIMPLAN: an agent-based methodology for planning*, Int. J. of Intelligent Information and Database Systems, 3, 2, 2009, s. 125–145.
[22] Centola D., *The Spread of Behavior in an Online Social Network Experiment*, Science, 329, 2010, s. 1194–1197.
[23] Centola D., Eguiluz V.M., Macy M.W., *Cascade dynamics of complex propagation*, Physica A, 374, 2007, s. 449–456.
[24] Cernuzzi L., Zambonelli F., *Profile based comparative analysis for AOSE methodologies evaluation*, [w:] Proc. of the ACM Symposium on Applied Computing, 2008, s. 60–65.
[25] Chakravarti A.J., Baumgartner G., Lauria M., *The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network*, IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, 35, 3, 2005, s. 373–384.
[26] Chang B.M., Jo J.W., Her S.H., *Visualization of exception propagation for java using static analysis*, [w:] Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation, IEEE Computer Society, Washington 2002, s. 173–182.
[27] Chmiel M., Gawinecki M., Kaczmarek P., Szymczak M., Paprzycki M., *Efficiency of JADE agent platform*, Scientific Programming, 13, 2005, s. 1–14.
[28] Chow K.P., Kwok Y.K., *On Load Balancing for Distributed Multiagent Computing*, IEEE Transactions on Parallel and Distributed Systems, 13, 8, 2002, s. 787–801.
[29] Cohen E., Shenker S., *Replication strategies in unstructured peer-to-peer*, [w:] ACM SIGCOMM, ACM, 2002, s. 177–190.
[30] Cormen T.H., Leiserson Ch.E., Rivest R.L., Stein C., *Wprowadzenie do algorytmów*, Wydawnictwo Naukowo-Techniczne, Warszawa 2007.
[31] Czech Z., *Wprowadzenie do obliczeń równoległych*, Wydawnictwo Naukowe PWN, Warszawa 2010.
[32] Delafosse M., Clerentin A., Delahoche L., Brassart E., Marhic B., *Multi sensor fusion and constraint propagation to localize a mobile robot*, [w:] IEEE International Conference on Industrial Technology, IEEE Computer Society, Hammamet 2004, s. 66–71.
[33] Delgado J., *Emergence of social conventions in complex networks*, Artificial Intelligence, 141, 2002, s. 171–185.
[34] Delgado J., Solé R.V., *Self-synchronization and task fulfilment in ant colonies*, Journal of Theoretical Biology, 205, 3, August 2000, s. 433–441.
[35] DeLoach S.A., *Moving multi-agent systems from research to practice*, Int. J. Agent-Oriented Software Engineering, 3, 4, 2001, s. 378–382.
[36] Deneubourg J.L., Aron S., Goss S., Pasteels J.M., *The self-organizing exploratory pattern of the Argentine ant*, J. Insect Behavior, 3, 1990, s. 159–168.
[37] Deneubourg J.L., Mondada F., Floreano D., Gambardella L.M., *Evolving self-organizing behaviors for a swarm-bot*, Autonomous Robots, 17, 2004, s. 223–245.
[38] Dobrowolski G., *Technologie agentowe w zdecentralizowanych systemach informacyjno-decyzyjnych*, Akademia Górniczo-Hutnicza w Krakowie, Kraków 2002.

[39] Dorigo M., Gambardella L.M., *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*, IEEE Trans. Evol. Comp., 1, 1, 1997, s. 53–66.
[40] Dorigo M., Stutzle T., *Ant Colony Optimization*, Massachusetts Institute of Technology, London 2004.
[41] Elliot M.A., *Stigmergic Collaboration: A Theoretical Framework for Mass Collaboration*, The University of Melbourne, Melbourne 2007.
[42] Engelbrecht A.P., *Computational Intelligence. An Introduction*, Wiley, England, 2002.
[43] Eugster P.T., Guerraoui R., Kermarrec A.M., Massoulie L., *Epidemic Information Dissemination in Distributed Systems*, Computer, May, 2004, s. 60–67.
[44] Fatima S., Wooldridge M., *Adaptive Task and Resource Allocation in Multi-Agent Systems*, [w:] Proc. of the 5th International Conference on Autonomous Agents, ACM, New York 2001, s. 537–544.
[45] Filipowicz B., Chmiel W., Kadłuczka P., *Ukierunkowane przeszukiwanie przestrzeni rozwiązań w algorytmach rojowych*, Automatyka, 13, 2, 2009, s. 247–255.
[46] Fronczak A., Fronczak P., *Świat sieci złożonych. od fizyki do Internetu*, Wydawnistwo Naukowe PWN, Warszawa 2009.
[47] Gabbar H.A., *Improved Qualitative Fault Propagation Analysis*, Journal of Loss Prevention in the Process Industries, 20, 2007, s. 260–270.
[48] Gaston M.E., des Jardins M., *The efect of network structure on dynamic team formation in multi-agent systems*, Computational Intelligence, 24, 2, 2008, s. 122–157.
[49] Givoni I.E., Frey B.J., *A Binary Variable Model for Affinity Propagation*, Neural Computation, 21, 2009, s. 1589–1600.
[50] Gkantsidis C., Mihail M., Saberi A., *Random walks in peer-to-peer networks*, [w:] Twenty-third Annual Joint Conference of the IEEE, IEEE Computer Society, 2004, s. 120–130.
[51] Grzech A., *Sterowanie ruchem w sieciach teleinformatycznych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2002.
[52] Guting H., de Almeida T., Ding Z., *Modelling and querying moving objects in networks*, The VLDB Journal, 15, 2, 2006, s. 165–190.
[53] Hadeli K., Valckenaers P., Kollingbaum M., Brussel H., *Multi-agent coordination and control using stigmergy*, Computers in Industry, 53, 2004, s. 75–96.
[54] Helsgaun K., *General k-opt submoves for the Lin-Kernighan TSP heuristic*, Math. Prog. Comp., 1, 2–3, 2009, s. 119–163.
[55] He Y., Ren H., Liu J., Yang B., *On the reliability of large-scale distributed systems – A topological view*, Computer Networks, 53, 2009, s. 2140–2152.
[56] Hmida F.B., Chaari W.L., Tagina M., *Performance Evaluation of Multiagent Systems: Communication Criterion*, [w:] LNAI4953, Springer-Verlag, 2008, s. 773–782.
[57] Huang C.Y., Cheng C.Y., Sun C.T., *Resource Limitations, Transmission Costs and Critical Thresholds in Scale-Free Networks*, Springer-Verlag, 2008, s. 485–494.
[58] Huang J., Nicol D.M., *A Formal-Semantics-Based Calculus of Trust*, IEEE Internet Computing, September–October 2010, s. 38–46.
[59] Jennings N., Sycara K., Wooldridge M., *A roadmap of agent research and development*, Autonomous Agents and Multi-Agent Systems, 1, 1998, s. 7–38.
[60] Jiang Y.C., Jiang J.C., *A multi-agent coordination model for the variation of underlying network topology*, Expert Systems with Applications, 29, 2005, s. 372–382.
[61] Johnson D.S., McGeoch L.A., *Experimental Analysis of Heuristics for the STSP*, [w:] *The Traveling Salesman Problem and its Variations*, G. Gutin, A. Punnen (red.), Kluwer, 2002.
[62] Jurasovic K., Jezic G., Kusek M., *A Performance Analysis of Multi-Agent Systems*, International Transactions on Systems Science and Applications, 1, 4, 2006, s. 335–342.

[63] Kalinowski P., Katarzyniak R., *Methods of Task Redistribution in Multiagent Systems*, [w:] LNAI 6070, Springer-Verlag, 2010, s. 72–81.
[64] Kazienko P., Musiał K., *On utilising social networks to discover representatives of human communities*, Int. J. Intelligent Information and Database Systems, 1, 3–4, 2007, s. 293–310.
[65] Kefi M., Ghedira K., Korbaa Yim P., *Container handling using multi-agent architecture*, Int. J. of Intelligent Information and Database Systems, 3, 3, 2009, s. 338–360.
[66] Kelly M., Serugendo G.D.M., *A Decentralised Car Traffic Control System Simulation Using Local Message Propagation Optimised with a Genetic Algorithm*, [w:] ESOA 2006, Springer-Verlag, 2007, s. 192–210.
[67] Klamut J., Durczewski K., Sznajd J., *Wstęp do fizyki przejść fazowych*, Zakład Narodowy im. Ossolińskich, Wrocław 1979.
[68] Kolasa T., Król D., *A Survey of Algorithms for Paper-Reviewer Assignment Problem*, IETE Tech Rev, 28, 2, 2011, s. 123–134.
[69] Kolp M., Giorgini J., Mylopoulos J., *Multi-Agent Architectures as Organizational Structures*, Autonomous Agents and Multi-Agent Systems, 13, 2006, s. 3–25.
[70] Koronacki J., Mielniczuk J., *Statystyka dla studentów kierunków technicznych i przyrodniczych*, Wydawnictwa Naukowo-Techniczne, Warszawa 2006.
[71] Korzeniowski Ł., Król D., *Instant messaging data processing of autonomous mobile systems*, [w:] Knowledge processing and reasoning for information society, Exit, 2008.
[72] Kotowski S., *Analiza algorytmów genetycznych jako układów dynamicznych*, Polska Akademia Nauk, Warszawa 2008.
[73] Koźlak J., Dobrowolski G., Kisiel-Dorohinicki M., Nawarecki E., *Anti-Crisis Management of City Traffic Using Agent-Based Approach*, Journal of Universal Computer Science, 14, 14, 2008, s. 2359–2380.
[74] Krivokapic N., Kemper A., *Migrating Autonomous Objects in a WAN Environment*, Journal of Intelligent Information Systems, 15, 2000, s. 221–251.
[75] Król D., *A Propagation Strategy Implemented in Communicative Environment*, [w:] LNAI 3682, Springer-Verlag, 2005, s. 527–533.
[76] Król D., *Dynamika danych w modelu obiektowym*, Politechnika Wrocławska, Wrocław 2001.
[77] Król D., *Example-Based Framework for Propagation of Tasks in Distributed Environments*, [w:] Intelligence Integration in Distributed Knowledge Management, IGI Global, 2008.
[78] Król Z.P., *Przegląd i komputerowa implementacja algorytmów genetycznych w oprogramowaniu edukacyjnym*, Wydział Elektryczny Politechniki Warszawskiej, Warszawa 2006.
[79] Król D., Drożdżowski M., *Use of MaSE methodology for designing a swarm-based multi-agent system*, Journal of Intelligent & Fuzzy Systems, 21, 3, 2010, s. 221–231.
[80] Król D., Kukla G., *Analysis of the Error Propagation Phenomenon in Network Structures*, Computing and Informatics, 28, 6, 2009, s. 811–842.
[81] Król D., Kukla G., *Distributed Class Code Propagation with Java*, [w:] LNAI4252, Springer-Verlag, 2006, s. 259–266.
[82] Król D., Kukla G., *Quantitative Analysis of the Error Propagation Phenomenon in Distributed Information Systems*, [w:] *Asian Conference on Intelligent Information and Database Systems*, IEEE Computer Society, 2009, s. 202–207.
[83] Król D., Kukla G., *Reusable Grid Computing Architecture Based on Code Propagation Method*, Int. J. Intelligent Information and Database Systems, 3, 1, 2009, s. 44–55.
[84] Król D., Lupa A., *Agent Migration: Framework for Analysis*, Journal of Universal Computer Science, 15, 4, 2009, s. 941–966.
[85] Król D., Lupa A., *Code and Data Propagation on a PC's Multi-Agent System*, [w:] New Trends in Multimedia and Network Information Systems, IOS Press, 2008.

[86] Król D., Mrożek M., *Swarm-Based Multi-Agent Simulation: a Case Study of Urban Traffic Flow in the City of Wroclaw*, [w:] International Conference on Computational Collective Intelligence – Technologies and Applications, Springer-Verlag, 2011, s. 191–200.
[87] Król D., Popiela Ł., *Modelling Shortest Path Search Techniques by Colonies of Cooperating Agents*, [w:] LNAI, Springer-Verlag, 2009, s. 665–675.
[88] Król D., Zelmozer M., *Structural Performance Evaluation of Multi-Agent Systems*, Journal of Universal Computer Science, 14, 7, 2008, s. 1154–1178.
[89] Kuczyński K., Stegierski R., *Routing w sieciach IP*. Uniwersytet Marii Curie-Skłodowskiej w Lublinie, Lublin 2011.
[90] Kukla G., Kazienko P., Bródka P., Filipowski T., *SocLaKE – Social Latent Knowledge Explorator*, The Computer Journal, 55, 3, 2012, s. 258–276.
[91] Kukla G., Król D., *Errors in Structure Self-organization: Statistical Analysis*, [w:] LNAI, Springer-Verlag, 2009, s. 450–459.
[92] Kurihara S., Tamaki H., Numao M., Yano J., Kagawa K., Morita T., *Traffic Congestion Forecasting based on Pheromone Communication Model for Intelligent Transport Systems*, [w:] IEEE Congress on Evolutionary Computation, IEEE Computer Society, 2009, s. 2879–2884.
[93] Lee W.P., *Parallelizing evolutionary computation: A mobile agent-based approach*, Expert Systems with Applications, 32, 2, 2007, s. 318–328.
[94] Levis P., Patel N., Culler D., Shenker S., *Trickle: A selfregulating algorithm for code propagation and maintenance in wireless sensor networks*, Berkely 2004.
[95] Lin S., Kernighan B.W., *An effective heuristic algorithm for the traveling salesman problem*, Operations Research, 21, 1973, s. 498–516.
[96] Lin J., Kuo S., *Resolving error propagation in distributed systems*, Information Processing Letters, 74, 5–6, 2000, s. 257–262.
[97] Liu L., Antonopoulos N., Mackin S., *Managing peer-to-peer networks with human tactics in social interactions*, J. Supercomput, 44, 2008, s. 217–236.
[98] Liu H., Jia X., Wan P.J., Liu X., Yao F.F., *A Distributed and Efficient Flooding Scheme Using 1-Hop Information in Mobile Ad Hoc Networks*, IEEE Transactions On Parallel And Distributed Systems, 18, 5, 2007, s. 1–14.
[99] Liu S., Sun T., Hung Ch.-Ch., *Multi-Robot Task Allocation Based on Swarm Intelligence*, [w:] *Multi-Robot Systems, Trends and Development*, T. Yasuda (red.), InTech, 2011.
[100] Martel M., *Semantics of roundoff error propagation in finite precision calculations*, Higher-Order and Symbolic Computation, 19, 1, 2006, s. 7–30.
[101] Meng X., Zerfos P., Samanta V., Wong S.H.Y., Lu S., *Analysis of the Reliability of a Nationwide Short Message Service*, [w:] Proceedings of 26th IEEE International Conference on Computer Communications, IEEE Computer Society, 2007, s. 1811–1819.
[102] Mezard M., *Where Are the Exemplars?*, Science, 315, 2007, s. 949–951.
[103] Michalewicz Z., *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, WNT, Warszawa 1996.
[104] Milano M., Poli A., *MAGMA: A multiagent architecture for metaheuristics*, IEEE Trans. Sys. Man and Cybernetics – Part B, 34, 2004, s. 925–941.
[105] Milo R., Shen-Orr S., Itzkovitz S., Kashtan N., Chklovskii D., Alon U., *Network Motifs: Simple Building Blocks of Complex Networks*, Science, 298, 2002, s. 824–827.
[106] Mitschang B., *Data propagation as an enabling technology for collaboration and cooperative information systems*, Computers in Industry, 52, 2003, s. 59–69.
[107] Moallemi C.C., van Roy B., *Consensus Propagation*, IEEE Transactions on Information Theory, 52, 11, 2006, s. 4753–4766.

[108] Nagel K., Schreckenberg M., *A cellular automaton model for freeway traffic*, Journal de Physique, 2, 1992, s. 2221–2229.
[109] Narzt W., Wilflingsede U., Pomberger G., Kolb D., Hortner H., *Self-organising congestion evasion strategies using ant-based pheromones*, Intelligent Transport Systems, 4, 1, 2009, s. 93–102.
[110] Navarro G., Ortega-Ruiz J.A., Ametller J., Robles S., *Distributed Authorization Framework for Mobile Agents*, [w:] LNCS, Springer-Verlag, 2005, s. 127–136.
[111] Newman M., *The structure and function of complex networks*, SIAM Review, 45, 2003, s. 167–256.
[112] Nguyen T.N., *Advanced Methods for Inconsistent Knowledge Management*, Springer-Verlag, London 2007.
[113] Nguyen T.N., *Inconsistency of Knowledge and Collective Intelligence*, Cybernetics and Systems, 39, 6, 2008, s. 542–562.
[114] Nguyen T.N., *Metody wyboru consensusu i ich zastosowanie w rozwiązywaniu konfliktów w systemach rozproszonych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2002.
[115] Nikoletseas S., Prasinos G., Spirakis P., Zaroliagis C., *Attack propagation in networks*, [w:] Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures, New York 2001, s. 67–76.
[116] Ni J., Luo D., Ou Y., Luo C., *Agent-based evolutionary optimisation of trading strategies*, Int. J. Intelligent Information and Database Systems, 2, 1, 2008, s. 25–48.
[117] Olfati-Saber R., *Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory*, IEEE Transactions on Automatic Control, 51, 3, 2006, s. 401–420.
[118] Olfati-Saber R., Fax J.A., Murray R.M., *Consensus and Cooperation in Networked Multi-Agent Systems*, Proceedings of the IEEE, 95, 1, 2007, s. 215–233.
[119] Parunak V.D., Brueckner S., *Engineering Swarming Systems*, [w:] *Methodologies and Software Engineering for Agent Systems*, F. Bergenti et al. (red.), Springer-Verlag, 2004.
[120] Pellegrini P., Moretti E., *A Computational Analysis on a Hybrid Approach: Quick-and-dirty ant colony optimization*, Applied Mathematical Sciences, 3, 23, 2009, s. 1127–1140.
[121] Piqueira J., Vasconcelos A., Gabriel C., Araujo V., *Dynamic models for computer viruses*, Computers & Security, 27, 2008, s. 355–359.
[122] Renfrew D., Yu X.H., *Traffic Signal Control with Swarm Intelligence*, [w:] 5th International Conference on Natural Computation, 2009, s. 79–83.
[123] Repantis T., *Adaptive Data Propagation in Peer-to-Peer Systems*, Riverside, 2004.
[124] Sakata S., Yamamori T., *Topological Relationships Between Brain and Social Networks*, Neural Networks, 20, 2007, s. 12–21.
[125] Santoro N., *Design and Analysis of Distributed Algorithms*, Wiley, 2007.
[126] Scellato S., Fortuna L., Frasca M., Gómez-Gardenes J., Latora V., *Traffic optimization in transport networks based on local routing*, The European Physical Journal B, 73, 2, 2010, s. 303–308.
[127] Scheurer C.A., Scheurer H.K., Kropf P., *Load Balancing Driven Process*, Institute of Informatics and Applied Mathematics, Berne 1995.
[128] Serazzi G., Zanero S., *Computer Virus Propagation Models*, Springer-Verlag, 2001, s. 1–25.
[129] Sheskin D.J., *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall, Boca Raton 2004.
[130] Shi X.H., Liang Y.C., Lee H.P., Lu C., Wang D.X., *Particle swarm optimization-based algorithms for TSP and generalized TSP*, Inf. Process. Lett., 103, 5, 2007, s. 169–176.
[131] Sobecki J., *Rekomendacja interfejsu użytkownika w adaptacyjnych webowych systemach informacyjnych*, Wrocław 2009.
[132] Strogatz S.H., *Exploring complex networks*, Nature, 410, 2001, s. 268–276.
[133] Sturm A., Dori D., Shehory O., *Comparative Evaluation of Agent-Oriented Methodologies*, [w:] Methodologies and Software Engineering for Agent Systems, Kluwer, 2004.

[134] Sznajd-Weron K., *Nowa lokalna dynamika w układzie spinów isingowskich*, Wrocław 2005.
[135] Sznajd-Weron K., Sznajd J., *Opinion evolution in closed community*, International Journal of Modern Physics C, 11, 6, 2000, s. 1157–1165.
[136] Szuba T., *Computational Collective Intelligence*, Wiley, New York 2001.
[137] Tan K., Zhang K., Zhu W., *Shortest Path Routing in Partially Connected Ad Hoc Networks*, [w:] IEEE Globecom, IEEE Computer Society, 2003, s. 1038–1042.
[138] Teodorovic D., *Swarm intelligence systems for transportation engineering: Principles and applications*, Transportation Research Part C: Emerging Technologies, 16, 6, 2008, s. 651–667.
[139] Thain D., Livny M., *Error Scope on a Computational Grid: Theory and Practice*, [w:] Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, IEEE Computer Society, 2002, s. 199–208.
[140] Timpanaro A., Prado C., *Generalized Sznajd model for opinion propagation*, Physical Review E, 80, 2009, s. 1–8.
[141] Trojanowski K., *Metaheurystyki praktyczne*, Wyższa Szkoła Informatyki Stosowanej i Zarządzania, Warszawa 2008.
[142] Uhruski P., Grochowski M., Schaefer R., *A Two-Layer Agent-Based System for Large-Scale Distributed Computation*, Computational Intelligence, 24, 3, 2008, s. 191–212.
[143] Watts D.J., Strogatz S.H., *Collective dynamics of "small-world" networks*, Nature, 393, 1998, s. 440–442.
[144] Wąs J., *Algorytmy modelowania inteligentnych zachowań w zagadnieniach dynamiki pieszych z zastosowaniem niehomogenicznych automatów komórkowych*, Akademia Górniczo-Hutnicza, Kraków 2006.
[145] Wedde H.F., Farooq M., *A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks*, Journal of Systems Architecture, 52, 2006, s. 461–484.
[146] Wellman B., *Computer Networks As Social Networks*, Science, 293, Sep. 14, 2001, s. 2031–2034.
[147] Werbos P.J., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences*, Boston, USA, 1974.
[148] Weron K., *Nowa lokalna dynamika w układzie spinów isingowskich*, Uniwersytet Wrocławski, Wrocław 2005.
[149] White T., Pagurek B., *Towards multi-swarm problem solving in networks*, International Conference on Multi Agent Systems, Proceedings, 3–7 July 1998, s. 333–340.
[150] Wierzchoń S.T., *Sztuczne systemy immunologiczne. Teoria i zastosowania*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001.
[151] Wolfram S., *A New Kind of Science*, Wolfram Media, Champaign, ILL, 2002, http://www.wolframscience.com/nksonline/toc.html
[152] Wooldridge M., *An Introduction to MultiAgent Systems*, Wiley, Chichester 2002.
[153] Wu B., Kshemkalyani A.D., *Modeling message propagation in random graph networks*, Computer Communications, 31, 2008, s. 4138–4148.
[154] Xie X.F., Liu J., *Multiagent Optimization System for Solving the TSP*, IEEE Trans. Sys., Man and Cybernetics – Part B, 39, 2, 2009, s. 489–502.
[155] Zeng J., Zhang S., Wu Ch., Ji X., *Modelling topic propagation over the Internet*, Mathematical and Computer Modelling of Dynamical Systems, 15, 1, 2009, s. 83–93.
[156] Zhang H.L., Leung C.H., Raikundalia G.K., *Topological analysis of AOCD-based agent networks and experimental results*, Journal of Computer and System Sciences, 74, 2008, s. 255–278.
[157] Zhang D., Simoff S., Aciar S., Debenham J., *A multi agent recommender system that utilises consumer reviews in its recommendations*, Int. J. Intelligent Information and Database Systems, 2, 1, 2008, s. 69–81.
[158] Zhu Q., *Topologies of Agents Interactions in Knowledge Intensive Multi-Agent Systems for Networked Information Services*, Advanced Engineering Informatics, 20, 2006, s. 31–45.

# Spis rysunków

# Spis tabel

# Selected data propagation methods in distributed systems

This work presents selected data propagation methods used to solve specific problems in collective intelligence as found in contemporary distributed systems. These methods enable a consistent serial transfer of data between multiple objects present in a common environment which have as a goal the realisation of specific predefined tasks. The main aspect of this research is to examine how data propagation methods can be used in a practical manner to improve task scheduling and delegation, and how to design fault tolerant network solutions in a distributed data processing environment. Further, it is presented how these techniques can be used to develop road traffic simulation models which can then be applied to optimise traffic flow in congested city areas.

Development of a data propagation model begins with a definition of the various types of interactive mechanisms occurring between the elements of a distributed system. These mechanisms can be collective, cooperative or collaborative. The next stage is to choose an appropriate research approach which often consists of several paradigms occurring concurrently. However, this choice is often not straightforward because it should be based on the specific functional requirements of the specific applications being researched. Unfortunately, current research does not give sufficiently comprehensive solutions to the range of problems which occur in propagation processes, thus further evaluation is essential. Two examples taken from literature detail some additional aspects regarding the application of propagation mechanisms. One discusses the way in which questions are propagated through the acquaintance chain existing in the social system and second, the way in which propagation methods can be used to detect critical cut vertices in a large-scale network system.

The book also describes techniques for data propagation enabling more efficient application of the power of distributed computing by applying a dynamic process for task redistribution. This process is strictly correlated to the individual loading factors on specific elements of the system. Applying a break-even point concept and a propagation limitation strategy reduces the frequency of occurrence of negative propagation cycles which may arise between similar architecture units. The examples presented and discussed here identify a wide range of problems which occur either in the application of methodology or its further implementation.

Further aspects of this work present the challenges in modelling error propagation in a distributed system. Research in this area is of fundamental significance in developing a maximum level of reliability in network information systems. Quantitative and qualitative analysis methods are presented which enable the analysis of various propagation scenarios for specific network architectures, errors or distributions. Applying statistical methods to verify results allows the most error resistant architecture to be selected as well as identifying the least harmful types of propagated errors occurring in network systems. Understanding the characteristics of these propagation mechanisms will assist the development of more reliable computer systems, implement more secure communications protocols as well as deepen the understanding of risks inherent in technologies which have already been introduced.

The last part of this book consists of a comparative analysis of direct and indirect communication that is generated during knowledge propagation on a multi-agent platform. Developing the propagation model around stigmergic indirect coordination mechanism (engineering concept inspired by insect social behaviour) proved to be a better solution than applying traditional message dispatch between components of a distributed system. This was despite applying specific threshold, sensitivity, durability and propagation characteristics for each node in the network. On the basis of research and analysis of simulation test results, indirect propagation methods were clearly identified as an effective method for the dissemination of information being specifically useful when applied to optimise traffic flow in transport network modernization programmes or in traffic light signalling.

What is original about this research is that it focuses on different data propagation mechanisms based to a large extent on indirect methods. This is an area where insufficient emphasis has been placed on existing network systems resulting in a less efficient application of propagation mechanisms. The results presented allow the author to substantiate the central thesis that data propagation methods provide effective mechanisms for task distribution, error analysis and knowledge integration in modern network systems.

# Skorowidz

Prezentowana monografia poświęcona jest zagadnieniom dotyczącym wybranych metod propagacji danych we współczesnych systemach rozproszonych. W szczególności badano, jak proces propagacji danych może być praktycznie wykorzystany do rozproszonego przetwarzania zadań, projektowania odpornych na błędy rozwiązań sieciowych oraz do optymalizacji natężenia ruchu drogowego w systemach sterowania ruchem pojazdów w mieście. Zamiarem autora było uporządkowanie i poszerzenie wiedzy w odniesieniu do mechanizmów propagacyjnych, które zwykle dość pobieżnie są omawiane w literaturze, a równocześnie mają istotne znaczenie dla rozwoju systemów rozproszonych, szczególnie z perspektywy ich zastosowań.

Najważniejsze z przedstawionych wyników badań dotyczą głównie propozycji związanych z efektywnym wykorzystywaniem metod propagacji bezpośredniej i pośredniej. Pokazano między innymi, że metody inspirowane przyrodą wykorzystywane w propagacji wiedzy są efektywniejsze od metod tradycyjnych bazujących na wymianie komunikatów. Poszukiwanie rozwiązań było realizowane metodami algorytmicznymi, w tym z zastosowaniem algorytmów inspirowanych przyrodą. Wszystkie badania zostały zweryfikowane eksperymentalnie.

Książka jest przeznaczona głównie dla studentów informatyki oraz pracowników naukowych poszukujących coraz lepszych rozwiązań w obszarze systemów komputerowych wykorzystujących metody propagacji danych. Może być również przydatna analitykom i projektantom nowoczesnych systemów rozproszonych.

Wydawnictwa Politechniki Wrocławskiej
są do nabycia w księgarni „Tech”
plac Grunwaldzki 13, 50-377 Wrocław
budynek D-1 PWr., tel. 71 320 29 35
Prowadzimy sprzedaż wysyłkową
zamawianie.ksiazek@pwr.wroc.pl

ISBN 978-83-7493-669-9