Autoreferat

Joanna Berlińska

Spis treści

1 Imię i nazwisko

Joanna Berlińska

2 Posiadane dyplomy i stopnie naukowe

2011 Doktor nauk matematycznych w zakresie informatyki (z wyróżnieniem)
Wydział Matematyki i Informatyki, Uniwersytet im. Adama Mickiewicza w Poznaniu
Tytuł rozprawy: Scheduling divisible loads in heterogeneous distributed systems
Promotor: prof. dr hab. inż. Maciej Drozdowski

2007 Magister (informatyka)
Wydział Matematyki i Informatyki, Uniwersytet im. Adama Mickiewicza w Poznaniu
Tytuł pracy magisterskiej: Algorytmy genetyczne i ich zastosowanie w szeregowaniu zadań w systemie gniazdowym
Promotor: prof. dr hab. Jerzy Jaworski

2007 Magister (matematyka, specjalność: matematyka finansowa i aktuarialna)
Wydział Matematyki i Informatyki, Uniwersytet im. Adama Mickiewicza w Poznaniu
Tytuł pracy magisterskiej: Modelowanie zmienności instrumentów finansowych za pomocą przełącznikowych modeli Markowa
Promotor: prof. dr hab. Witold Wnuk

3 Informacja o dotychczasowym zatrudnieniu w jednostkach naukowych

1.10.2011 – obecnie: adiunkt, Wydział Matematyki i Informatyki, Uniwersytet im. Adama Mickiewicza w Poznaniu
(urlopy macierzyńskie: 19.12.2011 – 6.05.2012 i 25.01.2015 – 23.01.2016)

4 Omówienie osiągnięć, o których mowa w art. 219 ust. 1 pkt. 2 Ustawy

Osiągnięciem naukowym, o którym mowa w art. 219 ust. 1 pkt. 2 Ustawy, jest cykl powiązanych tematycznie artykułów naukowych pod wspólnym tytułem

„Szeregowanie zadań w sieciach zbierających dane”.

Publikacje wchodzące w skład osiągnięcia:

[H1] J. Berlińska, *Communication scheduling in data gathering networks with limited memory*, Applied Mathematics and Computation 235 (2014) 530-537.

[H2] J. Berlińska, *Scheduling for data gathering networks with data compression*, European Journal of Operational Research 246 (2015) 744-749.

[H3] J. Berlińska, *Scheduling data gathering with maximum lateness objective*, w: R. Wyrzykowski et al. (eds), Parallel Processing and Applied Mathematics: 12th International Conference PPAM 2017, Part II, Lecture Notes in Computer Science 10778, Springer, Cham 2018, 135-144.

[H4] J. Berlińska, *Scheduling in a data gathering network to minimize maximum lateness*, w: B. Fortz, M. Labbé (eds), Operations Research Proceedings 2018, Springer, Cham 2019, 453-458.

[H5] J. Berlińska, *Makespan minimization in data gathering networks with dataset release times*, w: R. Wyrzykowski et al. (eds), Parallel Processing and Applied Mathematics: 13th International Conference PPAM 2019, Part II, Lecture Notes in Computer Science 12044, Springer, Cham 2020, 230-241.

[H6] J. Berlińska, *Heuristics for scheduling data gathering with limited base station memory*, Annals of Operations Research 285 (2020) 149-159.

[H7] J. Berlińska, *Scheduling in data gathering networks with background communications*, Journal of Scheduling 23 (2020) 681-691.

[H8] J. Berlińska, *A comparison of priority rules for minimizing the maximum lateness in tree data gathering networks*, Engineering Optimization (2021) DOI: 10.1080/0305215X.2020.1861263, 14 stron.

Liczby cytowań i inne dane naukometryczne dotyczące powyższych prac znajdują się w wykazie osiągnięć naukowych (załącznik 4).

W autoreferacie i wykazie osiągnięć przyjęłam następującą konwencję oznaczania cytowanych prac. Numery poprzedzone literą H odnoszą się do wymienionych powyżej artykułów składających się na prezentowane osiągnięcie naukowe. Numerami poprzedzonymi literą P zostały oznaczone moje pozostałe prace, przedstawione w punkcie 7.1 autoreferatu. Numery niepoprzedzone literami odwołują się do spisu literatury umieszczonego na końcu autoreferatu.

4.1 Wprowadzenie

Pozyskiwanie i przetwarzanie dużych ilości danych jest istotą wielu współczesnych zastosowań komputerowych. Wykonywanie obliczeń na pojedynczym komputerze często nie jest możliwe ze względu na niewystarczającą moc obliczeniową. Wykorzystuje się więc złożone z wielu komputerów systemy rozproszone, takie jak gridy i chmury obliczeniowe. W celu zapewnienia wydajnego działania złożonej sieci różnorodnych zasobów konieczne jest zarządzanie nimi w sposób pozwalający na maksymalne wykorzystanie ich możliwości. Obszarem informatyki zajmującym się problemami tego typu jest teoria szeregowania zadań. Analizuje się w niej zagadnienia polegające na przydzieleniu pewnych zadań do dostępnych maszyn w czasie. Taki przydział zadań, spełniający pewne dodatkowe założenia wynikające ze specyfiki rozważanego

problemu, nazywany jest uszeregowaniem. Zazwyczaj rozważa się problemy optymalizacyjne, w których należy skonstruować uszeregowanie najlepsze ze względu na wybrane kryterium lub kryteria.

Szeregowaniu zadań w systemach równoległych i rozproszonych poświęconych zostało wiele badań naukowych [14, 43, 39, 46]. Koncentrowały się one przede wszystkim na problemach podziału pracy między dostępne procesory, równoważenia ich obciążeń i szeregowania rozsyłania danych wejściowych z jednego lub wielu serwerów do węzłów wykonujących obliczenia. Zbieranie wyników obliczeń uzyskanych przez te węzły było najczęściej pomijane. Przyjmowano założenie, że mogą one być przechowywane w rozproszonym systemie plików lub że czas ich gromadzenia jest pomijalny w porównaniu z czasem obliczeń. Założenie takie jednak w wielu przypadkach nie jest uzasadnione. Jeśli otrzymane wyniki muszą zostać scalone lub należy zbadać zależności między nimi, konieczne może być zgromadzenie danych pochodzących z pewnego zbioru węzłów na jednym komputerze. W przypadku uzyskania dużego wolumenu wyników, czas ich gromadzenia może być podobnego rzędu co czas wcześniejszych obliczeń, a zatem nie powinien być pomijany. Zaniedbywanie problemów szeregowania zbierania danych doprowadziło więc do powstania istotnej luki w literaturze. Jej wypełnienie było celem badań składających się na prezentowane tu osiągnięcie naukowe.

Gromadzenie wyników obliczeń prowadzonych w systemach rozproszonych nie jest jedynym przypadkiem, w którym potrzebne są algorytmy szeregowania zbierania danych. Ogromne ilości danych są zbierane każdego dnia przez bezprzewodowe sieci czujników. Podobnie jak w przypadku wielu innych technologii, początkowo motorem rozwoju takich sieci były zwłaszcza zastosowania militarne. Czujniki są w szczególności wykorzystywane do wczesnego wykrywania ataków chemicznych czy biologicznych oraz aktywności wrogich sił na kontrolowanym terenie. Używane są również do nadzorowania wyposażenia i amunicji. Bezprzewodowe sieci czujników są obecnie szeroko wykorzystywane także w wielu innych obszarach [2]. Ich zastosowania w monitorowaniu środowiska obejmują między innymi wczesne wykrywanie i ostrzeganie o katastrofach naturalnych, takich jak pożar, powódź czy trzęsienie ziemi, oraz obserwację poziomu zanieczyszczenia powietrza, wody i gleby. Możliwość śledzenia dzięki czujnikom przemieszczania się ptaków, owadów i innych małych zwierząt znacznie ułatwia prowadzenie badań przyrodniczych. Domowe zastosowania bezprzewodowych sieci czujników już od dawna obejmują nie tylko systemy przeciwpożarowe i przeciwwłamaniowe, ale również coraz liczniejsze „inteligentne" sprzęty gospodarstwa domowego. Dynamiczny rozwój Internetu Rzeczy (ang. *Internet of Things*) powoduje lawinowy wzrost ilości danych zbieranych z różnorodnych urządzeń [11]. Bardzo szybko rośnie również rozmiar danych gromadzonych przez serwisy internetowe zbierające informacje o swoich użytkownikach, ich działaniach, zwyczajach i upodobaniach. Wydajność procesu przekazywania danych jest krytyczna we wszystkich tych zastosowaniach. Problemy szeregowania zadań w sieciach zbierających dane stają się więc coraz istotniejsze.

Za początek współczesnych badań dotyczących bezprzewodowych sieci czujników uznaje się projekt *Distributed Sensor Networks* rozpoczęty przez amerykańską agencję rządową DARPA (ang. *Defense Advanced Research Projects Agency*, Agencja Zaawansowanych Projektów Ba-

dawczych w Obszarze Obronności) około 1980 roku [10]. Głównym kierunkiem początkowych badań nad zbieraniem danych w takich sieciach było projektowanie protokołów komunikacji opartych na technice wielodostępu z podziałem czasowym (TDMA), przydzielających poszczególnym węzłom sieci pewne okna czasowe o ustalonej z góry długości [17, 22, 30, 44]. Problemy tego typu są wciąż intensywnie badane, powstają nowe, coraz lepiej oddające rzeczywistość modele oraz oparte na nich algorytmy [1, 23, 26, 41]. Niezaprzeczalną zaletą takiego podejścia jest możliwość niemal natychmiastowego zastosowania opracowanych metod w praktyce. Z reguły nie jest wymagana wiedza o ilości gromadzonych danych lub częstotliwości ich napływania. Opracowywane protokoły biorą pod uwagę istotne cechy sieci bezprzewodowych, jak słaba moc obliczeniowa i ograniczona pojemność baterii czujników. Węzeł sieci, który w danym oknie czasowym nie ma przesyłać danych, może przejść w tryb uśpienia, co obniży jego zużycie energii. Z drugiej jednak strony, specyficzna organizacja komunikacji narzuca pewne ograniczenia, które mogą uniemożliwić pełne wykorzystanie wiedzy o parametrach konkretnej sieci lub zbieranych przez nią danych. Co więcej, czynniki istotne dla sieci bezprzewodowych często nie występują w sieciach przewodowych. W przypadku gromadzenia wyników obliczeń w systemie rozproszonym dzielenie komunikacji z poszczególnymi procesorami na dużą liczbę bardzo krótkich wiadomości nie ma sensu. Minimalizacja zużycia energii przeznaczonej na komunikację nie jest potrzebna, tym bardziej że komputery zwykle nie przechodzą w stan uśpienia, kiedy nie przesyłają danych. Algorytmy opracowane dla sieci bezprzewodowych nie są więc uniwersalnymi narzędziami pozwalającymi sprawnie zarządzać zbieraniem danych. Podejście bardziej ogólne i bliższe klasycznej teorii szeregowania zadań, pozwalające na dowolne szeregowanie komunikacji i obliczeń zamiast wykorzystania ustalonych okien czasowych, może być zastosowane zarówno do systemów przewodowych, jak i bezprzewodowych.

Algorytmy szeregowania zbierania danych ogólnego przeznaczenia były początkowo rozwijane na podstawie teorii zadań jednorodnie podzielnych (ang. *divisible load theory*). W teorii tej przyjmuje się, że dane mogą być dzielone na części dowolnych (niekoniecznie całkowitych) rozmiarów, przetwarzane niezależnie w systemie rozproszonym [4, 14, 38, 39]. Zastosowania teorii zadań jednorodnie podzielnych obejmują między innymi modelowanie przetwarzania danych pomiarowych [6], wyszukiwania wzorców [15] i przetwarzania obrazów [31]. Metodologia teorii zadań jednorodnie podzielnych została zastosowana do analizy sieci zbierających dane w pracach [9, 25, 37, 40]. Rozważano w nich problem podziału wolumenu zbieranych danych pomiędzy czujniki przeprowadzające pewne pomiary, w celu minimalizacji całkowitego czasu pozyskiwania danych. W artykule [37] przedstawiono optymalne rozwiązania dla sieci gwiaździstych z różnymi strategiami przeprowadzania pomiarów i komunikacji, natomiast w [9, 25] analizowano sieci o topologii drzewa wysokości dwa. W pracy [40] zaprojektowano algorytmy podziału wolumenu danych pomiędzy węzły, a następnie poszerzono je o możliwość usypiania czujników w celu zmniejszenia zużycia energii.

Możliwości praktycznego zastosowania powyższych badań są niestety ograniczone, ponieważ w większości przypadków ilości danych zbieranych przez poszczególne węzły nie mogą być dowolnie dobierane. Zależą one od czynników zewnętrznych, takich jak wielkość obszaru znaj-

dującego się w zasięgu danego czujnika czy liczba wystąpień wydarzeń, które należy odnotować. Rozmiarów pozyskiwanych danych zazwyczaj nie można dowolnie wybrać także w przypadku, gdy są one rezultatami obliczeń, gdyż rozmiar wyników nie musi być prostą funkcją ilości danych wejściowych. Zagadnienia szeregowania zbierania danych o ustalonych rozmiarach nie były rozważane we wcześniejszej literaturze. Postawiłam więc sobie za cel sformułowanie i analizę takich problemów oraz skonstruowanie rozwiązujących je algorytmów. Po opracowaniu długofalowego planu badań złożyłam wniosek o ich finansowanie i w 2017 roku otrzymałam grant Narodowego Centrum Nauki, co utwierdziło mnie w przekonaniu o słuszności podjętego wyboru obszaru badawczego.

Artykuły składające się na prezentowane osiągnięcie naukowe stanowią cykl badań poświęconych szeregowaniu zadań w sieciach zbierających dane. Ponieważ występujące w praktyce sieci mają różną specyfikę, a wykorzystujące je podmioty dążą do różnych celów, w swoich pracach rozważam zróżnicowane modele sieci i rozmaite kryteria optymalizacji. Każda z prac zawiera zarówno część teoretyczną, jak i eksperymentalną. Wyniki teoretyczne obejmują analizę złożoności obliczeniowej rozważanych problemów, konstrukcję algorytmów dokładnych i przybliżonych oraz analizę ich złożoności i gwarancji jakości otrzymywanych rozwiązań. W części eksperymentalnej zaproponowane algorytmy są implementowane w języku C++ i testowane za pomocą eksperymentów obliczeniowych. Na podstawie uzyskanych wyników formułowane są wnioski dotyczące wyboru najlepszych w praktyce algorytmów rozwiązujących dany problem.

4.2 Modele sieci zbierających dane

Najprostszą i najbardziej rozpowszechnioną topologią sieci zbierających dane jest gwiazda. W tym punkcie przedstawię podstawowy model sieci gwiaździstej oraz jego warianty rozważane w pracach [H1] – [H8]. Znacząca część artykułu [H8] poświęcona jest problemowi szeregowania zbierania danych w sieci o topologii drzewa wysokości dwa, która zostanie opisana w rozdziale 4.6. Wykorzystywana przeze mnie notacja ewoluowała na przestrzeni lat, w związku z czym oznaczenia pojawiające się w różnych pracach nie zawsze są spójne. Dla zwiększenia czytelności niniejszego autoreferatu, używam w nim ujednoliconej notacji, a informacje o ewentualnych rozbieżnościach podaję, opisując wyniki uzyskane w poszczególnych pracach.

Sieć o topologii gwiazdy składa się z m węzłów z danymi, oznaczanych przez $P_1, \ldots, P_m$, oraz jednej stacji bazowej P_0. Węzeł P_j, gdzie $1 \leqslant j \leqslant m$, pozyskuje w chwili r_j zestaw danych D_j o rozmiarze α_j, który musi przesłać bezpośrednio do stacji bazowej. Wartość r_j nazywamy czasem gotowości zestawu danych D_j. W części prac zakładam, że wszystkie zestawy danych są gotowe w momencie rozpoczęcia ich zbierania, czyli $r_j = 0$ dla wszystkich j. W pracy [H1] wyjątkowo rozważam scenariusz, w którym węzły nie posiadają zestawów danych ustalonych rozmiarów, lecz zbierają dane w trybie ciągłym, z ustaloną prędkością.

W każdym momencie stacja bazowa może komunikować się z co najwyżej jednym węzłem. Tego typu założenia są często wykorzystywane w modelowaniu rozproszonych sieci komputerowych, w celu uniknięcia przyjmowania nadmiernie optymistycznych założeń dotyczących możliwości zrównoleglania komunikacji. W przypadku sieci bezprzewodowych taka organizacja

przesyłania danych dodatkowo pozwala na uniknięcie komplikacji spowodowanych interferencją fal. W zależności od rozważanego problemu, przyjmuję, że każdy zestaw danych musi być przekazany jako jeden komunikat lub że przerwanie przesyłania zestawu i jego późniejsze wznowienie jest możliwe. Wykorzystuję afiniczny model komunikacji. Łącze pomiędzy węzłem P_j a stacją bazową jest opisane przez dwa parametry: czas inicjalizacji komunikacji s_j oraz odwrotność prędkości komunikacji c_j. Zatem przesłanie w jednej wiadomości danych rozmiaru α zajmuje czas $s_j + c_j\alpha$. W przypadku sieci, w których czasy inicjalizacji komunikacji są bardzo małe, zasadne jest ich pominięcie, czyli przyjęcie, że $s_j = 0$ dla wszystkich j. Mamy wtedy do czynienia z liniowym modelem komunikacji. Różnice między tymi dwoma modelami są szczególnie istotne w przypadku, kiedy przesyłanie zestawu danych można przerywać. W modelu liniowym przerwania nie wpływają na łączny czas przesyłania danych. Natomiast w modelu afinicznym każdy dodatkowy komunikat zawierający część określonego zestawu danych D_j powoduje wydłużenie łącznego czasu przesyłania tego zestawu o s_j. Oznacza to, że choć przerwania są możliwe, ich występowanie pociąga za sobą pewną karę. Sieć, w której parametry wszystkich łącz są identyczne, czyli $s_j = s$ i $c_j = c$ dla wszystkich $j = 1, \ldots, m$, nazywamy homogeniczną. Z kolei sieć, w której parametry łącz różnią się między sobą, nazywamy heterogeniczną.

Dane przesłane do stacji bazowej w wielu przypadkach muszą zostać przez nią przetworzone. Rozpoczęcie przetwarzania zestawu danych D_j jest możliwe dopiero po zakończeniu odbierania go przez stację bazową. Stacja bazowa w każdym momencie przetwarza co najwyżej jeden zestaw danych, ale może jednocześnie przetwarzać pewien zestaw i odbierać inny. Prędkość obliczeń stacji bazowej wynosi $1/a$, a zatem zestaw D_j jest przetwarzany w czasie $a\alpha_j$. W części prac rozważam problemy, w których przetwarzanie danych nie jest konieczne, czyli $a = 0$.

W rzeczywistych sieciach zbierających dane występuje wiele dodatkowych możliwości i ograniczeń. Czujniki w sieciach bezprzewodowych dysponują z reguły małą pamięcią. Z drugiej strony, jeśli ogromne dane są zbierane z bardzo dużej liczby węzłów, to pamięć stacji bazowej może być niewystarczająca do przechowania ich wszystkich naraz. Kompresja danych umożliwia przyspieszenie ich przesyłania. Wydajność sieci komunikacyjnej może zmieniać się w czasie, w zależności od jej obciążenia przez różne aplikacje konkurujące o te same zasoby. Ujęcie tych praktycznych aspektów wymaga poszerzenia modelu sieci o dodatkowe parametry i założenia. Zostaną one szczegółowo omówione w punktach 4.3 – 4.6. Rozważane przeze mnie problemy szeregowania zadań są problemami optymalizacyjnymi, a więc polegają na znalezieniu uszeregowania najlepszego ze względu na ustalone kryterium. Bardzo naturalnym kryterium jest minimalizacja łącznego czasu zbierania i przetwarzania danych. Oprócz niego analizuję także maksymalizację czasu życia sieci, minimalizację kosztu kompresji danych oraz minimalizację maksymalnego opóźnienia zestawu danych. Kryteria te zostaną dokładniej opisane w dalszych punktach.

4.3 Maksymalizacja czasu życia sieci [H1]

W pracy [H1] rozważam sieć identycznych czujników stale prowadzących pomiary, których wyniki są przesyłane do stacji bazowej, ale nie wymagają dalszego przetwarzania (czyli $a = 0$). Zgodnie z metodologią teorii zadań jednorodnie podzielnych przyjmuję, że dane pomiarowe napływają do każdego czujnika w sposób ciągły, ze stałą prędkością $1/A$. Każdy z czujników ma ograniczoną pamięć rozmiaru B, w której zapisywane są pozyskane dane. W momencie przesłania ich do stacji bazowej zaalokowana pamięć jest zwalniana i można zapisać w niej kolejne dane. Zwalnianie pamięci odbywa się w sposób ciągły, z prędkością równą prędkości przesyłania danych do stacji bazowej. Opisany model znajduje zastosowanie w sytuacjach, kiedy pomiary przeprowadzane są z bardzo dużą częstotliwością. Przykładowo, podczas monitorowania eksperymentów fizycznych i chemicznych, w ciągu sekundy mogą zostać wykonane tysiące pomiarów [12]. Choć czujniki są identyczne, to znajdują się w różnych lokalizacjach, więc ich parametry komunikacyjne s_j i c_j (w [H1] oznaczane przez S_j i C_j) różnią się między sobą. Sieć zbierająca dane działa poprawnie do chwili, kiedy któremuś z czujników zabraknie wolnej pamięci do przechowywania napływających danych. Długość tego okresu poprawnego działania nazywana jest czasem życia sieci. Celem badań przedstawionych w [H1] było opracowanie algorytmu szeregowania komunikacji między czujnikami a stacją bazową maksymalizującego czas życia sieci. Przyjęłam założenie, że komunikacja odbywa się w rundach oraz że w każdej rundzie ilości danych przesyłane przez poszczególne czujniki są sobie równe.

Pierwszym krokiem było obliczenie maksymalnej ilości danych α^*, jaką można przesłać z pojedynczego czujnika w jednej rundzie, przy założeniu, że dana jest kolejność, w jakiej czujniki komunikują się ze stacją bazową. Naturalnie, ilość ta zależy od rozmiaru danych znajdujących się w pamięci czujników na początku rundy. Pokazałam, że czynnikiem decydującym o sposobie wyliczenia α^* jest zależność między prędkością gromadzenia danych a prędkościami komunikacji pierwszego i ostatniego czujnika w sekwencji komunikacji. Wyróżniłam cztery przypadki i dla każdego z nich wyprowadziłam wzór na α^*.

Na podstawie powyższego wyniku można określić optymalną kolejność komunikacji czujników w czasie $O(m^2)$, analizując wszystkie możliwości wyboru czujników aktywowanych jako pierwszy i ostatni. Pokazałam jednak, że wybór ten może być dokonany w czasie $O(m)$, ponieważ między czterema wyróżnionymi przypadkami zachodzą pewne relacje dominacji. Opierając się na tym spostrzeżeniu, przedstawiłam kompletny algorytm szeregowania jednej rundy komunikacji działający w czasie $O(m)$.

Eksperymentalna część pracy poświęcona była dwóm zagadnieniom. Pierwszym z nich był wpływ prędkości komunikacji czujników na łączny rozmiar danych zgromadzonych w czasie życia sieci. Wygenerowałam instancje testowe zawierające $m = 100$ węzłów o buforze pamięci rozmiaru $B = 1\text{E}6$ i odwrotności prędkości gromadzenia danych $A = 1\text{E}{-5}$. Czas inicjalizacji komunikacji wynosił $s_j = 1\text{E}{-3}$ dla wszystkich węzłów. Dla uproszczenia analizy przyjęłam, że odwrotność prędkości komunikacji przez najwolniejsze łącze wynosi c_{max}, a wszystkie pozostałe łącza charakteryzuje wspólna odwrotność prędkości przesyłania danych c_{other}. Rozważałam wartości $c_{max} \in \{1,5\text{E}{-5};\ 1{,}1\text{E}{-5};\ 9\text{E}{-6};\ 5\text{E}{-6}\}$ oraz c_{other} z zakresu od $1\text{E}{-8}$

do $\min\{c_{max}, 1E{-}5\}$. Pokazałam, że prędkość najwolniejszego czujnika wpływa nie tylko bezpośrednio na rozmiar zebranych danych, ale również na to, jak duże znaczenie mają zmiany prędkości pozostałych czujników. Drugim analizowanym zagadnieniem był sposób, w jaki zmieniają się ilości danych zbieranych w kolejnych rundach przy różnych prędkościach komunikacji poszczególnych czujników. Przedstawiłam wyniki dla czterech kombinacji wartości parametrów c_{max} i c_{other}: (1,1E−5, 1E−8), (9E−6, 2E−8), (5E−6, 1E−7) oraz (5E−6, 2E−8). Dla każdego z nich wykres obrazujący ilości danych przesyłanych w kolejnych rundach miał inny kształt: przypominający literę V, M lub Λ, lub stabilizujący się na pewnym ustalonym rozmiarze danych. Wyjaśniłam, w jaki sposób zmiany aktywnych wyrażeń we wzorach opisujących ilości danych przesyłanych w poszczególnych rundach powodują powstanie takich właśnie kształtów.

W pracy [H1] teoria zadań jednorodnie podzielnych została po raz pierwszy zastosowana do analizy sieci zbierającej stale napływające dane pomiarowe. Kluczowym wynikiem było wykazanie, że postać jawnego wzoru opisującego ilość danych przesyłanych w rundzie komunikacji zależy wyłącznie od tego, czy prędkości przekazywania danych przez najszybsze i najwolniejsze łącze są większe, czy mniejsze od prędkości napływania danych. Spostrzeżenie to umożliwiło opracowanie bardzo szybkiego algorytmu szeregowania zadań w rozważanej sieci.

4.4 Minimalizacja kosztu kompresji danych [H2]

Jedną z naturalnych metod zwiększenia wydajności sieci zbierających dane jest kompresowanie danych przed ich przesłaniem [27, 45]. Kompresja pozwala na ograniczenie liczby i długości przesyłanych komunikatów, z drugiej jednak strony, zajmuje pewien czas i może wymagać poniesienia pewnego kosztu, na przykład dodatkowego zużycia energii. W pracy [H2] rozważam sieć, w której każdy węzeł P_j może skompresować zestaw danych D_j, zmniejszając jego rozmiar z α_j do $\gamma\alpha_j$, gdzie $\gamma < 1$ jest stałą. Proces ten zajmuje czas $A\alpha_j$ i generuje koszt $f\alpha_j$. Jeśli węzeł kompresuje swój zestaw danych, to może zacząć go przesyłać dopiero po zakończeniu tej czynności. Czas inicjalizacji komunikacji jest zaniedbywalny, a dane nie są przetwarzane przez stację bazową, a zatem $s_j = 0$ dla wszystkich $j = 1, \ldots, m$ oraz $a = 0$. Czas przesyłania danych określonego rozmiaru z węzła P_j zależy więc wyłącznie od jego odwrotności prędkości komunikacji c_j (w [H2] oznaczanej symbolem C_j). Określony jest maksymalny czas T, w którym wszystkie dane muszą zostać zebrane. Należy skonstruować uszeregowanie długości nieprzekraczającej T minimalizujące łączny koszt kompresji danych.

Rozważany problem sprowadza się do wyboru zbioru węzłów, które powinny skompresować swoje dane. Istotnie, po ustaleniu tego zbioru koszt kompresji jest znany, a najkrótsze uszeregowanie można skonstruować w czasie $O(m \log m)$, rozwiązując znany problem minimalizacji długości uszeregowania na jednym procesorze dla zadań z terminami gotowości, $1|r_j|C_{max}$ [3].

Udowodniłam, że problem optymalnego wyboru węzłów kompresujących dane jest NP-trudny, nawet jeśli

- ilości danych zgromadzonych przez wszystkie węzły są sobie równe, czyli $\alpha_j = \alpha$ dla $j = 1, \ldots, m$ ([H2], Proposition 1); lub
- prędkości przesyłania danych z wszystkich węzłów są sobie równe, a kompresja danych

odbywa się w zerowym czasie, czyli $c_j = c$ dla $j = 1, \ldots, m$ oraz $A = 0$ ([H2], Proposition 3).

W obu przypadkach dowód opierał się na NP-zupełności problemu podziału zbioru [19]. Pokazałam również, że jeśli wszystkie węzły posiadają dane o równych rozmiarach i charakteryzują się równymi prędkościami komunikacji, to powyższy problem można rozwiązać w czasie $O(1)$ ([H2], Proposition 2).

Naturalny algorytm dokładny dla rozważanego problemu, o złożoności $O(2^m m \log m)$, polega na przejrzeniu wszystkich możliwych podzbiorów węzłów kompresujących dane i skonstruowaniu dla każdego z nich najkrótszego uszeregowania. Następnie należy wybrać najtańsze uszeregowanie o długości nieprzekraczającej T, o ile takie istnieje. Pokazałam, że generując podzbiory w porządku minimalnych zmian, można uzyskać nieco szybszy algorytm, działający w czasie $O(2^m m)$. Zaproponowałam zachłanne algorytmy heurystyczne o złożoności $O(m^2)$. Algorytm greedy-$C\alpha$ ma na celu wybór węzłów P_j, dla których wartość $c_j\alpha_j(1-\gamma)$, o którą czas przesyłania danych zostanie skrócony w wyniku kompresji, jest największa. W związku z tym, węzły są przetwarzane w kolejności nierosnących wartości $c_j\alpha_j$. Algorytm greedy-α stara się minimalizować koszty $f\alpha_j$ kompresji danych. Węzły są więc sortowane w porządku niemalejących wartości α_j. W obu algorytmach początkowym rozwiązaniem jest pusty zbiór węzłów kompresujących dane. Węzły analizowane w kolejnych krokach są dołączane do tego zbioru, o ile prowadzi to do skrócenia uszeregowania. Proces ten kończy się w momencie otrzymania uszeregowania długości co najwyżej T lub zakończenia przeglądania całej listy węzłów. Należy tu zwrócić uwagę, że powyższe algorytmy mogą nie znaleźć żadnego rozwiązania dla pewnych instancji, dla których istnieją dopuszczalne uszeregowania. Zaproponowałam także hiperheurystykę [5] greedy-min, która wybiera lepsze z rozwiązań uzyskanych przez algorytmy greedy-$C\alpha$ oraz greedy-α.

Jakość rozwiązań dostarczanych przez zaproponowane heurystyki została przetestowana za pomocą eksperymentów obliczeniowych. W podstawowej konfiguracji generowanych instancji sieć składała się z $m = 20$ węzłów, odwrotność prędkości kompresji danych wynosiła $A = 1\text{E}{-6}$, a koszt kompresji jednostki danych był równy 1. Wartość parametru γ została ustalona na 0,4. Odwrotności prędkości komunikacji c_j były wybierane losowo z przedziału $[1\text{E}{-8}, 1\text{E}{-7}]$, a rozmiary danych α_j z przedziału [1E6, 1E8]. W celu wygenerowania wyłącznie instancji, dla których istnieją uszeregowania długości nieprzekraczającej T, dla każdego zestawu wartości parametrów opisanych powyżej obliczyłam minimalną długość uszeregowania T^*, używając algorytmu enumeracyjnego. Następnie, dla danej wartości δ_T (domyślnie równej 1,1) ustalany był limit czasu $T = \delta_T T^*$. Przeprowadzone eksperymenty dotyczyły wpływu zmian wartości δ_T, γ, A i m na jakość uszeregowań otrzymywanych przez poszczególne heurystyki. Dla każdej rozważanej kombinacji parametrów wygenerowanych i rozwiązanych zostało 100 instancji problemu. Miarą jakości rozwiązań był iloraz kosztu wyliczonego przez analizowany algorytm i optymalnego kosztu zwróconego przez algorytm dokładny. Należy tu zwrócić uwagę, że minimalny koszt kompresji może być dla pewnych instancji zerowy. Jednak w takim przypadku wszystkie zaproponowane heurystyki znajdują optymalne rozwiązanie. W związku

z tym, jakość rozwiązań otrzymanych dla takich testów zdefiniowałam jako równą 1.

Algorytm greedy-$C\alpha$ nie potrafił znaleźć dopuszczalnych uszeregowań dla wielu spośród wygenerowanych testów. Natomiast algorytm greedy-α (i w konsekwencji greedy-min) dostarczył rozwiązania dla zaskakująco wielu instancji. Spośród wszystkich 3700 testów tylko dwa nie zostały rozwiązane przez ten algorytm. Średnia jakość rozwiązań generowanych przez greedy-$C\alpha$ i greedy-α była w większości przypadków zbliżona. Uszeregowania zwracane przez algorytm greedy-min były jednak z reguły zdecydowanie lepsze, co pokazuje, że żaden z algorytmów greedy-$C\alpha$ i greedy-α nie dominuje nad drugim dla ogółu instancji.

Główne wyniki przedstawione w pracy [H2] to sformułowanie nowego problemu szeregowania zadań, wykazanie jego NP-trudności i skonstruowanie, a następnie przetestowanie, wielomianowych heurystyk. Ponieważ algorytmy greedy-$C\alpha$ i greedy-α realizują bardzo odmienne strategie zachłanne, połączenie ich w algorytmie greedy-min doprowadziło do znacznej poprawy jakości otrzymywanych uszeregowań. Warto podkreślić, że przedstawiony problem szybko wzbudził zainteresowanie środowiska naukowego. Bezpośrednią kontynuacją badań opisanych w artykule [H2] są prace [35, 36], w których zaproponowano dalsze algorytmy dla tego problemu oraz dla jego uogólnienia.

4.5 Minimalizacja czasu zbierania danych [H5, H6, H7]

Minimalizacja długości uszeregowania jest najpopularniejszym kryterium optymalizacji analizowanym w teorii szeregowania zadań. W tym punkcie zostaną przedstawione wyniki dotyczące minimalizacji czasu zbierania danych przy uwzględnieniu pewnych dodatkowych ograniczeń wynikających ze specyfiki sieci.

Sieć z czasami gotowości zestawów danych. W artykule [H5] rozważana jest sieć, której węzły pozyskują zestawy danych w różnych momentach, a zatem czasy gotowości r_j są dowolne. Każdy zestaw danych D_j musi najpierw zostać dostarczony do stacji bazowej, a następnie przetworzony przez nią. Dozwolone jest przerywanie i późniejsze wznawianie przesyłania zestawu danych. Ponieważ jednak czasy inicjalizacji komunikacji s_j są niezerowe, przerwania wydłużają łączny czas przekazywania danych (w [H5] w miejsce s_j, c_j i a używane były oznaczenia S_j, C_j i A). Należy znaleźć uszeregowanie, dla którego czas T, w którym wszystkie dane zostaną zgromadzone i przetworzone, jest najkrótszy. Kolejność, w jakiej zestawy danych są przetwarzane przez stację bazową, nie wpływa na długość uszeregowania, o ile nie występują niepotrzebne przestoje. Problem sprowadza się zatem do znalezienia optymalnego uszeregowania komunikacji. Warto zauważyć, że rozważana sieć zbierająca dane jest dwumaszynowym systemem przepływowym (ang. *flow shop*). Istotnie, każdy zestaw danych przechodzi kolejno dwa etapy: przesyłanie i przetwarzanie, a w każdej chwili co najwyżej jeden zestaw może być przekazywany i co najwyżej jeden zestaw może być przetwarzany.

Wykazałam, że problem minimalizacji długości uszeregowania jest silnie NP-trudny, nawet jeśli wszystkie czasy inicjalizacji komunikacji są zerowe, czyli $s_j = 0$ dla wszystkich j ([H5], Proposition 1). Zauważmy, że przy takim założeniu mamy do czynienia ze specjalnym

przypadkiem problemu minimalizacji długości uszeregowania w systemie przepływowym z przerwaniami i czasami gotowości zadań, $F2|pmtn, r_j|C_{max}$ [8], w którym czasy wykonywania poszczególnych operacji nie są dowolne. Dowód silnej NP-trudności został więc skonstruowany w sposób analogiczny jak w [8], w oparciu o silną NP-zupełność problemu trójpodziału [19]. Pokazałam, że w ogólnym przypadku (dla $s_j \neq 0$) skonstruowanie optymalnego uszeregowania może wymagać przerwania przesyłania pewnego zestawu danych w momencie, kiedy żaden inny zestaw nie uzyskuje gotowości ([H5], Proposition 2). W konsekwencji, najprawdopodobniej nie jest możliwe stworzenie dokładnego algorytmu opartego na analizowaniu różnych wyborów momentów przerwania komunikacji, ponieważ nie dysponujemy skończoną listą potencjalnych punktów przerwań.

Zaprojektowałam rodzinę działających w czasie $O(m \log m)$ algorytmów przybliżonych J(γ), gdzie $\gamma \in [0, \infty]$ jest parametrem. Algorytm J(γ) opiera się na zastosowaniu reguły Johnsona konstruującej optymalne rozwiązanie problemu minimalizacji długości uszeregowania w dwumaszynowym systemie przepływowym bez czasów gotowości zadań, $F2||C_{max}$ [24]. Jeśli zestaw danych D_j uzyskuje gotowość w trakcie przesyłania zestawu D_i, to przerwanie komunikacji jest rozważane, o ile zadanie odpowiadające przesłaniu i przetworzeniu zestawu D_j poprzedza według reguły Johnsona zadanie odpowiadające przesłaniu pozostałej części zestawu D_i i przetworzeniu go. Przerwanie jest wykonywane, jeśli dodatkowo $s_i < \gamma p_{1i}(r_j)$, gdzie $p_{1i}(r_j)$ jest czasem pozostałym w chwili r_j do zakończenia przesyłania zestawu D_i. Oznacza to, że dodatkowy czas inicjalizacji komunikacji wynikający z ewentualnego przerwania nie jest duży w porównaniu z czasem, który pozostał do zakończenia przesyłania zestawu D_i. Wartość parametru γ określa zatem, jak duży koszt wykonania przerwania jest akceptowany. Wykazałam, że dla każdego $\gamma \in [0, \infty]$ heurystyka J(γ) jest algorytmem 2-aproksymacyjnym ([H5], Proposition 3). Pokazałam też, że jeśli $\gamma = 0$ lub $\gamma = \infty$, to oszacowanie to jest ścisłe, czyli istnieją instancje, dla których algorytm zwraca uszeregowanie dokładnie dwa razy dłuższe niż optymalne ([H5], Proposition 4 i 5).

Jakość rozwiązań otrzymywanych dla $\gamma \in \{0;\ 0{,}05;\ 0{,}1;\ 0{,}2;\ \infty\}$ została przeanalizowana za pomocą eksperymentów obliczeniowych. Wygenerowałam instancje testowe, w których liczba węzłów m należała do zbioru $\{10, 20, \ldots, 100\}$, odwrotność prędkości przetwarzania danych wynosiła $a = 1$, a rozmiary danych były losowane z przedziału $[1, 100]$. Wyniki wstępnych eksperymentów pokazały, że jeśli wartości c_j nie są bardzo zróżnicowane lub jeśli wszystkie są zdecydowanie mniejsze (albo większe) niż a, to bardzo łatwo jest znaleźć uszeregowanie bliskie optymalnemu. W celu otrzymania wymagających instancji wartości c_j były więc obliczane jako $c_j = t_j/\alpha_j$, gdzie t_j było losowane z przedziału $[1, 100]$. Dla ustalonej wartości $s_{max} \in \{0, 1, \ldots, 10\}$ czasy inicjalizacji komunikacji s_j były wybierane losowo z zakresu $[0, s_{max}]$. Czas gotowości pierwszego zestawu danych był równy $r_1 = 0$, natomiast kolejne czasy gotowości były obliczane ze wzoru $r_j = r_{j-1} + \delta_j \frac{t_C}{m}$, gdzie $t_C = \sum_{j=1}^{m}(s_j + c_j\alpha_j)$, a wartości δ_j były wybierane losowo z przedziału [0,5; 1,5]. Dla każdej kombinacji wartości m i s_{max} wygenerowałam 100 instancji. Ponieważ skonstruowanie optymalnych rozwiązań nie było możliwe, miarą jakości uszeregowania był iloraz jego długości i ograniczenia dolnego LB, obliczonego

w sposób analogiczny do metody zaproponowanej dla problemu $F2|r_j|C_{max}$ w pracy [42].

Okazało się, że zaproponowane algorytmy tworzą bardzo dobre rozwiązania. Dla wszystkich rozważanych kombinacji parametrów m i s_{max} długości otrzymywanych uszeregowań były w średnim przypadku większe od LB o mniej niż 3,5%. Co więcej, odpowiedni dobór wartości γ gwarantował w każdym przypadku otrzymanie średniego błędu względnego poniżej 1,5%. Eksperymenty pokazały również, że oba parametry m i s_{max} wpływają na to, jaką wartość γ należy przyjąć, aby uzyskać najkrótsze uszeregowania. Każdy z rozważanych wariantów algorytmu J(γ) wygenerował najkrótsze uszeregowania dla pewnego zestawu wartości tych dwóch parametrów. W ogólności, najlepsze rozwiązania uzyskiwał algorytm J(0.1).

Głównym wynikiem w pracy [H5] jest skonstruowanie algorytmów J(γ) i analiza ich efektywności. Wykazałam, że współczynnik aproksymacji algorytmów J(γ) wynosi 2, jednak dowód ścisłości tego oszacowania dla $\gamma \in \{0, \infty\}$ opiera się na specyficznych instancjach, w których sieć zawiera tylko dwa węzły. Wyniki eksperymentów obliczeniowych sugerują, że dla sieci składających się z wielu węzłów i posiadających realistyczne wartości parametrów s_j i c_j, uszeregowania konstruowane przez zaproponowane algorytmy są nieporównywalnie lepsze. Z praktycznego punktu widzenia istotna jest również obserwacja, że algorytm J(0.1) tworzy bardzo dobre rozwiązania niezależnie od przyjętych parametrów sieci. Oznacza to, że można go bezpiecznie wykorzystywać także w przypadku, kiedy wartości tych parametrów nie są znane.

Sieć z ograniczoną pamięcią stacji bazowej. Artykuł [H6] dotyczy minimalizacji czasu zbierania danych w homogenicznej sieci z zerowymi czasami inicjalizacji komunikacji i ograniczoną pamięcią stacji bazowej. Każdy zestaw danych D_j musi więc zostać najpierw przekazany do stacji bazowej, co zajmuje czas $c\alpha_j$, a następnie przetworzony przez nią w czasie $a\alpha_j$ (w [H6] używane były oznaczenia C i A w miejsce c i a). Przerywanie przesyłania lub przetwarzania zestawu danych nie jest dozwolone. W momencie rozpoczęcia przesyłania zestawu D_j stacja bazowa alokuje blok pamięci rozmiaru α_j. Zostaje on zwolniony po zakończeniu przetwarzania tego zestawu. Stacja bazowa dysponuje ograniczoną pamięcią rozmiaru B, a zatem łączny rozmiar zaalokowanych bloków nie może w żadnej chwili przekroczyć B. Zestawy danych są przetwarzane w takiej kolejności, w jakiej docierają do stacji bazowej, a więc problem szeregowania sprowadza się do ustalenia kolejności przesyłania zestawów. Rozważana sieć jest zatem dwumaszynowym permutacyjnym systemem przepływowym z ograniczoną pamięcią. Rozmaite zagadnienia szeregowania zadań w takim systemie doczekały się dużego zainteresowania społeczności naukowej [7, 18, 28, 29, 32, 33, 48]. Problem analizowany w [H6] jest specjalnym przypadkiem problemu relokacji z dodatkową załogą odzyskującą zasoby, $F2|rp|C_{max}$ [7, 33].

Korzystając z silnej NP-zupełności problemu pakowania (ang. *bin packing*) [19], udowodniłam silną NP-trudność problemu minimalizacji długości uszeregowania w opisanej powyżej sieci ([H6], Proposition 1). Ponadto pokazałam, że optymalne rozwiązanie można znaleźć w czasie wielomianowym, jeśli rozmiar pamięci B jest wystarczająco mały lub wystarczająco duży. Wykazałam także, że długość optymalnego uszeregowania nie zmienia się, kiedy wartości odwrotności prędkości przesyłania danych c i odwrotności przetwarzania danych a zostają zamienione

ze sobą. Upraszcza to analizę problemu, gdyż wystarczy rozważać przypadek $a \geqslant c$. Optymalne uszeregowanie można znaleźć za pomocą algorytmu całkowitoliczbowego programowania liniowego dla problemu $F2|rp|C_{max}$, przedstawionego w pracy [7]. Ponieważ wysoka złożoność obliczeniowa tego algorytmu uniemożliwia stosowanie go w praktyce dla dużych instancji, zaprojektowałam cztery heurystyki o złożoności $O(m \log m)$. Algorytm Inc przesyła zestawy danych w kolejności niemalejących rozmiarów α_j, natomiast algorytm Alter przesyła naprzemiennie największy i najmniejszy z dostępnych zestawów. Heurystyka LF opiera się na technice zachłannej: przesyłany jest zawsze największy zestaw danych, który mieści się w dostępnej w danym momencie pamięci. Algorytm Rnd wysyła zestawy danych w losowej kolejności. Został on zaimplementowany w celu sprawdzenia, na ile dobre są rozwiązania tworzone przez pozostałe heurystyki. Zaproponowałam także dwa typy algorytmów przeszukiwania lokalnego. Algorytmy pierwszego typu używają sąsiedztw opartych na zamianie pozycji (ang. *swap*) pary zestawów danych w uszeregowaniu, natomiast w algorytmach drugiego typu sąsiedztwo jest konstruowane przez przesuwanie (ang. *shift*) pojedynczych zadań na inną pozycję w uszeregowaniu. Wyniki uzyskane przez algorytm przeszukiwania lokalnego zależą oczywiście od wyboru początkowego rozwiązania. Było ono dostarczone przez jedną z czterech opisanych wyżej heurystyk. Analizowałam więc łącznie osiem wariantów algorytmów przeszukiwania lokalnego. Ich nazwy odzwierciedlały sposób generowania początkowego rozwiązania i wykorzystywaną definicję sąsiedztwa. Dla przykładu, algorytm LFSwap zaczynał poszukiwania od uszeregowania otrzymanego przez heurystykę LF i wykorzystywał sąsiedztwo oparte na zamianach par zadań. Zauważmy, że w rozważanym problemie długość dowolnego uszeregowania bez niepotrzebnych przestojów jest co najwyżej $1 + \min\{c/a, a/c\}$ razy większa od optimum. Zatem instancje, w których $c = a$, są w pewnym sensie najtrudniejsze do rozwiązania, a wszystkie zaprojektowane algorytmy mają współczynnik aproksymacji nieprzekraczający 2.

Wyniki otrzymywane przez poszczególne algorytmy zostały przeanalizowane za pomocą eksperymentów obliczeniowych. Wygenerowałam instancje z parametrami sieci $c = 1$ oraz $a \in \{1, 2\}$. Rozmiary zestawów danych α_j były wybierane losowo z przedziału $[1, 1 + \delta_\alpha]$ dla $\delta_\alpha \in \{0{,}5;\ 1;\ 1{,}5;\ 2\}$. Rozmiar pamięci stacji bazowej wynosił $B = \delta_B B_{min}$, gdzie $\delta_B = 1 + i/10$ dla $i = 1, \ldots, 8$, a $B_{min} = \min_{i \neq j}\{\alpha_i + \alpha_j\}$ to minimalny rozmiar bufora pozwalający na przechowanie w nim więcej niż jednego zestawu danych. Małe instancje testowe zawierały $m = 10$ węzłów, natomiast w dużych instancjach sieć składała się z $m = 100$ węzłów. Dla małych instancji miarą jakości uszeregowania był iloraz jego długości i wartości optymalnej obliczonej za pomocą algorytmu dokładnego. Skonstruowanie optymalnych rozwiązań dla dużych instancji nie było możliwe ze względu na zbyt wysoką złożoność tego algorytmu. Dla takich testów jakość uszeregowania była więc mierzona przez iloraz jego długości i ograniczenia dolnego *LoBo*, uzyskanego przez pominięcie ograniczenia pamięci i obliczenie długości najkrótszego uszeregowania za pomocą algorytmu Johnsona [24]. Dla każdego rozważanego zestawu wartości parametrów zostało wygenerowanych 100 instancji.

Spośród zaproponowanych prostych heurystyk najlepsze uszeregowania konstruował zachłanny algorytm LF, którego średni błąd dla małych testów nie przekraczał 4%. Algorytmy

Inc i Alter generowały znacznie gorsze rozwiązania, a dla pewnych wartości parametrów instancji działały nawet gorzej niż algorytm losowy Rnd. Wszystkie algorytmy przeszukiwania lokalnego uzyskały w średnim przypadku bardzo dobre uszeregowania, o mniej niż 2% dłuższe od optymalnych. Średni względny błąd najlepszego algorytmu IncShift wynosił poniżej 0,5% dla wszystkich rozważanych typów małych instancji. Wyniki uzyskane dla dużych instancji były zdeterminowane przez zmieniającą się odległość ograniczenia dolnego $LoBo$ od faktycznego optimum. Zależności między rozwiązaniami dostarczanymi przez poszczególne algorytmy nie różniły się jednak od tych zaobserwowanych dla małych instancji. Ciekawym wnioskiem z przeprowadzonych eksperymentów jest to, że wykorzystanie sąsiedztw opartych na przesunięciach pojedynczych zadań w uszeregowaniu prowadzi do lepszych wyników niż użycie sąsiedztw opartych na zamianach par zadań.

Praca [H6] prezentuje nowe zastosowanie problemu $F2|rp|C_{max}$ do analizy sieci zbierających dane. Dowód silnej NP-trudności rozważanego zagadnienia stanowi wkład w analizę złożoności obliczeniowej problemów minimalizacji długości uszeregowania w dwumaszynowym systemie przepływowym z ograniczoną pamięcią, których pełna klasyfikacja została później przedstawiona w [48]. Z praktycznego punktu widzenia najistotniejsze wydaje się skonstruowanie algorytmów przeszukiwania lokalnego znajdujących bardzo dobre uszeregowania w krótkim czasie. Wyniki opisane w [H6] stały się inspiracją i motywacją do dalszych badań nad systemem przepływowym z ograniczoną pamięcią, które prowadzę obecnie we współpracy z Yakovem Zinderem i Alexandrem Kononovem [P10, P16].

Sieć ze zmienną prędkością komunikacji. Choć w klasycznej teorii szeregowania zadań przyjmuje się założenie, że czasy wykonywania zadań są ustalone i znane z góry, wiele współczesnych badań dotyczy sytuacji, kiedy czas trwania zadania zależy od pewnych czynników, jak na przykład moment rozpoczęcia jego wykonywania [20]. We wpisującej się w ten nurt pracy [H7] rozważana jest minimalizacja czasu zbierania danych w sieci, w której łącza komunikacyjne są współdzielone z różnymi aplikacjami, przez co prędkość przesyłania przez nie danych nie jest stała. Prędkość komunikacji przez obciążone łącze jest $\delta > 1$ razy mniejsza niż w przypadku nieobciążonego łącza. Dla każdego węzła P_j dana jest lista n_j przedziałów czasu, w których wykorzystywane przez niego łącze jest obciążone. Łączna liczba takich przedziałów wynosi n. Dane zebrane przez stację bazową nie są przetwarzane, czyli $a = 0$, a czasy inicjalizacji komunikacji są pomijalne, a zatem $s_j = 0$ dla wszystkich j. W ogólnym przypadku, prędkości komunikacji przez nieobciążone łącza $1/c_j$ mogą być różne dla różnych węzłów, jednak sytuację taką można sprowadzić do przypadku równych prędkości za pomocą skalowania rozmiarów zestawów danych α_j. Wystarczy więc rozważać sieć homogeniczną, w której przesłanie danych rozmiaru α przez nieobciążone łącze zajmuje czas α, a przekazanie tej samej ilości danych przez obciążone łącze wymaga czasu $\delta\alpha$.

Rozważałam zarówno przypadek, kiedy możliwe jest przerywanie przesyłania poszczególnych zestawów danych, jak i scenariusz, w którym przerwania nie są dozwolone. Dla problemu z przerwaniami skonstruowałam dokładny algorytm wielomianowy, oparty na programowaniu

liniowym i wyszukiwaniu binarnym. Opierając się na silnej NP-zupełności problemu trójpodziału, wykazałam, że jeśli przerwania nie są dozwolone, to problem minimalizacji długości uszeregowania w sieci ze zmienną prędkością komunikacji jest silnie NP-trudny ([H7], Proposition 2). W związku z tym, dalsze badania dotyczyły właśnie tego wariantu problemu. Skoro przerwania nie są możliwe, polega on na ustaleniu kolejności przesyłania zestawów danych. Zaproponowałam dokładny algorytm programowania dynamicznego działający w czasie $O((m+n)2^m)$. Wskazałam też na własności problemu, które sprawiają, że poprawienie złożoności tego algorytmu dzięki wykorzystaniu informacji o strukturze optymalnych rozwiązań nie wydaje się w ogólnym przypadku możliwe ([H7], Proposition 4 i 5).

Następnie przedstawiłam trzy zachłanne heurystyki o złożoności $O(m(m+n))$. Algorytm gTime kieruje się zasadą minimalizacji czasu wysyłania kolejnego zestawu danych, algorytm gRate – maksymalizacji średniej prędkości przesyłania danych, a gSlowtime – minimalizacji czasu przesyłania danych przez obciążone łącze. Wykazałam, że każdy z tych algorytmów ma współczynnik aproksymacji δ i jest to oszacowanie dokładne ([H7], Proposition 6, 7, 8). Ponadto udowodniłam, że między tymi heurystykami nie zachodzą relacje dominacji, a dokładniej, że dla każdego z tych algorytmów istnieją instancje, dla których konstruuje on optymalne rozwiązania, a pozostałe dwa algorytmy ich nie znajdują ([H7], Proposition 9, 10, 11). Przedstawiłam także algorytmy przeszukiwania lokalnego wykorzystujące sąsiedztwa oparte na zamianie pozycji dwóch zestawów danych w uszeregowaniu. Każdy z tych algorytmów jako początkowe rozwiązanie wykorzystywał uszeregowanie wygenerowane przez jedną z heurystyk zachłannych, wskazywaną przez nazwę algorytmu. Dla przykładu, gRateLocal to algorytm przeszukiwania lokalnego rozpoczynającego się od rozwiązania dostarczonego przez heurystykę gRate. W celu dokładniejszej eksperymentalnej oceny efektywności powyższych algorytmów zaimplementowałam także algorytm Rnd przesyłający zestawy danych w losowej kolejności i algorytm RndLocal realizujący przeszukiwanie lokalne rozpoczynające się od losowego uszeregowania.

Wygenerowałam zbiór małych instancji testowych o liczbie węzłów $m \in \{1, \ldots, 25\}$ oraz zbiór dużych instancji, w których $m \in \{10, 20, \ldots, 100\}$. We wszystkich instancjach przyjęłam $\delta = 2$. Rozmiary danych α_j były losowane z przedziału $[1, 20]$. Częstotliwość zmian obciążenia łącz była kontrolowana przez parametry $F, L \in \{1, \ldots, 5\}$. W chwili 0 wszystkie łącza były nieobciążone. Długość pierwszego przedziału, w którym j-te łącze było nieobciążone, była losowana z zakresu $[0, F\sum_{i=1}^{m} \alpha_i/m]$. Następnie, długość pierwszego przedziału, w którym j-te łącze było obciążone, była losowana z zakresu $[0, L\sum_{i=1}^{m} \alpha_i/m]$. Długości kolejnych przedziałów były losowane analogicznie, aż do osiągnięcia horyzontu czasowego $\delta\sum_{i=1}^{m} \alpha_i$. Procedura ta była wykonywana niezależnie dla każdego łącza. Dla każdej analizowanej kombinacji parametrów wygenerowałam 100 instancji. Miarą jakości rozwiązań dla małych instancji był względny błąd w stosunku do optymalnej długości uszeregowania, obliczonej przez algorytm programowania dynamicznego. Uzyskanie rozwiązań optymalnych dla dużych instancji nie było możliwe, więc w ich przypadku miarą jakości uszeregowań był względny błąd w stosunku do oszacowania dolnego $\sum_{i=1}^{m} \alpha_i$.

Wyniki eksperymentów obliczeniowych pokazały, że najlepszą spośród heurystyk zachłan-

nych jest w większości przypadków gRate. Wykorzystanie wyniku działania tego algorytmu jako początkowego rozwiązania w algorytmie przeszukiwania lokalnego również prowadzi do uzyskania lepszych wyników niż wybór uszeregowań tworzonych przez pozostałe heurystyki. Dla małych instancji średni błąd względny algorytmu gRateLocal nie przekraczał 6,5%. Przeszukiwanie lokalne okazało się bardzo efektywną metodą rozwiązywania rozważanego problemu, ponieważ dobre wyniki były uzyskiwane nawet przez algorytm RndLocal, mimo że jakość losowych uszeregowań nie była wysoka. Wyniki otrzymane dla dużych instancji odzwierciedlały różnicę między ograniczeniem dolnym a rzeczywistym optimum. Zależności między jakościami wyników uzyskiwanych przez poszczególne algorytmy były analogiczne, jak w przypadku małych instancji.

Praca [H7] otwiera badania nad zbierającymi dane sieciami, których parametry zmieniają się w czasie. Porzucenie założenia o niezmienności prędkości komunikacji pozwala na tworzenie bardziej realistycznych modeli sieci. Wykazanie, że złożoność rozważanego problemu zależy od dopuszczenia możliwości przerwań, wskazuje na jego pewne podobieństwo do klasycznych problemów, takich jak $P||C_{max}$ i $P|pmtn|C_{max}$ czy $1|r_j|L_{max}$ i $1|pmtn, r_j|L_{max}$ [3]. Skonstruowane algorytmy przeszukiwania lokalnego, a w szczególności gRateLocal, mogą być wykorzystane do efektywnego zbierania danych w praktyce. Warto zauważyć, że algorytmy te nie opierają się w żaden sposób na założeniu, że prędkość komunikacji przez określone łącze przyjmuje tylko dwie wartości. Można więc wykorzystywać je także w przypadku, kiedy zmiany prędkości przesyłania danych są bardziej złożone.

4.6 Minimalizacja maksymalnego opóźnienia zestawu danych [H3, H4, H8]

Prace [H3, H4, H8] poświęcone są kolejnemu kryterium optymalizacji, którym jest minimalizacja maksymalnego opóźnienia zestawu danych. Dla każdego zestawu danych D_j określony jest jego czas gotowości r_j, a także pożądany czas przetworzenia d_j, czyli termin, do którego zestaw powinien zostać odebrany przez stację bazową i przetworzony. Dla ustalonego uszeregowania różnicę między faktycznym czasem zakończenia przetwarzania zestawu D_j a jego pożądanym czasem przetworzenia d_j nazywamy opóźnieniem tego zestawu i oznaczamy symbolem L_j. Należy skonstruować uszeregowanie minimalizujące wartość $L_{max} = \max_{j=1}^{m}\{L_j\}$.

W pracy [H3] rozważam sieć homogeniczną, w której możliwe jest przerywanie przesyłania zestawu danych, a odebrane dane nie są przetwarzane przez stację bazową. Mamy więc $s_j = s$, $c_j = c$ oraz $a = 0$ (w [H3] używałam oznaczeń S i C w miejsce s i c). Problem minimalizacji maksymalnego opóźnienia w takiej sieci jest równoważny ze specjalnym, silnie NP-trudnym przypadkiem znanego z literatury problemu szeregowania na jednym procesorze przerywalnych zadań z czasami przygotowania (ang. *setup time*) w celu minimalizacji maksymalnego czasu dostarczenia zadania (ang. *delivery time*), $1|pmtn, r_j, s_j|D_{max}$ [34].

Wykazałam, że podobnie jak w przypadku problemu rozważanego w [H5], skonstruowanie optymalnego uszeregowania może wymagać przerwania przesyłania pewnego zestawu danych w momencie, kiedy żaden inny zestaw nie uzyskuje gotowości ([H3], Proposition 1). Wynik ten jest szczególnie istotny w kontekście wcześniejszych badań dotyczących problemu

$1|pmtn, r_j, s_j|D_{max}$. W pracy [34] zaproponowano dla niego dokładny algorytm programowania dynamicznego oraz wielomianowy schemat aproksymacji (PTAS). Oba te algorytmy opierają się jednak na podanej w [34] bez dowodu obserwacji, że zawsze istnieje rozwiązanie optymalne, w którym zadania przerywane są tylko w momentach, kiedy inne zadania stają się gotowe. Uzyskany przeze mnie wynik dowodzi, że stwierdzenie to jest fałszywe i w konsekwencji algorytmy przedstawione w [34] są niepoprawne. Zagadnienie konstrukcji algorytmów rozwiązujących problem $1|pmtn, r_j, s_j|D_{max}$ pozostaje więc otwarte.

Dla rozważanego problemu szeregowania w sieciach zbierających dane zaproponowałam rodzinę działających w czasie $O(m \log m)$ algorytmów DS(γ), gdzie $\gamma \in [0, \infty]$ jest parametrem. Wykazałam, że dla każdej instancji istnieje rozwiązanie optymalne, w którym dla każdego przerwania przesyłania pewnego zestawu danych D_i przez przesyłanie innego zestawu D_j zachodzi nierówność $d_j < d_i$ ([H3], Proposition 2). Obserwacja ta została wykorzystana podczas konstrukcji algorytmu DS(γ). Jeśli pewien zestaw D_j uzyskuje gotowość w trakcie przesyłania innego zestawu D_i, to przesyłanie D_i zostaje przerwane, o ile $d_j < d_i - \gamma s$. Przerwanie jest zatem możliwe wyłącznie dla $d_j < d_i$, ale nie jest wykonywane, jeśli różnica między pożądanymi terminami przetworzenia zestawów D_i i D_j jest zbyt mała w porównaniu do kosztu s podzielenia przesyłania zestawu D_i na większą liczbę komunikatów. Parametr γ określa, jak duża musi być ta różnica, aby przerwanie miało miejsce.

Jakość rozwiązań generowanych przez algorytm DS(γ) dla $\gamma \in \{0, 1, 10, 100, \infty\}$ została oszacowana na podstawie eksperymentów obliczeniowych. Wygenerowałam instancje testowe, w których łączny rozmiar danych do zebrania wynosił $V = 1\text{E}9$, a sieć zawierała $m \in \{10, 20, \ldots, 100\}$ węzłów. W celu ustalenia odpowiednio zróżnicowanych rozmiarów poszczególnych zestawów danych, losowałam liczby β_j z przedziału $[1, 2]$, a następnie obliczałam $\alpha_j = \beta_j V / \sum_{j=1}^{m} \beta_j$. Odwrotność prędkości komunikacji wynosiła $c = 1\text{E-}8$, a czas inicjalizacji komunikacji s należał do zbioru $\{1\text{E}{-}3, 2\text{E}{-}3, 5\text{E}{-}3, 1\text{E}{-}2\}$. Dla danych wartości $\delta_d, \delta_r \in \{0{,}25; 0{,}5; 0{,}99\}$ oraz $T = ms + cV$ czasy gotowości r_j były wybierane losowo z przedziału $[0, \delta_r T]$, a pożądane terminy przetworzenia d_j z przedziału $[0, \delta_d T]$. Następnie wylosowane wartości były sortowane w taki sposób, że $r_1 \leqslant \cdots \leqslant r_m$ i $d_1 \geqslant \cdots \geqslant d_m$. Maksymalne opóźnienie występujące w konstruowanych uszeregowaniach było porównywane z ograniczeniem dolnym *LoBo*, obliczonym przez rozwiązanie instancji problemu $1|r_j, pmtn|L_{max}$ z czasami wykonywania zadań $p_j = s + c\alpha_j$ oraz czasami gotowości r_j i pożądanymi terminami zakończenia zadań d_j. W ogólnym przypadku używanie względnych miar jakości rozwiązań nie jest możliwe dla kryterium maksymalnego opóźnienia, ponieważ jego wartość optymalna może być nie tylko dodatnia, ale też zerowa lub ujemna. Jednak z opisanego powyżej sposobu generowania instancji wynika, że wartość *LoBo* jest dodatnia, ponieważ $\max_{j=1}^{m}\{d_j\} < \sum_{j=1}^{m} p_j$. Jako miarę jakości uszeregowania przyjęłam więc iloraz występującego w nim maksymalnego opóźnienia i wartości *LoBo*. Wyniki eksperymentów pokazały, że jeśli wartość γ jest odpowiednio dobrana, to średni błąd względny otrzymywanych rozwiązań nie przekracza 3% w stosunku do ograniczenia dolnego. Dla wszystkich rozważanych zestawów parametrów instancji najlepsze rozwiązania były otrzymywane przez jeden z algorytmów DS(10) lub DS(100).

Badania dotyczące minimalizacji maksymalnego opóźnienia kontynuowałam w pracy [H4]. Rozważałam w niej sieć, w której zestawy danych po dostarczeniu do stacji bazowej muszą zostać przetworzone, a zatem $a \neq 0$ (w [H4] używałam oznaczeń A i C zamiast a i c). Korzystając z silnej NP-zupełności problemu trójpodziału, wykazałam, że problem minimalizacji maksymalnego opóźnienia jest w tym przypadku silnie NP-trudny, nawet jeśli czas inicjalizacji komunikacji s jest zerowy ([H4], Proposition 1). W związku z tym, prowadzone rozważania dotyczyły właśnie przypadku $s = 0$, w którym analizowane zagadnienie jest specjalnym przypadkiem problemu minimalizacji maksymalnego opóźnienia w dwumaszynowym systemie przepływowym z czasami gotowości zadań i możliwymi przerwaniami, $F2|pmtn, r_j|L_{max}$. Dla ustalonego uszeregowania komunikacji między węzłami a stacją bazową, wybór optymalnego sposobu przetwarzania danych sprowadza się do rozwiązania problemu szeregowania na jednym procesorze przerywalnych zadań z czasami gotowości w celu minimalizacji maksymalnego opóźnienia, $1|r_j, pmtn|L_{max}$. Problem ten można rozwiązać w czasie $O(m \log m)$ [21]. Zaproponowałam więc algorytmy, które wykorzystują różne metody konstrukcji uszeregowania komunikacji, natomiast przetwarzanie danych realizują w sposób optymalny. Pokazałam, że dla $s = 0$ zawsze istnieje optymalne uszeregowanie, w którym przerwania komunikacji mają miejsce tylko w momentach uzyskiwania gotowości przez zestawy danych. Opierając się na tej obserwacji, przedstawiłam dokładny algorytmu podziału i ograniczeń (BB). Ponadto zaproponowałam trzy heurystyki działające w czasie $O(m \log m)$, oparte na znanych regułach priorytetowych: FIFO (*First In, First Out*), EDD (*Earliest Due Date*) i SRT (*Shortest Remaining Time*). W każdej z nich, jeśli zestaw danych D_j uzyskuje gotowość w trakcie przesyłania zestawu D_i, to przerwanie ma miejsce, o ile zestaw D_j ma wyższy priorytet niż D_i według wykorzystywanej reguły.

Eksperymenty obliczeniowe zostały przeprowadzone dla instancji z $m = 10$ węzłami, dla których możliwe było obliczenie optymalnych rozwiązań za pomocą algorytmu BB. Wartości parametrów sieci c i a należały do zbioru $\{1, \ldots, 5\}$, rozmiary danych α_j były losowane z przedziału $[1, 15]$, a pożądane terminy przetworzenia d_j z przedziału [0,100]. Czasy gotowości były ustalane według wzorów $r_1 = 0$, $r_j = r_{j-1} + \delta_j$ dla $j > 1$, gdzie wartości δ_j były losowane z przedziału $[1, 10]$. Ponieważ optymalne wartości maksymalnego opóźnienia mogły być niedodatnie, miarą jakości uszeregowania była różnica między maksymalnym opóźnieniem uzyskanym przez dany algorytm a wartością optymalną obliczoną przez BB. Wyniki eksperymentów pokazały, że wybór najlepszej reguły priorytetowej zależy przede wszystkim od tego, czy komunikacja jest wolniejsza, czy szybsza od przetwarzania danych. W większości przypadków najlepiej działa reguła EDD, ale jeśli przetwarzanie jest bardzo wolne, czyli a jest znacznie większe niż c, to najlepsze wyniki otrzymywane są dzięki zastosowaniu reguły FIFO, natomiast EDD daje najgorsze rezultaty.

Powyższe obserwacje dotyczące skuteczności reguł priorytetowych FIFO, EDD i SRT stały się inspiracją do skonstruowania nowej reguły SD (*Stage Difference*), którą przedstawiłam w artykule [H8]. Reguła ta bierze pod uwagę zarówno pożądane terminy przetworzenia d_j, jak i różnicę między wartościami c i a. Przesyłanie zestawu D_j zostaje przerwane w momencie

uzyskania gotowości przez zestaw D_k, jeśli spełnione są jednocześnie dwa warunki: $d_k < d_j$ oraz różnica między czasem potrzebnym do przetworzenia D_j a czasem pozostałym do zakończenia jego przesyłania jest wystarczająco mała w stosunku do wartości $d_j - d_k$. Dokładniej, jeśli w chwili r_k rozmiar pozostałej do przesłania części zestawu D_j wynosi β_j, to przerwanie ma miejsce, o ile $d_k < d_j$ i $m(a\alpha_j - c\beta_j) < d_j - d_k$ (w [H8] używane były oznaczenia n i c_1 w miejsce m i c).

Skuteczność nowej reguły została przetestowana za pomocą eksperymentów obliczeniowych. W wygenerowanych instancjach testowych, dla ustalonych wartości α_{max} i d_{max}, rozmiary zestawów danych były losowane z zakresu $[0, \alpha_{max}]$, a pożądane terminy przetworzenia z zakresu $[0, d_{max}]$. Terminy gotowości zestawów były określone wzorami $r_1 = 0$, $r_j = r_{j-1} + \rho_j$ dla $j > 1$, gdzie ρ_j było losowane z zakresu $[1, \rho_{max}]$. W podstawowej konfiguracji przyjęłam $m = 100$, $c = a = 1$, $\alpha_{max} = 15$, $d_{max} = 100$, $\rho_{max} = 10$. Poszczególne eksperymenty analizowały wpływ zmian wartości tych parametrów na otrzymywane maksymalne opóźnienia. Miarą jakości rozwiązań była różnica między maksymalnym opóźnieniem uzyskanym przez dany algorytm a dolnym ograniczeniem LB, obliczonym przez odrzucenie pewnych ograniczeń w rozważanym problemie i rozwiązanie otrzymanych w ten sposób instancji problemu $1|pmtn, r_j|L_{max}$.

Wyniki eksperymentów pokazały, że jeśli c jest znacznie mniejsze niż a, to algorytm SD generuje rozwiązania takie jak EDD, jeśli natomiast c jest znacznie większe niż a, to uszeregowania otrzymane przez SD są takie same jak w przypadku reguły FIFO. Co więcej, jeśli różnica między wartościami c i a nie jest bardzo duża, to uszeregowania generowane przez SD są zdecydowanie lepsze od wyników uzyskanych przy użyciu każdej z pozostałych trzech reguł. Wyniki uzyskiwane przez heurystykę SD były zatem co najmniej tak samo dobre jak uszeregowania dostarczone przez najlepszy z rozważanych wcześniej algorytmów FIFO, EDD i SRT dla wszystkich analizowanych kombinacji parametrów instancji. Próba skonstruowania algorytmu łączącego zalety reguł EDD i FIFO zakończyła się więc pełnym sukcesem.

Druga część pracy [H8] poświęcona jest uogólnieniu rozważanego problemu na przypadek sieci o topologii drzewa wysokości dwa. Sieć taka składa się z n węzłów pozyskujących dane, m węzłów pośrednich i jednej stacji bazowej. Węzeł pośredni P_i (gdzie $i = 1, \ldots, m$) otrzymuje zestawy danych D_{ij} z węzłów P_{ij} (gdzie $j = 1, \ldots, n_i$ oraz $n_1 + \cdots + n_m = n$), przetwarza je, a następnie przekazuje wyniki do stacji bazowej. Zestaw danych D_{ij} opisany jest przez rozmiar α_{ij}, czas gotowości r_{ij} oraz pożądany czas przetworzenia i dostarczenia do stacji bazowej d_{ij}. Parametrami określającymi wydajność sieci są: odwrotność prędkości przesyłania danych do węzłów pośrednich c_1, odwrotność prędkości przetwarzania danych przez węzły pośrednie a oraz odwrotność prędkości komunikacji między węzłami pośrednimi a stacją bazową c_2. Ponieważ czas przesyłania wyników przetwarzania zestawu danych D_{ij} do stacji bazowej wynosi $c_2\alpha_{ij}$, wartość c_2 reprezentuje zarazem wydajność komunikacji ze stacją bazową i ewentualną zmianę rozmiaru danych w wyniku ich przetworzenia. Każdy węzeł pośredni może jednocześnie odbierać pewien zestaw danych, przetwarzać inny zestaw i przesyłać jeszcze inny zestaw do stacji bazowej. Opisana sieć jest więc trzyetapowym hybrydowym systemem przepływowym (ang. *hybrid flow shop*) z m dedykowanymi maszynami w pierwszym i drugim etapie oraz jedną

maszyną w trzecim etapie.

Dla ustalonego sposobu przekazywania danych do węzłów pośrednich oraz przetwarzania ich, optymalne uszeregowanie komunikacji ze stacją bazową polega na rozwiązaniu problemu $1|r_j, pmtn|L_{max}$. Zaproponowałam więc działające w czasie $O(n \log n)$ heurystyki, które w pierwszym kroku szeregują komunikację węzłów z danymi z poszczególnymi węzłami pośrednimi, a w drugim kroku tworzą uszeregowanie dla etapu przetwarzania danych. W każdym z tych dwóch etapów wykorzystywana jest jedna z czterech reguł priorytetowych FIFO, EDD, SRT i SD, co daje łącznie 16 różnych algorytmów. Uszeregowania są tworzone osobno dla każdej podsieci składającej się z węzła pośredniego P_i i komunikujących się z nim węzłów P_{ij}. Na końcu konstruowane jest optymalne uszeregowanie komunikacji między węzłami pośrednimi a stacją bazową.

Wydajność zaproponowanych algorytmów została przeanalizowana za pomocą eksperymentów obliczeniowych. W podstawowej konfiguracji instancji testowych sieć zawierała 5 węzłów pośrednich, z których każdy zbierał dane od 20 węzłów. Sieć była opisana parametrami $c_1 = a = 1$, $c_2 = 0{,}2$. Parametry zestawów danych były generowane osobno dla każdej podsieci, w taki sam sposób, jak dla rozważanej wcześniej sieci o topologii gwiazdy. W szeroko zakrojonych eksperymentach przeanalizowałam wpływ zmian wartości c_2, m, c_1, a, α_{max}, d_{max}, ρ_{max} oraz zróżnicowania rozmiarów poszczególnych podsieci na jakość uszeregowań dostarczanych przez zaproponowane algorytmy. Uzyskane wyniki pokazały, że, w zależności od wartości parametrów instancji, najlepsze rozwiązania mogą być generowane przez różne heurystyki, jednak zastosowanie w obu etapach reguły SD prowadzi do uzyskania bardzo dobrych uszeregowań dla wszystkich rozważanych rodzajów testów.

Prace [H3, H4, H8] stanowią spójny cykl badań dotyczących minimalizacji maksymalnego opóźnienia w sieciach zbierających dane. Rozważany model był stopniowo rozbudowywany, najpierw przez dodanie etapu przetwarzania danych w [H4], a następnie przez wprowadzenie węzłów pośrednich w [H8]. Kolejne analizowane problemy były więc powiązane z zagadnieniami szeregowania na jednym procesorze [H3], w dwumaszynowym systemie przepływowym [H4, H8] i trzyetapowym hybrydowym systemie przepływowym [H8]. Wykazanie w [H3] niepoprawności algorytmów zaproponowanych w [34] ponownie otwiera badania dotyczące problemu $1|pmtn, r_j, s_j|D_{max}$. Udowodnienie silnej NP-trudności problemu przedstawionego w [H4, H8] stanowi wkład w analizę złożoności problemów szeregowania zadań w systemach przepływowych. Duże znaczenie praktyczne ma skonstruowanie dla wszystkich rozważanych tu problemów bardzo szybkich heurystyk, które, jak pokazują eksperymenty obliczeniowe, generują bardzo dobre uszeregowania. Szczególnie ciekawe wydaje się połączenie zalet szeroko stosowanych reguł priorytetowych EDD i FIFO w nowej regule SD.

4.7 Uwagi końcowe

Szeregowanie zadań w sieciach zbierających dane jest fascynującym obszarem, którego praktyczne znaczenie stale rośnie wraz ze zwiększającą się ilością danych gromadzonych i przetwarzanych w różnych systemach komputerowych. Na prezentowane osiągnięcie badawcze składa

się osiem prac poświęconych optymalizacji procesu zbierania danych. W problemach szeregowania zadań jakość rozwiązań może być oceniana za pomocą różnych kryteriów. Moje badania dotyczyły maksymalizacji czasu życia sieci, minimalizacji kosztu kompresji danych, minimalizacji czasu zbierania danych oraz minimalizacji maksymalnego opóźnienia. Istotą przedstawianego tu osiągnięcia naukowego jest określenie złożoności obliczeniowej rozważanych problemów szeregowania zadań w sieciach zbierających dane, opracowanie rozwiązujących je algorytmów dokładnych i przybliżonych oraz oszacowanie ich efektywności za pomocą metod teoretycznych i eksperymentalnych.

Większość rozważanych problemów okazała się trudna obliczeniowo. Skonstruowanie rozwiązujących je algorytmów wymagało użycia różnorodnych technik. Zaproponowane algorytmy dokładne wykorzystują analizę przypadków [H1], metodę enumeracyjną [H2], metodę podziału i ograniczeń [H4], programowanie dynamiczne [H7] i programowanie liniowe [H7]. Algorytmy heurystyczne opierają się na technice zachłannej [H2, H6, H7], szeregowaniu listowym [H6], regułach priorytetowych [H4, H8], wykorzystaniu parametru do równoważenia wpływu różnych czynników na jakość uszeregowania [H3, H5] oraz przeszukiwaniu lokalnym [H6, H7]. Dla wielu z zaproponowanych heurystyk wykazałam istnienie stałych współczynników aproksymacji. Wszystkie zaprojektowane algorytmy zostały zaimplementowane i przetestowane za pomocą licznych eksperymentów obliczeniowych. Dla większości analizowanych problemów zaproponowałam co najmniej kilka różnych algorytmów. Przeprowadzone eksperymenty pozwoliły określić, który z nich osiąga najlepsze wyniki w zależności od wartości parametrów sieci i zestawów danych. Pokazały również, że najlepsze spośród przedstawionych algorytmów tworzą uszeregowania wysokiej jakości w krótkim czasie.

Uzyskane wyniki mają znaczenie zarówno teoretyczne, jak i praktyczne. Określenie złożoności rozważanych problemów, jak również konstrukcja rozwiązujących je algorytmów wraz z analizą (w przypadku metod aproksymacyjnych) gwarancji jakości otrzymywanych rozwiązań, stanowią wkład w rozwój teorii szeregowania zadań. Ponieważ problemy szeregowania w sieciach zbierających dane są powiązane z zagadnieniami szeregowania na jednym procesorze i w systemach przepływowych, dotyczące ich wyniki mają znaczenie nie tylko dla analizy sieci zbierających dane, ale także dla klasycznych modeli teorii szeregowania zadań. Wskazują nowe zastosowania tych modeli oraz szczególnie interesujące specjalne przypadki problemów analizowanych we wcześniejszej literaturze. Teoretyczne wyniki pociągają za sobą praktyczne konsekwencje. Udowodnienie trudności obliczeniowej problemu sygnalizuje, że konstrukcja dokładnych algorytmów rozwiązujących go w akceptowalnym czasie najprawdopodobniej nie jest możliwa, a więc w praktyce należy skoncentrować się na algorytmach przybliżonych. Wyniki dotyczące współczynników aproksymacji takich algorytmów pozwalają na oszacowanie błędów przybliżenia, które mogą wystąpić przy ich zastosowaniu w rzeczywistości. Analiza eksperymentalna algorytmów dostarcza dokładniejszych informacji dotyczących ich przewidywanego zachowania w sieciach o znanych parametrach. Pozwala również wybrać algorytm, który będzie najodpowiedniejszy dla konkretnej, rzeczywistej sieci. Stosowanie w praktyce efektywnych algorytmów szeregowania zadań umożliwia zbieranie danych w krótszym czasie, przy niższych

kosztach i mniejszym zużyciu energii.

Większość badań składających się na prezentowane osiągnięcie była finansowana przez Narodowe Centrum Nauki w ramach kierowanego przeze mnie projektu 2016/23/D/ST6/00410 „Szeregowanie zadań w sieciach zbierających dane". Poza wynikami naukowymi, efektem tego projektu było nawiązanie współpracy z Yakovem Zinderem i Alexandrem Kononovem [P10, P16], Baruchem Morem (kontynuacja badań z [H7]) oraz Bartłomiejem Przybylskim [P9, P17].

5 Informacja o wykazywaniu się istotną aktywnością naukową albo artystyczną realizowaną w więcej niż jednej uczelni, instytucji naukowej lub instytucji kultury, w szczególności zagranicznej

1. W latach 2007 – 2011 prowadziłam badania naukowe na Politechnice Poznańskiej pod kierunkiem pracującego tam promotora mojego doktoratu, prof. dr. hab. inż. Macieja Drozdowskiego. Badania te były częścią grantu MNiSW nr N N519 188933 kierowanego przez prof. dr. hab. inż. Jacka Błażewicza.

2. W latach 2011 – 2014 prowadziłam badania naukowe w ramach zlokalizowanego na Politechnice Poznańskiej grantu NCN nr N N519 643340, również kierowanego przez prof. dr. hab. inż. Jacka Błażewicza.

3. Od 2018 r. współpracuję z Yakovem Zinderem z University of Technology Sydney i Alexandrem Kononovem z Nowosybirskiego Uniwersytetu Państwowego. Dotychczas opublikowaliśmy dwa wspólne artykuły [P10, P16]. Od Yakova Zindera otrzymałam zaproszenie do pobytu w University of Technology Sydney jako *visiting professor*. Mój przyjazd był zaplanowany na sierpień 2020 r., jednak musiał zostać odłożony w czasie z powodu pandemii COVID-19.

4. Od 2020 r. współpracuję z Baruchem Morem z Uniwersytetu Ariel w Izraelu. Obecnie przygotowujemy do publikacji wyniki będące kontynuacją pracy [H7].

6 Informacja o osiągnięciach dydaktycznych, organizacyjnych oraz popularyzujących naukę lub sztukę.

6.1 Działalność dydaktyczna

1. Zajęcia dydaktyczne prowadzone na Wydziale Matematyki i Informatyki UAM:

   wykłady: algorytmy i struktury danych,

   ćwiczenia/laboratoria: algorytmy i struktury danych, algorytmy i programowanie, algorytmy kombinatoryczne, algorytmika w projektowaniu systemów, podstawy programowania, programowanie obiektowe, projekt magisterski, zaawansowane algorytmy kombinatoryczne,

   ćwiczenia/laboratoria w języku angielskim: algorithms and data structures,

seminarium magisterskie.

Prowadzone przeze mnie zajęcia są bardzo wysoko oceniane w ankietach studenckich.

2. Opieka naukowa nad studentami: jestem promotorem siedmiu prac magisterskich na kierunku informatyka (w trakcie realizacji).

3. Wykonawca w grancie dydaktycznym Wydziału Matematyki i Informatyki UAM nr GD-03/2011: *Sprawdzarka – automatyczny serwis oceny rozwiązań zadań programistycznych* (2011).

4. Opracowanie materiałów na platformę OLAT do ćwiczeń z przedmiotu podstawy programowania (2011).

5. Opracowanie zestawu zadań programistycznych na sprawdzarkę acm.edu.pl dla przedmiotów algorytmy i struktury danych oraz repetytorium z algorytmiki i programowania, w ramach projektu pt. Zintegrowany program rozwoju Uniwersytetu im. Adama Mickiewicza w Poznaniu „Zaawansowane technologie dla rozwoju wysoko wykwalifikowanych kadr dla gospodarki", współfinansowanego ze środków Europejskiego Funduszu Społecznego w ramach Programu Operacyjnego Kapitał Ludzki UDA-POKL.04.03.00-00-152/12-00 (2014).

6. Opracowanie sylabusa przedmiotu zaawansowane algorytmy kombinatoryczne w ramach projektu Akademia Innowacyjnych Zastosowań Technologii Cyfrowych (2020).

6.2 Działalność organizacyjna

1. Członek komitetu programowego i *local chair* sekcji Scheduling and load balancing konferencji 26th International European Conference on Parallel and Distributed Computing (Euro-Par) 2020.

2. Członek komitetu organizacyjnego konferencji The Third International Workshop on Dynamic Scheduling Problems (konferencja została przeniesiona z czerwca 2020 r. na lipiec 2021 r. z powodu pandemii COVID-19).

3. Członek rady programowej grupy kierunków informatycznych przy Wydziale Matematyki i Informatyki UAM (2019 – 2020).

6.3 Działalność popularyzująca naukę

1. Członek Komitetu Okręgowego Olimpiady Matematycznej w Poznaniu (2009 – obecnie).

2. Prowadząca zajęcia Międzyszkolnego Koła Matematycznego przy Wydziale Matematyki i Informatyki UAM (2002 – 2004).

3. Weryfikator i koordynator ocen zadań podczas zawodów 3rd Middle European Mathematical Olympiad (2009).

4. Sędzia XIV i XV Akademickich Mistrzostw Polski w Programowaniu Zespołowym (2009, 2010).

5. Sędzia VI Mistrzostw Wielkopolski w Programowaniu Zespołowym (2011).

7 Pozostałe osiągnięcia

7.1 Pozostałe publikacje

W tym punkcie przedstawione są publikacje niewchodzące w skład prezentowanego osiągnięcia badawczego. Dla zachowania spójności dokumentacji, numeracja prac jest taka sama jak w wykazie osiągnięć naukowych (załącznik 4), w którym znajdują się również liczby cytowań i inne dane naukometryczne.

Publikacje w czasopismach przed uzyskaniem stopnia doktora:

[P11] J. Berlińska, M. Drozdowski, M. Lawenda, *Experimental study of scheduling with memory constraints using hybrid methods*, Journal of Computational and Applied Mathematics 232 (2009) 638-654.

[P12] J. Berlińska, M. Drozdowski, *Heuristics for multi-round divisible loads scheduling with limited memory*, Parallel Computing 36 (2010) 199-211.

[P13] J. Berlińska, M. Drozdowski, *Scheduling divisible MapReduce computations*, Journal of Parallel and Distributed Computing 71 (2011) 450-459.

Publikacje w czasopismach po uzyskaniu stopnia doktora:

[P14] J. Berlińska, M. Drozdowski, *Scheduling multilayer divisible computations*, RAIRO – Operations Research 49 (2015) 339-368.

[P15] J. Berlińska, M. Drozdowski, *Comparing load-balancing algorithms for MapReduce under Zipfian data skews*, Parallel Computing 72 (2018) 14-28.

[P16] J. Berlińska, A. Kononov, Y. Zinder, *Two-machine flow shop with dynamic storage space*, Optimization Letters (2020) DOI: 10.1007/s11590-020-01645-5, 22 strony.

[P17] J. Berlińska, B. Przybylski, *Scheduling for gathering multitype data with local computations*, European Journal of Operational Research (2021) DOI: 10.1016/j.ejor.2021.01.043, 7 stron.

Publikacje w recenzowanych materiałach konferencyjnych przed uzyskaniem stopnia doktora:

[P1] J. Berlińska, M. Drozdowski, *Heuristics for divisible loads scheduling in systems with limited memory*, Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory & Applications, Dublin 2009, 321-329.

[P2] J. Berlińska, *Fully polynomial time approximation schemes for scheduling divisible loads*, w: R. Wyrzykowski et al. (eds), Parallel Processing and Applied Mathematics: 8th International Conference PPAM 2009, Part II, Lecture Notes in Computer Science 6068, Springer, Berlin 2010, 1-10.

[P3] J. Berlińska, M. Drozdowski, *Scheduling and performance of divisible MapReduce applications*, 10th Workshop on Models and Algorithms for Planning and Scheduling Problems, Nymburk 2011, 162-164.

Publikacje w recenzowanych materiałach konferencyjnych po uzyskaniu stopnia doktora:

[P4] J. Berlińska, M. Drozdowski, *Mitigating partitioning skew in MapReduce computations*, Proceedings of the 6th Multidisciplinary International Scheduling Conference: Theory & Applications, Ghent 2013, 80-90.

[P5] J. Berlińska, *Scheduling data gathering with variable communication speed*, Proceedings of the First International Workshop on Dynamic Scheduling Problems, Poznań 2016, 29-32.

[P6] J. Berlińska, *Scheduling data gathering in 2-level tree networks*, Proceedings of the 13th Workshop on Models and Algorithms for Planning and Scheduling Problems, Seeon Abbey 2017, 17-19.

[P7] J. Berlińska, *Scheduling data gathering with limited base station memory*, Proceedings of the 16th International Conference on Project Management and Scheduling, Rome 2018, 42-45.

[P8] J. Berlińska, *Scheduling non-preemptive data gathering affected by background communications*, Proceedings of the Second International Workshop on Dynamic Scheduling Problems, Poznań 2018, 41-44.

[P9] J. Berlińska, B. Przybylski, *Scheduling for gathering multitype data*, Proceedings of the 14th Workshop on Models and Algorithms for Planning and Scheduling Problems, Renesse 2019, 170-172.

[P10] J. Berlińska, A. Kononov, Y. Zinder, *Two-machine flow shop with a dynamic storage space and UET operations*, w: H. Le Thi, H. Le, T. Pham Dinh (eds), Optimization of Complex Systems: Theory, Models, Algorithms and Applications. WCGO 2019. Advances in Intelligent Systems and Computing 991, Springer, Cham 2020, 1139-1148.

7.1.1 Sieci zbierające dane [P5, P6, P7, P8, P9, P17]

Napisane wspólnie z Bartłomiejem Przybylskim prace [P9, P17] dotyczą problemu zbierania danych, w którym zestawy danych są charakteryzowane przez przypisane im typy. Centralny serwer musi pozyskać określoną liczbę zestawów każdego typu. Zestawy danych mogą być

pobierane z posiadających je węzłów sieci lub generowane lokalnie na serwerze. Rozważaliśmy problem minimalizacji czasu gromadzenia wymaganych danych. W pracy [P9] pokazaliśmy, że problem ten jest NP-trudny i sformułowaliśmy go jako zagadnienie całkowitoliczbowego programowania liniowego (ILP). Wskazaliśmy też dwa specjalne przypadki rozwiązywalne w czasie wielomianowym. W artykule [P17] zaproponowaliśmy ponadto dokładny algorytm programowania dynamicznego (DP) oraz wielomianowy algorytm 2-aproksymacyjny oparty na programowaniu liniowym (ApLP). Przedstawiliśmy także w pełni wielomianowy schemat aproksymacji (FPTAS). Wydajność algorytmów ILP, DP oraz ApLP została przetestowana eksperymentalnie. Wskazaliśmy, dla jakich instancji szybciej działa algorytm ILP, a dla jakich DP. Pokazaliśmy, że algorytm ApLP generuje w praktyce bardzo dobre uszeregowania, o średnim błędzie rzędu poniżej jednego procenta.

Prace [P5] i [P8] zawierają wstępne wyniki dotyczące zagadnienia minimalizacji czasu zbierania danych w sieciach ze zmienną prędkością komunikacji, któremu poświęcony został późniejszy artykuł [H7]. Ponadto, w [P5] zostały przedstawione algorytmy online dla podzielnego wariantu tego problemu.

Praca [P6] dotyczy minimalizacji czasu zbierania danych w sieci o topologii drzewa wysokości dwa. Przedstawiłam w niej wielomianowe algorytmy dokładne dla dwóch specjalnych przypadków tego problemu oraz zaproponowałam algorytm heurystyczny dla przypadku ogólnego. Praca [P7] zawiera wstępne wyniki badań nad problemem rozważanym w [H6].

7.1.2 Zadania jednorodnie podzielne [P1, P2, P11, P12]

Artykuł [P11] powstały we współpracy z Maciejem Drozdowskim i Marcinem Lawendą oraz prace [P1, P12] napisane z Maciejem Drozdowskim poświęcone są szeregowaniu zadań jednorodnie podzielnych w heterogenicznych systemach rozproszonych. Wolumen danych zgromadzony na centralnym serwerze należy podzielić na części, które zostaną przesłane do różnych węzłów i przetworzone przez nie. Każdy z węzłów może otrzymać wiele porcji danych. W momencie rozpoczęcia przesyłania porcji danych odbierający je węzeł alokuje blok pamięci odpowiedniego rozmiaru, który zostanie zwolniony w momencie zakończenia ich przetwarzania. Każdy węzeł dysponuje ograniczoną pamięcią, a więc łączny rozmiar zaalokowanych na nim bloków nie może w żadnym momencie przekroczyć jej rozmiaru. Należy ustalić podział danych na części i kolejność komunikacji serwera z pozostałymi węzłami, tak aby wszystkie dane zostały przetworzone w najkrótszym czasie. NP-trudność tego problemu została wykazana w pracach [16, 47].

W artykule [P11] przedstawiliśmy dokładny algorytm podziału i ograniczeń (BB) oraz algorytm genetyczny (GA). Przeprowadziliśmy liczne eksperymenty obliczeniowe, w których przeanalizowaliśmy wpływ różnych parametrów systemu na jakość i strukturę otrzymywanych uszeregowań. Przedstawiliśmy również wyniki teoretyczne dotyczące jakości rozwiązań o specjalnej strukturze. Sformułowaliśmy obserwacje wskazujące, dla jakich instancji problemu jest łatwo (lub trudno) znaleźć dobre uszeregowanie oraz jakie cechy powinny mieć dobre algorytmy heurystyczne.

Na podstawie wyników otrzymanych w [P11] skonstruowaliśmy kilka grup heurystyk przedstawionych w pracach [P1] i [P12]. Otrzymywane przez nie uszeregowania zostały porównane za pomocą eksperymentów obliczeniowych z wynikami algorytmu genetycznego opisanego w [P11]. Dwa spośród zaproponowanych algorytmów heurystycznych otrzymywały wyniki zbliżone do tych zwracanych przez algorytm genetyczny, a ponadto działały bardzo szybko, w czasie o kilka rzędów wielkości krótszym niż GA.

Praca [P2] dotyczy innego zagadnienia szeregowania zadań jednorodnie podzielnych. W rozważanym w niej systemie pamięć poszczególnych węzłów nie jest ograniczona, a każdy z nich może otrzymać do przetworzenia co najwyżej jedną porcję danych. Ponadto, każde łącze w sieci charakteryzuje się stałym czasem inicjalizacji komunikacji, ale jego przepustowość jest nieograniczona. Innymi słowy, czas potrzebny na przekazanie danych przez ustalone łącze nie zależy od ich rozmiaru. Analizowałam dwa dualne problemy: minimalizacji czasu przetwarzania danych określonego rozmiaru oraz maksymalizacji rozmiaru danych przetworzonych w ustalonym czasie. W raporcie technicznym [47] wykazano NP-trudność tych problemów i zaproponowano dla nich pseudowielomianowe algorytmy programowania dynamicznego. W artykule [P2] przedstawiłam w pełni wielomianowe schematy aproksymacji (FPTAS) dla obu rozważanych zagadnień, domykając analizę ich złożoności i aproksymowalności.

7.1.3 Obliczenia MapReduce [P3, P4, P13, P14, P15]

W tym punkcie przedstawiam uzyskane wspólnie z Maciejem Drozdowskim wyniki dotyczące modelowania i szeregowania obliczeń MapReduce. MapReduce jest paradygmatem przetwarzania równoległego dużych zbiorów danych w klastrach komputerów, wprowadzonym i początkowo rozwijanym przez firmę Google [13]. Obliczenia podzielone są na dwie fazy: Map i Reduce. W fazie Map dane wejściowe są dzielone na części przetwarzane równolegle na wielu komputerach. Wynikiem obliczeń jest zbiór pośrednich par (klucz, wartość). Następnie wygenerowane pary są przesyłane do różnych komputerów, tak że pary o tym samym kluczu trafiają do tego samego komputera. W fazie Reduce pośrednie pary o tym samym kluczu są scalane, a jako wynik otrzymywana jest nowa lista finalnych par (klucz, wartość). W pracach [P3, P13] zaproponowaliśmy pierwszy w literaturze teoretyczny model takich obliczeń oparty na teorii zadań jednorodnie podzielnych. Skonstruowaliśmy także dwa algorytmy podziału danych i szeregowania komunikacji między procesorami. Przeprowadziliśmy eksperymenty obliczeniowe analizujące granice wydajności obliczeń MapReduce dla różnych konfiguracji parametrów systemu komputerowego i wykonywanej aplikacji.

W artykule [P14] rozważaliśmy jednorodnie podzielne obliczenia wielowarstwowe. Obliczenia takie składają się z wielu etapów. Wynik działania każdego etapu (z wyjątkiem ostatniego) stanowi dane wejściowe dla kolejnego etapu. Każdy etap wykonywany jest równolegle na wielu komputerach. Obliczenia MapReduce są przykładem jednorodnie podzielnej aplikacji dwuwarstwowej. W pracy [P14] zaproponowaliśmy model obliczeń wielowarstwowych oparty na teorii zadań jednorodnie podzielnych, algorytmy podziału danych między komputery w poszczególnych warstwach oraz algorytm szeregowania komunikacji między warstwami. Przedstawiliśmy

wyniki eksperymentów obliczeniowych dotyczących wydajności zaproponowanych algorytmów oraz wpływu parametrów systemu na strukturę otrzymywanych uszeregowań.

Prace [P4, P15] dotyczą równoważenia obciążeń poszczególnych procesorów wykonujących fazę Reduce obliczeń MapReduce w przypadku nierównomiernego rozkładu kluczy w danych wejściowych. W artykule [P4] rozważaliśmy rozkład jednostajny częstości poszczególnych kluczy w danych wejściowych, natomiast w [P15] rozkład opisany prawem Zipfa. Zaproponowaliśmy kilka algorytmów równoważenia obciążeń i przetestowaliśmy je za pomocą symulacji. Wskazaliśmy, dla jakich typów instancji problemu poszczególne algorytmy uzyskują wyniki lepsze niż pozostałe.

7.1.4 System przepływowy [P10, P16]

Napisane wspólnie z Yakovem Zinderem i Alexandrem Kononovem artykuły [P10, P16] dotyczą minimalizacji długości uszeregowania w dwumaszynowym systemie przepływowym z ograniczoną pamięcią, której rozmiar zmienia się w czasie. W pracy [P10] rozważane są zadania o jednostkowych czasach wykonywania operacji, a w [P16] zadania z dowolnymi czasami trwania operacji. Wykazaliśmy, że analizowany problem szeregowania jest silnie NP-trudny nawet w przypadku czasów jednostkowych. Pokazaliśmy, że przy pewnych dodatkowych założeniach istnieje optymalne uszeregowanie permutacyjne. Sformułowaliśmy problem jako zagadnienie całkowitoliczbowego programowania liniowego i skonstruowaliśmy dla niego wielomianowy schemat aproksymacji (PTAS). Zaproponowaliśmy również zachłanne heurystyki, algorytmy przeszukiwania lokalnego w [P10] oraz iteracyjny algorytm przeszukiwania zmiennego sąsiedztwa (IVNS) w [P16]. Przeprowadzone eksperymenty obliczeniowe wykazały, że algorytm IVNS generuje dobre rozwiązania w akceptowalnym czasie.

7.2 Nagrody i wyróżnienia

1. Nagroda absolutoryjna Dziekana Wydziału Matematyki i Informatyki UAM za bardzo dobre wyniki w nauce, działalność w Kole Naukowym Matematyków i działalność w Kole Naukowym Informatyków (2007).

2. Medal Uniwersytetu im. Adama Mickiewicza „Za wybitne osiągnięcia w nauce i wyróżniający udział w życiu Uniwersytetu" (2007).

3. Stypendium naukowe Fundacji Uniwersytetu im. Adama Mickiewicza dla doktorantów UAM na rok 2010.

4. Nagroda Naukowa Dziekana Wydziału Matematyki i Informatyki UAM za wyróżniającą się rozprawę doktorską (2011).

5. Roczne wynagrodzenie motywacyjne dla pracowników badawczo-dydaktycznych Wydziału Matematyki i Informatyki UAM za publikacje (2018, 2019, 2020).

6. Laureatka konkursu UAM „Wsparcie najbardziej produktywnej naukowo młodej kadry badawczej" w ramach projektu „Inicjatywa Doskonałości – Uczelnia Badawcza" (2020).

Literatura

[1] A. Ahmad, Z. Hanzalek, *An Energy-efficient Distributed TDMA Scheduling Algorithm for ZigBee-like Cluster-tree WSNs*, ACM Transactions on Sensor Networks 16 (2019) art. 3.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, *Wireless sensor networks: a survey*, Computer Networks 38 (2002) 393-422.

[3] J. Błażewicz, K. Ecker, E. Pesch, G. Schmidt, J. Węglarz, *Handbook on Scheduling: From Theory to Applications*, Springer, 2007.

[4] V. Bharadwaj, D. Ghose, V. Mani, T. Robertazzi, *Scheduling divisible loads in parallel and distributed systems*, IEEE Computer Society Press, Los Alamitos, California, 1996.

[5] E. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, *Hyper-heuristics: a survey of the state of the art*, Journal of the Operational Research Society 64 (2013) 1695-1724.

[6] Y.-C.Cheng, T.G.Robertazzi, *Distributed computation with communication delay*, IEEE Transactions on Aerospace and Electronic Systems 24 (1988) 700-712.

[7] T.C.E. Cheng, B.M.T. Lin, H.L. Huang, *Resource-constrained flowshop scheduling with separate resource recycling operations*, Computers & Operations Research 39 (2012) 1206-1212.

[8] Y. Cho, S. Sahni, *Preemptive Scheduling of Independent Jobs with Release and Due Times on Open, Flow and Job Shops*, Operations Research 29 (1981) 511-522.

[9] K. Choi, T.G. Robertazzi, *Divisible Load Scheduling in Wireless Sensor Networks with Information Utility*, IEEE International Performance Computing and Communications Conference 2008: IPCCC 2008, 9-17.

[10] C.-Y. Chong, S. Kumar, *Sensor Networks: Evolution, opportunities, and challenges*, Proceedings of the IEEE 91 (2003) 1247-1256.

[11] A. Chowdhury, S.A. Raut, *A survey study on Internet of Things resource management*, Journal of Network and Computer Applications 120 (2018) 42-60.

[12] T. Critchlow, K. Kleese van Dam (eds), *Data-Intensive Science*, CRC Press, 2013.

[13] J.Dean, S.Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Sixth Symposium on Operating System Design and Implementation, San Francisco, 2004, 137-150.

[14] M. Drozdowski, *Scheduling for Parallel Processing*, Springer, Londyn, 2009.

[15] M. Drozdowski, P. Wolniewicz, *Experiments with scheduling divisible tasks in clusters of workstations*, w: A.Bode, T.Ludwig, W.Karl, R.Wismuller (eds), Proceedings of 6th Euro-Par Conference. Lecture Notes in Computer Science 1900 (2000) 311–319.

[16] M. Drozdowski, P. Wolniewicz, *Optimum divisible load scheduling on heterogeneous stars with limited memory*, European Journal of Operational Research 172 (2006) 545-559.

[17] S.C. Ergen, P. Varaiya, *TDMA scheduling algorithms for wireless sensor networks*, Wireless Networks 16 (2010) 985 - 997.

[18] J. Fung, Y. Zinder, *Permutation schedules for a two-machine flow shop with storage*, Operations Research Letters 44 (2016) 153 - 157.

[19] M.R. Garey,D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[20] S. Gawiejnowicz, *Models and Algorithms of Time-Dependent Scheduling*, Springer, Berlin, 2020.

[21] W.A. Horn, *Some simple scheduling algorithms*, Naval Research Logistics Quarterly 21 (1974) 177-185.

[22] O.D. Incel, A. Ghosh, B. Krishnamachari, *Scheduling Algorithms for Tree-Based Data Collection in Wireless Sensor Networks*, w: S. Nikoletseas, J. Rolim (eds), Theoretical Aspects of Distributed Computing in Sensor Networks. Monographs in Theoretical Computer Science, Springer, Berlin, Heidelberg 2011, 407-445.

[23] R. Jain, *Comparative analysis of contention based and TDMA based MAC protocols for wireless sensor networks*, International Journal of Information Technology 12 (2020) 245–250.

[24] S.M. Johnson, *Optimal two- and three-stage production schedules with setup times included*, Naval Research Logistics Quarterly 1 (1954) 61-68.

[25] H. Kang, P. Guan, X. Liu, X. Li, *Utility-Based Divisible Sensing Task Scheduling in Wireless Sensor Networks*, Proceedings of the 2007 International Conference on Wireless Networks, Las Vegas, Nevada, USA, 2007, 342-348.

[26] T. Kaur, D. Kumar, *TDMA-based MAC protocols for wireless sensor networks: A survey and comparative analysis*, 2016 5th International Conference on Wireless Networks and Embedded Systems (WECON), Rajpura, 2016, 1-6.

[27] N. Kimura, S. Latifi, *A Survey on Data Compression in Wireless Sensor Networks*, Proceedings of International Conference on Information Technology: Coding and Computing: ITCC 2005, Vol. 2, 8-13.

[28] A. Kononov, J.S. Hong, P. Kononova, F.C. Lin, *Quantity-based buffer-constrained two-machine flowshop problem: active and passive prefetch models for multimedia applications*, Journal of Scheduling 15 (2012) 487 - 497.

[29] P. Kononova, Y. Kochetov, *The variable neighborhood search for the two machine flow shop problem with a passive prefetch*, Journal of Applied and Industrial Mathematics 7 (2013) 54–67.

[30] S. Kumar, S. Chauhan, *A Survey on Scheduling Algorithms for Wireless Sensor Networks*, International Journal of Computer Applications 20 (2011) 7-13.

[31] X. Li, V. Bharadwaj, C.C. Ko, *Distributed image processing on a network of workstations*, International Journal of Computers and Applications 25 (2003) 1–10.

[32] F.C. Lin, J.S. Hong, B.M. Lin, *A two-machine flow-shop problem with processing time-dependent buffer constraints – an application in multimedia presentations*, Computers & Operations Research 36 (2009) 1158 - 1175.

[33] B.M.T. Lin, H.L. Huang, *On the relocation problem with a second working crew for resource recycling*, International Journal of Systems Science 37 (2006) 27–34.

[34] Liu, Z., Cheng, T.C.E.: *Scheduling with job release dates, delivery times and preemption penalties*, Information Processing Letters 82, 107-111 (2002)

[35] W. Luo, Y. Xu, B. Gu, W. Tong, R. Goebel, G. Lin, *Algorithms for Communication Scheduling in Data Gathering Network with Data Compression*, Algorithmica, 80 (2018) 3158-3176.

[36] W. Luo, B. Gu, G. Lin, *Communication scheduling in data gathering networks of heterogeneous sensors with data compression: Algorithms and empirical experiments*, European Journal of Operational Research 271 (2018) 462-473.

[37] M. Moges, T.G. Robertazzi, *Wireless Sensor Networks: Scheduling for Measurement and Data Reporting*, IEEE Transactions on Aerospace and Electronic Systems 42 (2006) 327-340.

[38] T. Robertazzi, *Ten reasons to use divisible load theory*, IEEE Computer 36 (2003) 63-68.

[39] T. Robertazzi, L. Shi, *Networking and Computation. Technology, Modeling and Performance*, Springer, Cham, 2020.

[40] H. Shi, N. Kwok, W. Wang, S. Chen, *Divisible Load Theory Based Active-Sleep Workload Assignment Schemes for Wireless Sensor Networks*, International Journal of Distributed Sensor Networks 2014 (2014).

[41] R. Sinde, F. Begum, K. Njau, S. Kaijage, *Lifetime improved WSN using enhanced-LEACH and angle sector-based energy-aware TDMA scheduling*, Cogent Engineering 7 (2020) art. 1795049.

[42] R. Tadei, J.N.D. Gupta, F. Della Croce, M. Cortesi, *Minimising makespan in the two-machine flow-shop with release times*, Journal of the Operational Research Society 49 (1998) 77-85

[43] W. Tian, Y. Zhao, *Optimized Cloud Resource Management and Scheduling*, Elsevier, 2015.

[44] A.C. Valera, W.-S. Soh, H.-P. Tan, *Survey on wakeup scheduling for environmentally-powered wireless sensor networks*, Computer Communications 52 (2014) 21-36.

[45] P. Venkat Rao, C. Balaswamy, *A Review of Compressive Sensing (CS) Scheme in Cluster based Wireless Sensor Networks*, International Journal of Applied Engineering Research 13 (2018) 6118-6124.

[46] D. Vidyarthi, B.K. Sarker, A.K. Tripathi, L.T. Yang, Scheduling in Distributed Computing Systems. Analysis, Design and Models, Sprinder, 2009.

[47] Y. Yang, H. Casanova, M. Drozdowski, M. Lawenda, A. Legrand, *On the complexity of multi-round divisible load scheduling*, Research Report 6096, INRIA Rhône-Alpes 2007. https://hal.inria.fr/inria-00123711.

[48] Y. Zinder, A. Kononov, J. Fung, *A 5-parameter complexity classification of the two-stage flow shop scheduling problem with job dependent storage requirements*, Journal of Combinatorial Optimization (2021) doi:10.1007/s10878-021-00706-4.

Joanna Berlińska