Wydział Elektrotechniki, Informatyki i Telekomunikacji

Uniwersytet Zielonogórski

ROZPRAWA DOKTORSKA

Przetwarzanie równoległe w rozwiązywaniu problemu optymalnej obserwacji układów z czasoprzestrzenną dynamiką

mgr inż. Tomasz Zięba

Promotor:
prof. dr hab. inż. Dariusz Uciński

Zielona Góra, 2009

Praca naukowa finansowana ze środków na naukę w latach 2007-2010
jako projekt badawczy nr N N519 2971 33
pt. *Efektywne metody obliczeniowe dużej skali w planowaniu optymalnych strategii obserwacji procesów z czasoprzestrzenną dynamiką z zastosowaniem sieci sensorycznych.*

dla Asi

Spis treści

Spis rysunków

1

Wprowadzenie

1.1 Wstęp

W XX wieku, a zwłaszcza pod jego koniec, nastąpił gwałtowny rozwój nauki, techniki oraz przemysłu. Pojawienie się nowych, łatwo dostępnych nowoczesnych środków technicznych pozwoliło na znacznie szybsze i dokładniejsze badanie otaczającego nas świata.

Jedną z dziedzin, które dzięki temu dynamicznie się rozwijają jest szeroko rozumiane planowanie eksperymentu. Potrzeba szybszego, ale i bardziej szczegółowego modelowania otaczającego nas środowiska w celach późniejszych badań nad jego zachowaniem w różnych warunkach, stała się jednym z ważniejszych elementów rozwijającej się nauki i techniki. Oprócz opracowywanych coraz bardziej zaawansowanych technik obliczeniowych oraz metod rozwiązujących zagadnienia optymalnego planowania eksperymentu, znaczny udział w tak dynamicznym rozwoju ma gwałtowny rozwój systemów komputerowych umożliwiający szybsze, ale i przeprowadzane na znacznie większą skalę symulacje zjawisk zachodzących w otaczającym nas środowisku.

Obserwowany równolegle, obok technik informatycznych i elektroniki, rozwój mechaniki i inżynierii materiałowej powoduje, że urządzenia pozwalające na przeprowadzanie badań w rzeczywistych środowiskach stają się coraz bardziej zminiaturyzowane, dokładniejsze, a co najważniejsze — znacznie tańsze. Powszechny dostęp do tanich urządzeń i ich integracja w jednolite środowiska pomiarowe (np. sieci sensoryczne — *ang.* sensors networks) pozwalają na eksperymenty w znacznie większej skali niż w latach wcześniejszych. Potrzeba dokładnych badań i analizy coraz bardziej złożonych zjawisk przez znacznie większe instalacje pomiarowe powoduje jednak pojawianie się nowych problemów z pogranicza technik numerycznych oraz informatycznych. Olbrzymia ilość danych, jakie mogą dostarczyć przyrządy po-

miarowe, pomimo szybkiego rozwoju środowisk obliczeniowych, jest zbyt duża. W konsekwencji prowadzenie badań nad dalszym rozwojem optymalnych technik prowadzenia eksperymentu jest nadal jednym z ważniejszych priorytetów stawianych w dzisiejszych czasach naukowcom i inżynierom.

1.2 Modelowanie oraz identyfikacja

Ze względu na brak wydajnych narzędzi obliczeniowych, pierwsze modele opisujące rozmaite zjawiska fizyczne miały najczęściej postać równań różniczkowych zwyczajnych, czyli procesy były traktowane jako tzw. układy o parametrach skupionych. Na wczesnym etapie rozwoju technik komputerowych było to wystarczające, gdyż modele o parametrach skupionych były najczęściej wykorzystywane przy modelowaniu procesów chemicznych, biologicznych, zjawisk elektrycznych oraz mechanicznych i innych [120, 134, 164, 215]. Układy o parametrach skupionych pozwalają na modelowanie procesów, w których zakłada się, że stan układu jest jednorodny przestrzennie i zazwyczaj zależny jest tylko od czasu, bez uwzględniania zależności przestrzennych.

Współczesny rozwój techniki wymaga poszukiwania coraz dokładniejszych modeli matematycznych wraz z jednoczesnym uzyskiwaniem precyzyjniejszych informacji o otaczających nas zjawiskach. Wymaga to użycia coraz bardziej skomplikowanego i wysublimowanego aparatu matematycznego. Modele dla układów o parametrach skupionych przestały już wystarczać do opisywania skomplikowanych procesów zachodzących w modelowanych zjawiskach fizycznych. Procesy dynamiczne traktuje się coraz częściej jako tzw. układy z czasoprzestrzenną dynamiką (nazywane również układami o parametrach rozłożonych) [11, 44, 124, 135, 160] opisywane równaniami różniczkowymi cząstkowymi (RRCz), których rozwiązanie pozwala na wyznaczenie czasowo-przestrzennej odpowiedzi procesu dynamicznego. RRCz pozwalają nie tylko na modelowanie stanu obiektu dla zmieniającego się czasu, ale również na odwzorowanie zmian stanu obiektu w przestrzeni.

Przykład 1.1 Ochrona środowiska
Wpływ zanieczyszczeń na środowisko naturalne jest jednym z najczęściej badanych zjawisk. Rozwój przemysłu i związany z tym procesem problem narastania ilości odpadów przemysłowych oddziałujących na cały system ekologiczny planety są bardzo zauważalne i powodują coraz większe spustoszenie w przyrodzie. Badania nad opracowaniem systemów kontroli i prognozowania rozprzestrzeniania się zanieczyszczeń są prowadzone w wielu ośrodkach naukowych i agencjach rządowych [141]. Opracowywane modele pozwalają na przewidywanie możliwych skutków oraz kierunków przeciwdziałania w momencie pojawienia się zagrożenia, czy to w powietrzu, czy w wodach gruntowych. Przykładem modelu opisującego proces propagacji tlenków siarki (zanieczyszczenie pierwotne — SO_2, zanieczyszczenie wtórne — $SO_4^=$) jest następujący układ równań adwekcji-dyfuzji [89] zdefiniowany na prostokątnym obszarze przestrzennym $\Omega = L_x \times L_y \subset \mathbb{R}^2$ o brzegu $\partial\Omega$ bez uwzględnienia występowania

warstw mieszania:

$$\begin{aligned}\frac{\partial c_1}{\partial t} + v \cdot \nabla c_1 - K\nabla^2 c_1 + (v_{d_1} + k_{w_1} + k_t)c_1 &= (1-\beta)\,\mathrm{Q},\\ \frac{\partial c_2}{\partial t} + v \cdot \nabla c_2 - K\nabla^2 c_2 + (v_{d_2} + k_{w_2})c_2 - k_t c_1 &= \beta \mathrm{Q},,\end{aligned} \tag{1.1}$$

w $\Omega \times (0,T)$ z warunkami brzegowymi

$$\begin{aligned} c_k &= c_k^b \quad \text{na} \quad \{\partial\Omega \times (0,T) \mid v \cdot n < 0\},\\ \frac{\partial c_k}{\partial n} &= 0 \quad \text{na} \quad \{\partial\Omega \times (0,T) \mid v \cdot n \geqslant 0\},\end{aligned} \tag{1.2}$$

oraz warunkiem początkowym

$$c_k(0) = c_k^0 \quad \text{w} \quad \Omega \times \{t = 0\}, \tag{1.3}$$

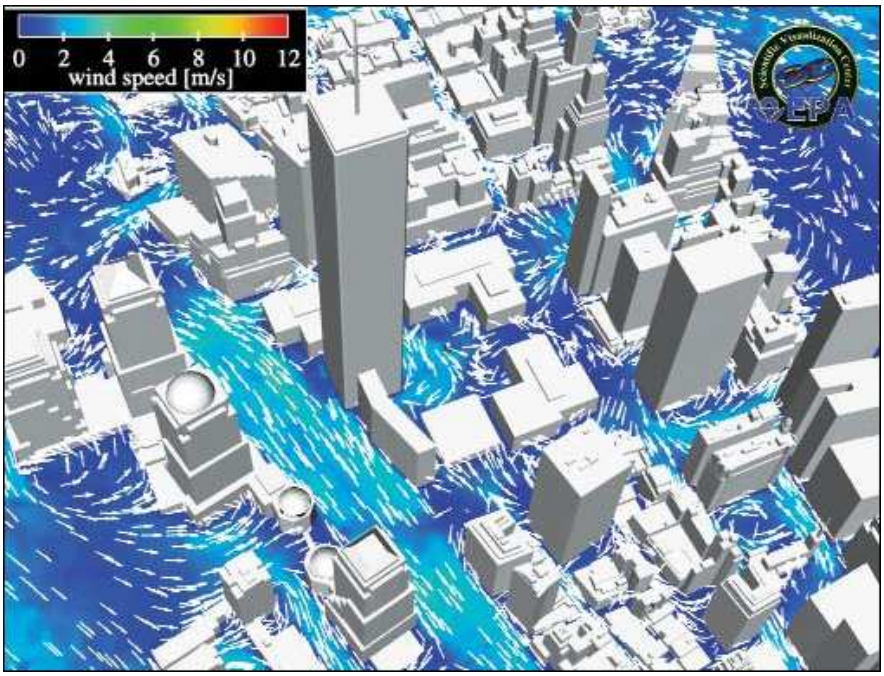
gdzie:

$x = (x_1, x_2)$ — zmienna przestrzenna,
t — czas,
$c_1 = c_1(x,t)$ — stężenia zanieczyszczeń pierwotnych (SO_2),
$c_2 = c_2(x,t)$ — stężenie zanieczyszczeń wtórnych ($SO_4^=$),
$Q = Q(x,t)$ — uśrednione pole emisji,
$k = 1, 2$ — indeks rodzaju zanieczyszczeń,
$v(x) = [u(x), v(x)]$ — wektor pola wiatru,
v_{d_k} — prędkość suchej depozycji dla $SO_2(SO_4^=)$,
k_{w_k} — współczynnik wymywania zanieczyszczeń dla $SO_2(SO_4^=)$,
k_t — współczynnik przemian chemicznych $SO_2 \to SO_4^=$,
K — współczynnik dyfuzji horyzontalnej,
T — horyzont czasowy prognozy,
β — względny udział $SO_4^=/SO_2$ w emisji,
n — jednostkowy wektor normalny skierowany na zewnątrz obszaru Ω,
$\nabla = [\partial/\partial x_1, \partial/\partial x_2]$,
$\nabla^2 = \nabla \cdot \nabla = \partial^2/\partial x_1^2 + \partial^2/\partial x_2^2$.

Ponadto, w (1.1)–(1.3) zastosowano standardowe oznaczenia rachunku wektorowego.

Przedstawiony model dotyczy rozprzestrzeniania się zanieczyszczeń atmosferycznych w skali miejsko-regionalnej. W tego typu modelach podstawowe znaczenie ma pole wiatru. W przypadku skali regionalnej większy wpływ ma tzw. *wiatr średni*, który jest definiowany jako w pełni deterministyczny ruch powietrza. Jednak w przypadku analizy modelu w skali miejskiej znaczenia nabiera *turbulencja*, która powoduje, że wiatr nabiera charakterystyki chaotycznej, znacznie komplikując proces rozprzestrzeniania się zanieczyszczeń. Przykładowe pole wiatru pomiędzy budynkami na Manhattanie w Nowym Jorku przedstawia rys. 1.1 [63]. W takim przypadku równania różniczkowe zwyczajne nie są w stanie ująć istotnej i złożonej dynamiki przestrzennej. Odwołanie się do modeli w postaci RRCz stało się więc koniecznością.

Niektóre współczynniki modelu (1.1)–(1.3) nie są bezpośrednio mierzalne (np. współczynnik dyfuzji K), więc jednym z podstawowych problemów staje się ich identyfikacja na podstawie danych pomiarowych (problem ten nazywa się czasem również kalibracją modelu). Przedstawiony model może również służyć do monitorowania stężenia smogu i określania źródeł emisji, co wiąże się z zagadnieniami estymacji stanu. Zagadnienia estymacji stanu, jak również sama identyfikacja parametryczna, są silnie związane z lokalizacją czujników pomiarowych (sensorów), która determinuje dokładność otrzymywanych rezultatów. Przykładowo, rozważmy problem wykrywania źródeł skażenia zagrażających życiu ludzkiemu oraz predykcji jego rozprzestrzeniania w metropolii. Zagadnienie optymalnego rozmieszczania oraz aktywacji sensorów zawierających detektory skażeń pozwala na wcześniejszą prawidłową kalibrację modelu rozprzestrzeniania się zanieczyszczenia, a dzięki temu — na późniejszą szybszą reakcję związaną z ewakuacją ludności lub neutralizacją źródła skażenia. Zagadnienia np. wykrywania źródeł skażeń, wymagają skomplikowanej analizy i obliczeń, dlatego odpowiednie metody i środowiska obliczeniowe powinny móc pozwolić na uzyskiwanie możliwie szybko najdokładniejszych rezultatów badań.



Rysunek 1.1: Pole wiatru na Manhattanie [63].

♦

Przykład 1.2 Ekologia
Modele o czasoprzestrzennej dynamice znajdują zastosowanie nie tylko podczas ana-

lizy wpływu ludzi na środowisko naturalne, ale również pomagają modelować zjawiska przyrodnicze w takich dziedzinach jak rolnictwo czy leśnictwo. W monografii [10] zauważono, że odpowiednie dobieranie rozkładu upraw ma wpływ na rozprzestrzenianie się populacji owadów (szkodników) wśród roślin. Do opisu zachowania się populacji w ograniczonym obszarze przestrzennym $\Omega \in \mathbb{R}^2$ najczęściej wykorzystuje się modele w postaci równań różniczkowych cząstkowych, np.

$$\begin{aligned} &\frac{\partial q}{\partial t} + \frac{\partial}{\partial x_1}(V_1 q) + \frac{\partial}{\partial x_2}(V_2 q) \\ &\quad = \frac{\partial}{\partial x_1}\left(D\frac{\partial q}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(D\frac{\partial}{\partial x_2}\right) + f \qquad \text{w } \Omega \times (0,T), \end{aligned} \tag{1.4}$$

z odpowiednimi warunkami początkowymi i brzegowymi, gdzie:

$q = q(x_1, x_2, t)$ — gęstość populacji,
t,x_1,x_2 — czas oraz zmienne przestrzenne,
V_1,V_2 — prędkości adwekcji/konwekcji,
$D = D(x_1, x_2, t)$ — współczynnik dyfuzji,
$f = f(x_1, x_2, t)$ — wymuszenie reprezentujące procesy śmierci i urodzin.

Parametry D,V_1,V_2 nie są wielkościami bezpośrednio mierzalnymi, co implikuje konieczność ich określenia na podstawie obserwacji, a to z kolei wymaga wcześniejszego rozwiązania problemu rozmieszczenia czujników pomiarowych tak, aby uzyskać możliwie najdokładniejsze estymaty.

♦

Przykład 1.3 Mechanika
Zastosowanie modeli z czasoprzestrzenną dynamiką jest szeroko rozpowszechnione w mechanice. Przykładowo, zastosowanie RRCz w optymalizacji parametrycznej oraz optymalizacji kształtu [203], a także w badaniach struktur elastycznych [10] pozwala na projektowanie i produkcję coraz bardziej wytrzymałych i niezawodnych konstrukcji przy coraz mniejszym nakładzie finansowym.

Jednym z prostszych modeli jest model drgań poprzecznych zamocowanej przegubowo belki [104]

$$\frac{\partial^2}{\partial x^2}\left(EJ\frac{\partial^2 u}{\partial x^2}\right) + \beta\frac{\partial u}{\partial t} + A\rho\frac{\partial^2 u}{\partial t^2} = f, \quad x \in (0,l],\ t \in (0,t_f], \tag{1.5}$$

z warunkami początkowymi

$$u(x,0) = \phi(x), \quad \frac{\partial u(x,0)}{\partial t} = \psi(x), \quad x \in (0,l), \tag{1.6}$$

i warunkami brzegowymi

$$\frac{\partial^2}{\partial x^2}u(0,t) = \frac{\partial^2}{\partial x^2}u(l,t) = u(0,t) = u(l,t) = 0, \quad t \in (0,t_f], \tag{1.7}$$

gdzie:

$u = u(x,t)$ — wychylenie belki w punkcie o współrzędnej x w chwili t,
$f = f(x,t)$ — liniowe obciążenie belki,
l — długość belki,
E — moduł Younga materiału belki,
J — powierzchniowy moment bezwładności przekroju belki względem osi poziomej,
β — współczynnik reprezentujący tłumienie zewnętrzne,
A — powierzchnia przekroju poprzecznego belki,
ρ — gęstość materiału belki,
ϕ — funkcja opisująca wychylenie belki w chwili początkowej,
ψ — funkcja opisująca prędkość belki w chwili początkowej.

Przypuśćmy, że zewnętrzna siła f powodująca liniowe obciążenie belki zależy, oprócz miejsca nacisku x oraz czasu t, również od pewnych nieznanych parametrów θ sterujących siłą nacisku na belkę. Problem optymalnego rozmieszczenia czujników pomiarowych można zdefiniować jako poszukiwanie miejsc rozmieszczenia sensorów nacisku (zbudowanych w oparciu o elementy piezoelektryczne), w celu poprawnej identyfikacji nieznanych parametrów θ, tak aby w konsekwencji siła nacisku nie powodowała wychylenia belki poza wychylenie dopuszczalne.

♦

Równania różniczkowe cząstkowe określają relację pomiędzy funkcją dwóch lub większej liczby zmiennych niezależnych, a pochodnymi cząstkowymi tej funkcji względem tych zmiennych. Ogólna postać quasi-liniowego RRCz drugiego rzędu z dwiema zmiennymi niezależnymi ma postać [88]:

$$A\frac{\partial^2 s}{\partial x^2} + B\frac{\partial^2 s}{\partial x \partial y} + C\frac{\partial^2 s}{\partial y^2} + D\frac{\partial s}{\partial x} + E\frac{\partial s}{\partial y} + Fs = G, \tag{1.8}$$

gdzie:

x,y — zmienne niezależne,
$s = s(x,y)$ — zmienna zależna (stan układu),
A,B,C — współczynniki mogące zależeć od x,y,$\partial s/\partial x$ lub $\partial s/\partial y$,
D,E,F — współczynniki mogące zależeć od x,y oraz s,
G — człon mogący zależeć od x oraz y.

Przedstawiona forma równania RRCz jest jedną z wielu możliwych. Równania mogą posiadać znacznie więcej zmiennych niezależnych (analiza procesu dyfuzji w przestrzeni trójwymiarowej wymaga czterech zmiennych niezależnych: trzech dotyczących zmiennych przestrzennych oraz jednej dotyczącej czasu), jak również można rozpatrywać pochodne cząstkowe wyższych rzędów. Aby równanie posiadało jednoznacznie określone rozwiązanie, należy dodatkowo podać warunki brzegowe oraz początkowe. Warunki brzegowe definiują warunki na granicy analizowanego obszaru. Wśród najczęściej spotykanych warunków brzegowych wyróżniamy: warunek Dirichleta (określa wartość zmiennej zależnej na brzegu) oraz warunek Neumanna (określa pochodną w kierunku normalnym do brzegu).

Równania różniczkowe cząstkowe można rozwiązywać w sposób analityczny, jednak ze względu na duży stopień skomplikowania odpowiedniej procedury, a najczęściej również niemożność określenia rozwiązania analitycznego [153, 164], w większości przypadków rozwiązanie określane jest w sposób numeryczny. Najczęściej wykorzystywanymi metodami numerycznego rozwiązywania RRCz są metody oparte na schematach różnicowych [3, 112] oraz metody elementu skończonego [73, 109, 112].

Zastosowanie metod różnic skończonych polega na dyskretyzacji równań w taki sposób, aby zastąpić występujące w nich pochodne cząstkowe ilorazami różnicowymi poprzez rozwinięcie funkcji w szereg Taylora w poszczególnych punktach siatki dyskretyzacji. Przykładowo, rozwinięcie funkcji s jednej zmiennej x w szereg Taylora w punkcie x_i przedstawia się następująco:

$$s(x_i + \Delta x) = s(x_i) + \Delta x \left.\frac{\mathrm{d}s}{\mathrm{d}x}\right|_{x=x_i} + \mathcal{O}(\Delta x^2), \tag{1.9}$$

gdzie: Δx — krok dyskretyzacji, $i = 0, \ldots, M$ — indeks kolejnych węzłów siatki dyskretyzacji, M — liczba węzłów siatki. Następnie wyodrębniamy pochodną

$$\left.\frac{\mathrm{d}s}{\mathrm{d}x}\right|_{x=x_i} = \frac{s(x_i + \Delta x) - s(x_i)}{\Delta x} - \mathcal{O}(\Delta x) \tag{1.10}$$

i otrzymujemy interesujący nas iloraz różnicowy, poprzez odrzucenie elementów szeregu stopnia wyższego

$$\left.\frac{\mathrm{d}s}{\mathrm{d}x}\right|_{x=x_i} \approx \frac{s_{i+1} - s_i}{\Delta x}, \tag{1.11}$$

gdzie: $s_i = s(x_i)$, nazywany pierwszym ilorazem różnicowym przednim [66, 112]. Analogiczne wzory można otrzymać dla ilorazów wyższych rzędów oraz funkcji wielu zmiennych. Przykładowo, przybliżenie Laplasjanu

$$\nabla^2 s = \frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2}, \tag{1.12}$$

funkcji s dwóch zmiennych niezależnych, przedstawia się następująco:

$$\nabla^2 s \approx \frac{s_{i+1,j} - 2s_{i,j} + s_{i-1,j}}{(\Delta x)^2} + \frac{s_{i,j+1} - 2s_{i,j} + s_{i,j-1}}{(\Delta y)^2}, \tag{1.13}$$

gdzie: $s_{i,j} = s(x_i, y_j)$ — wartość funkcji s w punkcie (x_i, y_j) będącym węzłem siatki dyskretyzacji, Δx i Δy — kroki dyskretyzacji odpowiednio względem zmiennych x i y.

Wybór sposobu dyskretyzacji jest jednym z kluczowych momentów rozwiązywania danego równania ponieważ ma niebagatelny wpływ na zbieżność oraz na wielkość błędów wynikających z dyskretyzacji równania. Przedstawione powyżej podstawowe sposoby dyskretyzacji zmiennych przestrzennych prowadzą do dużych układów równań różniczkowych zwyczajnych. W większości przypadków, zamiast

rozwiązywać tak zdefiniowany problem, dyskretyzuje się również zmienną związaną z czasem. Wśród metod dyskretyzacji względem zmiennej odpowiadającej czasowi można wyróżnić schematy jawne i niejawne, np. metodę Cranka-Nicolsona i inne [3, 44, 73, 88, 112, 135, 137].

W przypadku dyskretyzacji zmiennej czasowej, oprócz dokładności związanej z wyborem kroku dyskretyzacji zmiennej czasowej pojawia się problem zbieżności oraz stabilności wybranej metody dyskretyzacji. Przykładowo, rozważmy uproszczone jednowymiarowe paraboliczne równanie dyfuzji postaci

$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} \tag{1.14}$$

gdzie: D — współczynnik dyfuzji. W celu aproksymacji pochodnej związanej z czasem wykorzystajmy iloraz różnicowy przedni [66, 112], stąd

$$\frac{\partial s}{\partial t} \approx \frac{s_i^{\tau+1} - s_i^{\tau}}{\Delta t} \tag{1.15}$$

gdzie: Δt — krok czasowy, $s_i^{(\tau)}$ — wartość funkcji s w punkcie i w kroku czasowym τ.

W przypadku stosowania schematu jawnego (*ang.* explicit scheme) [44, 66, 112, 137], aproksymację pochodnych przestrzennych wykonuje się tak, aby do wyznaczenia rozwiązania w danym kroku czasowym $(\tau + 1)$ wykorzystać wartości obliczone w kroku poprzednim (τ). Przykładowo, stosując metodę różnic centralnych dla pochodnej zmiennej przestrzennej x otrzymujemy następujący schemat różnicowy (przy założeniu, że $D = 1$):

$$\frac{\partial^2 s}{\partial x^2} \approx \frac{s_{i+1}^{\tau} - 2s_i^{\tau} + s_{i-1}^{\tau}}{(\Delta x)^2}, \tag{1.16}$$

co, po przekształceniach, prowadzi do rozwiązywania następującego równania:

$$s_i^{\tau+1} = s_i^{\tau} + \lambda\Big(s_{i+1}^{\tau} - 2s_i^{\tau} + s_{i-1}^{\tau}\Big), \tag{1.17}$$

gdzie: $\lambda = \Delta t/(\Delta x)^2$.

Niestety, stosowanie schematu jawnego wymaga narzucenia ograniczeń na krok czasowy, w celu uzyskania stabilności tej metody. W przypadku dowolnego doboru kroku czasowego Δt oraz kroku dyskretyzacji zmiennej przestrzennej Δx mogą pojawiać się narastające, niepomijalne oscylacje błędów w uzyskiwanych wynikach.

Rozwiązaniem problemów związanych ze stosowaniem schematu jawnego jest wykorzystanie schematu niejawnego (*ang.* implicit scheme) [44, 112, 137]. Metody niejawne pozwalają na zachowanie stabilności, kosztem jednak znacznego skomplikowania sposobu wyznaczania rozwiązań. Przykładowo, w przypadku schematu różnic wstecznych pochodną przestrzenną wyznacza się w czasie $(\tau + 1)$, czyli

$$\frac{\partial^2 s}{\partial x^2} \approx \frac{s_{i+1}^{\tau+1} - 2s_i^{\tau+1} + s_{i-1}^{\tau+1}}{(\Delta x)^2}, \tag{1.18}$$

W połączeniu z aproksymacją pochodnej czasowej (1.15) otrzymujemy następujące równanie

$$s_i^\tau = -\lambda s_{i+1}^{\tau+1} + (1+2\lambda)s_i^{\tau+1} - \lambda s_{i-1}^{\tau+1}, \tag{1.19}$$

gdzie: $\lambda = \Delta t/(\Delta x)^2$. Rozwiązanie tak przedstawionego problemu, przy określeniu warunku początkowego oraz warunków brzegowych, wymaga rozwiązywania układów równań algebraicznych dla każdego kroku czasowego, co wydłuża czas określania rozwiązania końcowego.

Metodę różnic skończonych zazwyczaj wykorzystuje się jeżeli analizowany obszar ma nieskomplikowany i regularny kształt. W przypadku modelowania procesu w przestrzennym obszarze o kształcie nieregularnym, najczęściej wykorzystywana jest metoda elementów skończonych (MES). Metoda ta polega na podziale analizowanego obszaru na regularne elementy będące wielokątami (związanymi z odpowiednimi funkcjami kształtu) [88, 112, 127, 137]. W przypadku przestrzeni jednowymiarowej są to odcinki, w przypadku przestrzeni dwuwymiarowej elementem może być trójkąt lub czworokąt, a w przypadku przestrzeni trójwymiarowej obszar jest dzielony na czworościany lub sześciany. Na każdym z takich elementów definiowana jest funkcja przybliżająca lokalnie rozwiązanie wewnątrz danego elementu. Przybliżone rozwiązanie opisane jest następująco

$$s(\cdot) \approx \sum_{i=1}^{N} c_i \phi_i(\cdot), \tag{1.20}$$

gdzie: N — liczba elementów skończonych, c_i — i-ta waga skojarzona z i-tą funkcją kształtu. Rozwiązanie polega na takim dobraniu wag skojarzonych ze znanymi funkcjami kształtu, aby błąd aproksymacji był jak najmniejszy, czyli żeby MES jak najdokładniej przybliżała rozwiązanie równania RRCz. Zbiór takich elementów wraz z odpowiednimi funkcjami interpolującymi wewnątrz elementu prowadzi do sformułowania układu równań algebraicznych, który opisuje rozważany proces. W celu określenia wartości wag można zastosować np. metodę Galerkina lub metodę oparta na sformułowaniu wariacyjnym — metodę Ritza [112, 253]. W efekcie dyskretyzacja liniowych RRCz prowadzi do układu równań liniowych ze względu na wagi.

Dyskretyzacja występująca zarówno w metodzie różnic skończonych, jak i w metodzie elementu skończonego, powoduje, że aproksymacja rozwiązania liniowego RRCz sprowadza się do rozwiązania klasycznego układu równań liniowych [88] postaci:

$$\mathrm{Au} = \mathrm{b}, \tag{1.21}$$

gdzie: A, b — macierz i wektor współczynników określonych przez wybraną metodę, schemat dyskretyzacji czy też funkcję interpolującą, u — wartości rozwiązania przybliżonego w węzłach przyjętej siatki przestrzennej. Po sprowadzeniu RRCz do postaci układu równań algebraicznych wydaje się, że problem stał się trywialny. Sposoby bezpośredniego rozwiązywania układów równań liniowych są znane od dawna

i łatwe w implementacji. Jednakże, w tym przypadku macierz A jest prawie zawsze słabo uwarunkowaną macierzą o olbrzymiej wymiarowości (przykładowo, jeżeli obszar dwuwymiarowy, w którym analizowany jest model, zostanie podzielony na siatkę o wielkości 100×100, to macierz A będzie miała wymiar 10000×10000), chociaż sytuację do pewnego stopnia upraszcza fakt, że bardzo często będzie to macierz rzadka, składająca się w większości z elementów zerowych. Aby rozwiązać tego typu układy równań, oprócz klasycznej metody eliminacji Gaussa (w tym przypadku bardzo nieefektywnej) wykorzystuje się m.in. metodę Fouriera [82, 244], metodę spektralną Czebyszewa, [222] oraz metody należące do grupy metod relaksacyjnych [3]. Te ostatnie wykorzystują strukturę macierzy rzadkiej A i dekomponują ją na różnicę

$$\mathrm{A} = \mathrm{E} - \mathrm{F}, \tag{1.22}$$

gdzie: E — macierz, którą łatwo odwrócić. Otrzymuje się w ten sposób układ równań

$$\mathrm{Eu} = \mathrm{Fu} + \mathrm{b}, \tag{1.23}$$

stanowiący podstawę iteracyjnego wyznaczania rozwiązania wg wzoru (startując od pewnego $\mathrm{u}^{(0)}$)

$$\mathrm{Eu}^{k+1} = \mathrm{Fu}^{k} + \mathrm{b}, \tag{1.24}$$

aż do momentu osiągnięcia zadanej dokładności. Efektywność tej metody zależy od szybkości wyznaczania odwrotności macierzy E.

Na rynku dostępny jest szereg narzędzi mających na celu ułatwienie tworzenia modeli, doboru parametrów oraz rozwiązywania skomplikowanych równań różniczkowych cząstkowych. Możliwość wyboru bardzo złożonych obszarów, wielkości siatek dyskretyzacji oraz dostępność wielu predefiniowanych modeli odzwierciedlających dobrze znane zjawiska fizyczne daje możliwość skupienia się na tworzeniu modelu pozostawiając kwestie techniczne jako problem drugoplanowy. Narzędzia ułatwiające pracę z modelami w postaci RRCz dostępne są zarówno w postaci dedykowanych aplikacji, np. Comsol [36], FreeFEM++ [68] lub FASTFLO [55], jak i bibliotek procedur numerycznych, np. NAG [155], IMSL [95], PETSc [175], DiffPack [47]. Ze względu na wielkość oraz olbrzymi stopień skomplikowania układów równań opisujących obiekty rzeczywiste, do celów ich rozwiązywania wykorzystywane są często równoległe środowiska obliczeniowe umożliwiające uzyskanie wyników zarówno w krótszym czasie, jak i dla bardziej skomplikowanych modeli.

Obecnie modele złożonych obiektów/procesów opisywane za pomocą RRCz są wykorzystywane w niemal każdej dziedzinie nauki. Sam proces modelowania może być rozważany z kilku punktów widzenia. Obecny rozwój nauki i techniki wymaga coraz dokładniejszego i szybszego wytwarzania nowych produktów i technologii. Nierzadko są to rezultaty pionierskich badań, które wcześniej nie były przeprowadzane. W takich przypadkach konieczne staje się wykorzystanie modelowania matematycznego i przeprowadzenie badań na fizycznie nieistniejących obiektach, tak aby poznać ich cechy oraz właściwości. Tworzenie realnych modeli jest również bardzo kosztowne, stąd symulacje z zastosowaniem modeli matematycznych pomagają

znacznie zmniejszyć koszty badań i późniejszej produkcji. Innym punktem spojrzenia na zastosowanie modeli jest modelowanie obiektów/zjawisk występujących w przyrodzie.

Pomijając zagadnienie wyboru modelu, który najlepiej potrafi odwzorować zachowanie się rzeczywistego obiektu, bardzo ważnym zagadnieniem jest identyfikacja parametrów opisujących dany model. Zadanie identyfikacji parametrów obiektu polega na znalezieniu prawdziwych wartości wektora parametrów systemu lub takiego wektora współczynników, dla którego założony model najlepiej przybliża rzeczywisty system. Dokładne oszacowanie wartości nieznanych parametrów obiektu pozwala m.in. na poprawną symulację zachowania się modelu w warunkach, które rzadko występują w rzeczywistości, a mogą prowadzić np. do uszkodzenia elementów składowych badanego obiektu. Metody pozwalające na identyfikację parametrów modelu działają na zasadzie minimalizacji pewnego wskaźnika niedopasowania modelu (jakości identyfikacji) ze względu na poszukiwany wektor parametrów. Wskaźnik niedokładności modelu jest opisywany przez błąd wyjściowy modelu, definiowany jako różnica pomiędzy odpowiedzią generowaną przez model, a odpowiedzią rzeczywistego systemu. Do najpopularniejszych metod identyfikacji parametrycznej należą [151]: metoda najmniejszych kwadratów [116, 128, 220, 243], metoda zmiennych instrumentalnych [128, 220], metoda największej wiarygodności [128, 220, 243] czy też metoda największego prawdopodobieństwa *a posteriori* [128, 220]. Oprócz zalet w postaci dużej szybkości zbieżności, prostoty z inżynierskiego punktu widzenia, metody identyfikacji parametrycznej nie zabezpieczają przed błędnym wyborem modelu procesu jaki chcemy opisywać.

Oprócz przyjęcia właściwego modelu badanego obiektu, w celu możliwie dokładnej estymacji parametrów modelu, należy poprawnie zaplanować strategię obserwacji badanego obiektu/zjawiska. Proces planowania procesu obserwacji jest jednym z zagadnień rozważanych w dziedzinie optymalnego planowania eksperymentu. Dobrze zaplanowany proces obserwacji pozwala na późniejszą precyzyjną estymację parametrów modelu, a co za tym idzie — poprawne odwzorowanie zachowania obiektu/zjawiska przez opracowany model.

1.3 Optymalne planowanie eksperymentu

Początek okresu rozwoju technik planowania eksperymentu można umiejscawiać w latach dwudziestych i trzydziestych XX w. Teoria planowania eksperymentu była pierwotnie wykorzystywana w dziedzinie eksperymentów rolniczych, które miały na celu rozstrzygnięcie o doborze korzystnych gatunków lub sposobów upraw. Ze względu na długie okresy wegetacji roślin, złe zaplanowanie eksperymentu mogło zostać naprawione dopiero w kolejnych latach. W latach trzydziestych techniki planowania zostały dostrzeżone przez przemysł, przede wszystkim chemiczny [41], gdzie planowane eksperymentu służyło m.in. do doboru składu mieszanin o pożądanych własnościach. W kolejnych dekadach techniki planowania eksperymentu pojawiały

się w coraz to większej liczbie gałęzi przemysłu i w coraz to bardziej skomplikowanych zastosowaniach. Planowanie eksperymentu zaczęto stosować nie tylko w dziedzinach nauk przyrodniczych: meteorologii [16], hydrologii [206], oceanografii, czy w zagadnieniach monitorowania i prognozowania zanieczyszczeń w atmosferze [157], ale również w przemyśle farmaceutycznym [58], mechanice [11], inżynierii chemicznej [4] oraz przemyśle motoryzacyjnym i telekomunikacyjnym [196], energetyce jądrowej [113], a także w procesie monitorowania jakości produkcji [189].

Zagadnienia rozważane w niniejszej rozprawie dotyczą obiektów z czasoprzestrzenną dynamiką, których model jest znany, jednak zawiera nieznane parametry, których estymaty należy wyznaczyć. W celu oceny tych parametrów można wyobrazić sobie możliwość obserwacji rozważanego procesu we wszystkich punktach obszaru, jednak w praktyce tego typu podejście jest rzadko możliwe. W tym momencie pojawia się problem dobrania takich punktów obserwacji, aby osiągnąć możliwie największą, według założonego kryterium, dokładność identyfikacji [118, 170, 186, 189, 223, 228]. Przykładem może być proces dyfuzji i transportu zanieczyszczeń gazowych w atmosferze opisany za pomocą równania adwekcji-dyfuzji (zob. Przykład 1.1). Niektóre z jego parametrów nie są bezpośrednio mierzalne (np. współczynnik dyfuzji) i należy dokonać ich estymacji na podstawie danych pomiarowych. Jednak problem zebrania danych niosących najwięcej informacji o parametrach jest uwarunkowany przede wszystkim miejscem dokonywania pomiarów, co implikuje potrzebę znalezienia optymalnych położeń czujników, tak aby uzyskana dokładność identyfikacji była jak największa. Algorytmy planowania eksperymentu powinny umożliwić wybór punktów przestrzennych, w których należy wykonywać pomiary, zapewniające największą dokładność estymat nieznanych parametrów procesu, otrzymywanych w wyniku następującej później estymacji parametrycznej [10, 204]. Wybór miejsc dokonywania pomiarów przez zadaną liczbę urządzeń pomiarowych staje się problemem istotnym nie tylko z punktu widzenia technik obserwacji, ale również z punktu widzenia ekonomicznego oraz praktycznego. Zbyt duża liczba wykonywanych pomiarów przez zbyt dużą liczbę urządzeń prowadzić może nie tylko do znaczącego zwiększenia czasu przetwarzania danych, ale również do wydłużenia całego procesu pobierania danych. Tego typu opóźnienia są zazwyczaj nieznaczące w przypadku badań prowadzonych w laboratorium, jednak w przypadku zagadnień z dziedziny ochrony środowiska stają się priorytetowymi. Przykładowo, szybkie i dokładne wyznaczenie źródeł skażenia powietrza jest kluczowe dla ekip prowadzących akcję ratunkową.

Można wyróżnić trzy problemy implikowane przez możliwe strategie obserwacji procesów z czasoprzestrzenną dynamiką:

- problem optymalnego rozmieszczenia czujników stacjonarnych [136, 186, 170, 228],
- zagadnienie skanowania, czyli optymalnej aktywacji tylko niektórych stacjonarnych czujników w zadanych chwilach czasowych [152, 170, 228],
- określenie optymalnych trajektorii i prędkości poruszania się czujników rucho-

mych [26, 187, 228].

Problem optymalnej obserwacji przy pomocy czujników stacjonarnych można określić jako wybór miejsc wykonania pomiarów przez pewną liczbę czujników wśród określonej liczby możliwych do wykorzystania lokalizacji. Miejsca możliwych do wykonania przez czujniki pomiarów są często dane *a priori*. Sieci czujników stacjonarnych przez cały okres wykonywania pomiarów nie zmieniają położenia, ani konfiguracji aktywnych urządzeń.

Zastosowania czujników stacjonarnych pojawiają się w wielu dziedzinach, począwszy od stacji pogodowych czy sejsmicznych, a skończywszy na czujnikach mierzących naprężenia skrzydeł w samolotach. Sieci czujników stacjonarnych najczęściej montuje się w miejscach umożliwiających zarówno bezpośrednie dostarczenie zasilania do urządzenia, jak i obecność medium umożliwiającego transmisję danych.

Wraz z rozwojem techniki, ciągłym zmniejszaniem się kosztów produkcji czujników oraz dużą dostępnością, bardzo praktyczne stało się stosowanie dużych sieci czujników skanujących. Idea czujników skanujących polega na umieszczeniu czujników stacjonarnych w miejscach do tego przeznaczonych i odpowiedniej aktywacji pewnego podzbioru czujników w zależności od dynamiki badanego zjawiska. W przypadku skanujących sieci sensorycznych liczba aktywnych czujników jest często znacznie mniejsza niż liczba czujników stacjonarnych potrzebna do osiągnięcia tej samej dokładności. Dzięki możliwości dynamicznej rekonfiguracji czujników aktywnie wykonujących pomiary, sieci czujników skanujących pozwalają na zwiększenie dokładności prowadzonych pomiarów oraz obszaru na jakim prowadzone są obserwacje. Możliwość dynamicznej aktywacji czujników powoduje, że zmniejsza się wykorzystanie pasma transmisyjnego potrzebnego do przesłania wyników pomiarów, a dzięki temu zmniejszeniu ulega również wykorzystanie energii zasilającej urządzenia.

Badania związane z inżynierią materiałową, a zwłaszcza zwiększone zainteresowanie materiałami polimerowymi, oraz rozwój mikroelektroniki umożliwiły gwałtowny rozwój badań nad urządzeniami typu MEMS (*ang.* Micro Electro-Mechanical Systems) [94, 205, 245]. Jako MEMS określa się miniaturowe urządzenia elektromechaniczne, najczęściej rozmiarami zawierającymi się między kilkoma mikrometrami, a kilkoma centymetrami. Urządzenia MEMS mogą posiadać wbudowane akcelerometry, żyroskopy, mikrofony, czujniki ruchu, ciśnienia, temperatury oraz wiele innych urządzeń pomiarowych. Ich zastosowanie wydaje się być nieograniczone, od urządzeń wykorzystywanych w przemyśle motoryzacyjnym, poprzez urządzenia medyczne, aż po układy znajdujące się w urządzeniach elektroniki użytkowej, tj. telefony komórkowe czy też aparaty fotograficzne.

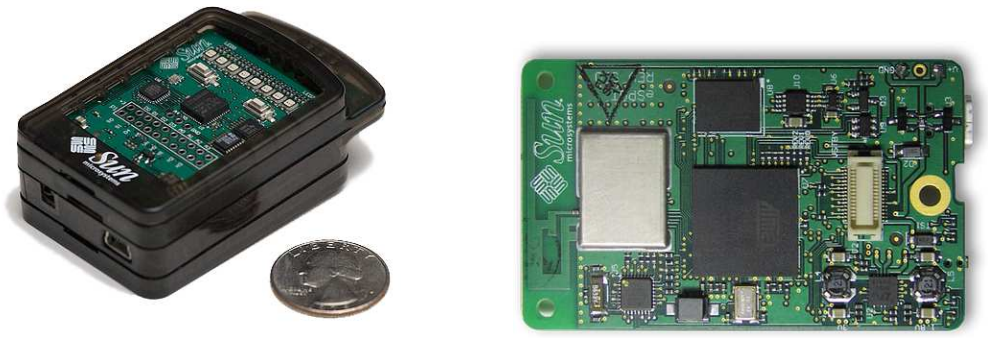
Rozwój mechaniki, a zwłaszcza urządzeń typu MEMS w połączeniu z, miniaturyzacją elementów zasilających oraz napędu elektrycznego spowodowały, że ostatnimi czasy prawdziwy boom przeżywają urządzenia mobilne zdolne do przenoszenia czujników. Łatwa dostępność urządzeń mobilnych, czy to w postaci pojazdów poruszających się po powierzchni ziemi (pojazdy lądowe, wodne), czy poruszających się w powietrzu — UAV [238] (*ang.* Unmanned Aerial Vehicle — bezzałogowy pojazd

latający) daje znacznie większe możliwości w przypadku analizy zjawisk posiadających trójwymiarową dynamikę przestrzenną. Przykładowo, pojazdy UAV najczęściej występują pod postacią samolotu z napędem odrzutowym lub śmigłowym, ale pojawiają się również rozwiązania, w których pojazd naśladuje ruchy skrzydeł owadów. Na pokładzie można zainstalować sprzęt obserwacyjny (optyczny, akustyczny i radiowy) oraz różnego rodzaju czujniki środowiskowe.

Olbrzymie możliwości tego typu obiektów przenoszących czujniki prowadzą jednocześnie do nietrywialnych zagadnieniem z punktu widzenia ich sterowania. Potencjalnie większa możliwość kolizji, konieczność uwzględniania dynamiki pojazdu lub monitorowanie wykorzystania energii w celu bezpiecznego powrotu pojazdu na ziemię są tylko nielicznymi zagadnieniami jakie należy uwzględniać przy wykorzystaniu tego typu urządzeń.

1.4 Sieci sensoryczne

W ostatnich latach bardzo szybko i gwałtownie rozwijają się badania nad projektowaniem, tworzeniem i zastosowaniem sieci czujników nazywanymi też sieciami sensorycznymi (*ang.* sensor networks). Ocenia się, że sieci czujników są jedną z najważniejszych technologii, która będzie się najmocniej rozwijała w XXI w. [32]. Na sieć czujników składa się wiele urządzeń, tzw. motów (*ang.* mote, zob. rys. 1.2 [208]), rozmieszczonych w obszarze działania badanego zjawiska, mogących komunikować się pomiędzy sobą i ewentualnie przenoszących różnego typu instrumenty badawcze. Sieci sensoryczne zbudowane są z motów, zazwyczaj zbudowanych z następujących podstawowych elementów: modułu zasilania, modułu komunikacji bezprzewodowej, mikroprocesora oraz ustandaryzowanego interfejsu komunikacyjnego umożliwiającego podłączenie szerokiej gamy zewnętrznych czujników, np. czujników ruchu, światła, temperatury, ciśnienia, biosensorów czy też detektorów skażeń chemicznych [205, 245].



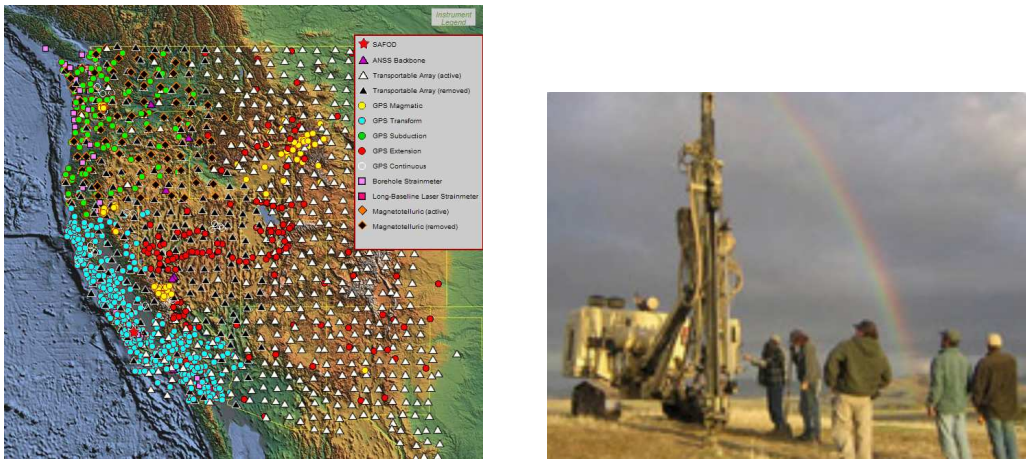
Rysunek 1.2: Przykład sensora/mota na podstawie produktu SunSPOT.
Źródło: http://www.btnode.ethz.ch/Projects/SunSPOT.

Zasadniczą różnicą pomiędzy sensorami/motami a klasycznymi czujnikami jest

obecność procesora mogącego wykonywać różne obliczenia. Czujniki w klasycznej postaci przesyłają dane do centralnego punktu przetwarzania, który na podstawie uzyskanych danych decyduje o kolejnych pomiarach czy też reakacji na pobrane próbki. Podejście zastosowane w sieciach sensorycznych powoduje, że przetwarzają one dane w sposób rozproszony, znacznie zmniejszając liczbę danych konieczną do przesłania pomiędzy sensorami. Procesor zamontowany w sensorze może przetworzyć dane lokalnie i przesłać do swoich sąsiadów tylko niezbędne im informacje. Zaletą tej metody jest znacznie mniejsze zapotrzebowanie na energię potrzebną do przesyłania informacji. Brak centralnego punktu przetwarzania informacji powoduje jednak, że w danej chwili sensory składające się na całość systemu pomiarowego nie znają globalnego stanu tego systemu. Dodatkowo czujniki mogą przesyłać sygnały tylko pomiędzy sąsiadami powodując, że dane przesyłane są na zasadzie fali od źródła sygnału do miejsca docelowego. Sposób organizowania komunikacji pomiędzy czujnikami oraz wybór ścieżki marszruty dla przesyłanej informacji jest problemem szeroko omawianym w literaturze [22, 247].

Z problemem komunikacji pomiędzy sensorami związany jest bezpośrednio problem oszczędności energii. Jedną z najbardziej energochłonnych funkcji sensora jest proces nawiązywania i przesyłania informacji. Opracowanie algorytmów zmniejszających częstość wymiany informacji pozwala oszczędzić energię zmagazynowaną w źródle zasilania, a tym samym wydłużyć czas pracy całego sensora [94]. Urządzenia wchodzące w skład sieci sensorycznych mogą również kontrolować inne urządzenia na podstawie instrukcji przesyłanych pomiędzy sensorami. W zależności od konstrukcji, sensory mogą być również wyposażone w mechanizmy napędowe umożliwiające ich poruszanie.

Początkowo sieci czujników były wykorzystywane do celów wojskowych przez amerykańską armię na potrzeby detekcji położeń nieprzyjacielskich okrętów podwodnych [32]. Wraz z rozwojem techniki, urządzenia stawały się coraz bardziej wyrafinowane, zminiaturyzowane, ale co najważniejsze — coraz tańsze. W literaturze można spotkać wiele publikacji poświęconych technicznym zagadnieniom związanym z sieciami sensorycznymi [201, 205]. Aktualne prace badaczy skupiają się na problemach organizacji sensorów w sieci, optymalnej komunikacji między nimi, zarządzaniu energią czy też bezpieczeństwem przesyłanych bezprzewodowo informacji [22, 245, 247]. Łatwy dostęp do wyrafinowanych sensorów pozwala na tworzenie sieci czujników liczących setki, a nawet tysiące urządzeń. Przykładem może być projekt *EarthScope* [49], którego celem jest poznanie struktury oraz zmian zachodzących w płaszczu skorupy ziemskiej pod kontynentem Ameryki Północnej, w celu przewidywania trzęsień ziemi oraz wybuchów wulkanów. Aktualnie na rzecz projektu pracuje ponad 1700 urządzeń (sejsmografy, urządzenia GPS, czujniki laserowe mierzące zmiany rozmiarów skał — LSM) znajdujące się przede wszystkim na zachodnim wybrzeżu USA (zob. rys. 1.3). Innym przykładowym projektem prowadzonym na szerszą skalę jest interdyscyplinarny projekt *SwissEx* (Swiss Experiment Interdisciplinary Environmental Research) [213]. Celem projektu jest stworzenie systemu umożliwiającego współpracę różnych grup naukowców badających środowisko natu-



(a) mapa rozmieszczenia czujników (b) czujnik LSM

Rysunek 1.3: Projekt EarthScope.
Źródło: http://www.earthscope.org/.

ralne w Szwajcarii. Na cały projekt składa się praca grup roboczych zajmujących się poszczególnymi zagadnieniami. Do najważniejszych projektów składowych można zaliczyć: budowę modelu i predykcję zachowania się wód na terenie Szwajcarii w kontekście występowania powodzi (APUNCH) [209], badanie interakcji pomiędzy biosferą, a geosferą (BigLink) [210], projekt badawczy dotyczący trzęsień ziemi (COGEAR) [211], badanie i przewidywanie osunięć ziemi ze wzgórzy (TRAMM) [212].

Zastosowanie sieci sensorycznych nie ogranicza się tylko do monitorowania środowiska naturalnego. Wśród ciekawych zastosowań można wyróżnić projekt „Loch Rannoch" dotyczący monitoringu stanu maszynerii pokładowej zlokalizowanej na tankowcach firmy BP [199]. Projekt był prowadzony przez czołową firmę dostarczającą rozwiązania oparte na sieciach sensorycznych — firmę Crossbow [40]. Projekt polegał na zaprojektowaniu sieci sensorycznej składającej się ze 150 czujników przyspieszenia (akcelerometrów) i zainstalowaniu jej w maszynowni tankowca przewożącego ropę naftową. Dane zbierane przez czujniki pozwalają na ocenę stanu maszyn zainstalowanych na statku, poprzez analizę powodowanych przez nie wibracji. Zebrane informacje pozwalają również na detekcję uszkodzeń i przewidywanie awarii urządzeń, co ma istotne znaczenie w przypadku tak dużej jednostki pływającej jaką jest tankowiec przewożący ładunek bardzo szkodliwy na środowiska naturalnego.

Wśród wielu badań nad zastosowaniami sieci czujników w przemyśle, ekologii, biologii [94] oraz wielu innych pojawiają się coraz częściej badania dotyczące optymalnego planowania rozmieszczenia oraz poruszania się tego typu urządzeń w badanym środowisku [45, 102, 103, 154, 165, 170, 234, 236]. Dotychczas sposób rozmieszczenia oraz poruszania się urządzeń był kontrolowany przez człowieka na podstawie jego wiedzy eksperckiej, najczęściej bez uwzględniania znajomości modelu matematycznego badanego zjawiska. Jednak wymagania stawiane przez rozwój techniki

oraz potrzeba coraz dokładniejszej i szybszej analizy obserwacji dostarczanych przez czujniki uzasadniają potrzebę stosowania podejść bardziej systematycznych.

Zagadnienia związane z projektowaniem, tworzeniem oraz zastosowaniem sieci sensorycznych idealnie wpasowują się w problematykę optymalnego rozmieszczenia czujników pomiarowych będącego zagadnieniem gwałtownie rozwijającej się dziedziny jaką jest technika optymalnego planowania eksperymentu. Zastosowanie wiedzy dotyczącej optymalnego planowania rozmieszczenia czujników, rozwijanej od początku lat 70-tych ubiegłego wieku wydaje się rozwiązywać wiele aktualnie rozważanych problemów związanych z zastosowaniem sieci sensorycznych. Zagadnienia optymalnego rozmieszczenia sensorów jak i planowania trajektorii ich poruszania, doboru optymalnej liczby czujników pomiarowych, problem optymalnej aktywacji tylko części z rozmieszczonych czujników, jak i wiele innych, należą do aktualnie rozpatrywanych problemów w dziedzinach związanych z sieciami sensorycznymi, a jednocześnie należą do klasy problemów szeroko rozpoznanych w zagadnieniach technik optymalnego planowania eksperymentu związanych z określaniem optymalnych położeń sensorów dla układów z czasoprzestrzenną dynamiką [117, 170, 183, 184, 187, 225, 234, 236, 239].

1.5 Optymalizacja rozmieszczenia czujników pomiarowych na potrzeby estymacji parametrów

W literaturze można spotkać dwa podejścia do zagadnienia optymalnego rozmieszczania czujników pomiarowych w zadaniach estymacji parametrycznej obiektów z czasoprzestrzenną dynamiką. Pierwszy z nurtów polega na sprowadzeniu zagadnienia identyfikacji parametrycznej do zadania estymacji stanu. Takie sformułowanie problemu jest stosunkowo szeroko omówione w literaturze [50, 113, 160]. Poprzez rozszerzenie wektora stanu o identyfikowane parametry zadanie estymacji staje się jednak silnie nieliniowe, co prowadzi do poważnych problemów zarówno natury teoretycznej (zadanie estymacji stanu dla układów z czasoprzestrzenną dynamiką jest dobrze zbadane jedynie dla układów liniowych), jak i obliczeniowej (oprócz problemów ze stabilnością i słabym uwarunkowaniem, istniejące techniki cechują się nadmiernymi wymaganiami pamięciowymi i czasowymi) co powoduje, że ich stosowalność ograniczała się do tej pory do zagadnień z jedną zmienną przestrzenną. Jako pewne remedium, w pracy [136] proponuje się zastosowanie ciągu linearyzacji wokół kolejnych trajektorii, a w [113] wykorzystanie suboptymalnego algorytmu filtru Kalmana. Sprowadzenie problemu rozmieszczania czujników do zagadnienia estymacji stanu powoduje, że problem formułuje się jako zadanie minimalizacji zagregowanego śladu macierzy kowariancji błędu estymacji przy ograniczeniach dynamicznych w postaci równań filtru Kalmana [113].

W niniejszej pracy wykorzystano podejście alternatywne, polegające na sformu-

łowaniu problemu w kategoriach teorii planowania eksperymentów optymalnych. W tym podejściu znalezienie optymalnego rozmieszczenia czujników odpowiada znalezieniu ekstremum funkcjonału określonego na Informacyjnej Macierzy Fishera (IMF) związanej z estymowanymi parametrami [4, 58, 150, 170, 172, 180, 188, 189, 223, 225, 228, 243]. Podstawą leżącą u podstaw takiego zdefiniowania problemu jest wyrażenie dokładności estymat parametrów poprzez macierz kowariancji tych estymat. Odwrotność macierzy informacyjnej okazuje się być przybliżeniem macierzy kowariancji identyfikowanych parametrów, w przypadku rozpatrywania długiego czasu obserwacji obiektu, gdy nieliniowość procesu ze względu na parametry jest niewielka oraz gdy błędy pomiarowe mają małe amplitudy [228, 243].

Pierwsze podejścia do problemu optymalnego rozmieszczenia czujników pomiarowych zostały zaproponowane w pracy [237], gdzie maksymalizowano kryterium D-optymalne, będące wyznacznikiem macierzy informacyjnej powiązanej z estymowanymi parametrami opisującymi źródło w prostym jednowymiarowym liniowym równaniu dyfuzji. Autorzy zaobserwowali, że liniowa zależność obserwowanych wyjść modelu od parametrów pozwala na bezpośrednie zastosowanie technik optymalnego planowania eksperymentu. Tak naszkicowane zagadnienie zostało rozszerzone w pracy [183], na potrzeby układów o parametrach rozłożonych opisywanych przez liniowe równania hiperboliczne ze znanymi funkcjami własnymi, a nieznanymi wartościami własnymi. Podobne zagadnienie było przedmiotem pracy [184] dla bardziej ogólnych postaci układów o parametrach rozłożonych, które mogą być opisane z zastosowaniem funkcji Greena.

Przez ostatnie dwie dekady, opracowana metodologia została znacznie rozszerzona w kierunku jej możliwych zastosowań praktycznych. Szeroki wachlarz zarówno metod wyznaczania optymalnych strategii rozmieszczenia czujników, jak i ich potencjalnych zastosowań został przedstawiony w monografii [234]. Ciekawe rozszerzenie tego podejścia przedstawiono w pracy [171], gdzie zaproponowano schemat detekcji uszkodzeń dla układów o parametrach rozłożonych bazujący na maksymalizacji wskaźnika opartego na mocy testu hipotezy parametrycznej odnoszącej się do nominalnego stanu układu.

Podejścia bazujące na maksymalizacji wyznacznika macierzy informacyjnej służą nie tylko rozważaniom teoretycznym ale istnieją również praktycznych zastosowaniach, w których pokazana została ich efektywność. W pracy [148] poszukiwano optymalnych położeń czujników stacjonarnych z wykorzystaniem technik programowania nieliniowego dla procesu fermentacji w pewnym systemie biotechnologicznym. Z drugiej strony, praca [206] przedstawia zastosowanie technik planowania w celu rozwiązania problemu identyfikacji w zagadnieniu badania podziemnych cieków wodnych. Również w przypadku tego zagadnienia, zaproponowano wykorzystanie technik programowania nieliniowego w celu określenia optymalnych położeń odwiertów.

Podobne podejście zostało przedstawione w pracach [100, 101] dla problemu identyfikacji modeli elastycznych struktur kosmicznych. Pomimo, że poszczególne modele nie były opisywane równaniami różniczkowymi cząstkowymi, a zostały przedstawio-

ne w postaci zdyskretyzowanej, po zastosowaniu metody elementów skończonych, zaproponowane podejścia są bardzo interesujące z punktu dużego podobieństwa obydwu sformułowań. Zaproponowano szybkie i efektywne podejście w celu redukcji relatywnie dużej liczby miejsc potencjalnego rozmieszczenia czujników do znacznie mniejszego zbioru zawierającego liniowo niezależne obiekty. Opracowane metody nie prowadzą również do znacznego zmniejszenia dokładności wyznaczanych estymat na podstawie wartości wyznacznika macierzy informacyjnej. Pewne ulepszenia przedstawionej metody z wykorzystaniem metody Tabu-Search zaproponowano w pracy [107].

W pracy [174] przedstawiono podobną metodę wyznaczania kryterium optymalności bazującą na maksymalizacji wyznacznika macierzy Grama, będącego miarą niezależności funkcji wrażliwości wyznaczanych w miejscach lokalizacji czujników. Autorzy wykazali, że takie podejście gwarantuje, że parametry są identyfikowalne, a korelacje pomiędzy pomiarami są zminimalizowane. Postać zaproponowanego kryterium przypomina kryterium D-optymalności, ale odpowiednik macierzy informacyjnej przyjmuje znacznie większe rozmiary, co sugeruje, że takie podejście może powodować konieczność wykonywania znacznie większej ilości obliczeń. Jednakże, przedstawiona metoda została wdrożona z sukcesem w laboratoryjnym reaktorze katalitycznym [240].

Warte odnotowania jest również to, że techniki planowania eksperymentów przestrzennych spokrewnione z problemem rozmieszczania czujników pomiarowych, są również domeną zainteresowania statystyków, a w literaturze można spotkać wiele pozycji rozważających powyższe zagadnienia [39, 147, 156, 157], które są motywowane praktycznymi zagadnieniami w takich dziedzinach jak rolnictwo, geologia, meteorologia czy też ekonomia. Jednakże, modele statystyczne rozważane w literaturze są zdefiniowane całkowicie inaczej niż modele w postaciach równań różniczkowych cząstkowych. Głównym zamierzeniem tych metod jest modelowanie procesów przestrzennych z wykorzystaniem teorii pól losowych. Zastosowanie pól losowych polega na wykorzystaniu informacji uzyskanych w poprzednich iteracjach do wyboru najlepszego zbioru punktów w celu najlepszej reprezentacji pola w aktualnym zagadnieniu. Motywacją jest potrzeba interpolacji obserwowanego zachowania procesu w położeniach przestrzennych, gdzie nie jest znany stan procesu, jak również poszukiwanie optymalnych miejsc obserwacji, które pozwalają na dokładniejsze obserwowanie procesu. Bardzo wartościowe informacje na temat optymalnego planowania z wykorzystaniem teorii pól losowych zawarto w monografii [147], która zawiera podstawowe informacje pozwalając na połączenie teorii statystyki przestrzennej z klasycznym planowaniem eksperymentu. Problem optymalnego planowania może być również sformułowany bazując na kryterium informacyjnym, w którym maksymalizowany jest wskaźnik oparty na miarze informacji (definiowanej jako miara Kullbacka-Leiblera) jaka może być uzyskana z eksperymentu pomiarowego [27, 28].

Do ciekawych zagadnień związanych z wykorzystaniem technik planowania jest problem dyskryminacji pomiędzy różnymi modelami systemów dynamicznych, który jest rozważany w pracy [118]. W pracy zaproponowano wykorzystanie ekspe-

rymentów T-optymalnych pozwalających na wybór modelu, który lepiej przybliża zachowanie się badanego procesu.

Rozwój techniki spowodował, że oprócz wykorzystania sieci sensorów stacjonarnych wykonujących pomiary ciągłe zaczęto prowadzić obserwacje procesów z wykorzystaniem sieci sensorów skanujących, które prowadzą obserwacje badanego procesu aktywnie zmieniając podzbiór czujników wykonujących pomiary. W konsekwencji, dzięki aktywnej rekonfiguracji sieci sensorycznej, informacja o identyfikowanych parametrach jest uwzględniana dynamicznie, co powoduje, że można oczekiwać uzyskania lepszej wartości kryterium optymalności w stosunku do pomiarów wykonywanych z zastosowaniem stacjonarnej sieci sensorycznej. Zastosowanie sieci sensorycznej składającej się z rekonfigurowanego zbioru aktywnych czujników stacjonarnych (sieć czujników skanujących) jak i z sieci sensorów mobilnych jest przedmiotem dużego zainteresowania wśród badaczy [29, 32, 138, 159, 201]. W artykule [187], rozważa się wykorzystanie miary dokładności identyfikacji w postaci kryterium D-optymalnego. Podejście jest bardzo zbliżone do metod klasycznej teorii planowania eksperymentu dla układów o parametrach skupionych, jednak zamiast poszukiwać samych trajektorii w tym przypadku poszukuje się pewnej miary przestrzennej zależnej od czasu, związanej z częstotliwością wykonywania pomiarów w czasie i przestrzeni. W pracach [170, 226, 228, 232] zagadnienie wyznaczania optymalnych trajektorii formułuje się jako zagadnienie sterowania optymalnego, dla którego rozwiązania są otrzymywane poprzez ciąg kolejnych linearyzacji problemu, co umożliwia uwzględnienie rozmaitych ograniczeń na ruch sensorów. Z kolei praca [230] została poświęcona próbie sformułowania i rozwiązania problemu czasowo-optymalnego poruszania się czujników obserwujących stan procesu o parametrach rozłożonych w celu estymacji jego parametrów.

Chociaż podejście w kategoriach teorii planowania eksperymentów optymalnych jest najszerzej stosowane, nie jest ono pozbawione wad. Analiza nawet stosunkowo prostych przykładów [182] pozwala zaobserwować zjawisko zależności optymalnych warunków eksperymentu od wartości estymowanych parametrów. Zjawisko to jest powszechnie znane w teorii planowania eksperymentu dla nieliniowych modeli regresji [4, 38, 56, 58, 106, 172, 180, 200]. W konsekwencji, planowanie eksperymentu mającego na celu estymację parametrów często poprzedza się eksperymentem wstępnym lub analizą fizyczną umożliwiającymi zgrubną ocenę wartości parametrów [225, 243]. Jako rozwiązanie stosuje się czasem podejście oparte na tzw. planowaniu sekwencyjnym [65, 183, 184, 185], gdzie etapy planowania, zbierania danych pomiarowych i estymacji parametrów wykonywane są naprzemiennie. Niestety, takie podejście jest rzadko spotykane w praktycznych realizacjach, przede wszystkim ze względu na koszty oraz ograniczenia czasowe eksperymentu.

Jednym ze sposobów uniezależnienia optymalnych położeń czujników od wartości identyfikowanych parametrów jest stosowanie tzw. planowania odpornego, które pozwala określić „możliwie najlepsze" położenia czujników dla danego zakresu parametrów. Planowanie odporne optymalnych położeń czujników w sensie uśrednionym [170, 228, 243] opiera się na probabilistycznym opisie niepewności wstępnej informa-

cji o prawdziwej wartości wektora parametrów, co jest równoznaczne z przyjęciem pewnego rozkładu prawdopodobieństwa *a priori*, którego postać można wydedukować np. na podstawie obserwacji zebranych w zbliżonych warunkach dla podobnych procesów. Wprowadzenie rozkładu prawdopodobieństwa pozwala pozbyć się zależności macierzy informacyjnej od wektora nieznanych parametrów poprzez ograniczenie się do ekstremalizacji wartości oczekiwanej oryginalnie zdefiniowanego lokalnego kryterium optymalności.

Złożoność zagadnienia planowania odpornego wyklucza analityczne wyznaczenie planów optymalnych w sensie uśrednionym, a to implikuje z kolei potrzebę odwołania się do technik numerycznych. Wykorzystanie klasycznych technik optymalizacji jest mocno utrudnione koniecznością wyznaczenia wartości oczekiwanej (całki wielowymiarowej) kryterium lokalnego, co jest w praktyce możliwe jedynie w przypadku dyskretnego rozkładu prawdopodobieństwa. Rozwiązaniem tego problemu może być zastosowanie metod aproksymacji stochastycznej [179, 241, 243], gdyż w tym przypadku do rozwiązania dochodzi się iteracyjnie bez potrzeby określania wartości minimalizowanego funkcjonału. Inne podejście zostało przedstawione w pracach [170, 228], gdzie zaproponowano zdefiniowanie problemu jako zagadnienie minimaksowe i rozwiązanie go z wykorzystaniem programowanie półnieskończonego. Mimo otrzymania wielu wartościowych rezultatów, olbrzymia złożoność zagadnienia implikuje dalsze istnienie wielu otwartych problemów badawczych, przede wszystkim w kontekście ogólnej teorii planowania eksperymentów optymalnych, stanowiących zarówno interesujące wyzwanie natury naukowej, jak i ważne zadania z punktu widzenia potencjalnych zastosowań.

Innym problemem jest zjawisko skupiania się optymalnych położeń grup czujników pomiarowych w jednym punkcie, co określa się jako klasteryzację czujników [170, 183, 228]. Takie zjawisko jest spowodowane założeniem o wzajemnej niezależności pomiarów wykonywanych przez różne czujniki. Jak remedium proponuje się uwzględnienie w sformułowaniu zadania optymalizacji ograniczenia na dopuszczalne minimalne odległości między czujnikami [228].

Metody rozpatrywane w literaturze w większości przypadków nie dają się zastosować bezpośrednio w praktyce. Zastosowania inżynierskie wymagają rozpatrzenia problemów znacznie bardziej ogólnych, np. uwzględniania oprócz szumu pomiarowego również szumu procesowego, możliwość wzięcia pod uwagę zakłóceń innych niż gaussowskie lub uwzględnienia rozmaitych ograniczeń na miejsca rozmieszczenia czujników. Przedstawione w literaturze metody rozwiązywania problemu optymalnego rozmieszczania czujników pomiarowych prowadzą do bardzo skomplikowanych problemów optymalizacji nieliniowej, zaawansowanych zagadnień rachunku prawdopodobieństwa i statystyki matematycznej. Ich rozwiązywanie wiąże się z bardzo dużym nakładem obliczeniowym, a w związku z tym z zapotrzebowaniem na moc obliczeniową. Dodatkowo zastosowanie omówionych metod w badaniach rzeczywistych zjawisk i obiektów powoduje konieczność uwzględniania bardziej skomplikowanej struktury szumu pomiarowego oraz procesowego niż szum biały. Należy również zadbać o to, aby ograniczyć zjawisko klasteryzacji oraz uwzględnić pojawianie się

korelacji pomiędzy pomiarami wykonywanymi przez czujniki znajdujące się w bliskiej odległości [170, 183, 225, 228]. Dotychczas proponowane metody wywodzące się z klasycznego planowania eksperymentu często operują na tzw. uogólnionym planie eksperymentu, w którym miejsce wykonywania pomiaru (punkt planu) jest dodatkowo charakteryzowane liczbą dodatnią, którą można interpretować jako częstość wykonywania pomiarów w tym punkcie w wielokrotnej serii powtórzeń eksperymentu [170, 189, 228]. W rzeczywistych warunkach, wielokrotne wykonywanie tego samego eksperymentu w niezmienionych warunkach jest praktycznie niemożliwe do zrealizowania. Metody wykorzystywane w zastosowaniach inżynierskich powinny umożliwiać uzyskiwanie rezultatów w jak najkrótszym czasie, bez wykonywania zbędnych powtórzeń. Zresztą konkretne zastosowania często wykluczają możliwość replikacji. Zastosowanie tzw. bezreplikacyjnych planów eksperymentu pozwala na wyeliminowanie konieczności powtarzania eksperymentu w celu wyznaczenia dokładnych lokalizacji czujników [170, 228]. Do rozwiązania problemu optymalnego rozmieszczania czujników pomiarowych w kategoriach bezreplikacyjnych planów eksperymentu można wykorzystać metaheurystyczne algorytmy dyskretne [235], pozwalające na efektywne przeszukiwanie przestrzeni możliwych konfiguracji czujników.

Próba uwzględnienia przedstawionych problemów związanych z optymalnym rozmieszczeniem czujników pomiarowych (skupianie się czujników, korelacje pomiędzy pomiarami) wymaga zastosowania i opracowania nowych metod badawczych oraz algorytmów. Dodatkowo wymagania dotyczące coraz większej dokładności i precyzji obliczeń, badań na coraz większym obszarze, zwiększanie liczby urządzeń biorących udział w badaniach powodują, że obliczenia stają się coraz trudniejsze, ale i przede wszystkim bardziej pracochłonne. Łatwa dostępność oraz niska cena urządzeń mogących realizować obserwacje procesów z czasoprzestrzenną dynamiką powoduje, że dodatkowym zagadnieniem staje się wybór optymalnej liczby węzłów sieci sensorycznej, która ma brać udział w obserwacji. Zbyt duża liczba urządzeń może powodować zakłócenia w transmisji danych pomiędzy urządzeniami, a przy zwiększonych kosztach, niekoniecznie musi pozwalać na uzyskanie lepszych rezultatów badań. Dodatkowo, coraz bardziej popularne stają się metody obserwacji, dotychczas rzadko rozpatrywane, tj. obserwacje z wykorzystaniem sieci czujników skanujących czy też zastosowanie mobilnych węzłów sieci sensorycznych. Wprowadzenie tych metod powoduje znaczne zwiększenie stopnia skomplikowania problemu optymalnej obserwacji układów z czasoprzestrzenną dynamiką. Implementacja algorytmów planowania wymaga zastosowania coraz to większych mocy obliczeniowych w celu otrzymania wyników w rozsądnym czasie. Dotychczas podjęto niewiele prób rozwiązania tych problemów z wykorzystaniem technik programowania równoległego [13, 14, 251], których stosowanie wydaje się rozwiązywać problemy związane z coraz to większym zapotrzebowaniem na moc obliczeniową. Można spodziewać się, że w przyszłości ten nurt będzie nabierał coraz większego znaczenia.

1.6 Obliczenia równoległe

Korzystanie z coraz bardziej zaawansowanych modeli matematycznych oraz wymagania coraz większej dokładności powodują, że obliczenia mające na celu wyznaczenie optymalnych strategii pomiarowych stają się coraz bardziej czasochłonne i pochłaniają coraz więcej zasobów maszyn, na których prowadzone są badania. Dodatkowo potrzeba uzyskiwania wyników badań w coraz krótszym czasie powoduje, że konieczne jest określenie i opracowanie nowych metod pozwalających sprostać wymaganiom nowoczesnych badań naukowych.

Rozwiązaniem przedstawionych problemów jest możliwość wykorzystania nowoczesnych środowisk obliczeniowych opartych o systemy wieloprocesorowe. Gwałtowny rozwój technik komputerowych, a zwłaszcza wprowadzenie środowisk klastrowych oraz gridowych, stwarza nadzieję na przynajmniej częściowe rozwiązanie problemów wynikających z istotnej złożoności obliczeniowej stosowanych algorytmów. Wykorzystanie algorytmów przetwarzania równoległego pozwala mieć nadzieję na znaczne skrócenie czasu obliczeń przy jednoczesnym zwiększeniu dokładności coraz bardziej skomplikowanych algorytmów optymalnego planowania rozmieszczenia czujników, zwłaszcza skanujących i ruchomych.

Dynamiczny wzrost mocy obliczeniowej uzyskiwanej przez mikroprocesory komputerów osobistych spowodował, że komputery klasy PC zaczęto łączyć przy pomocy sieci komputerowych w wyspecjalizowane środowiska obliczeniowe typu *HPC* (ang. *High Performance Computing*) [25]. Jedną z pierwszych instalacji tego typu był klaster *Beowulf* [217]. Zastosowanie popularnych jednostek obliczeniowych w postaci komputerów klasy PC, połączenie ich za pomocą ogólnodostępnych sieci typu Ethernet oraz wykorzystanie środowisk typu *PVM* [70] lub bibliotek implementujących standard *MPI* [67, 202] spowodowało znaczą popularyzację środowisk programowania równoległego i ich wykorzystanie w placówkach naukowo-badawczych na całym świecie. Początkowo zasoby te miały charakter rozwiązań tzw. NOW (Network of Workstations) i składały się z elementów budujących klasyczne systemy klasy PC, jednak wraz z upływem czasu również te środowiska obliczeniowe przeszły specjalizację i przekształciły się w klastry obliczeniowe. W klastrach obliczeniowych wykorzystuje się bardziej specjalizowane komponenty (np. dedykowaną wysocewydajną sieć łączącą węzły klastra Myrinet [149], Infiniband [96], SCI/Dolphin [218]), jednak podstawowa zaleta tego typu środowisk obliczeniowych, a mianowicie stosunek mocy obliczeniowej do ceny, pozostaje nadal na bardzo korzystnym wysokim poziomie.

Aktualnie pierwsze miejsce na liście TOP500 [219] zajmuje superkomputer Roadrunner zlokalizowany w Narodowym Laboratorium w Los Alamos. Jest to superkomputer zbudowany w architekturze klastra, w którym jako jednostki obliczeniowe wykorzystano procesory firmy IBM PowerXCell 8i oraz AMD Opteron Dual Core. Do wymiany danych pomiędzy węzłami klastra wykorzystano specjalizowaną sieć firmy Infiniband [96]. Maszyna działa w oparciu o system operacyjny Linux, i osiąga wydajność 1.1 PFLOPów, co czyni ją aktualnie najszybszym superkomputerem na świecie. Drugie miejsce, z wydajnością niewiele mniejszą, bo 1.05 PFLOPów,

zajmuje superkomputer firmy Cray zbudowany w architekturze MPP w oparciu o procesory AMD Opteron Quad Core. Do swojej pracy wykorzystuje system operacyjny CNL, a jako wewnętrzną magistralę komunikacyjną zastosowano rozwiązanie XT4 Internal Interconnect. Rozwój superkomputerów jest bardzo gwałtowny, a aktualnie budowany przez firmę IBM superkomputer w Lawrence Livermore National Laboratory (USA, Kalifornia) ma osiągnąć w 2012 roku wydajność 20 PFLOPów.

Ciągły spadek cen wykorzystywanych komponentów oraz ich szeroka dostępność powodują, że systemy komputerowe bardzo często są modernizowane. Częste modyfikacje platform sprzętowych zasobów wykonujących obliczenia powodują, że w szerszym spojrzeniu takie systemy coraz rzadziej są systemami jednorodnymi, zawierającymi jednostki obliczeniowe o podobnej mocy obliczeniowej. Ważnym elementem podczas syntezy algorytmów równoległych staje się optymalne wykorzystanie mocy obliczeniowych przydzielonych jednostek obliczeniowych. Problem równoważenia obciążenia (*ang.* load balancing) został już dawno dostrzeżony i jest szeroko omawiany w literaturze [42, 121]. Niejednorodność systemu obliczeniowego może być rozpatrywana zarówno względem posiadania przez procesory różnych mocy obliczeniowych, jak i pod względem dynamicznych zmian obciążenia tych procesorów w czasie pracy uruchomionego zadania [69]. Równoważenie obciążenia pozwala na określenie obciążenia niejednorodnego systemu obliczeniowego w sposób zapewniający uzyskanie jak największej wydajności uruchomionego w nim zadania.

W literaturze można spotkać wiele różnych podejść do zagadnienia równoważenia obciążenia: rozważane są algorytmy scentralizowane oraz rozproszone [46], globalne i lokalne [250], adaptujące wielkość zadania do wykonania przez procesory lub sterujące wielkością kolejki zadań dla poszczególnych procesorów [193]. W przypadku algorytmów scentralizowanych, sposób dystrybucji zadań pomiędzy procesory jest ustalany na jednym węźle głównym, w odróżnieniu od rozproszonego algorytmu równoważenia obciążenia, gdzie każdy procesor osobno oblicza sposób dystrybucji zadań pomiędzy procesory. Zastosowanie algorytmu globalnego powoduje, że przydział zadań procesorom wykonywany jest na podstawie globalnego stanu systemu, a w przypadku algorytmów lokalnych równoważenie obciążenia realizowane jest na podstawie danych zgromadzonych przez poszczególne procesory, bądź zbiory procesorów. Ważna jest również informacja na temat architektury połączeń pomiędzy procesorami (np. hiper-kostka, siatka) [121]. Architektura połączeń determinuje sposób projektowania algorytmów rozwiązujących rozważane zagadnienia. Przykładowo, w przypadku hiper-kostki o n wymiarach, każdy z procesorów może się komunikować bezpośrednio tylko z n sąsiadami, co znacząco wpływa na sposób przesyłania danych pomiędzy poszczególnymi procesorami.

Wykorzystanie zoptymalizowanych metod przydziału mocy obliczeniowych poszczególnym częściom wykonywanego algorytmu pozwala na znaczące przyspieszenie obliczeń. Zastosowanie metod równoważenia obciążenia wykorzystywanej mocy obliczeniowej umożliwia efektywne przydzielanie zasobów maszyn na których przeprowadzane są obliczenia i symulacje, a dzięki temu — szybsze uzyskanie wyników.

1.7 Zarys proponowanego podejścia

Optymalne planowanie rozmieszczenia czujników pomiarowych wymaga dokładnej analizy i rozpoznania procesu jaki podlega obserwacji. Wymagania stawiane modelom pozwalającym na późniejszą predykcję zachowania się procesu z czasoprzestrzenną dynamiką, powodują konieczność dokładnej identyfikacji parametrów występujących w modelu. Sformułowanie zagadnienia optymalnego planowania rozmieszczenia czujników jako zagadnienie optymalnego planowania eksperymentu pozwala na wykorzystanie potężnego, już istniejącego aparatu matematycznego związanego z optymalizacją globalną. Uzyskuje się to dzięki sprowadzeniu oryginalnego problemu do zadania poszukiwania ekstremum pewnej skalarnej miary określonej na informacyjnej macierzy Fishera. Teoretycznie pozwala na zastosowanie całej gamy algorytmów programowania nieliniowego.

Jednakże pewne specyficzne właściwości tak zdefiniowanego problemu powodują, że bezpośrednie zastosowanie istniejących algorytmów optymalizacji globalnej nie jest oczywiste. Wysoka nieliniowość problemu, znaczna wymiarowość przestrzeni poszukiwania, występowanie problemów związanych ze skupianiem się czujników, występowaniem korelacji pomiędzy obserwacjami czy też konieczność opracowywania bezreplikacyjnych planów eksperymentu powodują, że problem nie jest trywialny i należy analizować go w sposób całościowy: od momentu tworzenia modelu analizowanego zjawiska, poprzez odpowiedni wybór technik obliczeniowych pozwalających na komputerową implementację modelu, do wyboru algorytmu pozwalającego na ekstremalizację kryterium optymalności określonego na macierzy informacyjnej. Dekompozycja problemu na części składowe i rozwiązywanie ich niezależnie może nie dać zadowalających wyników, a w konsekwencji prowadzić do błędnie zaplanowanego eksperymentu.

Dodatkowym aspektem rozważanych zagadnień jest możliwość wykorzystania najnowszych typów obserwacji, możliwych dzięki rozwojowi sieci sensorycznych. Obserwacje procesów z czasoprzestrzenną dynamiką nie muszą już być realizowane tylko za pomocą czujników stacjonarnych, ale również najnowsze zdobycze techniki umożliwiają łatwą realizację obserwacji z wykorzystaniem skanujących sieci sensorycznych czy też sieci sensorycznych składających się z mobilnych węzłów przenoszących czujniki. Zwłaszcza te ostatnie przeżywają swój gwałtowny rozwój spowodowany zarówno upowszechnieniem się urządzeń, jak i znaczącym zmniejszeniem się kosztów ich zastosowań w badaniach. Łatwa dostępność tych urządzeń powoduje nieodpartą chęć zastosowania jak największej liczby urządzeń w celu obserwacji badanych procesów. Jednak potencjalne zyski związane z zastosowaniem nowych technologii związanych z sieciami sensorycznymi okupione są niestety znacznym skomplikowaniem opisu problemu optymalnej obserwacji. Chęć stosowania dużej liczby urządzeń, ale jednocześnie skomplikowany opis obserwacji prowadzonej już przez jeden czujnik powodują, że zagadnienie wykorzystania dużych sieci czujników w procesie optymalnej obserwacji układów z czasoprzestrzenną dynamiką staje się wybitnie nietrywialne.

Złożoność obliczeniowa poszczególnych etapów optymalnego planowania ekspe-

rymentu powoduje, że racjonalnym wydaje się zastosowanie technik programowania równoległego. Jednakże zastosowanie wprost np. równoległych algorytmów optymalizacji globalnej nie musi prowadzić do uzyskania zadowalającego rezultatu w postaci przyspieszenia obliczeń. Wykorzystanie obliczeń równoległych prowadzi do pojawienia się dodatkowych wątków związanych z rozpraszaniem obliczeń. Wykorzystanie zasobów sprzętowych wymaga, aby algorytmy miały np. możliwość adaptacji do zmieniających się warunków pracy w heterogenicznych środowiskach obliczeniowych. Niniejsza praca stanowi próbę odpowiedzi na występowanie rozważanych problemów, począwszy od dokładnej analizy problemu optymalnej aktywacji czujników (stacjonarnych, skanujących lub ruchomych), po zaproponowanie metod obliczeniowych pozwalających na optymalne wykorzystanie nowoczesnych równoległych środowisk obliczeniowych w celu przyspieszenia obliczeń.

1.8 Teza i cele pracy

Rozpoczęcie badań objętych niniejszą rozprawą było poprzedzone dokładną analizą istniejących algorytmów optymalnego planowania eksperymentu. Metody i algorytmy proponowane w literaturze [170, 225, 228] przedstawiają podejścia do problemu w sposób sekwencyjny, których zrównoleglenie nie jest bynajmniej oczywiste. Przeanalizowano również nieliczne pozycje przedstawiające koncepcje zrównoleglania algorytmów optymalnego planowania eksperymentu [12, 13, 14, 119, 251]. Analiza doprowadziła do wniosku, że możliwym jest udoskonalenie istniejących algorytmów sekwencyjnych, ich adaptacja do środowisk równoległych oraz poprawa wydajności istniejących algorytmów równoległych w heterogenicznych środowiskach klastrowych. W konsekwencji, pozwoliło to na sformułowanie następującej tezy:

> ***Fuzja algorytmów obliczeniowych optymalizacji ciągłej i dyskretnej, teorii planowania eksperymentu oraz technik przetwarzania równoległego pozwala na efektywne wyznaczanie optymalnych konfiguracji dużych sieci czujników w procesie obserwacji złożonych układów z czasoprzestrzenną dynamiką.***

Przyjęta metodologia badań, wykorzystująca wyniki współczesnej teorii metod obliczeniowych optymalizacji, a w szczególności teorii planowania eksperymentu, przetwarzania równoległego oraz numerycznego rozwiązywania równań różniczkowych cząstkowych, implikowała koncentrację badań wokół następujących problemów oraz postawienie następujących celów badawczych:

- udoskonalenie istniejących i konstrukcja nowych algorytmów optymalnej obserwacji na potrzeby estymacji parametrów procesów opisywanych równaniami różniczkowymi cząstkowymi,
- opracowanie efektywnych algorytmów planowania eksperymentu dla skorelowanych obserwacji,

- opracowanie efektywnych metod obserwacji procesów z czasoprzestrzenną dynamiką z zastosowaniem dużych sieci sensorycznych,
- opracowanie efektywnych metod optymalnej aktywacji sieci czujników stacjonarnych oraz skanujących z wykorzystaniem algorytmów optymalizacji dyskretnej,
- opracowanie efektywnych metod optymalnej parametryzacji trajektorii mobilnych węzłów sieci sensorycznej z zastosowaniem algorytmów optymalizacji ciągłej,
- opracowanie efektywnych metod wyznaczania optymalnych trajektorii mobilnych węzłów sieci sensorycznej z uwzględnieniem dynamiki poruszających się węzłów przy zastosowaniu algorytmów optymalizacji ciągłej,
- wykorzystanie w opracowanych algorytmach równoległych metod równoważenia obciążenia, które pozwalają na efektywne wykorzystanie całkowitej mocy obliczeniowej istniejących heterogenicznych środowisk obliczeniowych,
- opracowanie na potrzeby przedstawionych metod numerycznych, bazy programów oraz bibliotek wspierających wyznaczanie optymalnych obserwacji układów z czasoprzestrzenną dynamiką z zastosowaniem:
  - kompilatorów języka Fortran 95 wraz z biblioteką procedur numerycznych Intel®MKL, biblioteką do przetwarzanie równoległego MPI w środowiskach obliczeniowych,
  - środowisk MATLAB®, MAPLE® oraz COMSOL®.

1.9 Przegląd treści rozprawy

Praca składa się z sześciu rozdziałów, w tym wstępu, czterech rozdziałów zasadniczych oraz zakończenia.

Rozdział pierwszy przedstawia wprowadzenie w zagadnienia optymalnej obserwacji układów z czasoprzestrzenną dynamiką oraz przedstawia wynikające stąd problemy obliczeniowe. Sformułowano cel i tezę rozprawy oraz zamieszczono przegląd treści rozprawy.

W rozdziale drugim sformułowano problem rozmieszczania czujników pomiarowych w kategoriach teorii planowania eksperymentu. Omówiono zagadnienia związane z optymalną obserwacją układów z czasoprzestrzenną dynamiką wraz z ich specyfiką. Przedstawiono istniejące podejścia do rozwiązania zarysowanego problemu planowania oraz ich własności.

Rozdział trzeci zawiera opis i charakterystykę równoległych środowisk obliczeniowych. Przedstawiono architektury sprzętowe oraz typy sieci komputerowych stosowanych w klastrach obliczeniowych. Omówiono sposoby zrównoleglania algorytmów oraz scharakteryzowano heterogeniczne środowiska obliczeniowe.

W rozdziale czwartym przedstawiono rezultaty badań związanych z optymalną aktywacją czujników stacjonarnych oraz skanujących. Zaproponowano algorytmy bazujące na metaheurystycznych algorytmach optymalizacji dyskretnej — GRASP oraz Tabu-Search, a także przedstawiono sposoby ich zrównoleglania. Dodatkowo przedstawiono metodę redukcji czasochłonnych operacji związanych z wyznaczaniem macierzy informacyjnej.

Rozdział piąty poświęcono zagadnieniom związanym z optymalnym planowaniem trajektorii czujników ruchomych. Zaproponowano dwa podejścia do rozwiązania rozważanych problemów: jedno opierające się na parametryzacji trajektorii z wykorzystaniem funkcji sklejanych i zastosowaniem algorytmu tunelowego w celu wyznaczenia trajektorii optymalnych, oraz drugie, umożliwiające uwzględnienie dynamiki mobilnych węzłów sieci sensorycznej poprzez sprowadzenie problemu do zagadnienia sterowania optymalnego i rozwiązania go z zastosowaniem iteracyjnego programowania dynamicznego. Dla obydwu podejść przedstawiono sposoby ich zrównoleglenia w celu skrócenia czasu wykonywania obliczeń.

W rozdziale szóstym dokonano podsumowania rezultatów uzyskanych w niniejszej rozprawie. Ponadto wskazano otwarte problemy oraz kierunki dalszych badań.

2

Rozmieszczenie czujników pomiarowych jako zadanie optymalnego planowania eksperymentu

W rozdziale omawia się zagadnienie planowania eksperymentu optymalnego w kontekście obserwacji układów z czasoprzestrzenną dynamiką z zastosowaniem sieci czujników (sensorów). Wyjaśniona zostanie jego specyfika oraz przedstawione zostaną cele jakim mają służyć badania objęte niniejszą rozprawą.

2.1 Sformułowanie problemu

2.1.1 Model procesu

Procesy z czasoprzestrzenną dynamiką opisuje się zazwyczaj za pomocą liniowych bądź nieliniowych równań różniczkowych cząstkowych. W pracy zakłada się, że zjawisko opisuje równanie różniczkowe cząstkowe następującej postaci [228]:

$$\frac{\partial s(x,t)}{\partial t} = \mathcal{H}\left(x, t, s, \frac{\partial s}{\partial x_1}, \frac{\partial s}{\partial x_2}, \frac{\partial^2 s}{\partial x_1^2}, \frac{\partial^2 s}{\partial x_2^2}; \bar{\theta}\right), \quad (x,t) \in \Omega \times T \tag{2.1}$$

gdzie:

$\Omega \in \mathbb{R}^2$ — spójny i ograniczony obszar dwuwymiarowy o gładkim brzegu $\partial\Omega$,
t — czas,
$T = (0, t_f)$ — ustalony horyzont obserwacji ($t_f < \infty$),
$x = (x_1, x_2) \in \mathbb{R}^2$ — wektor zmiennych przestrzennych należący do $\bar{\Omega} = \Omega \cup \partial\Omega$,
$s = s(x,t)$ — zmienna stanu,

$\mathcal{H}$ — pewna znana funkcja mogąca zawierać człony związane ze znanymi wymuszeniami deterministycznymi.

Równanie (2.1) uzupełnia się warunkiem początkowym

$$s(x,0) = s_0(x), \quad x \in \Omega, \tag{2.2}$$

oraz brzegowym

$$\mathcal{E}\left(x, t, s, \frac{\partial s}{\partial x_1}, \frac{\partial s}{\partial x_2}; \bar{\theta}\right) = 0, \quad x \in \partial\Omega,\ t \in (0, t_f], \tag{2.3}$$

gdzie: $s_0 = s_0(x)$, $\mathcal{E}$ — pewna znana funkcja.

Jak można zauważyć, w równaniach (2.1)–(2.3) występuje wektor parametrów $\bar{\theta}$ należący do zbioru parametrów dopuszczalnych Θ. W przedstawionych badaniach zakłada się, że

$$\bar{\theta} = (\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_m) \in \Theta \subset \mathbb{R}^m, \tag{2.4}$$

czyli wektor $\bar{\theta}$ pozostaje stały przez cały okres obserwacji rozważanego procesu, jednak dokładne wartości jego elementów są nieznane.

Głównym zadaniem estymacji parametrycznej jest próba oceny prawdziwych wartości elementów wektora $\bar{\theta}$ na podstawie znajomości modelu (2.1)–(2.3), który odzwierciedla zachowanie rzeczywistego procesu. Z uwagi na to, że wektor parametrów $\bar{\theta}$ zazwyczaj nie jest bezpośrednio mierzalny (np. współczynnik dyfuzji w przypadku modelu rozprzestrzeniania się zanieczyszczeń w atmosferze), należy dokonać jego identyfikacji na podstawie danych pomiarowych rejestrowanych przez zestaw dostępnych czujników pomiarowych rozmieszczonych w rozważanym obszarze przestrzennym. Ponieważ dokładność estymacji parametrów zależy od lokalizacji czujników [186, 228], głównym zadaniem jest dobór miejsc obserwacji układu z czasoprzestrzenną dynamiką w taki sposób, aby otrzymać możliwie najdokładniejszą ocenę wektora $\bar{\theta}$.

Techniki wykonywania pomiarów można podzielić na cztery klasy [31]:

- obserwacje wykonywane przez czujniki stacjonarne w skończonej liczbie chwil czasowych,
- obserwacje wykonywane przez czujniki stacjonarne w czasie ciągłym,
- obserwacje wykonywane przez czujniki ruchome w skończonej liczbie chwil czasowych,
- obserwacje wykonywane przez czujniki ruchome w czasie ciągłym.

Z punktu widzenia stopnia skomplikowania problemu optymalnej obserwacji, dwie ostatnie klasy należą do najbardziej złożonych obliczeniowo i wymagają znacznie większej mocy obliczeniowej od pozostałych.

Na potrzeby niniejszej rozprawy, w której zasadniczo zakłada się, że pomiary wykonuje się w czasie ciągłym, wprowadzono następującą klasyfikację strategii pomiarowych:

- **Pomiary dokonywane przez czujniki stacjonarne**.
  Obserwacje ciągłe lub dyskretne są wykonywane w ustalonych miejscach w przestrzeni.

- **Pomiary dokonywane przez czujniki ruchome**.
  Pozwalają one na zwiększenie jakości uzyskiwanych planów optymalnych. Pomiary dokonywane przez czujniki stacjonarne nie zawsze dają bowiem możliwość skutecznego śledzenia czasoprzestrzennej dynamiki procesu, podczas gdy czujniki ruchome mogą podążać za punktami niosącymi w danej chwili najwięcej informacji o estymowanych parametrach. W zależności od zastosowanego podejścia, najczęściej należy również uwzględniać ograniczenia na trajektorie poruszających się czujników.

- **Pomiary dokonywane przez czujniki skanujące**.
  Ta strategia obserwacji odpowiada najczęściej sytuacji, gdy w rozważanym obszarze rozmieszczono pewną liczbę czujników stacjonarnych, z których w danej chwili czasowej aktywuje się tylko pewną część. Powodem ograniczenia w pobieraniu informacji jednocześnie ze wszystkich czujników może być np. oszczędność energii źródeł zasilania czujników. Ten tryb pracy jest charakterystyczny dla coraz bardziej rozpowszechniających się sieci sensorycznych. Zauważmy, że tę strategię obserwacji można również interpretować w kategoriach grupy czujników wykonujących co prawda pomiary stacjonarne, mogących jednak przemieszczać się w rozważanym obszarze przestrzennym w czasie pomijalnie małym względem długości horyzontu obserwacji.

2.1.2 Zagadnienie estymacji parametrycznej

Jak już wspomniano, stan układu opisanego równaniami (2.1)–(2.3) zależy od wektora parametrów $\bar{\theta} \subset \mathbb{R}^m$, który należy ocenić na podstawie pomiarów wykonywanych przez n czujników. Dla uproszczenia załóżmy również, że dokonuje się bezpośrednich pomiarów zmiennej stanu, a niedokładność pomiaru uwzględnia się jako zakłócenia addytywne. Ważnym założeniem przyjętym w poniższych rozważaniach jest brak korelacji przestrzennych oraz czasowych pomiędzy poszczególnymi obserwacjami. Zagadnienie pomiarów skorelowanych zostanie szerzej omówione w rozdziale 2.2.1.

Najczęściej spotyka się następujące postacie równania obserwacji, odpowiadające przedstawionym wyżej technikom wykonywania pomiarów [186, 225, 228]:

- czujniki stacjonarne, pomiar w skończonej liczbie chwil $t_1, \ldots, t_K$

$$y_k^j = s(x^j, t_k; \bar{\theta}) + \varepsilon(x^j, t_k), \quad j = 1, \ldots, n, \quad k = 1, \ldots, K, \tag{2.5}$$

- czujniki stacjonarne, pomiar ciągły w czasie

$$y^j(t) = s(x^j, t; \bar{\theta}) + \varepsilon(x^j, t), \quad j = 1, \ldots, n, \quad t \in [0, t_f], \tag{2.6}$$

- czujniki ruchome, pomiar ciągły w czasie

$$y^j(t) = s(x^j(t), t; \bar{\theta}) + \varepsilon(x^j(t), t), \quad j = 1, \ldots, n, \quad t \in [0, t_f], \tag{2.7}$$

gdzie:
x^j — położenie j-ego czujnika,
$\varepsilon = \varepsilon(x, t)$ — szum pomiarowy.

Zapis $s(x, t; \theta)$ podkreśla, że stan układu zależy od wektora parametrów θ. Zakłada się, że szum pomiarowy $\varepsilon(x, t)$ jest czasowo i przestrzenie nieskorelowanym gaussowskim szumem białym o charakterystyce [225, 228]:

$$E\{\varepsilon(x^j, t)\} = 0, \quad E\{\varepsilon(x^i, t)\varepsilon(x^j, \tau)\} = \sigma^2 \delta_{ij} \delta(t - \tau), \tag{2.8}$$

gdzie: $\sigma > 0$ — odchylenie standardowe szumu pomiarowego, δ_{ij} i $\delta(\cdot)$ — symbole odpowiednio delt Kroneckera i Diraca.

Przy powyższych założeniach, problem oceny wektora nieznanych parametrów $\bar{\theta}$ najczęściej formułuje się jako minimalizację odpowiednich kryteriów najmniejszych kwadratów, co prowadzi do ocen (estymat) $\hat{\theta}$ wektora $\bar{\theta}$ [10]:

- czujniki stacjonarne, pomiar w skończonej liczbie chwil $t_1, \ldots, t_K$

$$\hat{\theta} = \arg \min_{\theta \in \Theta} \sum_{j=1}^{n} \sum_{k=1}^{K} \{y_k^j - \hat{s}(x^j, t_k; \theta)\}^2, \tag{2.9}$$

- czujniki stacjonarne, pomiar ciągły w czasie

$$\hat{\theta} = \arg \min_{\theta \in \Theta} \sum_{j=1}^{n} \int_0^{t_f} \{y^j(t) - \hat{s}(x^j, t; \theta)\}^2 \, \mathrm{d}t, \tag{2.10}$$

- czujniki ruchome, pomiar ciągły w czasie

$$\hat{\theta} = \arg \min_{\theta \in \Theta} \sum_{j=1}^{n} \int_0^{t_f} \{y^j(t) - \hat{s}(x^j(t), t; \theta)\}^2 \, \mathrm{d}t, \tag{2.11}$$

gdzie: $\hat{s}(x, t; \theta)$ — rozwiązanie równań (2.1)–(2.3) odpowiadające danej wartości wektora θ.

Szczegółowe rozważania dotyczące zalet i wad stosowania kryterium najmniejszych kwadratów w rozważanych zagadnieniach zawarto w monografiach [225, 228].

2.1.3 Problem optymalnego rozmieszczenia czujników

Można zauważyć, że estymata $\hat{\theta}$ zależy od położeń czujników x^j. Fakt ten uzasadnia sensowność postawienia problemu doboru położeń czujników, które maksymalizowałyby dokładność otrzymywanych estymat parametrów. Formalizując to jako miarę dokładności estymacji, przyjmuje się najczęściej pewną funkcję rzeczywistą określoną na tzw. Informacyjnej Macierzy Fishera (IMF) [182, 183, 184, 187, 206]. Przy dostatecznie długim horyzoncie obserwacji odwrotność IMF jest asymptotyczną macierzą kowariancji estymowanych parametrów [105, 242, 243] stanowiąc statystyczną miarę rozproszenia estymat wokół wektora prawdziwych wartości parametrów (oczywiście, ta cecha wymaga spełnienia — przynajmniej w przybliżeniu — warunku nieobciążoności estymatora wg metody najmniejszych kwadratów zdefiniowanych przez (2.9)–(2.11)). Dokładniej, odwrotność IMF jest dolnym ograniczeniem macierzy kowariancji identyfikowanych parametrów i określana jest przez nierówność Craméra-Rao [76]

$$\operatorname{cov}(\hat{\theta}) \succeq \mathcal{M}^{-1}, \tag{2.12}$$

gdzie: $\mathcal{M}$ — informacyjna macierz (powyższa nierówność powinna być interpretowana w sensie Löwnera w odniesieniu do macierzy symetrycznych, tzn. $A \succeq B$ jest równoważne temu, że $A - B$ musi być nieujemnie określona). Podstawowa trudność w posługiwaniu się IMF polega jednak na tym, że zależy ona od wektora $\bar{\theta}$ prawdziwych wartości estymowanych parametrów, a przecież te są nieznane [53, 105, 170, 172, 173, 228, 242, 243]. W praktyce używa się więc następujących wyrażeń aproksymujących IMF odpowiednio dla przypadków (2.5)–(2.7) [225, 228]:

- czujniki stacjonarne, pomiar w skończonej liczbie chwil $t_1, \ldots, t_K$

$$\mathcal{M}(x^1, \ldots, x^n) = \sum_{j=1}^{n} \sum_{k=1}^{K} g(x^j, t_k) g^{\mathsf{T}}(x^j, t_k), \tag{2.13}$$

- czujniki stacjonarne, pomiar ciągły w czasie

$$\mathcal{M}(x^1, \ldots, x^n) = \sum_{j=1}^{n} \int_0^{t_f} g(x^j, t) g^{\mathsf{T}}(x^j, t)\, \mathrm{d}t, \tag{2.14}$$

- czujniki ruchome, pomiar ciągły w czasie

$$\mathcal{M}(x^1, \ldots, x^n) = \sum_{j=1}^{n} \int_0^{t_f} g(x^j(t), t) g^{\mathsf{T}}(x^j(t), t)\, \mathrm{d}t, \tag{2.15}$$

gdzie:

$$g(x, t) = \left[\frac{\partial \hat{s}(x, t; \theta)}{\partial \theta_1}, \quad \ldots, \quad \frac{\partial \hat{s}(x, t; \theta)}{\partial \theta_m} \right]^{\mathsf{T}}_{\theta = \theta^0}. \tag{2.16}$$

W roli oszacowania wstępnego θ^0 *a priori* wektora $\bar{\theta}$ najczęściej występują wartości znamionowe estymowanych parametrów lub rezultaty eksperymentu wstępnego [186, 225, 228].

2.1.4 Kryteria optymalności eksperymentu

Załóżmy, że zastosowany estymator jest nieobciążony, co w przypadku kryterium najmniejszej sumy kwadratów błędów ma miejsce przy spełnieniu pewnych założeń, najczęściej przynajmniej w przybliżeniu [243]. Oznacza to, że przy wielokrotnym powtarzaniu eksperymentu identyfikującego, otrzymywane oceny koncentrowałyby się wokół prawdziwego wektora $\bar{\theta}$, a miarą ich rozproszenia byłaby macierz kowariancji estymatora

$$\operatorname{cov}(\hat{\theta}) = \mathrm{E}\Big[(\hat{\theta} - \bar{\theta})(\hat{\theta} - \bar{\theta})^{\mathsf{T}}\Big]. \tag{2.17}$$

Ponieważ w praktyce wykonuje się tylko jeden eksperyment prowadzący do konkretnej oceny $\hat{\theta}$, najbardziej pożądaną sytuacją byłyby małe wartości elementów tej macierzy, bo wówczas byłoby bardzo prawdopodobnym, że błąd estymacji $\hat{\theta} - \bar{\theta}$ okaże się mały. Biorąc pod uwagę związek IMF z macierzą kowariancji estymatora parametrów podany w poprzednim podrozdziale, optymalne położenia czujników pomiarowych powinny więc pociągać za sobą możliwie najmniejsze wartości elementów odwrotności IMF. Aby to osiągnąć w możliwie prosty sposób, zadanie redukuje się do poszukiwania minimum odpowiednio skonstruowanej funkcji rzeczywistej Ψ (kryterium optymalności) określonej na IMF. Kryteria najczęściej stosowane w zagadnieniach optymalnego planowania eksperymentu należą do rodziny funkcji określonych wzorem ogólnym [139, 170, 228, 242, 243]:

$$\Psi_\gamma(\mathcal{M}) = \begin{cases} \left[\dfrac{1}{m}\operatorname{trace}(Q\mathcal{M}^{-1}Q^{\mathsf{T}})^\gamma\right]^{1/\gamma}, & \text{gdy } \det(\mathcal{M}) \neq 0, \\ +\infty, & \text{gdy } \det(\mathcal{M}) = 0, \end{cases} \tag{2.18}$$

gdzie: Q — macierz wag.

W optymalnym planowaniu eksperymentu najczęściej stosuje się następujące kryteria [53, 170, 172, 228]:

- gdy $\gamma = 0$ i $Q = I_m$, gdzie: I_m — macierz jednostkowa stopnia m, m — liczba elementów wektora θ, otrzymuje się kryterium D-optymalności

  $$\mathcal{J}_D(x^1, \ldots, x^n) = \det\Big(\mathcal{M}(x^1, \ldots, x^n)\Big) \longrightarrow \max, \tag{2.19}$$

  które jest najczęściej spotykane w zastosowaniach. Posługując się nim minimalizuje się objętość asymptotycznej elipsoidy ufności identyfikowanych parametrów [53, 172];

- gdy $\gamma = 1$ i $Q = I_m$, otrzymuje się kryterium A-optymalności

  $$\mathcal{J}_A(x^1, \ldots, x^n) = \operatorname{trace}\Big(\mathcal{M}^{-1}(x^1, \ldots, x^n)\Big) \longrightarrow \min, \tag{2.20}$$

  w którym minimalizuje się sumę kwadratów długości osi asymptotycznej elipsoidy ufności;

- gdy $\gamma = \infty$ i $Q = I_m$, otrzymuje się kryterium E-optymalności

$$\mathcal{J}_E(x^1, \ldots, x^n) = \lambda_{\min}(\mathcal{M}(x^1, \ldots, x^n)) \longrightarrow \max, \tag{2.21}$$

gdzie: $\lambda_{\min}(\mathcal{M})$ — najmniejsza wartość własna macierzy $\mathcal{M}$. Korzystając z tego kryterium minimalizuje się długość najdłuższej osi asymptotycznej elipsoidy ufności.

2.1.5 Plan eksperymentu

Sformułowanie problemu w postaci (2.13)–(2.15) zakłada niezależność pomiarów wykonywanych przez różne czujniki bez względu na ich położenie. Daje to możliwość wykonywania wielokrotnych pomiarów w tych samych miejscach. Dopuszczenie pomiarów replikowanych powoduje konieczność rozróżnienia położeń poszczególnych czujników/pomiarów od miejsc wykonywania pomiarów. Zdefiniujmy więc sekwencję $x^1, \ldots, x^\ell$ jako różne miejsca wykonywania pomiarów, oraz ciąg odpowiadających im wartości $r_1, \ldots, r_\ell$ określających liczby sensorów wykonujących pomiary w danym miejscu. Miejsca wykonywania pomiarów x^i nazywane są zwyczajowo *punktami nośnika planu*, a zbiór par

$$\xi = \left\{ \begin{matrix} x^1, x^2, \ldots, x^\ell \\ p_1, p_2, \ldots, p_\ell \end{matrix} \right\}, \tag{2.22}$$

gdzie: $p_i = r_i/n$ — wagi punktów planu, $n = \sum_{i=1}^{\ell} r_i$, nazywamy *dokładnym planem* eksperymentu. Z punktu widzenia praktycznego, łatwiej jest korzystać z tzw. *przybliżonego* planu eksperymentu, który również przyjmuje formę (2.22), jednak co do współczynników p_i zakłada się jedynie, że spełniają warunki

$$p_i \geqslant 0, \quad i = 1, \ldots \ell, \quad \sum_{i=1}^{\ell} p_i = 1, \tag{2.23}$$

tzn. osłabia się wymagania, aby p_i były całkowitymi wielokrotnościami $1/n$. Zauważmy, że współczynnik p_i można interpretować jako prawdopodobieństwo wykonania pomiaru w danym miejscu x^i. Takie podejście prowadzi do tzw. *ciągłych* planów eksperymentu, stanowiących podstawę nowoczesnej teorii planowania eksperymentu [4, 52, 56, 58, 76, 118, 170, 172, 186, 225, 228, 243].

2.1.6 Wyznaczanie współczynników wrażliwości

Jak można zauważyć w rozdziale 2.1.3, wyznaczenie macierzy informacyjnej (2.13)–(2.15) musi być poprzedzone określeniem tzw. współczynników wrażliwości (2.16) opisujących wrażliwość stanu układu na zmianę jego parametrów.

Problem jest blisko związany z analizą wrażliwości, która gra kluczową rolę w zagadnieniach identyfikacji, optymalizacji kształtu czy też analizie wrażliwości [9, 54, 85, 221]. W kontekście estymacji parametrów modeli z czasoprzestrzenną

dynamiką, problem wyznaczania współczynników wrażliwości został zapoczątkowany pracą [30], a do nowszych prac poruszających zagadnienie wyznaczania równań wrażliwości, można zaliczyć monografię [206], która jest poświęcona inżynierskim zagadnieniom związanym z zarządzaniem zasobami wód podziemnych.

W niniejszej pracy, do wyznaczania równań wrażliwościowych umożliwiających wyznaczenie współczynników wrażliwości (2.16), zastosowano metodę bezpośredniego różniczkowania [228]. Metoda ta polega na zróżniczkowaniu równań systemu (2.1),(2.2) oraz (2.3), względem poszczególnych nieznanych parametrach w celu otrzymania tzw. równań wrażliwości [228]

$$\begin{aligned}\frac{\partial}{\partial t}\left[\frac{\partial s}{\partial \theta_i}\right] &= \frac{\partial \mathcal{H}}{\partial s}\frac{\partial s}{\partial \theta_i} + \frac{\partial \mathcal{H}}{\partial s_{x_1}}\frac{\partial}{\partial x_1}\left[\frac{\partial s}{\partial \theta_i}\right] + \frac{\partial \mathcal{H}}{\partial s_{x_2}}\frac{\partial}{\partial x_2}\left[\frac{\partial s}{\partial \theta_i}\right] \\ &+ \frac{\partial \mathcal{H}}{\partial s_{x_1 x_1}}\frac{\partial^2}{\partial x_1^2}\left[\frac{\partial s}{\partial \theta_i}\right] + \frac{\partial \mathcal{H}}{\partial s_{x_2 x_2}}\frac{\partial^2}{\partial x_2^2}\left[\frac{\partial s}{\partial \theta_i}\right] \\ &+ \frac{\partial \mathcal{H}}{\partial \theta_i} \qquad \text{w } \ \Omega \times T\end{aligned} \tag{2.24}$$

z warunkami

$$\frac{\partial s}{\partial \theta_i}(x,0) = 0 \qquad \text{w } \ \Omega, \tag{2.25}$$

$$\frac{\partial s}{\partial \theta_i}(x,t) = 0 \qquad \text{na } \ \partial\Omega \times T, \tag{2.26}$$

dla $i = 1, \ldots, m$, gdzie s_{x_j} i $s_{x_j x_j}$ oznaczają

$$\frac{\partial s}{\partial x_j} \quad \text{oraz} \quad \frac{\partial^2 s}{\partial x_j^2}, \tag{2.27}$$

dla odpowiednio $j = 1, 2$.

Wyznaczenie pochodnych funkcji $\mathcal{H}$ ze względu na parametry odbywa się po rozwiązaniu oryginalnego zagadnienia (2.1)–(2.3) dla konkretnej wartości parametrów θ. Dopiero wtedy można rozwiązać układ równań wrażliwościowych (2.24)–(2.26) i obliczyć wartości wektorów wrażliwości (2.16) wymagane do określenia macierzy informacyjnych (2.13)–(2.15). Szczegółowe omówienie przedstawionej metody, jak również innych metod, zawarto w monografii [228].

2.2 Problemy komplikujące planowanie eksperymentu

Sformułowanie problemu optymalnego rozmieszczania czujników w kategoriach zadania optymalizacji pozornie pozwala interpretować ten problem jako standardowy problem programowania nieliniowego. Wydaje się więc, że wykorzystanie jednego

z wielu znanych algorytmów optymalizacji pozwoli na szybkie i proste znalezienia ekstremum wybranego kryterium. Niestety, w rozważanych zagadnieniach napotyka się na szereg komplikacji powodujących, że problem optymalnej obserwacji układów z czasoprzestrzenną dynamiką staje się problemem nietrywialnym. Występowanie zjawiska klasteryzacji czujników, konieczność uwzględnienia korelacji pomiędzy wykonywanymi pomiarami, duża wymiarowość oraz silna nieliniowość funkcji definiujących wybrane kryterium są jednymi z poważniejszych problemów jakie mogą się pojawić. W celu przedstawienia omawianych problemów przeanalizujmy następujący przykład.

Przykład 2.1
Rozważmy dwuwymiarowy obszar $\Omega = (0,1)^2$ o brzegu $\partial\Omega$, horyzont czasowy $T = [0,1]$ oraz proces przewodnictwa ciepła opisany równaniem

$$\frac{\partial s(x,t)}{\partial t} = \frac{\partial}{\partial x_1}\left(a(x)\frac{\partial s(x,t)}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(a(x)\frac{\partial s(x,t)}{\partial x_2}\right), \quad x \in \Omega, \quad t \in T, \tag{2.28}$$

gdzie:

$$a(x) = \theta_1 + \theta_2 x_1 + \theta_3 x_2, \tag{2.29}$$

uzupełniony warunkiem początkowym

$$s(x,0) = 5, \quad x \in \Omega, \tag{2.30}$$

i brzegowym

$$s(x,t) = 5(1-t), \quad x \in \partial\Omega, \quad t \in T. \tag{2.31}$$

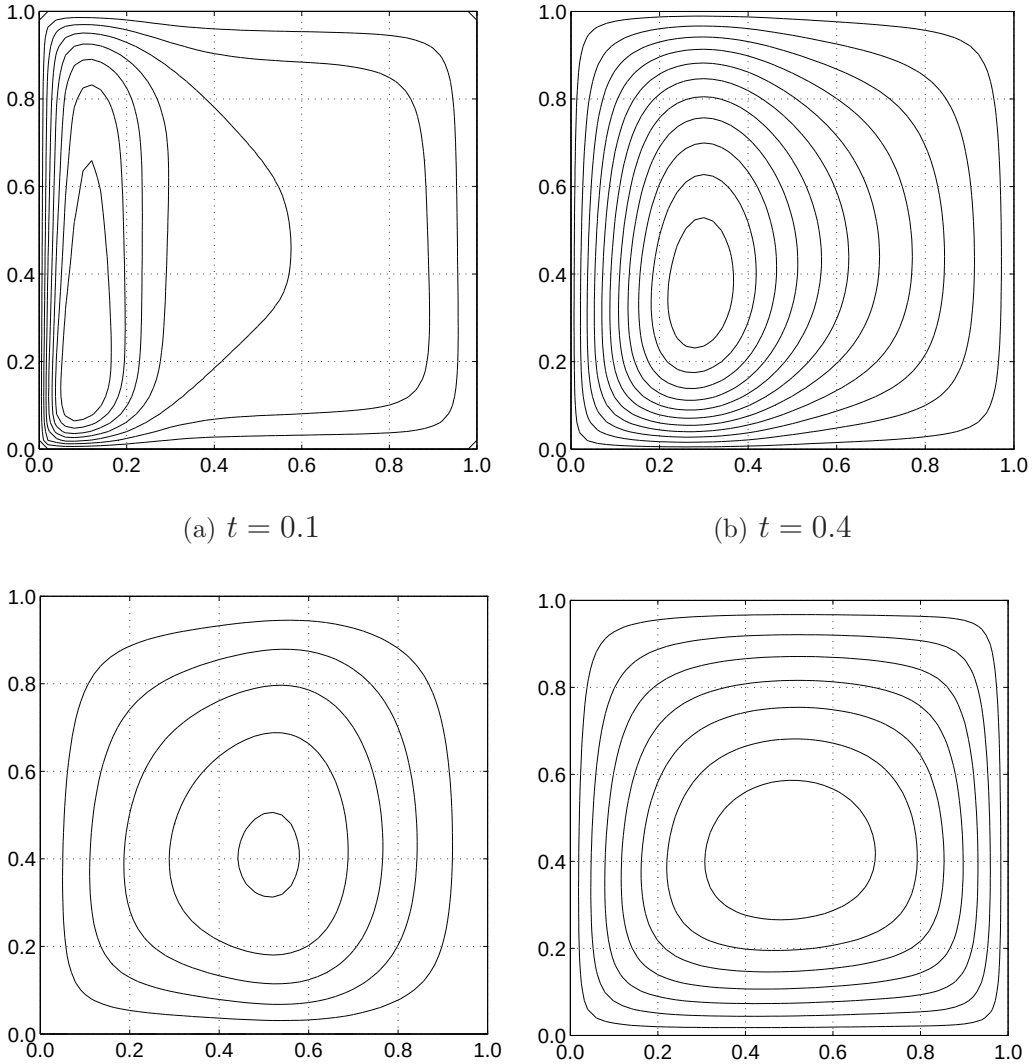
W celu numerycznego rozwiązania równań (2.28)–(2.31) oraz równań wrażliwości stanu układu na zmiany parametrów (patrz rozdz. 2.1.6) wykorzystano metodę elementu skończonego [88, 112, 127, 137]. Rozwiązanie wyznaczono w oparciu o triangulację obszaru Ω siatką o gęstości 21×21 i podział horyzontu czasowego T na 81 równych przedziałów czasowych. Obliczony rozkład temperatury w wybranych chwilach czasowych przedstawiono na rys. 2.1–2.2.

W obszarze Ω określono miejsca, w których można rozmieścić czujniki. Są one określone w postaci węzłów równomiernej siatki kwadratowej o wielkości 21×21. Zdefiniujmy problem jako zagadnienie optymalnego rozmieszczenia 21 dostępnych czujników stacjonarnych w tak wyznaczonych miejscach. W sformułowaniu wykorzystano informacyjną macierz Fishera (2.13) oraz określone na niej kryterium D-optymalności (2.19). Rozwiązanie przedstawionego problemu z wykorzystaniem zmodyfikowanej (umożliwienie tworzenia replikacyjnych planów eksperymentu bez uwzględniania korelacji pomiędzy obserwacjami) metody przedstawionej w rozdz. 4.2 pozwala na wyznaczenie D-optymalnego planu eksperymentu postaci

$$\xi = \left\{ \begin{array}{ccccccc} (4,4) & (5,15) & (6,15) & (12,13) & (13,12) & (15,5) & (16,6) \\ \frac{7}{21} & \frac{3}{21} & \frac{3}{21} & \frac{1}{21} & \frac{1}{21} & \frac{3}{21} & \frac{3}{21} \end{array} \right\}, \tag{2.32}$$

co implikuje optymalną konfigurację czujników przedstawioną na rys. 2.3.

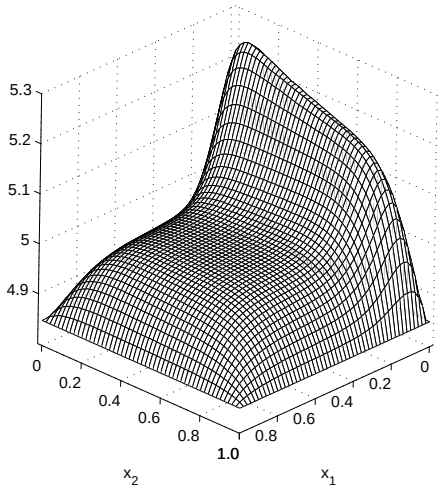
Jak można zauważyć, wyznaczona konfiguracja przedstawia 21 czujników skupionych w czterech grupach, tzw. skupieniach. W części miejsc planu eksperymentu umieszczonych zostało po kilka czujników. Takie rozmieszczenie czujników nie ma sensu przy założeniu tworzenia bezreplikacyjnego planu eksperymentu (zob. rozdz. 2.2.4), tzn. w sytuacji gdy wyklucza się możliwość umieszczenia w jednym miejscu kilku czujników, tak aby wykonywane przez nie pomiary nie wpływały na siebie.



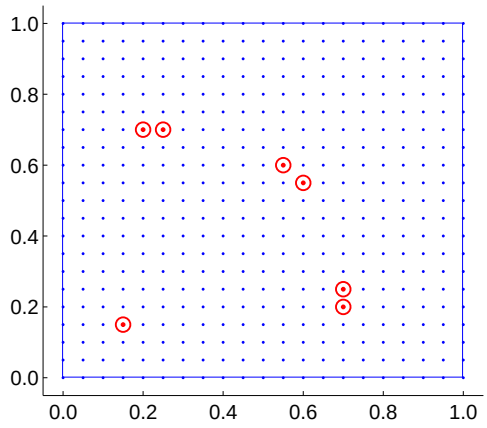
(a) $t = 0.1$ (b) $t = 0.4$

(c) $t = 0.7$ (d) $t = 1.0$

Rysunek 2.1: Wykresy konturowe pola temperatury w podanych chwilach czasowych w Przykładzie 2.1.



Rysunek 2.2: Pole temperatury w Przykładzie 2.1 w chwili $t = 0.1$.



Rysunek 2.3: D-optymalne konfiguracje 21 czujników bez uwzględnienia korelacji pomiędzy czujnikami w Przykładzie 2.1. Zauważyć można cztery skupienia optymalnych punktów pomiarowych.

♦

2.2.1 Skorelowane pomiary

W rzeczywistych środowiskach pomiarowych wykonanie pomiaru przez jeden czujnik niejednokrotnie powoduje wystąpienie zakłócenia w pomiarze wykonywanym przez inny, blisko położony czujnik. Zjawisko to jest pomijane w prawie wszystkich pracach dotyczących optymalnego rozmieszczenia czujników pomiarowych w celu estymacji parametrów obiektów z czasoprzestrzenną dynamiką. Z jednej strony takie uproszczenie powoduje możliwość uzyskania niewątpliwie eleganckich rezulta-

tów teoretycznych, ale z drugiej strony jest ono możliwe do zastosowania jedynie w przypadku niewielkiej liczby zagadnień. Monitorowanie stężenia smogu, meteorologia czy też sejsmografia są typowymi dziedzinami, w których istnieje częsta potrzeba uwzględniania bardziej wysublimowanej struktury zakłóceń niż szum biały.

2.2.2 Skupianie się czujników

Jak już wspomniano i zilustrowano w Przykładzie 2.1, jedną z wielu trudności w planowaniu optymalnych położeń czujników jest tendencja do ich skupiania się w niewielkiej liczbie punktów (określa się to czasem jako tzw. klasteryzację czujników), co jest bezpośrednią konsekwencją wyidealizowanego założenia o niezależności pomiarów wykonywanych przez czujniki (zob. rozdz. 2.2.1). Powoduje to konieczność uwzględnienia w sformułowaniu zadania optymalizacji ograniczenia na zachowanie minimalnych dopuszczalnych odległości między czujnikami. Różne czujniki nie mogą bowiem zazwyczaj wykonać pomiaru w jednym punkcie bez wpływu na pomiar wykonywany przez inny czujnik.

Jedną z technik, która ma zapobiegać zjawisku skupiania się jest metoda opisana szczegółowo w monografii [228]. Polega ona na uwzględnieniu w wyrażeniu definiującym macierz informacyjną członu opisującego korelację między obserwacjami wykonywanymi przez różne czujniki. Takie bezpośrednie podejście bywa nieco prostsze w zastosowaniu niż metoda zaproponowana przez Fedorova [57], opierająca się na ograniczaniu gęstości przestrzennej czujników (liczby czujników na jednostkę powierzchni). Wprawdzie wykorzystanie gęstości czujników prowadzi do bardzo czytelnych warunków optymalności, sprowadzających się do separowalności zbioru nośników planu optymalnego i jego dopełnienia ze względu na wartości funkcji pochodnej kryterium, zob. [228], jednak właściwa implementacja rozwiązania i synteza algorytmów w celu generowania rozwiązań numerycznych pozostają nadal problemami otwartymi.

Podejście oparte na wykorzystaniu korelacji opiera się na uwzględnieniu wzajemnych wpływów pomiędzy pomiarami wykonywanymi przez różne czujniki. Uwzględnienie korelacji między pomiarami wykonywanymi przez różne czujniki powoduje, że macierz Fishera przybiera następującą postać [170, 225, 228]:

$$\mathcal{M}(x^1,\ldots,x^n) = \sum_{i=1}^{n}\sum_{j=1}^{n}\int_T w_{ij}(t)g(x^i,t)g^{\mathsf{T}}(x^j,t)\,\mathrm{d}t, \tag{2.33}$$

gdzie:

$$g(x,t) = \left[\frac{\partial y(x,t;\theta)}{\partial\theta_1},\quad \ldots,\quad \frac{\partial y(x,t;\theta)}{\partial\theta_m}\right]^{\mathsf{T}}_{\theta=\theta^0}, \tag{2.34}$$

$$W(t) = [w_{ij}(t)] = C^{-1}(t), \tag{2.35}$$

przy czym założono przestrzenne skorelowanie szumu pomiarowego postaci

$$\mathrm{E}\{\varepsilon(x^i,t)\varepsilon(x^j,\tau)\} = q(x^i,x^j,t)\delta(t-\tau), \tag{2.36}$$

oraz

$$C(t) = [c_{ij}(t)], \quad c_{ij}(t) = q(x^i, x^j, t). \tag{2.37}$$

Po zastosowaniu zapisu macierzowego, otrzymujemy

$$\mathcal{M}(x^1, \ldots, x^n) = \int_T G(t) C^{-1}(t) G^{\mathsf{T}}(t)\, \mathrm{d}t, \tag{2.38}$$

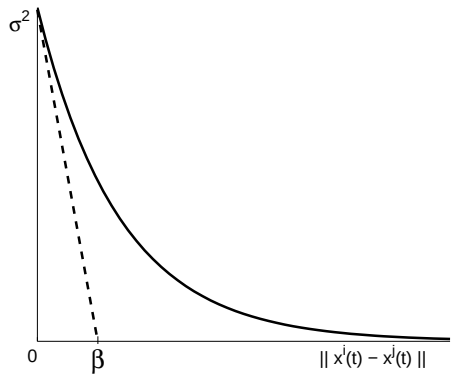
gdzie:

$$G(\cdot) = \left[g(x^1(t), \cdot) | \quad \ldots \quad | g(x^n(t), \cdot) \right]. \tag{2.39}$$

Macierz kowariancji C może być zależna od wielu czynników związanych z rozmieszczanymi czujnikami, jednak bardzo naturalne jest uwzględnienie zależności od odległości pomiędzy czujnikami, np. w następującej formie [157]:

$$q(x^i, x^j, t) = \sigma^2 \exp\left(\frac{-\|x^i - x^j\|}{\beta} \right), \tag{2.40}$$

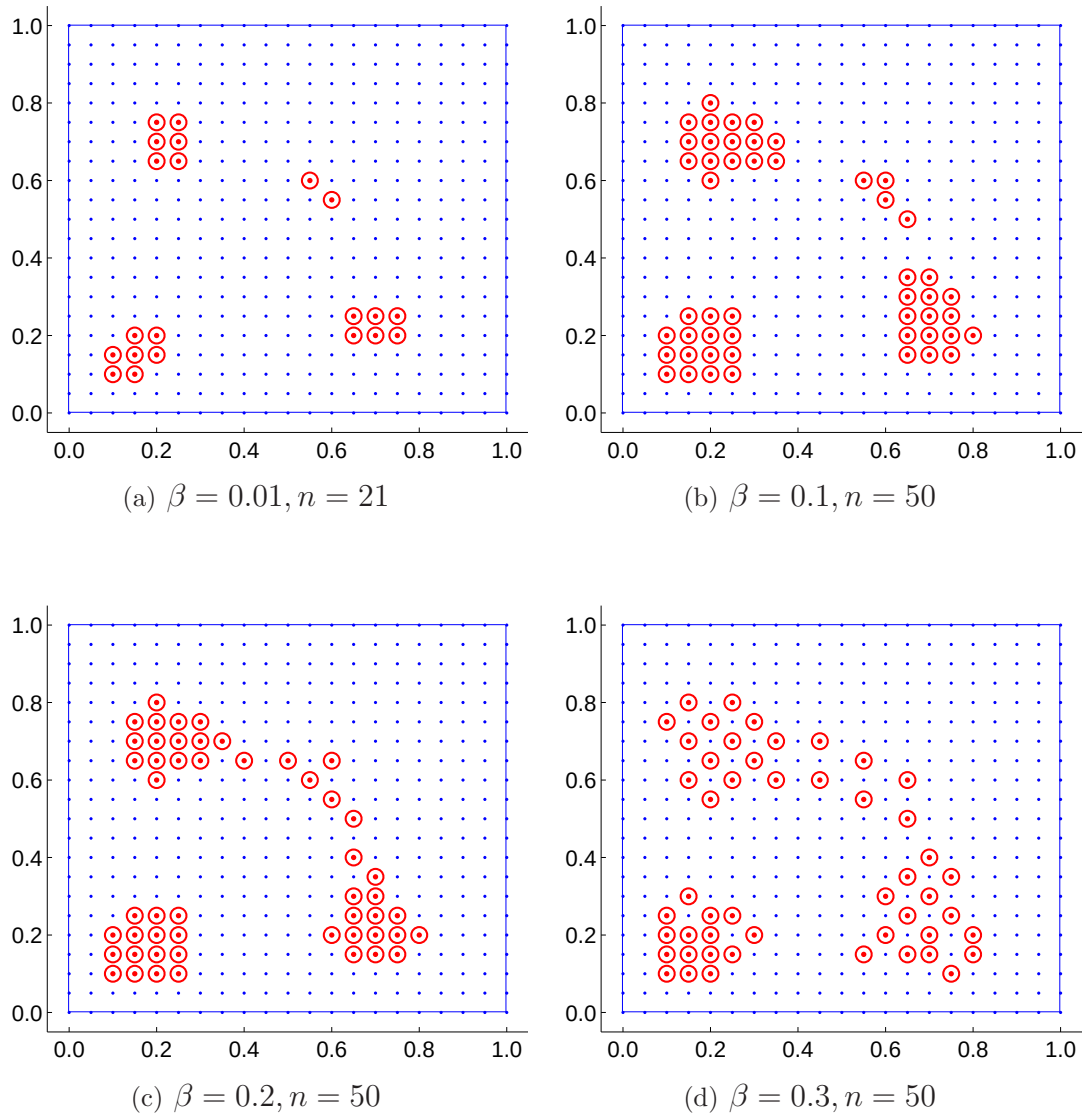
gdzie: β — współczynnik determinujący stopień skorelowania obserwacji (zob. rys. 2.4).



Rysunek 2.4: Kowariancja obserwacji wykonywanych przez i-ty i j-ty czujnik, modelowana przez wzór (2.40). Linia przerywana oznacza styczną do wykresu. Duże wartości β implikują duże skorelowanie obserwacji.

Warto odnotować, że jeżeli jakiekolwiek dwa czujniki zostaną umieszczone w tym samym punkcie, macierz C stanie się osobliwa (kolumny oraz wiersze odpowiadające tym czujnikom będą identyczne), i tym samym konfiguracje czujników z replikacjami punktów pomiarowych stają się wyeliminowane [228]. Wprowadzenie macierzy C (2.40) rozwiązuje więc, w pewien sposób, problem skupiania się czujników. Na rys. 2.5(a) można zaobserwować optymalne rozmieszczenie czujników w Przykładzie 2.1 przy uwzględnieniu korelacji pomiarów (w modelu (2.40) przyjęto wartość $\beta = 0.01$).

Dla porównania, rysunki 2.5(b)–2.5(d) przedstawiają wpływ wartości parametru β na stopień skorelowania obserwacji i na otrzymane optymalne konfiguracje czujników w Przykładzie 2.1.



(a) $\beta = 0.01, n = 21$ (b) $\beta = 0.1, n = 50$

(c) $\beta = 0.2, n = 50$ (d) $\beta = 0.3, n = 50$

Rysunek 2.5: Rozmieszczenia czujników w Przykładzie 2.1 dla różnej ich liczby oraz dla różnych wartości parametru skorelowania obserwacji.

Oczywiście, uwzględnienie korelacji pomiędzy pomiarami zgodnie z (2.38) jest tylko uogólnieniem wzorów (2.13)–(2.15), dla których

$$C = \sigma^2 I, \tag{2.41}$$

gdzie: I — $n \times n$ wymiarowa macierz jednostkowa. We wzorach tych przyjęto $\sigma^2 = 1$ z uwagi na to, że przy niezależności obserwacji wartość parametru σ^2 nie wpływa na rozwiązanie optymalne.

2.2.3 Problemy obliczeniowe

Jak można zauważyć na podstawie (2.38), macierz informacyjna zależy od odwrotności macierzy kowariancji C. Konieczność wykonywania operacji odwracania macierzy powoduje, że rozwiązywanie zagadnień optymalnego rozmieszczenia czujników przy uwzględnieniu korelacji pomiarów staje się bardzo czasochłonne. Problem ten został szeroko omówiony w [228, 229]. Zaproponowane tam rozwiązanie z zastosowaniem algorytmu wymiany pozwala na znaczącą redukcję najbardziej pracochłonnych operacji tj. odwracania macierzy kowariancji. Podczas działania algorytmu wymiany najbardziej pracochłonnymi operacjami są: usunięcie czujnika z aktualnego planu oraz dodanie nowego czujnika do aktualnego planu. Korzystanie z klasycznego podejścia opisanego w rozdz. 2.2.2 powoduje konieczność przeliczenia na nowo elementów macierzy informacyjnej, a wielokrotne wykonywanie tej operacji znacząco wydłuża czas obliczeń. Szczegółowe omówienie problemu redukcji tego czasu dla zagadnienia optymalnej aktywacji czujników skanujących znajduje się w rozdz. 4.4.

2.2.4 Bezreplikacyjne plany eksperymentu

Założenie mówiące o możliwości wykonywania pomiarów z replikacjami jest bardzo rzadko możliwe do zrealizowania w warunkach rzeczywistych. Z punktu widzenia inżynierskiego liczba punktów planu określająca liczbę różnych miejsc wykonania pomiarów powinna więc być równa liczbie czujników, tzn. $\ell = n$. Sformułowanie zadania z uwzględnieniem tego warunku pozwala rozważać plany eksperymentu postaci

$$\xi = \left\{x^1, x^2, \ldots, x^n\right\}, \tag{2.42}$$

przy czym $x^i \neq x^j$ o ile $i \neq j$. Nie ma sensu operowanie wagami, które są w tym przypadku wszystkie równe $1/n$. Otrzymuje się w ten sposób nieliniowe zagadnienie przydziału n czujników do n różnych położeń wybranych spośród N dostępnych położeń dopuszczalnych.

W celu zobrazowania przedstawionego zagadnienia rozważmy ponownie Przykład 2.1. Wyznaczone optymalne plany eksperymentów powinny mieć możliwość implementacji w rzeczywistych warunkach. Rozważając Przykład 2.1 i obliczony optymalny plan eksperymentu (2.32) można zauważyć, że jego realizacja wymaga wykonywania pomiarów przez kilka czujników umieszczonych w tych samych miejscach. Oczywiście, fizyczna realizacja tego typu replikacyjnego planu eksperymentu jest niemożliwa. Jednakże, liczbę czujników ($n = 21$) można również traktować jako liczbę powtórzeń doświadczenia, przy czym w pojedynczym powtórzeniu pomiary wykonuje tylko jeden czujnik. Przedstawienie problemu w tej postaci pozwala na fizyczną realizację rozwiązania. Jednak warunki rzeczywiste bardzo rzadko, wręcz nigdy, nie będą identyczne w przypadku wykonywania kolejnych pomiarów w punktach, gdzie są one wymagane wielokrotnie. Innym podejściem możliwym do zastosowania jest potraktowanie tak opracowanego planu eksperymentu (2.32) jako pewnego rozkładu prawdopodobieństwa wykonywania pomiarów w poszczególnych

punktach. Przy takiej interpretacji wagi planu (2.32) dla poszczególnych punktów nośnika planu oznaczają prawdopodobieństwo wykonania pomiaru w danym punkcie. Przedstawienie planu w postaci (2.32) pozwala traktować pewne miejsca kandydujące do umieszczenia czujników jako bardziej wartościowe (większa wartość wagi/prawdopodobieństwa) punktu.

Bezreplikacyjne plany eksperymentu można konstruować poprzez uwzględnienie korelacji pomiędzy wykonywanymi obserwacjami (zob. rozdz. 2.2.2). Kosztem zwiększenia nakładu obliczeń związanych z wyznaczaniem macierzy informacyjnej uzyskuje się realizowalne rozmieszczenia czujników. Wymaga to jednak znajomości funkcji kowariancji szumu pomiarowego. Innym podejściem jest zdefiniowanie ograniczeń na wagi planu eksperymentu w samej metodzie poszukiwania optymalnych rozmieszczeń czujników. Taką metodę przedstawiono w pracy [235], gdzie zaproponowano wykorzystanie dekompozycji symplicjalnej.

W niniejszej pracy przedstawiono podejścia oparte na zastosowaniu dyskretnych heurystycznych algorytmów optymalizacji globalnej (zob. rozdz. 4). Problem optymalnej obserwacji z wykorzystaniem sieci czujników stacjonarnych oraz skanujących został zdefiniowany jako zagadnienie wyboru spośród N czujników, rozlokowanych w obszarze działania badanego zjawiska, pewnego n elementowego podzbioru, którego aktywowanie dostarczy największej ilości informacji o badanym zjawisku. Zakłada się, że rozmieszczone czujniki znajdują się w różnych lokalizacjach w obszarze działania badanego zjawiska, a każdy z aktywowanych czujników może wykonać pomiar tylko w jednej serii pomiarowej.

Złożoność zagadnienia wyboru n czujników spośród N rozlokowanych w obszarze działania badanego zjawiska jest znaczna, a pełne przeszukanie przestrzeni możliwych rozwiązań w sensownym czasie jest, dla większości zagadnień, niemożliwe. Zastosowanie heurystycznych algorytmów dyskretnej optymalizacji globalnej pozwala na znaczną redukcję koniecznej do przeszukania przestrzeni poszukiwań, a w połączeniu z wykorzystaniem wieloprocesorowych środowisk obliczeniowych — na znaczne skrócenie czasu uzyskania rezultatów symulacji.

Przedstawienie zagadnienia optymalnej obserwacji jako kombinatorycznego zagadnienia wyboru odpowiedniego podzbioru czujników nie zwalnia z konieczności uwzględniania wzajemnego wpływu pomiarów dokonywanych przez różne czujniki, dlatego opracowane algorytmy uwzględniają korelacje pomiędzy obserwacjami (zob. rozdz. 2.2.2). Jak można zauważyć w równaniach (2.35),(2.38), konieczność wyznaczania odwrotności macierzy kowariancji znacznie zwiększa złożoność obliczeniową zagadnienia optymalnej aktywacji sensorów. Zastosowanie algorytmów optymalizacji dyskretnej pozwala na wykorzystanie metody redukcji czasochłonnych operacji opartej na formule Shermana-Morrisona-Woodbury'ego, przedstawionej w pracach [83, 170, 228]. Jak zostanie pokazane, zastosowanie tej metody pozwala na znacznie zmniejszenie czasu obliczeń związanych z wyznaczeniem optymalnych konfiguracji aktywnych czujników zarówno w stacjonarnych, jak i w skanujących sieciach sensorycznych.

3

Obliczenia równoległe

3.1 Wstęp

Rozwój technik programowania, zaawansowanie metod badawczych oraz dążenie do uzyskiwania coraz dokładniejszych wyników badań to tylko niektóre z licznych czynników motywujących rozwój systemów komputerowych wspomagających badania, zwłaszcza tych wspomagających obliczenia numeryczne. Gwałtowny rozwój technik komputerowych w drugiej połowie XX w. spowodował ogromne zmiany w sposobie przeprowadzania badań i ich weryfikacji. Zamiast budowania fizycznych modeli sprawdzających słuszność przyjętych założeń i opracowanych koncepcji, większość symulacji jest przeprowadzana w wirtualnym świecie utworzonym środkami programistycznymi.

Początkowo systemy komputerowe składały się z pojedynczych jednostek obliczeniowych wykonujących rozkazy sekwencyjne. Intensywne badania nad coraz bardziej zaawansowanymi materiałami półprzewodnikowymi pozwalały konstruować coraz wydajniejsze systemy obliczeniowe. Jednak nowsze konstrukcje zbliżają się do ograniczeń stawianych przez prawa fizyki. Najważniejszą nieprzekraczalną granicą jest prędkość światła. Jej istnienie z założenia ogranicza częstotliwość pracy procesorów, a co za tym idzie — liczbę instrukcji wykonywanych na sekundę. Jednocześnie prawa fizyki i ograniczenia technologiczne powodują, że wydajność systemów komputerowych ograniczona jest również przez szybkość działania układów pamięci, dysków twardych czy sieci komputerowych. Ciągły rozwój przemysłu półprzewodnikowego pozwala na pokonywanie kolejnych barier technologicznych, jednak konieczność zwiększania wydajności systemów komputerowych wymaga znajdowania nowych sposobów osiągnięcia tego celu.

Naturalnym sposobem rozwiązania problemów wydajnościowych było zwiększenie liczby jednostek wykonawczych, tak aby obliczenia wykonywane były jednocze-

śnie przez większą liczbę procesorów. Pierwsze środowiska równoległe pojawiły się pod koniec lat 70-tych ubiegłego wieku, a znacznie większą cenę w porównaniu z klasycznymi rozwiązaniami opartymi o procesory skalarne rekompensowała możliwość rozwiązywania skomplikowanych problemów naukowych. Projektanci od początku zaproponowali wiele różnych architektur systemów równoległych, jednak systemy oparte na procesorach wektorowych lub macierzowych zaczęły stopniowo ustępować miejsca systemom składającym się z wielu węzłów zbudowanych z zastosowaniem procesorów skalarnych, połączonych przy pomocy sieci komputerowych. Zasadniczą rolę odegrała cena systemu równoległego, znacznie niższa w przypadku systemów składających się z powszechnie dostępnych jednostek obliczeniowych. Aktualnie na liście najszybszych systemów komputerowych TOP500 [219] znajduje się tylko jeden system oparty o procesory wektorowe, a wiodącą architekturą systemów równoległych są klastry komputerowe (ponad 80%). Niska cena, a jednocześnie coraz większa specjalizacja i wykorzystanie wysoko wydajnych komponentów powodują, że klastry aktualnie przeżywają gwałtowny rozwój. Równocześnie rozwijane oprogramowanie pozwalające w łatwy i nieskomplikowany sposób tworzyć aplikacje umożliwiające rozwiązywanie skomplikowanych problemów powodują, że systemy równoległe i środowiska klastrowe są aktualnie wiodącą technologią pozwalającą rozwiązywać tzw. Wielkie Wyzwania Nauki (*ang.* Grand Challenges of Science) [48, 79].

3.2 Klasyfikacje równoległych systemów obliczeniowych

Intensywny rozwój technik komputerowych w ostatnich kilkudziesięciu latach skutkował opracowaniem wielu różnych architektur umożliwiających przetwarzanie równoległe. Jedną z pierwszych, a zarazem najbardziej znanych klasyfikacji jest podział zaproponowany przez Flynna w 1966 roku [64]. Opiera się ona na segregacji systemów komputerowych w zależności od liczby strumieni rozkazów i danych. Wyróżnione są następujące klasy systemów komputerowych [51, 169]:

- SISD (*ang.* Single Instruction – Single Data) — pojedynczy strumień rozkazów oraz danych. Do tej grupy zalicza się klasyczne komputery skalarne.
- SIMD (*ang.* Single Instruction – Multiple Data) — pojedynczy strumień rozkazów operujący na wielu strumieniach danych. Komputery należące do tej grupy składają się z wielu procesorów sterowanych przez centralną jednostkę sterującą. Przykładem powyższej architektury są komputery wektorowe oraz macierzowe.
- MISD (*ang.* Multiple Instruction – Single Data) — wiele strumieni rozkazów operujących na pojedynczym strumieniu danych. Grupa ta nie posiada reprezentantów w postaci działających znanych architektur sprzętowych.

- MIMD (*ang.* Multiple Instruction – Multiple Data) — wiele strumieni rozkazów oraz danych. Do tej grupy należą systemy wykonujące jednocześnie wiele zestawów instrukcji (programów) na różnych zestawach danych. Systemami należącymi do tej grupy są maszyny wieloprocesorowe oraz klastry komputerowe.

Szybko okazało się, że znaczna większość systemów komputerowych projektowanych w późniejszych latach należała do grupy MIMD. Dlatego, w roku 1988, Johnson [99] zaproponował rozszerzenie klasyfikacji systemów komputerowych. Nowy schemat podziału jest zależny od struktury pamięci oraz mechanizmów synchronizacji danych pomiędzy procesorami. Podział przedstawia się następująco [169]:

- GMSV (*ang.* Global Memory – Shared Variables) — globalna pamięć wraz ze współdzielonymi zmiennymi,
- GMMP (*ang.* Global Memory – Message Passing) — globalna pamięć wraz z przesyłaniem komunikatów,
- DMSV (*ang.* Distributed Memory – Shared Variables) — pamięć rozproszona wraz ze współdzielonymi zmiennymi,
- DMMP (*ang.* Distributed Memory – Message Passing) — pamięć rozproszona wraz z przesyłaniem komunikatów.

Rozszerzenie taksonomii Flynna o dodatkowy podział architektury MIMD pozwala na łatwiejsze uporządkowanie istniejących systemów równoległych. Do grupy GMSV należą systemy wieloprocesorowe z pamięcią współdzieloną. Systemy należące do grupy GMMP nie są rozpowszechnione. Grupa DMSV jest charakteryzowana jako systemy rozproszone z pamięcią współdzieloną. Pozwalają one na korzystanie z zalet systemów z pamięcią rozproszoną wraz z łatwością programowania z wykorzystaniem zmiennych współdzielonych. Ostatnia grupa, DMMP, są to systemy składające się z wielu procesorów, posiadających niezależną pamięć, komunikujących się za pośrednictwem komunikatów. Do tej grupy zaliczamy większość istniejących maszyn równoległych, w tym m.in. klastry komputerowe.

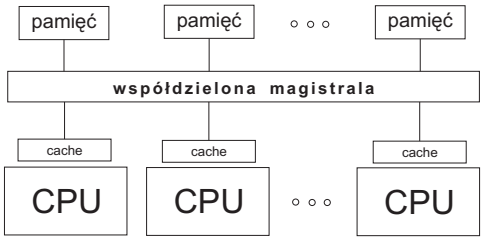
3.3 Architektury systemów obliczeniowych

Przedstawione powyżej klasyfikacje nie są jedynymi. Najczęściej spotykany jest podział ze względu na sposób dostępu do pamięci operacyjnej systemu równoległego. Systemy równoległe dzielą się na [51, 77, 169]

- maszyny o pamięci współdzielonej (*ang.* shared memory access),
- maszyny o pamięci rozproszonej (*ang.* distributed memory access).

3.3.1 Maszyny o pamięci współdzielonej

Komputery równoległe o pamięci współdzielonej składają się z wielu procesorów, które mogą uzyskiwać dostęp do wspólnej pamięci poprzez współdzieloną magistralę systemową (zob. rys. 3.1). Taka architektura powoduje, że każdy z procesorów ma dostęp do całej pamięci operacyjnej w postaci jednolitej przestrzeni adresowej (*ang.* Uniform Memory Access — UMA). Występowanie magistrali systemowej powoduje, że operacje odwoływania się do pamięci są bardzo szybkie i wydajne. Niestety, taka architektura ma wadę polegającą na konieczności współdzielenia magistrali pomiędzy wieloma procesorami. Jednoczesny dostęp wielu procesorów do pamięci powoduje konflikty i spowalnia działanie całego systemu. Dzielona magistrala ogranicza skalowalność systemu i dlatego też nie spotyka się systemów z więcej niż 64 procesorami dostępującymi do wspólnej pamięci. Dodatkowo występowanie bardzo szybkiej pamięci wewnętrznej procesorów (*ang.* cache) powoduje problemy z synchronizacją operacji na danych zawartych w pamięci cache i pamięci systemowej.



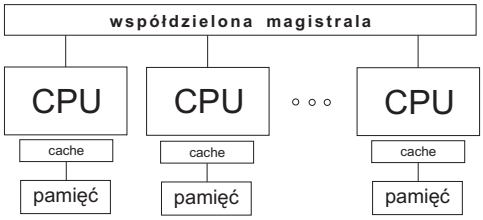
Rysunek 3.1: Maszyna o pamięci współdzielonej.

Spotyka się warianty przedstawionej architektury w postaci architektury NUMA (*ang.* Non-Uniform Memory Architecture), w której każdy z procesorów posiada własną lokalną pamięć oraz ma możliwość dostępu do pamięci innych procesorów. Patrząc na system równoległy z punktu widzenia systemu operacyjnego i aplikacji, jest on traktowany jak klasyczny system o pamięci współdzielonej.

3.3.2 Maszyny o pamięci rozproszonej

Na maszyny o pamięci rozproszonej składa się wiele osobnych systemów komputerowych połączonych ze sobą za pomocą wewnętrznej, bardzo szybkiej sieci (zob. rys. 3.2). Każdy procesor ma bezpośredni dostęp wyłącznie do własnej pamięci lokalnej. Dostęp do danych znajdujących się poza lokalnymi zasobami odbywa się za pomocą komunikatów przesyłanych między procesorami. Architektura maszyn o pamięci rozproszonej powoduje, że są to systemy bardzo skalowalne. Z drugiej strony, konieczność przesyłania danych pomiędzy procesorami, o które należy zadbać na etapie projektowania algorytmów powoduje, że są one bardziej wymagające od strony programistycznej.

Pewną odpowiedzią na problemy związane z programowaniem maszyn z pamięcią rozproszoną są systemy z rozproszoną pamięcią współdzieloną (*ang.* Distributed-



Rysunek 3.2: Maszyna o pamięci rozproszonej.

Shared Memory — DSM). W tego typu systemach pamięć pomiędzy procesorami jest rozproszona powodując, że system musi wymieniać dane pomiędzy procesorami za pomocą komunikatów. Jednak model programowania zakłada, że pamięć jest traktowana jako całość, a szczegóły architektury są ukrywane przed programistą.

3.3.3 Topologie połączeń

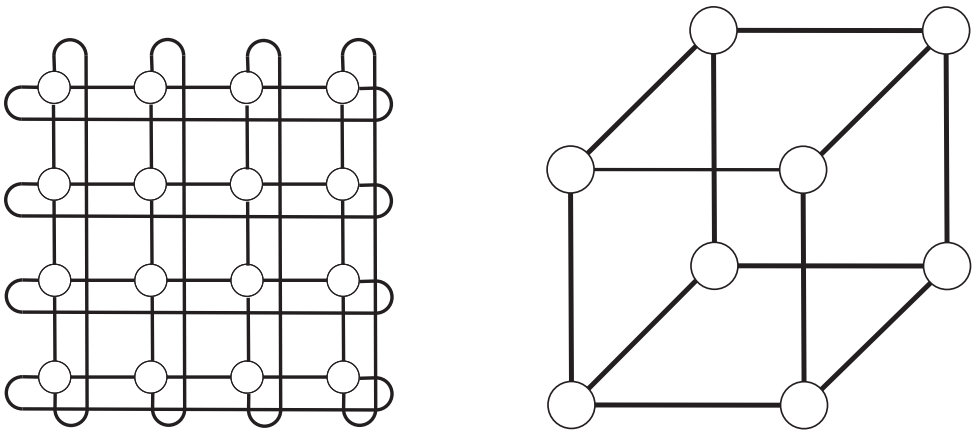
Wraz ze zwiększaniem się liczby procesorów w systemie równoległym, istotna staje się topologia połączeń pomiędzy elementami składowymi systemu równoległego (np. pomiędzy procesorami a bankami pamięci operacyjnej). Liczba i sposób połączeń jest ważnym czynnikiem wpływającym na wydajność i charakterystykę działania całego systemu równoległego. Sieci połączeń fizycznych dzielone są na dwa rodzaje [51, 177]:

- statyczne — realizowane jako bezpośrednie połączenia pomiędzy elementami sieci,
- dynamiczne — realizowane przez urządzenia pośredniczące, tworzące połączenie (*ang.* link) na żądanie.

Sieć fizyczna jest realizacją topologii sieci komunikacyjnej łączącej elementy systemu równoległego. Kształt oraz topologia sieci jest elementem bezpośrednio wpływającym na wydajność, skalowalność oraz na koszt systemu równoległego. Do najprostszych topologii możemy zaliczyć topologię magistrali (*ang.* bus). Jej zaletą jest bezpośrednie połączenie pomiędzy elementami sieci. Zasadniczą wadą jest sama współdzielona magistrala znacznie ograniczająca przepustowość, a przez to skalowalność systemu. Innym typem połączeń są połączenia krzyżowe (*ang.* crossbar). Zaletą stosowania połączeń typu krzyżowego jest uzyskiwanie połączeń nieblokujących się nawzajem. Jednak znaczącą wadą jest niska skalowalność rozwiązania, wynikająca z bardzo dużych kosztów tworzenia przełącznic krzyżowych o dużej liczbie elementów. Do bardziej zaawansowanych topologii połączeń należą połączenia typu każdy-z-każdym, połączenia w gwiazdę, różnego typu siatki (*ang.* mesh) — z pętlami oraz bez pętli, hiperkostki (*ang.* hyper-cube) oraz połączenia oparte na drzewach (zob. rys. 3.3).

Każdą z topologii charakteryzują pewne wielkości pozwalające opisać ich właściwości. Do podstawowych charakterystyk zalicza się [51, 77, 177]:

- średnicę definiowaną jako maksymalna najkrótsza ścieżka łącząca jakiekolwiek dwa elementy w sieci,
- stopień definiowany jako maksymalna liczba połączeń do każdego elementu sieci,
- koszt definiowany jako całkowita liczba połączeń pomiędzy elementami sieci.



(a) (b)

Rysunek 3.3: Przykładowe topologie połączeń: (a) siatka 2D z pętlami, (b) hiperkostka 3D.

Wartości charakterystyk dla kilku przykładowych topologii zawiera tabela 3.1. Jak można wywnioskować, najszybsze sieci zbudowane są w topologii każdy-z-każdym, jednak nie są to często spotykane instalacje ze względu na koszt związany z liczbą połączeń pomiędzy elementami. Wykorzystanie topologii oferujących mniejszy koszt powoduje pojawienie się zagadnień związanych z np. optymalnym rozdziałem zadań pomiędzy procesory, tak aby komunikacja między procesorami odbywała się po możliwie najkrótszej ścieżce.

3.3.4 Klastry

Do niedawna do najszybszych środowisk obliczeniowych należały superkomputery masywnie równoległe (*ang.* Massive Parallel Processors). MPP [25] są to systemy, w których większość węzłów tworzących maszynę posiada jedynie procesory oraz pamięć. W systemie znajdują się również węzły posiadające inne niezbędne komponenty, np. dyski twarde. Węzły wewnątrz komputera klasy MPP połączone są wydajną specjalizowaną siecią wewnętrzną. Koszt zaprojektowania i produkcji tak

Tabela 3.1: Charakterystyki wybranych topologii sieci o p elementach.

topologia	średnica	stopień	koszt
każdy z każdym	1	$p-1$	$p(p-1)/2$
gwiazda	2	1	$p-1$
pełne drzewo binarne	$2\log((p+1)/2)$	3	$p-1$
siatka 2-D bez pętli	$2(\sqrt{p}-1)$	4	$2(p-\sqrt{p})$
hiperkostka	$\log_2 p$	$\log_2 p$	$(p\log_2 p)/2$

wysoce wydajnych maszyn, składających się nierzadko z elementów zaprojektowanych specjalnie na potrzeby danej maszyny, jest bardzo wysoki.

Wysoki koszt zmusił do poszukiwania alternatywnych rozwiązań. Jedno z nich wypracowano w projekcie Beowulf [217] rozpoczętym w NASA Center of Excellence in Space Data and Information Sciences pod koniec 1993 roku. Jego główną ideą było zbudowanie równoległego środowiska obliczeniowego ze zwykłych komputerów klasy PC. Początkowo moc tak zainstalowanego systemu nie była duża, lecz wraz z rozwojem techniki stosunek mocy obliczeniowej do ceny maszyn stawał się coraz bardziej korzystny na rzecz klastrów typu Beowulf w stosunku do znanych komputerów typu MPP (np. nCube, CM5, Cray T3D).

Szybkie zdobycie popularności systemów obliczeniowych klasy Beowulf zaowocowało coraz większą specjalizacją tego typu systemów. Przestały one być już tylko środowiskami edukacyjnymi w ośrodkach naukowych, a stały się pełnoprawnymi środowiskami obliczeniowymi w instytucjach badawczych. Wraz z wykorzystywaniem klastrów, do coraz poważniejszych obliczeń zaczęto stosować coraz bardziej profesjonalne elementy składowe. Oprócz używanych początkowo procesorów klasy x86, wykorzystuje się również architektury PowerPC, UltraSPARC, Itanium, Opteron [219]. Rozwojowi uległy również sieci łączące węzły w klastrze. Początkowo stosowano zwykłe sieci Ethernet/FastEthernet łączone przy pomocy koncentratorów i przełączników. Obecnie wysokowydajne klastry obliczeniowe łączone są sieciami Gigabit Ethernet oraz specjalizowanymi połączniami typu Infiniband lub Myrinet. Gwałtowny rozwój technologii związanych z klastrami spowodował, że znacząca większość instalacji publikowanych na liście TOP500 [219] to właśnie klastry.

3.4 Projektowanie algorytmów równoległych

Projektowanie algorytmów równoległych wymaga znacznie więcej uwagi oraz doświadczenia niż projektowanie klasycznych algorytmów sekwencyjnych. Konieczność przesyłania danych pomiędzy równoległymi zadaniami oraz potrzeba ich synchronizacji stawia wysokie wymagania już na etapie projektowania algorytmu. Dodatkowe problemy pojawiają się w momencie analizy działającego algorytmu. Algorytmy równoległe rzadko należą do algorytmów deterministycznych, w których od razu można przewidzieć kolejne kroki w działaniu algorytmu. Trudności związane z za-

kleszczaniem się zadań oraz dostępem do współdzielonych zmiennych są typowymi problemami jakie należy rozwiązywać podczas opracowywania i testowania algorytmów równoległych.

Obliczenia równoległe definuje się jako zastosowanie dwóch lub więcej jednostek obliczeniowych do rozwiązania pojedynczego problemu obliczeniowego [198]. Jednostką obliczeniową może być proces, jeżeli poruszamy się w warstwie oprogramowania, lub fizyczny procesor w przypadku definicji z poziomu warstwy sprzętowej systemu równoległego.

3.4.1 Identyfikacja obszarów dekompozycji

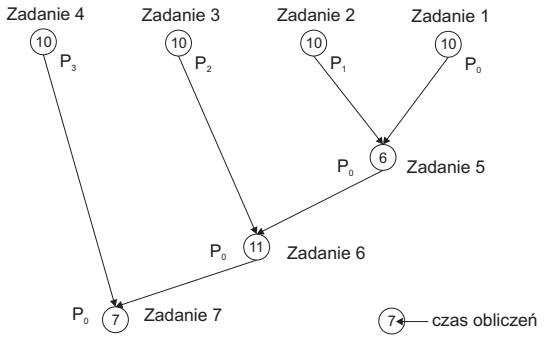
Proces projektowania algorytmów równoległych składa się z kilku etapów. Pierwszym z nich jest określenie obszarów algorytmu jakie mogą zostać zdekomponowane na zadania w celu jego zrównoleglenia [48]. Określenie obszarów podlegających zrównoleglaniu wymaga dogłębnej znajomości rozwiązywanego problemu oraz pewnego doświadczenia. Wybór odpowiedniej części algorytmu jaka ma ulec zrównolegleniu implikuje zarówno sposób działania algorytmu równoległego, jak również miejsca oraz momenty w których należy przesyłać informacje pomiędzy procesami wykonującymi zdefiniowane zadania.

3.4.2 Strategia dekompozycji

Po określeniu możliwych obszarów zrównoleglenia należy przejść do etapu dekompozycji algorytmu/problemu na mniejsze części/zadania, które mogą być rozwiązywane niezależnie. Liczba zadań na jaką podzielony zostaje rozważany problem określa ziarnistość (*ang.* granularity) dekompozycji. Podział problemu na wiele małych zadań określa się jako dekompozycję drobnoziarnistą (*ang.* fine-grained decomposition), a utworzenie mniejszej liczby większych zadań opisuje dekompozycję gruboziarnistą (*ang.* coarse-grained decomposition). Stopień ziarnistości dekompozycji ma wpływ np. na częstość komunikacji pomiędzy procesami wykonującymi poszczególne zadania. W literaturze wyróżnia się wiele technik dekompozycji, jednak najczęściej spotykany jest podział na dekompozycję danych (*ang.* data decomposition/parallelism) oraz dekompozycję proceduralną (*ang.* procedural/task decomposition/parallelism) [198].

Dekompozycja proceduralna pojawia się, gdy można podzielić algorytm lub jego część na niezależne segmenty, które mogą być uruchomione jako osobne zadania i nie jest wymagana komunikacja pomiędzy nimi. Tego typu dekompozycję nazywa się czasami w literaturze *trywialną* [198]. Takie określenie wynika z jej potencjalnej prostoty, jednak może ona być częścią składową bardziej zaawansowanego mechanizmu zrównoleglania tworzącego skomplikowaną sieć zależności pomiędzy zadaniami. Do opisu tak zdekomponowanego problemu najczęściej stosuje się grafy zależności pomiędzy utworzonymi zadaniami (zob. rys. 3.4).

Dekompozycję danych stosuje się, gdy algorytm operuje na dużej ilości danych



Rysunek 3.4: Przykładowy graf zależności pomiędzy zadaniami.

niezależnych od siebie. Podział danych na mniejsze części i wykonanie na nich operacji przez niezależne zadania w założeniu ma prowadzić do przyspieszenia obliczeń.

Bardzo często dekompozycja danych jest związana z zagadnieniem zrównoleglania pętli czyli dekompozycji przestrzeni iteracji (*ang.* iteration space decomposition) [198]. W algorytmach bardzo często występują zagnieżdżone pętle, które wykonują operacje na danych nie zmieniających się podczas wykonywania pętli. Podział przestrzeni iteracji na rozłączne podzbiory pozwala na wykonanie na nich obliczeń niezależnie w równoległych zadaniach. W zależności od tego czy podziałowi ulega jedna pętla czy też wiele zagnieżdżonych pętli, mamy do czynienia z dekompozycją warstwową (*ang.* strip/slab decomposition) lub blokową (*ang.* block decomposition). Czasami wykorzystuje się dekompozycję cykliczną (*ang.* cyclic decomposition), w której kolejne iteracje są cyklicznie przydzielane kolejnym zadaniom.

3.4.3 Model programowania

Kolejnym etapem jest wybór modelu programowania. Jest on ściśle uzależniony od systemu równoległego na jakim będą przeprowadzane symulacje zaprojektowanego algorytmu. Model programowania z wykorzystaniem pamięci współdzielonej pozwala wszystkim procesorom składającym się na system równoległy uzyskiwać dostęp bezpośrednio do wszystkich danych zawartych w pamięci systemu. W przypadku modelu programowania opartego na przesyłaniu komunikatów, wymiana danych pomiędzy procesami wykonującymi poszczególne zadania wymaga przesłania ich w postaci komunikatu o odpowiedniej strukturze. Na tym etapie również należy wybrać model komunikacji pomiędzy procesami np. „nadzorca-podwładni" (*ang.* master-slave), „każdy z każdym" (*ang.* peer-peer), „producent-konsument" (*ang.* producent-consumer) lub model hybrydowy. Na etapie wyboru modelu programowania należy również zaadaptować metody prowadzące do zmniejszenia narzutu czasowego związanego np. z przesyłaniem danych pomiędzy zadaniami [77]. Struktura algorytmu powinna prowadzić do operowania na maksymalnie dużej liczbie danych zdefiniowanych lokalnie dla każdego z zadań. Rzadsze przesyłanie mniejszej ilości danych pozwala na zmniejszenie narzutu czasowego związanego z komunikacją

między zadaniami. W przypadku konieczności przesłania danych pomiędzy zadaniami należy maksymalizować użycie mechanizmów komunikacji grupowej. Środowiska programowania równoległego posiadają bowiem zaimplementowane i zoptymalizowane funkcje pozwalające na przesyłanie danych pomiędzy wieloma zadaniami.

3.5 Warstwa oprogramowania

Jednym z kluczowych momentów w procesie projektowania aplikacji równoległych jest wybór środowiska oraz narzędzi, które będą wykorzystane do stworzenia działającej aplikacji. Równoległe środowiska obliczeniowe oferują zazwyczaj całą gamę aplikacji, bibliotek oraz narzędzi programistycznych.

Model systemu równoległego opartego na architekturze klastra możemy podzielić na następujące warstwy:

- warstwa fizyczna (architektura sprzętowa),
- warstwa komunikacyjna,
- warstwa pośrednia (*ang.* middleware),
- warstwa środowiska programowania równoległego,
- warstwa aplikacji użytkowych.

Na warstwę fizyczną składa się przede wszystkim architektura procesorów oraz pamięć dostępna w każdym z węzłów. Warstwa komunikacyjna określa sposób oraz topologię połączeń pomiędzy węzłami w klastrze (zob. rozdział 3.3).

3.5.1 Warstwa pośrednia

Do warstwy pośredniej zaliczamy system operacyjny wraz ze wszystkimi narzędziami systemowymi potrzebnymi do zarządzania klastrem. Największą popularność wśród systemów operacyjnych uzyskały tu rozmaite dystrybucje systemu operacyjnego Linux. Niezaprzeczalne zalety tego systemu są następujące:

- publikowanie na zasadach licencji GNU (ogólnie dostępny, darmowy),
- zgodność ze standardem POSIX,
- dostępność na większości architektur sprzętowych (x86, x86-64, IA-64, SPARC, PowerPC, POWER6),
- niskie zapotrzebowanie na zasoby systemowe (moc obliczeniową, pamięć operacyjną),
- duża liczba ogólnie dostępnych aplikacji i bibliotek.

Pomimo, że znacząca większość systemów równoległych posiada zainstalowany system operacyjny klasy Unix (Linux, MacOS X, AIX), to jednak pojawiają się instalacje oparte na systemach firmy Microsoft, np. Windows Compute Cluster Server 2003 czy Windows HPC 2008 [246]. Niewielka liczba ich instalacji spowodowana jest przede wszystkim dużym zapotrzebowaniem systemu operacyjnego na zasoby systemowe oraz kosztami związanymi z licencjami.

Jako element składowy warstwy pośredniej można również uważać system kolejkowy, który umożliwia uruchamianie oraz zarządzanie zadaniami zgodnie z polityką przyjętą przez administratorów klastra. Do najpopularniejszych systemów kolejkowych można zaliczyć OpenPBS/Torque [34], IBM LoadLeveler [93], LSF [86], Sun Grid Engine (SGE) [207], Condor [37].

3.5.2 Środowiska programistyczne

Do najpopularniejszych standardów środowisk programowania równoległego należy niewątpliwie standard MPI (*ang.* Message Passing Interfejs). Zdobył on znaczącą przewagę nad rozpowszechnionym środowiskiem PVM (*ang.* Parallel Virtual Machine). Standard interfejsu przesyłania komunikatów MPI przedstawiono w 1993 roku na konferencji *Supercomputing'93* w Portland, USA [143]. Przyjęto, że narzędzia zgodne ze standardem MPI muszą spełniać następujące wymagania podstawowe [80, 81, 144]:

- powinny mieć możliwość udostępniania zarówno komunikacji bezpośredniej (punkt-punkt), jaki i komunikacji grupowej,
- powinny być dostępne dla różnych architektur sprzętowych,
- powinny udostępniać interfejs dla języków programowania C/C++ oraz Fortran,
- powinny posiadać spójny interfejs, niezależny od języka programowania oraz architektury sprzętowej,
- powinny umożliwiać tworzenie rozszerzeń interfejsu.

W 1998 roku ukończono opracowywanie standardu MPI-2, w którym wprowadzono m.in. obsługę wątków. Do najbardziej popularnych bibliotek implementujących interfejs MPI należą MPICH [140], mpich2 [146], LAM-MPI [122] oraz Open MPI [161]. Wraz z powstaniem i wprowadzeniem środowisk gridowych udostępniono bibliotekę MPICH-G2 [145], która wspiera uruchamianie aplikacji równoległych w środowiskach gridowych.

Środowiska oparte na modelu przesyłania komunikatów nie są jedynymi środowiskami programowania równoległego. Odmienne podejście do problemu zrównoleglania zadań zostało zastosowane w środowisku OpenMP [162]. OpenMP jest specyfikacją dyrektyw kompilatora, bibliotek oraz zmiennych środowiskowych stworzonych na potrzeby programowania systemów równoległych z pamięcią współdzieloną

z wykorzystaniem wątków. Wprowadzenie OpenMP miało na celu uproszczenie mechanizmu programowania równoległego, tak aby programista nie musiał posiadać szczegółowej wiedzy na temat tworzenia, niszczenia oraz synchronizacji wątków.

3.5.3 Narzędzia wspomagające

Wybierając system komputerowy na którym będą wykonywane obliczenia, użytkownik ma rzadko możliwość wyboru i dowolnego konfigurowania architektury sprzętowej klastra, wyboru warstwy komunikacyjnej czy warstwy pośredniej. W ramach dostępu do systemu obliczeniowego uzyskuje się możliwość pracy na pewnej liczbie procesorów oraz możliwość zajęcia wyspecyfikowanej wielkości pamięci przez zadany z góry czas. Wybór bibliotek, pakietów numerycznych oraz narzędzi jest jednym z podstawowych elementów mających wpływ na czas powstawania aplikacji oraz uzyskiwane wyniki. Korzystanie z ogólnodostępnych oraz komercyjnych narzędzi wspomagających obliczenia numeryczne pozwala na znaczące przyspieszenie tego procesu oraz na zmniejszenie liczby błędów w przypadku samodzielnej implementacji skomplikowanych algorytmów numerycznych. Najczęściej spotykanymi pakietami matematycznymi w środowiskach klastrowych są Matlab, Maple, Mathematica, Abaqus, Scilab, Octave. Oprócz kompletnych środowisk wspomagających rozwiązywanie rozważanych problemów, możemy wykorzystać biblioteki numeryczne udostępniające zbiory funkcji do wykorzystania we własnych aplikacjach. Do najbardziej znanych bibliotek należą produkty dostępne bez opłat: BLAS [19], LAPACK [123], FFTW [62], PETSc [175], ATLAS [5], oraz produkty komercyjne: Intel® Math Kernel Library (Intel® MKL) [97], NAG [155] i inne. Istotny wpływ na wydajność tworzonej aplikacji ma zastosowany kompilator oraz powiązane z nim narzędzia. Kompilatory oraz biblioteki numeryczne dostarczane przez producentów systemów równoległych pozwalają na efektywniejsze wykorzystanie zasobów mocy obliczeniowej ponieważ narzędzia te są zaprojektowane tak, aby wykorzystać specyficzne własności architektury sprzętowej systemu równoległego.

3.6 Programowanie z wykorzystaniem przesyłania komunikatów

Systemy równoległe z pamięcią rozproszoną składają się z ze zbioru jednostek obliczeniowych, z których każda posiada bezpośredni dostęp tylko do własnej lokalnej pamięci. Potrzeba operacji na danych znajdujących się w zasobach lokalnych innej jednostki obliczeniowej wymaga komunikacji pomiędzy jednostkami obliczeniowymi w celu wymiany danych. Wymiana danych pomiędzy procesorami opiera się na zasadzie wymiany komunikatów posiadających specyficzne atrybuty.

3.6.1 Komunikacja

Wymiana informacji pomiędzy jednostkami obliczeniowymi posiadającymi lokalne zasoby pamięciowe implikuje rozważenie wielu możliwych strategii wymiany informacji. Jednym z podstawowych podziałów jest podział ze względu na liczbę jednostek obliczeniowych zaangażowanych w wymianę danych. Wyróżnić można komunikację:

- jeden do jednego (*ang.* point-to-point) — w komunikację zaangażowane są dwa zadania bezpośrednio wymieniające między sobą komunikaty zawierające dane;
- jeden do wielu (*ang.* broadcast) — jedno z zadań rozsyła komunikat do wszystkich lub do zdefiniowanej grupy zadań;
- wiele do wielu (*ang.* all-to-all) — następuje grupowa wymiana informacji pomiędzy zadaniami;
- wiele do jednego (*ang.* gather) — dane z wielu zadań przesyłane są do jednego z nich; wariantem tego typu komunikacji jest operacja redukcji, podczas której, oprócz przesłania danych do jednego z węzłów, następuje wykonanie zadanej operacji, np. sumowanie odebranych elementów.

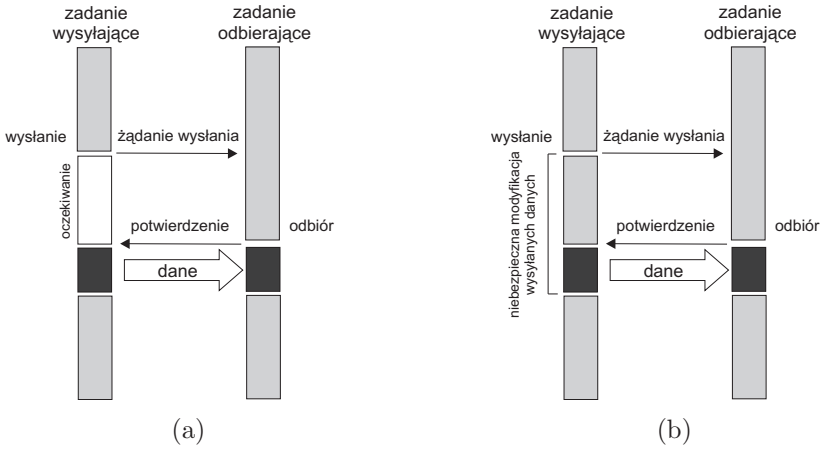
Oprócz podziału ze względu na liczbę zadań wysyłających i odbierających, można wyróżnić grupowanie ze względu na synchronizację komunikacji [177]. Podział przedstawia się następująco:

- komunikacja synchroniczna — zadanie wysyłające informacje oczekuje na potwierdzenie otrzymania danych przez zadanie odbierające;
- komunikacja asynchroniczna — zadanie wysyła dane w dogodnym dla siebie momencie, a zadanie odbierające przyjmuje je w dogodnej dla siebie chwili; w tym czasie dane przechowuje się w buforach.

Innym istotnym podziałem sposobu komunikacji jest podział ze względu na zachowanie sterowania w zadaniu wysyłającym lub odbierającym komunikat. Wyróżniamy:

- operacje blokujące — polegają na wstrzymaniu wykonywania operacji przez zadania wykonujące operację (wysyłanie/odbieranie) do czasu jej zakończenia; operacje blokujące mogą powodować powstawanie zakleszczeń (*ang.* deadlock) zadań;
- operacje nieblokujące — zadanie nie jest przerywane po inicjalizacji funkcji wysyłającej/odbierającej; do momentu zakończenia operacji wysyłania/odbierania danych nie powinno się wykonywać jakichkolwiek operacji na przesyłanych/odbieranych danych.

Oprócz przedstawionych kryteriów podziałów spotyka się również podział ze względu na występowanie buforowania przesyłanych danych oraz wykorzystanie sprzętowych mechanizmów buforowania. Przykład komunikacji niebuforowanej blokującej oraz nieblokującej przedstawiono na rys. 3.5 [77].



Rysunek 3.5: Przykład wymiany komunikatu: (a) niebuforowana operacja blokująca, (b) niebuforowana operacja nieblokująca.

Rysunek 3.5(a) przedstawia niebuforowaną operację blokującą, w której zadanie wysyła żądanie przesłania danych do innego zadania. W momencie wysłania żądania wysłania danych jego praca zostaje wstrzymana do momentu otrzymania potwierdzenia możliwości wysłania danych. Po przesłaniu danych do zadania odbierającego, praca obydwu zadań jest kontynuowana. Newralgicznym momentem w tego typu komunikacji jest czas oczekiwania na synchronizację dwóch zadań, umożliwiającą przesłanie danych. Rysunek 3.5(b) pokazuje sposób działania niebuforowanej operacji nieblokującej. Zasadniczą różnicą w stosunku do operacji blokującej jest moment wysłania żądania przesłania danych do zadania odbierającego, po którym następuje powrót sterowania do zadania umożliwiający dalsze operacje przez zadanie wysyłające. Po otrzymaniu potwierdzenia przez zadanie odbierające następuje przesłanie danych pomiędzy zadaniami. Ważnym elementem w przypadku tego typu operacji jest zadbanie o spójność danych, które mają być przesłane. Modyfikacja tych danych może spowodować ich niespójność oraz późniejsze przetwarzanie błędnych informacji.

3.6.2 Standard interfejsu MPI

Obecnie najczęściej spotykanym sposobem programowania systemów równoległych z pamięcią rozproszoną (tj. klastrów obliczeniowych) jest zastosowanie bibliotek implementujących interfejs MPI (*ang.* Message Passing Interface — zob. rozdz. 3.5.2). MPI unifikuje sposób pisania programów zarówno w języku C/C++, jak i Fortran.

Funkcje dostarczane przez biblioteki zgodne z MPI posiadają ten sam interfejs programowy (niezależnie od języka programowania). Olbrzymią zaletą interfejsu MPI jest jego prostota. Wprawdzie MPI dostarcza ponad 100 funkcji, jednak napisanie większości programów wymaga znajomości jedynie sześciu z nich. Do podstawowych funkcji należą [80, 81]:

- funkcje odpowiedzialne za włączanie i wyłączanie mechanizmów MPI (MPI_INIT, MPI_FINALIZE),
- funkcje odpowiedzialne za identyfikację uruchomionego procesu (MPI_COMM_SIZE, MPI_COMM_RANK),
- funkcje odpowiedzialne za przesłanie danych pomiędzy procesami (np. MPI_SEND, MPI_RECV).

Ważnym elementem jest również funkcja MPI_BARRIER, odpowiedzialna za synchronizację procesów (mechanizm bariery). Do grupy funkcji odpowiedzialnych za przesyłanie informacji należą również funkcje przesyłające dane pomiędzy wieloma procesami, np. MPI_BCAST, MPI_REDUCE, MPI_GATHER, MPI_SCATTER, MPI_ALLTOALL. Osobną grupę funkcji stanowią funkcje odpowiedzialne za przydział procesów do grup i komunikatorów oraz funkcje tworzące topologie połączeń pomiędzy procesami. Funkcje te pozwalają na rozdział przydzielonych jednostek obliczeniowych pomiędzy różnego typu zadania i kontrolowane przesyłanie danych pomiędzy nimi. Tworzenie topologii połączeń pozwala na wydajne przesyłanie danych dzięki odzwierciedleniu fizycznych połączeń pomiędzy jednostkami obliczeniowymi w warstwie aplikacji (zob. rozdz. 3.3.3).

Przykład 3.1 Przykład wykorzystania MPI
Celem programu w języku Fortran 95 oznaczonego jako Algorytm 3.1 jest przesłanie pojedynczej wartości liczbowej typu całkowitego między procesami identyfikowanymi etykietami 0 i 1 [48]. W liniach 1–4 następuje rozpoczęcie programu, dołączenie biblioteki *mpi* oraz deklaracja zmiennych na potrzeby programu. W zmiennych *bufin* oraz *bufout* przechowywane będą przesyłane wartości zmiennych. Zmienna *wielkosc* przechowuje liczbę uruchomionych kopii programu (liczbę procesów), a zmienna *id* jednoznacznie identyfikuje dany proces spośród wszystkich procesów programu opisywanych przez komunikator *mpi_comm_world* (komunikator to pewnego rodzaju topologia wirtualna definiująca grupę procesów w obrębie których odbywa się komunikacja). Linie 6–9 inicjują interfejs MPI, pobierają liczbę działających kopii programu oraz identyfikator aktualnie wykonywanego procesu. W liniach 9–17, w zależności od identyfikatora, wykonywany jest kod programu wysyłający bądź odbierający dane. Następnie interfejs MPI zostaje wyłączony, a program zakończony (linie 18 i 19).

♦

Algorytm 3.1 Przykład programu w Fortranie 95 z zastosowaniem interfejsu MPI.

```
 1: program przyklad
 2: use mpi
 3: integer :: blad, wielkosc, id
 4: integer :: bufout, bufin, stat(mpi_status_size)
 5:
 6: call mpi_init ()
 7: call mpi_comm_size (mpi_comm_world, wielkosc, blad)
 8: call mpi_comm_rank (mpi_comm_world, id, blad)
 9: if (id == 0) then
10:    bufout = 10
11:    call mpi_send (bufout,1,MPI_INTEGER,1,0,mpi_comm_world,blad)
12: elseif (id == 1) then
13:    bufin = −1
14:    print *, 'Bufor przed odebraniem danych = ', bufin
15:    call mpi_recv (bufin,1,MPI_INTEGER,0,0,mpi_comm_world,stat,blad)
16:    print *, 'Bufor po odebraniu danych = ', bufin
17: endif
18: call mpi_finalize (blad)
19: end
```

■

3.7 Ocena algorytmów równoległych

Ocena algorytmów równoległych pod względem ich wydajności wymaga znacznie dokładniejszej analizy w porównaniu z algorytmami sekwencyjnymi. W tych ostatnich w dość prosty sposób można przeanalizować złożoność obliczeniową na podstawie szkicu algorytmu czy też poprzez zbadanie liczby wykonanych operacji podstawowych. W algorytmach równoległych krytycznymi stają się czynniki związane z synchronizacją danych, równoważeniem obciążenia czy opóźnieniami transmisji. Uwzględnienie wszystkich czynników wpływających na pracę algorytmu równoległego jest zazwyczaj bardzo trudne.

3.7.1 Miary jakości algorytmów równoległych

Aby móc porównywać jakość algorytmów równoległych względem ich implementacji w postaci algorytmów sekwencyjnych, należy określić kryteria porównawcze. Do najistotniejszych i najczęściej spotykanych miar jakości należą [108, 169, 198]:

- przyspieszenie,

- efektywność,
- ziarnistość,
- wydajność,
- udział części sekwencyjnej.

Najważniejszym kryterium porównań algorytmów oraz systemów równoległych jest całkowity czas wykonywania obliczeń. Jest on mierzony od momentu rozpoczęcia pracy przez pierwszy procesor w systemie równoległym do zakończenia obliczeń przez ostatni procesor. Całkowity czas obliczeń określony jest wyrażeniem [108]

$$T_C = T_S + T_P, \tag{3.1}$$

gdzie:

T_C — całkowity czas obliczeń obserwowany przez końcowego użytkownika,
T_S — czas wykonywania sekwencyjnej części algorytmu,
T_P — czas wykonywania równoległej części algorytmu.

Ze względu na sposób działania równoległej części algorytmu, czas T_P możemy z kolei przedstawić jako sumę

$$T_P = T_p + T_c + T_i, \tag{3.2}$$

gdzie:

T_p — całkowity czas pracy procesorów w równoległej części algorytmu,
T_c — opóźnienie spowodowane komunikacją pomiędzy procesorami,
T_i — opóźnienie wynikające z bezczynności procesorów przy nierównomiernym ich obciążeniu obliczeniami.

Kolejne miary jakości przedstawione są w oparciu o założenie, że rozważa się problemy równoległe o stałym rozmiarze n, czyli takie problemy, których rozmiar nie jest funkcją liczby procesorów

$$n(p) = const, \tag{3.3}$$

gdzie: $n(p)$ — rozmiar problemu w zależności od liczby procesorów p. Jak zostanie pokazane w rozdz. 3.7.3, rozmiar danych jakimi operuje algorytm równoległy, ma zasadniczy wpływ na jego efektywność oraz sprawność. Dodatkowym warunkiem, powodującym uproszczenie notacji, jest założenie o równomiernej dekompozycji algorytmu sekwencyjnego, co oznacza, że każda z jednostek obliczeniowych wykonuje ilościowo taką samą porcję obliczeń. Bardziej szczegółowe rozważania dotyczące algorytmów bez równoważenia obciążenia przedstawiono w rozdz. 4.5.3.

W celu przedstawienia kolejnych miar ocen algorytmów równoległych, wprowadźmy następujące oznaczenia:

- p — liczba procesorów,
- $T_C(p)$ — całkowity czas pracy algorytmu równoległego uruchomionego na p procesorach,

Przyspieszenie

Najczęściej stosowaną miarą oceny jakości algorytmu równoległego jest przyspieszenie [169]

$$S(p) = \frac{T_C(1)}{T_C(p)}, \tag{3.4}$$

Oznacza ono względne przyspieszenie uzyskane dzięki uruchomieniu algorytmu na p procesorach w stosunku do czasu uzyskanego przy uruchomieniu tego samego algorytmu na jednym procesorze. Dokładniej, tak przedstawiona miara przyspieszenia jest określana jako *przyspieszenie względne* (*ang.* relative speedup). Alternatywnie, nieco rzadziej stosowana miara *przyspieszenia rzeczywistego* (*ang.* real speedup) przedstawia stosunek czasu wykonywania się najlepszego znanego algorytmu sekwencyjnego do czasu wykonania algorytmu równoległego na p procesorach, rozwiązujących identyczny problem obliczeniowy. W idealnej sytuacji, zachodzi $S(p) \approx p$, co oznacza, że procesory są równomiernie obciążone, a czas komunikacji między nimi oraz czas wykonywania części sekwencyjnej jest pomijalny.

Efektywność

Efektywność jest miarą przedstawiającą przyspieszenie przeskalowane w stosunku do idealnego przyspieszenia liniowego $S(p) = p$:

$$E(p) = \frac{T_C(1)}{p\,T_C(p)} \cdot 100\% = \frac{S(p)}{p} \cdot 100\%. \tag{3.5}$$

Wskaźnik ten określa w jaki sposób algorytm zachowuje się przy zastosowaniu różnej liczby procesorów. Efektywność algorytmu maleje wraz z wykorzystaniem większej liczby procesorów chociażby ze względu na większy narzut czasowy związany z komunikacją pomiędzy procesorami.

W szczególnych przypadkach można spotkać algorytmy, które posiadają własność

$$S(p) > p, \quad E(p) > 100\%. \tag{3.6}$$

Algorytm o takiej charakterystyce posiada tzw. przyspieszenie super-liniowe (*ang.* super-linear speedup). Jest to dość rzadko spotykana sytuacja, która może występować gdy algorytm uruchomiony na pojedynczej jednostce obliczeniowej powoduje maksymalne wykorzystanie jej pamięci operacyjnej. W takiej sytuacji system może zacząć korzystać np. z pliku wymiany (*ang.* swap file), czyli pamięci o znacznie wolniejszym czasie dostępu. W przypadku uruchomienia rozważanego algorytmu w środowisku równoległym, oprócz dodatkowych jednostek obliczeniowych, które przyspieszają obliczenia, system udostępnia większe zasoby pamięci operacyjnej, wstrzymując tym samym konieczność korzystania z pamięci o wolniejszym czasie dostępu (plik wymiany). Przyspieszenie super-liniowe można również uzyskać w sytuacji, gdy program uruchomiony na pojedynczej jednostce obliczeniowej wykorzystuje pamięć cache procesora oraz pamięć operacyjną komputera. Pamięć cache procesora,

ze względu na swoją budowę, jest znacznie szybsza od pamięci operacyjnej. Konieczność ciągłej synchronizacji danych pomiędzy pamięcią cache procesora a pamięcią operacyjną powoduje pewne opóźnienia. Podczas uruchomienia algorytmu w środowisku równoległym możemy mieć do czynienia z przypadkiem, gdy z powodu dekompozycji problemu na wiele procesorów, kod programu oraz dane znajdą się w całości w pamięciach cache procesorów. Taka sytuacja może powodować uzyskanie większego przyspieszenia niż przyspieszenie liniowe wynikające z dekompozycji problemu na wiele procesorów.

Ziarnistość

Wskaźnik ziarnistości pozwala ocenić jak mocno czas pracy algorytmu jest zdeterminowany przez opóźnienia w komunikacji pomiędzy procesorami. Miarę tę definiujemy jako

$$G = \frac{T_C}{T_c}. \tag{3.7}$$

W ogólności, powinno się dążyć do maksymalizacji tej miary (w szczególności, gdy procesory połączone są wolną siecią komunikacyjną) aby narzut związany z komunikacją był jak najmniejszy.

Wydajność chwilowa

Często do porównywania różnych algorytmów rozwiązujących ten sam problem używa się kryterium wydajności chwilowej [87, 218]. Jej wartość stanowi odwrotność całkowitego czasu obliczeń algorytmu:

$$R = T_C^{-1}. \tag{3.8}$$

Algorytm o wyższej wartości wskaźnika wydajności wykonuje obliczenia w krótszym czasie i z punktu widzenia użytkownika końcowego pozwala na wydajniejsze przeprowadzanie obliczeń.

Udział części sekwencyjnej

Znakomita większość algorytmów równoległych posiada część sekwencyjną, która nie podlega zrównolegleniu. W celu uzyskania jak największego przyspieszenia obliczeń powinno się dążyć do zminimalizowania wpływu czasu pracy algorytmu w części sekwencyjnej na ogólny czas pracy algorytmu równoległego. Udział części sekwencyjnej definiuje się następująco:

$$f = \frac{T_S}{T_C}. \tag{3.9}$$

3.7.2 Prawo Amdahla

Rozważmy zrównoleglenie algorytmu sekwencyjnego, który składa się z części niepodzielnej (tzw. takiej, której nie można zrównoleglić) oraz części podlegającej zrównolegleniu (dla uproszczenia zakładamy równomierną dekompozycję problemu).

W oparciu o parametr f określający udział części sekwencyjnej, por. (3.9), czas wykonania obliczeń w systemie równoległym, składającym się z p jednostek obliczeniowych, można oszacować następująco [177]:

$$T_C(p) = f \cdot T_C(1) + \frac{(1-f) \cdot T_C(1)}{p}. \tag{3.10}$$

Przyspieszenie rozważanego algorytmu wyraża się więc wzorem

$$S(p) = \frac{T_C(1)}{T_C(p)} = \frac{1}{f + \frac{1-f}{p}}. \tag{3.11}$$

Wraz ze wzrostem liczby jednostek obliczeniowych p, posiada ono wartość graniczną

$$S(p) \underset{p\to\infty}{\longrightarrow} \frac{1}{f}. \tag{3.12}$$

Zależności (3.11)–(3.12) opisują tzw. **prawo Amdahla** [48, 51, 177, 198] które mówi, że przyspieszenie algorytmu równoległego jest uzależnione od części sekwencyjnej algorytmu równoległego i nie może wynieść więcej niż odwrotność stopnia udziału części niezrównoleglonej.

3.7.3 Skalowalność algorytmów równoległych

Prawo Amdahla oraz przedstawione miary jakości pozwalające scharakteryzować algorytm równoległy zakładają znaczne uproszczenie polegające na założeniu, że zależą one tylko i wyłącznie od skali systemu równoległego (dokładniej, od liczby jednostek obliczeniowych), pomijając wpływ wielkości problemu obliczeniowego. Jednak intuicyjne podejście do programowania równoległego zakłada, że równoległe systemy obliczeniowe o większej liczbie jednostek obliczeniowych są wykorzystywane nie tylko do szybszego uzyskiwania wyników symulacji, ale przede wszystkim do rozwiązywania bardziej złożonych problemów z większym rozmiarem danych. W zadaniach tego typu bardzo często czas pracy w części sekwencyjnej nie jest zależny od wielkości danych, w odróżnieniu od części równoległej, w której czas pracy jest proporcjonalny do wielkości danych [51, 177]. Uwzględniając powyższe, w takich sytuacjach można przyjąć, że udział części sekwencyjnej

$$f(n) \underset{n\to\infty}{\longrightarrow} 0, \tag{3.13}$$

gdzie: n — wielkość zadania obliczeniowego.

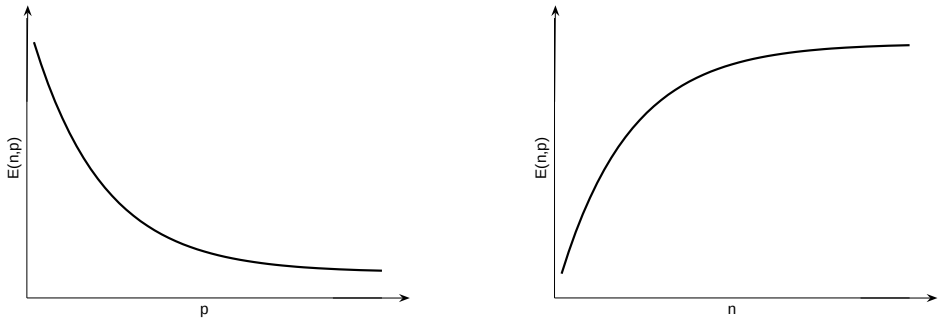
Uwzględniając zależność udziału części sekwencyjnej algorytmu od wielkości zadania (danych), przyspieszenie (3.11) modyfikuje się następująco:

$$S(n,p) = \frac{T_C(n,1)}{T_C(n,p)} = \frac{1}{f(n) + \frac{1-f(n)}{p}}, \tag{3.14}$$

co, biorąc pod uwagę (3.13), prowadzi do

$$S(n,p) \underset{n\to\infty}{\longrightarrow} p. \tag{3.15}$$

Dokładniej, zjawisko to przedstawiono na rys. 3.6. Dla problemu o stałym rozmiarze, efektywność E maleje wraz ze zwiększającą się liczbą jednostek obliczeniowych (zob. rys. 3.6(a)). W większości systemów równoległych zaobserwować można zwiększanie się efektywności algorytmu wraz ze zwiększaniem się rozmiaru problemu przy niezmienionej liczbie procesorów (zob. rys. 3.6(b)). Zależność (3.15) uzasadnia powód stosowania obliczeń równoległych do efektywniejszego rozwiązywania problemów obliczeniowych o coraz większej wymiarowości zamiast coraz szybszego rozwiązywania zagadnień małej skali.



(a) (b)

Rysunek 3.6: Efektywność systemu równoległego: (a) dla problemu o stałym rozmiarze, (b) dla stałej liczby jednostek liczących.

Poruszany problem jest ściśle powiązany z pojęciem **skalowalności** systemu równoległego. Jest to własność polegająca na elastycznym dostosowywaniu się systemu do zwiększającej się liczby jednostek obliczeniowych [177]. Niech elastyczność będzie definiowana przez sprawność $\eta(n,p)$ systemu równoległego, będąca funkcją zarówno wielkości zadania n, jak i liczby jednostek obliczeniowych p

$$\eta(n,p) = \frac{q(n)}{q(n) + h(n,p)} = \frac{1}{1 + \frac{h(n,p)}{q(n)}}, \tag{3.16}$$

gdzie:

$q(n)$ — liczba operacji obliczeniowych zależna od rozmiaru problemu n,
$h(n,p)$ — narzuty na komunikację pomiędzy p jednostkami obliczeniowymi.

Zwiększająca się liczba jednostek obliczeniowych p zwiększa narzut komunikacyjny (opóźnienia), co prowadzi do zmniejszania się sprawności systemu równoległego, w przypadku rozwiązywania problemu o niezmienionej skali. W przypadku zwiększania się rozmiaru rozwiązywanego problemu najczęściej obserwuje się znacznie zwiększenie liczby operacji związanych z obliczeniami niż zwiększenie się narzutu związanego z komunikacją, co prowadzi do zwiększenia sprawności systemu równoległego.

3.8 Zagadnienie równoważenia obciążenia

Intuicyjnie, dzięki równomiernemu podziałowi na zadania, proces dekompozycji problemu ma prowadzić do jednoczesnego i równotrwałego wykonywania zadań równoległych. Jednak bardzo często równomierny podział ilości informacji nie prowadzi do równoczesnego wykonania się poszczególnych zadań. Głównymi przyczynami są: opóźnienia związane z komunikacją pomiędzy zadaniami, czas oczekiwania przez procesy na synchronizację danych czy też niejednorodność środowiska równoległego, w którym przeprowadzane są obliczenia. Z jednej strony, częsta wymiana dużej ilości danych pomiędzy procesami prowadzi do odkładania żądań przesłania danych w kolejki, które są stopniowo opróżniane. Z drugiej strony, brak synchronizacji pomiędzy zadaniami prowadzi do sytuacji, gdy część procesów kończy swoje zadania wcześniej, a następnie muszą one oczekiwać na zakończenie obliczeń przez procesy wykonujące obliczenia dłużej. Metody równoważenia obciążenia (*ang.* load-balancing) powinny zapobiegać sytuacjom wymienionym powyżej poprzez

- redukcję czasu spędzanego przez procesy na interakcję pomiędzy nimi,
- redukcję całkowitego czasu bezczynności procesów związanego m.in. z ich synchronizacją.

Problem równoważenia obciążenia można sformułować następująco [198].

Definicja 3.1
Załóżmy, że zadania (numerowane jako $i = 1, \ldots, p$) ze zbioru zadań równoległych wykonują się w czasach t_i. Średni czas wykonania zadania wynosi

$$\operatorname{avg}\{t_i : 1 \leqslant i \leqslant p\} = \frac{1}{p} \sum_{i=1}^{p} t_i. \tag{3.17}$$

Współczynnik zrównoważenia obciążenia β definiuje się jako

$$\beta = \frac{\operatorname{avg}\{t_i : 1 \leqslant i \leqslant p\}}{\max\{t_i : 1 \leqslant i \leqslant p\}}. \tag{3.18}$$

Mówimy, że zbiór zadań jest zrównoważony pod względem obciążenia gdy

$$\beta \approx 1. \tag{3.19}$$

co odpowiada sytuacji, gdy średni czas jest równy maksymalnemu czasowi wykonywanych obliczeń. Względna różnica pomiędzy czasem średnim, a maksymalnym wynosi

$$1 - \beta = \frac{\max\{t_i : 1 \leqslant i \leqslant p\} - \text{avg}\{t_i : 1 \leqslant i \leqslant p\}}{\max\{t_i : 1 \leqslant i \leqslant p\}} \tag{3.20}$$

i przy zrównowaznonych obliczeniach powinna być pomijalnie mała.

Problem dynamicznego równoważenia obciążenia omawia się szeroko w wielu publikacjach [42, 78, 90, 114, 121, 142, 197]. W przypadku przedstawienia algorytmu równoległego w postaci grafu zależności pomiędzy zadaniami, zagadnienie równoważenia obciążenia definiuje się następująco.

Definicja 3.2
Przedstawmy zrównoleglany algorytm jako graf [42, 197]

$$\mathrm{G} = (\mathrm{Z}, \mathrm{E}), \tag{3.21}$$

gdzie:

Z — zbiór zrównoleglanych zadań,

E — zbiór zależności pomiędzy zadaniami.

Każde zadanie $z \in \mathrm{Z}$ składa się z podzbioru niepodzielnych operacji, które nie mogą być zrównoleglone. Na zbiór Q składają się wszystkie niepodzielne (atomowe) operacje całego rozważanego problemu, jakie chcemy przydzielić poszczególnym równoległym zadaniom. Dodatkowo, definiujemy funkcję przypisania $\pi : \mathrm{Q} \to \mathrm{Z}$, która przydziela każdą niepodzielną operację $q \in \mathrm{Q}$ równoległemu zadaniu $z \in \mathrm{Z}$.

Czas wykonywania zadania z w zależności od odwzorowania π wynosi

$$t_\pi(z) = \sum_{q \in \mathrm{Q}:\pi(q)=z} t(q, z), \tag{3.22}$$

gdzie: $t(q, z)$ — czas wykonania operacji niepodzielnej q w zadaniu z.

Zgodnie z Definicją 3.1, algorytm jest zrównoważony, gdy średni czas wykonywanych obliczeń jest równy maksymalnego czasowi wykonywanych obliczeń, czyli gdy wariancja czasów wykonywanych zadań jest bliska zeru. Zdefiniujmy globalną funkcję kosztu $\Gamma(\pi)$ jako wariancję czasów obliczeń poszczególnych zadań w zależności od odwzorowania π

$$\Gamma(\pi) = \frac{1}{|\mathrm{Z}|} \sum_{z \in \mathrm{Z}} \left(t_\pi(z) - \bar{t}_\pi\right)^2, \tag{3.23}$$

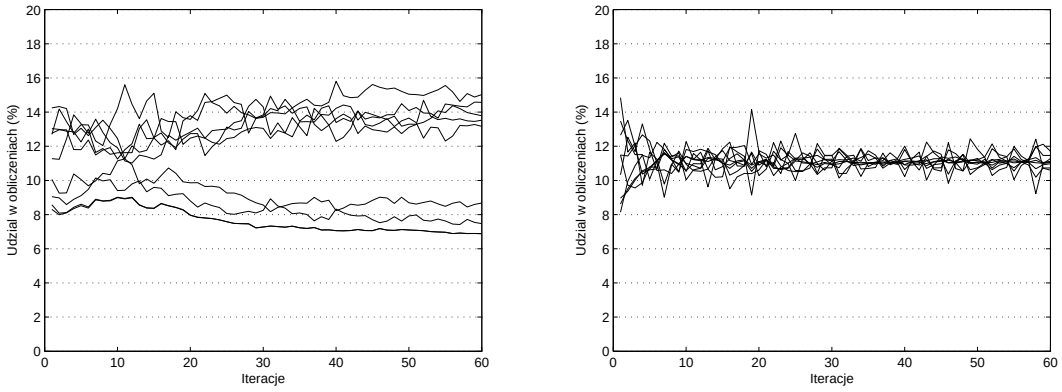
gdzie:

$$\bar{t}_\pi = \frac{1}{|\mathrm{Z}|} \sum_{z \in \mathrm{Z}} t_\pi(z). \tag{3.24}$$

Na podstawie (3.21)–(3.24) problem równoważenia obciążenia możemy sformułować następująco:

Znajdź odwzorowanie π, które minimalizuję globalną funkcję kosztu $\Gamma(\pi)$.

Algorytm równoległy powinien tak przydzielić operacje atomowe poszczególnym zadaniom aby zrównoważyć czas wykonywania obliczeń wszystkich zadań, tzn. aby koszt Γ (wariancja) dążył do zera (zob. rys. 3.7).



(a) (b)

Rysunek 3.7: Przykładowy czas wykonywania się równoległych zadań: (a) bez równoważenia obciążenia, (b) z równoważeniem obciążenia.

Badania przedstawione w niniejszej pracy można w dużym uproszczeniu określić jako problem zrównoleglania obliczeń wykonywanych w pętlach. Większa część algorytmów polega na przeszukiwaniu pewnej przestrzeni rozwiązań w celu znalezienia optymalnych wartości parametrów wymaganych przez kolejne kroki algorytmu. Problem zrównoleglania pętli nie posiadających zależności pomiędzy kolejnymi iteracjami jest jednym z klasycznych problemów zrównoleglania [48, 198]. Jednakże, prosty algorytm podziału przestrzeni poszukiwań (*ang.* domain decomposition) nie zawsze okazuje się efektywny. Działanie zrównoleglonego w ten sposób algorytmu jest zależne od wielu czynników. Przykładowo, czasy obliczeń poszczególnych iteracji mogą się różnić, a poszczególne węzły klastra mogą nie być identyczne, tworząc tzw. klaster heterogeniczny.

W literaturze można spotkać kilka metod rozwiązywania problemu optymalnego równoważenia obciążenia przy zrównoleglaniu niezależnych pętli. Wśród nich możemy wyróżnić: FAC (*ang.* factoring) [92], FRAC (*ang.* fractiling) [6], WF (*ang.* weight factoring) [91], AF (*ang.* adaptive factoring) [7], AWF (*ang.* adaptive weight factoring) [8, 193]. Przedstawione metody dzielą zbiór Q pomiędzy poszczególne zadania $z \in \mathrm{Z}$ (zob. def. 3.2).

Średnia liczba operacji atomowych jaką przydzielono równoległym zadaniom wynosi

$$\bar{q} = \frac{|\mathrm{Q}|}{|\mathrm{Z}|}. \tag{3.25}$$

Liczbę operacji jaką otrzyma zadanie z określa się wzorem

$$\hat{\eta}_z = w_z \bar{q}, \tag{3.26}$$

gdzie: w_z — współczynnik wagowy określany na podstawie czasu wykonania zadania z w przeszłości.

Do realizacji celów niniejszej pracy wykorzystano metodę adaptacyjnej ważonej faktoryzacji (AWF). AWF pozwala na określenie współczynników w_z po każdym kroku algorytmu. Na starcie współczynnik w_z dla każdego zadania z wynosi 1, czyli każde zadanie otrzymuje tę samą ilość danych. Nowe wagi są obliczane nie tylko na podstawie czasu wykonywania obliczeń przez procesory w poprzedniej iteracji, ale również bazują na całościowej informacji o czasie wykonywania operacji przez wszystkie procesory podczas wykonywania wszystkich wcześniejszych iteracji algorytmu. Takie podejście, w odróżnieniu od metod, gdzie zakłada się stały przydział ilości danych dla każdego z procesorów (metody FAC, FRAC, WF), czy też zakłada się pewne, dane *a priori*, odchylenie standardowe czasów wykonywania obliczeń przez poszczególne procesory (metoda AF) pozwala, nie tylko na reakcję metody równoważenia obciążenia na dynamicznie zmieniające się obciążenie procesorów, ale również pozwala na wykorzystanie informacji na temat całkowitego czasu wykonywania operacji przez wszystkie procesory, pozwalając uwzględnić pewien stały stosunek w niejednorodności mocy obliczeniowej procesorów.

W celu zademonstrowania działania metody rozważmy krok τ, w którym zadanie z wykonywało obliczenia na zbiorze danych o wielkości $n_z^{(\tau)}$ przez czas $t_z^{(\tau)}$. Współczynnik opisujący średni czas wykonania pojedynczej operacji atomowej przez zadanie z określa się jako

$$\gamma_z = \frac{\sum_{j=1}^{\tau} t_z^{(j)}}{\sum_{j=1}^{\tau} n_z^{(j)}}. \tag{3.27}$$

Konwersja współczynnika ξ_z do współczynnika wagowego zadania przeprowadzana jest następująco [8, 193]:

$$\bar{\gamma} = \frac{\sum_{z=1}^{|\mathrm{Z}|} \gamma_z}{|\mathrm{Z}|}, \quad \rho_z = \frac{\bar{\gamma}}{\gamma_z}, \quad \hat{\rho} = \sum_{j=z}^{|\mathrm{Z}|} \rho_z. \tag{3.28}$$

Następnie wagi zadań normalizuje się:

$$w_z = \frac{\rho_z |\mathrm{Z}|}{\hat{\rho}}. \tag{3.29}$$

Dzięki zastosowaniu opisanej procedury każde zadanie, które przeprowadzało obliczenia w poprzednich krokach dłużej od pozostałych zadań, w kolejnych krokach otrzymuje mniejsze ilości danych do przetwarzania.

3.8.1 Wpływ równoważenia obciążenia na efektywność

Przedstawione w rozdz. 3.7.1 miary jakości dotyczyły przypadku w którym zakładamy, że każda z jednostek obliczeniowych jest równomiernie obciążona przez algorytm. W przypadku, gdy jednostki obliczeniowe nie są równomiernie obciążone zakładamy, że i-te zadanie ($i = 1, \ldots, p$) ze zbioru zadań równoległych wykonuje się w czasie t_i. Czas pracy algorytmu równoległego definiujemy jako [198]

$$T_C(p) = \max\{t_i : 1 \leqslant i \leqslant p\}. \tag{3.30}$$

Zakładamy również, że algorytm równoległy nie ma przyspieszenia superliniowego, tzn.

$$T_C(1) \leqslant \sum_{i=1}^{p} t_i, \tag{3.31}$$

czyli, że sumaryczny czas wykonania poszczególnych zadań algorytmu równoległego jest nie mniejszy niż czas wykonania algorytmu sekwencyjnego. W przeciwnym razie, wykonywanie na jednym procesorze kolejnych zadań równoległych prowadziłoby do krótszego czasu sekwencyjnego wykonywania algorytmu. Uwzględniając (3.30)–(3.31) oraz (3.5), efektywność algorytmu równoległego bez równoważenia obciążenia nie może być większa niż

$$E(p) = \frac{T_C(1)}{p\,T_C(p)} \leqslant \frac{\sum_{i=1}^{p} t_i}{p\,\max\{t_i : 1 \leqslant i \leqslant p\}} = \frac{\operatorname{avg}\{t_i : 1 \leqslant i \leqslant p\}}{\max\{t_i : 1 \leqslant i \leqslant p\}}. \tag{3.32}$$

Równanie (3.32) określa, że efektywność algorytmu równoległego z nierównomiernie rozłożonymi obliczeniami nie może być większa od współczynnika zrównoważenia obciążenia β:

$$E(p) \leqslant \beta = \frac{\operatorname{avg}\{t_i : 1 \leqslant i \leqslant p\}}{\max\{t_i : 1 \leqslant i \leqslant p\}}. \tag{3.33}$$

Zależność (3.32)–(3.33) określa górne ograniczenie na efektywność algorytmu równoległego i uzasadnia konieczność uwzględniania metod równoważenia obciążenia w projektowaniu algorytmów równoległych.

3.9 Zastosowane środowiska równoległe

Celem niniejszej pracy jest opracowanie algorytmów pozwalających na rozwiązywanie zagadnień związanych z optymalną obserwacją układów z czasoprzestrzenną dynamiką z zastosowaniem przetwarzania równoległego. Symulacje z wykorzystaniem opracowanych algorytmów równoległych przeprowadzono w środowiskach równoległych z pamięcią rozproszoną, jakimi są klastry obliczeniowe.

3.9.1 Homogeniczne środowiska obliczeniowe

Do celów symulacji z wykorzystaniem homogenicznych środowisk obliczeniowych wykorzystano dwa systemy równoległe, każdy składający się z jednakowych węzłów obliczeniowych:

- klaster zainstalowany w Centrum Komputerowym Uniwersytetu Zielonogórskiego,
- klaster zainstalowany w Poznańskim Centrum Superkomputerowo-Sieciowym.

Klaster zainstalowany w Centrum Komputerowym Uniwersytetu Zielonogórskiego zbudowano w ramach projektu Clusterix [35, 248]. Jego specyfikację sprzętową przedstawiono w tabeli 3.2. Klaster udostępnia narzędzia programistyczne firmy Intel: kompilatory języków C/C++ oraz Fortran 95 z biblioteką procedur numerycznych Intel MKL (*ang.* Math Kernel Library). Programowanie w środowisku równoległym oparte jest na interfejsie programowania z zastosowaniem przesyłania komunikatów (MPI) w postaci bibliotek mpich-1 oraz mpich-2.

Tabela 3.2: Specyfikacja klastra CLUSTERIX w CK UZ.

architektura procesora	IA-64
liczba procesorów w węźle	2
liczba rdzeni w procesorze	1
częstotliwość pracy procesora	1.4 GHz
liczba węzłów	4
interfejsy sieciowe	Gigabit Ethernet
sumaryczna wielkość pamięci systemowej	16 GB
sumaryczna wielkość pamięci dyskowej	300 GB
sumaryczna moc obliczeniowa	44.8 GFLOPs

Drugi z wykorzystanych systemów równoległych (klaster sherwood) zainstalowany jest w Poznańskim Centrum Superkomputerowo-Sieciowym. W klastrze zainstalowano procesory o architekturze Intel IA-64. Dokładną specyfikację sprzętową przedstawiono w tabeli 3.3. Klaster wykorzystuje system operacyjny Debian GNU/Linux. Do gospodarowania zadaniami wykorzystano system zarządzania w trybie wsadowym PBS (*ang.* Portable Batch System). W środowisku dostępne są biblioteki do programowania równoległego z wykorzystaniem przesyłania komunikatów mpich-1 oraz mpich-2. Dodatkowo istnieje możliwość wykorzystania biblioteki procedur numerycznych Intel MKL (*ang.* Math Kernel Library). Użytkownikom udostępniono kompilatory języków C oraz Fortran 95 firmy Intel w wersji 10.1.

Tabela 3.3: Specyfikacja klastra sherwood w PCSS.

	Typ 1	Typ 2	Typ 3
architektura procesora	IA-64	IA-64	Intel Xeon
liczba procesorów w węźle	2	2	1
liczba rdzeni w procesorze	1	1	1
częstotliwość pracy procesora	1.3 GHz	1.4 GHz	2.4 GHz
liczba węzłów	109	15	2
interfejsy sieciowe	Gigabit Ethernet		
sumaryczna wielkość pamięci systemowej	500 GB		
sumaryczna wielkość pamięci dyskowej	9.2 TB		
sumaryczna moc obliczeniowa	1.4 TFLOPs		

3.9.2 Heterogeniczne środowiska obliczeniowe

Do symulacji z zastosowaniem heterogenicznego środowiska obliczeniowego wykorzystano tymczasowy klaster zbudowany ze stacji roboczych klasy PC. Klaster składał się z 10 maszyn posiadających procesory Intel®Celeron oraz Intel®Pentium IV. Dokładną specyfikację przedstawiono w tabeli 3.4. Jako system operacyjny zastosowano dystrybucję ParalleKnoppix [167] opartą na dystrybucji Debian GNU/Linux. Pozwala ona na utworzenie klastra obliczeniowego uruchamianego z płyt CD, bez potrzeby instalacji na dysku twardym stacji roboczej. Dystrybucję dostosowano pod kątem wymaganych bibliotek oraz narzędzi wspomagających obliczenia. Zainstalowano biblioteki procedur numerycznych Intel MKL (*ang.* Math Kernel Library), kompilatory języka C oraz Fortran 95 firmy Intel. Środowisko przetwarzania równoległego opierało się o mechanizm przesyłania komunikatów (MPI) udostępniany przez biblioteki mpich-1 i mpich-2.

Tabela 3.4: Specyfikacja klastra heterogenicznego.

	Typ 1	Typ 2
architektura procesora	Celeron	Pentium IV
liczba procesorów w węźle	1	1
liczba rdzeni w procesorze	1	1
częstotliwość pracy procesora	1.7 GHz	2.4 GHz
liczba węzłów	5	5
interfejsy sieciowe	Fast Ethernet	
sumaryczna wielkość pamięci systemowej	20 GB	
sumaryczna wielkość pamięci dyskowej	600 GB	

Dodatkowo, środowisko heterogeniczne zostało zasymulowane na klastrze obliczeniowym CLUSTERIX zainstalowanym na Uniwersytecie Zielonogórskim. W celu wprowadzenia niejednorodności w mocach procesorów zainstalowanych na poszczególnych węzłach, na niektórych z nich uruchomiono dodatkowo procesy wprowadzające istotne dodatkowe obciążenie jednostek obliczeniowych. W rezultacie, część procesorów pracowała ze znacznie zmniejszoną mocą obliczeniową powodując, że klaster udostępniał niejednorodną moc obliczeniową.

4

Obliczenia równoległe w optymalnej aktywacji czujników stacjonarnych i skanujących

Niniejszy rozdział poświęcono problemom optymalnej aktywacji czujników stacjonarnych oraz skanujących. Zagadnienia zostaną przedstawione z uwzględnieniem praktyki inżynierskiej tak, aby rozważania teoretyczne mogły być zastosowane w rzeczywistych eksperymentach. Problem optymalnej obserwacji obiektów z czasoprzestrzenną dynamiką sformułowano jako problem optymalizacji dyskretnej. Wyniki badań zaproponowanych metod zostaną porównane zarówno w kontekście przyspieszania obliczeń przy zastosowaniu metod zrównoleglania oraz efektywnych metod wyznaczania macierzy informacyjnej. Przyspieszenie obliczeń uzyskane dzięki zrównolegleniu algorytmów zostanie rozważone zarówno dla wieloprocesorowych środowisk homogenicznych, jak i heterogenicznych, z zastosowaniem metod równoważenia obciążenia.

4.1 Wprowadzenie

W klasycznym podejściu do problemu optymalnej obserwacji rozważa się ogólne zagadnienie optymalnego rozmieszczania czujników stacjonarnych oraz skanujących (wraz z ich aktywacją). Jednak rozwiązywanie tak zdefiniowanych problemów okazuje się często bardzo skomplikowane. Rzadko istnieje możliwość rozmieszczenia czujników we wszystkich miejscach wyznaczonych przez algorytm planujący. Wprowadzenie ograniczeń na dopuszczalne konfiguracje czujników powoduje dodatkową komplikację rozważanych zagadnień.

W niniejszym rozdziale problem optymalnej obserwacji czujników stacjonarnych i skanujących definiuje się jako wybór aktywnej konfiguracji czujników spośród wielu

czujników już istniejących i rozmieszczonych w badanym obszarze. Taki postawienie problemu, oprócz zalet w postaci prostszego sformułowania problemu, prowadzi do prostszych implementacji w warunkach rzeczywistych, do czego przyczynił się gwałtowny rozwój technik związanych z sieciami sensorycznymi. Niska cena tego typu urządzeń pozwala na rozmieszczenie dużej liczby czujników na obszarze działania badanego zjawiska.

Optymalna aktywacja jedynie wybranej liczby urządzeń ma na celu m.in. optymalizację zużycia energii przez czujniki pomiarowe. Największe zużycie energii następuje w momencie wykonywania pomiaru oraz podczas przesyłania wyników drogą radiową do stacji bazowych. Zagadnienie optymalnej aktywacji czujników stacjonarnych i skanujących ma na celu wybranie spośród wszystkich rozmieszczonych czujników jedynie takiej konfiguracji wybranej liczby czujników, która pozwoli na uzyskanie jak najlepszej wiedzy o badanym zjawisku w celu późniejszej identyfikacji nieznanych parametrów opisujących rozważany model zjawiska.

4.1.1 Rozmieszczenie czujników

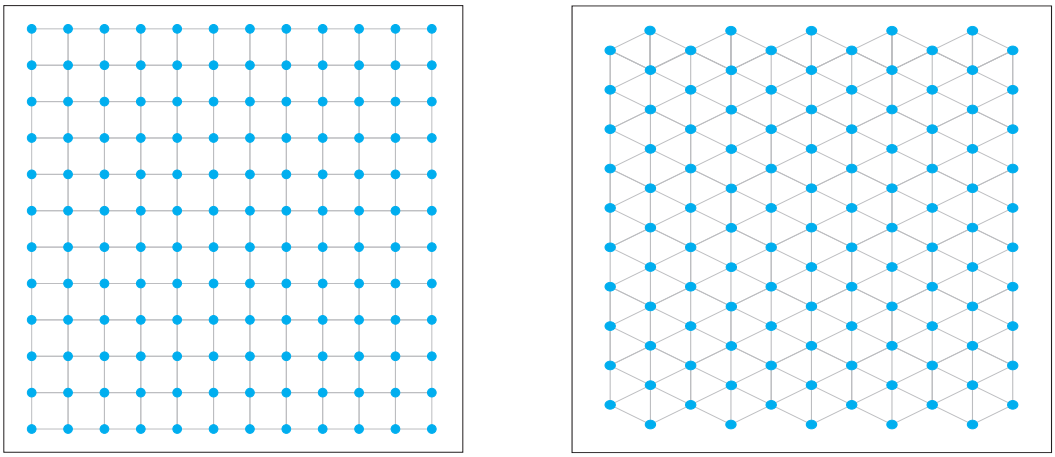
W celu skupienia uwagi, przedstawione rozważania dotyczą sytuacji, w których dziedziną określoności układu z czasoprzestrzenną dynamiką jest pewien obszar Ω w przestrzeni $\mathbb{R}^2$. Uogólnienie zaproponowanych metod i algorytmów na przypadek optymalnej aktywacji czujników stacjonarnych oraz skanujących w przestrzeni $\mathbb{R}^3$ nie stanowi jednak większego problemu.

Czujniki mogą być rozmieszczone w zadanym obszarze w węzłach nieregularnej bądź regularnej siatki przestrzennej. Rozmieszczenie nieregularne może być wynikiem np. rozrzucenia czujników przez samolot nad obszarem funkcjonowania badanego zjawiska. Rozmieszczenie czujników w postaci regularnej siatki pozwala na prostszą analizę rezultatów wykonywanych pomiarów. W przypadku siatek regularnych najczęściej spotyka się siatkę kwadratową oraz sześciokątną [33, 98] (zob. rys. 4.1).

4.1.2 Zastosowanie algorytmów optymalizacji dyskretnej

Wybór metod i algorytmów odpowiednich do rozwiązania problemu optymalnej aktywacji czujników stacjonarnych i skanujących poprzedzono analizą wymagań jakie powinny one spełniać. W pierwszym rzędzie powinny być to narzędzia, które

- wyznaczają bezreplikacyjne plany eksperymentu, czyli takie, w których możliwy jest tylko jeden eksperyment pozwalający na estymację nieznanych parametrów,
- uwzględniają występowanie skorelowania obserwacji, które występuje w praktyce,
- są możliwie proste, co ułatwiałoby ich zrównoleglenie.



(a) (b)

Rysunek 4.1: Przykładowe regularne rozmieszczenia czujników: (a) regularna siatka kwadratowa, (b) regularna siatka sześciokątna.

Analiza powyższych wymagań pozwoliła na zaproponowanie metod opartych na znanych schematach dyskretnych algorytmów metaheurystycznych: GRASP oraz Tabu-Search. Podejście z wykorzystaniem wybranych algorytmów dyskretnych pozwala na bezpośrednie opracowywanie bezreplikacyjnych planów eksperymentu. Ze względu na swoją specyfikę, związaną z eksploracją przestrzeni możliwych rozwiązań, schematy GRASP i Tabu-Search pozwalają również na wykorzystanie metody redukcji czasochłonnych operacji, co w przypadku rozważania skorelowanych obserwacji ma niebagatelny wpływ na wydajność.

4.1.3 Definicje sąsiedztwa

Rozważane badania dotyczą optymalnej aktywacji czujników stacjonarnych i skanujących znajdujących się w obszarze $\Omega \in \mathbb{R}^2$. Następstwem wyboru metod należących do metaheurystycznych algorytmów dyskretnych jest zdefiniowanie sąsiedztw czujników, które w przypadku regularnych siatek kwadratowych przedstawiają się następująco:

- sąsiedztwo von Neumanna o promieniu r (zob. rys. 4.2(a)),

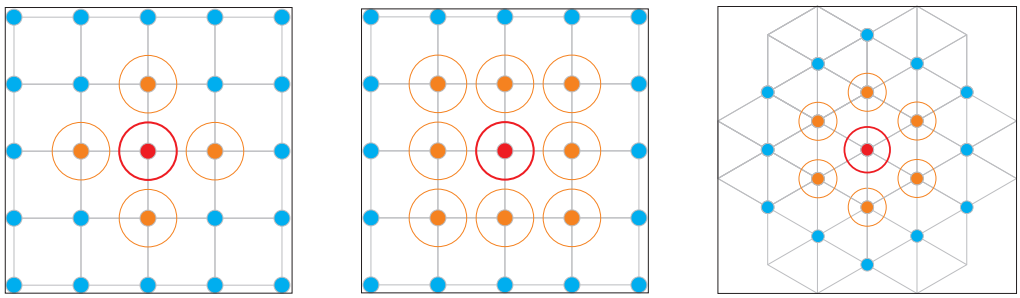
$$N_{VN}(x^i) = \left\{x^j \in \mathrm{X} : \rho_1(x^i, x^j) \leqslant r\right\} \tag{4.1}$$

- sąsiedztwo Moore'a o promieniu r (zob. rys. 4.2(b)),

$$N_M(x^i) = \left\{x^j \in \mathrm{X} : \rho_\infty(x^i, x^j) \leqslant r\right\} \tag{4.2}$$

gdzie:

$X \subset \Omega$ — zbiór położeń wszystkich czujników rozmieszczonych w badanych obszarze,
$\rho_1(x^i, x^j) = \sum_{\alpha=1}^{2} |x^i_\alpha - x^j_\alpha|$,
$\rho_\infty(x^i, x^j) = \max\{|x^i_\alpha - x^j_\alpha| : \alpha = 1, 2\}$,
x^i_α — oznacza wartość współrzędnej α położenia i-tego czujnika.



(a) (b) (c)

Rysunek 4.2: Sąsiedztwo dla regularnej siatki kwadratowej: (a) von Neumanna $r = 1$, (b) Moore'a $r = 1$ oraz (c) siatki sześciokątnej $r = 1$.

W przypadku regularnej siatki sześciokątnej powyższe sąsiedztwa przedstawiono na rys. 4.2(c).

4.2 Optymalna aktywacja czujników stacjonarnych

Problem optymalnej aktywacji czujników stacjonarnych polega na znalezieniu optymalnej konfiguracji ustalonej liczby aktywnych czujników, która pozostaje niezmienna podczas trwania procesu obserwacji.

Problem optymalnej aktywacji czujników stacjonarnych jest najprostszym i najczęściej rozważanym wariantem zagadnienia optymalnej obserwacji procesów z czasoprzestrzenną dynamiką spośród zadań przedstawionych w rozdz. 2. Łatwość zastosowania wyników obliczeń w aplikacjach oraz brak skomplikowanych założeń formalnych ułatwiają to zadanie. Bardzo często czujniki są urządzeniami stacjonarnymi znajdującymi się stale w jednym niezmiennym położeniu w przestrzeni, np. stacje pomiarowe służb meteorologicznych lub czujniki stężenia zanieczyszczeń w aglomeracjach miejskich. Optymalna aktywacja czujników tego rodzaju w miejscach określonych w procesie optymalizacji pozwala na dokładniejszą obserwację badanych zjawisk.

4.2.1 Sformułowanie problemu

Zagadnienie optymalnej aktywacji czujników stacjonarnych definiuje się jako szukanie najlepszej konfiguracji n aktywnych czujników spośród N dostępnych czujników rozmieszczonych w obszarze funkcjonowania badanego zjawiska $\Omega \in \mathbb{R}^2$. W takim przypadku macierz informacyjna związana z aktualnym planem eksperymentu

$$\xi = \left\{x^1, x^2, \ldots, x^n\right\}, \tag{4.3}$$

przy czym $x^i \neq x^j$ o ile $i \neq j$, opisującym aktywną konfigurację czujników, przedstawia się następująco:

$$\mathcal{M}(\xi) = \int_T G(\xi, t) C^{-1}(\xi) G^{\mathsf{T}}(\xi, t)\, \mathrm{d}t, \tag{4.4}$$

gdzie:

$$G(\xi, t) = \left[g(x^1, t) \mid \ \ldots \ \mid g(x^n, t)\right], \tag{4.5}$$

$$g(x, t) = \left[\frac{\partial y(x, t; \theta)}{\partial \theta_1}, \ \ldots \ , \frac{\partial y(x, t; \theta)}{\partial \theta_m}\right]^{\mathsf{T}}_{\theta=\theta^0}, \tag{4.6}$$

opisują wrażliwość stanu w punktach odpowiadających położeniom czujników na zmiany parametrów (θ^0 — wstępna ocena wektora estymowanych parametrów), a

$$C(\xi) = [c_{ij}(\xi)], \quad c_{ij}(\xi) = q(x^i, x^j), \tag{4.7}$$

opisuje kowariancję obserwacji czujników (q — zadana funkcja nieujemna, zob. równ. (4.39)).

Otrzymuje się w ten sposób nieliniowe zagadnienie aktywowania n różnych czujników spośród N dostępnych czujników rozmieszczonych w rozważanym obszarze przestrzennym.

4.2.2 Algorytm GRASP

Problem aktywowania n-elementowego podzbioru wybranych czujników spośród zbioru wszystkich N czujników dostępnych w obszarze funkcjonowania badanego zjawiska jest zadaniem kombinatorycznym. Pełne przeszukanie zbioru rozwiązań wymaga sprawdzenia

$$\binom{N}{n} = \frac{N!}{n!\,(N-n)!}, \tag{4.8}$$

różnych kombinacji położeń czujników. Już dla prostych problemów optymalnej aktywacji czujników stacjonarnych rozważanie pełnego przeszukania wszystkich możliwych konfiguracji aktywnych czujników jest nierealizowalne. Przykładowo, optymalna aktywacja 21 czujników spośród 441 czujników rozmieszczonych na regularnej siatce kwadratowej o rozmiarze 21×21 prowadzi do konieczności sprawdzenia około

$$\mathrm{C}^{21}_{441} = \frac{441!}{21!\,420!} \approx 4.12 \cdot 10^{35} \tag{4.9}$$

różnych konfiguracji aktywnych czujników. Przy założeniu, że w jednej sekundzie można wyznaczyć 10^6 wartości kryterium optymalności dla różnych położeń czujników, czas trwania pełnego przeszukiwania wynosiłby około $4.12 \cdot 10^{29}$ sekund, co przekłada się na konieczność wykonywania obliczeń przez około $1.3 \cdot 10^{22}$ lat. Rozwiązaniem jest zastosowanie metody opartej na metaheurystyce powodującej znaczne zmniejszenie przestrzeni poszukiwań optymalnych konfiguracji aktywnych czujników, rzecz jasna — kosztem pewnego błędu o wielkości wynikającej z przyjętych uproszczeń.

Obliczanie macierzy informacyjnej dla konfiguracji wielu aktywnych czujników wymaga przeprowadzenia obliczeń (4.4)–(4.7), a następnie określenia wartości wybranego kryterium optymalności na podstawie wyznaczonej macierzy informacyjnej. Oprócz przeliczenia macierzy wrażliwości związanej z aktualną konfiguracją czujników (4.5), operacje te wymagają przede wszystkim modyfikacji macierzy kowariancji (4.7) oraz, co za tym idzie, macierzy informacyjnej (4.4). Jednak wyznaczanie macierzy $\mathcal{M}$ oaz G, ale przede wszystkim odwrotności macierzy kowariancji C, są operacjami bardzo czasochłonnymi. Już proste zagadnienia wyznaczenia np. optymalnej konfiguracji 21 aktywnych czujników stacjonarnych wymaga wyznaczania odwrotności macierzy o wymiarach 21×21. W miarę możliwości, opracowana metoda powinna dokonać redukcji czasochłonnych operacji, która pozwoli na znaczne zmniejszenie liczby operacji koniecznych do wyznaczenia odwrotności macierzy kowariancji, a co za tym idzie — również macierzy informacyjnej.

Analiza przedstawionych wymagań pozwoliła na opracowanie metody rozwiązania problemu optymalnej aktywacji czujników stacjonarnych w celu obserwacji układów z czasoprzestrzenną dynamiką bazującej na zachłannej randomizowanej procedurze adaptacyjnych poszukiwań losowych (*ang.* Greedy Randomized Adaptive Search Procedure — GRASP). GRASP należy do klasy algorytmów heurystycznej optymalizacji dyskretnej [60, 61, 190]. Ogólny schemat działania proponowanej metody przedstawiono w postaci Algorytmu 4.1. Metoda składa się z dwóch faz: fazy zachłannej konstrukcji przybliżenia rozwiązania oraz fazy przeszukiwania lokalnego. Faza konstrukcji rozwiązania określa zbiór możliwych rozwiązań, z których następnie wyszukiwane jest optimum podczas fazy lokalnego przeszukiwania sąsiedztw możliwych rozwiązań. Obydwie fazy wykorzystują operacje aktywacji i deaktywacji czujników, co pozwala na zastosowanie metody redukcji czasochłonnych operacji.

Algorytm rozpoczyna pracę z pewną liczbą losowo aktywowanych czujników. Liczba aktywnych czujników powinna być przynajmniej równa liczbie parametrów układu jakie chcemy zidentyfikować ($\ell \geqslant m$). Takie założenie ma na celu wyeliminowanie sytuacji, w której macierz informacyjna, w pewnych warunkach, może się stać osobliwa. Liczba aktywnych czujników większa od liczby identyfikowanych parametrów pozwala uniknąć takiej sytuacji [228].

Wykorzystanie ilościowego warunku stopu algorytmu wymusza konieczność zdefiniowania maksymalnej liczby iteracji $\tau_{\max}$. Możliwe jest również zastosowanie jakościowego warunku stopu, który powoduje zakończenie pracy algorytmu w momencie, gdy wykonanie kolejnej iteracji algorytmu nie powoduje polepszenia wartości kryte-

rium optymalności.

Algorytm 4.1 Optymalna aktywacja czujników stacjonarnych

Dane wejściowe:

X — zbiór położeń wszystkich dostępnych czujników,
$\mathcal{N}$ — definicja otoczenia punktu X,
α — parametr sterujący tworzeniem listy czujników kandydujących (RCL) ,
$\xi^{(0)}$ — plan eksperymentu opisujący wstępną konfigurację aktywnych czujników,
ℓ — parametr określający liczbę czujników bieżącego przybliżenia planu optymalnego, uwzględnianych w fazie konstrukcji następnego rozwiązania przybliżonego ($\ell < n$),
$\tau_{\max}$ — maksymalna liczba iteracji algorytmu,

Dane wyjściowe:

Ψ_{opt} — obliczone przybliżenie optymalnej wartości kryterium,
ξ_{opt} — znalezione przybliżenie optymalnego planu eksperymentu.

Krok 0: (Inicjalizacja)
Określenie początkowego przybliżenia $\xi_{\text{opt}} \leftarrow \xi^{(0)}$ planu optymalnego oraz ustawienie wartości początkowej $\Psi_{\text{opt}} \leftarrow \Psi(\xi^{(0)})$. Inicjalizacja licznika iteracji $\tau \leftarrow 1$.

Krok 1: (Faza konstrukcji rozwiązania wstępnego)
Konstrukcja wstępnego planu eksperymentu $\hat{\xi}^{(\tau)}$ na podstawie planu aktualnego $\xi^{(\tau-1)}$ oraz tworzonej dynamicznie listy czujników kandydujących (RCL) definiowanej przez parametr α (zob. Alg. 4.2).

Krok 2: (Faza przeszukiwania lokalnego)
Określenie nowego planu $\xi^{(\tau)}$ na bazie przeszukiwania sąsiedztwa planu $\hat{\xi}^{(\tau)}$ (zob. Alg. 4.3).

Krok 3: (Sprawdzenie optimum oraz warunek stopu)
Jeżeli wartość kryterium $\Psi(\xi^{(\tau)})$ jest lepsza od Ψ_{opt} to następuje aktualizacja $\Psi_{\text{opt}} \leftarrow \Psi(\xi^{(\tau)})$ oraz $\xi_{\text{opt}} \leftarrow \xi^{(\tau)}$. Zwiększenie wartości licznika τ. Jeżeli $\tau < \tau_{\max}$ to następuje powrót do Kroku 1.

■

Schemat działania fazy konstrukcji wstępnego rozwiązania przybliżonego przedstawiono w postaci Algorytmu 4.2. W tej fazie, spośród czujników aktualnie uśpionych (nieaktywnych) wybieramy te, których aktywacja pozwala na dokładniejszą obserwację badanego zjawiska ze względu na założone kryterium optymalności.

Faza rozpoczyna się od inicjalizacji zmiennych. Następnie spośród wszystkich położeń czujników w planie wejściowym $\xi^{(\tau-1)}$ wybiera się losowo $\ell < n$ położeń (powinno również zachodzić $\ell \geqslant m$) stanowiących bazę ζ_ℓ do tworzenia kolejnego planu eksperymentu poprzez kolejne dołączanie pojedynczych punktów w sposób zbliżony do działania algorytmów zachłannych.

W kolejnym kroku bada się wszystkie czujniki nieaktywne pod kątem zmiany wartości kryterium optymalności po ich aktywacji. Krok ten wymaga wielokrotnej aktywacji nowych czujników w aktualnym planie eksperymentu w celu obliczenia wartości kryterium optymalności. Należy zauważyć, że na tym etapie można zastosować metodę redukcji czasochłonnych operacji opartą na metodzie wymiany czujników przedstawionej dokładniej w rozdz. 4.4. Wartości $\Psi_{\min}$ i $\Psi_{\max}$ definiują zakres, w którym mieszczą się wartości kryterium optymalności dla planów będących jednopunktowymi rozszerzeniami planu bazowego ζ_ℓ. Gdy do planu eksperymentu dołączony zostanie czujnik implikujący wystarczająco małą wartość kryterium optymalności, zostaje on dołączony do listy RCL (*ang.* Restricted Candidate List). Lista ta zawiera położenia czujników, których aktywacja powoduje „znaczące" polepszenie wartości kryterium optymalności w zadanych granicach. Następnie z listy RCL losowo wybiera się czujnik, którego położenie zostanie dołączone do konstruowanego planu wstępnego. Jeżeli plan wstępny nie posiada zadanej liczby aktywnych czujników, to następuje powrót do krok pierwszego. W momencie, gdy plan wstępny posiada wymaganą liczbę aktywnych czujników, następuje koniec fazy konstrukcji rozwiązania wstępnego, a ostatnio utworzony plan ζ_κ staje się planem wejściowym dla fazy przeszukiwania lokalnego.

Zasadnicze znaczenie dla działania fazy konstrukcji rozwiązania wstępnego ma wartość parametru α sterującego wyborem położeń czujników do ograniczonej listy kandydatów oraz sama metoda wyboru czujnika z listy RCL. Wybranie wartości granicznej, czyli $\alpha = 0$ oznacza, że akceptowane są tylko te czujniki, których aktywacja pozwala na wyznaczenie kryterium optymalności o wartości minimalnej $\Psi_{\min}$. Zwiększanie wartości α, z kolei oznacza, że granice akceptowalności czujników przy dołączaniu do ograniczonej listy kandydatów zostają rozszerzone. Przyjęcie α równego jedności oznacza, że lista RCL obejmuje wszystkie nieaktywne czujniki i jedynym parametrem sterującym aktywacją nowego czujnika będzie procedura wyboru czujnika z listy RCL.

Wybór czujnika z ograniczonej listy kandydatów można przeprowadzić na wiele sposobów. Często spotykaną metodą jest potraktowanie listy RCL jak kolejki FIFO i wybór pierwszego elementu wprowadzonego do listy. W niniejszej pracy zastosowano losowy wybór czujnika. Wprowadzenie elementu stochastycznego pozwala na lepsze zachowanie się algorytmu w przypadku „utknięcia" w minimum lokalnym.

Algorytm 4.2 Faza konstrukcji rozwiązania wstępnego

Dane wejściowe:

α — parametr sterujący tworzeniem listy RCL czujników kandydujących do

aktywacji,
$\xi^{(\tau-1)}$ — plan eksperymentu opisujący bieżącą konfigurację aktywnych czujników,
ℓ — parametr określający liczbę elementów planu $\xi^{(\tau-1)}$ uwzględnianych w fazie konstrukcji rozwiązania wstępnego $\hat{\xi}^{(\tau)}$.

Dane wyjściowe:
$\hat{\xi}^{(\tau)}$ — wstępne przybliżenie planu kandydującego na kolejne przybliżenie planu optymalnego.

Krok 0: (Inicjalizacja)
Losowy wybór ℓ elementów planu $\xi^{(\tau-1)}$, które stanowią plan ζ_ℓ. Inicjalizacja licznika iteracji $\kappa \leftarrow \ell$.

Krok 1: (Wyznaczenie zakresu zmienności kryterium podczas aktywacji dodatkowego czujnika)
Wyznaczenie najmniejszej i największej wartości kryterium optymalności jakie można uzyskać dołączając do planu ζ_κ pojedynczy punkt spośród położeń czujników uśpionych.

$$\begin{aligned} \Psi_{\min} &= \min_{\mathrm{s}\in\mathrm{S}} \Psi\Big(\zeta_\kappa \cup \{\mathrm{s}\}\Big), \\ \Psi_{\max} &= \max_{\mathrm{s}\in\mathrm{S}} \Psi\Big(\zeta_\kappa \cup \{\mathrm{s}\}\Big), \end{aligned} \tag{4.10}$$

gdzie: $\mathrm{S} = \mathrm{X} \setminus \{\zeta_\kappa\}$.

Krok 2: (Utworzenie listy RCL kandydatów do jednopunktowego powiększenia dotychczasowego planu)
Określenie listy RCL położeń czujników, których włączenie do planu eksperymentu zagwarantuje polepszenie wartości kryterium w granicach określonych przez $\Psi_{\min}$, i $\Psi_{\max}$ oraz α, zgodnie ze wzorem

$$\mathrm{RCL} = \Big\{ s \in \mathrm{S} \;\Big|\; \Psi\Big(\zeta_\kappa \cup \{\mathrm{s}\}\Big) \leqslant \Psi_{\min} + \alpha(\Psi_{\max} - \Psi_{\min}) \Big\}. \tag{4.11}$$

Krok 3: (Ostateczny wybór kolejnego czujnika do konstruowanego planu)
Z utworzonej listy RCL wybiera się losowo element $\{s_0\}$, który zostaje dołączony do dotychczasowego planu:

$$\zeta_{\kappa+1} = \zeta_\kappa \cup \{s_0\}. \tag{4.12}$$

Jeżeli $\kappa = n-1$, przyjmuje się $\hat{\xi}^{(\tau)} = \zeta_{\kappa+1}$ i kończy pracę. W przeciwnym razie zwiększa się licznik κ i następuje powrót do Kroku 1.

■

Schemat działania fazy przeszukiwania lokalnego przedstawiono jako Algorytm 4.3. W tej fazie następuje ulepszenie rozwiązania skonstruowanego przez fazę tworzenia rozwiązania przybliżonego (pseudozachłannego). Faza ta polega na powtarzaniu dwóch głównych etapów:

- etapu poszukiwania czujnika, którego deaktywacja powoduje najmniejszą utratę wartości wybranego kryterium optymalności,
- etapu poszukiwania czujnika, którego aktywacja powoduje największe zmniejszenie wartości wybranego kryterium optymalności.

Zasadniczy wpływ na sposób działania tej fazy ma wybór sąsiedztwa aktualnego planu. Spośród wyznaczonego sąsiedztwa algorytm wybiera takie czujniki, których aktywacja powoduje polepszenie właściwości opracowywanego planu eksperymentu. Sposób tworzenia sąsiedztwa pojedynczego czujnika przedstawiono w rozdz. 4.1.3.

Na początku, na podstawie wstępnego planu eksperymentu $\hat{\xi}^{(\tau)}$ tworzony jest plan bazowy $\zeta^{(0)}$, który będzie ulepszany w celu uzyskania możliwie najmniejszej wartości zadanego kryterium optymalności. Kolejnym krokiem algorytmu jest analiza aktualnej konfiguracji czujników planu bazowego pod względem wyboru czujnika, którego deaktywacja powoduje najmniejszą utratę wartości założonego kryterium optymalności eksperymentu. Po jego usunięciu z aktualnego planu bazowego, następuje etap poszukiwania czujnika uśpionego, którego włączenie spowoduje polepszenie właściwości zmodyfikowanego planu bazowego $\hat{\zeta}^{(\kappa)}$ pod względem wartości kryterium optymalności. Przeszukiwaniu podlega sąsiedztwo oryginalnego planu bazowego $\zeta^{(\kappa)}$ z wyłączeniem czujników już należących do omawianego planu.

Procedurę wymiany punktów planu powtarza się dopóki możliwa jest poprawa wartości kryterium planowania, a końcowa konfiguracja czujników stanowi bazę dla kolejnych iteracji całego algorytmu.

Algorytm 4.3 Faza przeszukiwania lokalnego

Dane wejściowe:
$\hat{\xi}^{(\tau)}$ — wstępny plan eksperymentu opisujący aktualną konfigurację aktywnych czujników.

Dane wyjściowe:
$\xi^{(\tau)}$ — skorygowany plan $\hat{\xi}^{(\tau)}$, zoptymalizowany poprzez wymiany jednopunktowe w $\hat{\xi}^{(\tau)}$ i jego sąsiedztwie.

Krok 0: (Inicjalizacja)
Inicjalizacja licznika iteracji $\kappa \leftarrow 0$ oraz utworzenie planu $\zeta^{(0)} \leftarrow \hat{\xi}^{(\tau)}$.
Ustawienie $\Psi_{\min} \leftarrow \Psi(\zeta^{(0)})$.

Krok 1: (Usunięcie najmniej wartościowego czujnika aktywnego)
W planie $\zeta^{(\kappa)}$ określa się położenie czujnika, którego uśpienie powoduje najmniejszą utratę wartości kryterium:

$$\hat{\mathrm{s}} = \arg\min_{\mathrm{s}\in\zeta^{(\kappa)}} \Psi\Big(\zeta^{(\kappa)} \setminus \{\mathrm{s}\}\Big), \tag{4.13}$$

a następnie usuwa się go z $\zeta^{(\kappa)}$:

$$\bar{\zeta}^{(\kappa)} = \zeta^{(\kappa)} \setminus \{\hat{\mathrm{s}}\} \tag{4.14}$$

Krok 2: (Dołączenie najbardziej wartościowego czujnika uśpionego)
W sąsiedztwie planu $\zeta^{(\kappa)}$ zdefiniowanym jako

$$\mathcal{N}\Big(\zeta^{(\kappa)}\Big) = \bigcup_{\mathrm{s}\in\zeta^{(\kappa)}} \mathcal{N}(s), \tag{4.15}$$

poszukuje się położenia czujnika uśpionego, którego aktywacja powoduje największy ubytek wartości kryterium:

$$\bar{\mathrm{s}} = \arg\min_{\mathrm{s}\in\mathcal{N}(\zeta^{(\kappa)})\setminus\zeta^{(\kappa)}} \Psi\Big(\bar{\zeta}^{(\kappa)} \cup \{\mathrm{s}\}\Big). \tag{4.16}$$

Jeżeli $\Psi\Big(\bar{\zeta}^{(\kappa)} \cup \{\bar{s}\}\Big) \geqslant \Psi_{\min}$, algorytm kończy działanie po przyjęciu

$$\xi^{(\tau)} \leftarrow \zeta^{(\kappa)}. \tag{4.17}$$

W przeciwnym razie następują kolejno podstawienia

$$\begin{aligned} \zeta^{(\kappa+1)} &\leftarrow \bar{\zeta}^{(\kappa)} \cup \{\bar{\mathrm{s}}\}, \\ \Psi_{\min} &\leftarrow \Psi\Big(\bar{\zeta}^{(\kappa)} \cup \{\bar{s}\}\Big), \\ \kappa &\leftarrow \kappa + 1. \end{aligned} \tag{4.18}$$

oraz powrót do Kroku 1.

■

Doświadczenia zebrane podczas planowania strategii aktywacji czujników stacjonarnych w zagadnieniach związanych np. z rozprzestrzenianiem się zanieczyszczeń w atmosferze, sugerują pewne modyfikacje opracowanego algorytmu. W przypadku analizy modeli dyfuzji można zaobserwować skupienia się czujników. Tworzenie się takich grup czujników jest dość charakterystyczne i pozwala na wykorzystanie tej właściwości w celu szybszego wyznaczania optymalnych konfiguracji aktywnych czujników.

Modyfikacji może ulec liczba czujników wybieranych z ograniczonej listy kandydatów: wyborowi może podlegać więcej niż jeden czujnik. W przypadku wybrania wartości parametru α bliskiej zeru, do listy RCL trafiają czujniki, których aktywacja powoduje największą utratę wartości kryterium optymalności. W związku z własnością grupowania się czujników, istnieje duże prawdopodobieństwo, że czujniki znajdujące się na liście RCL znajdują się obok siebie i pomimo wybrania (w standardowym podejściu) tylko jednego czujnika, to kolejne również znajdą się na tej liście w kolejnych iteracjach algorytmu. Dlatego warto liczbę czujników wybieranych z listy RCL potraktować jako parametr tak, aby w poszczególnych iteracjach fazy konstrukcji rozwiązania pseudozachłannego aktywowanych było więcej niż jeden czujnik.

Wybieranie większej liczby czujników możliwe jest również podczas fazy przeszukiwania lokalnego (zob. Algorytm 4.3). Modyfikacji polega Krok 1, w którym poszukujemy już nie jednego, a kilku czujników, których deaktywacja powoduje najmniejszą utratę wartości wybranego kryterium optymalności. Zmiany wystąpią również w Kroku 2, gdzie poszukiwać się będzie wybranej liczby czujników, których aktywacja pozwoli na polepszenie własności opracowywanego planu eksperymentu pod kątem minimalizacji wartości zadanego kryterium optymalności.

Przykład 4.1

Jako przykład rozważmy następujący model adwekcji-dyfuzji opisujący rozprzestrzenianie się skażenia na danym obszarem. W obszarze Ω występuje źródło skażenia, które przyczynia się do zmian w koncentracji skażenia $y = y(x,t)$. Zmiany zanieczyszczenia y, obserwowane w horyzoncie czasowym $T = (0,1]$, opisuje następujące równanie adwekcji-dyfuzji:

$$\frac{\partial y(x,t)}{\partial t} + \nabla \cdot \Big(\upsilon(x) y(x,t)\Big) = \nabla \cdot \Big(a(x) \nabla y(x,t)\Big) + u(x) \quad \text{w } \Omega \times T \tag{4.19}$$

określone przez następujące warunki początkowe oraz brzegowe:

$$\frac{\partial y(x,t)}{\partial n} = 0 \qquad \text{na } \partial\Omega \times T, \tag{4.20}$$

$$y(x,0) = 0 \qquad \text{w } \Omega, \tag{4.21}$$

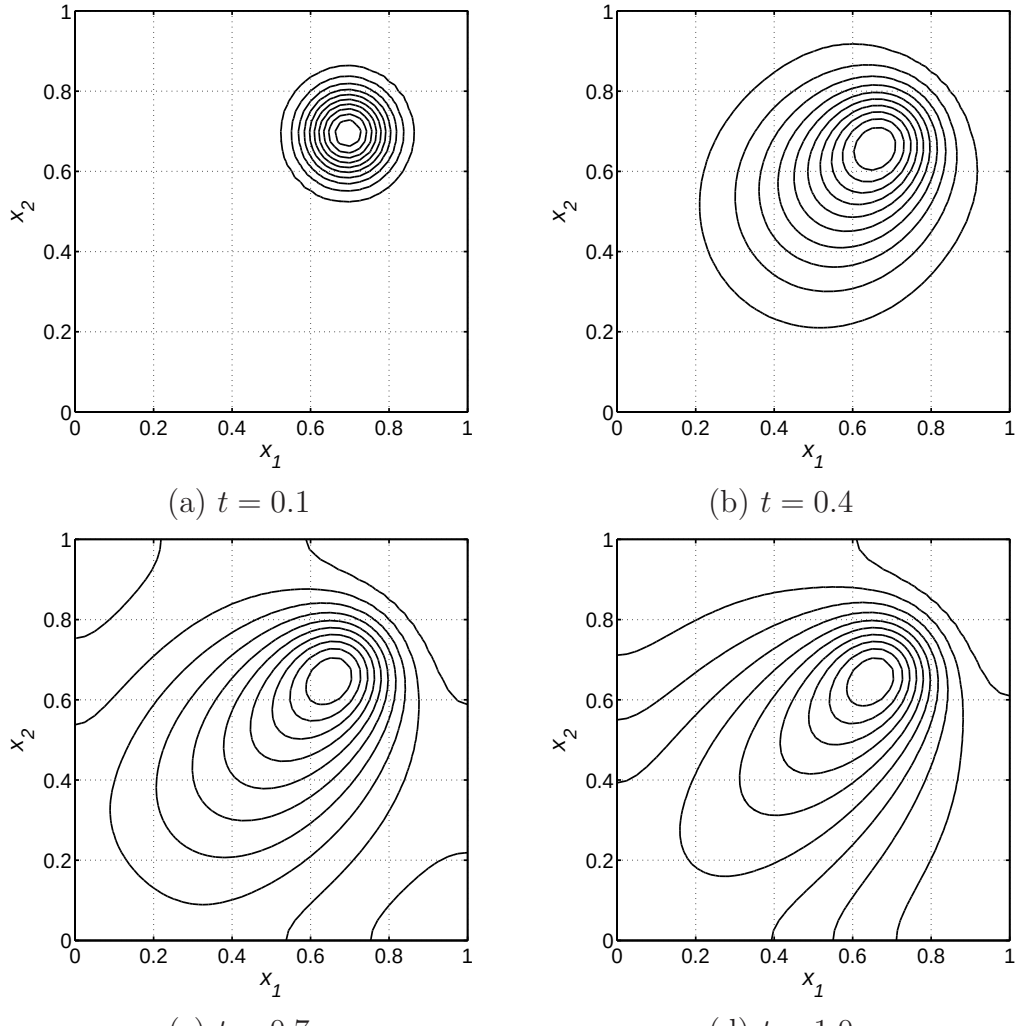
gdzie wyrażenie $u(x) = \exp\Big(-50\|x-c\|^2\Big)$ reprezentuje aktywne źródło zanieczyszczenia zlokalizowane w punkcie $c = (0.7, 0.7)$. Współczynnik adwekcji opisujący wiatr rozprzestrzeniający skażenie opisany jest jako $v = (v_1, v_2)$, gdzie:

$$v_1 = v_2 = \frac{1}{2}(x_1 - x_2)^2 - 1, \tag{4.22}$$

oraz $x_2 = x_1 + b, b \in [-1,1]$. Tak przedstawiony współczynnik adwekcji określa północno-wschodni wiatr wiejący ze zmienną prędkością. Funkcja opisująca współczynnik dyfuzji określona jest następująco:

$$a(x) = \theta_1 + \theta_2 x_1^2 + \theta_3 x_2^2, \tag{4.23}$$

przy czym parametry θ_1, θ_2 and θ_3 estymuje się bazując na pomiarach wykonywanych przez czujniki stacjonarne. Rozwiązanie zagadnienia (4.19)–(4.21) przedstawiono na rys. 4.3, gdzie można obserwować proces rozprzestrzeniania się skażenia. Chmura skażenia rozprzestrzenia się po całym obszarze i w ten sposób odzwierciedla skomplikowany proces dyfuzji oraz adwekcji. Jako wartości nominalne parametrów funkcji dyfuzji przyjęto $\theta_1^0 = 0.1$, $\theta_2^0 = \theta_3^0 = -0.02$. Rozważane zjawisko zostało zamodelowane w środowisku Matlab z wykorzystaniem przybornika Partial Differential Equation. Obszar Ω zdyskretyzowano równomierną siatką o rozmiarze 41×41, a horyzont czasowy podzielono na 80 równych podprzedziałów.



(a) $t = 0.1$ (b) $t = 0.4$

(c) $t = 0.7$ (d) $t = 1.0$

Rysunek 4.3: Koncentracja zanieczyszczeń w podanych chwilach czasowych dla Przykładu 4.1.

W celu estymacji wartości parametrów θ rozmieszczono 1681 czujników stacjonarnych na regularnej siatce kwadratowej 41×41. Założono, że jednoczesnej aktywa-

cji podlega 81 czujników spośród wszystkich rozmieszczonych na badanym obszarze. W badaniach uwzględniono występowanie korelacji pomiędzy pomiarami wykonywanymi przez sąsiednie czujniki. Korelacje zostały zdeterminowane przez odległość pomiędzy czujnikami wg wzoru

$$q(x^i, x^j) = \sigma^2 \exp\left(\frac{-\|x^i - x^j\|}{\beta}\right), \tag{4.24}$$

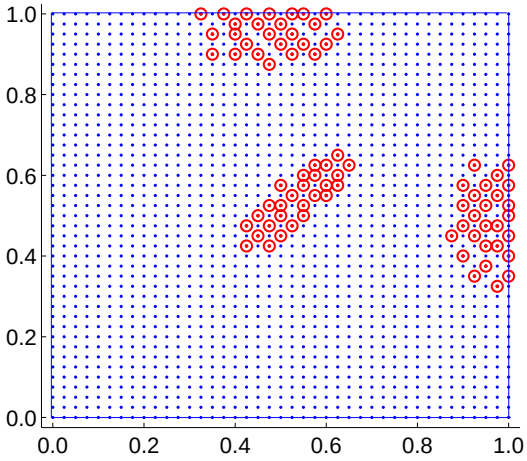
gdzie: β — współczynnik determinujący stopień skorelowania obserwacji. W przykładzie przyjęto następujące parametry opisujące korelacje: $\sigma = 1.0, \beta = 0.3$.

Do celów rozwiązania rozważanego problemu, Algorytm 4.1 zaimplementowano jako sekwencyjny. Przyjęto wykorzystanie sąsiedztwa Moore'a o promieniu $r = 1$. Parametr sterujący α wynosił 0.2, a maksymalną liczbę iteracji $\tau_{\max}$ ustawiono na 60. Założono wybieranie pojedynczego czujnika z listy RCL, a z bieżącego planu eksperymentu $\xi^{(\tau-1)}$ wybierano $\ell = 10$ czujników.

Obliczenia przeprowadzono w oparciu o uniwersytecki klaster zbudowany w ramach projektu CLUSTERIX [248] (zob. rozdz. 3.9). Klaster składa się z węzłów zawierających 64-bitowe procesory Intel Itanium^2 1.4GHz i działa pod kontrolą środowiska GNU/Linux Debian for ia64. Program napisano w języku Fortran 95 i skompilowano z zastosowaniem kompilatora *ifort* (Intel®Fortran Compiler v.8.1 for Linux 64-bit platforms). Aplikacja używa biblioteki procedur numerycznych Intel Math Kernel Library do wykonywania operacji mnożenia macierzy oraz wektorów.

Wyznaczoną aktywną konfigurację 81 czujników stacjonarnych przedstawiono na rys. 4.4. Jak można zauważyć, algorytm aktywował czujniki symetrycznie względem osi symetrii $x_1 = x_2$, co jest zbieżne z symetrią dynamiki rozważanego zjawiska. Na takie rozprzestrzenianie się zanieczyszczeń, a przez to — na takie rozmieszczenie czujników, ma wpływ przede wszystkim, wiatr północno-wschodni dominujący nad zjawiskiem dyfuzji. W rozmieszczeniu czujników widać pewne nieregularności zaburzające idealną symetrię ich rozmieszczenia. Mogą one być spowodowane, oprócz korelacji pomiędzy pomiarami wykonywanymi przez bliskie sobie czujniki, również brakiem wyznaczenia przez algorytm rozwiązania globalnie optymalnego. Co w pewien sposób zaskakujące, z punktu widzenia przeprowadzania eksperymentu, pomiary należy wykonywać nie tylko blisko źródła skażenia, ale również w miejscach wyznaczonych przez algorytm, które nie są oczywiste w przypadku analizy dynamiki procesu tylko i wyłącznie na podstawie koncentracji zanieczyszczenia w kolejnych chwilach czasowych.

Obliczenia sekwencyjne w przedstawionym środowisku trwały około 840 sekund, co stanowiło motywację do opracowania równoległej metody pozwalającej na szybsze uzyskiwanie rezultatów.



Rysunek 4.4: Konfiguracja aktywnych czujników dla Przykładu 4.1.

4.3 Optymalna aktywacja czujników skanujących

Gwałtowny rozwój technologii związanych z projektowaniem i wytwarzaniem urządzeń pomiarowych spowodował znaczne zmniejszenie cen tych urządzeń oraz ich popularyzację. Łatwa dostępność oraz prostota użytkowania pozwoliły na wykorzystywanie większych liczb czujników w procesie obserwacji układów z czasoprzestrzenną dynamiką.

Z drugiej strony, bardzo często badane zjawiska mają charakter bardzo dynamiczny i wymagają obserwacji na rozległym obszarze. Idea monitorowania zjawisk z zastosowaniem sieci czujników skanujących polega na rozstawieniu dużej liczby czujników w obszarze, w którym badane jest interesujące zjawisko. Jednak ciągłe wykonywanie pomiarów przez dużą sieć czujników nie zawsze jest efektywne. Wykonanie pomiaru przez czujnik i przesyłanie wyników drogą radiową wiąże się bowiem ze znacznym zwiększeniem zapotrzebowania na energię w stosunku do jego zapotrzebowania w stanie spoczynku/czuwania. Dodatkowo, częste wykonywanie pomiarów przez wszystkie czujniki powoduje, że kanały transmisyjne mogą zostać przeciążone powodując opóźnienia w dostarczaniu danych pomiarowych. Dlatego często stosuje się strategię aktywowania określonej liczby czujników spośród wszystkich rozmieszczonych na danym terenie. Dodatkowo określa się jak często następować będą zmiany w konfiguracji aktywnych czujników pomiarowych. Takie podejście pozwala na znaczne wydłużenie czasu pracy sieci czujników, które najczęściej zasilane są przy pomocy własnych źródeł zasilania (baterii, akumulatorów) oraz dodatkowo nie powoduje zwiększenia zapotrzebowania na pasmo transmisyjne dla przesyłanych informacji.

4.3.1 Sformułowanie problemu

Problem optymalnej aktywacji czujników skanujących możemy rozważać jako zagadnienie wybrania optymalnego zbioru n czujników spośród N rozmieszczonych w badanym obszarze $\Omega \in \mathbb{R}^2$. Dodatkowo należy wybrać częstość zmian konfiguracji czujników aktywnych. Oznacza to, że horyzont obserwacji $T = (0, t_f]$ dzieli się na rozłączne podprzedziały (etapy) $T_k = (t_{k-1}, t_k]$, $k = 1, \ldots, K$ poprzez arbitralny wybór chwil przełączeń $0 = t_0 < t_1 < \ldots < t_k = t_f$, a w k-tym poprzedziale aktywuje się czujniki znajdujące się w położeniach $x^1(k), \ldots, x^n(k)$, wybranych z ustalonego skończonego zbioru położeń dopuszczalnych X. Wyznaczany plan eksperymentu ma postać

$$\xi = \Big(\xi(1), \ldots, \xi(k)\Big), \tag{4.25}$$

gdzie:

$$\xi(k) = \Big\{x^1(k), x^2(k), \ldots, x^n(k)\Big\}, \quad k = 1, \ldots, K. \tag{4.26}$$

Takie przedstawienie problemu implikuje następującą postać macierzy informacyjnej związanej z aktualnym planem eksperymentu ξ opisującym aktywną konfigurację czujników skanujących

$$\mathcal{M}(\xi) = \sum_{k=1}^{K} \left(\int_{t_{k-1}}^{t_k} G\Big(\xi(k), t\Big) C^{-1}\Big(\xi(k)\Big) G^{\mathsf{T}}\Big(\xi(k), t\Big)\, \mathrm{d}t \right), \tag{4.27}$$

gdzie:

$$G\Big(\xi(k), t\Big) = \Big[g(x^1(k), t) \mid \ \ldots \ \mid g(x^n(k), t)\Big], \tag{4.28}$$

$$g(x, t) = \left[\frac{\partial y(x, t; \theta)}{\partial \theta_1}, \ \ldots \ , \frac{\partial y(x, t; \theta)}{\partial \theta_m}\right]^{\mathsf{T}}_{\theta=\theta^0}, \tag{4.29}$$

opisują wrażliwość stanu na zmianę parametrów, a macierz

$$C\Big(\xi(k)\Big) = \Big[c_{ij}\Big(\xi(k)\Big)\Big], \quad c_{ij}\Big(\xi(k)\Big) = q(x^i(k), x^j(k)), \tag{4.30}$$

opisuje kowariancje obserwacji wykonywanych przez czujniki (szczegóły opisano w rozdz. 2.2).

Dalej przyjmuje się, że przełączenia między konfiguracjami aktywnych czujników następują w chwilach $t_k = k\Delta t$, $\Delta t = (t_f - t_0)/K$. Takie podejście znacznie upraszcza rozwiązanie problemu, nie zmniejszając ogólności rozważań.

4.3.2 Algorytm Tabu-Search

Problem aktywowania K kolejnych, n-elementowych podzbiorów czujników spośród wszystkich N dostępnych czujników rozmieszczonych w obszarze badanego zjawiska stanowi, podobnie jak to było w przypadku czujników stacjonarnych, problem kombinatoryczny. Przestrzeń wszystkich możliwych rozwiązań składa się z

$$\binom{N}{n}^K = \left(\frac{N!}{n!\,(N-n)!}\right)^K \tag{4.31}$$

różnych kombinacji położeń czujników, co w praktyce wyklucza jej pełne przeszukanie. Przykładowo, optymalna aktywacja 6 konfiguracji 21 czujników spośród 441 czujników rozmieszczonych na regularnej siatce kwadratowej 21×21 prowadzi do konieczności sprawdzenia około

$$\left(\frac{441!}{21!\,420!}\right)^6 = 4.89 \cdot 10^{213}, \tag{4.32}$$

różnych konfiguracji aktywnych czujników. Przy założeniu, że w jednej sekundzie można wyznaczyć 10^6 wartości kryterium optymalności dla różnych położeń czujników, czas trwania pełnego przeszukiwania wynosiłby około $4.89 \cdot 10^{207}$ sekund co przekłada się na konieczność wykonywania obliczeń przez około $1.55 \cdot 10^{200}$ lat. Rozwiązaniem, podobnie jak w poprzednim przypadku, może być zastosowanie metody opartej na metaheurystyce powodującej znaczne zmniejszenie przestrzeni poszukiwań optymalnych konfiguracji aktywnych czujników, kosztem popełnienia pewnego błędu wynikającego z przyjętych uproszczeń.

Podobnie jak w zagadnieniu optymalnej aktywacji czujników stacjonarnych, obliczanie macierzy informacyjnej dla wielu konfiguracji wielu aktywnych czujników wymaga przeprowadzenia obliczeń (4.27)–(4.30), a następnie obliczenia wybranego kryterium optymalności na podstawie wyznaczonej macierzy informacyjnej. Te operacje wymagają, oprócz przeliczenia macierzy wrażliwości związanej z aktualną konfiguracją czujników (4.28), przede wszystkim modyfikacji macierzy kowariancji (4.30) oraz, co za tym idzie — macierzy informacyjnej (4.27). Jednak wyznaczanie macierzy $\mathcal{M}$ oraz G a przede wszystkim — odwrotności macierzy kowariancji C, są operacjami bardzo czasochłonnymi. Przykładowo, wyznaczenie macierzy informacyjnej dla konfiguracji 21 aktywnych czujników stacjonarnych z sześcioma etapami czasowymi wymaga sześciokrotnego wyznaczenia odwrotności macierzy o wymiarach 21×21. Potencjalna metoda powinna móc korzystać z metody redukcji czasochłonnych operacji (zob. rozdz. 4.4), podczas wyznaczenia odwrotności macierzy kowariancji, a co za tym idzie — również macierzy informacyjnej.

W celu rozwiązania przedstawianego problemu, można w zasadzie rozważyć adaptację metody GRASP, opracowanej dla problemu optymalnej aktywacji czujników stacjonarnych (zob. rozdz. 4.2). Zdecydowano się jednak na przetestowanie innej metody optymalizacji dyskretnej w celu dokonania porównania własności obu technik.

Poniżej zaproponowano metodę bazującą w swojej budowie na schemacie algorytmu Tabu-Search [71, 72]. Algorytm Tabu-Search należy do klasy algorytmów heurystycznych optymalizacji dyskretnej. Algorytm dąży do optimum globalnego w sposób iteracyjny, przeszukując lokalne sąsiedztwo aktualnego optimum. W celu uniknięcia utknięcia w minimum lokalnym, co pewien czas akceptowane jest rozwiązanie, które daje gorszą wartość optymalizowanej funkcji. Grozi to jednak powrotem do odwiedzanego już minimum, a w konsekwencji do powstawania cykli, kiedy algorytm manewruje między tymi samymi minimami lokalnymi. Procedura zapobiega takiej niepożądanej sytuacji poprzez tworzenie listy ruchów zabronionych (tzw.

TabuList). Lista ta przechowuje minima jakie algorytm osiągał w trakcie swojego działania, a jej długość definiuje się na starcie algorytmu.

Schemat opracowanego algorytmu przedstawiono w postaci Algorytmu 4.4. Jego pracę inicjuje się parametrami opisującymi: maksymalną liczbę iteracji jaką algorytm akceptuje bez polepszenia wartości optimum (tzw. kryterium jakościowe) oraz maksymalną liczbą elementów znajdujących się na liście zabronionych konfiguracji czujników. Zakłada się, że początkowa konfiguracja aktywnych czujników $\xi^{(0)}$ jest zadana arbitralnie lub aktywowana w sposób losowy. W przypadku rozważanego zagadnienia optymalnej aktywacji czujników skanujących, założono brak występowania korelacji czasowych pomiędzy konfiguracjami aktywnych czujników wykonującymi pomiary w różnych chwilach czasowych.

W przypadku rozwiązywania zagadnienia optymalnej aktywacji czujników skanujących, analiza działania algorytmu wskazała na konieczność wprowadzenia pewnych rozszerzeń do klasycznego schematu metody Tabu-Search. Oprócz klasycznej listy konfiguracji zabronionych (TabuList), wprowadzono zbiory sąsiadów zabronionych dla wszystkich etapów planu eksperymentu $\mathcal{Z} = (\mathcal{Z}_1, \ldots, \mathcal{Z}_K)$. Jak już wspomniano, oryginalna lista TabuList zapobiega tworzeniu cykli, w których algorytm oscyluje pomiędzy tymi samymi minimami lokalnymi. Zbiory sąsiadów zabronionych przechowują położenia czujników, które zmodyfikowały plan eksperymentu powodując, że nowo utworzony plan znajduje się na liście TabuList. Przechowywanie tych czujników ma na celu zapobiegnięcie ich ponownemu wybraniu w przypadku, gdy w poprzedniej iteracji ich wybór spowodował utworzenie konfiguracji znajdującej się już w liście TabuList. Innym podejściem spotykanym w literaturze jest losowy wybór pewnego podzbioru sąsiadów z całego sąsiedztwa planu eksperymentu. Takie rozwiązanie potencjalnie zmniejsza prawdopodobieństwo ponownego wyboru czujnika, który tworzy plan znajdujący się na liście TabuList, jednak powoduje również zmniejszenie poziomu eksploracji całkowitej przestrzeni poszukiwań.

Algorytm składa się z dwóch głównych kroków: kroku, w którym następuje określenie czujników, których aktywacja powoduje polepszenie własności opracowywanego planu eksperymentu ze względu na wartość wybranego kryterium optymalności oraz kroku, w którym następuje określenie wartości kryterium optymalności i wykonywane są odpowiednie operacje na liście TabuList i zbiorach sąsiadów zabronionych.

W pierwszym kroku, dla każdego k-tego etapu aktualnego planu eksperymentu wykonywane są kolejno następujące fazy: faza tworzenia oraz modyfikacji sąsiedztwa k-tego etapu aktualnego planu eksperymentu oraz fazy szukania najmniej oraz najbardziej wartościowego czujnika w rozważanym k-tym etapie.

Z utworzonego sąsiedztwa k-tego etapu aktualnego planu eksperymentu usuwane są czujniki wchodzące w skład planu oraz dodatkowo usuwane są czujniki znajdujące się w zbiorze sąsiadów zabronionych. Jeżeli zbiór sąsiadów zabronionych jest niepusty, oznacza to, że w poprzednich iteracjach nastąpiła już analiza planów z wykorzystaniem tych czujników jako nowo aktywowanych, lecz ich wybór spowodował powstanie konfiguracji, które znajdują się na liście TabuList. Czujniki te są usuwane z nowo utworzonego sąsiedztwa, w celu wyłączenia ich z procesu poszukiwania

nowych czujników w rozważanym k-tym etapie aktualnego planu eksperymentu.

Następnie, spośród wszystkich aktywnych czujników wybierany jest czujnik, którego usunięcie powoduje najmniejszą utratę wartości wybranego kryterium optymalności. Kolejnym krokiem jest przeszukanie wcześniej wyznaczonego i zmodyfikowanego sąsiedztwa w poszukiwaniu czujnika, którego aktywacja powoduje wyznaczenie kryterium optymalności o wartości najmniejszej. Na zakończenie, nowo aktywowany czujnik jest dodawany do zbioru sąsiadów zabronionych. Wykonanie powyższych faz w każdym etapie planu eksperymentu kończy pierwszy krok algorytmu i określa kandydata na nowy plan optymalny $\hat{\xi}$.

Następnym krokiem jest sprawdzenie czy nowo wyznaczony plan $\hat{\xi}$ znajduje się na liście konfiguracji zabronionych (TabuList). W przypadku niespełnienia tego warunku (co oznacza, że nowo opracowany plan eksperymentu wcześniej nie był analizowany) sprawdzane jest, czy nowo opracowany plan eksperymentu pozwala poprawić wartość aktualnego przybliżenia optymalnej wartości kryterium optymalności. W przypadku polepszenia wartości kryterium optymalności, nowo opracowany plan eksperymentu staje się nowym przybliżeniem optymalnego planu eksperymentu. W przypadku, gdy lista konfiguracji zabronionych osiągnęła maksymalną dozwoloną wielkość, usuwany jest z niej najstarszy wpis (zgodnie z działaniem kolejki FIFO), a do listy dopisywany jest aktualnie rozpatrywany plan eksperymentu. Dołączanie aktualnie rozpatrywanej konfiguracji aktywnych czujników do listy TabuList ma na celu zapobieganie oscylacjom algorytmu pomiędzy minimami lokalnymi. Następnie, rozpatrywany plan eksperymentu staje się konfiguracją początkową dla kolejnej iteracji algorytmu. Jednocześnie, następuje wyzerowanie zbioru sąsiadów zabronionych, gdyż nowo opracowany plan eksperymentu nie był wcześniej analizowany i nie spowodował powstania oscylacji pomiędzy minimami lokalnymi, co byłoby oznaczane umieszczeniem go na liście TabuList.

W ostatnim kroku zwiększa się licznik iteracji oraz sprawdza czy wykonano maksymalną dozwoloną liczbę iteracji bez polepszenia wartości kryterium optymalności (warunek stopu). W przypadku przekroczenia, ustalonej na początku algorytmu, liczby iteracji nie polepszających uzyskanego wcześnej przybliżenia minimalnej wartości kryterium optymalności, następuje zakończenie działania algorytmu.

Algorytm 4.4 Optymalna aktywacja czujników skanujących

Oznaczenia:

$\mathcal{T}$ — lista konfiguracji zabronionych (TabuList),

$\mathcal{Z} = (\mathcal{Z}_1, \ldots, \mathcal{Z}_K)$ — ciąg zbiorów sąsiadów zabronionych dla K kolejnych konfiguracji,

$\{s\}_k = \big(\underbrace{\emptyset, \ldots, \emptyset}_{k-1 \text{ razy}}, \{s\}, \underbrace{\emptyset, \ldots, \emptyset}_{K-k \text{ razy}}\big)$.

Dane wejściowe:

$\tau_{\max}$ — maksymalna liczba iteracji algorytmu bez polepszenia optimum,

$\varrho_{\max}$ — maksymalna liczba elementów listy TabuList,
$\xi^{(0)} = (\xi^{(0)}(1), \ldots, \xi^{(0)}(K))$ — plan eksperymentu opisujący wstępną konfigurację n aktywnych czujników w K kolejnych konfiguracjach.

Dane wyjściowe:
Ψ_{opt} — obliczone przybliżenie optymalnej wartości kryterium,
ξ_{opt} — znalezione przybliżenie optymalnego planu eksperymentu.

Krok 0: (Inicjalizacja)
Ustawienie przybliżenia optymalnego planu eksperymentu na wartość początkową $\xi_{\text{opt}} \leftarrow \xi^{(0)}$ oraz wyznaczenie wartości kryterium: $\Psi_{\text{opt}} \leftarrow \Psi(\xi^{(0)})$. Inicjalizacja licznika iteracji $\tau \leftarrow 1$ oraz wartości licznika iteracji ostatnio znalezionego optimum $\ell \leftarrow 1$. Wyzerowanie listy konfiguracji zabronionych (TabuList) $\mathcal{T} \leftarrow \emptyset$. Wyzerowanie zbiorów sąsiadów zabronionych $\mathcal{Z} \leftarrow (\underbrace{\emptyset, \ldots, \emptyset}_{K \text{ razy}})$.

Krok 1:
Szukanie czujników, których włączenie do planu pozwoli na poprawę wartości kryterium optymalności.
Dla każdego k-tego etapu $\xi(k)$ aktualnego planu eksperymentu ξ wykonuje się:

Faza 1:
Z utworzonego sąsiedztwa k-tego etapu planu ξ

$$\mathcal{N}_k = \bigcup_{\mathrm{s} \in \xi(k)} \mathcal{N}(s), \tag{4.33}$$

usuwane są czujniki należące do tego planu

$$\bar{\mathcal{N}}_k \leftarrow \mathcal{N}_k \setminus \xi(k) \tag{4.34}$$

Faza 2:
Z sąsiedztwa $\bar{\mathcal{N}}_k$ usuwamy czujniki znajdujące się w zbiorze sąsiadów zabronionych k-tego etapu $\mathcal{Z}_k$

$$\hat{\mathcal{N}}_k \leftarrow \bar{\mathcal{N}}_k \setminus \mathcal{Z}_k. \tag{4.35}$$

Faza 3:
W składowym planie $\xi(k)$ wyznaczany, a następnie usuwany jest czujnik, którego deaktywacja powoduje najmniejszą utratę wartości wybranego kryterium optymalności:

$$\begin{aligned} \bar{\mathrm{s}} &\leftarrow \arg\min_{\mathrm{s} \in \xi(k)} \Psi\big(\xi \setminus \{\mathrm{s}\}_k\big), \\ \bar{\xi} &\leftarrow \xi \setminus \{\bar{\mathrm{s}}\}_k \end{aligned} \tag{4.36}$$

Faza 4:

Wśród czujników znajdujących się w sąsiedztwie $\hat{\mathcal{N}}_k$ poszukuje się takiego, którego aktywacja powoduje największy ubytek wartości kryterium optymalności w zmodyfikowanym planie $\bar{\xi}$.

$$\begin{aligned} \hat{\mathrm{s}} &\leftarrow \arg\min_{\mathrm{s}\in\hat{\mathcal{N}}_k} \Psi\Big(\bar{\xi}\cup\{\mathrm{s}\}_k\Big), \\ \hat{\xi} &\leftarrow \bar{\xi}\cup\{\hat{\mathrm{s}}\}_k, \end{aligned} \tag{4.37}$$

Faza 5:

Do zbioru sąsiadów zabronionych k-tego etapu dodaje się nowo aktywowany czujnik

$$\mathcal{Z} \leftarrow \mathcal{Z}\cup\{\hat{s}\}_k. \tag{4.38}$$

Krok 2:

Jeżeli nowo wyznaczony plan eksperymentu $\hat{\xi}$ znajduje się na liście konfiguracji zabronionych $\mathcal{T}$ to następuje skok do Kroku 3, w przeciwnym przypadku wykonuje się

Faza 1: (Aktualizacja kryterium optymalnego)

Jeżeli wartość kryterium $\Psi(\hat{\xi})$ jest lepsza od Ψ_{opt} to następuje aktualizacja przybliżenia wartości optymalnej $\Psi_{\mathrm{opt}} \leftarrow \Psi(\hat{\xi})$ oraz $\xi_{\mathrm{opt}} \leftarrow \hat{\xi}$.

Faza 2: (Modyfikacja listy konfiguracji zabronionych)

Zapamiętanie iteracji, w której następuje modyfikacja listy konfiguracji zabronionych $\ell \leftarrow \tau$. Jeżeli długość listy Tabu osiągnęła wartość maksymalną $|\mathcal{T}| = \varrho_{\max}$ to następuje usunięcie z niej elementu, który został wpisany najwcześniej.

Faza 3: (Aktualizacja listy konfiguracji zabronionych)

Dołączenie do listy konfiguracji zabronionych nowego planu eksperymentu $\mathcal{T} \leftarrow \mathcal{T}\cup\{\hat{\xi}\}$.

Faza 4: (Ustawienie startowej konfiguracji)

Wyznaczony plan eksperymentu staje się startową konfiguracją w kolejnej iteracji, tzn. $\xi \leftarrow \hat{\xi}$ oraz następuje wyzerowanie zbioru sąsiadów zabronionych $\mathcal{Z} \leftarrow (\underbrace{\emptyset,\ldots,\emptyset}_{K \text{ razy}})$.

Krok 3:

Aktualizacja liczby wykonanych iteracji $\tau \leftarrow \tau + 1$. Jeżeli wykonano maksymalną liczbę iteracji bez polepszenia optimum $(\tau - \ell) \geqslant \tau_{\max}$ to następuje wyjście z algorytmu, w przeciwnym przypadku — skok do Kroku 1.

■

Podobnie jak w algorytmie znajdującym optymalne konfiguracje aktywnych czujników stacjonarnych, również w rozważanym algorytmie najbardziej pracochłonnymi częściami są: faza znajdowania najmniej wartościowego czujnika w aktualnym planie oraz faza poszukiwania najbardziej wartościowego czujnika spośród sąsiedztwa aktualnej konfiguracji. W tych częściach algorytmu zasadnym jest implementacja metody redukcji czasochłonnych operacji opisanej w rozdziale 4.4.

Przykład 4.2
Rozważmy ponownie model adwekcji-dyfuzji wprowadzony w Przykładzie 4.1 i opisujący rozprzestrzenianie się skażenia na danym obszarze. Postać modelu, jego parametry oraz sposób wyznaczenia współczynników wrażliwości pozostały bez zmian. Dynamikę rozprzestrzeniania się skażenia przedstawiono na rys. 4.3.

W celu estymacji wartości parametrów θ rozmieszczono 1681 czujników stacjonarnych na regularnej siatce kwadratowej 41×41. Założono, że w czasie obserwacji zjawiska nastąpi 5 przełączeń aktywnych konfiguracji czujników powodując, że badane zjawisko będzie obserwowane przez 6 konfiguracji aktywnych czujników. Konfiguracje będą inicjowane w chwilach $t_k = k\Delta t, \Delta t = (t_f - t_0)/K, k = 1, \ldots, K$, gdzie $K = 6$, $t_0 = 0$, $t_f = 1$.

Założono, że jednoczesnej aktywacji w każdej konfiguracji podlega 100 czujników spośród wszystkich rozmieszczonych na badanym obszarze. W badaniach uwzględniono występowanie korelacji pomiędzy pomiarami wykonywanymi przez sąsiednie czujniki. Korelacje zostały zdeterminowane przez odległość pomiędzy czujnikami wg wzoru

$$q(x^i(k), x^j(k)) = \sigma^2 \exp\left(\frac{-\|x^i(k) - x^j(k)\|}{\beta}\right), \tag{4.39}$$

gdzie: β — współczynnik determinujący stopień skorelowania obserwacji. W przykładzie przyjęto następujące parametry opisujące korelacje: $\sigma = 1.0, \beta = 0.1$.

Obliczenia przeprowadzono w oparciu o uniwersytecki klaster zbudowany w ramach projektu CLUSTERIX [248] (zob. rozdz. 3.9). Klaster składa się z węzłów zawierających 64-bitowe procesory Intel Itanium^2 1.4GHz i działa pod kontrolą środowiska GNU/Linux Debian for ia64. Program napisano w języku Fortran 95 i skompilowano z zastosowaniem kompilatora *ifort* (Intel®Fortran Compiler v.8.1 for Linux 64-bit platforms). Aplikacja używa biblioteki procedur numerycznych Intel Math Kernel Library do wykonywania operacji mnożenia macierzy oraz wektorów.

Przyjęto sąsiedztwo Moore'a o promieniu $r = 1$. Maksymalna liczba iteracji bez polepszenia optimum $\tau_{\max}$ została ustawiona na 15, a maksymalną długość listy TabuList, tzn. $\varrho_{\max}$ ustawiono na 30 konfiguracji. Wyznaczone konfiguracje 100 czujników w 6 konfiguracjach przedstawiono na rys. 4.5. Czujniki skanujące są aktywowane w sposób uzależniany dynamiką rozważanego procesu rozprzestrzeniania się zanieczyszczeń. Jak można zauważyć, optymalna aktywacja czujników stacjonarnych (zob. rys. 4.4) stanowi pewne uśrednienie konfiguracji uzyskanych przy optymalnej aktywacji czujników skanujących. Wraz ze wzrostem liczby dopuszczalnych konfiguracji, skanująca sieć sensoryczna może dostosować konfigurację aktywnych

czujników do dynamiki obserwowanego procesu. Takie właściwości sieci czujników skanujących pozwalają na dokładniejszą analizę badanego zjawiska, a dzięki temu — na dokładniejsze wyznaczenie estymat nieznanych parametrów.

Obliczenia mające na celu wyznaczenie optymalnych konfiguracji trwały około 2178 sekund, co, podobnie jak w przypadku optymalnej aktywacji czujników stacjonarnych, stanowiło motywację do opracowania równoległej metody pozwalającej na szybsze uzyskiwanie rezultatów.

♦

4.4 Problemy obliczeniowe

W przedstawionych algorytmach większość operacji związanych z przeliczaniem macierzy informacyjnej polega na deaktywacji jednego z czujników w aktualnym planie eksperymentu oraz na aktywacji nowego czujnika w bieżącym planie. W literaturze [228, 229] zaproponowano technikę pozwalającą na znaczne zredukowanie liczby czasochłonnych operacji wykonywanych podczas uaktualnień macierzy informacyjnej. Pomysł polega na zastosowaniu wzoru Frobeniusa (zob. dodatek A.5) opartego na wykorzystaniu formuły Shermana-Morrisona-Woodbury'ego [83]. Wspomniana redukcja opiera się na wyeliminowaniu najbardziej czasochłonnych operacji, tzn. odwracania macierzy o dużych rozmiarach przy wyznaczaniu macierzy informacyjnej (zob. rozdz. 2.2.2) postaci

$$\mathcal{M}(\xi) = \int_T G(\xi,t)C^{-1}(\xi,t)G^{\mathsf{T}}(\xi,t)\,\mathrm{d}t, \tag{4.40}$$

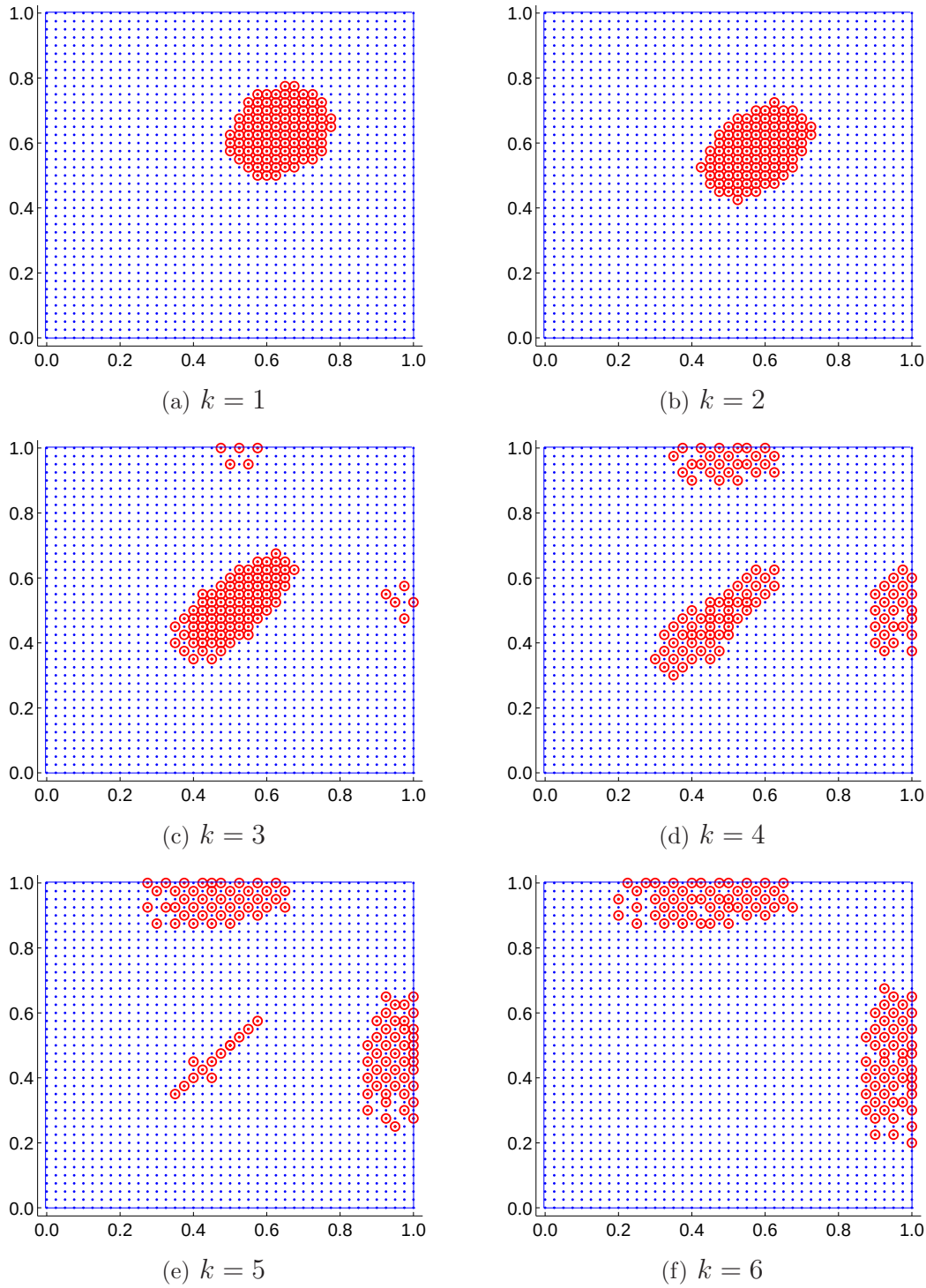
gdzie:

$$G(\xi,t) = \left[g(x^1(t),t) \mid \ \ldots \ \mid g(x^n(t),t)\right]. \tag{4.41}$$

$$C(\xi,t) = [c_{ij}(\xi,t)], \quad c_{ij}(\xi,t) = q(x^i(t),x^j(t)). \tag{4.42}$$

Dla tak przedstawionego problemu możemy wyszczególnić najbardziej pracochłonne operacje jakimi są: obliczenie odwrotności macierzy kowariancji $W = C^{-1}$ oraz wyznaczenie macierzy informacyjnej $\mathcal{M}$. Operacje te pojawiają się w momencie szukania czujnika, którego wyłączenie powoduje najmniejszą utratę wartości kryterium optymalności oraz w momencie szukania czujnika, którego aktywacja pozwoli na polepszenie kryterium optymalności. Operacje poszukiwania czujników oraz aktywacji polegają na wielokrotnym wyznaczaniu macierzy informacyjnej dla różnych konfiguracji czujników. Te operacje implikują konieczność wielokrotnego odwracania macierzy kowariancji C o dużych rozmiarach, co jest operacją bardzo czasochłonną. Przekształcenia macierzowe oparte na wzorze Frobeniusa pozwalają na znaczne zmniejszenie liczby operacji wykorzystywanych w obliczeniach.

Zagadnienie optymalnej aktywacji czujników stacjonarnych można traktować jako pewien specjalny przypadek problemu optymalnej aktywacji czujników skanujących w momencie, gdy rozważamy pojedynczą konfigurację sieci czujników występującą przez cały okres wykonywania pomiarów. Dlatego poniższe wyprowadzenia



(a) $k = 1$ (b) $k = 2$

(c) $k = 3$ (d) $k = 4$

(e) $k = 5$ (f) $k = 6$

Rysunek 4.5: Optymalne konfiguracje czujników skanujących dla Przykładu 4.2.

oparte są na rozważaniach dotyczących redukcji operacji w problemie optymalnej aktywacji czujników skanujących i mogą być w trywialny sposób zaadaptowane dla

problemu aktywacji czujników stacjonarnych (wystarczy przyjąć $K = 1$).

Uogólniając wyprowadzenia dla problemu optymalnej aktywacji czujników skanujących, zdefiniujmy horyzont czasowy obserwacji $T = (0, t_f]$ podzielony na podprzedziały $T_k = (t_{k-1}, t_k], k = 1, \ldots, K$ poprzez arbitralny wybór chwil przełączeń konfiguracji aktywnych czujników $0 = t_0 < t_1 < \ldots < t_K = t_f$.

Plan eksperymentu dla problemu optymalnej aktywacji czujników skanujących przedstawia się następująco

$$\xi = \Big\{\xi(1), \ldots, \xi(k)\Big\}, \tag{4.43}$$

gdzie:

$\xi(k) = \Big\{x^1(k), \ldots, x^n(k)\Big\}$ — konfiguracje n aktywnych czujników dla k-tego przedziału czasowego,

$k = 1, \ldots, K$ — indeks kolejnych etapów obserwacji.

Zauważmy, że macierz informacyjna $\mathcal{M}(\xi)$ jest addytywna względem poszczególnych etapów skanowania, tzn.

$$\mathcal{M}(\xi) = \sum_{k=1}^{K} \mathcal{M}_k\Big(\xi(x)\Big), \tag{4.44}$$

gdzie:

$$\mathcal{M}_k\Big(\xi(k)\Big) = \int_{t_{k-1}}^{t_k} G\Big(\xi(k), t\Big) C^{-1}\Big(\xi(k)\Big) G^{\mathsf{T}}\Big(\xi(k), t\Big)\, \mathrm{d}t. \tag{4.45}$$

4.4.1 Deaktywacja czujnika w planie

Operacja deaktywacji czujnika polega na usunięciu czujnika z planu eksperymentu. W przypadku Algorytmów 4.1–4.3 oraz 4.4 jest to czujnik, którego wyłączenie powoduje najmniejszy ubytek wartości kryterium optymalności. Nie ograniczając ogólności rozważań, przedstawiona metoda redukcji wymaga, aby deaktywowany czujnik był ostatnim czujnikiem w planie eksperymentu, tzn. usuwamy czujnik x^n jeżeli plan eksperymentu jest definiowany jako $\xi = \{x^1, \ldots, x^n\}$. W przypadku, gdy wymagane jest wyłączenie czujnika znajdującego się na pozycji innej niż ostatnia, należy przekształcić macierze kowariancji i wrażliwości tak, aby wybrany czujnik znajdował się na końcu, np. w przypadku deaktywacji czujnika znajdującego się na i-tej pozycji w planie eksperymentu ξ modyfikacja macierzy wrażliwości G wymaga zamiany miejscami i-tej i n-tej kolumny, a modyfikacja macierzy kowariancji C wymaga zamiany miejscami i-tej i n-tej kolumny oraz i-tego wiersza z n-tym.

Zdefiniujmy nowy plan eksperymentu $\xi_n(k) = \{x^1(k), \ldots, x^n(k)\}$ dla konfiguracji sieci czujników wykonującej pomiary w przedziale czasowym $T_k = (t_{k-1}, t_k]$ odpowiednim dla k-tej konfiguracji sieci czujników. Z planu $\xi_n(k)$ usuwamy czujnik $x^n(k)$ będący ostatnim czujnikiem w planie. Odpowiednia macierz wrażliwości związana z aktualnym planem $\xi_n(k)$ ma postać

$$G\Big(\xi_n(k), t\Big) = \left[\begin{array}{c:c} G_{rk} & g_{rk} \end{array}\right], \tag{4.46}$$

gdzie: $g_r = g(x^n(k), t)$ — wektor wrażliwości związany z deaktywowanym czujnikiem x^n w przedziale czasowym T_k. Macierz kowariancji związaną z planem $\xi_n(k)$ i jej odwrotność można przedstawić następująco:

$$C\Big(\xi_n(k)\Big) = \left[\begin{array}{c:c} C_{rk} & \upsilon_{rk} \\ \hdashline \upsilon_{rk}^{\mathsf{T}} & \alpha_{rk} \end{array}\right], \quad W\Big(\xi_n(k)\Big) = \left[\begin{array}{c:c} V_{rk} & b_{rk} \\ \hdashline b_{rk}^{\mathsf{T}} & \gamma_{rk} \end{array}\right], \tag{4.47}$$

gdzie:

$C_{rk}, V_{rk} \in \mathbb{R}^{(n-1)\times(n-1)}$ — macierze symetryczne,

$\upsilon_{rk} \in \mathbb{R}^{n-1}$ — wektor kowariancji związanych z deaktywowanym czujnikiem w k-tej konfiguracji,

$\alpha_{rk} \in \mathbb{R}$ — wariancja zakłóceń pomiarowych deaktywowanego czujnika,

$b_{rk} \in \mathbb{R}^{n-1}$, $\gamma_{rk} \in \mathbb{R}$.

Macierz $C\Big(\xi_n(k)\Big)$ jest odwrotnością macierzy $W\Big(\xi_n(k)\Big)$, a stąd po zastosowaniu formuły Frobeniusa można ją przedstawić jako

$$\begin{aligned} C\Big(\xi_n(k)\Big) &= W^{-1}\Big(\xi_n(k)\Big) \\ &= \left[\begin{array}{c:c} V_{rk} & b_{rk} \\ \hdashline b_{rk}^{\mathsf{T}} & \gamma_{rk} \end{array}\right]^{-1} = \left[\begin{array}{c:c} W_{rk}^{-1} & -\dfrac{1}{\gamma_{rk}} W_{rk}^{-1} b_{rk} \\ \hdashline -\dfrac{1}{\gamma_{rk}} b_{rk}^{\mathsf{T}} W_{rk}^{-1} & \dfrac{1}{\gamma_{rk}} + \dfrac{1}{\gamma_{rk}^2} b_{rk}^{\mathsf{T}} W_{rk}^{-1} b_{rk} \end{array}\right], \end{aligned} \tag{4.48}$$

gdzie:

$$W_{rk} = V_{rk} - \frac{1}{\gamma_{rk}} b_{rk} b_{rk}^{\mathsf{T}}. \tag{4.49}$$

Jak można zauważyć w równaniach (4.47) i (4.48), zachodzi $C_{rk} = W_{rk}^{-1}$, co implikuje $W_{rk} = C_{rk}^{-1}$, czyli macierz W_{rk} jest odwrotnością macierzy kowariancji dla konfiguracji czujników $\xi_{n-1}(k)$ z wyłączonym czujnikiem x^n na k-tym etapie.

Zgodnie z formułą Frobeniusa, operacje związane z wyznaczeniem macierzy informacyjnej $\mathcal{M}\big(\xi_n(k)\big)$ związanej z konfiguracją sieci czujników w przedziale $T_k = (t_{k-1}, t_k]$ można przedstawić następująco:

$$\begin{aligned}
&G\big(\xi_n(k), t\big) W\big(\xi_n(k)\big) G^{\mathsf{T}}\big(\xi_n(k), t\big) \\
&\qquad = \left[\begin{array}{c:c} G_{rk} & g_{rk} \end{array}\right] \left[\begin{array}{c:c} V_{rk} & b_{rk} \\ \hdashline b_{rk}^{\mathsf{T}} & \gamma_{rk} \end{array}\right] \left[\begin{array}{c} G_{rk}^{\mathsf{T}} \\ \hdashline g_{rk}^{\mathsf{T}} \end{array}\right] \\
&\qquad = \left[\begin{array}{c:c} G_{rk} & g_{rk} \end{array}\right] \left[\begin{array}{c:c} W_{rk} + \dfrac{1}{\gamma_{rk}} b_{rk} b_{rk}^{\mathsf{T}} & b_{rk} \\ \hdashline b_{rk}^{\mathsf{T}} & \gamma_{rk} \end{array}\right] \left[\begin{array}{c} G_{rk}^{\mathsf{T}} \\ \hdashline g_{rk}^{\mathsf{T}} \end{array}\right] \\
&\qquad = G_{rk} W_{rk} G_{rk}^{\mathsf{T}} + \frac{1}{\gamma_{rk}} G_{rk} b_{rk} b_{rk}^{\mathsf{T}} G_{rk}^{\mathsf{T}} \\
&\qquad\quad + G_{rk} b_{rk} g_{rk}^{\mathsf{T}} + g_{rk} b_{rk}^{\mathsf{T}} G_{rk}^{\mathsf{T}} + \gamma_{rk} g_{rk} g_{rk}^{\mathsf{T}} \\
&\qquad = G_{rk} W_{rk} G_{rk}^{\mathsf{T}} + h_{rk} h_{rk}^{\mathsf{T}},
\end{aligned} \tag{4.50}$$

gdzie:

$$h_{rk} = h_{rk}\big(\xi_n(k), t\big) = \frac{1}{\sqrt{\gamma_{rk}}} G_{rk} b_{rk} + \sqrt{\gamma_{rk}}\, g_{rk}, \tag{4.51}$$

Na podstawie (4.50)–(4.51) macierz informacyjną $\mathcal{M}_k\big(\xi_{n-1}(k)\big)$, z uwzględnieniem zmian wprowadzonych przez deaktywację czujnika w k-tym etapie procesu skanowania, można wyrazić jako

$$\mathcal{M}_k\big(\xi_{n-1}(k)\big) = \mathcal{M}_k\big(\xi_n(k)\big) - \int_{t_{k-1}}^{t_k} h_{rk}\big(\xi_n(k), t\big) h_{rk}^{\mathsf{T}}\big(\xi_n(k), t\big)\, \mathrm{d}t, \tag{4.52}$$

a macierz W_{rk} będąca odwrotnością macierzy korelacji oblicza się wg wzoru (4.49).

W przypadku deaktywowania pojedynczych czujników w każdym etapie skanowania otrzymuje się plan eksperymentu $\xi_{n-1} = \big(\xi_{n-1}(1), \ldots, \xi_{n-1}(k)\big), k = 1, \ldots, K$ dla którego ostateczna forma macierzy informacyjnej przedstawia się następująco:

$$\mathcal{M}(\xi_{n-1}) = \mathcal{M}(\xi_n) - \sum_{k=1}^{K} \int_{t_{k-1}}^{t_k} h_{rk}\big(\xi_n(k), t\big) h_{rk}^{\mathsf{T}}\big(\xi_n(k), t\big)\, \mathrm{d}t. \tag{4.53}$$

4.4.2 Aktywacja nowego czujnika

Aktywacja nowego czujnika x^{n+1} polega na dodaniu go do planu eksperymentu. Zdefiniujmy nowy plan eksperymentu $\xi_{n+1}(k) = \{x^1(k), \ldots, x^n(k), x^{n+1}(k)\}$ dla konfiguracji sieci czujników wykonującej pomiary w przedziale czasowym $T_k = (t_{k-1}, t_k]$

odpowiednim dla k-tej konfiguracji sieci czujników. Odpowiednia macierz wrażliwości ma postać

$$G\Big(\xi_{n+1}(k), t\Big) = \left[\begin{array}{c:c} G_{rk} & g_{ak} \end{array}\right], \tag{4.54}$$

gdzie: $g_{ak} = g\Big(x^{n+1}(k), t\Big)$ — wektor wrażliwości związany z pozycją x^{n+1} aktywowanego czujnika w k-tej konfiguracji sieci czujników. Nowa zdekomponowana macierz kowariancji przyjmuje formę

$$C\Big(\xi_{n+1}(k)\Big) = C_{ak} = \left[\begin{array}{c:c} C_{rk} & \upsilon_{ak} \\ \hdashline \upsilon_{ak}^{\mathsf{T}} & \alpha_{ak} \end{array}\right]. \tag{4.55}$$

gdzie:

$\alpha_{ak} = q\Big(x^{n+1}(k), x^{n+1}(k)\Big)$ — wariancja sygnału wyjściowego nowo dodanego czujnika,

$\upsilon_{ak} = \Big[q\Big(x^{n+1}(k), x^{1}(k)\Big), \ldots, q\Big(x^{n+1}(k), x^{n}(k)\Big)\Big]^{\mathsf{T}}$ — wektor kowariancji wyjścia nowo aktywowanego czujnika i wyjść dotychczas aktywnych czujników w k-tym etapie obserwacji.

Zdefiniujmy zmienną pomocniczą

$$\begin{aligned} \gamma_{ak} &= \frac{1}{\alpha_{ak} - \upsilon_{ak}^{\mathsf{T}} C_{rk}^{-1} \upsilon_{ak}} \\ &= \frac{1}{q\Big(x^{n+1}(k), x^{n+1}(k)\Big) - \upsilon_{ak}^{\mathsf{T}} W_{rk} \upsilon_{ak}}, \end{aligned} \tag{4.56}$$

Na podstawie formuły Frobeniusa (zob. dodatek A.5) można zauważyć, że odwrotność macierzy kowariancji związanej z k-tą konfiguracją po aktywowaniu nowego czujnika przedstawia się następująco:

$$\begin{aligned} W\Big(\xi_{n+1}(k)\Big) &= C^{-1}\Big(\xi_{n+1}(k)\Big) \\ &= \left[\begin{array}{c:c} C_{rk}^{-1} + \gamma_{ak} C_{rk}^{-1} \upsilon_{ak} \upsilon_{ak}^{\mathsf{T}} C_{rk}^{-1} & -\gamma_{ak} C_{rk}^{-1} \upsilon_{ak} \\ \hdashline -\gamma_{ak} \upsilon_{ak}^{\mathsf{T}} C_{rk}^{-1} & \gamma_{ak} \end{array}\right] \\ &= \left[\begin{array}{c:c} W_{rk} + \dfrac{1}{\gamma_{ak}} b_{ak} b_{ak}^{\mathsf{T}} & b_{ak} \\ \hdashline b_{ak}^{\mathsf{T}} & \gamma_{ak} \end{array}\right]. \end{aligned} \tag{4.57}$$

gdzie: $b_{ak} = -\gamma_{ak} C_{rk}^{-1} \upsilon_{ak} = -\gamma_{ak} W_{rk} \upsilon_{ak}$.

Operację związane z wyznaczeniem macierzy informacyjnej $\mathcal{M}_k\Big(\xi_{n+1}(k)\Big)$ związanej z pomiarami w przedziale czasowym $T_k = (t_{k-1}, t_k]$ po aktywacji nowego czuj-

nika można wyrazić jako

$$\begin{aligned} & G\Big(\xi_{n+1}(k),t\Big)W\Big(\xi_{n+1}(k)\Big)G^{\mathsf{T}}\Big(\xi_{n+1}(k),t\Big) \\ & \qquad = \left[\begin{array}{c:c} G_{rk} & g_{ak} \end{array}\right] \left[\begin{array}{c:c} W_{rk}+\dfrac{1}{\gamma_{ak}}b_{ak}b_{ak}^{\mathsf{T}} & b_{ak} \\ \hdashline b_{ak}^{\mathsf{T}} & \gamma_{ak} \end{array}\right] \left[\begin{array}{c} G_{rk}^{\mathsf{T}} \\ \hdashline g_{ak}^{\mathsf{T}} \end{array}\right] \\ & \qquad = G_{rk}W_{rk}G_{rk}^{\mathsf{T}} + \frac{1}{\gamma_{ak}}G_{rk}b_{ak}b_{ak}^{\mathsf{T}}G_{rk}^{\mathsf{T}} \\ & \qquad\quad + G_{rk}b_{ak}g_{ak}^{\mathsf{T}} + g_{ak}b_{ak}^{\mathsf{T}}G_{ak}^{\mathsf{T}} + \gamma_{ak}g_{ak}g_{ak}^{\mathsf{T}} \\ & \qquad = G_{rk}W_{rk}G_{rk}^{\mathsf{T}} + h_{ak}h_{ak}^{\mathsf{T}}, \end{aligned} \tag{4.58}$$

gdzie:

$$h_{ak} = h_{ak}\Big(\xi_{n+1}(k),t\Big) = \frac{1}{\sqrt{\gamma_{ak}}}G_{rk}b_{ak} + \sqrt{\gamma_{ak}}g_{ak}. \tag{4.59}$$

Na podstawie (4.58)–(4.59), macierz informacyjną $\mathcal{M}_k\Big(\xi_{n+1}(k)\Big)$, z uwzględnieniem zmian wprowadzonych przez aktywację nowego czujnika w k-tej konfiguracji sieci czujników, można wyrazić w postaci

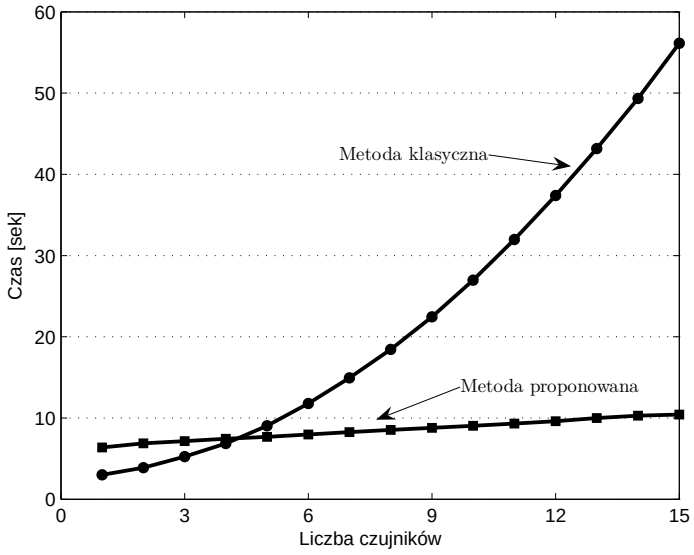
$$\mathcal{M}_k\Big(\xi_{n+1}(k)\Big) = \mathcal{M}_k\Big(\xi_n(k)\Big) + \int_{t_{k-1}}^{t_k} h_{ak}\Big(\xi_{n+1}(k),t\Big)h_{ak}^{\mathsf{T}}\Big(\xi_{n+1}(k),t\Big)\,\mathrm{d}t. \tag{4.60}$$

W przypadku aktywowania pojedynczych nowych czujników w każdym etapie skanowania, otrzymuje się plan eksperymentu $\xi_{n+1} = \Big\{\xi_{n+1}(1),\ldots,\xi_{n+1}(k)\Big\}, k = 1,\ldots,K$ dla którego ostateczna forma macierzy informacyjnej przedstawia się następująco:

$$\mathcal{M}(\xi_{n+1}) = \mathcal{M}(\xi_n) + \sum_{k=1}^{K}\int_{t_{k-1}}^{t_k} h_{ak}\Big(\xi_{n+1}(k),t\Big)h_{ak}^{\mathsf{T}}\Big(\xi_{n+1}(k),t\Big)\,\mathrm{d}t. \tag{4.61}$$

4.4.3 Podsumowanie

Zastosowanie metody redukcji czasochłonnych operacji pozwala na znaczne skrócenie czasu obliczeń w przypadku algorytmów korzystających bardzo często z operacji usuwania i dodawania czujników do planu. Jako przykład rozważmy model dyfuzji opisany w Przykładzie 2.1. Zadaniem było optymalne aktywowanie kolejnych czujników stacjonarnych rozmieszczonych na rozważanym obszarze w postaci regularnej siatki 21×21. Czujniki były aktywowane z zastosowaniem Algorytmu 4.1 traktowanego jako algorytm sekwencyjny. Zysk z zastosowania metody redukcji obliczeń przedstawiono na rys. 4.6. Już dla małej liczby czujników widać znaczne korzyści z zastosowania metody redukcji w postaci krótszych czasów wykonywania obliczeń.



Rysunek 4.6: Zysk z wykorzystania metody redukcji czasochłonnych operacji.

4.5 Zrównoleglenie algorytmów

Problem zrównoleglania algorytmów, na których bazowały przedstawione metody rozwiązywania problemów optymalnej obserwacji, jest stosunkowo szeroko omawiany w literaturze. Ciekawe zestawienie algorytmów optymalizacji globalnej zarówno deterministycznych, jak i heurystycznych oraz meta-heurystycznych, można znaleźć w pracy [110]. Metody GRASP omawiano w [2, 59, 168, 191, 192], a algorytm Tabu-Search stanowił w tym samym kontekście przedmiot prac [15, 214]. Rozważano m.in. metody zrównoleglania oparte na równoczesnym uruchamianiu algorytmów na wielu procesorach dla różnych parametrów startowych (*ang.* multi-start) [15, 168], a następnie wyborze rozwiązania odpowiadającego najmniejszej wartości funkcji kosztu, jak również metody oparte na dekompozycji przestrzeni poszukiwań [59, 214], w których następował podział np. zbioru sąsiadów rozważanego rozwiązania, a następnie równoległa analiza tak podzielonych zbiorów przez wiele procesorów.

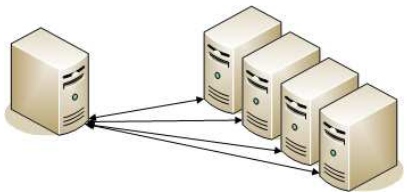
Przedstawione niżej podejście do zrównoleglania zaprezentowanych algorytmów skupia się na jak największym dostosowaniu algorytmów do rozważanych zagadnień optymalnej obserwacji układów z czasoprzestrzenną dynamiką. Takie podejście pozwala na uzyskanie możliwie najlepszych efektów bazując na specyfice rozważanych problemów.

Aby zrównoleglić przedstawione algorytmy, należy wybrać miejsce w algorytmie, które pozwala na uzyskanie maksymalnego przyspieszenia, nie powodując przy tym, powstania nadmiernego narzutu związanego z komunikacją. W obydwu algorytmach takim miejscem jest wyznaczanie wartości wybranego kryterium optymalności w zależności od różnych konfiguracji aktywnych czujników. W algorytmie szukającym

optymalnych konfiguracji aktywnych czujników stacjonarnych wyznaczanie wartości kryterium optymalności występuje zarówno w fazie tworzenia rozwiązania przybliżonego, jak i w fazie optymalizacji lokalnej. Algorytm określający optymalne konfiguracje aktywnych czujników skanujących posiada jedno miejsce, w którym następuje znaczny wzrost ilości obliczeń związanych z wyznaczaniem kryterium optymalności na podstawie analizowanego planu eksperymentu. Najbardziej pracochłonną operacją jest przeszukiwanie sąsiedztwa aktualnego planu eksperymentu w poszukiwaniu konfiguracji czujników o lepszej wartości kryterium optymalności.

4.5.1 Schemat „nadzorca-podwładni"

W zagadnieniu zrównoleglenia algorytmu przedstawionego w rozdz. 4.2 wykorzystano schemat komunikacji „nadzorca-podwładni". Oznacza to, że algorytm wyszczególnia jeden procesor jako procesor główny — nadzorcę (*ang.* master) — a reszta procesorów — podwładnych (*ang.* slaves) — odpowiada za wykonywanie obliczeń. Schematycznie taką strukturę przedstawia rys. 4.7. Takie zaprojektowanie komunikacji pomiędzy węzłami powoduje, że węzeł główny posiada całą informację o stanie algorytmu oraz czasie jakiego potrzebowały węzły na wykonanie obliczeń. Procesor główny kontroluje również typ oraz kolejność wykonywania obliczeń przez poszczególne procesory obliczeniowe.



Rysunek 4.7: Schemat komunikacji „nadzorca–podwładni".

W celu zaprojektowania efektywnego algorytmu rozwiązującego problem optymalnej aktywacji czujników stacjonarnych wyróżniono fazy algorytmu, w których obliczenia pochłaniają najwięcej czasu oraz mocy obliczeniowej. Są to fazy, podczas których następuje tworzenie rozwiązania pseudozachłannego poprzez eksplorację przestrzeni rozwiązań, mającego być bazą dla fazy przeszukiwania lokalnego, analiza podczas przeszukiwania lokalnego, mająca na celu wyłączenie czujnika mającego najmniejszy wpływ na wartość kryterium optymalności oraz aktywacja nowego czujnika, który pozwoli na uzyskanie największego przyrostu tego samego kryterium.

Po wykonaniu serii obliczeń związanych z utworzeniem nowego przybliżenia planu optymalnego, następuje analiza czasów wykonywania obliczeń przez poszczególne procesory liczące. Dzięki czasom obliczeń, które zostały przesłane do procesora nadzorcy, może on wyznaczyć nowe wartości wag w dla wszystkich procesorów liczących zgodnie z ich obciążeniem. Dokładniejsze informacje o metodzie równoważenia obciążenia przedstawiono w rozdz. 4.5.3.

Zrównoleglenie fazy konstrukcji rozwiązania wstępnego

Zrównoleglenie fazy konstrukcji rozwiązania wstępnego polega na przesłaniu przez procesor główny do procesorów liczących pozycji czujników, które należy zbadać pod kątem ich umieszczenia w planie pseudozachłannym.

Jeżeli przez X oznaczymy skończony zbiór wszystkich pozycji w jakich rozmieszczono czujniki sieci sensorycznej, a ξ oznaczać będzie dyskretny plan eksperymentu (czyli zbiór pozycji czujników aktywnych) to

$$E = X \setminus \xi \tag{4.62}$$

będzie oznaczać zbiór położeń czujników uśpionych.

Podzielmy zbiór E na rozłączne podzbiory $E_1, \ldots, E_P$, gdzie P — liczba procesorów „podwładnych" w systemie równoległym. Oznacza to, że zachodzi

$$E = \bigcup_{i=1}^{P} E_i \quad \text{oraz} \quad E_i \cap E_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.63}$$

Liczność zbioru E_i jaka ma być przydzielona każdemu procesorowi jest zdefiniowana następująco:

$$|E_i| = \left\lfloor w_i |E| \right\rfloor, \quad i = 1, \ldots, P, \tag{4.64}$$

gdzie: w_i — waga i-tego procesora wyznaczana przez algorytm dynamicznego równoważenia obciążenia lub ustalana arbitralnie, $\lfloor \cdot \rfloor$ — zaokrąglenie wartości liczbowej w dół. W przypadku gdy, moc podzbioru wynikająca z (4.64) nie jest liczbą naturalną, zostaje ona zaokrąglona w dół, a części ułamkowe pozostające po zaokrągleniu zostają zsumowane i przydzielone aktualnie najszybszemu procesorowi. Dokładniejsze informacje przedstawiono w rozdz. 4.5.3.

Dokładny schemat działania fazy konstrukcji pseudozachłannego rozwiązania wstępnego, zarówno dla węzła głównego, jak i węzłów liczących przedstawiono w postaci Algorytmów 4.5 oraz 4.6.

Algorytm 4.5 Faza konstrukcji rozwiązania wstępnego — procesor główny/nadzorca

Dane wejściowe:

α — parametr sterujący tworzeniem listy RCL czujników kandydujących do aktywacji,
$\xi^{(\tau-1)}$ — plan eksperymentu opisujący bieżącą konfigurację aktywnych czujników,
ℓ — parametr określający liczbę elementów planu $\xi^{(\tau-1)}$ uwzględnianych w fazie konstrukcji rozwiązania wstępnego $\hat{\xi}^{(\tau)}$,
$w^{(\tau-1)} = \{w_1^{(\tau-1)}, \ldots, w_P^{(\tau-1)}\}$ — wektor wag dla poszczególnych procesorów.

Dane wyjściowe:
$\hat{\xi}^{(\tau)}$ — wstępne przybliżenie planu kandydującego na kolejne przybliżenie planu optymalnego.

Krok 0: (Inicjalizacja)
Losowy wybór ℓ elementów planu $\xi^{(\tau-1)}$, które stanowią plan ζ_ℓ. Inicjalizacja licznika iteracji $\kappa \leftarrow \ell$.

Krok 1: (Wyznaczenie zakresu zmienności kryterium podczas aktywacji dodatkowego czujnika)
Wyznaczenie najmniejszej i największej wartości kryterium optymalności, jakie można uzyskać dołączając do planu ζ_κ pojedynczy punkt spośród położeń czujników uśpionych, przez procesory liczące.

Faza 1: (Podział zbioru czujników uśpionych pomiędzy procesory)
Podział zbioru czujników uśpionych pomiędzy procesory tak, aby

$$E = X \setminus \zeta_\kappa, \tag{4.65}$$

$$E = \bigcup_{i=1}^{P} E_i \quad \text{oraz} \quad E_i \cap E_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.66}$$

a liczność zbiorów $E_1, \ldots, E_P$ wynosiła

$$|E_i| = \left\lfloor w_i^{(\tau-1)} |E| \right\rfloor, \quad i = 1, \ldots, P, \tag{4.67}$$

Faza 2: (Wysłanie aktualnego planu eksperymentu do procesorów liczących)
Wysłanie aktualnego planu eksperymentu ζ_κ do wszystkich procesorów liczących.

Faza 3: (Wysłanie podzbiorów $E_1, \ldots, E_P$ do procesorów liczących)
Wysłanie do odpowiednich procesorów podzbiorów $E_1, \ldots, E_P$ zawierających zbiór czujników uśpionych, których aktywację w aktualnym planie bazowym ζ_κ należy zbadać.

Faza 4: (Odebranie wyników obliczeń)
Odebranie wyznaczonych wartości kryteriów optymalności dla różnych konfiguracji aktywowanych czujników opisanych przez elementy wysłanych zbiorów $E_1, \ldots, E_P$. Wartości kryteriów obliczone przez każdy z procesorów umieszczone zostają w tablicach $\tilde{\Psi}_1, \ldots, \tilde{\Psi}_P$.

Wyznaczenie najmniejszej i największej wartości kryterium optymalności wyliczanej na podstawie danych odebranych z procesorów liczących.

$$\begin{aligned} \Psi_{\min} &\leftarrow \min_{i=1,\ldots,P} \min \tilde{\Psi}_i, \\ \Psi_{\max} &\leftarrow \max_{i=1,\ldots,P} \max \tilde{\Psi}_i, \end{aligned} \tag{4.68}$$

Krok 2: (Utworzenie listy RCL kandydatów do jednopunktowego powiększenia dotychczasowego planu)
Określenie listy RCL położeń czujników, których włączenie do planu eksperymentu zagwarantuje polepszenie wartości kryterium w granicach określonych przez $\Psi_{\min}$, $\Psi_{\max}$ oraz α, zgodnie ze wzorem

$$\mathrm{RCL} = \left\{ \mathrm{e} \in E \;\middle|\; \Psi\Big(\zeta_\kappa \cup \{\mathrm{e}\}\Big) \leqslant \Psi_{\min} + \alpha(\Psi_{\max} - \Psi_{\min}) \right\}. \tag{4.69}$$

Krok 3: (Ostateczny wybór kolejnego czujnika do konstruowanego planu)
Z utworzonej listy RCL wybiera się losowo element $\{s_0\}$, który zostaje dołączony do dotychczasowego planu:

$$\zeta_{\kappa+1} \leftarrow \zeta_\kappa \cup \{s_0\}. \tag{4.70}$$

Jeżeli $\kappa = n - 1$, przyjmuje się $\hat{\xi}^{(\tau)} \leftarrow \zeta_{\kappa+1}$ i kończy pracę. W przeciwnym razie zwiększa się licznik κ i następuje powrót do Kroku 1.

■

Algorytm 4.6 Faza konstrukcji rozwiązania wstępnego — procesor liczący

Dane wejściowe:
p — unikalny identyfikator procesora liczącego taki, że $1 \leqslant p \leqslant P$, nadawany przez środowisko programowania równoległego.

Krok 1: (Odebranie danych z procesora głównego)
Odebranie aktualnego planu eksperymentu ζ_κ z procesora głównego oraz zbioru czujników uśpionych E_p.

Krok 2: (Obliczenie wartości kryterium optymalności)
Obliczenie wartości kryterium optymalności po aktywacji pojedynczego czujnika uśpionego zawartego w zbiorze E_p i zapamiętanie jej w tablicy $\tilde{\Psi}_p$

$$\tilde{\Psi}_p(\mathrm{e}) = \Psi\Big(\zeta_\kappa \cup \{\mathrm{e}\}\Big), \tag{4.71}$$

dla wszystkich $\mathrm{e} \in E_p$.

Krok 3: (Wysłanie obliczonych wartości kryterium optymalności do procesora głównego)
Wysłanie tablicy $\tilde{\Psi}_p$ do procesora głównego.

■

Zrównoleglenie fazy przeszukiwania lokalnego

W fazie przeszukiwania lokalnego obliczenia wykonuje się w oparciu o aktualny plan będący wynikiem działania fazy konstrukcji rozwiązania wstępnego oraz w oparciu o kolejno generowane sąsiedztwa tego planu $\mathcal{N}\left(\hat{\xi}^{(\tau)}\right)$. Działanie fazy lokalnej obejmuje dwa etapy: poszukiwanie czujnika, którego wyłączenie powoduje najmniejszą utratę wartości przyjętego kryterium optymalności oraz etap poszukiwania wśród czujników uśpionych, sąsiadujących z czujnikami znajdującymi się w aktualnym planie tego, którego włączenie spowoduje największe zmniejszenie wartości kryterium optymalności. Dokładny schemat działania równoległej fazy przeszukiwania lokalnego, zarówno dla węzła głównego, jak i węzłów liczących, przedstawiono w postaci Algorytmów 4.7 oraz 4.8.

Zrównoleglenie etapu pierwszego polega na dystrybucji do węzłów liczących zbiorów $S_1, \ldots, S_P$ zawierających położenia kandydatów do usunięcia z aktualnego planu. Rzecz jasna, musi zachodzić

$$\zeta^{(\kappa)} = \bigcup_{i=1}^{P} S_i \quad \text{oraz} \quad S_i \cap S_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.72}$$

Analogicznie do fazy poprzedniej wielkości zbiorów S_i zależą od wag w_i wyznaczanych na podstawie obciążenia lub statycznego przypisania.

Zrównoleglenie fazy drugiej przebiega podobnie jak w poprzednich przypadkach. Zdefiniujmy zbiór:

$$D = \mathcal{N}\left(\zeta^{(\kappa)}\right) \tag{4.73}$$

gdzie: $\mathcal{N}\left(\zeta^{(\kappa)}\right)$ — sąsiedztwo aktualnego (przybliżonego) planu eksperymentu $\zeta^{(\kappa)}$. Podział zbioru D na podzbiory $D_1, \ldots, D_P$ wygląda analogicznie jak w poprzednich przypadkach, tzn. musi zachodzić:

$$D = \bigcup_{i=1}^{P} D_i \quad \text{oraz} \quad D_i \cap D_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.74}$$

Liczność zbioru przydzielana i-temu procesorowi określa się jako

$$|D_i| = \left\lfloor w_i |D| \right\rfloor, \quad i = 1, \ldots, P, \tag{4.75}$$

gdzie: w_i — waga i-tego procesora wyznaczana przez algorytm dynamicznego równoważenia obciążenia lub arbitralnie, $\lfloor \cdot \rfloor$ — zaokrąglenie wartości liczbowej w dół.

Algorytm 4.7 Faza przeszukiwania lokalnego — procesor główny/nadzorca

Dane wejściowe:

$\hat{\xi}^{(\tau)}$ — wstępny plan eksperymentu opisujący aktualną konfigurację aktywnych czujników,
$w^{(\tau)} = \{w_1^{(\tau)}, \ldots, w_P^{(\tau)}\}$ — wektor wag dla poszczególnych procesorów.

Dane wyjściowe:
$\xi^{(\tau)}$ — skorygowany plan $\hat{\xi}^{(\tau)}$, zoptymalizowany poprzez wymiany jednopunktowe w $\hat{\xi}^{(\tau)}$ i jego sąsiedztwie.

Krok 0: (Inicjalizacja)
Inicjalizacja licznika iteracji $\kappa \leftarrow 0$ oraz utworzenie planu $\zeta^{(0)} = \hat{\xi}^{(\tau)}$.
Ustawienie $\Psi_{\min} \leftarrow \Psi\left(\zeta^{(0)}\right)$.

Krok 1: (Usunięcie najmniej wartościowego czujnika aktywnego)
W planie $\zeta^{(\kappa)}$ określa się położenie czujnika, którego uśpienie powoduje najmniejszą utratę wartości kryterium:

Faza 1: (Podział zbioru czujników w planie pomiędzy procesory)
Podział zbioru czujników znajdujących się w aktualnym planie eksperymentu $\zeta^{(\kappa)}$ pomiędzy procesory tak, aby

$$\zeta^{(\kappa)} = \bigcup_{i=1}^{P} S_i \quad \text{oraz} \quad S_i \cap S_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.76}$$

a liczność zbiorów $S_1, \ldots, S_P$ wynosiła

$$|S_i| = \left\lfloor w_i^{(\tau)} |S| \right\rfloor, \quad i = 1, \ldots, P, \tag{4.77}$$

Faza 2: (Wysłanie aktualnego planu eksperymentu do procesorów liczących)
Wysłanie aktualnego planu eksperymentu $\zeta^{(\kappa)}$ do wszystkich procesorów liczących.

Faza 3: (Wysłanie podzbiorów $S_1, \ldots, S_P$ do procesorów liczących)
Wysłanie do odpowiednich procesorów podzbiorów $S_1, \ldots, S_P$ zawierających informacje o czujnikach, których usunięcie z planu $\zeta^{(\kappa)}$ mają zbadać procesory liczące.

Faza 4: (Odebranie danych o kandydatach do usunięcia)
Odebranie wskazanych kandydatów $\bar{\mathrm{s}}_1, \ldots, \bar{\mathrm{s}}_P$ do usunięcia z aktualnego planu oraz odpowiednich wartości kryterium $\tilde{\Psi}_1, \ldots, \tilde{\Psi}_P$ otrzymywanych po przeprowadzeniu usunięcia.

Faza 5: (Usunięcie czujnika z planu)
Usuwany jest czujnik, którego wyłączenie powoduje najmniejszą utratę wartości wybranego kryterium optymalności

$$\begin{aligned} \mathrm{r} &\leftarrow \arg \min_{p=1,\ldots,P} \bar{\Psi}_p, \\ \bar{\zeta}^{(\kappa)} &\leftarrow \zeta^{(\kappa)} \setminus \{\bar{\mathrm{s}}_\mathrm{r}\}. \end{aligned} \tag{4.78}$$

Krok 2: (Dołączenie najbardziej wartościowego czujnika uśpionego)
W sąsiedztwie planu $\zeta^{(\kappa)}$ zdefiniowanym jako

$$\mathcal{N}\left(\zeta^{(\kappa)}\right) = \bigcup_{s\in\zeta^{(\kappa)}} \mathcal{N}(s), \tag{4.79}$$

poszukuje się położenia czujnika uśpionego, którego aktywacja powoduje największy ubytek wartości kryterium.

Faza 1: (Podział sąsiedztwa aktualnego planu pomiędzy procesory)
Podział sąsiedztwa czujników znajdujących się w planie eksperymentu $\zeta^{(\kappa)}$ pomiędzy procesory tak, aby

$$\mathcal{N}\left(\zeta^{(\kappa)}\right) = D = \bigcup_{i=1}^{P} D_i \quad \text{oraz} \quad D_i \cap D_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.80}$$

a liczność zbiorów $D_1, \ldots, D_P$ wynosiła

$$|D_i| = \left\lfloor w_i^{(\tau)} |D| \right\rfloor, \quad i = 1, \ldots, P, \tag{4.81}$$

Faza 2: (Wysłanie zmodyfikowanego planu eksperymentu)
Wysłanie zmodyfikowanego planu eksperymentu $\bar{\zeta}^{(\kappa)}$ do wszystkich procesorów liczących.

Faza 3: (Wysłanie podzbiorów $D_1, \ldots, D_P$ do procesorów liczących)
Wysłanie do odpowiednich procesorów podzbiorów $D_1, \ldots, D_P$ zawierających informacje o sąsiedztwie, z którego ma być wybrany czujnik zapewniający największy ubytek wartości kryterium optymalności.

Faza 4: (Odebranie danych o kandydatach do włączenia)
Odebranie wskazanych kandydatów $\tilde{\mathrm{d}}_1, \ldots, \tilde{\mathrm{d}}_P$ do włączenia do zredukowanego planu oraz odpowiednich wartości kryterium $\tilde{\Psi}_1, \ldots, \tilde{\Psi}_P$ otrzymanych po przeprowadzeniu włączenia.

Faza 5: (Dołączenie czujnika do zredukowanego planu)
Włączany jest czujnik, którego aktywacja powoduje największą utratę wartości kryterium optymalności w zmodyfikowanym planie $\bar{\zeta}^{(\kappa)}$:

$$\begin{aligned} \mathrm{e} &\leftarrow \arg \min_{p=1,\ldots,P} \tilde{\Psi}_p, \\ \hat{\mathrm{d}} &\leftarrow \tilde{\mathrm{d}}_{\mathrm{e}}. \end{aligned} \tag{4.82}$$

Jeżeli $\Psi\left(\bar{\zeta}^{(\kappa)} \cup \{\hat{\mathrm{d}}\}\right) \geqslant \Psi_{\min}$, algorytm kończy działanie po przyjęciu

$$\xi^{(\tau)} \leftarrow \zeta^{(\kappa)}. \tag{4.83}$$

W przeciwnym razie następują kolejno podstawienia

$$\begin{aligned} \zeta^{(\kappa+1)} &\leftarrow \bar{\zeta}^{(\kappa)} \cup \{\hat{\mathrm{d}}\}, \\ \Psi_{\min} &\leftarrow \Psi\Big(\bar{\zeta}^{(\kappa)} \cup \{\hat{\mathrm{d}}\}\Big), \\ \kappa &\leftarrow \kappa + 1. \end{aligned} \tag{4.84}$$

oraz powrót do Kroku 1.

■

Algorytm 4.8 Faza przeszukiwania lokalnego — procesor liczący

Dane wejściowe:

p — unikalny identyfikator procesora liczącego taki, że $1 \leqslant p \leqslant P$, nadawany przez środowisko programowania równoległego,

Krok 1: (Wyznaczenie najmniej wartościowego czujnika)

Faza 1: (Odebranie danych z procesora głównego)
Odebranie aktualnego planu eksperymentu $\zeta^{(\kappa)}$ z procesora głównego oraz zbioru czujników, których usunięcie z planu eksperymentu należy zbadać S_p.

Faza 2: (Wyznaczenie kandydata do usunięcia z planu)
Obliczenie

$$\begin{aligned} \bar{\mathrm{s}}_p &\leftarrow \arg\min_{\mathrm{s}\in\mathrm{S}_p} \Psi\Big(\zeta^{(\kappa)} \setminus \{\mathrm{s}\}\Big), \\ \bar{\Psi}_p &\leftarrow \Psi\Big(\zeta^{(\kappa)} \setminus \{\bar{\mathrm{s}}_p\}\Big). \end{aligned} \tag{4.85}$$

Faza 3: (Wysłanie informacji o proponowanym kandydacie)
Wysłanie wartości $\bar{\mathrm{s}}_p$ oraz $\bar{\Psi}_p$ do procesora głównego.

Krok 2: (Wyznaczenie i dodanie najbardziej wartościowego czujnika)

Faza 1: (Odebranie danych z procesora głównego)
Odebranie zmodyfikowanego planu eksperymentu $\bar{\zeta}^{(\kappa)}$ z procesora głównego oraz zbioru sąsiadów D_p planu eksperymentu $\zeta^{(\kappa)}$.

Faza 2: (Wyznaczenie kandydatów do włączenia do planu)
Obliczenie

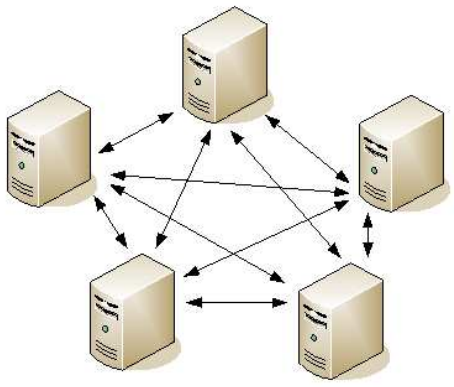
$$\tilde{\mathrm{d}}_p \leftarrow \arg\min_{\mathrm{d}\in\mathrm{D}_p} \Psi\Big(\bar{\zeta}^{(\kappa)} \cup \{\mathrm{d}\}\Big), \qquad (4.86)$$
$$\tilde{\Psi}_p \leftarrow \Psi\Big(\bar{\zeta}^{(\kappa)} \cup \{\tilde{\mathrm{d}}_p\}\Big).$$

Faza 3: (Wysłanie informacji o proponowanym kandydacie)
Wysłanie wartości $\tilde{\mathrm{d}}_p$ oraz $\tilde{\Psi}_p$ do procesora głównego.

■

4.5.2 Schemat „każdy z każdym"

W przypadku poszukiwania optymalnych konfiguracji aktywnych czujników skanujących zaproponowano wykorzystanie schematu komunikacji pomiędzy procesorami typu „każdy z każdym". W tym schemacie brak jest wyróżnionego procesora, który posiada informację o całym działającym algorytmie. Aby każdy z procesorów posiadał wiedzę o stanie algorytmu i fazie, w jakiej się znajduje, musi istnieć odpowiedni protokół komunikacji zrozumiały przez wszystkie procesory. Dzięki wymianie informacji każda z jednostek liczących może efektywnie wykonywać algorytm oraz odpowiednio przydzielać wielkość zadań w zależności od obciążenia. Dodatkowym wymogiem jest, aby poszczególne procesory wykonywały obliczenia na odpowiednich zestawach danych, przydzielonych tylko dla nich tak, aby obliczenia nie były wykonywane wielokrotnie przez różne procesory. Schemat wymiany informacji „każdy z każdym" przedstawiono na rys. 4.8.



Rysunek 4.8: Schemat komunikacji „każdy z każdym".

W Algorytmie 4.4 złożoność obliczeniowa jest zdeterminowana poprzez proces poszukiwania położeń czujników powodujących zmniejszenie wybranego kryterium optymalności. Operacje polegające na usunięciu czujnika i aktywowaniu czujnika

dotychczas uśpionego wymagają uaktualniania macierzy informacyjnej oraz wyznaczania wartości kryterium optymalności.

Metoda zrównoleglenia fazy przeszukiwania lokalnego jest bardzo podobna do metody zastosowanej do optymalnej aktywacji czujników stacjonarnych (zob. rozdz. 4.5.1). W rozważanym algorytmie zrównoleglenie zostało wykonane poprzez dekompozycję przestrzeni poszukiwań jaką jest sąsiedztwo aktualnie rozważanego planu eksperymentu, które podlega analizie w każdej iteracji algorytmu.

Pewną subtelną różnicą w stosunku do schematu „nadzorca-podwładni" jest określenie liczby procesorów P. W przypadku schematu komunikacji „każdy z każdym" wartość P oznacza wszystkie procesory na jakich uruchomiony jest algorytm, które są określane jako procesory liczące. W przypadku schematu „nadzorca-podwładni" wartość P oznaczała wszystkie jednostki liczące, czyli wszystkie procesory z wyłączeniem procesora nadzorcy.

Podobnie jak w przypadku równoległej wersji algorytmu aktywującego czujniki stacjonarne, po wykonaniu obliczeń związanych z optymalną aktywacją czujników skanujących, następuje analiza czasów wykonywania obliczeń przez poszczególne procesory. Czasy obliczeń, jakie zostały wymienione pomiędzy procesorami pozwalają na wyznaczenie nowych wartości wag w dla wszystkich procesorów, zgodnie z ich obciążeniem. Dokładniejsze informacje o metodzie równoważenia obciążenia przedstawione są w rozdz. 4.5.3.

Zrównoleglenie fazy przeszukiwania lokalnego

W tej fazie obliczenia zostają wykonywane na podstawie poszczególnych zbiorów zawierających sąsiedztwa $\hat{\mathcal{N}}_k$ czujników znajdujących się w aktualnym planie $\xi(k)$ dla k-tego etapu czasowego. Działanie fazy lokalnej obejmuje dwa etapy: szukanie czujnika, którego usunięcie powoduje najmniejszą utratę wartości szukanego kryterium optymalizacji oraz etap poszukiwania wśród sąsiadów aktualnego planu miejsca, w którym uzyska się największy ubytek przyjętego kryterium.

Zrównoleglenie etapu pierwszego polega na uruchomieniu przez procesory procedury dzielącej zbiór czujników należący do planu eksperymentu w k-tym etapie czasowym na podzbiory rozłączne zawierające czujniki, które dany procesor będzie sprawdzać

$$\xi(k) = \bigcup_{i=1}^{P} S_i \quad \text{oraz} \quad S_i \cap S_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.87}$$

Analogicznie do metody poprzedniej, wielkości zbiorów S_i zależą od wag w_i wyznaczanych na podstawie obciążenia lub statycznego przypisania.

Zrównoleglenie fazy drugiej, w której szukamy lokalizacji czujnika polepszającego własności tworzonego planu w k-tym etapie czasowym pod kątem minimalizacji kryterium optymalności, przebiega podobnie jak w poprzednich przypadkach. Zdefiniujmy zbiór:

$$D = \hat{\mathcal{N}}_k \tag{4.88}$$

gdzie: $\hat{\mathcal{N}}_k = \mathcal{N}\big(\xi(k)\big) \setminus \big\{\xi(k) \cup \mathcal{Z}_k\big\}$ — sąsiedztwo aktualnego planu eksperymentu $\xi(k)$. Podział zbioru D na podzbiory $D_1, \ldots, D_P$ wygląda analogicznie jak w poprzednich przypadkach, tzn. musi zachodzić:

$$D = \bigcup_{i=1}^{P} D_i \quad \text{oraz} \quad D_i \cap D_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.89}$$

Liczność zbioru przydzielana i-temu procesorowi określa się jako

$$|D_i| = \Big\lfloor w_i |D| \Big\rfloor, \quad i = 1, \ldots, P, \tag{4.90}$$

gdzie: w_i — waga i-tego procesora wyznaczana przez algorytm dynamicznego równoważenia obciążenia lub arbitralnie.

Zrównoleglenie Algorytmu 4.4 wymaga zmian w Kroku 1. Zmodyfikowany Krok 1 przedstawiono jako Algorytm 4.9. Dla uproszczenia pominięto indeksy oznaczające numer iteracji. Dodatkowym parametrem wejściowym jest unikatowy identyfikator procesora p. Jest on przydzielany przez system równoległy każdemu procesorowi i umożliwia jego jednoznaczną identyfikację.

Algorytm 4.9 Optymalna aktywacja czujników skanujących — wersja równoległa kroku 1

Dane wejściowe:

p — unikatowy identyfikator procesora wykonującego algorytm.

Krok 1:

Szukanie czujników, których włączenie do planu pozwoli na poprawę wartości kryterium optymalności.

Kolejno dla każdego k-tego etapu aktualnego planu eksperymentu ξ wykonuje się

Faza 1:

Z utworzonego sąsiedztwa planu $\xi(k)$ zdefiniowanego jako

$$\mathcal{N}_k = \bigcup_{s \in \xi(k)} \mathcal{N}(s), \tag{4.91}$$

usuwane są czujniki należące do tego planu:

$$\bar{\mathcal{N}}_k \leftarrow \mathcal{N}_k \setminus \xi(k) \tag{4.92}$$

Faza 2:

Z sąsiedztwa $\bar{\mathcal{N}}_k$ usuwamy czujniki znajdujące się w zbiorze sąsiadów zabronionych k-tego etapu $\mathcal{Z}_k$

$$\hat{\mathcal{N}}_k \leftarrow \bar{\mathcal{N}}_k \setminus \mathcal{Z}_k. \tag{4.93}$$

Faza 3: (Podział zbioru czujników w planie pomiędzy procesory)
Podział zbioru czujników znajdujących się w aktualnym k-tym etapie planu eksperymentu ξ pomiędzy procesory tak, aby zachodziło

$$\xi(k) = \bigcup_{i=1}^{P} S_i \quad \text{oraz} \quad S_i \cap S_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.94}$$

a liczność zbiorów $S_1, \ldots, S_P$ wynosiła

$$|S_i| = \Big\lfloor w_i |S| \Big\rfloor, \quad i = 1, \ldots, P, \tag{4.95}$$

Faza 4:
Każdy procesor wyznacza czujnik, którego deaktywacja powoduje najmniejszą utratę wartości wybranego kryterium optymalności. Każdy procesor analizuje tylko zbiór skojarzony ze swoim identyfikatorem p

$$\bar{\mathrm{s}}_p \leftarrow \arg\min_{\mathrm{s} \in S_p} \Psi\Big(\xi \setminus \{\mathrm{s}\}_k\Big). \tag{4.96}$$

Faza 5:
Pomiędzy procesorami następuje wymiana położeń czujników $\bar{s}_1, \ldots, \bar{s}_P$ kandydujących do usunięcia.

Faza 6:
Na podstawie danych otrzymanych od innych procesorów, każdy z nich wykonuje operację wyłączenia najmniej wartościowego czujnika w planie $\xi(k)$:

$$\begin{aligned} &\mathrm{r} \leftarrow \arg\min_{i=1,\ldots,P} \Psi\Big(\xi \setminus \{\bar{\mathrm{s}}_i\}_k\Big), \\ &\bar{\xi} \leftarrow \xi \setminus \{\bar{\mathrm{s}}_r\}_k. \end{aligned} \tag{4.97}$$

Faza 7: (Podział sąsiedztwa aktualnego planu pomiędzy procesory)
Podział sąsiedztwa czujników znajdujących się w k-tym etapie planu eksperymentu ξ pomiędzy procesory tak, aby zachodziło

$$\hat{\mathcal{N}}_k = D = \bigcup_{i=1}^{P} D_i \quad \text{oraz} \quad D_i \cap D_j = \emptyset \quad \text{dla} \quad i \neq j. \tag{4.98}$$

a liczność zbiorów $D_1, \ldots, D_P$ wynosiła

$$|D_i| = \Big\lfloor w_i |D| \Big\rfloor, \quad i = 1, \ldots, P, \tag{4.99}$$

Faza 8:
Wśród czujników znajdujących się w wyznaczonym podzbiorze sąsiedztwa D_p planu $\xi(k)$, każdy procesor szuka takiego, którego aktywacja powoduje

uzyskanie najmniejszej wartości kryterium optymalizacji po rozszerzeniu planu $\bar{\xi}$. Każdy procesor analizuje tylko zbiór skojarzony ze swoim identyfikatorem p:

$$\tilde{\mathrm{d}}_p \leftarrow \arg\min_{\mathrm{d}\in D_p} \Psi\Big(\bar{\xi} \cup \{\mathrm{d}\}_k\Big), \tag{4.100}$$

Faza 9:

Pomiędzy procesorami następuje wymiana informacji o czujnikach $\tilde{d}_1, \ldots, \tilde{d}_P$ kandydujących do aktywacji.

Faza 10:

Na podstawie danych otrzymanych od innych procesorów, każdy z nich dokonuje aktywacji czujnika powodującego największy ubytek wartości kryterium optymalności:

$$\begin{aligned} \mathrm{e} &\leftarrow \arg\min_{i=1,\ldots,P} \Psi\Big(\bar{\xi} \cup \{\tilde{\mathrm{d}}_i\}_k\Big), \\ \hat{\xi} &\leftarrow \bar{\xi} \cup \{\tilde{\mathrm{d}}_\mathrm{e}\}_k. \end{aligned} \tag{4.101}$$

Faza 11:

Do zbioru sąsiadów zabronionych k-tego etapu dodaje się nowo aktywowany czujnik

$$\mathcal{Z}_k = \mathcal{Z}_k \cup \{\hat{d}\}. \tag{4.102}$$

■

4.5.3 Równoważenie obciążenia

W przedstawionej pracy do celów równoważenia obciążenia procesorów zastosowano metodę adaptacyjnej ważonej faktoryzacji przedstawioną w rozdz. 3.8. Jest ona wykorzystywana do przydziału odpowiedniej liczby danych do obliczeń przez procesory systemu równoległego w zależności od ich obciążenia. Miarą wykorzystywaną do pomiaru obciążenia jest czas obliczeń wymagany przez każdy z procesorów do wykonania obliczeń związanych z wyznaczeniem wartości kryterium optymalności.

Wyznaczenie aktualnego obciążenia jednostek obliczeniowych ma miejsce pod koniec każdej iteracji zaproponowanych algorytmów. W przypadku algorytmu optymalnej aktywacji czujników stacjonarnych ma to miejsce w Kroku 3 (zob. Algorytm 4.1), a w algorytmie optymalnej aktywacji czujników skanujących ma to również miejsce w Kroku 3 (zob. Algorytm 4.4). Bazując na informacjach otrzymywanych od innych procesorów, procesor główny (w przypadku algorytmu optymalnej aktywacji czujników stacjonarnych) lub wszystkie procesory (w przypadku algorytmu optymalnej aktywacji czujników skanujących) uaktualizują wartości współczynników w_i.

Algorytm optymalnej aktywacji czujników stacjonarnych

Algorytm optymalnej aktywacji czujników stacjonarnych wymaga określenia wielkości trzech zbiorów danych, jakie powinny zostać przesłane do procesorów liczących:

- zbiór uśpionych czujników do aktywacji, które należy zbadać pod kątem minimalizacji wartości kryterium optymalności,
- zbiór aktywnych czujników do wyłączenia, które należy zbadać pod kątem najmniejszego pogorszenia wartości kryterium optymalności,
- zbiór uśpionych czujników z sąsiedztwa, które należy zbadać pod kątem minimalizacji wartości kryterium optymalności.

Algorytm optymalnej aktywacji czujników skanujących

Algorytm optymalnej aktywacji czujników stacjonarnych wymaga określenia wielkości dwóch zbiorów danych, jakie powinny zostać przesłane do procesorów liczących:

- zbiór aktywnych czujników do wyłączenia, które należy zbadać pod kątem najmniejszego utraty wartości kryterium optymalności,
- zbiór uśpionych czujników z sąsiedztwa, które należy zbadać pod kątem minimalizacji wartości kryterium optymalności.

Zastosowana metoda równoważenia obciążenia

Dynamiczne wielkości zbiorów na których wykonywane są obliczenia, ustalane są na podstawie tej samej metody. Do swojego działania wymaga ona podania czasów oraz wielkości danych jakie zostały dotychczas przetworzone przez procesory liczące. Na podstawie tych informacji wyznacza się średni czas wykonania obliczeń na pojedynczej operacji atomowej dla każdego z procesorów liczących (zob. (3.27)). W przypadku rozważanych zagadnień operacją atomową jest wyznaczenie wartości kryterium optymalności. Wykonanie kolejnych obliczeń (3.28)–(3.29) pozwala na wyznaczenie wag w_i dla wszystkich procesorów liczących. Wagi o wartości powyżej jedności oznaczają, że procesor był wcześniej niedoszacowany pod kątem ilości obliczeń i należy zwiększyć jego obciążenie. Analogicznie, wartość wagi poniżej jedności oznacza przeszacowanie ilości obliczeń na danym procesorze i konieczność pomniejszenia jego obciążenia w kolejnej iteracji.

Metoda adaptacyjnej ważonej faktoryzacji (patrz rozdz. 3.8) zakłada, że wagi o wartości 1 dla każdego z procesorów określają równomierny podział danych pomiędzy procesory liczące. W celu łatwiejszej implementacji, wyznaczone wagi poddane zostają kolejnemu procesowi normalizacji, tak aby wagi o wartości $1/P$ określały podział równomierny, gdzie: P — liczba procesorów liczących. Zachodzi więc

$$\bar{w_i} = \frac{w_i}{\sum_{j=1}^{P} w_j}, \quad i = 1, \ldots, P. \tag{4.103}$$

Odchyłki w górę od wartości $\bar{w} = 1/P$ określają niedoszacowanie procesora w poprzednich iteracjach, a odchyłki w dół, czyli $\bar{w} < 1/P$, określają przeszacowanie procesora w poprzednich iteracjach.

Na podstawie tak wyznaczonych wartości współczynników $\bar{w}_i$ następuje podział zbiorów danych, jakie mają być przesłane do procesorów liczących. W przypadku, gdy liczba elementów jakie mają być przesłane do poszczególnych procesorów na podstawie uaktualnianych wag nie jest liczba całkowitą, reszty wynikające z podziału zbiorów na podzbiory są sumowane, a następnie dodawane do obciążenia procesora, który przeprowadza obliczenia najszybciej. Takie rozwiązanie problemu podziału zbiorów danych na podzbiory może powodować nieznaczne fluktuacje w procesie równoważenia obciążenia procesorów (zob. rozdz. 4.6).

4.6 Uzyskane wyniki

Obliczenia przeprowadzono w oparciu o klaster zbudowany w Centrum Komputerowym Uniwersytetu Zielonogórskiego w ramach projektu CLUSTERIX [248] (zob. rozdz. 3.9). Homogeniczny klaster składa się z czterech węzłów zawierających dwa 64-bitowe procesory *Intel Itanium* 1.4GHz i działa pod kontrolą środowiska GNU/ Linux Debian for ia64. Połączenia pomiędzy węzłami realizuje się poprzez sieć Gigabit Ethernet. Aplikacje użyte do zrównoleglenia przedstawionych zagadnień napisano w języku Fortran 95 i skompilowano z zastosowaniem kompilatora *ifort* (Intel®Fortran Compiler v.8.1 for Linux 64-bit platforms) oraz implementacji interfejsu MPI [163] w postaci biblioteki *mpich* – 1.2.6. Aplikacje używają biblioteki procedur numerycznych Intel Math Kernel Library do wykonywania operacji mnożenia macierzy oraz wektorów.

Obliczenia w środowisku heterogenicznym przeprowadzano na tymczasowym klastrze składającym się z komputerów klasy PC (zob. rozdz. 3.9) oraz na homogenicznym klastrze CLUSTERIX, na którym uruchomiono dodatkowe procesy wprowadzające istotne dodatkowe obciążenie procesorów. W efekcie trzy procesory pracowały ze znacznie zmniejszoną mocą obliczeniową przydzielaną dla zadań związanych z badaniami.

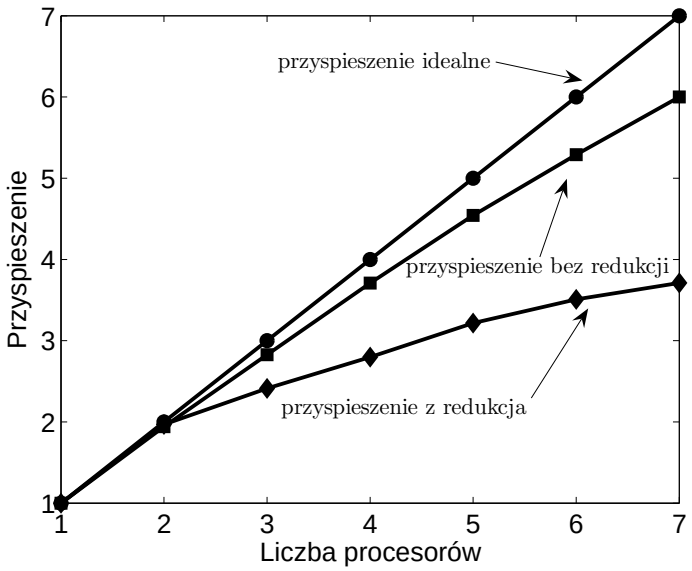
4.6.1 Zastosowanie redukcji czasochłonnych operacji

W celu przedstawienia zysków ze stosowania metody redukcji operacji w oparciu o formułę Shermana-Morrisona-Woodbury'ego, przeprowadzono symulacje aktywacji $n = 40$ czujników stacjonarnych dla Przykładu 4.1. Opracowano dwie wersje algorytmu optymalnej aktywacji czujników stacjonarnych: z metodą redukcji oraz bez niej. Tabela 4.1 przedstawia czasy uzyskane dla przeprowadzonych symulacji. Dla porównania, na rys. 4.9 znajdują się wykresy przyspieszenia obliczeń uzyskane po uruchamianiu obydwu wersji algorytmów na ośmioprocesorowym klastrze homogenicznym. Można zauważyć, że algorytm bez redukcji czasochłonnych operacji

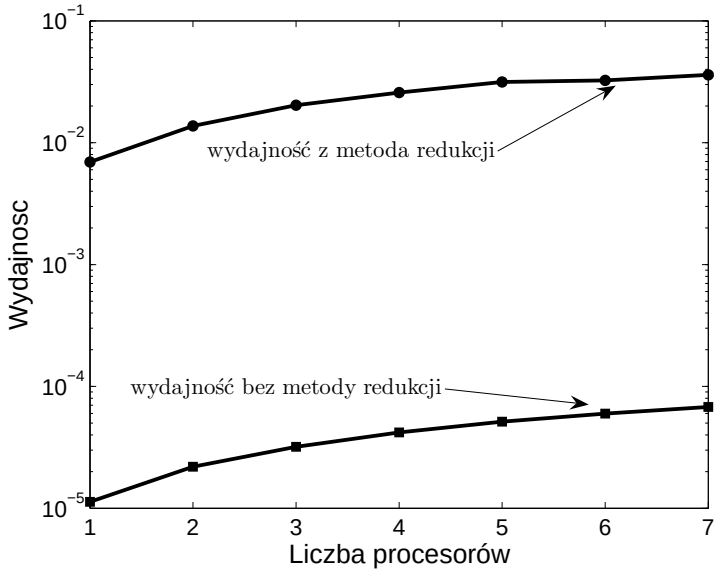
osiąga znacznie lepsze parametry przyspieszenia, jednak jest to okupione olbrzymim zwiększeniem się czasu obliczeń. Analizując tab. 4.1 widać, że obliczenia mogą trwać nawet dobę (w przypadku jednego procesora obliczeniowego). Wykorzystanie miary wydajności algorytmów równoległych pozwala na porównanie obydwu algorytmów optymalnej aktywacji czujników stacjonarnych rozwiązujących identyczny problem obliczeniowy. Na rys. 4.10 można zauważyć, że redukcja czasochłonnych operacji pozwala na uzyskanie około 1000-krotnie większej wydajności obliczeń przy rozważaniu problemu aktywacji 40 czujników stacjonarnych dla Przykładu 4.1.

	Liczba procesorów liczących P						
algorytm	1	2	3	4	5	6	7
bez redukcji	24:36:00	12:40:00	8:42:00	6:38:00	5:25:00	4:39:00	4:06:00
z redukcją	00:03:13	00:01:38	0:01:20	0:01:09	0:01:01	0:00:55	0:00:52

Tabela 4.1: Czasy obliczeń [h:min:s] uzyskane dla Algorytmu 4.5 oraz Przykładu 4.1 z redukcją liczby operacji w oparciu o formułę Shermana-Morrisona-Woodbury'ego oraz bez takiej redukcji.



Rysunek 4.9: Przyspieszenie uzyskane dla Algorytmu 4.5 oraz Przykładu 4.1 z redukcją liczby operacji w oparciu o formułę Shermana-Morrisona-Woodbury'ego oraz bez takiej redukcji w przypadku aktywacji $n = 40$ czujników stacjonarnych.



Rysunek 4.10: Porównanie wydajności algorytmu z redukcją liczby operacji w oparciu o formułę Shermana-Morrisona-Woodbury'ego oraz bez takiej redukcji liczby operacji.

4.6.2 Optymalna aktywacja czujników stacjonarnych

W badaniach dotyczących optymalnego planowania rozmieszczenia czujników stacjonarnych rozważano model przedstawiony w Przykładzie 4.1. W tab. 4.2 przedstawione są czasy obliczeń, jakie uzyskano podczas uruchamiania algorytmu optymalnej aktywacji czujników stacjonarnych na ośmioprocesorowym klastrze homogenicznym dla różnej liczby aktywowanych czujników przy zastosowaniu metody redukcji czasochłonnych operacji (zob. rozdz. 4.4).

	Liczba procesorów obliczeniowych P						
n	1	2	3	4	5	6	7
40	0:03:13	0:01:38	0:01:20	0:01:09	0:01:01	0:00:55	0:00:52
80	0:13:52	0:07:02	0:04:47	0:04:11	0:03:32	0:03:05	0:02:48
140	0:53:24	0:27:02	0:18:19	0:14:50	0:12:17	0:10:31	0:09:17

Tabela 4.2: Czasy obliczeń [h:min:s] uzyskane dla Algorytmu 4.5 oraz Przykładu 4.1 (klaster homogeniczny).

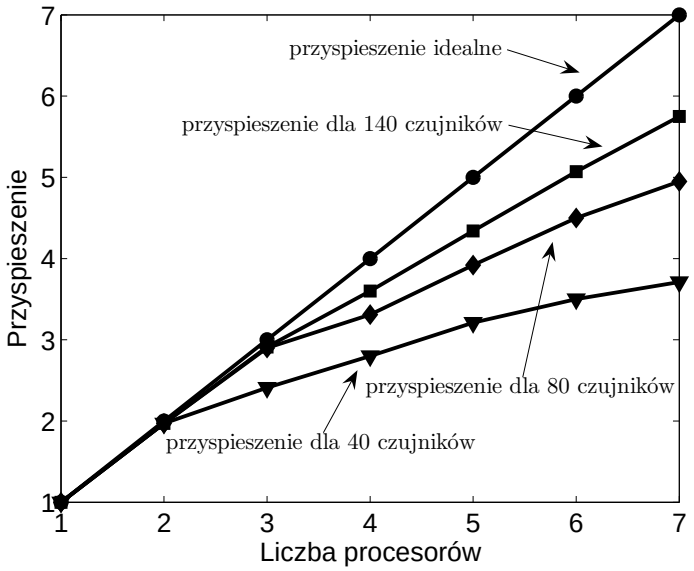
Uzyskane przyspieszenie oraz efektywność przedstawiono w tab. 4.3 oraz na rys. 4.11. Można zauważyć uzyskiwanie mniejszego przyspieszenia oraz efektywności

dla mniejszej liczby aktywowanych czujników. Ten efekt jest spowodowany mniejszą liczbą operacji jakie są potrzebne do znalezienia optymalnej konfiguracji mniejszej liczby czujników. Mniejsza liczba operacji powoduje, że większą rolę odgrywają np. opóźnienia związane z transmisją danych pomiędzy węzłami bądź obciążenie procesora głównego występujące w przypadku schematu „nadzorca-podwładni".

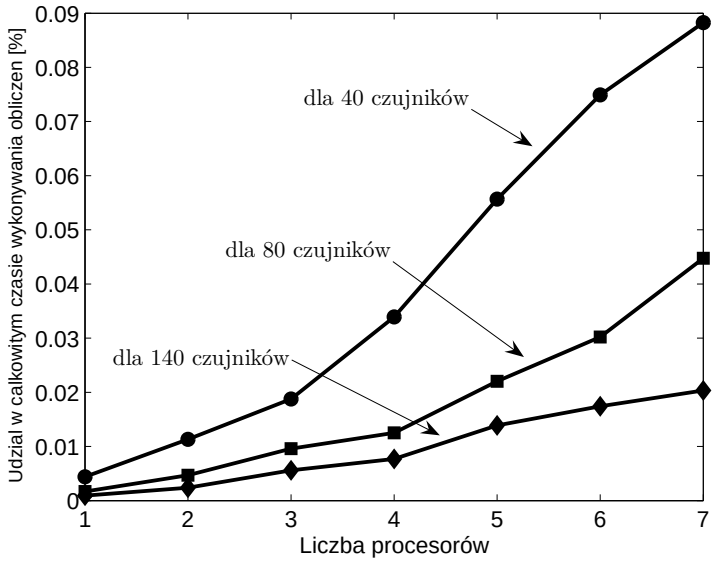
		Liczba procesorów obliczeniowych P						
n		1	2	3	4	5	6	7
40	$S(p)$	1.00	1.97	2.41	2.80	3.21	3.50	3.71
	$E(p)$	1.00	0.98	0.80	0.69	0.64	0.58	0.52
80	$S(p)$	1.00	1.97	2.90	3.31	3.92	4.50	4.95
	$E(p)$	1.00	0.98	0.96	0.82	0.78	0.75	0.70
140	$S(p)$	1.00	1.97	2.91	3.60	4.34	5.07	5.75
	$E(p)$	1.00	0.98	0.97	0.90	0.86	0.84	0.82

Tabela 4.3: Przyspieszenie i efektywność uzyskane dla Algorytmu 4.5 oraz Przykładu 4.1 (klaster homogeniczny).

Procentowy udział komunikacji w ogólnym czasie obliczeń przedstawiono na rys. 4.12. Konieczność częstej wymiany informacji pomiędzy procesorem–nadzorcą, a procesorami podwładnymi powoduje, że algorytm optymalnej aktywacji czujników stacjonarnych wymaga przesyłania znacznych ilości danych w trakcie swojej pracy. Można jednak zauważyć, że udział komunikacji w ogólnym czasie wykonywania obliczeń nie jest wysoki i wynosi poniżej jednego promila. Oczywiście, wraz ze wzrostem liczby procesorów, czas spędzony na komunikację pomiędzy nimi rośnie ze względu na konieczność przesyłania danych do większej liczby jednostek obliczeniowych. Jednakże, wraz ze wzrostem liczby aktywowanych czujników, udział komunikacji w ogólnym czasie pracy algorytmu maleje, co jest następstwem wydłużenia się czasu obliczeń wykonywanych przez procesory liczące. Należy jednak mieć na uwadze, że pomimo stosunkowo niewielkiego udziału komunikacji w ogólnym czasie obliczeń, algorytm ten jest wrażliwy na zwiększanie liczby procesorów. Fakt ten jest spowodowany koniecznością wymiany danych pomiędzy procesorem–nadzorcą, a procesorami liczącymi. W przypadku zbyt dużej liczby procesorów liczących może wystąpić sytuacja, gdy procesor-nadzorca zostanie zbyt obciążony obsługą żądań, co spowoduje w konsekwencji opóźnienia w transmisji danych do procesorów liczących. Pewnym rozwiązaniem tego problemu jest budowa heterogenicznego środowiska obliczeniowego, w którym najszybszy procesor będzie pełnić rolę nadzorcy.



Rysunek 4.11: Przyspieszenie uzyskane dla Algorytmu 4.5 oraz Przykładu 4.1 (klaster homogeniczny).



Rysunek 4.12: Procentowy udział czasu komunikacji w ogólnym czasie wykonywania obliczeń dla Algorytmu 4.5 oraz Przykładu 4.1 (klaster homogeniczny).

W badaniach uwzględniono również problem równoważenia obciążenia w przypadku obliczeń w środowisku klastra heterogenicznego. Klastry tego typu posiadają procesory o różnej mocy obliczeniowej i w przypadku uruchomienia synchronicznego

algorytmu, w którym poszczególne procesory oczekują na zakończenie obliczeń przez inne procesory, najwolniejszy procesor determinuje czas działania całego algorytmu.

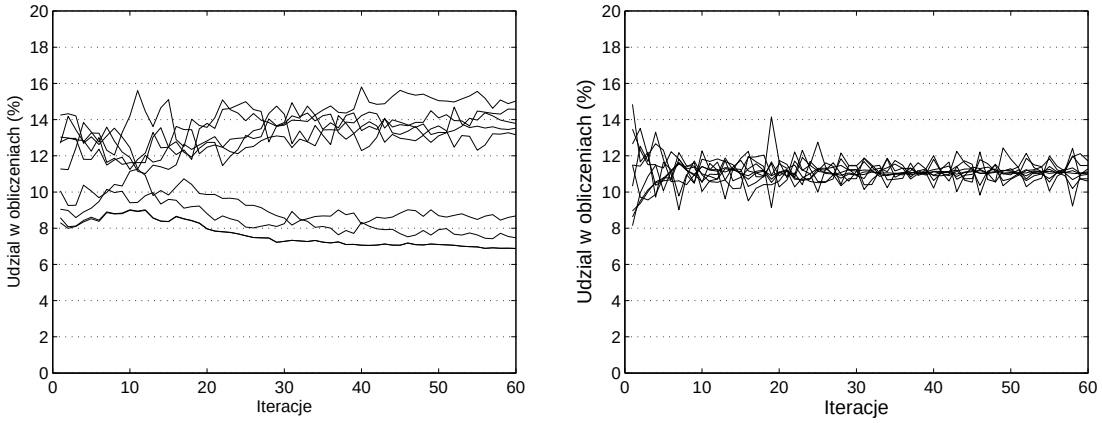
W przypadku badań dotyczących optymalnej aktywacji czujników stacjonarnych, heterogeniczny klaster składał się z 10 stacji roboczych posiadających procesory klasy Intel®Celeron oraz Intel®Pentium IV z różnymi częstotliwościami taktowania zegarów (zob. rozdz. 3.9). Na rys. 4.13 przedstawiono wyniki działania algorytmu rozwiązującego problem optymalnej aktywacji $n = 140$ czujników stacjonarnych dla Przykładu 4.1. Rysunek 4.13(a) pokazuje działanie algorytmu bez równoważenia obciążenia. Wykres utworzono na podstawie analizy czasów wykonywania obliczeń w poszczególnych iteracjach przez poszczególne procesory. W każdej iteracji sumowano czas wykonywania obliczeń, a następnie obliczano procentowy udział czasu obliczeń danego procesora w ogólnym zsumowanym czasie obliczeń. Można zauważyć, że cztery procesory wykonują swoje obliczenia znacznie szybciej i mają mniejszy udział w ogólnym czasie obliczeń. Uruchomienie metody równoważenia obciążenia (patrz rys. 4.13(b)) powoduje, że najszybsze procesory otrzymują większe porcje danych do obliczeń i dzięki temu czasy pracy procesorów zostają ustalone na zbliżonym poziomie. Taki przydział obliczeń procesorom obliczeniowym pozwala na uzyskanie większej wydajności algorytmu.

W przedstawionym przykładzie czas pracy algorytmu bez metody równoważenia obciążenia wynosił 7544 s. Włączenie równoważenia obciążenia powodowało zmniejszenie czasu obliczeń do 5741 s, czyli o 23 procent. W celu zbadania wpływu metody równoważenia obciążenia wykorzystano miarę wariancji procentowego udziału procesorów w obliczeniach (zob. rozdz. 3.8). Dla algorytmu bez równoważenia obciążenia uzyskano średnią wariancję udziału procesorów w obliczeniach równą $\Gamma(\pi) = 8.48$, a dla algorytmu z działającą metodą równoważenia obciążenia współczynnik ten wynosił $\Gamma(\pi) = 0.42$. Średnią wartość wariancji udziału procesorów w obliczeniach wyznaczono na podstawie cząstkowych czasów obliczeń w każdej iteracji przez poszczególne procesory. Wykorzystywano je do obliczenia procentowego udziału poszczególnych procesorów w całkowitym czasie obliczeń wykonywanym przez wszystkie procesory w danej iteracji. Następnie, dla każdej iteracji wyznaczana była wariancja procentowego udziału w czasie obliczeń, którą ostatecznie uśredniano względem wszystkich iteracji algorytmu.

4.6.3 Optymalna aktywacja czujników skanujących

W badaniach dotyczących optymalnej aktywacji czujników skanujących rozważano Przykład 4.2. W tab. 4.4 przedstawiono czasy obliczeń algorytmu uruchomionego na homogenicznym klastrze ośmioprocesorowym. W każdym badaniu algorytm uruchamiano dla identycznej (chociaż losowej) konfiguracji początkowej aktywnych czujników. W badaniach wykorzystano metodę redukcji czasochłonnych operacji opisaną w rozdz. 4.4.

Można zauważyć dość znaczące zwiększenie czasów obliczeń w stosunku do czasów optymalnej aktywacji czujników stacjonarnych. Jest to spowodowane trzema



(a) (b)

Rysunek 4.13: Wykorzystanie czasu przez poszczególne procesory dla Algorytmu 4.5 oraz Przykładu 4.1 (klaster heterogeniczny): (a) bez równoważenia obciążenia, (b) z równoważeniem obciążenia.

	Liczba procesorów obliczeniowych P							
n	1	2	3	4	5	6	7	8
40	0:02:13	0:01:15	0:00:56	0:00:41	0:00:36	0:00:32	0:00:29	0:00:27
80	0:18:55	0:09:32	0:06:44	0:05:01	0:04:00	0:03:35	0:03:12	0:02:45
140	1:59:44	1:00:23	0:40:42	0:29:56	0:25:17	0:21:08	0:18:05	0:16:37

Tabela 4.4: Czasy obliczeń [h:min:s] uzyskane dla Algorytmu 4.9 oraz Przykładu 4.2 (klaster homogeniczny).

zagadnieniami:

- *Poszukiwanie optymalnej aktywacji wszystkich wymaganych czujników.* Algorytm optymalnej aktywacji czujników skanujących poszukuje optymalnych konfiguracji wszystkich wymaganych czujników, jednocześnie dla wielu etapów. Wymaga to większego nakładu obliczeń w stosunku do algorytmu optymalnej aktywacji czujników stacjonarnych, gdzie liczba czujników jest zwiększana w czasie pracy fazy konstrukcji rozwiązania wstępnego. Algorytm optymalnej aktywacji czujników stacjonarnych w każdej iteracji przechodzi przez fazę konstrukcji rozwiązania wstępnego, w której iteracyjnie, do rozwiązania bazowego dołączane są kolejno pojedyncze czujniki, w celu wyznaczenia wstępnego przybliżenia planu optymalnego. Takie działanie powoduje, że algorytm w fazie konstrukcji rozwiązania wstępnego nie wykonuje, w większej

części, operacji związanych z optymalną aktywacją wszystkich wymaganych czujników. Znacznie większa przestrzeń poszukiwań, w przypadku problemu optymalnej aktywacji czujników skanujących wymaga analizy rozwiązań, w których aktywne są wszystkie wymagane czujniki, w celu zminimalizowania błędu wynikającego z heurystyki algorytmu. Oczywiście, takie zachowanie algorytmu rzutuje na całkowity czas wykonywania obliczeń.

- *Konieczność przeprowadzania obliczeń dla wielu konfiguracji czujników.*
  Optymalna aktywacja czujników skanujących wymaga określania optymalnych konfiguracji czujników dla każdego etapu skanowania. Wielokrotne określanie różnych konfiguracji powoduje zwiększenie nakładu obliczeń, a przez to wydłużenie czasu pracy algorytmu.

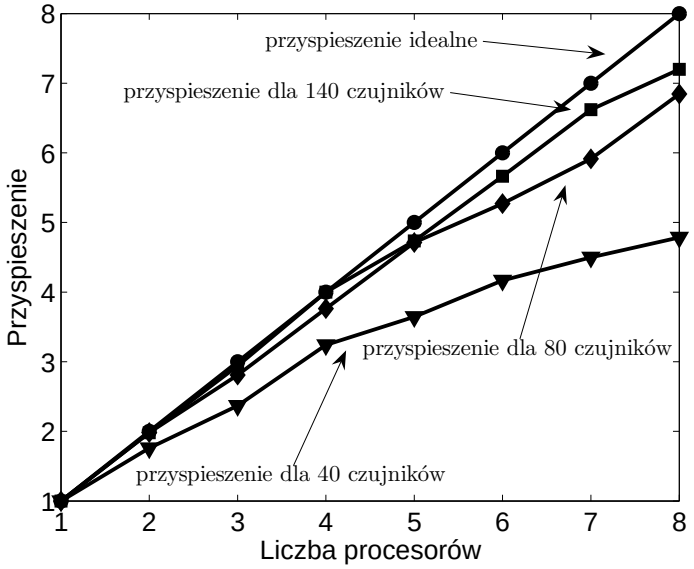
- *Różna liczba iteracji algorytmu.*
  Liczba iteracji algorytmu optymalnej aktywacji czujników skanujących jest zależna przede wszystkim od początkowej (losowej) konfiguracji aktywnych czujników. Zastosowanie kryterium stopu opartego na porównywaniu kolejnych wartości kryterium optymalności powoduje, że liczba iteracji umożliwiająca znalezienie przybliżenia optimum globalnego wybranego kryterium optymalności nie jest możliwa do wyznaczenia przed uruchomieniem algorytmu. Przykładowo, czasy wykonywania obliczeń przedstawione w tab. 4.4 zostały otrzymane po wykonaniu 32, 48 oraz 74 iteracji odpowiednio dla aktywacji 40, 80 oraz 140 czujników skanujących.

Przyspieszenie oraz efektywność wykonywania obliczeń równoległych, obliczona na podstawie danych z tab. 4.4, przedstawiono w tab. 4.5 oraz na rys. 4.14. Podobnie jak w przypadku algorytmu optymalnej aktywacji czujników stacjonarnych, można zauważyć większe przyspieszenia algorytmu dla zadań aktywacji większej liczby czujników. Jest to spowodowane większym wpływem części sekwencyjnej algorytmu, opóźnień w transmisji danych oraz czasów bezczynności procesorów w przypadku obliczeń wykorzystujących w mniejszym stopniu moc procesorów, co ma miejsce przy aktywacji mniejszej liczby czujników skanujących.

Procentowy udział komunikacji w ogólnym czasie obliczeń przedstawiony jest na rys. 4.15. Można zauważyć, że udział komunikacji w ogólnym czasie wykonywania obliczeń jest rzędu dziesiątych części promila. Oczywiście, wraz ze wzrostem liczby procesorów czas rośnie ze względu na konieczność przesyłania danych pomiędzy większą liczbą procesorów. Wraz ze wzrostem liczby aktywowanych czujników, udział komunikacji w ogólnym czasie pracy algorytmu maleje ze względu na większą liczbę koniecznych do wykonania obliczeń, a przez to dłuższy czas pracy procesorów liczących. Należy jednak mieć na uwadze, że pomimo mniejszego udziału komunikacji w ogólnym czasie obliczeń dla algorytmu optymalnej aktywacji czujników skanujących, algorytm ten jest bardziej wrażliwy na zwiększanie liczby procesorów. W odróżnieniu od schematu komunikacji „nadzorca–podwładni", gdzie problemem może się stać przeciążenie procesora-nadzorcy, w przypadku schema-

n		Liczba procesorów obliczeniowych P							
		1	2	3	4	5	6	7	8
40	$S(p)$	1.00	1.76	2.36	3.24	3.64	4.16	4.51	4.78
	$E(p)$	1.00	0.88	0.78	0.81	0.72	0.69	0.64	0.59
80	$S(p)$	1.00	1.98	2.80	3.76	4.71	5.27	5.91	6.84
	$E(p)$	1.00	0.99	0.93	0.94	0.94	0.87	0.84	0.85
140	$S(p)$	1.00	1.98	2.94	3.98	4.73	5.66	6.61	7.19
	$E(p)$	1.00	0.99	0.98	0.99	0.94	0.94	0.94	0.89

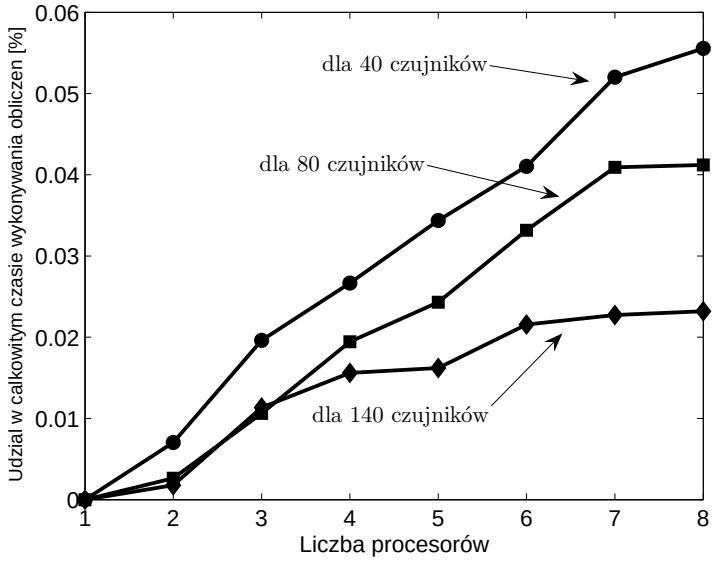
Tabela 4.5: Przyspieszenie i efektywność uzyskane dla Algorytmu 4.9 oraz Przykładu 4.2 (klaster homogeniczny).



Rysunek 4.14: Przyspieszenie uzyskane dla Algorytmu 4.9 oraz Przykładu 4.2 (klaster homogeniczny).

tu „każdy z każdym" problemem jest liczba połączeń koniecznych do przesłania informacji pomiędzy procesorami. Fakt ten jest spowodowany koniecznością wymiany danych przez wszystkie procesory pomiędzy sobą, w odróżnieniu od schematu komunikacji „nadzorca–podwładni", gdzie komunikacja następuje tylko pomiędzy procesorem–nadzorcą, a procesorami liczącymi. Schemat wymiany danych „każdy z

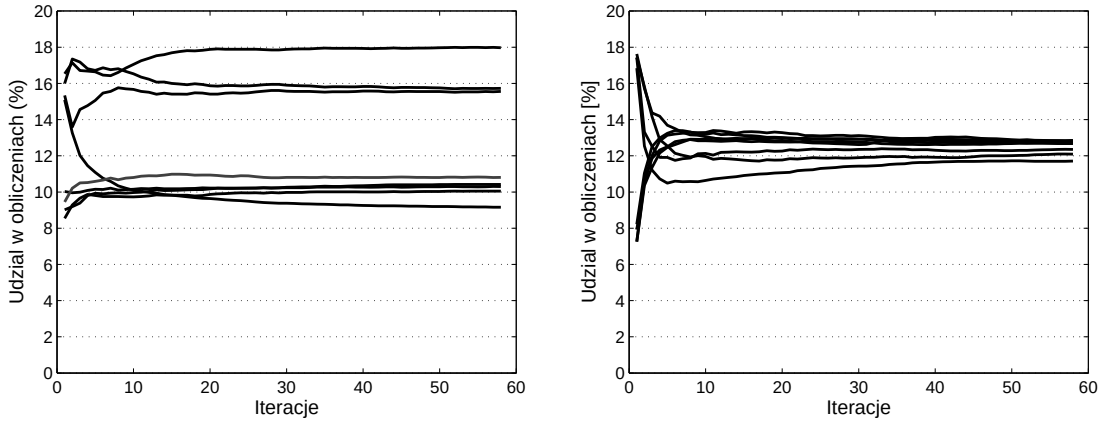
każdym" wymaga wykonania $p \cdot (p-1)$ operacji przesłania danych, co w porównaniu do $p-1$ koniecznych połączeń dla schematu „nadzorca–podwładni", może mieć duże znaczenie dla dużej liczby procesorów, nawet po uwzględnieniu przesyłania ilościowo znacznie mniejszych wielkości danych, co ma miejsce w algorytmie optymalnej aktywacji czujników skanujących.



Rysunek 4.15: Procentowy udział czasu komunikacji w ogólnym czasie wykonywania obliczeń dla Algorytmu 4.9 oraz Przykładu 4.2 (klaster homogeniczny).

Podobnie jak w poprzednich badaniach, rozważono również problem równoważenia obciążenia w przypadku obliczeń w klastrze heterogenicznym. W celu wprowadzenia niejednorodności w mocach procesorów, na niektórych z nich uruchomiono dodatkowe procesy wprowadzające istotne dodatkowe obciążenie procesorów. W efekcie trzy procesory pracowały ze znacznie zmniejszoną mocą obliczeniową przydzielaną zadaniom związanym z badaniami. Na rys. 4.16 przedstawiono wyniki działania algorytmu rozwiązującego problem optymalnej aktywacji czujników skanujących dla Przykładu 4.2. Algorytm wyznaczał optymalne konfiguracje $n = 100$ czujników w 58 iteracjach. Obliczenia bez zastosowania metody równoważenia obciążenia trwały 858 s, natomiast z wykorzystaniem metody równoważenia obciążenia obliczenia trwały 492 s, pozwalając w efekcie na zmniejszenie czasu obliczeń o 42 procent.

Dla wersji algorytmu bez metody równoważenia obciążenia uzyskano średnią wariancję udziału procesorów w obliczeniach $\Gamma(\pi) = 11.05$, a dla algorytmu z równoważeniem obciążenia średnia wariancja wynosiła $\Gamma(\pi) = 0.98$ (zob. rozdz. 3.8). Podobnie jak dla poprzedniego algorytmu, miara wariancji procentowego udziału procesorów w obliczeniach (zob. rozdz. 3.8) była obliczana na podstawie procentowego udziału każdego z procesorów w obliczeniach w każdej iteracji.



(a) (b)

Rysunek 4.16: Wykorzystanie czasu przez poszczególne procesory dla Algorytmu 4.9 oraz Przykładu 4.2 (klaster heterogeniczny): (a) bez równoważenia obciążenia, (b) z równoważeniem obciążenia.

4.7 Podsumowanie

W rozdziale zaproponowano dwie metody zrównoleglania procedur optymalnej aktywacji sieci czujników stacjonarnych oraz skanujących. Opierają się one na heurystycznych metodach optymalizacji dyskretnej. Analiza wymagań problemu pozwoliła na zidentyfikowanie czynników kluczowych dla działania wybranych metod oraz uzyskanie przyspieszenia obliczeń również dla algorytmów sekwencyjnych. Przedstawiona metoda redukcji czasochłonnych operacji oparta na formule Shermana-Morrisona-Woodbury'ego umożliwiła znaczną redukcję obliczeń związanych z koniecznością wyznaczania macierzy informacyjnej oraz odwrotności macierzy kowariancji pomiarów czujników.

W celu zrównoleglenia opracowanych metod, dokonano analizy algorytmów w celu ich dekompozycji. Zaproponowano zastosowanie dekompozycji przestrzeni poszukiwań opisującej różne konfiguracje aktywnych czujników. Jak pokazały przeprowadzone eksperymenty symulacyjne, zrównoleglenie pozwoliło na istotne skrócenie czasu obliczeń.

Zaproponowane metody wykorzystują różne sposoby i schematy komunikacji pomiędzy procesorami. Schemat „nadzorca-podwładni" ustanawia jeden z procesorów procesorem głównym, posiadającym całkowitą wiedzę na temat aktualnego stan algorytmu. Reszta z procesorów wykonuje ściśle wyspecyfikowane operacje na podstawie komunikatów odbieranych od procesora-nadzorcy. Taki schemat komunikacji wymaga dość znacznej ilości przesyłanych informacji pomiędzy procesorem-nadzorcą, a procesorami liczącymi i w konsekwencji może powodować opóźnienia w przesyłaniu danych z procesora głównego do procesorów podwładnych. Wydajność procesora głównego staje się również wąskim gardłem systemu równoległego rozwiązującego

dany problem.

W przypadku schematu komunikacji „każdy z każdym" procesory wymieniają między sobą znacznie mniejsze ilości danych. Mniejsza pula danych do przesłania pomiędzy procesorami jest konsekwencją umiejscowienia procedur realizujących cały algorytm we wszystkich procesorach liczących (w odróżnieniu od schematu „nadzorca-podwładni"). Taki schemat wymiany danych powoduje, że początkowa ilość przesyłanych danych nie zmniejsza efektywności algorytmu. Należy jednak zauważyć, że wraz ze wzrostem liczby procesorów, liczba danych do przesłania pomiędzy procesorami rośnie. Szybki wzrost ilości przesyłanych danych może w konsekwencji powodować dość duże opóźnienia w komunikacji pomiędzy procesorami i prowadzić do zwiększania się czasu bezczynności procesorów.

Zaproponowane metody zostały również przystosowane do uruchamiania w środowiskach heterogenicznych, w których jednostki liczące posiadają różną wydajność. Z względu na szybki rozwój technik komputerowych, tego typu środowiska obliczeniowe będą coraz częściej spotykane, a przedstawione metody będą potrafiły efektywnie wykorzystać ich moc obliczeniową.

5

Obliczenia równoległe w planowaniu trajektorii węzłów mobilnych sieci sensorycznych

Gwałtowny postęp techniki, a dzięki temu szybki spadek cen urządzeń, pozwolił na znaczne spopularyzowanie i dostępność rozmaitych przyrządów do wykonywania pomiarów. Przyrządy te coraz częściej mogą być umieszczana na platformach umożliwiających poruszanie się po powierzchni ziemi. W coraz większym stopniu dostępne są również pojazdy umożliwiające przenoszenie czujników w powietrzu oraz na wodzie. Popularność sieci czujników powoduje, że warto zająć się problemem optymalnego planowania rozmieszczania również czujników mających możliwość poruszania się.

Poniższy rozdział ma na celu przedstawienie problemu poszukiwania optymalnych trajektorii zarówno w sposób uproszczony (bez uwzględniania dynamiki pojazdów), jak i w sposób uwzględniający ograniczenia ruchu pojazdów. W celu rozwiązania przedstawionych zagadnień zaproponowano metody i algorytmy, które mogą być uruchamiane w środowisku klastrów obliczeniowych. Nietrywialność przedstawionych problemów, a zwłaszcza ich złożoność obliczeniowa, wielowymiarowość, czasochłonność oraz inne aspekty predestynują je do uruchamiania w wieloprocesorowych środowiskach obliczeniowych.

5.1 Przedstawienie problemu

Zagadnienie planowania optymalnych trajektorii czujników mobilnych jest zagadnieniem o wiele bardziej skomplikowanym niż rozmieszczanie lub aktywacja czujników

stacjonarnych czy też skanujących. Samo zagadnienie opisania trajektorii ruchu czujników, ze względu na różny sposób ich wyznaczania, nie jest trywialne. Dodatkowo, najczęściej wymaga się, aby podczas wyznaczania optymalnych trajektorii czujników mobilnych uwzględniać dynamikę poruszających się pojazdów. Nie każdy pojazd jest bowiem w stanie wykonać operację np. zmiany kierunku ruchu w czasie jaki wyznaczy mu algorytm. Inercja pojazdów jest tylko jedną z wielu własności fizycznych jakie należy brać pod uwagę na etapie planowania.

5.1.1 Opis trajektorii ruchu czujników

W celu skupienia uwagi, przedstawione rozważania dotyczą sytuacji, w których dziedziną określoności układu z czasoprzestrzenną dynamiką jest pewien obszar Ω w przestrzeni $\mathbb{R}^2$. Uogólnienie zaproponowanych metod i algorytmów na przypadek planowania trajektorii sensorów mobilnych w przestrzeni $\mathbb{R}^3$ nie stanowi jednak większego problemu.

W badaniach zakłada się również, że czujniki mogą się poruszać w pewnym zwartym obszarze dopuszczalnym $X \in \Omega$ zawartym w obszarze, w którym analizowany jest układ z czasoprzestrzenną dynamiką. Przykładem konieczności wprowadzenia obszaru dopuszczalnego może być analiza rozprzestrzeniania się zanieczyszczeń powietrza. W obszarze, w którym przeprowadzane są obserwacje może występować np. zbiornik wodny, który należy wykluczyć z obszaru położeń dopuszczalnych sensorów mobilnych.

5.1.2 Przedstawione podejścia

W niniejszej pracy zaproponowano opis ruchu mobilnych węzłów sieci sensorycznej w oparciu o:

- jawny parametryczny opis trajektorii z wykorzystaniem funkcji sklejanych,
- niejawny opis trajektorii za pomocą równań dynamiki pojazdów i sparametryzowanych sterowań.

Modelowanie ruchu pojazdów przenoszących czujniki z wykorzystaniem funkcji sklejanych stanowi pewne uproszczenie problemów, zjawisk oraz ograniczeń jakie występują w warunkach rzeczywistych. Zastosowanie funkcji sklejanych pozwala na opisanie kształtu ścieżki po jakiej powinny się poruszać pojazdy, jak również pozwala na wyznaczenie uproszczonego harmonogramu poruszania się pojazdów. Jednakże, takie podejście nie pozwala, w jawny sposób, na określenie ograniczeń dotyczących np. dynamiki poruszających się pojazdów, ich przyspieszenia, zużycia energii i wielu innych.

Uwzględnienie wielu ograniczeń opisanych powyżej umożliwia drugie zaproponowane podejście, w którym problem sprowadza się do zagadnienia sterowania optymalnego. Zastosowanie tej metody pozwala, nie tylko na dokładniejsze modelowanie

ruchu pojazdów przenoszących czujniki, ale również na uwzględnienie wielu czynników fizycznych związanych zarówno z samym ruchem pojazdu, jak i jego pracą w trakcie przenoszenia czujników podczas obserwacji obiektów z czasoprzestrzenną dynamiką.

Wyznaczanie trajektorii mobilnych węzłów sieci sensorycznej wymaga zastosowania metod optymalizacji ciągłej. Jak zostanie pokazane w niniejszym rozdziale, nawet stosunkowo proste przykłady określania optymalnych trajektorii powodują potrzebę rozwiązywania zagadnień złożonych numerycznie. Konieczność uwzględniania np. dynamiki poruszających się pojazdów wymaga zastosowania wysublimowanego aparatu matematycznego sterowania optymalnego. Szerokie spektrum koniecznych do rozważenia problemów ogólnych i szczegółowych czynią problem poszukiwania optymalnych trajektorii o wiele bardziej złożonym od zagadnienia optymalnej aktywacji czujników stacjonarnych lub skanujących.

5.1.3 Sformułowanie problemu

Problem optymalnej obserwacji z zastosowaniem sieci sensorów mobilnych można ogólnie sformułować jako poszukiwanie takich trajektorii ruchu węzłów sieci, aby obserwacje dostarczane przez poruszające się czujniki pozwalały na możliwie najdokładniejszą estymację nieznanych parametrów układu z czasoprzestrzenną dynamiką. Plan eksperymentu ξ składa się z trajektorii $x^j : \mathrm{T} = [0, t_f] \to \mathrm{X}, j = 1, \ldots, n$ węzłów sieci:

$$\xi = \left\{ x^1(\,\cdot\,), x^2(\,\cdot\,), \ldots, x^n(\,\cdot\,) \right\}, \tag{5.1}$$

W takim przypadku macierz informacyjna związana z aktualnym planem eksperymentu ξ opisującym trajektorie n czujników przedstawia się następująco

$$\mathcal{M}(\xi) = \int_T G(\xi, t) C^{-1}(\xi, t) G^{\mathsf{T}}(\xi, t)\, \mathrm{d}t, \tag{5.2}$$

gdzie:

$$G(\xi, t) = \left[g(x^1(t), t) \mid \ \ldots \ \mid g(x^n(t), t) \right], \tag{5.3}$$

$$g(x, t) = \left[\frac{\partial y(x, t; \theta)}{\partial \theta_1}, \ \ldots \ , \frac{\partial y(x, t; \theta)}{\partial \theta_m} \right]^{\mathsf{T}}_{\theta = \theta^0} \tag{5.4}$$

opisują wrażliwość stanu na zmiany parametrów, a

$$C(\xi, t) = [c_{ij}(\xi, t)], \quad c_{ij}(\xi, t) = q(x^i(t), x^j(t)), \tag{5.5}$$

opisuje kowariancję obserwacji czujników.

5.2 Optymalizacja sparametryzowanych trajektorii czujników ruchomych

Poszukiwany plan eksperymentu ξ powinien wyznaczać ekstremum wybranego kryterium optymalności. Jednak wyznaczone chwilowe położenia czujników powinny

składać się na pewną możliwą do zrealizowania trajektorię ruchu tych czujników. Odpowiedni kształt ścieżki, po której poruszać się mają mobilne węzły sieci sensorycznej oraz harmonogram poruszania się tych węzłów muszą być możliwe do uzyskania za pomocą dostępnych urządzeń. W niniejszym podrozdziale zakłada się opis trajektorii z zastosowaniem sześciennych funkcji sklejanych.

5.2.1 Parametryzacja trajektorii

Naturalnym wydaje się założenie, że trajektorie pojedynczego czujnika można przybliżyć następującym opisem parametrycznym:

$$x^i(t) = \eta(t, \omega^i), \quad t \in T, \tag{5.6}$$

gdzie: η — pewna funkcja spełniająca dwa warunki:

- $\eta(\,\cdot\,, \omega^i)$ jest ciągła dla każdej ustalonej wartości ω^i, $i = 1, \ldots, n$,
- $\eta(t, \,\cdot\,)$ jest ciągła dla każdej ustalonej wartości t.

Dodatkowo, parametry ω muszą opisywać trajektorie w ten sposób, aby czujniki poruszały się wewnątrz założonego obszaru dopuszczalnego $X \in \Omega$, czyli zbiór parametrów ω musi należeć do zbioru

$$A_{dop} = \{\omega \in A : \eta(t, \omega) \in X, \forall t \in T\} \tag{5.7}$$

gdzie: A — zbiór dopuszczalnych wartości parametrów ω.

Powyższe rozważania dotyczą poszukiwania optymalnych trajektorii czujników poruszających się w przestrzeni dwuwymiarowej. Trajektorię ruchu pojedynczego czujnika można więc opisać jako

$$x^i(t) = (\eta_1(t, \omega_1^i), \eta_2(t, \omega_2^i)). \tag{5.8}$$

gdzie: ω_j^i — parametry opisujące ruch i-tego czujnika w kierunku j-tej osi układu współrzędnych (w rozważanych zagadnieniach $j = 1, 2$).

W literaturze można spotkać wiele sposobów parametryzacji trajektorii [115]. W przedstawionych badaniach zaproponowano parametryzację trajektorii z wykorzystaniem splajnów (funkcji sklejanych) sześciennych (zob. Dodatek A.3). Parametryzacja trajektorii z zastosowaniem splajnów należących do funkcji klasy $\mathcal{C}^2$, tzn. posiadających ciągłe dwie pierwsze pochodne, pozwala na uzyskanie gładkich ścieżek, po których poruszać się będą mobilne węzły sieci sensorycznej, oraz harmonogramu nie powodującego gwałtownych zmian przyspieszenia oraz prędkości mobilnych węzłów sieci sensorycznej.

W szczególności, wykorzystano liniową kombinację funkcji bazowych będących splajnami sześciennymi:

$$\eta_j(t, \omega^i) = \sum_{\ell=1}^{L} \omega_{j\ell}^i \phi_\ell(t), \quad j = 1, 2, \tag{5.9}$$

gdzie: L — liczba uwzględnionych B-splajnów, determinujca dokładność aproksymacji optymalnych trajektorii.

W rezultacie, dla i-tego czujnika poszukuje się wektora

$$\omega^i = (\omega^i_{11}, \omega^i_{21}, \omega^i_{12}, \omega^i_{22}, \ldots, \omega^i_{1L}, \omega^i_{2L}) \in \mathbb{R}^{2L}. \tag{5.10}$$

5.2.2 Sformułowanie problemu

Opisanie trajektorii przy pomocy krzywych parametrycznych powoduje, że zgodnie z (5.2)–(5.5) rozważany problem formułuje się jako szukanie minimum kryterium optymalności

$$\Psi[\mathcal{M}(\xi)], \tag{5.11}$$

zdefiniowanego na macierzy informacyjnej postaci

$$\mathcal{M}(\xi) = \int_T G(\xi, t) C^{-1}(\xi, t) G^{\mathsf{T}}(\xi, t)\, \mathrm{d}t, \tag{5.12}$$

gdzie:

$$G(\xi, t) = \left[g(\eta(t, \omega^1), t) \mid \ \ldots \ \mid g(\eta(t, \omega^n), t) \right], \tag{5.13}$$

$$g(\eta(t, \omega^i), t) = \left[\frac{\partial y(\eta(t, \omega^i), t; \theta)}{\partial \theta_1}, \ \ldots \ , \frac{\partial y(\eta(t, \omega^i), t; \theta)}{\partial \theta_m} \right]^{\mathsf{T}}_{\theta = \theta^0}, \tag{5.14}$$

$$C(\xi, t) = [c_{ij}(\xi, t)], \quad c_{ij}(\xi, t) = q(\eta(t, \omega^i), \eta(t, \omega^j)), \tag{5.15}$$

Wyznaczany plan eksperymentu ma postać

$$\xi = \omega = \left\{ \omega^1, \omega^2, \ldots, \omega^n \right\}. \tag{5.16}$$

co, z nieznacznym nadużyciem notacji, pozwala na zapisanie kryterium optymalności w uproszczonej postaci

$$\Psi = \Psi(\omega). \tag{5.17}$$

Reprezentacja trajektorii czujników jako krzywych parametrycznych prowadzi do funkcji celu zależącej od stosunkowo dużej liczby zmiennych decyzyjnych. W dalszych rozważaniach dokonywana będzie optymalizacja funkcji $n \times d \times L$ zmiennych niezależnych, gdzie: n — liczba czujników, d — liczba zmiennych przestrzennych, L — liczba funkcji bazowych. Przykładowo, wyznaczenie trajektorii ruchu pięciu węzłów mobilnej sieci sensorycznej poruszających się w obszarze dopuszczalnym przestrzeni $\Omega \in \mathbb{R}^2$, w przypadku opisania trajektorii za pomocą 20 funkcji bazowych, powoduje konieczność optymalizacji kryterium optymalności opisanego za pomocą 200 zmiennych decyzyjnych. Tak znaczna wymiarowość problemu dla stosunkowo prostych zagadnień wymaga opracowania metody pozwalającej na efektywną optymalizację globalną silnie nieliniowych funkcji dużej liczby zmiennych.

5.2.3 Proponowane rozwiązanie

Skończenie-wymiarowa parametryzacja problemu poszukiwania optymalnych trajektorii mobilnych węzłów sieci sensorycznej prowadzi do zadania programowania nieliniowego z ograniczeniami o stosunkowo dużej liczbie zmiennych decyzyjnych. W miarę dokładna aproksymacja problemu poruszania się kilku sensorów mobilnych w obszarze dwuwymiarowym z wykorzystaniem metody przedstawionej w rozdz. 5.2.1 wymaga zastosowania algorytmu pozwalającego na optymalizację funkcji o dużej liczbie zmiennych decyzyjnych, mogącej posiadać wiele minimów lokalnych.

Do rozwiązania tak przedstawionego zagadnienia można zastosować algorytmy genetyczne [158]. Zaletą algorytmów genetycznych jest możliwość znajdowania optimum globalnego funkcji o bardzo dużej liczbie zmiennych decyzyjnych. Jednak proces poszukiwania optimum z zastosowaniem algorytmów genetycznych wiąże się z koniecznością wielokrotnego wyznaczania wartości funkcji celu. W przypadku rozważanych problemów poszukiwania optymalnych trajektorii mobilnych węzłów sieci sensorycznej, gdzie funkcja celu ma postać mocno uwikłaną, wielokrotne wyznaczanie jej wartości spowodowałoby znaczne spowolnienie działania metody opierającej się na algorytmach genetycznych.

Stąd, do rozwiązania tak przedstawionego zagadnienia zaproponowano wykorzystanie schematu algorytmu tunelowego zaproponowanego w pracach [75, 125, 126]. Algorytm tunelowy składa się z dwóch faz: fazy minimalizacji lokalnej oraz fazy tunelowej. Fazy te są używane naprzemiennie w celu aproksymacji minimum globalnego zadanej funkcji $f(x)$. W fazie minimalizacji lokalnej, dla zadanego punktu startowego $x^{(k)}$ wykorzystujemy jakikolwiek znany algorytm minimalizacji lokalnej w celu znalezienia punktu minimum $f(x)$ (oznaczonego jako $x^{(k*)}$). W fazie tunelowej, konstruuje się funkcję pomocniczą $\mathcal{T}(x)$, tzw. funkcję tunelową, która jest funkcją różniczkowalną zależną od pewnego zadanego zbioru parametrów oraz punktu $x^{(k*)}$. Wykorzystując tę funkcję poszukujemy nowego punktu $x^{(k+1)}$ takiego, że $\mathcal{T}(x^{(k+1)}) \leqslant 0$, startując z punktu znajdującego się w sąsiedztwie $x^{(k*)}$. Poprzez odpowiednią definicję funkcji tunelowej, implikować powinno to $x^{(k+1)} \neq x^{(k)}$ oraz $f(x^{(k+1)}) \leqslant f(x^{(k)})$. Naprzemiennie wywołując te dwie fazy, otrzymuje się sekwencję minimów lokalnych takich, że wartość oryginalnej funkcji celu dla każdego kolejnego z tych minimów jest nie większa od poprzednio znalezionych minimów.

5.2.4 Algorytm tunelowy

Jak wspomniano wyżej algorytm tunelowy należy do klasy algorytmów optymalizacji globalnej. Jego działanie polega na naprzemiennym wykonywaniu dwóch faz:

- **fazy minimalizacji lokalnej**.
  Celem tej fazy jest znalezienie minimum lokalnego $x^{(k*)}$ startując od znanego punktu $x^{(k)}$. Do celów minimalizacji wykorzystuje się jakikolwiek znany algorytm minimalizacji lokalnej.

- **fazy tunelowej**.
  Startując z punktu $x^{(k*)}$, celem tej fazy jest znalezienie punktu $x^{(k+1)}$ takiego, że $f(x^{(k*)}) = f(x^{(k+1)})$, ale nie będącego minimum lokalnym.

W celu znalezienia nowego punktu $x^{(k+1)}$, w fazie tunelowej konstruuje się funkcję pomocniczą, tzw. funkcję tunelową, postaci:

$$x \mapsto \mathcal{T}(x, x^{(k*)}, \lambda^{(k)}) = \frac{f(x) - f(x^{(k*)})}{[(x - x^{(k*)})^{\mathsf{T}}(x - x^{(k*)})]^{\lambda^{(k)}}} \tag{5.18}$$

gdzie: $\frac{1}{2}\lambda^{(k)}$ — stopień pierwiastka równania $\tilde{f}(x) \equiv f(x) - f(x^{(k*)}) = 0$ w punkcie $x = x^{(k*)}$.

Dzięki takiej postaci, dla pewnej wartości $\lambda^{(k)}$, funkcja tunelowa posiada wartość $\mathcal{T}(x, x^{(k*)}, \lambda^{(k)}) \neq 0$ w punkcie $x = x^{(k*)}$, a punkt $x^{(k+1)}$ będący pierwiastkiem równania $\tilde{f}(x) = 0$ może być wykorzystany jako punkt startowy do wyznaczenia nowego minimum lokalnego.

Parametr $\lambda^{(k)}$ nie jest znany i musi zostać wyznaczony podczas działania fazy tunelowej. W tym celu funkcja tunelowa $\mathcal{T}$ jest wyznaczana wielokrotnie w punkcie $x = x^{(k*)}$ dla rosnących wartości parametru $\lambda^{(k)}$. Gdy dla pewnej wartości $\lambda^{(k)}$ zostanie spełniony warunek

$$\mathcal{T}\Big(x^{(k*)}, x^{(k*)}, \lambda^{(k)}\Big) \neq 0, \tag{5.19}$$

oraz w pewnym otoczeniu Δx punktu $x^{(k*)}$ będzie zachodzić

$$\mathcal{T}\Big(x^{(k*)} + \Delta x, x^{(k*)}, \lambda^{(k)}\Big) \leqslant \mathcal{T}\Big(x^{(k*)}, x^{(k*)}, \lambda^{(k)}\Big). \tag{5.20}$$

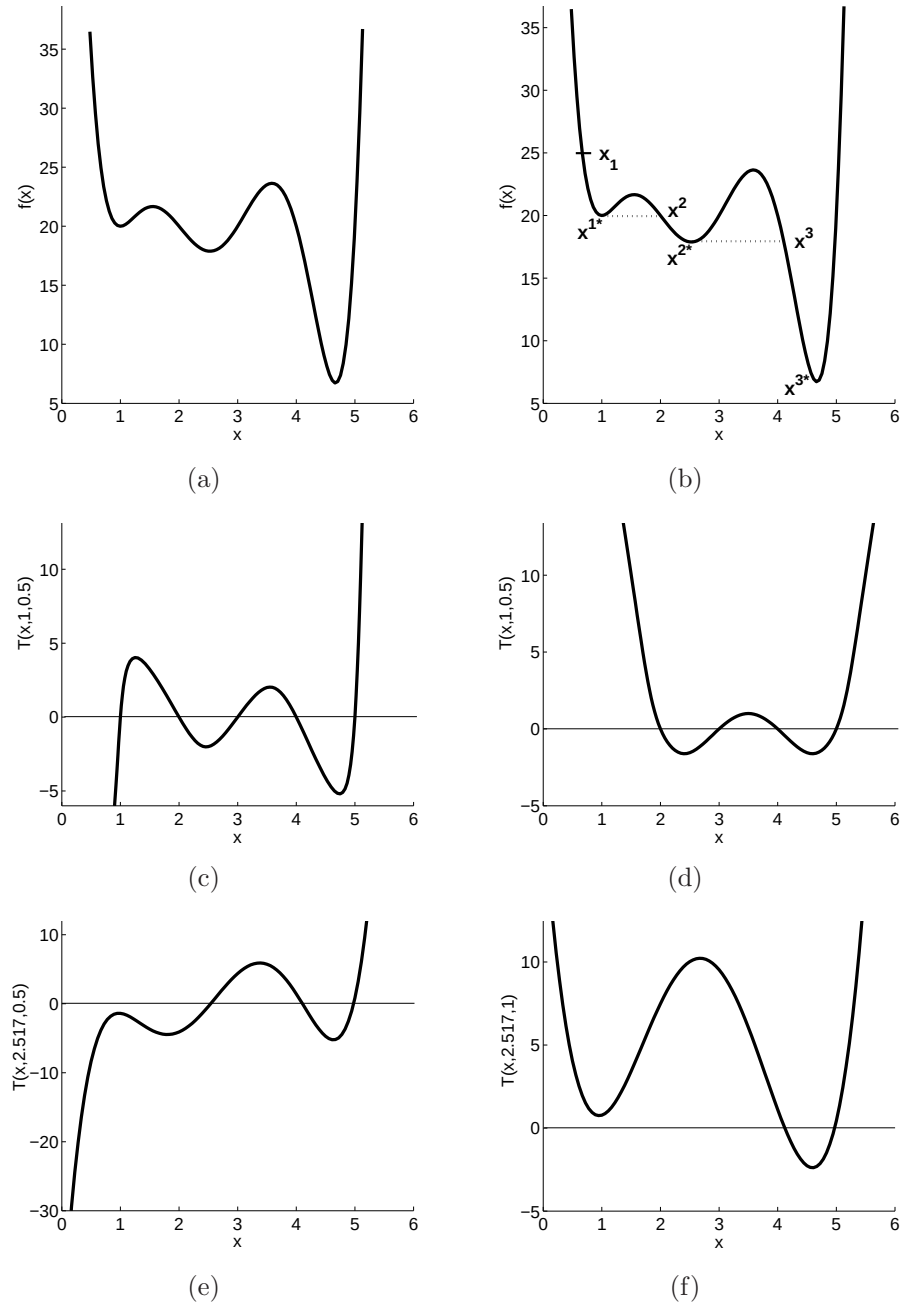
to aktualna wartość $\lambda^{(k)}$ jest szukanym stopniem pierwiastka funkcji, dla którego wyznaczany jest wstępnie punkt $x = x^{(k+1)}$.

Przykład 5.1
W celu przedstawienia działania algorytmu tunelowego, rozważmy prosty przykład znajdowania optimum globalnego funkcji jednej zmiennej [1] (zob. rys. 5.1(a)):

$$f(x) = x^6 - 16x^5 + 100x^4 - 310x^3 + 499x^2 - 394x + 140. \tag{5.21}$$

Działanie algorytmu prześledźmy dla punktu startowego $x^{(1)}$ (zob. rys. 5.1(b)). Wykonanie fazy minimalizacji lokalnej pozwala na znalezienie minimum lokalnego w punkcie $x^{(1*)}$, gdzie funkcja ma wartość $f(1) = 20$. Następnie wykonywana jest faza tunelowa. Funkcja tunelowa $\mathcal{T}(x, 1, \lambda^{(1)})$ dla wartości parametru $\lambda^{(1)} = 0.5$ przedstawiona jest na rys. 5.1(c) i zgodnie z (5.18) ma postać:

$$\mathcal{T}(x, 1, 0.5) = \frac{f(x) - f(1)}{[(x-1)(x-1)]^{0.5}} = x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120. \tag{5.22}$$



Rysunek 5.1: Przykład działania algorytmu tunelowego:
(a) wykres oryginalnej funkcji celu,
(b) kolejno wyznaczane minima lokalne,
(c) wykres funkcji tunelowej $\mathcal{T}(x, 1, 0.5)$,
(d) wykres funkcji tunelowej $\mathcal{T}(x, 1, 1)$,
(e) wykres funkcji tunelowej $\mathcal{T}(x, 2.517, 0.5)$,
(f) wykres funkcji tunelowej $\mathcal{T}(x, 2.517, 1)$.

Jak można zauważyć funkcja $\mathcal{T}(x, 1, 0.5)$ nie spełnia warunku (5.19), dlatego rozważana jest funkcja tunelowa $\mathcal{T}(x, 1, \lambda^{(1)})$ dla wartości parametru $\lambda^{(1)} = 1.0$, której wykres przedstawia rys. 5.1(d). Postać analityczna tej funkcji przedstawia się następująco

$$\mathcal{T}(x, 1, 1) = \frac{f(x) - f(1)}{[(x-1)(x-1)]} = x^4 - 14x^3 + 71x^2 - 154x + 120. \tag{5.23}$$

Funkcja (5.23) spełnia warunki (5.19),(5.20) i posiada swój pierwiastek w punkcie $x^{(2)} = 2.0$ (zob. rys. 5.1(d)). Startując z tego punktu fazę minimalizacji lokalnej otrzymuje się po jej zakończeniu kolejne minimum lokalne w punkcie $x^{(2*)} = 2.517$.

Kolejnym krokiem jest ponownie faza tunelowa. Funkcja tunelowa $\mathcal{T}(x, 2.517, \lambda^{(2)})$ dla wartości parametru $\lambda^{(2)} = 0.5$ przedstawiona jest na rys. 5.1(e) i zgodnie z (5.18) ma postać:

$$\begin{aligned} \mathcal{T}(x, 2.517, 0.5) &= \frac{f(x) - f(1)}{[(x-2.517)(x-2.517)]^{0.5}} \\ &= x^5 - 13.483x^4 + 66.063289x^3 \\ &\quad - 143.7187016x^2 + 137.2600281x - 48.51650926. \end{aligned} \tag{5.24}$$

Jak można zauważyć, podobnie jak w kroku poprzednim, funkcja $\mathcal{T}(x, 2.517, 0.5)$ nie spełnia warunku (5.19), dlatego rozważana jest funkcja tunelowa $\mathcal{T}(x, 2.517, \lambda^{(2)})$ dla wartości parametru $\lambda^{(2)} = 1.0$, której wykres przedstawia rys. 5.1(f). Postać analityczna tej funkcji przedstawia się następująco

$$\begin{aligned} \mathcal{T}(x, 2.517, 1) &= \frac{f(x) - f(1)}{[(x-2.517)(x-2.517)]} \\ &= x^4 - 10.966x^3 + 38.461867x^2 - 46.91018235x + 19.18709914. \end{aligned} \tag{5.25}$$

Funkcja (5.25) spełnia warunki (5.19),(5.20) i posiada swój pierwiastek w punkcie $x^{(3)} = 4.12$ (zob. rys. 5.1(f)). Startując z tego punktu fazę minimalizacji lokalnej oryginalnej funkcji $f(x)$ otrzymuje się po jej zakończeniu w punkcie $x^{(3*)} = 4.6685$ kolejne minimum lokalne $f(x^{(3*)}) = 6.72128$, które jest jednocześnie minimum globalnym.

♦

Ponieważ nie dla każdej optymalizowanej funkcji $f(x)$ można wyznaczyć analityczną postać funkcji tunelowej $\mathcal{T}$ w zależności od parametru $\lambda^{(k)}$, w praktyce, w celu zapobiegnięciu wykonywania operacji dzielenia przez zero przy wyznaczaniu wartości funkcji (5.18) jako punkty startowe wykorzystuje się punkty należące do otoczenia aktualnego minimum postaci

$$\tilde{x} \in \mathcal{B}(x^{(k*)}, \varepsilon), \tag{5.26}$$

gdzie: $\mathcal{B}(x^{(k*)}, \varepsilon)$ — kula o środku w punkcie $x^{(k*)}$ i promieniu ε, ε — mała liczba dodatnia.

Fazę tunelową można rozważać w kontekście minimalizacji funkcji tunelowej (5.18) dla różnych punktów startowych określonych jako (5.26) z warunkiem stopu postaci

$$\mathcal{T}(x) \leqslant 0. \tag{5.27}$$

Pozwala to na zastosowanie znanych metod optymalizacji lokalnej do wyznaczania, nie tylko minimum lokalnego w fazie minimalizacji lokalnej, ale również do minimalizacji funkcji tunelowej.

W ogólności, optymalizowana funkcja $f(x)$ jest najczęściej funkcją nieliniową wielu zmiennych decyzyjnych. Stąd może posiadać wiele minimów lokalnych i zbieżność do tych minimów na tym samym poziomie jest bardzo prawdopodobna, tzn. może się okazać, że $f(x^{(1*)}) = f(x^{(2*)}) = \cdots = f(x^{(q*)})$ dla pewnego $q \geqslant 2$. Aby zapobiec cyklicznemu powracaniu do już znalezionych punktów minimum o tej samej wartości, ważne jest zachowanie historii lokalnych minimów do czasu znalezienia lepszego minimum $x^{(q+1*)}$, dla którego funkcja celu przyjmuje wartość mniejszą niż napotkane do tej pory. Aby ten cel zrealizować, potrzebna jest modyfikacja funkcji tunelowej postaci

$$\mathcal{T}(x) = \frac{f(x) - f(x^{(q*)})}{\prod_{\ell=1}^{q-1}[(x - x^{(\ell*)})^{\mathsf{T}}(x - x^{(\ell*)})]^{\lambda^{(\ell)}}[(x - x^{(q*)})^{\mathsf{T}}(x - x^{(q*)})]^{\lambda^{(q)}}} \tag{5.28}$$

W równaniu (5.28) musi zostać zidentyfikowany tylko parametr $\lambda^{(q)}$, gdyż wartości pozostałych parametrów $\{x^{(1*)}, x^{(2*)}, \ldots, x^{(q-1*)}\}$ oraz $\lambda = \{\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(q-1)}$ zostały wyznaczone w poprzednich iteracjach algorytmu. Dodatkowo należy przyjąć $q = 1$ zaraz jak tylko znaleziony punkt $x^{(q+1*)}$ gwarantuje mniejszą wartość optymalizowanej funkcji niż $f(x^*)$. Szczegółowa analiza i opis wyznaczania kolejnych wartości parametrów przedstawiono w pracach [125, 126], gdzie wprowadza się również modyfikacje historycznych wartości parametrów λ.

Podczas poszukiwania minimum funkcji tunelowej może zaistnieć sytuacja, gdy warunek stopu (5.27) nie zostanie osiągnięty, gdyż funkcja tunelowa ze względu na swoją nieliniowość, może również posiadać minima lokalne. W takiej sytuacji, pomimo poprawnej identyfikacji parametru $\lambda^{(k)}$ proces minimalizacji funkcji tunelowej zostanie zatrzymany w pewnym punkcie krytycznym x^m (np. minimum lokalnym, punkcie siodłowym). W celu umożliwienia ominięcia tego punktu, wprowadza się modyfikację funkcji tunelowej postaci:

$$\mathcal{T}(x) = \frac{f(x) - f(x^{(q*)})}{\prod_{\ell=1}^{q}[(x - x^{(\ell*)})^{\mathsf{T}}(x - x^{(\ell*)})]^{\lambda^{(\ell)}}[(x - x^{m})^{\mathsf{T}}(x - x^{m})]^{\lambda^{m}}} \tag{5.29}$$

Wprowadzenie dodatkowego członu odpowiadającego za ominięcie punktu krytycznego x^m wymaga identyfikacji parametru λ^m, co wiąże się z pewnym dodatkowym nakładem obliczeń. Procedurę wyznaczania parametru λ^m przedstawiono w pracach [125, 126]. W momencie, gdy podczas minimalizacji funkcji tunelowej (5.29), zostanie spełniony warunek stopu (5.27), ulega ona uproszczeniu do postaci (5.18).

5.2.5 Ograniczenia

Oryginalna postać algorytmu tunelowego nie pozwala na definiowanie ograniczeń na zmienne decyzyjne. Zagadnienie wyznaczania optymalnych trajektorii wymaga istnienia ograniczeń (5.7) w celu zapewnienia, że mobilne węzły sieci sensorycznej poruszać się będą w obszarze dopuszczalnym $X \in \Omega$. Obszar ten jest najczęściej zadany układem nierówności

$$b_j(x) \leqslant 0, \quad j = 1, \ldots, J, \tag{5.30}$$

gdzie: $b_j(\cdot)$ — pewna znana funkcja.

W celu rozwiązania tego problemu powyżej zaproponowano wykorzystanie zewnętrznej funkcji kary [166] postaci

$$P(\omega, c) = c \sum_{i=1}^{n} \sum_{j=1}^{J} \int_T \Big[\max(0, b_j(\eta(t, \omega^i)))\Big]^2 \, \mathrm{d}t \tag{5.31}$$

gdzie:

$\eta(t, \omega^i)$ — funkcja określająca położenie i-tego czujnika (zob. rozdz. 5.2.1),

c — parametr określający wpływ funkcji kary na oryginalną funkcję celu.

W przypadku, gdy którykolwiek czujnik znajdzie się w jakimkolwiek momencie poza obszarem dopuszczalnym X, następuje aktywacja funkcji kary, powodująca zwiększenie wartości minimalizowanej funkcji celu, a tym samym pogorszenie własności aktualnego wektora zmiennych decyzyjnych ω opisujących aktualną trajektorię mobilnych węzłów sieci sensorycznej.

Wprowadzenie funkcji kary powoduje, że oryginalne kryterium optymalności $\Psi(\omega)$ zostaje zastąpione rozszerzonym kryterium postaci

$$\phi(\omega) = \Psi(\omega) + P(\omega, c), \tag{5.32}$$

W celu uproszczenia obliczeń w niniejszej pracy przyjęto, że wartość parametru c jest określana arbitralnie na starcie algorytmu, na podstawie doświadczeń wstępnych.

5.2.6 Proponowany algorytm

Zaproponowany algorytm ma na celu znajdowanie optymalnych trajektorii mobilnych węzłów sieci sensorycznej. W porównaniu do oryginalnie zaproponowanego rozwiązania w postaci algorytmu tunelowego, wprowadzone zostały pewne uproszczenia mające na celu przyspieszenie wykonywania obliczeń, jednak bez drastycznego zmniejszenia efektywności algorytmu.

Zaproponowany algorytm ma na celu określenie parametrów trajektorii mobilnych węzłów sieci sensorycznej skumulowanych w wektorze ω utożsamianym jednocześnie z optymalnym planem eksperymentu ξ. Ogólny schemat przedstawiono w

postaci Algorytmu 5.1. Podobnie jak oryginalny algorytm tunelowy, również zaproponowany algorytm składa się z dwóch faz. W fazie minimalizacji lokalnej następuje poszukiwanie wektora $\omega^{(k*)}$, który pozwala na osiągnięcie minimum lokalnego funkcji celu ϕ. W przedstawionej pracy wykorzystano metodę minimalizacji Rosenbrocka [194] (zob. Dodatek A.4). Jest to metoda niewymagająca wyznaczania gradientu minimalizowanej funkcji. Ze względu na skomplikowaną postać funkcji celu (zob. rozdz. 5.2.2) konieczność wyznaczania gradientu funkcji celu, jaka ma miejsce np. w metodzie gradientów sprzężonych, czy też hesjanu, co ma miejsce w metodzie Netwtona, spowodowałoby znaczne zwiększenie nakładu obliczeniowego wymaganego w procesie minimalizacji wymaganych funkcji. W fazie tunelowej poszukuje się punktu pozwalającego na znalezienie tunelu do doliny, w której możliwe jest wyznaczenie lepszego, pod względem wybranego kryterium optymalności, minimum lokalnego. Fazę tunelową dokładniej opisano w postaci Algorytmu 5.2.

Algorytm 5.1 Optymalizacja sparametryzowanych trajektorii mobilnych węzłów sieci sensorycznej

Dane wejściowe:
$\omega^{(0)}$ — wektor parametrów opisujący wstępne przybliżenie trajektorii czujników.

Dane wyjściowe:
ϕ_{opt} — obliczona optymalna wartość rozszerzonego kryterium optymalności,
ω_{opt} — wektor parametrów opisujący ostateczne przybliżenia optymalnych trajektorii czujników.

Krok 0: (Inicjalizacja)
Ustawienie przybliżenia optymalnego planu eksperymentu na wartość początkową $\omega_{\text{opt}} \leftarrow \omega^{(0)}$ oraz wyznaczenie wartości kryterium $\phi_{\text{opt}} \leftarrow \phi(\omega^{(0)})$. Inicjalizacja licznika iteracji $k \leftarrow 0$.

Krok 1: (Faza minimalizacji lokalnej)
Poszukiwanie minimum lokalnego $\omega^{(k*)}$ funkcji ϕ startując z punktu $\omega^{(k)}$. Znalezione minimum lokalne staje się aktualnym optimum globalnym $\omega_{\text{opt}} \leftarrow \omega^{(k*)}$.

Krok 2: (Faza tunelowa)
Przy pomocy odpowiednio skonstruowanej funkcji pomocniczej (tunelowej) $\mathcal{T}$, startując z punktu $\omega^{(k*)}$, poszukiwanie punktu $\omega^{(k+1)}$ takiego, że $\phi(\omega^{(k*)}) \geqslant \phi(\omega^{(k+1)})$ i jednocześnie nie będącego minimum lokalnym funkcji celu ϕ (zob. Alg. 5.2).

Krok 3: (Warunek stopu)
Jeżeli udało się znaleźć punkt $\omega^{(k+1)}$ taki, że $\mathcal{T}(\omega^{(k+1)}) \leqslant 0$ to następuje zwiększenie licznika iteracji k i skok to Kroku 1 w celu określenie nowego minimum

lokalnego.
Jeżeli w fazie tunelowej nie udało się znaleźć minimum funkcji tunelowej takiego, że $\mathcal{T}(\omega^{k+1}) \leqslant 0$ to przyjmuje się, że aktualna wartość punktu $\omega^{(k*)}$ określa poszukiwane minimum globalne i jednocześnie optymalny plan eksperymentu $\omega_{\text{opt}} \leftarrow \omega^{(k*)}$.

■

Po wystartowaniu z punktu $\omega^{(k*)}$, faza tunelowa ma na celu wyznaczenie takiego punktu $\omega^{(k+1)}$, dla którego $\phi(\omega^{(k*)}) = \phi(\omega^{(k+1)})$ a jednocześnie nowo wyznaczony punkt $\omega^{(k+1)}$ nie jest minimum lokalnym oryginalnej funkcji celu ϕ. W tym celu wykorzystana została funkcja pomocnicza (tunelowa) postaci (5.35). Startując minimalizację tej funkcji od pewnego punktu $\bar{\omega}$, szukamy takiego punktu $\hat{\omega}$, dla którego funkcja tunelowa będzie spełniała warunek $\mathcal{T}(\hat{\omega}) \leqslant 0$.

Zgodnie z informacjami z rozdz. 5.2.4, punkt startowy $\bar{\omega}$ wyznaczany jest w otoczeniu wyznaczonego wcześniej minimum lokalnego $\omega^{(k*)}$ zgodnie z (5.34). Ma to na celu zapobiegnięcie dzielenia przez zero podczas minimalizacji funkcji tunelowej (zob. rozdz. 5.2.4). Określenie punktu startowego w otoczeniu minimum lokalnego odbywa się na podstawie aktualnej wartości minimum lokalnego, wartości określającej wielkość sąsiedztwa definiującego wymagane otoczenie wokół minimum lokalnego oraz wartości losowej zgodnie z procedurą:

$$\mathcal{E}(\omega^{(k*)}, \varepsilon) = P_{\text{X}}\Big(\omega^{(k*)} + \varepsilon r \zeta\Big), \tag{5.33}$$

gdzie:
r — realizacja zmiennej losowej o rozkładzie równomiernym w przedziale $(0, 1]$,
ε — ustalony promień sąsiedztwa,
ζ — realizacja wektora losowego o wymiarze zgodnym z wymiarem $\omega^{(k*)}$ i rozkładzie równomiernym na sferze jednostkowej,
P_{X} — operator rzutowania na zbiór X.

Dla każdego z punktów startowych $\bar{\omega}$ dokonywana jest minimalizacja funkcji tunelowej dla zwiększanego stopnia pierwiastka funkcji tunelowej $\lambda^{(1)}$ (zob. rozdz. 5.2.4). Jeżeli parametr $\lambda^{(1)}$ osiągnie pewne określone arbitralnie maksimum, co oznacza, że wcześniejsze próby określenia punktu $\hat{\omega}$ takiego, że $\mathcal{T}(\hat{\omega}) \leqslant 0$ nie powiodły się, następuje wygenerowanie nowego punktu startowgo $\bar{\omega}$, dla którego ponownie wykonywana jest procedura znajdowania punktu $\hat{\omega}$ takiego, że $\mathcal{T}(\hat{\omega}) \leqslant 0$.

W przypadku, gdy nie uda się znaleźć punktu $\omega^{(k+1)}$ w najbliższym otoczeniu minimum lokalnego, parametr sterujący wielkością sąsiedztwa ε jest modyfikowany tak, aby nowo generowane punkty $\bar{\omega}$ mogły się znajdować w większym otoczeniu punktu $\omega^{(k*)}$. Zwiększenie sąsiedztwa pomaga w dokładniejszej eksploracji przestrzeni rozwiązań w celu znalezienia tunelu umożliwiającego wyznaczenie nowego minimum lokalnego funkcji celu ϕ.

Pewnym uproszczeniem w stosunku do oryginalnego algorytmu tunelowego jest brak uwzględniania minimów lokalnych znajdujących się na tym samym poziomie

(zob. (5.28)). Doświadczenie pokazuje, że dla rozważanych zagadnień, gdy funkcja celu dużej liczby zmiennych decyzyjnych jest bardzo silnie nieliniowa, praktycznie nie jest możliwe wylosowanie nowego punktu startowego w fazie tunelowej w kierunku umożliwiającym powrót do wcześniej wyznaczonego minimum lokalnego lub wyznaczenie minimum lokalnego funkcji celu ϕ o wartości identycznej jako poprzednio.

Algorytm 5.2 Faza tunelowa

Dane wejściowe:

$\omega^{(k*)}$ — aktualne minimum lokalne funkcji celu ϕ,
$\lambda^{(k)}$ — początkowa wartość stopnia pierwiastka funkcji tunelowej,
$\lambda^{(\max)}$ — maksymalna wartość stopnia pierwiastka funkcji tunelowej,
$\Delta\lambda$ — krok zwiększania parametru λ,
γ_1 — maksymalna liczba punktów startowych do poszukiwania nowego tunelu w sąsiedztwie aktualnego minimum $\omega^{(k*)}$,
γ_2 — maksymalna liczba punktów startowych do poszukiwania nowego tunelu w powiększonym sąsiedztwie aktualnego minimum $\omega^{(k*)}$,
ε — wielkość sąsiedztwa z jakiego ma startować proces minimalizacji funkcji tunelowej.

Dane wyjściowe:

$\omega^{(k+1)}$ — znaleziony punkt pozwalający na określenie nowego minimum funkcji.

Krok 0: (Inicjalizacja)

Wyzerowanie licznika iteracji $\ell \leftarrow 0$.

Krok 1: (Wyznaczenie punktu startowego)

Wyznaczenie punktu startowego dla procedury minimalizacji funkcji tunelowej

$$\bar{\omega} \leftarrow \mathcal{E}(\omega^{(k*)}, \varepsilon), \tag{5.34}$$

gdzie: $\mathcal{E}(\omega^{(k*)}, \varepsilon)$ — funkcja zwracająca losowe przesunięcie względem minimum lokalnego na podstawie wielkości sąsiedztwa ε (zob. (5.33)).

Krok 2: (Minimalizacja funkcji tunelowej)

Dla aktualnego parametru λ i punktu startowego $\bar{\omega}$ minimalizacja funkcji tunelowej postaci

$$\mathcal{T}(\omega) = \frac{\phi(\omega) - \phi(\omega^{(k*)})}{[(\omega - \omega^{(k*)})^{\mathsf{T}}(\omega - \omega^{(k*)})]^{\lambda^{(k)}}} \tag{5.35}$$

aż do momentu, gdy dla pewnego $\hat{\omega}$ warunek $\mathcal{T}(\hat{\omega}) \leqslant 0$ zostanie spełniony.

Krok 3: (Warunek stopu)

Faza 1: (Warunek znalezienia minimum funkcji tunelowa)
Jeżeli warunek $\mathcal{T}(\hat{\omega}) \leqslant 0$ został spełniony to następuje koniec fazy tunelowej, a wektor $\omega^{(k+1)}$ jest określony przez punkt $\hat{\omega}$.

Faza 2: (Warunek zwiększenia stopnia pierwiastka funkcji tunelowa)
Jeżeli nie znaleziono punktu ω takiego, że $\mathcal{T}(\omega) \leqslant 0$ to następuje zwiększenie wartości parametru $\lambda^{(k)} \leftarrow \lambda^{(k)} + \Delta\lambda$ i skok do Kroku 2.

Faza 3: (Warunek minimalizacji funkcji tunelowej z innego punktu startowego)
Jeżeli $\lambda^{(k)} \geqslant \lambda^{(\max)}$ to następuje zwiększenie licznika iteracji $\ell \leftarrow \ell+1$ oraz wyzerowanie $\lambda^{(k)} \leftarrow 0$. Jeżeli $\ell \leqslant \gamma_1$, wykonywany jest skok do Kroku 1.

Faza 4: (Warunek powiększenia sąsiedztwa)
Jeżeli $\ell \geqslant \gamma_1$, to poszukiwanie minimum funkcji tunelowej po wystartowaniu z bliskiego sąsiedztwa określonego przez parametr ε zakończyło się niepowodzeniem. Parametr ε jest modyfikowany tak, aby eksplorowane było większe sąsiedztwo punktu startowego w celu poszukiwania minimum funkcji tunelowej. Dodatkowo zerowany jest licznik iteracji $\ell \leftarrow 0$ i następuje skok do Kroku 2.

Jeżeli dla zmodyfikowanej wartości ε nie udało się znaleźć wektora ω takiego, że $\mathcal{T}(\omega) \leqslant 0$ w $\ell \leqslant \gamma_2$ iteracjach, następuje koniec fazy tunelowej.

■

Przykład 5.2
Jako przykład rozważmy proces dyfuzji z dwoma ruchomymi źródłami poruszającymi się w obszarze $\Omega \in \mathbb{R}^2 = (0,1) \times (0,1)$. Trajektorie poruszających się źródeł są opisane przez następujące dwie funkcje:

$$s_1(x,t) = q^1 + v^1 t, \tag{5.36}$$
$$s_2(x,t) = q^2 + v^2 t, \tag{5.37}$$

gdzie: $q^1 = (0.1, 0.5)$, $v^1 = (0.8, 0.4)$, $q^2 = (0.5, 0.1)$, $v^2 = (0.4, 0.8)$

Ruchome źródła generują zanieczyszczenie powietrza z następującymi intensywnościami:

$$f_1(x,t) = 0.45 \exp(-10.0\|x - s_1(x,t)\|), \tag{5.38}$$
$$f_2(x,t) = 0.45 \exp(-10.0\|x - s_2(x,t)\|). \tag{5.39}$$

Rozprzestrzenianie się skażenia $y = y(x,t)$ w znormalizowanym horyzoncie obserwacji $T = (0,1)$ w obszarze przestrzennym Ω o brzegu $\partial\Omega$ jest opisane przez następujące równanie dyfuzji:

$$\frac{\partial y(x,t)}{\partial t} = \frac{\partial}{\partial x_1}\left(a(x)\frac{\partial y(x,t)}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(a(x)\frac{\partial y(x,t)}{\partial x_2}\right) + f(x,t), \tag{5.40}$$

z uwzględnieniem warunków brzegowych i początkowych

$$\frac{\partial y(x,t)}{\partial n} = 0 \quad \text{na} \quad \partial\Omega \times T, \tag{5.41}$$

$$y(x,0) = 0 \quad \text{w} \quad \Omega, \tag{5.42}$$

gdzie wymuszenie $f(x,t) = f_1(x,t) + f_2(x,t)$ jest superpozycją ruchomych źródeł oddziaływania obu zanieczyszczeń. Założona funkcja opisująca współczynnik dyfuzji opisana jest następująco

$$a(x) = \theta_1 + \theta_2 x_1 + \theta_3 x_2, \tag{5.43}$$

gdzie stałe parametry θ_1, θ_2 i θ_3 powinny być określone na podstawie pomiarów otrzymywanych z dwóch ruchomych czujników. Przyjęto wartości znamionowe: $\theta_1^0 = 0.01, \theta_2^0 = \theta_3^0 = 0.005$. Pomiary dokonywane przez poruszające się rzeczywiste czujniki wpływają na siebie. W rozważanym przykładzie kowariancje obserwacji były zdeterminowane przez odległość pomiędzy czujnikami wg wzoru

$$q(\eta(t,\omega^i), \eta(t,\omega^j), t) = \sigma^2 \exp\left(\frac{-\|\eta(t,\omega^i) - \eta(t,\omega^j)\|}{\beta}\right), \tag{5.44}$$

gdzie: β — współczynnik określający stopień skorelowania obserwacji. W przykładzie przyjęto $\sigma = 1.0$ oraz $\beta = 0.2$.

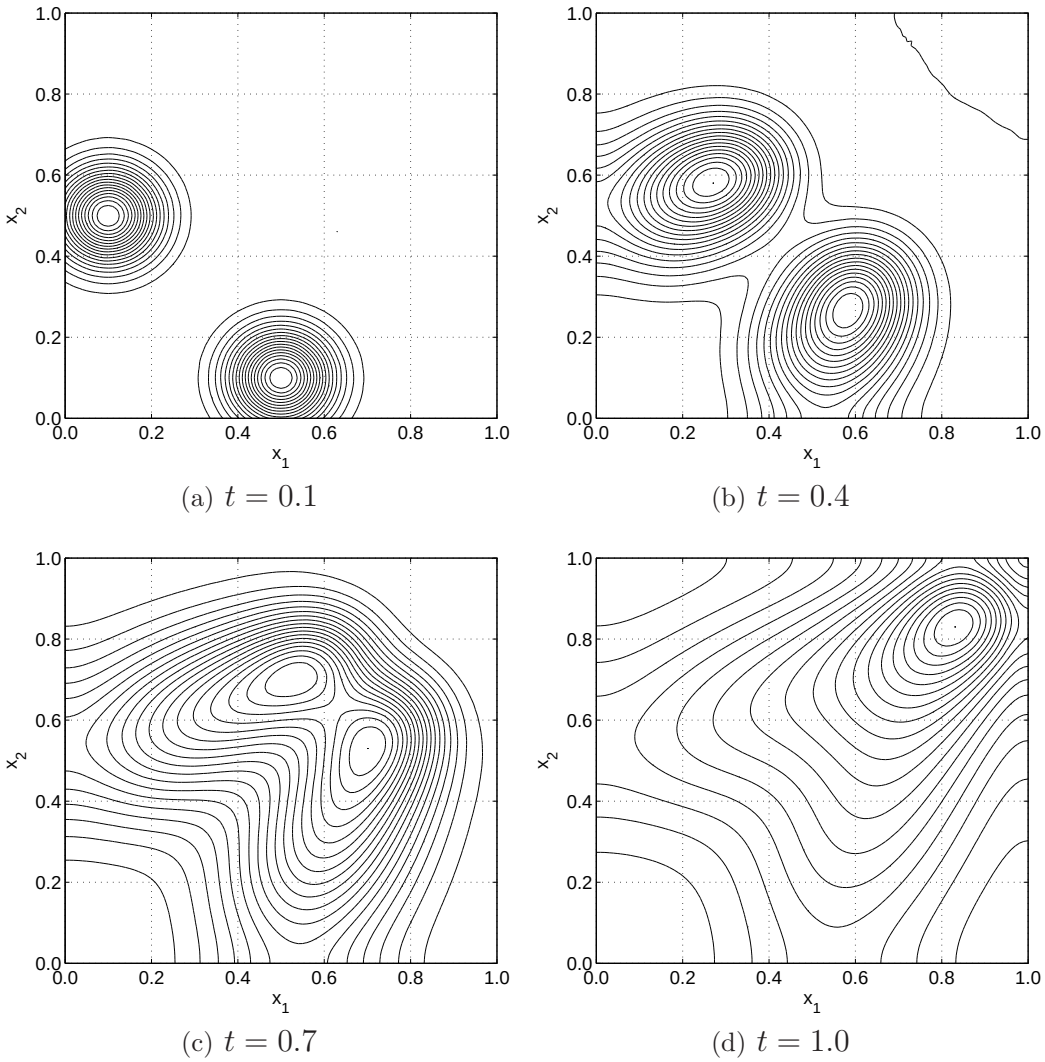
Rozwiązanie zagadnienia (5.36)–(5.44) przedstawiono na rys. 5.2, gdzie można zaobserwować efekt poruszania się dwóch źródeł zanieczyszczeń.

Do rozwiązania układu równań różniczkowych cząstkowych (5.36)–(5.43) wykorzystano aplikację COMSOL. Aplikacja rozwiązuje układ równań różniczkowych cząstkowych korzystając z metody elementu skończonego. Obszar przestrzenny Ω zdyskretyzowano siatką równomierną o rozmiarze 41×41, a horyzont czasowy podzielono na 80 podprzedziałów. Do całkowania numerycznego wymaganego do określenia macierzy informacyjnych użyto metody trapezów z krokiem czasowym $\Delta t = 1/81$.

Obliczenia przeprowadzono w oparciu o uniwersytecki klaster zbudowany w ramach projektu CLUSTERIX [248] (zob. rozdz. 3.9). Klaster składa się z węzłów zawierających 64-bitowe procesory Intel Itanium^2 1.4GHz i działa pod kontrolą środowiska GNU/Linux Debian for ia64. Program napisano w języku Fortran 95 i skompilowano z zastosowaniem kompilatora *ifort* (Intel®Fortran Compiler v.8.1 for Linux 64-bit platforms). Aplikacja używa biblioteki procedur numerycznych Intel Math Kernel Library do wykonywania operacji mnożenia macierzy oraz wektorów.

W przykładzie rozważono wyznaczenie trajektorii dwóch mobilnych węzłów sieci sensorycznej. W celu sparametryzowania trajektorii wykorzystano splajny sześcienne. Ruch każdego czujnika, w każdym z kierunków przestrzeni $\mathbb{R}^2$ opisywany jest przez kombinację liniową ośmiu funkcji bazowych (zob. rozdz. 5.2.1). Oznacza to, że minimalizowano funkcję celu o 32 zmiennych decyzyjnych.

Maksymalny stopień pierwiastka funkcji tunelowej $\lambda^{(\max)}$ ustalono jako 5.0, a krok $\Delta\lambda$ wynosił 0.05. Początkową wartość parametru $\lambda^{(k)}$ ustalano na 0.1. Maksymalne liczby punktów startowych do poszukiwania nowego tunelu γ_1, γ_2 zostały



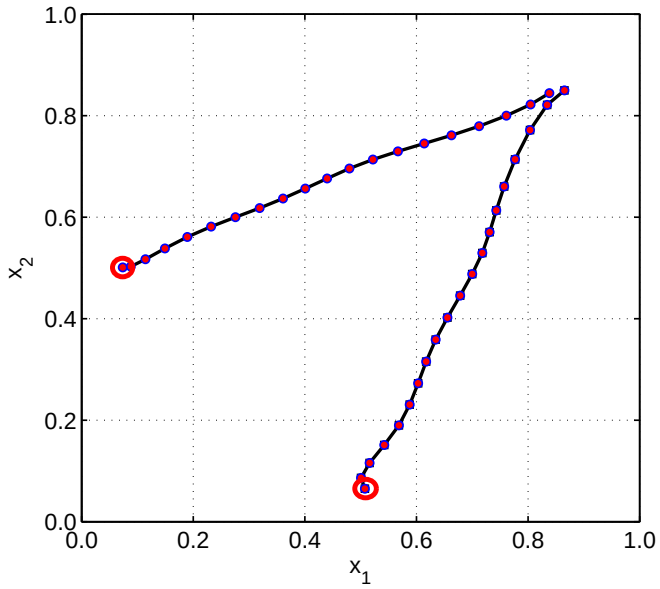
(a) $t = 0.1$ (b) $t = 0.4$

(c) $t = 0.7$ (d) $t = 1.0$

Rysunek 5.2: Koncentracja zanieczyszczeń w podanych chwilach czasowych dla Przykładu 5.2.

ustalone na 50. Parametr ε sterujący wielkością otoczenia zainicjowano wartością 0.01, a po wykonaniu γ_1 iteracji parametr ten zwiększano do wartości 0.1.

Trajektorie ruchu czujników obliczone dla kryterium D-optymalności przedstawiono na rys. 5.3, gdzie można zaobserwować jak mobilne czujniki śledzą ruchome źródła zanieczyszczeń. Pozycje każdego z czujników przedstawiono w chwilach czasowych $t_k = k\Delta t,\ \Delta t = 1/K,\ k = 1, \ldots, K$ dla $K = 20$, startując od oznaczonego miejsca. Obliczenia w przedstawionym środowisku trwały 42 minuty. Przy rozwiązywaniu zagadnienia wyznaczania optymalnych trajektorii dla dwóch mobilnych węzłów sieci sensorycznej, stanowiło to motywację do opracowania równoległej metody pozwalającej na szybsze uzyskiwanie rezultatów.



Rysunek 5.3: Wyliczona trajektoria dwóch czujników mobilnych dla Przykładu 5.2.

Interpolacja wektorów wrażliwościowych

Zastosowane metody numerycznego rozwiązywania równań różniczkowych cząstkowych dostarczają wartości wektora wrażliwości g (zob. (5.4)) jedynie w węzłach dyskretnej siatki czasoprzestrzennej. Planowanie trajektorii czujników ruchomych generuje jednak potrzebę dokonania interpolacji wektorów wrażliwości również w punktach nie będących węzłami wspomnianej siatki.

Interpolacja brakujących wartości względem czasu jest dokonywana przy pomocy klasycznej interpolacji liniowej [178], a interpolacja zmiennych przestrzennych jest realizowana za pomocą interpolacji biliniowej [178]. Sposób wyliczania wartości punktów pośrednich przedstawiono w Dodatkach A.1 oraz A.2.

♦

5.2.7 Równoległa wersja algorytmu

Istnieje stosunkowo niewiele publikacji poruszających zagadnienie zrównoleglania algorytmu tunelowego [74, 249]. Rozwiązania te mogą zostać adaptowane do rozważanych zagadnień optymalnej obserwacji układów z czasoprzestrzenną dynamiką, jednak wymagają pewnych modyfikacji oraz dostosowania ich charakterystyk do specyfiki problemu rozważanego w niniejszej pracy [252].

Aby zrównoleglić przedstawioną w rozdz. 5.2.6 metodę wyznaczania optymalnych trajektorii czujników ruchomych opartą na parametryzacji trajektorii, należało

wybrać miejsce w algorytmie, które pozwala na uzyskanie maksymalnego przyspieszenia, nie powodując powstania nadmiernego narzutu związanego z komunikacją. Analiza opracowanej metody sekwencyjnej oraz podejścia przedstawionego w pracy [74] pozwoliło na zaproponowanie metody dekompozycji algorytmu względem przestrzeni poszukiwań nowego tunelu w fazie tunelowej z uwzględnieniem ziarnistości obliczeń na poziomie określania minimum lokalnego kryterium optymalności.

Opracowane podejście jest w zasadzie zbliżone do podejścia zaproponowanego w pracy [74]. Zrównoleglenie fazy tunelowej jest bardzo naturalne i pozwala na uzyskanie wysokiej efektywności algorytmu. W przypadku zrównoleglenia fazy minimalizacji, procesory w klastrze musiałyby wymieniać znaczne ilości informacji, co przyczyniłoby się do spowolnienia obliczeń z powodu opóźnień podczas transmisji danych.

W przedstawionych badaniach wykorzystano schemat komunikacji „nadzorca-podwładni" (zob. rys. 4.7), w którym jeden z procesorów pełni rolę nadzorcy nad działaniem całego algorytmu, a resztę procesorów wykorzystuje się do wykonywania obliczeń. Takie podejście jest bardzo podobne do metody zaproponowanej w [74].

Metoda zrównoleglenia

Jak to już omówiono w rozdz. 5.2.4 oraz 5.2.6, punkt startowy $\bar{\omega}$ wyznacza się w otoczeniu wyznaczonego wcześniej minimum lokalnego $\omega^{(k*)}$ zgodnie z (5.34). Procedurę generowania nowych punktów startowych w określonym otoczeniu minimum lokalnego $\omega^{(k*)}$ powtarza się przez określoną arbitralnie liczbę iteracji.

Zaproponowana metoda zrównoleglenia zakłada, że każdy z procesorów liczących będzie minimalizować funkcję tunelową startując z innego przybliżenia początkowego, co w konsekwencji powinno zwiększyć szanse odkrycia korzystniejszego rozwiązania. Punkt startowy w otoczeniu minimum lokalnego określa się identycznie jak w algorytmie sekwencyjnym, na podstawie aktualnej wartości minimum lokalnego, wartości określającej wielkość sąsiedztwa definiującego wymagane otoczenie oraz wartości losowej zgodnie z procedurą (5.33).

Każdy procesor wykonujący obliczenia minimalizuje więc funkcję tunelową generując własną ścieżkę poszukiwań. Aby było to jednak możliwe, zanim algorytm tunelowy rozpocznie działanie, procesor główny losuje ziarno dla generatora liczb losowych każdego z procesorów liczących. Każdy procesor liczący wykorzystuje wstępnie swój generator liczb losowych zainicjowany innym ziarnem i dzięki temu następuje wybór różnych losowych punktów startowych procedury minimalizacji funkcji tunelowej.

Metodę zrównoleglania przedstawionego algorytmu zaprojektowano zgodnie z poniższymi założeniami:

- Istnieje procesor główny, który kontroluje działanie algorytmu oraz rozsyła potrzebne informacje do procesorów liczących.

- Każdy z procesorów liczących wykonuje zarówno fazę tunelową, jak i fazę minimalizacji.
- Każdy z procesorów liczących posługuje się generatorem liczb losowych zainicjowanym innym ziarnem.
- Minimalizacja funkcji tunelowej w fazie tunelowej startuje z różnych punktów początkowych na każdym z procesorów liczących.
- Po uruchomieniu algorytmu, każdy z procesorów liczących przechodzi do fazy minimalizacji w celu znalezienia startowego minimum lokalnego.
- Kiedy procesor liczący znajdzie tunel do innej doliny (faza tunelowa zakończy się sukcesem), przechodzi do fazy minimalizacji lokalnej.
- Po znalezieniu minimum lokalnego, procesor liczący przesyła natychmiast wynik do procesora głównego.
- Procesor główny sprawdza czy nowe optimum jest lepsze od dotychczas znalezionego. Jeżeli tak, to przesyła je do wszystkich procesorów liczących. Dodatkowo, procesor główny zeruje liczniki wykonanych iteracji poszczególnych procesorów.
- Każdy procesor liczący sprawdza, obecność wiadomości od procesora głównego tylko podczas wykonywania fazy tunelowej. W przypadku wykonywania fazy minimalizacji lokalnej, procesor wykonuje obliczenia do momentu znalezienia minimum lokalnego bez sprawdzania nowych informacji od procesora głównego.
- Gdy procesor liczący otrzyma wiadomość o nowym minimum globalnym, przerywa aktualnie wykonywanie obliczenia (faza tunelowa) i uruchamia ponownie algorytm startując z otoczenia nowego wyznaczonego minimum.
- Procesor główny kontroluje liczbę wykonanych iteracji dla każdego procesora liczącego i przesyła odpowiedni wektor ε.
- Procesor główny sprawdza warunki zatrzymania algorytmu i przesyła do procesorów liczących wiadomość o zakończeniu obliczeń gdy zostaną spełnione.

Schemat działania procesora głównego przedstawiono jako Algorytm 5.3. Głównym zadaniem procesora nadzorcy jest kontrolowanie działania całego algorytmu, a także przechowywanie oraz wymiana danych z procesorami liczącymi. Komunikację pomiędzy procesorami liczącymi a procesorem głównym realizuje się na podstawie kodów operacji przesyłanych pomiędzy procesorami. Przez większą część swojego czasu procesor główny oczekuje na żądania pojawiające się ze strony procesorów

liczących. Na podstawie odebranego żądania następuje uaktualnienie danych opisujących stan procesorów liczących, a wymagane dane są przesłanie do procesorów liczących.

Procesor główny przechowuję liczby iteracji wykonanych przez procesory liczące w tablicy T_γ. Celem aktualizacji tych informacji jest zmiana wartości parametru ε dla procesora, który przekroczył liczbę różnych punktów startowych dla procedury minimalizacji funkcji tunelowej. Po sprawdzeniu zadanej liczby punktów startowych zarówno dla bliskiego (parametr γ_1) jak i rozszerzonego otoczenia (parametr γ_2), procesor liczący jest ustawiany w tryb oczekiwania. Celem takiego zachowania algorytmu jest oczekiwanie na zakończenie obliczeń przez pozostałe procesory. Jeżeli żaden z nich nie wyznaczy lepszego minimum globalnego, następuje zakończenie obliczeń. Jednak, gdy którykolwiek z nich wyznaczy nowe, lepsze minimum lokalne, następuje wysłanie rozkazu o ponowienie obliczeń do procesorów liczących będących w stanie wstrzymania, z jednoczesnym wyzerowaniem licznika wykonanych iteracji przez te procesory. Dodatkowo następuje modyfikacja parametru ε, tak aby procesory znów rozpoczynały poszukiwanie punktów startowych procedury minimalizacji funkcji tunelowej od najbliższego sąsiedztwa aktualnego minimum lokalnego.

Algorytm 5.3 Schemat działania — procesor główny/nadzorca

Dane wejściowe:

$\omega^{(k*)}$ — aktualne minimum lokalne funkcji celu ϕ,
γ_1 — maksymalna liczba punktów startowych do poszukiwania nowego tunelu w sąsiedztwie aktualnego minimum $\omega^{(k*)}$,
γ_2 — maksymalna liczba punktów startowych do poszukiwania nowego tunelu w całym obszarze dopuszczalnym,
ε — wielkość otoczenia w jakim ma być generowany nowy punkt startowy dla procesu minimalizacja funkcji tunelowej.

Dane wyjściowe:

ϕ_{opt} — przybliżenie optymalnej wartości kryterium optymalności,
ω_{opt} — wyznaczony optymalny plan eksperymentu.

Krok 0: (Inicjalizacja)

Wylosowanie i wysłanie ziaren generatorów liczb pseudolosowych do procesorów liczących. Ustawienie optymalnego planu eksperymentu na wartość początkową $\omega_{\text{opt}} \leftarrow \omega^{(0)}$ oraz wyznaczenie wartości kryterium: $\phi_{\text{opt}} \leftarrow \phi(\omega^{(0)})$. Zerowanie tablicy przechowującej liczbę wykonanych losowań nowych punktów startowych minimalizacji funkcji tunelowej $T_\gamma = 0$. Przesłanie każdemu procesorowi parametru ε określającego losowanie punktu startowego dla procesu minimalizacji funkcji tunelowej w najbliższym sąsiedztwie punktu $\omega^{(k*)}$.

Krok 1: (Oczekiwanie na żądanie procesora liczącego)

Oczekiwanie w pętli na żądania od procesorów liczących.

Krok 2: (Odebranie żądania)
Odebranie kodu operacji od procesora liczącego p.

Krok 3: (Wykonanie procedury obsługi żądania)
Reakcja procesora głównego na wysłany kod operacji:

Operacja 1: (Wyślij aktualne minimum globalne)
Procesor główny wysyła do procesora p aktualne przybliżenie minimum globalnego.

Operacja 2: (Odbierz przybliżenie minimum globalnego)
Procesor główny odbiera wyznaczone przez procesor liczący przybliżenie minimum globalnego ϕ_p oraz odpowiadający mu wektor ω_p.
Jeżeli $\phi_p < \phi_{\text{opt}}$ to $\phi_{\text{opt}} \leftarrow \phi_p$ oraz $\omega_{\text{opt}} \leftarrow \omega_p$. Dodatkowo następuje wysłanie do wszystkich procesorów liczących, oprócz procesora p, informacji o wyznaczeniu nowego przybliżenia minimum globalnego oraz wyzerowanie tablicy wykonanych iteracji $\mathrm{T}_\gamma = 0$. Procesory będące w fazie wstrzymania obliczeń po wykorzystaniu wszystkich iteracji wznawiają obliczenia.

Operacja 3: (Pozwól na wykonywanie kolejnych obliczeń)

Faza 1: (Jeżeli procesor eksploruje najbliższe otoczenie punktu $\omega^{(k*)}$)
Jeżeli procesor p poszukuje minimum funkcji tunelowej w najbliższym sąsiedztwie i $\mathrm{T}_\gamma(p) < \gamma_1$, następuje wysłanie pozwolenia na kontynuowanie obliczeń oraz następuje zwiększenie licznika wykonanych iteracji przez procesor liczący $\mathrm{T}_\gamma(p) \leftarrow \mathrm{T}_\gamma(p) + 1$.
Jeżeli liczba iteracji $\mathrm{T}_\gamma(p) \geqslant \gamma_1$ to następuje wysłanie pozwolenia na kontynuowanie obliczeń. Dodatkowo zwiększane jest sąsiedztwo ε związane z danym procesorem p w celu eksploracji większego otoczenia w poszukiwaniu minimum funkcji tunelowej oraz następuje wyzerowanie liczby wykonanych iteracji $\mathrm{T}_\gamma(p) = 0$.

Faza 2: (Jeżeli procesor eksploruje rozszerzone otoczenie punktu $\omega^{(k*)}$)
Jeżeli procesor p poszukuje minimum funkcji tunelowej w rozszerzonym sąsiedztwie i $\mathrm{T}_\gamma(p) < \gamma_2$, następuje wysłanie pozwolenia na kontynuowanie obliczeń oraz następuje zwiększenie licznika wykonanych iteracji przez procesor liczący $\mathrm{T}_\gamma(p) \leftarrow \mathrm{T}_\gamma(p) + 1$.
Jeżeli liczba iteracji $\mathrm{T}_\gamma(p) \geqslant \gamma_2$, następuje wysłanie kodu wstrzymania wykonywania obliczeń przez procesor p.

Operacja 4: (Wyślij wielkość otoczenia)
Procesor główny wysyła do procesora liczącego wielkość otoczenia ε w jakim ma generować nowy punkt startowy dla procedury minimalizacji funkcji tunelowej.

Krok 4: (Sprawdzenie warunku stopu)
Jeżeli nie wszystkie procesory wykonały maksymalną liczbę przydzielonych iteracji w celu wygenerowania nowych punktów startowych w procesie minimalizacji funkcji tunelowej $\mathcal{T}$, następuje skok do Kroku 1.

Krok 5: (Zakończenie obliczeń)
Wysłanie do procesorów liczących żądania zakończenia obliczeń.

■

Schemat działania procesora liczącego przedstawiono jako Algorytm 5.4. Procesory liczące wykonują obliczenia na podstawie danych otrzymanych od procesora-nadzorcy. Głównym zadaniem procesorów liczących jest wykonywanie minimalizacji funkcji tunelowej startując z różnych punktów początkowych.

Wielkość otoczenia, w jakim generowane są punkty startowe dla procedury minimalizacji funkcji tunelowej, jest przesyłana przez procesor główny. Wielkość otoczenia zależy od liczby wykonanych do tej pory iteracji, podczas których nie zostało wyznaczone nowe minimum lokalne. Gdy procesor liczący nie znajdzie nowego minimum funkcji tunelowej, wysyła zapytanie do procesora nadzorcy o możliwość kontynuacji obliczeń. Jeżeli procesor liczący sprawdził wyznaczoną liczbę punktów startowych dla procedury minimalizacji funkcji tunelowej, następuje zwiększenie sąsiedztwa w jakim generowane są punkty startowe.

Po przekroczeniu liczby różnych punktów startowych dla rozszerzonego sąsiedztwa, procesor główny wstrzymuje pracę danego procesora liczącego. W tym momencie rozpoczyna się oczekiwanie na wyniki badań przesłane przez pozostałe procesory. Jeżeli którykolwiek z nich wyznaczy nowe minimum lokalne, następuje wznowienie obliczeń, startując od nowo wyznaczonego punktu minimum. Jeżeli jednak żaden z procesorów nie określi lepszego minimum lokalnego, następuje zakończenie obliczeń.

Algorytm 5.4 Schemat działania — procesor liczący

Krok 0: (Inicjalizacja)
Odebranie ziarna generatora liczb losowych oraz jego inicjalizacja.

Krok 1: (Odebranie danych od procesora-nadzorcy)
Wysłanie żądania przesłania aktualnego minimum globalnego ϕ_{opt} oraz parametru ε określającego wielkość otoczenia w jakim generowany będzie punkt startowy procedury minimalizacji funkcji tunelowej do procesora nadzorcy.

Krok 2: (Obliczenia)
Minimalizacja funkcji tunelowej czyli poszukiwanie punktu $\hat{\omega}$ takiego, że $\mathcal{T}(\hat{\omega}) \leqslant 0$.

Warunek 1: (Szukanie tunelu zakończone sukcesem)
Jeżeli znaleziony został punkt $\hat{\omega}$ spełniający $\mathcal{T}(\hat{\omega}) \leqslant 0$, dokonuje się minimalizacji lokalnej, której celem jest wyznaczenie punktu ω_p takiego, że $\phi(\omega_p) \leqslant \phi_{\text{opt}}$. Wyznaczone minimum lokalne ϕ_p oraz wektor ω_p przesyłane są do procesora głównego i następuje skok do Kroku 1.

Warunek 2: (Szukanie tunelu zakończone niepowodzeniem)
Jeżeli poszukiwanie punktu $\hat{\omega}$ spełniającego $\mathcal{T}(\hat{\omega}) \leqslant 0$ zakończyło się niepowodzeniem, do procesora głównego wysyła się żądanie przesłania aktualnego przybliżenia minimum globalnego. Jeżeli odebrana wartość ϕ_{opt} jest mniejsza niż aktualnie przechowywana przez procesor liczący, następuje skok do Kroku 1. W przeciwnym przypadku do procesora nadzorcy wysyłane jest zapytanie o kontynuację obliczeń.

Rozkaz 1: (Czekaj)
Jeżeli procesor liczący otrzymuje rozkaz wstrzymania obliczeń, następuje oczekiwanie na kolejny rozkaz. Jeżeli kolejnym rozkazem jest operacja zakończenia obliczeń, następuje skok do Kroku 3. W przeciwnym razie następuje skok do Kroku 1.

Rozkaz 2: (Kontynuuj)
Jeżeli procesor liczący otrzymuje pozwolenie na kontynuowanie obliczeń, następuje wysłanie żądania o przesłanie zaktualizowanej wartości parametru ε, a następnie wykonywany jest skok na początek Kroku 2.

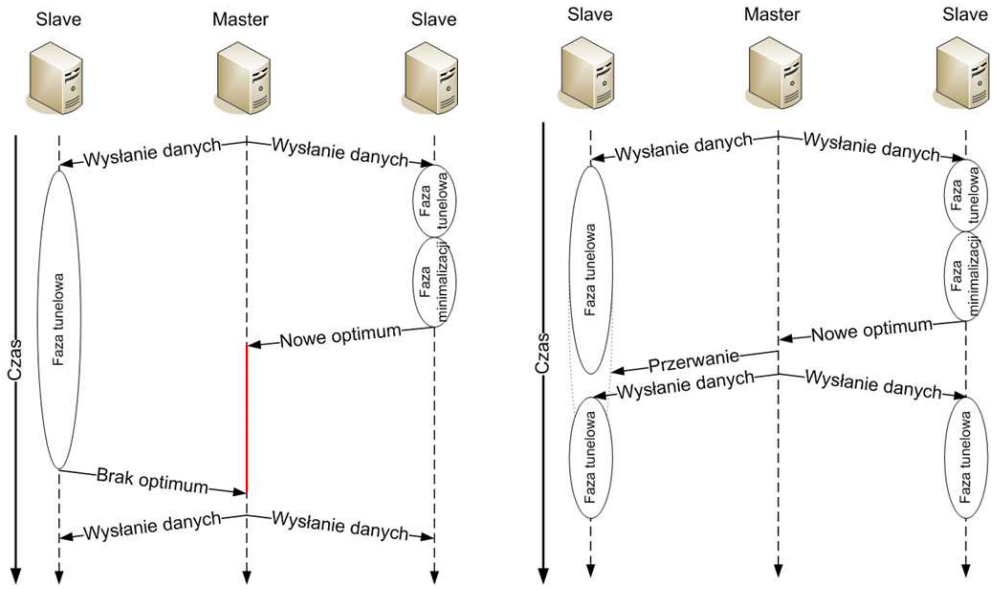
Krok 3: (Zakończenie obliczeń)
Procesor liczący kończy pracę.

■

Modyfikacje schematu

W przykładach przedstawionych w [74, 126] założono, że czas wyznaczenia wartości funkcji celu $\phi(\omega)$ jest stosunkowo krótki. W takim przypadku czas zużyty na obliczanie wartości funkcji ϕ nie wpływa znacząco na szybkość algorytmu tunelowego. Gdy jeden z procesorów liczących znajduje nowe minimum lokalne, wysyła je do procesora głównego, który rozsyła je do pozostałych procesorów liczących. Jeżeli czas potrzebny na wyznaczenia $\phi(\omega)$ jest krótki, czas potrzebny na minimalizację funkcji tunelowej (5.18) jest najczęściej również krótki, więc procesor liczący może stosunkowo często sprawdzać czy przybyły nowe informacje od procesora głównego. Ta sytuacja oznacza, że równoległy algorytm tunelowy jest bardzo elastyczny i może szybko reagować na kolejne wyznaczane minima lokalne.

W przypadku rozważania problemu wyznaczania optymalnych trajektorii mobilnych węzłów sieci sensorycznej dla obiektów z czasoprzestrzenną dynamiką oraz



Rysunek 5.4: Schemat wymiany informacji pomiędzy procesorem głównym a procesorami liczącymi: (a) schemat oryginalny, (b) schemat zmodyfikowany.

skorelowanych obserwacji należy zaznaczyć, że czas na wyznaczenie wartości funkcji $\phi(\omega)$ może być stosunkowo długi (zwłaszcza w przypadku dużej liczby B-splajnów sześciennych oraz dużej liczby węzłów mobilnych). Zwróćmy uwagę, że w przypadku procesów o złożonej dynamice czasoprzestrzennej, np. procesów opisywanych równaniami adwekcji-dyfuzji, samo wyznaczenie macierzy informacyjnej jest skomplikowane i czasochłonne. Gdy metoda optymalizacji wyznacza wartość tej funkcji stosunkowo często (kilka tysięcy razy), czas reakcji algorytmu na nowo znalezione minimum lokalne przez procesor liczący jest dość długi (zob. rys. 5.4(a)).

Aby zapobiec tego typu sytuacjom, zaimplementowano sprawdzanie informacji od procesora głównego już w metodzie optymalizacji lokalnej (w rozważanym algorytmie jest to metoda optymalizacji Rosenbrocka; zob. Dodatek A.4). Ta modyfikacja pozwala na ominięcie problemu długiej zwłoki w przypadku, gdy jeden z procesorów liczących znajdzie lepsze minimum lokalne (zob. rys. 5.4(b)). Przerwanie obliczeń następuje wyłącznie wtedy, gdy procesor liczący przeprowadza minimalizację funkcji tunelowej. W przypadku gdy procesor liczący jest w fazie minimalizacji lokalnej, obliczenia kontynuuje się ze względu na potencjalną możliwość wyznaczenia nowego minimum lokalnego, lepszego niż dotychczas wyznaczone.

5.3 Planowanie optymalnych trajektorii czujników ruchomych z uwzględnieniem ich dynamiki

Parametryczny opis trajektorii może okazać się dość mocnym uproszczeniem rozważanych problemów. Opisywanie trajektorii w sposób nie uwzględniający fizycznych właściwości (np. inercji, ograniczeń prędkościowych i energetycznych itp.) zarówno samych czujników, jak i urządzeń je przenoszących może powodować, że uzyskane rezultaty będą mało użyteczne w praktyce.

W celu uwzględnienia dynamiki poruszających się czujników, problem poszukiwania optymalnych trajektorii ruchu czujników można rozważyć jako problem sterowania optymalnego. Takie podejście zaproponowano w pracach [111, 224, 225, 226, 227, 228, 231, 233].

5.3.1 Opis dynamiki węzłów sieci sensorycznej

Równania ruchu czujników

W celu sformułowania problemu znajdowania optymalnych trajektorii czujników ruchomych z uwzględnieniem ich kinematyki i dynamiki, zdefiniujmy

$$s(t) = (x^1(t), x^2(t), \ldots, x^n(t)), \quad \forall t \in T = [0, t_f] \tag{5.45}$$

jako wektor położeń n czujników w czasie t. Załóżmy, że czujniki są przenoszone przez pojazdy wewnątrz obszaru dopuszczalnego $X \in \Omega$, a równanie ruchu można opisać w następującej ogólnej formie:

$$\dot{s}(t) = f(s(t), u(t)), \quad t \in T, \quad s(0) = s_0, \tag{5.46}$$

gdzie: $f : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ — funkcja różniczkowalna w sposób ciągły, $s_0 \in \mathbb{R}^n$ — wektor początkowych położeń czujników, $u : T \to \mathbb{R}^r$ — poszukiwane sterowanie spełniające ograniczenia

$$u_{\min} \leqslant u(t) \leqslant u_{\max}, \quad t \in T. \tag{5.47}$$

dla pewnych zadanych wektorów $u_{\min}$ oraz $u_{\max}$. Ograniczenie sterowań jest konieczne ze względów technicznych i/lub ekonomicznych.

Dla danych położeń początkowych s_0 i istotnie ograniczonego sterowania u, istnieje jednoznacznie określona, absolutnie ciągła funkcja $s : T \to \mathbb{R}^n$ spełniająca równanie (5.46), nazywana trajektorią powiązaną z s_0 oraz u. W literaturze [228] rozważa się kilka równań stanu (5.46):

- liniowe równanie pierwszego rzędu:

$$\dot{s}(t) = C(t)s(t) + D(t)u(t), \quad s(0) = s_0, \tag{5.48}$$

  gdzie: C oraz D — ciągłe funkcje o wartościach macierzowych.

- liniowe równanie drugiego rzędu:

$$E\ddot{s}(t) + F\dot{s}(t) = Gu(t), \quad \dot{s}(0) = 0, \quad s(0) = s_0, \tag{5.49}$$

gdzie: E i F — macierze diagonalne, $E \succ 0, F \succ 0$.

- przypadek, gdy nie uwzględnia się dynamiki pojazdów przenoszących czujniki, a skupia na samych trajektoriach ($r = n$):

$$\dot{s}(t) = u(t), \quad s(0) = s_0, \tag{5.50}$$

Ograniczenia na trajektorie ruchu czujników

Aby wyznaczone trajektorie mogły być traktowane jako realizowalne, często należy uwzględnić ograniczenia narzucone przez konkretne aplikacje. Jednym z ograniczeń powinno być zachowanie minimalnej bezpiecznej odległości pomiędzy czujnikami w celu uniknięcia problemów związanych ze skupianiem się czujników lub wręcz zapewnienia bezkolizyjności. Równanie uwzględniające takie ograniczenia może mieć następującą formę:

$$\beta_{ij}(s(t)) = R^2 - \|x^i(t) - x^j(t)\|^2 \leqslant 0, \quad \forall t \in T, \tag{5.51}$$

gdzie: $1 \leqslant i < j \leqslant n$, R — minimalna dopuszczalna odległość, gwarantująca niezależność pomiarów wykonywanych przez czujniki.

W skróconej postaci, ograniczenia (5.51) można zapisać w następującej formie:

$$\gamma_\ell(s(t)) \leqslant 0, \quad \forall t \in T, \tag{5.52}$$

gdzie: $\ell = 1, \ldots, (n-1)n/2$.

Szersze omówienie ograniczeń dotyczących trajektorii ruchu czujników znajduje się w pracy [228].

5.3.2 Sformułowanie problemu

Zdefiniowanie problemu optymalnej obserwacji

W rozważanym kontekście, problem optymalnej obserwacji można zdefiniować jako poszukiwanie takich sterowań, które spowodują ruch pojazdów przenoszących czujniki w sposób gwarantujący osiągnięcie ekstremum założonego kryterium optymalności $\Psi[\mathcal{M}(s)]$ zdefiniowanego na macierzy informacyjnej:

$$\mathcal{M}(s) = \sum_{i=1}^{n} \int_0^{t_f} g(x^i(t), t) g^{\mathsf{T}}(x^i(t), t)\, \mathrm{d}t. \tag{5.53}$$

gdzie: g — wektor wrażliwości wyznaczany zgodnie z (2.15). Trajektoria s jest jednoznacznie określana przez sterowanie u oraz położenie początkowe s_0, co implikuje,

że problem określenia optymalnego sterowania można sformułować jako problem optymalizacji zdefiniowany na zbiorze par dopuszczalnych

$$\mathcal{P} = \{(s_0, u) : s_0 \in X^n, u : T \to \mathbb{R}^r \text{jest mierzalne}, u_{\min} \leqslant u(t) \leqslant u_{\max},\ t \in T\}. \tag{5.54}$$

W konsekwencji, rozważany problem optymalnej obserwacji można zapisać następująco:

$$\min_{(s_0,u)\in\mathcal{P}} J(s_0, u) \tag{5.55}$$

przy ograniczeniach nierównościowych

$$h(s_0, u) \leqslant 0, \tag{5.56}$$

gdzie:

$$J(s_0, u) = \Psi[M(s)], \tag{5.57}$$

$$h(s_0, u) = \max_{(\ell,t)\in\bar{\nu}\times T} \{\gamma_\ell(s(t))\}. \tag{5.58}$$

$\bar{\nu}$ — zbiór indeksów $\{1, \ldots, \nu\}, \nu = (n-1)n/2$.

Powyższy problem jest silnie nieliniowy i z reguły niemożliwym jest znalezienie jego rozwiązania analitycznie. Implikuje to konieczność wykorzystania technik numerycznych. Bardzo praktyczne podejście zostało przedstawione w [228], gdzie problem optymalizacyjny sprowadzono do formy kanonicznej znanej jako klasyczny problem Mayera. Taka transformacja otwiera możliwość zastosowania szeregu znanych metod numerycznych algorytmicznego sterowania optymalnego, umożliwiających rozwiązanie rozważanego problemu.

Przekształcenie w problem Mayera

Rozwiązanie problemu optymalnego planowania trajektorii ruchu mobilnych węzłów sieci sensorycznej jako problemu sterowania optymalnego z zastosowaniem technik numerycznych, np. iteracyjnego programowania dynamicznego opisanego dalej, wymaga przekształcenia problemu (5.55)–(5.58) do postaci klasycznego problemu Mayera. Oznacza to, że kryterium optymalności ma być wyznaczone wyłącznie przez pewną funkcję określoną na osiągniętym końcowym stanie układu.

W celu przekształcenia zadania do postaci Mayera, zdefiniujmy następujące wielkości:

$$\chi_{ij}(s(t), t) = \sum_{i=1}^{n} g_i(x^i(t), t) g_j(x^i(t), t) \tag{5.59}$$

a następnie definiujemy macierz $\Pi(t) = \{\varpi_{ij}(t)\} \in \mathbb{R}^{m\times m}$ gdzie

$$\varpi_{ij}(t) = \int_0^t \chi_{ij}(s(\tau), \tau)\, \mathrm{d}\tau, \quad 1 \leqslant i, j \leqslant m. \tag{5.60}$$

Z definicji macierzy informacyjnej (5.53) oraz (5.59) i (5.60) wynika równość

$$\mathcal{M}(s) = \Pi(t_f) \tag{5.61}$$

oraz zależność

$$\dot{\varpi}_{ij}(t) = \chi_{ij}(s(t), t), \quad \forall t \in T. \tag{5.62}$$

Wprowadzenie nowego wektora stanu v, jego wartości początkowej v_0 oraz określonego na nim odwzorowania F postaci

$$v(t) = \begin{pmatrix} s(t) \\ \varpi_{11}(t) \\ \vdots \\ \varpi_{m1}(t) \\ \varpi_{12}(t) \\ \vdots \\ \varpi_{mm}(t) \end{pmatrix}, \quad v_0 = \begin{pmatrix} s(0) \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad F(v(t), u(t), t) = \begin{pmatrix} f(s(t), u(t)) \\ \chi_{11}(s(t), t) \\ \vdots \\ \chi_{m1}(s(t), t) \\ \chi_{12}(s(t), t) \\ \vdots \\ \chi_{mm}(s(t), t) \end{pmatrix} \tag{5.63}$$

pozwala na zdefiniowanie nowego, rozszerzonego równania stanu

$$\dot{v}(t) = F(v(t), u(t), t), \quad t \in T, \quad v(0) = v_0. \tag{5.64}$$

Powyższe podstawienia pozwalają na równoważne sformułowanie rozważanego problemu sterowania optymalnego w postaci

$$\min_{(v_0,u)\in\bar{\mathcal{P}}} \bar{J}(v_0, u) \tag{5.65}$$

przy spełnieniu ograniczenia nierównościowego

$$\bar{h}(v_0, u) \leqslant 0, \tag{5.66}$$

gdzie:

$$\bar{\mathcal{P}} = \left\{ (v_0, u) : v_0 = (v_{01}, \ldots, v_{0n}, \underbrace{0, \ldots, 0}_{m^2\text{razy}}), (v_{01}, \ldots, v_{0n}, u) \in \mathcal{P} \right\} \tag{5.67}$$

$$\bar{J}(v_0, u) = G(v(t_f)) = \Psi[\Pi(t_f)], \tag{5.68}$$

$$\bar{h}(v_0, u) = h(v_{01}, \ldots, v_{0n}, u). \tag{5.69}$$

Przedstawienie problemu poszukiwania optymalnego sterowania w kategoriach problemu Mayera ze wskaźnikiem jakości $\bar{J}$ pozwala na wykorzystanie znanych technik algorytmicznego sterowania optymalnego przy ograniczeniach nierównościowych.

5.3.3 Proponowane rozwiązanie

Rozwiązywanie problemu sterowania optymalnego nie jest zadaniem trywialnym. W większości przypadków uzyskanie rozwiązania analitycznego jest praktycznie niemożliwe. Określenie sterowań optymalnych wymaga zazwyczaj zaadaptowania skomplikowanych metod algorytmicznych i numerycznych [23, 43, 181]. Jedną z najogólniejszych technik tego typu jest programowanie dynamiczne [17, 176]. Pozwala ono

na uzyskanie sterowania optymalnego, jednakże okupione jest to bardzo znaczącym nakładem obliczeń. Programowanie dynamiczne wykorzystuje się przy rozwiązywaniu wielu problemów, jednak jego największa wada — tzw. przekleństwo wymiarowości polegające na wykładniczym wzroście nakładu obliczeniowego przy próbie stosowania dokładnej dyskretyzacji problemu ciągłego — powoduje, że jest ono niemożliwe do zastosowania w przypadku wielu problemów praktycznych. Podobne problemy napotyka się podczas próby wykorzystania tej techniki do bezpośredniego rozwiązania zadania określenia optymalnych sterowań deterministycznych ruchu mobilnych węzłów sieci sensorycznej, sformułowanego w rozdz. 5.3.2. Stanowiło to motywację do wykorzystania algorytmu iteracyjnego programowania dynamicznego zaproponowanego przez Luusa [133] jako narzędzia mającego w zamiarze autora zachować największe zalety oryginału (uniwersalność oraz prostotę sformułowania i implementacji) przy jednoczesnym ograniczeniu wpływu najbardziej dokuczliwych wad (wymagana czasowe i pamięciowe nieakceptowalne w praktyce). Redukcję czasu obliczeń osiąga się poprzez stosowanie siatek dyskretyzacji przestrzeni sterowań i stanów, które wprawdzie składają się ze stosunkowo niewielkiej liczby węzłów, ale są adaptowane w miarę kolejnych iteracji w taki sposób, że środek siatki odpowiada rozwiązywaniu otrzymanemu w poprzedniej iteracji, a obszar pokrywany przez daną siatkę ulega stopniowemu zawężeniu. Z kolei eksplozji wymagań pamięciowych zapobiega się generując losowo niewielką liczbę przebiegów sterowań, które determinują dyskretną siatkę węzłów w przestrzeni stanów. W literaturze podaje się rezultaty badań, w których metoda iteracyjnego programowania dynamicznego zastosowano z powodzeniem do rozwiązania problemów o 250 zmiennych stanu i 250 składowych wektora sterowań ([133], rozdz. 5).

Dyskretyzacja problemu

W rozdziale 5.3.2 dokonano transformacji problemu optymalnej obserwacji z zastosowaniem sieci sensorów mobilnych do problemu sterowania optymalnego w postaci Mayera. Postać ta stanowi punkt wyjścia dla poniższych rozważań (dlatego też zachowano wprowadzone tam oznaczenia). W celu skupienia uwagi załóżmy więc, że rozważamy ciągły układ dynamiczny jest opisany układem równań różniczkowych postaci

$$\frac{\mathrm{d}v}{\mathrm{d}t}(t) = F(v(t), u(t), t), \quad v(0) = v_0, \tag{5.70}$$

gdzie: $v(t) \in \mathbb{R}^q$ — wektor stanu, $u(t) \in \mathbb{R}^r$ — wektor sterowań, na który nałożono ograniczenia

$$u_{\min} \leqslant u(t) \leqslant u_{\max}. \tag{5.71}$$

Dla uproszczenia zakłada się, że wektor v_0 nie jest składową nieznaną i jest ustalany arbitralnie przed uruchomieniem algorytmu. Rozważmy problem optymalnego sterowania sprowadzający się do minimalizacji funkcjonału

$$J(v(0), u) = G(v(t_f)) + \int_0^{t_f} C(v(t), u(t), t)\,\mathrm{d}t. \tag{5.72}$$

Zauważmy, że w celu zwiększenia ogólności rozważań, powyższe kryterium zawiera dodatkowy składnik całkowy. Podcałkowa funkcja kosztu $C : \mathbb{R}^{q+r+1} \to \mathbb{R}$ może stanowić np. miarę chwilowej mocy zużywanej przez pojazdy do poruszania się. Problem optymalnego sterowania zdefiniowany jest jako problem znalezienia takiego sterowania u, aby wskaźnik jakości J osiągał wartość minimalną.

Wprawdzie programowanie dynamiczne można bezpośrednio stosować do układów dynamicznych czasowo-ciągłych, jednak zasadniczą komplikacją staje się wówczas konieczność rozwiązywania równania Hamiltona-Jacobiego-Bellmana [17], będącego ze swej natury skomplikowanym nieliniowym równaniem różniczkowym cząstkowym. O wiele prostszym sposobem jest dokonanie dyskretyzacji problemu (5.70)–(5.72), tak aby otrzymać układ czasowo-dyskretny z funkcją kosztu addytywną względem czasu, a następnie zastosowanie programowania dynamicznego do problemu czasowo-dyskretnego. Takie podejście przyjęto w niniejszej pracy.

W celu dyskretyzacji problemu (5.70)–(5.72), podzielmy horyzont czasu $T = (0, t_f]$ na K równych podprzedziałów $T_k = (t_{k-1}, t_k]$, nazywanych etapami, gdzie: $t_k = k\Delta t$, $k = 1, \ldots, K$, $\Delta t = t_f/K$. Załóżmy również, że sterowania u będziemy poszukiwać w klasie funkcji przedziałami stałych o postaci

$$u(t) = u_k, \quad t \in T_k, \quad k = 1, \ldots, K. \tag{5.73}$$

Dla wystarczająco dużej liczby podprzedziałów T_k, czyli gdy Δt jest wystarczająco małe, pochodną po lewej stronie równania (5.70) można przybliżyć ilorazem różnicowym, tzn.

$$\frac{v_{k+1} - v_k}{\Delta t} \approx F(v_k, u_{k+1}, t_k), \tag{5.74}$$

gdzie: $v_k = v(t_k), k = 0, \ldots, K$. Prowadzi to do schematu Eulera [112, 66]

$$v_{k+1} = v_k + \Delta t\, F(v_k, u_{k+1}, t_k), \quad k = 0, \ldots, K-1. \tag{5.75}$$

Postępując dalej wg podobnej zasady, całkę w funkcjonale (5.72) można przybliżyć sumą wg metody prostokątów [66]:

$$J(u) \approx G(v_K) + \Delta t \sum_{k=0}^{K-1} C(v_k, u_{k+1}, t_k). \tag{5.76}$$

Podsumowując, rozważana dalej dyskretna wersja problemu (5.70)–(5.72) jest następująca: należy określić ciąg kolejnych poziomów sterowania $u_1, \ldots, u_K$, który dla układu dynamicznego opisanego równaniem stanu

$$v_{k+1} = F_k(v_k, u_{k+1}), \quad k = 0, \ldots, K-1, \tag{5.77}$$

doprowadzi do minimum funkcję kosztu

$$J(u) = g_K(v_K) + \sum_{k=0}^{K-1} g_k(v_k, u_{k+1}), \tag{5.78}$$

przy ograniczeniach

$$u_k \in U, \quad k = 1, \ldots, K, \tag{5.79}$$

gdzie:

$$\begin{aligned} F_k(v_k, u_{k+1}) &= \Delta t\, F(v_k, u_{k+1}, t_k), \quad k = 0, \ldots, K-1, \\ g_k(v_k, u_{k+1}) &= \Delta t\, C(v_k, u_{k+1}, t_k), \quad k = 0, \ldots, K-1, \\ g_K(v_K) &= G(v_K), \\ U &= \{u : u_{\min} \leqslant u \leqslant u_{\max}\}. \end{aligned} \tag{5.80}$$

Programowanie dynamiczne

U podstaw programowania dynamicznego leży twierdzenie ([17], str. 23), że minimalna wartość funkcji kosztu $J(u^*)$ odpowiadająca optymalnemu ciągowi poziomów sterowania $u^* = (u_1^*, \ldots, u_k^*)$ jest równa wielkości $J_0(v_0)$, otrzymanej w ostatnim kroku następującego algorytmu programowania dynamicznego, który operuje rekurencyjnie we wstecznym kierunku w czasie, począwszy od etapu T_K, a kończywszy na etapie T_1:

$$J_K(v_K) = g_K(v_K), \tag{5.81}$$

$$J_k(v_k) = \min_{u_{k+1} \in U} \left\{ g_k(v_k, u_{k+1}) + J_{k+1}(F_k(v_k, u_{k+1})) \right\}, \quad k = 0, \ldots, K-1. \tag{5.82}$$

Jako u_{k+1}^* przyjmuje się poziom sterowania u_{k+1} stanowiący punkt minimum po prawej stronie wzoru (5.82).

W praktyce, proces rozwiązywania zadania programowania dynamicznego (5.81)–(5.82) dla ogólnej postaci funkcji g_k oraz F_k niezmiernie komplikuje konieczność znajdowania minimum globalnego po prawej stronie równania (5.82). Najczęściej stosowanym i zarazem najprostszym sposobem redukcji nakładu obliczeniowego jest dyskretyzacja zbiorów sterowań dopuszczalnych U oraz przestrzeni stanów V. Oznacza to, że w ich miejsce rozważa się odpowiednie skończone zbiory punktów

$$\widetilde{U} = \left\{ u^1, \ldots, u^I \right\} \tag{5.83}$$

oraz

$$\widetilde{V} = \left\{ v^1, \ldots, v^L \right\}, \tag{5.84}$$

od których wymaga się, aby tworzyły wystarczająco gęste siatki w zbiorach U i V.

Przykładowo, narzucającym się wyborem są siatki równomierne: gdy $u \in \mathbb{R}^r$, dokonuje się dyskretyzacji każdej ze składowych wektora sterowania z osobna i dalej rozważa się wartości dopuszczalne sterowania $u = (u_{(1)}, \ldots, u_{(r)})$ postaci

$$u_{(\rho)} = u_{\min(\rho)} + p\, \Delta u_{(\rho)}, \quad p = 0, \ldots, P; \quad \rho = 1, \ldots, r, \tag{5.85}$$

gdzie: P — arbitralnie ustalona liczba przedziałów,

$$\Delta u_{(\rho)} = \frac{u_{\max(\rho)} - u_{\min(\rho)}}{P}, \tag{5.86}$$

a zapis $u_{(i)}$ oznacza i-tą składową wektora u. W odniesieniu do (5.83), oznacza to wybór $I = r(P+1)$.

W analogiczny sposób, przyjmując założenie, że zmienna stanu może przyjąć wartości w pewnym zbiorze zawartym w hiperkostce

$$\bar{V} = \{v : v_{\min} \leqslant v \leqslant v_{\max}\}, \tag{5.87}$$

gdzie: $v_{\min}, v_{\max}$ — pewne ustalone wektory (ma to swoje uzasadnienie w ograniczoności zbioru sterowań U), i ustalając arbitralnie liczbę podziałów S, można ograniczyć uwagę do stanów dopuszczalnych $v = (v_{(1)}, \ldots, v_{(q)})$ postaci

$$v_{(\pi)} = v_{\min(\pi)} + s\,\Delta\, v_{(\pi)}, \quad s = 0, \ldots, S, \quad \pi = 1, \ldots, q, \tag{5.88}$$

gdzie:

$$\Delta v_{(\pi)} = \frac{v_{\max(\pi)} - v_{\min(\pi)}}{S}. \tag{5.89}$$

W kontekście (5.84), oznacza to wybór $L = q(S+1)$.

Niezaprzeczalną zaletą przedstawionej metody jest jej prostota: obliczenia wykonywane są w sposób rekurencyjny, począwszy od ostatniego etapu, poprzez kolejne etapy pośrednie, kończąc na etapie pierwszym, a określenie punktu minimum po prawej stronie równania (5.82) sprowadza się do porównania skończonej liczby wartości bieżącej funkcji kryterialnej. Niestety, zaleta ta nie kompensuje największej wady algorytmu programowania dynamicznego, którą jest eksplozja kombinatoryczna pojawiająca się w złożonych problemach praktycznych.

Przybliżenie oryginalnych przestrzeni V i U skończonymi zbiorami $\widetilde{V}$ i $\widetilde{U}$ wymaga stosowania odpowiednio gęstych siatek dyskretyzacji. Przestrzeń wszystkich możliwych rozwiązań problemu dyskretnego przeglądanych z wykorzystaniem programowania dynamicznego liczy sobie $(L \cdot I)^{K-1} + L$ przebiegów sterowań, co jest liczbą pozostającą poza zasięgiem współczesnych komputerów nawet przy umiarkowanych gęstościach zastosowanych siatek dyskretyzacji.

Zjawisko eksplozji kombinatorycznej (inaczej nazywane „przekleństwem wymiarowości") od lat stymulowało badania nad tzw. przybliżonymi algorytmami programowania dynamicznego [18, 176], które osiągnęły już dojrzałość wystarczającą do rozwiązywania wieloetapowych problemów decyzyjnych o wymiarowościach spotykanych w przemyśle, a przy tym z uwzględnieniem rozmaitych typów niepewności.

W badaniach objętych niniejsza rozprawą doktorską zwrócono uwagę na jeden z algorytmów tej klasy, tzw. iteracyjne programowanie dynamiczne, zaproponowane przez Luusa [20, 129, 130, 131, 132, 216], przede wszystkim z uwagi na bardzo duże (chociaż nie wykorzystane) możliwości zrównoleglania.

5.3.4 Iteracyjne Programowanie Dynamiczne

W celu złagodzenia przedstawionych wyżej problemów Luus opracował metodę nazywaną iteracyjnym programowaniem dynamicznym [133]. Wykorzystanie iteracyj-

nego programowania dynamicznego pozwala również na rozwiązanie wielu problemów pojawiających się podczas implementacji programowania dynamicznego, wśród których można wymienić następujące:

- potrzeba gęstej dyskretyzacji przestrzeni stanu oraz sterowań,
- konieczność całkowitego przeszukiwania przestrzeni stanów oraz sterowań,
- konieczność interpolacji punktów znajdujących się pomiędzy węzłami siatki.

Poniżej dokonuje się krótkiej charakterystyki każdego z nich.

Gęsta dyskretyzacja przestrzeni stanu i sterowań

Gęsta dyskretyzacja przestrzeni stanu oraz sterowań pozwalała na zwiększanie poziomu dokładności przybliżeń sterowań optymalnych wyznaczanych w procesie programowania dynamicznego. Jednak zagęszczanie zarówno przestrzeni stanów jak i przestrzeni sterowań powoduje olbrzymie zwiększenie nakładu obliczeń, z którego istotna część dotyczy eksploracji przestrzeni, w których sterowanie optymalne nigdy nie zostanie wyznaczone. Rozwiązaniem zaproponowanym przez Luusa [133] jest stopniowe zagęszczanie siatek stanów i sterowań wokół rozwiązań otrzymywanych w poprzednich iteracjach. Takie postępowanie umożliwia znaczne zmniejszenie wymaganej gęstości stosowanych siatek, z uwagi na stopniowe skupianie się algorytmu wokół węzłów umożliwiających uzyskanie rozwiązania optymalnego.

Eksploracja przestrzeni stanu oraz sterowań

Oryginalny algorytm programowania dynamicznego w postaci dyskretnej zakłada podział przestrzeni stanów oraz sterowań na siatki o zadanej gęstości ((5.83)–(5.89) odpowiadają podziałowi równomiernemu). W pracach [20, 21, 131, 133, 216] pokazane zostało, że przy rozwiązywaniu nawet skomplikowanych zagadnień sterowania optymalnego nie występuje konieczność arbitralnego podziału przestrzeni stanu oraz sterowań na siatki o określonej gęstości (nawet z uwzględnieniem dynamicznego zagęszczania siatek wokół rozwiązań wyznaczonych w poprzednich iteracjach). Dyskretyzacja przestrzeni stanu została zastąpione przez generowanie węzłów (trajektorii) w przestrzeni stanu, których liczba $\bar{L}$ jest znacznie mniejsza niż liczba węzłów siatki stanu, czyli $\bar{L} \ll L$ (patrz (5.87)–(5.89)). Proces tworzenia siatki sterowań zgodnie z (5.85)–(5.86) został zastąpiony przez stochastyczne przeszukiwanie przestrzeni sterowań, dla liczby różnych sterowań $\bar{I} \ll I$.

Pokazany w pracach [20, 21, 131, 133, 216] wpływ takiego uproszczenia na zbieżność oraz dokładność pracy metody iteracyjnego programowania dynamicznego predestynuje go do zastosowań praktycznych. Dobór dokładnych wartości parametrów $\bar{L}$ oraz $\bar{I}$ ściśle zależy od rozwiązywanego problemu i powinien wynikać z pewnego doświadczenia nabytego podczas stosowania metody iteracyjnego programowania dynamicznego.

Interpolacja punktów pomiędzy węzłami siatki

Jak łatwo zauważyć, wprowadzona dyskretyzacja zbiorów U i V (5.83)–(5.84) powoduje, że stają się one niekompatybilne, tzn. jeżeli w pewnej chwili k układ jest w stanie $v_k \in \widetilde{V}$, to po wybraniu sterowania $u_{k+1} \in \widetilde{U}$ tranzycja zdefiniowana równaniem (5.77) doprowadzi do stanu v_{k+1} spoza wprowadzonej siatki $\widetilde{V}$, co ilustruje przykład z rys. 5.5, w którym założono $q = 2$, $L = 16$ oraz $I = 4$. W oryginalnym podejściu programowania dynamicznego stany nie mające swojego odzwierciedlenia w węzłach istniejącej siatki są interpolowane, co znacząco zwiększa koszt obliczeniowy całego algorytmu.

Luus zaproponował [133] modyfikację równania stanu (5.77), polegającą na zamianie wyznaczonego stanu v_{k+1} najbliższym mu węzłem siatki $\widetilde{V}$:

$$v_{k+1} = \tilde{F}_k(v_k, u_{k+1}), \quad k = 0, \ldots, K-1, \tag{5.90}$$

gdzie:

$$\tilde{F}_k(v_k, u_{k+1}) = \arg\min_{v \in \widetilde{V}} \|F_k(v_k, u_{k+1}) - v\|, \tag{5.91}$$

$\|\cdot\|$ — odległość euklidesowa.

W rezultacie powyższego uproszczenia, w pełni dyskretna wersja algorytmu iteracyjnego programowania dynamicznego przyjmuje następującą postać:

$$J_K(v_K) = g_K(v_K) \tag{5.92}$$

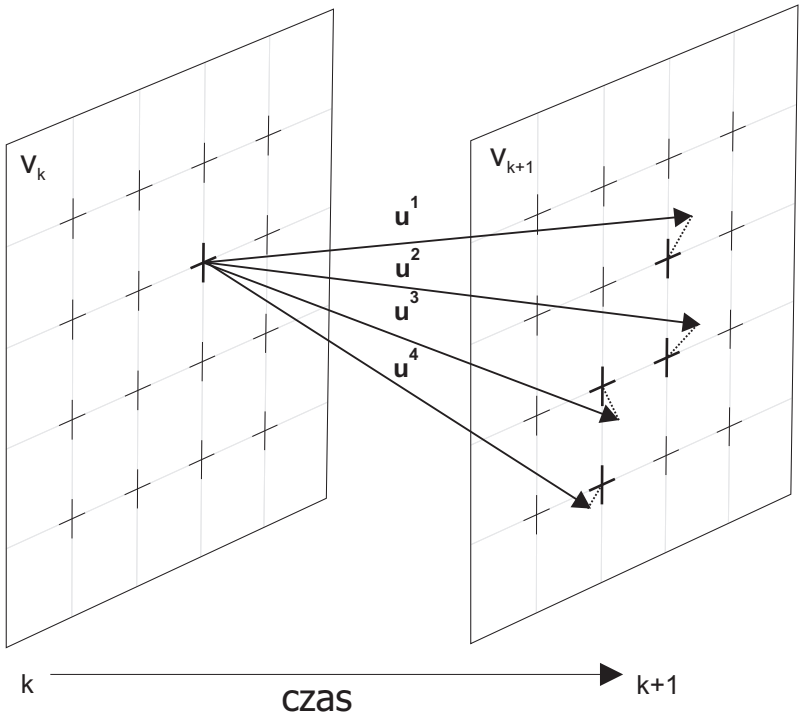
$$J_k(v_k) = \min_{u_{k+1} \in \tilde{U}} \left\{ g_k(v_k, u_{k+1}) + J_{k+1}(\tilde{F}(v_k, u_{k+1})) \right\}, \quad k = 0, \ldots, K-1. \tag{5.93}$$

Przedstawione modyfikacje spowodowały, że algorytm iteracyjnego programowania dynamicznego pozwala na rozwiązywanie skomplikowanych problemów optymalizacyjnych [20, 131, 132]. Jego zalety przyczyniły się również do tego, że w niniejszej rozprawie został wykorzystany w metodzie wyznaczania optymalnych trajektorii mobilnych węzłów sieci sensorycznej uwzględniającej dynamikę pojazdów przenoszących czujniki.

5.3.5 Proponowany algorytm

Szablon metody przedstawiono w postaci Algorytmu 5.5. Algorytm opiera się na głównej pętli, w której wykonywana jest procedura wyznaczania optymalnego sterowania dla zadanego początkowego wektora stanu. Po wyznaczeniu sterowania optymalnego modyfikowany jest parametr sterujący gęstością siatki sterowań oraz stanów. Zmniejszanie jego wartości zgodnie z przyjętym współczynnikiem redukcji ma na celu utworzenie w kolejnej iteracji nowej dyskretnej przestrzeni stanów wokół wektorów stanu wyznaczonych na podstawie aktualnego sterowania optymalnego. Algorytm kończy swoją pracę po wykonaniu arbitralnie określonej liczby iteracji.

Procedurę znajdowania sterowania optymalnego przedstawioną jako Algorytm. 5.5 można podzielić na trzy części:



Rysunek 5.5: Wybór stanu systemu w zależności od sterowania.

- tworzenie dyskretnej przestrzeni stanów,
- generacja siatek sterowania z równoczesnym rozwiązywaniem zadania programowania dynamicznego,
- wyznaczenie optymalnej strategii sterowania.

Krok pierwszy ma na celu utworzenie nowej przestrzeni stanów. Na podstawie początkowego wektora stanu oraz optymalnego sterowania wyznaczonego w poprzedniej iteracji (bądź początkowego przybliżenia sterowania początkowego) generowana jest $\bar{L}$-elementowa siatka w przestrzeni stanów V. Każdy węzeł tej siatki powstaje na podstawie początkowego wektora stanu oraz losowego zaburzania aktualnego sterowania optymalnego. Wielkość zaburzenia/modyfikacji sterowania jest określona wektorem ρ. Wektor ρ steruje gęstością tworzonej dyskretnej przestrzeni stanu i pozwala na dokładniejszą eksplorację wraz z wykonywaniem kolejnych iteracji algorytmu. Tworzenie $\bar{L}$-elementowej przestrzeni stanu V ma na celu zapobiegnięcie przeszukiwaniu olbrzymiej liczby węzłów jakie byłoby potrzebne w przypadku zastosowania oryginalnego algorytmu programowania dynamicznego.

W kroku drugim następuje generacja siatek sterowania z równoczesnym rozwiązywaniem zadania programowania dynamicznego. Proces rozpoczyna się od ostatniego K-tego interwału czasowego i jest wykonywany wstecz, kolejno w kierunku pierwszego etapu czasowego. W każdym etapie analizuje się każdy węzeł aktualnej

dyskretnej przestrzeni stanów. Dla każdego elementu v_k przestrzeni stanów $V_k^{(\tau)}$ testuje się $\bar{I}$ sterowań składających się na aktualną dyskretną siatkę w przestrzeni sterowań. Dla każdego wygenerowanego testowego wektora sterowań i aktualnego stanu rozwiązuje się zadanie minimalizacji (5.93). Podczas tej czynności wykorzystywana jest zmodyfikowana funkcja przejścia $\tilde{F}_k$ (5.91), która zwraca węzeł w aktualnej dyskretnej przestrzeni stanów najbliższy stanowi wyznaczonemu wg równania (5.77). Najlepsze ze sterowań zapamiętuje się $\tilde{u}_{k+1}^{(\tau)[v_k]}$

W ostatnim kroku, z utworzonej na podstawie optymalnych wartości sterowań dla każdej trajektorii v_k, przestrzeni sterowań $\tilde{U}_{k+1}^{(\tau)}$ wyznaczone są te sterowania, które pozwalają na uzyskanie optimum wskaźnika jakości. Procedura polega na rozwiązaniu zadania programowania dynamicznego dla początkowego wektora stanu v_0 z jednoczesnym zapamiętywaniem kolejnych wybieranych sterowań jako sterowanie optymalne $\bar{u}^{\tau+1}$. Wokół tak wyznaczonego sterowania optymalnego w kolejnej iteracji zostanie wygenerowana $\bar{L}$-elementowa siatka przestrzeni stanu umożliwiających wyznaczenie kolejnego przybliżenia sterowania optymalnego.

Algorytm 5.5 Procedura wyznaczania sterowania optymalnego

Oznaczenia:

$\mathrm{rand}(\rho)$ — dla r-wymiarowego wektora $\rho \geqslant 0$ symbol $\mathrm{rand}(\rho)$ oznacza realizację wektora pseudolosowego o rozkładzie równomiernym w hiperkostce $[-\rho_{(1)}, \rho_{(1)}] \times \ldots \times [-\rho_{(r)}, \rho_{(r)}]$.

$\mathrm{Proj}_U(u)$ — operacja rzutowania wektora $u \in \mathbb{R}^r$ na zbiór sterowań dopuszczalnych U; zachodzi

$$[\mathrm{Proj}_U(u)]_{(j)} = \begin{cases} u_{\min(j)} & \text{jeżeli } u_{(j)} < u_{\min(j)}, \\ u_{\max(j)} & \text{jeżeli } u_{(j)} > u_{\max(j)}, \\ u_{(j)} & \text{w przeciwnym razie.} \end{cases} \tag{5.94}$$

Dane wejściowe:

K — liczba etapów,

$\bar{L}$ — wielkość siatki w przestrzeni stanów,

$\bar{I}$ — wielkość siatki w przestrzeni sterowań,

$\tau_{\max}$ — maksymalna liczba iteracji,

$\bar{u}^{(0)} = (\bar{u}_1^{(0)}, \ldots, \bar{u}_K^{(0)})$ — początkowa strategia sterowania,

$\rho^{(0)} \in \mathbb{R}_+^r$ — wektor określający początkowy promień poszukiwań sterowań po poszczególnych współrzędnych,

$\gamma \in (0, 1)$ — współczynnik określający szybkość zmniejszania promienia poszukiwań sterowań.

Dane wyjściowe:

$\bar{u}$ — optymalna strategia sterowania.

Krok 0: (Inicjalizacja)
Ustaw licznik iteracji $\tau \leftarrow 0$.

Krok 1: (Tworzenie dyskretnej siatki przestrzeni stanów)
Ustaw $V_0^{(\tau)} = \{v_0\}$.
Dla $k = 0, \ldots, K-1$ wyznacz kolejno:

$$\begin{aligned}
\xi_{k+1}^{\ell(\tau)} &= \mathrm{rand}(\rho^{(\tau)}),\\
\tilde{u}_{k+1}^{\ell(\tau)} &= \mathrm{Proj}_U(\bar{u}_{k+1}^{(\tau)} + \xi_{k+1}^{\ell(\tau)}),\\
v_{k+1}^{\ell(\tau)} &= F(v_k^{\ell(\tau)}, \tilde{u}_{k+1}^{\ell(\tau)}),
\end{aligned}$$

gdzie: $\ell = 1, \ldots, \bar{L}$, tworząc w ten sposób kolejne siatki przestrzeni stanów

$$V_{k+1}^{(\tau)} = \{v_{k+1}^{1(\tau)}, \ldots, v_{k+1}^{\bar{L}(\tau)}\},$$

Krok 2: (Generacja siatek sterowania z równoczesnym rozwiązywaniem zadania programowania dynamicznego)
Dla każdego $v_K \in V_K^{(\tau)}$ ustaw $J_K(v_K) = g_K(v_K)$.
Kolejno dla $k = K-1, \ldots, 0$ wyznaczaj

Faza 1:
Dla każdego $v_k \in V_k^{(\tau)}$ oblicz

$$\begin{aligned}
\xi^{i(\tau)} &= \mathrm{rand}(\rho^{(\tau)}),\\
u_{k+1}^{i(\tau)} &= \mathrm{Proj}_U(\bar{u}_{k+1}^{(\tau)} + \xi^{i(\tau)}),
\end{aligned}$$

gdzie: $i = 1, \ldots, \bar{I}$, otrzymując dla każdego v_k siatkę sterowań

$$\widetilde{U}_{k+1}^{(\tau)[v_k]} = \{u_{k+1}^{1(\tau)}, \ldots, u_{k+1}^{\bar{I}(\tau)}\}.$$

Faza 2:
Dla każdego $v_k \in V_k^{(\tau)}$ oblicz

$$J_k(v_k) = \min_{u_{k+1} \in \widetilde{U}_{k+1}^{(\tau)[v_k]}} \{g_k(v_k, u_{k+1}) + J_{k+1}(\tilde{F}_k(v_k, u_{k+1}))\}.$$

przy czym punkt minimum w wyrażeniu po prawej stronie zapamiętaj jako $\tilde{u}_{k+1}^{(\tau)[v_k]}$.

Krok 3: (Wyznaczenie optymalnej strategii sterowania)
Utwórz ciąg sterowań optymalnych

$$\bar{U}^{(\tau+1)} = \{\bar{u}_1^{(\tau+1)}, \ldots, \bar{u}_k^{(\tau+1)}\}.$$

wg reguły

$$\bar{u}_{k+1}^{(\tau+1)} = \tilde{u}_{k+1}^{(\tau)[v_k^{(\tau+1)}]}, \quad k = 0, \ldots, K-1,$$

przy czym

$$\begin{aligned} \bar{v}_0^{(\tau+1)} &= \bar{v}_0 \\ \bar{v}_{k+1}^{(\tau+1)} &= \widetilde{F}_k\Big(\bar{v}_k^{(\tau+1)}, \bar{u}_{k+1}^{(\tau+1)}\Big), \quad k = 0, \ldots, K-1. \end{aligned} \tag{5.95}$$

Krok 4: (Warunek stopu)
Jeżeli $\tau = \tau_{\max} - 1$, zakończ. W przeciwnym razie ustaw $\rho^{(\tau+1)} = \gamma\rho^{(\tau)}$ oraz $\tau \leftarrow \tau + 1$ i wróć do Kroku 1.

■

5.3.6 Zbieżność algorytmu

Parametr $\bar{I}$ określający wielkość siatki w przestrzeni sterowań jest definiowany na starcie algorytmu i oprócz wpływu na dokładność znajdowanych wyników ma również wpływ na szybkość zbieżności algorytmu. Wykorzystanie większej liczby wartości testowych wektora sterowań pozwala na lepszą eksplorację przestrzeni rozwiązań, jednakże odbywa się to kosztem zwiększenia nakładu obliczeń [216]. Na rys. 5.6 można zaobserwować szybkość zbieżności algorytmu do optimum globalnego w zależności od wartości parametru $\bar{I}$ dla Przykładu 5.3.

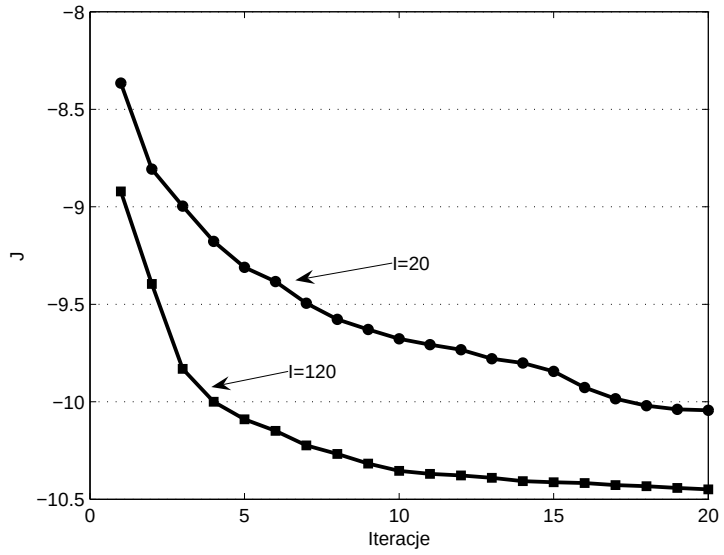
Przykład 5.3
Jako przykład numeryczny rozważmy proces przewodnictwa ciepła w niejednorodnej płycie metalowej zajmującej obszar przestrzenny $\Omega = (0,1)^2 \in \mathbb{R}^2$ o brzegu $\partial\Omega$, opisany w horyzoncie czasowym $T = (0,1)$ następującym równaniem

$$\begin{aligned} \frac{\partial y(x,t)}{\partial t} = \frac{\partial}{\partial x_1}\left(\kappa(x)\frac{\partial y(x,t)}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(\kappa(x)\frac{\partial y(x,t)}{\partial x_2}\right) \\ + 20\exp(-50(x_1 - t)^2), \qquad x \in \Omega, \quad t \in T, \end{aligned} \tag{5.96}$$

z warunkami początkowymi oraz brzegowymi

$$\begin{aligned} y(x,0) &= 0, \quad x \in \Omega, \\ y(x,t) &= 0, \quad (x,t) \in \partial\Omega \times T. \end{aligned} \tag{5.97}$$

Wymuszenie po prawej stronie równania (5.96) imituje działanie źródła ciepła, którego nośnikiem jest linia prosta równoległa do osi x_2, poruszająca się od lewego do prawego brzegu obszaru Ω.



Rysunek 5.6: Zbieżność algorytmu w zależności od parametru $\bar{I}$.

Współczynnik dyfuzji określony jest następująco

$$\kappa(x) = \theta_1 + \theta_2 x_1 + \theta_3 x_2, \tag{5.98}$$

przy czym wartości parametrów $\theta_1, \theta_2, \theta_3$ powinny być możliwie najdokładniej oszacowane na podstawie pomiarów wykonanych przez sieć składającą się z $n = 3$ sensorów mobilnych. Do celów planowania użyto następujących wartości nominalnych $\theta_1^0 = 0.1, \theta_2^0 = -0.05, \theta_3^0 = 0.2$.

Poszukiwane są D-optymalne trajektorie ruchu trzech mobilnych węzłów sieci sensorycznej opisane jako:

$$(s_1(t), s_2(t)), (s_3(t), s_4(t)), (s_5(t), s_6(t)).$$

Dynamika ruchu węzłów mobilnych jest opisana układem równań różniczkowych

$$\begin{cases} \dot{s}_1(t) = u_1(t), & s_1(0) = s_{10}, \\ \quad \vdots & \quad \vdots \\ \dot{s}_6(t) = u_6(t), & s_6(0) = s_{60}, \end{cases} \tag{5.99}$$

gdzie $u_1, \ldots, u_6$ określają sterowanie, którego wyznaczenie jest przedmiotem rozważań. Powyższy opis oznacza, że steruje się bezpośrednio składowymi prędkości sensorów wzdłuż obu osi bazowego układu współrzędnych przestrzennych.

Jako kryterium optymalności przyjęto kryterium D-optymalności, a zgodnie z rozważaniami zawartymi w rozdz. 5.3.2 dotyczącymi transformacji do problemu

Mayera, problem poszukiwania optymalnych trajektorii możemy zapisać jako problem minimalizacji wskaźnika jakości następującej postaci:

$$\tilde{J} = -\log(\det\left(\begin{bmatrix} \zeta_1(t_f) & \zeta_2(t_f) & \zeta_3(t_f) \\ \zeta_2(t_f) & \zeta_4(t_f) & \zeta_5(t_f) \\ \zeta_3(t_f) & \zeta_5(t_f) & \zeta_6(t_f) \end{bmatrix}\right)) + c\sum_{i=1}^{6}\int_0^{t_f} u_i^2(t)\mathrm{d}t \tag{5.100}$$

określonego wzdłuż rozwiązań następującego rozszerzonego układu stanu

$$\begin{cases} \dot{s}_1(t) = u_1(t), & s_1(0) = s_{10}, \\ \quad\vdots & \quad\vdots \\ \dot{s}_6(t) = u_6(t), & s_6(0) = s_{60}, \\ \dot{\zeta}_1(t) = \sum_{j=1}^{3} g_1^2(s_{2j-1}(t), s_{2j}(t), t), & \zeta_1(0) = 0, \\ \dot{\zeta}_2(t) = \sum_{j=1}^{3} g_2(s_{2j-1}(t), s_{2j}(t), t)\cdot g_1(s_{2j-1}(t), s_{2j}(t), t), & \zeta_2(0) = 0, \\ \dot{\zeta}_3(t) = \sum_{j=1}^{3} g_3(s_{2j-1}(t), s_{2j}(t), t)\cdot g_1(s_{2j-1}(t), s_{2j}(t), t), & \zeta_3(0) = 0, \\ \dot{\zeta}_4(t) = \sum_{j=1}^{3} g_2^2(s_{2j-1}(t), s_{2j}(t), t), & \zeta_4(0) = 0, \\ \dot{\zeta}_5(t) = \sum_{j=1}^{3} g_3(s_{2j-1}(t), s_{2j}(t), t)\cdot g_2(s_{2j-1}(t), s_{2j}(t), t), & \zeta_5(0) = 0, \\ \dot{\zeta}_6(t) = \sum_{j=1}^{3} g_3^2(s_{2j-1}(t), s_{2j}(t), t), & \zeta_6(0) = 0, \end{cases} \tag{5.101}$$

gdzie g_1, g_2, g_3 oznaczają współczynniki wrażliwości (tzn. pochodne rozwiązania zagadnienia (5.96), (5.97)) odpowiednio względem parametrów θ_1, θ_2 i θ_3. Wektor sterowań u jest ograniczony przez następujący układ nierówności

$$\begin{cases} u_{1\,\min} & \leqslant u_1(t) & \leqslant u_{1\,\max}, \\ \quad\vdots & \quad\vdots & \quad\vdots \\ u_{6\,\min} & \leqslant u_6(t) & \leqslant u_{6\,\max}. \end{cases} \tag{5.102}$$

przy czym $u_{1\,\min} = \ldots = u_{6\,\min} = -0.7$ oraz $u_{1\,\max} = \ldots = u_{6\,\max} = 0.7$. W kryterium (5.100) uwzględniono składnik całkowy mający na celu wygładzanie wyznaczanych trajektorii. Można go interpretować w kategoriach energii poruszających się pojazdów, a wpływ na końcowe rozwiązanie jest zależny od wartości parametru c.

Rozwiązania zagadnienia (5.96),(5.97) wraz z odpowiadającymi mu równaniami wrażliwościowymi g_1, g_2, g_3 względem odpowiednio $\theta_1, \theta_2, \theta_3$ są zdefiniowane na dyskretnej siatce o rozmiarach 41×41. Otrzymano jest z zastosowaniem metody elementu skończonego i przybornika Partial Differential Equation dostępnego w programie Matlab (por. Dodatek I.2.1 w pracy [228]). Jednak rozwiązanie równań

(5.100)–(5.102) wymaga określenia wartości wektorów wrażliwości również w punktach nie należących do siatki dyskretyzacji. Interpolacja zmiennych przestrzennych jest wykonywana za pomocą interpolacji biliniowej. Sposób wyznaczania wartości punktów pośrednich przedstawiono w Dodatkach A.1 oraz A.2.

Dodatkowo, zdefiniowane są pozycje początkowe czujników

$$s_0 = (0.2, 0.1, 0.2, 0.5, 0.2, 0.8). \tag{5.103}$$

Rozwiązanie zagadnienia (5.96),(5.97) przedstawiono na rys. 5.8. Obliczenia przeprowadzono w oparciu o uniwersytecki klaster zbudowany w ramach projektu CLUSTERIX [248] (zob. rozdz. 3.9). Klaster składa się z węzłów zawierających 64-bitowe procesory Intel Itanium^2 1.4GHz i działa pod kontrolą środowiska GNU/Linux Debian for ia64. Algorytm 5.5 zaimplementowano w języku Fortran 95 i skompilowano z zastosowaniem kompilatora *ifort* (Intel®Fortran Compiler v.8.1 for Linux 64-bit platforms). Aplikacja używa biblioteki procedur numerycznych Intel Math Kernel Library do wykonywania operacji mnożenia macierzy oraz wektorów.

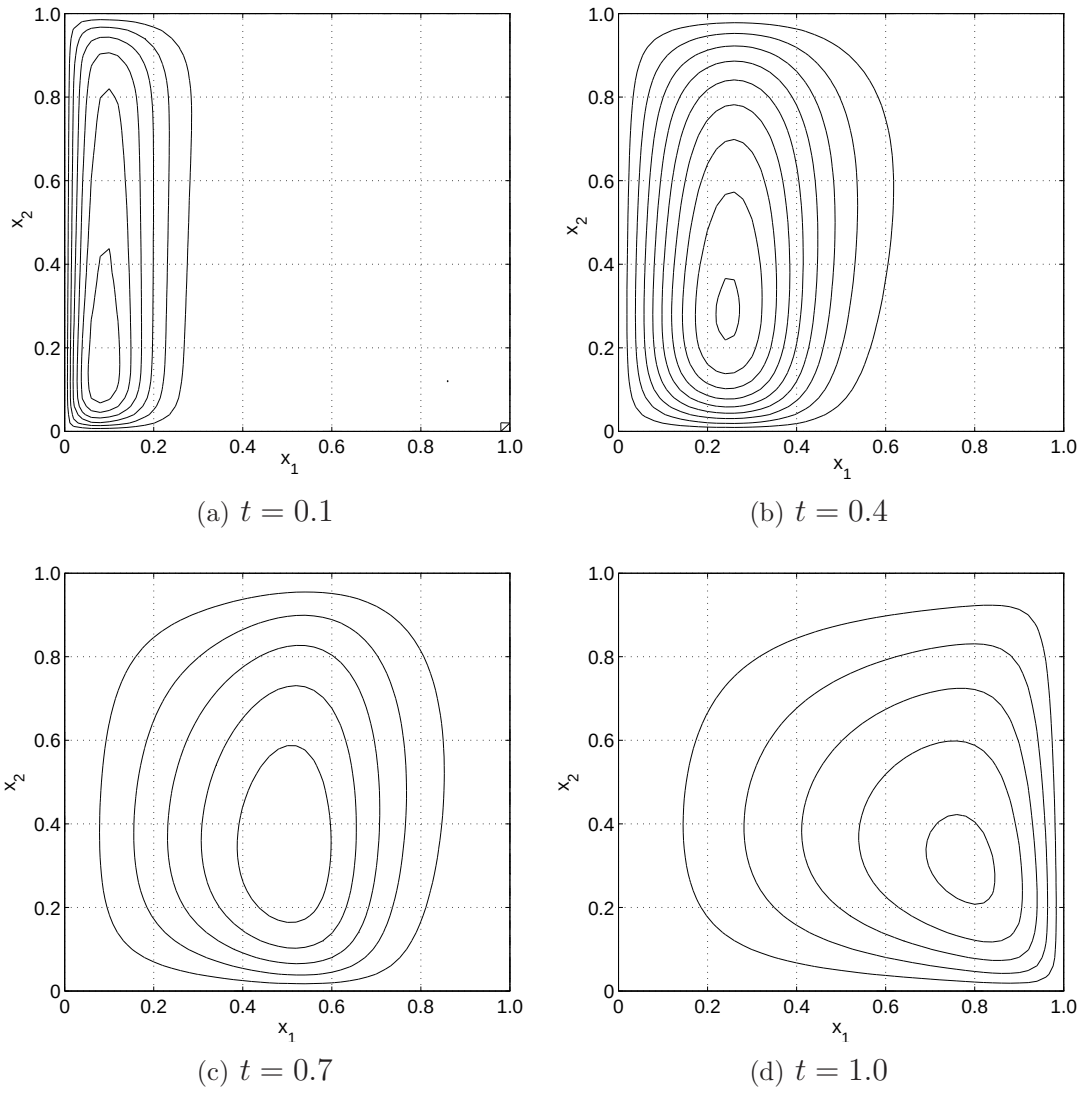
Przyjęto następujące wartości parametrów algorytmu:

- liczba etapów $K = 31$,
- wielkość siatki w przestrzeni sterowań $\bar{I} = 24$,
- parametr zagęszczania siatek dyskretyzacji $\gamma = 0.9$,
- początkowy promień poszukiwań sterowań $\rho^{(0)} = 0.6$,
- wielkość siatki w przestrzeni stanu $\bar{L} = 11$,
- maksymalna liczba iteracji $\tau_{\max} = 35$,
- początkowy wektor sterowań $\bar{u}^{(0)} = (0.4, 0.0, 0.4, 0.0, 0.4, 0.0)$,
- parametr kary $c = 0.5$.

Analizując uzyskane rezultaty obliczeń można zauważyć, że wyznaczone optymalne strategie sterowania pokrywają się z intuicyjnym wyobrażeniem, że czujniki powinny śledzić poruszające się źródło ciepła. Oczywiście, złożona postać równania (5.98) opisująca współczynnik dyfuzji równania powoduje niejednorodne rozprzestrzenianie się ciepła, co również ma odzwierciedlenie w postaci trajektorii mobilnych węzłów sieci sensorycznej. Uzyskane rezultaty są również zbieżne z wynikami przedstawionymi w pracach [154, 228].

Obliczenia w przedstawionym środowisku trwały ponad 36 minut co, przy rozwiązywaniu zagadnienia wyznaczania optymalnych trajektorii dla 3 mobilnych węzłów sieci sensorycznej, stanowiło motywację do opracowania równoległej metody pozwalającej na szybsze uzyskiwanie rezultatów.

♦



(a) $t = 0.1$ (b) $t = 0.4$

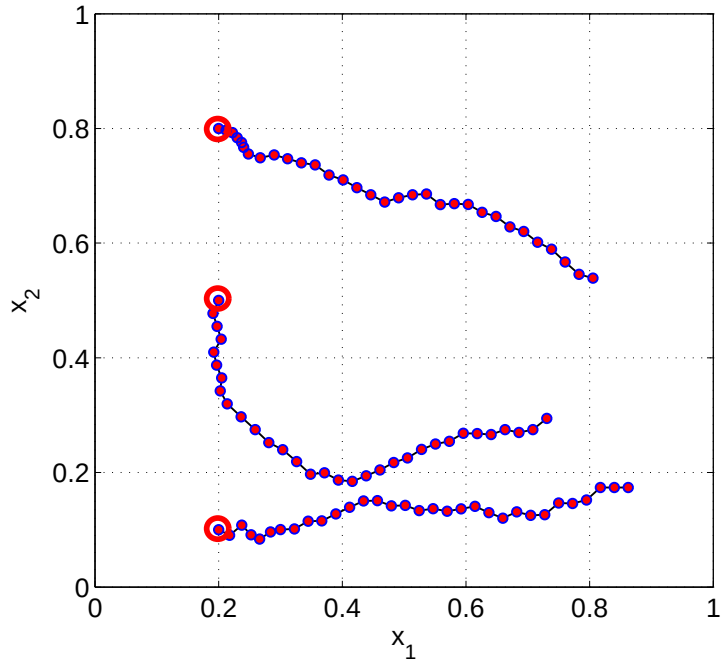
(c) $t = 0.7$ (d) $t = 1.0$

Rysunek 5.7: Izolinie rozwiązania równania dyfuzji z Przykładu 5.3.

5.3.7 Równoległa wersja algorytmu

Zaproponowanie metody zrównoleglania przedstawionego algorytmu zostało poprzedzone analizą rozwiązań zaproponowanych w literaturze. Również w rozważanym zagadnieniu, w literaturze istnieje stosunkowo niewiele publikacji poruszających zagadnienie zrównoleglania iteracyjnego programowania dynamicznego [84]. Analiza pokazała, że rozwiązania te wprawdzie mogą zostać adaptowane do rozważanych zagadnień optymalnej obserwacji układów z czasoprzestrzenną dynamiką, jednak po uwzględnieniu pewnych modyfikacji oraz dostosowaniu do specyfiki rozważanych problemów.

Proponowany algorytm oparty na iteracyjnym programowaniu dynamicznym przedstawiony w rozdz. 5.3.5 zawiera kilka zagnieżdżonych pętli, których wykonanie



Rysunek 5.8: Wyznaczone trajektorie dla Przykładu 5.3.

można zrównoleglić na wielu procesorach. W pracy [84] zaproponowano dekompozycję przestrzeni stanów czyli podział pętli badającej wygenerowane trajektorie opisujące przestrzeń stanów. Takie podejście spowodowało, że algorytm posiada stosunkowo małą ziarnistość, a przedstawione wyniki pokazują uzyskiwanie efektywności obliczeń na poziomie 50% dla klastra składającego się z 11 węzłów.

Wykorzystanie dekompozycji algorytmu w oparciu o podział siatki stanu, mogłoby spowodować sztuczne uzależnienie gęstości siatki przestrzeni stanu od liczby procesorów. W przypadku prowadzenia obliczeń np. na kilkuset procesorach, taki schemat wymuszałby konieczność stosowania siatki przestrzeni stanu w postaci kilkuset trajektorii. Jak pokazała przeprowadzona analiza problemu, rozwiązanie opierające się na zwiększaniu gęstości siatki stanu nie daje zadowalających rezultatów w postaci lepszego przybliżenia optimum globalnego lub też szybszej zbieżności algorytmu do optimum.

Szczegółowa analiza sekwencyjnej wersji algorytmu pozwoliła na określenie lepszego sposobu dekompozycji algorytmu. Szczególnie wartościowa była obserwacja dotycząca szybkości zbieżności algorytmu w zależności od gęstości dyskretnej siatki sterowań kontrolowanej przez parametr $\bar{I}$ algorytmu (zob. rozdz. 5.3.6). Przyspieszenie zbieżności wraz ze wzrostem liczby testowanych sterowań pozwoliło na zaproponowanie metody dekompozycji algorytmu w oparciu o podział siatki sterowań. Pozwala to na rozwiązywanie bardziej skomplikowanych zagadnień sterowania optymalnego związanych ze znajdowaniem optymalnych trajektorii węzłów mobilnych

sieci sensorycznej. Bardziej zaawansowane problemy wymagają wyznaczania bardziej dokładnych sterowań, a zwiększanie liczby generowanych sterowań testowych pozwala na dokładniejszą eksplorację przestrzeni sterowań.

Metoda zrównoleglenia

Algorytm wyznaczający optymalne trajektorie mobilnych węzłów sieci sensorycznej z uwzględnieniem dynamiki pojazdów przenoszących czujniki został zaprojektowany z wykorzystanie schematu komunikacji pomiędzy procesorami typu „każdy z każdym". W tym schemacie brak jest wyróżnionego procesora, który posiada informację o całym działającym algorytmie. Aby każdy z procesorów posiadał wiedzę o stanie algorytmu i fazie, w jakiej się znajduje, musi istnieć odpowiedni protokół komunikacji zrozumiały przez wszystkie węzły. Dzięki wymianie informacji, każdy z węzłów może efektywnie wykonywać algorytm oraz adaptować wielkość zadań w zależności od obciążenia. Schemat wymiany informacji „każdy z każdym" przedstawiono na rys. 4.8.

Równoległa wersja procedury znajdowania sterowania optymalnego została przedstawiona jako Algorytm 5.6. Podczas inicjalizacji algorytmu jeden z procesorów generuje ziarna generatorów liczb losowych i przesyła je do pozostałych procesorów liczących. Każdy procesor liczący wykorzystuje swój generator liczb losowych zainicjowany innym ziarnem i dzięki temu następuje wybór różnych losowych wartości wymaganych do tworzenia siatki przestrzeni stanów oraz sterowań.

Następnie, w pierwszym kroku następuje tworzenie siatki przestrzeni stanów w oparciu o $\bar{L}$ losowo wygenerowanych trajektorii. Każdy z procesorów liczących tworzy swoją dyskretną siatkę w przestrzeni stanów składającą się z $\bar{L}_p$ węzłów. Liczba trajektorii tworzona przez p-ty procesor zależy od obciążenia tego procesora w poprzednich iteracjach algorytmu (zob. rozdz. 5.3.8):

$$\bar{L}_p = \left\lfloor w_p \cdot \bar{L} \right\rfloor, \quad p = 1, \ldots, P, \tag{5.104}$$

gdzie: w_p — waga p-tego procesora wyznaczana przez algorytm dynamicznego równoważenia obciążenia lub arbitralnie, a P — liczba procesorów liczących. Po wygenerowaniu siatek składowych przez wszystkie procesory, następuje pomiędzy nimi wymiana tych siatek w celu utworzenia globalnej siatki $\bar{V}_{k+1}^{\tau}$.

W kolejnym kroku generowane są siatki sterowania z równoczesnym rozwiązywaniem zadania programowania dynamicznego w celu wyznaczenie sterowań optymalnych. Dekompozycja algorytmu została dokonana na poziomie generowania siatki sterowań dla każdego węzła siatki w przestrzeni stanów. Każdy z procesorów analizuje $\bar{I}_p$ różnych sterowań dla każdego etapu czasowego k i każdego węzła $v_k^{\ell(\tau)} \in V_k^{(\tau)}$. Liczbę punktów siatki w przestrzeni sterowań dla każdego procesora wyznacza się analogicznie do liczby generowanych trajektorii (5.104):

$$\bar{I}_p = \left\lfloor w_p \cdot \bar{I} \right\rfloor, \quad p = 1, \ldots, P, \tag{5.105}$$

gdzie: w_p — waga p-tego procesora wyznaczana przez algorytm dynamicznego równoważenia obciążenia lub arbitralnie, a P — liczba procesorów liczących.

Po wyznaczeniu przez każdy z procesorów optymalnego sterowania dla węzła v_k w k-tym etapie czasowym, następuje wymiana optymalnych sterowań pomiędzy procesorami. Operacja polega na początkowej wymianie wskaźników jakości, a następnie wyznaczeniu procesora, który określił sterowania pozwalające na uzyskanie najlepszej wartości wskaźnika jakości. Wyznaczony procesor przesyła pozostałym optymalny wektor sterowania $\tilde{u}_{k+1}^{(\tau)[v_k]}$.

Po wyznaczeniu optymalnych sterowań dla każdego węzła v_k w k-tym przedziale czasowym, procesory wymieniają między sobą czasy wykonywania obliczeń, niezbędne dla wyznaczenia nowych wag procesorów w metodzie równoważenia obciążenia dla kolejnych iteracji algorytmu.

Algorytm 5.6 Równoległa procedura wyznaczania sterowania optymalnego

Oznaczenia:

rand(ρ) — dla r-wymiarowego wektora $\rho \geqslant 0$ symbol rand(ρ) oznacza liczbę pseudolosową o rozkładzie równomiernym w hiperkostce $[-\rho_{(1)}, \rho_{(1)}] \times \ldots \times [-\rho_{(r)}, \rho_{(r)}]$.

$\mathrm{Proj}_U(u)$ — operacja rzutowania wektora $u \in \mathbb{R}^r$ na zbiór sterowań dopuszczalnych U; zachodzi

$$[\mathrm{Proj}_U(u)]_{(j)} = \begin{cases} u_{\min(j)} & \text{jeżeli } u_{(j)} < u_{\max(j)} \\ u_{\max(j)} & \text{jeżeli } u_{(j)} > u_{\max(j)} \\ u_{(j)} & \text{w przeciwnym razie} \end{cases} \tag{5.106}$$

Dane wejściowe:

p — unikalny identyfikator każdego procesora,
K — liczba etapów,
$\bar{L}$ — wielkość siatki w przestrzeni stanów,
$\bar{I}$ — wielkość siatki w przestrzeni sterowań,
$\tau_{\max}$ — maksymalna liczba iteracji,
$\bar{u}^{(0)} = (\bar{u}_1^{(0)}, \ldots, \bar{u}_K^{(0)})$ — początkowa strategia sterowania,
$\rho^{(1)} \in \mathbb{R}_+^r$ — wektor określający promień poszukiwań sterowań po poszczególnych współrzędnych,

Dane wyjściowe:

$\bar{u}$ — optymalna strategia sterowania.

Krok 0: (Inicjalizacja)

Ustaw $\tau \leftarrow 1$. Jeżeli $p = 1$, to wygeneruj ziarna generatorów liczb pseudolosowych i prześlij je do pozostałych procesorów. Jeżeli $p \neq 1$, to odbierz ziarno generatora liczb pseudolosowych od procesora $p = 1$.

Krok 1: (Tworzenie przestrzeni stanu)
Każdy procesor p ($p = 1, \ldots, P$) wykonuje następujące czynności:

Faza 1: (Tworzenie przestrzeni stanu)
Ustawia $V_0^{(\tau)} \leftarrow \{v_0\}$.
Dla $k = 0, \ldots, K-1$ wyznacza kolejno:

$$\begin{aligned} \xi_{k+1}^{\ell(\tau)} &= \mathrm{rand}(\rho^{(\tau)}), \\ \tilde{u}_{k+1}^{\ell(\tau)} &= \mathrm{Proj}_U(\bar{u}_{k+1}^{(\tau)} + \xi_{k+1}^{\ell(\tau)}), \\ v_{k+1}^{\ell(\tau)} &= F(v_k^{\ell(\tau)}, \tilde{u}_{k+1}^{\ell(\tau)}), \end{aligned}$$

gdzie: $\ell = 1, \ldots, \bar{L}_p$, tworząc w ten sposób siatki przestrzeni stanów

$$V_{k+1,p}^{(\tau)} = \{v_{k+1,p}^{1(\tau)}, \ldots, v_{k+1,p}^{\bar{L}_p(\tau)}\},$$

Faza 2: (Wymiana danych pomiędzy procesorami)
Procesory wymieniają się utworzonymi siatkami przestrzeni stanu $V_{k+1,p}^{(\tau)}$. Operacja powoduje utworzenie globalnej siatki przestrzeni stanu, która jest przechowywana na każdym procesorze

$$V_{k+1}^{(\tau)} = V_{k+1,1}^{(\tau)} \cup \ldots \cup V_{k+1,p}^{(\tau)}.$$

Krok 2: (Generacja siatek sterowania z równoczesnym rozwiązywaniem zadania programowania dynamicznego)
Każdy procesor p ($p = 1, \ldots, P$) dla każdego $v_K \in V_K^{(\tau)}$ ustawia $J_K(v_K) = g_K(v_K)$, a następnie kolejno dla $k = K-1, \ldots, 0$ wyznacza

Faza 1:
Dla każdego $v_k \in V_k^{(\tau)}$ oblicz

$$\begin{aligned} \xi^{i(\tau)} &= \mathrm{rand}(\rho^{(\tau)}), \\ u_{k+1,p}^{i(\tau)} &= \mathrm{Proj}_U(\bar{u}_{k+1}^{(\tau)} + \xi^{i(\tau)}), \end{aligned}$$

gdzie: $i = 1, \ldots, \bar{I}_p$, otrzymując dla każdego v_k siatkę sterowań

$$\widetilde{U}_{k+1,p}^{(\tau)[v_k]} = \{u_{k+1,p}^{1(\tau)}, \ldots, u_{k+1,p}^{\bar{I}_p(\tau)}\}.$$

Faza 2:
Dla każdego $v_k \in V_k^{(\tau)}$ oblicz

$$J_{k,p}(v_k) = \min_{u_{k+1,p} \in \widetilde{U}_{k+1,p}^{(\tau)[v_k]}} \{g_k(v_k, u_{k+1,p}) + J_{k+1}(\tilde{F}_k(v_k, u_{k+1,p}))\}.$$

przy czym punkt minimum w wyrażeniu po prawej stronie dla każdego procesora zapamiętaj jako $\tilde{u}_{k+1,p}^{(\tau)[v_k]}$.

Faza 3:

Dla każdego $v_k \in V_k^{(\tau)}$ procesory wymieniają się wartościami wskaźników jakości $J_{k,p}(v_k)$.

Faza 4:

Dla każdego $v_k \in V_k^{(\tau)}$, każdy z procesorów określa, który spośród nich wyznaczył optymalną wartość wskaźnika jakości

$$\hat{\mathrm{p}} = \arg \min_{i=1,\ldots,P} J_{k,i}(v_k).$$

Faza 5:

Jeżeli $p = \hat{\mathrm{p}}$, to wyślij do pozostałych procesorów aktualne przybliżenie optymalnego wektora sterowań $\tilde{u}_{k+1,\hat{\mathrm{p}}}^{(\tau)[v_k]}$.

Jeżeli $p \neq \hat{\mathrm{p}}$, to odbierz optymalny wektor sterowań od węzła $\hat{\mathrm{p}}$. Przyjmij

$$\tilde{u}_{k+1}^{(\tau)[v_k]} \leftarrow \tilde{u}_{k+1,\hat{\mathrm{p}}}^{(\tau)[v_k]}.$$

Faza 6:

Po wyznaczeniu i wymianie pomiędzy procesorami sterowań optymalnych $\tilde{u}_{k+1,\hat{\mathrm{p}}}^{(\tau)[v_k]}$ dla każdego węzła $v_k \in V_k^{(\tau)}$, następuje wymiana czasów wykonywania obliczeń, a następnie określenie wartości współczynników równoważenia obciążenia w przez każdy procesor (zob. rozdz. 5.3.8) na podstawie odebranych czasów obliczeń.

Krok 3: (Wyznaczenie optymalnej strategii sterowania)

Każdy procesor p ($p = 1, \ldots, P$) tworzy ciąg sterowań optymalnych

$$\bar{U}^{(\tau+1)} = \{\bar{u}_1^{(\tau+1)}, \ldots, \bar{u}_k^{(\tau+1)}\}.$$

wg reguły

$$\bar{u}_{k+1}^{(\tau+1)} = \tilde{u}_{k+1}^{(\tau)[v_k^{\tau+1}]}, \quad k = 0, \ldots, K-1,$$

przy czym

$$\begin{aligned} \bar{v}_0^{(\tau+1)} &= \bar{v}_0 \\ \bar{v}_{k+1}^{(\tau+1)} &= \widetilde{F}_k\Big(\bar{v}_k^{(\tau+1)}, \bar{u}_{k+1}^{(\tau+1)}\Big), \quad k = 0, \ldots, K-1. \end{aligned} \tag{5.107}$$

Krok 4: (Warunek stopu)

Jeżeli $\tau = \tau_{\max}$, zakończ. W przeciwnym razie ustaw $\rho^{(k+1)} = \gamma\rho^{(k)}$ oraz $\tau \leftarrow \tau + 1$ i wróć do Kroku 1.

■

5.3.8 Równoważenie obciążenia

Podobnie jak w rozdz. 4.5.3, w celu zrównoważenia obciążenia procesorów zastosowano metodę adaptacyjnej ważonej faktoryzacji przedstawioną w rozdz. 3.8. Jest ona wykorzystywana do przydziału odpowiedniej liczby danych do obliczeń procesorom systemu równoległego w zależności od ich obciążenia. Miarą wykorzystywaną do pomiaru obciążenia jest czas obliczeń potrzebny przez każdy z procesorów do wykonania obliczeń związanych z wyznaczaniem wartości kryterium optymalności.

W Fazie 6 Kroku 3 Algorytmu 5.6 następuje określenie aktualnego obciążenia jednostek obliczeniowych. Bazując na informacjach otrzymywanych od innych procesorów, każdy z procesorów uaktualnia wartości, przechowywanych przez siebie, współczynników w dla każdego z procesorów. Dzięki temu, każdy z procesorów posiada identyczny wektor w przechowujący wagi procesorów dla metody równoważenia obciążenia.

Liczbę węzłów dyskretnej przestrzeni sterowań analizowaną przez każdy z procesorów wyznacza się na podstawie czasów oraz wielkości danych jakie zostały dotychczas przetworzone przez procesory liczące. Miarą wykorzystywaną do pomiaru obciążenia jest czas obliczeń jaki jest potrzebny przez każdy z procesorów do wykonania obliczeń związanych z wyznaczeniem wartości wskaźnika jakości J.

Na podstawie tych informacji wyliczany jest średni czas wykonania obliczeń na pojedynczej operacji atomowej — wyznaczenie wartości wskaźnika jakości J — dla każdego z procesorów liczących (zob. (3.27)). Wykonanie kolejnych obliczeń (3.28)–(3.29) pozwala na wyznaczenie wag w_p dla wszystkich procesorów liczących. Wagi o wartości powyżej 1 oznaczają, że procesor był wcześniej niedoszacowany pod względem ilosci obliczeń i należy zwiększyć jego obciążenie. Analogicznie, wartość wagi poniżej jedności oznacza przeszacowanie ilości obliczeń na danym procesorze i konieczność zmniejszenia jego obciążenia w kolejnej iteracji.

Metoda adaptacyjnej ważonej faktoryzacji zakłada, że wagi o wartości 1 dla każdego z procesorów określają równomierny podział danych pomiędzy procesory liczące. W celu łatwiejszej implementacji wyznaczone wagi poddane zostają kolejnemu procesowi normalizacji, tak aby wagi o wartości $1/P$ określały pożądany podział równomierny, gdzie: P — liczba procesorów liczących. Zachodzi

$$\bar{w}_p = \frac{w_p}{\sum_{j=1}^{P} w_j}, \quad p = 1, \ldots, P \tag{5.108}$$

Odchyłki w górę, czyli $\bar{w}_p = 1/P$, określają niedoszacowanie możliwości obliczeniowych procesora w poprzednich iteracjach, a odchyłki w dół, czyli $\bar{w}_p < 1/P$, określają przeszacowanie możliwości obliczeniowych procesora w poprzednich iteracjach.

Na podstawie tak wyznaczonych wartości współczynników $\bar{w}_p$ następuje wyznaczenie wielkości przestrzeni sterowań jaka ma być analizowana przez każdy z procesorów liczących. W przypadku, gdy wyznaczone wielkości przestrzeni sterowań nie są liczbami całkowitymi, reszty wynikające z podziału przestrzeni sterowań pomiędzy procesory liczące są sumowane, a następnie dodawane dla procesora liczącego, który przeprowadza obliczenia najszybciej.

5.4 Uzyskane wyniki

Jasnym jest, że wyznaczanie optymalnych trajektorii czujników ruchomych jest problemem skomplikowanym. Duża skala problemów, złożoność rozważanych zagadnień oraz wielomodalność minimalizowanych funkcji powodują, że wykorzystanie algorytmów równoległych jest jedną z dróg prowadzących do szybszego otrzymania wyników badań. Przeformułowanie problemu poprzez parametryzację trajektorii po których poruszają się czujniki, jak również dogłębna analiza zagadnienia i uwzględnienie dynamiki pojazdów przenoszących czujników powodują, że obliczenia stają się bardzo skomplikowane i długotrwałe. Przedstawione modele i przykłady pozwalają na zobrazowania skali problemów oraz stopnia ich skomplikowania. W przypadku zastosowań w praktyce, należy się spodziewać znacznie bardziej skomplikowanych problemów, a co za tym idzie — potrzeby wykorzystania znacznie większych mocy obliczeniowych dostarczanych przez nowoczesne środowiska obliczeniowe.

Obliczenia weryfikujące proponowane techniki przeprowadzono na zielonogórskim klastrze obliczeniowym zbudowanym w ramach projektu CLUSTERIX [248]. Homogeniczny klaster tworzą cztery węzły zawierające po dwa 64-bitowe procesory *Intel Itanium* 1.4 GHz (zob. rozdz. 3.9). Obliczenia przeprowadzane były również na klastrze Sherwood znajdującym się w Poznańskim Centrum Superkomputerowo-Sieciowym. Klaster ten składa się z stu dziewięciu węzłów zawierających po dwa 64-bitowe procesory *Intel Itanium* 1.3 GHz, z piętnastu dwuprocesorowych węzłów zawierających po dwa 64-bitowe procesory *Intel Itanium* 1.4 GHz oraz z dwóch węzłów zawierających po dwa procesory *Intel Xeon* 2.4 GHz. Implementacje przedstawionych metod oraz algorytmów napisano w języku Fortran 95 z zastosowaniem kompilatora Intel®Fortran Compiler oraz biblioteki Intel Math Kernel Library. Jako środowisko równoległe wykorzystano interfejs MPI [163] w postaci biblioteki *mpich*.

5.4.1 Testy Algorytmu 5.3

W badaniach dotyczących zachowania algorytmu opartego na parametryzacji trajektorii mobilnych węzłów sieci sensorycznej rozważono ponownie Przykład 5.2. Każdy procesor obliczeniowy uruchomił algorytm z losowo wygenerowanymi trajektoriami ruchu czujników.

W tabeli 5.1 przedstawiono czasy obliczeń jakie uzyskano podczas uruchamiania Algorytmu 5.3 na ośmioprocesorowym klastrze homogenicznym dla różnych wartości parametrów γ_1 i γ_2. Uzyskane przyspieszenie i efektywność algorytmu przedstawiono w tab. 5.2 oraz na rys. 5.9. Można zauważyć zwiększające się przyspieszenie oraz efektywność wraz ze wzrostem parametrów γ_1 i γ_2. Jest to spowodowane zwiększaniem się nakładu obliczeniowego bez znaczącego zwiększania opóźnień w przesyłaniu informacji pomiędzy węzłami. Dodatkowo algorytm nie posiada części sekwencyjnej zależnej od wielkości zadania, co pozwala na uzyskiwanie dużych przyspieszeń dla bardziej zaawansowanych problemów optymalnej obserwacji. W takich zagadnieniach parametrom γ_1 i γ_2 można nadać dwie wartości, tak aby eksploracja otoczenia

γ_1	γ_2	Liczba procesorów obliczeniowych P						
		1	2	3	4	5	6	7
50	50	0:42:00	0:23:13	0:17:47	0:14:10	0:11:25	0:10:28	0:09:26
210	210	2:36:04	1:20:45	0:55:47	0:42:06	0:35:11	0:29:54	0:27:33
420	420	5:03:22	2:34:05	1:43:19	1:18:17	1:04:07	0:54:22	0:48:23

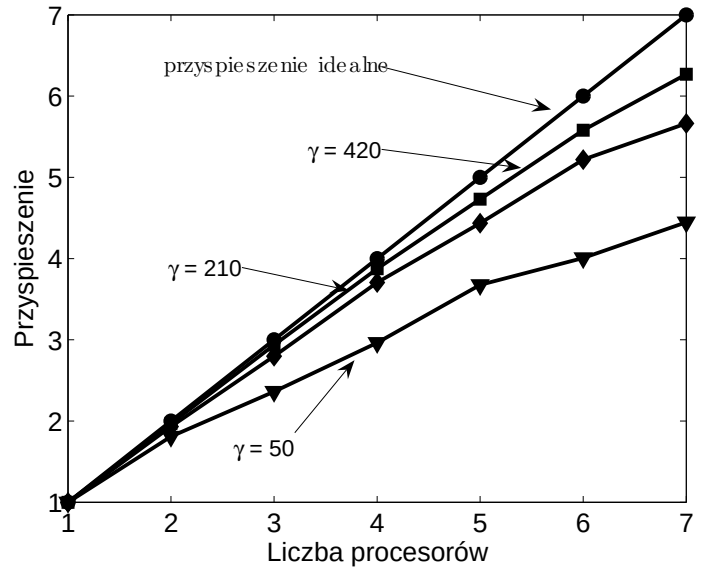
Tabela 5.1: Czasy obliczeń [h:min:s] uzyskane dla Algorytmu 5.3 oraz Przykładu 5.2.

minimum lokalnego w celu znalezienia nowej doliny była jak najdokładniejsza.

γ_1	γ_2		Liczba procesorów obliczeniowych P						
			1	2	3	4	5	6	7
50	50	$S(p)$	1.00	1.80	2.36	2.96	3.67	4.00	4.44
		$E(p)$	1.00	0.90	0.78	0.74	0.73	0.66	0.63
210	210	$S(p)$	1.00	1.93	2.79	3.70	4.43	5.21	5.66
		$E(p)$	1.00	0.96	0.93	0.92	0.88	0.86	0.80
420	420	$S(p)$	1.00	1.96	2.93	3.87	4.73	5.57	6.26
		$E(p)$	1.00	0.98	0.97	0.96	0.94	0.92	0.89

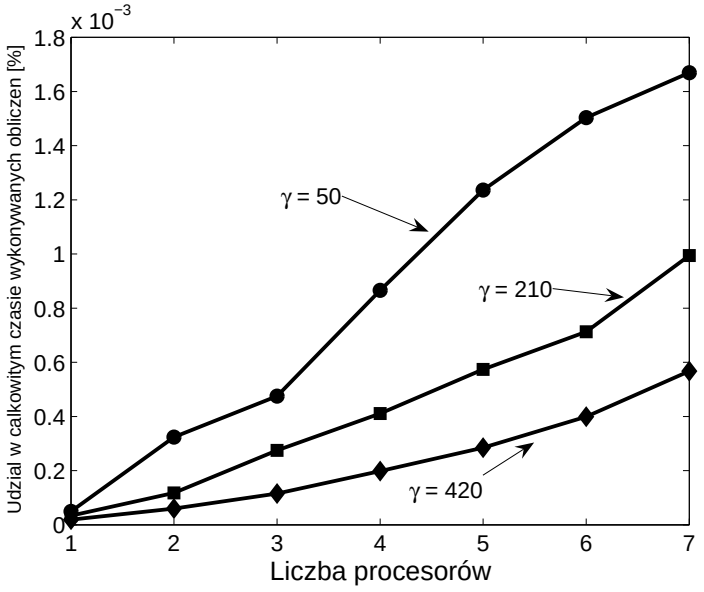
Tabela 5.2: Przyspieszenie i efektywność uzyskane dla Algorytmu 5.3 oraz Przykładu 5.2.

Procentowy udział komunikacji w ogólnym czasie obliczeń przedstawiono na rys. 5.10. Jak można zauważyć, udział komunikacji w ogólnym czasie wykonywania obliczeń nie jest wysoki, stanowiąc mniej niż dziesiątą część promila. Wykonywanie obliczeń dla bardziej skomplikowanych zagadnień wymagać będzie przesłania większych ilości informacji, jednakże czas wykonywania obliczeń również się zwiększy powodując, że udział opóźnień związanych z komunikacją nadal będzie niewielki. Należy jednak mieć na uwadze, że pomimo stosunkowo niewielkiego udziału komunikacji w ogólnym czasie obliczeń, algorytm ten, podobnie jak Algorytm 4.5, jest wrażliwy na zwiększanie liczby procesorów. Fakt ten jest spowodowany koniecznością wymiany danych pomiędzy procesorem–nadzorcą, a procesorami liczącymi. W



Rysunek 5.9: Przyspieszenie uzyskane dla Algorytmu 5.3 oraz Przykładu 5.2.

przypadku zbyt dużej liczby procesorów liczących może wystąpić sytuacja, gdzie procesor-nadzorca zostanie zbyt obciążony obsługą żądań, co spowoduje w konsekwencji opóźnienia w transmisji danych do procesorów liczących. Pewnym rozwiązaniem tego problemu jest budowa heterogenicznego środowiska obliczeniowego, w którym najszybszy procesor będzie pełnić rolę nadzorcy.



Rysunek 5.10: Procentowy udział czasu komunikacji w ogólnym czasie wykonywania obliczeń dla Algorytmu 5.3 oraz Przykładu 5.2.

5.4.2 Testy Algorytmu 5.6

W badaniach dotyczących planowania D-optymalnych trajektorii mobilnych węzłów sieci sensorycznej z uwzględnieniem dynamiki pojazdów przenoszących czujniki rozważono Przykład 5.3. Algorytm 5.6 uruchamiano zarówno na klastrze homogenicznym, jak i na klastrze heterogenicznym.

Uruchomienie algorytmu na ośmioprocesorowym klastrze homogenicznym dla różnych wartości parametru $\bar{I}$ pozwoliło na uzyskanie czasów obliczeń przedstawionych w tab. 5.3. Uzyskane przyspieszenie i efektywność algorytmu przedstawiono w tab. 5.4 oraz na rys. 5.3. Można zauważyć, że obliczenia dla ośmiuprocesorów przeprowadzane są z efektywnością bliską jedności dla parametru $\bar{I} = 240$.

	Liczba procesorów obliczeniowych P							
$\bar{I}$	1	2	3	4	5	6	7	8
10	0:13:08	0:07:09	0:04:49	0:03:57	0:03:19	0:02:46	0:02:28	0:02:08
24	0:36:38	0:19:42	0:13:25	0:10:02	0:08:08	0:06:45	0:05:43	0:05:05
120	3:10:40	1:39:46	1:06:12	0:51:17	0:40:18	0:33:31	0:28:47	0:25:29
240	6:40:49	3:20:59	2:15:59	1:43:28	1:20:18	1:07:10	0:57:38	0:52:10

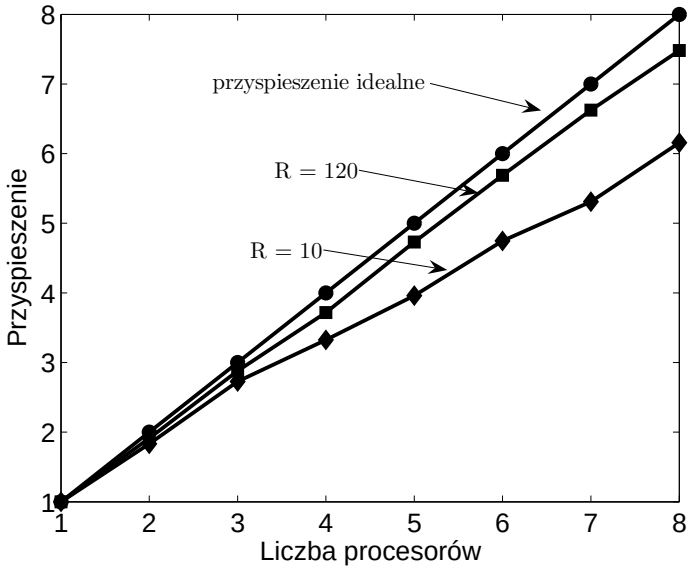
Tabela 5.3: Czasy obliczeń [h:min:s] uzyskane dla Algorytmu 5.6 oraz Przykładu 5.3 (klaster homogeniczny).

Procentowy udział komunikacji w ogólnym czasie obliczeń przedstawiono na rys. 5.12. Na przedstawionym wykresie można zauważyć, że udział komunikacji pomiędzy procesorami maleje wraz ze wzrostem parametru sterującego dokładnością eksploracji przestrzeni sterowań. Należy jednak mieć na uwadze, że pomimo stosunkowo niewielkiego udziału komunikacji w ogólnym czasie obliczeń, algorytm ten, podobnie jak Algorytm 4.9, jest „wrażliwy" na zwiększanie liczby procesorów. W odróżnieniu od schematu komunikacji „nadzorca–podwładni", gdzie problemem może się stać przeciążenie procesora-nadzorcy, w przypadku schematu „każdy z każdym", problemem jest liczba połączeń koniecznych do przesłania informacji pomiędzy procesorami. Fakt ten jest spowodowany koniecznością wymiany danych przez wszystkie procesory pomiędzy sobą, w odróżnieniu od schematu komunikacji „nadzorca–podwładni", gdzie komunikacja następuje tylko pomiędzy procesorem–nadzorcą, a procesorami liczącymi. Schemat wymiany danych „każdy z każdym" wymaga wykonania $p \cdot (p-1)$ operacji przesłania danych, co w porównaniu do $p-1$ koniecznych połączeń dla schematu „nadzorca–podwładni", może mieć duże znaczenie dla dużej liczby procesorów.

W celu sprawdzenia skalowalności algorytmu dla większej liczby procesorów li-

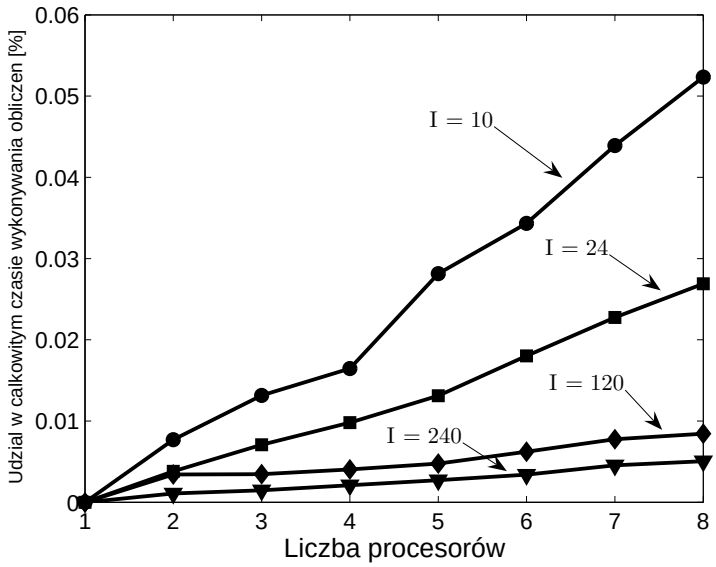
$\bar{I}$		Liczba procesorów obliczeniowych P							
		1	2	3	4	5	6	7	8
10	$S(p)$	1.00	1.61	2.73	3.32	3.96	4.75	5.71	6.16
	$E(p)$	1.00	0.92	0.90	0.83	0.79	0.79	0.76	0.76
24	$S(p)$	1.00	1.86	2.73	3.65	4.50	5.42	6.40	7.20
	$E(p)$	1.00	0.93	0.91	0.91	0.90	0.90	0.91	0.90
120	$S(p)$	1.00	1.91	2.88	3.71	4.73	5.68	6.62	7.48
	$E(p)$	1.00	0.95	0.96	0.92	0.94	0.94	0.94	0.93
240	$S(p)$	1.00	1.99	2.93	3.87	4.99	5.96	6.95	7.68
	$E(p)$	1.00	0.99	0.97	0.96	0.99	0.99	0.99	0.96

Tabela 5.4: Przyspieszenie i efektywność uzyskane dla Algorytmu 5.6 oraz Przykładu 5.3 (klaster homogeniczny).



Rysunek 5.11: Przyspieszenie uzyskane dla Algorytmu 5.6 oraz Przykładu 5.3 (klaster homogeniczny).

czących, przeprowadzono symulacje na na klastrze Sherwood znajdującym się w Poznańskim Centrum Superkomputerowo-Sieciowym. Uzyskane czasy obliczeń dla



Rysunek 5.12: Procentowy udział czasu komunikacji w ogólnym czasie wykonywania obliczeń dla Algorytmu 5.6 oraz Przykładu 5.3 (klaster homogeniczny).

parametru $\bar{I} = 240$ przedstawiono w tabeli 5.5, a przyspieszenie oraz efektywność zamieszczono w tabeli 5.6 oraz na rys. 5.13. Jak można zauważyć, dla większej liczby procesorów i stałego parametru $\bar{I} = 240$ algorytm zachowuje się zgodnie z prawem Amdahla (zob. rozdz. 3.7.2). Jest to spowodowane stopniowym zwiększaniem udziału części sekwencyjnej algorytmu gdyż dekompozycja zagadnienia jest przeprowadzana na większej liczbie procesorów.

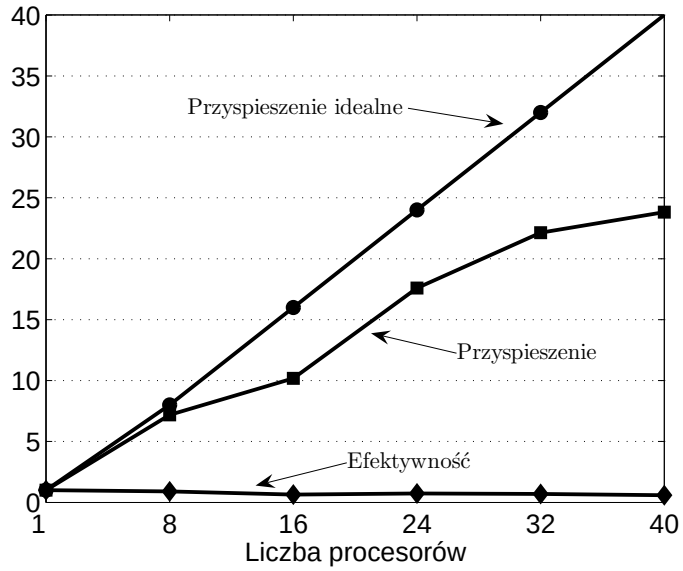
	Liczba procesorów obliczeniowych P					
n	1	8	16	24	32	40
240	0:57:11	0:07:58	0:05:37	0:03:15	0:02:35	0:02:24

Tabela 5.5: Czasy obliczeń [h:min:s] uzyskane dla Algorytmu 5.6 oraz Przykładu 5.3 (klaster homogeniczny).

W badaniach dotyczących poszukiwania optymalnych trajektorii czujników ruchomych z uwzględnieniem dynamiki poruszających się pojazdów dodatkowo zbadano wpływ poziomu eksploracji przestrzeni sterowań $\bar{I}$ na czas obliczeń w zależności od liczby procesorów liczących. Wyniki badań przedstawiono w tab. 5.7 (puste miejsca w tabeli oznaczają brak przeprowadzonych badań dla podanych parametrów). Pozwalają one na stwierdzenie, że algorytm jest dobrze skalowalny dla zwiększającej się liczby procesorów i jednoczesnym wzroście rozmiaru rozważanego zagadnienia.

n		Liczba procesorów obliczeniowych P					
		1	8	16	24	32	40
240	$S(p)$	1.00	7.18	10.18	17.6	22.14	23.83
	$E(p)$	1.00	0.90	0.64	0.73	0.69	0.60

Tabela 5.6: Przyspieszenie i efektywność uzyskane dla Algorytmu 5.6 oraz Przykładu 5.3 (klaster homogeniczny).



Rysunek 5.13: Przyspieszenie i efektywność uzyskane dla Algorytmu 5.6 oraz Przykładu 5.3 (klaster homogeniczny).

Czas wykonywania obliczeń przy proporcjonalnym zwiększaniu liczby procesorów wraz ze wzrostem rozmiaru zagadnienia są porównywalne. Czas pracy jednego procesora dla rozwiązania Przykładu 5.3 z parametrem $\bar{I} = 10$ wynosi 176 sekund. Przy proporcjonalnym zwiększaniu rozmiaru problemu oraz liczby procesorów uzyskujemy zbliżony czas pracy, tj. dla $\bar{I} = 120$ i 12 procesorów czas wynosi 189 sekund, dla $\bar{I} = 240$ i 24 procesorów — 185 sekund, oraz dla $\bar{I} = 360$ i 36 procesorów — 181 sekund. Te obserwacje uzasadniają tezę, że wielkość środowiska obliczeniowego powinna być dostosowana do wielkości rozwiązywanego zagadnienia. W przypadku uruchamiania problemów małej skali w dużych środowiskach obliczeniowych, moc procesorów pozostanie niewykorzystana. Większy udział w czasie obliczeń będą odgrywać czasy opóźnień w transmisji danych pomiędzy procesorami oraz czas oczekiwania procesorów na przesłanie danych.

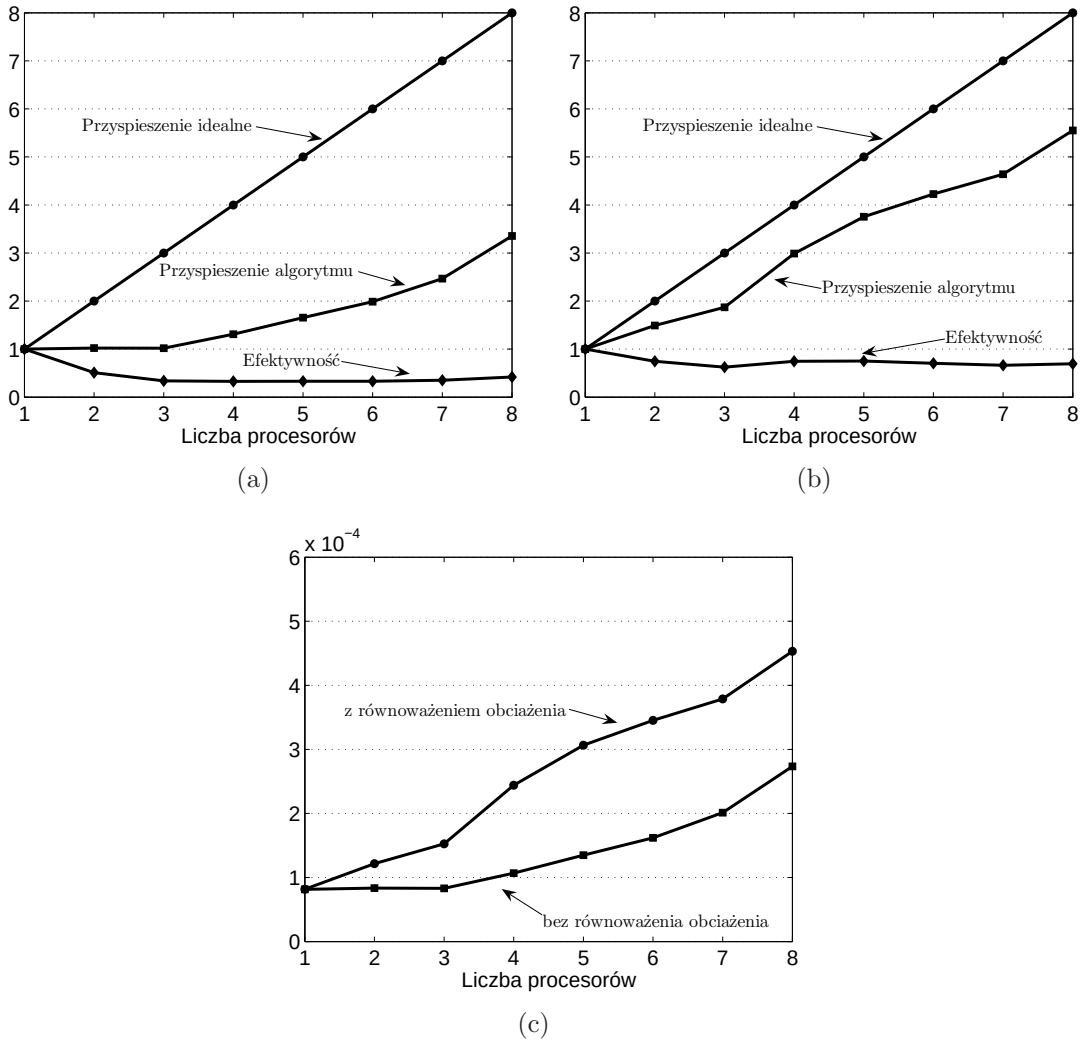
$\bar{I}$	Liczba procesorów obliczeniowych P			
	1	12	24	36
10	0:02:56	—:—:—	—:—:—	—:—:—
24	0:05:43	0:01:08	0:00:46	—:—:—
120	0:26:42	0:03:09	0:02:03	—:—:—
240	0:57:11	0:04:52	0:03:05	—:—:—
360	—:—:—	—:—:—	—:—:—	0:03:01

Tabela 5.7: Czasy obliczeń [h:min:s] uzyskane dla Algorytmu 5.6 oraz Przykładu 5.3 w zależności od parametru $\bar{I}$ (klaster homogeniczny).

Uruchomienie algorytmu na klastrze heterogenicznym pozwoliło na uzyskanie wyników przedstawionych na rys. 5.14. Jak można zauważyć, zastosowanie metod równoważenia obciążenia pozwala na uzyskanie lepszego przyspieszenia, a co za tym idzie — lepszej wydajności algorytmu w przypadku uruchamiania algorytmu na klastrze posiadającym niejednorodną moc obliczeniową. Uruchomienie algorytmu bez równoważenia obciążenia powoduje, że czas obliczeń jest zdeterminowany przez obliczenia prowadzone na najwolniejszym węźle. Procesory szybciej wykonujące obliczenia muszą czekać na procesory wolniejsze w miejscach algorytmu, gdzie następuje wymiana danych oraz wspólne rozpoczęcie pracy wszystkich procesorów w dalszych etapach pracy algorytmu. Dla algorytmu z wyłączoną metodą równoważenia obciążenia uzyskano średnią wariancję udziału procesorów w obliczeniach $\Gamma(\pi) = 14.85$, a dla algorytmu z włączoną metodą równoważenia obciążenia współczynnik wynosił $\Gamma(\pi) = 0.012$ (zob. rozdz. 3.8). Wartości parametru $\Gamma(\pi)$ mają bezpośrednie przełożone na czas pracy algorytmu. W rozważanym przypadku algorytm bez włączonej metody równoważenia obciążenia dla parametru $\bar{I} = 120$ i 8 procesorów pracował przez 60 min i 56 sekund, a przy włączonej metodzie równoważenia obciążenia czas pracy algorytmu skracał się do 36 min i 47 sekund co daje przyspieszenie obliczeń o około 39%.

5.5 Podsumowanie

W rozdziale zaproponowano metody optymalnego planowania trajektorii mobilnych węzłów sieci sensorycznych. Zaproponowane podejścia do przedstawionego problemu opierają się na modelowaniu ruchu pojazdów przenoszących czujniki z wykorzystaniem funkcji sklejanych oraz na sformułowaniu problemu jako zagadnienia sterowania optymalnego, który pozwala na jawne określenie ograniczeń dotyczących np.



Rysunek 5.14: Uzyskane przyspieszenie, efektywność oraz wydajność na klastrze heterogenicznym dla Algorytmu 5.6 oraz Przykładu 5.3: (a) bez równoważenia obciążenia, (b) z równoważeniem obciążenia, (c) porównanie wydajności.

dynamiki poruszających się pojazdów, ich przyspieszenia, zużycia energii i wielu innych.

Zaproponowane metody opierają się na metodach optymalizacji ciągłej. Rozważane zagadnienia należą do klasy problemów silnie nieliniowych o dużej liczbie zmiennych decyzyjnych, stąd zaproponowano zastosowanie zaawansowanego aparatu matematycznego w postaci algorytmu tunelowego oraz iteracyjnego programowania dynamicznego.

Metoda optymalnej parametryzacji trajektorii z wykorzystaniem funkcji sklejanych opierająca się na algorytmie tunelowym, pomimo wykorzystywania uproszczo-

nych równań ruchu mobilnych węzłów sieci sensorycznej, pozwoliła na uzyskanie satysfakcjonujących rezultatów. Rozważony przykład pozwolił zademonstrować problem parametryzacji trajektorii dwóch mobilnych sensorów. Już dla tak prostego przykładu zauważalne jest znaczne skomplikowanie opisu problemu optymalnej obserwacji, co wiąże się przede wszystkim ze znaczną wymiarowością zagadnienia.

Zrównoleglenie przedstawionej metody pozwoliło na efektywne przeprowadzanie obliczeń w środowisku klastra obliczeniowego. Malejący wpływ opóźnień związanych z komunikacją pomiędzy procesorami, wraz ze wzrostem dokładności eksploracji przestrzeni w poszukiwaniu tuneli umożliwiających określanie nowych minimów lokalnych, pozwala mieć nadzieję na efektywne przeprowadzanie obliczeń w środowiskach równoległych posiadających znacznie większą liczbę procesorów dla znacznie bardziej skomplikowanych zagadnień optymalnej obserwacji.

Druga z przedstawionych metoda rozwiązywania zagadnienia optymalnej obserwacji z wykorzystaniem mobilnych węzłów sieci sensorycznej pozwala na stosowanie bardziej skomplikowanych równań ruchu, uwzględniających dynamikę poruszających się pojazdów. Przedstawienie problemu optymalnej obserwacji jako zagadnienia sterowania optymalnego umożliwia wyznaczanie optymalnych trajektorii mobilnych węzłów sieci sensorycznej z uwzględnieniem czynników fizycznych związanych z samym ruchem pojazdów, ich dynamiki, przyspieszenia, zużycia energii oraz wielu innych parametrów związanych z pracą w trakcie przenoszenia czujników podczas obserwacji obiektów z czasoprzestrzenną dynamiką. Zaproponowanie metody rozwiązania zagadnienia sterowania optymalnego w postaci iteracyjnego programowania dynamicznego pozwala na zaadaptowanie przedstawionej metody w rzeczywistych eksperymentach. Zalety iteracyjnego programowania dynamicznego w stosunku do klasycznego programowania dynamicznego umożliwiają znaczne zmniejszenie wymiarowości problemów, redukując tym samym znaczne wymagania systemowe dotyczące środowisk obliczeń równoległych.

Zrównoleglenie metody uwzględniającej dynamikę mobilnych węzłów sieci sensorycznej pokazało, że możliwie jest efektywne przeprowadzanie obliczeń związanych z rozważanymi problemami w środowiskach obliczeń równoległych. Przeprowadzone badania oraz eksperymenty dowodzą słuszności wybrania algorytmów równoległych jako metody pozwalającej na efektywną optymalizację obserwacji obiektów z czasoprzestrzenną dynamiką. W badaniach uzyskano zadowalające rezultaty przyspieszenia obliczeń i związanej z tym efektywności w przypadku uruchamiania algorytmów na klastrach obliczeniowych składających się z kilkudziesięciu procesorów. Dodatkowo, przeprowadzone eksperymenty pokazały elastyczność wybranej metody zrównoleglania w przypadku uruchamiania algorytmu w heterogenicznych środowiskach obliczeniowych, co oznacza, że zaproponowana metoda pozwala na efektywne wykorzystanie mocy obliczeniowej znajdujących się w nich procesorów.

6

Zakończenie

6.1 Podsumowanie

Niniejsza praca została poświęcona zagadnieniom związanym z zastosowaniem technik przetwarzania równoległego do wyznaczania optymalnych strategii pomiarowych z wykorzystaniem sieci sensorycznych w procesie obserwacji układów z czasoprzestrzenną dynamiką. W rozprawie rozważono zagadnienia optymalnej obserwacji z wykorzystaniem:

- sieci czujników stacjonarnych,
- sieci czujników skanujących,
- sieci czujników mobilnych.

Problem optymalnej obserwacji z wykorzystaniem sieci czujników stacjonarnych oraz skanujących został sformułowany jako zagadnienie optymalizacji dyskretnej. W przypadku optymalnego planowania trajektorii mobilnych węzłów sieci sensorycznej zaproponowano metody wykorzystujące optymalizację ciągłą lub techniki optymalnego sterowania.

Zagadnienia optymalnej aktywacji czujników stacjonarnych oraz skanujących zostały przedstawione jako problemy kombinatoryczne, w których spośród wielu czujników rozmieszczonych w obszarze funkcjonowania badanego zjawiska należy wybrać zadaną liczbę czujników. W celu ich rozwiązania zaproponowano metody adaptujące znane algorytmy optymalizacji dyskretnej, tj. GRASP oraz Tabu-Search.

Opracowanie przedstawionych metod zostało poprzedzone wnikliwą analizą zagadnień i problemów, które pojawiają się przy tak opisanym zagadnieniu. Ważnym założeniem, determinującym sposób konstrukcji przedstawionych metod była konieczność uwzględnienia zależności pomiędzy pomiarami wykonywanymi przez czujniki. Zagadnienie korelacji pomiarów wykonywanych przez czujniki jest zjawiskiem

naturalnym, a proponowane metody powinny uwzględniać ich występowanie, tak aby efekty symulacji z ich wykorzystaniem miały uzasadnienie stosowalności w rzeczywistym eksperymencie.

Jednak takie podejście powoduje znaczne zwiększenie nakładu obliczeń, a przez to wydłużenie czasu symulacji. Zaproponowanie metod obliczeniowych opierających się na algorytmie wymiany czujników w planie eksperymentu umożliwiło redukcję czasochłonnych operacji. Metoda ta wykorzystuje formułę Shermana-Morrisona-Woodbury'ego i umożliwia zmniejszenie liczby czasochłonnych operacji wymaganych przy wyznaczaniu odwrotności macierzy kowariancji pomiarów wykonywanych przez czujniki, a w konsekwencji — przy wyznaczaniu macierzy informacyjnej. Zastosowanie metody redukcji czasochłonnych operacji pozwala na znaczne skrócenie czasu obliczeń jeszcze przed odwołaniem się do metod programowania równoległego, co przedstawiono w formie przykładu numerycznego.

W rozprawie skupiono się na przedstawieniu równoległych metod rozwiązania problemu optymalnej obserwacji z zastosowaniem sieci czujników stacjonarnych i skanujących. Już na etapie analizy rozważanych zagadnień skupiono się przede wszystkim na takim zaprojektowaniu metod, aby umożliwić ich efektywną pracę w środowisku wieloprocesorowym. W celu zrównoleglenia metod optymalnej aktywacji czujników stacjonarnych oraz skanujących zastosowano dekompozycję przestrzeni poszukiwań. Takie podejście jest uzasadnione występowaniem największego nakładu obliczeń podczas analizy przestrzeni rozwiązań i jest związane z koniecznością wielokrotnego wyznaczania wartości kryterium optymalności dla różnych konfiguracji aktywnych czujników.

Przedstawione równoległe metody rozwiązania zagadnienia optymalnej obserwacji wykorzystują różne sposoby i schematy komunikacji pomiędzy procesorami. Schemat nadzorca/podwładni ustanawia jeden z procesorów jako procesor główny posiadający całkowitą wiedzę na temat aktualnego stan algorytmu. Reszta z procesorów wykonuje ściśle wyspecyfikowane operacji na podstawie komunikatów odbieranych z procesora nadzorcy. Jak się okazało, taki schemat komunikacji powoduje dość znaczną liczbę przesyłanych informacji pomiędzy procesorem nadzorcą, a procesorami liczącymi i może w konsekwencji powodować opóźnienia w przesyłaniu danych z procesora głównego do procesorów podwładnych. Również w skrajnych przypadkach wydajność procesora głównego staje się wąskim gardłem systemu równoległego rozwiązującego pewien problem. W przypadku schematu komunikacji „każdy z każdym" procesory wymieniają między sobą znacznie mniejsze ilości danych. Mniejsza ilość danych do przesłania pomiędzy procesorami jest konsekwencją umiejscowienia procedur realizujących cały algorytm we wszystkich procesorach liczących (w odróżnieniu od schematu nadzorca/podwładni). Taki schemat wymiany danych powoduje, że ilość przesyłanych danych dla niewielkiej liczby procesorów nie zmniejsza efektywności algorytmu, jednak wraz ze wzrostem liczby jednostek liczących, ilość danych do przesłania pomiędzy procesorami rośnie. Szybki wzrost może w konsekwencji prowadzić do dość dużych opóźnień w przesyłaniu danych pomiędzy procesorami i powodować wydłużenie czasu bezczynności procesorów.

Przedstawione wyniki symulacji wskazują na efektywność zaproponowanych rozwiązań. Opracowane metody wykazują cechę skalowalności, gdzie wraz ze wzrostem rozmiaru problemu oraz liczby procesorów obliczenia są przeprowadzane efektywnie pod względem uzyskiwanych przyspieszeń. Dodatkowe zastosowanie metody redukcji czasochłonnych operacji pozwoliło na wielokrotne skrócenie czasu obliczeń jeszcze przed odwołaniem się do metod programowania równoległego. Badania dotyczyły optymalnej aktywacji zarówno kilkudziesięciu, jak i kilkuset czujników stacjonarnych oraz skanujących podczas symulacji procesu obserwacji zjawiska dyfuzji zanieczyszczeń atmosferycznych. Zrównoleglenie zaproponowanych algorytmów pozwoliło na znaczące przyspieszenie obliczeń, a co za tym idzie — efektywne wykorzystanie systemu równoległego. Przeprowadzone badania z zastosowaniem heterogenicznych środowisk równoległych pokazały zalety metod równoważenia obciążenia. Ich adaptacja pozwoliła na znacznie efektywniejsze wykorzystanie środowisk równoległych, a konsekwencji — na skrócenie czasów wykonywania symulacji.

W przypadku obserwacji z wykorzystaniem czujników ruchomych problem sformułowano jako zagadnienie poszukiwania optymalnych trajektorii mobilnych węzłów sieci sensorycznej. Na etapie wstępnej analizy przedstawionego problemu założono, że opracowane zostaną dwie metody: jedna opisująca trajektorię jako krzywe parametryczne oraz druga — umożliwiającą uwzględnienie dynamiki pojazdów przenoszących czujniki.

W celu parametryzacji trajektorii zaproponowano wykorzystanie funkcji (splajnów) sklejanych sześciennych. Modelowanie trajektorii ruchu pojazdów przenoszących czujniki z wykorzystaniem splajnów sześciennych stanowi pewne uproszczenie zjawisk jakie występują w rzeczywistych warunkach przeprowadzania eksperymentu. Parametryzacja pozwala na opisanie kształtu ścieżek oraz na wyznaczenie uproszczonego harmonogramu poruszania się pojazdów. Jednak, takie podejście nie pozwala na określenie w jawny sposób ograniczeń dotyczących np. dynamiki poruszających się pojazdów, ich przyspieszenia, zużycia energii oraz wielu innych zjawisk dynamicznych.

Reprezentacja trajektorii czujników jako krzywych parametrycznych prowadzi do konieczności wyznaczania kryterium optymalności opartego na funkcji zależnej od dużej liczby zmiennych decyzyjnych. Znaczna wymiarowość problemu dla stosunkowo prostych zagadnień wymagała opracowania metody pozwalającej na efektywną optymalizację globalną wielowymiarowych i silnie nieliniowych funkcji celu. Do rozwiązania tak sformułowanego zagadnienia, zaproponowano schemat algorytmu tunelowego. Zaproponowana metoda wykorzystała zalety tego algorytmu pozwalając na optymalizację wielowymiarowych funkcji celu, umożliwiając efektywne wyznaczenie optymalnych trajektorii mobilnych węzłów sieci sensorycznej. Jako element składowy algorytmu tunelowego zaproponowano bezgradientową metodę minimalizacji Rosenbrocka.

W kontekście opracowania metody wykorzystującej środowiska wieloprocesorowe, analiza zagadnienia poszukiwania optymalnych trajektorii mobilnych węzłów sieci sensorycznej z zastosowaniem krzywych parametrycznych, pokazała konieczność

wprowadzenia modyfikacji schematu zrównoleglania algorytmu tunelowego, spotykanego w literaturze. Skomplikowana postać funkcji celu, a co za tym idzie — wymagany duży nakład obliczeniowy podczas jej wyznaczania — wymusiły opracowanie odpowiedniego protokołu komunikacji pomiędzy procesorem nadzorującym obliczenia, a procesorami liczącymi. Przeprowadzone badania z wykorzystaniem opracowanej równoległej metody wyznaczania optymalnych trajektorii opisanych za pomocą krzywych parametrycznych pozwoliły na uzyskanie zadowalającego przyspieszenia obliczeń. Wykorzystany schematu komunikacji nadzorca/podwładni oraz odpowiedni protokół komunikacyjny wymagają niewielkiego udziału komunikacji w ogólnym czasie wykonywania obliczeń. Uzyskane wyniki pozwoliły na stwierdzenie, że zaproponowana metoda pozwala na wyznaczanie optymalnych trajektorii oraz efektywne wykorzystanie mocy obliczeniowych środowisk równoległych wraz ze wzrostem złożoności rozważanego problemu.

Parametryczny opis trajektorii jest dość mocnym uproszczeniem rozważanych problemów. Opisywanie trajektorii w sposób nie uwzględniający fizycznych właściwości (np. inercji, ograniczeń prędkości i energii, itp.) zarówno samych czujników jak i urządzeń je przenoszących może powodować, że uzyskane rezultaty będą mało użyteczne w praktyce. Uwzględnienie wielu ograniczeń opisanych powyżej umożliwia drugie zaproponowane w niniejszej pracy podejście, w którym problem definiuje się jako zagadnienie sterowania optymalnego. Zastosowanie tej metody stanowi istotne rozszerzenie metody traktującej trajektorie jako krzywe parametryczne i pozwala nie tylko na modelowanie ruchu pojazdów przenoszących czujniki, ale również na uwzględnienie wielu czynników fizycznych związanych z ruchem pojazdu w trakcie przenoszenia czujników. Takie podejście stanowi ważny atut w przypadku zastosowań inżynierskich.

W celu uwzględnienia dynamiki poruszających się czujników, problem poszukiwania optymalnych trajektorii ruchu czujników sprowadzono do postaci kanonicznej Mayera, co umożliwia wykorzystanie narzędzi algorytmicznego sterowania optymalnego. Tak otrzymane zagadnienie sterowania optymalnego jest problemem silnie nieliniowym i z reguły niemożliwe jest wyznaczenie jego rozwiązania analitycznego. Do jego rozwiązania numerycznego zaadaptowano więc metodę iteracyjnego programowania dynamicznego, która jest bardzo efektywnym przybliżeniem znanej metody programowania dynamicznego. Przeprowadzone symulacje pozwoliły potwierdzić, że zaproponowana metoda pozwala na efektywne wyznaczanie trajektorii ruchu mobilnych węzłów sieci sensorycznej z uwzględnieniem ich dynamiki.

Podczas rozwiązywania problemu znajdowania optymalnych strategii pomiarowych z zastosowaniem sieci sensorów mobilnych, główny nacisk położono na zaproponowanie równoległej metody opierającej się na iteracyjnym programowaniu dynamicznym. W pracy przedstawiono wykorzystanie schematu komunikacji „każdy z każdym”, w którym każdy z procesorów wykonuje obliczenia związane z pracą algorytmu oraz wymienia tylko niezbędne dane z innymi procesorami liczącymi. Podobnie jak w przypadku metody optymalnej aktywacji czujników skanujących, również w zaproponowanej metodzie ilość danych wymienianych pomiędzy procesorami

jest niewielka, co pozwoliło na uzyskanie niskiego udziału komunikacji w ogólnym czasie obliczeń. Jednak, co pokazały obliczenia przeprowadzone z wykorzystaniem środowiska równoległego składającego się z kilkudziesięciu procesorów, również ta metoda cechuje się dość znacznym zwiększeniem udziału komunikacji w ogólnym czasie obliczeń wraz ze wzrostem liczby procesorów.

Przedstawione przykłady pozwoliły na zobrazowanie skali problemów oraz stopnia ich skomplikowania w przypadku syntezy algorytmów optymalnego wyznaczania trajektorii mobilnych węzłów sieci sensorycznej. W przypadku zastosowań praktycznych należy się spodziewać znacznie bardziej skomplikowanych problemów, a co za tym idzie — potrzeby wykorzystania znacznie większych mocy obliczeniowych dostarczanych przez nowoczesne środowiska obliczeniowe.

Podsumowując, głównym celem pracy było zaproponowanie metod i algorytmów rozwiązywania zagadnień optymalnej obserwacji procesów z czasoprzestrzenną dynamiką oraz ich zaprojektowanie z zastosowaniem technik programowania równoległego. W tym celu rozważane zagadnienia zostały przeanalizowane pod kątem ich złożoności złożoności obliczeniowej, tak aby opracowane metody i algorytmy równoległe efektywnie wykonywały obliczenia w systemach wieloprocesorowych oraz prowadziło do możliwie najkrótszych czasów obliczeń. Do celów badań wykorzystano różne schematy komunikacji pomiędzy procesorami. Zaproponowane metody pozwalają na przeprowadzanie efektywnych (pod względem zarówno uzyskiwanych czasów, jaki i przyspieszeń) obliczeń prowadzących do uzyskania optymalnych strategii pomiarowych z wykorzystaniem sieci czujników stacjonarnych, skanujących oraz ruchomych.

Niestety, ze względów technicznych nie udało się przetestować wszystkich zaproponowanych algorytmów w środowisku równoległym składającym się z kilkudziesięciu czy klikuset procesorów. Przeprowadzona analiza pozwala na przypuszczenia dotyczące zachowania się algorytmów w środowiskach równoległych większej skali, jednak dopiero przeprowadzenie rzeczywistych testów pozwoli na zweryfikowanie tych przypuszczeń.

Dodatkowym aspektem, uwzględnionym w pracy, było zaadaptowanie przedstawionych metod pod kątem ich uruchamiania w heterogenicznych środowiskach obliczeniowych. W systemie równoległym składającym się z procesorów o różnej mocy obliczeniowej, czas obliczeń algorytmów zrównoleglonych poprzez dystrybucję obliczeń pomiędzy procesory liczące jest zdeterminowany przez najwolniej działający procesor. Wykorzystanie metod równoważenia obciążenia pozwala na wyeliminowanie wpływu najwolniejszego procesora na czas obliczeń i umożliwia optymalne wykorzystanie dostępnej mocy obliczeniowej, co prowadzi do skrócenia czasu wykonywania obliczeń.

6.2 Wnioski

Na podstawie przeprowadzonych badań można stwierdzić, że:

- wykorzystanie technik programowania równoległego pozwala na efektywne wyznaczanie optymalnych konfiguracji sieci czujników,
- wykorzystanie metod opierających się na optymalizacji dyskretnej pozwala na optymalną konfigurację sieci czujników stacjonarnych oraz skanujących z wykorzystaniem technik programowania równoległego,
- wykorzystanie równoległych metod opierających się na optymalizacji ciągłej pozwala na wyznaczanie optymalnych trajektorii mobilnych węzłów sieci sensorycznej również przy uwzględnieniu rzeczywistych ograniczeń związanych z dynamiką węzłów mobilnych,
- w celu uzyskania znaczących przyspieszeń oprócz wykorzystania technik programowania równoległego należy dokładnie rozpoznać specyfikę rozważanych zagadnień i poszukiwać metod przyspieszania obliczeń również na etapie konstrukcji metod optymalnego planowania eksperymentu,
- konstruowanie równoległych metod obliczeniowych powinno prowadzić do minimalizacji udziału komunikacji pomiędzy procesorami,
- konstruowane algorytmy równoległe powinny uwzględniać niejednorodność środowisk obliczeniowych w celu efektywnego wykorzystania dostępnej mocy obliczeniowej.

6.3 Oryginalne wyniki naukowe pracy

Za oryginalny wkład własny autor uznaje opracowanie metod umożliwiających efektywne wyznaczanie optymalnych konfiguracji sieci czujników z wykorzystaniem technik programowania równoległego. Opracowano algorytmy i programy komputerowe (wykorzystano język programowania Fortran 95 oraz środowisko programowania równoległego MPI) umożliwiające przeprowadzenie testów numerycznych oraz potwierdzających skuteczność przedstawionych metod. W ramach przeprowadzonych badań opracowano, zaimplementowano i przetestowano następujące metody i algorytmy:

- równoległa metoda rozwiązywania układów dwuwymiarowych parabolicznych równań różniczkowych cząstkowych oparta na metodzie elementu skończonego,
- równoległa metoda optymalnej aktywacji sieci czujników stacjonarnych w oparciu o schemat metody GRASP z wykorzystaniem schematu komunikacji równoległej pomiędzy procesorami typu „nadzorca–podwładni”,
- równoległa metoda optymalnej aktywacji sieci czujników skanujących w oparciu o schemat metody Tabu-Search z wykorzystaniem schematu komunikacji równoległej pomiędzy procesorami typu „każdy z każdym”,

- metodę redukcji czasochłonnych operacji w oparciu o formułę Shermana-Morrisona-Woodbury'ego umożliwiającą jej zastosowanie w zagadnieniu wyznaczania optymalnej konfiguracji czujników skanujących,
- równoległa metoda wyznaczania optymalnych trajektorii mobilnych węzłów sieci sensorycznych w oparciu o algorytm tunelowy z wykorzystaniem schematu komunikacji równoległej pomiędzy procesorami typu „nadzorca–podwładni",
- równoległa metoda wyznaczania optymalnych trajektorii mobilnych węzłów sieci sensorycznych z uwzględnieniem ich dynamiki w oparciu o iteracyjne programowanie dynamiczne z wykorzystaniem schematu komunikacji równoległej „każdy z każdym",
- implementacja metod równoważenia obciążenia w przedstawionych algorytmach pozwalająca na efektywne przeprowadzanie obliczeń w heterogenicznych środowiskach równoległych.

6.4 Kierunek dalszych badań

Niniejsza praca może stanowić punkt wyjściowy do dalszych badań nad rozwojem i zastosowaniem środowisk programowania równoległego na potrzeby optymalnej obserwacji układów z czasoprzestrzenną dynamiką. Do dalszych kierunków badań można zaliczyć m.in.

- **Badania nad zastosowaniem zaproponowanych metod i algorytmów w środowiskach równoległych składających się większej liczby procesorów**.
  Implikuje to konieczność opracowania i zbadania innych metod dekompozycji rozważanych zagadnień na wiele procesorów nowych metod komunikacji pomiędzy procesorami liczącymi w celu minimalizacji wpływ opóźnień komunikacji na uzyskiwane przyspieszenie obliczeń.
- **Implementacja przedstawionych oraz zaproponowanie nowych metod rozwiązywania rozważanych zagadnień w środowiskach gridowych**.
  Rozwiązywanie przedstawionych zagadnień w środowiskach gridowych wymaga opracowania i zbadania nowych metod pozwalających na przeprowadzanie efektywnych obliczeń w środowiskach składających się z rozproszonych systemów obliczeniowych o różnej mocy obliczeniowej. Dodatkowym aspektem jest konieczność uwzględnienia znacznych opóźnień pomiędzy środowiskami obliczeniowymi wynikającymi z ich geograficznego rozproszenia.
- **Rozwinięcie opracowywanych metod dla zagadnień planowania odpornego**.

Poważnym problemem rozważanych zagadnień jest zależność optymalnych położeń czujników od rzeczywistych wartości identyfikowanych parametrów, które są nieznane. Jednym ze sposobów uniezależnienia optymalnych położeń czujników od wartości identyfikowanych parametrów jest stosowanie tzw. planowania odpornego [228], które pozwala określić „możliwie najlepsze" położenia czujników dla danego zakresu parametrów. Złożoność problemu optymalizacji wyklucza analityczne wyznaczenie optymalnych warunków eksperymentu, co implikuje potrzebę odwołania się do technik numerycznych. Wyznaczenie wartości oczekiwanej (całki wielowymiarowej) kryterium lokalnego powoduje konieczność zastosowanie metod aproksymacji stochastycznej. Olbrzymia złożoność zagadnienia implikuje dalsze istnienie wielu otwartych problemów badawczych stanowiących zarówno interesujące wyzwanie natury naukowej, jak i ważne zadania z punktu widzenia potencjalnych zastosowań.

- **Zaproponowanie metod planowania eksperymentu powiązanych z procesem identyfikacji parametrów**.
  Przedstawione metody pozwalają na opracowanie strategii pomiarowych przed rozpoczęciem właściwego eksperymentu (opracowane są tzw. strategie offline). Interesujące wydaje się zastosowanie podejścia pozwalającego na wykonywanie pomiarów, identyfikację parametrów i na tej podstawie dynamiczną aktualizację strategii pomiarowej z zastosowaniem mobilnych węzłów sieci sensorycznej (tzw. strategie online). Niewielka liczba publikacji podejmujących to zadanie stanowi motywację do dalszych wysiłków, zwłaszcza w kontekście rosnącego znaczenia potencjalnych rozwiązań dla coraz bardziej rozprzestrzeniających się sieci sensorycznych.

A

Dodatek

A.1 Interpolacja liniowa funkcji jednej zmiennej

Przybliżenia funkcji jednej zmiennej w rozdz. 5 dokonuje się z zastosowaniem interpolacji liniowej [24, 66]. Przypuśćmy, że znamy wartość funkcji f w punktach t_1 oraz t_2, czyli $y_1 = f(t_1)$ oraz $y_2 = f(t_2)$. Interpolacja liniowa wartości funkcji f w punkcie t takim, że $t_1 < t < t_2$, polega na połączeniu odcinkiem punktów $(t_1, f(t_1))$ oraz $(t_2, f(t_2))$ i wyznaczeniu $y = f(t)$ jako odległości punktu $(t, 0)$ od punktu przecięcia wspomnianego odcinka z prostą $y = t$. Prowadzi to do równania

$$\frac{y - y_1}{y_2 - y_1} = \frac{t - t_1}{t_2 - t_1}. \tag{A.1}$$

Po rozwiązaniu równania (A.1) ze względu na y otrzymujemy równanie określające szukane przybliżenie $f(t)$:

$$y = y_1 + (t - t_1)\frac{y_2 - y_1}{t_2 - t_1}. \tag{A.2}$$

A.2 Interpolacja biliniowa

Jako metodę interpolacji funkcji dwóch zmiennych zadanej na regularnej siatce prostokątnej wykorzystano rozszerzenie klasycznej interpolacji liniowej [24, 66]. Przypuśćmy, że znamy wartość funkcji f w czterech punktach $(x_1, y_1), (x_2, y_1), (x_1, y_2), (x_2, y_2)$, gdzie: $x_1 < x < x_2$ oraz $y_1 < y < y_2$.

Interpolacja wzdłuż kierunku osi x pozwala na określenie wartości funkcji w punktach:

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(x_1, y_1) + \frac{x - x_1}{x_2 - x_1} f(x_2, y_1), \tag{A.3}$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(x_1, y_2) + \frac{x - x_1}{x_2 - x_1} f(x_2, y_2). \tag{A.4}$$

Interpolując wzdłuż kierunku osi y otrzymujemy:

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2). \tag{A.5}$$

Ostatecznie, uwzględniając równania (A.3)–(A.5) otrzymujemy równanie interpolacji liniowej dwuwymiarowej funkcji f w punkcie (x, y)

$$\begin{aligned} f(x, y) \approx & \frac{f(x_1, y_1)}{(x_2 - x_1)(y_2 - y_1)} (x_2 - x)(y_2 - y) \\ & + \frac{f(x_2, y_1)}{(x_2 - x_1)(y_2 - y_1)} (x - x_1)(y_2 - y) \\ & + \frac{f(x_1, y_2)}{(x_2 - x_1)(y_2 - y_1)} (x_2 - x)(y - y_1) \\ & + \frac{f(x_2, y_2)}{(x_2 - x_1)(y_2 - y_1)} (x - x_1)(y - y_1). \end{aligned} \tag{A.6}$$

A.3 Splajny sześcienne

W celu parametryzacji trajektorii mobilnych węzłów sieci sensorycznej (zob. rozdz. 5.2.1) zaproponowano wykorzystanie liniowej kombinacji B-splajnów sześciennych [24, 66]. Metoda opiera się na L funkcjach bazowych, których liniowa kombinacja pozwala na opisanie trajektorii ruchu sensora wzdłuż jednego kierunku (zob. rys. A.1):

$$\eta(t, \omega) = \eta_{\min} + \sum_{i=1}^{L} \omega_i \phi_i(t), \quad t \in T = [0, t_f], \tag{A.7}$$

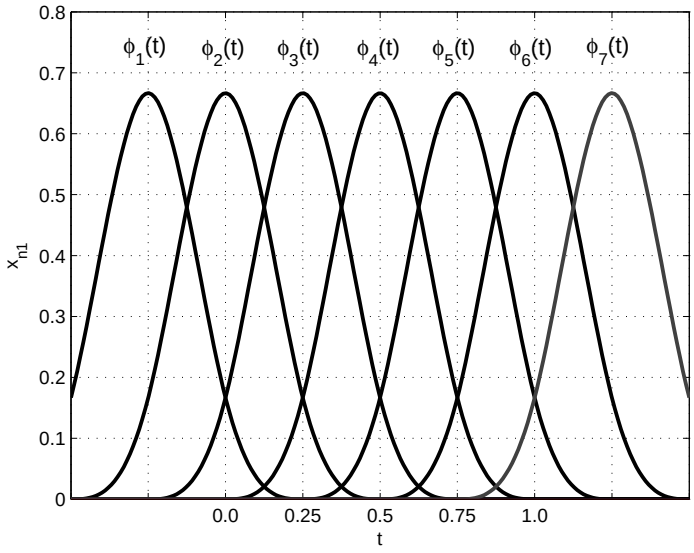
gdzie:

$$\phi_i(t) = (\eta_{\max} - \eta_{\min}) g(\frac{t}{h} - i + 2), \tag{A.8}$$

$$g(\tau) = \begin{cases} \frac{2}{3} - \frac{1}{2}|\tau|^2(2 - |\tau|), & 0 \leqslant |\tau| < 1, \\ \frac{1}{6}(2 - |\tau|)^3, & 1 \leqslant |\tau| < 2, \\ 0, & 2 \leqslant |\tau|, \end{cases} \tag{A.9}$$

a wartość h obliczana jest jako $h = t_f/(L - 3)$.

Wartości $\eta_{\min}, \eta_{\max}$ oznaczają minimalną oraz maksymalną wartość jaką funkcja może osiągnąć — w przypadku określania trajektorii wartości te mogą definiować obszar w jakim poruszać się będą mobilne węzły sieci sensorycznej. Kombinacja liniowa jest zdefiniowana poprzez funkcje pomocnicze $\phi_i(t)$ oraz $g(\tau)$ tak, że maksymalna wartość funkcji $\eta(t, \omega)$ w każdym punkcie wynosi $\eta_{\max}$. Do prawidłowego określenia kształtu trajektorii wymagana jest liczba funkcji bazowych wynosi $L > 3$, a kształt krzywej definiowany jest przez wektor ω gdzie: $0 \leqslant \omega_i \leqslant 1,\ i = 1, \ldots, L$.



Rysunek A.1: B-splajny sześcienne.

A.4 Metoda optymalizacji Rosenbrocka

Metoda optymalizacji Rosenbrocka pozwala na znajdowanie minimum bez obliczania gradientu funkcji. Została zaprezentowana przez Rosenbrocka w 1970 r [194].

Na początku algorytm zachowuje się jak klasyczny algorytm bezgradientowy minimalizacji wielowymiarowej. Algorytm startuje z punktu początkowego i zaczyna wykonywać procedurę minimalizacji względem wektora bazowego (na początku jest to procedura minimalizacji po współrzędnych) dla każdego z kierunków optymalizowanej funkcji. Metoda wykorzystuje adaptacyjne dostosowywanie wielkości kroku (przesunięcia) w zależności od wyznaczonych wartości optymalizowanej funkcji. Jeżeli algorytm wyznaczy lepsze przybliżenie punktu minimum funkcji celu, dzięki wyznaczeniu nowego punktu (poprzez dodanie przesunięcia) dla danego kierunku, to w kolejnej iteracji algorytmu wielkość kroku (przesunięcia) dla rozważanego kierunku będzie zwiększona w celu szybszej minimalizacji w tym kierunku. W przypadku, gdy minimalizacja wzdłuż danego kierunku dla określonego kroku spowoduje uzyskanie wartości funkcji gorszej od aktualnego minimum, następuje zmniejszenie wartości kroku poszukiwania oraz zmiana jego zwrotu na przeciwny, a następnie przystąpienie do minimalizacji wzdłuż kolejnego kierunku optymalizowanej funkcji.

Po zakończeniu cyklu minimalizacji jednokierunkowych dla wszystkich kierunków funkcji, następuje obrót osi współrzędnych tak, aby nowy wektor bazowy był obrócony w kierunku największego spadku. Następnie algorytm rozpoczyna swoją pracę od początku wykorzystując nowy wektor bazowy. Obrót osi współrzędnych jest obliczany przy wykorzystaniu procedury ortogonalizacji Gram-Schmidta [195].

A.5 Formuła Frobeniusa

Niech $A \in \mathbb{R}^{n\times n}$ będzie przedstawiona w postaci

$$A = \left[\begin{array}{c:c} A_{11} & A_{12} \\ \hdashline A_{21} & A_{22} \end{array}\right], \tag{A.10}$$

gdzie A_{11}, A_{22} są macierzami kwadratowymi. Jeżeli istnieje A_{11}^{-1}, wtedy

$$\det(A) = \det(A_{11})\det(F), \tag{A.11}$$

gdzie $F = A_{22} - A_{21}A_{11}^{-1}A_{12}$. Dodatkowo, jeżeli $\det(F) \neq 0$, wtedy

$$A^{-1} = \left[\begin{array}{c:c} A_{11}^{-1} + A_{11}^{-1}A_{12}F^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}F^{-1} \\ \hdashline -F^{-1}A_{21}A_{11}^{-1} & F^{-1} \end{array}\right]. \tag{A.12}$$

Podobnie, jeżeli istnieje A_{22}^{-1}, wtedy

$$\det(A) = \det(A_{22})\det(G), \tag{A.13}$$

gdzie $G = A_{11} - A_{12}A_{22}^{-1}A_{21}$. Dodatkowo, jeżeli $\det(G) \neq 0$, wtedy

$$A^{-1} = \left[\begin{array}{c:c} G^{-1} & -G^{-1}A_{12}A_{22}^{-1} \\ \hdashline -A_{22}^{-1}A_{21}G^{-1} & A_{22}^{-1} + A_{22}^{-1}A_{21}G^{-1}A_{12}A_{22}^{-1} \end{array}\right]. \tag{A.14}$$

Bibliografia

[1] Adeli, H. *Advances in Design Optimization.* Chapman and Hall, London, 1994.

[2] Aiex, R. M., Resende, M. G. C., Pardalos, P. M., and Toraldo, G. Grasp with path-relinking for the three-index assignment problem. Tech. rep., INFORMS Journal on Computing, 2000.

[3] Ames, W. F. *Numerical Methods for Partial Differential Equations.* Academic Press, Orlando, Florida 32887, 1977.

[4] Atkinson, A. C., and Donev, A. N. *Optimum Experimental Designs.* Clarendon Press, Oxford, 1992.

[5] Atlas homepage. http://math-atlas.sourceforge.net/.

[6] Banicescu, I., and Hummel, S. F. Balancing processor loads and exploiting data locality in n-body simulations. In *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)* (New York, NY, USA, 1995), ACM, p. 43.

[7] Banicescu, I., and Velusamy, V. Load balancing highly irregular computations with the adaptive factoring. In *IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium* (Washington, DC, USA, 2002), IEEE Computer Society, p. 195.

[8] Banicescu, I., Velusamy, V., and Devaprasad, J. On the scalability of dynamic scheduling scientific applications with adaptive weighted factoring. *Cluster Computing 6*, 3 (2003), 215–226.

[9] Banichuk, N. V. *Introduction to Optimization of Structures*. Springer-Verlag, New York, 1990.

[10] Banks, H. T., and Kunisch, K. *Estimation Techniques for Distributed Parameter Systems*. Systems & Control: Foundations & Applications. Birkhäuser, Boston, 1989.

[11] Banks, H. T., Smith, R. C., and Wang, Y. *Smart Material Structures: Modeling, Estimation and Control*. Research in Applied Mathematics. Masson, Paris, 1996.

[12] Baranowski, P., and Kuczewski, B. Diagnostyka procesów z czasoprzestrzenną dynamiką z zastosowaniem przetwarzania równoległego. *Pomiary, Automatyka, Kontrola*, nr 9, wyd. spec. (2005), 74–76.

[13] Baranowski, P., and Zięba, T. Parallel processing for optimal measurement scheduling in parameter estimation of dynamic systems. In *1st. International Conference for Young Researchers in Computer Science, Control, Electrical Engineering and Telecommunications - ICYR : abstracts* (Zielona Góra, Polska, 2006), Faculty of Electrical Engineering, Computer Science and Telecommunications, University of Zielona Góra, Zielona Góra, p. 81.

[14] Baranowski, P., Zięba, T., and Uciński, D. A parallel sensor selection technique for optimal observation of distributed parameter systems. In *Methods and Models in Automation and Robotics - MMAR 2006 : proceedings of the 12th IEEE international conference; ISBN: 83-60140-93-6* (Międzyzdroje, Polska, 2006), Institute of Control Engineering, Szczecin University of Technology, [Szczecin], PPH Zapol, Dmochowski, Sobczyk, Spółka Jawna, pp. 157–164 [CD–ROM].

[15] Battiti, R., and Tecchiolli, G. Parallel biased search for combinatorial optimization: genetic algorithms and tabu. *Microprocess. Microsyst. 16*, 7 (1992), 351–367.

[16] Berliner, M. L., Nychka, D., and Hoar, T. *Studies in the Atmospheric Sciences*. Springer-Verlag, New York, 2000.

[17] Bertsekas, D. P. *Dynamic Programming and Optimal Control*, 2nd ed., vol. I. Athena Scientific, Belmont, MA, 2000.

[18] Bertsekas, D. P., and Tsitsiklis, J. N. *Neuro-dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[19] Blas homepage. http://www.netlib.org/blas/.

[20] Bojkov, B., and Luus, R. Use of random admissible values for control in iterative dynamic programming. *Ind. Eng. Chem. Res. 31* (1992), 1308–1314.

[21] Bojkov, B., and Luus, R. Evaluation of the parameters used in iterative dynamic programming. *Ind. Eng. Chem. Res. 71* (1993), 451–459.

[22] Boukerche, A. *Handbook of Algorithms for Wireless Networking and Mobile Computing (Chapman & Hall/Crc Computer & Information Science)*. Chapman & Hall/CRC, 2005.

[23] Bryson, Jr., A. E. *Dynamic Optimization*. Addison-Wesley, New York, 1999.

[24] Burden, R. L., and Faires, J. D. *Numerical Analysis*. Prindle, Weber & Schmidt, Boston, 1985.

[25] Buyya, R. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[26] Carotenuto, L., Muraca, P., and Raiconi, G. Optimal location of a moving sensor for the estimation of a distributed-parameter process. *International Journal of Control 46*, 5 (1987), 1671–1688.

[27] Caselton, W. F., Kan, L., and Zidek, J. V. Quality data networks that minimize entropy. In *Statistics in the Environmental and Earth Sciences*, A. Walden and P. Guttorp, Eds. Halsted Press, New York, 1992, ch. 2, pp. 10–38.

[28] Caselton, W. F., and Zidek, J. V. Optimal monitoring network design. *Statistics & Probability Letters 2* (1984), 223–227.

[29] Cassandras, C. G., and Li, W. Sensor networks and cooperative control. *European Journal of Control 11*, 4–5 (2005), 436–463.

[30] Chavent, G. Identification of functional parameters in partial differential equations. In *Identification of Parameters in Distributed Systems*, R. E. Goodson and M. Polis, Eds. The American Society of Mechanical Engineers, New York, 1974, pp. 31–48.

[31] Chen, W. H., and Seinfeld, J. H. Optimal location of process measurements. *International Journal of Control 21*, 6 (1975), 1003–1014.

[32] Chong, C.-Y., and Kumar, S. P. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE 91*, 8 (2003), 1247–1256.

[33] Chopard, B., and Droz, M. *Cellular Automata Modeling of Physical Systems*. Cambridge University Press, 1998.

[34] Cluster resources :: Products - torque resource manager. http://www.clusterresources.com/pages/products/torque-resource-manager.php.

[35] Clusterix homepage. http://www.clusterix.pl/.

[36] Comsol - multiphysics modeling. http://www.comsol.com/.

[37] Condor project homepage. http://www.cs.wisc.edu/condor/.

[38] Cox, D. R., and Reid, N. *The Theory of the Design of Experiments*. Chapman and Hall, Boca Raton, FL, 2000.

[39] Cressie, N. A. C. *Statistics for Spatial Data*, revised ed. John Wiley & Sons, New York, 1993.

[40] Crossbow technology : Crossbow wireless sensor casestudies. http://www.xbow.com/industry_solutions/customerreference.aspx.

[41] Cuthbert, D. *Applications of Statistics to Industrial Experimentation*. John Wiley & Sons, New York, 1976.

[42] Cybenko, G. Dynamic load balancing for distributed memory multiprocessors. *J. Parallel Distrib. Comput. 7*, 2 (1989), 279–301.

[43] Davis, J. H. *Foundations of Deterministic and Stochastic Control*. Birkhäuser, Boston, 2002.

[44] de Cogan, D., and de Cogan, A. *Applied Numerical Modelling for Engineers*. Oxford University Press, New York, 1997.

[45] Debouk, R., Lafortune, S., and Teneketzis, D. On an optimization problem in sensor selection. *Discrete Event Dynamic Systems 12*, 4 (2002), 417–445.

[46] Deng, X., Liu, H.-N., Long, J., and Xiao, B. Competitive analysis of network load balancing. *J. Parallel Distrib. Comput. 40*, 2 (1997), 162–172.

[47] Diffpack homepage. http://www.diffpack.com/.

[48] Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., and White, A., Eds. *Sourcebook of parallel computing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[49] Earthscope: An earth science program homepage. http://www.earthscope.org/.

[50] El Jai, A., and Pritchard, A. J. *Sensors and Controls in the Analysis of Distributed Systems*. John Wiley & Sons, New York, 1988.

[51] El-Rewini, H., and Abd-El-Barr, M. *Advanced Computer Architecture and Parallel Processing (Wiley Series on Parallel and Distributed Computing)*. Wiley-Interscience, 2005.

[52] Ermakov, S. M., Ed. *Mathematical Theory of Experimental Design.* Nauka, Moscow, 1983. (In Russian).

[53] Ermakov, S. M., and Zhigljavsky, A. A. *Mathematical Theory of Optimal Experiments.* Nauka, Moscow, 1987. (In Russian).

[54] Eslami, M. *Theory of Sensitivity in Dynamic Systems. An Introduction.* Springer-Verlag, Berlin, 1994.

[55] Fastflo homepage. http://www.cmis.csiro.au/fastflo/.

[56] Fedorov, V. V. *Theory of Optimal Experiments.* Academic Press, New York, 1972.

[57] Fedorov, V. V. Optimal design with bounded density: Optimization algorithms of the exchange type. *Journal of Statistical Planning and Inference 22* (1989), 1–13.

[58] Fedorov, V. V., and Hackl, P. *Model-Oriented Design of Experiments.* Lecture Notes in Statistics. Springer-Verlag, New York, 1997.

[59] Feo, T., Resende, M., and Smith, S. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research 42* (1994), 860–878.

[60] Feo, T., and Resende, M. G. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters 8* (1989), 67–71.

[61] Feo, T., and Resende, M. G. Greedy randomized adaptive search procedures. *J. of Global Optimization 6* (1995), 109–133.

[62] Fftw homepage. http://www.fftw.org/.

[63] Fluent newsletter: Pollution dispersion in urban landscapes – summer 2006. http://www.fluent.com/about/news/newsletters/06v15i2/a3.pdf.

[64] Flynn, M. J. Very high-speed computing systems. *Proceedings of the IEEE 54*, 12 (1966), 1901–1909.

[65] Ford, I., Titterington, D. M., and Kitsos, C. P. Recent advances in nonlinear experimental design. *Technometrics 31*, 1 (1989), 49–60.

[66] Fortuna, Z., Macukow, B., and Wąsowski, J. *Metody numeryczne.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1992.

[67] Foster, I. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* "Addison-Wesley Longman Publishing Co., Inc.", "Boston, MA, USA", 1995.

[68] Freefem homepage. http://www.freefem.org/.

[69] Freund, R. F., and Siegel, H. J. Guest editor's introduction: Heterogeneous processing. *Computer 26*, 6 (1993), 13–17.

[70] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing.* "The MIT Press", 1994.

[71] Glover, F. Tabu search — part i. *ORSA Journal on Computing 1*, 3 (1989), 190–206.

[72] Glover, F., Taillard, E., and de Werra, D. A user's guide to tabu search. *Ann. Oper. Res. 41*, 1-4 (1993), 3–28.

[73] Golub, G. H., and Ortega, J. M. *Scientific Computing and Differential Equations. An Introduction to Numerical Methods.* Academic Press, San Diego, California, 1992.

[74] Gómez, S., del Castillo, N., Castellanos, L., and Solano, J. The parallel tunneling method. *Elsevier 29*, 4 (2003), 523–533.

[75] Gómez, S., and Levy, A. V. The tunneling method for solving the constrained global optimization problem with several non-connected feasible regions. *Lecture Notes in Mathematics*, 909 (1982), 34–47.

[76] Goodwin, G. C., and Payne, R. L. *Dynamic System Identification. Experiment Design and Data Analysis.* Mathematics in Science and Engineering. Academic Press, New York, 1977.

[77] Grama, A., Gupta, A., Karypis, G., and Kumar, V. *Introduction to Parallel Computing, Second Edition.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[78] Grama, A., and Kumar, V. State of the art in parallel search techniques for discrete optimization problems. *IEEE Transactions on Knowledge and Data Engineering archive 11*, 1 (1999), 28–35.

[79] Grand challenges for engineering homepage. http://www.engineeringchallenges.org/.

[80] Gropp, W., Lusk, E., and Skjellum, A. *Using MPI, 2nd Edition. Portable Parallel Programming with the Message-Passing Interface.* MIT Press, Cambridge, MA, USA, 1999.

[81] Gropp, W., Lusk, E., and Thakur, R. *Using MPI-2: Advanced Features of the Message-Passing Interface.* MIT Press, Cambridge, MA, USA, 1999.

[82] Haberman, R. *Elementary Applied Partial Differentail Equations with Fourier Series and Boundary Value Problems*. Prentice Hall, Englewood Cliffs, New Jersey 07632, 1983.

[83] Hager, W. W. Updating the inverse of a matrix. *SIAM Review 31*, 2 (1989), 221–239.

[84] Hartig, F., Mandel, K., and Keil, F. J. Parallelization of iterative dynamic programming (idp). *Per. Pol. Chem. Eng. 43*, 1 (1999), 3–16.

[85] Haug, E. J., Choi, K. K., and Komkov, V. *Design Sensitivity Analysis of Structural Systems*. Mathematics in Science and Engineering. Academic Press, Orlando, FL, 1986.

[86] High performance computing - platform hpc management software. http://www.platform.com/.

[87] Hockney, R. W. *The Science of Computer Benchmarking*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1996.

[88] Hoffman, J. D. *Numerical Methods for Engineers and Scientists, Second Edition*. CRC Press, New York, 2001.

[89] Holnicki-Szulc, P. *Modele propagacji zanieczyszczeń atmosferycznych w zastosowaniu do kontroli i sterowania jakością środowiska*. "Akademicka Oficyna Wydawnicza EXIT", Warszawa, 2006.

[90] Huang, J., and Lee, S.-Y. A heterogeneity-aware approach to load balancing of computational tasks: a theoretical and simulation study. *Cluster Computing* (2007).

[91] Hummel, S. F., Schmidt, J., Uma, R. N., and Wein, J. Load-sharing in heterogeneous systems via weighted factoring. In *SPAA '96: Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures* (New York, NY, USA, 1996), ACM, pp. 318–328.

[92] Hummel, S. F., Schonberg, E., and Flynn, L. E. Factoring: a method for scheduling parallel loops. *Commun. ACM 35*, 8 (1992), 90–101.

[93] Ibm cluster software: Tivoli workload scheduler loadleveler. http://www-03.ibm.com/systems/clusters/software/loadleveler/index.html.

[94] Ilyas, M., Mahgoub, I., and Kelly, L. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[95] Imsl numerical libraries homepage. http://www.vni.com/products/imsl/.

[96] Infiniband trade association homepage. http://www.infinibandta.org/home.

[97] Intel performance libraries homepage. http://www.intel.com/cd/software /products/asmo–na/eng/perflib/219780.htm.

[98] Jacewicz, P. *Model analysis and synthesis of complex physical systems using cellular automata, ISBN: 83-89321-67-X.* Lecture Notes in Control and Computer Science, Vol. 4. University of Zielona Góra Press, Zielona Góra, 2003.

[99] Johnson, E. E. Completing an mimd multiprocessor taxonomy. *SIGARCH Comput. Archit. News 16*, 3 (1988), 44–47.

[100] Kammer, D. C. Sensor placement for on-orbit modal identification and correlation of large space structures. In *Proc. American Control Conf.,* San Diego, California, 23–25 May 1990 (1990), vol. 3, pp. 2984–2990.

[101] Kammer, D. C. Effects of noise on sensor placement for on-orbit modal identification of large space structures. *Transactions of the ASME 114* (Sept. 1992), 436–443.

[102] Kang, F., jie Li, J., and Xu, Q. Virus coevolution partheno-genetic algorithms for optimal sensor placement. *Adv. Eng. Inform. 22*, 3 (2008), 362–370.

[103] Karnik, A., Kumar, A., and Borkar, V. Distributed self-tuning of sensor networks. *Wirel. Netw. 12*, 5 (2006), 531–544.

[104] Kazimierczyk, P. Optimal experiment design; Vibrating beam under random loading. Tech. Rep. 7/1989, Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, 1989.

[105] Khuri, A. I., and Cornell, J. A. *Response Surfaces. Design and Analyses*, 2nd ed. Statistics: Textbooks and Monographs. Marcel Dekker, New York, 1996.

[106] Kiefer, J., and Wolfowitz, J. Optimum designs in regression problems. *The Annals of Mathematical Statistics 30* (1959), 271–294.

[107] Kincaid, R. K., and Padula, S. L. D-optimal designs for sensor and actuator locations. *Computers & Operations Research 29* (2002), 701–713.

[108] Kitowski, J. *Współczesne systemy komputerowe.* "CCNS", Kraków, 2000.

[109] Knabner, P., and Angermann, L. *Numerical Methods for Elliptic and Parabolic Partial Differential Equations.* Springer-Verlag, New York, 2003.

[110] Kontoghiorghes, E. J. *Handbook of Parallel Computing and Statistics (Statistics, Textbooks and Monographs).* Chapman & Hall/CRC, 2005.

[111] Korbicz, J., and Uciński, D. Sensors allocation for state and parameter estimation of distributed systems. In *Proc. IUTAM Symposium,* Zakopane, Poland, 31 August–3 September 1993 (Berlin, 1994), W. Gutkowski and J. Bauer, Eds., Springer-Verlag, pp. 178–189.

[112] Korbicz, J., Zajda, Z., Solnik, W., and Uciński, D. *Wybrane metody numerycznego rozwiązywania równań różniczkowych cząstkowych.* Wydaw. Wyższej Szkoły Inżynierskiej, Zielona Góra, 1991.

[113] Korbicz, J., and Zgurowski, M. Z. *Estymacja i sterowanie stochastyczne układami o parametrach rozłożonych.* Państwowe Wydawnictwo Naukowe, Warszawa, 1991.

[114] Korkhov, V., and et al. A grid-based virtual reactor: Parallel performance and adaptive load balancing. *Journal of Parallel and Distributed Computing* (2007).

[115] Kozłowski, K., Dutkiewicz, P., and Wróblewski, W. *Modelowanie i sterowanie robotów.* Wydawnictwo Naukowe PWN, 2003.

[116] Kravaris, C., and Seinfeld, J. H. Identification of parameters in distributed parameter systems by regularization. *SIAM Journal on Control and Optimization 23*, 2 (Mar. 1985), 217–241.

[117] Kubrusly, C. S., and Malebranche, H. Sensors and controllers location in distributed systems — A survey. *Automatica 21*, 2 (1985), 117–128.

[118] Kuczewski, B. *Computational aspect of discrimination between models of dynamic systems.* PhD thesis, Uniwersytet Zielonogórski, Zielona Góra, 2005.

[119] Kuczewski, B., Baranowski, P., and Uciński, D. Parallel processing in discrimination between models of dynamic systems. *Lecture Notes in Computer Science : Parallel processing and applied mathematics - 6th international conference Vol. 3911* (2006), 340–348.

[120] Kulakowski, B. *Dynamic Modeling and Control of Engineering Systems.* Cambridge University Press, Cambridge, 2007.

[121] Kumar, V., Grama, A. Y., and Vempaty, N. R. Scalable load balancing techniques for parallel computers. *J. Parallel Distrib. Comput. 22*, 1 (1994), 60–79.

[122] Lam/mpi parallel computing homepage. http://www.lam-mpi.org/.

[123] Lapack homepage. http://www.netlib.org/lapack/.

[124] Lasiecka, I., and Triggiani, R. *Control Theory for Partial Differential Equations: Continuous and Approximation Theories*, vol. I and II of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 2000.

[125] Levy, A. V., and Gómez, S. The tunneling method applied to global optimization. In *SIAM Numerical Optimization 1984* (1985), SIAM, pp. 213–244.

[126] Levy, A. V., and Montalvo, A. The tunneling algorithm for the global minimization of functions. *SIAM Journal on Scientific and Statistical Computing 6*, 1 (1985), 15–29.

[127] Li, R., Chen, Z., and Wu, W. *Generalized Difference Method for Differential Equations. Numerical Analysis of Finite Volume Methods*. Marcel Dekker, Madison Avenue, New York, 2000.

[128] Ljung, L. *System Identification: Theory for the User (2nd Edition)*. Prentice Hall PTR, 1998.

[129] Luus, R. Optimal control by dynamic programming using accessible grid points and region reduction. *Hung. J. Ind. Chem. 17* (1989), 523–543.

[130] Luus, R. Optimal control by dynamic programming using systematic reduction in grid size. *Int. J. Control 19* (1990), 995–1013.

[131] Luus, R. Application of iterative dynamic programming to very high-dimensional systems. *Hung. J. Ind. Chem. 21* (1993), 243–250.

[132] Luus, R. Numerical convergence properties of iterative dynamic programming when applied tohigh dimensional systems. *Chem. Eng. Res. Des. 74* (1996), 55–62.

[133] Luus, R. *Iterative Dynamic Programming*. CRC Press, Inc., Boca Raton, FL, USA, 2000.

[134] Luyben, W. L. *Process modeling, simulation, and control for chemical engineers*. McGraw-Hill, New York, 1973.

[135] Malanowski, K., Nahorski, Z., and Peszyńska, M., Eds. *Modelling and Optimization of Distributed Parameter Systems*. International Federation for Information Processing. Kluwer Academic Publishers, Boston, 1996.

[136] Malebranche, H. Simultaneous state and parameter estimation and location of sensors for distributed systems. *International Journal of Systems Science 19*, 8 (1988), 1387–1405.

[137] Marczuk, G. I. *Analiza numeryczna zagadnień fizyki matematycznej.* PWN, Warszawa, 1983.

[138] Martínez, S., and Bullo, F. Optimal sensor placement and motion coordination for target tracking. *Automatica 42* (2006), 661–668.

[139] Mehra, R. K. Optimal input signals for parameter estimation in dynamic systems-survey and new results. *IEEE Transactions on Automatic Control 19*, 6 (Dec. 1974), 753–768.

[140] Message passing interface homepage. http://www-unix.mcs.anl.gov/mpi/.

[141] Michalczyk, J. K. *Transport gazowych zanieczyszczeń w powietrzu - symulacje numeryczne w skali lokalnej. Rozprawa doktorska.* Politechnika Lubelska, Lublin, 2003.

[142] Mickle, M. H., and Paul, J. M. Load balancing using heterogeneous processors for continuum problems on a mesh. *Journal of Parallel and Distributed Computing 39* (1996), 66–73.

[143] MPI, F. Mpi: A message-passing interface. In *Supercomputing '93. IEEE Computational Science and Engineering Proceedings. 15–19 November 1993* (Portland, Washington, USA, 1993), IEEE Computational Science and Engineering, pp. 878–883.

[144] Mpi documents homepage. http://www.mpi-forum.org/docs/docs.html.

[145] Mpich-g2 homepage. http://www3.niu.edu/mpi/.

[146] Mpich2 : High-performance and widely portable mpi homepage. http://www.mcs.anl.gov/research/projects/mpich2/.

[147] Müller, W. G. *Collecting Spatial Data. Optimum Design of Experiments for Random Fields*, 2nd revised ed. Contributions to Statistics. Physica-Verlag, Heidelberg, 2001.

[148] Munack, A. Optimal sensor allocation for identification of unknown parameters in a bubble-column loop bioreactor. In *Analysis and Optimization of Systems, Part 2*, A. V. Balakrishnan and M. Thoma, Eds., Lecture Notes in Control and Information Sciences, volume 63. Springer-Verlag, Berlin, 1984, pp. 415–433.

[149] Myrinet homepage. http://myri.com.

[150] Myszka, W. *Komputerowy system obsługi eksperymentu.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1991.

[151] Mzyk, G. *Parametryczna identyfikacja systemów o złożonej strukturze. Rozprawa doktorska.* Politechnika Wrocławska, Wrocław, Poland, 2002.

[152] Nakano, K., and Sagara, S. Optimal scanning measurement problem for a stochastic distributed-parameter system. *International Journal of Systems Science 19*, 7 (1988), 1069–1083.

[153] Niedoba, J., and Niedoba, W. *Równania różniczkowe zwyczajne i cząstkowe. Zadania z matematyki.* Uczelnianie Wydawnictwa Naukowo-Dydaktyczne, Akademia Górniczo-Hutnicza, Kraków, 2001.

[154] Nieradzik, K., and Uciński, D. A dynamic programming approach to sensor network design for parameter estimation of distributed systems. In *Methods and Models in Automation and Robotics - MMAR 2006 : proceedings of the 12th IEEE international conference; ISBN: 83-60140-93-6* (Międzyzdroje, Polska, 2006), Institute of Control Engineering, Szczecin University of Technology, [Szczecin], PPH Zapol, Dmochowski, Sobczyk, Spółka Jawna, pp. 131–138 [CD–ROM].

[155] Numerical algorithms group homepage. http://www.nag.co.uk/.

[156] Nychka, D., Piegorsch, W. W., and Cox, L. H., Eds. *Case Studies in Environmental Statistics.* Lecture Notes in Statistics, volume 132. Springer-Verlag, New York, 1998.

[157] Nychka, D., and Saltzman, N. Design of air-quality monitoring networks. In *Case Studies in Environmental Statistics*, D. Nychka, W. W. Piegorsch, and L. H. Cox, Eds., Lecture Notes in Statistics, volume 132. Springer-Verlag, New York, 1998, pp. 51–76.

[158] Obuchowicz, A. *Evolutionary algorithms for global optimization and dynamic system diagnosis, ISBN: 83-88317-02-4.* Monographs, Vol. 3. Lubusky Scientific Society, Zielona Góra, 2003.

[159] Ögren, P., Fiorelli, E., and Leonard, N. E. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control 49*, 8 (2004), 1292–1302.

[160] Omatu, S., and Seinfeld, J. H. *Distributed Parameter Systems: Theory and Applications.* Oxford Mathematical Monographs. Oxford University Press, New York, 1989.

[161] Open mpi: Open source high performance computing homepage. http://www.open-mpi.org/.

[162] Openmp homepage. http://openmp.org/wp/.

[163] Pacheco, P. S. *Programming parallel with MPI.* Morgan Kaufmann, San Francisco, 1997.

[164] Palczewski, A. *Równania różniczkowe zwyczajne. Teoria i metody numeryczne z wykorzystaniem komputerowego systemu obliczeń symbolicznych.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2004.

[165] Papadimitriou, C. Optimal sensor placement methodology for parametric identification of structural systems. *Journal of Sound and Vibration*, 278 (2004), 923–947.

[166] Papalambros, P. Y., and Wilde, D. J. *Principles of Optimal Design. Modeling and Computation*, 2nd ed. Cambridge University Press, Cambridge, 2000.

[167] Parallelknoppix homepage. http://pareto.uab.es/mcreel/parallelknoppix/.

[168] Pardalos, P. M., Pitsoulis, L. S., and Resende, M. G. C. A parallel grasp implementation for the quadratic assignment problem. In *Parallel Algorithms for Irregularly Structured Problems – Irregular'94* (1995), Kluwer Academic Publishers, pp. 111–130.

[169] Parhami, B. *Introduction to Parallel Processing: Algorithms and Architectures.* Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[170] Patan, M. *Optimal Observation Strategies for Parameter Estimation of Distributed Systems.* PhD thesis, University of Zielona Góra, Zielona Góra, Poland, 2004.

[171] Patan, M., and Patan, K. Optimal observation strategies for model-based fault detection in distributed systems. *International Journal of Control 78*, 18 (2005), 1497–1510.

[172] Pázman, A. *Foundations of Optimum Experimental Design.* Mathematics and Its Applications. D. Reidel Publishing Company, Dordrecht, The Netherlands, 1986.

[173] Pázman, A. *Nonlinear Statistical Models.* Mathematics and Its Applications. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.

[174] Point, N., Vande Wouwer, A., and Remy, M. Practical issues in distributed parameter estimation: Gradient computation and optimal experiment design. *Control Engineering Practice 4*, 11 (1996), 1553–1562.

[175] Portable extensible toolkit for scientific computation homepage. http://www-unix.mcs.anl.gov/petsc/petsc-as/.

[176] Powell, W. B. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics).* Wiley-Interscience, 2007.

[177] praca zbiorowa pod red. Andrzeja Karbowskiego i Ewy Niewiadomskiej-Szynkiewicz. *Obliczenia Równoległe i Rozproszone.* Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 2001.

[178] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, 2007.

[179] Pronzato, L., and Walter, É. Robust experiment design via stochastic approximation. *Mathematical Biosciences 75* (1985), 103–120.

[180] Pukelsheim, F. *Optimal Design of Experiments.* Probability and Mathematical Statistics. John Wiley & Sons, New York, 1993.

[181] Pytlak, R. *Numerical Methods for Optimal Control Problems with State Constraints.* Springer-Verlag, Berlin, 1999.

[182] Quereshi, Z. H., Ng, T. S., and Goodwin, G. C. Optimum experimental design for identification of distributed parameter systems. *International Journal of Control 31*, 1 (1980), 21–29.

[183] Rafajłowicz, E. Design of experiments for eigenvalue identification in distributed-parameter systems. *International Journal of Control 34*, 6 (1981), 1079–1094.

[184] Rafajłowicz, E. Optimal experiment design for identification of linear distributed-parameter systems: Frequency domain approach. *IEEE Transactions on Automatic Control 28*, 7 (July 1983), 806–808.

[185] Rafajłowicz, E. Optimization of actuators for distributed parameter systems identification. *Problems of Control and Information Theory 13*, 1 (1984), 39–51.

[186] Rafajłowicz, E. *Dobór sterowań optymalnych w identyfikacji systemów liniowych o parametrach rozłożonych.* Monografie. Wydawnictwo Politechniki Wrocławskiej, Wrocław, 1986.

[187] Rafajłowicz, E. Optimum choice of moving sensor trajectories for distributed parameter system identification. *International Journal of Control 43*, 5 (1986), 1441–1451.

[188] Rafajłowicz, E. *Algorytmy planowania eksperymentu z implementacjami w środowisku MATHEMATICA.* Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1996.

[189] Rafajłowicz, E. *Optymalizacja eksperymentu z zastosowaniami w monitorowaniu jakości produkcji*. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2005.

[190] Resende, M. G. Greedy randomized adaptive search procedures (grasp). *AT&T Labs Research Technical Report: 98.41.1* (12 1998).

[191] Resende, M. G., and Ribeiro, C. C. Greedy randomized adaptive search procedures. *State of the Art Handbook in Metaheuristics* (2002).

[192] Resende, M. G., and Ribeiro, C. C. Parallel greedy randomized adaptive search procedures. *Parallel Metaheuristics* (2005).

[193] Ricolindo L. Cari n., and Banicescu, I. Dynamic load balancing with adaptive factoring methods in scientific applications. *J. Supercomput. 44*, 1 (2008), 41–63.

[194] Rosenbrock, H. H. An automatic method for findong the greatest or least value of a function. *Computer Journal 4*, 3 (1960), 175–184.

[195] Rosenbrock method. http://www.applied-mathematics.net/.

[196] Schittkowski, K. *Numerical Data Fitting in Dynamical Systems — A Practical Introduction with Applications and Software*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[197] Schonveld, A., Lees, M., Karyadi, E., and Sloot, P. M. Dynamic load balancing in parallel finite element simulations. In *Proc. 7th Int. Conf. High-Performacne Computing and Networking,* Amsterdam, The Netherlands, 12–14 April 1999 (1999), vol. 1593, Springer-Verlag, pp. 409–419.

[198] Scott, L. R., Clark, T., and Bagheri, B. *Scientific Parallel Computing*. Princeton University Press, Princeton, NJ, USA, 2005.

[199] Sensors magazine : Shipboard machine monitoring for predictive maintenance. http://www.sensorsmag.com/sensors/article/articledetail.jsp?id=314716&pageid=1&sk=&date=.

[200] Silvey, S. D. *Optimal Design. An Introduction to the Theory for Parameter Estimation*. Chapman and Hall, London, 1980.

[201] Sinopoli, B., Sharp, C., Schenato, L., Schaffert, S., and Sastry, S. S. Distributed control applications within sensor networks. *Proceedings of the IEEE 91*, 8 (2003), 1235–1246.

[202] Snir, M., Otto, S. W., Walker, D. W., Dongarra, J., and Huss-Lederman, S. *MPI: The Complete Reference*. "MIT Press", "Cambridge, MA, USA", 1995.

[203] Sokołowski, J. *Zagadnienia analizy wrażliwości i optymalizacji parametrycznej w zadaniach sterowania optymalnego układów o parametrach rozłożonych*, prace naukowe - elektronika - z.13 ed. "Wyd. Politechniki Warszawskiej", Warszawa, 1985.

[204] Stoica, T. P. S. *Identyfikacja systemów.* Polish Scientific Publishers, Warszawa, 1997.

[205] Stojmenovic, I., Ed. *Handbook of Sensor Networks: Algorithms and Architectures.* John Wiley & Sons, 2005.

[206] Sun, N.-Z. *Inverse Problems in Groundwater Modeling.* Theory and Applications of Transport in Porous Media. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

[207] Sun grid engine homepage. http://www.sun.com/software/gridware/.

[208] Sunspotworld - project sun spot products homepage. http://sunspotworld.com/products/.

[209] Swissexperiment homepage – advanced process understanding and prediction of hydrological extremes and complex hazards. http://www.swiss-experiment.ch/index.php/apunch:home.

[210] Swissexperiment homepage – biosphere-geosphere interactions: Linking climate change, weathering, soil formation and ecosystem evolution. http://www.swiss-experiment.ch/index.php/biglink:home.

[211] Swissexperiment homepage – coupled seismogenic geohazards in alpine regions. http://www.swiss-experiment.ch/index.php/cogear:home.

[212] Swissexperiment homepage – triggering of rapid mass movements in steep terrain. http://www.swiss-experiment.ch/index.php/tramm:home.

[213] Swissexperiment homepage. http://www.swiss-experiment.ch/.

[214] Taillard, E. Robust tabu search for the quadratic assignment problem. *parcomput 17* (1991), 443–455.

[215] Tarnowski, W., and Bartkiewicz, S. *Modelowanie matematyczne i symulacja komputerowa dynamicznych procesów ciągłych. Wydanie 2.* Wydownictwa Pol. Koszalińskiej, Koszalin, 2000.

[216] Tassone, V., and Luus, R. Reduction of allowable values for control in iterative dynamic programming. *Chem. Eng. Sci. 48* (1993), 3864–3867.

[217] The beowulf project homepage. http://www.beowulf.org.

[218] Tomaś, A. *Projektowanie klastrów komputerów PC oraz metody zwiększania ich wydajności i niezawodności. Rozprawa doktorska.* Politechnika Częstochowska, Częstochowa, 2006.

[219] Top500 supercomputing sites. http://www.top500.org.

[220] Torsten Söderstriöm, P. S. *Identyfikacja systemów.* Polish Scientific Publishers, 1997.

[221] Tortorelli, D. A., and Michaleris, P. Design sensitivity analysis: Overview and review. *Inverse Problems in Engineering 1*, 1 (1994), 71–105.

[222] Trefethen, L. N. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations.* Lloyd N. Trefethen, http://www.comlab.ox.ac.uk/nick.trefethen/pdetext.html, 1994.

[223] Uciński, D. *Optymalizacja procesu pomiarowego w identyfikacji parametrycznej obiektów z czasoprzestrzenną dynamiką. Rozprawa doktorska.* Wydawnictwo Politechniki Wrocławskiej, Wrocław, 1992.

[224] Uciński, D. Optimal design of moving sensors trajectories for identification of distributed parameter systems. In *Proc. 1st Int. Symp. Methods and Models in Automation and Robotics,* Międzyzdroje, Poland, 1–3 September 1994 (Szczecin, 1994), vol. 1, Wyd. Uczelniane Polit. Szczecińskiej, pp. 304–309.

[225] Uciński, D. *Measurement Optimization for Parameter Estimation in Distributed Systems.* Technical University Press, Zielona Góra, 1999.

[226] Uciński, D. Optimal sensor location for parameter estimation of distributed processes. *International Journal of Control 73*, 13 (2000), 1235–1248.

[227] Uciński, D. Optimization of sensors' allocation strategies for parameter estimation in distributed systems. *Systems Analysis — Modelling — Simulation 37* (2000), 243–260.

[228] Uciński, D. *Optimal Measurement Methods for Distributed-Parameter System Identification.* CRC Press, Boca Raton, FL, 2005.

[229] Uciński, D., and Atkinson, A. C. Experimental design for time-dependent models with correlated observations. *Studies in Nonlinear Dynamics & Econometrics 8*, 2 (2004). Article No. 13.

[230] Uciński, D., and Chen, Y. Time-optimal path planning of moving sensors for parameter estimation of distributed systems. In *Proc. 44th IEEE Conference on Decision and Control, and the European Control Conference 2005,* Seville, Spain (2005). Published on CD-ROM.

[231] Uciński, D., and Korbicz, J. Optimization of sensors' allocation strategies for parameter estimation in distributed systems. In *Proc. Symp. Modeling, Analysis and Control of 2nd IMACS Int. Multiconf. CESA'98,* Nabeul-Hammamet, Tunisia, 1–4 April 1998 (1998), P. Borne, M. Ksouri, and A. El Kamel, Eds., vol. 1, pp. 128–133.

[232] Uciński, D., and Korbicz, J. Optimal sensor allocation for parameter estimation in distributed systems. *Journal of Inverse and Ill-Posed Problems 9*, 3 (2001), 301–317.

[233] Uciński, D., Korbicz, J., and Zaremba, M. B. On optimization of sensors motions in parameter identification of two-dimensional distributed systems. In *Proc. 2nd European Control Conf.,* Groningen, The Netherlands, 28 June–1 July 1993 (1993), J. W. Neuwenhuis, C. Praagman, and H. L. Trentelman, Eds., vol. 3, ECCA, pp. 1359–1364.

[234] Uciński, D. Sensor network design for parameter estimation of distributed systems using nonsmooth optimality criteria. In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference - CDC-ECC '05; ISBN: 0-7803-9568-9* (Seville, Hiszpania, 2005), [B. m.], pp. [6] CD–ROM.

[235] Uciński, D., and Patan, M. D-optimal design of a monitoring network for parameter estimation of distributed systems. *Journal of Global Optimization Vol. 39*, no 2 (2007), 291–322.

[236] Uciński, D., Patan, M., and Kuczewski, B. Sensor network design for identification of distributed parameter systems. In *Measurements models systems and design, ISBN: 978-83-206-1644-6*, ed. by J. Korbicz, Ed. Wydaw. Komunikacji i Łączności, Warszawa, 2007, pp. 121–155.

[237] Uspenskii, A. B., and Fedorov, V. V. *Computational Aspects of the Least-Squares Method in the Analysis and Design of Regression Experiments.* Moscow University Press, Moscow, 1975. (In Russian).

[238] Valavanis, K. P. *Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy (Intelligent Systems, Control and Automation: Science and Engineering).* Springer, 2007.

[239] van de Wal, M., and de Jager, B. A review of methods for input/output selection. *Automatica 37* (2001), 487–510.

[240] Vande Wouwer, A., Point, N., Porteman, S., and Remy, M. On a practical criterion for optimal sensor configuration — Application to a fixed-bed reactor. In *Proc. 14th IFAC World Congress,* Beijing, China, 5–9 July, 1999 (1999), vol. I: Modeling, Identification, Signal Processing II, Adaptive Control, pp. 37–42.

[241] Walter, É., and Pronzato, L. Robust experiment design: Between qualitative and quantitative identifiabilities. In *Identifiability of Parametric Models*, É. Walter, Ed. Pergamon Press, Oxford, 1987, pp. 104–113.

[242] Walter, É., and Pronzato, L. Qualitative and quantitative experiment design for phenomenological models — A survey. *Automatica 26*, 2 (1990), 195–213.

[243] Walter, É., and Pronzato, L. *Identification of Parametric Models from Experimental Data*. Communications and Control Engineering. Springer-Verlag, Berlin, 1997.

[244] Weinberger, H. F. *Partial Differential Equations with complex variables and transform methods*. Dover Publications, New York, 1995.

[245] Wilson, J. S. *Sensor Technology Handbook*. Elsevier, Burlington, USA, 2005.

[246] Windows hpc: Home page. http://www.microsoft.com/hpc.

[247] Wu, J. *Handbook On Theoretical And Algorithmic Aspects Of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*. Auerbach Publications, Boston, MA, USA, 2005.

[248] Wyrzykowski, R., Meyer, N., and Stroiński, M. Clusterix: National cluster of linux systems. In *Proc. 2nd European Across Grids 2004 Conf.* (Nicosia, Cyprus, 2004), http://grid.ucy.ac.cy/axgrids04/AxGrids.

[249] Yuanqiao, W., Yu, S., Zhou, J., and Huang, L. Agent based distributed parallel tunneling algorithms. In *PDCAT* (2004), pp. 226–229.

[250] Zaki, M. J., Li, W., and Parthasarathy, S. Customized dynamic load balancing for a network of workstations. *J. Parallel Distrib. Comput. 43*, 2 (1997), 156–162.

[251] Zięba, T., and Baranowski, P. A parallel technique of optimal sensor selection for distributed parameter systems subject to correlated observations. In *1st. International Conference for Young Researchers in Computer Science, Control, Electrical Engineering and Telecommunications - ICYR : abstracts* (Zielona Góra, Polska, 2006), Faculty of Electrical Engineering, Computer Science and Telecommunications, University of Zielona Góra, Zielona Góra, p. 97.

[252] Zięba, T., and Uciński, D. Mobile sensor routing for parameter estimation of distributed systems using the parallel tunneling method. *International Journal of Applied Mathematics and Computer Science Vol. 18*, no 3 (2008), 307–318.

[253] Ziółko, M. *Modelowanie zjawisk falowych.* Uczelniane Wydawnictwa Naukowo-Dydaktyczne, Kraków, 2000.