Instytut Informatyki, Automatyki i Robotyki

Seria: PREPRINTY nr 35/2006

Czas wykonania programów współbieżnych dla modeli synchronizacji opisanych siecią Petri

(rozprawa doktorska)

Wojciech Noworyta

Promotor: dr hab. inż. Janusz Biernat, prof. PWr

Słowa kluczowe:

- rozkład losowy
- czas realizacji
- synchronizacja procesów
- współbieżność
- BSP

Wrocław,15.06.2006 r.

Oznaczenia

GPS	– Graf Przepływu Sterowania
GRP	– Graf Realizacji Programu
G	– zbiór procesorów maszyny
J	– funkcja przyporządkowująca przejściom sieci, dystrybuantę rozkładu losowego czasu
$K = \{ k_1, k_2, ...\}$	– zbiór krawędzi sieci Petriego
$L = \{ l_1, l_2, ...\}$	– zbiór przejść sieci Petriego
$M = \{ m_1, m_2, ...\}$	– zbiór miejsc sieci Petriego
P	– proces (wątek) programu
PGM	– program
PGM_i	– program dla i-tego procesu (fragment PGM)
$Q = \{ q_1, q_2, ...\}$	– zbiór ścieżek w grafie
r	– instrukcja
R	– zbiór instrukcji
SSG	– stan sprawności procesora
SP	– stan procesu
SPG	– stan układu procesor + proces
t	– czas
U	– wektor określający czas przebywania znaczników we wszystkich miejscach sieci
U_0	– stan początkowy sieci Petriego
γ	– liczba stanów sprawności procesora
Γ	– liczba stanów sprawnych procesora ($1 \leq \Gamma \leq \gamma$)
Λ	– macierz intensywności przejść procesu Markowa uszkodzeń i napraw
λ	– intensywność przejścia procesu Markowa
η	– efektywność wykonywania kodu przez (częściowo) sprawną maszynę
$\Phi(t)$	– dystrybuanta rozkładu losowego czasu wykonania operacji
$\|\|X\|\|$	– liczność zbioru X
*	– splot dystrybuant (dla sumy zmiennych losowych)
$\underline{\mathbf{1}}$	– funkcja Heaviside'a (jedynka Heaviside'a)

Wstęp

We współczesnym świecie systematycznie wzrasta zapotrzebowanie na maszyny liczące o coraz większej mocy przetwarzania. Dotyczy to zarówno zapotrzebowania na „czystą moc obliczeniową", jak również na szybki dostęp do rozbudowanych baz danych czy też przetwarzanie potężnych strumieni danych generowanych przez urządzenia telekomunikacyjne.

Jednym ze sposobów zapewnienia wymaganych parametrów jest zastosowanie maszyn zdolnych do równoległej realizacji wielu zadań. Ze względu na trudności technologiczne w produkcji coraz szybszych układów cyfrowych rośnie zainteresowanie maszynami wykorzystujących współbieżność dla osiągnięcia wymaganej szybkości przetwarzania. Rachunek ekonomiczny oraz względy niezawodności również przemawiają na korzyść maszyn równoległych.

Współczesne maszyny cyfrowe stają się jednak coraz bardziej skomplikowane, uniemożliwiając użytkownikowi szczegółowe poznanie i zrozumienie detali ich architektury. Na początku lat 80-tych praktycznie każdy mógł w pamięci policzyć ile cykli maszynowych zajmie procesorowi wykonanie danej procedury. Obecnie wymagałoby to dokumentacji niedostępnej zwykłym użytkownikom, bardzo złożonego symulatora VHDL i niezwykle precyzyjnego określenia warunków początkowych. Dla maszyn równoległych pojęcie wartości pamięci pod danym adresem również nabrało nowego wymiaru. W modelach słabo-spójnych poszczególne procesory mogą „widzieć" różną wartość pod tym samym adresem.

Czas wykonania aplikacji współbieżnych nie jest stały i coraz częściej do jego opisu używa się parametrów statystycznych – wartość średnia, maksymalna, najbardziej prawdopodobna. Stochastyczna analiza czasu realizacji programów zyskuje na popularności ze względu na coraz większą liczbę aplikacji o miękkich ograniczeniach czasowych (np. multimedia), jak również uruchamiania aplikacji czasu rzeczywistego na powszechnie dostępnym sprzęcie i oprogramowaniu systemowym.

W ostatnich latach większą uwagę przykłada się również do kształtu i właściwości rozkładu losowego czasu realizacji programu (rozmiaru pliku, ruchu sieciowego). Wyniki badań wskazują przy tym na zdecydowanie większą zmienność obserwowanych parametrów niż wynikałoby to z modeli stosowanych w latach uprzednich.

Cel i tezy pracy

Celem pracy jest opracowanie metody obliczania szybkości realizacji programu współbieżnego w obecności uszkodzeń. Przyjęto, że uszkodzenia są niezależnymi zdarzeniami losowymi, jak również, że czas realizacji programu nie jest stały i podlega pewnej zmienności losowej. Ze względu na przyjęty sposób realizacji celu pracy autor posługuje się modelem przetwarzania, który jest złożeniem osobnych modeli programu i maszyny. Przyjęte założenia i sposób realizacji celu powinny doprowadzić do wyznaczenia nie tylko średnich czasów realizacji programów, ale także ich rozkładów losowych.

W pracy zaproponowany został model realizacji programu współbieżnego, oparty o sieć Petriego. Zawarte w nim konstrukcje umożliwiają zapis synchronizacji procesów wykorzystujących pamięć dzieloną, buforowane i niebuforowane komunikaty, jak również synchronizację ogólną BSP (Bulk Synchronous Processing). Realizacja programu jest przedstawiana i analizowana jako proces stochastyczny o strukturze i parametrach zależnych od treści programu, wydajności i niezawodności maszyny. W opisie realizacji programu współbieżnego przyjęto uproszczony model maszyny cyfrowej, redukując jej opis do zestawu współczynników określających statystyczne parametry. Takie podejście umożliwia przybliżone obliczenie czasu wykonywania aplikacji przy użyciu maszyny, której model architektury jest nieznany, natomiast dostępny jest statystyczny opis jej parametrów użytkowych. W pracy podano również propozycje dotyczące sposobów określania parametrów analizowanego programu i maszyny. Następnie przeprowadzono eksperymentalną weryfikację prezentowanej metody.

Sformułowano następujące tezy pracy:

1. Możliwe jest przedstawienie programu równoległego ze statycznie definiowanymi instrukcjami synchronizacji w postaci uproszczonej stochastycznej sieci Petriego typu PERT. Rozmiar grafu jest liniowo zależny od ilości instrukcji synchronizacji. Za pomocą uzyskanego grafu można obliczyć kształt rozkładu losowego czasu realizacji programu.
2. Czas realizacji programów współbieżnych może być traktowany jako zmienna losowa, której rozkład jest determinowany przez strukturę programu oraz realizującej go platformy.
3. Rozkład losowy czasu realizacji całej aplikacji nie dąży do kształtu normalnego, mimo że jest sumą (bardzo) wielu zmiennych losowych (czasów realizacji poszczególnych procedur). Rozkłady losowe czasów realizacji fragmentów programu nie spełniają założeń centralnego twierdzenia granicznego.
4. Rozkład losowy czasu realizacji programu (dla większości aplikacji) należy do klasy „heavy tail”, czyli ma w przybliżeniu kształt hiperboliczny. Rozkłady tej klasy mają nieskończoną wariancję.

Struktura pracy

W rozdziale pierwszym opisano podstawowy model sieci Petriego oraz jego rozszerzenia umożliwiające uwzględnienie upływu czasu oraz niedeterministyczne zachowanie programu. Następnie zaproponowano klasyfikację metod synchronizacji i ich modele sieciowe. Przedstawiono też wybrane sposoby reprezentacji uszkodzeń za pomocą sieci Petriego.

Drugi rozdział zawiera model programu i jego struktury logicznej. Przedstawiono opis grafu przepływu sterowania programu współbieżnego oraz jego reprezentacji w postaci sieci Petriego. Opisano metody konstrukcji struktury sieci, a następnie określania parametrów opisujących każde z przejść. Zaproponowano kilka sposobów redukcji rozmiarów grafu prowadzących do uzyskania uproszczonego modelu. Przedstawiono algorytm acyklizacji grafu prowadzący do uzyskania sieci klasy PERT.

W trzecim rozdziale przedstawiono model zawodnej maszyny cyfrowej zdolnej do spowolnienia pracy w wyniku uszkodzeń (graceful degradation). Przedstawiono proces Markowa uszkodzeń i napraw powiązany z jednoczesnym procesem semi-Markova wykonywania programu. Zaprezentowano uproszczenia mające na celu otrzymanie grafu acyklicznego.

Czwarty rozdział zawiera opis połączenia modelu zawodnej maszyny cyfrowej i grafu przepływu sterowania programu. Podana jest również metoda połączenia grafu przepływu sterowania z modelem maszyny, prowadząca do otrzymania grafu realizacji programu. Opisane zostały również metody uzyskania rozkładu losowego czasu wykonania programu na podstawie grafu realizacji programu.

Piąty rozdział zawiera wyszczególnienie założeń modelu, jakie należy zweryfikować doświadczalnie. Przedstawiony jest skrócony opis wykonanych eksperymentów oraz wnioski uzyskane na podstawie otrzymanych wyników.

Szósty rozdział przedstawia doświadczenia mające za zadanie weryfikację opracowanej metody analizy szybkości wykonywania programu. Podane są przykłady testowe, przewidywania teoretyczne, co do ich zachowania oraz wyniki doświadczeń.

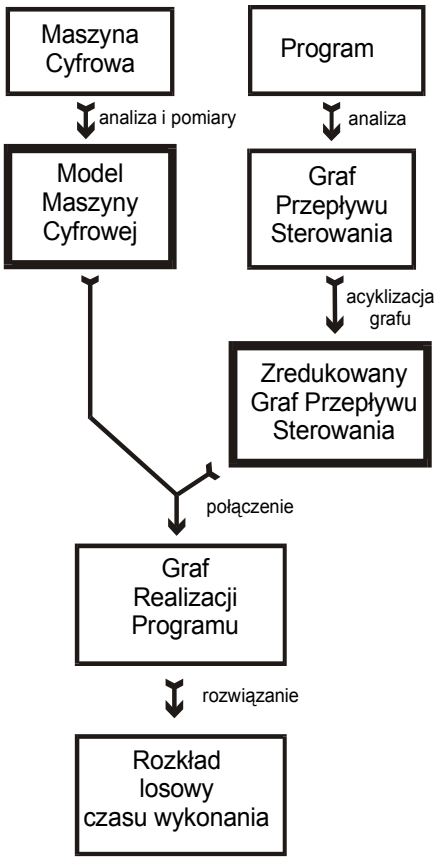
Siódmy rozdział stanowi podsumowanie i zakończenie pracy.

Dodatek A zawiera szczegóły doświadczalnej weryfikacji założeń modelu. Przedstawiono zestaw metod pomiaru szybkości wykonywania kodu przez procesor uzasadniając wybór jednej z nich. Następnie określono wzajemną korelację czasu realizacji fragmentów kodu realizowanych przez ten sam procesor oraz procesory współpracujące.

Dodatek B zawiera opis badań charakteru rozkładów losowych czasu realizacji programu i jego fragmentów. Przedstawiono propozycje dotyczące wyboru optymalnej reprezentacji rozkładów losowych oraz sposobów dokonywania na nich przekształceń i obliczeń. Opisano próby teoretycznego przewidzenia kształtów rozkładów losowych oraz ich konfrontacja z danymi zebranymi doświadczalnie. Szczególną uwagę zwrócono na wolną zbieżność rozkładu i związane z tym niekorzystne właściwości.

Szkic metody analizy zastosowanej w pracy

Wzajemne zależności pomiędzy modelami i opisami użytymi w pracy przedstawia rys. 1. Punktem wyjścia jest program poddawany analizie oraz maszyna cyfrowa, na której będzie on realizowany. Pierwszym etapem jest konstrukcja modelu programu i maszyny uwzględniająca parametry, które będą użyte do dalszych obliczeń. Modele zaznaczone grubszą ramką są od siebie niezależne. Raz opracowane mogą służyć do analizy innych zestawień maszyna + program.



Rys. 1. Schemat metody analizy wykonywania programu

Maszyna cyfrowa opisana jest w formie procesu zmian stanów niezawodności. Dla każdego stanu określone są rozkłady losowe prawdopodobieństwa przejścia do innego stanu sprawności, jak również wykonania kolejnego kroku obliczeń. Prawdopodobieństwo przejścia zależy zarówno od czasu, jak też złożoności bloku instrukcji, jaki maszyna w tym momencie realizuje.

Na podstawie treści programu tworzony jest odpowiadający mu graf przepływu sterowania. Opisuje on kolejność wykonywania instrukcji oraz zależności pomiędzy współpracującymi procesami. Graf przepływu sterowania przedstawiony jest za pomocą sieci Petriego, której odpowiednie fragmenty odpowiadają konstrukcjom logicznym programu (pętla, alternatywa, synchronizacja). Każde przejście sieci zawiera informację na temat rozkładu losowego czasu, jaki jest wymagany, aby zrealizować modelowany fragment. Zastosowanie rozkładów losowych umożliwia uwzględnienie zmiennej ilości wykonanych instrukcji w zależności od przetwarzanych danych, jak również prawdopodobieństwo występowania losowych uszkodzeń oprogramowania.

Kolejnym etapem analizy jest redukcja grafu przepływu sterowania do możliwie najmniejszych rozmiarów. Udowodniono, że dla statycznie zdefiniowanych synchronizacji pomiędzy procesami rozmiar grafu jest ograniczony z góry liniową kombinacją ilości synchronizacji. Uzyskiwany jest zatem graf o rozmiarach umożliwiających analizę dostępnymi narzędziami.

Kolejnym etapem jest połączenie modeli programu i maszyny. W wyniku zespolenia otrzymywany jest graf realizacji programu uwzględniający parametry maszyny, w tym możliwość jej całkowitego lub częściowego uszkodzenia. Liczba miejsc otrzymanego grafu jest równa iloczynowi ilości miejsc zredukowanego grafu

przepływu sterowania oraz liczby stanów sprawności maszyny. Od skuteczności redukcji GPS zależy, w dużej mierze rozmiar zagadnienia i stopień komplikacji jego realizacji.

Graf realizacji programu analizowany jest począwszy od stanu początkowego, poprzez kolejne przejścia aż do stanu końcowego. Zadaniem analizy jest określenie rozkładu losowego czasu odpalenia każdego z przejść. Czas ten określony ma być względem początku realizacji programu. Rozkład losowy odpalenia końcowego przejścia odpowiada rozkładowi losowemu czasu realizacji całego programu. Uwzględnia on przy tym prawdopodobieństwo niemożności zakończenia obliczeń na skutek uszkodzenia maszyny lub programu.

Przegląd literatury

Stochastyczne Sieci Petriego

W niniejszej pracy do opisu procesów współbieżnych zdecydowano się użyć sieci Petriego [Pet62]. Elastyczność i ogólność modelu sieci Petriego umożliwia reprezentację praktycznie każdego problemu z dziedziny analizy wykonywania programu [Cin81], [Deg88] [Mol82], [Jar87], [Old97], [Alp97]. Szczególnym uznaniem cieszą się modele uwzględniające upływ czasu wyrażonego w postaci liczb [Coo83], [Mur80], [Ram80], [Sif80] lub też rozkładów losowych [Ajm95], [Bau96], [Mol82], [Wan98]. Poszczególni autorzy wiążą pojęcie upływu czasu z miejscami [Coo83], [Sif80] lub przejściami [Mol82], [Ram80], [Jar87]. Rzadziej spotyka się powiązania upływu czasu ze znacznikami lub krawędziami sieci.

Czas odpalania przejść opisywany jest:

- stałą wartością (czasu)
- rozkładem wykładniczym [Alp97], [Amm85]
- rozkładem hipo-wykładniczym (hypo-exp) [Mag93]
- rozkładem normalnym [Kam85a], [Kam86]
- dowolnym rozkładem losowym [Jar87], [Kam87]
- rozkładem Erlanga [Abd03]
- rozkładem Weibula [Abd04]
- stałą ilością instrukcji (przeliczaną na czas trwania w późniejszych etapach analizy)
- stałym wektorem ilości instrukcji o trzech współrzędnych [Val90], [Mcc94], [Ger96]

Ponieważ nie istnieje rozkład losowy (o niezerowej wariancji), dla którego suma zmiennych losowych oraz minimum (maksimum) zmiennych losowych posiadałyby rozkład tego samego typu, autorzy proponują metody przybliżone [Kam85b], [Mag93], [Dod85] lub numeryczne. Szczególny nacisk kładziony jest na określenie górnego i dolnego ograniczenia rozkładu wynikowego [Kam85], [Kam85c], [Kam87], [Dod85a], co pozwala użyć model sieciowy do planowania przedsięwzięć oraz procesów produkcyjnych i związanego z tym ryzyka.

Część autorów proponuje analizę jedynie wartości oczekiwanych, co zbliża zagadnienie do znacznie prostszego problemu sieci z deterministycznymi obciążeniami łuków [Ave85] [Kam86]. Poszukiwane są też bardziej dokładne oszacowania [Kam87].

Od czasu upowszechnienia się modelu Petriego pojawiło się bardzo dużo prac dotyczących analizy struktury grafu sieci i jego użycia do modelowania wykonywania aplikacji [Lin98] oraz współbieżności [Bra86], [Bal95]. Podano szereg algorytmów badających osiągalność stanów, żywotność, ograniczoność, bezpieczeństwo oraz inne właściwości sieci. Opracowano algorytmy redukcji rozmiaru sieci dla poszczególnych rodzajów modeli sieciowych [Amm85], [Buc95], [Buc97], [Jar87], [Mag89]. Publikowano również prace z zakresu określania złożoności algorytmów bazujących na analizie sieci [Mag84], [Kam87].

Sieci PERT

Szczególne znaczenie w analizie sieciowej mają skierowane grafy acykliczne opisujące relacje wzajemnej zależności przyczynowo – skutkowej. Sieci typu PERT znajdują szerokie zastosowanie w opisie i modelowaniu przedsięwzięć gospodarczych, stanowisk produkcyjnych, jak również złożonych układów elektronicznych.

Opracowano algorytmy określania maksymalnej i minimalnej drogi, jak również maksymalnego i minimalnego przepływu [Dod84]. Dla sieci ze stałymi obciążeniami łuków stworzono algorytmy o wielomianowej złożoności, natomiast dla sieci stochastycznych udowodniono NP-trudność zagadnienia [Kam87]. Pokazano jednocześnie, że dla ogólnych sieci Petriego złożoność analizy jest zdecydowanie większa niż dla sieci PERT [Mag84], [Mag89], [Mag91].

Zaproponowano metody przybliżone umożliwiające określenie górnego i dolnego ograniczenia wartości najdłuższej drogi lub też rozkładów ograniczających z góry i z dołu rozkład losowy czasu realizacji przedsięwzięcia opisanego siecią PERT [Dod85], [Dod85a], [Elm89] [Fis83], [Fis85], [Kam85], [Mag93]. Dla potrzeb modelowania zagadnień gospodarczych opracowano przybliżone metody umożliwiające dokonywanie obliczeń przy niepełnej informacji probabilistycznej na temat obciążeń łuków [Kam85b], [Kam87].

Zaproponowano również sposoby zmniejszania rozmiaru sieci PERT [Kam92]

W celu stworzenia doskonalszego modelu przy zachowaniu korzystnych właściwości sieci PERT rozważano możliwość połączenia ich z modelami kolejkowymi, stochastycznymi sieciami Petriego [Mag89a].

Określenie szybkości maszyny i złożoności programu

O momentu powstania maszyn liczących opracowywano sposoby określenia ich wydajności umożliwiające porównanie różnych konkurencyjnych komputerów, jak również oszacowanie czasu wykonywania aplikacji. Ze względu na różnorodność dostępnego sprzętu nie udało się opracować metody umożliwiającej precyzyjne i proste opisanie wydajności maszyny i szybkości realizacji programu. Najbardziej znanymi parametrami określającymi szybkość komputera są:

- przepustowość czyli liczba instrukcji wykonywanych w jednostce czasu (MIPS, MFloPS)
- wydajność czyli liczba instrukcji wykonywanych w jednym cyklu zegara

Prezentowane miary stanowią bardzo przybliżone oszacowanie możliwości sprzętu ze względu na znaczące różnice list rozkazów procesorów oraz zależność szybkości wykonywania instrukcji od ich rodzaju i kolejności [Bie01]. Proponowano również określenie wydajności komputera na podstawie czasu realizacji konkretnego złożonego zadania matematycznego. Powstało tysiące programów testowych (benchmarków), z czego najbardziej znane to SpecInt i SpecFp określające szybkość operacji stało- i zmiennoprzecinkowych. Ze względu na przybliżony charakter opisanych metod nie przyjęło się określanie złożoności zadania (programu) jako liczby instrukcji koniecznych do wykonania, bądź też odpowiedniej ilości punktów SpecInt czy SpecFp. Nie opracowano metody umożliwiającej niezależne określenie parametrów dowolnego problemu (programu) oraz maszyny, na podstawie których można by w sposób uniwersalny przewidzieć szybkość wykonywania każdego zadania na każdej maszynie.

Wprowadzenie maszyn zdolnych do równoległego wykonywania wielu operacji dodatkowo skomplikowało problem analizy ich szybkości. Zidentyfikowano szereg zjawisk utrudniających określenie na drodze teoretycznej czasu realizacji zadania na maszynie równoległej [Ben89] [Col98] [Tan95]. Najważniejsze z nich to:

- Problem rozproszenia obliczeń.

  Większość algorytmów stworzono dla maszyn sekwencyjnych. Wykonanie ich współbieżnie wymaga zwykle poważnych modyfikacji algorytmu a często jego zupełnej zmiany. Ponadto różne maszyny, ze względu na odmienny sposób interakcji równolegle wykonywanych instrukcji, mogą wymagać różnych algorytmów.
- Problem skalowalności.

  Z faktu, że jeden procesor potrzebuje dla realizacji pewnego zadania N jednostek czasu wcale nie wynika, że N procesorów zdoła wykonać tę pracę w jednej jednostce czasu.
- Problem komunikacji.

  Równoległa praca wielu jednostek obliczeniowych nad jednym zadaniem wymusza konieczność efektywnej komunikacji. Jest to jeden z ważniejszych elementów wpływających na szybkość systemu równoległego.
- Problem synchronizacji.

  Większość algorytmów projektowanych z myślą o maszynach współbieżnych wymaga określonej synchronizacji poszczególnych etapów przetwarzania pomiędzy uczestniczącymi procesorami. Oznacza to między innymi konieczność bezczynnego oczekiwania pewnych procesorów (procesów) na zakończenie operacji przez inne.
- Różne modele współbieżności.

  Zaproponowano kilka modeli współbieżności określających sposoby wzajemnej komunikacji i synchronizacji procesów [Hoa78], [MPI95], [Val90]. Na tej podstawie zbudowano wiele architektur różniących się w stopniu uniemożliwiającym wykorzystanie tego samego algorytmu (tym bardziej programu).

Przegląd stosowanych modeli współbieżności

Konieczność zastosowania maszyny zdolnej do współbieżnego wykonywania wielu operacji wynika najczęściej z ograniczeń technologii uniemożliwiających uzyskanie odpowiednio szybkiej maszyny sekwencyjnej. Ze względu na zalety modelu von Neumana, jego bogaty opis i przyzwyczajenia użytkowników, dąży się do konstruowania maszyn i modeli możliwie podobnych do pierwowzoru, uwzględniających jednak jednoczesność działań. Proponowane rozwiązania można podzielić na trzy rodzaje:

- Modele zakładające istnienie N procesów wykonujących się jednocześnie i wymieniających pomiędzy sobą dane. Interakcja może zachodzić zarówno poprzez istnienie obszaru pamięci dzielonej jak też poprzez przesyłanie komunikatów. W klasyfikacji Flynna [Fly95] modele te znajdują się w kategorii MIMD. Ze względu na sposób zrównoleglenia operacji zespół współbieżnych procesów opisywany jako równoległość zgrubna (ang. coarse grain).
- Modele opisujące możliwości zrównoleglenia złożonych operacji na obiektach przetwarzanych przez pojedynczy proces. Typowymi przykładami są modele procesorów wektorowych, systolicznych, użycia N jednostek przetwarzania w jednym procesorze. W klasyfikacji Flynna modele te znajdują się częściowo w kategorii SIMD, częściowo w SISD, częściowo poza klasyfikacją. Ze względu na sposób zrównoleglenia operacji modele opisywane są jako równoległość drobna.
- Modele niekonwencjonalne, odrzucające koncepcję von Neumana i poszukujące alternatywnych rozwiązań. Można tu wspomnieć o modelach przetwarzania sterowanego przepływem danych, sieciach neuronowych.

W dalszej części pracy rozpatrywane będą jedynie rozwiązania z pierwszej grupy, czyli zbiór oddzielnych procesów wykonywanych na osobnych maszynach klasy von Neumana wzajemnie się ze sobą komunikujących i synchronizujących. Nie wyklucza to jednoczesnego istnienia mechanizmów równoległości wewnątrz procesorów składowych, jednakże w pracy nie będą one rozpatrywane. Analizowane w dalszej części pracy modele współbieżności można podzielić na trzy podstawowe grupy:

- Modele pamięci dzielonej silnie i słabo spójnej.
- Modele synchronicznego i asynchronicznego przesyłania komunikatów.
- Modele synchronizacji ogólnej Bulk Synchronous Processing (BSP).

Dwa pierwsze modele są opisane w bardzo wielu pozycjach literatury [Hoa78], [MPI95] i zakłada się, że są powszechnie znane. Bardzo bogata jest również literatura zajmująca się problemem analizy realizacji programów opartych o te modele [Cin81], [Deg88], [Mag92], [Sch94]. Model BSP [Val90], [Mcc93], [Mcc94] jest mniej popularny.

Model BSP

Twórcom modelu BSP przyświecały nieco inne cele niż autorom wcześniejszych modeli. Skupiono się na opracowaniu sposobu tworzenia aplikacji współbieżnych, który byłby prosty w opisie i analizie. W założeniu program napisany zgodnie z zaleceniami BSP powinien na każdej maszynie wykonywać się w taki sam, bardzo przewidywalny sposób. Różnice mogą, co najwyżej dotyczyć globalnej szybkości realizacji.

W modelu BSP rozpatruje się wykonywanie zadania jako sekwencję super-kroków. W trakcie każdego z nich procesy nie komunikują się ze sobą, prowadząc obliczenia na lokalnych danych. Wszelkie żądania dostępu do nielokalnych danych są kolejkowane i wykonywane po zakończeniu obliczeń. Każdy super-krok zakończony jest globalną barierą wymuszającą dokończenie zaległych operacji komunikacji oraz synchronizację wszystkich procesów.

Model BSP charakteryzuje się następującymi cechami:

- szeregową strukturą (super-kroków),
- rozdzieleniem obliczeń, komunikacji i synchronizacji,
- silną synchronizacją procesów, często nadmiarową,
- dopuszczeniem obszarów pamięci wspólnej procesów,
- słabą spójnością wspólnych obszarów pamięci (weak cosistency).

Dzięki bardzo silnej wzajemnej synchronizacji procesów, wykonywanie programu opartego o model BSP można przedstawić w postaci łańcucha super-kroków. W każdym ogniwie uczestniczą wszystkie procesy. Czas wykonywania aplikacji jest, zatem taki sam dla wszystkich uczestniczących w niej procesów i jest równy sumie czasów trwania każdego z kroków. Co więcej czas trwania każdego kroku jest taki sam dla wszystkich procesów aplikacji.

Twórcy modelu BSP zaproponowali metodę obliczania czasu trwania każdego z kroków. Określono go jako czas wykonywania części obliczeniowej procesu o największej ilości obliczeń zsumowany z czasem koniecznym do przesłania wymaganej ilości danych oraz czasem wymaganym dla osiągnięcia globalnej synchronizacji.

$$t = t_i N_i + t_g N_g + t_b N_b$$

Uznano, że dla maszyny zdolnej realizować program znane będą następujące trzy parametry:

- średni czas wykonywania pojedynczej instrukcji obliczeniowej t_i,
- średni czas przesłania bajtu pomiędzy maszynami t_g,
- średni czas wykonania instrukcji globalnej synchronizacji t_b.

Na podstawie analizy programu można będzie określić następujące parametry:

- liczba instrukcji obliczeniowych N_i,
- liczba bajtów do przesłania N_g,
- liczba instrukcji synchronizacji N_b.

W modelu BSP przyjęto, że użycie wartości średnich wszystkich parametrów umożliwi uzyskanie wystarczająco dobrego przybliżenia średniej wartości czasu realizacji programu.

Śmiały pomysł twórców modelu BSP, niezależnego określenia parametrów programu i maszyny, aby następnie przewidzieć zachowanie dowolnego programu na dowolnej maszynie współbieżnej był podstawą badań zawartych w prezentowanej pracy. Stanowi ona próbę rozwiązania zadanego problemu dla mniej restrykcyjnych założeń odnośnie struktury i synchronizacji programu, kosztem ewentualnej komplikacji modelu i obliczeń.

Cechy wspólne rozpatrywanych modeli współbieżności

- Sprzęt, oprogramowanie i dane

Każdy z wymienionych modeli zakłada istnienie (fizycznego lub symulowanego) uniwersalnego układu cyfrowego zdolnego do wykonywania operacji matematycznych i logicznych. Aby jakiekolwiek (użyteczne) przekształcenia mogły zostać wykonane, konieczne jest opracowanie algorytmu i na jego podstawie programu sterującego maszyną. Modele zakładają, że operacje wykonywane przez maszynę na podstawie programu nie powodują zmiany maszyny ani programu. Przekształceniom podlegają informacje klasyfikowane jako dane. Między poszczególnymi elementami (maszyna, program, dane) istnieje relacja jeden do wielu. Konkretna maszyna jest w stanie wykonywać jeden z wielu programów, konkretny program dla podanej maszyny akceptuje wiele zestawów danych wejściowych i dla każdego z nich powinien wygenerować sensowny wynik.

- Równoległość sekwencyjnych bloków

Cechą wspólną rozpatrywanych modeli współbieżności jest założenie o równoległej pracy wielu sekwencyjnych układów. Równoległość zgrubna (ang. coarse grain) zakłada możliwie niezależną pracę poszczególnych procesorów maszyny realizujących oddzielne bloki programu. Ich wzajemna interakcja zachodzi jedynie w pewnych, założonych przez programistę momentach. Każdy blok programu jest realizowany w sposób sekwencyjny, co nie wyklucza mechanizmów SIMD w obrębie jednego bloku.

- Lokalność

Wszystkie rozpatrywane modele współbieżności zakładają lokalność przetwarzania wyrażającą się określeniem procesu jako odrębnej całości, jak też lokalnych i globalnych danych. Nawet model pamięci dzielonej pod presją rzeczywistych rozwiązań rozróżnia odwołania lokalne i globalne. Modele opisują działanie pojedynczego procesu w sposób możliwie najbardziej zbliżony do powszechnie akceptowanego modelu Von Neumana natomiast różnią się sposobem interakcji pomiędzy procesami.

- Przekazywanie informacji

Rozpatrywane modele współbieżności zakładają pełny dostęp do informacji lokalnej – w obrębie jednego procesu. Wymiana informacji pomiędzy procesami odbywa się natomiast za pomocą specjalnych mechanizmów określanych przez model. W zależności od modelu do komunikacji służą instrukcje podobne do manipulujących danymi lokalnymi lub też wyraźnie się od nich różniące.

Rozkład losowy czasu realizacji programu

Wcześniejsze generacje komputerów, systemów operacyjnych i programów wykazywały zdecydowanie bardziej deterministyczne działanie niż najnowsze konstrukcje. Problem nieregularnej, losowej zmienności czasu realizacji aplikacji był przez pewien czas odsuwany na plan dalszy, natomiast obecnie zyskuje na znaczeniu. Stosowane modele czasu realizacji programu można podzielić na:

- modele w pełni deterministyczne
- modele deterministyczne wielościeżkowe (instrukcje warunkowe)
- analiza najdłuższej ścieżki (worst case execution time, WCET)
- modele Markowa (wykładniczy rozkład losowy)
- modele semi-Markowowskie (dowolny rozkład losowy) [Ger01]
- modele uwzględniające korelację (nie-Markowowskie)

Najczęściej spotykane są modele umożliwiające oszacowanie WCET. Wynika to z ich względnej prostoty obliczeniowej – model prowadzi często do poszukiwania najdłuższej drogi w grafie o stałych obciążeniach łuków. Wynikiem analizy jest liczba, co niezwykle ułatwia łączenie mniejszych modeli w jedną całość.

Coraz więcej badaczy określa jednak założenia WCET jako zbyt pesymistyczne dla zdecydowanej większości zastosowań. Postulują określanie czasu realizacji aplikacji jako rozkładu losowego (a nie jednej liczby). Na tej podstawie można określić, z jakim prawdopodobieństwem proces „zmieści" się w narzuconym przedziale czasowym [Pet02].

Wraz z ideą składania programu z komponentów popularność zyskują modele sieciowe o prostym grafie i dokładnym opisie funkcjonalnym użytych fragmentów. W pracy [Nol03] autor proponuje, aby do każdego elementu biblioteki określać jego rozkład losowy czasu wykonania, który potem będzie użyty do stochastycznego szeregowania procesów (stochastic shedulability analysis).

Można zaobserwować, że rozkłady losowe czasu realizacji fragmentów programu czy też funkcji systemu operacyjnego nie mają regularnego kształtu i nie przypominają rozkładu wykładniczego ani normalnego. Proponowane są funkcje:

- Erlanga [Abd03]
- Weibula [Abd04]
- Gumbla (logWeibul) [Pet02]

mimo iż oznacza to znaczącą komplikację obliczeń.

Dla modelu BSP stanowiącego sekwencję wielu (wewnętrznie równoległych) kroków proponowany jest rozkład normalny [Xu02]. Czas realizacji programu jest prostą sumą zmiennych losowych (czasów każdego bloku), więc w tym wyjątkowym przypadku autorzy mogą powołać się na centralne twierdzenie graniczne.

Poważnym problemem w sieciowych modelach programu jest określenie prawdopodobieństw wyboru jednej z dróg w grafie [Lin99]. Większość autorów pomija ten aspekt, koncentrując się na dalszych etapach analizy. Oryginalne podejście do problemu prawdopodobieństwa wyboru ścieżki i wzajemnej korelacji kolejnych instrukcji warunkowych zaproponowano w pracy [Dav04]. Przyjęto tam, że każdy warunek da się wyrazić jako funkcję danych wejściowych. Znając rozkład losowy parametrów programu, można zatem ustalić nie tylko prawdopodobieństwo wyboru danej drogi, ale również prawdopodobieństwa (i rozkłady) warunkowe. Pomysłu tego nie zastosowano w pracy. Przyjęto, że w typowych zastosowaniach obliczeniowych ilość powtórzeń pętli jest stała (zależna od rozmiaru zagadnienia) lub uwarunkowana zbyt wieloma danymi wejściowymi.

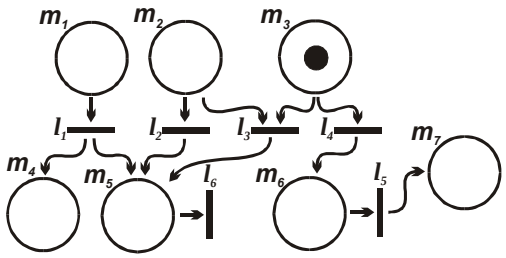
1. Wykorzystanie sieci Petriego

1.1. Klasyczna sieć Petriego i modelowanie upływu czasu

Sieć Petriego jest grafem składającym się z dwóch rodzajów węzłów – miejsc i przejść oraz łączących je łuków (krawędzi). Stan sieci określają znaczniki (żetony) znajdujące się w miejscach i przemieszczające się na skutek wykonywania (odpalania) przejść. Sieć Petriego stanowi wygodne narzędzie do modelowania powiązanych zjawisk współbieżnych i jest często wykorzystywana w pracach dotyczących analizy działania maszyn cyfrowych i ich oprogramowania.

Def. 1-1

Podstawowa sieć Petriego jest grafem skierowanym opisywanym czwórką uporządkowaną <M, L, K, U_0> gdzie $M=\{m_1, m_2, ...\}$ jest zbiorem miejsc, $L=\{l_1, l_2, ...\}$ zbiorem przejść, $K=\{k_1, k_2, ...\}$ zbiorem krawędzi, a U_0 to stan początkowy sieci. Zbiory miejsc M i przejść L grafu są rozłączne $M \cap L = \varnothing$. Krawędzie są skierowane i mogą jedynie łączyć miejsce z przejściem lub przejście z miejscem - zatem $K \subset (M \times L) \cup (L \times M)$.



Rys. 1-1. Przykład sieci Petriego

Aby wykorzystać sieć Petriego do analizy szybkości wykonywania programu należy rozszerzyć model o pojęcie upływu czasu, definiując jego wpływ na działanie sieci. Znane są następujące rozszerzenia:

- przypisanie każdemu z przejść intensywności z jaką jest ono odpalane,
- przypisanie intensywności tylko niektórym przejściom, wraz z założeniem, że pozostałe odpalane są natychmiast, gdy tylko zaistnieją wymagane warunki,
- przypisanie każdemu z przejść stałego czasu koniecznego do jego odpalenia,
- przypisanie każdemu z przejść rozkładu losowego czasu koniecznego do jego odpalenia,
- przypisanie każdemu z miejsc czasu jaki znacznik musi w nim przebywać zanim zostanie wykorzystany do odpalenia kolejnego przejścia.

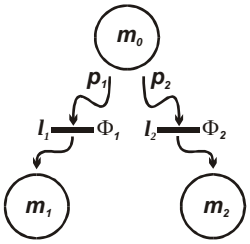
W pracy wybrano model zakładający przypisanie każdemu z przejść rozkładu losowego czasu jego wykonania. Przypisanie każdemu z przejść rozkładu losowego a nie konkretnej wartości czasu, jest rozwiązaniem elastycznym, umożliwiającym uwzględnienie niedeterministycznego zachowania się wykonywanego programu.

Def. 1-2.

Sieć Petriego ze stochastycznym opisem przejść jest siecią Petriego, w której każdemu z przejść przypisano rozkład losowy czasu, po jakim zostanie odpalone. Jest ona grafem opisanym piątką uporządkowaną <M, L, K, U_0, J> zachowującą wszystkie cechy podstawowej sieci Petriego <M, L, K, U_0> wzbogacając ją o funkcję J przypisującą każdemu z przejść ze zbioru L rozkład prawdopodobieństwa jego odpalenia jako funkcję czasu, jaki upłynął od zaistnienia warunków do odpalenia przejścia.

W szczególnym przypadku, gdy wszystkie rozkłady czasów odpaleń są punktowe uzyskuje się sieć Petriego o deterministycznym czasie odpalania przejść stosowaną często do modelowania działania maszyn cyfrowych [Mag89].

W sieci Petriego może zachodzić zjawisko konkurowania przejść o znacznik. Odpalenie jednego z przejść powoduje pobranie znacznika z miejsca i niemożność odpalenia drugiego z przejść. Konieczne jest, zatem zdefiniowanie kryterium wyboru jednej z dróg. Najprostszym sposobem jest określenie prawdopodobieństw p_i odpaleń poszczególnych przejść oraz dystrybuant Φ_i określających rozkład losowy czasu odpalenia każdego z przejść o ile zostanie ono wybrane. Wówczas prawdopodobieństwo odpalenia przejścia l_i w ciągu czasu t określa funkcja $p_i\Phi_i(t)$.



Rys. 1-2. Wybór jednej z dróg w grafie

Zastosowane w pracy przypisanie znanych z góry prawdopodobieństw odpaleń konkurujących przejść jest jednym z możliwych sposobów rozwiązania kwestii wyboru drogi w grafie. Druga, równoważna z metod zakłada dynamiczną „walkę" przejść o znacznik. Zdecydowana większość autorów wykorzystuje model ze znanymi prawdopodobieństwami wyboru dróg (ze względu na uproszczenie obliczeń i analizy). W sieciach stochastycznych, gdzie konkurują przejścia o rozkładach wykładniczych, prawdopodobieństwa wyborów dróg są proporcjonalne do intensywności, zatem również można przyjąć, że są znane.

1.2. Synchronizacja procesów w sieci Petriego

W maszynie równoległej szczególnego znaczenia nabiera synchronizacja poszczególnych operacji. Ze względu na współdziałanie wielu jednostek przetwarzających nie wystarczy jedynie ułożenie sekwencji działań (programu) dla każdej z nich osobno. Konieczne jest powiązanie kolejności wykonywania operacji przez wszystkie procesy. Każdy model maszyny równoległej musi, zatem zapewniać mechanizmy jawnej synchronizacji procesów. Może być ona ściśle związana z wymianą danych – synchroniczne i asynchroniczne przekazywanie komunikatów. W modelach pamięci dzielonej i BSP do synchronizacji służą osobne instrukcje.

W niektórych programach używana bywa synchronizacja ukryta, wykorzystującą wiedzę o szybkości procesorów, albo o synchronizacji innych fragmentów kodu. Jest ona w dużej mierze nieprzewidywalna i często

zależna od konkretnej konfiguracji sprzętu i systemu operacyjnego. Mechanizmy niejawnej synchronizacji nie będą rozpatrywane w pracy.

Synchronizacja SYNC(r_{Ai}, r_{Bj}, ...) jest relacją pomiędzy instrukcjami r_{Ai}, r_{Bj}, ... różnych procesów dopuszczającą jedynie niektóre zależności czasów realizacji poszczególnych instrukcji. Można je sklasyfikować ze względu na przyczynę:

- w synchronizacji pozytywnej warunkiem kontynuacji jest osiągnięcie przez inne procesy zadanego stanu w dowolnej poprzedzającej chwili,
- w synchronizacji negatywnej warunkiem kontynuacji jest pozostawanie zadanych procesów w danej chwili w stanie nie zawierającym się w zbiorze stanów zakazanych (sekcji krytycznej),

oraz symetrię:

- w synchronizacji symetrycznej wzajemna relacja procesów uczestniczących w synchronizacji jest symetryczna (przemienna). SYNC(r_A, r_B) $\Leftrightarrow$ SYNC(r_B, r_A)
- w synchronizacji asymetrycznej jeden z procesów jest uprzywilejowany i możliwość jego dalszego wykonywania nie zależy od stanu drugiego. SYNC(r_A, r_B) $<\neq>$ SYNC(r_B, r_A)

W relacji synchronizacji może uczestniczyć dwa lub więcej procesów. Synchronizacja dwóch procesów jest szczególnym przypadkiem synchronizacji N procesów.

Relacja wzajemnej synchronizacji może dotyczyć pojedynczych instrukcji (w synchronizacji pozytywnej) jak również zbiorów instrukcji (w negatywnej). Możliwe jest również synchronizowanie instrukcji z obszarem (w negatywnej asymetrycznej). Zakłada się, że instrukcje synchronizacji są niepodzielne.

Synchronizacja pozytywna asymetryczna

Synchronizacja pozytywna charakteryzuje się oczekiwaniem procesu na osiągnięcie przez inny proces zadanego stanu, czyli wykonanie określonej instrukcji. Asymetria synchronizacji oznacza, że konieczność oczekiwania dotyczy jedynie procesu podrzędnego P_B, natomiast proces nadrzędny P_A może być wykonywany niezależnie od stanu procesu podrzędnego.

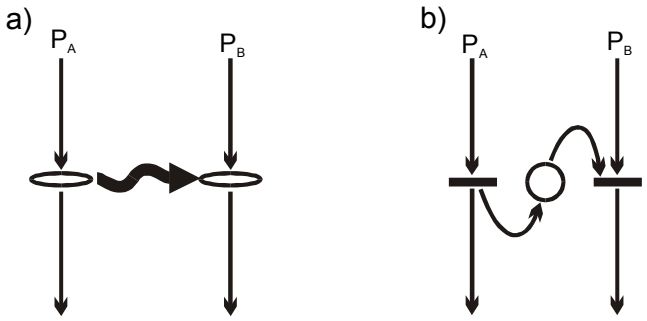
Pojęcie kolejności wykonania bywa różnie interpretowane w różnych modelach. Jeżeli procesowi synchronizacji towarzyszy wymiana danych, wymaga się, aby instrukcja odbioru rozpoczęła swoje właściwe funkcjonowanie po zakończeniu instrukcji nadawania. Należy wziąć pod uwagę, że instrukcja odbioru może dokonać czynności przygotowawczych przed zakończeniem nadawania oraz instrukcja nadawcza może dokonać czynności końcowych po nadaniu komunikatu. Najbardziej elastyczny i łatwy do zrealizowania jest, zatem wymóg, aby wykonanie instrukcji odbioru nastąpiło nie wcześniej niż wykonanie instrukcji nadawania z nią zsynchronizowanej.

Def. 1-3

Instrukcje r_{Ai} oraz r_{Bj} należące do różnych procesów P_A i P_B są w relacji synchronizacji pozytywnej asymetrycznej, jeżeli zagwarantowane jest, że wykonanie instrukcji r_{Bj} nie zakończy się wcześniej niż wykonanie instrukcji r_{Ai}.

$$t_A \leq t_B \tag{1-1}$$

Graficzną reprezentację synchronizacji pozytywnej asymetrycznej i odpowiadający jej fragment sieci Petriego przedstawia rys. 1-3. Miejsce sieci Petriego umożliwia buforowanie komunikatu i kontynuację procesu nadającego A, pomimo niegotowości procesu odbierającego B.



Rys. 1-3. Synchronizacja pozytywna asymetryczna i odpowiadająca sieć Petriego

Synchronizacja pozytywna asymetryczna występuje w modelach, które bazują na asynchronicznym (buforowanym) przesyłaniu komunikatów. Proces udostępniający dane nie musi czekać na gotowość odbiorcy o ile istnieje bufor umożliwiający przechowanie danych.

Synchronizacja pozytywna symetryczna

Synchronizacja pozytywna charakteryzuje się oczekiwaniem procesu na osiągnięcie przez inny proces zadanego stanu, czyli wykonanie określonej instrukcji. Symetria synchronizacji oznacza, że każdy z procesów musi poczekać na pozostałe uczestniczące w synchronizacji.

Pojęcie jednoczesnego wykonania bywa różnie interpretowane w zależności od poziomu, na jakim rozważane jest wykonanie instrukcji oraz stopnia synchronizacji zapewnianego przez sprzęt. Najbardziej elastyczny i łatwy do zrealizowania jest wymóg, aby zakończenie instrukcji synchronizacji nastąpiło nie wcześniej niż zsynchronizowanej instrukcji drugiego procesu.

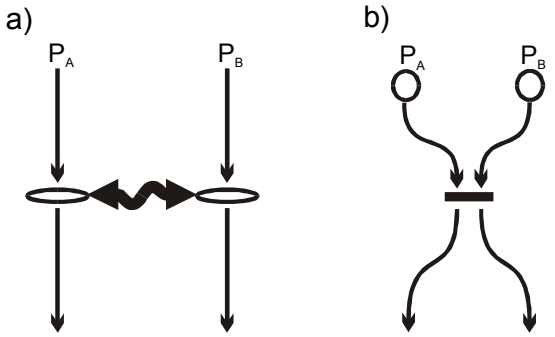
Def. 1-4

Instrukcje r_{Ai} oraz r_{Bj} należące do różnych procesów są w relacji synchronizacji pozytywnej symetrycznej, jeżeli zagwarantowane jest ich jednoczesne wykonanie.

$$t_A = t_B \tag{1-2}$$

Synchronizacja pozytywna symetryczna może być rozpatrywana jako złożenie dwóch synchronizacji asymetrycznych. Pierwsza z nich wymusza oczekiwanie procesu P_A na proces P_B natomiast druga procesu P_B na proces P_A.

Ze względu na konieczność jednoczesnego wykonania instrukcji synchronizacyjnych przez oba procesy synchronizację pozytywną symetryczną nazywa się czasem spotkaniem. Jej schemat graficzny oraz odpowiadający mu fragment sieci Petriego przedstawia rys. 1-4.



Rys. 1-4. Synchronizacja pozytywna symetryczna i odpowiadająca sieć Petriego

Opis w postaci sieci Petriego jest w tym przypadku szczególnie prosty i sprowadza się do pojedynczego przejścia o dwóch krawędziach wejściowych i dwóch wyjściowych. Do odpalenia przejścia jest konieczna obecność znaczników w obu gałęziach wejściowych (gotowość), a w wyniku jego odpalenia oba znaczniki jednocześnie przekazywane są do miejsc wyjściowych. Synchronizacja pozytywna symetryczna jest stosowana w modelach zakładających synchroniczne przesyłanie komunikatów, gdzie wymagana jest gotowość obu uczestniczących procesów. Mimo przepływu danych w jedną stronę synchronizacja jest symetryczna. Od odbiorcy do nadawcy przesyłana jest niejawnie informacja „jestem gotowy".

Bariera jest również przykładem synchronizacji pozytywnej symetrycznej. Znajduje ona zastosowanie w modelu BSP a także niektórych modelach pamięci dzielonej. Jej stosunkowo mała popularność wynika z trudności realizacji w systemach o dużej szybkości i słabym powiązaniu procesorów.

Synchronizacja negatywna asymetryczna (priorytet)

Synchronizacja negatywna charakteryzuje się wstrzymaniem procesu przed wejściem do obszaru synchronizowanego, jeżeli drugi z procesów wykonuje obszar kodu objęty synchronizacją. Asymetria synchronizacji oznacza, że konieczność oczekiwania dotyczy jedynie procesu podrzędnego P_B natomiast proces nadrzędny P_A może być wykonywany niezależnie od stanu procesu podrzędnego.

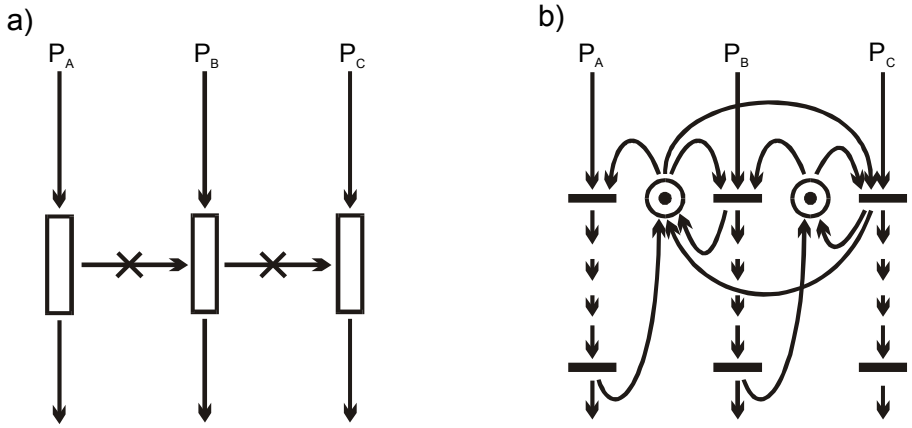
Def. 1-5

Zbiory instrukcji R_A, R_B, R_C, ... należące do różnych procesów są w relacji synchronizacji negatywnej asymetrycznej jeżeli nie jest możliwe rozpoczęcie realizacji instrukcji ze zbioru R_i procesu o niższym priorytecie w przypadku gdy w danym momencie są realizowane instrukcje zbioru R_j któregoś z procesów o wyższym priorytecie.

Jeśli oznaczymy czas wykonywania zbioru instrukcji R_A jako przedział $[t_{A1}, t_{A2}]$ zbioru instrukcji R_B jako $[t_{B1}, t_{B2}]$, a zbioru instrukcji R_C jako $[t_{C1}, t_{C2}]$, to dla każdej realizacji programu konieczne zagwarantowanie spełnienia zależności:

$$[t_{A1}, t_{A2}] \cap [t_{B1}, t_{B1}] = \varnothing \qquad (1\text{-}3)$$
$$[t_{A1}, t_{A2}] \cap [t_{C1}, t_{C1}] = \varnothing$$
$$[t_{B1}, t_{B2}] \cap [t_{C1}, t_{C1}] = \varnothing$$

Proces A może wejść do sekcji krytycznej niezależnie od tego czy proces B jest w sekcji czy też nie, natomiast proces B musi poczekać aż proces A opuści sekcję. Synchronizację negatywną asymetryczną dla dwóch procesów można zastąpić synchronizacją negatywną symetryczną, w której długość sekcji krytycznej procesu B jest zredukowana do zera (ale pozostaje warunek wejścia). Synchronizacja negatywna asymetryczna nakłada ograniczenia nie tyle na możliwość jednoczesnego przebywania w sekcji krytycznej, co na warunek wejścia dla procesu o niższym priorytecie. Reprezentację graficzną oraz sieć Petriego dla synchronizacji negatywnej asymetrycznej przedstawia rys. 1-5.



Rys. 1-5. Synchronizacja negatywna asymetryczna i odpowiadająca sieć Petriego

Synchronizacja negatywna może być przedstawiona za pomocą sieci z łukami wzbraniającymi. Reprezentuje to w sposób bardziej naturalny synchronizację negatywną, jednak jeżeli sekcja krytyczna zawiera więcej przejść, to konieczne jest poprowadzenie większej ilości łuków wzbraniających. W pracy zdecydowano się zastosować sieć Petriego bez łuków wzbraniających jako model prostszy, o mniejszej mocy opisowej.

W wielu programach współbieżnych stosuje się dostęp do niepodzielnych zasobów na zasadzie priorytetu i czasowego wywłaszczania. Rzadko jednak modele współbieżności zawierają taki sposób synchronizacji. Występuje on często w sposób niejawny w oprogramowaniu systemowym lub jest tworzony z wykorzystaniem innych mechanizmów synchronizacji.

Synchronizacja negatywna symetryczna

Synchronizacja negatywna charakteryzuje się wstrzymaniem procesu przed wejściem do obszaru synchronizowanego, jeżeli drugi z procesów wykonuje obszar kodu objęty synchronizacją. Symetria synchronizacji oznacza, że konieczność oczekiwania dotyczy w jednakowym stopniu obu procesów.

Def. 1-6

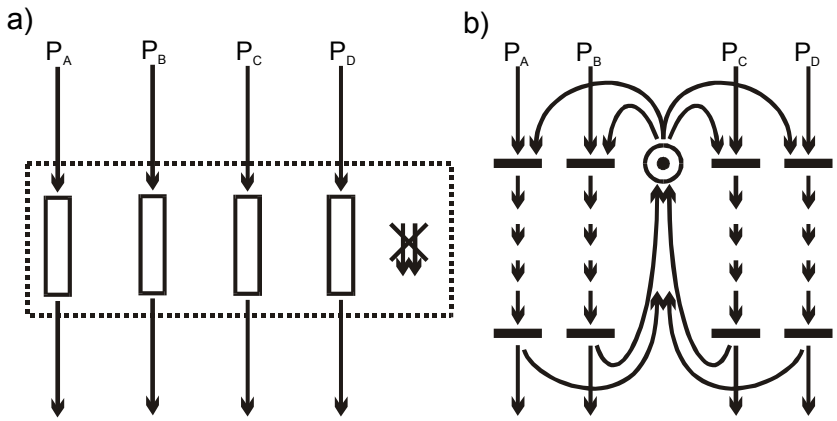
Zbiory instrukcji R_A oraz R_B należące do różnych procesów są w relacji synchronizacji negatywnej symetrycznej jeżeli nie jest możliwe jednoczesne wykonanie instrukcji ze zbioru R_A i R_B.

Jeśli oznaczymy czas wykonywania zbioru instrukcji R_A jako przedział $[t_{A1}, t_{A2}]$ a zbioru instrukcji R_B jako $[t_{B1}, t_{B2}]$, to konieczne jest spełnienie zależności:

$$[t_{A1}, t_{A2}] \cap [t_{B1}, t_{B2}] = \varnothing \qquad (1\text{-}4)$$

Przedział czasowy realizacji sekcji krytycznej przez procesy P_A i P_B nie może się (nawet częściowo) pokrywać. Jest to równoznaczne z wykluczeniem możliwości rozpoczęcia realizacji R_B jeżeli jest aktualnie realizowany R_A i vice versa.

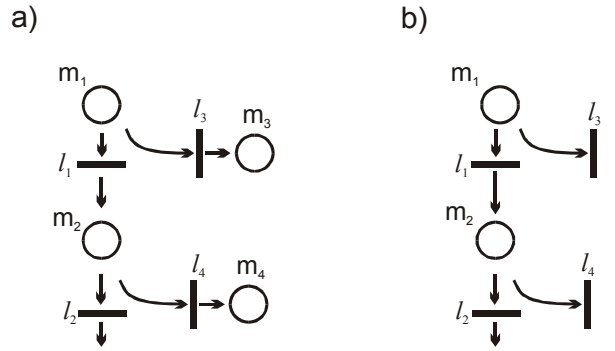
Aby zapewnić realizację synchronizacji negatywnej umieszcza się specjalne instrukcje na początku i końcu obszaru kodu podlegającego synchronizacji. Warunkiem koniecznym wejścia w obszar synchronizowany jest „zagarnięcie" niepodzielnego wspólnego znacznika (semafora). Znacznik ten jest oddawany przez instrukcję wyjścia, umożliwiając innym procesom wejście. Do momentu oddania znacznika przez proces wykonujący kod podlegający synchronizacji żaden inny proces nie może do zsynchronizowanego obszaru kodu. Schemat graficzny oraz uproszczony fragment sieci Petriego dla tego typu synchronizacji przedstawia rys. 1-6.



Rys. 1-6. Synchronizacja negatywna symetryczna i odpowiadająca sieć Petriego

1.3. Modelowanie uszkodzeń w sieci Petriego

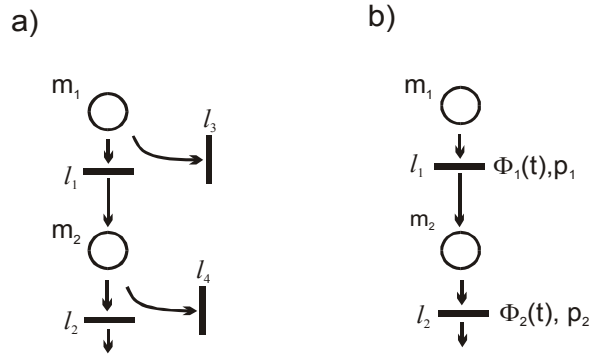
Najprostszym sposobem uwzględnienia niezawodności w sieci Petriego jest dodanie przejść modelujących uszkodzenia. Umożliwiają one pobranie znacznika, a przez to zablokowanie wykonania poprawnych przejść oraz umieszczenie znacznika w miejscu symbolizującym awarię.



Rys. 1-7. Dodanie przejść l_3 i l_4 opisujących uszkodzenia do sieci Petriego
a) przejścia obrazujące uszkodzenie oraz miejsca oznaczające stan niesprawny
b) przejścia ślepe pochłaniające znacznik

W przypadku, gdy wynikiem uszkodzenia jest zatrzymanie pracy modelowanego układu można zredukować przejścia modelujące awarię do przejść ślepych pochłaniających znacznik, a tym samym uniemożliwiających kontynuację pracy. Upraszcza to strukturę grafu, a co za tym idzie jego analizę.

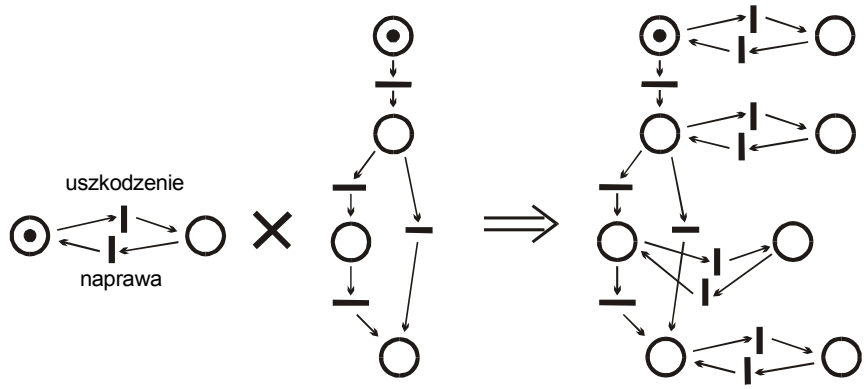
Przypisując do przejść grafu prawdopodobieństwa wyboru danej drogi można podczas analizy grafu nie rozpatrywać przejść ślepych, a jedynie prawdopodobieństwo i rozkład czasu poprawnej realizacji grafu.



Rys. 1-8. Przejścia ślepe l_3 i l_4 pochłaniające znacznik
a) przejścia ślepe pochłaniające znacznik
b) równoważny graf zawierający prawdopodobieństwa wyboru drogi

Możliwości sieci Petriego nie ograniczają się jedynie do modelowania prostych uszkodzeń nienaprawialnych. Możliwe jest opisanie za pomocą sieci złożonego procesu stochastycznego uszkodzeń i napraw. Opis samego procesu uszkadzania i napraw można przedstawić za pomocą prostej sieci, której liczba miejsc jest równa liczbie stanów procesu. Przejścia o jednej krawędzi wejściowej i jednej wyjściowej opisują możliwości zmian stanów sprawności.

Tworząc graf uwzględniający jednocześnie zmiany stanów sprawności maszyny jak też zmiany stanu realizowanego programu łączy się ze sobą model niezawodnościowy i model realizacji programu. Wychodząc z założenia, że na każdym etapie realizacji programu procesor może znajdować się w jednym z możliwych stanów sprawności, wypadkowy graf ma tyle stanów (miejsc) ile wynosi iloczyn liczby stanów (miejsc) grafów składowych. Konstrukcja grafu łącznego polega na zastąpieniu każdego miejsca sieci opisującej realizację programu poprzez graf opisujący proces uszkadzania i napraw. Przykładową procedurę połączenia grafów przedstawiono na rys. 1-9. Warto zwrócić uwagę na brak przejść pionowych symbolizujących postęp obliczeń pomiędzy miejscami opisującymi stan niesprawny. Z założenia stan niesprawny to stan uniemożliwiający kontynuację pracy, czyli przejście do kolejnego etapu programu. Konieczne jest najpierw dokonanie naprawy, a dopiero potem wykonanie kolejnego kroku obliczeń.



Rys. 1-9. Złożenie procesu uszkodzeń i napraw z procesem wykonywania programu

2. Graf Przepływu Sterowania

2.1. Model programu

Celem metody opisywanej w pracy jest określenie czasu wykonywania programu równoległego wykorzystującego jeden z powszechnie przyjętych modeli współbieżności. Ponieważ pojęcie programu jest bardzo szerokie – może on być wyrażony w postaci kodu źródłowego, kodu pośredniego, kodu wykonywalnego w pliku, kodu wykonywalnego w pamięci – konieczne jest przyjęcie jednoznacznej definicji.

Def. 2-1

Programem PGM=<In, Nast, Rs, In_{start}> nazywany będzie zbiór instrukcji In powiązanych w logiczną całość regułami Nast bezwarunkowego i warunkowego następstwa instrukcji, oraz powiązań synchronizacyjnych Rs spełniający następujące warunki:

- *dla każdej instrukcji określony jest zbiór jej następników,*
- *znane jest prawdopodobieństwo wyboru określonego następnika,*
- *jest znana liczba procesów jaka będzie uruchomiona aby wykonać program,*
- *początkowe instrukcje procesów określa zbiór In_{start} ,*
- *występuje jedynie jawna synchronizacja procesów,*
- *wszystkie instrukcje synchronizacji wykonywane są bezwarunkowo,*
- *występują jedynie mechanizmy synchronizacji pozytywnej i negatywnej,*
- *powiązania Rs pomiędzy instrukcjami synchronizacji różnych procesów są statycznie zdefiniowane i możliwe do określenia na podstawie kodu programu,*
- *program PGM składa się programów PGM_i dla poszczególnych procesów.*

Def. 2-2

Procesem (jednowątkowym) P_i=<In_i, $Nast_i$, Rs, in_{start}> nazywany będzie fragment stanu maszyny składający się z fragmentu programu PGM_i=<In_i, $Nast_i$, Rs, in_{start}> umieszczonego w pamięci wykonywany przez dedykowany procesor. W szczególności proces:

- *wykonywany jest sekwencyjnie, bez możliwości kreowania procesów potomnych,*
- *posiada swój program PGM_i o zbiorach instrukcji In_i i następstw $Nast_i$,*
- *rozpoczyna się od instrukcji in_{start}.*

Z powyższych definicji wynika, że dla każdej poprawnej realizacji programu prawdopodobieństwo wykonania każdej zawartej w nim instrukcji synchronizacji jest równe jeden. W przypadku instrukcji synchronizacji zawartych w pętli wymaga się, aby liczba wykonań mogła być określona statycznie.

Dla każdej instrukcji synchronizacji musi być możliwe znalezienie instrukcji z nią współpracującej na podstawie struktury programu, bez konieczności jego wykonania.

Przedstawione założenia ograniczają zbiór programów poddawanych analizie do programów napisanych zgodnie z wymienionymi regułami. Większość prac dotyczących badania realizacji programu nakłada podobne ograniczenia [Jar87] [Deg88] [Bra86], gdyż są one konieczne, aby możliwe było skonstruowanie grafu

obrazującego działanie programu. Większość aplikacji dokonujących przekształceń matematycznych posiada regularną strukturę programu i spełnia powyższe warunki. Liczba instrukcji synchronizacji w głównej pętli oraz ich powiązanie jest statycznie zdefiniowane. Liczba obiegów pętli jest w większości przypadków funkcją rozmiaru przetwarzanego zagadnienia i nie zależy od wartości przetwarzanych danych. W przypadku algorytmów o nieznanej liczbie powtórzeń głównej pętli (np. typu branch & bound) analizie można poddać wnętrze pętli, a następnie skonstruować rekurencyjny wzór określający czas realizacji w zależności od liczby powtórzeń.

2.2. Model programu opisany siecią Petriego

Podstawowy model sieci Petriego nie umożliwia reprezentacji rozkładu losowego czasu koniecznego do wykonania poszczególnych fragmentów programu. W tym celu zastosowano rozszerzony model sieci zwanej dalej grafem przepływu sterowania.

Def. 2-3 Graf przepływu sterowania – (patrz def. 1.2)
Grafem przepływu sterowania GPS=<M, L, K, U_0, J> nazywana będzie sieć Petriego wraz z funkcją $J(l_i)\rightarrow\Phi_i(t)$, przypisującą każdemu z przejść rozkład losowy czasu wykonania instrukcji. Sieć Petriego <M, L, K, U_0> stowarzyszona z GPS ma strukturę odpowiadająca zależnościom następstw instrukcji analizowanego programu.

Każde z miejsc tworzących zbiór M określa stan jednego z procesów, natomiast przejścia tworzące zbiór L opisują możliwe zmiany stanu. W przypadku warunkowego następstwa instrukcji znacznik może wydostać się z określonego miejsca przez kilka alternatywnych przejść. Zbiór K tworzą krawędzie łączące przejścia i miejsca grafu. Stan początkowy sieci U_0 określa, w jakich miejscach sieci GPS znajdują się znaczniki stanu odpowiadającego uruchomieniu analizowanego programu. Na początku znaczniki są w miejscach symbolizujących stan początkowy każdego z uczestniczących procesów oraz w miejscach przedstawiających semafory (o ile występują).

Ostatni element piątki <M, L, K, U_0, J> to funkcja przyporządkowująca każdemu z przejść sieci rozkład losowy czasu wykonania instrukcji. Autor dopuszcza dowolny rozkład losowy lub wartość stałą. Zakłada się brak wpływu historii odpalania poprzednich przejść na rozkład losowy czasu realizacji danego przejścia. Analizowany GPS jest więc grafem procesu stochastycznego semi-Markova.

Model GPS opisuje jedynie strukturę programu, nie uwzględniając wartości przetwarzanych danych. Wykonywanie instrukcji warunkowych w zależności od wartości zmiennych i wynikające z tego różnice czasu realizacji poszczególnych procedur zostają uwzględnione w kształcie rozkładu losowego. Przyjmuje się, że wybór jednego z możliwych do odpalenia przejść w grafie odbywa się w sposób losowy.

Zgodnie z przyjętym modelem, wartość oczekiwana czasu realizacji programu (lub fragmentu) będzie równa średniej ważonej czasów t_x dla każdej drogi w grafie prowadzącej od instrukcji startowej do końcowej. Prawdopodobieństwo wyboru danej drogi p_x stanowi jej wagę.

$$E(t) = \sum p_x t_x \tag{2-1}$$

W dalszej części pracy do opisu rozkładu ilości instrukcji wykonywanych przy realizacji fragmentu opisywanego przez pojedyncze przejście sieci Petriego stosowany będzie iloczyn prawdopodobieństwa p_i wyboru danego przejścia i dystrybuanty $\Phi_i(t)$ rozkładu czasu jego realizacji.

W najprostszym przypadku istnieje tylko jedna droga (sekwencja) prowadząca od instrukcji początkowej fragmentu do końcowej. Liczba koniecznych do wykonania instrukcji jest wtedy stała i równa t_c. Funkcja dystrybuanty $\Phi(t)$ przyjmuje postać:

$$\forall\ t_i : t_i > t_c \rightarrow \Phi(t_i) = 1 \tag{2-2}$$

$$\forall\ t_i : !(t_i > t_c) \rightarrow \Phi(t_i) = 0 \tag{2-3}$$

W razie możliwości uszkodzenia oprogramowania zatrzymującego (zapętlającego) proces wartość sumy iloczynów prawdopodobieństw i dystrybuant przejść wychodzących z danego miejsca m_n jest mniejsza od jedności dla nieskończonej wartości czasu. Oznacza to że program z pewnym prawdopodobieństwem nigdy nie opuści danego stanu i nie zostanie poprawnie zakończony.

$$p[\text{uszkodzenie w } m_n] = 1 - \sum_{l_i \in L:(m_n, l_i) \in K} p_i \Phi_i(+\infty) \tag{2-4}$$

W warunkach ograniczeń wynikających z przyjętego modelu możliwe jest:

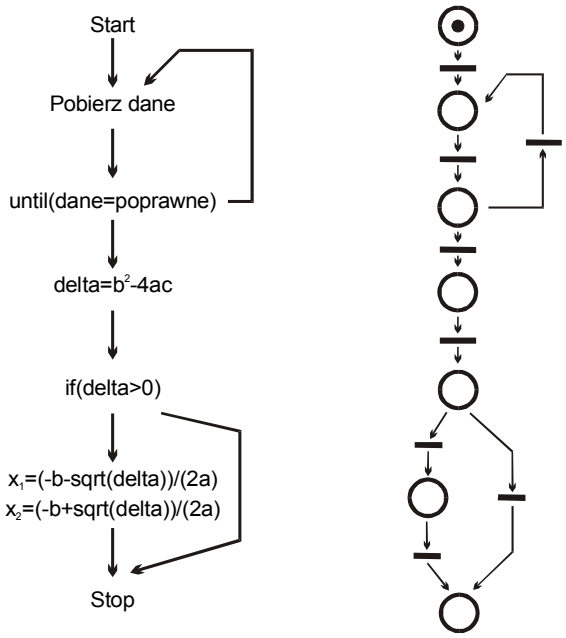
- reprezentowanie niektórych uszkodzeń oprogramowania
- reprezentowanie stałej i losowej (zależnej od wartości danych) ilości wykonanych instrukcji
- probabilistyczny opis wyboru ścieżki w grafie

2.3. Konstruowanie GPS

Graf przepływu sterowania tworzony jest na podstawie programu poprzez wykonanie następujących czynności:

- dla każdej instrukcji $in_i \in In$ utworzenie miejsca $m_i \in M$ w grafie GPS,
- dla każdej relacji następstwa $nast_{ij} \in Nast$ utworzenie przejścia $l_{ij} \in L$ wraz z pojedynczą krawędzią przychodzącą z miejsca m_i i wychodzącą do miejsca m_j,
- dla każdej relacji synchronizacji utworzenie fragmentu sieci odwzorowującego jej działanie (według opisu przedstawionego w rozdziale 1),
- dla każdej instrukcji $in_x \in I_{start}$ przypisanie obecności znacznika w miejscu m_x w znakowaniu początkowym U_0,
- przypisanie każdemu przejściu $l_{ij} \in L$ dystrybuanty czasu realizacji instrukcji opisanej funkcją $J(l_{ij})$.

Ze względów praktycznych konieczne jest dokonywanie wstępnej redukcji rozmiaru grafu już na etapie tworzenia GPS. Jeżeli instrukcja posiada tylko jeden (bezwarunkowy) następnik, to obie te instrukcje można przedstawić w postaci pojedynczego przejścia o sumarycznej złożoności. Wstępne grupowanie sekwencyjnych instrukcji umożliwia uzyskanie sieci Petriego, której rozmiar pozwala na analizę za pomocą dostępnych narzędzi.



Rys. 2-1. Prosty program i odpowiadająca mu sieć Petriego

Instrukcje synchronizacji zawarte w programie są reprezentowane w GPS w postaci połączeń pomiędzy podgrafami opisującymi poszczególne procesy. W trakcie konstruowania GPS interakcje pomiędzy procesami zostają przedstawione w postaci odpowiedniego dla nich fragmentu sieci Petriego prezentowanego w rozdziale pierwszym. Mimo, iż na poziomie kodu maszynowego instrukcje synchronizacji mogą być podobne do innych instrukcji i procedur, to najczęściej są one składnikami konstrukcji języka wyższego poziomu, w jakim napisano program lub składnikami podstawowej biblioteki. Wykrycie ich w treści programu (nawet skompilowanego i zoptymalizowanego) nie powinno stanowić trudności. W procesie automatycznej analizy programu trudniejszym aspektem jest połączenie instrukcji synchronizacji poszczególnych procesów. Jeżeli program zgodnie z wymaganiami def. 2-1 posiada jedynie takie instrukcje synchronizacji, które w sposób jednoznaczny dają się rozwikłać poprzez statyczną analizę kodu programu to również ten etap konstruowania GPS jest możliwy do automatycznego wykonania.

Po określeniu sieci Petriego opisującej strukturę GPS oraz określeniu liczby instrukcji wchodzących w skład poszczególnych kroków pozostaje określenie prawdopodobieństwa wyboru poszczególnych dróg w grafie. Na podstawie treści programu trudno jest określić szanse poszczególnych instrukcji warunkowych. Jest to możliwe jedynie w przypadku pętli o stałej liczbie powtórzeń. Czasami prawdopodobieństwo prawdziwości warunku można oszacować na podstawie analizy algorytmu, jednak trudno tego dokonać automatycznie. W niektórych przypadkach zastosowanie reguł 50/50 lub 90/10 daje dobre przybliżenie zwłaszcza, gdy kompilator w celach optymalizacji kodu próbuje określać prawdopodobieństwo skoku na podstawie treści programu w języku wyższego poziomu (np. instrukcje typu assert() mają znikome prawdopodobieństwo realizacji).

Dla części instrukcji warunkowych określenia prawdopodobieństw wyborów poszczególnych dróg można dokonać jedynie w sposób doświadczalny. Wykorzystując programy typu profiler można uzyskać statystyczne dane na temat ilości wykonań poszczególnych funkcji, prawdziwości warunków i podjętych w związku z tym działań programu. W przypadku programów przetwarzających dane wybór drogi jest w bardzo małym stopniu zależny od szybkości sprzętu oraz rozkładu zdarzeń zewnętrznych. Dane statystyczne na temat prawdopodo-

bieństw wyboru poszczególnych ścieżek można, zatem zebrać na podstawie wykonywania programu tylko na jednej maszynie rzeczywistej lub emulowanej. Będą one prawdziwe również dla wykonania tego samego programu na innej maszynie. Zdecydowanie trudniej jest statystycznie określić wybór drogi dla programów działających w czasie rzeczywistym, które w intensywny sposób współpracują z urządzeniami zewnętrznymi. Prawdziwość warunku zależy często od wzajemnej relacji szybkości programu oraz jego otoczenia.

W literaturze często zakłada się, że prawdopodobieństwo wyboru danej gałęzi jest znane, nie wnikając w metody jego określenia.

Problemy występujące przy konstrukcji GPS

W językach programowania zakłada się sekwencyjne wykonywanie instrukcji wymienionych w treści programu. Istnienie warunków i pętli jest jawnie zadeklarowanych przez użycie instrukcji warunkowych oraz iteracyjnych. Dopuszczalne są jednak konstrukcje uniemożliwiające statyczne określenie zbioru następników. W języku C/C++ problem mogą stanowić:

- wywołania funkcji na podstawie wartości wskaźnika do funkcji,
- wywołania metod na podstawie selektora metody,
- wywołania funkcji virtualnych.

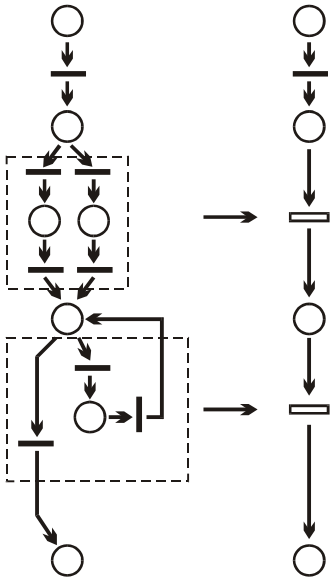
Przyjęta struktura GPS wymaga ścisłego powiązania ze sobą instrukcji synchronizacji należących do współdziałających procesów. Część języków programowania umożliwia użycie komunikacji i synchronizacji w sposób nie pozwalający na jednoznaczne określenie współdziałających procesów ani utworzenie zbioru instrukcji różnych procesów wchodzących w skład danej synchronizacji. Szczególne trudności mogą stwarzać:

- nadawanie komunikatu do wszystkich (ang. broadcast),
- nadawanie komunikatu do pierwszego, który odbierze,
- dynamiczne określanie adresata,
- komunikaty bez wyróżnika (ang. tag) interpretowane wg kolejności nadchodzenia,
- bariery bez wyróżnika interpretowane według kolejności.

2.4. Redukcja grafu przepływu sterowania

Otrzymany w wyniku analizy programu graf przepływu sterowania poddaje się redukcji mającej na celu zmniejszenie jego rozmiarów i likwidację pętli nie wynikających z synchronizacji procesów.

Celem redukcji grafu przepływu sterowania jest uzyskanie możliwie najprostszego opisu wykonywania programu. Sekwencyjnie następujące po sobie instrukcje grupowane są w większe bloki opisane przez wypadkowy rozkład losowy czasu realizacji. Konstrukcje dopuszczające wybór jednej z wielu ścieżek, a co za tym idzie grup instrukcji zastępowane są odpowiednim rozkładem losowym uwzględniającym prawdopodobieństwo każdej z decyzji.



Rys. 2-2. Graf przepływu sterowania i odpowiadający mu graf zredukowany

Przykład redukcji prostego grafu przedstawia rysunek 2-2. Instrukcja wyboru oraz pętla zostaje zastąpiona pojedynczym przejściem uwzględniającym całkowitą liczbę instrukcji koniecznych do wykonania. Rozkład losowy ilości instrukcji przypisanych do przejść zastępczych (oznaczonych pustym prostokątem) uwzględnia prawdopodobieństwa wyborów poszczególnych dróg GPS oraz nieprzewidywalnej statycznie ilości powtórzeń pętli.

W rzeczywistym programie decyzje wyborów poszczególnych ścieżek zależą od wartości przetwarzanych danych oraz od stopnia zaawansowania programu. Wiele instrukcji wyboru jest ze sobą skorelowanych. Aby jednak móc traktować graf przepływu sterowania jako proces stochastyczny o akceptowalnym stopniu komplikacji zrezygnowano z analizy wartości przetwarzanych danych a co za tym idzie z rozpatrywania korelacji instrukcji warunkowych.

W pracy zaprezentowano kilka metod zmniejszania rozmiaru GPS. Można je podzielić na dwie kategorie:

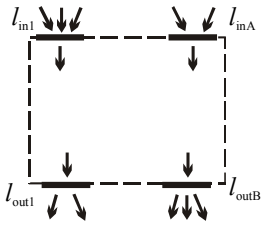
- metody ogólne (linowa i kombinatoryczna)
- metody szczególne (konstrukcje if, while, switch, fork-join)

Podobnie jak w większości publikowanych modeli i metod wykorzystujących sieć Petriego jako odwzorowanie struktury logicznej programu [Mag89] [Bra86] [Jar87] [Deg88] [Car88], w niniejszej pracy przyjęto, że istotne są korelacje wynikłe ze struktury grafu, natomiast wybory poszczególnych przejść są niezależne

Analiza całego grafu jest przeprowadzana etapami poprzez grupowanie najmniejszych elementów w większe. Scalanie fragmentów prowadzone jest od najniższego do najwyższego poziomu. Dla najczęściej wykorzystywanych konstrukcji języków programowania podano gotowe wzory ułatwiające redukcję grafu. Dla niestandardowych powiązań stosowana jest jedna z metod ogólnych umożliwiających redukcję podgrafu.

Podgraf izolowany – definicja i właściwości

Ogólne metody redukcji GPS umożliwiają zastąpienie pewnego fragmentu sieci pojedynczym przejściem lub grupą przejść. Aby można było dokonać takiej redukcji fragment ten musi być połączony z resztą sieci jedynie za pomocą wyróżnionych przejść.



Rys. 2-3. Podgraf izolowany o A wejściach i B wyjściach

Def. 2-4

Fragment grafu przepływu sterowania $GR_x=<M_x, L_x, K_x, U_{x0}, J_x> \subset GPS=<M, L, K, U_0, J>$ *z wyróżnionymi zbiorami przejść* $L_{in}=\{l_{in1}, l_{in2}, ..., l_{inA}\} \subset L_x$ *i* $L_{out}=\{l_{out1}, l_{out2}, ..., l_{outB}\} \subset L_x$ *jest podgrafem izolowanym jeżeli spełnia następujące warunki:*

- *wszystkie następniki miejsc podgrafu należą do podgrafu:*

$$\forall m_i \in M_x \ \forall l_j \in L:(m_i,l_j) \in K \rightarrow l_j \in L_x,$$

- *wszystkie poprzedniki miejsc podgrafu należą do podgrafu:*

$$\forall m_i \in M_x \ \forall l_j \in L:(l_j,m_i) \in K \rightarrow l_j \in L_x,$$

- *następniki przejść z wyjątkiem wyjściowych należą do podgrafu:*

$$\forall l_j \in L_x/\{L_{out}\} \ \forall m_i \in M:(l_j,m_i) \in K \rightarrow m_i \in M_x,$$

- *poprzedniki przejść z wyjątkiem wejściowych należą do podgrafu:*

$$\forall l_j \in L_x/\{L_{in}\} \ \forall m_i \in M:(m_i,l_j) \in K \rightarrow m_i \in M_x.$$

Z powyższych warunków wynika, że wszystkie krawędzie łączące miejsca i przejścia podgrafu muszą należeć do podgrafu.

Skoro podgraf izolowany jest połączony z resztą sieci jedynie za pomocą wyróżnionych przejść wejściowych i wyjściowych, procesy dziejące się w jego wnętrzu nie mogą wpływać na pozostałe elementy sieci bez pośrednictwa przejść wyjściowych. Podobnie otoczenie podgrafu może wpłynąć na jego stan jedynie poprzez przejścia wejściowe. W celu uproszczenia analizy działania podgrafu wprowadzono pojęcie ścieżki.

Def. 2-5

Ścieżką prostą w q podgrafie izolowanym Gx nazywany będzie ciąg odpaleń przejść podgrafu prowadzący od przejścia wejściowego do dowolnego przejścia w podgrafie.

Ścieżka prosta prowadząca do jednego z przejść wyjściowych nazywana będzie ścieżką prostą pełną w q podgrafie Gx.

Ścieżka w podgrafie opisuje drogę, jaką przebywa znacznik wprowadzony do niego przez przejście wyjściowe do momentu pochłonięcia lub opuszczenia grafu przez przejście wyjściowe. Opisuje, zatem jeden z możliwych sposobów, w jaki podgraf izolowany reaguje na pobudzenie polegające na odpaleniu jednego z przejść wejściowych. W niektórych grafach pobudzenie przejścia wejściowego może spowodować proces odpaleń prowadzący do powstania większej ilości znaczników i współbieżnego odpalania wielu przejść. Nie jest możliwe przedstawienie takiego procesu za pomocą ścieżki prostej, gdyż odpalanie przejść nie następuje

w sposób przyczynowy i sekwencyjny. Warunki wystarczające dla zapewniania istnienia jedynie ścieżek prostych precyzuje poniższe twierdzenia.

W pracy [Mag89] pokazano, że:

Twierdzenie 2-1

Podgraf izolowany zawierający jedynie miejsca i przejścia należące do jednego procesu spełnia warunki:

- *w stanie początkowym żadne z miejsc podgrafu nie zawiera znaczników*

$$U_{x0} = 0$$

- *wewnątrz podgrafu każde przejście posiada co najwyżej jedną krawędź wyjściową*

$$\forall l_j \in L_x/L_{out} \rightarrow ||\{k \in K_x: (l_j,m_i) \in K_x\}|| \leq 1$$

- *wewnątrz podgrafu każde przejście ma jedną krawędź wejściową*

$$\forall l_j \in L_x/L_{in} \rightarrow ||\{k \in K_x: (m_i,l_i) \in K_x\}|| = 1$$

Dowód oparto na wykazaniu, że rozpatrywany graf jest siecią bezpieczną, co w połączeniu z brakiem znaczników w stanie początkowym zapewnia spełnienie wymaganych warunków.

Twierdzenie 2-2

Jeżeli podgraf izolowany GR_X spełnia następujące warunki:

- *w stanie początkowym żadne z miejsc podgrafu nie zawiera znaczników*

$$U_{x0} = 0$$

- *wewnątrz podgrafu każde przejście posiada co najwyżej jedną krawędź wyjściową*

$$\forall l_j \in L_x/L_{out} \rightarrow ||\{k \in K_x: (l_j,m_i) \in K_x\}|| \leq 1$$

- *wewnątrz podgrafu każde przejście ma jedną krawędź wejściową*

$$\forall l_j \in L_x/L_{in} \rightarrow ||\{k \in K_x: (m_i,l_i) \in K_x\}|| = 1$$

to w odpowiedzi na pojedyncze pobudzenie nastąpi odpalenie sekwencji przejść dających się opisać ścieżką prostą.

Dowód:

Z warunku $\forall l_j \in L_x/L_{in} \rightarrow ||\{k \in K_x: (m_i,l_i) \in K_x\}|| = 1$ wynika, że wewnątrz podgrafu nie ma miejsc generujących znaczniki (bez krawędzi wejściowych).

Z kolei z warunku $\forall l_j \in L_x/L_{out} \rightarrow ||\{k \in K_x: (l_j,m_i) \in K_x\}|| \leq 1$ wynika, że w wyniku odpalania przejść wewnętrznych podgrafu liczba znaczników wewnątrz podgrafu nie może ulec zwiększeniu.

Zatem do momentu odpalenia przejścia wejściowego w podgrafie nie ma znaczników. W wyniku pojedynczego odpalenia jednego z przejść wejściowych w podgrafie pojawia się jeden i tylko jeden znacznik, który w wyniku odpaleń kolejnych przejść wewnętrznych może przemieszczać się z miejsca do miejsca. Odpalanie przejść wewnętrznych podgrafu nie powoduje zwiększenia ilości znaczników. Odpalenie przejścia wyjściowego lub ślepego oznacza pochłonięcie znacznika.

Z właściwości sieci Petriego wynika, że skoro w podgrafie może znajdować się co najwyżej jeden znacznik oraz każde przejście wymaga pobrania znacznika dla odpalenia to odpalanie przejść może zachodzić jedynie

pojedynczo, w sposób sekwencyjny. Odpalanie przejść następujące w wyniku pobudzenia grafu tworzy zatem ścieżkę prostą.

c.b.d.o.

Z twierdzeń 2-1 i 2-2 oraz przyjętych założeń wynika możliwość redukcji wnętrza podgrafu izolowanego.

Twierdzenie 2-3

Jeżeli podgraf izolowany spełnia założenia twierdzenia 2-2, to dowolne przekształcenie wnętrza podgrafu pod warunkiem zachowania rozkładu losowego reakcji (odpalania przejść wyjściowych, po określonym czasie) na pobudzenie (odpalanie przejść wejściowych) nie zmienia działania zewnętrza podgrafu.

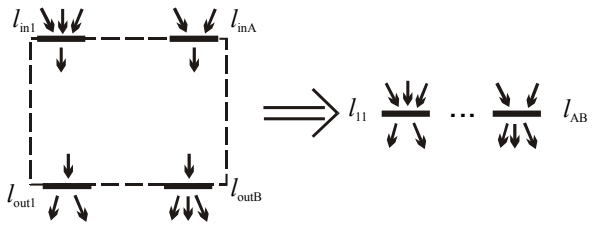
Dowód:

Jeżeli graf GR_X jest grafem izolowanym to z definicji wynika, że jest on w stanie oddziaływać na swoje zewnętrze jedynie poprzez oznaczone przejścia wyjściowe tworzące zbiór $L_{out}=\{l_{out1}, l_{out2}, ..., l_{outB}\}\subset L_x$. Podobnie zewnętrze grafu jest w stanie oddziaływać na jego wnętrze jedynie poprzez przejścia wejściowe tworzące zbiór $L_{in}=\{l_{in1}, l_{in2}, ..., l_{inA}\}\subset L_x$. Założenia dowodzonego twierdzenia zawarte w twierdzeniu 2-1 wykluczają samorzutne powstawanie znaczników wewnątrz grafu, jak również ich obecność przed zaistnieniem pobudzenia. Bez odpalenia jednego z przejść wejściowych nie jest możliwe zatem odpalenie jakiegokolwiek przejścia wewnątrz podgrafu. Pobudzenie powoduje przemieszczanie się znacznika w grafie po jednej z dopuszczalnych ścieżek prostych aż do momentu odpalenia przejścia wyjściowego lub pochłonięcia znacznika. Z definicji podgrafu izolowanego do momentu odpalenia przejścia wyjściowego zmiany odpalenia przejść wewnątrz podgrafu nie mogą być obserwowalne na zewnątrz gdyż nie istnieją krawędzie łączące przejścia wewnętrzne z elementami zewnętrza podgrafu. Analogicznie stan zewnętrza podgrafu nie ma wpływu na odpalanie przejść wewnętrznych z powodu braku wymienionych krawędzi.

Jeżeli zatem procesy dziejące się wewnątrz podgrafu nie są obserwowalne na zewnątrz, to oznacza, że struktura wewnętrzna podgrafu może być dowolnie przekształcona przy zachowaniu obserwowalnych właściwości podgrafu, czyli reakcji przejść wyjściowych na pobudzenie przejść wejściowych.

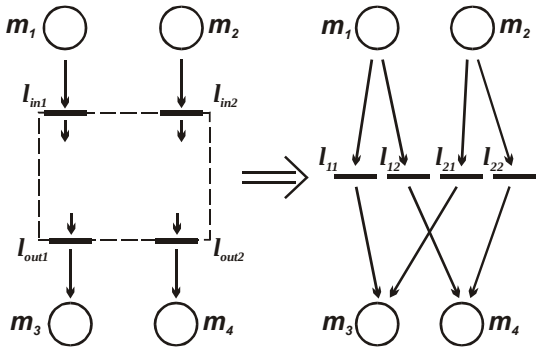
c.b.d.o.

Wnętrze podgrafu o A wejściach i B wyjściach można zredukować zastępując go zbiorem przejść o liczności A×B. Każde z przejść wynikowych zastępuje parę przejść: wejściowe i wyjściowe. Zbiór o liczności A×B zawiera wszystkie możliwe kombinacje wejść i wyjść, zatem w sposób kompletny opisuje reakcję wnętrza podgrafu na pobudzenie.



Rys. 2-4. Redukcja grafu o A wejściach i B wyjściach

Pewnego wyjaśnienia wymagać może sposób połączenia nowo otrzymanych przejść z resztą sieci. Dla każdego przejścia bazującego na przejściu wejściowym l_x krawędzie wejściowe prowadzą od wszystkich poprzedników przejścia l_x. Dla każdego przejścia bazującego na wyjściu l_y krawędzie wyjściowe prowadzą do wszystkich następników przejścia l_y.



Rys. 2-5. Łączenie przejść zastępczych z resztą sieci Petriego

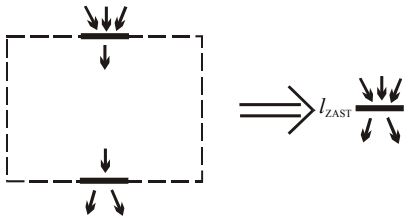
Przejście zastępcze l_{xy} odpowiada pobudzeniu podgrafu przez przejście $l_{in\,x}$ i uzyskaniu reakcji w postaci odpalenia przejścia $l_{out\,y}$. Zatem przejścia l_{x1}, l_{x2}, ..., l_{xB} konkurują pomiędzy sobą o znacznik, który w grafie przed redukcją zawsze był pobierany przez przejście $l_{in\,x}$. Prawdopodobieństwo odpalenia przejścia l_{xy} jest zatem równe prawdopodobieństwu wyboru ścieżki prowadzącej do przejścia wyjściowego $l_{out\ y}$ pod warunkiem pobudzenia grafu przez wejście $l_{in\,x}$.

$$p_{xy} = p[l_{out\,y} \mid l_{in\,x}] \tag{2-5}$$

2.5. Liniowa metoda redukcji

Liniowa metoda redukcji wykorzystuje zapis funkcjonowania podgrafu w postaci równań liniowych. Jej zaletą jest wielomianowa złożoność obliczeniowa – rozwiązanie sprowadza się do rozwiązania układu równań linowych. Stosując tę metodę można obliczyć średnią liczbę instrukcji koniecznych do wykonania, nie uzyskuje się natomiast informacji na temat kształtu rozkładu ani jego wariancji. Wiele publikacji dopuszcza uproszczenie opisu rozkładu losowego czasu wykonywania programu do podania jego wartości oczekiwanej [Mol82] [Jar87] [Mag89].

Redukcji metodą liniową można poddać izolowany fragment grafu posiadający jedno przejście wejściowe i jedno wyjściowe. Wymagane jest przy tym, aby istniał zbiór ścieżek prostych w redukowanym podgrafie oraz aby wartość oczekiwana czasu wykonania instrukcji dla podgrafu była skończona. Taki graf ma zatem jedynie ścieżki proste pełne (wynika z twierdzenia 2-2 oraz skończonej wartości oczekiwanej czasu realizacji). W wyniku redukcji otrzymuje się pojedyncze przejście zastępcze.



Rys. 2-6. Redukcja izolowanego fragmentu grafu z jednym wejściem i wyjściem

Pierwszym krokiem metody liniowej jest obliczenie wartości oczekiwanej ilości wejść znacznika do każdego z miejsc (po pobudzeniu podgrafu przez przejście wejściowe). Dla podgrafu cyklicznego liczba wejść znacznika do miejsca może być większa od jedności. Nie należy mylić ilości wejść znacznika do miejsca z wartością oczekiwaną czasu przebywania w danym miejscu, często liczoną przy analizie procesów losowych.

Przy obliczaniu wartości oczekiwanej ilości wejść znacznika do danego miejsca korzysta się z faktu, że aby wejść do danego miejsca musiał on się najpierw znaleźć w jednym z miejsc poprzedzających. Na tym etapie nie bierze się pod uwagę czasu realizacji przejść.

Twierdzenie 2-4

Wartość oczekiwana ilości wejść znacznika do danego miejsca m_i podgrafu izolowanego po jego pobudzeniu przez przejście wejściowe jest sumą iloczynów wartości oczekiwanych ilości wejść znacznika do miejsc go poprzedzających i prawdopodobieństw, że znacznik przejdzie z nich do danego miejsca, plus prawdopodobieństwo, że znacznik znajdzie się w miejscu m_i bezpośrednio na skutek odpalenia przejścia wejściowego.

$$E_i = a_i + \sum_{j:m_j \in M} E_j p_{ji} \tag{2-6}$$

gdzie:

E_i – wartość oczekiwana ilości wejść znacznika do miejsca m_i

p_{ji} – wartość prawdopodobieństwa przejścia znacznika z miejsca m_j do m_i

$$p_{ji} = \begin{cases} \Phi_{CZh}(+\infty) \Leftrightarrow \exists l_h : (m_j, l_h) \in K, (l_h, m_i) \in K \\ 0 \Leftrightarrow \sim \exists l_h : (m_j, l_h) \in K, (l_h, m_i) \in K \end{cases}$$

a_i – współczynnik określający połączenie miejsca m_i z przejściem wejściowym

$$a_i = \begin{cases} 1 \Leftrightarrow (l_{in}, m_i) \in K \\ 0 \Leftrightarrow (l_{in}, m_i) \notin K \end{cases}$$

Dowód:

Z własności sieci Petriego oraz twierdzenia 2-2 o braku znaczników w niepobudzonym podgrafie wynika, że znacznik może pojawić się w miejscu m_i należącym do podgrafu jedynie w wyniku odpalenia jednego z przejść, którego krawędź wyjściowa dochodzi do m_i. Zatem liczba wejść znacznika N_i do miejsca m_i jest równa sumie liczb NL_x odpaleń poprzedzających go przejść l_x.

$$N_i = \sum_{x:(l_x, m_i) \in K} NL_x \tag{2-7}$$

Wartość oczekiwana E_i liczby wejść znacznika do miejsca m_i jest więc równa sumie wartości oczekiwanych EL_x odpaleń poprzedzających go przejść l_x (wszystkie wartości oczekiwane muszą być skończone zatem EΣ jest równe ΣE)

$$E_i = \mathrm{E} \sum_{x:(l_x,m_i)\in K} L_x = \sum_{x:(l_x,m_i)\in K} \mathrm{E}\, L_x \tag{2-8}$$

Dla przejść wewnętrznych l_x podgrafu wartość oczekiwana ilości odpaleń danego przejścia jest równa liczbie wejść znacznika do miejsca m_i poprzedzającego przejście l_x pomnożonej przez prawdopodobieństwo p_{ix}, że znacznik opuści miejsce m_i właśnie przez przejście l_x.

$$EL_x = E_i p_{ix} \tag{2-9}$$

Dla przejścia wejściowego l_{in} przyjmuje się wartość oczekiwaną liczby odpaleń równą jeden, gdyż analiza dotyczy reakcji podgrafu na pojedyncze pobudzenie przejścia l_{in}. Wynika to z faktu, iż podgraf izolowany jest fragmentem grafu procesu sekwencyjnego, czyli zawierającego pojedynczy znacznik opisujący stan procesu.

Z twierdzenia 2-2 wynika, iż dla każdego przejścia wewnętrznego l_x istnieje jedno i tylko jedno miejsce poprzedzające m_j. Oznaczając przez m_j miejsce poprzedzające przejście l_x czyli takie że: $(m_j,\ l_x)\in K$ otrzymujemy:

dla miejsc do których nie dochodzi krawędź z przejścia:

$$E_i = \sum_{x:(l_x,m_i)\in K} E_j p_{ij} \qquad \text{dla miejsc } m_i \text{ takich że } (l_{in}, m_i)\notin K \tag{2-10}$$

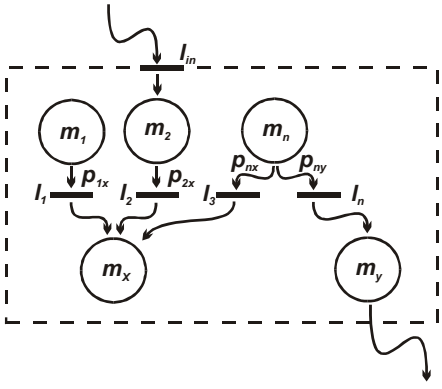
dla miejsc dla których istnieje krawędź dochodząca od przejścia l_{in}:

$$E_i = NL_{in} + \sum_{x:(l_x,m_i)\in K} E_j p_{ij} \qquad \text{dla miejsc } m_i \text{ takich że } (l_{in}, m_i)\in K \tag{2-11}$$

c.b.d.o.

Na przykład dla fragmentu sieci przedstawionego na poniższym rysunku wartość oczekiwana ilości wejść znacznika do miejsca m_x wynosi:

$$E_x = E_1 p_{1x} + E_2 p_{2x} + E_n p_{nx} \tag{2-12}$$



Rys. 2-7. Obliczanie wartości oczekiwanej jako kombinacji liniowej

Prawdopodobieństwa przejść znaczników są znane i wynikają ze struktury grafu, zatem wartość oczekiwana ilości wejść znacznika do każdego z miejsc jest opisana równaniem liniowym. Otrzymuje się zatem układ równań:

$$\begin{bmatrix} E_1 \\ \dots \\ E_n \end{bmatrix} = \begin{bmatrix} p_{11} & \dots & p_{n1} \\ \dots & \dots & \dots \\ p_{1n} & \dots & p_{nn} \end{bmatrix} \times \begin{bmatrix} E_1 \\ \dots \\ E_n \end{bmatrix} + \begin{bmatrix} a_1 \\ \dots \\ a_n \end{bmatrix} \tag{2-13}$$

Gdzie:

$[E_1,\dots E_n]$ – wektor poszukiwanych wartości oczekiwanych ilości wejść znaczników do miejsc $m_1, \dots m_n$

$[p_{ij}]$ - znane prawdopodobieństwa przejść pomiędzy miejscami podgrafu

$[a_1, \dots a_n]$ – wektor wyrazów wolnych – istnienie krawędzi od przejścia wejściowego do miejsca $m_1, \dots m_n$

Wynikiem metody liniowej jest obliczenie wartości oczekiwanej ilości instrukcji, jakie musi wykonać maszyna w odpowiedzi na pojedyncze pobudzenie podgrafu izolowanego, aby uzyskać odpowiedź na jego wyjściu. Skoro rozpatrywana jest reakcja grafu na pojedyncze pobudzenie, to wartość oczekiwana ilości odpaleń przejścia wejściowego jest równa jeden. Możliwość docierania znacznika z zewnątrz podgrafu opisuje wektor wyrazów wolnych, którego elementy mają następujące wartości:

$$a_i = \begin{cases} 1 \Leftrightarrow (l_{in}, m_i) \in K \\ 0 \Leftrightarrow (l_{in}, m_i) \notin K \end{cases} \tag{2-14}$$

Metoda liniowa znajduje zastosowanie jedynie do podgrafów o pojedynczym przejściu wejściowym, co w połączeniu z ograniczeniem ilości krawędzi odchodzących od przejścia wewnątrz podgrafu powoduje, że wektor wyrazów wolnych może mieć tylko jedną niezerową współrzędną.

Po przekształceniu równania otrzymuje się:

$$\begin{bmatrix} p_{11}-1 & p_{21} & \dots & p_{(n-1)1} & p_{n1} \\ p_{12} & p_{22}-1 & \dots & p_{(n-1)2} & p_{n2} \\ \dots & \dots & \dots & \dots & \dots \\ p_{1(n-1)} & p_{2(n-1)} & \dots & p_{(n-1)(n-1)}-1 & p_{n(n-1)} \\ p_{1n} & p_{2n} & \dots & p_{(n-1)n} & p_{nn}-1 \end{bmatrix} \times \begin{bmatrix} E_1 \\ E_2 \\ \dots \\ E_{n-1} \\ E_n \end{bmatrix} = \begin{bmatrix} -a_1 \\ -a_2 \\ \dots \\ -a_{n-1} \\ -a_n \end{bmatrix} \tag{2-15}$$

Powyższy układ równań liniowych ma rozwiązanie, gdy wyznacznik główny macierzy współczynników jest różny od zera. Można zatem sformułować następujące twierdzenie:

Twierdzenie 2-5

Jeżeli wartość oczekiwana ilości instrukcji fragmentu programu opisywanego przez podgraf jest skończona oraz podgraf nie zawiera fragmentów nie połączonych z resztą sieci to wyznacznik macierzy **(P-1)** *jest różny od zera.*

Dowód (nie wprost):

Przyjmijmy, że wyznacznik macierzy **(P-1)** jest równy zero mimo spełnienia wymaganych założeń. Wynika z tego, że istnieje możliwość utworzenia wiersza złożonego z samych zer jako kombinacji liniowej wierszy macierzy. Zatem musi istnieć zbiór współczynników $\alpha_1, \dots, \alpha_n$ takich że co najmniej jeden z nich jest różny od zera oraz:

(2-16)

$$\begin{bmatrix}\alpha_1 & \alpha_2 & \dots & \alpha_{n-1} & \alpha_n\end{bmatrix}\times\begin{bmatrix}p_{11}-1 & p_{21} & \dots & p_{(n-1)1} & p_{n1}\\ p_{12} & p_{22}-1 & \dots & p_{(n-1)2} & p_{n2}\\ \dots & \dots & \dots & \dots & \dots\\ p_{1(n-1)} & p_{2(n-1)} & \dots & p_{(n-1)(n-1)}-1 & p_{n(n-1)}\\ p_{1n} & p_{2n} & \dots & p_{(n-1)n} & p_{nn}-1\end{bmatrix}=\begin{bmatrix}0 & 0 & \dots & 0 & 0\end{bmatrix}$$

W skróconym zapisie:

$$\mathbf{A}\times(\mathbf{P}-\mathbf{1})=\mathbf{0} \tag{2-17}$$

W i-tej kolumnie macierzy **P** znajdują się prawdopodobieństwa wyboru jednej z dróg wyjścia znacznika z miejsca m_i. Wybory poszczególnych przejść są zdarzeniami wykluczającymi się, zatem suma prawdopodobieństw w każdej kolumnie macierzy **P** zawiera się zawsze w przedziale domkniętym [0, 1]. Wobec tego suma elementów każdej kolumny **(P-1)** zawiera się w przedziale [-1, 0].

Dla potrzeb dowodzenia twierdzenia o istnieniu rozwiązania prezentowanego równania miejsca wewnętrzne podgrafu zostaną ponumerowane ze względu na ich powiązanie z przejściem wyjściowym podgrafu. Oznaczmy numerem jeden miejsce, z którego prowadzi krawędź do przejścia wyjściowego podgrafu, czyli $(m_1,l_{\text{out}})\in K$. Z założenia o spójności podgrafu wynika, że miejsce takie istnieje.

Ponieważ z miejsca m_1 wychodzi na zewnątrz podgrafu krawędź o niezerowym prawdopodobieństwie wyboru, więc suma elementów pierwszej kolumny jest mniejsza od zera (nie może być równa).

W kombinacji liniowej mającej dać zero pierwszy element jest równy:

$$0=-1\alpha_1+(\alpha_1 p_{11}+\dots+\alpha_n p_{1n}) \tag{2-18}$$

Rozpatrzmy dwa przypadki dla $\alpha_1\neq 0$ i $\alpha_1=0$.

Niech $\alpha_1\neq 0$ wtedy:

Jeżeli istnieje zbiór współczynników α, taki, że $\alpha_l\neq 0$, to dla uproszczenia przyjmijmy, że α_l jest dodatnie. Jeżeli bowiem istnieje kombinacja liniowa dająca zero dla współczynników $(\alpha_1, \dots, \alpha_n)$ to istnieje również dla $(-\alpha_1, \dots, -\alpha_n)$.

Suma prawdopodobieństw w pierwszej kolumnie jest mniejsza od jedności, zatem aby uzyskać zerowy wynik kombinacji liniowej:

$$0=-1\alpha_1+(\alpha_1 p_{11}+\dots+\alpha_n p_{1n}) \tag{2-19}$$

co najmniej jeden ze współczynników $(\alpha_2, \dots, \alpha_n)$ musi być większy od α_l zatem:

$\alpha_1 < \text{MAX}(\alpha_2,\dots,\alpha_n)$ z czego wynika $\alpha_1 < \text{MAX}(\alpha_1,\dots,\alpha_n)$

Podobnie gdy $\alpha_1=0$ otrzymujemy:

Jeżeli istnieje zbiór współczynników α, taki że $\alpha_1=0$ to aby suma:

$$z_1=-1\alpha_1+(\alpha_1 p_{11}+\dots+\alpha_n p_{1n})=(\alpha_2 p_{11}+\dots+\alpha_n p_{1n}) \tag{2-20}$$

była równa zero co najmniej jeden ze współczynników musi być dodatni i co najmniej jeden ujemny (bo prawdopodobieństwa są nieujemne). Prawdziwa jest zatem nierówność:

$\alpha_1 < \text{MAX}(\alpha_2,...,\alpha_n)$ co jest równoważne $\alpha_1 < \text{MAX}(\alpha_1,...,\alpha_n)$

Wynika z tego, że dla dowolnego otrzymujemy:

$$\alpha_1 < \text{MAX}(\alpha_1,...,\alpha_n) \tag{2-21}$$

Niech teraz m_2 będzie miejscem, które poprzedza m_1 w podgrafie. Z definicji spójności grafu miejsce takie istnieje. Ponieważ poprzedza ono m_1 więc $p_{21} > 0$. Dla drugiej i dalszych kolumn suma prawdopodobieństw może być zarówno mniejsza jak i równa jedności zatem:

$$\alpha_2 \le \text{MAX}(\alpha_1, \alpha_3...,\alpha_n) \tag{2-22}$$

jako że:

$$0 = -1\alpha_2 + (\alpha_1 p_{11} + ... + \alpha_n p_{1n}) \tag{2-23}$$

Aby otrzymać zerową wartość kombinacji liniowej możliwe są dwa alternatywne (acz niewykluczające się) przypadki

1) $\alpha_2 < \text{MAX}(\alpha_1, \alpha_3...,\alpha_n)$ z czego wynika $\alpha_2 < \text{MAX}(\alpha_1,...,\alpha_n)$
2) $\forall i : p_{2i} > 0 \rightarrow \alpha_i = \alpha_2$ tylko gdy suma prawdopodobieństw w kolumnie jest równa jeden. Ponieważ jednak p_{21} jest niezerowe, więc α_1=α_2 a dla α_1 zachodzi ostra nierówność to przypadek drugi redukuje się do pierwszego czyli $\alpha_2 < \text{MAX}(\alpha_1,...,\alpha_n)$.

Oznaczmy indeksem trzy miejsce, które poprzedza m_1 lub m_2. Z założenia o spójności podgrafu wynika, że miejsce takie istnieje. W sposób analogiczny jak dla m_2 otrzymujemy:

$$\alpha_3 < \text{MAX}(\alpha_1,...,\alpha_n) \tag{2-24}$$

Czynność powtarzamy kolejno dla pozostałych miejsc otrzymując zbiór nierówności o ogólnej postaci:

$$\forall i \in 1..n \rightarrow \alpha_i < \text{MAX}(\alpha_1,...,\alpha_n) \tag{2-25}$$

Zatem w wektorze współczynników α obowiązuje zasada, że dla każdego współczynnika istnieje od niego większy. Jest to sprzeczny układ nierówności. Nie da się uzyskać wiersza samych zer jako kombinacji liniowej wierszy macierzy, zatem wyznacznik nie może być równy zero.

c.b.d.o.

Rozwiązując układ równań oblicza się wartości oczekiwane E_x liczby wejść znacznika do każdego z miejsc podgrafu. Na podstawie E_x oblicza się wartości oczekiwane e_y liczby odpaleń każdego z przejść.

$$e_y = p_{ij} E_i : (m_i, l_y) \in K, (l_y, m_j) \in K \tag{2-26}$$

W podgrafie może znajdować się co najwyżej jeden znacznik, zatem wszystkie przejścia, które mogą być odpalone, mają jedno i tylko jedno poprzedzające je miejsce. Wartość oczekiwana ilości odpaleń przejścia jest zatem równa wartości oczekiwanej wejść znacznika do poprzedzającego miejsca m_i pomnożonej przez prawdopodobieństwo, że znacznik opuści miejsce poprzez odpalenie danego przejścia l_y, czyli dotrze do miejsca m_j znajdującego się za przejściem. Ponieważ wyliczone uprzednio wartości oczekiwane E_i liczby wejść znacznika do miejsc są skończone oraz $p_{ij} \le 1$ więc wartości e_x są również skończone.

Liczba instrukcji, jaka musi zostać wykonana, aby w odpowiedzi na pobudzenie przejścia wejściowego nastąpiło odpalenie przejścia wyjściowego jest równa sumie liczby instrukcji koniecznych do odpalenia każdego z użytych przejść. Zatem wartość oczekiwana ilości instrukcji koniecznych do „przebycia" podgrafu jest równa sumie iloczynów średniej ilości odpaleń każdego z przejść i średniej ilości instrukcji koniecznych do odpalenia przejścia.

$$E(O) = \sum_{i:l_i \in L_x} e_i E(t_i) \tag{2-27}$$

Liniowa metoda redukcji podgrafu umożliwia określenie wartości oczekiwanej liczby instrukcji wykonywanych przez fragment programu reprezentowany przez podgraf. W metodzie tej rozpatruje się jedynie liczbę wejść i wyjść znacznika do każdego z miejsc bez analizy, jaką drogą tam dotarł i jaką podążył aż do opuszczenia grafu. W związku z tym nie jest możliwe określenie wyższych momentów rozkładu losowego ilości instrukcji, ani też jego kształtu.

2.6. Kombinatoryczna metoda redukcji

Metodę kombinatoryczną można zastosować do podgrafu izolowanego o A wejściach i B wyjściach, dla którego reakcja na każde pobudzenie da się zapisać jako ścieżka prosta. Nie jest przy tym wymagane, aby wartość oczekiwana liczby instrukcji była skończona. Dopuszczalne jest pochłonięcie znacznika wewnątrz podgrafu.

Metoda kombinatoryczna wymaga przeanalizowania wszystkich możliwych ścieżek prostych w podgrafie, skutkiem czego jest bardzo złożona obliczeniowo. Umożliwia jednak uzyskanie funkcji dystrybuanty rozkładu wypadkowego, a nie tylko wartości średniej.

Oznaczmy przez $Q=\{q_1, \ldots q_n\}$ zbiór wszystkich możliwych ścieżek pełnych w analizowanym grafie. Przy pobudzeniu podgrafu izolowanego pojedynczym odpaleniem jednego z przejść wejściowych wybór jednej ze ścieżek jest zdarzeniem wykluczającym wybór innej ścieżki. Prawdopodobieństwo wyboru ścieżki q_i oznaczono jako $p(q_i)$. Wybór dowolnej ścieżki pełnej ze zbioru Q nie jest zdarzeniem pewnym, gdyż na skutek modelowanych uszkodzeń oprogramowania znacznik może nie opuścić podgrafu. Podgraf izolowany stanowi fragment grafu pojedynczego sekwencyjnego procesu, zatem rozpatrywane będą jedynie reakcje na pojedyncze pobudzenie.

$$\sum_{q_i \in Q} p(q_i) \le 1 \tag{2-28}$$

Dla przebycia danej ścieżki konieczne jest odpalenie wszystkich przejść wchodzących w jej skład. Prawdopodobieństwo wyboru danej ścieżki jest zatem iloczynem prawdopodobieństw odpaleń wszystkich przejść l_i wchodzących w jej skład.

W przypadku istnienia pętli i wielokrotnego odpalania tego samego przejścia zawiera się ono wielokrotnie w ścieżce. Prawdopodobieństwo odpalenia przejścia wejściowego i wyjściowego również jest uwzględnione, gdyż należą one do podgrafu.

$$p(q_x) = \prod_{i:l_i \in q_x} p(l_i) \tag{2-29}$$

Liczba instrukcji t_x konieczny do wykonania całej ścieżki q_x jest sumą czasów t_i realizacji instrukcji koniecznych do wykonania poszczególnych etapów (przejść) danej ścieżki.

$$t_x = \sum_{i:l_i \in q_x} t_i \tag{2-30}$$

Gdy liczba instrukcji konieczna do odpalenia przejścia nie jest stała, dla drogi q_x określa się dystrybuantę $\Phi_x(t)$ rozkładu losowego czasu realizacji instrukcji wykonywanych w trakcie ścieżki q_x .

$$\Phi_x(t) = \Pr\left[\bigcup_{l_j \in q_x} t_j \le t\right] \tag{2-31}$$

Mając obliczone prawdopodobieństwa p_x wyboru każdej ścieżki $q_x \in Q$ oraz dystrybuantę $\Phi_x(t)$ rozkładu losowego czasu wykonania instrukcji w trakcie realizacji ścieżki q_x można obliczyć dystrybuantę rozkładu losowego czasu realizacji podgrafu.

Dla podgrafu z jednym wejściem i jednym wyjściem dystrybuantę określa wzór na prawdopodobieństwo całkowite.

$$\Phi(t) = \sum_{x:q_x \in Q} p_x \Phi_x(t) \tag{2-32}$$

W powyższym wzorze uwzględniono jedynie ścieżki prowadzące od przejścia wejściowego do wyjściowego. W przypadku modelowania zawodności programu znacznik może zostać pochłonięty wewnątrz podgrafu, a zatem suma prawdopodobieństw ścieżek poprawnych jest mniejsza od jedności. Otrzymana w (2-32) ważona suma dystrybuant będzie wtedy funkcją ograniczoną z góry poprzez (1-Pr[uszkodzenia_programu]).

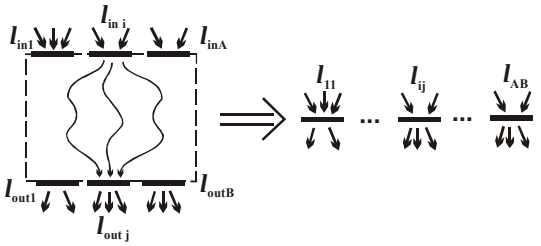
$$\sum_{x:q_x \in Q} p_x \Phi_x(t) \le \left(1 - \Pr\left[uszkodzenia\_programu\right]\right) \tag{2-33}$$

Podgraf izolowany o jednym przejściu wejściowym i jednym wyjściowym w wyniku redukcji zastąpiony zostaje pojedynczym przejściem opisanym dystrybuantą wyliczoną z wzoru (2-32).

Pograf izolowany o A wejściach i B wyjściach w wyniku redukcji zostaje zastąpiony A*B przejściami, z których l_{xy} reprezentuje odpowiedź grafu na pobudzenie przejścia wejściowego l_x jako odpalenie przejścia wyjściowego l_y. Tworząc dla podgrafu izolowanego o A wejściach i B wyjściach macierz dystrybuant określających reakcję na pobudzenie, dokonuje się podziału zbioru ścieżek w grafie na A*B rozłącznych podzbiorów. Każdy z podzbiorów grupuje ścieżki o tych samych przejściach wejściowym i wyjściowym.

$$Q_{ij} = \left\{ q : l_{in\,i}, l_{out\,j} \in q \right\} \;;\; Q = \bigcup_{\substack{1 \le i \le A \\ 1 \le j \le B}} Q_{ij} \tag{2-34}$$

Podzbiór Q_{ij} zawiera wszystkie ścieżki prowadzące od przejścia $l_{in\,i}$ do przejścia $l_{out\,j}$ więc na podstawie tego zbioru tworzony jest rozkład losowy pojedynczego przejścia zastępczego l_{ij} powstałego w wyniku redukcji grafu.



Rys. 2-8. Zbiór ścieżek Q_{ij} i odpowiadające mu przejście l_{ij}.

Dla pobudzenia podgrafu poprzez odpalenie przejścia l_i istnieje wiele możliwych ścieżek, z których tylko część prowadzi do przejścia wyjściowego l_j. Dla każdego przejścia wejściowego l_i oblicza się B prawdopodobieństw p_{i1} ... p_{iB} określających szansę wyboru określonego przejścia wyjściowego $l_{1...}$ l_B jako odpowiedź na pobudzenie l_i.

$$p_{ij} = \sum_{q_x \in Q_{ij}} p(q_x) \tag{2-35}$$

gdzie p(q_x) obliczane jest na podstawie wzoru (2-29).

Wybór konkretnej ścieżki jako odpowiedzi na pobudzenie wyklucza wybór innej, zatem rozkład losowy ilości instrukcji dla przejścia l_{ij} jest złożeniem zdarzeń wykluczających się polegających na wyborze jednej ze ścieżek q_x. Dystrybuanta rozkładu losowego ilości instrukcji dla zastępczego przejścia l_{ij} jest zatem obliczana na podstawie wzoru na prawdopodobieństwo całkowite.

$$\Phi_{ij}(O) = \sum_{q_x \in Q_{ij}} \frac{p(q_x)}{p_{ij}} \Phi_x(O) \tag{2-36}$$

Prawdopodobieństwo p(q_x) wyboru ścieżki q_x obliczone na podstawie wzoru (2-28) uwzględnia wszystkie ścieżki rozpoczynające się przejściem l_i, a do obliczenia dystrybuanty użyte muszą zostać wagi w postaci prawdopodobieństw tylko tych ścieżek, które kończą się przejściu l_j. Waga (iloraz prawdopodobieństw) jest prawdopodobieństwem warunkowym p($q_x|l_j \in q_x$).

Złożoność metody kombinatorycznej:

Metoda kombinatoryczna wymaga przeanalizowania wszystkich możliwych ścieżek w podgrafie.

Właściwość 2-6

W podgrafie acyklicznym istnienie ścieżki, w której przejście l_A poprzedza przejście l_B, wyklucza istnienie ścieżki, w której przejście l_B poprzedza przejście l_A.

Dowód (nie wprost):

Jeżeli istniałyby dwie ścieżki q_x w której l_A poprzedza l_B oraz q_y w której l_B poprzedza l_A to podgraf zawierający obie ścieżki posiadałby pętlę od l_A do l_B i z powrotem do l_A. Nie byłby on zatem acykliczny, co przeczy założeniom twierdzenia.

c.b.d.o.

Twierdzenie 2-7

Dla skończonego podgrafu pozbawionego pętli, w którym jest N przejść, liczba możliwych ścieżek jest skończona i nie większa niż 2^N.

Dowód

Z acykliczności grafu wynika, że żadne przejście grafu nie może występować w ścieżce więcej niż jeden raz. Dla każdej poprawnej ścieżki w podgrafie acyklicznym nie istnieją ścieżki będące jej permutacjami. Zatem wybór ścieżki w podgrafie sprowadza się do wyboru zbioru przejść tworzących ścieżkę ze zbioru przejść w podgrafie bez uwzględniania kolejności. Jeżeli graf zawiera N przejść to liczba możliwych wyborów podzbioru jest równa 2^N (bez uwzględnienia poprawności ścieżki wynikającej ze struktury grafu).

c.b.d.o.

Metoda kombinatoryczna ma zatem złożoność wykładniczą. W rzeczywistym podgrafie acyklicznym występuje wiele zależności pomiędzy odpaleniami poszczególnych przejść, więc liczba możliwych ścieżek jest zdecydowanie mniejsza.

Dla skończonego podgrafu zawierającego pętle, liczba możliwych ścieżek może być nieskończona, gdyż to samo przejście może być odpalane wielokrotnie. W takim przypadku konieczne jest ograniczenia analizy do zbioru najbardziej prawdopodobnych ścieżek, z pominięciem pozostałych. Uproszczenie takie wpływa na dokładność obliczeń, pozwala jednak wykonać je w skończonym czasie.

Nieuwzględnienie ścieżki q_x o prawdopodobieństwie wyboru p_x jest równoważne uznaniu, że z prawdopodobieństwem p_x znacznik nie opuści grafu. Można przyjąć, że pomijane będą ścieżki o największej długości, a co za tym idzie o najmniejszym prawdopodobieństwie wyboru. Dla typowych grafów ścieżki o największej długości będą odpowiadać realizacji (w pętli) bardzo dużej liczby instrukcji. Pominięcie tych ścieżek oznacza uznanie ich za błędne, powodujące pochłonięcie znacznika w podgrafie. W układach współbieżnych zbyt długi czas oczekiwania na kooperujący proces jest często uznawany za błąd. Zatem uproszczenie polegające na zastąpieniu ścieżek o bardzo dużej liczbie instrukcji, ścieżkami błędnymi odpowiada często rzeczywistemu zachowaniu systemu.

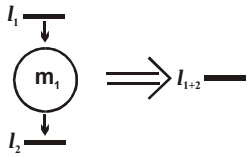
2.7. Redukcja typowych struktur grafu

Zdecydowana większość programów zawiera typowe konstrukcje języków programowania łatwe do zidentyfikowania i przeanalizowania. Poniższe metody redukcji opracowano z myślą o najmniejszych, najbardziej typowych elementach grafu przepływu sterowania. Dzięki algorytmowi zastępowania elementarnych konstrukcji języków programowania pojedynczym przejściem możliwe jest stopniowe zmniejszanie rozmiarów grafu poczynając od najmniejszych elementów i stopniowo przechodząc na coraz wyższy poziom. Otrzymane w wyniku redukcji przejście symbolizujące grupę instrukcji może być następnie użyte jako jedno z wielu elementów będących składnikami następnej redukcji.

Redukcja szeregowych instrukcji

Jeżeli dwie grupy instrukcji wykonywane są sekwencyjnie przez jeden procesor to można zastąpić je pojedynczą grupą instrukcji o sumarycznym czasie wykonania. Rys. 2-9 przedstawia fragment grafu przepływu sterowania obrazujący szeregowe wykonanie grup instrukcji opisanych przez przejścia l_1 i l_2. Warunkiem

koniecznym i wystarczającym do przeprowadzenia opisanej redukcji jest aby miejsce łączące miało jednego i tylko jednego poprzednika oraz jednego i tylko jednego następnika.



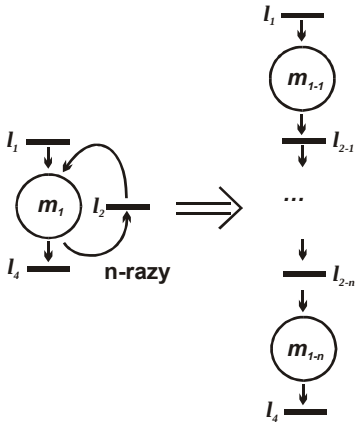
Rys. 2-9. Redukcja szeregowych instrukcji

Rozkład losowy liczby instrukcji koniecznych do wykonania jest rozkładem sumy zmiennych losowych. Jego dystrybuantę oblicza się:

$$\Phi_{1+2}(t) = \Phi_{1+2}(t) * \Phi_{1+2}(t) \tag{2-37}$$

Redukcja pętli stałych

Pętle o znanej podczas kompilacji ilości powtórzeń można zastąpić n-krotnym powtórzeniem wnętrza pętli. Następnie przeprowadzana jest redukcja scalającą n bloków w jeden zastępczy.



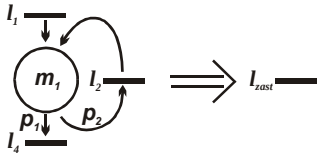
Rys. 2-10. Redukcja pętli o znanej ilości powtórzeń

Współczesne procesory pobierając instrukcje z pamięci przechowują je w buforach, zatem kolejne wykonanie tego samego fragmentu jest szybsze. Zatem n-krotne powtórzenie pętli zazwyczaj zostanie wykonane szybciej niż n-krotne wykonanie tego samego kodu bez użycia pętli. Mimo że instrukcje sterowania pętlą wymagają dodatkowego działania procesora, kolejne powtórzenia pętli dotyczą instrukcji wcześniej pobranych. Wykonanie tego samego kodu n razy wpisanego w treść programu wymaga każdorazowego pobrania nowych instrukcji a zatem jest wolniejsze. W pracy w celu uproszczenia analizy i obliczeń przyjęto założenie o jednakowym czasie wykonywania n-powtórzeń pętli i n-krotnego, sekwencyjnego wykonania kodu wnętrza pętli.

Redukcja pętli o nieznanej z ilości powtórzeń

W każdym nietrywialnym programie występują pętle o ilości powtórzeń zależnej od danych wejściowych, czasem w prosty sposób (od rozmiaru zagadnienia) czasem w nieprzewidywalny (powtarzaj aż do uzyskania rozwiązania o zadanej dokładności). Jeżeli nie bada się wartości przetwarzanych danych to najwygodniej jest

traktować taką pętlę jako blok o losowym czasie wykonania. Określając prawdopodobieństwa wyboru drogi w instrukcji powtarzania można utworzyć zastępczy blok o wypadkowym rozkładzie czasu wykonania (ilości wykonanych instrukcji). Rozkład losowy ilości wymaganych instrukcji będzie charakteryzował się dużą wariancją, gdyż uwzględnia ona wykonanie większej lub mniejszej ilości powtórzeń kodu w pętli. Najprościej potraktować instrukcję sterującą pętlą jako losowanie pomiędzy powtórzeniem a zakończeniem wykonywane niezależnie przy każdym powtórzeniu pętli.



Rys. 2-11. Redukcja pętli o nieznanej liczbie powtórzeń

Rozkład losowy liczby instrukcji określa się na podstawie wzoru na prawdopodobieństwo całkowite, gdzie zdarzeniami są prawdopodobieństwa $p_1p_2^i$ i-krotnego wykonania pętli. Dystrybuanty składowe $\Phi_i(t)$ są dystrybuantami i-krotnej sumy zmiennych losowych liczby instrukcji koniecznych do wykonania przejścia l_2.

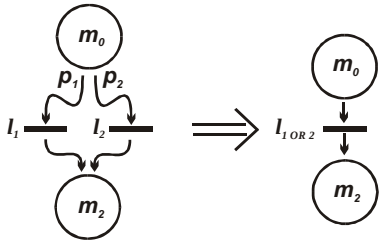
$$\Phi_n(t) = \sum_{i=0..\infty} p_1 p_2^i \Phi_i(t) \tag{2-38}$$

Jeżeli czas realizacji wnętrza pętli jest wartością stałą to $\Phi_n(t)$ jest dystrybuantą rozkładu geometrycznego (losowania do pierwszego sukcesu).

Opis pętli w grafie przepływu sterowania uwzględnia jedynie prawdopodobieństwo powtórzenia pętli, zatem nie daje pełnego obrazu rozkładu losowego ilości wykonań pętli. Na podstawie struktury grafu zakłada się, że liczba wykonań pętli ma rozkład wykładniczy, co nie zawsze musi być prawdziwe. Dopuszczalność takiego uproszczenia leży jednak u podstaw tworzenia GPS bez uwzględniania wartości przetwarzanych danych.

Redukcja instrukcji wyboru

Instrukcja wyboru zakłada, że w trakcie wykonywania programu zostanie zrealizowany jeden i tylko jeden z N bloków składowych wchodzących w skład instrukcji wyboru. Rys. 2-12 przedstawia fragment grafu ilustrujący instrukcję wyboru.



Rys. 2-12. Redukcja instrukcji wyboru

Na podstawie prawdopodobieństwa, z jakim zostanie wybrana każda z dróg dystrybuantę wypadkową określa wzór na prawdopodobieństwo całkowite.

$$\Phi_{1\ OR\ 2}(t)=p_1\Phi_1(t) + p_2\Phi_2(t) \tag{2-39}$$

Opisane metody redukcji umożliwiają zredukowanie grafu jednowątkowego do pojedynczego miejsca. Wiąże się to oczywiście ze stratą dokładności modelu. Dla procesów współbieżnych nie jest dopuszczalne redukowanie miejsc i przejść należących do różnych procesów, gdyż ich wzajemna synchronizacja zależy od parametrów maszyny.

2.8. Właściwości GPS oraz zredukowanego grafu

Redukcja grafu przepływu sterowania umożliwia zmniejszenie go do rozmiarów umożliwiających dalsze etapy analizy. Zredukowany graf charakteryzuje się nie tylko mniejszą ilością miejsc i przejść, ale również bardziej regularną strukturą. Podział grafu na fragmenty w miejscu synchronizacji przedstawiony jest na rys 2-14.

Podział grafu na fragmenty

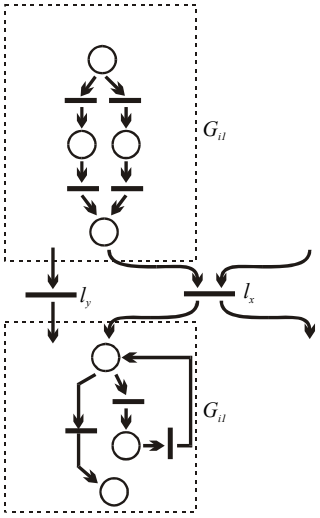
Autor przyjął w pracy założenie (def. 2-1), że każda instrukcja synchronizacji (a więc odpowiadające jej przejście) wykonywana jest z prawdopodobieństwem równym jeden. Dla każdego podgrafu G_i grafu GPS, takiego że G_i zawiera wszystkie miejsca i przejścia opisujące proces P_i, nie zawiera natomiast miejsc i przejść opisujących pozostałe procesy oraz semafory, prawdziwe jest następujące twierdzenie:

Twierdzenie 2-8

Jeżeli przejście l_x jest przejściem synchronizacji, to pograf pozbawiony tego przejścia $G_i/\{l_x\}$ nie jest grafem spójnym, lecz składa się z dwóch nie połączonych ze sobą podgrafów.

Dowód (nie wprost)

Załóżmy, że podgraf $G_i/\{l_x\}$ przedstawiony na rys 2-13 jest podgrafem spójnym. Zatem musi istnieć co najmniej jedno przejście l_y łączące podgraf poprzedzający G_{i1} przejście l_x z podgrafem G_{i2} następującym po l_x. Krawędzie przejścia l_y mogą być skierowane w kierunku G_{i1} lub G_{i2}. Zakładamy, że prawdopodobieństwo odpalenia tego przejścia w wyniku działania programu jest niezerowe i wynosi p_y. (gdyby p_y=0 to przejście l_y odpowiadałoby „martwemu kodowi” czyli de-facto nie istniałoby)



Rys. 2-13. Podział podgrafu G_i na poprzedzający i następujący po przejściu l_x.

Dla przejścia l_y skierowanego od G_{i2} do G_{i1} przejście l_y zostanie odpalone z prawdopodobieństwem p_y, a zatem przejście synchronizacyjne z prawdopodobieństwem p_y zostanie odpalone więcej niż jeden raz. Przeczy to przyjętemu przez autora założeniu o wykonywaniu wszystkich synchronizacji tylko raz i to z prawdopodobieństwem równym jeden.

Dla przejścia l_y skierowanego od G_{i1} do G_{i2} przejście l_y zostanie odpalone z prawdopodobieństwem p_y, a zatem przejście synchronizacyjne zostanie odpalone z prawdopodobieństwem $1-p_y$. Przeczy to przyjętemu przez autora założeniu o wykonywaniu wszystkich synchronizacji z prawdopodobieństwem równym jeden.

Przyjęcie założenia o istnieniu przejścia l_y prowadzi do zaprzeczenia właściwości grafu GPS, zatem niewprost udowodniono, że nie może ono istnieć. Wynika z tego, że grafy G_{i1} do G_{i2} są ze sobą połączone jedynie przejściem synchronizacyjnym l_x. Jego usunięcie powoduje rozpad grafu na dwa nie połączone ze sobą podgrafy.

c.b.d.o.

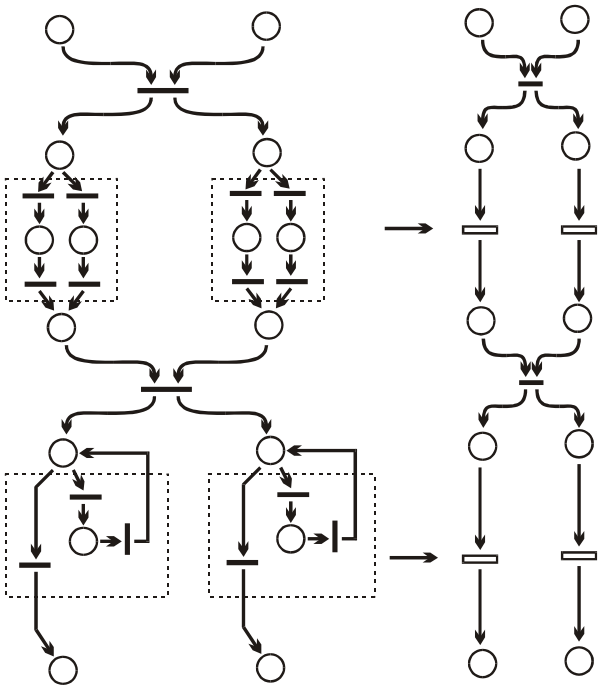
Rozmiar zredukowanego grafu

Analizując rozkłady losowe czasu realizacji programu, niezwykle ważne jest, aby reprezentujący go graf miał rozmiar umożliwiający analityczne lub numeryczne dokonanie obliczeń w akceptowalnym czasie. Oszacowanie rozmiarów uzyskiwanych grafów jest wic krytyczne dla określenia zakresu stosowalności metod opisanych w pracy.

Przy spełnieniu założenia o bezwarunkowym wykonaniu instrukcji synchronizujących procesy, z twierdzenia 2.8 wynika że:

Dla każdego z procesów fragment GPS zawierający się pomiędzy dwoma kolejnymi przejściami symbolizującymi synchronizację jest podgrafem izolowanym o jednym wejściu i jednym wyjściu. Można zatem ten podgraf zredukować do pojedynczego przejścia.

Uwzględniając miejsca znajdujące się pomiędzy przejściami otrzymuje się konstrukcję:



Rys. 2-14. Graf przepływu sterowania i odpowiadający mu graf zredukowany.

W przypadku synchronizacji negatywnej dla każdego procesu istnieją dwa przejścia synchronizacyjne – wejście do sekcji synchronizowanej (krytycznej) i wyjście z niej.

Twierdzenie 2-9

Oznaczając:

PS_i – liczba synchronizacji pozytywnych symetrycznych procesu P_i,

PA_i – liczba synchronizacji pozytywnych asymetrycznych procesu P_i,

NS_i – liczba synchronizacji negatywnych symetrycznych procesu P_i,

NA_i – liczba synchronizacji negatywnych symetrycznych procesu P_i,

SPA – liczba miejsc grafu nie należących do żadnego z procesów, wynikająca ze struktur synchronizacji PA

SNA – liczba miejsc grafu nie należących do żadnego z procesów, wynikająca ze struktur synchr. NA

SNS – liczba miejsc grafu nie należących do żadnego z procesów, wynikająca ze struktur synchronizacji NS

NEGAS – liczba miejsc wynikająca z synchronizacji negatywnych asymetrycznych,

$||M_i||$ - liczność zbioru miejsc grafu Petri opisującego proces P_i,

$||M||$ - całkowita liczba miejsc grafu Petri opisującego program,

liczba miejsc sieci Petriego w zredukowanym grafie przepływu sterowania jest ograniczona następującymi formułami:

$$\|M_i\| \le 2\left(PS_i + PA_i + 2NS_i + 2NA_i\right) + 2 \tag{2-40}$$

$$\|M\| \le \sum_{i:P_i \in PGM} \left(2PS_i + \frac{5}{2}PA_i + \frac{9}{2}NS_i + 5NA_i + 2\right) \tag{2-41}$$

Dowód (2-40):

Z twierdzenia 2.8 wynika że fragment sieci Petriego znajdujący się pomiędzy instrukcjami synchronizacji jest podgrafem izolowanym. Może być, więc zredukowany do pojedynczego przejścia. Ze względu na konieczność jego połączenia z przejściami synchronizacji zastępowany jest konstrukcją miejsce-przejście-miejsce. Po każdym przejściu synchronizacyjnym w zredukowanym grafie znajdują się więc maksymalnie dwa miejsca i następne przejście synchronizacyjne. Liczba miejsc fragmentu sieci Petriego odpowiadającemu pojedynczemu procesowi nie przekracza dwukrotności liczby przejść synchronizacyjnych plus miejsce startowe i końcowe.

$$\|M_i\| \le 2\|L_{sync}\| + 2 \tag{2-42}$$

Instrukcja synchronizacji pozytywnej zastępowana jest pojedynczym przejściem synchronizacji, instrukcja synchronizacji negatywnej dwoma (ustawienie i zwolnienie semafora). Zatem

$\|M_i\| \le 2\left(PS_i + PA_i + 2NS_i + 2NA_i\right) + 2$

c.b.d.o. (2-40)

Dowód (2-41):

Liczba miejsc w całym zredukowanym grafie przepływu sterowania jest sumą liczby miejsc należących do poszczególnych procesów oraz liczby miejsc modelujących semafory (nie należących do procesów).

$$\|M\| \le SPA + SNA + SNS + \sum_{i:P_i \in PGM} 2\left(PS_i + PA_i + 2NS_i + 2NA_i\right) + 2 \tag{2-43}$$

W celu reprezentacji synchronizacji pozytywnej asymetrycznej (komunikat buforowany) konieczne jest umieszczenie w grafie dodatkowego miejsca reprezentującego bufor. Jest to miejsce wspólne dla dwóch procesów uczestniczących w synchronizacji.

$$SPA \le \frac{1}{2} \sum_{i:P_i \in PGM} PA_i \tag{2-44}$$

W celu reprezentacji synchronizacji negatywnej symetrycznej (semafora) konieczne jest umieszczenie w grafie dodatkowego miejsca reprezentującego semafor. Jest to miejsce wspólne dla wszystkich procesów uczestniczących w synchronizacji (niemniej niż dwóch).

$$SNS \le \frac{1}{2} \sum_{i:P_i \in PGM} NS_i \tag{2-45}$$

W celu reprezentacji synchronizacji negatywnej asymetrycznej (priorytetu), w której uczestniczy N procesów konieczne jest umieszczenie w grafie (N-1) miejsc określających priorytet aktualnie wykonującego sekcję procesu. Zatem ilość dodatkowych miejsc wynikających z synchronizacji jest niewiększa niż ilość synchronizacji negatywnych asymetrycznych zawartych w procesach.

$$SNA \le \sum_{i:P_i \in PGM} NA_i \tag{2-46}$$

Zatem:

$$\|M\| \le \frac{1}{2} \sum_{i:P_i \in PGM} PA_i + \frac{1}{2} \sum_{i:P_i \in PGM} NS_i + \sum_{i:P_i \in PGM} NA_i + \sum_{i:P_i \in PGM} \left(2\left(PS_i + PA_i + 2NS_i + 2NA_i\right) + 2\right) \tag{2-47}$$

$$\|M\| \le \sum_{i:P_i \in PGM} \left(2PS_i + \frac{5}{2} PA_i + \frac{9}{2} NS_i + 5NA_i + 2\right)$$

c.b.d.o. (2-41)

Uzyskane przybliżenie pokazuje, że liczba miejsc zredukowanego grafu jest ograniczona. Co więcej ograniczenie górne jest liniową funkcją liczby instrukcji synchronizacji zawartych w procesach wchodzących w skład analizowanego programu.

Acykliczność globalna i lokalna

Z założeń dotyczących jawności synchronizacji, niezmiennej liczby instrukcji synchronizacji oraz statycznego ich powiązania wynika, że dla każdej sekwencji odpaleń przejść GPS prowadzącej do poprawnego stanu końcowego każde z przejść synchronizacyjnych zawartych w GPS jest odpalane jeden i tylko jeden raz. Zatem każdy podgraf G_i grafu GPS taki, że G_i zawiera jedynie miejsca i przejścia opisujące proces P_i posiada następujące właściwości:

- dla każdej sekwencji odpaleń GPS prowadzącej do poprawnego stanu końcowego przejścia synchronizacyjne zawarte w podgrafie G_i są odpalane zawsze w tej samej kolejności.

- dla dowolnych przejść synchronizacyjnych l_a, $l_b \in G_i$ jeżeli istnieje taka sekwencja odpaleń przejść GPS, że przejście l_a jest odpalane wcześniej niż l_b to nie istnieje taka sekwencja odpaleń przejść GPS, że przejście l_b jest odpalanie wcześniej niż l_a.
- dla dowolnego przejścia $l_a \in G_i$ oraz dowolnego przejścia synchronizacyjnego $l_b \in G_i$ jeżeli istnieje taka sekwencja odpaleń przejść GPS, że przejście l_a jest odpalane wcześniej niż l_b to nie istnieje taka sekwencja odpaleń przejść GPS, że przejście l_b jest odpalanie wcześniej niż l_a.
- dla dowolnego przejścia $l_a \in G_i$ oraz dowolnego przejścia synchronizacyjnego $l_b \in G_i$ jeżeli istnieje taka sekwencja odpaleń przejść GPS, że przejście l_a jest odpalane później niż l_b to nie istnieje taka sekwencja odpaleń przejść GPS, że przejście l_b jest odpalanie później niż l_a.
- dla każdej sekwencji odpaleń, jeżeli do miejsca $m_x \in G_i$ wszedł znacznik, a następnie go opuścił to w ramach tej samej sekwencji odpaleń nie pojawi się on powtórnie w miejscu m_x.

Z właściwości acykliczności lokalnej wynikają również właściwości acykliczności globalnej:

- jeżeli istnieje sekwencja zmian stanów grafu taka, że stan U_x występuje przed stanem U_y, to nie istnieje taka sekwencja w której stan U_y występuje przed U_x.
- w obrębie sekwencji zmian stanów U_i sieci GPS żaden ze stanów się nie powtarza

Zredukowany graf przepływu sterowania jest lokalnie acykliczny. W każdym miejscu sieci należącym do któregoś z procesów znacznik może pojawić się tylko raz. Nie dotyczy to miejsc modelujących semafor, które jednak nie należą do żadnego z procesów (należą do systemu operacyjnego). Każde przejście sieci odpalane jest co najwyżej jeden raz, zatem każdemu przejściu można przypisać punkt na osi czasu obrazujący moment jego odpalenia.

Liczba znaczników

Podczas wykonywania sekwencji odpaleń sieci GPS w każdym podgrafie G_i grafu GPS znajduje się jeden i tylko jeden znacznik. Jeżeli modelowany przez G_i proces jeszcze nie rozpoczął działania to znacznik znajduje się w miejscu początkowym procesu, jeżeli natomiast już się zakończył to w końcowym. Dodatkowe znaczniki znajdują się w miejscach GPS modelujących semafory, jak również bufory komunikatów.

Podsumowanie

Graf przepływu sterowania stanowi opis programu współbieżnego, który po redukcji staje się de-facto grafem przepływu synchronizacji pomiędzy procesami. Fragmenty obliczeń dokonywane lokalnie zostają zredukowane do pojedynczych przejść opisanych dystrybuantą rozkładu losowego czasu ich wykonania.

Uzyskana struktura ma rozmiar ograniczony z góry liniową funkcja ilości instrukcji synchronizacji zawartych w procesach. Stanowi więc zwięzły model programu współbieżnego, który można w następnym etapie połączyć z modelem zawodnej maszyny lub też rozwiązać obliczając rozkład losowy czasu realizacji programu.

Zredukowany graf przypomina model BSP [Val90] z występującymi naprzemiennie synchronizacjami i przetwarzaniem lokalnym. W szczególnym przypadku, gdy analizowany program wykorzystuje jedynie synchronizacje pozytywne symetryczne wszystkich procesów (bariery globalne) GPS staje się grafem modelu BSP.

3. Model realizacji programu przez maszynę cyfrową

3.1. Model maszyny cyfrowej

Stosowany w dalszej części pracy model maszyny cyfrowej określają następujące założenia:

- maszynę tworzy jednorodny zbiór jednostek przetwarzających (procesorów z niezbędnym otoczeniem) $G=\{g_1, g_2, ...\}$, z których każda jest zdolna realizować wymagany przez program zestaw instrukcji,
- działanie procesorów jest niezależne, niezależne są też procesy uszkodzeń i napraw,
- liczba procesorów jest równa liczbie procesów zawartych w programie,
- możliwe jest wykonywanie jednego i tylko jednego procesu na każdym procesorze. W przypadku, gdy proces na danym etapie programu nie istnieje, nie rozpatruje się działania procesora go wykonującego,
- nie uwzględnia się migracji procesów pomiędzy procesorami ani wykonywania kilku procesów przez jeden procesor.

Stosowany w dalszej części pracy model procesora określają następujące założenia:

- procesor (wraz z otoczeniem) stanowi element zdolny do realizacji pojedynczego procesu wchodzącego w skład aplikacji współbieżnej,
- procesor ulega uszkodzeniom i naprawom i może znajdować się w jednym z γ stanów sprawności $ssg_1, ...ssg_\gamma$ tworzących zbiór SSG,
- zmiany stanu sprawności procesora określa proces Markowa opisany macierzą Λ intensywności przejść pomiędzy stanami sprawności,
- stany sprawności procesora dzielimy na „sprawny", „częściowo sprawny" i „niesprawny",
- w stanie sprawnym procesor jest zdolny do realizacji kodu programu, w stanie niesprawnym realizacja programu jest wstrzymana,
- stany niesprawne dzielą się na naprawialne (dla których istnieje przejście do stanu sprawnego o niezerowej intensywności) oraz nienaprawialne (dla których nie istnieje),
- szybkość realizacji programu dla poszczególnych stanów sprawności określają współczynniki efektywności $\eta_1, ... \eta_\gamma$ przyjmujące wartości od 0 do 100%. Dla stanu w pełni sprawnego $\eta=100\%$, dla stanu spowolnienia maszyny $0\%<\eta<100\%$.
- dla stanów niesprawnych $\eta=0$,
- czas realizacji kodu w stanie częściowo sprawnym wynosi (t/η) a dystrybuanta rozkładu czasu $\Phi(t/\eta)$
- zmiana stanu sprawności pociąga za sobą dodatkowy narzut czasowy t_R stały lub losowy $\Phi_R(t)$. wymagany dla rekonfiguracji maszyny i ewentualnego powtórzenia części obliczeń.

W celu uproszczenia modelu i obliczeń autor przyjął (typowe w teorii niezawodności) założenia o sporadycznym występowaniu pojedynczych, niezależnych uszkodzeń:

Założenie 3-1

Ryzyko uszkodzenia w czasie realizacji pojedynczego bloku instrukcji jest małe. Średni czas do uszkodzenia jest dużo większy niż czas realizacji bloku.

Założenie 3-2

Prawdopodobieństwo dwóch lub więcej przejść pomiędzy stanami sprawnymi w trakcie realizacji pojedynczego bloku instrukcji jest pomijalnie małe.

Założenie 3-3

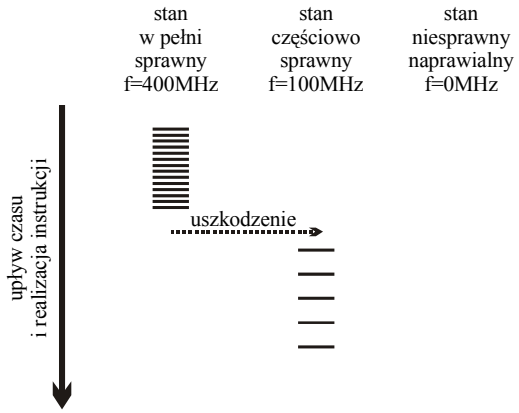
Prawdopodobieństwo przejść pomiędzy stanami uszkodzonymi naprawialnymi jest równe zero. (Brak dalszego uszkadzania niesprawnej maszyny, oraz fragmentarycznych napraw nie umożliwiających dalszej pracy)

Założenie 3-4

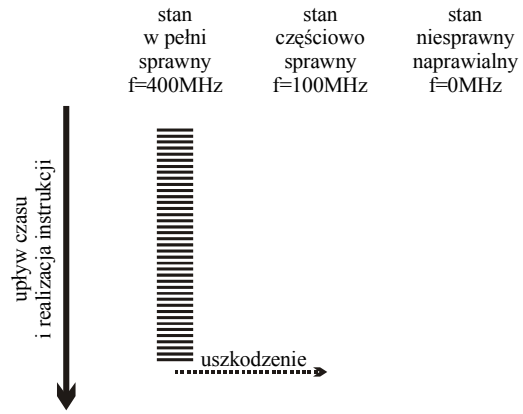
Zakłada się, że blok instrukcji zostanie w całości wykonany w jednym stanie sprawności procesora.

- *Przejścia pomiędzy dwoma stanami sprawnymi dokonują się w momencie kończenia bloku instrukcji.*
- *Przejścia od stanu sprawnego do niesprawnego dokonują się w momencie kończenia bloku instrukcji. Aby możliwa była kontynuacja następuje naprawa w czasie t_R i procesor wraca do stanu sprawnego.*

Przykładowe przejścia pomiędzy stanami sprawności w trakcie wykonywania instrukcji programu przedstawione są na rysunkach 3-1, 3-2 i 3-3. Przykładowy procesor posiadający γ=3 stany sprawności może ulegać częściowemu uszkodzeniu (spowolnieniu) lub całkowitemu zatrzymaniu. Stany sprawności przedstawione są w trzech kolumnach. Czas biegnie z góry do dołu rysunku. Realizowane instrukcje programu zaznaczone są przez poziome kreski. W stanie pełnej sprawności procesor realizuje program z pełną szybkością by w pewnym momencie ulec uszkodzeniu i dalej kontynuować pracę cztery razy wolniej. Zgodnie z założeniem 3-4 przyjmuje się, że przejścia pomiędzy stanami sprawności dokonują się po zakończeniu wykonywania aktualnego bloku instrukcji. Na rys 3-2 przestawiono sytuację równoważną prezentowanej na rys 3-1 lecz z uwzględnieniem wymogu zmian stanu sprawności po zakończeniu aktualnie realizowanego bloku instrukcji



Rys. 3-1. Zmiana stanu sprawności procesora w trakcie realizacji bloku instrukcji



Rys. 3-2. Zmiana stanu sprawności procesora przesunięta na koniec bloku instrukcji

Założenie 3-5

Czas realizacji bloku instrukcji, w trakcie którego nastąpiło przejście z jednego stanu sprawnego do drugiego stanu, jest obliczany na podstawie szybkości procesora w pierwszym ze stanów sprawności.

Przyjęcie upraszczającego założenia 3-5 wnosi pewien błąd do obliczeń czasu realizacji bloku. Jeżeli czas realizacji bloku instrukcji w stanie sprawności γ_1 oznaczymy przez τ_1 , natomiast w stanie sprawności γ_2 przez τ_2 to błąd maksymalny (gdy uszkodzenie wystąpiło na początku bloku) wynosi:

$$\Delta\tau_{\max} = \tau_2 - \tau_1 \tag{3-1}$$

Powołując się na założenie 3-2 można przyjąć, że prawdopodobieństwo zmiany stanu sprawności jest jednakowe w każdym momencie realizacji bloku instrukcji wartość średnia błędu wynosi:

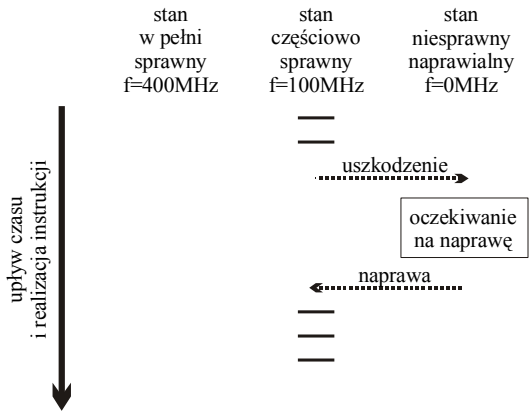
$$\Delta\tau_{\max} = \frac{\tau_2 - \tau_1}{2} \tag{3-2}$$

Znak (dodatni lub ujemy) błędu obliczenia czasu zależy od tego czy procesor zmieniając stan sprawności zwolnił czy przyspieszył swoje działanie.

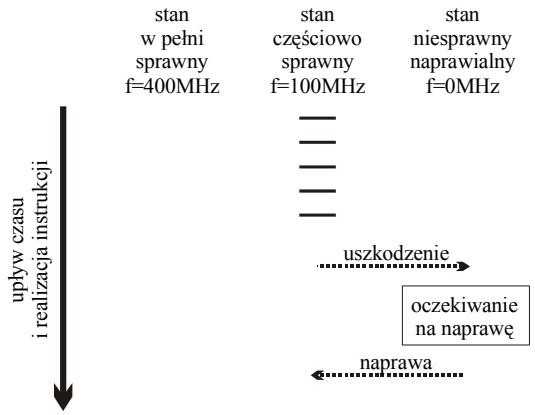
Dla przejść pomiędzy stanem niesprawnym (naprawialnym) a sprawnym przyjmuje się, że czas realizacji bloku instrukcji określa szybkość procesora w stanie sprawnym.

Dla przejść pomiędzy stanem sprawnym a uszkodzonym przyjmuje się, że cały fragment kodu programu zostanie wykonany w stanie sprawnym, a dopiero po jego zakończeniu procesor przejdzie do stanu niesprawnego a następnie odczeka pewien okres czasu t_R wymagany dla dokonania naprawy. Realizacja bloku, podczas której nastąpiło przejście do stanu uszkodzonego (wstrzymanie obliczeń) przedstawiono na rys. 3-3. Wersję równoważną, w której uszkodzenie i naprawa następuje po zakończeniu obliczeń na rys. 3-4.

Przejście pomiędzy stanem uszkodzonym a sprawnym wymaga najpierw określonego czasu dla naprawienia procesora, a dopiero potem kontynuowania obliczeń w stanie sprawnym. Sytuacja ta zobrazowana jest na rys. 3-3. Całkowity czas realizacji przedstawionego bloku jest zatem sumą czasu naprawy oraz czasu realizacji programu w stanie sprawnym.



Rys. 3-3. Dokonanie naprawy a następnie kontynuacja obliczeń



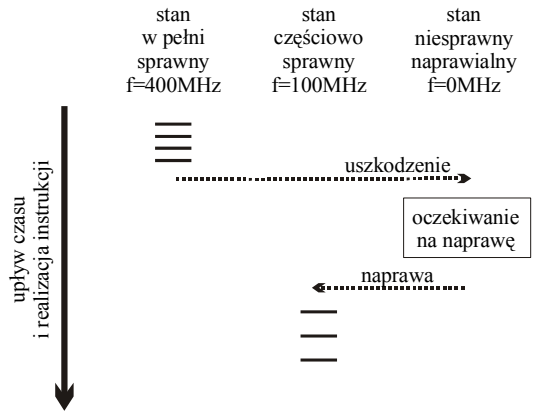
Rys. 3-4. Dokonanie naprawy a następnie kontynuacja obliczeń

Założenie 3-6

Czas realizacji bloku instrukcji, w trakcie którego nastąpiło przejście ze stanu sprawnego do niesprawnego, jest sumą czasu koniecznego do dokonania naprawy oraz realizacji bloku w stanie sprawnym.

Zakładając, że analizowany fragment programu zostanie wykonany do końca - dokonanie naprawy przed zakończeniem realizacji bloku instrukcji jest zdarzeniem pewnym (gdyż inaczej nie można by go ukończyć). Naprawa nie jest więc zjawiskiem losowym, a bezpośrednią konsekwencją uszkodzenia. Nie przeczy zatem założeniu 3-2. Wykonywanie programu w stanie uszkodzonym jest niemożliwe, więc całość obliczeń musi być dokonana w stanie sprawnym (zgodnie z założeniem 3-4).

Założenie 3-6 wprowadza błąd w obliczeniu czasu realizacji bloku, jeżeli w wyniku naprawy nie powrócono do stanu sprawności występującego przed uszkodzeniem. Opisaną sytuację przedstawia rys. 3-5.



Rys. 3-5. Dokonanie naprawy a następnie kontynuacja obliczeń

Wnioski:

- Po wykonaniu bloku instrukcji maszyna zawsze znajduje się w jednym ze stanów sprawnych (po ewentualnej naprawie) lub stanie niesprawnym nienaprawialnym (przerwanie obliczeń).
- Przejścia maszyny do stanu niesprawnego oraz z powrotem do sprawnego zachodzą wewnątrz bloku instrukcji i nie są bezpośrednio obserwowalne przez współpracujące procesory (procesy). Zauważalny jest jedynie dłuższy czas realizacji bloku.
- Traktując stany niesprawności jako „przejściowe" można zredukować liczbę stanów sprawności maszyny, jaka będzie użyta do konstrukcji grafów opisanych w rozdziale 3 i 4 tylko do stanów sprawnych.
- Przejście z stanu sprawnego do sprawnego, jak również pozostanie w stanie sprawnym może odbyć się bezpośrednio, lub poprzez uszkodzenie i naprawę (rys 3-5).

W modelu nie wyklucza się przejść ze stanu niesprawnego naprawialnego do stanu niesprawnego nienaprawialnego (zatrzymanie obliczeń). Nie powoduje to znaczącego skomplikowania obliczeń, zwiększa natomiast istotnie możliwości opisowe modelu.

Wartości liczbowe opisujące modelowaną maszynę to:

- liczba stanów sprawności γ,
- macierz intensywności uszkodzeń i napraw Λ o rozmiarze $\gamma \times \gamma$,
- zestaw współczynników efektywności $\eta_1, \ldots \eta_\gamma$ określających szybkości realizacji programu.

Zakres praktycznej przydatności modelu

Przyjęty model maszyny cyfrowej umożliwia reprezentację uszkodzeń naprawialnych i nienaprawialnych objawiających się spowolnieniem pracy procesora lub jego zatrzymaniem. W modelu przyjęto, że uszkodzenia i naprawy nie powodują utraty przetwarzanych danych i konieczności powtórzenia obliczeń od początku. Konieczność odtworzenia stanu maszyny i powtórzenia fragmentu obliczeń jest uwzględniona w czasie naprawy. Założono, że maszyna posiada pewną redundancję oraz mechanizmy wczesnego wykrywania uszkodzeń zapobiegające propagacji błędów do innych procesów. Dąży się do konstruowania maszyn zdolnych do stopniowego ograniczania swojej funkcjonalności w przypadku uszkodzeń (ang. graceful degradation)

zamiast niespodziewanego i całkowitego zatrzymania. W najprostszym przypadku może to być zmniejszenie częstotliwości zegara przy wykryciu przegrzania lub konieczności oszczędzania energii.

Przykładem uszkodzenia naprawialnego może być chwilowe wstrzymanie pracy na czas rekalibracji dysku (okresowej lub po błędnym odczycie).

Mechanizmy reprezentacji uszkodzeń mogą zostać użyte do przedstawienia zakłóceń realizacji programu wynikających z wykonywanych współbieżnie funkcji systemu operacyjnego.

- Długotrwałe obciążenie procesora równolegle wykonywanym programem (zadaniem systemu operacyjnego lub aplikacją użytkownika) może być modelowane jako przejście do stanu uszkodzonego o mniejszej szybkości procesora.
- Przerwanie systemu operacyjnego może być modelowane jako wejście w stan uszkodzony, podczas którego nie wykonuje się aplikacji, a następnie naprawa czyli powrót z przerwania.

3.2. Stan niezawodnościowo-funkcjonalny procesora

Poza czynnościami uszkodzenia i naprawy procesor zmienia swój stan wykonując program. W pracy założono ścisłe powiązanie procesora z wykonywanym procesem, zatem stan wykonywanego procesu w powiązaniu ze stanem układów procesora opisanych przez architekturę listy rozkazów będzie nazywany stanem funkcjonalnym.

Def. 3-1

Stan funkcjonalny procesora $sp \in SP$ *określa rodzaj aktualnie wykonywanej czynności wchodzącej w skład realizowanego programu (procesu). Jest to stan niezależny od stanu niezawodnościowego, mimo że od stanu niezawodnościowego zależy możliwość zakończenia realizowanej aktualnie instrukcji.*

Stan funkcjonalny [Zam80] realizowanego procesu bywa w literaturze określany również jako „czas logiczny" lub „czas wirtualny". Oznacza on stopień realizacji danego procesu. Dla procesów nie zawierających pętli odpowiadałby on w pewnym stopniu wartości licznika rozkazów wskazującemu aktualnie realizowaną instrukcję. W niniejszej pracy autor pozostanie przy określeniu „stan" natomiast pojęcie czasu będzie używane jedynie w stosunku do „czasu fizycznego" mierzonego w sekundach.

Def. 3-2

Zbiorem stanów niezawodnościowo-funkcjonalnych układu procesor + proces nazywany będzie zbiór $SPG = SSG \times SP$ *będący iloczynem kartezjańskim zbioru stanów niezawodności procesora i zbioru stanów funkcjonalnych procesu.*

Działaniem procesora będzie nazywana każda zmiana stanu niezawodnościowo-funkcjonalnego.

Wzajemna relacja dwóch procesów stochastycznych

W modelu maszyny przyjęto, że zmiany stanu niezawodności opisuje proces Markowa o znanych intensywnościach. Zachodzące równocześnie zmiany stanu funkcjonalnego (postęp obliczeń) opisuje proces semi-Markowa, dla którego czas trwania poszczególnych przejść wynika z grafu przepływu sterowania oraz

szybkości maszyny (opisanej efektywnością η). W najbardziej uproszczonym modelu czasy przebywania w poszczególnych stanach funkcjonalnych są stałe.

W systemach cyfrowych zmiany stanów funkcjonalnych zachodzą kilka rzędów wielkości częściej niż procesy uszkodzeń i napraw, wobec czego przyjęto założenia 3-1, 3-2, 3-3 i 3-4. Wynika z nich, że zmiany stanów procesu semi-Markowa definiują przedziały czasowe, w trakcie których może (ale nie musi) zajść uszkodzenie lub naprawa. Proces Markowa uszkodzeń i napraw odbywa się zatem w dyskretnej przestrzeni czasu (o nierównych odcinkach czasu) określanej przez proces semi-Markowa zmian stanu funkcjonalnego.
W granicznym przypadku proces uszkodzeń i napraw staje się „normalnym" procesem Markowa w ciągłej przestrzeni czasu.

$t_{semM}\ \lambda_{Mar} \rightarrow 0$

gdzie:

t_{semM} – średni czas kroku obliczeń procesu semi-Markowa,

λ_{Mar} – intensywność uszkodzeń i napraw procesu,

Uproszczenia grafu procesu

W podstawowym modelu działania procesora dopuszcza się dwa rodzaje przejść w zbiorze stanów układu procesor + proces:

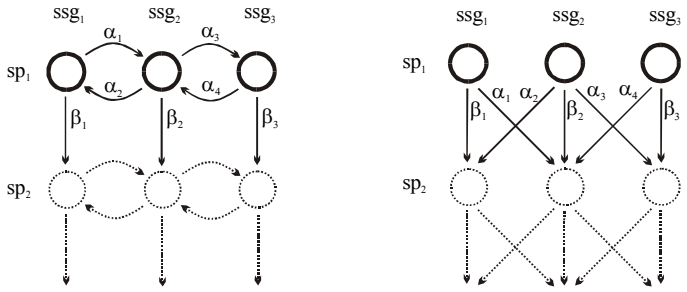
- przejścia opisujące uszkodzenia i naprawy α
- przejścia opisujące postęp pracy użytecznej β.

Liczba możliwych stanów grafu działania procesora jest iloczynem liczności zbiorów SSG i SP. Liczba krawędzi jest iloczynem liczby krawędzi SP oraz kwadratu liczby stanów niezawodności.

Wadą opisanego modelu jest jego cykliczność. Istnienie nieskończonych pętli uszkodzeń i napraw utrudnia analizę grafu. Dzięki przyjęciu założeń 3-1, 3-2, 3-3, 3-4 możliwe jest zaproponowanie modelu działania procesora, w którym przejście z jednego stanu sprawności maszyny do drugiego odbywa się jednocześnie z wykonaniem kolejnego kroku obliczeń. W modelu acyklicznym rozróżniane są dwa rodzaje przejść:

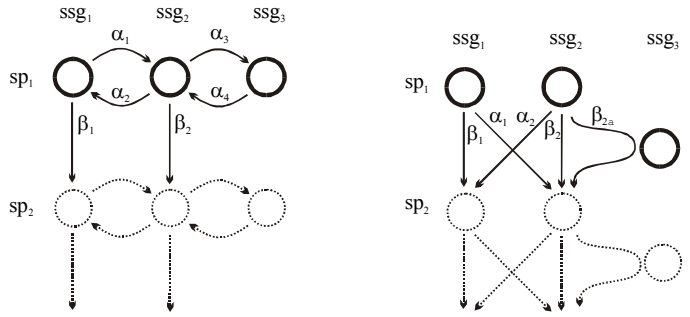
- przejścia β opisujące postęp pracy użytecznej,
- przejścia α opisujące uszkodzenia i naprawy przy jednoczesnym wykonaniu obliczeń.

Przy założeniu jednoczesnej zmiany stanu sprawności procesora i wykonania kroku obliczeń, graf przedstawiający proces uszkodzeń, napraw i dokonywania obliczeń jest acykliczny, co ułatwia jego analizę. Podstawowy graf działania procesora oraz odpowiadający mu graf acykliczny przedstawia rys 3-6.



Rys. 3-6. Podstawowy i acykliczny model zmian stanu procesora i procesu

Jeżeli procesor może przechodzić do stanu niesprawnego, uniemożliwiającego dokonywanie obliczeń, to realizacja danego bloku odbywa się w stanie sprawnym poprzedzającym uszkodzenie. Aby zakończyć realizację bloku (i być gotowym do następnej) procesor musi zostać naprawiony. Przejścia pomiędzy stanami funkcjonalnymi zachodzą jedynie pomiędzy stanami, w których procesor jest sprawny. Przejścia do stanu niesprawności i naprawy są zatem zawarte wewnątrz przejść do kolejnego stanu sprawności.



Rys. 3-7. Podstawowy i acykliczny graf ze stanem niesprawności ssg_3

Określanie parametrów maszyny

Istotnym elementem prezentowanego modelu jest opis maszyny i jej szybkości wykonywania kodu. Parametry komputera można określić na podstawie jego architektury oraz szybkości taktowania lub też zbadać doświadczalnie.

Teoretyczne określenie parametrów maszyny (np. na podstawie modelu VHDL procesora i otoczenia) wymaga posiadania bardzo dokładnego opisu sprzętu i jest niezwykle złożone. Może być ona jednak dokonane jeszcze przed wprowadzeniem maszyny do produkcji.

Doświadczalne określenie parametrów komputera wymaga wykonania szeregu programów testowych badających szybkość realizacji poszczególnych rodzajów instrukcji. Podczas określania szybkości maszyny wyniki pomiarów podlegają zafałszowaniu na skutek przyjęcia zbyt prostego modelu, lub założenia o regularności rozkładu losowego (a zatem o możliwości określenia parametrów na podstawie małej liczby prób). Błędy systematyczne mogą być spowodowane zależnością szybkości od parametrów nieuwzględnianych podczas testów (np. kolejności dostępów do pamięci). Błędy losowe mogą być spowodowane fluktuacjami szybkości maszyny niezależnymi od rodzaju wykonywanego programu. Aby w sposób wiarygodny określić parametry maszyny konieczne jest zatem zbadanie charakteru losowych zmian czasu realizacji i dobranie odpowiedniej metody pomiarowej.

4. Graf realizacji programu

4.1. Struktura grafu

Graf realizacji programu GRP powstaje w wyniku połączenia modelu programu reprezentowanego przez GPS i modelu maszyny. Obrazuje on działanie systemu podczas wykonywania analizowanej aplikacji. Przekształcenie ma na celu uzyskanie grafu, w którym uwzględniono możliwość awarii i napraw maszyny realizującej program, a w szczególności wykonywania kodu ze zmniejszoną prędkością.

Def. 4-1

Graf realizacji programu jest siecią Petriego wraz z funkcją Jt opisywaną przez piątkę uporządkowaną $GRP=<M', L', K', U_0', Jt>$ *na którą składają się:*

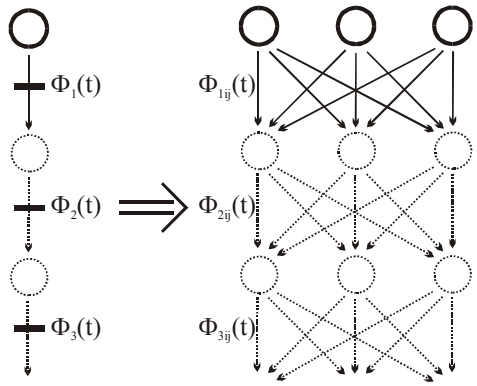
- *zbiór miejsc M',*
- *zbiór przejść L',*
- *zbiór krawędzi* $K' \subseteq M' \times L' \cup L' \times M'$,
- *stan początkowy grafu* U_0 *określający w jakich miejscach znajdują się znaczniki w chwili rozpoczęcia działania sieci,*
- *funkcja Jt przypisująca każdemu przejściu dystrybuantę rozkładu losowego czasu jego wykonania* $Jt(l_i)=\Phi_i(t)$.

Dla grafu realizacji programu utworzonego z GPS=<M, L, K, U_0, Jt> oraz modelu maszyny o γ stanach niezawodnościowych zbiory miejsc, przejść i krawędzi oznaczane przez M', L' i K' są odpowiednio liczniejsze niż odpowiadające im zbiory M, L, K.

- Zbiór miejsc M' o liczności γ razy większej niż zbiór M zredukowanego grafu przepływu sterowania powstaje poprzez zastąpienie każdego z miejsc GPS zbiorem miejsc odpowiadających zbiorowi możliwych stanów procesora.
- Podobnie zbiór przejść L' o liczności γ^2 razy większej niż zbiór L zredukowanego grafu przepływu sterowania. Na skutek zwiększenia zbioru miejsc konieczne jest również zwiększenie zbioru przejść, aby połączyć wszystkie miejsca poprzedniego stanu funkcjonalnego z wszystkimi miejscami następnego stanu funkcjonalnego.
- Zbiór krawędzi $K' \subseteq M' \times L' \cup L' \times M'$ podobnie jak zbiór przejść ma γ^2 razy większą liczność niż odpowiadający mu zbiór w grafie przepływu sterowania.

4.2. Konstruowanie GRP

Na podstawie grafu GPS oraz liczby stanów niezawodnościowych konstruowana jest struktura grafu GRP. Kolejnym etapem jest przypisanie każdemu z przejść prawdopodobieństwa jego wyboru oraz określenie rozkładu losowego czasu jego realizacji.



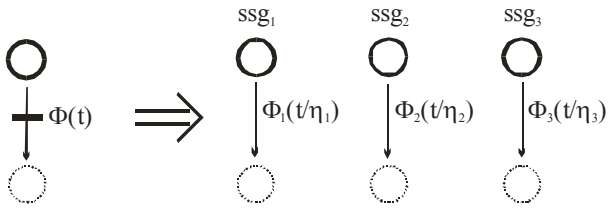
Rys. 4-1. Tworzenie grafu realizacji programu na podstawie grafu przepływu sterowania.

Ze względu na acykliczność zredukowanego grafu przepływu sterowania, jak również uzyskanego na jego podstawie grafu realizacji programu, w każdym miejscu należącym do jakiegoś procesu znacznik pojawia się tylko raz (miejsca obrazujące semafor nie należą do żadnego procesu). Możliwe jest zatem przypisanie do każdego z miejsc rozkładu losowego czasu, jaki musi upłynąć od startu aplikacji do momentu pojawienia się w nim znacznika.

Wzajemna relacja dwóch procesów stochastycznych

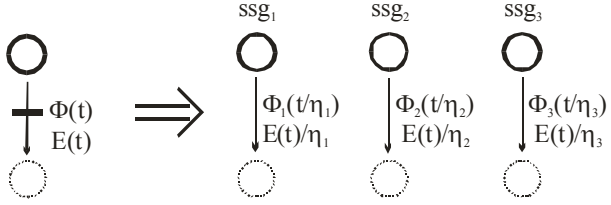
W systemach cyfrowych zmiany stanów funkcjonalnych zachodzą co najmniej kilka rzędów wielkości częściej niż procesy uszkodzeń i napraw. Przyjęto zatem, że proces uszkodzeń i napraw odbywa się w dyskretnej przestrzeni czasu (o nierównych odcinkach czasu) określanej przez proces semi-Markowa zmian stanu funkcjonalnego.

Czas potrzebny do realizacji danego bloku instrukcji opisanego przez przejście sieci Petriego wynika z rozkładu losowego czasu realizacji (opisanego przez graf GPS) oraz stanu sprawności, w jakim znajduje się maszyna.



Rys. 4-2. Rozkłady losowe czasów realizacji w zależności od stanu sprawności maszyny.

Wartości oczekiwane czasów realizacji bloku (zależne od stanu sprawności) definiują przedział czasowy dla procesu Markowa (uszkodzeń i napraw), w jakim może dojść do zmiany stanu niezawodnościowego.



Rys. 4-3. Wartości oczekiwane czasów realizacji w zależności od stanu sprawności maszyny.

Określanie prawdopodobieństw uszkodzeń i napraw

Na podstawie założeń 3-1 i 3-2 przybliżone prawdopodobieństwo zmiany stanu niezawodnościowego jest równe iloczynowi czasu i intensywności przejścia procesu Markowa uszkodzeń i napraw. Dla $\Delta t << 1/\lambda$ można przyjąć przybliżenie:

$$p_{ij} \approx \mathrm{E}(t)\lambda_{ij} \tag{4-1}$$

Prawdopodobieństwo pozostania w dotychczasowym stanie sprawności:

$$p_{ii} = 1 - \sum_{i \neq j} p_{ij} \tag{4-2}$$

Obliczanie dystrybuanty czasu realizacji przejścia

W przypadku pozostania procesora w dotychczasowym stanie sprawności rozkład losowy czasu realizacji przejścia określa dystrybuanta zawarta w GPS rozciągnięta w osi czasu w zależności od współczynnika efektywności (szybkości) dla danego stanu sprawności.

$$\Phi_{ii}(t) = \Phi(\frac{t}{\eta_i}) \tag{4-3}$$

Dla zmian stanu sprawności dochodzi dodatkowo czas niezbędny na rekonfigurację maszyny.

$$\Phi_{ij}(t) = \Phi(\frac{t}{\eta_i}) * \Phi_R(t) \quad \text{dla} \quad i \neq j \tag{4-4}$$

Dla każdego przejścia w grafie realizacji programu obliczone zostaje prawdopodobieństwo jego wyboru oraz rozkład losowy czasu jego realizacji. Na podstawie tych danych oblicza się następnie rozkład losowy czasu realizacji całego programu – czyli osiągnięcia stanu funkcjonalnego oznaczonego jako końcowy.

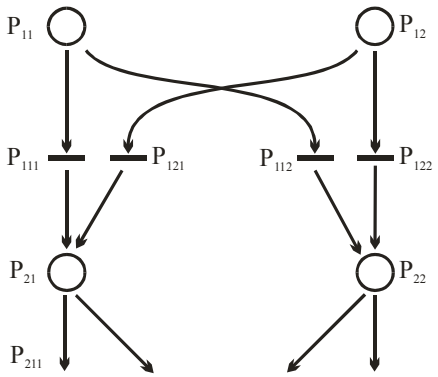
4.3. Analiza wykonania kolejnych bloków programu

Obliczanie rozkładów losowych czasów odpalania poszczególnych przejść dokonywane jest poprzez analizę kolejnych zmian stanu GRP od stanu początkowego aż do końcowego. Dla każdego miejsca w grafie obliczane jest prawdopodobieństwo jego osiągnięcia oraz moment jego osiągnięcia przez znacznik (jako wartość dystrybuanty rozkładu losowego).

Osiągnięcie danego stanu w grafie realizacji programu jest możliwe wieloma wykluczającymi się drogami. Jest ono zatem obliczane z wzoru na prawdopodobieństwo całkowite. Szansa p_{ij} osiągnięcia miejsca m_{ij} do którego z miejsc $m_{(i-1)x}$ dochodzą krawędzie $l_{(i-1)xj}$ określona jest wzorem

$$p_{ij} = \sum_{x=1}^{\gamma} p_{(i-1)x} p_{(i-1)xj} \tag{4-5}$$

Prawdopodobieństwa osiągnięcia miejsc $m_{(i-1)x}$ oznaczane są jako $p_{(i-1)x}$ natomiast prawdopodobieństwa wyboru odpowiedniego przejścia (prowadzącego do m_{ij}) jako $p_{(i-1)xj}$.

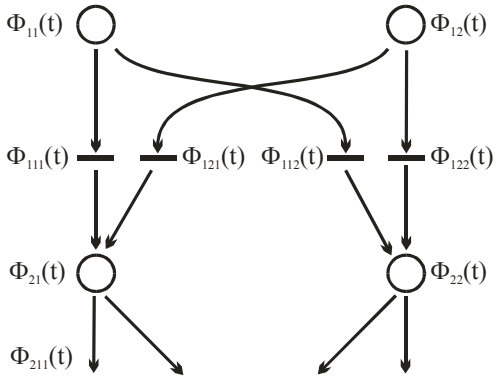


Rys. 4-4. Prawdopodobieństwa osiągnięcia poszczególnych miejsc.

Czas osiągnięcia stanu GRP określany jest na podstawie czasu osiągnięcia poprzedniego stanu oraz czasu przejścia ze stanu poprzedniego do danego. Czas osiągnięcia kolejnego etapu jest więc sumą zmiennych losowych, dystrybuantę jego rozkładu oblicza się za pomocą splotu.

$$\Phi_{ij}(t) = \frac{1}{p_{ij}} \sum_{x=1}^{\gamma} p_{(i-1)x} p_{(i-1)xj} \Phi_{(i-1)x}(t) * \Phi_{(i-1)xj}(t) \tag{4-6}$$

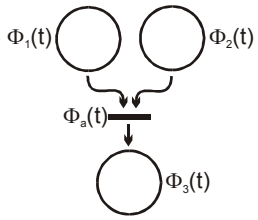
Wagami składników sumowania są prawdopodobieństwa dróg dochodzących, czyli jednoczesnego zajścia dwóch niezależnych zdarzeń losowych – dojścia do wybranego miejsca poprzedzającego $m_{(i-1)x}$ oraz wyboru przejścia $l_{(i-1)xj}$ prowadzącego do miejsca analizowanego m_{ij}. Sumowaniu podlegają dystrybuanty sumy zdarzeń losowych – czasu dojścia do miejsca poprzedzającego oraz czasu wykonania przejścia.



Rys. 4-5. Rozkłady losowe czasu osiągnięcia poszczególnych miejsc.

4.4. Synchronizacja pozytywna

Synchronizacja pozytywna uzależnia możliwość kontynuacji procesu od osiągnięcia przez inny proces odpowiedniego stanu. W grafie Petriego reprezentowana jest jako przejścia z wieloma dochodzącymi krawędziami. Warunkiem koniecznym do rozpoczęcia procesu odpalania przejścia jest obecność znaczników we wszystkich miejscach wejściowych.



Rys. 4-6. Obliczanie czasu synchronizacji

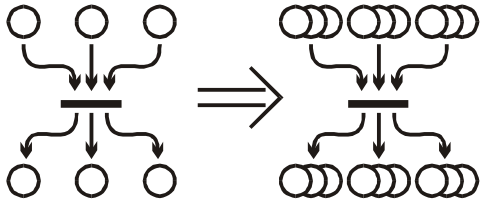
Dystrybuanta $\Phi_{got}(t)$ rozkładu losowego czasu osiągnięcia gotowości odpalenia przejścia l do którego dochodzą krawędzie z miejsc m_1 ... m_n jest iloczynem dystrybuant rozkładów losowych czasów osiągnięć tych miejsc.

$$\Phi_{got}(t) = \prod_{x=1}^{n} \Phi_x(t) \tag{4-7}$$

Czas osiągnięcia miejsca po przejściu synchronizacji jest sumą dwóch zmiennych losowych czasu osiągnięcia gotowości oraz czasu realizacji przejścia. Dystrybuanta rozkładu czasu osiągnięcia miejsca po synchronizacji jest splotem dystrybuant.

$$\Phi_{po\_sync}(t) = \Phi_{got}(t) * \Phi_l(t) \tag{4-8}$$

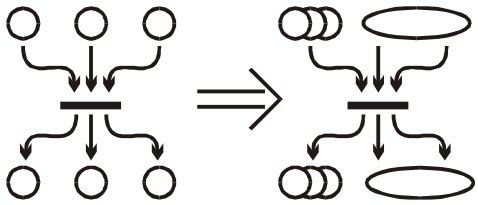
W grafie przepływu sterowania synchronizacja pozytywna jest reprezentowana przez pojedyncze przejście o wielu krawędziach wejściowych i wielu wyjściowych. Podczas konstruowania grafu realizacji programu każde miejsce poprzedzające i następujące po przejściu jest zastępowane przez zbiór miejsc reprezentujących stany sprawności procesora. Przed momentem synchronizacji każdy z procesorów może znajdować się w dowolnym stanie sprawności. W momencie synchronizacji możliwa jest zmiana stanu sprawności każdego z uczestniczących procesorów. Przejścia muszą uwzględnić wszystkie kombinacje stanów wejściowych i wyjściowych. W związku z tym dla procesorów o γ stanach sprawności przejście GPS o A krawędziach wejściowych i B krawędziach wyjściowych musi być zastąpione $\gamma^A\gamma^B$ przejściami w grafie realizacji programu. Prowadzi to do znacznego rozbudowania grafu, zwłaszcza jeżeli zawiera on synchronizację z użyciem barier.



Rys. 4-7. Przejście z wieloma krawędziami w GPS i GRP

Przykład zwiększenia liczby miejsc w przypadku procesora z trzema stanami sprawności przedstawia rys. 4-7. Dla zachowania czytelności przedstawiono tylko jedno przejście zamiast 3^6 przejść uwzględniających wszystkie możliwe kombinacje wyborów miejsc obrazujących stany niezawodności.

Ograniczenie liczby alternatywnych przejść można uzyskać analizując działanie przejścia z punktu widzenia pojedynczego procesu i traktując inne procesy jako pojedynczy, zbiorczy rozkład losowy. Przejściu GPS o A krawędziach wejściowych i B krawędziach wyjściowych odpowiada w takim przypadku $B*\gamma^2$ przejść w grafie realizacji programu.



Rys. 4-8. Analiza przejścia synchronizacyjnego z punktu widzenia pojedynczego procesu

Analiza przejścia synchronizacyjnego z punktu widzenia pojedynczego procesu wymaga uwzględnienia możliwości przejścia z każdego stanu sprawności procesora do dowolnego innego. Podobnie jak w przypadku prostego następstwa instrukcji występuje γ^2 możliwych przejść.

Rozkład losowy czasu osiągnięcia gotowości do synchronizacji przez jeden z procesów jest rozkładem alternatywy zdarzeń wykluczających się. Każde z tych zdarzeń to osiągnięcie stanu funkcjonalnego (x-1) i stanu niezawodnościowego ssg_i.

$$p_{GOT}\Phi_{GOT}(t) = \sum_{i=1}^{\gamma} p_{(x-1)i}\Phi_{(x-1)i}(t) \tag{4-9}$$

Osiągnięcie gotowości do synchronizacji przez wszystkie procesy współpracujące z procesem Pg jest iloczynem zdarzeń składowych – osiągnięcia gotowości przez jeden z procesów. Dystrybuanta zdarzenia losowego oznaczającego gotowość wszystkich procesów za wyjątkiem procesu P_g jest iloczynem dystrybuant. We wzorze (4-9) indeks *g* oznacza numer procesu oczekującego na gotowość wszystkich pozostałych, indeks *k* oznacza kolejny numer procesu uczestniczącego w synchronizacji. Wartości *p* i Φ zależą od procesu, którego dotyczą, jednak aby zachować dotychczasowy sposób indeksowania nie dopisano indeksu *k* do oznaczeń prawdopodobieństw i dystrybuant.

$$p_{EXT\_g}\Phi_{EXT\_g}(t) = \prod_{\substack{k=1 \\ k\neq g}}^{A} p_{GOT\_k}\Phi_{GOT\_k}(t) \tag{4-10}$$

Zatem:

$$p_{EXT\_g}\Phi_{EXT\_g}(t) = \prod_{\substack{k=1 \\ k\neq g}}^{A} \sum_{i=1}^{\gamma} p_{(x-1)i}\Phi_{(x-1)i}(t) \tag{4-11}$$

Mając obliczony rozkład losowy czasu gotowości innych procesów można obliczyć rozkład czasu osiągnięcia każdego z miejsc po synchronizacji. Każda droga prowadząca do pojawienia się znacznika w miejscu m_{xy} (stan funkcjonalny *x*, stan niezawodnościowy *y*) składa się z dwóch etapów:

- osiągnięcia gotowości do odpalenia przejścia $l_{(x-1)iy}$ czyli jednoczesnego:
  - osiągnięcia przejścia $m_{(x-1)i}$ opisanego przez $p_{(x-1)i}\Phi_{(x-1)i}(t)$
  - osiągnięcia gotowości prze inne procesy $p_{EXT\_g}\Phi_{EXT\_g}(t)$
- losowego czasu odpalenia przejścia $l_{(x-1)iy}$ opisanego $p_{(x-1)iy}\Phi_{(x-1)iy}(t)$

Dla pojedynczej drogi przez przejście $l_{(x-1)iy}$ mamy zatem:

$$p_{(x-1)iy}\Phi_{(x-1)iy}(t) = \left(p_{EXT\_g}\Phi_{EXT\_g}(t)\, p_{(x-1)i}\Phi_{(x-1)i}(t)\right) * \left(p_{(x-1)iy}\Phi_{(x-1)iy}(t)\right) \tag{4-12}$$

Osiągnięcie miejsca jest alternatywą wykluczających się zdarzeń losowych opisanych równaniem (4-12). Zatem prawdopodobieństwo (dystrybuantę) określa wzór na prawdopodobieństwo całkowite (wykluczających się ścieżek).

$$p_{xy}\Phi_{xy}(T) = \sum_{i=1}^{\gamma}\left(\left(p_{EXT\_g}\Phi_{EXT\_g}(t)\,p_{(x-1)i}\Phi_{(x-1)i}(t)\right) * \left(p_{(x-1)iy}\Phi_{(x-1)iy}(t)\right)\right) \tag{4-13}$$

Po wyciągnięciu wartości stałych przed znak sumy otrzymujemy

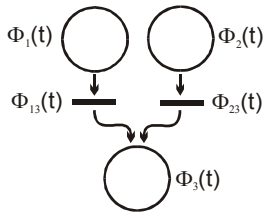
$$\Phi_{xy}(t) = \frac{1}{p_{xy}}\,p_{EXT\_g}\Phi_{EXT\_g}(t)\sum_{i=1}^{\gamma}\left(p_{(x-1)i}\,p_{(x-1)iy}\Phi_{(x-1)i}(t) * \Phi_{(x-1)iy}(t)\right) \tag{4-14}$$

Zatem:

$$\Phi_{xy}(t) = \frac{1}{p_{xy}}\prod_{\substack{k=1\\k\neq g}}^{A}\ \sum_{i=1}^{\gamma} p_{(x-1)i}\Phi_{(x-1)i}(t)\ \sum_{i=1}^{\gamma}\left(p_{(x-1)i}\,p_{(x-1)iy}\Phi_{(x-1)i}(t) * \Phi_{(x-1)iy}(t)\right) \tag{4-15}$$

4.5. Alternatywa

W systemach z rezerwą może istnieć sytuacja, w której warunkiem wystarczającym kontynuacji obliczeń jest dotarcie do oznaczonego punktu któregokolwiek z procesów z podanego zbioru. Czas dotarcia znacznika do miejsca dopuszczającego alternatywne drogi jest zatem równy najmniejszemu z wylosowanych elementów składowych. Z właściwości zmiennych losowych wynika, że dla alternatywy doświadczeń losowych dopełnienie dystrybuanty rozkładu wynikowego jest iloczynem dopełnień dystrybuant składowych.



Rys. 4-9. Fragment grafu modelujący alternatywne drogi osiągnięcia miejsca

Dystrybuanta rozkładu losowego czasu osiągnięcia miejsca m_x poprzez jedną z *n* alternatywnych dróg wyraża się wzorem

$$\Phi_x(t) = 1 - \prod_{i=1}^{n}\left(1 - \Phi_{drogi\_i}(t)\right) \tag{4-16}$$

Czas realizacji drogi jest sumą dwóch zmiennych losowych – czasu osiągnięcia poprzedzającego miejsca oraz czasu odpalania przejścia. Dystrybuanta rozkładu losowego czasu realizacji drogi jest zatem splotem dwóch dystrybuant.

$$\Phi_x(t) = 1 - \prod_{i=1}^{n}\left(1 - \left(\Phi_i(t) * \Phi_{ix}(t)\right)\right) \tag{4-17}$$

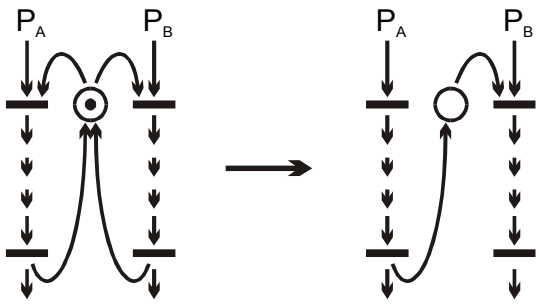
Obecność w sieci fragmentu reprezentującego alternatywę wymaga dodania przejść pochłaniających nadmiarowe znaczniki. Możliwe jest bowiem pojawienie się więcej niż jednego znacznika w miejscu do którego dochodzi wiele krawędzi. Można przyjąć, że w momencie zaistnienia wystarczającego warunku (dojścia pierwszego znacznika) dalsze przyjmowanie znaczników przez przejście zostaje zablokowane, czyli alternatywne procesy (rezerwowe) są wyłączane.

Przy analizie sieci konieczne jest odróżnienie alternatywy wykorzystującej nadmiar procesów (w celu zapewnienia niezawodności) od instrukcji wyboru, gdzie tylko w jednej z alternatywnych ścieżek może pojawić się znacznik.

4.6. Modelowanie synchronizacji negatywnej

Struktura i sposób działania fragmentu modelującego synchronizację negatywną wymaga odmiennego podejścia niż pozostałe fragmenty grafu GPS i GRP. Obszar synchronizacji negatywnej tworzy pętle w grafie uniemożliwiając prostą analizę krok po kroku od miejsc wejściowych do końcowych. Miejsce symbolizujące semafor posiada krawędzie konkurujące o znacznik w sposób dynamiczny, w przeciwieństwie do innych miejsc gdzie wartość prawdopodobieństwa wyboru drogi jest z góry ustalona. Do miejsca modelującego semafor znacznik może wejść kilkakrotnie, podczas gdy do innych miejsc jednokrotnie.

Analiza grafu zawierającego synchronizację negatywną polega na zamianie na równoważny zestaw grafów zawierających synchronizację pozytywną. Jeżeli znana jest kolejność, w jakiej procesy pobierają znacznik z miejsca modelującego semafor to synchronizację negatywną można zamienić na pozytywną asymetryczną, w której procesy wzajemnie przekazują sobie komunikat „skończyłem, możesz zaczynać". Poniższy rysunek ilustruje przekształcenie, jeżeli wiadomo, że proces P_A zawsze wchodzi jako pierwszy do obszaru synchronizowanego.

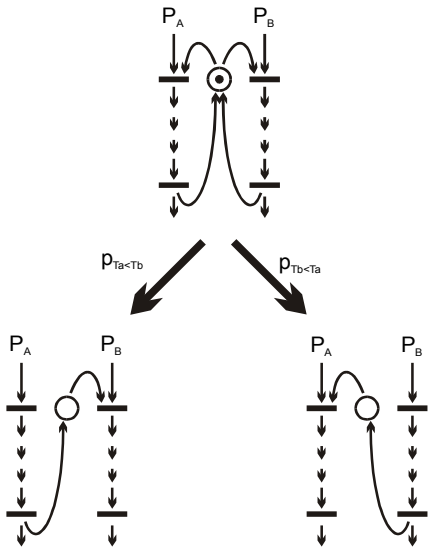


Rys. 4-10. Przekształcenie synchronizacji negatywnej w graf acykliczny

Jeżeli nie jest ustalona kolejność, w jakiej procesy wchodzą do obszaru synchronizowanego konieczna jest analiza wszystkich możliwych wariantów. Dla N konkurujących procesów jest N! możliwych kolejności w jakiej wchodzą do sekcji krytycznej. Dla każdego uporządkowania procesów oblicza się jego prawdopodobieństwo, konstruuje odpowiedni graf równoważny oraz oblicza dystrybuantę rozkładu osiągnięcia końca sekcji. W wyniku uśrednienia rozkładów dla każdej możliwej kolejności wchodzenia procesów w obszar synchronizowany uzyskiwany jest rozkład wypadkowy czasu realizacji sekcji krytycznej. Przy uśrednianiu rozkładów prawdopodobieństwo wyboru każdego wariantu pełni rolę wagi.

Warunkowe przekształcenie grafu

Analizując graf przy założeniu, że proces P_i wszedł jako pierwszy do sekcji krytycznej, można przekształcić graf na równoważny. W nowym grafie proces P_i po zakończeniu sekcji krytycznej wysyła komunikat odblokowujący inne czekające procesy. Jeżeli więcej niż jeden proces dalej oczekuje na wejście to analizę wyboru drogi przeprowadza się dla zbioru pozostałych procesów.



Rys. 4-11. Warunkowe przekształcenie semafora na dwa grafy

Na rys 4-11 przedstawiony został warunkowy podział synchronizacji negatywnej na dwa acykliczne grafy. O wadze poszczególnych wariantów, a co za tym idzie o ich udziale w późniejszym uśrednianiu wyników decyduje prawdopodobieństwo, z jakim każdy z nich może zostać wybrany. O prawdopodobieństwie tym decydują zaś rozkłady losowe czasów osiągnięcia gotowości do wejścia do sekcji krytycznej dla procesów P_A i P_B.

Rozkład czasu wykonania sekcji krytycznej

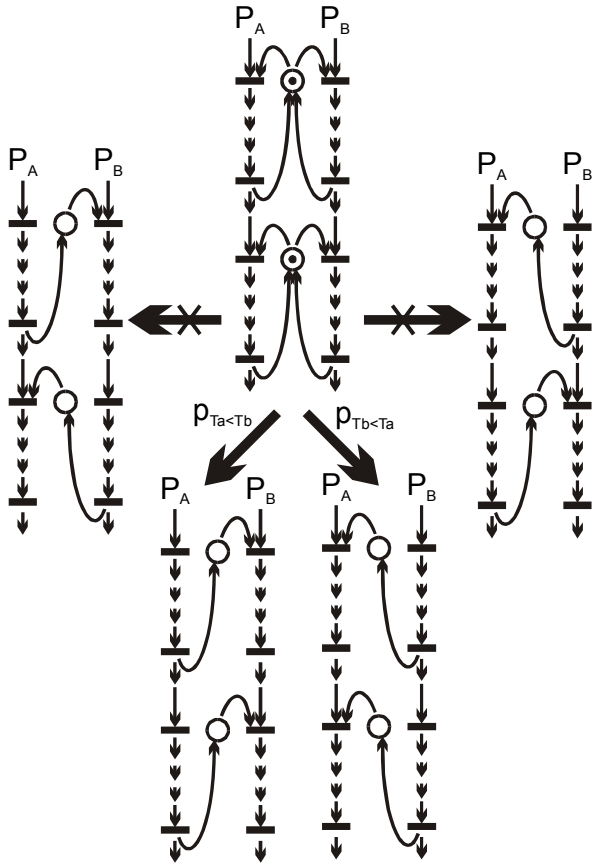
Po obliczeniu rozkładów czasu osiągnięcia końca obszaru synchronizowanego dla poszczególnych kombinacji kolejności wchodzenia procesów oblicza się z wzoru na prawdopodobieństwo całkowite rozkład czasu wykonania sekcji krytycznej. Uzyskane w ten sposób dystrybuanty dla procesów P_A i P_B nie zawierają informacji o wzajemnej korelacji, (jeżeli proces P_i był szybszy to P_j będzie wolniejszy), jednak upraszcza to dalsze obliczenia, gdyż dla przejść sieci znajdujących się po sekcji krytycznej zdefiniowany będzie tylko jeden rozkład czasu wykonania. Jeżeli po synchronizacji negatywnej następuje kilkakrotna pozytywna synchronizacja, to ze względu na jej uśredniające działanie niedokładności związane z łączeniem wyników są pomijalne.

Synchronizacja negatywna może przyczyniać się do zwiększenia różnicy czasów wykonywania współpracujących procesów. Mała różnica czasu gotowości do wejścia może zmienić kolejność wchodzenia procesów do obszaru synchronizowanego, a co za tym idzie, spowodować duże różnice szybkości realizacji uczestniczących procesów. Dla rozkładów czasu dotarcia do semafora o gęstościach skupionych wokół jednego maksimum i posiadających podobne wartości oczekiwane, rozkład wypadkowy czasu wykonania sekcji krytycznej może posiadać N! maksimów gęstości związanych z różną możliwą kolejnością wchodzenia w sekcję krytyczną. Pomiędzy rozkładami czasów dla poszczególnych procesów istnieje silna korelacja. Jeżeli proces A wszedł jako pierwszy, to jego czas realizacji będzie krótszy, ale za to procesu B będzie dłuższy. Uśrednianie rozkładu gubi informację o wzajemnej korelacji czasów wykonywania procesów.

Silna korelacja kolejnych synchronizacji negatywnych

Większość programów jest pisana w taki sposób, aby czas przebywania w sekcji krytycznej był jak najmniejszy. Z reguły przyjmuje się, że jest on zdecydowanie krótszy niż czas wykonywania czynności poza

obszarem synchronizowanym. W wyniku wykonywania lokalnych zadań przez każdy z procesorów następuje losowa zmiana wzajemnej szybkości procesów i zmniejszenie wzajemnej korelacji wynikającej z przebycia obszaru synchronizacji negatywnej. Jeżeli po jednej sekcji krytycznej występuje w krótkim czasie następna, to wzajemna korelacja szybkości procesów może mieć znaczenie dla ustalenia porządku, w jakim procesy wejdą do drugiego obszaru synchronizacji. Proces, który jako pierwszy wszedł do pierwszej sekcji krytycznej zakończy ją zdecydowanie szybciej niż drugi, a co za tym idzie najprawdopodobniej również jako pierwszy wejdzie do drugiej sekcji. Zatem drugi semafor należałoby rozważać w połączeniu z pierwszym. Na rys 4-12 przedstawiony został fragment sieci zawierający dwa semafory. Teoretycznie istnieją cztery przypadki kolejności wykonywania sekcji krytycznych, jednak dwa z nich są bardzo mało prawdopodobne.



Rys. 4-12. Korelacja pomiędzy kolejnymi semaforami

Oddzielne rozpatrywanie rozkładów warunkowych

Analizując program zawierający dużą liczbę synchronizacji negatywnych przedzielonych małymi obszarami kodu niezsynchronizowanego nie przeprowadza się uśredniania czasu realizacji każdej sekcji. Rozpatruje się natomiast wpływ kolejności zakończenia sekcji na kolejność rozpoczęcia następnej. Dokonywana w ten sposób eliminacja najmniej prawdopodobnych przypadków umożliwia dokonanie analizy grafu przy akceptowalnym stopniu złożoności obliczeniowej. W przykładzie pokazanym na rys 4-12, dzięki eliminacji konieczna jest analiza dwóch przypadków zamiast czterech.

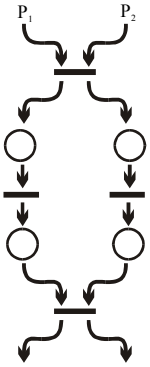
4.7. Korelacja pomiędzy czasami realizacji procesów

Przedstawione powyżej metody obliczania rozkładu losowego czasu synchronizacji procesów zakładały znajomość rozkładów czasów realizacji procesów składowych do momentu synchronizacji oraz niezależność

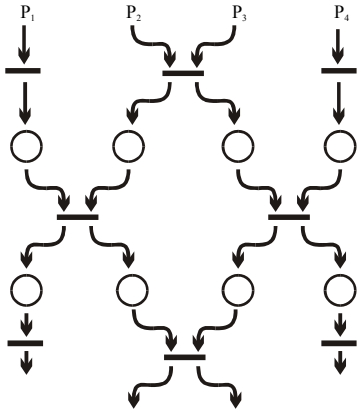
tych czasów. W rzeczywistym systemie istnieje korelacja pomiędzy czasami realizacji poszczególnych procesów wynikająca z wcześniejszych synchronizacji. Korelacja może być zarówno dodatnia jak i ujemna.

Synchronizacja pozytywna powoduje powstanie dodatniej korelacji czasów realizacji procesów, gdyż dłuższy czas wykonywania pewnego procesu wymusza wydłużenie czasu realizacji powiązanych procesów.

Synchronizacja negatywna może powodować zarówno ujemną (w przypadku współzawodnictwa) jak i dodatnią korelację (przy naprzemiennym wchodzeniu do sekcji).



Rys. 4-13. Korelacja czasu realizacji procesów na skutek poprzedniej synchronizacji



Rys. 4-14. Korelacja czasu realizacji wielu procesów

Prowadząc obliczenia stosuje się jedną z następujących metod:

- Zakłada się brak korelacji. Założenie to jest bliskie prawdy, gdy synchronizacja pomiędzy procesami następuje rzadko i jest przedzielona długimi czasami pracy samodzielnej. [Mag91]
- Dąży się do oszacowania współczynnika korelacji na podstawie analizy struktury powiązań poprzedzających elementów grafu. W przypadku, gdy dla wszystkich uczestniczących procesów poprzednia synchronizacja jest wspólną synchronizacją pozytywną (barierą) możliwe jest przedstawienie czasu realizacji każdego z procesów jako sumy czasu wspólnego i niezależnego. W przypadku przeplecionych synchronizacji wielu procesów (rys. 4-14) dokładne określenie stopnia korelacji jest zadaniem bardzo trudnym obliczeniowo.
- Oblicza się rozkłady losowe dla skrajnych możliwych wartości współczynnika korelacji otrzymując górne i dolne oszacowanie czasu realizacji [Kam85], [Kam85c], [Kam87].

Ze względu na duże rozmiary grafów GPS i GPR aby zachować rozsądny poziom komplikacji modelu i obliczeń w pracy przyjęto założenie o braku korelacji pomiędzy czasami realizacji procesów.

W rozdziale czwartym przedstawiono metodę umożliwiającą połączenie grafu przepływu sterowania opisującego program z modelem maszyny cyfrowej zdolnej go realizować. Zaprezentowano również w jaki sposób na podstawie grafu uzyskać rozkład losowy czasu realizacji całej aplikacji. W trakcie konstruowania modelu programu oraz maszyny poczyniono szereg założeń, upraszczając naturę procesu stochastycznego realizacji programu. Aby uznać model za kompletny, Konieczne jest doświadczalne sprawdzenie prawdziwości założeń, dzięki którym osiągnięto ograniczenie stopnia komplikacji grafu.

5. Doświadczalna weryfikacja założeń modelu

W celu obliczenia rozkładów losowych czasu realizacji zadania przez maszynę współbieżną zaproponowano uproszczone modele sprzętu oraz programu. Wiązało się to z koniecznością przyjęcia założeń, które trudno udowodnić formalnie. Aby upewnić się, co do usłużności przyjętych uproszczeń wykonano liczne eksperymenty.

W szczególności zbadano:

- Celowość traktowania czasu realizacji (fragmentu) kodu w kategoriach zjawisk losowych
- Kształt i właściwości rozkładu losowego czasu realizacji fragmentu programu
- Kształt i właściwości rozkładu losowego czasu komunikacji (i synchronizacji)
- Możliwość przybliżania kształtu rozkładu prostą funkcją analityczną
- Zgodność przybliżonych obliczeń analitycznych i numerycznych
- Niezależność czasów realizacji kolejnych bloków programu
- Niezależność czasów realizacji równoległych bloków programu
- Charakter i właściwości stochastyczne opóźnień generowanych przez system operacyjny

5.1. Zmienność losowa czasu realizacji fragmentu kodu

Tworząc model maszyny i wykonywanego przez nią programu przyjęto założenie, że zarówno liczba (i rodzaj) instrukcji, jak też czas ich realizacji są zjawiskami losowymi. Aby to potwierdzić należy przeprowadzić serię pomiarów czasu realizacji programu i określić stopień zmienności mierzonego parametru. W przypadku gdyby odchylenie standardowe okazało się znacząco mniejsze od wartości średniej należałoby raczej przyjąć przybliżenie, że czas realizacji jest wartością stałą. Jeżeli natomiast rozrzut poszczególnych wyników okaże się znaczny, to potwierdzona zostanie celowość użycia metod probabilistycznych.

Przeprowadzono pomiary czasu realizacji bloków programu o różnej długości (od kilku do kilku milionów instrukcji) dla każdego z nich określając empiryczny rozkład losowy. Przebadano również czasy przepływu danych przez sieć oraz działanie procesów współbieżnych. Dla otrzymanych rozkładów losowych obliczono statystyki opisowe aby na ich podstawie określić charakter rozkładu. Wyniki doświadczeń wykonanych w celu określenia zmienności czasu realizacji bloków programu zawarte są w dodatku B. Przedstawione są również badania czasu komunikacji i synchronizacji procesów.

Rozkłady losowe czasu wykonywania fragmentu programu mają następujące właściwości:

- Wartości średniej, mediany i mody są bardzo bliskie wartości minimalnej. Świadczy to o skupieniu centrum rozkładu w pobliżu dolnego ograniczenia zbioru wylosowanych wartości.
- Współczynnik zmienności rozkładu (stosunek odchylenia std. do średniej) przyjmuje wartości od 6,41 do 0,023 i maleje wraz ze wzrostem długości analizowanego fragmentu programu. Tendencja spadkowa jest jednak słabsza niż $1/\sqrt{średnia}$, jakiej należy oczekiwać od doświadczenia będącego sumą wielu niezależnych doświadczeń składowych.

- Dodatnia, wysoka wartość skośności świadczy, iż wykres funkcji gęstości jest szybko zbieżny lewostronnie do zera a wolno prawostronnie. Wartości skośności przekraczające 7 świadczą o bardzo dużej asymetrii wykresu.
- Wartości średniej wewnętrznej (z 98% próbek) są mniejsze od średniej z wszystkich próbek, co potwierdza dodatnią skośność rozkładu.
- Rozstęp kwartyli jest dla wszystkich przypadków zdecydowanie mniejszy (co najmniej 3 razy) od odchylenia standardowego. Stosunek ten świadczy o skupieniu obserwowanych wartości w wąskim przedziale oraz istnieniu wartości zdecydowanie odbiegających od średniej wpływających na wysoką wartość odchylenia standardowego. Wykres posiada wyraźne, skupione maksimum.
- Wartość kurtozy jest dodatnia i przybiera wysokie wartości.

Rozkłady losowe czasu transmisji ramki w sieci lokalnej mają kształt zbliżony do rozkładów czasów wykonywania obliczeń. Dla długich odcinków sieci internet (np. Wrocław – Kopenhaga) rozkład przybiera kształt zbliżony do normalnego.

Przedstawione właściwości rozkładów umożliwiają wyciągnięcie następujących wniosków:

- Rozkłady losowe czasu realizacji krótkich i długich fragmentów kodu mają takie same właściwości. Wydaje się zatem możliwe stworzenie jednolitego opisu umożliwiającego reprezentację rozkładu losowego czasu wykonywania procedur o różnych długościach.
- Kształt rozkładu losowego czasu realizacji fragmentu kodu zdecydowanie różni się od rozkładu normalnego.
- Rozkłady losowe czasów realizacji długich fragmentów kodu (ponad 10^7 instrukcji) mają wyraźnie asymetryczny kształt mimo, że na podstawie centralnego twierdzenia granicznego należałoby oczekiwać kształtu zbliżonego do normalnego.

5.2. Obliczenia analityczne i numeryczne wypadkowych rozkładów losowych

W celu uzyskania rozkładu losowego czasu realizacji całego programu niezbędne jest wielokrotne obliczanie rozkładu sumy zmiennych losowych oraz maksimum zmiennych losowych. Ze względu na konieczność stosowania uproszczeń i przybliżeń szczególnego znaczenia nabiera określenie kształtu i parametrów przetwarzanych rozkładów losowych. W szczególności:

- Czy możliwe jest wiarygodne przybliżenie kształtu rozkładu prostą funkcją, dla której można analitycznie obliczyć wartość splotu ?
- Czy możliwe jest zapisanie rozkładu losowego w postaci kilku parametrów liczbowych, czy też konieczne jest dokładne opisanie kształtu rozkładu – punkt po punkcie ?
- Czy możliwe jest modelowanie opóźnień powodowanych przez system operacyjny i przewidywanie kształtu rozkładu losowego czasu realizacji fragmentu programu ?
- Czy jest możliwe przybliżanie splotu dystrybuant (gęstości) rozkładów inną, prostszą funkcją ?

Opis doświadczeń wykonanych w celu określenia charakteru rozkładów losowych czasów realizacji poszczególnych bloków programu, oraz uzyskane wyniki zawarte są w dodatku B. Przedstawione są tam też próby dopasowania typowych rozkładów do otrzymanych wyników.

Na podstawie statystyk opisowych postanowiono sprawdzić w jakim stopniu empiryczny kształt rozkładów losowych jest podobny do kształtów najpopularniejszych rozkładów losowych. Niestety, uzyskane rozkłady losowe nie mają właściwości zbliżonych do rozkładu normalnego ani wykładniczego.

Przy poszukiwaniu funkcji mogącej stanowić dobre przybliżenie gęstości rozkładu czasu realizacji istotne są następujące kryteria:

- Dla stosunkowo dużego odcinka wartości czasu od zera do pewnej wartości minimalnej prawdopodobieństwo jest równe zero. Pod uwagę brane są więc jedynie takie funkcje gęstości rozkładu, które umożliwiają zdefiniowanie wartości minimalnej możliwych do wylosowania wartości.
- Dla krótkich bloków wartość mody pokrywa się z kresem dolnym przyjmowanych wartości.
- Dla dłuższych fragmentów wartość minimalna rozkładów jest zbliżona do mody, jednak od niej mniejsza. Funkcja przybliżająca rozkład powinna zatem posiadać maksimum dla wartości wyższych niż kres dolny wartości o niezerowym prawdopodobieństwie.
- Nie stwierdzono wyraźnej zależności pomiędzy wartością średnią a odchyleniem standardowym. Sugeruje to, że poszukiwana funkcja powinna mieć oddzielne parametry określające położenie i zmienność rozkładu.
- Rozstęp kwartyli jest mały w porównaniu do odchylenia standardowego. Maksimum gęstości, jest zatem bardzo „wysokie" i szpiczaste.
- Kurtoza obliczona dla próbek zebranych doświadczalnie jest (bardzo) duża.
- Wysoka zmierzona wartość skośności i kurtozy świadczy o wolnej zbieżności prawostronnej do zera funkcji gęstości rozkładu, znacznie wolniejszej niż zbieżność wykładnicza.
- Zaobserwowano, że dla zebranych próbek wartość kurtozy podzielona przez kwadrat skośności przyjmuje wartości z przedziału 1,024 do 1,56 mimo, że sama skośność zmienia się w zdecydowanie szerszym zakresie 7,73 do 68,79. Może to sugerować, że poszukiwana funkcja może posiadać jeden (a nie dwa) parametry określające kształt.
- Duże zmierzone wartości wariancji i wyższych momentów, oraz brak powtarzalności ich pomiarów sugerują użycie w modelu rozkładów losowych o nieskończonej wariancji (np. rozkład Pareto).
- Duży wpływ na parametry rozkładu mają wyniki o wartościach zdecydowanie przekraczających średnią. Mają one tendencję do grupowania się w dodatkowych punktach skupienia. W większości przypadków dodatkowe maksima gęstości są wyraźne i łatwe do wyodrębnienia. Funkcja (lub model) przybliżająca kształt rozkładu musi zatem uwzględniać dodatkowe punkty skupienia.

Badając właściwości rozkładów określono jednocześnie jak wiele doświadczeń jest potrzebnych dla wiarygodnego zmierzenia parametrów statystycznych czasu realizacji bloku. Porównano w tym celu wyniki 100, 1 tys. i 10 tys. pomiarów dochodząc do następujących wniosków:

- Dla wiarygodnego określenia wariancji rozkładu czasu realizacji fragmentu kodu konieczne jest przeprowadzenie bardzo dużej ilości doświadczeń. Wykonanie 10 tys. prób okazuje się być niewystarczające. Wcześniej wykazano, że na wartość wariancji bardzo duży wpływ mają wartości pomiarów znajdujące się w dodatkowych punktach skupienia. Ponieważ prawdopodobieństwo ich wystąpienia jest małe, to uzyskanie wiarygodnej wartości wariancji jest trudne. Szczególnie jest to widoczne przy 100 pomiarach, gdzie nie zaobserwowano opóźnienia o bardzo dużej wartości, więc otrzymano zdecydowanie mniejszą wartość odchylenia standardowego niż dla 10 tys. pomiarów.

- Pośrednie metody obliczania wariancji okazują się być dokładniejsze niż liczenie według definicji na podstawie pomiarów. W przypadku przybliżania kształtu rozkładu znaną funkcją w większości wypadków istnieje relacja pomiędzy wariancją a innymi parametrami opisowymi rozkładu, które można określić z mniejszym błędem.
- Wartości minimum i mediany można określić z dużym poziomem ufności przy małej liczbie prób.
- Mimo że dla skończonej próbki uzyskuje się skończone wartości wariancji, uzyskane wyniki sugerują, że rozkład losowy czasu realizacji programu posiada nieskończoną wartość drugiego momentu.

W celu weryfikacji dokładności użytych metod pomiarowych przeprowadzono test chi-kwadrat dla czterech doświadczeń przeprowadzonych na tym samym komputerze, przy zachowaniu możliwie identycznych warunków pomiaru. Dla liczby przedziałów 10-20 dobranych według kryterium porównywalnej liczności test wykazał zdecydowaną niezgodność uzyskanych rozkładów. Jest to sprzeczne z oczekiwaniami, gdyż wszystkie doświadczenia przeprowadzano w jednakowych warunkach. Potwierdza to wcześniejsze obserwacje, gdzie zauważono wpływ niemożliwych do wyeliminowania losowych czynników wpływających na kształt i parametry rozkładu losowego czasu realizacji kodu.

Możliwe również, że testy zgodności rozkładów nie dają pozytywnego wyniku, gdyż rozkład losowy czasu realizacji programu nie spełnia założeń o skończonej wartości drugiego momentu.

5.3. Niezależność czasów realizacji bloków programu

Tworząc model wykonywania programu założono, że czas realizacji kodu pomiędzy kolejnymi instrukcjami synchronizacji może być określany w oderwaniu od reszty programu. Przyjęto, że czynności które wcześniej zostały wykonane przez procesor oraz czynności aktualnie wykonywane przez inne procesory, nie wpływają znacząco na czas realizacji badanego fragmentu. Ze względu na istotność tego założenia konieczne jest określenie zakresu jego stosowalności. W tym celu należy:

- Zbadać wpływ uprzednio wykonanych instrukcji na czas realizacji bloków, a w szczególności fragmentów zawierających niewielką liczbę instrukcji, dla których w istotny sposób może uwidocznić się wpływ pamięci podręcznej oraz przetwarzania potokowego i superskalarnego.
- Zbadać wpływ czynności wykonywanych przez inne procesy na czas realizacji badanego fragmentu kodu, a w szczególności przypadku gdy „konkurencyjne” procesy intensywnie używają wspólnego, ograniczonego zasobu (wspólny dysk, wspólny segment sieci).
- Zbadać korelację czasów wykonania dwóch kolejno następujących po sobie bloków. Badanie przeprowadza sie mierząc parami czasy realizacji bloku poprzedzającego i następującego po nim.
- Zbadać korelację czasów wykonania równolegle realizowanych bloków. Badanie wykonywane jest jako pomiar parami czasów bloków wykonywanych jednocześnie na dwóch maszynach (procesorach).

Testując hipotezy o niezależności rozkładów losowych czasu realizacji bloku programu od czynników mogących wpływać na kształt rozkładu, przeprowadzano pomiary dla różnych wartości czynnika zakłócającego. Następnie badano zgodność uzyskanych rozkładów, określając przedziały ufności i sprawdzając stopień ich pokrywania się, jak również wykonując test serii Walda-Wolfowitza. Szczegółowy opis doświadczeń wykonanych w celu określenia wzajemnych zależności pomiędzy blokami programu oraz ich wyniki zawarte są w dodatku A.

Podczas wykonywania doświadczeń oraz opracowywania wyników napotkano na pewne trudności związane z niedeterministycznym zachowaniem maszyny cyfrowej.

- W systemach współbieżnych występuje bardzo wiele trudnych do wyeliminowana czynników losowych mających wpływ na czas realizacji zadania. Trudno zatem ustalić wpływ konkretnego, badanego czynnika na kształt rozkładu losowego.
- Poszczególne pomiary dokonane w tych samych warunkach różnią się znacznie pomiędzy sobą. Testy pokazują niezgodność rozkładów, mimo realizacji doświadczeń w takich samych warunkach.
- Zwiększanie liczby doświadczeń nie prowadzi do osiągnięcia zgodności rozkładów dla pomiarów wykonywanych w tych samych warunkach. Przyczyną rozbieżności jest zatem niezauważalna dla obserwatora zmiana środowiska wpływająca na wynik doświadczenia.
- Wzajemny wpływ realizowanych procesów był obserwowalny jedynie dla doświadczeń, gdzie celowo tak dobrano warunki, aby maksymalnie zwiększyć stopień korelacji. W pozostałych przypadkach zakłócenia utrudniały wyciągnięcie jednoznacznych wniosków. Można zatem stwierdzić, że korelacja jest mała, trudno natomiast określić jej wartość.
- Mimo obecności zakłóceń zdecydowano się nie cenzurować wyników pomiarów, ze względu na niedostępność lub znaczne skomplikowanie testów statystycznych dla ocenzurowanych danych. Trudno też jednoznacznie określić jakie doświadczenia należy odrzucić a jakie nie.

Na podstawie przeprowadzonych doświadczeń wyciągnięto następujące wnioski:

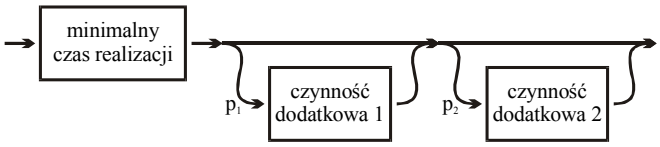
- Dla typowych programów, zależność czasu realizacji bloków dłuższych niż 1000 instrukcji od historii procesu jest minimalna. Dla badanych przypadków była ona mniejsza od zakłóceń pomiaru.
- Dla dobranych „złośliwie" programów obserwowano 5-10% zmianę wartości średniej czasu realizacji. Głównym czynnikiem był w tych przypadkach współczynnik trafień pamięci podręcznej danych. Wynik ten nie stanowi podstawy do odrzucenia hipotezy o niezależności czasu realizacji od historii procesu, gdyż prawdopodobieństwo wystąpienia wymienionych konstrukcji programowych w rzeczywistej aplikacji jest niewielkie, a zmierzona korelacja nie jest wysoka.
- Zależność rozkładu czasu realizacji bloku od rodzaju czynności wykonywanych przez inne procesory obserwowano jedynie dla przypadków, gdzie występował konflikt dostępu do powolnego urządzenia peryferyjnego. W pozostałych przypadkach potwierdzona została niezależność czasu operacji od obciążenia równolegle pracujących jednostek.
- Jedynym z ograniczonych zasobów powodujących konflikt jest limitowana przepustowość sieci lokalnej. Dla sieci typu CSMA-CD (np. ethernet-coax) obserwowana jest silna zależność czasów realizacji równolegle pracujących procesów, związana z oczekiwaniem na możliwość przesłania danych. Dla sieci o połączeniach bezkonfliktowych (np. duplex ethernet + switch) nie obserwowano wzajemnego wpływu czynności wykonywanych przez inne procesy na czas realizacji bloku.

Przeprowadzone pomiary potwierdziły dopuszczalność reprezentacji programu, w postaci powiązanych logicznie bloków o niezależnych, losowych czasach realizacji. Uzyskano uzasadnienie poprawności metod redukcji grafu opierających się na postulacie niezależności zmiennych losowych opisujących poszczególne fragmenty programu.

5.4. Model opóźnień występujących w systemie komputerowym

Uzyskiwane empirycznie rozkłady losowe czasu trwania operacji mają skomplikowany kształt złożony z głównego maksimum gęstości oraz kilku dodatkowych. Nie udało się zaproponować prostego wzoru funkcji mogącej stanowić wystarczająco dokładne przybliżenie gęstości lub dystrybuanty. Analizując naturę zjawiska zaproponowano stochastyczny model opóźnień w wykonywaniu aplikacji.

Przyjęto, że czas realizacji zadania jest równy sumie czasów składowych opisanych prostymi dwupunktowymi rozkładami. Każdy z czasów składowych opisuje w sposób stochastyczny jeden z elementów systemu mający wpływ na szybkość wykonywania instrukcji (np. zajętość magistrali, chybienie pamięci podręcznej) rys 5-1. Założono, że składniki sumy są niezależne od siebie. Opracowując model działania procesora przyjęto uproszczenie, że każdy ze składników sumy ma rozkład dwupunktowy. Krótszy czas odpowiada korzystnemu zjawisku (np. trafienie w pamięci podręcznej), które w większości przypadków jest bardziej prawdopodobne niż zjawisko niekorzystne (np. chybienie). Przyjęto więc, że prawdopodobieństwo wylosowania mniejszej wartości jest większe niż prawdopodobieństwo wylosowania większej. Tworząc model opóźnień dokładnie odwzorowujący pracę systemu należałoby uwzględnić co najmniej kilka możliwych opóźnień dla każdej realizowanej instrukcji. Autor postanowił sprawdzić, czy uda się skonstruować model opisujący rozkład losowy czasu realizacji bloku kilkuset instrukcji za pomocą 5-10 parametrów.



Rys. 5-1 Czas realizacji instrukcji jako suma czasu minimalnego oraz opóźnień występujących z określonym prawdopodobieństwem.

W celu weryfikacji przyjętego modelu wykonano symulacje i doświadczenia mające na celu:

- Zbadanie zgodności kształtu rozkładu losowego generowanego przez model z kształtem rzeczywistych rozkładów losowych czasów realizacji programu.
- Potwierdzenie możliwości określenia parametrów modelu na podstawie pomiarów.
- Sprawdzenie dokładności odwzorowania działania procesora przez przyjęty model poprzez porównanie przewidywanego rozkładu czasu wykonania procedury z rzeczywistym rozkładem.
- Określenie liczby parametrów modelu koniecznych do wiernego odwzorowani rozkładu rzeczywistego.
- Zbadanie skalowalności modelu od opisu realizacji pojedynczych instrukcji do dużych procedur oraz całości wykonywanego programu.

Na podstawie przeprowadzonych doświadczeń (opisanych w dodatku B) uzyskano następujące wnioski:

- Rozkłady losowe generowane przez zaproponowany model swoim kształtem i właściwościami w dużym stopniu przypominają rozkłady uzyskane empirycznie.
- Określenie parametrów modelu na podstawie danych empirycznych napotyka na trudności obliczeniowe. Długie opóźnienia określa się na podstawie dodatkowych maksimów gęstości badanego rozkładu. Wartości i prawdopodobieństwa krótkich opóźnień uzyskano za pomocą minimalizacji różnicy pomiędzy kształtem rozkładu generowanym przez model a rzeczywistym.

- Dla wielu rozkładów uzyskano model wiernie odwzorowujący kształt rozkładu. Dla około 1% rozkładów nie udało się przedstawić za pomocą modelu wszystkich dodatkowych maksimów gęstości. Dla maksimum odległego o t_x jednostek czasu oraz odległego o t_y w rozkładzie generowanym przez model powstaje również maksimum gęstości odległe o t_x+t_y. Jego prawdopodobieństwo jest bliskie $p_x p_y$ co nie zawsze pokrywa się z danymi empirycznymi (dla części doświadczeń brak wyników w tym obszarze).
- Przy zaproponowanej konstrukcji modelu stosunkowo dużo parametrów konieczne jest dla wiernego odtworzenia głównego maksimum gęstości. Jeżeli jednak dopuści się większą dyskretyzację skali czasu (brak opóźnień 1, 2, 3, 4 jednostki), to dla większości rozkładów uzyskano zadowalającą zgodność już przy 6 do 9 parametrach modelu.
- Zaproponowany model dobrze odwzorowywał rozkłady losowe czasu realizacji zarówno krótkich jak i długich procedur. Jego skalowalność umożliwia stosowanie go na każdym etapie analizy czasu realizacji aplikacji współbieżnej.

Zaproponowany model opóźnień umożliwia zapis informacji o rozkładzie losowym czasu realizacji procedury za pomocą zdecydowanie mniejszej liczby parametrów niż reprezentacja tabelaryczna. W pewnym stopniu powinien on również ułatwiać obliczenie rozkładu sumy niezależnych zmiennych losowych. Model rozkładu wynikowego zawiera bowiem wszystkie opóźnienia modeli składowych. Niestety do jego opisu konieczne jest dwa razy więcej parametrów (parametry obu sumowanych modeli). Nie znaleziono prostych metod umożliwiających redukcję liczby parametrów modelu przy zachowaniu kształtu i właściwości opisywanego rozkładu.

Przy pisaniu programów do automatycznej analizy grafu przepływu sterowania zastosowano zarówno reprezentację rozkładów w postaci tabel jak i wymienionego modelu.

5.5. Badanie charakteru opóźnień generowanych przez system operacyjny

W celu lepszego zbadania opóźnień wprowadzanych przez czynności systemu operacyjnego przeprowadzono badania dodatkowych maksimów gęstości rozkładu losowego czasu realizacji bloku. Dokonano 10 miliardów (10^{10}) pomiarów czasu realizacji pustej pętli, odrzucając pomiary dla których czas realizacji nie odbiegał od wartości minimalnej. Otrzymane 608 tys. wartości zdecydowanie przekraczających średnią przedstawiono w tabeli i na ich podstawie opracowano model opóźnień wnoszonych przez dany system operacyjny.

Na podstawie pomiarów czasu najkrótszej pętli stworzono model opóźnień wprowadzanych przez system. Następnie dokonano pomiarów czasów realizacji dłuższych fragmentów (10 i 100 razy) porównując czasy realizacji uzyskane z modelu z rzeczywistymi. Uzyskano przy tym bardzo dobrą zgodność wartości przewidywanych i empirycznych. Potwierdzona została zatem przydatność zaproponowanego modelu rozkładu losowego. Stwierdzono również, że opóźnienia wprowadzane przez system operacyjny co najmniej stukrotnie przekraczają czas realizacji instrukcji, mają więc duży wpływ na zakłócenie synchronizacji procesów.

5.6. Wnioski

Przeprowadzone doświadczenia potwierdziły słuszność przyjętych założeń dotyczących:

- Losowego charakteru czasu realizacji fragmentu kodu
- Zbliżonych kształtów i właściwości rozkładów losowych czasu obliczeń i komunikacji
- Niezależności czasów realizacji bloków (sekwencyjnych i równoległych)
- (ograniczonej) możliwości prowadzenia obliczeń analitycznych na przybliżonych funkcjach dystrybuant

Ze względu na charakter rozkładów losowych, konieczne jest przeprowadzanie bardzo dużych ilości doświadczeń aby uzyskać wiarygodne wyniki. Występujące sporadycznie bardzo duże wartości czasu mają istotny wpływ na wariancję i wyższe momenty rozkładu. Prawdopodobieństwo wystąpienia dużego opóźnienia jest poniżej $10^{-3} – 10^{-4}$ zatem dla dokładnego zbadania parametrów rozkładu konieczne jest przeprowadzenie co najmniej $10^{5} – 10^{6}$ doświadczeń.

Typowe testy statystyczne, zostały opracowane aby wyciągać wnioski przy (bardzo) małej liczbie przeprowadzonych doświadczeń. Dla bardzo dużej liczby próbek, minimalne błędy zaokrągleń powodują bezwarunkowe odrzucanie hipotez zgodności rozkładów lub identyczności parametrów.

Stwierdzono, że najpopularniejsze typy rozkładów (normalny i wykładniczy) nie są dobrym przybliżeniem kształtu rozkładu losowego czasu realizacji programu. Aby przeprowadzać analityczne obliczenia koniecznie jest użycie bardziej złożonych funkcji dystrybuanty. Pociąga to za sobą odpowiednio zawiłe wzory na splot dystrybuant (zwłaszcza w kolejnych krokach obliczeń) . Stawia to pod znakiem zapytania stosowalność metod analitycznych.

W przypadku numerycznych obliczeń konieczne jest wierne odwzorowanie zarówno głównej części rozkładu (skupionej wokół niskich wartości) jak i bardzo dużych incydentalnych wartości. Wymusza to konieczność stosowania tabel o bardzo dużej liczbie rekordów (większości zerowych). Alternatywnym rozwiązaniem jest reprezentacja osi argumentów w skali logarytmicznej (co wymaga odpowiedniej modyfikacji algorytmów numerycznych).

6. Automatyczna analiza grafu

6.1. Określenie czasu realizacji programu na podstawie opisu fragmentów

W celu weryfikacji tezy o możliwości uzyskania rozkładu losowego czasu realizacji programu na podstawie czasów realizacji fragmentów przeanalizowano kilka typowych struktur programów współbieżnych. Każdy z nich przedstawiono w postaci zredukowanego grafu przepływu sterowania. Następnie zbadano czasy realizacji fragmentów i na ich podstawie obliczono rozkład losowy czasu wykonania całego programu. Porównanie wyników uzyskanych na podstawie analizy grafu z zebranymi doświadczalnie pozawala na potwierdzenie (lub zaprzeczenie) poprawności prezentowanej metody.

Do celów analizy wybrano kilka struktur programów typowych dla algorytmów równoległych obliczeń numerycznych. Kształt grafu i wzajemne relacje synchronizacji są w tych przypadkach znane i jednoznacznie określone. Zrezygnowano z automatycznej interpretacji gramatyki języka C++ i tworzenia grafu na podstawie kodu źródłowego (lub wynikowego) gdyż zaciemniłoby to znacznie obraz analizowanego zagadnienia. Większość programów obliczeniowych jest optymalizowana pod kątem stosowanego sprzętu i kompilatora, przez co tracą przejrzystość struktury i łatwość automatycznej analizy.

Jako najbardziej reprezentatywne wybrano następujące algorytmy:

- Symulacja wzajemnego oddziaływania elementów sąsiadujących (transfer ciepła, prognoza pogody, aerodynamika, analiza drgań). W algorytmie tym możliwe jest prowadzenie obliczeń w trakcie oczekiwania na dane z innych procesorów.
- Symulacja wzajemnego oddziaływania wszystkich elementów (ruch obiektów w polu grawitacyjnym). Pomiędzy poszczególnymi krokami symulacji musi nastąpić całkowita wymiana danych pomiędzy procesorami.
- Wzajemna wymiana komunikatów (kontrola i sterowanie).
- Niezależne obliczenia (poszukiwanie klucza szyfru, grafika komputerowa). Po etapie rozdziału pracy pomiędzy procesami pracują one niezależnie.
- Algorytmy przeszukiwania z ograniczaniem (branch and bound). Procesory pracują w dużym stopniu niezależnie wymieniając się informacją o najlepszym dotychczas znalezionym rozwiązaniu.

6.2. Sposób reprezentacji rozkładu losowego i dokonywania obliczeń

Rozkład losowy czasu wykonywania fragmentu programu można przedstawić w postaci tabeli opisującej z zadanym przybliżeniem kształt funkcji gęstości lub dystrybuanty. Ze względu na szerokość dziedziny (przedział możliwych wartości czasów realizacji) i nietypowy kształt konieczne jest użycie bardzo wielu punktów dla wiernego odwzorowania kształtu rozkładu. Pociąga to za sobą odpowiednio większą złożoność operacji liczenia splotu i maksimum z doświadczeń losowych. Na podstawie przeprowadzonych pomiarów czasów realizacji fragmentów kodu stwierdzono, że przy reprezentacji tabelarycznej istotne jest zapewnienie:

- dokładnego odwzorowania niskich wartości czasu
- odwzorowania sporadycznie występujących bardzo dużych wartości czasu
- reprezentacji najkrótszego czasu (realizacji bloku) oraz najdłuższego (realizacji całego programu)
- akceptowalnego rozmiaru tabeli

Aby spełnić podane warunki zaproponowano następujące rozwiązania:

- Obszerne tabele o stałym kroku i dziedzinie pozwalającej objąć całkowity czas realizacji programu
- Skalowane tabele pokrywające zakres zmienności czasu aktualnie analizowanego fragmentu
- Tabele o logarytmicznej skali czasu.

Pierwsze rozwiązanie wymaga największej pamięci oraz ilości kroków przy numerycznym liczeniu splotu. Algorytmy obliczeniowe są natomiast proste, przez co szybkie (mimo większej liczby iteracji) i dosyć dokładne. Czasochłonność obliczania splotu rośnie z kwadratem liczby punktów, co ogranicza maksymalny rozmiar stosowanych tabel. W dalszej części pracy użyto tabel o stałym kroku i rozmiarze we wszystkich zaprezentowanych przykładach obliczeniowych.

Zastosowanie tabel o różnej skali czasu oszczędza pamięć i zmniejsza wyraźnie ilość iteracji. Wymusza jednak komplikację algorytmów obliczania splotu i iloczynu dystrybuant. W wielu przypadkach spada również dokładność obliczeń ze względu na to, że punkty, w których opisane są funkcje podlegające splataniu nie pokrywają się. Zastosowanie skalowanych tabel wymaga też precyzyjniejszego sprawdzenia czy nie wystąpią błędy zaokrągleń przy dodawaniu małego (rozkładu) czasu kolejnego bloku do dużego (rozkładu) czasu realizacji dotychczas wykonanej części programu.

Zamiast reprezentować rozkład losowy w formie (obszernej) tabeli można przybliżyć go jedną z opisanych funkcji dystrybuanty i zapamiętać jej kilka parametrów. Kosztem zmniejszenia dokładności uzyskuje się uproszczenie opisu. Wygodniej jest również przeprowadzać obliczenia gdyż całki i sploty typowych rozkładów są znane. Stosując ten sam typ rozkładu we wszystkich przekształceniach można stablicować najczęściej używane funkcje (np. splot) uzyskując znaczne przyśpieszenie analizy.

W literaturze do opisu czasu realizacji zadań proponowane są następujące typy rozkładów:

- wykładniczy [Amm85]
- hiper-wykładniczy (hypo-exp) [Mag93]
- normalny [Kam85a], [Kam86]
- Erlanga [Abd03]
- Weibula [Abd04]
- normalny ucięty – dodatnia połowa

Kształt najpopularniejszych rozkładów nie odpowiada jednak kształtowi rozkładu czasu realizacji programu zmierzonemu doświadczalnie. Na podstawie przedstawionych uprzednio badań kształtu rozkładu można zaproponować w charakterze przybliżeń następujące rozkłady losowe:

- logarytmo-normalny
- wykładniczy przesunięty
- Erlanga [Abd03]
- Weibula [Abd04]
- Hiperboliczny (Pareto)

Rozkład logarytmo-normalny najlepiej przybliża kształt funkcji gęstości w pobliżu maksimum. Jest jednak zbyt szybko prawostronnie zbieżny do zera i zbyt wolno lewostronnie (zwłaszcza dla krótszych fragmentów kodu).

Rozkład wykładniczy przesunięty jest najprostszym rozkładem mogącym przybliżyć rzeczywisty kształt funkcji gęstości czasu wykonania programu. Jest jednak zbyt szybko prawostronnie zbieżny do zera w porównaniu z wynikami doświadczeń. Nie umożliwia też przedstawienia ostrego maksimum gęstości dla rozkładów czasu realizacji krótszych fragmentów kodu.

Idea zastosowania rozkładów Erlanga i Weibula do opisu czasu realizacji zadań jest stosunkowo nowa. Stanowią one kompromis pomiędzy jakością odwzorowania kształtu a stopniem skomplikowania funkcji dystrybuanty.

Żaden z opisanych w literaturze rozkładów losowych nie umożliwia reprezentacji dodatkowych maksimów gęstości jakie obserwuje się w rzeczywistych pomiarach. Gęstości znanych rozkładów są też zbyt szybko prawostronnie zbieżne do zera w porównaniu do empirycznych. Prowadząc obliczenia z użyciem standardowych rozkładów uzyskuje się zatem bardziej zafałszowany wynik dla dłuższych (niż średnie) czasów realizacji programu. Stosunkowo wierne odwzorowanie mniejszych wartości stanowi niewielką zaletę gdyż w większości praktycznych zastosowań szczególnie ważna jest analiza przypadków, gdy program wykonuje się dłużej niż oczekiwano i może nie zmieścić się w narzuconych ramach czasowych.

W takcie analizy grafu realizacji programu oblicza się splot dystrybuant, ich sumę oraz iloczyn. W przypadku typowych kształtów rozkładu można dokonać obliczeń analitycznych. Każdy kolejny krok obliczeń prowadzi do coraz bardziej skomplikowanych wyrażeń opisujących rozkład ograniczając zakres stosowania metody. Przyjęto, zatem założenie o jednakowym kształcie wszystkich rozkładów czasu. Nie znaleziono jednak rozkładu, dla którego zarówno splot, suma ważona jak i iloczyn dystrybuant zachowywałyby typ rozkładu. Konieczne jest więc po każdym kroku obliczeń przybliżanie wyniku najlepiej pasującym rozkładem o założonym wcześniej kształcie. Dla przykładu sumę wykładniczych zmiennych losowych tworzącą rozkład Erlanga zastępuje się najlepiej dopasowanym rozkładem wykładniczym i przystępuje się do następnego kroku obliczeń. Prowadzi to do zmniejszenia dokładności obliczeń, oraz uniemożliwia określenie rzeczywistej postaci rozkładu losowego czasu wykonania całego programu.

6.3. Program analizujący

W celu weryfikacji opisanej w pracy metody napisano program automatycznie analizujący graf i obliczający rozkłady losowe czasu osiągnięcia każdego miejsca. Program został napisany w języku C++ co zapewnia jego uniwersalność i umożliwia łatwą adaptację do nowych kryteriów. Tworząc program założono, że:

- Wszystkie czasy opisane są rozkładem tego samego typu (funkcja dystrybuanty lub tabela)
- Miejsce grafu ma jednego poprzednika i jednego następnika
- Prawdopodobieństwo uszkodzenia danego fragmentu jest stałe i znane

Wszystkie obliczenia w programie przeprowadzane są na zmiennych obiektowych definiowanych na podstawie klasy wirtualnej „Rozklad”. Program jest przystosowany do analizy z użyciem różnych typów rozkładów pod warunkiem zdefiniowania operatorów sumowania i wyboru największej wartości z N losowań. Aby uniknąć warunkowej analizy grafu nie zaimplementowano procedur obliczania rozkładu losowego czasu realizacji synchronizacji negatywnej.

W celu sprawdzenia, w jakim stopniu przyjęcie założeń o kształcie rozkładu wpływa na wynik obliczeń przeprowadzono obliczenia dla następujących typów rozkładów:

- Normalnego
- Wykładniczego przesuniętego
- Opisanego w formie tabeli

Klasa C++ „Rozklad" przy założeniu rozkładu normalnego zawiera trzy parametry (średnią, wariancję i prawdopodobieństwo uszkodzenia). Rozkład sumy zmiennych losowych jest liczony w sposób analityczny jako suma wartości oczekiwanych i wariancji. Dystrybuanta rozkładu będącego wynikiem wyboru większej wartości z N losowań jest uzyskiwana numerycznie jako iloczyn dystrybuant składowych. Następnie obliczana zostaje wartość oczekiwana i wariancja otrzymanego rozkładu i na ich podstawie konstruowany rozkłada normalny stanowiący przybliżenie. Wartość dystrybuanty rozkładu normalnego zostaje obliczona na początku programu jako całka z gęstości i przechowywana w tablicy. Dla rozkładu normalnego N(0,1) zastosowano zakres (-100, 100) przy kroku 1/1000. W przypadku analizy złożonych grafów opłacalne może okazać się stablicowanie operacji obliczenia rozkładu zastępczego dla przypadku większej wartości czasu z dwóch losowań.

Jeżeli obliczenia przeprowadza się przy założeniu, że wszystkie rozkłady są wykładnicze przesuniete to klasa „Rozklad" zawiera trzy parametry (lambda, przesunięcie i prawdopodobieństwo uszkodzenia). Jak wykazano we wcześniejszych rozdziałach zastosowanie „klasycznego" rozkładu wykładniczego (bez przesunięcia) jest zbyt niedokładne. W programie rozkład sumy zmiennych losowych obliczano jako splot, a następnie dokonywano przybliżenia otrzymanego rozkładu rozkładem wykładniczym. Suma zmiennych losowych o rozkładach wykładniczych tworzy rozkład Erlanga, który różni się istotnie od rozkładu wykładniczego w zakresie mniejszych wartości. Przybliżając tak otrzymany rozkład rozkładem wykładniczym zbadano dwa kryteria doboru parametrów:

- Wykorzystanie wartości minimalnej oraz średniej otrzymanego rozkładu. Parametry te są estymatorami o największym prawdopodobieństwie dla próbki losowej, o ile wiadomo, że rozkład jest wykładniczy [Joh72]. Dodatkową zaletą jest szybkość obliczeń, gdyż parametry te są sumą parametrów rozkładów składowych. Wadą metody, jest gorsze przybliżenie rozkładu od strony mniejszych wartości.
- Przybliżenie rozkładu rozkładem wykładniczym metodą aproksymacji (minimalny kwadrat błędu). Otrzymany rozkład gorzej reprezentuje próbki o większych wartościach.
- Przybliżenie rozkładu rozkładem wykładniczym metodą aproksymacji (minimalna wartość bezwzględna błędu). Otrzymany rozkład stosunkowo najlepiej reprezentuje próbki o małych i dużych wartościach.

Dla rozkładów reprezentowanych w postaci tabelarycznej obliczenia splotu i iloczynu dokonywane są w sposób numeryczny. Rozkład wynikowy zapamiętywany jest w postaci tabeli i bez dodatkowych przekształceń używany do dalszych obliczeń. Reprezentacja numeryczna umożliwia obliczenie kształtu rozkładu czasu realizacji programu, jednak jest najbardziej skomplikowana obliczeniowo i wymaga odpowiednio dużej pamięci do przechowywania danych.

6.4. Dane wejściowe i wyniki

Dane wejściowe pobierane są z plików tekstowych. Plik GRAF.TXT określa strukturę grafu Petriego, natomiast CZAS.TXT określa rozkłady losowe czasów trwania poszczególnych przejść. Przyjęto reprezentację grafu w pliku wejściowym jako zbioru miejsc wraz z ich poprzednikami i następnikami. Zbiór przejść w opisywanym grafie jest tworzony jako suma zbiorów poprzedników i następników każdego przejścia. Przejście o numerze zero zdefiniowane jest jako przejście startowe. Czas osiągnięcia każdego z miejsc nie jest znany na początku obliczeń. Zostaje on wyliczony w trakcie pracy programu analizującego na podstawie rozkładów losowych czasu osiągnięcia poprzedzających miejsc oraz przejść łączących. Plik zawierający opis struktury grafu składa się z rekordów składających się z trzech pól.

- Pierwsza linia rekordu oznacza identyfikator miejsca.
- Druga linia określa numer przejścia będącego poprzednikiem danego miejsca
- Trzecia linia określa numer przejścia będącego następnikiem danego miejsca

Przykładowy fragment pliku opisującego graf:

```
M0015      // miejsce nr 15
 0000      // poprzednikiem miejsca jest przejście nr 0
 0025      // następnikiem  miejsca jest przejście nr 25
M0016      // miejsce nr 16
 0025      // poprzednikiem miejsca jest przejście nr 25
 0026      // następnikiem  miejsca jest przejście nr 26
```

Powyższy plik opisuje graf o dwóch miejscach i trzech przejściach. Miejsce M0015 otrzymuje znacznik z przejścia startowego i oddaje go do przejścia 0025. Miejsce M0016 otrzymuje znacznik z przejścia 0025 i oddaje go do przejścia 0026.

Rozkład losowy czasu wykonania każdego z przejść musi być znany przed przystąpieniem do analizy grafu. Potrzebne informacje zawiera plik CZAS.TXT. Jest to plik tekstowy zawierający rekordy danych wejściowych dla każdego z przejść. Dla ułatwienia analizy złożonych grafów o powtarzających się parametrach dopuszczone jest używanie gwiazdek na każdej z czterech pozycji identyfikatora przejścia.

Przykład rekordu (dla rozkładu normalnego)

```
P00*1      // przejścia 0001, 0011, 0021, ..., 0091
 12.57     // wartość średnia
 3.16      // odchylenie standardowe
 0.01      // prawdopodobieństwo uszkodzenia
```

Przykład rekordu (dla rozkładu wykładniczego przesuniętego)

```
P00*1      // przejścia 0001, 0011, 0021, ..., 0091
 10.27     // wartość minimalna
 4.00      // 1/lambda
 0.01      // prawdopodobieństwo uszkodzenia
```

Przykład rekordu (dla rozkładu stablicowanego)

```
P00*1       // przejścia 0001, 0011, 0021, ..., 0091
 R001.TXT   // Plik zawierający kształt funkcji gęstości
 0.01       // prawdopodobieństwo uszkodzenia
```

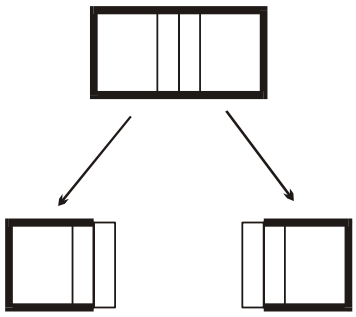
Wynikiem analizy jest plik tekstowy zawierający parametry rozkładów losowych czasu odpalenia każdego z przejść. Przy przeprowadzaniu analizy w oparciu o rozkład normalny rozkład wynikowy zawiera trzy parametry: wartość średnią, odchylenie standardowe i prawdopodobieństwo uszkodzenia. W przypadku użycia innych rozkładów w pliku wyjściowym zostają wydrukowane wymagane parametry charakteryzujące rozkład czasu realizacji całego zadania.

6.5. Analiza programu „life”

Jako pierwszą aplikację testową wybrano algorytm „life” symulujący żywotność „komórek” w zależności od liczby sąsiadów. Działanie programu sprowadza się do obliczenia wartości wszystkich elementów tablicy w kolejnym kroku na podstawie wartości w kroku poprzednim. Algorytm cechuje się dużą lokalnością danych, gdyż jedynie sąsiednie elementy brane są pod uwagę przy obliczaniu wartości. Możliwe jest zatem proste i efektywne wykonywanie działań przez maszynę współbieżną. Analogiczne właściwości ma wiele zadań realizowanych obecnie przez superkomputery (dynamika płynów, prognoza pogody, analiza drgań)

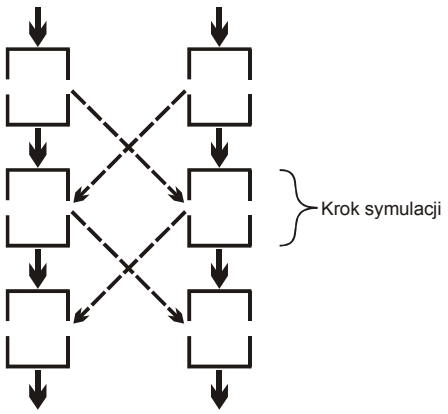
Podział zadania pomiędzy procesory

Podział zadania pomiędzy procesory sprowadza się do podziału tablicy „komórek” na fragmenty i prowadzeniu równoległych obliczeń dla każdego z nich. Stan elementu tablicy w następnym kroku zależy jedynie od bezpośrednich sąsiadów, co pozwala do minimum ograniczyć komunikację pomiędzy procesorami. W każdym kroku konieczna jest bowiem wymiana informacji o wartościach komórek „granicznych” a nie całych wartościach tablic.



Rys. 6-1 Podział tablicy N kolumn na dwie tablice N/2+1 kolumn

Algorytm „life” skaluje się bardzo dobrze, gdyż dla N*N elementów tablicy potrzeba przesłać jedynie const*N wartości w każdym kroku. Stosując macierze o kształcie prostokątnym (np. 1000x5) można w dużych granicach modyfikować stosunek długości „obszarów granicznych” tablicy do ilości wszystkich elementów. Można zatem dla tego samego algorytmu (i jego grafu) przeprowadzać doświadczenia o skrajnie różnych proporcjach części obliczeniowej i komunikacyjnej. Kolejną właściwością programu „life” jest możliwość takiego uszeregowania kolejności obliczeń i komunikacji, aby uniknąć biernego oczekiwania na dane z drugiego procesora. Odpowiednio wczesne obliczenie warunków brzegowych i wysyłanie wyników do współpracującego procesu umożliwia zniwelowanie opóźnień wnoszonych przez sieć. Na początku następnego kroku prowadzone są obliczenia lokalne, zatem istnieje dodatkowy margines czasowy zanim dane o warunkach brzegowych stają się konieczne.

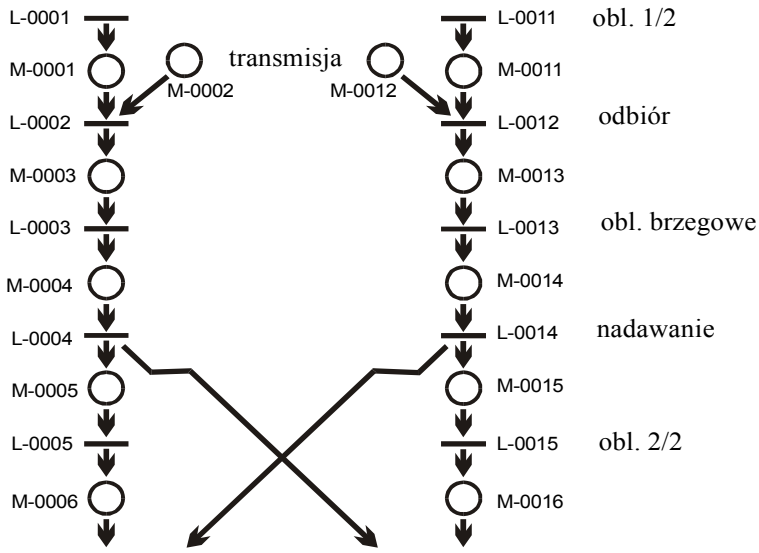


Rys. 6-2 Równoległa symulacja „life”

Na Rys. 6-2 przedstawiono sposób realizacji programu i przesyłania informacji. Grubsze strzałki symbolizują kolejność wykonywania instrukcji przez każdy z procesów, przerywane symbolizują przesyłane komunikaty. Prostokąty symbolizują kroki symulacji wykonywane przez procesory. Są one podzielone na dwie części. Najpierw wykonywana jest połowa obliczeń nie wymagająca informacji o „warunkach brzegowych”. Po wykonaniu tych obliczeń oczekiwana jest informacja o wartości granicznych komórek z poprzedniego kroku symulacji. Na podstawie tej informacji każdy proces oblicza nową wartość skrajnych komórek swojego obszaru i natychmiast rozsyła ją do pozostałych procesów. Po wysłaniu informacji kontynuowane są obliczenia pozostałej części tablicy. Opisany algorytm powstał z myślą o wykorzystaniu komunikatów asynchronicznych (buforowanych).

Graf Petriego

Dla przedstawionego algorytmu można skonstruować stosunkowo prosty graf Petriego pojedynczego kroku symulacji.



Rys. 6-3 Graf Petriego kroku równoległej analizy life

Krok obliczeń składa się z obliczenia nowych wartości dla połowy tablicy co jest reprezentowane przez przejścia L-0001 i L-0011 odpowiednio dla pierwszego i drugiego procesu. Jednocześnie przez sieć transmitowane są dane o warunkach brzegowych obliczonych w poprzednim kroku symulacji M-0002 i M-0012. Kolejnym krokiem jest odbiór komunikatów L-0002 i L-0012. Następnie następuje obliczenie nowych warunków brzegowych L-0003 i L-0013 i wysłanie ich do kooperującego procesu L-0004 i L-0014. Ostatnim etapem obliczeń jest uzyskanie nowych wartości dla drugiej połowy analizowanej tablicy L-0005 i L-0015.

Graf programu wykonującego więcej niż jeden krok symulacji jest złożeniem odpowiedniej ilości przedstawionych powyżej grafów. Konieczne jest przy tym odpowiednie przenumerowanie miejsc i przejść.

6.6. Pomiar szybkości realizacji programu „life”

W oparciu o przedstawiony algorytm został napisany program dokonujący obliczeń kolejnych kroków symulacji life. Aplikacja przeznaczona jest dla systemów klasy UNIX zapewniających komunikację TCP/IP. Program zawiera część wstępną, dokonującą pomiaru szybkości wykonywania każdego fragmentu programu osobno. Potem następuje synchronizacja i rozpoczęcie wykonywania właściwego algorytmu. W trakcie pracy programu każdy fragment jest również otoczony instrukcjami pomiaru czasu wykonania. Umożliwia to stwierdzenie różnic pomiędzy szybkością realizacji tych samych fragmentów kodu wykonywanych osobno oraz będących składnikami kompletnego programu. Umieszczenie instrukcji pomiaru czasu po każdym fragmencie dodaje do każdego bloku narzut czasowy związany z wykonaniem procedury pomiarowej równoważny narzutowi przy pomiarach odosobnionych fragmentów. Ułatwia to dokonywanie porównań, gdyż procedury pomiarowe w jednakowy sposób obciążają system.

Testy przeprowadzone zostały na dwóch identycznych komputerach klasy Pentium pracujących pod kontrolą systemu Linux. Podłączone są one do jednego segmentu sieci ethernet 10Mb-TP. Do tego segmentu podłączonych jest jeszcze kilka pracujących komputerów, co może mieć wpływ na zajętość sieci i opóźnienia z tym związane. Dla każdego rozmiaru zagadnienia 2000 razy uruchamiano program wykonujący po 20 kroków. Każdorazowe uruchomienie dokonywało również 20 pomiarów czasu wykonania fragmentu programu. Dla każdego rozmiaru zagadnienia otrzymano zatem 40 tys. pomiarów czasu wykonania każdego z fragmentów oraz 2000 pomiarów czasu wykonania całego programu. Wyniki otrzymano zarówno dla pierwszego jak i drugiego komputera. O ile czasy wykonywania pojedynczych fragmentów przez oba procesory można uznać za niezależne to czas realizacji właściwego programu jest silnie skorelowany dla obu maszyn (co wynika ze struktury algorytmu).

Wartości średnie oraz odchylenia standardowe czasów realizacji programu przedstawione są w tabelach 6-1 i 6-2. Czasy podane są w mikrosekundach. Pola tabeli zawierają następujące dane:

- Rozmiar zagadnienia określa liczbę komórek tablicy podlegającej przekształceniom w obrębie każdego procesu. Druga współrzędna rozmiaru określa jednocześnie liczbę komórek sąsiadujących z tablicą przechowywaną przez drugi proces. Liczbę bajtów koniecznych do przesłania przy każdym korku symulacji jest zatem proporcjonalna do wartości drugiej współrzędnej rozmiaru zagadnienia. Litery „a” i „b” oznaczają pomiar dokonany dla pierwszego i drugiego procesora.
- Kolumna pomiar przedstawia wyniki uzyskane doświadczalnie. Wartości czasów podane są w mikrosekundach.

- Kolumna zatytułowana analiza przedstawia symulowany czas wykonania programu life wyznaczony na podstawie grafu przedstawionego na rys. 6-2. Czasy wykonania elementów składowych każdego kroku zostały zebrane doświadczalnie.
- Kolumna suma szer. przedstawia symulowany czas realizacji programu przy założeniu, że komunikat dociera zawsze szybciej, niż występuje konieczność jego odczytu. Zatem fragmenty grafu opisujące komunikację można pominąć. Oznacza to, że czas realizacji programu jest sumą czasów realizacji instrukcji lokalnych procesora.
- Kolumna suma sieć przedstawia symulowany czas wykonywania programu przy założeniu, że procesor zawsze musi czekać na otrzymanie komunikatu, zatem szybkość komunikacji ma decydujący wpływ na czas wykonywania zadania.

Tabela 6-1. Wartości średnie czasu realizacji programu „life”

rozmiar	pomiar	analiza	suma szer.	suma sieci
4x4a	14700	9510	5385	8475
4x4b	14864	7452	3336	6420
10x10a	10827	14872	6400	12275
10x10b	11453	13618	4693	10462
20x20a	14430	10095	11121	9780
20x20b	14815	8783	9565	8071
100x100a	245807	175353	288103	11832
100x100b	246050	120548	241092	9060
10x1000a	250454	189469	289285	65709
10x1000b	250759	116311	232617	63710

Tabela 6-2. Odchylenie standardowe czasu realizacji programu „life”

rozmiar	pomiar	analiza	suma szer.	suma sieci
4x4a	12549	1113	1945	3911
4x4b	12715	837	386	3415
10x10a	3785	1733	1556	6892
10x10b	3911	1629	318	6720
20x20a	2287	837	1266	3320
20x20b	2344	698	515	3105
100x100a	800	19287	49218	2437
100x100b	720	515	739	357
10x1000a	669	21386	58085	1361
10x1000b	530	601	861	706

Wyniki analizy programu „life” przy założeniu wykładniczego kształtu rozkładów przedstawione są w tabelach 6-3 i 6-4. Czasy podane są w mikrosekundach. Pola tabeli zawierają następujące dane:

- Rozmiar zagadnienia określa liczbę komórek tablicy podlegającej przekształceniom w obrębie każdego procesu.
- Kolumna pomiar przedstawia wyniki uzyskane doświadczalnie. Wartości czasów podane są w mikro-sekundach.
- Kolumna zatytułowana analiza przedstawia symulowany czas wykonania programu „life" wyznaczony na podstawie grafu przedstawionego na rys. 6-2. Czasy wykonania elementów składowych każdego kroku zostały zebrane doświadczalnie.
- Kolumna suma szer. przedstawia symulowany czas realizacji programu przy założeniu, że komunikat dociera zawsze szybciej, niż występuje konieczność jego odczytu. Zatem fragmenty grafu opisujące komunikację można pominąć. Oznacza to, że czas realizacji programu jest sumą czasów realizacji instrukcji lokalnych procesora.
- Kolumna suma sieć przedstawia symulowany czas wykonywania programu przy założeniu, że procesor zawsze musi czekać na otrzymanie komunikatu, zatem szybkość komunikacji ma decydujący wpływ na czas wykonywania zadania.

Tabela 6-3. Wartości minimalne czasu realizacji programu „life" przy założeniu wykładniczego przesuniętego kształtu rozkładu czasu.

rozmiar	pomiar	analiza	suma szer.	suma sieci
4x4a	9011	6570	1260	6570
4x4b	9052	6510	1200	6510
10x10a	7559	6760	2460	6760
10x10b	7899	6740	2440	6740
20x20a	12454	7740	7740	7100
20x20b	12617	7720	7720	7080
100x100a	244840	236440	236440	11530
100x100b	245175	236460	236460	11470
10x1000a	249980	229900	229900	65290
10x1000b	250248	229700	229700	65150

Tabela 6-4. Wartości średnie czasu realizacji programu „life” przy założeniu wykładniczego przesuniętego kształtu rozkładu czasu.

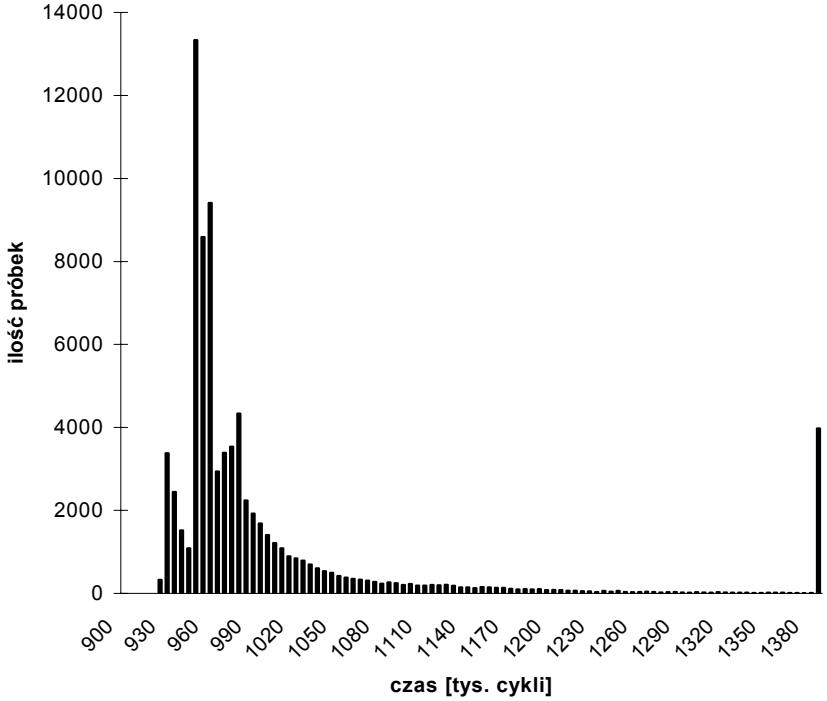
rozmiar	pomiar	analiza	suma szer.	suma sieci
4x4a	14700	9510	3385	6475
4x4b	14864	9452	3336	6420
10x10a	10827	14872	4400	10275
10x10b	11453	13618	4693	10462
20x20a	14430	12095	11121	9780
20x20b	14815	12783	9565	9071
100x100a	245807	255353	248103	11832
100x100b	246050	250548	241092	10060
10x1000a	250454	259469	249285	65709
10x1000b	250759	266311	232617	63710

Kształt rozkładu losowego czasu realizacji programu „life”

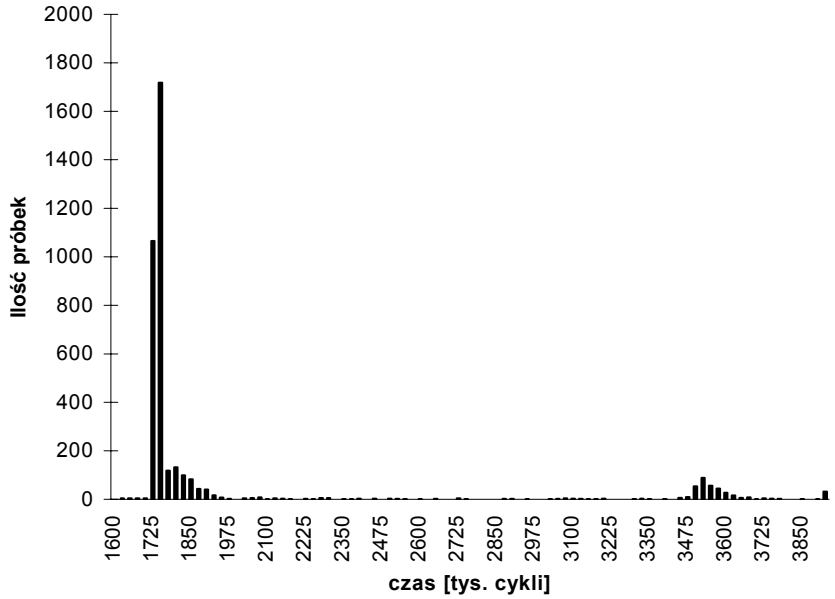
W celu określenia kształtu rozkładu losowego czasu wykonania programu „life” przeprowadzono pomiary wykorzystując licznik cykli procesora pentium. Zdecydowano się dokonać pomiarów dla rozmiaru 10x10. Przy tym rozmiarze zagadnienia czas dokonywania obliczeń lokalnych jest w przybliżeniu równy czasowi transmisji danych przez sieć. Program testowy uruchamiano 4 razy, jednocześnie na komputerach gorg1 i gorg2. Każdorazowo dokonano 10 tys. pomiarów. Wyniki dla komputera gorg1 opisane są w tabeli jako G1, dla gorg2 jako G2. Histogram stworzono uwzględniając wyniki z obu komputerów, czyli razem 80 tys. pomiarów. Tabela 6-5 przedstawia parametry rozkładu losowego czasu wykonania programu „life” dla rozmiaru 10x10. Mediana, średnia, odchylenie std., średnia logarytmiczna są podane w tysiącach cykli procesora Pentium.

Tabela 6-5. Parametry rozkładu czasu wykonania programu „life” dla rozmiaru 10x10

	G1-1	G1-2	G1-3	G1-4	G2-1	G2-2	G2-3	G2-4
Mediana	955	961	972	951	955	963	975	950
Śr.-log	984	1031	1045	981	984	1031	1045	981
Średnia	993	1052	1070	991	993	1053	1070	991
Odch. S.	219	270	306	215	217	270	307	216

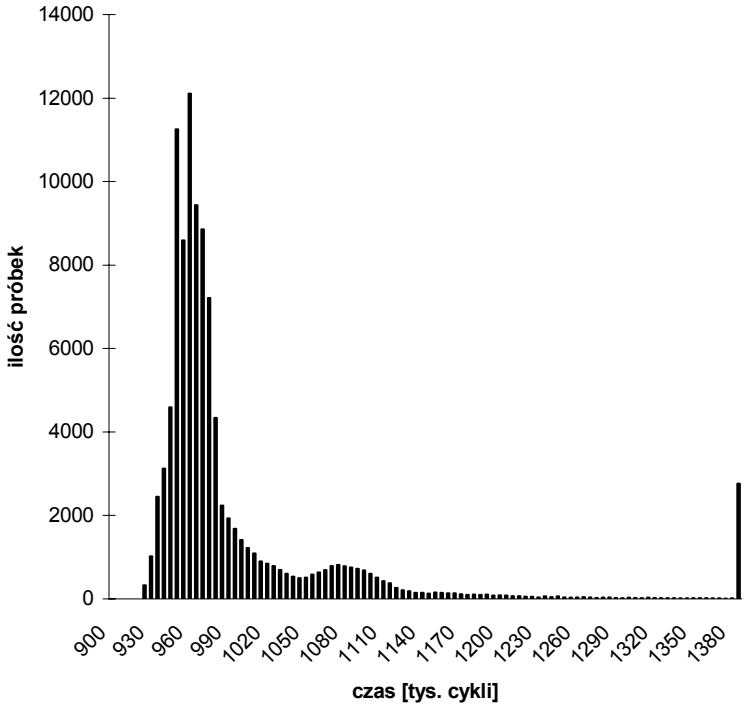


Rys. 6-4. Histogram czasu wykonania programu „life" – rozmiar 10x10



Rys. 6-5. Histogram dodatkowych maksimów czasu programu „life" (powiększony)

Prowadząc analizę grafu obrazującego program „life" przy użyciu rozkładów reprezentowanych numerycznie otrzymano przewidywany rozkład losowy czasu realizacji całego programu. Przedstawiono go na rys. 6-6 w skali identycznej jak wyniki doświadczeń na rys. 6-4.



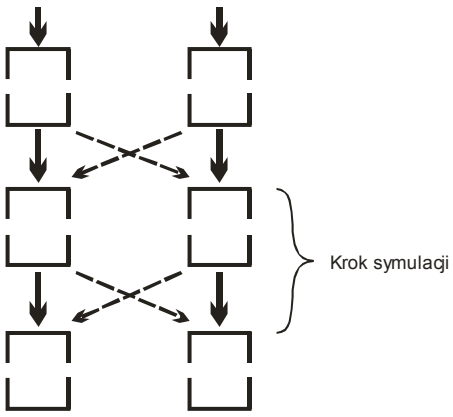
Rys. 6-6. Histogram przewidywanego czasu wykonania programu „life" – rozmiar 10x10

Zaobserwowano istotne różnice pomiędzy danymi zebranymi doświadczalnie, a wynikami analizy oraz niezgodności pomiędzy wynikami doświadczeń.

- Wartość średnia czasu wykonania programu jest większa dla rozmiaru 4x4 niż 10x10.
- Występuje znaczna różnica pomiędzy wartościami zebranymi doświadczalnie, a uzyskanymi na drodze analizy grafu.
- Uzyskiwane są inne wartości czasów dla procesu A i B pomimo ich symetrii.
- Wartości odchylenia standardowego rozkładu normalnego uzyskane w drodze analizy i doświadczeń bardzo mocno się różnią.
- Kształt rozkładu losowego uzyskany w wyniku analizy grafu jest bardziej „gładki" oraz posiada maksima lokalne przesunięte w stosunku do rozkładu otrzymanego doświadczalnie.

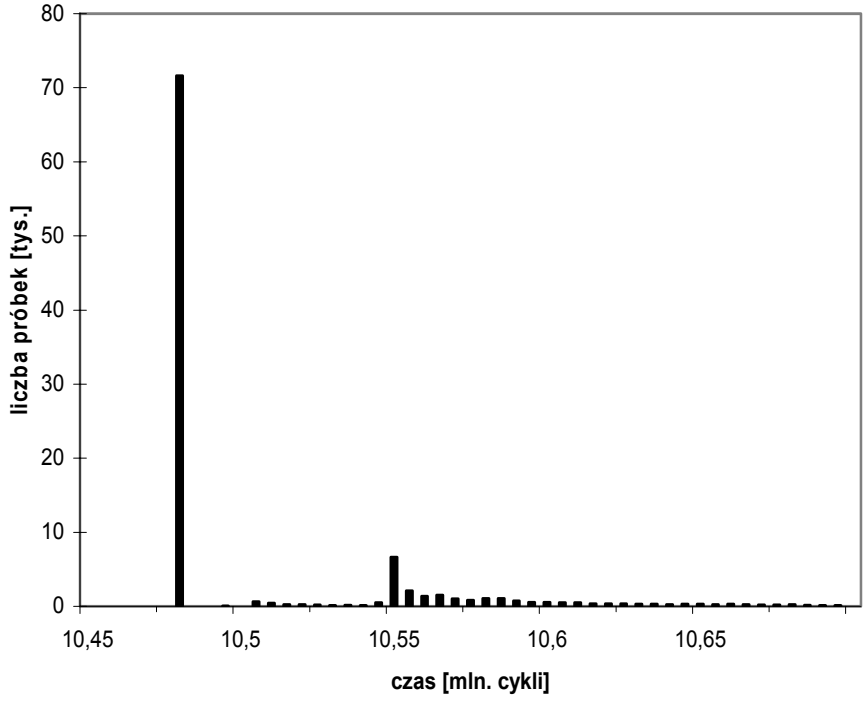
6.7. Analiza programu „star-dust"

Przykładem równoległej symulacji oddziaływań wszystkich elementów jest ruch planetoid w polu grawitacyjnym. Obiekty opisane są rekordami zawierającymi masę, położenie i prędkość. Każdy z pracujących równolegle procesów przetwarza przydzielony mu zbiór planet, potrzebując jednak aktualnych danych o wszystkich obiektach. Po każdym kroku obliczeń konieczna jest, więc pełna wymiana danych. Wzajemny przeplot faz obliczeń, komunikacji i synchronizacji idealnie odpowiada modelowi BSP [Val90]. Prowadząc obliczenia dla n planet na N procesorach złożoność obliczeniowa jest rzędu n^2/N (każda planeta oddziałuje z każdą) natomiast komunikacyjna nN (każdy procesor potrzebuje danych o każdej planecie).



Rys. 6-7. Struktura programu „star-dust”

Przeprowadzono pomiary czasu wykonania 100 tys. kolejnych kroków obliczeń zapisując po każdym etapie aktualną wartość licznika cykli procesora (Pentium 2,5GHz). Następnie na podstawie zebranych danych utworzono histogramy czasów realizacji pojedynczego bloku (100 tys. pomiarów) i grup po sto bloków (2000 pomiarów). Oczekuje się, że ze względu na strukturę programu czas realizacji będzie prostą sumą czasów składowych. Dla programów o strukturze BSP (łańcuch kolejno realizowanych współbieżnych bloków) powinno mieć zastosowanie centralne twierdzenie graniczne, czyli rozkład wynikowy powinien mieć kształt normalny.

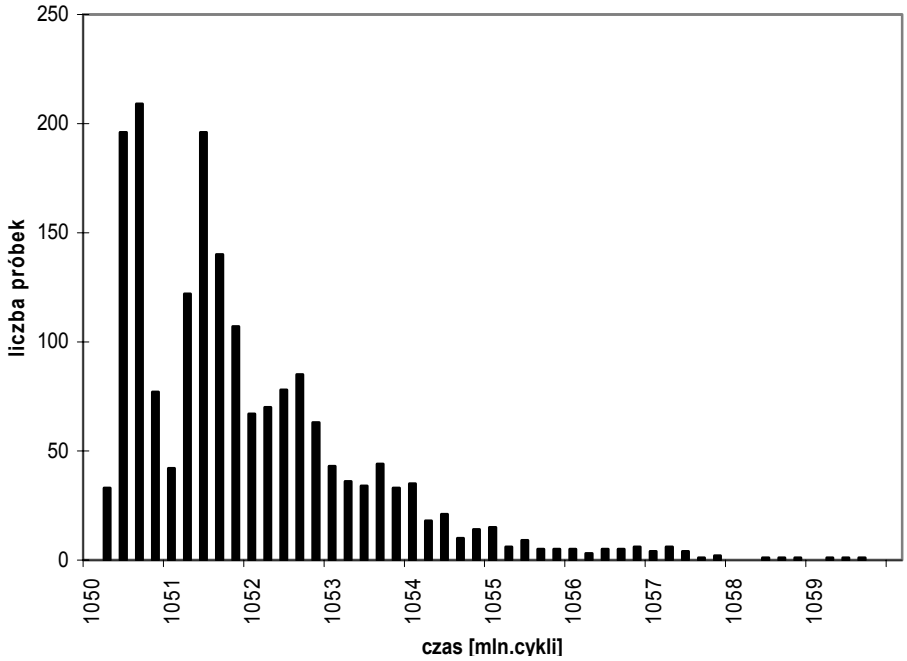


Rys. 6-8. Histogram czasu realizacji pojedynczego kroku obliczeń programu „star-dust”

Na podstawie pomiarów obliczono parametry rozkładu losowego czasu realizacji pojedynczego kroku uzyskując śred = 10,5427 mln oraz var = 0,376e12. Histogram przedstawiony na rysunku 6-8 zawiera obszar głównego skupienia wartości. Dodatkowo 3,36% próbek znajduje się poza wykresem, rozproszone szeroko po stronie większych wartości.

Czas realizacji stu kolejnych bloków stanowi sumę niezależnych zmiennych losowych. Oczekiwać należy zatem że rozkład będzie miał kształt zbliżony do normalnego i parametry odpowiednio śred = 1054,27 mln oraz

var = 37,6e12. Uzyskany rozkład jest jednak w dalszym ciągu niesymetryczny a jego parametry śred = 1054,27 mln oraz var = 90,15e12 (2,4 razy większa). Skośność rozkładu wyliczona na podstawie próbek wynosi 6,0 zatem jest on zdecydowanie niesymetryczny. Histogram przedstawiony na rysunku 6-9 zawiera obszar największego prawdopodobieństwa. Dodatkowo 6,9% próbek znajduje się poza wykresem, po stronie większych wartości.



Rys. 6-9. Histogram czasu realizacji stu kroków obliczeń programu „star-dust"

Mimo, że algorytm „star-dust" wybrano tak, aby posiadał jak najbardziej regularną strukturę i przewidywalne właściwości statystyczne, otrzymano wyniki znacznie odbiegające od oczekiwanych. Na podstawie przeprowadzonych pomiarów można wyciągnąć następujące wnioski:

- Rozkład czasu realizacji programu nie ma kształtu normalnego. Nawet dla modelu BSP, gdzie struktura programu jest wręcz stworzona aby spełniać założenia centralnego twierdzenia granicznego, otrzymany kształt rozkładu sumy stu zmiennych bardziej przypomina wykładniczy niż normalny.
- Wartość skośności rozkładu rzędu 6,0 jest niezwykle duża (rozkład wykładniczy ma 2,0). Główny wpływ mają na nią sporadycznie występujące próbki o bardzo dużych opóźnieniach.
- W miarę wydłużania mierzonego fragmentu kodu rozkład losowy czasu jego wykonania stopniowo upodabnia się go Gaussowskiego. Zachodzi to jednak niezwykle „opornie" – dla porównania suma 20 symetrycznych rozkładów dwupunktowych lub 15 równomiernych daje całkiem dobre przybliżenie kształtu $\exp(-x^2)$.
- Wariancja rozkładu czasu realizacji fragmentu programu jest parametrem niezwykle trudnym do wiarygodnego zmierzenia. Obecność sporadycznie występujących opóźnień o dużych wartościach wymaga przeprowadzenia bardzo dużej liczby doświadczeń. Obliczenia z użyciem wariancji obarczone są zdecydowanie większym błędem niż dla wartości średniej.
- Wariancja sumy czasów realizacji bloków jest większa niż suma wariancji składowych. Świadczy to o (niewielkiej) dodatniej korelacji kolejnych sumowanych zmiennych. Ze względu na przerwania

o charakterze periodycznym należałoby się spodziewać raczej ujemnej korelacji. Większy wpływ na czas realizacji kolejnych bloków mają zatem dłuższe czynności sytemu operacyjnego lub/i przerwania nadchodzące w pakietach (ang. burst).

- Autorzy [Xu02] analizujący model BSP z definicji zakładają nieograniczoną stosowalność centralnego twierdzenia granicznego. Aby uniknąć poważnych błędów konieczne jest jednak dokładne sprawdzenie spełnienia jego założeń oraz kształtów otrzymywanych rozkładów losowych.

6.8. Analiza programu „ping-pong"

W celu zbadania zachowania programu, w którym czas komunikacji procesów odgrywa kluczową rolę napisano program „ping-pong" wymieniający dwudziestokrotnie komunikat o stałej treści pomiędzy dwoma komputerami. Zmierzono czasy trwania poszczególnych operacji i na ich podstawie określono przewidywany rozkład losowy czasu wykonania całego programu, a następnie porównano z wynikami doświadczalnymi.

Tabela 6-6 przedstawia parametry doświadczalnie określonego rozkładu losowego czasu wykonania programu „ping-pong". Mediana, średnia, odchylenie std., średnia logarytmiczna są podane w tysiącach cykli procesora Pentium. Program testowy uruchamiano 4 razy, jednocześnie na komputerach gorg1 i gorg2. Każdorazowo dokonano 10 tys. pomiarów. Wyniki dla komputera gorg1 opisane są w tabeli jako G1, dla gorg2 jako G2. Histogram stworzono uwzględniając wyniki z obu komputerów, czyli razem 80 tys. pomiarów.

Tabela 6-6. Parametry rozkładu czasu realizacji programu ping-pong.

	G1-a	G1-b	G1-c	G1-d	G2-a	G2-b	G2-c	G2-d
min.	848	851	851	850	849	850	849	848
mediana	863	884	857	862	864	885	857	862
śr.-log	868	1075	864	869	866	1075	864	868
średnia	885	1210	873	870	885	1210	873	870
odch. S.	1321	1728	844	66	1321	1731	844	66

Na podstawie pomiarów czasów poszczególnych operacji przeprowadzono analizę grafu programu obliczając przewidywany czas trwania całego programu i porównując ze zmierzonym.

Przy założeniu normalnego kształtu rozkładu w wyniku analizy otrzymano rozkład losowy wykonania całości programu o parametrach E=1811 i σ=684. Przybliżając wyniki doświadczeń rozkładem normalnym otrzymuje się E=955 i $\sigma \approx 800$.

Porównując wartości przewidziane poprzez analizę grafu oraz zebrane doświadczalnie można zauważyć, że wartość oczekiwana uzyskana w wyniku analizy przekracza wartość uzyskiwaną doświadczalnie. Jest to spowodowane uproszczeniem zakładającym normalny kształt rozkładu. W przypadku komunikacji pomiędzy komputerami odchylenie standardowe rozkładu czasu przesłania informacji przekracza często wartość oczekiwaną. Wynika z tego, że rozkład normalny przybliżający rozkład rzeczywisty posiada powyżej 15% próbek w obszarze liczb ujemnych. Powoduje to zawyżanie wartości oczekiwanej podczas obliczania rozkładu losowego czasu synchronizacji (oczekiwania na komunikat).

Wniosek:

Przybliżanie rozkładu czasu przesłania komunikatu rozkładem normalnym prowadzi do poważnych błędów podczas obliczeń.

Na podstawie 80 tys. doświadczeń nie udało się uzyskać wiarygodnej wartości odchylenia standardowego rozkładu czasu przesyłania pojedynczego komunikatu. Wartość ta (kluczowa dla rozkładu normalnego) podlega znacznym wahaniom powodowanym sporadycznie występującymi bardzo dużymi wartościami.

Wniosek:

Trudności z uzyskaniem wiarygodnej wartości odchylenia standardowego zarówno fragmentu, jak i całości programu wykluczają użycie rozkładu normalnego jako przybliżenia rozkładu czasu przesłania komunikatu.

Przy założeniu wykładniczego przesuniętego kształtu rozkładu w wyniku analizy grafu wyliczono Min=821 i $1/\lambda$=215. Dla porównania przybliżając wyniki doświadczeń rozkładem wykładniczym otrzymuje się Min=848 i $1/\lambda$=108.

Wykorzystanie rozkładów wykładniczych zapewnia większą dokładność analizy, jednak również w tym przypadku występują rozbieżności. Parametry rozkładu uzyskiwane są na podstawie wartości minimalnej oraz średniej z próbek. Na podstawie doświadczeń można stosunkowo dokładnie wyznaczyć wartość minimalną, natomiast wartość średnia podlega wahaniom spowodowanym przez sporadycznie pojawiające się bardzo znaczące opóźnienia w pracy programu. Próby liczenia lambdy na podstawie wariancji obarczone są błędem przekraczającym wartość mierzoną.

Wniosek:

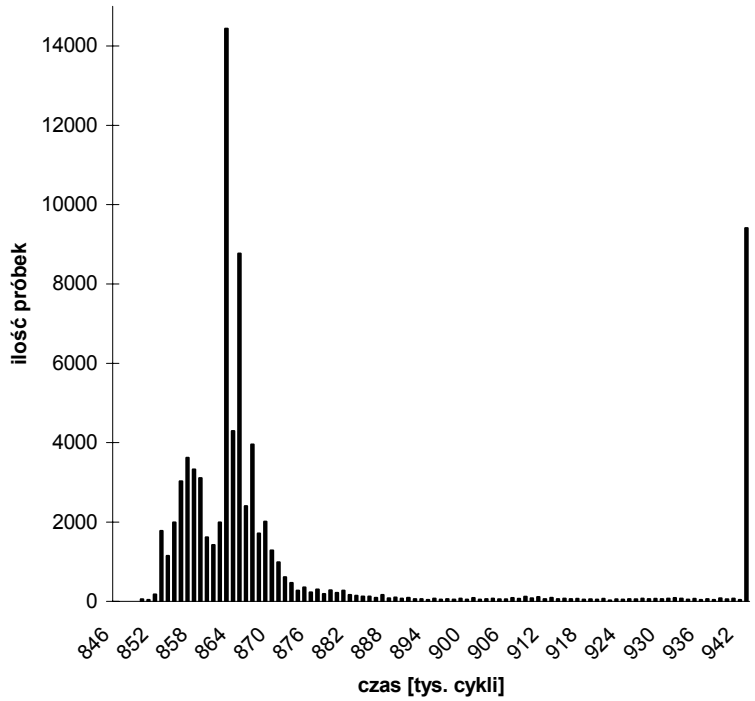
Użycie odpowiednich estymatorów, niewrażliwych na sporadyczne próbki o bardzo dużych wartościach jest kluczowe dla uzyskania wiarygodnych wyników.

Wniosek:

Zarówno dla rozkładu normalnego jak i wykładniczego okazuje się konieczne przeprowadzenie większej ilości doświadczeń, lub przyjęcie innej metody określania parametrów rozkładu (np. na podstawie ocenzurowanej próbki).

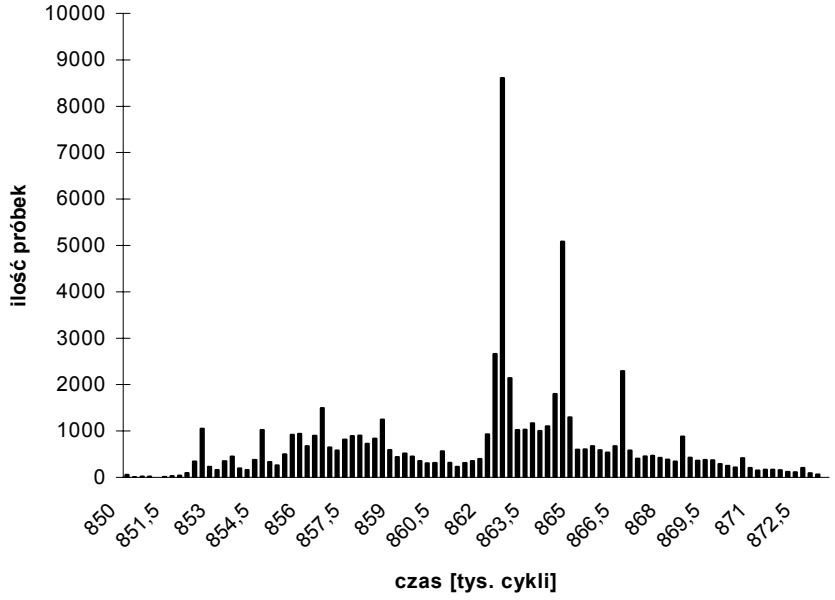
Kształt rozkładu losowego czasu realizacji programu „ping-pong”

W celu znalezienia funkcji w możliwie dokładny sposób przybliżającej rozkład losowy czasu wykonania programu przeprowadzono 80 tys. pomiarów czasu wykonania programu „ping-pong”. Wyniki przedstawiono na rys. 6-10. Rysunki 6-11 i 6-12 przedstawiają zbliżenia maksimum gęstości rozkładu, oraz obszaru próbek o dużych wartościach.

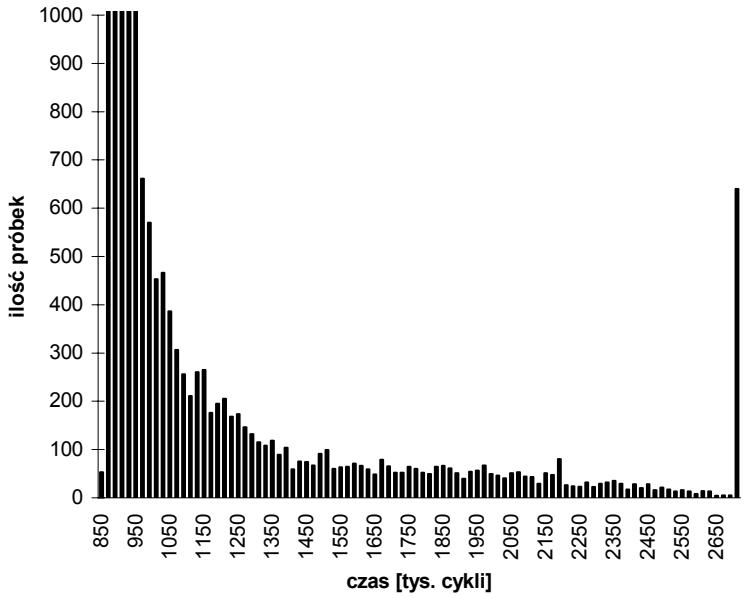


Rys. 6-10. Histogram czasu wykonania programu „ping-pong”

Podobnie jak analizowane wcześniej rozkłady czasu wykonania programu, rozkład czasu programu „ping-pong” ma kształt zbliżony do hiperboli w prawej części. Gęstość prezentowanego rozkładu jest wolniej zbieżna prawostronnie do zera niż prezentowany poprzednio rozkład czasu realizacji programu „life”. Jest to spowodowane dominującym wpływem czasu trwania komunikacji. Znaczące przedłużenie czasu transmisji pakietu jest zdecydowanie bardziej prawdopodobne niż porównywalne przedłużenie czasu trwania obliczeń.



Rys. 6-11. Histogram centrum gęstości rozkładu czasu wykonania programu ping-pong



Rys. 6-12. Histogram – prawe zbocze rozkładu gęstości programu „ping-pong"

Porównując wyniki doświadczeń z wartościami obliczonymi zauważa się bardzo istotny wpływ sposobu reprezentacji rozkładów losowych na dokładność uzyskanych rezultatów. Zaobserwowano, że przybliżanie rzeczywistych rozkładów losowych przez rozkład normalny prowadzi do istotnych błędów. W wielu przypadkach uzyskuje się charakterystykę czasu realizacji programu, z której wynika, że z prawdopodobieństwem ponad 10% czas realizacji jest ujemny! Poważne niedokładności wynikają również z przyjęcia założenia o czysto wykładniczym rozkładzie czasu realizacji fragmentów kodu. Lepsze rezultaty uzyskiwane są podczas przeprowadzania obliczeń w oparciu o rozkłady wykładnicze przesunięte. Zadowalającą dokładność uzyskuje się jednak dopiero przy przeprowadzaniu obliczeń uwzględniających rzeczywisty kształt rozkładów losowych. Wiąże się to jednak ze znacznym spowolnieniem procesu analizy grafu i koniecznością ograniczenia rozmiaru badanych zagadnień. Można zatem stwierdzić, że przy zapewnieniu dużej dokładności danych wejściowych, oraz sposobu reprezentacji i przekształceń rozkładów losowych możliwe jest uzyskanie wiarygodnego rozkładu losowego czasu realizacji aplikacji współbieżnej.

6.9. Programy słabo powiązane

W celu zbadania zachowania aplikacji o (bardzo) niewielkim udziale komunikacji przeprowadzono pomiar szybkości realizacji programów „crypt" i „tsp-bb". Pierwszy z nich jest zrównoleglonąą wersją programu poszukującego klucza do szyfru metodą przeszukiwania wszystkich możliwości. Komunikacja sprowadza się w tym algorytmie do rozdzielenia zadań pomiędzy procesory i ewentualnego (bardzo rzadkiego) informowania procesu nadrzędnego o znalezieniu klucza, który umożliwia uzyskanie tekstu zawierającego dopuszczalne językowo połączenia samogłosek i spółgłosek.

Program „tsp-bb" realizuje algorytm poszukiwania optymalnej drogi w grafie (traveling salesman problem) za pomocą szacowania i ograniczania (branch and bound). Komunikacja sprowadza się do rozdziału zadań oraz informowania o znalezieniu lepszego ograniczenia. Nie zaimplementowano wyrównywania obciążeń pomiędzy procesorami, jeżeli dzięki efektywniejszemu ograniczaniu część z nich wcześniej zakończyła pracę.

Analiza opisanych programów metodami opisanymi w pracy napotyka na trudności, ponieważ:

- Brak jest jawnej synchronizacji
- Liczba i rozmieszczenie (zrealizowanych) operacji komunikacji nie są statycznie zdefiniowane.
- Algorytm poszukiwania klucza jest de-facto powielonym wielokrotnie algorytmem jednowątkowym. Wzajemne interakcje procesorów praktycznie nie istnieją. Najlepszą w tym przypadku metodą analizy jest określenie rozkładu losowego pojedynczego procesu a następnie dodanie prostego modelu rozdziału zadań i zbierania wyników.
- Efektywność ograniczania (branch and bound) zależy między innymi od sprawnej wymiany informacji o najlepszym znalezionym dotychczas rozwiązaniu. Nie jest to jednak żaden rodzaj synchronizacji lecz zwiększenie wydajności dzięki posiadaniu dodatkowej informacji.

Mimo ograniczonej stosowalności metody opisanej w pracy przeprowadzono porównanie teoretycznego i empirycznego rozkładu losowego czasu realizacji aplikacji. Dla programu „crypt" uzyskano dużą zgodność dzięki niezależności czasu realizacji od danych wejściowych. Rozkład losowy czasu poszukiwania drogi w grafie jest w dużym stopniu zależny od rozmiaru i trudności rozwiązywanego problemu. W większości przypadków uzyskiwano dużą zgodność przewidywań z rezultatami pomiarów. Niektóre wyniki różniły się nawet o rząd wielkości, gdyż prezentowana w pracy metoda przyjmuje stochastyczny opis instrukcji warunkowych i pętli. Nie uwzględnia bardzo silnej zależności liczby iteracji „branch and bound" od wartości przetwarzanych danych.

6.10. Przyczyny niepełnej zgodności wyników przewidywań i doświadczeń

Przyczyny rozbieżności pomiędzy wynikami uzyskanymi w drodze analizy grafu oraz doświadczeń mogą mieć źródło zarówno w niedokładnościach pomiarów, jak i ograniczeniach programu analizującego. Najbardziej prawdopodobnymi przyczynami niezgodności wyników są:

- zbyt mała liczba przeprowadzonych doświadczeń,
- generowanie przerwań zakłócających pomiar czasu następnego bloku,
- opóźnienia wprowadzane przez protokół sieciowy,
- wpływ pamięci podręcznej na szybkość realizacji kodu,
- ładowanie kodu programu w trakcie jego wykonywania,
- usypianie i budzenie procesów oczekujących na komunikację,
- niedokładna metoda pomiaru czasu przesyłania danych przez sieć,
- niejawna konkurencja o dostęp do sieci,
- założenie normalnego kształtu rozkładu czasu realizacji programu.
- założenie braku korelacji czasu wykonywania kolejnych bloków

Za mała próbka

Otrzymane wyniki wskazują, że liczba doświadczeń w niektórych przypadkach była zbyt mała. Często jednak zwiększenie liczby doświadczeń nie było możliwe ze względów czasowo-organizacyjnych. Dla niezależnych pomiarów szybkości wykonywania poszczególnych fragmentów kodu widoczne są znaczne różnice czasu dla obu (identycznych) maszyn, co teoretycznie nie powinno mieć miejsca. Spowodowane są one przerwaniami systemu operacyjnego wprowadzającymi duże opóźnienia, aczkolwiek z występującymi z małym prawdopodobieństwem. Aby uzyskać pełniejszy obraz rozkładu losowego należałoby przeprowadzić

zdecydowanie więcej doświadczeń (które trwałyby tygodniami) lub określić zdarzenia mało prawdopodobne na drodze teoretycznej.

Przerwania zakłócające pomiar czasu kolejnych bloków

Instrukcje asynchronicznej komunikacji składają się z części jawnej – wykonywanej jako funkcja programu użytkownika, oraz niejawnej inicjowanej przez przerwanie sprzętowe. Przerwanie i związany z nim narzut czasowy może wystąpić w momencie wykonywania innego fragmentu programu niż ten, który go wywołał. Powoduje to przekłamania w pomiarze czasu wykonywania fragmentów programu. Częściowym rozwiązaniem jest pomiar szybkiego, n-krotnego wykonania tego samego fragmentu. Przerwania generowane przez ten fragment zostaną wtedy zaliczone w poczet czasu wykonania tego samego fragmentu w którymś z kolejnych kroków pomiarowych. Problem w takim przypadku może stanowić przeciążenie interfejsu sieciowego żądaniami nadchodzącymi znacznie szybciej niż w normalnym programie.

Protokół sieciowy

Zaawansowany protokół sieciowy wymaga dokonania wielu czynności wstępnych umożliwiających uruchomienie interfejsu i przesłanie pakietu. Część z nich jest dokonywana w momencie inicjowania komunikacji, część przed wysłaniem pakietu. W szczególności system musi sprawdzić i ew. uzupełnić informację dotyczącą routingu i ARP. W związku z tym wysłanie pierwszego pakietu obarczone jest dodatkowym narzutem czasowym. Przy wysyłaniu dalszych system korzysta z kopii podręcznych. Obserwując wyniki pomiarów można stwierdzić, że pierwsza instrukcja send() wykonuje się o 30-50% dłużej niż kolejne.

Pamięć podręczna

Ze względy na ograniczoną pojemność pamięci podręcznej obserwowany jest efekt szybszego wykonywania fragmentu kodu, kiedy jest on n-krotnie wykonywany w procedurze pomiarowej, niż później we właściwym programie. Rozbieżności spowodowane tym zjawiskiem można zniwelować umieszczając w procedurze pomiaru szybkości fragmentu dodatkowy kod mający za zadanie wypełnienie pamięci podręcznej w stopniu zbliżonym do właściwego programu. Trudność może nastręczać oszacowanie stopnia użycia pamięci podręcznej przez właściwy program.

Ładowanie na żądanie

Większość systemów operacyjnych umożliwia szybki start programu i oszczędność pamięci dzięki mechanizmowi ładowania na żądanie. Do pamięci ładowane są tylko te strony kodu programu, które są rzeczywiście potrzebne. Innymi słowy wystąpienie błędu strony powoduje jej załadowanie z pliku zawierającego treści programu. Operacja ładowania brakującego kodu wymaga czasu, zatem pierwsze wykonanie danego fragmentu będzie trwało dłużej niż następne iteracje.

Powolne „budzenie" procesu

Jeżeli dane odbierane przez recv() nie dotarły jeszcze do komputera, proces odbierający jest wprowadzany w stan uśpienia, a procesor przystępuje do wykonywania innych zadań. Po nadejściu żądanej informacji proces oczekujący jest ponownie przywracany do normalnego stanu i kontynuowany. Operacja usypiania i budzenia procesu wymaga znacznego zaangażowania procesora. Wynika z tego, że minimalna różnica w czasie dotarcia

pakietu może wywołać dużą różnicę czasu wykonania instrukcji odbioru. Zaprzecza to częściowo tezie, że synchronizacja pozytywna prowadzi zawsze do wyrównania szybkości poszczególnych procesów. Dla zagadnienia o rozmiarze 4x4 wyraźnie zaobserwowano kolejne wysuwanie się na prowadzenie raz jednego, raz drugiego procesu. Czas wykonania instrukcji odbioru oscylował pomiędzy wartością 450 μsec a 18 μsec.

Niedokładny pomiar czasu przesyłania komunikatu przez sieć

Ze względu na trudności pomiaru opóźnienia przesyłania danych przez sieć dokonano pomiaru szybkości procedury symulującej komendę ping a następnie „odjęto" czas wykonania instrukcji send() i recv(). Przyjęta metoda nie zapewnia dużej dokładności, zwłaszcza w odniesieniu do wcześniej poruszanych problemów związanych z pomiarem szybkości komend send() i recv(). Jej zaletą jest jednak uniwersalność, dostępność i brak ingerencji w system operacyjny. Dodatkowo komenda ping nie powoduje konkurowania systemów o dostęp do kanału, natomiast w rzeczywistym programie ze względu na symetrię komunikacji sytuacja taka będzie występować stosunkowo często.

Niejawna konkurencja o dostęp do sieci

W programie „life" przyjęte jest założenie, że oba procesy mogą jednocześnie wysłać komunikat. Gdy komputery połączone są pojedynczym segmentem sieci ethernet następuje niejawna konkurencja o dostęp do niepodzielnego kanału. Powoduje ona dodatkowe czasy oczekiwania na zwolnienie kanału, kolizje pakietów, generację nadprogramowych przerwań.

Założenie normalności rozkładu dla dużych wariancji

Rozkład losowy czasu wykonania fragmentu programu swoim kształtem znacznie odbiega od rozkładu normalnego. W rzeczywistym rozkładzie maksimum gęstości jest położone w obszarze liczb wyraźnie mniejszych od wartości średniej. Przyjęcie do obliczeń rozkładów normalnych równoważnych, co do wartości średniej i odchylenia powoduje duży błąd obliczeń. Jest on szczególnie istotny przy liczeniu rozkładu maksimum z dwóch liczb losowych. Przy liczeniu pochodnej iloczynu dystrybuant ważniejsze jest maksimum gęstości rozkładu niż jego wartość średnia. W wyniku pomiarów otrzymano rozkłady, dla których odchylenie standardowe znacznie przekracza wartość oczekiwaną, co kłóci się z założeniem o nieujemnym czasie wykonywania dowolnej czynności. Stosowanie rozkładu normalnego o takich parametrach prowadzi do poważnych błędów w obliczeniach. W związku z tym przy analizie rozkładów losowych o dużych wariancjach powinny one być przybliżane raczej rozkładem wykładniczym (przesuniętym) niż normalnym.

Założenie braku korelacji czasu wykonywania kolejnych bloków

W przyjętej metodzie założono, że czasy realizacji poszczególnych bloków są od siebie niezależne. W rzeczywistym systemie, istnieje duże prawdopodobieństwo zmniejszenia szybkości procesora podczas wykonywania kilku kolejnych bloków. Zdarzenie zewnętrzne lub wywołane działalnością systemu operacyjnego może obciążać procesor przez dłuższy czas, jak również spowodować wypełnienie pamięci podręcznych inną zawartością, co spowoduje wydłużenie czasu wykonywania kilku kolejnych bloków.

7. Podsumowanie i wnioski

W rozdziale 2 zaprezentowano metodę przekształcenia aplikacji współbieżnej w graf przepływu sterowania opisany siecią Petriego. Przyjęto, że czas realizacji fragmentu programu będzie reprezentowany przez dowolny rozkład losowy o zadanej funkcji dystrybuanty. Zaproponowane w pracy metody redukcji rozmiaru grafu umożliwiają uzyskanie sieci o rozmiarze liniowo zależnym od liczby instrukcji synchronizacji w programie. Przyjęte założenia dotyczące statycznie zdefiniowanych instrukcji synchronizacji umożliwiają uzyskanie grafu acyklicznego klasy PERT. Powiązanie grafu programu z przedstawionym w rozdziale 3 modelem zawodnej maszyny umożliwia analizę stanów niezawodnościowo-funkcjonalnych i obliczenie rozkładu losowego czasu realizacji programu.

Doświadczalnie potwierdzono słuszność przyjętych w modelu założeń dotyczących losowego charakteru czasu realizacji programu oraz niezależności czasów realizacji poszczególnych bloków.

Dla najbardziej reprezentatywnych aplikacji współbieżnych określono doświadczalnie kształt rozkładu losowego czasu realizacji programu i porównano go z wynikami uzyskanymi na podstawie analizy grafu. Stosując dokładną, tabelaryczną reprezentację nietypowych kształtów rozkładów losowych uzyskano zadowalającą zgodność wyników.

Świadczy to o potwierdzeniu tezy nr 1:

Możliwe jest przedstawienie programu równoległego ze statycznie definiowanymi instrukcjami synchronizacji w postaci uproszczonej stochastycznej sieci Petriego typu PERT. Rozmiar grafu jest liniowo zależny od ilości instrukcji synchronizacji. Za pomocą uzyskanego grafu można obliczyć kształt rozkładu losowego czasu realizacji programu.

Przeprowadzone pomiary czasów realizacji programów współbieżnych oraz ich fragmentów jednoznacznie potwierdzają bardzo duży rozrzut uzyskiwanych wyników. Mimo zapewnienia maksymalnie zbliżonych warunków przeprowadzania doświadczeń otrzymywano wyraźnie różniące się rezultaty poszczególnych prób. Wynika stąd, że system komputerowy, mimo iż teoretycznie w pełni deterministyczny, jest w takim stopniu skomplikowany, że jedyną możliwością opisu przewidywanego czasu realizacji zadania jest zapis probabilistyczny. Próby przeprowadzone dla różnych systemów operacyjnych potwierdziły wpływ platformy na kształt rozkładu (zwłaszcza na występowanie dodatkowych maksimów w jego prawej części).

Potwierdzono zatem w sposób doświadczalny tezę nr 2:

Czas realizacji programów współbieżnych może być traktowany jako zmienna losowa, której rozkład jest determinowany przez strukturę programu oraz realizującej go platformy.

Przeprowadzono doświadczenia mające na celu zbadanie możliwości uzyskania rozkładu losowego czasu realizacji programu na podstawie czasów realizacji jego fragmentów. Przebadano regularne struktury programów sekwencyjnych i współbieżnych, dla których czas realizacji był sumą zmiennych losowych o jednakowych rozkładach. Zaobserwowano, że nawet dla stu sekwencyjnie występujących po sobie bloków (o mniej lub bardziej złożonej strukturze wewnętrznej) rozkład sumaryczny wyraźnie nie przypomina kształtu rozkładu Gaussa. Dla porównania przeprowadzono proste obliczenia sumy zmiennych losowych o rozkładach wykładniczym, dwupunktowym i ciągłym. W tych przypadkach już dla sumy 10-20 zmiennych kształt rozkładu jest już bardzo dobrym przybliżeniem rozkładu normalnego.

Zaobserwowano, że obliczona na podstawie próbek wariancja czasu realizacji fragmentu programu jest bardzo wysoka. Co więcej, powtarzanie doświadczenia przy zachowaniu możliwie takich samych warunków prowadzi do uzyskania znacznie różniących się wartości wariancji. Za każdym razem przeprowadzano 1000 lub więcej pomiarów – należałoby więc oczekiwać dużej zgodności wyników. Uznano zatem, że wariancja rozkładów losowych czasów realizacji procedur jest nieskończona.

Potwierdzono zatem w sposób doświadczalny tezę nr 3:

Rozkład losowy czasu realizacji całej aplikacji nie dąży do kształtu normalnego, mimo że jest sumą (bardzo) wielu zmiennych losowych (czasów realizacji poszczególnych procedur). Rozkłady losowe czasów realizacji fragmentów programu nie spełniają założeń centralnego twierdzenia granicznego.

W celu zbadania właściwości rozkładów losowych przeprowadzono doświadczenia zawierające od 10 tys. do 10 miliardów (10^{10}) prób. Na ich podstawie stwierdzono, że rozkład losowy ma kształt nieregularny, posiada wiele lokalnych maksimów gęstości rozkładu, a rozpiętość przyjmowanych wartości wielokrotnie przewyższa wartość średnią. Rozkłady losowe czasów realizacji są mocno niesymetryczne. Posiadają one bardzo wyraźne ograniczenie lewostronne i praktycznie nie są ograniczone prawostronnie. Kształt rozkładu nie jest zbliżony ani do rozkładu normalnego ani wykładniczego, często stosowanych do opisu zjawisk losowych ze względu na łatwość obliczeń. Stwierdzono, że stosunkowo dobrym przybliżeniem rzeczywistego rozkładu może być hiperbola. Przybliżając empirycznie uzyskiwane histogramy rozkładem Pareto (hiperbolicznym) uzyskiwano parametr α mniejszy od dwóch. Dla $\alpha<2$ rozkład ma nieskończoną wartość wariancji (i wyższych momentów). Powtarzając kilkukrotnie doświadczenia stwierdzono, że w tych samych warunkach otrzymywano zdecydowanie różne, bardzo duże wartości wariancji. Potwierdza to przypuszczenie, że dla rozkładów czasu realizacji programu wariancja jest nieskończona.

Potwierdzono zatem w sposób doświadczalny tezę nr 4:

Rozkład losowy czasu realizacji programu (dla większości aplikacji) należy do klasy „heavy tail" czyli ma w przybliżeniu kształt hiperboliczny (Pareto) z parametrem $\alpha<2$. Rozkłady tej klasy mają nieskończoną wariancję.

7.1. Wnioski

Posługując się modelem sieciowym programu i maszyny oraz wykonując pomiary uzyskano potwierdzenie tez postawionych w pracy. Prezentowana metoda analizy jest możliwa do praktycznego zastosowania, jednak wymaga dokładnego określenia rozkładów losowych czynności składowych, oraz precyzyjnych obliczeń. Tworząc osobne modele programu i maszyny potwierdzono możliwość i celowość oddzielenia analizy sprzętu i oprogramowania. Proponowany model aplikacji w postaci sieci Petriego może okazać się bardzo przydatny do oszacowywania przewidywanej szybkości działania programu na różnych maszynach. Nabiera to szczególnego znaczenia przy rosnących kosztach tworzenia oprogramowania i wynikającej z tego konieczności pracy jednego programu na różnych platformach sprzętowych.. Znajomość rozkładu losowego czasu realizacji programu jest niezwykle istotna przy tworzeniu aplikacji o miękkich ograniczeniach czasowych (soft real time). Szczególne korzyści można uzyskać przy składaniu programu z gotowych komponentów. Przypisanie rozkładów czasu realizacji komponentom programowym umożliwia wygodne obliczenie zmienności losowej całej aplikacji

7.2. Kierunki dalszych badań

Zaprezentowany w pracy model stanowi zwięzłą reprezentację programu i maszyny umożliwiając oszacowanie przewidywanego czasu realizacji programu. W miarę komplikacji stosowanych aplikacji i maszyn cyfrowych można oczekiwać powstawania coraz bardziej złożonych modeli umożliwiających dokładniejszy opis zachodzących zjawisk. W związku ze wzrostem wydajności obliczeń i udoskonaleniami architektury współczesnych procesorów obserwuje się odchodzenie od opisu szybkości realizacji poszczególnych instrukcji na rzecz statystycznych miar efektywności systemu. Szybkość przetwarzania danych lokalnych zdecydowanie przewyższa szybkość przesyłania informacji do sąsiednich jednostek obliczeniowych, popularność zyskują więc modele współbieżności o stosunkowo luźnej synchronizacji procesów. Proces wykonywania aplikacji rozpatrywany jest jako realizacja powiązanych, większych bloków (np. BSP), a nie pojedynczych instrukcji (np. PRAM).

Określenie właściwości statystycznych maszyny na podstawie jej budowy

Jednym z poważniejszych wyzwań występujących przy stosowaniu przedstawionej w pracy metody (lub podobnych) jest konieczność określenia statystycznych parametrów maszyny i systemu operacyjnego. Dla każdego współczesnego systemu istnieje jego opis (w formacie Hardware Description Language) mogący posłużyć do automatycznego opracowania rozkładu losowego szybkości realizacji programów przez dany komputer. Na szczególną uwagę zasługują opisy na poziomie behawioralnym, gdyż niższy poziom jest zbyt złożony. Interesującym przedmiotem badań będzie stworzenie i zbadanie metod pozwalających na określenie właściwości procesora na podstawie jego opisu. Przydatne mogą okazać się zarówno metody w pełni analityczne, jak też bazujące całkowicie lub częściowo na metodzie Monte-Carlo.

Określenie właściwości statystycznych aplikacji podczas jej kompilacji

Współczesne kompilatory w celu optymalizacji kodu przeprowadzają złożoną analizę programu poddawanego tłumaczeniu. Wydaje się możliwe wykorzystanie tych mechanizmów w celu uzyskania statystycznego obrazu realizacji programu. W szczególności należałoby zwrócić uwagę na automatyczne określanie prawdopodobieństw skoków. Prawdopodobieństwa te stanowią kluczowy element opisu grafu prezentowanego w pracy.

Dynamiczne powiązanie współpracujących procesów

Prezentowana w pracy metoda analizy grafu programu wymaga, aby synchronizacja pomiędzy procesami była statycznie zdefiniowana w momencie startu aplikacji. Założenie to prosto spełnić dla programów prowadzących obliczenia matematyczne w systemie wieloprocesorowym. Dużo trudniej wymusić statyczną synchronizację komputerów pracujących w sieci rozległej. Rozwijający się w szalonym tempie sektor usług internetowych bazuje na współpracy komputerów w warunkach ciągłej rekonfiguracji struktury połączeń, obciążeń systemów, dostępności zasobów. Ciekawym wyzwaniem będzie opracowanie metody analizy uwzględniającej dynamiczną zmianę „partnerów” oraz ilości instrukcji synchronizacyjnych, przy zachowaniu akceptowalnej złożoności modelu.

Probabilistyczny charakter synchronizacji współczesnych aplikacji

W pracy założono, że synchronizacja poszczególnych procesów jest zjawiskiem pewnym. We współczesnych aplikacjach rozproszonych założenie to jest jednak coraz trudniejsze do spełnienia. Wiele aplikacji musi działać przy występujących okresowo zanikach łączności. Komunikacja i synchronizacja staje się więc zjawiskiem losowym. Zarówno dla dużych systemów (np. rezerwacja biletów) jak i małych (np. sieci ZigBee) kluczowe jest działanie w warunkach dynamicznej rekonfiguracji połączeń. Osłabienie kluczowego w pracy założenia o pewnej synchronizacji poszerzyłoby zakres stosowania prezentowanej metody.

Zastąpienie sieci PERT grafem cyklicznym

W pracy zaproponowano, przekształcenie złożonego grafu przepływu sterowania programu w acykliczny graf PERT godząc się na niezbędne, związane z tym uproszczenia. Wadą takiego rozwiązania jest konieczność acyklizacji pętli, co powoduje wzrost wielkości grafu. Występuje też problem w rozwijaniu „dużych" pętli – zawierających w sobie procedury komunikacji i synchronizacji. Interesujące byłoby opracowanie metody umożliwiającej wykorzystanie cennych właściwości sieci PERT przy połączeniu ich z modelem zapewniającym efektywniejszą reprezentację pętli.

8. Bibliografia

[Abd03] – Abdelkader Y. H. – Erlang distributed activity times in stochastic activity networks – Kybernetika, 2003, Vol.39, No 3, str. 347-258.

[Abd04] – Abdelkader Y. H. – Evaluating project completion times when activity times are Weibull distributed – European Journal of Operational Research, 2004, Vol. 157, No 3, Sept., str. 704-715.

[Ajm95] – Ajmone M., Balbo G., Conte G. – Modelling with generalised stochastic Petri nets – Willey & Sons, 1995.

[Alp97] – Alpan G., Jafari M. – Dynamic Analysis of Timed Petri Nets: A Case of Two Processes and a Shared Resource – IEEE Trans. On Robotics and Automation 1997, str. 338-346.

[Amm85] – Analysis of the generalized stochastic Petri nets by state aggregation – Proc. Int. Workshop on timed Petri Nets, str. 88-95, IEEE Computer Soc. Press 1985.

[Ave85] – Aven T. – Upper (lower) bounds on the mean of the maximum (minimum) of a number of random variables – J. Appl. Prob. Vol. 22 1985, str. 723-728.

[Bal95] – Baldoni R., Raynal M., - A graph based characterization of communication modes in distributed executions – Journal of Foundations of Computing and Decision Sciences 25(1) 1995 str. 3-12.

[Bau96] – Bause F., Kritzinger P. – Stochastic Petri Nets – an introduction to the theory – Vieweg Publ. 1996.

[Ben89] – Ben-Ari M. – Podstawy programowania współbieżnego – Warszawa 1989.

[Bra86] – Brauer W., Reisig W., Rozenberg G. – Petri nets: Applications and relationships to other models of concurrency – Advances in Petri Nets 1986.

[Bie01] – Biernat J. – Architektura Komputerów – Oficyna Wyd. Politechniki Wrocławskiej – Wrocław 2001.

[Bre94] – Bremaud P. – An Introduction to Probabilistic Modeling – Springer-Verlag 1994.

[Buc95] – Buchholtz P. – Hierarchical Markovian models: symmetries and reduction – Performance Evaluation, vol. 22, no. 1, 1995 str. 93-110.

[Buc97] – Buchholtz P. – Hierarchical structuring of superposed GSPNs – Proc. 7-th Int. Workshop on Petri Net and Performance Models str. 166-181, St Malo 1997.

[Car88] – Carller J., Chretlenne P. – Timed Petri Net Schedules – Advances in Petri nets 1988, str. 62-84.

[Cin81] – De Cindio F., De Michelis G., Pomello L., Simone C. – A Petri Net Model for CSP – Proc. CIL ’81 str. 392-406, Barcellona 1981.

[Col98] – Colouris G., Dollimore J., Kindberg T. – Systemy rozproszone, podstawy i projektowanie. – WNT 1998 Warszawa.

[Coo83] – Coolahan J., Roussopoulos N.– Timing requirements for time-driven systems using augmented Petri nets – IEE Trans. Soft. Eng. 5/1983 str. 603-616.

[Dav04] – David L., Punaut I., – Static Determination of Probabilistic Execution Times – Proceeding of 16th Euromicro Conference on Real Time Systems str. 223-230 – Catania 2004– IEEE 2004

[Deg88] – Degano P., Gorrieri R., Marchetti S. - A CSP process as a Condition/Event System - Advances in Petri nets 1988 str. 85-105 – Springer-Verlag 1988.

[Dod84] – Dodin B. – Determining k most critical paths in PERT networks – Oper. Res. 32, str. 859-877 - 1984.

[Dod85] – Dodin B. – Approximating the distribution functions in stochastic networks – Computers & Oper. Res. 12, str 251-264 - 1985.

[Dod85a] – Dodin B. – Bounding the project completion time distribution in PERT networks – Oper. Res. 33, str. 862-881, 1985.

[Elm89] – Elmaghraby S. – The estimation of some network parameters in the PERT model of activity networks: review and critique – Adv. in Project Scheduling , Chapter 1, Elsevier 1989.

[Fis83] – Fisher D., Goldstein W. – Stochastic PERT networks as models of cognition: derivation of the mean, variance and distribution of the reaction time using order of processing diagram – Computers & Oper. Res. 1983.

[Fis85] – Fisher D., Goldstein W. – Stochastic PERT networks: OP diagrams, critical paths and the project completion time – Computers & Oper. Res. 12, str. 811-815, 1985.

[Fly95] – Flynn M. – Computer Architecture: Pipelined and Parallel Processor Design – Jones and Bartlett Publishers, 1995.

[Gaj96] – Gajek L. – Wnioskowanie statystyczne: Modele i metody – WNT 1996.

[Ger96] – Gerbessiotis A., Siniolakis C. – Communication efficient data structures on the BSP model with applications" – Oxford Univ. Technical Report, PRG-TR-13-96 May 1996.

[Ger01] – German R. – Performance Analysis of Communication Systems – Modeling with Non-Markovian Stochastic Petri Nets – Willey 2001.

[Hoa78] – Hoare C. A. R.: Communicating Sequential Processes – Communication of the ACM, 21(8), 1978.

[Ive99] – Iverson, M. A., Ozguner, F., Potter, L.C. – Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment – Heterogeneous Computing Workshop, 1999. (HCW '99) Proceedings. Eighth 12 April 1999 str. 99 - 111.

[Jar87] – Jarnicki J., Magott J. – Hierarchic performance evaluation of concurent programming using Petri nets – Proc. 4th Int. Conf. Relcomex 87, Książ Poland.

[Joh72] – Johnson N. – Distributions in statistics – John Willey and Sons 1972.

[Kam85] – Kamburowski J. – Bounds in temporal analysis of stochastic networks – Found. Contr. Eng. 1985 vol. 10 nr 4 str. 177-189.

[Kam85a] – Kamburowski J. – Normally distributed activity durations in PERT networks – J. Oper. Res. Soc. 1985 vol. 36 str. 1051-1057.

[Kam85b] – Kamburowski J. – Two point approximations for activity times in PERT network – RAIRO Rech. Oper. 1985 vol. 19 nr 3 str. 301-313.

[Kam85c] – Kamburowski J. – An upper bound on the expected completion time of PERT networks – Eur. J. Oper. Res. 1985 vol. 21 nr 2 str 206-212.

[Kam86] – Kamburowski J. – Bounds on the mean of the maximum of a number of normal random variables – Wydawnictwo PWr, Raporty instytutu Org. 1986.

[Kam87] – Kamburowski J. – Drogi i przepływy ekstremalne w sieciach stochastycznych – Wydawnictwo PWr 1987.

[Kam92] – Kamburowski J. Bein W. Stallmann M. – Optimal Reduction of Two-Terminal Directed Acyclic Graphs – SIAM Jour. on Computing 21, 1992, str. 1112-1129.

[Lin98] – Lindemann C – Performance modelling with deterministic and stochastic Petri nets – Willey & Sons 1998

[Lin99] – Lindemann C, Thummler A – Transient Analysis of Deterministic and Stochastic Petri Nets with Concurent Deterministic Transitions – Performance Evaluation, Vol. 36&37, str. 35-54 1999.

[Mag84] – Magott J. – New NP–complete problems in performance evaluation of concurrent systems using Petri nets – IEEE transactions on Soft. Eng 1987 Vol 13 str. 578-581.

[Mag89] – Magott J. – Sieci Petriego w ocenie wydajności systemów komputerowych – Wydawnictwo Politechniki Wr. 1989.

[Mag89a] – Magott J., Skudlarski K. – Combining generalised stochastic Petri nets and PERT networks for the performance evaluation of concurrent processes – Proc. Third Int. Workshop on Petri Nets and Performance Models, str. 249-256, Kyoto Japan Dec 1989.

[Mag91] – Magott J. – Złożność obliczeniowa algorytmów i probelmów czasu cyklu dla systemów procesów sekwencyjnych z wzajemnym wykluczaniem – Arch. Autom. Robot. 1991 t.36 z. 1.

[Mag92] – Magott J. – Performance evaluation of communicating sequential processes /CSP/ using Petri nets – IEE Proc. E 1992 vol. 139 nr 3 str. 237-241.

[Mag93] – Magott J., Skundlarski K. – Estimating the mean completion time of PERT networks with exponentially distributed durations of activities – Eur. Jou. of Oper. Res. 71 (1993) str. 70-79.

[Mcc93] – McColl W. – An architecture independent programming language for scalable parallel computing – Tech. Rep. 93-072-3-9025-1, NEC Research Institute 1993.

[Mcc94] – McColl W. – Scalable Parallel Computing: A grand unified theory and its practical development – Proc of IFIP World Congress, str. 539-546, Hamburg 1994.

[Mol82] – Molloy M. – Performance analysis using stochastic Petri nets – IEEE Trans. Comput. 9/1982 str. 913-917.

[MPI95] – MPI Forum – MPI: A message Passing Interface Standard – 1995.

[Mur80] – Murata T. – Synthesis of decision-free concurrent systems for prescribed resources and performance – IEEE Trans. Soft. Eng. 6/1980 str. 525-530.

[Nol00] – Thomas Nolte T. Möller A. Nolin M. – Using Components to Facilitate Stochastic Schedulability Analysis – Proceedings of the WIP session Real Time Systems Symposium str. 7-10 – Cancun 2003 Mexico.

[Now00] – Noworyta W. – Kształt rozkładu losowego czasu realizacji programu: Systemy czasu rzeczywistego – Kraków 2000.

[Now04] – Noworyta W. – Probability Distribution Function of Time between Interrupt Requests – Proceeding of 16th Euromicro Conference on Real Time Systems (Work in Progress session) – IEEE 2004.

[Now04a] – Noworyta W. – Distribution of Time Interval between Successive Interrupt Requests – IFIP 18th World Computer Congress – DIPES 2004 str. 53-62 – Tuluza – Kulwer 2004.

[Old87] – Olderog E. – Operational Petri Net Semantics for CCSP – Advances in Petri Nets 1997 str. 196-223.

[Pet62] – Petri C. A. – Kommunikation mit Automaten – praca doktroska, uniwersytet Bonn 1962.

[Pet02] – Peters S. M. – How much Worst Case is Needed in WCET Estimation? – 2nd Intl Workshop on Worst-Case Execution Time Analysis, Technical University of Vienna, Austria 2002

[Ram80] – Ramamoorthy C., Ho G. – Performance evaluation of asynchronous concurrent systems using Petri nets – IEEE Trans. Soft. Eng. 5/1980 str. 440-449.

[Sch94] – Schwartz R., Mattern F. – Detecting casual relationships in distributed computations – Distributed Computing 7(3) 1994 str. 149-174.

[Sif80] – Sifakis J., Brauer W. – Net theory and Applications – Lecture Notes in Computer Science – Springer 1980

[Tan95] – Tanenbaum A. – Distributed Operating Systems – Prentice-Hall 1995.

[Val90] – Valiant L. – A bridging model of parallel computation – Communications of the ACM 33(8) str. 103-111, 1990.

[Vin98] – Vinitotis Y. – Probability and Random Processes for Electrical Engineers – McGraw-Hill 1998.

[Wan98] – Wang J. – Timed Petri Nets – Kluwer Academic Publ. 1998.

[Xu02] – Xu C-Z. Wang L.Y. Fong N-T. – Stochastic Prediction of Execution Time for Dynamic Bulk Synchronous Computations – The Journal of Supercomputing, January 2002, vol. 21, no. 1, str. 91-103(13).

[Zam80] – Zamojski W. – Modele niezawodnościowo-funkcjonalne systemów cyfrowych ze szczególnym uwzględnieniem systemów jednoprocesorowych – Wydawnictwo Politechniki Wroc. 1980.

9. Dodatek A - Pomiar niezależności czasu realizacji bloku programu

9.1. Metody pomiaru szybkości wykonywania programu

Umieszczenie procedur pomiarowych w programie

Najprostszym sposobem pomiaru czasu wykonania poszczególnych fragmentów programu jest umieszczenie procedur pomiarowych w treści programu. Procedury te mają za zadanie uzyskać i zapamiętać wartość czasu na początku i na końcu mierzonego fragmentu kodu. Metoda ta ze względu na wnoszone błędy dobrze sprawdza się jedynie przy pomiarze długich odcinków czasu (np. całego programu). Przy pomiarze czasu wykonania małych fragmentów kodu otrzymywane są błędne wyniki ze względu na:

- zmianę treści programu przez procedury pomiarowe – konieczność wykonania dodatkowych czynności,
- zmianę długości programu i jego rozmieszczenia w pamięci, mogącą mieć wpływ na zarządzanie pamięcią przez system operacyjny,
- wywoływanie dodatkowych funkcji systemowych, zajmujące dużo czasu i wymagające przełączenia procesora w tryb uprzywilejowany,
- zmianę zawartości pamięci podręcznych,
- rozsynchronizowanie procesów współbieżnych,
- rozsynchronizowanie szybkości procesów i urządzeń we/wy.

Umieszczenie procedur pomiarowych w systemie operacyjnym

Możliwe jest umieszczenie funkcji pomiarowych wewnątrz systemu operacyjnego. Może on zapisywać czas każdego wywołania funkcji systemowej, co daje obraz postępu prac badanego programu. Dodatkowo może on zapamiętywać adres przerwanej instrukcji podczas każdego przerwania zegarowego. Opisana metoda w mniejszym stopniu zakłóca mierzony obiekt ponieważ:

- nie generuje dodatkowych wywołań funkcji systemowych ani przerwań.
- nie zmienia treści ani długości programu.
- wyniki pomiarów zapisywane są w innym obszarze pamięci niż dane programu.

Modyfikacja systemu operacyjnego jest zdecydowanie trudniejsza od wprowadzenia procedur pomiarowych do programu. Wymaga bardzo dobrej znajomości systemu, posiadania uprawnień i narzędzi do jego modyfikacji oraz liczenia się z koniecznością restartów maszyny i przeinstalowywania oprogramowania.

Rozbudowa sprzętu o możliwości pomiarowe

Pomiaru szybkości wykonywania programu można dokonywać za pomocą specjalnie do tego przeznaczonych modułów sprzętowych. Mają one za zadanie zaobserwować moment występowania charakterystycznych fragmentów programu i zapamiętać czas ich wystąpienia. Wynika z tego konieczność posiadania przez moduł pomiarowy własnej pamięci (żeby nie zakłócać pracy mierzonego programu dostępem do pamięci). Dużym problemem jest wychwycenie przez moduł faktu dojścia badanego programu do charakterystycznego punktu. Karty rozszerzeń podłączane poprzez szyny ISA lub PCI są w stanie obserwować jedynie aktywność we/wy badanego programu. W celu dokonania pomiaru czasu wykonywania procedur obliczeniowych należy

umieścić pomiędzy nimi specjalne instrukcje we/wy (co jest modyfikacją programu wpływającą na jego szybkość). Produkuje się moduły pomiarowe nadzorujące bezpośrednio komputery jednoukładowe, jednak skonstruowanie układu rejestrującego pracę współczesnych, szybkich procesorów wiąże się z bardzo dużym nakładem kosztów.

Użycie maszyny wirtualnej

W celu określenia szybkości wykonywania niewielkich fragmentów programu można posłużyć się maszyną wirtualną emulującą działanie rzeczywistego systemu. Wymagana jest przy tym bardzo wierna symulacja, uwzględniające wszystkie aspekty mające wpływ na czas realizacji instrukcji. Użycie maszyny wirtualnej ma następujące zalety:

- możliwość określenia czasu realizacji programu dla systemu niedostępnego na rynku,
- możliwość wykorzystania projektu procesora (systemu) jako podstawy do konstrukcji maszyny wirtualnej,

Wadami takiego rozwiązania są:

- mała szybkość maszyny wirtualnej – konieczność długiego oczekiwania na wyniki,
- mała wierność symulacji maszyn wirtualnych optymalizowanych pod kątem szybkości symulacji.

Wybrana metoda pomiaru

Pomiaru czasu wykonania badanej procedury dokonano przez obliczenie różnicy czasu jej początku i końca. Jako źródło aktualnego czasu wykorzystano funkcję *gettimeofday,* wchodzącą w skład podstawowych funkcji UNIXa. Przeprowadzając pomiary przy użyciu komputerów wyposażonych w procesor typu Pentium wykorzystano licznik cykli maszynowych umożliwiający uzyskanie największej możliwej rozdzielczości pomiaru, jak również uniknięcie wywołania funkcji systemowej.

Ponieważ procedura pomiarowa sama w sobie stanowi fragment programu, zatem wynikiem pomiaru jest suma czasów realizacji badanego programu i procedury pomiarowej. Dodatkowy narzut czasowy związany z dokonywaniem pomiarów musi zostać uwzględniony podczas opracowywania wyników analizy i porównania ich z przewidywaniami teoretycznymi.

Aby określić rozkład losowy czasu realizacji danej procedury przeprowadzono wielokrotny pomiar w ramach jednego uruchomienia programu testowego. Wyniki pomiarów były zapisywane do tablicy. Po zakończeniu wszystkich pomiarów następowało formatowanie wyników i zapis na dysk. Wymuszono również wpisanie całej tablicy do pamięci podręcznej przed rozpoczęciem pomiarów. Udało się dzięki temu zmniejszyć błąd wprowadzany przez same procedury pomiarowe.

9.2. Zależność czasu realizacji od historii procesu

W celu określenia wpływu uprzednio wykonanych czynności na czas realizacji bloku programu wykonano po 4000 pomiarów czasu realizacji bloku. Wybrano trzy rodzaje bloków testowych:

- Blok zawierający obliczenia na rejestrach (obliczanie ciągu $a_n=a_{n-1}+a_{n-2}$)
- Blok przetwarzający obszar pamięci (sortowanie)
- Blok przeglądający obszar pamięci (duży)

Na początku pomiaru wykonywana jest procedura usuwania kodu i danych badanego programu z pamięci podręcznej. Ze względu na kompatybilność i ograniczenia uprawnień nie użyto rozkazów sterujących pamięcią, lecz wykonano procedurę, która posiadając długi kod oraz operując na dużej ilości danych wypełniła cache

„swoimi" wartościami. Następnym krokiem jest wykonanie bloku tworzącego historię procesu. Wybrano fragmenty o następującej charakterystyce:

- Pusty blok
- Blok obliczania wyznacznika i drukowania wyników
- Blok taki sam jak następny (mierzony)

Pomiarów dokonano przy użycia komputera z procesorem Intel-Pentium pracującego pod kontrolą systemu Linux. Podsumowanie wyników (w μsec) przedstawiono w tabeli 10-1. Kolejne kolumny zawierają czasy dla trzech bloków poprzedzających. Kolejne wiersze reprezentują fragmenty następujące w drugiej kolejności, których czas jest mierzony. Dla każdego cyklu pomiarów obliczono wartość średnią oraz odchylenie standardowe.

Tabela 9-1 Zależność czasu realizacji bloku od rodzaju bloku go poprzedzającego

	brak		wyznacznik		ten sam kod	
	średnia	odch. std.	średnia	odch. std.	Średnia	odch. std.
ciąg	5380	392	5346	418	5288	215
sortowanie	5893	815	5913	780	5882	419
przegląd	7483	593	7531	654	6676	354

Weryfikacja hipotezy H_0 o niezależności wartości średniej rozkładu losowego czasu realizacji bloku od rodzaju bloku go poprzedzającego, przeciwko hipotezie alternatywnej, że wartość średnia jest zależna:

Na podstawie nierówności Czebyszewa określono przedziały ufności wartości średniej na poziomie 95%. Nie czyniono żadnych założeń co do kształtu rozkładu losowego.

$$p\left(\overline{X} - \varepsilon < X_{śr} < \overline{X} + \varepsilon\right) \geq 1 - \frac{\sigma^2}{n\varepsilon^2} \tag{9-1}$$

Jeżeli przedziały ufności mają część wspólną to nie ma podstaw do odrzucenia hipotezy o równości średnich, w przeciwnym przypadku należy hipotezę odrzucić.

Tabela 9-2 Przedziały ufności 95% dla wartości średniej

	brak		wyznacznik		ten sam kod	
	min	max	min	max	Min	Max
Ciąg	5352	5408	5316	5376	5273	5303
Sortowanie	5835	5951	5858	5968	5852	5912
Przegląd	7441	7525	7485	7577	6651	6701

Wniosek 1:

Przedziały w trzecim wierszu tabeli 10-2 są rozłączne, zatem z prawdopodobieństwem co najmniej 95% można stwierdzić, że rozkłady mają różne wartości średnie, a zatem są różne.

Wniosek 2:

Przedziały w kolumnie „brak" i „wyznacznik" mają dużą część wspólną, zatem nie ma podstaw do odrzucenia hipotezy o zgodności rozkładów. Co więcej istnieje duże prawdopodobieństwo, że wartości średnie (rzeczywiste, a nie z próby) różnią w niewielkim stopniu. Ma to szczególne znaczenie wobec faktu, że przedziały ufności oszacowane przy braku informacji na temat kształtu rozkładu losowego są szersze niż dla rozkładów o kształtach normalno-podobnych [Gaj96].

Test serii Walda-Wolfowitza:

W celu stwierdzenia zgodności rozkładów o nieznanych kształtach przeprowadzono test Walda-Wolfowitza porównując parami rozkłady z każdego wiersza tabeli 10-2. Literatura [Gaj96] zaleca ten typ testu zamiast zmodyfikowanych wersji testu Chi-kwadrat do badania czy dwie próby pobrane są z tej samej populacji. Wyższość testów nieparametrycznych jest szczególnie wyraźna dla dużej liczby próbek zawierających sporadycznie bardzo duże wartości (badania kształtu rozkładu opisano w rozdziale 7).

Testujemy hipotezę H_0 o równości rozkładów przeciw hipotezie, że są one różne. Wald i Wolfowitz wykazali że dla liczby pomiarów m, n pochodzących z dwóch badanych rozkładów dążącej do nieskończoności liczba serii R ma rozkład normalny.

Dla $m/n \to \gamma$ oraz $m, n \to \infty$

$$\frac{R - \frac{2m}{1+\gamma}}{\sqrt{\frac{4\gamma m}{(1+\gamma)^3}}} \cong N(0,1) \tag{9-2}$$

Dla jednakowej liczby pomiarów

$$\frac{R - m}{\sqrt{\frac{1}{2}m}} \cong N(0,1) \tag{9-3}$$

Zbiory krytyczne przy m=n=4000 na poziomach ufności 95% i 99% mają postać

W(95%)=[2, 3926]

W(99%)=[2, 3895]

Tabela 9-3 Liczba serii przy porównaniu dwóch prób po 4000 pomiarów każda

	brak & wyzn.	brak & ten sam kod	wyzn. & ten sam kod
Ciąg	4012	3942	3989
Sortowanie	3933	**3901**	4012
Przegląd	4032	**1614**	**1433**

W tabeli 9-3 przedstawiono obserwowaną liczbę serii w teście Walda – Wolfowitza. Pogubioną czcionką zaznaczono wartości przemawiające za odrzuceniem hipotezy o równości rozkładów. Uzyskane wyniki pokrywają się w dużym stopniu z wynikami uzyskanymi na drodze analizy przedziałów ufności.

Tabela 9-4 Test zgodności Walda – Wolfowitza dla różnych systemów

system	program	brak & wyzn.	brak & ten sam kod	wyzn. & ten sam kod
Linux-486	ciąg	+/+	+/+	+/+
	sortowanie	?/?	+/+	+/-
	przegląd	+/+	--/--	--/--
Linux-PII	ciąg	+/+	+/+	+/+
	sortowanie	+/+	+/-	--/--
	przegląd	--/--	--/--	--/--
Linux-Embeded	ciąg	?/?	+/+	+/+
	sortowanie	+/+	+/?	+/+
	przegląd	+/+	-/-	+/?
Win95-PII	ciąg	+/-	+/+	+/+
	sortowanie	+/+	?/?	+/+
	przegląd	?/?	--/--	--/--
WinMe-PII	ciąg	+/?	+/+	+/-
	sortowanie	+/-	-/-	--/--
	przegląd	-/-	--/--	--/--
DOS-486	ciąg	+/+	+/+	+/+
	sortowanie	+/+	+/+	+/?
	przegląd	+/+	--/--	+/-
DOS-486 bez cache	ciąg	+/+	+/-	+/+
	sortowanie	+/+	+/+	+/+
	przegląd	?/?	+/-	+/+

W tabeli 9-4 przedstawiono wyniki porównań testem Walda – Wolfowitza dla różnych systemów komputerowych. W każdym polu pierwsza pozycja odpowiada poziomowi ufności 99% druga 95%. Symbole oznaczają:

+ brak podstaw do odrzucenia hipotezy o równości rozkładów

- należy odrzucić hipotezę o równości

-- zdecydowanie należy odrzucić (ufność powyżej 99,999%)

? pomiary niejednoznaczne

Na podstawie wyników przedstawionych w tabeli 9-4 można wyciągnąć następujące wnioski:

- Poszczególne pomiary dokonane w tych samych warunkach różnią się znacznie, co utrudnia wyciąganie kategorycznych wniosków co do zgodności rozkładów (dużo znaków zapytania). Zdecydowano się nie cenzurować wyników pomiarów, ze względu na niedostępność lub znaczne skomplikowanie testów statystycznych dla ocenzurowanych danych.
- Tabela posiada 63 pola zatem przy testach na poziomie ufności 95% w statystycznie trzech polach zamiast plusa błędnie jest wpisany minus. Ryzyko odrzucenia prawdziwej hipotezy jest bowiem 5%.

- Obserwowane są znaczne zakłócenia pomiarowe prowadzące czasem do wniosku, że dwa pomiary tego samego procesu, w (pozornie) tych samych warunkach mają różne rozkłady losowe. Zwiększanie ilości prób nie daje pożądanych rezultatów, zatem przyczyna leży prawdopodobnie w powolnej zmianie parametrów systemu wykonującego program. Wyniki porównań wyników z tych samych doświadczeń nie są zawarte w tabeli.
- Program przeglądania pamięci jest zdecydowanie najbardziej czuły na kod jaki go poprzedza. Dotyczy to szczególnie systemów z rozbudowaną pamięcią podręczną. Potwierdza to wpływ współczynnika trafień pamięci podręcznej na szybkość wykonywania kodu. Dotyczy to w szczególności pamięci danych, gdyż w zastosowaniach obliczeniowych (czasowo krytyczny) kod ma wielokrotnie mniejszy od rozmiar od przetwarzanych danych.
- Czas realizacji procedury liczenia ciągu jest najmniej zależny od poprzedzających go procedur, jednakże obserwuje się ponadprzeciętną korelację. Poza współczynnikiem trafień pamięci podręcznej danych istnieją zatem jeszcze inne przyczyny zróżnicowania rozkładów w zależności od poprzedzającego kodu. Może być to współczynnik trafień pamięci kodu, sposób rozmieszczenia programu w pamięci, ...

Wpływ zależności na analizę powiązanych bloków programu:

Na podstawie powyższych pomiarów można stwierdzić, że istnieje zauważalny wpływ historii programu na czas wykonywania bloku jedynie wtedy, gdy poprzednie czynności w istotny sposób wpływają na współczynnik trafień pamięci podręcznej. W szczególności dotyczy to operacji na dużych strukturach danych. Stosunkowo mały jest wpływ współczynnika trafień pamięci kodu. Wiąże się to ze strukturą badanych fragmentów. Każdy z nich zawiera pętlę niewielkiej długości, zatem po pierwszym wykonaniu pętli wszystkie jej instrukcje znajdują się już w cache pierwszego poziomu lub nawet kolejce procesora.

Proponowane w rozdziale 2 i 3 metody grupowania instrukcji, że podział programu na fragmenty poddawane analizie będzie w dużym stopniu odzwierciedlał logiczną strukturę programu. Można zatem przyjąć, że operacje dotyczące tej samej struktury danych zostaną zgrupowane w jednym bloku. Wykonanie następujących po sobie bloków wymagać będzie dostępu do innych obszarów pamięci, zatem korelacja wynikająca z zawartości pamięci podręcznej pierwszego poziomu powinna być niewielka. Po redukcji grafu poszczególne bloki przedzielone będą instrukcjami komunikacji i synchronizacji. W wielu systemach są to złożone procedury zapewniające jako efekt „wyczyszczenie" cache I poziomu a zatem zredukowanie korelacji z poprzednim blokiem.

Należy zatem uznać za słuszne przyjęte założenie o niezależności czasu realizacji poszczególnych bloków programu przyjęte w rozdziale 2.

9.3. Zależność czasu realizacji od czynności wykonywanych przez inne procesory

W celu określenia wpływu innych procesów na czas realizacji bloku programu wykonano po 4000 pomiarów czasu realizacji bloku. Wybrano trzy rodzaje bloków testowych:

- Blok zawierający obliczenia na rejestrach (obliczanie ciągu $a_n=a_{n-1}+a_{n-2}$)
- Blok przetwarzający obszar pamięci (sortowanie)
- Blok wysyłający dane poprzez sieć

W czasie dokonywania pomiaru drugi procesor wykonywał zadanie mogące w mniejszym lub większym stopniu spowalniać realizację mierzonego bloku. Jako najbardziej reprezentatywne wybrano następujące zadania:

- Brak procesów użytkowych
- Obliczenia z użyciem lokalnych struktur danych
- Przetwarzanie obszaru pamięci (sortowanie)
- Przesyłanie danych przez sieć

Pomiary przeprowadzono z użyciem systemów wyraźnie różniących się architekturą, a co za tym idzie wzajemnym wpływem współbieżnych procesów. Wyniki pomiarów (w μsec) dla komputera wyposażonego w dwa procesory Intel-Pentium i wspólną pamięć RAM przedstawia tabela 9-5. Dla każdego zestawu 4000 pomiarów obliczono wartość średnią i odchylenie standardowe.

Tabela 9-5 Zależność czasu realizacji bloku od czynności wykonywanych współbieżnie

	brak		lokalne		sort.		sieć	
	średnia	odch.	średnia	odch.	średnia	odch.	średnia	odch.
ciąg	15360	4392	14788	5618	15531	8113	15611	7632
sort.	23312	12871	24154	8561	27615	15478	25118	11697
sieć	26741	14587	27814	15417	28963	12874	32687	28644

Weryfikacja hipotezy H_0 o niezależności wartości średniej rozkładu losowego czasu realizacji bloku od rodzaju bloku go poprzedzającego, przeciwko hipotezie alternatywnej, że wartość średnia jest zależna:

Na podstawie nierówności Czebyszewa (9-1) określono przedziały ufności wartości średniej na poziomie 95%. Nie czyniono żadnych założeń co do kształtu rozkładu losowego.

Jeżeli przedziały ufności mają część wspólną to nie ma podstaw do odrzucenia hipotezy o równości średnich, w przeciwnym przypadku należy hipotezę odrzucić.

Tabela 9-6 Przedziały ufności dla wartości średniej na poziomie 95%

	brak		lokalne		sort.		sieć	
	min	max	min	max	min	max	min	max
ciąg	15049	15670	14391	15185	14957	16104	15071	16151
sort.	22402	24222	23549	24759	**26520**	**28710**	24291	25945
sieć	25709	27773	26724	28904	28052	29873	**30659**	**34714**

W pierwszym wierszu tabeli przedziały ufności mają część wspólną, zatem nie ma podstaw do odrzucenia hipotezy o równości rozkładów. Program liczący elementy ciągu nie używa żadnych współdzielonych zasobów (sam program ma kilka bajtów) zatem wpływ równoległych procesów powinien być znikomy.

W przypadku sortowania (konflikt dostępu RAM) i komunikacji (wspólny interface sieciowy) obserwowane jest wydłużenia czasu realizacji, gdy współbieżny proces jest „złośliwie” dobranym procesem konkurującym w maksymalnym stopniu. W pozostałych przypadkach nie ma podstaw do odrzucenia hipotezy o równości wartości średnich.

Weryfikacja hipotezy H_0 o równości rozkładów czasu realizacji bloku, niezależnie od czynności wykonywanych równocześnie za pomocą testu Walda-Wolfowitza. Ilość serii przy porównaniu ciągów przedstawiono w tabeli 9-7.

Zbiory krytyczne przy m=n=4000 na poziomach ufności 95% i 99% mają postać

W(95%)=[2, 3926]

W(99%)=[2, 3895]

Tabela 9-7 Liczba serii przy porównaniu dwóch prób po 4000 pomiarów

	brak & lok	brak & sort	brak & sieć	lok & sort	lok & sieć	sort & sieć
ciąg	4095	4011	3933	4001	4023	3899
sort	3999	**3005**	3925	**3215**	4090	**3353**
sieć	4012	3916	**2014**	3985	**2451**	**2511**

Uzyskanie liczby serii skłaniają do odrzucenia hipotezy o równości rozkładów losowych czasów realizacji bloków przy różnych zadaniach pracujących równolegle. Można zauważyć szczególnie wyraźną nierówność rozkładów dla par w której jeden proces współbieżny nie konkuruje z mierzonym a drugi konkuruje o jakiś niepodzielny zasób. Dla procesów nie konkurujących o zasoby nie ma podstaw do odrzucenia hipotezy o braku wpływu współbieżnych procesów na szybkość wykonywania kodu.

Badane rozkłady mają dużą wartość odchylenia standardowego, a co za tym idzie szerokie przedziały ufności. Z faktu niemożności odrzucenia hipotezy nie można zatem wyciągać kategorycznych wniosków co do jej prawdziwości.

Drugim rodzajem badanej architektury jest układ samodzielnych komputerów podłączonych do wspólnego segmentu sieci ethernet 10MBps. Jeden z komputerów wykonywał proces mierzony, drugi komputer proces obciążający system. Komputery posiadały lokalne dyski z umieszczonym na nich systemem linux. Wyniki pomiarów (w μsec) przedstawia tabela 9-8. Dla każdego zestawu 4000 pomiarów obliczono wartość średnią i odchylenie standardowe.

Tabela 9-8 Zależność czasu realizacji bloku od czynności wykonywanych współbieżnie

	brak		lokalne		sort.		sieć	
	średnia	odch.	średnia	odch.	średnia	odch.	średnia	odch.
ciąg	32912	8932	30111	4643	35251	9113	33143	7932
sort.	48277	21851	49932	28227	46985	14658	48332	22934
sieć	44561	30623	42129	23772	43261	27113	50122	38871

Weryfikacja hipotezy H_0 o niezależności wartości średniej rozkładu losowego czasu realizacji bloku od rodzaju bloku go poprzedzającego, przeciwko hipotezie alternatywnej, że wartość średnia jest zależna:

Na podstawie nierówności Czebyszewa (9-1) określono przedziały ufności wartości średniej na poziomie 95%. Nie czyniono żadnych założeń co do kształtu rozkładu losowego.

Jeżeli przedziały ufności mają część wspólną to nie ma podstaw do odrzucenia hipotezy o równości średnich, w przeciwnym przypadku należy hipotezę odrzucić. Badane rozkłady mają dużą wartość odchylenia standardowego, a co za tym idzie szerokie przedziały ufności. Z faktu niemożności odrzucenia hipotezy nie można zatem wyciągać kategorycznych wniosków co do jej prawdziwości.

Tabela 9-9 Przedziały ufności dla wartości średniej na poziomie 95%

	brak		lokalne		sort.		sieć	
	min	max	min	max	min	max	min	max
ciąg	32280	33544	29783	30440	34607	35895	32582	33704
sort.	46732	49822	47936	51928	45948	48022	46710	49954
sieć	42395	46727	40448	43810	41344	45178	**47373**	**52871**

Weryfikacja hipotezy H_0 o równości rozkładów czasu realizacji bloku, niezależnie od czynności wykonywanych równocześnie za pomocą testu Walda-Wolfowitza. Ilość serii przy porównaniu ciągów przedstawiono w tabeli 9-10.

Tabela 9-10 Liczba serii przy porównaniu dwóch prób po 4000 pomiarów

	brak & lok	brak & sort	brak & sieć	lok & sort	lok & sieć	sort & sieć
ciąg	3958	3962	3951	4065	3988	4003
sort	3966	4087	4032	4027	4021	3991
sieć	4009	4106	**3008**	4101	**2300**	**2288**

Przedstawione wyniki potwierdzają, że wpływ innych procesów na szybkość realizacji kodu obserwowany jest jedynie w przypadku, gdy procesy współzawodniczą o dostęp do wspólnego, ograniczonego zasobu. Dla komputera z dwoma procesorami, krytycznymi zasobami okazują się być wspólna karta sieciowa oraz pamięć RAM (w mniejszym stopniu). W przypadku komputerów połączonych siecią o niewielkiej przepustowości krytycznym zasobem okazuje się wspólny kanał komunikacyjny. Nawet dla kombinacji dobranych tak aby uzyskać największy wpływ sąsiedniego procesu nie przekracza on 22%, dla pozostałych kombinacji jest on poniżej błędu statystycznego. Można zatem uznać, że dla typowych programów wykonywanych na maszynie równoległej wpływ poszczególnych procesorów na siebie można pominąć, o ile zapewniona jest wymagana przepustowość kanałów komunikacyjnych oraz wydajność współdzielonych urządzeń.

9.4. Korelacja czasów realizacji kolejnych bloków

Celem poniższych doświadczeń było określenie wzajemnej korelacji czasów wykonania dwóch kolejno realizowanych bloków. Na podstawie analizy działania typowego komputera można spodziewać się zarówno korelacji pozytywnej (powolne zmiany obciążenia maszyny) jak i negatywnej (przerwania zegarowe). W przeciwieństwie do poprzednich doświadczeń nie było badane jaki wpływ na czas wykonania ma rodzaj

bloku go poprzedzającego. Dla ustalonej pary bloków zmierzone zostały czasy wykonania obu i określona ich wzajemna korelacja. Przeprowadzono po 4000 pomiarów czasów realizacji dziewięciu par bloków. Doświadczenia przeprowadzano bez dodatkowego obciążania komputera innymi zadaniami (poza czynnościami systemu operacyjnego). Uzyskane współczynniki korelacji przedstawiono w tabeli 9-11.

Tabela 9-11 Współczynnik korelacji czasu realizacji bloku poprzedzającego i następującego

	ciąg	sortowanie	przegląd
Ciąg	+0,4%	-1,0%	-1,3%
Sortowanie	+1,8%	+0,9%	+1,1%
Przegląd	-1,8%	-0,4%	+2,2%

Uzyskane współczynniki korelacji można uznać za równe zero, biorąc pod uwagę duże odchylenie standardowe mierzonego czasu, jak również nietypowy kształt rozkładu czasu wykonania. Powtarzanie doświadczenia w tych samych warunkach daje za każdym razem inny wynik. Uzyskanie zarówno dodatnich jak i ujemnych wartości współczynnika korelacji jest kolejnym argumentem przemawiającym przeciwko hipotezie o istnieniu korelacji.

Medianowy test serii

W celu stwierdzenia, że próba jest prosta (czyli nie ma korelacji pomiędzy wartościami kolejnych doświadczeń) przeprowadzono medianowy test serii. Każdemu wynikowi doświadczenia przypisano wartość jeden jeżeli był większy od mediany oraz zero jeżeli był mniejszy. W otrzymanym ciągu zer i jedynek policzono ilość serii.

Testujemy hipotezę H_0 że korelacja pomiędzy kolejnymi elementami ciągu jest zerowa, przeciwko hipotezie złożonej, że jest ona dodatnia lub ujemna. Zbiór krytyczny dla próby 8000 pomiarów na poziomie 99% ma postać [2, 3884] [4116, 8000]. Zbyt mała liczba serii przemawia za dodatnią korelacją objawiającą się grupowaniem wyników (clustering). Zbyt duża liczba serii oznacza korelację ujemną objawiającą się naprzemiennym przyjmowaniem dużych i małych wartości. Prosty test serii umożliwia zbadanie jedynie korelacji pomiędzy kolejnymi wynikami. Liczbę serii dla prób jednorodnych (4000+4000 pomiarów) przedstawiono w tabeli 9-12.

Tabela 9-12 Liczba serii testu medianowego

	ciąg	sortowanie	przegląd
Ciąg	4094	-	-
Sortowanie	-	3916	-
Przegląd	-	-	3985

Uzyskanie wyniki nie dają podstaw do odrzucenia hipotezy, że próba jest prosta, czyli że nie ma korelacji pomiędzy kolejnymi wartościami czasu realizacji bloków.

Na podstawie przeprowadzonych doświadczeń można przyjąć, że czasy realizacji kolejno wykonywanych bloków są od siebie niezależne, co zdecydowanie upraszcza obliczanie rozkładu losowego czasu wykonania całego zadania.

9.5. Korelacja czasów realizacji równoległych bloków

W celu weryfikacji hipotezy o niezależności czasów realizacji współbieżnych procesów zbadano korelację czasu fragmentów jednoczenie wykonywanych przez dwa procesory. Wyniki pomiarów dla komputera wyposażonego w dwa procesory Intel-Pentium i wspólną pamięć RAM przedstawia tabela 9-13.

Tabela 9-13 Współczynnik korelacji czasów realizacji bloków współbieżnych

	ciąg	sortowanie	sieć
Ciąg	-0,7%	+1,1%	-0,4%
Sortowanie	+1,8%	+0,8%	-0,1%
Sieć	-1,8%	-1,2%	-8,9%

Wyniki pomiarów współczynnika korelacji czasów realizacji bloków wykonywanych przez dwa komputery podłączone do wspólnego segmentu sieci przedstawia tabela 9-14.

Tabela 9-14 Współczynnik korelacji czasów realizacji bloków współbieżnych

	ciąg	sortowanie	sieć
Ciąg	-0,3%	0,0%	-1,1%
Sortowanie	-1,1%	+1,6%	+2,1%
Sieć	-0,3%	-0,5%	-4,1%

Zaobserwowano również, że przy 4000 pomiarach nie uzyskuje się powtarzalnych wartości współczynnika korelacji. Współczynnik obliczony na podstawie pierwszych 2000 doświadczeń wyraźnie różni się od wyliczonego na podstawie pozostałych 2000 pomiarów. Spowodowane jest to nietypowym kształtem badanych rozkładów losowych. Otrzymane wartości współczynników korelacji można uznać za równe zero biorąc pod uwagę błąd pomiaru. Za przyjęciem hipotezy o braku korelacji przemawiają zarówno niskie wartości współczynników korelacji, jak i występowanie zarówno dodatniej jak oraz ujemnej korelacji.

10. Dodatek B - Kształt rozkładu losowego czasu realizacji programu

Opisując metodę obliczania czasu realizacji programu współbieżnego nie czyniono założeń odnośnie kształtu czy właściwości rozkładów czasu trwania operacji składowych. Po utworzeniu sieci obrazującej strukturę programu znajomość właściwości rozkładów okazuje się konieczna ze względu na:

- Wybór odpowiedniej reprezentacji rozkładu przez narzędzie analizujące. Jeżeli możliwe jest przyjęcie założenia o kształcie rozkładu opisanym jedną z funkcji używanych w statystyce, to możliwa jest reprezentacja każdego z rozkładów składowych za pomocą kilku liczb opisujących jego charakterystyczne parametry (np. średnia i wariancja dla rozkładu normalnego). Jeżeli natomiast kształt rozkładu jest nietypowy, to konieczne jest zapamiętanie go w postaci tabeli.
- Wybór odpowiedniej metody przeprowadzania obliczeń. Dla rozkładów o typowych kształtach znane są wzory umożliwiające obliczenie kształtu i parametrów rozkładu sumy wylosowanych wartości bądź też największej z wylosowanych wartości [Ave85] [Kam86] [Mag93]. Gdy kształt rozkładu jest nietypowy, konieczne jest obliczanie powyższych wartości za pomocą ogólnych wzorów rachunku prawdopodobieństwa, wymagających czasochłonnego całkowania.
- Dobranie optymalnej metody pomiaru czasu trwania czynności składowych oraz parametrów maszyny. W przypadku rozkładów losowych o znanym kształcie można skorzystać ze znanych estymatorów poszczególnych parametrów. W większości przypadków istnieją estymatory, dla których dowiedziono największą wiarygodność. Znana jest również zależność dokładności od ilości przeprowadzonych doświadczeń, co pozwala optymalnie dobrać liczbę przeprowadzonych pomiarów.
- Określenie zakresu możliwych do wylosowania wartości, co nabiera szczególnego znaczenia w systemach czasu rzeczywistego.

Badając rozkłady losowe czasów realizacji fragmentów kodu oraz całości zadania postanowiono:

- Zbadać parametry opisujące kształt rozkładu: wartości poszczególnych momentów, wzajemną relację średniej, mediany i mody (wartości najbardziej prawdopodobnej) symetrię (skośność), szpiczastość (kurtozę), zakres przyjmowanych wartości (rozstęp), zakres wartości najbardziej prawdopodobnych (rozstęp kwartyli).
- Zbadać możliwość przybliżenia rozkładu rozkładem normalnym lub wykładniczym.
- Określić, która z typowych funkcji kształtu rozkładu najlepiej przybliża rzeczywisty rozkład.
- Zbadać czy rozkład posiada pojedynczy obszar skupienia przyjmowanych wartości czy kilka oddzielnych.
- Zbadać możliwość utworzenia modelu opisującego z zadowalającą dokładnością rzeczywisty kształt rozkładu losowego.

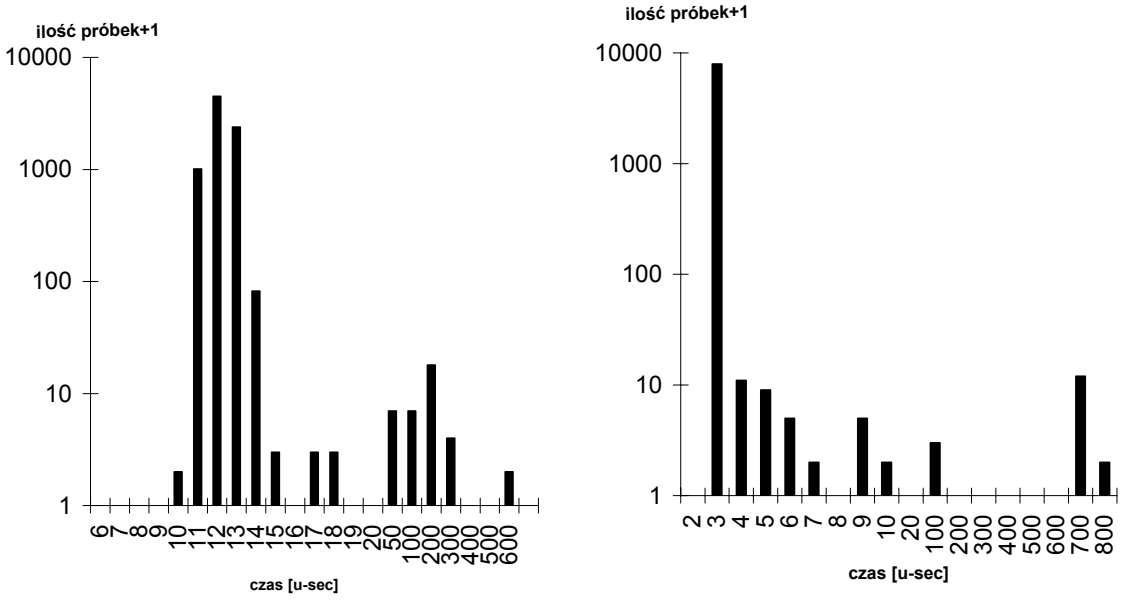
10.1. Wyniki pomiarów szybkości realizacji programu sekwencyjnego

W celu zbadania kształtu i właściwości rozkładów losowych czasów realizacji procedur sekwencyjnych przeprowadzono pomiar szybkości realizacji grup kilku instrukcji maszynowych, pojedynczych funkcji systemowych oraz dłuższych programów. Dokonując pomiarów czasu wykonywania fragmentów różniących się

złożonością o kilka rzędów wielkości dążono do uzyskania w miarę możliwości pełnego obrazu możliwych rozkładów losowych czasu realizacji kodu.

Szybkość realizacji funkcji systemowej gettimeofday

Zbadano program wykonujący 2000 razy procedurę pomiarową *gettimeofday*, a następnie drukujący wyniki. Wykonując pomiar uruchomiono cztery razy program testowy otrzymując łącznie 8000 pomiarów. Pomiary przeprowadzono oddzielnie dla dwóch różnych komputerów pracujących pod kontrolą systemu klasy UNIX. Maszyna o nazwie *monster* jest wyposażona w procesor Intel-486 i system Debian-Linux, natomiast maszyna *cyber* wyposażona w procesor Sparc II i system Solaris. Histogram wyników jest przedstawiony na wykresach 10-1 i 10-2. Ze względu na duże skupienie wyników w pobliżu maksimum gęstości rozkładu i mniejsze w pozostałym obszarze ilości próbek przedstawiono w skali logarytmicznej. Wystąpił przy tym problem reprezentacji wartości zero czyli braku próbek w danym przedziale. Aby zachować czytelność wykresu zdecydowano się na dodanie do każdej wartości liczby jeden. Brak próbek w przedziale reprezentowany jest zatem przez słupek zerowej wysokości, mimo, że opisany na osi liczbą jeden. Skala dla osi czasu również nie jest liniowa, aby umożliwić przedstawienie na wykresie dodatkowego, odległego maksimum gęstości.



Rys. 10-1. Histogram czasu wykonania pustej pętli dla komputera monster i cyber

W celu zbadania właściwości rozkładów losowych czasu wykonywania funkcji pomiarowej obliczono statystyki opisowe określające położenie rozkładu.

- Wartość minimalna – czyli najmniejszy zaobserwowany wynik pomiaru.
- Wartość maksymalna – czyli największy zaobserwowany wynik pomiaru.
- Wartość średnia – liczona jako średnia arytmetyczna wszystkich pomiarów.
- Wartość średnia wewnętrzna – liczona jako średnia arytmetyczna z 98% pomiarów. 1% pomiarów o największych wartościach oraz 1% pomiarów o największych wartościach odrzuca się. Średnia wewnętrzna obliczana jest w celu zbadania wpływu mało prawdopodobnych zakłóceń o wartościach znacznie odbiegających od średniej. Mogą mieć one duży wpływ na wartość średniej, jednak nie powinny mieć

wpływu na wartość średniej wewnętrznej, gdyż zostaną odrzucone (o ile ich liczba nie przekroczy 1% pomiarów).

- Mediana (kwantyl 50%) – stanowi liczbę, od której połowa zmierzonych wartości jest większa, a połowa mniejsza. Nie ma przy tym znaczenia o ile wyniki pomiarów są większe lub mniejsze.
- Moda – oznacza wartość najbardziej prawdopodobną. W przypadku danych doświadczalnych wartość jaka wystąpiła najczęściej. Jeżeli rozdzielczość pomiaru jest duża w porównaniu do liczby pomiarów obliczając modę dokonuje się grupowania wyników w przedziałach o jednakowej szerokości. W przeciwnym przypadku może okazać się, że żadna ze zmierzonych wartości nie powtarza się.

Następnie obliczono parametry określające zmienność rozkładu.

- Odchylenie standardowe – obliczone zostało za pomocą wzoru na nieobciążony estymator wariancji ze skończonej ilości próbek.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}\left(t_i - \tilde{t}\right)^2}{n-1}}$$

- Odchylenie średnie – obliczane jako średnia arytmetyczna wartości bezwzględnych różnicy pomiędzy wynikami a wartością średnią.

$$odch.średnie = \frac{\sum_{i=1}^{n}\left|t_i - \tilde{t}\right|}{n}$$

- Rozstęp kwartyli – obliczany jako różnica pomiędzy kwantylem 75% a kwantylem 25%. Wartość ta określa szerokość obszaru w którym mieści się środkowe 50% wyników pomiarów.

W celu określenia kształtu rozkładu obliczono:

- Skośność – jest miarą asymetrii kształtu rozkładu. Oblicza się ją na podstawie trzeciego momentu według wzoru:

$$Skos = \frac{nM_3}{(n-1)(n-2)\sigma^3}$$

gdzie:

M_3 – trzeci moment rozkładu

σ^3 – odchylenie standardowe próbki podniesione do trzeciej potęgi

n – liczba próbek

Dla rozkładów symetrycznych (w tym normalnego) wartość skośności jest równa zero. Dodatnia skośność świadczy o szybkiej lewostronnej a wolnej prawostronnej zbieżności do zera funkcji gęstości.

- Kurtoza – jest miarą szpiczastości rozkładu. Jest ona obliczana na podstawie czwartego momentu według wzoru:

$$Kurt = \frac{n(n-1)M_4 - 3M_2M_2(n-1)}{(n-1)(n-2)(n-3)\sigma^4}$$

gdzie:

M_4 – czwarty moment rozkładu

M_2 – drugi moment rozkładu

σ^4 – odchylenie standardowe próbki podniesione do czwartej potęgi

n – liczba próbek

Dla rozkładu normalnego wartość kurtozy jest równa zero. Dla rozkładów bardziej „szpiczastych" (np. wykładniczego) przyjmuje wartości dodatnie, dla bardziej płaskich wartości ujemne.

Parametry rozkładów uzyskanych wyników przedstawione zostały w tabeli 10-1. W pierwszych kolumnach tabeli przedstawiono wartości parametrów określających położenie rozkładu. Wartości minimalna i maksymalna stanowią ograniczenie zakresu wylosowanych wartości. Przedstawiono wartość średniej arytmetycznej oraz średniej arytmetycznej po odrzuceniu 1% największych i najmniejszych próbek. Wartości mediany oraz mody (maksimum prawdopodobieństwa) należą również do parametrów opisujących położenie centrum rozkładu. W dalszej części przedstawiono parametry opisujące zmienność rozkładu: odchylenie standardowe, odchylenie średnie oraz rozstęp kwartyli (zakres w jaki zawiera się środkowe 50% próbek). Ostatnie dwie kolumny przedstawiają wartości parametrów opisujących kształt rozkładu.

Tabela 10-1 Czas realizacji samej procedury pomiarowej *gettimeofday*

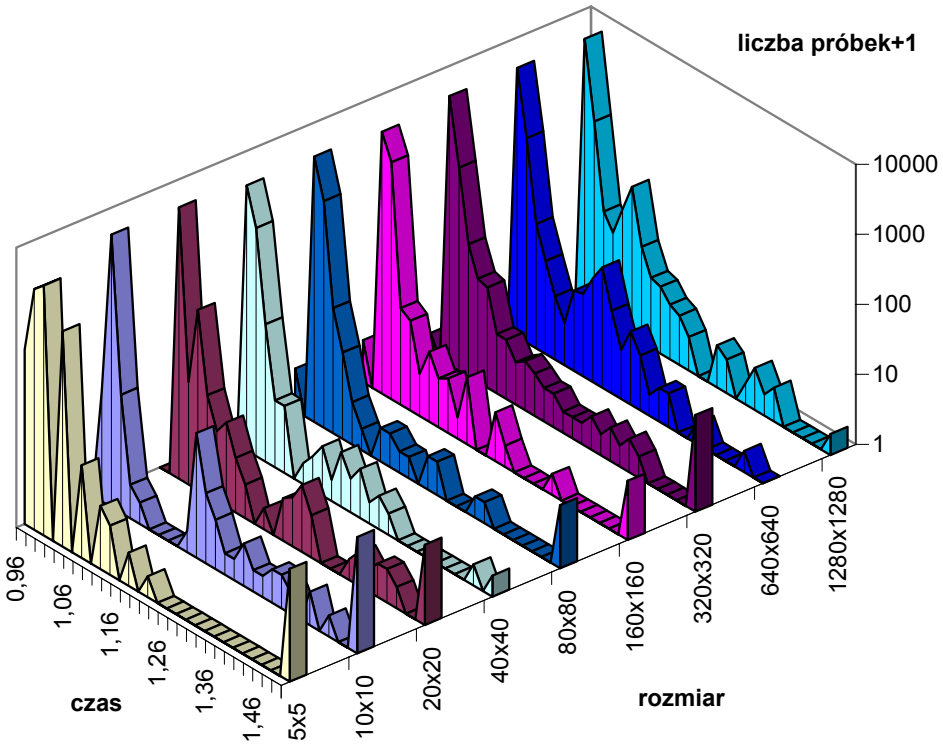
	Min.	Max.	Śred.	Śr. 98%	Med.	Moda	σ	Od. śr.	R. kw.	Skos.	Kurt.
Monster	11	571	12,66	12,19	12	12	9,70	1,16	1	34,25	1592
Cyber	3	701	4,02	3,00	3	3	25,77	2,04	0	25,71	661

Sekwencyjna symulacja gry „life"

W celu zbadania kształtu i właściwości rozkładu losowego czasu realizacji krótszego i dłuższego programu zmierzono czas realizacji programu realizującego algorytm gry „life". Przeprowadzono testy dla rozmiarów tablicy (5x5, 10x10, 20x20, 40x40, 80,x80, 160x160, 320x320, 640x640, 1280x1280). Dla każdego rozmiaru zagadnienia czterokrotnie uruchamiano program, uzyskując razem 8000 pomiarów. Program testowano na komputerze monster.iic.pwr.wroc.pl. w godzinach małego obciążenia.

Aby zapewnić czytelność wykresu, dla każdego rozmiaru zagadnienia obliczona została wartość średnia, a następnie wyniki pomiarów zostały podzielone przez tą wartość. Dzięki temu zarówno wykresy czasów realizacji złożonego zagadnienia jak i prostego można przedstawić za pomocą tej samej skali. Umożliwia to łatwiejsze porównanie kształtów wykresów.

Histogramy rozkładów losowych czasu realizacji gry „life" zostały przedstawione na wspólnym wykresie. Ze względu na znaczne różnice ilości próbek dla różnych przedziałów oś pionowa (ilości próbek) przedstawiona jest w skali logarytmicznej. Oś czasu opisana jest wartościami względnymi. Jedynka oznacza wartość czasu równą średniej. Ostatni punkt każdego wykresu oznacza liczbę próbek większych niż zakres prezentowany na wykresie. Wykres znajdujący się w lewej dolnej części odpowiada rozmiarowi 5x5. Wykres w prawej górnej części rozmiarowi 1280x1280.



Rys. 10-2. Histogram szybkości wykonywania kroku symulacji „life”

Parametry rozkładów czasu realizacji programu life przedstawione zostały w tabeli 10-2. Wartości czasów dla małych rozmiarów zagadnienia podane są w mikrosekundach, dla większych zagadnień w tysiącach i milionach mikrosekund.

Tabela 10-2 Parametry rozkładów losowych czasów realizacji sekwencyjnego programu „life”

Rozm.	Min.	Max.	Średnia	Śr. 98%	Mediana	Moda	σ	Odch. śr.	R. kwart.	Skośność	Kurtoza
5	26	227	28,10	27,74	28	28	6,61	1,01	1	21,76	529
10	112	638	114,6	113,4	113	113	16,43	2,61	1	18,88	432
20	541	1128	546	544	544	543	19,44	4,57	1	17,03	359
40	2360	5188	2374	2371	2364	2362	45,75	15,76	7	34,62	1875
80	10,6k	75,0k	10,6k	10,6k	10,6k	10,6k	862	57,2	39	68,79	5111
160	43,0k	150k	43,7k	43,6k	43,6k	43,4k	2937	371	527	35,01	1256
320	169k	360k	180k	179k	179k	178k	7021	1400	454	15,44	264
640	718k	1070k	732k	731k	729k	728k	18,6k	5877	1046	7,73	68,7
1280	2,89M	4,76M	2,92M	2,92M	2,91M	----	66,7k	22,1k	4294	9,89	152

Na podstawie uzyskanych parametrów czasów realizacji funkcji *gettimeofday* oraz programu „life” można określić następujące właściwości rozkładów:

- Wartości średniej, mediany i mody są bardzo bliskie wartości minimalnej. Świadczy to o skupieniu centrum rozkładu w pobliżu dolnego ograniczenia zbioru wylosowanych wartości.

- Dodatnia, wysoka wartość skośności świadczy, iż wykres funkcji gęstości jest szybko zbieżny lewostronnie do zera a wolno prawostronnie. Wartości skośności przekraczające 7 świadczą o bardzo dużej asymetrii wykresu.
- Wartości średniej wewnętrznej (z 98% próbek) są mniejsze od średniej z wszystkich próbek, co potwierdza dodatnią skośność rozkładu.
- Rozstęp kwartyli jest dla wszystkich przypadków zdecydowanie mniejszy (co najmniej 3 razy) od odchylenia standardowego. Stosunek ten świadczy o skupieniu obserwowanych wartości w wąskim przedziale oraz istnieniu wartości zdecydowanie odbiegających od średniej wpływających na wysoką wartość odchylenia standardowego. Wykres jest zatem wyraźnie „szpiczasty".
- Dodatnia, wysoka wartość kurtozy również potwierdza „szpiczastość" wykresu funkcji gęstości.
- Współczynnik zmienności rozkładu (stosunek odchylenia std. do średniej) przyjmuje wartości od 6,41 do 0,023 i maleje wraz ze wzrostem długości analizowanego fragmentu programu. Tendencja spadkowa jest jednak słabsza niż $1/\sqrt{średnia}$ jakiej należy oczekiwać od doświadczenia będącego sumą wielu niezależnych doświadczeń składowych.

Przedstawione właściwości rozkładów umożliwiają wyciągnięcie następujących wniosków:

- Rozkłady losowe czasu realizacji krótkich i długich fragmentów kodu mają takie same właściwości. Wydaje się zatem możliwe stworzenie jednolitego opisu umożliwiającego reprezentację rozkładu losowego czasu wykonywania procedur o różnych długościach.
- Kształt rozkładu losowego czasu realizacji fragmentu kodu zdecydowanie różni się od rozkładu normalnego.
- Rozkłady losowe czasów realizacji długich fragmentów kodu (ponad 107 instrukcji) mają wyraźnie asymetryczny kształt mimo, że na podstawie centralnego twierdzenia granicznego należałoby oczekiwać kształtu zbliżonego do normalnego.

Weryfikacja hipotez dotyczących kształtu rozkładu losowego

Na podstawie wartości parametrów opisujących rozkłady losowe czasów realizacji fragmentów kodu można zweryfikować niektóre hipotezy dotyczące kształtu rozkładu. Szczególnie ważne, jest stwierdzenie czy rozkład jest zbliżony do normalnego lub wykładniczego. W celu potwierdzenia powyższych hipotez przeprowadzono weryfikację na podstawie statystyk opisowych.

Badania wykazały nieprzydatność testu chi-kwadrat do celów określania kształtu rozkładu losowego. Przeprowadzenie tego testu daje odpowiedź z jakim prawdopodobieństwem dane próbki mogą pochodzić z doświadczenia losowego o podanym rozkładzie losowym. Nie odpowiada on natomiast na pytanie jak bardzo niedokładne jest przybliżanie rozkładu doświadczalnego wybraną funkcją. Dla dużej ilości próbek test chi-kwadrat jest bardzo czuły. Zmiana granic przedziałów lub ich ilości powoduje bardzo istotne zmiany wartości wyniku. Dla rozkładu 10 tys. próbek wygenerowanego sztucznie na podstawie wzorów do tworzenia rozkładu normalnego po przeprowadzeniu testu chi-kwadrat uzyskano zaprzeczenie hipotezy o normalności rozkładu. Precyzja obliczeń arytmetycznych użytych narzędzi okazuje się być niewystarczająca i powoduje bardzo poważne błędy. Większość narzędzi statystycznych jest przystosowana do wyciągania wniosków na podstawie jak najmniejszej liczby próbek. W prowadzonych badaniach uzyskiwano od 2 tys. do 10 mln. próbek co wielokrotnie prowadziło do wadliwego działania użytych programów statystycznych.

Postanowiono sprawdzić, jakie znane rozkłady losowe o pojedynczym centrum gęstości mogą być użyte do przybliżania rzeczywistych rozkładów czasu realizacji kodu.

Weryfikacja hipotezy o normalności rozkładu czasu realizacji programu.

Zakładając że kształt rozkładu losowego jest zbliżony do kształtu rozkładu normalnego o funkcji gęstości opisanej wzorem:

$$f(t) = \frac{1}{\sqrt{2\Pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$$

gdzie:

μ – wartość średnia

σ – odchylenie std.

należy oczekiwać następujących właściwości rozkładu:

- Równych wartości średniej, średniej wewnętrznej, mediany, mody
- Wartości średniej znajdującej się w połowie przedziału przyjmowanych wartości.
- Rozstępu kwartyli równego 1,35 odchylenia standardowego
- Obecności 2,5% próbek w obszarze poniżej μ-2σ i 2,5% w obszarze powyżej μ+2σ
- Zerowej wartości skośności.
- Zerowej wartości kurtozy.

Dla badanych rozkładów żaden z wymienionych postulatów nie jest spełniony (tabela 10-2) co daje pełne podstawy do odrzucenia hipotezy o kształcie rozkładu zbliżonym do normalnego. Test chi-kwadrat przyjmuje wartości przemawiające za kategorycznym odrzuceniem hipotezy o normalności rozkładu.

Weryfikacja hipotezy o wykładniczym kształcie rozkładu

Zakładając że kształt rozkładu losowego jest zbliżony do kształtu rozkładu wykładniczego o funkcji gęstości opisanej wzorem:

$$f(t) = \lambda e^{-\lambda t}$$

należy oczekiwać następujących właściwości rozkładu:

- mody równej zero.
- minimum równego zero.
- wartości średniej równej odchyleniu standardowemu (obie wartości równe 1/λ).
- wartości mody równej kresowi dolnemu zbioru przyjmowanych wartości.
- następującej relacji wartości parametrów: moda < mediana < średnia
- skośności równej +2
- kurtozy równej +6.

Trzy pierwsze postulaty dotyczące położenia centrum rozkładu nie są spełnione. Dla wszystkich przypadków wartość skośności oraz kurtozy jest zdecydowanie większa od oczekiwanej. Pozostałe natomiast związane z kształtem rozkładu zostały potwierdzone doświadczalnie (tabela 10-2). Hipotezę o wykładniczym charakterze rozkładu czasu realizacji fragmentów kodu należy zatem odrzucić.

Weryfikacja hipotezy o wykładniczym-przesuniętym charakterze rozkładu

Zakładając że kształt rozkładu losowego jest zbliżony do kształtu rozkładu wykładniczego-przesuniętego o funkcji gęstości opisanej wzorem:

$f(t) = \lambda e^{-\lambda(t-\mu)}$ dla $t \geq \mu$

$f(t) = 0$ dla $t < \mu$

należy oczekiwać następujących właściwości rozkładu:

- mody równej wartości minimalnej (stała μ).
- wartości średniej minus minimalnej równej odchyleniu standardowemu ($1/\lambda$).
- następującej relacji wartości parametrów: moda < mediana < średnia
- skośności równej +2.
- kurtozy równej +6.

Rozkład losowy czasu realizacji bloków o niewielkiej liczbie instrukcji ma modę równą wartości minimalnej, natomiast dla dłuższych bloków wartości te są różne. Różnica wartości średniej i minimalnej jest dla każdego doświadczenia wyraźnie mniejsza od odchylenia standardowego. Wartości skośności i kurtozy są zdecydowanie większe od przewidywanych. Hipoteze o wykładniczym-przesuniętym charakterze rozkładu należy zatem odrzucić.

Dobór funkcji przybliżającej gęstość rozkładu na podstawie statystyk opisowych.

Poszukując funkcji mogącej stanowić dobre przybliżenie gęstości rozkładu czasu realizacji należy kierować się następującymi wskazaniami.

- Dla stosunkowo dużego odcinka wartości czasu od zera do pewnej wartości minimalnej prawdopodobieństwo jest równe zero. Brane pod uwagę są więc jedynie takie funkcje gęstości rozkładu, które umożliwiają zdefiniowanie wartości minimalnej możliwych do wylosowania wartości.
- Wartość minimalna rozkładów jest zbliżona do mody, jednak od niej mniejsza. Funkcja przybliżająca rozkład powinna zatem posiadać maksimum dla wartości wyższych niż kres dolny wartości o niezerowym prawdopodobieństwie.
- Nie stwierdzono wyraźnej zależności pomiędzy wartością średnią a odchyleniem standardowym. Sugeruje to, że poszukiwana funkcja powinna mieć oddzielne parametry określające położenie i zmienność rozkładu.
- Rozstęp kwartyli jest mały w porównaniu do odchylenia standardowego. Maksimum gęstości, jest zatem bardzo „wysokie" i szpiczaste.
- Zaobserwowano, że dla zebranych próbek wartość kurtozy podzielona przez kwadrat skośności przyjmuje wartości z przedziału 1,024 do 1,56 mimo, że sama skośność zmienia się w zdecydowanie szerszym zakresie 7,73 do 68,79. Może to sugerować, że poszukiwana funkcja może posiadać jeden (a nie dwa) parametry określające kształt.
- Kurtoza obliczona dla próbek zebranych doświadczalnie jest (bardzo) duża, jednak skończona. Należy zatem przyjąć, że szukana funkcja gęstości rozkładu powinna mieć skończoną wartość czwartego momentu.
- Wysoka wartość skośności i kurtozy świadczy o wolnej zbieżności prawostronnej do zera funkcji gęstości rozkładu, wolniejszej niż zbieżność wykładnicza.

Poszukiwanie obszarów skupienia dla rozkładu czasu wykonywania fragmentu kodu

Przy poszukiwaniach optymalnego sposobu reprezentacji rozkładów losowych istotna jest informacja czy wszystkie obserwowane wyniki zawierają się w jednym, zwartym zakresie czasu, czy też istnieje kilka rozłącznych, oddzielonych od siebie zakresów przyjmowanych wartości. Za kryterium istnienia dodatkowego punktu skupienia przyjęto istnienie pewnego zakresu (t_a, t_b) o następujących właściwościach:

- Gęstość prawdopodobieństwa w obszarze (t_a, t_b) jest znacząco wyższa niż w jego najbliższym otoczeniu. Przyjęto, że średnia gęstość wewnątrz musi być 10 razy większa, niż dla otoczenia. Szerokość badanego otoczenia przyjęto jako równą połowie szerokości przedziału.
- Liczba próbek zawarta w dodatkowym obszarze skupienia w istotny sposób wpływa na parametry rozkładu. W przypadku systemów z miękkimi ograniczeniami czasowymi miarą istotności jest wpływ na wartość średnią. Dla systemów o twardszych wymaganiach czasowych miarą istotności może być wpływ na wartość wariancji lub też znajdowanie się przedziału o niezerowej gęstości prawdopodobieństwa powyżej zadanej wartości granicznej czasu. Dla potrzeb badania kształtu rozkładu przyjęto jako istotne skupienia wpływające na wartość wariancji w stopniu większym niż 1%.

Tabela 10-3 Dodatkowe punkty skupienia rozkładów losowych czasu realizacji fragmentu

	Punkt skupienia	Liczba próbek	Wpływ na śred.	Wpływ na VAR.
monster	52	7	0,36%	1,50%
	131	17	2,22%	30,66%
cyber	650	8	1,31%	63,28%
	699	4	0,65%	36,33%
life 5x5	49	17	0,16%	2,13%
	142	10	0,50%	36,57%
	215	5	0,42%	50,50%
life 10x10	140	85	0,31%	1,98%
	313	12	0,26%	22,23%
life 320x320	290k	22	0,18%	74,40%
life 1280x1280	3,16M	117	0,12%	18,82%

Na podstawie parametrów dodatkowych punktów skupienia rozkładów przedstawionych w tabeli 10-3 można określić następujące właściwości:

- Rozkłady losowe poza głównym punktem skupienia posiadają dodatkowe punkty skupienia w istotny sposób wpływające na parametry rozkładu.
- Dodatkowe punkty skupienia zawierają małą liczbę obserwowanych próbek, są jednak znacznie oddalone od głównego obszaru skupienia. Ponieważ wartość czasu dla pomiarów zawartych w dodatkowych punktach skupienia jest duża mają one zauważalny wpływ na wartość średnią i bardzo duży na wariancję.
- W przypadku niektórych rozkładów wartość wariancji jest w 99% determinowana przez próbki zawarte w dodatkowych punktach skupienia. Dla pomiaru szybkości komputera cyber wariancja głównego obszaru skupienia wynosi 0,0358 a całości rozkładu 664,15. Dla każdego z badanych rozkładów dodatkowe skupienia podwyższają wariancję o co najmniej 18%.

- Ze względu na małą liczbę próbek zawartych w obszarach dodatkowych skupień i ich duży wpływ na wartości parametrów rozkładu konieczne jest przeprowadzenie dużej ilości doświadczeń w celu wiarygodnego określenia kształtu i parametrów badanych rozkładów.
- Wpływ dodatkowych obszarów skupienia na wartość średnią czasu realizacji jest stosunkowo niewielki, w większości przypadków poniżej 1% zatem przy analizie systemów z miękkimi uwarunkowaniami czasowymi przybliżenie kształtu rozkładu funkcją o pojedynczym obszarze skupienia nie powinno wprowadzić znaczących błędów.
- Liczba dodatkowych punktów skupienia nie przekracza trzech (dla badanych rozkładów) zatem można przypuszczać, że dla innych rozkładów czasu realizacji kodu również będzie niewielka.
- Część punktów skupienia mieści się w wąskim zakresie o ostro zaznaczonych granicach, część posiada granice rozmyte. W przypadku nieostrych granic ustalenie ilości próbek wchodzących w skład punktu skupienia jest często kwestią umowną i zależy od przyjętego algorytmu selekcji (pojedynczego wiązania, pełnego wiązania, średnich połączeń, środków ciężkości, Warda). Przyjęcie odpowiedniego algorytmu selekcji może mieć wpływ na dokładność reprezentacji analizowanych rozkładów a co za tym idzie na wynik końcowy.

Na podstawie przeprowadzonych badań wykazano istotny wpływ dodatkowych punktów skupienia na parametry rozkładu losowego czasu realizacji fragmentu kodu. Uznano więc, że nie jest możliwe przybliżanie kształtu rozkładu poprzez prostą funkcję posiadającą pojedynczy obszar skupienia obserwowanych wartości. Jako, że prezentowane w literaturze funkcje gęstości typowych rozkładów posiadają pojedynczy obszar skupienia, nie jest możliwa reprezentacja rozkładu losowego czasu realizacji kodu za pomocą powszechnie znanego typu rozkładu losowego.

Badanie dokładności odwzorowania rozkładu przy małej ilości doświadczeń.

Podstawowym zadaniem metod statystycznych jest wyciągnięcie wniosków o jak największej wiarygodności na podstawie jak najmniejszej ilości pomiarów. W przypadku pomiaru szybkości komputera można wykonać znacznie większą ilością doświadczeń niż w przypadku badania innych obiektów. Nie zmienia to jednak faktu, że liczba pomiarów jest ograniczona przez koszt i czas konieczny do ich przeprowadzenia.

Aby zbadać możliwość określenia rozkładu losowego czasu realizacji fragmentu kodu na podstawie niewielkiej ilości doświadczeń przeprowadzono osiem doświadczeń, w jednakowych warunkach mierząc 10 tys. razy szybkość realizacji fragmentów kodu algorytmu „life”. Ponieważ badanie dotyczy tego samego obiektu, w tych samych warunkach, należy spodziewać się jednakowych rozkładów czasu realizacji kodu w każdym z doświadczeń. W tabeli 10-4 przedstawiono wartość mediany, średniej oraz odchylenia standardowego z próbek przy pomiarze czasu wykonywania obliczeń dla połowy tablicy algorytmu „life”. Dodatkowo przedstawiono wartość średniej wewnętrznej po odrzuceniu skrajnych 20% próbek. Przeprowadzono również badania szybkości przeglądania tablicy dla 10 tys. 1000 i 100 pomiarów.

Tabela 10-4. Czas wykonywania obliczeń dla pierwszej połowy tablicy 10 tys. pomiarów

	Gorg1-a	Gorg1-b	Gorg1-c	Gorg1-d	Gorg2-a	Gorg2-b	Gorg2-c	Gorg2-d
Min.	4059	4059	4059	4059	4059	4059	4059	4059
Mediana	4059	4059	4059	4059	4059	4059	4059	4059
Śr. 10/90	4061	4060	4063	4063	4214	4098	4201	4061
Średnia	4067	4065	4077	4068	4865	4329	4861	4077
Odch. S.	367	115	1009	381	59488	26614	59485	946

Tabela 10-5. Czas wykonywania obliczeń dla warunków brzegowych 10 tys. pomiarów

	Gorg1-a	Gorg1-b	Gorg1-c	Gorg1-d	Gorg2-a	Gorg2-b	Gorg2-c	Gorg2-d
Min.	1070	1070	1070	1070	1070	1070	1070	1070
Mediana	1070	1070	1070	1070	1070	1070	1070	1070
Śr. 10/90	1070	1070	1070	1071	1070	1071	1071	1071
Średnia	1071	1070	1074	1074	1070	1071	1075	1071
Odch. S.	34	21	232	197	30	41	418	37

Tabela 10-6. Czas wykonywania obliczeń dla drugiej połowy tablicy 10 tys. pomiarów

	Gorg1-a	Gorg1-b	Gorg1-c	Gorg1-d	Gorg2-a	Gorg2-b	Gorg2-c	Gorg2-d
Min.	3157	3157	3157	3157	3157	3157	3157	3157
Mediana	3157	3157	3157	3157	3157	3157	3157	3157
Śr. 10/90	3160	3161	3160	3158	3160	3178	3160	3187
Średnia	3160	3164	3160	3159	3162	3695	3160	3958
Odch. S.	75	377	62	51	209	53207	59	59486

Tabela 10-7. Czas wykonywania obliczeń przy 10 tys. pomiarów

	Gorg1-a	Gorg1-b	Gorg1-c	Gorg1-d
Min.	4040	4040	4040	4040
Mediana	4058	4052	4052	4052
Śr. 10/90	4062	4052	4051	4052
Średnia	4114	4117	4083	4079
Odch. S.	909	1307	161	246

Tabela 10-8. Czas wykonywania obliczeń przy 1 tys. pomiarów

	Gorg1-a	Gorg1-b	Gorg1-c	Gorg1-d
Min.	4043	4040	4046	4040
Mediana	4062	4052	4052	4052
Śr. 10/90	4062	4052	4051	4052
Średnia	4139	4077	4086	4076
Odch. S.	1338	148	179	152

Tabela 10-9. Czas wykonywania obliczeń przy 100 pomiarach

	Gorg1-a	Gorg1-b	Gorg1-c	Gorg1-d
Min.	4053	4046	4046	4046
Mediana	4062	4052	4052	4052
Śr. 10/90	4063	4052	4052	4052
Średnia	4084	4072	4102	4071
Odch. S.	80	70	256	70

Porównując parametry rozkładów uzyskanych w wyniku doświadczeń można zauważyć:

- Stosunkowo dużą zgodność współczynników określających centrum rozkładu (w szczególności wartości minimalnej i mediany).
- Bardzo duże różnice wartości odchylenia standardowego.
- Bardzo wyraźną zgodność wartości minimalnych.
- Wzrost odchylenia standardowego przy wzroście ilości pomiarów.

Można zatem wyciągnąć następujące wnioski:

- Dla wiarygodnego określenia wariancji rozkładu czasu realizacji fragmentu kodu konieczne jest przeprowadzenie bardzo dużej ilości doświadczeń. Wykonanie 10 tys. prób okazuje się być niewystarczające. Wcześniej wykazano, że na wartość wariancji bardzo duży wpływ mają wartości pomiarów znajdujące się w dodatkowych punktach skupienia. Ponieważ prawdopodobieństwo ich wystąpienia jest małe to zmienność wariancji jest duża. Szczególnie jest to widoczne przy 100 pomiarach, gdzie nie zaobserwowano opóźnienia o bardzo dużej wartości, więc otrzymano zdecydowanie mniejszą wartość odchylenia standardowego niż dla 10 tys. pomiarów.
- Bardziej dokładne mogą okazać się pośrednie metody obliczania wariancji rozkładu niż liczenie według definicji na podstawie pomiarów. W przypadku przybliżania kształtu rozkładu znaną funkcją w większości wypadków istnieje relacja pomiędzy wariancją a innymi parametrami opisowymi rozkładu, które można określić z mniejszym błędem.
- Wartości minimum i mediany można określić z dużym poziomem ufności przy małej liczbie prób.

W celu weryfikacji zgodności rozkładów przeprowadzono test chi-kwadrat dla czterech doświadczeń przeprowadzonych na tym samym komputerze. Dla liczby przedziałów 10-20 dobranych według kryterium porównywalnej liczności test wykazał zdecydowaną niezgodność rozkładów co jest sprzeczne z oczekiwaniami, gdyż wszystkie doświadczenia przeprowadzano w jednakowych warunkach.

W celu sprawdzenia identyczności wag dodatkowych punktów skupienia przeprowadzono test chi-kwadrat dla 3 przedziałów zawierających odpowiednio pierwszy (główny), drugi i trzeci punkt skupienia. W tabeli 10-7 przedstawiono wartości prawdopodobieństw rozkładu chi-kwadrat dla wartości statystyki obliczonej według wzoru:

$$\chi^2 = \sum_{i=1}^{3} \frac{(L_i - E_i)^2}{E_i} \qquad (10\text{-}1)$$

Poprzez L_i oznaczano liczność obserwowaną, a poprzez E_i liczność wzorcową dla danego przedziału. Porównując ze sobą dwa rozkłady uzyskane doświadczalnie jeden z nich przyjmowano jako wzorcowy porównując z drugim. W kolejnych wierszach umieszczono kolejno rozkłady porównywane, w kolumnach wzorcowe.

Tabela 10-10 Test Chi-kwadrat zgodności rozkładów przy 3 przedziałach

	Gorg1-a	Gorg1-b	Gorg1-c	Gorg1-d	średnia
Gorg1-a	1	0,920905	0,238150	0,698166	0,781587
Gorg1-b	0,919737	1	0,331452	0,871931	0,902253
Gorg1-c	0,148613	0,233141	1	0,523655	0,492655
Gorg1-d	0,679208	0,862554	0,586555	1	0,979784

Uzyskany stopień zgodności pozwala na przyjęcie hipotezy o powtarzalnym rozkładzie prawdopodobieństwa pomiędzy poszczególne punkty skupienia. Warto jednak zauważyć, iż uzyskano rozbieżności, w szczególności dla rozkładu oznaczonego „C". W połączeniu z obserwowaną niezgodnością parametrów opisowych (tabele 10-4, 10-5, 10-6) oraz niezgodnością rozkładów według testu chi-kwadrat skłaniają do orzeczenia niezgodności rozkładów czasu realizacji tego samego fragmentu kodu w tych samych warunkach. Dla doświadczenia zawierającego 10 tys. pomiarów uzyskano średnio 9340, 626 i 34 pomiary w obszarze poszczególnych punktów skupienia, zatem nie można rozbieżności wytłumaczyć zbyt małą licznością próby.

Przypuszczalnie istnieje dodatkowy parametr nie uwzględniony przez eksperymentatora, powodujący iż poszczególne doświadczenia nie były przeprowadzane w identycznych warunkach, więc wartości były losowane na podstawie innych rozkładów losowych. Przykładem dodatkowych czynników mogą być:

- Sposób umieszczenia programu i bibliotek w pamięci RAM, wpływający na efektywność pamięci podręcznych procesora. Przeprowadzenie pomiarów w krótkim odcinku czasu zapewnia ładowanie programu w ten sam obszar pamięci, zwalniany każdorazowo po zakończeniu programu. W dłuższym okresie czasu ilość i sposób fragmentacji pamięci będzie się zmieniał prowadząc do rozbieżności wyników poszczególnych doświadczeń.
- Aktywność innych programów. Część programów (systemowych) pracujących w czasie przeprowadzania doświadczeń wykazuje aktywność skupioną w odcinkach czasu dłuższych niż czas pojedynczego pomiaru, a krótszych niż czas trwania doświadczenia (10 tys. pomiarów). Jeżeli zatem doświadczenie zostanie

wykonane w trakcie wzmożonego obciążenia systemu jego rezultaty będą się różnić od doświadczenia przeprowadzonego w czasie bezczynności programów systemowych.

Powolna zmienność wydajności systemu w czasie okazuje się być ważnym elementem wpływającym na dokładność pomiaru rozkładu losowego czasu realizacji programu. Uwzględnienie fluktuacji wydajności systemu wymaga przeprowadzenia zwiększonej liczby pomiarów, rozłożonych w czasie. Zebrane rezultaty obserwacji wydajności maszyny umożliwiają uzyskanie ogólnego rozkładu losowego czasu realizacji programu, w którym powolna zmienność parametrów sprzętu jest również uwzględniona jako zjawisko losowe.

Pochłanianie dodatkowych punktów skupienia

W przypadku pomiaru bardzo małych odcinków czasu każde przerwanie systemu operacyjnego, lub inne nieoczekiwane opóźnienie jest bardzo wyraźnie widoczne, gdyż w znaczący sposób wydłuża czas realizacji badanego fragmentu. W miarę wzrostu długości badanego kodu drobne opóźnienia, występujące wielokrotnie powodują rozmycie głównego punktu skupienia. Przyjmuje on kształt zbliżony do fragmentu hiperboli i nie jest możliwe wyodrębnienie w prosty sposób składowych opóźnień w jego skład. Opóźnienia o dużych wartościach tworzą dodatkowe punkty skupienia, które jednak w miarę wzrostu długości kodu zostaną wchłonięte przez centralny obszar rozkładu.

Dla przykładu przerwanie zegara w systemie Linux-486 dla komputera monster powoduje konieczność wykonania dodatkowych 2000 cykli. Zatem przy pomiarach szybkości wykonywania kodu wymagającego poniżej 10 tys. cykli będzie ono wyraźnie obserwowalne w postaci dodatkowego punktu skupienia wyników odległego o 2000 cykli od głównego maksimum. W przypadku badania czasu realizacji dłuższych fragmentów kodu zlewa się ono jednak z głównym obszarem mierzonych wartości.

W wyniku połączenia się głównego obszaru skupienia oraz pobocznych uzyskuje się kształt gęstości rozkładu zbliżony do hiperboli. Jeżeli istnieje wiele czynników powodujących powstawanie opóźnień o różnej długości to uzyskiwana jest stosunkowo „gładka” funkcja gęstości rozkładu losowego czasu realizacji fragmentu. Jeżeli natomiast istnieje niewiele czynników to wykres gęstości zawiera wyraźne maksima lokalne.

10.2. Metody pomiaru szybkości komunikacji

W celu porównania rozkładów losowych czasu realizacji fragmentu kodu oraz czasu przesyłania informacji pomiędzy współpracującymi komputerami dokonano pomiarów szybkości przesyłania pakietu przez sieć komputerową. Różnorodność układów umożliwiających przekazywanie informacji pomiędzy procesorami (komputerami) jest zdecydowanie większa niż architektur pojedynczego komputera, toteż przeprowadzenie kompleksowych badań wykracza poza zakres i tematykę prezentowanej pracy. Przeprowadzono doświadczenia w oparciu o najbardziej rozpowszechniony typ sieci pracujący w oparciu o protokół TCP/IP. Zaproponowane metody pomiaru umożliwiają eksperymentatorowi zweryfikowanie wyników także w innej sieci. Podobnie jak w przypadku pomiaru szybkości pojedynczego procesora w zależności od dostępnych możliwości i uprawnień można dokonać pomiarów za pomocą modyfikacji programu, dodania funkcji pomiarowych do systemu operacyjnego lub zastosowania specjalistycznego sprzętu.

Modyfikacja programu

Program komunikujący się z otoczeniem można wzbogacić o dodatkowe procedury mierzące czas operacji we/wy. Możliwy jest pomiar czasu wykonania instrukcji nadawczej i odbiorczej, jak również określenie czasu pomiędzy wysłaniem zapytania a otrzymaniem odpowiedzi. Jeżeli zegary komputerów uczestniczących w komunikacji są zsynchronizowane w wymaganym stopniu to możliwe jest również określenie czasu transmisji informacji przez sieć. Modyfikując program należy mieć na uwadze, że dodatkowe procedury pomiarowe spowalniają wykonywanie właściwego programu. Zakłóca to relację pomiędzy szybkością procesora a wydajnością układów we/wy oraz wzajemną synchronizację procesów. W programach współbieżnych zmiany szybkości programów powodowane przez dodanie kodu mierzącego szybkości mają zdecydowanie większy wpływ na zafałszowanie wyniku niż w przypadku programów jednowątkowych.

Modyfikacja systemu operacyjnego

Zdecydowana większość komputerów dostarcza użytkownikowi gotowe funkcje komunikacji z otoczeniem czy to w postaci funkcji systemu, czy tez bibliotek standardowych. Praktycznie nie spotyka się programów użytkowych całkowicie samodzielnie obsługujących układy komunikacyjne. Większość z nich korzysta z funkcji systemu operacyjnego lub bibliotek standardowych. Można zatem zmodyfikować funkcje odpowiedzialne za komunikacje dodając do nich procedurę pomiaru czasu. W większości nowoczesnych systemów operacyjnych nie będzie nawet konieczna powtórna kompilacja programu użytkownika.

Dodatkowy sprzęt

Pomiar szybkości operacji komunikacyjnych przy użyciu dodatkowego sprzętu podlega podobnym zasadom jak pomiar szybkości wykonywania programu przez maszynę. Łatwiej jest podłączyć układ pomiarowy do kanału komunikacyjnego, niż do nowoczesnego procesora. Mierzone czasy zazwyczaj są większe niż czasy wykonania fragmentu kodu przez procesor. Obecnie stosunkowo niewielkim nakładem kosztów można rozbudować komputery o układy zapewniające bardzo dokładną synchronizację zegarów (DCF, GPS).

Pomiar w jednym punkcie

Pomiar opóźnień związanych z przesyłaniem danych może być dokonany w jednym punkcie jeżeli topografia sieci połączeń umożliwia obserwację nadawcy i odbiorcy przez to samo urządzenie pomiarowe. Pojęcie urządzenia pomiarowego zawiera zarówno zewnętrzne specjalizowane układy elektroniczne, jak i procedury pomiarowe wykonywane przez komputer wykonujący obserwowane procesy. Przykładem pomiaru w jednym punkcie jest pomiar czasu nadania wiadomości, przetworzenia i odpowiedzi (np. komendą ping). Pomiar w jednym punkcie jest najłatwiejszym przypadkiem, gdyż nie wymaga synchronizacji zegarów układów pomiarowych.

Wybrana metoda

Na podstawie przeprowadzonej klasyfikacji metod pomiaru szybkości komunikacji zdecydowano się wykonać doświadczenia najprostszymi dostępnymi środkami. Celem pracy jest określenie charakteru i właściwości statystycznych opóźnień występujących podczas przesyłania informacji. Istotne jest wykonanie wystarczająco licznej próby w powtarzalnych warunkach aby można było określić kształt rozkładu losowego czasu transmisji, nie jest natomiast wymagany szczegółowy opis architektury badanej sieci i jej wpływu

na uzyskane wyniki. W celu zbadania szybkości transmisji pakietów poprzez sieć dokonano pomiaru czasu odpowiedzi na pakiet generowany przez program ping.

Zaletami takiego rozwiązania są:

- dostępność użytych narzędzi,
- możliwość uzyskania odpowiedzi od komputerów do których nie ma się uprawnień użytkownika, ani administratora,
- możliwość pomiaru na długich i krótkich odcinkach sieci,
- implementacja odpowiedzi na ping w jądrze systemu, co zapewnia szybszą odpowiedź,
- pomiar na odcinku zamkniętym, bez konieczności synchronizacji zegarów,
- stosunkowo dobra rozdzielczość pomiaru (0,1 msec)

Do wad tej metody można zaliczyć:

- pomiar czasu przesyłania pakietu do odległego hosta i z powrotem, bez możliwości pomiaru czasu transmisji w jedną stronę,
- niewystarczająca dokładność przy pomiarze w szybkich sieciach.

10.3. Wyniki pomiaru szybkości komunikacji

Pomiaru szybkości komunikacji dokonano badając czas odpowiedzi kilku reprezentatywnych komputerów na pytanie (ping) z komputera monster.iic.pwr.wroc.pl (i-486, Linux). Kolejne ramki wysyłane były co 30 sekund przez 24 godziny co daje 2880 pomiarów. Pomiaru dokonywano w środku tygodnia. Pomiar miał na celu zbadanie kształtu rozkładów losowych, a nie określenie dokładnych parametrów sieci. W związku z tym nie dokonywano badań w dni świąteczne, gdyż stopień obciążenia sieci zmniejsza się wtedy kilkukrotnie. Połączenie wyników dni powszednich i świątecznych dałoby fałszywy obraz rozkładu losowego ze względu na silną korelację wartości czasu z dniem tygodnia.

Przed przystąpieniem do właściwych pomiarów wykonano badania szybkości odpowiedzi komputerów z poza uczelni (sieć rozległa) w poszczególne dni tygodnia. Zaobserwowano, że kształt rozkładu losowego czasu transmisji w dni wolne ma takie same właściwości jak w dni robocze, z tym że jest wyraźnie przesunięty w kierunku mniejszych wartości. We wszystkich badaniach przyjęto założenie o niezależności poszczególnych pomiarów, zatem odrzucono pomiary z dni świątecznych aby uniknąć błędów wynikających z korelacji wyników spowodowanej obciążeniem sieci.

Podsumowanie wyników pomiarów zebrano w tabeli 10-11. Wartości czasów wyrażone są w mili-sekundach.

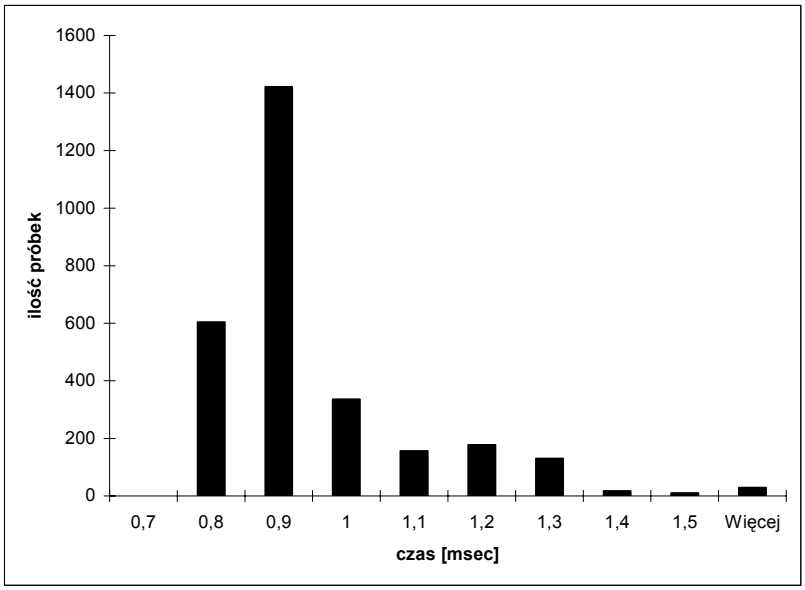
Tabela 10-11 Parametry rozkładów losowych czasów realizacji sekwencyjnego programu „life”

Host.	Min.	Max.	Śred.	Śr. 98%	Med.	Moda	σ	Od. śr.	R. kw.	Skos.	Kurt.
Localhost	0,8	61,1	1,0	0,95	0,9	0,9	1,324	0,18	0,1	37,01	1556
Własne IP	0,8	38,2	1,0	0,98	0,9	0,9	1,019	0,19	0,1	31,09	1064
Ghost	0,9	22,7	1,7	1,6	1,4	1,4	1,012	0,56	0,4	8,74	131
Router	2,9	407,6	8,0	7,0	4,5	4	15,95	5,66	2,5	9,71	162
Cyber	2,8	212,3	6,0	5,2	4,4	4	9,701	2,71	1,9	11,1	153
AE-WROC	3,3	879,7	6,6	5,2	4,7	4,4	22,74	3,38	1,0	26,7	878
Warszawa	222,4	1840,1	401,3	396,9	388,6	353,4	84,94	48,98	76,1	7,26	95
DIKU	60,7	1470,3	723,1	723,4	718,1	806,4	220,0	175,55	294,3	-0,023	0,0574
MIT	179,3	841,1	341,7	340,6	335,9	272,2	71,76	58,20	107,7	0,907	1,955

Przy pomiarach na dłuższych odcinkach występuje znaczny odsetek zagubionych pakietów. Częściowo są to pakiety stracone, częściowo pakiety, które nie zmieściły się w oknie czasowym programu ping. Przy obliczeniach parametrów rozkładu nie były one brane pod uwagę, gdyż obrazują one zjawisko zawodności sieci, a nie zmienności czasu transmisji pakietu.

Localhost

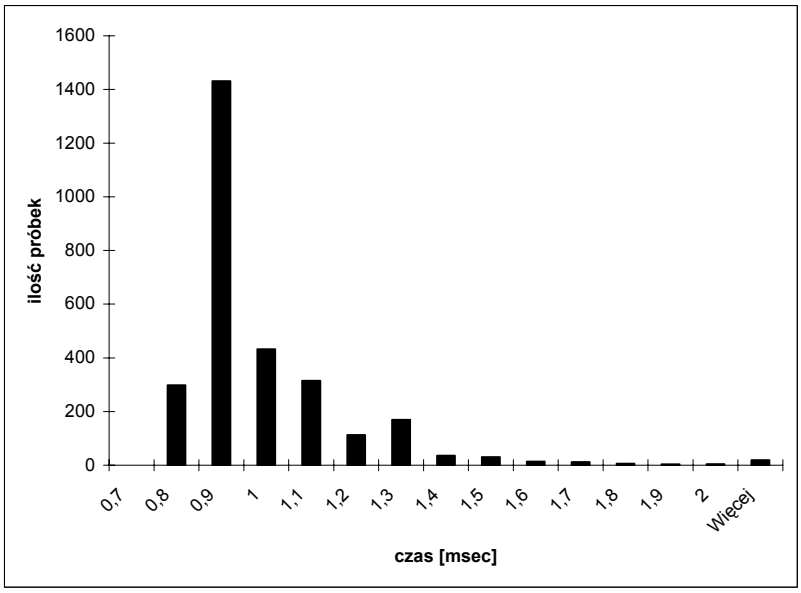
Teoretycznie najszybszą odpowiedź na ping można uzyskać zadając pytanie pod adres localhost 127.0.0.1. Wykorzystywany jest wtedy mechanizm zapętlenia pakietów IP bez konieczności uaktywniania karty sieciowej. Szybka odpowiedź w porównaniu do rozdzielczości pomiaru (0,1 msec) uniemożliwia uzyskanie dokładnego histogramu. Trudno stwierdzić czy pojawiające się drugie maksimum dla wartości 1,2 msec jest spowodowane niedokładnością pomiaru, czy też jest własnością kształtu rozkładu. Biorąc jednak pod uwagę, że inne pomiary mają podobny kształt należy przychylać się do hipotezy o występowaniu więcej niż jednego maksimum gęstości rozkładu losowego czasu odpowiedzi na ping.



Rys. 10-3. Histogram czasu odpowiedzi dla localhost

Własne IP

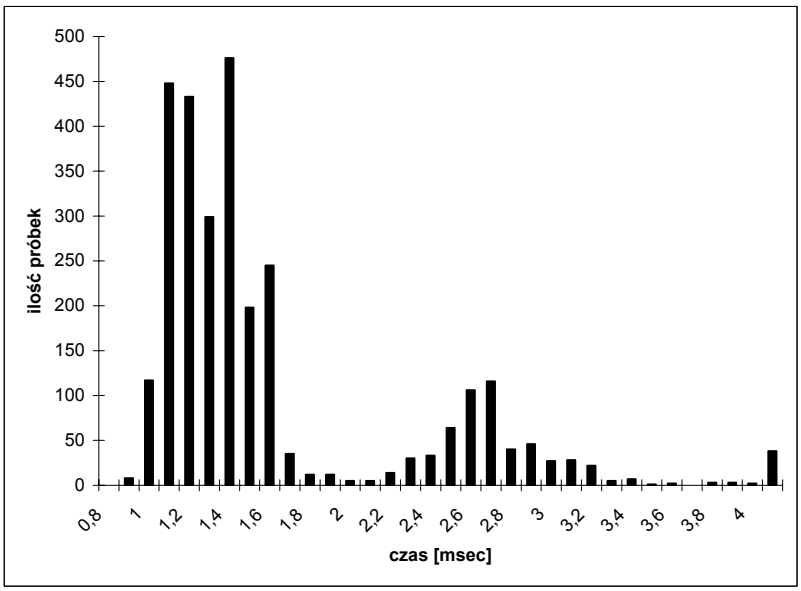
Odpowiedź na ping bez konieczności transmisji poprzez sieć można uzyskać kierując pytanie pod własny adres IP. Wymaga to sformowania pakietu dla potrzeb niższych warstw protokołu. W zależności od rodzaju i konfiguracji systemu operacyjnego możliwe jest fizyczne wysłanie pakietu i jego odbiór lub zapętlenie na etapie wewnętrznej tabeli routingu. Następnie konieczne jest rozkodownie i analiza pakietu. Powoduje to zwiększenie czasu koniecznego do wykonania danej operacji. Są to jednak operacje na tyle szybkie że powodują niewielkie zwiększenie średniego czasu oczekiwania na odpowiedź. Podobnie jak w przypadku kontaktu z localhost występuje drugie maksimum gęstości dla wartości 1,3 msec.



Rys. 10-4. Histogram czasu odpowiedzi dla własnego numeru IP

Ghost

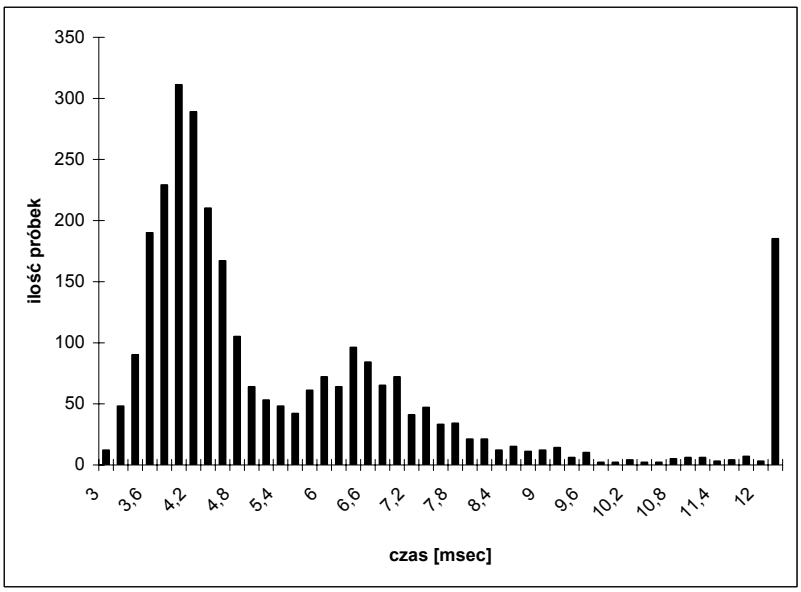
Komputer o nazwie ghost.iic.pwr.wroc.pl jest systemem Pentium133 + Windows95 podłączonym do tego samego odcinka sieci ethernet co komputer pytający. Można więc stwierdzić, że są bezpośrednio połączone, jednak w kanale mogą występować zakłócenia związane z przepływem innych ramek. W skład czasu odpowiedzi wchodzą więc czasy związane z formowaniem i nadawaniem pakietu, jego przepływem do odbiorcy, formowaniem odpowiedzi, jej przepływem do nadawcy i odczytem odpowiedzi przez nadawcę. Na wykresie widoczne są dwa bardzo wyraźne maksima gęstości. Trudno natomiast jednoznacznie stwierdzić istnienie lub brak większej ilości maksimów. Mogą one istnieć zarówno wewnątrz dwóch obszarów gdzie skupione są wartości, jak również dla wartości większych. Niezależnie od tego zdecydowanie można stwierdzić, że rozkład nie przypomina rozkładu normalnego. Warto również zwrócić uwagę na wartości nie mieszczące się na wykresie reprezentowane przez słupek o nazwie „więcej". Świadczą one o możliwości zdecydowanie wolniejszej niż przewidywana reakcji systemu na pytanie „ping". Ten wydłużony czas może być traktowany jako uszkodzenie, o ile w programie zadany jest krótki dopuszczalny czas oczekiwania na odpowiedź.



Rys. 10-5. Histogram czasu odpowiedzi dla komputera ghost

Router

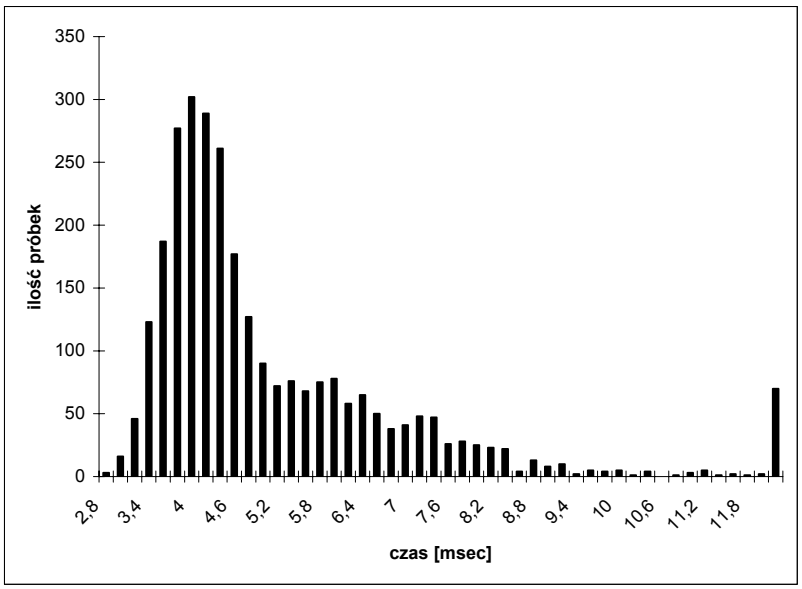
Kolejnym etapem badań było określenie czasu odpowiedzi routera łączącego sieć lokalną z siecią szkieletową Politechniki Wrocławskiej. Aby uzyskać dłuższą drogę pakietu zapytanie kierowano na numer IP nie należący do naszej sieci (router posiada kilka numerów IP). Histogram przedstawiający wyniki pomiarów również posada dwa wyraźne maksima dla wartości 4,0 i 6,4. Można doszukiwać się dalszych maksimów gęstości dla 8,8 oraz 11,2 tworzących razem ciąg arytmetyczny. Podobnie jak w poprzednim przypadku otrzymano stosunkowo wiele dużych wartości nie mieszczących się na wykresie, reprezentowanych poprzez słupek „więcej". Nie są one rozłożone całkowicie równomiernie, jednak trudno doszukać się wyraźnych maksimów w tym obszarze. Obserwuje się zatem zjawisko rozmycia i wzajemnego nachodzenia na siebie poszczególnych punktów skupienia rozkładu. Potwierdza to istnienie wielu niezależnych czynników wpływających na czas transmisji pakietu w sposób trudny do analitycznego przewidzenia.



Rys. 10-6. Histogram czasu odpowiedzi dla routera podsieci iic.pwr.wroc.pl

Cyber

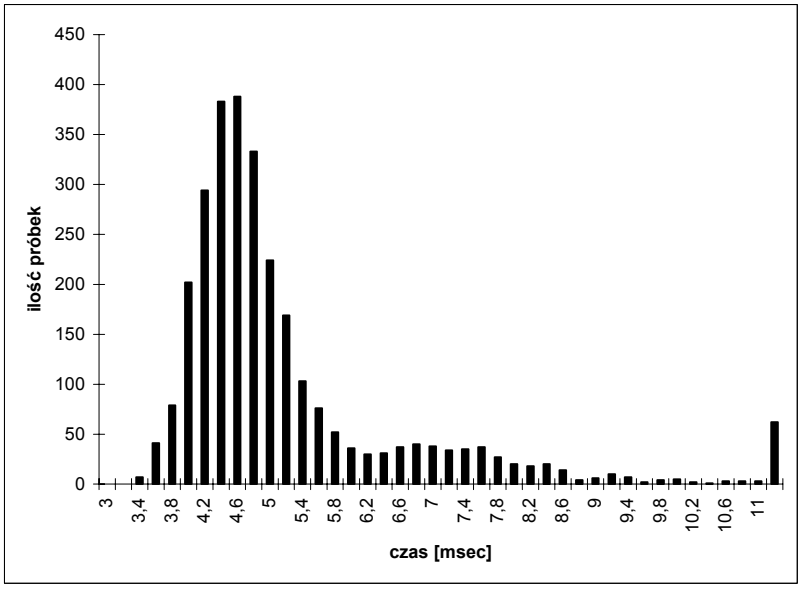
Pomiaru szybkości komunikacji pomiędzy komputerami Politechniki Wrocławskiej dokonano badając czas odpowiedzi komputera cyber.ict.pwr.wroc.pl znajdującego się w innym budynku. Histogram przedstawiający wyniki zawiera wyraźnie widoczne główne maksimum gęstości dla wartości 4,0 msec, oraz bardzo niewyraźne drugie i trzecie dla wartości 5,8 i 7,2 msec. Rozkład ma kształt bardziej zbliżony do normalnego, co można tłumaczyć faktem, iż czas jest sumą kilkunastu czasów o podobnych wartościach średnich i wariancjach.



Rys. 10-7. Histogram czasu odpowiedzi komputera cyber

Akademia Ekonomiczna

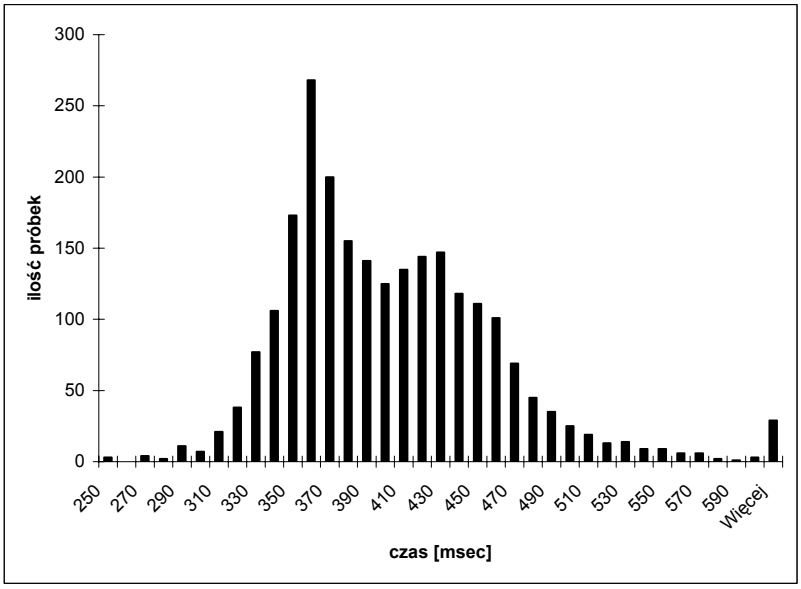
W celu zbadania szybkości komunikacji w sieci miejskiej zbadano czas odpowiedzi komputera Akademii Ekonomicznej. Histogram przedstawiający wyniki zawiera dwa wyraźne maksima gęstości. Inne maksima są na tyle niewyraźne, że mogą być błędami pomiarowymi.



Rys. 10-8. Histogram czasu odpowiedzi komputera Akademii Ekonomicznej

Warszawa

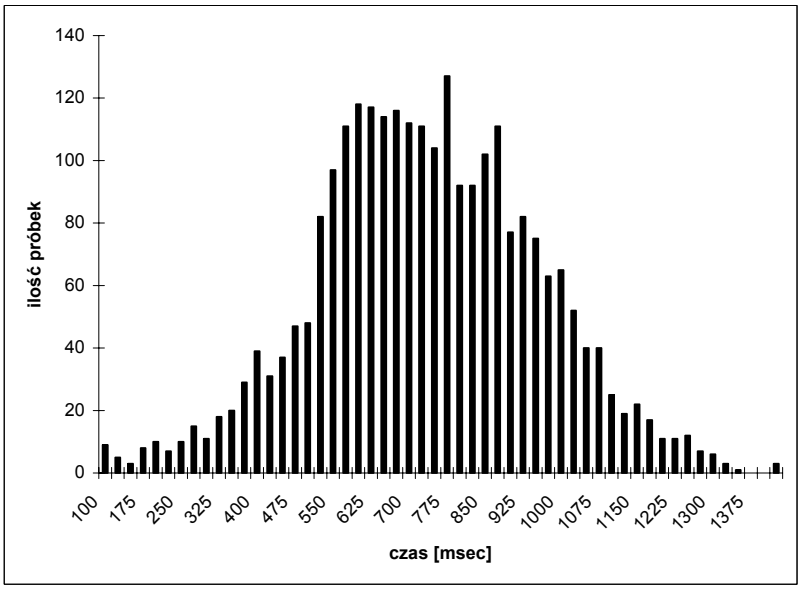
Badanie szybkości transmisji w obrębie kraju zrealizowano na podstawie pomiaru czasu odpowiedzi komputera znajdującego się w Warszawie. Histogram przedstawiający wyniki również zawiera dwa maksima gęstości prawdopodobieństwa, jednak rozkład jest zdecydowanie bardziej podobny do rozkładu normalnego niż rozkłady prezentowane poprzednio.



Rys. 10-9. Histogram czasu odpowiedzi komputera z Warszawy

Dania

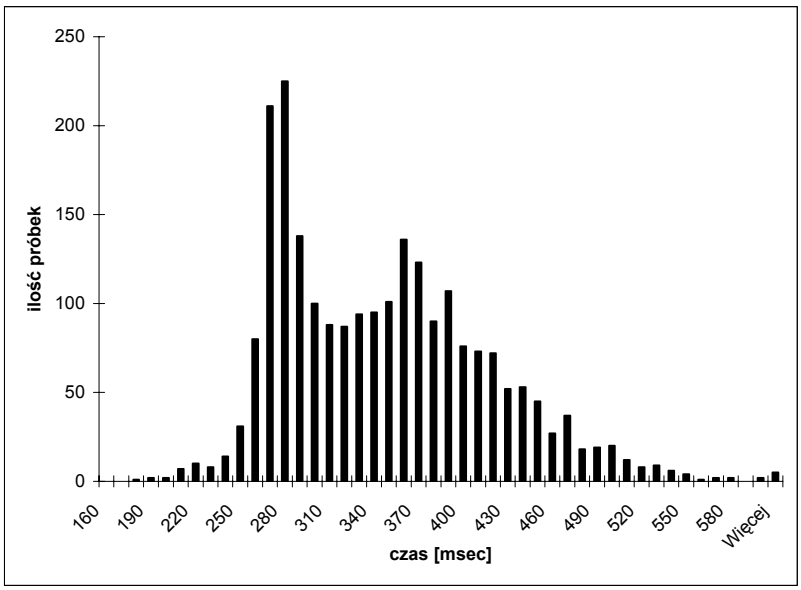
Badanie szybkości odpowiedzi komputera poza granicami kraju przeprowadzono na przykładzie www.diku.dk znajdującego się w Kopenhadze. Rozkład czasu odpowiedzi przedstawiony na histogramie w dużym stopniu przypomina rozkład normalny. Można to tłumaczyć faktem, iż czas odpowiedzi jest sumą kilku czasów o podobnej długości.



Rys. 10-10. Histogram czasu odpowiedzi komputera z Danii

USA

Czas odpowiedzi komputera z innego kontynentu badano na przykładzie www.mit.edu znajdującego się w USA. Ze względu na asymetrię odcinków sieci wchodzących w skład drogi (łącze między kontynentami) rozkład jest w mniejszym stopniu normalny niż rozkład czasu odpowiedzi komputera z Danii.



Rys. 10-11. Histogram czasu odpowiedzi komputera z USA

Wnioski z badań

Kształt i parametry rozkładu losowego czasu transmisji pakietu przez sieć w istotny sposób zależą od odległości pomiędzy komputerami. Jeżeli odległość jest znaczna, a parametry składowych odcinków sieci porównywalne to rozkład czasu upodabnia się do rozkładu normalnego. Sytuację tę uwidacznia rozkład czasu komunikacji z komputerem w Danii. Rozkład ten posiada cechy rozkładu normalnego a w szczególności:

- Równe wartości średniej, średniej wewnętrznej i mediany
- Wartość średnia znajduje się w połowie przedziału przyjmowanych wartości.
- Rozstęp kwartyli równy jest 1,34 odchylenia standardowego (powinien 1,35)
- Wartość skośności –0,025 jest bliska zera
- Wartość kurtozy 0,057 jest bliska zera
- Test Chi-kwardrat nie daje podstaw do odrzucenia hipotezy o normalności rozkładu

Rozkłady losowe czasu komunikacji na krótszych odcinkach wykazują cechy zbliżone do rozkładów czasu realizacji fragmentu kodu, w szczególności:

- Wartości średniej, mediany i mody są bardzo bliskie wartości minimalnej. Świadczy to o skupieniu centrum rozkładu w pobliżu dolnego ograniczenia zbioru wylosowanych wartości.
- Dodatnia, wysoka wartość skośności świadczy, iż wykres funkcji gęstości jest szybko zbieżny lewostronnie do zera a wolno prawostronnie.
- Wartości średniej wewnętrznej (z 98% próbek) są mniejsze od średniej z wszystkich próbek, co potwierdza dodatnią skośność rozkładu.

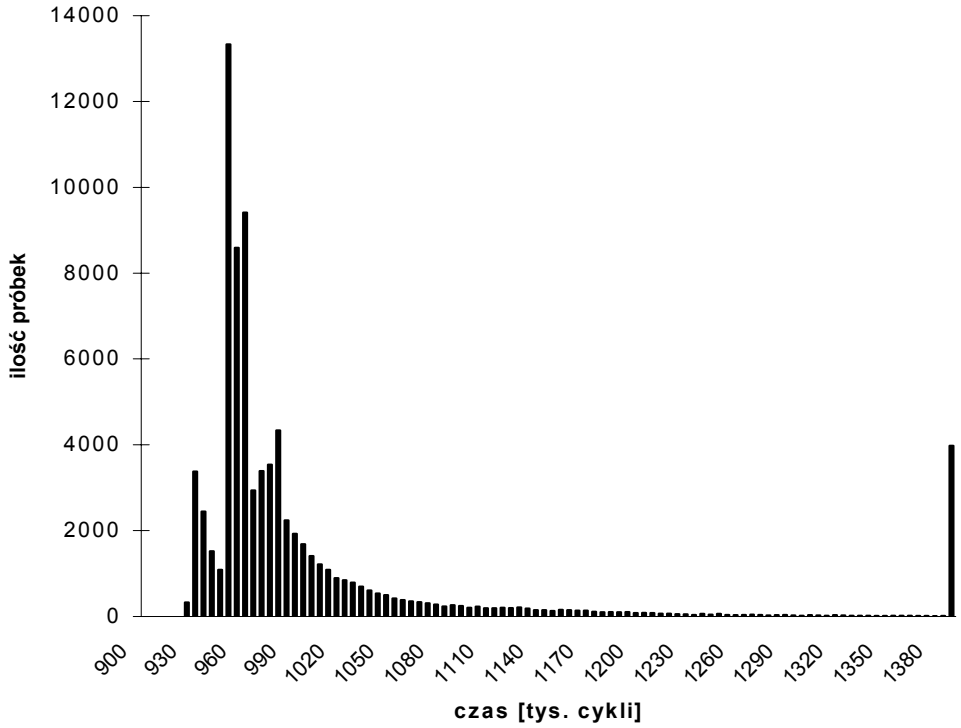
- Rozstęp kwartyli jest mniejszy od odchylenia standardowego. Stosunek ten świadczy o skupieniu przyjmowanych wartości w wąskim przedziale, oraz istnieniu wartości zdecydowanie odbiegających od średniej wpływających na wysoką wartość odchylenia standardowego. Wykres jest zatem wyraźnie „szpiczasty".
- Dodatnia, wysoka wartość kurtozy również potwierdza „szpiczastość" wykresu funkcji gęstości.
- Występowanie dodatkowych punktów skupienia o dużym wpływie na wartość średnią i wariancję.

Rozkłady losowe czasu komunikacji na krótkich odcinkach różnią się od rozkładów czasu realizacji kodu następującymi cechami:

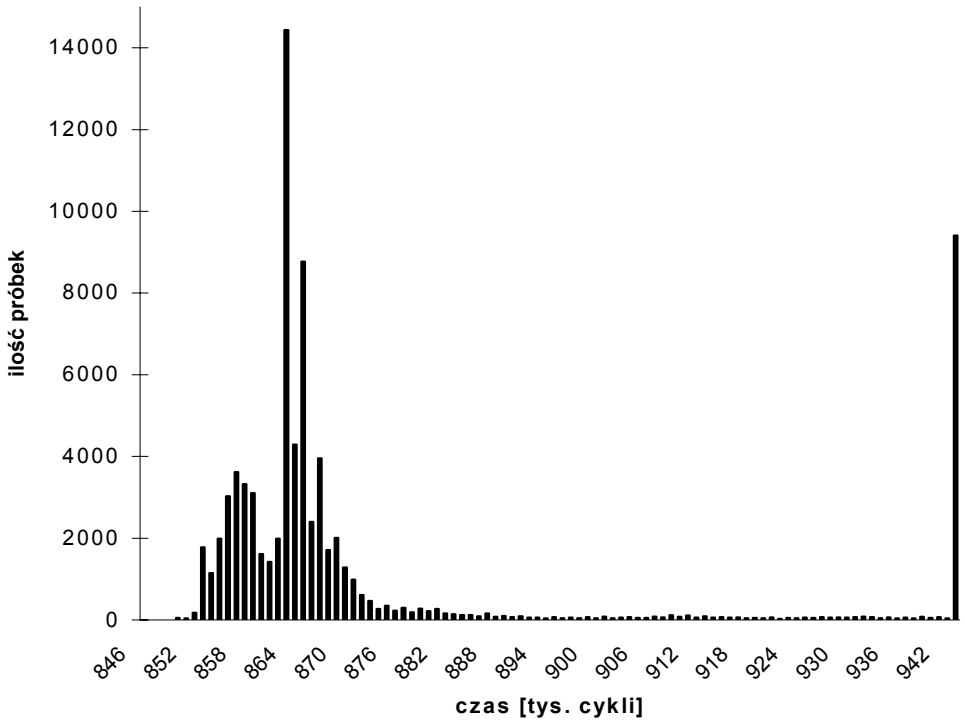
- Dodatkowe punkty skupienia znajdują się bliżej centrum rozkładu.
- Zdecydowanie większe jest prawdopodobieństwo wylosowania wartości znajdującej się w obrębie dodatkowego punktu skupienia niż ma to miejsce w przypadku rozkładu czasu realizacji kodu.
- Mniejsza jest rozpiętość pomiędzy wartością minimalną a maksymalną zbioru przyjmowanych wartości.

10.4. Wyniki pomiarów szybkości realizacji programów współbieżnych

W celu zbadania kształtu i właściwości rozkładów losowych czasu wykonywania programów współbieżnych zmierzono czas realizacji dwóch przykładowych aplikacji współbieżnych pracujących na dwóch oddzielnych komputerach klasy Pentium połączonych siecią ethernet 10Mb. Pierwszy program jest równoległą symulacją gry „life", drugi sprowadza się do wzajemnego przesyłania komunikatów. Czas trwania każdego kroku programu przedstawiają wykresy 10-13 i 10-14. W programie „life" pojedynczy krok aplikacji składa się z przesłania komunikatów w obie strony, oraz przekształcenia tablicy w trzech oddzielnych etapach z wykorzystaniem danych otrzymanych od współpracującego komputera. Program „ping-pong" ogranicza się do wzajemnego przesyłania komunikatów. Wykonanie programu składa się z dwudziestu kroków. Ostatni słupek histogramu opisuje liczbę próbek większych niż maksymalna wartość czasu widoczna na wykresie.



Rys. 10-12 Histogram czasu wykonania programu life – rozmiar 10x10



Rys. 10-13 Histogram czasu wykonania programu ping-pong

Tabela 10-12 Parametry rozkładów czasu realizacji aplikacji współbieżnych

	Min.	Max.	Śred.	Śr. 98%	Med.	Moda	σ	Od. śr.	R. kw.	Skos.	Kurt.
Life	920k	9.83M	1.03M	1.01M	959k	948k	258k	108k	45k	8,75	136
P-P	815k	119M	960k	915k	863k	862k	1176k	167k	8.35k	62,9	4837

Należałoby się spodziewać, że rozkłady losowe czasu realizacji całych programów współbieżnych będą miały kształt zbliżony do normalnego gdyż stanowią złożenie czasów realizacji poszczególnych fragmentów. Na podstawie centralnego twierdzenia granicznego oczekuje się, że rozkład sumy wielu niezależnych zmiennych losowych będzie miał kształt dążący do rozkładu normalnego. W przypadku synchronizujących się ze sobą procesów występuje konieczność oczekiwania na gotowość drugiego procesu. Rozkład czasu realizacji zadania nie jest zatem rozkładem sumy zmiennych losowych.

Rozkłady losowe czasu realizacji aplikacji współbieżnych posiadają wiele cech wspólnych z rozkładani czasu realizacji mniejszych fragmentów kodu oraz procedur komunikacyjnych.

W szczególności:

- Wartości średniej, mediany i mody są bliższe wartości minimalnej niż środkowi przedziału przyjmowanych wartości. Świadczy to o skupieniu centrum rozkładu w pobliżu dolnego ograniczenia zbioru wylosowanych wartości.
- Dodatnia, wysoka wartość skośności świadczy, iż wykres funkcji gęstości jest szybko zbieżny lewostronnie do zera a wolno prawostronnie.
- Wartości średniej wewnętrznej (z 98% próbek) są mniejsze od średniej z wszystkich próbek, co potwierdza dodatnią skośność rozkładu.
- Wartości mediany i mody są mniejsze od średniej.
- Rozstęp kwartyli jest zdecydowanie mniejszy od odchylenia standardowego. Świadczy to o skupieniu przyjmowanych wartości w wąskim przedziale, oraz istnieniu wartości zdecydowanie odbiegających od średniej wpływających na wysoką wartość odchylenia standardowego. Wykres jest wyraźnie „szpiczasty".
- Dodatnia, wysoka wartość kurtozy również potwierdza „szpiczastość" wykresu funkcji gęstości.

Rozkłady klasy „heavy tail"

Na podstawie przeprowadzonych doświadczeń można stwierdzić, że rozkłady czasu realizacji fragmentów kodu, wykazują cechy charakterystyczne dla rozkładów klasy „heavy tail". Ich wspólną cechą wolna prawostronna zbieżność do zera funkcji gęstości. Rezultatem jest bardzo silny wpływ sporadycznie występujących dużych wartości na parametry rozkładu. Najbardziej reprezentatywnym przedstawicielem tej klasy rozkładów jest rozkład hiperboliczny (Pareto)

$$\Phi(t) = 1 - \left(\frac{\beta}{t}\right)^{\alpha} \tag{10-2}$$

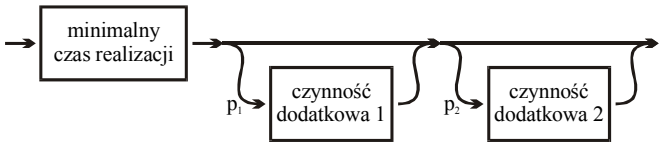
Dla rozkładu Pareto k-ty moment istnieje dla $k<\alpha$.

Przybliżanie kształtu rozkładu losowego czasu realizacji programu dystrybuantą Pareto prowadzi do uzyskania wartości parametru $1<\alpha<2$. Wynika z tego, że wartość oczekiwana tak modelowanego czasu realizacji programu jest skończona, ale wariancja jest nieskończona.

10.5. Model stochastyczny opóźnień występujących podczas realizacji programu

W celu stworzenia jednolitego sposobu reprezentacji kształtu rozkładów losowych czasów realizacji kodu i przesyłania komunikatu proponuje się wykorzystanie stochastycznego modelu działania procesora i kanału komunikacyjnego.

Przyjęto, że czas realizacji zadania jest równy sumie czasów składowych opisanych prostymi dwupunktowymi rozkładami. Każdy z czasów składowych opisuje w sposób stochastyczny jeden z elementów systemu mający wpływ na szybkość wykonywania instrukcji (np. zajętość magistrali, chybienie pamięci podręcznej). Założono, że składniki sumy są niezależne od siebie. Opracowując model działania procesora przyjęto uproszczenie, że każdy ze składników sumy ma rozkład dwupunktowy. Krótszy czas odpowiada korzystnemu zjawisku (np. trafienie w pamięci podręcznej), które w większości przypadków jest bardziej prawdopodobne niż zjawisko niekorzystne (np. chybienie). Przyjęto więc, że prawdopodobieństwo wylosowania mniejszej wartości jest większe niż prawdopodobieństwo wylosowania większej.



Rys. 10-14 Czas realizacji instrukcji jako suma czasu minimalnego oraz opóźnień występujących z określonym prawdopodobieństwem.

Prawdopodobieństwa wykonania przez procesor czynności dodatkowych są w ogólnym przypadku zależne zarówno od rodzaju wykonywanych instrukcji, jak też od właściwości maszyny. Upraszczając model przyjęto, że prawdopodobieństwo wystąpienia czynności dodatkowych jest stałe w jednostce czasu i nie zależy od rodzaju wykonywanych instrukcji, a jedynie od ilości czasu, przez jaki procesor je wykonuje. W przypadku szczegółowej analizy opóźnień związanych z obsługą pamięci wirtualnej można wyróżnić podzbiór instrukcji dokonujących zapisu i odczytu pamięci i przyjąć, że instrukcje z tego zbioru w znaczący sposób zwiększają prawdopodobieństwo wystąpienia czynności dodatkowych zawiązanych z wymianą strony przez system operacyjny.

Przyjęto, że czas trwania czynności dodatkowych nie jest zależny od rodzaju wykonywanych instrukcji, a jedynie od właściwości maszyny wraz z systemem operacyjnym. W przypadku zdarzeń będących przerwaniami systemu operacyjnego realizującymi złożone procedury można przyjąć, że czas realizacji przerwania nie jest stały, lecz również podlega zmienności losowej.

Przewidywane właściwości rozkładu losowego czasu realizacji instrukcji.

Oznaczając:

P_i – prawdopodobieństwo wystąpienia i-tego rodzaju opóźnienia

t_i – czas trwania i-tego rodzaju opóźnienia

Wartość oczekiwana czasu realizacji instrukcji.

$$E(t) = t_0 + \sum_{i=1}^{n} p_i t_i \tag{10-3}$$

W typowym systemie komputerowym można przyjąć, że opóźnienia mogą powstać na skutek niegotowości praktycznie każdego elementu architektury. Model zawierać będzie często występujące opóźniania rzędu pojedynczych cykli, jak również sporadycznie występujące opóźnienia rzędu sekund (np. rekalibracja głowicy CD-ROM). Przyjmijmy zatem następujące założenia:

- ilość rodzajów opóźnień *n* jest większa niż 10
- wartości czasów opóźnień t_i różnią się o kilka rzędów wielkości
- wartość oczekiwana E(t) czasu realizacji instrukcji jest skończona
- wpływ każdego z opóźnień na wartość E(t) jest porównywalny

Ostatnie z założeń ma uzasadnienie natury ekonomicznej. Jeżeli wpływ jakiegoś elementu architektury komputera na (średnią) wydajność maszyny jest nieistotny to nie przeznacza się środków na jego modernizację. Dla kontrastu element wnoszący największe średnie opóźnienia będzie podlegał intensywnej optymalizacji. W „stanie ustalonym" wszystkie rodzaje opóźnień będą miały w przybliżeniu jednakowy wpływ na wartość oczekiwaną czasu realizacji. Zatem:

$$\forall i, j : p_i t_i \approx p_j t_j \tag{10-4}$$

Przy założonej bardzo dużej zmienności t_i – 6-10 rzędów wielkości dla kompletnego opisu systemu,

$$t_i \to +\infty,\ p_i t_i \approx const \Rightarrow p_i t_i^2 \to +\infty \tag{10-5}$$

Zatem wariancja rozkładu dąży do nieskończoności:

$$Var(t) = \sum_{i=1}^{n} p_i t_i^2 - E^2(t) \tag{10-6}$$

Rozkład losowy czasu realizacji instrukcji wynikający z proponowanego modelu będzie miał skończoną wartość tylko pierwszego momentu. Zatem będzie on należał do klasy rozkładów „heavy tail".

10.6. Doświadczalna weryfikacja modelu stochastycznego

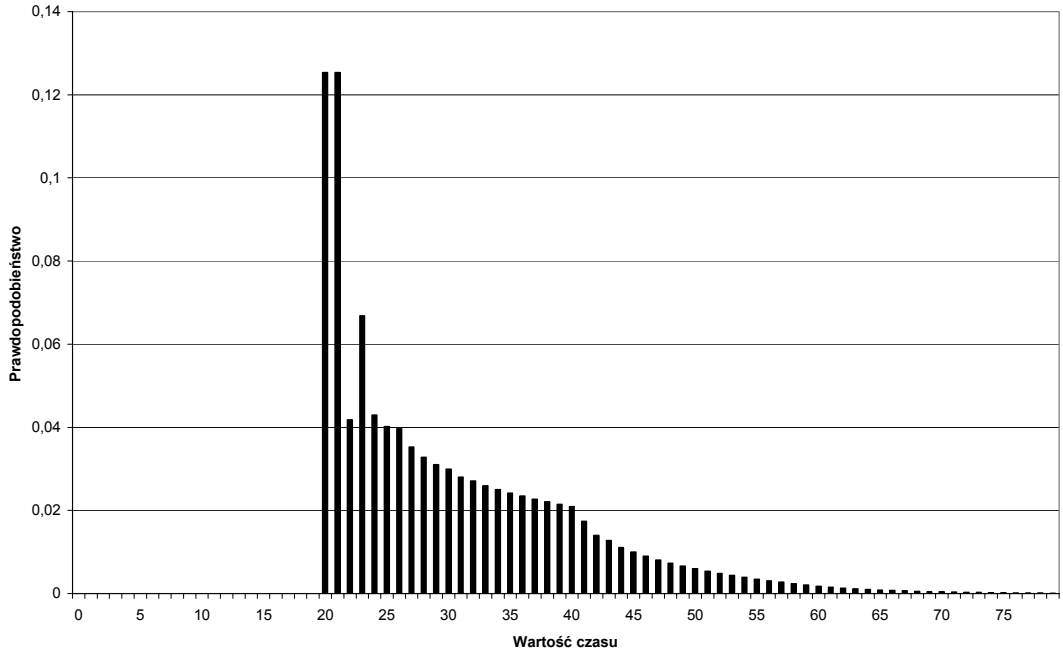
W celu weryfikacji przyjętego modelu wykonano symulacje i doświadczenia mające na celu:

- Potwierdzenie lub zaprzeczenie zgodności kształtu rozkładu losowego generowanego przez model z kształtem rzeczywistych rozkładów losowych czasów realizacji programu.
- Potwierdzenie możliwości określenia parametrów modelu na podstawie pomiarów.
- Sprawdzenie dokładności odwzorowania działania procesora przez przyjęty model poprzez porównanie przewidywanego rozkładu czasu wykonania procedury z rzeczywistym rozkładem.
- Stwierdzenie skalowalności modelu od opisu realizacji pojedynczych instrukcji do dużych procedur oraz całości wykonywanego programu.

W celu sprawdzenia w jakim stopniu rozkłady generowane przez zaproponowany model mają kształt zbliżony do rzeczywistych rozkładów czasu realizacji kodu lub szybkości transmisji przeprowadzono badania dla różnych kształtów rozkładów składowych oraz ilości elementów modelu podlegających sumowaniu. Zauważono, że stosując najprostsze rozkłady dwupunktowe (rozkład stały oraz prawdopodobieństwo zaistnienia danego opóźnienia) uzyskuje się kształty bardzo zbliżone do rzeczywistych już przy sumowaniu 10-20 składników.

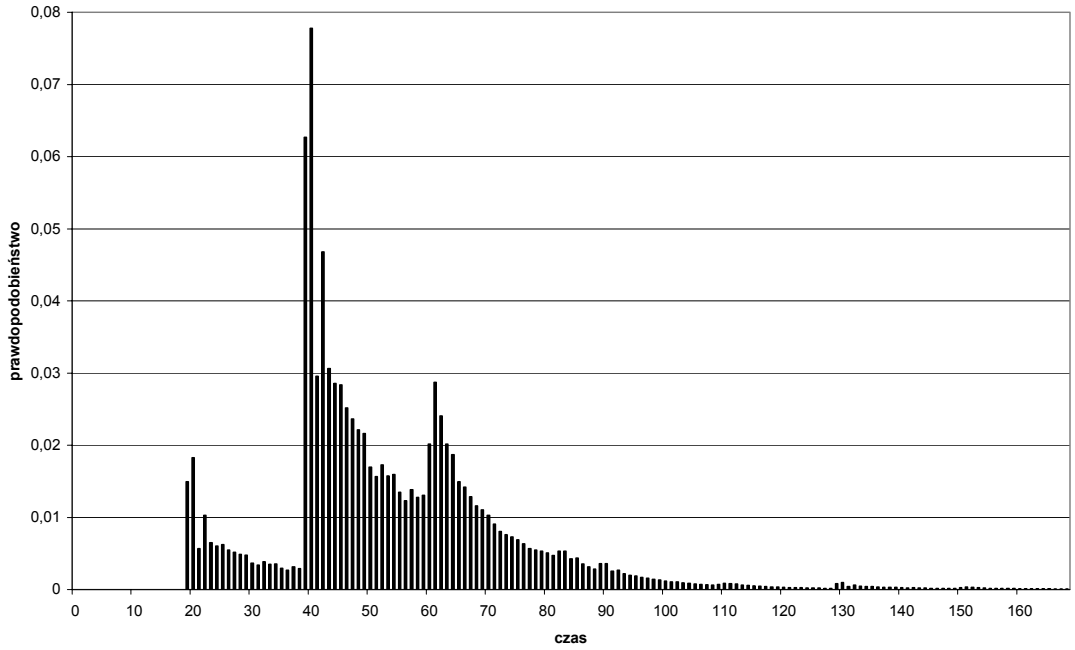
Rys. 10-15 przedstawia histogram rozkładu losowego czasu wykonywania instrukcji. Prezentowany rozkład uzyskano w wyniku sumowania 20 niezależnych zmiennych losowych o rozkładach dwupunktowych.

Rozkład i-tej zmiennej przyjmował wartości {1, 1+i} oraz prawdopodobieństwo wylosowania większej wartości p_i=0.5/i . Nie uwzględniono czynników o wysokich wartościach i niskich prawdopodobieństwach ze względu na stopień komplikacji obliczeń i nieczytelność wykresu. Określenia kształtu rozkładu dokonano metodą kombinatoryczną zatem wynik jest obarczony co najwyżej błędami wynikającymi z niedokładności obliczeń zmiennoprzecinkowych.



Rys. 10-15. Symulowany rozkład losowy czasu realizacji instrukcji

Rys. 10-16 przedstawia prawdopodobieństwo wylosowania danej wartości czasu wykonywania instrukcji jako funkcję czasu. Prezentowany jest rozkład losowy sumy 20 zdarzeń niezależnych z których i-ty rozkład dwupunktowy przyjmował wartości {1, 1+i} oraz prawdopodobieństwo wylosowania większej wartości p_i=0.5/i. Dodatkowo uwzględniono 4 składniki sumy o wyższych wartościach czasu oraz jeden czynnik, dla którego prawdopodobieństwo wylosowania większej wartości jest większe niż mniejszej wartości. Określenia kształtu rozkładu dokonano metodą kombinatoryczną zatem wynik jest obarczony co najwyżej błędami wynikającymi z niedokładności obliczeń zmiennoprzecinkowych.



Rys. 10-16. Symulowany rozkład czasu realizacji instrukcji w systemie operacyjnym

Prezentowane na rys. 10-16 i 10-17 rozkłady losowe uzyskane w wyniku symulacji posiadają wszystkie cechy rozkładów losowych czasu realizacji fragmentu kodu oraz czasu przesyłania pakietu w szczególności:

- Występuje czas t_{min} poniżej którego wartość gęstości prawdopodobieństwa jest zerowa.
- Wartości opisujące położenie są w następującej relacji: moda < mediana < średnia.
- Wartość średnia jest zbliżona do dolnego ograniczenia zbioru przyjmowanych wartości.
- Rozstęp kwartyli jest mniejszy od odchylenia standardowego.
- Rozkład jest wyraźnie asymetryczny (wartość skośności jest dodatnia).
- Rozkład jest „szpiczasty" (wartość kurtozy jest dodatnia).
- Występują dodatkowe punkty skupienia.

Zaproponowany model umożliwia zatem generowanie rozkładów losowych o kształtach i parametrach zbliżonych do rzeczywistych rozkładów czasu realizacji kodu lub transmisji informacji.

Określenie parametrów modelu

Kolejnym etapem badania jakości modelu było dopasowanie jego parametrów do zmierzonych rozkładów szybkości realizacji funkcji *gettimeofday*. Dopasowania dokonano numerycznie – najpierw wyszukując odseparowane punkty skupienia, następnie dobierając pozostałe parametry w celu jak najlepszego odwzorowania centrum rozkładu. Jako kryterium dopasowania użyto statystyki *chi-kwadrat* dążąc do minimalizacji otrzymanej sumy.

$$\chi^2 = \sum_{i=1}^{n} \frac{(L_i - E_i)^2}{E_i} \tag{10-7}$$

Jako zbiór wzorców $\{E_i\}$ przyjęto ilości wystąpień generowane przez model, natomiast ilości próbek uzyskane doświadczalnie jako zbiór analizowany $\{L_i\}$.

Dla komputera monster uzyskano następujące parametry modelu:

t_{min}= 12μs

t_1=1μs p_1=0.75

t_2=2μs p_2=0.6

t_3=3μs p_3=0.1

t_4=18μs p_4=0.002

t_5=122μs p_5=0.01

t_6=364μs p_6=0.023

Dla komputera monster uzyskano następujące parametry modelu:

t_{min}= 4μs

t_1=1μs p_1=0.001

t_2=2μs p_2=0.0009

t_3=3μs p_3=0.0006

t_4=4μs p_4=0.0002

t_5=10μs p_5=0.0006

t_6=218μs p_6=0.0005

t_7=758μs p_7=0.0015

Przeprowadzone testy chi-kwadrat pozwalają na przyjęcie na poziomie ufności 95% hipotezy o zgodności rozkładu modelowego z rzeczywistym dla obu przypadków. Potwierdzona została zatem możliwość doświadczalnego określenia parametrów modelu jak również wierność odwzorowania rzeczywistych rozkładów przez model.

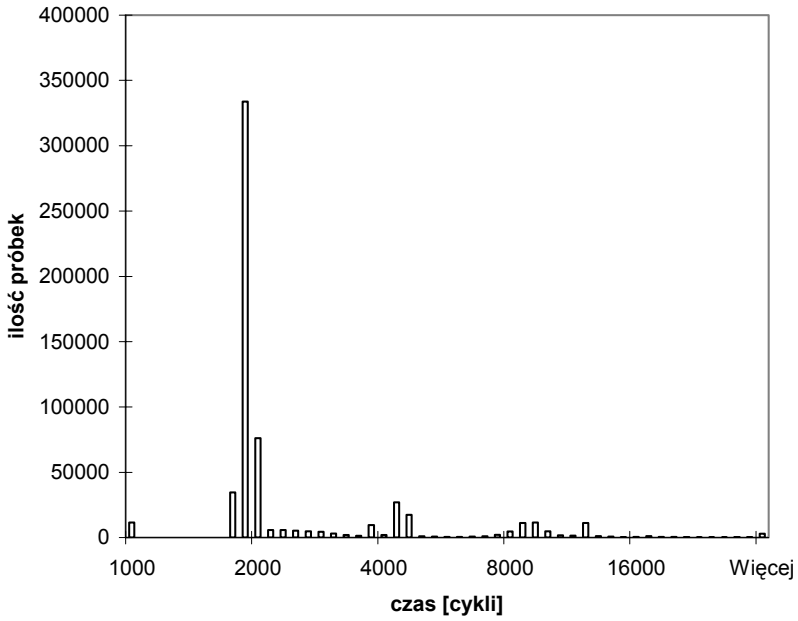
Określenie losowych opóźnień generowanych przez system operacyjny

W celu określenia wpływu systemu operacyjnego (środowiska w którym wykonywany jest program) na czas wykonywania programów dokonano pomiarów mających na celu określenie parametrów modelu opóźnień procesora, aby następnie na podstawie otrzymanego modelu przewidywać czas wykonania innych programów na tej samej maszynie. Zmierzono czas realizacji najmniejszego możliwego fragmentu, dzięki czemu uzyskano rozkład losowy z oddzielonymi, wyraźnymi maksimami gęstości. Na podstawie takiego rozkładu można dokładniej wyznaczyć parametry modelu. Aby uzyskać wiarygodny obraz rozkładu losowego konieczne jest przeprowadzenie bardzo dużej liczby pomiarów, gdyż dla większości przypadków podczas wykonywania najmniejszego możliwego fragmentu nie nastąpi żadna niespodziewana aktywność systemu lub innych programów.

Dokonano 10 miliardów ($10*10^9$) pomiarów czasu wykonania pustej instrukcji pomiarowej. W przypadku braku niespodziewanych opóźnień czas realizacji procedury wynosił 51 cykli maszynowych. Testy wykonywano na komputerze wyposażonym w procesor Pentium pracującym pod kontrolą systemu Linux.

W wyniku przeprowadzonych doświadczeń otrzymano 608559 wartości przekraczających 60 cykli. Daje to prawdopodobieństwo wystąpienia opóźnienia $61*10^{-6}$ dla 51 cykli. Przy założeniu, że zdarzenia wywołujące

opóźnienia występowały pojedynczo otrzymuje się prawdopodobieństwo wystąpienia akcji systemu operacyjnego $1{,}2*10^{-6}$ na każdy cykl maszynowy realizacji programu użytkownika.



Rys. 10-17. Histogram czasu losowego opóźnienia generowanego przez środowisko.

Histogram czasu losowych opóźnień generowanych przez system operacyjny przedstawiono na rys. 10-18. Skala czasu jest logarytmiczna, zatem szerokość przedziałów jest rosnąca. Histogram przedstawia jedynie próbki dla których wystąpiło opóźnienie związane z akcją systemu. Zaobserwowano wartości opóźnień nie mieszczące się w dziedzinie histogramu. 2853 próbki przekraczają 30 tys. cykli, natomiast największa zmierzona wartość opóźnienia wyniosła 27 mln. cykli.

Można dostrzec następujące właściwości rozkładu:

- Występują trzy główne maksima gęstości związane prawdopodobnie z trzema rodzajami czynności wykonywanych w losowy sposób przez system operacyjny.
- Maksima są rozmyte, co świadczy o zmienności czasu realizacji procedur systemu operacyjnego.

Na podstawie otrzymanych wyników pomiarów skonstruowano model procesora uwzględniający 5 rodzajów opóźnień generowanych przez system operacyjny. Dla każdego z nich przyjęto, że nie jest ono stałe lecz posiada rozkład losowy o kształcie wynikającym z otrzymanych pomiarów (dla opóźnienia 1000 rozkład punktowy, dla 2000 opisany numerycznie, dla 9500 normalny)

Należy oczekiwać, że przy realizacji fragmentu wymagającego większej ilości cykli wystąpi odpowiednio większa liczba przerwań systemu operacyjnego i związanych z nimi opóźnień. Pożądaną cechą modelu jest możliwość określenia opóźnień generowanych przez system podczas realizacji dłuższych fragmentów na podstawie parametrów uzyskanych w wyniku pomiarów krótszych fragmentów. W celu sprawdzenia zgodności przewidywań na podstawie modelu z rzeczywistym wynikami przeprowadzono pomiary czasu realizacji 10-krotnie dłuższego fragmentu kodu. Aby jednocześnie sprawdzić wpływ rodzaju kodu

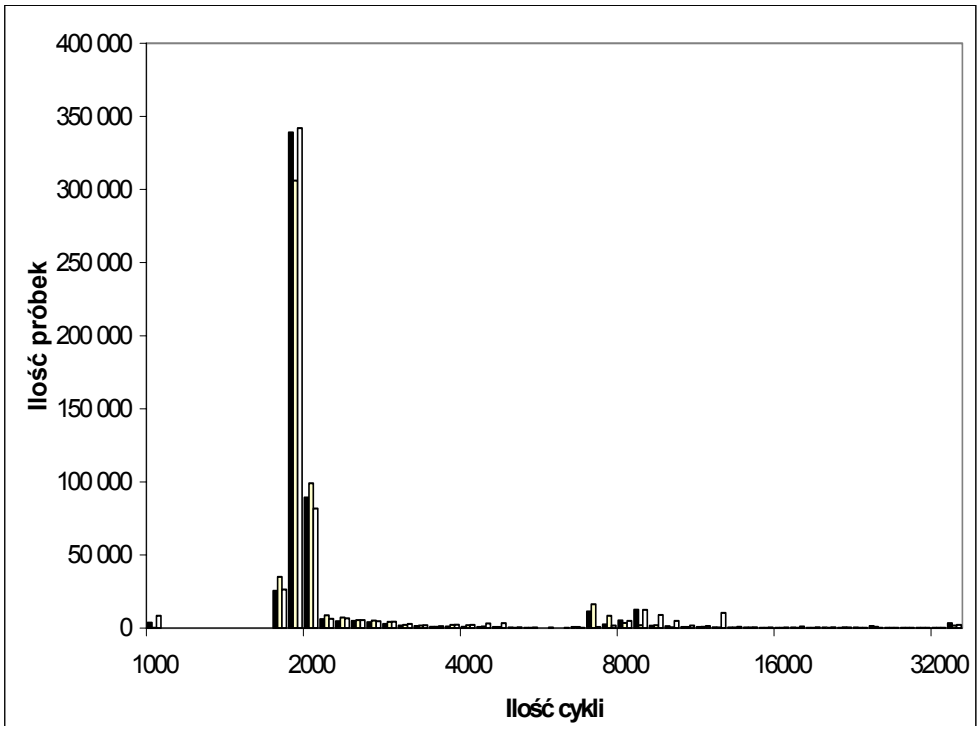
wykonywanego przez procesor dokonano pomiarów różnych fragmentów programu zawierającego różne instrukcje o sumarycznej długości ok. 510 cykli. Dokonano miliard (10^9) pomiarów następujących programów:

- 10-krotne wykonanie procedury pomiarowej (t_{min}=516 cykli)
- Obliczenie 27 elementów ciągu Fibonacciego (t_{min}=518 cykli)

Podczas pomiarów zapisywano czas realizacji fragmentów dla których wystąpiło opóźnienie. Wykres przedstawia porównanie wartości opóźnień przy realizacji:

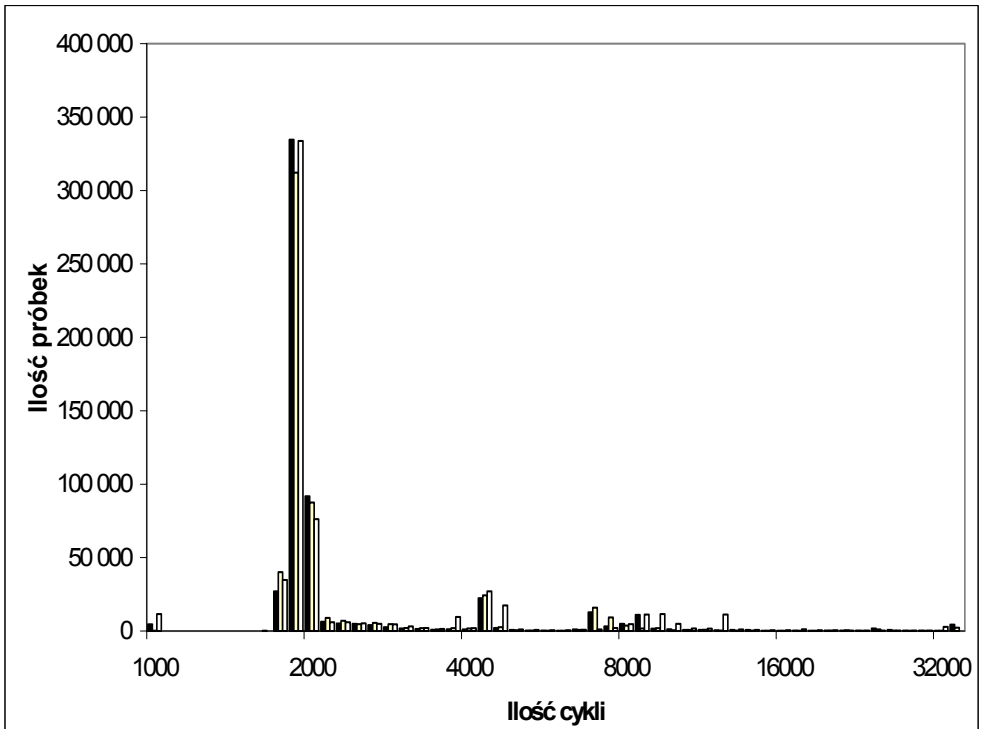
- 10 krotnej procedury pomiarowej – czarne słupki
- Obliczania ciągu Fibonacciego – szare słupki
- Wartości przewidywanej na podstawie modelu – białe słupki

Pomiarów dokonano w nocy, kiedy komputer nie był obciążony dodatkowymi pracami i wykonywał jedynie standardowe czynności systemu operacyjnego.



Rys. 10-18. Czas realizacji fragmentu zmierzony i przewidywany

Dokonano również pomiarów przy obciążeniu komputera transferem danych poprzez sieć (serwer FTP). Zaobserwowano przy tym pojawienie się dodatkowego maksimum gęstości związanego z obsługą zdarzeń sieciowych.

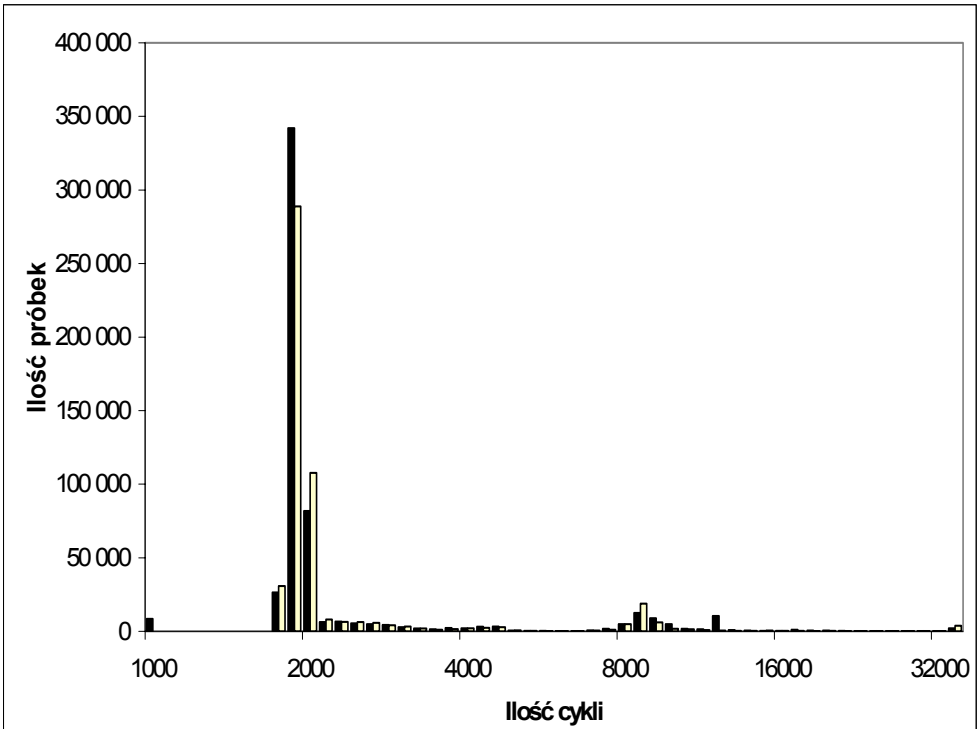


Rys. 10-19. Czas realizacji fragmentu przy obciążeniu sieci komputerowej

- Zarówno w przypadku nieobciążonego, jak i obciążonego komputera zaobserwowano bardzo dużą zgodność wyników przewidywanych oraz zmierzonych dla najczęściej występującego opóźnienia 2000 cykli i gorszą dla pozostałych opóźnień. Gorsza dokładność dla pozostałych opóźnień wynikać może ze zbyt małej ilości próbek lub wpływu rodzaju wykonywanych instrukcji na czas trwania czynności systemu operacyjnego. Ze względu na dużą liczbę pomiarów czas trwania każdego doświadczenia przekraczał dobę. Można zatem brać pod uwagę zachodzącą w międzyczasie zmianę parametrów systemu (np. zawartości i uporządkowania struktur systemu) mogącą w znaczący sposób wpływać na czas realizacji przerwań.
- Pomiary wykazały, że obciążenie systemu dodatkowymi pracami powoduje pojawienie się dodatkowych opóźnień. W prowadzonych doświadczeniach występowało obciążenie systemu przez inne programy, takie same efekty można osiągnąć w przypadku gdy analizowany program wymusza dodatkową aktywność systemu operacyjnego.
- Maksimum gęstości w okolicy 8000 cykli jest wyraźnie przesunięte w stronę mniejszych wartości ok. 1400 cykli przy pomiarze dłuższego fragmentu, niż by to wynikało z analizy krótszych fragmentów. Wynika z tego, że pewna funkcja systemu operacyjnego wykonuje się średnio o 1400 cykli szybciej. Prawdopodobnie jest to spowodowane rekonfiguracją systemu zwiększającą jego efektywność, jaka zaszła pomiędzy pierwszą a drugą serią pomiarów. System był w tym czasie resetowany i podczas uruchamiania mogło nastąpić korzystniejsze rozmieszczenie buforów systemowych w pamięci, oraz uporządkowanie tablic systemowych zwiększające szybkość ich przeglądania. Wynika z tego, iż istotna jest długookresowa, powolna zmienność parametrów systemu.

W celu sprawdzenia zgodności czasów opóźnień występujących przy pomiarach małych fragmentów i większych dokonano sto milionów (10^8) pomiarów czasu realizacji fragmentu o nominalnym czasie wykonania

5106 cykli (100 razy dłuższy niż pomiar składowy). Zmierzone czasy opóźnień (czarne słupki) oraz przewidywane na podstawie pomiarów składowych (białe słupki) przedstawiono na wykresie.



Rys. 10-20. Histogram czasu opóźnienia dla 5100 cykli oraz przewidywany na podstawie pomiarów składowych o długości 51 cykli.

Zaobserwowano bardzo dużą zgodność zmierzonego i przewidywanego rozkładu losowego opóźnień dla fragmentu o długości 5100 cykli. Rzeczywiste wartości są o 30-50 cykli mniejsze od przewidywanych stąd różnice w wysokościach sąsiadujących słupków histogramu. Uzyskano bardzo dobrą zgodność kształtu rozkładu dla opóźnień 2000, 4000 i 9500 cykli. Stanowi to poparcie tezy, że możliwe jest określenie parametrów maszyny i na ich podstawie przewidywanie kształtu rozkładu losowego czasu realizacji dowolnego fragmentu kodu o znanej złożoności. Potwierdzono również przydatność zaproponowanego modelu dla potrzeb reprezentacji rozkładów losowych czasu realizacji fragmentu kodu.

Ocena modelu

Zaproponowany model opóźnień w wykonywaniu kodu i transmisji pakietów umożliwia wierne odwzorowanie rzeczywistych rozkładów losowych uzyskanych w wyniku pomiarów. Obliczając przewidywany czas realizacji dłuższych fragmentów kodu na podstawie badania realizacji krótszych procedur w oparciu o proponowany model uzyskano wyniki bliskie rzeczywistym. Dokładność modelu można zatem uznać za dobrą.

Czas realizacji zadania w systemie jest sumą czasu jaki jest wymagany dla wykonania zadanego kodu oraz czasu jaki procesor poświęci innym czynnościom będąc w trakcie realizacji badanego fragmentu kodu. Zatem zaproponowany model odzwierciedla zjawiska mające wpływ na działanie maszyny. Jest on zatem lepszy od modeli czysto matematycznych, powstałych na zasadzie najlepszego dopasowania krzywej przez arbitralnie wybraną funkcję.

Prezentowany model jest bardziej skomplikowany niż większość prezentowanych dotychczas funkcji przybliżających rozkład losowy czasu realizacji fragmentu kodu [Dod85] [Kam85a] [Kam85b] [Mag93]. Duża

liczba parametrów jest niewątpliwie wadą opisywanego modelu utrudniającą uzyskanie danych wejściowych oraz obliczenia. Zaletą rozpatrywanego rozwiązania jest fakt, że składniki wpływające na całkowity czas realizacji są sumowane. Można zatem w trakcie obliczeń wykorzystywać przemienność i łączność operacji sumowania zmiennych losowych.