Złożoność asynchronicznych sieci logicznych

Maciej Gębala

Praca napisana w Instytucie Informatyki
Uniwersytetu Wrocławskiego
pod kierunkiem dr Macieja Liśkiewicza

Wrocław 1995

Spis treści

Wstęp

Synchroniczna bramka logiczna generuje wartość wynikową w chwili, gdy otrzyma wszystkie wartości wejściowe. Dlatego czas działania sieci zbudowanych z takich bramek nie zależy od aktualnych wartości danych wejściowych lecz jest stały, równy długości najdłuższej ścieżki obliczeń, czyli głębokości. Stąd starania, aby sieci synchroniczne miały jak najmniejszą głębokość.

W sieciach asynchronicznych bramki generują rezultat w momencie, gdy na podstawie już poznanych wartości wejściowych mogą jednoznacznie taki rezultat określić. Dla takich sieci wskazana jest więc analiza oczekiwanego czasu działania sieci, przeprowadzona dla określonej klasy rozkładów prawdopodobieństwa danych wejściowych. W pracy zajmujemy się taką analizą. Wprowadzamy również podział rozkładów prawdopodobieństwa na odpowiednie klasy, które wykorzystujemy w rozważaniach na temat złożoności czasowej sieci asynchronicznych. Odpowiednie definicje potrzebne do analizy złożoności takich sieci pochodzą z [JRS94].

W pracy prezentujemy rezultaty dotyczące złożoności sieci asynchronicznych uzyskane wcześniej przez innych autorów. Przedstawiamy podstawowe funkcje logiczne takie jak OR, AND, PARITY, MAJORITY, THRESHOLD i omawiamy ich złożoność w średnim przypadku. Oprócz tego zamieszczamy wyniki dla równoległego problemu sum prefiksowych i pokazujemy ich zastosowanie do obliczania sumy dwóch liczb ([JRSW93]).

Na koniec prowadzimy rozważania na temat czasu oczekiwanego dla sieci sortujących zbudowanych z asynchronicznych komparatorów. Koncentrujemy się tutaj na danych wejściowych o rozkładzie jednorodnym. Szczególną uwagę poświęcamy przy tym sieciom scalającym Odd–Even i bitonicznym.

W pracy przyjmujemy zasadniczo model sieci, w którym bramki mają stały fanin i fanout. Jednak w niektórych przypadkach czynimy odstępstwo od tej zasady. Wówczas jednak wyraźnie to zaznaczamy.

1 WŁASNOŚCI SIECI

1.1 Podstawowe definicje

W niniejszym rozdziale zdefiniujemy podstawowe pojecia dotyczące funkcji boolowskich oraz sieci logicznych.

Definicja 1 Pojęcia podstawowe

Niech B_n^m *oznacza zbiór funkcji boolowskich* $f : \{0,1\}^n \to \{0,1\}^m$ *i niech* $B_n = B_n^1$.

Niech $\mathcal{D}_n$ *oznacza zbiór wszystkich rozkładów prawdopodobieństwa* μ *określonych na* $\{0,1\}^n$. *Jednorodny rozkład na* $\{0,1\}^n$, *który określa równe prawdopodobieństwa wystąpienia dla wszystkich* 2^n *możliwych wektorów oznaczamy przez* μ_n^{UNI}.

Dla dowolnego $\mu \in \mathcal{D}_n$ *niech* $\mathrm{Supp}(\mu)$ *będzie zbiorem wszystkich wektorów* $X \in \{0,1\}^n$ *dla których* $\mu(X) \neq 0$. *Nazwijmy rozkład* $\mu \in \mathcal{D}_n$ *ściśle pozytywnym jeśli* $\mathrm{Supp}(\mu) = \{0,1\}^n$.

Bazą sieci (lub krócej bazą) nazywamy dowolny zbiór funkcji logicznych, z których zbudowana jest sieć. Niech $\mathrm{Cir}(f)$ *oznacza zbiór wszystkich sieci nad ustaloną bazą, które liczą* f. *Dla sieci należącej do* $\mathrm{Cir}_\mu(f)$ *żądamy, żeby rezultaty równały się* $f(X)$ *tylko dla wejściowych wektorów* $X \in \mathrm{Supp}(\mu)$.

Jak łatwo zauważyć, informacje w sieciach zbudowanych z bramek asynchronicznych mogą być rozpowszechniane dużo szybciej niż w sieciach synchronicznych. Dla przykładu, jeśli jedno z wejść bramki OR jest znane wcześniej niż drugie i ma ono wartość 1, wtedy wyjście jest zdeterminowane i nie zależy od innych wartości wejściowych. Pisząc bardziej formalnie, możemy dla każdej bramki v w asynchronicznej sieci C zdefiniować funkcję czasu

$$\mathbf{time} : \{0,1\}^n \to \mathbb{N}$$

która wyznacza dla każdego wejścia $X \in \{0,1\}^n$ numer pierwszego kroku, w którym bramka v może podać swój końcowy rezultat $\mathrm{res}_v(X)$.

Definicja 2 *Niech* C *będzie siecią i niech* v *będzie bramką w* C *oraz niech* X *będzie dowolnym wejściem sieci* C. *Niech* $\mathrm{res}_v(X)$ *będzie rezultatem uzyskanym przez bramkę* v *w sieci* C *z danymi wejściowymi* X. *Dla bramek dających stały rezultat i bramek wejściowych* v *ustalamy* $\mathbf{time}_v(X) := 0$. *Dla pozostałych bramek* v *z* k *bezpośrednimi przodkami* $v_1, \ldots, v_k$ *definiujemy*

$$\mathbf{time}_v(X) := 1 + \min\{\ t \ \mid \ \textit{obliczone wartości } \mathrm{res}_{v_i}(X) \textit{ w czasie } \mathrm{time}_{v_i}(X) \leqslant t \textit{ jednoznacznie wyznaczają } \mathrm{res}_v(X)\}.$$

Dla sieci C z bramkami wyjściowymi $y_1, \ldots, y_m$ definiujemy globalną funkcję czasu w następujący sposób

$$\mathbf{time}_C(X) := \max_i \ \mathbf{time}_{y_i}(X).$$

Dla przykładu, czas dla bramki v typu OR z poprzednikami v_1 i v_2 jest dany przez następującą zależność

$$\mathbf{time}_v(X) := 1 + \begin{cases} \max\{\mathbf{time}_{v_1}(X), \mathbf{time}_{v_2}(X)\} & gdy\ \mathrm{res}_{v_1}(X) = \mathrm{res}_{v_2}(X) = 0 \\ \min\{\mathbf{time}_{v_i}(X) \mid \mathrm{res}_{v_i}(X) = 1\} & w\ p.p. \end{cases}$$

Tak zdefiniowana funkcja opisuje nam minimalny czas po upływie którego wewnętrzna bramka jest zdolna do podania swojej wartości końcowej. Bramka nie przekazuje żadnego sygnału do swego następnika w przypadku, gdy zna już swój rezultat końcowy. Sieci z takimi bramkami będziemy nazywać **sieciami z czasem niejawnym**.

Dla przeprowadzenia analizy własności sieci możemy ograniczyć nasze rozważania tylko do standardowej bazy bramek, czyli AND, OR i NOT. Wynika to z następującego twierdzenia:

Twierdzenie 1 *[JRS94] Dla dowolnej pary skończonych baz B_1,B_2 istnieje stała k o następującej własności: Dla każdej sieci C_1 zbudowanej z bramek należących do B_1 istnieje równoważna sieć C_2 zbudowana z bramek należących do B_2 o następującej własności*

$$\forall_X \quad \mathbf{time}_{C_1}(X) \leqslant k \cdot \mathbf{time}_{C_2}(X).$$

1.2 Bramki z jawnym czasem informacji

Aby otrzymać bramki logiczne, które w sposób jawny sygnalizują moment obliczenia swojego wyniku, należy rozszerzyć logikę dwuwartościową do logiki trójwartościowej, poprzez dodanie symbolu "?", który oznacza "jeszcze nie wiem". Na przykład, dla funkcji OR i AND dostaniemy wtedy następujące tabelki funkcji

OR	?	0	1
?	?	?	1
0	?	0	1
1	1	1	1

AND	?	0	1
?	?	0	?
0	0	0	0
1	?	0	1

Zakładamy, że wszystkie bramki wewnętrzne w sieci są inicjowane wartością "?".

Sieci z bramkami, które sygnalizują obliczenie swego wyniku, nazywamy **sieciami czasu jawnego**. Realizując sieć czasu jawnego w praktyce, używamy binarnego kodowania trójwartościowej logiki. Prostym rozwiązaniem jest dodanie dodatkowego bitu

– OK-bit, który wskazuje czy drugi bit – DATA-bit jest aktualnym rezultatem. W ten sposób otrzymamy następujące kody: kod(?)={ (0,0) , (0,1) }, kod(0)={ (1,0) }, kod(1)={ (1,1) }.

1.3 Równoważność modeli czasowych

Do realizacji poszerzonych trójwartościowych funkcji OR i AND użyjemy jednak innego kodowania, które wymaga tylko dwóch równoległych bramek dla obliczenia kodowanej wersji OR lub AND.

Do nowego kodowania użyjemy pary bitów (OK $\wedge$ DATA , OK $\wedge$ $\neg$ DATA) zamiast (OK , DATA). W ten sposób 0 jest kodowane jako (0,1), wartość 1 jako (1,0) a "?"jako (0,0). Kod (1,1) w ogóle nie występuje. Poniżej mamy tabelki opisujące sposób realizacji bramek AND i OR (negacja może być realizowana w sposób dowolny). Dla $z = x \vee y$ słowo kodowe (z_1, z_2) jest liczone ze słów kodowych (x_1, x_2),(y_1, y_2), będących przedstawieniem x i y, w następujący sposób

$$z_1 = x_1 \vee y_1 \qquad z_2 = x_2 \wedge y_2$$

x	(x_1, x_2)	y	(y_1, y_2)	z	(z_1, z_2)
0	(0,1)	0	(0,1)	0	(0,1)
1	(1,0)	*	$\neq$(1,1)	1	(1,0)
?	(0,0)	$\neq$1	$\neq$(1,*)	?	(0,0)

gdzie * oznacza dowolną wartość. Analogicznie dla $z = x \wedge y$ mamy

$$z_1 = x_1 \wedge y_1 \qquad z_2 = x_2 \vee y_2$$

x	(x_1, x_2)	y	(y_1, y_2)	z	(z_1, z_2)
0	(0,1)	*	$\neq$(1,1)	0	(0,1)
1	(1,0)	1	(1,0)	1	(1,0)
?	(0,0)	$\neq$0	$\neq$(*,1)	?	(0,0)

Wykorzystując powyższy sposób kodowania otrzymujemy następującą prostą zależność, która znacznie ułatwia nam dalsze rozważania

Twierdzenie 2 *[JRS94] Każda trójwartościowa sieć czasu jawnego C (zbudowana z bramek należących do standardowej bazy AND, OR i NOT) może być symulowana przez konwencjonalną dwuwartościową sieć C' o tej samej głębokości i czasie działania jak C oraz o co najwyżej dwa razy większym rozmiarze.*

W związku z powyższym twierdzeniem nasze dalsze rozważania ograniczamy do sieci z czasem niejawym, które są zbudowane z bramek należących do bazy standardowej.

2 MIARA ŚREDNIEJ ZŁOŻONOŚCI

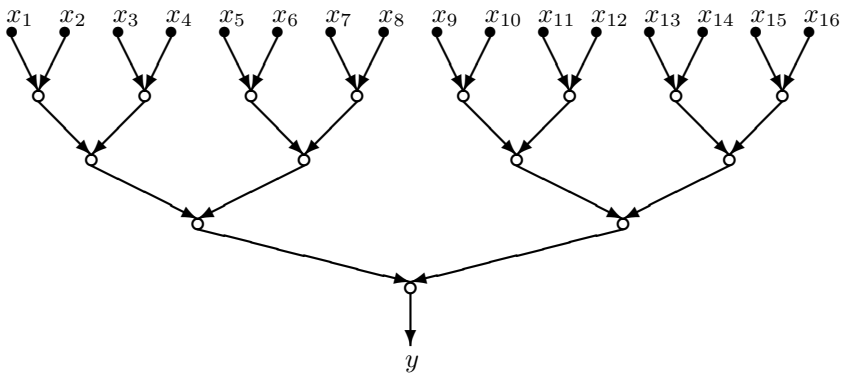
Definicja 3 *Dla funkcji* $t : \{0,1\}^n \to \mathbb{N}$ *i rozkładu prawdopodobieństwa* $\mu : \{0,1\}^n \to [0;1]$ *niech*

$$\mathrm{E}_\mu(t) := \sum_{X \in \{0,1\}^n} t(X)\mu(X)$$

będzie wartością oczekiwaną t *dla rozkładu* μ*. Jeśli* D *jest zbiorem rozkładów prawdopodobieństwa, to definiujemy*

$$\mathbf{etime}(f, D) := \max_{\mu \in D} \min_{C \in \mathrm{Cir}_\mu(f)} \mathrm{E}_\mu(\mathbf{time}_C)$$

jako optymalny **czas oczekiwany sieci** *dla funkcji* f *z uwzględnieniem rozkładów należących do* D*.*

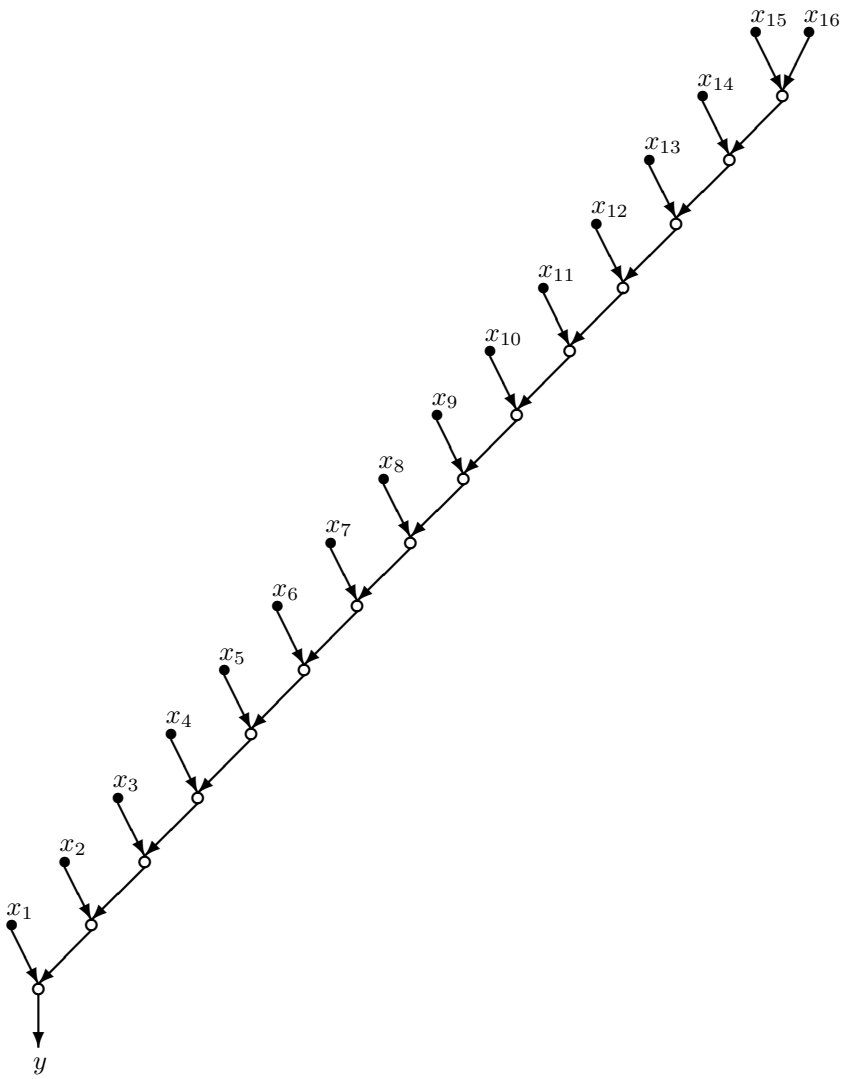


Rysunek 1: Tradycyjna sieć licząca funkcję OR_n o złożoności najgorszego przypadku równej dokładnie $\lceil \log n \rceil$ (przykład dla $n = 16$).

Przykładem funkcji logicznej, która może być policzona dużo szybciej za pomocą sieci z bramkami asynchronicznymi niż synchronicznymi jest n-argumentowa funkcja OR zdefiniowana następująco

$$\mathrm{OR}_n(x_1, \ldots, x_n) := x_1 \vee \ldots \vee x_n$$

W przykładzie tym posłużymy się bramkami ze standardowej bazy {OR,AND,NOT}. Klasyczna, optymalna synchroniczna sieć licząca OR_n ma postać pełnego drzewa binarnego i jej przykład jest pokazany na rysunku 1. Jak łatwo zauważyć, wszystkie ścieżki w tej sieci prowadzące od liści od korzenia mają długość $\lceil \log n \rceil$ lub $\lfloor \log n \rfloor$, w związku z czym czas obliczeń jest stały i wynosi $\lceil \log n \rceil$.



Rysunek 2: Optymalna sieć asynchroniczna licząca funkcję OR_n dla danych o rozkładzie jednorodnym (przykład dla $n = 16$).

Używając do obliczania funkcji OR_n sieci $C = C_n$ z wejściami $x_1, \ldots, x_n$, zdefiniowanej w następujący sposób

$$\begin{array}{lcll} C_2 & = & x_{n-1} \vee x_n & \\ C_i & = & x_{n-i+1} \vee C_{i-1} & i = 3, \ldots, n \end{array}$$

(przykład takiej sieci dla $n = 16$ przedstawiono na rysunku 2), otrzymamy optymalne ograniczenie czasu oczekiwanego dla funkcji OR_n:

Twierdzenie 3 *[JRS94]*

$$\mathbf{etime}(\mathrm{OR}_n, \mu_n^{\mathrm{UNI}}) \leqslant 2 - 2^{-n+2}.$$

Dowód: Zgodnie z definicją wartości **etime**, dla sieci C mamy

$$\begin{array}{l} \mathbf{etime}(\mathrm{OR}_n, \mu_n^{\mathrm{UNI}}) \leqslant \mathrm{E}_{\mu_n^{\mathrm{UNI}}}(\mathbf{time}_C) = \\ \quad = \sum\limits_{t=1}^{n-1} t \cdot \Pr[\text{dopiero } t\text{-ty bit jest równy 1}] \\ \qquad + (n-1) \cdot \Pr[\text{tylko ostatni bit jest równy 1}] \\ \qquad + (n-1) \cdot \Pr[\text{wszystkie bity są równe 0}] = \\ \quad = \sum\limits_{t=1}^{n-1} t \cdot \frac{1}{2^t} + (n-1) \cdot \frac{1}{2^n} + (n-1) \cdot \frac{1}{2^n} = \sum\limits_{t=1}^{n} \frac{t}{2^t} + \frac{n-2}{2^n} = \\ \qquad = 2 - \frac{n+2}{2^n} + \frac{n-2}{2^n} = 2 - 2^{-n+2} \end{array}$$

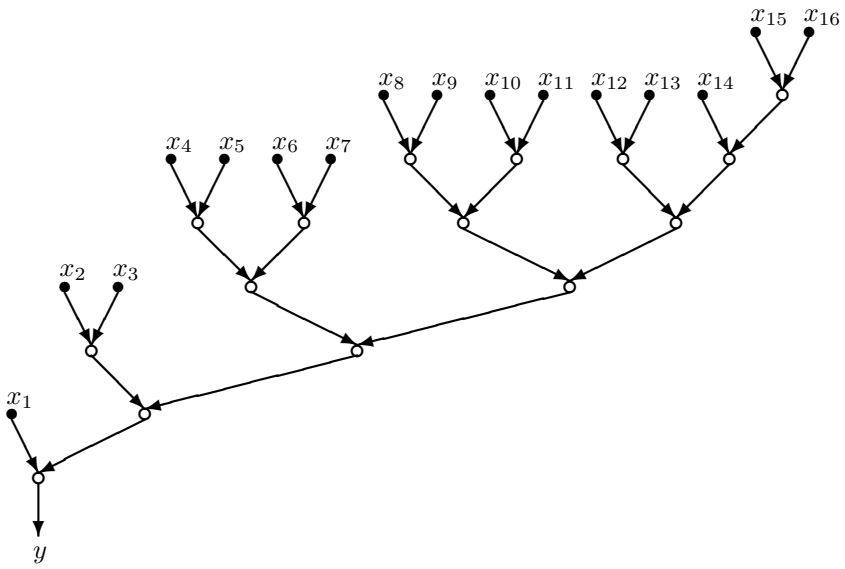
□

Jest jednak rzeczą oczywistą, że sieć zbudowana tak, jak na rysunku 2 ma bardzo złą złożoność w najgorszym przypadku, równą $n - 1$, co niekiedy jest niepożądane. Można temu zaradzić niewiele pogarszając średni czas. Dla sieci skonstruowanej tak, jak na rysunku 3, gdzie i-te wejście jest na poziomie co najwyżej $2\lfloor \log i \rfloor + 1$, maksymalny czas obliczeń wynosi też co najwyżej $2 \log n + 1$ a czas oczekiwany jest mniejszy niż 2.3.

Oczywiście podana tutaj definicja czasu oczekiwanego nie jest jedyną możliwą. Możemy na przykład zastanowić się nad istnieniem jednej optymalnej sieci dla danej funkcji logicznej i zbioru rozkładów. W takim wypadku definicja czasu oczekiwanego będzie miała postać

$$\mathbf{etime}(f, D) := \min_{C \in \mathrm{Cir}_\mu(f)} \max_{\mu \in D} \mathrm{E}_\mu(\mathbf{time}_C)$$

Ciekawe wyniki dla takiej definicji czasu oczekiwanego uzyskano dla alternatywy i koniunkcji w pracy [BHPS94].



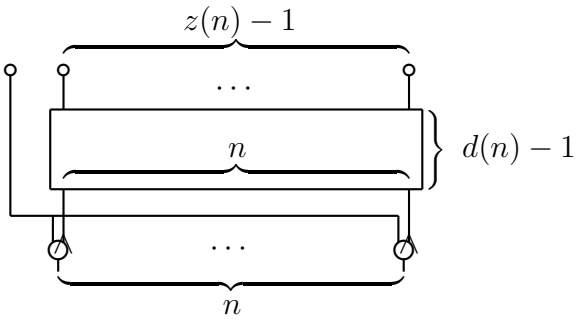
Rysunek 3: Sieć asynchroniczna licząca funkcję OR_n o logarytmicznej głębokości (przykład dla $n = 16$).

3 ZŁOŻONOŚĆ ROZKŁADÓW

Omawiane do tej pory wyniki dotyczyły tylko sieci z wejściami danymi przez rozkład jednorodny. W rzeczywistości jednak wejścia mają często inne, bardziej skomplikowane rozkłady. W związku z tym, analizując złożoność średniego przypadku dla sieci asynchronicznych, należy zdefiniować również miarę złożoności dla rozkładów prawdopodobieństwa, które generują dane wejściowe. Złożoność tą możemy mierzyć za pomocą głębokości sieci, które startując z wektorem o rozkładzie jednorodnym na wejściu, na wyjściu dają nam potrzebny rozkład.

Definicja 4 *Niech sieć C posiadająca r bramek wejściowych i n bramek wyjściowych realizuje przekształcenie zmiennej losowej Z zdefiniowanej na $\{0,1\}^r$ w zmienną losową X zdefiniowaną na $\{0,1\}^n$ w następujący sposób: Wejściowy wektor dla sieci C jest wybierany zależnie od Z. Wtedy X równa się wartości otrzymywanej na bramkach wyjściowych C.*

Jeśli Z ma rozkład jednorodny określony na $\{0,1\}^r$, to sieć nazywamy **siecią generującą rozkład** *(DG-siecią), która generuje rozkład odpowiadający zmiennej losowej X.*



Rysunek 4: Przykład DG-sieci z nieograniczonym fanout posiadającej $z(n) = n + 1$ losowych bitów wejściowych i n-bitowe wyjście.

Dla uzależnienia skomplikowania rozkładu prawdopodobieństwa od głębokości sieci generującej ten rozkład, nie będziemy rozważać DG-sieci zbudowanych z bramek o nieograniczonym fanout lecz ograniczymy się do DG-sieci z bramkami o stałym fanout równym dwa. W przeciwnym razie dopuścilibyśmy do przypadków, w których prosta sieć może generować bardzo niezrównoważony rozkład. Jeśli na rysunku 4 podstawimy $d(n) = 1$, to wyjściowy rozkład wyraża się następującymi własnościami

$$\Pr[X = 0^n] = \frac{1}{2} + 2^{-(n+1)} \qquad i \qquad \Pr[X = Y] = 2^{-(n+1)} \text{ dla dowolnego } Y \neq 0^n$$

Dla funkcji OR_n powyższy rozkład daje dolną granicę $\frac{1}{2}\log n$ dla czasu oczekiwanego.

Z pojęciem DG-sieci wiążemy teraz rozkłady w następujący sposób

Definicja 5

$$\mathcal{D}Depth_n(d) := \{\ \mu \in \mathcal{D}_n \ \mid \ \textit{istnieje } r\textit{-wejściowa i } n\textit{-wyjściowa DG-sieć } C \textit{ o głębokości } d\textit{, która przekształca zmienną losową } Z \textit{ określoną na } \{0,1\}^r \textit{ o rozkładzie } \mu_r^{\mathrm{UNI}} \textit{ w zmienną losową } X \textit{ określoną na } \{0,1\}^n \textit{ o rozkładzie } \mu\ \}.$$

Dla sieci należących do tak zdefiniowanych klas zachodzą następujące własności

Lemat 1 *[JRS94] Niech rozkład μ, odpowiadający zmiennej losowej X, należy do $\mathcal{D}Depth_n(d)$ i niech X_i oznacza i-ty bit zmiennej losowej X. Ponadto zdefiniujmy $I_X := \{i \mid 0 < \Pr[X_i = 1] < 1\}$. Wtedy*

1. *dla każdego $i \in I_X$ zachodzi*

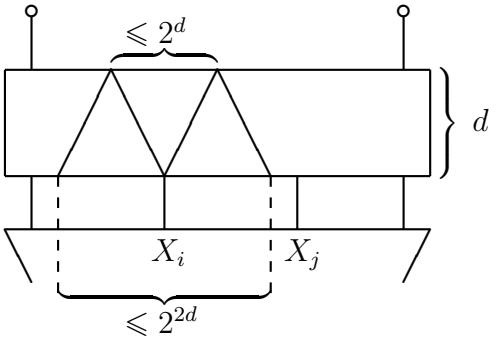
$$2^{-2^d} \leqslant \Pr[X_i = 1] \leqslant 1 - 2^{-2^d}.$$

2. *istnieje zbiór $k \geqslant |I_X| \cdot 2^{-2d}$ pozycji bitów $X_{i_1}, \ldots, X_{i_k}$ które są niezależne.*

Dowód: Na początek pokażemy pierwszą część lematu. Przez indukcję po głębokości sieci udowodnimy, że jeśli bramka w DG-sieci jest na głębokości d, to przyjmuje ona wartości wynikowe z prawdopodobieństwem, które jest całkowitą wielokrotnością 2^{-2^d}.

1. Niech $d = 0$. Wtedy prawdopodobieństwo z jakim wyjściowy bit równy jest 0 wynosi $\frac{1}{2}$, ponieważ ma on rozkład jednorodny. Stąd własność którą dowodzimy jest prawdziwa dla $d = 0$.

2. Załóżmy, że $d > 0$. Rozpatrzmy bramkę v znajdującą się na głebokości d. Posiada ona dwa wejścia generowane przez bramki na głebokości co najwyżej $d - 1$. Zgodnie z założeniem indukcyjnym bramki te przyjmują swoje wartości z prawdopodobieństwem będącym całkowitą wielokrotnością $2^{-2^{d-1}}$. Bramka v może więc przyjmować wartości z prawdopodobieństwem będącym sumą iloczynów prawdopodobieństw swoich wejść, czyli całkowitą wielokrotnością $2^{-2^{d-1}} \cdot 2^{-2^{d-1}} = 2^{-2^d}$. Stąd mamy prawdziwość dowodzonej własności dla wszystkich d.



Rysunek 5: Ilustracja zależności bitów wyjściowych dla DG-sieci należącej do $\mathcal{D}Depth(d)$.

W szczególności, jeśli prawdopodobieństwo pewnej wyjściowej wartości dla bramki na poziomie d jest różne od zera, to musi przyjmować wartość co najmniej równą 2^{-2^d}.

Drugą część lematu dowodzimy w następujący sposób. Wybieramy dowolny bit wyjściowy X_i. Bit ten zależy od co najwyżej 2^d losowych bitów wejściowych Z_j. Wynika to z ograniczenia na fanin, wynoszącego co najwyżej dwa. Ponieważ fanout bramek jest również ograniczony przez dwa, każdy taki Z_j może wpływać na co najwyżej 2^d bitów wyjściowych. Stąd mamy co najwyżej $2^{2d}-1$ różnych X_k takich, że są zależne od X_i (schemat tego rozumowania jest przedstawiony na rysunku 5). □

4 PROSTE FUNKCJE LOGICZNE

4.1 Funkcje z podlogarytmicznym czasem oczekiwanym

Na początek rozpatrzymy OR_n – jedną z prostszych funkcji logicznych i pokażemy, że może ona być policzona w średnim przypadku bardzo efektywnie przez sieć asynchroniczną dla dużej klasy rozkładów prawdopodobieństwa. Przypomnijmy, że złożoność funkcji OR_n liczonej na sieciach synchronicznych wynosi $\lceil \log n \rceil$.

Twierdzenie 4 *[JRS94] Dla* $0 \leqslant d \leqslant \log\log n$ *mamy*

$$2^{d-1} - 1 \leqslant \mathbf{etime}(\mathrm{OR}_n, \mathcal{D}Depth_n(d)) \leqslant 3 + 2^d.$$

Dowód: Na początek udowodnimy dolne ograniczenie. W tym celu musimy znaleźć taki rozkład należący do $\mathcal{D}Depth_n(d)$, dla którego średni czas obliczania funkcji OR_n będzie duży. Niech $X = X_1 \ldots X_n$ będzie zmienną losową, dla której $\Pr[X_i = 1] = 2^{-2^d}$ i niech wszystkie X_i będą niezależne. Rozkład taki może być łatwo wygenerowany przez DG-sieć o głębokości d posiadającą $n \cdot 2^d$ bramek wejściowych. Sieć taka składa się z n pełnych drzew binarnych o głębokości d zbudowanych z bramek AND. Dla X generowanego przez taką sieć prawdopodobieństwo, że ustalony podciąg jego bitów o długości 2^{2^d-1} jest identyczny z ciągiem $0^{2^{2^d-1}}$ wynosi

$$(1 - 2^{-2^d})^{2^{2^d-1}} = ((1 - \frac{1}{2^{2^d}})^{2^{2^d}})^{\frac{1}{2}}$$

Łatwo zauważyć podobieństwo tego wzoru do $(1 - \frac{1}{n})^n$, dla którego można pokazać, że dla $n \geqslant 2$ jego wartość jest większa lub równa $\frac{1}{4}$. W związku z tym prawdopodobieństwo uzyskania ciągu 2^{2^d-1} zer wynosi co najmniej $\frac{1}{2}$. Czytając więc tylko takiej długości sekwencję łańcucha wejściowego nie możemy dla co najmniej połowy wejść jednoznacznie wyznaczyć wyniku w sieci liczącej OR_n. Stąd otrzymamy dolną granicę, ponieważ wszystkie bramki mają ograniczony fanin.

$$\mathbf{etime}(\mathrm{OR}_n, \mathcal{D}Depth_n(d)) \geqslant \frac{1}{2} \cdot \log 2^{2^d-1} = \frac{1}{2} \cdot (2^d - 1) \geqslant 2^{d-1} - 1.$$

Aby wyznaczyć górne ograniczenie musimy rozpatrzyć dowolny rozkład X należący do $\mathcal{D}Depth_n(d)$. Załóżmy dodatkowo, że dla każdego $k \in [1 \ldots n]$ $0 < \Pr[X_k = 0] < 1$. Możemy przyjąć takie założenie, ponieważ jeśli dla jakiegoś k $\Pr[X_k = 0]$ wynosiłoby 0, to wtedy X_k byłoby zawsze równe 1 i funkcja OR_n dla takich danych byłaby stała, jej wartość byłaby równa 1. Wówczas nie potrzebowalibyśmy żadnej

sieci, aby policzyć $\mathrm{OR}_n(X)$. Jeśli natomiast $\Pr[X_k = 0]$ byłoby równe 1, to X_k moglibyśmy pominąć w obliczeniach. Tak więc korzystając z powyższego założenia i z lematu 1, otrzymamy, że istnieje co najmniej $n \cdot 2^{-2d}$ niezależnych bitów wejściowych X_i. Podzielmy je na grupy Y_j o rozmiarze 2^{2^d} każda i obliczmy dla każdej z nich funkcję $\mathrm{OR}(Y_j)$ za pomocą tradycyjnej sieci w postaci pełnego drzewa binarnego o głębokości 2^d. Niech W będzie zbiorem pozostałych bitów. Ich alternatywa może być policzona przez sieć o głębokości co najwyżej $\log n$. Teraz zrealizujemy funkcję OR_n w następujący sposób

$$\mathrm{OR}(X) := \mathrm{OR}(Y_1) \vee (\mathrm{OR}(Y_2) \vee (\ldots(\mathrm{OR}(Y_p) \vee \mathrm{OR}(W))\ldots))$$

gdzie $p \geqslant n \cdot 2^{-2d-2^d}$. Ponieważ podsieci liczące $\mathrm{OR}(Y_j)$ dają nam w wyniku jedynkę z prawdopodobieństwem co najmniej $\frac{1}{2}$, to jeśli zastosujemy sieć z rysunku 2 do zsumowania wyników z tych podsieci, otrzymamy

$$\mathbf{etime}(\mathrm{OR}_n, \mathcal{D}Depth_n(d)) \leqslant 3 + 2^d$$

□

Często posługujemy się funkcjami będącymi złożeniami innych prostych funkcji. Dla nich możemy uzyskać granice średniej złożoności w następujący sposób

Twierdzenie 5 *[JRS94] Niech $f \in B_q^m$, $g \in B_p^q$, $h \in B_n^p$ będą funkcjami logicznymi i przyjmijmy, że h może być obliczona przez sieć o głębokości δ zbudowaną z bramek o ograniczonym fanout. Niech $depth(f)$ oznacza minimalną głębokość sieci liczącej funkcję f i analogicznie $depth(h) = \delta$ głębokość sieci liczącej h. Wtedy*

$$\mathbf{etime}(f \circ g \circ h, \mathcal{D}Depth(d)) \leqslant depth(f) + depth(h) + \mathbf{etime}(g, \mathcal{D}Depth(d + \delta)).$$

Dowód: Niech D będzie dowolną DG-siecią, która generuje rozkład prawdopodobieństwa należący do klasy $\mathcal{D}Depth(d)$. Oznaczmy przez $h(\mathrm{res}(D))$ sieć otrzymaną przez połączenie wyjść sieci D z wejściami sieci liczącej funkcję h. Ponieważ h jest liczona przez sieć o głębokości δ to otrzymana sieć $h(\mathrm{res}(D))$ musi zawierać się w klasie rozkładów $\mathcal{D}Depth(d+\delta)$. W ten sposób, korzystając z definicji czasu oczekiwanego, otrzymamy

$$\mathbf{etime}(g, \{h(\mathrm{res}(D))\}) \leqslant \mathbf{etime}(g, \mathcal{D}Depth(d + \delta)).$$

Obliczamy $g \circ h$ przez złożenie sieci liczących g i h. Zauważmy, że losowe wejścia dla g w obu sieciach, to jest $g \circ h$ i g, mają równe prawdopodobieństwa. Stąd

$$\mathbf{etime}(g \circ h, \{D\}) \leqslant \mathbf{etime}(g, \{h(\mathrm{res}(D))\}) + depth(h)$$

Ponieważ D wybraliśmy dowolnie ze zbioru $\mathcal{D}Depth(d)$, czas oczekiwany dla g może tylko wzrosnąć, jeśli dopuścimy wszystkie DG-sieci o głębokości $d+\delta$ zamiast tylko sieci $h(\mathrm{res}(D))$. Dlatego

$$\mathbf{etime}(g \circ h, \mathcal{D}Depth(d)) \leqslant \mathbf{etime}(g, \mathcal{D}Depth(d+\delta)) + depth(h)$$

Dla dokonania pełnego obliczenia $f \circ g \circ h$ musimy dołączyć jeszcze tylko sieć liczącą f i wtedy otrzymamy

$$\mathbf{etime}(f \circ g \circ h, \mathcal{D}Depth(d)) \leqslant \mathbf{etime}(g, \mathcal{D}Depth(d+\delta)) + depth(f) + depth(h).$$

□

Niech teraz AND_n oznacza n-arną funkcję AND a EQUAL_n $2n$-arną funkcję, która decyduje czy dwa n-bitowe łańcuchy są równe. Używając teraz powyższego twierdzenia oraz równości

$$\mathrm{OR}_n(x_1, \ldots, x_n) = \neg\mathrm{AND}_n(\neg x_1, \ldots, \neg x_n),$$

$$\mathrm{EQUAL}_n(x_1, y_1, \ldots, x_n, y_n) = \neg\mathrm{OR}_n(x_1 \oplus y_1, \ldots, x_n \oplus y_n),$$

gdzie $\oplus$ oznacza XOR, otrzymamy

Twierdzenie 6 *[JRS94]*

$$\begin{aligned}&\mathbf{etime}(\mathrm{OR}_n, \mathcal{D}Depth(d))\\ &\quad\leqslant \mathbf{etime}(\mathrm{EQUAL}_n, \mathcal{D}Depth(d))\\ &\quad\leqslant \mathbf{etime}(\mathrm{OR}_n, \mathcal{D}Depth(d+1)) + 2,\end{aligned}$$

$$\begin{aligned}&\mathbf{etime}(\mathrm{OR}_n, \mathcal{D}Depth(d))\\ &\quad\leqslant \mathbf{etime}(\mathrm{AND}_n, \mathcal{D}Depth(d+1)) + 2\\ &\quad\leqslant \mathbf{etime}(\mathrm{OR}_n, \mathcal{D}Depth(d+2)) + 4.\end{aligned}$$

Istnieje wiele podstawowych funkcji logicznych, które mają podobne oczekiwane złożoności czasowe jak OR, na przykład opisana w następnych rozdziałach funkcja rozstrzygająca, która z dwóch podanych liczb binarnych jest mniejsza.

4.2 Logarytmiczna dolna średnia granica

Istnieją jednak funkcje, dla których czas oczekiwany w sieciach asynchronicznych jest tego samego rzędu co czas obliczeń w sieciach synchronicznych. Funkcja parzystości — PARITY, dająca jedynkę wtedy i tylko wtedy gdy na wejściu pojawi się parzysta liczba jedynek, jest jedną z nich. Dzieje się tak dlatego, że wyjście zależy od każdego

bitu wejściowego. Zmiana tylko jednego bitu wejściowego zmienia nam wynik funkcji PARITY. W tym podrozdziale chcemy wprowadzić technikę wyznaczania dolnych granic średniej złożoności czasowej.

Niech $\mathrm{THRESHOLD}_n^a$ oznacza n-arną funkcję logiczną, która jest równa 1 jeśli co najmniej a wejściowych bitów jest równa 1, dla $0 \leqslant a \leqslant n$. Niech $\mathrm{MAJORITY}_n$ będzie specjalnym przypadkiem THRESHOLD z $a = \lfloor \frac{n}{2} \rfloor$. Niech $\mathrm{BITSORT}_n$ oznacza posortowanie n bitów, a MIN_n podanie mniejszej z dwóch n-bitowych liczb.

Definicja 6 *Niech $f \in B_n^m$. Definiujemy wówczas miarę zależności bitów wejściowych funkcji f w sposób następujący*

$$\mathbf{idepen}_j(f) := \min\{\ k \ \mid\ \exists_{a_{i_1},\ldots,a_{i_k}} \textit{ takie, że wartość } j\textit{-tego bitu wyjściowego } f_{|x_{i_1}=a_{i_1},\ldots,x_{i_k}=a_{i_k}} \textit{ jest stała }\}$$

$$\mathbf{idepen}(f) := \max_{1 \leqslant j \leqslant m} \mathbf{idepen}_j(f)$$

Definiujemy również miarę zależności bitów wyjściowych funkcji f

$$\mathbf{odepen}(f) := \max\{\ k \ \mid\ \textit{istnieje taki bit wejściowy, że jego wartość niezależnie od wartości innych bitów wejściowych wpływa bezpośrednio na wartości } k \textit{ bitów wyjściowych}\}$$

Wniosek 1

$$\begin{aligned}
\mathbf{idepen}(\mathrm{OR}_n) &= 1,\\
\mathbf{idepen}(\mathrm{PARITY}_n) &= n,\\
\mathbf{idepen}(\mathrm{THRESHOLD}_n^a) &= \min(a, n-a+1),\\
\mathbf{idepen}(\mathrm{MAJORITY}_n) &= \lfloor \tfrac{n}{2} \rfloor,\\
\mathbf{idepen}(\mathrm{BITSORT}_n) &\geqslant \lceil \tfrac{n}{2} \rceil,\\
\mathbf{odepen}(\mathrm{MIN}_n) &= n.
\end{aligned}$$

Dowód: Jak łatwo zauważyć, w przypadku funkcji OR_n ustalenie wartości tylko jednego bitu wejściowego na 1 jednoznacznie wyznacza nam wynik. Tak więc uzyskujemy $\mathbf{idepen}(\mathrm{OR}_n) = 1$.

W przypadku funkcji PARITY_n zmiana jakiegokolwiek bitu wejściowego zmienia nam wynik funkcji, więc $\mathbf{idepen}(\mathrm{PARITY}_n) = n$.

W funkcji $\mathrm{THRESHOLD}_n^a$ z kolei, aby jednoznacznie wyliczyć wynik musimy mieć albo a jedynek albo $n-a+1$ zer. Stąd $\mathbf{idepen}(\mathrm{THRESHOLD}_n^a) = \min(a, n-a+1)$. Funkcja $\mathrm{MAJORITY}_n$ jest tylko szczególnym przypadkiem $\mathrm{THRESHOLD}_n^a$ dla $a = \lfloor \frac{n}{2} \rfloor$.

Z kolei i-ty bit wyjściowy funkcji $\mathrm{BITSORT}_n$ nie może być liczony szybciej niż funkcja $\mathrm{THRESHOLD}_n^i$. Stąd otrzymamy $\mathbf{idepen}(\mathrm{BITSORT}_n) \geqslant \lceil \frac{n}{2} \rceil$.

W przypadku funkcji MIN_n mamy trochę inną sytuację. W tym wypadku musimy zauważyć, że o tym, która liczba jest większa możemy stwierdzić tylko w jednym miejscu. To miejsce to najbardziej znacząca pozycja na której liczby będące argumentami funkcji się różnią. Te dwa bity decydujące o wyniku funkcji MIN_n muszą być znane na każdym wyjściu. Stąd $\mathbf{odepen}(\mathrm{MIN}_n) = n$. □

Lemat 2 *Niech $f \in B_n^m$ i $C \in \mathrm{Cir}(f)$. Wtedy*

$$\forall_{X \in \{0,1\}^n} \mathbf{time}_C(X) \geqslant \log \max \{\mathbf{idepen}(f), \mathbf{odepen}(f)\}.$$

Dowód: Niech $X \in \{0,1\}^n$ będzie ustalonym wektorem wejściowym. Z definicji $\mathbf{idepen}(f)$ mamy, że żaden mniejszy niż $\mathbf{idepen}(f)$ podzbiór bitów wejściowych nie wyznacza nam jednoznacznie wartości $f(X)$. Stąd, ponieważ fanin jest ograniczony przez dwa, każda sieć C obliczająca f nie przeczyta w czasie mniejszym niż $\log \mathbf{idepen}(f)$ wystarczającej liczby bitów aby jednoznacznie wyznaczyć wynik. Równocześnie, ponieważ fanout jest również ograniczony przez dwa w każdej sieci liczącej f, w czasie krótszym niż $\log \mathbf{odepen}(f)$ wartość pewnego bitu wejściowego nie wpłynie na wartości odpowiednich bitów wyjściowych, co uniemożliwiłoby nam wyznaczenie jednoznacznie wartości $f(X)$. □

Na podstawie lematu i wniosku otrzymujemy

Twierdzenie 7 *[JRS94]*

$$\mathbf{etime}(\mathrm{PARITY}_n, \mu_n^{\mathrm{UNI}}) \geqslant \log n,$$

$$\mathbf{etime}(\mathrm{MAJORITY}_n, \mu_n^{\mathrm{UNI}}) \geqslant \log n - 1.$$

Twierdzenie 8

$$\mathbf{etime}(\mathrm{BITSORT}_n, \mu_n^{\mathrm{UNI}}) \geqslant \log n - 1,$$

$$\mathbf{etime}(\mathrm{MIN}_n, \mu_n^{\mathrm{UNI}}) \geqslant \log n.$$

5 RÓWNOLEGŁY PROBLEM SUM PREFIKSOWYCH

5.1 Własności ogólne

Problem liczenia sum prefiksowych dla skończonej półgrupy pojawia się w bardzo wielu zagadnieniach informatycznych. Dlatego w rozdziale tym zajmiemy się analizą średniej złożoności problemu sum prefiksowych dla sieci asynchronicznych.

Definicja 7 *[JRSW93] Niech G z binarnym operatorem $\otimes$ będzie skończoną półgrupą, to jest niech $\otimes : G \times G \to G$ będzie łączne. Zastosowanie $\otimes$ do pary argumentów chcemy zapisać jako $\otimes(g_1, g_2)$ lub prościej jako $g_1 \otimes g_2$. Niech Σ będzie podzbiorem G, który nie koniecznie musi być półgrupą. Równoległa funkcja prefiksowa*

$$\mathrm{PP}_{\Sigma,n} : \Sigma^n \to G^n$$

odwzorowuje wektor wejściowy $X = x_1, \ldots, x_n \in \Sigma^n$ w wektor $Y = y_1, \ldots, y_n$, gdzie $y_i = x_1 \otimes \ldots \otimes x_i$.

Oczywiście, w zależności od operatora $\otimes$, znajomość obu argumentów nie zawsze jest konieczne do określenia wartości $\otimes(g_1, g_2)$.

Definicja 8 *[JRSW93] Zbiór lewych, odpowiednio prawych, argumentów, które jednoznacznie określają wynik działania $\otimes$ jest dany przez*

$$L_\otimes := \{a \in G \mid \otimes(a, G) \equiv const\}$$

$$R_\otimes := \{a \in G \mid \otimes(G, a) \equiv const\}$$

Mówimy, że G ma przyspieszenie z lewej strony wtedy i tylko wtedy gdy $L_\otimes \neq \emptyset$, odpowiednio z prawej wtedy i tylko wtedy gdy $R_\otimes \neq \emptyset$.

Jak łatwo zauważyć, logiczna półgrupa z operatorem OR ma przyspieszenie zarówno z lewej jak i prawej strony, podczas gdy z operatorem PARITY nie posiada jakiegokolwiek przyspieszenia.

Użycie odpowiedniego binarnego kodowania elementów półgrupy G może, dla dowolnej sieci nad G, umożliwić nam stworzenie sieci logicznej, w której bramkę $\otimes$ zastąpimy przez stałą logiczną podsieć o stałym rozmiarze. W związku z tym nasze rozważania możemy ograniczyć do sieci nad G zawierającej tylko bramki liczące

$\otimes$. Ograniczamy w takiej sieci również fanin i fanout do dwóch. W przypadku gdy potrzebujemy więcej wyników z jednej bramki stosujemy proste bramki podwajające.

Możemy teraz zdefiniować pojęcie czasu (**time**) dla sieci liczących problem prefiksowy

Definicja 9 *Niech $X \in G^n$ będzie wektorem wejściowym dla sieci C określonej dla G i niech v będzie wewnętrzną bramką C z bezpośrednimi poprzednikami v_1 i v_2. Niech* $\mathrm{res}(X)$ *oznacza wartość obliczoną przez bramkę v dla wejścia X. Wtedy oszacowanie czasu, w którym bramka v dla wejścia X poda nam swój rezultat jest dane przez*

$$\mathbf{time}_v(X) := 1 + \begin{cases} \max_i \ \mathbf{time}_{v_i}(X) & gdy\ \mathrm{res}_{v_1}(X) \notin L_\otimes \wedge \mathrm{res}_{v_2}(X) \notin R_\otimes \\ \mathbf{time}_{v_1}(X) & gdy\ \mathrm{res}_{v_1}(X) \in L_\otimes \wedge \mathrm{res}_{v_2}(X) \notin R_\otimes \\ \mathbf{time}_{v_2}(X) & gdy\ \mathrm{res}_{v_1}(X) \notin L_\otimes \wedge \mathrm{res}_{v_2}(X) \in R_\otimes \\ \min_i \ \mathbf{time}_{v_i}(X) & gdy\ \mathrm{res}_{v_1}(X) \in L_\otimes \wedge \mathrm{res}_{v_2}(X) \in R_\otimes \end{cases}$$

Dla bramek wejściowych v ustalamy $\mathbf{time}_v(X) := 0$. Czas obliczeń sieci C dla wejścia X, $\mathbf{time}_C(X)$, jest wtedy definiowany jako maksymalny $\mathbf{time}_v(X)$ dla wszystkich bramek wyjściowych v w C.

Łatwo zauważyć podobieństwo tej definicji do zamieszczonej wcześniej definicji czasu dla sieci logicznych.

Wszystkie definicje DG-sieci działających na elementach półgrup oraz definicje klas ich rozkładów otrzymujemy poprzez naturalną modyfikację definicji dla przypadku sieci logicznych.

Nowa wersja lematu 1 dla przypadku sieci działających na elementach półgrup jest następująca:

Lemat 3 *[JRSW93] Niech $X \in \mathcal{D}Depth_{n,\Sigma}(d)$. Wtedy dla wszystkich $i \in [1 \ldots n]$ i wszystkich $\sigma \in \Sigma$ oraz wszystkich $\omega \in \Sigma^{n-1}$ mamy*

$$|\Sigma|^{-2^d} \leqslant \Pr[X_i = \sigma] \leqslant 1 - |\Sigma|^{-2^d}$$

$$|\Sigma|^{-2^{3d}} \leqslant \Pr[X_i = \sigma \mid X_1 \ldots X_{i-1} X_{i+1} \ldots X_n = \omega] \leqslant 1 - |\Sigma|^{-2^{3d}}$$

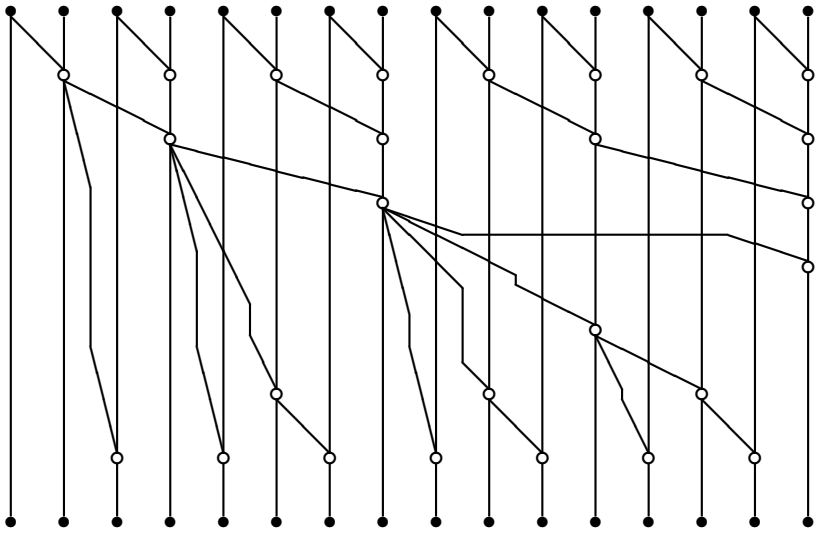
Dowód tego lematu jest podobny do dowodu lematu 1.

5.2 Dodawanie jako przykład przyspieszenia w średnim przypadku

Problem dodawania dwóch liczb binarnych można łatwo zredukować do obliczania przeniesień (znając przeniesienia możemy dodać dwie liczby w czasie stałym). Obliczanie przeniesień jest szczególnym przypadkiem obliczania sum prefiksowych w półgrupie $G_{\text{CARRY}} = (\{del, gen, pro\}, \otimes_{\text{CARRY}})$, gdzie poszczególne elementy, czyli $(delete, generate, propagate)$ oznaczają odpowiednio brak przeniesienia, generowanie przeniesienia, uzależnienie istnienia przeniesienia od jego wystąpienia na poprzedniej pozycji. Elementy półgrupy G_{CARRY} składamy w następujący sposób

$$u \otimes_{\text{CARRY}} v = \begin{cases} u & gdy\ v = pro \\ v & w\ p.p. \end{cases}$$

Oczywiście łatwo zauważyć, że ponieważ dla najmniej znaczących bitów możemy zawsze określić czy bit przeniesienia będzie generowany (*generate*) czy nie będzie (*delete*), wektor wyjściowy będzie składał się tylko z elementów *gen* i *del* mówiących nam czy z tej pozycji jest generowany bit przeniesienia.



Rysunek 6: Sieć licząca efektywnie sumy prefiksowe dla złożoności najgorszego przypadku (dla uproszczenia rysunku dopuszczono bramki o niestałym fanout).

Znana jest sieć licząca sumy prefiksowe, która posiada logarytmiczną głębokość i liniową wielkość. Posiada ona tą własność, że bramka wyjściowa y_i jest oddalona o co najwyżej $O(\log i)$ od każdej bramki wejściowej, która może wpłynąć na jej rezultat. To daje nam natychmiast, że dla dowolnego wejścia X czas działania sieci $\mathbf{time}(X)$

jest logarytmicznego rzędu. Poniżej pokażemy, że średni czas w tym przypadku może być zredukowany do $\log\log n$.

Dla obliczania przeniesień wejściowy alfabet równy jest zbiorowi elementów półgrupy G_{CARRY}. Dla $X \in \{del, gen, pro\}^n$ niech $pro(X)$ oznacza najdłuższy podciąg zawierający tylko elementy pro, natomiast $|pro(X)|$ długość tego podciągu. Ważność tej wielkości wynika z następującego lematu

Lemat 4 *Jeśli C jest siecią nad G_{CARRY} która liczy sumy prefiksowe, to dla każdego $X \in \{del, gen, pro\}^n$ mamy*

$$\log |pro(X)| \leqslant \mathbf{time}_C(X).$$

Ponadto, istnieje sieć dla której

$$\mathbf{time}_C(X) \leqslant 3 \cdot \log |pro(X)|.$$

Dowód: Dolna granica musi być tej wielkości, gdyż sieć musi rozpoznać cały ciąg elementów pro, aby móc podać wynik. Wobec ograniczenia stopnia wejściowego bramek, taki fragment podsieci musi mieć głębokość co najmniej logarytmiczną.

Górna granica $3\log |pro(X)|$ jest osiągnięta przez sieć o konstrukcji pokazanej na rysunku 7. □

Powyższe rezultaty pokazują, że analiza średniego przypadku dla dodawania redukuje się do obliczenia rozkładu wartości $pro(X)$. Możemy pokazać następujące ograniczenie

Lemat 5 *Niech X będzie zmienną losową nad G_{CARRY} z rozkładem należącym do $\mathcal{D}Depth_n(d)$. Wtedy dla dowolnego $l \leqslant \frac{n}{2}$ mamy*

$$\Pr[|pro(X)| \geqslant 2l-1] \leqslant 1 - (1 - \exp(-l \cdot 3^{-2^{3d}}))^{n/l}$$

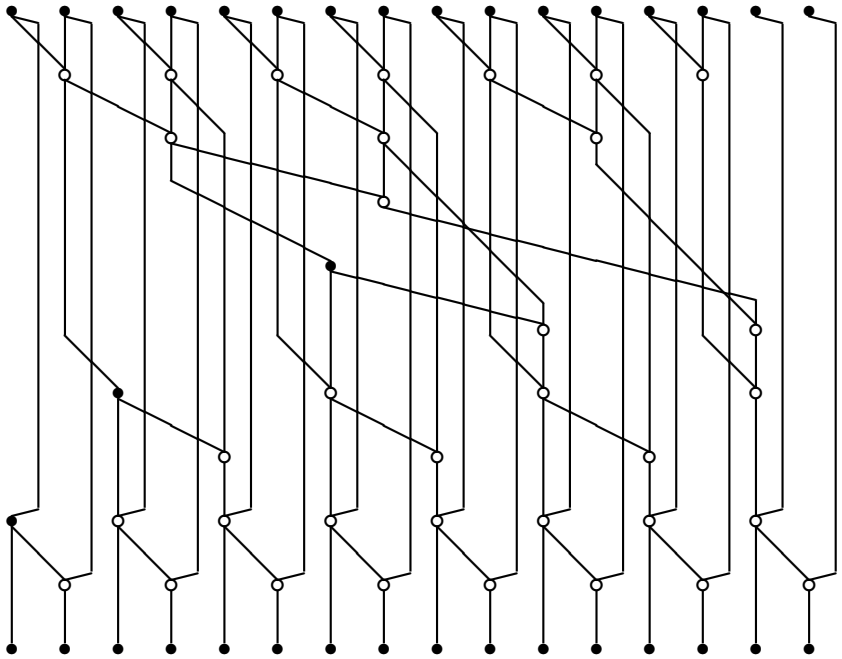
gdzie $\exp(x) = e^x$.

Dowód: Podzielmy wejście X na bloki długości l. Aby istniał łańcuch przeniesień długości co najmniej $2l-1$ przynajmniej jeden blok musi zawierać tylko symbole pro. Zgodnie z lematem 3 prawdopodobieństwo takiego zdarzenia wynosi co najwyżej $(1 - 3^{-2^{3d}})^l$. Stąd mamy

$$\Pr[|pro(X)| \geqslant 2l-1] \leqslant 1 - (1 - (1 - 3^{-2^{3d}})^l)^{n/l} \leqslant 1 - (1 - \exp(-l \cdot 3^{-2^{3d}}))^{n/l}$$

□

Możemy również pokazać dolne ograniczenie na istnienie długiego łańcucha przeniesień



Rysunek 7: Sieć licząca efektywnie sumy prefiksowe dla złożoności średniego przypadku.

Lemat 6 *Istnieje zmienna losowa X z rozkładem należącym do $\mathcal{D}Depth_n(d)$ taka, że*

$$\Pr[|pro(X)| \geqslant 3^{2^d} \cdot \log n] \;\geqslant\; \frac{1}{2}.$$

Dowód: Definiujemy X tak, że $\Pr[X_i = pro] = 1 - 3^{-2^d}$ i wszystkie X_i są niezależne. Wtedy otrzymamy

$$\begin{aligned}\Pr[|pro(X)| \geqslant 3^{2^d} \cdot \log n] \;&\geqslant\; \Pr[\forall_{i\in[1\ldots 3^{2^d}\cdot\log n]} X_i = pro] \\ &\geqslant\; 1 - (1-3^{2^d})^{3^{2^d}\cdot\log n} \;\geqslant\; 1 - 2^{-\log n} \;\geqslant\; \tfrac{1}{2}\end{aligned}$$

□

Te granice pozwalają nam teraz określić czas średniego przypadku dla dodawania

Twierdzenie 9 *[JRSW93] Dla $d \leqslant \frac{1}{3}\log\log n$ mamy*

$$\frac{1}{2}(\log\log n + 2^d\cdot\log 3) \;\leqslant\; \mathbf{etime}(\mathrm{PP}_{G_{\mathrm{CARRY}},n}, \mathcal{D}Depth_n(d)) \;\leqslant\; 6\cdot(\log\log n + 2^{3d}\cdot\log 3).$$

Dowód: Najpierw pokażemy ograniczenie dolne. Z lematu 4 i lematu 6 otrzymujemy, że istnieje rozkład $X \in \mathcal{D}Depth_n(d)$ taki, że dla dowolnej sieci C

$$\begin{aligned}\Pr[\mathbf{time}_C(X) \geqslant \log\log n + \log 3 \cdot 2^d] &\geqslant \Pr[\log|pro(X)| \geqslant \log\log n + \log 3 \cdot 2^d] \\ &\geqslant \Pr[|pro(X)| \geqslant \log n \cdot 3^{2^d}] \geqslant \tfrac{1}{2}\end{aligned}$$

Dlatego

$$\sum_t t \cdot \Pr[\mathbf{time}_C(X) = t] \geqslant \frac{1}{2} \cdot (\log\log n + \log 3 \cdot 2^d)$$

Teraz pokażemy ograniczenie górne. Sieć C z rysunku 7 oblicza sumy prefiksowe przeniesień szybciej niż optymalna sieć dla najgorszego przypadku. Obliczenie prefiksu $Y_i = X_1 \otimes_{\text{CARRY}} \ldots \otimes_{\text{CARRY}} X_i$ jest szacowane dla podciągów rosnącej długości.

Rozmiar sieci jest ograniczony przez $4n$. Jej czas obliczeń dla wejścia X jest nie większy niż $3\log|pro(X)|$. Z lematu 5 mamy

$$\Pr[|pro(X)| \geqslant 2l - 1] \leqslant 1 - (1 - \exp(-l \cdot 3^{2^{3d}}))^{n/l}$$

i

$$\Pr[\mathbf{time}_C(X) \geqslant 3t] \leqslant 1 - (1 - \exp(-2^t 3^{2^{3d}}))^{\frac{n}{2^t}} \leqslant \frac{n}{2^t}\exp(-2^t 3^{2^{3d}})$$

Biorąc teraz $t' := \log\log n + \log 3 \cdot 2^{3d}$ możemy ograniczyć czas oczekiwany przez

$$\begin{aligned}E(\mathbf{time}_C(X)) &\leqslant \Pr[\mathbf{time}_C(X) \leqslant 3t'] \cdot 3t' + \Pr[\mathbf{time}_C(X) > 3t'] \cdot 3\log n \\ &\leqslant 3 \cdot t'(1 - \exp(-2^{t'} \cdot 3^{2^{3d}}))^{\frac{n}{2^{t'}}} + 3\log n \cdot \tfrac{n}{2^{t'}} \cdot \exp(-2^{t'} \cdot 3^{2^{3d}}) \\ &\leqslant 6 \cdot (\log\log n + \log 3 \cdot 2^{3d})\end{aligned}$$

□

6 ASYNCHRONICZNE SIECI KOMPARATORÓW

Jednym z ciekawszych zastosowań sieci jest sortowanie liczb. Najczęściej stosowanym modelem są sieci komparatorów i nimi zajmiemy się w tym rozdziale.

6.1 Asynchroniczny komparator

Niech $\mathrm{COMPARE}_n : \{0,1\}^{2n} \to \{0,1\}$ oznacza funkcję, która porównuje dwie liczby n-bitowe. Jeśli jej wynikiem jest 0, to pierwsza liczba jest mniejsza lub równa drugiej, jeśli 1 to druga liczba jest mniejsza niż pierwsza. Ponieważ w rozdziale 4 pokazaliśmy, że przyspieszenie w wypadku funkcji zwracającej na wyjściu mniejszą z dwóch liczb nie jest możliwe (funkcja MIN_n), dlatego konstruując komparator odchodzimy od zasady, że fanout jest stały. Model komparatora, którym posłużymy się w tym rozdziale, liczy dla dwóch liczb n-bitowych funkcję $\mathrm{COMPARE}_n$ a następnie bezpośrednio przekazuje jej wynik do $2n$ bramek wyjściowych. Wyjściowa bramka i-ta zwraca wtedy i-ty bit mniejszej liczby a wyjściowa bramka $(n+i)$-ta zwraca i-ty bit większej liczby (każda bramka wyjściowa ma trzy wejścia, po jednym bicie z dwóch liczb i bit decyzyjny).

Dla realizacji funkcji $\mathrm{COMPARE}_n$ posłużymy się następującą metodą. Jeśli mamy porównać dwie liczby $x = x_1 \ldots x_n$ i $y = y_1 \ldots y_n$, to najpierw dla każdego x_i i y_i liczymy z_i według następującej reguły

$$z_i = \begin{cases} w & gdy \quad x_i > y_i \\ r & gdy \quad x_i = y_i \\ m & gdy \quad x_i < y_i \end{cases}$$

gdzie w oznacza "większy", r — "równy", a m odpowiednio "mniejszy". Następnie liczymy $z = z_1 \otimes \ldots \otimes z_m$, gdzie $\otimes$ jest zdefiniowane w sposób następujący

$$u \otimes v = \begin{cases} u & gdy \quad u \neq r \\ v & gdy \quad u = r \end{cases}$$

Wyjściowa bramka licząca z automatycznie zamienia r i m na 0 a w na 1, aby funkcja $\mathrm{COMPARE}_n$ posiadała logiczne wartości wyjściowe.

Do obliczenia z możemy posłużyć się asynchronicznymi sieciami z rysunku 2 lub 3 (będziemy odtąd nazywać komparator zbudowany w swej głównej części tak jak na rysunku 2 asynchronicznym komparatorem typu $\mathcal{A}$ a tak jak na rysunku 3 asynchronicznym komparatorem typu $\mathcal{B}$).

Twierdzenie 10

$$\mathbf{etime}(\mathrm{COMPARE}_n, \mu_{2n}^{\mathrm{UNI}}) \leqslant 3.$$

Dowód: Zgodnie z definicją, dla sieci typu $\mathcal{A}$ mamy

$$\begin{aligned}
&\mathbf{etime}(\mathrm{COMPARE}_n, \mu_{2n}^{\mathrm{UNI}}) \\
&= 1 + \sum_{t=1}^{n-1} t \cdot \Pr[\text{obie liczby mają jednakowy prefiks długości } t-1] \\
&\quad + (n-1) \cdot \Pr[\text{tylko ostatnie bity obu liczb są różne}] \\
&\quad + (n-1) \cdot \Pr[\text{obie liczby są równe}] \ \leqslant \\
&\leqslant 1 + \sum_{t=1}^{\infty} t \cdot \tfrac{1}{2^t} \ \leqslant \ 3
\end{aligned}$$

□

Z powyższego twierdzenia wynika, że średni czas dla komparatora typu $\mathcal{A}$, jeśli uwzględnimy warstwę generującą wynik, jest mniejszy niż 4 dla dwóch liczb o rozkładzie jednorodnym. Równocześnie możemy zauważyć, że średni czas w przypadku zastosowania komparatora typu $\mathcal{B}$ jest mniejszy niż 4.3.

Oprócz tego, jeśli zauważymy podobieństwo obliczania przeniesień sumy dwóch liczb oraz sprawdzenia, która liczba jest większa opisaną powyżej metodą, to łatwo uzyskamy

Twierdzenie 11 *Dla* $d \ \leqslant \ \frac{1}{3}\log\log n$ *mamy*

$$\frac{1}{2}(\log\log n + 2^d \cdot \log 3) \ \leqslant \ \mathbf{etime}(\mathrm{COMPARE}_n, \mathcal{D}Depth(d)) \ \leqslant \ 6 \cdot (\log\log n + 2^{3d} \cdot \log 3).$$

Dowód: Dowód tego twierdzenia jest modyfikacją dowodu twierdzenia 9. □

6.2 Średnia złożoność sieci asynchronicznych komparatorów — granica górna

Dużo ciekawszym problemem jest jednak oczekiwany czas obliczeń dla sieci zbudowanych z asynchronicznych komparatorów.

W przypadku sieci zbudowanych z komparatorów synchronicznych czas działania jest prosty do obliczenia. Dla sieci pracującej na n liczbach m-bitowych, która zawiera $\mathcal{N}(n)$ warstw komparatorów, czas działania (licząc operacje bitowe) jest niezależny od postaci danych wejściowych a więc zależny tylko od ich wielkości i wynosi

$$\mathcal{N}(n) \cdot (\log m + 2).$$

Wielkość $\log m + 2$ uzyskujemy stosując do obliczania funkcji $\mathrm{COMPARE}_n$ sieć w postaci pełnego drzewa binarnego. Jak pokażemy niżej, w przypadku zastosowania asynchronicznych komparatorów możemy uzyskać znaczne przyspieszenie.

Na początek pokażmy następujący lemat

Lemat 7 *Weźmy $n = 2^k$ liczb m-bitowych o jednorodnym rozkładzie prawdopodobieństwa. Wówczas prawdopodobieństwo, że dla dowolnej wybranej z pośród nich pary, długość ich wspólnego prefiksu jest mniejsza niż $i + k$ (dla $i = 0 \ldots m - k$) wynosi*

$$\Gamma(n, i) := \frac{(2^i \cdot n)!}{(2^i \cdot n)^n \cdot ((2^i - 1) \cdot n)!}.$$

Dowód: Jak łatwo zauważyć prawdopodobieństwo, że najdłuższy wspólny prefiks dla dwóch z n wybranych liczb m-bitowych o rozkładzie jednorodnym jest mniejszy niż $i + k$ jest równe prawdopodobieństwu, że każda z wybranych liczb wpadnie do innego z 2^{i+k} równych przedziałów, na które podzielimy całe uniwersum liczb. Prawdopodobieństwo to jest równe

$$1 \cdot \frac{n2^i - 1}{n2^i} \cdot \ldots \cdot \frac{n2^i - (n-1)}{n2^i} = \Gamma(n, i)$$

□

Teraz możemy pokazać następujące twierdzenia opisujące nam średnią złożoność sieci zbudowanych z asynchronicznych komparatorów.

Twierdzenie 12 *Dla dowolnej sieci komparatorów C z n wejściami, z których każde jest ciągiem m-bitowym i posiadającej $\mathcal{N}(n)$ poziomów asynchronicznych komparatorów typu $\mathcal{A}$ oczekiwany czas obliczeń (licząc operacje bitowe) wynosi co najwyżej*

$$\mathcal{N}(n) \cdot \min(2 \log n + 4.25 \ , \ m + 1).$$

jeżeli dane wejściowe pochodzą z rozkładu jednorodnego.

Dowód: Niech $n = 2^k$. Dla każdego poziomu komparatorów w sieci C czas przejścia przez tą warstwę wynosi co najwyżej tyle, ile wynosi długość najdłuższego wspólnego prefiksu dla dowolnej pary z pośród n wejściowych liczb plus trzy (co widać na podstawie budowy komparatora typu $\mathcal{A}$). Stąd możemy już wywnioskować, że średni czas dla pojedyńczej warstwy asynchronicznych komparatorów typu $\mathcal{A}$ wynosi co najwyżej

$$\begin{aligned} 2 \ + \ & \sum_{j=k}^{m-1} \Pr[\text{najdłuższy wspólny prefiks dla 2 z n liczb równa się } j-1] \cdot j \\ & + \Pr[\text{najdłuższy wspólny prefiks dla 2 z n liczb równa się } m-1] \cdot (m-1) \\ & + \Pr[\text{przynajmniej 2 z n liczb są równe}] \cdot (m-1) \end{aligned}$$

Teraz korzystając z lematu 7 i przyjmując $\Gamma(n, -1) = 0$ otrzymamy, że powyższa suma wynosi co najwyżej

$$\begin{aligned} 2 \ + \ & \sum_{j=k}^{\infty} (\Gamma(n, j-k) - \Gamma(n, j-k-1)) \cdot j \\ & \leqslant \ 2 + k + \sum_{j=0}^{\infty} (\Gamma(n, j) - \Gamma(n, j-1)) \cdot j \\ & \leqslant \ 2 + k + \sum_{j=0}^{\infty} (1 - \Gamma(n, j)) \qquad (*) \end{aligned}$$

Zauważmy teraz, że

$$1-\Gamma(n,j) \;=\; 1-\tfrac{(n2^j)\cdot(n2^j-1)\cdot\ldots\cdot(n2^j-(n-1))}{(n2^j)^n} \;\leqslant\; 1-\tfrac{(n2^j-n)^n}{(n2^j)^n} \;=$$
$$=\; 1-(1-\tfrac{1}{2^j})^n \;=\; 1-((1-\tfrac{1}{2^j})^{2^j})^{\frac{n}{2^j}} \;\leqslant\; 1-(\tfrac{1}{4})^{\frac{n}{2^j}}$$

dla $j>0$. Stąd

$$\begin{aligned}(*) \;&\leqslant\; 2+k+1+\sum_{j=1}^{\infty}(1-(\tfrac{1}{4})^{\frac{n}{2^j}}) \;= \\ &=\; 3+k+\sum_{j=1}^{k-1}(1-(\tfrac{1}{4})^{\frac{n}{2^j}})+\sum_{j=k}^{\infty}(1-(\tfrac{1}{4})^{\frac{n}{2^j}}) \;= \\ &=\; 3+k+\sum_{j=1}^{k-1}(1-(\tfrac{1}{4})^{2^j})+\sum_{j=0}^{\infty}(1-(\tfrac{1}{4})^{2^{-j}}) \\ &\leqslant\; 3+k+k-1+\sum_{j=0}^{\infty}(1-(\tfrac{1}{4})^{2^{-j}}) \qquad (**)\end{aligned}$$

Musimy teraz oszacować ostatni składnik sumy. Otrzymujemy

$$(1-(\frac{1}{4})^{2^{-i}}) \;=\; (1+(\frac{1}{4})^{2^{-i-1}})(1-(\frac{1}{4})^{2^{-i-1}}) \;\geqslant\; \frac{3}{2}(1-(\frac{1}{4})^{2^{-(i+1)}})$$

dla $i\geqslant 0$, a stąd mamy

$$\sum_{i=0}^{\infty}(1-(\frac{1}{4})^{2^{-i}}) \;\leqslant\; \sum_{i=0}^{\infty}(1-\frac{1}{4})\cdot(\frac{2}{3})^i \;=\; \frac{3}{4}\cdot\frac{1}{1-\frac{2}{3}} \;=\; \frac{9}{4}$$

Teraz otrzymujemy

$$(**) \;\leqslant\; 2+2k+\frac{9}{4} \;=\; 2\log n+4.25$$

Ponieważ m było dowolne, rezultat ten jest prawdziwy dla liczb każdej długości. □

Twierdzenie 13 *Dla dowolnej sieci komparatorów C z n wejściami, z których każde jest ciągiem m-bitowym i posiadającej $\mathcal{N}(n)$ poziomów asynchronicznych komparatorów typu $\mathcal{B}$ oczekiwany czas obliczeń (licząc operacje bitowe) wynosi co najwyżej*

$$\mathcal{N}(n)\cdot\min(2\log\log n+5\ ,\ 2\log m+3).$$

jeżeli dane wejściowe pochodzą z rozkładu jednorodnego. Ponadto taka sieć działa w najgorszym przypadku tylko dwa razy wolniej niż sieć z synchronicznymi komparatorami o optymalnej złożoności najgorszego przypadku.

Dowód: Niech dla ułatwienia $n = 2^{2^k}$. Średni czas dla sieci C możemy wówczas ograniczyć z góry przez wyrażenie

$$\sum_{j=k}^{m} \Pr[\text{najdłuższy wspólny prefiks dla 2 z n liczb równa się } j-1] \cdot (2\lfloor \log j \rfloor + 1) + \Pr[\text{przynajmniej 2 z n liczb są równe}] \cdot (2\lfloor \log m \rfloor + 1) \; + \; 2$$

Korzystając z lematu 7 i przyjmując $\Gamma(n,-1) = 0$ otrzymamy, że powyższa suma jest nie większa niż

$$\begin{aligned} 2 \; + \; & \sum_{j=2^k}^{\infty} (\Gamma(n, j-2^k) - \Gamma(n, j-2^k-1)) \cdot (2\lfloor \log j \rfloor + 1) \; = \\ & = \; 2 + 1 + 2 \sum_{j=0}^{\infty} (\Gamma(n,j) - \Gamma(n,j-1)) \cdot \lfloor \log(j+2^k) \rfloor \; = \\ & = \; 3 + 2 \sum_{j=1}^{\infty} (\Gamma(n, (2^j-1)\cdot 2^k - 1) - \Gamma(n, (2^{j-1}-1)\cdot 2^k - 1)) \cdot (k+j-1) \; = \\ & = \; 3 + 2(k-1) + 2 \sum_{j=1}^{\infty} (\widetilde{\Gamma}(n,j) - \widetilde{\Gamma}(n,j-1)) \cdot j \qquad (*) \end{aligned}$$

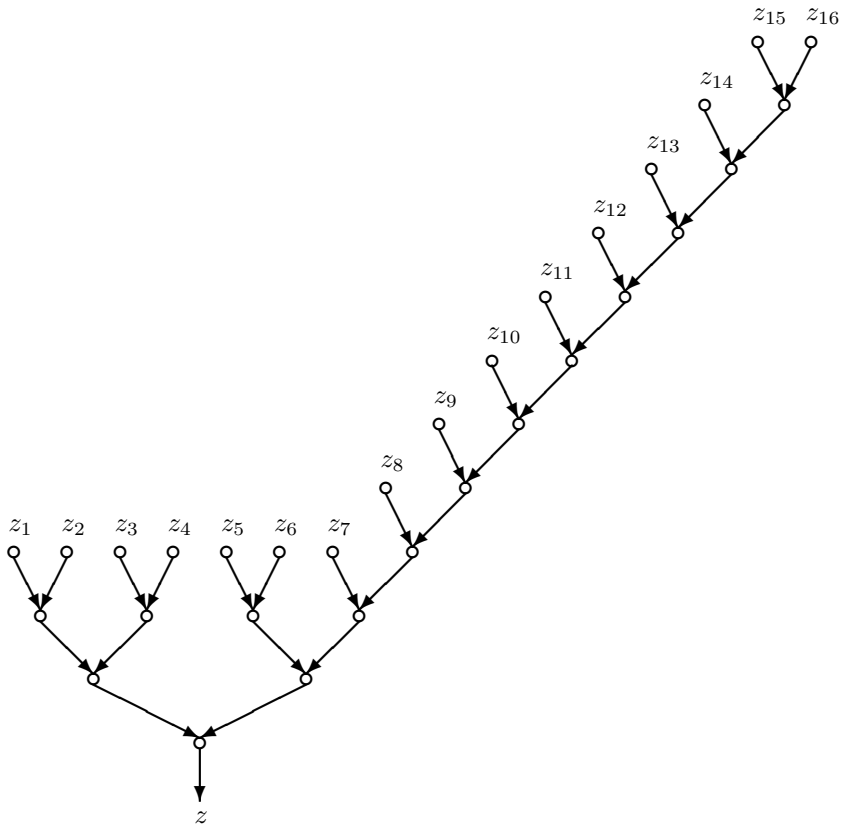
gdzie oczywiście $\widetilde{\Gamma}(n,j) = \Gamma(n, (2^j-1)\cdot 2^k - 1)$. Stąd

$$\begin{aligned} (*) \; & = \; 1 + 2k + 2 \sum_{j=0}^{\infty} (1 - \widetilde{\Gamma}(n,j)) \; = \; 2k + 3 + 2 \sum_{j=1}^{\infty} (1 - \widetilde{\Gamma}(n,j)) \\ & \leqslant \; 2k + 3 + 2 \; = \; 2\log\log n + 5 \end{aligned}$$

□

Możemy zastanowić się również, czy dla sieci komparatorów ustalonej wielkości nie istnieje asynchroniczny komparator, dla którego moglibyśmy jeszcze bardziej zmniejszyć ograniczenie górne na średni czas obliczeń. Możemy przy tym wykorzystać fakt, że dla sieci sortującej n liczb średnia długość porównywanych prefiksów wynosi około $2\log n$ (oczywiście jeśli liczby są odpowiednio długie). Skonstruujmy więc asynchroniczny komparator typu $\mathcal{C}^n$. Opis konstrukcji ograniczymy tutaj tylko do części liczącej sumę w półgrupie $(\{w,r,m\},\otimes)$. Niech $l = \min_i\{4\log n \; \leqslant \; 2^i\}$. Jeśli długość porównywanych liczb — m jest nie większa niż 2^l, to sieć ma postać pełnego drzewa binarnego, takiego jak na rysunku 1. Jeśli natomiast $m \; > \; 2^l$, to tworzymy pełne drzewo binarne o głębokości l, w którym w miejsce ostatniej bramki wejściowej podłączamy sieć taką, jak na rysunku 2. W ten sposób uzyskujemy potrzebną nam liczbę bramek wejściowych, czyli m, z czego $2^l - 1$ będzie na poziomie l. Przykładowa sieć licząca sumę w półgrupie $(\{w,r,m\},\otimes)$ i stanowiąca główną część komparatora typu $\mathcal{C}^n$ jest przedstawiona na rysunku 8.

Teraz możemy pokazać następujące twierdzenie



Rysunek 8: Komparator asynchroniczny $\mathcal{C}^4$ dla liczb 16-bitowych (na rysunku nie ma warstwy zamieniającej bity na elementy półgrupy $(\{w, r, m\}, \otimes)$).

Twierdzenie 14 *Dla dowolnej sieci komparatorów C z n wejściami, z których każde jest ciągiem m-bitowym i posiadającej $\mathcal{N}(n)$ poziomów asynchronicznych komparatorów typu $\mathcal{C}^n$ oczekiwany czas obliczeń (licząc operacje bitowe) wynosi co najwyżej*

$$\mathcal{N}(n) \cdot \min(\log\log n + 6.5\ ,\ \log m + 2).$$

jeżeli dane wejściowe pochodzą z rozkładu jednorodnego.

Dowód: Niech $n = 2^k$ i niech $l = \min_i\{4k \leqslant 2^i\}$ oraz $m > 2^l$. Wówczas średni

czas obliczeń w sieci C możemy ograniczyć z góry przez

$$\begin{aligned} 2 \;+\; & \Pr[\text{najdłuższy wspólny prefiks dla 2 z n liczb jest mniejszy niż } 2^l - 1] \cdot l \\ & + \sum_{j=2^l}^{m-1} (\Pr[\text{najdłuższy wspólny prefiks dla 2 z n liczb równa się } j-1] \cdot \\ & \qquad \cdot (j - (2^l - 1) + l)) \\ & + \Pr[\text{najdłuższy wspólny prefiks dla 2 z n liczb równa się } m-1] \cdot (m - 2^l + l) \\ & + \Pr[\text{przynajmniej 2 z n liczb są równe}] \cdot (m - 2^l + l) \end{aligned}$$

Korzystając z lematu 7 otrzymamy, że powyższa suma jest nie większa niż

$$\begin{aligned} & 2 \;+\; \Gamma(n, 2^l - 1 - k) \cdot l \\ & \qquad + \sum_{j=2^l}^{\infty} (\Gamma(n, j-k) - \Gamma(n, j-k-1)) \cdot (j - (2^l - 1) + l) \\ & \leqslant\; 2 + l + \sum_{i=0}^{\infty} (\Gamma(n, i + 2^l - k) - \Gamma(n, i + 2^l - k - 1)) \cdot (i+1) \\ & \leqslant\; 2 + l + \sum_{j=0}^{\infty} (1 - \Gamma(n, j + 2^l - k - 1)) \\ & \leqslant\; 2 + l + \sum_{j=0}^{\infty} (1 - \Gamma(n, j + 3k - 1)) \\ & \leqslant\; 2 + l + \sum_{j=0}^{\infty} (1 - (\tfrac{1}{4})^{2^{-(j+2k-1)}}) \\ & \leqslant\; 2 + l + \sum_{j=0}^{\infty} (1 - (\tfrac{1}{4})^{2^{-(2k-1)}}) \cdot (\tfrac{2}{3})^j \\ & \leqslant\; 2 + l + 1.5 \;\leqslant\; 3.5 + \lceil \log 4k \rceil \;\leqslant\; \log\log n + 6.5 \end{aligned}$$

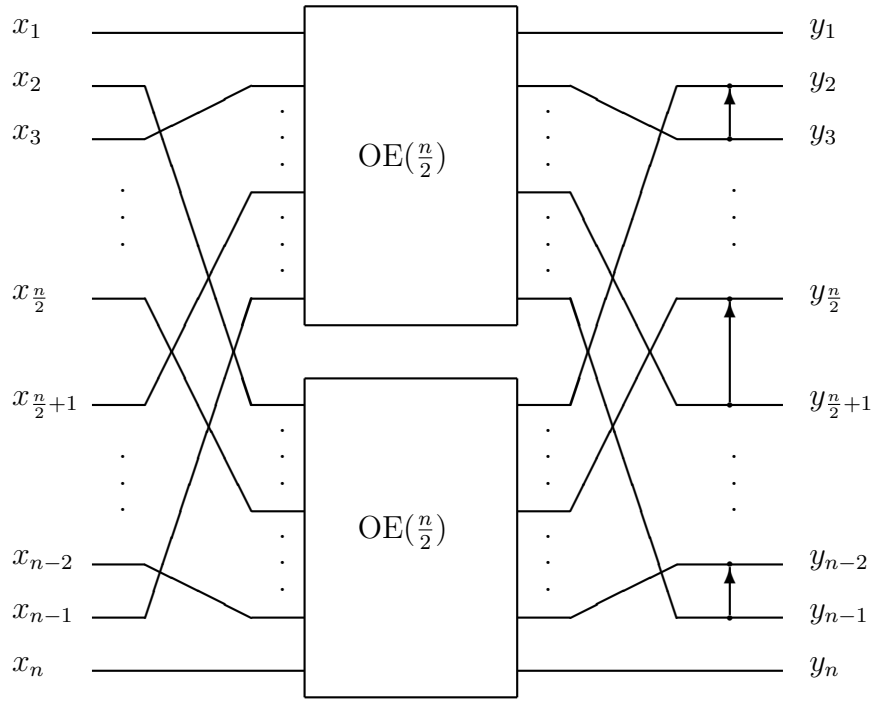
□

Możemy zauważyć jednocześnie, że w przypadku zastosowania komparatora typu $\mathcal{C}^n$ średni czas sieci komparatorów pracującej na n liczbach m-bitowych jest ograniczony z dołu przez wielkość

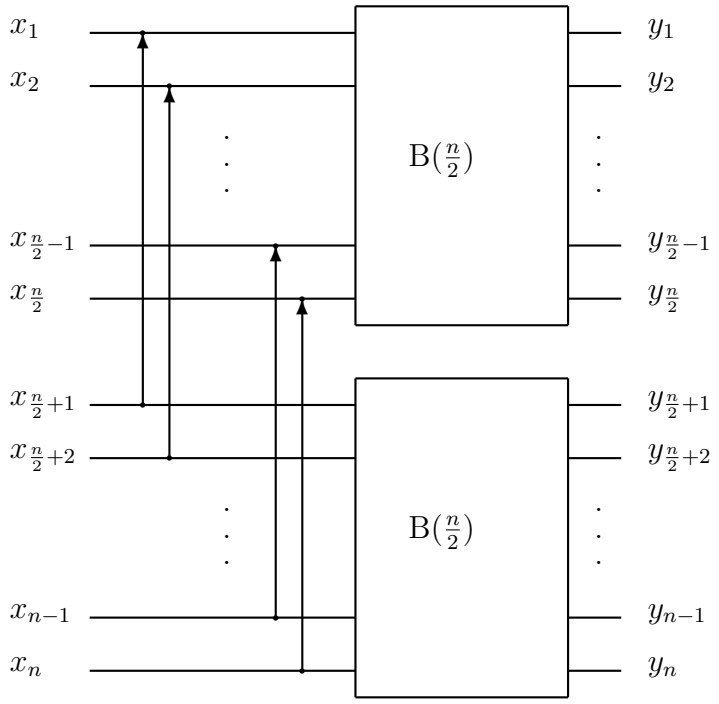
$$\mathcal{N}(n) \cdot \min(\log\log n + 4 \;,\; \log m + 2).$$

6.3 Doświadczalna weryfikacja wyników

Aby zbadać empirycznie przyspieszenie dla sieci zbudowanej z asynchronicznych komparatorów przeprowadziliśmy symulacje ich działania. Z uwagi na prostotę budowy oraz powszechne zastosowanie, próby przeprowadziliśmy dla sieci Odd-Even (rysunek 9) i sieci bitonicznej (rysunek 10). Symulacje przeprowadziliśmy zarówno dla sieci scalających, jak i dla zbudowanych z nich sieci sortujących. Do porównywania liczb w tych sieciach użyliśmy asynchronicznych komparatorów typu $\mathcal{A}$ oraz $\mathcal{B}$. Nie używaliśmy komparatorów typu $\mathcal{C}^n$, ponieważ dla nich ograniczenie górne różni się od



Rysunek 9: Schemat budowy sieci scalającej Odd–Even dla n liczb i o $\log n$ poziomach komparatorów. Sieć scala dwa posortowane niemalejąco ciągi $x_1, \ldots, x_{\frac{n}{2}}$ i $x_{\frac{n}{2}+1}, \ldots, x_n$ w posortowany niemalejąco ciąg $y_1, \ldots, y_n$. (Strzałka oznacza komparator i kierunek w którym jest przesuwana mniejsza liczba).



Rysunek 10: Schemat budowy bitonicznej sieci scalającej dla n liczb i o $\log n$ poziomach komparatorów. Sieć scala posortowany niemalejąco ciąg $x_1, \ldots, x_{\frac{n}{2}}$ i posortowany nierosnąco ciąg $x_{\frac{n}{2}+1}, \ldots, x_n$ w posortowany niemalejąco ciąg $y_1, \ldots, y_n$. (Strzałka oznacza komparator i kierunek w którym jest przesuwana mniejsza liczba).

dolnego tylko o około 2 w związku z czym przepływ liczb jest prawie równomierny i jego czas musi być niewiele mniejszy od górnego oszacowania.

Symulacje przeprowadziliśmy dla sieci scalających pracujących na ilości liczb od 2 do 512, będących potęgami dwójki. Dla sieci sortujących ograniczyliśmy się tylko do czterech wielkości sieci, mianowicie ilość sortowanych liczb wynosiła odpowiednio 64, 128, 256 i 512. Również wielkość liczb, dla których wykonywaliśmy doświadczenie została ograniczona do czterech przypadków : 64, 128, 256 i 512 bitów. Tablice od 1 do 4 przedstawiają wyniki tych symulacji zawierając średni czas działania uzyskany w 10000 prób dla każdego przypadku. Równocześnie dla porównania podajemy czas działania sieci z optymalnymi synchronicznymi komparatorami.

Jak łatwo zauważyć na podstawie tablic, zgodnie z wynikami teoretycznymi, średni czas asynchronicznej sieci komparatorów nie zależy od długości liczb wejściowych. Również uzyskany średni czas obliczeń jest mniejszy niż jego oszacowanie na podstawie twierdzeń 12 i 13. Na przykład dla sieci sortującej 256 liczb i posiadającej 36 poziomów komparatorów, na podstawie twierdzenia 12 można policzyć, że średni czas działania takiej sieci nie przekracza 441. W trakcie doświadczeń dla sieci Odd-Even czas ten wynosił około 271 a dla sieci bitonicznej 274. Również oszacowanie górnej granicy czasu średniego na podstawie twierdzenia 13 jest wystarczające. Wynosi ono 396, gdy uzyskany w trakcie prób średni czas dla sieci Odd-Even wyniósł około 244, a dla sieci bitonicznej 243. Jak więc widać w rzeczywistych sieciach dane przepływają dużo szybciej niż to wynika na podstawie oszacowań dokonanych w twierdzeniach.

Wielkość sieci	Długość liczb											
	64			128			256			512		
	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$
2	3.99	4.29	8	4.00	4.27	9	3.99	4.26	10	3.95	4.22	11
4	9.52	10.15	16	9.57	10.14	18	9.53	10.13	20	9.44	10.06	22
8	17.36	17.88	24	17.37	17.80	27	17.30	17.75	30	17.37	17.84	33
16	26.32	25.58	32	26.25	25.56	36	26.25	25.57	40	26.21	25.56	44
32	36.16	33.70	40	36.13	33.61	45	36.13	33.61	50	36.06	33.56	55
64	47.06	41.95	48	47.09	41.95	54	47.11	41.96	60	47.04	41.93	66
128	58.97	50.52	56	58.96	50.51	63	58.93	50.51	70	58.02	50.56	77
256	71.77	59.41	64	71.74	59.45	72	71.74	59.43	80	71.92	59.41	88
512	85.48	68.61	72	85.49	68.61	81	85.54	68.61	90	85.53	68.58	99

Tablica 1: Porównanie czasu działania sieci scalającej Odd-Even dla komparatorów asynchronicznych $\mathcal{A}$ i $\mathcal{B}$ oraz komparatora synchronicznego ($\mathcal{S}$).

Wielkość sieci	Długość liczb											
	64			128			256			512		
	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$
2	3.99	4.27	8	4.01	4.27	9	4.01	4.27	10	4.00	4.24	11
4	9.94	10.53	16	9.75	10.42	18	9.98	10.57	20	9.88	10.49	22
8	17.21	17.44	24	17.26	17.57	27	17.38	17.57	30	17.35	17.57	33
16	25.88	25.03	32	25.81	25.99	36	25.84	25.06	40	25.72	25.04	44
32	35.59	33.09	40	35.40	33.04	45	35.53	33.07	50	35.56	33.06	55
64	46.20	41.39	48	46.13	41.39	54	46.19	41.38	60	46.27	41.43	66
128	57.81	49.95	56	57.70	49.92	63	57.76	49.93	70	57.79	49.92	77
256	70.46	58.78	64	70.32	58.77	72	70.46	58.75	80	70.42	58.79	88
512	84.02	67.82	72	83.90	67.90	81	83.97	67.88	90	83.88	67.82	99

Tablica 2: Porównanie czasu działania bitonicznej sieci scalającej dla komparatorów asynchronicznych $\mathcal{A}$ i $\mathcal{B}$ oraz komparatora synchronicznego ($\mathcal{S}$).

Wielkość sieci	Długość liczb											
	64			128			256			512		
	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$
64	142.49	134.60	168	142.66	134.69	189	142.60	134.67	210	142.04	134.64	231
128	200.54	184.71	224	200.81	184.88	252	200.65	184.77	280	200.10	184.80	308
256	270.50	243.27	288	270.61	243.35	324	270.55	243.34	360	270.12	243.28	396
512	353.13	310.53	360	353.20	310.60	405	353.36	310.66	450	353.21	310.61	495

Tablica 3: Porównanie czasu działania sieci sortującej Odd-Even dla komparatorów asynchronicznych $\mathcal{A}$ i $\mathcal{B}$ oraz komparatora synchronicznego ($\mathcal{S}$).

Wielkość sieci	Długość liczb											
	64			128			256			512		
	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{S}$
64	144.71	134.92	168	144.23	134.61	189	144.78	134.88	210	144.57	134.67	231
128	203.60	184.82	224	203.10	184.44	252	203.40	184.71	280	202.86	184.49	308
256	273.96	243.02	288	273.67	242.74	324	273.89	242.99	360	273.59	242.83	396
512	356.98	309.94	360	357.10	309.77	405	356.84	309.91	450	357.20	310.05	495

Tablica 4: Porównanie czasu działania bitonicznej sieci sortującej dla komparatorów asynchronicznych $\mathcal{A}$ i $\mathcal{B}$ oraz komparatora synchronicznego ($\mathcal{S}$).

Literatura

[BHPS94] B. Bollig, M. Hühne, S. Pölt, P. Savický,
On the Average Case Circuit Delay of Disjunction,
Forschungsberichte des Fachbereichs Informatik der Universität Dortmund, 1994.

[JRS94] A. Jakoby, R. Reischuk, C. Schindelhauer,
Circuit Complexity: from the Worst Case to the Average Case,
STOC 94.

[JRSW93] A. Jakoby, R. Reischuk, C. Schindelhauer, S. Weis,
The Average Case Complexity of the Parallel Prefix Problem,
Technical Report, TH Darmstadt, 1993.

[L92] T. Leighton,
Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes,
Morgan Kaufmann Publishers, California, 1992.