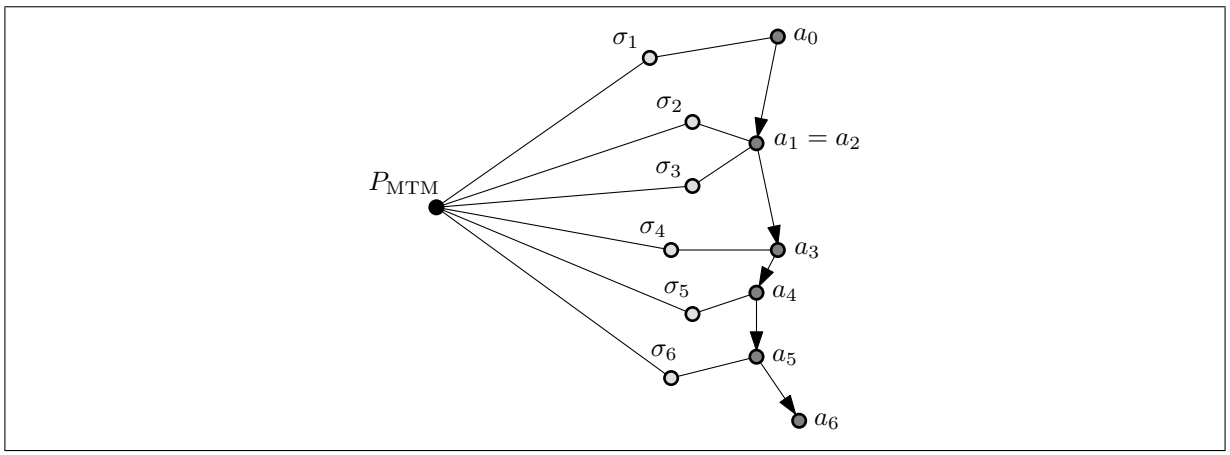

Rysunek 1: Ilustracja algorytmu Mtm w jednej fazie

1 Przenoszenie plików

1.0.1 Algorytm Move-To-Min

Rozważmy teraz następujący algorytm deterministyczny Move-To-Min (Mtm). Algorytm ten dzieli całą sekwencję na fazy długości D. W każdej fazie algorytm przebywa w jednym wierzchołku, który oznaczamy P_{MTM} a pod koniec fazy przenosi się do tzw. *centrum grawitacji*, v^*. Wierzchołek v^* jest wierzchołkiem w którym byłoby najlepiej przebywać w danej fazie, tj. jest wirzchołkiem minimalizującym sumę $\sum_{i=1}^{D} d(v^*, \sigma_i)$.

Twierdzenie 1. *Algorytm* Mtm *jest* 7*-konkurencyjny.*

Aby pokazać powyższe twierdzenie, zdefiniujemy funkcję potencjału równą $2 \cdot D \cdot d(P_{\text{MTM}}, P_{\text{OPT}})$ i pokażemy że zamortyzowany koszt w pojedynczej fazie f jest ograniczony.

Poniżej koncentrujemy się na pojedynczej fazie f. Numerujemy kroki w tej fazie od 1 do D. Wprowadzimy dodatkowe oznaczenie, które zilustrowane zostało na Rysunku 1. Mianowicie oznaczamy wierzchołek w którym Opt ma plik na początku kroku j przez a_{j-1}, a wierzchołek w którym Opt ma plik na końcu kroku j przez a_j. W szczególności a_0 i a_D są wierzchołkami, w których Opt ma plik odpowiednio na początku i końcu fazy.

Na początku pokażemy następujący pomocniczy lemat. Mówi on, że dolnym ograniczeniem na koszt algorytmu optymalnego jest koszt algorytmu, który całą fazę spędzi w dowolnym wierzchołku a_ℓ.

Lemat 1. *Dla dowolnej fazy f i dowolnego kroku $0 \leqslant \ell \leqslant D$, zachodzi* $\textsc{Opt}(f) \geqslant \sum_{i=1}^{D} d(a_\ell, \sigma_i)$.

Dowód. Zgodnie z oznaczeniami z rysunku koszt algorytmu optymalnego wynosi $\sum_{i=1}^{D}(d(a_{i-1}, \sigma_i) + D \cdot d(a_{i-1}, a_i))$. W powyższej sumie każda z odległości pomiędzy kolejnymi a_i jest liczona D razy. Z nierówności trójkąta otrzymujemy, że

$$\begin{aligned}
\textsc{Opt}(f) &= \sum_{i=1}^{D} (d(a_{i-1}, \sigma_i) + D \cdot d(a_{i-1}, a_i)) \\
&\geqslant \sum_{i=1}^{D} (d(a_{i-1}, \sigma_i) + d(a_{i-1}, a_\ell)) \\
&\geqslant \sum_{i=1}^{D} d(a_\ell, \sigma_i) \ .
\end{aligned}$$

□

Podzielmy koszt $\textsc{Mtm}(f)$ na dwie składowe: $\textsc{Mtm}^{\text{REQ}}(f)$ jest kosztem obsługi żadań w fazie f, a $\textsc{Mtm}^{\text{MOVE}}(f)$ jest kosztem przenosin pliku do wierzchołka v^*. Dodatkowo definiujemy $\Phi_{\text{B}}(f)$ i $\Phi_{\text{F}}(f)$ jako potencjał odpowiednio na początku i końcu fazy f. Należy zatem pokazać, że prawdziwe jest następujące stwierdzenie.

Lemat 2. *Dla dowolnej fazy f zachodzi*

$$\textsc{Mtm}^{\text{REQ}}(f) + \textsc{Mtm}^{\text{MOVE}}(f) + \Phi_{\text{F}}(f) \leqslant \Phi_{\text{B}}(f) + 7 \cdot \textsc{Opt}(f).$$

Dowód. Z warunku trójkąta oraz Lematu 1 wynikają następujące nierówności:

$$\begin{aligned}
\textsc{Mtm}^{\text{REQ}}(f) &= \sum_{i=1}^{D} d(P_{\text{MTM}}, \sigma_i) \leqslant \sum_{i=1}^{D} \left(d(P_{\text{MTM}}, a_0) + d(a_0, \sigma_i)\right) && (1)\\
&\leqslant D \cdot d(P_{\text{MTM}}, a_0) + \sum_{i=1}^{D} (a_0, \sigma_i) \leqslant \Phi_B/2 + \textsc{Opt}(f)\\
\textsc{Mtm}^{\text{MOVE}}(f) &= D \cdot d(P_{\text{MTM}}, v^*) \leqslant D \cdot d(P_{\text{MTM}}, a_0) + D \cdot d(a_0, v^*) && (2)\\
&\leqslant \Phi_B/2 + D \cdot d(a_0, v^*)\\
\Phi_F &= 2 \cdot D \cdot d(a_D, v^*) && (3)
\end{aligned}$$

Wystarczy zatem ograniczyć $d(a_\ell, v^*)$ dla $0 \leqslant \ell \leqslant D$.

$$D \cdot d(a_\ell, v^*) \;=\; \sum_{i=1}^{D} d(a_\ell, v^*) \leqslant \sum_{i=1}^{D} d(a_\ell, \sigma_i) + \sum_{i=1}^{D} d(v^*, \sigma_i) \;\; .$$

Ponieważ v^* został wybrany jako wierzchołek który minimalizuje sumaryczną odległość od żądań w całej fazie, otrzymujemy:

$$D \cdot d(a_\ell, v^*) \;\leqslant 2 \cdot \sum_{i=1}^{D} d(a_\ell, \sigma_i) \leqslant 2 \cdot \textsc{Opt}(f)$$

Podstawiając to ograniczenie do nierówności (1), (2) i (3), otrzymujemy tezę lematu. □

Dowód twierdzenia 1. Weźmy dowolną sekwencję σ i jej podział na fazy $\sigma = (f_1, f_2, \ldots, f_k, f_{k+1})$. Ostatnia faza ma być może mniej niż D kroków. Zauważmy, że zamortyzowany koszt w dowolnej z faz może zostać ograniczony przez funkcję zależną D i średnicy grafu G. Oznaczmy to ograniczenie przez β. Otrzymujemy wtedy

$$\begin{aligned}
\textsc{Mtm}(\sigma) + \Delta\Phi(\sigma) &= \sum_{i=1}^{k} \left(\textsc{Mtm}(f_i) + \Delta\Phi(f_i)\right) + \textsc{Mtm}(f_{k+1}) + \Delta\Phi(f_{k+1})\\
&\leqslant \sum_{i=1}^{k} \left(7 \cdot \textsc{Opt}(f_i)\right) + \beta\\
&\leqslant 7 \cdot \textsc{Opt}(\sigma) + \beta
\end{aligned}$$

Ponieważ $\Delta\Phi(\sigma)$ jest nieujemne otrzymujemy $\textsc{Mtm}(\sigma) \leqslant 7 \cdot \textsc{Opt}(\sigma) + \beta$, co kończy dowód konkurencyjności algorytmu Mtm. □

1.0.2 Algorytm Move-To-Local-Min

Zauważmy, że Mtm kiepsko radzi sobie z sytuacją, w której jest wiele minimów będących kandydatami na v^*. Wtedy Mtm wybiera jedno z nich, podczas gdy mógłby wybierać najbliższe. Warto również wybrać wierzchołek prawie minimalny jeśli jest on znacznie bliżej aktualnej pozycji pliku niż globalne minimum. Prowadzi nas to do algorytmu Move-To-Local-Min (Mtlm), który różni się od Mtm tylko wyborem v^*. Mtlm wybiera na v^* ten wierzchołek v, który minimalizuje sumę $2 \cdot \sum_{i=1}^{D} d(v, \sigma_i) + D \cdot d(P_{\text{MTLM}}, v)$.

Twierdzenie 2. Mtlm *jest* 5*-konkurencyjny.*

Dowód. Wystarczy pokazać odpowiednik Lematu 2, reszta dowodu jest identyczna jak w przypadku algorytmu MTM. Ograniczenie na $\text{MTLM}^{\text{REQ}}(f)$ jest identyczne, tj.

$$\text{MTLM}^{\text{REQ}}(f) \leqslant \Phi_{\text{B}}/2 + \text{OPT}(f) \ .$$

Z kolei

$$\begin{aligned}
\text{MTLM}^{\text{MOVE}}(f) + \Phi_{\text{F}} &\leqslant D \cdot d(P_{\text{MTLM}}, v^*) + 2 \cdot D \cdot d(a_D, v^*) \\
&\leqslant D \cdot d(P_{\text{MTLM}}, v^*) + 2 \cdot \sum_{i=1}^{D} d(v^*, \sigma_i) + 2 \cdot \sum_{i=1}^{D} d(a_D, \sigma_i) \\
&\leqslant D \cdot d(P_{\text{MTLM}}, v^*) + 2 \cdot \sum_{i=1}^{D} d(v^*, \sigma_i) + 2 \cdot \text{OPT}(f)
\end{aligned}$$

Z minimalności v^* w pierwszych dwóch składnikach powyżej możemy zamienić v^* na a_0 otrzymując

$$\begin{aligned}
\text{MTLM}^{\text{MOVE}}(f) + \Phi_{\text{F}} &\leqslant D \cdot d(P_{\text{MTLM}}, a_0) + 2 \cdot \sum_{i=1}^{D} d(a_0, \sigma_i) + 2 \cdot \text{OPT}(f) \\
&\leqslant \Phi_{\text{B}}/2 + 4 \cdot \text{OPT}(f) \ .
\end{aligned}$$

□

1.1 Randomizacja przeciwko adwersarzowi nieświadomememu

Jeśli przyjrzymy się algorytmowi MTM zauważymy, że dzieli on sekwencję wejściową na fazy po D kroków i przenosi plik na końcu takiej fazy. Gdybyśmy zaczęli fazę nie na początku sekwencji wejściowej, lecz po paru krokach nie zmieniłoby to analizy algorytmu. Nie pomogło (ani też nie przeszkodziłoby) nam to przeciwko adwersarzowi adaptującemu się. Natomiast może to pomóc przeciwko adwersarzowi oblivious, który nie wie gdzie dana faza się kończy czy zaczyna.

Te obserwacje są podstawą algorytmu CNT_k. Algorytm ten przechowuje globalny licznik C, który przyjmuje wartości z zakresu $[0, k]$. Jest on inicjowany wylosowaną (z rozkładem jednostajnym) liczbą naturalną z przedziału $[1, k]$. Przy dowolnym odwołaniu do pliku, C jest zmniejszane o 1. Jeśli w efekcie $C = 0$, plik jest przesuwany do wierzchołka, który właśnie zażądał do niej dostępu, a następnie licznikowi C jest przypisywane k.

Lemat 3. *Algorytm* CNT_k *jest* $\max\{2 + \frac{2d}{k}, 1 + \frac{k+1}{2d}\}$*-konkurencyjny przeciwko adwersarzowi oblivious.*

Dowód. Definiujemy funkcję potencjału równą

$$\Phi = (D + C) \cdot d(P_{\text{COUNT}}, P_{\text{OPT}}) \ .$$

Dzielimy każdy krok na dwa etapy; pokażemy, że zamortyzowany koszt algorytmu w każdym etapie jest ograniczony.

Etap 1. Przyjrzyjmy się najpierw zamortyzowanemu kosztowi obsługi żądania w σ_t. Oczywiście

$$\text{CNT}_k^{\text{odwołanie}} = d(P_{\text{COUNT}}, \sigma_t) \ .$$

Za obsługę tego żądania płaci spadek potencjału (związany ze spadkiem wartości licznika), tj. $\Delta\Phi = -d(P_{\text{COUNT}}, P_{\text{OPT}})$. Otrzymujemy

$$\text{CNT}_k^{\text{odwołanie}} + \Delta\Phi \leqslant d(P_{\text{OPT}}, \sigma_t) = \text{OPT} \ .$$

Dodatkowo, z prawdopodobieństwem $\frac{1}{k}$, po obsłudze żądania zachodzi $C = 0$. Z takim prawdopodobieństwem następuje przeniesienie pliku, którego koszt jest równy $D \cdot d(P_{\text{COUNT}}, \sigma_t)$ a

związana z nią zmiana potencjału to $\Delta\Phi = (D+k)\cdot d(\sigma_t, P_{\text{OPT}}) - D\cdot d(P_{\text{COUNT}}, P_{\text{OPT}})$. Otrzymujemy zatem, że

$$\begin{aligned}\mathbf{E}[\text{CNT}_k^{\text{przenosiny}} + \Delta\Phi] &= \frac{1}{k}\cdot[D\cdot d(P_{\text{COUNT}}, \sigma_t) - D\cdot d(P_{\text{COUNT}}, P_{\text{OPT}}) + (D+k)\cdot d(\sigma_t, P_{\text{OPT}})] \\ &\leqslant \frac{1}{k}\cdot[D\cdot d(P_{\text{OPT}}, \sigma_t) + (D+k)\cdot d(\sigma_t, P_{\text{OPT}})] \\ &= \frac{2D+k}{k}\cdot \text{OPT}\ .\end{aligned}$$

Sumując otrzymujemy

$$\mathbf{E}[\text{CNT}_k + \Delta\Phi] \leqslant \left(2 + \frac{2D}{k}\right)\cdot \text{OPT}\ .$$

Etap 2. Załóżmy teraz, że OPT przenosi plik z wierzchołka P_{OPT} do P'_{OPT}. Wtedy zmiana potencjału to $(D+C)\cdot[d(P_{\text{COUNT}}, P'_{\text{OPT}}) - d(P_{\text{COUNT}}, P_{\text{OPT}})] \leqslant (D+C)\cdot d(P_{\text{OPT}}, P'_{\text{OPT}})$. Zauważmy teraz, że wartość oczekiwana licznika C wynosi $\frac{k+1}{2}$, a zatem oczekiwana zmiana potencjału to

$$\begin{aligned}\mathbf{E}[\Delta\Phi] &= \left(D + \frac{k+1}{2}\right)\cdot d(P_{\text{OPT}}, P'_{\text{OPT}}) \\ &= \left(1 + \frac{k+1}{2D}\right)\cdot \text{OPT}\ .\end{aligned}$$

Porównując wyniki w dwóch etapach otrzymujemy tezę lematu. □

Poniższe twierdzenie pozostawiamy bez dowodu jako proste ćwiczenie rachunkowe.

Twierdzenie 3. *Istnieje taki wybór k (jako funkcji D), że dla D dążącego do nieskończoności, konkurenyjność algorytmu* CNT_k *zbiega do $(1+\phi)$, gdzie $\phi = \frac{1+\sqrt{5}}{2}$ jest złotym podziałem.*

1.2 Optymalny algorytm dla dwóch wierzchołków

W tej części przedstawimy pewną technikę nazywaną funkcją pracy (*work function*), którą wykorzystamy do konstrukcji optymalnego algorytmu zrandomizowanego dla grafu dwuwierzchołkowego.

1.2.1 Opis rozkładowy algorytmu

Okazuje się, że do prezentacji tej techniki, wygodniej jest korzystać z innego opisu algorytmu, opisu rozkładowego. Mianowicie zamiast określać, że algorytm przenosi plik z pewnym prawdopodobieństwem, będziemy definiować algorytm przez jego rozkład prawdopodobieństwa nad możliwymi stanami. W przypadku problemu przenoszenia pliku, w kroku t będziemy określać jaki jest rozkład prawdopodobieństwa pozycji pliku.

W naszych rozważaniach ograniczymy się do grafu dwuwierzchołkowego, którego wierzchołki oznaczymy przez a i b. Bez straty ogólności możemy założyć, że odległość między a i b wynosi 1. W każdym kroku będziemy określać rozkład μ; taki rozkład oznacza, że algorytm ma plik w a z prawdopodobieństwem $\mu(a)$ a w b z prawdopodobieństwem $\mu(b) = 1 - \mu(a)$. Oczywiście, żeby taki opis miał sens, adwersarz musi być nieświadomy.

Zauważmy, że jeśli algorytm ma w kroku t rozkład μ, a odwołanie jest w wierzchołku a, to w wartości oczekiwanej algorytm płaci za to odwołanie $\mu(a)\cdot 0 + \mu(b)\cdot 1 = \mu(b)$. Musimy teraz pokazać, że przejście między dwoma rozkładami da się wykonać i określić jaki jest koszt takiego przejścia.

Lemat 4. *Dla grafu dwuwierzchołkowego (a,b) z $d(a,b) = 1$ i dwóch rozkładów prawdopodobieństwa μ i μ', koszt przejścia pomiędzy nimi wynosi $D\cdot|\mu(a) - \mu'(a)|$.*

Dowód. Załóżmy, że mamy zrandomizowany algorytm, który po przeczytaniu pewnego fragmentu wejścia ma plik w wierzchołku a z prawdopodobieństwem $\mu(a)$ i w wierzchołku b z prawdopodobieństwem $\mu(b)$.

Jeśli $\mu'(a) = \mu(a)$, to algorytm nie przenosi pliku, związany z tym koszt jest równy zero i teza twierdzenia jest spełniona. W przeciwnym przypadku, bez straty ogólności możemy założyć, że $\mu(a) > \mu'(a)$. Wtedy strategia dla algorytmu jest następująca:

- jeśli plik jest w b, nic nie rób,
- jeśli plik jest w a, przenieś ją do b z prawdopodobieństwem $p = \frac{\mu(a)-\mu'(a)}{\mu(a)}$.

Zauważmy, że prawdopodobieństwo że algorytm skończy ze plikiem w wierzchołku a wynosi $\mu(a) \cdot (1 - p) = \mu'(a)$, a więc otrzymaliśmy zadany rozkład prawdopodobieństwa. Oczekiwany koszt, jaki poniósł nasz algorytm wynosi $p \cdot \mu(a) \cdot D = D \cdot (\mu(a) - \mu'(a))$. □

1.2.2 Funkcje pracy i algorytm EDGE

Funkcje pracy (*work functions*) są techniką, która często pozwala na konstrukcję algorytmów online osiągających optymalne współczynniki. Idea polega na tym, że w każdym kroku t dla dowolnego stanu x obliczamy $w_t(x)$, koszt optymalnego rozwiązania widzianej do tej pory sekwencji, które kończy działanie w stanie x. Funkcję w_t nazywamy funkcją pracy w kroku t; indeks t będziemy zawyczaj pomijać.

Oczywistą własnością funkcji w jest: $\textsc{Opt}(\sigma) = \min_x w_{|\sigma|}(x)$. Warto zwrócić uwagę, że choć koszt optymalnego rozwiązania na dowolnym prefiksie wejścia o długości ℓ jest równy $\min w_\ell(x)$, to optymalny algorytm nie musi być (często nie może być) optymalny na każdym z prefiksów. Jednak w analizie algorytmów możemy (i będziemy) zakładać, że koszt Opt w danym kroku t jest równy $\min_x w_t(x) - \min_x w_{t-1}(x)$.

Zależność między funkcją pracy a kosztem optimum prowadzi do pomysłu, że algorytm powinien przebywać możliwie najczęściej w stanie, który minimalizuje funkcję w. Jeśli jednak tak zdefiniujemy algorytm, to zazwyczaj adwersarz jest w stanie wygenerować sekwencję, w której to minimum zmienia się często i algorytm płaci dużo za zmianę stanów.

Przy algorytmach zrandomizowanych można sobie z tym poradzić, zmieniając stan z małym prawdopodobieństwem. Ta idea jest podstawą algorytmu Edge. Zdefiniujmy przesunięcie $g := w(b) - w(a)$. Zauważmy, że jeśli algorytm optymalny może skończyć z pewnym kosztem $w(a)$ w wierzchołku a, to może skończyć z kosztem co najwyżej $w(a) + D$ w wierzchołku b (może wziąć strategię kończącą w a i na samym końcu przenieść plik do b). Dlatego też $w(b) \leqslant w(a) + D$ i zatem z symetrii wynika, że $g \in [-D, D]$.

Algorytm Edge ustala pewien rozkład prawdopodobieństwa na podstawie wartości g, mianowicie

$$\mu(a) := \frac{D+g}{2D} \ , \qquad \mu(b) := \frac{D-g}{2D} \ . \tag{4}$$

Twierdzenie 4. *Algorytm* Edge *jest ściśle* $(2 + \frac{1}{2D})$*-konkurencyjny przeciwko adwersarzowi nieświadomememu na grafach dwuwierzchołkowych.*

Dowód. Dowód jest tak naprawdę klasyczną analizą zamortyzowaną, choć nie napiszemy bezpośrednio funkcji potencjału. Po pierwsze obliczymy jakie są koszty Edge i Opt w zależności od zmiany g. Po pierwsze zauważmy, że jeśli g się zmienia, to zmienia się o 1 i wtedy związany z tym oczekiwany koszt przenosin pliku to $D \cdot \frac{1}{2D} = \frac{1}{2}$.

Bez straty ogólności, możemy założyć, że g jest nieujemne, czyli $w(b) \geqslant w(a)$. Rozpatrzmy parę przypadków:

1. Jeśli odwołanie jest w a i $g < D$, to g zwiększa się o 1. W efekcie Edge płaci $\mu(b)$ za żądanie i $1/2$ za przenosiny pliku, zaś Opt nie płaci nic.
2. Jeśli odwołanie jest w a i $g = D$, to g nie zmienia się. Zauważmy, że w takim przypadku $\mu(a) = 1$ i zatem Edge nie płaci nic. Opt również nic nie płaci.

3. Jeśli odwołanie jest w b i $g = 0$, to jest tak samo jak gdyby odwołanie było w a (patrz przypadek 1).

4. Jeśli odwołanie jest w b i $g > 0$, to g zmniejsza się o 1. W efekcie EDGE płaci $\mu(a)$ za żądanie i $1/2$ za przenosiny pliku, zaś OPT płaci 1.

Zobaczmy teraz jak wyglądają koszty w zależności od zmiany $|g|$:

1. Jeśli $|g|$ się nie zmienia, to nic się nie dzieje.

2. Jeśli $|g|$ rośnie o 1, to $\mathbf{E}[\text{EDGE}] = 1 - \frac{g}{2D}$, a $\text{OPT} = 0$.

3. Jeśli $|g|$ maleje o 1, to $\mathbf{E}[\text{EDGE}] = 1 + \frac{g}{2D}$, a $\text{OPT} = 1$.

Zauważmy, że trudny jest przypadek 2, bo wtedy $\text{OPT} = 0$. Zróbmy zatem następujący eksperyment myślowy: W momencie kiedy $|g|$ się zmniejsza (np. z $x + 1$ do x) oprócz płacenia za faktyczny oczekiwany koszt EDGE odłożmy dodatkowo kwotę, która zapewni nam, że będziemy mogli zapłacić za zwiększenie $|g|$ od x do $x + 1$. Oznacza to, że zamortyzowany koszt w momencie zmniejszenia się $|g|$ to $(1 - \frac{x+1}{2D}) + (1 + \frac{x}{2D}) = 2 + \frac{1}{2D}$, natomiast zamortyzowany koszt w momencie zwiększenia się $|g|$ to 0. Zatem konkurencyjność wynosi $2 + \frac{1}{2D}$.

Zauważmy dodatkowo, że ponieważ zaczynamy od maksymalnej możliwej wartości $|g|$, tj D, konkurencyjność ta jest ścisła. □