國 立 交 通 大 學

電 信 工 程 學 系 碩 士 班

碩 士 論 文

在變動位元速率串流下之平順演算法

A Smoothing Algorithm in Variable Bit Rate Streaming

研 究 生：謝 明 昇

指導教授：張 文 鐘 博士

中 華 民 國 九 十 三 年 八 月

在變動位元速率串流下之平順演算法

A Smoothing Algorithm in Variable Bit Rate Streaming

研 究 生：謝明昇　　Student：Ming-Sheng Hsieh

指導教授：張文鐘 博士　　Advisor：Dr. Wen-Thong Chang

國立交通大學

電信工程學系碩士班

碩士論文

A Thesis

Submitted to Department of Communication Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Communication Engineering

August 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年八月

在變動位元速率串流下之平順演算法

研究生：謝明昇　　指導教授：張文鐘　博士

國立交通大學電信工程學系碩士班

中文摘要

隨著網路頻寬的增加與視訊壓縮技術的進步，藉由隨選視訊、即時視訊轉播等服務，即時接收到高品質的視訊已慢慢變成可能而且可接受。VBR constant quality 的視訊壓縮的方式可以得到較佳的品質，不過也會造成資料流突衝(bursty)的特性，它會造網路服務較大的負擔。這時我們就可利用平順演算法來改善這個問題，所謂平順簡單講就是在減少資料流突衝變異情況。最佳平順演算法目的是達成速率整體變異度最小，也就是整體速率標準差最小的一種平順演算法。本論文首先針對此演算法的方法作介紹與探討，並將它用在實際的 video trace 上作模擬，觀察客戶端緩衝區大小與統計特性的關係，像是峰值與標準差。根據模擬的結果再作討論與分析。其次我們引入每單位時間傳送最小單位精確度為位元組的情況，針對所會遇到的問題作修正，並用實際的 video trace 作模擬，最後並比較原本與修正過後表現的差異，並探討其差異的原因。

A Smoothing Algorithm in Variable Bit Rate Streaming

Student: Ming-Sheng Hsieh Advisor: Dr. Wen-Thong Chang

Institute of Communication Engineering

National Chiao Tung University

Abstract

With the increase of bandwidth and the progress in image compression, it has become acceptable to watch real-time high quality video through the network.. VBR compression can gain better quality but it introduces burstier traffic. This will cause a burden to the network service, making the bandwidth usage inefficient. Smoothing algorithm can deal with this problem; it can make the traffic less burstier. The optimal smoothing algorithm can optimizes the traffic in terms of the variance. In this thesis, first we introduce and analyze this algorithm; then we simulate it on real video traces, discussing the relation between buffer size and statistical data, such as peak frame size and variance. Second, we introduce the minimum unit(byte) in transmission rate to this algorithm and solve the problems it meets with. Then we simulate this algorithm on the same video trace and discuss its characteristics. Finally, we compare the original algorithm and the modified one.

誌　謝

研究所的生涯，已經要結束了。學習愈多才愈知自己的不足，更明白要謙虛的道理以及終身學習的重要。

首先，我想要感謝我的指導教授張文鐘博士，以很開放的方式，引領我進多媒體通訊的領域，並在論文裡面給我指導。還有感謝當天的口試委員的范國清博士、余孝先博士、何文楨博士，謝謝你們當天不吝惜提出您們的看法與指教，讓我的論文更臻完備。

其次，我要感謝我實驗室的伙伴承軒、政儒、承霈、旃偉、心賢，大家一起砥礪打氣使研究的路不孤單；遠青、豬頭、小盧、趙爸、德倫學長與孋雅學姐，碩一剛來，有你們親切的帶領和有經驗分享，讓我們比較能進入狀況；學弟們，謝謝你們的幫忙、協助；室友們永隆、明宗、震軒，謝謝你們常常和我分享生活點滴，生活上有照應。家庭聯合會的伙伴，在我剛來新竹時人生地不熟的時候，給我人際上的支援，讓我在這兩年課餘時還有機會接觸到其他東西，讓我有志願服務的機會。玫伶，同鄉的好同學，謝謝妳不吝惜和我交流一些生活上的意見。老猴，謝謝的來電，可以在研究之餘談比較輕鬆的話題。302的全體伙伴，謝謝你們不經意捎來的問候，大學畢業後大家比較忙了，比較少聯絡，但是每當覺得很難過或壓力很很大的時候，我常常會想到你們，想到我們一起出遊的快樂時光，就更有力量前進。

最後，我要感謝我的家人，爸爸、媽媽、姐姐、哥哥、弟弟，因為你們在背後的支持，才可以讓我可以在研究上無後顧之憂；可愛beer，讓枯躁日子增添趣味。感謝上天，也許這一切冥冥都有您的代領，也願您看顧以上每一個關心我和我所關心的人。

目　　錄

表　目　錄

圖　目　錄

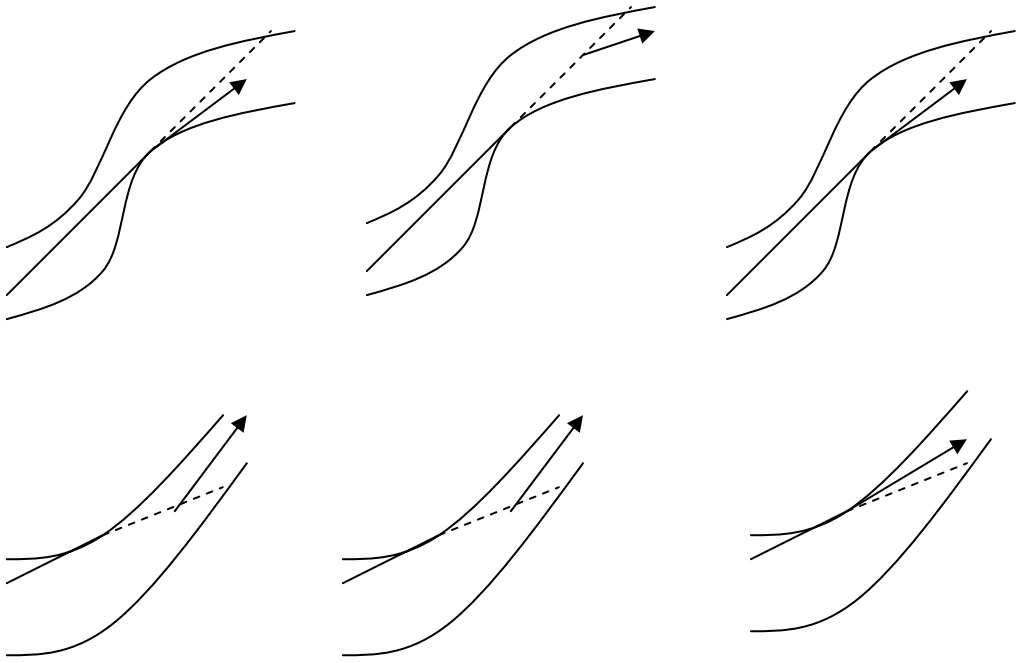
第一章

緒論

1.1 研究動機

隨著頻寬的增加與視訊壓縮技術的進步，要透過網路欣賞到高品質的即時視訊服務，已不再是遙不可及的夢想。不過其實在傳送即時視訊上是有許多地方要克服的，每個不同類型網路有不同的限制，更何況每個使用者的資源也不同，原本網路規格適用性其實也不足，要解決這樣的問題需要每個階層一起努力克服的。其中一個問題，VBR (變動位元速率) constant quality 的視訊壓縮方式雖然可以保有每個畫面品質的一致，不過卻造成這個資料流有比較突衝(bursty)的特性，資料量隨時間的變異度很大，這在網路服務供給面來說是一大負擔。如果可以把它變成一個比較平緩的資料流，對整個網路的負擔應該可以大為減輕。平順演算法就是在解決這樣問題的方法，利用客戶端的緩衝區，用先導(work-ahead)的方式將部分資料傳到客戶端，有了緩衝區的幫忙，就可以比較彈性調整傳輸排程，即可以讓整個資料流較為平順。

平順演算法依不同的目標，有不同的運作方式，差異主要是在速率增加和速率減少的情況區段選取原則的不同，像是 MCBA (minimum changes bandwidth allocation) [13]它是最小化速率改變次數，如圖 1.1(b)，在速率增加或速率減少的情形都是以選取延伸愈長的區段為目標，以達到速率改次數最小；MVBA(minimum variability) [1]最小化整體速率的標準差，如圖 1.1(c)，不管是速率增加或速率減少的情況，都是以速率變異最小為目標，以達到整體整率標準差最小；CBA (critical bandwidth allocation) [12]最小化速率增加的次數，如圖 1.1(a)，雖然有速率增加和速率減少的情況，但是它只在最小化速率增加的次數，在速率增加的情形採用和 MCBA 相同的選取方式，在速率減少的情況，則是採用和 MVBA 相同的的選取方式，可以說是兩者的綜合體。為了達成不同的目標，其選取原則也會有差異。本論文是對最佳平順演算法作探討，它就是上面介紹的第二種平順演算法，主要是在整體速率變異度作最佳化，最小化整體速率的標準差。最佳平順演算法的運作方式以及其成效與分析，就是本論文的主要探討核心。



(a) (b) (c)

圖 1.1 三種不同的平順演算法【註 1】

1.2 論文架構

基本上論文的編排方式，第一章緒論，從研究動機與基本背景作出發，第二章是平順演算法概念的介紹與最佳平順演算法的說明，第三章則是在利用此演算法在真實 video trace 的模擬結果與分析，第四章是架構在最小單位限制的修正演算法與其結果分析，並與原演算法作比較，第五章則是結論。

【註 1】圖 1.1 上圖代表速率減少的狀況，下圖代表速率增加的狀況。

第二章

最佳平順演算法的說明與分析

CBR、variable quality 的視訊，對於網路服務供給面上來說，複雜性相較起來單純，可是視訊品質在人眼感覺上並不夠好；VBR constant quality 的壓縮方式影像品質佳，由於它有資料量隨時間變異性很大的特性，所以複雜度較高。以現今天數位視訊壓縮技術 MPEG 而言，短期的資料暴衝(burst)、資料變異，是由於 I、P、B frame 的關係；長期則是由於 scene content 的不同。

針對像 VBR 這樣的資料，我們如果直接傳，對於整個網路服務來說，會造成資源的浪費，網路的使用效能也不夠高。為了因應這個問題，才有了 smoothing 技術的探討。smoothing 照字面來講，就是使平順的意思，平順什麼呢？就是平順資料量，也是是使資料量的變異度降低，在資料量負擔小一點的時間點多傳一點，資料量負擔大的時間點少傳一點。最理想的狀況就是 CBR 情況，如圖 2.1(b)，也就是傳輸速率剛好為原本資料量總和取平均，都沒變就最平順，可是因為有網路資源緩衝區大小、等待的時間的限制，造成傳輸上會受限，所以大部分只能採取 piece-wise CBR 的方式才可行，以下均會對這些

問題作探討。

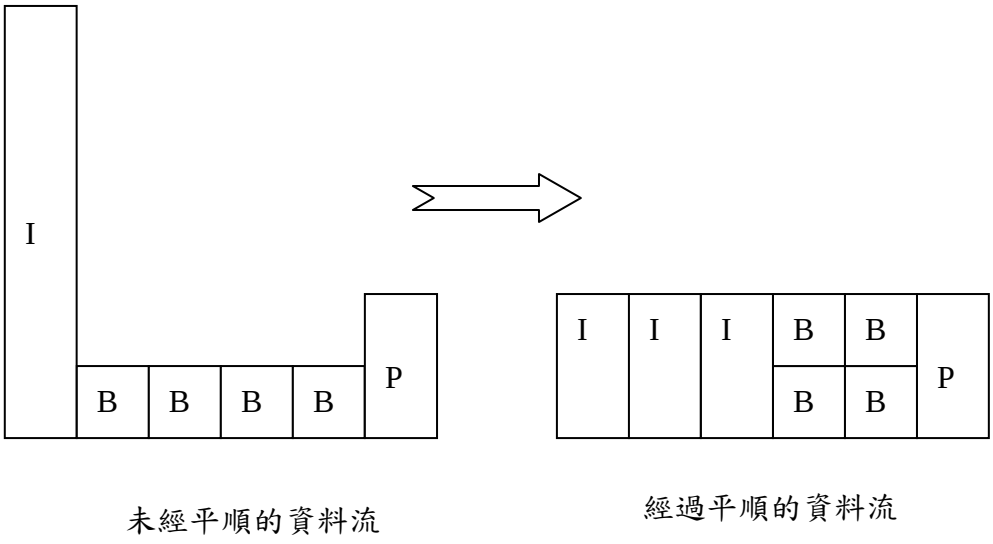


圖 2.1 平順示意圖(橫軸為時間，縱軸為資料量)

根據[1]，經過最佳平順後，可以大大地使資料變動率降低，網路資源的使用率可以大大的提升。以下就對最佳平順的架構與其原理作一番探討。

2.1 傳輸架構

用一般影像傳輸架構，伺服端(server)和用戶端(client)先建好連線後，伺服端經由傳輸路徑將資料送至用戶端，用戶端利用其緩衝區大小作為緩衝的要件。用戶端的緩衝區就是讓我們可以作平順(smoothing)的關鍵。藉由先導(work-ahead)將一部分視訊資料先送到客端的緩衝區暫存，下個時刻的工作量就可減輕了，尤其當下個時刻資料量特

別大的時候。其本上它的原理，就是資料量的分擔，儘可能的利用資源。

由於針對的是視訊資料，因而有即時性的問題。伺服端傳輸的資料量如果不及用戶端播放的資料量就會發生 underflow 的問題，接下來的資料不夠播；伺服端傳輸量如果大於用戶端播放量和可暫存住的量的和又會發生 overflow 的問題，會造成資料的遺失。所以由伺服端傳至用戶端的資料必須介於這兩個限制之中，才算是有意義的排程。如圖 2.2 所示。以下就對上下限作一番說明。

2.2 傳輸排程上下限

N	為串流視訊資料的 frames 數。
b	客戶端的緩衝區大小
$d(t)$	單位時間的播放量
$D(t)$	播放的累積資料量
$s(t)$	伺服端傳送給客戶端每單位時間的資料量
$S(t)$	伺服端傳送給客客端累積資料量
$B(t)$	客戶端所能接收的累積資料量上限
$L(t)$	客戶端所需接收的累積資料量下限

表 2.1 所用到的參數表

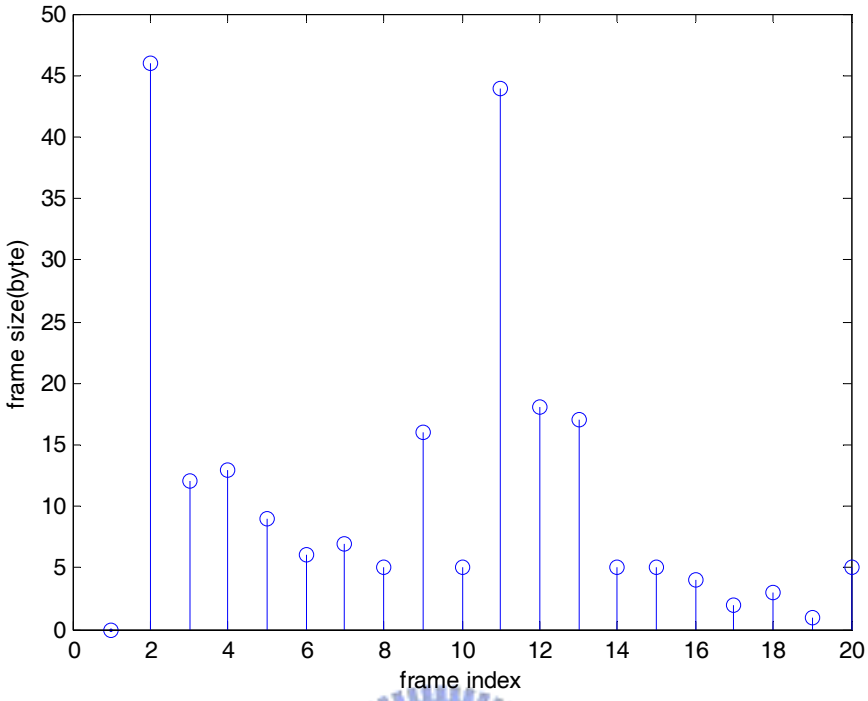


圖 2.2 輸入的 video trace 【註 2】

2.2.1 傳輸下限

$S(t)=s(1)+s(2)+\cdots+S(t)$ 為傳送到 t 時刻的傳送累積資料量，$D(t)=d(1)+d(2)+\cdots+d(t)$ 為到 t 時刻時的播放累積資料量。累積傳進來的資料至少要大於累積即將要播放的資料量，所以 $S(t)\geq D(t)$，故可得傳輸下限:

$$L(t)=D(t) \tag{2.2-1}$$

如圖 2.2 就是原本的 video trace。利用 video trace 就可以建出一個下限，圖 2.3。

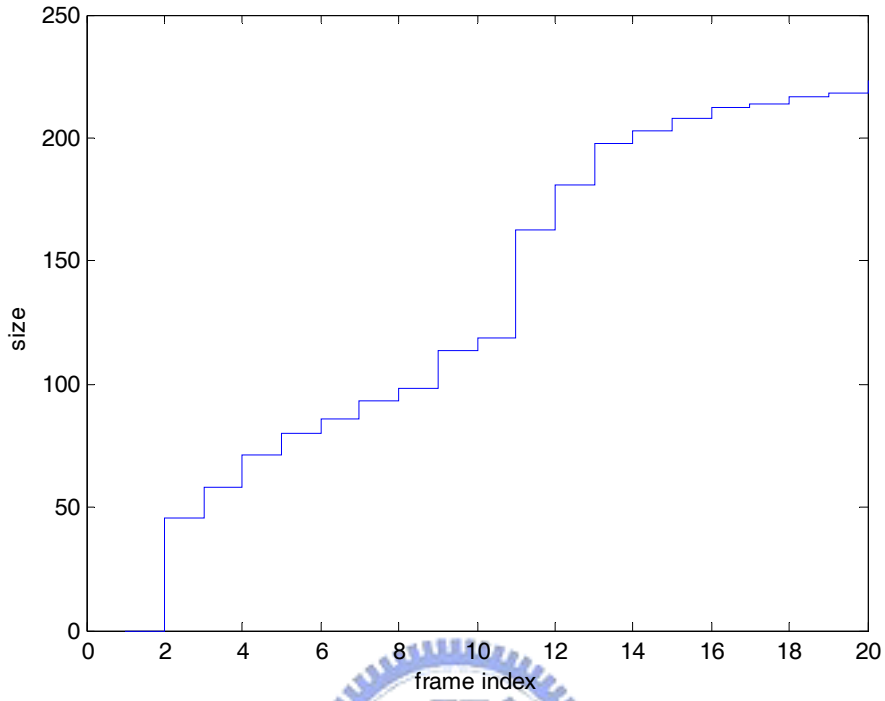


圖 2.3 傳輸下限

2.2.2 傳輸上限

傳送累積的資料量扣除播放累積的資料量必須小於緩衝區量，意即 $S(t)-D(t-1)\le b$ 所以 $S(t)\le D(t-1)+b$ ；又累積傳送量不能大於全部播放累積播放量，所以 $S(t)\le D(N)$ 。故我們可以推出傳輸上限:

$$B(t)=\min(D(t-1)+b, D(N)) \qquad (2.2\text{-}2)$$

利用 video trace 和緩衝區大小，即可建立一個上限。如圖 2.4。

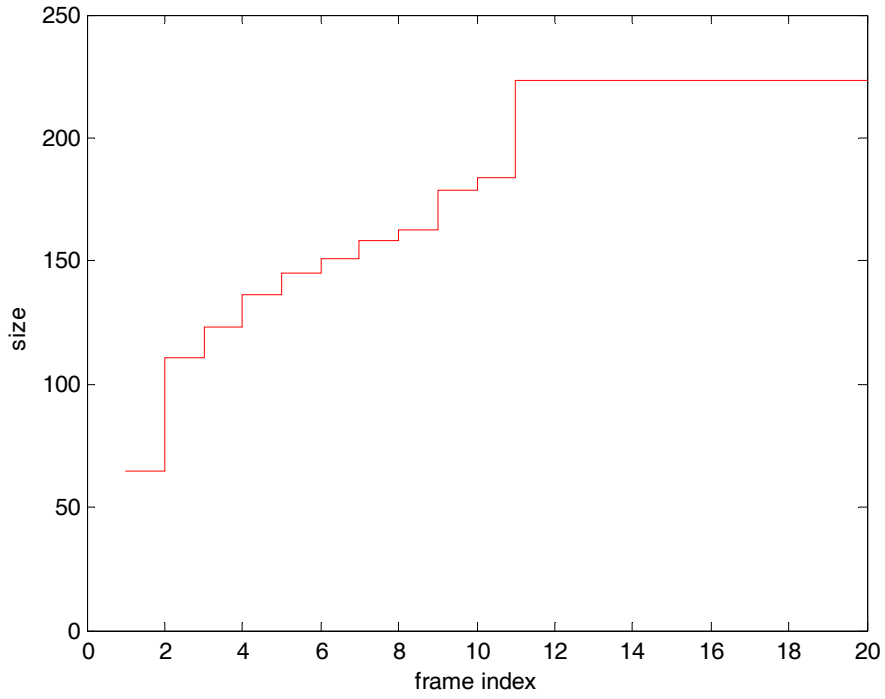


圖 2.4 傳輸上限

2.3 可行的傳輸排程

由上已知傳輸的下限就是 $L(t)$ $(D(t))$，上限就是 $B(t)$，上下限一起看，如圖 2.5 所示。只要是符合 $D(t) \le S(t) \le B(t)$ 的傳輸排程，也就是介於上下限之間的排程，就是所謂可行的排程，如圖 2.7 所示；不符合的就稱為不可行排程，如圖 2.6 所示。(像圖 2.6 這個情況下，CBR 就不可行。)

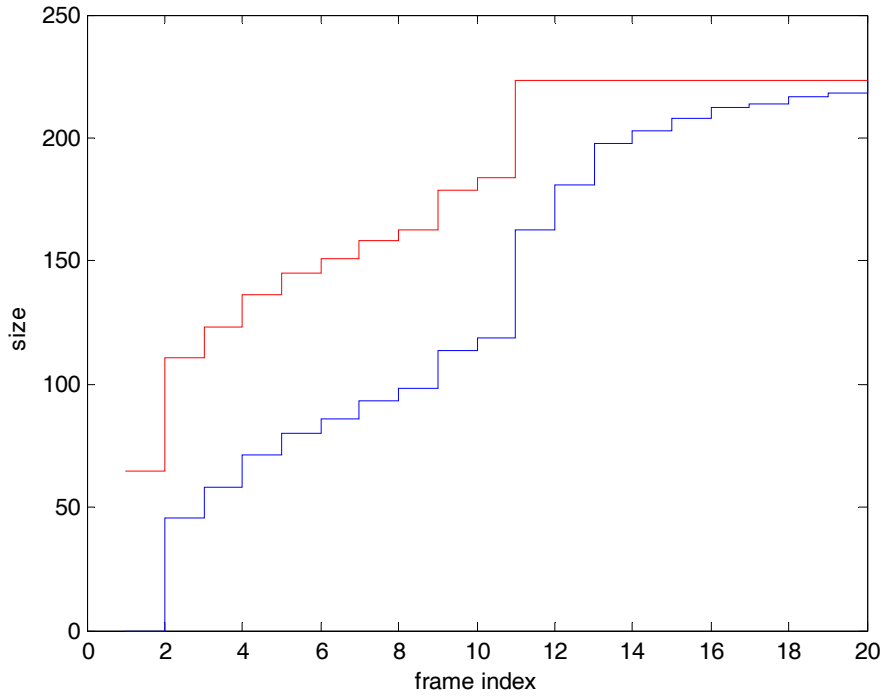


圖 2.5 傳輸上下限

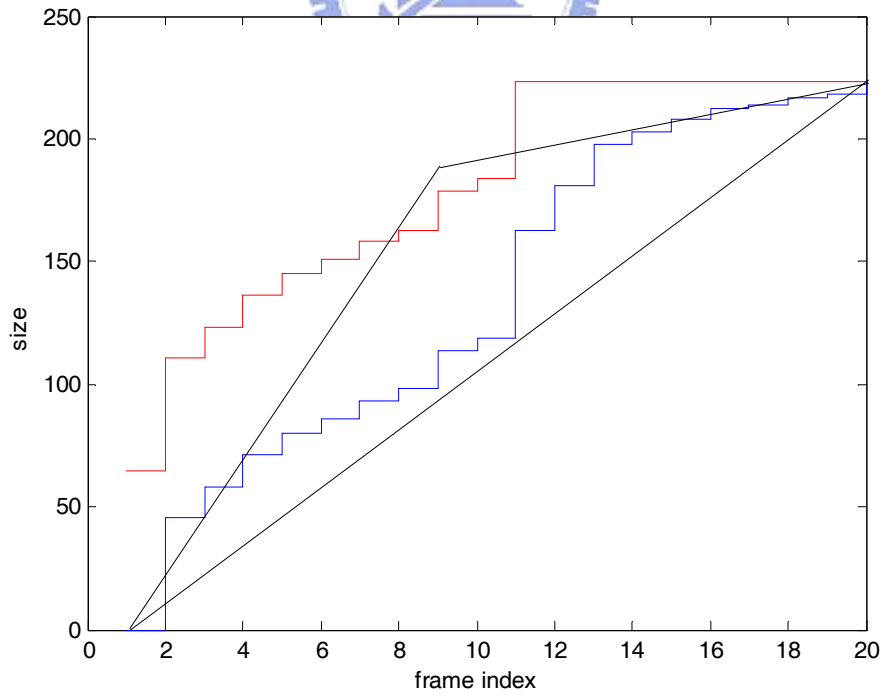


圖 2.6 不可行排程(infeasible schedules)

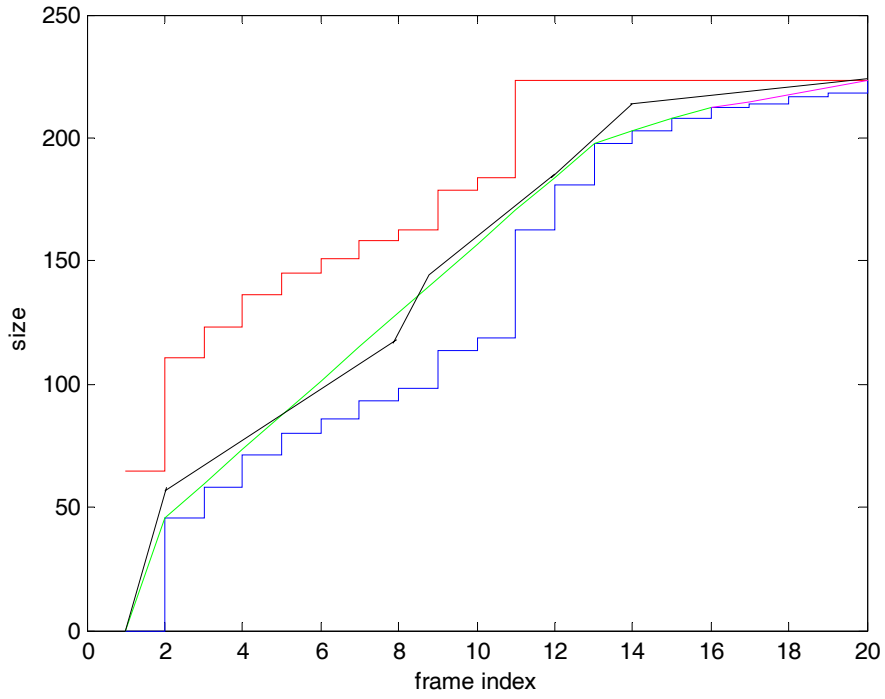


圖 2.7 可行排程(feasible schedule)

2.4 最佳平順演算法(Optimal Smoothing Algorithm)[1]

要符合以上在上下限之間的條件就是可行，一般情況通常有很多組可能。哪一組才是最佳的呢？根據[1]它提出一種演算法，達到了所謂最佳，它的最佳是在 rate variability 上，也就是整體速率變異度最小的一種平順方式，在數學統計上也就是以整體傳輸速率標準差最小為目標，它的演算法如下所示。其證明主要是利用蓋理論(Majorization theory)[14]，我們在附錄一有節錄證明，是引自[1]，詳細的證明請參考[1][14][15]。

2.4.1 演算法說明

因為 CBR 是最平順的，但是整個都採 CBR 不可行，所以我們採用 piecewise CBR 的方式，也就是由許多段 CBR 所組成的排程(schedule)。選取的方式是以可以延伸最長的片段為主，延到不能再延時，例如碰到 underflow 或 overflow 的時候，再從那一點作轉折。轉折點都剛好在碰到上下限的地方。其演算法的 pseudo-code[1]如圖 2.8 所示，輸入為 video trace 和緩衝區大小，輸出為最佳化排程(optimal schedule)。

```
PROCEDURE Find_optimal_schedule (d(t), b)
ts = 0, te = 1, q = 0
c_max = b, t_B = 1, c_min = d(1)
t_D = 1
do{
    te' = te + 1
    if (c_max < (D(te') − (D(ts) + q)) / (te' − ts))
    {
        output_segment(t_B − ts, c_max)
        ts = t_B, te = t_B + 1, q = B(t_B) − D(t_B)
    }
    elseif (c_min > (B(te') − (D(ts) + q)) / (te' − ts) || te' == n)
    {
        output_segment(t_D − ts, c_min)
        ts = t_D, te = t_D + 1, q = 0
    }
    else
    {
        te = te'
    }
}while(ts <= N)
END PROCEDURE
```

圖 2.8 Optimal smoothing algorithm pseudo-codes[1]

詳細的運作方式以下文字配合圖來作說明：

1、 假設建立好原本的上下限，如圖 2.9(a)。

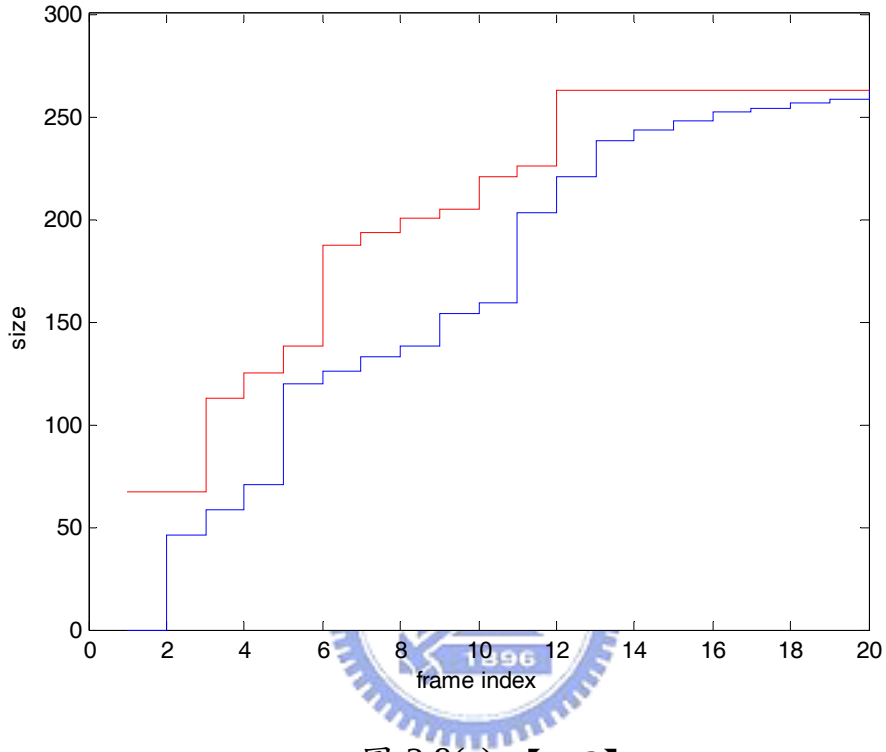


圖 2.9(a) 【註 3】

2、由時間點 1 至 2 的一個單位時間間隔，我們可以決定初始的最大速率 cmax=(B(2)-B(1))/(2-1)=B(2)-B(1)，最小傳輸速率 cmin= (D(2)- D(1))/(2-1)=D(2)-D(1)。cmax、cmin 分別代表圖中上下兩條斜線段的斜率，也就是在這個時間間隔內所容許的最大傳輸速率與最小傳輸速率，如圖 2.9(b)。

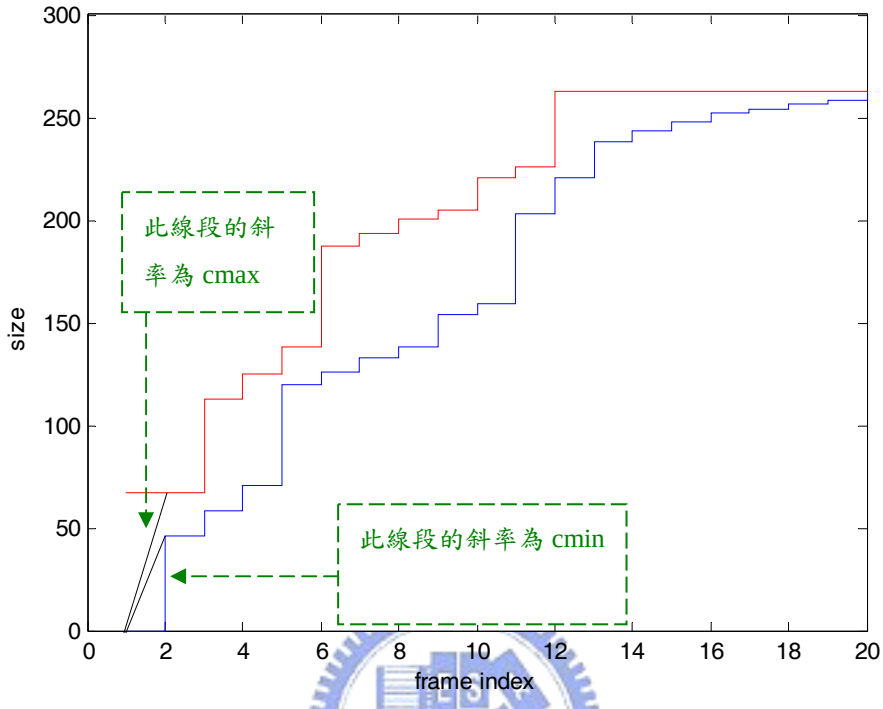


圖 2.9(b)

3、起始點相同，延伸到下一時刻，也就是由時間點 1 到 3 的兩單位時間間隔，從起始到這個時間間隔結束點滿足需求的最大平均速率與最小平均速率，為 cmax'與 cmin'，如圖 2.9(c)。(cmax'、cmin'只是用來作為判斷，並不一定可行。cmax'=(B(3)-B(1))/(3-1)= (B(3)- B(1))/2，cmin'=(D(3)-D(1)) /(3-1)= cmin'=(D(3)-D(1)) /2。)

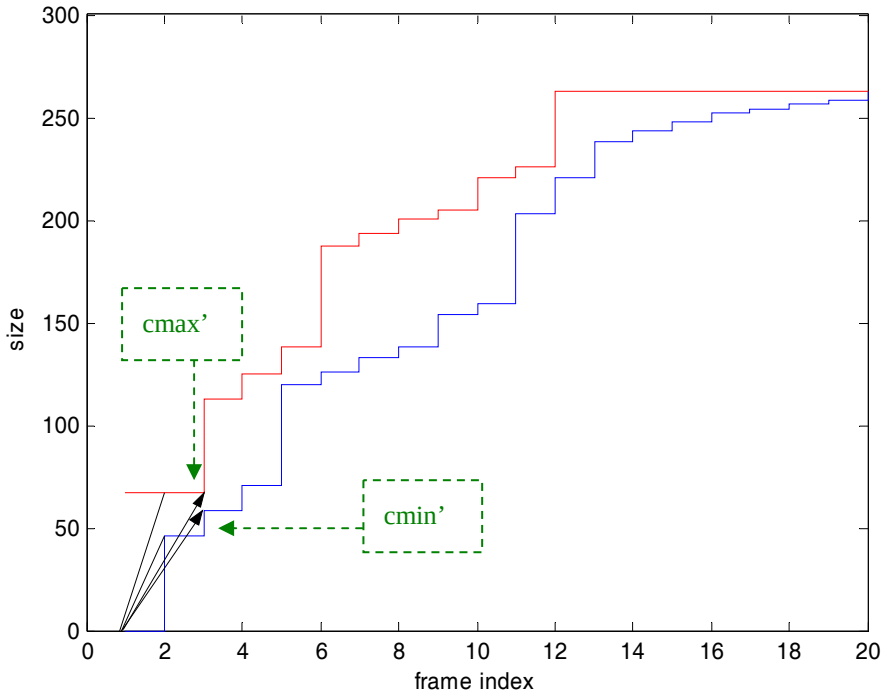


圖 2.9(c)

4、比較 cmax、cmin 與 cmax'、cmin'，如圖 2.9(d)。

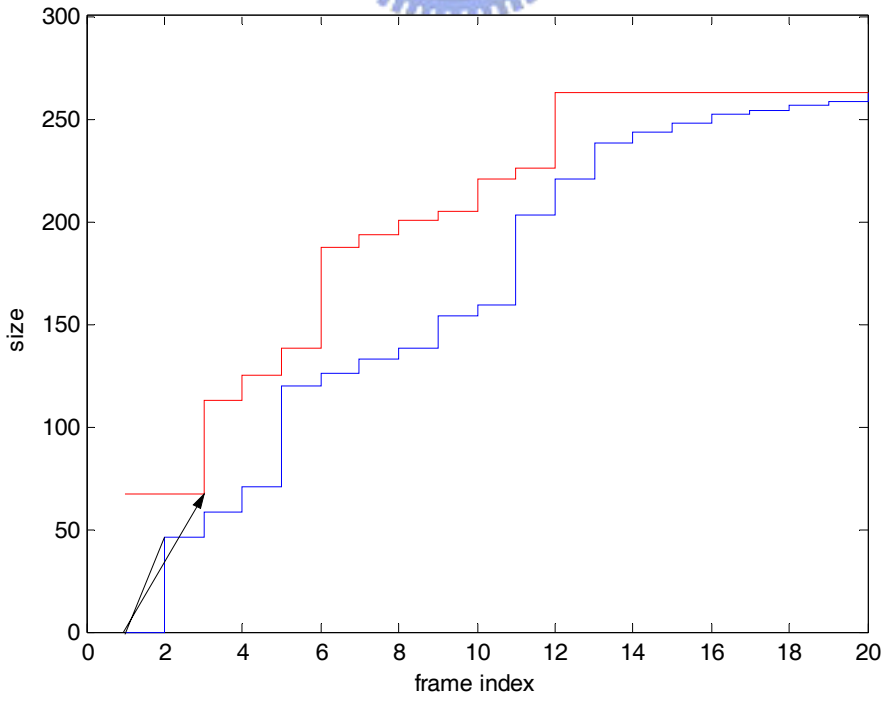


圖 2.9(d)

5、因為 cmin 大於 cmax’，case2 的情況，這意味著如果繼續用這個時刻最小的速率去傳到下個時刻還是會 overflow，所以有必要作速率的改變。可是這個區段有很多選擇，要選擇哪一個速率呢？就是因為下個時刻速率變慢，而我們的目標是使變動率最小，又這個時刻的速率最接近下一個時刻的速率的線段就是 cmin，所以我們便選擇 cmin 為這個區段的速率。決定好的線段，如圖 2.9(e)。

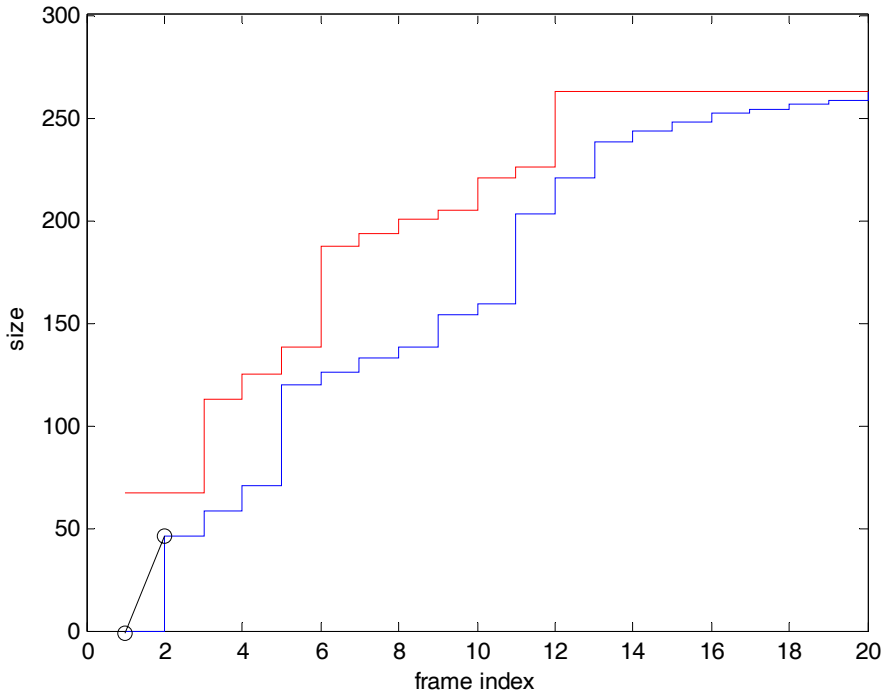


圖 2.9(e)

6、由這個線段的結束當下一個線段的起始，也就是以時間點 2 為起始點，由時間點 2 至時間點 3 的區間，我們又可得到初始的 cmax、cmin，如圖 2.9(f)。

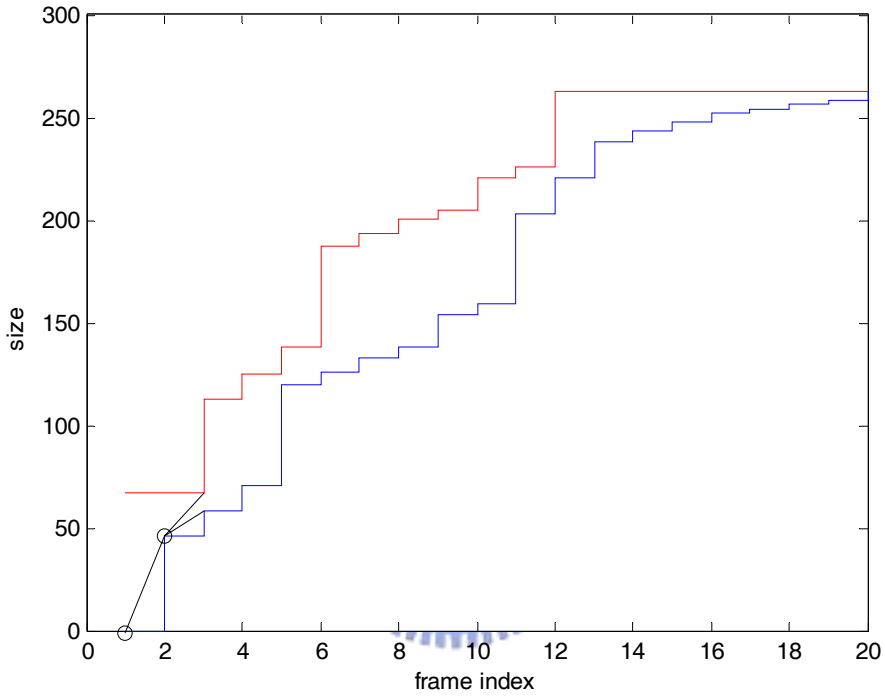


圖 2.9(f)

7、由相同的起始點，延伸到下一個時刻，也就是時間點 2 至時間點 4 的兩個時間間隔，與之前方法相同，我們可以得到 cmax'和 cmin'，如圖 2.9(g)。

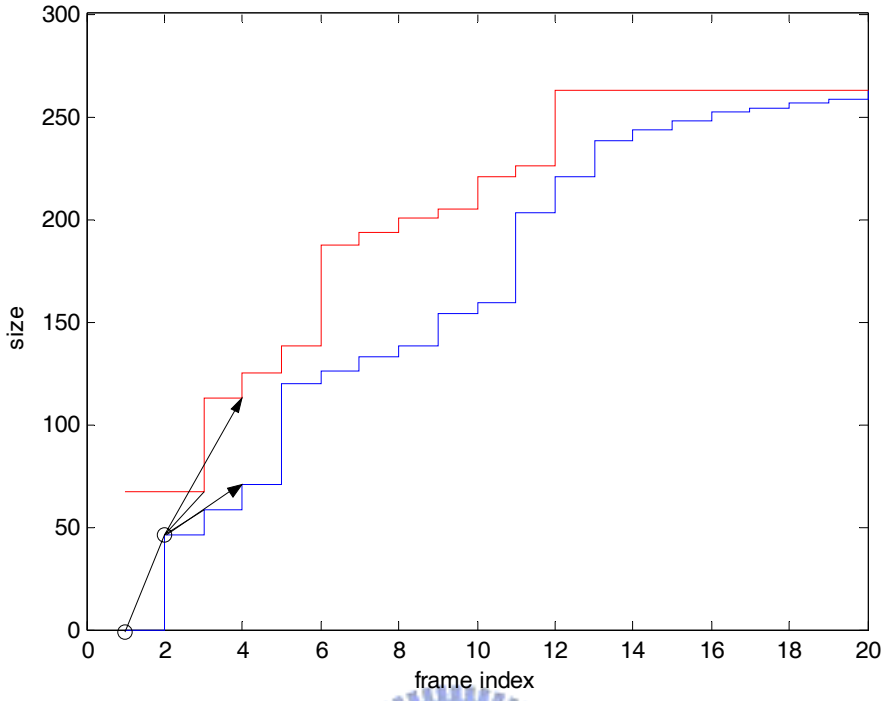


圖 2.9(g)

8、不在 case1、case2 的情況下，cmax’和 cmin’和原本的 cmax 和 cmin 比較可得新的 cmax、cmin。假如 cmax’小於等於 cmax 的話，就取 cmax’為 cmax；cmin’大於等於 cmin 的話，就取 cmin’為 cmin，否則的話，維持不變。得到時刻到這個時刻可行的 cmax、cmin，如圖 2.9(h)。這個步驟即是在決定這個區間內可行的最大、最小速率。

9、由相同的起始點延伸到再下一個時刻，同上的方式，可以得到 cmax’和 cmin’，如圖 2.9(i)。

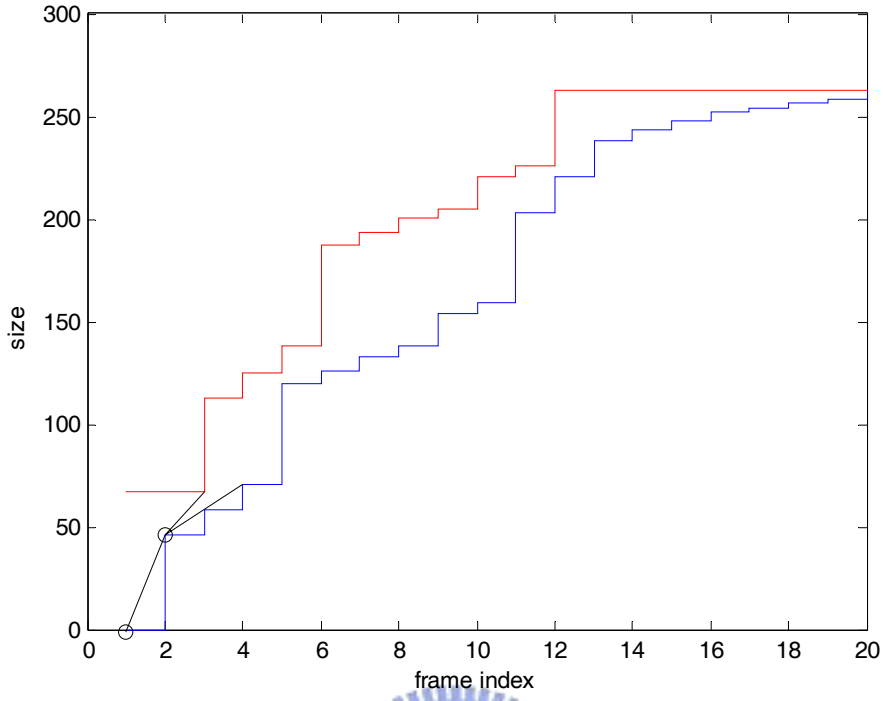

size
frame index
0
50
100
150
200
250
300
0
2
4
6
8
10
12
14
16
18
20

圖 2.9(h)

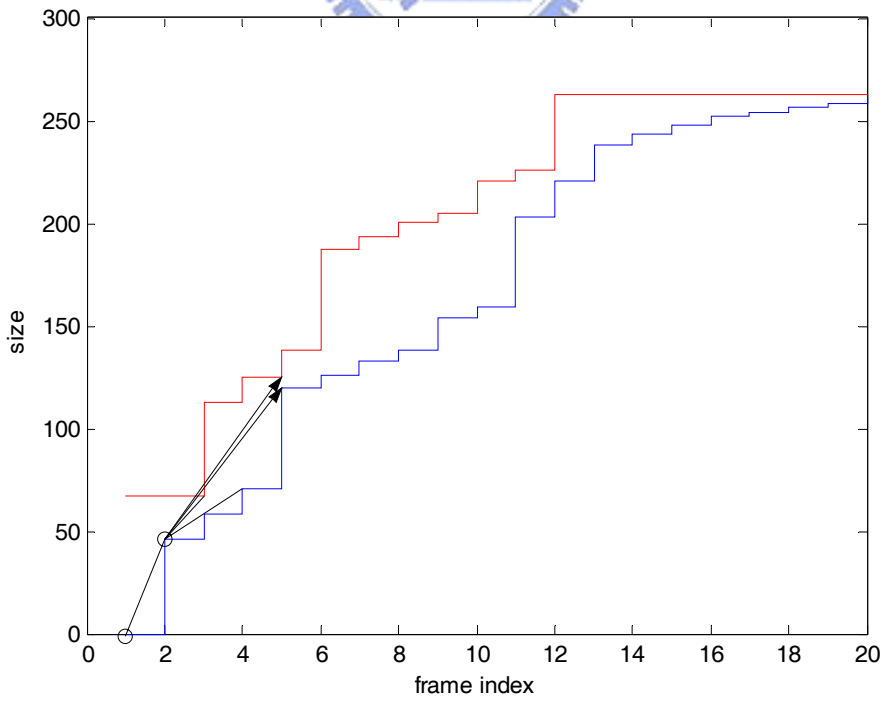

size
frame index
0
50
100
150
200
250
300
0
2
4
6
8
10
12
14
16
18
20

圖 2.9(i)

10、比較 cmax’、cmin’與 cmax、cmin，如圖 2.9(j)。

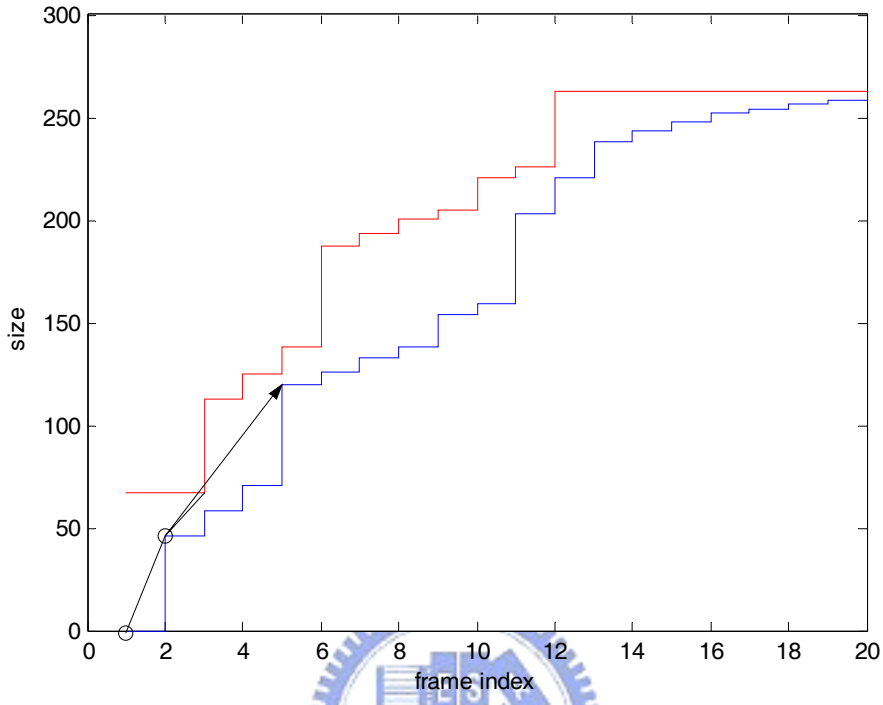


圖 2.9(j)

11、因為 cmax 小於 cmin’，為 case 1 的情況，這意味著如果繼續用這個時刻最大的速率去傳到下個時刻還是會 underflow，所以有必要作速率的改變。可是這個區段有很多選擇，要選擇哪一個速率呢？因為下個時刻速率變快，而我們的目標是使變動率最小，同 case1 的選取原則，又這個時刻的速率最接近下一個時刻的速率的線段就是 cmax，所以我們便選擇 cmax 為這個區段的速率。圖 2.9(k)。

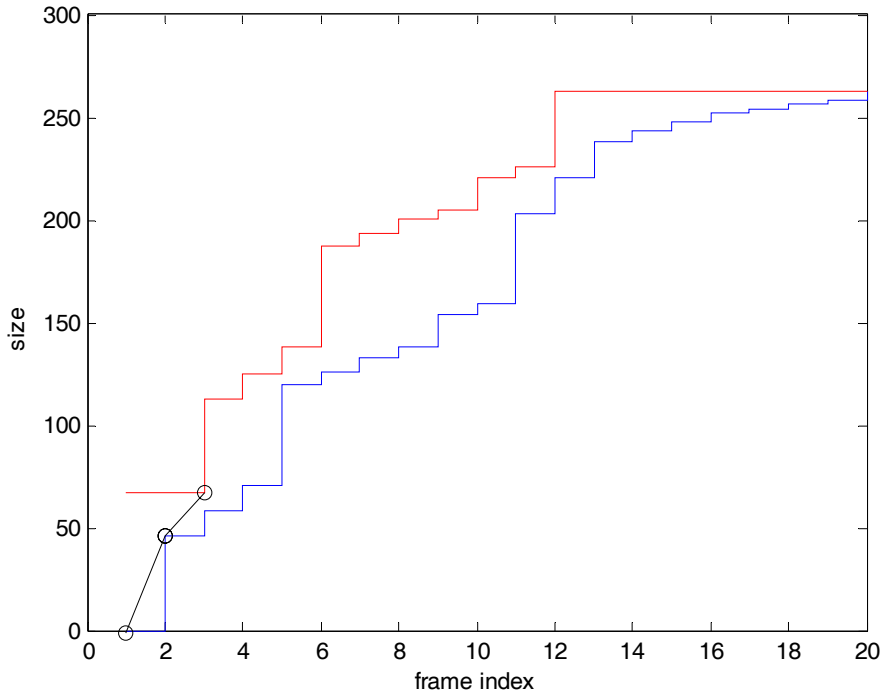


圖 2.9(k)

12、重覆相同的步驟，最後就可以得到我們要的結果。最後一個 run 因為 cmax 和 cmin 相同，所以我們就選擇 cmin。如此就可以得到完整的平順排程，如圖 2.9(l)。

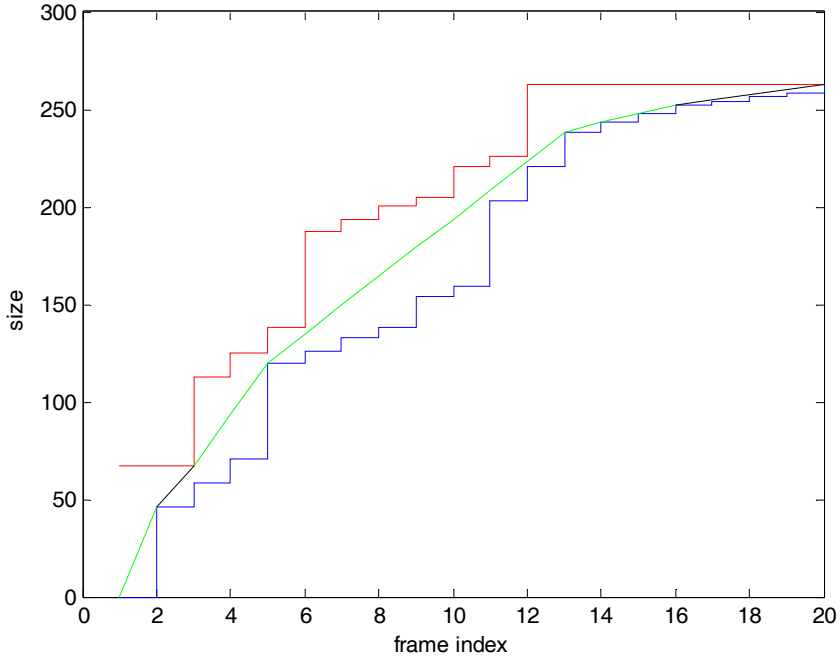


圖 2.9(l)

針對 underflow 和 overflow 的情況，以下再作詳述：

1、case 1 的情況 (再下去會發生 underflow 的情況):cmin'×(t+d)指的是下個時間至少所需累積的下限資料量。cmax×(t+d)指的是繼續用這樣速率所能累積的資料量。因為 cmin'×(t+d)>cmax×(t+d)，所以繼續用 cmax 這樣速率傳輸是不可行的，資料量不夠，會造成 underflow，所以在 t 點時就要作速率增加，才有辦法滿足其需求。(d 指的是時刻差)

2、case 2 的情況 (再下去會發生 overflow 的情況): 同理 cmin×(t+d) > cmax'×(t+d)表示用最小速率傳都會 overflow 了，所以在 t 時間就必需作速率的減少，才不會在 t+d 時間點造成 overflow。

這個演算的基本原則就是儘量延長 constant bit rate 區段的長度，直到說碰到 case 1、case 2 種情形不得作改變才改變，改變點也就是速轉折點，我們就稱為 critical point。所謂不得不作改變的情況就是以同樣的速度再傳下去會造成 overflow 或是 underflow 的情形。在會造成 overflow 的情況，我們就需要讓速度下降，同理，在會造成 underflow 的情況下，我們就要讓速度上升。當時如果是會造成 overflow，就選擇 cmin，因為這樣的速率差異才比較小；當時如果是 underflow 的情況，就選擇 cmax，同樣也是希望跟之後速率的差異可以比較小。

簡言之，這個演算法基本就是在 constraint 下，利用 cmax 和 cmin 與 cmax'和 cmin'來當作判斷的準則。線段速率的決定在兩個情形，一是 cmin>cmax'，一是 cmax<cmin'，一個選取 cmin，一個選取 cmax，選取的準則就是選則讓 rate 變化最小的那一個。

這個演算法的特性，經我們的觀察可作下面陳述：一、區段延伸愈長愈好，直到遇到需改變的地方，如 underflow 或 overflow 才作轉折，決定區段。二、遇到需改變的地方，才決定原本的區段，區段選擇是以速率差異最小為原則，若是遇到 overflow，也就是需要速率下降的時候，我們會選擇 cmin；遇到 underflow，也就是需要速率上升的時候，我們會選 cmax，因為這樣才會使前後的速率差異最小。三、區段轉折點都在邊緣，雖然區段愈長愈好，可是區段端點需在邊緣，

如果較長但端點不在邊緣，在這樣的選取原則下並不會留下來。處理的流程一開始決定初始的 cmax、cmin，延伸到下個時間點得到 cmax’和 cmin’，兩者比較，在沒遇到轉折的狀況，判斷準則如演算法說明步驟 8，可行且愈長的會留下，然後再繼續下探，不過有時因下探的的幾個 cmax’或 cmin’可能均不可行，所以 cmax’與 cmin’與原本的 cmax、cmin 可能不一定只差一個時間間隔，像圖 2.9(f)~(k)，決定下來的區段可能已是向後下探測試很多區段才決定出來的結果；之後由決定好的區段的終點再當初始點，中間一些剛剛有測試過的地方，要再重新作測試，所以這會造成運算量的增加。如果我們沒有區段端點需在上下限邊緣的限制，區段可以延伸比較長，所以中間需重覆測試比較短，計算量較少，可是這便不會達成最佳排程。換句話說，我們達到最佳排程的代價就是付出較大的計算量。整個演算法其實可以用文字歸納為幾個步驟，如表 2.2 所示。

2.5 總結

進行最佳平順(optimal smoothing)的流程在此作一個總整理，首先根據緩衝區(buffer)和 playback video trace，依照 2.2-1、2.2-2 兩式子建立上下限。接著在上下限的限制之下，利用演算法找到最平順的

一個結果。最佳平順演算法指的是在 rate variance 下的最佳化，也就是這個排程的標準差必會小於等於其他排程的標準差。

這個演算法主要是針對 stored video 作探討，對於時況(live)影像傳播和倒轉、快轉的問題並沒包含。針對此，有一篇是針對此篇再作延伸至 on-line smoothing 的問題[3]，另一篇是針對 buffer size 不定，並且考慮到倒轉、快轉問題的提出的快速演算法[5]。

1、由起始點決定一開始的 cmax 與 cmin。
2、從相同的起始點，延伸到下一個時刻，在這個間隔內得到 cmax’與 cmin’。
3、比較 1、2 得到的值。
(1)假如 cmin 大於 cmax’ 為 case2 的情況，決定區段，
此區段速率為 cmin。由這區段的結束當起始，回到 1。
(2)假如 cmax 小於 cmin’，為 case 1 的情況，決定區段，
此區段速率為 cmax。由這區段的結束當起始，回到 1。
(3)其他情況，不決定區段，而是回到 1。
（決定目前 cmax 和 cmin 的值：cmax’和 cmin’和
原本的 cmax 和 cmin 比較可得新的 cmax、cmin。
假如 cmax’小於等於 cmax 的話，就取
cmax’為 cmax；cmin’大於等於 cmin 的話，就取 cmin’為
cmin，否則的話，維持不變。）

表 2.2 演算法文字說明步驟

【註 2】此情況為 startup 為 1 個 frame time 的情況。
【註 3】為了說明方便，刻意將 constraint 選成會發生所有情況都的的例子。

第三章

最佳平順演算法的模擬與討論

經由上一章的介紹，對於最佳平順演算法的基本原理、方法應該有些了解了。本章就進一步來分析最佳平順演算法實際帶來的效果有多大。我們首先會針對單一實際的 video trace，作模擬與分析。但是如果只針對一個視訊，普遍性不足，所以我們再會針對幾個不一樣的 video trace 來作相同的分析，看其效益是否相同；依據這些結果，我們可以看其統計特性的關係 e.g. 緩衝區大小和標準差、峰值的關係。

通常我們在評量一個平順演算法的好壞，通常是利用它的標準差(variance)和峰值(peak rate)的值來作評估。標準差愈小，代表整個排程的變異度愈小，平順度愈大；標準差愈大，代表整個排程的變異度愈大，平順度愈小。峰值愈小，代表所需預留的最大頻寬愈小。我們以下主要探討的是標準差與 標值和緩衝區大小的關係。除此，我們有還探討了另一個特性，也就是速率變動次數和緩衝區大小的關係。

3.1 最佳平順演算法針對 video trace 的模擬結果

測試 video 為經 MPEG4 壓縮產生的 video trace file，測試檔來源於[11]，檔名 terminator1。它的統計資料為 mean frame size=325.6381 byte，peak frame size=3455 byte，variance = 274.2755，video trace 如圖 3.1 所示。可以看出其變異程度相當大。

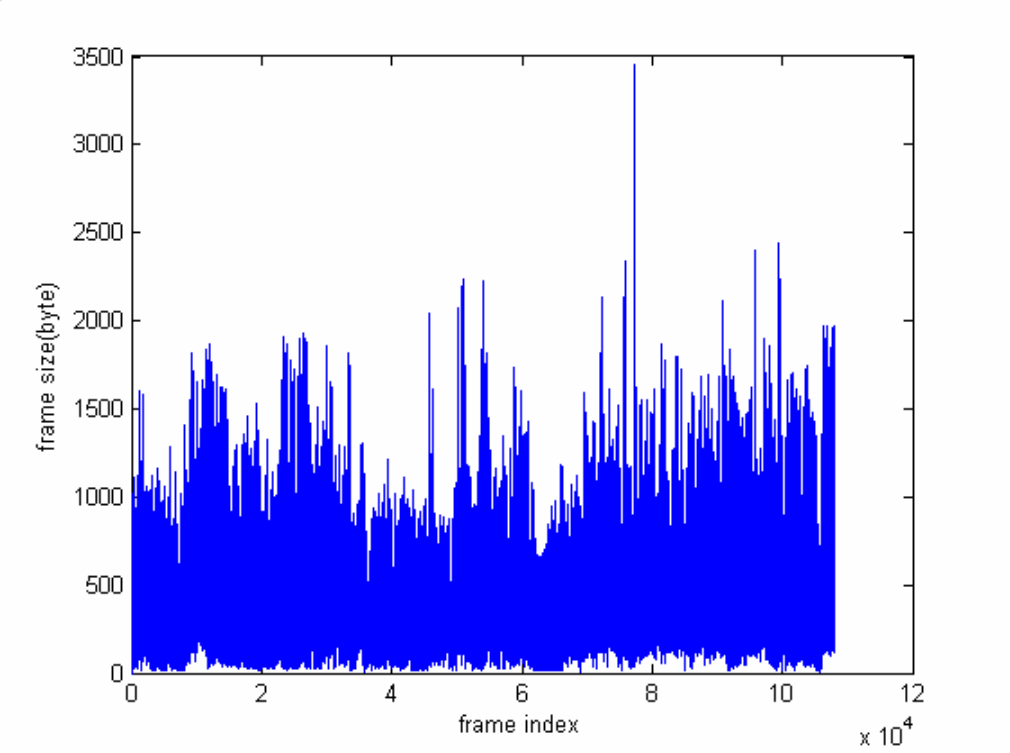


圖 3.1(a)

以下分別就不同緩衝區大小，來看其結果：

1、以 startup delay =0.5s, buffer=8KB 情況去作 smoothing，8K 相當是約是原本 mean frame size 的 25 倍，大約可以一次裝 25 個 frame，

容許延遲大概是 0.8sec。以這樣的緩衝區去作模擬，得到結果如圖 3.1(b)。其 peak frame size 變為 1210.1byte, variance 變為 118.4318。

2、以 startup delay =0.5s, buffer=64KB 情況去作 smoothing，64K 相當是約是原本 mean frame size 的 202 倍，大約可以一次裝 202 個 frame，容許延遲大概是 6.7sec。以這樣的緩衝區去跑得到結果如圖 3.1(c)。其 peak frame size 變為 690.5363byte, variance 變為 91.0084。

3、以 startup delay =0.5s, buffer=128KB 情況去作 smoothing，128K 相當是約是原本 mean frame size 的 402 倍，大約可以一次裝 403 個 frame，容許延遲大概是 13.4sec。得到結果如圖 3.1(d)。其 peak frame size 變為 568.4664byte, variance 變為 79.1860。

4、以 startup delay =0.5s, buffer=1024KB 情況去作 smoothing，1024K 相當是約是原本 mean frame size 的 3226 倍，大約可以一次裝 3226 個 frame，容許延遲大概是 1min 48sec。以這樣緩衝區去跑，得到結果如圖 3.1(e)。peak frame size 變為 375.7135byte, variance 變為 34.4548。

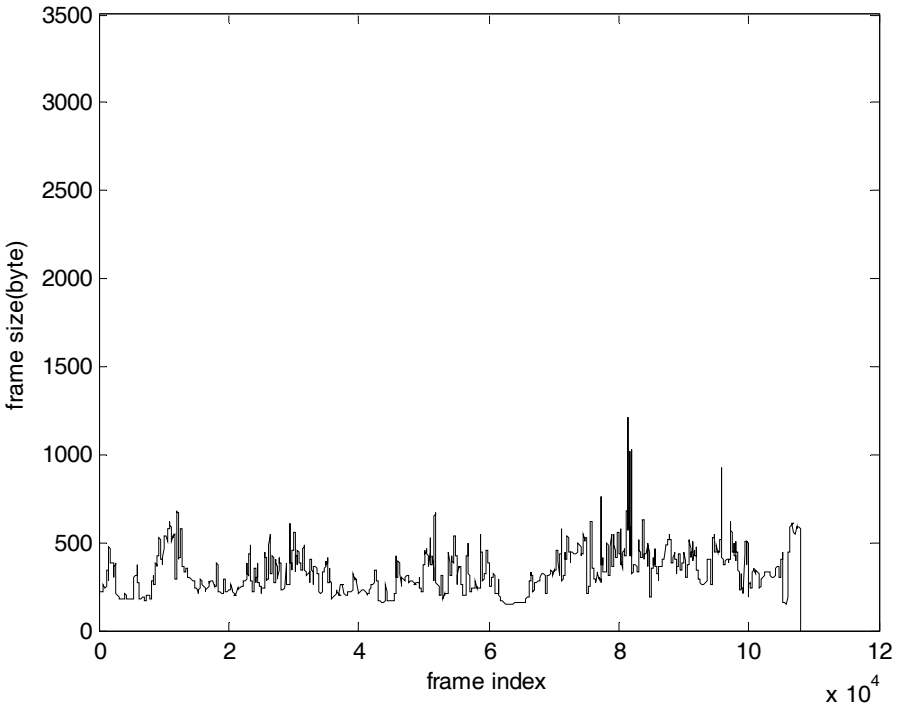

3500
3000
2500
2000
1500
1000
500
0
frame size(byte)
0
2
4
6
8
10
12
frame index
x 10^4

圖 3.1(b)

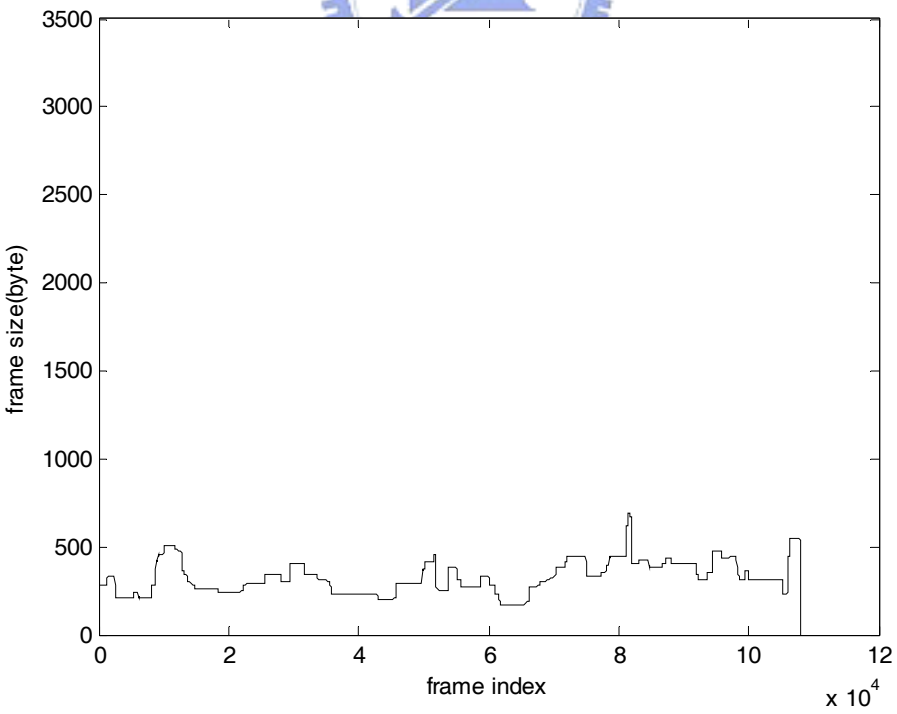

3500
3000
2500
2000
1500
1000
500
0
frame size(byte)
0
2
4
6
8
10
12
frame index
x 10^4

圖 3.1(c)

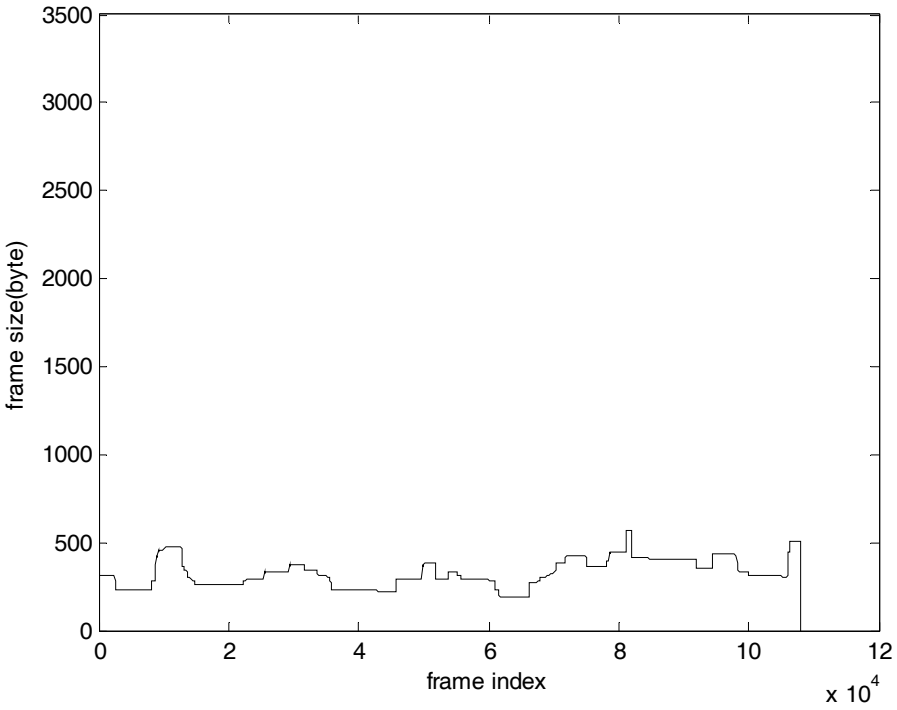

3500
3000
2500
2000
1500
1000
500
0
frame size(byte)
0
2
4
6
8
10
12
frame index
x 10^4

圖 3.1(d)

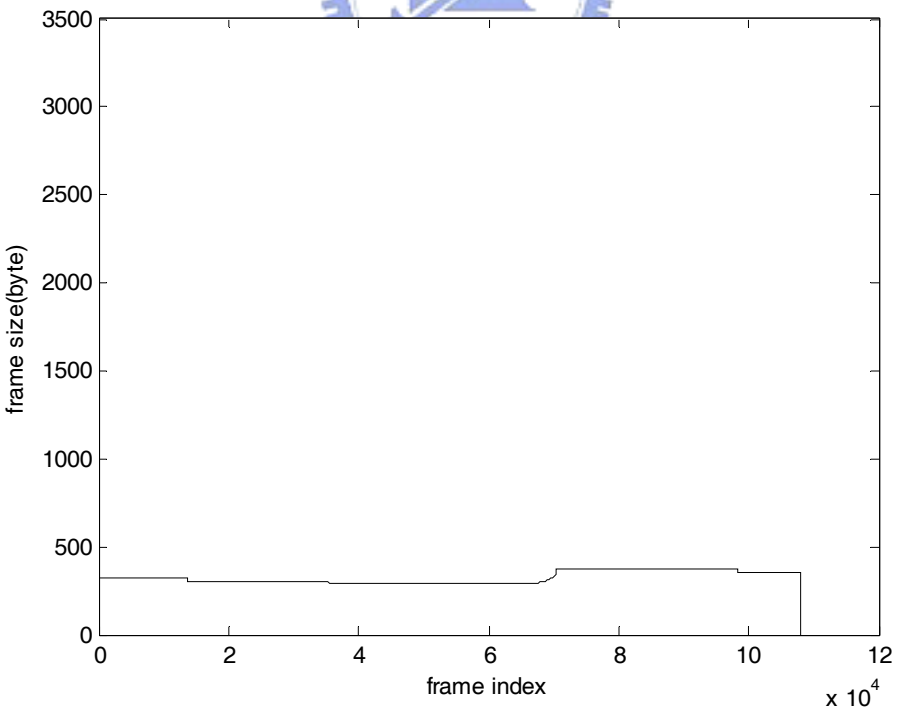

3500
3000
2500
2000
1500
1000
500
0
frame size(byte)
0
2
4
6
8
10
12
frame index
x 10^4

圖 3.1(e)

以眼睛直觀的來看，也可以明顯看出平順程度有大大的增加，由數據來看標準差(variance)也確實隨緩衝區大小的增加而減小，峰值(peak rate)也是有隨緩衝區大小的增大而減小的趨勢(peak rate = peak frame size×30)。不過光是數據分別看，可能還看不清其差異度有多大，整合在一起關係圖可能更看的出其差別：標準差和緩衝區大小間、峰值和緩衝區大小間關係圖如圖 3.2(a)、3.2(b)所示。

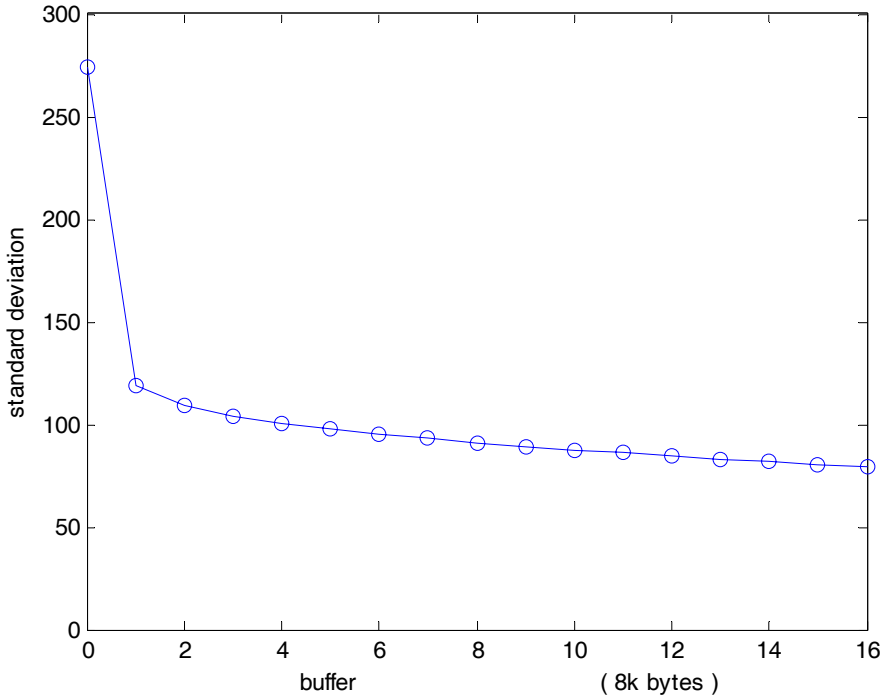


圖 3.2(a) standard deviation versus buffer size

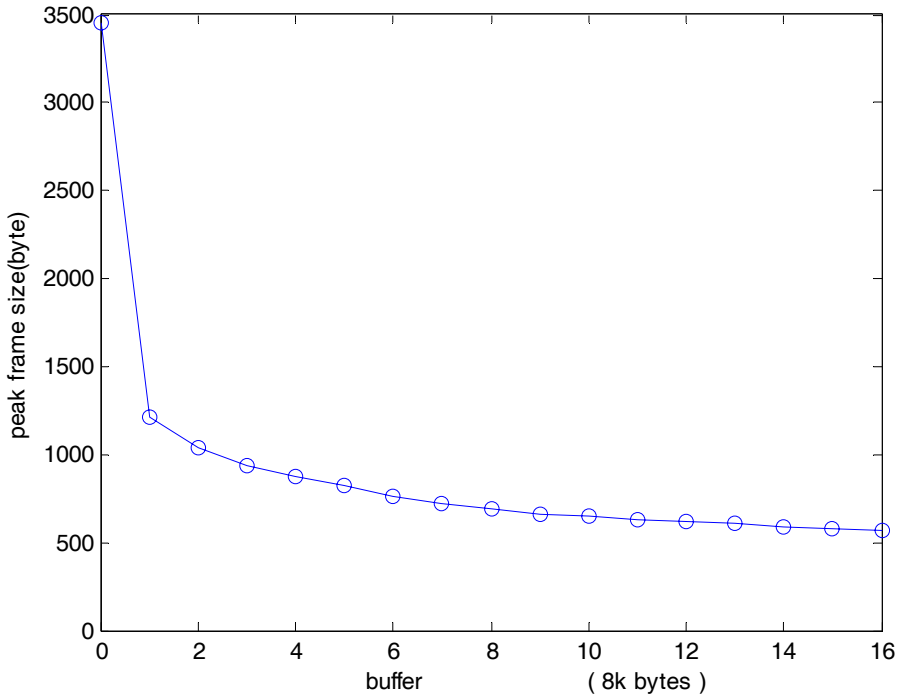


圖 3.2(b) peak frame size versus buffer size

光是看數值關係圖可能還看不出效益的差異程度，所以我們以下用 normalized 過的值再來看看。所用的 normalize 的方式，就是將每個經平順產生的結果值除以原本未平順的值。圖如 3.2(c)3.2(d)所示。

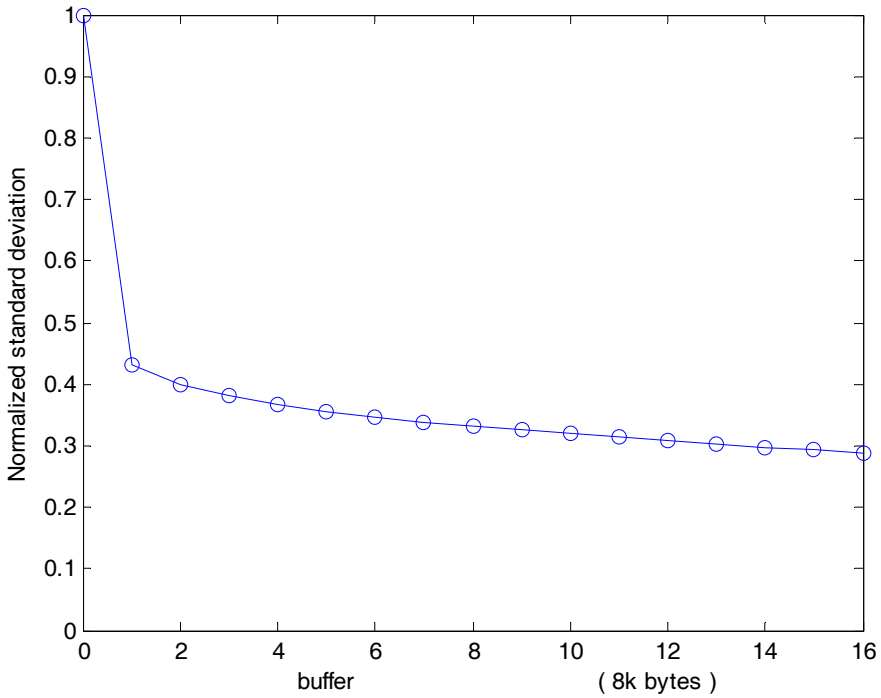


圖 3.2(c) Normalized standard deviation versus buffer size

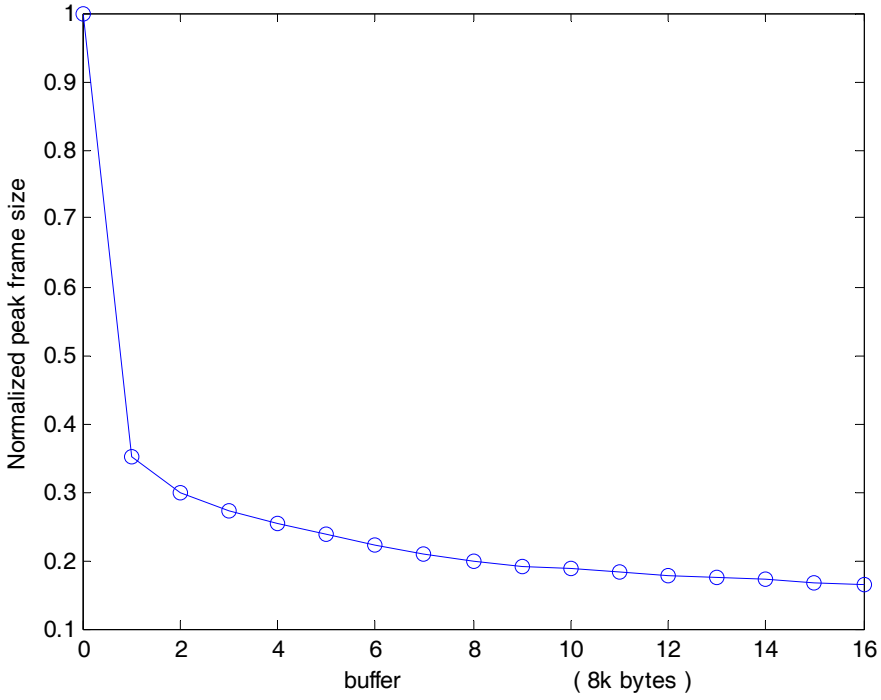


圖 3.2(d) Normalized peak frame size versus buffer size

由圖 3.2(c)看出只用了 16KB 的緩衝區大小就可以將標準差降低 67%；由圖 3.2(d)看出只用了 16KB 的緩衝區大小就可以將峰值降低 80%。效益其實相當大，用一點點的緩衝區(16KB 相當可以存這樣的 video trace 50 張 frame)就可以使得原本的值降到很低。

由於以上緩衝區大小只測到 128kbyte，我們將它延深至 1024k byte 可能更看的出全面的效果。如圖 3.3 所示。

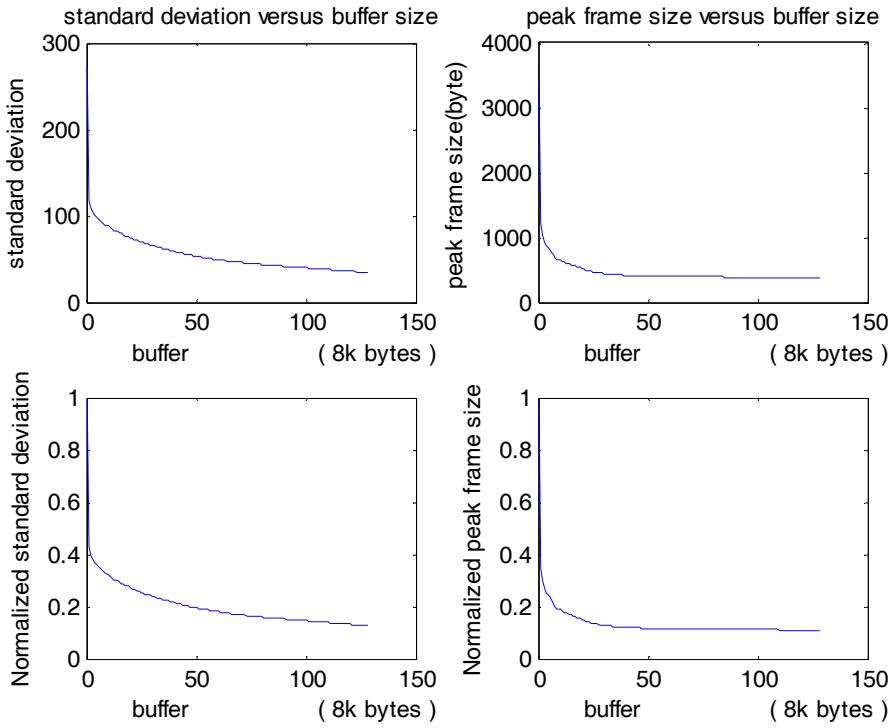


圖 3.3 variance and peak rate versus buffer size(0~1024KB)

因為個演算法本來就是在標準差上作上的最佳化，每一段都儘量延伸，當緩衝區愈大的時候，可以選擇的區間愈大，可以選到一個延伸更遠不轉折的線段的可能性也愈大，而且它在轉折的選擇上也是選取使速率改變最小的線段，所以標準差會隨著緩衝區增大而漸減。不過對峰值而言，光是可以延伸愈遠不變，並不表示 rate 就一定比較小，不過因為緩衝區愈大，轉折愈少，需要一下子 rate 爬升的機會也會減少，rate 變小的機率也比較大，但這似乎不也意味峰值也必然因此減少。有關峰值隨緩衝區變大而變小的原因，詳細說明如下。

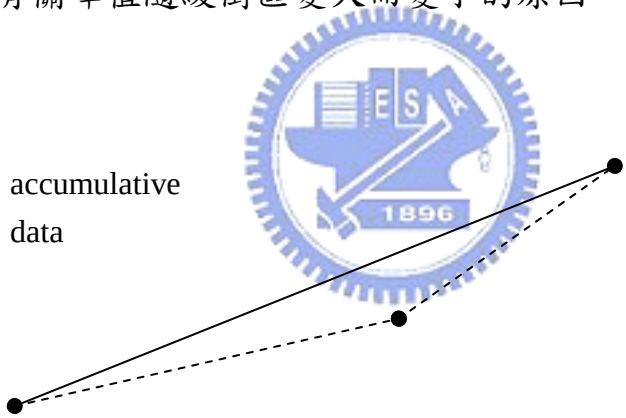


圖 3.4(a)

[5]中有證明 buffer size 大的轉折點是 buffer size 小的轉折點的子集，所以我們可以作以下的假設。圖 3.4(a)虛線部分為原本排程的某兩兩個區段，如果我們將 buffer size 加大到轉折點不見變一個區段。如實線所示，明顯可以看出後來的區段 peak rate 較小。因為原本的

區段雖有兩個，但是一個斜率雖小，另一個卻大，那個大的就已經超過原本的 peak rate 了。

如果兩個區段不準，我們看更多區段的情況，圖 3.4(b)如果原本 buffer size 小的時候 schedule 如下圖的虛線，buffer size 大之後的是實線，明顯也看出原本的 peak rate 較大。雖然最後的累積量是一樣，可是平緩上升的方式，peak rate 會比較小。想要全部都用比較小的 rate 要累積相同的量根本就不可能，只要一段變小一點，另一段就要變大一點，那個變大的部分就會超過平順之後的 peak rate。(這次重疊的轉折點有三點)

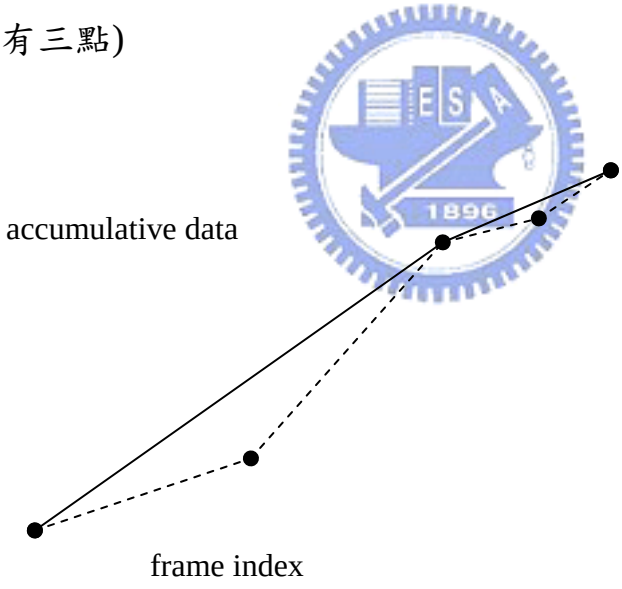


圖 3.4(b)

圖 3.4(c)是每單位時間傳送的資料量，不是累積的資料量，虛線是 buffer size 小的時候最佳平順的結果，實線是 buffer size 大的時候最佳平順的結果。雖然累積的量相同，不過原本 buffer size 小的時候

peak rate 比較大，由這樣的示意圖也可以看出。

所以 buffer size 愈大，peak rate 愈小也是正確的結果。只是 buffer size 大到一個程度再加大，peak rate 改變的幅度會愈來愈小，後來會趨近理想值，也就是整個 video trace 的 mean rate。

以簡單的數學式子也可以說明：假設有四個區段 $t1$、$t2$、$t3$、$t4$ 速率不同分別是 $r1$、$r2$、$r3$、$r4$ 我們藉由增大 buffer size，大到讓原本的轉折點消失，得到一個新的區段，其值速率為 r。因為用原本四區段和後來單區段增加的累積資料量相同。所以 r 為四個區段 $t1$、$t2$、$t3$、$t4$ 的 weighted 平均速率，我們可以到下面的式子：

$$r = (r1 \times t1 + r2 \times t2 + r3 \times t3 + r4 \times t4) / (t1 + t2 + t3 + t4)$$
$$r \times (t1 + t2 + t3 + t4) = r1 \times t1 + r2 \times t2 + r3 \times t3 + r4 \times t4$$
$$t1 \times (r - r1) + t2 \times (r - r2) + t3 \times (r - r3) + t4 \times (r - r4) = 0$$

$r1$、$r2$、$r3$、$r4$ 同時有大於、小於或是同時有大於、小於、等於 r 的狀況上式才成立。所以可以推得：

$$r < \max(r1, r2, r3, r4) \qquad (3.1\text{-}1)$$

($r = r1 = r2 = r3 = r4$ 也會使上式成立，不過 $r = r1 = r2 = r3 = r4$ 代表是同一區段，它跟原本的假設不符)

每個部分的 peak rate 都變小了，整體 peak rate 當然也變小。

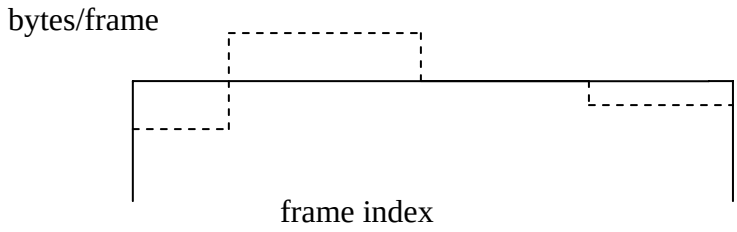


圖 3.4(c)

如果我們在某一時間隔想讓那時間間隔裡的峰值變小，我們就可以調大緩衝區，大到讓一些轉折點消失，那部分的排程變會比較平順。如果可以將彈性調整緩衝區的機制加到排程內，便可以讓整個排程的可適性更大，資源利用也更有效率。利用這個理論，同理也可以推出整體速率標準差會隨緩衝區愈大而愈來愈小。

3.2 最佳順滑演算法針對其他樣本 video trace 的模擬結果

單對某一個影片檔也許普遍度不夠，所以我們再針對幾個不同的 video trace 作平順，看其是否有相似的結果。每個樣本模擬時均設定 startup delay 為 0.5s。測試檔來源同樣於[11]。

1、針對音樂錄影帶 MTV 檔案，其原本的 peak frame size=7430byte，mean frame size=349.728byte，結果如圖 3.5 所示。

2、針對美式足球賽 football 檔案，其原本的 peak frame size = 2475byte mean frame size=355.762byte，結果圖 3.6 所示。

3、針對卡通 aladdin 檔案，其原本的 peak frame size=3291byte, mean frame size=378.181byte，結果圖 3.7 所示。

4、針對電視訪談節目 oprah 檔案，其原本的 peak frame size = 2370byte，mean frame size=283.792byte，結果圖 3.8 所示。

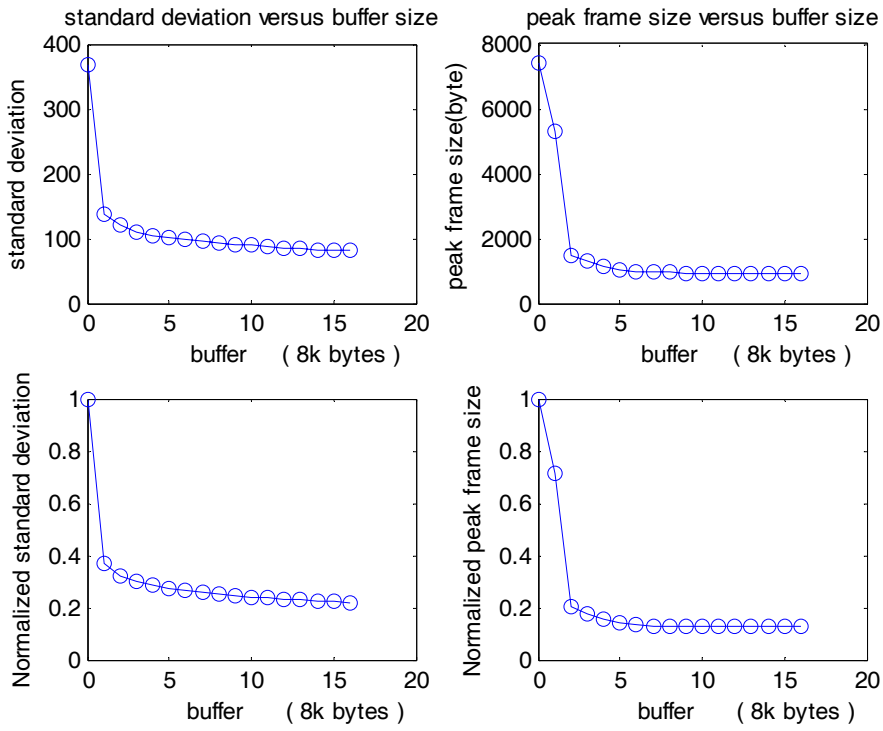


圖 3.5 測試檔為 MTV 的結果

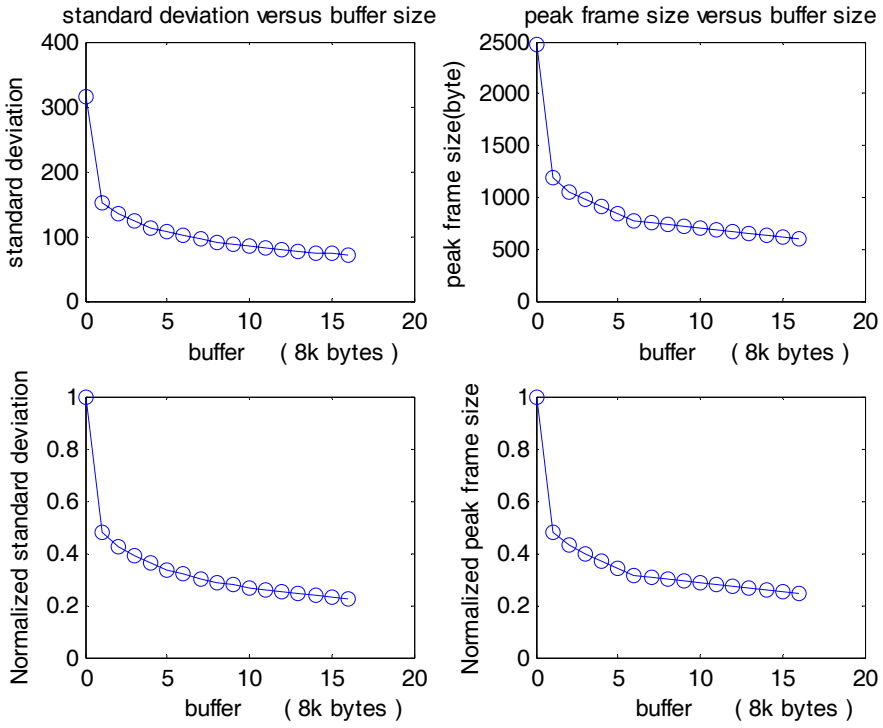


圖 3.6 測試檔為 football 的結果

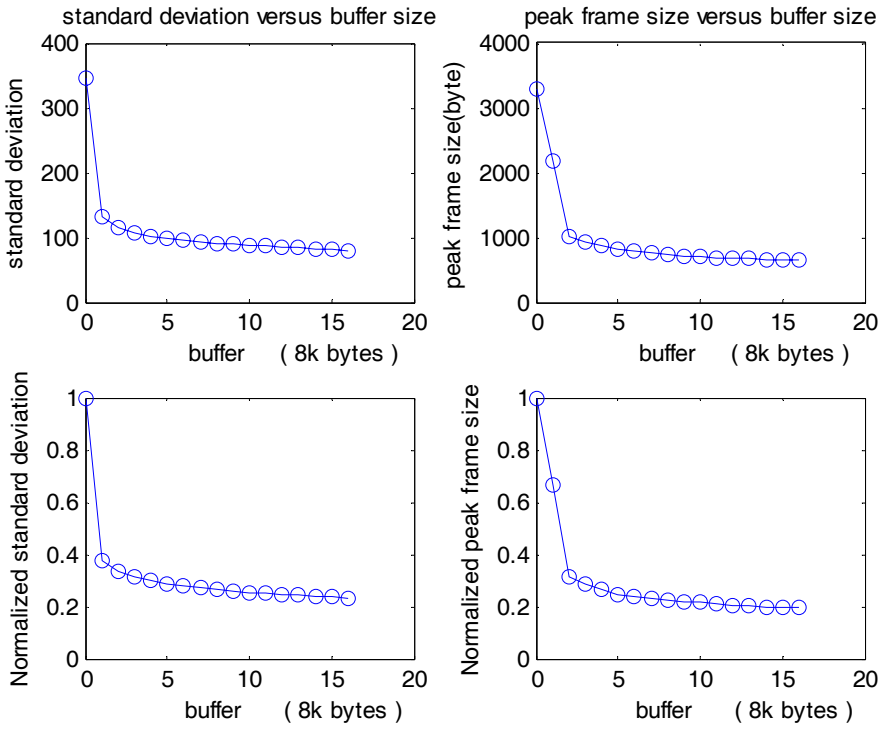


圖 3.7 測試檔為 aladdin 的結果

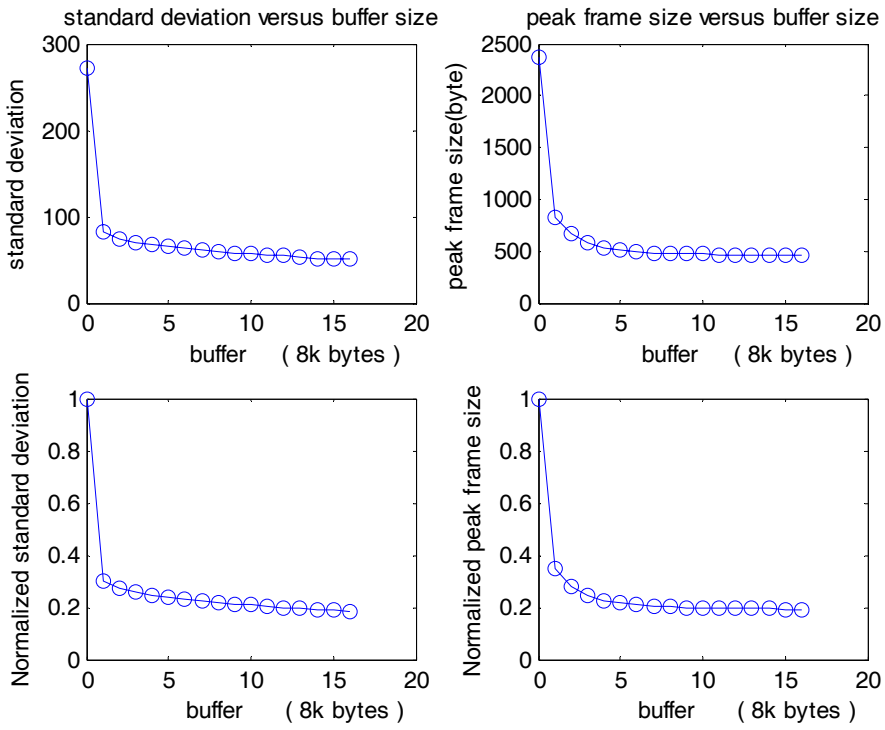


圖 3.8 測試檔為 oprah 的結果

測試樣本含蓋電影、美式足球、訪談節目、卡通、音樂錄影帶等各類型的動態影像檔。經由觀察，雖然因為測試樣本統計統性的不同，造成同樣的緩衝區大小下降幅度的比例有差，不過我們可以看出變動趨勢相當一致，同樣都是相當程度的隨緩衝區大小愈大，標準差、峰值愈小的趨勢。所以最佳平順演算法是有普遍適用性的。

其次，我們觀察峰值與緩衝區大小的關係圖，發現到其實蠻相近於 y=min(1/x+b,1)這個式子(b 為一個 constant offset)。我們設 b=0.1 可得圖 3.9。

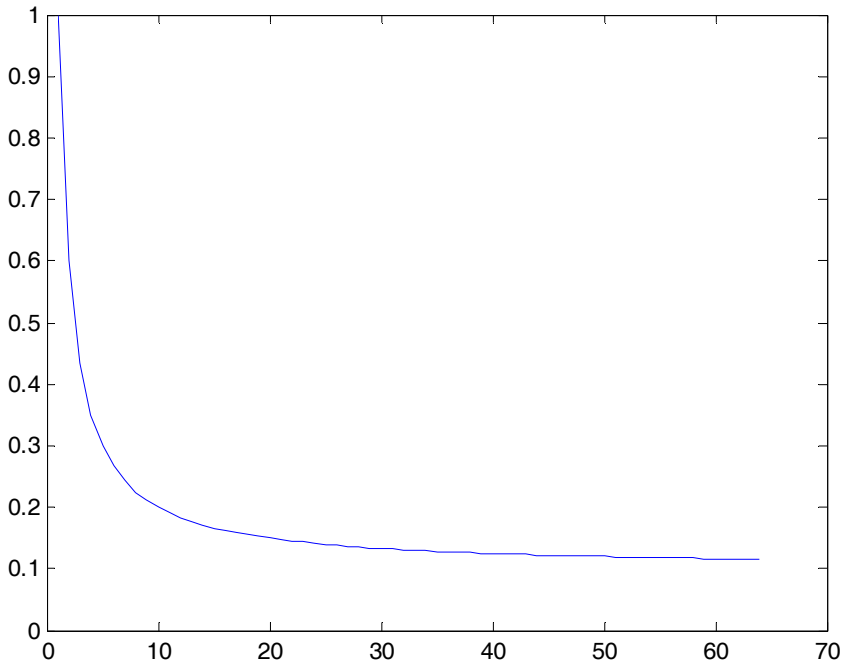


圖 3.9

總之，我們如果如道了峰值與緩衝區大小的關係，這對我們在資源利用上會更有效率。不管是伺服端給用戶端通知，根據頻寬，要客戶端端先預留適宜的緩衝區大小(緩衝區大小未知的情況)或者是緩衝區大小已知，我們就可以約略估出所需的頻寬，好讓網路資源配置的裝置可以更好管理。

3.3 其他特性的比較-速率改變次數

不過光利用 variance 來看它的平順度的衡量標準，並不一定足夠，因為 variance 只能反映出變動幅度的大小，並不能反映出變動的頻率。

平順的排程，除了變動幅度小之外，變動頻率小看起來才真的平順。由平順過後的圖直觀來看是可以看出不僅變動幅度小，變動的頻率也較低，由數值看來也是如此。圖 3.10 是速率改變次數和緩衝區大小的關係。(所使用的測試檔也是 terminator1)

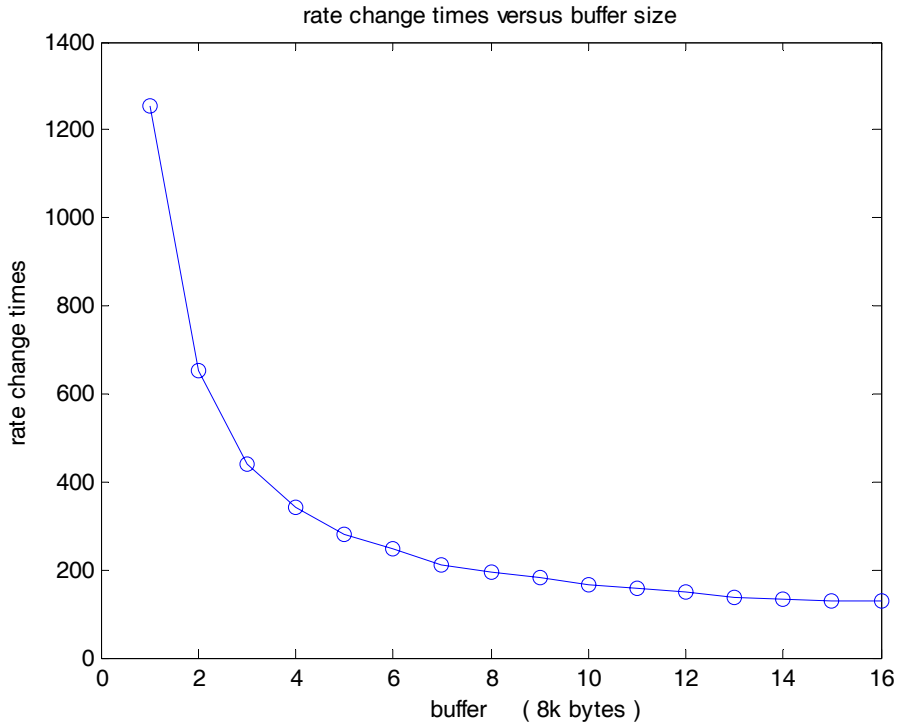


圖 3.10 rate change times versus buffer(不含原本未經平順的部分)

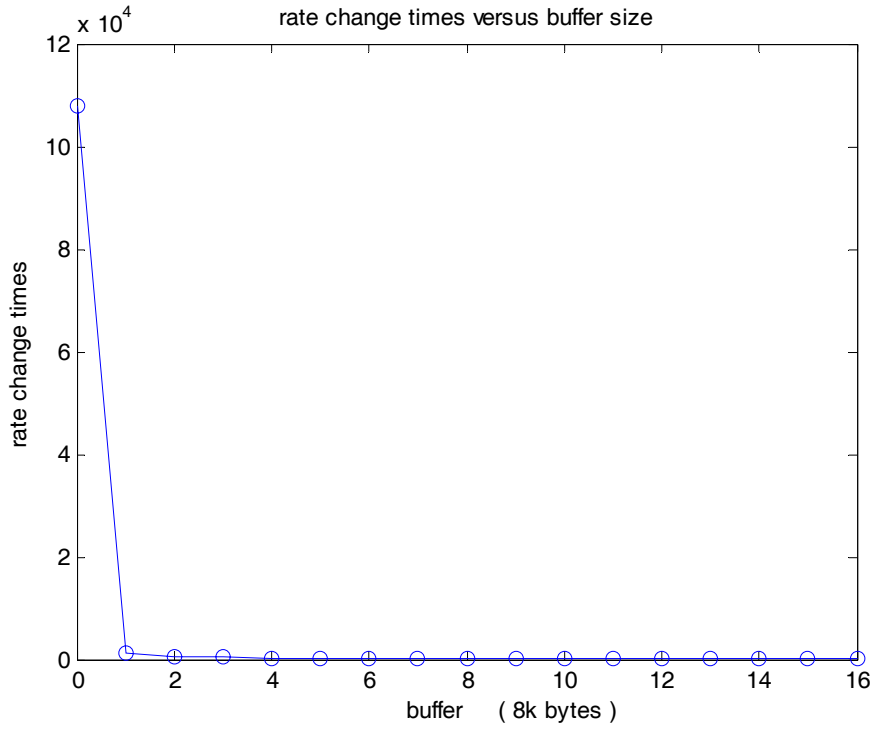


圖 3.11 rate change times versus buffer(含原本未經平順的部分)

我們看一下緩衝區為 16kbyte 的情況，它的速率變動次只有 652，原本的 frame 總數是 108000。這相當於說平均一個區段有 166 個 frame，這表示平均一個區段是維持 5.5sec。

因為緩衝區大小變大，所以可以找到最長不間斷的機率就大，線段就可以儘量延伸不轉折，故 rate change 次數會比較少，但這樣的說法雖然合理，但似乎還少了點證實，我們如果再利用[5]的證明，buffer size 大的轉折點是 buffer size 小的轉折點的子集，很明顯就可以證實這樣的結果。

原本未經平順每個 frame size 幾乎均不同，所以 rate change 次數近似它的 frame 個數，圖 3.11 就是將未經平順的部分考慮進去，不過

因為差異懸殊，不容易看出不同緩衝區大小間的差異程度，所以我們不考慮未經平順的部分再來看，如圖 3.10，就比較可以看出比較細微的差異。

3.4 總結

經由以上的模擬結果，我們可以知道最佳平順演算法對各類型的 VBR 視訊資料均有普遍的適用性。資料的峰值會隨著緩衝區的變大而變小，標準差也是會隨著緩衝區的增大而變小。所以如果想要有比較好的平順的表現，是需要客戶端緩衝區資源的配合才能達成。不過光看標準差的大小有一個缺點，因為標準差的大小只能反映變動幅度的大小，並沒法看出變動頻率的大小，所謂平順應該不只看標準差還要看它的變動次數。比較佳的平順除了標準差變小之外，變動次數也要變小。我們看了所跑的結果，隨著緩衝區大小的增大，速率改變次數也遞減。有了這些關係式後，我們也許就可以利用它來作應用，像是依據峰值和緩衝區大小的關係，我們便可根據我們現有的頻寬限制來決定說客戶端需要預留多少緩衝區量，才能讓讓傳輸可行。

第四章

最佳平順演算法的延伸探討

經過第三章的模擬結果之後，對於最佳平順演算法的效益應該有更進一步的了解了。此章想就此最佳平順演算法可能可以改進的地方或會遇到的問題再作進一步的探討。這就需要再從原本的演算法觀察。

4.1 修正位元組精準度最佳演算法

4.1.1 原本運作方法的觀察

觀察原本演算法的情況，它的轉折點都是在 upper bound 或 lower bound 上，整個演算法是在假設理想狀態下，也就是傳輸速率可以在非整數的狀態下，如此可以達到最佳平順。不過在一般的情況下，每單位時間傳的值是非最小單位整數倍並不是非常合理。所以有必要將此演算法作一下修正，讓它可以運用在一般的環境下。

4.1.2 位元組精準度(byte-precision)

在原本的演算法加上 byte-precision 的機制，也就是每單位時刻

傳送的最小單位是 byte 的整數倍，情況會如何呢？原本的在理想狀態的演算法結果如 4.1(a)所示，加了 byte-precision 的機制為 4.1(b)，byte-precision 的機制是將原本轉折在 lower bound 的區段，若不為整數，作無條件進入；原本轉折在 upper bound 的區段，若不為整數，作無條件捨去。觀察原圖 4.1(a)在轉折點確實就是在 upper bound 和 lower bound，不過在作完 byte-precision 的情況下，轉折點就不再停在 upper bound 或 lower bound 上，反之，可能是 upper bound 下一點的地方，或是 lower bound 上面一點的地方。兩者的輸入皆相同。b=68，原本的 peak 為 49，variance 為 15.3379，d=[46 12 13 49 6 7 5 16 5 44 18 17 5 5 4 2 3 1 5]。

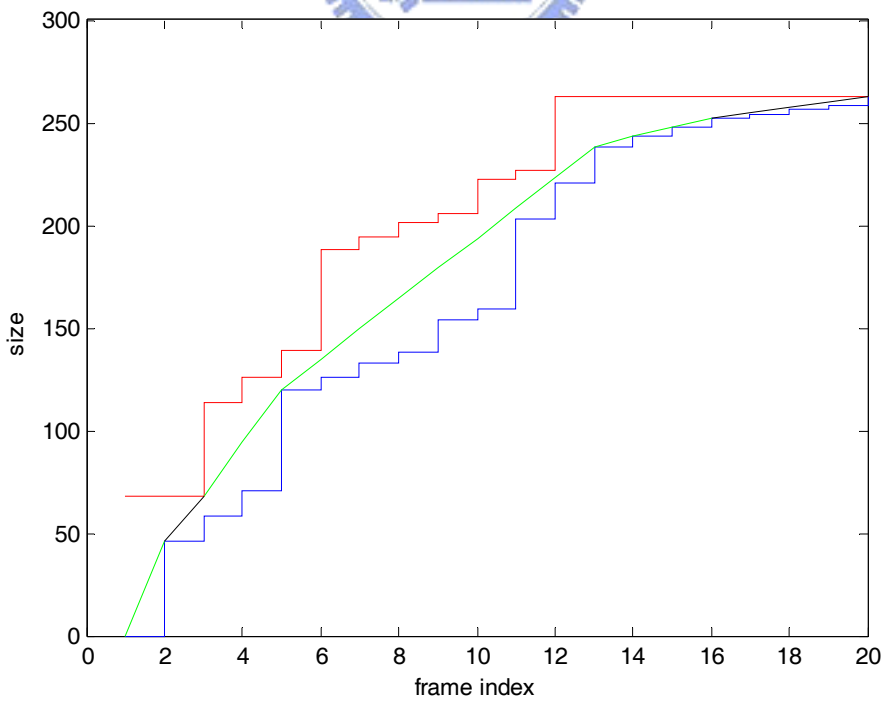


圖 4.1(a) 最佳順滑演算法的結果

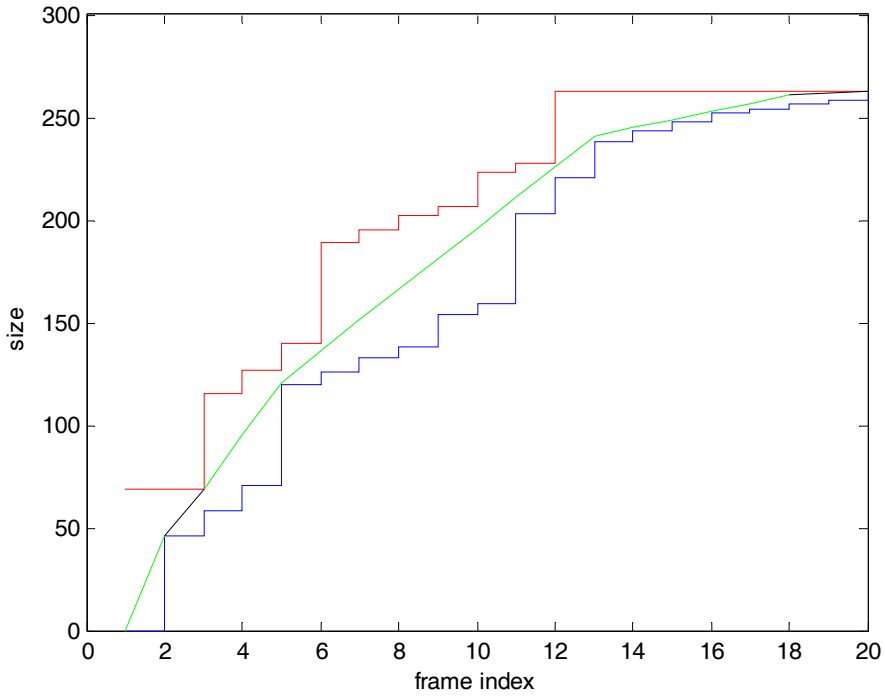


圖 4.1(b)位元組精確度最佳順滑算法的結果【註 4】

兩個圖的結果分別是 4.1(a)的 peak 為 46，variance 為 11.0746；4.1(b)的 peak 為 46，variance 為 11.2870。我們觀察到兩者的 critical points，除了剛開始外，中間有兩點不同原本可行的選擇後的情況變為不可行。比較其數值，peak rate 相同，variance 是 BPOS(Byte-precision Optimal Smoothing)的方式比較大一點。兩個都是在同一限制下的可行排程，不過原本最佳平順的標準差比較小，雖然不能直接證明最佳平順演算法標準差最佳的特性，不過可以作為一個例子。

不過如果光取是在最佳平順演算法加上位元組精準度的方式它可行性不算非常佳，就算是在 upper bound、lower bound 正常的情況，還是有可能會遇到一些問題。以下就將所可能會遇到的問題列出來：

1、cmax 無條件捨去有可能發生小於 lower bound 造成 underflow，所以不可行了。2、cmin 無條件進位造成大於 upper bound 造成 overflow 所以不可行。3、final run 的 rate 取了一個非整數的 rate。

問題一的情況舉例如下，這個區段的原本的傳輸速率 14.7143 非整數，這個總共有 7 個單位時間，一般是無條件捨棄用 14 來傳，不過整個 run 其累積資料 14*7=98 在 lower bound (其值=101)下，如此會產生 underflow 的狀況。

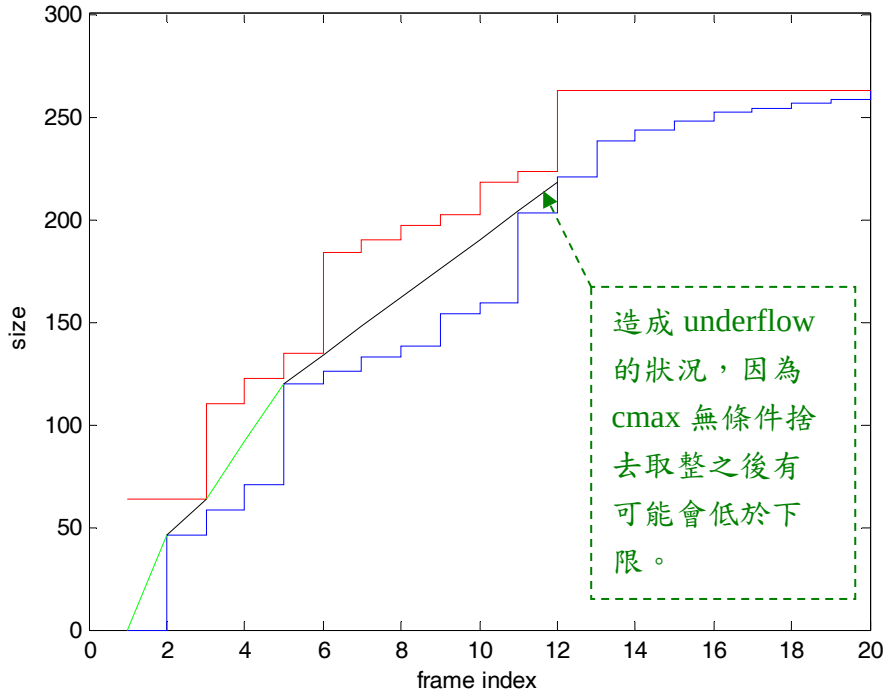


圖 4.2(a) 問題 1 的情形

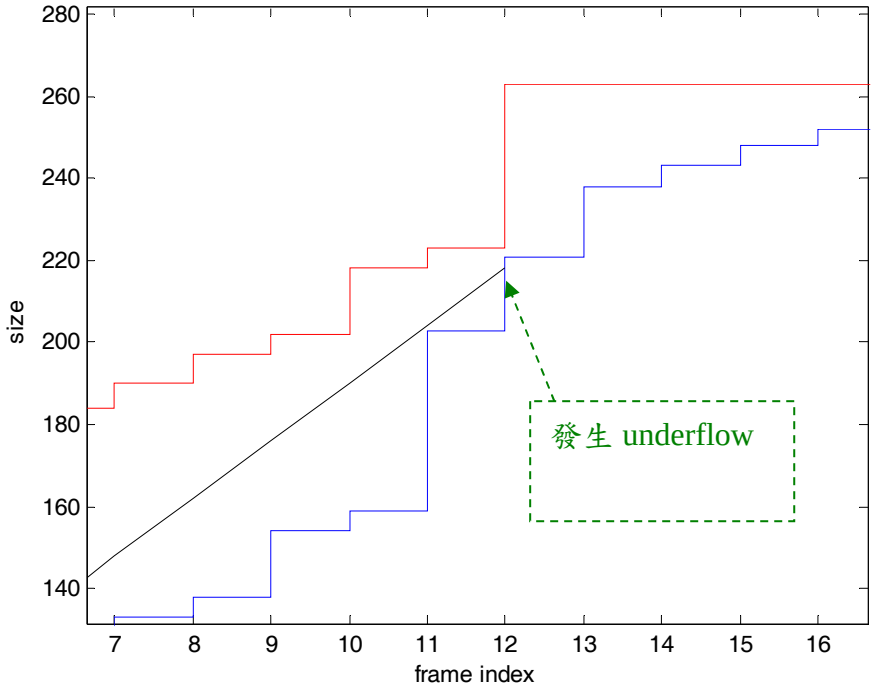


圖 4.2(b) 圖 4.2(a)的放大圖

4.1.3 修正位元組精準度最佳平順演算法(MBPOS)

針對這種情況我們提出一種修正的方式，在遇到這種情況的時候讓這個 run 變成兩個 run 以次數來換取可行性，前半段的 run 還是採用和原本 run 速率相同來傳，而把剩餘的資料留在後半段的 run。(事實上後半 run 的長度只留一個單位時間)。

以圖 4.2(a)發生的問題而為例，總共有 7 個單位時間，我們在前 6 個單位時間還是用 14 的速率來傳，最後一單位時間再 101-14*6= 17 來傳，圖 4.3(a)(b)。如此就至少可以達到那點的最低值，至於為什麼

選 cmin 呢？主要還是希望它的變異可以比較小，因為原本就是速率不夠不情況發生 underflow，cmin 是最接近的值差異最小，所以我們便取 cmin。同理如果是在速率大於 upper bound 的情況，也是用類似的處理方式，只不過它最後一個單位時間的速率是選 cmax，因為原本就是 overflow 的關係，所以選 cmax 是最接近的，這樣變異度最小。在 final run 的情況也是用相似的方法來解決。經過修正演算法跑出來結果如圖 4.3(c)所示。

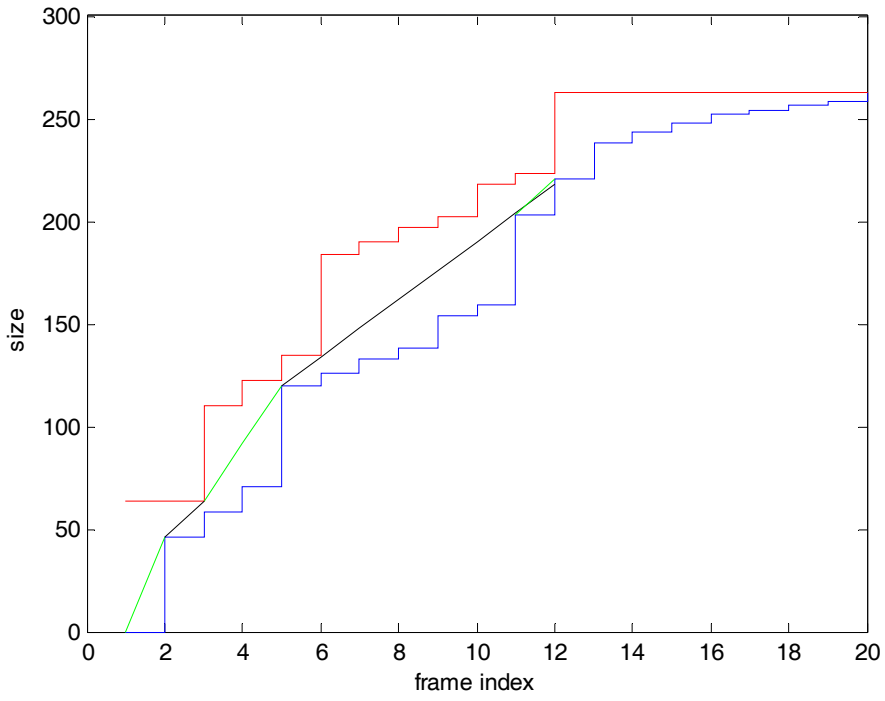


圖 4.3(a) 針對某 run 修正和原本的比較

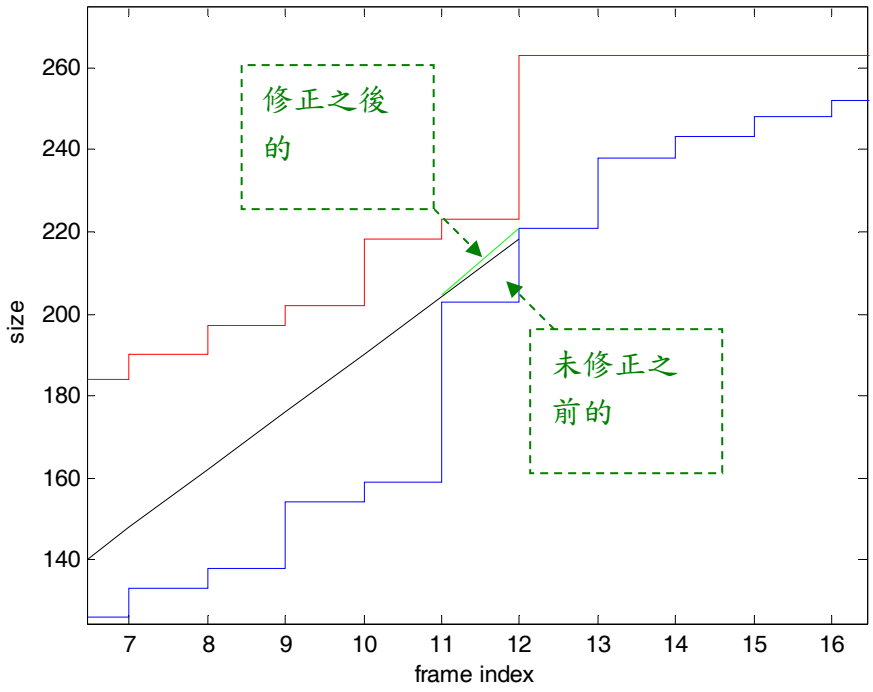


圖 4.3(b) 圖 4.3(a)的放大圖

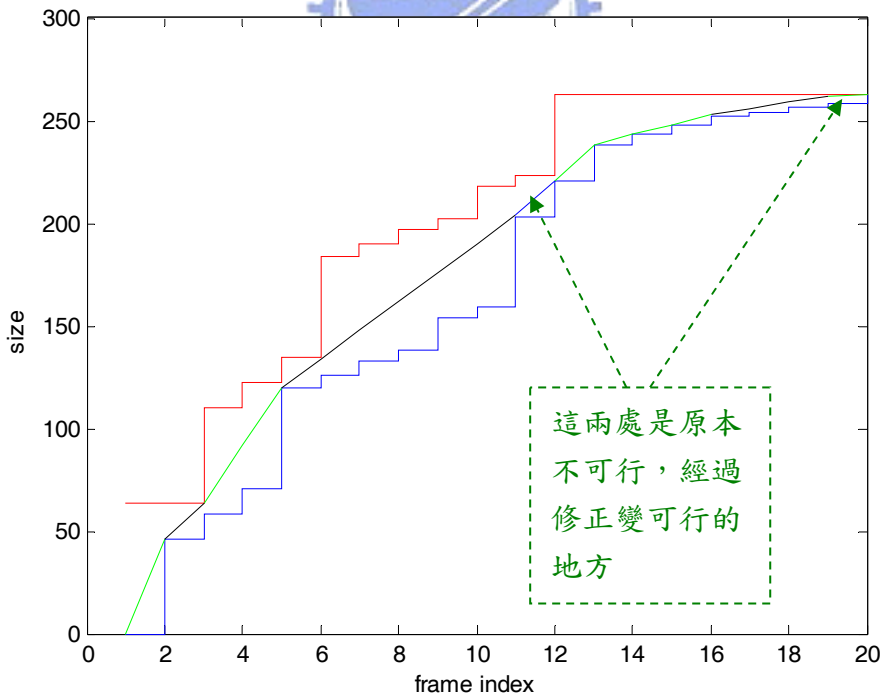


圖 4.3(c) 修正位元精準平順演算法的結果

原本演算法只要是 upper bound 與 lower bound 沒有交錯，就一定可以找到一組最佳的排程。不過在加入 byte-precision 的情況下，如果還是直接用原本最佳平順的方式，有可能會發生不可行的狀況。針對 byte-precision 的情況，原本的演算法需作一修正以達到可行的方法。只追求最佳化但是有可能不可行，如果不可行，就算可以達到最佳情況也沒有用。可行的優先順序還是優於最佳，先求可行，再看有沒有辦法最佳化。

4.2 觀察 MBPOS 演算法標準差、峰值、速率改變次數之於緩衝區大小的關係

使用測試的資料和第三章第一節相同。

1、 當 buffer=8k byte 得到結果 variance=118.61，peak frame size=1211byte，如圖 4.4(a)。

2、當 buffer=64kbyte 得到結果 variance=91.348，peak frame size=691byte，如圖 4.4(b)。

3、當 buffer=128kbyte 得到結果 variance=79.535，peak frame size=569，如圖 4.4(c)。

4、當 buffer=1024kbyte 得到結果 variance=34.832，peak frame size=376，如圖 4.4(d)。

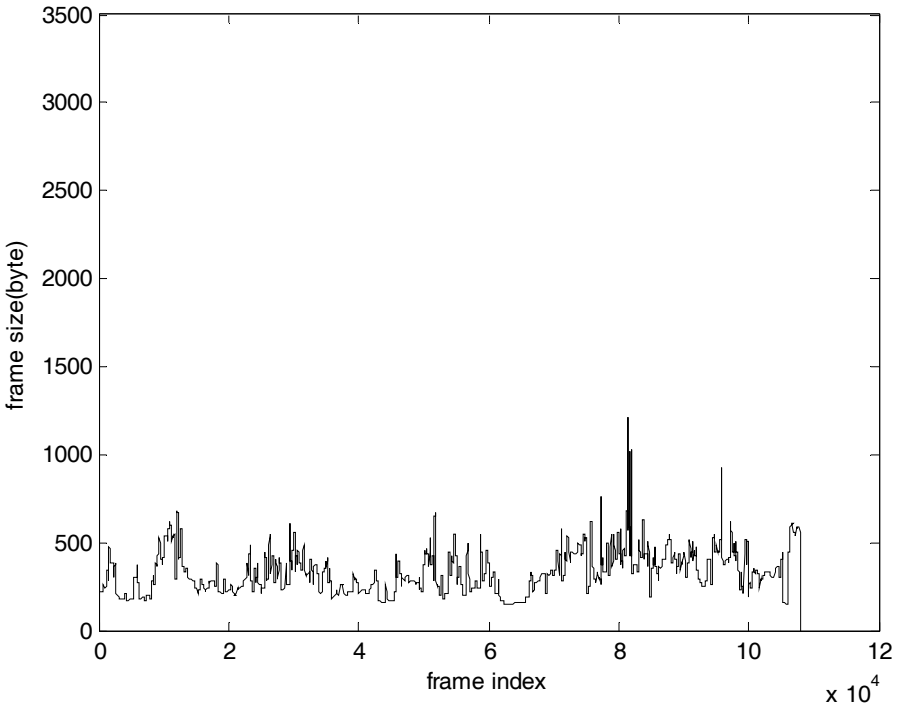

3500
3000
2500
2000
1500
1000
500
0
frame size(byte)
0
2
4
6
8
10
12
frame index
x 10^4

圖 4.4(a)

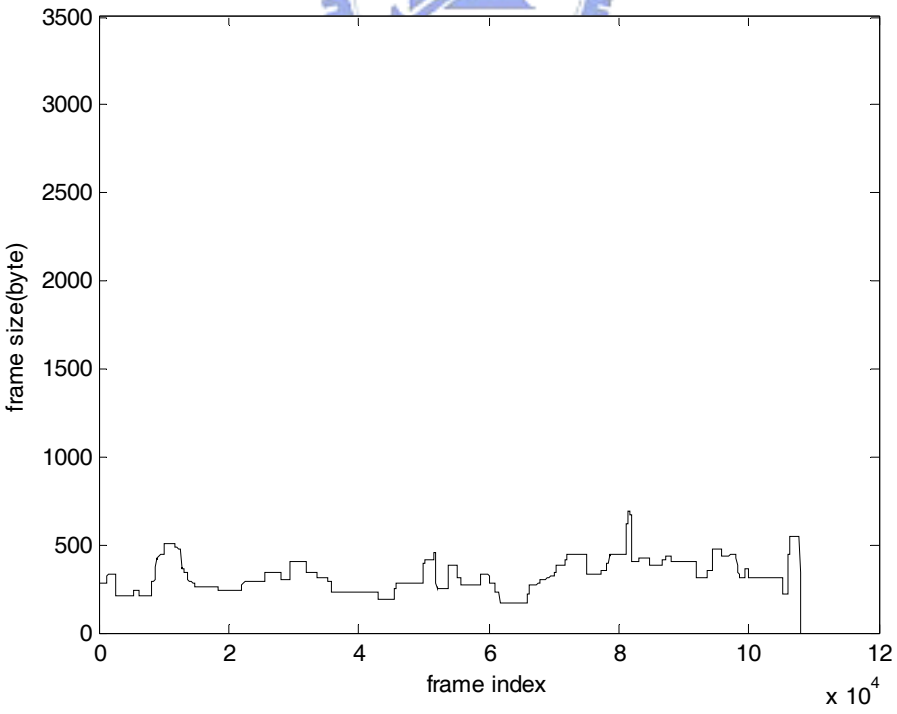

3500
3000
2500
2000
1500
1000
500
0
frame size(byte)
0
2
4
6
8
10
12
frame index
x 10^4

圖 4.4(b)

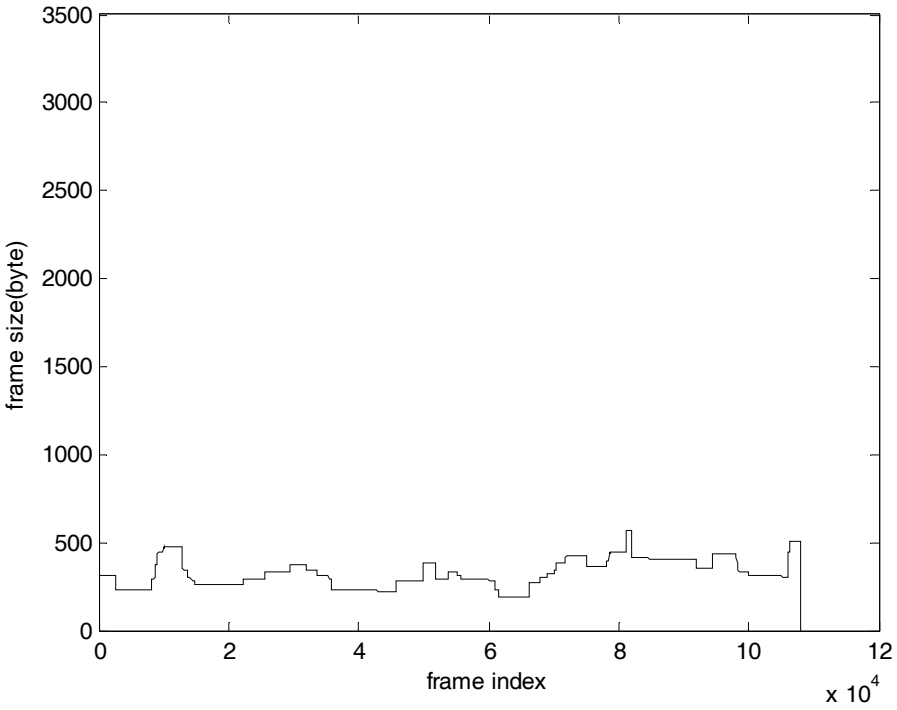

3500
3000
2500
2000
1500
1000
500
0
frame size(byte)
0
2
4
6
8
10
12
frame index
x 10^4

圖 4.4(c)

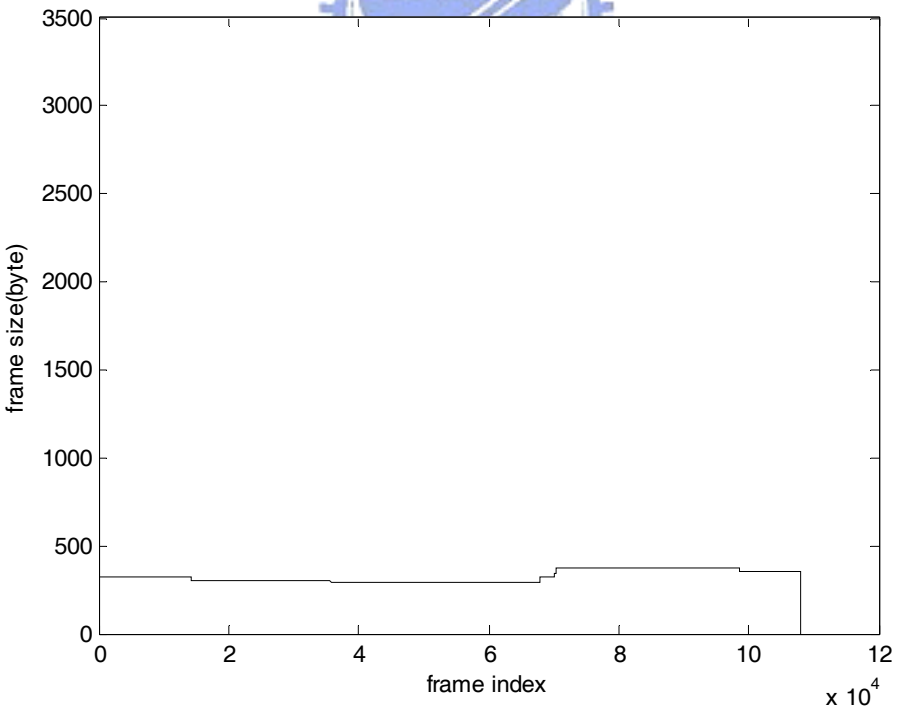

3500
3000
2500
2000
1500
1000
500
0
frame size(byte)
0
2
4
6
8
10
12
frame index
x 10^4

圖 4.4(d)

像第三章一樣我們觀察標準差之於緩衝區大小，峰值之於緩衝區大小的關係圖。

一、 標準差之於緩衝區大小

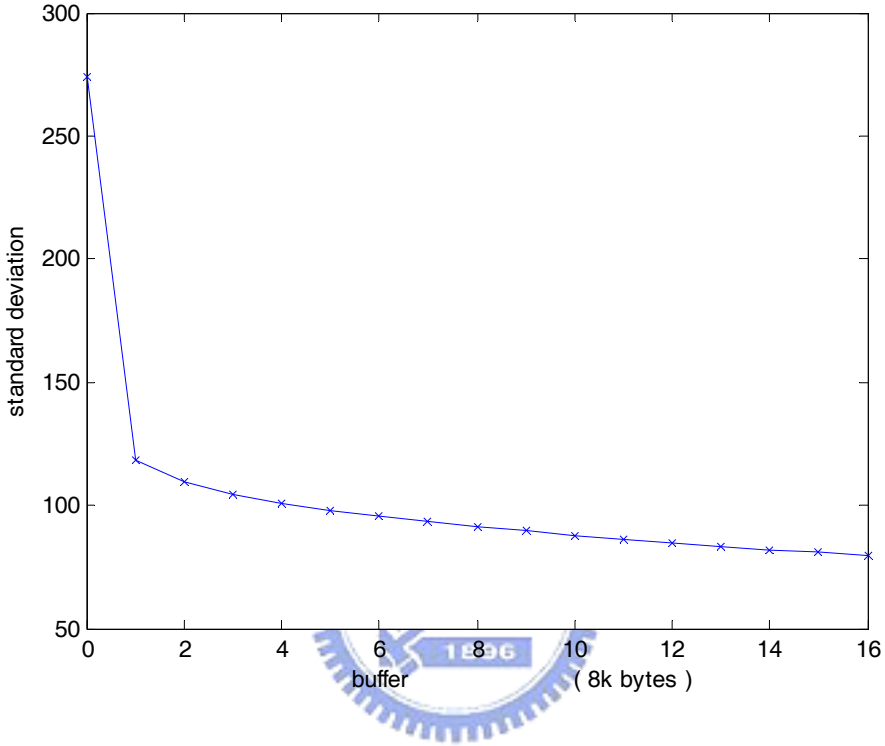


圖 4.5(a) standard deviation versus buffer size

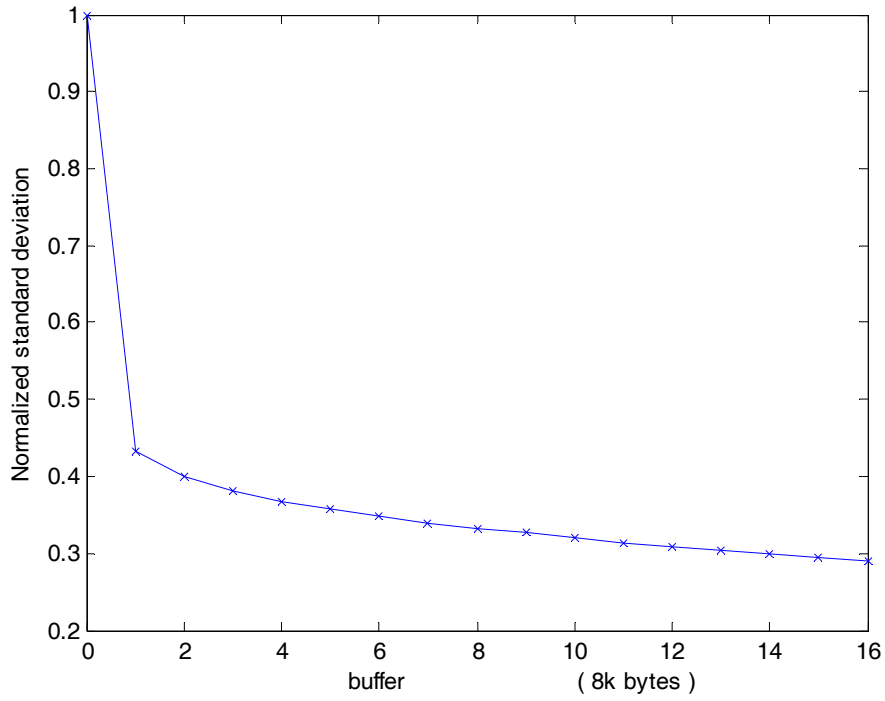


圖 4.5(b) Normalized standard deviation versus buffer size

二、峰值之於緩衝區大小

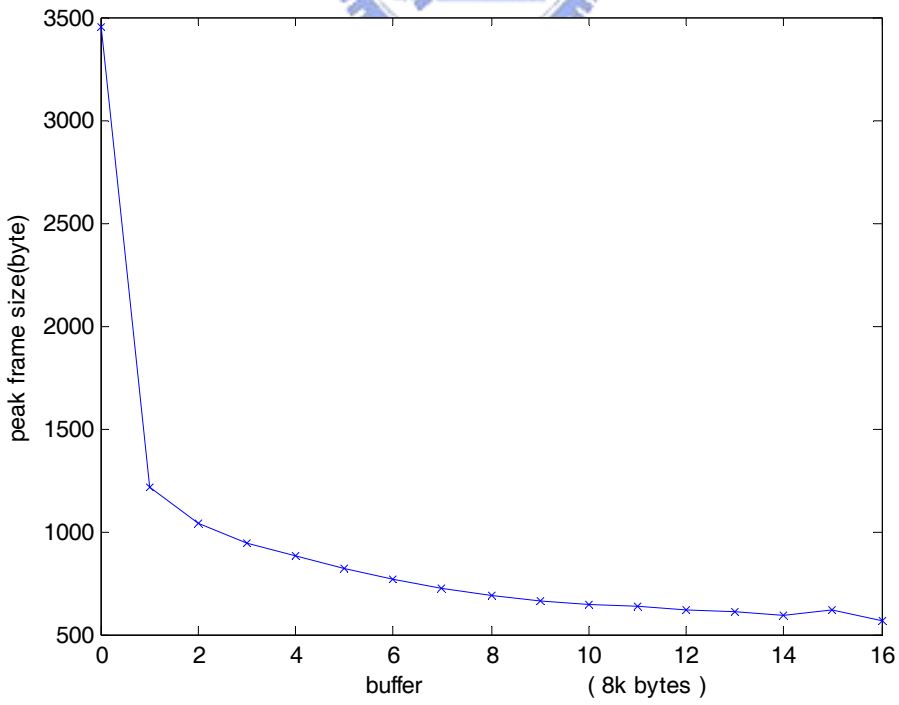


圖 4.5(c) peak frame size versus buffer size

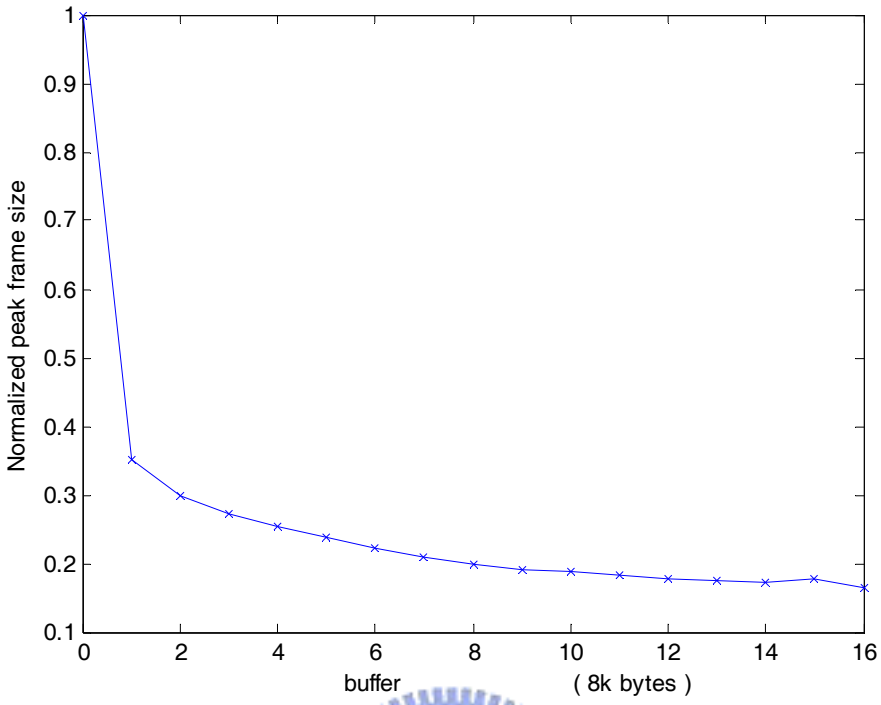
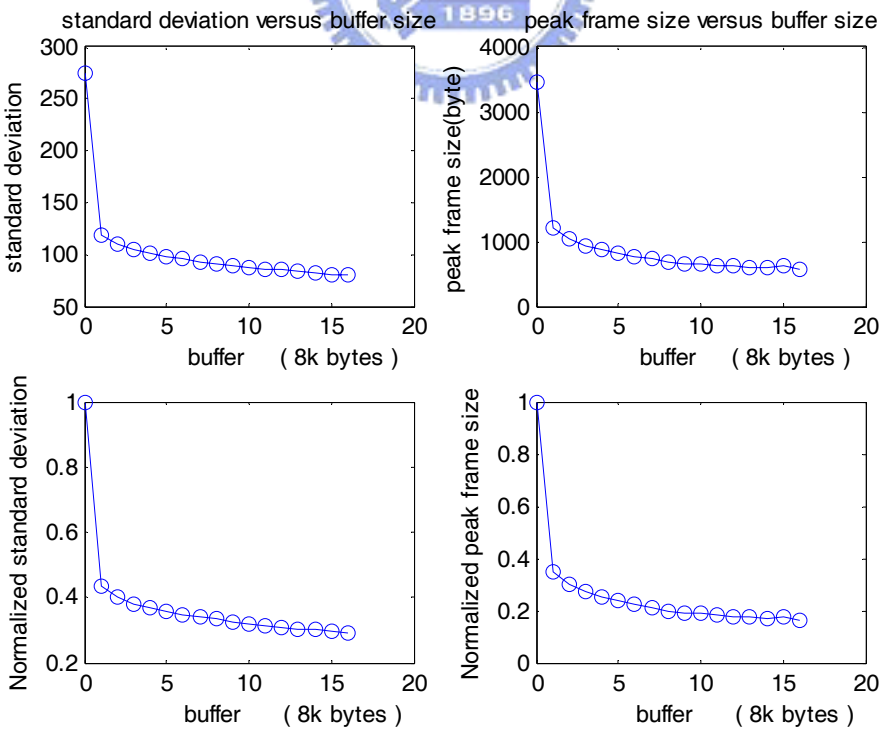


圖 4.5(d) Normalized peak frame size versus buffer size



三、速率改變次數之於緩衝區大小

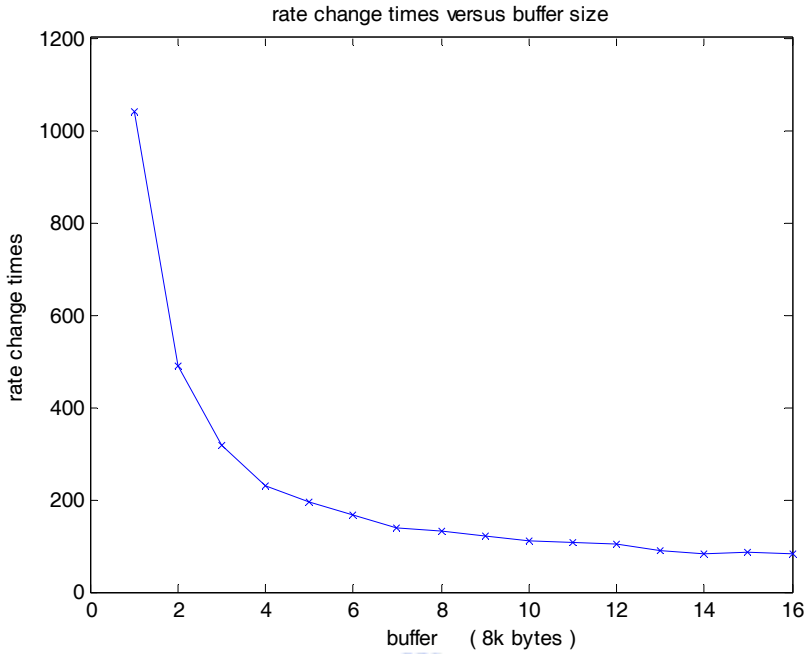


圖 4.5(e) rate change times versus buffer size

(不含未經平順的部分)

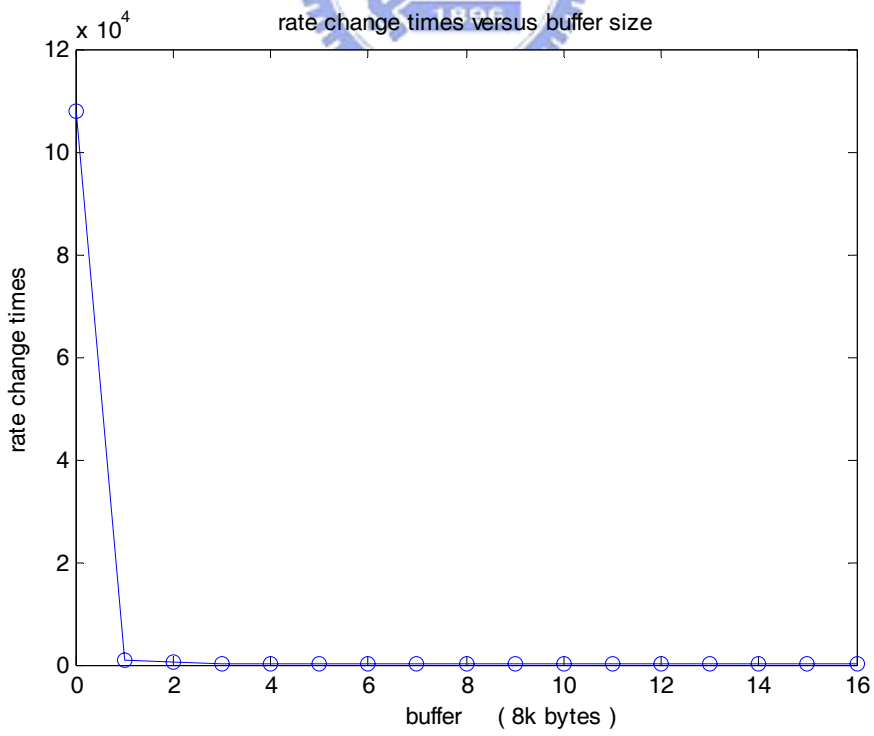


圖 4.5(f) rate change times versus buffer size(含未經平順的部分)

觀察幾個衡量的數據，我們發現 MBPOS 不論是在標準差、峰值及速率改變次數的表現趨勢幾乎和原本的最佳平順演算法(Optimal Smoothing Algorithm)結果雷同，不過實際差別有多少，並看不太出來，我們下節就有針對兩者細部的比較作探討。

4.3 比較 MBPOS 與最佳平順演算法(OS)的差別

本想針對標準差、峰值及速率改變次數比較 MBPOS 與 OS 的差異情形作比較及探討。

一、標準差

首先我們針對標準差，觀查圖 4.6(a)可能還看不出實際的情況，只能看出它們非常接近。我們可以由數值來比較，表 4.1(a)(b)。

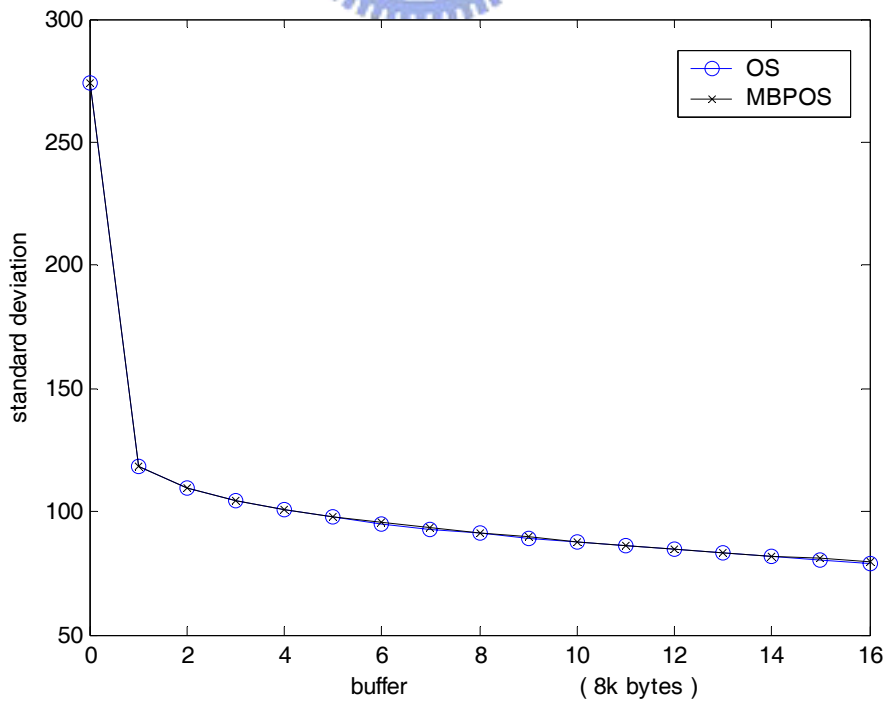


圖 4.6(a) OS 與 MPOS 標準差的比較圖

buffer size(KB)	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**
OS	118.4318	109.5624	104.3506	100.6735	97.7224	95.1868	92.9841	91.0084
MBPOS	118.6132	109.7632	104.5867	100.9357	97.9957	95.4742	93.2900	91.3485
buffer size(KB)	**72**	**80**	**88**	**96**	**104**	**112**	**120**	**128**
OS	89.2024	87.5386	85.9772	84.4986	83.0910	81.7382	80.4381	79.1860
MBPOS	89.5500	87.8567	86.2631	84.8166	83.4653	82.1048	80.7649	79.5355

表 4.1(a) OS 與 MBPOS 標準差詳細數值

buffer size(KB)	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**
difference	-0.1813	-0.2008	-0.2361	-0.2623	-0.2732	-0.2874	-0.3060	-0.3401
buffer size(KB)	**72**	**80**	**88**	**96**	**104**	**112**	**120**	**128**
difference	-0.3476	-0.3182	-0.2859	-0.3180	-0.3743	-0.3666	-0.3268	-0.3494

表 4.1(b) OS 與 MBPOS 標準差的差值

經由觀察的結果，標準差雖非常接近，不過 OS 還是比 MBPOS 些微小一點。因為原本的最佳平順演算法中，標準差已經是理論最佳化了。所以之後和它有稍微不一樣作法的，比它大也是很正常的結果。不過因為這個演算法一樣是採取最小差異速率的選取原則，所以標準差只有些微差異也是很正常的結果。

二、峰值

觀察圖 4.6(b)，峰值的差異也是幾乎不大，除了在 120KB 的時候有一個明顯的差別，其他的差異並不大。同樣可用數值來比較，表 4.1(c)(d)。

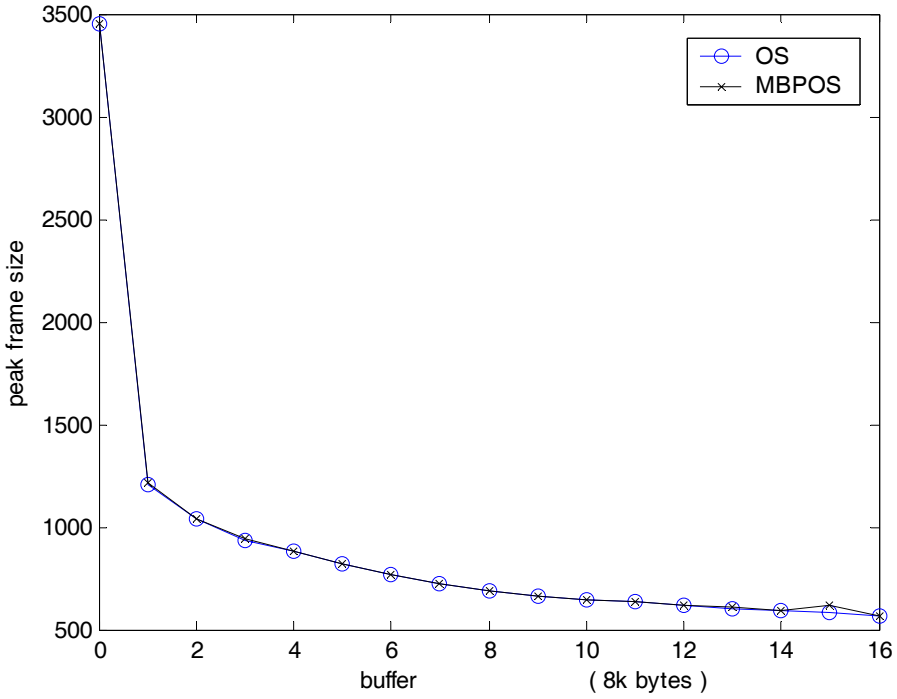


圖 4.6(b) OS 與 MBPOS 峰值的比較圖

buffer size(KB)	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**
OS	1210.1	1035.1	938.3	880	822.7	766.3	723.6	690.5
MBPOS	1211	1036	939	881	824	768	725	691

buffer size(KB)	**72**	**80**	**88**	**96**	**104**	**112**	**120**	**128**
OS	662.3	647.1	631.9	617.4	605.2	593	580.7	568.5
MBPOS	663	648	632	618	606	594	615	569

表 4.1(c) OS 與 MBPOS 峰值詳細數值

buffer size(KB)	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**
difference	-0.8684	-0.8800	-0.6563	-1.0420	-1.3287	-1.7192	-1.4315	-0.4637
buffer size(KB)	**72**	**80**	**88**	**96**	**104**	**112**	**120**	**128**
difference	-0.7333	-0.9037	-0.0741	-0.5531	-0.7982	-1.0433	-34.2885	-0.5336

表 4.1(d) OS 與 MBPOS 峰值的差值

經由觀察的結果，峰值雖非常接近，不過 OS 還是比 MBPOS 些微小一點。因為原本的最佳平順演算法中，峰值已經是理論最佳化了。所以之後和它有稍微不一樣作法的，比它大也是很正常的結果。雖然還是有愈來愈小的趨趨，但因為 MBPOS 已經不是理論上最佳了，所以不適用[5]所推論的式子，故 buffer size 愈大，peak frame size 愈小的結論在此不成立，故才會有 120Kbyte 中 peak frame size 反而沒有變更小的結果發生。

三、速率改變次數

如圖 4.6(c)所示，圖上可以很明顯看出，MBPOS 的速率改變次數均比 OS 小一個比例。觀察數值同樣可以明顯看出，表 4.1(e)(f)。

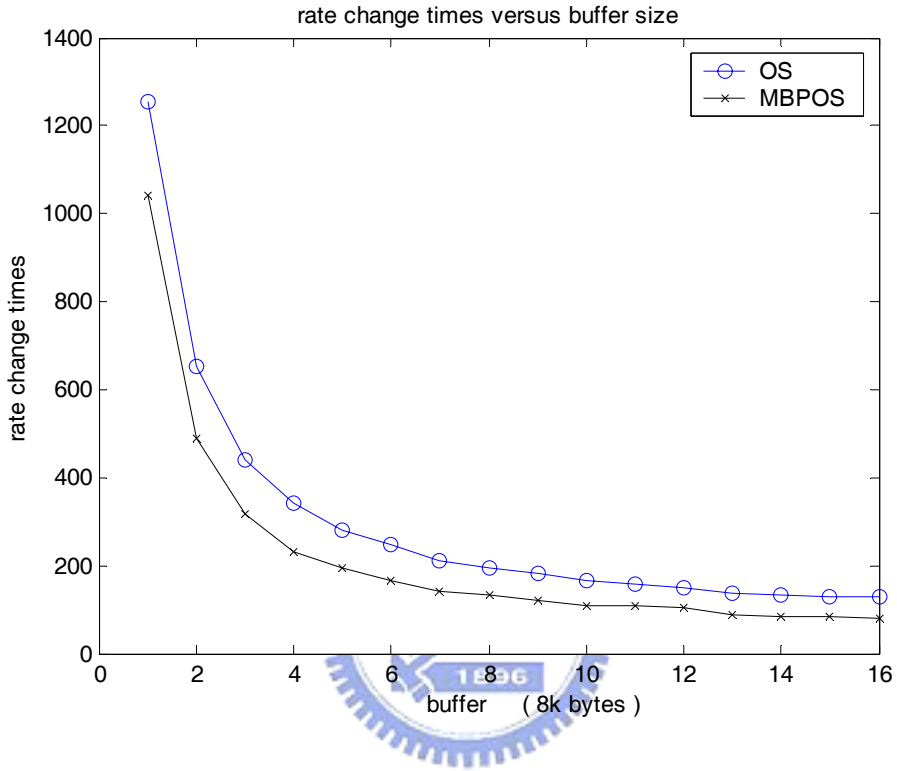


圖 4.6(c) OS 與 MBPOS 速率改變次數比較圖

buffer size(KB)	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**
OS	1256	652	441	340	282	247	210	195
MBPOS	1040	489	318	230	194	165	140	131
buffer size(KB)	**72**	**80**	**88**	**96**	**104**	**112**	**120**	**128**
OS	181	166	158	148	139	132	129	128
MBPOS	121	109	107	103	90	82	85	81

表 4.1(e) OS 與 MBPOS 速率改變次數詳細的數值

buffer size(KB)	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**
difference	216	163	123	110	88	82	70	64
buffer size(KB)	**72**	**80**	**88**	**96**	**104**	**112**	**120**	**128**
difference	60	57	51	45	49	50	44	47

表 4.1(f) OS 與 MBPOS 速率改變次數詳細的差值

由數值觀察的結果，我們可以看到 MBPOS 比原本 OS 明顯速率改變次數減低了。因為 OS 均是在邊緣作下個線段的起始點，當然可以選擇的範圍比較小，所以要延更長的比例也比較小；但是 MBPOS 並不在邊緣，而是離一個距離才是起始點，這樣一來，反而讓它的選擇範圍變大；雖然我們是用增加速率改變次數來增加它的可行性，但在一般緩衝區大小夠大的情況，遇到 4.1 節所討論遇到那種不可行的情況的機會其實不多，影響有限，故 MBPOS 速率改變次數會下降。因為 OS 本來就是以標準差最小化為目標，不是以改變次數最小為目標，所以可能會用較多的改變次數來達成標準差最小。有的平順演算法是在最小化速率改變次數[13]，它達成速率改變次數最小，不過可能會犧牲一些標準差。標準差與速率變動次數似乎存在著 trade-off 的關係。

4.4 總結

由以上模擬結果，MBPOS 與最佳平順的結果相比是增加了些微的標準差與峰值，但速率改變次數顯著的減少。MBPOS 可以說是 OS 修正在更實際狀況的版本，它失掉一些原本理想的特性，不過相對可適性更大。以後如果遇到不同的最小單位限制的問題，也可以用類似的方式解決。

【註 4】這個限制下產生的結果剛好是特例，剛好沒發生問題，最後一個區段也剛好是整數。

第五章

結論

本論文主要是針對最佳化平順演算法作探討與研究，模擬並分析其結果，之後則延伸去看去探討比較實際會遇到的問題。我們先是將原本的最佳平順演算法作一番討論、模擬與分析，之後則是延伸到位元組精確最小單位限制的問題上，由於理想狀態被打破，所以又遇到了其他的問題。我們對此作了修正，並將修正過的演算法作模擬分析並與原本演算法的方式作比較，由結果可看出其效果在標準差與峰值是相當接近但大於原本的結果，但是速率改變次數卻是小於原本結果。基本上最小單位限制下的問題是相當類似的，現在是以位元組，如果之後最小單位有改變，我們還是可以用相似的方法來解決。

由於篇幅時間能力有限，所以探討僅止於 stored video 上的討論，實際情形有可能會遇到 live 即時傳送的情況下。不過事實上兩者相關性很大，只是差別在 stored video 是可以一次針對一整個 video trace，live 的情況是一次針對一部分。

參考文獻

[1] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. Towsley, “Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing, ” *IEEE/ACM Trans. Networking*, vol. 6, pp. 397-410, 1998.

[2] W. Feng and J. Rexford, “A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video,” in *Proc. IEEE INFOCOM*, Apr. 1997, pp.58-66.

[3] S. Sen, J. Rexford, J. Dey, J. Kurose, and D. Towsley, “Online smoothing of variable-bit-rate streaming video,” *IEEE Trans. Multimedia*, vol. 2, pp.37-48. Mar. 2000.

[4] J. Rexford and D. Towsley, “Smoothing variable-bit-rate video in an internet-work,” *IEEE/ACM Trans. Networking*, vol. 7, pp. 202-215, Apr. 1999.

[5] G.. Sanjay and S. V. Raghavan, “Fast techniques for the optimal smoothing of stored video,” *ACM Multimedia Syst. J.*, vol. 7. pp. 222-233, 1999.

[6] S. Sen, D. Towsley, Z.-L. Zhang, and J. K. Dey, “Optimal multicast smoothing of streaming video over the internet,” *IEEE Journal on Selected Areas in Communications*, vol. 20, no.7, Sep. 2002.

[7] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, “Streaming Video over the Internet: Approaches and Direct- ions,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.11, No.3, March 2001, pp.282-300.

[8] Fred halsall, “Multimedia Communications-Applications, Networks, Protocol and Standards”,2001.

[9]M. Grossglauser, S. Keshav, and D. N. C. Tse, “RCBR: A simple and efficient service for multiple time-scale traffic,” *IEEE/ACM Trans.*

Networking, vol. 5, pp. 741-755, Dec. 1997.

[10]Z.-L. Zhang, J. Kurose, J. Salehi, and D. Towsley, “Smoothing, Statistical multiplexing and call admission control for stored video,” *IEEE J. Select. Areas Commun..*, vol. 15, pp.1148-1166, Aug. 1997.

[11] “Video traces for network performance evaluation,” Online Website. [Online]. Available: http://trace.eas.asu.edu

[12]W. Feng and S. Sechrest, “Critical bandwidth allocation for delivery of compressed video”, *Comp. Comm.*, vol. 18, pp.709-717, Oct. 1995.

[13]W. Feng and S. Sechrest, “Smoothing and buffering for delivery of precorded compressed video,” in *Proc. of the IS&T/SPIE Symp. on Multimedia Comp. and Networking*, pp. 234-242,Feb. 1995.

[14]A. W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and its Applications*. New York: Academic, 1979.

[15]J. Salehi, “Scheduling network processing on multimedia and multiprocessor servers,” Ph. D. dissertation, Univ. Massachusetts, Amherst, Sept. 1996.

附錄一

最佳平順演算法標準差最佳化的證明，主要是利用蓋理論(Majorization Theory)[14][15]所推得的結論：有兩個向量 X 和 Y，如果 Y 涵蓋(Majorize) X(數學符號記成 $X \prec Y$)，則 X 的標準差和峰值均會小於等於 Y 的標準差和峰值。[1]證明了最佳平順排程一定會被其他排程所涵蓋(Majorize)，由以上結論便可以推得最佳平順排程有最小的標準差和峰值。以下是詳細的證明，節錄自[1]。

定義一

蓋理論(Majorization Theory)[14]可以用在很多領域，它在這裡是被用來當作比較排程平順(smoothness)的數學工具。以下的說明可以幫助理解它的精神。

假設有 X, Y 兩個向量，其大致運作步驟如下所示：

步驟 1、將兩個向量裡的元素以遞減的方式排列。

步驟 2、向量內元素累加起來得到新的向量。

步驟 3、比較新向量中每個元素的大小。Y 裡每個元素均大於等於 X 裡每個元素的。就稱 Y 涵蓋 X，記為 $X \prec Y$。

舉例說，假如 X=[2,3,3,2] ，Y=[1,8,1,0]

步驟 1、X=[3,3,2,2],Y=[8,1,1,0]

步驟 2、累加之後 X=[3,6,8,10],Y=[8,9,10,10]

步驟 3、比較元素大小。

因為 $3 \le 8$

$6 \le 9$

$8 \le 10$

$10=10$

所以 Y 涵蓋 X($X \prec Y$)

步驟 1 比的就是最大值也就是峰值(max(Y)、max(X)就是代表峰值)。所以由 Y majorize X，可以推出 $\max(Y) \ge \max(X)$。

結果一：$X \prec Y$ if and only if $\sum_i \phi(x_i) \le \sum_i \phi(y_i)$ 對所有的 continuous convex functions $\phi : \mathbb{R} \to \mathbb{R}$。因為 variance 的公式 $\phi(x) = (x - \bar{x})^2 / N$ 是一個 continuous convex function，所以若 $X \prec Y$，即意味著 $\text{var}(X) \le \text{var}(Y)$。[14]

由上面兩個結果知道，某一向量只要是被另一向量所涵蓋，其峰值與標準差就比較小。下面就是要證明最佳排程一定會被其他排程所涵蓋。

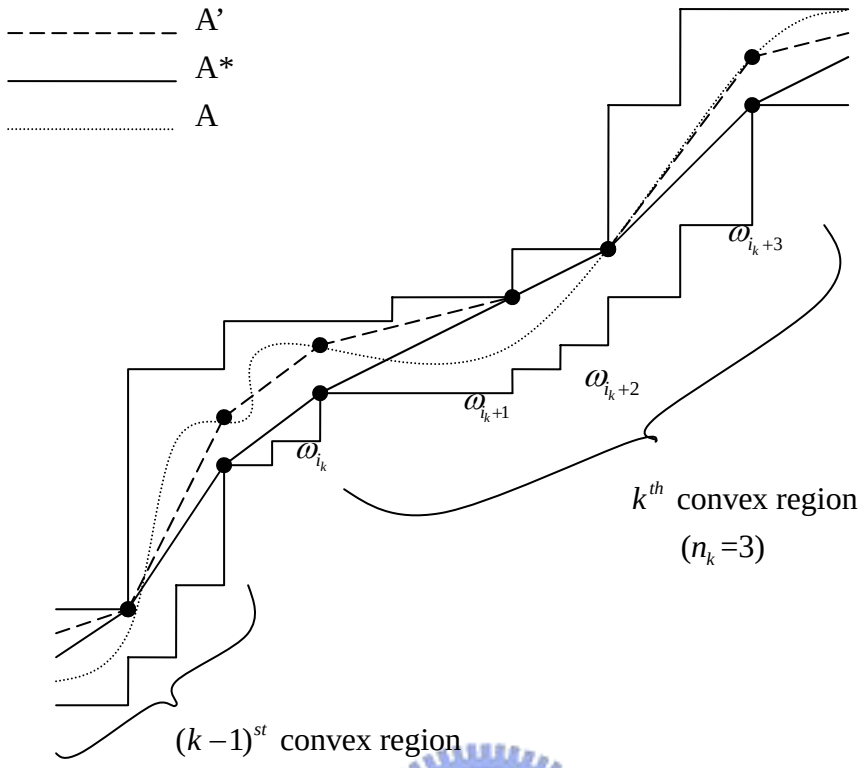


圖 b

這個證明是依據關於蓋理論的兩個已知的事實和一個輔助定理。([15]有證明)

事實一：U^k 和 V^k 是 m_k -dimensional real vectors $k = 1, 2, \ldots n$ ，$U^k \prec V^k$ 。假如我們藉由連接 $U^1, U,^2 \cdots, U^k (V^1, V^2, \cdots, V^k)$ 起來建立一個 $\sum_{i=k}^{n} m_k$ -dimensional vector $U(V)$ ，可推得 $U \prec V$ 。

事實二：假如 X 是 linear(i.e.，假如 $x_1 = x_2 = \cdots = x_n = c$)，則使得所有的 Y 存在 $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_i = nc$ 和 $X \prec Y$ 。

定義二：假如存在一個 X 轉折點的集合$\{\omega_k\}$($X = [x_1, x_2, \cdots, x_n]$)，$k = 0,1,2,\cdots,m$, 其中 $\omega_0 = 0 < \omega_1 < \omega_2 \cdots < \omega_{m-1} < \omega_m = n$ 使得對所有的 $i, j \in \{\omega_{k-1}+1, \cdots, \omega_k\}$ 存在 $x_i = x_j$ ，我們就稱 X 是 m-segment piecewise linear。

定義三：假如 X 是 m-segment piecewise linear 而且對 $k = 1,2,\cdots m-1$ 存在 $x_{\omega_k} > x_{\omega_{k+1}}$ ($x_{\omega_k} < x_{\omega_{k+1}}$)。$X = [x_1, x_2, \cdots, x_n]$ 就稱為 concave(convex) m-segment piecewise linear。

輔助定理一： $X, Y \in \mathbb{R}^n$ 使得 X 是有轉折點 ω_k($k = 0,1,\cdots m$) 的 concave (convex) m-segment piecewise linear。假如對 $k = 0,1,\cdots$ m-1(以嚴格的等式 $k = m$) $\sum_{i=1}^{\omega_k} x_i \le \sum_{i=1}^{\omega_k} y_i \left(\sum_{i=1}^{\omega_k} x_i \ge \sum_{i=1}^{\omega_k} y_i\right)$，則 $X \prec Y$ 。

定義四：假如 $\omega_{i+1}, \cdots, \omega_{i+n-1}$ 是 convex change points 而 ω_i 和 ω_{i+n} 兩個均是 concave change points，我們用 $\langle \omega_i, n \rangle$ 表示一個 convex region。S^* 有有限個數 K 個 convex regions，或者全無。假如 $K \ge 1$，讓 $\langle \omega_{i_k}, n_k \rangle$ 代表 kth convex region， $k = 1,2,3,\cdots,K$ 。

定理一：讓 $S^* = [a^*(1), \cdots, a^*(N)]$ 代表由最佳平順演算法根據已知的緩衝區大小和 video trace 所產生的傳輸排程。假如 $S(t) = [a(1), \cdots, a(N)]$ 代表那個 video 的任一可行排程，則 $S^* \prec S$ 。

證明：

我們的方法是建立一個 $S' = [a'(1), \cdots, a'(N)]$ 使得 $S^* \prec S'$ 且 $S' \prec S$ 。(S' 不一定要可行)。令 $A^*(t) = \sum_{i=1}^{t} a^*(i)$, $A(t) = \sum_{i=1}^{t} a(i)$, $A'(t) = \sum_{i=1}^{t} a'(i)$ 。假如 S^* 不含 convex regions。***定理一***是由***輔助定理一***直接而來，其中 $X = S^*$ ，$Y = S$ 。因此，我們假設 K≥1 的情況。看圖 b，我們基本想法就是建立一個 m-segment piecewise linear S' ，使得 A' 會與 A^* 在 S^* 的 convex change points 交會，A' 會與 A 在 S^* 的 concave change points 交會。

S' 以下面的方式定義。ω_i 和 ω_{i+1} 為 S^* 連續的轉折點，$i = 0,1,\cdots, m-1$ 。對所以 $t \in \{\omega_i + 1, \cdots, \omega_i\}$ ，當 ω_i 和 ω_{i+1} 同時都是 concave，定義 $a'(t)$ 為 $((A(\omega_{i+1}) - A(\omega_i))/(\omega_{i+1} - \omega_i))$ ，當 ω_i 和 ω_{i+1} 都是 convex，定義 $a'(t)$ 為 $((A^*(\omega_{i+1}) - A^*(\omega_i))/(\omega_{i+1} - \omega_i))$ ，當 ω_i 是 concave，ω_{i+1} 是 convex，定義 $a'(t)$ 為 $((A^*(\omega_{i+1}) - A(\omega_i))/(\omega_{i+1} - \omega_i))$ ，當 ω_i 是 convex，ω_{i+1} 是 concave，定義 $a'(t)$ 為 $((A(\omega_{i+1}) - A^*(\omega_i))/(\omega_{i+1} - \omega_i))$ 。

我們由下面的方式建立 $S' \prec S$ 。讓 $X[i, j] = [x_i + 1, \cdots, x_j]$ 代表一個 $(j - i)$- dimensional X subvector。在 S^* 的 k^{th} convex region，$\langle \omega_{i_k}, n_k \rangle$ ，我們由***輔助定理一***會得到 $S'[\omega_{i_k}, \omega_{i_k + n_k}] \prec S[\omega_{i_k}, \omega_{i_k + n_k}]$ 的結果。這對所有的 convex regions 均成立。A' 會在所有的 concave change points 與 A 交會(其中 A' 在改變點間是 linear)這個事實，由***事實二***來看這意味著在

convex region 外，$S' \prec S$。因此，由***事實一***我們可以知道在[1,*N*]區間內，$S' \prec S$。

要證明$S^* \prec S'$，我們考慮對每個區間$[i, j] \in$ {$[0, \omega_{i_1+1}], [\omega_{i_1+n_1-1}, \omega_{i_2+1}], \cdots, [\omega_{i_k+n_k-1}, \omega_m$]}(它們是S^*的 convex region)我們可以直接從***輔助定理一***得到$S^*[i, j] \prec S'[i, j]$。而由***事實一***，我們可以得到$S^* \prec S'$，又$S' \prec S$，所以推得$S^* \prec S$。故根據 Majorization，S^*是佳平順。

由 Majorization 的定義，我們知道S^*有最小的峰值，而由***結果一***，我們便可以知道S^*有最小的標準差。