I854675

(19)中華民國智慧財產局

(12)發明說明書公告本　**(11)證書號數：TW I854675 B**

(45)公告日： 中華民國 113 (2024) 年 09 月 01 日

(21)申請案號：112120129　(22)申請日： 中華民國 112 (2023) 年 05 月 30 日

(51)Int. Cl.：　*H03M13/00 (2006.01)*　*H03M13/03 (2006.01)*

(71)申請人：睿寬智能科技有限公司 (中華民國)　(TW)

臺北市內湖區堤頂大道 2 段 201 號 4 樓

(72)發明人：魏大鈞 (TW) ；曾戈忠 (TW)

(74)代理人：謝佩玲；王耀華；陳仕勳

(56)參考文獻：

CN	107404320A	US	8489962B2
US	2004/0098517A1	US	2016/0197701A1
US	2022/0231698A1		

審查人員：蘇齊賢

申請專利範圍項數：4 項　圖式數：14　共 44 頁

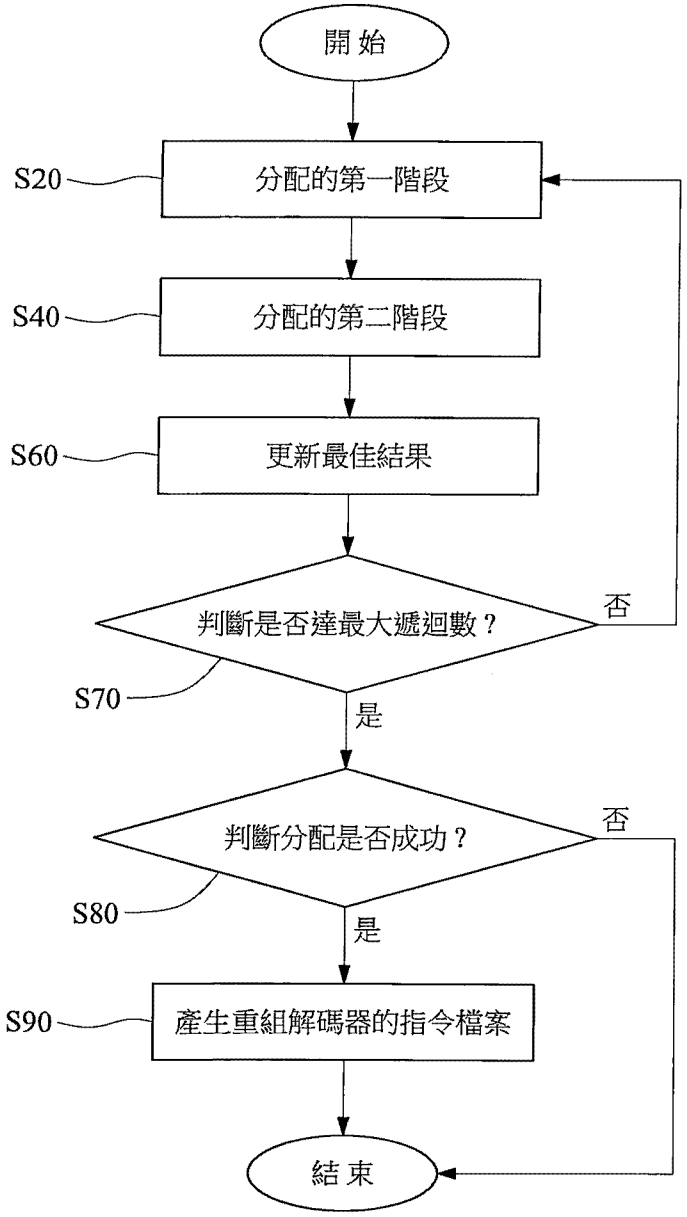
(54)名稱

重排ＬＤＰＣ碼而改善ＬＤＰＣ重組解碼器並行的方法

(57)摘要

LDPC 重組解碼器(LDPC shuffle decoder)有兩個核心及兩個記憶體，依序處理一個 LDPC 矩陣的所有欄位。處理每一欄位時，把它的非零元素的讀(reading)與寫(writing)均分給兩個記憶體。依此分派方式，LDPC 重組解碼器可在最少的循環完成此 LDPC 矩陣的讀及寫。

指定代表圖：



【圖 3】

符號簡單說明：

S20:分配的第一階段

S40:分配的第二階段

S60:更新最佳結果

S70:判斷是否達最大遞迴數

S80:判斷分配是否成功

S90:產生重組解碼器的指令檔案

I854675

公告本

發明摘要

【發明名稱】(中文/英文)

重排 LDPC 碼而改善 LDPC 重組解碼器並行的方法/

Method for Improving Parallelization of an LDPC Shuffle Decoder by Reordering an LDPC Code

【中文】

LDPC 重組解碼器(LDPC shuffle decoder)有兩個核心及兩個記憶體，依序處理一個 LDPC 矩陣的所有欄位。處理每一欄位時，把它的非零元素的讀(reading)與寫(writing)均分給兩個記憶體。依此分派方式，LDPC 重組解碼器可在最少的循環完成此 LDPC 矩陣的讀及寫。

【英文】

【代表圖】

【本案指定代表圖】：圖（3）。

【本代表圖之符號簡單說明】：

S20：分配的第一階段

S40：分配的第二階段

S60：更新最佳結果

S70：判斷是否達最大遞迴數

S80：判斷分配是否成功

S90：產生重組解碼器的指令檔案

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：

本案無化學式

發明專利說明書

（本說明書格式、順序，請勿任意更動）

【發明名稱】（中文/英文）

重排 LDPC 碼而改善 LDPC 重組解碼器並行的方法/

Method for Improving Parallelization of LDPC Shuffle Decoder by Re-Ordering LDPC Codes

【技術領域】

【0001】 本發明關於對LDPC重組解碼器，尤其關於重排LDPC碼而改善LDPC重組解碼器並行的方法。

【先前技術】

【0002】 在LDPC重組解碼(LDPC shuffle decoding)中，解碼器以欄位為單位處理上述矩陣。

【發明內容】

【0003】 有鑑於上述習知技藝之問題，本發明之目的是提供一種重排LDPC碼欄位而縮短LDPC重組解碼器延遲的方法。

【0004】 為達成上述目的，上述方法包括：為上述LDPC重組解碼器提供兩個記憶體，平均分配上述LDPC碼的矩陣的非零元素給上述兩個記憶體，依上述分配方式，從上述兩個記憶體讀上述非零元素，並寫上述非零元素到上述兩個記憶體，更新分配方式，判斷是否達最大遞迴數。若未達最大遞迴數，則回上述平均分配上述非零元素給上述兩個記憶體的步驟。若達最大遞迴數，則判斷分配是否成功。若分配成功，則產生排程檔案，否則結束。

【圖式簡單說明】

【0005】

〔圖1〕現一個 LDPC 碼；

〔圖2〕呈現一個典型的 LDPC 並行解碼器；

〔圖3〕是本發明的重排 LDPC 碼而改善 LDPC 重組解碼器並行的方法的流程圖；

〔圖4〕是圖 3 所示之重排 LDPC 碼而改善 LDPC 重組解碼器並行的方法的一道子程序的流程圖；

〔圖5〕呈現一個被簡化的 LDPC 碼與其平衡讀寫之分配；

〔圖6〕呈現另一個 LDPC 碼的一種失衡分配方式；

〔圖7〕呈現圖 6 之 LDPC 碼回溯後的一種平衡分配；

〔圖8〕呈現圖 6 之 LDPC 碼回溯後的另一種失衡分配；

〔圖9〕呈現圖 6 之 LDPC 碼可能產生的巢狀回溯分配；

〔圖10〕是圖 3 所示之重排 LDPC 碼而改善 LDPC 重組解碼器並行的方法的另一道子程序的流程圖；

〔圖11〕呈現以一種記憶體的容量處理一個 LDPC 碼；

〔圖12〕呈現處理圖 11 所示的 LDPC 碼的第一次環繞；

〔圖13〕呈現處理圖 11 所示的 LDPC 碼的第二次環繞；及

〔圖14〕呈現以另一種記憶體容量處理圖 11 的 LDPC 碼的第一次環繞。

【實施方式】

【0006】 以下參考相關圖式進一步說明本發明的較佳實施例。為便於理解本發明，以下用相同符號標示相同元件。

【0007】 參考圖 1，LDPC (low-density parity check)碼以矩陣的形式存

在。此矩陣有 M 列(row)*N 欄位(column)的元素，每一個元素是一個較小的循環矩陣(circulant)。若循環矩陣僅包括數值 0，則此元素是「零元素」(圖中，用 X 表示)。若循環矩陣包含數值 1，則此元素是「非零元素」(圖中，用數字表示循環位移值)。實務上，運算LDPC碼時僅須用非零元素。受限於解碼器硬體限制，LDPC 重組解碼以「欄」為單位向後進行運算。

【0008】 參考圖2，為增加LDPC解碼效率，可拓展LDPC解碼器，此時解碼器擁有兩個記憶體(A及B)及兩個核心(1及2)。核心1及2都是處理器。理想上，核心1從記憶體A(或B)讀資料時，核心2從記憶體B(或A)讀資料。接著，LDPC解碼器處理這些資料。然後，核心1把被處理的資料寫回記憶體A(或B)時，核心2把被處理的資料寫回記憶體B(或A)。核心1及核心2同時運作被稱為「並行」(parallelization)。然而，同一列之相異欄位的非零元素有相依性(dependency)，就是處理每一個非零元素時所需的資料皆是讀取同列先前欄位所載內容(因記憶體位置須相同)。此外，LDPC矩陣中非零元素的分布是稀疏且不規則，因此運算每一欄時，兩個核心可能須同時讀(或寫)單一記憶體A(或B)。此時，只有一個核心工作，須閒置另一核心，導致整體系統延宕。

【0009】 圖 3 呈現本發明的較佳實施例的重排 LDPC 碼而改善 LDPC 重組解碼器的方法。本發明的方法重排上述 LDPC 碼而改善上述 LDPC 重組解碼器並行。換言之，本發明的方法避免一個核心(1 或 2)運作時，另一個核心(2 或 1)閒置。具體而言，本發明的方法把每一欄位的非零

元素的資料讀取與寫入排序(sort)或分配，致平行且平衡處理這些非零元素而最佳化上述 LDPC 重組解碼器的效能。

【0010】 在步驟 S20，進行分配的第一階段。檢視整個矩陣，標示且安排每一欄位的一些非零元素到記憶體 A，標示且安排每一欄位的另一些非零元素到記憶體 B，並確保兩個記憶體有最適當的平衡狀態。重複此步驟，兩個記憶體平衡或達最大遞迴數(兩個記憶體失衡)時才停。

【0011】 在步驟 S40，進行分配的第二階段。用此舉發現記憶體 A 或 B 是否因上述分配而導致記憶體溢出(overflow)的錯誤，並傳遞錯誤標記到步驟 S60。

【0012】 在步驟 S60，依限制及硬體要求，更新最佳結果。只留一個結果當最後結果。為此，把本次遞迴的結果與先前貯存的結果相比。若把本次遞迴的結果優於先前貯存的結果，則用本次遞迴的結果取代先前貯存的結果，否則留先前貯存的結果。

【0013】 在步驟 S70，判斷是否達最大遞迴數。若是，則到步驟 S80，否則回步驟 S20。因此，重複步驟 S20、S40 及 S60，達最大遞迴數才停。

【0014】 在步驟 S80，判斷分配是否成功。若是，則到步驟 S90，否則結束。

【0015】 在步驟 S90，產生重組解碼器的指令檔案當最後輸出資料。此指令檔案被稱為「排程檔案」(scheduling files)。

【0016】 參考圖 4，詳細描述步驟 S20。步驟 S20 的目標是以平衡的

方式把非零元素投入記憶體 A 及記憶體 B。步驟 S20 包括步驟 S21、S23、S25、S27、S29、S31、S33、S35、S37、S39。步驟 S21、S23、S25、S27、S29 及 S31 屬於讀的階段，步驟 S33 屬於寫的階段，步驟 S35、S37、S39 屬於後續處理的階段。

【0017】 在步驟 S21，作為讀取的第一步驟，累計某欄位中分別從記憶體 A 及 B 來的非零元素的數量。為此，須檢視前一或數個欄位在同一列的非零元素是寫入記憶體 A 或記憶體 B。應注意，此時矩陣最前面的幾個欄位可能無先前記憶體資訊，故暫不累計此欄位的元素數量，但假定其來自兩個記憶體的數量平衡，使在步驟 S25 滿足條件並進入步驟 S33 而繼續寫入記憶體的步驟。這些未確認的讀取記憶體資訊，稍後會在步驟 S37 被處理。

【0018】 在步驟 S23，若當今欄位是新欄位(表示分配已進行到新欄位)，則暫存當今整體分配狀態當最佳分配狀態，並記錄現今最大欄位序號。此最大欄位序號紀錄可用來判斷現今欄位是否為新欄位，若現今欄位序號大於先前最大欄位序號，則此欄位是新欄位。

【0019】 在步驟 S25，取用步驟 S21 的結果，判斷分別從兩個記憶體 A 及 B 讀取的數量是否平衡。換言之，判斷此欄位之非零元素們從記憶體 A 讀取的量是否等於從記憶體 B 讀取的量(若非零元素總數為奇數，則滿足平衡之條件為相差 1)。若是，則到步驟 S33，否則到步驟 S27。

【0020】 在步驟 S27，表示從兩記憶體讀取的數量不平衡，須執行回溯(rolling back)予以修正。首先判斷回溯次數是否達上限。若是，則到

步驟 S31，否則到步驟 S29 執行回溯。回溯次數為統計在固定欄位範圍內(未往後執行到任何新欄位的狀況下)執行回溯的次數，即是在此範圍裡，每回溯 1 次，就把回溯次數加 1。同時定義回溯次數上限，以避免在找不到任何平衡的分配狀態時，陷於無限回溯之迴路。此次回溯可能在先前的回溯中，即在執行回溯時，可能觸發另一次的回溯，但只要發生在此固定範圍內，每次回溯都應觸發次數加 1。

【0021】 在步驟 S29，實際執行回溯。隨機選一個造成失衡狀況的非零元素，回到同一列中前一個非零元素，並從步驟 S21 重新開始處理它所在的那一個欄位。重新處理此欄位時，因在步驟 S33 是依平衡寫入條件隨機分配寫到記憶體 A 或記憶體 B，將有機會把後續欄位(包括上述造成失衡的那一個欄位)的讀取數量重新分配而達到平行狀態。

【0022】 在步驟 S31，若回溯時未發現更佳組合且回溯次數已達上限，則回復先前暫存之最佳狀況(在步驟 S23 被貯存)。

【0023】 在步驟 S33，依平衡的寫入條件，隨機分配非零元素的資料寫到記憶體 A 或記憶體 B。

【0024】 在步驟 S35，判斷是否已達矩陣的最後欄位。若是，則到步驟 S37，否則回步驟 S21 繼續處理下一欄位。

【0025】 在步驟 S37，處理開頭欄位中未確認的記憶體讀取資訊。參考最後幾個欄位的寫入資訊，得知資料所存放之記憶體，再回到開頭欄位中填入其相對應讀取的記憶體。換言之，視最後幾個欄位視為在最前面幾個欄位以前的先前記憶體資訊。

【0026】 在步驟 S39，判斷是否每一欄位的讀取數量皆平衡(不必判

斷寫入，因寫入時依平衡之數量寫入)或已達最大遞迴數。若是，則到步驟 S40，否則回步驟 S21。

【0027】 參考圖 5，舉例描述如何執行本發明的方法。

【0028】 在步驟 S21，暫不處理欄位 1 之資料讀取，因不知列 2、3、5 或 6 先行資料的寫被分配到記憶體 A 或記憶體 B。在步驟 S23，記錄當今狀態，並假設讀取數量平衡而由步驟 S25 跳至步驟 S33。在步驟 S33，將非零元素資料依平衡之數量寫入記憶體 A 與記憶體 B。

【0029】 在步驟 S21，暫不處理欄位 2 之資料讀取，因不知列 1、4 先行資料的寫被分配到記憶體 A 或記憶體 B。在步驟 S23，記錄當今狀態，並假設讀取數量平衡而由步驟 S25 跳至步驟 S33。在步驟 S33，將非零元素資料依平衡之數量寫入記憶體 A 與記憶體 B。

【0030】 在步驟 S21，處理欄位 3，可知列 1 的資料須從記憶體 A 讀取(由欄位 2 列 1 寫入)，列 2 的資料須從記憶體 A 讀取(由欄位 1 列 2 寫入)，列 3 的資料須從記憶體 B 讀取(由欄位 2 列 3 寫入)，列 4 的資料須從記憶體 B 讀取(由欄位 2 列 4 寫入)。

【0031】 在步驟 S23，判斷欄位 3 是新欄位，並因此存當今分配狀態當最佳狀態。

【0032】 在步驟 S25，從步驟 S21 的結果，判斷兩個記憶體平衡，並因此到步驟 S33。

【0033】 在步驟 S33，把非零元素資料依平衡之數量寫入記憶體。此時，分配列 1 及 2 的寫到記憶體 A，並分配列 3 及 4 的寫到記憶體 B。

【0034】 在步驟 S35，判斷欄位 3 非最後欄位，並因此回步驟 S21。

【0035】 在步驟 S21，處理欄位 4，可知列 2 的資料須從記憶體 A 讀取(由欄位 3 列 2 寫入)，列 4 的資料須從記憶體 B 讀取(由欄位 3 列 4 寫入)，列 5 的資料須從記憶體 A 讀取(由欄位 2 列 5 寫入)，列 6 的資料須從記憶體 B 讀取(由欄位 1 列 6 寫入)。

【0036】 在步驟 S23，判斷欄位 4 是新欄位，並因此存當今分配狀態當最佳狀態。

【0037】 在步驟 S25，從步驟 S21 的結果，判斷兩個記憶體平衡，並因此到步驟 S33。

【0038】 在步驟 S33，把非零元素資料依平衡之數量隨機分配寫入記憶體。此時分配列 2 及 5 的寫到記憶體 A，並分配列 4 及 6 的寫到記憶體 B。

【0039】 在步驟 S35，判斷欄位 4 非最後欄位，並因此回步驟 S21。

【0040】 在步驟 S21，處理欄位 5，可知列 2 的資料須從記憶體 A 讀取(由欄位 4 列 2 寫入)，列 3 的資料須從記憶體 B 讀取(由欄位 3 列 3 寫入)，列 5 的資料須從記憶體 A 讀取(由欄位 4 列 5 寫入)，列 6 的資料須從記憶體 B 讀取(由欄位 4 列 6 寫入)。

【0041】 在步驟 S23，判斷欄位 5 是新欄位，並因此存當今分配狀態當最佳狀態。

【0042】 在步驟 S25，從步驟 S21 的結果，判斷兩個記憶體平衡，並因此到步驟 S33。

【0043】 在步驟 S33，把非零元素資料依平衡之數量寫入記憶體。

此時，分配列 2 及 3 的寫到記憶體 A，並分配列 5 及 6 的寫到記憶體 B。

【0044】 在步驟 S35，判斷欄位 5 是最後欄位，並因此到步驟 S37。

【0045】 在步驟 S37，開始進行後續的環繞分配，由欄位 1 開始填補缺失的讀取資訊。此時可知列 2 的資料須從記憶體 A 讀取(由欄位 5 列 2 寫入)，列 3 的資料須從記憶體 A 讀取(由欄位 5 列 3 寫入)，列 5 的資料須從記憶體 B 讀取(由欄位 5 列 5 寫入)，列 6 的資料須從記憶體 B 讀取(由欄位 5 列 6 寫入)。

【0046】 以此方式繼續填補欄位 2 缺的讀取資訊。可知列 1 的資料須從記憶體 A 讀取(由欄位 3 列 1 寫入)，列 3 的資料須從記憶體 A 讀取(由欄位 1 列 3 寫入)，列 4 的資料須從記憶體 B 讀取(由欄位 4 列 4 寫入)，列 5 的資料須從記憶體 B 讀取(由欄位 1 列 5 寫入)。

【0047】 在步驟 S39，判斷所有欄位在讀取數量上皆為平衡(從記憶體 A 的讀與記憶體 B 的讀平衡)，並因此結束步驟 S20。

【0048】 參考圖 6 及圖 7，描述回溯 1 次，從一種失衡的分配狀態變成一種平衡的分配狀態。

【0049】 在圖 6，欄位 1 到欄位 k+2 中，每一欄位從兩個記憶體讀取的數量平衡。在欄位 k+3 讀取資料時，從兩個記憶體讀取的數量失衡，可知列 1 到列 3 的讀取是從記憶體 A，僅列 4 的讀取是從記憶體 B。

【0050】 在步驟 S21，發現欄位 k+3 有 3 個非零元素的讀取是從記憶體 A，1 個非零元素的讀取是從記憶體 B。

【0051】 在步驟 S23，判斷欄位 k+3 是新欄位，並因此存當今分配狀態當最佳狀態。

【0052】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的數量失衡，並因此到步驟 S27。

【0053】 在步驟 S27，判斷回溯次數未達上限，並因此到步驟 S29。

【0054】 在步驟 S29，隨機選列 1(圖 7)，回欄位 k+2，並回步驟 S21。此時，回溯次數須加 1。

【0055】 參考圖 7，在步驟 S21，發現欄位 k+2 有 2 個非零元素的讀取是從記憶體 A，2 個非零元素的讀取是從記憶體 B。

【0056】 在步驟 S23，判斷欄位 k+2 是舊欄位，並因此不存當今分配狀態當最佳狀態。

【0057】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的數量平衡，並因此到步驟 S33。

【0058】 在步驟 S33，把非零元素資料依平衡數量且隨機分配的方式寫入記憶體，此時分配列 3 及 5 的寫到記憶體 A，並分配列 1 及 4 的寫到記憶體 B。

【0059】 在步驟 S35，判斷欄位 k+2 非最後欄位，並因此回步驟 S21。

【0060】 在步驟 S21，發現欄位 k+3 有 2 個非零元素的讀被分配到記憶體 A，2 個非零元素的讀被分配到記憶體 B。

【0061】 在步驟 S23，判斷欄位 k+3 是舊欄位，並因此不存當今分配狀態當最佳狀態。

【0062】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的

數量平衡，並因此到步驟 S33 執行資料寫入步驟。此時，資料讀取已由原本失衡的分配狀態，變成一種平衡的分配狀態。

【0063】 參考圖 6 及圖 8，描述回溯 1 次，從一種失衡的分配狀態到另一種失衡分配狀態的可能狀況及後續處置方式。

【0064】 參考圖 6，在步驟 S21，發現欄位 k+3 有 3 個非零元素的讀取是從記憶體 A，1 個非零元素的讀取是從記憶體 B。

【0065】 在步驟 S23，判斷欄位 k+3 是新欄位，並存當今分配狀態當最佳狀態。

【0066】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的數量失衡，並因此到步驟 S27。

【0067】 在步驟 S27，判斷回溯次數未達上限，並因此到步驟 S29。

【0068】 在步驟 S29，隨機選列 3，回欄位 k+2，並回步驟 S21。此時回溯次數須加 1。

【0069】 參考圖 8，在步驟 S21，發現欄位 k+2 有 2 個非零元素的讀取是從記憶體 A，2 個非零元素的讀取是從記憶體 B。

【0070】 在步驟 S23，判斷欄位 k+2 是舊欄位，並因此不存當今分配狀態當最佳狀態。

【0071】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的數量平衡，並因此到步驟 S33。

【0072】 在步驟 S33，將非零元素資料依平衡數量且隨機分配的方式寫入記憶體，分配列 1 及 4 的寫到記憶體 A，並分配列 3 及 5 的寫到記憶體 B。

【0073】 在步驟 S35，判斷欄位 k+2 非最後欄位，並因此回步驟 S21。

【0074】 在步驟 S21，發現欄位 k+3 有 3 個非零元素的讀取是從記憶體 A，1 個非零元素的讀取是從記憶體 B。

【0075】 在步驟 S23，判斷欄位 k+3 是舊欄位，並因此不存當今分配狀態當最佳狀態。

【0076】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的數量依舊失衡，並因此到步驟 S27。在此失衡的狀態，須在步驟 S27 判斷是否達回溯次數上限，並根據結果再執行回溯(回溯次數須加 1)或如後述在 S31 執行回復。

【0077】 若一直未能使分配狀態從失衡到平衡，則在某次回溯的步驟 S27 時，判斷回溯次數已達上限，並因此到步驟 S31。在步驟 S31，因無法找到更佳分配狀態，所以採用先前儲存之最佳分配狀態作為回復(在此狀況中為欄位 k+3)。接著進入步驟 S33 執行資料寫入部分，並在步驟 S35 判斷是否處理下一欄位或已處理完畢。

【0078】 參考圖 6 及圖 9，描述回溯 1 次後，從一種失衡的分配狀態到另一種失衡分配狀態的可能狀況，並須進行第 2 次且巢狀式(多次涵蓋式)回溯，並敘述相關後續處置。

【0079】 參考圖 6，在步驟 S21，發現欄位 k+3 有 3 個非零元素的讀取是從記憶體 A，1 個非零元素的讀取是從記憶體 B。

【0080】 在步驟 S23，判斷欄位 k+3 是新欄位，並存當今分配狀態當最佳狀態。

【0081】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的

非零元素的數量失衡，並因此到步驟 S27。

【0082】 在步驟 S27，判斷回溯次數未達上限，並因此到步驟 S29。

【0083】 在步驟 S29，隨機選列 2，回欄位 k+2，並回步驟 S21。此時，回溯次數須加 1。

【0084】 參考圖 9，在步驟 S21，發現欄位 k+1 有 2 個非零元素的讀取是從記憶體 A，2 個非零元素的讀取是從記憶體 B。

【0085】 在步驟 S23，判斷欄位 k+1 是舊欄位，並因此不存當今分配狀態當最佳狀態。

【0086】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的數量平衡，並因此到步驟 S33。

【0087】 在步驟 S33，把非零元素資料依平衡數量且隨機分配的方式寫入記憶體，分配列 3 及 5 的寫到記憶體 A，並分配列 2 及 6 的寫到記憶體 B。

【0088】 在步驟 S35，判斷欄位 k+1 非最後欄位，並因此回步驟 S21。

【0089】 在步驟 S21，發現欄位 k+2 有 3 個非零元素的讀取是從記憶體 A，1 個非零元素的讀取是從記憶體 B。

【0090】 在步驟 S23，判斷欄位 k+2 是舊欄位，並因此不存當今分配狀態當最佳狀態。

【0091】 在步驟 S25，從步驟 S21 的結果，判斷從兩個記憶體讀取的數量依舊失衡，並因此到步驟 S27。在此失衡的狀態，須在步驟 S27 判斷是否達回溯次數上限，並根據結果再執行回溯(回溯次數須加 1)，或在 S31 執行回復。若判斷為可執行回溯，因此時仍在先前未完成之

回溯中，同時再執行回溯，就是巢狀回溯。可重複執行回溯，直到達回溯次數上限，或成功向後續欄位推進(在步驟 S23 判斷現今欄位為新欄位)才結束。

【0092】 分配的第二階段(步驟 S40)是在前一階段已完成的記憶體A(或B)分配情況下，進一步安排所對應之記憶體位址，以檢查記憶體的可用性。參考圖 10，步驟 S40 包括步驟 S41、S43、S45、S47、S49、S51、S53、S55、S57、S59。步驟 S41、S43 屬於讀的階段，步驟 S45、S47、S49 屬於寫的階段，步驟 S53、S55、S57 則是後續處理環繞(wrap around)的階段。

【0093】 步驟 S41，在某一欄位中，依在步驟 S20 決定的記憶體分配方式，從對應的記憶體依序讀取一個非零元素的資料，讀取後將此元素資料從記憶體中移除。

【0094】 在步驟 S43，判斷是否讀完此欄位所有的非零元素。若是，則到步驟 S45 執行安排寫入的步驟，否則回步驟 S41 繼續讀取。

【0095】 在步驟 S45，先判斷目的記憶體是否為滿。若是，則表示記憶體空間不足以實現此分配方法，立即到步驟 S59 結束第二階段，否則到步驟 S47 繼續寫入的步驟。

【0096】 在步驟 S47，依在步驟 S20 決定的記憶體分配方式，依序把上述欄位中的一個非零元素的資料，寫入對應的記憶體中一個可用的位址(宜從可用的空間隨機選擇位址以增加分配多樣性)。

【0097】 在步驟 S49，判斷是否寫完此欄位的非零元素資料。若是，則到步驟 S51，否則回步驟 S45 繼續寫入。

【0098】 在步驟 S51，判斷是否達最後欄位。若是，則到步驟 S53，否則回步驟 S41 處理下一欄位。

【0099】 在步驟 S53，執行環繞(wrap around)，再處理整個矩陣的記憶體讀取與寫入。若寫入時發生衝突(前次指定的寫入位址已被佔)，則從現今可用的記憶體空間，隨機指定一個位址覆蓋(overwrite)之前的分配。此外，此衝突發生時，註記此次環繞曾發生覆蓋。

【00100】 在步驟 S55，判斷在此次環繞時(步驟 S53)是否曾有覆蓋發生。若有，則到步驟 S57 判斷是否須再環繞，否則表示第二階段完成，分配無誤可直接結束至步驟 S60。

【00101】 在步驟 S57，判斷是否達最大環繞次數。若是，則到步驟 S59 準備結束，否則回步驟 S53 再執行環繞。

【00102】 在步驟 S59，表示在第二階段發現記憶體空間無法實現此分配方法，記錄 S20 與 S40 兩階段分配的失敗一次，到步驟 S60 執行後續工作。

【00103】 參考圖 11，舉例描述步驟 S40 的執行。令記憶體 A 及 B 都有 3 個記憶空間。依序處理欄位 1、欄位 2、欄位 3。

【00104】 處理欄位 1 時，暫不執行步驟 S41 及 S43，因無任何記憶位址的資訊。

【00105】 在步驟 S45，判斷預期寫入之記憶體未滿，因此到步驟 S47，把一個非零元素的資料寫入對應的記憶體中一位址。

【00106】 重複執行步驟 S45、S47 及 S49，直到把欄位 1 元素對應資料皆寫入記憶體為止。參考圖 11 下方結果，列 1 之資料被寫到記憶

體 A 的位址 1，列 2 之資料被寫到記憶體 A 的位址 2，列 3 之資料被寫到記憶體 B 的位址 1，列 5 之資料被寫到記憶體 B 的位址 2。

【00107】 在步驟 S51，判斷欄位 1 非最後欄位，並回步驟 S41。

【00108】 處理欄位 2 時，重複執行步驟 S41 及 S43，直到讀完欄位 2(列 4 暫不執行步驟 S41 及 S43，因無任何記憶位址的資訊)。參考圖 11 下方結果，列 1 從記憶體 A 的位址 1 讀，列 3 從記憶體 B 的位址 1 讀，列 5 從記憶體 B 的位址 2 讀。讀取時還把資料從相對應的記憶體位址移除，此時剩記憶體 A 的位址 2 仍有資料。

【00109】 在步驟 S45，判斷預期寫入之記憶體未滿，並因此到步驟 S47，把一個非零元素的資料寫入對應的記憶體中一位址。

【00110】 重複執行步驟 S45、S47 及 S49，直到把欄位 2 元素對應資料皆寫入記憶體為止。參考圖 11 下方結果，列 1 之資料被寫到記憶體 A 的位址 1，列 3 之資料被寫到記憶體 A 的位址 3(因位址 2 已有資料)，列 4 之資料被寫到記憶體 B 的位址 1，列 5 之資料被寫到記憶體 B 的位址 2。若記憶體 A 及 B 都只有 2 個記憶空間，則在寫列 3 以前在步驟 S45 會判斷記憶體 A 已滿而無法執行寫入，直接步驟 S59 記錄錯誤與結束排序。

【00111】 在步驟 S51，判斷欄位 2 非最後欄位，並回步驟 S41。

【00112】 處理欄位 3 時，重複執行步驟 S41 及 S43，直到讀完欄位 2。參考圖 11 下方結果，列 1 從記憶體 A 的位址 1 讀，列 2 從記憶體 A 的位址 2 讀，列 4 從記憶體 B 的位址 2 讀，列 5 從記憶體 B 的位址 2 讀。讀取時還把資料從相對應的記憶體位址移除，此時剩記憶體 A 的

位址 3 仍有資料。

【00113】 重複執行步驟 S45、S47 及 S49，先判斷對應記憶體空間再寫入，直到寫完欄位 3 資料為止。參考圖 11 下方結果，列 1 之資料被寫到記憶體 A 的位址 1，列 2 之資料被寫到記憶體 B 的位址 1，列 4 之資料被寫到記憶體 A 的位址 2，列 5 之資料被寫到記憶體 B 的位址 2。

【00114】 在步驟 S51，判斷欄位 3 是最後欄位，並因此到步驟 S53。

【00115】 參考圖 12，在步驟 S53，執行第一次環繞，就是返回開頭重新檢查整個矩陣、填補缺少的資訊、並處理任何覆蓋的情形。

【00116】 檢驗欄位 1 的讀取。列 1 從記憶體 A 的位址 1 讀，列 2 從記憶體 B 的位址 1 讀，列 3 從記憶體 A 的位址 3 讀(因圖 11 中列 3 最後是寫入記憶體 A 的位址 3)，列 5 從記憶體 B 的位址 2 讀。讀取時還把資料從相對應的記憶體位址移除，此時剩記憶體 A 的位址 2 仍有資料。

【00117】 接著，檢驗欄位 1 的寫入。列 1 之資料被寫到記憶體 A 的位址 1，列 2 之資料發生衝突無法寫到記憶體 A 的位址 2，列 3 之資料被寫到記憶體 B 的位址 1，列 5 之資料被寫到記憶體 B 的位址 2。注意，因列 2 之資料原指定之記憶體 A 位址 2 被佔，故寫入位址 3。發生覆蓋情形，予以紀錄。

【00118】 檢驗欄位 2 的讀取。列 1 從記憶體 A 的位址 1 讀，列 3 從記憶體 B 的位址 1 讀，列 4 從記憶體 A 的位址 2 讀，列 5 從記憶體 B 的位址 2 讀。讀取時還把資料從相對應的記憶體位址移除，此時剩記憶體 A 的位址 3 仍有資料。

【00119】 接著，檢驗欄位 2 的寫入。列 1 之資料被寫到記憶體 A 的位址 1，列 3 之資料被寫到記憶體 A 的位址 2，列 4 之資料被寫到記憶體 B 的位址 1，列 5 之資料被寫到記憶體 B 的位址 2。

【00120】 檢驗欄位 3 的讀取。列 1 從記憶體 A 的位址 1 讀，列 3 從記憶體 A 的位址 3 讀，列 4 從記憶體 B 的位址 1 讀，列 5 從記憶體 B 的位址 2 讀。讀取時還把資料從相對應的記憶體位址移除，此時剩記憶體 A 的位址 2 仍有資料。

【00121】 接著，檢驗欄位 3 的寫入。列 1 之資料被寫到記憶體 A 的位址 1，列 2 之資料被寫到記憶體 B 的位址 1，列 4 之資料發生衝突而無法寫到記憶體 A 的位址 2，列 5 之資料被寫到記憶體 B 的位址 2。注意，列 4 之資料因原指定之記憶體 A 位址 2 被佔，故寫入位址 3。發生覆蓋情形，予以紀錄。第 1 次環繞結束，離開步驟 S53。

【00122】 在步驟 S55，由上述紀錄判斷此次環繞有覆蓋發生，並到步驟 S57。

【00123】 在步驟 S57，令最大環繞次數為 10，當今環繞次數為 1，其數目小於 10，因此回步驟 S53 再環繞。

【00124】 參考圖 13，在步驟 S53，執行第二次環繞，就是返回開頭重新檢查整個矩陣、填補缺少的資訊、並處理任何覆蓋的情形。

【00125】 與第一次環繞作法相同，依序檢驗欄位 1 的讀跟寫，欄位 2 的讀跟寫，欄位 3 的讀跟寫。注意，第 1 欄列 2 寫資料時，因原指定之記憶體 A 位址 3 被佔，故寫入位址 2。發生覆蓋情形，予以紀錄。第 3 欄列 4 寫入資料時，因原指定之記憶體 A 位址 3 被佔，故寫

入位址 2。發生覆蓋情形，予以紀錄。第 2 次環繞結束，離開步驟 S53。

【00126】 在步驟 S55，由上述紀錄判斷此次環繞依舊有覆蓋發生，並到步驟 S57。

【00127】 在步驟 S57，因最大環繞次數為 10，當今環繞次數為 2，其數目小於 10，因此回步驟 S53 再環繞。

【00128】 在此例，因執行步驟 S53 環繞時，第 1 欄列 2 之寫入與第 3 欄列 4 之寫入，會不斷在記憶體 A 的位址 2 與位址 3 切換，故每次環繞時皆會導致覆蓋情況發生，並經步驟 S55 到步驟 S57 判斷環繞次數上限，然後回步驟 S53 執行下次環繞。最後一次環繞在步驟 S57 因為達到環繞次數 10 次的上限，到步驟 S59 記錄錯誤並結束第二階段排序。

【00129】 在以上描述，每一個記憶體都有 3 個記憶空間。在以下描述，每一個記憶體都有 4 個記憶空間。記憶體的空間愈多，發生覆蓋的可能性愈低，因而更有機會成功找到排序。

【00130】 參考圖 14，舉例描述在圖 11 排列後，使用 4 個記憶空間的記憶體(記憶體 A 及 B 都各有 4 個記憶空間)來做環繞，並在環繞時執行檢驗與覆蓋，最後成功完成排序的過程。環繞時，依序處理欄位 1、欄位 2、欄位 3。

【00131】 檢驗欄位 1 的讀取。列 1 從記憶體 A 的位址 1 讀，列 2 從記憶體 B 的位址 1 讀，列 3 從記憶體 A 的位址 3 讀(因圖 11 中列 3 最後是寫入記憶體 A 的位址 3)，列 5 從記憶體 B 的位址 2 讀。讀取時還把資料從相對應的記憶體位址移除，此時剩記憶體 A 的位址 2 仍有資

料。

【00132】 接著，檢驗欄位 1 的寫入。列 1 之資料被寫到記憶體 A 的位址 1，列 2 之資料發生衝突無法寫到記憶體 A 的位址 2，列 3 之資料被寫到記憶體 B 的位址 1，列 5 之資料被寫到記憶體 B 的位址 2。注意，列 2 之資料因原指定之記憶體 A 位址 2 被佔，故依隨機方式選擇現今可用之空間，此處假設隨機指定結果為使用記憶體 A 之位址 4。由於列 2 在此發生覆蓋情形，予以紀錄。

【00133】 檢驗欄位 2 的讀取。列 1 從記憶體 A 的位址 1 讀，列 3 從記憶體 B 的位址 1 讀，列 4 從記憶體 A 的位址 2 讀，列 5 從記憶體 B 的位址 2 讀。讀取時還把資料從相對應的記憶體位址移除，此時剩記憶體 A 的位址 4 仍有資料。

【00134】 接著，檢驗欄位 2 的寫入。列 1 之資料被寫到記憶體 A 的位址 1，列 3 之資料被寫到記憶體 A 的位址 3，列 4 之資料被寫到記憶體 B 的位址 1，列 5 之資料被寫到記憶體 B 的位址 2。

【00135】 檢驗欄位 3 的讀取。列 1 從記憶體 A 的位址 1 讀，列 3 從記憶體 A 的位址 4 讀，列 4 從記憶體 B 的位址 1 讀，列 5 從記憶體 B 的位址 2 讀。讀取時還把資料從相對應的記憶體位址移除，此時剩記憶體 A 的位址 3 仍有資料。

【00136】 接著，檢驗列 3 的寫入。列 1 之資料被寫到記憶體 A 的位址 1，列 2 之資料被寫到記憶體 B 的位址 1，列 4 之資料被寫到記憶體 A 的位址 2，列 5 之資料被寫到記憶體 B 的位址 2。第 1 次環繞結束，離開步驟 S53。

【00137】 在步驟 S55，由上述紀錄判斷此次環繞有覆蓋發生，並到步驟 S57。

【00138】 在步驟 S57，假定最大環繞次數為 10，當今環繞次數為 1，其數目小於 10，因此回步驟 S53 再次環繞。

【00139】 在步驟 S53 執行第二次環繞，用第一次環繞結果進行分配檢驗，確認分配方式無變動，因此任何覆蓋紀錄。

【00140】 在步驟 S55 判斷無覆蓋，分配成功，並因此離開此第二階段排序。

【00141】 以上僅為描述本發明的較佳實施方式，非用以限定本發明的範圍。本技術領域內的一般技術人員根據上述實施例所作的均等變化，以及本領域內技術人員熟知的改變，仍在本發明的範圍內。

【符號說明】

【00142】

S20：分配的第一階段

S21：累計某欄位中分別從兩個記憶體來的非零元素數量

S23：暫存當今整體分配狀態當最佳分配狀態

S25：從兩個記憶體讀取的數量平衡？

S27：回溯次數達上限？

S29：執行回溯

S31：回復先前暫存之最佳狀況

S33：依平衡寫入條件隨機分配給記憶體

S35：已達矩陣的最後欄位？

S37：處理開頭欄位中未確認的記憶體讀取資訊

S39：每一欄位的讀取數量皆平衡或已達最大遞迴數？

S40：分配的第二階段

S41：依記憶體分配方式，從對應的記憶體依序讀取非零元素

S43：讀完此欄位所有的非零元素？

S45：目的記憶體滿？

S47：依記憶體分配方式，把一個非零元素寫入對應的記憶體

S49：寫完此欄位的非零元素？

S51：達最後欄位？

S53：執行環繞

S55：在此次環繞時有覆蓋發生？

S57：達最大環繞次數？

S59：記錄失敗一次

S60：更新最佳結果

S70：判斷是否達最大遞迴數

S80：判斷分配是否成功

S90：產生重組解碼器的指令檔案

【生物材料寄存】

國內寄存資訊【請依寄存機構、日期、號碼順序註記】

無

國外寄存資訊【請依寄存國家、機構、日期、號碼順序註記】

無

【序列表】(請換頁單獨記載)

無

申請專利範圍

【請求項1】 一種重排 LDPC 碼而改善 LDPC 重組解碼器並行的方法，包括下列步驟：

為上述 LDPC 重組解碼器提供兩個記憶體；

平均分配上述 LDPC 碼的矩陣的非零元素給上述兩個記憶體(S20)；

依上述分配方式，寫上述非零元素到上述兩個記憶體(S40)；

更新分配方式(S60)；

判斷是否達最大遞迴數(S70)；

若未達最大遞迴數，則回上述平均分配上述非零元素給上述兩個記憶體的步驟(S20)；

若達最大遞迴數，則判斷分配是否成功(S80)；

若分配成功，則產生排程檔案(S90)；及

若分配失敗，則結束。

【請求項2】 如請求項 1 所述之重排 LDPC 碼而改善 LDPC 重組解碼器並行的方法，其中達最大遞迴數，或每一欄位的資料讀取皆已平衡後，還包括以下步驟：

依上述分配方式，在某一欄位中，從對應的記憶體依序讀取一個非零元素，讀取後將此元素從記憶體中移除(S41)；

判斷是否讀完此欄位所有的非零元素資料(S43)；

若未讀完此欄位的非零元素資料，則回上述從對應的記憶體讀某一欄位中一個非零元素資料的步驟(S41)；

若讀完此欄位的所有非零元素資料，則判斷預期寫入記憶體是否滿(S45)；

若任何上述記憶體滿，則記錄分配失敗一次並結束(S59)；

若預期寫入記憶體未滿，則依上述分配方式，依序把上述欄位中的一個非零元素資料，寫入對應的記憶體中一個可用的位址(S47)；

判斷是否寫完此欄位的非零元素資料(S49)；

若未寫完此欄位的所有非零元素資料，則回上述判斷任何上述記憶體是否滿的步驟(S45)；

若寫完此欄位的所有非零元素資料，則判斷是否達最後欄位(S51)；

若未達最後欄位，則回上述從對應的記憶體依序讀某一欄位一個非零元素的步驟(S41)；

若達最後欄位則執行環繞，檢驗整個矩陣是否有位址衝突需重新分配(S53)；

判斷此次環繞時有無覆蓋(S55)；

若此次環繞時無覆蓋，則結束；

若此次環繞時有覆蓋，則判斷環繞次數是否達一個上限(S57)；

若環繞次數未達上述上限，則回上述執行環繞的步驟(S53)；及

若環繞次數達上述上限，則記錄分配失敗一次並結束(S59)。

【請求項3】 一種改善 LDPC 重組解碼器並行的方法，其包括下列步驟：

為上述 LDPC 重組解碼器提供兩個記憶體；

計算上述 LDPC 碼的矩陣的一個欄位中非零元素資料，分別從上述兩個記憶體讀取之數量(S21)；

若當今欄位是新欄位，則存當今整體分配狀態當最佳狀態(S23)；

判斷此欄位中分屬於上述兩個記憶體的非零元素資料讀取量是否相等(S25)；

若分屬於上述兩個記憶體的非零元素資料讀取量不相等，則判斷回溯次數是否達上限(S27)；

若回溯次數未達上限，則隨機選一個造成失衡狀況的非零元素，回到同一列中前一個非零元素，並重新處理它所在的那一個欄位(S29)；

若回溯次數達上限，則回復先前最佳狀況(S31)；

若分屬於上述兩個記憶體的非零元素量相等，則平均且隨機地分配上述非零元素的資料寫入到上述兩個記憶體(S33)；

判斷是否達最後欄位(S35)；

若未達最後欄位，則回上述計算下個欄位中非零元素資料，分別從上述兩個記憶體讀取之數量(S21)；

若達最後欄位，則利用最後幾個欄位的寫入資訊，得知資料所存放之記憶體，再回到開頭欄位中填入其相對應讀取的記憶體，

並記錄前面每個欄位中分別從兩個記憶體讀取之數量(S37)；

判斷是否達最大遞迴數，或每一欄位的資料讀取皆已平衡(S39)；

若未達最大遞迴數，且每一欄位的資料讀取皆未平衡，則回上述計算LDPC碼的矩陣的一個欄位中非零元素資料，分別從兩個記憶體讀取之數量(S21)；及

若達最大遞迴數，或每一欄位的資料讀取皆已平衡，則結束。

【請求項4】 如請求項3所述之改善LDPC重組解碼器並行的方法，其中達最大遞迴數，或每一欄位的資料讀取皆已平衡後，還包括以下步驟：

依上述分配方式，在某一欄位中，從對應的記憶體依序讀取一個非零元素，讀取後將此元素從記憶體中移除(S41)；

判斷是否讀完此欄位所有的非零元素資料(S43)；

若未讀完此欄位的非零元素資料，則回上述從對應的記憶體讀某一欄位中一個非零元素資料的步驟(S41)；

若讀完此欄位的所有非零元素資料，則判斷預期寫入記憶體是否滿(S45)；

若任何上述記憶體滿，則記錄分配失敗一次並結束(S59)；

若預期寫入記憶體未滿，則依上述分配方式，依序把上述欄位中的一個非零元素資料，寫入對應的記憶體中一個可用的位址(S47)；

判斷是否寫完此欄位的非零元素資料(S49)；

若未寫完此欄位的所有非零元素資料，則回上述判斷任何上述記憶體是否滿的步驟(S45)；

若寫完此欄位的所有非零元素資料，則判斷是否達最後欄位(S51)；

若未達最後欄位，則回上述從對應的記憶體依序讀某一欄位一個非零元素的步驟(S41)；

若達最後欄位則執行環繞，檢驗整個矩陣是否有位址衝突需重新分配(S53)；

判斷此次環繞時有無覆蓋(S55)；

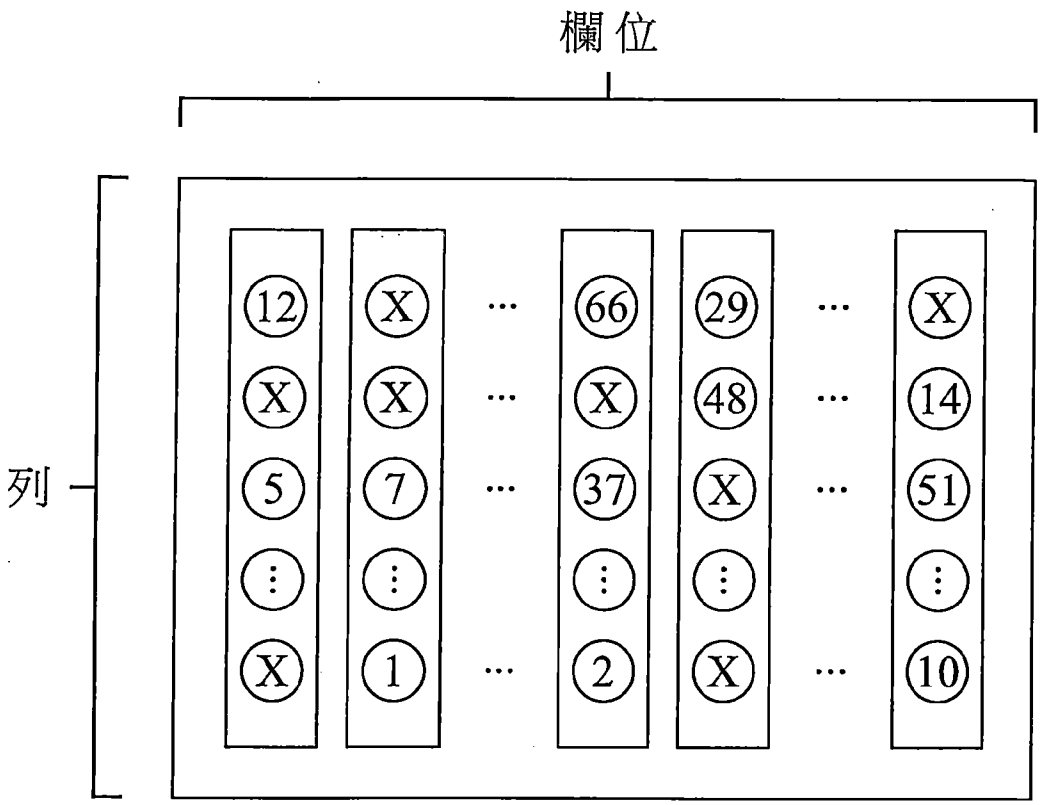
若此次環繞時無覆蓋，則結束；

若此次環繞時有覆蓋，則判斷環繞次數是否達一個上限(S57)；

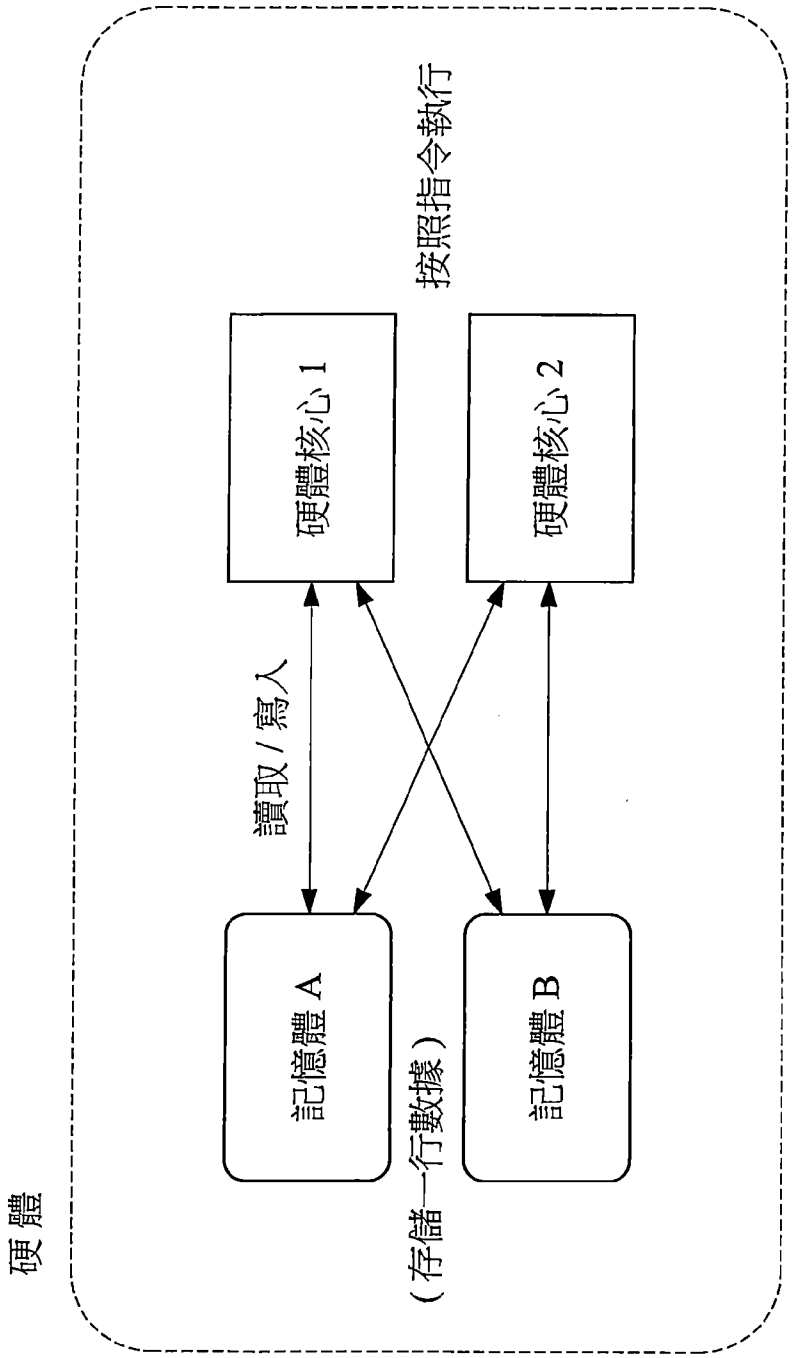
若環繞次數未達上述上限，則回上述執行環繞的步驟(S53)；及

若環繞次數達上述上限，則記錄分配失敗一次並結束(S59)。

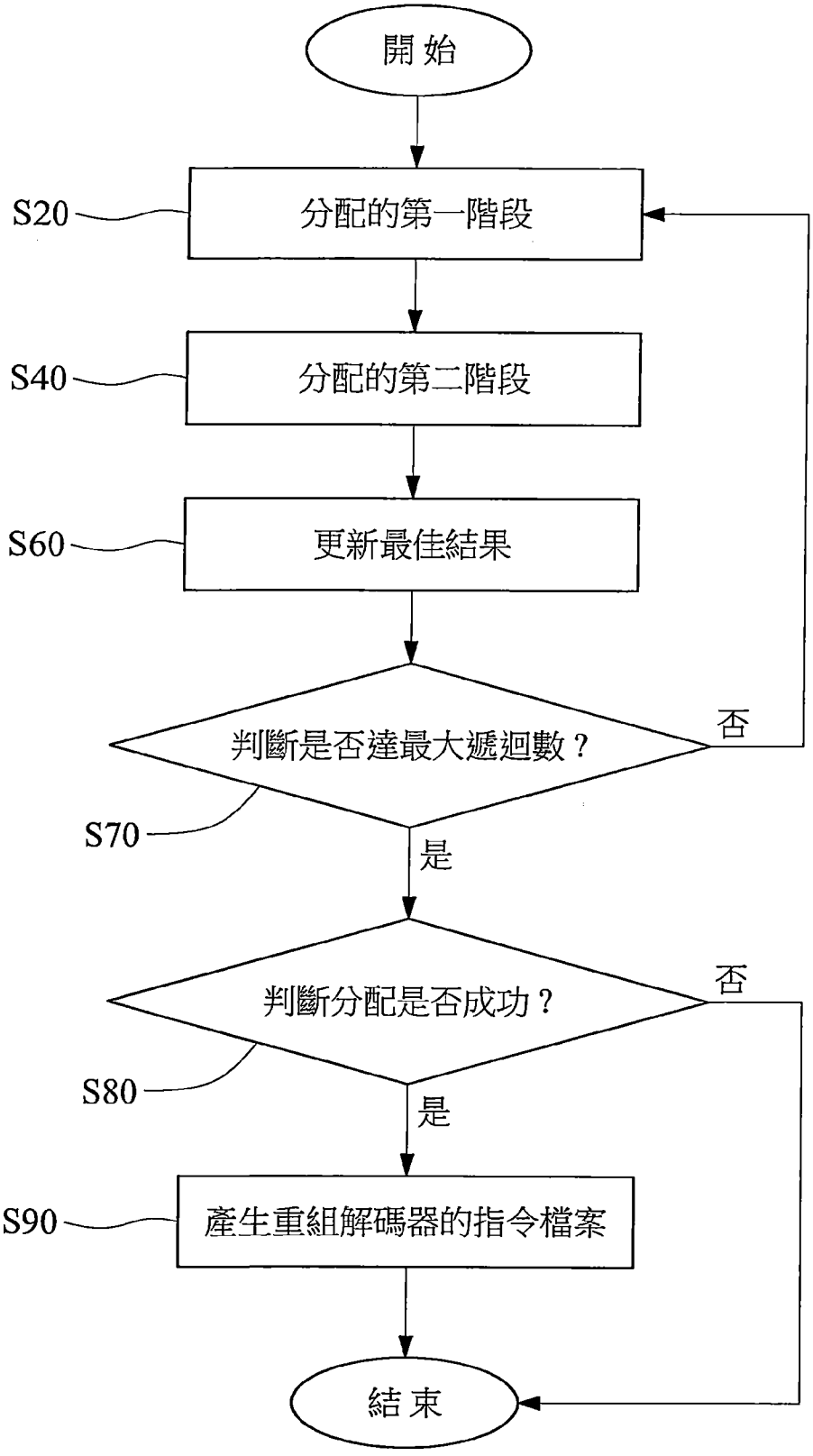
圖式



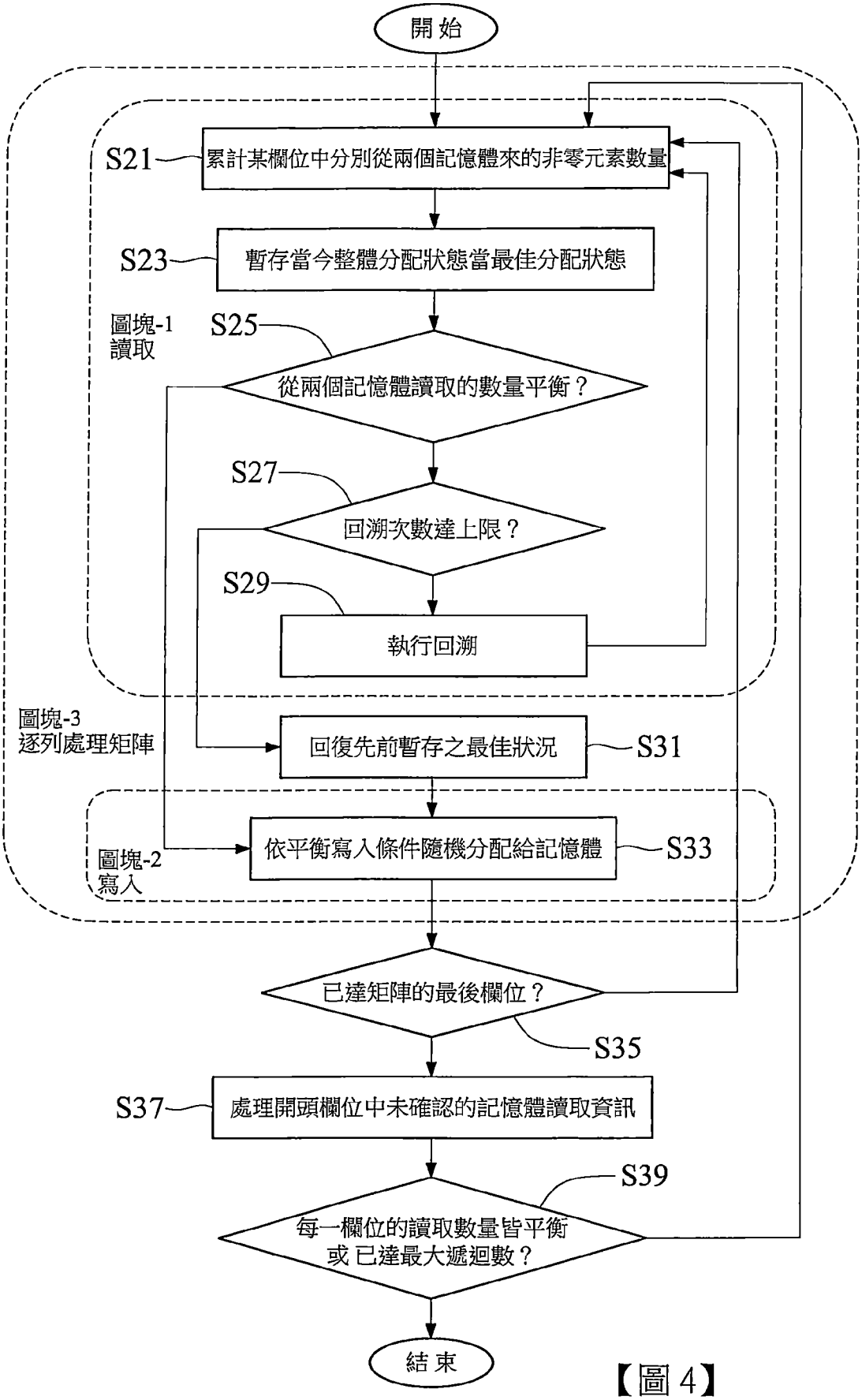
【圖 1】



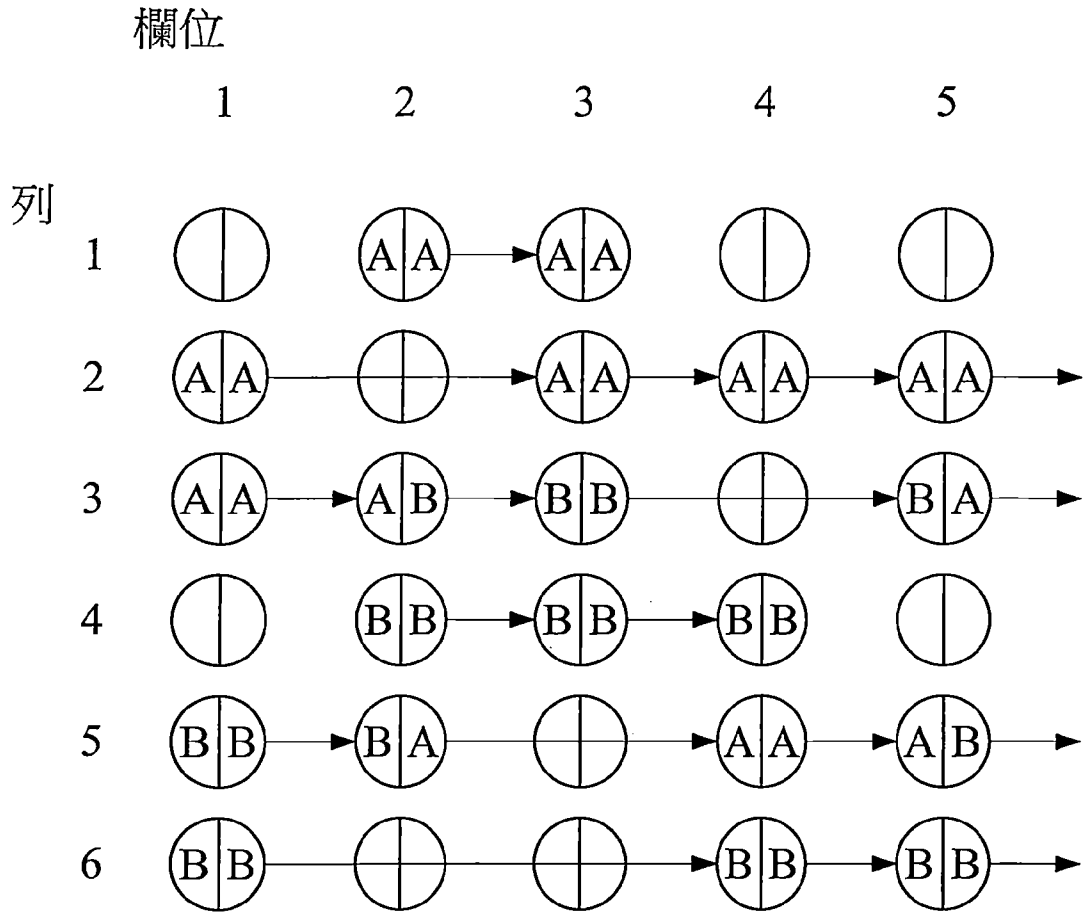
【圖 2】


開始
S20
分配的第一階段
S40
分配的第二階段
S60
更新最佳結果
判斷是否達最大遞迴數？
否
S70
是
判斷分配是否成功？
否
S80
是
S90
產生重組解碼器的指令檔案
結束

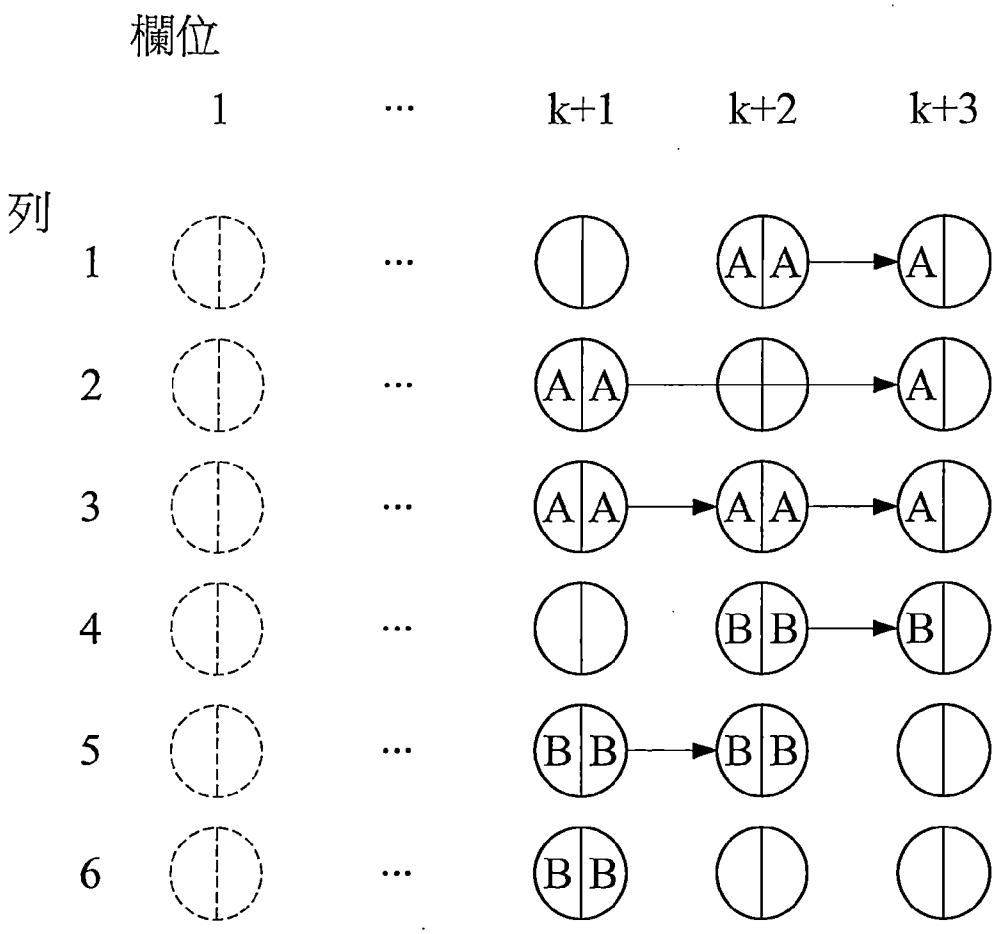
【圖 3】


開始
S21
累計某欄位中分別從兩個記憶體來的非零元素數量
S23
暫存當今整體分配狀態當最佳分配狀態
圖塊-1
讀取
S25
從兩個記憶體讀取的數量平衡？
S27
回溯次數達上限？
S29
執行回溯
圖塊-3
逐列處理矩陣
回復先前暫存之最佳狀況
S31
圖塊-2
寫入
依平衡寫入條件隨機分配給記憶體
S33
已達矩陣的最後欄位？
S35
S37
處理開頭欄位中未確認的記憶體讀取資訊
S39
每一欄位的讀取數量皆平衡
或 已達最大遞迴數？
結束

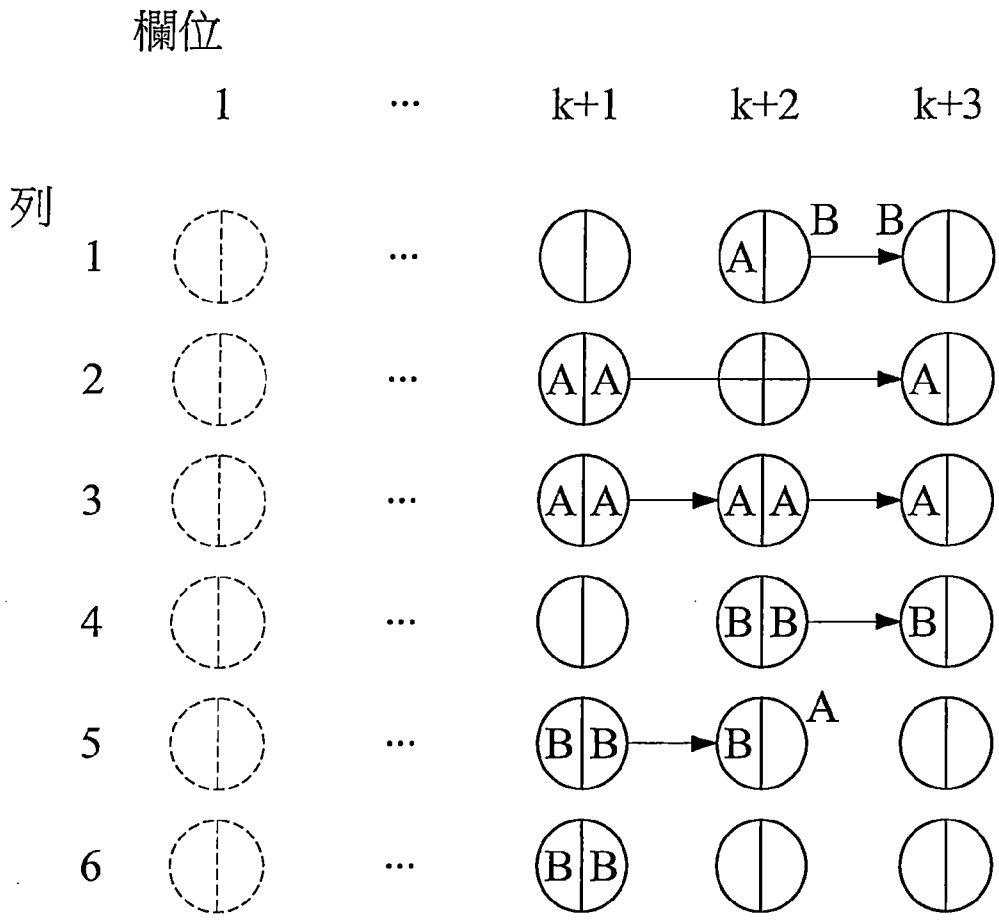
【圖 4】



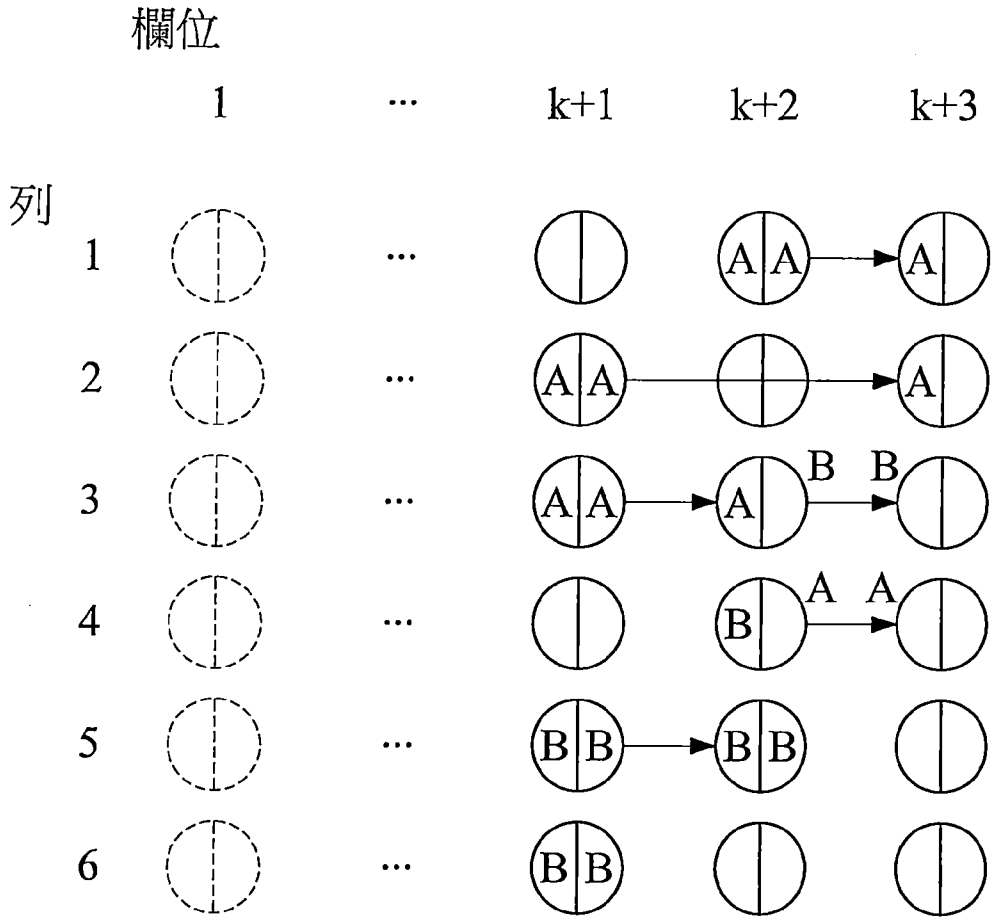
【圖 5】



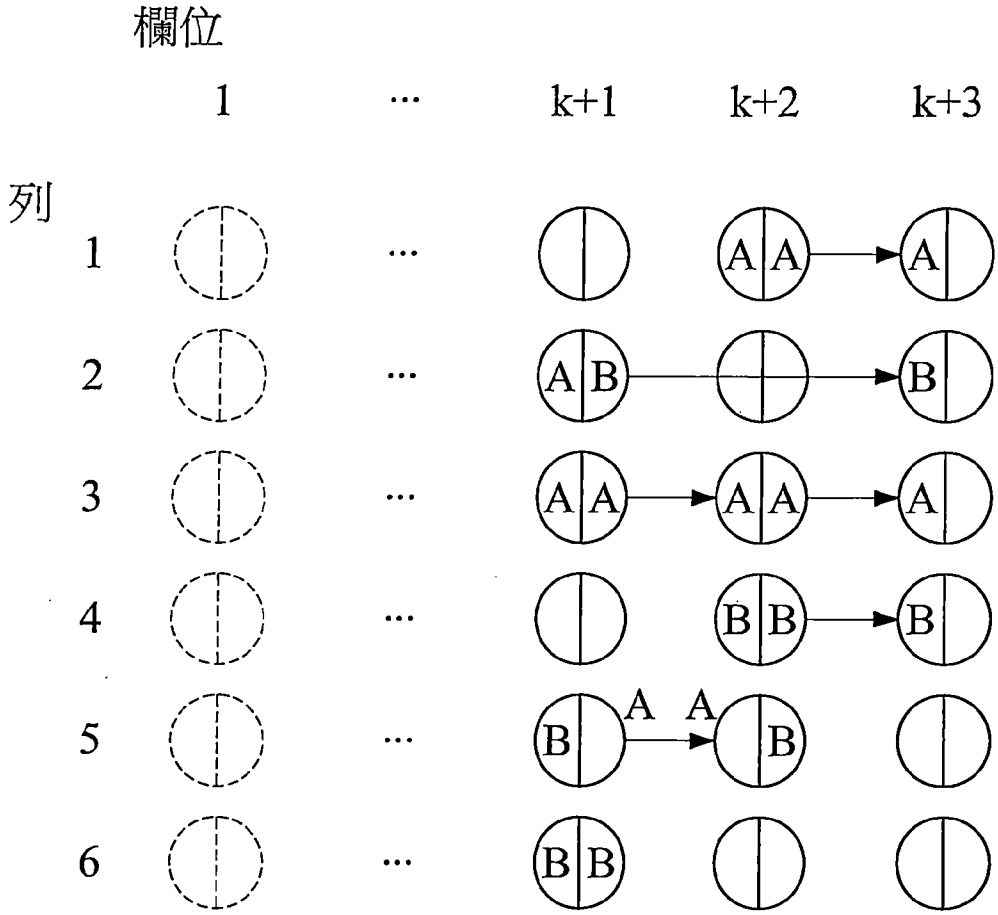
【圖 6】



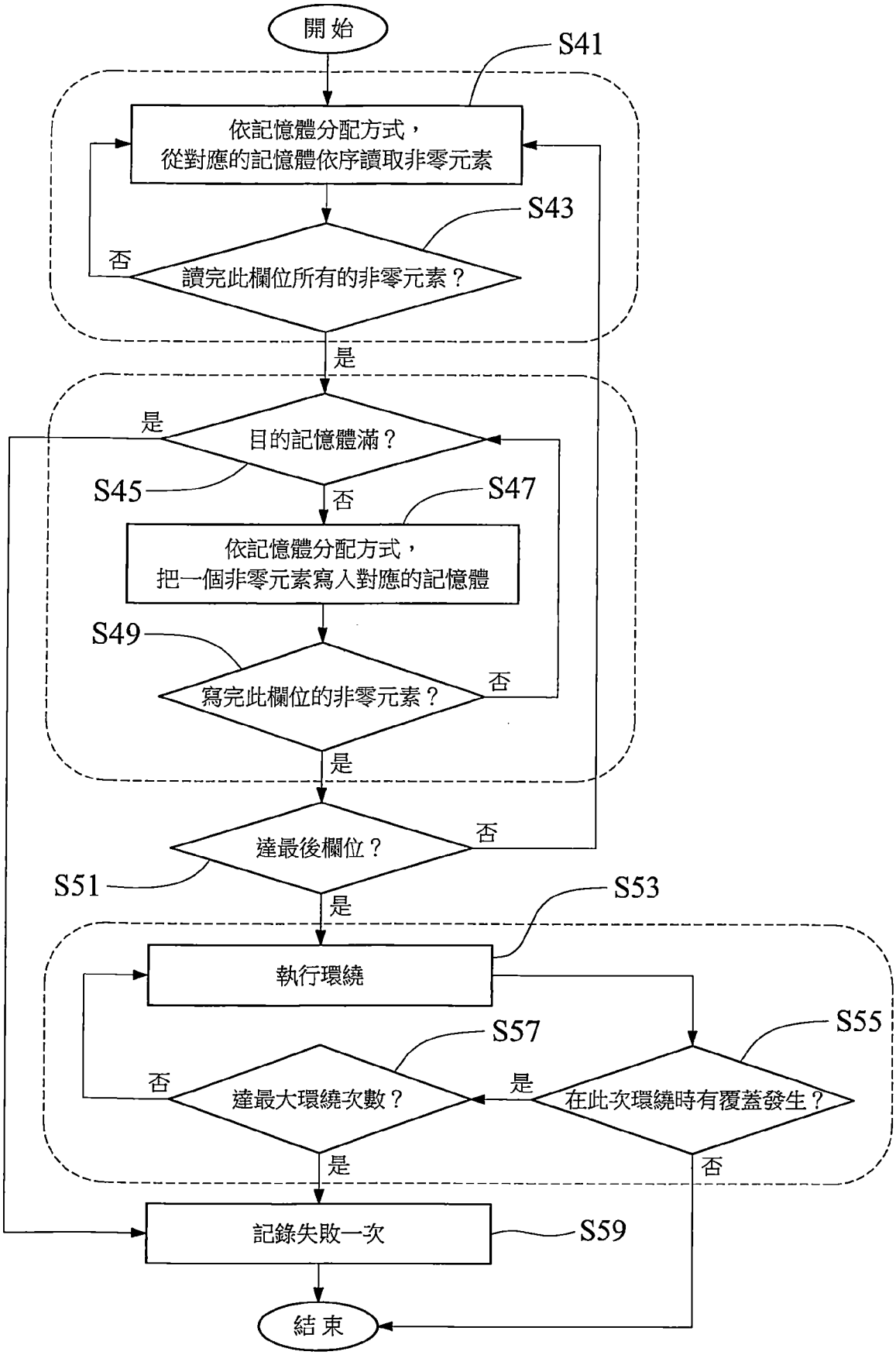
【圖 7】



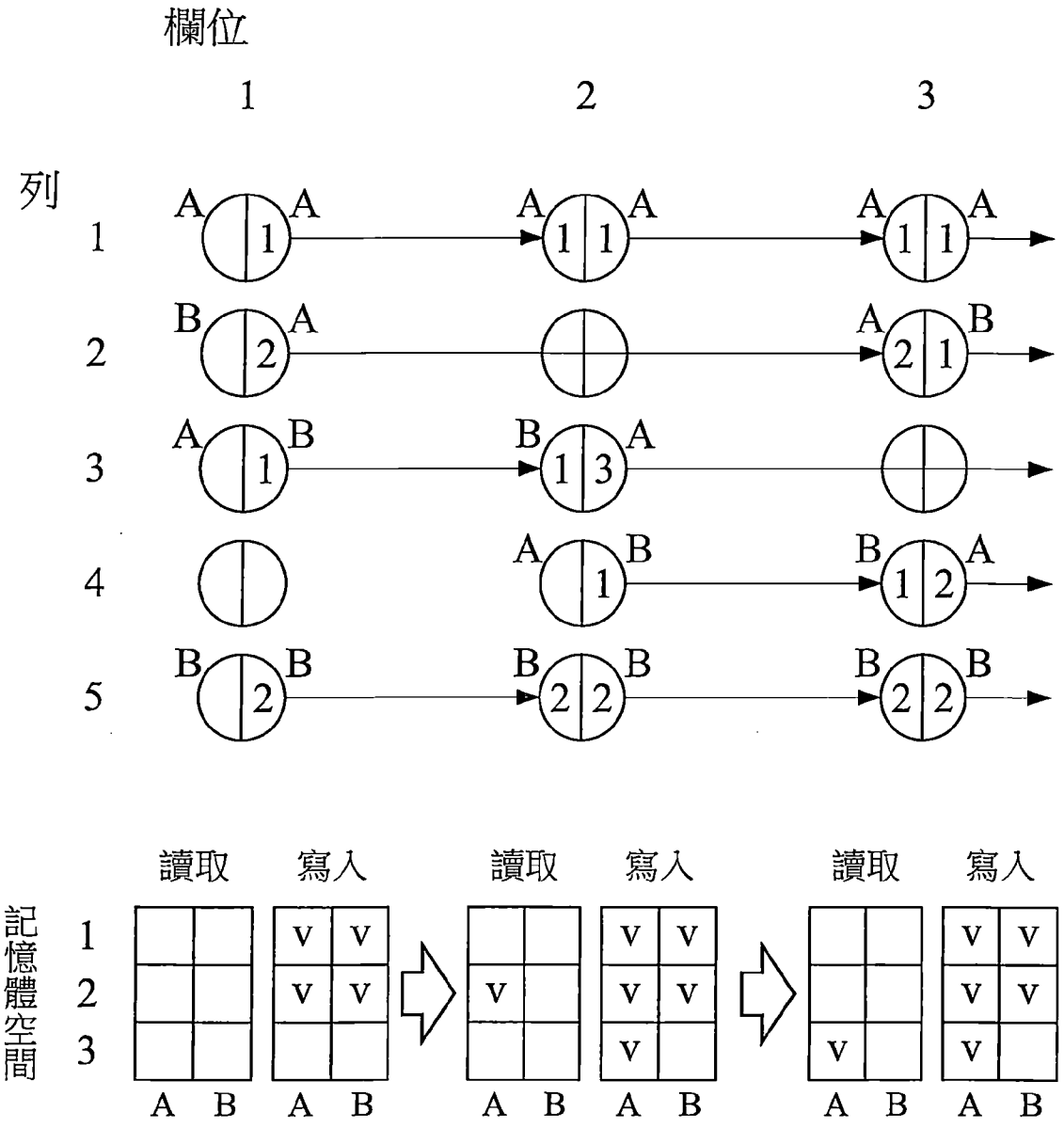
【圖 8】



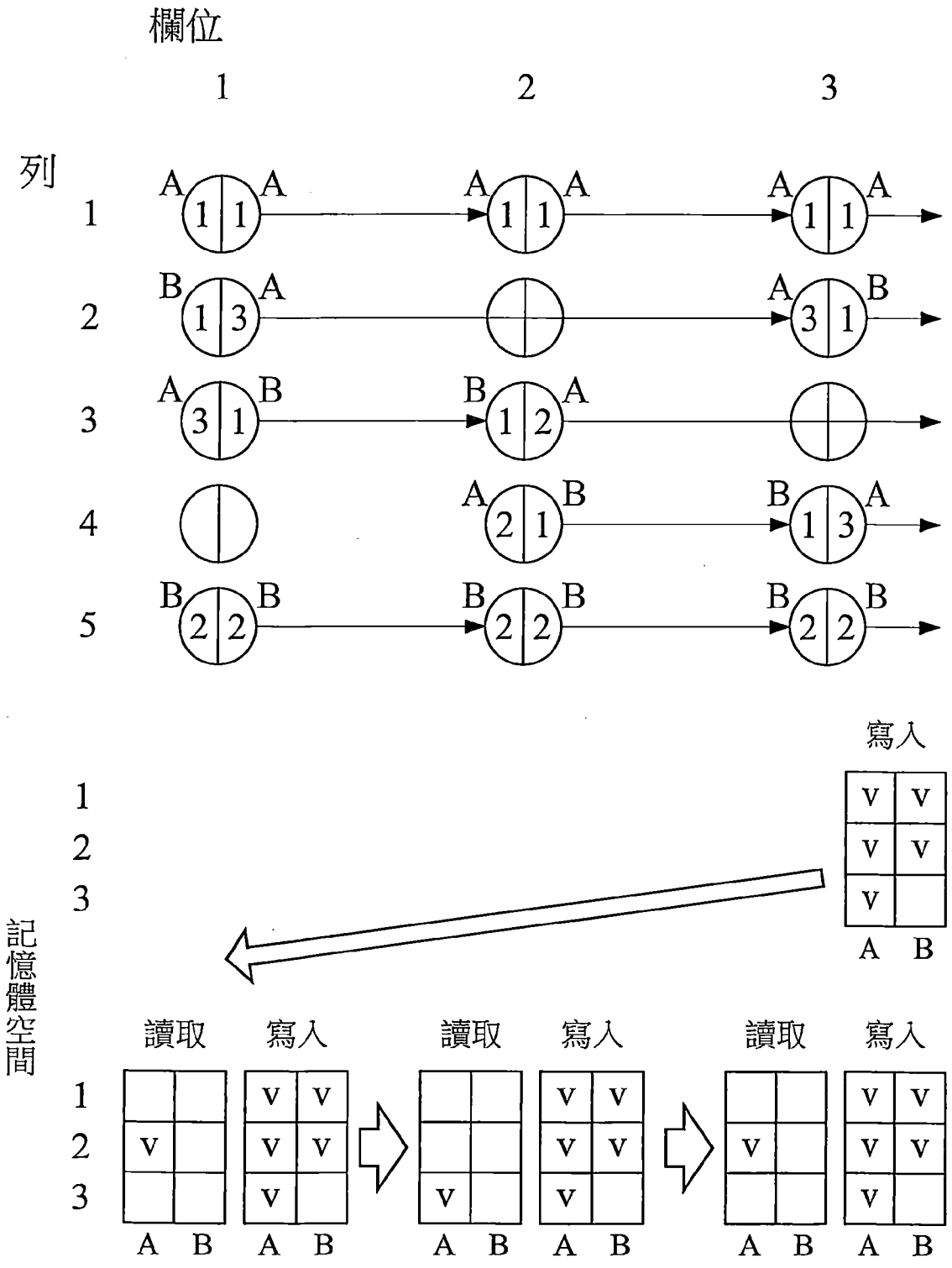
【圖 9】


開始
S41
依記憶體分配方式，
從對應的記憶體依序讀取非零元素
S43
否
讀完此欄位所有的非零元素？
是
是
目的記憶體滿？
S45
否
S47
依記憶體分配方式，
把一個非零元素寫入對應的記憶體
S49
寫完此欄位的非零元素？
否
是
達最後欄位？
否
S51
是
S53
執行環繞
S57
S55
否
達最大環繞次數？
是
在此次環繞時有覆蓋發生？
是
否
記錄失敗一次
S59
結束

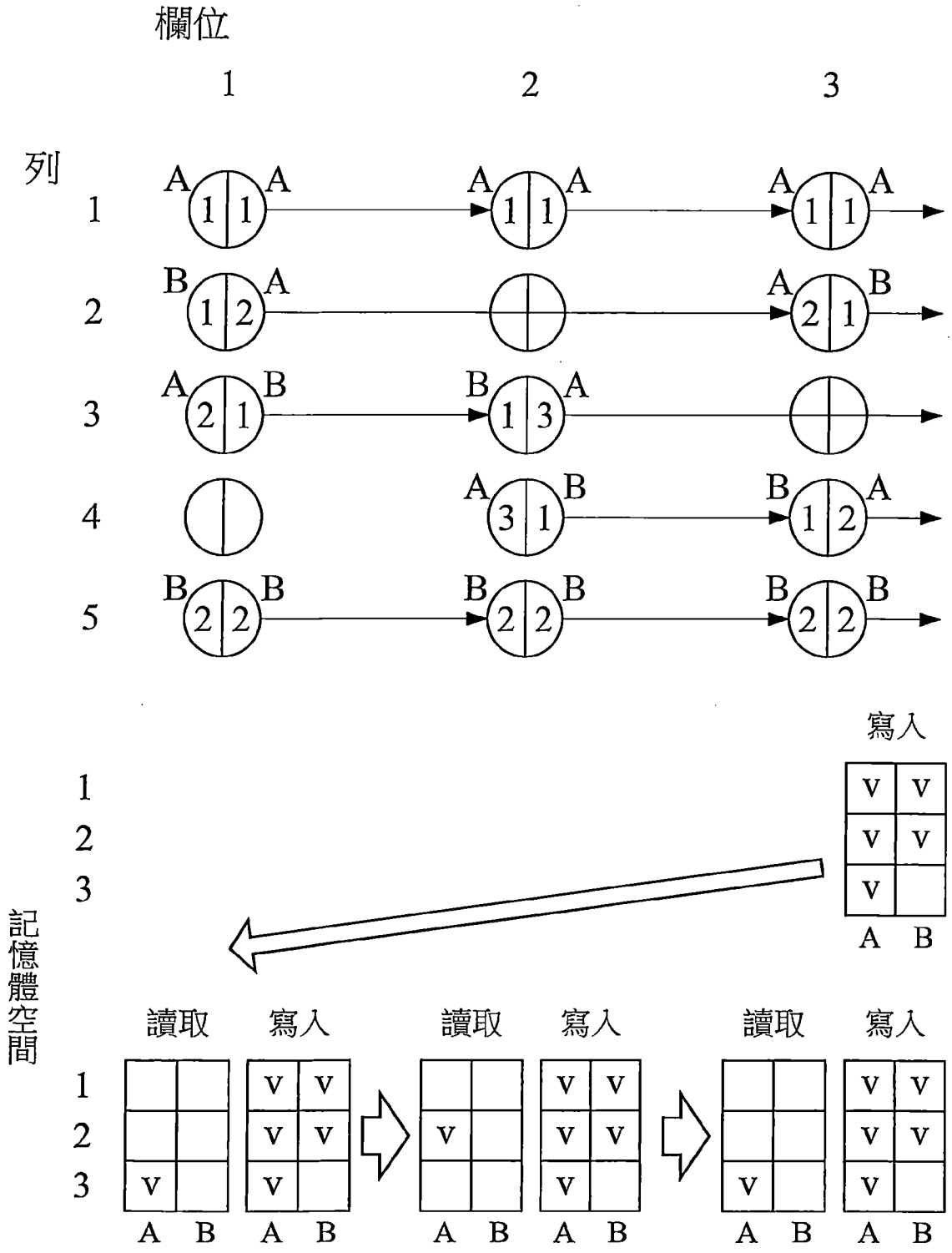
【圖 10】



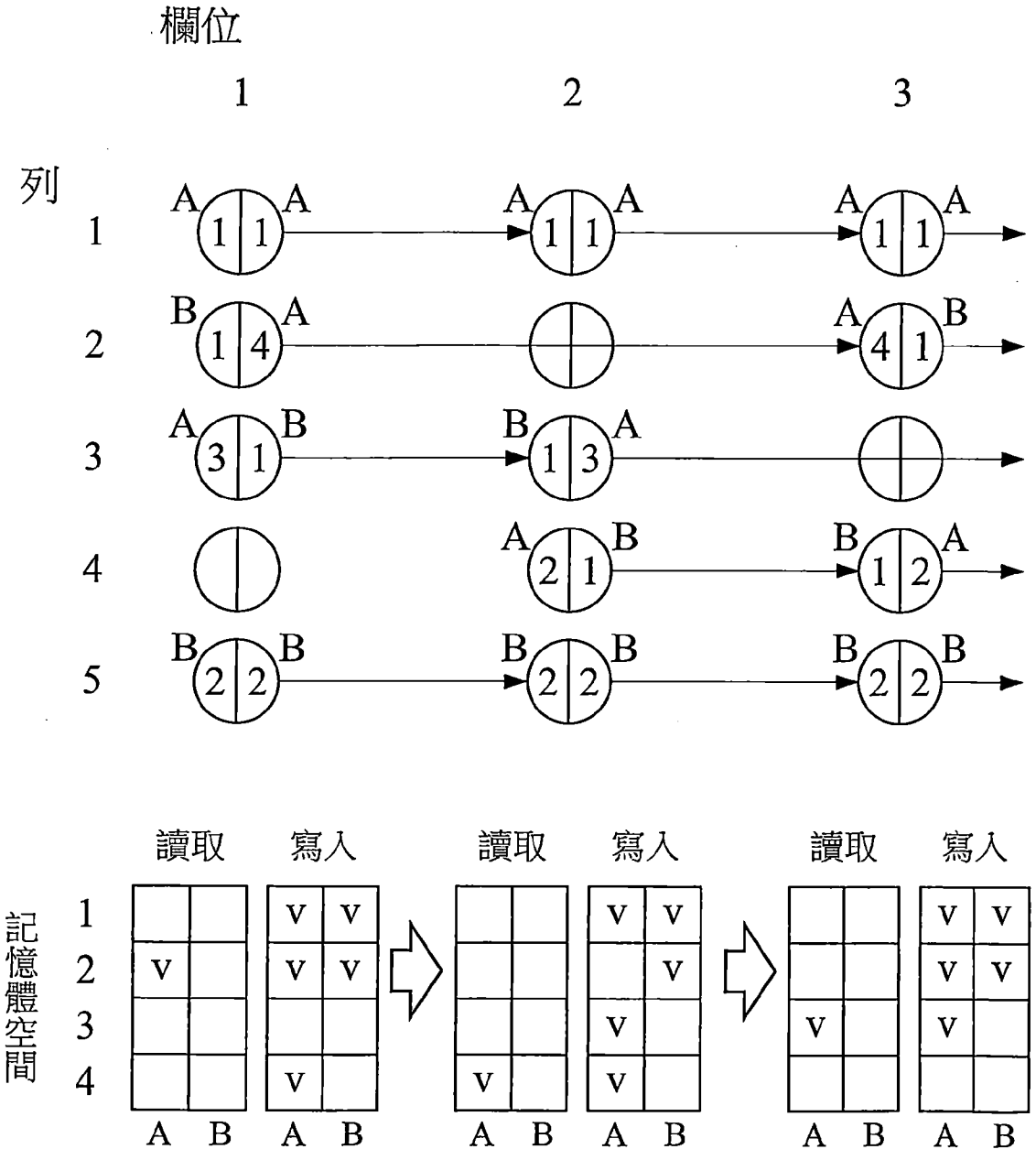
【圖 11】



【圖 12】



【圖 13】



【圖 14】