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）

用於改善路線規劃計算裝置之裝置及方法

DEVICE AND METHOD FOR IMPROVING ROUTE PLANNING COMPUTING DEVICES

【技術領域】

【0001】 本揭示內容大致上有關路線規劃裝置及操作該裝置的方法。本揭示內容更明確地是有關路線規劃裝置中之計算操作中的改善，而增加其速度及效率。

【先前技術】

【0002】 路線規劃裝置、諸如那些採用全球定位系統(GPS)者能被使用於規劃由一地點至另一地點之路線。A*(“A星號”)係被寬廣地使用於路徑搜尋及圖的遍歷之計算機演算法，用於繪製被稱為節點的多數個點間之有效率的能橫越路徑。A*係有情報根據之搜尋演算法、或最佳優先搜尋，意指其藉由搜尋在對用於招致該最小成本(最少行進距離、最短時間等)者之解決方法(目標)的所有可能路徑之中、及在其首先考量顯現為最快速導致該解決方法者的這些路徑之中來解決問題。其被以加權圖的觀點來公式化：由曲線圖之特定節點開始，其建構由該節點開始的樹狀路徑，每次擴展路徑一步進，直至其路徑之其中一者終止在該預定的目標節點。

【0003】 當路線規劃被限制至道路時，用於行進之有限選擇(有

限的節點組)能允許A*在使用者可接收之時間跨度中計算路徑、即使長路徑。然而，經由A*的路線規劃可為耗費大量時間的，且用於即時選路、尤其用於越野式行進偶而太慢。當用於行進之潛在區域(節點)不被限制於道路(諸如步行或藉由休旅車或以別的方式之越野式行進)，節點中的增加能充滿A*，以增加抵達該輸出路徑所需要之重要時間。

【0004】 又再者，藉由選路裝置的處理器之A*指令的解譯在該選路裝置內造成特定操作。A*之操作禁止對於一節點所作成的計算具有撞擊其他節點之可能性。據此，節點通常以連續、每次一個的方式被處理，而沒有考量來自多數個同時處理來源／核芯之計算的能力。相關地，A*指令不被建構用於允許大規模並行線程、諸如在GPU中所採用者。

【0005】 據此，所需要者係用於改善選路裝置之操作及能夠在越野式行進的大節點環境中迅速及有效率地操作之裝置及方法。

【發明內容】

【0006】 本揭示內容包括操作路線產生器的第一實施例方法，包括：同時地計算用於第一群組中之複數個區塊的路線遍歷值，每一區塊包括複數個胞元，遍歷值係考量地面移動成本資料及指示朝以每胞元為基礎之路線終點的進展之資料的值，其中該複數個區塊被選擇，使得該第一群組中之區塊無法與該第一群組中的其他區塊共享任何邊緣。

【0007】 本揭示內容亦包括操作路線產生器之第二實施例方法，包括：將用於成本圖的胞元之第一區塊的資料載入圖形處理器

之快取記憶體，其中用於該第一區塊中的一胞元之資料的載入需要用於該第一區塊中之所有胞元的資料之載入，並載入第二組胞元，其中該第二組中的每一胞元係不在該第一區塊中，且共享任一邊緣或為與該第一區塊中之胞元對角線性地鄰接，其中該第一區塊中之胞元數目對該第二組中的胞元數目之比率係少於1:8，其中胞元的第一區塊及第二組胞元之結合提供所需要的所有地圖資料，以產生用於該第一區塊之所有胞元的遍歷值，其中遍歷值係考量該成本資料及指示朝路線終點之進展的資料之值。

【0008】 在又另一實施例中，本揭示內容包括操作路線產生器的方法，包括：決定待處理之第一胞元當作決定路線的一部份；決定該第一胞元係在含有至少二胞元之第一區塊內；由第一非揮發性記憶體將分析該第一區塊內的所有胞元所需要之資料載入可藉由處理器存取的揮發性記憶體，該資料被需要，以分析該第一區塊內之所有胞元，該第一區塊包括與該第一區塊中的胞元共享一邊緣及一角落之至少一者的所有胞元；及一旦分析該第一區塊內之所有胞元所需要之資料被載入該揮發性記憶體，分析該第一區塊內的所有胞元。

【圖式簡單說明】

【0009】 圖1係概要圖，顯示示範的路線規劃系統；

【0010】 圖2係藉由圖1之系統所考量的例示土地覆蓋圖；

【0011】 圖3係使用圖2之土地覆蓋圖所產生的例示成本圖；

【0012】 圖4係應用至圖2及3之地圖的同位區塊之示範組；

【0013】 圖5係流程圖，顯示圖1的系統之操作；

【0014】 圖6係流程圖，顯示圖5的操作之路線洪泛的操作；

【0015】 圖7係流程圖，顯示圖6之節點的處理；

【0016】 圖8係例示之邏輯遍歷區塊構造及因此的影響區域；

【0017】 圖9A-B係流程圖，顯示圖7之區塊掃除器(block sweeper)的操作；

【0018】 圖10係流程圖，顯示圖7之終點胞元播種器(cell seeder)的操作；

【0019】 圖11係流程圖，顯示關於圖9A-B之區塊掃除的更多細節；

【0020】 圖12係流程圖，顯示在圖7的鄰接節點加入之後所採用的區塊縮減器(block reducer)之操作；

【0021】 圖13係流程圖，顯示圖6的區塊優先級分配器(block prioritizer)之操作；

【0022】 圖14係流程圖，顯示圖5的路線向量化之操作；

【0023】 圖15係示範圖像，顯示具有覆疊在其上面的路線之遍歷圖的視覺表示法；及

【0024】 圖16係圖解，顯示用於胞元之供選擇的鄰接考量。

【0025】 對應參考字母指示遍及該數個視圖之對應零件。在此中所提出的範例說明本發明之實施例，且此等範例不被解釋為以任何方式限制本發明的範圍。

【實施方式】

【0026】 在此中所揭示之實施例不意欲為詳盡的或將該揭示內容限制至該以下詳細敘述中所揭示之精確形式。反之，該等實施

例被選擇及敘述，以致其他熟習此技術領域者可利用其教導。

【0027】 如在此中所使用，該“邏輯”或“控制邏輯”一詞可包括在一或多個可程式化處理器、特殊應用積體電路(ASICs)、場可程式化邏輯閘陣列(FPGAs)、數位信號處理器(DSPs)、固定線路邏輯、或其組合上執行的軟體及／或韌體。因此，按照該等實施例，各種邏輯可被以任何適當之方式實施，且將按照在此中所揭示的實施例留下。

【0028】 圖1顯示路線規劃計算系統10。系統10包括一或多個處理器12、14、快取記憶體16a,b、隨機存取記憶體(RAM)18、及硬碟20。處理器12係中央處理器(CPU)12，且處理器14係圖形處理器14。每一處理器12、14具有在其上面之快取記憶體16a,b。RAM 18被顯示為待分享於處理器12、14之間。然而，實施例被設想，在此每一處理器12、14具有分開的RAM 18。視訊RAM 22係專門及／或主要被提供用於藉由GPU 14使用之RAM。硬碟20係傳統的大量儲存裝置。

【0029】 如在該電腦技術領域中被輕易地了解，更快之記憶體大致上係更昂貴。據此，用於長期間儲存，計算系統已被設計成具有大的、相對緩慢之非揮發性記憶體。一或多層的較小及更快之記憶體亦被提供，以保有目前正被使用的資料，或多半可能被處理器所使用。據此，RAM 18及視訊RAM 22具有比該硬碟20之記憶體更快的存取時間。RAM 18及視訊RAM 22之尺寸(資料容量)係比該硬碟20的尺寸較小。類似地，快取記憶體16a,b具有比RAM 18及視訊RAM 22更快之存取時間。快取記憶體16a,b的尺寸(資料容量)係比RAM 18及視訊RAM 22之尺寸較小。應被了解這些尺寸不被要求。

然而，在以價格點為折衷來提供計算速率的關注已導致上述為常見之相對定尺寸。

【0030】 據此，用於處理器(12或14)在資料上操作，該資料主要被儲存於硬碟20中。當處理器12、14呼叫資料時，該等記憶體16、18、20、22以典型的高速存取形式操作，在此該最接近及最快速之記憶體(快取記憶體16a,b)首先被檢查。如果該想要的資料不存在於一層級之記憶體(16、18、20、22)，該下一層級被檢查。一旦該想要的資料被找到，其經過每一層級被拉出，直至其係存在於快取記憶體16a,b中及可供處理器12、14直接存取。

【0031】 快取記憶體16a,b之較小尺寸提供在其中所保有的資料通常係受限制於馬上被處理器12、14所需要之資料，用於該目前操作藉此被施行。記憶體間之資料的移動招致該處理及功率消耗中之間接費用，其中此移動費時及需要位元以於記憶體中雙態觸變。

【0032】 GPU 14係被最佳化用於取得大批資料及一再很迅速地施行該相同操作的處理器。CPU微處理器傾向於到處跳過處理請求。CPU 12係由一些核心所構成，且能同時處理一些軟體線程。於對比下，GPU 14係由可同時地處理數千線程之數百核心所構成。

【0033】 圖2顯示地圖資料庫70中所儲存的土地覆蓋圖200之一部份。在一範例中，該瓦片資料係由USGS資料庫所獲得及／或係經由標準地理資訊系統(GIS)、諸如藉由加州紅地的ESRI公司之MapInfo所獲得。土地覆蓋圖200被分開成瓦片(看圖15)。每一瓦片含有多數個區塊210(說明性地64個區塊)。圖2顯示一個此區塊210。每一區塊210說明性地包括8x8柵格的胞元220，每區塊總共64個胞元。每一胞元220含有單一土地類型。如此，用於選路之目的，在一

胞元220內之土地類型被考慮為同質的。地圖資料庫70另包括高度圖(未示出)。

【0034】 該土地覆蓋圖200及高度圖係藉由處理器12、14所處理，以產生成本圖300(圖3)。在此處理之詳情上的更多細節被提供在下面。於所提供之範例中，成本圖300係藉由成本決定性因子96所產生，以表示多麼昂貴、或緩慢，每一地圖胞元係遍歷用於運輸(步行、有輪子的輸送等)之目前模式。5=快速，15=緩慢，及∞=不能通行的成本值係基於該胞元之特色所分派，如由該土地覆蓋圖所決定。該成本圖300被說明性地保有在一或多個快取記憶體16b、視訊RAM 22、及RAM 18中。應被了解不同的成本圖被產生用於藉由選路常式所考慮之每一運輸模式。再者，雖然成本決定性因子96被顯示為係坐落在GPU 14內，實施例被設想，在此成本決定性因子96及路線產生器80更大致上被樣例化在CPU 12內。

【0035】 當產生一越野式路線時，一路線能夠潛在地由一胞元持續進行至8個鄰接胞元(四個橫側鄰接胞元、及四個角落鄰接胞元)的任何一個。當由任何胞元A行進至鄰接胞元B時，成本被累計，等於胞元A及B間之平均成本乘以胞元A及B的中心間之距離。

【0036】 如此，當一路線被計算時，用於待考量的胞元之任何成本被載入可存取至處理器12、14的快取記憶體16a,b。如此，在某些實施例中，當一次考量一個胞元時，感興趣之胞元及其環繞的八個胞元被移入快取記憶體16a,b。雖然在此存在有該等胞元之部份可業已為在快取記憶體16a,b中的可能性，由於其他胞元之考量，此一事件不能被已依賴或期待。如此，每一胞元潛在地呼叫敘述記憶體(16、18、20)間之九個胞元的資料之移動。不同地陳述，胞元的考

量招致移動另一八個胞元之間接費用。類似地，一胞元的處理考量該環繞八個胞元中之資料。

【0037】 於本實施例中，胞元在該區塊210層級被處理。圖2顯示一區塊，其係8x8以含有64個胞元。區塊210的其他實施例係4x4、16x16或32x32胞元大。然而，應被了解區塊之其他尺寸被設想。為簡單起見，4x4區塊410將被討論(圖4)。大致上，具有2的次方數之側面-胞元-長度的區塊被考慮。

【0038】 區塊410藉由它們為什麼方格顏色、或奇偶而被分類。如於圖4中所顯示，該灰色區塊具有0之方格值(checker value)，且該白色區塊具有1的奇偶值(parity value)。該奇偶值係藉由以下所計算：

【0039】 parity=(blockX+blockY)mod2，在此MOD2係熟知之操作，如果其在奇數值上操作，其返回1的值，且如果其在偶數值上操作，返回0之值。

【0040】 這產生一“棋盤”型圖案，在此相像的奇偶區塊係只對角線性地彼此鄰接。具有相像奇偶之區塊未與具有該相同奇偶的其他區塊共享任何邊緣。應被了解在此沒有資料之任何實際著色，僅只有被分派至區塊410的奇偶值。為易於理解，該著色在此被討論及說明。此奇偶分派之實用係在下面進一步討論。

【0041】 每一節點／區塊包括鍵值。該鍵值指示該節點的優先權(或在提供相關選路資訊中之認為的重要性)。具有較低鍵值之節點被指示為已增加的重要性及將被預定為更快地處理。

【0042】 用於給定之選路工作，系統10使用成本圖300，以產生遍歷圖。路線產生器80係執行編碼的說明性處理器14。然而，路

線產生器80為離散邏輯之實施例被設想。路線產生器80建立一遍歷圖1500，該遍歷圖與其成本圖具有1:1胞元對應(成本圖300未對應於遍歷圖1500)。路線產生器80的操作係經由圖5被說明。搜尋初始化建立一最初之遍歷圖，其係與該成本圖相同的尺寸，且準備用於該路線搜尋之開始，看單元510。路線洪泛淹沒來自該路線開始點的搜尋區域，並以表示該累計成本之有限值更新該遍歷圖，以由該路線開始達至該路線終止，看單元520。如在此中所使用，該“洪泛(flooding)”及“洪泛(flood)”一詞意指產生一路線及填充一路線圖的過程。於圖15之圖解說明中，應被了解該路線搜尋、產生、及／或群體由一路線開始擴展出去，好像在此有一水源，使得由該處散發的水洪泛該搜尋區域。返馳初始化準備該路線洪泛器(route flooder)，以返馳所決定之路線，看單元530。返馳係藉由該路線洪泛器所施行，及由該路線終點順著該搜尋區域返回朝該開始處，由正號至負號翻轉該遍歷值之符號，看單元540。路線向量化順著負遍歷值的痕跡，且建構表示該最後路線之坐標清單，看單元550。總的來說，應被了解該遍歷值考量地面移動成本資料及指示以每胞元為基礎朝一路線終點之進展的資料，以抵達用於每一被考量胞元之遍歷值。

【0043】 看單元510經由搜尋初始化器(Initializer)85，搜索初始化將系統10初始化，以提供用於路線洪泛的操作及路線搜尋之性能。洪泛模式被設定為遍歷模式，其稍後被使用在路線掃瞄中(洪泛可模式被設定為遍歷或返馳的任一者)。遍歷圖係藉由建立一與該成本圖具有1:1胞元比率之地圖所初始化。該遍歷圖被設定，使得所有胞元被給與一無限值(不可能遍歷)。目前節點的清單被產生，使得

該清單包括一節點，其包括該開始位置。當搜尋係越過一區塊被施行時，胞元係藉由從區塊之一側面至另一側面處理(掃瞄)所考量。掃瞄係能夠在四個方向(上、下、左、右)中施行。初始化亦設定該所有四個掃瞄方向被賦能。此掃瞄過程係在下面進一步討論。

【0044】 看單元520，經由路線洪泛器90的路線洪泛係產生“初選”路線之過程。用於只要未處理的節點存在於目前節點清單中、看單元610、及用於只要最大數目之迭代尚未被超過、看單元620，路線洪泛操作。所允許迭代的數目基於路線開始及終止間之距離而變動。路線洪泛的第一部份係計算出哪些區域係最可能為與該目前搜尋有關。這係經過區塊優先級分配器之使用所達成。該區塊優先級分配器將該目前節點清單中的節點分開成那些現在待處理及那些稍後待處理之節點，看單元630。現在待處理的節點停留於該目前節點清單中。稍後待處理之節點被移至未來節點清單中。

【0045】 在本實施例中，節點係區塊。本實施例追蹤用於該胞元而非在胞元的邊緣之。成本及遍歷值如此，雖然先前的選路操作產生用於每一胞元之八個遍歷值(一值用於每一橫側及角落的鄰接遍歷點)，本實施例以用於每一胞元而非八個胞元之單一值操作。

【0046】 用於該目前節點清單中的每一節點，該區塊被處理，看單元640。此執行更新用於該給定區塊之遍歷圖，並將用於任何受影響的附近區塊之節點加入至鄰接的節點清單。在GPU 14上，該目前節點清單中之區塊可被平行地執行，每一者在其自身流式多處理器上(其先前已遭受在下面討論的奇偶檢查)。

【0047】 一旦該目前節點清單中之所有節點被處理(於目前節點清單中不再有節點，看單元610)，用於只要該未來及鄰接節點清

單不是空的，看單元660，來自該未來節點清單及該鄰接節點清單之節點被移至該目前節點清單及被處理，看單元650。

【0048】 相對於節點／區塊的處理，看單元640，額外之細節係參考圖7被顯示。首先，用於該目前區塊的遍歷圖被載入，看單元710。如上述，在初始化，該遍歷圖係以用於所有胞元之無窮大值填充。當一區塊被載入時，該單元810的所有該等胞元隨著一胞元寬之填塞環820被載入，看圖8。該填塞環820係由胞元所組成，該等胞元實際上係鄰接(橫向與對角)區塊的一部份。因為遍歷胞元中之變化能影響其附近胞元的任何一者，該等胞元沿著遍歷區塊之邊緣中的任何變化能影響某些鄰接遍歷區塊中之胞元。影響鄰接區塊的遍歷區塊中之邊緣胞元組據說為該鄰接區塊的影響區域之一部份。如此，該填塞環820被載入，使得潛在地影響單元810內的胞元之包羅寬廣的胞元清單係存在為該處理之一部份。當胞元一次一個地被處理時，載入一填塞環呈現用於一胞元的填塞之八個胞元(或待處理胞元的填塞間接費用之8:1比例)。(至少)64個胞元區塊的本範例呈現用於該區塊之64個胞元的填塞之(僅只)36個胞元。這呈現待處理胞元的填塞間接費用之.5625:1比例。如此，該區塊處理提供用於大約間接費用中的14x減少。如此，本實施例考慮少於8:1之胞元的填塞之比率。實際上，在具有16個胞元的區塊中，該填塞具有僅只20個胞元。大致上，用於填塞所需要之胞元的數目係該區塊寬度之四倍(以胞元為單位)加上四(用於該等角落)。

【0049】 其次，對應於該目前處理區塊的成本圖亦被載入，看單元720。該成本圖同樣地被以一胞元填塞環載入。然後，區塊散列值被設定至零(區塊散列在下面被更詳細地討論)，看單元730。

【0050】 該系統10使用優先處理清單中之節點，以排定搜尋工作項目。工作項目被表示為遍歷胞元的區塊，而非使用邊緣，藉此，減少數目及排定工作項目之間接費用。不同地陳述，系統10係從本質上係以光域為基礎，在此其將每一區塊表示為單一節點。雖然路線測繪系統的其他實施例依靠曲線圖之以向量為基礎的表示、亦即所有頂點及邊緣之明確組，在此它們必需表示具有8個節點的每一胞元，一個節點用於每一連接邊緣，提供每一區塊當作單一節點提供記憶體需求中之減少(大約8倍(由於不需對每一胞元邊緣作出解釋，乘以一區塊中的胞元之數目；例如8x32x32=8192)。此記憶體需求減少大幅地改善計算產量，並增加能被支援的路線搜尋之大小。

【0051】 如先前所討論，每一區塊具有八個鄰接區塊。用於目前區塊的外緣胞元之遍歷值中的改變具有可能性，以影響在鄰接區塊中之胞元的值。該散列值保持當用於這些外部胞元之遍歷值改變時的行跡，且接著調整用於該區塊之散列值。該散列值如此保持鄰接區塊係藉由目前區塊上的操作所潛在地改變之行跡。如此，該散列值中的改變造成適當之鄰接區塊被識別，且被放置進入該“鄰接節點”清單。

【0052】 如此，區塊的處理首先將區塊散列至零，看單元730。預處理散列接著基於用於被處理區塊之遍歷圖的最初狀態被設定，看單元740。散列藉由將散列函數施加至區塊內之資料被設定，以便輸出表示每一胞元內的資料之值。該終點接著被播種，看單元745。播種該終點包括設定路線來源地點的值，以具有 ε 之遍歷值(1.4x10-45)。(當在返馳模式中時，在下面討論，播種該終點涉及倒轉(正至負)與該路線終點有關聯的值)。當掃瞄操作被施行時(在下面

討論)，如適當地，對該遍歷圖之任何修改被作成，看單元750。一旦該區塊被處理，該散列操作再次被施加至該遍歷圖，以產生遍歷圖的散列，即後處理，看單元760。此後處理散列接著被倒轉，使得任何正值接著為負的，且任何負值被改變至正的。該後處理散列接著被加至該預處理散列。此操作具有比較該預處理散列及該後處理散列之作用，看單元770。如果在此對給定胞元尚未有改變，則該預處理散列及被倒轉的後處理散列將於該加入期間彼此取消。如此，當被改變時，在具有該預處理及後處理散列之非零總和的影響區域中之任何胞元被識別。潛在地受此被改變胞元所影響的鄰接區塊被識別為待藉由建立用於每一受影響區塊之新節點所進一步處理的鄰接區塊，看單元780。此受影響區塊被加至該鄰接節點清單，看單元790。

【0053】 為合適之一散列函數係由IEEE-754所指定的32位元浮動至32位元整數之按位元拷貝，於C/C++及其他類似程式語言中的操作：

【0054】 int HashFunction(float traversal){

【0055】 int hash=*(int*)(&traversal);

【0056】 return hash;

【0057】 }

【0058】 圖10顯示在終點播種操作上之增加的細節，看單元745。終點播種首先決定該洪泛模式被採用(遍歷或返馳)，看單元1010。返馳模式在下面被討論。當該洪泛模式係遍歷時，系統10決定該路線之開始點是否在該目前遍歷區塊內，看單元1020。如果該路線的開始點係在該目前遍歷區塊內，則該開始點胞元被給與 ε 之

值，看單元1030。這是有差別地大於零的最小數目。未行進任何距離(抵達該開始點)即沒有花費任何成本。然而，既然該系統於該返馳洪泛模式中區別正數與負數，使用 ε 達成兩目的。

【0059】 當該洪泛模式係返馳時，系統10決定該路線之終止點是否在該目前遍歷區塊內，看單元1040。如果該路線的終止點係在該目前遍歷區塊內，則該終止點胞元被給與一倒轉值(5之遍歷值被改變至為-5)，看單元1050。

【0060】 圖9A-B顯示在掃瞄操作上之增加的細節，看圖7之單元750。區塊掃瞄潛在地包括高達四次掃瞄(向上／向北“N”、向下／向南“S”、向左／向西“W”、及向右／向東“E”)。於第一次接收一區塊時，所有四次掃瞄被預定來施行。據此，該區塊在掃瞄暫存器中具有與其有關聯的指令，陳述所有四次掃瞄(N、S、E、W)正待決。如果在藉由一或多個鄰接區塊的處理之後，由於先前已被處理，該區塊已被輸入供處理，指示本區塊被潛在地改變，少於所有掃瞄可為經由該掃瞄暫存器被預定。

【0061】 該掃瞄過程首先決定任何掃瞄是否正待決用於該目前考量的區塊，看單元910。如果至少一掃瞄正待決，則系統10開始執行掃瞄。在“執行掃瞄”上之額外細節係在下面參考圖11被提供。系統10首先檢查以看“N”掃瞄是否正待決，看單元915。如果為否，系統10持續進行至看“S”掃瞄是否正待決，看單元940。

【0062】 如果N掃瞄正待決，則N掃瞄係運行，看單元920。當完結時，該掃瞄暫存器被更新，以反映該N掃瞄被完成，看單元925。系統10接著決定該N掃瞄是否改變任何遍歷值，看單元930。如果沒有遍歷值被改變，系統10持續進行至看“S”掃瞄是否正待決，看單

元940。如果該N掃瞄確實改變至少一遍歷值，則該掃瞄暫存器被更新，以將該E及W掃瞄標記為待決的，看單元935。應被了解該E&W掃瞄可業已被注意為待決的。系統10接著持續進行至看“S”掃瞄是否正待決，看單元940。

【0063】 如果S掃瞄未正待決，系統10持續進行至看“E”掃瞄是否正待決，看單元965。如果S掃瞄正待決，則S掃瞄係運行，看單元945。當完結時，該掃瞄暫存器被更新，以反映該S掃瞄被完成，看單元950。系統10接著決定該S掃瞄是否改變任何遍歷值，看單元955。如果沒有任何遍歷值被改變，系統10持續進行至看“E”掃瞄是否正待決，看單元965。如果該S掃瞄確實改變至少一遍歷值，則該掃瞄暫存器被更新，以將該E及W掃瞄標記為待決的，看單元960。再者，應被了解該E&W掃瞄可業已經由開始值、經由來自該N掃瞄之改變、或以別的方式被注意為待決的。系統10接著持續進行至看“E”掃瞄是否正待決，看單元965。

【0064】 如果E掃瞄未正待決，系統10持續進行至看“W”掃瞄是否正待決，看單元990。如果E掃瞄正待決，則E掃瞄係運行，看單元970。當完結時，該掃瞄暫存器被更新，以反映該E掃瞄被完成，看單元975。系統10接著決定該E掃瞄是否改變任何遍歷值，看單元980。如果沒有任何遍歷值被改變，系統10持續進行至看“W”掃瞄是否正待決，看單元990。如果該E掃瞄確實改變至少一遍歷值，則該掃瞄暫存器被更新，以將該N及S掃瞄標記為待決的，看單元985。系統10接著持續進行至看“W”掃瞄是否正待決，看單元990。

【0065】 如果W掃瞄未正待決，系統10持續進行回至看是否有

任何掃瞄正待決，看單元910。如果W掃瞄正待決，則W掃瞄係運行，看單元992。當完結時，該掃瞄暫存器被更新，以反映該W掃瞄被完成，看單元994。系統10接著決定該W掃瞄是否改變任何遍歷值，看單元996。如果沒有任何遍歷值被改變，系統10持續進行至看是否有任何掃瞄正待決，看單元910。如果該W掃瞄確實改變至少一遍歷值，則該掃瞄暫存器被更新，以將該N及S掃瞄標記為待決的，看單元998。系統10接著持續進行至看是否有任何掃瞄正待決，看單元910。

【0066】 最後，該等掃瞄將中止，以造成該遍歷值中之改變，使得該遍歷值抵達穩定狀態。當這發生時，無改變將允許所有掃瞄被完成，而未呼叫其他掃瞄。沒有任何掃瞄待決，系統10持續進行至保留該遍歷值，看單元915。一旦被保留，系統10繼續運行該後處理散列，看單元760。

【0067】 如所討論，於很多案例中，該等掃瞄造成遍歷值中的改變，且明確地是造成遍歷值沿著給定區塊之邊緣改變，使得該遍歷變化潛在地影響用於鄰接區塊中的胞元之計算。此受影響的區塊被加至該鄰接節點清單，看單元790。將區塊加入至該鄰接節點清單包括建立用於該被增加的節點之性質。這些性質包括鍵值(其決定其優先權)、其地點(潛在地一組座標)、及最初的掃瞄(當該區塊被處理時，該掃瞄被施行)。

【0068】 該鍵值係經由該公式被說明性地決定：

Key=min(影響區域中之遍歷胞元)+最小成本*抵達距離。

“影響區域中之遍歷胞元”係該目前區塊中的那些胞元，其毗鄰所考量之鄰接胞元，使得如果該鄰接區塊被載入，“影響區域中之遍

歷胞元”亦因此將被載入當作該填塞環的一部份。在8x8胞元區塊中，用於橫向鄰接之區塊，該系統10將預期影響區域中的八個遍歷胞元。用於斜對鄰接之區塊，該系統10將預期該影響區域中的一遍歷胞元。如此，該公式之“min()”部份將發現該胞元具有來自該有關群組的最低遍歷值及使用該值。

【0069】 “最小成本”係該最小可能成本胞元之值。在一範例中，該最低成本值可能為“5”。於胞元長度中，抵達距離係由影響區域中的胞元及該鄰接區塊中之最接近胞元行進的距離。用於正好北方、南方、東方或西方之鄰接區塊(橫向鄰接)，該接觸距離係1。用於該四個對角線鄰接區塊(對角線鄰接)，該接觸距離係$\sqrt{2}$。

【0070】 如上面所討論，“掃瞄”係什麼更新遍歷胞元值的事物。此掃瞄發生在四個方向中(北、南、東、西)，看單元920、940、970、992。圖11提供在什麼掃瞄必需上之額外細節。區塊掃瞄聚焦在掃瞄四個方向的其中一者中的區塊，在此每一掃瞄均勻地評估用於每一胞元之8個鄰近者的其中3個。

【0071】 首先，指示是否任何遍歷區塊胞元已藉由該區塊掃瞄被修改之布林值的掃除器改變遍歷(Sweeper Changed Traversal)被設定至假，看單元1110。在此點，系統10建立外部迴圈，以經過由後面至前面之等級值來迭代，看單元1120，在此一等級係用於W或E掃瞄的區塊中之一列胞元、或用於N或S掃瞄的區塊中之一行胞元。如果處理器XX係中央處理器(CPU)，則有一迭代經過所有胞元的內部迴圈，看單元1130。如果處理器XX係GPU，則一行之所有胞元被同時地執行，每一行在一分開的串流處理器上。當被處理時，用於每一胞元，“後面”、“後左側”、及“後右側”胞元被識別，看

單元1140。該等方向“後面”、“後左側”及“後右側”係參考掃瞄之方向。該遍歷胞元的現值亦被設定至“目前胞元”，以能夠稍後決定被改變之遍歷值。

【0072】 來自該後面胞元的遍歷值被決定為(單元1150)：

traversal via back=traversal (back cell)+1/2*(cost(current cell)+cost(back cell))

【0073】 來自該後左側胞元之遍歷值被決定為：

traversal via back left=traversal (back left cell)+√2/2*(cost(current cell)+cost(back left cell))

【0074】 來自該後右側胞元的遍歷值被決定為：

traversal via back right=traversal (back right cell)+√2/2*(cost(current cell)+cost(back right cell))

【0075】 該目前遍歷值接著藉由計算該成本所決定，以由該“後面”、“後左側”及“後右側”胞元之每一者到達該目前胞元，以發現該最低成本可能性。這麼做，該系統運行遍歷胞元最佳化器(Traversal Cell Optimizer)(單元1170)、或遍歷胞元返馳器(Traversal Cell Retracer)(單元1180)的任一者，視該目前洪泛模式而定，看單元1160，以基於該舊值及該三個後面之遍歷值計算該適當的新遍歷胞元值。如果該目前遍歷胞元被修改，任一模組將掃瞄器改變遍歷(Sweeper Changed Traversal)設定至真的。

【0076】 應被了解在給定等級中之值只取決於來自先前等級的值。如此，在一等級內的胞元經由GPU之同時處理係可能的，因該共同處理之胞元係彼此獨立的。

【0077】 當該洪泛模式係遍歷(Traverse)時，遍歷胞元最佳化器

被使用於區塊掃瞄，以界定遍歷胞元如何改變值及傳播用於該最佳路線之搜尋。其工作如下：

首先，系統10評估該新的遍歷值：

traversal (current cell)=min({old traversal, traversal via back, traversal via back right, traversal via back left})

如果：abs(traversal (current cell)-old traversal)>容許誤差(Tolerance)，則該掃瞄器改變遍歷值被設定至真。這讓系統10得知該遍歷區塊中之最佳已知路徑係仍然正改進，且於不同方向中的更多掃瞄被需要。

【0078】 當該洪泛模式係返馳(Retrace)時，遍歷胞元返馳器被區塊掃瞄器所使用，在下面進一步討論。

【0079】 用於鄰接節點中之每一節點，系統10計算用於該節點的試探函數(Heuristic function)，並將其加至該節點之鍵值，看單元642。關於該有名的A*演算法，該鍵值表示至目前為止所累加之總成本加上至該目標的試探或預計成本。因為鄰接節點中之節點鍵值只表示該總成本、或遍歷。

【0080】 該試探函數被計算為由該路線的末端胞元至該區塊中之最接近胞元的距離乘以該最小成本圖。為計算此，系統10計算水平距離dx=max({end X-(node X*B+(B-1)), (node X*B)-end X,0})，在此如果路線終點係區塊之東方，該大括號中的第一項決定dx，如果在區塊之西方則該第二項決定dx，且如果末端係在區塊的東及西界線內該第三項決定dx。下一系統10計算計算(Compute)垂直距離dy=max({end Y-(node Y*B+(B-1)), (node Y*B)-end Y,0})。系統10接著計算距離d=√(dx^2+dy^2)。該試探函數接著被計算為

d*MinimumCost。

【0081】 在路線洪泛已執行目前節點中的所有區塊之後，目前節點可被清除，看單元645。剩餘工作被發現於未來節點及鄰接節點清單中。下一區塊計畫者(Next Block Planner)優先化鄰接節點中的節點，以該鄰接節點及未來節點清單之序連連接替換該目前節點清單，且藉由呼叫區塊縮減器合併任何重複節點，看單元650。

【0082】 既然鄰接節點加入，看單元790，且下一區塊規劃將更多節點加入至該目前節點清單，而未考慮其他節點業已於該清單中，可有重複節點存在。區塊縮減器發現在此有表示該相同區塊的多數個節點情況，並將它們組合成一個。當多數個節點被組合時，區塊縮減器提供該經組合之情況適當地捕捉該原來情況的資訊。譬如，如果節點A具有北方及東方之最初掃瞄，而節點B具有南方及東方的最初掃瞄，該組合之節點將具有北方及南方、及東方的最初掃瞄。類似地，如果節點A具有100之鍵值(非常緊急)及節點B具有800的鍵值(較不緊急)，該組合之鍵值將為100。

【0083】 更明確地是，區塊縮減器如圖12所示地操作。該區塊縮減器檢查看多少節點係在該目前節點清單中。如果有比目前清單中的二節點更少，看單元12，略過至單元1260。如果有大於二節點，則該區塊縮減器持續將該節點SortBy值設定為等於節點之X XOR(節點的y<<13)，在此XOR係邏輯異或(exclusive OR)操作，且“<<”係邏輯位元左移操作，兩操作可輕易地用在任何軟體平臺上。該XOR及<<操作係可用在大部分所有處理器上的熟知操作。當作一散列碼之該SortBy特性函數允許表示示該相同區塊的任何節點具有該相同之SortBy值。用於目前節點中的每一節點，這被做成，

看單元1215。

【0084】 該目前節點清單之節點接著被其SortBy特性所分類，看單元1220。諸如在中央處理器上的Quicksort或在GPU上之Bitonic Sort的程式說明性地被使用於此過程。分類之方向沒有關係。重要的是用於該相同區塊之節點係在該被分類清單中彼此毗連。

【0085】 區塊縮減器接著初始化二節點指標。節點A在該目前節點中的第一節點上開始，且節點B在該第二節點上開始，看單元1225。

【0086】 該系統接著檢查節點B是否在該目前節點之末端，看單元1230。如果區塊縮減器係在該目前節點的末端，其持續進行至單元1260。如果在此區塊縮減器不在該目前節點之末端，則區塊縮減器看節點A值是否等於節點B值，看單元1235。如果它們不相等，該節點A被設定至該節點B值，看單元1240，且節點B被增量一，以指向目前節點中的下一值(下一節點A係該先前節點B)，看單元1242。如果節點A及節點B被發現具有該相同之值，則重複節點已被發現。節點合併器(Node Merger)係在該等節點上運行，以將摘錄自每一節點的資訊(鍵值、掃瞄)之節點的性質合併進入一被組合之節點，且其被儲存為節點A，看單元1245、1250。用於節點B的鍵值被設定至無窮大，且藉此被標記供刪除，看單元1255。節點B接著被推進，以指向該目前節點清單中之下一節點，且區塊縮減器返回至單元1230。

【0087】 一旦該目前節點清單被處理，區塊縮減器持續進行至移去具有無窮大的鍵值之目前節點清單中的節點，看單元1260。

【0088】 圖13提供在運行單元630之區塊優先級分配器上的額

外細節(圖6)。藉由優先級分配該等目前節點中之節點及將稍後被完成的任何節點移出及移入未來節點，該區塊優先級分配器決定哪一個待決之遍歷區塊應其次於區塊執行器(Block Executer)中被處理。目前節點中的剩餘節點被選擇，以致它們可被區塊執行器以任何順序或甚至平行地執行。所選擇之節點包括具有該最小鍵值(亦即，該最緊急)的節點及具有小鍵值之任何其他節點，看單元1310，且具有該相同的區塊奇偶(看圖4)，看單元1320。即刻類似的，該等鍵值將藉由鍵值截止值所決定，其可為想要之性能而被實驗性地調整，看單元1330。於所顯示範例中，該截止值被設定等於該MinimumCost值乘以二十五乘以“B”(在此B係一區塊的等級(或檔案)中之胞元數目。該數目25被經驗性地選擇，以允許合適數目之區塊被同時發生地處理，而不會招致過度需要以再處理區塊)。選擇所有具有相同奇偶的節點確保它們不會在邊緣上彼此接觸(僅只角落)，以便使相互依存減至最小及支援平行地執行它們。該MinParity值被設定至Partity(MinNode)，使得該最小成本節點界定該奇偶，以在該下一處理波中被使用，看單元1320。一旦該截止鍵值被設定，該未來節點及鄰接節點清單被清除，看單元1340。其次其被決定該目前節點清單中之哪一節點具有大於該截止鍵值的節點鍵值、或具有與該MinParity不同之奇偶，看單元1350。此等節點被由該目前節點清單移去，且被附加至該未來節點清單，看單元1360。

【0089】 看單元530，返馳初始化係藉由返馳初始化器92所做成。該返馳初始化器重新建構系統10，用於第二輪或該路線洪泛器，以施行一路線返馳。該返馳初始化器將該洪泛模式設定至“返馳”。該返馳初始化器亦設定該目前節點清單，以含有具有該區塊

的單一節點，而該區塊在其中設有該路線終點，並設定用於該節點之掃瞄方向。

【0090】 系統10接著持續進行至返馳所建立的路線。當該洪泛模式係返馳時，遍歷胞元返馳器被區塊掃瞄器所使用，(執行區塊掃瞄模式之處理器)標記在該最後路線中的任何遍歷胞元，看單元540。不像在所有方向中由該路線開始擴展該遍歷圖之遍歷胞元最佳化器，遍歷胞元返馳器由該路線終點回至該開始只更新遍歷胞元的薄帶尾。遍歷胞元返馳器標記該等遍歷胞元，該等遍歷胞元係可在它們上翻轉該符號之路線的一部份，使得它們變負的。其工作如下：如果：

1.舊遍歷>0及

2.以下之其中一者為真：

a.abs(舊遍歷+經由後面胞元的遍歷)<容許誤差，

b.abs(舊遍歷+經由後左側胞元的遍歷)<容許誤差，或

c.abs(舊遍歷+經由後右側胞元的遍歷)<容許誤差，

則：

1.遍歷(目前胞元)=-(舊遍歷)，亦即標記目前胞元當作該最後路線之一部份。

2.將掃瞄器改變遍歷設定為真，亦即通知路線執行器，即在不同方向中之更多掃瞄係需要的。

【0091】 為了進一步說明該等上面條件：條件1確保該目前遍歷胞元尚未被標記為該最後路線之一部份，以防止折回或在一迴圈中被捕獲。條件2a檢查舊遍歷及經由後面胞元的遍歷是否相等且相反，因為容許誤差實際上為零。如果條件1及2a皆為真，其能被推論

經由後面胞元之遍歷係負的，如此該後面胞元業已被標記為該最後路線之一部份。吾人亦可下結論，即該舊遍歷及經由後面胞元的遍歷在該返馳過程被開始之前係相等的，如此該目前胞元係至後面胞元之最快速路徑的一部份。因此，遍歷(目前胞元)應被標記。條件2b-c係與條件2a相同，除了它們應用至該後面及後左側附近胞元以外。

【0092】 其次，該路線係經由路線向量化器(Vectorizer)94向量化，看單元550。路線向量化器94說明性地為處理器執行碼。在該路線向量化器94之操作上的額外細節被提供於圖14中。該路線向量化器94產生頂點之清單，其由該路線產生的區塊及胞元界定該最後路線。路線向量化器94在該遍歷胞元於該路線的末端開始，且向後工作至該開始胞元。在每一點，路線向量化器94發現哪一鄰接胞元具有該最大的負遍歷值，將該點加至路線點之清單，且持續進行至該點。一旦該路線的開始被抵達，路線點之清單被顛倒，於測繪胞元坐標中產生該最後路線。

【0093】 路線向量化器94首先清除該路線點，看單元1410。然後，該路線終止點被設定為開始點，看單元1420。該路線終止點(向量化開始點)被附加至該路線點清單，看單元1430。路線向量化器94接著檢查看其是否已抵達該路線開始點，看單元1440。當該路線開始點尚未抵達時，路線向量化器94檢查毗連該目前考慮的胞元之所有胞元，以發現具有最大值的負值之胞元(- ε ，將為具有該最大值的負值)，看單元1450。該發現點被設定為該路線中之下一點，看單元1460，且被附加至該路線，看單元1470。單元1450、1460及1470之此過程被重複，直至該路線向量化器94抵達該路線開始點。一旦

該路線向量化器94抵達該路線開始點，該路線被由頂點的清單所組成，並由該路線終止運行至該路線開始。如此，該路線點接著被顛倒，以產生界定該路線之頂點的清單，看單元1480。

【0094】 用於長路線、諸如50英哩長，需要被搜尋之區域係非常大。為造成一成本圖及遍歷圖大到足以表示此一大區域可需要被配置至其上的非常高數量之記憶體。區塊條帶(Block Swath)係一簡單及有效的資料結構，用於藉由：1)對該圖上之胞元的任何區塊提供快速隨機存取，及2)提供稀疏測繪以減少記憶體浪費來表示大圖。

```
class BlockSwath
  {
   int BlockSize；
   int ChunkSize；
   int WidthChunks；
   int HeightChunks；
   float [][][]Data；//float ***Data in C/C++
  }
```

【0095】 區塊尺寸(BlockSize)敘述於胞元中之每一區塊的寬度及高度。BlockSize應為2之乘方。信息塊量度(ChunkSize)敘述於胞元中之每一信息塊的寬度及高度。ChunkSize應為2之乘方，且大於BlockSize。WidthChunks敘述條帶中的許多行之信息塊。HeightChunks敘述條帶中的許多列之信息塊。資料(Data)敘述含有該實際胞元資料、可定址為Data[chunk][block][cell]的鋸齒狀列陣，在此：信息塊表示含有該胞元之信息塊的指標，

chunk=[x/ChunkSize]+[y/ChunkSize]*WidthChunks

區塊表示該信息塊內之區塊的指標：

block=([x/BlockSize]+[y/BlockSize]*ChunkSize) MOD ChunkSize^2

胞元表示該區塊內之胞元的指標：

cell=(x+y*BlockSize) MOD BlockSize^2

x及y係來自該整個區塊條帶之西北角落的x及y胞元坐標。

【0096】 藉由以此方式表示資料，該未抵達遍歷區塊將不需被配置，每區塊節省(例如)4KB之記憶體。同樣地，未抵達的信息塊將不需要被配置，每信息塊節省256位元組(假設ChunkSize為256位元組)，其當處理非常大規模之搜尋區域時最終成為一大節省。

【0097】 遍歷圖1500係所產生的遍歷值及藉由系統10所產生之路線1510的視覺說明。圖1500包括開始點“d”及終點“e”，在其間具有路線1510。被定位在共用條紋(諸如條紋1520、1530、1540、1550)上之地點表示離該開始點“d”具有相像的行進時間距離之地點。

【0098】 實施例亦被設想，其採用雙向搜尋。雙向搜尋涉及產生二路線搜尋，一路線搜尋在該路線的每一端部開始，且一旦它們在該中間會合即停止。這是更有效率的，因其減少被搜尋之總面積，且幫助偵測不能通行的路線，在此該路線端部係在不能達到之島嶼上。

【0099】 實施例被進一步設想，其採用多分辨率選路。多分辨率選路涉及使用道路成本地圖來產生粗略層級路線，並使用它，以在一精細層級沿著環繞該先前路線的走道限制該路線產生。更明確地，於一實施例中，64個胞元之區塊被處理當作具有單一成本值的單一實體。該上述選路係使用只用於此粗略區塊之值來施行，以使

該等區塊極可能變窄，以含有該最後的路線。

【0100】 實施例被進一步設想，其擴展超出至八個鄰接胞元之移動的評估，且代替地考慮16點鄰接。該額外之鄰接被顯示在圖16中。該額外的鄰接包括“西洋棋-騎士(chess-knight)”鄰接。相對八點鄰接，十六點鄰接具有該潛力，以提供增加平滑之路線、增加精確的成本計算，並增加該角度行進選擇。

【0101】 實施例被設想，具有拐角線切除。拐角線切除有助於消除鋸齒狀或“拐角線”路徑(例如去北方達100公尺後接去東北方達100公尺之路徑，代替直接去北北東方達約180公尺)，其係當試著產生越過一致(或接近一致)成本之區域的路線時常見之人工因素。

【0102】 實施例亦被設想，其採用先行圖載入(Look-Ahead Map Loading)。先行圖載入看該路線搜尋係多麼接近被卸載的地圖圖磚，並優先化哪些下一次被載入之地圖圖磚，以確保該路線產生不等候在該圖載入上。

【0103】 雖然本發明已被敘述為具有一示範設計，本發明可在此揭示內容的精神及範圍內被進一步修改。此申請案因此係意欲涵蓋本發明使用其一般原理之任何變動、使用、或修正。再者，此申請案係意欲涵蓋此等由本揭示內容的偏離，如來至在本發明有關之技術領域中的習知或慣常實例內。

【符號說明】

10 路線規劃計算系統

12 中央處理器

14 圖形處理器

16a 快取記憶體

16b 快取記憶體

18 隨機存取記憶體

20 硬碟

22 視訊RAM

70 地圖資料庫

80 路線產生器

85 初始化器

90 路線洪泛器

92 返馳初始化器

94 路線向量化器

96 成本決定性因子

200 土地覆蓋圖

210 區塊

220 胞元

300 成本圖

410 區塊

510～1480 單元

1500 遍歷圖

1510 路線

1520～1550 條紋

A 胞元

B 胞元

【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

公告本

I651515

發明摘要

【發明名稱】（中文/英文）

用於改善路線規劃計算裝置之裝置及方法

DEVICE AND METHOD FOR IMPROVING ROUTE PLANNING COMPUTING DEVICES

【中文】

路線產生器及操作該路線產生器之方法包括；同時地計算用於第一群組中之複數個區塊的路線遍歷值，每一區塊包括複數個胞元，遍歷值係考量地面移動成本資料及指示朝向每胞元為基礎之路線終點的進展之資料的值，其中該複數個區塊被選擇，使得該第一群組中之區塊無法與該第一群組中的其他區塊共享任何邊緣。

【英文】

A route generator and method of operating the same including; calculating route traversal values for a plurality of blocks in a first group simultaneously, each block including a plurality of cells, traversal values being values that consider terrain movement cost data and data indicating progress towards a route endpoint on a per-cell basis, wherein the plurality of blocks are chosen such that the blocks in the first group fail to share any edges with other blocks in the first group.

【代表圖】

【本案指定代表圖】：圖1。

【本代表圖之符號簡單說明】：

10　路線規劃計算系統

12　中央處理器

14　圖形處理器

16a　快取記憶體

16b　快取記憶體

18　隨機存取記憶體

20　硬碟

22　視訊RAM

70　地圖資料庫

80　路線產生器

85　初始化器

90　路線洪泛器

92　返馳初始化器

94　路線向量化器

96　成本決定性因子

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：

無

申請專利範圍

1.一種操作路線規劃系統之路線產生器之方法，其改良包括：

識別地圖中之區塊，該區塊對應於不同部分之物理地面；以及

同時地計算用於第一群組中之複數個區塊的路線遍歷值，每一區塊包括複數個胞元，遍歷值係考量地面移動成本資料及指示朝向每胞元為基礎之路線終點的進展之資料的值，

其中該複數個區塊被選擇，使得該第一群組中之區塊無法與該第一群組中的其他區塊共享任何邊緣。

2.如申請專利範圍第1項之方法，其中用於該複數個區塊的路線遍歷值係藉由圖形處理器同時地計算。

3.如申請專利範圍第1或2項之方法，其中每一區塊被分派二奇偶值之其中一者。

4.如申請專利範圍第3項之方法，其中具有第一奇偶的每一區塊係橫側地鄰接只具有第二奇偶之其他區塊。

5.如申請專利範圍第3項之方法，其中具有第一奇偶的每一區塊係對角線性地鄰接只具有該第一奇偶之其他區塊。

6.如申請專利範圍第3項之方法，其中該第一群組中的區塊共享同奇偶。

7.如申請專利範圍第1項之方法，其中當用於任何區塊中的任何胞元之遍歷值被計算時，用於該區塊中的所有胞元之遍歷值被要求待計算。

專利申請案第106131529號
ROC Patent Appln No. 106131529
修正後無線之中文圖式替換本 附件(1)
Amended Drawings in Chinese - Encl. (1)
(民國107年11月15日送呈)
(Submitted on November 15, 2018)

圖式

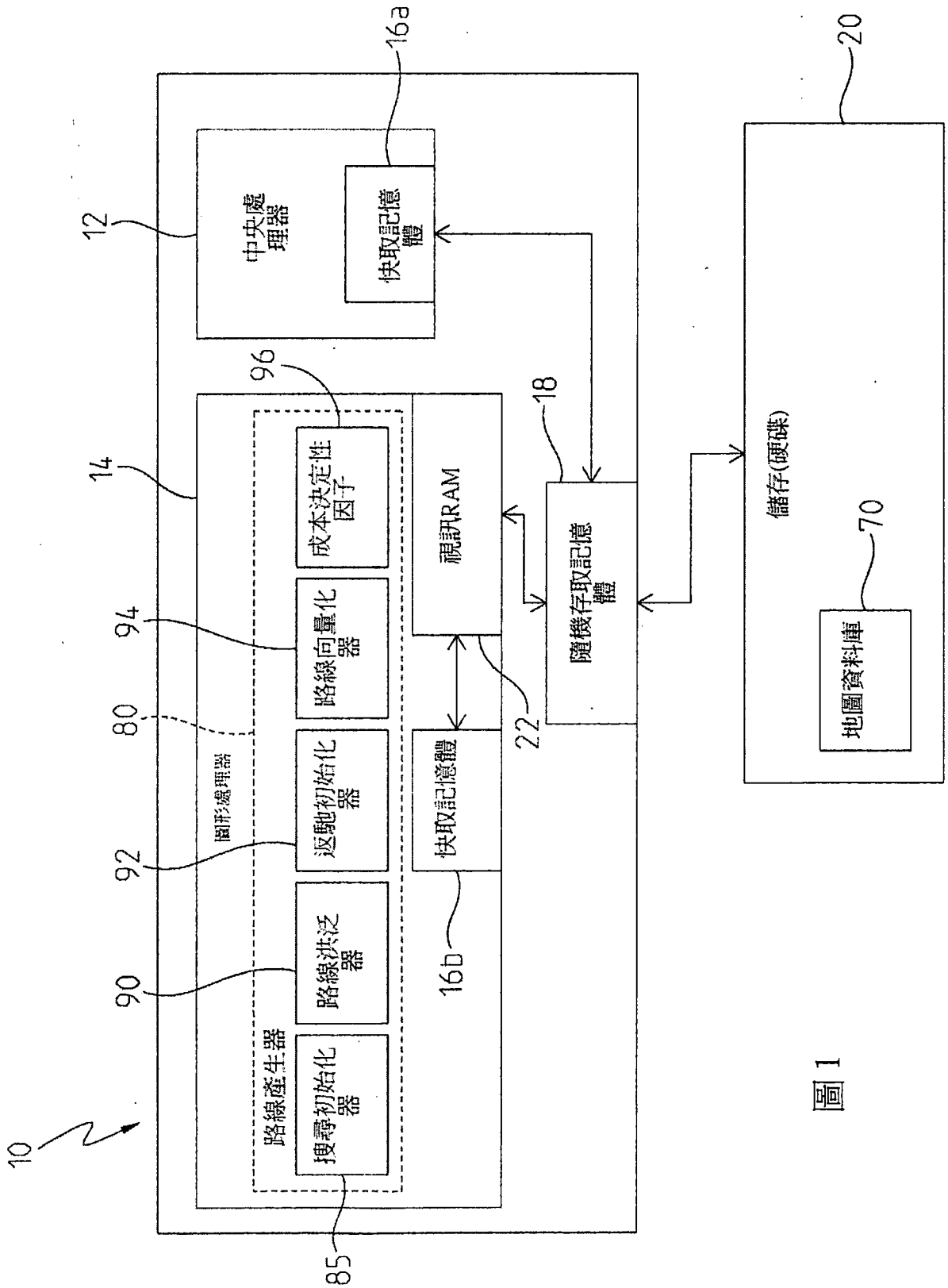


圖 1

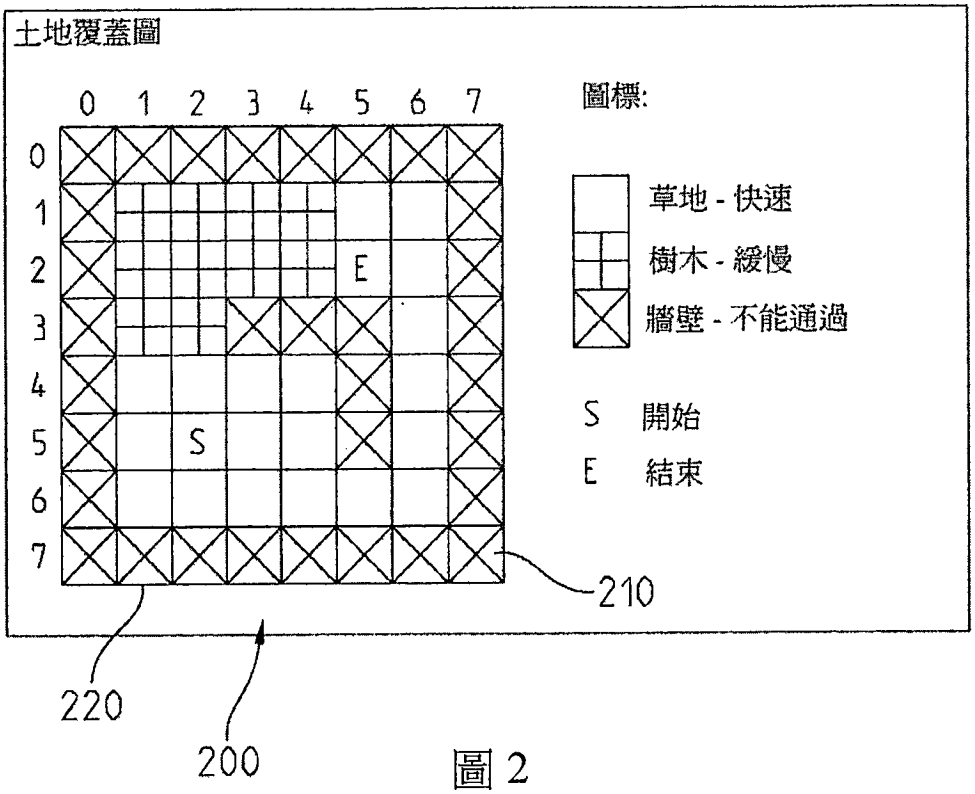


圖 2

成本圖

	0	1	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	15	15	15	15	5	5	∞
2	∞	15	15	15	15	5	5	∞
3	∞	15	15	∞	∞	∞	5	∞
4	∞	5	5	5	5	∞	5	∞
5	∞	5	5	5	5	∞	5	∞
6	∞	5	5	5	5	5	5	∞
7	∞	∞	∞	∞	∞	∞	∞	∞

300

圖 3

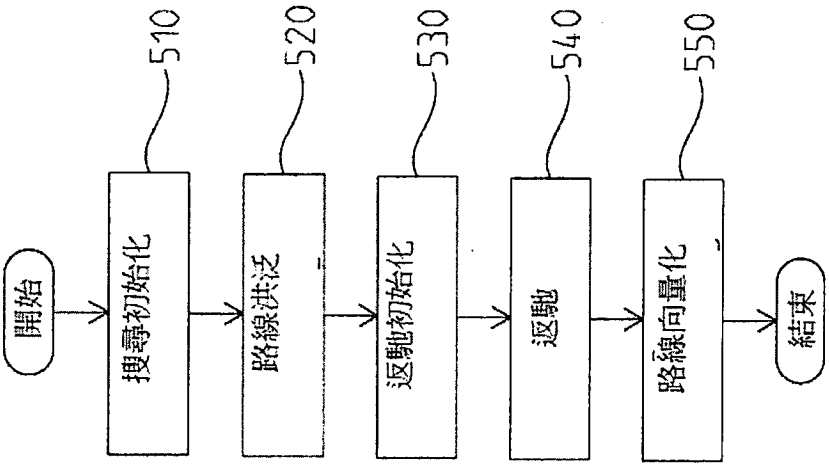

開始
搜尋初始化
510
路線洪泛
520
返馳初始化
530
返馳
540
路線向量化
550
結束

圖 5

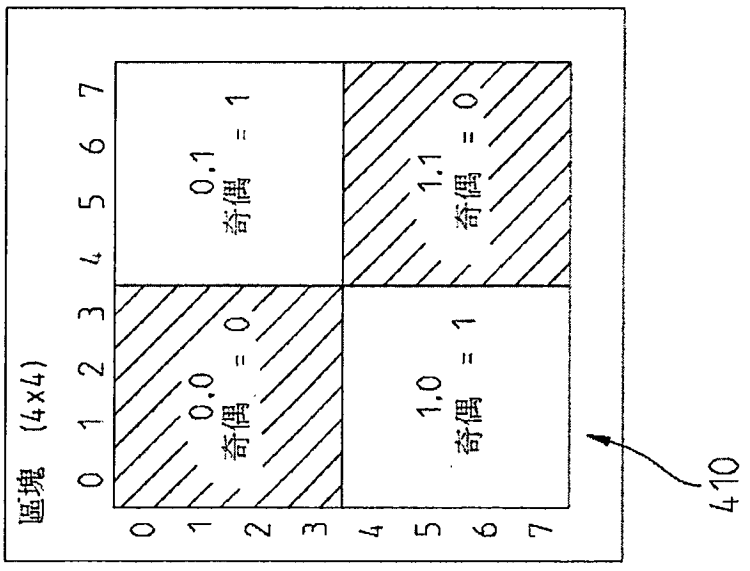

區塊 (4x4)
0 1 2 3 4 5 6 7
0,0
奇偶 = 0
0,1
奇偶 = 1
1,0
奇偶 = 1
1,1
奇偶 = 0
0 1 2 3 4 5 6 7
410

圖 4

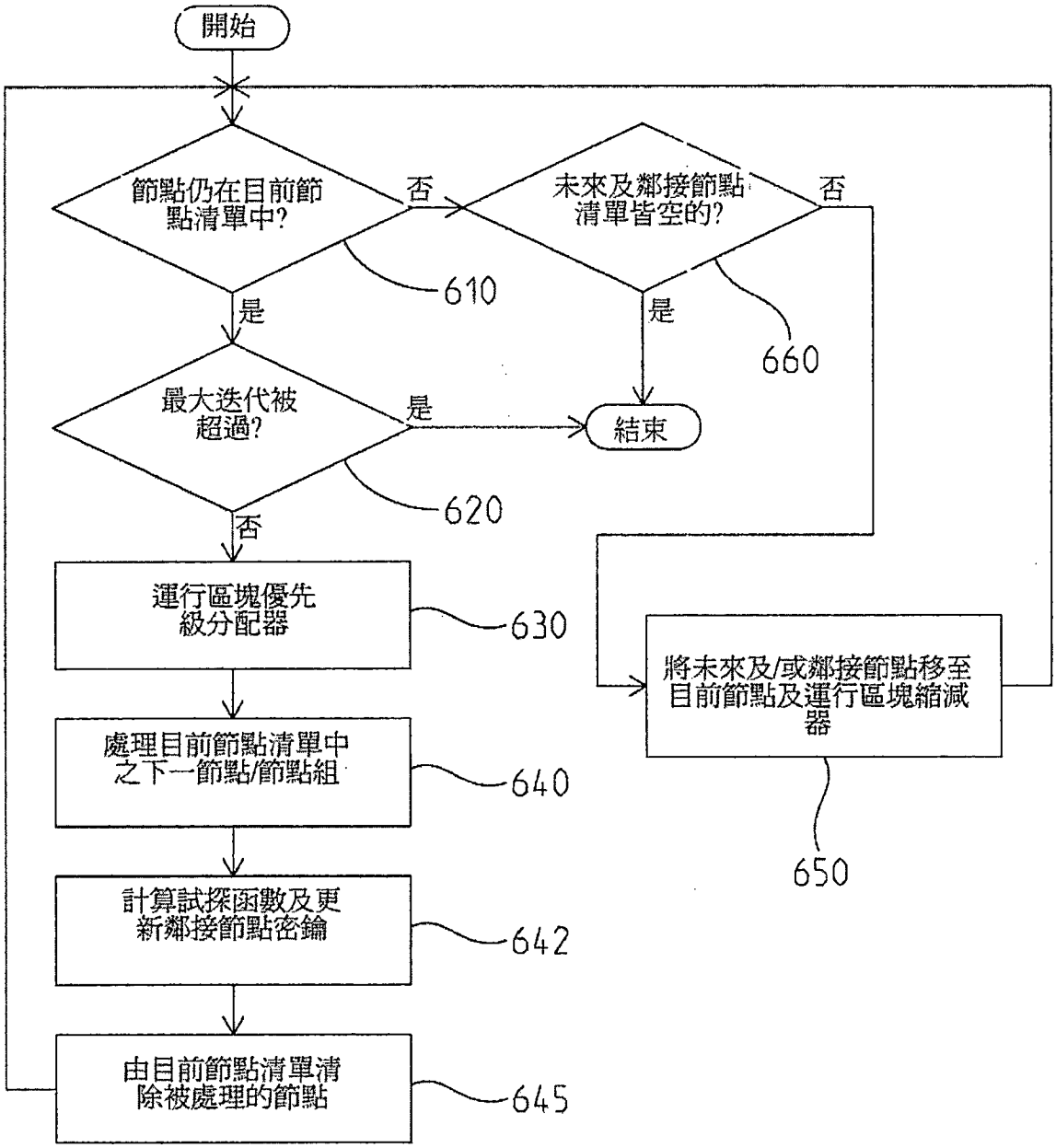

開始
節點仍在目前節點清單中?
否
610
是
最大迭代被超過?
是
結束
620
否
運行區塊優先級分配器
630
處理目前節點清單中之下一節點/節點組
640
計算試探函數及更新鄰接節點密鑰
642
由目前節點清單清除被處理的節點
645
未來及鄰接節點清單皆空的?
否
660
是
將未來及/或鄰接節點移至目前節點及運行區塊縮減器
650

圖 6

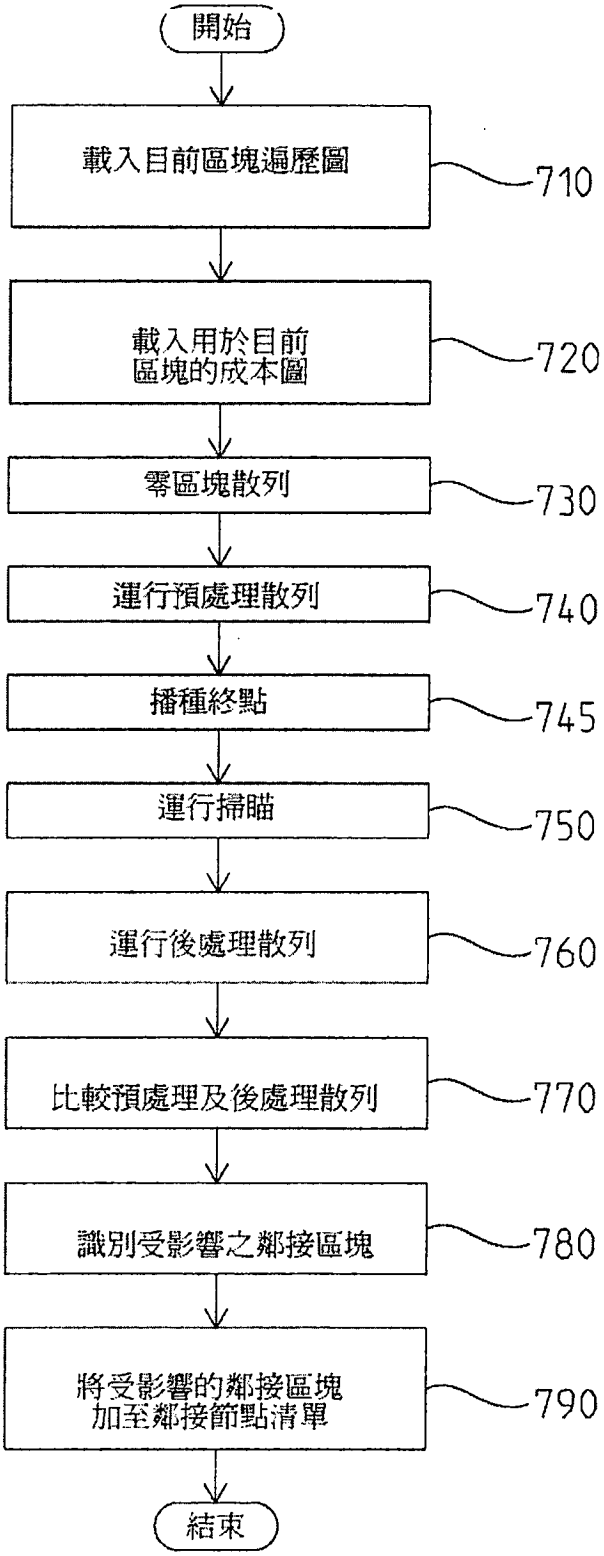
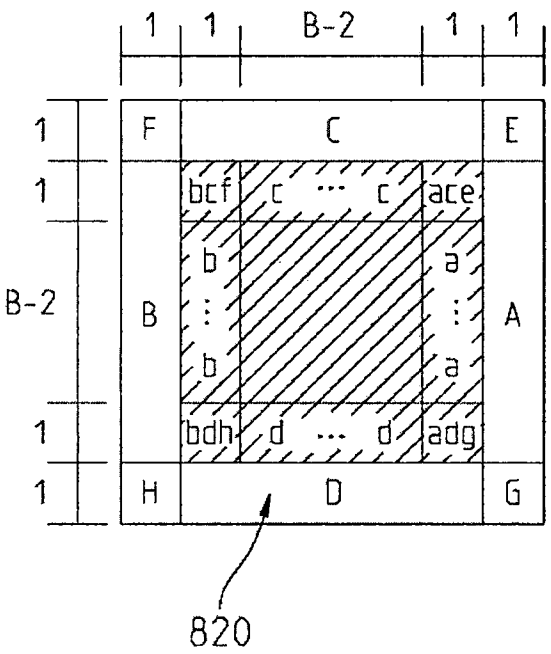

開始
載入目前區塊遍歷圖
710
載入用於目前區塊的成本圖
720
零區塊散列
730
運行預處理散列
740
播種終點
745
運行掃瞄
750
運行後處理散列
760
比較預處理及後處理散列
770
識別受影響之鄰接區塊
780
將受影響的鄰接區塊加至鄰接節點清單
790
結束

圖 7



□ 具有填塞的遍歷區塊

▨ 沒有填塞的遍歷區塊

a 用於鄰接區塊A的影響區域(東方)

b 用於鄰接區塊B的影響區域(西方)

c 用於鄰接區塊C的影響區域(北方)

d 用於鄰接區塊D的影響區域(南方)

e 用於鄰接區塊E的影響區域(東北方)

f 用於鄰接區塊F的影響區域(西北方)

g 用於鄰接區塊G的影響區域(東南方)

h 用於鄰接區塊H的影響區域(西南方)

A 鄰接區塊A的部份(東方)

B 鄰接區塊B的部份(西方)

C 鄰接區塊C的部份(北方)

D 鄰接區塊D的部份(南方)

E 鄰接區塊E的部份(東北方)

F 鄰接區塊F的部份(西北方)

G 鄰接區塊G的部份(東南方)

H 鄰接區塊H的部份(西南方)

圖 8

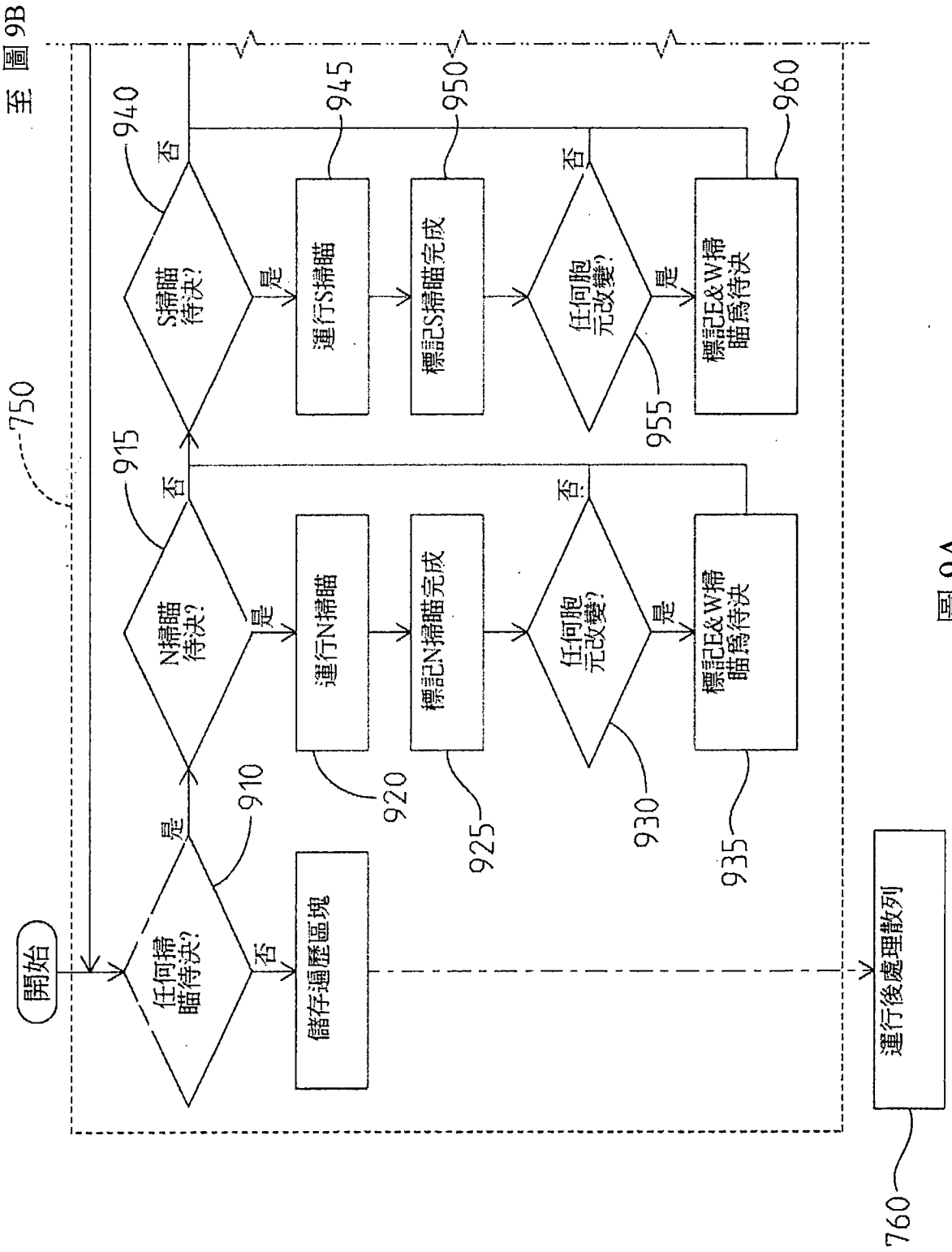

至 圖 9B
750
開始
任何掃瞄待決?
是
否
910
儲存遍歷區塊
N掃瞄待決?
是
否
915
運行N掃瞄
920
標記N掃瞄完成
925
任何胞元改變?
是
否
930
標記E&W掃瞄爲待決
935
S掃瞄待決?
是
否
940
運行S掃瞄
945
標記S掃瞄完成
950
任何胞元改變?
是
否
955
標記E&W掃瞄爲待決
960
運行後處理散列
760

圖 9A

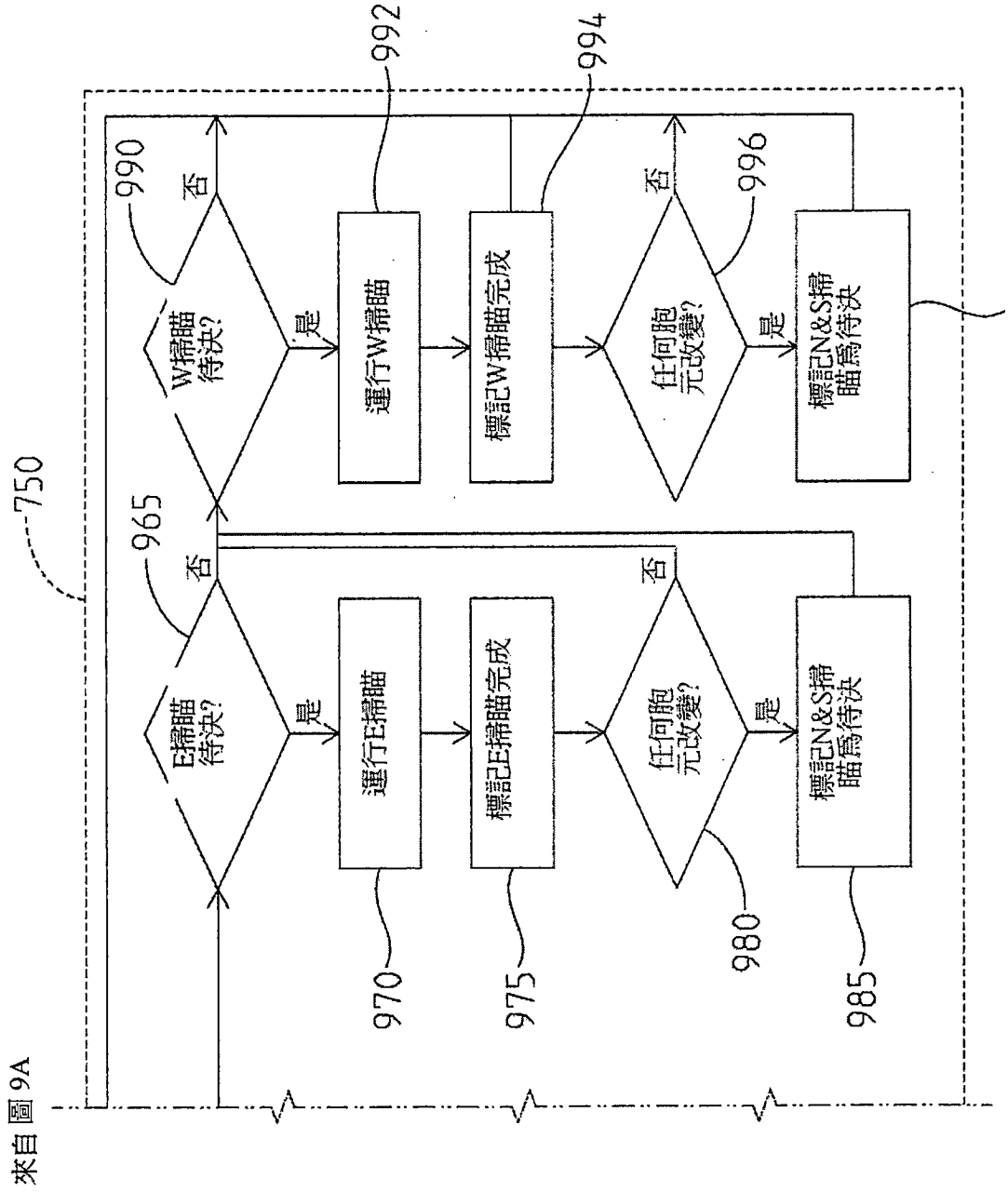

來自 圖 9A
750
965
否
E掃瞄待決?
是
運行E掃瞄
970
標記E掃瞄完成
975
任何胞元改變?
980
是
標記N&S掃瞄爲待決
985
990
否
W掃瞄待決?
是
運行W掃瞄
992
標記W掃瞄完成
994
任何胞元改變?
996
否
是
標記N&S掃瞄爲待決
998

圖 9B

終點胞元播種器

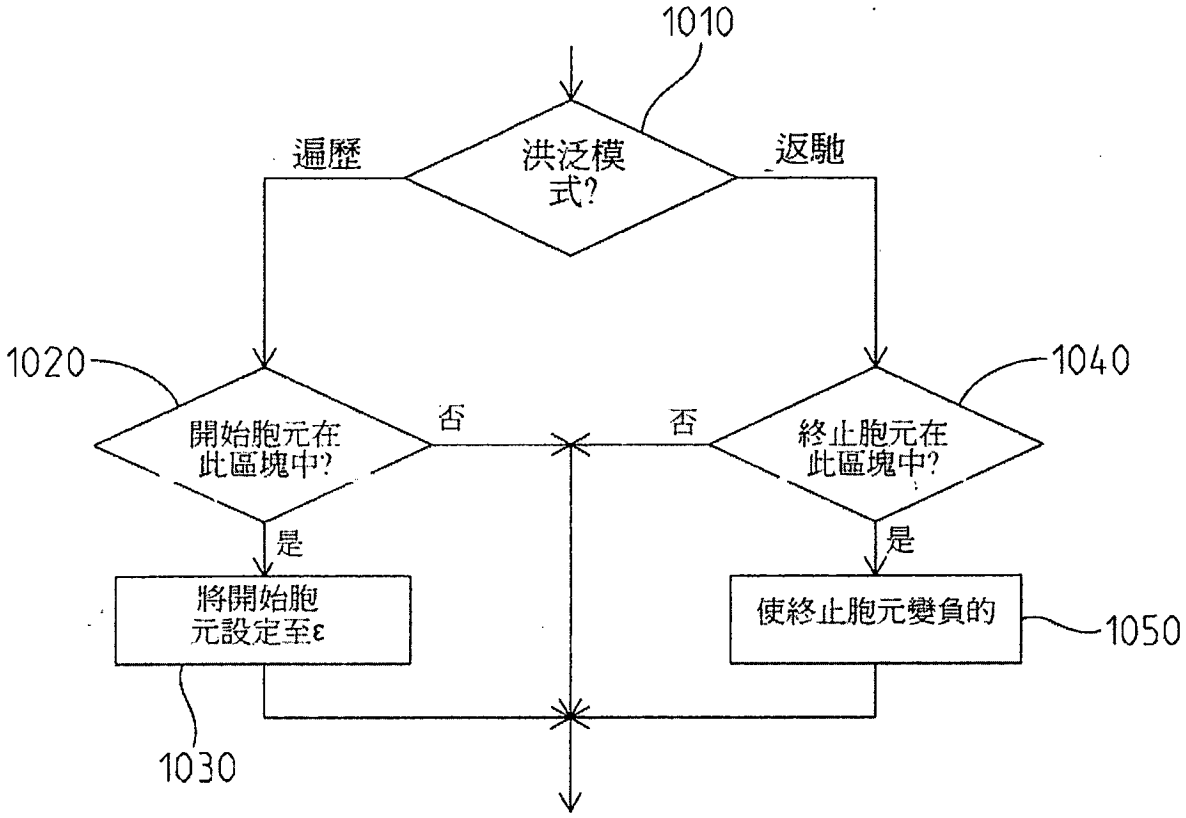

1010
洪泛模式?
遍歷
返馳
1020
開始胞元在此區塊中?
否
否
1040
終止胞元在此區塊中?
是
是
將開始胞元設定至ε
使終止胞元變負的
1050
1030

圖 10

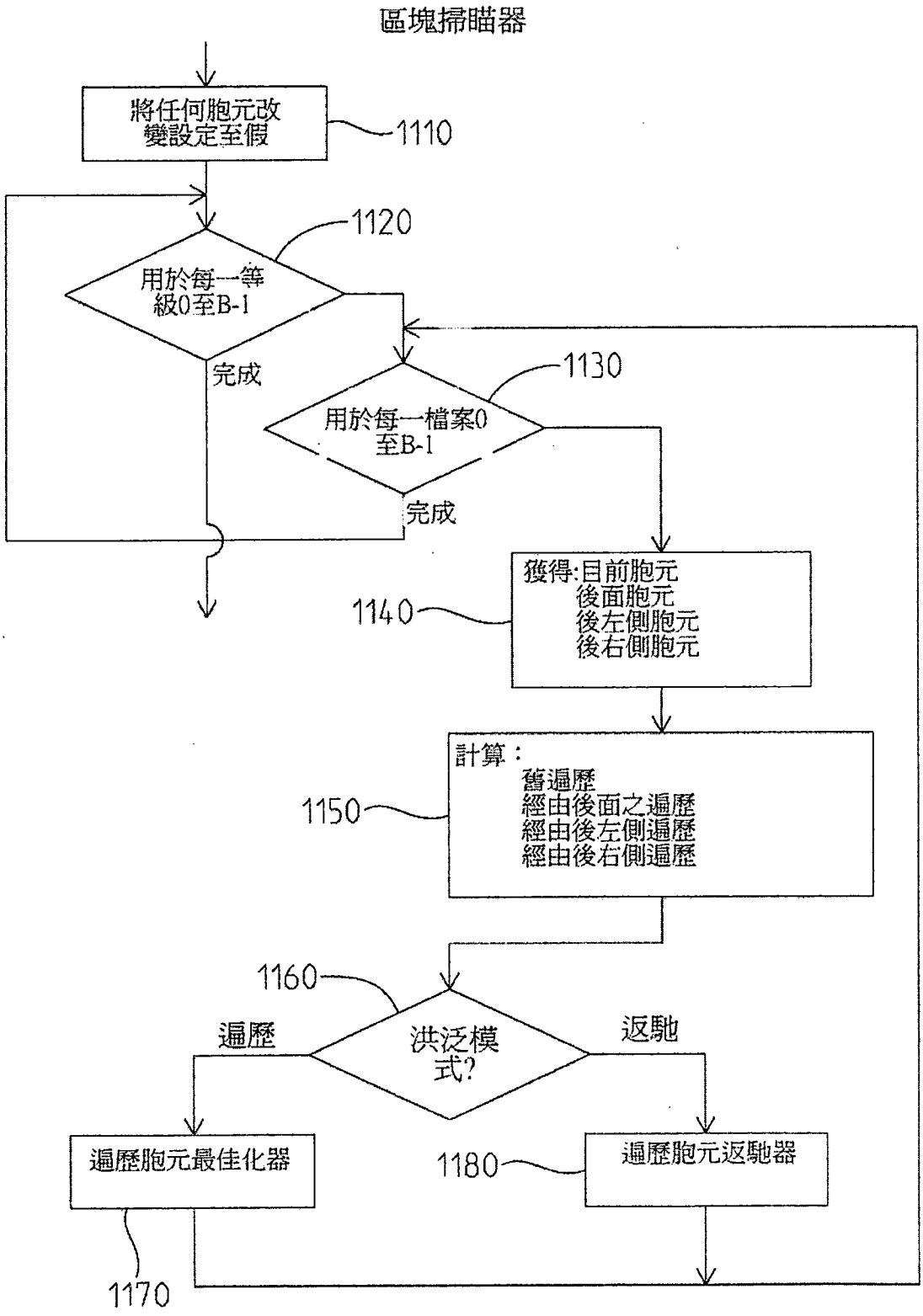

區塊掃瞄器
將任何胞元改變設定至假
1110
1120
用於每一等級0至B-1
完成
1130
用於每一檔案0至B-1
完成
獲得:目前胞元
後面胞元
後左側胞元
後右側胞元
1140
計算：
舊遍歷
經由後面之遍歷
經由後左側遍歷
經由後右側遍歷
1150
1160
洪泛模式?
遍歷
返馳
遍歷胞元最佳化器
1180
遍歷胞元返馳器
1170

圖 11

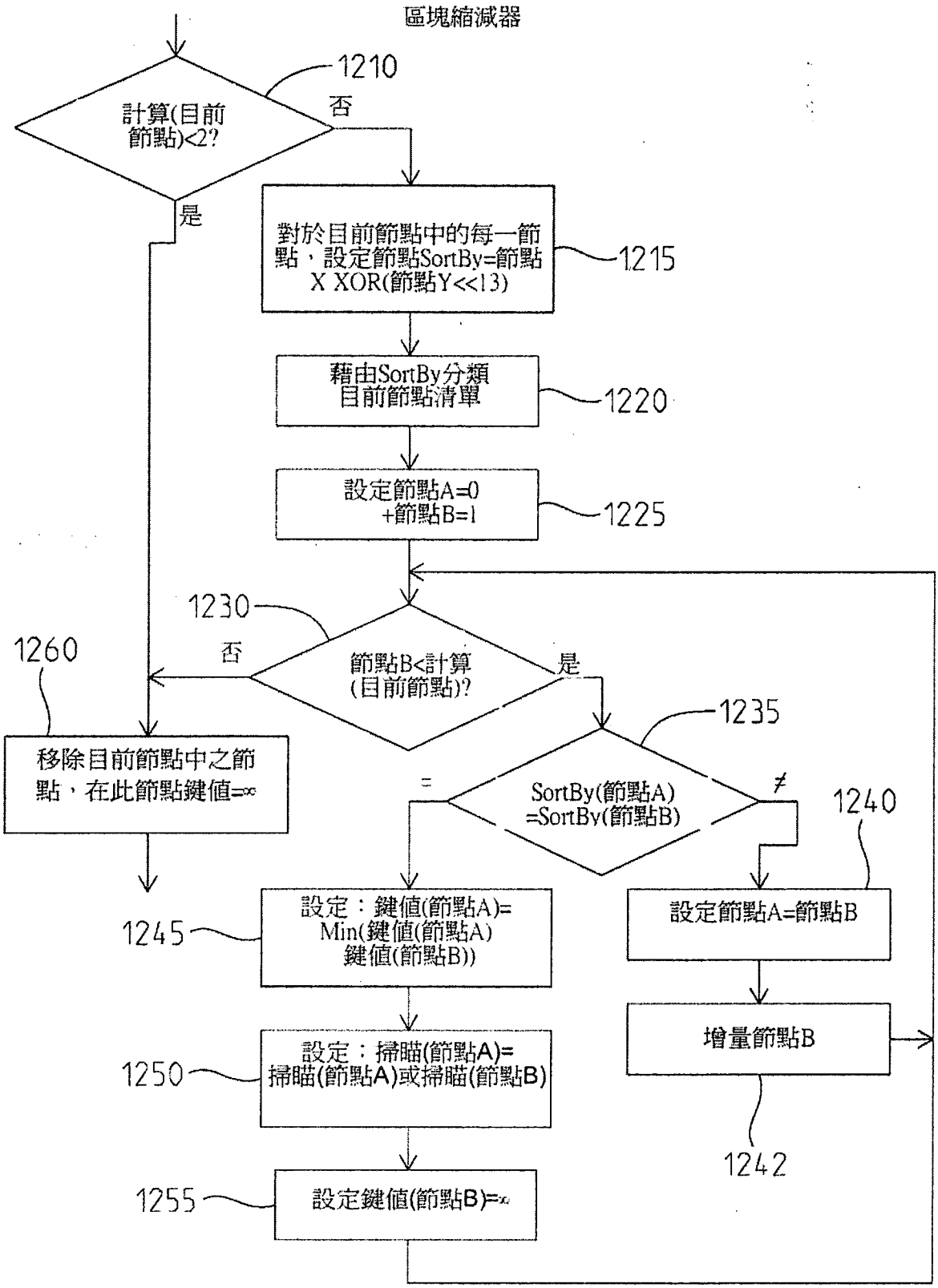

區塊縮減器
計算(目前節點)<2?
1210
否
是
對於目前節點中的每一節點，設定節點SortBy=節點X XOR(節點Y<<13)
1215
藉由SortBy分類目前節點清單
1220
設定節點A=0 +節點B=1
1225
1230
節點B<計算(目前節點)?
否
是
1260
移除目前節點中之節點，在此節點鍵值=∞
1235
SortBy(節點A) =SortBy(節點B)
=
≠
1240
設定：鍵值(節點A)= Min(鍵值(節點A) 鍵值(節點B))
1245
設定節點A=節點B
增量節點B
1242
設定：掃瞄(節點A)= 掃瞄(節點A)或掃瞄(節點B)
1250
設定鍵值(節點B)=∞
1255

圖 12

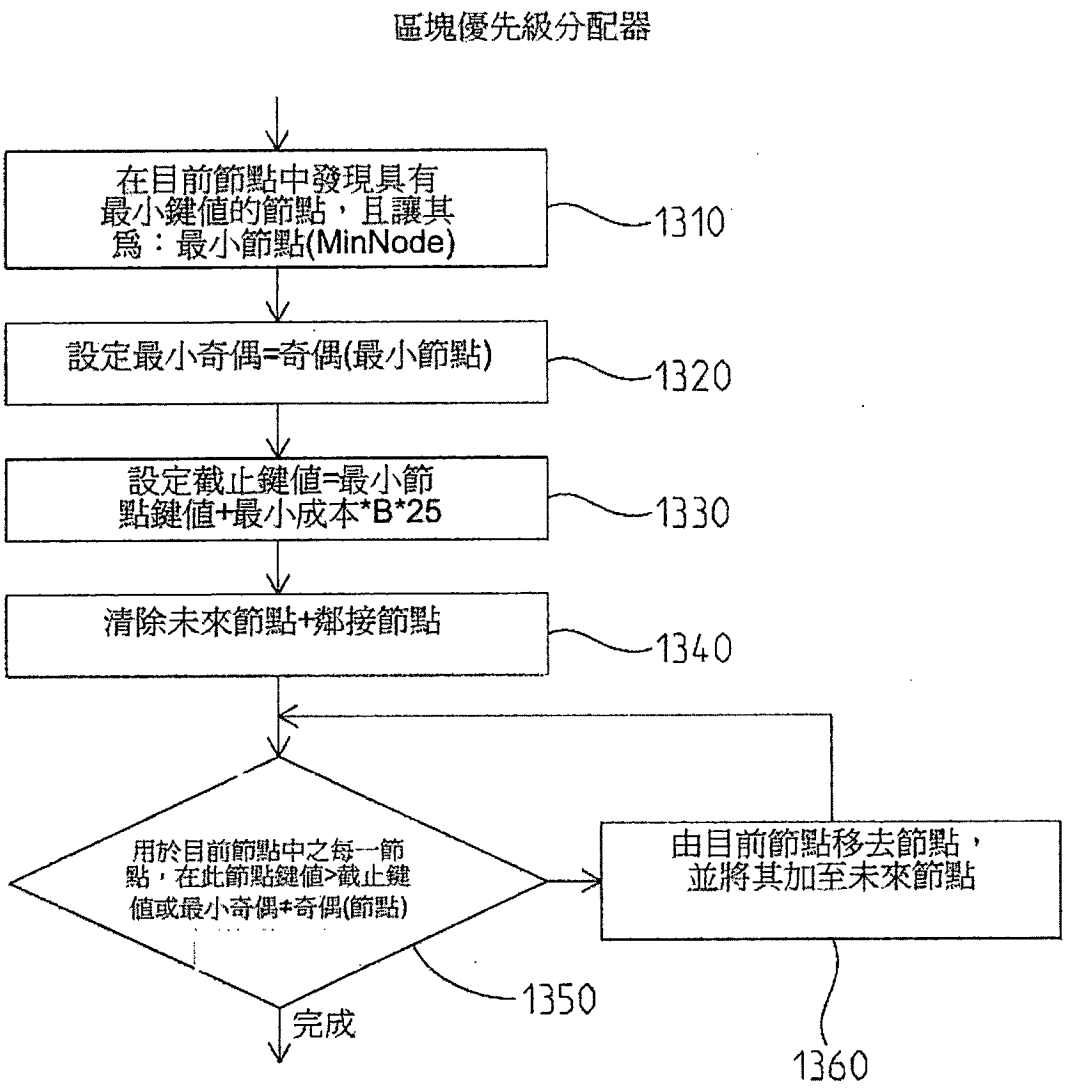

區塊優先級分配器
在目前節點中發現具有最小鍵值的節點，且讓其爲：最小節點(MinNode)
1310
設定最小奇偶=奇偶(最小節點)
1320
設定截止鍵值=最小節點鍵值+最小成本*B*25
1330
清除未來節點+鄰接節點
1340
用於目前節點中之每一節點，在此節點鍵值>截止鍵值或最小奇偶≠奇偶(節點)
1350
完成
由目前節點移去節點，並將其加至未來節點
1360

圖 13

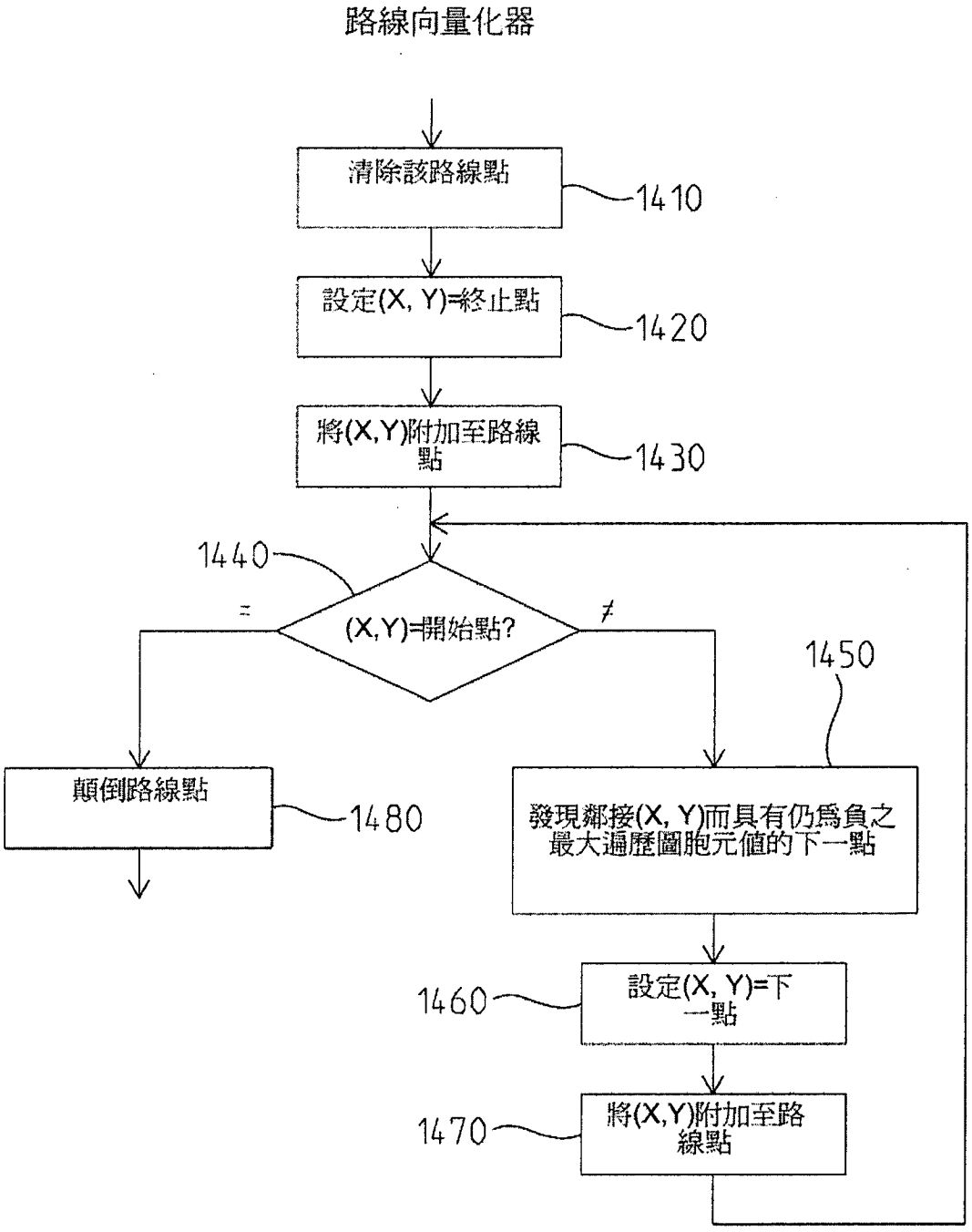

路線向量化器
清除該路線點
1410
設定(X, Y)=終止點
1420
將(X,Y)附加至路線點
1430
1440
(X,Y)=開始點?
=
≠
顛倒路線點
1480
1450
發現鄰接(X, Y)而具有仍爲負之最大遍歷圖胞元值的下一點
1460
設定(X, Y)=下一點
1470
將(X,Y)附加至路線點

圖 14

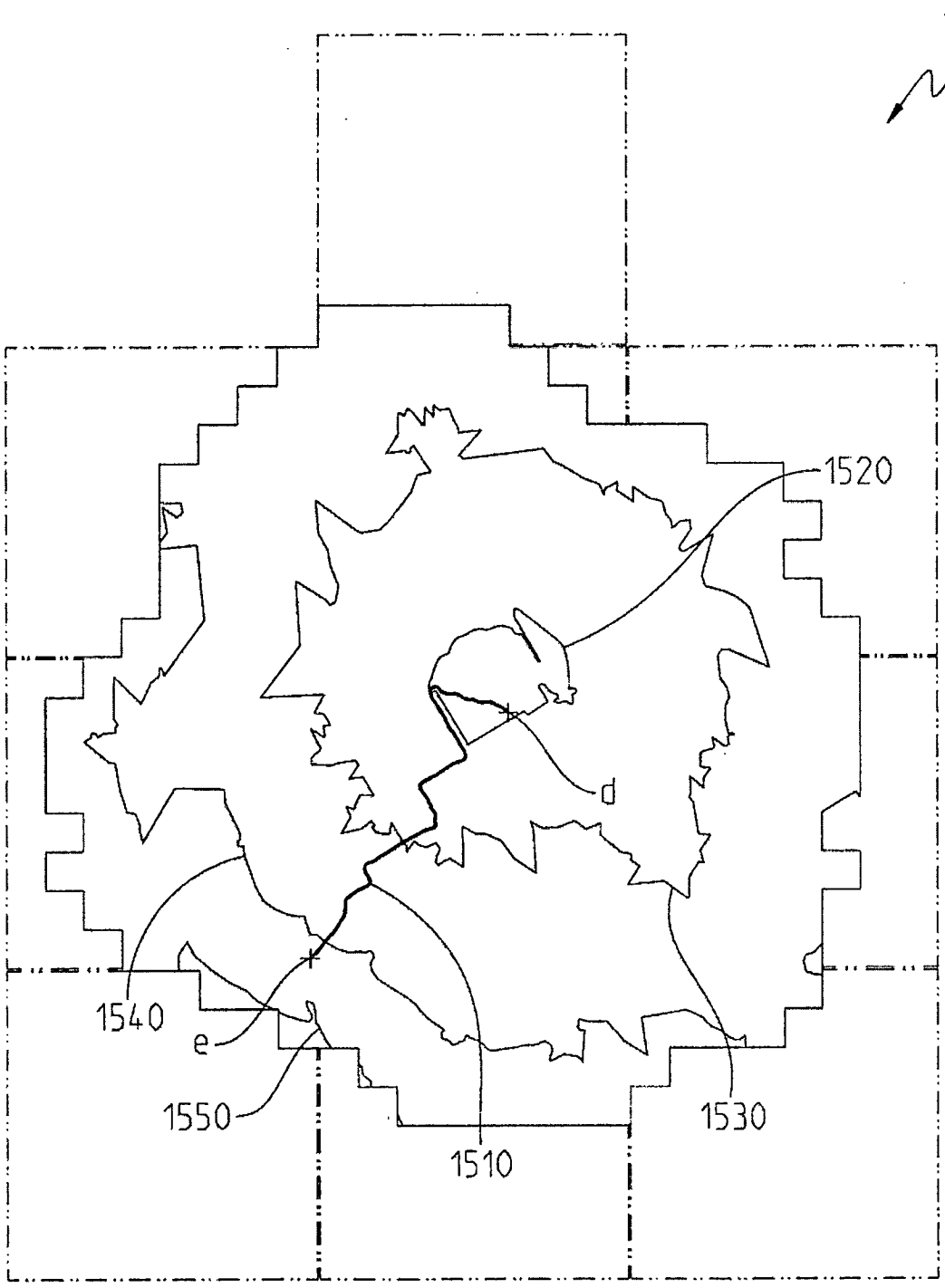


圖 15

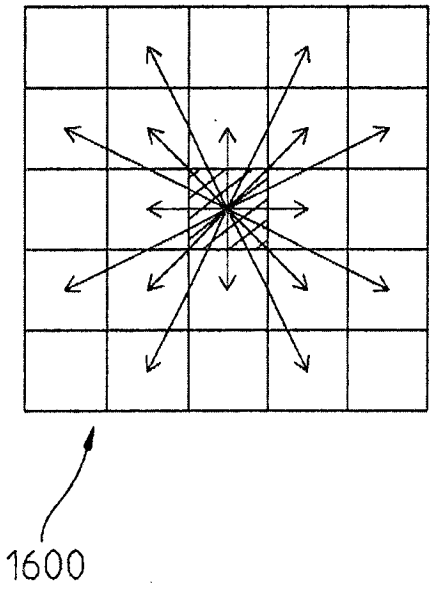


圖16