一個解決二維閒置空間定位問題的快速演算法
An Efficient Algorithm for 2D Dead Space Identification Problem

卓政達
大葉大學
r9403030@mail.dyu.edu.tw

程仲勝
大葉大學
jscherng@mail.dyu.edu.tw

摘要

本文提出一個對於二維區間內閒置空間定位問題的快速演算法。給定一個固定大小的二維區間，區間內有多個大小不一的矩形以不重疊但相連方式擺放其中，本論文討論如何找出此二維區間內所有矩形以外之閒置空間的位置及形狀。在實驗結果中我們將利用一個測試例子來加以驗證所提演算法。

關鍵詞：閒置空間、演算法

Abstract

In this paper, an efficient algorithm for 2D dead space identification problem is proposed. Given a 2D region of fixed area, many rectangles of various dimensions are placed inside with non-overlapped but connected each other. A novel approach is proposed to identify the location and shape of each dead space, which is a region existed among rectangles. In the experiment, an example with 100 rectangles is used to certify the proposed algorithm.

Keywords: dead space, algorithm

一、簡介

我們常常會在作業系統中聽到一個名詞—記憶體碎片(memory fragmentation)[1]，碎片的產生有可能是記憶體的配置器(allocator)沒有把資料分配在連續記憶體位置或者這些記憶體位置已經不使用並釋放出來所造成，如圖 1 所示，前者即為

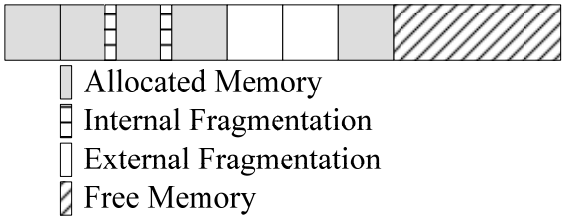

圖 1.記憶體碎片分類與位置

Internal Fragmentation，External Fragmentation 則屬於後者，則這兩種碎片就是在現有空間分配沒有被利用的空間，這些空間即被定義為閒置空間(Dead Space)，在記憶體這種一維架構下定位(Identification)這些閒置空間並不困難，僅需從頭至尾掃描一次，把閒置的空間依序記錄下來即可，這些資訊可以提供記憶體配置器將新的資料配置在合適的地方，以免造成記憶體的餘額足夠但是卻沒足夠大的地方放至新資料的窘境。

但是在二維區間的閒置空間定位問題上並不是那麼簡單，舉個例子來說倉儲管理[2]中已經配置好的儲存物品的區間大小和位置因為情況的改變而需要有所變更，例如某個儲存物品的區間需要加大佔地面積，則必須要先尋找附近是否有足夠的閒置空間來因應這個情況，若附近無法提供足夠的面積，則需要思考是否有些儲存物品的區間能夠挪動一些位置來因應，這些策略運用皆需要事先把閒置空間定位出來，閒置空間定位問題在解決儲存物品的區間空間變更方面亦顯重要。

另外一個例子，在超大型積體電路後端平面規劃(floorplanning)[3]設計中，設計師因應實際情況需要來修改原先的電路區塊，原先的電路模組可能因為新加入的功能或是規格變更導致比原先的電路模組面積還大，在以往如果沒有對閒置空間做分析，則需要進入重新平面規劃的流程，但

這個流程相當耗費時間，同樣的若如果附近有足夠的閒置空間，則可由其吸收此面積的差值，如果附近的閒置空間不足，則可能需要挪動其他電路模組或者更甚者需要進行重新平面規劃，閒置空間定位問題在這方面將能夠提供資訊輔助判斷是否要進入重新平面規劃流程。

我們對於二維區間的閒置空間定位問題可定義為給定一個固定大小的二維區間，區間內有多個大小不一的矩形以不重疊但相連方式擺放其中，本論文討論如何找出此二維區間內所有矩形以外之閒置空間的位置及形狀。對於閒置空間定位問題來說，我們需要一個初始的配置圖，如圖 2 所示，在配置圖中需要這些內部矩形的四個角落的座標，還有一個外部矩形的四個角落的座標，且這些矩形不能彼此重

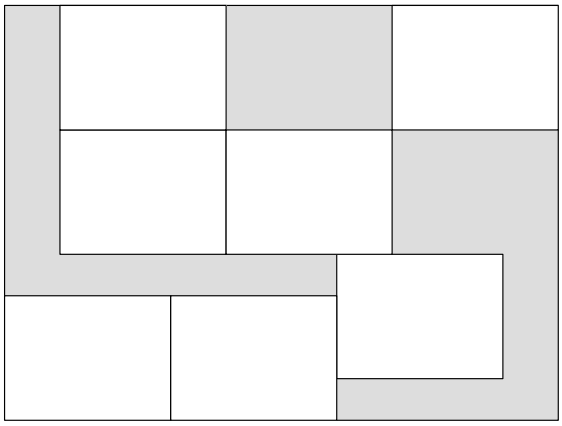

圖 2.一個具有 7 個矩形的配置圖

疊，我們將提出演算法找出這些灰色閒置空間區域的轉角(corner)座標。

一般而言直觀的作法採取先做水平掃描再做垂直掃描來定位閒置空間的轉角座標，但是這個做法比較難處理如圖 3 的例

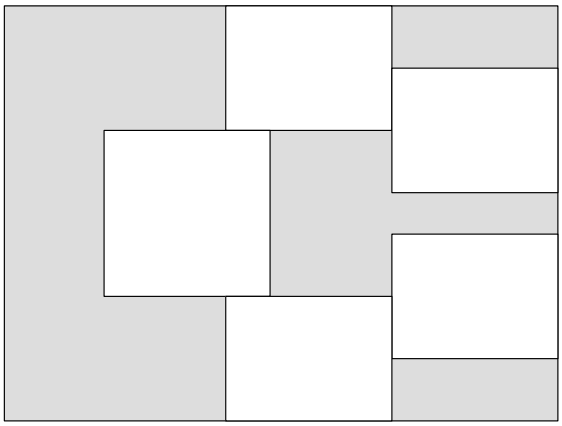

圖 3.較困難用掃描求解的配置圖

子。圖 3 中左邊灰色凹型閒置空間，則需要進行分割合併的動作才能順利利用水平垂直掃描法解決，若不分割則需要做逐點掃瞄，所需時間將隨二維區間矩形面積增加而驟增。在本文中我們將捨棄這種掃描的方式，改以將矩形轉角座標點編碼來求解，在時間消耗上只與內部矩形的數量有關，與二維區間矩形面積大小無關，如此可有效率的解決二維區間閒置空間定位問題。

二、問題描述

考慮一固定大小之二維區間，其內含有 n 個座標位置固定之不重疊但相連矩形，二維區間閒置空間定位問題即欲求得這些沒被使用到的閒置空間區域的角落座標串列(依逆時鐘排序)。如圖 2 所示的配置圖，由圖中可知包含 7 個固定坐標之矩形(白色)，並且包含了 3 個閒置空間(灰色)，所提方法中能夠找出每一個閒置空間各角落座標串列。

三、解決方法

在此我們先介紹演算法流程中最重要的概念，我們稱之象限角編碼(Quadrant Encoding)，以方便我們描述各個角落所佔用的角度，如圖 4 可歸納出 16 種二元碼格

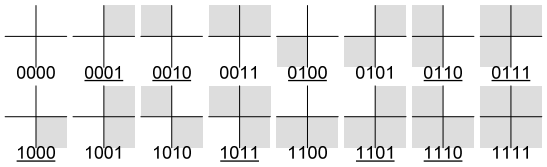

圖 4.16 種表示角落的編碼(binary)

式，其中下底線標示的是重要的格式一共有八種，這個編碼由第Ⅰ象限開始依循逆時針方向權重分為 1、2、4、8，舉例如圖 4 中上排左起第五個則將其編碼成 0*1+0*2+1*4+0*8=4。又如 0101(上排右起第三個)這個圖形格式代表佔用了Ⅰ跟Ⅲ象限，圖 5 列出幾個角落編碼實際情形。

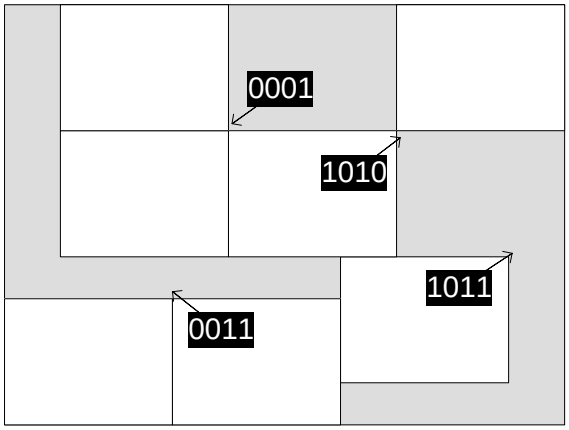

圖 5.編碼實際應用的例子

以下為我們解決二維閒置空間問題的五個步驟：

Step1：

拆解與編碼(Dismantling & Encode)

在這個步驟中將把每個矩形拆解成四個角落點的概念並且使用象限角編碼來描述它所佔用的象限角，我們將原先的圖 2 改寫如圖 6，我們在最外圍矩形反而以 D、

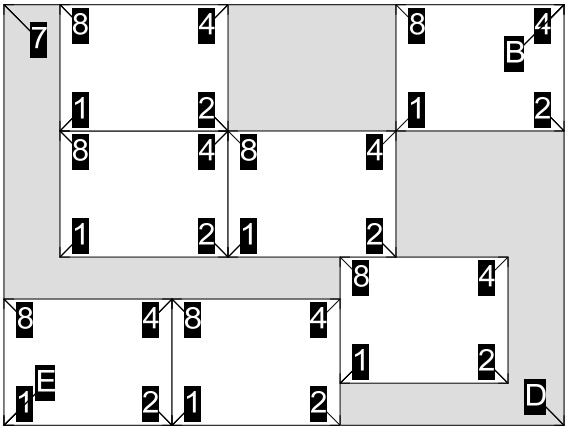

圖 6.對各角落做象限角編碼

B、7、E 的順序來編，這是因為我們把外圍矩形當成內部挖空得矩形所致。

Step2：

唯一與疊加(Unique & Superposition)

在這個步驟中我們把具有相同座標的象限角編碼予以疊加並且僅保留一組資料我們將上一步驟的圖形修改如圖 7。

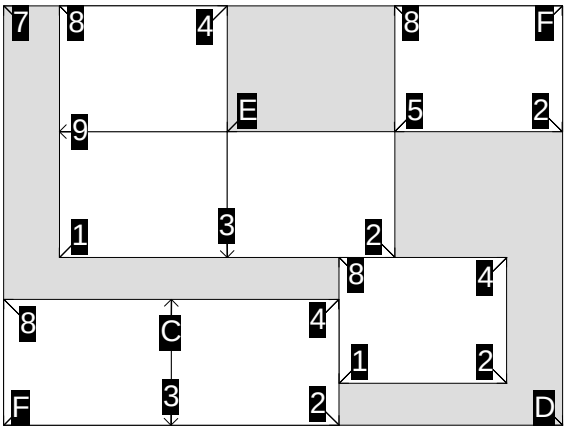

圖 7.將重複的座標予以疊加

Step3：

移除與分裂(Remove & Splitting)

我們對於上個步驟中象限角編碼為 3、6、9、C、F 予以移除，且把 5、A 一分為二分別為 1&4、2&8，我們將上一步驟的圖形修改如圖 8，我們刪除 3、6、9、

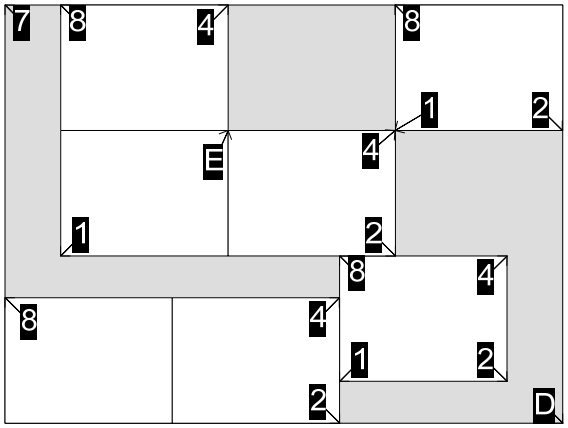

圖 8.僅保留轉角的資料

C 的原因為這些編碼為所謂的平角是不可能是閒置空間的轉角，刪除 F 的原因是滿角也不可能是閒置空間的轉角。

Step4：

邊緣修正(Edge Offset)

我們於上個步驟中發現有某些角落需要修正，原因為該角落剛好位於其他矩形的邊上，所以需要針對這些問題予以修正而這些編碼又正好是 1、2、4、8 所以僅需對這四種檢查他是否位於其他矩形的邊上，需要對這些邊緣做快速排序(quick sort)[4]，我們將對照圖詳列如圖 9，舉例來

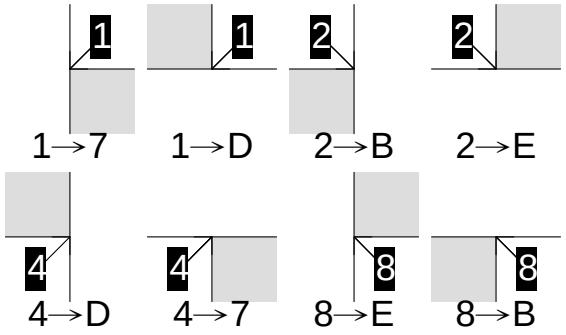

圖 9.邊緣補正的對照圖

說圖 8 左下角的編號 8 正好位於其他矩形的右側正好滿足上圖 8→E 的條件，所以在此步驟將改寫為 E 以符合真實狀態，我們將改寫後的圖詳列如圖 10。

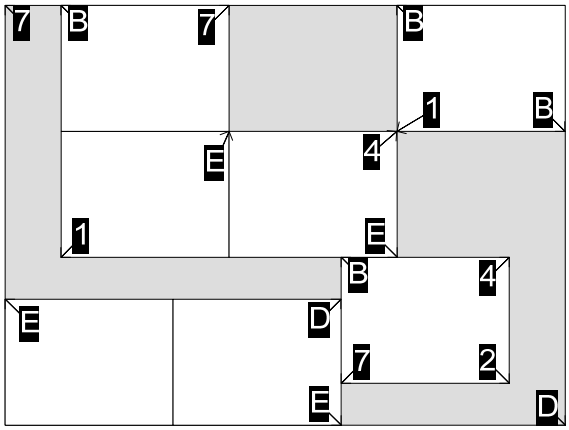

圖 10.進行邊緣補正後各點編碼值

Step5：

搜尋(Seraching)

首先我們必須把所有的編碼全部予以做 NOT 運算，原因在於之前的步驟都是用矩形的觀點投射在角落的象限角編碼上，需要給予轉換後用來表示閒置空間的角落，我們將圖 10 改寫如下圖 11，我們

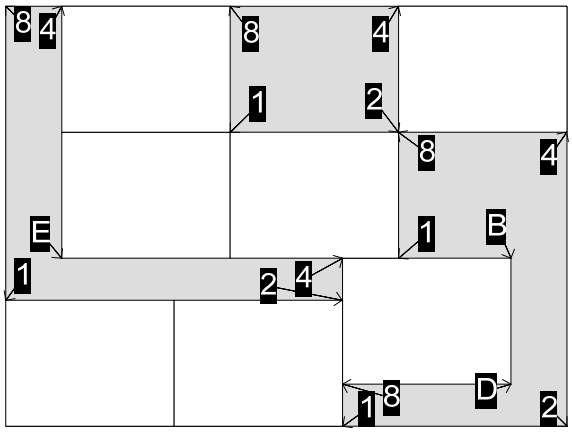

圖 11.執行 NOT 運算後的原圖

將依圖 12 方向對照圖將座標列出來，以象

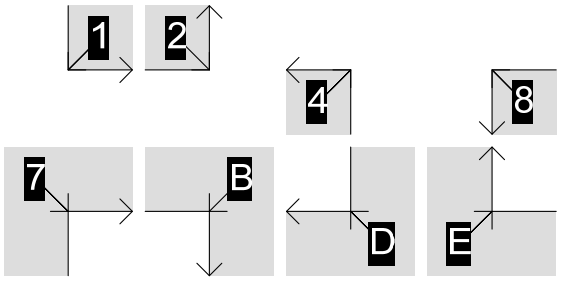

圖 12.編碼值方向對照圖

限角編碼為 1 做為起點，由方向對照圖查得必須要往上找最近的點即為下一點，亦

即象限角編碼為 8 或是 B，若該點存在必然有一個滿足此條件，若是無法找到任何點滿足此條件或是已經回到當初出發的點則代表此塊閒置空間角落座標點被列舉完畢，則可以進行下一塊的閒置空間的座標點列舉，此時則必須要刪除已經列舉過的角落座標點以加速其他閒置空間的搜尋速度。

四、實驗結果與討論

我們將利用如圖 13 之圖形做為測試例子，其中有 100 個大小不一的矩形以不重疊但相連方式擺放其中，其大小與位置皆固定。實驗結果顯示所提演算法一共找出 40 個大小不一的閒置空間如圖 14 所示，其中每個閒置空間的各角落座標皆可找出。

在實驗中我們發現時間消耗大多集中在步驟四邊緣修正上面這是因為我們並不知道該矩形的附近有哪些矩形以致需要使用逐一驗證的方式來處理此步驟，若日後能夠搭配矩形彼此之間的關係資料結構將能夠提升不少效率，在尚未搭配關係資料結構下時間複雜度為矩形數量的平方。

五、結論與未來展望

經由實驗結果得知我們的方法不僅有效且快速來解決二維區間閒置空間定位問題，未來將解決區間內有多個大小不一的矩形以不重疊但亦不相連方式擺放其中的問題。

六、參考文獻

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts", 2004

[2] P. Westerman, "Data Warehousing: Using the Wal-Mart Model", 2000.

[3] N. A. Sherwani, "Algorithms for VLSI Physical Design Automation", 1998.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms", 2001

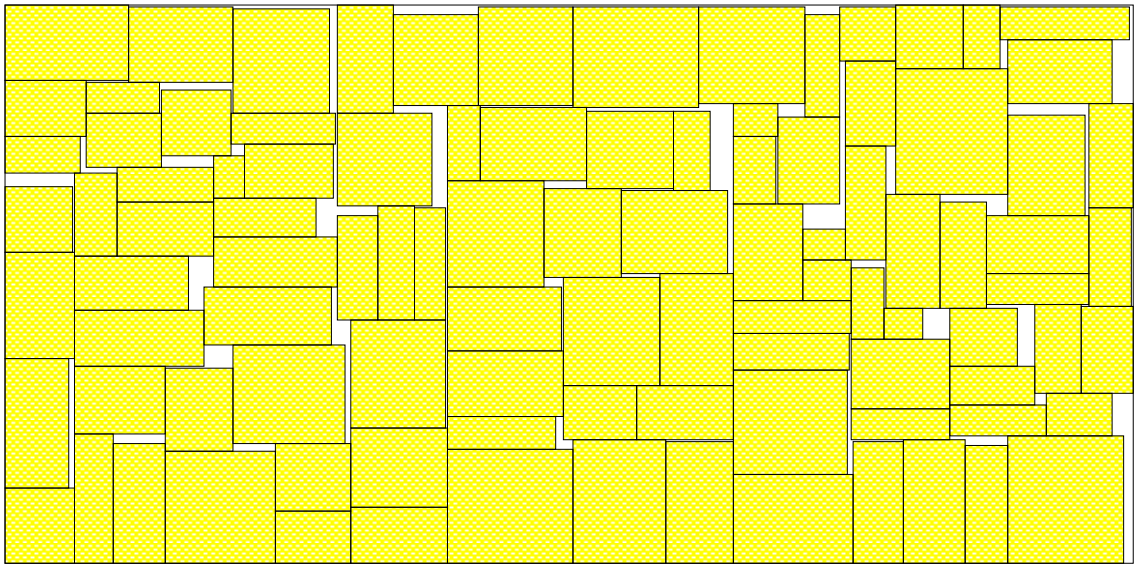

圖 13.初始化輸入的平面圖

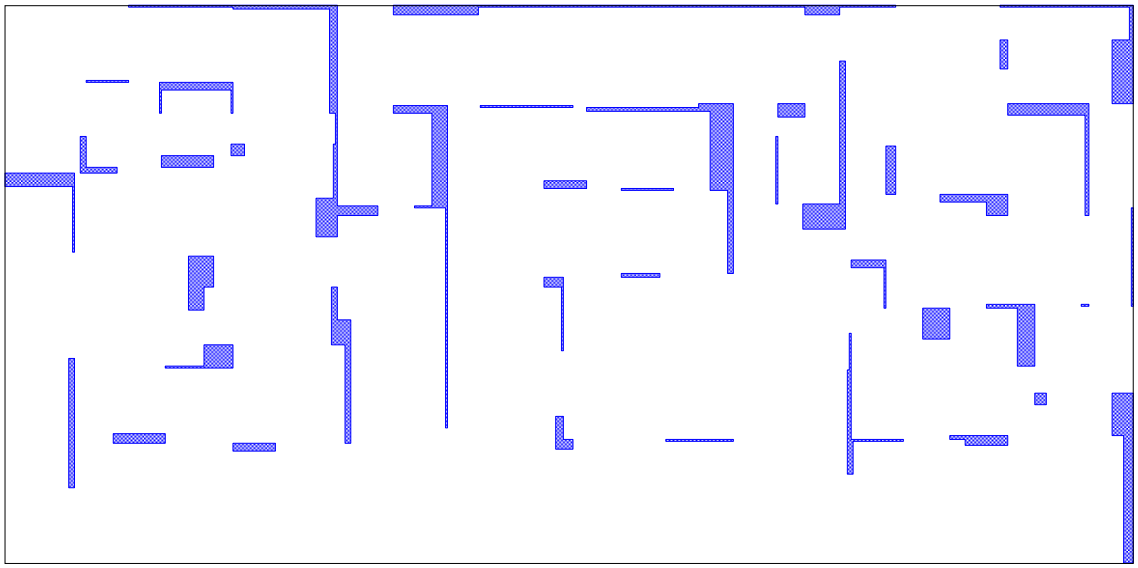

圖 14.輸出的二維區間閒置空間定位圖