2025年臺灣國際科學展覽會
優勝作品專輯

作品編號　190008

參展科別　電腦科學與資訊工程

作品名稱　矩形密鋪及其應用

得獎獎項　三等獎

國際科學博覽會(ESI)

就讀學校　國立臺灣師範大學附屬高級中學
指導教師　蔡孟宗
林俊卿
作者姓名　林郁城

關鍵詞　dynamic programming 、2D rectangle tiling problem、RTILE problem

作者簡介

本研究「矩形密鋪及其應用」的作者為林郁城，目前就讀師大附中科學班，喜歡研究並分析演算法，很投入本次的專題也收穫良多，在此特別感謝中研院的蔡孟宗教授及學校老師給予的諸多協助。

2025 年臺灣國際科學展覽會
研究報告

區　　別：

科　　別：　電腦科學與資訊工程學科

作品名稱：　矩形密鋪及其應用

關 鍵 詞：　dynamic programming、
2D rectangle tiling problem、
RTILE problem

編　　號：

摘要

「在格狀平面中用矩形以互不重疊的方式鋪滿(2D rectangle tiling problem)」為一 NP-complete 問題(Dani`ele Beauquier et al ,1995)，目前多項式時間只能求出盡可能覆蓋最大面積的近似解。本研究所創的階梯演算法 stair algorithm 透過改變動態規劃紀錄狀態的方式，使狀態數大幅減少，進而改善求準確解的時間複雜度，也成功證明此演算法的正確性。本研究的演算法可被應用於平行計算中的負載平衡、積體電路設計等方面。隨後，本研究寫了一個互動展示品清楚呈現此演算法的功能。且以階梯演算法成功檢驗並比較 RTILE PROBLEM 的 7/3-approximation algorithm (Krzysztof Lorys and Katarzyna E. Paluch,2000 [4]) 與 11/5-approximation algorithm (Piotr Berman et al,2001[7])進行比較與分析。

Abstract

The problem of tiling a grid plane with non-overlapping rectangles (2D rectangle tiling problem) is NP-complete (Danièle Beauquier et al., 1995). Currently, polynomial-time algorithms can only find approximate solutions that cover the largest possible area. In this research, the proposed stair algorithm significantly reduces the number of states recorded in dynamic programming, thereby improving the time complexity of finding exact solutions. The correctness of the algorithm has also been successfully proven. This algorithm can be applied to areas such as load balancing in parallel computing and integrated circuit design. Furthermore, an interactive demonstration was created to clearly present the functionality of the algorithm. Using the stair algorithm, this research successfully tested, compared, and analyzed the 7/3-approximation algorithm (Krzysztof Lorys and Katarzyna E. Paluch, 2000) and the 11/5-approximation algorithm (Piotr Berman et al., 2001) for the RTILE problem.

壹、 前言

一、研究動機

多米諾骨牌覆蓋問題，在討論以 1×2 的矩形，以不重疊而完全覆蓋一個 8×8 的棋盤之方法數，其中此問題有一變種，在詢問棋盤有殘缺的情況下，是否仍可完全覆蓋。

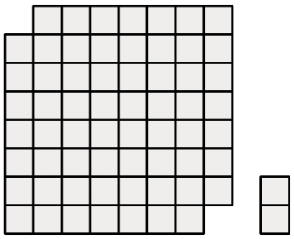


圖 1-1
(photo credit : 作者)

圖 1-1 的 8×8 棋盤的左上角右上角各被挖去一格，雖然剩餘 62 個格子看似有機會鋪滿，但有一巧妙的方式可以說明辦不到，便是將棋盤中黑色白色的方格分開來看，如**圖 1-2**，此圖中黑格子有 32 個，白格子有 30 個，而每個 1×2 的矩形會保證覆蓋黑白格子各一，由此便可說明殘缺棋盤無法被鋪滿。這讓我很好奇若用其他形狀的矩形去覆蓋殘缺棋盤，是否也能有如此漂亮的結果，例如以**圖 1-3** 斜線排列的圖色可以有相似效果。

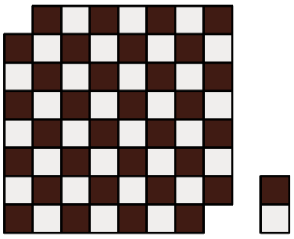


圖 1-2
(photo credit : 作者)

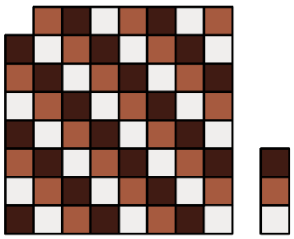


圖 1-3
(photo credit : 作者)

但此方式有一個極大的缺點，就是它只保證不符合條件者不存在排列方式，但符合條件者也有可能無解。**圖 1-4** 中我們可以發現，雖然黑白格子各有 3 格，但矩形無法將其鋪滿。我便想到也許程式能幫我們解決此問題。

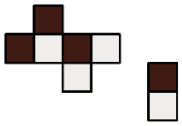


圖 1-4
(photo credit : 作者)

二、研究目的與文獻回顧

本研究所創的階梯演算法，本欲解決 2D rectangle tiling problem（即研究目的一），隨後基於演算法的特性，拓展至柱體側面與 RTILE PROBLEM，以下為問題的定義：

(一) 使用給定的多個矩形密鋪一個格狀平面(2D rectangle tiling problem)

> 輸入：一個 n×n 的二維 01 陣列和 k 組正整數 a、b，表示 k 種矩形的長、寬
>
> 目標：n×n 二維陣列表示一格狀平面，用給定的 k 種矩形密鋪此格狀平面，使得二維陣列中為 0 的位置皆被矩形覆蓋，而二維陣列中為 1 的位置皆未被矩形覆蓋。其中，矩形不可以被旋轉，可以重複使用。詢問是否能完成密鋪？

如果希望 a×b 的矩形密鋪時可以被旋轉，則將其視為 a×b、b×a 兩種矩形即可。

(Dani`ele Beauquier et al [3]，Tiling figures of the plane with two bars)文中說明，給定一個有限平面網格圖形和兩個矩形，1×A(horizontal bar)與 B×1(vertical bar)，其中 A、B 至少為 2。詢問是否能以這兩種矩形密鋪此平面網格圖形？

若 A、B 至少一個大於等於 3，則此問題為 NP-complete，若額外限制 A、B 皆為 2 (dominoe)，則此問題為 P。

由以上文中內容可知，Tiling figures of the plane with two bars 可被規約(reduce)至 2D rectangle tiling problem，且 2D rectangle tiling problem 的是否為合法的密鋪也可在多項式時間被驗證，故 2D rectangle tiling problem 為 NP-complete。

(二) 使用給定的多個矩形密鋪一個格狀柱體側面

與研究目的一相似，但不是密鋪格狀平面，而是密鋪格狀柱體側面，階梯演算法透過一遞減數列紀錄狀態，但頭尾相接的環會使此方式出現問題，而本研究巧妙地成功解決了這個問題。

(三) RTILE PROBLEM：以矩形密鋪一個格狀平面（不限制矩形的形狀）， 將每個矩形內格子裡的數字加總，使最大加總值最小化。

> 輸入：一個由非負實數構成的 n×n 二維陣列和一個正整數 p
>
> 目標：將此二維陣列切割成 p 個矩形且不重疊的子陣列，將子陣列內的數值加總得到 p 個子陣列權重和，目標是讓 p 個子陣列權重和中最大者最小。

(Grigni & Manne[1])證明了最佳化 n×n 陣列平鋪是 NP-hard，而 4/3 倍內的近似解也由(Grzegorz Gluch and Krzysztof Lorys,2017 [2])說明了仍是 NP-hard，若保證二維陣列內皆為整數也被(Sanjeev Khanna, S. Muthukrishnan, and Mike Paterson,1998[5])證得其仍為 NP-hard。

本研究的階梯演算法可以得到 RTILE PROBLEM 的正解。在此以隨機生成測資的方式，以階梯演算法將 7/3-approximation algorithm (Krzysztof Lorys and Katarzyna E. Paluch,2000 [4]) 與 11/5-approximation algorithm (Piotr Berman et al,2001[7])進行比較與分析。

貳、 研究設備及器材

一、硬體：

(一)筆記型電腦（ROG Zephyrus G14 GA401II_GA401II , CPU ： AMD Ryzen™ 7 4800HS Processor 2.9 GHz (8M Cache, up to 4.2 GHz) , GPU ： NVIDIA GeForce GTX 1650 Ti GDDR6 4GB）

二、軟體：

(一) Dev-C++ IDE（版本 5.11.0.0）

本研究的 C++程式在此軟體運行（互動展示品除外）。

(photo credit : 圖片來源[1])
圖 2-1

(二) Microsoft Visual Studio 2022

互動展示品的程式載體，與 OpenCV 連結。

(photo credit : 圖片來源[2])
圖 2-2

(三) OpenCV

用於互動展示品的製作，作為程式到使用者介面的橋樑。

(photo credit : 圖片來源[3])
圖 2-3

(四) Aseprite

用於互動展示品上的像素圖形繪製。

(photo credit : 圖片來源[4])
圖 2-4

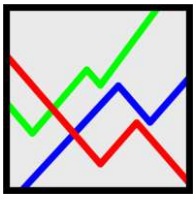
(五) CLIP STUDIO

用於解說圖的繪製。

(photo credit : 圖片來源[5])
圖 2-5

(六) gnuplot (版本 5.4.10)

用於圖表繪製



(photo credit : 圖片來源[6])
圖 2-6

參、 研究過程或方法

一、研究步驟與流程

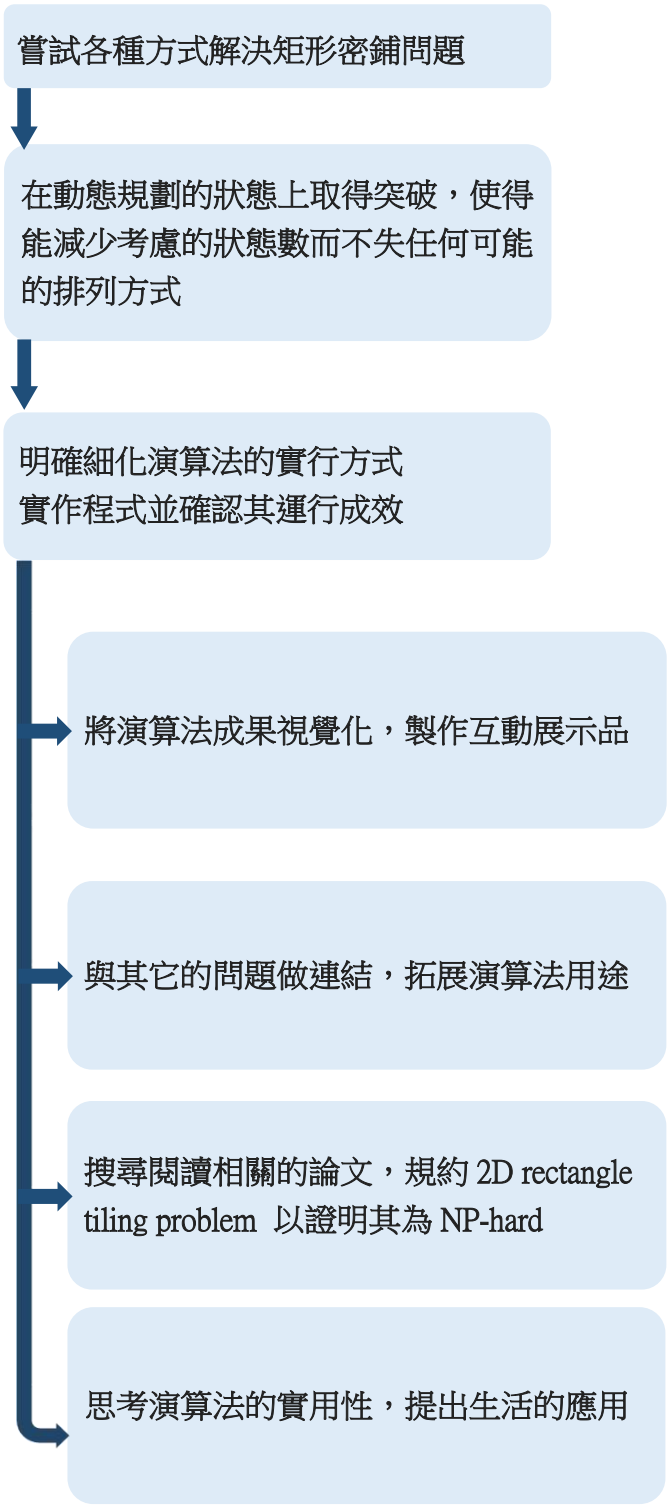


圖 3-1(photo credit： 作者)

二、階梯法的說明及證明

(一) 使用給定的多個矩形密鋪一個格狀平面

1. 定義：

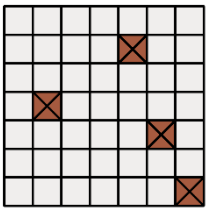


圖 3-2(photo credit : 作者)

格狀平面：一個 n×n 的二維陣列，二維陣列內的每個元素有兩種狀態，可放和挖空。「可放 □ 」是指矩形可覆蓋的元素，「挖空 ⊠ 」則是指矩形不可覆蓋的元素。

2. 引理：

對一個格狀平面，欲求得其是否可用矩形以互不重疊的方式鋪滿，可利用階梯法求得並得其排列方式。

3. 階梯法說明：

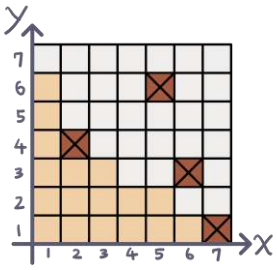


圖 3-3 中的數列 V 為 6,3,3,2,2,1,0 (photo credit : 作者)

我們將格狀平面視為由一大正方形挖空剩餘區域所形成的圖形。

我們以數列 V：$v_1,v_2,v_3,v_4,\cdots,v_n$ 表示一折線，在此稱為階梯，其中 v_i 表示 $x = i$ 時所對應的 y 值，並保證遞減 $\forall i < j$，$v_i \geq v_j$，以 V_0：0,0,0,0,⋯,0 作為初始階梯。當我們著手處理一個階梯，首先會嵌入挖空，方法如下：

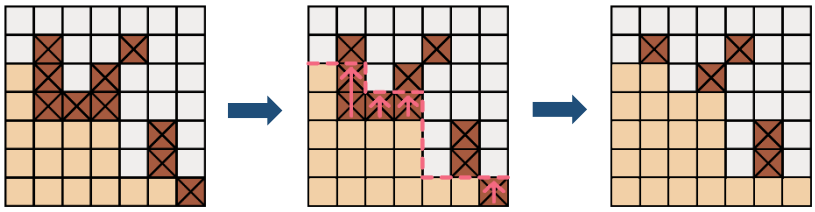


圖 3-4 嵌入挖空解說圖，為解釋填補挖空的諸多情況，另舉了一個挖空較多的例子 (註：與**圖 3-3**、**圖 3-5** 無直接關聯) (photo credit： 作者)

自 v_1、v_2、v_3、v_4…處理到 v_n，v_i在（i , v_i）上方若為挖空，則將 v_i加上 h，其中 h 表示該連續挖空的高，為保證遞減，若 $i \geq 2$ 則要求 $v_i \leq v_{i-1}$，如上圖所示。（意即若 $v_i+h > v_{i-1}$，則 $v_i = v_{i-1}$）

對鑲嵌好挖空的階梯 V，我們試圖在原階梯上放一個矩形，放入後形成的新階梯 V’須保持遞減特性，不可有鏤空、超出邊界、覆蓋挖空區域的狀況發生。

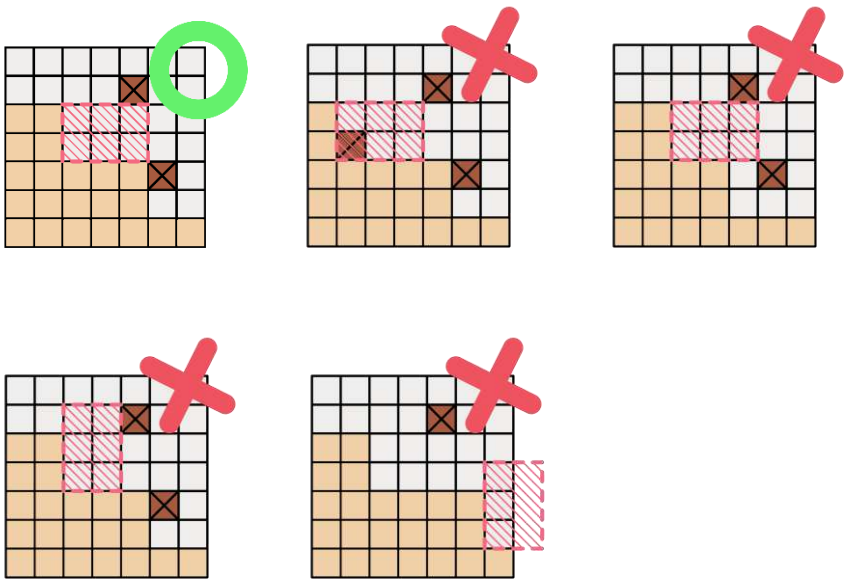


圖 3-5 在當前階梯放 2×3 或 3×2 的矩形
(photo credit： 作者)

對新產生的階梯進行相同操作，直到沒有未處理的階梯，此時若有 V：n,n,n,n,…,n 的數列，則表示此格狀平面可用矩形以互不重疊的方式鋪滿，而其堆疊的方式即為其中一種矩形的排列方式，此特性將在後續證明中被使用。反之，若無 V：n,n,n,n,…,n 的數列，則此圖形無法被矩形鋪滿。

範例：以 1×2 和 2×1 的矩形密鋪 3×3 的格狀平面

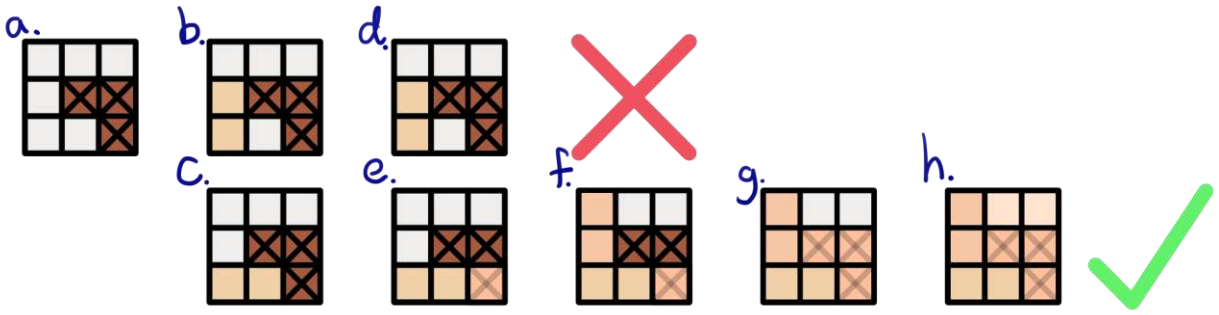


圖 3-6(photo credit：作者)

a. 嵌入挖空（無挖空可嵌入）

⇒ V：0 , 0 , 0

b. 放入第一塊矩形的第一種情況

⇒ V：2 , 0 , 0

c. 放入第一塊矩形的第二種情況

⇒ V：1 , 1 , 0

d. 嵌入挖空（無挖空可嵌入）

⇒ V：2 , 0 , 0（無法放入下個矩形）

e. 嵌入挖空（可嵌入其中一格）

⇒ V：1 , 1 , 1

f. 放入第二塊矩形（僅一種情況）

⇒ V：3 , 1 , 1

g. 嵌入挖空（可嵌入其中兩格）

⇒ V：3 , 2 , 2

h. 放入第三塊矩形（僅一種情況）

⇒ V：3 , 3 , 3（無挖空可嵌入）

出現 V：3 , 3 , 3 的階梯，成功以 1×2 和 2×1 的矩形密鋪 3×3 的格狀平面。

5. 時間複雜度：

n 為格狀平面的邊長、m 為所有矩形最長的寬、k 為矩形的種類數(k 為常數)

如果直接用一個格狀平面大小的二維陣列儲存當前狀態，則因為有 n^2 個格子，每個格子有「已放、未放」兩種狀態，故狀態數量會達到驚人的至多2^{n^2}個狀態，此時，就算每個狀態都能用 O(1)就解決，整體的時間複雜度就要 O(2^{n^2})。

如果了解到可已由格狀平面的一邊密鋪向另一邊而不影響結果，那就能想到使用一個長度為 n 的陣列來存狀態，此陣列的每個元素可能是 0 到 n 之間的整數，即每個元素有 n+1 種可能，此時狀態數為 $n^{(n+1)} \fallingdotseq n^n$，時間複雜度至少也是 O($n^n$)。

但如果使用階梯演算法，使用一長度為 n 的陣列來存，並保證每個元素大於等於下一個元素，則狀態數量來到C_n^{2n}，(∵可以想像從 n×n 格狀平面的左上角走到右下角，即排列 n 步向下及 n 步向右)，我們將其轉換成較常見的指數來表示，方法如下：

階梯動態規劃的狀態數量：

$$C_n^{2n} = C_n^{2n-1} + C_{n-1}^{2n-2} + C_{n-2}^{2n-3} + \cdots + C_1^n + C_0^{n-1} \quad (\because \text{巴斯卡定理})$$

$$\leq \sum_{k=0}^{n} C_k^{2n-1}$$

$$\leq 2^{2n-1}$$

$$< 2^{2n}$$

由此，我們一路將狀態數量從2^{n^2}、n^n降低至2^{2n}，進行了極大幅度的優化。

每個狀態花費時間為 O(n(n+m)k)∵在程式中放入方塊時，會從階梯 V：$v_1,v_2,v_3,v_4,\cdots,v_n$ 遍歷，遍歷時會紀錄一個數字 t，如果 v_{i-1} 的值與此 v_i 相同，則代表位於同一個階梯(同一高度)並執行 t=t+1，如果 v_{i-1} 的值與此 v_i 不同，則執行 t = 1，若 t 等於 k 個矩形中任一個的寬，則代表該矩形可放置於此處，會產生一個新階梯，產生此階梯花費 n+m(複製長度為 n 的陣列，修改 m 筆資料)。

故時間複雜度為 O(n(n+m)k $\cdot 2^{2n}$)

又∵n>m.∴時間複雜度為 O(kn^2 $\cdot 2^{2n}$)

此外，當格狀平面是 n×r 時，階梯動態規劃的狀態數量C_r^{n+r}，時間複雜度是 O(kn^2 · C_r^{n+r})

例如當 r = 3 時，時間複雜度是 O(n^2 · C_3^{n+3}) = O(n^5)。

→當 r = O(1)時，階梯演算法是**多項式時間演算法**。

6. 階梯法證明：

若圖形可用矩形鋪滿，則若能保證對任意階梯（初始階梯 V_0 除外），都能拔掉一塊矩形，使其仍保持階梯則得證。

在拔除矩形之前，我們會先清除嵌上去的挖空，依照我們階梯法嵌上挖空的方式，我們會把右方、上方都沒有矩形的挖空移除。

任選一 x = k，1≤k≤n，k∈Z，在 x = k 處，y 座標最大的矩形（即位於當前階梯表面）。

(1)若其上方、右方皆空，則可以拿掉此矩形仍會保持遞減階梯。反之，其上方或右方至少一邊有其他矩形。

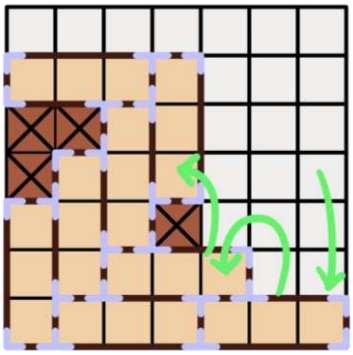


圖 3-7(photo credit： 作者)

(2)若上方有矩形，則我們改選為該矩形（若上方的矩形不只一塊，則選擇其中最靠右的）；若上方是挖空，則我們可改選挖空上方的矩形，我們能保證上方存在矩形，因為若其上方沒有矩形，則這個挖空會在之前被移除，因為右方、上方皆無矩形。接著，若此矩形上仍被壓住，則再改選為上方的矩形。重複此步驟直到所選的矩形上面沒被壓住，我們可以保證能夠依此方式找到沒被壓住的矩形，因為階梯最上面那塊矩形是沒被壓住的。此時，這個矩形上方、右方皆沒其他矩形，故拔除此矩形後仍會保持階梯。

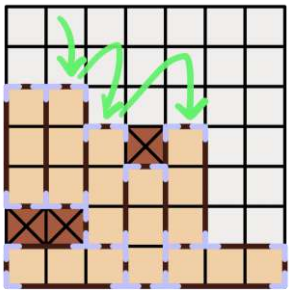


圖 3-8(photo credit： 作者)

(3)若右方有矩形，則我們改選為右方的矩形(若右方有多塊矩形，則選最上面那塊)，若右方是挖空，則我們改選挖空右方的矩形，我們同樣能保證挖空右方還有一塊矩形，因為若其右方沒矩形，則此挖空會在之前被移除。此外，我們能保證新改選的矩形上方沒被壓住，因為若上方被壓住，則左邊原本選的那塊會被整塊壓住，但依照我們一開始的選法，我們選不到整塊被壓住的矩形。重複往右邊選直到右邊沒有矩形。此時，矩形上方、右方都沒被壓住，拿掉此塊會保持階梯。

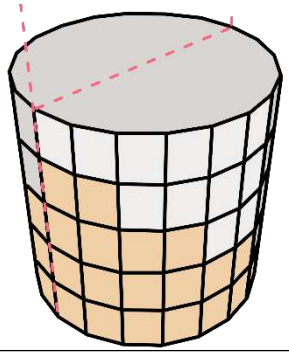


圖 3-9 (photo credit : 作者)

(二) 使用給定的多個矩形密鋪一個格狀柱體側面

1. 引理：

對一個格狀柱體側面，欲求得其是否可用矩形以互不重疊的方式鋪滿，可利用階梯法求得並得其排列方式。

2. 環狀階梯法說明：

如果有挖空，我們需要額外枚舉 v_1 的位置，因為環狀圖形沒有邊界決定 v_1 的位置。枚舉的數量為矩形較長的邊長。(不用枚舉一圈的原因在證明中有說明)

同樣以數列 V：$v_1,v_2,v_3,v_4,\cdots v_{n/2},\cdots v_n$ 表示階梯，其中 v_i 表示 $x=i$ 時所對應的 y 值，並保證在前半遞減 $\forall i<j\leq n/2$，$v_i \geq v_j$，在後半遞增 $\forall n/2<i<j$，$v_i \leq v_j$。我們將 v_1、v_n 之間的鉛直線稱為高線，$v_{n/2}$、$v_{n/2+1}$ 之間的鉛直線稱為谷線。

以 V_0：0,0,0,0,⋯,0 作為初始階梯。當我們著手處理一個階梯，首先會嵌入挖空，方法與平面階梯法相似。且我們額外要求第一塊矩形要被高線從正中央切過，因為若不在高線正中央，他會與枚舉 v_1 的其它狀況重複。

對鑲嵌好挖空的階梯 V，我們試圖在原階梯上放一個矩形，放入後形成的新階梯 V' 須保持遞增遞減特性，不可有鏤空、覆蓋挖空區域的狀況發生。（只要符合遞增遞減規則，一個矩形能橫跨遞增遞減區域）

對新產生的階梯進行相同操作，直到沒有未處裡的階梯，此時若有 V：n,n,n,n,⋯,n 的數列，則表示此環狀圖形可用矩形以互不重疊的方式鋪滿。反之，則此圖形無法被矩形鋪滿。

3. 環狀階梯法證明：

枚舉的數量為矩形較長的邊長，而不用到 n，是因為我們枚舉的目的是避免遺漏矩形的排列方式，我們可以得知以 k 作為 v_1 所得到的排列方式，會包含於以 k-a、k-b 作為 v_1 的排列方式之聯集（a、b 為矩形的邊長，其中 a≤b），依此 k>b 的狀況，都可以被拆解成 k-a、k-b 的聯集，故我們只需檢查 1≤k≤b 的排列方式即可。

若圖形可用矩形鋪滿，則若能保證對任意階梯(初始階梯 V_0 除外 ），都能拔掉一塊矩形，使其仍保持階梯則得證。

在拔矩形之前，我們會先清除嵌上去的挖空，遞減處依照我們階梯法嵌上挖空的方式，我們會把右方、上方都沒有矩形的挖空移除。遞增處則是左方、上方都沒有矩形的挖空。

任選一 $x = k$，$1 \le k \le n$，$k \in Z$，在 $x = k$ 處，y 座標最大的矩形（即位於當前階梯表面）。

若 k 在遞減處：

(1)若其上方、右方皆空，則拿掉此矩形仍會保持階梯。反之，其上方或右方至少一邊有其他矩形。

(2)若上方有矩形，則我們改選為該矩形(若上方的矩形不只一塊，則選擇最靠右的)，若上方是挖空，則我們可改選挖空上方的矩形，我們能保證上方存在矩形，因為若其上方沒有矩形，則這個挖空會在之前被移除。接著，若此矩形上仍被壓住，則再改選為上方的矩形。重複此步驟直到所選的矩形上面沒被壓住，我們可以保證能夠依此找到沒被壓住的矩形，因為最上面那塊是沒被壓住的。此時，這個矩形上方、右方皆沒矩形，故拿掉此矩形會保持階梯。

(3)若右方有矩形，則我們改選為右邊的矩形(若右邊有多塊矩形，選則最上面那塊)，若右方是挖空，則我們改選挖空右方的矩形，我們同樣能保證挖空右方還有一個矩形，因為若其右方沒矩形，則此挖空會在之前被移除。此外，我們能保證新改選的方塊上方沒被壓住，因為若上方被壓住，則左邊原本選的那塊被整塊壓住，但依照我們一開始的選法，我們選不到這塊，重複往右邊選又直到右邊沒有。此時，矩形上方、右方都沒被壓住，拿掉此塊會保持階梯。往右找時，找到最右方那塊並不能保證其能被移除，因為它有可能被谷線穿過而在遞增那邊上面被壓住，此時可依照遞增處向上找的方式，找到左方、上方皆空的矩形。

若 k 在遞增處則同理，因為遞增圖形的鏡像即為遞減，故與遞減的處裡方式相似，只是左右相反。

(三) 不同的格狀平面切割方式

在先前的階梯演算法中，我們都以固定矩形形狀的方式堆疊階梯，但實際上，此演算法並沒有這個限制，可以同時用多種矩形，當然也可自訂其他的約束條件來決定矩形的取法。此改動與挖空不衝突，故演算法仍可切割各式各樣的格狀平面或曲面。

例如，規定每個矩形內的權重合不能超過 k，並嘗試其是否能被排出，依此進行二分搜，所得的最小 k 即對應到矩形切割問題中，最小的最大子陣列合，此方式用以驗證 7/3-approximation algorithm (Krzysztof Lorys and Katarzyna E. Paluch,2000 [4]) 與 11/5-approximation algorithm (Piotr Berman et al,2001[7])。

肆、 研究結果

一、互動展示品

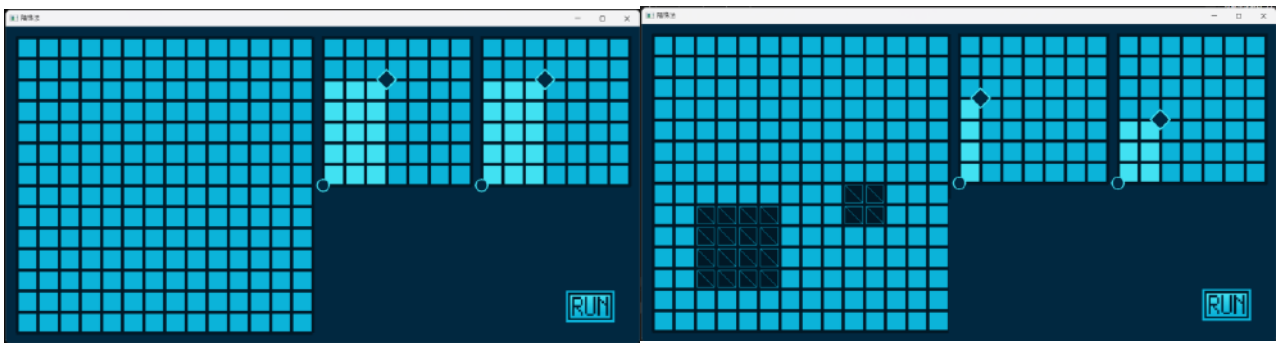


圖 4-1 互動展示品的操作示範(photo credit： 作者)

視覺化的呈現階梯法的作用，圖 4-1 左側為程式執行的起始畫面，其中大正方形為方格圖形，點擊或以滑鼠拖移過方格即可切換其為挖空，再次點擊即可變為可放狀態。右邊兩個亮色方塊即為用以密鋪的矩形(可旋轉)，拖移菱形即可改變矩形的大小。按下「RUN」後，階梯法的程式會開始運行，若存在排列方式，則會顯示 Yes 和其中一種排列方式，反之顯示 No。

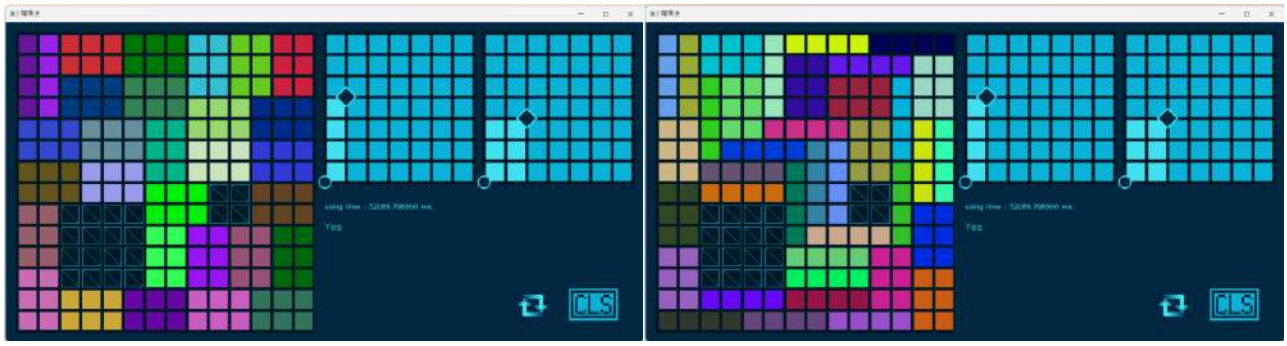


圖 4-2 互動展示品顯示的其中兩種排列(photo credit： 作者)

成功排列後，可按下「🔁」，使其隨機顯示一種排列方式。按下「CLS」則可以清空輸出並重新更改輸入。（此處可以顯示多種排列方式，並不是在 dp 表中紀錄每種排列方式，而是建立一個陣列儲存可以變為此狀態的階梯，事後再依此找回階梯上一步的所有情況，進而達到隨機顯示的效果。）

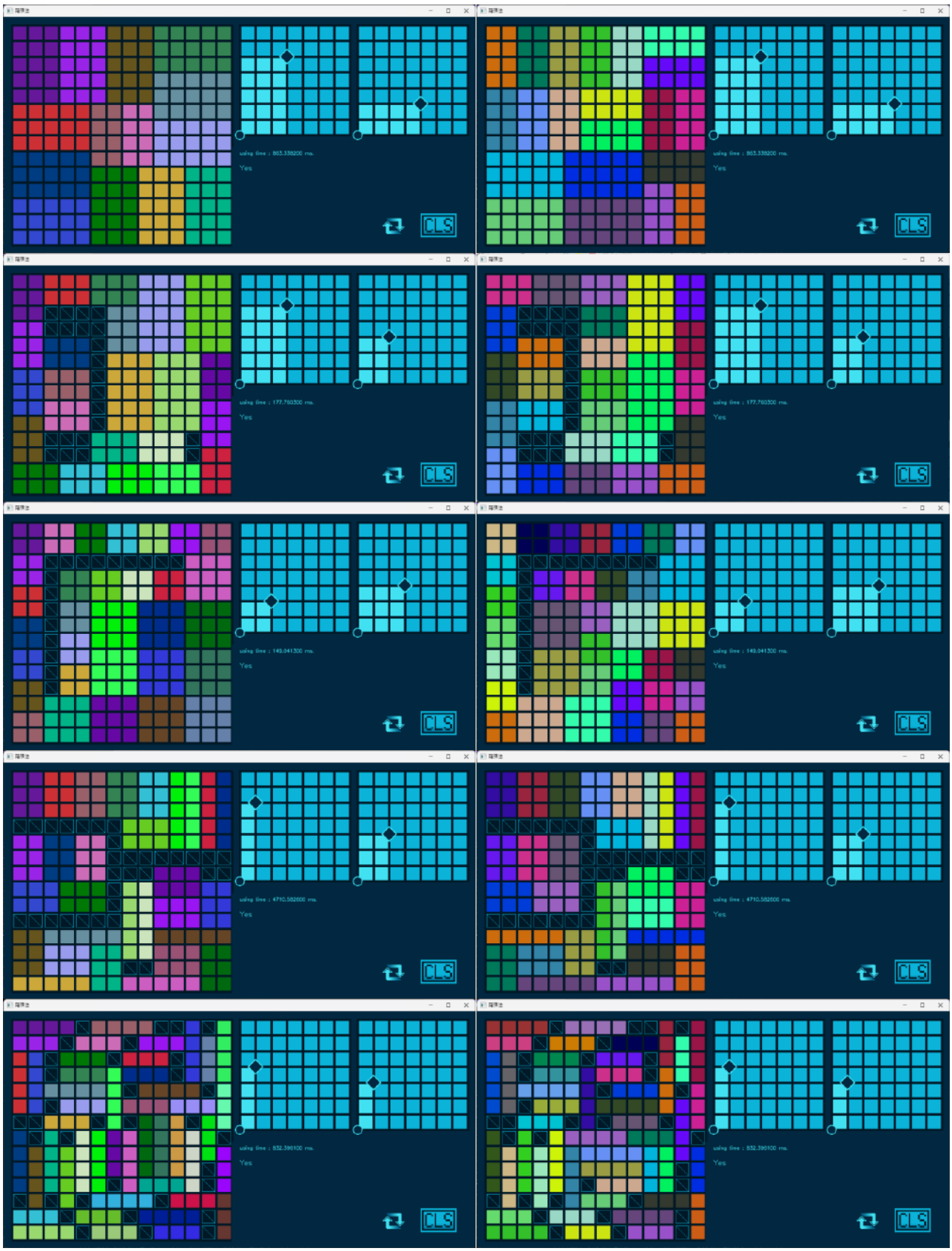


圖 4-3 互動展示品顯示的其他運行結果(photo credit： 作者)

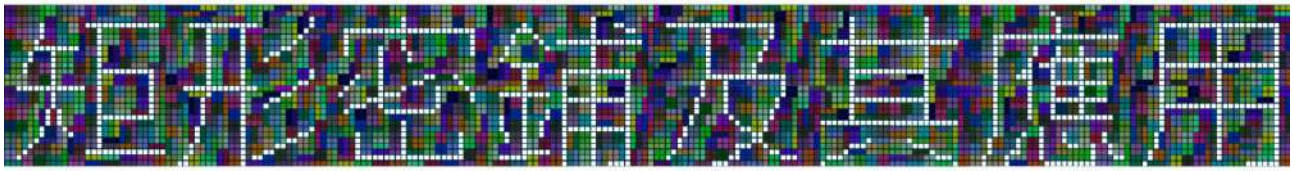


圖 4-4 為用挖空出本研究標題並密鋪剩餘區域的結果(為了讓字明顯，我有調整挖空的顏色荷和密鋪矩形的亮度) (photo credit： 作者)

二、驗證 RTILE PROBLEM 的 7/3-approximation algorithm (Krzysztof Lorys and Katarzyna E. Paluch,2000 [4]) 與 11/5-approximation algorithm (Piotr Berman et al,2001[7])並比較

本研究以 7×7 的陣列，裡面放有 0~99 的隨機整數，塊數 p 則取 1~14 的隨機整數，產生 10000 筆隨機測資，隨後分別讓本研究的演算法與近似解演算法執行並輸出最小的最大子陣列和 k。以下為所得的結果：

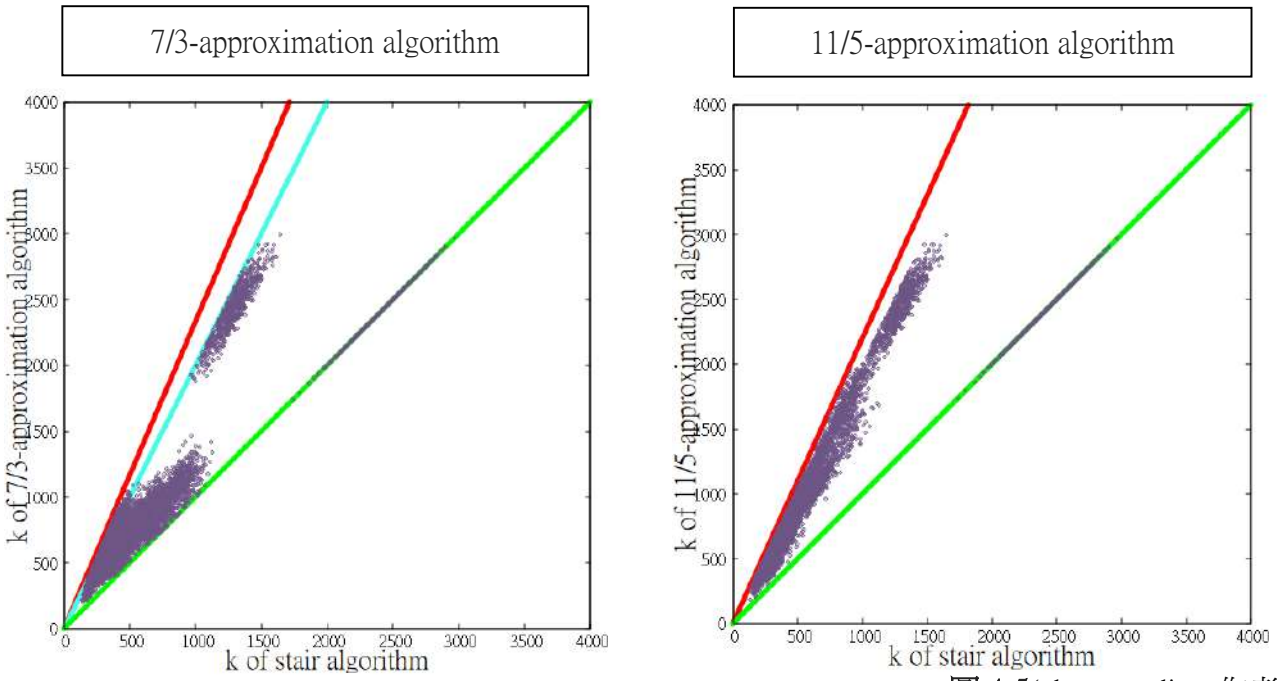


圖 4-5(photo credit： 作者)

圖 4-5 的每個點為其中一項測資，x 座標為階梯演算法求出的 k 值，y 座標為兩種近似解演算法求出的 k 值。在 7/3 近似演算法的圖中紅線斜率為 7/3、藍線斜率為 2、綠線斜率為 1。在 11/5 近似演算法的圖中紅線斜率為 11/5、綠線斜率為 1。

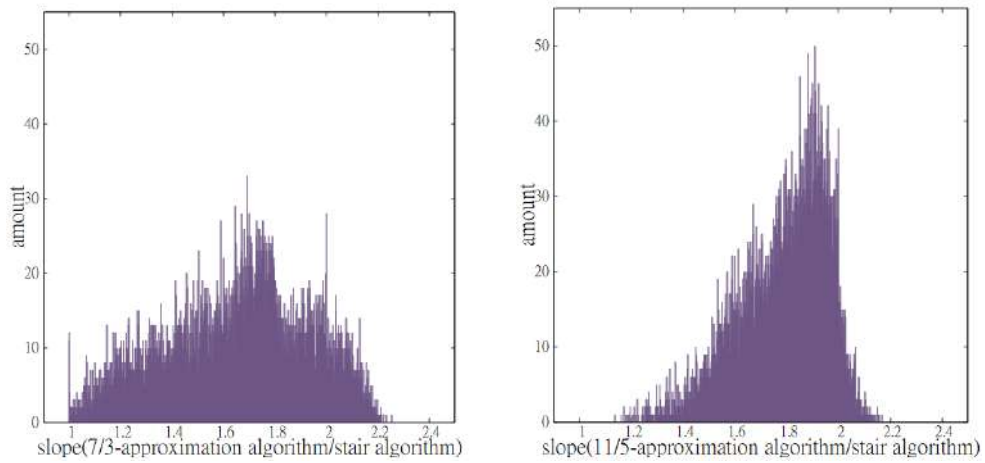


圖 4-6(photo credit： 作者)

approximation ratio	1.0~1.1	1.1~1.2	1.2~1.3	1.3~1.4	1.4~1.5	1.5~1.6	1.6~1.7	1.7~1.8	1.8~1.9	1.9~2.0	2.0~2.1	2.1~2.2	2.2~2.3
7/3	212	446	573	678	806	945	1226	1526	904	943	702	317	11
11/5	0	9	56	140	374	829	1203	1619	2221	2271	534	33	0

圖 4-7 為測資中兩種演算法的近似比分佈(去除 p=1)，其中 1.0~1.1 表示 1.0≦近似比＜1.1。

圖 4-6 為各 $\frac{\text{近似解演算法的 k 值}}{\text{階梯演算法的 k 值}}$ (即近似比)對應到點的數量(去除 p=1 的情況)。由此圖可以直觀的了解近似解的分布狀況。經統計發現，7/3 近似演算法平均近似比為 1.61，11/5 近似演算法平均近似比為 1.73。7/3 近似演算法的表現反而較佳。比較他們最壞的情況，11/5 近似演算法如期表現較佳。而兩演算法的結果也都如期在 7/3、11/5 以內。

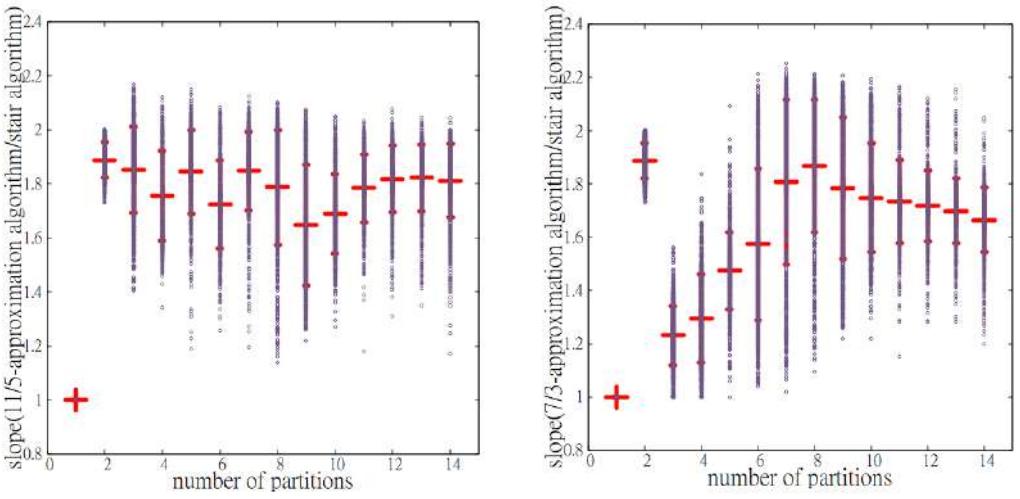


圖 4-8

(photo credit : 作者)

圖 4-8 的 x 座標為分割成矩形塊數，y 軸為近似比，即兩種演算法的近似表現。紫色點為資料，紅色橫線和直條是平均值與樣本標準差。可以看出 7/3 倍近似演算法在切割數較少時有特別出色的表現。

伍、討論

一、階梯演算法的更多發展

(一) Squared square

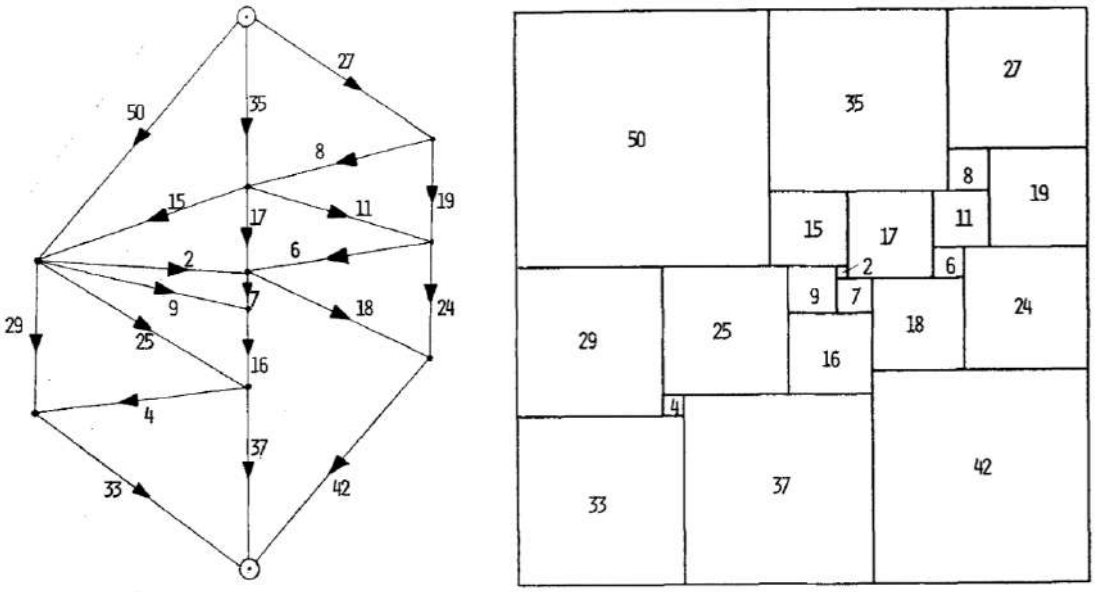


圖 5-1 photo credit : (A. J. W. Duijvestijn,1978[9])

此問題為用多個小正方形鋪滿一大正方形，其中小正方形的邊長不可重複，**圖 6-1** 為其中一種例子，而階梯演算法能透過改變矩形的約束條件（限制為正方形、限制數量），即可適用此問題，並找出更多存在的解。

(二) RTILE PROBLEM 上的拓展

除了上述兩篇演算法外，還有更多關於 RTILE PROBLEM 的演算法，例如(Sanjeev Khanna, S. Muthukrishnan, and Mike Paterson,1998[5])、(Jonathan Paul Sharp,1999[6])、(Piotr Berman et al,2001[7])…我們能以類似的方式去檢驗更多演算法。並進行更有系統地進行比較，例如本研究已檢驗的兩個演算法在隨機的測資中，7/3 近似演算法在平均的表現反而較 11/5 近似演算法佳，這從只討論最差情況的原始文本無法得知。而本研究的階梯演算法可以較快算出 RTILE PROBLEM 的準確解，進而大幅加快這類分析的運行。

二、階梯演算法的應用

(一)面積分配：

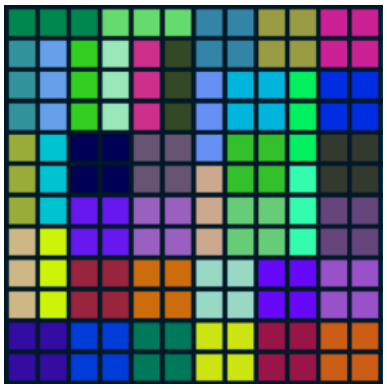
在一空間中分配土地，又有些土地不宜居住不應被分配，詢問分成 p 塊相同面積、相同形狀矩形土地的方式。可利用基本平面階梯法解決。相同的問題，若只要求相同面積，則改用有權重的階梯法，將每個方格的權重設為 1，並要求每次取矩形的權重合皆為 k 即可（k 為要求的面積大小）。這類問題在生活中常常出現，舉凡磁磚拼貼、房屋隔間、掃區分配等。

(二)資源分配：

在一土地中種植作物，希望每株作物得到養分養分最少者的養分 k 盡可能多，有些土地可能有水池、巨石、房屋等而無法耕作，今已探勘好各單位空間的養分含量，並給定希望種植的植物株數，求合理的分配方式。規定每個矩形上的權重合需大於等於 k，且長寬比例不可超過 2 倍（可依實際情況調整，此限制是避免出現過於狹長的矩形），根據階梯法結果進行二分搜，得到結果後，每個矩形中心即為作物的種植位置。水井設置、油田挖取等類似的問題都能使用。

圖 5-2 種植分配示意圖
(photo credit： 作者)

舉例：在晶片生產的應用



假設**圖 5-3** 是原本打算進行的晶圓裁切方式

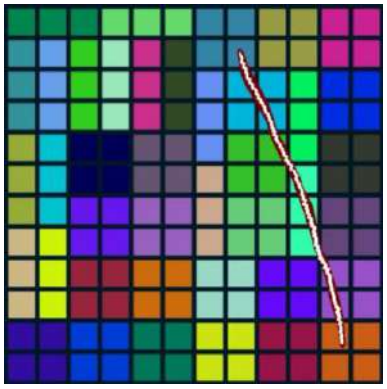


圖 5-4 為晶片量產時，其中一片晶圓，不小心產生一道刮痕。

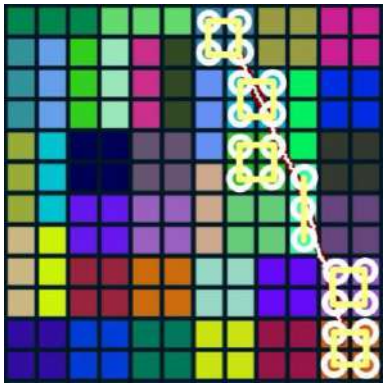


圖 5-5 如果按原定的方式切割，將導致 6 塊晶片受損無法使用，刮痕雖然只經過 11 個單位方格，但實際導致 23 單位方格無法使用。

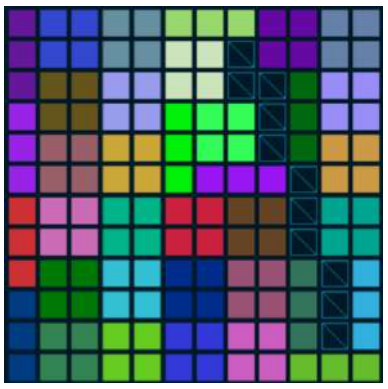


圖 5-6 可利用我們的階梯演算法，將受損處設為挖空，產生完美利用剩餘空間的方案。
在此案例中，本研究的階梯演算法減少了 **52%**因為此刮痕導致的虧損。

(**圖 5-3～6** photo credit： 作者)

陸、 結論

一、本研究創新的階梯演算法之特色

(一) 改善動態規劃紀錄狀態

狀態數從二維紀錄的2^{n^2}種狀態、一維紀錄的n^n種狀態降低至一維遞減數列紀錄的2^{2n}種狀態，進行了極大幅度的優化。

(二) 密鋪狹長形格狀平面

當格狀平面是 n×r 時，r 為常數，階梯演算法是多項式時間演算法。

(三) 處理挖空的能力

處理挖空意味著階梯演算法，不再侷限於用矩形去密鋪一個大正方形，而是任意的格狀平面都能處理，這讓演算法有更廣泛的應用空間。

(四) 於柱體側面上應用

透過將矩形側面切割成遞增、遞減區域，我們巧妙的解決了連成環後，遞減數列會出問題的狀況，讓此密鋪演算也能於柱體側面這類環狀表面進行密鋪。

(五) 保持矩形選取的自由度

階梯演算法並不會額外限制矩形的選取方式，這讓我們能處理類似的密鋪問題，例如本研究所討論的 RTILE，它的要求是格狀平面的每個格子上有權重，要根據權重去切割平面，而非根據特定的矩形形狀去切割。

柒、參考文獻資料

一、參考文獻

[1] M. Grigni, F. Manne, On the complexity of generalized block distribution, Proc. 3rd intern. workshop on parallel algorithms for irregularly structured problems (IRREGULAR’ 96), Springer, 1996, LNCS 1117, 319-326.

[2]Grzegorz Gluch and Krzysztof Lorys. “4/3 Rectangle Tiling lower bound”.In: CoRR abs/1703.01475 (2017). arXiv: 1703.01475. url: http://arxiv.org/abs/1703.01475.

[3]Dani`ele Beauquier et al. “Tiling Figures of the Plane with Two Bars”. In: Comput. Geom. 5 (1995), pp. 1 – 25. doi: 10.1016/0925-7721(94) 00015-N. url: https://doi.org/10.1016/0925-7721(94)00015-N.

[4] Krzysztof Lorys and Katarzyna E. Paluch. Rectangle tiling. In Klaus Jansen and Samir Khuller, editors, Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Saarbr¨ucken, Germany, September 5-8, 2000, Proceedings, volume 1913 of Lecture Notes in Computer Science, pages 206 – 213. Springer, 2000.

[5] Sanjeev Khanna, S. Muthukrishnan, and Mike Paterson. “On Approximating Rectangle Tiling and Packing”. In: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA. Ed. by Howard J. Karloff. ACM/SIAM, 1998, pp. 384 – 393. url: http://dl.acm.org/citation. cfm?id=314613.314768

[6] Jonathan Paul Sharp. “Tiling Multi-dimensional Arrays”. In: Fundamentals of Computation Theory, 12th International Symposium, FCT ’ 99, Iasi, Romania, August 30 - September 3, 1999, Proceedings. Ed. by Gabriel Ciobanu and Gheorghe Paun. Vol. 1684. Lecture Notes in Computer Science. Springer, 1999, pp. 500 – 511.

[7] Piotr Berman et al. “Efficient Approximation Algorithms for Tiling and Packing Problems with Rectangles”. In: J. Algorithms 41.2 (2001),pp. 443 – 470. doi:10.1006/JAGM.2001.1188. url: https://doi.org/10.1006/jagm.2001.1188

[8] Katarzyna E. Paluch. “A 2(1/8)-Approximation Algorithm for Rectangle Tiling”. In: Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings. Ed. by Josep D´ıaz et al. Vol. 3142. Lecture Notes in Computer Science. Springer, 2004, pp. 1054 – 1065.

[9]A. J. W. Duijvestijn. “Simple perfect squared square of lowest order”. In: J. Comb. Theory, Ser. B 25.2 (1978), pp. 240 – 243. doi: 10.1016/ 0095 - 8956(78) 90041 - 2. url: https : / / doi . org / 10 . 1016 / 0095 - 8956(78)90041-2.

二、圖片來源

[1] Dev C++ Logo Icon PNG Transparent Background 取自 https://www.freeiconspng.com/img/28406

[2] Zt-freak.(2022 年 10 月 4 日). Logo mark of Visual Studio 2022.取自 https://visualstudio.microsoft.com/wp-content/uploads/2021/10/Product-Icon.svg

[3] (2020 年 12 月 31 日). opencv.取自 https://iconduck.com/icons/27753/opencv

[4] (2020 年 12 月 31 日).aseprit.取自 https://iconduck.com/icons/80137/aseprite

[5] CSP Clip Studio Icon.取自 https://www.versluis.com/csp-clip-studio-icon/

[6] (2020 年 12 月 31 日). file type gnuplot.取自 https://iconduck.com/icons/101969/file-type-gnuplot

【評語】190008

本作品是一個延伸性研究。有基本理論, 亦有實際應用價值。為精益求精，下列建議供作者參考：

1. 部分階梯理論應詳細說明或舉例，如何使用本作品的演算法求解。

2. 可探討由 2D 延伸至 3D 或 4D 應用的可能。