國立臺灣師範大學
資訊工程研究所碩士論文

指導教授：　林順喜　博士

提升點燈遊戲之拼圖法搜尋速度

Improving the Solving Time of the Patching Method for All-Ones Problem

研究生：　蔡宗賢　撰

中華民國　一百　年　六 月

摘 要

提升點燈遊戲之拼圖法搜尋速度

蔡宗賢

點燈遊戲，又名 all-ones problem，最早是由 Sutner 於 1988 年提出的論文中所定義的名詞。這個問題假設有一個 N×M 大小的棋盤狀盤面，一開始是全暗，當我們點其中一格燈泡時，與它上、下、左、右相鄰的鄰居包含自己的狀態會由亮變暗或是由暗變亮，這條規則稱為 σ^{+}-rule。此遊戲的目標是找出一組點燈法則使得整個盤面上每一格子都變成全亮。

本論文將先前陳昱璇的論文所提出的縮小線性系統之演算法與陳昶安同學所提出拼圖法之理論作一種結合，使得收集拼圖法所需的基本盤面解能更加快速。成果方面，針對 N×M 的盤面，N 與 M 為整數，從原本拼圖法 $N \leq 20$ 且 $M \geq 1$ 已得到解提升到 $N \leq 63$ 且 $M \geq 1$ 都已破解。

關鍵詞：縮小線性系統、拼圖法 、全一問題、點燈遊戲

ABSTRACT

Improving the Solving Time of the Patching Method for All-Ones Problem

by

Tsung-Hsien Tsai

The lamp lighting game, also known as all-ones problem, was first proposed in 1988 by Sutner. The problem can be described as follows: suppose that there is an N×M chessboard, each square represents a light bulb. When we press on one square, the status of each of its non-diagonal adjacent squares including itself will be changed from lit to unlit or from unlit to lit. This rule is called the σ^+-rule. Initially all squares are in unlit status. The goal is to identify a set of squares that need to be pressed in order to turn all lights on.

In this paper, we combine the reduced linear system proposed by Yu-Xuan Chen and the patching method proposed by Chang-An Chen to improve the solving time substantially. Then we are able to find all solutions for the chessboards with a size N×M, $1 \leq N \leq 63$ and $1 \leq M$.

Keyword：reduced linear system, patching method, all-ones problem, lamp lighting problem

目錄

摘 要 i

ABSTRACT ii

目錄 iii

附圖目錄 v

第一章　簡介 1
第一節　問題描述 1
第二節　研究目的與背景 3
第三節　論文組織 5
第二章　相關研究探討 6
第一節　全一問題在任意圖中必定有解之證明 6
第二節　線性代數求解的方法 7
第三節　搜尋演算法 9
第四節　有限狀態機 11
第五節　縮小的線性系統 13
第六節　拼圖法 17
第三章　改進的演算法 22
第一節　進一步的加速方法 22
第二節　縮小的線性系統與拼圖法之優缺點 29
第三節　N×P 盤面之拼圖法 31
第四節　證明 N×P 盤面之存在性 36
第五節　程式環境與流程圖 38
第六節　處理時間 40
第七節　程式介面 42
第四章　結論 46
第一節　結論與未來研究 46
第二節　研究成果 47
附錄 A 列出本法求出的一些小盤面的解 48
參考文獻 59

附表目錄

表 2-1　對照表 19
表 3-1　對照表 31
表 3-2　收集盤面所耗費的時間 41
表 4-1　成果表 47

附圖目錄

圖 1-1　σ^{+}-rule 1
圖 1-2　5×5 all-ones problem 的初始盤面，所有燈全暗 2
圖 1-3　黑點為此 5×5 all-ones problem 的一種解答 2
圖 1-4　Tiger Toys 的 Lights Out 2000 [9] 3
圖 2-1　3×3 的盤面，從左上開始依序以 A 到 I 的字母順序編號 7
圖 2-2　3×3 盤面轉成圖 G 8
圖 2-3　圖 G 轉化成相鄰矩陣 T 8
圖 2-4　產生第 1 列第一種點燈法 10
圖 2-5　重新產生第 2 列 10
圖 2-6　退回去重新產生第 1 列 10
圖 2-7　找到一組點燈解法 11
圖 2-8　有機狀態 12
圖 2-9　3×3 的盤面 13
圖 2-10　第二列與第一列之關係 13
圖 2-11　第三列與第一列之關係 14
圖 2-12　3×3 盤面第一列點燈法對於整個盤面的影響 14
圖 2-13　3×3 盤面解 16
圖 2-14　4 種 3×2 盤面解 16
圖 2-15　5×5 的內部矩形是一個 complete state 17
圖 2-16　推導過程 18
圖 2-17　若 Row0≠Row10，則不可能產生 Row12 19
圖 2-18　5×53 盤面解 20
圖 2-19　5×28 盤面解 21
圖 3-1　5×8 盤面解 23
圖 3-2　5×6 盤面解 24
圖 3-3　5×7 盤面解 24
圖 3-4　鏡射圖 24
圖 3-5　N = 20 之欄位編號 26
圖 3-6　此圖即為 3-7 第一列點燈法 27
圖 3-7　20×81 盤面解 28
圖 3-8　5×7 盤面解 30
圖 3-9　8 種 Row1 在 3×5 盤面，都可找出全一問題的解 32
圖 3-10　收集 5 種基本盤面解 33
圖 3-11　3×6 盤面解 34
圖 3-12　3×12 盤面解 34

圖 3-13　k = 1，盤面解之組成 35
圖 3-14　k = 2，盤面解之組成 35
圖 3-15　N = 3，R_2 的 L_2 = 5 36
圖 3-16　循環長度之情況 37
圖 3-17　將 Row0 也列為循環長度 37
圖 3-18　程式環境與電腦配備 38
圖 3-19　流程圖 39
圖 3-20　程式介面 43
圖 3-21　顯示完整的盤面解 44
圖 3-22　顯示盤面解之第一列點燈法 45

第一章　簡介

第一節　問題描述

假設有一大小為 N×M 的棋盤狀盤面，盤面上每一格代表一個燈泡，而每盞燈只有燈亮與燈暗兩種狀態。當我們在盤面上任點一格，其上、下、左、右的鄰居包含自己的狀態會同時改變，也就是亮的變暗，暗的變亮，這條規則稱為σ^+-rule，如圖 1-1 所示，一個 5×5 的陣列如圖 1-1(a)，如果我們在 C3 那格點了一下（以黑圓點表示），則 C2、C4、B3、D3 跟 C3 都會變亮(如圖 1-1(b)，以灰色格子表示)，如果我們繼續在 D4 點一下，則 D4、E4、D5 會變亮，而原本就是亮的 D3 跟 C4 則會變暗(如圖 1-1(c)，以空白格子表示變暗)，當我們將表格中所有的格子都點亮表示得到一組解。

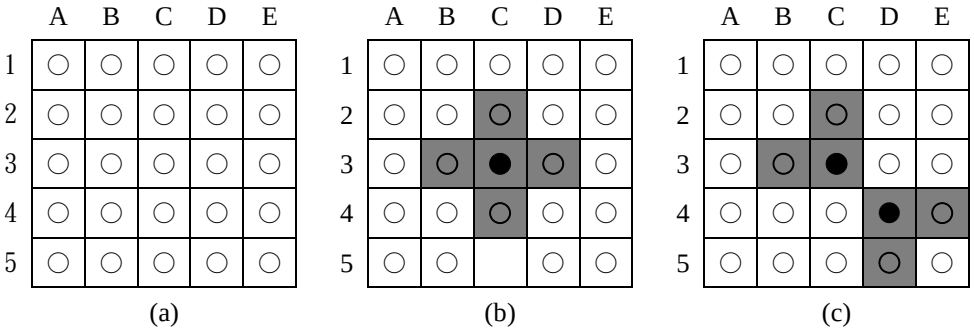


圖 1-1　σ^+-rule

給定一初始狀態全為暗的盤面（如圖 1-2），若能找到某種點燈法則（如圖 1-3），使得整個盤面點成全亮即達成目標。這就是所謂的全一問題（以下統稱 all-ones

problem)，也就是網路上常看到的點燈遊戲(all lights 、lights out game)。而如何找出最少點燈數的解法的問題叫做最小全一問題(minimum all-ones problem)。

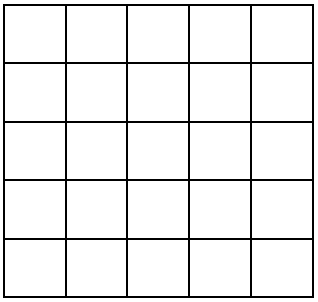


圖 1-2　5×5 all-ones problem 的初始盤面，所有燈全暗

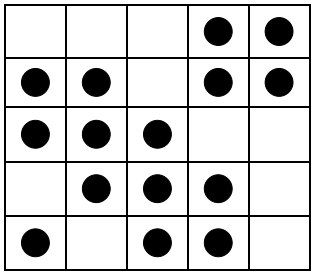


圖 1-3　黑點為此 5×5 all-ones problem 的一種解答

若以圖論的觀點來看這個問題，給定一 graph G = (V, E)，其中 V 為 G 上面的 vertex-set，E 為 edge-set。X 為 V 的子集合，若 V 中每一個 vertex $v \notin X$ 的四周有奇數個 vertex $x \in X$ 與之相鄰，且 X 中每一個元素的四周包含自己在內都有奇數個 vertex $x \in X$ 與之相鄰，則 X 即為該盤面的一組解。其中 X 就叫做 odd parity cover，或稱 odd dominating set(簡稱 ODS)。

all-ones problem 有一些值得注意的特性。第一，同一格子按偶數次將會回到它原先的狀態，其作用等同於沒有按過。第二，點燈的順序改變並不會影響結果，也就是說，我們所找的解答是一組點燈的組合(combination)，而不是排列

(permutation)。

另外,假使初始盤面是一不全為亮的盤面,也就是有些燈是亮的有些是暗的,是否能夠找到一組解能將那些亮著的燈變暗,最終達到所有燈全為暗的盤面,這個問題稱為全零問題(all-zeros problem)。Tiger Toys 公司所生產的著名的電子玩具"lights out"系列便是以盤面大小為 5×5 的全零問題為基礎而製作[9]。以圖論的觀點來說,這個問題其實是在找圖中的 even dominating set(簡稱 EDS),但是並非任意的圖都可以找到一組 EDS,也就是說並非所有的初始盤面都會有解,因此許多的研究者投入在找出有解的初始盤面的研究,其中最有名的是 John Goldwasser 與 William F. Klostermeyer 等人利用線性代數去分析,見[5]。

圖 1-4　Tiger Toys 的 Lights Out 2000 [9]

第二節 研究目的與背景

點燈遊戲,又名 all-ones problem,最早是由 Sutner 於 1988 年提出的論文[7]

中所定義的名詞，這個問題假設有一個 N×M 大小的棋盤狀盤面，當我們點其中一格燈泡時，與它上、下、左、右相鄰的鄰居包含自己的狀態也就是會由亮變暗或是由暗變亮，這條規則稱為 σ^{+}-rule，而最終目的是找出一組點燈法則使得整個盤面上每一格子之上、下、左、右相鄰的鄰居包含自己的燈泡數總數為奇數，也就是盤面狀態由開始的全暗變成全亮。並於 1989 年利用 cellular automata 證明了任意大小的 N×M 的盤面都必定有解。Lossers 也利用線性代數證明了任意大小的 N×M 的棋盤狀盤面的 all-ones problem 都必定有解的事實[6]，因此後來許多研究者都是利用線性代數來分析這個問題，但其解法需耗費 $O(\mathbf{N}^3 \times \mathbf{M}^3)$的時間。

Sutner 接著又提出了 minimum all-ones problem 也就是尋找最少點燈數，即求最佳解的問題，並証明了任意盤面的 minimum all-ones problem 是一種 NP-Complete 的問題[8]，也就是說很難找出多項式時間的解法。但是對於某些較簡單的盤面而言，卻仍然有可能找出多項式時間甚至線性時間的演算法來求解。Chen 等人提出了一種針對求解 minimum all-ones problem 於樹狀圖以及 unicyclic、bicyclic graphs 的盤面的線性時間演算法[4]。

本校陳昶安同學與林順喜教授提出了一種搜尋演算法[1]能夠有效率的利用拼圖法使得 all-ones problem 在 20×M 以下之盤面上都可以快速求得解。演算法[1]找尋拼圖法所需的盤面解，找尋的過程是屬於窮舉法，第一列的點法不同但只要依照[2]就能得到一個盤面解，當第一列的行數每增加一行的時候，所能形成的盤面解之總數卻呈指數成長，盤面所組合成的循環長度雖然有增有

減，但整體來看卻是逐漸增加使得所需時間甚至比指數成長還多。

本論文將縮小線性系統之演算法與拼圖法之理論作一種結合，使得收集拼圖法所需的基本盤面解能更加快速，進而將 N×M 盤面的破解從 N = 20 提升到 N = 63，且 M ≥ 1。

第三節 論文組織

本篇論文的組織架構如下：第一章先描述什麼是 all-ones problem，做一些詳細的定義與解說,並介紹這個領域的研究背景、前人的一些研究成果與重要發現。第二章會詳述重要的證明，以及現有的求解 all-ones problem 的演算法。第三章是本篇論文的重點，也就是我們所提出的新的改良演算法，能夠將 N×M 盤面的破解從 N = 20 提升到 N = 63，且 M ≥ 1。最後一章將會把整篇論文做個總結，並提出未來可能的研究議題。

第二章　相關研究探討

第一節　全一問題在任意圖中必定有解之證明

all-ones problem 在任意圖中都必定至少存在一組解，這是已經被證明的事實[8]。接下來將介紹如何證明這件事。

給一 graph　G = (V,　E)，其中 V 為 G 上面的 vertex-set，E 為 edge-set。將 G 轉成相鄰矩陣M，並令向量X 為all-ones problem 的一組 解。 假 設 1 代表 燈亮 的狀 態 ，0 代表 燈暗 的 狀態 ，則 令 **1** = (1, 1, …, 1)這個向量代表初始盤面的狀態，而**0** = (0, 0, …,0)這個向 量 代 表 終 止 盤 面 的 狀 態 。 而 σ 這 個 函 數 代 表 的 就 是 σ^{+}-rule。

我們要先證明一件事，就是當存在一個X 使得σ(X) = **1** 為真時，若且唯若，向量**1**與σ的核空間(nullspace)互為正交(orthogonal)。由於σ具有selfadjoint的性質，所以 <σ(X), γ> = < X, σ(γ)> 成立。Z為G上任意一個pattern，**X = 0**若且唯若 < Z, X >= 0 。令W為σ 的值域(range)

γ為$W^{\perp}$ 上的一組向量$\Leftrightarrow$<σ(X), γ> = 0

$\Leftrightarrow$< X, σ(γ) > = 0

$\Leftrightarrow \sigma(\gamma) = 0$

也就是說，假設$K_{G\sigma}$為 σ 的 kernel, 則$W^{\perp} = K$。根據定理$(W^{\perp})^{\perp} = W$ ，所

以$W = K_{G\sigma}$。因此上述命題成立。

假設 Y 是 σ 之核空間的一組基底，任意一組 Y 使得σ(Y) = 0 成立，根據 Handshaking 原理[8]，Y 必定具有偶數的基數。所以，<Y, **1**> = 0 成立，因此向量**1** 與σ 的核空間互為正交，也就是說，必定存在一個X 使得σ(X) = **1** 為真。而此 X 就是G 上的一組ODS，也就是all-ones problem 的解。

第二節　線性代數求解的方法

假設有一個 3×3 的盤面如圖 2-1。若將每一格當做一個 vertex，而相鄰(不包括對角線相鄰)的 vertex 之間有邊相連，則這個盤面可以看成一個無向圖 G，如圖 2-2。接著將圖 G 轉化成相鄰矩陣 T，其中 vertex 之間有邊相連以 1 表示，否則以 0 表示，如圖 2-3 所示。

A	B	C
D	E	F
G	H	I

圖 2-1　3×3 的盤面，從左上開始依序以 A 到 I 的字母順序編號

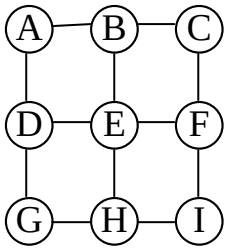


圖 2-2　3×3 盤面轉成圖 G

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
A	1	1	0	1	0	0	0	0	0
B	1	1	1	0	1	0	0	0	0
C	0	1	1	0	0	1	0	0	0
D	1	0	0	1	1	0	1	0	0
E	0	1	0	1	1	1	0	1	0
F	0	0	1	0	1	1	0	0	1
G	0	0	0	1	0	0	1	1	0
H	0	0	0	0	1	0	1	1	1
I	0	0	0	0	0	1	0	1	1

圖 2-3　圖 G 轉化成相鄰矩陣 T

令 **1** = (1, 1, …, 1)這個向量代表初始盤面的狀態，其中 1 表示燈亮的狀態，藉由解 T**x** = **1** 這個線性系統我們可以得到解 **x**，而向量 **x** 代表的就是 all-ones problem 的解，其中 1 代表要點燈，0 代表不要點燈。要注意的是這裡的運算都是在 Z_2 的集合上做運算，也就是說做矩陣運算時一定要取 2 的餘數，因此這個線性系統裡的數不是 0 就是 1。

這個方法巧妙的將 all-ones problem 化成矩陣運算的問題[6]，使我們可以很方便的利用電腦來做有效率的運算。由於我們需要儲存相鄰矩陣 T 的空間，因此這個演算法的空間複雜度估計為 $O(N^2 \times M^2)$。若是用高斯消去法解 T**x** = **1** 這個線

性系統的話，則時間複雜度估計需要 $O(N^3 \times M^3)$的時間。

第三節　搜尋演算法

本校蔡明原學長與林順喜教授的論文[2]中，提出了一種搜尋演算法能夠有效率的找出 all-ones problem 在較小的 N×M 盤面上的所有解法，以下將大略介紹此搜尋演算法。

以 5×5 大小的盤面為例，首先產生第一列的第一種點燈法，如圖 2-4，其中 1 代表點燈，0 代表不點燈，接著產生下一列的點燈法。而從第二列開始需一格一格檢查上一列的點燈法是否符合其上、下、左、右的鄰居包含自己共有奇數個燈被點(即有奇數個 1)，若全部符合，則繼續往下產生下一列的點燈法，並重複上述檢查的動作。否則只要有一格不符合就要重新產生該列另一種點燈法，如圖 2-5。若無法搜尋到合法的點燈法，就退回去重新產生上一列的另一種點燈法。也就是說，在產生第 k 列的點燈法時檢查 k-1 列的元素，而當程式產生到最後一列時還要再檢查最後一列是不是也符合上述規則，如果是，此盤面即為一組點燈解法。否則若無法找到一組最後一列的點燈法使這個盤面有解，如圖 2-6，則必須退回到第一列去重新產生另一種點燈法，直到搜尋到合法的盤面才輸出，如圖 2-7。

	A	B	C	D	E
1	0	0	0	0	0
2					
3					
4					
5					

圖 2-4　產生第 1 列第一種點燈法

	A	B	C	D	E	
1	0	0	0	0	0	
2	**0**	**0**	**0**	**0**	**0**	×
3						
4						
5						

圖 2-5　重新產生第 2 列

	A	B	C	D	E	
1	0	0	0	0	0	
2	1	1	1	1	1	
3	1	0	0	0	1	
4	1	1	0	1	1	
5	**0**	**0**	**0**	**0**	**1**	×

圖 2-6　退回去重新產生第 1 列

	A	B	C	D	E
1	1	1	0	0	0
2	1	1	0	1	1
3	0	0	1	1	1
4	0	1	1	1	0
5	0	1	1	0	1

圖 2-7　找到一組點燈解法

實驗結果發現，12×12 以下的盤面幾乎都能在一秒內找出一組解答。但是處理更大的盤面時，解題時間卻要數十秒甚至數百秒、數千秒。因此這種搜尋演算法並不適合 all-ones problem 大盤面的求解。

這個演算法是一列一列的產生點燈法，假設有一 N×M 的盤面，則第一列有 2^N 種點燈法，共要往下產生 M 列。而每一列又要一格一格檢查是否符合奇數個點燈數的規則，共要檢查 N 格。因此若要用這個演算法找出 N×M 大小盤面的“所有”解答估計需要 $O(2^N \times N \times M)$的時間複雜度，其所需的空間複雜度為 $O(N \times M)$。然而若只要找出“一組”解答，則估計平均需要 $O(2^{N-1} \times N \times M)$的時間複雜度。因此當盤面越大時，所需運算時間會呈現指數性的暴漲。

第四節　有限狀態機

陳昱璇學姐論文[3]利用有限狀態機來分析這個 all-ones problem。令每一列

點燈的狀態(1 代表點燈，0 代表不點燈)為有限狀態機中的一個 state，而令 σ^{+}-rule 作為 transition rule，所以 N×M 大小的盤面將會有 2^{N} 個 states。

假設第一列的上面有一元素全為 0 的一列，並且令這一列(稱為第 0 列)為 initial state。首先讀取第一列的元素作為有限狀態機的 input，而 output 代表的就是下一列點燈法。假設第一列的狀態為 $a_1\ a_2\ a_3 = 101$，則第二列的 $b_1 = (0+a_1+a_2+1) \bmod 2 = (0+1+0+1) \bmod 2 = 0$，$b_2 = (0+a_1+a_2+a_3+1) \bmod 2 = (0+1+0+1+1) \bmod 2 = 1$，$b_3 = (0+a_2+a_3+1) \bmod 2 = (0+0+1+1) \bmod 2 = 0$，所以 output 就是 $b_1\ b_2\ b_3 = 010$。接著讀取下一列(第二列)作為 input，而令前一列(第一列)作為 state，重複上述動作直到讀取到最後一列，如圖 2-8。若在最後一列時會 output 一全為 0 的 state，則表示找到一組成功的點燈解法。

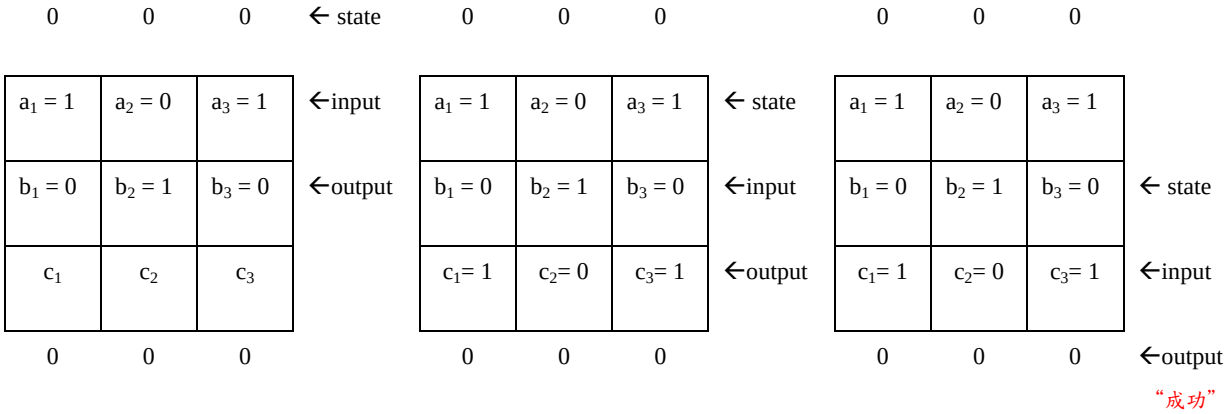


圖 2-8　有機狀態

當第一列決定後，其後續列的點法也就固定了，故本論文為了壓縮資料，對於盤面解只需要儲存第一列即可。

第五節　縮小的線性系統

論文[2]提到任意 N×M 盤面的 all-ones problem，其解法都是取決於第一列的點燈法，論文[3]利用這種特性發展出一套改良的解法。令第一列的元素依序為 a_1…… a_N，令第二列的元素依序為 b_1 …… b_N，依此類推到第 M+1 列。

假設有一 3×3 大小的盤面如圖 2-9：

0	0	0
a_1	a_2	a_3
b_1	b_2	b_3
c_1	c_2	c_3
d_1	d_2	d_3

圖 2-9　3×3 的盤面

$b_1 = (0+a_1+a_2+1) \bmod 2$，$b_2 = (0+a_1+a_2+a_3+1) \bmod 2$，$b_3 = (0+a_2+a_3+1) \bmod 2$，如圖 2-10：

0	0	0
a_1	a_2	a_3
$(a_1+a_2+1) \bmod 2$	$(a_1+a_2+a_3+1) \bmod 2$	$(a_2+a_3+1) \bmod 2$
c_1	c_2	c_3
d_1	d_2	d_3

圖 2-10　第二列與第一列之關係

$c_1 = (a_1+b_1+b_2+1) \bmod 2 = (a_1+a_3+1) \bmod 2$ ，$c_2 = (a_2+b_1+b_2+b_3+1) \bmod 2 = (0) \bmod 2 = 0$，$c_3 = (a_3+b_2+b_3+1) \bmod 2 = (a_1+a_3+1) \bmod 2$，如圖 2-11：

0	0	0
a_1	a_2	a_3
$(a_1+a_2+1) \bmod 2$	$(a_1+a_2+a_3+1) \bmod 2$	$(a_2+a_3+1) \bmod 2$
$(a_1+a_3+1) \bmod 2$	0	$(a_1+a_3+1) \bmod 2$
d_1	d_2	d_3

圖 2-11　第三列與第一列之關係

注意這裡的係數都只取除以 2 的餘數，也就是值為 0 或 1，因此所有的變數乘以偶數倍都會直接當作是 0 而被省略掉。依此法則推算到第 4 列得知，$d_1 = (b_1+c_1+c_2+1) \bmod 2 = (a_2+a_3+1) \bmod 2$，$d_2 = (b_2+c_1+c_2+c_3+1) \bmod 2 = (a_1+a_2+a_3) \bmod 2$，$d_3 = (b_3+c_2+c_3+1) \bmod 2 = (a_1+a_2+1) \bmod 2$，如圖 2-12：

0	0	0
a_1	a_2	a_3
$(a_1+a_2+1) \bmod 2$	$(a_1+a_2+a_3+1) \bmod 2$	$(a_2+a_3+1) \bmod 2$
$(a_1+a_3+1) \bmod 2$	0	$(a_1+a_3+1) \bmod 2$
$(a_2+a_3+1) \bmod 2$	$(a_1+a_2+a_3) \bmod 2$	$(a_1+a_2+1) \bmod 2$

圖 2-12　3×3 盤面第一列點燈法對於整個盤面的影響

以 3×3 的盤面為例，根據有限狀態機的分析發現，一個成功的點燈盤面必須要回到全為 0 的 state，因此可知第四列全部的元素都必須等於 0。於是我們得到了一組由第一列的變數組成的聯立方程式：

$$\begin{cases} (a_2 + a_3 + 1) \bmod 2 = 0 \\ (a_1 + a_2 + a_3) \bmod 2 = 0 \\ (a_1 + a_2 + 1) \bmod 2 = 0 \end{cases}$$

以 XOR 運算方式(以⊕表示)得到聯立方程式如下：

$$\begin{cases} a_2 \oplus a_3 \oplus 1 = 0 \\ a_1 \oplus a_2 \oplus a_3 = 0 \\ a_1 \oplus a_2 \oplus 1 = 0 \end{cases}$$

則其中一解為：

$a_1 = 1$

$a_2 = 0$

$a_3 = 1$

由此得知 3×3 盤面第一列的點燈法是 101，接著把 a_1、a_2、a_3 代入圖 2-12 中每一格所得到的算式中，就可以算出整個盤面的解答如圖 2-13：

1	0	1
0	1	0
1	0	1

圖 2-13　3x3 盤面解

此外，以 3×2 的盤面為例，用上述的方法求得聯立方程式如下：

$$\begin{cases} c_1 = a_1 \oplus a_3 \oplus 1 = 0 \\ c_2 = 0 \\ c_3 = a_1 \oplus a_3 \oplus 1 = 0 \end{cases}$$

解此聯立方程式得到

$a_1 = a_3 \oplus 1$

$a_2 = 0$ 或 1

$a_3 = 0$ 或 1

其中 a_2、a_3 是自由變數，因此第一行會有 100、001、110、011 四種情況。由於第一行的一種點燈法會唯一決定一組解，所以 3×2 的盤面會有如圖 2-14 的四種解答：

1	0	0
0	0	1

0	0	1
1	0	0

1	1	0
1	1	0

0	1	1
0	1	1

圖 2-14　4 種 3x2 盤面解

第六節　拼圖法

拼圖法，顧名思義就是由小盤面拼成大盤面的方法。本校陳昶安同學[1]提出完整的組合型態(complete state)作為拼圖法所需的小盤面，完整的組合型態其本身就是一組盤面的解，如圖 2-15，從第 1~5 列的 B~F 欄都是點亮的狀態，如此我們就可以說此 5×5 的內部矩形是一個 complete state。

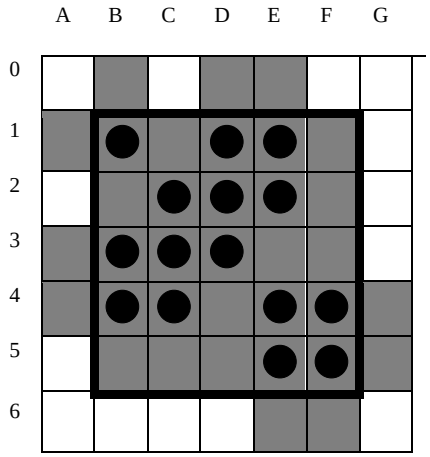


圖 2-15　5×5 的內部矩形是一個 complete state

只要決定第一列的點燈法就可以依照有限狀態機的處理模式，一直往下點燈，直到找出一個 N×M 的完整盤面(complete state)，我們稱為「基本盤面」(basic state)。也就是說，第 M+1 列不須再點燈就可使所有 N×M 盞燈均亮起來。

這些盤面總共有 2^N 個，假如 N = 3，則有 2^3 = 8 個完整盤面。而這 2^N 個盤面我們再將其兩兩可做結合的歸成一類，每一類可由一個或一個以上的基本盤面組成循環，然後再去計算每一個分類的循環其長度 j 是多少，而這些不同長度的

j 值經過數學歸納後將可證明對於此 N 值任意一個 M 值的盤面都有解，同時也決定了當欄數為 N 時，列數為 M 的盤面的第一列點燈法。

本校陳昶安同學[1]證明上述方法的可行性，首先假設一個盤面的第一列點燈法為 Row1，而 Row1 的上面有一列 initial Row 稱之為 Row0，其中 Row0 為空白的一列、Row1 為任意點燈法。如圖 2-16，利用有限狀態機的原理可以經由 Row0 和 Row1 的排列方式推導出 Row2 的點燈法，依次一直延伸出 Row3、Row4…，直到碰到空白列就停止，這樣就可得到一個完整盤面。我們利用上述方法程式一定會停止，這是因為每個 Row 的點燈法只有 2^N 種，有限種的點燈法要點出無限列勢必會出現重複的點燈法。

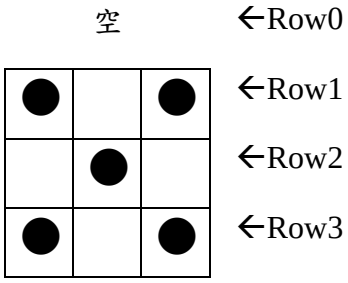


圖 2-16　推導過程

利用反證法，如圖 2-17 所示，假設 Row2 和 Row3 這樣連續兩列的點燈法會重複出現在下方的兩列 Row12 和 Row13 中，則 Row11 的上面必為 Row10 = Row0，如此一來，在重複出現在下方之前必定會先出現如 Row10 的一個空白列，然而我們產生盤面的程式碰到空白列 Row10 時就停止而不再往下產生 Row11、Row12、Row13 等列。

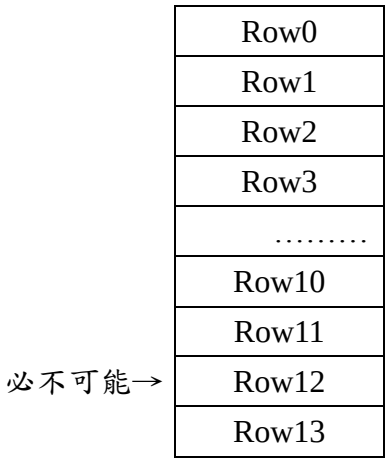


圖 2-17　若 Row0≠Row10，則不可能產生 Row12

一個經過觀察與實驗所推得的公式來解任意 N×M 的盤面($1 \le N \le 20$)如下：

令 $M = y \times k + p$

其中 y 為一常數，每個 N 值都擁有一個特定的 y 值，如表 2-1 所示，

k 為一變數，隨 M 值變化，代表循環次數，

p 為一變數，隨 M 值變化，代表組合種類。

N	y	N	y	N	y	N	y
1	3	6	9	11	48	16	255
2	4	7	12	12	63	17	168
3	6	8	28	13	18	18	513
4	5	9	30	14	340	19	60
5	24	10	31	15	24	20	2340

表 2-1　對照表

以N = 5為例，求N×M = 5×53盤面解：

Step 1.　查表2-1，N = 5時，y = 24。

Step 2.　M = 53 = 24 × 2 + 5。

Step 3.　盤面解$[D1 \rightarrow D2 \rightarrow D3 \rightarrow D4]^2 \rightarrow D1$ ，如圖2-18所示。注意其中D1的的列數=5，剛好等於p值，也就是等於M mod N。

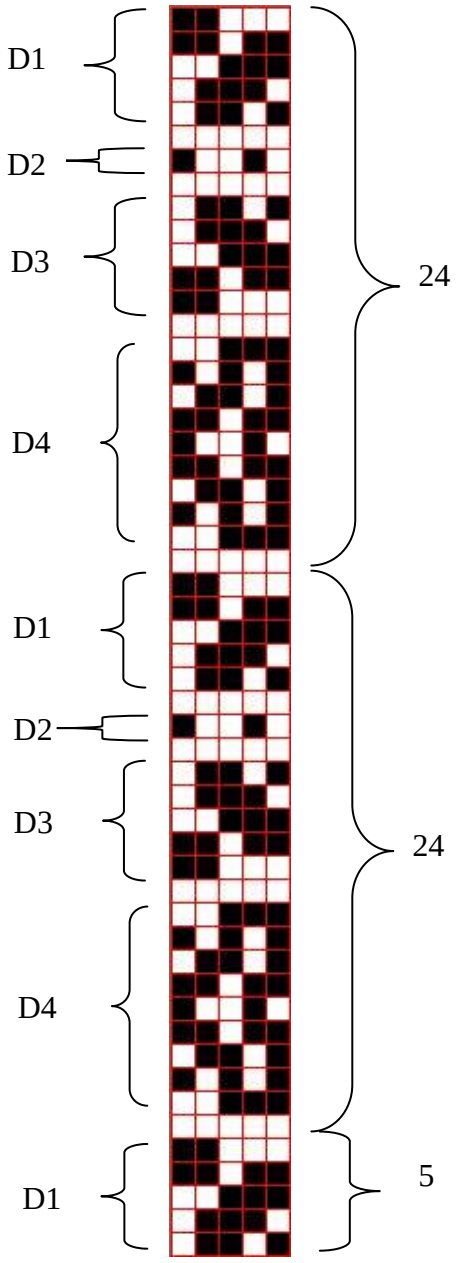


圖 2-18　5×53 盤面解

另舉一例，求N×M = 5×28盤面解：

Step 1.　查表2-1，N = 5時，y = 24。

Step 2.　M = 28 = 24 × 1 + 4，此時p值= 4。

Step 3.　盤面解[P1→P2→P3→P4]^{1}→P1　，如圖 2-18 所示。注意其中 P1 的列數 = M mod N = 24 mod 5 = 4。

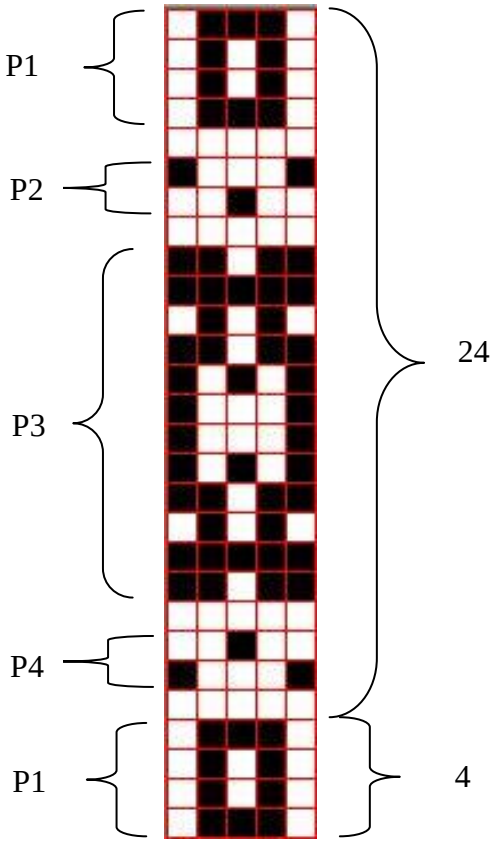


圖 2-19　5×28 盤面解

第三章 改進的演算法

第一節 進一步的加速方法

本節首先在討論避免拼圖法窮舉第一列的情況發生，且說明經過觀察有些基本盤面的第一列或者最後一列存在鏡射的關係、XOR 之規則。

刪除盤面重複：在資料結構方面，陣列都是使用一維陣列為主，二維陣列不使用以減少程式在執行時，二維會轉換成一維之時間浪費。N×M 的盤面中每種循環都由數種基本盤面所組成，且盤面與盤面之組合都是呈現互補關係。當找尋完一個基本盤面，假設稱之為盤面甲，下一個會與之組合的基本盤面稱之盤面乙，盤面乙的第一列就是盤面甲最後一列之互補，依據此特徵我們可以持續往下延展直至下一個盤面已經重複出現，當盤面重複出現以後即表示，之前的盤面組合即可形成一種循環，就可以把這些出現過的基本盤面之第一列與最後一列、還有其左右翻轉後所呈現的數字，例如說 11100=28 把 11100 翻轉即呈現 00111=7 這種情況都給排除掉，這樣下次在找尋新的盤面組合即不會有重覆之發生，因而可加快處理速度。這一節所提到的方法雖然可以加速，但後來並未使用，因為最後論文所使用的方法其效率較好。

至於循環長度的取得，只需要初始列以不點燈的情況下去找尋循環組合即可，如 N =5 時，圖 3-1~圖 3-3 就是以第一列就是 00000 之狀態往下點燈所找尋到的基本盤面循環組合，循環長度 y =（8+1+6+1+7+1）= 24，其中圖 3-1 若把第一列

給拿掉也滿足 all-ones problem 之解。循環 y 所需的盤面組合，必須滿足 y 值以內的各種餘數組合，經過觀察當第一列點燈法全為 0 再經過延展，其循環發生的時候，循環長度等於 N 值所對應的 y 值，例如 Row1 全為 0，則第(y+1)列= Row1 且第(y+2)列= Row2，當小於 y 的各種餘數之盤面循環組合都找到時，即可終止搜尋。

盤面組合是以互補的情況來進行之，所以 N = 5 時第一列的點燈情況並不需要從 00000~11111 全部窮舉而是只需要 00000~01111 窮舉一半即可，因為每個 Row1 一直延展下去最後都會形成循環，以圖 3-1~圖 3-3 來說明，圖 3-1 與 3-3 是互補方式組合，圖 3-1 的 Row1 與圖 3-3 的最後一列 Row7 互補，假設 N=5 時都是三個盤面組合形成循環，第一個盤面 Row1 從 00000~01110 都窮舉完，那第三個盤面的最後一列也就會跟第一個盤面的 Row1 互補，其範圍是 11111~10000，故可以減少一半的窮舉時間。

	A	B	C	D	E
1					
2	●	●	●	●	●
3	●				●
4	●	●		●	●
5		●	●	●	
6	●				●
7		●		●	
8	●		●		●

圖 3-1　5×8 盤面解

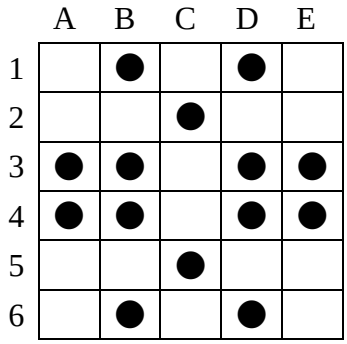


圖 3-2　5×6 盤面解

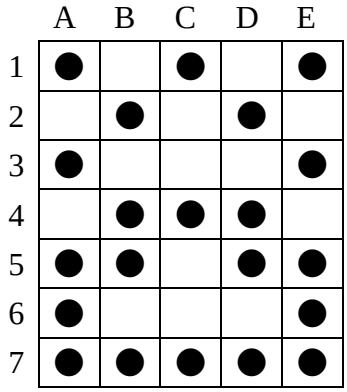


圖 3-3　5x7 盤面解

針對第一列之組合進一步刪減：循環盤面的組合中有些盤面之第一列或者最後一列呈現左右鏡射之關係，如圖 3-4，A1 欄位的狀態與 F1 相同，B1 與 E1 相同，C1 與 D1 相同，利用這樣左右對稱的特徵去跑第一列，就只需窮舉 $2^3 = 8$ 種組合，而不需窮舉 $2^6 = 64$ 種組合若能找出此第一列所形成的盤面解與其可與之形成循環組合的其他盤面，則就更有效率了。

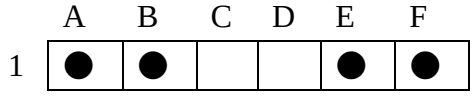


圖 3-4　鏡射圖

經過幾次實驗在 N×M 的盤面(20≤N≤38)之測試結果，N 屬於奇數的情況下無法利用左右鏡射的方式找尋到拼圖法所需的盤面組合，但 N 屬於偶數的 N×M 盤面(20≤N≤38 且 N 屬於偶數，除了 20、26、32、38 以外)，都可以利用這種左右鏡射之第一列的方式找尋到拼圖法所需的盤面組合，與先前初始列之窮舉法如沒做任何刪減，初始列有2^N種；經過這種方式則只需要搜尋$2^{\frac{N}{2}}$種即可。

至於 N×M 的盤面中 N = 20、26、32、38 時，雖然無法只使用鏡射關係就可以全部找尋拼圖法所需的盤面組合，但其所無法找尋的盤面組合其長度經過 mod y 之後卻是呈現 1+4X 之值（1+4X＜y-1，y 即是 N 所需的循環長度）。但經過對 20×1、26×1、32×1 盤面之解作 XOR 處理之後卻發現到 XOR 的結果呈現出鏡射關係，且由左至右呈現 110 110 110…之循環，說明如下。

在 20×1 盤面：

原來第一列之解：　　10010010010010010010

經左右翻轉後之值：⊕）　01001001001001001001

XOR→　　11011011011011011011 (左右鏡射且為 110 110 110…之循環)

在 26×1 盤面：

原來第一列之解：　　10010010010010010010010010

經左右翻轉後之值：⊕）　　01001001001001001001001001

XOR→　　11011011011011011011011011 (左右鏡射且為

110 110 110…之循環)

在 32×1 盤面：

原來第一列之解：　　10010010010010010010010010010010

經左右翻轉後之值：⊕）　　01001001001001001001001001001001

XOR→　　11011011011011011011011011011011 (左右鏡射且為

110 110 110…之循環)

所以在搜尋 1+4X 盤面組合的時候，例如 N = 20，如圖 3-5 我們只需要跑其中第一列中 10 個欄位，其範圍值 2^0~2^{10}-1，例如只窮舉 11~20 的欄位而 1~10 的欄位則是由 XOR 轉換而來，然後所得到的 Row1 就去延展，以此方法就可以得到 1+4X 盤面的組合。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

圖 3-5　N = 20 之欄位編號

「原本的 11 到 20 欄位」的數值與「還原後的 10 到 1 欄位的」數值互相 XOR 後必須滿足「N = 20 其 XOR 之結果」。舉例來說，若要產生 80×81 盤面解之第一列轉換方式如下：

原本的 11 到 20 欄位：　　1000011001

未知的 10 到 1 欄位：⊕）　　??????????

N = 20 其 XOR 之結果→　　1011011011

原本的 11 到 20 欄位：　　1000011001

還原後的 10 到 1 欄位：⊕）　　0011000010

N = 20 其 XOR 之結果→　　1011011011

驗證所得到的 1 到 20 欄位的解是否符合 N=20 之 XOR 規則：

1 到 20 欄位：　　01000011001000011001

20 到 1 欄位：⊕）　　10011000010011000010

XOR→　　11011001011011011011

下面就是我們需要的第一列，然後在繼續點燈下去找尋其盤面組合																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	1

圖 3-6　此圖即為 3-7 第一列點燈法

圖 3-7，即為圖 3-6 的第一列之盤面解，其盤面大小為 20×81。

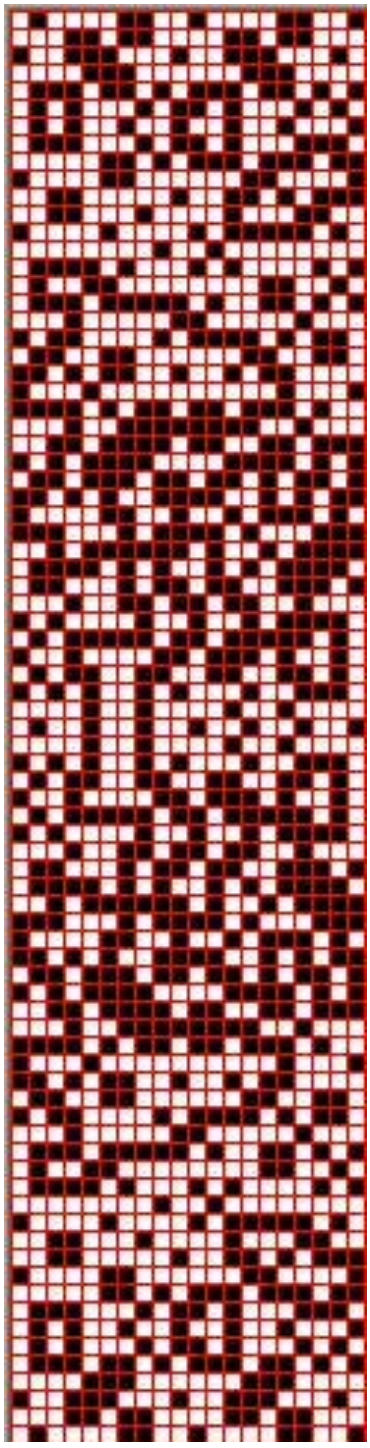


圖 3-7　20×81 盤面解

盤面 20×M、26×M、32×M、38×M 之第一列的種類經過鏡射關係與 XOR 之篩選以後，不但搜尋速度加快且能找尋到拼圖法所需的盤面組合，至於 XOR 的

第一列之種類有 $2^{\frac{N}{2}}$ 再加上鏡射關係也是需要 $2^{\frac{N}{2}}$ 種，所以總共需要跑 $2\times 2^{\frac{N}{2}}$ 種第一列之組合還是比窮舉法要跑 2^{N} 種第一列之組合快許多。

第二節 縮小的線性系統與拼圖法之優缺點

本研究結合陳昱璇學姐論文[3] 縮小的線性系統之演算法與本校陳昶安同學[1]拼圖法的理論來提升拼圖法之搜尋速度。

縮小的線性系統[3]之演算法特性在於解 N×M 盤面，例如說 21×30 盤面，其聯立方程式的產生會是從 M = 1 開始產生一直到 M = 30，也就是說中間 1 ≤ M ≤ 29 所產生的聯立方程式是不需要求出其解，而只需求出最後 M = 30 的聯立方程式的解。當 M 很大時，也必須耗費大量時間產生 N×M 盤面所形成的聯立方程式，並求出其解。

拼圖法[1]的特性在於可以小盤面與小盤面互相結合而成一個大盤面，且盤面只要事先收集即可，例如圖 3-8，D1 為 5×5 盤面、D2 為 5×1 盤面，D1 與 D2 可拼成 5×7 的盤面。收集小盤面的過程若是已經找到拼圖法所需的組合，則收集過程就會暫停，而最壞的情況若有 2^{N} 種 Row1 點燈法則要跑完 2^{N} 種盤面才能收集所需的盤面，另一個情況是在收集的過程中可能會有同樣大小之盤面或者盤面組合卻重複產生，造成時間耗費。

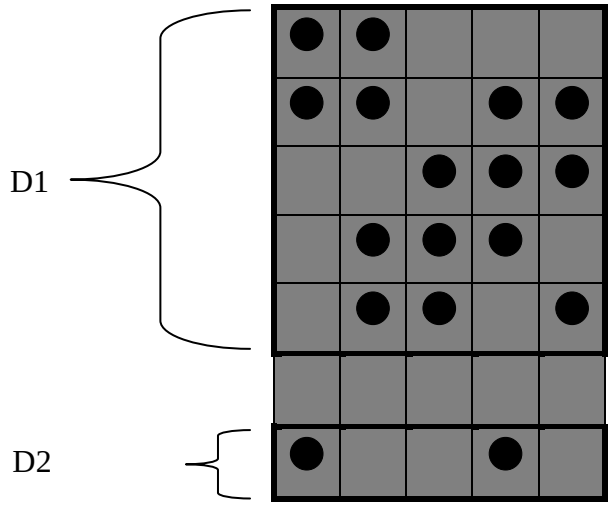


圖 3-8　5×7 盤面解

本研究在收集拼圖法所需小盤面之第一列是用縮小的線性系統來求出其解，解 N×M 盤面時是先固定 N 值然後列數 k 從 1 開始收集，例如說 N = 21，則盤面收集從 21×1、21×2、21×3…，換言之，縮小的線性系統所產生的每一組聯立方程式都會被運算求出其解，且每一種大小盤面只需要儲存一組 Row1 之解即可，最終目的我們希望能找到一種 N×P 這樣大小的盤面，它的特性是任意輸入一組 Row1 後經過有限狀態機的處理都能使得由此所產生的 N×P 盤面均有解。若找到這種 N×P 盤面則表示已經收集足夠的小盤面，也就是小盤面 N×k 且 $1 \leq k \leq P$ 都必須收集到。每個 N 值都有對應的 y 值且 P = y - 1，如表 3-1，例如 N = 5 時對應的 y = 24，N×P 盤面大小為 5×(y - 1) = 5×(24 - 1) = 5×23。

N	y	N	y	N	y	N	y
1	3	17	168	33	510	49	6150
2	4	18	513	34	4095	50	262140
3	6	19	60	35	336	51	252
4	5	20	2340	36	3591	52	67108865
5	24	21	186	37	1026	53	12264
6	9	22	2047	38	16380	54	25575
7	12	23	96	39	120	55	72
8	28	24	1025	40	1048575	56	1048572
9	30	25	126	41	4680	57	19662
10	31	26	2044	42	16383	58	536870911
11	48	27	36	43	372	59	4080
12	63	28	3277	44	92820	60	1073741823
13	18	29	2040	45	12282	61	2046
14	340	30	341	46	8388607	62	16380
15	24	31	48	47	192	63	96
16	255	32	4092	48	2097153		

表 3-1　對照表

第三節 N×P 盤面之拼圖法

本節之目的在於說明如何用 N×P 盤面來拼成 N×M 之盤面解，其中 P = y-1。

定義盤面 Q 與盤面 A_k。盤面 Q：其盤面大小為 N×P 且任意輸入一組 Row1 後經過有限狀態機的處理都能使得盤面 Q 滿足 all-ones problem，如圖 3-9，例如 N = 3 時盤面 Q 大小為 3×5，Row1 總共有 $2^3 = 8$ 種組合，分別填入盤面 Q 再經過

有限狀態機的處理，盤面大小為 3×5 的情況下都能得出 all-ones problem 的一組解。

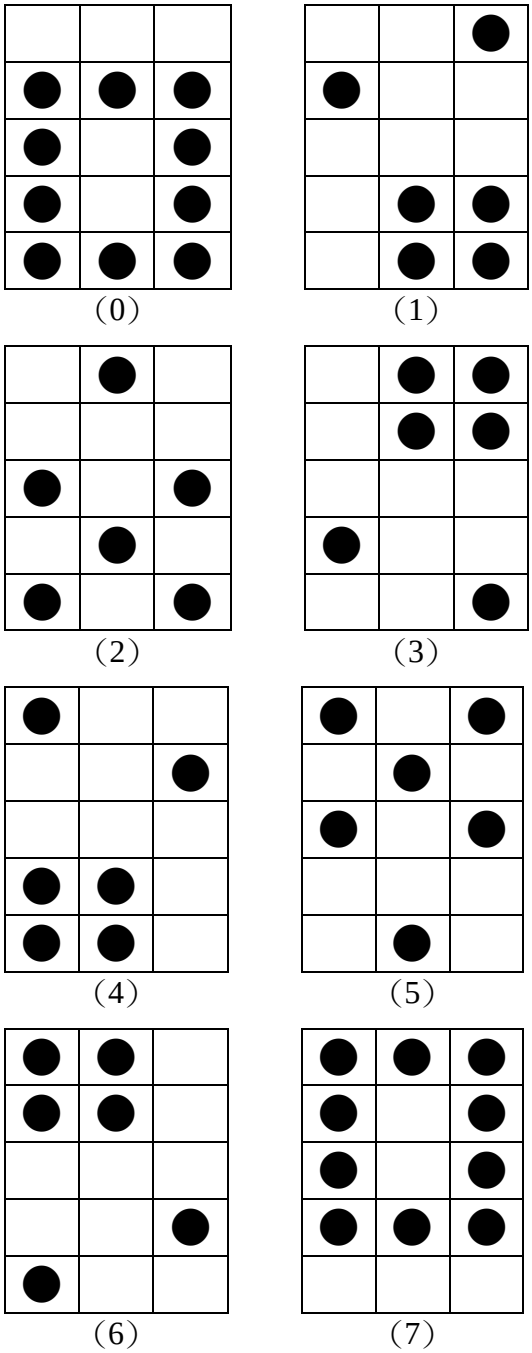


圖 3-9　8 種 Row1 在 3×5 盤面，都可找出全一問題的解

盤面 A_k：盤面大小為 N×k 且為 all-ones problem 的解，k 值為整數且範圍 1

≤ k ≤ P，例如 N = 3 時所對應的 y = 6，P = y – 1 = 6 – 1 = 5，所以 1 ≤ k ≤ 5 總共需收集 5×1、5×2、5×3、5×4、5×5 這 5 種盤面(分別稱之為 A_1、A_2、A_3、A_4、及 A_5)即可，如圖 3-10。

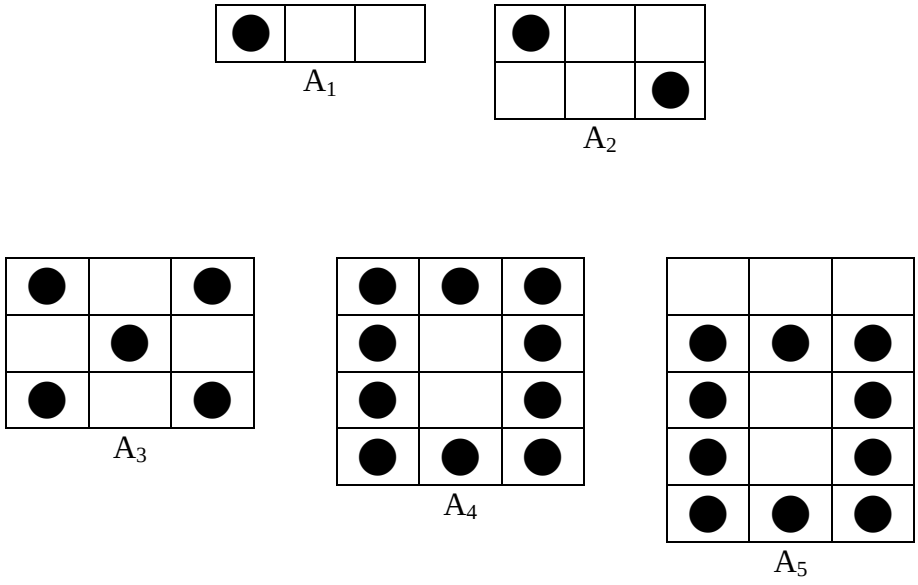


圖 3-10　收集 5 種基本盤面解

當收集完 $A_1, A_2, \ldots, A_P$ 這 P 個盤面就可以用來拼出所有的 N×M 盤面，其中 N 與 M 為整數且 N 為有限長度、M ≥ 1。令 P = y - 1，y、P 皆為整數，下列有三種情況來證明此作法之可行性：

情況一：當 N×M 盤面且 M < y 時， M < y 也就是 M ≤ P，因盤面 A_k 事先已經收集好，故 N×M 盤面會有解。

情況二：當 N×M 盤面且 M = k×y，k 為整數、k ≥1 時，盤面 Q 的 Row1 全為 1。舉例來說，若 k = 1、N = 3 且 y = 6 則盤面 Q 的 Row1 點法為 111 且盤面 Q

大小為 3×5，如圖 3-11，因 σ^{+}-rule，Row0 也會全被點亮， 因此 N×M = 3×6 的盤面也有解。

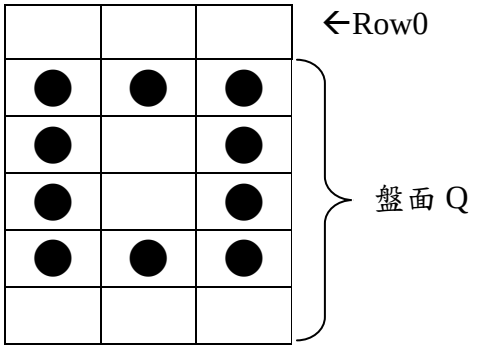


圖 3-11　3×6 盤面解

若 k > 1，例如 k = 2、N = 3 且 y = 6 則 N×M = 3×12 盤面，可由兩個盤面結合起來滿足 3×12 的盤面之解，如圖 3-12，所以當 k = 3、4、5…，盤面 Q 一直拼圖下去即可，故有解。

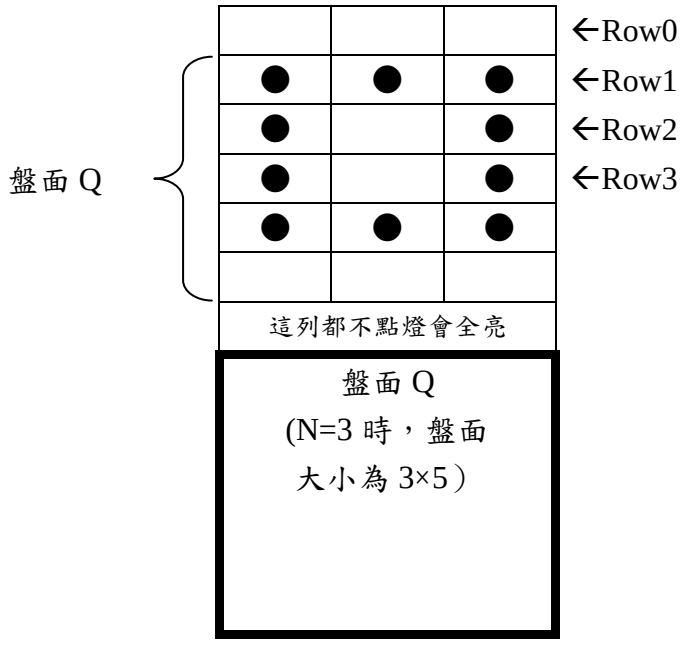


圖 3-12　3×12 盤面解

情況三：當 N×M 盤面且 M = k×y+ p，k 與 p 皆為整數、1 ≤ p ≤P 且 k ≥1 時，若 k = 1，N×M = N×(p+y) = N×p + N×y = N×p + N×(1+P) = N×p + N×1 + N×P = 盤面 A_p+ N×1 盤面 + 盤面 Q，如圖 3-13：

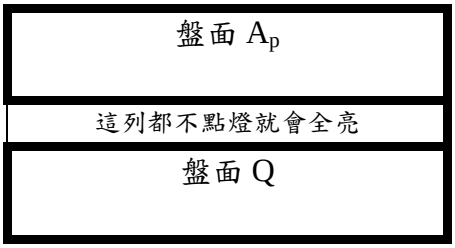


圖 3-13　k = 1，盤面解之組成

若 k = 2，N×M = N×(p+2×y) = N×p+N×(2×y) = N×p+N×(2×(1+P)) = N×p+N×(1+P+1+P) = N×p +N×1+N×P+N×1+N×P = 盤面 A_k +N×1 盤面+盤面 Q+ N×1 盤面+盤面 Q，如圖 3-14。所以當 k = 3、4、5…，盤面 Q 一直拼圖下去即可，故有解。

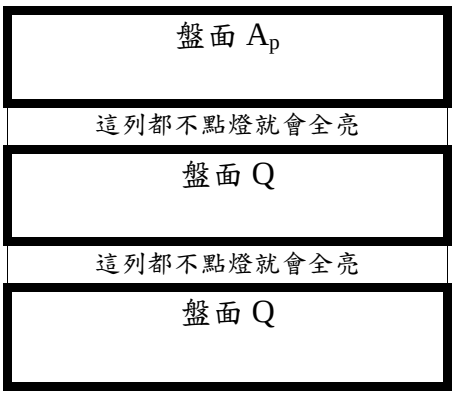


圖 3-14　k = 2，盤面解之組成

第四節 證明 NxP 盤面之存在性

本節之目的在於證明對任意的 N 值，必存在一個 P 值，使得尺寸 NxP 的盤面，任意輸入一組 Row1，都可以使得此 NxP 盤面有解。

定義盤面 R_k、L_k。盤面 R_k：Row1 其十進位數字表示為 k，k 值為整數且範圍 $0 \le k \le 2^N-1$，例如 N = 3，Row1 給予 010 則十進位數字表示為 2。

L_k：R_k 經過有限狀態機所呈現的盤面長度，其特性為 $N \times L_k$ 的盤面滿足 all-ones problem 之解，在第(L_k+1)列沒有任何燈泡必需被點亮且第(L_k+2)列的點燈法等同於第一列，第(L_k+2)列的點燈法與第 L_k 列的點燈法互補，如圖 3-15，N = 3 時 R_2 的 $L_2 = 5$。每一種 R_k 都有其對應的 L_k。

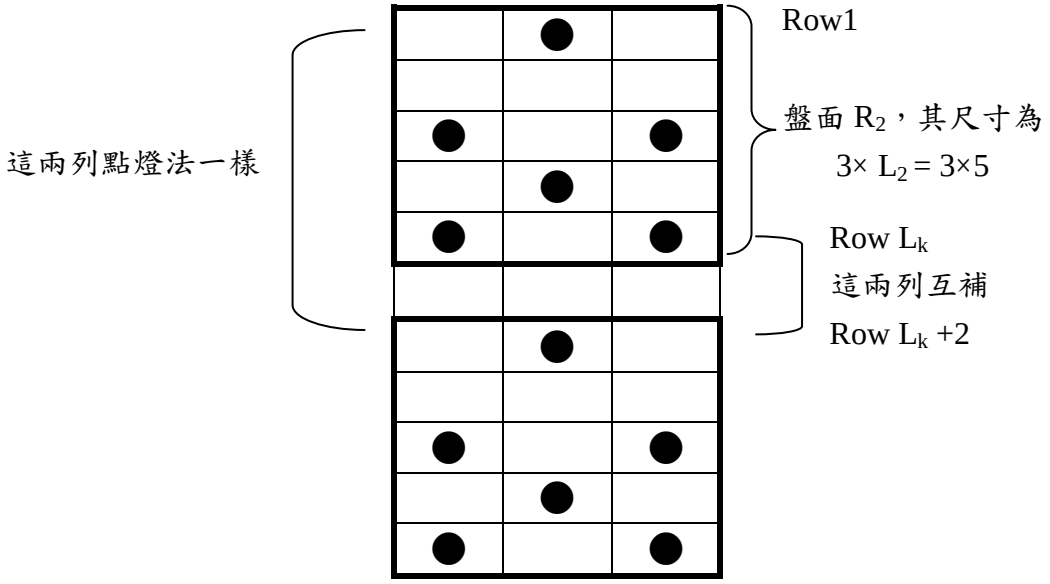


圖 3-15　N = 3，R_2 的 $L_2 = 5$

在第二章第六節有證明提到，有限種的點燈法要點出無限行勢必會出現重複

的點燈法，如圖 3-16：

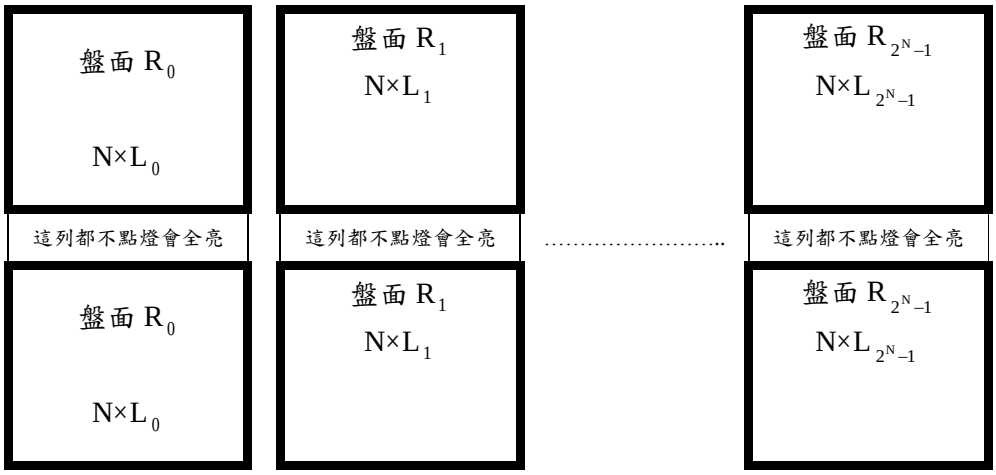


圖 3-16　循環長度之情況

令 $Y_k = 1+L_k$、$0 \le k \le 2^N-1$，如圖 3-16 所示，盤面 R_0 與盤面 R_0 上下拼在一起則尺寸為 $N\times(L_0+1+L_0) = N\times(L_0+Y_0)$的盤面的解也找出來了，盤面 R_1 與盤面 R_1 上下拼在一起則尺寸為 $N\times(L_1+1+L_1) = N\times(L_1+Y_1)$的盤面的解也找出來了，直到盤面 R_{2^N-1} 與盤面 R_{2^N-1} 上下拼在一起則尺寸為 $N\times(L_{2^N-1}+1+L_{2^N-1}) = N\times(L_{2^N-1}+Y_{2^N-1})$的盤面的解也找出來了。從上述的情形會發現到兩兩相同盤面 R_k 拼在一起，其尺寸為 $N\times(L_k+Y_k) = N\times(Y_k-1+Y_k)$。假設把 Row0 都算進去，如圖 3-17：

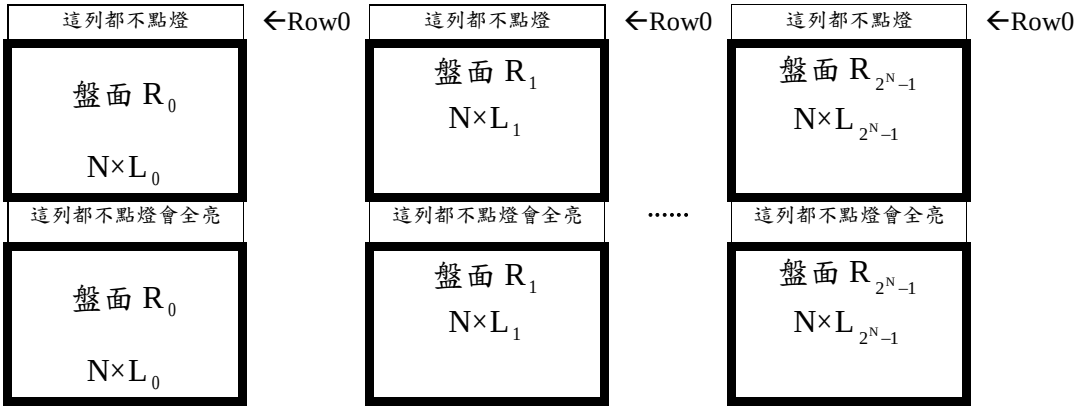


圖 3-17　將 Row0 也列為循環長度

當只用一個盤面 R_k 時，尺寸為 $N\times Y_k$ 的盤面有解。當兩個相同盤面 R_k 上下拼在一起時，尺寸為 $N\times(2Y_k)$ 的盤面亦有解。以此類推。故存在 $Y_0,Y_1,Y_2,\ldots,Y_{2^N-1}$ 的最小公倍數，令最小公倍數為 W，令 P = W - 1，若 Y_k 都互質則 $W = Y_0\times Y_1\times Y_2\times\ldots\times Y_{2^N-1}$。由上所述，我們知道 Row1 的點燈法可為任一種 k 值（$0\le k\le 2^N-1$），則尺寸為 $N\times(Y_0\times\ldots\times Y_{k-1}\times Y_{k+1}\times\ldots\times Y_{2^N-1}\times Y_k) = N\times M$ 的盤面亦會有解。因為 Row0 是我們假設多算進去的也可以扣掉，故存在一種盤面其大小為 N×P，任意輸入一組 Row1 的點燈法都可以使得此 N×P 盤面有解。

第五節 程式環境與流程圖

本節之目的在於說明程式開發的環境與使用的編譯軟體，以及電腦配備等相關資訊，如圖 3-18：

作業系統	電腦	程式軟體
Microsoft Windows Xp Professional Version 2002 Service Pack3	Intel(R) Core (TM)2 CPU E8500 @ 3.16GHz 3.17GHz,3.25GB 的 RAM	Dev-C++ 4.9.9.2

圖 3-18　程式環境與電腦配備

程式執行的架構如圖 3-19：

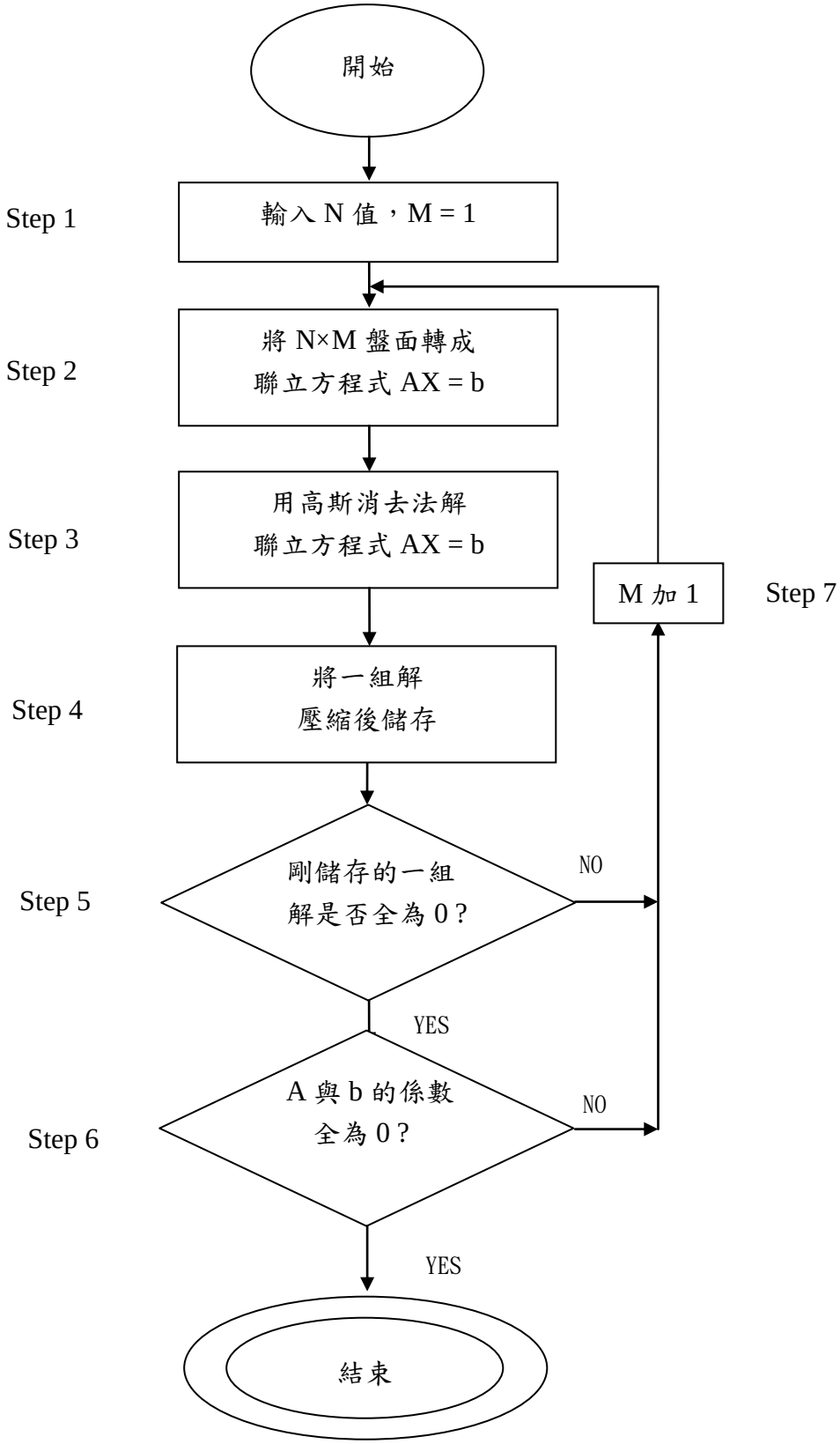


圖 3-19　流程圖

圖 3-19 流程圖的 Step 1~7 的說明如下：

Step 1：輸入 N 值且 M 值從 1 開始。

Step 2：將 N×M 盤面轉成聯立方程式。

Step 3：利用高斯消去法解出聯立方程式之解，在要一步一步回填變數值的時候，若是變數值可填 0 或者 1，則以 0 為優先。

Step 4：找出一組解即壓縮成 64bits 然後儲存。

Step 5：若 Step 4 剛儲存的解值全為 0，則跳到 Step 6，若不是則跳到 Step 7。

Step 6：檢查剛剛經過高斯消去法的聯立方程式 Ax = b 中的 A 之係數與 b 之係數是否全為 0，若是全為 0 則表示 x 值可為任意值，也就是現在的 N×M 這樣大小的盤面任意輸入一組 Row1 經過延展以後，都會滿足 all-ones problem，就可以結束程式，因為現在的 N×M 就是我們要找尋的 N×P，若 A 與 b 之係數不全為 0，則跳到 step 7。

Step 7：將 M 值累加一，然後跳到 Step2 進行下一個 N×M 盤面之求解。

第六節 處理時間

用[3]提出的縮小線性系統在找出 N×M 盤面上的一組解答，無論在時間複雜度或是空間複雜度上都減少許多。空間複雜度只需要處理 N^2 大小的係數矩陣，

時間複雜度為$(N\times M+N^3)=O(N^3)$，其中 $O(N\times M)$的時間用在列出聯立方程式，另外 $O(N^3)$是利用高斯消去法解聯立方程式之時間複雜度。本論文在收集每一個盤面解時間跟[3]所耗費的時間複雜度一樣，不同的是N×M盤面解的M值範圍為$1 \le M \le P$，而不是單一值，每一組盤面解都會依序儲存好，當整個拼圖法所需的盤面解都收集好，只需要利用一個公式即可知道第一列點燈法是屬於哪一種。

$M = y\times k + p$

其中 y 為一常數，每個 N 值都擁有一個特定的 y 值，

k 為一變數，隨 M 值變化，代表需要幾個盤面 Q 來作組合，

p 為一變數，隨 M 值變化，代表第一列點燈法是儲存在哪一個位置。

N	y	秒	N	y	秒	N	y	秒
21	186	0.015	36	3591	0.25	51	252	0.046
22	2047	0.046	37	1026	0.078	52	67108865	11110.6
23	96	<0.001	38	16380	1.156	53	12264	1.875
24	1025	0.046	39	120	<0.001	54	25575	4.703
25	126	<0.001	40	1048575	81.562	55	72	0.015
26	2044	0.063	41	4680	0.437	56	1048572	183.562
27	36	<0.001	42	16383	1.671	57	19662	3
28	3277	0.14	43	372	0.031	58	536870911	90239.3
29	2040	0.062	44	92820	9.718	59	4080	0.718
30	341	0.015	45	12282	1.109	60	1073741823	221653
31	48	<0.001	46	8388607	913.578	61	2046	0.39
32	4092	0.187	47	192	0.031	62	16380	3.453
33	510	0.015	48	2097153	253.64	63	96	0.031
34	4095	0.218	49	6150	0.781			
35	336	0.016	50	262140	37.89			

表 3-2　收集盤面所耗費的時間

表 3-2 顯示收集盤面所耗費的時間，例如說 N = 38 對應的 y = 16380 其收集盤面從 38x1、38x2、38x3、……、一直到 38x16379，總共所花費的時間為 1.156 秒。

第七節 程式介面

使用者程式介面，如圖 3-20，其目的提供使用者顯示所選擇的 N×M 盤面解，若要顯示的盤面解之圖的總 pixels 數≤1024×768，則會顯示整個盤面解的點燈法情況，如圖 3-21，若要顯示的盤面解之圖的總 pixels 數＞1024×768，則只會顯示所選擇的盤面解之第一列點燈法。圖 3-20 中 A 是讓使用者選擇 N 值，其範圍 $1 \leq N \leq 63$。B 是讓使用者選擇 M 值，其範圍 $M \geq 1$。C 是讓使用者選擇棋盤格子之 size 為多少個 pixels，例如說初始是給予 4，表示格子大小為 4pixels×4pixels。當使用者選擇好 N 值與 M 值，程式會讀取 N 值所對應的 y 值，且 address = M mod y，就到儲存 N 值所對應的盤面解之壓縮資料檔中，根據 address 來讀取盤面解之第一列的點燈法，N 值所對應的盤面解之壓縮資料是事先收集好的，收集過程就如圖 3-19 所示。

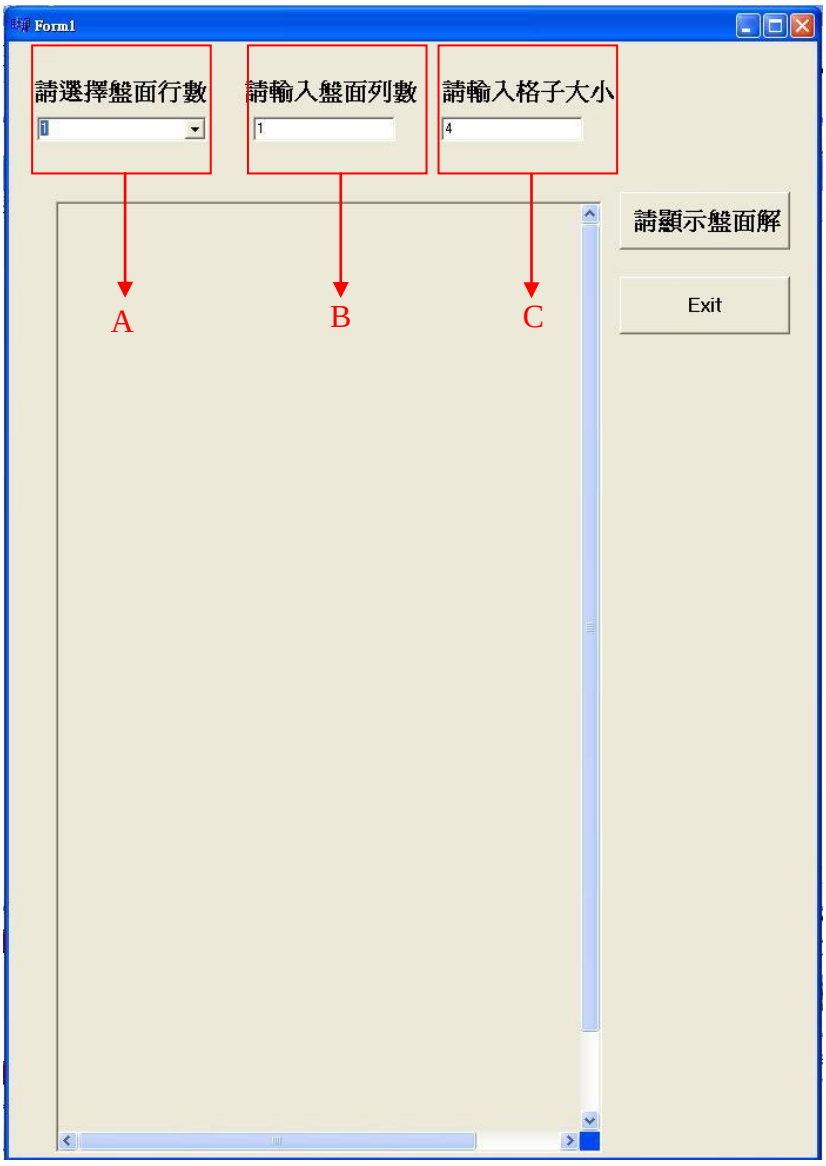


圖 3-20　程式介面

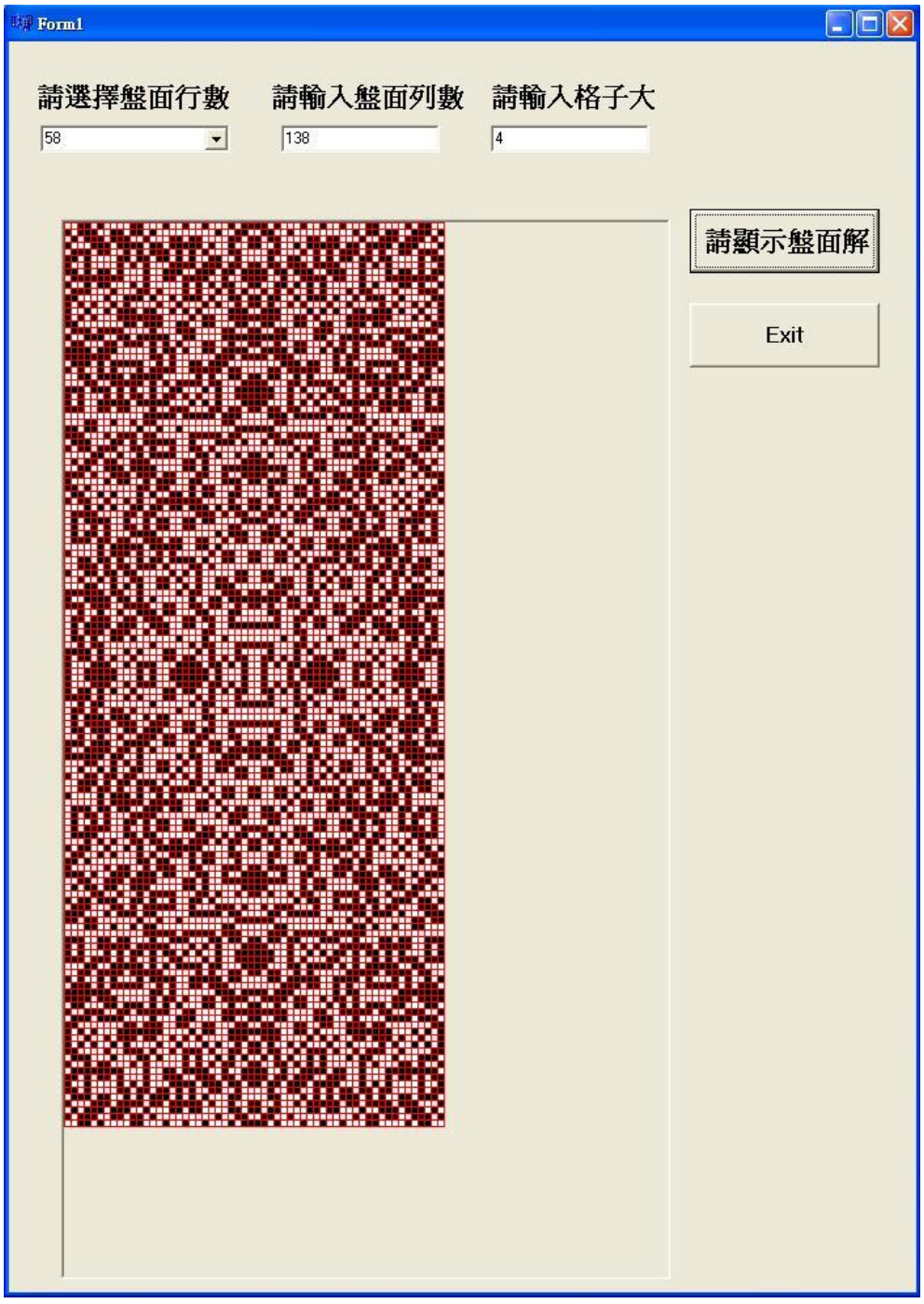


圖 3-21　顯示完整的盤面解

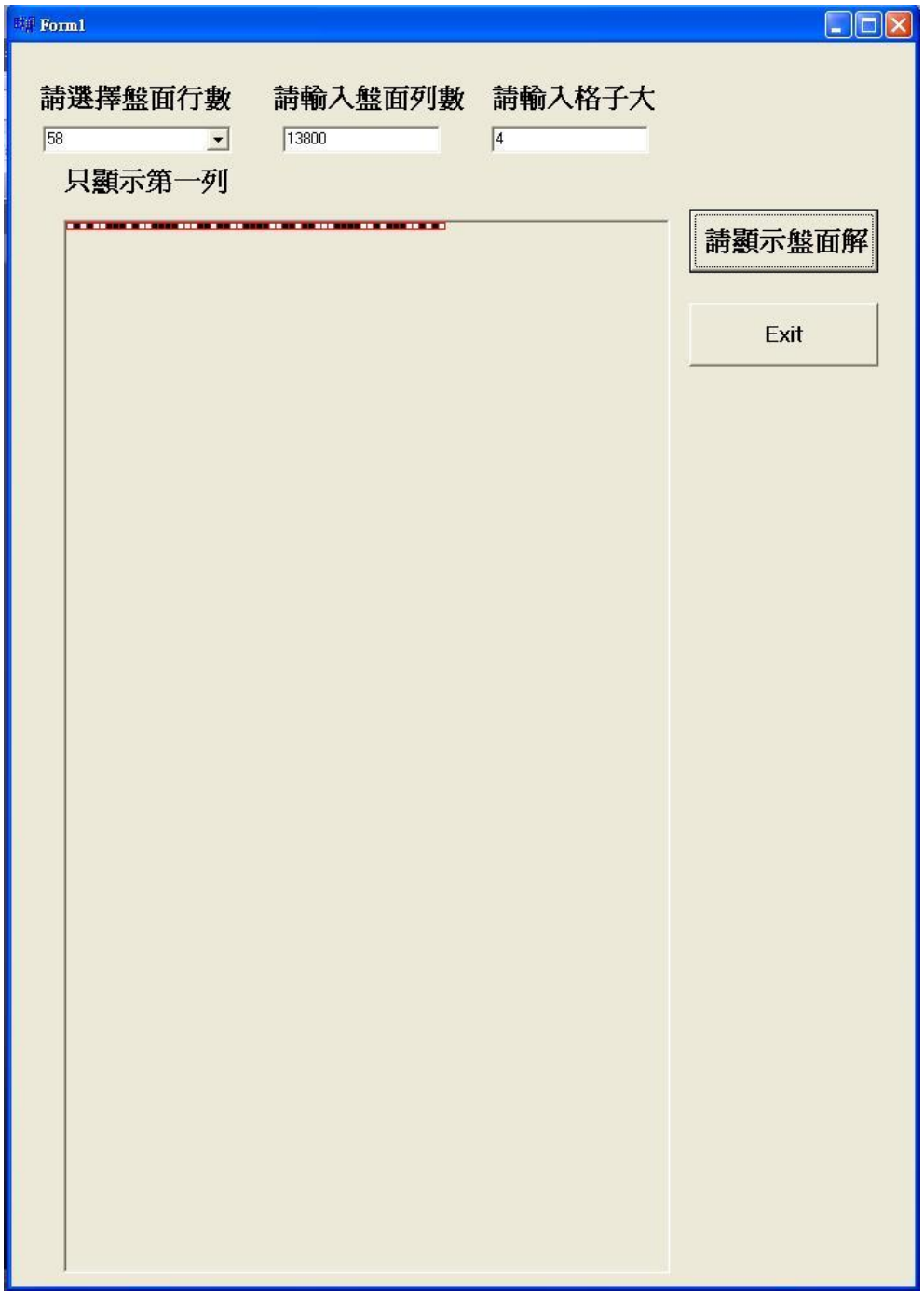


圖 3-22　顯示盤面解之第一列點燈法

第四章 結論

第一節 結論與未來研究

本研究有效率地提升搜尋拼圖法所需的盤面之速度，使得 N×M 盤面的破解數量從原本[1]的成果 N ≤ 20 且 M ≥ 1 提升到 N ≤ 63 且 M ≥ 1 都已破解，如表 4-1。盤面收集所耗費的時間，也就是比以前的方法少了很多。在研究的過程中，我們也曾經針對拼圖法所使用的盤面之第一列，利用左右對稱的特徵方式去產生第一列以避免窮舉法的情況，對於盤面的破解也是有幫助。此外，本論文盤面的破解都是屬於一維延展，對於二維延展目前還是沒有比較可行的方法實現之。依據 N×M 盤面大小所產生的聯立方程式，是否有規則可以一次就顯示出聯立方程式，而不需要一列一列地演變直到 M 列？這是未來可探討的一個方向。

第二節 研究成果

我們的成果如表 4-1 的 21≤N≤63 的部分，其中顯示破解 21≤N≤63 各自所耗費的時間，例如說 N=38 對應的 y = 16380，其收集盤面從 38x1、38x2、38x、……、一直到 38x16379 一共所花費的時間為 1.156 秒。

N	y	秒	N	y	秒	N	y	秒
1	3	<0.001	22	2047	0.046	43	372	0.031
2	4	<0.001	23	96	<0.001	44	92820	9.718
3	6	<0.001	24	1025	0.046	45	12282	1.109
4	5	<0.001	25	126	<0.001	46	8388607	913.578
5	24	<0.001	26	2044	0.063	47	192	0.031
6	9	<0.001	27	36	<0.001	48	2097153	253.64
7	12	<0.001	28	3277	0.14	49	6150	0.781
8	28	<0.001	29	2040	0.062	50	262140	37.89
9	30	<0.001	30	341	0.015	51	252	0.046
10	31	<0.001	31	48	<0.001	52	67108865	11110.6
11	48	<0.001	32	4092	0.187	53	12264	1.875
12	63	<0.001	33	510	0.015	54	25575	4.703
13	18	<0.001	34	4095	0.218	55	72	0.015
14	340	0.016	35	336	0.016	56	1048572	183.562
15	24	<0.001	36	3591	0.25	57	19662	3
16	255	0.015	37	1026	0.078	58	536870911	90239.3
17	168	0.016	38	16380	1.156	59	4080	0.718
18	513	0.015	39	120	<0.001	60	1073741823	221653
19	60	<0.001	40	1048575	81.562	61	2046	0.39
20	2340	0.047	41	4680	0.437	62	16380	3.453
21	186	0.015	42	16383	1.671	63	96	0.031

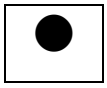
表 4-1　成果表

附錄 A 中列出本研究所求出的一些較小盤面的解。

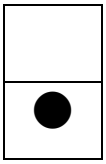
附錄 A 列出本法求出的一些小盤面的解

N = 1，y = 3：

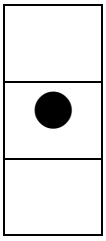
1×1



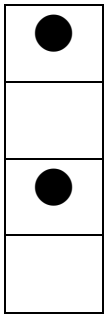
1×2



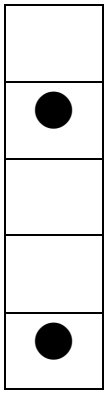
1×3



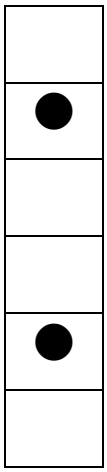
1×4



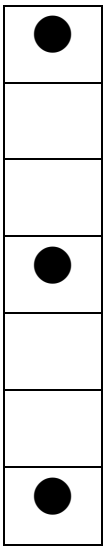
1×5



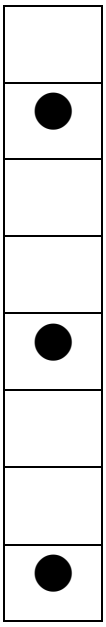
1×6



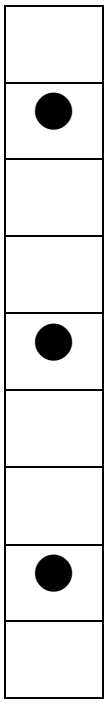
1×7



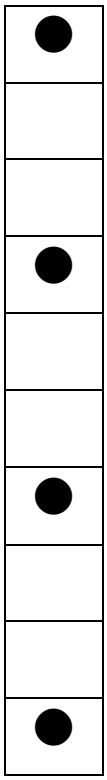
1×8



1×9

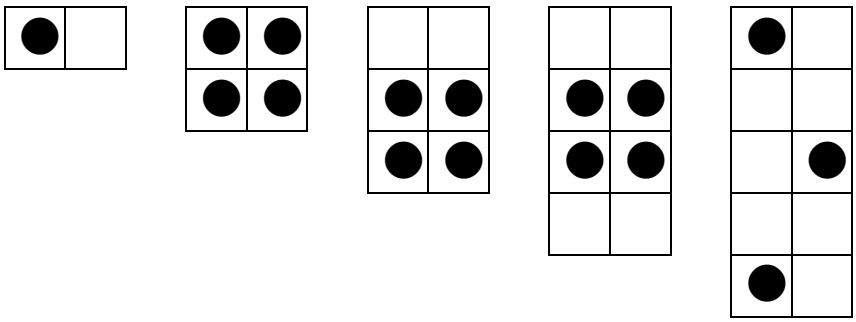


1×10

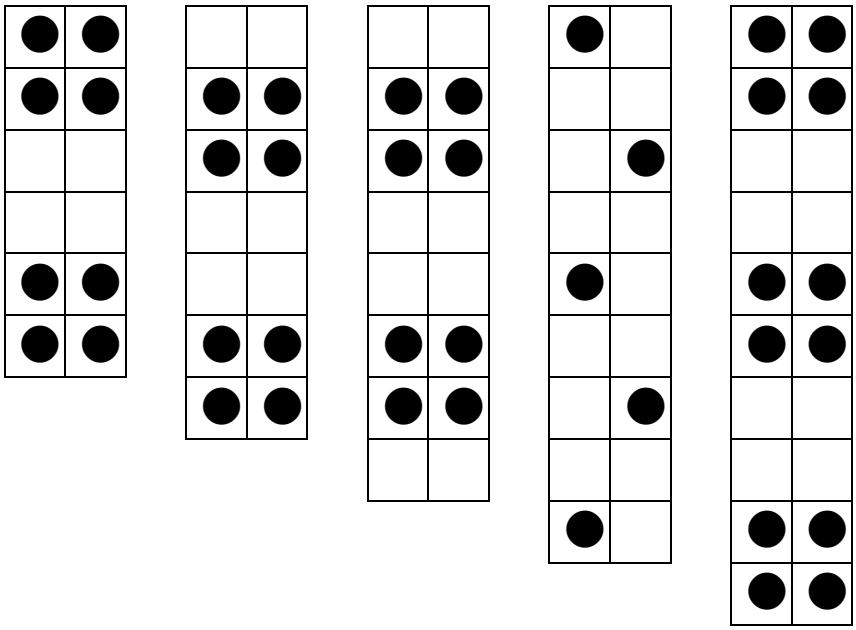


N = 2，y = 4：

2x1 2x2 2x3 2x4 2x5

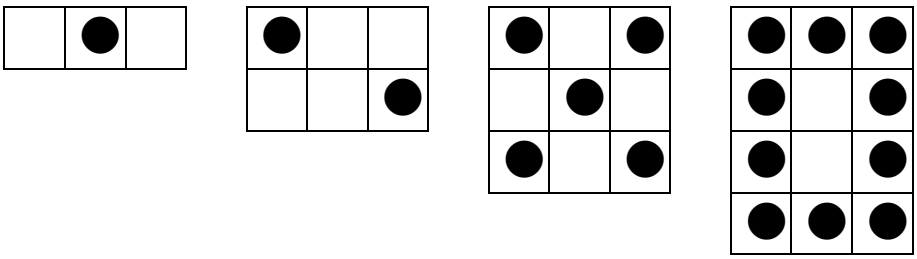


2x6 2x7 2x8 2x9 2x10

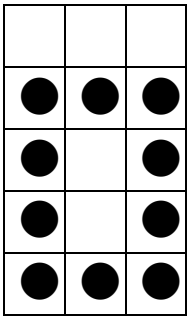


N = 3，y = 6：

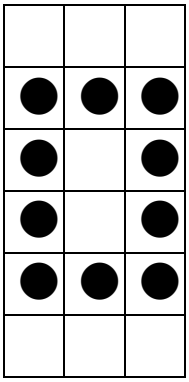
3x1 3x2 3x3 3x4



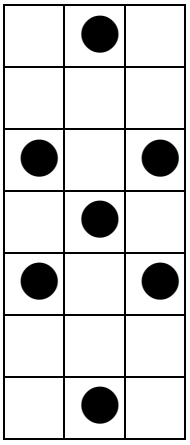
3x5



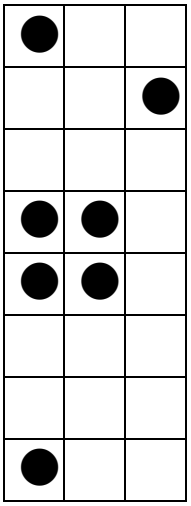
3x6



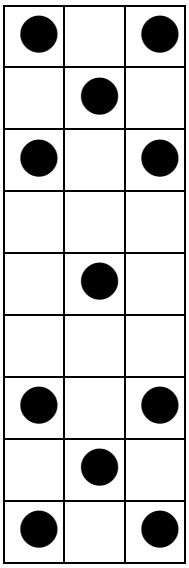
3x7



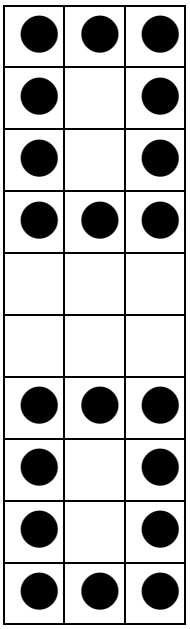
3x8



3x9



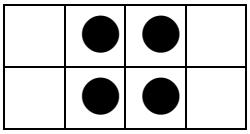
3x10



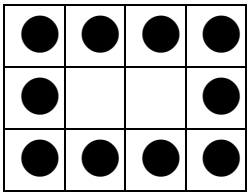
N = 4，y = 5：

4x1

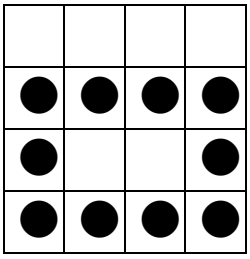
4x2



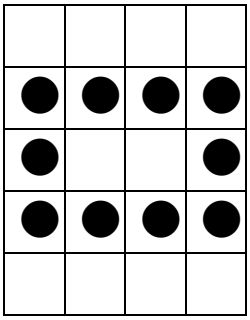
4x3



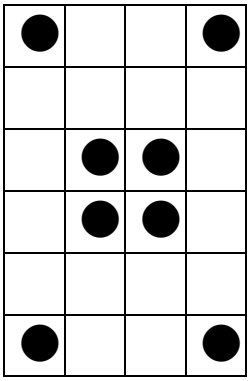
4x4



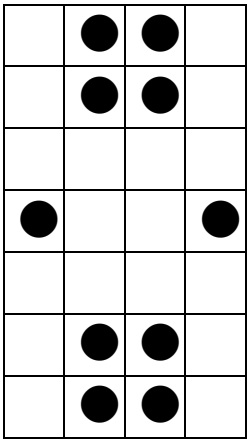
4x5



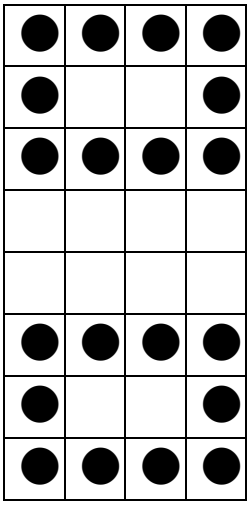
4x6



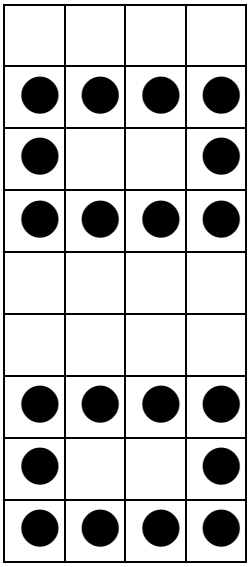
4x7



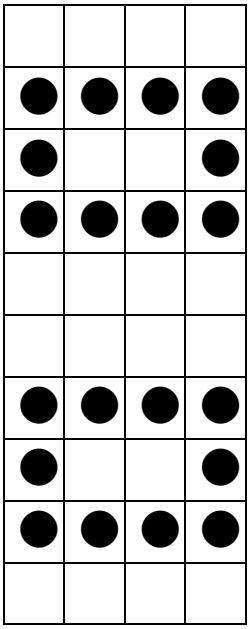
4x8



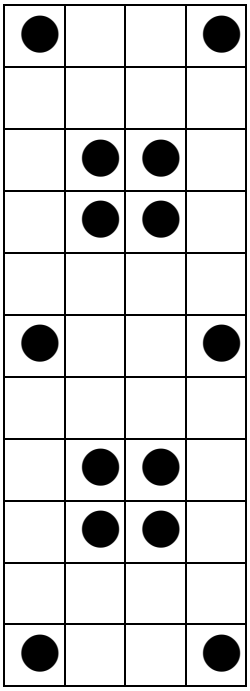
4x9



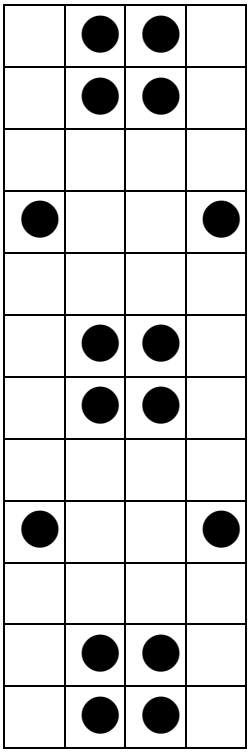
4x10



4x11



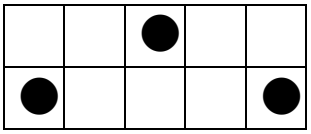
4x12



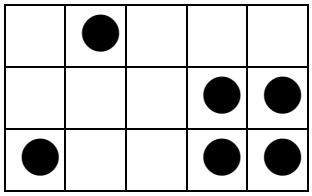
N = 5，y = 24：

5x1

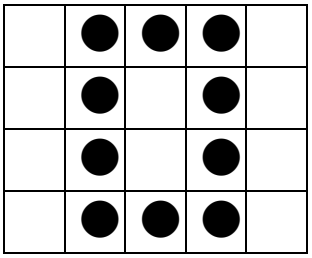
5x2



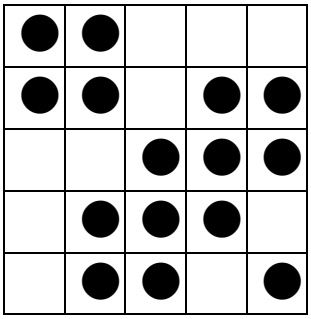
5x3



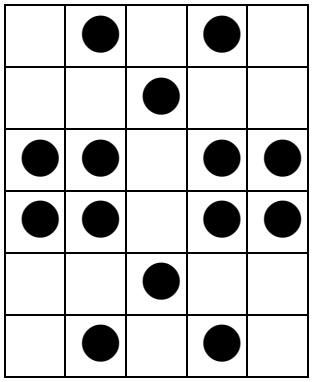
5x4



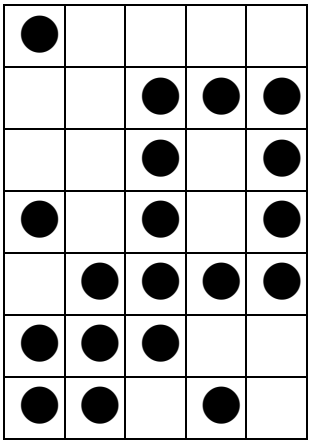
5x5



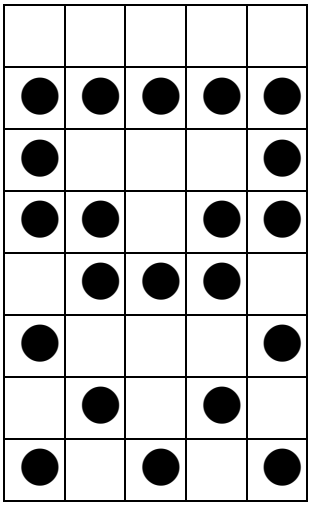
5x6



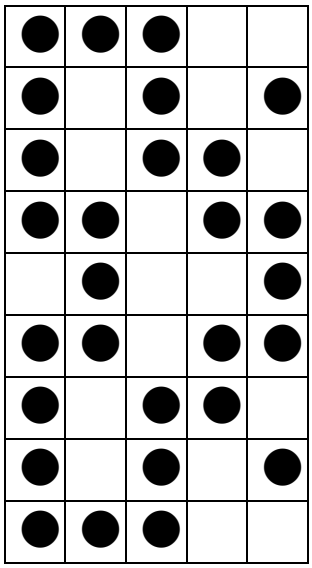
5x7



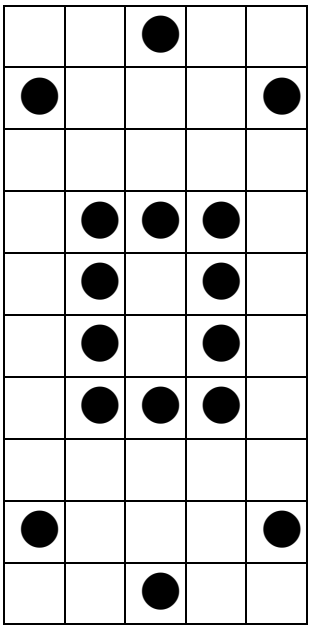
5x8



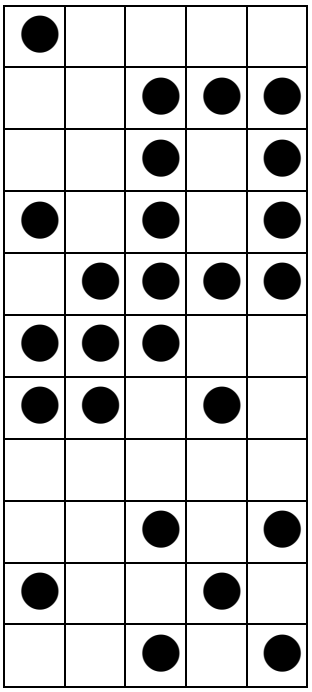
5x9



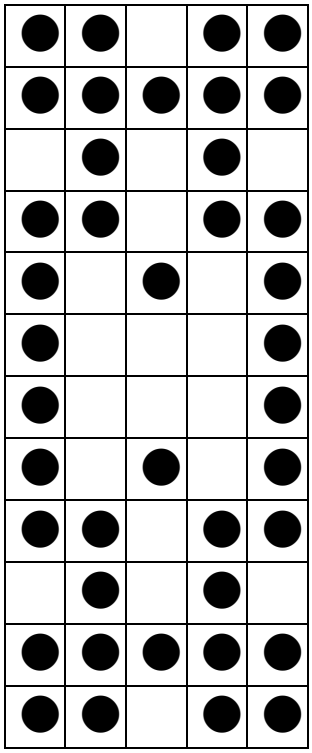
5x10



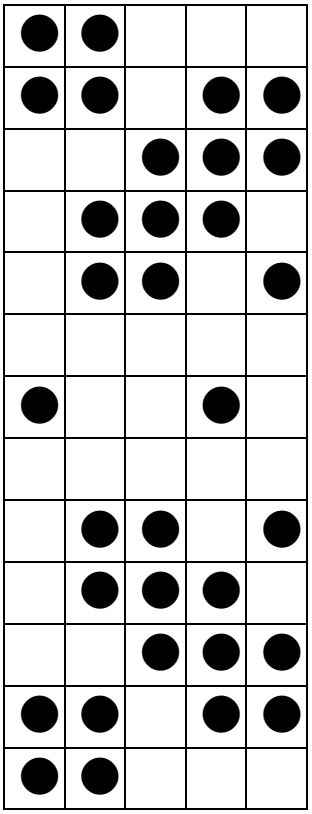
5x11



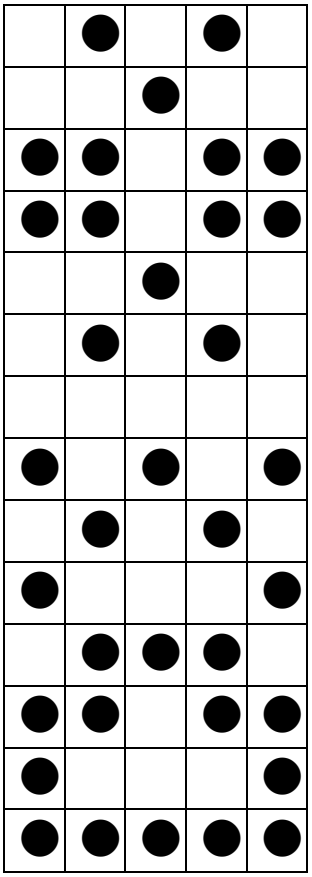
5x12



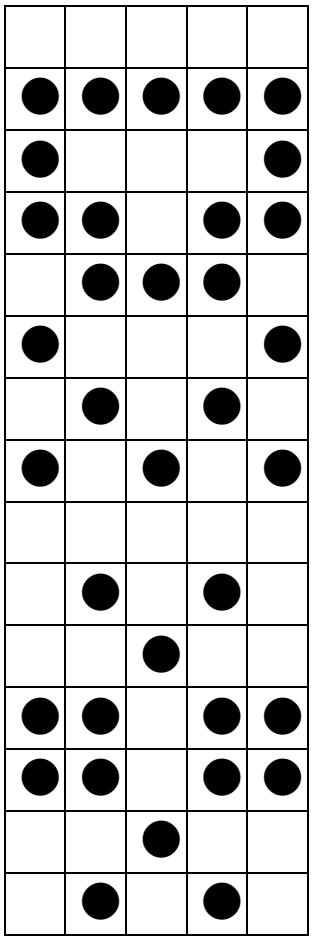
5x13



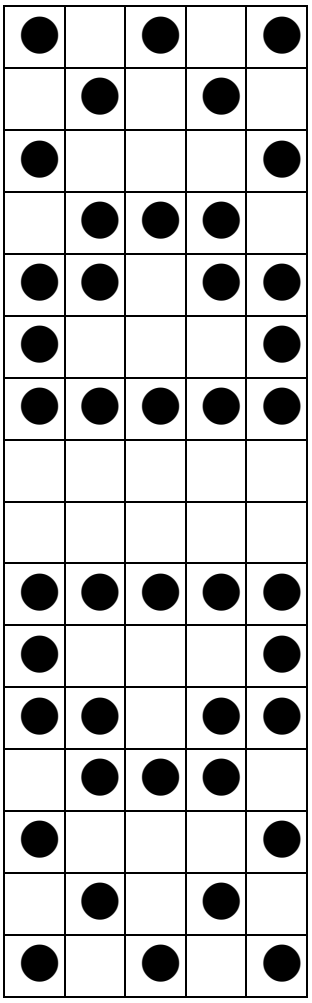
5x14



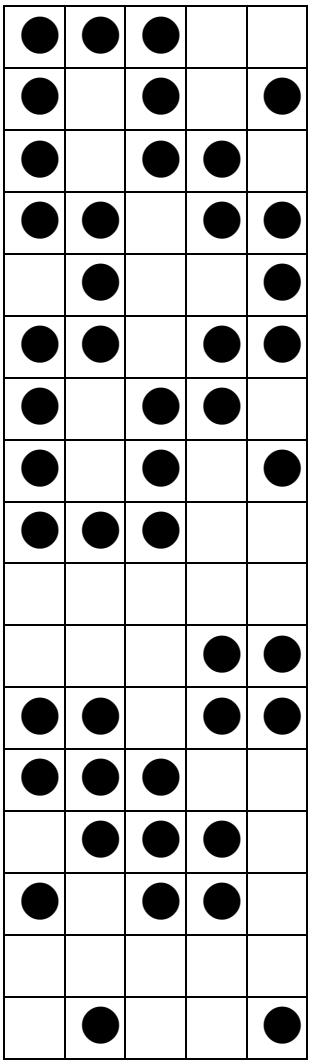
5x15



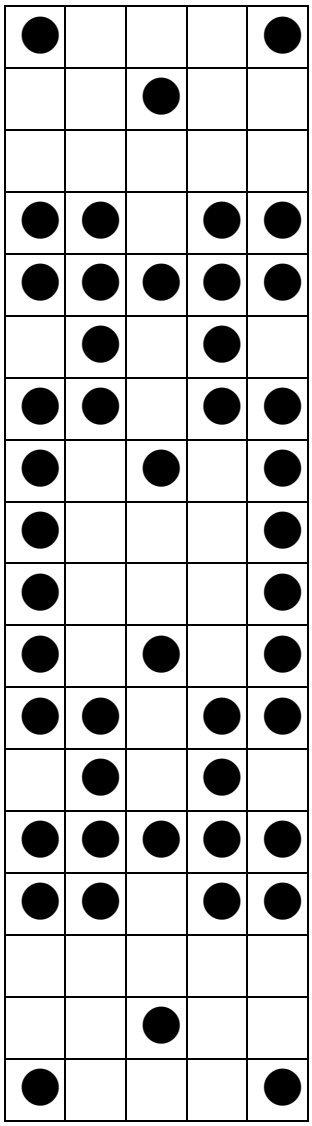
5x16



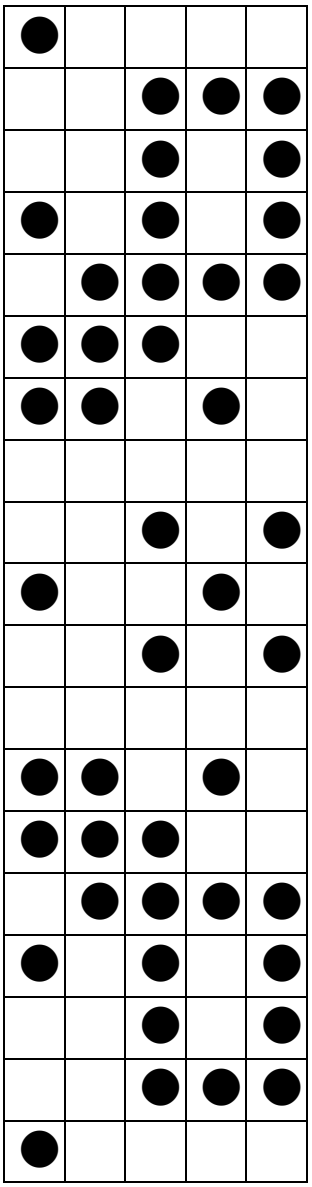
5x17



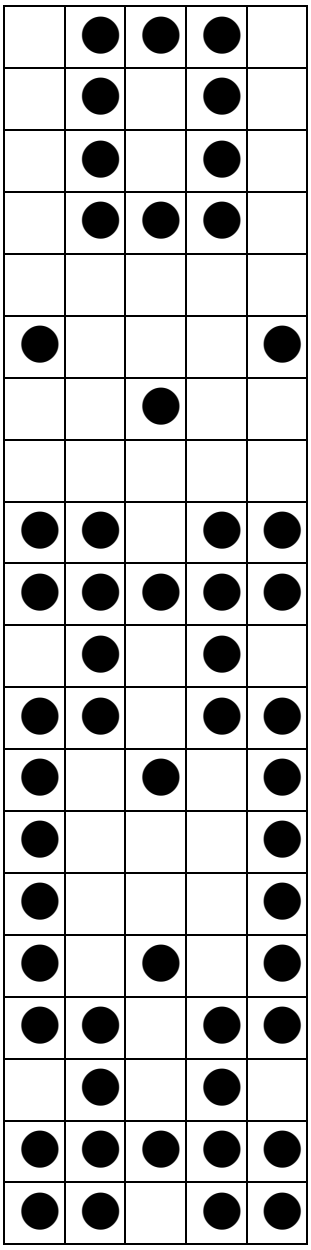
5x18



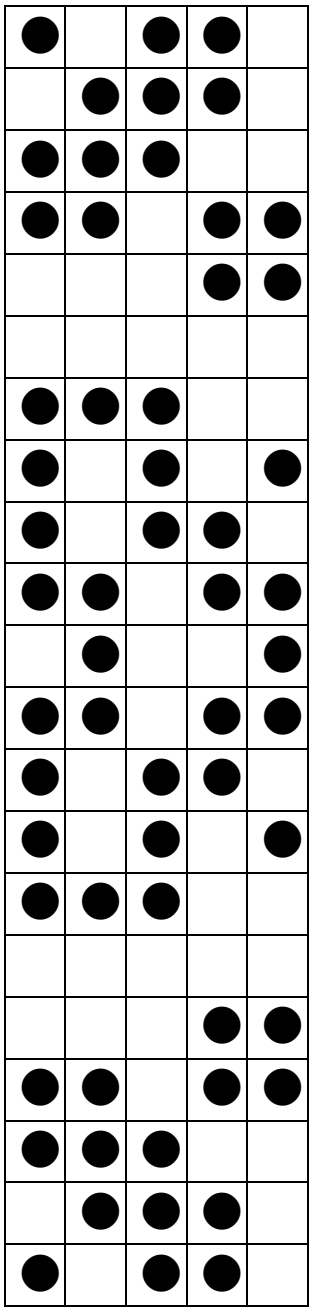
5x19



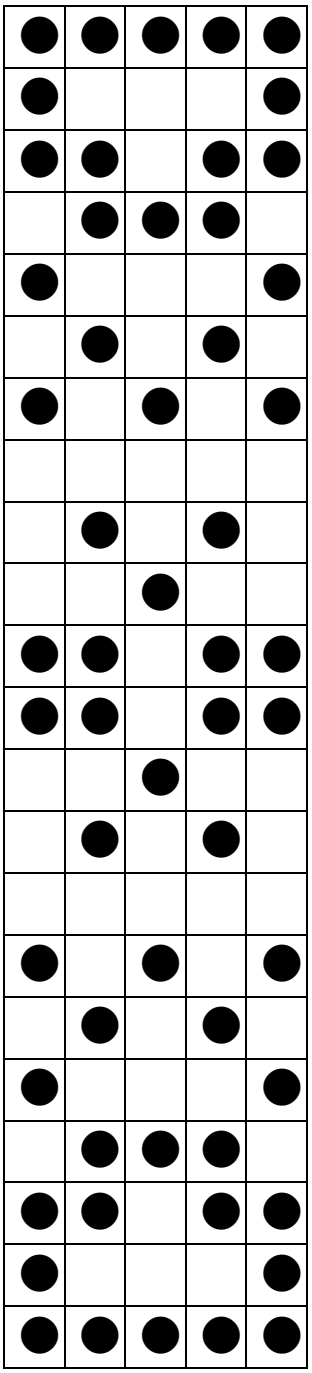
5x20



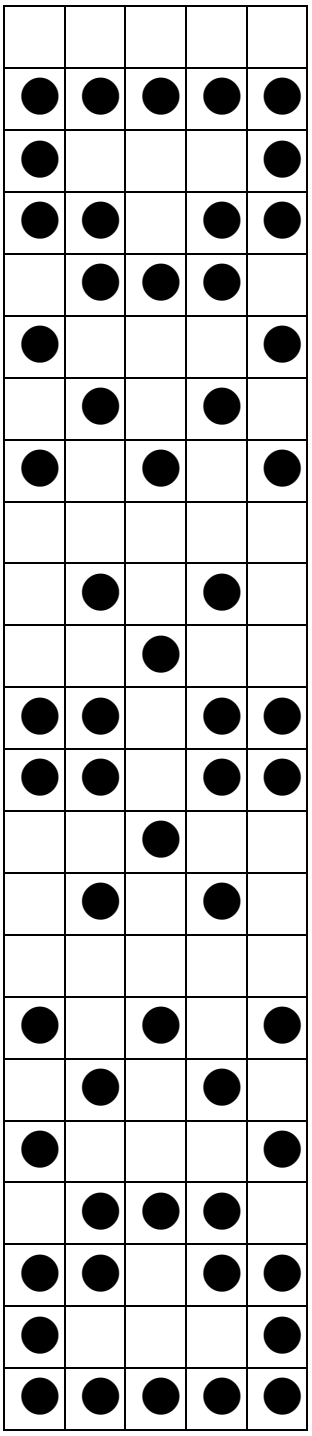
5x21



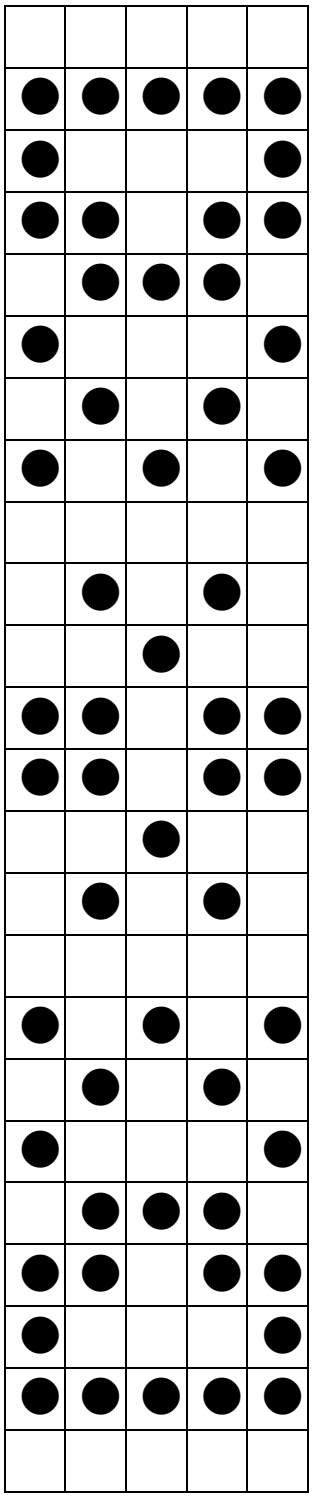
5x22



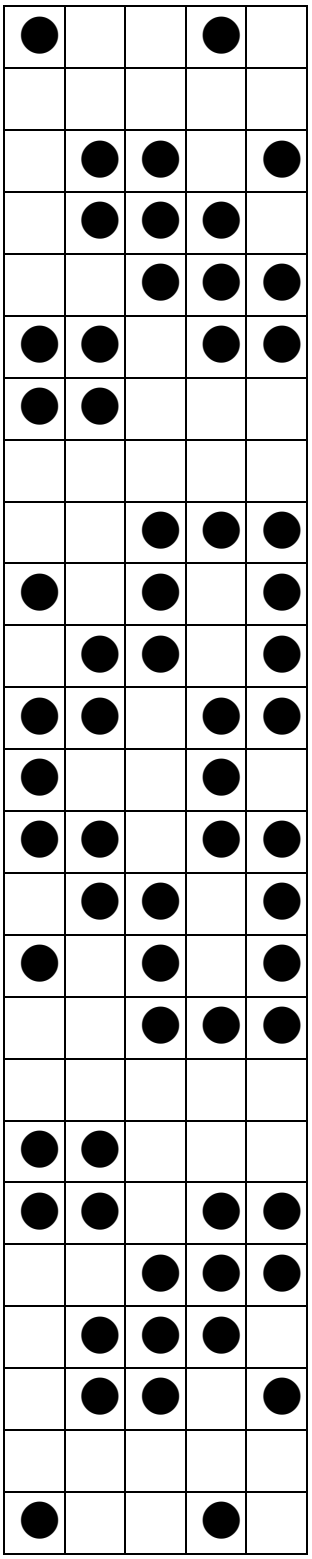
5x23



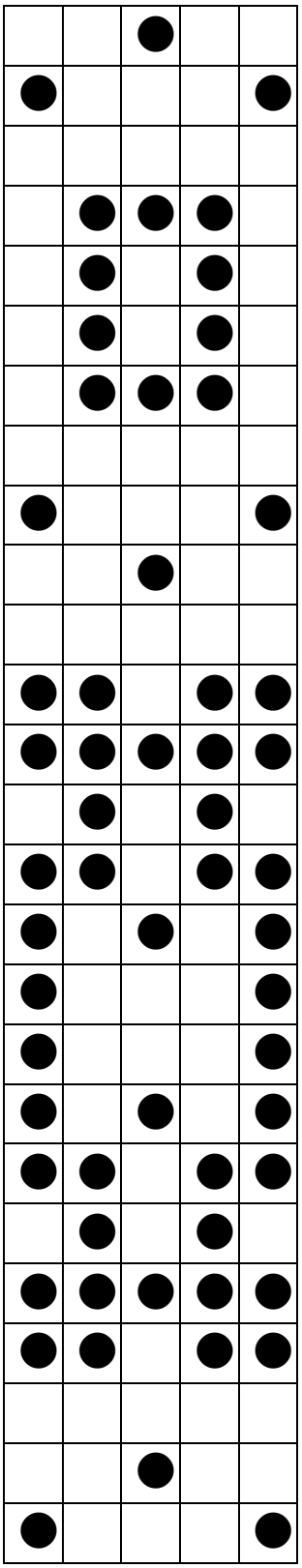
5x24



5x25



5x26



參考文獻

1. 陳昶安、林順喜，欄數少於 20 之點燈遊戲更快速的電腦解法之研究，師大資工系大四專題研究報告，2009。
2. 蔡明原、林順喜，一個有趣的益智遊戲「點燈」之電腦解法及分析，中華民國資訊學會通訊，第一卷第四期，7-13 頁，1988。
3. 陳昱璇，全一問題的改良演算法研究，國立台灣師大資工所碩士論文，1997。
4. W. Y. C. Chen, X. Li, C. Wang and X. Zhang, "Linear time algorithms to the minimum all-ones problem for unicyclic and bicyclic graphs", Electronic Notes in Discrete Mathematics, Vol.17, pp.93-98, 2004.
5. J. Goldwasser, W. F. Klostermeyer, and H. Ware, "Fibonacci polynomials and parity domination in grid graphs", Graphs and Combinatorics, Vol.18, pp.271-283, 2002
6. O. P. Lossers, "Solution to problem 10197", American Mathematical Monthly, Vol.100-8, pp.806-807, 1993.
7. K. Sutner, "Problem 88-8", The Mathematical Intelligencer, Vol.10-3, 1988。
8. K. Sutner, "Linear cellular automata and the garden-of-eden", The Mathematical Intelligencer, Vol.11-2, pp.49-53, 1989.
9. Jaap's Puzzle Page, http://www.geocities.com/jaapsch/puzzles/lights.htm