110 學年度第二學期 圖形理論

Sudoku as SAT/SMT Problem solved by Microsoft Z3 Solver

Group：G group

姓名：王梓帆｜顏子傑｜林冠名

111 年 05 月 06 日

目 錄

內容

表目錄

圖目錄

第一章 前言

1-1　專案動機與目的

在圖形理論的研究領域，可滿足性（Satisfiability）的相關研究，一直是其中一個相當重要的研究對象，可滿足性模理論（Satisfiability Modulo Theories, SMT）的求解器（Solver），更是近幾年大家爭先恐後投入的範疇。現在網路上已經有眾多的 SMT Solver 在被廣為使用，而每次提出新的求解器，不免俗的都會與之前已經開源的求解器做過的內容，去做求解效率、執行時間等等的比較，這次 Project 主要實踐的目標，就是其中一個著名的範例。很多新的 SMT Solver 在被提出的同時，都會以「數獨」（Sudoku），作為與先前的求解器比較的內容，而數獨亦是一個最小著色數（Chromatic number）問題，更是圖形理論中重要的問題之一，這也讓我們不得不去深入探討這個題目，也才有了這次期末專案的最後方向。

1-2　專案內容

這次的專案主要有以下三個部分，第一個部分是將數獨的問題，調整成布林可滿足性問題（Boolean satisfiability problem, SAT），以及基於可滿足性模理論（Satisfiability Modulo Theories, SMT）可以求解的命題變量（Propositional Variables）與求解所需的限制條件（Constraints）；第二個部分則要將分別以傳統 C 語言、SAT 求解法以及 SMT 求解法，完成數獨求解器的撰寫；最後一個部分則要針對我們寫出來的幾個求解器，去做一系列的探討，以上就是本次專案的目標與貢獻。

第二章 文獻回顧

2-1　數獨的重要規則

數獨是一種數學邏輯遊戲，遊戲由 9 × 9 個格子組成，玩家需要根據格子提供的數字推理出其他格子的數字。遊戲設計者會提供最少 17 個數字使得解答謎題只有一個答案，這種遊戲只需要邏輯思維能力，與數字運算無關，雖然玩法簡單，但提供的數字卻千變萬化，所以也有不少教育者認為數獨是鍛鍊腦筋的好方法。

其主要規則如下：

- 第一：數獨的網格由 9 × 9 的空格組成，共 9 個九宮格。
- 第二：只能使用從 1 到 9 的數字。
- 第三：每個九宮格中只能含有數字 1 到 9，每個數字只能使用一次。
- 第四：每一行只能含有數字 1 到 9，每個數字只能使用一次。
- 第五：每一列只能含有數字 1 到 9，每個數字只能使用一次。
- 第六：當整個數獨網格都正確填滿數字，遊戲就會結束。

數獨（Sudoku）這個單字來自日文，但概念源自於十八世紀瑞士數學家歐拉，所發明的拉丁方塊（Latin Square）。方格裡擺幾個數字，乍看之下好像沒什麼，然而其有趣之處，就在其中推敲的過程，由於規則簡單，卻變化無窮，在推敲之中完全不必用到數學計算，只需運用邏輯推理能力，所以容易上手也容易入迷，在這次的 Project 查詢相關資料時，更可以發現到使用任何一個搜尋引擎鍵入「sudoku」或「數獨」後，有千百萬個符合的網頁將被條列出來，更可以驗證其流行的程度遠遠超乎想像。

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

圖 2-1-1　數獨的格式

2-2　著色（Coloring）問題 － 最小著色數（Chromatic Number）

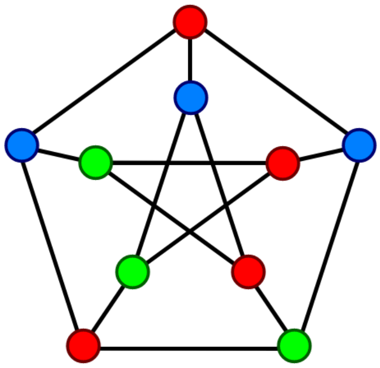


圖 2-2-1　佩特森圖（Petersen Graph）的著色問題

給定圖 G 的最小著色數記為 $X(G)$，是在滿足「相鄰頂點接收不同的顏色」的前提下，標記所有頂點所需的最小顏色數，若我們使用 k 種顏色對圖進行著色，則這種著色便被稱作 k-著色（k-coloring），例如圖 2-2-1 即為對於佩特森圖（Petersen Graph）的「3-著色（3-coloring）」。對於一個最大度數（degree）為 d 的無向圖，我們需要不超過 $d+1$ 種顏色即可完成對圖的著色，而對給定圖完成著色所需的顏色數目的最小值就是該圖的最小著色數(Chromatic number)。求一個圖的最小著色數是一個 NP-完全(NP-Complete)問題，目前沒有多項式時間（Polynomial time）內的算法。

2-3　數獨與著色問題之間的對應關係

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	3

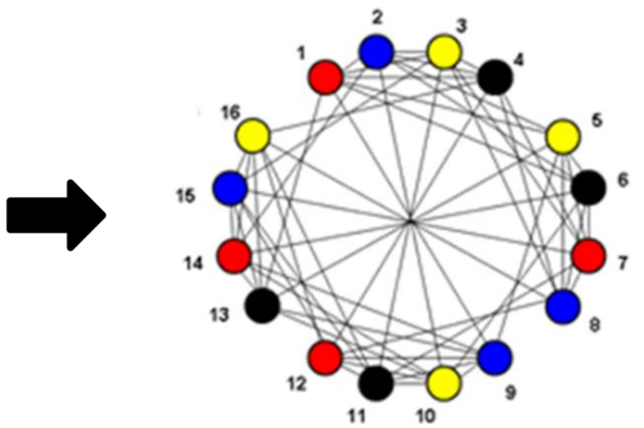


圖 2-3-1　數獨的頂點相鄰關係

我們藉助圖形理論來解數獨問題，關鍵就是將數獨轉化為圖著色問題。為了簡化說明，我們先以 4 × 4 的數獨為例，如圖 2-3-1 所示，我們把數獨中每一個格子看做一個頂點，並且連接在同一行、同一列，或者同一個九宮格的任意兩個頂點。

如果我們能對該圖進行 4-著色，則我們就解決了數獨問題。同樣我們可以將該方法擴展到 9 × 9 的網格或者其他尺寸。數獨的題目，在一開始已經提供了一些格子中的數字，我們可以將相對應的頂點視為已經著上了某種顏色。但是需要注意的是，以 4 × 4 的數獨為例，對應的圖的最大度數為 7，所以只能保證使用的顏色不超過 8 種而並不能保證只採用 4 種顏色。所以還不足夠解決所有的數獨問題，我們還需要配合其它更加先進的著色演算法才能完美解決數獨問題。除了解決數獨問題，圖著色在現實生活中還有許多應用，例如鐵路調度，時間表安排等等。如何快速高效的對圖（尤其是大規模的圖）進行著色也是一個非常熱門的問題。除此之外，值得一提的是數獨也是一個 NP-完全問題，更多在有關數獨問題建模和解決的介紹，可以查看 Modeling Sudoku Puzzles with Python 這篇文章。

2-4　數獨的填入範例

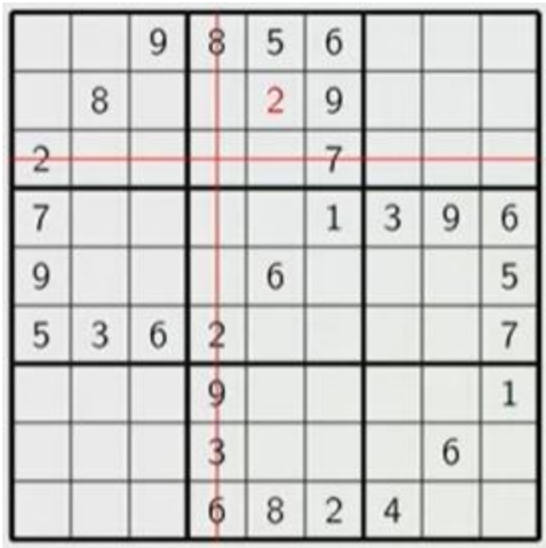


圖 2-4-1　數獨的填入範例

就上述的數獨題目，我們可以在紅色標示的地方填入 2，因為這樣的填法將滿足所有的主要規則，最重要的是滿足在每一行、每一列以及每一個九宮格中，數字 1 到 9 都恰好僅出現一次，透過這樣的做法，以單雙向掃描各行、列與九宮格的各種可能，就可以得到最後的解，當然，針對數獨的問題，也存在許多各式各樣的輔助解法，但這並不在今天討論的範圍內，所以這部分就不再多做贅述。

2-5　數獨作為布林可滿足性問題（Boolean satisfiability problem, SAT）解法

在將數獨問題描述成 SMT 問題之前，要先將其轉換為 SAT 可以求解的形式，這邊的作法是參考 I Lynce 與 J Ouaknine 提出的「Sudoku as a SAT Problem」[1]。

2-5-1 SAT 的命題變量（Propositional Variables）

使用 SAT 求解問題時，會先面臨到一個問題，那就是要怎麼將每個情況，轉成 SAT Solver 可以吃的布林變數，針對每個 9×9 的輸入，共有 81 個單元（cell），每個都可能是 0 到 9 的其中一種數字，所以如果要將編碼寫成合取範式（Conjunction Normal Form, CNF），總共需要宣告 $9^3 = 729$ 個布林變數作為命題變量，才能完成編碼。之後的介紹裡我們會使用 s_{xyz} 作為變數的指稱，而若且唯若 x 列（row）和 y 行（column）中的條目被分配編號 z 時，變量 s_{xyz} 才被分配為真。舉例來說，$s_{483} = 1$ 意味著該數獨網格的第 4 行第 8 列填的數字為 3，座標可以表示為 S [4, 8] = 3。

2-5-2 如何訂定 SAT Solver 限制條件（Constraints）

關於限制條件的編碼，從拉丁網格的角度看，又可以分為解數獨問題上最直觀的擴充編碼法，以及除去冗餘限制條件的最小編碼法，這邊提到的編碼方法是參考 Carla P. Gomes 與 David Shmoys 於 2002 年提出的 Completing Quasigroups or Latin Squares: A Structured Graph Coloring Problem [2]。該篇論文探討的對象為拉丁方陣的編碼，拉丁方陣（Latin Square）是一種 n × n 的方陣，在這種 n × n 的方陣裡，恰有 n 種不同的元素，每一種不同的元素在同一行或同一列裡只出現一次。

在我們的 SAT 編碼中，每個布林變量x_{ijk}表示將顏色 k 分配給單元格（entry）i, j，其中 i, j, k = 1, 2, ..., n；n 是順序。共 n^3 個變量。最基本的編碼，我們稱之為最小編碼（Minimal Encoding），包括表示以下條件限制的子句（clause）：

1. 必須為每個單元分配一些顏色（長度為 n 的子句）：

$$\forall_{ij}\ (x_{ij1} \vee x_{ij2} \vee \ldots \vee x_{ijn}) \qquad (1)$$

2. 同一列（row）沒有顏色重複（負二元子句集）：

$$\forall_{ik}\ (\neg x_{i1k} \vee \neg x_{i2k})(\neg x_{i1k} \vee \neg x_{i3k}) \ldots (\neg x_{i1k} \vee \neg x_{ink}) \qquad (2)$$

3. 同一行（column）中沒有顏色重複（負二元子句集）：

$$\forall_{jk}\ (\neg x_{1jk} \vee \neg x_{2jk})(\neg x_{1jk} \vee \neg x_{3jk}) \ldots (\neg x_{1jk} \vee \neg x_{njk}) \qquad (3)$$

限制條件 1 成為每個單元的長度為 n 的子句，並且 2 和 3 成為否定二元子句的集合。子句的總數是 $O(n^4)$。

拉丁方格的二進制表示可以看作是一個立方體，其中維度是行、列和顏色。這種觀點揭示了另一種表述拉丁方性質的方法：通過保持兩個維度固定而確定的任何一組變量必須恰好包含一個真實變量。擴展編碼（Extended Encoding）通過還包括以下限制條件來捕獲此條件：

1. 每種顏色在每一列（row）中至少出現一次（長度為 n 的子句）：

$$\forall_{ik}\ (x_{i1k} \vee x_{i2k} \vee \ldots \vee x_{ink}) \tag{4}$$

2. 每種顏色在每行（column）中至少出現一次（長度為 n 的子句）：

$$\forall_{jk}\ (x_{1jk} \vee x_{2jk} \vee \ldots \vee x_{njk}) \tag{5}$$

3. 沒有兩種顏色分配給同一個單元格（entry）（負二元子句集）：

$$\forall_{ij}\ (\neg x_{ij1} \vee \neg x_{ij2})(\neg x_{ij1} \vee \neg x_{ij3}) \ldots (\neg x_{ij1} \vee \neg x_{ijn}) \tag{6}$$

這兩種編碼方法，如果應用在數獨上，最為直觀的反而是擴充編碼法，這邊所使用的布林變量將以第 2-5-1 節提到的 s_{xyz} 作為變數的指稱，而若且唯若 x 行和 y 列中的條目被分配編號 z 時，變量 s_{xyz} 才被分配為真。

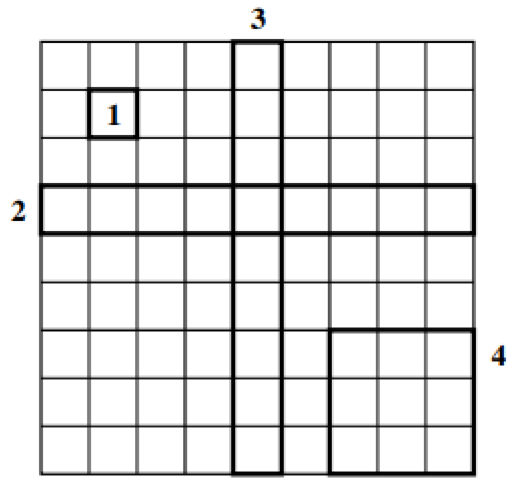


圖 2-5-1　數獨的限制條件

數獨的限制條件簡單可以分成四塊如上圖所示，分別是針對每個單元格(entry)、列（row）、行（column）以及九宮格，接下來會講解如何以不同的編碼方法解決這個問題以最小編碼的方式來做定義問題的限制條件的話，可以表示成以下的四種限制條件子句：

1. 每個單元格（entry）中至少有一個數字：

$$\bigwedge_{x=1}^{9}\bigwedge_{y=1}^{9}\bigvee_{z=1}^{9} s_{xyz} \tag{7}$$

2. 每個數字（顏色）在每一列（row）中最多出現一次：

$$\bigwedge_{y=1}^{9}\bigwedge_{z=1}^{9}\bigwedge_{x=1}^{9}\bigwedge_{i=x+1}^{9}(\neg s_{xyz} \vee \neg s_{iyz}) \tag{8}$$

3. 每個數字（顏色）在每一行（column）中最多出現一次：

$$\bigwedge_{x=1}^{9}\bigwedge_{z=1}^{9}\bigwedge_{y=1}^{9}\bigwedge_{i=y+1}^{9}(\neg s_{xyz} \vee \neg s_{xiz}) \tag{9}$$

4. 每個數字（顏色）在每一個九宮格中最多出現一次：

$$\bigwedge_{z=1}^{9}\bigwedge_{i=0}^{2}\bigwedge_{j=0}^{2}\bigwedge_{x=1}^{3}\bigwedge_{y=1}^{3} \bigwedge_{k=y+1}^{9}(\neg s_{(3i+x)(3j+y)z} \vee \neg s_{(3i+x)(3j+k)z}) \tag{10}$$

在最小編碼中，除了代表預分配（pre-assigned）條目的單元子句，生成的 CNF 將有 8829 個子句。在這些子句中，81 個子句是九元的，其餘 8748 個子句是二元的。九元子句來自一組「至少一個」子句 $(9 \times 9 = 81)$，而$3 \times 9 \times 9 \times C_{9,2} = 8748$ 個二元子句來自三組「最多一個」子句總計需要 $81 + 8748 = 8829$ 個限制條件。

以擴充編碼的方式來做定義問題的限制條件的話，擴展編碼明確斷言網格中的每個條目都只有一個數字，對於每一行、每一列和每個 3×3 九宮格也是如此。擴展編碼包括最小編碼的所有子句，以及以下四種限制條件子句：：

1. 沒有兩種顏色分配給同一個單元格（entry）（負二元子句集）=> 至多一個數字在一個單元格（entry）：

$$\bigwedge_{x=1}^{9}\bigwedge_{y=1}^{9}\bigwedge_{z=1}^{9}\bigwedge_{i=z+1}^{9}(\neg s_{xyz} \vee \neg s_{xyi}) \tag{11}$$

2. 每個數字（顏色）在每一列（row）中至少出現一次：

$$\bigwedge_{y=1}^{9}\bigwedge_{z=1}^{9}\bigvee_{x=1}^{9} s_{xyz} \tag{12}$$

3. 每個數字（顏色）在每一行（column）中至少出現一次：

$$\bigwedge_{x=1}^{9}\bigwedge_{z=1}^{9}\bigvee_{y=1}^{9} s_{xyz} \qquad (13)$$

4. 每個數字（顏色）在每一個九宮格中至少出現一次：

$$\bigvee_{z=1}^{9}\bigwedge_{i=0}^{2}\bigwedge_{j=0}^{2}\bigwedge_{x=1}^{3}\bigwedge_{y=1}^{3} s_{(3i+x)(3j+y)z} \qquad (14)$$

在擴展編碼中，除了代表預分配（pre-assigned）條目的單元子句，生成的 CNF 將有 11988 個子句。在這些子句中，324 個子句是九元的，其餘 11664 個子句是二元的。九元子句來自四組「至少一個」子句 $(4 \times 9 \times 9 = 324)$，而$4 \times 9 \times 9 \times C_{9,2} = 11664$ 個二元子句來自四組「最多一個」子句總計需要 $324 + 11664 = 11988$ 個限制條件。

2-6 基於可滿足性模理論（Satisfiability Modulo Theories, SMT）的解法[3]

最後我們要將其轉換為 SMT 可以求解的形式，這邊的作法是參考 Milan Bankovi´c and Filip Mari´提出的「An Alldifferent Constraint Solver in SMT」[3]。以 SMT 求解器求解的話，問題的限制就不需要那麼繁瑣了，因為多了很多線性條件，除了可以減少命題變量之外，也可以減少限制條件需要用到的子句數量。

2-6-1 SMT 的命題變量（Propositional Variables）

關於 SMT 的命題變量，只需要針對座標進行定義就可以了，因為數字的部分不再需要定義成只能顯示 0/1 的布林變量，可以定義成整數變量，以一個變量表示填入的數字 1 到數字 9。所以僅需要宣告 $9^2 = 81$ 個整數變數作為命題變量，就可以完成編碼。之後的介紹裡我們會使用 S_{xy} 作為變數的指稱，而第 x 列（row）、y 行（column）的單元格（entry）中的整數數值，就是該單元格填入的數值。舉例來說，$s_{48} = 3$ 意味著該數獨網格的第 4 行第 8 列填的數字為 3，座標可以表示為 S [4, 8] = 3。

2-6-2 如何訂定 SMT Solver 限制條件（Constraints）

接下來我們會帶入如何從 SMT Solver 的角度，去定義限制條件，因為可以使用上下界範圍的限制，所以其實主要分成兩個部分即可達到最小編碼法可以完成的編碼效果。可以表示成以下的四種限制條件子句：

1. 每個單元格（entry）中隨機填入一個數字 0 到 9：

$$\bigwedge_{x=1}^{9}\bigwedge_{y=1}^{9}\left(s_{xy}>0\right)\wedge\left(s_{xy}<10\right) \tag{15}$$

2. 每一列（row）的數字都不一樣：

$$\bigwedge_{y=1}^{9}\bigwedge_{i=1}^{9}alldifferent\left(s_{iy}\right) \tag{16}$$

3. 每一行（column）的數字都不一樣：

$$\bigwedge_{x=1}^{9}\bigwedge_{i=1}^{9}alldifferent\left(s_{xi}\right) \tag{17}$$

4. 每一九宮格的數字都不一樣：

$$\bigwedge_{i=0}^{2}\bigwedge_{j=0}^{2}\bigwedge_{x=1}^{3}\bigwedge_{y=1}^{3}alldifferent\left(s_{(3i+x)(3i+y)}\right) \tag{18}$$

照這樣的算法下去看，只需要$9\times9+3*9=98$個限制條件子句，就可以完成整個數獨求解器的定義，定義命題變量的部分也只需要 81 個宣告子句即可。整體來說限制數量有極大幅的成長。

2-7 SAT 與 SMT Solver 共同的限制條件子句

當然，SMT 與 SAT 求解器，存在共同的限制條件，但因為每個不同題目會有不一樣的限制子句，所以這邊獨立出來撰寫，這個限制條件就是題目已經填好數字的空格。假設今天的題目第 4 行第 8 列填的數字為 3，以 SAT 表示的限制條件可以寫作：

$$s_{483}=true \tag{21}$$

而以 SMT 表示的限制條件，則可以寫作：

$$s_{48}=3 \tag{21}$$

加上這個限制條件之後，就可以完成 SAT/SMT 的數獨求解器，以求解不同的 9×9 數獨網格題目輸入，並輸出填充完成的最後結果。

2-8 SMT Solver – Z3 求解器[4]

可滿足性模理論問題是邏輯一階公式關於背景理論組合的決策問題，例如：算術、位元向量、數列和未解釋的函數。 Z3 是一種新的高效 SMT 求解器，可從 Microsoft Research 免費獲得。由 Microsoft Research 開發，用於檢查邏輯表達式的可滿足性，且內部擁有許多的函式可以使用，關於內部函式的使用，我們有參考 Leonardo de Moura 以及 Nikolaj Bjørner 於 2008 年出版的「Z3: An efficient SMT solver」[4]，過程中難免碰到一些與認知不同的函數使用方式，則參考網路上許多人針對限制條件撰寫的方法，所進行的學術討論，作為我們程式撰寫的依據。

第三章 Project 程式撰寫

3-1 以回測方法去實踐數獨求解器

回測（Backtracking）的作法，在大學部的課程有概略介紹過，也可以應用在這次的 Project 上，除了可以求解之外，還可以計算需要執行的步數，更可以觀察出數獨是是一個 NP-完全(NP-Complete)問題，目前沒有多項式時間（Polynomial time）內的算法。

3-1-1 定義的結構（Struct）

定義結構 EntryData 儲存三個資訊，分別是對應數獨網格位置的 x, y 座標，以及該格所填入的數值。並以該資料結構，定義一個 9 × 9 的二維矩陣，存取數獨的各座標對應的數值。

3-1-2 函數與功能簡介

- fileInput：用來讀取題目輸入的文字檔
- reset：用來讀取題目輸入的文字檔
- print：輸出數獨的 9 × 9 矩陣
- notValid：確認輸入是否符合規則
- checkEmpty：找出原本沒有填入數值的空格（即輸入填 0 的部分）
- checkPlacement：判斷該格能否填入數值
- solve：果我們找到解決方法則回傳 1 否則回傳 0

3-1-3 主程式功能撰寫

程式執行的流程圖如下圖所示，首先進行讀檔，接著判斷是否符合數獨的規則：每個九宮格中只能含有數字 1 到 9，每個數字只能使用一次；每一行只能含有數字 1 到 9，每個數字只能使用一次；每一列只能含有數字 1 到 9，每個數字只能使用一次。接著判斷這次的輸入有沒有解，填上一個數字後判斷是否已經填完，是的話程式結束，否的話程式則回到判斷填上數字後的數獨是否違反規則，除了獨檔之外，其他一直重複上述步驟值到填完整張表格即可完成數獨的求解。

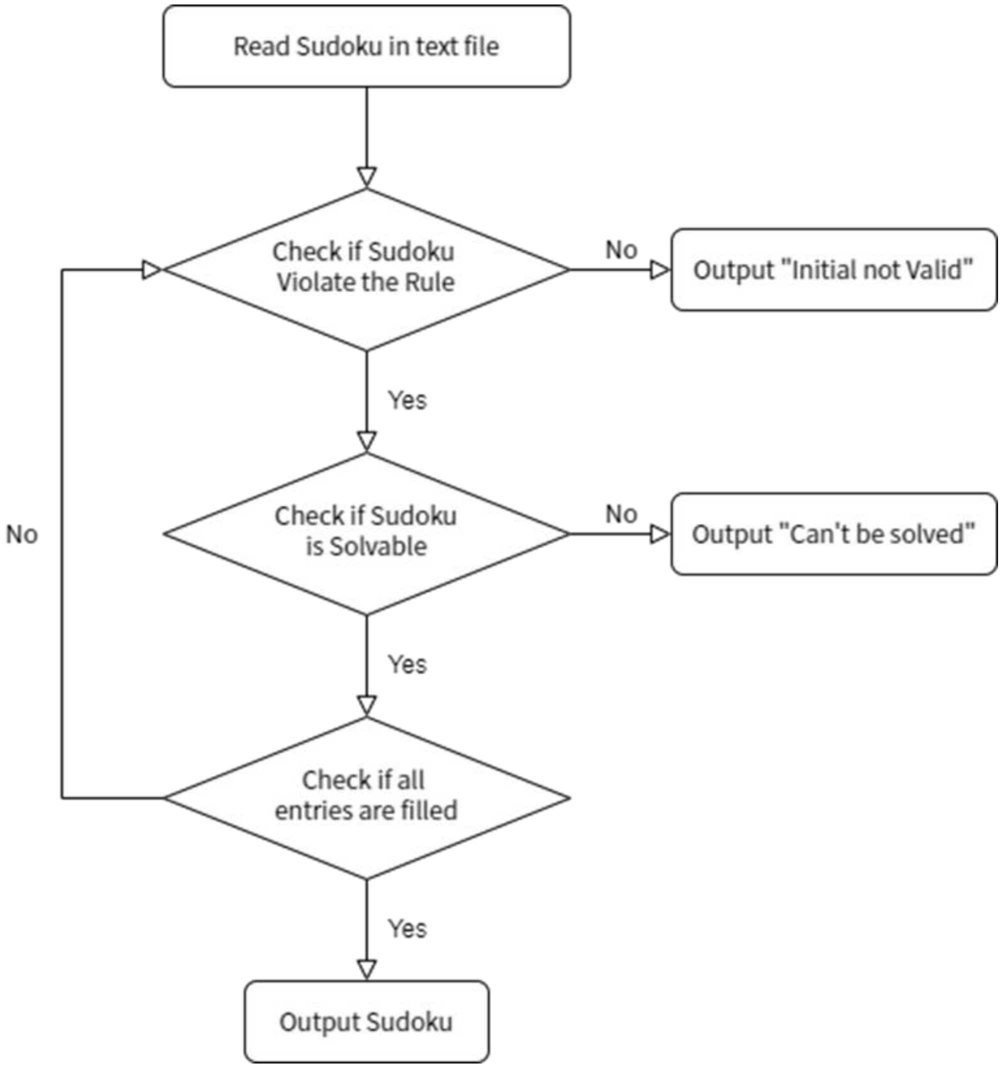


圖 3-1-1 以回測法求解數獨的程式流程圖

3-2　以 SAT Solver 去實踐數獨求解器

首先一樣是讀入數獨的輸入，接著會開始建模，首先定義 729 個布林變數，接著根據已經填入的數字，加上 equal 的條件限制，再加上每個九宮格、行以及列中只能含有數字 1 到 9，每個數字只能使用一次的條件，最後再看看是否有解，並輸出求解的結果或是無法求解的敘述。

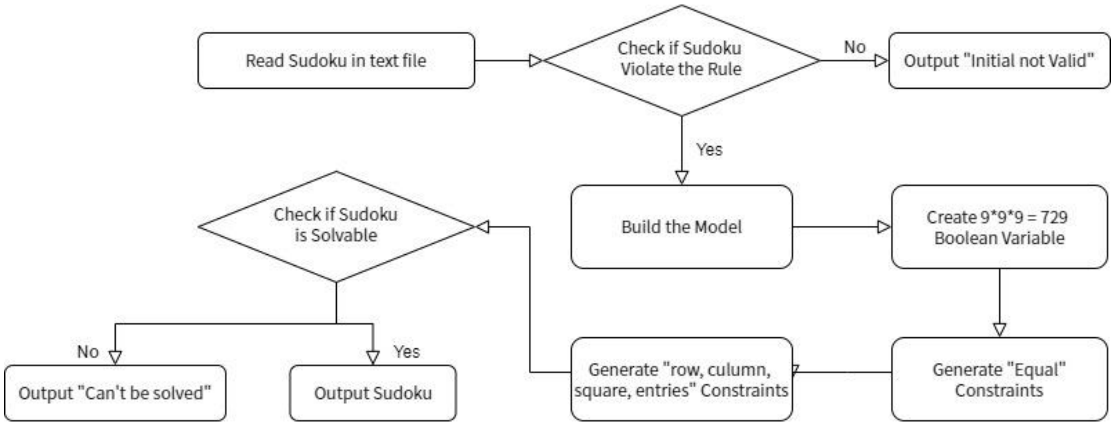


圖 3-2-1 以 SAT 求解數獨的程式流程圖

3-3　以 SMT Solver 去實踐數獨求解器

SMT Solver 的解題步驟其實與 SAT Solver 的解題步驟差不多，差別在於建構限制條件的地方，以及定義命題變量時，只需要定義 81 個整數變數，條件限制的部分前面有作比較詳細的敘述，這邊就不多作贅述。

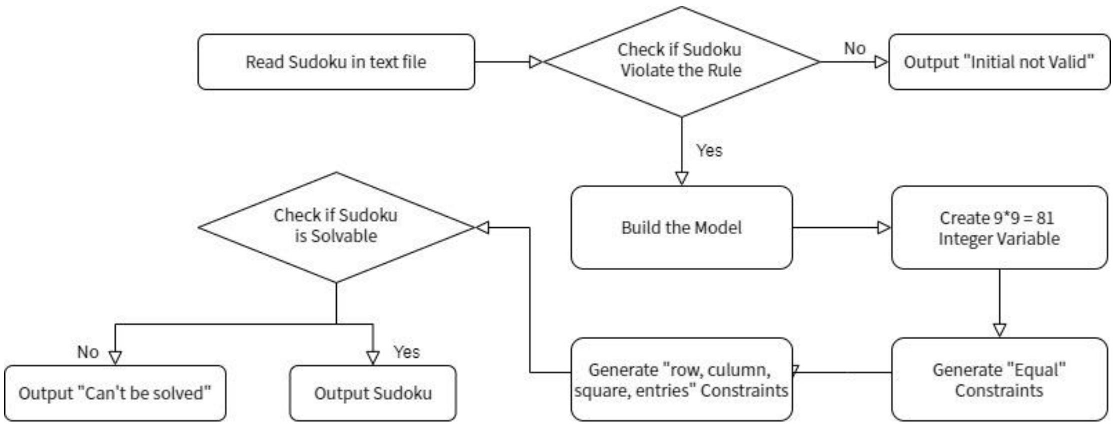


圖 3-3-1 以 SMT 求解數獨的程式流程圖

第四章 實驗結果

4-1 輸入的形式

輸入的形式為一個純文字（.txt）檔，並且以逗號將輸入的數字分開，輸入若為 0 則表示該格尚未填入任意數值，輸入為數字 1 到數字 9 則代表該格所填的數字。

```
5,3,0,0,7,0,0,0,0
6,0,0,1,9,5,0,0,0
0,9,8,0,0,0,0,6,0
8,0,0,0,6,0,0,0,3
4,0,0,8,0,3,0,0,1
7,0,0,0,2,0,0,0,6
0,6,0,0,0,0,2,8,0
0,0,0,4,1,9,0,0,5
0,0,0,0,8,0,0,7,9
```

圖 4-1-1 數獨的輸入

4-2 輸出的形式

輸出分為三個部份，第一部分為輸出 input 給定的數獨，以”|”和”—”區分不同的 square。第二部分為輸出所建立的限制條件，分為

(1) the cell has a value
(2) The value of each cell must be 1-9
(3) in each row, each digit must appear exactly once.
(4) in each column, each digit must appear exactly once.
(5) in each 3x3 square, each digit must appear once.

第三部分為輸出 SMT solver 求解的結果，若求解成功會如圖(4-2-1)，在適當的格子填入正確的數字，最後再輸出文字 The final SUDOKU set up is valid。若求解失敗會如圖(4-2-2)，輸出文字 The puzzle is unsolvable。

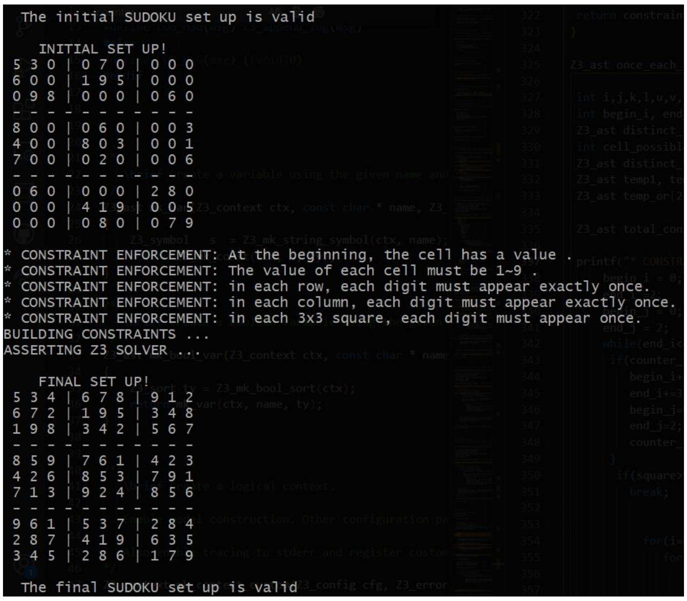


圖 4-2-1　數獨求解正確的輸出

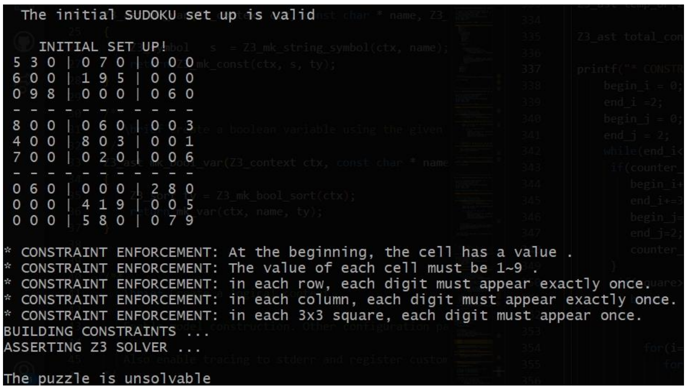
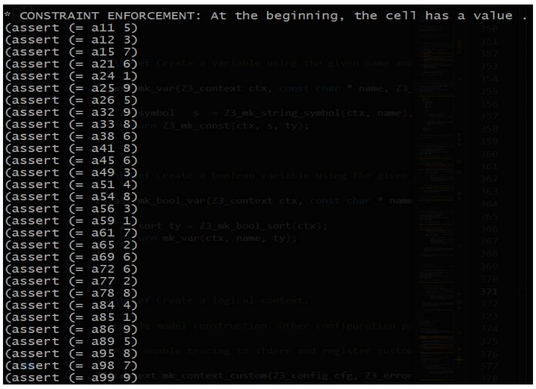


圖 4-2-2　數獨求解失敗的輸出

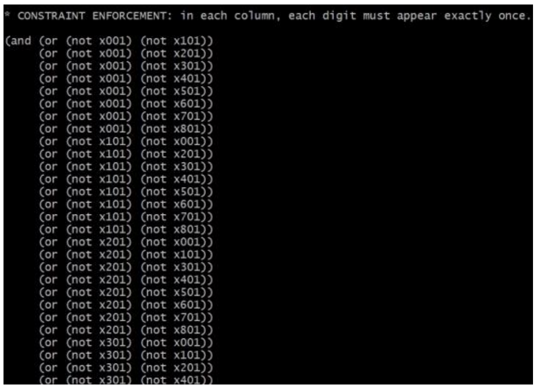
4-3　SAT 的限制條件

SAT 的限制條件總共分為五個部分，第一部分為對於已經給定 cell 之值，直接賦予它給定的值，如圖 4-3-1。第二部分為同一行（column）沒有數字重複，如圖 4-3-2，以及每一行（column）每一個數字 1 到數字 9 至少出現一次，如圖 4-3-3。第三部分為同一列（row）沒有數字重複，如圖 4-3-)，以及每一列（row）每一個數字數字 1 到數字 9 至少出現一次，如圖 4-3-5。第四部分為每一個九宮格每一個數字 1 到數字 9 至少出現一次，如圖 4-3-6。第五部分為每一個單元格（entry）只有一個值，如圖 4-3-7，以及每一個單元格（entry）為數字 1 到數字 9，如圖 4-3-8。



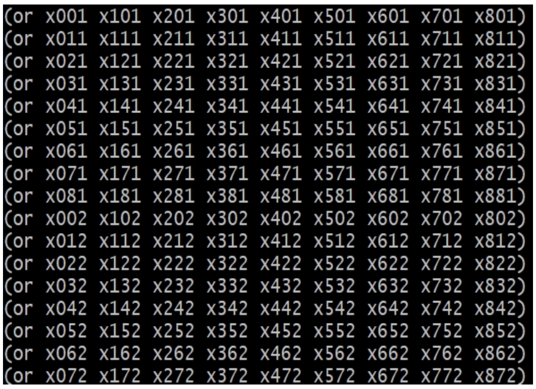
```
* CONSTRAINT ENFORCEMENT: At the beginning, the cell has a value .
(assert (= a11 5)
(assert (= a12 3)
(assert (= a15 7)
(assert (= a21 6)
(assert (= a24 1)
(assert (= a25 9)
(assert (= a26 5)
(assert (= a32 9)
(assert (= a33 8)
(assert (= a38 6)
(assert (= a41 8)
(assert (= a45 6)
(assert (= a49 3)
(assert (= a51 4)
(assert (= a54 8)
(assert (= a56 3)
(assert (= a59 1)
(assert (= a61 7)
(assert (= a65 2)
(assert (= a69 6)
(assert (= a72 6)
(assert (= a77 2)
(assert (= a78 8)
(assert (= a84 4)
(assert (= a85 1)
(assert (= a86 9)
(assert (= a89 5)
(assert (= a95 8)
(assert (= a98 7)
(assert (= a99 9)
```

圖 4-3-1 給定單元格，並賦予它給定值



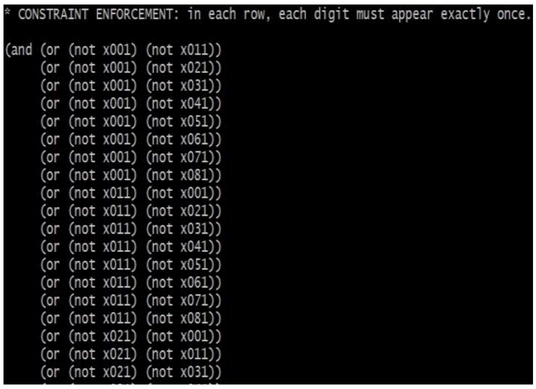
```
* CONSTRAINT ENFORCEMENT: in each column, each digit must appear exactly once.

(and (or (not x001) (not x101))
     (or (not x001) (not x201))
     (or (not x001) (not x301))
     (or (not x001) (not x401))
     (or (not x001) (not x501))
     (or (not x001) (not x601))
     (or (not x001) (not x701))
     (or (not x001) (not x801))
     (or (not x101) (not x001))
     (or (not x101) (not x201))
     (or (not x101) (not x301))
     (or (not x101) (not x401))
     (or (not x101) (not x501))
     (or (not x101) (not x601))
     (or (not x101) (not x701))
     (or (not x101) (not x801))
     (or (not x201) (not x001))
     (or (not x201) (not x101))
     (or (not x201) (not x301))
     (or (not x201) (not x401))
     (or (not x201) (not x501))
     (or (not x201) (not x601))
     (or (not x201) (not x701))
     (or (not x201) (not x801))
     (or (not x301) (not x001))
     (or (not x301) (not x101))
     (or (not x301) (not x201))
     (or (not x301) (not x401))
```

圖 4-3-2 同一行(column)沒有數字重複



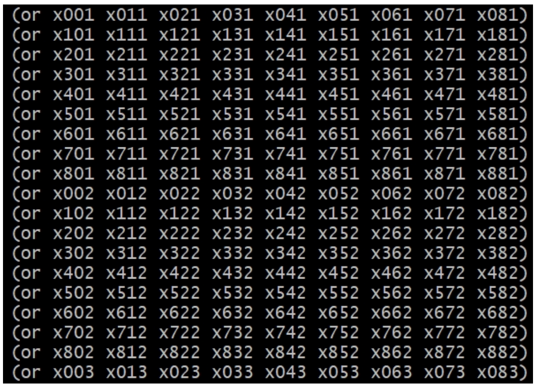
```
(or x001 x101 x201 x301 x401 x501 x601 x701 x801)
(or x011 x111 x211 x311 x411 x511 x611 x711 x811)
(or x021 x121 x221 x321 x421 x521 x621 x721 x821)
(or x031 x131 x231 x331 x431 x531 x631 x731 x831)
(or x041 x141 x241 x341 x441 x541 x641 x741 x841)
(or x051 x151 x251 x351 x451 x551 x651 x751 x851)
(or x061 x161 x261 x361 x461 x561 x661 x761 x861)
(or x071 x171 x271 x371 x471 x571 x671 x771 x871)
(or x081 x181 x281 x381 x481 x581 x681 x781 x881)
(or x002 x102 x202 x302 x402 x502 x602 x702 x802)
(or x012 x112 x212 x312 x412 x512 x612 x712 x812)
(or x022 x122 x222 x322 x422 x522 x622 x722 x822)
(or x032 x132 x232 x332 x432 x532 x632 x732 x832)
(or x042 x142 x242 x342 x442 x542 x642 x742 x842)
(or x052 x152 x252 x352 x452 x552 x652 x752 x852)
(or x062 x162 x262 x362 x462 x562 x662 x762 x862)
(or x072 x172 x272 x372 x472 x572 x672 x772 x872)
```

圖 4-3-3 每行每個數字至少出現一次



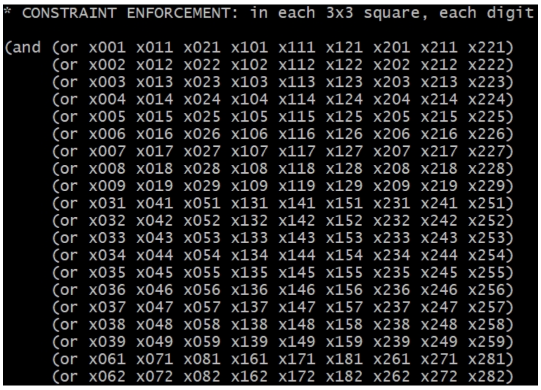
```
* CONSTRAINT ENFORCEMENT: in each row, each digit must appear exactly once.

(and (or (not x001) (not x011))
     (or (not x001) (not x021))
     (or (not x001) (not x031))
     (or (not x001) (not x041))
     (or (not x001) (not x051))
     (or (not x001) (not x061))
     (or (not x001) (not x071))
     (or (not x001) (not x081))
     (or (not x011) (not x001))
     (or (not x011) (not x021))
     (or (not x011) (not x031))
     (or (not x011) (not x041))
     (or (not x011) (not x051))
     (or (not x011) (not x061))
     (or (not x011) (not x071))
     (or (not x011) (not x081))
     (or (not x021) (not x001))
     (or (not x021) (not x011))
     (or (not x021) (not x031))
```

圖 4-3-4 同一列沒有數字重複



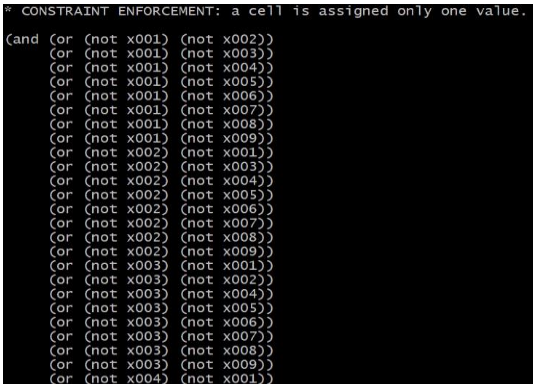
```
(or x001 x011 x021 x031 x041 x051 x061 x071 x081)
(or x101 x111 x121 x131 x141 x151 x161 x171 x181)
(or x201 x211 x221 x231 x241 x251 x261 x271 x281)
(or x301 x311 x321 x331 x341 x351 x361 x371 x381)
(or x401 x411 x421 x431 x441 x451 x461 x471 x481)
(or x501 x511 x521 x531 x541 x551 x561 x571 x581)
(or x601 x611 x621 x631 x641 x651 x661 x671 x681)
(or x701 x711 x721 x731 x741 x751 x761 x771 x781)
(or x801 x811 x821 x831 x841 x851 x861 x871 x881)
(or x002 x012 x022 x032 x042 x052 x062 x072 x082)
(or x102 x112 x122 x132 x142 x152 x162 x172 x182)
(or x202 x212 x222 x232 x242 x252 x262 x272 x282)
(or x302 x312 x322 x332 x342 x352 x362 x372 x382)
(or x402 x412 x422 x432 x442 x452 x462 x472 x482)
(or x502 x512 x522 x532 x542 x552 x562 x572 x582)
(or x602 x612 x622 x632 x642 x652 x662 x672 x682)
(or x702 x712 x722 x732 x742 x752 x762 x772 x782)
(or x802 x812 x822 x832 x842 x852 x862 x872 x882)
(or x003 x013 x023 x033 x043 x053 x063 x073 x083)
```

圖 4-3-5 每列每個數字至少出現一次



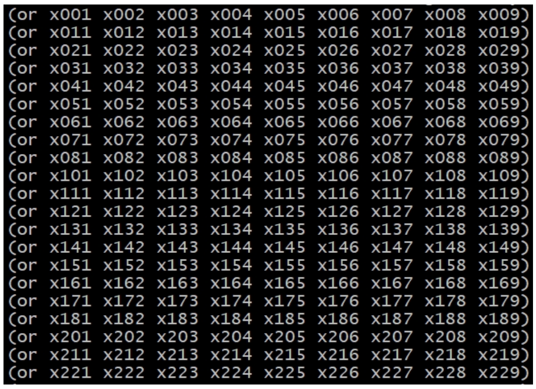
```
* CONSTRAINT ENFORCEMENT: in each 3x3 square, each digit

(and (or x001 x011 x021 x101 x111 x121 x201 x211 x221)
     (or x002 x012 x022 x102 x112 x122 x202 x212 x222)
     (or x003 x013 x023 x103 x113 x123 x203 x213 x223)
     (or x004 x014 x024 x104 x114 x124 x204 x214 x224)
     (or x005 x015 x025 x105 x115 x125 x205 x215 x225)
     (or x006 x016 x026 x106 x116 x126 x206 x216 x226)
     (or x007 x017 x027 x107 x117 x127 x207 x217 x227)
     (or x008 x018 x028 x108 x118 x128 x208 x218 x228)
     (or x009 x019 x029 x109 x119 x129 x209 x219 x229)
     (or x031 x041 x051 x131 x141 x151 x231 x241 x251)
     (or x032 x042 x052 x132 x142 x152 x232 x242 x252)
     (or x033 x043 x053 x133 x143 x153 x233 x243 x253)
     (or x034 x044 x054 x134 x144 x154 x234 x244 x254)
     (or x035 x045 x055 x135 x145 x155 x235 x245 x255)
     (or x036 x046 x056 x136 x146 x156 x236 x246 x256)
     (or x037 x047 x057 x137 x147 x157 x237 x247 x257)
     (or x038 x048 x058 x138 x148 x158 x238 x248 x258)
     (or x039 x049 x059 x139 x149 x159 x239 x249 x259)
     (or x061 x071 x081 x161 x171 x181 x261 x271 x281)
     (or x062 x072 x082 x162 x172 x182 x262 x272 x282)
```

圖 4-3-6 每九宮格每數字至少出現一次



```
* CONSTRAINT ENFORCEMENT: a cell is assigned only one value.

(and (or (not x001) (not x002))
     (or (not x001) (not x003))
     (or (not x001) (not x004))
     (or (not x001) (not x005))
     (or (not x001) (not x006))
     (or (not x001) (not x007))
     (or (not x001) (not x008))
     (or (not x001) (not x009))
     (or (not x002) (not x001))
     (or (not x002) (not x003))
     (or (not x002) (not x004))
     (or (not x002) (not x005))
     (or (not x002) (not x006))
     (or (not x002) (not x007))
     (or (not x002) (not x008))
     (or (not x002) (not x009))
     (or (not x003) (not x001))
     (or (not x003) (not x002))
     (or (not x003) (not x004))
     (or (not x003) (not x005))
     (or (not x003) (not x006))
     (or (not x003) (not x007))
     (or (not x003) (not x008))
     (or (not x003) (not x009))
     (or (not x004) (not x001))
```

圖 4-3-7 每一個單元格只有一個值



```
(or x001 x002 x003 x004 x005 x006 x007 x008 x009)
(or x011 x012 x013 x014 x015 x016 x017 x018 x019)
(or x021 x022 x023 x024 x025 x026 x027 x028 x029)
(or x031 x032 x033 x034 x035 x036 x037 x038 x039)
(or x041 x042 x043 x044 x045 x046 x047 x048 x049)
(or x051 x052 x053 x054 x055 x056 x057 x058 x059)
(or x061 x062 x063 x064 x065 x066 x067 x068 x069)
(or x071 x072 x073 x074 x075 x076 x077 x078 x079)
(or x081 x082 x083 x084 x085 x086 x087 x088 x089)
(or x101 x102 x103 x104 x105 x106 x107 x108 x109)
(or x111 x112 x113 x114 x115 x116 x117 x118 x119)
(or x121 x122 x123 x124 x125 x126 x127 x128 x129)
(or x131 x132 x133 x134 x135 x136 x137 x138 x139)
(or x141 x142 x143 x144 x145 x146 x147 x148 x149)
(or x151 x152 x153 x154 x155 x156 x157 x158 x159)
(or x161 x162 x163 x164 x165 x166 x167 x168 x169)
(or x171 x172 x173 x174 x175 x176 x177 x178 x179)
(or x181 x182 x183 x184 x185 x186 x187 x188 x189)
(or x201 x202 x203 x204 x205 x206 x207 x208 x209)
(or x211 x212 x213 x214 x215 x216 x217 x218 x219)
(or x221 x222 x223 x224 x225 x226 x227 x228 x229)
```

圖 4-3-8 每一個單元格為數字 1 到 9

4-4 SMT 的限制條件

SMT 的限制條件總共分為五個部分，第一部分的限制條件為初始給數獨的值，如圖(4-4-1)。第二部分的限制條件為每個 cell 的值皆要在 1-9 當中，如圖(4-4-2)。第三部分的限制條件為 1-9 的值皆要在每一個 row 出現一次，如圖(4-4-3)。第四部分的限制條件為 1-9 的值皆要在每個 column 出現一次，如圖(4-4-4)。第五部分的限制條件為 1-9 的值皆要在每個 square 出現一次，如圖(4-4-5)。

```
* CONSTRAINT ENFORCEMENT: At the beginning, the cell has a value .
(assert (= a11 5)
(assert (= a12 3)
(assert (= a15 7)
(assert (= a21 6)
(assert (= a24 1)
(assert (= a25 9)
(assert (= a26 5)
(assert (= a32 9)
(assert (= a33 8)
(assert (= a38 6)
(assert (= a41 8)
(assert (= a45 6)
(assert (= a49 3)
(assert (= a51 4)
(assert (= a54 8)
(assert (= a56 3)
(assert (= a59 1)
(assert (= a61 7)
(assert (= a65 2)
(assert (= a69 6)
(assert (= a72 6)
(assert (= a77 2)
(assert (= a78 8)
(assert (= a84 4)
(assert (= a85 1)
(assert (= a86 9)
(assert (= a89 5)
(assert (= a95 8)
(assert (= a98 7)
(assert (= a99 9)
```

圖 4-4-1 限制條件為數獨的初始值

```
* CONSTRAINT ENFORCEMENT: The value of each cell must be 1~9 .
(assert (and (> a11 0) (< a11 10))
(assert (and (> a12 0) (< a12 10))
(assert (and (> a13 0) (< a13 10))
(assert (and (> a14 0) (< a14 10))
(assert (and (> a15 0) (< a15 10))
(assert (and (> a16 0) (< a16 10))
(assert (and (> a17 0) (< a17 10))
(assert (and (> a18 0) (< a18 10))
(assert (and (> a19 0) (< a19 10))
(assert (and (> a21 0) (< a21 10))
(assert (and (> a22 0) (< a22 10))
(assert (and (> a23 0) (< a23 10))
(assert (and (> a24 0) (< a24 10))
(assert (and (> a25 0) (< a25 10))
(assert (and (> a26 0) (< a26 10))
(assert (and (> a27 0) (< a27 10))
(assert (and (> a28 0) (< a28 10))
(assert (and (> a29 0) (< a29 10))
(assert (and (> a31 0) (< a31 10))
(assert (and (> a32 0) (< a32 10))
(assert (and (> a33 0) (< a33 10))
(assert (and (> a34 0) (< a34 10))
(assert (and (> a35 0) (< a35 10))
(assert (and (> a36 0) (< a36 10))
```

圖 4-4-2 限制條件為每個 cell 的值皆要在 1-9

```
* CONSTRAINT ENFORCEMENT: in each row, each digit must appear exactly once.
TOTAL CONSTRAINT =
 (and (distinct a11 a12 a13 a14 a15 a16 a17 a18 a19)
     (distinct a21 a22 a23 a24 a25 a26 a27 a28 a29)
     (distinct a31 a32 a33 a34 a35 a36 a37 a38 a39)
     (distinct a41 a42 a43 a44 a45 a46 a47 a48 a49)
     (distinct a51 a52 a53 a54 a55 a56 a57 a58 a59)
     (distinct a61 a62 a63 a64 a65 a66 a67 a68 a69)
     (distinct a71 a72 a73 a74 a75 a76 a77 a78 a79)
     (distinct a81 a82 a83 a84 a85 a86 a87 a88 a89)
     (distinct a91 a92 a93 a94 a95 a96 a97 a98 a99))
```

圖 4-4-3 限制條件為 1-9 的值皆要在每個 row 出現一次

```
* CONSTRAINT ENFORCEMENT: in each column, each digit must appear exactly once.
TOTAL CONSTRAINT =
 (and (distinct a11 a21 a31 a41 a51 a61 a71 a81 a91)
     (distinct a12 a22 a32 a42 a52 a62 a72 a82 a92)
     (distinct a13 a23 a33 a43 a53 a63 a73 a83 a93)
     (distinct a14 a24 a34 a44 a54 a64 a74 a84 a94)
     (distinct a15 a25 a35 a45 a55 a65 a75 a85 a95)
     (distinct a16 a26 a36 a46 a56 a66 a76 a86 a96)
     (distinct a17 a27 a37 a47 a57 a67 a77 a87 a97)
     (distinct a18 a28 a38 a48 a58 a68 a78 a88 a98)
     (distinct a19 a29 a39 a49 a59 a69 a79 a89 a99))
```

圖 4-4-4 限制條件為 1-9 的值皆要在每個 column 出現一次

```
* CONSTRAINT ENFORCEMENT: in each 3x3 square, each digit must appear once.
TOTAL CONSTRAINT =
 (and (distinct a11 a12 a13 a21 a22 a23 a31 a32 a33)
     (distinct a14 a15 a16 a24 a25 a26 a34 a35 a36)
     (distinct a17 a18 a19 a27 a28 a29 a37 a38 a39)
     (distinct a41 a42 a43 a51 a52 a53 a61 a62 a63)
     (distinct a44 a45 a46 a54 a55 a56 a64 a65 a66)
     (distinct a47 a48 a49 a57 a58 a59 a67 a68 a69)
     (distinct a71 a72 a73 a81 a82 a83 a91 a92 a93)
     (distinct a74 a75 a76 a84 a85 a86 a94 a95 a96)
     (distinct a77 a78 a79 a87 a88 a89 a97 a98 a99))
```

圖 4-4-5 限制條件為 1-9 的值皆要在每個 square 出現一次

4-5 求解器的效能比較

總結來說，我們可以發現到求解數獨問題時，SAT 求解法比較不會受到輸入難易度的影響，SMT 以及回測法，受到問題複雜度的影響較大，只能說各有各的優勢。

表 4-5-1 求解器的效能比較（簡單題）

	回測求解器	SAT 求解器	SMT 求解器
產生限制的時間		643.0089(ms)	13.4789(ms)
求解的時間	1.4581(ms)	68.3863(ms)	60.1699(ms)
總時間	1.9854ms)	719.1925(ms)	82.5657(ms)
限制的數量		17820	108

表 4-5-2 求解器的效能比較（困難題）

	回測求解器	SAT 求解器	SMT 求解器
產生限制的時間		636.4202(ms)	13.4490(ms)
求解的時間	95.9016(ms)	63.1917(ms)	228.9840(ms)
總時間	96.4829ms)	708.2504 (ms)	250.1779(ms)

至於各自的優勢,我們認為回測法可以最快求解簡單問題,如果已知的數字愈多,效果會愈明顯；而 SAT Solver 有更穩定的解題時間，對於任意問題都能有不錯的效果，缺點就是建模較久；SMT Solver 的優點就是建模速度較快，但因為運算較為複雜，解題通常比較慢。

第五章 問題與討論

5-1 實作過程中遇到的問題

這次的專案因為是之前比較沒有實作過的內容,所以在過程中遇到了比較多困難,除了一些在實作上本來就會遇到的程式撰寫問題之外,還遇到了很多函式庫引入、套件安裝以及指令不熟等等的運行問題，以下將會一一進行說明。

5-1-1 Microsoft Z3 安裝問題

在開始 project 之前，遇到最大的困難是 Github 上的 Z3 無法成功安裝，即使按照微軟（Microsoft）開源的網站步驟下指令仍然會遇到各式各樣的錯誤，因此我們花了一段時間上網，不論是在各大論壇，或是 Stack Overflow 上，查詢各種錯誤的解決辦法，最後得出結論是需要先在 Cygwin 裡載 Python3.9 以及因為 Cygwin 並不支援 Z3 Solver 以.so 檔撰寫的動態函式庫，所以後來我們找到網路上，功能相同的動態函式庫(.dll)來取代.so 檔。

5-1-2 Microsoft Z3 函式使用問題

因為是第一次使用 SMT/SAT Solver，很多的指令該怎麼使用、Z3 Solver 是怎麼吃那些條件限制或是要構建那些限制需要那些函式，完全沒有任何概念，最大的困難就在於 Z3 有一大堆自己定義的函式，因此我們必須了解每個函式的意思以及使用方式，因為每個函式引入變數的資料型別都不是平常我們使用的,例如我們希望產生一條 constraint 是 a < 10，使用了 Z3_mk_lt 生出一條小於的不等式，然而它吃的變數卻非 int 10 而是需要吃另外一個函式才能表示 10 這個數字，為此我們花了很多時間上網查資料以及瀏覽 Z3 的 API 應用介面的介紹。

參考文獻

1. Lynce, I., Ouaknine, J.: Sudoku as a SAT problem. In: Proceedings of the 9th Symposium on Artificial Intelligence and Mathematics (2006)

2. C.P. Gomez and D. Shmoys, ‘Completing quasigroups or latin squares: a structured graph coloring problem’, in Proceedings of Computational Symposium on Graph Coloring and Generalization, (2002)

3. Banković, M., & Marić, F.: An Alldifferent constraint solver in SMT. In Proceedings of the 8th international workshop on satisfiability modulo theories (2010)

4. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), vol. 4963, pp. 337–340. Springer, Heidelberg (2008)