利用基因演算法產生表示影像之補片之方法

梁恩輝
淡江大學資管系副教授
enliang@mail.im.tku.edu.tw

林詠翔
淡江大學資管系
steven.lin@mail.im.tku.edu.tw

摘要

在許多資訊領域當中，如多媒體系統、影像處理等，影像資料總是大且繁瑣，但在影像資料中，常常包含了許多不重要的資訊，而事實上，只有重要的資訊才是我們有興趣的。然而在本研究當中，我們使用一組利用基因演算法產生較佳的補片(patches)來表示、索引出影像資料中較重要的部份。而使用了這些patch來表示影像，可以降低使用的儲存空間，以及增加處理影像資訊的效率。

Abstract

In many areas of computer applications, such as MultiMedia Systems and Image Information Systems , image data is always huge and complicated. However, it includes some information which is not important. In fact, we are only interested in the information of importance. In this paper, we use the genetic algorithm to find a set of patches to represent and index the important part of an image. By using the set of patches to represent the image, the storage required can be reduced and the efficiency of processing can be increased.

關鍵字 : Patch、Genetic Algorithm

第一章　前言

在諸多探討建置演算法的研究中，可發現多數的研究只針對特定要素設計演算法，難以應付眾多的可能性，於面對特殊情況時，可能獲得不理想的結果。在許多情況中可利用基因演算法[1][2]「尋求最佳解」的能力，獲得一較佳的結果，有其使用上之價值。

Patching是一種表示影像中重要資訊的方法，其目的在於產生一組patches來表示一張影像。其主要缺點為patch相互重疊過多，不易取得最佳解。本研究假設影像中之物件為我們有興趣以及重要的資訊，我們利用基因演算法，找出蓋滿圖片中物件最佳的一組patches配置方式。

第二章　相關研究

2.1 補片(Patching)

在很多相關影像處理及多媒體的議題當中，所需要處理的影像資料總是大且繁瑣，並且在影像或者圖片的處理當中，其所要處理的圖片中物件雖然重要，但卻有可能只是佔整張圖片的小部分。因此如何有效的表示這些重要的資訊，而省略掉不重要之資訊，是在此影像處理及多媒體的研究中之一重要的課題。

如在圖 1中真正所需要被處理、表示或者索引的影像，只有一長方形和圓形，但整張圖片卻有很多其他不

重要的部分，也和影像中重要的部份(長方形、圓形)，一起被考慮，也因此處理時可能會增加更多的時間複雜度。除此之外在資料庫的運用當中，儲存的影像資料，也是愈小愈好，愈小愈不浪費儲存空間；尤其日後在資料庫當中，欲搜尋影片或者在建立索引的同時，有需要利用到影像處理的部份，則會浪費更多的時間與效率。

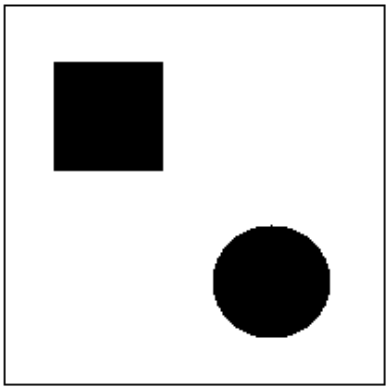


圖 1.影像處理圖中的物件

為了有效率的解決上述的問題，因此更應該將電腦運算及處理著重於圖片中較重要的部份，也因為要解決這個問題，在過去的研究中採用了「Patching」這個方法[3]， patch為一大小適中的長方形，並且將原圖中重要的影像資料藉由一組patches表示，也可說是利用一組patches表示出圖片中的物件，如圖 2所示。也因此日後在建立圖片的索引或者其他處理等，能更加有效率的運算，除此之外patch能夠愈少，電腦處理的patch也愈少，因此能夠處理的更有效率。

patch 配置的方式有兩種，一種為配置 patch 的同時「各個 patch 不可以可互相重疊」(此種方式又稱為 tiling)，如圖 3 所示，另外一種方式為配置 patch 的時候「各個 patch 可以互相重疊」，如圖 4 所示。

在本研究中所採行的方法，為「各個patch可以互相重疊」的配置，而此方法較tiling更有彈性及優點，例如圖 3是tiling配置patch的結果，很明顯的我們可以觀察出此張圖所需要使用4個patch才能蓋滿，而「各個patch可以互相重疊」配置patch的結果如圖 4，只需要3個patch就能蓋滿，也因此本研究中所討論的方式為可重疊的patch配置方式。

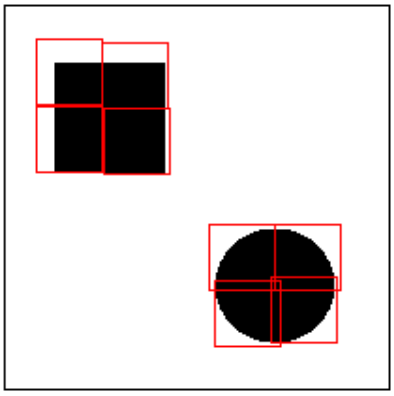


圖 2.利用一組 patch 來表示圖中的物件

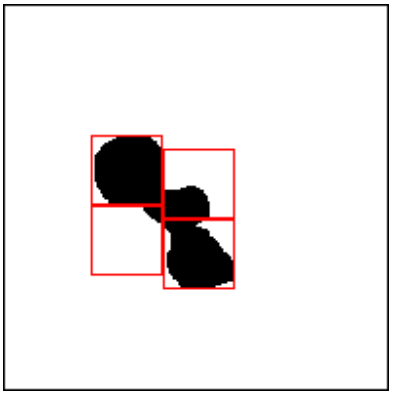


圖 3. tiling 配置 patch 的結果

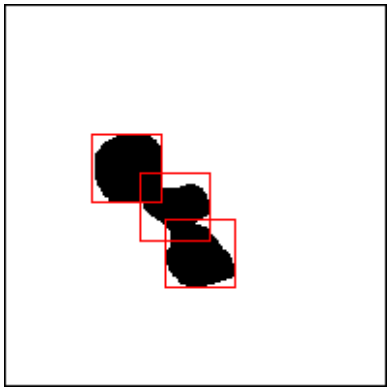


圖 4.patch 可重疊配置的結果

2.2 基因演算法

2.2.1 基因演算法介紹

基因演算法則的應用是由Holland於1960 年首先提倡[4]，而基因演算法是達爾文（Darwin）經過多年的研究與觀察於1859年出版「物種原始」一書中所提出的演化思想，提出「物競天擇」的理論，亦即自然界的生物是以「適者生存，不適者淘汰」的規則來進化的。適合目前生存環境的物種擁有較大的機會生存下來，而不適合的物種會被自然環境所淘汰，這種方式就是「天擇」。經過天擇後的生物繼續繁衍，使得子代遺傳親代的優勢更加適應環境的變化，以期在環境中得以生存。如此代代演化下去，整個物種就會朝著更適應生存環境的方向發展。

基因演算法是利用遺傳學上的生存法則－物競天擇適者生存的原理，運用電腦運算模擬所發展出來的一種搜尋法。依照物種生存的原理，由電腦程式決定一個初始族群，模擬物種的演化與篩選作用，並經由複製、交換及突變等運算子，在交互作用下產生優良個體，如此反覆，最後留下的必然是最適合在目前環境中生存的個體。

2.2.2 基因演算法的基本架構

基因演算法是較為隨機性的搜尋法，與一般傳統的搜尋法不同，傳統的最佳化方法例如非線性規劃法起始於一初始設計，而基因演算法為一群由電腦隨機產生的初始設計值，稱為族群(Population)，而族群中的每一組設計值則稱為個體(Individual)，每一個體均有代表其特性的染色體(Chromosome)而染色體則是遺傳因子即所謂基因(Gene)串聯組成，代表了對於某個最佳化問題的解[2][5]。

基因演算法是遵循了自然界有性生殖的法則所推演出來的，其中包含染色體中基因的架構方式，以及選擇、複製、交換、突變、產生新的基因個體組合等操作機制。

2.2.3 基因演算法主要運算過程

基因演算法包含下列主要運算過程[1][2][6]：(1)決定初始設定，(2)編碼與解碼，(3)選擇與複製，(4)基因交換，(5)基因突變等過程。以下簡述各過程意義：

(1)決定初始設定

相對於傳統最佳化方法一次只能處理一組設計變數，基因演算法一次能處理許多組的設計變數。在進行演算法之前必須先由使用者設定族群的個數與設計變數所需之基因數，經由電腦以隨機變數的方式產生初始族群(Initial Population)的染色體，或經由使用者給定設計好的參數組合，再經程式解碼成二進制的字串組合成染色體，以利複製、交換與突變等運算子的操作使用。

(2)編碼與解碼

以基因演算法解最佳化問題時，需先將設計變數編碼，使之成為二進制串列的表示法，以利複製，交換與突變等運算子使用。而在演化結束後，再將原先編碼的串列加以解碼，還原成原設計變數，以求出族群中每一個個體的「適存度」參與演化的進行。而在每一代演化之後，亦需對新產生的個體編碼與解碼並計算個體的「適存度」，以求

出演化完成的最佳值。

(3)選擇與複製

自從於1989年Goldberg奠定基因演算法的基礎後，基因演算法經過多年來諸位學者的研究，在運算子的處理上發展出許多改良的方法，使能更快速的有效的收斂。在基因演算法中，選擇與複製的用意為：將適存度較差的個體淘汰，把適存度高的個體與以保留並且透過複製、交換與基因突變等作用之下，產生適存度較高的新個體，並逐步取代適存度較差的個體。所以，優良的個體會被保存下來，且數量越來越多，而族群的整體素質也會跟著提高。

(4)基因交換

將依所選擇與複製的兩染色體，以模擬自然界生物有性生殖的交換現象，交換彼此基因字串組的其中部分基因，以產生新的個體。其目的是希望在交換部分的基因後，新產生的染色體能有較好的適應值。

(5)基因突變

自然界的生物，有時會突變一小部份的基因，藉由此一突變機制來增加族群成員對所生存環境產生極鉅變化時的適應能力。而遵此一生存原理，基因演算法中的突變主要是在當搜尋過程中， 對某些設計空間的忽略做出彌補的措施。其作法為將染色體上的基因以隨機選取的方式對位元做更改，例如將某依基因上的位元由原本的1變為0，或由0變為1，以產生新的設計變數。

第三章　以基因演算法產生補片

本研究利用基因演算法尋求最佳解的特性對一影像，產生出一組表示此影像之patches，令此組patches為S，一個優良的S應具有「S之patches應完全蓋滿物件」及「S包含最少patches」兩特性。演算法說明如下。

3.1初始族群(Initial Population)

在基因演算法中，初始族群由許多第一代的個體組成，每一個個體皆為一組染色體，代表了最佳化問題的其中一個解決方案，亦即「解(Solution)」，且每一道染色體皆由許多基因組成。

本研究中以基因代表patch，是一個被預設為大小適當的長方形，我們可依照patch的中心點，建造出圖中其正確的所在位置，因此我們將每個patch的中心點座標(x,y)，設定為染色體中的基因，意即每條染色體中所代表的就為所有patch的位置。

欲建立初始族群時，本研究以隨機方式產生M組染色體，每條染色體皆為一組patches，我們首先說明如何訂定染色體的長度。在一般情況，利用patches表示影像直接之方法為，以patch的大小切割原圖，如圖 5所示，需要N = 23個patches才能完全表示完整個物件。因此我們要尋求一個利用少於N個patches，且能夠完全表示出物件之最佳解。但又因為一個patch對應一組座標(x , y)共有兩個值要存入，所以我們訂定染色體的結構如圖 6所示，而其長度為2N。

在解譯階段，第 1 至第 2N 個基因所代表的，就是第一個 patch 到第 N 個 patch 的座標，如圖 6 所示，第一個基因為 patch 1 的 x 軸座標, 而第二個基因則為 patch 1 的 y 軸座標，第三個基因為 patch2 的 x 軸座標，第四個基因為 patch2

的 y 軸座標，依序到類推到最後一個 patch 為 patch N 其 x 軸座標為第 2N-1 個基因，y 軸座標為第 2N 個基因。

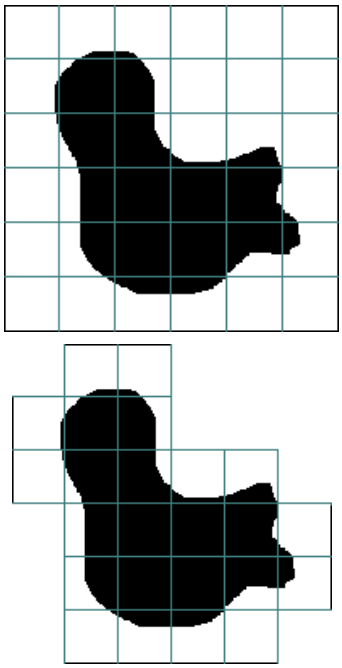


圖 5. 利用patch切割原圖之範例

3.2 適應函式(Fitness Function)

本研究在找尋一組最佳的patches S以表示物件，而S需滿足下列兩條件「S之patches應完全蓋滿物件」及「S包含最少patches」兩條件為基礎來考慮在基因演算法內所採取的適應函式。並且我們先根據這兩條件產生出三個子適應值，分別為適應值一、適應值二、適應值三，而每個染色體之適應值為此三值之和，本研究是以此值愈高表示解愈差，反之值愈低則解愈好。

適應值(Fitness Value)一乃根據「S之patches應完全蓋滿物件」之條件計算，因此，一個染色體之適應值一 = (染色體內未被此染色體中之patches覆蓋到的物件面積)/(物件總面積)，未被蓋到的面積愈大則其值愈大。

至於「S包含最少patches」，此項條件較為複雜，不能直接使用「染色體中和物

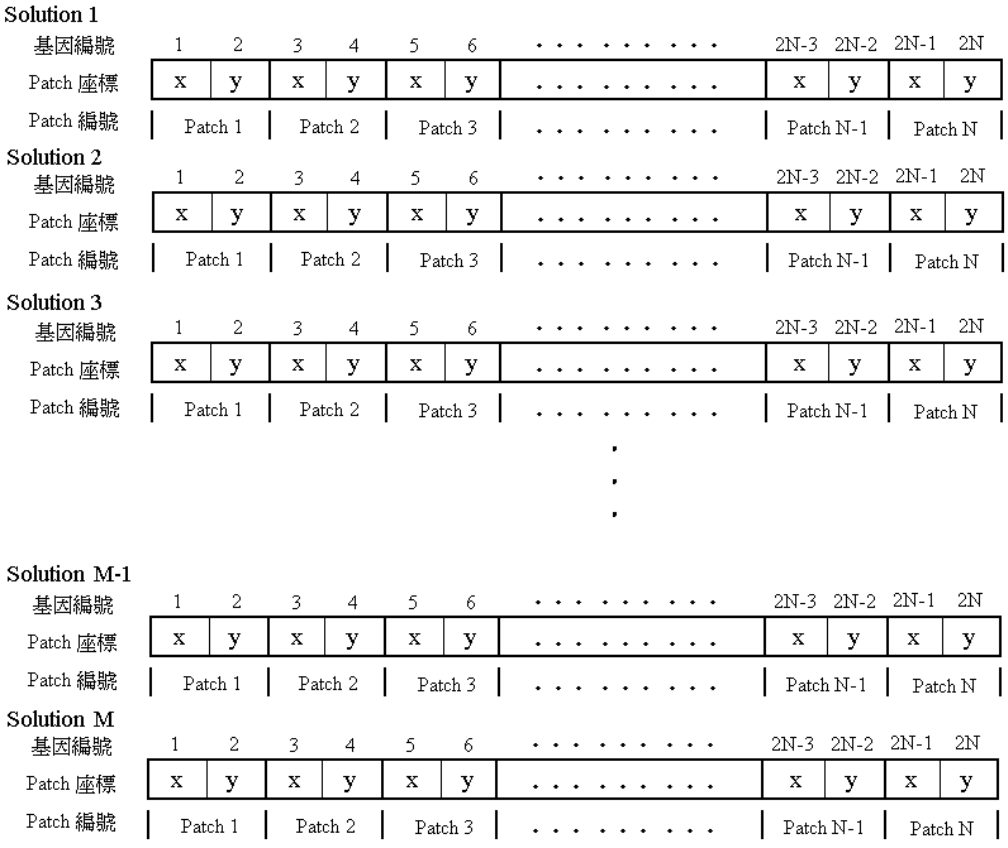


圖 6.染色體的結構

件重疊的patches個數」為計算適應值的方式；其原因在若以此計算適應值，最好的解則為0個patch，因為0個patch為覆蓋物件最少的patch數，而此解表示所有patches皆和物件不重疊，因為我們希望patch要能完全蓋滿物件，由此可知不能直接將「染色體中和物件重疊的patches個數」加入我們的適應值計算方法。

因此本研究中在求得「S包含最少patches」的條件時，以考慮下列兩個因素來取代：(1)「染色體中patch和其他patch在物件區域內重複的情形」、(2)「染色體中patch落在物件外的面積」。

而適應值二乃是根據「染色體中patch和其他patch在物件區域內重複的情形」所產生，一條染色體中包含了N個patches，我們只考慮和物件重疊的patches，假設此染色體有w個和物件有重疊的patches，令其為P_1、P_2 … P_w，我們定義此染色體的適應值二為：

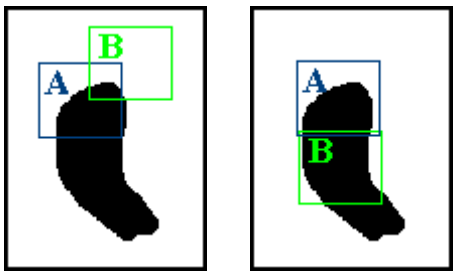
$$\sum_{i=1}^{w}\left[\sum_{\substack{j=1\\ \text{and } j\neq i}}^{w}\frac{\text{在物件區域內}P_i\text{和}P_j\text{重疊的面積}}{P_i\text{和物件重疊之面積}}\right]$$

如圖 7中patch A和patch B的覆蓋情況，我們可以明顯的看出圖 7(a)中，patch A表示物件的部分，有一些和patch B所重複，patch B表示物件的部分，大部分也都和patch A重複，由此我們可以判斷此組patches的適應值二，因patch B加的值非常高，並且patch A也加了重疊patch B的值，所以其解較差。而我們可以看到圖 7(b)，patch A和patch B重複的面積很小，所以patch A加的值小，patch B加的值也很小，這是較好的解。

而適應值三乃根據「染色體中patch落在物件外的面積」所產生，而我們也只考慮和物件重疊有重疊之patches，假設染色體中有w個和物件有重疊之patches，令其為P_1、P_2 … P_w，定義此染色體之適應值三為：

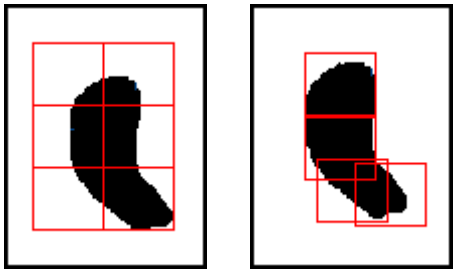
$$\sum_{i=1}^{w}\left[\frac{\text{在物件區域外}P_i\text{的面積}}{\text{Patch的面積}}\right]$$

圖 8(a)為一組patches，此組patches落在物件外的面積卻是比圖 8(b)大的多，並且圖 8(a)需要六個patches才能蓋滿圖片，而圖 8(b)卻只需要四個patches就能蓋滿。



(a) (b)

圖 7.說明適應值二之範例



(a) (b)

圖 8.說明適應值三之範例

適應值的計算在本研究中，乃是考慮三種不同的因素，適應值一、適應值二、適應值三，且三種不同的因素各所需要的權重皆有不同(此權重也為一自訂參數)，適應值一的權重是為權重一，適應值二的權重為權重二，適應值三的權重為權重三，將此三種不同的因素依照不同的權重獲做加總，則為此組染色體的適應值。公

式如下所示：

適應值 = [權重一 * (適應值一) + 權重二 * (適應值二) + 權重三 * (適應值三)]。

3.3 選擇函式(Selection Function)

本研究首先將所有染色體依適應值高低加以排序，保留前半適應值較高的染色體，並淘汰另一半適應值較低的染色體，接著實作Roulette Wheel Selection[6]，令適應值較高的染色體有較大的機會被選出進行交配。

3.4 交配方式(Crossover)

若Chromosome A與Chromosome B為選出欲進行交配的二道染色體，產生後代(Chromosome C與Chromosome D)的運算過程如下：

a. 本研究中採用的交配方式為單點交配，在染色體上隨機產生一個 cross point S，cross point 的選擇以 patch 為單位，因為每兩個基因各代表一個 patch 的(x , y)座標，因而在 Chromosome A 或 Chromosome B 上決定一段 Matching Section，如圖 9 所示。

b. 參照 Chromosome B 之染色體位於 matching section 中的基因，對 Chromosome A 之染色體位於 matching section 中進行基因交換，所得的新染色體即為 Chromosome C 和 Chromosome D。

3.5 突變方式(Mutation)

演算法中加入「突變機制」的主要目的為降低演算法所求得的解為「局部最佳解」的可能性，其發生時機在於交配動作結束後，每道新產生的染色體皆有特定的機率(自訂參數)發生突變。

本研究中突變的方式皆是以一個patch為單位(一個patch為兩個基因)。也就是染色體進行突變的時候，將染色體中隨機取出一patch，將此patch的(x , y)座標位置，隨機的更換到圖中不同的(x , y)座標上。實驗中突變的操作方式，是在染色體中隨機選擇出部分patch(patch個數為自訂參數)，再予以進行突變，突變的方式如下：

將這些選擇出要突變的patch進行「將(x , y)座標位置隨機的更換到此張圖中其他的座標」。以圖 10為例，假若Chromosome A要進行突變，且選擇出的patch個數為1，其發生突變的方式，有可能突變到此張圖隨機的位置上，如圖 10突變的patch為patch 3，所以要替換的基因就為此patch 3的座標(x , y)，也就是Chromosome A的基因編號5與6。

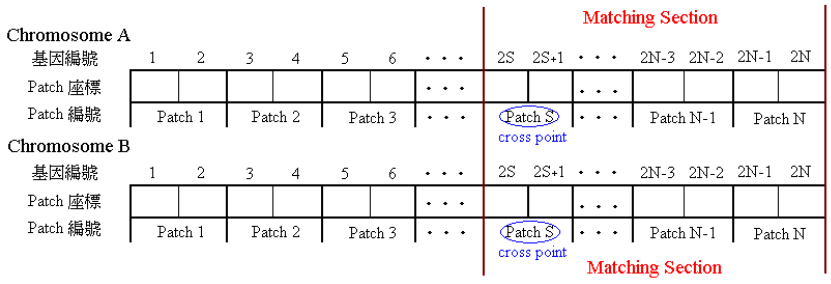


圖 9.單點交配方式

3.6 產生補片之演算法

在演算法執行之前，需先指定各項系統參數，如：Fitness Function中不同因素的比重、Population Size、突變率等。演算法的執行，藉由訂定出的Population Size M而決定初始族群應該產生M個Chromosome，每一個Chromosome都是一個Solution，每一個Solution就為N個patch在此張圖的座標位置，Fitness Function就會計算出每個Chromosome中patch配置方式的值；繼續基因演算法其中的過程，選擇Chromosome，將這些選擇出的Chromosome進行交配，每一個Chromosome都會有一定機率產生突變，在將這些交配過後的Chromosome和原先的Chromosome選出較好的留給下一代，反覆的操作，直到滿足終止條件為止，滿足終止條件後，以最好的一個Chromosome來建置patches。

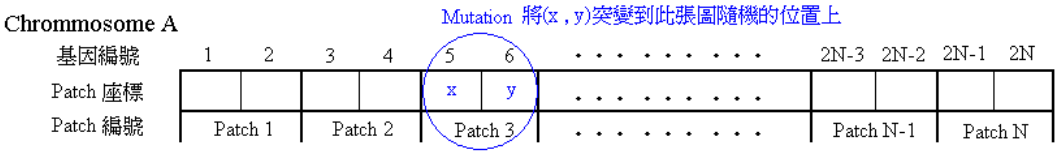


圖 10.突變方式

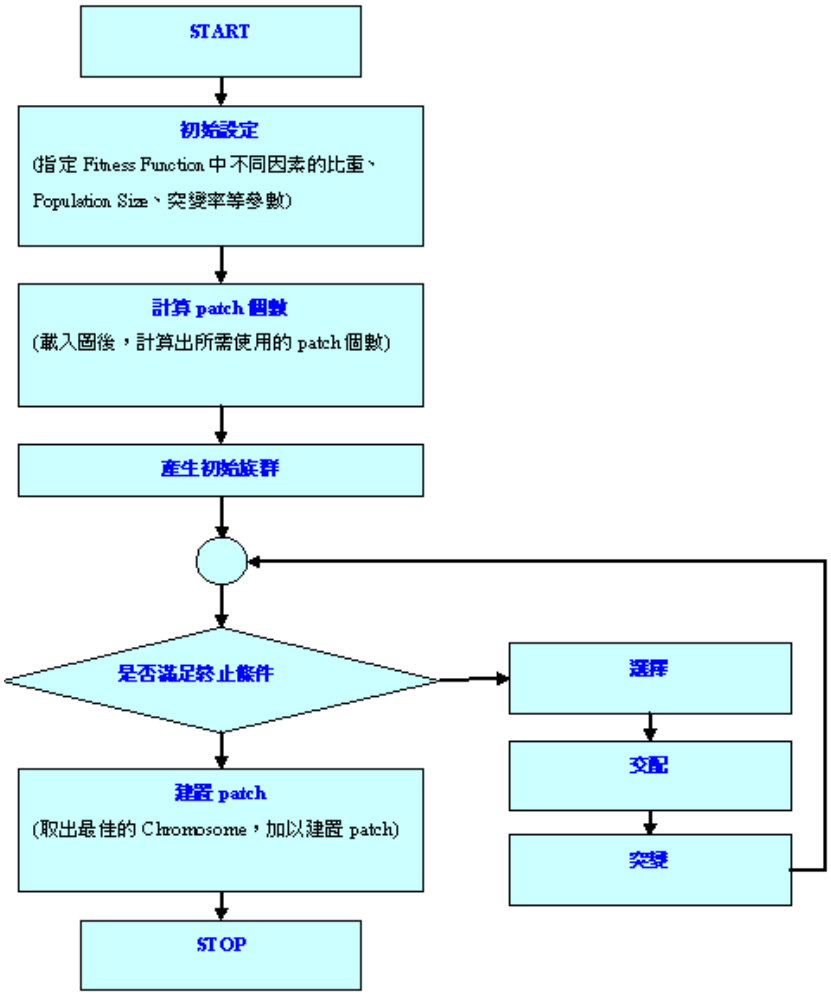


圖 11.演算法執行流程圖

第四章　實驗與分析

表示影像中重要資訊的方法，一般直接做法可將影像切割成一個個大小相同之patch表示，再計算其表示物件之所需個數。我們將利用本研究提出的方法進行實驗，計算其所需之patch個數，和一般直接的做法比較，已說明其效率。由於其他patching之方法重疊部分都太多，效果不佳，因此我們不予以比較。

參數設計上，基因演算法所建置出的population size 為 250 組，突變率為百分之七，適應值共有三個衡量的因素，分別為適應值一、適應值二、適應值三；而此三種因素各有三種不同的權重，權重一設定為 4，權重二設定為 2，權重三設定為 1。

本研究中所採行的測試資料有兩張圖片，在此我們將圖片使用基因演算法做測試，patch所使用的個數也會因為圖形中的物件而不同。在實驗當中，我們的基因演算法一共執行50000代，而其最終的結果為第50000代最佳的Chromosome。

因為每一張圖形皆不同，也因此本研究中載入圖形之後，會利用直接的做法，計算此張圖中的物件，會用到多少個patch，也因此patch的總個數，我們將其訂定為直接做法的個數。如圖 12測試資料一，我們藉由直接做法的方式，將整張圖直接切割成一個個patch，並且這些patch有蓋到物件，所需的patch總個數為23個；圖 13測試資料二，所需要的patch總個數為32個。算出這些patch總個數後，才能訂定出Chromosome的長度，以實行基因演算法的前置動作。

本實驗中使用基因演算法來找尋最佳patch的配置方法，圖 14、圖 15黑色的部分則為物件，紅色的方框為patch，而圖中則是我們所做出的實驗結果。

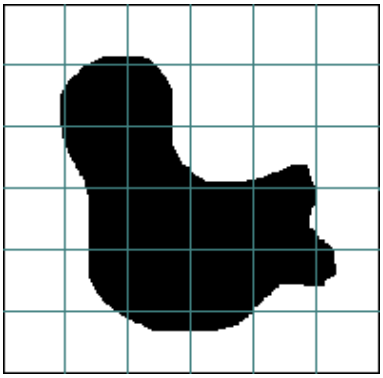


圖 12.測試資料一

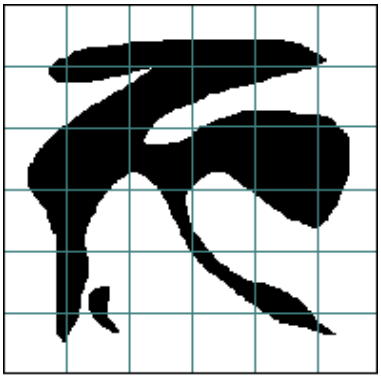


圖 13.測試資料二

在此實驗當中，我們利用基因演算法來減少patch使用的個數，也就是利用其找尋最佳解的能力，找出覆蓋滿此圖中物件的最佳patch配置方法。由上列測試資料一的圖中顯示出，原先我們利用直接表示的方式找出，如圖 12中所示，需要23個patches才能完全覆蓋圖中的物件，我們藉由基因演算法所尋求最佳解的結果，如圖 14所示，我們則需要16個patches就可以完全覆蓋圖中的物件。並且我們也在測試資料二的圖中發現，原先我們用直接表示的方式，如圖 13所示，需要用32個patches才能完全覆蓋圖中的物件，而我們藉由基因演算法所尋求最佳解的結果，如圖 15所示，則只需要21個patches就能完全覆蓋圖中的物件。

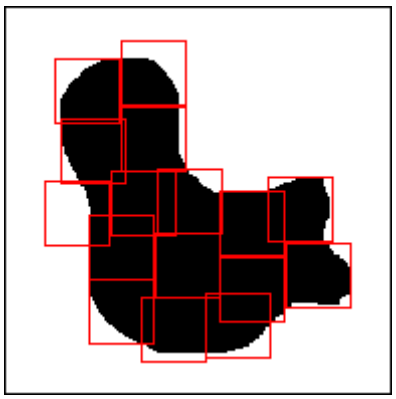


圖 14.測試資料一
由基因演算法配置 patches 的結果

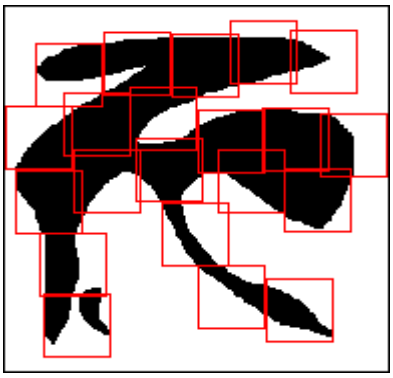


圖 15. 測試資料二
由基因演算法配置 patches 的結果

第五章　結論

本研究藉由基因演算法，在patch的議題上，找出最少patch的配置方法，有別於過去針對特定因素所設計的演算法，在遭遇到不同問題的同時，也更能將較好的解留下來。

本研究在基因演算法的實做上，採用了「S之patches應完全蓋滿物件」、「染色體中patch和其他patch在物件區域內重複的情形」、「染色體中patch落在物件外的面積」三個要素來設計適應函數，並且利用突變，使其能夠較快速的找到要搜尋的最佳解。

為了要使影像處理時間縮短，或者傳輸時間縮短等等因素，我們使用了patching的方式，藉由將重要的部分以patch來表示，將圖中不必要的部分省略不做計算，以此大幅節省時間、空間等等。在求算patch的議題，當然能夠將patch個數減到最低，也愈能夠降低運算的時間、空間。由本實驗結果得知，基因演算法所搜尋出的最佳解，能夠優於一般配置的方法，也因此提供了在求算patch最佳配置的方式。

參考文獻

[1] Chambers, L., "*The Practical Handbook of Genetic Algorithms Applications*", CHAPMAN & HALL / CRC, 2001.

[2] Goldberg, D. E., "*Genetic Algorithms in Search, Optimization and Machine Learning*", Addison-Wesley, 1998.

[3] Tanimoto, S. L., R. J. Fowler, "*Covering image subsets with patches*", NATO Conference Series, (Series) 4: Marine Sciences 2, pp. 835-839, 1980

[4] Holland, J. H., "*Adaptation in Natural and Artificial System*", Ann Arbor: The University of Michigan Press, 1975.

[5] Koza, J. R., "*Survey of Genetic Algorithms and Genetic Programming*", Wescon Conference Record, pp. 589-594, 1995.

[6] Kumar, A., R. M. Pathak, M. C. Gupta, "*Genetic algorithm based approach for designing computer network topology*", 1993 ACM Computer Science Conference, pp. 358-365, 1993.

[7] Bhuyan, J. N., V. V. Raghavan, V. K. Elayavalli, "*Genetic algorithm for clustering with an ordered representation*",

Belew and Booker, pp. 408-415, 1991.

[8] De Lit, P., Falkenauer, E., Delchambre, A. "*Grouping genetic algorithms: An efficient method to solve the cell formation problem "*, Mathematics and Computers in Simulation 51 (3-4), pp. 257-271

[9] Haupt, R. L., "*An introduction to genetic algorithms for electromagnetics*", Antennas and Propagation Magazine, IEEE**,** Vol.37 , pp.7-15, 1995.

[10] Jones, D. R., M. A. Beltramo, "*Solving partitioning problems with genetic algorithms*", Belew and Booker, pp. 442-449, 1991.

[11] Tanimoto, S. L., "*Covering and indexing an image subset*" Proc. Recognition and Image Processing .IEEE Computer Society Conf., IL, 239-245 , 1979

[12] Fowler , R. J., M. S. Paterson, and S. L. Tanimoto, "*Optimal packing and covering in the plane are NP-complete*" Inform. Processing Lett., vol. 12, pp. 133-137, Apr. 1981.