雙向考量之多餘讀取器移除演算法

李怡慧　　劉鈞凱　　林志龍
嶺東科技大學 講師　嶺東科技大學 學生　嶺東科技大學 學生
sanity@mail.ltu.edu.tw　ss19990516@gmail.com　toyo2054@gmail.com

摘要

近年來由於無線射頻識別(Radio Frequency Identification, RFID)日臻成熟，在各種相關應用領域蓬勃發展，但受到硬體成本的限制，在使用 RFID 的應用上需要減少多餘讀取器，以減少成本的浪費。針對此一問題，已有許多學者提出移除多餘讀取器之演算法，但在相同環境下多餘讀取器移除的結果卻仍有改善空間。因此，本研究提出一個雙向考量之多餘讀取器移除演算法(Dual-Objective Elimination; DOE)，同時以鄰近讀取器數量做為讀取器的加權值，並以標籤的密集度為考量的加權值，進而判斷出不需要被使用的讀取器並加以移除。實驗結果顯示，DOE 在不同環境下能有效率地找到較多的多餘讀取器。

關鍵詞：RFID、多餘讀取器、鄰近讀取器

Abstract

Recently, radio frequency identification (RFID) applications increase widely. However, in a highly intensive RFID tags environment, it needs to use large number of reader, which easily causes a redundant reader's problem. Many studies had solved this problem, but there are still many cases cannot be found well by these proposed methods. This paper presents a Dual-Objective Elimination (DOE) algorithm to check reader and tag weights at the same time, while determining redundant RFID readers. The simulation results reveal that DOE algorithm can find more redundant RFID readers than comparison methods.

Keywords: RFID, Redundant Reader, Neighbor Reader.

1. 前言

RFID 發展已有一段時間，直到近年來才愈來愈被重視，主要原因為技術日漸成熟及被應用在供應鏈及一些物流產業上。由於 RFID 的便利性逐漸被大眾所接受，使得更多人開始研究應用在不同的領域。RFID 主要特點是可以快速的回應無線資料處理。因此，每個標籤需要快速正確地被讀取器讀到，否則無法達到快速的作業流程。然而，為了使多個標籤同時被讀取，進而衍生出碰撞的問題[1]。在讀取器過於密集 RFID 網路的環境中，容易造成讀取器之間的碰撞，使得讀取器所發出的訊號彼此干擾而造成無法判別的問題。

為了解決多餘讀取器的問題，讓 RFID 網路在有限的電量下有效的應用，本研究提出以密集度及加權值為基礎之 DOE 多餘讀取器移除演算法，此方法能夠判斷 RFID 網路中，每一個讀取器在周遭的環境下所提供的好處，並在計算後留下效益最高的移除效益最低的讀取器，且不會造成讀取器因移除而標籤無法讀到的情況發生。多餘讀取器的移除除了用來解決 RFID 碰撞問題，降低碰撞的機率外，還可以減少訊號的流失，來降低讀取器電量的耗損。基於上述原因，我們主要是對於多餘讀取器的移除加以改進。因此本篇論文的研究貢獻在於如何有效的找出最大量的多餘讀取器。

本論文的其他章節架構如下: 我們在第2節做一些相關文獻介紹，包括RFID系統架構，以及多餘讀取器問題的常見解決方法。第3節詳述本研究如何利用DOE演算法來解決多餘讀取器問題。第4節為實驗與結果分析。最後一節為結論與未來展望。

2. 文獻探討

RFID 最早可追溯到第二次世界大戰時期，英軍用於分辨飛機所屬陣營。應用的原理是將類似的今日使用的主動式標籤裝設在飛機上，透過雷達發射訊號到飛機上的標籤，標籤就會發出適當的回應的訊號，以此判斷飛機所屬陣營，目前世界上的飛行管制仍是以此為基礎[2][3]。既然這不是一項新興的技術，那為什麼會在這短短的幾年內就成為如此熱門的

話題及研究的主題呢？這要歸功於年營業額占全球零售業兩成，美國零售業的六成，被美國「商業週刊」稱為全球企業新獨裁者的 Wal-Mart 百貨公司。Wal-Mart 百貨在 2003 年 6 月即宣佈從 2005 年 1 月起，所有供應 Wal-Mart 百貨的商品裝箱上，都要有應用 RFID 技術的電子商品條形碼，全球一些零售體系如德國 Metro 等及美國國防部也都相繼跟進使用 RFID 技術，因此讓 RFID 成為了 2003 年最熱門的話題之一。

2.1 RFID 系統架構

RFID 是利用無線電波在讀取器與標籤之間傳遞訊息，其主要內容分成以下三類[2][4]：

2.1.1 電子標籤(Tag)

為資料的存放元件，能儲存產品的價格、基本特徵、組裝日期、出貨工廠、目前位置及其他數據...等，內含微細的晶片及天線，通常以電池的有無區分為主動式與半被動式及被動式三種類型[5]。

2.1.2 讀取器

區分為主動式與被動式其主要由類比、數位、中央處理單元及 RF 模組(天線、收發器)組成，可利用高頻電磁波傳遞能量與訊號，在每秒辨識出數百個電子標籤；則被動式 Reader 連接介面可區分成 TCP/IP Server、RS232 及 Mobile 等連接介面，主動式 Reader 除了包含了被動式介面，還有 TCP/IP Client、UDP Client 等連接介面[3]。

2.1.3 應用系統

包含中介軟體(Middleware)以及後端系統整合(System Integration)。中介軟體介於前端硬體與後端應用系統間，處理讀取器所擷取或接收到的資訊，且將資料過濾、處理，讓後端系統具有判斷與決策的資料[3]。Middleware 具有協調 Reader、資料過濾與彙總等功能。

2.2 RRE 演算法

Bogdan [6] 等人於 2005 年針對 Redundant Reader Elimination Problem 提出一演算法，簡稱 RRE，其研究目的在於用最少數量的讀取器讀取所有標籤，其運作方式是將每個標籤都分配一個讀取器作為持有者，當所有標籤出現的持有者皆不是該讀取器則可判為多餘讀取器。RRE 演算法的主要是利用貪婪法(Greedy Algorithm)的觀念，當面臨決策時，都做眼前最有利的，也就是說 RRE 演算法會選擇擁有最大標籤數量的讀取器來當該標籤的擁有者。不過也因為貪婪法的原因，造成 RRE 可能有誤判的情形發生。

2.2.1 RRE 可能出現的誤差

在 RRE 中，所有讀取器一開始均先掃描範圍內的標籤數量，之後回傳每個讀取器所擁有的標籤數量並作排序，數量最多的優先搜尋，但是在某些狀況下，將會無法找出最佳解，此構想並不滿足所有案例，圖 1 為 RRE 未能找出最佳解之範例。

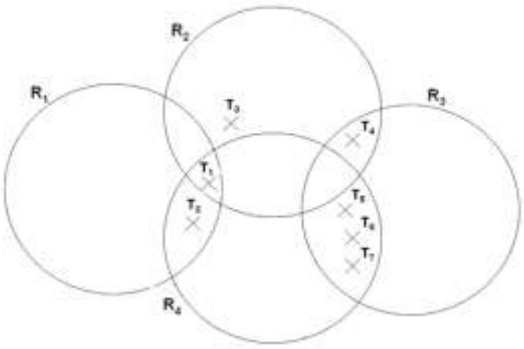

圖 1. RRE 未能找出最佳解之範例[7][8]

由表 1 顯示出，RRE 在圖 1 範例找出 R1 為多餘讀取器，但 R3 也是多餘讀取器，因此使用 RRE 演算法無法判斷出該結果。

表 1. RRE 演算法移除多餘讀取器結果

Reader	擁有之 Tag	執行順序	結果
R_1	T_1、T_2	3	
R_2	T_1、T_3、T_4	4	T_3
R_3	T_4、T_5、T_6、T_7	2	T_4
R_4	T_1、T_2、T_5、T_6、T_7	1	T_1、T_2、T_5、T_6、T_7
多餘讀取器 R_1			

2.3 LEO 演算法

Hsu 等人[9] 於 2007 所提出，A Layered Optimization Approach for Redundant Reader

Elimination in Wireless RFID Networks，簡稱 LEO。它與 RRE 演算法之多餘讀取器移除方式類似，主要還是將所有標籤分配給一個持有者，確保每個標籤至少有一個讀取器讀取。LEO 演算法主要以分層的方式，一開始採『先讀先贏』的概念做為選擇讀取器之方式，接著再執行 RRE 演算法，結合成整體的 LEO 演算法。

2.3.1 LEO 可能出現的誤差

由於 LEO 演算法一開始採用的是『先讀先贏』的概念，接著執行 RRE 演算法，因此在判斷的時間上，LEO 演算法明顯比 RRE 演算法少，因為在執行 RRE 演算法之前多了一層過濾多餘的讀取器。由於 LEO 演算法的讀取順序通常都是隨機選擇，所以每一次的執行結果可能不儘相同。

2.4 DRRE 演算法

有上述兩種演算法，經過分析比較後，可以知道 RRE、LEO 演算法在架構上均有不足的地方，RRE 演算法以計算的方式找出應該保留的讀取器，但是他在愈複雜的環境下，所能偵測到的多餘讀取器愈少；LEO 演算法雖然讀取/寫入次數有所減少，但對於所能偵測到的多餘讀取器數量則不盡相同。

因此林政頤[1]提出一種新的多餘讀取器移除架構(Density-based Redundant Reader Elimination, DRRE)，其架構是以鄰近讀取器數量作為讀取器是否的主要考量，在此將鄰近讀取器定義為兩個讀取器的可讀取範圍互相覆蓋，且這範圍裡有標籤可以讀取，則此兩個讀取器即為鄰近讀取器。一個讀取器對於鄰近讀取器所帶來的效益，就是能夠代替的讀取器數量愈多，其鄰近讀取器所能安全移除的數量愈多。因此，一個讀取器的鄰近讀取器之數量代表該讀取器對於周遭的讀取器所能產生的效益。

2.4.1 DRRE 可能出現的誤差

在 DRRE 是依照鄰近讀取器數量作優先行動的依據，但是在某些情況下無法找到最佳解。以圖 2 為例。

圖 2 中，若某些鄰近讀取器數量是相同者，則以讀取器 ID 判斷；表 2 即是較大的先執行。由表 2 顯示，DRRE 演算法在圖 2 範例找出 R_4、R_7 為多餘多取器，但實際上 R_2 亦是多餘讀取器，因此使用 DRRE 演算法在較複雜的情況下，無法找出其結果。

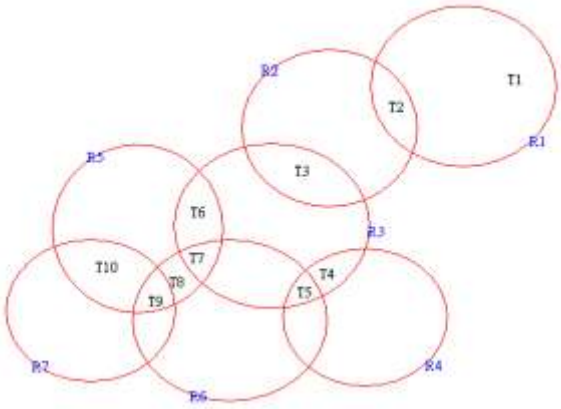

圖 2. DRRE 未能找出最佳解之範例

表 2. DRRE 演算法結果

Reader	擁有之 Tag	鄰近讀取器數量	執行順序	結果
R_1	T_1、T_2	1	7	T_1
R_2	T_2、T_3	2	4	T_2
R_3	T_3、T_4、T_5 T_6、T_7	4	1	T_3、T_4、T_5、T_6、T_7
R_4	T_4、T_5	2	5	
R_5	T_6、T_7 T_8、T_9、T_{10}	3	3	T_{10}
R_6	T_5、T_7 T_8、T_9	4	2	T_8、T_9
R_7	T_9、T_{10}	2	6	
多餘讀取器 R_4、R_7				

圖 2 中，若某些鄰近讀取器數量是相同者，則以讀取器 ID 判斷；表 2 即是較大的先執行。由表 2 顯示，DRRE 演算法在圖 2 範例找出 R_4、R_7 為多餘多取器，但實際上 R_2 亦是多餘讀取器，因此使用 DRRE 演算法在較複雜的情況下，無法找出其結果。

2.5 SDRRE 演算法

SDRRE (Simply Density-based Redundant Reader Elimination) 方法是由林政頤[9]提出的一種多餘讀取器移除架構，其流程是主要的 DRRE 之架構另外加上三個輔助方法，分別是標記法、部分寫入法、時間排序法；「標記法」是標籤進行持有者的寫入動作之前，各讀取器先對標籤寫入一個計算數量的變數，若標籤內

無任何紀錄標籤的持有者則變數設為 1，反之則此變數加 1，當所有讀取器寫入動作皆已完成，則會對每個標籤發出查詢的訊號，若查詢此變數的值為 1 表示為獨立標籤，代表僅被一個讀取器所讀取，因此該讀取器不可被移除，否則無法正常運作；「部分寫入法」是將原有取得鄰近讀取器的資訊，讀取器 ID 分別寫入標籤的動作，降低為止需寫入一定的比例即可。此方法可以大幅降低寫入次數，但本研究因時間上的考量沒有套用部分寫入法；「時間排序法」則是利用各讀取器的優先權當作執行時間的計算，在開始時所有讀取器皆被寫入一個相同的初始值，在 DRRE 演算法中的鄰近讀取器數量完成計算後，以初始值扣掉讀取器的優先權後，得到的數值為讀取器的等待時間值，當讀取器經過這段時間後，便會開始進行寫入動作。

SDRRE 演算法運作中，各讀取器事先不知彼此的存在，但利用標籤來達到資訊交換的功能，並進一步計算各讀取器的存在效益，再利用時間排序法訂出每個讀取器執行的時間，移除不會影響標籤的正常運作，留下效益最高者。

2.5.1 SDRRE 可能出現的誤差

在 SDRRE 中，是以精簡執行流程來減少讀取/寫入的使用，但是在某些情況下無法找到最佳解。以圖 2 為例。執行結果與 DRRE 相同如表 2 所示，其主要差別在於他會判斷 T_1 為獨立標籤，而 R_1 予以保留。而誤差部份也與 DRRE 相同。

2.6 First Search Right Algorithm

由洪瑞文[7][8]等人於 2010 提出改進的優先搜尋權演算法(First Search Right Algorithm, FSRA)，進而提昇多餘讀取器的判斷。FSRA 演算法運作流程如圖 3 所示。

2.6.1 FSRA 演算法可能出現的誤差

在 FSRA 演算法，採用的是『獨立標籤』的概念，也就是標籤的 rrt 是否等於 1，但在某些情況下，可能無法找出最佳解。以圖 4 為例。

由此範例圖顯示沒有標籤的 rrt 等於 1，有就是沒有獨立標籤，因此全部標籤的數量扣除 mk 得到優先搜尋權。表 3 顯示，FSRA 演算法在圖 4 範例找出 R1、R5 為多餘多取器，但實

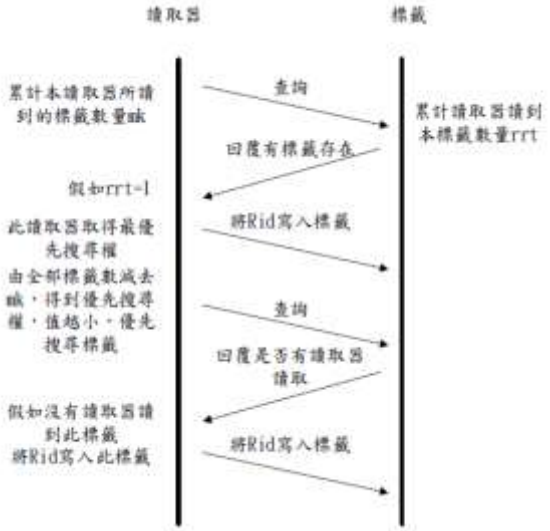

圖 3. FSRA 演算法運作流程[2][8]

際上 R3 亦是多餘讀取器，因此使用 FSRA 演算法無法判斷出該結果。

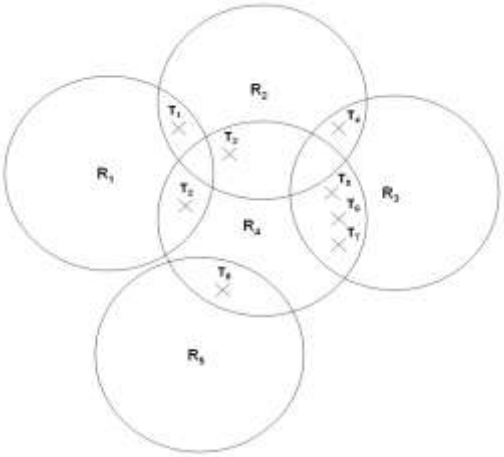

圖 4. FSRA 演算法未能找出最佳解之範例

表 3. FSRA 演算法結果

Reader	擁有之 Tag	Tag 數量	獨立標籤	執行順序	結果
R_1	T_1、T_2	2	×	4	
R_2	T_1、T_3、T_4	3	×	3	T_1
R_3	T_4、T_5、T_6、T_7	4	×	2	T_4
R_4	T_2、T_3、T_5、T_6、T_7、T_8	6	×	1	T_2、T_3、T_5、T_6、T_7、T_8
R_5	T_8	1	×	5	
多餘讀取器 R_1、R_5					

3. 雙向考量之多餘讀取器移除演算法

從上一節分析可知，RRE 演算法由於讀取器的重疊關係因此在找尋多餘讀取器時會產生誤判，FSRA 演算法雖然改進了 LEO 的缺點，但是當沒有獨立標籤時會有跟 RRE 一樣的誤判，DRRE 演算法主張用讀取器的角度來思考但還是有其缺陷。因此，我們提出 Dual-Objective Elimination (DOE) 演算法，雙向考量讀取器的加權值及標籤的加權值來加以改善。

假設 RFID 碰撞問題已經被避免，且 RFID 標籤採用的是半被動式標籤，擁有部分記憶空間可供讀寫。我們定義「鄰近讀取器」為兩個讀取器的可讀取範圍互相覆蓋，且這範圍裡有標籤可以讀取，則此兩讀取器即為鄰近讀取器。定義「鄰近讀取器加權值」為鄰近讀取器的數量，記為 R_weight。定義「標籤加權值」T_weight 代表該標籤被讀取器覆蓋的狀況，例：假若 A 標籤只被一個甲讀取器所讀到，其優先權較高，加權值會較大; 若 B 標籤被三個讀取器(甲、乙、丙)所讀到，則 B 標籤的加權值較 A 標籤的加權值小。T_weight 的計算如公式(1)，其中 N 為該標籤被幾個讀取器覆蓋的數量。若 N=1 則 T_weight=1，否則由公式 (1) 計算之。

$$T\_weight = \begin{cases} N * 0.1^{N}, N \geq 2 \\ 1, N = 1 \end{cases} \quad (1)$$

$$R\_priority = R\_weight * T\_weight \quad (2)$$

而讀取器所擁有的標籤的 T_weight 再乘上 R_weight 為讀取器的優先權值稱為 R_ priority，如公式(2)。R_ priority 越高者越不應被移除，越低者則可能會是多餘讀取器。此公式設計理念來自於圖 5，若 T_weight 的值沒有與獨立標籤劃分開來，則會發生誤判，原因是獨立標籤的加權值不夠大。

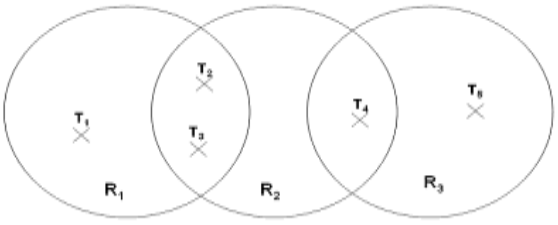

圖 5. DOE 在某些情況下的誤判

以下 3.1 小節將介紹 DOE 演算法，並舉例說明。3.2 小節將以 DOE 演算法來執行第二節中各演算法會誤判的範例。

3.1 Dual-Objective Elimination 演算法

DOE 演算法詳列如下:

Initialization：

- 各標籤給定一變數 N 用來存放被讀取到的數量。
- 各標籤給定一變數 T_weight 存放該標籤的加權值。
- 各讀取器給定一個變數 rred 用以判斷是否為多餘讀取器，並將 rred 設為 0。
- 各讀取器給定一變數 R_sumT_weight 存放所有覆蓋到的標籤的加權值。
- 各讀取器給定一變數 R_weight 存放鄰近讀取器數量。
- 各讀取器給定一變數 R_ priority 存放優先權值。

Procedure

Step 1：

廣播所有讀取器搜尋範圍內有的標籤紀錄。；

1. 記錄讀取器覆蓋哪些標籤。
2. 記錄標籤被幾個讀取器讀取到此值為 N。

Step 2：

1. 計算 T_weight 的值。
2. 計算該讀取器的鄰近讀取器的數量並存入 R_weight。
3. 計算該讀取器所覆蓋的標籤加權值總合存入 R_sumT_weight。

Step 3：

計算每個讀取器的優先權值

Step 4：

依照讀取器的優先權值 R_ priority 依照大到小的順序進行標籤的擁有，若該讀取器有擁有標籤則 rred=1。

Step 5：

若該讀取器 rred=0 則關閉。

我們以圖 2 網路環境為例來說明此演算法執行流程。圖 2 中，{R_1, R_2, R_3, R_4, R_5, R_6, R_7} 為 RFID 讀取器，{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}} 為 RFID 標籤，假設執行順序為 $R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_4 \rightarrow R_5 \rightarrow R_6 \rightarrow R_7$，其執行步驟如下：

1. 所有讀取器均先發送一個訊號掃描範圍內的標籤數量並將將本身的 ID 寫入掃描範圍內的標籤。以 R_1、R_2 為例 R_1 會寫入到 T_1 及 T_2 中，R_2 會寫入到 T_2 及 T_3 中，因此當所有讀取器皆完成寫入動作，T_1 內會保留有 R_1 的資訊，T_2 內會保有

R_1 及 R_2 的資訊，以此類推。

2. 計算 R_weight 與 T_weight 的值。以表 4、表 5 所示;所有讀取器對範圍內的標籤進行查詢,確認標籤內是否紀錄高標籤的持有者及優先權，因此從 R_1 到 R_7 的 R_weight 分別為{1, 2, 4, 2, 3, 4, 2}；而標籤的的值套入公式 N*(0.1)^N，N 為被幾個讀取器給涵蓋的數量，若 N 等於 1 則該標籤的 T_weight=1，不是則套入公式計算，因此 T_1 到 T_{10} 的 T_weight 分別為{1, 0.02, 0.02, 0.02, 0.003, 0.02, 0.003, 0.02, 0.003, 0.02}。
3. 所有讀取器對範圍內的標籤進行查詢，並進行過濾比對,計算讀取器所覆蓋的標籤的 T_weight 並乘上該讀取器的 R_weight 則為 R_ priority。
4. 各讀取器以 R_ priority 大小順序來對於其掃描範圍內的標籤進行查詢，查看標籤內是否紀錄該標籤的持有者，若是沒有，則發出寫入訊息，將該讀取器的讀取器 ID 及優先權寫入至該標籤內，若有則放棄。因此以此圖為例 R_1 到 R_7 的 R_ priority 為(1.02, 0.08, 0.264, 0.046, 0.198, 0.116, 0.046)，則 R_1 會先寫入 T_1 及 T_2 為擁有者並紀錄不為多餘讀取器，接下來由 R_3 寫入 T_3、T_4、T_5 及 T_6 與 T_7 並紀錄不為多餘讀取器，再來由 R_5 寫入 T_8 及 T_9 與 T_{10} 為擁有者並紀錄不為多餘讀取器。之後的 R_2、R_4、R_6、R_7 街無寫入標籤則紀錄為多餘讀取器。
5. 確定所有讀取器都已完成資料寫入後，再次對讀取器發出詢問訊號,詢問標籤持有者的分配狀況,各讀取器若是紀錄為多餘讀取器則自動進入睡眠狀態。其執行結果如表 6 所示

表 4. DOE 演算法之標籤加權值

Tag	T_weight
T_1	1
T_2	0.02
T_3	0.02
T_4	0.02
T_5	0.003
T_6	0.02
T_7	0.003
T_8	0.02
T_9	0.003
T_{10}	0.02

表 5. DOE 演算法之讀取器加權值

Reader	R_weight
R_1	1
R_2	2
R_3	4
R_4	2
R_5	3
R_6	4
R_7	2

表 6. DOE 演算法結果

Reader	擁有之 Tag	R_priority	執行順序	結果
R_1	T_1、T_2	1.0200	1	T_1、T_2
R_2	T_2、T_3	0.0800	5	
R_3	T_3、T_4、T_5 T_6、T_7	0.2640	2	T_3、T_4、T_5、T_6、T_7
R_4	T_4、T_5	0.0460	6	
R_5	T_6、T_7、T_8 T_9、T_{10}	0.1980	3	T_8、T_9、T_{10}
R_6	T_5、T_7 T_8、T_9	0.1160	4	
R_7	T_9、T_{10}	0.0460	7	
多餘讀取器 R_2、R_4、R_6、R_7				

3.2 其他多餘讀取器誤判

DOE 演算法在找尋多餘讀取器的數量上要比其他的演算法多，本節以本演算法去解第二節其他演算法會有誤判的圖。

3.2.1 RRE 演算法誤判

以圖 1 為例，DOE 演算法執行結果如表 7-9 所示，較 RRE 多找尋出一個多餘讀取器

表 7. DOE 演算法之標籤加權值

Tag	T_weight
T_1	0.003
T_2	0.02
T_3	1
T_4	0.02
T_5	0.02
T_6	0.02
T_7	0.02

表 **8. DOE** 演算法之讀取器加權值

Reader	R_weight
R_1	2
R_2	3
R_3	2
R_4	3

表 **9. DOE** 演算法結果

Reader	擁有之 Tag	R_priority	執行順序	結果
R_1	T_1、T_2	0.046	4	
R_2	T_1、T_3、T_4	3.069	1	T_1,T_3, T_4
R_3	T_4、T_5 T_6、T_7	0.16	3	
R_4	T_1、T_2、T_4 T_5、T_6、T_7	0.309	2	T_2,T_5 T_6,T_7
多餘讀取器：R_1、R_2				

3.2.2 LEO 演算法誤判

由於 LEO 演算法一開始是以讀取器來進行判別，判別方式是先讀取到先擁有，缺點是無法決定先後順序來對標籤去進行擁有，因此演算法在任何一個圖裡都可能會有誤判的情況發生。

3.2.3 DRRE 演算法誤判

由於 DRRE 演算法主要是以鄰近讀取器的數量多寡來決定多餘讀取器，當鄰近讀取器的重疊的數量過多時，會有誤判的情況發生。此誤判前面已敘述過，因此參考 3.1 節。

3.2.4 SDRRE 演算法誤判

SDRRE 演算法主要是 DRRE 演算法架構下再加上計算獨立標籤的概念，但是在某些情況下會發生與 DRRE 相同的誤判情況，因為是發生與 DRRE 同樣的問題因此參考 3.1 節。

3.2.5 FSRA 演算法誤判

FSRA 演算法的誤判發生在於，若是該範例沒有獨立標籤，即會發生與 RRE 相同的誤判。以圖 4 為例，其執行結果如表 10-12 所示，較 FSRA 演算法多找出一個多餘讀取器 R3。

表 **10. DOE** 演算法之標籤加權值

Tag	T_weight
T_1	0.02
T_2	0.02
T_3	0.02
T_4	0.02
T_5	0.02
T_6	0.02
T_7	0.02
T_8	0.02

表 **11. DOE** 演算法之讀取器加權值

Reader	R_weight
R_1	2
R_2	3
R_3	2
R_4	5
R_5	1

表 **12. DOE** 演算法結果

Reader	擁有之 Tag	R_ priority	執行順序	結果
R_1	T_1、T_2	0.8	4	
R_2	T_1、T_3、T_4	0.18	2	T_1、T_3、T_4
R_3	T_4、T_5 T_6、T_7	0.16	3	
R_4	T_2、T_3、T_5、T_6、T_7、T_8	0.6	1	T_2、T_3、T_5、T_6、T_7、T_8
R_5	T_8	0.02	5	
多餘讀取器：R_1、R_3、R_5				

4. 模擬實驗及結果比較

本研究參考[2][8]的實驗設計，以 JAVA 設計一個 RFID 模擬環境，在固定區域大小 1000m× 1000m 的單位距離範圍內，佈下 500 個讀取器，利用不同的環境變數進行比較，標籤數量預設為 1000 個，每個讀取器的半徑範圍預設為 50m，並且確定每個標籤一定會被所有的讀取器所讀取。

除了實作本研究提出的 DOE 演算法，同時實作 RRE 演算法、LEO 演算法、DRRE 演算法、SDRRE 演算法與 FSRA 演算法做比較。比較項目為平均執行時間，以及找到的多餘讀取器數量。

本實驗設計兩種模擬環境，詳細內容如以下兩個小節。

4.1 模擬環境-隨機佈點

本實驗利用隨機產生讀取器與標籤的座標位置方式，觀察各種不同的情況下的執行結果。此實驗主要探討的是在讀取器的半徑大小不同之下對於多餘讀取器移除數量的影響，以及在相同的環境裡標籤數量多寡是否會影響多餘讀取器的數量變化。此實驗又細分為兩個子實驗：

(1) 變數為讀取器讀取半徑

其中不變量為讀取器數量為 500；標籤數量為 8000；變數讀取器的讀取範圍為 50m~100m 進行比較，並且重複 10 次取平均值。

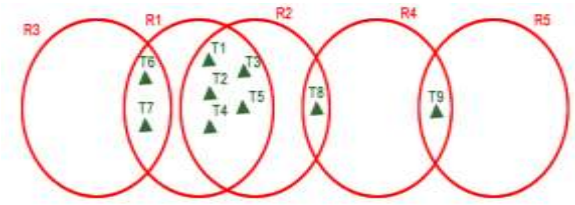

圖6. 佈點過於密集的誤判範例圖

佈點過於密集的情況，會導致所有的演算法都會造成誤判，如圖 6 中 R_1 與 R_2 過於類似，因此 R_1 執行完換 R_2 執行，但此範例來說 R_2 是多餘的讀取器因此會造成誤判。由圖 7 可看出半徑改變對於多餘讀取器偵測數量的影響，在半徑 50m~60m 的時候 DOE 演算法效果比其他的演算法還要好。理論上 DOE 可以找出 LEO 找不到的情況，但是半徑為 70m 之後 LEO 會大幅領先其他演算法，因為模擬實驗的距離過於小，1000mX1000m 的距離下，半徑 100m 的相較之下約 100 個讀取器就可佈滿整個範圍。在於過於密集的時候每個讀取器的優先值都差不多，因此會造成誤判。

由於目前所有的演算法主要是以密集的概念為主軸，因此目前的演算法都無法避免此情形，而 LEO 用了分層概念，但主軸還是使用 RRE 來做第二次搜尋，所以還是有多餘讀取器未移除。

由圖 8 可知，半徑變動並不會影響執行時間，但可知道 RRE 勝於 LEO、DRRE、FSRA 與 DOE，更勝於 SDRRE。

(2) 變數為標籤數量

此實驗中的不可變量為讀取器數量為 500；讀取器的讀取範圍為50m；變動量為標籤數量為1000~8000進行比較。圖9為以標籤數量改變來比較多餘讀取器偵測數量的曲線圖，從圖中可看出DOE演算法不論標籤數量為多少

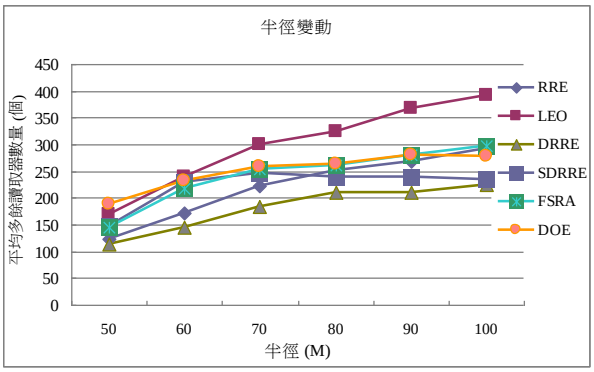

圖7. 變數為讀取器讀取半徑之數量比較

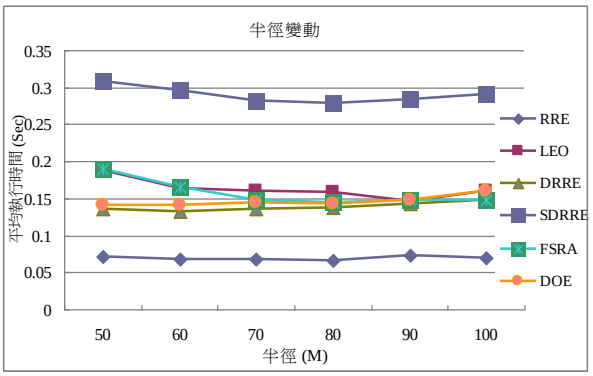

圖8. 變數為讀取器讀取半徑之時間比較

都比其餘演算法所找出的多餘讀取器還要來的多，而隨著標籤數量上升，各種演算法所能找出的多餘讀取器數量差距越來越大，到標籤數量為 8000 時，比 RRE 多找出 31%的多餘讀取器，比 LEO 多出 10%的多餘讀取器，比 DRRE 多出 43%的多餘讀取器，比 SDRRE 多出 17%的多餘讀取器，比 FSRA 多出 20%的多餘讀取器。

圖 10 為標籤數量變動的時間比較圖，RRE 由於只有一個判斷所花費的時間為最少，LEO 較 RRE 多一個搜尋的步驟因此時間較 RRE 多，DRRE、FSRA 與 DOE 由於多了一個判斷的步驟，因此時間較 RRE 多出約 50%的時間，但從數據中可得知對於多餘讀取器偵測的最終結果與 DOE 相比，此影響幾可忽略不計。而 SDRRE 由於多比 DOE 多出一個處理步驟，所花費的時間比 DOE 演算法還要多出約 1 倍的時間，但是找出的多餘讀取器數量並沒有優於 DOE。

4.2 模擬環境-固定佈點

由於實際佈建 RFID 網路時，讀取器不可能是隨機佈點。通常是佈好點再視情況將多餘得讀取器休眠或是關閉。因此，本實驗環境為

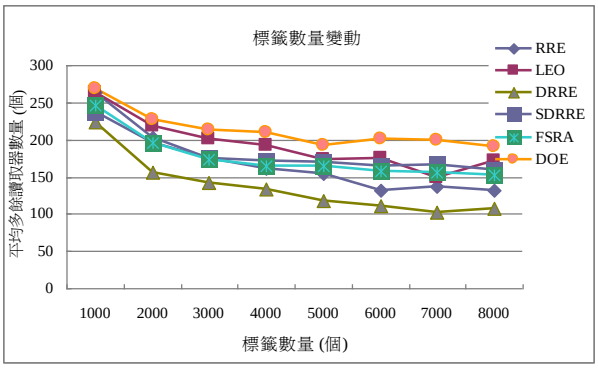

圖 9. 變數為標籤數量之數量比較

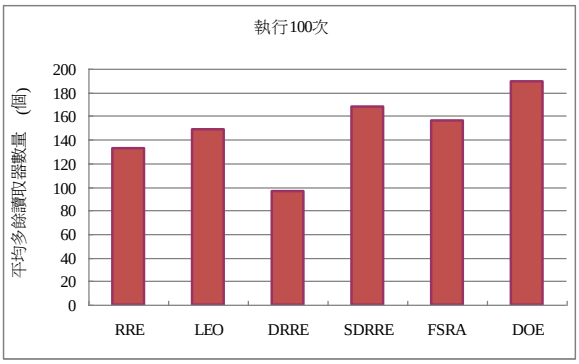

圖11. 執行100次之數量比較

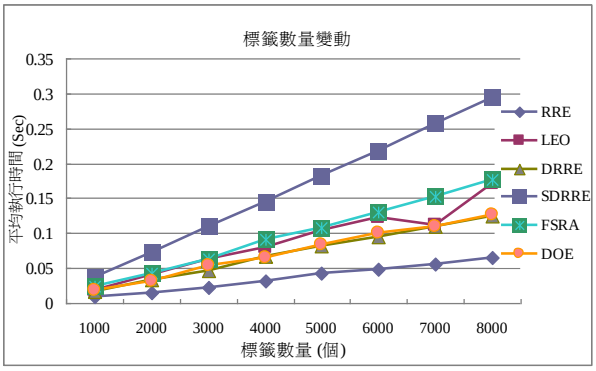

圖10. 變數為標籤數量之時間比較

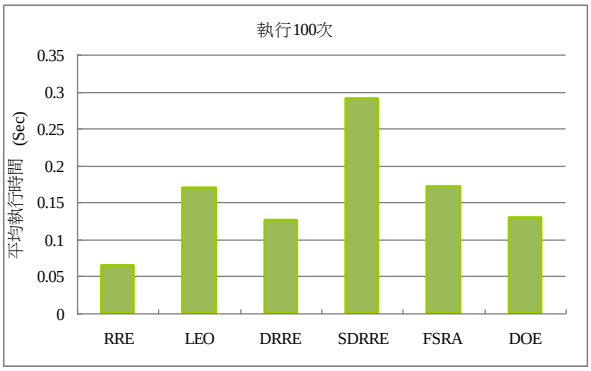

圖12. 執行100次之時間比較

一些固定數量及位置的讀取器，與固定數量及隨機分配位置的讀取器，以模擬在動態環境下標籤會隨時變動的狀況。讀取器數量固定為 500，讀取器的讀取半徑固定為 50m，標籤數量固定為 8000 個，並隨機佈點標籤。本實驗重複進行 100 次，取執行結果平均值。由圖 11 可知在動態環境下，DOE 所找出的多餘讀取器數量比其他的演算法多。比 RRE 多出約 30%的多餘讀取器，比 LEO 多出約 21%的多餘讀取器，比 DRRE 多出約 50%的多餘讀取器，比 SDRRE 多出約 11%的多餘讀取器，比 FSRA 多出約 17%的多餘讀取器。

由圖 12 可知 RRE 雖然所需時間消耗最少，但是找出的多餘讀取器數量卻不盡理想，DRRE雖然時間比DOE少但是找到的讀取器數量卻是最差的，LEO、SDRRE與FSRA不但找出的數量無法與DOE找出的多餘讀取器數量相比，連時間與DOE都有差距，在動態環境下時間很重要，若標籤的位址又變動，則會有誤判。

5. 結論與未來展望

本論文提出移除多餘讀取器的DOE演算法，與以往的多餘讀取器相關研究比較下，DOE 演算法對於多餘讀取器的偵測數量獲得大幅的改善。在一個密集的 RFID 網路環境下，DOE 可以找出比 RRE 多約 30%的多餘讀取器數量，雖然時間上的花費較 RRE 多出 50%，但是找出的多餘讀取器的數量報酬應該是平衡的。DOE 能找出比 LEO 約 20%的多餘讀取器數量，但是時間上的花費相比之下卻比 LEO 還要快上 30%，不過在範圍小的地方 LEO 卻是大勝各個演算法，原因是因為在小範圍裡，太多的判斷會造成誤判。而 DOE 可以找出比 DRRE 多出約 50%的多餘讀取器數量，而且時間比 DRRE 還要快上約 1 倍的時間。另外 DOE 演算法可以找出比 SDRRE 約 11%的多餘讀取器，但是時間上就是大勝 SDRRE。最後，DOE 演算法可以找出比 FSRA 多出 17%的多餘讀取器，而時間上比 FSRA 快上約 32%。由以上可知 DOE 演算法基本上在找尋多餘讀取器的效用上要比其他的演算法還要來的好。

在實驗中我們發現所有的演算法都是以密集的情況來作為讀取器優先存取的順序，因此在於圖6的情況上是無解的，而LEO因為用了分層概念先過濾一次，而第一步驟的概念並不是以密集度為主軸的概念，因此可以避開此誤判的情況，但是就以理論上依然未能找出最佳的多餘讀取器數量。

以密集度為主軸的方法都會有上述問題，所以本研究正在思考此一問題的解法。此外，如何將DOE演算法運用在動態環境下，病仍能有效率地偵測出的多餘讀取器數量，也是未來的研究方向。

參考文獻

[1] 林政頤，***在 RFID 網路中以密集度為基礎之多餘讀取器移除演算法***，碩士論文,中華大學資訊管理學系，2008。

[2] 奚正德，***RFID 相關應用與安全機制簡介***，博士論文，長庚大學電機研究所，2005。

[3] 張登訓、陳農坤、陳俊良，”高效率 RFID Middleware 設計”, ***TANet 2008 研討會,*** pp.42, 2008。

[4] “RFID 介紹與校園 RFID 之應用”, http://ppt.cc/eS6E.

[5] 陳昱仁、廖耕億、許建隆、林仲志、鍾乾癸著，***RFID 概論***，華泰文化，2009。

[6] Bogdan, C., Murali, K. R., Mehmet, K., Christoph, H. and Ananth, G., ”Redundant-Reader Elimination in RFID Systems,” ***2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks***, pp. 176- 184, 2005.

[7] 洪瑞文、李怡慧、林信宏、蔡竣安，”以優先搜尋權法於 RFID 網路多餘讀取器移除之研究”, ***Network and RFID Technology Conference (NRT 2009),*** pp.70-75, 2009。

[8] Hung, J. W., Li, I. H., Lin, H. H. and Cai, J. A., "The First Search Right Algorithm for Redundant Reader Elimination in RFID Network," ***The 9th WSEAS International Conference on SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS (SEPADS '10)***, pp.177-183, 2010.

[9] Hsu, C. H., Chen, Y. M. and Yang, C. T., “A Layered Optimization Approach for Redundant Reader Elimination in Wireless RFID Networks,” ***The 2nd IEEE Asia-Pacific Service Computing Conference***, Vol. 11, No. 14, pp. 138-145, 2007.