國　立　臺　灣　科　技　大　學

資訊管理系

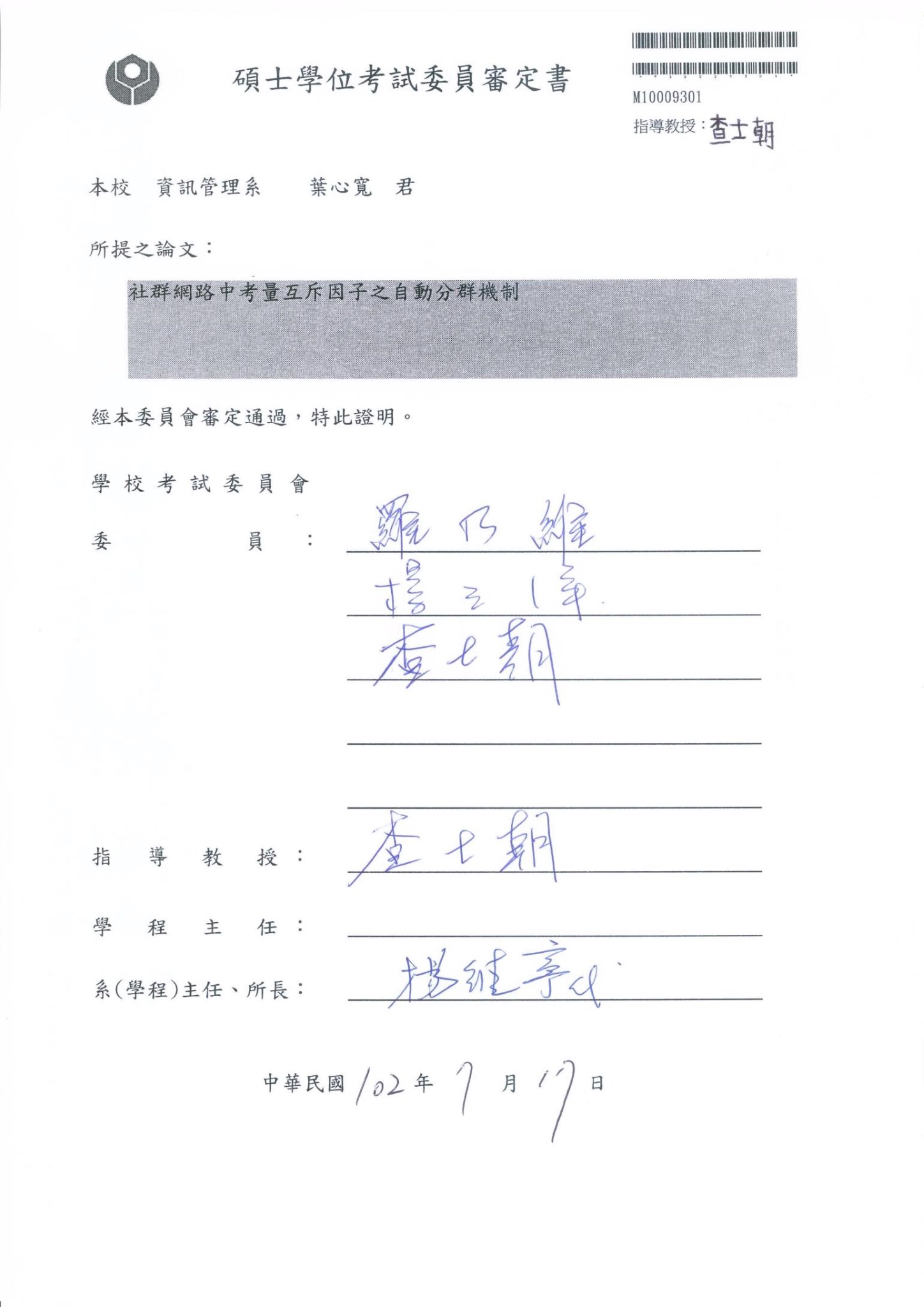
碩 士 學 位 論 文

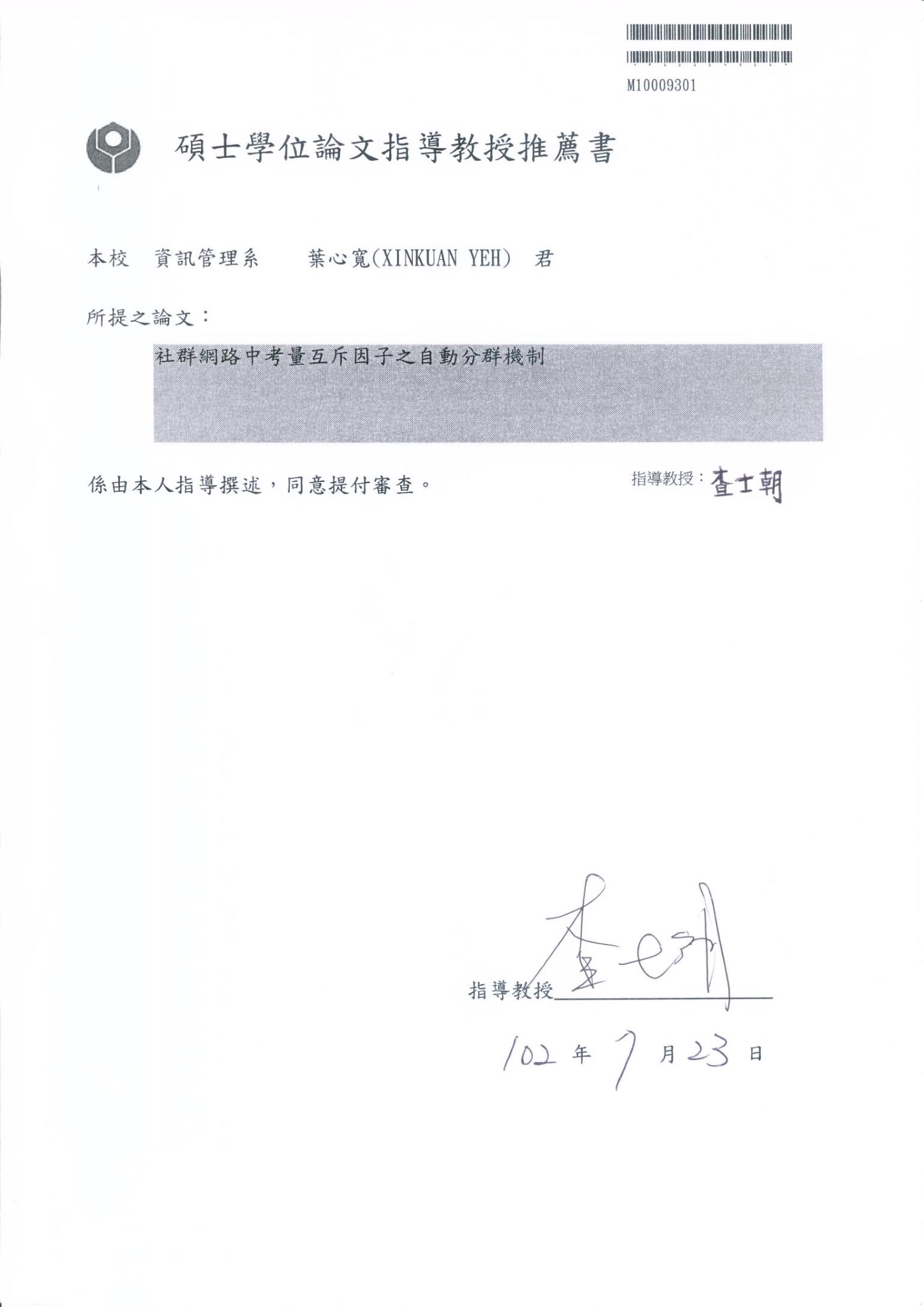
|  |
| --- |
| 學號：M10009301 |

社群網路中考量互斥因子之  
自動分群機制  
  
An automatic clustering mechanism considering conflicts among friends   
for social network

研 究 生： 葉心寬  
指 導 教 授： 查士朝 博士

中華民國一○二年七月





**摘要**

在現實生活中使用者透過Facebook社群網站分享資訊時，為了管理分享資訊的散播，此時可以針對分享資訊個別選擇特定的朋友，可是當特定的朋友數越多時會讓造成使用者花費更多的管理成本，所以為了方便分享資訊時的操作，在Facebook 社群網路允許使用者將朋友分至特定的朋友名單，讓使用者在分享資訊時就能限制朋友名單中的所有朋友，簡化操作，此外雖然可以透過智慧型清單或自動分群機制能協助朋友名單管理，然而過去不論是智慧型清單或自動分群機制皆是將相似的朋友直接分成相同群組，卻未考量到朋友之間彼此衝突的情況。

因此本研究提出考量互斥因子之自動分群機制，主要是比較多種分群演算法進行群組適當性和時間複雜度的探討，最後選擇 BGLL分群演算法為基礎進行改良，主要考量到朋友之間彼此衝突的情況，讓使用者提供回饋資訊進行條件設定之後，再針對使用者在Facebook社群網路中之個人網路進行分群，此時將會產生符合使用者回饋資訊的需求條件的群組。

然而在分群群組的評估分析上，過往研究者主要是透過評估指標進行比較分析，因為若要請大量使用者直接針對所有群組進行評估，在實驗上會非常的困難，所以少有研究是直接根據使用者回饋進行比較分析。所以本研究會實作系統協助使用者針對考量互斥因子之自動分群機制產生的群組進行直接的調整，待確認之後即時透過資訊檢索領域中廣泛被應用的查準率與查全率進行比較分析。

關鍵字：社群網路、社群偵測、分群演算法、個人網路分析

|  |
| --- |
| **Abstract** |
| When users use social network services, such as Facebook, Twitter and Google+, to share information, users may cluster their friends into groups and share information based on the groups to reduce costs of setting who can access the information. In this case, the more friends a user has, the more cost the user needs to put the user’s friends into groups.  Therefore, researchers develop approaches to help users to cluster or group their friends in social network services automatically. For example, Facebook provides a function to put friends of a user into friendlists based on user profiles. Therefore, users can restrict the information to be accessed by friends in selected friend lists. However, current automatic friends grouping researches focus on the similarity among user friends. We may consider conflicts among friends to increase effectiveness of friends grouping.  In addition to similarity among friends of users, this paper proposes a novel approach to group friends of users in social network services considering conflicts among friends. After comparing several current friends grouping approaches, this study select BGLL as basis and extend BGLL to consider conflicts among friends. This research further implements a system to help users consider conflicts among friends and put their friends into groups based on the proposed approach automatically.  Finally, current researches usually do not collect user feedback to evaluate effectiveness of grouping results directly. This paper proposes a method to evaluate precision and recall of friend grouping approaches base on user feedback. Therefore, this research also contributes to provide a scheme for evaluating effectiveness of friend grouping approaches in social network services.  Keywords: social network, clustering algorithm, community detection, ego-centric network analysis |

**誌謝**

　　當口試結束的那一刻，兩位口試委員伸出雙手向我說恭喜時，在我心中充滿感謝，感謝指導教授查士朝博士，這段期間內的辛勤指導，若有人問我在研究所中學會最多的是什麼？我會回答做人。在過去研究所的二年內我太過於自私自利的態度，使得老師非常頭痛，可是老師並沒有因此放棄我，反而還給我許多的機會，更細心的教導我正確的態度，並且指導我正確的研究方向，讓我能如此順利完成此篇碩士論文，對於老師的諄諄教誨與愛護，我銘感五內非常感謝。此外亦很感謝兩位口試委員楊立偉教授和羅乃維教授，兩位口試委員的建議句句金玉良言，每一句話都讓我有新的想法。這些建議讓我可以從多元的角度來看我的研究議題，尤其在於未來有趣的研究建議，將使得這篇論文更臻完善。

　　感謝風險管理與隱私保護實驗室的學長姐們，彥谷學長、漣真學姐、世峰學長，雅萍學姐和允廷學長，以及 MA404 實驗室的優秀學長姐們，在研究所時對我的包容與照顧，此外，特別感謝俊甫學長和鎮宇學長，當我徬徨無助時給予我許多的鼓勵，若沒有學長不厭其煩細心的指導，此篇論文就很難順利的完成，真的非常感謝學長的溝通與指引。另外，感謝同學們，王婕同學、乃華同學、信延同學、伃婷同學和家慧同學，謝謝妳/你們的不離不棄，更在我研究期間給予的扶持和鼓勵，很開心我們 都能順利畢業。學弟妹，士杰學弟、冠衡學弟、志騰學弟、士程學弟、辰宇學弟和育軒學弟，在研究所時對我的包容與幫助。最後感謝我的母親及妹妹，對我的愛與關懷，是我精神上最大的支柱。尤其母親無怨無悔地支持我想清楚之後所做的任何決定，在此，謹以我的畢業論文獻給她，也希望大家能分享我努力的成果與喜悅，以及非常感謝所有協助我進行實驗之 Facebook 社群網站中的朋友們。

葉心寬　謹誌於

國立台灣科技大學資訊管理研究所

風險管理與隱私保護實驗室

2013 年 7 月 23 日

**目錄**

[**第一章、** **緒論** 1](#_Toc360928669)

[1.1 研究背景 1](#_Toc360928670)

[1.2 研究動機 6](#_Toc360928671)

[1.3 研究目的與貢獻 7](#_Toc360928672)

[1.4 章節簡介 8](#_Toc360928677)

[**第二章、** **文獻探討** 9](#_Toc360928678)

[2.1 社群網路之社群偵測 9](#_Toc360928680)

[2.2 Modularity – Q值 14](#_Toc360928684)

[2.3 BGLL 分群演算法 24](#_Toc360928685)

[2.4 社群網站之群組應用 31](#_Toc360928686)

[2.5 評估分群結果 34](#_Toc360928687)

[**第三章、** **問題定義與解決機制** 43](#_Toc360928688)

[3.1 問題定義 43](#_Toc360928689)

[3.2 做法說明 44](#_Toc360928694)

[3.3 範例驗證 48](#_Toc360928695)

[**第四章、** **系統開發與實作** 59](#_Toc360928696)

[4.1 基本介紹 59](#_Toc360928697)

[4.2 應用說明 66](#_Toc360928702)

[**第五章、** **實驗與評估分析** 74](#_Toc360928703)

[5.1 實驗設計 74](#_Toc360928704)

[5.2 評估分析 78](#_Toc360928709)

[**第六章、** **結論與未來建議** 81](#_Toc360928710)

[6.1 結論 81](#_Toc360928711)

[6.2 未來研究方向與建議 83](#_Toc360928712)

[**參考文獻** 84](#_Toc360928713)

**圖目錄**

[圖1-1、Facebook 社群網站朋友名單的應用 1](#_Toc361446658)

[圖1-2、Facebook 社群網站朋友名單畫面 2](#_Toc361446659)

[圖1-3、Facebook 社群網站智慧型清單畫面 4](#_Toc361446660)

[圖1-4、Facebook 社群網站隱私設定與工具畫面 4](#_Toc361446661)

[圖2-1、個人網路 (ego-centric network) 社群偵測示意圖 9](#_Toc361446623)

[圖2-2、適用社群偵測之分群演算法之分類 10](#_Toc361446624)

[圖2-3、文獻探討架構圖 11](#_Toc361446625)

[圖2-4、空手道俱樂部成員之關係網路 12](#_Toc361446626)

[圖2-5、空手道俱樂部成員之關係網路 – 分裂階層圖 13](#_Toc361446627)

[圖2-6、簡單範例 – 分群結果不適當 16](#_Toc361446628)

[圖2-7、簡單範例 – 分群結果適當 16](#_Toc361446629)

[圖2-8、評估群組的適當性示意圖 18](#_Toc361446630)

[圖2-9、最大化Q值之整數線性規劃應用至空手道俱樂部之分群結果 23](#_Toc361446631)

[圖2-10、BGLL 分群演算法 – 兩階段聚合示意圖 27](#_Toc361446632)

[圖2-11、BGLL 分群演算法 – 流程圖 28](#_Toc361446633)

[圖2-12、社群網站之群組應用的關係圖 32](#_Toc361446634)

[圖2-13、考量互斥因子之自動分群機制之社群網站應用 33](#_Toc361446635)

[圖2-14、LFR 基準圖之示意圖 36](#_Toc361446636)

[圖2-15、系統判斷群組與人工判斷群組 – 查準率示意圖 38](#_Toc361446637)

[圖2-16、系統判斷群組與人工判斷群組 – 查全率示意圖 39](#_Toc361446638)

[圖2-17、評估分群結果之架構圖 42](#_Toc361446639)

[圖3-1、考量互斥因子之自動分群機制 – 流程圖 46](#_Toc361582032)

[圖3-2、R Studio 實際操作畫面 48](#_Toc361582033)

[圖3-3、空手道俱樂部的成員關係網路 – 不同分群演算法比較表 49](#_Toc361582034)

[圖3-4、空手道俱樂部的成員關係網路 – 真實情況示意圖 50](#_Toc361582035)

[圖3-5、空手道俱樂成員之關係網路 – 相關互斥條件 50](#_Toc361582036)

[圖3-6、空手道俱樂部的成員關係網路 – 考量互斥因子之自動分群機制比較 51](#_Toc361582037)

[圖3-7、MATLAB實際操作畫面 52](#_Toc361582038)

[圖3-8、逐步分解之初始化設定 52](#_Toc361582039)

[圖3-9、考量互斥因子之自動分群機制之逐步分解 – 第一階段 53](#_Toc361582040)

[圖3-10、考量互斥因子之自動分群機制之逐步分解 – 第二階段 54](#_Toc361582041)

[圖3-11、考量互斥因子之自動分群機制之逐步分解 – 分群結果 54](#_Toc361582042)

[圖3-12、BGLL 分群演算法之逐步分解 – 第一階段 55](#_Toc361582043)

[圖3-13、BGLL 分群演算法之逐步分解 – 第二階段 56](#_Toc361582044)

[圖3-14、BGLL 分群演算法之逐步分解 – 分群結果 56](#_Toc361582045)

[圖3-15、考量互斥因子之自動分群機制後續調整比較表 58](#_Toc361582046)

[圖4-1、朋友分群系統示意圖 61](#_Toc362083046)

[圖4-2、Google 雲端硬碟公開發佈網站系統 62](#_Toc362083047)

[圖4-3、Facebook API 開發者教學網站 63](#_Toc362083048)

[圖4-4、OAuth 授權協定流程圖 64](#_Toc362083049)

[圖4-5、Facebook 應用程式之OAuth授權應用 – 使用者登入畫面 67](#_Toc362083050)

[圖4-6、Facebook 應用程式之OAuth授權應用 – 同意存取朋友資訊畫面 67](#_Toc362083051)

[圖4-7、Facebook 應用程式之OAuth授權應用 – 同意管理朋友名單畫面 68](#_Toc362083052)

[圖4-8、朋友分群系統 – 讀取朋友相關資訊畫面 68](#_Toc362083053)

[圖4-9、朋友分群系統 – 設定多組朋友互斥條件畫面 69](#_Toc362083054)

[圖4-10、朋友分群系統 – 針對所有分群結果進行調整畫面 69](#_Toc362083055)

[圖4-11、朋友分群系統 – 針對個別分群結果進行調整畫面 70](#_Toc362083056)

[圖4-12、朋友分群系統 – 詢問受測者是否要將調整結果建立朋友名單畫面 70](#_Toc362083057)

[圖4-13、朋友分群系統 – 輸入建立 Facebook 朋友名單之名稱畫面 71](#_Toc362083058)

[圖4-14、朋友分群系統 – 完成社群偵測結果調整與確認之圖表資訊畫面 72](#_Toc362083059)

[圖4-15、朋友分群系統 – 確認使用者皆有調整過每個群組之訊息畫面 72](#_Toc362083060)

[圖4-16、朋友分群系統 – 確保每個朋友皆必須設定至群組中之操作畫面 73](#_Toc362083061)

[圖4-17、朋友分群系統 – 將使用者回饋資訊回傳至Google文件之畫面 73](#_Toc362083062)

[圖5-1、Give Me My Data 應用程式畫面 74](#_Toc361062986)

[圖5-2、Excel 試算表之資料分析功能畫面 75](#_Toc361062987)

[圖5-3、實驗設計相關步驟示意圖 76](#_Toc361062989)

[圖5-4、受測者的回饋資訊圖表 78](#_Toc361062990)

[圖5-5、不同朋友數之群組的回饋資訊圖表 79](#_Toc361062991)

[圖5-6、受測者之朋友數分佈圖表 80](#_Toc361062992)

**表目錄**

[表2-1、簡單範例 – 簡化最佳化模組性之整數線性規劃問題 21](#_Toc361589699)

[表2-2、適用社群偵測之分群演算法 – 時間複雜度比較表 25](#_Toc361589700)

[表2-3、小規模真實社群網路之公開資料集 34](#_Toc361589701)

[表2-4、小規模真實社群網路之公開資料集 – 分群演算法比較表 35](#_Toc361589702)

[表3-1、空手道俱樂部的成員關係網路 – 不同分群演算法Q值比較表 58](#_Toc361070991)

[表4-1、系統開發與實作解決所面臨問題之系統需求說明表 59](#_Toc361068857)

[表4-2、Facebook 社群網站 Graph API 之相關存取權限 65](#_Toc361068858)

[表5-1、受測者實驗調查資訊 77](#_Toc361068871)

1. **緒論**

## 研究背景

根據 2012 年創市際市場研究顧問所公佈的「2012年網路社群白皮書」調查報告中可以得知在 2012 年 6 月至 7 月的台灣民眾使用社群網站比例相較於部落格、微網誌和即時通訊的使用比例高達 96%，其中使用社群網站的動機有 63% 為可以隨時知道親朋好友的感受 / 生活點滴 / 情緒和有 52% 為可以即時表達自己的感受 / 生活點滴 / 情緒，其中關注資訊類型更有 84% 為朋友近況 / 生活心情 / 情緒 [1]。然而現實生活中當在特定的情況下時，使用者會只想分享資訊給在社群網站中適當的朋友，像是透過一位名為艾德的大學生進行情境 (scenario) 的說明，假設情境中的人物皆有使用 Facebook 社群網站，同時在 Facebook 社群網站中也是艾德的朋友，此外艾德的父母與艾德的部份朋友也存在朋友的關係。艾德是一位瘋狂的滑板粉絲，艾德的所有朋友都認為玩滑板是件非常酷的事件，然而艾德的父母卻覺得這是一個非常危險的運動，因此禁止他玩滑板，即便如此艾德仍然迷上了滑板的運動，繼續在社群網路中與朋友分享有關滑板相關的圖片與影片內容。可是當艾德的父母在社群網站中看到艾德分享有關滑板相關的圖片與影片內容，就會造成衝突的發生，所以艾德為了避免衝突的發生，所以艾德分享的內容漸漸不再​​涉及滑板資訊，此時就會造成艾德使用Facebook 社群網站的動機降低 [2]。

圖1-1、Facebook 社群網站朋友名單的應用

(資料來源：本研究整理)

然而 Facebook 社群網站都有提供朋友名單的分類功能，為了能讓使用者將朋友分別加入至不同的自訂朋友名單，針對特定的朋友進行資訊的分享，此時透過朋友名單能夠只和對的人分享資訊，直接將最新動態發佈到那些對使用者貼文最感興趣的家人和朋友牆上，而不必擔心會對其他人造成干擾，以及直接看到使用者最在乎的朋友的相片和最新動態，略過那些只是點頭之交的消息 [3]，如圖1-1所示，像是艾德的父母只要不存在於朋友名單內，此時艾德在分享與滑板相關的的圖片與影片內容時，只需要選擇分享給朋友名單中的朋友，就能避免艾德的父母得知艾德分享所有與滑板相關的的圖片與影片內容。在 2007 年 Ellison 學者等人也提出社群網站中的使用者除了關心特定的資訊或內容只適用於特定觀眾 (particular audience)，同時也關心從上下文中能接觸到此資訊或內容的其它觀眾，此時以群組為基礎的存取控制 (group-based access control) 能有效防止在使用者的社群網絡中無意分享資訊給不適當對象之解決方案 [4]。



圖1-2、Facebook 社群網站朋友名單畫面

(資料來源：本研究整理)

可是當在分享資訊且在 Facebook 社群網站中使用者會有幾百位的朋友，在 2011 年 Ugander 學者等人研究發現在 Facebook 社群網站中的使用者平均有 190 位以上的朋友 [5]，這時若要在 Facebook 社群網站中將朋友加入至建立新建的自訂朋友名單至少需要五個步驟 [3]。

1. 前往您所需要新增至新名單之朋友的動態時報。
2. 在他們的封面相片上，點擊右下角的**「朋友」**選項。
3. 從出現的功能表中，點擊**「加到新名單……」**選項。
4. 向下捲動，然後點擊**「 + 新名單」**選項。
5. 在新名單中輸入名稱，然後按下**「確認」**鍵。

此外若要將朋友分別加入至不同的朋友名單，則需要個別手動去選擇適當的朋友加入至朋友名單中，如圖1-2所示，所以不論是建立自訂名單或將朋友分別加入至不同的朋友名單皆會增加使用者管理的成本 [6]，所以為了要簡化朋友名單管理，因此在 2011 年 Facebook 社群網站則提出智慧型清單的新功能，主要是根據與所選朋友在個人檔案資料上的共同資訊，如學校、工作或城市等基本資訊，自動更新朋友名單，可是許多使用者不會提供很詳細的屬性資訊，除此之外使用者若透過自訂名單，則會根據朋友名單中的朋友資訊，提供建議朋友給使用者快速加入朋友名單中，如圖1-3所示。此外更在 2012 年 Staddon 學者等人針對 Facebook 社群網站的隱私進行探討，在 1,075 位受測者中發現有30% 的回應對於社群網站的隱私會產生恐懼，像是受測者會關心人們知道了他不想讓他們知道的資訊 [7]。

因此在 2012 年 Facebook 社群網站更推出隱私捷徑的新功能，此時可直接將朋友名單應用到誰可以看得到使用者往後貼文的功能設定，如圖1-4 所示，透過此功能就可以避免讓陌生人窺探使用者日常生活一舉一動，造成個人資訊在 Facebook 社群網站中外洩的情況發生。



圖1-3、Facebook 社群網站智慧型清單畫面

(資料來源：本研究整理)



圖1-4、Facebook 社群網站隱私設定與工具畫面

(資料來源：本研究整理)

同時在 2012年 Liu 學者等人意識到朋友名單管理的問題提出 Friendlist Manager朋友名單管理工具，主要實作 Facebook 應用程式透過分群演算法產生建議朋友名單的自動分群機制，就能夠簡化使用者管理朋友名單 [8]。此外在過去有許多學者進行社群偵測的研究，提出許多不同類型的分群演算法，在 2012 年 Papadopoulos 學者等人說明社群偵測主要透過圖形分群演算法，並且分成五大分類進行比較分析 [9]。本研究主要比較能適用於大規模網路在最短時間內產生群組的多種分群演算法，其中以 BGLL 分群演算法最能在短時間內有效將不同規模社群網路中的節點分成最適當的群組 [10] [11]。

然而過往分群演算法主要是將相似性高的朋友歸為相同群組，卻未考量到朋友之間彼此衝突的情況，此時在社群網站的應用中當在分享資訊時會有一些特別情況的需求考量，必需將朋友先設定必須屬於不同群組後再進行分群，這時就需要針對使用者設定的互斥因子條件設定進行分群。在過往研究中許多學者發現在社群網站中和使用者相關的多個群組之間會有共同存在 (co-presence) 的關係，同時挑選20位 Facebook 使用者和 20位非Facebook 使用者以質性研究的方式進行訪談，發現即使是無憂無慮的非 Facebook 社群網站使用者也曾經嘗試管理日常生活中群組之間共同存在的關係，同時更發現 20 位 Facebook 使用者會針對群組之共同關係進行策略性的調整，主要就是為了避免衡突的發生。此外在日常生活中也常見社群網站需要管理群組的相關研究，像是在 2009年 Skeels 學者等人則是針對公司內員工在社群網路的關係進行探討，提出當員工在進行資訊分享時渴望有多階層存取控制 (multi-level access control) 的機制 (mechanism) 建立群組，以降低使用者操作時所造成的負擔 [12] [13]。

最後過去學者針對分群結果的評估，主要是以模組性 (modularity) 為主 [14]，以及針對每個社群網路已經先分出符合真實情況的群組或透過產生隨機網路的方式模擬社群網路的情況此時就能透過參數先定義群組的分布情況，可是卻很少有研究是直接透過使用者回饋的方式即時進行符合使用者回饋之需求的實際分析，在2008年學者提到分群結果評估最直接的方式就是由使用者直接評估，可是會浪費許多的時間成本，所以才會透過評估指標進行比較分析 [15]，此外近年來少有學者針對在真實生活中使用 Facebook 社群網站的使用者，透過系統直接取得個人網路 (ego-centric network) 立即進行分群，直接讓使用者進行每個群組的調整，最後即時取得使用者回饋的資訊進行分析與比較 [6][8][15]。

## 研究動機

在Facebook 社群網路允許使用者將朋友分至特定的朋友名單，雖然透過分群演算法能有效簡化朋友名單的管理 [8]，可是在現實生活中，當使用者在Facebook社群網站中分享資訊時，會需要區隔分享資訊給相同群組中不同使用者，避免發生不必要的衝突的情況發生 [12]。這時過往分群演算法主要是將相似性的朋友歸為相同群組，卻未考量到朋友之間彼此衝突的情況發生，然而在社群網站的應用中在某些特定情況下當分享資訊時，為了避免衝突的情況發生，此時會需要先將朋友先設定屬於不相同的群組之後再進行分群，也就需要考量可能會發生衝突的朋友為互斥因子條件設定之後，接著再透過分群將 Facebook 社群網路中之個人網路的朋友分成群組。

此外過往分群演算法主要是透過評估指標進行比較分析，卻沒有直接針對分群結果即時根據使用者回饋進行比較分析 [15]，此時將 Facebook 社群網路中之個人網路的朋友分成群組，若只透過評估指標是無法有效了解使用者對於自動分群機制產生的分群結果的程度探討，因此本研究的研究動機，分別為：

* + - 1. 分群演算法主要是將相似關係的朋友歸為相同的群組，可是卻未考量朋友之間彼此衝突情況發生。
      2. 分群演算法的群組評估主要是透過評估指標進行分析，可是卻未直接透過使用回饋資訊進行分析。

  2. 研究目的與貢獻

為了解決過往分群演算法並沒有考量朋友之間彼此衝突的問題，所以本研究主要比較能適用於不同規模社群網路且同時在最短時間內產生群組的多種分群演算法，其中以 BGLL 分群演算法最能在短時間內有效將不同規模社群網路中的節點分成最適當的群組，因此本研究主要將傳統 BGLL 分群演算法進行改良，讓使用者可以設定多組兩位朋友不能屬於相同群組的互斥因子條件，有別於過去分群演算法數值參數的設定，互斥因子條件設定更能直接符合使用者的需求，接著將 Facebook 社群網站中之個人網路的朋友分成適當的群組，提出考量互斥因子之自動分群機制。

本研究透過系統的實作，能讓使用者能在系統介面中設定哪兩位朋友必需屬於不同的群組，接著以考量互斥因子之自動分群機制將 Facebook 社群網站中之朋友分成適當的群組，根據使用者對應群組結果的回饋資訊，透過在資訊檢索領域中廣泛被應用的查準率與查全率進行分析比較 [15][16][17][18][19]。

總結在本研究中主要目的為依據朋友之間衝突關係，提出考量互斥因子之自動分群機制，接著透過考量互斥因子之自動分群機制產生的群組中有多少比例是相關的朋友資訊和有多少比例是準確的朋友資訊，最後本研究主要有二大貢獻：

1. 主要進行多種分群演算法的比較分析，最後選擇 BGLL 分群演算法進行改良，提出考量互斥因子之自動分群機制，相較於數值參數設定的方式，能夠更直接的根據需求進行互斥因子條件設定，並且透過系統的實作自動將朋友分成符合互斥因子條件設定適當的群組。
2. 此外透過實作系統更能取得使用者針對所有群組直接回饋資訊，再根據資訊檢索領域中的查準率與查全率進行比較分析，相較於直接透過評估指標分析群組對於社群結構的適當性，更能有效了解群組中有多少比例是相關的資訊和群組中有多少比例是準確的資訊。



   4. 章節簡介

本研究總共分成六個章節，其內容如下：

1. 緒論，主要會說明研究背景、研究動機、研究目的與研究貢獻。
2. 文獻探討，主要會介紹社會網路之社群偵測、Modularity – Q值、BGLL 分群演算法、社群網站之群組應用和評估分群結果，透過文獻的比較分析挑選最適合的分群演算法和評估分群結果的方式
3. 問題定義與解決機制，主要會說明本研究的問題定義，針對問題定義提出的考量互斥因子之自動分群機制，並且說明相關的做法，最後透過 1977年Zachary 學者整理的空手道俱樂部的成員關係網路進行範例驗證。
4. 系統開發與實作，主要會說明本研究如何透過系統實作如何解決實驗設計所會面臨的問題，取得使用者的同意透過朋友資訊建置出在 Facebook 社群網站中的個人網路，接著能讓使用者能在系統介面中設定哪兩位朋友必需屬於不同的群組的互斥因子條件設定，再透過考量互斥因子之自動分群機制將個人網路中的朋友分成適當的群組，最後受測者能針對所有群組進行直接的評估分析即時取得回饋的資訊。
5. 實驗與評估分析，主要透過資訊檢索領域中廣泛被應用的查準率與查全率針對受測者回饋的資訊進行比較分析，有效了解群組中有多少比例是相關的資訊和群組中有多少比例是準確的資訊。
6. 結論與未來建議，會對實驗結果進行統整，並說明在研究上面臨的限制以及未來可以延續研究的部分。
7. **文獻探討**

## 社群網路之社群偵測

社群網路之社群偵測 (community detection) 主要是源自於社群網路分析 (social network analysis) 中的凝聚子群 (cohesive subgroup) 的分析。社群網路分析主要是針對社群網路的關係結構和屬性進行分析的一套規範和理論方法，其中有許多不同領域的研究，像是凝聚子群的分析，主要是針對社群網路中某些行動者 (actor) 之間的關係 (relation) 會特別緊密，以至於凝聚成一個群組 (group) [20]。所謂群組在不同的研究領域會有不同的解釋與定義，包括子群 (subgroup)、社群 (community)、叢集 (cluster) 和模組 (module) 等解釋 [22]。

高中朋友

大學朋友

家庭成員

個人

(ego)

圖2-1、個人網路 (ego-centric network) 社群偵測示意圖

(資料來源：McAuley, J. et al. (2012) [6])

自從1930年開始社會網路分析基本上就會分成全域網路 (socio-centric network) 和個人網路 (ego-centric network) 這兩種方法進行分析 [21]，全域網路分析像是在2011年 Ugander 學者等人主要在臉書圖譜 (facebook graph) 中找出各別使用者的個人圖譜 (ego graph) 進行比較分析，總結出每位使用者平均有 190 位以上朋友且最大的朋友上限為5000位朋友 [5]。個人網路分析像是在 2012 年Leskovec 學者等人提到目前個人網路很龐大且雜亂，可是卻沒有適當的方式進行整理，雖然Google+ 有朋友圈 (friend circles) 和 Facebook 與 Twitter有朋友清單 (friend lists) ，可是使用者除了建立時費力之外，更需要在當個人網路中有朋友增加時進行動態更新，因此提出透過機器學習 (machine learning) 的方式進行分群識別出使用者的社交圈 (social circle)，如圖2-1所示，然而在使用者社群網路中偵測社交圈的問題就是針對個人網路 (ego-centric network) 進行分群的問題 [6]，然而本研究主要就是針對在 Facebook 社群網站中的個人網路進行社群偵測的分析與探討。

社群偵測 (community detection) 一般會採用圖形分群演算法解決此問題，在 2012 年 Papadopoulos學者等人將圖形分群演算法分成五大類別，分別為分裂 (divide)、基於模型 (model-based)、節點分群 (vertex clustering) 、子圖探索 (subgraph discovery)、和品質最佳化 (quality optimization)，如圖2-2所示[9]。透過社群偵測之分群演算法的分類，可以得知在過去已經有許多學者專門研究在如何在社群結構 (community structure) 中找出社群 (community) 並且透過圖形分群演算法進行社群偵測，然而分群問題在某些條件下是屬於 NP-complete 的問題 [23]，且早期分群演算法主要皆是採用啟發式演算法 (heuristics algorithm) [24]。

* Girvan and Newman (2002)
* Information centrality   
  (2004)
* Edge clustering coefficient  
  (2004)
* Max flow   
  (2005)
* Greedy optimization (2004)
* Spectral optimization  
  (2006)
* Exteral optimization   
  (2007)
* **BGLL  
  (2008)**
* Embedding in space + k-means
* Walktrap  
  (2005)
* k-core dectection (2003)
* **SCAN   
  (2007)**
* MCL   
  (2000)
* Infomap   
  (2008)
* **LPA  
  (2009)**

**社群偵測  
圖形分群**

**演算法**

**分裂**

**品質最佳化**

**節點分群**

**基於模型**

**子圖探索**

圖2-2、適用社群偵測之分群演算法之分類

(資料來源：Papadopoulos, S. et al. (2012) [9])

社群網路之社群偵測 [9][11]

分割與階層分群演算法 [11]

啟發式演算法 [24]

GN 分群演算法 [25]

Modularity Q值 [24][27][28][30][31][24]

最佳化模組性為主之  
相關演算法 [9]

圖形分群演算法  
 [9]

五大不同類型

一、分裂

三、節點分群

二、基於模型

四、子圖探索

五、品質最佳化

SCAN 分群演算法 (2007)

BGLL分群演算法 (2008)

LPA分群演算法 (2009)

BGLL 分群演算法適用至社群偵測中能產生適當的群組[9][10][11]

**章節**

**2.1**

社群網站之群組應用

[4][9][11][12][13][35][36][37][38][39]

評估分群結果

[14][15][16][17][18][19][46][47][48][49][50][51]

近似解演算法 [24][29]

**章節**

**2.2**

**章節**

**2.3**

**章節**

**2.4**

**章節**

**2.5**

適用於大規模網路

線性時間複雜度

最短時間找出

適當的群組

相關

圖2-3、文獻探討架構圖

(資料來源：本研究整理)

然而在 2012 年 Papadopoulos 學者等人已經很詳細的針對五大分類的圖形分群演算法進行比較分析，像是時間複雜度、空間複雜度和評估指標 [9]，所以本研究主要會從啟發式演算法探討至近似解的演算法，接著在進行五大分類的圖形分群演算法的時間複雜度與適用規模大小的比較，比較分析之後會以 BGLL 分群演算法為本研究考量互斥因子之自動分群機制主要改良的分群演算法。此外探討過往學者評估分群演算法產生群組適當性的方式，考量互斥因子之自動分群機制在社群網站之群組應用，再進行分群結果評估的比較分析，如圖 2-3 所示。

在 2012 年 Chaturvedi 學者等人先將分群分成分割分群方法 (partition clustering approach) 和階層分群方法 (hierarchical clustering approach)，接著進行不同類型的分群演算法的比較分析 [11]。然而不論是分割分群方法或是階層分群方法早期學者以啟發式演算法為主，像是在1970年Kernighan Lin所提出的分割分群方法的啟發式演算法或是在 2002 年Girvan 和Newman 學者認為在社群結構中的若在相同群組中的節點，彼此之間會有許多相互關係，代表群組中的邊很密集；若在不同群組中的節點，節點彼此之間相互關係相對較少，代表群組中的邊很稀疏，基於這樣的概念提出以階層分群方法的 GN 分群演算法 [25]，在 1977 年 Zachary 學者花費二年的時間觀察而整理出空手道俱樂部成員之關係網路，這是個複雜的網路、在社群網路分析等領域中最常用的小規模真實網路，且在現實生活中會因為衡突分成兩個不同的群組，如圖2-4所示 [26]。

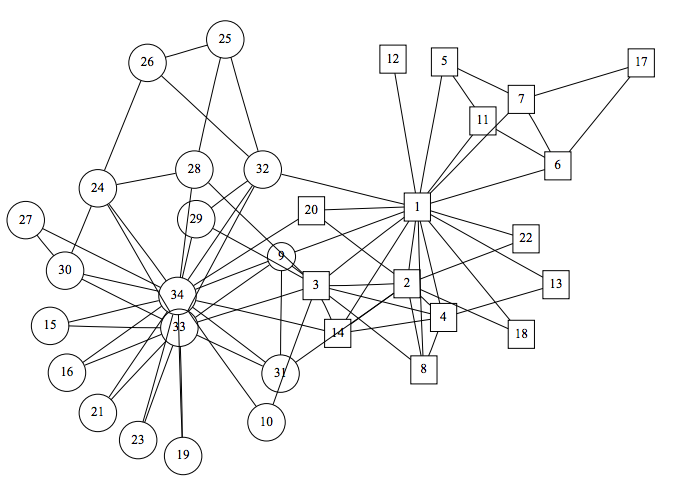


圖2-4、空手道俱樂部成員之關係網路

(資料來源：Girvan, M., & Newman, M. E. (2002) [25])

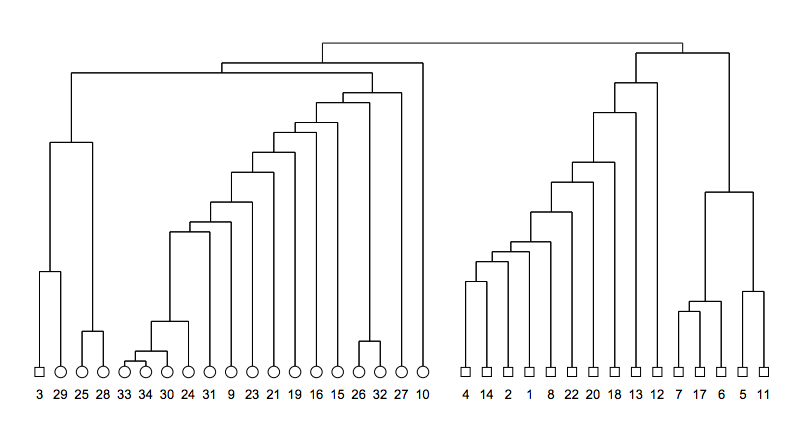


圖2-5、空手道俱樂部成員之關係網路 – 分裂階層圖

(資料來源：Girvan, M., & Newman, M. E. (2002) [25])

雖然透過 GN 分群演算法可以適用於許多不同種類的網路，從社群結構中找出不同的群組搭配，然而會面臨如何找出最適當的群組搭配，所以在 2004年Girvan學者等人定義衡量群組的適當性，稱之為模組性 (modularity)，定義 Q 值主要是計算出每個群組對於社群結構的適當性，當Q 值越大時，會使得群組之內的邊越密集和群組之間的邊越鬆散，則此圖中社群結構就會更加的明顯，代表群組對於社群結構最適當 [27]，並且在 2004 年 Newman 學者等人透過 Q 值改良GN 分群演算法，解決從多種群組搭配中找出最適當的群組搭配 [28]，在此之後只要是以最大化Q值為主的分群演算法即為最佳化模組性之啟發式演算法，像是 2004 年 Clauset 學者等人提出 CNM 分群演算法和 2008 年 Blondel 學者等人提出BGLL 分群演算法 [24]。



## Modularity – Q值

在2004 年 Newman 學者等人提出衡量群組適當性的 Q值，若將圖中的節點分成 *c* 個群組， 是群組 i中的邊所佔全部邊之比例，此時 Q 值為 *c* 個群組中的邊所佔全部邊之比例加總，如公式2-1所示 [27]。

|  |  |
| --- | --- |
|  | ( 2-1 ) |
|  |  |

可是當 Q 值越大時，群組分的就是最適當的，可是當只有一個群組時 Q 值為最大值，為了解決此問題，重新定義Q值的計算公式，如公式2-2 所示， 是所有與群組 *i* 中各個節點所有相連邊所佔全部邊之比例，其值範圍介於 -1 至 1 之間，定義中當Q 值越大時，會使得群組之內的邊密集和群組之間的邊更鬆散，則圖中社群結構更加的明顯，代表群組在社群結構中最適當 [27]，接著透過 Q 值改良GN 分群演算法，解決從多種群組搭配中找出對於社群結構最適當的群組搭配 [28]。

|  |  |
| --- | --- |
|  | ( 2-2 ) |

在 2004 年 Clauset 學者等人提出以貪婪演算法最大化Q值之 CNM 分群演算法，並且重新定義Q值的計算公式，圖中將節點*v* 和節點 *w* 分別為 和 兩個群組，當 = 時 ，當 時 ，接著當 = 1 時，節點*v* 與節點 *w*之間存在關係，當 = 0 時，則節點 *v* 與節點 *w*之間不存在關係，其中 *m* 是在圖中所有邊的總數， 與 分別為節點 *v* 與節點 *w* 的鄰居節點數，如公式2-3所示 [29]。

|  |  |
| --- | --- |
|  | ( 2-3 ) |

可是無法直接透過公式2-3解釋定義中當Q 值越大時，會使得群組內的邊密集和群組之間的邊更鬆散，則圖中社群結構更加的明顯，此時需將公式進行調整，如公式2-4所示。接著重新定義 ，其所代表的意義為群組*i*之內邊的比例，如公式2-5所示，以及重新定義 ，其所代表的意義為群組之間邊有一個節點屬於群組 *i* 的比例，如公式2-6所示，最後再透過公式2-5和公式2-6將公式2-4 導出Q值最早提出的公式2-2，如公式 2-7 所示，其所代表的意義為群組之內邊的比例減去群組之間邊的比例，當Q值越大越好，代表群組之內邊的比例要越大越好和群組之間邊的比例要越小越好，此時群組對於整個社群結構也會越適當。

|  |  |
| --- | --- |
|  | ( 2-4) |
|  |  |
|  | ( 2-5 ) |
|  |  |
|  | ( 2-6 ) |
|  |  |
|  | ( 2-7 ) |

接著本研究透過簡單範例進行說明，如圖2-6 所示，圖中有5個節點，6個邊，若將節點1和節點2分至群組1，表示式為 ，將節點3、節點4和節點5分至群組2，表示式為 ，此時 代表邊的節點同時在群組 i 和群組 *j* 的總和數除以2倍的邊數， ，， 和 ，以及 代表群組 i 中節點所連結邊數的總和除以2倍的邊數，，，則 ，當 Q 值為負時代表分群結果不適當，此時將節點3指定至群組1，則 ，當 Q 值為正時代表分群結果適當 ，如圖2-7所示[30]。

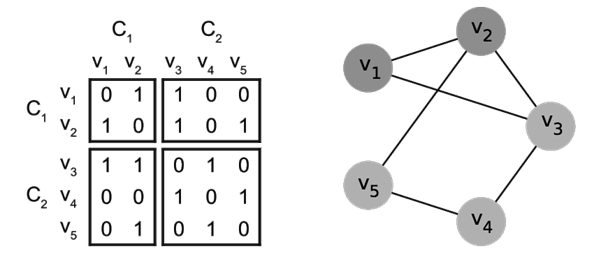


圖2-6、簡單範例 – 分群結果不適當

( 資料來源：Ovelgönne, M. et al. (2010) [30] )

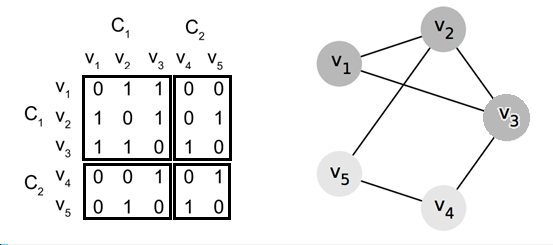


圖2-7、簡單範例 – 分群結果適當

( 資料來源：Ovelgönne, M. et al. (2010) [30] )

在2005年 Smyth 學者等人則以機率的概念探討群組適當性之Q值，將節點分割至*k*個群組之後，針對k個群組分別計算Q值。給予 *G ( V , E , W )* 由節點集合*V、*邊集合 *E*以及對稱的權重矩陣，*n* 為節點數所構成的無向圖 (undirected graph)，其中 若節點*i*和節點*j* 存在邊則 ，不存在邊則 [31]。

定義 是將節點分割至*k*個群組，= ， 主要是在群組*c*中邊的權重總和以及 主要是邊中至少有一節點在群組 *c*中的權重總和以及 主要是圖中所有邊的權重總和，則Q值計算公式，如公式2-8 所示。

|  |  |
| --- | --- |
|  | ( 2-8 ) |

因此Q值也能應用至權重圖之中，可是若要從機率的角度去探討Q值，則所有權重值皆為1，此時 為隨機選一條邊所連結的節點皆是在相同的群組 *c* 中的機率 ， 為隨機選一條邊所連結的節點中，至少有一個節點在群組 *c* 中的機率 ，因此 Q 值為所有群組*c*中隨機選一條邊所連結的節點皆是在相同的群組 和隨機選一條邊所連結的節點中至少有一個節點在群組之誤差值，當 值越大時代表群組對於社群結構越明顯。

在2008年 Brandes 學者等人則以覆蓋率的概念探討群組適當性之Q值 [24] ，若給予圖 *G = ( V , E )* 由節點*V、*邊 *E* 所構成的無向連接圖 (undirected connected graph)，定義 *n* = *|V|* 為節點中節點數，*m* = *|E|* 為節點中節點數*，* 為將節點分割成 *k* 個群組，其中 為*G*的子圖 (subgraph)。

接著定義 為群組內 (intra-cluster) 邊的集合， 為群組間 (inter-cluster) 邊的集合，以及 為邊的節點分別在群組 i與群組 *j* 的集合。最後定義 代表群組的模組性 (modularity) ，如公式2-9 和公式 2-10 所示。

|  |  |
| --- | --- |
|  | ( 2-9 ) |
|  |  |
|  | ( 2-10 ) |

此時進行Q值評估群組的適當性，需要最大化群組內邊的覆蓋率 ，以及最小化群組之間邊的覆蓋率，當 值越大時代表群組對於社群結構越明顯，如圖2-8所示。

群組之間邊的覆蓋率

群組之內邊的覆蓋率

圖2-8、評估群組的適當性示意圖

(資料來源：本研究整理)

　　在2008年 Brandes 學者等人證明最佳化模組性 (optimizing modularity) 為決定性問題時屬於NP-complete 的問題，同時將此問題之最佳化模組性轉換成整數線性規劃 (integer linear programming) 的數學問題進行探討 [24]。

若給予圖 *G = ( V , E )* 屬於無向圖，其中 *n = | V |* 節點數， *m = | E |* 邊數，定義 個決策變數 (decision variable) 為 為節點的所有配對，其中 *u , v*  *V* ， 0 代表節點*u* 和節點*v* 不屬於相同群組，1 代表節點*u* 和節點*v* 屬於相同群組，接著定義目標函數 (objective function) 為：

接著為了確保一致性 (consistency) ，所以需要定義限制式 (constraints)：

1. 反身性 (reflexivity)：
2. 對稱性 (symmetry)：
3. 遞移性 (transitivity)：

整理之後可將變數簡化成 個以及限制式簡化成 個

此時將最佳化模組性轉換成整數線性規劃的數學問題去解決圖2-6和圖2-7的簡單範例，圖有5個節點 (*n* = 5) ，6個邊 (*m* = 6)，目標函數為：

總共有 25個變數，接著因為 為節點的所有配對，其中 *u , v*  *V*，0 代表節點*u* 和節點*v* 不屬於相同群組，1 代表節點*u* 和節點*v* 屬於相同群組，此時相同的節點一定會是屬於相同群組，所以需要加入反身性 (reflexivity) 限制式 ，將會使得變數簡化為 20 個，其目標函數能簡化為：

再來因為圖屬於無向圖，當節點*u* 和節點*v* 屬於相同群組時，則 = 1 和 ，反之當節點*u* 和節點*v 不*屬於相同群組時，則 = 0 和 ，所以需要加入對稱性 (symmetry) 限制式 會使得變數簡化為：

此時變數已經化簡為 個，且因為 ，所以當 時，目標函數才會達到最大值 ，可是若根據 變數的定義，此時會產生邏輯的矛盾，當 和 時，代表節點1與節點2和節點2與節點5屬於相同節點，則節點1與節點5也會屬於相同的群組，可是 就產生矛盾了，因此還需要加入遞移性 (transitivity) 限制式，且因為對稱性的限制式，所以將遞移性的限制式化簡為 。

此時限制式已經化簡為 個，接著若以分支界定演算法 (branch-and-bound algorithm) 解此問題時，為了求最大值，且係數為負數，因此已知 ，接著當固定 時再進行 分支判斷時，帶入限制式條件進行判斷，只有當 時才能符合限制式條件。

表2-1、簡單範例 - 簡化最佳化模組性之整數線性規劃問題

|  |
| --- |
| maximize  subject to |

(資料來源：本研究整理)

因此在符合限制式件件的情況下，簡化最佳化模組性整數線性規劃之問題的最佳解為 ，如表2-1所示，若以模組性 (modularity) 為衝量群組適當性的評估指標，則就能最佳解之變數組合推論出的最佳的分群結果。

所以此範例的最佳解的變數分別為

根據對稱性的限制式，就能夠反推出

最後再根據反身性的限制條件，就能夠反推出

因此根據定義 為節點的所有配對，其中 *u , v*  *V*，0 代表節點*u* 和節點*v* 不屬於相同群組，1 代表節點*u* 和節點*v* 屬於相同群組最佳解的分群結果為將節點1、節點2和節點3分至群組1，表示式為 ，以及將節點4和節點5分至群組2，表示式為 ，如圖 2-7 所示。

雖然透過整數線性規劃的近似解演算法 (approximation algorithm) 能求得最精確的模組性最佳解 [24]，可是當網路中有*n* 個節點時至少需要 個變數和 個限制式，以空手道俱樂部為例，就需要 561 個變數和5984 個限制式，因此當進行社群網路求最佳解時，若採用分支界定法 (branch-and-bound algorithm) ，則時間複雜度為 ，雖然能精確的解出最大化Q值之最適當的分群結果，如圖2-9所示，可是其所需花費的時間，就時間複雜度來推斷會非常不適合應用至社群網路中進行社群偵測，所以許多學者針對社群網路中進行社群偵測的問題，提出以品質最佳化 (quality optimization) 方法為主的圖形分群演算法，主要就是透過最佳化模組性的方式找出最適當的群組，像是在 2004 年 Clauset學者等人提出以貪婪演算法最大化Q值之 CNM 分群演算法屬於最快速的最佳化模組性之近似解演算法，可是卻仍然無法解決模組性的解析限制 (resolution limit) 的問題，在 2006年 Fortunato學者等人提出模組性會有解析限制的問題，當網路規模越大時，此時若在大規模的網路中有兩個互斥群組之間存在單一連結邊，此時透過模組性進行評估則會判斷這兩個互斥群組有很強的相關性，此時會將兩個群組進行合併，因此在大規模網路時會找不到適當的小群組 [32]。

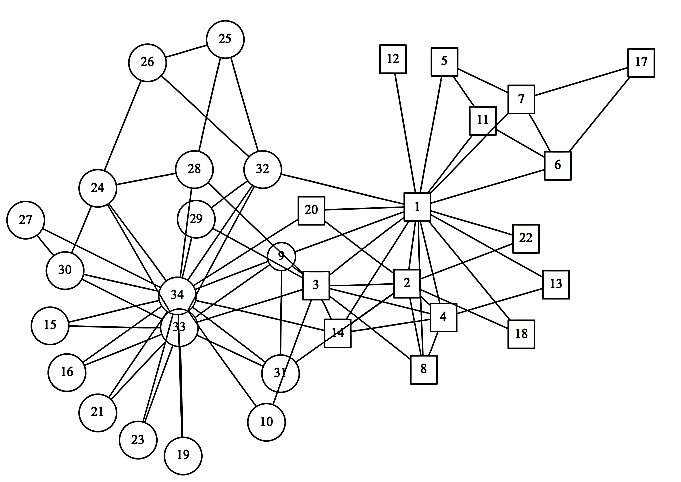


圖2 -9、最大化Q值之整數線性規劃應用至空手道俱樂部之分群結果

(資料來源：Brandes, U. (2008) [24])

## BGLL 分群演算法

由於模組性 (modularity) 存在解析限制 (resolution limit) 的問題，且不論是以啟發式或者近似解之最大化Q值的方法找出對於社群結構最適當的群組等分群演算法皆無法有效解決此問題，所以在 2008 年 Blondel 學者等人提出BGLL 分群演算法除了能解決模組性存在解析限制的問題，更能在不同規模網路中以最短的時間內找出最適當的群組，且時間複雜度接近線性時間，屬於最佳化模組性之啟發式演算法 [10]，在 2012 年 Papadopoulos 學者等人將應用至社群偵測之圖形分群演算法，主要分成五大類別，根據具代表性的分群演算法進行時間複雜度與網路規模大小的比較分析與整理，如表 2-2 所示。其中分裂 (divisive) 方法和節點分群 (vertex clustering) 方法的分群演算法之時間複雜度相對較高為第一個問題，以及沒有適當的方法能應用至大規模的社群結構有效找出適當的社群為第二個問題，基於上述二個問題，子圖探索 (graph discovery) 方法、基於模型 (model-based) 方法和品質最佳化 (quality optimization) 方法皆有提供能應用至大規模的社群結構在線性時間複雜度內有效找出適當的群組。接著從子圖探索、基於模型和品質最佳化這三大類別之中各選擇圖形分群演算法進行比較分析 ，分別為 2007年 Xu 學者等人提出的 SCAN 分群演算法、2008年 Blondel 等學者提出的BGLL分群演算法和 2009年 Leung 等學者提出的 LPA 分群演算法。

在2007 年Xu學者等人提出的SCAN 分群演算法，全名為 Structural Clustering Algorithm for Networks [33]，進行分群之前必需設定多項與密度相關的參數，讓分群結果更適當，可是針對不同規模的社群結構卻很難決定適當的參數值，因此在面對多變化關係的社群網路，SCAN 分群演算法會比較不適合，可是 SCAN 相較於以最佳模組性為主的分群演算法，更能透過基於結構計算節點之間相似性 (similarity) 以有效找出連接群組的節點 (hubs) 和群組邊際的節點 (outliers)。然而本研究提出以朋友互斥關係概念為基礎，設定哪些節點必須不能屬於相同群組的參數機制，讓分群結果更適當，相較於 SCAN 分群演算法多項與密度相關的參數，朋友互斥關係更容易設定適當的參數值，此外在本研究中主要探討針對Facebook 社群網路進行分群之後，探討群組中有多少比例是相關的朋友和有多少比例是準確的朋友為主，因此會以最佳化模組性的分群演算法為主  
，使得群組之內的邊越密集以及群組之間的邊越鬆散，所以會採用 BGLL 分群演算法。此外在2009 年Leung學者等人提出的 LPA 分群演算法的全名為 Label Propagation Algorithm [34]，在本研究中需要選出最大化Q值的分群演算法，針對個人的社群網路 (ego-centric network) 產生適當的分群結果，再進行使用者回饋的評估，可是LPA分群演算法針對相同的網路進行分群，每次產生的分群結果皆不一樣，因此 LPA分群演算法會比較不適合。

表2-2、適用社群偵測之分群演算法 – 時間複雜度比較表

|  |  |  |
| --- | --- | --- |
| **方法類型 (具代表性分群演算法)** | **時間複雜度** | **適用網路規模** |
| **分裂 (divide)** | | |
| Girvan and Newman (2002) |  | 小 (<) |
| Information centrality (2004) |  | 小 (<) |
| Edge clustering coefficient (2004) |  | 中 (<) |
| Max flow (2005) |  | 小 (<) |
| **節點分群 (vertex clustering)** | | |
| Embedding in space + k-means |  | 中 (<) |
| Walktrap (2005) |  | 中 (<) |
| **基於模型 (model-based)** | | |
| MCL (2000) |  | 中 (<) |
| Infomap (2008) |  | 大 (<) |
| **LPA (2009)** |  | **大 (<)** |
| **子圖探索** (**subgraph discovery)** | | |
| k-core dectection (2003) |  | 大 (<) |
| **SCAN (2007)** |  | **大 (<)** |
| **品質最佳化 (quality optimization)** | | |
| Greedy optimization (2004) |  | 中 (<) |
| Spectral optimization (2006) |  | 中 (<) |
| Extermal optimization (2007) |  | 中 (<) |
| **BGLL (2008)** |  | **大 (<)** |

(資料來源：Papadopoulos, S. et al. (2012) [9])

在 2008 年 Blondel 學者等人提出以階層分群方法為主的 BGLL 分群演算法，主要是由下至上聚合階層的方式找出在社群結構中最適當的群組 [10]，BGLL 分群演算法相較於 SCAN 分群演算法不需要設定多項很難決定的參數值，就能產生適當的分群結果，且若以 Q 值衡量群組的適當性，則 BGLL 分群演算法相較於 LPA 分群演算法，更能產生針對社群結果接近最適當群組之分群結果，所以本研究主要選擇BGLL分群演算法。BGLL 分群演算法基本上會分成兩個階段進行分群，第一個階段會先進行群組初始化，圖中的每個節點都會分配一個群組編號，使得每個節點皆是一個群組。隨機從圖中找出一個節點開始，依序進行直接所有的節點皆進行調整，接著對於任意節點 *i* 和節點 *j* ，當節點 *i* 加入到它的每個鄰居節點 *j* 所屬的群組時，計算Q值的變動，如公式 2-11，其中 是在群組中所有邊的權重和 是與該群組中所有節點有連結邊的權重合， 是節點 *i* 的鄰居節點邊的權重和， 為節點 *i* 與群組內連接邊的權重和，這公式需從兩個部份進行探討，公式中第一部份為 代表將節點 *i*在節點 *j* 所屬的群組中加入之後所計算的 Q 值，以及公式中第二部份為 代表將節點 *i* 從節點 *i* 所屬的群組中移除之後所計算的 Q 值，將第一部份計算的 Q 值減去第二部份計算的 Q 值，即可快速計算出 Q 值的變量。

|  |  |
| --- | --- |
|  | ( 2-11 ) |

當 為正時，從其中選出 最大值所對應鄰居節點，把節點*i* 加入到該鄰居節點所在的群組中，若所有 皆為負值，則節點 *i* 所屬的群組不變。重複進行群組的合併過程，直到不再出現群組合併的情況時，即找出了第一階段的群組。接著第二階段，首先建立一個全新的圖，該圖的節點為第一階段所找出的各個群組，該節點之間的連邊權重是兩個群組之間所有連結邊的權重和。接著再以第一階段中的演算法再次針對全新的圖找出群組，重複直到不再找出更高一層的社群結構為止，因此 BGLL 分群演算法能夠產生層次性的社群結構，且是一種自下而上的凝聚過程，如圖2-10所示。

最佳化模組性

modularity

optimization

3

6

7

3

11

13

10

9

8

15

14

12

2

1

0

4

5

6

7

11

13

10

9

8

15

14

12

2

1

0

4

5

社群聚合

community

aggregation

第一階段

第二階段

1

1

1

4

3

14

16

2

4

26

24

3

圖2-10、BGLL 分群演算法 – 兩階段聚合示意圖

(資料來源：Blondel, V. D. et al. (2008) [10])

此外 BGLL 分群演算法主要由兩階段聚合的方式，根據群組重構出新的圖，所以能有效解決模組性 (modularity) 之解析限制 (resolustion limit) 的問題，相關演算法如演算法1所示。將BGLL分群演算法轉換成流程圖如圖2-11所示，以及更重要的是時間複雜度為線性，更適用於大規模真實世界網路的群組偵測，如表2-3所示。

表2-3、不同規模社群網路之公開資料集 – BGLL和CNM分群演算法比較表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **不同規模之 真實社群網路** | **節點數** | **邊數** | **CNM (Q值)** | **CNM**  **(時間)** | **BGLL**  **(Q值)** | **BGLL**  **(時間)** |
| Karate | 34 | 77 | 0.381 | 0s | 0.418 | 0s |
| Arxiv | 9k | 24k | 0.772 | 3.6s | 0.813 | 0s |
| Internet | 70k | 351k | 0.692 | 799s | 0.781 | 1s |
| Web nd.edu | 325k | 1M | 0.927 | 5034s | 0.935 | 3s |

(資料來源：Blondel, V. D. et al. (2008) [10])

第一階段

Q 值是否沒改變?

第二階段

否

是

初始化

透過所有節點逐步移至相鄰節點的群組中計算個別的 ，最後選擇個別的 > 0，將該節點加入 最大值的群組。

Q 值是否沒改變?

否

是

第一階段

基於第一階段偵測出  
的群組建立新的網路

圖2-11、BGLL 分群演算法 – 流程圖

(資料來源：Blondel, V. D. et al. (2008) [10])

在2012 年學者為了解決在 Facebook 社群網站中大量資訊和隱私權限的問題進行朋友名單簡化的管理，提出Friendlist Manager管理工具，主要是針對使用者朋友之間的社群網路關係，偵測出群組以建立新的朋友名單。主要進行CNM 和 BGLL 分群演算法的比較分析，主要以 Facebook New Orleans Network之全域網路的資料集進行比較分析 [19]。最後因為 BGLL 分群演算法相較於其它分群演算法更能找出較小的群組，所以能夠將 Facebook 社群網站中之朋友分成更適當的群組 [8]。此外在 2012 年 Chaturvedi 學者等人先將分群分成階層分群方法 (hierarchical clustering approach) 和分割分群方法 (partition clustering approach) ，接著進行不同類型的分群演算法的比較 [11]。從 1970 年 Kernighan Li 學者提出的啟發式演算法解決圖分割的問題 (graph partitioning problem) 至 2008 年Blondel學者等人提出最佳化模組性之啟發式演算法，主要以階層分群方法為主的 BGLL 分群演算法等多種不同種類的分群演算法，總結出因為 BGLL 分群演算法能在線性的時間產生最適當的群組，所以在複雜的社群網路中進行社群偵測時，BGLL 分群演算法將會是最佳的選擇 [11]。

總結為了解決將使用者回饋的多組兩位朋友屬於不同的群組設定，本研究所提出的考量互斥因子之自動分群機制會將 BGLL 分群演算法進行改良，主要考量 BGLL 分群演算法之四個特性。

1. 非重疊分群演算法，為了要能將使用者設定的兩位朋友屬於不同的群組，避免衝突發生，關鍵在於兩位朋友所屬於的群組必需是互斥，因此需要選擇符合群組之間有互斥特性的非重疊分群演算法。
2. 找出最適當的群組，為了要能將使用者設定的兩位朋友屬於不同的群組，除了兩個人屬於的群組彼此互斥，同時也是適當的群組，所以主要選擇以模組性為理論基礎的分群演算法，確保在社群結構中群組之內的邊越密集和群組之間的邊越鬆散，代表群組會越適當。
3. 階層分群方法，聚合階層的分群方式能適用於任何規模的社群結構，除了能有效解決模組性應用在大規模社群網路中找不到最適當群組之解析限制的問題，更能考量使用者設定的兩位朋友屬於不同的群組的多組互斥因子條件設定，所以分群演算法最好是透過階層式進行分群，直到在符合所有組合的條件搭配情況下找出適當的群組。
4. 時間複雜度為線性時間，為了要能將使用者設定的兩位朋友屬於不同的群組，在執行效能為最好且最短的時間，在考量使用者設定多組互斥因子條件設定的情況下，有效將 Facebook 社群網站中之朋友分成適當的群組。

|  |  |
| --- | --- |
| **演算法1、BGLL 分群演算法** | |
| **輸入:**  **輸出:** | g 為個人網路  c 為分群群組 |
| while *g can be contracted* do  // 第一階段  forall the *vertices* vi do  c[vi] ← *i* // 初始化  end  while totalIncrease > 0 do  forall the *vertices v* do  best ← v  maxDeltaQ ← 0  totalIncrease ← 0  forall the *neighbors w of v* do  deltaQ ← deltaQ from c[v] to c[w]  if deltaQ > maxDeltaQ then  maxDeltaQ ← deltaQ  best ← w  end  end  c[v] ←c[best]  totalIncrease ← totalIncrease + maxDeltaQ  end  end  // 第二階段  g ← contract(g, c[])  end  return c | |

(資料來源：Blondel, V. D. et al. (2008) [10])

## 社群網站之群組應用

在 2012 年 Nadkarni 學者等人主要探討為何使用者要使用社群網站的動機，透過篩選出 42 篇以證據為基礎與Facebook 使用因素相關的論文進行比較分析，總結出使用者要使用社群網站的動機，主要有歸屬感的需要 (the need to belong) 和為了自我呈現的需要 (the need for self-presentation) [36]。在 2007 年DiMicco 學者等人發現使用者會在Facebook 社群網站中維持真實生活中不同方面之自我呈現以在適當的場合符合最適當的形象 [37]，且 Ellison 學者等人也提出社群網站中的使用者除了關心特定的資訊或內容只適用於特定觀眾 (particular audience)，同時也關心從上下文中能接觸到此資訊或內容的其它觀眾，此時以群組為基礎的存取控制 (group-based access control) 能有效防止在使用者的社群網絡中無意分享資訊給不適當對象之解決方案 [4]。

**在 2009 年Lampinen 學者等人發現使用者在社群網站中的多個群組之間會有共同存在 (co-presence) 的關係，挑選20位 Facebook 使用者、20位非Facebook 使用者以質性研究的方式進行訪談，發現即使是無憂無慮的非 Facebook 社群網站使用者也曾經嘗試管理日常生活中群組之間共同存在的關係，同時更發現20位 Facebook 使用者會針對群組之共同關係進行策略的調整，主要就是為了避免衡突的發生** [12]。此外在日常生活中也常見社群網站需要管理群組的相關研究，像是在 2009年 Skeels 學者等人則是針對公司內員工在社群網路的關係進行探討，提出當員工在進行資訊分享時渴望有多階層存取控制 (multi-level access control) 的機制 (mechanism) 建立群組，以降低使用者操作時所造成的負擔 [13]。且在 2012 年 Wilson學者等人整理出 412 篇與社群網路相關的文獻，其中有 27% 的 112 篇文獻在探討個人與群組之間的關係，像是學生與教師，員工的管理，企業與客戶，醫生與患者，以及合作夥伴等關係進行群組應用的探討 [38]，且在 2013 年 Das 學者等人更提出透過在社群網站中的自我審查 (self-censorship) 的策略，又可解釋為邊界調節 (boundary regulation) 策略能管理群組共同存在的關係。此外在社群網站中的自我呈現和自我審查是相輔相成，且當意識到潛在隱私問題的使用者，透過社群網站工具，像是 Facebook 隱私設定，避免內容撤回和允許更少訊息警惕，因此能降低自我審查所造成操作的困擾 [39]，最後整理成社群網站之群組應用的關係圖，如圖2-12所示。

Facebook 隱私設定

使用者

自我呈現

(Self-presentation)

以群組為基礎  
存取控制

群組

自我審查

(Self-censorship)

群組共存關係  
(Group Co-presence)

關心特定的資訊或內容只適用於  
特定觀眾

降低 [39]

產生

朋友名單

[2][3]

管理

[13]

產生

動機[36]

Facebook 社群網站

使用[37]

產生

朋友名單

[2][3]

管理[39]

影響

[12][38]

影響

[39]

管理

[4]

管理 [2][3]

圖2-12、社群網站之群組應用的關係圖

(資料來源：本研究整理)

當在使用者關心特定的資訊或內容只適用特定觀眾時，會針對社群網路中的群組進行管理 [4]，使用者透過自動分群機制產生特定觀眾的群組就能直接套用至特定的資訊或內容，此外不同群組之間會存在共存關係，只有當使用者關心特定的資訊或內容只適用特定觀眾時才會有相關需求 [38]，這時自動分群機制根據使用者回饋的資訊進行分群就能產生更符合使用者需求的群組，如圖2-13所示。

Facebook 隱私設定

使用者

自我呈現

(Self-presentation)

以群組為基礎  
存取控制

群組

自我審查

(Self-censorship)

群組共存關係  
(Group Co-presence)

關心特定的資訊或內容只適用於  
特定觀眾

降低 [39]

產生

朋友名單

[2][3]

管理

[13]

產生

動機[36]

Facebook 社群網站

使用[37]

產生

朋友名單

[2][3]

管理 [39]

影響

[12][38]

影響

[39]

管理

[4]

管理 [2][3]

**互斥因子條件設定**

**考量互斥因子之自動分群機制**

圖2-13、考量互斥因子之自動分群機制之社群網站應用

(資料來源：本研究整理)

## 評估分群結果

　　許多學者在分析分群演算法，皆是針對過去探討分群結果的公開網路資料集[11][14] [24][25][28][35]，其中與社群網路相關最具代表性的小規模公開網路資料集，如表2-3所示，像是在1977年 Zachary 花了二年時間觀察空手道俱樂部中成員彼此之間的關係，整理出與人相關的小規模真實社群網路資料集 (karate)。

表2-3、小規模真實社群網路之公開資料集

|  |  |  |  |
| --- | --- | --- | --- |
| **小規模真實社群網路** | **節點數** | **邊數** | **說明** |
| Karate | 34 | 78 | Zachary’s karate club [26] |
| Dolphin | 62 | 159 | Dolphin social network [40] |
| Football | 115 | 613 | American college football [28] |
| Jazz | 198 | 2742 | Jazz musicians network [41] |
| Email | 1133 | 5451 | Email network of human interactions [42] |

(資料來源：Agarwal, G et al. (2008) [14]、Jin, D. et al. (2012) [35])

透過小規模真實社群網路資料集解釋確認分群演算法，最後再透過透過評估指標Q值評估分群結果，如表2-4所示，像是在 2008 年Brandes 學者等人根據證明最大化模組性為NP-complete 問題，且提出整數線性規劃的分群方法，主要挑選具代表性的小規模真實社群網路公開資料集，其中就包括與人相關的小規模真實社群網路資料集 (karate) 和有關比賽的小規模社網路資料集 (football) ，除了確認改良的分群演算法，更透過評估指標Q值進行表2-6中所有小規模真實社群網路資料集的比較分析與探討 [24]。

此外在 2012 年學者等人將 BGLL 分群演算法應用至社群網路，主要挑選具代表性的小規模真實社群網路資料集，其中就包括有關動物的小規模真實社群網路資料集 (dolphin) 和有關比賽的小規模社網路資料集 (football)，透過評估指標Q值等資訊進行比較分析與探討 [11]。

表2-4、小規模真實社群網路之公開資料集 – 分群演算法之Q值比較表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **小規模真實社群網路** | **節點數** | **邊數** | **GN** | **CNM** | **LPA** | **BGLL** | **最佳** |
| Karate | 34 | 78 | 0.401 | 0.381 | 0.365 | 0.419 | 0.420 |
| Dolphin | 62 | 159 | 0.520 | 0.510 | 0.480 | 0.527 | 0.531 |
| Football | 115 | 613 | 0.601 | 0.549 | 0.587 | 0.605 | 0.606 |
| Jazz | 198 | 2742 | 0.405 | 0.439 | 0.342 | 0.443 | 0.446 |
| Email | 1133 | 5451 | 0.532 | 0.494 | 0.371 | 0.541 | 0.579 |

(資料來源：Agarwal, G et al. (2008) [14]、Jin, D. et al. (2012) [35])

近年來除了針對小規模真實社群網路資料集，透過 Q 值評估指標進行比較分析，更有學者針對大規模真實世界網路進行不同的問題進行比較分析時，會以Q 值評估指標進行比較分析 [5]，像是在 2012 年Ferrara 學者等人針對大規模真實世界網路資料集進行抽樣方式的探討，主要就是基於最大化模組性 (maximize modularity ) 挑選最接近線性時間複雜度之 BGLL 和 LPA 等分群演算法進行選擇 [43]。雖然最後在考量社群網站中會存在重疊關係的情況，根據 Gregory 學者等人提出能解決重疊群組之改良 LPA分群演算法的 COPRA 分群演算法，因此選擇 LPA分群演算法 [44]，但是在 2012 年 Wu 學者等人提出能解決重疊群組之改良BGLL 分群演算法的 BGLL- CONA 分群演算法，且針對大規模真實世界網路進行分群，不論是執行時間和群組適當性皆比COPRA分群演算法更佳 [45]。

可是評估指標 Q 值主要針對群組對於社群結構衡量群組的適當性，仍有些問題，在 2006年 Fortunato 學者等人提出模組性會有解析限制的問題，當網路規模越大時，此時若在大規模的網路中有兩個互斥群組之間存在單一連結邊，此時透過模組性進行評估則會判斷這兩個互斥群組有很強的相關性，此時會將兩個群組進行合併，因此在大規模網路時會找不到適當的小群組，因此在 2008 年 Lancichinetti學者等人提出能產生符合冪次法則分佈 (power law distribution) 的 LFR 基準圖 (benchmark graphs)，以能符合真實世界網路，如圖2-12所示，且透過不同參數的設定能產生不同規模與類型的 LFR 基準圖，主要透過 NMI 評估指標進行分群演算法之分群結果的衡量 [46]。接著在 2009 年Fortunato學者等人以產生四種不同規模與類型的 LFR 基準圖，分別為：

1. 規模大小為1000個節點，每個群組是由10至50個節點所組成。
2. 規模大小為1000個節點，每個群組是由20至100個節點所組成。
3. 規模大小為5000個節點，每個群組是由10至50個節點所組成。
4. 規模大小為5000個節點，每個群組是由20至100個節點所組成。

其中規模大小 5000 個節點剛好就是 Facebook 社群網站中使用者的最大朋友數 [5]，接著進行多種分群演算法的比較分析，其中BGLL 分群演算法透過 NMI 評估指標相較於其它分群演算法更能在不同規模與種類的LFR 基準圖中有效產生最適當的群組 [47]。

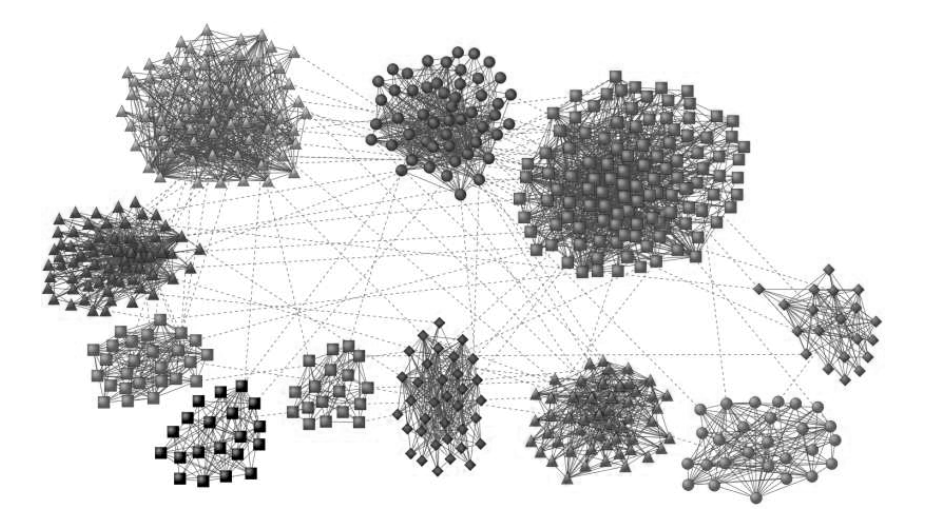


圖2-14、LFR 基準圖之示意圖

(資料來源：Lancichinetti, A. et al. (2008) [46])

　　在 2008 年 Manning 學者等人介紹分群應用至資訊檢索 (information retrieval) 中，且將評估分群的質量標準 (criterion of quality) 分成內部 (internal) 和外部 (external) 兩種，所謂內部品質標準主要的目標在於達到越高的群組之內 (intra-cluster) 相似性越低的群組之間 (inter-cluster) 相似性，模組性 (modularity) 之評估指標 Q值就屬於內部質量標準 (internal criterion of quality)，可是評估指標 Q值越高不一定代表在實際應用時最符合真實的情況，此時內部質量標準的另一種方法就是直接針對分群結果中之群組進行評估，雖然這是最直接的評估，可是若要直接請大量的使用者分別針對分群結果中之群組進行判斷評估，此時研究所花費的時間的成本將會是非常昂貴。此時為了替代使用者分別針對分群結果中之群組進行判斷評估，因此才會有針對基準 (benchmark) 透過外部質量標準 (external criterion of quality) 進行評估，像是 LRF 基準圖透過 NMI 評估指標進行評估，其中 LRF 基準圖為基準和NMI 評估指標為外部質量標準 [15]。

在 2010 年Steinhaeuser 學者等人主要比較應用於評估分群結果之各個群組的不同評估指標，其中具代表性的評估指標有三種，分別為芮氏指標 (rand index, RI)、調整後芮氏指標 (adjusted rand index, ARI) 和標準化共同資訊量 (normalized mutual information, NMI) [48]。

然而本研究主要會採用資訊檢索 (information retrieval) 領域中最常被應用於成效評估指標查準率 (precision) 和查全率 (recall) 解釋回傳的結果是不是使用者需要的資訊和回傳的效率之問題，查準率如公式 2-13 所示和查全率如公式 2-14 所示，若以文件檢索的為例，則查準率為取得相關文件 (retrieved relevant documents) 除以取得文件 (retrieved documents) 和查全率為取得相關文件 (retrieved relevant documents) 除以相關文件 (relevant documents) [16]。

|  |  |
| --- | --- |
|  | ( 2-13 ) |
|  |  |
|  | ( 2-14 ) |

此時本研究主要則是以系統判斷與人工判斷交叉計算出資訊檢索領域中廣泛應用的查準率與查全率進行分析，如表2-6所示。

表2-6、查準率和查全率之系統判斷與人工判斷 - 交叉分析表

|  |  |  |
| --- | --- | --- |
| **人工判斷**  **系統判斷** | **相關** | **非相關** |
| **相關** | 正確正例 (true positive, TP) | 錯誤正例 (false positive, FP) |
| **非相關** | 錯誤負例  (false negative, FN) | 正確負例  (true negative, TN) |

(資料來源：van Rijsbergen (1977) [16] )

簡單來說，人工判斷朋友屬於群組A和系統判斷朋友屬於群組B，此時查準率就是群組B中有多少比例的朋友同時屬於群組A和群組B，如圖2-15所示，以及查全率就是群組A中有多少比例的朋友同時屬於群組A和群組B中，如圖2-16所示，此外在 2012 年 McAuley 學者等人針對在社群網站中進行社群偵測的實驗結果發現查全率會高於查準率，其中查全率的定義是已建立的社交圈中有多少比例是相關的朋友和查準率是在社交圈建立且系統附加朋友之後有多少比例是準確的朋友 [6]。

正確負例

(TN)

錯誤負例

(FN)

錯誤正例

(FP)

正確正例

(TP)

系統判斷群組之相關朋友 (群組B)

人工判斷群組之相關朋友 (群組A)

**查準率 =**

圖2-15、系統判斷群組與人工判斷群組 – 查準率示意圖

(資料來源：本研究整理)

正確負例

(TN)

錯誤負例

(FN)

錯誤正例

(FP)

正確正例

(TP)

系統判斷群組之相關朋友 (群組B)

人工判斷群之相關朋友 (群組A)

組

**查全率 =**

圖2-16、系統判斷群組與人工判斷群組 – 查全率示意圖

(資料來源：本研究整理)

在 1971 年 Rand 學者等人提出客觀標準 (objective criteria) 之芮氏指標 (rand index, RI) 評估分群的方法，將統計調查的方式應用於測量兩個群組之間的相似性 (similarity) 程度，其值範圍介於 0 至 1 之間， 0 代表系統判斷的群組和人工判斷的群組完全不相同， 1 代表系統判斷的群組和人工判斷的群組完全相同[49]，若透過表2-6表示則如公式 2-15 所示。

|  |  |
| --- | --- |
|  | ( 2-15 ) |

在 1985 年 Lawrence 學者等人基於芮氏指標評估分群的方法進行調整，提出調整後的芮氏指標 ( adjusted rand index, ARI )，其值範圍介於 -1 至 1 之間， -1 代表系統判斷的群組和人工判斷的群組完全不相同， 1 代表系統判斷的群組和人工判斷的群組完全相同[50]，若透過表2-6表示則如公式 2-16 所示。

|  |  |
| --- | --- |
|  | ( 2-16 ) |

　　因此只需要透過正確正例、錯誤正例、錯誤負例和正確負例的概念就能計算出常用於評估分群結果的評估指標，像是芮氏指標和調整後芮氏指標。此外在 2003 年 Fred 學者等人基於資訊理論 (information theory) 提出標準化共同資訊量 (normalized mutual information, NMI) 評估指標衡量兩個群組之間的一致性 (consistency) 程度 [51]，是目前最常使用於隨機網路判斷的評估指標，其中BGLL 分群演算法透過 NMI 評估指標相較於其它分群演算法更能在不同規模與種類的LFR 基準圖中有效產生最適當的群組 [47]，所以這也是本研究提出互斥因子之自動分群機制將傳統的 BGLL 分群演算法進行改良的原因之一。

然而雖然在過往的研究中主要也會透過查準率和查全率的調和平均數之 F 度量 (f-measure) 評估分群結果，其值範圍介於 0 至 1 之間，且相較於芮氏指標、調整後芮氏指標和標準化共同資訊量，F 度量之通式 (general formula) 可以透過 參數能控制查準率與查全率的權重，當 參數為 2 時代表強調查全率和當 參數為 時代表強調查準率，如公式 2-17 所示 [15]，接著過往學者若透過 F 度量評估分群結果時，則其 參數主要會以 1 為主，代入通式調整之後，如公式 2-18 所示 [6][18]，此外若透過表2-6 表示則如公式 2-19 所示。

|  |  |
| --- | --- |
|  | ( 2-17 ) |

|  |  |
| --- | --- |
|  | ( 2-18 ) |

|  |  |
| --- | --- |
|  | ( 2-19 ) |

可是 F 度量無法直接解釋分群結果之各別群組中有多少比例的朋友資訊是符合使用者需求和分群結果之各別群組中有多少比例的朋友資訊是相關的關係，此時查準率能直接解釋群組中有多少比例的朋友資訊是符合使用者需求呈正比的關係，以及查全率能直接解釋群組中有多少比例的朋友資訊是相關，呈正比的關係。此外在分群相關的研究中還是有少數學者以查準率和查全率評估分群結果，像是 Chen 學者等人針對小規模真實社群網路，Mislove 學者等人針對大規模的Facebook社群網路，以及Grob 學者等人針對行動社群網路進行分群，其中皆有透過查準率和查全率進行比較分析 [17][18][19]，所以本研究將以在資訊檢索領域中的評估指標查準率和查全率針對受測者對於考量互斥因子之自動分群機制產生所有群組之回饋資訊進行檢索程度的評估，如圖2-17所示。

總結在2008年Manning 學者等人介紹分群應用至資訊檢索中提到直接針對分群結果中之群組進行評估，雖然這是最直接的評估，可是若要直接請大量的使用者分別針對分群結果中之群組進行判斷評估，此時研究所花費的時間的成本將會是非常昂貴 [15]。且在 2010 年 O'Neill 學者等人提出在社群網路中基於群組隱私控制之結構網路分群探討，為了取得參與者在 Facebook 的朋友相關資訊進行分析與探討，所以必需要透過客制化的應用程式系統取得資訊，且將相關朋友資訊轉成參與者朋友們的社群網路，以分群演算法依據社群結構自動將朋友分成不同的群組。接著再請使用者透過另一套軟體手動將朋友根據隱私分享的相似程度分成不同的群組，最後進行30分鐘的面試，根據使用者回饋的資訊，進行分析結果的驗證 [52]。最後在 2012年Liu學者等人提出一個系統工具主要目的在於能有效簡化朋友名單的管理，且主要應用BGLL分群演算法進行分群，產生群組，接著將相關的使用者回饋資訊回傳至伺服器端進行比較分析 [8]。

所以本研究為了要評估互斥因子之自動分群機制所產生群組，主要是讓使用者分別進行調整之後，產生使用者回饋的資訊，再透過資訊檢索中的查準率與查全率進行比較分析。可是Manning 學者等人提到直接請大量的使用者分別針對分群結果中之群組進行判斷評估，此時研究所花費的時間的成本將會是非常昂貴，所以本研究的評估方式會有實作系統的需求。此外 O'Neill 學者等人主要是透過至少兩套軟體分別先取得參與者在 Facebook 社群網站中朋友相關資訊，再讓參與者透過朋友相關資訊進行手動分群，此時對於參與者會造成許多的困擾，所以降低參與者的困擾就是本研究實作系統之主要目的。最後 Liu 學者等人已經提出網站系統的工具主要目的在於能有效簡化朋友名單的管理，並且將相關的使用者回饋資訊回傳至伺服器端進行比較分析，因此除了考量參與者操作的便利性之外，更需要即時接收使用者回饋的資訊進行比較分析。

評估分群結果 [15] [48]

內部評估指標 [15] [48]

芮氏指標 (RI) [49]

外部評估指標 [15][48]

模組性 (modularity) [27][14][35]

調整後芮氏指標 (ARI) [50]

標準化共同資訊量 (NMI) [51]

產生大規模隨機網路 [46][47]

小規模真實社群網路 [25][26][40][41][42]

**雖然使用者直接進行群組評估將會更適當，但時間成本過高 [15]**

正確正例 TP

錯誤正例 FP

錯誤負例 FN

正確負例 TN

產生Facebook 社群網路中之

個人網路 (ego network) [6]

資訊檢索中查準率和查全率

[16][17][18]

使用者直接進行所有群組的評估

使用者回饋資訊設定–互斥因子

考量互斥因子之自動分群機制

F度量 (F-measure) [15]

相關

關

計算

**透過系統的實作能夠簡化使用者直接進行群組評估的操作，以降低時間的成本**

圖2-17、評估分群結果之架構圖

(資料來源：本研究整理)

# **問題定義與解決機制**

## 問題定義

若將使用者在 Facebook 社群網站中之個人網路 (ego-centric network) 當成圖 (graph) ，其中節點 (vector) 代表朋友，邊 (edge) 代表關係，在過去有許多學者研究如何將圖中的節點，透過節點與節點形成的邊，將所有的節點分成不同的群組，稱之為社群偵測 [9]，然而本研究要解的問題為依據圖 (graph) 中節點 (vector) 間之邊 (edge) 的關係，在考量互斥因子的條件下，將節點分成符合條件且最適當的群組 [27]。

若給予圖*G* = ( *V , E , F* ) ，定義圖 *G* 為使用者在 Facebook 社群網站中朋友關係所形成之個人網路 (ego-centric network)，定義 是使用者之朋友集合，*n* 為朋友個數，定義 是朋友關係集合， *i* = *1*…*n* ， *j =* *1*…*n* ， 值的範圍為 0 和 1 ，當 為 0 代表 與 不相鄰，則朋友*i* 和朋友*j*不存在朋友關係，當 為 1 代表 與 相鄰，則朋友*i* 和朋友 *j*存在朋友關係， *m* 為朋友關係集合中所有元素的加總，定義 是互斥因子條件集合，*x* 為互斥因子條件數，其中每個互斥因子條件為，， 和 值的範圍為 1 至 *n* ，分別代表對應的節點，且 與 ，，*x* 為互斥因子條件數。

因此本研究的問題定義為給予圖 *G* = ( *V , E , F* ) 為使用者在 Facebook 社群網站中朋友關係所形成之個人網路 (ego-centric network)，依朋友關係集合 *E* 的關係找出朋友集合*V*中，所有與該朋友相鄰的朋友，再以朋友之間彼此相似的程度，以及使用者所提供互斥因子條件集*F*。**試求 之最佳解**，其中朋友 *i* 和朋友 *j* 分別屬於 和 兩個群組，當 = 時 ，當 時 ，接著當 = 1 時，則朋友 *i* 與朋友 *j* 之間存在朋友關係，當 = 0 時，則朋友 *i* 與朋友 *j*之間不存在朋友關係， *m* 是在朋友關係集合中所有元素的加總， 與 分別為朋友 *i* 與朋友 *j* 的朋友數，並且**必須滿足互斥因子條件集合 *F* 中的 *x* 個互斥因子條件，使得 ，*x，x* 為互斥因子條件數**。



## 做法說明

　　在第二章文獻探討中除了將 BGLL分群演算法與SCAN 分群演算法與LPA 分群演算法進行比較分析，更考量了BGLL分群演算法的四個特性，一、非重疊分群演算法。二、找出最適當的群組。三、聚合階層式分群。四、時間複雜度為線性時間。因此本研究提出考量互斥因子之自動分群機制以解決3.1的問題定義，主要是改良BGLL分群演算法，使其能滿足互斥因子條件之多組兩位朋友不能屬於相同群組的判斷。

　　首先 BGLL 分群演算法一開始主要是先將每個節點當成獨立的群組，接著隨機從的網路中找一個節點開始，每個節點再與相鄰節點所屬於的群組，計算該節點加入該群組後的群組適當性的差異程度 值，接著將節點加入差異程度 值最大之節點所屬於的群組中。因此只需在第*i*個節點找出差異程度 值最大之第*j*個節點所屬於的群組加入節點彼此不能屬於相同群組的互斥判斷，使得產生的群組能符合使用者預先設定哪兩個朋友不能屬於相同群組的需求。

此外 BGLL 分群演算法主要重複進行群組的合併過程，不再出現群組合併的情況，即找出了該層的群組。此時若加入互斥判斷，則重複進行群組的合併過程，直到當不符合使用者設定哪兩個朋友不能屬於相同群組的需求時，即找出適當的朋友互斥的群組，因此將上述做法改良 BGLL 分群演算法，總共七個步驟：

1. 使用者設定哪兩個朋友不能屬於相同的群組。
2. 將所有節點設定為各自獨立的群組編號。
3. 每個節點再與相鄰節點所屬於的群組，計算該節點加入該群組後的群組適當性的差異程度 值。
4. 判斷與相鄰節點所屬於的群組之差異程度 值是否最大，若大於之前的差異程度 值，**且同時符合使用者設定哪兩個朋友不能屬於相同的群組的互斥條件 (改良部份一)**，就設定差異程度 值最大之群組為與相鄰節點所屬於的群組。
5. 接著將節點加入差異程度 值最大之群組中。
6. 重複執行步驟三至步驟五，**直到不再出現群組合併的情況或者當不符合使用者設定哪兩個朋友不能屬於相同群組的條件設定時 (改良部份二)**，即找出第一階段符合朋友互斥條件的群組。
7. 建立一個全新的圖，該圖的節點為第一階段所找出的各個群組，該節點之間的連邊權重是兩個群組之間所有連結邊的權重和。接著再重複執行步驟三至步驟六，針對全新的圖找出群組，重複直到不再找出更高一層的社群結構為止，即找出符合朋友互斥條件的群組。

假設使用者設定朋友 *i* 和朋友 *j* 不能屬於相同的群組時，此時*Ci* 代表朋友*i* 所屬於的群組，*Cj* 代表朋友 *j* 所屬於的群組，且朋友*i*與相鄰朋友*k*所屬於的群組*Ck* 之差異程度 值為最大時，則改良部份一的條件判斷式為 *Cj* != *Ck* ，當考慮多組朋友互斥因子條件設定主要是宣告標記 (flag)，初始值為 FALSE，透過迴圈的方式判斷每組朋友互斥因子條件設定，若符合則將標記設為 TRUE。此外以上條件只有當朋友為多組朋友互斥因子條件設定其中之一，才會進行朋友互斥因子條件設定的判斷，因此會再宣告標記，初始值為 TRUE，透過迴圈的方式判斷朋友為多組朋友互斥因子條件設定其中之一的朋友，若符合則將標記設為 FALSE ，若標記其中之一為 TRUE ，就設定差異程度 值最大之群組為與朋友*k*所屬於的群組*Ck* 。最後在針對每個朋友調整之後需要判斷是否符合所有互斥因子條件，此時會宣告標記，初始值為 TRUE，透過迴圈的方式，若有其中一項不符合就將標記設為 FALSE，結束第一階段。

第一階段

Q 值是否沒改變?

第二階段

否

是

初始化

透過所有節點逐步移至相鄰節點的群組中計算個別的 ，最後**選擇符合互斥因子條件**下，同時個別的 >0，將該節點加入 最大值的群組。

**符合所有互斥因子條件**

是

是

第一階段

基於第一階段偵測出  
的群組建立新的網路

Q 值是否沒改變?

否

否

圖3-1、考量互斥因子之自動分群機制 – 流程圖

(資料來源：本研究整理)

接著根據 BGLL 分群演算法第一階段的流程，會重複執行無法再進行群組合併為止，此時評估群組適當性的 Q 值已達最大值，可是因為使用者根據需求設定朋友互斥的條件，會使得 Q 值不一定到達最大值，則改良部份二的條件判斷式則為 ( *Ci =Cj* ) ，若有多組設定主要是主要是宣告標記，初始值為 TRUE，透過迴圈的方式進行判斷，若符合則將標記設為 FALSE，若標記為 FALSE，代表找出適當的朋友互斥群組，考量互斥因子之自動分群機制相關流程圖，如圖3-1所示。

|  |  |
| --- | --- |
| **演算法2、考量互斥因子之自動分群機制** | |
| **輸入:**  **輸出:** | g 為個人網路、f為互斥因子  *c* 為分群群組 |
| ***flag0* ← *true***  while *g can be contracted* do  // 第一階段  forall the *vertices* vi do  c[vi] ← i // 初始化  end  while totalIncrease > 0 **and *flag0* = *true*** do  forall the *vertices v* do  best ← v  maxDeltaQ ← 0  totalIncrease ← 0  forall the *neighbors w of v* do  deltaQ ← deltaQ from c[v] to c[w]  if deltaQ > maxDeltaQ then  ***flag1* ← *false***  ***flag2* ← *true***  **for *x* = 0 to f*.*length**  **if f[x][0] -1 and f[x][1] -1 then**  **if ( v = f[x][0] and c[ f[x][1] ] c[w] ) or**  **( v = f[x][1] and c[ f[x][0] ] c[w] ) then**  ***flag1* ← true**  **end**  **end**  **if w =f[x][0] or w= f[x][1]then *flag2* ← *false***  **end**  **if *flag1* = *true* or *flag2 = true* then**  maxDeltaQ ← deltaQ  best ← w  **end**  end  end  c[v] ← c[best]  totalIncrease ← totalIncrease + maxDeltaQ  end  **for *x* = 0 to f*.*length**  **if c[f[x][0]] = c[f[x][1]] then *flag0* ← *false***  **end**  end  // 第二階段  g ← contract(g,c[])  end  return c | |

(資料來源：本研究整理)

## 範例驗證

本研究主要是以 1977 年 Zachary 學者花費二年時間觀察空手道俱樂部的成員關係整理成空手道俱樂部的成員關係網路為範例，Zachary 空手道俱樂部的成員關係網路是個複雜的網路、是在社會網路分析等領域中最常用的一個小規模檢測網路之一 [26]，透過範例進行考量互斥因子之自動分群機制的驗證。

首先透過 R Studio工具，載入專門進行圖 (graph) 分析之 igraph 函式庫，透過R語言執行函式庫中的以不同分群演算法，將 Zachary 空手道俱樂部網路的成員分成不同的群組，確認BGLL分群演算法相較於其它分群演算法會產生最適當的群組，如圖 3-2所示。

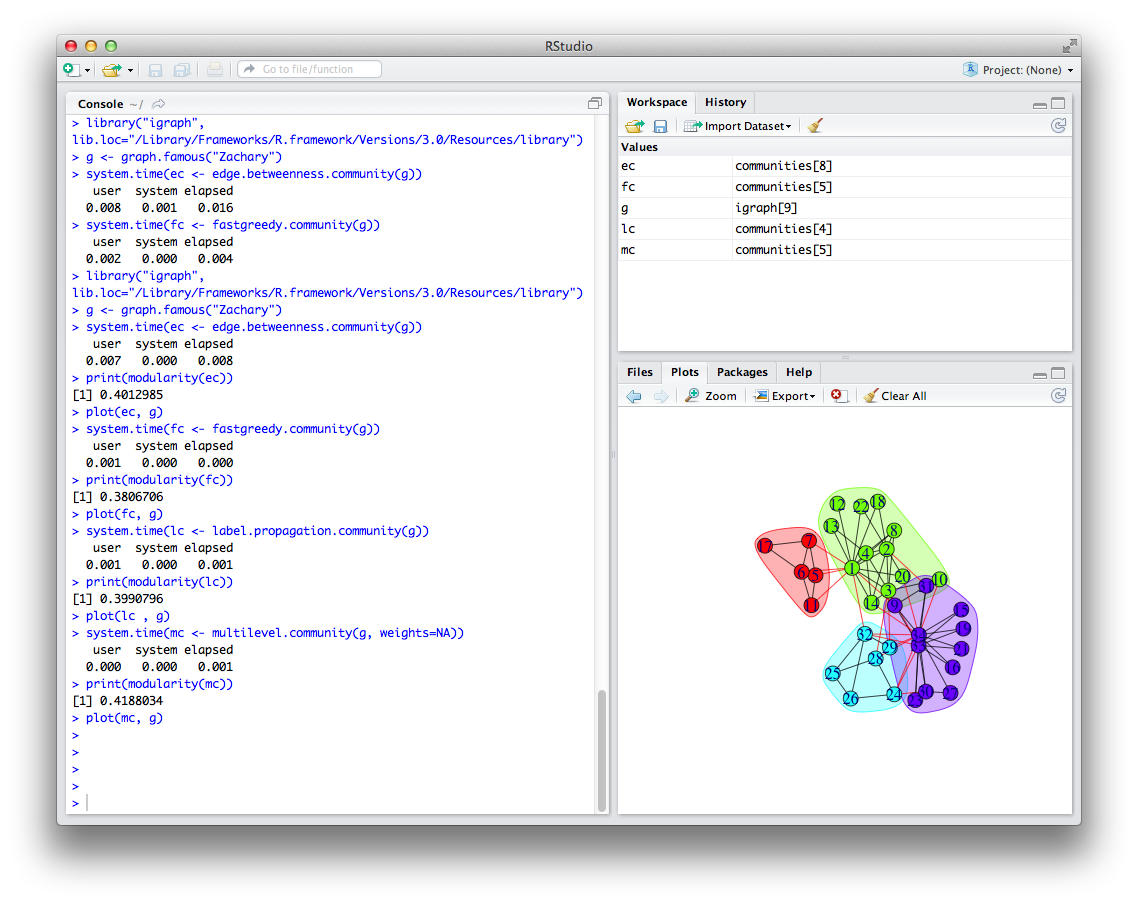


圖3-2、R Studio 實際操作畫面

(資料來源：本研究整理)

根據不同的分群演算法會有不同的分群結果，若以 Q 值衡量群組的適當性，則透過 BGLL 分群演算法就能產生最適當的分群結果，如圖 3-3 所示。

|  |  |
| --- | --- |
|  |  |
| GN 分群演算法  ( Q = 0.4012985 ) | CNM分群演算法 ( Q = 0.3806706) |
|  |  |
| LPA分群演算法 ( Q = 0.3744247) | BGLL分群演算法 ( Q = 0.4188034 ) |

圖3-3、空手道俱樂部的成員關係網路 – 不同分群演算法比較表

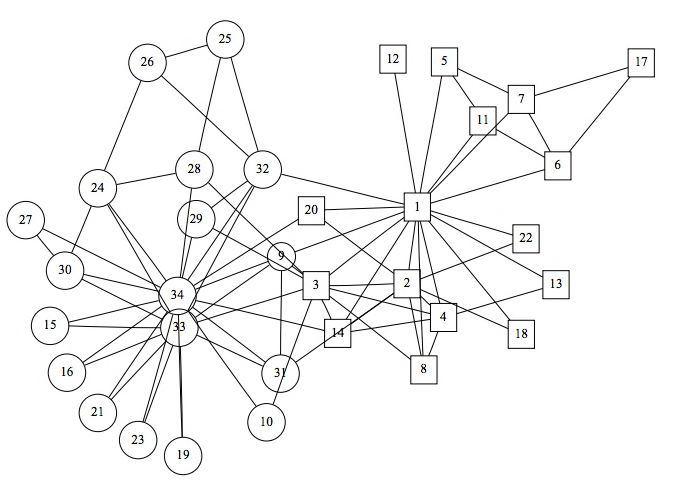
(資料來源：本研究整理)

雖然屬於相同的空手道俱樂部，可是在Zachary的調查過程中，該空手道俱樂部的校長Mr. Hi (節點1) 與主管 John (節點 34) 因為是否提高俱樂部收費的問題產生了一些糾紛，導致該俱樂部分裂成兩大群組，如圖3-4所示。此時針對空手道俱樂部，可以先設定校長 (節點1) 和主管 (節點34) 彼此不能屬於相同的群組的互斥條件，避免直接衝突的發生，如圖3-5所示。

|  |  |
| --- | --- |
|  |  |

圖3-4、空手道俱樂部的成員關係網路 – 真實情況示意圖  
(資料來源：本研究整理)

接著根據Zachary的調查結果得知成員 (節點10) 與當時主管John有直接關係，可是 GN 分群演算法卻將該成員不屬於任何群組之獨立的群組，CNM分群演法產生的結果中該成員不屬於屬於主管John 所在的群組，LPA分群演算法則是每次產生的群組皆會不一樣，BGLL分群演算法卻將該成員屬於校長 Mr. Hi 所在的群組，所以各種分群演算法皆無法有效將該成員屬於主管John 所在的群組，此時就有可能發生衝突，因此需要設定成員 (節點3) 和成員 (節點10) 彼此不能屬於相同的群組的互斥條件，避免間接衝突的發生，如圖3-5所示。



**互斥條件一**

校長 (節點1) 和主管 (節點34)

彼此不能屬於相同的群組  
**(現實生活已知發生對立衝突)**

**互斥條件二**

成員 (節點3) 和成員 (節點10)

彼此不能屬於相同的群組  
**(避免可能間接發生對立衝突)**

圖3-5、空手道俱樂成員之關係網路 – 相關互斥條件  
(資料來源：本研究整理)

該成員 (節點10) 只有與空手道俱樂部的另一位成員 (節點3) 和主管John (節點1) 兩者存在關係，此時為了要能讓該成員與主管 John 屬於相同的群組，因此需要設定該成員 (節點10) 和另一位成員 (節點3) 彼此不能屬於相同的群組，接著再進行朋友互斥的分群。將上述兩個條件進行設定之後，此時透過考量互斥因子之自動分群機制進行分群，接著根據衡量群組適當性的Q值進行評估，就能有效將空手道俱樂部分成最適當的群組，如圖3-6所示。

|  |  |
| --- | --- |
| BGLL 分群演算法  ( Q = 0.4188034 ) | 考量互斥因子之自動分群機制  ( Q = 0.4197896) |

圖3-6、空手道俱樂部的成員關係網路 – 考量互斥因子之自動分群機制比較

(資料來源：本研究整理)

　　主要透過 Louvain學者等人在網站中所提供的 BGLL 分群演算法之MATLAB程式碼，透過 MATLAB 工具方便進行逐步的探討分析，如圖3-7所示。接著改良BGLL分群演算法之MATLAB 程式碼，加入考量互斥因子之自動分群機制的判斷條件之後，進行 BGLL 分群演算法與考量互斥因子之自動分群機制逐步比較分析。首先針對空手道俱樂部已知的情況，會先設定朋友互斥的條件。

* 1. 校長 (節點1) 和主管 (節點34) 彼此不能屬於相同的群組。
  2. 成員 (節點3) 和成員 (節點10) 彼此不能屬於相同的群組。

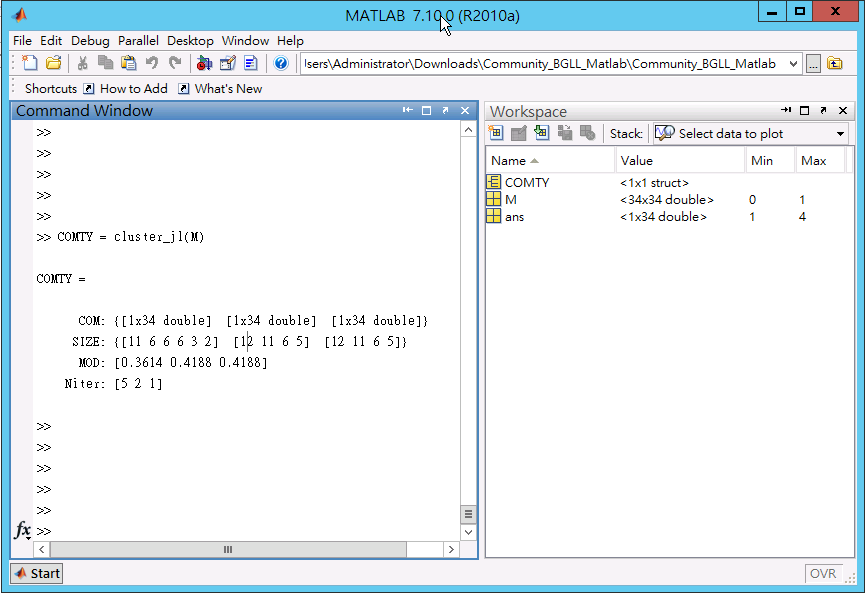


圖3-7、MATLAB實際操作畫面

(資料來源：本研究整理)

其步驟主要是先將所有節點設定為各自獨立的群組編號，如圖3-8所示。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **初始化 - 所有節點設定為各自獨立的群組編號**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  0 | 2  ↓  1 | 3  ↓  2 | 4  ↓  3 | 5  ↓  4 | 6  ↓  5 | 7  ↓  6 | 8  ↓  7 | 9  ↓  8 | 10  ↓  9 | 11  ↓  10 | 12  ↓  11 | 13  ↓  12 | 14  ↓  13 | 15  ↓  14 | 16  ↓  15 | 17  ↓  16 | | 18  ↓  17 | 19  ↓  18 | 20  ↓  19 | 21  ↓  20 | 22  ↓  21 | 23  ↓  22 | 24  ↓  23 | 25  ↓  24 | 26  ↓  25 | 27  ↓  26 | 28  ↓  27 | 29  ↓  28 | 30  ↓  29 | 31  ↓  30 | 32  ↓  31 | 33  ↓  32 | 34  ↓  33 | |

圖3-8、逐步分解之初始化設定

(資料來源：本研究整理)

當進行考量互斥因子之自動分群機制時，則每個節點再與相鄰節點所屬於的群組，計算該節點加入該群組後的群組適當性的差異程度 值，判斷與相鄰節點所屬於的群組之差異程度 值是否最大，若大於之前的差異程度 值，**且同時符合使用者設定哪兩個朋友不能屬於相同的群組的朋友互斥條件**，就設定差異程度 值最大之群組為與相鄰節點所屬於的群組。接著將節點加入差異程度 值最大之群組中。

重複執行**直到不再出現群組合併的情況或者當不符合使用者設定哪兩個朋友不能屬於相同群組的條件設定時**，即找出第一階段符合朋友互斥條件的群組，接著進行第二階段，建立一個全新的圖，該圖的節點為第一階段所找出各個群組，該節點之間的邊權重是兩個群組之間所有連結邊的權重和。最後再重複執行直到不再找出更高一層的社群結構為止，即找出符合朋友互斥條件最適當的群組，如圖3-9、圖3-10和圖3-11所示。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **考量互斥因子之自動分群機制之第一階段分群過程**  第一回合分群結果 (Q=0.205539)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  11 | 2  ↓  17 | 3  ↓  28 | 4  ↓  12 | 5  ↓  10 | 6  ↓  16 | 7  ↓  16 | 8  ↓  12 | 9  ↓  30 | 10  ↓  33 | 11  ↓  10 | 12  ↓  11 | 13  ↓  12 | 14  ↓  17 | 15  ↓  32 | 16  ↓  32 | 17  ↓  16 | | 18↓  17 | 19↓  32 | 20↓  17 | 21↓  33 | 22↓  11 | 23↓  33 | 24↓  25 | 25↓  27 | 26↓  25 | 27↓  29 | 28↓  27 | 29↓  31 | 30↓  29 | 31↓  30 | 32↓  31 | 33↓  32 | 34↓  32 |   第二回合分群結果 (Q=0.287558)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  17 | 2  ↓  17 | 3  ↓  12 | 4  ↓  12 | 5  ↓  10 | 6  ↓  16 | 7  ↓  16 | 8  ↓  12 | 9  ↓  30 | 10↓  32 | 11↓  10 | 12↓  17 | 13↓  12 | 14↓  12 | 15↓  32 | 16↓  32 | 17↓  16 | | 18↓  17 | 19↓  32 | 20↓  17 | 21↓  32 | 22↓  17 | 23↓  32 | 24↓  25 | 25↓  27 | 26↓  25 | 27↓  29 | 28↓  27 | 29↓  31 | 30↓  29 | 31↓  32 | 32↓  31 | 33↓  32 | 34↓  32 |   第三、四回合分群結果 (Q=0.307528)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  17 | 2  ↓  17 | 3  ↓  12 | 4  ↓  12 | 5  ↓  10 | 6  ↓  16 | 7  ↓  16 | 8  ↓  12 | 9  ↓  32 | 10↓  32 | 11↓  10 | 12↓  17 | 13↓  12 | 14↓  12 | 15↓  32 | 16↓  32 | 17↓  16 | | 18↓  17 | 19↓  32 | 20↓  17 | 21↓  32 | 22↓  17 | 23↓  32 | 24↓  25 | 25↓  27 | 26↓  25 | 27↓  29 | 28↓  27 | 29↓  31 | 30↓  29 | 31↓  32 | 32↓  31 | 33↓  32 | 34↓  32 | |

圖3-9、考量互斥因子之自動分群機制之逐步分解 – 第一階段

(資料來源：本研究整理)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **考量互斥因子之自動分群機制之第二階段分群過程**  第一回合分群結果 (Q=0.307528)   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 32  ↓  1 ↓ 0 | 17  ↓ 2 ↓ 1 | 12 ↓ 3 ↓ 2 | 16 ↓ 4 ↓ 3 | 10 ↓ 5 ↓ 4 | 25 ↓ 6 ↓ 5 | 27 ↓ 7 ↓ 6 | 29 ↓ 8 ↓ 7 | 31 ↓ 9 ↓ 8 |   第二、三回合分群結果 (Q=0. 419790)   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 32  ↓  1 ↓ 7 | 17  ↓ 2 ↓ 2 | 12 ↓ 3 ↓ 2 | 16 ↓ 4 ↓ 4 | 10 ↓ 5 ↓ 4 | 25 ↓ 6 ↓ 6 | 27 ↓ 7 ↓ 6 | 29 ↓ 8 ↓ 7 | 31 ↓ 9 ↓ 6 | |

圖3-10、考量互斥因子之自動分群機制之逐步分解 – 第二階段

(資料來源：本研究整理)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **考量互斥因子之自動分群機制之產生分群結果**  將第二階段之分群結果對應至所有節點 (Q=0. 419790)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  2 | 2  ↓  2 | 3  ↓  2 | 4  ↓  2 | 5  ↓  4 | 6  ↓  4 | 7  ↓  4 | 8  ↓  2 | 9  ↓  7 | 10↓  7 | 11↓  4 | 12↓  2 | 13↓  2 | 14↓  2 | 15↓  7 | 16↓  7 | 17↓  4 | | 18↓  2 | 19↓  7 | 20↓  2 | 21↓  7 | 22↓  2 | 23↓  7 | 24↓  6 | 25↓  6 | 26  ↓  6 | 27  ↓  7 | 28  ↓  6 | 29↓  6 | 30↓  7 | 31↓  7 | 32↓  6 | 33↓  7 | 34↓  7 |   產生考量朋友互斥條件最適當的分群結果 (Q=0. 419790)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  2 | 2  ↓  2 | 3  ↓  2 | 4  ↓  2 | 5  ↓  4 | 6  ↓  4 | 7  ↓  4 | 8  ↓  2 | 9  ↓  1 | 10↓  1 | 11↓  4 | 12↓  2 | 13↓  2 | 14↓  2 | 15↓  1 | 16↓  1 | 17↓  4 | | 18↓  2 | 19↓  1 | 20↓  2 | 21↓  1 | 22↓  2 | 23↓  1 | 24↓  3 | 25↓  3 | 26↓  3 | 27↓  1 | 28↓  3 | 29↓  3 | 30↓  1 | 31↓  1 | 32↓  3 | 33↓  1 | 34↓  1 | |

圖3-11、考量互斥因子之自動分群機制之逐步分解 – 分群結果

(資料來源：本研究整理)

　　當進行 BGLL 分群演算法時，則每個節點再與相鄰節點所屬於的群組，計算該節點加入該群組後的群組適當性的差異程度 值，判斷與相鄰節點所屬於的群組之差異程度 值是否最大，若大於之前的差異程度 值，就設定差異程度 值最大之群組為與相鄰節點所屬於的群組。接著將節點加入差異程度 值最大之群組中。重複執行直到不再出現群組合併的情況時，即找出第一階段符合朋友互斥條件的群組，如圖3-12、圖3-13和圖3-14所示。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BGLL分群演算法之第一階段分群過程**  第一回合分群結果 (Q=0.217045)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  11 | 2  ↓  17 | 3  ↓  9 | 4  ↓  12 | 5  ↓  10 | 6  ↓  16 | 7  ↓  16 | 8  ↓  12 | 9  ↓  30 | 10  ↓  9 | 11↓  10 | 12↓  11 | 13↓  12 | 14↓  17 | 15↓  32 | 16↓  32 | 17↓  16 | | 18↓  17 | 19↓  32 | 20↓  17 | 21↓  33 | 22↓  11 | 23↓  32 | 24↓  25 | 25↓  27 | 26↓  25 | 27↓  29 | 28↓  27 | 29↓  31 | 30↓  29 | 31↓  30 | 32↓  31 | 33↓  32 | 34↓  32 |   第二回合分群結果 (Q=0.289530)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  17 | 2  ↓  17 | 3  ↓  12 | 4  ↓  12 | 5  ↓  10 | 6  ↓  16 | 7  ↓  16 | 8  ↓  12 | 9  ↓  30 | 10↓  12 | 11↓  10 | 12↓  17 | 13↓  12 | 14↓  12 | 15↓  32 | 16↓  32 | 17↓  16 | | 18↓  17 | 19↓  32 | 20↓  17 | 21↓  33 | 22↓  17 | 23↓  32 | 24↓  25 | 25↓  27 | 26↓  25 | 27↓  29 | 28↓  27 | 29↓  31 | 30↓  32 | 31↓  32 | 32↓  31 | 33↓  32 | 34↓  32 |   第三回合分群結果 (Q=0.350920)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  17 | 2  ↓  17 | 3  ↓  12 | 4  ↓  12 | 5  ↓  10 | 6  ↓  16 | 7  ↓  16 | 8  ↓  12 | 9  ↓  32 | 10↓  12 | 11↓  10 | 12↓  17 | 13↓  12 | 14↓  12 | 15↓  32 | 16↓  32 | 17↓  16 | | 18↓  17 | 19↓  32 | 20↓  17 | 21↓  32 | 22↓  17 | 23↓  32 | 24↓  32 | 25↓  25 | 26↓  25 | 27↓  32 | 28↓  25 | 29↓  31 | 30↓  32 | 31↓  32 | 32↓  25 | 33↓  32 | 34↓  32 |   第四、五回合分群結果 (Q=0.361358)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  17 | 2  ↓  17 | 3  ↓  12 | 4  ↓  12 | 5  ↓  10 | 6  ↓  16 | 7  ↓  16 | 8  ↓  12 | 9  ↓  32 | 10↓  12 | 11↓  10 | 12↓  17 | 13↓  12 | 14↓  12 | 15↓  32 | 16↓  32 | 17↓  16 | | 18↓  17 | 19↓  32 | 20↓  17 | 21↓  32 | 22↓  17 | 23↓  32 | 24↓  25 | 25↓  25 | 26↓  25 | 27↓  32 | 28↓  25 | 29↓  25 | 30↓  32 | 31↓  32 | 32↓  25 | 33↓  32 | 34↓  32 | |

圖3-12、BGLL 分群演算法之逐步分解 – 第一階段

(資料來源：本研究整理)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BGLL分群演算法之第二階段分群過程**  第一回合分群結果 (Q=0.361358)   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 32  ↓  1 ↓ 0 | 17  ↓  2 ↓ 1 | 25  ↓  3 ↓ 2 | 12  ↓  4 ↓ 3 | 16  ↓  5 ↓ 4 | 10  ↓  6 ↓ 5 |   第二、三回合分群結果 (Q=0.418803)   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 32  ↓  1 ↓ 0 | 17  ↓  2 ↓ 2 | 12  ↓  3 ↓ 2 | 25  ↓  4 ↓ 3 | 16  ↓  5 ↓ 5 | 10  ↓  6 ↓ 5 | |

圖3-13、BGLL 分群演算法之逐步分解 – 第二階段

(資料來源：本研究整理)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BGLL分群演算法之產生分群結果**  將第二階段之分群結果對應至所有節點 (Q=0.418803)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  2 | 2  ↓  2 | 3  ↓  2 | 4  ↓  2 | 5  ↓  5 | 6  ↓  5 | 7  ↓  5 | 8  ↓  2 | 9  ↓  0 | 10↓  2 | 11↓  5 | 12↓  2 | 13↓  2 | 14↓  2 | 15↓  1 | 16↓  1 | 17↓  5 | | 18↓  2 | 19↓  0 | 20↓  2 | 21↓  0 | 22↓  2 | 23↓  0 | 24  ↓  3 | 25↓  3 | 26↓  3 | 27↓  0 | 28↓  3 | 29↓  3 | 30↓  0 | 31↓  0 | 32↓  3 | 33↓  1 | 34↓  1 |   產生考量朋友互斥條件最適當的分群結果 (Q=0.418803)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1  ↓  1 | 2  ↓  1 | 3  ↓  1 | 4  ↓  1 | 5  ↓  4 | 6  ↓  4 | 7  ↓  4 | 8  ↓  1 | 9  ↓  2 | 10↓  1 | 11↓  4 | 12↓  1 | 13↓  1 | 14↓  1 | 15↓  2 | 16↓  2 | 17↓  4 | | 18↓  1 | 19↓  2 | 20↓  1 | 21↓  2 | 22↓  1 | 23↓  2 | 24↓  3 | 25↓  2 | 26↓  3 | 27↓  3 | 28↓  2 | 29↓  3 | 30↓  3 | 31  ↓  2 | 32↓  3 | 33↓  2 | 34↓  2 | |

圖3-14、BGLL 分群演算法之逐步分解 – 分群結果

(資料來源：本研究整理)

因此透過空手道俱樂部的例子，可以得知透過考量互斥因子之自動分群機制相較於 BGLL 分群演算法，不論從Q值的群組適當性評估或是執行回合的效能評估皆更能有效的產生最適當的分群結果。

所以透過以上的實驗確認考量互斥因子之自動分群機制，在設定多組適當的互斥因子條件，就能有效解決章節 3.1的問題定義。可是最適當的分群結果並不一定符合真實的情況，因此若是針對 Facebook 中之朋友進行分群時，會有許多的相對較小群組產生，像是一面之交的朋友，此時為了讓分群結果更符真實的情況且降低群組的數量，所以需要根據真實情況，透過設定群組中最少成員數，接著將不滿足最少成員數之群組中的成員依照順序加入至滿足最少成員數之群組中且同時將該成員加入至該群組的 值為最大值， 值的範圍介於 -1 至 1 之間，同時考量多組朋友互斥的條件。此時針對空手道俱樂部的例子，透過考量互斥因子之自動分群機制產生最適當的群組，如圖3-14所示，四個群組分別為：

* 群組一：1、2、3、4、8、12、13、14、18、20。
* 群組二：5、6、7、11、17。
* 群組三：9、10、15、16、19、21、23、27、30、31、33、34。
* 群組四：24、25、26、28、29、32。

接著設定群組中最少成員數為6人，這四個群組中有二個群組為不滿足最少成員數之群組，分別為群組二和群組四，且另二個群組滿足最少成員之群組，分別為群組一和群組三。這時先針對群組二的成員5、成員6、成員7、成員11和成員17、分別計算加入至群組一與加入至群組三的 值，其中這5個成員與群組一的 值皆比與群組三的 值大，因此會將群組二的所有成員加入至的群組一。再來針對群組二的成員24、成員25、成員26、成員28、成員29和成員29、分別計算加入至群組一與加入至群組三的 值，其中這6個成員與群組三的 值皆比與群組一的 值大，因此會將群組四的所有成員加入至的群組三，因此最後的分群結果就會符合真實的情況，如圖 3-15 所示。

|  |  |
| --- | --- |
| 考量互斥因子之自動分群機制 | 考量互斥因子之自動分群機制 (後續調整) |

圖3-15、考量互斥因子之自動分群機制後續調整比較表

(資料來源：本研究整理)

總結針對空手道俱樂部的成員關係網路，若透過模組性 (modularity) 之評估指標 Q 值進行比較，**則考量互斥因子之自動分群機制，設定適當的互斥因子條件之後，所產生群組對於社群結構的適當性為 Q = 0.4197896 比 BGLL 分群演算法產生群組對於社群結構的適當性為 Q = 0.4188034 更加適當，並且符合所設定的多組互斥條件，分別為校長 (節點1) 和主管 (節點34) 彼此不能屬於相同的群組，以及成員 (節點3) 和成員 (節點10) 彼此不能屬於相同的群組，此外若與其它分群演算法進行衡量群組適當性之評估指標Q值的比較，如表3-1所示**。最後若再進行群組中最少朋友數的後續調整之後，所產生的群組將能完全符合空手道俱樂部的成員關係網路在於當時真實生活的情況。

表3-1、空手道俱樂部的成員關係網路 - 不同分群演算法Q值比較表

|  |  |  |
| --- | --- | --- |
| **分群演算法** | **Q 值** | **互斥條件** |
| Girvan and Newman (GN) [25] | 0.4012985 | 、 |
| Greedy optimization (CNM) [28] | 0.3806706 | 、 |
| Label propagation (LPA) [34] | 0.3744247 | 、 |
| Community folding (BGLL) [10] | 0.4188034 | 、 |
| 考量互斥因子之自動分群機制 | 0.4197896 | 、 |

(資料來源：本研究整理)

# **系統開發與實作**

## 基本介紹

雖然透過空手道俱樂部的成員關係網路的分析驗證考量互斥因子之自動分群機制能產生對於社群結構符合條件最適當的群組，且再經過後續調整之後能產生最符合真實情況的群組，可是若要將此機制應用至Facebook社群網站中進行個人網路的比較分析時，當要進行實驗設計時就會面臨三大問題，分別為：

1. 取得使用者的個人網路資訊和資料來源必需經過使用者同意。
2. 透過考量互斥因子之自動分群機制產生符合使用者回饋的群組。
3. 了解使用者對於考量互斥因子之自動分群機制產生群組的檢索程度。

為了解決所面臨的三大問題，所以本研究會需要進行朋友分群系統的開發與實作，如表4-1所示。

表4-1、面臨實驗設計之需求必需實作系統 - 系統功能應用表

| **面臨實驗設計之需求必需實作系統** | **系統功能應用畫面** |
| --- | --- |
| 一、取得使用者的個人網路資訊和資料來源必需經過使用者同意。  因為本研究資料來源是以取得使用者在 Facebook 社群網站中與朋友之間存在的共同朋友關係為主，接著建構出的個人網路 (ego-centric network)，同時資料來源必須由使用者本人同意，才能取得使用，所以本研究主要是採用 Facebook 社群網站中之OAuth 授權協定解決此問題，相關文獻探討請參考2.1章節。 | 螢幕快照 2013-06-13 下午11  螢幕快照 2013-06-13 下午11 |
| **面臨實驗設計之需求必需實作系統** | **系統功能應用畫面** |
| 二、透過考量互斥因子之自動分群機制產生符合使用者回饋的群組。  因為本研究所提出的考量互斥因子之自動分群機制，主要能針對使用者回饋的資訊進行分群，因此需要讓使用者能設定多組兩位朋友不能屬於相同群組的條件設定，接著再進行分群。並且將產生的分群結果呈現給使用者進行協助本研究實驗進行調整與直接將群組產生至 Facebook 社群網站中當成自訂的朋友名單使用，相關文獻探討請參考2.2至2.4章節和相關資訊請參考3.1至3.3章節。 |  |
| 三、了解使用者對於考量互斥因子之自動分群機制產生群組的檢索程度。  因為本研究需要了解每個群組的檢索程度是否符合使用者在Facebook 社群網站中之真實的情況，所以必須要能讓使用者針對所有的群組進行個別的調整。接著還必須確認使用者是否已經調整且確認所有產生的群組，此時才能將使用者回饋資訊透過資訊檢索領域中的查準率和查全率進行比較分析，相關文獻探討請參考2.5章節。 |  |

(資料來源：本研究整理)

本研究實作系統主要是以網站為主，主要可以分成前端和後端兩大部份，前端主要呈現網頁畫面讓使用者直接進行操作，後端主要是即時接收來使用者針對各個群生調整後的真實回饋資訊，如圖4-1所示。

網站

資料庫

HTML5  
網頁

前端

後端

Google  
表單

Google 雲端空間

使用者

操作

傳送  
資訊

同意

取得資訊

朋友資訊

Facebook   
Graph

圖4-1、朋友分群系統示意圖  
(資料來源：本研究整理)

前端網站主要是透過HTML5網頁彼此之間的動態搭配進行開發，HTML5 廣義是指的是包括HTML、CSS和JavaScript在內的技術組合，其中HTML (hypertext markup language) 為超文件標示語言，一開始被設計的目的，只是用來傳遞純文字，讓世界各地的物理學家能夠方便的進行合作研究。後來網路應用程式的概念出現，因為具有隨時隨地，不需額外安裝和更新，不佔儲存空間，資料同步等優點，所以 HTML 早就被拿來做超出本來設計要做的事情，且大量依賴 Javascript 的函式庫 (library) ，像是jQuery 和 Facebook javascript SDK，Javascript又稱ECMAScript，是一種廣泛用於客戶端網頁開發的腳本語言，最早是在HTML上使用的，用來給HTML網頁添加動態的互動功能，此時網站互動的功能會讓參與者的操作更佳便利。

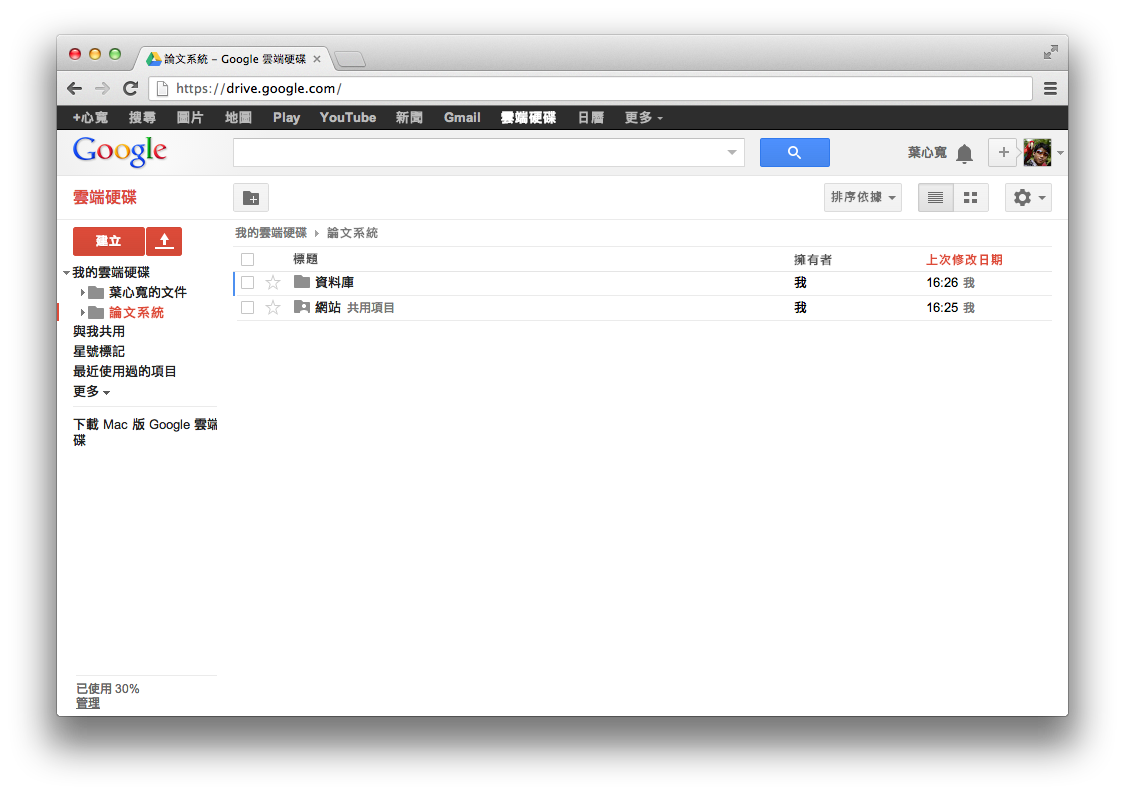


圖4-2、Google 雲端硬碟公開發佈網站系統  
(資料來源：本研究整理)

後端資料庫主要是以Google表單為主，方便前端網站只需透過HTTP POST就能傳送資訊至 Google 表單中，即時接收使用者回饋的資訊，同時能直接在 Google 試算表中進行比較分析。最後本研究所實作的系統，不論是前端網站或者後端資料庫皆存放至 Google 雲端空間之中，讓參與者能直接透過網際網路開啟朋友分群系統協助本研究實驗的進行，如圖4-2所示。

針對取得使用者的個人網路資訊和資料來源必需經過使用者同意的問題，首先取得使用者的個人網路資訊，主要是透過Graph API，Graph API 是取得在 Facebook 社群圖譜 (social graph) 中資訊的主要方法。這是基於 HTTP 的 API，使用者可透過 Graph API 查詢，建立資訊和刪除資訊，如圖4-3所示。

系統開發與實作的資料來源，主要是透過社群網路 Facebook 的Graph API ，登入 Facebook 使用者進行身份認證之後，再透過 OAuth 授權協定 [53]，請使用者同意系統取得資料存取的權限，待使用者同意之後，系統就會取得Access Token ，最後透過 Facebook 提供 Javascript 函數庫中的指令 FB.api()，搭配 Graph API 即可取得使用者已經被授予權限存取使用者相關內容資訊。

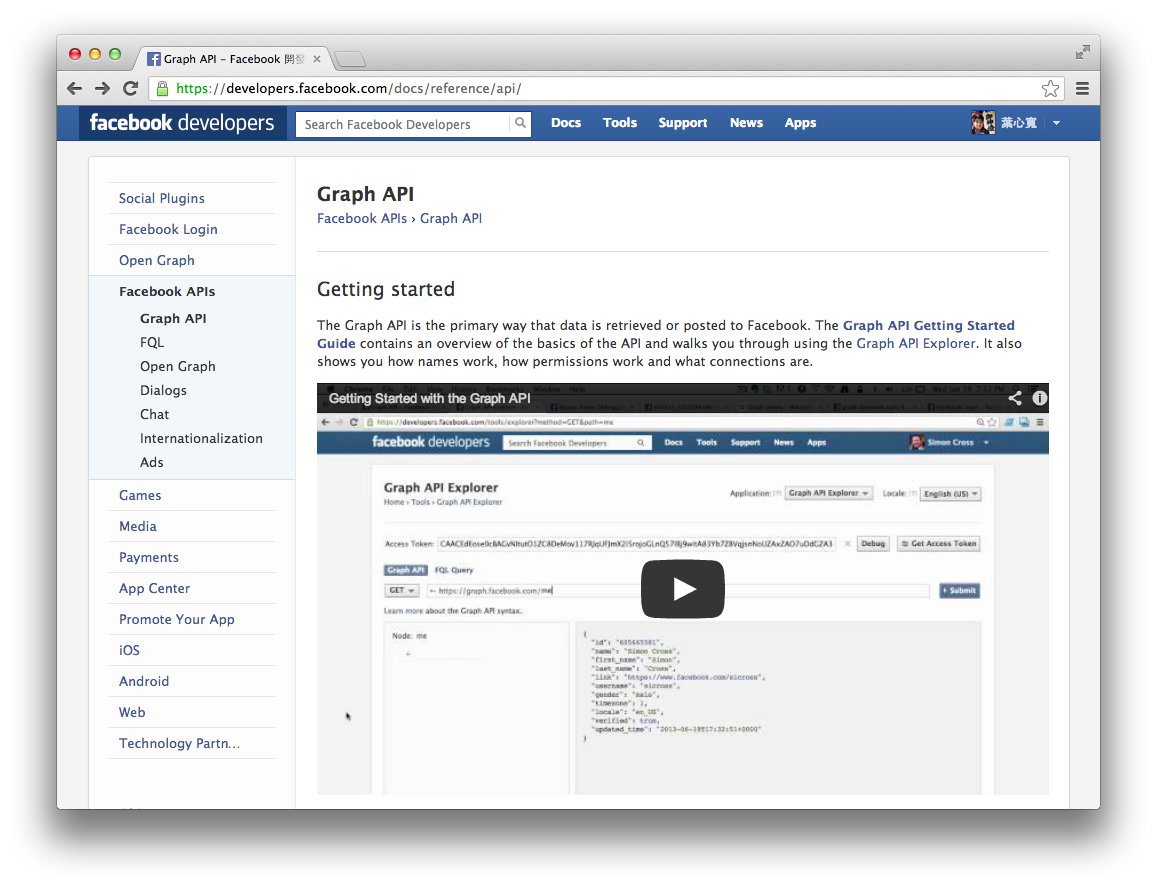


圖4-3、Facebook API 開發者教學網站

(資料來源：本研究整理)

OAuth 授權協定在認證及授權的過程中涉及的四種角色，一、資源擁有者 (resource owner) 授予存取受保護資源的能力。二、資源伺服器 (resource server) 託管受保護的資源的伺服器，能透過存取權杖 (access token) 回應受保護資源的請求。三、用戶端 (client) 代表受保護資源請求的應用程式資源授權的擁有者。四、授權伺服器 (authorization server) 發出存取權杖 (access token) 成功後，用戶端存取伺服器驗證資源擁有者獲得授權。基本處理的流程步驟，如圖4-4所示：

1. 用戶端會向資源擁有者請求授權。
2. 資源擁有者會授予用戶端權限。
3. 用戶端會將被授予的權限和用戶端憑證傳給授權伺服器。
4. 授權伺服器會將存取權杖傳給用戶端。
5. 用戶端會透過存取權杖請求資源伺服器存取受保護的資源。
6. 資源伺服器驗證存取權杖之後，會將受保護的資源給用戶端。

客戶端

資源

伺服器

授權

伺服器

資源

擁有者

授權請求

授權允許

授權允許 & 客戶端憑證

存取權杖 (Access Token)

存取權杖 (Access Token)

被保護的資源

圖4-4、OAuth 授權協定流程圖

(資料來源：Hammer et al. (2012) [53] )

本研究的系統會需要讀取使用者的朋友資訊，共同朋友資訊、朋友名單資訊以及管理朋友名單等權限，可是其中朋友資訊和共同朋友資訊預設皆屬於公開資訊，因此系統只需要使用者同意取得朋友名單和管理朋友名單的權限。系統會先取得朋友名單再以自動化的方擴展朋友名單，簡化操作讓使用者能根據需求更快速的管理朋友名單，最後再將朋友名單的結果儲存至 Facebook 社群網站中。

其中使用者資訊主要是透過 Graph API ，像是 http://graph.facebook.com/　USER\_ID?fields=id,name，其中USER\_ID為使用者的ID**，**主要在不需要特別的 Access Token授權的情況下以 HTTP GET 的方式進行請求，回傳資料為 JSON 格式，最後解析 JSON 格式，即可取得使用者 ID 和名稱的基本資訊。接著共同朋友資訊主要是透過Graph API，其中PROFILE\_ID為使用者A的ID，而FRIEND\_ID為使用者B的ID，主要在不需要特別的 Access Token授權的情況下時，以 HTTP GET 的方式進行請求，回傳資料為 JSON 格式，最後解析 JSON 格式，即可讀取使用者A和使用者B兩位使用者之間共同朋友的清單資訊，每一個共同朋友皆有 ID 和名稱的基本資訊，Facebook 社群網站 Graph API 之相關存取權限，如表4-2所示。

表4-2、Facebook 社群網站 Graph API 之相關存取權限

|  |  |  |  |
| --- | --- | --- | --- |
| **名稱** | **描述** | **權限** | **回傳值** |
| USER\_ID/  friends | 使用者的朋友  (參數:無)  (請求:HTTP GET) | - | 物件陣列包括 朋友 ID 和名稱欄位 |
| USER\_ID/  mutualfriends | 使用者與朋友 之間的共同朋友  (參數:朋友ID)  (請求:HTTP GET) | - | 物件陣列包括 朋友 ID 和名稱欄位 |
| USER\_ID/  friendlists | 管理使用者的 朋友名單  (參數:朋友名單名稱)  (請求:HTTP POST) | manage\_frinedlists | 新增朋友名單 |

(資料來源：本研究整理)

　　所以當使用者透過朋友分群系統登入 Facebook 社群網站的帳號之後，再透過OAuth 授權協定讓使用者同意朋友分群系統取得朋友相關資訊，接著再透過朋友與共同朋友的資訊產生個人網路 (ego-centric network) ，相關的文獻探討請參考2.1章節，再透過介面讓使用者設定至少一組最多三組之兩位朋友不能屬於相同群組的互斥因子之使用者回饋條件，再透過考量互斥因子之自動分群機制進行分群，相關的資訊請參考3.1章節至3.3章節，接著使用者可以針對各個群組進行調整和應用，相關的文獻探討請參考2.2章節至2.4章節，最後針對每個群組產生使用者回饋資訊，透過資訊檢索領域中的查準率和查全率進行比較分析，相關的文獻探討請參考2.5章節。



## 應用說明

本研究實作出的朋友分群系統有四個主要的應用：

1. 讓受測者透過朋友分群系統進行Facebook 帳號登入，並且透過OAuth授權協助同意取的朋友相關等資訊，接著讀取朋友和共同朋友的資訊以建構出個人網路 (ego-centric network) 進行比較分析。
2. 接著提供系統操作介面讓受測者設定哪兩位朋友不能屬於相的群組之多個條件，透過考量互斥因子之自動分群機制進行分群，並且將產生的群組資訊呈現給受測者。
3. 再請受測者針對每個群組進行調整，最後取得使用者回饋的資訊，進行查準率（precision）與查全率（recall）之比較分析。
4. 方便受測者在進行群組的調整之後，就能直接替群組命名，將適當的群組新增至 Facebook 社群網站中產生對應的朋友名單之實務應用。

接著受測者需要透過朋友分群系統完成問卷調查，當完成問卷調查之後資料會匯整至 Google 文件之試算表中，方便本研究針對受測者對於考量互斥因子之自動分群機制進行分群所產生的群組之查全率和查準率等使用者回饋資訊進行比較與分析。

1. 開啟朋友分群系統進行Facebook 帳號的登入，如圖4-5所示，接著同意朋友分群系統存取朋友名單資訊，如圖4-6所示，以及同意朋友分群系統管理自訂朋友名單，如圖4-7所示，此時朋友分群系統就會開始從 Facebook 社群網站中讀取相關的朋友資訊，如圖 4-8 所示。



圖4-5、Facebook 應用程式之 OAuth 授權應用 – 使用者登入畫面

(資料來源：本研究整理)



圖4-6、Facebook 應用程式之 OAuth 授權應用 – 同意存取朋友資訊畫面

(資料來源：本研究整理)



圖4-7、Facebook 應用程式之 OAuth 授權應用 – 同意管理朋友名單畫面

(資料來源：本研究整理)



圖4-8、朋友分群系統 – 讀取朋友相關資訊畫面

(資料來源：本研究整理)

1. 接著可以選擇設定哪兩位朋友不能屬於相同群組，最多可以設定三組，且透過下拉選單和搜尋方式選擇朋友，如圖4-9所示。

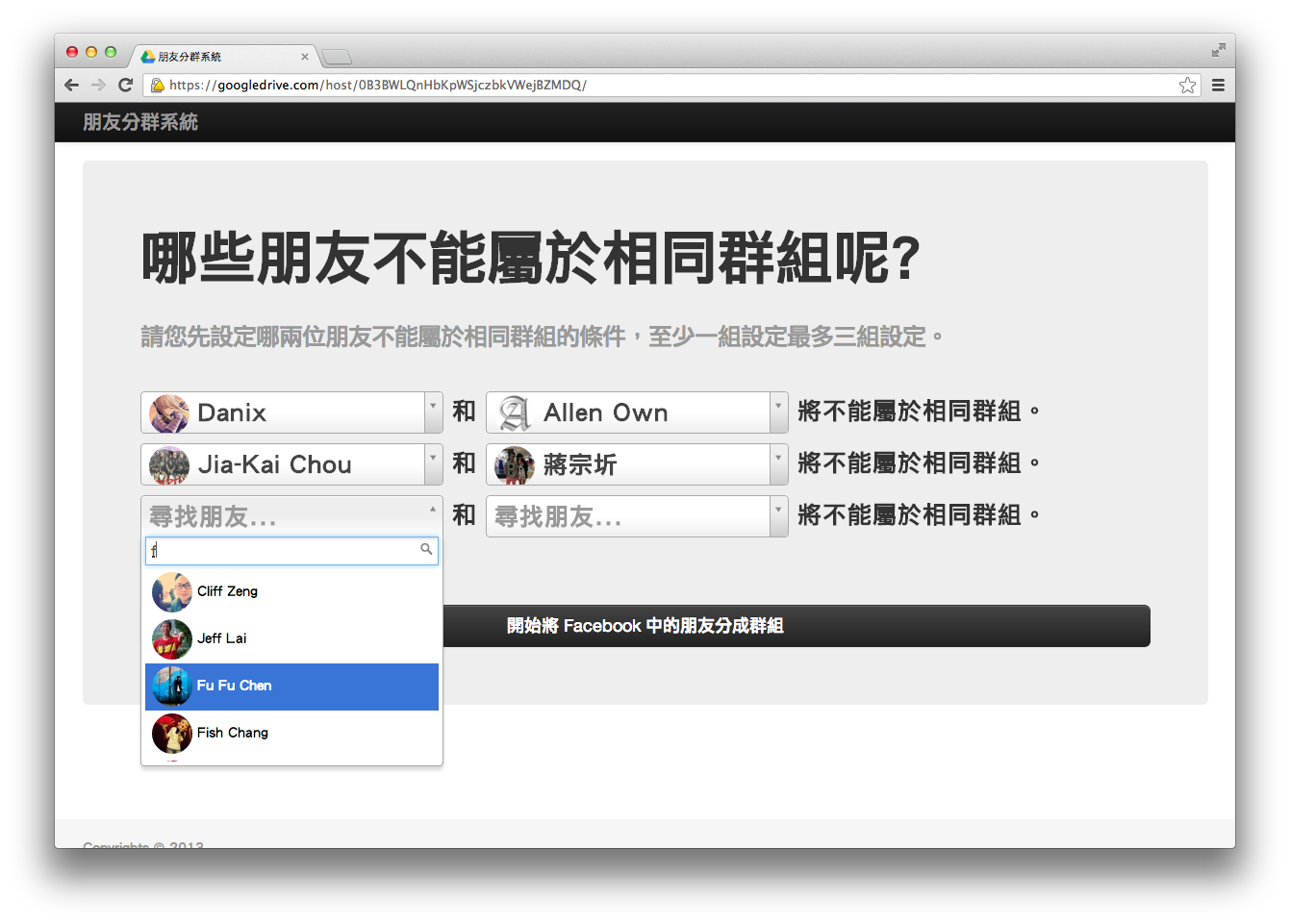


圖4-9、朋友分群系統 – 設定多組朋友互斥條件畫面

(資料來源：本研究整理)

1. 當朋友分群系統將 Facebook 社群網站中之朋友分成群組之後，請受測者針對所有產生的群組，進行群組的調整，如圖 4-10 和圖 4-11 所示。

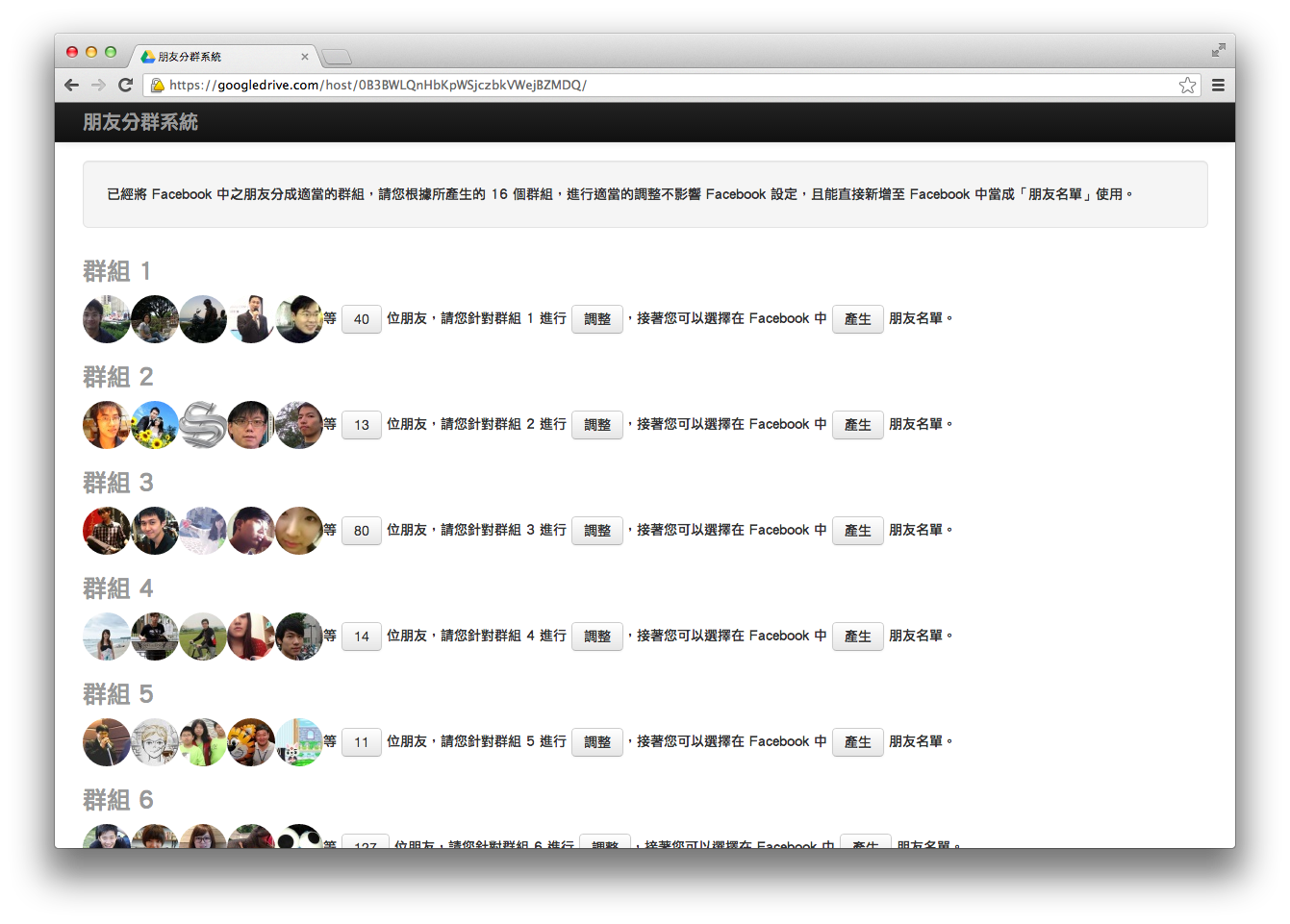


圖4-10、朋友分群系統 – 針對所有分群結果進行調整畫面

(資料來源：本研究整理)

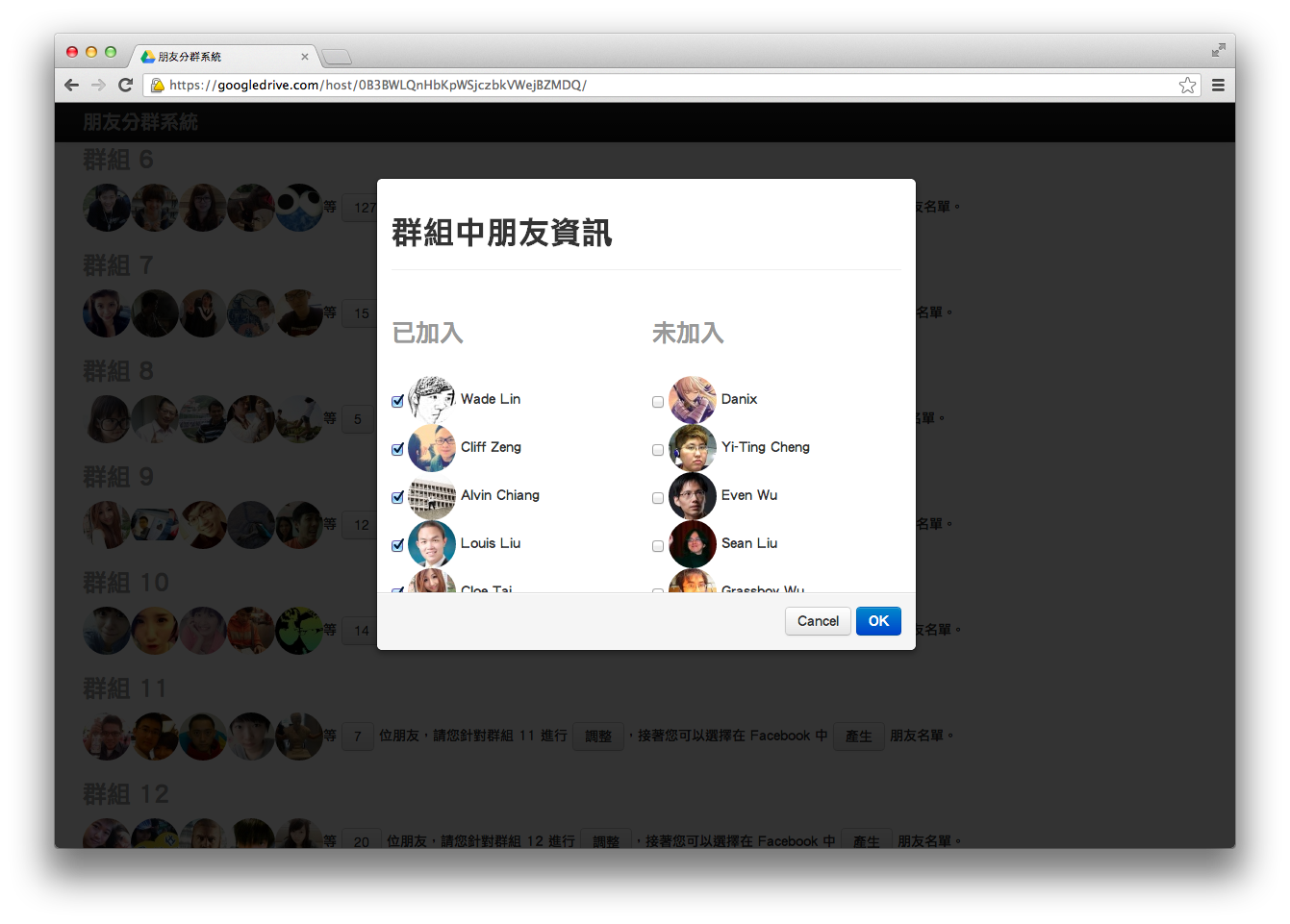


圖4-11、朋友分群系統 – 針對個別分群結果進行調整畫面

(資料來源：本研究整理)

1. 每當群組確認調整之後，會詢問受測者是否需要新增至 Facebook 當朋友名單，亦可直接產生朋友名單，如圖4-12和圖4-13所示。

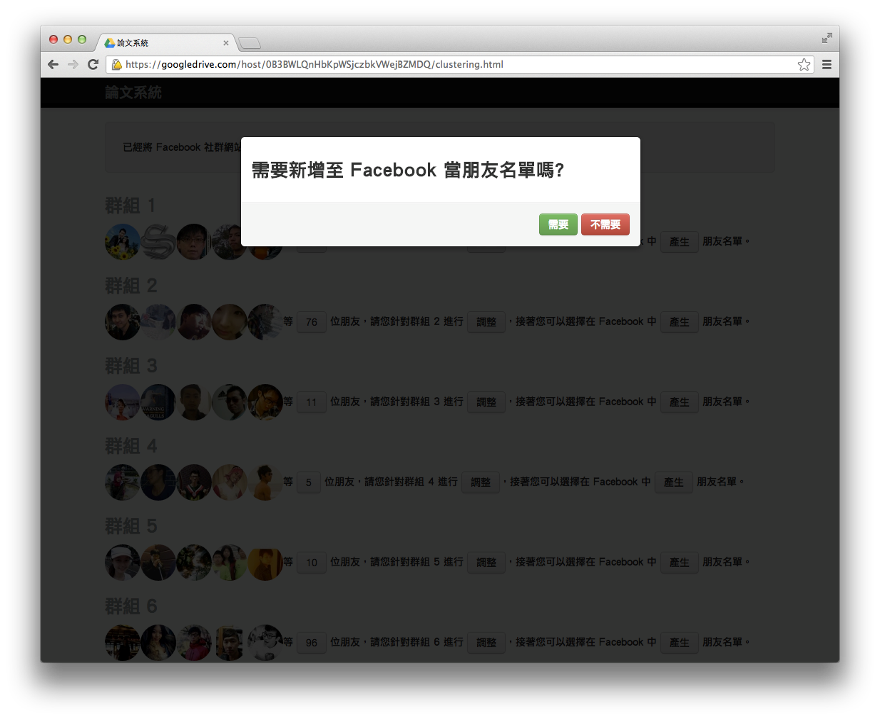


圖4-12、朋友分群系統 – 詢問受測者是否要將調整結果建立朋友名單畫面

(資料來源：本研究整理)



圖4-13、朋友分群系統 – 輸入建立 Facebook 朋友名單之名稱畫面

(資料來源：本研究整理)

1. 當所有群組確認調整之後，按下最下方的「我已經確認且完成了所有群組的調整」之確認鈕，就會產生各個群組經使用者調整後之查準率和查全率的圖表資訊，如圖4-14所示。

此外為了確認使用者皆有調整過每個群組，因此會有第一項條件判斷的相關提醒訊息，如圖4-15所示，且為了確保每個朋友皆必須設定至適當的群組中，因此會有第二項條件判斷的相關提醒訊息，如圖4-16 所示。

透過上述二項條件判斷，雖然能提醒受測者針對所有群組調整是否夠完整，可是卻也可能會造成受測者操作時的困擾，但為了確保使用者回饋資訊的可信度，因此等待受測者針對所有群組調整之後，必須審查是否符合上述二項條件，才會產生各個群組經使用者調整後之查準率和查全率的圖表資訊。

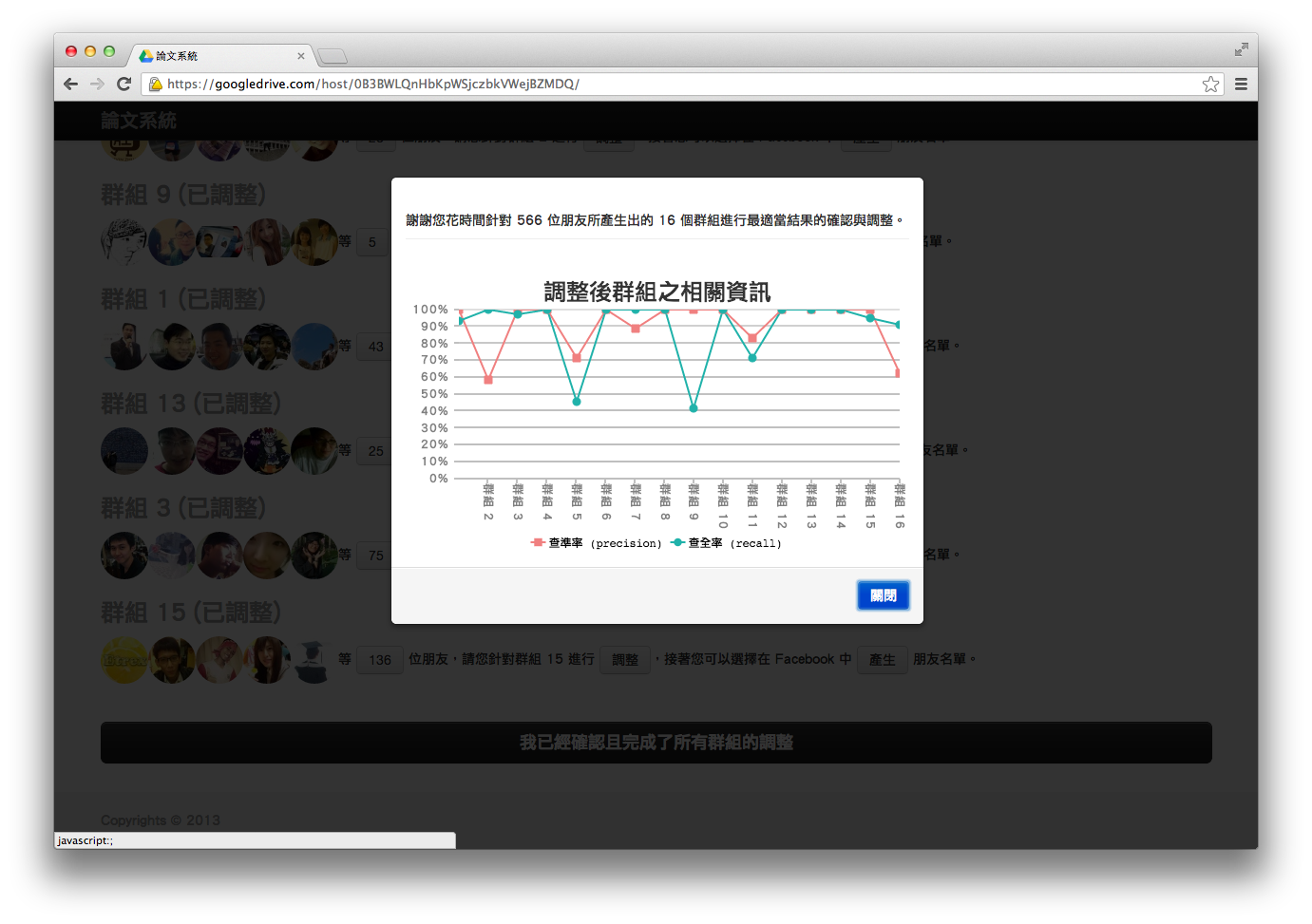


圖4-14、朋友分群系統 – 完成社群偵測結果調整與確認之圖表資訊畫面

(資料來源：本研究整理)

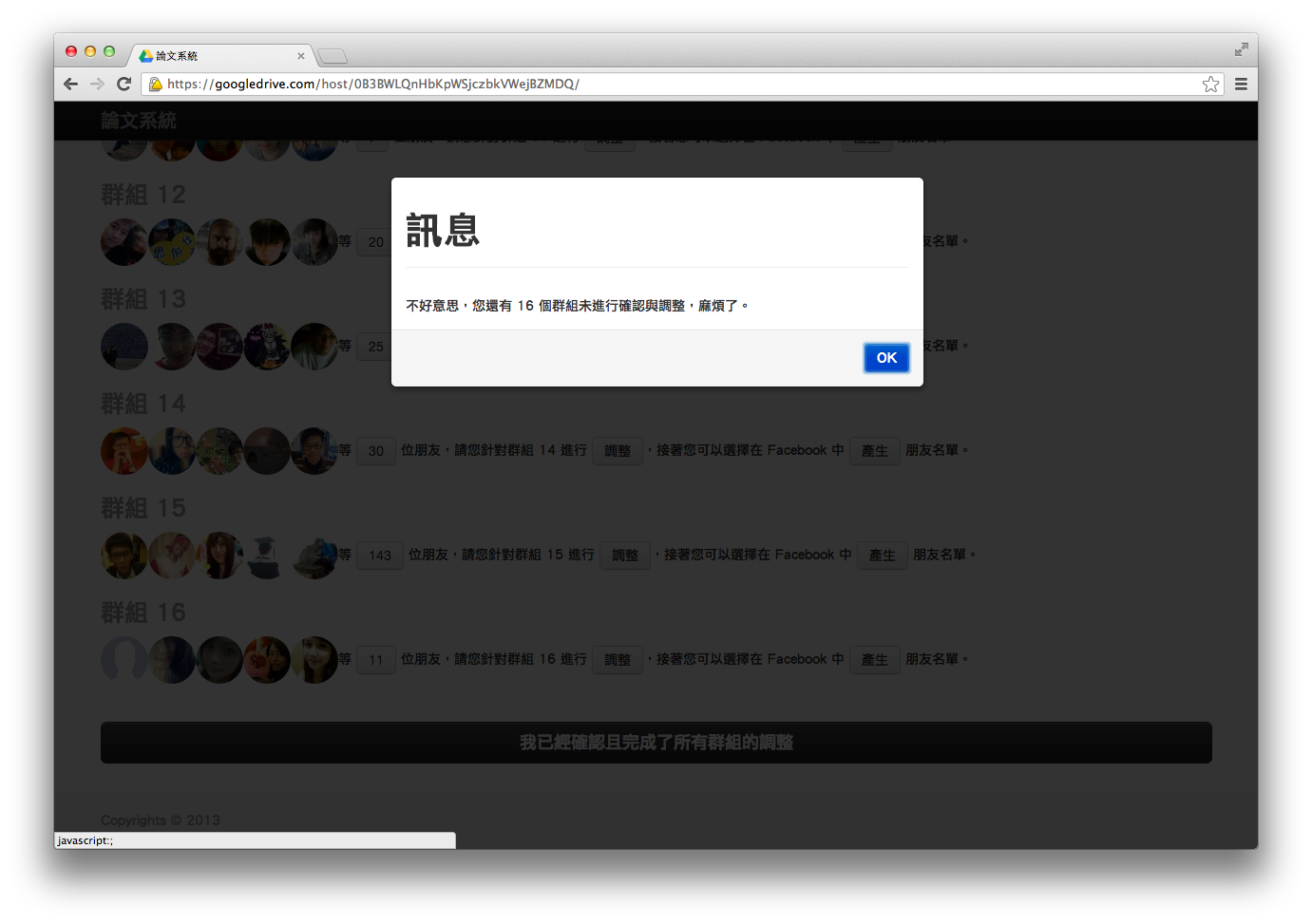


圖4-15、朋友分群系統 - 確認使用者皆有調整過每個群組之訊息畫面

(資料來源：本研究整理)

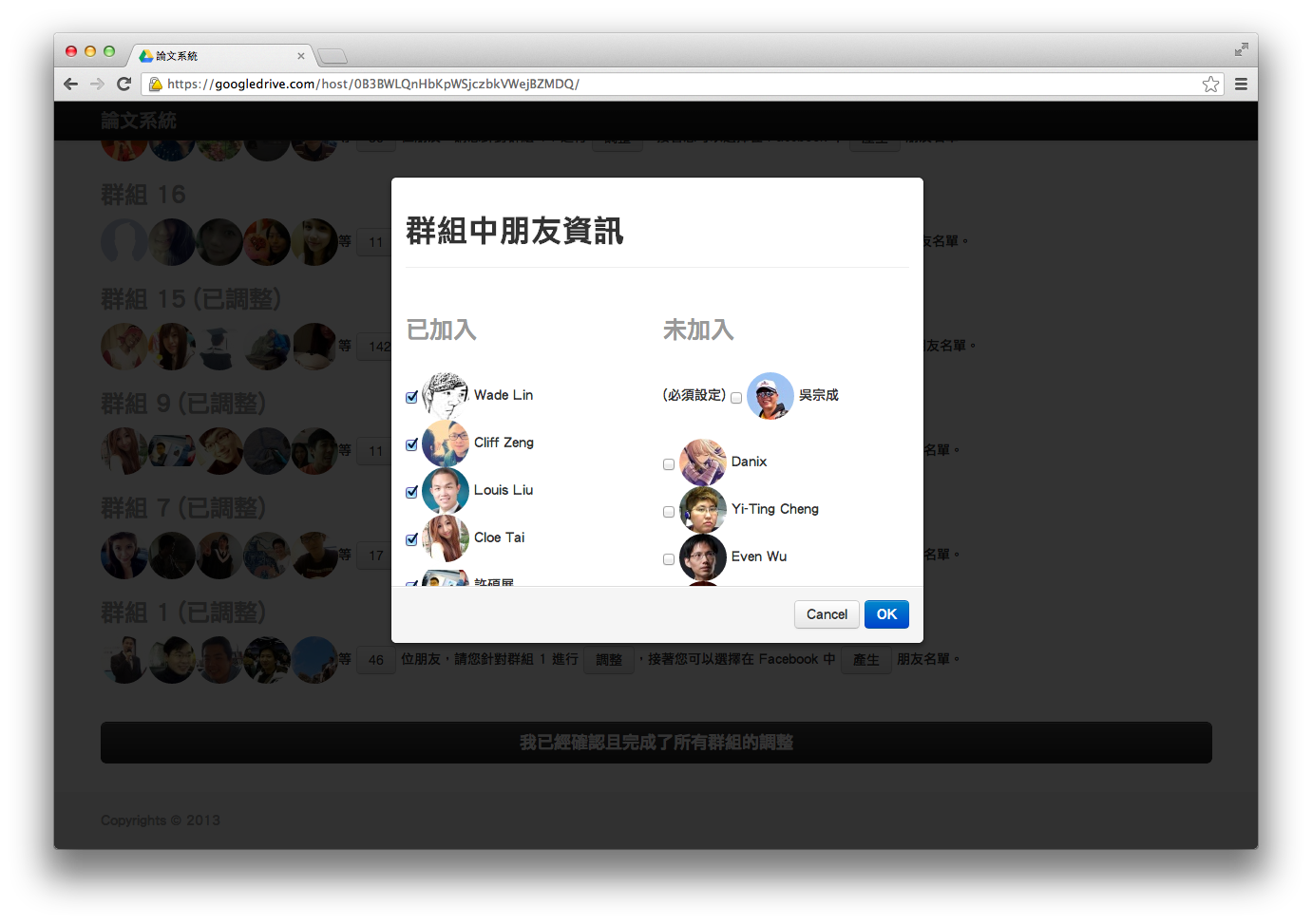


圖4-16、朋友分群系統 - 確保每個朋友皆必須設定至群組中之操作畫面

(資料來源：本研究整理)

1. 最後系統在產生圖表時，會同時將受測者針對群組相關的回饋資訊會傳送至 Google 文件中，如圖4-17所示。

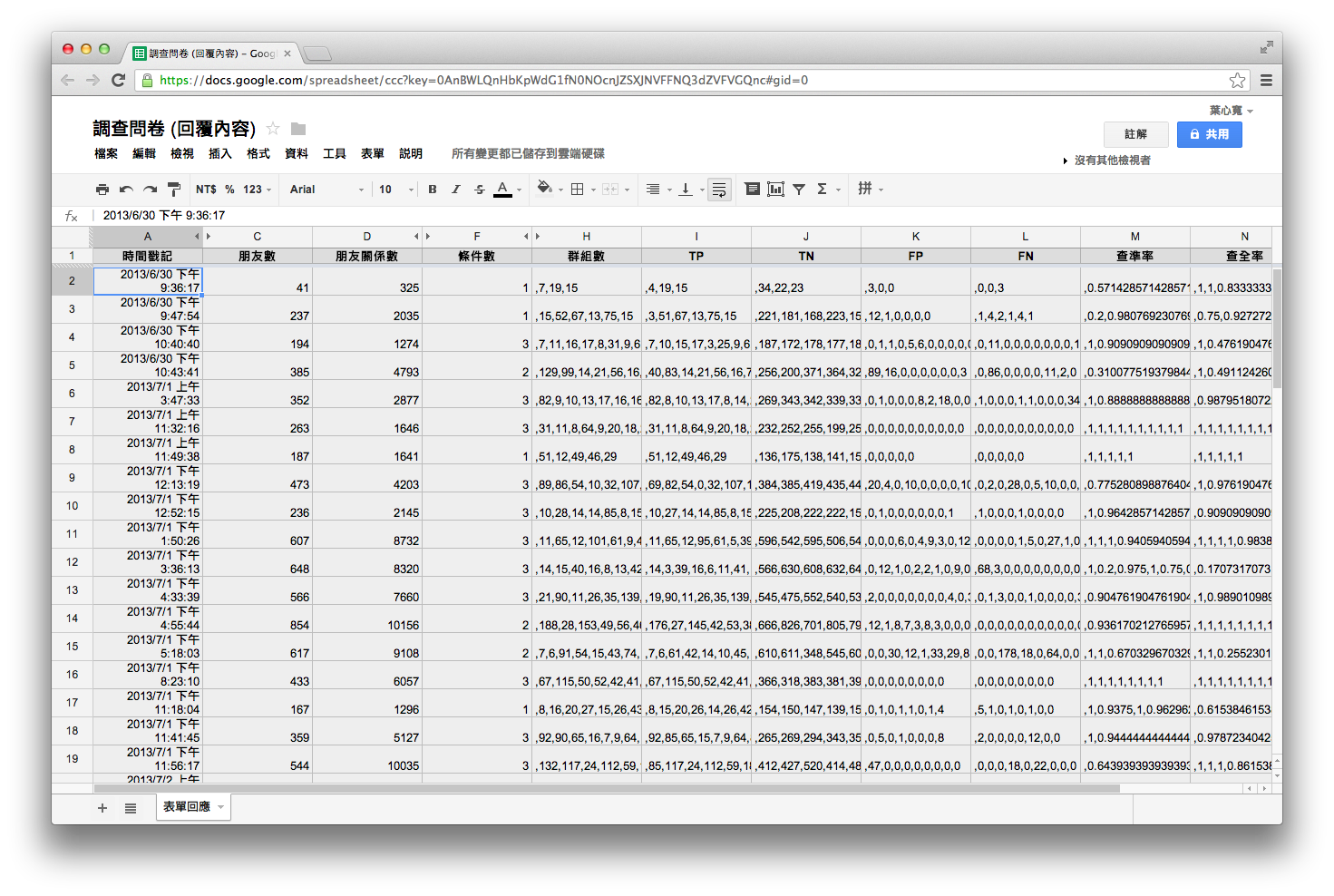


圖4-17、朋友分群系統 – 將使用者回饋資訊回傳至Google文件之畫面

(資料來源：本研究整理)

# **實驗與評估分析**

* 1. 實驗設計

在 2012 年Liu學者等人提出網站系統的工具主要目的在於能有效簡化朋友名單的管理，且主要應用BGLL分群演算法進行分群，該實驗設計主要是以該篇論文的 5 位作者 (author) 進行朋友名單管理的實驗，實驗數據主要是以朋友名單偵測總數、使用者建立總數和總共花費時間 [8]。上述的實驗設計以使用者建立總數當成使用者回饋資訊進行分析，平均有69% 的朋友名單會被建立，可是卻沒有針對偵測朋友名單以資訊檢索中的查準率（precision）與查全率（recall）進行使用者回饋資訊的比較分析。

首先本研究的樣本取得主要是透過 Give Me My Data 的應用程式，取得我本人目前在 Facebook 社群網站中之朋友資訊，總共 563 筆，如圖5-1所示。

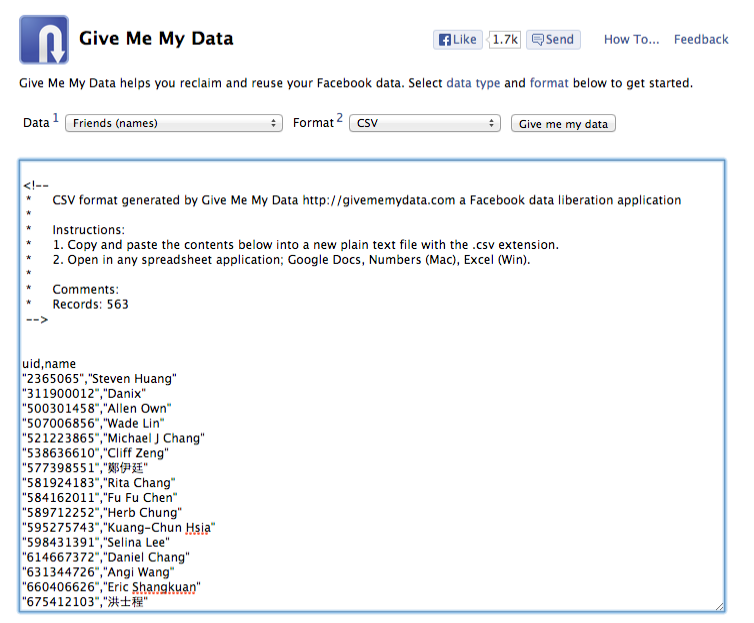
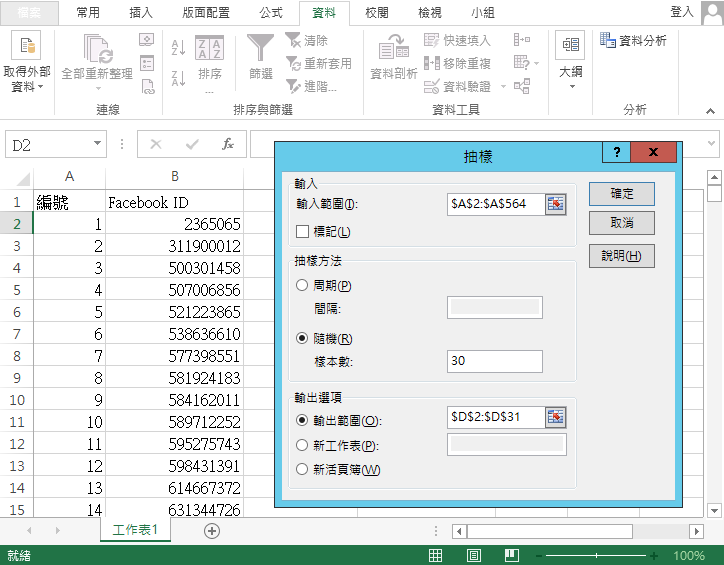


圖5-1、Give Me My Data 應用程式畫面

(資料來源：本研究整理)

接著將563位朋友的Facebook ID匯入至 Excel 試算表中，此時總體是563位朋友的Facebook ID，單位為朋友個數，再以總體的所有單位進行編號，接著透過增益集中的「資料分析」功能，選擇「抽樣」分析工具，選取輸入範圍，抽樣方法選擇「隨機」，輸入樣本數為30 人 [8][12][52][54][55]。根據中央極限定理 (central limit theorem) 當樣本大小大於等於30人時，樣本平均數分佈會呈現常態分配 (normal distribution) [54]，此外本研究屬於個人問答的方式，所以根據1994年 Nielsen學者建議透過此方式的實驗設計中的抽樣樣本至少 30 人以上 [55]，最後選取輸出範圍，透過簡單隨機抽樣法 (simple random sampling) 產生隨機的抽樣樣本，如圖5-2所示。



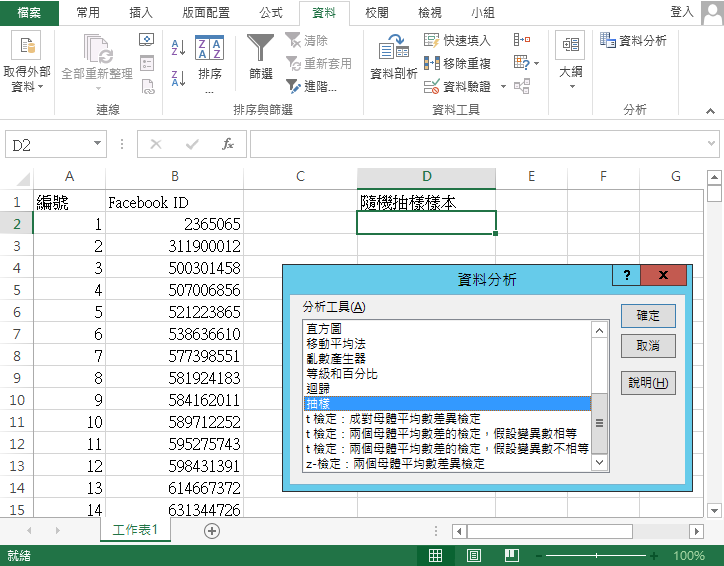


圖5-2、Excel 試算表之資料分析功能畫面

(資料來源：本研究整理)

本研究的實驗設計總共有五個主要的步驟，分別為：

1. 本研究會根據抽樣的 30 位受測者，透過 Facebook 社群網站中的訊息功能，請 30 位受測者透過網站操作朋友分群系統。
2. 受測者先會詢問相關的情況，在溝通之後同意協助本研究的實驗。
3. 受測者會透過 OAuth 授權協定登入至Facebook 社群網站，並且同意朋友分群系統取得朋友相關資訊。
4. 受測者設定至少一組最多三組之兩位朋友不能屬於相同群組的條件設定，接著進行考量互斥因子之自動分群機制產生至少5位朋友數的群組，再請受測者針對每個群組進行調整。
5. 受測者已調整和確認全部群組之後，按下確認鈕就會即時計算每個群組以資訊檢索中的查準率和查全率產生圖表讓受測者得知操作已經完成，同時傳送相關資訊至資料庫中。協助本研究進行受測者針對每個群組調整之後的使用者真實回饋資訊進行比較分析，如圖5-3所示。

Facebook 社群網站平台之訊息功能

…

我

受測者

…

**1**

**2**

朋友分群系統

網站

資料庫

**3**

**5**

**4**

圖5-3、實驗設計相關步驟示意圖

(資料來源：本研究整理)

雖然實驗設計皆能直接在網際網路中完成，可是受測者最在意的問題皆為會不會取得個人隱私資訊和會不會改變 Facebook 社群網站設定。此時答案皆為不會，朋友分群系統並不會取得個人隱私資訊，主要取得受測者相關朋友的資訊和調整之後有關查準率和查查率的資訊，以及不會改變 Facebook 社群網站設定，可是受測者可以針對以考量互斥因子之自動分群機制產生群組進行調整，並且直接產生至 Facebook 當成朋友名單。

最後當受測者確認且調整所有群組之後，就會產生群組相關資訊的圖表，且同時將相關的數據傳送至資料庫，也就是 Google 文件中，以協助本研究實驗數據的取得，如表5-1所示。

表5-1、受測者實驗調查資訊

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **朋友編號** | **朋友數** | **朋友 關係數** | **互斥 條件數** | **產生**  **群組數** | **查準率**  **(Precision)** | **查全率**  **(Recall)** |
| 朋友1 | 41 | 325 | 1 | 3 | 85.71% | 94.44% |
| 朋友2 | 44 | 229 | 1 | 4 | 97.22% | 95.83% |
| 朋友3 | 91 | 627 | 3 | 4 | 73.16% | 72.78% |
| 朋友4 | 167 | 1296 | 1 | 8 | 93.47% | 93.49% |
| 朋友5 | 187 | 1641 | 1 | 5 | 100.00% | 100.00% |
| 朋友6 | 194 | 1274 | 3 | 12 | 91.90% | 95.27% |
| 朋友7 | 196 | 1112 | 3 | 10 | 85.46% | 93.13% |
| 朋友8 | 229 | 2482 | 3 | 8 | 87.27% | 90.54% |
| 朋友9 | 236 | 2145 | 3 | 9 | 98.95% | 98.86% |
| 朋友10 | 237 | 2035 | 1 | 6 | 86.35% | 91.06% |
| 朋友11 | 263 | 1646 | 3 | 10 | 100.00% | 100.00% |
| 朋友12 | 263 | 3239 | 3 | 7 | 100.00% | 100.00% |
| 朋友13 | 308 | 2525 | 3 | 12 | 83.67% | 91.39% |
| 朋友14 | 333 | 4074 | 1 | 9 | 100.00% | 100.00% |
| 朋友15 | 352 | 2877 | 3 | 17 | 88.09% | 93.83% |
| 朋友16 | 359 | 5127 | 3 | 8 | 92.27% | 92.59% |
| 朋友17 | 359 | 3654 | 2 | 14 | 95.32% | 98.10% |
| 朋友18 | 385 | 4793 | 2 | 9 | 89.02% | 86.59% |
| 朋友19 | 418 | 5378 | 3 | 13 | 96.46% | 96.52% |
| 朋友20 | 433 | 6057 | 3 | 8 | 100.00% | 100.00% |
| 朋友21 | 473 | 4203 | 3 | 12 | 83.85% | 87.57% |
| 朋友22 | 494 | 4367 | 3 | 16 | 100.00% | 100.00% |
| 朋友23 | 544 | 10035 | 3 | 9 | 96.04% | 92.35% |
| 朋友24 | 566 | 7660 | 3 | 12 | 95.16% | 95.98% |
| 朋友25 | 597 | 8732 | 3 | 12 | 87.75% | 87.05% |
| 朋友26 | 617 | 9108 | 2 | 14 | 71.71% | 82.55% |
| 朋友27 | 648 | 8320 | 3 | 15 | 75.80% | 84.03% |
| 朋友28 | 687 | 16907 | 3 | 16 | 95.41% | 94.37% |
| 朋友29 | 795 | 9800 | 1 | 20 | 72.80% | 75.52% |
| 朋友30 | 854 | 10156 | 2 | 18 | 89.63% | 95.52% |



(資料來源：本研究整理)

## 評估分析

本研究主要探討使用者對於分群結果的回饋資訊，透過資訊檢索領域中廣泛被應用的查準率 (precision) 和查全率 (recall) 進行分析比較 [16]，在針對社群網路進行分群的相關研究中，有少數學者也會透過查準率和查全率進行分析比較  
，像是 Chen 學者等人針對小規模真實社群網路、Mislove 學者等人針對大規模的Facebook社群網路，以及Grob 學者等人針對行動社群網路進行分群，其中皆有透過查準率和查全率進行比較分析 [17][18][19]。透過實驗結果得知考量互斥因子之自動分群機制所產生的群組，根據受測者針對群組實際的回饋資訊，各個群組之查準率平均為 90.42% ，以及查全率平均為 92.65%，如圖5-4所示。因此透過考量互斥因子之自動分群機制，只需根據朋友彼此之間的朋友關係進行分群，則查準率平均為 92.65% 和查全率平均為 90.42%，代表本研究所提出的考量互斥因子之自動分群機制能除了能有效檢索有 92.65% 與群組中相關的資訊，其中有 90.42% 的資訊是群組中準確的資訊。

圖5-4、受測者的回饋資訊圖表

(資料來源：本研究整理)

考量互斥因子之自動分群機制會隨著朋友數與查準率和查全率呈反比的關係，當朋友數越少時，則查準率和查全率越高，當朋友數越多時，則查準率和查全率越低，如圖5-4所示。且在 2012 年McAuley 學者等人針對 Facebook、Google + 和Twitter個人網路透過機器學習的方式進行社群偵測的分析之後觀察出查全率會比查準率高，其查全率為還原在每個群組中的朋友和查準率為還原在群組建立後加入的朋友，與本研究透過數據也觀察出查全率會比查準率高，且該篇論文主要是透過 F 度量針對 Facebook 社群網路進行群組的評估為59%，而本研究若透過查全率和查準率計算出 F 度量則高達 91% ，因此會較適合應用至 Facebook 社群網路 進行分群的應用 [6]。

此外在這次的研究實驗中將總共 320 個不同朋友數的群組，這時根據使用者回饋資訊進行查準率和查全率的統計分析，發現不同朋友數之群組的回饋資訊之查全率和查準率皆至少達80% 以上，且查全率平均皆比查準率高，代表根據朋友關係進行分群之後的群組中的朋友會有相關，但屬於該群組的朋友，在現實生活中的準確度還是會有些誤差，如圖5-5所示。

圖5-5、不同朋友數之群組的回饋資訊圖表

(資料來源：本研究整理)

最後根據受測者之朋友數產生分佈圖表，間隔範圍以 100 人為單位，可以發現累積的個數會呈現常態分布，30位受測者之朋友數之眾數為 263 、中位數為 356 和平均數為 379 ，因為眾數 < 中位數 < 平均數，所以屬於正偏態的常態分佈，其右側的尾部更長，分布的主體集中在左側，非常接近在 2013 年Wolfram學者根據大量 Facebook 社群網站中之使用者的資訊所分析出的朋友數分布圖，如圖 5-6 所示 [56]。

圖5-6、受測者之朋友數分布圖表

(資料來源：本研究整理)

1. **結論與未來建議**
   1. 結論

　　本研究為了解決當使用者在Facebook 社群網站中分享資訊時，將會需要將兩位朋友屬於不同的群組，避免產生衝突的問題，主要在比較多種針對社群偵測不同的分群演算法，根據衡量群組適當性的評估指標 Q 值、階層分群方法和時間複雜度，最後選擇以最大化模組性為主之聚合階層方法，且時間複雜度接近線性的 BGLL 分群演算法，接著本研究考量使用者回饋的資訊，將傳統 BGLL 分群演算法進行改良，提出考量互斥因子之自動分群機制。

首先透過 Zachary 學者針對社群網路分析的研究，整理出空手道俱樂部小規模真實社群網路之公開資料集進行比較分析，確認考量互斥因子之自動分群機制透過適當的條件設定能夠比 BGLL分群演算法產生符合條件組更適當的群組，接著若要符合真實的情況，則需要在進行後續的調整，使得群組中的朋友數皆至少會大於某個數值，除了能更符合真實的情況之外，更能方便使用者進行調整。

　　在透過公開資料集分析中確認考量互斥因子之自動分群機制，除了能有效分成適當的群組，且接著再透過後續的處理，更能使得群組符合真實的情況。可是若將考量互斥因子之自動分群機制應用至 Facebook 社群網站中，當設定多組兩位朋友屬於不同的群組條件之後，此時產生出的群組是否就能夠符合使用者的需求，所以需要受測者針對每個群組進行調整，這將會為受測者帶來非常的困擾，所花費的時間成本也非常高，且目前也沒有適當的軟體能針對分群結果產生的所有群組，以所見即所得的方式進行操作，降低受測者操作時的困擾。

因此本研究為了解決受測者在進行實驗時所會面臨的困擾，讓實驗進行的更順利，實作出朋友分群系統透過所見即所得的方式降低使用者操作時的困擾，在確保數據的真確性的前提下，大量縮短實驗進行時間，並且透過線上的朋友進行系統操作的方式，能不受實驗地點的限制進行相關的實驗。此外取得 Facebook 社群網站中受測者的朋友資訊主要是透過 OAuth 授權協定，取得使用者的同意後才進行資料的分析，最後傳送的資訊主要是以計算後的數值為主，並未取得任何個人隱私的資訊。可是當在進行實驗時，還是有許多受測者會提出朋友分群系統是否會更改 Facebook 社群網站中的設定，待確認之後才會同意協助實驗，因此過程中就算透過系統還是需要花額外的時間成本進行溝通，讓受測者放心的使用朋友分群系統協助本研究實驗的進行。

接著受測者會透過朋友分群系統，在讀取資訊之後會進行朋友互斥條件得設定，再產生多個群組，此時許多受測者一開始會搞不清楚各個群組所代表的意義，待調整後才慢慢了解不同群組所代表的意義，反應大部份是還分的很準，但還是有受測者對分群結果產生質疑，明明在現實生活中就沒有分到這麼多組，或者應該要再細分出幾組，可是這是本研究的實驗限制，目前並無法提供受測者進行新增和刪除的調整，且還是會存在些誤差在於許多對於受測者來說只有一面之交的朋友必須被調整至較適當的群組或整理至相同的群組中，此時就可能會使得查準率降低，且根據分群演算法的特性會使得查全率提高。

最後根據受測者回饋的資訊發現本研究所提出的考量互斥因子之自動分群機制只需根據使用者回饋資訊，像是可能會發生衝突的朋友之互斥，因此依據朋友之間衝突關係，提出考量互斥因子之自動分群機制進行分群，則其查準率平均為90.42% 和查全率平均為 92.65%，總結透過考量互斥因子之自動分群機制產生的群組，其群組中有高達 92.65% 朋友是相關的資訊，群組中有高達 90.42% 朋友是準確的資訊，因此只需要透過使用者的朋友資訊所產生的群組中就有高達九成以上朋友是相關且準確的資訊。

* 1. 未來研究方向與建議

本研究目前針對考量互斥因子之自動分群機制，透過實作的系統進行所有群組平均查全率和查準率的比較分析，未來主要可以往以下三大研究方向進行發展  
，首先將互斥因子條件應用至其它分群演算法進行比較，接著如何透過互斥因子條件達到動態分群的探討，最後在多元情境應用中透過文字探勘的方式轉換為互斥因子條件。

1. 將互斥因子條件應用至其它分群演算法進行比較，可以直接應用至以模組性為基礎的 LPA 分群演算法和以非模組性為基礎的 SCAN 分群演算法進行探討，可是會面臨 LPA 分群演算法每次產生群組不固定的問題和 SCAN 分群演算法必須先決定 和 這兩個參數值該如何設定的問題。待問題解決後，改良 LPA 分群演算法和 SCAN 分群演算法應用互斥因子條件，接著直接將本研究的朋友分群系統進行修改，實作應用互斥因子條件的分群演算法，再以類似的實驗設計請受測者針對群組直接進行評估分析，最後透過資訊檢索領域中查準率和查全率和考量互斥因子條件之自動分群機制進行比較分析。
2. 如何透過互斥因子條件達到動態分群機制的探討，可以當使用者有需求時主動透過系統進行群組的更新，此時互斥因子條件的主要應用於當使用者每次調整好的群組進行資訊檢索中人工判斷與系統判斷的比較轉換為最少數量的互斥因子條件，接著只需透過儲存互斥因子條件就能透過考量互斥因子之自動分群機制產生使用者調整好的群組且同時完成群組更新。
3. 在多元情境應用中透過文字探勘的方式轉換為互斥因子條件，可以針對多元情境的應用問題透過互斥因子條件之自動分群機制解決，主要先透過文字探勘的方式找出關聯資訊，接著再將關聯資訊轉換成互斥因子條件，此時再根據互斥因子條件進行分群就能有效縮小相關且準確的資訊範圍。

# **參考文獻**

1. 創市際公佈「2012年網路社群白皮書」調查報告. (2012).  
   <http://www.insightxplorer.com/news/news_12_18_12.html>
2. Li, W. (2012). Research on the design of social spheres in social network sites.
3. Facebook 使用說明中心 (2013).  
   <https://www.facebook.com/help/>
4. Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. Journal of Computer‐Mediated Communication, 13(1), 210-230.
5. Ugander, J., Karrer, B., Backstrom, L., & Marlow, C. (2011). The anatomy of the facebook social graph. arXiv preprint arXiv:1111.4503.
6. McAuley, J., & Leskovec, J. (2012). Learning to discover social circles in ego networks. In Advances in Neural Information Processing Systems 25 (pp. 548-556).
7. Staddon, J., Huffaker, D., Brown, L., & Sedley, A. (2012). Are privacy concerns a turn-off?: engagement and privacy in social networks. In Proceedings of the Eighth Symposium on Usable Privacy and Security (p. 10). ACM.
8. Liu, Y., Viswanath, B., Mondal, M., Gummadi, K. P., & Mislove, A. (2012). Simplifying friendlist management. In Proceedings of the 21st international conference companion on World Wide Web (pp. 385-388). ACM.
9. Papadopoulos, S., Kompatsiaris, Y., Vakali, A., & Spyridonos, P. (2012). Community detection in social media. Data Mining and Knowledge Discovery, 24(3), 515-554.
10. Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008.
11. Chaturvedi, P., Dhara, M., & Arora, D. (2012). Community Detection in Complex Network via BGLL Algorithm. International Journal of Computer Applications.
12. Lampinen, A., Tamminen, S., & Oulasvirta, A. (2009). All my people right here, right now: management of group co-presence on a social networking site. In Proceedings of the ACM 2009 international conference on Supporting group work (pp. 281-290). ACM.
13. Skeels, M. M., & Grudin, J. (2009). When social networks cross boundaries: a case study of workplace use of facebook and linkedin. In Proceedings of the ACM 2009 international conference on Supporting group work (pp. 95-104). ACM. Jones, S., & O'Neill, E. (2010, July).
14. Agarwal, G., & Kempe, D. (2008). Modularity-maximizing graph communities via mathematical programming. The European Physical Journal B, 66(3), 409-418.
15. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval (Vol. 1). Cambridge: Cambridge University Press.
16. van Rijsbergen, C. J. (1977). A theoretical basis for the use of co-occurrence data in information retrieval. Journal of documentation, 33(2), 106-119.
17. Chen, J., Zaïane, O., & Goebel, R. (2009). Local community identification in social networks. In Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in (pp. 237-242). IEEE.
18. Grob, R., Kuhn, M., Wattenhofer, R., & Wirz, M. (2009). Cluestr: mobile social networking for enhanced group communication. In Proceedings of the ACM 2009 international conference on Supporting group work (pp. 81-90). ACM.
19. Mislove, A., Viswanath, B., Gummadi, K. P., & Druschel, P. (2010). You are who you know: inferring user profiles in online social networks. In Proceedings of the third ACM international conference on Web search and data mining (pp. 251-260). ACM.
20. Wasserman, S., & Faust, K. (1994). Social network analysis: Methods and applications (Vol. 8). Cambridge university press.
21. Chung, K. K., Hossain, L., & Davis, J. (2005). Exploring sociocentric and egocentric approaches for social network analysis. In Proceedings of the 2nd international conference on knowledge management in Asia Pacific.
22. Tang, L., & Liu, H. (2010). Community detection and mining in social media. Synthesis Lectures on Data Mining and Knowledge Discovery, 2(1), 1-137.
23. JIS, M. T. C. S. (1979). Computers and intractability A Guide to the Theory of NP-Completeness.
24. Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., & Wagner, D. (2008). On modularity clustering. Knowledge and Data Engineering, IEEE Transactions on, 20(2), 172-188.
25. Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 99(12), 7821-7826.
26. Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. Journal of anthropological research, 452-473.
27. Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. Physical review E, 69(2), 026113.
28. Newman, M. E. (2004). Fast algorithm for detecting community structure in networks. Physical review E, 69(6), 066133.
29. Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. Physical review E, 70(6), 066111.
30. Ovelgönne, M., Geyer-Schulz, A., & Stein, M. (2010). 0 greedy modularity optimization for group detection in huge social networks. In SNA-KDD’10: Proceedings of the 4th Workshop on Social Network Mining and Analysis.
31. Smyth, S., & White, S. (2005). A spectral clustering approach to finding communities in graphs. In Proceedings of the fifth SIAM international conference on data mining (Vol. 119, p. 274).
32. Fortunato, S., & Barthelemy, M. (2007). Resolution limit in community detection. In Proceedings of the National Academy of Science (Vol. 104, pp. 36-41).
33. Xu, X., Yuruk, N., Feng, Z., & Schweiger, T. A. (2007). SCAN: a structural clustering algorithm for networks. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 824-833). ACM.
34. Leung, I. X., Hui, P., Liò, P., & Crowcroft, J. (2009). Towards real-time community detection in large networks. Physical Review E, 79(6), 066107.
35. Jin, D., Liu, D., Yang, B., Liu, J., & He, D. (2011). Ant colony optimization with a new random walk model for community detection in complex networks. Advances in Complex Systems, 14(05), 795-815.
36. Nadkarni, A., & Hofmann, S. G. (2012). Why do people use Facebook?. Personality and individual differences, 52(3), 243-249.
37. DiMicco, J. M., & Millen, D. R. (2007). Identity management: multiple presentations of self in facebook. In Proceedings of the 2007 international ACM conference on supporting group work (pp. 383-386). ACM. Hewitt, A., & Forte, A. (2006).
38. Wilson, R. E., Gosling, S. D., & Graham, L. T. (2012). A review of Facebook research in the social sciences. Perspectives on Psychological Science, 7(3), 203-220.
39. Das, S., & Kramer, A. (2013). Self-Censorship on Facebook.
40. Lusseau, D. (2003). The emergent properties of a dolphin social network. Proceedings of the Royal Society of London. Series B: Biological Sciences, 270(Suppl 2), S186-S188.
41. Gleiser, P. M., & Danon, L. (2003). Community structure in jazz. Advances in complex systems, 6(04), 565-573.
42. Guimera, R., Danon, L., Diaz-Guilera, A., Giralt, F., & Arenas, A. (2003). Self-similar community structure in a network of human interactions. Physical review E, 68(6), 065103.
43. Ferrara, E. (2012). A large-scale community structure analysis in Facebook. EPJ Data Science, 1(1), 1-30.
44. Gregory, S. (2010). Finding overlapping communities in networks by label propagation. New Journal of Physics, 12(10), 103018.
45. Wu, Z., Lin, Y., Wan, H., Tian, S., & Hu, K. (2012). Efficient overlapping community detection in huge real-world networks. Physica A: Statistical Mechanics and its Applications, 391(7), 2475-2490.
46. Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. Physical Review E, 78(4), 046110.
47. Fortunato, S., & Lancichinetti, A. (2009). Community detection algorithms: a comparative analysis: invited presentation, extended abstract. In Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools (p. 27). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
48. Steinhaeuser, K., & Chawla, N. V. (2010). Identifying and evaluating community structure in complex networks. Pattern Recognition Letters, 31(5), 413-421.
49. Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association, 66(336), 846-850.
50. Hubert, L., & Arabie, P. (1985). Comparing partitions. Journal of classification, 2(1), 193-218.
51. Ana, L. N. F., & Jain, A. K. (2003). Robust data clustering. In Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on (Vol. 2, pp. II-128). IEEE.
52. Jones, S., & O'Neill, E. (2010). Feasibility of structural network clustering for group-based privacy control in social networks. In Proceedings of the Sixth Symposium on Usable Privacy and Security (p. 9). ACM.
53. Hammer-Lahav, E., Recordon, D., & Hardt, D. (2011). The OAuth 2.0 authorization protocol. Network Working Group Internet-Draft.
54. Triola, M. F. (2008). Essentials of statistics. Boston: Pearson Addison Wesley.
55. Nielsen, J. (1994). Usability engineering. Access Online via Elsevier.
56. DATA SCIENCE OF THE FACEBOOK WORLD (2013).  
    <http://blog.wolframalpha.com/2013/04/24/data-science-of-the-facebook-world/>

