科技部補助專題研究計畫成果報告
期末報告

搜尋 k-clique community 之研究

計 畫 類 別 ： 個別型計畫
計 畫 編 號 ： MOST 104-2221-E-343-004-
執 行 期 間 ： 104年08月01日至105年07月31日
執 行 單 位 ： 南華大學資訊工程學系

計 畫 主 持 人： 金家豪

中 華 民 國 105 年 10 月 24 日

中 文 摘 要 ： 真實世界網路中普遍存在社群結構，而偵測網路中的社群可以幫助我們更了解真實世界的運作。例如：偵測社群可以幫助我們在人際關係網路中找出社會群體、在蛋白質交互作用網中找出蛋白質複合體。現今已有許多偵測社群的方法，其中 CPM (Clique Percolation Method) 是一種常被使用的偵測方法。CPM 找到的社群被稱之為 k-clique community，它是由一群 k-clique 所聯集，其中任何一個 k-clique 可經由一連串相鄰 k-clique 到達其他處在相同社群的 k-clique，而兩個 k-clique 相鄰代表這兩個大小為 k 的clique 共用 k-1 個節點。

CPM 所需的計算時間較長導致它無法處理大型網路，因此已有人提出如何減少計算尋找 k-clique community 所需時間。我們參考前人的作法設計出一個平行演算法以加速尋找 k-clique community，實驗顯示我們方法的整體效能比前人方法來得好但其缺點為對於記憶體的要求較高，因此我們希望在這個成果下繼續改進。此外，我們將嘗試把空間資訊加入 CPM的方法中，例如：在人際關係網路中考慮人們所在的地理位置或在蛋白質交互作用網路中加入蛋白質在細胞內之位置資訊，以提升偵測網路社群的準確性。

中 文 關 鍵 詞 ： 網路分析；偵測社群結構

英 文 摘 要 ： Many problems can be represented as graph theory problems, such as human relationship network analysis in social studies and detecting protein complexes in Protein-Protein Interaction networks in biological studies. We can construct a network by adding nodes to represent objects in the original problem and linking any two nodes with an edge if there is a relationship between the two objects. After we create networks and transfer the original problems to graph theory problems, the problems can be viewed as graph theory problems and solved by graph theory’s skills.

Community, in which nodes are joined tightly together, exists in many real network networks. Detecting community in a network is a very important research topic, because it has many practical applications. For example, detecting communities can help us find out real social groupings in a social network, related papers on a single topic in a citation network, protein complexes in a Protein-Protein Interaction network and web pages on related topics in the internet. Therefore, detecting communities in real networks can help to understand how real world works.

Because detecting community in a network is a very important research topic, there are many detecting community methods are developed. Among these methods, CPM (Clique Percolation Method) is a widely used method. CPM detects communities by finding k-clique community which is

the union of all cliques of size k that can be reached through adjacent (sharing k-1 nodes) k-cliques. Finding k-clique communities is a time-consuming task. To solve this problem, we developed a parallel algorithm to reduce the running time of finding k-clique communities. Experimental results showed that our method outperforms previous ones, however, our method is high memory-demanded and we want to improve it in this project. Besides, we will try to use objects' localization information to help us detect community structure from networks.

英文關鍵詞：網路分析；偵測社群結構

一、前言

對於真實世界這些看似無序的複雜網路(例如: 人際關係網路、電力網路、網際網路網頁的連結網、蛋白質交互作用網路)，經由科學家研究後發現有許多共同特性存在[1, 2]，例如：節點的鄰居間彼此成為鄰居的機會很高，若我們從人際關係網路的表現來看，也就是某個人的朋友間彼此會相互認識的機會通常比茫茫人海中隨意選取兩人而這兩人恰好相互認識的機會來得高，這種「物以類聚」的特性造就出複雜網路中的社群，也就是說處於相同社群之成員的互動遠比不同社群成員間的互動來得高。

二、研究目的

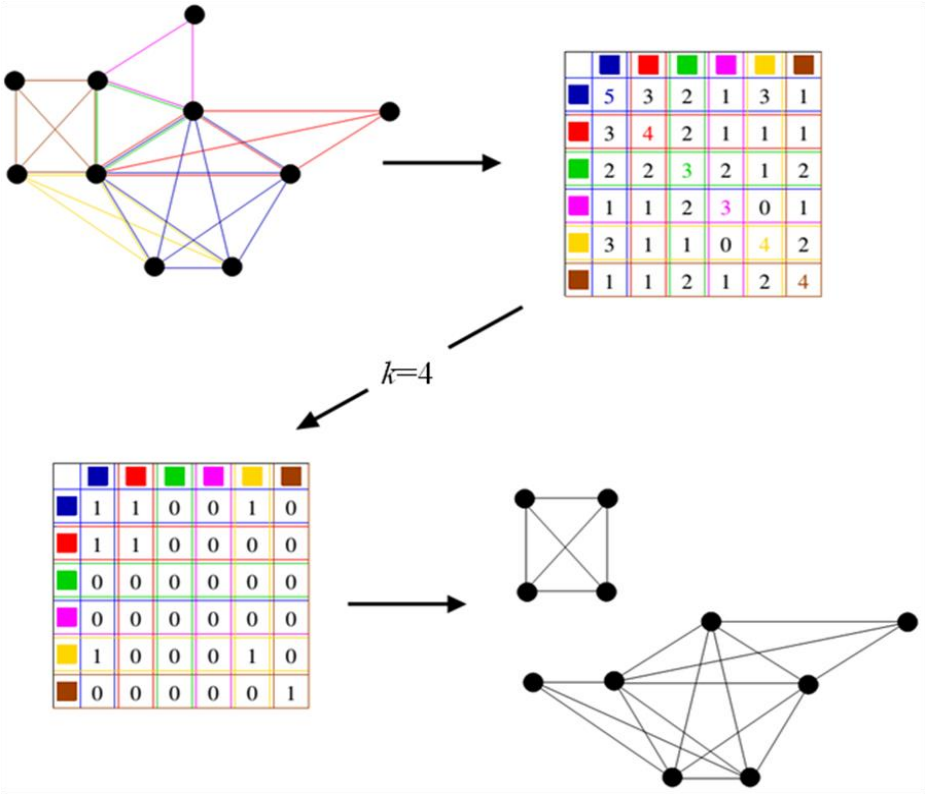
我們過去曾利用複雜網路的特性協助預測必要性蛋白質[3]與蛋白質複合體[4, 5]，由於蛋白質常與功能相近的蛋白質形成複合體，以合作的方式以執行生物功能，因此可藉由偵測蛋白質交互作用網路中的社群結構(Community Structure)來預測蛋白質複合體，而偵測社群結構方法已有許多人提出[6-8]，但我們認為這些偵測方法只是針對社群的單一特性進行設計所以只能看到面片的結果，於是我們透過整合六種偵測社群的方法[5, 9-13]以得到更好的偵測結果[4]。在實作的過程中發現在這些方法中，CPM (Clique Percolation Method)[11]的預測能力最好但也最耗時，導致它無法在較大的網路上進行偵測，因此我們希望設計出一個平行演算法以加速尋找 *k*-clique community。

三、文獻探討

以下將簡介 CPM 並說明為什麼它的時間複雜度會這麼高，CPM 是由 Palla 等人在 2005 於 Nature 所提出的一種透過找出 *k*-clique community 以偵測社群的方法，在解釋什麼是 *k*-clique community 前，容我先定義什麼是「相鄰」的 *k*-clique，若我們說兩個 *k*-clique 是相鄰的，這意謂者這兩個大小為 *k* 的 clique 有重疊 *k*-1 個節點，而 *k*-clique community 是由一群 *k*-clique 所聯集，其中任何一個 *k*-clique 可經由一連串的相鄰 *k*-clique 到達其他處在相同社群的 *k*-clique。顯而易見的，一個節點數大於或等於 *k* 的 maximal clique 必定是某個 *k*-clique community 的子集，所以在圖 $G=(V, E)$ 找 *k*-clique community 的問題可轉成在圖 $G_k=(V_k, E_k)$ 上找 connected component，其中 V_k 的成員是由圖 G 中節點數大於或等於 *k* 的 maximal clique 所構成；如果 u 和 v 屬於 V_k 且 u 和 v 對映到圖 G 的 maximal clique 共用 *k*-1 (含)以上的節點則邊 (u, v) 屬於 E_k。

為了讓大家能更清楚了解 CPM 這個方法，接下來用個簡單範例說明它是如何運作的。如圖一的左上角所示，原圖 G 中存在用 6 種顏色代表不同的 maximal clique，這些 maximal clique 間的重疊數如圖一右上角的 clique-clique overlapping matrix(CCOM)所示，若我們要找出圖 G 中的 4-clique community，則可先利用 CCOM 建出圖 G_4 的鄰接矩陣(如圖一左下角所示)，再來找出圖 G_4 中的 connected component 並回推出最後的結果。而 G_4 的鄰接矩陣

之建法為若 CCOM 對角線元素大於 3 的話，則 G_4 的鄰接矩陣相對映元素則填 1 否則填 0；在非對角線元素部份，若 CCOM 的元素大於 2 的話，則對映的 G_4 鄰接矩陣元素填 1 否則填 0。

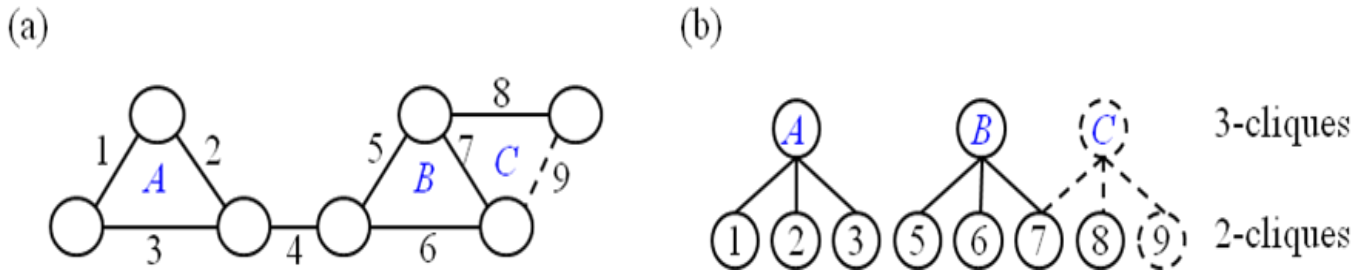


圖一: 找尋 4-clique community 的範例[11]

找出圖形中全部的 maximal clique 是 NP-complete 的問題，但所幸在真實世界網路中找 maximal clique 所需時間並沒有想像中那麼久[14]，找 k-clique community 真正瓶頸在於新圖 $G_k=(V_k, E_k)$ 遠比原圖 G 大很多。假設在原圖 $G=(V, E)$ 中有 $O(|V|^{\alpha})$ 個 maximal clique，光是建構出新圖就要花去 $O(|V|^{2\alpha})$ 的時間，除此之外還必須加上找 connected component 的時間，因此 CPM 需要大量的時間才有辦法找出 k-clique community。由於 CPM 的應用十分廣泛(註: 原始論文被引用次數超過 3303 次)但需要大量計算時間，所以近年來有學者研究如何加速計算 k-clique community[15-18]。

以下將簡要說明前人如何加速計算 k-clique community。我們目前查到最早一篇的改進的方法為 2008 年 Kumpula 等人所提出[15]，他們提出一個名叫 Sequential Clique Percolation 的方法(簡稱為 SCP)， 為了加速計算 k-clique community，SCP 不計算圖 G_k 的 connected component 改而計算由原圖 G 對映出的 bipartite graph 之 connected component，對映方法為 bipartite graph 的兩大節點集合分別對映到圖 G 的 k-clique 與 (k-1)-clique 所構成的兩集合，而邊代表其端點所對映的 clique 有包含的關係，透過找尋這個 bipartite graph 的 connected component 就可幫忙找出圖 G 的 k-clique community。以下將以圖二為例來說明 SCP 如何找 3-cliuqe community，首先 SCP 將圖 G 中的 2-clique(也就是邊)進行編號然後全部移除再依序插入，在插邊的過程中同時偵測 clique 的產生，舉例來說，如圖二(a) 當插入第 9 個邊(虛線)時，在圖二(b) 的 bipartite graph 會產生一個由三個 2-cliuqe 所構成的 3-cliuqe，

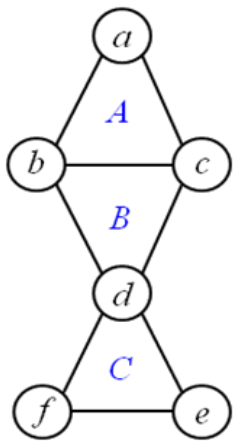
也就是圖二(b)虛線的部份，SCP 利用這種方式產生出 bipartite graph 進而找到圖 G 的 3-cliuqe community。SCP 用在稀疏的圖形上確實可以縮短計算時間，但當網路中有許多大 clique 時，則 k-clique 與 (k-1)-clique 的數量會爆增導致計算時間大幅增加。除此之外，當 k 變大時，SCP 要偵測 k-clique 的時間也要增加，這也造成 SCP 在 k 值變大時效能不佳。



圖二: 說明 SCP 如何找圖 G 的 3-cliuqe community 的範例

為了克服 SCP 的缺點，Reid 等人回到計算圖 G_k 的 connected component，但他們的方法並不會產生整個圖 G_k 而是只產生建 G_k 的 spanning forest 所需的邊[19]。此外，他們分別透過利用兩種資料結構以縮短找尋 maximal clique 鄰居的時間，第一種為 node-to-clique 資料表，該表對每個節點記錄有那些 maximal cliques 包含該節點，透過這個資料表以加速確認 clique 間的重疊數量。以下我將用圖三闡明它的用法，圖三(a)中有 7 個節點和 3 個 maximal clique，在 k=3 時假設我們要找 clique A 的鄰居時，則只需要看有包含 A 的成員(即 a, b, c)的這些 maximal clique，也就是說只要計算 a, b, c 在 node-to-clique 表中 maximal cliques 出現次數，若某個 maximal cliques(除了 A 以外)在圖三(c) 出現次數大於 1 時則代表該 maximal clique 即為 A 的鄰居。透過這種方式可大量減少不必要的檢查進而減少計算時間。Reid 等人的第二個方法為建一顆 complete binary tree，這顆樹的葉子(leaves)對映到原圖的 maximal clique，而 common ancestor 則對映 maximal cliques 的共同部份，當我們要檢查某個給定的 maximal clique 與那些 maximal cliques 重疊時，可先檢查指定的 maximal clique 與 internal nodes 的重疊以減少檢驗的時間。

(a)



(b)

Node	Maximal cliques
a	A
b	A, B
c	A, B
d	B, C
e	C
f	C

(c)

	A	B	C
a	1	0	0
b	1	1	0
c	1	1	0
總合	**3**	**2**	**0**

圖三: 說明如何利用 node-to-clique 找到 maximal clique A 所有鄰居

上述加速計算 k-clique community 方法都屬於循序演算法(serial algorithm)，另一類型的

方式為透過平行演算法(parallel algorithm)以加速計算[16, 17]。Pollner 等人於 2012 年所提出一個直覺的平行演算法[16]，其做法為把一張大圖分割成 n 張小圖，然後再把這些小圖分派給 n 個處理器計算，最後再把各自計算的結果匯整成正確的結果。Gregori 等人於 2013 年提出另一種名為 COS (CPM On Steroids) 的平行演算法，其作法為透過多個處理器共同計算出圖 G_k 的 connected component 以改善計算的效率[17]。本研究是主要基於 COS 的平行演算法再發展下去。

四、研究方法

首先，我們把 CPM 分成以下四大部份：

一、讀入圖形並做前置處理。

二、找出圖形中所有大小大於等於 3 的 clique。

三、找出 k-clique community 對映 k-clique disjoint sets。

四、把 k-clique disjoint sets 轉成 k-clique community 並輸出。

以下針對上述大四部份分別進行說明。在第一大部份，由於我們處理的圖形是 Simple Graph 且 undirected 和 unweighted，所以在讀入圖檔時，會把重覆的邊和 self loop 的邊忽略，並把節點進行編號(即 $\{v_0, \ldots, v_{n-1}\}$) 後建置 adjacency lists。在第二大部份，要找出圖形中所有大小大於等於 3 的 clique，一般常用找全部 clique 的方法是 Bron–Kerbosch algorithm，但由於我們只要找大小大於等於 3 的 clique，所以我們修改 Bron–Kerbosch algorithm，我們的演算法如下：

Input: $G(V, E)$
Output: $c_0,\ldots c_{l-1}$ // c_i=list of nodes in the i-th maximal clique in G
Chin(G):
    foreach edge (v_i, v_j) of G **do**
        R:={ v_i, v_j }
        $P := N(v_i) \cap N(v_j) \cap \{v_{j+1}, \ldots, v_{n-1}\}$
        $X := N(v_i) \cap N(v_j) \cap \{v_0, \ldots, v_{j-1}\}$
        BronKerbosch(R, P, X)

BronKerbosch(R, P, X):
    if P and X are both empty **do**
        report R as a maximal clique
    choose a pivot vertex u in $P \cup X$
    foreach vertex v in $P \setminus N(u)$:
        BronKerbosch($R \cup \{v\}$, $P \cap N(v)$, $X \cap N(v)$)
        $P := P \setminus \{v\}$
        $X := X \cup \{v\}$

利用前二大部的結果，第三部份我們可以開始找 *k*-clique community，這部份主要是參考中央資工何錦文教授指導的碩士班研究生鄭豐叡先生的研究成果[20]；而本研究所提出的方法如下所示，它和鄭豐叡先生的方法主要的不同點在於紅色的部份，其改進的理論基礎為在找 *k*-clique community 時，我們的 *k* 是由大往小找，也就是先找較密再找較疏的 *k*-community，若我們已知在較密的 *k*-community 中兩個 clique 是標註連在一起，則表示較疏的 *k*-community 中兩個 clique 已經標註連在一起，因此可以找 *k*-community 節省時間。

Input: $c_0,\ldots c_{l-1}$ // c_i=list of nodes in the i-th maximal clique in G
Output: (k_{max} − 1) collections of disjoint sets F_k,
$k \in [2, k_{max}]$, corresponding to the k-clique communities of G

Chin($c_0,\ldots c_{l-1}$):
all processors q s.t. $q \in [0, p\text{-}1]$ **do in parallel**
foreach $k \in [2, k_{max}\text{-}1]$ **do**
$F_{q,k} \leftarrow <$ L_k singleton sets $\{0\}, \ldots, \{L_k\text{-}1\} >$
foreach $i \in [0, l\text{-}1]$ s.t. $i \bmod p = q$ **do**
for $j \leftarrow i + 1$ **to** $l-1$ **do**
for $k \leftarrow$ OVERLAP(c_i,c_j) **to** 2 **do**
if $F_{q,k}[i] = F_{q,k}[j]$ **then**
break
else
$\text{UNION}_{q,k}(i, j)$
foreach $k \in [2, k_{max}\text{-}1]$ **do**
foreach $q \in [1, p\text{-}1]$ **do**
foreach $i \in [0, l\text{-}1]$ **do**
if $F_{q,k}[i]$!= i **then**
$\text{UNION}_{0,k}(F_{q,k}[i] , i)$
Return $F_{q,k}$, $k \in [2, k_{max}\text{-}1]$

在第三部份的輸出中，我們取得 $G_k=(V_k, E_k)$ 上 connected component 所對映的 disjoint set，再把第二部份的結果找出 Clique 所包含的在原圖中節點編號，最後再依第一部份節點編號的方式反推其節點名稱，並去除重覆節點，即可得到 *k*-community。

五、結果與討論.

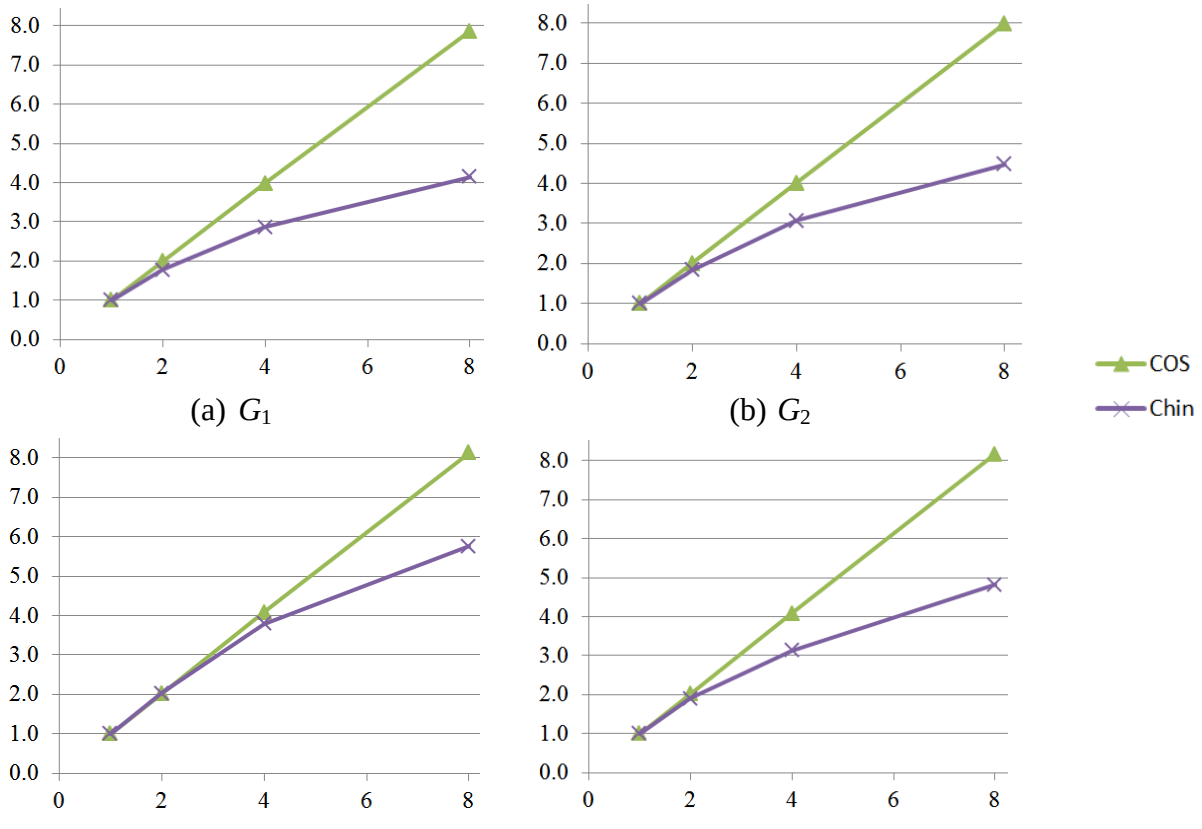
為了驗證我們的方法比 COS 好，我們在 LINUX 系統下，用 C 語言實作我們的演算法，演算法中平行化的部分是使用標準 POSIX thread 的 API 來控制多顆處理器，而執行環境是在一台有兩顆 Intel® Xeon® Processor E5-2620 的 CPU(每顆內含 6 個處理器核心，所以共計有 12 個處理器核心)，記憶體為 20GB 的伺服器。而我們測試的圖形其基本資料如下：

	G_1	G_2	G_3	G_4
$\|V\|$	5916	36692	7114	5349
$\|E\|$	71518	183831	100761	157244
k_{max}	31	20	17	33
l	71522	212789	450347	806973

下表為單處理器的環境下 COS 與我們方法執行時間的比較，我們的方法明顯比 COS 快的主因在於我們是採用 node-to-clique 表以降低計算 OVERLAP(c_i, c_j)的時間。

	G_1	G_2	G_3	G_4
COS	305.6	3602.7	15119.8	63515.9
Chin	8.2	50.5	248.4	947.2

下圖為比較 COS 與我們方法在平行化的表現，其中 x 軸為處理器數量， y 軸為以單處理器為基準除出來的執行倍率，綠線代表 COS 而紫線為我們的方法，下圖顯示 COS 的平行化表現比我們的方法好。



(a) G_1 (b) G_2

(c) G_3 (d) G_4

整體而言我們的方法的效能比 COS 好很多，但在平行度的表現上不如 COS，我們推測可能是因為我們的方法比 COS 需要多一個 node-to-clique 的資料表導致記憶體需求比 COS 高，當記憶體頻寬不足時則有些處理器核心因無法取得需要的資料進行計算，以致被迫進入閒置狀態，使得整體執行時間變長，針對這個問題，未來我們將朝如何減少使用記憶體或對記憶體有效分配進行改善。

六、參考文獻

1. Watts, D.J. and S.H. Strogatz, *Collective dynamics of 'small-world' networks.* Nature, 1998. **393**(6684): p. 440-442.
2. Barabási, A.-L. and R. Albert, *Emergence of Scaling in Random Networks.* Science, 1999. **286**(5439): p. 509-512.
3. Chin, C.H., et al., *cytoHubba: identifying hub objects and sub-networks from complex interactome.* BMC Systems Biology, 2014. **8**(Suppl 4): p. S11.
4. Chin, C.H., et al., *Spotlight: assembly of protein complexes by integrating graph clustering methods.* Gene, 2013. **518**(1): p. 42-51.
5. Chin, C.H., et al., *A hub-attachment based method to detect functional modules from confidence-scored protein interactions and expression profiles.* BMC Bioinformatics, 2010. **11 Suppl 1**: p. S25.
6. Ma, H.-S. and J.-W. Huang. *CUT: community update and tracking in dynamic social networks*. in *The 7th Workshop on Social Network Mining and Analysis (SNA-KDD'13) joint with the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'13)*. 2013: ACM.
7. Chang, C.-S., et al. *A general probabilistic framework for detecting community structure in networks*. 2011: IEEE INFOCOM 2011 Proceedings
8. Qi, G.-J., C.C. Aggarwal, and T. Huang. *Community detection with edge content in social media networks*. in *IEEE International Conference on Data Mining (ICDE)*. 2012: IEEE.
9. Newman, M.E.J., *Fast algorithm for detecting community structure in networks.* Physical Review E, 2004. **69**(6): p. 066133.
10. Van Dongen, S.M., *Graph clustering by flow simulation*. 2000, University of Utrecht.
11. Palla, G., et al., *Uncovering the overlapping community structure of complex networks in nature and society.* Nature, 2005. **435**(7043): p. 814-8.
12. Pons, P. and M. Latapy, *Computing communities in large networks using random walks*, in *Computer and Information Sciences-ISCIS 2005*. 2005, Springer. p. 284-293.
13. Reichardt, J. and S. Bornholdt, *Statistical mechanics of community detection.* Physical Review E, 2006. **74**(1): p. 016110.
14. Eppstein, D., M. Löffler, and D. Strash, *Listing all maximal cliques in sparse graphs in near-optimal time*, in *Algorithms and Computation*. 2010, Springer Berlin Heidelberg. p. 403-414.
15. Kumpula, J.M., et al., *Sequential algorithm for fast clique percolation.* Physical Review E, 2008. **78**(2).
16. Pollner, P., G. Palla, and T. Vicsek, *Parallel clustering with CFinder.* Parallel Processing Letters, 2012. **22**(01): p. 1240001.
17. Gregori, E., L. Lenzini, and S. Mainardi, *Parallel k-Clique community detection on large-scale networks.* IEEE Transactions on Parallel and Distributed Systems, 2013. **24**(8): p. 1651-1660.
18. Fu, C., et al. *K-clique community detection based on union-find*. in *2014 International Conference on Computer, Information and elecommunication Systems (CITS)*. 2014: 2014 International Conference on Computer.

19. Reid, F., A. McDaid, and N. Hurley, *Percolation Computation in Complex Networks*, in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*. 2012, IEEE Computer Society. p. 274-281.
20. 鄭豐叡, *在多核心電腦上實作偵測k-clique community 之平行演算法*, in *資訊工程學系*. 2014, 國立中央大學: 桃園縣.

科技部補助計畫衍生研發成果推廣資料表

日期:2016/10/02

科技部補助計畫	計畫名稱: 搜尋 k-clique community 之研究
	計畫主持人: 金家豪
	計畫編號: 104-2221-E-343-004- 學門領域: 生物資訊與醫療資訊

無研發成果推廣資料

104年度專題研究計畫成果彙整表

計畫主持人：金家豪	**計畫編號：**104-2221-E-343-004-
計畫名稱：搜尋 k-clique community 之研究	

成果項目					量化	單位	質化 （說明：各成果項目請附佐證資料或細項說明，如期刊名稱、年份、卷期、起訖頁數、證號...等）
國內	學術性論文	期刊論文			0	篇	
		研討會論文			0		
		專書			0	本	
		專書論文			0	章	
		技術報告			0	篇	
		其他			0	篇	
	智慧財產權及成果	專利權	發明專利	申請中	0	件	
				已獲得	0		
			新型/設計專利		0		
		商標權			0		
		營業秘密			0		
		積體電路電路布局權			0		
		著作權			0		
		品種權			0		
		其他			0		
	技術移轉	件數			0	件	
		收入			0	千元	
國外	學術性論文	期刊論文			0	篇	
		研討會論文			0		
		專書			0	本	
		專書論文			0	章	
		技術報告			0	篇	
		其他			0	篇	
	智慧財產權及成果	專利權	發明專利	申請中	0	件	
				已獲得	0		
			新型/設計專利		0		
		商標權			0		
		營業秘密			0		
		積體電路電路布局權			0		
		著作權			0		
		品種權			0		
		其他			0		

	技術移轉	件數	0	件	
		收入	0	千元	
參與計畫人力	本國籍	大專生	18	人次	
		碩士生	0		
		博士生	0		
		博士後研究員	0		
		專任助理	0		
	非本國籍	大專生	0		
		碩士生	0		
		博士生	0		
		博士後研究員	0		
		專任助理	0		
其他成果 （無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。）					

科技部補助專題研究計畫成果自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現（簡要敘述成果是否具有政策應用參考價值及具影響公共利益之重大發現）或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估
   ■達成目標
   □未達成目標（請說明，以100字為限）
   　□實驗失敗
   　□因故實驗中斷
   　□其他原因
   說明：

2. 研究成果在學術期刊發表或申請專利等情形（請於其他欄註明專利及技轉之證號、合約、申請及洽談等詳細資訊）
   論文：□已發表　□未發表之文稿　■撰寫中　□無
   專利：□已獲得　□申請中　■無
   技轉：□已技轉　□洽談中　■無
   其他：（以200字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性，以500字為限）
   本研究成果可協助生物學家探索蛋白質的功能與協助社會學家進行人際關係網路分析。

4. 主要發現
   本研究具有政策應用參考價值：■否　□是，建議提供機關
   （勾選「是」者，請列舉建議可提供施政參考之業務主管機關）
   本研究具影響公共利益之重大發現：■否　□是
   說明：（以150字為限）