

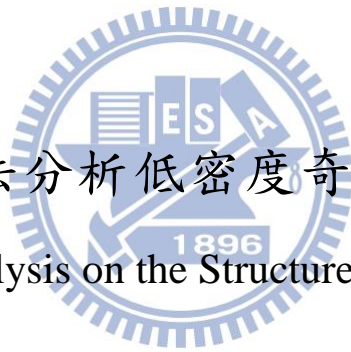
國立交通大學

生醫工程研究所

碩 士 論 文

利用分群演算法分析低密度奇偶檢查碼的結構

Clustering Analysis on the Structure of LDPC Codes



研 究 生：李亞錦

指 導 教 授：邵家健 副教授

中 華 民 國 1 0 2 年 9 月

利用分群演算法分析低密度奇偶檢查碼的結構
Clustering Analysis on the Structure of LDPC Codes


研究生：李亞錦

Student : Ya-Chin Li

指導教授：邵家健

Advisor : John Kar-Kin Zao

國立交通大學
生醫工程研究所
碩士論文

The logo of National Chiao Tung University is a circular seal. It features a gear-like outer border. Inside the circle, there is a stylized representation of a book and a lamp, with the letters 'ES' and 'A' visible. The year '1896' is inscribed at the bottom of the inner circle.

A Thesis
Submitted to Institute of Biomedical Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

September 2013

Hsinchu, Taiwan, Republic of China

中華民國 102 年 9 月

利用分群演算法分析低密度奇偶檢查碼的結構

學生：李亞錦

指導教授：邵家健 副教授

國立交通大學資訊學院生醫工程研究所

摘要

低密度奇偶檢查碼 (Low-density parity check codes, LDPC codes) 可以透過奇偶檢驗矩陣 (parity-check matrix) 表示，奇偶檢驗矩陣能夠利用 Tanner graph 圖形化顯示，但是效能好的碼從 Tanner graph 觀察不具有特定的結構特性。

本文利用分群演算法分析低密度奇偶檢查碼的結構，利用馬可夫分群演算法 (Markov cluster algorithm) 與模組性分群演算法 (Modularity cluster algorithm) 將 LDPC codes 的 nodes 分類，找到容易形成陷阱集合 (trapping set) 的小 cluster；並且透過網路參數 (Network parameter) 找出與 LDPC codes 解碼過程相關的參數。

關鍵字：低密度奇偶檢查碼、網路參數、馬可夫分群演算法、模組性分群演算法、陷阱集合

Clustering Analysis on the Structure of LDPC Codes

Student : Ya-Chin Li

Advisor : Dr. John Kar-Kin Zao

Institute of Biomedical Engineering
Computer Science College
National Chiao Tung University

Abstract

Low-density parity-check (LDPC) codes are defined by a sparse parity-check matrix and can be described by a Tanner graph. But there is no structural property to confirm the performance.

We use clustering algorithms to analyze the structure of LDPC codes. We use the Markov Cluster Algorithm and the Modularity cluster algorithm to group the nodes of LDPC codes. We find that the small clusters have a higher probability of being the trapping set. Also, we find that some network parameters can explain the decoding process of LDPC codes.

Keywords: LDPC codes, network parameter, Markov Cluster algorithm, Modularity cluster algorithm, trapping set

致謝

感謝指導老師邵家健老師，從老師身上學到許多學術知識還有待人處事的道理，有老師的諄諄教誨我才能完成論文。感謝口試委員王忠炫老師與林敬堯博士，感謝你們撥空參加我的口試並給予詳盡的建議，使我的論文更加完整。

再來，感謝同組的力仁學長，很開心能有機會與學長討論；感謝 Martin 與柏崴學長，感謝你們這段時間的鼓勵。感謝實驗室的戰友們游傑、志凱、鍾豪、乙濤、恆源、Chatrpol，很開心能在同實驗室一起奮鬥。還要感謝學弟妹們郁善、智宇、正吉、文豪，感謝你們口試當天幫我舒緩緊張的情緒，非常感謝你們口試當天還幫我到系辦申請表格資料，我才能完成口試。

另外，感謝譚寧、詩涵、子緣、宛蓉、子寧這段時間的陪伴與鼓勵，有你們真好。還有許多同學們，感謝你們的關心與鼓勵。

最後，感謝我的家人，這段時間讓你們擔心了。



李亞錦@EC620 2013.09

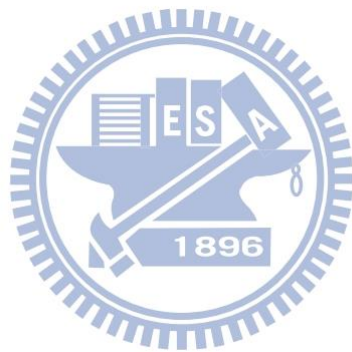
目錄

摘要	i
Abstract	ii
致謝	iii
目錄	iv
圖目錄	v
表目錄	vii
一. 緒論	1
1.1. 研究背景與動機	1
1.2. 研究方法	2
1.3. 論文架構	2
二. 背景知識	3
2.1. 簡介低密度奇偶檢查碼	3
2.1.1. 建構低密度奇偶檢查碼演算法的分類	5
2.2. 社群模組 (Community/Block models)	10
三. 宏觀 (Macroscopic) 分析低密度奇偶檢查碼結構	11
3.1. 網路參數 (Network Parameters)	12
3.2. 度譜 (Degree Spectrum)	18
四. 局部 (Mesoscopic) 分析低密度奇偶檢查碼結構	23
4.1. 角色模組 (Role model)	23
4.2. 馬可夫分群演算法 (Markov Cluster Algorithm)	25
4.3. 模組性 (Modularity)	40
五. 結論	49
5.1. 研究成果	49
5.2. 未來展望	49
參考文獻	50

圖目錄

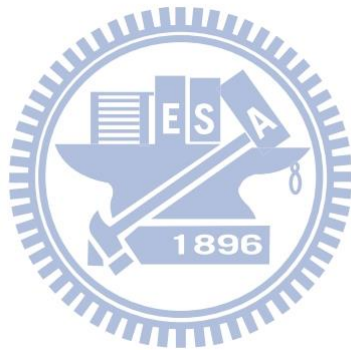
圖 1 檢驗矩陣與 Tanner graph 的關係	1
圖 2 通訊系統架構示意圖	3
圖 3 ACE 示意圖	4
圖 4 (a, b) trapping set 示意圖	4
圖 5 LDPC codes process	5
圖 6 Degree distribution of check nodes	8
圖 7 不同方法效能圖	9
圖 8 不同方法效能趨勢示意圖	9
圖 9 Structurally equivalent and Regularly equivalent	10
圖 10 Bipartite graph 與 Unipartite graph 的轉換	11
圖 11 Neighborhood connectivity illustration	12
圖 12 Neighborhood connectivity 比較圖	13
圖 13 Random 與 PEG-based 不同 degree 的 Average shortest path length	14
圖 14 不同方法所有節點的 average shortest path length	14
圖 15 Betweenness Centrality illustration	15
圖 16 Random 與 PEG-based 不同 degree 的 Betweenness centrality	16
圖 17 不同方法所有節點的 betweenness centrality	16
圖 18 Degree Spectrum illustration	18
圖 19 Random, PEG-based average degree spectrum distribution	22
圖 20 不同方法的 role model 分類	24
圖 21 Topology by MCL	28
圖 22 Average shortest path length of PEG-based cluster	30
圖 23 Betweenness centrality of PEG-based cluster	31
圖 24 Cluster histogram by MCL	31
圖 25 MCL cluster 範例圖	32
圖 26 MCL intra-cluster 範例圖	32
圖 27 MCL intra-cluster $a = 2$ 範例圖	32
圖 28 MCL inter-cluster 範例圖	33
圖 29 MCL in-out-cluster 範例圖	34
圖 30 MCL out-cluster 範例圖	34
圖 31 PEG MCL trapping $a = 2$ 比較結果	35
圖 32 IPEG MCL trapping $a = 2$ 比較結果	35
圖 33 MIPEG MCL trapping $a = 2$ 比較結果	35
圖 34 PEG MCL trapping $a = 3$ 比較結果	36
圖 35 IPEG MCL trapping $a = 3$ 比較結果	36
圖 36 MIPEG MCL trapping $a = 3$ 比較結果	37
圖 37 The betweenness centrality in clusters of PEG	38
圖 38 The betweenness centrality in clusters of IPEG	38
圖 39 The betweenness centrality in clusters of MIPEG	38
圖 40 Topology by Modularity	41

圖 41 Cluster histogram by Modularity	41
圖 42 Average shortest path length of each cluster	42
圖 43 Betweenness centrality of each cluster	43
圖 44 Modularity cluster 範例圖	44
圖 45 Modularity intra-cluster 範例圖	44
圖 46 Modularity inter-cluster 範例圖	44
圖 47 Modularity in-out-cluster 範例圖	45
圖 48 Modularity out-cluster 範例圖	45
圖 49 PEG Modularity trapping set $a = 2$ 比較結果	45
圖 50 IPEG Modularity trapping set $a = 2$ 比較結果	46
圖 51 MIPEG Modularity trapping set $a = 2$ 比較結果	46
圖 52 PEG Modularity trapping set $a = 3$ 比較結果	47
圖 53 IPEG Modularity trapping set $a = 3$ 比較結果	47
圖 54 MIPEG Modularity trapping set $a = 3$ 比較結果	47



表目錄

表	1 Progress-edge-growth algorithm.....	6
表	2 Modified-Improved PEG (MIPEG) algorithm.....	7
表	3 Degree distribution of variable nodes	8
表	4 QC-LDPC codes	8
表	5 Distance illustration	19
表	6 Pattern and distance of PEG-based with check node degree=8.....	21
表	7 Markov cluster algorithm.....	26
表	8 Markov cluster algorithm example	26
表	9 Markov cluster algorithm parameter setting.....	27
表	10 大 cluster 內 node 原始的 degree distribution.....	29
表	11 MCL intra-cluster $a = 2, b$ 分布表.....	33



一. 緒論

1.1. 研究背景與動機

低密度奇偶檢查碼 (Low-density parity check code, LDPC code) 是一種用於更正傳輸過程中發生錯誤的編碼方式，具有好的錯誤校正能力。低密度奇偶檢查碼可以透過奇偶檢驗矩陣 (parity-check matrix, H) 表示，奇偶檢驗矩陣由變量節點 (variable node) 與檢驗節點 (check node) 構成，奇偶檢驗矩陣能夠利用 Tanner graph 圖形化表示，讓我們更清楚了解低密度奇偶檢查碼的結構。奇偶檢驗矩陣與 Tanner graph 的關係如圖 1 所示，而 Tanner graph 中不可避免的存在著環 (cycle)，短環 (short cycle) 已被證實影響訊息傳播的獨立性降低解碼效能。

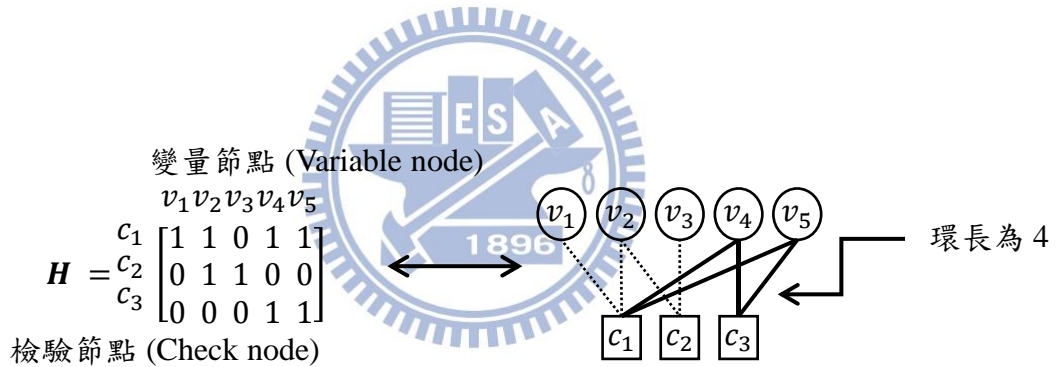


圖 1 檢驗矩陣與 Tanner graph 的關係

Progressive edge- growth (PEG) algorithm [1] 是一種簡單且可避免構成短環用來建構有限長度之低密度奇偶檢查碼的方法，許多其他演算法 [2] [3] 都是基於 PEG 演算法加以改善來提升解碼效能。然而針對 Tanner graph 中除了短環降低解碼效能之外，是否還存在其他結構上的因素影響解碼效能目前還是開放性問題。

1.2. 研究方法

建構低密度奇偶檢查碼的方法 [4] [5] 提到了複雜網路理論 (Complex network theory) 在建構低密度奇偶檢查碼時的關聯性，文中利用複雜網路理論所提到的 scale-free network [6] 網路結構特徵，來建構低密度奇偶檢查碼，讓低密度奇偶檢查碼的節點符合 scale-free network 的節點特性，來提升解碼效能。

針對低密度奇偶檢查碼的結構，我們使用分群演算法將節點分群，利用馬可夫分群演算法 (Markov Cluster Algorithm) [7] 與模組性分群演算法 (Modularity cluster algorithm) [8] 將結構性強的節點分在同一群中，找出影響 error floor region 的結構，並且利用網路參數 (network parameter) 觀察不同群的節點特性，解釋與解碼效能相關的群的特性與網路參數。

1.3. 論文架構

本文第二章將簡介低密度奇偶檢查碼與影響效能的因素，並且概述本文所觀察的低密度奇偶檢查碼的結構所使用的建構方法，同時介紹社群模組與低密度奇偶檢查碼的關聯；第三章將探討並觀察 Tanner graph 結構中節點的特性，本文稱為“宏觀”分析低密度奇偶檢查碼的結構；第四章將透過不同的分類方法找出子結構，本文稱為“局部”分析低密度奇偶檢查碼的結構，並且觀察子結構與效能的關係；第五章將總結影響低密度奇偶檢查碼結構上的因素，與未來可深入討論的方向。



二. 背景知識

2.1. 簡介低密度奇偶檢查碼

低密度奇偶檢查碼 (Low-density parity check code, LDPC code) [9] [10] 是一種用來更正傳輸過程中發生錯誤的編碼方式。如圖 2 資料傳送先經過編碼器 (encoder)，經過編碼器後傳出來的資料除了原始資料還多了一些多餘的資料 (redundant-bits)，使得裡面的資料具有數學結構，當經過通道後不論是原始資料或多餘的資料都有可能發生錯誤，但是因為資料存在著數學結構，所以解碼器 (decoder) 就能夠透過原有的數學結構將錯誤更正，這就是錯誤更正碼的基本概念 [11]。低密度奇偶檢查碼的解碼演算法主要使用和積演算法 (Sum-Product Algorithm)，透過 Tanner graph 能夠更清楚了解解碼過程。

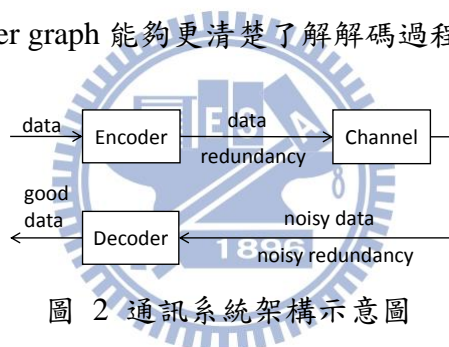


圖 2 通訊系統架構示意圖

以下為常見的影響低密度奇偶檢查碼效能的因素：

(1) Girth(shortest cycle length) 是表示所有 cycle 裡面 cycle length 最短的 length，cycle 的存在使得某一節點所送出的訊息經過 cycle 又回到自己，而破壞了訊息的獨立性，使得和積演算法無法計算正確的事後機率，影響解碼在 waterfall region 的效能。

(2) ACE 定義為 $\sum_j (d_{s_j} - 2)$ ， s_j 為 variable node， d_{s_j} 表示 s_j variable node 的 degree 數，ACE 值計算 cycle 上 variable node degree 減 2 的 degree 數總和，圖 3 為 ACE 值計算方式示意圖，圖中環上有 3 個 check node，3 個 variable node，而每一個 variable node degree 減掉 2

後得到的 ACE 值為 7，ACE 值越大表示有更高的機會能夠將迴圈上的錯誤拯救回來，提高錯誤訊息被更正的機率。

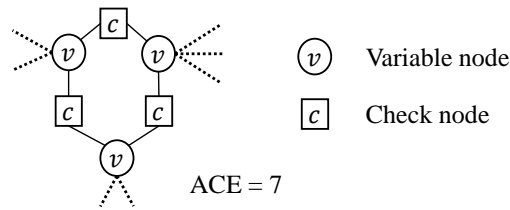


圖 3 ACE 示意圖

(3) (a, b) trapping set [12] 影響 error floor region， a 為發生錯誤的 variable node 數， b 為連接錯誤 variable nodes 的 check nodes 且 degree 為奇數的 check node 數。如圖 4，error pattern 中錯誤的 variable nodes 為 v_1, v_2, v_3 ，所以 a 值為 3，再來我們觀察 c_1, c_1 與 v_1, v_2, v_3 相連，所以 c_1 的 degree 為 3； c_2 與 v_1, v_2, v_4 相連，但是 v_4 並不是 error variable node，所以 c_2 degree 為 2； c_3 與 v_3 相連，所以 c_3 的 degree 為 1，由以上結果發現 check node degree 是奇數的只有 c_1 與 c_3 2 個 check nodes，所以 b 值為 2。

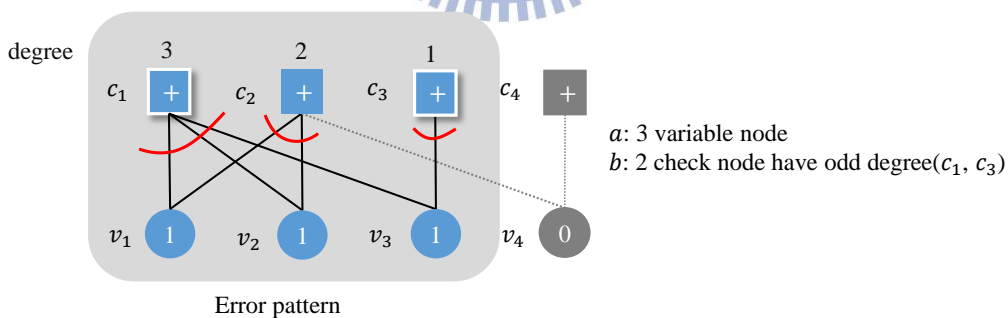


圖 4 (a, b) trapping set 示意圖

a 越小表示越少 variable node 發生錯誤，所以發生機率越高，而 b 值表示能被多少個 check nodes 偵測到錯誤，當 b 值越小表示被偵測的機會越小，所以 (a, b) 值越小所形成 trapping set 錯誤率越高，而本文將透過計算 (a, b) trapping set 的計算方式，評估挑選 variable node 形成錯誤率高的 trapping set 的機率。

2.1.1. 建構低密度奇偶檢查碼演算法的分類

建構低密度奇偶檢查碼時從 Tanner graph 的角度來看，是給定 1 組 variable node degree distribution，variable node 依序選擇 check node 相連，如圖 5，假設目前 tanner graph 已經長到左圖，右圖 variable node v_5 選擇 check node c_3 連接，不同的方法選擇不同的 check nodes 連接。

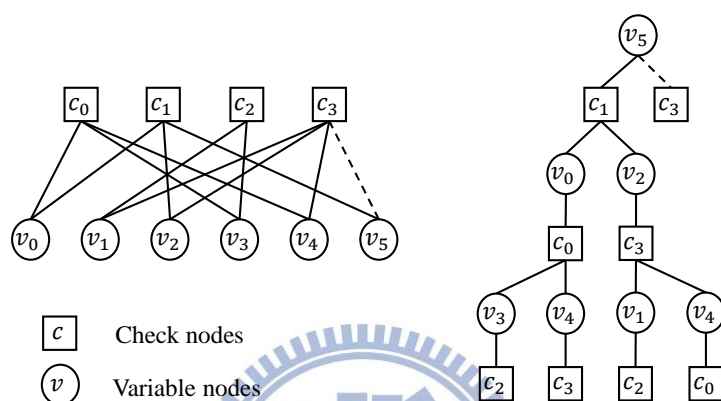


圖 5 LDPC codes process

我們觀察的低密度奇偶檢查碼 variable node 數約 1000，check node 數約 500，利用以下的方法建構，我們將所使用的方法分類，以下依序介紹：

□ Random

這一類的 LDPC codes 是最原始的選擇方式，沒有加入改善的機制。

· Random

在建構低密度奇偶檢查碼時“任意選擇” check node 連接，導致產生的 check node degree 分布是 normal distribution，並沒有特別集中在某些值。

- Zigzag

建構低密度奇偶檢查碼時，優先選擇沒有連接過的 check node 連接。

□ PEG-based

建構低密度奇偶檢查碼時不可避免會產生迴圈 (cycle)，PEG 演算法主要目的是拉長迴圈長度 (cycle length) 來提高解碼效能，這一類的 LDPC codes 是基於 PEG 加以改善建構方法。

- Progressive-edge-growth (PEG) algorithm [13]

表 1 為 PEG 演算法：

<ul style="list-style-type: none"> • for $j = 1 \rightarrow n$ <ul style="list-style-type: none"> — for $k = 1 \rightarrow d_{s_j}$ <ul style="list-style-type: none"> * if $k = 1$ <ul style="list-style-type: none"> • Connect the first edge of s_j with a check node, which has the lowest check node degree under the current graph setting. * else <ul style="list-style-type: none"> • Expand a tree from s_j up to depth l under the current graph setting such that $\bar{N}_{s_j}^l \neq \emptyset$ but $\bar{N}_{s_j}^{l+1} = \emptyset$, or the cardinality of $N_{s_j}^l$ stops increasing but is less than m. Connect the k-th edge of s_j with a check node from $\bar{N}_{s_j}^l$ that has the lowest degree. * end if — end for • end for
--

表 1 Progress-edge-growth algorithm

- Improved progressive-edge-growth (IPEG) algorithm

基於 PEG 演算法，[2] 提出了 ACE (approximated cycle extrinsic message degree) 的概念，建構低密度奇偶檢查碼時不只拉長 cycle length 還加入 ACE 值，ACE 定義為 $\sum_j (d_{s_j} - 2)$ ，計算迴圈 variable node 的 degree 減掉 2，ACE 值越大表示迴圈上的 variable node 有更多機會透過其他不在迴圈上的邊 (edge) 接收訊息，提高救回迴圈上 variable node 的錯誤訊息的機率。

- Modified Improved PEG (MIPEG) algorithm

[13] 結合了 PEG 與 IPEG，在效能上優於 PEG 與 MIPEG，表 2 為

MIPEG [13]演算法：

- for $u = 1 \rightarrow w$
 - for each $s_j \in S_u$ (First Edge Connection)
 - * Find the subset of C of check nodes, which has the lowest check node degree under the current graph setting. Among the subset C , the check node candidate of the first edge which can create the largest cycle length on its second edge is chosen.
 - end for
 - while at least one variable nodes not reaches its degree
 - * For each $s_j \in S_u$ whose degree is smaller than d_{s_j} , expand a tree from s_j up to depth l under the current graph setting such that $\overline{N}_{s_j}^l \neq \emptyset$ but $\overline{N}_{s_j}^{l+1} = \emptyset$, or cardinality of $N_{s_j}^l$ stops increasing but is less than m .
 - * Let V a set of comprising the variable nodes that have the largest depth l_{max} among all tree expansions.
 - * Let R be a subset of V , which can form cycles with the largest cycle ACE.
 - * Let Ω_j be a subset of check nodes that consists of the check nodes in $\overline{N}_{s_j}^{l_{max}}$ and has the lowest degree.
 - * Randomly choose a variable node $s_i \in R$. The check node in Ω_i that can make the new forming cycles have the largest cycle ACE is selected.
 - end
- end for

表 2 Modified-Improved PEG (MIPEG) algorithm

上述五種演算法在建造碼的時候皆已避免產生環長為 4 (short cycle length=4) 的情況，都是使用相同長度與相同 variable node degree distribution 如表 3 所建構，variable node 數為 1008，check node 數為 504，圖 6 為建構後 check node degree distribution，可以發現 PEG-based 的 check node degree 只會集中在 7~9，而 Random 的分布像 normal distribution。

Degree	Nodes
2	480
3	282
4	36
5	110
15	100

表 3 Degree distribution of variable nodes

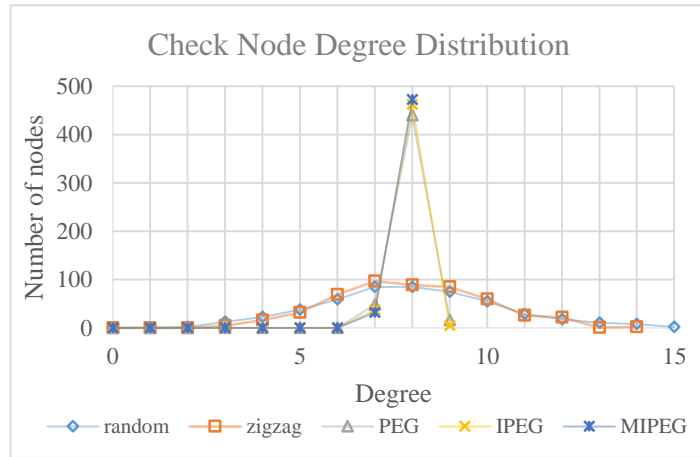


圖 6 Degree distribution of check nodes

□ QC-LDPC codes

QC-LDPC codes 是 regular LDPC codes，variable node degree 與 check node degree 都只有一種，表 4 為我們所建構的 QC-LDPC codes 的資訊。

	QC-LDPC codes
Number of variable node	1038
Number of check node	519
The degree of variable node	3
The degree of check node	6

表 4 QC-LDPC codes

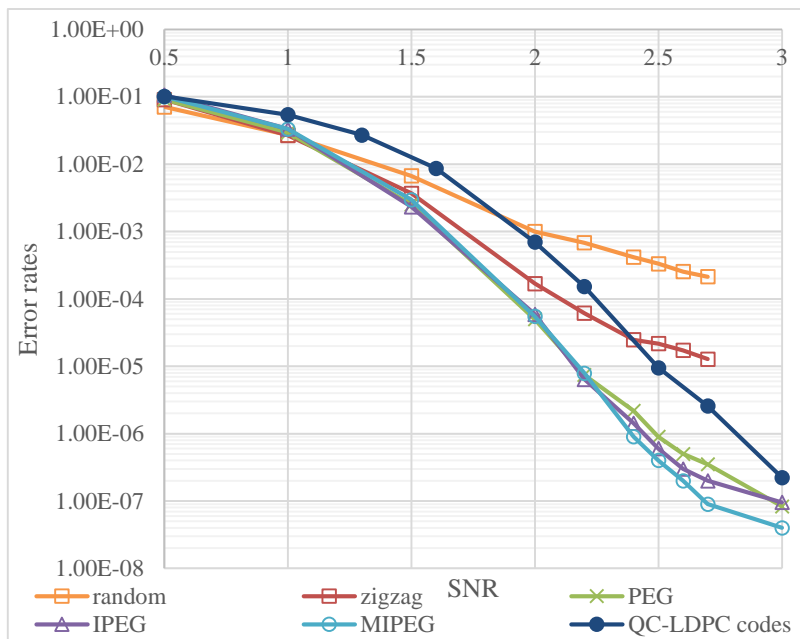


圖 7 不同方法效能圖

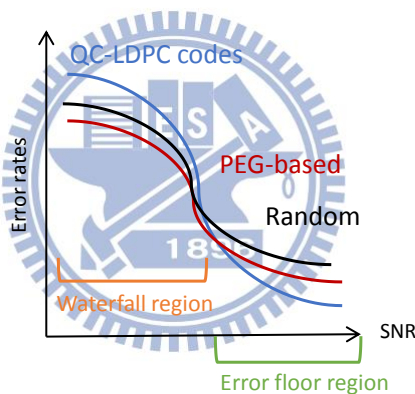


圖 8 不同方法效能趨勢示意圖

圖 7 為上述建構方法效能模擬的表現，圖 8 為效能趨勢示意圖，由圖 8 可以發現在 waterfall region 的部分，Error rates 由大到小為 QC-LDPC codes > Random > PEG-based，Error rates 越大表示錯誤率越高效能越差，PEG-based 效能最好，再來是 random，表現較差的是 QC-LDPC codes；而在 error floor region 的部分 Error rates 由大到小為 Random > PEG-based > QC-LDPC codes，QC-LDPC codes 表現最好，再來是 PEG-based，而 Random 在 error floor 表現最差。

2.2. 社群模組 (Community/Block models)

透過分群的方法會將存在 equivalent 關係的節點分在同一個 Community/Block 中，而 equivalent 又分為 Structurally equivalent 與 Regularly equivalent；如果兩個節點的鄰近節點都相同，那就表示這兩個節點是 Structurally equivalent，如圖 9 中 node A 與 node B 都連接 node C 與 node D，所以 node A 與 node B 是 structurally equivalent 的關係；而 node C 與 node D 被相同 equivalent 的 node A 與 node B 相連，且 node C 連接到 node D，而 node D 連接到 node C，節點連接到相同 equivalent 的節點稱為 regularly equivalent。

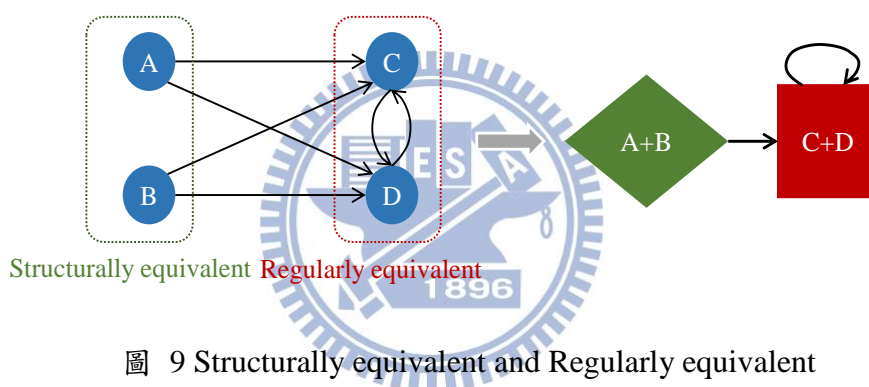


圖 9 Structurally equivalent and Regularly equivalent

本文使用“局部”(Mesoscopic) 觀察來表示利用不同的分類方法分析 LDPC codes，分類方法為：角色模組 (role model)、馬可夫分群演算法 (Markov Cluster algorithm)、模組性 (Modularity)，其中 role model 為 regularly equivalent 的分類方法，Markov Cluster algorithm 與 Modularity cluster algorithm 為 structurally equivalent 的分類方法，本文第四章將詳細說明其演算法與分析結果。

三. 宏觀 (Macroscopic) 分析低密度奇偶 檢查碼結構

我們使用 2.1.1 所提到的各種演算法 (Random, zigzag, PEG, IPEG, MIPEG, QC-LDPC codes) 建構出來的 Tanner graph 進行以下的分析, 找出不同類型的演算法影響效能的結構差異。

Tanner graph 是二分圖 (bipartite graph), variable node 只會與 check node 連接, 透過簡單的轉換能夠將 bipartite graph 轉換成 unipartite graph, 如圖 10 將 bipartite graph 轉換成 variable node 的 unipartite graph, unipartite graph 上的每一條 edge 表示 check node, 例如 variable node v_2 透過 check node c_1 可走到 variable node v_3 。

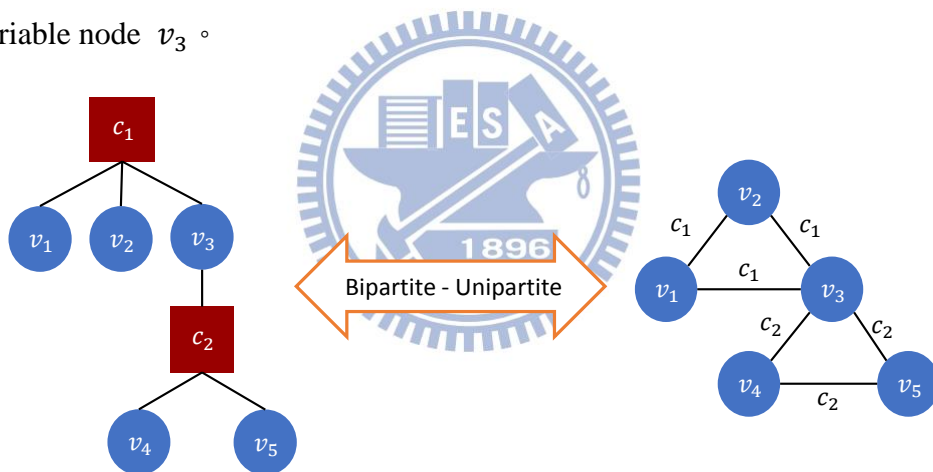


圖 10 Bipartite graph 與 Unipartite graph 的轉換

LDPC codes 解碼過程中 variable node 的訊息傳遞影響解碼效能, 所以我們觀察 variable node 的 unipartite graph, 將 Tanner graph 轉換成只有 variable node 與 variable node 連接的 unipartite graph, 找出 variable node 到 variable node 的 unipartite graph 中, 是不是存在著影響 variable node 訊息傳遞的因素。

3.1. 網路參數 (Network Parameters)

網路理論中存在許多參數來表現網路結構特徵，我們先將網路參數進行分類與說明參數的表現意義，再來將網路參數與低密度奇偶檢查碼的關聯性加以描述，最後針對低密度奇偶檢查碼不同方法的參數分析結果進行討論。

□ 與鄰近節點 (neighborhood) 相關的參數

1. Neighborhood connectivity

Connectivity 代表節點的度 (degree)，Neighborhood connectivity 計算鄰近節點 degree 的平均值，圖 11 計算 node *e* 的 Neighborhood connectivity。

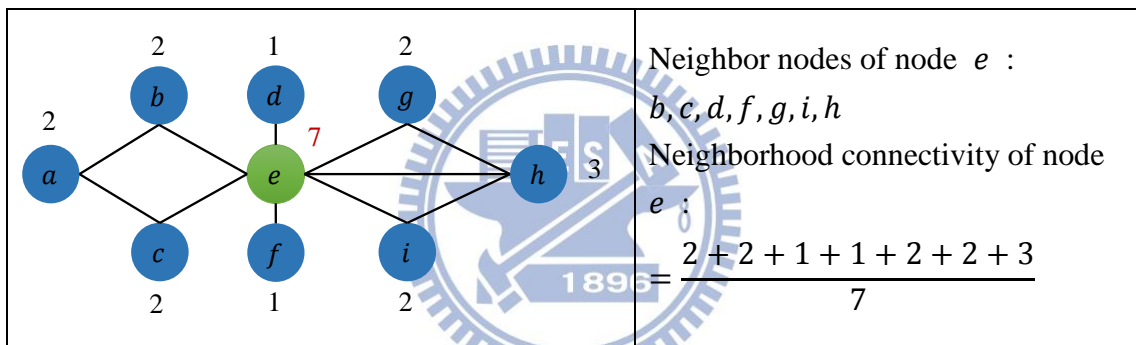


圖 11 Neighborhood connectivity illustration

我們是計算 variable node 的 neighborhood connectivity，而 variable node 只與 check node 連接，所以觀察 variable nodes 的 neighborhood connectivity 與 check node degree distribution 有關，圖 12 為不同方法的 neighborhood connectivity，由圖可以發現效能差的碼，例如 random, zigzag，Neighborhood connectivity 值較不一致，效能好的碼，例如 PEG, IPEG, MIPEG，Neighborhood connectivity 值較一致，QC-LDPC codes 的 variable node 與 check node 的 degree 都只有一個值，並不適合使用 neighborhood connectivity 觀察。

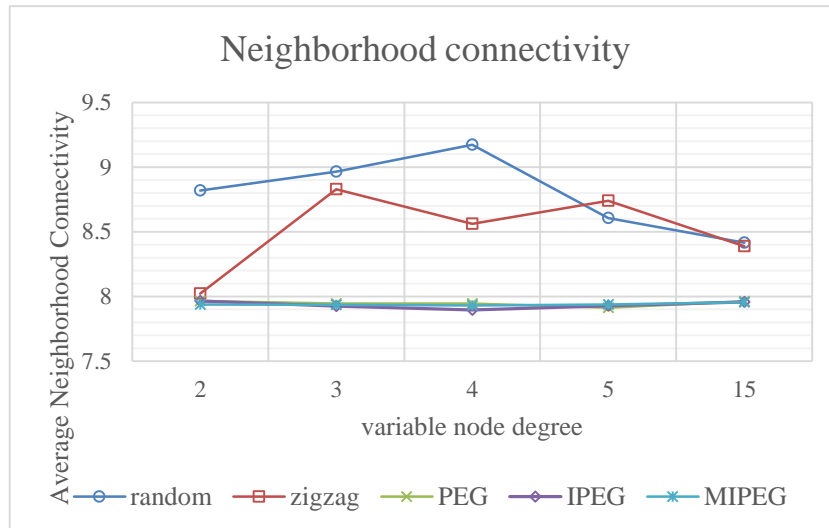


圖 12 Neighborhood connectivity 比較圖

□ 與最短路徑 (shortest path) 相關的參數

LDPC codes 解碼效能與 variable node 的訊息傳遞距離有關，所以接下來有關最短路徑參數本文使用 variable node 的 unipartite graph 分析，並統計不同 degree 下的分布情形，找出不同 degree 中的差異。

1. Average shortest path length 是計算節點到其他節點的最短路徑。

$$\text{The average shortest path length of node } n: \sum_{i=1, i \neq n}^N \frac{L(n, i)}{N-1} \quad (1)$$

$L(n, m)$: The length of shortest path between node n and m .

N : all nodes in graph.

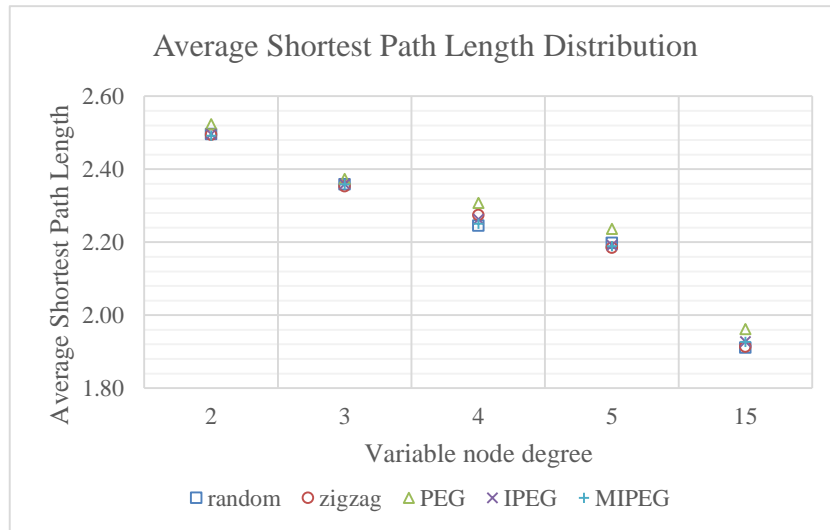


圖 13 Random 與 PEG-based 不同 degree 的 Average shortest path length

由圖 13 可以發現，high degree variable node 的 shortest path length 較短，high degree 在傳遞訊息比較容易將訊息傳遞給其他 node，所以 high degree 的 variable node 容易將自己的訊息傳送給其他 nodes。接下來觀察所有 node 的 average shortest path length，由圖 14 可以發現，不同方法所有 node 的 average shortest path length 沒有差異，透過觀察所有節點的 shortest path 無法區分不同建構方法。

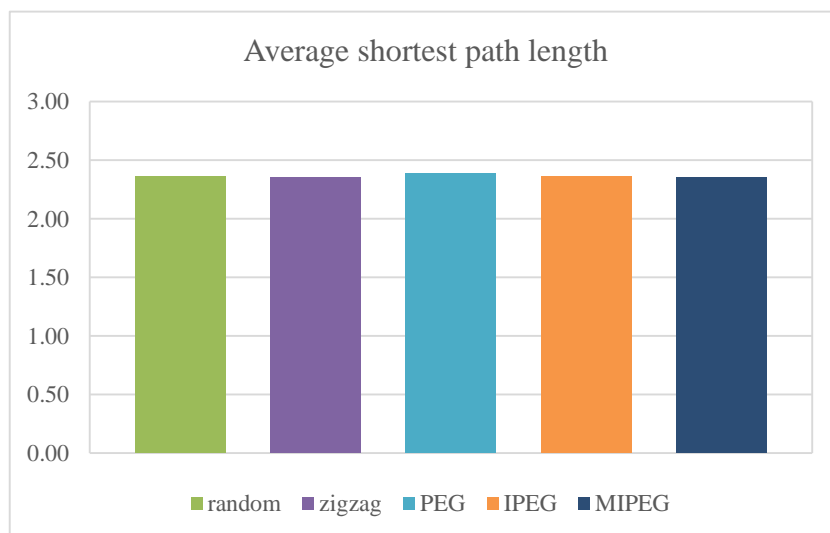


圖 14 不同方法所有節點的 average shortest path length

2. Betweenness centrality 是計算節點在最短路徑上的機率。

$$C_b(n) = \frac{\sum_{s \neq n \neq t} \left(\frac{\sigma_{st}(n)}{\sigma_{st}} \right)}{\binom{N-1}{2}} \quad (2)$$

s : node s , t : node t

σ_{st} : node s 到 node t 的總 shortest path 數

$\sigma_{st}(n)$: node s 到 node t 的 shortest path 且經過 n 的 path 數

N : the number of nodes

圖 15 計算 node b 的 Betweenness centrality :

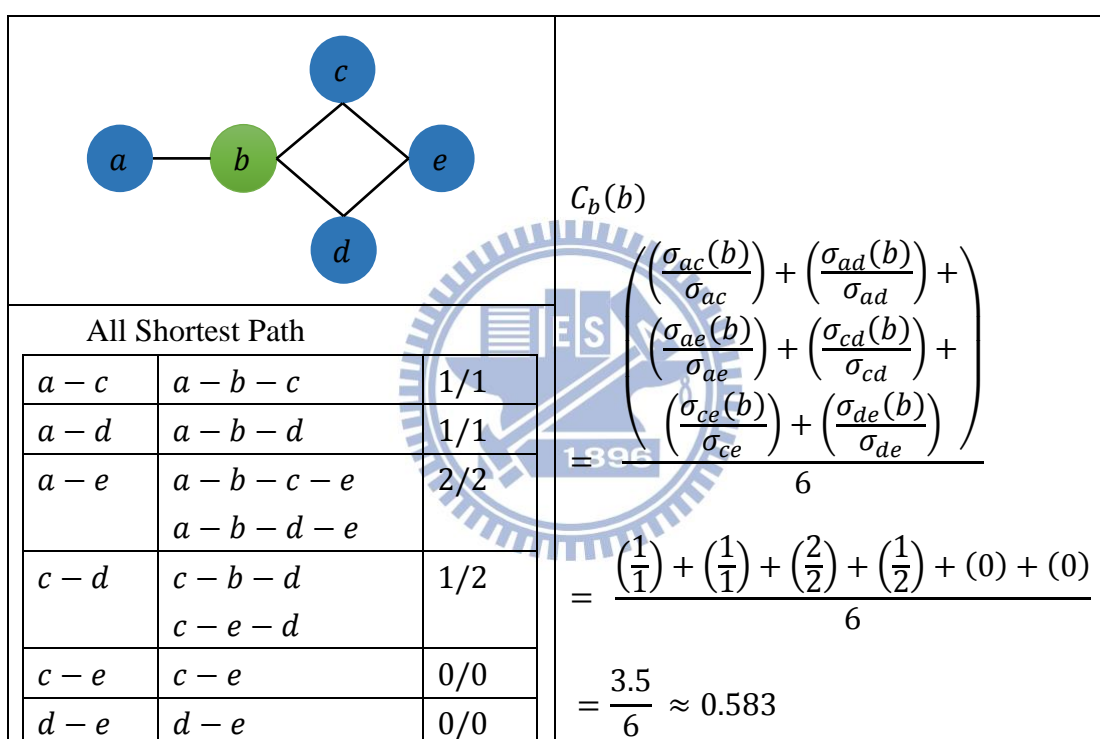


圖 15 Betweenness Centrality illustration

節點的 Betweenness centrality 值越大表示節點位在最短路徑上的機率越高，也就表示圖中傳送訊息時容易經過 betweenness centrality 高的節點，那也就表示這個節點自己的訊息容易被更改，不易形成 trapping set。

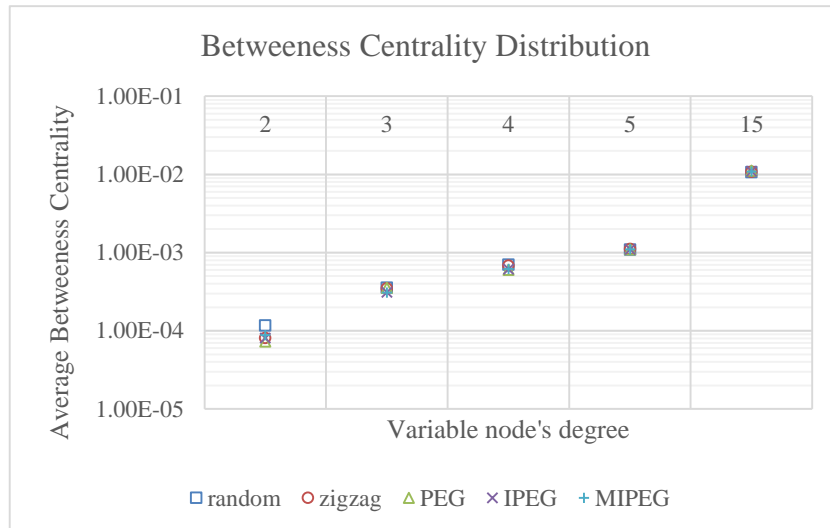


圖 16 Random 與 PEG-based 不同 degree 的 Betweenness centrality

由圖 16 可以發現 high degree variable node 的 betweenness centrality 值較大，而我們知道 high degree variable node 不容易形成 trapping set，low degree variable node 容易形成 trapping set，透過 betweenness centrality 的觀察發現相對應的趨勢；接下來計算所有節點的 betweenness centrality，可以由圖 17 發現，不同方法差異不大，無法透過所有節點的 betweenness centrality 區分不同建構方法。

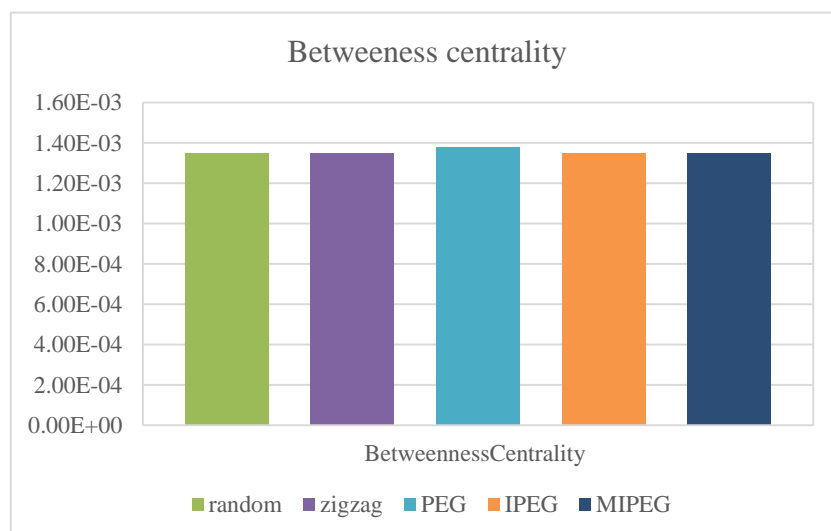
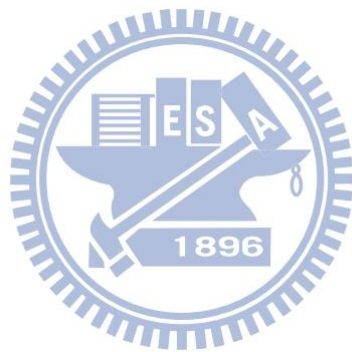


圖 17 不同方法所有節點的 betweenness centrality

本節所使用的網路參數與 neighborhood 有關的 neighborhood connectivity 可以用來觀察 node degree distribution，不同方法建構的 check node degree distribution 不同，透過 neighborhood connectivity 可以觀察 degree distribution 的差異。與 shortest path length 有關的參數可以用來觀察節點傳遞訊息的距離，低密度奇偶檢查碼的解碼過程與 variable node 傳遞訊息的路徑有關，可以透過計算 variable 到 variable node 的 shortest path length 來觀察傳遞的情形；而 low degree node 的 betweenness centrality 較低，較容易成為 trapping set。



3.2. 度譜 (Degree Spectrum)

[14] 提出度譜 (Degree Spectrum) 的概念，定義一個節點的鄰近節點 (neighbor node) 的 degree 稱為 degree spectrum，如圖 18 所示，check node c_1 的 degree = 7，則 c_1 的 degree spectrum 為 2, 2, 3, 3, 4, 5, 15。文中觀察 check node 的 degree spectrum，文中提到透過 PEG 演算法所建構出來的低密度奇偶檢查碼 check node 的 degree spectrum 會集中在特定的 degree spectrum，且 degree spectrum 間相似，如果增加 degree spectrum 相異的程度就能夠提升解碼效能。

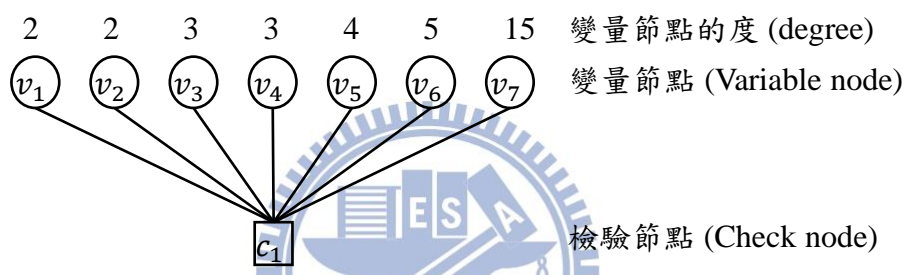


圖 18 Degree Spectrum illustration

本文定義了樣式 (pattern) 與距離 (distance) 兩種參數，pattern 用來觀察 degree spectrum 的相異性，distance 用來觀察 degree spectrum 的相似程度。QC-LDPC codes 的 variable node 與 check node 的 degree 只有一種，不適合用 degree spectrum 來觀察。

表 3 為我們所分析的低密度奇偶檢查碼演算法的 variable node degree distribution，pattern 的定義為 (Check node's degree; $N(15)$ $N(5)$ $N(4)$ $N(3)$ $N(2)$)， $N(n)$ 表示 degree= n 的 variable node 個數。圖 18 check node c_1 的 degree = 7，degree spectrum 為 2, 2, 3, 3, 4, 5, 15，則我們用 pattern (7; 1 1 1 2 2) 來表示。

我們統計所有 check node 的 pattern 出現的頻率，以最高頻率出現的 pattern 為基準，計算其他 pattern 與最高頻率出現 pattern 的差值，定義為 distance，計算方式如表 5。

Pattern	頻率	Distance
(7; 2 1 1 1 2)	2	$(7; 2 \ 1 \ 1 \ 2 \ 1) - (7; 2 \ 1 \ 1 \ 1 \ 2)$ $= 7 - 7 + 2 - 2 + 1 - 1 + 1 - 1 + 2 - 1 + 1 - 2 $ $= 2$
(7; 2 1 1 2 1)	10	最高頻率 pattern
(7; 2 1 0 2 2)	7	$(7; 2 \ 1 \ 1 \ 2 \ 1) - (7; 2 \ 1 \ 0 \ 2 \ 2)$ $= 7 - 7 + 2 - 2 + 1 - 1 + 1 - 0 + 2 - 2 + 1 - 2 $ $= 2$
(7; 1 0 2 2 2)	1	$(7; 2 \ 1 \ 1 \ 2 \ 1) - (7; 1 \ 0 \ 2 \ 2 \ 2)$ $= 7 - 7 + 2 - 1 + 1 - 0 + 1 - 2 + 2 - 2 + 1 - 2 $ $= 4$

表 5 Distance illustration

利用不同演算法所建構出來的 check node degree 不同，圖 6 為不同方法產生的 check node degree distribution，由圖中可以觀察到 Random 所產生出來的 check node degree distribution 較分散，類似 normal distribution，而 PEG-based 所產生出來的 check node degree distribution 較集中，這是因為 PEG 演算法每次選擇連接的 check node 時，優先選擇不造成短環的情況下 degree 最小的 check node，所以選擇的過程幾乎都是 degree 最小的 check node 優先被選到，這就使不同的 check node 最後的 degree 集中在特定的值；而 IPEG 與 MIPEG 都是基於 PEG 改善，產生後的 check node degree 也都集中在特定的值。

由圖 6 可以發現 PEG-based 在 check node degree=8 的數量最多，我們選擇 check node 來觀察每一個 check node 的 degree spectrum，利用 pattern 與 distance 來觀察。

PEG Number Of Check Node: 440

n(15)	n(5)	n(4)	n(3)	n(2)	Frequency	Probability	Distance
4	1	0	2	1	10	2.3%	2
4	1	0	1	2	2	0.5%	2
4	0	1	2	1	1	0.2%	4
4	0	1	1	2	3	0.7%	4
3	2	0	2	1	7	1.6%	2
3	2	0	1	2	9	2.0%	2
3	1	1	1	2	131	29.8%	2
3	1	0	2	2	260	59.1%	0
3	0	1	2	2	1	0.2%	2
2	2	0	2	2	16	3.6%	2

IPEG Number Of Check Node: 462

n(15)	n(5)	n(4)	n(3)	n(2)	Frequency	Probability	Distance
4	0	1	1	2	5	1.1%	4
4	0	0	2	2	11	2.4%	2
3	2	0	2	1	8	1.7%	2
3	2	0	1	2	41	8.9%	2
3	1	1	2	1	8	1.7%	2
3	1	1	1	2	116	25.1%	2
3	1	0	3	1	1	0.2%	2
3	1	0	2	2	265	57.4%	0
3	0	0	3	2	1	0.2%	2
2	2	0	2	2	6	1.3%	2

MIPEG Number Of Check Node: 472

n(15)	n(5)	n(4)	n(3)	n(2)	Frequency	Probability	Distance
4	1	0	2	1	2	0.4%	2
4	1	0	1	2	2	0.4%	2
4	0	1	2	1	3	0.6%	4
4	0	1	1	2	14	3.0%	4
4	0	0	3	1	3	0.6%	4
4	0	0	2	2	12	2.5%	2
3	2	0	2	1	11	2.3%	2
3	2	0	1	2	52	11.0%	2
3	1	1	2	1	14	3.0%	2

3	1	1	1	2	97	20.6%	2
3	1	0	3	1	9	1.9%	2
3	1	0	2	2	227	48.1%	0
3	0	2	1	2	1	0.2%	4
3	0	0	3	2	7	1.5%	2
2	3	0	2	1	1	0.2%	4
2	3	0	1	2	2	0.4%	4
2	2	1	1	2	4	0.8%	4
2	2	0	2	2	11	2.3%	2

表 6 Pattern and distance of PEG-based with check node degree=8

表 6 列出 PEG, IPEG, MIPEG 的 check node degree 為 8 的 pattern 與 distance, PEG 的 pattern 數是 10 組, distance 差異為 2 的有 7 組, 差異為 4 的有 2 組, 表示 pattern 間與最高頻率發生的 pattern 相似, PEG 的 degree spectrum 相似; IPEG 的 pattern 數是 10 組, distance 差異為 2 的有 8 組, 差異為 4 的有 1 組, 表示 pattern 間與最高頻率發生的 pattern 相似, IPEG 的 degree spectrum 相似; 而 MIPEG 的 pattern 數是 18 組, distance 差異為 2 的有 10 組, 差異為 4 的有 7 組, 相較於 PEG 與 IPEG, MIPEG 的 pattern 間與最高頻率發生的 pattern 不相似, MIPEG 的 degree spectrum 不相似, 驗證了 [14] 所提出的觀察。

透過 degree spectrum 的觀察, Random 沒有集中的 pattern, pattern 種類多; PEG-based pattern 種類不多, 與建構後的 check node degree 有關, degree spectrum 集中在特定的 pattern, 且 distance 差異大, pattern 間相似程度小。

當沒有集中在特定 pattern 時, 我們使用以下將介紹的平均度譜 (Average Degree spectrum) 來觀察。

平均度譜 (Average Degree Spectrum) 為 degree spectrum 的平均值, 如圖 18, check node c_1 的 average degree spectrum 為 $\frac{2+2+3+3+4+5+15}{7} = \frac{34}{7}$ 。我們利

用 average degree spectrum 來觀察不同演算法所建構的低密度奇偶檢查碼的 average degree spectrum 分布，觀察不同建構方法的分布情形。

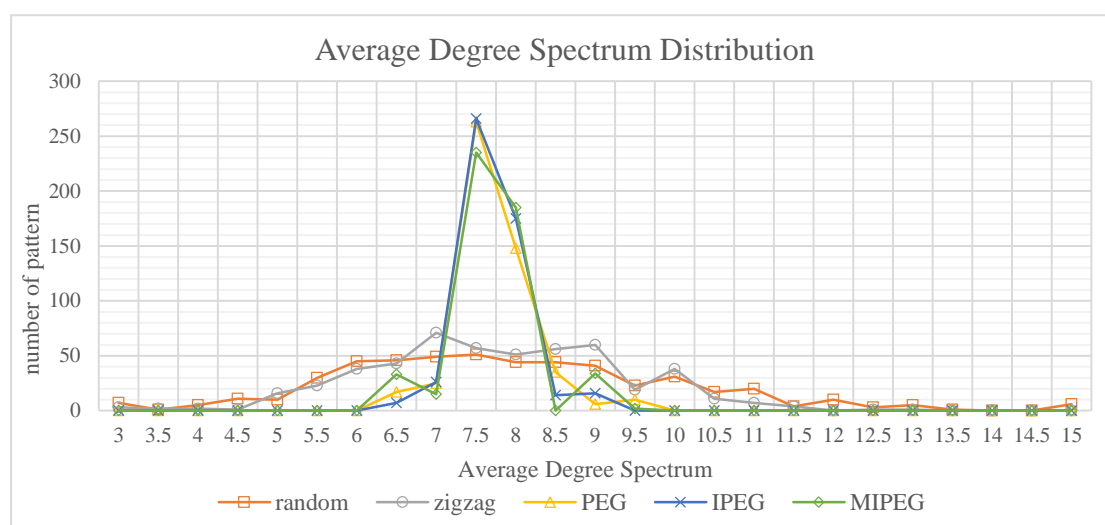


圖 19 Random, PEG-based average degree spectrum distribution

Random 分布是常態分布 (normal distribution)，PEG-based 的 average degree spectrum 值集中在 7 到 8.5 之間，表示 PEG-based 產生的 average degree spectrum 相似度高，皆集中在某些值。

由以上的結果發現 degree spectrum 與建構後的 check node degree distribution 有關，效能好的碼 check node degree 集中在某些值，degree spectrum 也集中，degree spectrum 間與高頻率發生的 degree spectrum 越不相似效能更好；效能差的碼 check node degree 分散，沒有高頻率發生的 degree spectrum，我們也驗證了 [14] 所提出的結論，透過本文提供 pattern 與 distance 的表示方式，能更方便的觀察 degree spectrum。

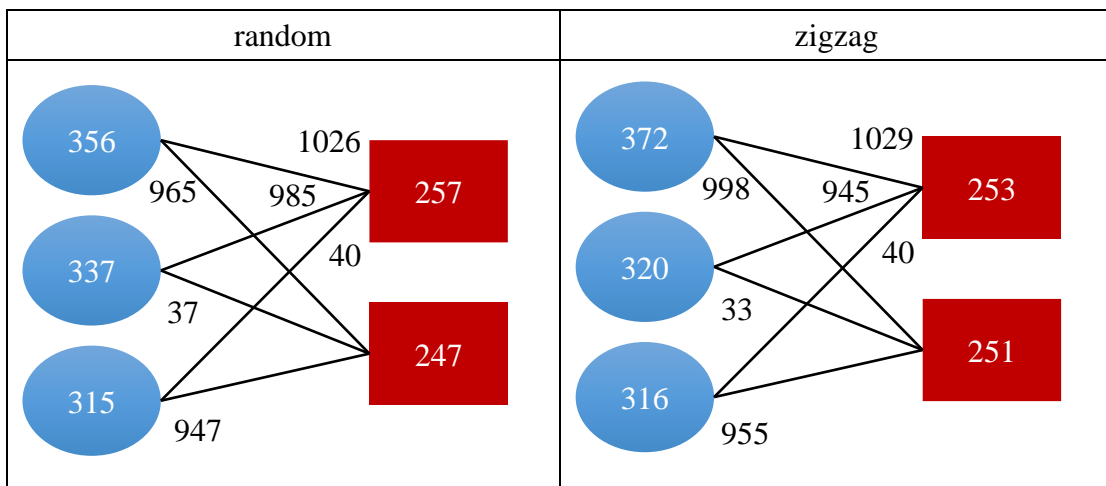
四. 局部 (Mesoscopic) 分析低密度奇偶檢查碼結構

此章節我們將利用不同的分類方式將低密度奇偶檢查碼的節點分類，同一類的節點具有一些相似的特性，透過分類的結果我們觀察每一類中的節點特性，找出不同類中節點的差異，並觀察是否存在影響解碼效能的因素。

以下介紹我們所使用的分類方法，以及將低密度奇偶檢查碼的節點分類後的結果與討論。

4.1. 角色模組 (Role model)

角色模組 (Role model) 是一種 Regularly equivalent 的分類方法，我們使用 Jörg Reichardt [15] [16] 提供的 Role model for complex network [17] 分析方法將低密度奇偶檢查碼分類，設定分成 5 種 role，圖 20 中每一個單位是 1 個 role，左側藍色圓圈 role 中都是 variable nodes，右側紅色方框 role 中都是 check nodes。



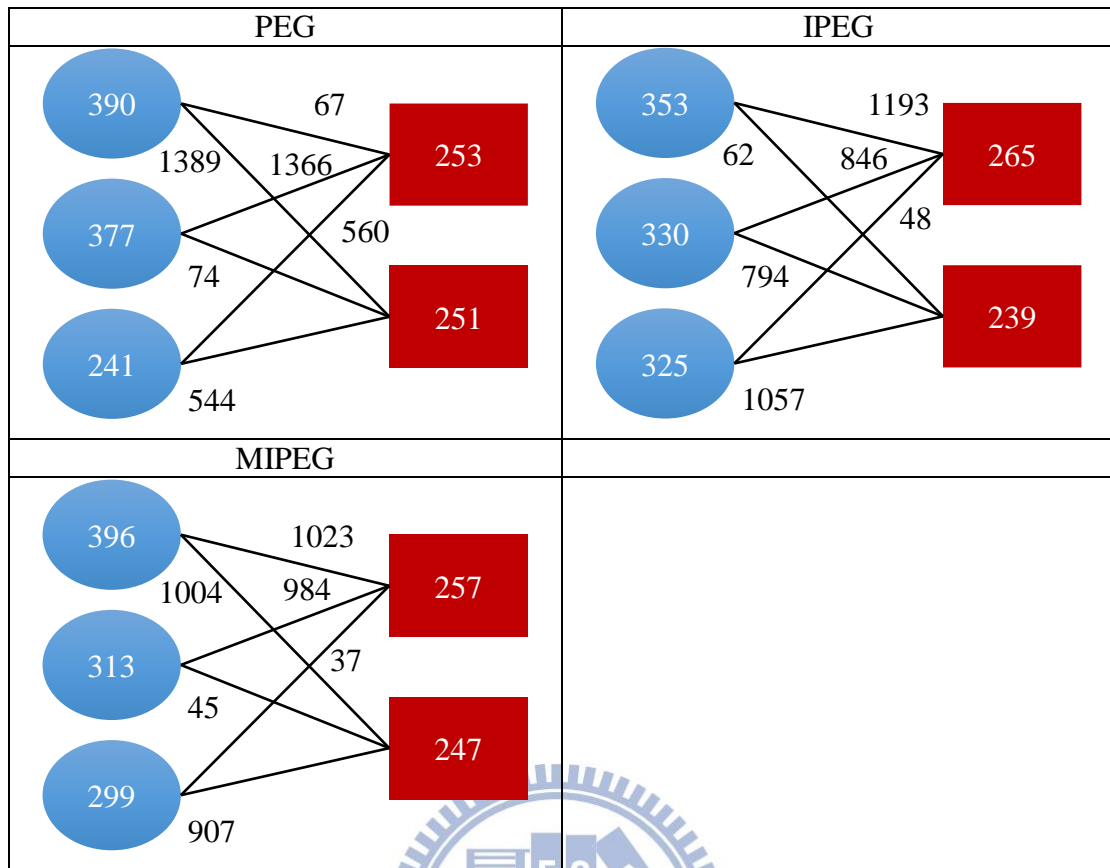


圖 20 不同方法的 role model 分類

由圖 20，不同方法分類後藍色 role 中 variable nodes 數量分布平均，紅色 role 中 check nodes 數量分布平均，role 間的連線每一個方法差不多，無法觀察到不同建構方法的差異。

透過實驗的結果發現 role model 這種 regularly equivalent 分類方法將低密度機偶檢查碼區分為 variable node 與 check node 兩類，而 variable node 本來就只會跟 check node 連線，透過 regularly equivalent 並無法區分不同建構方法的連線方式，所以 regularly equivalent 這種分類方式並不適合將低密度奇偶檢查碼的節點進行分類。

4.2. 馬可夫分群演算法 (Markov Cluster Algorithm)

Markov Cluster Algorithm (MCL) 是屬於 structurally equivalent 的分類方法，由 Stijn van Dongen [7] 提出，是一種能夠將大量節點分群 (cluster) 的方法，MCL 已應用在分析複雜生物網路，例如蛋白質交互作用的網路拓撲 (Topological Similarity of Protein Interaction Network)，透過 MCL 進行 protein family detection；還有人類的疾病基因研究 (Disease-Gene network)，人類的疾病基因網路 (Disease-Gene network) 是一種二分圖 (bipartite graph)，利用 MCL 可以預測疾病基因網路，分群後的結果表示同 cluster 內的基因容易透過生物反映程序作用影響同 cluster 內的疾病，相較於任意選擇的基因，同 cluster 內的基因有比較大的機率影響同 cluster 內的疾病。

MCL 是一種基於隨機漫步 (random walk) 的方法來不斷模擬節點在圖中行走的路徑，經過一段時間後，節點在某個區域內行走的機率較高，這些區域稱為群 (cluster)。MCL 主要的兩種運算是 expansion 和 inflation，expansion 的目的主要是讓節點能夠在不同節點間行走，inflation 的目的主要是放大每一條線 (edge) 行走的機率，透過初始的 stochastic matrix 不斷的執行 expansion 和 inflation 將節點分群 (cluster)。表 7 為 MCL 的演算法。

- | |
|---|
| <ol style="list-style-type: none">Step 1. Input a Graph, expansion parameter e, inflation parameter r;Step 2. Create the adjacency matrix from the graph;Step 3. Add self-loops;Step 4. Normalize the matrix M;Step 5. Expand the matrix with e^{th} power, i.e. $(M)^e$;Step 6. Inflate by taking inflation of the resulting matrix with parameter r;Step 7. Repeat step 5 and step 6 until a steady state is achieved;Step 8. Interpret resulting matrix to discover clusters. |
|---|

Definition: [18]

Given a matrix $M \in \mathcal{R}^{k \times l}$, $M \geq 0$, and a real nonnegative number r , the matrix resulting from rescaling each of the columns of M with power coefficient r is called $\Gamma_r M$, and Γ_r is called the **inflation** operator with power coefficient r . Formally, the action of $\Gamma_r: \mathcal{R}^{k \times l} \rightarrow \mathcal{R}^{k \times l}$ is defined by

$$(\Gamma_r M)_{pq} = (M_{pq})^r / \sum_{i=1}^k (M_{iq})^r$$

表 7 Markov cluster algorithm

		<p>expansion parameter = 2, inflation parameter = 2</p>
Step 1		
$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$ <p style="text-align: center;">Step 2</p>	$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$ <p style="text-align: center;">Step 3</p>	$M = \begin{pmatrix} 1/3 & 1/3 & 1/4 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/4 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/4 & 1/4 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/4 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/4 & 1/3 & 1/3 \end{pmatrix}$ <p style="text-align: center;">Step 4</p>
$M^2 = \begin{pmatrix} 0.31 & 0.31 & 0.23 & 0.06 & 0 & 0 \\ 0.31 & 0.31 & 0.23 & 0.06 & 0 & 0 \\ 0.31 & 0.31 & 0.29 & 0.13 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.13 & 0.29 & 0.31 & 0.31 \\ 0 & 0 & 0.06 & 0.23 & 0.31 & 0.31 \\ 0 & 0 & 0.06 & 0.23 & 0.31 & 0.31 \end{pmatrix}$ <p style="text-align: center;">Step 5</p>	$\Gamma_2 M^2 = \begin{pmatrix} 0.33 & 0.33 & 0.25 & 0.02 & 0 & 0 \\ 0.33 & 0.33 & 0.25 & 0.02 & 0 & 0 \\ 0.33 & 0.33 & 0.40 & 0.07 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.07 & 0.30 & 0.33 & 0.33 \\ 0 & 0 & 0.02 & 0.25 & 0.33 & 0.33 \\ 0 & 0 & 0.02 & 0.25 & 0.33 & 0.33 \end{pmatrix}$ <p style="text-align: center;">Step 6</p>	
<p style="text-align: center;">Step 8</p>		

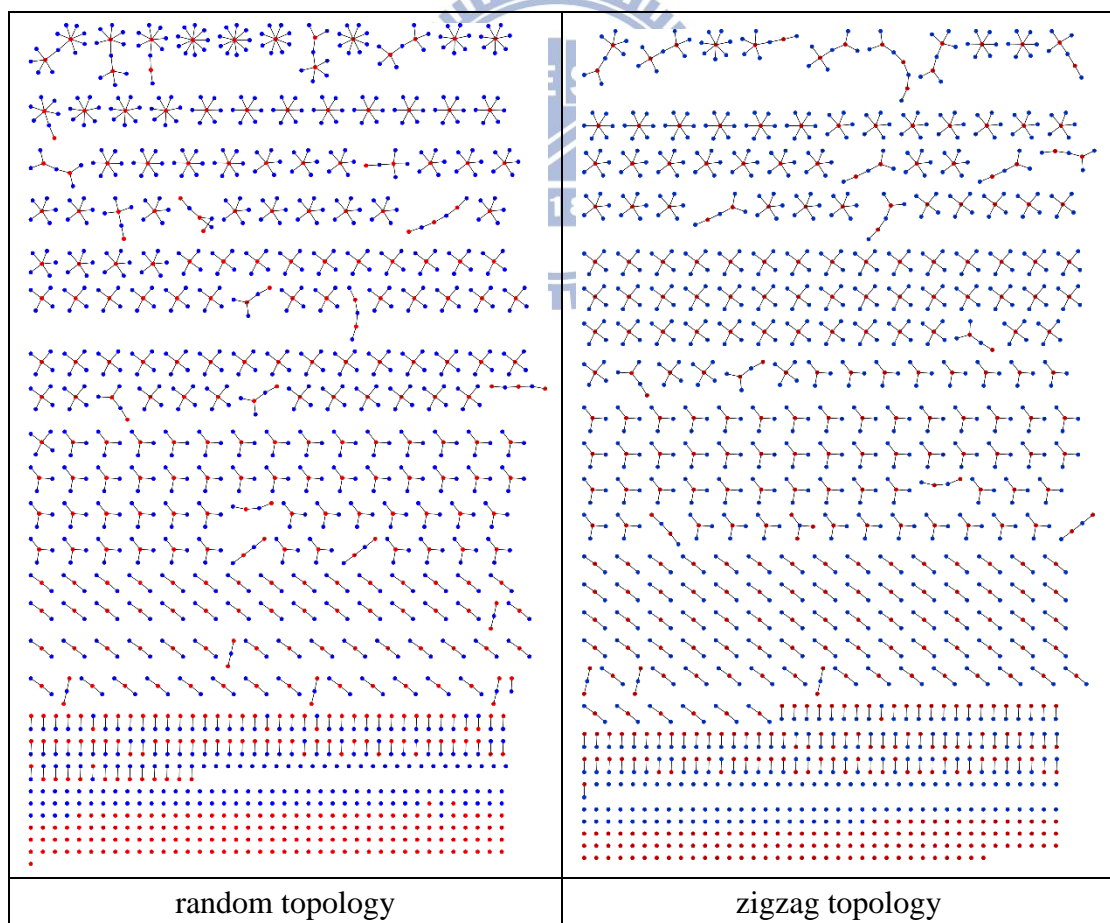
表 8 Markov cluster algorithm example

表 8 為 Markov cluster algorithm 範例圖，經過 MCL 運算後節點 1, 2, 3 分在同一個 cluster，節點 4, 5, 6 分在同一個 cluster。

我們所分析的低密度奇偶檢查碼的 Tanner graph 就是 bipartite graph，且節點數多，適合利用 MCL 將低密度奇偶檢查碼進行分群。本文使用 Cytoscape [19] 一種用於分析複雜網路分析的平台來分析，其中提供許多插件用於分析複雜網路，例如 clusterMaker [20]，clusterMaker 提供了 MCL 這種分群演算法，表 9 為 MCL 參數設定，藉由 MCL 的分析後，不同方法的群結構分布如圖 21，MCL 會將 bipartite graph 切割出許多不同大小的 cluster，圖 24 統計不同大小的 cluster 數量。

MCL Parameter	Value
Expansion parameter	2
Inflation parameter	2
Number of iteration	16

表 9 Markov cluster algorithm parameter setting



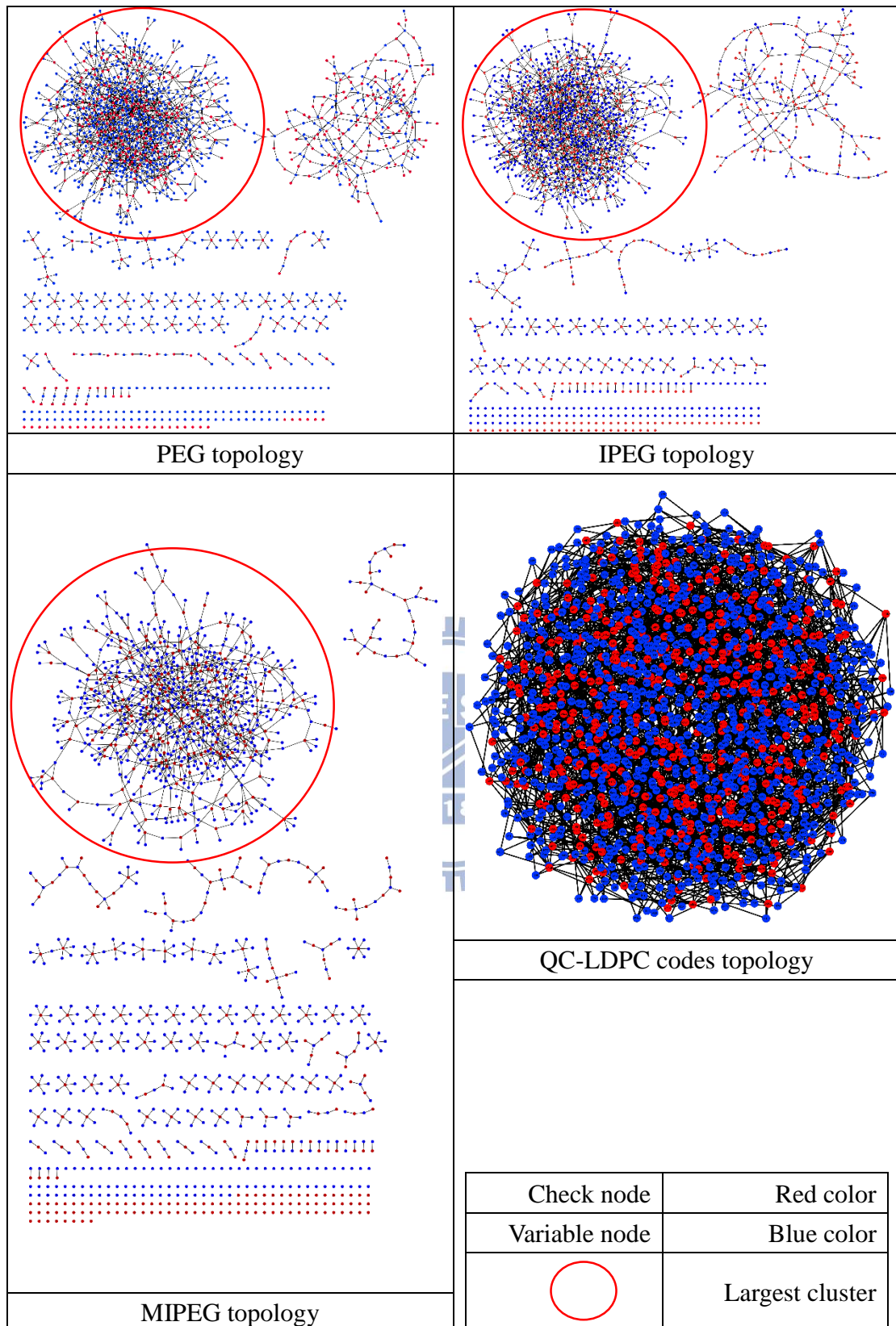


圖 21 Topology by MCL

由圖 21 發現 PEG-based 的碼透過 MCL 的分群後，都存在著 1 個大的 clusters，接下來我們分析 PEG-based 大 cluster 內節點的特性。

- 度分布 (Degree distribution)

由馬可夫分群演算法 (Markov Cluster Algorithm) 的分析我們發現 high degree 的 variable node 不屬於任何 cluster，cluster 內的 variable node 都是 low degree variable node，表 10 為 PEG-based 最大群內 variable node 與 check node 原本的 degree distribution。

number of variable nodes				number of check nodes			
Degree	PEG	IPEG	MIPEG	Degree	PEG	IPEG	MIPEG
2	307	331	306	7	4		
3	201	210	172	8	260	264	223
5	54	65	55	total nodes	264	264	223
total nodes	562	606	533				

表 10 大 cluster 內 node 原始的 degree distribution

MCL 將 high degree variable node 作為 isolated node 不屬於任何 cluster，這是因為 MCL inflation 的步驟，high degree 的分母較大，透過不斷的疊代後，high degree node 上的 edge 走的機率就趨近於 0。

- 網路參數 (Network Parameter)

我們利用網路參數觀察每一個 cluster 內有關 variable node 到 variable node 與最短路徑相關的參數，比較 PEG-based 中不同的方法 PEG, IPEG, MIPEG 的差異，由圖 7 知道 $PEG > IPEG > MIPEG$ ，MIPEG 表現最好，透過以下的分析本文找到了反映效能趨勢的網路參數。

- Average shortest path length

圖 22 為 PEG-based cluster 在不同 degree 下 average shortest path length 的表現，由結果可以發現，不同 degree 下 MIPEG 的 average shortest path length 皆比 IPEG 和 PEG 長，由 Tanner graph 的觀點，希望 cycle length 越長越好，而我們觀察到 variable node 間的 shortest path length 效能較好的碼都比較長，

如果在設計低密度奇偶檢查碼時考量 variable node 到 variable 的 shortest path length，增加 shortest path length 或許也能增加 cycle length 的長度。MCL 切出的大 cluster，所有節點的 average shortest path length 影響 waterfall region 的效能。

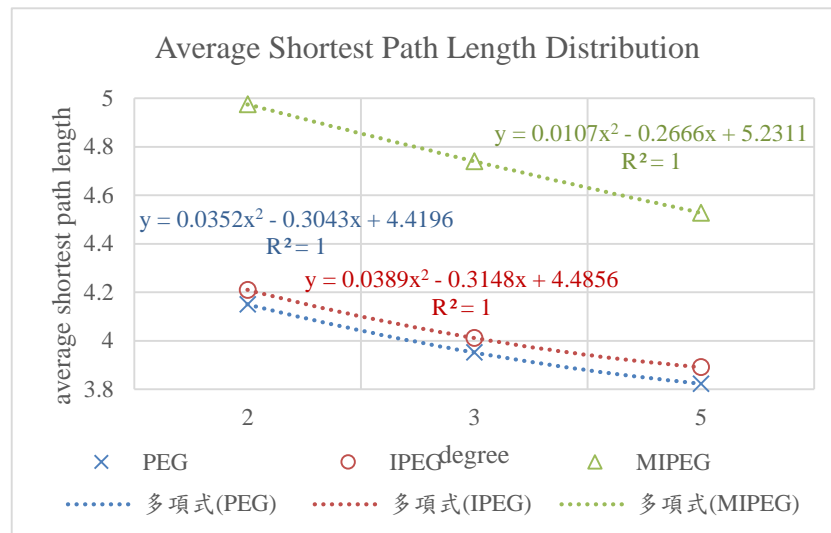


圖 22 Average shortest path length of PEG-based cluster

2. Betweenness centrality

由圖 23 的結果可以發現，MIPEG 的 betweenness centrality 都比其他方法值較大，也就表示 MIPEG 不同 degree 下的 variable node 成為 trapping set 的機率比其他方法低。在設計低密度奇偶檢查碼時考慮節點的 betweenness centrality 或許能夠避免節點成為 trapping set，提升 error floor region 的效能。

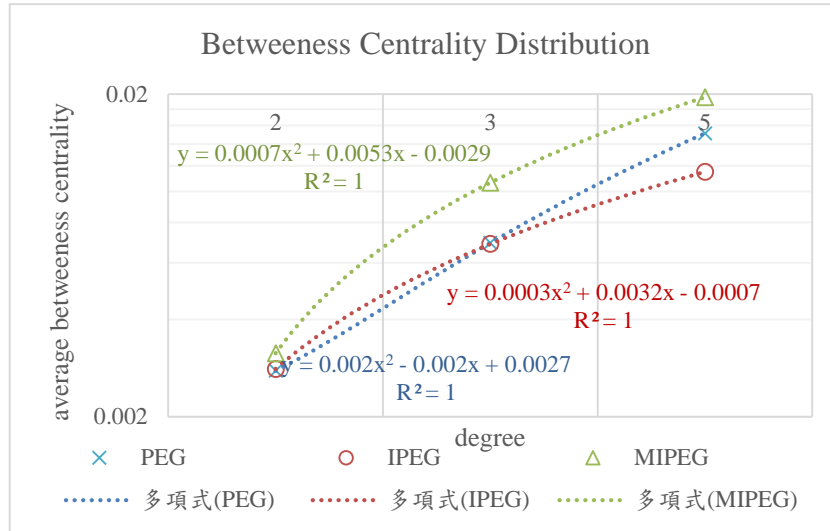


圖 23 Betweenness centrality of PEG-based cluster

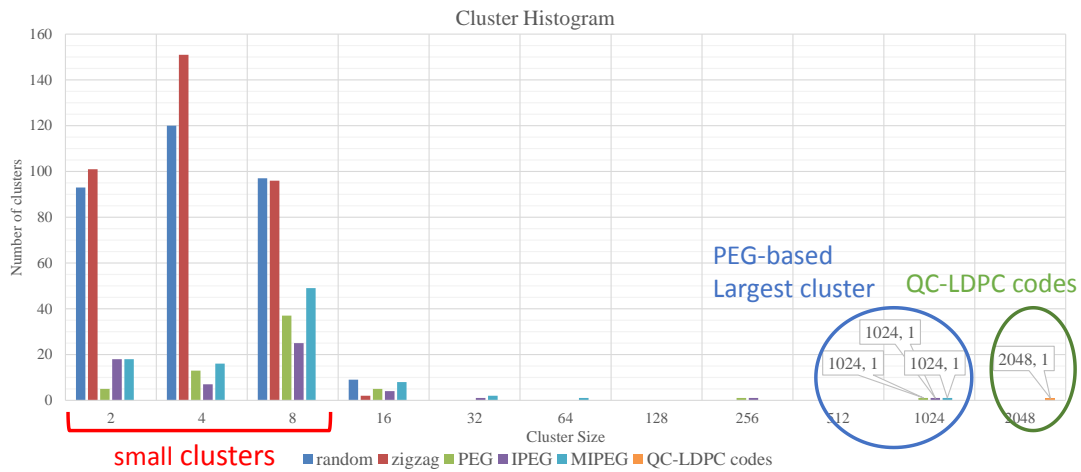


圖 24 Cluster histogram by MCL

由圖 21 與圖 24 發現 QC-LDPC codes 本身是一個大的 cluster，而 PEG-based 的部分擁有一個比較大的 cluster，cluster size 約 1000，但也有一些小的 cluster，而 Random 的部分都是由小的 clusters 組成，表現了 Random, PEG-based 與 QC-LDPC codes 的 error floor region 的效能趨勢如圖 7，小 clusters 數量越多，錯誤率越高，我們猜測是不是這些小的 clusters 造成 trapping set 導致 error floor region 的表現不佳，所以我們討論小 clusters 中 (a, b) trapping set 的分布情形。

由 2.1.1 我們知道 a 值越小發生機率越高， b 值越小錯誤機率越大，本文比較以下 4 種情況下比較 (a, b) trapping set 發生機率：

假設透過 MCL 分析後，cluster 分布如圖 25。

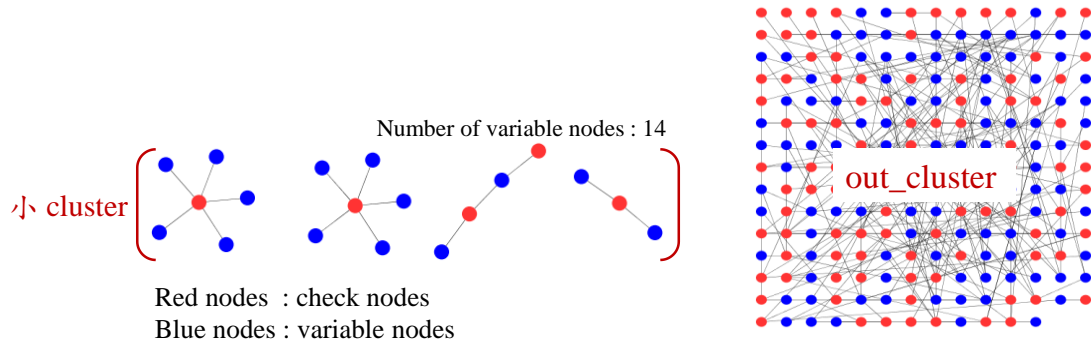


圖 25 MCL cluster 範例圖

1. intra-cluster：同一個小 cluster 內選 (a, b) trapping set 如圖 26。

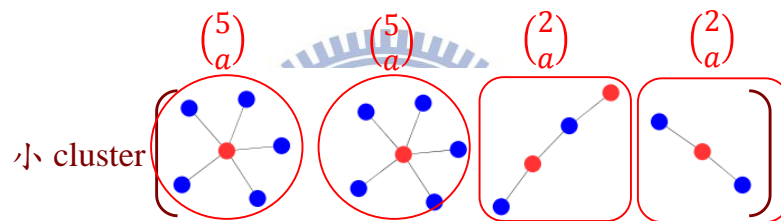


圖 26 MCL intra-cluster 範例圖

假設在每一個小 cluster 內選 2 個 variable node，如圖 27，選到的 b 值分布如表 11：

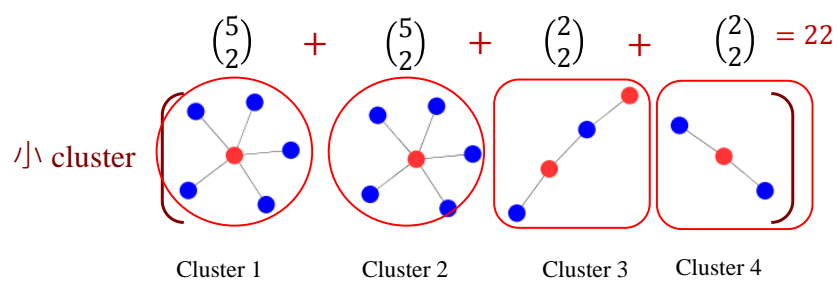


圖 27 MCL intra-cluster $a = 2$ 範例圖

b	Cluster1	Cluster 2	Cluster 3	Cluster 4	Ratio
2	4	5	1	1	$\frac{11}{22} = 0.50$
3	3	3			$\frac{6}{22} = 0.27$
4	3	2			$\frac{5}{22} = 0.23$

表 11 MCL intra-cluster $a = 2, b$ 分布表

綠底的部分表示在 Cluster1 選 2 個 variable node, $(a, b) = (2, 2)$ 的組合數有 4 個；而在 intra-cluster 內任選 2 個 variable node $(a, b) = (2, 2)$ 的組合數就是把 $b = 2$ 那一列相加，總共有 11 個；接下來 normalize 所有小 cluster 內任選 2 個 variable nodes 的組合數，總共是 22 個，所以 $\frac{11}{22}$ 就表示在 intra-cluster 內任選 2 個 variable node $(a, b) = (2, 2)$ 比例 (Ratio) 為 0.50，利用這種計算方式表示 intra-cluster 內任選 a 個 variable node 的比例分布。

2. inter-cluster: 屬於小 cluster 的 node 選 (a, b) trapping set 如圖 28，normalize 的分母為 $\binom{14}{a}$ 。

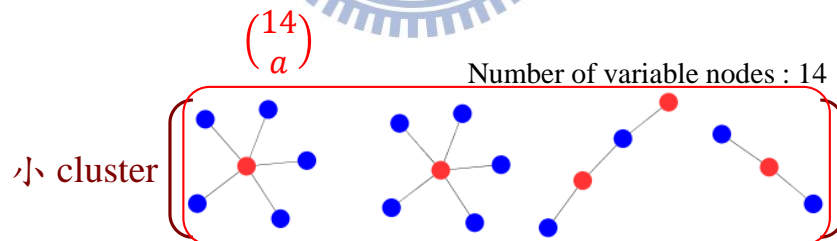


圖 28 MCL inter-cluster 範例圖

3. in-out-cluster : 小 cluster 內選 1 個 node , out-cluster 內任選 $a - 1$ 個 nodes , 如圖 29 , normalize 的分母為 $\binom{14}{1} \times \binom{N}{a-1}$ 。

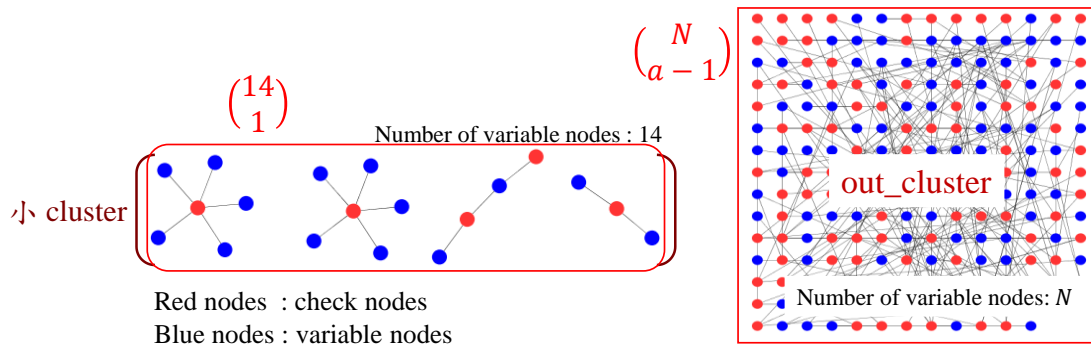


圖 29 MCL in-out-cluster 範例圖

4. out-cluster : out-cluster 內選 (a, b) trapping set 如圖 30 , normalize 的分母為 $\binom{N}{a}$ 。

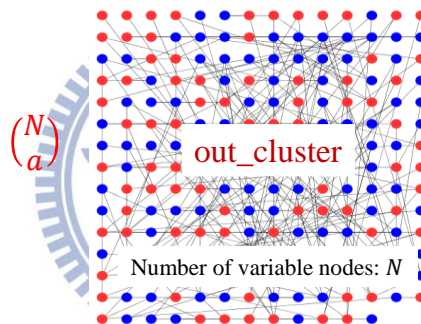


圖 30 MCL out-cluster 範例圖

本文以下比較 PEG, IPEG, MIPEG 上述的 4 種情況挑選 (a, b) trapping set 的 ratio 。

PEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
2	2	6.74E-02	1.91E-03	2.73E-04	1.19E-03
2	3	2.41E-01	6.81E-03	1.89E-03	3.57E-03
2	4	1.12E-01	1.76E-01	2.08E-01	2.43E-01
2	5	2.39E-01	2.75E-01	2.72E-01	2.62E-01
2	6	2.02E-01	1.33E-01	1.18E-01	1.08E-01
2	7	5.59E-02	2.03E-01	1.61E-01	9.02E-02
2	8	6.36E-02	1.42E-01	8.33E-02	3.88E-02
2	9		1.29E-02	1.04E-02	5.26E-03
2	10	1.93E-02	4.94E-02	1.64E-02	5.20E-03
2	15			3.04E-03	7.76E-03

圖 31 PEG MCL trapping ($a = 2$) 比較結果

IPEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
2	2	4.72E-02	1.78E-03	3.24E-04	1.12E-03
2	3	1.63E-01	6.64E-03	1.86E-03	3.55E-03
2	4	1.61E-01	1.72E-01	2.04E-01	2.41E-01
2	5	1.85E-01	2.48E-01	2.60E-01	2.70E-01
2	6	2.23E-01	1.59E-01	1.33E-01	1.00E-01
2	7	4.72E-02	2.15E-01	1.65E-01	9.62E-02
2	8	1.33E-01	1.24E-01	8.30E-02	4.67E-02
2	9		3.35E-02	1.19E-02	4.05E-03
2	10	4.08E-02	4.07E-02	1.76E-02	7.21E-03
2	15			2.93E-03	7.29E-03

圖 32 IPEG MCL trapping ($a = 2$) 比較結果

MIPEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
2	2	5.58E-02	1.33E-03	5.96E-04	1.13E-03
2	3	1.88E-01	4.51E-03	2.14E-03	3.80E-03
2	4	2.65E-01	2.17E-01	2.27E-01	2.33E-01
2	5	1.92E-01	2.73E-01	2.67E-01	2.61E-01
2	6	1.88E-01	1.75E-01	1.29E-01	7.43E-02
2	7	5.48E-02	1.89E-01	1.37E-01	8.22E-02
2	8	4.56E-02	9.26E-02	6.32E-02	4.58E-02
2	9	8.11E-03	2.71E-02	8.96E-03	
2	10	4.06E-03	2.07E-02	1.22E-02	7.30E-03
2	15			4.07E-03	9.28E-03

圖 33 MIPEG MCL trapping ($a = 2$) 比較結果

由圖 31、圖 32、圖 33 可以發現， $a = 2$ intra-cluster 挑選到 (a, b) trapping set 的機率最大。

PEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
3	2	1.22E-03	1.31E-05	7.69E-07	4.38E-06
3	3	6.11E-03	8.07E-05	1.68E-05	3.71E-05
3	4	1.83E-02	2.47E-03	1.04E-03	1.79E-03
3	5	1.14E-01	1.05E-02	5.14E-03	6.20E-03
3	6	1.36E-01	8.06E-02	1.39E-01	1.23E-01
3	7	1.54E-01	1.76E-01	2.50E-01	1.93E-01
3	8	2.75E-01	1.59E-01	1.84E-01	1.32E-01
3	9	1.21E-01	1.73E-01	1.78E-01	9.89E-02
3	10	9.28E-02	1.84E-01	1.39E-01	6.40E-02
3	11	6.59E-02	8.51E-02	5.38E-02	2.31E-02
3	12	3.66E-03	6.98E-02	3.29E-02	1.15E-02
3	13	1.22E-02	4.50E-02	1.37E-02	4.16E-03
3	14		4.05E-03	1.72E-03	5.44E-04
3	15	1.22E-03	1.03E-02	1.55E-03	7.34E-04

圖 34 PEG MCL trapping ($a = 3$) 比較結果

IPEG		intra-cluster	inter-cluster	in-out_cluster	out-cluster
a	b	ratio	ratio	ratio	ratio
3	2	4.15E-03	1.36E-05	1.18E-06	3.86E-06
3	3	8.31E-04	7.21E-05	1.71E-05	3.52E-05
3	4	2.57E-02	2.28E-03	1.04E-03	1.67E-03
3	5	5.15E-02	9.88E-03	4.95E-03	6.11E-03
3	6	1.10E-01	7.86E-02	1.33E-01	1.21E-01
3	7	8.14E-02	1.59E-01	2.40E-01	1.98E-01
3	8	1.67E-01	1.61E-01	1.86E-01	1.30E-01
3	9	1.86E-01	1.90E-01	1.82E-01	1.00E-01
3	10	7.31E-02	1.73E-01	1.42E-01	7.26E-02
3	11	1.89E-01	9.73E-02	5.59E-02	2.49E-02
3	12	1.25E-02	7.61E-02	3.54E-02	1.35E-02
3	13	8.14E-02	3.69E-02	1.53E-02	5.78E-03
3	14		9.62E-03	1.78E-03	4.90E-04
3	15	1.66E-02	7.55E-03	1.91E-03	9.28E-04

圖 35 IPEG MCL trapping ($a = 3$) 比較結果

MIPEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
3	2	7.80E-04	4.82E-06	2.24E-06	3.90E-06
3	3	7.41E-03	4.52E-05	2.48E-05	3.93E-05
3	4	4.02E-02	1.90E-03	1.55E-03	1.66E-03
3	5	9.09E-02	7.45E-03	6.31E-03	6.38E-03
3	6	2.91E-01	1.06E-01	1.57E-01	1.15E-01
3	7	1.56E-01	1.93E-01	2.69E-01	1.88E-01
3	8	2.22E-01	1.84E-01	1.90E-01	1.07E-01
3	9	9.68E-02	1.95E-01	1.66E-01	7.86E-02
3	10	5.74E-02	1.51E-01	1.25E-01	6.47E-02
3	11	2.81E-02	8.04E-02	4.43E-02	1.83E-02
3	12	7.02E-03	5.17E-02	2.72E-02	1.01E-02
3	13	2.73E-03	2.09E-02	1.21E-02	5.60E-03
3	14		5.55E-03	9.38E-04	2.21E-06
3	15		2.83E-03	1.47E-03	1.01E-03

圖 36 MIPEG MCL trapping ($a = 3$) 比較結果

由圖 34、圖 35、圖 36 可以發現 $a = 3$ 時 intra-cluster 挑選到 (a, b) trapping set 的機率也是最大。

透過以上的結果發現在小 cluster 內確實容易挑到 (a, b) trapping set。我們一開始提到 trapping set 與 betweenness centrality 的關係，接下來我們統計小 clusters 內 nodes 的 betweenness centrality 如圖 39 發現小 clusters 內 nodes 的 betweenness centrality 都比較小，而 high degree node 的 betweenness centrality 比其他節點高出許多，圖最右邊紅色表示 high degree node 的 betweenness centrality。

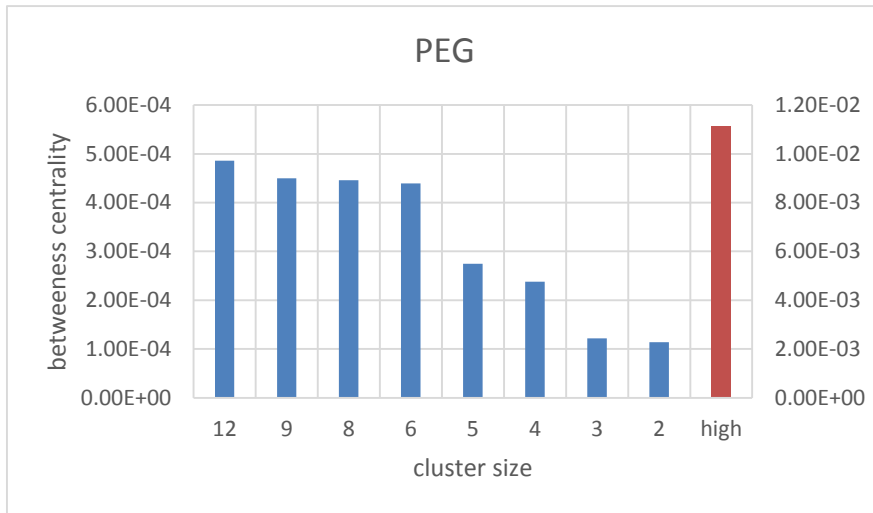


圖 37 The betweenness centrality in clusters of PEG

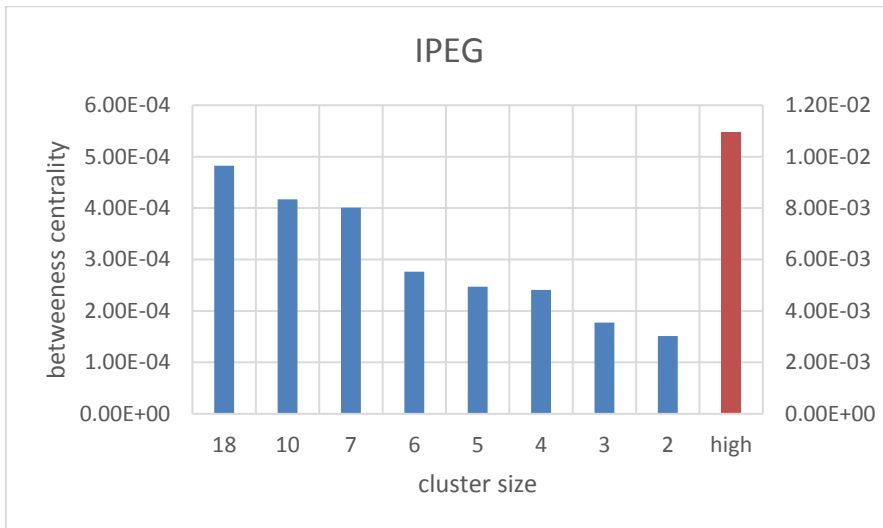


圖 38 The betweenness centrality in clusters of IPEG

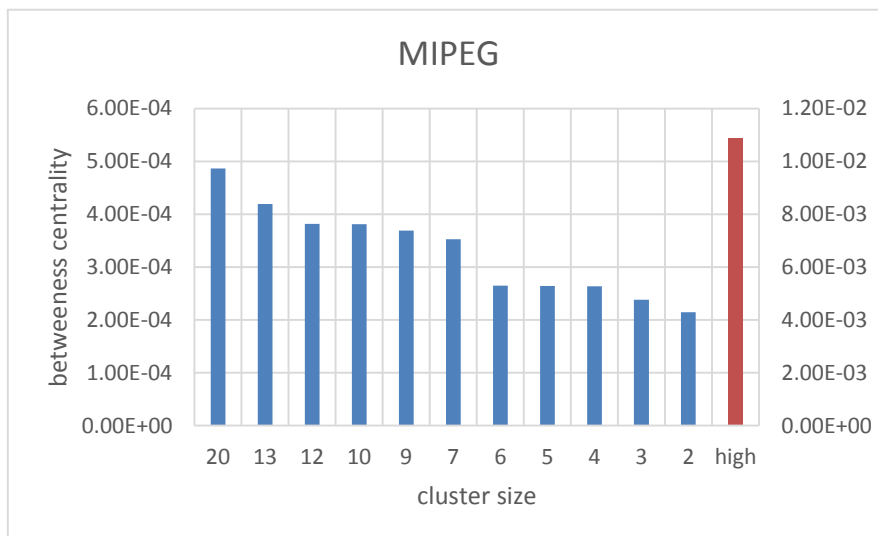


圖 39 The betweenness centrality in clusters of MIPEG

由本節的實驗結果可以發現 LDPC codes 的 Tanner graph 透過 MCL 的分群後，QC-LDPC codes 只有一個 cluster，表示不容易受到其他節點突發狀況而影響，也反映了 QC-LDPC codes 在 error floor region 表現較好的趨勢；而 Random 則是有許多小 clusters，表示 Random 的 LDPC codes 容易受到突發狀況影響，也反映了 Random 表現都比其他方法差的趨勢；而 PEG-based 透過 MCL 的分群後，具有 1 個較大的 cluster 和少許的小 clusters，表示 PEG-based 中的大 cluster 內部救回錯誤節點的機會較高，透過 network parameter 的分析也反映了 MIPEG 參數值 (average shortest path length, betweenness centrality) 都大於其他方法；另外在小 cluster 的部分，透過比較 intra-cluster, inter-cluster, in-out-cluster 與 out-cluster，我們發現 intra-cluster 最容易選到 (a, b) trapping set，也就是說這些小 cluster 確實容易形成 (a, b) trapping set。



4.3. 模組性 (Modularity)

Modularity 是一種網路結構性質，由 M.E.J Newman 提出 [8] [21]，用來衡量網路在分割後的結構品質，表示分割後每一個子結構中連線較緊密，但子結構間只有少數的連線。Modularity 的定義如下：

$$\text{Modularity } Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w) \quad (3)$$

m : $\frac{1}{2} \sum_{vw} A_{vw}$ is the number of edges in graph

$$A_{vw} = \begin{cases} 1 & \text{if vertices } v \text{ and } w \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

k_v : the degree of vertex v

Vertex v belong to community c_v

$\delta(i, j)$ is 1 if $i = j$ and 0 otherwise

先假設每一個 node 屬於不同的 cluster，挑選能夠增加 Modularity 值的 node 加入同一個 cluster，直到無法增加 Modularity 值後完成此次 cluster 的挑選；接下來選擇其他 node 依序再次選擇能夠增加 Modularity 值的 node 加入同一個 cluster，依序不斷挑選。

由 (3) 可以發現 Modularity 是與隨機網路的連線情況比較，連線比隨機網路的連線越緊密則表示分割的結果越好，每一個子結構內的連線緊密，表示 node 彼此間傳遞訊息需要經過的 path 較短，在 LDPC codes decoding 的情況下，越近的越容易將錯誤更正，所以發生錯誤時 cluster 內的 node 能夠互相幫忙更正，cluster 越大，有越多 nodes 能夠幫忙，cluster 越小則表示幫忙更正的 nodes 數少，所以我們利用 Modularity 分割 variable node 的 unipartite graph。

本文使用 Cytoscape [19] 的插件 clusterMaker [20] 提供的 Modularity 分群演算法分析 variable node 的 unipartite graph，以下為分析結果圖：

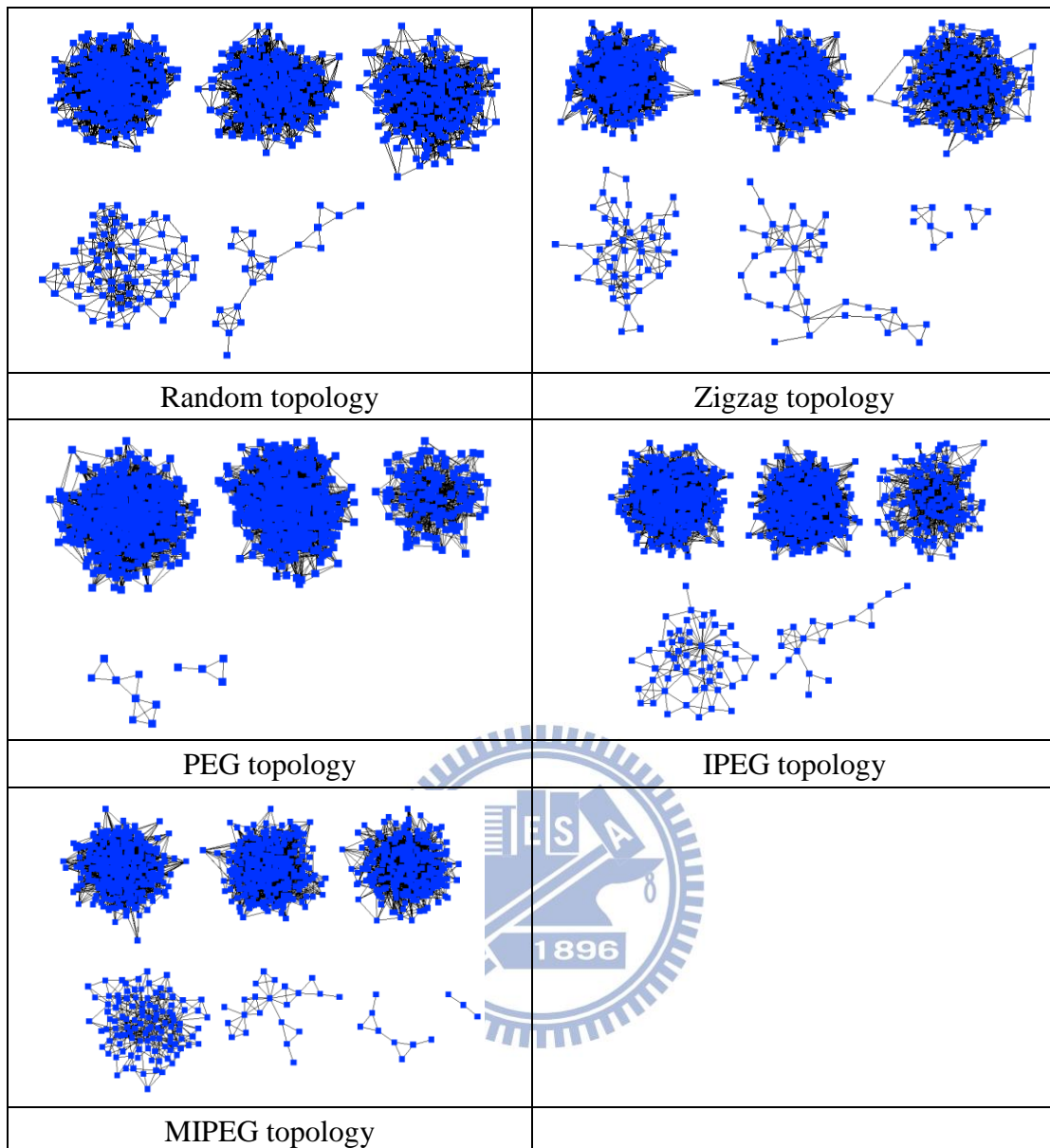


圖 40 Topology by Modularity

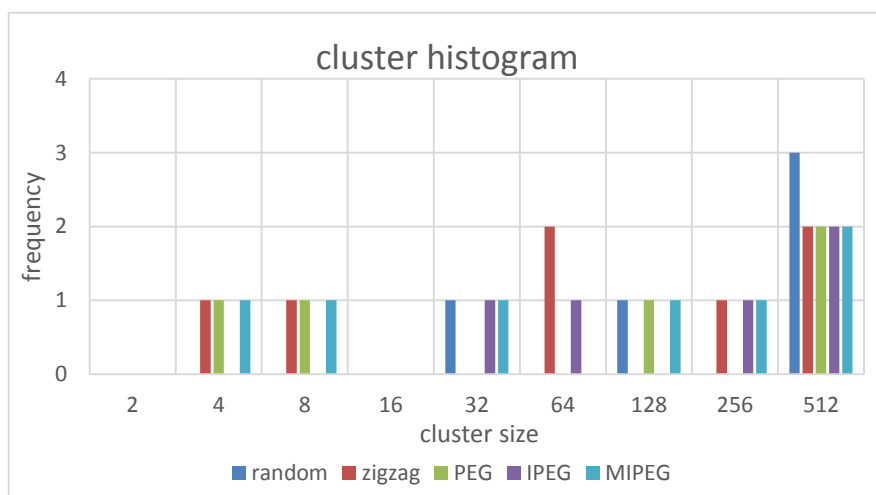


圖 41 Cluster histogram by Modularity

由圖 40、圖 41 無法從 topology 與 cluster histogram 的觀點觀察出不同方法的差異，接下來我們觀察每一個 cluster 內 network parameter (average shortest path length, betweenness centrality) 的特性。

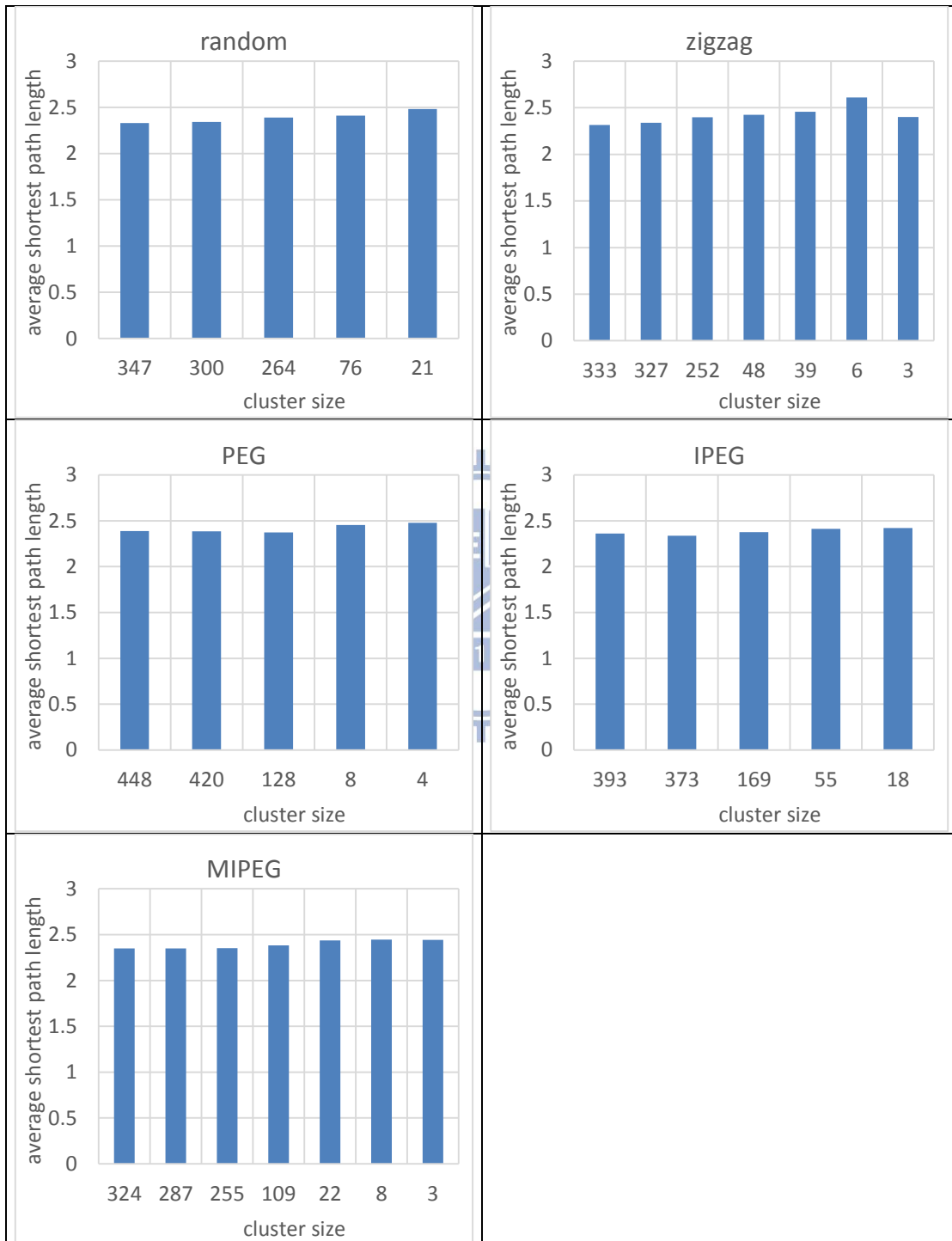


圖 42 Average shortest path length of each cluster

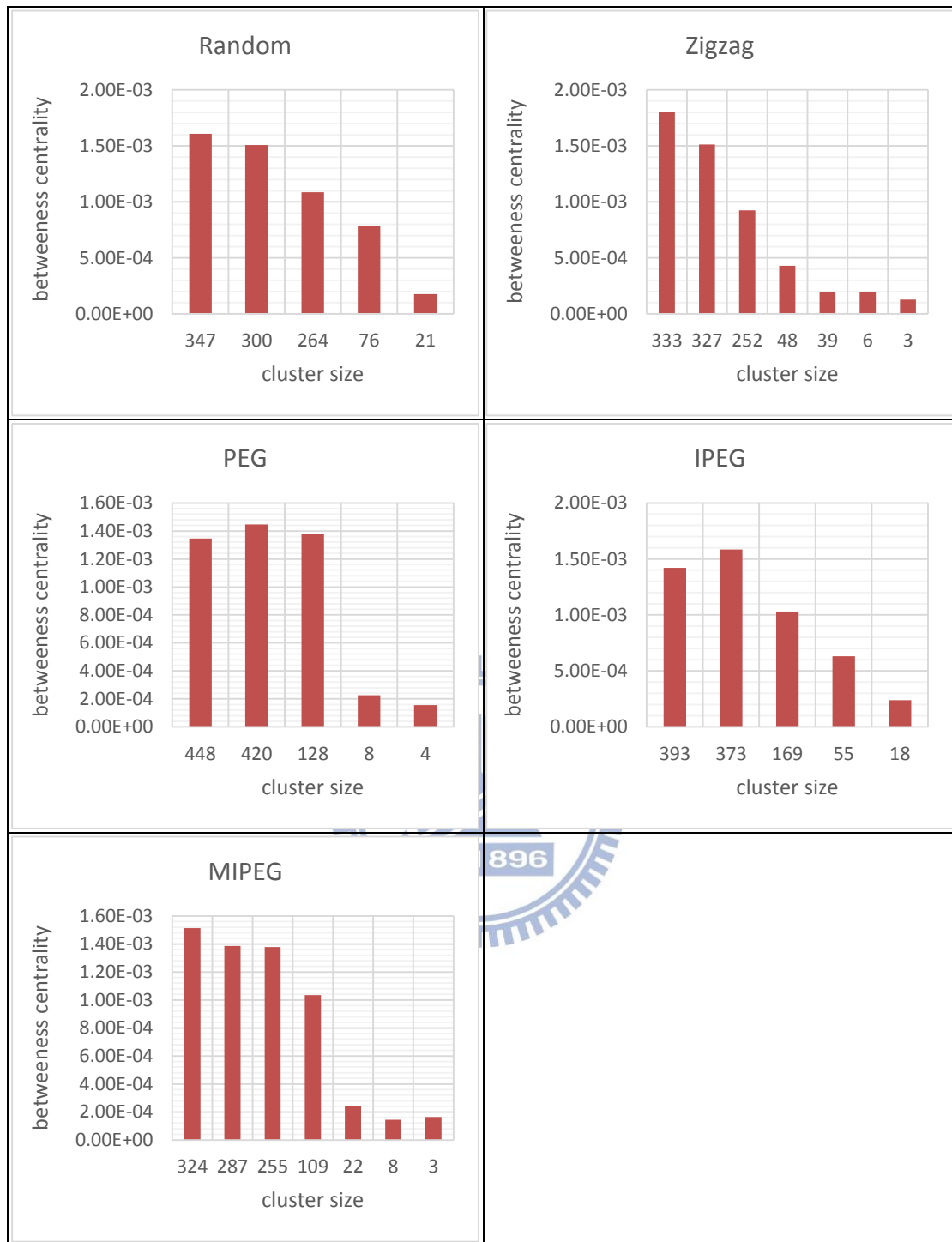


圖 43 Betweenness centrality of each cluster

如圖 42 所示由 average shortest path length 無法看出不同 cluster size 的差異，但是透過圖 43 可以發現 cluster size 越小的 betweenness centrality 越小，我們假設這些小 cluster 的 node 容易形成 trapping set，本文比較以下 4 種情況下比較 (a, b) trapping set 發生機率：

假設透過 Modularity 分析後，cluster 分布如圖 44。

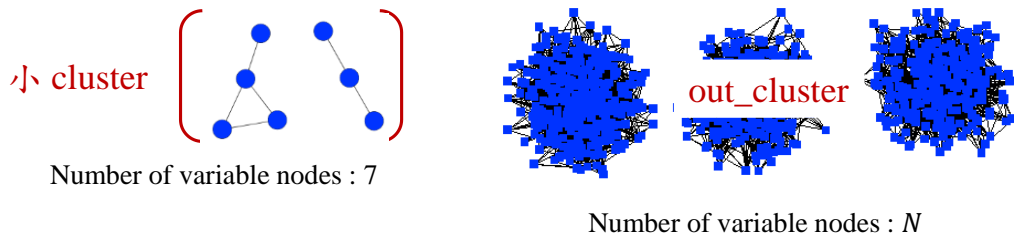


圖 44 Modularity cluster 範例圖

1. intra-cluster: 同一個小 cluster 內選 (a, b) trapping set 如圖 45，normalize 的分母為 $\binom{4}{a} \times \binom{3}{a}$ 。

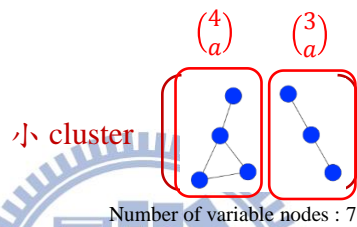


圖 45 Modularity intra-cluster 範例圖

2. inter-cluster: 小 cluster 選 (a, b) trapping set 如圖 46，normalize 的分母為 $\binom{7}{a}$ 。

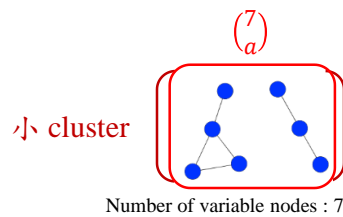


圖 46 Modularity inter-cluster 範例圖

3. in-out-cluster : 小 cluster 內選 1 個 node , out-cluster 內任選 $a - 1$ 個 nodes , 如圖 47 , normalize 的分母為 $\binom{7}{1} \times \binom{N}{a-1}$ 。

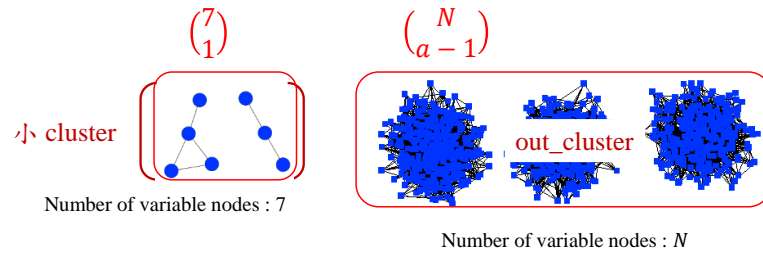


圖 47 Modularity in-out-cluster 範例圖

4. out-cluster : out-cluster 內選 (a, b) trapping set 如圖 48 , normalize 的分母為 $\binom{N}{a}$ 。

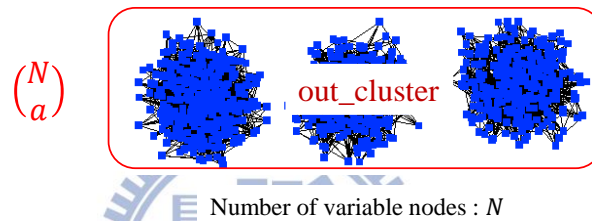


圖 48 Modularity out-cluster 範例圖

以下比較 PEG, IPEG, MIPEG 上述的 4 種情況選 (a, b) trapping set 的 ratio 。

PEG	intra_cluster	inter_cluster	in-out_cluster	out_cluster
a b	ratio	ratio	ratio	ratio
2 2	1.47E-01	7.58E-02	5.02E-04	8.98E-04
2 3	1.76E-01	9.09E-02	2.34E-03	3.15E-03
2 4	3.82E-01	4.24E-01	3.17E-01	2.25E-01
2 5	2.06E-01	2.73E-01	3.05E-01	2.65E-01
2 6	2.94E-02	9.09E-02	1.33E-01	1.12E-01
2 7	5.88E-02	4.55E-02	1.04E-01	1.22E-01
2 8			2.98E-02	6.14E-02
2 9			8.87E-03	7.50E-03
2 10				1.20E-02
2 15			4.02E-03	5.66E-03

圖 49 PEG Modularity trapping set ($a = 2$) 比較結果

IPEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
2	2	1.16E-02	7.99E-03	6.15E-04	9.00E-04
2	3	3.48E-02	2.28E-02	2.65E-03	3.09E-03
2	4	3.26E-01	3.25E-01	2.70E-01	2.20E-01
2	5	3.68E-01	3.82E-01	3.18E-01	2.57E-01
2	6	1.06E-01	1.13E-01	1.17E-01	1.12E-01
2	7	5.62E-02	5.86E-02	1.03E-01	1.25E-01
2	8	3.05E-02	3.39E-02	5.26E-02	6.22E-02
2	9	0.00E+00	0.00E+00	2.02E-03	8.46E-03
2	10	1.83E-03	2.28E-03	6.18E-03	1.26E-02
2	15	1.47E-02	1.03E-02	3.72E-03	5.95E-03

圖 50 IPEG Modularity trapping set ($a = 2$) 比較結果

MIPEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
2	2	4.20E-02	2.08E-02	7.15E-04	8.89E-04
2	3	8.40E-02	4.17E-02	2.49E-03	3.12E-03
2	4	4.27E-01	4.62E-01	3.27E-01	2.20E-01
2	5	2.94E-01	3.26E-01	3.09E-01	2.63E-01
2	6	4.96E-02	5.68E-02	9.39E-02	1.14E-01
2	7	7.63E-02	6.82E-02	1.13E-01	1.23E-01
2	8	2.29E-02	2.27E-02	4.30E-02	6.21E-02
2	9			2.24E-03	7.92E-03
2	10	3.82E-03	1.89E-03	6.56E-03	1.20E-02
2	15			4.23E-03	5.72E-03

圖 51 MIPEG Modularity trapping set ($a = 2$) 比較結果

由圖 49、圖 50、圖 51 可以發現 $a = 2$ 時 intra-cluster 挑選到 (a, b) trapping set 的機率最大。

PEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
3	2	3.33E-02	9.09E-03	1.01E-06	2.69E-06
3	3	5.00E-02	1.36E-02	1.61E-05	2.69E-05
3	4	2.00E-01	1.41E-01	1.09E-03	1.30E-03
3	5	3.00E-01	2.14E-01	4.83E-03	5.22E-03
3	6	2.00E-01	2.86E-01	1.53E-01	1.09E-01
3	7	2.00E-01	2.32E-01	2.33E-01	1.89E-01
3	8	1.67E-02	5.91E-02	1.60E-01	1.36E-01
3	9		3.64E-02	1.30E-01	1.22E-01
3	10		9.09E-03	7.92E-02	9.38E-02
3	11			2.96E-02	3.62E-02
3	12			1.44E-02	2.26E-02

圖 52 PEG Modularity trapping set ($a = 3$) 比較結果

IPEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
3	2	7.39E-05	8.04E-05	1.41E-06	2.74E-06
3	3	1.63E-03	9.49E-04	1.95E-05	2.64E-05
3	4	1.90E-02	1.40E-02	1.11E-03	1.29E-03
3	5	6.29E-02	4.60E-02	4.90E-03	5.08E-03
3	6	2.10E-01	2.02E-01	1.29E-01	1.06E-01
3	7	2.96E-01	3.14E-01	2.23E-01	1.81E-01
3	8	1.67E-01	1.87E-01	1.53E-01	1.33E-01
3	9	7.55E-02	8.21E-02	1.21E-01	1.23E-01
3	10	4.67E-02	5.26E-02	9.06E-02	9.44E-02
3	11	1.22E-02	1.50E-02	3.15E-02	3.74E-02
3	12	2.88E-03	3.38E-03	1.65E-02	2.39E-02

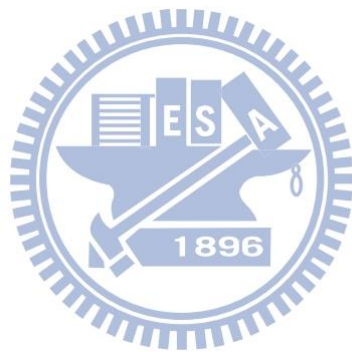
圖 53 IPEG Modularity trapping set ($a = 3$) 比較結果

MIPEG		intra_cluster	inter_cluster	in-out_cluster	out_cluster
a	b	ratio	ratio	ratio	ratio
3	2	1.88E-03	5.50E-04	1.72E-06	2.51E-07
3	3	1.19E-02	3.48E-03	1.98E-05	2.63E-06
3	4	6.76E-02	4.25E-02	1.31E-03	1.27E-04
3	5	1.54E-01	9.86E-02	5.11E-03	5.12E-04
3	6	2.94E-01	3.20E-01	1.56E-01	1.06E-02
3	7	2.67E-01	3.11E-01	2.36E-01	1.85E-02
3	8	9.46E-02	1.12E-01	1.43E-01	1.36E-02
3	9	6.01E-02	6.52E-02	1.25E-01	1.22E-02
3	10	3.69E-02	3.68E-02	8.70E-02	9.42E-03
3	11	7.51E-03	6.60E-03	2.69E-02	3.71E-03
3	12	3.76E-03	2.57E-03	1.71E-02	2.28E-03

圖 54 MIPEG Modularity trapping set ($a = 3$) 比較結果

由圖 52、圖 53、圖 54 可以發現 $a = 3$ 時 intra-cluster 挑選到 (a, b) trapping set 的機率最大。

由本節 Modularity 觀察 variable node unipartite graph 的結果可以發現，透過 Modularity 的分群後，小 cluster 內的 variable node 的 betweenness centrality 較小，且透過比較 intra-cluster, inter-cluster, in-out cluster 與 out-cluster 可以發現 intra-cluster 選到 (a, b) trapping set 的機率最高，也證實了小 cluster 內確實容易形成 (a, b) trapping set。



五. 結論

5.1. 研究成果

本文的研究問題是找出與效能有關的結構因素，透過實驗結果本文以下歸納與效能相關的結構特性：

Markov cluster algorithm 與 Modularity 分群後小 clusters 數量越多，error floor region 的錯誤率越大，所以小 cluster 容易形成 (a, b) trapping set，而且小 cluster 內的 betweenness centrality 都比其他 node 小，所以 betweenness centrality 越小 error floor region 的效能越差。

在 waterfall region 表現好的碼，透過 Markov cluster algorithm 分群後具有 1 個大 cluster，大 cluster 結構中的 average shortest path length 較大。

5.2. 未來展望

不同的分類方式能夠聚集的節點特性不同，本文嘗試了 Markov Chain Cluster Algorithm 與 Modularity 這兩種分群演算法後，發現了 clustering algorithm 與 LDPC codes 間的關聯性。

還有許多其他 clustering algorithm 用於不同領域與不同網路結構的分析，哪一種 clustering algorithm 最適合 LDPC codes，還是一個值得探討的問題。



參考文獻

- [1] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, pp. 386-398, Jan 2005.
- [2] H. Xiao and A. H. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Comm. Letts.*, pp. 715-717, Dec 2004.
- [3] S. M. v. Dongen, "Network analysis," [Online]. Available: http://micans.org/mcl/index.html?sec_mcledge.
- [4] Francis C.M. LAU, Chi K. TSE and Zhiliang ZHU, "Future Design of Channel Codes: A Complex Network Perspective," *Fourth International Workshop on Chaos-Fractals Theories and Applications*, pp. 156-160, 2011.
- [5] X. Zhen, F.C.M Lau, C.K. Tse, Y. He, S. Hau, "Application of complex-network theories to the design of short-length low-density-parity check codes," *IET Communication*, pp. 1569-1577, 2009.
- [6] ALBERT-U\SZLOBARABASIANDERICBONABEAU, "Scale-Free Networks," *SCIENTIFIC AMERICAN*, pp. 50-59, May 2003.
- [7] S. v. Dongen, "Graph clustering by flow simulation," University of Utrecht, May 2000. [Online]. Available: <http://www.library.uu.nl/digiarchief/dip/diss/1895620/inhoud.htm>.
- [8] Aaron Clauset, M. E. J. Newman, and Cristopher Moore, "Finding community structure in very large networks," pp. 1-6, 30 Aug 2004.
- [9] R. G. Gallager, in *Low-Density Parity-Check Codes*, 1963.
- [10] "Low-density parity-check code," [Online].
- [11] 趙啟超, "錯誤更正碼簡介," [Online]. Available: http://w3.math.sinica.edu.tw/math_media/d184/18404.pdf.
- [12] "Publications," [Online]. Available: <http://www2.engr.arizona.edu/~vasiclab/Projects/CodingTheory/ErrorFloorPublications.html>.
- [13] J.-D. Du, "Performance Improvement of PEG-based Construction for Finite-Length LDPC Codes," 2012.
- [14] F.C.M. Lau W.M. Tam C.K. Tse, "Increasing the local girth of irregular low-density parity-check codes based on degree-spectrum analysis," *IET Communications*, p. 1506-1511, May 2011.
- [15] J. Reichardt. [Online]. Available: <http://intersci.ss.uci.edu/wiki/pub/JR/clustertoolmanual.pdf>.
- [16] J. Reichardt, "Jörg Reichardt," [Online]. Available: <http://theorie.physik.uni-wuerzburg.de/~reichardt/index.html>.
- [17] J. Reichardt, and D.R. White, "Role models for complex network," *THE EUROPEAN PHYSICAL JOURNAL B*, pp. 217-224.
- [18] K. Macropol, "Clustering on Graphs: The Markov Cluster Algorithm (MCL)," 2009. [Online]. Available: http://www.cs.ucsb.edu/~xyan/classes/CS595D-2009winter/MCL_Presentation

2.pdf.

- [19] "Cytoscape," [Online]. Available: <http://www.cytoscape.org/>.
- [20] "clusterMaker: Creating and Visualizing Cytoscape Clusters," [Online]. Available: <http://www.cgl.ucsf.edu/cytoscape/cluster/clusterMaker.html>.
- [21] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E* 69, pp. 1-16, August 2003.
- [22] F. R. Kschischang, B. J. Frey and H. A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE TRANSACTIONS ON INFORMATION THEORY*, VOL. 47, NO. 2, p. 498–519, FEBRUARY 2001.
- [23] Rodolfo Baggio, Noel Scott, Chris Cooper, *Network Science – A review focused on tourism*.
- [24] J. D. a. S. Horvath, "Understanding network concepts in modules," *BMC Systems Biology* 2007, 1:24 doi:10.1186/1752-0509-1-24, pp. 1-2, 4 June 2007.
- [25] Yassen Assenov (Max Planck Institute for Informatics), Mario Albrecht (Max Planck Institute for Informatics), Mike Smoot (University of California, San Diego), "NetworkAnalyzer Online Help," Dec 2010. [Online]. Available: <http://med.bioinf.mpi-inf.mpg.de/netanalyzer/help/2.7/>.

