นิยามที่จะใช้ในวิทยานิพนธ์ของข้าพเจ้า

Graph: ยังไม่แน่ใจว่าจะให้เป็นกราฟที่เป็น highly connected หรือ ไม่ เพราะอยากให้support กราฟได้ทุกชนิด และ ถ้าเป็น highly connected อาจคล้ายกับงาน [GCE] มากเกินไป

Candidate Overlapped Zone

Candidate Overlapped Zone คือ Sub-graph ที่ประกอบด้วยโหนดมากกว่า 3 โหนด [GCE] โดยทุกโหนดเชื่อมต่อถึงกันหมด (Fully Connected Sub-graph) และมีค่าความต่างของความหนาแน่น (Difference Density) ระหว่างภายใน (Intra Cluster Density) กับ ภายนอก (Inter Cluster Density) สูง

Greedy Overlapped Zone

นำ Candidate Overlapped Zone มาเพิ่มจำนวนโหนดทีละ 1 โหนด [GCE] พิจารณาความต่างของความหนาแน่นภายในกับภายนอกโดยต้องมากขึ้นหรือเท่าเดิม เลือก Candidate Overlapped Zone ที่สามารถเพิ่มจำนวนได้มากที่สุด (Remark: การเพิ่มโหนดในขั้นตอนนี้เพื่อกันปัญหา near-duplicates ถ้ากราฟเป็น Highly ขั้นตอนการเพิ่มโหนดทีจะไม่เกิดขึ้น)

Cluster

Cluster คือ กลุ่มที่เกิดจากการเพิ่มจำนวนโหนดใน Greedy Overlapped Zone แล้วทำให้ค่า ความต่างของความหนานแน่นภายในกับภายนอกของกลุ่มมีค่าน้อยกว่าค่าเดิม แต่ต้องมากกว่า “ค่าเฉลี่ยการเชื่อมต่อของกราฟ” [3]

“ค่าเฉลี่ยการเชื่อมต่อของกราฟ” คือ อัตราส่วนระหว่าง Edge ของกราฟกับการเชื่อมต่อสูงสุดที่เป็นไปได้ (n (n-1) / 2)

สูตรต่างๆ (ทุกสูตรมีเอกสารอ้างอิง)

- **Complete graphs** [1] = n (n-1) / 2

- **Highly Connected graphs** [2]
  1. edge ใน graph > n / 2
  2. edge ขั้นต่ำของแต่ละโหนด >= (n/2) + 1

3. เส้นผ่านศูนย์กลางของ Highly connect graph ไม่เกิน 2 << ไม่ค่อยเห็นด้วยเท่าไหร่ เพราะในกรณีที่เป็น 6 โหนด ก็ไม่ใช่แล้ว แต่ไม่ทราบว่าควรเขียน แล้ว อภิปราย หรือ ไม่ควรกล่าวถึงในงานแบบไหนจะดีกว่ากัน
4. มีความคล้ายกันใน Highly Connected graph 0.5 ถึง เกือบจะ Completed graph

- **Inter-Intra Clustering** [3]

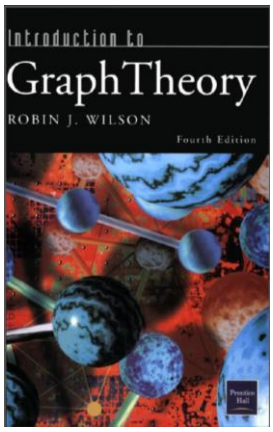
inter-cluster density = number of edges running from the vertices of C to the rest of the graph / $n_c(n - n_c)$

intra-cluster density = number of internal edges of C / $n_c(n_c - 1) / 2$

- **Difference Density** [3] = Internal - External

อ้างอิง

1. Complete graphs



Complete graphs
A simple graph in which each pair of distinct vertices are adjacent is a **complete graph**. We denote the complete graph on n vertices by K_n; K_4 and K_5 are shown in Fig. 3.2. You should check that K_n has $n(n-1)/2$ edges.

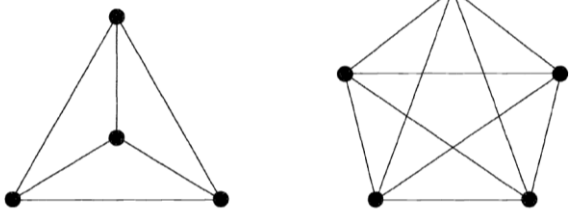


Fig. 3.2

2. Highly Connected graph

Finding Highly Connected Subgraphs*

Falk Hüffner, Christian Komusiewicz, and Manuel Sorge

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin,
{falk.hueffner,christian.komusiewicz,manuel.sorge}@tu-berlin.de

1 Introduction

A popular method of analyzing complex networks is to identify *clusters* or *communities*, that is, subgraphs that have many interactions within themselves and fewer with the rest of the graph (e. g. [18, 19]). Hartuv and Shamir [9] proposed a prominent clustering algorithm producing *highly connected* clusters, formalized as follows: the *edge connectivity* $\lambda(G)$ of a graph G is the minimum number of edges whose deletion results in a disconnected graph, and a graph G with n vertices is called *highly connected* if $\lambda(G) > n/2$. An equivalent characterization is that a graph is highly connected if each vertex has degree at least $\lfloor n/2 \rfloor + 1$ [3]. Moreover, highly connected graphs have diameter at most two [9].

Related work. The algorithm by Hartuv and Shamir [9] partitions a graph heuristically into highly connected components; another algorithm tries to explicitly minimize the number of edges that need to be deleted for this [11]. Highly connected graphs can be seen as *clique relaxation* [18], that is, a graph class that has many properties similar to cliques, without being as restrictive. Highly connected graphs are very similar to *0.5-quasi-complete graphs* [17], that is, graphs where every vertex has degree at least $(n-1)/2$. These graphs are also referred to as *(degree-based) 0.5-quasi-cliques* [15]. Recently, also the task of finding subgraphs with high *vertex* connectivity has been examined [20].

3. Inter-Intra Clustering

Community detection in graphs

Santo Fortunato*

Complex Networks and Systems Lagrange Laboratory, ISI Foundation, Viale S. Severo 65, 10133, Torino, I-ITALY.

We define the *intra-cluster density* $\delta_{int}(\mathcal{C})$ of the subgraph $\mathcal{C}$ as the ratio between the number of internal edges of $\mathcal{C}$ and the number of all possible internal edges, i. e.

$$\delta_{int}(\mathcal{C}) = \frac{\#\text{ internal edges of } \mathcal{C}}{n_c(n_c-1)/2}. \qquad (1)$$

Similarly, the *inter-cluster density* $\delta_{ext}(\mathcal{C})$ is the ratio between the number of edges running from the vertices of $\mathcal{C}$ to the rest of the graph and the maximum number of inter-cluster edges possible, i. e.

$$\delta_{ext}(\mathcal{C}) = \frac{\#\text{ inter-cluster edges of } \mathcal{C}}{n_c(n-n_c)}. \qquad (2)$$

For $\mathcal{C}$ to be a community, we expect $\delta_{int}(\mathcal{C})$ to be appreciably larger than the average link density $\delta(\mathcal{G})$ of $\mathcal{G}$, which is given by the ratio between the number of edges of $\mathcal{G}$ and the maximum number of possible edges $n(n-1)/2$. On the other hand, $\delta_{ext}(\mathcal{C})$ has to be much smaller than $\delta(\mathcal{G})$. Searching for the best tradeoff between a large $\delta_{int}(\mathcal{C})$ and a small $\delta_{ext}(\mathcal{C})$ is implicitly or explicitly the goal of most clustering algorithms. A simple way to do that is, e. g., maximizing the sum of the differences $\delta_{int}(\mathcal{C}) - \delta_{ext}(\mathcal{C})$ over all clusters of the partition[3] (Mancoridis *et al.*, 1998).

A required property of a community is *connectedness*. We expect that for $\mathcal{C}$ to be a community there must be a path between each pair of its vertices, running only through vertices of $\mathcal{C}$. This feature simplifies the task of community detection on disconnected graphs, as in this case one just analyzes each connected component separately, unless special constraints are imposed on the resulting clusters.

4. GCE: Greedy Clique Expansion

Detecting Highly Overlapping Community Structure by Greedy Clique Expansion

Conrad Lee, Fergal Reid, Aaron McDaid, Neil Hurley

University College Dublin
Clique Research Cluster
Dublin 4, Ireland
{conradlee,fergal.reid,aaronmcdaid}@gmail.com, neil.hurley@ucd.ie

2. METHOD

Given a graph G with vertices V and edges E, GCE works by first detecting a set of seeds in G, then expanding these seeds in series by greedily maximizing a local community fitness function, and then finally accepting only those communities that are not near-duplicates of communities that have already been accepted.

Overview of GCE. Now that we have covered the requisite concepts of community fitness, expanding a seed, choosing seeds, and near-duplicate seeds, we can outline the GCE algorithm. Given a graph G, a minimum clique size k, a minimum community distance ϵ, and a scaling parameter α, our algorithm:

1. Finds seeds by detecting all maximal cliques in G with at least k nodes.
2. Creates a candidate community C' by choosing the largest unexpanded seed and greedily expanding it with a community fitness function F until the addition of any node would lower fitness.
3. If C' is within ϵ of any already accepted community C, then C and C' are near-duplicates, so discard C'. Otherwise, if no near-duplicates are found, accept C'.
4. Continues to loop back to step 2 until no seeds remain.

We note that although GCE allows the user to specify the values of three parameters, two of these—k and ϵ—allow for versatile default values. The value of k should usually be 4; if one is interested in very small communities (as in the case of the protein complexes presented in section 5), then k should be set to 3. We find 0.25 to be a good default value for ϵ—if one finds too many near-duplicate communities in the output, then ϵ should be increased.